МВССО УССР

Харьковский институт радиоэлектроники

Кафедра теоретической кибернетики

Учебно-методическое пособие по математическому обеспечению машин серии „АСВТ" и „РЯД"

Харьков
1973

МВССО УССР

Харьковский институт радиоэлектроники
Кафедра теоретической кибернетики

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ ПО МАТЕМАТИЧЕСКОМУ ОБЕСПЕЧЕНИЮ МАШИН СЕРИИ АСВТ И "РЯД"

Харьков
1973 год

Под редакцией

д.т.н. проф. Кузьмина И.В.

Эта книга является учебно-методическим пособием при изучении курсов:

"Проектирование подсистем и звеньев АСУ"

"Моделирование систем"

"Теория вычислительных комплексов"

"Теория систем"

"Математическое обеспечение АСУ и ЦВМ"

"Теория информации и информационной техники"

ВВЕДЕНИЕ

В соответствии с Программой КПСС и решениями XXIУ съезда КПСС в нашей стране уделяется большое внимание развитию электронно-вычислительной техники и ее широкому применению в производстве, научно-исследовательских и проектно-конструкторских работах, плановых расчетах, в сфере учета, статистики и управления.

Решения XXIУ съезда КПСС предусматривает широкое развитие работ по созданию и внедрению автоматизированных систем управления, основанных на использовании экономико-математических методов и электронно-вычислительной, организационной техники.

Широкое развертывание работ по разработке, внедрению и эксплуатации АСУ требует высококвалифицированных специалистов. Современный инженер по АСУ должен в совершенстве владеть методами обработки информации с помощью ЭВМ.

Данная книга является учебно-методическим пособием для студентов специальностей "АСУ" и "ПМ" по математическому обеспечению машин серии АСВТ и ЕС ЭВМ. В книге систематически изложены основы программирования и системного математического обеспечения ЭВМ АСВТ и ЕС ЭВМ.

Разделы I и II содержат детальное описание системных макрокоманд. В разделе Ш описан язык управления заданием. Раздел IУ посвящен описанию языка оператора системы. В разделе У дано подробное описание универсальной системы команд АСВТ и ЕС ЭВМ (ассемблер).

Разделы УI и УП содержат описание транслятора и псевдокоманд мнемокода. Каждый раздел учебного пособия сопровождается примерами, способствующими усвоению излагаемого материала.

Макрокоманды

Системные макрокоманды представляют собой обобщенные операторы мнемокода, предназначенные для обращения к супервизору и программам управления данными.

Системные макрокоманды обрабатываются транслятором с мнемокода, используя макроопределения. В результате обработки в транслированной программе появляются поля данных и машинные команды, которые в совокупности образуют макрорасширение.

Макрорасширения макрокоманд супервизора и управления данными находятся в тексте транслированной программы потребителя и, как правило, содержат или команду ВСУ, или команду ПВР для выполнения затребованного вида обслуживания. Во время выполнения программы макрорасширение передает поля информации (параметры) подпрограмме управляющей программы для указания конкретной информации для обслуживания.

I. МАКРОКОМАНДЫ СУПЕРВИЗОРА

I.I. Системные соглашения

Введем некоторые определения, которыми будем пользоваться в дальнейшем изложении. Шагом задания называется работа, которая задается системе управляющим оператором ВПЛ языка управления заданием, или работа, которая задается системе оператором системы по команде СТАРТ, с названием программы, не являющимся названием диспетчера заданий - ДЗДН. Понятие задания дается в языке управления заданием и применимо только к тем шагам задания, которые определяются оператором ВПЛ. Под задачей понимается работа, которую может выполнить система независимо от других подобных работ. В одном шаге задания могут выполняться одна или несколько задач. Первой программой, которая начинает работать от имени задачи является программа, название которой указывается в команде оператора СТАРТ, или программа, название которой указывается в операторе ВПЛ, или же программа, указанная в операндах макрокоманды ОБРЗД.

Понятие программы и программного модуля равноценны и в дальнейшем будут использоваться как тот, так и другой термины.

Программы супервизора, управления данными, диспетчер заданий и главный диспетчер относятся к управляющим программам операционной системы.

Обрабатывающие программы операционной системы, стандартные программы, хранящиеся в системной библиотеке и все другие программы будем называть программами потребителя.

Программа, обращающаяся к другой программе и требующая после окончания ее работы возврата управления обратно, называется вызывающей.

Программа, к которой обращаются, называется соответственно вызываемой.

В качестве вызывающей и вызываемой программы может быть как потребительская, так и управляющая программа. При составлении программ должен выполняться ряд принятых соглашений, касающихся правил использования общих регистров, их сохранения при переходах от одной программы к другой, передачи информации управляющими программами и т.д. Ниже описываются эти соглашения.

I.I.I. Сохранение регистров. Для сохранения нормальных взаимосвязей между программами первым действием программы должно быть сохранение содержимого общих регистров, которые используются программой, а также содержимого регистров 0,I,I3,I4,I5. Последние сохраняются ввиду того, что при использовании в данной программе макрокоманд супервизора и управления данными, содержимое их вместе с кодом условия может изменяться.

Общие регистры запоминаются в области сохранения, занимающей I8 слов, адрес которой содержится при входе в программу в регистре I3. Формат области сохранения показан в табл.I. Содержимое каждого общего регистра запоминается в строго определенном месте области сохранения, например, регистре 0 запоминается в слове 6, регистр 9 – в слове I5. Правильность места запоминания в области сохранения должен обеспечивать программист.

Надежнее всего запоминать все регистры, это гарантирует, что последующие изменения программы не повлияют на сохранность информации в регистрах.

Таблица I

Область сохранения

Номер слова	Содержимое
I	Используется операционной системой
2	Адрес предыдущей области сохранения
3	Адрес последующей области сохранения
4	Регистр I4 (Адрес возврата)
5	Регистр I5 (Адрес входной точки)
6	Регистр 0
7	Регистр I
8	Регистр 2
9	Регистр 3
I0	Регистр 4
II	Регистр 5
I2	Регистр 6
I3	Регистр 7
I4	Регистр 8
I5	Регистр 9
I6	Регистр I0
I7	Регистр II
I8	Регистр I2

Для сохранения содержимого общих регистров в области сохранения можно использовать команду "Запись группы", например:

ЗПГ I4, I2, I2 (I3)

В данном примере эта команда помещает содержимое всех регистров, кроме регистра I3, в соответствующие слова области сохранения (сохранение содержимого регистра I3 описывается ниже).

Если же нужно запомнить регистры с 0 по 6, то можно использовать команду

ЗПГ 0, 6, 20 (I3)

Вызывающая программа перед передачей управления должна предоставить для вызываемой программы область сохранения. Предоставление области сохранения позволяет вызванной программе запоминать регистры, не разбираясь в том, была ли эта программа вызвана другой обрабатывающей программой или управляющей программой. Если программа не использует макрокоманд супервизора или управления данными и не вызывает других программ, то она может не предоставлять области сохранения. Но это применимо только для очень простых программ.

Предоставляет ли программа новую область сохранения или нет, в любом случае адрес области сохранения, которую использует программа, должен быть сохранен до конца программы. Это необходимо для восстановления содержимого общих регистров при выходе из программы. Если другая (новая) область сохранения не предоставляется, то можно хранить адрес области сохранения в регистре I3 или же запомнить его в полном слове в программе. Если же программой формируется новая область сохранения, то необходимо проделать следующие процедуры:

- запомнить адрес области сохранения, которая используется данной программой (этот адрес при входе в эту программу находился в регистре I3), во втором слове новой области сохранения;

- запомнить адрес новой области сохранения (то есть адрес, который надо поместить в регистр I3) в третьем слове старой области сохранения.

Сохранение адреса старой области сохранения в новой требуется для того, чтобы найти эту старую область сохранения, когда понадобится восстановить содержимое регистров.

Запоминание адреса новой области сохранения в старой позволяет проследить, как запоминались регистры при возникновении необычных ситуаций.

Примеры I и 2 иллюстрируют два способа получения новой области сохранения и запоминания адресов в областях сохранения.

Пример I. Обеспечение области сохранения в теле программы

```
          ...
          ЗПГ     I4,I2,I2(I3)
          ЗП      I3,ОБЛСОХР+4
          ЗР      I2,I3
          ЗА      I4,ОБЛСОХР
          ЗП      I3,8(,I2)
          ...
ОБЛСОХР   СК      I8Р'0'
```

В примере I регистры запоминаются в области сохранения, предоставленной вызывающей программой. Адрес этой области затем запоминается во втором слове новой области сохранения. Память для новой области резервируется с помощью команды СК. Регистр I2 (можно использовать любой регистр вместо него) загружается адресом предыдущей области сохранения. Адрес новой области сохранения загружается в регистр I3, и затем запоминается в третьем слове старой области сохранения.

Пример 2. Получение области сохранения вне программы

```
...
ЗПГ     I4,I2,I2(I3)
ВЫДПМ   КБ=72
ЗП      I3,4(,I)
ЗП      I,8(,I3)
ЗР      I3,I
...
```

В примере 2 регистры, как и в примере I, запоминаются в области сохранения, предоставленной управляющей программой, а затем выдается макрокоманда ВЫДПМ, по которой супервизор выделяет вне программы 72 байта и загружает адрес этой выделенной области в регистр I. Адреса новой и старой области сохранения запоминаются в соответствующих местах, причем после этого адрес новой области загружается в регистр I3.

I.I.2. Регистры связей. При обращениях к управляющей программе с помощью макрокоманд содержимое регистров 0,I,I3,I4,I5 может изменяться. Эти регистры называются регистрами связей и используются соответствующим образом управляющей программой. Рекомендуется использовать эти регистры подобным образом и для связей между любыми программами. Заметим, что регистры 2-I2 никогда не изменяются управляющей программой.

Регистры 0 и I. Эти регистры используются для передачи параметров вызываемой программе.

Макрорасширения (последовательность машинных команд, которыми транслятор заменяет макрокоманду) системных макрокоманд состоят из команд, которые загружают соответствующие значения в регистр 0 и I или же в тот и другой.

Другой метод передачи параметров заключается в загрузке регистра I адресом списка параметров. Список параметров - это

набор последовательно расположенных полных слов, содержащих адреса полей параметров. В списке каждый адрес должен помещаться в трех младших байтах слова, а старший байт должен содержать все нули, за исключением последнего слова списка, нулевой бит которого содержит единицу для указания последнего параметра в списке.

Рекомендуется передавать параметры между подпрограммами только через список параметров.

Регистр I3. Этот регистр содержит адрес области сохранения, которую выделяет программе. Управляющая программа может использовать эту область для сохранения регистров при обработке запросов, сделанных программой. Вызываемые программы запоминают регистры в этой области при входе и восстанавливают их при выходе.

Регистр I4. При входе в программу этот регистр содержит адрес, куда должна передать управление вызываемая программа после завершения своей работы. В макрорасширениях некоторых системных макрокоманд имеется команда, которая загружает регистр I4 адресом команды, следующей за макрокомандой. Команда ПУР I5,I4 в конце любой вызываемой программы возвратить управление вызывающей программе, если только содержимое регистра I4 не изменено.

Регистр I5. Когда передается управление от управляющей программы к программе потребителя, этот регистр содержит адрес точки входа в программу. Регистр I5 должен также содержать адрес точки входа и при прямых связях между проблемными программами. Кроме того, при некоторых обращениях к управляющей программе в регистр I5 загружается адрес списка параметров управляющей программы. Когда управление возвращается вызывающей программе, регистр I5 содержит код возврата.

I.I.3. Установление базового регистра. Когда управление передается от управляющей программы к программе потребителя,

адрес входной точки (то есть той точки программы, в которую должно быть передано управление от другой программы) содержится в регистре I5. Этот адрес может использоваться для установления базового регистра программы. Например:

```
ИМЯПРОГ     СТАРТ
            ИСПЛЗ   ж,I5
```

Но, как видно из п.I.I.2. регистр I5 можно использовать как постоянный базовый регистр только в программах, которые не используют системных макрокоманд и не пользуются другими программами. Поэтому в качестве базового устанавливаются один или несколько регистров 2-I2.

Например:

```
ИМЯПРОГ     СТАРТ
            ИСПЛЗ   ж,3
            ЗПГ'    I4,I2,I2(I3)
            ЗР      3,I5
            ВЫДПМ   КБ=72
            ЗП      I3,4(I)
            ЗП      I,8(,I3)
            ЗР      I3,I
```

I.I.4. Получение информации из поля ПАРМ оператора ВПЛ. Способ, которым управляющая программа передает информацию из поля ПАРМ оператора входного потока ВПЛ, является примером того, как управляющая программа использует регистр I для передачи информации. Когда управляющая программа передает управление программе, указанной в операторе ВПЛ, регистр I содержит адрес полного слова (в границах слова), находящегося в памяти задачи (рис.I). Старший разряд (разряд 0) этого слова установлен в I. Это означает, что

данное слово является единственным в списке параметров. Три младших байта слова содержат адрес двухбайтного поля длины, находящегося в границах полуслова. Поле длины (счетчик) содержит количество байтов информации поля ПАРМ. Если поле ПАРМ опущено, в поле счетчика находятся нули. Сразу же за полем длины следует информация поля ПАРМ. Если программа не собирается немедленно использовать эту информацию, адрес в регистре I надо или загрузить в один из регистров 2-I2, или запомнить его в полном слове в программе.

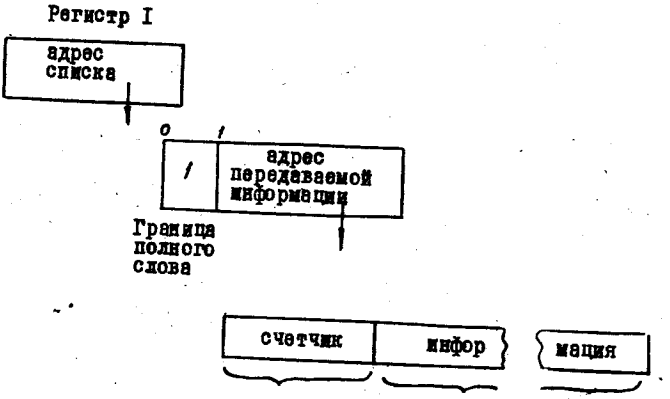


Рис.I Получение информации из поля ПАРМ

I.2. Связи между программами

Каждый программный модуль, используемый в задаче, может иметь или простую или динамическую структуру. Модуль, имеющий простую структуру, загружается в память целиком и во время выполнения не требует загрузки других программных модулей. Программный

модуль с динамической структурой выполнения может требовать загрузку и передачу управления другим программным модулям.

Программный модуль с простой структурой может состоять из нескольких программ, объединенных компоновщиком. Такое объединение весьма эффективно в смысле времени выполнения, так как передачи управления между программами не требуют вмешательства управляющей программы.

I.2.I. Передача управления в простой структуре. Ниже обсуждаются процедуры передачи управления между программами, скомпонованными в один модуль с помощью компоновщика.

Передача управления без возврата. При передаче управления от одной программы к другой без возврата необходимо сначала восстановить содержимое регистра I4, который содержит адрес возврата к вызывающей программе (например, к управляющей программе). Так как текущая программа не возвращает управления к вызывающей программе, адрес возврата нужно передать программе, которая выполнит этот возврат. К тому же, когда будет выполняться возврат, содержимое регистров 2-I2 должно остаться таким же, как и при входе в начальную программу, поэтому эти регистры также надо восстановить.

Если управление передается во входную точку программы от управляющей программы, адрес входной точки содержится в регистре I5.

Подобным же образом можно использовать регистр I5 и при непосредственной передаче управления. В этом случае программа, получающая управление, становится независимой от программы, которая передает ей управление.

Параметры между программами должны передаваться через список параметров, состоящий только из адресов; адрес списка должен загружаться в регистр I. Любые отклонения от этого правила ограничивают использование программы программистами, незнакомыми с правилами передачи параметров.

Так как все регистры вновь загружены, используемая область сохранения освобождается, и этой областью может пользоваться вызываемая программа. Адрес области сохранения передается в регистре I3.

Общим методом передачи управления от одной программы к другой, находящейся в том же программном модуле, является загрузка регистра I5 адресной константой, которая описана как внешняя, и затем передача управления по регистру I5.

Внешняя входная точка должна быть описана в вызываемой программе как ВХОД, если она не является полем названия команды СТАРТ.

Пример 3. Передача управления в простой структуре.

```
Программа I                                  Программа 2.
    ...
    ВНЕШ     ВХОДI
    ...
     S       I4,I2(,I3)                      СТАРТ
     S       I5,АДРЕСI                       ВХОД     ВХОДI
     SГ      0,I2,20(I3)                     ...
     ПУР     I5,I5           ВХОДI           ЗПГ      I4,I2,I2(I3)
     ...                                     ...
АДРЕСI ОК    A(ВХОДI)
     ...
```

Здесь входная точка ВХОДI описана как внешняя в вызывающей программе и как входная точка в вызываемой. В данном примере новая область сохранения не используется, поэтому регистр I3 содержит адрес старой области сохранения.

Кроме того, подразумевается, что вызывающая программа передает вызываемой те же параметры, которые сама получила. Сначала адрес возврата загружается в регистр I4, затем регистр I5 загружается адресом входной точки ВХОДI. Восстанавливаются регистры

0-I2 и управление передается по регистру I5.

Передача управления с возвратом. Регистры I5 и I используются точно так же, как ии при передаче управления без возврата управления. Регистр I5 содержит адрес входной точки в программу, регистр I используется для передачи списка параметров.

В соответствии со стандартными соглашениями регистр I4 должен содержать адрес, по которому надо передать управление, когда вызываемая программа закончит работу. Регистры 2-I2 при передаче управления не используются, вызываемая программа не должна зависеть от содержимого этих регистров.

Как было описано выше, перед вызовом программы необходимо выделить новую область сохранения и адрес этой области записать в регистр I3. Заметим, что та же самая область сохранения может вновь использоваться после возврата управления вызываемой программой. Обычно достаточно в программе построить одну новую область сохранения независимо от количества вызываемых программ.

Передача управления с возвратом является простым расширением передачи управления без возврата. Пример 4 иллюстрирует эту передачу.

Пример 4.

```
          ...
          ВНЕШ        ВХОДI
          ...
          З           I5,АДРI
          УНСП        0,4
          ПВ          I,ПЕРЕХСД
СПИСОК    СК          А(БУДВВСД)      адрес БУД ввода
          СК          А(БУДВЫВСД)     адрес БУД вывода
```

```
ОТВЕТ     ОК     X'80'          I в старшем разряде слова
          ОК     AL3(ОБЛОТВ)    адрес области ответа
АДРI      ОК     A (ВХОДI)
ПЕРЕХОД   ПВР    I4,I5
          ...
ОБЛОТВ    ОК     I2F'0'         область ответа
```

В примере также иллюстрируется использование списка параметров. Адрес "входной точки" загружается в регистр I5, затем используется команда "Переход с возвратом", чтобы обойти список параметров и загрузить его адрес в регистр I, а затем командой ПВР передается управление вызываемой программе.

Заметим, что старший разряд последнего адресного параметра ОТВЕТ установлен в единицу для указания конца списка.

Область, указываемая адресом в параметре ОТВЕТ, - это область, используемая вызываемой программой для передачи параметров обратно вызывающей программе. Такой способ является одним из возможных способов передачи параметров из вызываемой программы в вызывающую. Параметры передаются обратно таким образом, что не нужно никаких дополнительных регистров. Область, используемая в этом примере, состоит из двенадцати полных слов; размер области для любых специфичных приложений зависит от требований двух участвующих программ.

Анализ возврата. Если управление возвращается от управляющей программы после выполнения системной макрокоманды, содержимое регистров 2-I3 остается неизменным. Когда управление возвращается от вызванной программы, регистры 2-I4 содержат ту же информацию, что и до передачи управления, конечно, если соблюдены все системные соглашения. Вызываемая программа не обязана восстанавливать регистры 0 и I, так что содержимое этих регистров может изменяться.

Регистр I5 при возврате управления может содержать код

возврата, указывающий на исход выполнения вызываемой программы. Значение кода возврата в регистре I5 должно быть кратным 4, для того чтобы можно было легко построить таблицу переходов. При этом нулевой код возврата должен соответствовать нормальному завершению выполнения. Смысл каждого кода возврата должен быть определен заранее. В некоторых случаях для принятия решения о последующих действиях достаточно указать "хорошо" или "плохо" закончилась программа. Если это так, то для анализа результатов могут быть использованы команды, приведенные в следующем примере.

Пример 5. Анализ возврата.

```
          ...
ТЧКВОЗВР  ЗАР     I5,I5
          ПУ      7,ОШИБКА
          ...
```

Здесь анализируется на нуль код возврата и, если он не равен нулю, осуществляется выход на программу ОШИБКА.

Часто, однако, альтернативных исходов более чем два; и для выбора программы обработки различных условий требуется использовать таблицу переходов, как показано в следующем примере.

Пример 6. Анализ кода возврата с использованием таблицы переходов

```
ТЧКВОЗВР  ПУ      I5,ТАБЛИЦА(I5)
          ...
ТАБЛИЦА   ПУ      I5,ПРОДОЛЖ
          ПУ      I5,УСЛ1
          ПУ      I5,УВЛ2
          ПУ      I5,АВАРИЯ
```

Здесь в точке возврата стоит команда перехода на начало таблицы переходов плюс содержимое регистра I5, в котором находится код возврата.

Возврат управления вызывающей программе. Как указывалось выше, перед возвратом управления необходимо восстановить содержимое регистров 2-I4 до тех значений, которые они имели при входе в программу, заслать в регистр I5 код возврата и перейти по регистру I4. Регистр I4 содержит при входе в программу адрес возврата к вызывающей программе (это или обычная потребительская программа, или управляющая программа), и управление должно всегда переходить по этому адресу.

Ниже приводится пример выхода из программы, в которой ошибки подсчитываются в байте с адресом СЧЕТОШ.

Пример 7. Выход из программы

```
         З       I3,4(,I3)
         З       I4,I2(,I3)
         ВР      I5,I5
         ЗС      I5,СЧЕТОШ
         СЛИ     I5,2(0)
         ЗГ      2,I2,28(I3)
         ПУР     I5,I4
         ...
         ...
СЧЕТОШ   ОК      X'00'
```

Сначала регистр I3 загружается адресом предыдущей области сохранения. Затем адрес возврата восстанавливается в регистре I4; очищается регистр I5, после чего значение в байте СЧЕТОШ (число ошибок) помещается в младшие восемь разрядов регистра. Содержимое регистра I5 сдвигается влево на 2 разряда, чтобы сделать значение кратным 4. Регистры 2-I2 восстанавливаются, и управление передается по адресу, находящемуся в регистре I4.

Возврат в управляющую программу. Если возврат осуществляется программой, получившей управление от управляющей программы вследствие того, что ее название было указано в управляющем операторе

задания ВПЛ, управляющая программа заканчивает задачу, образованную для шага задания. Код возврата в регистре I5 сравнивается с кодом в параметре УСЛ и на основании этого управляющая программа запускает или не запускает следующий шаг задания.

I.2.2. Передача управления в динамической структуре. Если до программирования известно, какие другие программы понадобятся, и если объединение этих программ вместе с составляемой программой помещается в основной памяти, то лучше передавать управление между программами, не обращаясь к управляющей программе. Если же такое объединение невозможно, то при программировании пользуются динамическим вызовом программ, при котором обеспечивается вызов программы перед их непосредственным использованием и отказ от них после того, как они закончили работу. Макрокоманды, обсуждаемые ниже, расширяют возможности связей между программами, но они требуют вмешательства управляющей программы и могут увеличить время выполнения задачи.

Занесение программных модулей в основную память. Программный модуль, определенный в операторе ВПЛ, автоматически загружается управляющей программой в основную память. Любые другие модули, которые нужны при выполнении шага задания, заносятся в основную память управляющей программой в результате динамических запросов. Эти запросы выдаются с помощью макрокоманд СВЯЗЬ, ЗАГР, ЗАМЕН, СБРЗД.

Состояния и хранение программных модулей. Все трансляторы операционной системы выдают программу как отдельный файл, имеющий карточный формат записей (К-формат).

Компоновщик объединяет несколько программ в одну и выдает результирующий модуль как отдельный файл, но уже в Л-формате (библиотечный формат). Кроме того, любую программу в Л-формате

можно поместить в системную библиотеку, представляющую собой также отдельный файл, но содержащий уже упорядоченное по именам определенное множество программ.

Любая программа, находящаяся в одном из трех перечисленных выше состояний, может быть затребована динамически.

Если программа находится в системной библиотеке, то для вызова достаточно указать ее название. Если же она представлена отдельным файлом, то при вызове, кроме названия, необходимо указывать еще и адрес открытого блока управления данными /БУД/. Название программы в этом случае требуется для ее опознавания, а также для ссылок на нее, если она уже находится в основной памяти.

Если программа представлена файлом не в системной библиотеке, и ее необходимо вызвать из входного потока, то в операторе ВПЛ вместо названия программы указывается название оператора ОД, который описывает этот файл. В этом случае управляющая программа сама формирует и открывает блок управления данными /БУД/ для этого файла.

За исключением программ, которые заносятся в основную память при начальном запуске системы, все программы заносятся в память, отведенную для шага задания.

Поиск программ. Порядок поиска программ следующий. По названию программы, указанному в макрокоманде, просматриваются все программы, находящиеся в области памяти, отведенной шагу задания. Если программа с таким названием уже находится в этой области, поиск прекращается и используется эта программа.

Если этой программы нет, то анализируется, указан ли в макрокоманде адрес блока управления данными. Если указан адрес БУД, то программа загружается с внешнего носителя, определенного в данном БУДе. Если адрес БУД не указан, то просматриваются про-

граммы, загруженные в главную память при начальном запуске системы (сюда же относятся программы, хранящиеся в постоянном запоминающем устройстве), и если нужная программа найдена, то она используется. Если и среди этих программ нет программы с указанным названием, то осуществляется поиск ее в системной библиотеке, последующая загрузка и выполнение.

Примечание. В случаях, когда не известно заранее название программы, сформированной в виде файла на внешнем носителе, нельзя пользоваться макрокомандами динамического вызова. В этом случае необходимо писать соответствующее макрорасширение, указывая в константе, определяющей адрес имени программы, нули.

I.2.3. Макрокоманда СВЯЗЬ. Для передачи управления с возвратом между двумя программными модулями используется макрокоманда СВЯЗЬ. По этой макрокоманде вызывается управляющая программа, которая проверяет, нет ли указанного модуля в основной памяти, и если нет, он загружается из библиотеки или указанного файла. Если программы с заданным названием уже находится в основной памяти, то ей сразу же передается управление. Вызванная программа возвращает управление командой перехода по регистру I4, после чего управляющая программа возвращает управление команде, следующей за макрокомандой СВЯЗЬ.

Как и при вызове программы в простой структуре, в регистре I3 должен содержаться адрес области сохранения для вызываемой программы. Параметры к вызываемой программе должны передаваться через список, адрес которого загружается в регистр I перед макрокомандой.

Соглашения относительно регистров 2-I2 соблюдаются и здесь, управляющая программа не изменяет содержимого этих регистров, и вызываемая программа должна восстановить эти регистры перед возвратом управления. Регистр 0 используется управляющей программой

и параметры через него передавать нельзя.

Формат макрокоманды СВЯЗЬ

[идентификатор]	СВЯЗЬ	ТВХ=имя	[,БУД=адрес буд]

Если программа не именованная, то макрокоманду следует писать так:

[идентификатор]	СВЯЗЬ	ТВХ='⌴'	[,БУД=адрес БУД]

ТВХ – точка входа, идентификатор указывает название программы, по которому она отыскивается в библиотеке или в памяти.

БУД – есть адрес открытого блока управления данными, который описывает файл, содержащий программу. Если этот операнд опущен и программы нет в основной памяти, то программа загружается из системной библиотеки.

Макрорасширение макрокоманды СВЯЗЬ формирует список параметров для управляющей программы, загружая его адрес в регистр I5, после чего командой ВСУ 6 вызывается супервизор. Список параметров состоит из двух слов, содержащих название программы и адрес БУД'а Например, последовательность команд, приведенная в примере 8, равносильна макрокоманде

СВЯЗЬ ТВХ = КОСИНУС,

по которой из системной библиотеки загружается программа КОСИНУС, и ей передается управление.

Пример 8. Динамический вызов программы без макрокоманды СВЯЗЬ.

```
...
ПВ      I5, * + I2
ОК      A(ПРОГР)
ОК      A(0)        Список параметров управляющей
                    программы
ВСУ     6
...
```

```
ПРОГР    . ОК     СЛ8'КОСИНУС'
...
```

В примере указан нулевой адрес БУД'а. Это равносильно тому, что операнд БУД в макрокоманде опущен.

Макрорасширением можно воспользоваться, если по каким-либо причинам применение макрокоманды окажется затруднительным (например, название нужной подпрограммы задается во входных данных к программе, в этом случае макрокоманду применить нельзя).

I.2.4. Макрокоманда ЗАМЕН. Если при передаче управления между программными модулями не требуется возврата управления, то применяется макрокоманда ЗАМЕН.

Макрокоманда ЗАМЕН, кроме передачи управления другой программе, указывает управляющей программе, что использование программы, содержащей макрокоманду ЗАМЕН, завершилось. Поскольку управление возвращать не нужно, то адрес старой области сохранения должен быть вновь загружен в регистр I3. Адрес точки возврата должен быть восстановлен в регистре I4 из старой области сохранения так же, как и содержимое регистров 2-I2.

При использовании макрокоманды ЗАМЕН может передаваться список параметров, адрес которого содержится в регистре I. Однако в этом случае список параметров должен находиться вне программы, которая использует макрокоманду ЗАМЕН. Это делается потому, что память, занимаемая модулем, освобождается до того, как вызываемый программный модуль может использовать эти параметры.

С точки зрения способа передачи управления макрокоманда ЗАМЕН аналогична макрокоманде СВЯЗЬ (управление передается управляющей программе, использующей свой список параметров). Управляющая программа загружает, если это необходимо, копию программного модуля, устанавливает в регистре I5 адрес точки входа, запоминает

адрес, переданный в регистре I4, помещает в него новый адрес возврата, находящийся внутри управляющей программы и передает управление по регистру I5. Если программа, использовавшая макрокоманду ЗАМЕН, не была загружена макрокомандой ЗАГР, то память, занимаемая модулем, освобождается до загрузки вызванного программного модуля.

Формат макрокоманды ЗАМЕН:

[идентификатор]	ЗАМЕН	ТВХ=имя	[,БУД=адрес буд]

ТВХ – идентификатор, указывает название новой программы.

БУД – есть адрес открытого блока управления данными, который описывает файл, содержащий эту программу. Если этот операнд опущен и программы нет в главной памяти, она загружается из системной библиотеки.

Так же, как и в макрокоманде СВЯЗЬ, макрорасширение макрокоманды ЗАМЕН передает список параметров управляющей программе через регистр I5 и командой ВСУ 7 вызывается супервизор.

Вместо макрокоманды можно использовать команды, заменяющие макрорасширение.

Пример 9. Динамическая замена программы без макрокоманды ЗАМЕН

```
          ...
          ПВ      I5, * + I2
          ОК      A(ПРОГР)
          ОК      A(БУДПРОГР)
          ВСУ     7
          ...
ПРОГР     ОК      CL8'ИМЯ'
БУДПРОГР  БУД     ФОРМ=ВВП,ИМЯОД=ПРОГРI,ДЛБУФ=80
          ...
```

Список параметров управляющей программе не обязательно писать рядом с командой вызова супервизора, его можно писать вместе с константами или даже вынести его за программу и загружать адрес списка перед вызовом супервизора.

I.2.5. Макрокоманда ЗАГР. Применяется для загрузки программы в основную память без передачи ей управления. Когда выдается макрокоманда ЗАГР, управляющая программа просматривает, нет ли указанного модуля в памяти и, если он не находится в памяти, загружает его из системной библиотеки или из указанного набора данных в память пага задания. После загрузки модуля управление возвращается команде, следующей за макрокомандой ЗАГР, при этом в регистре 0 содержится адрес точки входа в загруженную программу. Если указанная программа уже находится в памяти пага задания, управляющая программа выдает адрес входа в эту программу и возвращает управление следующей команде.

Используя адрес, выданный в регистре 0, вызывающая программа может связываться с загруженной программой без использования управляющей программы. Соглашения о связях при передаче управления между программами те же, что и при передачах между отдельными программами, имеющими простую структуру (в регистр I5 загружается адрес точки входа, в регистр I3 адрес области сохранения, в регистре I4 должен находиться адрес возврата, а регистр I используется для указания списка параметров). Для передачи управления можно использовать или команду "Переход по условию";или команду "Переход с возвратом".

Кроме прямых связей с загруженным модулем можно пользоваться и макрокомандами СВЯЗЬ и ЗАМЕН, которые в этом случае передают управление программному модулю без загрузки его в память.

Формат макрокоманды ЗАГР:

[идентификатор]	ЗАГР	ТВХ=имя [,БУД=адрес буд]

ТВХ – указывает название программы, подлежащей загрузке в память.

БУД – есть адрес открытого блока управления данными, описывающего файл, содержащий требуемый программный модуль. Если операнд опущен и программы нет в главной памяти, то она загружается из системной библиотеки.

Макрорасширение этой макрокоманды состоит из команд загрузки адресов названия программы и блока управления данными соответственно в 0-й и 1-й регистры и команды вызова супервизора ВСУ 8.

Например, последовательность команд в примере 10 требует загрузки программы КОСИНУС из системной библиотеки.

Пример 10. Загрузка модуля без макрокоманды

```
         ...
         ЗА      0,АДРТВХ
         ЗА      1,0(,0)
         ВСУ     8
         ...
АДРТВХ   ОК      СL8'КОСИНУС'
```

Первая команда в этом примере загружает в регистр 0 адрес названия программы, вторая команда заносит нулевой адрес в регистр 1, что равносильно тому, что операнд БУД в макрокоманде ЗАГР опущен. Управляющая программа вызывается командой ВСУ 8.

1.2.6. Макрокоманда ОТСЕЧ. Используется для указания управляющей программе, что модуль, загруженный в память макрокомандой ЗАГР, больше не надо держать в памяти. Управляющая программа вычитает 1 из счетчика использования данной программы, и если значение счетчика достигло нуля, освобождает память, занимаемую модулем. Счетчик использования назначается управляющей программой

для каждой программы, находящейся в памяти шага задания и в каждый момент он указывает разницу между количеством макрокоманд ЗАГР и ОТСЕЧ, выданных задачей на данную программу.

Когда управление возвращается команде, следующей за макрокомандой ОТСЕЧ, регистр I5 содержит 0 в том случае, если указанный программный модуль находится в памяти до выполнения макрокоманды. Регистр I5 содержит при возврате код 4, если указанный программный модуль не обнаружен в главной памяти (в большинстве случаев это означает, что макрокоманда ОТСЕЧ была выдана без предварительной макрокоманды ЗАГР).

Формат макрокоманды ОТСЕЧ - отсечь программный модуль:

[идентификатор]	ОТСЕЧ	ТВХ=имя

ТВХ - идентификатор указывает название программы, которая была предварительно загружена в память макрокомандой ЗАГР.

Макрорасширение макрокоманды ОТСЕЧ загружает в регистр 0 адрес названия программы и вызывает управляющую программу командой ВСУ 9. Эти действия показаны в примере II.

Пример II. Отсечение программного модуля без макрокоманды

```
        ...
        ЗА          0,АДРТВХ
        ВСУ         9
        ...
АДРТВХ  ОК          СL8'КОСИНУС'
        ...
```

В этом примере отсекается программа КОСИНУС. Эта программа ранее была загружена в главную память макрокомандой:

```
ЗАГР    ТВХ = КОСИНУС
```

I.2.7. Создание подзадач. Макрокоманда ОБРЗД. При работе любой задачи шага задания любая ее программа может подключиться

параллельно, независимо от собственного выполнения цепочки программ. Такой вновь созданный процесс подсоединяется как новая задача. Задачу, в которой было создание новой задачи, называют исходной задачей, а созданную задачу подзадачей. Любая созданная задача может в свою очередь создать свои подзадачи. Создание подзадачи осуществляется с помощью макрокоманды "Образовать задачу" (ОБРЗД). Подзадачи создаются обычно на ресурсах исходной задачи, т.е. подзадаче присваивается ключ защиты памяти исходной задачи. Программы подзадачи могут использовать только ту память и только те устройства ввода-вывода, которые закреплены за шагом задания.

При создании подзадачи можно предусмотреть возможность ждать завершения этой подзадачи в задачах более высокого уровня.

Каждой задаче при ее создании присваивается некоторый приоритет, играющий роль при конкурировании задач за использование процессора. Приоритет может меняться от I до I5. Чем большее число взято в качестве приоритета, тем приоритет считается выше. Если приоритеты задач окажутся равными, преимущество отдается той задаче, которая была создана ранее.

Формат макрокоманды ОБРЗД:

[идентификатор]	ОБРЗД	ТВХ=имя [,БУД=адрес буд]
		[, БУС=адрес БУС] ,ДПР=приоритет

ТВХ - указывает название программы, по которому эта программа отыскивается. Управление программе передается по адресу, который соответствует этому имени.

БУД - есть адрес открытого блока управления данными, который описывает файл, содержащий программу. Если этот операнд опущен и программы нет в основной памяти, то программа загружается из системной библиотеки.

БУС – адрес блока управления событием, в котором супервизор производит отметку о завершении подзадачи. Адрес БУС задается тогда, когда задаче более высокого уровня нужно знать о завершении подзадачи. В этом случае в программе необходимо использовать макрокоманду ЖДАТЬ с адресом этого БУС.

ДПР – разность приоритетов создаваемой и исходной задачи. Значение может быть в диапазоне от +I5 до -I5. Подзадаче присваивается приоритет равный сумме приоритета исходной задачи и заданной разности. Если эта сумма окажется больше приоритета исходной задачи, то присваивается приоритет равный приоритету исходной задачи.

Макрорасширение макрокоманды ОБРЗД формирует список параметров для управляющей программы, загружая его адрес в регистр I5, после чего командой ВСУ 26 вызывается супервизор. Список параметров состоит из пяти слов. В I-м слове этого списка указывается адрес названия программы, во 2-м адрес БУДа, в 3-м адрес БУСа, 4-е слово резервное и заполняется нулями, в I-м полуслове 5-го слова указывается разность приоритетов, 2-е полуслово 5-го слова резервное, заполняется нулями. Формат этого списка можно видеть в примере I2 Последовательность команд в примере I2 равносильна макрокоманде ОБРЗД ТВХ=МОДУЛЬ, БУС=АДБУС, ДПР=2, которая вызывает образование подзадачи.

Пример I2. Создание подзадачи без макрокоманды ОБРЗД.

```
...
ПВ      I5, в + 24
ОК      А (ПРОГР)
ОК      А(0)
ОК      А(АДРБУС)
ОК      А(0)
```

```
                OK      H' -2'
                OK      H'0'
                BCY     26
                ...
ПРОГР           OK      CL8'МОДУЛЬ'
АДРБУС          OK      A(0)
```

В примере указан нулевой адрес БУДа, что равносильно тому, что операнд БУД в макрокоманде опущен. Подзадаче будет присвоен приоритет на 2 единицы меньше приоритета исходной задачи. По завершению подзадачи код завершения подзадачи будет помещен в ячейку с адресом АДРБУС.

Исходная задача, получив управление от супервизора после выполнения макрокоманды ОБРЗД, в регистре I имеет адрес БУЗа (блока управления задачей - специального системного блока) создаваемой подзадачи. Этот адрес БУЗа используется при отключении подзадачи.

I.2.8. Макрокоманда ОТКЗД. Отключить подзадачу можно в задаче, создавшей эту подзадачу. Для отключения подзадачи используется макрокоманда ОТКЗД. Если подзадача уже завершилась, то управляющая программа действий никаких не производит и возвращает управление в программу потребителя. Если же подзадача не завершилась, макрокоманда ОТКЗД прекращает выполнение всех программ этой подзадачи и при наличии операнда БУС в макрокоманде ОБРЗД, вызвавшей создание этой подзадачи, делает отметку в этом БУСЕ кодом 00IIB000.

Формат макрокоманды ОТКЗД:

[идентификатор]	ОТКЗД	адрес БУЗ

адрес БУЗ - идентификатор, указывает адрес ячейки, содержащий адрес БУЗа, отключаемой подзадачи.

Макрорасширение состоит из команд загрузки адреса БУЗа в регистр I и команды ВСУ 27.

Пример I3. Отключение подзадачи без макрокоманды ОТКЗД

```
....
ЗР      I,6
ВСУ     27
...
```

В примере I3 адрес адреса БУЗа отключаемой подзадачи пересылается из регистра 6 в регистр I и вызывается супервизор.

I.3. Распределение главной памяти

I.3.I. Общие сведения. Вся главная память машины делится на две области: постоянно распределенная область и динамическая область.

В постоянно распределенной области располагаются таблицы, блоки и резидентные подпрограммы управляющей программы. Здесь же могут располагаться и часто используемые обрабатывающие программы. Постоянно распределенная область занимает часть адресов оперативных запоминающих устройств (обычно начиная с младших адресов) и все адреса постоянных запоминающих устройств. Величина постоянно распределенной области зависит от назначения и конфигурации конкретной установки.

Память из динамической области распределяется для работы отдельных шагов задания. Для одного шага может быть доступна или вся динамическая область, или же эта область может быть распределена между параллельно работающими шагами задания.

Объем памяти, необходимый для работы шага задания, задается в команде оператора СТАРТ при запуске или отдельного шага или целого потока задания. Если запускается поток задания, то в этом случае указывается объем памяти, достаточный для работы любого шага задания из потока.

При полной загрузке системы динамическая область окажется разбитой на несколько непрерывных участков, в каждом из которых обрабатывается входной поток задания, или отдельная задача (например, управление процессом).

Если при запуске потока заданий в команде оператора СТАРТ объем памяти опущен, подразумевается, что надо выделить первый встретившийся свободный участок динамической области. В частности, этим участком может быть вся динамическая область.

В участке памяти, который отведен шагу задания, размещаются все программы, понадобившиеся в ходе выполнения шага, и из этого же участка памяти удовлетворяются запросы программы на память. Кроме того, на этом же участке управляющая программа располагает свои блоки и таблицы, связанные с данным шагом задания. Управляющая программа ведет учет свободной памяти в участке, который отведен шагу задания и, если памяти в участке окажется недостаточно для удовлетворения какого-либо требования шага задания, этот шаг задания будет ненормально завершен.

Запросы на память делятся на две категории: неявные и явные. Использование макрокоманд СВЯЗЬ, ЗАМЕН, ЗАГР является примером неявных запросов на память, так как эти макрокоманды обычно требуют загрузки программных модулей в память шага задания. Макрокоманда ОТКР также требует память, обеспечивая рабочую область для программы доступа к данным.

Основная память для активной задачи может быть затребована явно с помощью макрокоманды ВЫДПМ (выделить память). По этой макрокоманде выделяется указанное количество памяти из участка, который отведен шагу задания, вне программы, выполнившей макрокоманду. Память может быть освобождена макрокомандой ОСВПМ.

Эти макрокоманды допускают многократное использование одних и тех же участков памяти различными программами шага задания.

Разумное использование этих макрокоманд уменьшает общий объем памяти, необходимый для шага задания. Кроме того, эти макрокоманды позволяют создать параллельно-входимые программы, то-есть программы, которые могут одновременно использоваться параллельно работающими задачами.

Рабочие области, величина которых зависит от входных данных, лучше резервировать макрокомандой ВЫДПМ, чем резервировать их в самой программе командами ОП или ОК. Память, резервированная в самой программе трансляторными командами ОК и ОП, остается занятой все то время, пока и программа находится в главной памяти. Макрокоманды же ВЫДПМ и ОСВПМ позволяют резервировать память только на время работы тех участков программы, которые действительно нуждаются в этой памяти. В дальнейшем эта память может быть использована для загрузки и работы других програми шага задания.

I.3.2. Макрокоманда ВЫДПМ. Макрокоманда ВЫДПМ используется для требования области главной памяти. Область главной памяти выделяется из участка, отведенного шагу задания, в границах двойного слова. Отведенная область памяти не очищается. Управление возвращается команде, следующей за макрокомандой ВЫДПМ, при этом в регистре I будет находиться адрес выделенной области памяти. Если памяти в участке шага задания недостаточно для удовлетворения требования, шаг задания будет ненормально окончен.

Формат макрокоманды ВЫДПМ:

[идентификатор]	ВЫДПМ	КБ=значение

КБ - числовой параметр, указывает длину требуемой области главной памяти в байтах.

КБ - (количество байтов) должно быть кратным восьми, если это не так, то управляющая программа округляет его с избытком до числа, кратного восьми.

Когда макрокоманду ВЫДПМ нельзя применить, можно воспользоваться макрорасширением, которое загружает в регистр 0 длину требуемой области, после чего командой ВСУ 4 вызывается супервизор.

Например, последовательность команд в примере I4 требует от управляющей программы выделить область длиной в 70 байтов, что равносильно макрокоманде:

ВЫДПМ = КБ = 70

Управляющая программа выделит область памяти, загрузит ее адрес в регистр I и возвратит управление команде, следующей за командой ВСУ 4. Будет выделено 72 байта вместо 70, так как число 70 не кратно 8.

Пример I4. Требование памяти

```
...
ЗА      0,70(,0)
ВСУ     4
...
```

I.3.3. Макрокоманда ОСВПМ. Макрокоманда ОСВПМ освобождает область памяти, которая ранее была затребована макрокомандой ВЫДПМ. Начало освобождаемой области должно лежать в границах двойного слова. Управляющая программа ненормально окончит шаг задания, если указанная область не лежит в границах двойного слова или освобождается память, которая ранее не была затребована макрокомандой ВЫДПМ. Память на участке шага задания, которая уже освободилась макрокомандой ОСВПМ или вообще еще не была затребована макрокомандой ВЫДПМ, нельзя использовать в программе, так как это может привести к ненормальному окончанию всего шага.

Формат макрокоманды ОСВПМ (освободить память)

[идентификатор]	ОСВПМ	КБ=значение, А=адрес

КБ – числовой параметр, указывает длину освобождаемой области главной памяти в байтах. Значение должно быть кратным восьми, иначе управляющая программа округляет его с избытком до кратного восьми.

А – указывает адрес полного слова, которое содержит адрес освобождаемой области. Адрес освобождаемой области должен быть кратным восьми.

Макрорасширение состоит из команд загрузки длины области в регистр 0 и адреса начала области в регистр 1. Управляющая программа вызывается командой ВСУ 5. Соответствующие команды можно применить и в том случае, если, к примеру, адрес освобождаемой области или ее длина, или длина и адрес находятся в каком-либо регистре.

Пример 15. Освобождение области памяти

```
...
ЗР      1,6
ВСУ     5
...
```

Здесь адрес начала области пересылается из регистра 6 в регистр 1 и сразу же вызывается супервизор, так как предполагается, что длина области уже находилась в регистре 0.

1.4. Синхронизация выполнения задач

1.4.1. Общие сведения. Обычно в программе команды выполняются последовательно одна за другой. Такое выполнение называется синхронным. Более полное использование возможностей вычислительной системы, таких, как одновременная работа центрального процессора и устройств ввода-вывода, нескольких центральных процессоров может привести к написанию программ, в которых устройства ввода-вывода начинают или заканчивают работу в непредсказуемые моменты времени, или отдельные ветви программ могут выполняться параллельно

и независимо. Такое выполнение называется асинхронным и согласование в нем отдельных процессов требует специальных средств. Параллельно выполняющиеся цепочки программ оформляются в системе как задачи и согласование работы их друг с другом приводит к синхронизации задач между собой. Согласование же параллельной работы программы и устройств ввода-вывода приводит к синхронизации между задачей, от имени которой выполняется программа, и управляющей программой.

Выполнение программы может быть приостановлено макрокомандой ЖДАТЬ до тех пор, пока не будет достигнута некоторая точка при выполнении другого параллельного процесса. Это может быть определенная ситуация, возникающая при выполнении другой задачи, завершение операции ввода-вывода, получение ответа от оператора. Достижение этой точки рассматривается как событие и отметка о его завершении делается макрокомандой ОТМЕТ.

Каждое событие представляется блоком управления событием (БУС) и обе макрокоманды должны указывать адрес этого блока в качестве одного из операндов. Естественно, что этот адрес должен быть известен обоим параллельно работающим процессам, поэтому при любом запуске такого процесса ему сообщается адрес блока управления событием.

Блок управления событием представляет собой полное слово, расположенное в границах слова.

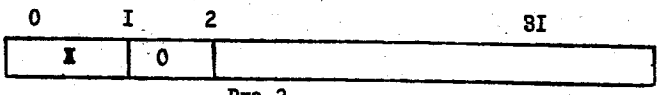


Рис.2

Когда блок управления событием впервые формируется, биты 0-й и I-й должны равняться нулю. БУС может использоваться повторно, в этом случае перед его использованием разряды 0 и I должны быть установлены в нуль. Когда выдается макрокоманда ЖДАТЬ,

разряд 0 соответствующего блока управления событием устанавливается в единицу. Когда же выдается макрокоманда ОТМЕТ, разряд I становится равным единице, а нулевой разряд устанавливается в нуль (рис.2).

Биты 2-31 БУСа используются для помещения кода завершения события, характеризующего успешно или неуспешно завершилось событие.

I.4.2. Макрокоманда ЖДАТЬ. Макрокоманда ЖДАТЬ информирует управляющую программу о том, что программа, выполнившая макрокоманду, не может продолжать вычисления до тех пор, пока не завершится событие, представленное блоком управления событием.

Управляющая программа делает следующее:

- если событие уже завершилось (в блоке управления событием уже сделана отметка), управление возвращается команде, следующей за макрокомандой ЖДАТЬ.

- если событие еще не произошло, управление не возвращается программе до тех пор, пока другой задачей или управляющей программой не будет включена макрокоманда ОТМЕТ, указывающая тот же блок управления событием. Управление возвращается команде, следующей за макрокомандой.

Формат макрокоманды ЖДАТЬ (ждать завершения события):

[идентификатор]	- ЖДАТЬ	БУС = адрес

БУС - указывает адрес блока управления событием, представляющего событие, которое должно свершиться, прежде чем управление возвратится программе.

Макрорасширение макрокоманды состоит из команды загрузки регистра I адресом блока управления событием и команды вызова супервизора ВСУ I.

Пример I6. Команды, требующие ожидания программы

```
          ...
          ЗА        I,АДРБУС
          ВСУ       I
          ...
АДРБУС    ОК        А(0)
          ...
```

Последовательность команд в примере I6 равносильна макрокоманде:

ЖДАТЬ БУС=АДРБУС

требующей приостановки программы, пока не свершится событие, представленное блоком управления событием, расположенным в полном слове по адресу АДРБУС.

I.4.3. Макрокоманда ОТМЕТ. Макрокоманда ОТМЕТ указывает, что событие, представленное блоком управления событием, свершилось. По этой макрокоманде в блок управления событием заносится код завершения события и устанавливается I в I-ом бите. Если задача ожидала этого события, она из состояния ожидания переводится в состояние готовности.

Формат макрокоманды ОТМЕТ (отметка о завершении события):

[идентификатор]	ОТМЕТ	адрес бус, код завершения

адрес бус – позиционный операнд, указывающий адрес блока управления событием;

код завершения – указывает код, который надо поместить в указанный БУС. Значение должно быть в диапазоне от 0 до $2^{30}-1$. Если код завершения опущен, подразумевают код 0.

Макрорасширение макрокоманды состоит из команды загрузки адреса БУС в регистр I и кода завершения в регистр 0. После этого

командой ВСУ 2 вызывается супервизор.

I.5. Обработка программных прерываний

Необычайные ситуации, обнаруженные в программе, вызывают программные прерывания. Эти ситуации вызываются наличием неправильных операндов, неправильной спецификации операндов или получением особых результатов. Некоторые из этих программных прерываний (переполнение при фиксированной запятой, переполнение десятичное, отрицательное переполнение порядка, потеря значимости) можно запретить (маскировать), установив в нуль соответствующие разряды слова состояния программы (ССП).

Когда задаче впервые становится действующей, все программные прерывания, которые могут маскироваться, маскируются управляющей программой. При возникновении любого программного прерывания управление передается управляющей программе, которая ненормально оканчивает задачу. Можно также указывать собственную вспомогательную подпрограмму выхода, на которую будет передаваться управление в случае определенных программистом программных прерываний. Для этого специальным образом (пример I7) указывается адрес входа в эту подпрограмму и те виды прерываний, при которых управление должно передаваться на данную подпрограмму выхода. При этом, если указывается выход по прерываниям, которые могут маскироваться, эти прерывания разрешаются. Все эти данные помещаются в области из 6-ти байтов, которая называется областью управления программными прерываниями (ОУПП). Формат ОУПП показан на рисунке 3.

Смещение
(в байтах)

0	I	2	3	4	5
0000	Маска программы	Адрес выхода		Типы прерываний	

Рис.3. Область управления программными прерываниями

Первые четыре разряда нулевого байта ОУПП содержат нули, а в оставшиеся четыре разряда помещается маска программы.

Байты I,2 и 3 содержат адрес подпрограммы выхода, которая должна получить управление при прерываниях указанного типа.

В байте 4 разряд 0 резервируется, а остальные разряды 4-го и 5-го байтов (с I-го по I5-й) используются для указания прерываний, при которых управление должна получать подпрограмма выхода. Типы прерываний указываются в соответствии с таблицей I, то есть, если, к примеру, указывается выход по прерыванию "переполнение десятичное" (прерывание № I0 по таблице 2), то разряд I0 должен быть установлен на I. Если же какое-либо прерывание не указывается, то соответствующий разряд должен содержать нуль.

Таблица 2

Типы программных прерываний

№№ пп	Типы прерываний	Примечание
I	Несуществующая операция	
2	Привилегированная операция	
3	Двойная подмена	
4	Нарушение защиты памяти	
5	Неправильная адресация	
6	Нарушение спецификации	
7	Неправильные данные	
8	Переполнение при фиксированной запятой	маскируется
9	Недопустимое деление при фиксированной запятой	
I0	Переполнение десятичное	маскируется
II	Недопустимое деление десятичное	
I2	Положительное переполнение порядка	
I3	Отрицательное переполнение порядка	маскируется

Продолжение табл.2

№№ ПП	Типы прерываний	Примечание
I4	Потеря значимости	маскируется
I5	Недопустимое деление с плавающей запятой	

В примере I7 иллюстрируется указание адреса выхода по прерываниям "неправильная адресация", "переполнение с фиксированной запятой" и "недопустимое деление с плавающей запятой" (прерывания 5, 8 и I5 по таблице I).

Пример I7. Указание адреса выхода по программным прерываниям

```
...
УНОП        0,4
ПВ          I; * + I0
ОК          A(АДРЕСВЫХ)
ОК          X '048I'
ВСУ         I8
...
```

В этом примере производится настройка границы ОУПП на границу полнослова, затем адрес ее загружается в регистр I и управление передается команде ВСУ I8. В первое слово области помещается адрес входной точки подпрограммы выхода АДРЕСВЫХ, а в оставшееся полуслово помещаются типы прерываний.

По команде ВСУ I8 управляющая программа создает новую маску программы и формирует 32-байтную область, установленную на границу двойного слова, которая называется элементом программных прерываний (ЭПП). Этот элемент создается только при первом выполнении указания адреса выхода. Формат ЭПП показан на рисунке 4.

Кроме того, по команде ВСУ I8 управляющая программа заносит в первое слово ЭПП адрес ОУПП (оставшиеся байты элемента) программных прерываний управляющая программы заполняет впоследствии, при программном прерывании). Затем управление передается команде, следующей за командой ВСУ I8.

	I	2	3
0	Адрес ОУПП		
4	Старое ССП		
I2	Регистр I4		
I6	Регистр I5		
20	Регистр 0		
24	Регистр I		
28	Регистр 2		

Рис.4 Элемент программных прерываний.

После того, как выполнилось первое указание адреса выхода, регистр I будет содержать нули. После выполнения в той же задаче каждого последующего указания адреса выхода регистр I будет содержать адрес ОУПП, сформированный для предыдущего указания адреса выхода.

Каждое последующее указание адреса выхода полностью подавляет действие предыдущего указания адреса выхода, а также действие предыдущей команды УМП (установить маску программы).

Если все байты ОУПП содержат нули, то действие предыдущего указания адреса выхода полностью подавляется и освобождается память, занимаемая элементом программных прерываний.

Можно сформировать ОУПП заранее, не используя ее сразу. Тогда для указания адреса выхода на основании этой ОУПП достаточно в любом месте программы поставить команду ВСУ I8, загрузив предварительно адрес ОУПП в регистр I.

Когда происходит программное прерывание, управляющая программа проверяет наличие ЭПП. Если ЭПП не существует в момент прерывания, то управляющая программа отдает управление программе аварийного окончания задачи. В противном случае управляющая программа проверяет типы прерываний в текущей ОУПП задачи и, если в ОУПП указаны прерывания, не совпадающие с текущим прерыванием, передает управление также программе аварийного окончания. Если же текущее прерывание указано в ОУПП, то управляющая программа запоминает в байтах 4-II элемента программного прерывания старое ССП, а содержимое регистров I4,I5,0,I,2 помещает в байты I2-3I. Затем управление передается подпрограмме выхода, адрес которой находится в ОУПП.

При входе в подпрограмму выхода содержимое регистров таково:

Регистр 0 - то же самое, что и до прерывания.

Регистр I - адрес ЭПП той задачи, которая вызвала прерывание.

Регистры 2-I3 - то же самое, что и до прерывания.

Регистр I4 - адрес возврата (точка внутри управляющей программы)

Регистр I5 - адрес данной подпрограммы выхода (этот адрес может использоваться как базовый адрес).

Для определения типа происшедшего прерывания подпрограмма выхода может опрашивать ССП, находящееся в ЭПП. Путем корректировки этогоССП подпрограмма выхода может изменить точку возврата в основную программу.

При завершении подпрограммы выхода содержимое регистра I4 должно быть таким же, как и при входе в эту подпрограмму. Подпрограмма выхода должна заканчиваться переходом по регистру I4. Управляющая программа возвращает управление к прерванной (основной) программе путем засылки ССП из ЭПП. Регистры I4,I5,0,I,2 восстанавливаются перед этим управляющей программой из ЭПП.

Если программное прерывание происходит в то время, когда подпрограмма выхода находится в действии, то управление передается на подпрограмму аварийного завершения.

I.6. Управление аварийными ситуациями

I.6.I. Общие сведения. Управляющая программа проводит большой объем проверок для обнаружения ошибок. При возникновении программных прерываний управление получает специальная программа обработки программных прерываний. Подпрограммы супервизора и управления данными также представляют некоторые средства для обнаружения ошибок.

При обнаружении ошибок, которые, возможно, могут быть исправлены, управляющая программа возвращает управление программе, при выполнении которой произошла эта ошибка. При этом выдается код возврата, указывающий возможный источник ошибок. В тех же ситуациях, когда дальнейшая обработка приведет к разрушению системы или к уничтожению существующих данных, управление передается подпрограмме аварийного завершения, которая является частью управляющей программы.

Возможно возникновение и таких аварийных ситуаций, которые являются специфичными для программы потребителя и которые управляющая программа не сможет обнаружить. Программа должна указать на свою неспособность продолжать обработку, возвратив управление вызывающей программе с кодом возврата, указывающим на ошибку.

Вызывающая программа должна затем попытаться расшифровать код возврата и избавиться от ошибки. Если вызывающая программа не может этого сделать, она должна выполнить процедуру обычного окончания и возвратить управление той программе, которая ее вызвала, также с кодом возврата, указывающим на ошибку. В конце

концов управление возвратится управляющей программе, которая закончит выполнение шага задания. Если программа запускалась через входной поток, то управляющая программа, сравнив код возврата со значениями, указанными в параметре УСЛ оператора ВПЛ, решит должен ли выполняться следующий шаг задания.

Другим случаем по отношению к этой процедуре является передача управления в управляющую программу на программу аварийного окончания с помощью макрокоманды АВОК.

Когда задача заканчивается аварийно, управляющая программа выполняет следующие действия:

1. Выдает макрокоманду ЗАКР для всех блоков управления данными, которые не были закрыты при выполнении задачи.

2. Сбрасывает все невыполненные запросы на ввод-вывод.

3. Сбрасывает временной интервал, если он был установлен для данной задачи.

4. Отменяет все запросы на ответы оператора, сделанные макрокомандой ЗКСО.

5. Код завершения, указанный в макрокоманде АВОК, записывается на системное выходное устройство для данного потока заданий.

6. Оставшиеся в задании шаги пропускаются. При этом операторы языка управления заданием проходят синтаксический контроль.

I.6.2. Макрокоманда АВОК. Макрокоманда АВОК вызывает управляющую программу для аварийного окончания задачи. В случае использования макрокоманды АВОК в задаче, образованной входным потоком, можно требовать распечатку дополнительной информации для потребителя на системное устройство: название программы, начальный адрес загрузки программы (обычно он совпадает с адресом входной точки программы), ССП в момент возникновения аварийной ситуации,

общие и хранящие регистры программы.

Формат макрокоманды АВОК (аварийно окончить задачу):

[идентификатор]	АВОК	код завершения	,ДАМП

код завершения – есть число, не превышающее 4095. Этот код будет отпечатан на системном выходном устройстве;

ДАМП – записывается, как показано. Используется, чтобы требовать распечатку дополнительной информации, относящейся к задаче.

Если использование макрокоманды вызывает затруднение, можно воспользоваться макрорасширением, которое засылает в регистр I код завершения и вызывает супервизор командой ВСУ I3. Код завершения должен находиться в разрядах 20-3I регистра I, разряды I-I9 этого регистра должны быть 0, и 0 бит должен быть установлен в I, если необходимо произвести распечатку дополнительной информации.

I.7. Связь с оператором системы

I.7.I. Введение. Возможность послать сообщение оператору системы обеспечивается макрокомандами ЗКО и ЗКОО. Кроме того, макрокоманда ЗКОО позволяет получить ответ от оператора. Сообщение может содержать любые символы, определенные клавиатурой пульта оператора; его максимальная длина не должна превышать I26 символов.

При использовании макрокоманды ЗКОО система выдает сообщение на пульт оператора системы вместе с идентификатором из двух символов, который связывает ответ оператора с сообщением. Прежде чем выдать ответ на сообщение, оператор должен выдать системе этот идентификатор. Кроме сообщения в макрокоманде ЗКОО указывается адрес области, в которую управляющая программа поместит ответ, длину ответа и адрес блока управления событием, в котором управляющая программа сделает отметку, когда ответ будет передан в

указанную область.

I.7.2. Макрокоманда ЗКО. Макрокоманда ЗКО служит для записи сообщения на пульт оператора системы.

Формат макрокоманды ЗКО (запись к оператору).

[идентификатор]	ЗКО	'сообщение'

сообщение – указывает текст, который управляющая программа должна отпечатать на пульт оператора системы. При записи макрокоманды текст заключается в кавычки. Отпечатанный текст не будет содержать их. Длина сообщения не должна превышать I30 байтов.

Текст сообщения должен иметь следующий формат:

длина текста	пробелы	текст
2 байта	2 байта	до I30 байтов

В тех случаях, когда программист не ставит макрокоманду ЗКО, он может указать в P_{15} – нули, в P_1 – адрес текста сообщения, задать текст сообщения и командой ВСУ 29 вызвать супервизор.

Пример I8. Реализация макрокоманды ЗКО

```
        ЗА      I,KI
        ВР      I5,I5
        ВСУ     29
        .
        .
        .
        ОП      ОН
KI      ОК      Х'00ДД'     длина текста сообщения
        ОК      Х'4040'     пробелы
        ОК      СЛДД'ТЕКСТ'
```

I.7.3. Макрокоманда ЗКОО. Макрокоманда ЗКОО служит для записи сообщения, требующего ответа оператора, на пульт оператора системы. Она указывает управляющей программе информацию для получения ответа.

Формат макрокоманды ЗКОО– запись к оператору с ответом:

[идентификатор]	ЗКОО	'сообщение'адрес ответа, длина ответа, адрес бус

сообщение – указывает текст, который управляющая программа должна отпечатать на пульт оператора системы. При записи макрокоманды текст заключается в кавычки, отпечатанный текст не будет их содержать. Длина сообщения не превышает I20 байтов;

адрес ответа – есть адрес области главной памяти, в которую управляющая программа разместит ответ от оператора;

длина ответа – указывает длину (в байтах) ответа на сообщение. Максимальная величина I20 байтов;

адрес бус – есть адрес блока управления событием, используемого управляющей программой, чтобы отметить получение ответа. Перед использованием ответа надо ставить макрокоманду ЖДАТЬ, указывая этот адрес.

Текст сообщения макрокоманды ЗКОО имеет тот же формат, что и в ЗКО. Область адреса ответа имеет следующие форматы:

длина текста I байт	адрес ответа 3 байта	Адрес БУС для данного ответа 4 байта

Пример I9. Реализация макрокоманды ЗКОО

```
•
•
•
ЗА      I,KI
ЗА      I5,K2
ВСУ     29
•
•
•
ОП      ОН
```

```
K1    ОК    X'00ДД'      длина текста сообщения
      ОК    X'4040'
      ОК    C'ДД'ТЕКСТ'
      ОП    ОН
K2    ОК    X'КК'        длина текста ответа
      ОК    А(облответа)
      ОК    А(БУС)
```

Если программа выдала ЗКОО и ответ в настоящий момент ей не требуется, она может продолжать работу. В тот момент, когда программе требуется ответ, обычно ставится макрокоманда "ЖДАТЬ". Ответ, полученный от оператора, фиксируется отметкой в БУСе, после чего программа может продолжить работу.

I.8. Служба времени

I.8.I. Общие сведения. При начальном запуске системы оператор устанавливает дату и время дня. После этого система сама корректирует время дня (каждые 20 мсек) и дату. Дата изменяется по достижении текущего времени 24 часов, при этом текущее время сбрасывается системой в нуль.

Время дня и текущая дата доступны для программ. Они запрашиваются макрокомандой ВРЕМЯ.

Кроме того, с помощью макрокоманды ЗВИ (задать временной интервал) может быть установлен интервал времени для задачи, по истечении которого управление будет передано программе, указанной в макрокоманде. Для задач реального времени в макрокоманде ЗВИ имеется режим задания циклического выполнения программы, по которому программа запускается через заданный промежуток времени. Установленный для задачи интервал отсоединяется макрокомандой ПВИ (проверить временной интервал).

I.8.2. Макрокоманда ВРЕМЯ. Эта макрокоманда выдает время дня в регистре 0 и текущую дату в регистре I.

Формат макрокоманды ВРЕМЯ:

[идентификатор]	ВРЕМЯ	{ДВ / ЕТ / ДС}

ДВ - указывает, что время дня выдается как беззнаковое 32-разрядное двоичное число, в котором самый младший разряд имеет значение 0,0I секунды;

ЕТ - указывает, что время дня выдается как беззнаковое 32-разрядное число, в котором самый младший разряд имеет величину I единицы таймера (I ед. тайм. = 26 мсек);

ДС - указывает, что время дня выдается в упакованном десятичном формате в форме:

ЧЧММССдс

где ЧЧ - часы, ММ - минуты, СС - секунды, д- десятые секунды, с - сотые секунды.

Текущая дата выдается в регистре I в форме:

ООГГДДЗ

где ГГ - год, ДДД - дни года, З - код знака '+'.

I.8.3. Макрокоманда ЗВИ. При помощи макрокоманды ЗВИ может быть установлен временной интервал. Эта макрокоманда указывает способ уменьшения интервала, а также действия, которые должны быть выполнены по истечении интервала.

Формат макрокоманды ЗВИ (задать временной интервал):

[идентификатор]	ЗВИ	{ЗДЧ , адрес выхода, / РЛН , адреса выхода, / ЖДН / цикл , адрес БУС}	ДС=адрес

ЗДЧ – указывает, что временной интервал должен уменьшаться только тогда, когда соответствующая задача является активной;

РЛН – указывает, что интервал должен уменьшаться непрерывно независимо от того, является ли соответствующая задача активной;

ЖДН – указывает, что интервал должен уменьшаться непрерывно и что задача, выполнившая ЗВИ, переходит в состояние ожидания до тех пор, пока не истечет интервал;

цикл – указывает, что интервал должен уменьшаться непрерывно. После истечения интервала система сама восстанавливает его и начинается отсчет заново;

адрес БУС – указывает адрес блока управления событием, в котором будет отмечаться завершение интервала времени;

адрес выхода – указывает адрес программы, на которую должно быть передано управление после завершения временного интервала. Если этот операнд опущен, то не производится никаких действий после завершения временного интервала. Этот операнд не задается при ЖДН, если он есть, то игнорируется;

ДС – указывает адрес двойного слова, содержащего десятичный интервал, который должен быть установлен в таймере. Интервал должен быть расположен в границах двойного слова и содержать 8 неупакованных цифр и в форме ЧЧММССдс. Интервал не должен превышать 24 часа.

Каждая задача может одновременно ставить две макрокоманды ЗВИ, то есть задавать интервал типа ЦИКЛ и обычный интервал типа ЖДН, РЛН, ЗДЧ.

Действие последующей макрокоманды ЗВИ типа ЦИКЛ подавляет действие предыдущей ЗВИ этого типа, встретившийся в той же задаче.

I.8.4. Макрокоманда ПВИ. Эта макрокоманда запрашивает время, оставшееся в интервале ЗДЧ РЛН, цикл который раньше был установлен макрокомандой ЗВИ. Супервизор записывает величину оставшегося

времени в регистр 0. ПВИ может также использоваться для отключения ранее указанного интервала.

Формат макрокоманды ПВИ (проверить временной интервал):

[идентификатор]	ПВИ	[ЦИКЛ / ОТКЛ / ОТКЦ]

ЦИКЛ – указывает на то, что должен быть проверен циклический интервал.

ОТКЛ – указывает, что действующий интервал типа РЛН и ЗДч должен быть отключен.

ОТКЦ – указывает, что интервал типа ЦИКЛ должен быть отключен. Если этот операнд опущен, то обработка продолжается с оставшейся частью интервала.

Время, оставшееся в интервале, записывается в регистр 0 независимо от того был или не был операнд ОТКЛ или ОТКЦ. Время выдается как беззнаковое 32-разрядное двоичное число, в котором самый младший разряд имеет величину единицы таймера. Если интервал окончился перед ПВИ, то в регистр 0 записывается нуль.

Пример 20. Работа с временным интервалом

```
          ...
          ЗВИ   ЗДН,ВРВЫХ,ДС=ВРЕМЯ   установка временного
                                      интервала
ЦИКЛ      ...
          АМ    ПРИЗНАК,X'01'       проверка,введена ли
                                      вспомогательная про-
                                      грамма ВРВЫХ
          ПУ    1,ВЫХОД             выход из цикла,если
                                      прошел временной
                                      интервал
          ПМ    12,6,ЦИКЛ           если обработка не за-
                                      кончена, снова цикл
          ПВИ   ОТКЛ                если цикл закончен,
                                      отмена оставшегося
                                      времени
```

```
ВЫХОД    ...
         ...
ВРЕМЯ    ОК    ZL8'5I2'        время 5,I2 сек
ПРИЗНАК  ОК    X'00'           переключатель тай-
                               мера.
         ...
         ИСПЛЗ ВРВЫХ,I5        обеспечение адресации
ВРВЫХ    ДЗО   ПРИЗНАК,X'0I'   временной интервал
                               прошел, установка
                               переключателя
         ...
         ПУР   I5,I4
```

В примере 20 показано, как следует использовать макрокоманды при проверке цикла в программе. Макрокоманда ЗВИ устанавливает временной интервал, равный 5,I2 секунды, который должен отсчитываться только в те моменты, когда задача будет активной, и обеспечивает адрес вспомогательной программы, называемой ВРВЫХ, на которую должно быть передано управление по истечении временного интервала. Цикл управляется командой ПМ.

Цикл продолжается до тех пор, пока значение в регистре I2 не будет меньше, или равно значению в регистре 6. Если цикл завершился, макрокоманда ПВИ сбросит время, оставшееся до конца интервала. На вспомогательную программу управление в этом случае не передается. Однако, если цикл еще не будет закончен, а интервал времени уже истечет, то управление будет передано на вспомогательную программу ВРВЫХ. Эта программа установит в нужное значение переключатель, который проверяется внутри цикла.

Программа ВРВЫХ могла бы также распечатать сообщение о том, что цикл не завершился удачно. Затем управление возвращается управляющей программе. Управляющая программа возвращает управление основной программе и обработка продолжается. На этот раз проверка переключателя приводит к выходу из цикла.

2. МАКРОКОМАНДЫ УПРАВЛЕНИЯ ДАННЫМИ

2.1. Общие сведения

2.1.1. Логическая организация данных. Логическая организация данных- это порядок представления данных на внешних носителях, предусмотренный системой, независимо от физического носителя, на котором расположены эти данные. Единицами в логической организации являются файл и логическая запись. Минимальная логическая единица информации - это логическая запись. За одно обращение к системе ввода-вывода программист может записать или прочитать только одну логическую запись. На содержание логической записи не накладывается никаких ограничений и программист вправе любую часть своей программы или данных оформить как логическую запись. Файл- более крупная единица информации, состоящая из одной или нескольких логических записей. Записи, составляющие файл, могут читаться или писаться только последовательно в порядке их расположения на носителе. Файлу может быть присвоено имя - идентификатор длиной не более 8 символов, и это имя используется системой управления данными при поиске файла на носителе или для проверки правильности поиска. Как и для логической записи, на содержание файла не накладывается никаких ограничений, и программист вправе оформить произвольную совокупность своих логических записей как файл.

Например, логическая запись- анкетные данные на сотрудника, файл- анкетные данные всех сотрудников учреждения.

Помимо записей, составляющих содержание файла, он может содержать дополнительные записи, называемые метками, содержащие служебную информацию. Такой фал называется помеченным, а файл не содержащий меток - непомеченным.

2.1.2. Физическая организация данных. Независимо от логической организации, данные, записанные на носителе, всегда разби-

ваются на части, определяемые спецификой самого носителя. Организация данных, получающаяся вследствие такого деления, носит название физическая организация. Например, данные, записанные на магнитной ленте, разбиваются на части, записываемые на отдельных бобинах. Такая минимальная физическая единица информации, которую можно сменить на внешнем устройстве, носит название том. Другие примеры томов - колода перфокарт, бобина с перфолентой.

Информация, размещенная на одном томе, как правило, может быть разбита на более мелкие физические единицы. Например, информация на бобине с магнитной лентой разбивается на зоны, отделяемые межзонными промежутками. Такая физическая единица информации носит название физической записи или блока. Характерной особенностью физической записи является то, что за одной обращение к внешнему устройству читается или записывается одна физическая запись. Другие примеры физических записей - перфокарта, строка АЦПУ.

2.I.3. Взаимоотношения между логической и физической организацией данных. Логическая и физическая организация данных могут и не совпадать друг с другом. Например, на одном томе может быть расположено несколько файлов. Далее, несколько логических записей могут быть объединены в одну физическую. Такой прием называется блокированием записей, а сами записи сблокированными или блочными. Обратный процесс выделения отдельных логических записей, объединенных в одну физическую, называется разблокированием. Однако, не допускается размещение одной логической записи в несколько физических.

2.I.4. Форматы записей. Записи в данном файле могут быть в одном из следующих 3-х форматов:

I. Фиксированные.

2. Переменные.

3. Неопределенные.

При фиксированных - записях длины всех логических записей в данном файле постоянны. Количество логических записей, объединяемых в одной физической, также постоянны для всего файла. Исключение может составлять только последняя физическая запись, которая может содержать неполное количество логических записей, и, следовательно, быть короче остальных. Таким образом, и длины всех физических записей в файле, кроме последней, постоянны. Для выделения логических записей из физической программисту должна быть известна длина логических записей. Пример фиксированного формата записей приведен на рис.5 (стр.60).

При переменных записях как длины логических записей, так и их количество в одной физической записи могут принимать на протяжении файла различные значения. Длина каждой логической записи указывается в ее первых двух байтах в виде 16-разрядного положительного двоичного числа. Следующие два байта каждой записи резервируются для использования их организующей системой. Точно таким же образом длина каждой физической записи указывается в ее первых двух байтах, а следующие два байта резервируются для использования организующей системой.

Таким образом, для чтения переменных записей программисту не нужно знать длины записей, однако он должен знать максимально возможные длины физических и логических записей для отведения рабочей области и буферов. Пример переменных записей приведен на рис.6 (стр.60).

При неопределенных записях длина записей также может меняться на протяжении файла. Однако никакого указания о длине записи внутри ее самой не содержится и система управления данными считает, что каждая физическая запись содержит только одну логическую запись. При чтении неопределенной записи ее длина сообщается программисту системой управления данными. Однако программист должен

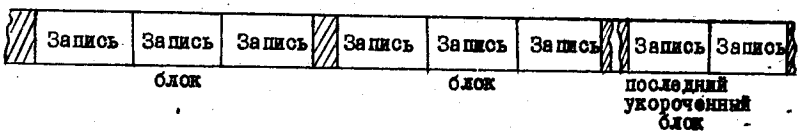
заранее знать максимально возможную длину записи, чтобы отвести область в оперативной памяти размера, достаточного для размножения любой из читаемых записей. Пример неопределенных записей приведен на рис. 7, /стр. 60/.

Записи фиксированные и переменные могут быть блокированными или нет. Таким образом, с учетом блокирования, возможны следующие 5 форматов записи:

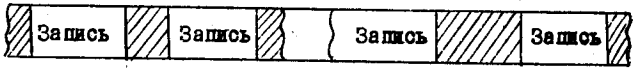
1. ФН - записи фиксированной длины, неблокированные.
2. ФБ - записи фиксированной длины, блокированные.
3. ПН - записи переменной длины, неблокированные.
4. ПБ - записи переменной длины, блокированные.
5. НН - записи неопределенной длины /неблокированные/.

2.I.5. Техника буферирования. Назначение. Область оперативной памяти, в которую данные попадают в процессе операции ввода или из которой они берутся при операции вывода называется буфером. Область оперативной памяти, из которой данные берутся или в которую заносятся во время обработки называется рабочей областью. Буфер может совпадать, а может и не совпадать с рабочей областью. Более того, для одной рабочей области может быть два буфера. Совокупность буферов, выделяемых для ввода или вывода некоторого файла, называется его буферным фондом. Применение буферов, не совпадающих с рабочей областью, является необходимым условием для организации совмещения обработки текущих логических записей с вводом-выводом последующих. Способы создания буферов и взаимодействия их с рабочими областями называются техникой буферирования, являющейся одной из функций системы ввода-вывода.

Размеры буферов и рабочих областей связаны с размерами физических и логических записей, подлежащих вводу-выводу. Конкретно размер буфера должен быть достаточен для размещения в нем наиболее длинной физической записи, а размер рабочей области, не сов-



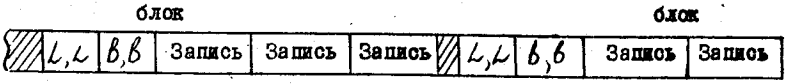
Сблокированные записи



Несблокированные записи

Межблочные разделители

Рис. 5 Фиксированные записи



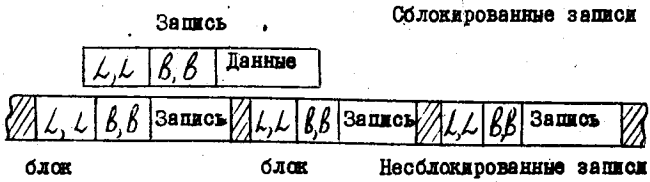


Рис.6 Переменные записи

L – длина блока;
l – длина записи;
b – байты, резервируемые для системы

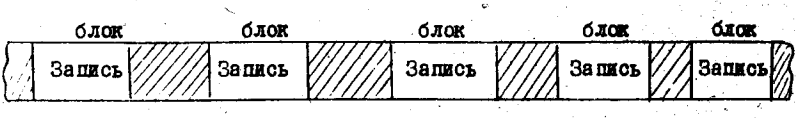


Рис. 7 Неопределенные записи

надающей с буфером - для размещения в ней наиболее длинной логической записи. В случае блокированных записей части буфера, в которые попадают отдельные логические записи, называются сегментами буфера.

Методы работы с буферами. В зависимости от количества буферов и их взаимодействия с рабочей областью в системе управления данными допускаются следующие методы (режимы) работы с буферами:

1. Режим указания места с одним буфером (УМ1)

2. Режим указания места с двумя буферами (УМ2).

3. Режим пересылки с одним буфером (ПС1).

4. Режим пересылки с двумя буферами (ПС2).

При режиме указания места (УМ) самостоятельная рабочая область отсутствует и ее роль выполняют поочередно сегменты буферов. При запросе очередной записи система управления данными сообщает (указывает) программисту адрес внутри буфера, где начинается запрошенная запись. При режиме указания места с одним буфером совмещение операции ввода-вывода с обработкой записей невозможно, так как пока не закончена обработка последней записи блока, находящегося в буфере, невозможен ввод-вывод следующего блока. При режиме указания места с 2-мя буферами возможно значительное совмещение, так как в то время, пока идет обработка записей в одном буфере, возможен ввод в другой буфер (или вывод из него). По окончании обработки записей в первом буфере роли буферов меняются.

При режиме пересылки (ПС) имеется самостоятельная рабочая область. Адрес этой области программист указывает при каждом запросе на ввод-вывод логической записи. СС при запросе на вывод пересылает запись из этой области в выходной буфер. При запросе на ввод логическая запись из входного буфера пересылается в эту область (где и возможна ее обработка); по завершении пересылки (из буфера) последней записи считанного блока буфер оказывается

свободным и возможно совмещение обработки этой последней записи с чтением в буфер следующего блока.

2.2. Организация данных на внешних носителях

2.2.1. Организация данных на магнитной ленте. На отдельной катушке магнитной ленты размещается один том. В томе на магнитной ленте может быть размещено до 9999 файлов, которым присваиваются соответственно последовательно номера от 0001 до 9999 (естественным ограничителем количества файлов на данной ленте – в пределах 9999 – является длина этой ленты). Том может полностью занимаяться одним файлом. Переход файла с одного тома на другой недопустим. Если при образовании файла при выводе будет исчерпан полезный объем тома, то файл следует закрыть, а на другом томе (катушке магнитной ленты) начать образование нового файла, являющегося продолжением старого.

Если неудобно разбить свой файл на два, можно при встрече условия "конец тома" завершить задачу аварийно, и повторить наг задания снова на новом томе, изменив информацию о томе и файле в операторе СД.

Признаком конца файла на магнитной ленте является ленточный маркер.

Для идентификации носителей на магнитной ленте, а также файлов данных, которые находятся на этих носителях, используются метки. Метки на магнитной ленте представляют собой 80-байтные записи, содержащие служебную информацию. Метка тома обязательна. Она записывается один раз до начала эксплуатации тома с помощью специальной программы. Метка тома располагается в начале ленты и содержит идентификатор тома, используемый для проверки соответствия установленного тома требуемому. Отличительными признаками метки тома являются:

- комбинация VOL1 в первых четырех байтах метки;

- начальная запись на томе состоит из 80 символов.

Номер тома задается в операторе определения данных (ОД) при вводе задания в систему. Если в результате чтения метки окажется, что номер установленного тома не совпадает с номером, указанным в ОД, система выдает сообщение об этом на пульт оператора.

Файл имеет 2 метки: головную и концевую.

Головная метка содержит информацию, описывающую файл, и указания о его использовании. Эта метка создается операционной системой в момент, когда файл записывается на ленту, и затем используется для поиска и проверки файла и для защиты файла от постороннего использования.

Система, считывая запись, устанавливает, что данная запись является головной меткой файла при совпадении следующих 2-х критериев:

- запись состоит из 80 символов;

- первые четыре символа записи образуют сочетание HDR1

Только при совпадении этих критериев система переходит к проверке метки.

Метка конца файла данных (EOF1). Содержит ту же информацию, что и головная метка файла, за исключением того, что вместо HDR1 пишется EOF1 и метка конца файла данных содержит поле счета блоков, которое показывает, сколько блоков содержится в файле. Оно используется в целях контроля считанных записей. В метке HDR1 это поле не используется. Метка конца файла повторяет головную метку файла, чтобы иметь возможность чтения ленты в обратном направлении.

Примечание. Как указывалось раньше, за меткой конца файла располагается один маркер ленты. Исключение представляет последний файл на томе; за концевой меткой которого записываются два маркера.

Непомеченные ленты. В операционной системе могут быть использованы и непомеченные файлы на ленте. Лента в этом случае называется непомеченной. Система имеет в своем распоряжении программы для автоматического поиска и установки таких лент.

Примечание. Метка тома обязательна и для непомеченных лент. В одном томе должны быть файлы одного типа: или все помеченные, или все непомеченные (рис.8). Потребитель должен указывать в операторе ОД тип метки файла (наличие или отсутствие метки).

Формат полей и содержание меток тома и файла для магнитной ленты см.в приложении (таблицы 6-9).

При работе с магнитной лентой потребитель должен знать последовательные номера файлов. Последовательные номера присваиваются начиная с I по порядку без пропусков и повторений. При поиске файла на помеченной ленте система сверяет заданный номер файла из оператора ОД с номером, записанным в метке. Совпадение означает, что искомый файл на ленте найден, в противном случае лента перематывается в начало и поиск повторяется. После второй неуспешной попытки выдается сообщение о том, что искомый файл на ленте не существует. При непомеченной ленте ответственность потребителя за указание правильного последовательного номера файла возрастает, поскольку в этом случае система никакими данными для проверки не располагает.

Новый файл на магнитной ленте может быть добавлен обычно только после тех файлов, которые уже имеются в томе. Изменять информацию в файле, за которым находится другой файл, и написать новый файл на месте старого файла нельзя, поскольку при этом не гарантируется сохранение последующих файлов.

Предположим, что на магнитной ленте размещено n файлов ($n > 10$). Необходимо образовать (записать) файл № 10. Системная программа (программа ОТКР) определяет, что файл № 10 не последний

и далее проверяет файл № IO на защиту по времени (создавал файл, потребитель указал дату, по истечении которой файл может быть разрушен). Если эта дата не достигнута, макрокоманда ОТКР выдает сообщение (с ожиданием ответа) оператору с запрещением записи. Если оператор в ответе дает разрешение писать, значит и этот и все последующие файлы потребителю не нужны, и запись будет произведена на месте файла № IO.

Если же файл № IO и все файлы, следующие за десятым, должны быть сохранены, то ответ оператора будет отрицательным, и запись не выполняется. Если файл № IO не защищен, а все последующие защищены, запись все-таки будет произведена на месте файла № IO, разрушив последующие файлы.

Организацию магнитной ленты см. на рис.8.

На магнитной ленте допустимы все 5 форматов записей (ФН, ШН, НН, ФБ, ШБ). Длина физических записей не может быть меньше 32 байтов. На магнитную ленту можно вывести любую логическую информацию.

2.2.2. Организация данных на перфоленте. На одной бобине (томе) перфоленты можно разместить только один файл- помеченный или непомеченный. Помеченный файл имеет лишь одну метку- головную, т.к. обратное чтение на перфоленте не реализуется. Эта метка записывается ОС при создании файла на перфоленте. Используется она при открытии файла для чтения (см.макрокоманду ОТКР) для проверки файла и защиты его от постороннего использования. Длина этой метки 28 байтов, формат ее полей и их содержание см. в таблице 9.

Метки тома на перфоленте нет.

На перфоленте допустимы неблокированные записи либо фиксированной, либо неопределенной длины.

Информация на данной бобине перфоленты может быть записана в одном из двух режимов:

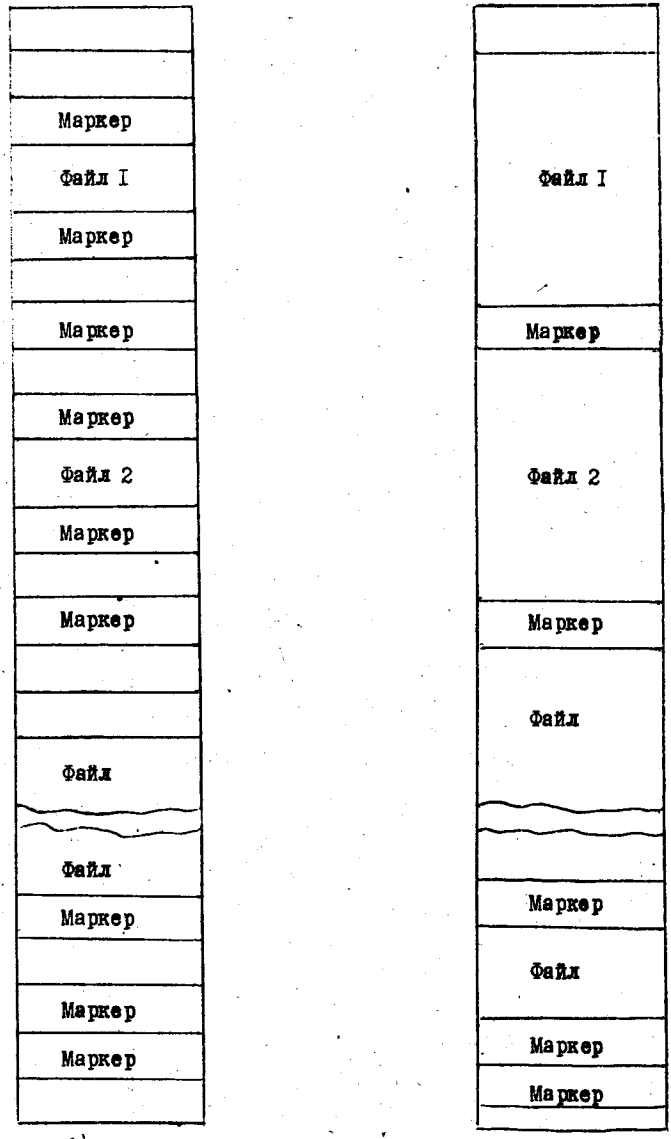


Рис.8 Организация магнитной ленты:
а) помеченной;
б) непомеченной.

а) с контрольным разрядом (алфавитно-цифровая информация, используются 7 дорожек перфоленты, 8-я дорожка - контрольная);

б) без контрольного разряда (двоичная информация, используются все 8 дорожек).

Перфолента с контрольным. Информация перед выводом на перфоленту должна быть представлена в соответствии с кодами, приведенными в приложении "Алфавитно-цифровые символы и их кодировка". При выводе информации ОС в конце каждого блока (физической записи) записывает байт со специальным кодом - разделителем информации. Во время операции чтения устройство опознает этот код и останавливает ленту (прекращает чтение). В начале файла (перед его созданием) записывается 2 таких кода подряд - они используются при установке перфоленты для чтения. Для помеченных файлов 2 разделителя подряд вводятся и после метки. Признаком конца файла является 2-х байтная зона, содержащая код /ж.

Организация перфоленты с контрольным представлена на рис. 9-10.

Примечание. При заготовке перфоленты с контрольным можно воспользоваться кодами "замена" и "аннулирование" ОС (программа ВВОД) корректирует каждую считанную зону: байт, предшествующий байту с кодом "замена", заменяется последующим байтом (сам байт с кодом "замена" аннулируется); все байты в зоне, предшествующие байту с кодом "аннулирование" (и сам этот байт) исключаются из рассмотрения - первым байтом записи, предоставляемой пользователю, является байт, следующий за байтом с этим кодом.

Перфолента без контрольного. Зоны на такой перфоленте не отделяются одна от другой - конец зоны при вводе информации определяется программным путем (по счетчику числа вводимых байтов).

Для контроля информации на двоичной перфоленте в конце каждой

зоны записывается один служебный байт - байт циклического контроля. Он анализируется ОС при чтении зоны. Как и для перфоленты с контрольным, при создании файла в начале последнего записывается 2 разделителя информации. Однако, после метки файла /для помеченной перфоленты/ они отсутствуют. Признаком конца файла является код EOF1, выводимый ОС за последней записью данного файла.

Организация перфоленты без контрольного представлена на рис. II-I2

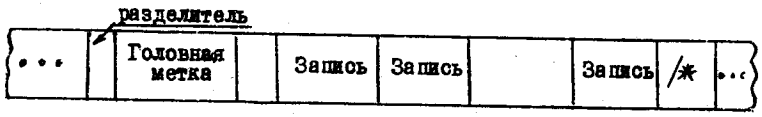


Рис. 9 Помеченный файл на перфоленте с контрольным

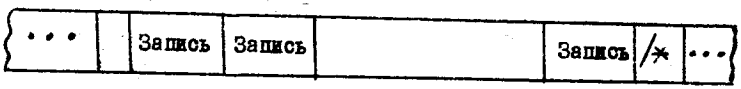


Рис.10. Непомеченный файл на перфоленте с контрольным

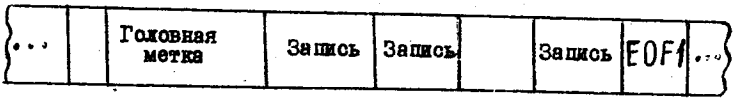


Рис. II. Помеченный файл на перфоленте без контрольного

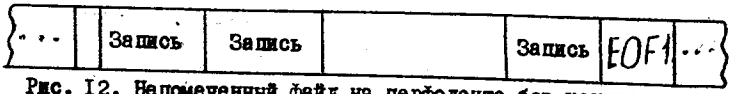


Рис. I2. Непомеченный файл на перфоленте без контрольного

2.2.3. Организация данных на перфокарте. Перфокарты представляют собой записи фиксированной длины /неблокированные/. Каждая карта рассматривается как логическая запись длиной в 80 байтов, Информация на картах должна быть представлена в соответствии с ко-

дами, приведенными в приложении "Алфавитно-цифровые символы и их кодировка".

Концом колоды карт является карта, в которой в первых двух позициях пробиты символы /* (остальные позиции произвольны).

2.2.4. Организация данных для устройств печати. На пульте программиста, на УАЦП, УАЦР допускаются неблокированные записи фиксированной или неопределенной длины. Максимальный размер записи на УАЦР - 2048 байтов, на УАЦП - 128 байтов, на пульте программиста (консоли) - 150 байтов. Минимальный размер - 1 байт. При выводе на печать информации используются алфавитно-цифровые символы. Потребитель должен представить символьную информацию в соответствии с кодами, приведенными в приложении "Алфавитно-цифровые символы и их кодировка".

2.3. Программирование операций ввода-вывода

2.3.1. Общие сведения. Операционная система содержит ряд системных программ управления данными, основное назначение которых состоит в эффективной организации работы с данными, хранящимися во внешней запоминающей среде.

Системные подпрограммы управления данными обеспечивают программисту планирование и управление передачей данных между главной памятью и устройствами ввода-вывода.

Эти подпрограммы выполняют следующие основные функции:

- считывание данных;
- запись данных;
- блокирование и разблокирование записей;
- совмещение во времени операций ввода-вывода с обработкой данных в процессоре;
- считывание и проверку меток томов и файлов;
- запись меток файла;

– заказы на установку и переустановку тома (бобины магнитной ленты);

– обнаружение ошибочных ситуаций и корректировка их (если это возможно);

– обеспечение выхода на программы, написанные потребителем (например – на подпрограммы анализа ошибок по вводу-выводу).

Программы управления данными совместно с другими средствами операционной системы представляют программисту ряд преимущества, основные из которых заключаются в следующем:

– освобождают программиста от необходимости указаний в своей программе конкретного устройства ввода-вывода;

– информацию о типе устройства, длине блока, размере буфера и другие данные можно предоставлять перед выполнением программы в операторе ОД, а не во время ее составления;

– экономится время при разработке программы;

– позволяют программисту сконцентрировать свои усилия на обработке записей, а не на передаче данных между главной памятью и внешними запоминающими устройствами.

Для подготовки и выполнения операций ввода-вывода операционная система обеспечивает программиста группой макрокоманд управления данными:

– Макрокоманда БУД – создает в теле программы блок управления данными (БУД) и вносит в этот блок характеристики файла, указанные программистом.

– Макрокоманда ОТКР – открывает файл, т.е.производит окончательное заполнение блока БУД из разных источников, завершает предварительную подготовку файла к непосредственным операциям ввода-вывода.

– Макрокоманда ВВОД – вводит логическую запись для обработки.

– Макрокоманда ВЫВОД – выводит логическую запись после обработки.

- Макрокоманда УПР – обеспечивает операции ввода-вывода, не связанные с непосредственной передачей данных (например, пропуск блока на магнитной ленте).

- Макрокоманда ЗАКР – закрывает файл, то есть приводит в исходное состояние БУД и делает файл недоступным к непосредственным операциям ввода-вывода.

Макрокоманду БУД можно ставить в любом месте программы, однако обычно ее пишут там, где определяются константы.

Макрокоманда ОТКР должна предшествовать макрокомандам ВВОД, ВЫВОД и УПР.

Макрокоманда ЗАКР используется после того, когда все необходимые операции ввода-вывода, тип которых указан в макрокоманде ОТКР, выполнены.

Примечание. Перед созданием макрокоманд управления данными пользователь должен обеспечить область сохранения регистров и указать ее адрес в R13.

2.3.2. Макрокоманда БУД (определить блок управления данными). Для каждого файла, подлежащего вводу или выводу, пользователь обязан при составлении программы указать ряд сведений о характеристиках файла и способах его обработки. Это делается с помощью макрокоманды БУД – определить блок управления данными. На этапе трансляции программы в том месте, где написана макрокоманда БУД, резервируется область, объемом в 64 байта, называемая блоком управления данными (БУД). В БУДе собирается вся информация, необходимая для выполнения операции ввода-вывода, формат блока приведен в приложении. Ряд полей БУД заполняется одновременно с его созданием на основании параметров макрокоманды БУД.

Как указывалось ранее, программисту предоставляется большая свобода в том, какие данные о файле указывать при составлении программы и какие непосредственно перед ее выполнением с помощью

оператора ОД. Конкретно, какие поля БУД, из каких источников и когда можно заполнять или изменять, обсуждается ниже при рассмотрении параметров макрокоманды БУД.

Ниже приводится формат макрокоманды БУД и кратко обсуждается ее параметры. Все параметры - ключевые. Можно задавать их в любом порядке. Обязательным является лишь параметр ФОРМ.

Формат макрокоманды БУД:

Наименование	Операция	Операнд
[идентификатор]	БУД	ФОРМ = {ВВП / ВВМ / ВП / ВМ / ВВПУ / ВВМУ / ВПУ / ВМУ} [,ИМЯОД=символ] [,АДКФ =символ] [,АДОШ =символ] [,ДЛБУФ =n] [,КЛБУФ =n] [,РЕЖПЧ = {ППЧ / ОПЧ / ЗПС / ЗБТ / ЗЧС}] [,РЕЖПЛ = {СК / БК}] [,ДЛЗП =n] [,ДЛБЛСК =n] [,ФСРЗП = {ФН / ПН / НН / ФБ / ПБ}] [,ВЫБОР= {ДП / ПР / ЗП}]

ФОРМ=код – форма макрокоманды, указывает, какая макрокоманда используется для обработки файла и режим ее работы в виде одного из следующих кодов:

ВВП – используется макрокоманда ВВОД в режиме пересылки;

ВП – используется макрокоманда ВЫВОД в режиме пересылки;

ВВМ – используется макрокоманда ВВОД в режиме указания места;

ВМ – используется макрокоманда ВЫВОД в режиме указания места.

Если в программе пользователя употребляется макрокоманда УПР, указываемый код должен содержать еще один символ справа – У (например, ФОРМ = ВВПУ). Единственным поставщиком информации в поле ФОРМ блока управления данными является макрокоманда БУД.

ИМЯОД= символ – идентификатор оператора ОД входного потока. Используется для идентификации оператора ОД, описывающего обрабатываемый файл. Это наименование может быть поставлено в БУД также программой пользователя (непосредственно) – до выполнения макрокоманды ОТКР. В частности, это необходимо сделать при использовании старого БУДа для нового файла (до открытия последнего).

АДКФ=символ – индентификатор подпрограммы пользователя "конец файла". Этой подпрограмме операционная система передает управление, если при считывании (или пропуске) очередного блока на носителе достигнут признак "конец файла" (например, маркер на магнитной ленте) или при выводе информации на носитель достигнут признак "конец тома".
Основное назначение подпрограммы "конец файла"– закрыть файл.
АДКФ может быть поставлен в БУД и непосредственно программой пользователя (в любой момент). Если АДКФ не указан, задача ненормально завершается (при обнаружении признака "конец файла" или "конец тома").

АДОШ- символ - идентификатор подпрограммы пользователя "анализ ошибок". Эта подпрограмма дополняет стандартную процедуру корректировки ошибок, предусмотренную в операционной системе для обработки ошибок ввода-вывода. Если указанная процедура оказалась безуспешной (неисправимая ошибка), управление передается подпрограмме "анализ ошибок". Последняя может определить причину и типошибки (путем анализа содержимого общих регистров, байтов состояния в блоке ввода-вывода и т.д.).

Однако, она не должна пытаться повторить операцию ввода-вывода, так как система уже делала это. Завершив анализ, эта подпрограмма может вернуть управление операционной системе (той макрокоманде, при выполнении которой обнаружилась ошибка ввода-вывода). Операционная система в этом случае сделает выбор, указанный в поле ВЫБОР блока управления данными (смотри следующий параметр макрокоманды БУД). Если же пользователь не возвращает управление операционной системе, рассматриваемый файл может быть лишь закрыт.

При передаче управления подпрограмме "анализ ошибок" операционная система обеспечивает следующее содержимое общих регистров:

Таблица 3

Регистр	Биты	Содержимое
0	0-7	Не используются
	8-3I	Адрес БВВ
I	0	Устанавливается "I", если ошибка обнаруживается при выполнении макрокоманды ВВОД
	I	Устанавливается "I",если ошибка обнаруживается при выполнении макрокоманды ВЫВОД
	2	Устанавливается "I", если ошибка обнаруживается при выполнении макрокоманды УПР

Продолжение табл.3

Регистр	Биты	Содержимое
	3	Устанавливается "I", если при выполнении макрокоманды ВВОД обнаружена неправильная длина. Этот признак устанавливается в следующих случаях: I. Длина физической записи оказалась нулевой. 2. Каналом выработан сигнал "Неправильная длина" при чтении неблокированных записей фиксированной длины. 3. Длина физической записи оказалась некратной длине логической записи при чтении блокированных записей фиксированной длины. 4. Длина физической записи оказалась неравной длине указанной в первых 2-х байтах считанной записи переменной длины.
	4 - 7	Не используются
	8 - 3I	Адрес БУД
2-I3		То же, что и до исполнения макрокоманды, при выполнении которой обнаружилась ошибка ввода-вывода (т.е.операционная система восстановит содержимое регистров 2-I3).
	I4	Адрес возврата (в операционную систему из подпрограммы "анализ ошибок").
	I5	Адрес подпрограммы "анализ ошибок".

Подпрограмма "анализ ошибок" не должна пользоваться областью сохранения, указанной в регистре I3 и не должна портить содержимое этого регистра.

Если пользователь не указал этой подпрограммы, а ошибка ввода-вывода имела место, операционная система делает выбор, предусмотренный в поле ВЫБОР БУДа. Адрес рассматриваемой подпрограммы может быть поставлен в блок управления данными, помимо макрокоманды БУД, непосредственно программой пользователя в любой момент.

Следующие параметры могут быть указаны как в макрокоманде БУД, так и в операторе СД (приоритет принадлежит первой, если па-

раметр определен в обоих источниках), кроме того, информация в соответствующие поля БУДа может быть занесена непосредственно программой пользователя до выполнения макрокоманды СТКР (исключение составляют поля ВЫБОР, ДЛЗП и ДЛБЛОК, куда информацию можно поставить и после макрокоманды СТКР).

ВЫБОР=код – выбор, который выполняет операционная система, если имеет место ошибка, а подпрограммы "анализ ошибок" либо нет, либо управление из последней возвращается операционной системе. Может быть указан один из следующих кодов;

ПР – пропустить ошибочный блок,
ДП – допустить ошибочный блок,
ЗШ – завершить задачу аварийно.
Код ПР может иметь место лишь для операций чтения (информации с носителя) и предполагает, что считанная (с ошибкой) информация обработке не подлежит, а система должна перейти к считыванию следующего блока. Код ДП – для операций чтения, записи на печатающем устройстве и операций управления. Операционная система работает так, как если бы ошибки не было. Если ни один из кодов не задан (ни одним из указанных выше источников), предполагается ЗШ.

КЛБУФ = n – количество буферов. Указывается число буферов (I или 2) для ввода-вывода файла. Если в соответствующем байте БУДа этой информации не окажется или будет указано n > 2, принимается n = I.

ДЛБУФ = n – длина буфера. Указывается длина каждого буфера в байтах. Если эта информация в БУДе отсутствует, операционная система использует длину, указанную в поле ДЛБЛОК (см.ниже – параметр ДЛБЛОК). Если отсутствует и последняя, задача аварийно завершается.

ФОРЗП = код – формат записи. Указывается формат записей в файле в виде одного из следующих кодов;
ФН – записи фиксированной длины, неблокированные,
ПН – записи переменной длины, неблокированные,

НН – записи неопределенной длины, неблокированные,
ФБ – блокированные записи фиксированной длины,
ПБ – блокированные записи переменной длины.
Коды ФБ и ПБ допустимы для магнитной ленты (признак блочности для других устройств игнорируется). Если эта информация в БУДе отсутствует, операционная система предполагает, что записи в файле – неопределенной длины.

ДЛЗП = n – размер логической записи в байтах. Наличие этой величины в соответствующем поле БУДа для вывода обязательно во всех случаях, для ввода – в случае блокированных записей. Для записей фиксированной длины – это длина каждой логической записи в файле. Для записей переменной длины – это максимальная длина логической записи в файле (заметим, что при выводе блокированных записей переменной длины этой величиной в поле ДЛЗП БУДа можно варьировать – перед каждой макрокомандой ВЫВОД – чтобы поместить в физической записи заданной длины максимальное количество логических записей). Для вывода каждой записи формата НН ее длина должна быть поставлена в поле ДЛЗП БУДа перед использованием соответствующей макрокоманды ВЫВОД; при чтении каждой записи формата НН ОС поставляет в это поле величину этой записи после соответствующей макрокоманды ВВОД.

Значение ДЛЗП (n) не должно превышать ДЛБЛОК.

ДЛБЛОК =n – размер блока (физической записи) в байтах. Наличие этой величины в соответствующем поле БУДа обязательно во всех случаях.

Для записей формата ФБ эта величина должна быть кратна длине логической записи. Для записей переменной длины – это максимальная длина блока в файле, включающая 4-х байтное поле длины, предшествующее каждому блоку.

Для записей формата НН необходимо указать максимальную длину блока в файле. Но поскольку при вводе этих записей ОС не проверяет – имеет ли место неправильная

длина, можно, если необходима лишь часть физической записи, указать в поле ДЛБЛОК БУда длину, меньшую чем действительная длина блока на носителе.

Рекомендуется $n \leq 2048$. длина блока не должна превышать длины буфера (если последняя задается).

РЕЖПЧ = код – режим работы печатающего устройства. Указывается один из следующих кодов:

ППЧ – периодическая печать,
ОПЧ – оперативная печать, } для УАЦР

ЗПС – запись,
ЗЧС – запись через строку
ЗБТ – запись без транспорта. } для АЦПУ

Если соответствующей информации в БУДе не окажется, то для УАЦР операционная система принимает ОПЧ, для АЦПУ – ЗПС.

РЕЖПЛ=код – режим работы с перфолентой. Указывается один из следующих кодов:

СК – с контрольным (алфавитно-цифровая кодировка перфоленты);
БК – без контрольного (двоичная кодировка).

Если соответствующей информации в БУДе нет, операционная система принимает БК.

2.3.3. Макрокоманда ОТКР (открыть файл). Перед началом ввода или вывода записей какого-либо файла должен быть выполнен ряд подготовительных операций, носящих название "открытие файла". Эти действия выполняются системой управления данными по макрокоманде ОТКР.

Процедура открытия файла заключается в завершении образования существующих и образовании новых управляющих блоков, установлении адресных ссылок. На основе информации, находящейся в этих блоках и описывающей файл, макрокоманда ОТКР вызывает нужную подпрограмму обработки данных из набора этих подпрограмм. Она формирует

список управляющих слов канала (УСК) в зависимости от типа устройства, отводит необходимые для операций ввода-вывода буферы. Для магнитной ленты и перфоленты проверяются либо записываются метки для помеченных файлов, производится поиск нужного файла на магнитной ленте.

Макрокомандой ОТКР может быть открыт только один массив.

Формат макрокоманды ОТКР следующий:

Поле названия	Поле операции	Поле операнда
[идентификатор]	ОТКР	[адрес буд ,вид файла]

адрес буд - идентификатор блока БУД для данного файла,

вид файла - код, указывающий метод обработки файла, может принимать одно из следующих значений:

ЧТ - для входного файла, считываемого в прямом направлении;

ОЧТ- для входного файла, считываемого в обратном направлении;

ЗП - для выходного файла.

Если "вид файла" опущен, предполагается ЧТ.

Можно вызвать программу ОТКР, используя макрорасширение, т.е. поместить в нулевой байт I-го регистра один из кодов:

00 - ЧТ

01 - ОЧТ

0Р - ЗП

в младшие три байта этого регистра - адрес БУД и поставить ВСУ I9.

Макрокоманда ОТКР использует информацию из соответствующего управляющего оператора ОД для заполнения пустых полей БУД (в макрокоманде БУД не присутствовали соответствующие этим полям параметры). Если длина буфера в БУДе (и операторе ОД) равна нулю, в качестве таковой вводится длина блока. Если и последняя отсутст-

вует, задача оканчивается аварийно.

При работе с магнитной лентой производится выдача сообщения оператору о необходимости установки тома, указанного в ОД (если он не установлен), проверяется метка установленного тома. Если установлен не тот том, макрокоманда ОТКР выдает сообщение о его снятии и повторяет выдачу сообщения об установке правильного тома Убедившись, что требуемый том установлен ОТКР, осуществляет поиск требуемого файла на этом томе (с проверкой для помеченных файлов головной или концевой метки - в зависимости от указанного вида обработки). Лента устанавливается либо перед первой записью файла (прямое чтение, запись) либо за последней записью (обратное чтение).

В случае помеченного файла при записи макрокоманда ОТКР проверяет прежде записанный файл под тем же номером на так называемую защиту по времени. Для этого метка файла содержит поле, содержащее дату, которая указывает, когда можно уничтожить записанный файл. Если текущая дата не превосходит даты, указанной в этом поле, макрокоманда ОТКР выдает сообщение о "защищенности" файла и ждет указаний о дальнейших действиях. Пользователь может указать системе писать новый файл, несмотря на "защищенность" прежнего. В этом случае (а также если файла с указанным номером нет или записанный файл не защищен по времени), формируется головная метка файла. Метка и маркер, разделяющий метку и основные записи файла, записываются на ленту.

Перед чтением с перфоленты помеченного файла ОТКР проверяет метку и устанавливает перфоленту перед первой записью. Если файл не помечен, ОТКР только устанавливает перфоленту перед первой записью.

Макрокоманда ОТКР резервирует буферный фонд из одного либо двух буферов в зависимости от количества, указанного в БУД. Она

вычисляет адрес последнего байта каждого буфера в случае чтения в обратном направлении. Если к моменту работы макрокоманды ОТКР в поле АДБУФ1 блока БУД уже есть адрес, это значит, что буферный фонд обеспечил сам программист и он настроил его на границу слова. В этом случае на его ответственности лежит и корректировка адресов буферов, если это необходимо.

Сообщения, выдаваемые макрокомандой ОТКР на пульт оператора, приведены в "инструкции оператора системы".

После выполнения макрокоманды ОТКР файл готов для первой операции ввода-вывода.

2.3.4. Макрокоманда ВЫВОД (вывести логическую запись). Макрокоманда ВЫВОД управляет заполнением выходного буфера (области главной памяти, из которой данные выводятся на устройства) и осуществляет передачу блока записей или отдельной записи на устройство (магнитную ленту А22IO, перфоленту А2II2, пульт программиста А34IO, УАЦП, А32IO, УАЦР А3220).

Формат макрокоманды ВЫВОД:

Поле названия	Поле операции	Поле операнда
[идентификатор]	ВЫВОД	адрес буд [,адрес области]

В поле операнда первый параметр указывает идентификатор блока БУД, относящегося к данному файлу, второй параметр - идентификатор рабочей области (т.е. области главной памяти, используемой пользователем для обработки отдельной логической записи). Наличие второго параметра в поле операнда необязательно. Если в поле операнда указан только первый операнд, макрокоманды ВЫВОД работает в режиме указания места, если же указаны оба операнда - в режиме пересылки.

Пользователь может вызвать программу ВЫВОД, используя макрорасширение. В этом случае он должен загрузить адрес БУД в РІ, адрес рабочей области (если она используется) в РО, в РІ5 - адрес модуля ВЫВОД (содержимое 32-35 байтов блока БУД, где после ОТКР находится адрес модуля ВЫВОД), в РІ4 - адрес возврата к программе пользователя после выполнения макрокоманды ВЫВОД и сделать переход на модуль ВЫВОДа по содержимому РІ5.

Для своей работы макрокоманда ВЫВОД использует информацию из ранее сформированных системой блоков БУД, БВВ, БУС.

В БУДе проверяется корректность заданной информации о длинах буфера, блока и логической записи, именно:

$$ДЛБУФ \geqslant ДЛБЛСК \geqslant ДЛЗП > 0$$

(Заметим, что указанные длины в БУДе - это 16-разрядные двоичные положительные числа). При нарушении этого условия задача оканчивается аварийно.

Режим указания места. Макрокоманда ВЫВОД работает в режиме указания места, если записи обрабатываются пользователем непосредственно в буфере. Обработка не должна увеличивать размер записи в буфере. Каждая макрокоманда ВЫВОД сообщает пользователю в РІ адрес области буфера, куда нужно поместить запись для вывода. Наполненный буфер не выводится до тех пор, пока не будет издана следующая макрокоманда ВЫВОД. Последнюю запись (или последний блок записей) выводит макрокоманда ЗАКР.

При неблочных записях каждая макрокоманда ВЫВОД (кроме первой) передает запись из буфера на устройство. Если же записи блочные, макрокоманда ВЫВОД не передает записи на устройство, пока буфер не заполнится. Буфер считается заполненным, если разность между длиной блока (указанной в БУДе) и суммарной длиной имеющихся в буфере логических записей меньше длины логической записи (указанной в БУДе).

Чтобы каждый блок содержал максимальное число записей переменной длины, программист может заменять максимальную величину логической записи в БУДе на ожидаемую действительную длину.

Далее рассматривается порядок работы макрокоманды ВЫВОД в режиме указания места.

Один буфер без рабочей области. Неблочные записи.

Первая макрокоманда ВЫВОД указывает адрес буфера в РI. По этому адресу нужно поместить первую запись для вывода. Вторая макрокоманда ВЫВОД начинает передавать на устройство первую запись, ждет завершения этой операции, анализирует результат завершения операции и при нормальном завершении выдает в РI адрес буфера для второй записи. Третья макрокоманда ВЫВОД начинает передавать на устройство 2-ю запись, проверяет завершение передачи 2-й записи, при нормальном исходе помещает в РI адрес буфера для 3-й записи и т.д.

Один буфер без рабочей области. Блочные записи.

Первая макрокоманда ВЫВОД указывает в РI адрес, куда нужно поместить I-ю запись, вторая - в РI адрес, куда нужно поместить 2-ю запись и т.д. Если буфер окажется заполненным после помещения " n " записей, то (n +I)-я макрокоманда ВЫВОД передает на устройство блок из " n " записей, ожидает завершения передачи, указывает в РI адрес, куда нужно поместить первую запись второго блока. Следующая макрокоманда ВЫВОД сообщает в РI адрес, куда нужно поместить вторую запись второго блока и т.д.

Два буфера без рабочей области. Неблочные записи.

Первая макрокоманда ВЫВОД указывает в РI адрес I-го буфера, куда нужно поместить I-ю запись. Вторая макрокоманда ВЫВОД начинает передачу на устройство I-й записи и сообщает в РI адрес 2-го буфера для 2-й записи. Третья макрокоманда ВЫВОД проверяет завершение передачи I-ой записи из I-го буфера, начинает передачу на

устройство 2-й записи из второго буфера и указывает в РІ адрес I-го буфера, куда нужно поместить третью запись и т.д.

Два буфера без рабочей области. Блочные записи.

Первая макрокоманда ВЫВОД указывает в РІ адрес I-го буфера, куда нужно поместить первую запись, вторая - в РІ адрес в I-ом буфере, куда нужно поместить 2-ю запись и т.д. Если буфер заполнится после помещения " n " записей, то (n +I)-я макрокоманда ВЫВОД начинает передавать на устройство блок из " n " записей из I-го буфера и сообщает в РІ адрес второго буфера, куда нужно поместить первую запись второго блока.

Следующая макрокоманда ВЫВОД указывает в РІ адрес во втором буфере, куда нужно поместить 2-ю запись блока и т.д. Когда второй буфер наполнится, соответствующая макрокоманда ВЫВОД проверит завершение передачи I-го блока записей из буфера I, начнет передачу второго блока записей из буфера 2 и укажет в РІ адрес первой записи для 3-го блока и т.д. Максимальное совмещение во времени обработки записей с выводом на устроиство происходит при двух буферах.

Режим пересылки. Макрокоманда ВЫВОД работает в режиме пересылки, если пользователь обрабатывает записи в рабочей области. В этом режиме каждая макрокоманда ВЫВОД передвигает обработанную пользователем запись из указанной рабочей области к соответствующему месту в буфере. Записи могут быть созданы и обработаны в одной и той же рабочей области или в различных рабочих областях, но только одна рабочая область может быть указана в одной макрокоманде ВЫВОД.

Если записи неблочные, каждая макрокоманда ВЫВОД передает запись из буфера на устройство.

Если записи блочные и буфер оказывается заполненным после помещения в него n записей, блок из n записей выводится на

устройство, а (n +1)-я запись становится первой в новом блоке.

При наличии двух буферов и рабочей области никакого выигрыша во времени по сравнению с двумя буферами без рабочей области нет.

Ниже рассматривается порядок работы макрокоманды ВЫВОД в режиме пересылки.

Один буфер и рабочая область. Неблочные записи.

Первая макрокоманда ВЫВОД перемещает из рабочей области первую запись в буфер и начинает передачу этой записи на устройство. Вторая макрокоманда ВЫВОД проверяет завершение передачи I-й записи, пересылает из рабочей области 2-ю запись в буфер, начинает передачу 2-й записи на устройство и т.д.

Один буфер и рабочая область. Блочные записи.

Первая макрокоманда ВЫВОД перемещает из рабочей области первую запись в буфер, 2-я - перемещает из рабочей области 2-ю запись в буфер и т.д. Если буфер заполняет " n " записей, n-я макрокоманда ВЫВОД перемещает из рабочей области в буфер n -ю запись и начинает передачу I-го блока из " n " записей на устройство. Следующая макрокоманда ВЫВОД проверяет завершение передачи I-го блока, перемещает первую запись второго блока в буфер и т.д.

Два буфера и рабочая область. Неблочные записи.

Первая макрокоманда ВЫВОД перемещает первую запись из рабочей области в I-й буфер и начинает передачу записи на устройство. Следующая макрокоманда ВЫВОД перемещает 2-ю запись во второй буфер, проверяет завершение передачи первой записи и начинает передачу на устройство второй записи из 2-го буфера. Третья макрокоманда ВЫВОД перемещает 3-ю запись из рабочей области в I-й буфер, проверяет завершение передачи 2-й записи из 2-го буфера и начинает передачу на устройство 3-й записи из I-го буфера и т.д.

Два буфера и рабочая область. Блочные записи.

Первая макрокоманда ВЫВОД перемещает I-ю запись из рабочей

области в I-й буфер. Следующая макрокоманда ВЫВОД пересылает 2-ю запись из рабочей области в I-й буфер. Если буфер заполняет n записей, n-я макрокоманда ВЫВОД пересылает n-ю запись из рабочей области в буфер, начинает передачу первого блока записей из I-го буфера на устройство. Следующая макрокоманда ВЫВОД пересылает I-ю запись второго блока во 2-й буфер и т.д.

После заполнения 2-го буфера соответствующая макрокоманда ВЫВОД проверяет завершение передачи первого блока из I-го буфера и начинает передачу на устройство 2-го блока из 2-го буфера. Следующая затем макрокоманда ВЫВОД пересылает I-ю запись 3-го блока в I-й буфер из рабочей области и т.д.

Синхронизация вывода с обработкой записи достигается за счет макрокоманды ВЫВОД, которая организует с этой целью ожидание выполнения канальной программы, прежде чем передавать на устройство следующую запись (или блок записей). После выполнения макрокоманды ВЫВОД, если она работала в режиме указаний места, регистр I содержит адрес следующей логической записи; в режиме пересылки - адрес БУДа. Регистр 0 содержит адрес рабочей области, если макрокоманда ВЫВОД работала в режиме пересылки, Р2-Р13 содержат то же, что и до выполнения макрокоманды ВЫВОД, Р14 - адрес возврата в программу пользователя после ВЫВОДа, Р15 - адрес модуля ВЫВОД.

2.3.5. Макрокоманда ВВОД (ввести логическую запись). Макрокоманда ВВОД управляет считыванием записи с внешнего устройства (магнитной ленты, перфоленты, перфокарты, пульта программиста) во входной буфер.

Формат макрокоманды ВВОД:

Поле названия	Поле операции	Поле операнда
[идентификатор]	ВВОД	адрес буд [,адрес области]

Поля операнда:

адрес буд – идентификатор блока управления данными (БУД), относящегося к данному файлу;

адрес области– идентификатор рабочей области.

Рабочая область – область главной памяти, используемая пользователем для обработки отдельной логической записи.

Второй параметр в поле операнда необязателен. Если в поле операнда указан только первый парамер, макрокоманда ВВОД работает в режиме указания места, если указаны оба параметра–в режиме пересылки.

Пользователь может вызвать программу ВВОД, используя макрорасширение. В этом случае он должен загрузить адрес БУД в РІ, адрес рабочей области (если она используется) в РО, адрес модуля ВВОД (содержимое 32–35 байтов БУД после ОТКР) – в РІ5, адрес возврата к своей программе после выполнения макрокоманды ВВОД – в РІ4 и по содержимому РІ5 сделать переход на модуль ВВОДа.

Для своей работы программа ВВОД использует информацию из ранее сформированных системой блоков БУД, БВВ, БУС.

В БУДе проверяется корректность заданной информации о длинах буфера, блока и логической записи, именно:

$$ДЛБУФ \geqslant ДЛБЛОК \ [\geqslant ДЛЗП] > 0$$

При нарушении этого условия задача оканчивается аварийно.

Режим указания места. Макрокоманда ВВОД работает в режиме указания места, если записи обрабатываются пользователем непосредственно в буфере. Обработка не должна увеличивать размер записи в буфере. Каждая макрокоманда ВВОД сообщает пользователю в РІ адрес записи в буфере.

Схема работы макрокоманды ВВОД (в режиме указания места):

I) I буфер, неблочные записи.

Каждая макрокоманда ВВОД считывает с устройства одну физическую запись (блок) рассматриваемого файла в буфер, ожидает завершения этой процедуры, при нормальном завершении выдает в I-м регистре адрес записи в буфере.

2) I буфер, блочные записи.

Пусть данный блок состоит из n логических записей. Соответственно (в программе пользователя) должно фигурировать n макрокоманд ВВОД. Первая из них считывает с устройства данный блок, ожидает завершения этой операции, при нормальном исходе выдает в РI адрес I-й логической записи этого блока (в главной памяти), 2-я - выдает в РI адрес 2-й логической записи этого блока и т.д. Наконец, n-я выдает адрес последней логической записи данного блока.

3) 2 буфера, неблочные записи.

Первый блок данного файла начинает считывать в I-й буфер макрокоманда СТКР. Первая (после СТКР) макрокоманда ВВОД ожидает завершения этой процедуры, при нормальном исходе выдает в РI адрес этого блока (в I-м буфере). Наконец, эта же макрокоманда начинает считывать во 2-й буфер 2-й блок данного файла. Следующая макрокоманда ВВОД ожидает завершения считывания во 2-й буфер, при успешном исходе выдает в РI адрес блока во 2-м буфере и начинает считывание 3-го блока данного файла в I-й буфер и т.д.

4) 2 буфера, блочные записи.

Первая макрокоманда ВВОД (после СТКР) работает аналогично предыдущему случаю, но в РI здесь выдается адрес I-й логической записи I-го блока в I-м буфере, 2-я макрокоманда ВВОД выдает в РI адрес 2-й логической записи в данном блоке, 3-я - адрес 3-й логической записи, n - (вновь полагаем, что в блоке n - логических записей) - адрес n-й логической записи. Следующая макрокоманда ВВОД аналогична 2-й макрокоманде ВВОД в предыдущем случае (но

здесь адрес в РI - это адрес I-й логической записи 2-го блока данного файла) и т.д.

Режим пересылки. В этом режиме каждая макрокоманда ВВОД передает в рабочую область пользователя логическую запись из входного буфера.

Ниже приводится схема работы макрокоманды ВВОД в режиме пересылки.

I) I буфер и рабочая область, неблочные записи. Заполнение буфера начинает макрокоманда ОТКР. Первая (после ОТКР) макрокоманда ВВОД ждет завершения операции, при нормальном исходе пересылает запись в рабочую область, затем начинает считывать в буфер 2-ю запись. Последующие макрокоманды ВВОД работают аналогично.

2) I буфер и рабочая область, блочные записи. Макрокоманда ОТКР начинает чтение I-го блока данного файла в буфер. Первая макрокоманда ВВОД ждет завершения операции и при нормальном исходе пересылает I-ю логическую запись считанного блока в рабочую область. 2-я макрокоманда ВВОД пересылает 2-ю логическую запись в рабочую область и т.д. Наконец, n-я ВВОД пересылает n-ю логическую запись, а затем начинает считывать в буфер 2-й блок. Следующая макрокоманда ВВОД ждет завершения этой операции, затем пересылает в рабочую область I-ю логическую запись 2-го блока и т.д.

3) 2 буфера и рабочая область. Макрокоманда ВВОД работает также, как в режиме указания места (с двумя буферами). Отличие лишь в том, что очередная запись пересылается в рабочую область пользователя.

Примечание. В случае блочных записей, если нужны не все записи в блоке, программист все-таки должен поставить макрокоманды ВВОД на все записи в блоке.

Содержимое общих регистров после выполнения макрокоманды

ВВОД то же, что и после макрокоманды ВЫВОД.

2.3.6. Макрокоманда УПРП (управление устройствами ввода-вывода). Макрокоманда УПР обеспечивает программу пользователя операциями управления устройствами ввода-вывода (операциями, не связанными с передачей данных).

Формат макрокоманды УПР:

Наименование	Операция	Операнд
[идентификатор]	УПР	адрес буд, код [,n]

Параметры макрокоманды УПР:

адрес буд – идентификатор блока управления данными для обрабатываемого файла;

код – мнемонический код операции управления, которая должна быть выполнена.

В следующей таблице приведены допустимые коды:

Таблица 4

Мнемонический код	I6-ричное значение	Операция управления	Устройство ввода-вывода
I	2	3	4
ПБВ	Р7	Пропуск блока вперед	Магнитная лента
ПБН	Е7	Пропуск блока назад	–"–
ПМВ	0I	Поиск маркера вперед	–"–
ПМН	02	Поиск маркера назад	–"–
ПРК	0Р	Пропуск карты	Считыватель с перфокарт
ПРС	04	Пропуск строки	Печатающие устройства (УАЦР, УАЦП, мультипрограммиста)

Продолжение табл.4

I	2	3	4
ЧРВ	I2	Чтение разделителей вперед	Считыватель с перфоленты
ЧРН	IC	Чтение разделителей назад	-"-

n - число, указывающее, сколько необходимо пропустить строк на печатающем устройстве, карт на считывателе с перфокарт, блоков на перфоленте и магнитной ленте n =256.

Все параметры макрокоманды УПР - позиционные, первые два - обязательные. Если не указано n , предполагается - I.

Можно вызвать программу УПР, пользуясь макрорасширением. Для этого необходимо:

занести в 3-й байт нулевого регистра n , в 4-й байт - КОД (шестнадцатеричное значение), в I-й регистр - адрес БУД, в I5-й - адрес входной точки УПР (содержимое байтов 5-7 БУДа после ОТКР), в I4-й - адрес возврата и выполнить переход по содержимому PI5.

При успешном завершении программы УПР управление возвращается пользователю по адресу, указанному в регистре I4. При этом восстанавливается содержимое регистров 2-I3, имевшее место до УПР, и содержимое регистров 0,I,I4,I5, созданное макрорасширением. В случае ошибки ввода-вывода управление передается подпрограмме пользователя "анализ ошибок" (см.соответствующий параметр макрокоманды БУД). Наконец, в некоторых случаях УПР передает управление подпрограмме пользователя "конец файла". Эти случаи отмечены ниже. Далее кратко поясняются отдельные процедуры, выполняемые макрокомандой УПР.

Магнитная лента. Операции ПБВ и ПБН позволяют пропустить блоков на ленте (каждый между двумя последовательными межблочными

промежутками). Если при пропуске очередного блока считан маркер ленты (являющийся признаком конца файла), лента останавливается в конце зоны маркера. Операционная система (программа УПР) при этом передает управление подпрограмме пользователя "конец файла". Предварительно в регистр I5 заносится адрес этой подпрограммы (взятый из соответствующего поля БУД) и число блоков, оставшихся непропущенными - из намеченных n (в старший байт регистра).

При выполнении операции ПМВ (ПМН) лента перемещается вперед (назад) без считывания информации в канал до обнаружения ближайшего маркера, а затем возвращается в межблочный промежуток, предшествующий этому маркеру. Таким образом, при любых перемещениях магнитной ленты с помощью макрокоманды УПР движение совершается в границах данного файла. Для выхода за пределы файла необходимо использовать макрокоманду ОТКР (открыть новый файл).

Перфолента. Операции ЧРВ и ЧРН позволяют пропустить на перфоленте блоков, каждый между двумя последовательными разделителями информации (ЧРВ - в прямом направлении, ЧРН - в обратном). Если перед выполнением ЧРВ или ЧРН лента стоит не на разделителе, первым пропускаемым блоком является участок до ближайшего разделителя.

Считыватель с перфокарт. При выполнении операции ПРК n очередных перфокарт перемещается из подающего кармана в приемный без считывания информации.

Печатающие устройства. Операция ПРС - бумага транспортируется на n строк (без печати). Конкретно вид печатающего устройства (УАЦП, УАЦР, пульт программиста) можно указать во входном потоке.

2.3.7. Особенности использования макрокоманды УПР. При использовании операций ПБВ, ПБН, ЧРВ, ЧРН, ПРК нужно учесть следующее.

При работе с одним буфером в режиме пересылки или с двумя в любом режиме макрокоманда ВВОД считывает очередной блок данных с

носителя без последующей обработки. Обрабатывать этот блок (проверять - успешно или нет он введен, перемещать логическую запись в рабочую область потребителя в режиме пересылки или выдавать в регистре I адрес этой записи в главной памяти в режиме указания места и т.д.) должна лишь следующая макрокоманда ВВОД. Это значит, что макрокоманда ВВОД, выполняющаяся перед УПР (или макрокоманда ОТКР, если речь идет о пропуске первых блоков в файле), считывает в буфер блок, который должен быть пропущен (первым), а макрокоманда ВВОД, выполняющаяся после УПР, будет этот блок обрабатывать.

Поэтому необходимо:

I) в качестве значения третьего параметра макрокоманды УПР ("n") указать $m-I$, если речь идет о пропуске m блоков на носителе ($m \geqslant 2$, если $m=I$, УПР ставить не нужно, а одна из макрокоманд ВВОД должна быть "пустой").

2) в случае неблочных записей первую после УПР макрокоманду ВВОД считать "пустой", т.е. игнорировать блок, обрабатываемый ею. Отметим, что эта макрокоманда, кроме того, считывает в буфер первый - после пропущенного - блок. В случае блочных записей должно быть К таких "пустых" макрокоманд ВВОД, где К - число логических записей в блоке (т.к. обрабатывает один блок уже К макрокоманд ВВОД).

Примечание. Если (в случае блочных записей) блок, предшествующий пропускаемым, "усекается", т.е. пользователю не нужны последние L логических записей в нем, то при работе с двумя буферами до или после макрокоманды УПР необходимо ввести еще L "пустых" макрокоманд ВВОД (помимо К макрокоманд ВВОД, упомянутых выше); а при работе с одним буфером (в режиме пересылки) достаточно после УПР ввести лишь L "пустых" макрокоманд ВВОД. В последнем случае в качестве значения третьего параметра макрокоманды УПР необходимо указать m (макрокоманда ВВОД, предшествующая УПР, здесь не считывает блок в бу-

фер, она лишь разблокирует последнюю из рассматриваемых в этом блоке логических записей).

Пример 21. Пользователю необходимы лишь первый и седьмой блоки некоторого файла на магнитной ленте. Предполагается работа с одним буфером в режиме пересылки. Записи - неблочные.

```
          .
          .
          .
          ОТКР      ЛЕНТА
          ВВОД      ЛЕНТА, РОБЛ
          .
          .
          .
          УПР       ЛЕНТА, ПБВ, 4
          ВВОД      ЛЕНТА, РОБЛ
          ВВОД      ЛЕНТА, РОБЛ
          .
          .
          .
ЛЕНТА     БУД       ФОРМ=ВВПУ,...
РОБЛ      ОП        ..............
```

Блоку управления данными (БУД) для рассматриваемого файла даем наименование ЛЕНТА, рабочей области потребителя - РОБЛ.

Макрокоманда ОТКР считывает первый блок данного файла в буфер. Первая макрокоманда ВВОД пересылает этот блок из буфера в рабочую область и считывает второй блок в буфер. Далее следует работа пользователя с информацией в рабочей области (первым блоком). После этого макрокоманда УПР пропускает 4 блока, начиная с третьего; первая за УПР макрокоманда ВВОД ("пустая") перемещает содержимое буфера (второй блок, не требующийся пользователю) в рабочую область и считывает седьмой блок в буфер; следующая макрокоманда ВВОД перемещает этот блок в рабочую область и считывает восьмой блок в буфер (если в данном файле всего 7 блоков, будет считан маркер ленты). Далее - работа пользователя с информацией в

рабочей области (седьмым блоком).

2.3.8. Закрытие файла. Макрокоманда ЗАКР. Макрокоманда ЗАКР производит действия, связанные с завершением операций ввода-вывода данного файла. Эти действия по своему смыслу обратны операциям, производимым макрокомандой ОТКР. После выполнения этой макрокоманды нельзя больше непосредственно обращаться к данному файлу (пока он снова не будет открыт).

Формат макрокоманды ЗАКР:

Наименование	Операция	Операнды
[идентификатор]	ЗАКР	адрес буд [, положение]

адрес буд – идентификатор блока БУД для данного файла,

положение – код, имеет смысл только для магнитной ленты, для других устройств игнорируется. Этот параметр указывает, в каком положении нужно оставить магнитную ленту по окончании работы с ней, и может принимать одно из следующих значений:

ОСТВ – оставить ленту на логическом конце данного файла;

СНОВ – перемотать ленту на начало данного файла для повторной работы с ним.

РАСП – указание о расположении ленты взять из оператора ОД. Там может быть указано ПРД (передача файла другому шагу), по которому требуется оставить на логическому конце ИЛИ УДЛ – перемотать ленту на начало тома.

Если параметр "положение" опущен, система полагает, что задан код РАСП.

Пользователь может вызвать программу ЗАКР, используя макрорасширение. Для этого он должен загрузить в I-й регистр адрес БУД-в младшие 3 байта, а в старший байт этого регистра поместить один

из кодов, указанных в таблице 5 и поставить ВСУ 20.

Таблица 5

Бит	Двоичное содержимое	Значение
0-I	00	Игнорируются
2-3	00	РАСП(расположение, указанное в ОД)
	0I	Расположение СНОВ
	II	Расположение ОСТВ
4-7	-	Игнорируются

Макрокоманда ЗАКР должна быть написана для каждого файла, который был открыт. Во время выполнения ЗАКР любые записи, оставшиеся в буфере, выводятся, концевые метки считываются или записываются и устанавливается флаговый бит в БУД, указывающий на прекращение дальнейших ссылок к этому файлу. Выполнение макрокоманды ЗАКР перед концом работы гарантирует системе то, что все операции ввода-вывода завершены. Если работа завершена без издания ЗАКР для этого файла, вполне вероятно, что последняя выводная запись для файла не записана верно, т.к. не было проверки на завершение работы.

Таким образом: основные функции, выполняемые макрокомандой ЗАКР:

I. Закрытие блока БУД.

2. Формирование и запись концевых меток файла при выводе.

3. Проверка концевых или головных меток файла при вводе.

4. Установка магнитной ленты в стандартное положение.

5. Освобождение главной памяти.

6. Проверка на завершение и правильность последней операции ввода-вывода.

Закрытие блока БУД заключается в восстановлении содержимого его полей, имевшего место перед выполнением макрокоманды ОТКР. После закрытия БУД может быть использован для нового файла. Для этого необходимо ввести имя нового оператора ОД - в первые 8 байтов БУДа и открыть файл - с помощью макрокоманды ОТКР.

При вводе файла с помеченной магнитной ленты макрокоманда ЗАКР читает и проверяет концевую или головную метку (в зависимости от направления чтения файла). Для выводимого на магнитную ленту помеченного файла ЗАКР сразу за последним блоком записывает маркер, затем концевую метку и 2 маркера. Если указан код ОСТВ, лента устанавливается между этими двумя маркерами, если указан код СНОВ, лента устанавливается перед головной меткой данного файла, код ПРД в параметре РАСП оператора ОД аналогичен коду ОСТВ.

Для выводимого на магнитную ленту непомеченного файла макрокоманда ЗАКР записывает 2 маркера за последней записью данных; в отличии от предыдущего случая, если указан код СНОВ, лента устанавливается перед маркером, предшествующим файлу.

При выводе файла на перфоленту с контрольным ЗАКР за последней записью данных записывает код /ж и разделитель информации; в случае перфоленты без контрольного записывается код КОИ-I.

Следует иметь в виду, что для всех устройств, если работала макрокоманда ВЫВСД в режиме указания места, макрокоманда ЗАКР выводит последнюю для данного файла запись (или блок записей) из буфера.

Макрокоманда ЗАКР проверяет завершилась ли последняя (для данного файла) операция ввода-вывода. Если не завершилась, организует ожидание завершения. При ненормальном исходе операции управление не передается подпрограмме пользователя "анализ ошибок", выполняется лишь выбор, предусмотренный в поле ВЫБОР БУДа.

Если записи вводятся, макрокоманду ЗАКР следует поставить в

той части программы, которая идентифицируется как подпрограмма "конец файла" (этой подпрограмме пользователя ОС передает управление при опознании признака "конец файла"). Однако, если требуется закончить ввод данных, не доходя до конца файла, ЗАКР следует поставить безотносительно к этой подпрограмме. Поскольку при чтении в обратном направлении I-го файла с непомеченной магнитной ленты признак "конец файла" не опознается (этому файлу не предшествует маркер - см.рис.8), следует в этом случае либо программно опознавать последний блок (а затем ставить макрокоманду ЗАКР), либо оформлять первый файл под номером 2.

ЗАКР по окончании работы устройства выключает двигатель (если это предусмотрено на данном устройстве).

Сообщения, выдываемые макрокомандой ЗАКР на пульт оператора, приведены в "Инструкции оператора системы".

При аварийном завершении задачи система проверяет - есть ли незакрытые файлы и, если таковые имеются, автоматически закрывает их.

Пример 22. Считанная с перфокарт информация после обработки выводится на магнитную ленту 320-байтными блоками (каждые четыре последовательные карты объединяются в блок). Предлагаются две из возможных реализации этой задачи.

I)

```
                .
                .
                .
          ОТКР      КАРТБУД
          ОТКР      МАГНЛБУД, ЗП
                .
                .
                .
EIO       ВВОД      КАРТБУД, ЕЗО
                .
                .
                .
          ВЫВОД     МАГНЛБУД,ЕЗО
          ПУ        I5, EIO
```

```
E20         ЗАКР        КАРТБУД
            ЗАКР        МАГНЛБУД
             .
             .
             .
КАРТБУД     БУД         ФОРМ=ВВП, ИМЯОД=КАРТА,
                        ФОРЗП=ФН, ДЛБЛОК=80,
                        АДКФ=E20
МАГНЛБУД    БУД         ФОРМ=ВП, ИМЯОД=МЛЕНТА,
                        ФОРЗП=ФБ,ДЛБЛОК=320,
                        ДЛЗП=80
E30         ОП          CL80
```

Макрокоманды ВВОД и ВЫВОД работают в режиме пересылки. Здесь используется одна рабочая область - общая для ввода и вывода (ей дано наименование E30). Макрокоманда ВВОД считывает перфокарту во входной буфер, пересылает соответствующую информацию в рабочую область, где возможна ее обработка. Затем макрокоманда ВЫВОД пересылает 80 байтов из рабочей области в выходной буфер. Эта процедура повторяется еще 3 раза, но 4-я макрокоманда ВЫВОД после пересылки очередных 80 байтов в выходной буфер уже осуществляет вывод на магнитную ленту - блока из 4-х логических записей и т.д.

Выполнение задачи заканчивается тогда, когда макрокоманда ВВОД обнаружит последнюю карту в массиве (карту с кодом /* в первых двух позициях). Управление будет передано рассматриваемой программе по адресу, идентифицированному как E20 (этот адрес задан в макрокоманде БУД для файла на перфокартах - в параметре АДКФ). Здесь оба файла закрываются. Далее в программе может быть предусмотрено решение иных задач. Но если потребуется вновь обратиться к данным файлам, их необходимо снова открыть.

Заметим, что в этой программе в макрокомандах БУД не указана длина буфера - в качестве таковой ОС использует длину блока. Не фигурируют здесь и параметры КЛБУФ и ВЫБОЧ - ОС использует стандартные значения (соответственно 1 и 3Ш).

2)

```
                .
                .
                .
        ОТКР    КАРТБУД
        ОТКР    МАГНЛБУД, ЗП
                .
                .
                .
Е10     ВЫВОД   МАГНЛБУД
        ЗР      0,I
        ЗА      I,КАРТБУД
        З       I5,32(,I)
        ПВР     I4,I5
                .
                .
                .
        ПУ      I5, Е10
Е20     ЗАКР    КАРТБУД
        ЗАКР    МАГНЛБУД
КАРТБУД БУД     ФОРМ=ВВП,ИМЯОД=КАРТА,
                ФОРЗП=ФН, ДЛБЛОК=80,
                АДКФ=Е20
МАГНЛБУД БУД    ФОРМ=ВМ,ИМЯОД=МЛЕНТА,
                ФОРЗН=ФБ,КЛБУФ=2, ДЛБЛОК=320,
                ДЛЗП=80
```

Вызов макрокоманды ВВОД

Макрокоманда ВЫВОД работает с двумя буферами в режиме указания места, ВВОД – с одним буфером в режиме пересылки.

Первая макрокоманда ВЫВОД лишь выдает в РI адрес первого выходного буфера. Этот буфер затем используется в качестве рабочей области для ввода – для этого его адрес помещается в P_0(и используется макрорасширение макрокоманды ВВОД). Первая макрокоманда ВВОД введет первую карту во входной буфер и перешлет соответствующую информацию в рабочую область (т.е. в первый выходной буфер). Затем вторая макрокоманда ВЫВОД выдает в P_I адрес на 80 байтов больший, чем адрес, выданный первой макрокомандой (ВЫВОД); вторая

макрокоманда ВВОД по этому адресу поместит 2-ю логическую запись для вывода (после чтения 2-й карты). И так до пятой макрокоманды ВЫВОД. Эта макрокоманда начнет передачу на магнитную ленту блока (из 4-х логических записей) и сообщит в P_I адрес второго выходного буфера, куда будет переслан результат чтения 5-й карты (при выполнении пятой макрокоманды ВВОД) и т.д.

Пример 23. Блоки из трех логических записей - по 100 байтов каждая - считываются с магнитной ленты, и каждая логическая запись после обработки выводится на УАЦП. Возможна следующая реализация этой задачи.

```
                .
                .
                .
            ОТКР    МЛБУД
            ОТКР    ПЕЧБУД, ЗП
                .
                .
                .
НАЧАЛО      ВВОД    МЛБУД, РАБОБЛ
                .
                .
                .
            ВЫВОД   ПЕЧБУД
            ПП      О(100,1), РАБОБЛ
            ПУ      15,НАЧАЛО
КОНЕЦ       ЗАКР    МЛБУД
            ЗАКР    ПЕЧБУД
                .
                .
                .
МЛБУД       БУД     ФОРМ=ВВП,ИМЯОД=ЛЕНТА,
                    ФОРЗП=ФБ,ДЛБЛОК=300,
                    ДЛЗП=100, АДКФ=КОНЕЦ
ПЕЧБУД      БУД     ФОРМ=ВМ,ИМЯОД=СТРОКА,
                    ФОРЗП=ФН,ДЛБЛОК=100,
                    ДЛЗП=200, РЕЖПЧ=ЗПС
РАБОБЛ      ОП      СЧ  100
```

Первая макрокоманда ВВОД введет во входной буфер первый блок и перешлет первую логическую запись этого блока в рабочую область, где она может быть обработана. Затем макрокоманда ВЫВОД выдает в P_I адрес выходного буфера. По этому адресу пересылается 100 байтов из рабочей области (команда ПП). Далее 2-я макрокоманда ВВОД пересылает в рабочую область следующую логическую запись из входного буфера, а 2-я макрокоманда ВЫВОД выполняет печать 1-й строки на УАЦП, после чего вновь выдает в P_I адрес выходного буфера. По этому адресу будут пересланы следующие 100 байтов из рабочей области (ПП). Третья макрокоманда ВВОД перешлет в рабочую область 2-ю логическую запись из входного буфера и начнет ввод второго блока с магнитной ленты и т.д.

Выполнение этой задачи заканчивается тогда, когда при очередном запросе на ввод с магнитной ленты считывается ленточный маркер - макрокоманда ВВОД при этом передаст управление рассматриваемой программе по адресу идентифицированному как КОНЕЦ.

Заметим, что параметр РЕЖПЧ в макрокоманде БУД для файла на УАЦП можно не задавать; указанное значение этого параметра - стандартное.

Приложение 2.

АЛФАВИТНО-ЦИФРОВЫЕ СИМВОЛЫ И ИХ КОДИРОВКА

В настоящем приложении определен набор алфавитно-цифровых символов, которые можно вводить с перфокарт, перфоленты и клавиатуры и выводить на печать в АСВТ, а также способ кодирования этих символов на носителях и в главной памяти.

Набор символов, их графическое изображение и кодировка приведены в таблице 6.

Операционной системой АСВТ символ ¦ воспринимается так же, как и &, символ ◊ так же, как и ⌘, символ ≠ также, как пробел (пропуск позиции), символ + и ≡ , как разделители информации, символы ⊃ и ¬ как "замена" и "аннулирование".

В графе 3 приведены кодовые изображения символов на перфокартах. Пробел может изображаться на перфокарте как отсутствием пробивок, так и кодом 0-7-8. Символы, для которых в графе 3 не указаны коды, в настоящее время не могут вводиться с помощью перфокарт.

В графе 4 приведены кодовые изображения символов на перфоленте. Символы, для которых в графе 4 не указаны коды, в настоящее время не могут вводиться с помощью перфоленты.

В графе 5 отмечено, какие символы могут вводиться с клавиатуры пульта программиста (устройство А34IO).

В графах 6,7,8 отмечено, какие символы могут печататься на пульте программиста (ПП), устройство алфавитно-цифровой печати (УАЦП), устройстве алфавитно-цифрввой регистрации (УАЦР).

В устройствах УАЦП, УАЦР символы А, В,ЕЁ, М, Н, Р, С,Т , У, Х русского и латинского алфавитов, цифра "нуль" и буква "О" русского и латинского алфавитов, цифра "три" и буква "З" русского

алфавита графически не различаются.

В устройстве УАЦП вместо символов "$\&$", "$\asymp$", "$\neq$" печатаются символы "$\dagger$", "$\diamond$", "$\neq$".

В устройстве УАЦР вместо буквы русского алфавита Д печатается буква латинского алфавита D, вместо буквы Й- буква И. Символы "IO" (основание десятичной системы счисления), "x" (знак умножения), "$\backslash$", "?", "—" (подчеркивание), "$\leq$", "$\geqslant$", "$\supset$", "$\top$", "$\div$", "$\equiv$", "|" (вертикальная черта), "—" (горизонтальная черта), "°" (градус), "Ъ" на УАЦП и символы "$\smile$", "$\cdot$", "IO" на УАЦР могут печататься без использования автоматической перекодировки (на физическом уровне). При работе с операционной системой, обеспечивающей автоматическую перекодировку на входе и выходе, эти символы не могут быть отпечатаны.

В графе 9 приведены двоичные кодовые представления символов в главной памяти системы (после соответствующего перекодирования операционной системой). Символы, отмеченные прочерками, рассматриваются операционной системой, как пустые символы, т.е. исключаются из потока входной информации.

Символы, отмеченные звездочкой, используются только операционной системой и недоступны обычным программам потребителя. Остальные символы могут использоваться программами потребителя произвольным способом.

Таблица 6

Название символа	Графическое изображение	Кодировка на перфокартах	Кодирование на перфоленте (разряды 8765432I)	Ввод с ПП	Печать: ПП	Печать: УАЦП	Печать: УАЦР	Машинный код (разряды 01234567)	Графическое изображение
I	2	3	4	5	6	7	8	9	I0
Буквы русского алфавита	А	II	00I00000	+	+	+	+	IIIOOOOI	А
	Б	II-0-I	I0I00000	+	+	+	+	IIIOOOIO	Б
	В	II-0-2	I0I0000I0	+	+	+	+	IIIIOIII	В
	Г	II-0-8	00I00011	+	+	+	+	IIIOOIII	Г
	Д	II-0-4	I0I00I00	+	+	+	–	IIIOOIOO	Д
	Е	II-0-5	00I00I0I	+	+	+	+	IIIOOIOI	Е
	Ж	II-0-6	00I00II0	+	+	+	+	IIIIOIIO	Ж
	З	II-0-7	I0I00III	+	+	+	+	IIIIIOIO	З
	И	II-0-8	I0I0I000	+	+	+	+	IIIOIOOI	И
	Й	II-0-9	00I0I00I	+	+	+	–	IIIOIOIO	Й
	К	II-2-8	00I0I0I0	+	+	+	+	IIIOIOII	К
	Л	II-8-8	I0I0I0II	+	+	+	+	IIIOIIOO	Л
	М	II-4-8	00I0II00	+	+	+	+	IIIOIIOI	М
	Н	II-5-8	I0I0II0I	+	+	+	+	IIIOIIIO	Н

Продолжение табл. 6

I	2	3	4	5	6	7	8	9	IO
	О	II-6-8	IOIOIIIO	+	+	+	–	IIIOIIII	О
	П	II-7-8	OOIOIIII	+	+	+	+	IIII0000	П
	Р	I2-II-0	IOII0000	+	+	+	+	IIII00I0	Р
	С	I2-II-I	OOII000I	+	+	+	+	IIII00II	С
	Т	I2-II-2	OOII00I0	+	+	+	+	IIII0I00	Т
	У	I2-II-3	IOII00II	+	+	+	+	IIII0I0I	У
	Ф	I2-II-4	OOII0I00	+	+	+	+	IIII00II0	Ф
	Х	I2-II-5	IOII0I0I	+	+	+	+	IIII0I000	Х
	Ц	I2-II-6	IOII0II0	+	+	+	+	IIII000II	Ц
	Ч	I2-II-7	OOII0III	+	+	+	+	IIIIIII0	Ч
	Ш	I2-II-8	OOIII000	+	+	+	+	IIIII0II	Ш
	Щ	I2-II-9	IOIII00I	+	+	+	+	IIIIII0I	Щ
	Ы	I2-II-0-2-8	IOIII0I0	+	+	+	+	IIIII00I	Ы
	Ь	I2-II-0-3-8	OOIII0II	+	+	+	+	IIIII000	Ь
	Э	I2-II-0-4-8	IOIIII00	+	+	+	+	IIIIII00	Э
	Ю	I2-II-0-5-8	OOIIII0I	+	+	+	+	III00000	Ю
	Я	I2-II-0-6-8	OOIIIII0	+	+	+	+	IIII000I	Я

Продолжение табл. 6

I	2	3	4	5	6	7	8	9	IO
Буквы латинского алфавита	A	II	OOIOOOOO	+	+	+	+	IOIOOOOI	A
	B	II-0-2	IOIOOOIO	+	+	+	+	IOIOOOIO	B
	C	I2-II-I	OOIIOOOI	+	+	+	+	IOIOOOII	C
	D	I2-II-0-7-8	IOIIIIII	+	+	+	+	IOIOOIOO	D
	E	II-0-5	OOIOOIOI	+	+	+	+	IOIOOIOI	E
	F	0-8-9	OIOOOOOO	+	+	+	+	IOIOOIIO	F
	G	I-8-9	IIOOOOOI	+	+	+	+	IOIOOIII	G
	H	II-5-8	IOIOIIOI	+	+	+	+	IOIOIOOO	H
	I	2-8-9	IIOOOOIO	+	+	+	+	IOIOIOOI	I
	J	0-I-2-8-9	OIOOOOII	+	+	+	+	IOIOIOIO	J
	K	II-2-8	OOIOIOIO	+	+	+	+	IOIOIOII	K
	L	8-4-9	IIOOOIOO	+	+	+	+	IOIOIIOO	L
	M	II-4-8	OOIOIIOO	+	+	+	+	IOIOIIOI	M
	N	8-5-9	OIOOOIOI	+	+	+	+	IOIOIIIO	N
	O	II-6-8	IOIOIIIO	+	+	+	+	IOIOIIII	O
	P	I2-II-0	IOIIOOOO	+	+	+	+	IOIIOOOO	P
	Q	8-6-9	OIOOOIIOO	+	+	+	+	IOIIOOOI	Q

Продолжение табл.6

I	2	3	4	5	6	7	8	9	I0
	R	3-7-9	IIOOOIII	+	+	+	+	IOIIOOIO	R
	S	3-8-9	IIOOIOOO	+	+	+	+	IOIIOOII	S
	T	I2-II-2	OOIIOOIO	+	+	+	+	IOIIOIOO	T
	U	0-I-8-8-9	OIOOIOOI	+	+	+	+	IOIIOIOI	U
	V	0-2-8-8-9	OIOOIOIO	+	+	+	+	IOIIOIIO	V
	W	I-2-8-8-9	IIOOIOII	+	+	+	+	IOIIOIII	W
	X	I2-II-5	IOIIOIOI	+	+	+	+	IOIIIOOO	X
	Y	I2-II-8	IOIIOOII	+	+	+	+	IOIIIOOI	Y
	Z	0-8-4-8-9	OIOOIIOO	+	+	+	+	IOIIIOIO	Z
	0	0	IOOOOOOO	+	+	+	+	OIOIOOOO	0
	I	I	OOOOOOOI	+	+	+	+	OIOIOOOI	I
	2	2	OOOOOOIO	+	+	+	+	OIOIOOIO	2
	3	3	IOOOOOII	+	+	+	+	OIOIOOII	3
Цифры	4	4	OOOOOIOO	+	+	+	+	OIOIOIOO	4
	5	5	IOOOOIOI	+	+	+	+	OIOIOIOI	5
	6	6	IOOOOIIO	+	+	+	+	OIOIOIIO	6
	7	7	OOOOOIII	+	+	+	+	OIOIOIII	7

Продолжение табл.6

I	2	3	4	5	6	7	8	9	I0
	8	8	0000I000	+	+	+	+	0I0II000	8
	9	9	I000I00I	+	+	+	+	0I0II00I	9
Плюс	+	0-2-8	I000I0I0	+	+	+	+	0I00I0II	+
Минус	-	0-3-8	0000I0II	+	+	+	+	0I00II0I	-
Дробная черта	/	0-4-8	I000II00	+	+	+	+	0I00IIII	/
Запятая	,	0-5-8	0000II0I	+	+	+	+	0I00II00	,
Точка	.	0-6-8	0000III0	+	+	+	+	0I00III0	.
Пробел	␣	0-7-8	I000IIII	-	-	-	-	0I000000	␣
Основание десятичной системы счисления	I0	I2	000I0000	-	-	-	-	-	I0
Стрелка вверх	↑	I2-0-I	I00I000I	-	-	+	-	0I000II00	↑
Круглая скобка левая	(	I2-0-2	I00I00I0	+	+	+	-	0I00I000	(
Круглая скобка правая	)	I2-0-8	000I00II	+	+	+	-	0I00I00I	)
Умножение	x	I2-0-4	I00I0I00	-	-	-	-	-	x
Равно	=	I2-0-5	000I0I0I	+	+	+	+	0I0II00I	=
Точка с запятой	;	I2-0-6	000I0II0	+	+	+	+	0I0II0II	;
Квадратная скобка левая	[	I2-0-7	I00I0III	+	+	+	-	I0III0II	[
Квадратная скобка правая	]	I2-0-8	I00II000	+	+	+	-	I0IIIIOI	]

Продолжение табл. 6

I	2	3	4	5	6	7	8	9	I0
Звездочка	*	I2-0-9	0001100I	+	+	+	+	0I00I0I0	*
Открывающая кавычка	‘	I2-2-8	0001I0I0	–	–	+	–	0I000I0	‘
Закрывающая кавычка	’	I2-3-8	I00II0II	–	–	+	–	0I000I0	’
Не равно	≠	I2-4-8	0001II00	–	–	+	–	0I000II	≠
Меньше	<	I2-5-8	I00III0I	+	+	+	–	0I0III00	<
Больше	>	I2-6-8	I00III10	+	+	+	–	0I0III10	>
Двоеточие	:	I2-7-8	0001III1	+	+	+	+	0I0I10I0	:
Подчеркивание	_	0-3-5-8-9	II00II0I	–	–	–	–	–	_
Меньше равно	≤	0-3-6-8-9	II00III0	–	–	–	–	–	≤
Больше равно	≥	0-3-7-8-9	0I00III1	–	–	–	–	–	≥
Стрелка вниз	∨	I2-8-9	II0I0000	–	–	+	–	I0III100	∨
Стрелка вверх	∧	I2-0-I-8-9	0I0I0001	–	–	+	–	I0III110	∧
Импликация	⊃	I2-0-2-8-9	0I0I00I0	–	–	–	–	000II0I0	⊃
Логическое "нет"	¬	I2-I-2-8-9	II0I00II	–	–	–	–	000II000	¬
Знак промежутка	+	I2-0-3-4-9	0I0I0I00	–	–	–	–	000III10	+
Тождества	≡	I2-0-3-5-9	II0I0I0I	–	–	–	–	000II001	≡
Процент	%	I2-0-3-6-9	1I0I0II0	+	+	+	+		%

Продолжение табл. 6

I	2	3	4	5	6	7	8	9	IO
Ромбик	◊	I2-0-8-7-9	OIOIOIII	–	–	+	–	OIOOOIOO	◊
Вертикальная черта	I	I2-0-8-8-9	OIOIIOOO	–	–	–	–	–	I
Горизонтальная черта	—	I2-I-8-8-9	IIOIIOOI	–	–	–	–	–	—
Подчеркивание	_	I2-2-8-8-9	IIOIIOIO	–	–	+	–	IOIIIIII	_
Восклицательный знак	!	I2-0-I-2-8-9	OIOIIOII	+	+	+	–	OIOOOOOI	!
Кавычки прямые	"	–	–	+	+	+	–	OIOOOOIO	"
Номер	≠	–	–	+	+	–	–	OIOOOOII	≠
Знак денежной единицы	¤	–	–	+	+	–	–	OIOOOIOO	¤
Коммерческое "И"	&	–	–	+	+	–	–	OIOOOIIOO	&
Пропуск (пробел)		Отсутствие пробивок	IIIIIIIO	+	+	+	+	OIOOOOOO	
Возврат каретки		–	OIIIIIOO	+	+	–	–	OOOOIIOI	
Замена		–	–	+	–	–	–	OOOIIOIO	
Аннулирование		–	–	+	–	–	–	OOOIIOOO	

Язык управления заданием

3. ЯЗЫК УПРАВЛЕНИЯ ЗАДАНИЕМ

3.1. Управляющие операторы задания.

Для выполнения вычислительных работ в системе необходимо определенным образом оформить задание для работы машины. Задание может состоять из одного или нескольких шагов. В каждом шаге задания выполняется некоторая программа. Шаги могут быть связаны между собой, т.е. результаты работы предыдущего шага используются в последующих шагах.

Любое задание и шаг задания описывается с помощью специальных карт - управляющих операторов задания. Одно или несколько заданий, подготовленных к работе, образуют входной поток заданий.

Имеется пять управляющих операторов задания.

Оператор задания (мнемоническое обозначение ЗДН).

Оператор выполнения (ВПЛ)

Оператор определения данных (ОД).

Разделительный оператор (не имеет мнемоники).

Ограничительный оператор (не имеет мнемоники).

Анализ и обработка управляющих операторов выполняется программой "диспетчер заданий".

Параметры, записываемые в управляющих операторах, позволяют диспетчеру заданий влиять на выполнение задания и отдельных его шагов, распределять ресурсы ввода-вывода, осуществлять связь с оператором.

Управляющие операторы задания записываются на тех же бланках, на которых пишутся программы на мнемокоде. С них они переносятся на носители (обычно перфокарты), с которых вводятся в систему. В дальнейшем при описании управляющих операторов мы будем всегда ссылаться на позиции карты, не уточняя в каждом конкретном случае на бланке или карте записан оператор. Управляющие операторы

записываются в соответствии с определенными правилами, описанными ниже.

Признаком управляющих операторов задания являются символы // (дробная черта) в I-й и 2-й позициях карты, разделительный оператор имеет в этих позициях символы /*

Управляющие операторы содержат 4 поля: поле названия, поле операции, поле операнда и поле комментария. В некоторых операторах отдельные поля могут не использоваться.

В таблице 7 дается использование полей управляющих операторов

Поля отделяются друг от друга одним или несколькими пробелами Границы полей на картах не фиксированы.

В поле названия указывается имя, присвоенное данному оператору. Это идентификатор, содержащий не более 8 символов. Название должно записываться начиная с 3-й колонки карты.

В поле операции указывается тип управляющего оператора, заданный своей мнемоникой.

Использование полей в управляющих операторах задания

Таблица 7

Символы в I и 2 колонках карты	Поле названия	Поле операции	Поле операнда	Поле комментария
I	2	8	4	5
//	Обязательно	ЗДН	Не обязательно	Не обязательно
//	Не обязательно	ВПЛ	Обязательно	Не обязательно
//	Обязательно	ОД	Обязательно	Не обязательно
/*				
//	Пробелы	Пробелы	Пробелы	Пробелы

Примечание. Информация в остальных позициях разделительного оператора (за исключением I-2 колонок) не анализируется.

Разделительный и ограничительный операторы не имеют мнемонических обозначений. Поле операнда содержит параметры управляющих операторов. Параметры отделяются друг от друга запятыми. После последнего параметра запятая не ставится. Пробелы в поле операнда не допускаются. Поле комментария может содержать любую описательную информацию.

Вся информация записывается в колонках с I по 7I. Если информация занимает более 7I символа, то она продолжается на другую карту следующим образом:

а) последний параметр, помещаемый на первой карте должен быть записан полностью, включая запятую после него, в колонках включительно по 7I. Если запятая помещается не в 7I-й колонке, то все оставшиеся колонки, включая 7I-ю, заполняются пробелами;

б) в 72-й колонке нужно записать символ продолжения (любой отличной от пробела);

в) в I-2 колонках карты-продолжения записываются символы //, в колонках 3-I5 должны быть пробелы;

г) с I6-й колонки записываются последующие параметры поля операнда.

Продолжение информации, относящейся к комментариям, производится аналогично описанному выше, за исключением того, что в первой карте до 72-й колонки при записи комментария может быть любая информация, и на картах продолжения информация может быть записана с любой колонки после I5-и.

Количество карт-продолжения для управляющих операторов не ограничивается.

При написании управляющих операторов используются русские и латинские буквы, цифры и специальные символы. Перечень применяемых символов приведен в табл. 8.

Таблица 8

Название символа	Графическое изображение	Название символа	Графическое изображение
Буквы русского алфавита	А	Буквы русского алфавита	П
	Б		Р
	В		С
	Г		Т
	Д		У
	Е		Ф
	Ж		Х
	З		Ц
	И		Ч
	Й		Ш
	К		Щ
	Л		Ы
	М		Ь
	Н		Э
	О		Ю
			Я
Буквы латинского	D	Запятая	,
	F	Точка	.
	G	Дробная черта	/
	I	Открывающая кавычка	'
	J	Открывающая скобка	(
	L		
	N		

Продолжение табл.8

Название символа	Графическое изображение	Название символа	Графическое изображение
алфавита	Q R S V W Z U	Закрывающая скобка	)
		Звездочка	*
		Плюс	+
		Минус	-
		Равно	=
		Пробел	␣

Примечание. В поле комментария и в параметре ПАРМ оператора ВПЛ может быть записана любая информация, определенная приложением "алфавитно-цифровые символы и их кодировка". (См.приложение 2, стр.103).

При описании управляющих операторов приняты следующие обозначения и соглашения.

Информация, написанная прописными буквами, а также числа и специальные символы переносятся без изменения в реальные управляющие операторы (на карты).

Информация, написанная строчными (малыми) буквами является переменной и на картах заменяется информацией, определенной программистом.

Квадратные и фигурные скобки используются для придания информации, заключенной в них определенных признаков. На картах эти скобки не записываются.

Информация, заключенная в квадратные скобки, является не обязательной и может вводиться и исключаться по усмотрению программиста.

Информация, заключенная в фигурные скобки, содержит выборочные элементы. Это значит, что на карте программист указывает только один из них по своему выбору. Если один из выборочных элементов подчеркнут, то этот элемент считается автоматическим выбором, т.е. выбирается системой, если ни один из выборочных элементов не задан в управляющем операторе.

Круглые скобки используются для записи списков подпараметров. Круглые скобки вместе со списком подпараметров записываются и на картах.

Параметры в поле операнда могут быть двух типов - позиционные и ключевые.

Позиционные параметры характеризуются своим местом относительно друг друга и записываются в строго определенном порядке (т.е. на первом месте всегда стоит I-й позиционный параметр, на втором - 2-й и т.д.). Отсутствие позиционного параметра указывается запятой, записанной на его место.

Например, запись

$$P_1, P_2,,P_4\cdots$$

указывает, что отсутствует 3-й позиционный параметр, а запись

$$,P_2, P_3\cdots$$

указывает на отсутствие I-го параметра.

Однако, если отсутствуют один или несколько последних позиционных параметров или все позиционные параметры, то запятые, указывающие их отсутствие, могут не ставиться.

Ключевые параметры являются позиционно-независимыми, т.е. записываются в произвольном порядке. Вид параметра характеризуется определенным названием (ключом), а содержательная часть ключевого параметра является переменной или выборочной и записывается программистом. Ключ объединяется с содержательной частью знаком равенства.

Например, K_1 = XXXX, K_2 = XXXX, K_5 = XXXX

Поскольку ключевые параметры не зависят от места, их отсутствие не указывается запятой.

Позиционные параметры в поле операнда записываются первыми, а за ними записываются ключевые параметры.

Как позиционные параметры, так и содержательнаяч часть ключевых параметров могут представляться списком подпараметров. При этом список подпараметров заключается в круглые скобки. Подпараметры в списке могут быть как позиционными, так и ключевыми. Позиционные подпараметры в списке также стоят первыми.

Понятие "пробел" имеет двоякий смысл:

а) как специальный символ ␣ ;

б) отсутствие пробивок на перфокарте.

В дальнейшем, если это не оговорено особо, понятие пробела будет подразумеваться в смысле б).

3.2. Назначение отдельных управляющих операторов.

Оператор ЗДН. В этом операторе может быть задан приоритет, который программист присваивает заданию, а также форма сообщений, выдаваемых системой программисту.

Оператор ВПЛ. В этом операторе указывается название программы, которая должна быть выполнена, а также может задаваться информация, передаваемая выполняемой программе (если это требуется), условия для пуска оставшихся шагов задания, предельное время выполнения данного шага задания.

Оператор СД. Содержит информацию о файле данных, который вводится или выводится, об устройстве, с которого вводится или на которое выводится файл, о носителе, на котором хранится файл, и другие сведения.

Разделительный оператор используется для отделения данных,

которые вводятся с того же устройства, что и управляющие операторы задания, от последующих управляющих операторов.

Ограничительный оператор используется для указания конца входного потока.

Ниже дается описание параметров управляющих операторов задания и правила их использования.

3.2.I. Оператор ЗДН. Оператор ЗДН (рис.I6) является первым в задании. Он должен содержать название задания в его поле названия и слово ЗДН в поле операции. Все параметры в поле операнда необязательные.

Название	Операция	Операнд
//имяздн	ЗДН	[ПРТ = приоритет задания] [УРС = {0 / I}]

Рис.I6. Операнд ЗДН

Имяздн указывает имя данного задания, которое используется диспетчером заданий. Оно должно быть идентификатором, содержащим не более восьми символов. Имя задания в операторе ЗДН является обязательным параметром.

Параметр УРС. Программа диспетчер заданий при обнаружении ошибок в управляющих операторах выдает сообщения на устройство, назначенное оператором при запуске задания. При этом, если параметр УРС=0, или отсутствует в операторе ЗДН, то выдается только сам оператор ЗДН, управляющие операторы, в которых обнаружены ошибки и сообщения об этих ошибках.

Если управляющий оператор нанесен на нескольких картах и

ошибка обнаружена на карте-продолжении, то выдается только содержимое этой карты, а не весь оператор.

Если УРС=I, то кроме того, выдаются все управляющие операторы данного задания.

Параметр ПРТ служит для назначения данному заданию приоритета. Для этого в реальном операторе XX заменяется числами от 0 до I5 (высший приоритет I5). Если ПРТ отсутствует, то заданию назначается стандартный приоритет.

3.2.2. Оператор ВПЛ. Оператор ВПЛ (рис.I7) является первым оператором в каждом шаге задания. Он должен содержать слово ВПЛ в поле операции и должен быть расположен во входном потоке перед оператором ОД и данными, относящимися к шагу.

Основной функцией ВПЛ является определение программы, которая должна быть выполнена.

Название	Операция	Операнд
//имяшага	ВПЛ	Ключевые параметры: {ПРТ = название программы ПРТ = № . имяод} [ПАРМ ='XXX...'] [УСЛ = (код,указатель)] [ВРЕМЯ = (минуты,секунды)]

Рис.I7 Оператор ВПЛ

Имяшага. Это идентификатор, определяющий шаг задания.

ПРТ - параметр, определяющий программу, которая должна быть выполнена. Эта программа может быть задана двумя способами в зависимости от того, является ли она программой из системной

библиотеки, или она хранится как файл данных на одном из внешних носителей.

Если программа, которую требуется выполнить в данном шаге, находится в системной библиотеке, то необходимо записать

ПРГ = название программы,

где вместо названия программы указывается идентификатор, определяющий эту программу в системной библиотеке.

Если программа является файлом данных, то для ее выполнения необходимо записать

ПРГ = *.имяод

т.е. в правой части (после звездочки с точкой) указывается идентификатор того оператора ОД, который описывает файл данных (программу).

Выполнение отдельных шагов, как правило, связано между собой, т.е. необходимость выполнения последующих шагов зависит от правильности выполнения текущего шага.

УСЛ - параметр, с помощью которого задаются условия, при которых выполнение последующих шагов задания нецелесообразно и они должны быть пропущены.

Для определения условий пропуска шагов задания должен быть записан параметр УСЛ = (код, указатель). Вместо слова "код" должно быть написано число от 0 до 4095, а вместо слова "указатель" одно из следующих выражений:

БЛ (больше)

БР (больше или равно)

РВ (равно)

МН (меньше)

МР (меньше или равно)

НР (не равно)

Например, УСЛ = (20, БР) означает, что если код возврата текущего шага задания будет больше или равен числу 20, то оставшиеся шаги задания будут пропущены.

Допускается до восьми параметров УСЛ в одном операторе ВПЛ. Если опускается параметр, то никаких проверок не будет.

Для обрабатывающих программ, которые требуют управляющей информации во время выполнения, в операторе ВПЛ имеется параметр ПАРМ, с помощью которого можно передать информацию.

Для передачи информации программе необходимо записать

ПАРМ= 'информация'

Здесь между кавычками можно поместить до 40 символов любой информации. При этом, если в передаваемой информации содержатся кавычки, то каждая из них должна быть записана как пара кавычек.

Например, для передачи программе информации

РЕЖИМ 'А'

необходимо записать

ПАРМ = 'РЕЖИМ''А'' '

Программа определяет адрес, по которому хранится переданная информация по содержимому регистра I.

Для ограничения времени выполнения шага задания можно назначить максимальное время, по истечении которого шаг задания будет закончен. Для этого должен быть записан параметр

ВРЕМЯ = (минуты, секунды)

Вместо слов "минуты" и "секунды" должно быть подставлено максимальное число минут и секунд, отведенных для шага задания. Число минут не может превосходить 1439, число секунд - 59.

Если шаг задания не закончился за это время, то его выполнение прекращается и все оставшиеся шаги задания будут пропущены, то есть задание будет закончено. Если параметр ВРЕМЯ опускается, то автоматически устанавливается стандартный предел времени.

Если предел времени дается только в минутах, то можно не записывать круглых скобок.

Например, ВРЕМЯ = 5, означает время выполнения шага равно 5 минутам.

Если предел времени дается только в секундах, то должна быть написана запятая для указания отсутствия минут, например, ВРЕМЯ = (,45), означает, что время выполнения равно 45 секундам.

Максимальный интервал времени для шага задания является 24 часа, то есть 1440 минут. Если программист указал больший интервал, то система устанавливает интервал, равный 24 часам.

3.2.3. Оператор ОД.

Файл данных, используемый обрабатывающей программой, должен быть описан во входном потоке оператором ОД (рис.18). Оператор ОД, относящийся к определенному шагу задания, должен содержать имяод в поле названия и слово ОД в поле операции. Все параметры в поле операнда не обязательны, однако, отсутствие всех параметров является недопустимым.

Название	Операция	Операнд
//имяод	ОД	[*] [ИМЯ ФД = идентификатор] [УСТР = (список подпараметров)] [ТОМ = (список подпараметров)] [БУД = (список подпараметров)] [МЕТКА =(список подпараметров)] [{РАСП =(список подпараметров) {СВЫВОД = А]

Рис. 18

Если в данном шаге задания требуется обработка нескольких файлов данных, то каждый из них должен быть описан своим оператором ОД. При этом управляющие операторы ОД могут располагаться в произвольном порядке после оператора ВПЗ за исключением оператора ОД, содержащего в поле операнда символ *, который должен быть среди них последним.

Оператор ОД используется для задания основных характеристик файла данных, устройства ввода-вывода, носителя и его расположения. Это позволяет программисту не заводить эти характеристики при разработке программы и наиболее полно использовать ресурсы системы по вводу-выводу с учетом конкретного состояния системы непосредственно перед запуском программы на выполнение. Ниже дается описание полей и параметров оператораОД.

1. Имяод - идентификатор, содержащий не более 8 символов, используемый для ссылок на данный оператор ОД.

В одном шаге задания не должно быть нескольких операторов ОД с одним именем.

2. Параметр * (звездочка) единственный позиционный параметр. Если в операторе указан этот параметр, то никакие другие параметры не должен быть указаны. Параметр звездочка служит для сообщения системе о том, что за данным оператором ОД следуют не управляющие операторы, а файл данных, который должен быть введен обрабатывающей программой (с того же устройства, с которого вводится входной поток). Для определения конца файла данных в конце его должен быть разделительный оператор (/*).

В одном шаге задания может быть только один файл данных, вводимый с входного потока. Его данные не должны содержать сочетания символов /* в первых двух позициях карты.

3. ИМЯОД - идентификатор, содержащий до 8 символов, служит

для опознания системой файла, с которым должна работать программа, а также для создания и проверки метки файла для файла со стандартными метками.

Запись параметра для файла с названием УЧЕТ

ИМЯ ФД = УЧЕТ

Параметр УСТР используется для задания устройства ввода-вывода, которое требуется для работы с файлом.

4. УСТР = (({адр. / тип} [,ОТСРЧ] ОБЩУ = имя од))

Приведенная запись означает, что если указывается параметр УСТР, то должен быть сделан один из двух выборов. При первом выборе (верхняя строка) программист должен обязательно задать устройство, указывая его адрес или тип (по выбору) и не обязательно задавать подпараметр ОТСРЧ. Запятая перед ОТСРЧ в квадратных скобках означает, что она ставится только в том случае, когда указывает ОТСРЧ, и не ставится, когда этот подпараметр отсутствует.

При втором выборе обязательно указывать подпараметр ОБЩУ = = имяод.

Параметр УСТР не требуется указывать, если:

- файл данных был определен и использовался в предыдущем шаге и был передан для работы в текущем шаге (с помощью параметра РАСП см.ниже);

- файл данных записывается на том магнитной ленты, который является общим для нескольких файлов в данном шаге.

Можно определить информацию об устройстве двумя способами

а) указывая определенное устройство либо группу устройств. В этом случае устройства могут быть идентифицированы их адресом или типом.

Идентифицировать устройство его адресом можно, записав параметр УСТР = адрес, заменяя слово"адрес" на трехсимвольный

адреи устройства (адреса устройств определяются при создании системы).

Пример.

УСТР = (I80, ОТСРЧ)

или УСТР = I80,

где I - номер канала,

80 - номер устройства ввода-вывода.

Для задания типа устройства необходимо записать параметр УСТР = тип, заменяя слово "тип" на определенный тип устройства, согласно таблице 9.

Пример.

УСТР = (A2320, ОТСРЧ)

или УСТР = A2320

где A2320 определяет устройство ввода с перфокарт;

б) требуя то же устройство, которое используется другим файлом данных в том же шаге задания.

Для экономии количества устройств, используемых в шаге задания, можно потребовать, чтобы файлу данных было назначено то же устройство, как и файлу данных, ранее определенному в том же шаге задания (родственность устройств). Этим указывается, что файлы данных последовательно используют одно и то же устройство.

Для требования родственности устройств должен быть записан параметр (УСТР = ОБЩУ = имяод), заменяя слово имяод на название ранее определенного оператора ОД в том же шаге задания. Этому файлу данных будет назначено устройство, как и файлу данных, определенному оператором ОД с названием имяод.

Если необходимо, чтобы том был установлен не перед запуском задачи, а во время ее выполнения (отсроченный монтаж) то в параметре УСТР записывается подпараметр ОТСРЧ. В этом случае о необходимости установки тома на пульт оператора выдается соответствующее

ТИПЫ УСТРОЙСТВ ВВОДА-ВЫВОДА

Таблица 9

Тип	Название устройства
A2III	устройство ввода с перфоленты
A2II2	устройство вывода на перфоленту
A22I0	накопитель на магнитной ленте
A2320	устройство ввода с перфокарт
A2330	устройство ввода-вывода с перфокарт
A27II	устройство записи информации
A32I0	устройство алфавитно-цифровой печати
A32II	устройство алфавитно-цифровой и графической регистрации
A3220	устройство алфавитно-цифровой регистрации
A33I0	пульт оператора-технолога
A34I0	пульт программиста
A42I0	устройство ввода-вывода информации по телеграфным каналам
A5II0	устройство ввода дискретной информации
A52I0	устройство аналого-цифрового преобразования
A532I	устройство ввода частотных сигналов
A53II	устройство кодового управления
A532I	устройство аналогового управления
A5330	групповое устройство релейного и аналогового управления
A55I0	групповое устройство ввода аналоговых и дискретных сигналов
A2II0	устройство ввода-вывода на перфоленту
A24II	накопитель на магнитном барабане
A5I20	преобразователь кодов интегрирующий

сообщение. При указании УСТР = СВНУ =имяод подпараметр СТСРЧ не ставится, но система автоматически подразумевает отсроченный монтаж;

5. Параметр ТОМ обеспечивает программу „диспетчер заданий" информацией о томе, с которого читается или на который записывается информация в данном шаге задания.

Операционная система различает три состояния томов на устройстве:

- постоянно установленные;
- резервные;
- съемные.

Каждый том находится в одном из 3-х состояний. Постоянно установленным является том, с которого загружается операционная система (СИСОС) и том, содержащий системную библиотеку (СИСБИБ). Состояние постоянно установленных томов назначается при генерации системы.

Состояние "резервный" тому назначается с помощью команды оператора РЕЗЕРВ. Резервный том остается установленным до тех пор, пока не будет выдана оператором команда СНЯТЬ. Съемными являются тома, которые не являются постоянно установленными или резервными.

$$\text{ТОМ} = \left(\left[\text{ОСТВ}\right]\ \left[\left\{\begin{array}{l}\text{ПРНТ = порядковый номер тома}\\ \text{ОНМТ =имяод}\end{array}\right\}\right]\right)$$

Параметр ТОМ позволяет:

- идентифицировать том;
- требовать, чтобы том оставался на устройстве после выполнения шага для дальнейшего использования.

Идентифицировать том можно либо задавая порядковый номер тома в виде

ТОМ = (ОСТВ, ПРНТ =порядковый номер тома)

или ТОМ = ПРНТ = порядковый номер тома,

где вместо порядкового номера тома указывается название, содержащее до шести алфавитно-цифровых символов, либо используя том, который был определен раньше для другого файла в том же шаге. Записывается это в виде

ТОМ = (ОСТВ, ОБЩТ =имяод)

или ТОМ = ОБЩТ =имяод ,

где имяод заменяется названием оператора ОД определенного ранее.

Требование оставить том на устройстве задается параметром ОСТВ. В этом случае по окончанию шага на пульт оператора не выдается сообщение о состоянии тома.

Примечание. Запрещается использовать в качестве порядкового номера тома слово СВВОД, поскольку оно используется системой.

6. Параметр БУД используется для передачи недостающей информации блоку управления данными, который формируется в потребительской программе и дополняется из оператора ОД во время запуска шага задания. Ниже приводятся подпараметры, которые могут задаваться как в параметре БУД оператора ОД, так и при разработке программы (с помощью макрокоманды БУД).

ВЫБОР - определяет действия системы при обнаружении ошибки.

ДЛБУФ - размер буфера (области для ввода-вывода с главной памяти) в байтах.

ДЛБЛОК - максимальный размер блока (зоны) в байтах.

КЛБУФ - количество используемых буферов.

ФОРЗП - формат записей.

РЕЖПЧ - режим печати

РЕЖПЛ - режим перфорации

Все подпараметры ключевые.

Подробное описание этих подпараметрово дано в описании макрокоманды БУД (см.стр. 71)

7. МЕТКА = ([последов.номер файла] [[СМ / НМ]] [[ПДСХР = КККК / ДТОКЧ = ГТДДД]])

Параметр МЕТКА указывает положение файла на магнитной ленте, тип метки, период сохранения файла.

Положение файла на магнитной ленте задается последовательным номером файла, который задается десятичным числом от I до 9999 (первый позиционный подпараметр). Если номер файла равен I, то он может не указываться, но его отсутствие должно отмечаться запятой. Если номер файла указан равным нулю, то система сама присваивает ему номер I, второй позиционный подпараметр указывается как СМ (если файл имеет метку), или НМ (если файл не имеет метки). Тип метки может быть и не указан, в этом случае система считает, что файл с метками.

Период сохранения файла на магнитной ленте может быть задан в виде

ПДСХР = КККК

где КККК - число дней, в течение которых требуется сохранить данные, либо задавая дату, когда может быть уничтожен файл в виде

ДТОКЧ = ГТДДД

где ГТ - две последние цифры года

ДДД - число дней от начала года (не более 365-366).

Примеры записи.

МЕТКА = 2 - означает, что файл имеет метку, и последовательный номер файла равен 2.

МЕТКА = (,НМ) - означает, что файл не помечен и последовательный номер файла равен I

МЕТКА = (5, ПДСХР = 20) - означает, что файл - 5, период сохранения - 20 дней

3.2.4. Порядок выполнения задания в системе и примеры записи управляющих операторов задания.

Подготовленное задание в виде карт управляющих операторов задания закладывается в устройство чтения перфокарт и по команде

оператора СТАРТ задание запускается на выполнение. При этом загружается программа "диспетчер заданий", находящаяся в системной библиотеке, и ей передается управление.

Диспетчер заданий является набором программ, обеспечивающих управление прохождением и выполнением задания в системе.

Получив управление, диспетчер заданий производит чтение и анализ управляющих операторов задания, закрепляет необходимые устройства ввода-вывода, подготавливает необходимые блоки, регистры, освобождает память для выполнения программы, обеспечивает загрузку и запуск программы.

Потребительская программа загружается в память на то же место, что и программа "диспетчер заданий", так что после окончания выполнения программы производится автоматическая загрузка диспетчера заданий в главную память с системной ленты.

Шаги задания выполняются последовательно в том порядке, в каком они стоят на входном потоке.

По окончании шага или всего задания программа "диспетчер заданий" производит освобождение устройств, закрепленных за данным шагом.

Если система прекратила выполнение шага задания из-за нарушений возникших в системе, то диспетчер заданий так же освобождает устройства и шаг задания завершается аварийно. При аварийном завершении шага или, если при его выполнении получены коды, совпадающие с кодами параметра УСЛ в операторе ВПЛ, то диспетчер заданий пропускает оставшиеся шаги в данном задании без выполнения и, если во входном потоке есть еще задания, начинает выполнять следующее задание.

После окончания выполнения всех заданий, если в конце последнего задания стоит ограничительный оператор, диспетчер заданий освобождает память, устройства, закрепленные за шагом, приводит

систему в исходное состояние и выдает сообщение об окончании работы входного потока. После этого система может быть запущена вновь командой СТАРТ.

Для работы входного потока ряд устройств являются системными, т.е. такими, наличие которых всегда предполагается и не требуется их назначения и закрепления. Системные устройства определяются при разработке операционной системы и ее генерации.

В стандартных генерациях системы в качестве системных приняты:

- для размещения тома системной библиотеки (СИСБББЛ) - ленто-протяжный механизм с номером ноль (ЛПМС);

- для запуска входного потока и выдачи системных сообщений - пульт программиста, который в этом случае служит пультом оператора системы;

- для выдачи результатов работы и сообщений потребителю - устройство алфавитно-цифровой печати (УАЦП);

- для чтения управляющих операторов задания и файлов данных, вводимых из входного потока - устройство ввода перфокарт (УВвК).

Если по каким-либо причинам нельзя использовать назначенное УВвК для чтения входного потока, то можно вводить управляющие операторы задания с другого устройства. Для этого при запуске входного потока в команде СТАРТ нужно указать другое устройство, которое на данный период работы становится системным. Аналогичное изменение нужно сделать, если необходимо использовать другое устройство для выдачи результатов потребителю.

Каждое задание определяется набором управляющих операторов, помещаемых между двумя операторами ЗДН. Первый оператор ЗДН характеризует начало задания. Второй оператор ЗДН характеризует окончание предыдущего задания и начало нового задания. Если задание является последним во входном потоке, то оно заканчивается

ограничительным оператором. Каждый шаг задания определяется набором управляющих операторов помещаемых или между двумя операторами ВПЛ, или между оператором ВПЛ и оператором ЗДН (последний шаг в задании), или между оператором ВПЛ и ограничительным оператором (последний шаг в задании, являющемся последним заданием во входном потоке).

3.2.5. Пример. Ниже приводится пример оформления задания с использованием описанных выше возможностей параметров управляющих операторов задания.

Задание состоит из 7 шагов, из которых пять - трансляция программ, записанных на мнемокоде, шестой шаг - объединение (компоновка) транслированных программ в одну программу и ее выполнение - 7-й шаг. Поскольку шаги 2-5 полностью аналогичны шагу I; их описание не приводится.

Шаг первый

```
// ЗАДАНИЕ3  ЗДН  УРС=I
//ШАГI  ВПЛ  ПРГ=ИЕУМКД, УСЛ=(4,РВ), УСЛ =(I6,РВ)
//СИСРУСТI  ОД  УСТР=II2, ТОМ =(ОСТВ, ПРНТ =ТОМI), МЕТКА=(,НМ)
//СИСРУСТ2  ОД  ТОМ = (ОСТВ, ОБЩТ = СИСРУСТI),МЕТКА =(,НМ)
//СИСВЫВДМ ОД УСТР = II3, ТОМ = (ОСТВ, ПРНТ =ТОМ2),МЕТКА=(I,СМ),
                                                  ИМЯФД=РЕЗЛТI
//СИСПЕЧТЬ  ОД  СВЫВОД=А
//СИСВВОД  ОД  *
```

I. В операторе ЗДН указывается, что требуется выполнить работу с названием задание 3. При возникновении ошибок выдавать на печать все управляющие операторы (УРС=I).

2. В операторе ВПЛ указывается, что требуется выполнить шаг задания с названием ШАГI. В этом шаге выполняется программа из библиотеки (ИЕУМКД - транслятор с мнемокода). Условия пропуска шагов задания - при кодах возврата равных 4 и I6.

3. Оператором ОД с названием СИСРУСТ1 задается устройство (ПЛ2 - магнитная лента), на которое выводятся результаты трансляции (первой прогонки) и том (порядковый номер тома - ТОМ1). В параметре МЕТКА указано, что вводимый файл является первым на томе с непомеченными файлами.

4. Оператором ОД с названием СИСРУСТ2 задается устройство и том, с которого вводятся результаты первой прогонки. Естественно, что они вводятся с того же устройства и тома, на который выводились. Поэтому в параметре ТОМ дано ОБЩТ =СИСРУСТ1 (родственность томов), указанием подпараметра ОСТВ задается требование сохранения тома для последующего использования. В параметре МЕТКА указывается та же информация, что и в предыдущем операторе ОД.

5. Оператором ОД с названием СИСВЫВД задается устройство, на которое выдаются результаты 2-й прогонки (ПЗ - магнитная лента), том - порядковый номер тома - ТОМ 2 и требование его сохранения для последующего использования (ОСТВ). В параметре МЕТКА задается номер файла, который выводится и указывается, что файл должен иметь метку (СМ). В параметре ИМЯФД указывается, что название файла - РЕЗЛТ1.

6. Оператором ОД с названием СИСПЕЧТЬ задается устройство, на которое выдаются результаты трансляции. Параметром СВЫВОД = А задается системное устройство, которое определено при генерации системы (в данном случае УАЦП).

7. Оператором ОД с названием СИСВВОД задается устройство, с которого вводятся данные на трансляцию. Параметр * указывает, что данные вводятся с системного устройства, с которого вводятся управляющие операторы задания. В этом случае говорят, что данные находятся во входном потоке. Системное устройство для входного потока определяется при генерации системы.

Шаг шестой

```
//ШАГ6  ВПЛ  ПРГ=ИЕВКОМПН, ПАРМ='ПЕЧАТЬ', УСЛ = (4,РВ)
//СИСВХОД1 ОД ИМЯФД =РЕЗЛТ1, УСТР = 113, ТОМ = ПРНТ =ТОМ2, *
//            МЕТКА = (1,СМ)
//СИСВХОД2 ОД ИМЯФД =РЕЗЛТ2,ТОМ = ОБЩТ = СИСВХОД1, МЕТКА=(2,СМ)
//СИСВХОД3 ОД ИМЯФД=РЕЗЛТ3, ТОМ =ОБЩТ=СИСВХОД1, МЕТКА=(3,СМ)
//СИСВХОД4 ОД ИМЯФД=РЕЗЛТ4, ТОМ=ОБЩТ=СИСВХОД1, МЕТКА =(4,СМ)
//СИСВХОД5 ОД ИМЯФД=РЕЗЛТ5, ТОМ=ОБЩТ=СИСВХОД1,МЕТКА =(5,СМ)
//СИСРУСТ1 ОД ТОМ=ПРНТ=ТОМ1, МЕТКА =(,НМ), УСТР =112
//СИСВЫВДК ОД ТОМ=ОБЩТ=СИСВХОД1, МЕТКА = (1,СМ),РАСП =(, ПРД)
                                        ИМЯФД = РЕЗЛТ
//СИСПЕЧТЬ ОД СВЫВОД = А
//СИСВВОД ОД *
```

Оператор ВПЛ с названием ШАГ6 указывает, что вд данном шаге должна быть выполнена программа компоновщтк (ИЕВКОМПН), которая хранится в системной библиотеке. Этой программе должна быть передана информация ПЕЧАТЬ (параметр ПАРМ = 'ПЕЧАТЬ'). Условия пропуска последующих шагов при коде возврата, равном 4.

Оператор ОД с названием СИСВХОД1 требует для работы устройство с адресом 113 (УСТР), том с порядковым номером ТОМ2, файл с названием РЕЗЛТ1 (ИМЯФД). Этот файл имеет метку (СМ) и последовательний номер 1 (параметр МЕТКА). Данный управляющий оператор ОД записан на двух картах. В 72 позиции записан символ * (не пробел) и в карте продолжения информации записана, начиная с 16 позиции.

Операторы ОД с названием СИСВХОД2, СИСВХОД3, СИСВХОД4 и СИСВХОД5, аналогично предыдущему оператору ОД описывает свои файлы со своими последовательными номерами и метками размещенные на том же устройстве и ТОМе (поэтому у них отсутствует параметр УСТР и указывается родственность томов - ТОМ = ОБЩТ = СИСВХОД1.

Оператор ОД с названием СИСВЫВДК создан для вывода файла,

полученного в результате работы программы компоновщик с названием РЕЗЛТ (ИМЯФД) на то же устройство и тот же том, на котором записаны отдельные файлы, (например, ТОМ = ОБЩТ =СИСВХОДЖ) файл передается следующему шагу РАСП = (,ПРД).

Оператор ОД с названием СИСРУСТ1 описывает рабочую ленту для программы компоновщик. Задается устройство с адресом 112 (магнитная лента), том с порядковым номером ТОМ1. Информация записывается файлом без метки с последовательным номером 1 (параметр МЕТКА).

Оператор ОД с названием СИСПЕЧТЬ написан для выдачи результатов на печать. Задан единственный параметр СВЫВОД =А, требующий системное устройство.

Оператор ОД с названием СИСВВОД определяет устройство, с которого вводятся данные (здесь управляющие операторы компоновщика) Параметр * указывает, что эти данные вводятся с того же устройства, с которого вводится входной поток (системное устройство).

Шаг седьмой.

```
//ШАГ7   ВПЛ  ПРГ  = *.СИСВЫВДК
//СИСВЫВДК  ОД  ИМЯФД = РЕЗЛТ, РАСП =(СТ,УДЛ)
//СИСПЕЧТЬ  ОД  СВЫВОД=А
//
```

Оператор ВПЛ с названием шаг 7 указывает, что в данном шаге должна быть выполнена программа, находящаяся не в системной библиотеке, а на внешнем носителе как файл данных. Программа идентифицирована названием оператора ОД, описывающего этот файл данных (СИСВЫВДК).

Оператор ОД с названием СИСВЫВДК задает устройство, том и файл, с которого вводится программа. В данном случае параметр УСТР для задания устройства отсутствует, т.к. имеется параметр РАСП = (СТ, УДЛ), указывающий, что файл был передан из предыдущих

шагов (СТ), где бых определен и что после окончания шага этот файл не будет использован (УДИ). Используется файл с названием РЕЗЛТ. Непосредственного задания ТОМа и метки не требуется, эти параметры передаются из начального ОД.

Оператор ОД с названием СИСПЕЧТЬ указывает, что результаты должны быть выведены на системное устройство.

//. Ограничительный оператор указывает на окончание входного потока.

4. ЯЗЫК ОПЕРАТОРА СИСТЕМЫ

4.I. Общие сведения для оператора

Для управления системой оператор использует пульт оператора (ПО) и частично средства управления, имеющиеся на инженерном пульте (ИП).

Пульт оператора позволяет передавать в систему команды с одновременной печатью информации на бланке и выводить сообщения из системы на печать в виде алфавитно-цифровой информации. В качестве пульта оператора используется пульт программиста А34IO.

Назначение кнопок управления

I. "Аннулирование" - для аннулирования ненужного или ошибочного массива при вводе.

2. "Замена" - для замены ненужного или ошибочного символа при вводе.

3. "Запрос" - для передачи в вычислительный комплекс заявки нав ввод информации с пульта оператора.

4. "Конец ввода" - для сигнализации каналу об окончании ввода.

5. "Особый случай" - для извещения канала об окончании ввода в необычной ситуации.

6. "Сброс" - осуществляет общий сброс всей информации в устройстве управления пульта оператора.

Действия оператора и работа системы индицируются сигнальными лампочками.

"Запрос".

"Ввод".

"Вывод".

"Блокировка".

"Неисправность устройства".

"Потеря символа"

"Питание".

Инженерный пульт. Включение системы и все действия, связанные с начальным запуском, выполняются на инженерном пульте. Дальнейшее управление работой системы осуществляется через пульт оператора.

Для включения ВК необходимо на ИП нажать клавишу "Режим работы - Серия" и осуществить следующие действия:

Нажать кнопку "Сеть-Вкл".

Включение сети сигнализируется засвечиванием лампочки "Сеть включена".

Нажать клавишу "Система - Сброс".

Если лампочка "Ручное обращение" до нажатия была погашена, то после нажатия указанной клавиши она должна светиться. В любом случае сброс должен подтверждаться свечением лампочки "Ручное обращение".

Нажать клавишу "Ошибка - Сброс".

Подтверждением сброса индикаторов ошибок является погашенное состояние лампочки "Ошибка ВК".

Нажать клавишу "ВНП - Префикс I".

Лампочка "Префикс - Перемен" должна погаснуть.

Для начального ввода программы необходимо:

- на регистре П ИП набрать адрес УВВ (или адрес ячейки ПЗУ, в которой хранится информация об устройстве), с которого вводится начальная программа. В первом случае нулевой разряд регистра П должен быть сброшен. Во втором случае он должен быть установлен в "I".

- привести устройство, с которого вводится информация в рабочее состояние;

- нажать клавишу "ВНП - Ввод".

В течении ввода должна светиться лампочка "ВНП". По окончании ввода эта лампочка должна погаснуть. Если во время ввода была обнаружена ошибка, засвечивается лампочка "Ошибка ВНП". В последнем случае необходимо выяснить причину ошибки, устранить причину и повторить действия включения.

После выполнения ВНП процессор должен быть в состоянии останова, что подтверждается свечением лампочки "Ручное обращение".

4.2. Язык оператора системы

4.2.1. Общие сведения. Любая команда, вводимая с пульта оператора может иметь длину не более 150 байтов и состоит из трех полей: поля операции, поля операнда и поля комментария (последнее поле не обязательно). Поля отделяются друг от друга по крайней мере одним пробелом.

Поле операции содержит идентификатор выполняемой команды, содержащий до 8 символов.

Поле операнда содержит параметры команды, разделяемые запятыми. Различают позиционные и ключевые параметры. Позиционные параметры указываются всегда первыми и в определенном порядке. Ключевые параметры могут появляться в любом порядке. Они содержат ключевое слово, за которым следует знак равенства. Информация справа от знака равенства представляет собой значение параметра. Параметр может быть представлен списком подпараметров. В этом случае подпараметры разделяются запятыми и весь список заключается в круглые скобки.

Запятые, круглые скобки, пробелы, кавычки используются как разделительные символы. Информация, которая должна быть передана программе потребителя, заключается в кавычки. (Кавычки как смысловая информация кодируются двумя кавычками). Конец поля операнда

оператор должен указывать по крайней мере одним пробелом.

Система обозначений, используемая при описании форматов команд:

все символы, представленные в формате команды, за исключением фигурных и квадратных скобок, кодируются;

параметры внутри квадратных скобок являются необязательными. Они могут быть опущены по усмотрению оператора;

параметры внутри фигурных скобок являются выборочными (один из них должен быть указан оператором);

слова, написанные большими буквами, в реальных командах указываются без изменения;

слова, написанные маленькими буквами, оператор должен заменить соответствующим значением.

4.2.2. Команда СТАРТ запускает задачу, имя которой указано в поле операнда команды. По команде СТАРТ с названием ДЗДН в поле операнда запускается программа диспетчера заданий. Наличие параметров ВВОД, ВЫВОД или одного из них обеспечивает назначением устройств системного ввода-вывода для входного потока. Если эти параметры не указаны, используются системные устройства ввода-вывода, назначенные при начальном запуске системы.

Формат команды:

Операция	Операнд
СТАРТ	имя программы [,память] [,приоритет] [,ВВОД = (адрес [, прнт , пснф])] [,ВЫВОД =(адрес [, прнт, пснф])] [,ПАРМ "текст"]

где имя программы – идентификатор, определяющий название задачи, которая должна быть запущена для выполнения (количество символов не должно превышать 8);

память – числовой параметр, определяющий размер оперативной памяти (в килобайтах), необходимый для задачи. Количество символов I-2;

Если оператор будет опущен, задаче будет отведена вся динамическая область памяти.

приоритет – числовой параметр, определяющий приоритет задачи. Если этот параметр опущен, системой будет назначен стандартный приоритет (количество цифр I-2);

ВВОД, ВЫВОД – (имяустр, прнт, пснф) определяет системное устройство ввода-вывода для программы ДЗДН;

адрес – конкретный адрес устройства, назначенного для системного ввода-вывода(3 алфавитно-цифровых символа);

Параметры для магнитной ленты:

прнт – порядковый номер тома (до 6 алфавитно-цифровых символов);

пснф – числовой параметр, определяющий последовательный номер файла данных (I-4 символа);

ПАРМ = "текст" – смысловая информация, передаваемая программе. Этот параметр используется только при запуске задач реального времени и исключает использование параметров ВВОД и ВЫВОД.

Примеры.

СТАРТ ДЗДН

Оператор набирает команду для запуска входного потока без указания приоритета, памяти и системных устройств – при этом назначаются стандартные системные устройства, установленные при генерации, отводится вся свободная память динамической области

СТАРТ ПТО, 4, 4

Оператор запускает диспетчер реального времени перед запуском

входного потока, указывая приоритет задачи 4 и необходимую память на задачу - 4 килобайта.

СТАРТ ДЗДН, ВВОД = 00Р

У оператора при запуске входного потока возникла необходимость изменить системное устройство ввода, назначенное при генерации на устройство ввода-вывода и перфокарт (конкретный адрес 00Р).

4.2.3. Команда УСТАНОВ позволяет ввести в систему дату и время дня.

Формат команды:

Операция	Операнд
УСТАНОВ	ДАТА гг,ддд,ВРЕМЯ,=чч,мм,сс

где ДАТА гг,дд - 6-символьный числовой параметр, определяющий дату в следующем формате;

гг - год (00-99)

ддд - день (001-366)

ВРЕМЯ чч,мм,сс - 8-символьный числовой параметр, определяющий время в формате:

чч - часы (00-23)

мм - минуты (0-59)

сс - секунды (00-59)

Пример.

УСТАНОВ ДАТА = 70.115, ВРЕМЯ = 10.40,30

На сообщение системы:

ВВЕДИТЕ ДАТУ И ВРЕМЯ

Оператор вводит команду УСТАНОВ, где параметр ДАТА 70.115 указывает текущую дату ввода команды 25 апреля 1970 г. и текущее время 10 часов 40 минут 30 секунд

4.2.4. Команда ПЕРЕВОД дает возможность перевести устройство ввода-вывода в одно из состояний: внутреннее (ВНТ) или автономное (АВТН). Устройства в состоянии ВНТ могут назначаться системой для работы. Перевод устройства в состояние АВТН делает это устройство недопустимым для назначения диспетчером заданий.

Формат команды:

Операция	Операнд
ПЕРЕВОД	адрес {,АВТН / ,ВНТ}

где адрес – трехсимвольный конкретный адрес устройства ввода-вывода, состояние которого должно быть изменено (ВНТ или АВТН указывается, как записано)

В автономное состояние устройство будет переведено по окончании текущего задания, если в момент выдачи команды оно было назначено для работы диспетчером задания.

Пример.

ПЕРЕВОД II2, ВНТ

ЛПМ II2 переводится в состояние доступное для использования.

4.2.5. Команда ПЕЧАТЬ дает возможность выдавать на консоль название каждого выполняемого задания. С помощью этой команды можно также отпечатать текущее время.

Формат команды:

Операция	Операнд
ПЕЧАТЬ	{ИМЯЗДН / ВРЕМЯ}

где ИМЯЗДН – указывает, что название каждого задания должно быть

ть отпечатано при запуске задания и его окончании.

ВРЕМЯ – указывает, что должно быть отпечатано текущее время, оно печатается в следующем формате: ВРЕМЯ, чч, мм, сс.

4.2.6. Команда СТОП используется для отмены команды ПЕЧАТЬ.

Формат команды:

Операция	Операнд
СТОП	ИМЯЗН

4.2.7. Команда РЕЗЕРВ позволяет резервировать том и устройство ввода-вывода, указанное в поле операнда команды, для входного потока.

Формат команды:

Операция	Операнд
РЕЗЕРВ	адрес, ПРНТ = поряд.номер тома

где адрес – определяет трехсимвольный конкретный адрес устройства.

ПРНТ – порядковый номер тома – идентифицирует носитель (до 6 алфавитно-цифровых символов).

После получения команды РЕЗЕРВ система выдает сообщение о необходимости установить требуемый том на устройство. Оператор, устанавливает необходимый том на указанное в команде устройство и переводит устройство в состояние готовности переключением тумблера "авт" в рабочий режим. Система проверяет метку тома, и, если установлен не тот том, или неправильно сосчитана или записана метка тома, выдает сообщения об ошибке при РЕЗЕРВ.

Подтверждением выполнения команды является сообщение о резервировании указанного устройства.

Пример.

РЕЗЕРВ II2, ПРНТ =ТОМ1

Перед запуском входного потока при использовании несколькими заданиями тома с порядковым номером ТОМ1 оператор устанавливает этот том на свободное устройство (лентопротяжка II2) и указывает системе о резервировании этого устройства.

4.2.8. Командой СНЯТЬ оператор может потребовать от системы подготовить том к снятию с данного устройства. Снять данный том с устройства оператор имеет право лишь после получения соответствующего сообщения (сообщение будет выдано после завершения текущего задания).

Формат команды:

Операция	Операнд
СНЯТЬ	адрес

где адрес – определяет трехсимвольный конкретный адрес устройства, которое должно быть разгружено.

Пример.

СНЯТЬ II2

Оператор требует от системы подготовить устройство II2, резервированное ранее, к разгрузке по окончании задания. Система дает сообщение о возможности разгрузки устройства.

4.2.9. Команда ОТМЕНА используется для немедленного окончания задания, которое обрабатывается диспетчером заданий или выполняется.

Формат команды:

Операция	Операнд
ОТМЕНА	имя задания

где имя задания – определяет название задания, которое должно быть окончено. Максимальная длина названия 8 символов.

4.2.IO. Команда ОТВЕТ используется для ответа на сообщение из системы.

Формат команды:

Операция	Операнд
ОТВЕТ	ид, "текст"

где ид – двухсимвольный числовой параметр, приформированный системой слева к выданному сообщению с ответом, идентифицирует программу, запрашивающую ответ, и вводится оператором для опознания ответа системой.

"текст"– определяет информацию, поступающую в ответ на сообщение. Информация, переданная программе, не включает заключающие кавычки; максимальная длина текста I40 символов.

4.2.II. Команда ПАУЗА используется оператором для того, чтобы диспетчер заданий по окончании текущего задания закрыл входной поток, если даже текущее задание не является последним.

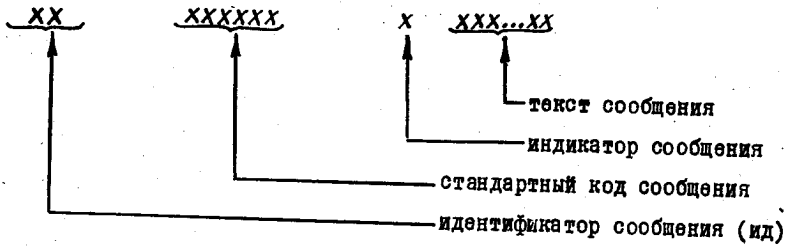
Формат команды:

Операция	Операнд
ПАУЗА	

4.3. Сообщение на консоль

Сообщение на консоль содержит до I28 символов.

Формат сообщения:



Идентификатор сообщения (ид) - двухсимвольный числовой параметр, приформированный системой слева к сообщению для индикации программы, выдавшей сообщение с ответом. Этот параметр вводится оператором с командой ОТВЕТ для опознания ответа системой. После принятия команды ОТВЕТ система может использовать этот идентификатор для другого сообщения с ответом.

Стандартный код сообщения - шестисимвольный код, идентифицирующий программу операционной системы АСВТ, выдавшей сообщение. Он состоит из трех буквенных символов, которые являются тремя первыми символами названия программы, и трех цифр, определяющих номер сообщения этой программы.

Индикатор сообщения - буква И или Л, определяющих соответственно информационное сообщение, по которому оператор не должен предпринимать никаких действий, и сообщение, требующее каких-либо действий от оператора. Вид действия обычно указывается в тексте сообщения.

4.4. Рекомендации по вводу с пульта оператора

При начальном запуске системы пульт оператора переводится в готовое состояние (включается двигатель печатающей машинки, тумблера "вкл" и "работа" на панели слева и тумблер "канал" на панели справа). После печати сообщения "введите дату и время" оператор может вводить команды в машину. Для этого нужно нажать кнопку "запрос" на панели управления справа и ждать, когда загорится лам-

почка "ввод" после чего набирается нужная команда.

Максимальная длина всей набираемой информации - 150 символов. По окончании набора на клавиатуре оператор должен нажать кнопку "конец ввода" на панели управления справа.

При вводе с клавиатуры может случиться переполнение (загорается лампочка на панели "потеря символа"), то есть предыдущий символ в канал не отправлен, а уже нажата следующая клавиша. Оператор обязан, получив этот сигнал, начать операцию снова с нажатия кнопки "запрос".

Оператор может, нажав кнопку "аннулирование", аннулировать неверно набранную информацию и продолжать ввод, учитывая, что общее количество набранных символов, включая аннулированные, а также коды аннулирования и замены, не должно превышать 150 байтов. В противном случае оператор должен начать операцию снова с нажатия кнопки "запрос". Служебные символы "перевод строки", "возврат каретки с переводом строки", "возврат каретки на 1 шаг" могут использоваться только перед операцией ввода. Исправление текущего символа производится нажатием кнопки "замена", после чего набирается правильный символ.

Кнопка "особый случай" может использоваться оператором, чтобы информировать систему о временном отключении устройства при смене бланка на печатающей машинке по окончании операции.

Первой командой, введенной оператором, должна быть команда УСТАНОВ, устанавливающая дату и время.

Рекомендуется дов ввода команды СТАРТ установить частные тома, используемые достаточно часто несколькими заданиями, на свободные устройства и сообщить об этом управляющей программе командой РЕЗЕРВ, и по команде ПЕРЕВОД перевести (если необходимо) устройства в состояние, недоступное для использования.

Затем вводится команда СТАРТ для запуска программы, имя кото-

рой указано в поле операнда этой команды. Командой СТАРТ с параметром ДЗДН в поле операнда запускается к выполнению диспетчер заданий.

Желательно использование оператором команды НАЧАТЬ с параметром ИМЯЗДН в поле операнда. Это позволяет ему определить по сообщению на консоль всю проделанную за день работу и какие задания выполнялись в определенное время. Кроме того, при этом оператор знает название выполняемого задания в каждый данный момент и может ввести команду СТИЕНА для известного задания.

Оператор должен следить за сообщениями из системы и согласно индикатору действия, указанному в коде сообщения, или выполнить немедленно нужное действие, или послать команду ОТВЕТ с текстом, который необходим для программы, пославшей данное сообщение.

Если оператору необходимо закрыть входной поток до его окончания, он может сделать это, выдав команду ПАУЗА.

Если же оператору необходимо изменить устройство для входного потока, то, выдав команду ПАУЗА, он должен ждать сообщения о закрытии входного потока и только тогда вводить новую команду СТАРТ для диспетчера задания.

Оператор не имеет права разгружать устройство, если он не получил соответствующее сообщение. Он может устанавливать тома только на устройства, разгруженные системой.

Для запуска задач управления необходимо, чтобы программа "диспетчер задач реального времени" и задачи управления находились в главной памяти (и оформлены как задачи) или в системной библиотеке. Помимо устройств связи с объектом (УСО) в задачах управления используются устройства - пульт оператора-технолога(ПТО) и устройство алфавитно-цифровой регистрации (УАЦР).

4.5. Постановка задач управления технологическими процессами

Чтобы поставить задачи управления технологическими процессами на выполнение, оператор должен произвести следующие действия:

а) все устройства, к которым будут обращаться задачи управления, должны находиться в исходном состоянии;

б) с пульта программиста запустить программу "диспетчер задач реального времени" с помощью команды СТАРТ, указав приоритет и количество килобайтов под рабочую область в расчете на все задачи управления.

Пример:

СТАРТ ⏎ ПТО, 4,3 ⏎ ,

где ПТО - условное название программы "диспетчер задач реального времени" (в машине).

4 - приоритет.

3 - количество килобайтов под рабочую область;

в) с пульта технолога-оператора включить задачи управления в порядке убывания приоритета, т.е. первой включается задача наивысшего приоритета.

Включение задач производится следующим образом:

- включить "контур регулирования -ВКЛ";
- на цифровой клавиатуре набрать номер задачи (ХХСС);
- нажать кнопку "работа".

При этом на блоке сигнализации БС-I должна загореться соответствующая лампочка.

Отключение задач производится аналогично включению, только включается "контур регулирования - СТКЛ".

Машинные команды
(основной набор)

5. МАШИННЫЕ КОМАНДЫ

5.1 КОМАНДЫ АРИФМЕТИКИ С ФИКСИРОВАННОЙ ЗАПЯТОЙ

5.1.1. НАЗНАЧЕНИЕ

С помощью набора команд с фиксированной запятой осуществляются операции двоичной арифметики над операндами, которые, с одной стороны, могут быть адресами, индексами и счетчиками, а с другой - представлять собой данные с фиксированной запятой. В общем случае каждый операнд имеет знак, и его длина составляет 32 разряда. Отрицательные числа представляются в дополнительном коде. Один из операндов всегда находится в одном из 16 общих регистров; другой может находиться или в общем регистре или в памяти.

Набор команд предусматривает засылку, сложение, вычитание, сравнение, умножение, деление и запись в память, а также операции над знаком, преобразование основания системы счисления и сдвиги операндов с фиксированной запятой.

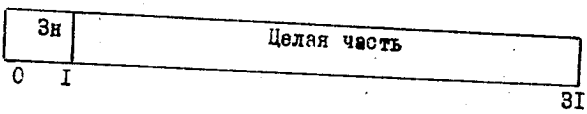
После выполнения операций над знаком, а также операций сложения, вычитания, сравнения и сдвига устанавливается код условия (признак результата).

5.1.2. ФОРМАТ ДАННЫХ

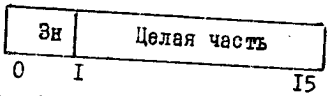
Числа с фиксированной запятой имеют формат фиксированной длины. В этом формате один разряд отводится под знак, а последующие разряды образуют поле целой части числа. Когда число находится в одном из общих регистров, его целая часть имеет 31 разряд, а само число занимает все 32 разряда регистра. Некоторые операции, такие, как умножение, деление, сдвиг, могут выполняться над 64-разрядными операндами: целая часть которых имеет 63 разряда. Такие операнды помещаются в два смежных общих регистра, а при адресации указывается общий регистр с четным

номером, который является левым регистром пары. В этом случае знаковый разряд регистра, содержащего первую часть числа, используется в целой части числа. Если выполняется операция типа регистр-регистр, в качестве обоих операндов может быть использовано содержимое одного и того же регистра.

Число с фиксированной запятой длиной в слово



Число с фиксированной запятой длиной в полуслово



Данные с фиксированной запятой, находящиеся в основной памяти, представляют собой 32-разрядные слова или I6-разрядные полуслова, поле целой части которых соответственно равно 3I или I5 разрядам. Команды преобразования основания системы счисления используют 64-разрядное десятичное поле. Данные различной длины должны помещаться в память в соответствии с границами, установленными для каждого случая, т.е. операнды, являющиеся словами двойной длины, обычными словами или полусловами, должны иметь адреса, в которых соответственно три, два или один младших разряда равны нулю.

Когда операнд длиной в полуслово выбирается из основной памяти, он превращается в операнд длиной в слово. Затем этот операнд участвует в операции как операнд длиной в слово.

5.1.3 ПРЕДСТАВЛЕНИЕ ЧИСЕЛ

Все операнды с фиксированной запятой рассматриваются как целые числа со знаком. Положительные числа представлены в прямом коде со знаковым разрядом, равным нулю. Отрицательные числа представлены в дополнительном коде со знаковым разрядом, равным единице. Дополнительный код числа получается инвертированием каждого разряда числа с последующим прибавлением единицы к младшему разряду.

Такой способ представления чисел обладает тем свойством, что любое число может рассматриваться как младшие разряды неопределенно длинного числа. Когда число положительно, все разряды левее старшего значащего разряда числа, в том числе и знаковый разряд, равны нулю. Когда число отрицательно, все эти разряды, включая знаковый разряд, равны единице. Поэтому, при необходимости увеличить длину операнда в сторону старших разрядов, увеличение достигается расширением поля целой части числа в сторону старших разрядов и присвоением каждому новому разряду значения, равного значению знакового разряда исходного числа.

При представлении числа в дополнительном коде отсутствует отрицательный нуль. При использовании дополнительного кода диапазон представляемых чисел таков, что общее количество отрицательных чисел на единицу больше, чем количество положительных. Наибольшее положительное число состоит из целой части, все разряды которой равны единице, и знакового разряда, равного нулю. Наибольшее отрицательное число состоит из целой части, все разряды которой равны нулю, и знакового разряда, равного единице.

В УП дополнение наибольшего отрицательного числа не может быть представлено. Когда в результате некоторой операции, напри-

мер, вычитания из нуля, образуется дополнение наибольшего отрицательного числа, число остается без изменения и формируется сигнал переполнения в операции с фиксированной запятой. Однако сигнал переполнения отсутствует, когда дополнение берется от наибольшего отрицательного числа в процессе выполнения операции, а окончательный результат находится внутри диапазона представляемых чисел. Примером такого случая является вычитание из -I. Произведением двух наибольших отрицательных чисел является положительное число двойной длины.

Знаковый разряд в числе является самым левым. При переполнении за счет переноса в знаковый разряд, значение последнего изменяется на противоположное. Однако при алгебраическом сдвиге значение знакового разряда сохраняется даже в том случае, когда старший значащий разряд выходит за разрядную сетку.

5.1.4 ЗАМЕЧАНИЯ ПО ПРОГРАММИРОВАНИЮ

Представление чисел в дополнительном коде особенно удобно при вычислении адресов и при операциях с повышенной точностью.

При использовании дополнительного кода можно считать, что отрицательное число представляет собой сумму целой части этого числа, рассматриваемой как положительное число, и наибольшего отрицательного числа. Поэтому в операциях с повышенной точностью поля младших разрядов должны обрабатываться как положительные числа. Аналогично, если при сдвиге вправо отрицательных чисел результат округляется, округление производится в сторону $-\infty$ а не в сторону нуля.

5.1.5 КОД УСЛОВИЯ (ПРИЗНАК РЕЗУЛЬТАТА)

В результате выполнения операций со знаком, а также операций сложения, вычитания, сравнения и сдвига над числами с фиксированной запятой, устанавливается код условия, который фиксируется в слове состояния программы (ССП). Остальные операции над числами с фиксированной запятой оставляют код условия без изменения. Значение кода условия может быть использовано для выбора пути при выполнении последующих команд условного перехода.

С помощью признака результата можно отразить три типа результатов операций над числами с фиксированной запятой. Для большинства операций значения признака результата 0,1 или 2 указывают на то, что содержимое регистра, хранящего результат операции, равно соответственно нулю, меньше нуля или больше нуля. Значение кода условия, равное 3, используется при переполнении. В операции сравнения значение 0,1 или 2 указывает на то, что первый операнд соответственно равен, меньше или больше второго.

При операциях СЛОЖЕНИЕ СПЕЦИАЛЬНОЕ и ВЫЧИТАНИЕ СПЕЦИАЛЬНОЕ значения 0 и1 характеризуют соответственно нулевое и ненулевое содержимое регистра результата при отсутствии логического переноса из знакового разряда, а значения 2 и 3 характеризуют нулевое и ненулевое содержимое регистра результата при наличии логического переноса из знакового разряда.

Значение кода условия для операций с фиксированной запятой

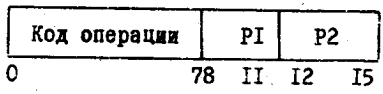
Сложение*	нуль	нуля	нуля	переполнение
сложение специальное	нуль, переноса нет	не нуль, переноса нет	нуль, есть перенос	не нуль, есть перенос
Сравнение*	равно	меньше	больше	–
Засылка с анализом	нуль	<нуля	>нуля	–
Засылка дополнения	нуль	<нуля	>нуля	переполнение
Засылка отрицательного	нуль	<нуля	–	–
Засылка модуля	нуль	–	>нуля	переполнение
Сдвиг влево двойной	нуль	<нуля	>нуля	переполнение
Сдвиг влево	нуль	<нуля	>нуля	переполнение
Сдвиг вправо двойной	нуль	<нуля	>нуля	переполнение
Сдвиг вправо	нуль	<нуля	>нуля	переполнение
Вычитание*	нуль	<нуля	>нуля	переполнение
Вычитание специальное	–	не нуль, переноса нет	нуль, есть перенос	не нуль, есть перенос

* Для слов и полуслов

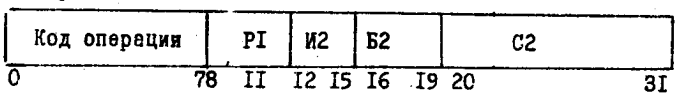
5.1.6 ФОРМАТ КОМАНД

В операциях с фиксированной запятой используются три формата команд:

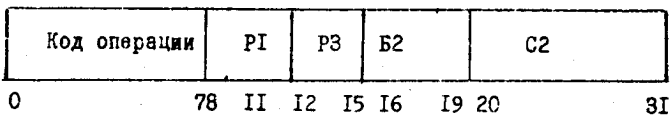
Формат РР



Формат РИ



Формат РП



Во всех форматах в поле РI, указывается адрес общего регистра, содержащего первый операнд. Адрес ячейки, хранящей второй операнд, если такой имеется, указывается для каждого формата по-разному.

В формате РР в поле Р2 указывается номер общего регистра, содержащего второй операнд. Допускается, чтобы и первый, и второй операнды находились в одном и том же общем регистре.

В формате РИ для получения адреса второго операнда содержимое общих регистров, заданных полями И2 и Б2, складывается с полем С2.

В формате РП для получения адреса второго операнда содержимое общего регистра, адрес которого указан в поле Б2, складывается

с полем С2. Этот адрес указывает место в памяти, где хранится второй операнд, при операциях ЗАСЫЛКА ГРУППЫ и ЗАПИСЬ ГРУППЫ. При операциях сдвига младшие 6 разрядов этого адреса указывают на сколько двоичных разрядов должен быть произведен сдвиг. В операциях ЗАСЫЛКА ГРУППЫ и ЗАПИСЬ ГРУППЫ поле РЗ указывает номер общего регистра. При операциях сдвига поле РЗ не используется.

Если в поле И2 или Б2 находятся нули, это говорит об отсутствии соответствующей компоненты адреса.

В команде может быть указан номер одного и того же общего регистра как для модификации адреса, так и для указания нахождения операнда. Модификация адреса всегда производится до выполнения операции.

Во всех операциях результат операции помещается на место первого операнда. Исключение составляют операции ЗАПИСЬ и ПЕРЕВОД в ДЕСЯТИЧНУЮ СИСТЕМУ, когда результат помещается на место второго операнда.

Содержимое всех общих регистров и ячеек памяти, принимающих участие в формировании адреса или в операции, остается неизменным, за исключением содержимого того регистра или ячеек, куда помещается результат операции.

5.1.7 КОМАНДЫ

В нижеследующей таблице приведены команды с фиксированной запятой, их мнемонические обозначения, форматы и коды операций. В таблице также указано, при выполнении каких команд устанавливается код условия и для каждой команды перечислены особые случаи, вызывающие прерывание программы.

Таблица 10

Название	Мнемоника	Тип	Особые случаи	Код
I	2	3	4	5
Засылка	ЗР З	РР РИ	А С	I8 58
Засылка полуслова	ЗК	РИ	А С	48
Засылка с анализом	ЗАР	РР У		I2
Засылка дополнения	ЗДР	РР У	ПФ	I3
Засылка модуля числа	ЗМР	РР У	ПФ	I0
Засылка отрицательного числа	ЗОР	РР У		II
Засылка группы	ЗГ	РП	А С	98
Сложение	СЛР СЛ	РР У РИ У	ПФ ПФ	IA 5A
Сложение с полусловом	СЛК	РИ У	А С ПФ	4A
Сложение специальное	СЛРС СЛС	РР У РИ У	А С	IE E
Вычитание	ВР В	РР У РИ У	ПФ А С ПФ	IB 5B
Вычитание полуслова	ВК	РИ У	А С ПФ	4B
Вычитание специальное	ВСР ВС	РР У РИ У	А С	IF 5F
Сравнение	СРР СР	РР У РИ У	А С	I9 59
Сравнение с полусловом	СРК	РИ У	А С	49
Умножение	УР- У	РР РИ	С А С	IC 5C
Умножение полуслова	УК	РИ	А С	4C
Деление	ДР Д	РР РИ	С ДФ А С ДФ	ID 5D
Перевод в двоичную систему	ПДВ	РИ	А СД ДФ	4F

I	2	3	4			5
Перевод в десятичную систему	ПДС	РИ	А	С	З	4
Запись	ЗП	РИ	А	С	З	5С
Запись полуслова	ЗПК	РИ	А	С	З	40
Запись группы	ЗПГ	РИ	А	С	З	90
Сдвиг влево арифметический	СЛА	РП У			ПФ	8В
Сдвиг вправо арифметический	СПА	РП У				8А
Сдвиг влево двойного слова арифметический	СЛДА	РП У		С	ПФ	8F
Сдвиг вправо двойного слова арифметический	СПДА	РП У		С	ПФ	8Е

Примечание.

А – неправильная адресация

С – нарушение спецификации

Д – неправильные данные

ПФ – переполнение с фиксированной запятой

ДФ – недопустимое деление при фиксированной запятой

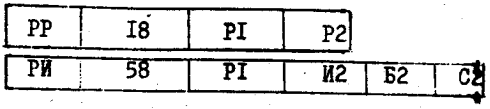
З – нарушение защиты памяти

У – устанавливается признак результата (код условия)

5.1.8 З А С Ы Л К А

По команде "Засылка" выполняется следующая операция: второй операнд длиной в слово без изменения помещается на место первого операнда. Результирующий код условия не вырабатывается (сохраняется старый).

Машинный формат:



Мнемокод-фо[illegible]:

ЗР P1, P2

З P1, C2 (И2, Б2)

Прерывания программы:

Адресация (только для З) .

Спецификация (только для З).

Если адрес какого-либо байта при выполнении команды "Засылка" формата РИ является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация". Если исполнительный адрес второго операнда не кратен четырем, возникает прерывание программы по причине "Нарушение спецификации".

Пример: I8 56 ЗР 5, 6

до выполнения команды:

регистр № 5	0I	23	45	67
регистр № 6	89	AB	CD	EF

После выполнения команды:

регистр № 5	89	AB	CD	EF
регистр № 6	89	AB	CD	EF

5.1.9 ЗАСЫЛКА ПОЛУСЛОВА.

По команде "Засылка полуслова" выполняется следующая операция: второй операнд длиной в полуслово расширяется до полного слова путем распространения влево кода знакового разряда и помещается на место первого операнда. Результирующий код условия не изменяется.

Машинный формат:

48	P1	И2	Б2	С2

Мнемокод-формат:

ЗК R1, C2 (И2, Б2)

Прерывание программы:

Адресация

Спецификация.

Прерывания программы по причине "Неправильная адресация" возникает аналогично соответствующему прерыванию в команде "Засылка". Если исполнительный адрес второго операнда не кратен двум, возникает прерывание программы по причине "Нарушение спецификации".

Пример: 48 51 23 02 ЗК 5, X'302' (1,2)

до выполнения команды:

регистр № 1		00	00	00	00
регистр № 2		00	00	00	00
регистр № 5		89	AB	CD	EF
Адреса	000300÷000303	01	23	45	67

после выполнения команды:

регистр № 1		00	00	00	00
регистр № 2		00	00	00	00
регистр № 5		00	00	45	67
Адреса	000300÷000303	01	23	45	67

5.1.10 ЗАСЫЛКА С АНАЛИЗОМ

По команде "Засылка с анализом" выполняется следующая операция: второй операнд без изменения помещается на место первого операнда. В зависимости от знака и величины второго операнда вырабатываются следующие результирующие коды условия:

0 – если результат равен нулю;

I – если результат меньше нуля;

2 – если результат больше нуля.

Машинный формат:

PP	I2	PI	P2

Мнемокод-формат:

ЗАР PI P2

Прерывания программы отсутствуют.

Примечание. В случае, когда в качестве и первого и второго операнда задано содержимое одного и того же регистра, выполнение операции заключается лишь в выработке кода условия.

Пример: I2 56 ЗАР 5,6

до выполнения команды:

регистр № 5	00	00	45	67
регистр № 6	89	AB	CD	EF

после выполнения команды:

регистр № 5	89	AB	CD	EF
регистр № 6	89	AB	CD	EF
Код условия		I		

5.1.11 ЗАСЫЛКА ДОПОЛНЕНИЯ.

По команде "Засылка дополнения" выполняется следующая операция: все разряды второго операнда инвертируются и к самому правому разряду (младшему) прибавляется единица: сформированный таким образом дополнительный код второго операнда помещается на место первого. При этом вырабатываются следующие коды условия:

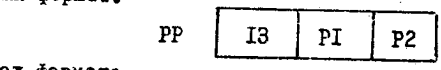
0 – если результат равен нулю;

1 – если результат меньше нуля;

2 – если результат больше нуля;

3 – если имело место переполнение (второй операнд до выполнения операции был равен 2^{31}).

Машинный формат:



Мнемокод-формат:

ЗДР P1 P2

Прерывание программы:

Переполнение с фиксированной запятой.

Прерывание возникает, когда берется дополнительный код от наибольшего отрицательного числа; в этом случае число остается без изменения. Переполнение вызывает прерывание программы, если разряд маски переполнения в операции с фиксированной запятой равен единице.

Примечание. Нуль обладает свойством инвариантности по отношению к операции получения дополнительного кода.

Пример: I3 56 ЗДР 5,6

до выполнения команды:

регистр № 5	89	AB	CD	EF
регистр № 6	89	AB	CD	EF

после выполнения команды:

регистр № 5	76	54	32	II
регистр № 6	89	AB	CD	EF

Код условия 2

5.1.12 ЗАСЫЛКА МОДУЛЯ ЧИСЛА

По команде "Засылка модуля числа" выполняется следующая операция: второй операнд выбирается из общего регистра и анализируется код в знаковом разряде; при коде "0" в знаковом разряде второй операнд без изменения помещается на место первого операнда; при коде "1" все разряды второго операнда инвертируются, к самому правому разряду прибавляется единица и сформированный таким образом код помещается на место первого операнда. В обоих случаях вырабатываются следующие результирующие коды условия:

0 – если результат равен нулю;

1 – если результате меньше нуля;

2 – если результат больше нуля;

3 – если имело место переполнение (второй операнд до выполнения операции был равен -2^{31}).

Машинный формат:

РР	I0	P1	P2

Мнемокод-формат:

ЗМР P1 P2

Прерывание программы:

Переполнение с фиксированной запятой:

Пример: I0 65 ЗМР 6,5

до выполнения команды:

регистр № 5	76	54	32	1I
регистр № 6	89	AB	CD	EF

после выполнения команды:

регистр № 5	76	54	32	11
регистр № 6	76	54	32	11

Код условия 2

5.1.13 ЗАСЫЛКА ОТРИЦАТЕЛЬНОГО ЧИСЛА.

По команде "Засылка отрицательного числа" выполняется следующая операция: второй операнд выбирается из общего регистра и анализируется код в знаковом разряде; при коде "I" в знаковом разряде второй операнд без изменения помещается на место первого операнда; при коде "0" - все разряды второго операнда инвертируются, к самому правому разряду прибавляется единица и сформированный, таким образом код помещается на место первого операнда. В обоих случаях вырабатываются следующие коды условия:

0 - если результат равен нулю;

I - если результат меньше нуля;

Машинный формат:

РР	II	РI	Р2

Мнемокод-формат:

ЗОР РI, Р2

Прерывания программы отсутствуют.

Пример: II 65 ЗОР 6,5

до выполнения команды:

регистр № 5	76	54	32	II
регистр № 6	76	54	32	II

после выполнения команды:

регистр № 5	76	54	32	II
регистр № 6	89	AB	CD	EF
Код условия		I		

5.1.14 ЗАСЫЛКА ГРУППЫ

По команде "Засылка группы" выполняется следующая операция: Общие регистры, начиная с регистра, заданного полем РI, и кончая

регистром, заданным полем Р3, заполняются байтами главной памяти, определяемых адресом второго операнда. Область памяти, содержимое которой пересылается в общие регистры, начинается с байта, определяемого адресом второго операнда, и содержит столько слов, сколько необходимо, чтобы загрузить указанные регистры. Общие регистры загружаются в порядке возрастания их адресов, начиная с регистра, заданного полем Р1, до тех пор, пока не будет загружен регистр, заданный полем Р3. При этом за регистром с номером I5 следует регистр с номером 0. Второй операнд остается без изменения. Код условия не вырабатывается.

Машинный формат:

РП	98	Р1	Р3	Б2	С2

Мнемокод-формат:

ЗГ Р1, Р3 С2(Б2)

Прерывания программы:

Адресация.

Спецификация.

Если адрес какого-либо байта при выполнении команды "Засылка группы" является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация". Если адрес байта, подлежащего загрузке первым, не кратен четырем, возникает прерывание программы по причине "Нарушение спецификации".

Примечание. Возможны любые комбинации адресов общих регистров, определяемых в команде полями Р1 и Р3. Когда адреса регистров совпадают, выполняется передача только одного слова. Если адрес, заданный в поле Р3 меньше адреса, заданного в поле Р1,

последовательность используемых адресов регистров проходит через I5 и 0.

Пример: 98 57 28 00 3Г 5,7,X'300'(2)

до выполнения команды:

регистр № 2		00	00	00	00
регистр № 5		76	54	32	II
регистр № 6		89	AB	CD	EF
регистр № 7		00	00	00	00
Адреса	000300÷000303	0I	23	45	67
	000304÷000307	89	AB	CD	EF
	000308÷00030B	I2	34	56	78

после выполнения команды:

регистр № 2		00	00	00	00
регистр № 5		0I	23	45	67
регистр № 6		89	AB	CD	EF
регистр № 7		I2	34	56	78
Адреса	000300÷000303	0I	23	45	67
	000304÷000307	89	AB	CD	EF
	000308÷00030B	I2	34	56	78

5.1.15 С Л О Ж Е Н И Е

По команде "Сложение" выполняется следующая операция: второй операнд длиной в слово складывается с первым операндом по правилам сложения целых двоичных чисел, и сумма помещается на место первого операнда. Несовпадение переносов из знакового разряда и из старшего разряда целой части свидетельствует о переполнении, при этом знаковый разряд суммы сохраняется таким, каким он получился (не корректируется).

В результате выполнения команды "Сложение" вырабатываются следующие коды условия:

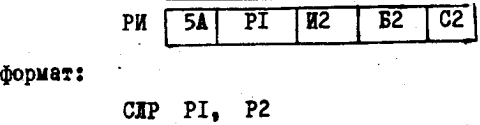
0 - если результат равен нулю;

1 - если результат меньше нуля;

2 - если результат больше нуля;

3 - если имело место переполнение.

Машинные форматы:



Мнемокод-формат:

СЛР R1, R2

СЛ R1, С2 (И2, Б2)

Прерывания программы:

Адресация (только для СЛ)

Спецификация (только для СЛ).

Переполнение с фиксированной запятой.

Если при выполнении команды "Сложение" (СЛ) формата РИ адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация", если исполнительный адрес второго операнда не кратен четырем, возникает прерывание программы по причине "Нарушение спецификации".

Если соответствующий разряд маски программы в ССП (переполнение в операции с фиксированной запятой) равен единице, возникает прерывание программы по причине "Переполнение с фиксированной запятой".

Пример: IA 56 СЛР 5,6

до выполнения команды:

регистр № 5	0I	23	45	67
регистр № 6	79	AB	CD	EF

после выполнения команды:

регистр № 5	7A	CF	I3	56
регистр № 6	79	AB	CD	EF
Код условия		2		

5.1.16 СЛОЖЕНИЕ С ПОЛУСЛОВОМ

По команде "Сложение с полусловом" выполняется следующая операция: второй операнд длиной в полуслово расширяется до полного слова путем распространения влево кода знакового разряда, складывается с первым операндом по правилам сложения целых двоичных чисел, и сумма помещается на место первого операнда. Несовпадение переносов из знакового разряда и из старшего разряда целой части свидетельствует о переполнении, при этом знаковый разряд суммы сохраняется таким, каким он получился.

В результате выполнения команды "Сложение с полусловом" вырабатываются следующие коды условия:

0 - если результат равен нулю;

I - если результат меньше нуля;

2 - если результат больше нуля;

3 - если имело место переполнение.

Машинный формат:

РИ

4A	RI	И2	Б2	C2

Мнемокод-формат:

СЛК RI, C2 (И2, Б2)

Прерывания программы:

Адресация

Спецификация.

Переполнение с фиксированной запятой.

Если при выполнении команды "Сложение с полусловом" было засвидетельствовано переполнение и соответствующий разряд маски программ равен "I", возникает прерывание программы по причине "Переполнение при фиксированной запятой".

Если адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация". Если исполнительный адрес второго операнда не кратен двум, возникает прерывание программы по причине "Нарушение спецификации".

Примечание. Поскольку используется дополнительный код, нулевой результат всегда положителен.

Пример: 4A 57 28 02 СЛК 5, 2(0, 302(0) (7,2)

до выполнения команды:

регистр № 2		00	00	I0	00
регистр № 5		8A	CF	I3	56
регистр № 7		00	00	00	00
Адреса	00I300-00I303	0I	23	45	67

после выполнения команды:

регистр № 2		00	00	I0	00
регистр № 5		8A	CF	58	BD
регистр № 7		00	00	00	00
Адреса	00I300-00I303	0I	23	45	67
Код условия			I		

5.1.17 СЛОЖЕНИЕ СПЕЦИАЛЬНОЕ

По команде "Сложение специальное" выполняется следующая операция: второй операнд складывается с первым по правилам сложения целых двоичных чисел, и сумма помещается на место первого операнда, причем несовпадение переносов из знакового разряда и из старшего разряда целой части не свидетельствует о переполнении.

Команда "Сложение специальное" предназначена для сложения младших частей чисел в подпрограммах сложения чисел двойной длины в двоичной системе счисления с фиксированной запятой. В результате выполнения данной команды вырабатываются следующие коды условия:

0 – если сумма равна нулю и нет переноса из знакового разряда;

1 – если сумма не равна нулю и нет переносов из знакового разряда;

2 – если сумма равна нулю и есть перенос из знакового разряда;

3 – если сумма не равна нулю и есть перенос из знакового разряда.

Машинный формат:

РР	1Е	Р1	Р2		
РИ	5Е	Р1	И2	Б2	С2

Мнемокод-формат:

СЛСР Р1 Р2

СЛС Р1, С2 (И2, Б2)

Прерывания программы:

Адресация (только для СЛС)

Спецификация (только для СЛС)

Если при выполнении команды "Сложение специальное" формата РИ (СЛС) адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание по причине "Неправильная адресация"; если исполнительный адрес второго операнда не кратен четырем, возникает прерывание программы по причине "Нарушение спецификации".

Пример:

IE 56 СЛСР 5,6

до выполнения команды:

регистр № 5	8A	CF	58	BD
регистр № 6	89	AB	CD	EF

после выполнения команды:

регистр № 5	I4	7B	26	AC
регистр № 6	89	AB	CD	EF

Код условия 3

5.1.18 В Ы Ч И Т А Н И Е

По команде "Вычитание" выполняется следующая операция: все разряды второго операнда инвертируются, к самому правому (младшему) разряду прибавляется единица и сформированный таким образом дополнительный код второго операнда складывается с первым операндом по правилам сложения целых двоичных чисел, а сумма помещается на место первого операнда. Несовпадение переносов из знакового и из старшего разрядов целой части свидетельствует о переполнении, при этом знаковый разряд суммы сохраняется (не корректируется) таким, каким он получился.

В результате выполнения команды "Вычитание" вырабатываются следующие коды условия:

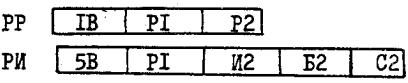
0 - если результат равен нулю;

1 - если результат меньше нуля;

2 - если результат больше нуля;

3 - если имело место переполнение.

Машинный формат:



Мнемокод-формат:

ВР	Р1,	Р2
В	Р1,	С2 (И2, Б2)

Прерывания программы:

Адресация (только В).

Спецификация (только В)

Переполнение с фиксированной запятой.

Если при выполнении команды "Вычитание" было засвидетельствовано переполнение и соответствующий разряд маски программы равен "1", возникает прерывание программы по причине "Переполнение при фиксированной запятой".

Если при выполнении команды "Вычитание" (В) формата РИ адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация". Если исполнительный адрес второго операнда не кратен четырем, возникает прерывание программы по причине "Нарушение спецификации".

Примечание. Когда в качестве первого и второго операндов задано содержимое одного и того же общего регистра, вычитание эквивалентно очистке этого регистра.

Вычитание наибольшего отрицательного числа из другого наибольшего отрицательного числа дает нулевой результат. Перепол-

нение при этом не возникает.

Пример: IB 56 BP 5,6

до выполнения команды:

регистр № 5	0A	CP	I3	56
регистр № 6	09	AB	CD	EF

после выполнения команды:

регистр № 5	0I	23	45	67
регистр № 6	09	AB	CD	EF
Код условия		2		

5.1.19 ВЫЧИТАНИЕ ПОЛУСЛОВА. По команде "Вычитание полуслова" выполняется следующая операция: второй операнд длиной в полуслово расширяется до полного слова путем распространения влево кода знакового разряда; все разряды расширенного второго операнда инвертируются, к самому правому (младшему) разряду прибавляется единица и сформированный таким образом код складывается с первым операндом по правилам сложения целых двоичных чисел, а сумма помещается на место первого операнда. Несовпадение переноса из знакового разряда и из старшего разряда целой части свидетельствует о переполнении, при этом код в знаковом разряде суммы не корректируется.

В результате выполнения команды "Вычитание полуслова" вырабатываются следующие коды условия:

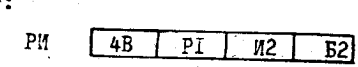
0 - если результат равен нулю;

I - если результат меньше нуля;

2 - если результат больше нуля;

3 - если имело место переполнение.

Машинный формат:



Мнемокод-формат:

ВК P1, C2 (И2, Б2)

Прерывания программы:

Адресация.

Спецификация.

Переполнение с фиксированной запятой.

Если при выполнении команды "Вычитание полуслова" было засвидетельствовано переполнение и соответствующий разряд маски программы равен "I", возникает прерывание программы по причине "Переполнение с фиксированной запятой".

Если адрес какого-либо байта соответствует несуществующим ячейкам главной памяти, возникает прерывание программы по причине "Неправильная адресация". Если исполнительный адрес второго операнда не кратен двум, возникает прерывание программы по причине "Нарушение спецификации".

Пример: 4B 57 23 02 ВК 5, X'302'(7,2)

до выполнения команды:

регистр № 2	00	00	I0	00
регистр № 5	8A	CP	58	BD
регистр № 7	00	00	00	00
Адреса 001300-001303	0I	23	45	67

после выполнения команды:

регистр № 2	00	00	I0	00
регистр № 5	8A	CP	I3	56
регистр № 7	00	00	00	00
Адреса 001300-001303	0I	23	45	67

Код условия I

5.1.20 ВЫЧИТАНИЕ СПЕЦИАЛЬНОЕ. По команде "Вычитание специальное" выполняется следующая операция: все разряды второго операнда (слова) инвертируются, к самому младшему (правому) прибавляется единица, и сформированный таким образом дополнительный код второго операнда складывается с первым операндом по правилам сложения целых двоичных чисел, и сумма помещается на место первого операнда, причем несовпадение переносов из знакового разряда и из старшего разряда целой части не свидетельствует о переполнении.

Команда "Вычитание специальное" предназначена для вычитания младших частей чисел в подпрограммах сложения чисел двойной длины в двоичной системе счисления с фиксированной запятой. В результате выполнения данной команды вырабатываются следующие коды условия:

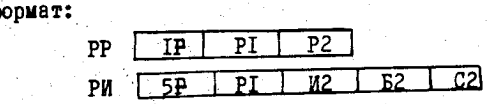
0 - если сумма равна нулю и нет переноса из знакового разряда;

I - если сумма не равна нулю и нет переноса из знакового разряда;

2 - если сумма равна нулю и есть перенос из знакового разряда;

3 - если сумма не равна нулю и есть перенос из знакового разряда;

Машинный формат:



Мнемокод-формат:

ВСР PI, P2

ВС PI, C2 (И2, Б2)

Прерывания программы:

Адресация (только для ВС)

Спецификация (только для ВС)

Если при выполнении команды "Вычитание специальное" формата РИ (ВС) адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация". Если исполнительный адрес второго операнда не кратен четырем, возникает прерывание программы по причине "Нарушение спецификации".

Примечание. При нулевой разности всегда имеет место перенос из знакового разряда.

Пример. IP 5,6 BCP 5,6

до выполнения команды:

регистр № 5	I4	7B	26	AC
регистр № 6	89	AB	CD	EF

после выполнения команды:

регистр № 5	8A	CF	58	BD
регистр № 6	89	AB	CD	EF
Код условия		I		

5.1.21 СРАВНЕНИЕ. По команде "Сравнение" выполняется следующая операция: первый операнд сравнивается со вторым операндом, и устанавливаются следующие результирующие коды условия:

0 - если операнды равны;

I - если первый операнд меньше второго;

2 - если первый операнд больше второго.

Сравнение выполняется алгебраически. Сравниваемые операнды рассматриваются как целые двоичные числа со знаком и сохраняются на своих местах неизменными.

Машинный формат:

РР	I9	PI	P2		
РИ	59	PI	И2	Б2	С2

Мнемокод-формат:

CPP	P1,	P2	
CP	P1,	C2 (И2, Б2)	

Прерывания программы:

Адресация (только для CP).

Спецификация (только для CP).

Если при выполнении команды "Сравнение" (CP) формата РИ адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация". Если исполнительный адрес второго операнда не кратен четырем, возникает прерывание программы по причине "Нарушение спецификации".

Пример: I9 56 CPP 5,6

до выполнения команды:

регистр № 5	8A	CF	58	BD
регистр № 6	89	AB	CD	EF

после выполнения команды:

регистр № 5	8A	CF	58	BD
регистр № 6	89	AB	CD	EF
Код условия		2		

5.1.22 СРАВНЕНИЕ С ПОЛУСЛОВОМ. По команде "Сравнение с полусловом выполняется следующая операция: второй операнд длиной в полуслово расширяется до полного слова путем распространения влево кода знакового разряда, и сформированный таким образом код сравнивается с первым операндом как целое двоичное число со знаком.

Сравнение выполняется алгебраически, в результате устанавливаются следующие коды условия:

0 - если операнды равны;

1 - если первый операнд меньше расширенного второго;

2 - если первый операнд больше расширенного второго операнда;

Операнды сохраняются на своих местах (в общем регистре и главной памяти) неизменными.

Машинный формат:

РИ	49	Р1	И2	Б2	С2

Мнемокод-формат:

СРК Р1, С2 (И2, Б2)

Прерывания программы:

Адресация.

Спецификация.

Если при выполнении команды "Сравнение с полусловом" адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация". Если исполнительный адрес второго операнда не кратен двум, возникает прерывание программы по причине "Нарушение спецификации".

Пример: 49 57 23 00 СРК 5,X'300' (7,2)

до выполнения команды:

регистр № 2	00	00	I0	00
регистр № 5	8A	CP	I3	56
регистр № 7	00	00	00	00
Адреса 001300-001303	01	23	45	67

после выполнения команды:

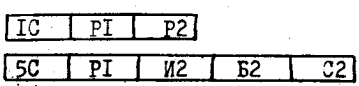
регистр № 2	00	00	I0	00
регистр № 5	8A	CP	I3	56
регистр № 7	00	00	00	00
Адреса 001300-001303	01	23	45	67

Код условия 1

5.1.23 У М Н О Ж Е Н И Е.

По команде "Умножение" выполняется следующая операция. Произведение множителя (второй операнд) и множимого (первый операнд) помещается на место множимого. Как множитель, так и множимое рассматриваются как целые 32-разрядные числа со знаком. Произведение всегда представляет собой целое 64-разрядное число со знаком и помещается в два смежных общих регистра с четным и нечетным номерами. Поскольку произведение замещает множимое, поле РI команды должно указывать на общий регистр с четным номером. Если в поле РI указан регистр с нечетным номером, это рассматривается как неправильная спецификация. Множимое берется из нечетного регистра пары. Содержимое нечетного регистра, замещаемое произведением, не влияет на операцию умножения, если только в этом регистре не находился множитель. При выполнении операции умножения переполнение произойти не может. Знак умножения определяется по алгебраическим правилам в зависимости от знаков сомножителей. В случае нулевого результата знак произведения всегда положителен. Результирующий код условия остается без изменения.

Машинный формат:



Мнемокод-формат:

УР РI, Р2

У РI, С2 (И2, Б2)

Прерывания программы:

Адресация (только для У).

Спецификация.

Если при выполнении команды "Умножение" формата РИ адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация".

Прерывание программы возникает по причине "Нарушение спецификации", если в команде "Умножение" указан нечетный номер общего регистра в поле РI и если исполнительный адрес первого байта множителя не кратен четырем.

Примечание. Значащая часть произведения обычно содержит 62 или меньшее количество разрядов. Только при перемножении двух наибольших отрицательных чисел получается произведение с 63 значащими разрядами. Так как используется представление чисел в дополнительном коде, значение знакового разряда произведения распространяется вправо на старшие разряды произведения вплоть до первой значащей цифры.

Пример. IC 6 5 УР 6,5

до выполнения команды:

регистр № 5 (множитель)	РР	РР	РР	РВ
регистр № 6	89	AB	CD	EF
регистр № 7	00	00	00	03

после выполнения команды:

регистр № 5	РР	РР	РР	РВ
регистр № 6	РР	РР	РР	РР
регистр № 7	РР	РР	РР	РI

5.1.24 **УМНОЖЕНИЕ НА ПОЛУСЛОВО.** По команде "Умножение на полуслово" выполняется следующая операция: множитель длиной в полуслово расширяется до полного слова путем распространения влево кода знакового разряда; затем расширенный множитель умножается на

множимое длиной в слово по правилам умножения целых двоичных чисел, выраженных в дополнительном коде, и младшие четыре байта произведения помещаются на место множимого. При этом код условия не изменяется, а остальные (старшие) байты произведения теряются. Результирующий код условия остается без изменения.

Машинный формат:

РИ	4С	Р1	И2	Б2	С2

Мнемокод-формат:

УК Р1, С2 (И2, Б2).

Прерывания программы:

Адресация.

Спецификация.

Если при выполнении команды "Умножение на полуслово" адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация" , если исполнительный адрес первого байта кода множителя не кратен двум, возникает прерывание программы по причине "Нарушение спецификации".

Примечание. Значащая часть произведения обычно содержит 46 или меньшее количество разрядов, за исключением случая, когда перемножается два наибольших числа, и произведение имеет 47 значащих разрядов.

Так как в результате выполнения операции запоминаются лишь 32 младших разряда произведения, а все более "левые" разряды произведения теряются, значение знакового разряда результата может отличаться от истинного при наличии переполнения.

Пример: 4С 57 23 СС УК 5,Х'300'(7,2)

до выполнения команды:

регистр № 2	00	00	I0	00
регистр № 5	РР	РР	РР	РВ
регистр № 7	00	00	00	00
Адреса 00I300÷00I30I	0I	23		

после выполнения команды:

регистр № 2	00	00	I0	00
регистр № 5	РР	РР	РА	5I
регистр № 7	00	00	00	00
Адреса 00I300÷ 00I30I	0I	23		

5.1.25 Д Е Л Е Н И Е

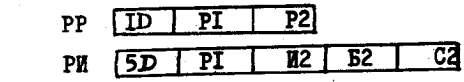
Делимое (первый операнд) делится на делитель (второй операнд), и частное с остатком помещается на место делимого. Делимое представляет собой 64-разрядное число со знаком и располагается в двух смежных регистрах с четным и нечетным номерами, определяемых полем РI команды. Если в поле указан регистр с нечетным номером, это рассматривается как неправильная спецификация. 32-разрядный остаток со знаком и 32-разрядное частное со знаком помещаются на место делимого в общие регистры с четным и нечетным номерами соответственно. Делитель представляет собой целое 32-разрядное число со знаком.

Знак частного определяется по алгебраическим правилам. Остаток имеет тот же знак что и делимое. Нулевое частное и нулевой остаток всегда положительны. Все операнды, частное и остаток рассматриваются, как целые числа со знаком.

Если соотношения между величинами делимого и делителя таковы, что частное нельзя представить в виде целого 32-разрядного числа со знаком, такой случай классифицируется как некорректность деления в операции с фиксированной запятой (в этом случае программа прерывается: операция деления не выполняется, и делимое в общих регистрах остается без изменения).

Код условия остается без изменения.

Машинный формат:



Мнемокод-формат:

ДР Р1, Р2

Д Р1, С2 (И2, Б2)

Прерывания программы:

Адресация (только для Д)

Спецификация.

Деление с фиксированной запятой.

Примечание: При делении всегда операнд выбирается из памяти как операнд длиной в обычное слово.

Пример. 5 6A 29 00 Д 6,X'900'(10,2)

до выполнения команды:

регистр № 2	00	00	00	00
регистр № 6	~~РР~~	~~РР~~	~~РР~~	~~РР~~
регистр № 7	~~РР~~	~~РР~~	РА	5I
регистр № A	00	00	00	00
Адреса 00I900÷00I903	00	00	0I	23

после выполнения команды:

регистр № 2	00	00	I0	00
регистр № 6 (остаток)	00	00	00	00

регистр № 7 (частное)	??	??	??	??
регистр № А	00	00	00	00
Адреса 001900÷001903	00	00	01	28

5.1.26 ПЕРЕВОД В ДВОИЧНУЮ СИСТЕМУ.

По команде "Перевод в двоичную систему" выполняется следующая операция: второй операнд (двойное слово) извлекается из главной памяти, переводится из десятичной системы счисления в двоичную и результат помещается в общий регистр с номером Р1. Для представления отрицательных чисел в двоичной системе счисления используется дополнительный код. Код условия не изменяется.

Машинный формат:

РИ	4Р	Р1	И2	Б2	С2

Мнемокод-формат

ПДВ Р1, С2(И2, Б2)

Прерывания программы:

Адресация.

Спецификация.

Данные.

Деление с фиксированной запятой.

Если при выполнении команды "Перевод в двоичную систему" адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация"; если исполнительный адрес первого байта исходного операнда не кратен восьми, возникает прерывание программы по причине "Нарушение спецификации".

Если при выполнении данной команды оказалось, что результат преобразования не помещается в отведенном общем регистре (имеет длину более четырех байтов), возникает прерывание программы по причине "Недопустимое деление с фиксированной запятой".

При выполнении команды "Перевод в двоичную систему" цифры и знак исходного операнда проверяются на допустимость кодов (недопустимыми считаются коды A, B, C, D , E, F во всех полубайтах, кроме самого правого (знакового), и коды 0-9 в самом правом полубайте). При обнаружении недопустимых кодов возникает прерывание программы по причине "Неправильные данные".

Пример: 4F 6A 23 0C ПДВ 6, X'300'(10,2)

до выполнения команды:

регистр № 2		00	00	40	00
регистр № 6		FF	FF	FF	FB
регистр № A		00	00	00	00
Адреса	004300÷004303	00	00	00	00
	004304÷004307	00	25	59	4A

после выполнения команды:

регистр № 2		00	00	40	00
регистр № 6		00	00	63	FA
регистр № A		00	00	00	00
Адреса	004300÷004303	00	00	00	00
	004304÷004307	00	25	59	4A

5.1.27 ПЕРЕВОД В ДЕСЯТИЧНУЮ СИСТЕМУ.

По команде "Перевод в десятичную систему" выполняется следующая операция: первый операнд преобразуется из двоичной системы счисления в десятичную и результат, имеющий фиксированную длину восемь байтов, помещается в главную память. Код знака помещается в самый правый полубайт, а неиспользованные старшие разряды поля результата заполняются нулями. Код условия не изменяется.

Машинный формат:

РИ	4E	Р1	И2	Б2	С2

Мнемокод-формат:

ПДС P1, C2 (И2, Б2)

Прерывание программы:

Защита памяти.

Адресация.

Спецификация.

Если при выполнении команды "Перевод в десятичную систему" адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация"; если адрес первого байта результата не кратен восьми, возникает прерывание программы по причине "Нарушение спецификации".

Если при выполнении данной команды ненулевой программный ключ не совпал с ненулевым ключом защиты области главной памяти, предназначенной для размещения кода результата, возникает прерывание программы по причине "Нарушение защиты памяти".

Пример: 4E 6A 23 00 ПДС 6,X'300'(I0,2)

до выполнения команды:

регистр № 2		00	00	40	00
регистр № 6		00	00	5B	4I
регистр № A		00	00	00	00
Адреса	004300÷004303	99	99	99	99
	004304÷004307	00	25	59	4A

после выполнения команды:

регистр № 2		00	00	40	00
регистр № 6		00	00	5B	4I
регистр № A		00	00	00	00
Адреса	004300÷004303	00	00	00	00
	004304÷004307	00	23	36	IA

5.1.28 З А П И С Ь

По команде "Запись" выполняется следующая операция: первый операнд без изменения записывается в главную память по адресу второго операнда, при этом сохраняется предыдущий код условия. Оба операнда имеют фиксированную длину слова.

Машинный формат:

РИ	50	Р1	И2	Б2	С2

Мнемокод-формат

ЗП Р1; С2 (И2, Б2)

Прерывания программы:

Защита памяти.

Адресация.

Спецификация.

Если при выполнении команды "Запись" адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация"; если исполнительный адрес второго операнда не кратен четырем, возникает прерывание программы по причине "Нарушение спецификации".

Если при выполнении данной команды ненулевой программный ключ не совпал с ненулевым ключом защиты области главной памяти, содержащей второй операнд, возникает прерывание программы по причине "Нарушение защиты памяти".

Пример: 50 5A 23 00 ЗП 5,Х'300'(10,2)

до выполнения команды:

регистр № 2	00	00	40	00
регистр № 5	??	??	?A	5I
регистр № A	00	00	00	00
Адреса 004300÷004303	00	00	00	00

после выполнения команды:

регистр № 2	00	00	40	00
регистр № 5	РР	РР	РА	5I
регистр № А	00	00	00	00
Адреса 004300÷004303	РР	РР	РА	5I

5.1.29 ЗАПИСЬ ПОЛУСЛОВА.

По команде "Запись полуслова" выполняется следующая операция: младшие I6 разрядов первого операнда без изменения помещаются в главную память на место второго операнда, при этом сохраняется предыдущий код условия.

Машинный формат:

РИ 40 РI И2 Б2 С2

Мнемокод-формат

ЗПК РI, С2 (И2, Б2)

Прерывания программы:

Защита памяти.

Адресация.

Спецификация.

Если при выполнении команды "Запись полуслова" адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация", если исполнительный адрес второго операнда не кратен четырем, возникает прерывание программы по причине "Нарушение спецификация".

Если при выполнении данной команды ненулевой программный ключ не совпал с ненулевым ключом защиты области главной памяти, содержащей второй операнд, возникает прерывание программы по причине "Нарушение защиты памяти".

Пример: 40 6А 23 С0 ЗПК 6,Х'300'(I0,2)

до выполнения команды:

регистр № 2	00	00	40	00
регистр № 6	00	00	5B	4I
регистр № A	00	00	00	00
Адреса 004300÷004303	FF	FF	FA	5I

после выполнения команды:

регистр № 2	00	00	40	00
регистр № 6	00	00	5B	4I
регистр № A	00	00	00	00
Адреса 004300÷004303	5B	4I	FA	5I

5.1.30 ЗАПИСЬ ГРУППЫ.

По команде "Запись группы" выполняется следующая операция: в главную память, начиная с адреса, равного сумме кодов базы и смещения, записывается содержимое группы общих регистров, начиная с регистра с номером R1 и кончая регистром с номером R3 включительно (за регистром с номером F следует регистр с номером 0); операция прекращается после записи в главную память содержимого общего регистра с номером R3. Код условия не вырабатывается.

Машинный формат:

RП	9C	R1	R3	Б2	C2

Мнемокод-формат

ЗПГ R1, R3, C2 (Б2)

Прерывания программы:

Защита памяти.

Адресация.

Спецификация.

Если при выполнении команды "Запись группы" адрес какого-либо байта является несуществующим адресом главной памяти, возникает прерывание программы по причине "Неправильная адресация"; если начальный адрес записи не кратен четырем, возникает преры-

вание программы по причине "Нарушение спецификации".

Если при выполнении данной команды ненулевой программный ключ не совпал с ненулевым ключом защиты области главной памяти, предназначенной для записи, возникает прерывание программы по причине "Нарушение защиты памяти".

Пример: 90 57 23 00 ЗПГ 5,7,X'300'(2)

до выполнения команды:

регистр № 2		00	00	00	00
регистр № 5		01	23	45	67
регистр № 6		89	AB	CD	EF
регистр № 7		12	34	56	78
Адреса	000300÷000303	00	00	00	00
	000304÷000307	00	00	00	00
	000308÷00030B	00	00	00	00

после выполнения команды:

регистр № 2		00	00	00	00
регистр № 5		01	23	45	67
регистр № 6		89	AB	CD	EF
регистр № 7		12	34	56	78
Адреса	000300÷000303	01	23	45	67
	000304÷000307	89	AB	CD	EF
	000308÷00030B	12	34	56	78

5.1.31 СДВИГ ВЛЕВО АРИФМЕТИЧЕСКИЙ.

По команде "Сдвиг влево арифметический" выполняется следующая операция: все разряды первого операнда, кроме знакового, сдвигаются влево на количество разрядов, определяемое исполнительным адресом второго операнда. Освободившиеся при сдвиге младшие разряды первого операнда заполняются нулями. Вытесненные влево разряды теряются. Знак первого операнда остается на месте.

В результате выполнения команды "Сдвиг влево арифметический" вырабатываются следующие коды условия:

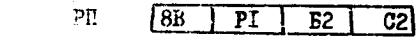
0 - если результат равен нулю;

1 - если результат меньше нуля;

2 - если результат больше нуля;

3 - если имело место переполнение (значение хотя бы одного вытесненного разряда не совпало со значением знакового).

Машинный формат:



Мнемокод-формат:

СЛА Р1, С2(Б2)

Прерывание программы:

Переполнение с фиксированной запятой.

Если при выполнении команды "Сдвиг влево арифметический", имело место переполнение и соответствующий разряд маски программы равен "1", возникает прерывание программы по причине "Переполнение с фиксированной запятой".

Примечание. Наличие базового регистра, принимающего участие в формировании адреса второго операнда, позволяет осуществить косвенное указание количества сдвигов. Нулевое поле Б2 команды указывает на то, что количество сдвигов задано непосредственно.

Пример: 8В 50 00 06 СЛА 5, 6(0)

до выполнения команды:

регистр № 5 01 28 45 67

после выполнения команды:

регистр № 5 48 D1 59 C0

Код условия 2

5.1.32 СДВИГ ВПРАВО АРИФМЕТИЧЕСКИЙ.

По команде "Сдвиг вправо арифметический" выполняется следующая операция: все разряды первого операнда, кроме старшего (знакового), сдвигаются вправо на количество разрядов, определяемое исполнительным адресом второго операнда. Исполнительный адрес второго операнда для обращения к главной памяти не используется, старшие восемнадцать разрядов исполнительного адреса игнорируются, а остальные (младшие) шесть разрядов указывают, на сколько двоичных разрядов должен быть произведен сдвиг. Нулевой общий регистр в формировании исполнительного адреса второго операнда не участвует. Освободившиеся при сдвиге старшие разряды первого операнда заполняются двоичными цифрами, равными значению знакового разряда. Вытесненные вправо разряды теряются. Знак первого операнда остается на месте.

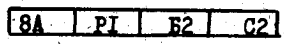
В результате выполнения команды "Сдвиг вправо арифметический" вырабатываются следующие коды условия:

0 - если результат равен нулю;

1 - если результат меньше нуля;

2 - если результат больше нуля.

Машинный формат:



Мнемокод-формат:

СПА РІ, С2(Б2)

Прерывания программы отсутствуют.

Примечание. Операция сдвига вправо аналогична делению на число 2 в некоторой степени с отбрасыванием младших разрядов. Так как отрицательные числа представлены в дополнительном коде, отбрасывание младших разрядов приводит к смещению отрицательных и положительных чисел на числовой оси в

отрицательном направлении, а не в сторону нуля, как это имеет место в операциях над десятичными числами. Если сдвиги выполняются на количество разрядов от 32 до 63, все значащие цифры выдвигаются за разрядную сетку. В этом случае результат равен нулю для положительных чисел и - I для отрицательных.

Пример: 8A 60 00 0A СПА 6,X'A'(0)

до выполнения команды:

регистр № 6	89	AB	CD	EF

после выполнения команды:

регистр № 6	FF	E2	6A	F8
Код условия		I		

5.1.33 СДВИГ ВЛЕВО ДВОЙНОГО СЛОВА АРИФМЕТИЧЕСКИЙ.

По этой команде выполняется следующая операция: все разряды первого операнда, представляющего собой слово двойной длины, расположенные в двух регистрах (четный и нечетный), кроме самого старшего (знакового), сдвигаются влево на количество разрядов, определяемое исполнительным адресом второго операнда. Исполнительный адрес второго операнда для обращения к главной памяти не используется: старшие восемнадцать разрядов исполнительного адреса игнорируются, а остальные (младшие) шесть разрядов указывают, на сколько двоичных разрядов должен быть произведен сдвиг. Нулевой общий регистр в формировании исполнительного адреса не участвует. Общий регистр РI+I содержит младшую часть операнда. Освободившиеся при сдвиге младшие разряды теряются. Знак первого операнда остается на месте.

В результате выполнения команды "Сдвиг влево двойного слова арифметический" вырабатываются следующие коды условия:

0 – если результат равен нулю;

1 – если результат меньше нуля;

2 – если результат больше нуля;

3 – если имело место переполнение (значение хотя бы одного вытесненного разряда не совпало со значением знакового).

Машинный формат:

РП	8Р	Р1	Б2	С2

Мнемокод-формат:

СЛДА Р1, С2(Б2)

Прерывания программы:

Спецификация.

Переполнение с фиксированной запятой.

Пример: 8Р 60 08 88 СЛДА 6,Х'888'(0)

до выполнения команды:

регистр № 6	РР	Р2	6А	Р8
регистр № 7	Р0	00	00	01

после выполнения команды:

регистр № 6	Р2	6А	Р8	Р0
регистр № 7	00	00	01	00

Код условия 1

5.1.34 СДВИГ ВПРАВО ДВОЙНОГО СЛОВА АРИФМЕТИЧЕСКИЙ.

По команде "Сдвиг вправо двойного слова арифметический" выполняется следующая операция: все разряды первого операнда представляют собой слово двойной длины из пары регистров Р1 и Р1+1, кроме самого старшего (знакового), сдвигаются вправо на количество разрядов, определяемое исполнительным адресом второго операнда. Исполнительный адрес второго операнда для обращения к главной памяти не используется: старшие восемнадцать разрядов кода исполнительного адреса игнорируются, а остальные (младшие)

шесть разрядов указывают, на сколько двоичных разрядов должен быть произведен сдвиг. Нулевой общий регистр в формировании исполнительного адреса не участвует. Номер общего регистра, хранящего младшую половину кода первого операнда, равен PI+I. Освободившиеся при сдвиге разряды первого операнда заполняются двоичными цифрами, равными значению знакового разряда. Вытесненные вправо разряды теряются.

В результате выполнения команды "Сдвиг вправо двойного слова арифметический" вырабатываются следующие коды условия:

0 - если результат равен нулю;

I - если результат меньше нуля;

2 - если результат больше нуля.

Машинный формат:

PI	8E	PI	B2	C2

Мнемокод-формат:

СПДА	PI	C2	/B2/

Прерывание программы:

Спецификация.

Если в команде "Сдвиг вправо двойного слова арифметический" указан нечетный номер общего регистра PI, возникает прерывание программы по причине "Нарушение спецификации".

Примечание. Если в операции сдвига слова двойной длины указано количество сдвигов нуль, операция заключается в проверке знака и величины исходного числа.

Пример: 8E 60 08 88 СПДА 6,X'888'/0/

до выполнения команды:

регистр № 6	F2	6A	F3	F0
регистр № 7	00	00	01	00

после выполнения команды:

регистр № 6	FF	F2	6A	F3
регистр № 7	F0	00	00	01

Код условия I

5.1.35 ОСОБЫЕ СЛУЧАИ ПРИ ВЫПОЛНЕНИИ ОПЕРАЦИЙ С ФИКСИРОВАННОЙ ЗАПЯТОЙ.

Неправильные команды, данные или результаты вызывают прерывание программы. При прерывании текущее ССП запоминается в качестве старого ССП и выбирается новое ССП. Код прерывания в старом ССП указывает на причину прерывания. Для операций с фиксированной запятой возможны следующие прерывания программы.

5.1.36 ЗАЩИТА ПАМЯТИ. Ключ памяти для ячейки результата не совпадает с ключом защиты в ССП. Операция не выполняется. Поэтому код условия, данные в регистрах и в памяти остаются без изменения. Исключение составляет команда "Запись группы", выполнение которой прекращается в тот момент, когда производится попытка записать содержимое очередного регистра в защищенную область памяти. Нельзя предсказать заранее, какое количество данных будет записано в память. Эти данные не должны использоваться в последующих вычислениях.

5.1.37 АДРЕСАЦИЯ. Адрес указывает на ячейку, которая отсутствует в памяти данной системы. Выполнение операции прекращается. Поэтому результат операции нельзя предсказать заранее, и он не должен использоваться в последующих вычислениях. Проверка адресов операндов выполняется только в том случае, когда они используются для адресации памяти. Адреса, используемые для указания количества сдвигов, не проверяются. Ограничения, налагаемые на адрес, не распространяются на компоненты, из которых формируется адрес , т.е. на содержимое поля С2 и содержимое регистров, заданных полями И2 и Б2.

5.1.38 СПЕЦИФИКАЦИЯ. Операнд двойной длины не попадает в границы 64-разрядных слов или операнд длиной в слово не попадает в

границы 32-разрядных слов, или операнд длиной в полуслово не попадает в границы 16-разрядных слов. Кроме того, неправильная спецификация имеет место, если в команде указан номер нечетного регистра для пары общих регистров, содержащих 64-разрядный операнд. Во всех случаях операция не выполняется. Поэтому результирующий код условия, данный в регистрах и в памяти остаётся без изменения.

5.1.39 ДАННЫЕ. Неправильный код знака или цифры десятичного операнда при выполнении команды "Перевод в двоичную систему". Операция не выполняется. Поэтому код условия, данные в регистрах и в памяти остаются без изменения.

5.1.40 ПЕРЕПОЛНЕНИЕ С ФИКСИРОВАННОЙ ЗАПЯТОЙ. Вышел за допустимый диапазон результат выполнения операции над знаком, операции сложения, вычитания или сдвига. Если разряд маски переполнения в операции с фиксированной запятой равен единице, происходит прерывание программы. В результате выполнения операции в регистр помещается усеченный результат, у которого сохранены только младшие разряды. Значение кода условия устанавливается равным 3. Разряды переполнения теряются. В операциях типа сложения значение знакового разряда оказывается противоположным истинному знаку суммы или разности. В операциях сдвига значение знакового разряда сдвинутого числа остается без изменения. Состояние разряда маски не влияет на результат операций.

5.1.41 НЕКОРРЕКТНОСТЬ ДЕЛЕНИЯ С ФИКСИРОВАННОЙ ЗАПЯТОЙ. Частное не помещается в регистр, включая деление на нуль, или результат операции перевод в двоичную систему превышает 31 разряд. Деление не выполняется. Поэтому данные в регистрах остаются без изменения. При выполнении операции преобразования в регистр помещается усеченный результат, у которого сохранены только младшие разряды.

5.2 ЛОГИЧЕСКИЕ ОПЕРАЦИИ

5.2.1 НАЗНАЧЕНИЕ.

Для логической обработки данных предусмотрен соответствующий набор команд. Операнды обычно рассматриваются как группы восьмиразрядных байтов. В отдельных случаях четыре левых или правых разряда байта рассматриваются отдельно, а иногда операнды сдвигаются на разряд за такт. Операнды находятся или в памяти, или в общих регистрах. Некоторые операнды берутся из команд программы.

Обработка данных в памяти идет слева направо по полю, которое может начинаться с любого байта. В общих регистрах работа, как правило, ведется над содержимым всего регистра.

За исключением команд редактирования, данные трактуются как нечисловая информация. При редактировании упакованные десятичные цифры преобразуются в восьмиразрядные символы.

Группа логических операций включает в себя пересылки, сравнения, поразрядные операции, проверки разрядов, перекодирование, редактирование и сдвиги. Все логические операции, за исключением операции редактирования, входят в стандартную систему команд. Команды редактирования относятся к средствам обработки десятичных данных.

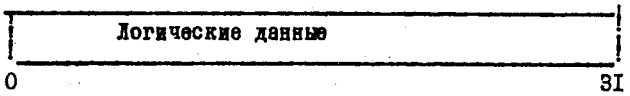
В результате выполнения всех операций логического сравнения, поразрядных операций, а также операций проверки и редактирования вырабатывается код условия.

5.2.2 ФОРМАТ ДАННЫХ

Данные находятся в общих регистрах, в памяти или берутся из команд. В качестве данных может быть использовано обычное слово, двойное слово, один символ или поле переменной длины.

Во всех командах, кроме команд редактирования, оба операнда, участвующие в операции, имеют одну и ту же длину.

Логическая информация фиксированной длины

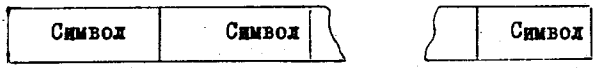


В общих регистрах данные обычно занимают все 32 разряда. Все разряды, в том числе знаковый, обрабатываются одинаково. В некоторых операциях участвуют только младшие восемь разрядов регистра, а остальные 24 разряда не меняются. В некоторых операциях сдвига участвуют 64 разряда, входящие в состав пары регистров, четного и нечетного.

Команда "Засылка адреса" помещает 24-разрядный адрес в общий регистр. Старшие восемь разрядов регистра устанавливаются в нуль.

В операциях формат РИ данные в памяти занимают или 32-разрядное слово или восьмиразрядный байт. Слово должно располагаться в памяти, начиная с границы слова, т.е. в двух младших разрядах адреса слова должны стоять нули.

Логическая информация переменной длины



В операциях "память-память" данные имеют переменную длину (не более 256 байтов) и могут начинаться с любого байта. Обработка идет слева направо. Операции, перемещающие данные непосредственно из команд в память, работают с одним восьмиразрядным байтом. Только один байт берется из команды, и только один байт в памяти участвует в операции.

В командах "Перекодирование с анализом" и "Редактирование с отметкой" подразумевается использование общего регистра I. Во время этих операций в данном регистре может быть замещен 24-разрядный адрес. В команде "Перекодирование с анализом" подразумевается такое участие общего регистра 2. Во время этой операции младшие восемь разрядов регистра 2 могут быть замещены байт-функцией.

Команда редактирования работает над полем упакованных десятичных цифр, результатом ее являются десятичные цифры в восьмиразрядном представлении. Цифры, знаки и зоны обрабатываются как в десятичной арифметике. В остальных командах считается, что у данных нет никакой внутренней структуры и допустимы все комбинации разрядов.

Операции перекодирования используют список произвольных величин. Список (словарь) задает связь между аргументом (величиной, используемой для обращения в словарь) и функцией (содержимым ячейки, которую ставят в соответствие аргументу). Целью перекодирования может быть преобразование данных из одного кода в другой или какая-то переработка текста.

Список определяется начальным адресом, который задает расположение самого левого байта списка. Байт из операнда, который надо перекодировать, является аргументом. Исполнительный адрес, используемый для адресации в списке, получается сложением аргумента с младшими разрядами начального адреса. Следовательно, список содержит 256 восьмиразрядных байт-функций. Список может быть уменьшен, если известно, что будут встречаться не все возможные значения восьмиразрядного аргумента.

В операции "память-память" поля операндов могут перекрываться. Влияние перекрытия зависит от типа операции. Перекрытие не влияет на выполнение операции, когда операнды остаются неизмен-

ными, как, например, в командах "Сравнение" или "Перекодирование с анализом". В командах же "Перемещение", "Редактирование и перекодирование" один операнд замещается новыми данными и выполнение операции может зависеть от степени перекрытия и от того, как данные выбираются из памяти и посылаются в нее. Для того, чтобы оценить влияние перекрытия операндов, можно считать, что данные перерабатываются последовательно байт за байтом. Любое перекрытие полей считается допустимым, но при редактировании в этом случае получаются заранее непредсказуемые результаты.

5.2.3 КОД УСЛОВИЯ (ПРИЗНАК РЕЗУЛЬТАТА)

Результаты большинства логических операций используются, чтобы установить в ССП код условия: ЗАСЫЛКА АДРЕСА, ЗАСЫЛКА СИМВОЛА, ЗАПИСЬ СИМВОЛА, ПЕРЕКОДИРОВАНИЕ, а также операции пересылки и сдвига оставляют этот код неизменным. Код условия может быть использован для выбора направления в последующих командах переходов по признаку.

С помощью кода условия при логических операциях можно отразить пять типов результатов. Для команды "Сравнение байтов" значения 0,1 или 2 соответствуют тем случаям, когда первый операнд соответственно равен, меньше или больше второго.

Для поразрядных операций значения 0 или 1 указывают соответственно на нулевое или ненулевое поле результата.

Для команды "Анализ под маской" значения 0,1 или 3 указывают на то, что выбранные разряды либо все равны нулю, либо среди них есть и нули и единицы, либо все они единицы.

Для команды "Перекодирование с анализом" значения 0,1 и 2 указывают соответственно на то, что все байт-функции оказались нулевыми, ненулевой байт-функция найден до окончания просмотра

первого операнда, последний байт-функция оказался ненулевым.

Для редактирования значения 0,1 или 2 указывают на то, что содержимое последнего поля результата равно нулю, меньше нуля или больше нуля.

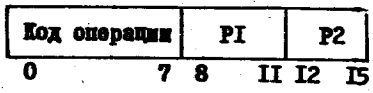
Значение кода условия для логических операций

Конъюнкция	нуль	не нуль	–	–
Сравнение байтов	равно	меньше	больше	–
Редактирование	нуль	>нуля	<нуля	–
Редактирование с отметкой	равно	>нуля	<нуля	–
Неэквивалентность	нуль	не нуль	–	–
Дизъюнкция	нуль	не нуль	–	–
Анализ под маской	нуль	нули и единицы	–	единицы
Перекодирование с анализом	нуль	не завершилась	завершилась	–

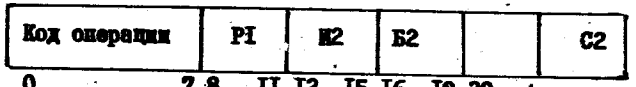
5.2.4 ФОРМАТ КОМАНД

Имеются следующие пять форматов для логических команд:

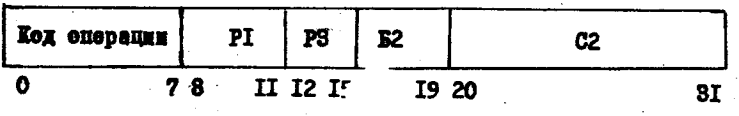
Формат РР



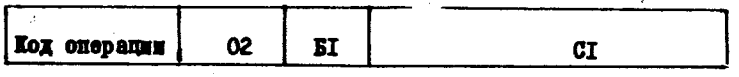
Формат РП



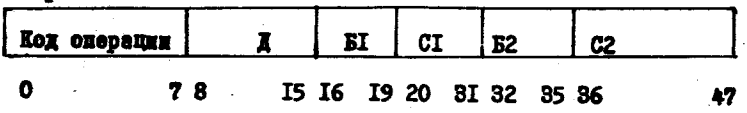
Формат РП



Формат ПО



Формат ПП



В форматах РР, РИ и РП содержимое регистра, определяемого полем Р1, берется в качестве первого операнда.

В форматах ПО, ПП адрес получается сложением содержимого общего регистра, определяемого Б1, с полем С1. Этот адрес указывает самый левый байт поля первого операнда. Число байтов справа от первого байта в формате ПП определяется полем Д. В формате ПО длина операнда равна одному байту.

В формате РР регистр, содержащий второй операнд, определяется полем Р2. Один и тот же регистр может быть указан как для первого, так и для второго операнда.

В формате РИ, чтобы получить адрес второго операнда, содержимое общих регистров, определяемых полями И2 и Б2 складывается с содержимым поля С2.

В формате РП, который используется для операций сдвига, содержимое общего регистра, определяемого полем Б2, складывается

с содержимым поля С2. Эта сумма определяется не адрес, а число разрядов, на которое необходимо произвести сдвиг. Поле Р8 в операциях сдвига не используется.

В формате ПО в качестве второго операнда непосредственно из команды берутся восемь разрядов поля О2.

В формате ПП, чтобы получить адрес второго операнда, содержимое общего регистра, определяемого полем Б2, складывается с содержимым поля С2. Поле второго операнда имеет ту же самую длину, что и поле первого операнда. Если в поле И2, Б2 или С2 находятся нули это указывает на то, что соответствующая компонента адреса или константы сдвига равна нулю. Один и тот же общий регистр может быть использован в команде и для формирования адреса, и как операнд.

Адрес формируется всегда до выполнения операции.

Результат помещается на место первого операнда. Исключение представляет команда "Запись символа", в которой результат засылается на место второго операнда. Результат переменной длины никогда не выходит в памяти за пределы поля, определяемого адресом и длиной.

Содержимое всех общих регистров и ячеек памяти, участвующих в адресации или в выполнении операции, остается, вообще говоря, неизменным. Исключение представляют ячейки результата, общий регистр I в команде "Редактирование с отметкой" и общие регистры I и 2 в команде "Перекодирование с анализом".

5.2.5 КОМАНДЫ

Ниже приведены все логические команды, их мнемонические обозначения, форматы и коды операций. В таблице также отмечены команды, относящиеся к средствам обработки десятичных данных,

указано, в каких командах устанавливается код условия и для каждой команды приведены особые случаи, которые вызывают прерывание программы.

Название	Мнемоника	Тип	Особые случаи				Код
Перемещение	ПП	ПП		А		З	D2
	ПО	ПО		А		З	92
Перемещение цифр	ПЦ	ПП		А		З	D1
Перемещение зон	ПЗ	ПП		А		З	D3
Сравнение логическое	СРЛП	ПП	У	А			5
Сравнение слов логическое	СРЛР	РР	У				I5
	СРЛ	РИ	У	А	С		55
Сравнение байтов	СРЛО	ПО	У	А			95
Конъюнкция	КП	ПП	У	А		З	D4
Конъюнкция слов	КР	РР	У				I4
	К	РИ	У	А	С		54
Конъюнкция байтов	КО	ПО	У	А		З	94
Дизъюнкция	ДЗП	ПП	У	А		З	D6
Дизъюнкция слов	ДЗР	РР	У				I6
	ДЗ	РИ	У	А	С		56
Дизъюнкция байтов	ДЗО	ПО	У	А		З	96
Неэквивалентность	НП	ПП	У	А		З	D7
Неэквивалентность слов	НР	РР	У				I7
	Н	РИ	У	А	С		57
Неэквивалентность байтов	НО	ПО	У	А		З	97
Анализ под маской	АМ	ПО	У	А			9I
Анализ с установкой	АУ	ПО		А		З	98
Засылка символа	ЗС	РИ		А			48
Запись символа	ЗПС	РИ		А		З	42
Засылка адреса	ЗА	РИ					4I
Перекодирование	ПК	ПП		А		З	DC
Перекодирование с анализом	ПКА	ПП	У	А			DD
Редактирование	РД	ПП	ТУ	А	Д	З	DE
Редактирование с отметкой	РДО	ПП	ТУ	А	Д	З	DF
Сдвиг влево логический	СЛЛ	РП					89

Название	Мнемоника	Тип	Особые случаи	Код
Сдвиг вправо логический	СПЛ	РП		88
Сдвиг влево двойного слова логический	СЛДЛ	РП	С	8D
Сдвиг вправо двойного слова логический	СПДЛ	РП	С	8C

Примечание: А – неправильная адресация
С – нарушение спецификации
Д – неправильные данные
З – нарушение защиты-памяти
Т – средства обработки десятичных данных
У – устанавливается код условия.

5.2.6 ЗАМЕЧАНИЯ ПО ПРОГРАММИРОВАНИЮ. В качестве логических операций могут быть использованы также команды записи в память и загрузки с фиксированной запятой.

5.2.7 ПЕРЕМЕЩЕНИЕ. Команда "Перемещение" имеет форматы ПП и ПО. По этой команде второй операнд замещает в главной памяти первый операнд. Формат ПП используется для перемещения "память-память". В формате ПО восьмиразрядный байт берется из команды. При перемещении "память-память" поля могут любым способом перекрываться. Перемещение выполняется слева направо байт за байтом. Перемещаемые байты не анализируются и не изменяются. В формате ПП операнды имеют одинаковую длину (Д+1) байтов.

Код условия остается без изменения.

Машинный формат:

ПО

92	02	БІ	СІ

ПП

D2	Д	БІ	СІ	Б2	С2

Мнемокод-формат:

ПО C1(Б1), 02

ПП C1(Д, Б1), C2 (Б2)

Прерывания программы:

защита памяти,

адресация.

Примечание. Один символ можно расписать по всему полю, размножить на все поле, задав начало поля первого операнда на символ правее начала поля второго операнда.

Пример:

D2 03 4 100 4102 ПП X'100'(4,4),X'102'(4)

До выполнения команды:

регистр № 4		00	00	01	00
адреса	000200÷000203	3A	00	00	FF
	000204÷000207	EE	00	00	00

После выполнения команды:

регистр № 4		00	00	01	00
адреса	000200÷000203	00	FF	EE	00
	000204÷000207	00	00	00	00

92 FF 02 00 ПО X'200'(0),X'FF'

До выполнения команды:

адреса	000200÷000203	10	EE	00	00

После выполнения команды:

адреса	000200÷000203	FF	EE	00	00

5.2.8 ПЕРЕМЕЩЕНИЕ ЦИФР. Команда "Перемещение цифр" имеет формат ПП. По этой команде четыре младших разряда (цифра) каждого байта второго операнда заменяют четыре младших разряда соответствующих байтов первого операнда. Старшие четыре разряда

(зона) каждого байта остаются неизменными в обоих операндах. Поля могут перекрываться любым способом. Перемещаемые разряды (цифры) не анализируются и не изменяются. Операнды имеют одинаковую длину (Д+I) байтов.

Код условия остается без изменения.

Машинный формат:

ПП	DI	Д	БI	CI	Б2	C2

Мнемокод-формат:

ПЦ CI(Д, БI), C2(Б2)

Прерывания программы:

защита памяти,

адресация.

Пример:

```
DI   03   0   200   0204    ПЦ  X'200'(4,0),X'204'(0)
```

До выполнения команды:

```
адреса   000200÷000203      I0   30   50   70
         000204÷000207      22   44   66   88
```

После выполнения команды:

```
адреса   000200÷000203      I2   34   56   78
         000204÷000207      22   44   66   88
```

5.2.9 ПЕРЕМЕЩЕНИЕ ЗОН. Команда "Перемещение зон" имеет формат ПП. По этой команде четыре старших разряда (зона) каждого байта второго операнда замещают четыре старших разряда соответствующих байтов первого операнда. Младшие четыре разряда (цифра) каждого байта остаются неизменными в обоих операндах. Перемещение идет слева направо байт за байтом. Поля могут перекрываться любым способом. Перемещаемые разряды (зоны) не анализируются

и не изменяются. Операнды имеют одинаковую длину (Д+I) байтов. Код условия остается без изменения.

Машинный формат:

ПП	D3	Д	БI	CI	Б2	C2

Мнемокод-формат:

ПП ПЗ CI (Д, БI), C2 (Б2)

Прерывания программы:

защита памяти,

адресация.

Пример.

```
D3      03   0   200   0204   ПЗ  X'200'(4,0),X'204'(0)
```

До выполнения команды:

```
адреса   000200-000203      00   РР   00   РР
         000204-000207      РР   00   РР   00
```

После выполнения команды:

```
адреса   000200-000203      РО   0Р   РО   0Р
         000204-000207      РР   00   РР   00
```

5.2.10 СРАВНЕНИЕ ЛОГИЧЕСКОЕ. Команда "Сравнение логическое" имеет форматы РР, РИ, ПО, ПП. По этой команде первый операнд сравнивается со вторым, и по результату сравнения устанавливается код условия. Сравнение производится поразрядно слева направо и любые коды являются допустимыми. Операция прекращается, как только встречаются несовпадающие разряды. Операнды равны, если они тождественны. Из двух не равных между собой операндов больший считается тот, у которого в самом старшем из несовпадающих разрядов стоит единица. Операнды сохраняются в памяти без изменения. Оба операнда в команде формата РР и РИ имеют фиксирован-

мую длину четыре байта, формата ПО – один байт, формата ПП – (Д+I) байтов.

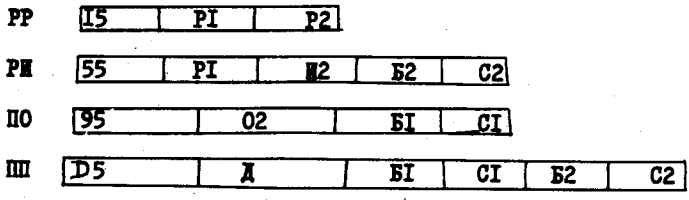
Код условия:

0 – операнды равны;

I – первый операнд меньше второго;

2 – первый операнд больше второго.

Машинный формат:



Мнемокод-формат:

СРЯР	РI, Р2
СРИ	РI, С2(И2, Б2)
СРЯО	СI (БI), 02
СРЯП	СI (Д, БI), С2 (Б2)

Прерывания программы:

адресация (только для РИ, ПО, ПП),

спецификация (только для РИ).

Примечание. Команда "Сравнение логическое" характерна тем, что все разряды обрабатываются как элементы двоичной величины без знака. Максимальная длина полей составляет 256 байтов. Команда может быть использована для сравнения буквенно-цифровых символов.

Пример:

```
D5    03  0  200  0204  СРЯП  X'200'(4,0),X'204'(0)
```

До выполнения команды:

адреса	000200 ÷ 000203	FF	00	FF	F0
	000204 ÷ 000207	FF	00	FF	FF

После выполнения команды:

адреса	000200 ÷ 000204	FF	00	FF	F0
	000204 ÷ 000207	FF	00	FF	FF

Код условия I

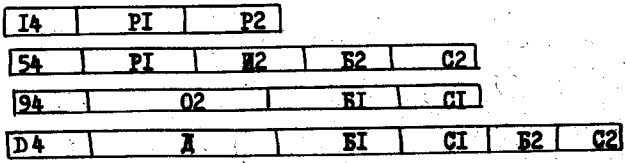
5.2.11 КОНЪЮНКЦИЯ. Команда "Конъюнкция) имеет форматы РР, РИ, ПО, ПП. По этой команде поразрядное логическое произведение (И) первого и второго операндов помещается на место первого операнда. Второй операнд сохраняется в памяти без изменения. Обработка производится слева направо. Все операнды и результаты являются допустимыми. Оба операнда в команде формата РР и РИ имеют длину четыре байта, формата ПО - один байт, формата ПП - (Д+I) байтов.

Код условия:

0 - результат равен нулю;

I - результат не равен нулю.

Машинный формат:



Мнемокод-формат:

КР	PI, P2
К	PI, C2(И2, Б2)
КО	CI (БI), 02
КП	CI (Д, БI), C2 (Б2)

Прерывания программы:

защита (только для ПО, ПП),

адресация (только для РИ, ПО, ПП),

спецификации (только для РИ)

Примечание. Команда "Конъюнкция" может быть использована для того, чтобы установить некоторый разряд в нуль.

Пример:

```
D4    08   0  200   0  204  ПП  X'200'(4,0), X'204'(0)
```

До выполнения команды:

адреса	000200 ↓ 000208	0F	FF	00	FF
	000204 ↓ 000207	F8	0F	FF	88

После выполнения команды:

адреса	000200 ↓ 000208	08	0F	00	88
	000204 ↓ 000207	F8	0F	FF	88

Код условия I

5.2.42 ДИЗЪЮНКЦИЯ. Команда "Дизъюнкция" имеет форматы РР, РИ, ПО, ПП. По этой команде поразрядная логическая сумма (ИЛИ) первого и второго операндов помещается на место первого операнда. Второй операнд сохраняется в памяти без изменения. Обработка производится слева направо. Все операции и результаты являются допустимыми. Оба операнда в команде формата РР и РИ имеют длину четыре байта, формата ПО - один байт, формата ПП - (Д+I) байтов.

Код условия:

0 - результат равен нулю;

I - результат не равен нулю.

Машинный формат:

РР	I6	R1	R2		
РИ	56	R1	И2	Б2	С2

ПО	96	02	БI	CI		
ПП	D 6	Д	БI	CI	Б2	С2

Мнемокод- формат:

ДЗР РI, Р2

ДЗ РI, С2(И2, Б2)

ДЗО CI (БI), 02

ДЗП CI (Д, БI), С2 (Б2)

Прерывания программы:

защита (только для ПО, ПП),

адресация (только для РИ, ПО, ПП),

спецификация (только для РИ),

Примечание. Команда "Дизъюнкция" может быть использована для установки некоторого разряда в единицу.

Пример:

D6 02 0 20I 0 204 ДЗП X'20I'(3,0),X'204'(0)

До выполнения команды:

адреса 000200 ÷ 000203 0I РО 0I 55

000204 ÷ 000207 РР 02 II ОР

После выполнения команды:

адреса 000200 ÷ 000203 0I РР 03 55

000204 ÷ 000207 РР 02 II ОР

Код условия I

5.2.13 НЕЭКВИВАЛЕНТНОСТЬ. Команда "Неэквивалентность" имеет форматы РР, РИ, ПО, ПП. По этой команде поразрядная сумма по модулю два (исключающее или) первого и второго операндов помещается на место первого операнда. Второй операнд сохраняется в памяти без изменения. Обработка производится слева направо. Все операн-

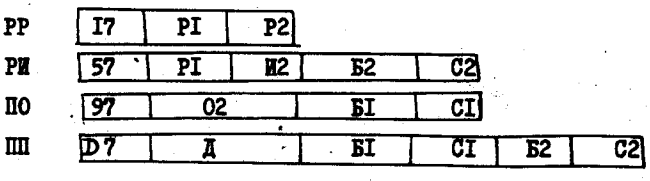
ды и результаты являются допустимыми. Оба операнда в команде формата РР и РИ имеют длину четыре байта, формата ПО - один байт, формата ПП - (Д+I) байтов.

Код условия:

0 - результат равен нулю;

I - результат не равен нулю.

Машинный формат



Мнемокод-формат:

НР	РI, Р2
И	РI, С2 (И2, Б2)
НО	СI (БI), 02
НП	СI(Д, БI), С2 (Б2)

Прерывания программы:

защита (только для ПО, ПП),

адресация (только для РИ, ПО, ПП),

спецификация (только для РИ).

Примечание. Команда "Неэквивалентность" может использоваться для инвертирования значения разряда.

Пример.

```
D7    02   0  20I  0  205  НП  X'20I'(3,0),X'205'(0)
```

До выполнения команды:

```
адреса      000200 ÷ 000203      ОР    0I   02   34
            000204 ÷ 000207      РО    0I   02   34
```

После выполнения команды:

адреса	000200 ÷ 000203	0F	00	00	00
	000204 ÷ 000207	F0	0I	02	34

Код условия 0

5.2.14 АНАЛИЗ ПО МАСКЕ. Команда "Анализ по маске" имеет формат ПО. По этой команде состояние выбираемых при помощи маски разрядов первого операнда используется для того, чтобы установить код условия.

Байт данных непосредственно из поля 02 используется как восьмиразрядная маска. Разряды маски соответствуют разрядам символа в памяти, определяемого адресом первого операнда.

Разряд маски, равный единице, указывает на то, что соответствующий разряд памяти выбирается для анализа. Если разряд маски равен нулю, соответствующий разряд памяти игнорируется. Если все отобранные таким образом разряды памяти равны нулю, код условия устанавливается равным нулю. Код условия также устанавливается в нуль, если все разряды маски равны нулю. Если все выбранные разряды равны единице, код условия устанавливается равным 3; в остальных случаях код устанавливается равным I. Операнды в главной памяти остаются без изменения.

Код условия:

0 - все выбранные разряды равны нулю, нулевая маска;

I - среди выбранных разрядов есть и нули и единицы;

2 -

3 - во всех выбранных разрядах стоят единицы.

Машинный формат:

ПО	9I	02	БI	CI

Мнемокод-формат:

АМ CI (БI), 02

Прерывание программы:

адресация.

Пример:

```
91        55    0  201        АМ   X'201'(0),X'55'
```

До выполнения команды:

```
адреса    000200 ÷ 000203        РР    55    00    00
```

После выполнения команды

```
адреса    000200 ÷ 000203        РР    55    00    00
код условия                                 3
```

9.2.15 АНАЛИЗ С УСТАНОВКОЙ. Команда "Анализ с установкой" имеет формат ПО. По этой команде крайний левый разряд (разряд 0), адресованного байта, используется для установки признака результата, при этом в сам байт заносятся единицы.

В байт памяти все единицы заносятся сразу после осуществления выборки, необходимой для анализа разряда. Между моментом выборки и моментом занесения всех единиц другие обращения к этому байту невозможны.

Содержимое второго байта команды игнорируется. Операнд имеет длину один байт.

Код условия:

0 - крайний левый разряд указанного байта равен 0;

1 - крайний левый разряд указанного байта равен 1.

Машинный формат:

ПО	93	00	Б1	С1

Мнемокод-формат:

АУ С1 (Б1)

Примечание. Эта команда может быть полезна при совместном пользовании несколькими программами одной общей области памяти. Для этого разряд 0 некоторого байта должен рассматриваться как управляющий разряд. Необходимая блокировка может быть получена соблюдением в программах соглашения, по которому нуль в управляющем разряде указывает на доступность общей области, а единица означает, что область в данный момент использовать нельзя. Если код условия устанавливается в нуль, то область доступна; если же признак результата устанавливается в единицу, то область недоступна. Когда программа использует команду "Анализ с установкой" для проверки управляющего разряда и последующей установки его в единицу, доступ любой другой программы к этому разряду между моментом проверки и моментом установки невозможен.

Пример:

93	00	0	201	АУ	Х'201'(0)		
До выполнения команды:							
адреса	000200 ÷ 000203			РО	90	00	00
После выполнения команды:							
адреса	000200 ÷ 000203			РО	РР	00	00
Код условия					I		

5.2.16 ЗАСЫЛКА СИМВОЛА. Команда "Засылка символа" имеет формат РИ. По этой команде байт, указанный адресом второго операнда, помещается в разряды 24–31 регистра, определяемого адресом первого операнда. Остальные разряды регистра не меняются. Байт переносится из памяти в регистр без изменения и проверки.

Код условия остается без изменения.

Машинный формат:

РИ	43	РI	И2	Б2	С2

Мнемокод-формат:

ЗС, РI, С2(И2, Б2)

Прерывание программы:

адресация.

Пример:

```
43    00    0  20I    ЗС   0,Х'20I' (, 0)
```

До выполнения команды:

регистр № 0	I2	34	56	78
адреса 000200 ÷ 000203	00	4Р	00	I2

После выполнения команды:

регистр № 0	I2	34	56	4Р
адреса 000200÷000203	00	4Р	00	I2

5.2.47 ЗАПИСЬ СИМВОЛА. Команда "Запись символа" имеет формат РИ. По этой команде разряды 24-3I регистра, указанного в качестве первого операнда, помещаются по адресу второго операнда. Байт переносится из регистра в память без проверки и изменения.

Код условия остается без изменения.

Машинный формат:

РИ	42	РI	И2	Б2	С2

Мнемокод-формат:

ЗПС РI, С2 (И2, Б2)

Прерывания программы:

защита памяти;

адресация.

Пример:

```
42    00    0  20I    ЗПС  0,Х'20I'( , 0)
```

До выполнения команды:

регистр № 0	01	02	03	04
адреса 000200 ÷ 000203	РР	РР	РР	РР

После выполнения команды:

регистр № 0	01	02	03	04
адреса 000200 ÷ 000203	РР	04	РР	РР

5.2.18 ЗАСЫЛКА АДРЕСА. Команда "Засылка адреса" имеет формат РИ. По этой команде адрес второго операнда помещается в разряды 8-31 общего регистра, указанного в качестве первого операнда. Разряды 0-7 общего регистра устанавливаются в нуль. Обращение к памяти за операндами отсутствует. Никаких проводок по адресации, защите памяти и спецификации не производится.

Вычисление адреса происходит по правилам адресной арифметики. Любой перенос за 24 разряд пропадает.

Код условия остается без изменения .

Машинный формат:

РИ	41	Р1	И2	Б2	С2

Мнемокод-формат:

ЗА Р1, С2 (И2, Б2)

Прерывания программы отсутствуют.

Примечание. В полях Р1, И2, Б2 можно задавать один и тот же общий регистр. Исключение представляет общий регистр.0, который, может быть указан только в поле Р1. Таким образом, к младшим 24 разрядам любого общего регистра, за исключением нулевого, можно прибавить содержимое поля С2. Адрес регистра, к которому нужно прибавить поле С2, следует поместить в Р1, а также либо в И2 (Б2 равно нулю), либо Б2 (И2 равно нулю).

Пример:

```
  4I      0I     0  357      3A  0,X'357'(I,0)
```

До выполнения команды:

регистр № 0	P7	I2	34	77
регистр № I	IE	PP	PP	A9

После выполнения команды:

регистр № 0	00	00	03	00
регистр № I	IE	PP	PP	A9

5.2.19 ПЕРЕКОДИРОВАНИЕ. Команда "Перекодирование" имеет формат ПП. Второй байт команды содержит код длины первого операнда. По этой команде байты замещаются соответствующими байтами из списка (словаря), начальный адрес которого равен адресу второго операнда. Список (словарь) содержит не более 256 байтов. Байты первого операнда выбираются для перекодирования один за другим слева направо. Команда выполняется в следующем порядке: выбирается очередной байт первого операнда и складывается с начальным адресом списка (словаря). Полученный адрес является адресом байта, который замещает очередной байт первого операнда, используемый в качестве слагаемого для формирования адреса.

Все данные являются допустимыми.

Операция продолжается до тех пор, пока не кончится поле первого операнда.

Список (словарь) не изменяется, если только он не перекрывается с полем первого операнда.

Первый операнд имеет длину (Д+I) байтов.

Код условия остается без изменения.

Машинный формат:

ПП	DC	Д	Б1	С1	Б2	С2

Мнемокод-формат:

ПК О1 (Д, Б1), С2 (Б2)

Прерывания программы:

защита памяти,

адресация.

Пример:

```
    DC     08   0  200    0 900  ПК  X'200'(4,0), X'900'(0)
```

До выполнения операции:

адреса	000200 + 000203	02	01	00	08
	000900 + 000903	5B	5	52	58

После выполнения операции:

адреса	000200 + 000203	52	5	5B	58
	000900 + 000903	5B	5	52	58

5.2.20 ПЕРЕКОДИРОВАНИЕ С АНАЛИЗОМ. Команда "Перекодирование с анализом" имеет формат ПП. Второй байт команды содержит код длины первого операнда. По этой команде с помощью байтов первого операнда выбираются соответствующие байты из списка (словаря), которые анализируются и определяют дальнейший ход операции и результирующий код условия. Начальный адрес списка (словаря) равен адресу второго операнда. Список (словарь) содержит не более 256 байтов. Байты первого операнда выбираются один за другим слева направо.

Команда выполняется в следующем порядке: выбирается очередной байт первого операнда и складывается с начальным адресом

списка (словаря). Полученный адрес является адресом байта, который анализируется; если он равен нулю, выбирается очередной байт первого операнда и т.д., если он не равен нулю, операция заканчивается занесением ненулевого байта списка (словаря) в восемь младших разрядов общего регистра № 2, а текущего адреса байта первого операнда в 8-31 разряды общего регистра № 1. Разряды 0-28 общего регистра № 1 остаются без изменения. Первый операнд в памяти не меняется.

Если один или более байтов первого операнда не были обработаны, код условия устанавливается равным 1.

Если же ненулевой байт из списка (словаря) оказался соответствующим последнему байту первого операнда, то код условия - 2.

Если анализируемый байт из списка (словаря) равен нулю, операция продолжается со следующим байтом первого операнда.

Если поле первого операнда исчерпывается до того, как встретится ненулевой байт из списка (словаря), операция заканчивается, при этом код условия устанавливается равным 0. Содержимое общих регистров 1 и 2 не меняется.

Первый операнд имеет длину (Д+1) байтов.

Список (словарь) не изменяется.

Код условия:

0 - все байты, выбранные из списка (словаря), равны нулю;

1 - из списка (словаря) выбран ненулевой байт до окончания поля первого операнда;

2 - последний, выбранный из списка (словаря) байт, не равен нулю.

Машинный формат:

ПП

DD	Д	Б1	С1	Б2	С2

Мнемокод-формат:

ПКА C1 (Д, Б1), C2(Б2)

Прерывание программы:

адресация.

Примечание. Команда "Перекодирование с анализом" полезна для просмотра входного потока данных и для локализации ограничителей. С ее помощью поток можно легко разбить на предложения или поля данных для дальнейшей обработки.

Пример:

```
DD  03  0  200  0  202  ПКА  X'200'(4,0),X'202'
```

До выполнения команды:

адреса	000200 ÷ 000203	05	03	00	02
	000204 ÷ 000207	0Р	00	РР	00
регистр 1		12	34	77	55
регистр 2		34	56	88	99

После выполнения команды:

адреса	000200 ÷ 000203	05	03	00	02
	000204 ÷ 000207	0Р	00	РР	00
регистр 1		12	00	02	04
регистр 2		34	56	88	0Р

5.2.24 РЕДАКТИРОВАНИЕ. По команде "Редактирование" второй операнд (источник) преобразуется из уплотненного формата в зонированный и редактируется под управлением первого операнда (шаблона).

Результат замещает первый операнд.

Одной операцией можно отредактировать несколько полей, причем числовая информация может сочетаться с текстовой.

Указание длины Д относится только к первому операнду (шаблону). Шаблон содержит (Д+I) любых 8-разрядных символов.

Второй операнд (источник) представляет собой десятичное число со знаком или без него, записанное в уплотненном формате или же группу таких чисел. Этот операнд проверяется на допустимость кодов. Левый полубайт каждого байта может содержать только десятичную цифру, в то же время как правый полубайт любого байта может содержать, либо десятичную цифру, либо знак. Операнды обрабатываются слева направо по одному символу.

Перекрытие полей источника и шаблона дает непредсказуемый результат, поэтому перекрытие полей недопустимо.

Символ, который должен запоминаться в поле первого операнда, определяется 3-мя факторами: байтом шаблона, полубайтом источника и значением двоичной переменной S.

Может быть осуществлено одно из трех действий:

I. К цифре источника добавляется зона и полученный символ замещает соответствующий байт шаблона.

2. Символ шаблона не изменяется.

3. В соответствующий байт шаблона заносится заполняющий символ.

Двоичная переменная S используется для того, чтобы управлять занесением или замещением цифр источника и шаблона, а также для того, чтобы записать знак исходного числа и соответствующим образом установить код условия.

В начале операции двоичная переменная S устанавливается в нуль, а затем ее состояние меняется в зависимости от чисел источника и символов шаблона.

Когда цифра источника помещается в поле результата (шаблон) к ней добавляется зона (код зоны CIOI).

Цифры источника при операции редактирования анализируются только один раз. Они выбираются из поля второго по восемь разрядов сразу. Левые четыре разряда анализируются первыми, а правые четыре разряда хранятся до следующего символа шаблона, который вызывает анализ цифры. Однако проверка, не находится ли в правых четырех разрядах код знака, производится сразу же после анализа левых четырех разрядов.

Знак "плюс" /1010/ устанавливает двоичную переменную S в нуль, а знак "минус" /1011/ оставляет двоичную переменную S неизменной.

Заполняющий символ в ходе операции редактирования берется из шаблона. В качестве заполняющего символа используется первый символ шаблона, причем этот символ оставляется без изменения в поле результата, за исключением того случая, когда первым символом будет символ выбора цифры или символ начала значащей части. В этих случаях происходит анализ цифры, и, если цифра не нуль, она вставляется в поле результата.

Три символа шаблона имеют особое значение для редактирования.

1. Символ выбора цифры /0010, 0000/ заменяется цифрой источника с приформированной зоной, если эта цифра не равна нулю, или, если $S = 1$.

После этого S всегда устанавливается в единицу.

Если же цифра источника / d / равна нулю и $S = 0$, символ выбора цифры заменяется заполняющим символом, а S не изменяет своего значения.

2. Символ начала значащей части / 0020, 0002 / выполняет те же функции, что и символ выбора цифры, но после обработки этого символа S всегда устанавливается в единицу, указывая, что следующая цифра - значащая.

3. Символ разделения полей (0010, 0010) ограничивает отдельные поля при редактировании нескольких полей с помощью одной команды. Этот символ замещается символом заполнителем. Значение S устанавливается в нуль.

Код условия вырабатывается в конце операции и характеризует только последнее поле результата.

Код условия:

0 - нулевое поле результата;

1 - поля результата меньше нуля;

2 - содержимое поля результата больше нуля.

Ниже в таблице приведены некоторые сведения об операции редактирования. В двух левых столбцах даются символы шаблона и их коды. В следующих столбцах приведено значение цифр и двоичной переменной S, которые используются для выбора одного из возможных действий. В самом правом столбце указано новое состояние двоичной переменной S.

Код символа	Название и назначение символа	Анализируется ли цифра	Состояние двоичной переменной	Значение цифры	Символ, помещаемый в результат	Новое состояние двоичной переменной
0010.0000	Выбор цифры	да	$S=1$		цифра	
			$S=0$	$d\neq 0$	цифра	$S=1$
			$S=0$	$d=0$	заполнитель	
0010.0001	Начало значащей части	да	$S=1$		цифра	
			$S=0$	$d\neq 0$	цифра	$S=1$
			$S=0$	$d=0$	заполнитель	$S=1$
0010.0010	Разделение полей	нет			заполнитель	$S=0$
другие	включение текста	нет	$S=1$		оставить заполнитель	
			$S=0$			

Обозначения:

d - цифра источника,

s - двоичная переменная.

Оставить - символ шаблона не изменяется;

цифра - цифра источника замещает цифру шаблона;

заполнитель - заполняющий символ замещает символ шаблона.

Машинный формат:

ШП	DE	Д	Б1	С1	Б2	С2

Мнемокод-формат:

РД С1 (Д, Б1), С2 (Б2)

Прерывания программы:

операция (если отсутствуют средства обработки данных),

защита памяти,

адресация,

неверные данные.

Примечание. Как правило, второй операнд короче, чем шаблон, так как в каждом символе исходных данных содержится две цифры или цифра и знак. Если одной командой редактируется несколько чисел, указание о нулевом поле дается только для последнего поля.

Пример:

DE I3 0 200 0 308, на мнемокоде: РД X'200'(X'I4';0), X'308'(0)

До выполнения команды:

адреса	000200 ÷ 000203	40	20	47	20
	000204 ÷ 000207	20	20	47	20
	000208 ÷ 00020B	20	20	22	2I
	00020C ÷ 00020F	20	4E	30	20
	0002I0 ÷ 0002I3	20	00	22	40

адреса	000308 ÷ 00030B	I2	34	56	7B
	00030C ÷ 00030F	I2	34	5A	I2

После выполнения команды:

адреса	000200 ÷ 000203	40	5I	47	52
	000204 ÷ 000207	53	54	47	55
	000208 ÷ 00020B	56	57	40	5I
	00020C ÷ 00020F	52	4E	30	53
	0002I0 ÷ 0002I3	54	00	40	40
адреса	000308 ÷ 00030B	I2	34	56	7B
	00030C ÷ 00030F	I2	34	5A	I2

код условия 2

5.2.22 РЕДАКТИРОВАНИЕ С ОТМЕТКОЙ. По команде "Редактирование с отметкой" второй операнд (источник) преобразуется из уплотненного формата в зонированный под управлением первого операнда (шаблона). Результат замещает первый операнд. Адрес каждой первой значащей цифры результата записывается в общий регистр I.

За исключением дополнительного занесения адреса байта в общий регистр I, операция совпадает с операцией "Редактирование".

Адрес регистра I явно не задается. Адрес байта заносится в разряды 8-3I этого регистра. Адрес байта заносится каждый раз, когда двоичная переменная $S = 0$, а в поле результата помещается ненулевая цифра. Если же значимость вводится символом начала значащей части, стоящем в шаблоне, и при этом в результат помещается заполняющий символ, то адрес байта не заносится.

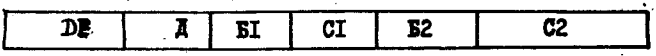
Код условия:

0 - нулевое поле результата;

I - содержимое поля результата меньше нуля;

2 - содержимое поля результата больше нуля;

Машинный формат:



Мнемокод-формат:

РДО CI (Д, БI), C2 (Б2)

Прерывания программы:

операция (если отсутствуют средства обработки десятичных данных),

защита памяти,

адресация,

неверные данные.

Пример:

```
DP  08  0  200  0  300   РДО X'200'(8,0),X'300'(0)
```

До выполнения команды:

регистр № I	I2	34	56	78
адреса 000200 ÷ 000203	40	20	20	6B
адреса 000300 ÷ 000303	0I	20	40	6B

После выполнения команды:

адреса 000200 ÷ 000203	40	40	5I	6B
адреса 000300 ÷ 000303	0I	20	40	6B
регистр № I	I2	00	02	02
код условия		2		

5.2.23 СДВИГ ВЛЕВО ЛОГИЧЕСКИЙ. Команда "Сдвиг влево логический" имеет формат РП. По этой команде первый операнд сдвигается влево на число разрядов, определяемое адресом второго операнда.

Адрес второго операнда для обращения к памяти не используется, младшие шесть разрядов адреса указывают число позиций,

на которое нужно произвести сдвиг. Остальная часть адреса игнорируется.

В сдвиге участвуют все 32 разряда общего регистра, определяемого полем Р2. Старшие разряды без проверки выдвигаются за пределы регистра и теряются. Освобождаемые младшие разряды регистра заполняются нулями.

Код условия остается без изменения.

Машинный формат:

РП	89	Р1		Б2	С2

Мнемокод-формат:

СДЛ Р1, С2(Б2)

Прерывания программы отсутствуют.

Пример:

```
89      I0  0   0I8          СДЛ  I,X'I8'
```

До выполнения команды:

регистр I Р7 33 55 66

После выполнения команды:

регистр I 66 00 00 00

5.2.24 СДВИГ ВПРАВО ЛОГИЧЕСКИЙ. Команда "Сдвиг вправо логический" имеет формат РП. По этой команде первый операнд сдвигается вправо на число разрядов, определяемое адресом второго операнда.

Адрес второго операнда для обращения к памяти не используется, младшие шесть разрядов адреса указывают число позиций, на которое нужно произвести сдвиг. Остальная часть адреса игнорируется.

В сдвиге участвуют все 32 разряда общего регистра, определяемого полем Р1. Младшие разряды без проверки выдвигаются за пре-

делы регистра и теряются. Освобождаемые старшие разряды регистра заполняются нулями.

Код условия остается без изменения.

Машинный формат:

РП	88	РI	Б2	С2

Мнемокод-формат:

СПИ РI, С2 (Б2)

Прерывания программы отсутствуют.

Пример:

```
88      I0    0  0I8          СПИ I,24
```

До выполнения команды:

регистр I Р7 33 55 66

После выполнения команды:

регистр I 00 00 00 Р7

5.2.25 СДВИГ ВЛЕВО ДВОЙНОГО СЛОВА ЛОГИЧЕСКИЙ. Команда "Сдвиг влево двойного слова логический" имеет формат РП. По этой команде первый операнд двойной длины сдвигается влево на число разрядов, определяемое адресом второго операнда.

Поле РI в команде определяет пару регистров с четным и нечетным номерами и должно содержать адрес четного регистра.

Адрес второго операнда для адресации данных не используется, младшие шесть разрядов указывают число позиций, на которое нужно произвести сдвиг. Остальная часть адреса игнорируется.

В сдвиге участвуют все 64 разряда пары регистров с четным и нечетным номерами, определяемой полем РI. Старшие разряды без проверки выдвигаются за пределы четного регистра и теряются. Освобождаемые разряды заполняются нулями.

Код условия остается без изменения.

Машинный формат:

РП	8D	Р1	Б2	С2

Мнемокод-формат:

СЛДЛ		Р1, С2 (Б2)

Прерывание программы:

спецификация.

Пример:

8D	00	0	020		СЛДЛ	0,32

До выполнения команды:

регистр 0	I2	3Р	5Р	6Р
регистр I	66	00	55	66

После выполнения команды:

регистр 0	66	00	55	66
регистр I	00	00	00	00

5.2.26 СДВИГ ВПРАВО ДВОЙНОГО СЛОВА ЛОГИЧЕСКИЙ. Команда "Сдвиг вправо двойного слова логический" имеет формат РП. По этой команде первый операнд двойной длины сдвигается вправо на число разрядов, определяемое адресом второго операнда.

Поле Р1 определяет пару регистров с четным и нечетным номерами и должно содержать адрес четного регистра. Адрес второго операнда для адресации данных не используется, его младшие шесть разрядов указывают число позиций, на которое нужно произвести сдвиг,. Остальная часть адреса игнорируется.

В сдвиге участвуют все 64 разряда пары регистров с четным и нечетным номерами, определяемой полем Р1. Младшие разряды без проверки выдвигаются за пределы нечетного регистра и теряются.

Освобождаемые разряды выполняются нулями.

Код условия остается без изменения.

Машинный формат:

РП	8С	РI		Б2	С2

Мнемокод-формат:

СПДИ РI, С2 (Б2)

Прерывание программы:

спецификация.

Примечание. Логические сдвиги отличаются от арифметических тем, что старший разряд участвует в сдвиге, а не размножается, код условия не меняется и не бывает переполнения.

Пример:

8С 00 0 0IF СПДИ 0,3I(0)

До выполнения команды:

регистр 0 99 88 44 7Р

регистр I РР 77 I2 34

После выполнения команды:

регистр 0 00 00 00 0В

регистр I 33 I0 88 РР

5.2.27 ОСОБЫЕ СЛУЧАИ ПРИ ВЫПОЛНЕНИИ ЛОГИЧЕСКИХ ОПЕРАЦИЙ.

Неправильные команды, данные или результаты вызывают прерывание программы; когда происходит прерывание, текущее ССП записывается в память в качестве старого ССП и выбирается новое ССП. Код прерывания в старом ССП указывает на причину прерывания. При логических операциях прерывания программы вызываются следующими особыми случаями.

Операция. Указана команда "Редактирование " и "Редактирование с отметкой", а средства обработки десятичных данных отсутствуют. Операция не выполняется. Поэтому код условия и данные в регистрах и памяти не меняются.

Защита памяти. Ненулевой программный ключ защиты в ССП не совпал с ненулевым ключом данной страницы главной памяти. Операция не выполняется. Поэтому код условия и данные в регистрах и памяти не меняются.

Исключение составляют операции типа "память-память" над полями переменной длины, выполнение которых прекращается. В этом случае данные и код условия получают заранее непредсказуемые значения, которые нельзя использовать для дальнейших вычислений.

Адресация. Если при выполнении команды появляется адрес ячейки, отсутствующей в памяти данной модели, возникает прерывание программы по причине "Неправильная адресация". Выполнение операции прекращается. Для данных и кода условия получаются заранее непредсказуемые значения, которые нельзя использовать для дальнейших вычислений.

Спецификация. В операциях формата РИ 32-разрядный операнд не попадает в границы 32-разрядных слов или для пары общих регистров в командах формата РП, содержащих 64-разрядный операнд, указан адрес нечетного регистра. Операция не выполняется. Код условия и данные в регистрах и памяти не меняются.

Данные. В команде "Редактирование" и "Редактирование с отметкой" имеется неправильный код цифры во втором операнде. Операция прекращается. Для данных и кода условия получаются заранее непредсказуемые значения, которые нельзя использовать для дальнейших вычислений.

Адреса операндов проверяются лишь тогда, когда они используются для обращения к памяти. Адреса, используемые для определения величины сдвига, не проверяются. Точно также не проверяется адрес, получаемый при использовании команды "Засылка адреса". Ограничения, накладываемые на адрес, не распространяются на компоненты, из которых он получается, а именно, на содержимое полей С1 и С2 и содержимое регистров, определяемых И2, Б1 и Б2.

5.3 КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

5.3.1 НАЗНАЧЕНИЕ

Обычно при выполнении команд универсальный процессор берет их из последовательно расположенных ячеек основной памяти. Такая последовательность операций может нарушаться, если встречается передача управления или, другими словами, переход. Наличие команды перехода позволяет при решении задачи делать выбор одного из двух направлений, обращаться к подпрограммам и выполнять несколько раз отдельные части программы, т.е. осуществлять циклы.

Переход выполняется путем засылки адреса перехода в качестве нового адреса команды. Последний может быть выбран из общего регистра или может находиться в самой команде. Адрес перехода не зависит от продвинутого адреса команды.

Выполнение перехода определяется признаком результата, который входит в состав слова состояния программы (ССП), или информацией, оказавшейся в общих регистрах в результате выполнения связанных с циклом операций.

При выполнении перехода правая половина ССП, в которой находится продвинутый адрес команды, может быть сохранена до того, как этот адрес будет заменен адресом перехода. Эта информация может быть использована в дальнейшем для того, чтобы связать новую последовательность команд с предыдущей последовательностью.

В группу команд перехода входит команда "Подмена". Адрес перехода в команде "Подмена" указывает одну команду, которая должна быть вставлена в данную последовательность команд.

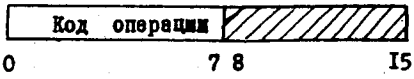
При выполнении этой команды продвинутый адрес команды обычно не изменяется, а только выполняется команда, находящаяся по адресу перехода.

5.3.2 ОБЫЧНАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ КОМАНД

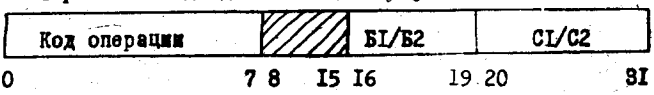
Обычной работой УП управляют команды, которые берутся из последовательно расположенных ячеек основной памяти. Команда выбирается из ячейки, определяемой полем адреса команды в ССП. Для адресации следующей команды из последовательности команд адрес выполненной команды увеличивается на число единиц, равное числу байтов в этой команде. Новое значение адреса команды, называемое продвинутым адресом команды, замещает прежнее содержимое поля адреса в ССП. Выполняется текущая команда, и все повторяется. Каждый раз для выбора следующей команды используется продвинутый адрес команды.

Команды могут быть длиной в полуслово или несколько полуслов. Максимальное число полуслов в команде равно трем. Число полуслов в команде указывается двумя первыми разрядами команды. Код 00 говорит о том, что команда имеет длину в полуслово, коды 01 и 10 используются для обозначения команд длиной в два полуслова, а код 11 служит для обозначения команд длиной в три полуслова:

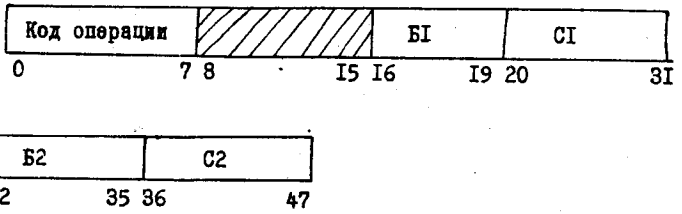
Формат команды длиной в полуслово



Формат команды длиной в два полуслова



Формат команды длиной в три полуслова



Адресация в памяти организована циклически. За байтом с максимальным адресом 16777215 следует байт с адресом 0. За командой, у которой последнее полуслово располагается в ячейке с максимальным адресом, следует команда с адресом 0. Кроме того, команда, состоящая из нескольких полуслов, может переходить через верхнюю границу памяти; при этом никакой специальной индикации для этого случая не предусмотрено.

Можно считать, что команда выбирается из памяти после завершения выполнения предыдущей операции и до выполнения текущей, даже если физический размер слова в памяти и совмещение выполнения команды с обращением к памяти могут повлиять на то, как будет происходить выборка команд в действительности.

Нарушение нормальной последовательности операций может происходить при переходе, переключении состояния, прерывании или при вмешательстве оператора. Последовательность операций начинается и прекращается с пульта управления системы.

Можно модифицировать команду в памяти с помощью предыдущей команды.

5.3.3 Особые случаи, вызывающие нарушение нормальной последовательности команд

Если адрес команды или код операции заданы неправильно, возникает прерывание программы. При прерывании текущее ССП запоминается в качестве старого ССП и выбирается новое ССП. Код прерывания в старом ССП указывает на причину прерывания. (В данном руководстве при описании каждого класса команд перечисляются те прерывания программы, которые могут возникнуть при выполнении команд из этого класса). Вне зависимости от выполняемой команды при нормальной последовательности команд могут иметь место следующие прерывания программы.

Операция. Заданный код операции не соответствует никакой операции.

Адресация. Хотя бы одно полуслово команды выходит за пределы памяти, имеющейся в данной установке.

Спецификация. Младший разряд адреса команды равен единице.

В любом случае операция не выполняется, поэтому код условия и данные в памяти и регистрах остаются без изменения. Адрес команды, который хранится как часть старого ССП, продвигается на число единиц, равное числу полуслов в команде. Число полуслов указывается кодом длины команды в старом ССП.

Замечание по программированию. Случай неправильного адреса команды может возникнуть, когда при выполнении обычной последовательности команд происходит выход за пределы имеющейся области памяти или когда встречается операция передачи управления или переключения состояния.

Нечетный адрес команды может появиться только тогда, когда встречаются операции передачи управления или переключения состояния.

Если последняя ячейка имеющейся в наличии памяти содержит команду, в которой, в свою очередь, указан правильный адрес команды, прерывания программы не происходит, даже если продвинутый адрес команды соответствует ячейке, которая отсутствует в данной установке.

Может случиться, что в команде с несуществующей операцией произойдет прерывание по адресации или спецификации, тогда как адресация или спецификация противоречат требованиям данного класса команд.

5.3.4 ВЫБОР НАПРАВЛЕНИЯ

Переходы бывают условные и безусловные. При выполнении безусловного перехода продвинутый адрес команды заменается адресом перехода. При выполнении команды условного перехода может быть использован или адрес перехода, или продвинутый адрес команды, который в последнем случае ничем не заменяется и не изменяется. Если используется адрес перехода, то говорят, что переход произошел; в противном случае говорят, что он не произошел.

Произойдет переход или не произойдет - зависит от результата операций, выполняемых одновременно с переходом или предшествующих ему. Первый случай представлен командой "Передача управления по счетчику" и командами перехода по индексу. Второй случай представлен командой "Условная передача управления" , которая анализирует признак результата, характеризующий результат ранее выполненной арифметической или логической операции, или операции ввода-вывода.

Признак результата позволяет выбирать направление в зависимости от полученных данных. Он анализируется в командах условного перехода. Признак результата устанавливается при выполнении

ряда операций, характеризуя результат операции, и не зависит от своего предыдущего значения. При выполнении других операций признак результата остается без изменения.

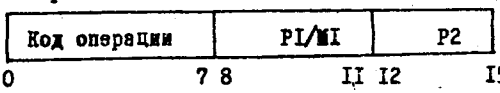
Признак результата хранится в 34-м и в 35-м разрядах ССП. Когда при переключении состояния происходит запоминание ССП, признак сохраняется как часть ССП. Он запоминается также, как часть правой половины ССП в операции "переход с возвратом". Новое значение признака получают по команде "Засылка ССП" или "Установка маски программы" или при загрузке нового ССП в результате прерывания.

Признак результата характеризует результат выполнения некоторых арифметических и логических операций, а также операций ввода-вывода. В случае команды "Подмена" признак изменяется или остается без изменения в зависимости от типа выполняемой команды, т.е. как если бы выполняемая команда встретилась в нормальной последовательности команд.

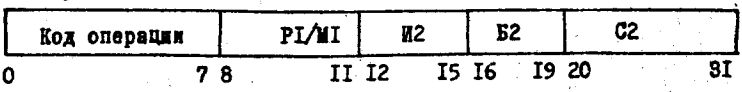
5.3.5 ФОРМАТЫ КОМАНД

Команды передачи управления бывают трех форматов:

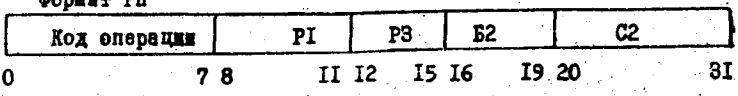
Формат РР



Формат РИ



Формат РП



Во всех форматах поле Р1 указывает адрес общего регистра. В команде "Условная передача управления" поле маски М1 указывает значение признака результата. Адрес перехода определяется в зависимости от типа формата.

В формате РР поле Р1 указывает адрес общего регистра, содержащего адрес перехода. Исключение составляет случай, когда в поле Р2 находятся нули, что говорит об отсутствии перехода. В полях Р1 и Р2 может быть указан адрес одного и того же общего регистра.

В формате РИ для получения адреса перехода содержимое общих регистров, заданных полями И2 и Б2, складывается с содержимым поля С2.

В формате РП для получения адреса перехода содержимое общего регистра, заданного полем Б2, складывается с содержимым поля С2.

Поле Р3 в формате РП одновременно указывает, где находится второй операнд и где находится третий операнд. Первый операнд задается полем Р1. Ячейка третьего операнда всегда имеет нечетный адрес. Если поле Р3 содержит четный адрес, третий операнд выбирается из общего регистра с адресом на единицу больше указанного. Если поле содержит нечетный адрес, третий операнд находится там же, где и второй.

Нулевое содержимое полей Б2 и И2 указывает на отсутствие соответствующих компонент адреса.

В команде может быть указан один и тот же общий регистр как для модификации адреса, так и для выбора операнда.

Содержимое общих регистров используется для различных частей операции в следующем порядке:

1. Формирование адреса.
2. Запоминание арифметических данных или информации о возврате.
3. Замещение адреса команды адресом перехода, полученным на первом шаге.

Результаты операции помещаются в общий регистр, заданный полем Р1. Общие регистры и ячейки памяти, принимающие участие в формировании адреса и выполнении операции, за исключением тех, которые использованы для записи конечных результатов, остаются без изменения.

5.3.6 ЗАМЕЧАНИЕ ПО ПРОГРАММИРОВАНИЮ. В командах передачи управления адрес перехода может быть задан двумя способами. В формате РИ адрес перехода - это адрес, заданный полями И2, Б2 и С2. В формате РР адрес перехода представляет собой содержимое общего регистра, заданного полем Р2. Следует отметить, что соотношение между обоими форматами при задании адреса перехода отличается от соотношения этих форматов при задании адреса операнда. В случае операндов адрес, заданный полями И2, Б2 и С2, является адресом операнда, тогда как общий регистр, заданный полем Р2, содержит сам операнд.

5.3.7 КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

В приведенной ниже таблице перечислены команды передачи управления и даны их мнемонические обозначения, форматы и коды операций. В таблице приведены также особые случаи, которые вызывают прерывание программы. Команда, на которую указывает команда "Подмена" , подчиняется своим собственным правилам прерывания. При выполнении команд перехода признак результата всегда остается без изменения.

Название	Мнемоника	Тип	Особые случаи	Код
Условная передача управления	ПУР	РР		07
	ПУ	РИ		47
Передача управления с возвратом	ПВР	РР		05
	ПВ	РИ		45
Передача управления по счетчику	ПСЧР	РР		06
	ПСЧ	РИ		46
Передача управления по превышению	ПБ	РП		86
Передача управления по непревышению	ПМ	РП		87
Подмена	ПОД	РИ	A C B	44

Примечание. A – неправильная адресация;

B – двойная подмена;

C – нарушение спецификации.

5.3.8 УСЛОВНАЯ ПЕРЕДАЧА УПРАВЛЕНИЯ. По команде "Условная передача управления" выполняется следующая операция: продвинутый адрес команды замещается адресом перехода, если значение кода условия соответствует коду, указанному в поле MI; в противном случае продолжается выполнение обычной последовательности команд с использованием продвинутого адреса. Поле MI используется в качестве четырехразрядной маски. Разряды маски соответствуют значениям кода условия следующим образом:

Код условия	Разряды маски	Значение маски
0	I	
I	2	4
2	3	2
3	4	I

Переход происходит, если значение соответствующего разряда маски равно единице. Коду маски может соответствовать несколько кодов условия.

Из определения следует, что при коде маски "F" (двоичный код IIII) осуществляется безусловная передача управления, а при коде маски "0" сохраняется нормальный порядок выполнения команд.

Код условия остается без изменения.

Машинный формат:

РР	07	MI	P2		
РИ	47	MI	И2	Б2	С2

Мнемокод-формат:

РР	ПУР	MI, P2
РИ	ПУ	MI, C2(И2, Б2)

Прерывания программы отсутствуют.

Примечание. Если поле P2 в формате РР содержит нуль, то передача управления отсутствует (т.е. происходит увеличение счетчика адреса на длину команды).

Пример:

47 I 0 I 230 ПУ I,X'230'(, I)

До выполнения команды:

регистр № I	00	00	I0	00
код условия	3			

После выполнения команды:

регистр № I	00	00	I0	00
адрес передачи управления		00	I2	30

5.5.9 ПЕРЕДАЧА УПРАВЛЕНИЯ С ВОЗВРАТОМ. По команде "Передача управления с возвратом" выполняется следующая операция: в общем регистре, заданном полем Р1, запоминается информация для возврата (правая половина ССП, содержащая продвинутый адрес команды); после этого происходит безусловная передача управления по адресу перехода.

Адрес перехода определяется перед запоминанием возврата.

Если в формате РР поле Р2 содержит нули, информация, необходимая для обеспечения возврата, запоминается, но перехода не происходит. Поля Р1 и Р2 могут определять один и тот же регистр.

Код условия остается без изменения.

Машинный формат:

РР	05	Р1	Р2		
РИ	45	Р1	И2	Б2	С2

Мнемокод-формат:

РР ПВР Р1, Р2

РИ ПВ Р1, С2 (И2, Б2).

Прерывания программы отсутствуют.

Примечание. Если команда "Передача управления с возвратом" выполняется в результате ссылки на нее в команде "Подмена", код длины команды равен 2. Команда удобна при загрузке базовых регистров и для обращения к подпрограммам.

Пример:

05 I I ПВР I, I

До выполнения команды:

регистр № I		00	00	40	04
адреса	000100 ÷ 000103	05	11	0A	1Р
текущее ССП		00	00	0C	00
		10	00	01	02

После выполнения команды:

регистр № I	I0	00	0I	02
текущее ССП	00	00	00	00
	I0	00	40	04

5.3.10 ПЕРЕДАЧА УПРАВЛЕНИЯ ПО СЧЕТЧИКУ. По команде "Передача управления по счетчику" выполняется следующая операция: определяется адрес передачи управления, затем содержимое общего регистра с номером PI (счетчик) алгебраически уменьшается на единицу; если результат равен нулю, сохраняется порядок выполнения команд; если результат не равен нулю, осуществляется передача управления. При этом уменьшение содержимого регистра PI на единицу осуществляется как вычитание целых двоичных чисел, выраженных в дополнительном коде.

При выполнении команды "Передача управления по счетчику" формата PP, если P2=0, передачи управления не происходит, но счетчик уменьшается на единицу.

Код условия остается без изменения.

Машинный формат:

PP

06	PI	P2

PИ

46	PI	И2	Б2	C2

Мнемокод-формат:

PP ПСЧР PI, P2

PИ ПСЧ PI, C2 (И2, Б2)

Прерывания программы отсутствуют.

Примечание. Начальное содержимое счетчика может быть равно нулю. В этом случае при выполнении команды содержимое счетчика станет равным минус единице и произойдет переход.

Пример:

```
46      I  0    2   I02        ПСЧ  I,X'I02'(, 2)
```

До выполнения команды:

регистр № I	00	00	02	00
регистр № 2	00	00	I0	00

После выполнения команды:

регистр № I	00	00	0I	РР
регистр № 2	00	00	I0	00
адрес передачи управления	00	II	02	

5.3.11 ПЕРЕДАЧА УПРАВЛЕНИЯ ПО ПРЕВЫШЕНИЮ. По команде "Передача управления по превышению" производится следующая операция: первый операнд алгебраически складывается с третьим и полученная сумма, имеющая длину четыре байта, сравнивается с уставкой, также имеющей длину четыре байта и расположенной в общем регистре, номер которого равен или Р3 (если Р3 нечетное, то есть третий операнд совпадает с уставкой), или Р3 + I (если Р3 четное, то есть третий операнд является шагом приближения к уставке). Если сумма больше уставки, осуществляется передача управления по исполнительному адресу передачи управления, в противном случае сохраняется нормальный порядок выполнения команд; затем код суммы запоминается на месте первого операнда. При этом сложение и сравнение операндов выполняются как сложение и сравнение целых двоичных чисел, выраженных в дополнительном коде.

Если первый операнд и уставка размещаются в одном и том же общем регистре, уставка совпадает с кодом первого операнда, а не с кодом суммы.

Код условия остается без изменения.

Машинный формат:

РП	86	Р1	Р3	Б2	С2

Мнемокод-формат:

РП ПБ Р1, Р3, С2 (Б2)

Прерывания программы отсутствуют.

Примечание. Основной целью команд является увеличение и проверка значения индекса. Приращение может быть положительным или отрицательным и иметь любое значение.

Пример:

86 А 4 I СОА ПБ I0,4,I0(I)

До выполнения команды:

регистр № I	00	00	I0	00
регистр № 4	00	52	76	AB
регистр № 5 (уставка)	I5	BD	BA	DA
регистр № А	I5	48	AI	FA

После выполнения команды:

регистр № I	00	00	0I	00
регистр № 4	00	52	76	AB
регистр № 5 (уставка)	I5	BD	BA	DA
регистр № А	I5	9C	I8	9F

Передача управления отсутствует.

5.3.12 ПЕРЕДАЧА УПРАВЛЕНИЯ ПО НЕПРЕВЫШЕНИЮ. Команда "Передача управления по непревышению" отличается от команды "Передача управления по превышению" только тем, что передача управления происходит в том случае, если сумма меньше или равна уставке. Если сумма больше уставки, сохраняется нормальный порядок выполнения команд.

Код условия не меняется.

Машинный формат:

РП	87	РI	Р3	Б2	С2

Мнемокод-формат:

РП ПМ РI, Р3, С2(Б2)

Прерывания программы отсутствуют.

Пример:

87 А4 I 004 ПМ I0,4,4(I)

До выполнения команды:

регистр № I	00	00	I0	00
регистр № 4	00	52	76	AB
регистр № 5 (уставка)	I5	8D	BA	DA
регистр № А	I5	49	AI	F4

После выполнения команды:

регистр № I	00	00	I0	00
регистр № 4	00	52	76	AB
регистр № 5 (уставка)	I5	BD	BA	DA
регистр № А	I5	9C	I8	9F
адрес перехода	00	I0	04	

5.3.13 П О Д М Е Н А. По команде "Подмена" производится следующая операция: команда, выбранная по адресу перехода, модифицируется содержимым общего регистра, заданного полем РI, и затем выполняется; после выполнения модифицированной подменной команды (если это не команда передачи управления, которая осуществляет передачу) выполняется команда, следующая непосредственно за командой "Подмена".

Модификация проводится путем логического сложения (дизъюнкции) содержимого 8-I5 разрядов команды, указанной адресом перехода, с содержимым 24-3I разряда общего регистра, заданного полем РI (если РI=0, подменная команда не модифицируется).

Модификация команды не изменяет ни содержимого общего регистра, заданного полем РI, ни подменной команды в главной памяти. Она действительна только для интерпретации выполняемой команды.

Выполнение команды, на которую указывает "Подмена", и обработка особых ситуаций производятся точно так же, как если бы указанная команда встретилась в обычной последовательности команд.

Исключение составляют случаи, когда неправильно задан адрес команды или ее длина.

Если подменной командой является команда "Передача управления с возвратом", или если возникает прерывание, то запоминается продвинутый адрес и код длины команды "Подмена".

Код условия устанавливается в зависимости от выполняемой команды.

Машинный формат:

РИ	44	РI	И2	Б2	С2

Мнемокод-формат:

РИ ПОД РI, С2 (И2, Б2)

Прерывания программы:

двойная подмена;

адресация;

спецификация.

Прерывание программы по причине "Двойная подмена" возникает в случае, если подменной командой является, в свою очередь, команда "Подмена".

Если команда "Подмена" указывает адрес команды, у которой хотя бы одно полуслово выходит за пределы главной памяти, имен-

щейся в данной модели АСВТ, происходит прерывание программы по причине "Неправильная адресация".

Если адрес подменной команды не кратен двум, происходит прерывание программы по причине "Нарушение спецификации".

Пример:

```
44      IO      50      20      ПОД   I,X'20'(, 5 )
```

До выполнения команды:

регистр № I	80	00	00	II
регистр № 5	00	00	40	00
адреса 004020 ÷ 00402I			IA	23 (подменная команда)

После выполнения команды:

регистр № I	80	00	00	II
регистр № 5	00	00	40	00
адреса 004020 ÷ 00402I			IA	23
Фактически выполняемая команда			IA	33

5.3.14 ОСОБЫЕ СЛУЧАИ ПРИ ПЕРЕДАЧАХ УПРАВЛЕНИЯ. Особые случаи вызывают прерывание программы. При прерывании текущее ССП запоминается в качестве старого ССП и выбирается новое ССП. Код прерывания в старом ССП указывает на причину прерывания. Для команд перехода возможны следующие причины прерывания.

Двойная подмена. Команда "Подмена", в свою очередь, указывает на команду "Подмена".

Адресация. Команда "Подмена" указывает адрес команды, у которой хотя бы одно полуслово выходит за пределы памяти, имеющейся в данной уставке.

Спецификация. Нечетный адрес перехода в команде "Подмена"

Все три указанных случая имеют смысл только для команды "Подмена".

Команда не выполняется. Поэтому признак результата и данные в регистрах и в памяти остаются без изменения.

Особые случаи, возникающие при выполнении команды, на которую указывает команда "Подмена", такие же, как если бы указанная команда встретилась в обычной последовательности команд. Однако в старом ССП будет запомнен адрес команды, следующей за командой "Подмена". Аналогично код длины команды в старом ССП будет равен коду длины команды "Подмена" (2).

Ограничения, налагаемые на адрес, не распространяются на компоненты, из которых формируется адрес, т.е. на содержимое поля С2 и содержимое регистра, заданного полем Б2.

Примечание. Отсутствие в имеющейся памяти ячейки с адресом, по которому передается управление, или нечетность этого адреса обнаруживаются при выполнении следующей команды, а не в процессе выполнения команды перехода.

Машинные команды

(дополнительный набор)

5.4 КОМАНДЫ АРИФМЕТИКИ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ.

5.4.1 НАЗНАЧЕНИЕ. ПРЕДСТАВЛЕНИЕ ЧИСЕЛ.

Набор команд с плавающей запятой предназначен для выполнения вычислений, в которых значения операндов изменяются в широком диапазоне и за счет автоматического масштабирования получаются результаты без потери точности.

В командах с плавающей запятой число состоит из порядка со знаком и мантиссы со знаком.

Порядок рассматривается в диапазоне от -64 до +63, как число с избытком 64, т.е. порядок, как число со знаком складывается с числом 64. Результат такого сложения называется характеристикой.

Диапазон изменения характеристик от 0 до 127 или 0000000 до 1111111 (двоичный). Порядок указывается с помощью характеристики, которая находится в разрядах 1-7.

Мантисса выражается шестнадцатеричным числом, у которого запятая находится левее самой старшей цифры. Для получения соответствующей величины числа с плавающей запятой мантисса умножается на число 16 в степени, равной порядку.

Как для положительных, так и для отрицательных величин мантисса выражается в прямом коде, различие в знаке отражается на значении знакового разряда (нулевой разряд). Нулевое значение знакового разряда соответствует положительной мантиссе, единичное значение - отрицательной мантиссе. Диапазон абсолютных величин (M) представимых нормализованных чисел с плавающей запятой составляет:

для операций с обычной точностью

$$16^{-65} \quad M \quad (1 - 16^{-6}) \cdot 16^{63}$$

для операций с повышенной точностью

$$16^{-65} \leq M \leq (1 - 16^{-14}) \cdot 16^{63}$$

или приблизительно $2{,}4 \cdot 10^{-78} \leq M \leq 7{,}2 \cdot 10^{75}$ для обоих случаев.

Число с нулевой характеристикой, нулевой мантиссой и положительным знаком называется истинным нулем.

Истинный нуль может появиться как результат арифметической операции при определенных величинах операндов. Результат операции принудительно делается равным истинному нулю, если имеет место исчезновение порядка или если мантисса результата равна нулю и не предусмотрено прерывание программы при потере значимости. Если происходит прерывание программы, то результат не делается равным истинному нулю, а характеристика и знак результата остаются без изменения. Когда результат имеет нулевую мантиссу, переполнение порядка и исчезновение порядка не вызывают прерывания программы. Когда делитель имеет нулевую мантиссу деление не выполняется; такой случай рассматривается, как некорректность деления с плавающей запятой, и происходит прерывание программы. В других случаях нулевые мантиссы и нулевые характеристики участвуют во всех арифметических операциях как обычные числа.

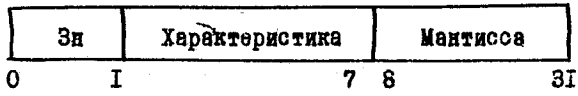
Знак суммы, разности, произведения или частного, имеющего нулевую мантиссу, всегда положителен. Знак нулевой мантиссы для других операций устанавливается в соответствии с правилами алгебры в зависимости от знаков операндов.

5.4.2 ФОРМАТ ДАННЫХ

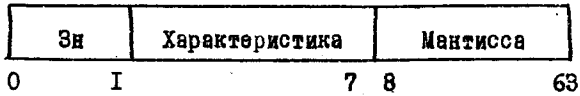
Данные в командах с плавающей запятой имеют формат фиксированной длины, который может быть в зависимости от требуемой точности вычислений, или коротким, длиной в полное слово (4 байта), или длинным, длиной в двойное слово (8 байтов). Оба формата

могут использоваться в операциях над памятью и регистрами плавающей запятой. Четырем регистрам плавающей запятой присвоены номера 0,2,4,6.

Короткое число с плавающей запятой:



Длинное число с плавающей запятой:



Набор команд с плавающей запятой включает команды с низкой и высокой точностью, для коротких и для длинных операндов.

При операциях с низкой точностью все операнды и результаты представляют собой 32-разрядные слова с плавающей запятой. В этом случае правые 32 разряда регистров плавающей запятой участия в операциях не принимают и их содержимое остается без изменения. Исключением из этого правила является произведение в операции умножения, которое представляет собой 64-разрядное слово и занимает весь регистр. При операциях с повышенной точностью все операнды и результаты являются 64-разрядными словами с плавающей запятой.

Несмотря на то, что конечные результаты в операциях с обычной точностью имеют мантиссу из 6-ти цифр, промежуточные результаты в операциях сложения, вычитания и деления могут иметь мантиссу из 7-ми цифр. Младшая цифра семиразрядной мантиссы называется дополнительной цифрой и служит для увеличения точности конечного результата. Промежуточные результаты в операциях с высокой точностью имеют в мантиссе не более I4 цифр.

Для получения максимальной точности сложение, вычитание, умножение и деление выполняются с нормализацией результатов. Сложение и вычитание могут также выполняться и без нормализации результатов. Нормализованные и ненормализованные операнды могут использоваться во всех операциях с плавающей запятой.

Нормализованное число с плавающей запятой имеет отличную от нуля старшую шестнадцатиричную цифру мантиссы.

Процессы нормализации заключаются в сдвигах мантиссы влево на одну шестнадцатиричную цифру и в уменьшении характеристики на величину, равную числу сдвигов, до тех пор, пока число не будет нормализовано. Нулевая мантисса не может быть нормализована, поэтому в таком случае сдвигов мантиссы не производится, а характеристика остается без изменения.

Обычно нормализация имеет место, когда промежуточный арифметический результат превращается в конечный результат. Такая нормализация называется нормализацией результата. При умножении и делении перед выполнением арифметической операции производится нормализация операндов. В этом случае нормализация называется предварительной нормализацией.

Операции с плавающей запятой могут выполняться с нормализацией и без нормализации. Большинство операций выполняется только одним из этих двух способов. Сложение и вычитание могут выполняться с нормализацией и без нормализации.

Как в операциях с нормализацией, так и в операциях без нормализации, исходные операнды необязательно должны быть в нормализованной форме.

При переполнении в мантиссе промежуточного результата мантисса сдвигается вправо. Если при этом длина получаемой мантиссы превосходит длину конечного результата лишние разряды отбрасываются.

Поскольку нормализация производится для шестнадцатиричных цифр, три старших двоичных разряда нормализованного числа могут быть нулями.

5.4.3 КОД УСЛОВИЯ

При выполнении операций над знаками чисел, при операциях сложения, вычитания и сравнения вырабатывается код условия. При выполнении операций умножения, деления, записи в память и засылки код условия остается без изменения.

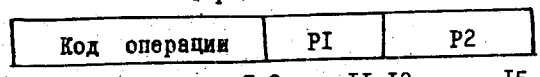
Код условия может быть использован для выбора направления при выполнении последующих команд условного перехода.

Код условия может быть установлен для того, чтобы отразить два типа результатов операции с плавающей запятой: без переполнения и с переполнением. Для большинства операций значения 0,1 или 2 указывают на то, что содержимое регистра результата соответственно равно нулю, меньше нуля или больше нуля. Нулевым результатом считается результат, имеющий нулевую мантиссу (сюда же относится и принудительный нуль). Значение 3 используется в случае переполнения порядка.

5.4.4 ФОРМАТ КОМАНД

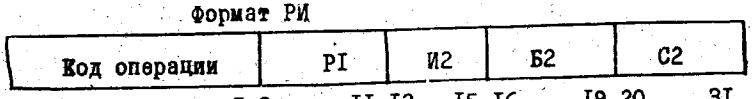
Для команд с плавающей запятой используются два формата:

Формат РР



Команда формата РР занимает два байта.

Формат РИ



Команда формата РИ занимает четыре байта.

В этих форматах первый байт содержит код операции. Р1 указывает номер регистра плавающей запятой. Содержимое этого регистра является первым операндом. Второй операнд указывается по-разному в зависимости от формата.

В формате РР Р2 указывает номер регистра плавающей запятой, в котором находится второй операнд.

В формате РИ для получения адреса второго операнда содержимое регистров индексного (И2), базового (Б2) складывается с полем смещения (С2).

Адрес второго операнда в формате РИ должен быть кратен четырем (для команд низкой точности) и восьми (для команд высокой точности). В противном случае происходит прерывание программы по причине "Нарушение спецификации".

Нулевое содержимое полей И2 или Б2 указывает на отсутствие соответствующей компоненты адреса, т.е. нулевой общий регистр не может участвовать в формировании адреса второго операнда.

Номер регистров в полях Р1 и Р2 должны быть равны 0,2,4 или 6. Если в этих полях находится другой номер регистра, такой случай рассматривается как неправильная спецификация и происходит прерывание программы.

Результаты операций помещаются на место первого операнда. Исключение представляют операции записи в память, в которых результаты помещаются на место второго операнда.

Содержимое всех общих регистров, регистров плавающей запятой и ячеек памяти, принимавших участие в формировании адреса и в выполнении операции, за исключением того регистра или ячейки, куда записывается конечный результат, остается без изменения.

Команды с плавающей запятой являются единственными командами, при выполнении которых используются регистры плавающей запятой.

5.4.5 КОМАНДЫ

Ниже в таблице приведены все команды с плавающей запятой, их мнемонические обозначения, форматы и коды операций. Все перечисленные операции относятся к средствам обработки данных с плавающей запятой и могут выполняться с обычной и повышенной точностью. В таблице указано также, когда имеет место нормализация, когда устанавливается код условия и в каких случаях происходит прерывание программы.

Название	Мнемоника	Тип		Особые случаи		Код
I	2	3		4		5
Засылка (низкая точность)	ЗНР	РР	К	А	С	38
	ЗН	РИ	К	А	С	78
Засылка (высокая точность)	ЗВР	РР	К		С	28
	ЗВ	РИ	К	А	С	68
Засылка с анализом (низкая точность)	ЗАНР	РР	К		С	32
Засылка с анализом (высокая точность)	ЗАВР	РР	КУ		С	22
Засылка дополнения (низкая точность)	ЗДНР	РР	КУ		С	33
Засылка дополнения (высокая точность)	ЗДВР	РР	КУ		С	23
Засылка модуля (низкая точность)	ЗМНР	РР	КУ		С	30
Засылка модуля (высокая точность)	ЗМВР	РР	КУ		С	20
Засылка отрицательного числа(низкая точность)	ЗОНР	РР	КУ		С	3I
Засылка отрицательного числа(высокая точность)	ЗОВР	РР	КУ		С	2I
Сложение с нормализацией (низкая точность)	СЛНР	РР	КУ Н		С П Е Б	3А
	СЛН	РИ	КУ	А	С П Е Б	7А

1	2	3			4					5
Сложение с нормализацией (высокая точность)	СЛВР СЛВ	РР РИ	КУ КУ	Н	А	С С	П П	Е Е	Б Б	2A 6A
Сложение без нормализации (низкая точность)	СЛБНР СЛБН	РР РИ	КУ КУ		А	С С		Е Е	Б Б	3E 7E
Сложение без нормализации (высокая точность)	СЛБВР СЛБВ	РР РИ	КУ КУ		А	С С		Е Е	Б Б	2E 6E
Вычитание с нормализацией (низкая точность)	ВНР ВН	РР РИ	КУ КУ	Н	А	С С	П П	Е Е	Б Б	3B 6B
Вычитание без нормализации (низкая точность)	ВБНР ВБН	РР РИ	КУ КУ		А	С С		Е Е	Б Б	3F 7F
Вычитание без нормализации (высокая точность)	ВБВР ВБВ	РР РИ	КУ КУ		А	С С		Е Е	Б Б	2F 6F
Сравнение (низкая точность)	СРНР СРН	РР РИ	КУ КУ		А	С С				39 79
Сравнение (высокая точность)	СРВР СРВ	РР РИ	К К		А	С С				29 69
Деление пополам (высокая точность)	ДПВР	РР	К			С				24
Деление пополам (низкая точность)	ДПНР	РР	К			С				34
Умножение (низкая точность)	УНР УН	РР РИ	К К	Н Н	А	С С	П П	Е Е		3C 7C
Умножение (высокая точность)	УВР УВ	РР РИ	К К	Н Н	А	С С	П П	Е Е		2C 6C
Деление (низкая точность)	ДНР ДН	РР РИ	К К	Н Н	А	С С	П П	Е Е	ПЗ ПЗ	3D 7D
Деление (высокая точность)	ДВР ДВ	РР РИ	К К	Н Н	А	С С	П П	Е Е	ПЗ ПЗ	2D 6D
Запись (низкая точность)	ЗПН	РИ	К		А	С	З			70
Запись (высокая точность)	ЗПВ	РИ	К		А	С	З			60

Примечание.

А – неправильная адресация;

С – нарушение спецификации;
З – нарушение защиты памяти;
П – исчезновение порядка;
Б – потеря значимости;
Е – переполнение порядка;
ПЗ – некорректность деления с плавающей запятой;
У – устанавливается код условия;
К – средства обработки данных с плавающей запятой;
Н – имеет место нормализация.

5.4.6 ЗАСЫЛКА

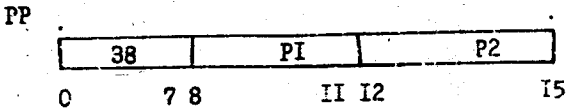
По команде "Засылка" второй операнд помещается на место первого. При операциях с низкой точностью младшие 32 разряда регистров плавающей запятой в операции не участвуют.

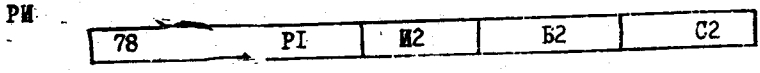
Переполнение порядка, исчезновение порядка или потери значимости произойти не может.

Код условия остается без изменения.

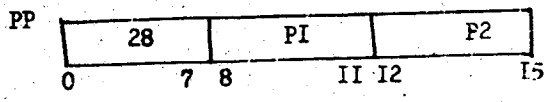
Машинный формат:

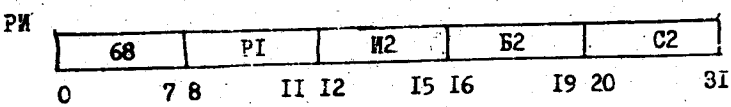
Низкая точность





Высокая точность





Мнемокод-формат:

ЗНР	Р1, Р2	низкая точность
ЗН	Р1, С2 (И2, Б2)	
ЗВР	Р1, Р2	высокая точность
ЗВ	Р1, С2 (И2, Б2)	

Прерывания программы:

Операция (если в данной установке отсутствуют средства обработки данных с плавающей запятой).

Адресация (только для ЗН и ЗВ).

Спецификация.

Примеры использования команды "Засылка".

а) Команда "Засылка" (низкая точность: формат РР)

38 20 или на мнемокоде ЗНР 2,0

До выполнения команды:

Регистр плавающей арифметики № 2	4I	II	II	II
	II	II	II	II
Регистр плавающей арифметики № 0	4I	22	22	22
	00	00	00	00

После выполнения команды:

Регистр плавающей арифметики № 2	4I	22	22	22
	II	II	II	II
Регистр плавающей арифметики № 0	4I	22	22	22
	00	00	00	00

б) Команда "Засылка" (высокая точность, формат РИ)

68 06 0 IPE, на мнемокоде - ЗВ 0,X'IP E'(6,0)

До выполнения команды:

Регистр плавающей арифметики № 0	00	00	00	00
	00	00	00	00

Общий регистр № 6							00	00	0I	02
Адреса	00	03	00	-	00	03	4I	22	22	22
	00	03	04	-	00	03	II	II	II	II

После выполнения команды:

Регистр плавающей арифметики № 0							4I	22	22	22
							II	II	II	II
Общий регистр № 6							00	00	0I	02
Адреса	00	03	00	-	00	03	4I	22	22	22
	00	03	04	-	00	03	II	II	II	II

5.4.7 ЗАСЫЛКА С АНАЛИЗОМ

По команде "Засылка с анализом" второй операнд помещается на место первого операнда и в зависимости от его знака и величины устанавливается код условия.

Второй операнд не изменяется. При операции с низкой точностью младшие 32 разряда регистра результата остаются без изменения и не проверяются.

Код условия:

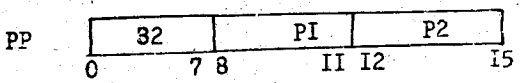
0 - мантисса результата равна нулю;

I - результат меньше нуля;

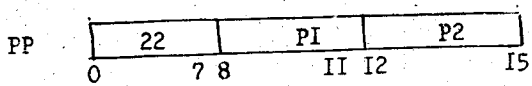
2 - результат больше нуля.

Машинный формат:

низкая точность



высокая точность



Мнемокод-формат:

ЗАНР P1, P2 - низкая точность,
ЗАВР P1, P2 - высокая точность.

Прерывания программы:

Операция (если в данной установке отсутствуют средства обработки данных с плавающей запятой).

Спецификация.

Примечание. Если один и тот же регистр указан в качестве первого и второго операнда, операция эквивалентна проверке операнда без перемещения данных.

Примеры использования команды "Засылка с анализом":

а) Команда "Засылка с анализом" (низкая точность; формат РР) 32 02, на мнемокоде ЗАНР 0,2

До выполнения команды:

Регистр плавающей арифметики № 0	42	56	I0	32
	00	00	00	00
Регистр плавающей арифметики № 2	C3	II	34	56
	00	00	00	00

После выполнения команды:

Регистр плавающей арифметики № 0	08	II	34	56
	00	00	00	00
Регистр плавающей запятой № 2	C3	II	34	56
	00	00	00	00

Код условия I

б) Команда "Засылка с анализом" (высокая точность, формат РР) 22 02, на мнемокоде ЗАВР 0,2

До выполнения команды:

регистр плавающей арифметики № 0	4I	II	II	II
	II	II	II	II

Регистр плавающей арифметики № 2	00	00	00	00
	00	00	00	00
После выполнения команды:				
Регистр плавающей арифметики № 0	00	00	00	00
	00	00	00	00
Регистр плавающей арифметики № 2	00	00	00	00
	00	00	00	00
Код условия				0

3.4.8 ЗАСЫЛКА ДОПОЛНЕНИЯ. По команде "Засылка дополнения" второй операнд помещается на место первого операнда. Значение знакового разряда второго операнда изменяется на противоположное, характеристика и мантисса не изменяется. При операции с низкой точностью младшие 32 разряда регистра результата остаются без изменения и не проверяются.

Код условия.

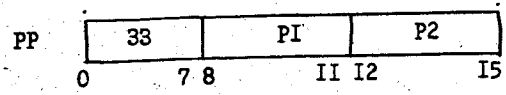
0 – мантисса равна нулю;

1 – результат меньше нуля;

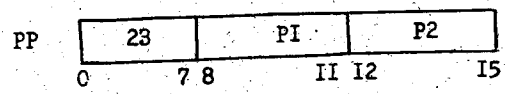
2 – результат больше нуля.

Машинный формат:

низкая точность



высокая точность



Мнемокод-формат:

ЗДНР РI, Р2 – низкая точность;

ЗДВР РI, Р2 – высокая точность.

Прерывание программы:

Операция (если в данной установке отсутствуют средства обработки данных с плавающей запятой).

Спецификация.

Примеры использования команды "Засылка дополнения".

а) Команда "Засылка дополнения" (низкая точность, формат РР)

33 20 , на мнемокоде ЗДНР 2,0

До выполнения команды:

Регистр плавающей арифметики № 2	00	00	00	00
	00	00	00	00
Регистр плавающей арифметики № 0	4I	80	00	00
	29	99	99	99

После выполнения команды:

Регистр плавающей арифметики № 2	CI	80	00	00
	00	00	00	00
Регистр плавающей арифметики № 0	4I	80	00	00
	99	99	99	99

Код условия . I

б) Команда "Засылки дополнения" (высокая точность, формат РР)

23 0 2 , на мнемокоде ЗДВР 0,2

До выполнения команды:

Регистр плавающей арифметики № 0	5D	I2	25	63
	3I	A2	38	75
Регистр плавающей арифметики № 2	C5	2I	34	AI
	42	5I	82	9I

После выполнения команды:

Регистр плавающей арифметики № 0	45	2I	34	AI
	42	5I	82	9I
Регистр плавающей арифметики № 2	C5	2I	34	AI
	42	5I	82	9I

Код условия 2

5.4.9 ЗАСЫЛКА МОДУЛЯ ЧИСЛА. По команде "Засылка модуля" второй операнд помещается на место первого операнда со знаком плюс в знаковом разряде. Значение знакового разряда второго операнда устанавливается равным нулю, характеристика и мантисса остаются без изменения. При операциях с низкой точностью младшие 32 разряда регистра результата остаются без изменения и не проверяются.

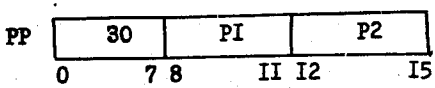
Код условия:

0 – мантисса результата равна нулю;

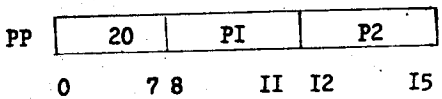
2 – результат больше нуля.

Машинный формат:

низкая точность



высокая точность



Мнемокод-формат:

ЗМНР PI, P2 – низкая точность,

ЗМВР PI, P2 – высокая точность.

Прерывание программы:

Операция (если в данной установке отсутствуют средства обработки данных с плавающей запятой).

Спецификация.

Примеры использования команды "Засылка модуля":

а) Команда "Засылка модуля" (низкая точность, формат PP)

30 20 на мнемокоде ЗМНР 2,0

До выполнения команды:

Регистр плавающей арифметики № 2	47	56	80	00
	00	00	00	00
Регистр плавающей арифметики № 0	42	25	00	00
	00	00	00	00

После выполнения команды:

Регистр плавающей арифметики № 2	42	25	00	00
	00	00	00	00
Регистр плавающей арифметики № 0	42	25	00	00
	00	00	00	00

Код условия 2

б) Команда "Засылка модуля" (высокая точность, формат РР)

20 20, на мнемокоде ЗМВР 2,0

До выполнения команды:

Регистр плавающей арифметики № 2	42	82	00	00
	00	00	00	00
Регистр плавающей арифметики № 0	DI	77	II	77
	II	II	II	II

После выполнения команды:

Регистр плавающей арифметики № 2	5I	77	II	77
	II	II	77	II
Регистр плавающей арифметики № 0	DI	77	II	77
	II	II	77	II

Код условия 2

5.4.10 ЗАСЫЛКА ОТРИЦАТЕЛЬНОГО ЧИСЛА. По команде "Засылка отрицательного числа второму операнду присваивается знак минус, и результат помещается на место первого операнда.

Значение знакового разряда второго операнда устанавливается равным единице, даже если мантисса равна нулю. Характеристика и мантисса не изменяются. При операции с низкой точностью младшие

32 разряда регистра результата остаются без изменения и не проверяются.

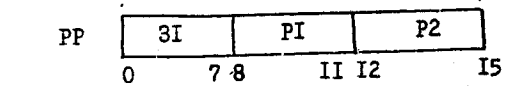
Код условия:

0 - мантисса результата равна нулю,

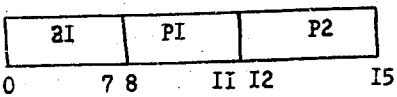
I - результат меньше нуля.

Машинный формат:

низкая точность



высокая точность



Мнемокод-формат:

ЗОНР PI, P2 - низкая точность

ЗОВР PI, P2 - высокая точность

Прерывания программы:

Операция (если в данной установке отсутствуют средства обработки данных с плавающей запятой).

Спецификация.

Примеры использования команды "Засылка отрицательного числа".

а) Команда "Засылка отрицательного числа"

(низкая точность, формат РР)

3I 20, из мнемокоде ЗОНР 2,0

До выполнения команды:

Регистр плавающей арифметики № 2 00 00 00 00
00 00 00 00

Регистр плавающей арифметики № 0	45	83	25	I0
	00	00	AB	CD
После выполнения команды:				
Регистр плавающей арифметики № 2	C5	83	25	I0
	00	00	00	00
Регистр плавающей арифметики № 0	45	83	25	I0
	00	00	AB	CD
Код условия			I	

б) Команда "Засылка отрицательного числа" (высокая точность , формат РР)

2I 20, на мнемокоде ЗОВР 2,0

До выполнения команды:

Регистр плавающей арифметики № 2	5A	48	PA	3I
	0I	48	52	32
Регистр плавающей арифметики № 0	00	00	00	00
	00	00	00	00
После выполнения команды:				
Регистр плавающей арифметики № 2	80	00	00	00
	00	00	00	00
Регистр плавающей арифметики № 0	00	00	00	00
	00	00	00	00
Код условия			0	

5.4.11 СЛОЖЕНИЕ С НОРМАЛИЗАЦИЕЙ. По команде "Сложение с нормализацией" второй операнд складывается с первым операндом и нормализованная сумма помещается на место первого операнда.

При операциях с низкой точностью содержимое младших 32 разрядов регистров плавающей запятой игнорируется и остается без изменения.

Сложение двух чисел с плавающей запятой заключается в выравнивании характеристик и сложении мантисс. Характеристики обоих операндов сравниваются, и мантисса операнда с меньшей характе-

ристикой сдвигается вправо. При каждом сдвиге на одну шестнадцатиричную цифру характеристика увеличивается на единицу. Сдвиги происходят до тех пор, пока характеристики обоих операндов станут равными. Затем происходит алгебраическое сложение мантисс, и получается промежуточная сумма. Если при сложении возникает переполнение, промежуточная сумма сдвигается вправо на одну шестнадцатиричную цифру, а характеристика увеличивается на единицу. Если это увеличение вызовет переполнение характеристики, формируется сигнал переполнения порядка и происходит прерывание программы.

При выполнении операции сложения с низкой точностью промежуточная сумма состоит из семи шестнадцатиричных цифр. Младшая дополнительная цифра появляется при сдвиге одной из мантисс вправо, только одна такая цифра принимает участие в сложении мантисс. Эта цифра равна нулю, если отсутствовало выравнивание характеристик. При выполнении операции сложения с высокой точностью промежуточная сумма состоит из I4 шестнадцатиричных цифр, а при наличии переноса - из I5 цифр. Дополнительная цифра отсутствует.

После сложения промежуточная сумма сдвигается влево на получение нормализованной мантиссы; в освобождающиеся разряды записываются нули, а характеристика уменьшается на число единиц, равное числу сдвигов.

Если нормализация вызовет исчезновение порядка, характеристике и мантиссе присваиваются нулевые значения, такая ситуация рассматривается как исчезновение порядка, и происходит прерывание программы, если соответствующий разряд маски равен единице. Если сдвиги влево промежуточной суммы отсутствовали, промежуточная сумма обрезается до нужного числа разрядов.

Если промежуточная сумма равна нулю, а разряд маски потери значимости равен единице, такая ситуация рассматривается как потеря значимости и происходит прерывание программы. Нормализация не производится; характеристика промежуточной суммы остается без изменения. Если промежуточная сумма равна нулю и разряд маски потери значимости равен нулю, прерывания программы из-за потери значимости не происходит; характеристике присваивается нулевое значение, и результатом операции является истинный нуль. При нулевой мантиссе не бывает исчезновения порядка.

Знак суммы определяется по алгебраическим правилам. Знак суммы с нулевой конечной мантиссой всегда положителен.

Код условия:

0 – мантисса результата равна нулю;

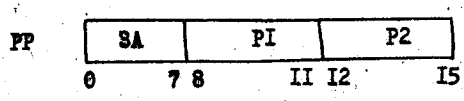
1 – результат меньше нуля;

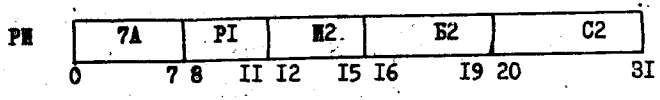
2 – результат больше нуля;

3 – переполнение порядка результата.

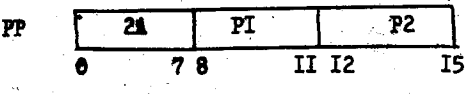
Машинный формат:

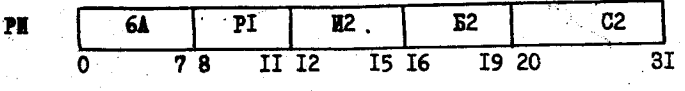
низкая точность





высокая точность





Мнемокод-формат:

СЛНР	Р1, Р2	низкая точность
СЛН	Р1, С2 (И2, Б2)	
СЛВР	Р1, Р2	высокая точность
СЛВ	Р1, С2 (И2, Б2)	

Прерывания программы:

Операция (если в данной установке отсутствуют средства обработки данных с плавающей запятой).

Адресация (только для СЛН и СЛВ).

Спецификация.

Значимость.

Переполнение порядка.

Исчезновение порядка.

Примеры использования команды "Сложение с нормализацией":

а) Команда "Сложение с нормализацией" (низкая точность, формат РР)

ЗА 24 , на мнемокоде СЛНР 2,4

До выполнения команды:

Регистр плавающей арифметики № 2	41	50	00	00
	00	00	00	00
Регистр плавающей арифметики № 4	42	25	00	00
	00	00	00	00

После выполнения команды:

Регистр плавающей арифметики № 2	42	2А	00	00
	00	00	00	00
Регистр плавающей арифметики № 4	42	25	00	00
	00	00	00	00
Код условия		2		

б) Команда "Сложение с нормализацией" (высокая точность, формат РИ).

6A 4,7 7 IPE на мнемокоде СЛВ 4,X'IPE'(7,7)

До выполнения команды:

Регистр плавающей арифметики № 4	42 44 44 44
	44 44 44 44
Общий регистр № 7	00 00 00 8I
Адреса 00 03 00 - 00 03 03	4I 55 55 55
00 03 04 - 00 03 07	55 55 55 55

После выполнения команды:

Регистр плавающей арифметики № 4	42 49 99 99
	99 99 99 99
Общий регистр № 7	00 00 00 8I
Адреса 00 03 00 - 00 03 03	4I 55 55 55
00 03 04 - 00 03 07	55 55 55 55
Код условия	2

5.4.12 СЛОЖЕНИЕ БЕЗ НОРМАЛИЗАЦИИ. По команде "Сложение без нормализации" второй операнд складывается с первым операндом, и ненормализованная сумма помещается на место первого операнда.

При операциях с низкой точностью младшие 32 разряда регистров плавающей запятой игнорируются и остаются без изменения.

После сложения промежуточная сумма обрезается до нужного числа разрядов.

Если мантисса результата равна нулю, а разряд маски потери значимости равен единице, такая ситуация рассматривается как потеря значимости и происходит прерывание программы. Если мантисса результата равна нулю и разряд маски потери значимости равен нулю, прерывания программы не происходит; характеристике присваивается нулевое значение и результатом операции является истинный нуль.

Результат не нормализуется. Исчезновение порядка произойти не может.

Знак суммы определяется по алгебраическим правилам. Знак суммы с нулевой конечной мантиссой всегда положителен.

Код условия:

0 – мантисса результата равна нулю;

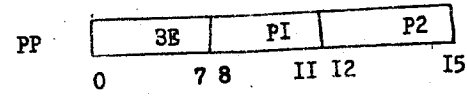
1 – результат меньше нуля;

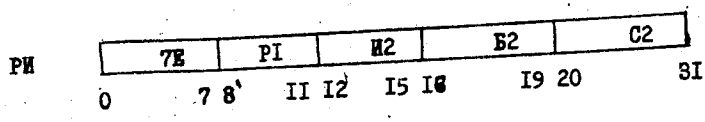
2 – результат больше нуля;

3 – переполнение порядка результата.

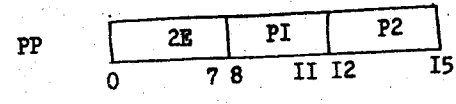
Машинный формат:

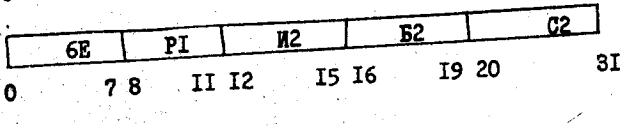
низкая точность





высокая точность





Мнемокод-формат:

СЛБНР	Р1, Р2	низкая точность
СЛБН	Р1, С2 (И2, Б2)	
СЛБВР	Р1, Р2	высокая точность
СЛБВ	Р1, С2 (И2, Б2)	

Прерывания программы:

Операция (если в данной установке отсутствуют средства обработки данных с плавающей запятой).

Адресация (только для СЛБН и СЛБВ).

Спецификация.

Значимость.

Переполнение порядка.

Примеры использования команды "Сложение без нормализации"

а) Команда "Сложение без нормализации" (низкая точность, формат РР)

3Е 26, а на мнемокоде СЛБНР 2,6

До выполнения команды:

Регистр плавающей арифметики № 2	45	02	55	50
	99	99	99	99
Регистр плавающей арифметики № 6	43	0I	I2	00
	00	00	00	00

После выполнения команды:

Регистр плавающей арифметики № 2	45	02	56	62
	99	99	99	99
Регистр плавающей арифметики № 6	43	0I	I2	00
	00	00	00	00
Код условия			2	

б) Команда "Сложение без нормализации" (высокая точность, формат РИ).

6Е 20 0 300 , на мнемокоде СЛБВ2, X'300'(0,0)

До выполнения команды:

Регистр плавающей арифметики № 2	CI	02	22	22
	22	22	22	22

Код в разряде "Потеря значимости" текущего ССП										0	
Адреса	00	03	00	-	00	03	03	4I	02	22	22
	00	03	04	-	00	03	07	22	22	22	22

После выполнения команды:

Регистр плавающей арифметики № 2								00	00	00	00
								00	00	00	00
Адреса	00	03	00	-	00	03	03	4I	02	22	22
	00	03	04	-	00	03	07	22	22	22	22
Код условия									0		

5.4.3 ВЫЧИТАНИЕ С НОРМАЛИЗАЦИЕЙ. По команде "Вычитание с нормализацией" второй операнд вычитается из первого операнда и нормализованная сумма помещается на место первого операнда.

При операциях с низкой точностью младшие 32 разряда регистров плавающей запятой игнорируются и остаются без изменения.

Команда "Вычитание с нормализацией" аналогична команде "Сложение с нормализацией", за исключением того, что перед сложением знак второго операнда изменяется на противоположный.

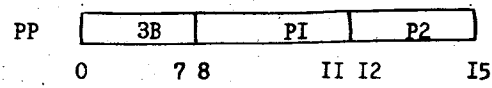
Знак разности определяется по алгебраическим правилам. Знак разности с нулевой конечной мантиссой всегда положителен.

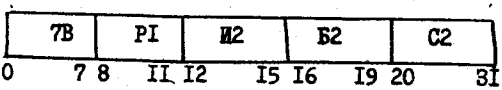
Код условия:

0 - мантисса результата равна нулю;
I - результат меньше нуля;
2 - результат больше нуля;
3 - переполнение порядка результата.

Машинный формат:

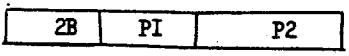
низкая точность



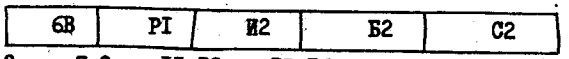


высокая точность

РР



РИ



Мнемокод-формат:

ВНР РI, Р2 — низкая точность

ВН РI, С2 (И2, Б2)

ВВР РI, Р2 — высокая точность

ВВ РI, С2 (И2, Б2)

Прерывания программы:

Операция (если в данной установке отсутствуют средства для обработки данных с плавающей запятой).

Адресация.

Спецификация.

Значимость.

Переполнение порядка.

Исчезновение порядка.

Примеры использования команды "Вычитание с нормализацией":

а) Команда "Вычитание с нормализацией" (низкая точность, формат РР)

3В 24 , на мнемокоде ВНР 2,4

До выполнения команды:

Регистр плавающей арифметики № 2 — 4I 80 00 00 00 00 00 00

Регистр плавающей арифметики № 4	4I	60	00	00
	00	00	00	00
После выполнения команды:				
Регистр плавающей арифметики № 2	4I	20	00	00
	00	00	00	00
Регистр плавающей арифметики № 4	4I	60	00	00
	00	00	00	00
Код условия		2		

б) Команда "Вычитание с нормализацией" (высокая точность, формат РИ)

6B 60 0 300, на мнемокоде BB 6,X'300'(0,0)

До выполнения команды:				
Регистр плавающей арифметики № 6	42	50	00	00
	00	00	00	00
Адреса 00 03 00 - 00 03 03	00	I0	00	00
00 03 04 - 00 03 07	00	00	00	00
После выполнения команды:				
Регистр плавающей арифметики № 6	42	50	I0	00
	00	00	00	00
Адреса 00 03 00 - 00 03 03	00	I0	00	00
00 03 04 - 00 03 07	00	00	00	00
Код условия		2		

5.4.14 ВЫЧИТАНИЕ БЕЗ НОРМАЛИЗАЦИИ. По команде "Вычитание без нормализации" второй операнд вычитается из первого операнда, и ненормализованная разность помещается на место первого операнда.

При операциях с низкой точностью младшие 32 разряда регистров плавающей запятой игнорируются и остаются без изменения.

Команда "Вычитание без нормализации" аналогична команде "Сложение без нормализации", за исключением того, что перед сложением знак второго операнда изменяется на противоположный.

Знак разности определяется по алгебраическим правилам. Знак разности с нулевой конечной мантиссой всегда положителен.

Код условия:

0 – мантисса результата равна нулю;

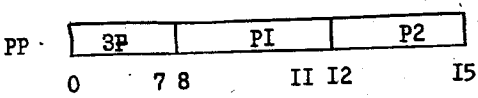
I – результат меньше нуля;

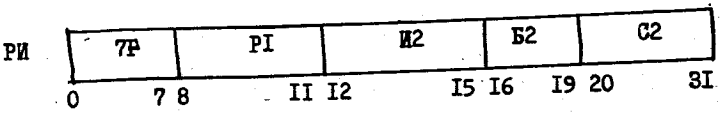
2 – результат больше нуля;

3 – переполнение порядка результата

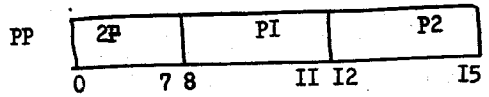
Машинный формат:

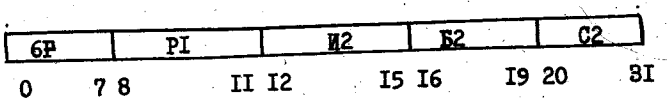
низкая точность





высокая точность





Мнемокод-формат:

ВБНР	РI, Р2	низкая точность
ББН	РI, С2 (И2, Б2)	
ВББР	РI, Р2	высокая точность
ВББ	РI, С2 (И2, Б2)	

Прерывания программы:

Операция (если в данной установке отсутствуют средства для обработки данных с плавающей запятой).

Адресация (только для ВБН, ВБВ).

Спецификация.

Значимость.

Переполнение порядка.

Примеры использования команды "Вычитание без нормализации"

а) Команда "Вычитание без нормализации".

ЗР 46, на мнемокоде ВБНР 4,6

До выполнения команды:

Регистр плавающей арифметики № 4	CI	20	00	00
	00	00	00	00
Регистр плавающей арифметики № 6	C5	22	55	40
	00	00	00	00

После выполнения команды:

Регистр плавающей арифметики № 4	45	22	55	20
	00	00	00	00
Регистр плавающей арифметики № 6	C5	22	55	40
	00	00	00	00

Код условий 2

б) Команда "Вычитание без нормализации" (высокая точность, формат РР)

2Р 20, на мнемокоде ВБВР 2,0

До выполнения команды:

Регистр плавающей арифметики № 2	CI	05	55	55
	55	55	55	55
Регистр плавающей арифметики № 0	C2	06	66	66
	66	66	66	66

После выполнения команды:

Регистр плавающей арифметики № 2	42	06	II	II
	II	II	II	II
Регистр плавающей арифметики № 0	C2	06	66	66
	66	66	66	66
Код условия		2		

5.4.15 СРАВНЕНИЕ. По команде "Сравнение" первый операнд сравнивается со вторым операндом, и в зависимости от результата сравнения устанавливается код условия.

В операциях с низкой точностью содержимое младших 32 разрядов регистров плавающей запятой игнорируется. Сравнение производится алгебраически с учетом знака мантиссы и порядка каждого числа. Неравенство порядков не является определяющим при установлении величины числа, так как мантиссы могут иметь различное количество нулевых старших цифр. Равенство устанавливается с помощью вычитания с нормализацией чисел с плавающей запятой. Если промежуточная сумма, включая дополнительную цифру, равна нулю, операнды равны. В результате операции ни один из операндов не изменяется.

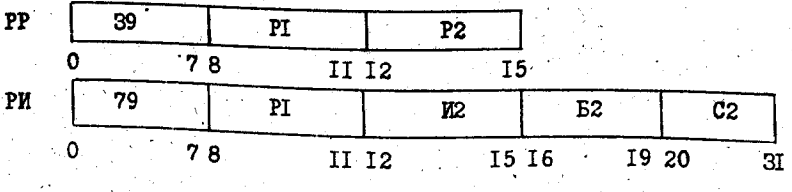
Никогда не имеет места переполнения порядка, исчезновение порядка или потеря значимости.

Код условия:

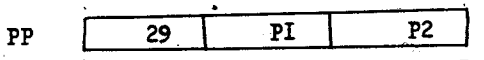
0 - операнды равны;
I - первый операнд меньше;
2 - первый операнд больше.

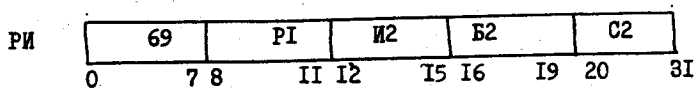
Машинный формат:

низкая точность



высокая точность





Мнемокод-формат:

СРНР	РI, Р2	низкая точность
СРН	РI, С2 (И2, Б2)	
СРВР	РI, Р2	высокая точность
СРВ	РI, С2 (И2, Б2)	

Прерывания программы:

Операция (если в данной установке отсутствуют средства обработки данных с плавающей запятой).

Адресация (только для СРН и СРВ)

Спецификация.

Примечание. Два числа с нулевыми мантиссами считаются равными, даже если у них знаки и характеристики различны.

Примеры использования команды "Сравнение".

а) Команда "Сравнение" (низкая точность формат РИ)

79 2I 7 0РА, на мнемокоде СРН 2,Х'0FA'(I,7)

До выполнения команды:

Регистр плавающей арифметики № 2	CI	I2	35	79
	00	00	00	00
Общий регистр № I	00	00	0I	02
Общий регистр № 7	00	00	0I	04
Адреса 00 03 00 - 00 03 03	5I	I3	24	52

После выполнения команды:

Регистр плавающей арифметики № 2	CI	I2	35	79
	00	00	00	00
Общий регистр № I	00	00	0I	02
Общий регистр № 7	00	09	0I	04
Адреса 00 08 00 - 00 08 08	5I	I3	24	52
Код условия		I		

б) Команда "Сравнение" (высокая точность, формат РР)

29 42, на мнемокоде СРВР 4,2

До выполнения команды:

Регистр плавающей арифметики № 4	4I	52	48	2I
	22	43	55	20
Регистр плавающей арифметики № 2	4I	I2	22	3I
	I4	5I	48	20

После выполнения команды:

Регистр плавающей арифметики № 4	4I	52	48	2I
	22	43	55	20
Регистр плавающей арифметики № 2	4I	I2	22	3I
	I4	5I	48	20
Код условия		2		

5.4.16 ДЕЛЕНИЕ ПОПОЛАМ. По команде "Деление пополам" второй операнд уменьшается вдвое, и результат помещается на место первого операнда.

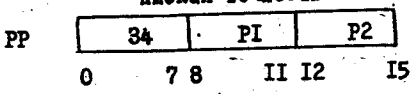
При операции с низкой точностью содержимое младших 32 разрядов регистра плавающей запятой, в котором находится результат, остается без изменения.

Операция заключается в сдвиге мантиссы вправо на один двоичный разряд; знак и характеристика не изменяются. Нормализация не производится, и отсутствует проверка на нулевую мантиссу.

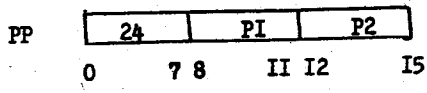
Код условия остается без изменения.

Машинный формат:

низкая точность



высокая точность



Мнемокод-формат:

ДПНР PI, P2 – низкая точность;

ДПВР PI, P2 – высокая точность.

Прерывания программы:

Операция (если в данной установке отсутствуют средства обработки данных с плавающей запятой);

Спецификация.

Примечание. Данная операция отличается от операции деления на два, отсутствием предварительной нормализации и нормализации конечного результата, а также отсутствием проверки на нулевую мантиссу.

Примеры использования команды "Деление пополам".

а) Команда "Деление пополам" (низкая точность, формат PP)

34 20, на мнемокоде ДПНР 2,0

До выполнения команды:

Регистр плавающей арифметики № 2	4I	50	00	00
	00	00	00	00
Регистр плавающей арифметики № 0	42	66	66	66
	66	66	66	66

После выполнения команды:

Регистр плавающей арифметики № 2	42	33	33	33
	00	00	00	00
Регистр плавающей арифметики № 0	42	66	66	66
	66	66	66	66

б) Команда "Деление пополам" (высокая точность, формат РР)

До выполнения команды:

Регистр плавающей арифметики № 0	00	00	00	00
	00	00	00	00
Регистр плавающей арифметики № 2	52	66	72	48
	25	43	6F	A2

После выполнения команды:

Регистр плавающей арифметики № 0	52	33	39	24
Регистр плавающей арифметики № 2	I2	AI	B7	DI
	52	66	72	48
	25	A3	6F	A2

5.4.17 УМНОЖЕНИЕ. По команде "Умножение" нормализованное произведение множителя (второй операнд) и множимого (первый операнд) помещается на место множимого.

Умножение двух чисел с плавающей запятой заключается в сложении характеристик и умножении мантисс. Из суммы характеристик вычитается число 64, и результат используется в качестве характеристики промежуточного произведения. Знак произведения определяется по алгебраическим правилам.

Мантисса произведения оказывается нормализованной за счет нормализации исходных операндов и нормализации промежуточного произведения, если в этом есть необходимость. При этом характеристика промежуточного произведения уменьшается на число единиц,

равное числу единиц влево. Для длинных операндов перед сдвигом влево, если таковой имеется, мантисса промежуточного произведения обрезается до 14 цифр. Для коротких операндов (мантисса из шести цифр) мантисса произведения имеет 14 цифр, причем две младшие цифры мантиссы всегда равны нулю.

Переполнение порядка имеет место, если значение характеристики конечного произведения превышает 127. Выполнение операции прекращается, и происходит прерывание программы. Переполнение порядка отсутствует, если значение характеристики превышает 127 только для промежуточного произведения, а характеристика конечного результата благодаря нормализации находится в пределах допустимых значений. Исчезновение порядка имеет место, если значение характеристики конечного произведения меньше нуля. В этом случае характеристике и мантиссе присваиваются нулевые значения, и происходит прерывание программы, если соответствующий разряд маски равен единице, сигнала исчезновения порядка не возникает, если меньшее нуля значение характеристики получается за счет предварительной нормализации, а в операции умножения используются допустимые значения характеристики мантиссы.

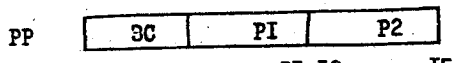
Если все 14 цифр мантиссы результата равны нулю, знаковому разряду и характеристике произведения присваиваются нулевые значения, что приводит к получению истинного нуля.

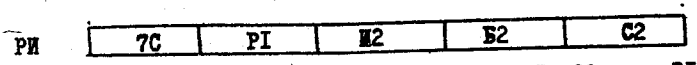
Переполнение порядка или исчезновение порядка при этом отсутствует, и прерывание программы не происходит. При умножении никогда не возникает прерывание программы из-за потери значимости.

Код условия: остается без изменения.

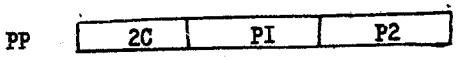
Машинный формат:

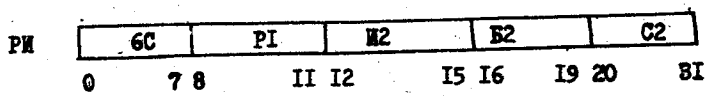
низкая точность





высокая точность





Мнемокод-формат:

УНР	РI, Р2	низкая точность
УН	РI, С2 (И2, Б2)	
УВР	РI, Р2	высокая точность
УВ	РI, М2 (И2, Б2)	

Прерывания программы:

Операция (если в данной установке отсутствуют средства обработки данных с плавающей запятой);

Адресация (только для УН, УВ)

Спецификация;

Переполнение порядка;

Исчезновение порядка.

Примеры использования команды "Умножение",

а) Команда "Умножение" (низкая точность, формат РР)

3С 24, на мнемокоде УНР 2,4

До выполнения команды:

Регистр плавающей арифметики № 2	00	РР	00	00
	00	00	00	00
Регистр плавающей арифметики № 4	7С	РР	00	00
	00	00	00	00

После выполнения команды:

Регистр плавающей арифметики 2	3С	ВЕ	0I	00
	00	00	00	00
Регистр плавающей арифметики 4	7С	РР	00	00
	00	00	00	00

б) Команда "Умножение" (высокая точность, формат РИ)

6С 00 7 080, на мнемокоде УВ 0,Х*80*(,7)

До выполнения команды:

Общий регистр 7	00	00	0I	00
Регистр плавающей арифметики 0	46	99	99	99
	00	00	00	00
Адреса 00 00 80 - 00 00 83	46	99	99	99
00 00 84 - 00 00 87	00	00	00	00

После выполнения команды:

Общий регистр 7	00	00	0I	00
Регистр плавающей арифметики 0	4С	5С	28	Р5
	05	3D	7I	00
Адреса 00 00 80 - 00 00 83	46	99	99	99
00 00 84 - 00 0С 87	00	00	00	00

5.4.18 ДЕЛЕНИЕ. По команде "Деление" делимое (первый операнд) делится на делитель (второй операнд) и заменяется частным. Остаток не сохраняется.

В операциях с низкой точностью содержимое младших 32 разрядов регистров плавающей запятой игнорируется и остается без изменения.

Деление с плавающей запятой заключается в вычитании характеристик и делении мантисс. Сумма числа 64 и разности характеристик делимого и делителя используется как характеристика промежуточного частного. Знак частного определяется по алгебраическим правилам.

Мантисса частного оказывается нормализованной за счет предварительной нормализации исходных операндов. Необходимость в нормализации промежуточного частного никогда не возникает, хотя может потребоваться сдвиг вправо. При сдвигах характеристика промежуточного частного корректируется. В образовании частного принимают участие все цифры мантиссы делимого, даже если нормализованная мантисса делимого больше нормализованной мантиссы делителя. Мантисса частного обрезается до нужного числа цифр.

Если значение характеристики конечного частного превышает 127, происходит прерывание программы из-за переполнения порядка. Выполнение операции в этом случае прекращается.

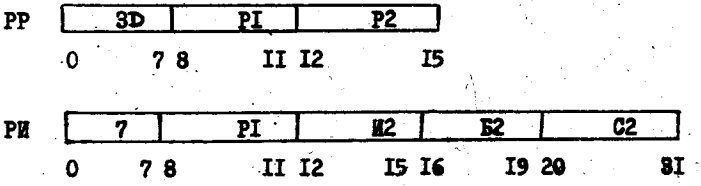
Если значение характеристики конечного частного меньше нуля, происходит прерывание программы из-за исчезновения порядка. Характеристике, знаковому разряду и мантиссе присваивается нулевые значения: и происходит прерывание программы, если соответствующий разряд маски равен единице. Сигнал об исчезновении порядка не возникает для промежуточного частного и при предварительной нормализации операндов.

При попытке разделить на нуль операция не выполняется. Делимое остается без изменения, и происходит прерывание программы из-за некорректности деления с плавающей запятой. Если мантисса делимого равна нулю, мантисса частного также будет равна нулю. Знаковому разряду и характеристике частного присваиваются нулевые значения, что приводит к получению в качестве результата операции истинного нуля. При этом отсутствуют прерывания программы из-за переполнения или исчезновения порядка. При операции деления никогда не имеет места прерывание из-за потери значимости.

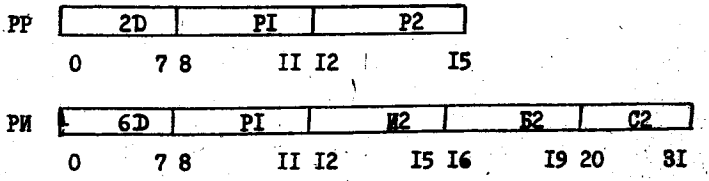
Код условия: остается без изменения.

Машинный формат:

низкая точность



высокая точность



Мнемокод-формат:

ДНР	Р1, Р2	низкая точность
ДН	РІ, С2 (И2, Б2)	
ДВР	РІ, Р2	высокая точность
ДВ	РІ, С2 (И2, Б2)	

Прерывания программы:

Операция (если в данной установке отсутствуют средства обработки данных с плавающей запятой).

Адресация (только для ДН и ДВ).

Спецификация.

Переполнение порядка.

Исчезновение порядка.

Деление с плавающей запятой.

Примеры использования команды "Деление"

а) Команда "Деление" (низкая точность, формат РР)

3D 26, на мнемокоде ДНР 2,6.

До выполнения команды:

Регистр плавающей арифметики 2	45	5Р	5Е	I0
	00	00	00	00
Регистр плавающей арифметики 6	43	27	I0	00
	00	00	00	00

После выполнения команды:

Регистр плавающей арифметики 2	43	27	I0	00
	00	00	00	00
Регистр плавающей арифметики 6	43	27	I0	00
	00	00	00	00

б) Команда "Деление" (высокая точность; формата РР)

2D 20, на мнемокоде ДВР 2,0

До выполнения команды:

Регистр плавающей арифметики 2	4C	5C	28	Р5
	05	3D	7I	00
Регистр плавающей арифметики 0	46	99	99	99
	00	00	00	00

После выполнения команды:

Регистр плавающей арифметики 2	46	99	99	99
	00	00	00	00
Регистр плавающей арифметики 0	46	99	99	99
	00	00	00	00

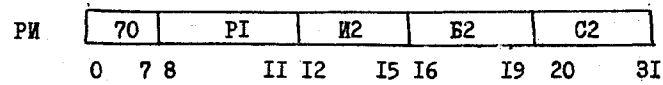
5.4.19 ЗАПИСЬ. По команде "Запись" первый операнд записывается в память по адресу второго операнда.

В операциях с низкой точностью содержимое младших 32 разрядов регистра первого операнда игнорируется. Первый операнд остается без изменения.

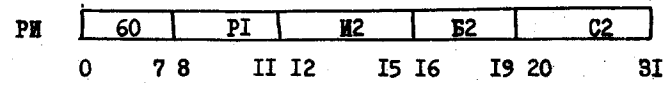
Код условия: остается без изменения.

Машинный формат:

низкая точность



высокая точность



Мнемокод-формат:

ЗПН РI, С2 (И2, Б2) - низкая точность;

ЗПВ РI, С2 (И2, Б2) - высокая точность.

Прерывание программы:

Операция (если в данной установке отсутствуют средства обработки данных с плавающей запятой).

Адресация.

Защита памяти

Спецификация.

Примеры использования команды "Запись"

а) Команда "Запись" (низкая точность, формат РИ)

7D 2I P IPC, на мнемокоде ЗПН 2,X'IPC'(I,I5)

До выполнения команды:

Регистр плавающей арифметики № 2	5	2I	32	45
	II	23	43	PA
Общий регистр № I	00	00	0I	04
Общий регистр № P	00	00	I0	00
Адреса 00 I3 00 - 00 I3 03	4I	80	00	00
00 I3 04 - 00 I3 07	99	99	99	99

После выполнения команды:

Регистр выполнения арифметики № 2	5D	2I	32	45
	II	23	43	PA
Общий регистр № I	00	00	0I	04
Общий регистр № P	00	00	I0	00
Адреса 00 I3 00 - 00 I3 03	5D	2I	32	45
00 I3 04 - 00 I3 07	99	99	99	99

б) Команда "Запись" (высокая точность, формат РИ)

60 4I 3 IPC, на мнемокоде ЗПВ 4,X'IPC'(I,3)

До выполнения команды:

Регистр плавающей арифметики № 4	4I	58	4A	I4
	2I	55	BC	IP
Общий регистр № I	00	00	0I	04
Общий регистр № 3	00	00	I0	00
Адреса 00 I3 00 - 00 I3 03	5D	2I	32	45
00 I3 00 - 00 I3 07	99	99	99	99

После выполнения команды:

Регистр плавающей арифметики № 4	4I	58	4A	I4
	2I	55	BC	IP
Общий регистр № I	00	00	0I	04
Общий регистр № 3	00	00	I0	00
Адреса 00 I3 00 - 00 I3 03	4I	58	4A	I4
00 I3 04 - 00 I3 07	2I	55	BC	IP

3.4.20 ОСОБЫЕ СЛУЧАИ ПРИ ВЫПОЛНЕНИИ ОПЕРАЦИЙ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ. Неправильные команды, данные или результаты вызывают прерывание программы. При прерывании текущее ССП запоминается в качестве старого ССП и выбирается новое ССП. Код прерывания в старом ССП указывает на причину прерывания. При операциях с плавающей запятой возможны следующие особые случаи, вызывающие прерывания программы.

Операция. Производится попытка выполнить команду с плавающей запятой, а средства обработки данных с плавающей запятой в данной установке отсутствуют. Команда не выполняется. Код условия и данные в регистрах и в памяти остаются без изменения.

Защита памяти. Ключ памяти для ячейки результата не совпадает с ключом защиты, находящимся в ССП. Операция не выполняется, поэтому код условия и данные в регистрах и в памяти остаются без изменения.

Адресация. Адрес указывает на ячейку, которая отсутствует в памяти данной установки. Выполнение операции прекращается. Получаемые данные и код условия, если он изменяется, нельзя предсказать заранее, поэтому их нельзя использовать в дальнейших вычислениях.

Спецификация. Короткий операнд не располагается в границах 32-разрядных слов или длинный операнд не располагается в границах 64-разрядных слов, или при указании номера регистра плавающей запятой указан номер регистра, отличный от 0,2,4 или 6. Команда не выполняется, поэтому код условия и данные в регистрах и в памяти остаются без изменения. Ограничения на адрес не распространяются на компоненты адреса, т.е. на содержимое поля С2 и содержимое регистров, заданных полями И2 и Б2.

Переполнение порядка. В порядке результата при сложении, вычитании, умножении и делении возникает переполнение, а мантисса результата отлична от нуля. Выполнение операции прекращается, получаемые данные нельзя предсказать заранее, и поэтому они не должны использоваться в последующих вычислениях. Значение кода условия устанавливается равным 3 при сложении и вычитании, а при умножении и делении остается без изменения.

Исчезновение порядка. При сложении, вычитании, умножении или делении происходит исчезновение порядка, а мантисса результата отлична от нуля. Если разряд маски исчезновения порядка равен единице, происходит прерывание программы. Выполнение операции завершается тем, что на место результата записывается истинный нуль. Значение кода условия устанавливается равным нулю при сложении и вычитании, а при умножении и делении остается без изменения. Значение разряда маски не влияет на результат.

Значимость. При сложении или вычитании мантисса результата равна нулю. Если разряд маски потери значимости равен единице, происходит прерывание программы. Кроме того, значение разряда маски влияет на результат операции. При нулевом значении разряда маски выполнение операции завершается тем, что на место резуль-

тата записывается истинный нуль. При единичном значении разряда маски операция заканчивается без последующего изменения характеристик результата. В обоих случаях значение кода условия устанавливается равным нулю.

Деление с плавающей запятой. Производится попытка деления на делитель с нулевой мантиссой. Операция деления не выполняется, поэтому код условия и данных в регистрах и в памяти остается без изменения.

5.5 КОМАНДЫ ДЕСЯТИЧНОЙ АРИФМЕТИКИ

5.5.1 НАЗНАЧЕНИЕ

В операциях над десятичными числами используется упакованный формат данных. В этом формате две десятичные цифры помещены в один восьмиразрядный байт. Так как данные при обменах с внешними устройствами часто передаются в формате с зоной (в одном восьмиразрядном байте находится одна цифра), в эту группу команд включены операции преобразования формата.

Данные рассматриваются как целые числа, выравненные по правым концам соответствующих полей. Числа представлены в прямом коде со знаком в самом правом восьмиразрядном байте.

Обработка ведется в ячейках основной памяти справа налево. Все команды десятичной арифметики используют двухадресный формат. В адресе задается самый левый байт операнда. В длине поля, связанной с этим адресом, указывается общее число байтов в операнде.

Набор команд десятичной арифметики обеспечивает сложение, вычитание, сравнение, умножение и деление, а также преобразование формата операндов переменной длины. Большинство десятичных команд относится к средствам обработки десятичных данных.

В результате всех операций типа сложения и сравнений устанавливается признак результата, т.е. вырабатывается код условия.

5.5.2 ФОРМАТ ДАННЫХ

Десятичные операнды могут располагаться только в основной памяти. Поля, которые они занимают, могут начинаться с любого байта и иметь длину от одного до 16 восьмиразрядных байтов. Два операнда, определяемые в одной команде, могут иметь разную длину. Предполагается, что при необходимости они дополняются нулями слева от старшей цифры. Результаты никогда не выходят

за пределы, устанавливаемые адресом и длиной. О потере перенос-
сов или о потере значащих цифр сигнализируется как о десятичном
переполнении. Операнды могут быть как в упакованном формате, так
и в формате с зоной.

Упакованное десятичное число:

Цифра	Цифра	Цифра	...	Цифра	Цифра	Цифра	Цифра	Знак

В упакованном формате две десятичные цифры расположены ря-
дом в одном байте. Исключение составляет самый правый байт, в
котором справа от цифры помещен знак. И цифры, и знак представ-
лены в виде четырехразрядных кодов.

Десятичное число в формате с зоной:

Зона	Цифра	Зона	...	Цифра	Зона	Цифра	Знак	Цифра

В формате с зоной младшие четыре разряда байта, числовые
разряды, обычно заняты десятичной цифрой. Старшие четыре разряда
называются зоной; исключение составлнет самый правый байт поля,
в котором место зоны обычно занято знаком.

В десятичных операциях и операнды, и результаты представлены
в упакованном формате. В формате с зоной цифра рассматривается как
буквенно-числовой символ. Среди команд предусмотрены команда
"Перевод в уплотненный формат", которая преобразует данные с
зоной в упакованные данные, и команда "Перевод в зонированный
формат", которая выполняет обратное преобразование. Кроме того,
чтобы преобразовать данные из упакованного формата в формат с
зоной, можно использовать команды редактирования.

Во всех командах десятичной арифметики, за исключением ко-
манд "Перевод в уплотненный формат", "Перевод в зонированный
формат" и "Перемещение со сдвигом", поля либо не должны перекры-
ваться совсем, либо у них должны совпадать самые правые байты.

В команде "Очистка со сложением десятичным" поле результата может перекрываться с полем исходных данных, но при этом самый правый байт первого операнда должен находиться правее самого правого байта второго операнда или совпадать с ним.

Во время выполнения операции коды цифр и знака проверяются. Поэтому, если поля операндов перекрываются неправильно, т.е. положение знака одного операнда будет совпадать с положением цифры другого операнда, такая ситуация будет обнаружена и классифицирована как некорректность данных. В пересылочных операциях цифры и знаки операндов не проверяются и поля операндов могут перекрываться произвольным образом.

Правила обработки перекрывающихся полей установлены в предположении, что операнды выбираются справа налево из памяти, по восемь разрядов за один раз, непосредственно перед тем как они обрабатываются. Точно так же результаты помещаются в память, как только они получены, по восемь разрядов за один раз. В действительности процедура обработки может существенно отличаться от указанной из-за использования специальной памяти для промежуточных результатов. Тем не менее во всех случаях соблюдаются сформулированные выше правила.

5.5.3 ПРЕДСТАВЛЕНИЕ ЧИСЕЛ

Числа представляются в виде выравненных по правым концам целых чисел в прямом коде со знаком плюс или минус.

Цифры 0 ÷ 9 имеют двоичные коды 0000 ÷ 1001. Коды 1010 и 1011 используются только как коды знака, причем код 1010 рассматривается как плюс , а код 1011 - как минус. Коды 0000 ÷ 1001 недопустимы в качестве кодов знака. Так как при переводе данных из формата с зоной в упакованный формат зоны отбрасываются, их коды на допустимость не проверяются. При всех десятич-

ных операциях получаемые в результате коды знака и зоны различаются. Код зоны в АСВТ - 0101.

5.5.4 КОД УСЛОВИЯ (ПРИЗНАК РЕЗУЛЬТАТА)

В процессе выполнения всех операций типа сложения и сравнений устанавливается код условия. Все другие операции десятичной арифметики код условия не меняют. Код условия может быть использован для выбора направления в последующих командах передачи управления.

В десятичной арифметике код условия может быть установлен для того, чтобы отразить два типа результатов. Для большинства операций состояния 0,1 и 2 указывают соответственно на нулевое, меньшее нуля и большее нуля содержимое поля результата: состояние 3 используется, когда происходит переполнение.

Для операций сравнения состояния 0, 1 и 2 указывают на то, что первый из сравниваемых операндов соответственно равен, меньше или больше второго.

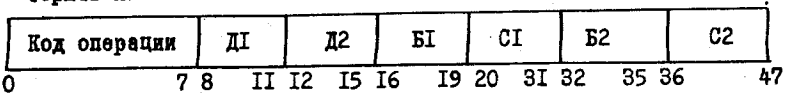
Значение кода условия для операций десятичной арифметики.

Сложение десятичное	нуль	<нуля	>нуля	переполнение
Сравнение десятичное	равны	меньше	больше	-
Вычитание десятичное	нуль	<нуля	>нуля	переполнение
Очистки со сложением десятичным	нуль	<нуля	>нуля	переполнение

5.5.4а ФОРМАТ КОМАНД

Десятичные команды используют следующий формат:

Формат ПП



В этом формате, чтобы получить адрес, содержимое общего регистра, определяемого Б1, складывается с содержимым поля С1. Этот адрес указывает самый левый байт поля первого операнда. В поле Д1 указывается число байтов операнда, находящихся справа от этого байта. Следовательно, длина поля первого операнда в байтах может быть равна 1 + 16 в соответствии с интервалом значений (0000 ÷ 1111). Точно так же поля Д2, Б2 и С2 определяют поле второго операнда.

Нуль в поле Б1 или Б2 указывает на отсутствие соответствующей компоненты адреса.

Результаты операций всегда помещаются в поле первого операнда. Результат никогда не выходит за пределы поля, определенного адресом и длиной. Если первый операнд длиннее чем второй, то второй операнд дополняется нулями в недостающих разрядах.

Такое расширение никогда не отражается на состоянии памяти. Поле второго операнда и содержимое всех общих регистров не меняется.

5.5.5 КОМАНДЫ

Ниже приведены команды десятичной арифметики, их мнемонические обозначения и коды операций. Во всех командах используется формат ПП и предполагается, что операнды и результаты представлены в упакованном формате. Единственное исключение представляют команда "Перевод в уплотненный формат", у которой операнд имеет формат с зоной, и команда "Перевод в зонированный формат", у которой результат представлен в формате с зоной.

В таблице отмечены те команды, которые относятся к средствам обработки десятичных данных, указано, когда устанавливается код условия, а также перечислены особые случаи, вызывающие прерывания программы.

Название	Мнемоника	Тип		Особые случаи				Код
Сложение десятичное	СЛД	ПП	ТУ	А	Д	З	ДП	FA
Вычитание десятичное	ВД	ПП	ТУ	А	Д	З	ДП	FB
Очистка со сложением десятичным	ОСЛД	ПП	ТУ	А	Д	З	ДП	F8
Сравнение десятичное	СРД	ПП	ТУ	А	Д			F9
Умножение десятичное	УД	ПП	Т	А	СД	З		FC
Деление десятичное	ДД	ПП	Т	А	СД	З	ДД	FD
Перевод в уплотненный формат	ПУФ	ПП		А		З		F2
Перевод в зонированный формат	ПЗФ	ПП		А		З		F8
Перемещение со сдвигом	ПСД	ПП		А		З		F1

Примечание. А – неправильная адресация.

У – Устанавливается признак результата.

Д – Неправильные данные

ДП– Десятичное переполнение

ДД– Недопустимое деление десятичное

З – Нарушение защиты памяти

С – Нарушение спецификации

Т – Команда относится к средствам обработки десятичных данных.

При обработке десятичных данных могут быть также использованы команды пересылки, редактирования и логического сравнения.

5.5.6 СЛОЖЕНИЕ ДЕСЯТИЧНОЕ. По команде "Сложение десятичное" выполняется следующая операция: операнды складываются, начиная с младших десятичных разрядов, с учетом знаков, и сумма помещается на место первого операнда. Все знаки и цифры операндов проверяются на допустимость кодов. Недопустимыми считаются коды A ÷ F во всех полубайтах, кроме знакового и коды 0 ÷ 9 в знаковом полубайте.

Если сумма превышает по длине первый операнд (второй операнд длиннее первого, или же при сложении возник перенос из старшего разряда), на место первого операнда записывается только соответствующее число младших разрядов суммы. Если среди отброшенных при этом старших десятичных разрядов был хотя бы один отличный от нуля, имеет место переполнение десятичное.

Поля первого и второго операндов могут перекрываться только в том случае, если их младшие байты совпадают.

В младшем полубайте записывается знак суммы. Если сумма равна нулю, знак ее "плюс". Но при переполнении в знаковом полубайте записывается истинный знак суммы, даже если все записанные в поле первого операнда десятичные цифры - нули.

При выполнении команды "Сложение десятичное" формируются следующие коды условия:

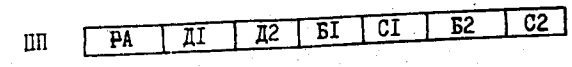
0 - если результат равен нулю;

1 - если результат меньше нуля;

2 - если результат больше нуля;

3 - если имело место десятичное переполнение.

Машинный формат:



Мнемокод -формат:

СЛД C1(Д1, Б1), C2 (Д2,Б2).

Прерывания программы:

операция (если отсутствуют средства обработки десятичных данных);

защита памяти;

адресация;

данные;

десятичное переполнение.

Если адрес какого-либо байта при выполнении команды "Сложение десятичное" соответствует несуществующим ячейкам главной памяти, возникает прерывание программы по причине "Неправильная адресация". Если ненулевой программный ключ не совпал с ненулевым ключом страницы главной памяти, содержащей первый операнд, возникает прерывание программы по причине "Нарушение защиты памяти".

Если при выполнении команды "Сложение десятичное" были обнаружены недопустимые коды, имеет место прерывание по причине "Неправильные данные".

Если результирующий код условия равен 3 и соответствующий разряд маски программы хранит код "1", возникает прерывание по причине "Десятичное переполнение".

Пример:

```
РА    32    Р    0РЕ  Р I02   СЛД X'РЕ'(4,I5),X'I02'(3,I5).
```

До выполнения операции:

```
регистр № Р                    00  00  0I  02
адреса  000200 ÷ 000203        94  I2  37  0A
        000204 ÷ 000207        I2  34  5A  4I
```

После выполнения операции:

регистр № Р	00	00	0I	02
адреса 000200 ÷ 000203	94	24	7I	5A
000204 ÷ 000207	I2	34	5A	4I

Код условия 2

5.5.7 ВЫЧИТАНИЕ ДЕСЯТИЧНОЕ. По команде "Вычитание десятичное" выполняется точно такая же операция, как по команде "Сложение десятичное", но предварительно знак второго операнда изменяется на противоположный.

При выполнении команды "Вычитание десятичное" формируются следующие коды условия:

0 – если результат равен нулю;

I – если результат меньше нуля;

2 – если результат больше нуля;

3 – если имело место десятичное переполнение.

Машинный формат:

Ш	РВ	Д1	Д2	Б1	С1	Б2	С2

Мнемоход-формат:

ВД С1 (Д1, Б1), С2 (Д2, Б2).

Прерывания программы:

операция (если отсутствуют средства обработки десятичных данных);

защита памяти;

адресация;

данные;

десятичное переполнение.

При выполнении команды "Вычитание десятичное" прерывания программы аналогичны прерываниям при выполнении команды "Сложе -

ние десятичное".

Примечание. Операнды в команде "Вычитание десятичное", даже когда их длины не равны, могут перекрываться, но так, чтобы их младшие байты совпадали. Это свойство может быть использовано для того, чтобы установить нуль или все поле целиком, или его младшую, правую часть.

Пример:

```
PB    45    7   I25   7   I2A   ВД X'I25'(5,7),X'I2A'(6,7)
```

До выполнения команды:

регистр № 7		00	00	0I	I0
адреса	000234 ÷ 000237	00	I2	34	56
	000238 ÷ 00023B	78	9B	65	4I
	00023C ÷ 00023F	32	89	I4	6B

После выполнения команды:

регистр № 7		00	00	0I	I0
адреса	000234 ÷ 000237	00	28	98	32
	000238 ÷ 00023B	35	7A	65	4I
	00023C ÷ 00023F	32	89	I4	6B
Код условия		3			

5.5.8 ОЧИСТКА СО СЛОЖЕНИЕМ ДЕСЯТИЧНЫМ. По команде "Очистка со сложением десятичным" выполняется следующая операция.

Второй операнд помещается на место первого операнда. Нулевой результат положительный. Если в результате переполнения теряются цифры старшего порядка, нулевой результат имеет знак второго операнда. Результат операции полностью аналогичен результату операции "Сложение десятичное", если все десятичные разряды первого операнда равны нулю. На допустимость кодов проверяется только второй операнд.

Поля первого и второго операнда могут перекрываться, но так, чтобы крайний правый байт первого операнда совпадал с крайним правым байтом второго операнда или находился справа от него.

При выполнении команды "Очистка со сложением десятичным" вырабатываются следующие коды условия:

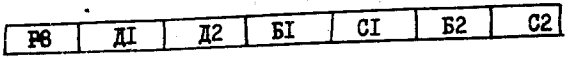
0 - если результат равен нулю;

I - если результат меньше нуля;

2 - если результат больше нуля;

3 - если имело место переполнение.

Машинный формат:



Мнемокод-формат:

ОСЗД С1(Д1,Б1), С2(Д2,Б2)

Прерывания программы:

операция (если отсутствуют средства обработки десятичных данных);

защита памяти;
адресация;
данные;
десятичное переполнение.

Если адрес какого-либо байта при выполнении команды "Очистка со сложением десятичным" соответствует несуществующим ячейкам главной памяти, возникает прерывание программы по причине "Неправильная адресация". Если ненулевой программный ключ не совпал с ненулевым ключом страницы главной памяти, содержащей первый операнд, возникает прерывание программы по причине "Нарушение защиты памяти".

Если при выполнении команды "Очистка со сложением десятичным" были обнаружены недопустимые коды, имеет место прерывание по причине "Неправильные данные".

Если результирующий код условия равен 3 и соответствующий разряд маски программы в текущем ССП хранит код "I", возникает прерывание по причине "Десятичное переполнение".

Пример:

```
F8    32    5    256  5  25A   ОСЛД X'256'(4,5),X'25A'(3,5)
До выполнения команды:
регистр № 5                    00    00    00    I0
адреса 000264 ÷ 000267         2I    4A    5I    I0
       000268 ÷ 00026B         23    48    I4    98
       00026C ÷ 00026F         5A    32    67    02
После выполнения команды:
регистр № 5                    00    00    00    00
адреса 000264 ÷ 000267         2I    4A    00    I4
       000268 ÷ 00026B         98    5A    I4    98
       00026C ÷ 00026F         5A    32    67    02
код условия                          2
```

5.5.9 СРАВНЕНИЕ ДЕСЯТИЧНОЕ. По команде "Сравнение десятичное" выполняется следующая операция: операнды сравниваются с учетом знаков и всех цифр обоих операндов. Сравнение производится справа налево. Сами операнды сохраняются в памяти без изменения. Все знаки и цифры первого и второго операндов проверяются на допустимость кодов. Положительный нуль считается равным отрицательному. Поля операндов могут перекрываться, но так, чтобы их младшие байты совпадали.

При выполнении команды "Сравнение десятичное" вырабатываются следующие коды условия:

0 – если операнды равны;

I – если первый операнд меньше второго;

2 – если первый операнд больше второго.

Машинный формат:

ПП	Р9	Д1	Д2	Б1	С1	Б2	С2

Мнемокод-формат:

СРД С1 (Д1, В1), С2 (Д2,Б2).

Прерывания программы:

операция (если отсутствуют средства обработки десятичных данных);

адресация;

данные.

Если адрес какого-либо байта при выполнении команды "Сравнение десятичное" соответствует несуществующим ячейкам главной памяти, возникает прерывание программы по причине "Неправильная адресация". Если при выполнении команды "Сравнение десятичное" были обнаружены недопустимые коды, имеет место прерывание по причине "Неправильные данные".

Примечание. Команда "Сравнение десятичное" - единственная среди команд сравнения, в которой обработка ведется справа налево с учетом знаков и нулей, с анализом правильности кодировки цифр и знаков и, наконец, с дополнением короткого числа нулями слева.

Пример:

Р9 32 5 I8D 5 I9I СРД X'I8D'(4,5),X'I9I'(3,5)

До выполнения команды:

регистр № 5	00	00	I0	I0
адреса 00II9C ÷ 00II9F	0I	I2	34	58
00IIA0 ÷ 00IIA3	4A	28	95	6A

После выполнения команды:

регистр № 5	00	00	I0	I0
адреса 00II9C ÷ 00II9F	0I	I2	34	58
00IIA0 ÷ 00IIA3	4A	28	95	6A

Код условия 2

5.5.10 УМНОЖЕНИЕ ДЕСЯТИЧНОЕ. По команде "Умножение десятичное" выполняется следующая операция: произведение множимого (первый операнд) на множитель (второй операнд) замещает множимое. Операция выполняется справа налево. Знак произведения определяется правилами алгебры в соответствии со знаками сомножителей, даже если один или оба операнда равны нулю.

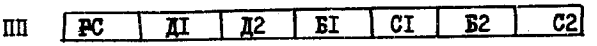
Длина множителя не должна превышать восемь байтов (15 десятичных цифр со знаком).

Множимое должно иметь в старших разрядах столько нулей, сколько разрядов в множителе.

Поля множимого и множителя могут перекрываться, но так, чтобы совпадали их младшие байты.

При выполнении команды "Умножение десятичное" результирующий код условия не изменяется.

Машинный формат:



Мнемокод-формат:

УД С1 (Д1,Б1), С2 (Д2,Б2).

Прерывания программы:

операция (если отсутствуют средства обработки десятичных данных);

защита памяти;

адресация;

спецификация;

данные.

Если адрес какого-либо байта при выполнении команды "Умножение десятичное" соответствует несуществующим ячейкам главной памяти , возникает прерывание программы по причине "Неправильная адресация".

Если ненулевой программный ключ не совпал с ненулевым ключом страницы главной памяти, содержащей первый операнд, возникает прерывание программы по причине "Нарушение защиты памяти".

Если код длины второго операнда Д2 равен или больше кода длины первого операнда Д1, то возникает прерывание программы по причине "Нарушение спецификации".

Если при выполнении команды "Умножение десятичное" были обнаружены недопустимые коды, имеет место прерывание программы по причине "Неправильные данные".

Если множимое не имеет в старших разрядах столько нулей, сколько разрядов в множителе, возникает прерывание программы по причине "Неправильные данные".

Примечание: Чтобы множимое имело необходимое количество старших нулей, команде "Умножение десятичное" может предшествовать команда "Очистка со сложением десятичным" (ОСЛД).

Пример:

РС 32 Р 24В Р 24Р УД X'24B'(4,I5),X'24P'(3,I5)

До выполнения команды:

регистр № Р		00	00	0I	02
адреса 00034C + 00034P		42	00	00	00
000350 + 000353		5A	00	02	IB

После выполнения команды:

регистр № Р		00	00	0I	02
адреса 00034C + 00034P		42	00	00	I0
000350 + 000353		5B	00	02	IB

5.5.11 ДЕЛЕНИЕ ДЕСЯТИЧНОЕ. По команде "Деление десятичное" (ДД) выполняется следующая операция: первый операнд (делимое) делится на второй операнд (делитель) и замещается частным и остатком.

Частное помещается в левой части поля первого операнда. Адрес его равен адресу делимого, а длина равна разности между длинами

делимого и делителя. Остаток помещается в правой части поля первого операнда. Адрес остатка равен сумме адреса делимого и длины частного. Длина остатка равна длине делителя (второго операнда). Следовательно, частное и остаток вместе занимают все поле делимого.

Делитель должен быть не длиннее восьми байтов и обязательно короче делимого.

Знак частного определяется правилами алгебры в соответствии со знаком делимого и делителя. Знак остатка совпадает со знаком делимого. Это правило остается в силе и тогда, когда частное или остаток равны нулю.

Поле делимого и делителя могут перекрываться, но так, чтобы совпадали их младшие байты.

При выполнении команды "Деление десятичное" результирующий код условия не изменяется.

Машинный формат:

ПП	PD	Д1	Д2	Б1	С1	Б2	С2

Мнемокод-формат:

ДД С1 (Д1, Б1), С2 (Д2, Б2).

Прерывания программы:

операция (если отсутствуют средства обработки десятичных данных);

защита памяти;

адресация;

спецификация;

данные;

десятичное деление.

Если адрес какого-либо байта при выполнении команды "Деление

десятичное" соответствует несуществующим ячейкам главной памяти, возникает прерывание программы по причине "Неправильная адресация".

Если ненулевой программный ключ не совпал с ненулевым ключом страницы главной памяти, содержащей первый операнд, возникает прерывание программы по причине "Нарушение защиты памяти".

Если код длины делителя равен или больше кода длины делимого или больше семи, возникает прерывание программы по причине "Нарушение спецификации".

Если результат деления, частное и остаток, не помещается в поле первого операнда или делитель равен нулю, возникает прерывание по причине "Недопустимое деление десятичное". При этом делимое и делитель остаются в главной памяти без изменения.

Если при выполнении команды "Деление десятичное" были обнаружены недопустимые коды, имеет место прерывание программы по причине "Неправильные данные".

Примечание. Максимальная длина делимого - 31 цифра и знак. Так как минимальная длина остатка - одна цифра и знак, максимальная длина частного - 29 цифр и знак.

Некорректность деления можно заранее обнаружить пробным вычитанием. Крайний левый разряд поля делителя помещается на одну позицию правее крайнего левого разряда поля делимого. Если выравненный таким образом делитель меньше или равен делимому, то это указывает на некорректность деления.

Некорректность десятичного деления возникает, если делимое не имеет по крайней мере одного нуля слева.

Пример:

```
PD    70   5  000  5   101   ДД  0 (8,5),X'101'(1,5)
```

До выполнения команды:

```
регистр № 5              00    00    10    00
```

адреса	001000 ÷ 001003	01	00	00	00
	001004 ÷ 001007	00	00	00	0A
	001100 ÷ 001103	56	7B	12	34
После выполнения команды:					
	регистр № 5	00	00	10т	00
адреса	001000 ÷ 001003	14	28	57	14
	001004 ÷ 001007	28	57	1B	3A
	001100 ÷ 001103	56	7B	12	34

5.5.12 ПЕРЕВОД В УПЛОТНЕННЫЙ ФОРМАТ. По команде "Перевод в уплотненный формат" (ПУФ) выполняется следующая операция: формат второго операнда переводится из зонированного в уплотненный и результат помещается на место первого операнда. Обработка ведется справа налево. В младшем байте второго операнда меняются местами полубайты (знак и младшая десятичная цифра) и результат записывается в младший байт поля первого операнда. Затем берутся два следующих байта второго операнда, зоны удаляются, а байт, образованный из оставшихся десятичных цифр, помещается во второй справа байт поля первого операнда и т.д. Если второй операнд будет исчерпан до заполнения всего поля первого операнда, оставшиеся разряды заполняются нулями. Если же поле первого операнда будет заполнено до исчерпания второго операнда, оставшиеся байты второго операнда игнорируются.

Знаки, цифры и удаляемые зоны второго операнда не проверяются на допустимость кодов.

При команде "Перевод в уплотненный формат" поля операндов могут произвольно перекрываться.

При выполнении команды "Перевод в уплотненный формат" результирующий код условия не изменяется.

Машинный формат:

ПП	Р2	Д1	Д2	Б1	С1	Б2	С2

Мнемокод-формат:

ПУФ С1 (Д1; Б1), С2 (Д2;Б2)

Прерывания программы:

защита памяти;

адресация;

Если адрес какого-либо байта при выполнении команды "Перевод в уплотненный формат" соответствует несуществующим ячейкам главной памяти, возникает прерывание программы по причине "Неправильная адресация".

Если ненулевой программный ключ не совпал с ненулевым ключом страницы главной памяти, содержащей первый операнд, возникает прерывание программы по причине "Нарушение защиты памяти".

Пример:

Р2 ПП Р 20D Р 2ID ПУФ X'20D'(I6,I5),X'2ID'(I6,I5)

До выполнения команды:

регистр № Р	00	00	00	60
адреса 00026С÷00026Р	3А	00	5I	82
000270÷00027З	I5	24	89	47
000274÷000277	0I	02	00	I2
000278÷00027В	4З	52	58	47
00027С÷00027Р	4В	5I	5I	58
000280÷000283	54	55	56	57
000284÷000287	58	59	50	5I
000288÷00028В	52	53	54	55
00028С÷00028Р	А6	II	23	42

После выполнения команды:

регистр № Р	00	00	00	60
адреса 00026С ÷ 00026Р	3А	00	00	00
000270 ÷ 000273	00	00	00	00
000274 ÷ 00027Р	0I	I3	45	67
000278 ÷ 00027В	89	0I	23	45
00027С ÷ 00027Р	6А	5I	5I	53
000280 ÷ 000283	54	55	56	57
000284 ÷ 000287	58	59	50	5I
000288 ÷ 00028В	52	53	54	55
00028С ÷ 00028Р	А6	II	23	42

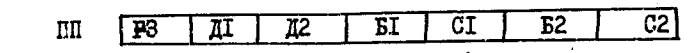
5.5.13 ПЕРЕВОД В ЗОНИРОВАННЫЙ ФОРМАТ. По команде "Перевод в зонированный формат" (ПЗФ) выполняется следующая операция: второй операнд переводится из уплотненного в зонированный формат, и результат помещается на место первого операнда. Обработка ведется справа налево. В младшем байте второго операнда меняются местами полубайты (младшая десятичная цифра и знак), и результат записывается в младший байт поля первого операнда. После обработки младшего байта извлекается следующий байт второго операнда, вставляются зоны и полученные два байта запоминаются в поле первого операнда, затем извлекается третий байт второго операнда, преобразуется в четвертый и пятый, байты результата, которые запоминаются и т.д.

Если второй операнд будет исчерпан до заполнения всего поля первого операнда, он дополняется нулями. Если же поле первого операнда будет заполнено до исчерпания второго операнда, оставшаяся часть второго операнда игнорируется.

При команде "Перевод в зонированный формат" поля операндов могут произвольно перекрываться.

При выполнении команды "Перевод в зонированный формат" результирующий код условия не изменяется.

Машинный формат:



Мнемокод-формат:

ПЗФ С1 (Д1,Б1), С2 (Д2,Б2).

Прерывания программы:

адресация;

защита памяти.

Если адрес какого-либо байта при выполнении команды "Перевод в зонированный формат" соответствует несуществующим ячейкам главной памяти, возникает прерывание программы по причине "Неправильная адресация".

Если ненулевой программный ключ не совпал с ненулевым ключом страницы главной памяти, содержащей первый операнд, возникает прерывание по причине "Нарушение защиты памяти".

Пример:

```
РЗ     78     Р   3I0    Р   3I8  ПЗФ X'3I0'(8,I5),
                                     X'3I8'(4,I5)

До выполнения команды:
регистр № Р              00   00   00   20
адреса 000330 ÷ 000333 4I   32   56   78
       000334 ÷ 000337 92   8I   30   4A
       000338 ÷ 00033B I2   34   78   2A

После выполнения команды:
регистр № Р              00   00   00   20
адреса: 000330 ÷ 000333  50   5I   52   53
        000334 ÷ 000337  54   57   58   A2
        000338 ÷ 00033B  I2   34   78   2A
```

3.5.14 ПЕРЕМЕЩЕНИЕ СО СДВИГОМ. По команде "Перемещение со сдвигом" (ПСД) выполняется следующая операция: четыре младших разряда первого операнда присоединяются справа ко второму операнду, и результат помещается на место первого операнда.

Допускается произвольное перекрытие полей первого и второго операнда, поэтому устанавливается следующий порядок выполнения операции.

Извлекаются самые младшие байты первого и второго операнда. К младшему полубайту второго операнда приформировывается справа младший полубайт первого операнда и результат записывается вместо младшего байта первого операнда. Затем извлекается второй справа байт второго операнда, и к младшей его половине приформировывается справа оставшаяся старшая часть младшего байта второго операнда. Результат записывается на второе справа место в поле первого операнда. Далее извлекается третий байт результата и т.д. Если второй операнд будет исчерпан до заполнения всего первого операнда, оставшаяся часть поля результата заполняется нулями. Если после заполнения всего поля первого операнда остается неиспользованная информация из второго операнда (нечетное число полубайтов), эта информация теряется (игнорируется). Ни первый, ни второй операнд не проверяются на допустимость кодов.

При выполнении команды "Перемещение со сдвигом" результирующий код условия не изменяется.

Машинный формат:

ПП	Р1	Д1	Д2	Б1	С1	Б2	С2

Мнемокод-формат:

ПСД С1 (Д1, Б1), С2 (Д2, Б2)

Прерывания программы:

защита памяти;

адресация.

Если адрес какого-либо байта при выполнении команды "Перемещение со сдвигом" соответствует несуществующим ячейкам главной памяти, возникает прерывание программы по причине "Неправильная адресация".

Если ненулевой программный ключ не совпал с нулевым ключом страницы главной памяти, содержащей первый операнд, возникает прерывание программы по причине "Нарушение защиты памяти".

Примечание. Среди команд десятичной арифметики нет команд сдвига, так как эквивалент сдвига может быть получен программно. Используя команду перемещения со сдвигом и команды логических пересылок, можно написать программы для сдвига вправо и влево на четное и нечетное число позиций.

Пример:

```
PI    7B  A  200  A  200  ПСД X'200'(8,I0),X'200'(4,I0)
```

До выполнения команды:

регистр № A		00	00	00	05
адреса	000204 ÷ 000207	IA	2I	22	34
	000208 ÷ 00020B	56	00	00	00
	00020C ÷ 00020F	7B	32	I4	5I

После выполнения команды:

регистр № A		00	00	00	05
адреса	000204 ÷ 000207	IA	00	00	00
	000208 ÷ 00020B	02	I2	28	45
	00020C ÷ 00020F	6B	32	I4	5I

9.5.15 ОСОБЫЕ СЛУЧАИ ПРИ ВЫПОЛНЕНИИ КОМАНД ДЕСЯТИЧНОЙ АРИФМЕТИКИ. Неправильные команды, данные или результаты вызывают прерывание программы. Когда происходит прерывание, текущее ССП сохраняется как старое ССП и выбирается новое ССП. Код прерывания в старом ССП указывает на причину прерывания. Для операций над десятичными числами возможны следующие прерывания программы.

Операция. Задана команда "Сложение десятичное", "Вычитание десятичное", "Очистка со сложением десятичным", "Сравнение десятичное", "Умножение десятичное" или "Деление десятичное", а средства обработки десятичных данных отсутствуют. Команда не выполняется. Поэтому код условия и данные в регистрах и памяти не меняются.

Защита памяти. Ключ памяти для ячейки, в которую должен быть записан результат, не совпадает с ключом защиты в ССП.

Адресация. Адрес указывает на ячейку, которая отсутствует в памяти данной установки.

В двух последних случаях выполнение операции прекращается. Значения результата и код условия заранее предсказать нельзя, и поэтому они не должны использоваться для дальнейших вычислений.

Ограничения, касающиеся адреса в целом, не распространяются на компоненты, из которых он получается, а именно, на содержимое полей С1 и С2 и содержимое регистров, определяемых Б1 и Б2.

Спецификация. Длина множителя или делителя больше, чем 15 цифр плюс знак или больше, чем длина множимого или делимого. Команда не выполняется, поэтому код условия и данные в памяти и регистрах остаются без изменения.

Данные. Знак или цифры операнда в командах "Сложение десятичное", "Вычитание десятичное", "Очистка со сложением десятичным", "Сравнение десятичное", "Умножение десятичное" и "Деление десятичное" имеет неправильную кодировку. В старших разрядах

множимого недостаточное количество нулей. Поля операндов в перечисленных операциях неправильно перекрываются. Операция прекращается. Значение результата и кода условия заранее предсказать нельзя, поэтому они не должны использоваться в дальнейших вычислениях.

Десятичное переполнение. Переполнение результата в командах "Сложение десятичное", "Вычитание десятичное" или "Очистка со сложением десятичным". Прерывание происходит только тогда, когда разряд маски десятичного переполнения равен единице. Операция завершается тем, что младшие разряды результата помещаются в поле результата и коду условия присваивается значение 3; старшие разряды результата теряются. Знак и младшие цифры в поле результата такие же, какие они были бы при бесконечно длинном поле результата.

Десятичное деление. Частное не помещается в отведенное для него поле; к этому случаю относится деление на нуль. Деление не выполняется. Поэтому делимое и делитель в памяти не меняются.

Транслятор. Отладчик. Трансляторные команды

6. РАБОТА С ОБСЛУЖИВАЮЩИМИ ПРОГРАММАМИ

6.I. Транслятор с МНЕМОКОДа

6.I.I. Введение. Транслятор с МНЕМОКОДа предназначен для обработки программ, написанных на МНЕМОКОДе. МНЕМОКОД обеспечивает составление программ на уровне машинных команд с использованием операторов трансляторных команд, вызывающих выполнение определенных операций во время трансляции, и системных макрокоманд - специальных операторов обращения к супервизору и логическому уровню ввода-вывода.

МНЕМОКОД предусматривает задание констант и резервирование памяти для них, указание адресов памяти и регистров в явном и символическом виде (с автоматическим распределением памяти транслятором), задание информации, необходимой для использования транслятором базовой адресации, а также для связи раздельно транслированных программ.

При написании исходных программ допустимо десятичное, шестнадцатеричное или символьное представление данных.

6.I.2. Основные функции транслятора. Обработка программ включает преобразование последовательности исходных операторов в программу на внутреннем языке машины, присвоение адресов командам и другим элементам программы, выполнение вспомогательных функций, указанных программистом.

Выходом транслятора является рабочий модуль, созданный в форме, отвечающей требованиям программы "компоновщик". Форма рабочего модуля позволяет перемещать программы из первоначально указанной области памяти в любую другую подходящую область.

Для каждой транслируемой программы транслятор составляет печатный документ. Программист может частично управлять содержанием печатного документа.

Во время трансляции исходная программа анализируется на наличие ошибок при использовании МНЕМОКОДа. Обнаруженные ошибки указываются в печатном документе.

6.I.3. Средства ввода-вывода. Транслятор с МНЕМОКОДа может использовать все средства ввода-вывода АСВТ.

Для ввода операторов исходной программы может быть использовано одно из следующих внешних устройств: УВвК, УВвЛ, НМЛ. Для промежуточной работы транслятора, НМЛ или же УВЛ и УВвЛ. Выход транслятора может быть помещен на магнитной ленте, либо перфоленте. Печатный документ о программе выдается на УАЦП или НМЛ.

6.I.4. Связь с операционной системой. Транслятор с МНЕМОКОДа относится к обрабатывающим программам операционной системы и функционирует под ее контролем, Операционная система обеспечивает транслятор вводом-выводом. Транслятор работает на логическом уровне ввода-вывода, поэтому не зависит от конкретных внешних устройств. Устройства ввода-вывода определяются в управляющих операторах ОД в задании на трансляцию. Транслятор помещен в системную библиотеку.

6.I.5. Структура транслятора. Программа состоит из 3-х частей: общая часть, I-я прогонка, 2-я прогонка. Каждая из этих частей в системной библиотеке хранится как отдельная программа с названиями соответственно: ИЕУМКД, ИЕУМКДI, ИЕУМКД2 (для трансляции операторов с русской мнемоникой) и ИЕУКМДЛ, ИЕУМКДЛI, ИЕУМКД2 (для трансляции оператора с латинской мнемоникой).

Транслятор работает в два этапа: I-я прогонка и 2-я прогонка. Общая часть находится в памяти во время работы обеих прогонок. Программы I-й и 2-й прогонок работают последовательно, занимая одну и ту же область, т.е. по окончании I-й прогонки ее программа заменяется программой 2-й прогонки.

Во время первой прогонки составляется таблица идентификаторов, осуществляется синтаксический контроль операторов МНЕМОКОДа. Макрокоманды, имеющиеся в программе, заменяются соответствующими макрорасширениями, и подготовленная таким образом программа записывается на внешний носитель (перфоленту или магнитную ленту) и, если требуется , печатается на бланке УАЦП. Классификация ошибок, отпечатанных транслятором, приведена в приложении 3.

Во второй прогонке составляется рабочая программа. Входной информацией для 2-й прогонки является программа, записанная на внешний носитель во время I-й прогонки.

Результаты 2-й прогонки выводятся на внешний носитель (перфолента или магнитная лента) и на УАЦП.

6.I.6. Коды завершения и сообщения оператору. По окончании работы каждого этапа трансляции транслятор выдает код условия, характеризующий правильность выполнения программы:

0 - ошибки в программе не обнаружены;

4 - обнаружены несущественные ошибки, позволяющие выполнить второй этап трансляции;

I6 - обнаружены существенные ошибки, трансляция прекращается

Если код I6 выдается после I-й прогонки, то 2-я прогонка не выполняется. Если код I6 выдается после 2-й прогонки , то не выполняется следующий шаг задания, использующий данную программу. Если во время трансляции обнаружена ошибка ввода-вывода, трансляция прекращается. Если в исходной программе нет оператора КОНЕЦ, то выдается диагностическое сообщение: "ОТСУТСТВУЕТ ОПЕРАТОР КОНЕЦ" в печатный документ и "ПРЕКРАЩАЕТСЯ ТРАНСЛЯЦИЯ ПРОГРАММЫ". При обнаружении несущественных ошибок в печатный документ заносится сообщение: "ОБНАРУЖЕННЫЕ ОШИБКИ ПОЗВОЛЯЮТ ПРОДОЛЖИТЬ ТРАНСЛЯЦИЮ". Если результат I-й прогонки выведен на

перфоленту, то в конце I-й прогонки на пульт оператора выдается сообщение: "ИЕУООIД ПОДГОТОВИТЬ ПЕРФОЛЕНТУ ДЛЯ ПРОДОЛЖЕНИЯ ТРАНСЛЯЦИИ".

6.I.7. Параметры транслятора. В поле ПАРМ оператора ВПЛ могут быть заданы следующие параметры: ПЧ, ТИ, ПЛ, ВИ (в любой последовательности).

ПЧ - указывается тогда, когда требуется печатать исходную программу на первом этапе трансляции. Если ПЧ не указан, то будут печататься только ошибочные операторы в I-й прогонке. Необязательный параметр.

ТИ - указывается тогда, когда требуется печатать таблицу идентификаторов. Необязательный параметр.

ПЛ - указывается тогда, когда вывод программы (I-й прогонка) производится на перфоленту. Необязательный параметр.

ВИ - указывается тогда, когда требуется вывести таблицу идентификаторов на устройство, на которое выводится результат трансляции. Необязательный параметр.

Результаты 2-й прогонки всегда выдаются на печать. Печатный документ 2-й прогонки содержит по каждому оператору следующую информацию (сдева направо):

- коды ошибок (приложение 3) указываются буквами русского и латинского алфавита;

- текущее значение счетчика адреса (заносится значение счетчика адреса после настройки его на соответствующую границу);

- машинный код, т.е. машинная команда или константа в шестнадцатеричной форме.

Машинные команды печатаются с пробелами, отделяющими код операции, поля Р1 и Р2, Б1 и С1, Б2 и С2. В случае ошибочного

оператора машинной команды в рабочую программу и, соответственно, в печатный документ заносится только код операции данной команды, остальные поля - нулевые.

Максимальная длина константы в строке печати 8 байтов, поэтому константы с большей длиной печатаются по частям.

Символ "+" означает, что данный оператор создан в результате обработки макрокоманды.

Оператор исходной программы.

Порядковый номер оператора, присвоенный транслятором.

Печатный документ разбивается на страницы по 64 строки в каждой.

Программист, однако, может изменять величину страниц, используя трансляторные команды ПРПСК и НСТР.

6.I.8. Задание на трансляцию. Для выполнения трансляции необходимы следующие файлы данных:

- для ввода исходной программы на I-ю прогонку (СИСВВОД);
- для вывода результатов I-й прогонки (СИСРУСТI);
- для ввода результатов I-й прогонки для проведения 2-й прогонки (СИСРУСТ2);
- для вывода окончательных результатов трансляции (СИСВЫВДМ);
- для выдачи печатного документа (СИСПЕЧТЬ);
- для вывода таблицы идентификатора на перфоленту или магнитную ленту (СИСВЫВТИ).

Наименование операторов СД для этих файлов (определено заранее) соответственно СИСВВОД, СИСРУСТI, СИСРУСТ2, СИСВЫВДМ, СИСПЕЧТЬ, СИСВЫВТИ.

Существует несколько режимов трансляции в зависимости от состояния программы, наличия внешних устройств, порядка даль-

нейшего использования результатов трансляции и т.д. Для основных режимов ниже приводится набор управляющих карт, которые обеспечивают нужные действия.

6.I.9. Примеры набора управляющих операторов для основных режимов:

I. Трансляция с печатью исходной программы (ПЧ) и таблицы идентификаторов (ТИ). Вывод результатов I-й прогонки на МЛ и результаты 2-й прогонки на ПЛ.

```
//ТЕСТ      ЗДН  УРС=I
//ШАГI      ВПЛ  ПРГ=ИЕУМКД, ПАРМ='ПЧ, ТИ'
//СИСРУСТI  ОД   УСТР=OI2, ТОМ=(ПРНТ=ТОМI), МЕТКА=(,НМ)
//СИСРУСТ2  ОД   ТОМ=ОБЩТ=СИСРУСТI, МЕТКА=(,НМ)
//СИСВЫВДМ  ОД   УСТР=ООВ, МЕТКА(,НМ)
//СИСПЕЧТЬ  ОД   СВЫВОД=А
// СИСВВОД  ОД   *
```

Устройства: ПП, УАЦП, УВвК, ЛПМ (2 шт.), УВД.

2. Трансляция без печати исходной программы, но с печатью таблицы идентификаторов и выводом этой таблицы идентификаторов на магнитную ленту. Результаты I-й и 2-й прогонки записываются на магнитную ленту.

```
//ТЕСТ      ЗДН   УРС=I
//ШАГI      ВПЛ ПРГ=ИЕУМКД, ПАРМ='ТИ,ВИ'
            УСЛ =(4,РВ), УСЛ=(I6,РВ)
//СИСРУСТI  ОД  УСТР=OI2, ТОМ=(,ПРНТ=ТОМI),МЕТКА=(,НМ)
//СИСРУСТ2  ОД  ТОМ=ОБЩТ=СИСРУСТI, МЕТКА=(,НМ)
//СИСПЕЧТЬ  ОД  СВЫВОД=А
//СИСВЫВТИ  ОД  ТОМ=ОБЩТ=СИСВЫВДМ, МЕТКА(I,СМ)
//СИСВЫВДМ  ОД  УСТР=О13,ТОМ=(,ПРНТ=ТОМ2),МЕТКА=(2,СМ)
//СИСВВОД   ОД  *
```

Устройства: ПП, УАЦП, УВвК, ЛПМ (3 шт.)

3. Трансляция без печати исходной программы и таблицы идентификаторов. Результаты I-й прогонки выдаются на перфоленту, результаты 2-й прогонки - на магнитную ленту.

```
//ТЕСТ        ЗДН   УРС=I
//МАГI        ВПЛ   ПРГ=МКУМКД, ПАРМ='ПД'
//СИСРУСТI    ОД    УСТР=00В, МЕТКА=(,НМ)
//СИСРУСТ2    ОД    УСТР=00А, МЕТКА=(,НМ)
//СИСВЫВДМ    ОД    УСТР=0I3,ТОМ=(,ПРНТ=ТОМ3),МЕТКА=(,НМ)
//СИСПЕЧТЬ    ОД    СВЫВОД=А
//СИСВВОД     ОД    *
```

Устройства: ПШ, УАЦП, УВвК, НПМ (2 шт.), УВИ, УВвИ.

КЛАССИФИКАЦИЯ ОШИБОК

Код ошибки	Определение
А	Отсутствие поля названия
Б	Неправильное задание поля названия, поля операции или идентификатора в поле операнда
В	Идентификатор ранее определен
Е	Несуществующая мнемоника
Ж	Отсутствие поля операнда
И	Идентификатор в поле операнда не определен
К	Нарушение уровня скобок (> 5)
Л	Количество термов в выражении > I6
М	Неправильный разделитель
Н	Злоупотребление полем названия
О	Неправильное задание самоопределенного терма
П	Неправильное использование знаков арифметической операции
Р	Значения выражения > $(2^{3I}-I)$
С	Несовместимые признаки
Т	Общая ошибка в выражении
У	Неправильное вхождение СТАРТ
Ф	Недействительная карта продолжения
Ц	Переполнение счетчика адреса (> I6777215)
Ш	Количество внешних идентификаторов > 255, количество входных точек > I00
Э	Неправильное следование операндов макрокоманды
Щ	Несуществующий регистр

Код ошибки	Определение
Ч	Ошибка в операторе ИЗМЕН
Ю	Неправильное задание поля операнда
Я	Неправильное использование коэффициента кратности
D	Несуществующий тип константы
F	Неправильное задание модификатора длины
G	Неправильное задание порядка константы
I	Неправильный предел константы
J	Поле названия и поле операции не на одной строке
L	Переполнение таблицы идентификаторов
N	Нет подходящего базового регистра
Q	Нулевой регистр загружен ненулевым значением
V	Неправильное задание смещения
W	Отметка привилегированной операции
Д	Отсутствие выборочного параметра в таблице кодов
З	Неправильное задание параметр-идентификатора
Й	Нечетное количество амперсендов ( ) в операнде макрокоманды
Ы	Неправильное использование знака равенства (=)
S	Параметр (- идентификатор/выборочный) имеет более 7/4 символов
U	Отсутствие параметра в таблице соответствия
Z	Недействительная карта продолжения в сгенерированном операторе
R	Перемещаемое выражение
Г	Отсутствие поля операции
Х	Недопустимый символ

6.2. Отладчик

6.2.1. Описание программы "отладчик".

Назначение. Программа "отладчик" предназначена для проверки правильности составления и выполнения отлаживаемых программ, написанных на МНЕМОКОДе. Проверяемая (отлаживаемая) программа может быть представлена в К-формате или L -формате. При этом не требуется вмешательство оператора, так как операторы отладки, подготовленные программистом вводятся автоматически с устройства ввода с перфокарт. При проверке не делается специальных изменений в отлаживаемой программе ни программистом, ни программой "отладчик".

Функциональные возможности программы "отладчик". Программа "отладчик" позволяет выдавать на печать содержимое общих регистров, регистров с плавающей запятой, участков памяти в символьной и (или) шестнадцатеричной форме, системных таблиц, информацию о прослеживаемых командах после их выполнения. Кроме того программа "отладчик" позволяет осуществлять: динамическую проверку отлаживаемой программы в зависимости от результатов, получаемых при работе отлаживаемой программы, выполнение необходимых проверочных действий в зависимости от условий, полученных в результате проверки.

Программа "отладчик" разрешает программисту вызывать данные в отлаживаемой программе для ее проверки.

Проверочные (контрольные) точки программист может вводить в тех местах, где он желает проверить правильность получения результатов при выполнении отлаживаемой программы или правильность алгоритма. Данные и информация для проверки могут задаваться на уровне языка МНЕМОКОД и на уровне относительных

(физических) адресов. При этом результаты выдаются на печать.

Требования к конфигурации АСВТ. Для работы программы "отладчик" необходима следующая конфигурация оборудования:

- объем оперативной памяти не менее I5K для программы "отладчик";
- системные устройства ввода-вывода (устройство ввода с карт, устройство печати, пульт программиста);
- лентопротяжный механизм (ЛПМ) с системной библиотекой,
- ЛПМ с операционной системой (ОС);
- устройства ввода-вывода, которые требуются для работы отлаживаемой программы.

Размещение и вызов программы "отладчик". Программа "отладчи является системной программой ленточной операционной системы (ОС-А/АСВТ) и вызывается во входном потоке в качестве шага задания. Программа состоит из двух частей, каждая из которых помещена в системной библиотеке и идентифицирована соответственно именем: ИЕГОТЛАД - первая часть.

ИЕГОТЛД2 - вторая часть.

Входные и выходные данные программы "отладчик".

Входные данные:

- задание для отладки, представляющее собой последовательность оператора отладки и данных для отлаживаемой программы (если необходимо);
- отлаживаемая программа.

Выходные данные:

- сообщения об операторах отладки. Операторы отладки проверяются на правильность и ошибки регистрируются на устройстве печати;
- сообщения о результатах выполнения отлаживаемой про-

граммы в режиме отладки (выдаются на печать).

Связь программы "отладчик" с ОС и потребительской программой. Программа "отладчик" работает под управлением управляющей программы и при своей работе требует память переменной длины и не может работать в мультипрограммном режиме.

Программа "отладчик" получает управление от отлаживаемой программы, когда имеется "ВСУ отладка". Точка входа в программу "отладчик" есть анализатор прерывания.

После выполнения всех действий программа "отладчик" передает управление отлаживаемой программе.

Краткая характеристика программы "отладчик". Программа состоит из двух частей, помещенных в библиотеку под именами ИЕГОТЛАД и ИЕГОТДЛ2 соответственно. Программы работают последовательно, занимая одну и ту же область, т.е. по окончании работы первой части в память, занятую ею, загружается вторая часть.

Первая часть программы "отладчик" состоит из подпрограмм:

- расшифровка операторов отладки и составление таблиц, необходимых при работе второй части;
- выдача диагностических сообщений об ошибках, обнаруженных при расшифровке операторов отладки.

Подпрограмма расшифровки выполняет следующие функции:

- проверяет оператор отладки на правильность. При написании оператора потребитель должен строго придерживаться общих правил, перечисленных в п.6.2.2., в противном случае будет выдано диагностическое сообщение об ошибке и оператор аннулирован из задания отладки.

Подпрограмма по данному оператору составляет таблицу оператора отладки, информация которой используется при работе второй части программы "отладчик".

Вторая часть программы "отладчик" состоит из подпрограмм:

- ввод отлаживаемой программы;
- внесение изменений в отлаживаемую программу указанных в операторах ВСТАВ;
- расстановка "ВСУ отладка" в контрольных точках отлаживаемой программы, указанных программистом;
- защита программы "отладчик" и памяти, занятой ОС, от действий отлаживаемой программы;
- анализатор прерываний. Осуществляет анализ типа прерывания, передает управление на соответствующую подпрограмму реакции на прерывание;
- выполнение вытесненной команды;
- реакция оператора СНИМК;
- реакция оператора СЛЕДП;
- реакция оператора ВСТАВ;
- реакция оператора ПРИСВ;
- реакция на программные прерывания;
- печать "отладчика".
- реакция на остальные нарушения.

Подпрограмма печать "отладчика" осуществляет выдачу на печать содержимого памяти, регистров, результатов прослеживания и состоит из частей:

- печать при работе оператора СНИМК;
- печать при работе оператора СЛЕДП;
- печать при работе оператора ПРИСВ;
- печать при программных прерываниях;
- печать операторов отладки при расшифровке.

Требования к контрольной точке. Команда, которая указывается в качестве контрольной точки не должна быть:

а) привилегированной командой;

б) командой, которая изменяется любой другой командой в отлаживаемой программе. Если это правило нарушается, то результаты могут быть неожиданными;

в) командой "Вызов супервайзера" (ВСУ);

г) командой "Подмена" (ПОД), которая вызывает выполнение команды, нарушающей правила а), б), в);

д) командой, которая выполняется командой "Подмена" (ПОД).

Примечания. I. Допускается задание операторов СНИМК, ПРИСВ внутри области прослеживания оператора СЛЕДП.

2. Не допускается задание более одного оператора на одной контрольной точке.

Если при вставке "ВСУ отладка" на место вытесненной команды, последняя является одним из типов команд, перечисленных выше, то вставка "ВСУ отладка" в заданной контрольной точке не происходит. Об этом выдается сообщение программисту.

Программа, находящаяся в режиме отладки, не является повторно используемой.

Требование по памяти. Программа "отладчик" требует при своей работе примерно I5К байтов. Чтобы определить требуемый минимум памяти при пользовании программой "отладчик", необходимо сложить объем памяти, занимаемый ОС, объем памяти, занимаемый отлаживаемой программой (вместе с данными и рабочими областями), объем памяти, занимаемый операторами отладки и объем памяти, занимаемый программой "отладчик". Объем памяти, занимаемый одним оператором отладки, равен 80 байтам.

6.2.2. Входной язык программы "отладчик".

Общие сведения. Связь между потребителем операционной системы и программой "отладчик" осуществляется с помощью шести операторов отладки:

- оператора СТАРТ;
- оператора СНИМК;
- оператора СЛЕДП;
- оператора ПРИСВ;
- оператора ВСТАВ;
- оператора КОНЕЦ.

Параметры, записываемые в этих операторах отладки позволяют программе "отладчик" осуществлять выгрузку содержимого регистров, участков главной памяти, системных таблиц, прослеживание за выполнением команд, а также динамическую проверку отлаживаемой программы в зависимости от результатов, получаемых при работе отлаживаемой программы; выполнение необходимых проверочных действий можно производить в зависимости от условий, случившихся в результате проверки. Когда происходит ошибка, то управление передается либо отлаживаемой программе, либо программе "отладчик".

Требуемые шаги работы при проверке отлаживаемой программы-трансляция, (редактирование, если необходимо) отладка.

Трансляция и редактирование (если необходимо) выполняются как шаги задания и должны быть сделаны перед отладкой программы.

Общие характеристики операторов отладки. С помощью оператора СТАРТ задают информацию об отлаживаемой программе, необходимую при работе программы "отладчик". Оператор СТАРТ в задании на отладку обязателен.

С помощью оператора СНИМК задают информацию об областях памяти, общих регистрах, регистрах с плавающей запятой, содержимое которых необходимо вывести на печать.

С помощью оператора СЛЕДП задают информацию тех участков отлаживаемой программы, в которых необходимо проверить выполнение команд. При работе программы "отладчик" в этом случае на печать выдается вся информация о выполняемых командах в указанной области или таблица передач управления "от" и "к" в указанной области программы.

С помощью оператора ВСТАВ задают информацию о замене старых данных в отлаживаемой программе на новые, об удалении части программы (т.е. не использование ее при добавлении новых участков ("заплат") в конце отлаживаемой программы.

С помощью оператора ПРИСВ задают информацию о тех действиях, которые необходимо выполнить при появлении ситуаций, указанных в этом операторе программистом.

В задании на отладку оператор КОНЕЦ обязателен и указывает на окончание операторов отладки.

Структура операторов отладки. Операторы отладки состоят из 3-х полей (поле названия, поле операции, поле операнда). Поля отделяются друг от друга наличием хотя бы одного пробела.

Поле названия содержит идентификатор, создаваемый программистом для наименования оператора отладки. Идентификатор записывается, начиная с первой позиции. Если же в первой позиции стоит пробел, то это указывает на отсутствие поля названия.

Поле названия не обязательно.

Поле операции содержит один из кодов оператора отладки: СТАРТ, СНИМК, СЛЕДП, ПРИСВ, ВСТАВ, КОНЕЦ.

Поля операции и операнда обязательны и должны начинаться

на первой карте.

Поле операнда содержит параметры двух типов:

- позиционные, которые записываются первыми в поле операнда (отсутствие позиционного параметра не указывается);

- ключевые, которые характеризуются ключами, расположенными перед знаком равенства.

Ключевые параметры записываются произвольно в поле операнда по отношению к другим ключевым параметрам.

Ниже при описании форматов операторов отладки и их полей используются следующие обозначения. Большие буквы и символы должны быть записаны так, как показано в инструкции, за исключением фигурных скобок, которые никогда не записываются в реальном операторе. Элементы в круглых скобках позиционные (т.е. они характеризуются позицией в круглых скобках по отношению к другим элементам), отсутствие необязательного элемента в круглых скобках не указывается. Элементы, заключенные в фигурные скобки $\{\ldots\}$, представляют собой выборочные элементы, т.е. в реальном операторе программист указывает только один из указанных в этих скобках элементов.

Смысл элементов (общих для всех операторов отладки), заключенных в круглые скобки приведен ниже.

Ф = КККККК - задание относительного (физического) адреса после трансляции.

И = идентификатор $\pm$ ККК - задание адреса через идентификатор и смещение (К- шестнадцатеричные цифры), т.е. адрес есть сумма относительного адреса идентификатора и смещения. Смещение не обязательно, его отсутствие не указывается.

Примечание: При задании адресов в операторах отладки через идентификаторы необходимо, чтобы в

задании на трансляцию в управляющем операторе ВПЛ в поле ПАРМ был указан параметр ВИ (см.6.I.), что означает выдачу таблицы идентификаторов отдельным файлом на внешнее устройство, отличное от УАЦП и УВвК. Кроме того во входном потоке на трансляцию должен быть указан оператор СД (определить данные), содержащий все характеристики, описывающие таблицу идентификаторов, как файл и устройство, на которое выводится этот файл. Таблица идентификаторов отлаживаемой программы и сама отлаживаемая программа могут выдаваться на одно и то же устройство как отдельные файлы.

Б = №, №, ККК - задание адреса через индексный регистр, базовый и смещение, т.е. адрес определяется при выполнении отлаживаемой программы как сумма содержимого регистров индексного, базового и смещения.

Параметры должны разделяться запятыми и не могут содержать пробелов между разделяющими их запятыми. Разрыв параметров не допускается.

Если один или несколько параметров на одной карте не вмещаются, то после последнего параметра на данной карте ставится запятая, а в 72 позиции любой символ (кроме пробела) - признак продолжения. Продолжить писать следующий параметр на карте продолжения с I6 позиции.

Исключение составляют параметры ДОБЗ, ЗАМН, ГРУП. В параметрах ДОБВ, ЗАМН разрыв допускается только для непосредственной информации, которая дается в самом операторе и следует за знаком равенства в элементах Х = (шестнадцатеричная информа-

ция). С = (символьная информация).

В случае символьной информации в самом операторе допускаются любые символы, причем, если будет указан символ ")" - закрывающая скобка, он должен быть указан двумя символами "))". В противном случае информация будет ошибочной.

Ограничения на параметр ГРУП описываются при описании оператора отладки ПРИСВ.

Оператор СТАРТ фиксирует начало ввода операторов отладки, кроме того, задает информацию об отлаживаемой программе.

Формат оператора

Название	Операция	Операнд
	СТАРТ	ИДЕН, АНТР = ({Ф=}{И=})
		АТВХ = ({Ф=}{И=})
		ОДПГ = имя ОД
		ОДВИ = имя ОД

ИДЕН - позиционный параметр указывает на наличие адресов, заданных через идентификаторы.

АНТР - адрес начала трансляции отлаживаемой программы.

АТВХ - адрес точки входа в отлаживаемую программу.

Параметр ОДПГ содержит имя ОД - имя, которое должно быть идентично имени ОД оператора определения данных, указывающего устройства, с которого будет загружаться отлаживаемая программа.

Параметр ОДВИ содержит имя ОД, которое идентифицирует ОД характеризующее файл и устройство, на котором находится таблица идентификаторов.

Примеры. Имя ОД отлаживаемой программы - ОДПРОГ.

I. Начальный адрес трансляции отлаживаемой программы = 0 и совпадает с точкой входа в программу:

СТАРТ АНТР = (Ф=0), АТВХ = (Ф=0), ОДПГ = ОДПРОГ

2. Идентификатор ВХОДІ идентифицирует точку входа в отлаживаемую программу, начальный адрес трансляции программы = 0, а имя ОД таблицы идентификатора программы - ОДТАБЛИД

СТАРТ ждем, АТВХ = (И=ВХОДІ), АНТР=(Ф=0), ОДПГ=ОДПРОГ, ОДВИ=ОДТАБЛИД

3. Значение идентификатора ВХОДІ =2В0

СТАРТ АТХВ =(Ф=2В0), ОДПГ = ОДПРОГ, АНТР = (Ф=0).

Рабочие операторы отладки СНИМК, СЛЕДП, ПРИСВ имеют общие параметры (АДКТ, Д, М, Т), смысл которых следующий:

АДКТ - указывает адрес в отлаживаемой программе, которая используется программой "отладчик" как контрольная точка, после выполнения которой программа "отладчик" выдает на печать либо содержимое общих регистров (если указан параметр РОБЩ=ПЗ, то и содержимое регистров с плавающей запятой) и областей памяти, указанных в операторе СНИМК, либо результаты прослеживания областей, указанных в операторе СЛЕДП, либо передает управление самому оператору в случае оператора ПРИСВ.

Параметр АДКТ в операторах СНИМК, СЛЕДП, ПРИСВ обязателен.

Д - количество прохождений через контрольную точку, до первой выдачи на печать.

М - количество прохождений через контрольную точку между двумя последовательными выдачами на печать.

Т - общее количество выдачей на печать при прохождении через контрольную точку.

Если в операторе отсутствует один (или все) из параметров Д,М,Т, то от тствующему параметру будет присвоено значение I.

С помощью оператора СНИМК задают информацию об областях памяти, о наличии регистров с плавающей запятой, содержимое которых необходимо вывести на печать.

Формат оператора

Название	Операция	Операнд
	СНИМК	ж.АДКТ = (Ф=КККККК / И=идент±ККК) РОБЩ=ПЗ, Д=КККК,М=КККК, Т=КККК, ОБЛП= (Ф=КККККК / И=идентиф±ККК / Б=№,№,ККК), (Ф= / И= / Б=), (Х / С / СШ)

Первый параметр в поле операнда (ж) - позиционный, назначение этого параметра указано при описании оператора ПРИСВ.

При наличии оператора СНИМК программа "отладчик" всегда выдает на печать содержимое общих регистров.

Примечание. Если необходимо иметь только содержимое общих регистров, программист должен в поле операнда указать только АДКТ.

Параметр РОБЩ указывает на наличие регистров с плавающей запятой, содержимое которых будет выдано на печать. Данный параметр не обязателен.

Первый элемент параметра СБЛП указывает первый адрес в отлаживаемой программе, который является началом области памяти.

Второй элемент указывает адрес первого байта, на который не распространяется действие оператора СНИМК, т.е. адрес конца области памяти.

Третий элемент указывает, в каком виде нужно выдавать на

печать содержимое памяти, а именно:

Если X - в I6-ричном,

С - в символьном.

Первые два элемента обязательны в параметре ОБЛП, а третий может отсутствовать (отсутствие не указывается).

При отсутствии третьего элемента содержимое области на печать выдается в две строки: I строка - в символьном виде,

2 строка - в I6-ричном виде.

В операторе СНИМК допускается повторение параметра ОБЛП (не более 8).

Примеры.

I. Необходимо выдать в I6-ричном виде на печать содержимое таблицы, сформированной вне программы, адрес которой находится в регистре 3, длина таблицы = 80_{I6}, после выполнения команды, физический адрес которой Ф=24:

СНИМК АДКТ = (Ф=24), ОБЛП = (Б=0,3,000,Б=0,3,080,X)

2. После выполнения команды Ф=240 выдать на печать содержимое регистров с плавающей запятой:

СНИМК РОБЩ=ПЗ, АДКТ =(Ф=240)

3. После выполнения команды Ф=I0 выдать на печать только содержимое общих регистров:

СНИМК АДКТ = (Ф=I0)

4. Выдать на печать в символьном виде $I00_{I6}$ байтов, после выполнения команды Ф=40. Начальный адрес задан через идентификатор ТАБЛ5.

СНИМК АДКТ =(Ф=40), ОБЛП = (И=ТАБЛ5, И=ТАБЛ5+I00,С)

5. После выполнения команды Ф=2048 выдать на печать содержимое трех областей памяти:

I. область : начальный адрес 3048

конечный адрес 3070

И область: начальный адрес 4000

конечный адрес 4100

И область: начальный адрес 4600

конечный адрес 4С00

СНИМК АДКТ= (Ф=2048), ОБЛП =(Ф=3048, Ф=3070), И

ОБЛП = (Ф=4000, Ф=4100), ОБЛП (Ф=4600, Ф=4С00)

Оператор СЛЕДП указывает участки команд программы, которые нужно выполнить в одном из указанных режимов с выдачей информации о командах на УАШП.

Формат оператора

Название	Операция	Операнд
	СЛЕДП	И,АДКТ = (Ф= / И=) ,
		ОБЛП = ((Ф= / И= / Б=) , (Ф= / И= / Б=)),
		ТИПС = (ПРОК / ПУПР / ПЗВР) , Д=КККК, И=КККК, Т=КККК

Параметр ОБЛП оператора СЛЕДП содержит только два позиционных элемента.

Первый и второй элементы параметра ОБЛП указывают соответственно адрес начала и конца области (участка) прослеживания.

Повторение параметра ОБЛП в операторе СЛЕДП не допускается.

Примечание. В области прослеживания допускается контрольная точка только данного оператора СЛЕДП.

Параметр ТИПС указывает, какой тип прослеживания нужно применить к заданной области памяти, а именно:

ПРОК означает, что отладчик выдает на печать результат прослеживания каждой команды.

ПУПР – будет выполнено прослеживание только команд передач управления.

ПЗВР – прослеживание только команд передач с возвратом.

В операторе СЛЕДП обязательны параметры АДКТ, ОБЛП, ТИПС.

Примеры.

I. После выполнения команды Ф=I024 выдать на печать результат прослеживания области, начальный адрес которой Ф=I048, конечный адрес – Ф=I080).

СЛЕДП ТИПС = ПРОК, ОБЛП = (Ф=I048, Ф=I080), АДКТ = (Ф=I024)

2. После прохождения метки КОРЕКТ выдать результат прослеживания только команд передач управления, начиная с метки НАЧРАБ до метки КОНРАБ.

СЛЕДП АДКТ = (И=КОРЕКТ), ТИПС = ПУПР,
ОБЛП = (И=НАЧРАБ , И = КОНРАБ)

Оператор ВСТАВ.

Формат оператора

Название	Операция	Операнд
	ВСТАВ	УДАЛ = $\left(\begin{Bmatrix}Ф=\\И=\end{Bmatrix}, \begin{Bmatrix}Ф=\\И=\end{Bmatrix}\right)$,
		ЗАМН = $\left(\begin{Bmatrix}Ф=\\И=\end{Bmatrix}, \begin{Bmatrix}Ф=\\И=\\С=\\Х=\end{Bmatrix}\right)$,
		ДОБВ = $\left(\begin{Bmatrix}Ф=\\И=\end{Bmatrix}, \begin{Bmatrix}Х=\ldots\\С=\ldots\end{Bmatrix}\right)$,

Оператор ВСТАВ выполняет до работы отлаживаемой программы следующие функции.

– Удаление команд, – т.е., чтобы при работе отлаживаемой программы указанные команды не выполнялись. В этом случае в поле оператора БСТАВ должен быть указан один параметр УДАЛ,

первый элемент которого указывает адрес начала удаления команд, второй - адрес конца. Адрес конца удаления равен адресу первой команды, стоящей за последней командой удаления.

Длина удаляемой информации должна быть четной и не меньше 2-х байтов. Максимальная длина удаляемой информации-256 байтов.

- Замены - информация (заменяемая) одной области заменяется на информацию (вставляемую) другой области (вставляемая информация может быть указана непосредственно в параметре ЗАМН после X = . . . или C = . . .). В случае замены в поле операнда должны быть параметры УДАЛ и ЗАМН. Первый и второй элементы параметра УДАЛ указывают соответственно адрес начала и конца заменяемой информации (т.е. область памяти, куда надо поместить вставляемую информацию). Первый и второй элементы параметра ЗАМН указывают адрес начала и конца вставляемой информации (т.е. область памяти, откуда надо брать вставляемую информацию). Если вставляемая информация в самом параметре, тогда указывается один элемент $\begin{Bmatrix} X \\ C \end{Bmatrix}$.

Если выполняется замена команд, то необходимо, чтобы длины заменяемой и вставляемой информации были четными, и длина вставляемой информации меньше или равна длине заменяемой. Для замены констант достаточно, равенства длин информации.

- Вставки- информация, указанная непосредственно в параметре ДОБВ, вставляется в область памяти. Адрес области указывается в самом параметре ДОБВ. В случае вставки команд, в поле операнда должен быть параметр ДОБВ = $\left(\begin{Bmatrix} Ф= \\ И= \end{Bmatrix} ; \quad X=\ldots \atop C=\ldots\right)$

Длина непосредственной информации в параметрах ЗАМН и ДОБВ должна не превышать 256 байтов. Правильность информации в самом операторе не проверяется.

Примеры.

I. Требуется удалить информацию от метки СИНI до метки М2

ВСТАВ УДАЛ = (И=СИНI, И=М2).

2. Удалить по адресу Ф=40 2 байта

ВСТАВ УДАЛ = (Ф=40, Ф=42).

3. Требуется информацию, расположенную по адресу Ф=240 длиной 80_{I6} байтов, поместить по адресу Ф = 48.

ВСТАВ УДАЛ =(Ф=48, Ф=СВ),
ЗАМН = (Ф=240, Ф=2СО)

4. 000020 5950
2 В020
4 4780
.
.
. В840
ССDРОО
.
.
.

Пропущена команда засылки адреса

4I50 по адресу Ф=20
ВСОО

Свободная память начинается с адреса Ф=РОО

Два оператора ВСТАВ выполняют эту вставку:

ВСТАВ ДОБВ = (Ф=20, Х=47РОВРОО);

ВСТАВ ДОБВ = (Ф=РОО, Х=4I50ВС005950В020 47РОВ024)

5. По адресу, определяемому меткой ТАБЗНАК, поместить символьную информацию х+-,.))'

ВСТАВ ДОБВ = (И=ТАБЗНАК, С=х+-,.))')

Оператор ПРИСВ. Оператор ПРИСВ применяется для динамического внесения изменений в отлаживаемую программу. Изменения могут вноситься в зависимости от поставленного условия или безусловно.

Например, в определенном месте программы (по желанию программиста) можно сравнить полученный результат с ожидаемым и, в зависимости от результата сравнения, или продолжить вычисления дальше, заслав верный результат, или начать вычисления заново уже в режиме прослеживания для нахождения ошибки.

При помощи оператора ПРИСВ можно изменять содержимое общих регистров, регистров с плавающей запятой, ячеек: памяти в пределах, указанных после трансляции на печатном документе, правой части ССП. Кроме того можно управлять операторами СНИМК, СЛЕДП, вводя их в действие при выполнении какого-либо условия, указанного в операторе ПРИСВ. Для этого поле операнда операторов СЛЕДП или СНИМК необходимо начать с параметра звездочка (*).

Результат работы оператора ПРИСВ выдается на печать.

Формат оператора ПРИСВ. Общее описание

Поле названия	Поле операции	Поле операнда
	ПРИСВ	АДКТ = $\left(\left\{\begin{matrix}Ф=\\И=\end{matrix}\right\}\right)$, Д = КККК, М=КККК, Т=КККК, ГРУП = (УСЛВ = [условие 1] , [условие 2] , ПРСВ = [присвоение 1], [присвоение 2] , [присвоение 3], [присвоение 4])

Параметры АДКТ, Д, М, Т, ГРУП являются ключевыми.

Обязательными параметрами являются АДКТ и ГРУП. Параметров ГРУП может быть один или два.

В групповом параметре ГРУП после ключевого слова УСЛВ может быть от 0 до 2-х условий, а после ключевого слова ПРСВ может быть от 1 до 4 присвоений. В групповом параметре ГРУП условия должны располагаться первыми (т.е. перед присвоениями).

Примеры:

1. ГРУП = (УСЛВ = [условие 1] , [условие 2] ,
ПРСВ = [присвоение 1], [присвоение 2] ,
[присвоение 3] , [присвоение 4])

2. ГРУП = (УСЛВ = [условие 1], ПРСВ = [присвоение 1] ,
[присвоение 2]) ,

3. ГРУП = (ПРСВ = [присвоение 1]) ,

4. ГРУП = (УСЛВ = [условие 1] , [условие 2] , ПРСВ =
[присвоение 1])

Формат условий и присвоений. Условие (присвоение) состоит из левого и правого подпараметров и, если это необходимо, дополнительного подпараметра. Формат условий (присвоений) приведен в таблице 11.

Правило записи условий (присвоений) приведено ниже:

а) дополнительный подпараметр отсутствует:

[условие (присвоение)] = [{левый подпараметр}]
{разделитель} {правый подпараметр}]

б) имеется дополнительный подпараметр:

[условие (присвоение)] = [{левый подпараметр}]
{разделитель} {правый подпараметр} {запятая}
{дополнительный подпараметр}]

Разделителем левого и правого подпараметров в присвоении служит запятая (,) .

Разделителем левого и правого подпараметров в условии служат знаки операций отношения (=, ≠ , , , =, =).

Таблица II

Левый подпараметр	Разделитель	Правый подпараметр	Дополнительный подпараметр
РОБЩ N РПЛЗ N ЯЧФ= NNNNNN ЯЧБ= (N,N,NNN) КУ АДР ССП	= ≠ > < >= <= ,	РОБЩ N РПЛЗ N ЯЧФ =NNNNNN ЯЧБ= (N,N,NNN) КУ АДР ССП : X' NN...N' C'C...C' СЛЕДП СНИМК N Ф= NNNNNN Б= (N,N,NNN) NN Ф= NNNNNN NN Б= (N,N,NNN) ДЛ=N, {X'NN...N' / C'C...C'}	ДЛ=NN

Через обозначена I6-ричная цифра, через С - элемент символьной информации.

Описание подпараметров

РОБЩ N задает номер (N) общего регистра.

РПЛЗ N задает номер (N) регистра с плавающей запятой.

ЯЧФ =NNNNNN задает адрес ячейки (байта) памяти в отлаживаемой программе после трансляции. Группа NNNNNN может содержать от I до 6 I6-ричных цифр.

ЯЧБ = (N,N,NNN) – группа цифр в скобках задает соответственно номера индексного, базового регистров и смещения. Смещение может быть задано одной, двумя, тремя 16-ричными цифрами. Номер индексного и базового регистров, указывается одной 16-ричной цифрой.

КУ – указывает, что правым подпараметром (или левым подпараметром) есть значение кода условия.

АДР – указывает, что в качестве значения левого(или правого подпараметра) берется адресная часть ССП.

ССП – указывает, что в качестве значения левого (или правого подпараметров)берется правая половина ССП.

Двоеточие, стоящее после разделителя, указывает, что после двоеточия следует непосредственное значение, записанное как 16-ричная или символьная константа, а также как это описано ниже:

:X'NN ... N' – задает значение в виде 16-ричной информации.

:C'CC...C' – задает значение в виде символьной информации

: ДЛ= N, $\left\{\begin{matrix} X'NN \ldots N' \\ C'NN \ldots N' \end{matrix}\right.$ – задает 16-ричное или символьное значение с предварительным указанием кода длины значения. Значение длины на 1 меньше фактической длины. Такое задание подпараметра применяется

линь в случае, если левым подпараметром является ЯЧФ или ЯЧБ. Код длины изменяется от 0 до 7.

:N – задает значение кода условия для присвоения или сравнения с полученным кодом условия при выполнении программы.

:Ф = NNNNNN
Б=(N,N,N,NN) – задает значение адреса, используемое для присвоения или сравнения с адресной частью ССП при выполнении программы.

:NN Ф=NNNNNN
Б=(N,N,NNN) – задает значение адреса, используемое для присвоения правой части ССП или сравнения с правой частью ССП.

: СЛЕДП
: СНИМК – задает значение правого подпараметра в виде кода операции оператора отладки.

В этом случае левым подпараметром является ЯЧФ или ЯЧБ. Адрес, который указан в левом подпараметре, должен быть адресом контрольной точки для соответствующего оператора отладки, стоящего в общем потоке отладочных операторов.

ДЛ=NN – задает значение дополнительного подпараметра. Используется, когда и левым и правым подпараметром является ячейка памяти. Значение длины на I меньше фактического.

Правила переноса. В том случае, если поле операнда оператора ПРИСВ нельзя разместить на одной карте, можно воспользоваться картами продолжения, количество которых не ограничено. Карта, за которой следует карта продолжения, должна иметь в 72-й позиции любой символ, отличный от пробела.

Разрешается переносить на карту продолжения ту информацию, которая следует за запятой, стоящей между параметрами. В операторе ПРИСВ параметрами являются:

АДКТ = $\left(\left\{\begin{matrix}Ф=\\И=\end{matrix}\right\}\right)$, Д =*NNNN* , М=*NNNN* , Т =*NNNN*

ГРУП = (УСЛВ = [условие I] , [условие 2] ,
ПРСВ = [присвоение I] , [присвоение 2] ,
[присвоение 3], [присвоение 4])

Параметров ГРУП может быть в операторе ПРИСВ два.

Недопустимые пары левого и правого подпараметров приведены в таблице I2. При наличии недопустимых пар подпараметров в параметре ГРУП выдается сообщение об ошибке:

ОШ В ПОЛЕ СП-ДА

Таблица I2

Левый п/п \ Правый п/п	РОБЩ	РПЛЗ	ЯЧФ	ЯЧБ	КУ	АДР	ССП	:
РОБЩ	-	не доп						
РПЛЗ	не доп				не доп	не доп	не доп	
ЯЧФ								
ЯЧБ								
КУ		не доп			не доп	не доп	не доп	
ССП		не доп			не доп	не доп	не доп	
АДР		не доп			не доп	не доп	не доп	

Семантика. Условие заключает в себе следующее:

а) при наличии одного условия

если [{левый подпараметр} {код операции отношения} {правый подпараметр}] то

б) при наличии двух условий

если [{левый подпараметр} {код операции отношения} {правый подпараметр}] и

[{левый подпараметр} {код операции отношения}
{правый подпараметр} {запятая} {дополнительный
параметр}] то

Присвоение заключает в себе следующее:

[{левому подпараметру} присвоить значение
{правого подпараметра}]

Если условия отсутствуют, что присвоения имеют смысл безусловных.

Проверка условий в параметре ГРУП производится последовательно.

Присвоение в параметре ГРУП производится последовательно и независимо в том порядке, в каком информация записана в операторе ПРИСВ. Присвоение выполняется только в тех случаях, когда выполнены все условия, указанные в параметре ГРУП.

Печать результата работы оператора ПРИСВ. В начале работы оператора ПРИСВ на АЦПУ печатается строка:

печать отладчика ПРИСВ адрес контр точки =

Затем печатается сообщение, указывающее произведена ли операция присвоения. Если все условия выполнены, или, если параметр ГРУП не содержит условий, то строка печатается так:

ГРУП *N* УСЛВ выполнено ,

где означает порядковый номер параметра ГРУП в данном операторе.

Если не выполнено какое-либо условие в параметре ГРУП, то выдается следующее сообщение:

ГРУП *N* УСЛВ К не выполнено,

где *N* - порядковый номер параметра ГРУП в операторе ПРИСВ;

К - порядковый номер условия в параметре ГРУП, которое не выполнено.

Примеры:

ГРУП = (УСЛВ = [РОЩ5 ≠ : X'0'] , ПРСВ = [РОЩ5,:X'0'])

Данный пример читается так:

условие

если содержимое Р5 не равно "0", то

Р5 присвоить значение равное "0"

присвоение

ГРУП = (СПРСВ = [ЯЧФ = I00, : ДЛ=3, ОР X'47РОРI04'])

Данный пример читается так:

начиная с адреса I00 заслать значение

47РО РI04 длиной в 4 байта.

Длина указывается на I меньше истинной.

Такое присвоение может быть использовано, в частности, для изменения какой-либо команды на момент отладки.

Допустим, что требуется проверить участок программы 050 ÷ I00 на нескольких примерах. С помощью оператора ПРИСВ это можно сделать автономно, что значительно ускорит и повысит качество отладки.

04С - адрес контрольной точки оператора ПРИСВ;

050 - начало участка программы;

.

.

.

.

I00 - конец участка программы;

.

.

.

.

200 - стандартное поле памяти для исходных данных к данному участку программы (длина поля 8 байтов);

208 - стандартное поле, в которое помещается результат;

.

.

.

.

500 - поле исходных данных для примера I;

508 - поле результата для примера I;

50C - смещение следующего примера относительно адреса 500;

5IO - поле исходных данных для примера 2;

. поле исходных данных для последнего примера

. поле результата

. смещение следующего примера относительно адреса 500 равно нулю

В предположении, что у программиста есть свободный общий регистр (например, регистр 2), а базовым регистром является регистр I5, составляются следующие операторы.

```
ПРИСВ    АДКТ = (Ф=4С),  Т=I,
         ГРУП = (ПРСВ = [РСБЩ2,:Х'0'] ,
        [ЯЧФ  = 200, ЯЧФ=500, ДЛ=7] )
ПРИСВ     АДКТ = (Ф=I00),  Т=А ,
          ГРУП =(ПРИСВ = [ЯЧБ =(2,Р,508),ЯЧФ=208,ДЛ=3] ),
          ГРУП =(УСЛ   = [ЯЧБ =(2,Р,50С)≠:ДЛ=3,Х'0000000С'] ,
          ПРИСВ= [РСБЩ2, ЯЧБ =(2,Р,50С)] ,
         [ЯЧФ  = 200, ЯЧБ = (2,Р,500) ДЛ=7] ,
         [АДР,:Ф=50] )
```

Первый оператор очищает регистр 2 (который будет служить индексным регистром для переадресации примеров), заносит исходные данные для первого примера.

Второй оператор полученный результат перемещает в поле результата данного примера (позже можно при помощи оператора СНИМК отпечатать все полученные результаты), затем (если данный пример не является последним) заполняет индексный регистр 2 значением смещения для следующего примера, перемещает исходные данные следующего примера в стандартную ячейку, производит

передачу управления по адресу 050.

Оператор КОНЕЦ

Название	Операция	Операнд
	КОНЕЦ	

Поле операнда не используется.

Оператор КОНЕЦ отмечает окончание операторов отладки в задании на отладку.

Количество операторов отладки в задании на отладку определяется объемом свободной памяти.

Оформление задания на отладку. Программа "отладчик" находится в системной библиотеке. При работе программы "отладчик" используются следующие файлы:

- задание на отладку, представляющее собой последовательность операторов отладки;
- отлаживаемая программа;
- данные для работы отлаживаемой программы (если они необходимы);
- таблица идентификаторов (если имеются в операторах отладки адреса, заданные через идентификаторы).

Пример оформления задания на отладку.

```
1. //ОТЛАДКА      ЗДН   УРС=1
2. //ШАГ2         ВПЛ   ПРГ=ИЕГОТЛАД, ПАРМ='A=5'
3. //СИСВЫВОД     ОД     СВЫВОД=A
4. //ОДОТЛАЖП     ОД     УСТР=ОСА,МЕТКА=(,НМ),РЕЖПЛ=БК
5. //СИСВВ        ОД     УСТР=113,МЕТКА=(2,НМ),ТОМ=(ПРНТ=
                                                =АЛГ1)
6. //ДАН1         ОД     УСТР=112,МЕТКА=(,НМ),*
                         ТОМ=(ПРНТ=АЛГ2)
```

```
7. //РЕЗУЛЬТ    ОД  УСТР=ООВ,МЕТКА=(,НМ),РЕЖПЛ=БК
8. //СИСВВОД    ОД  *
9. //НАЧАЛО     СТАРТ ИДЕН,АНТР=(Ф=0),АТВХ=(И=МI),
                      ОДПГ=ОДОТЛАЖП
I0. ВСТАВКАI    ВСТАВ ДОБВ=(Ф=I5 8,Х=9I407СО)
II. ВСТАВКА2    ВСТАВ ДОБВ=(Ф=I23С. С=ВЕ)
I2              СНИМК АДКТ=(Ф=I0),Д=5,М=I0,Т=3,*
                      ОБЛП=(Б=0,     ,ОАС,Б=0,     , РАО)
I3.             СЛЕДП  АДКТ=(И=КI),ОБЛП=(И=КI, И=К5),
                       ТИПС=ПРОК, М=I0,Т=I00
I4.             ВСТАВ УДАЛ=(Ф=I00,Ф=I04)
I5.             ВСТАВ ЗАМН=(Ф=I50,Х=47РО    ОРА )
I6.             СНИМК АДКТ =(Ф=I000), *
                      ОБЛП=(Ф=I000,Ф=I050,Х),*
                      ОБЛП=(Ф=2Д50,Ф=2IР0,С)
I7.             ПРИСВ АДКТ=(Ф=АС),
                      ГРУП=(УСЛВ= [ЯЧБ =(2, D, 50С) ≠
                      :ДЛ=3,Х'ССССССССО'] ,
                      ПРСВ= [РОБЩ2,ЯЧБ=(2,Р,50С)],
                      ЯЧФ=200, ЯЧБ=(2, D, 50С),
                      ДЛ=7] , [АДР,:Ф=I0АО ] )
I8.             КОНЕЦ
19.             РЕЙС5С  МЕСТОIА
20.             РЕЙС55  МЕСТОIВ
2I. /*
22. //
```

Карта I ÷ 8 - управляющие операторы шага задания;

Карта 9 ÷ I8 - операторы задания отладки;

Карта I9 ÷ 20 - данные для отлаживаемой программы;

Карты 2I ÷ 22 - указывают на окончание шага задания;

Карты I и 2 - имеют стандартное назначение;

Карта 3 - оператор СД, имя которого СИСВЫВОД, задает информацию о файле и устройство для печати сообщений диспетчера заданий и программы "отладчик". Имя СД стандартное.

Карта 4 - оператор СД задает информацию о файле и устройство, с которого загружается отлаживаемая программа;

Карта 5 - оператор СД задает информацию о файле и устройство, с которого загружается таблица идентификаторов отлаживаемой программы. Данный оператор присутствует во входном потоке, если есть адреса в операторах задания на отладку, заданные через идентификаторы;

Карта 6 ÷ 7 - операторы СД, с помощью которых пользователь задает информацию о файлах и устройства для своей программы;

Карта 8 - оператор СД. указывающий на окончание управляющих операторов диспетчера заданий и что управление необходимо передать программе, указанной в операторе ВПЛ (карта 2) имя СД стандартное

Карта 9 ÷ I8 - задают задание на отладку;

Карты I9 ÷ 20 - задают данные для отлаживаемой программы;

Карты 2I ÷ 22 - указывают об окончании шага задания (карта 2I) и всего задания (карта 22).

Если потребитель хочет передать данные для своей программы через параметр ПАРМ в операторе ВПЛ, то он должен после указания имени программы (ПРГ=ИЕГОТЛАД), записать параметр ПАРМ, в котором указывается информация, передаваемая потребительской программе. Программа "отладчик" при передаче управления отлаживаемой программе в регистре I (РI) передает адрес, где размещена информация, указанная в поле ПАРМ.

Вся информация об отлаживаемой программе и таблице идентификаторов как о файле данных указывается в операторах ОД.

Имя ОД, заданное в операторе отладки СТАРТ (ОДПГ=ОДОТЛАЖП, карта 9), должно совпадать с именем ОД, указывающего устройства, с которого вводится отлаживаемая программа как файл данных (карта 4).

Сообщения, выдаваемые на АЦПУ и пульт программиста (ПП) при работе программы "отладчик".

Сообщения, выдаваемые на УАЦП:

I. НАЧ КРТ НЕ СТАРТ - указывает, что начальная карта не СТАРТ. Кроме того на пульт программиста будет выдан потребительский АВОК ПС272 и входной поток задания на отладку закрывается.

2. ОШ В ПОЛЕ НАЗВН - указывает, что ошибка в поле названия оператора отладки (первый символ не буква).

3. ОШ В ПОЛЕ ОП-ЦИИ - указывает, что ошибка в поле операции оператора отладки (имя оператора отладки отлично от СНИМК, СЛЕДП, ВСТАВ, ПРИСВ, КОНЕЦ).

4. ОШ В ПОЛЕ ОП-ДА – указывает, что ошибка в поле операнда оператора отладки (т.е. один из операндов задан неправильно).

При выдаче на УАЦП сообщений 1-4 на пульт программиста будет выдан потребительский АВОК П 0272, а также на УАЦП - дополнительное сообщение:

НЕВЫПОЛ ОПЕРАТОР

5. ПОВТОРЕН СТАРТ – указывает на повторение оператора СТАРТ в задании на отладку.

6. ПОВТОРЕН ПАРАМ – указывает, что в поле операнда повторен параметр (кроме параметра ОБЛП в операторе СНИМК).

7. ОТСУТ РАБ ОПЕРАТ – указывает, что в задании на отладку отсутствует рабочий оператор отладки (т.е. СНИМК, СЛЕДП, ПРИСВ, ВСТАВ).

При выдаче на УАЦП сообщений 5-7 на ПП будет выдан потребительский АВОК П0272 и входной поток закроется.

8. В СЛЕДЕ ВСУ ОТЛ – указывает, что в области прослеживания встретилась контрольная точка не данного оператора СЛЕДП, т.е. контрольная точка СЛЕДП другого оператора отладки. Такое сообщение будет выдано, так как в области прослеживания не допускаются контрольные точки других операторов СЛЕДП.

При выдаче на УАЦП сообщения 8 на ПП будет выдан потребительский АВОК ПОСОН. При этом входной поток закрывается и отладка программы прекращается.

Краткое описание работы программы "отладчик". Программа "отладчик" состоит из 2-х- частей, помещенных в библиотеку под именами ИЕГОТЛАД и ИЕГОТЛД2 соответственно, которые работают последовательно, занимая одну и ту же область, т.е. по окончании работы первой части в память, занятую ею, по макрокоманде ЗАМЕН записывается вторая часть.

При работе первой части расшифровываются операторы отладки и составляются таблицы: операторов отладки, блока управления отладкой, описателя таблицы операторов отладки. Кроме этого происходит загрузка 2-й части.

При работе второй части загружается отлаживаемая программа по макрокоманде ЗАГР, делаются вставки, замены, удаления в отлаживаемой программе (если это указал пользователь в задании на отладку) расставляются "ВСУ отладка" на указанных программистом контрольных точках и затем передается управление с помощью макрокоманды СВЯЗЬ отлаживаемой программы.

Если при выполнении отлаживаемой программы, встречается "ВСУ отладка", управление вновь передается 2-й части. После этого происходит анализ типа оператора, необходимые действия с выдачей на УАЦП информации и управление вновь возвращается отлаживаемой программе. По окончании работы отлаживаемой программы управление вновь возвращается программе "отладчик", которая заканчивает свою работу. Если при работе программы "отладчик" встречаются ошибки в задании на отладку или уже при выполнении отлаживаемой программы в режиме отладки, то на УАЦП выдаются соответствующие сообщения.

Примеры оформления входных потоков, заданий на отладку и информация, выдаваемая при работе операторов отладки, приведены в приложении 4.

где РС – означает общие регистры;

АТРНС – означает, что печатается относительный адрес;

АЗАГР – означает, что печатается адрес загрузки, т.е. где помещена в оперативной памяти команда или константа в момент выполнения отлаживаемой программы;

СПМН – означает, что печатается мнемонический код операции;

ОП – означает, что печатается 16-ричный код операции;

РР – означает, что печатаются номера регистров, длины;

БІ(Б2) – означает, что печатается номер базового регистра;

СМІ(СМ2) – означает, что печатается смещение;

СОДЕРЖ РЕГІ (СОДЕРЖ РЕГ2) – означает, что печатается содержимое соответствующего регистра;

СДЖБІ(СДЖБ2) – означает, что печатается содержимое соответствующего базового регистра;

АДРІ(АДР2) – означает, что печатается соответствующий адрес, равный сумме соответствующего базового регистра и смещения;

КУ – означает, что печатается код условия;

СПЕРАНДІ (СПЕРАНД2) – означает, что печатается соответствующий операнд

ТИНФ – указывает тип печатаемой информации;

АНАЧ (АНЧ) – указывает адрес начала области;

АКОН (АК) – указывает адрес конца области;

АКТФ – указывает относительный адрес контрольной точки.

КОДЫ ЗАВЕРШЕНИЯ

Код завершения - это число, определяющее условие, вызвавшее ненормальное завершение задачи.

Существует два типа колов завершения: системные и коды потребителя. Системные коды выдаются управляющей программой и печатаются как 3-значные шестнадцатеричные числа; коды потребителя выдаются потребительскими программами и печатаются как 4-значные десятичные числа.

Системный код указывает программисту природу ошибки, которую обнаружила управляющая программа. Код потребителя есть код завершения, выдаваемый программой в макрокоманде АВОК. В таблице приводятся системные коды завершения. Слева от кода завершения указывается макрокоманда или программа, при использовании которой может быть выдан данный код. Справа объясняется возможная причина, по которой задача ненормально завершена.

Коды завершения выдаются на пульт оператора. Системные коды завершения печатаются с признаком С, потребительские - с признаком П. Следует обратить внимание на то, что при работе входного потока сообщения об аварийном окончании задачи могут быть выданы двух видов сообщения супервизора и сообщения, выданные диспетчером заданий.

Пример сообщений аварийного завершения в системной программе

```
ИЕА000И  АВ.ОК   00006160  80113000 FPDD0005000006280
ИЕФ017И  ЗАДАЧА      ШАГ1        АВОК     С113
```

Пример аварийного завершения потребительской программы

```
ИЕА000И  АВ.ОК   МАТР     80000002A    PPI0000500000AE44
ИЕФ017И  МАТР         ШАГ2I              АВОК  П0042  ,
```

где 42 - это переведенный в десятичную систему код завершения 2A.

В таблице I3 и I4 приведены системные коды завершения. Коды завершения обслуживающих и сервисных программ приведены в описаниях этих программ.

Системные коды завершений

Таблица I3

Мнемоника программы	Код	Возможные причины возникновения нарушения
I	2	3
ВКПВСУ	400	а) БВВ не в границах памяти задачи; б) Адрес БУУ в БВВ задан неправильно; в) Ключ защиты в БУЗ не совпадает с ключом защиты в БУУ (УВВ не закреплено за шагом задания); г) адрес БУС не кратен слову; д) БУС не в границах области памяти задачи; е) БУД не в границах области памяти задачи;
	800	В БС канала обнаружен один из битов: "нарушение защиты" или "ошибка в программе".
БУСВНВСУ	I0E	а) Программа, используемая БУСВНВСУ, имеет ненулевой ключ защиты; б) Индекс таблицы внимания БУУ равен нулю (т.е. в ОС не предусмотрена обработка сигнала "внимание" от данного УВВ).
ВСУПРЕРВ	FXX	В ОС нет программы обслуживания ВСУ с данным кодом. XX - код вызова супервизора.
ЖДАТЬ	20I	Адрес БУС указан неправильно (адрес БУС не кратен слову, или находится вне памяти задачи).
СТМЕТ	I02	Адрес БУС указан неправильно. (Адрес БУС не кратен слову или же находится вне памяти постируемой задачи).
	202	а) Разрушена информация в БУС (адрес БУП, записываемый ОС в БУС, не кратен слову, или же находится вне памяти постируемой задачи); б) Программой потребителя разрушены системные блоки, построенные ОС в области памяти задачи (нарушение связок в БУП).
ВЫДПМ	604	Программа потребителя использует не выделенную ей память (нарушение связок свободных участков памяти).

Продолжение таблицы I3

1	2	3
	804	Нет свободного участка памяти требуемого размера.
ОСВПМ	605	Освобождение памяти, не запрашиваемой ранее. Перед обращением к ОСВПМВСУ программой потребителя была использована не выделенная ей память (нарушение связок свободных участков памяти).
ВЫХОДВСУ	203	Использование потребителем памяти не выделенной ему, в результате чего разрушена системная информация (нарушение связок в БУП).
	303	Разрушена системная информация (ТВВЗ).
ЗПРЕСВСУ	417	а) ЗПРЕСВСУ использует не системная программа; б) Система неправильно сгенерирована (нет запрашиваемого ресурса).
ОСРЕСВСУ	418	а) ОСРЕСВСУ использует не системная программа. б) Система неправильно сгенерирована (нет освобождаемого ресурса).
ПРОЦЗАВ	103	Разрушена информация, используемая системной программой (адрес БУД из БПД не в границах памяти задачи).
ВКЛБУПАВ	115	Момент подключения асинхронной программы совпадает по времени с работой программы ВСУ.
ОТКЗД	21В	Адрес БУЗ указан неверно.
	31В	Использование потребителем памяти, не выделенной ему, в результате чего разрушена системная информация (нарушение связок в БУП).
ОБРЗД	41А	а) Адрес БУС не кратен слову; б) БУС не в границах области памяти задачи.
	Е1А	Нет свободного БУЗ.

Продолжение таблицы I3

I	2	3
СТАРТВСУ	РI9	СТАРТВСУ использует не системная программа.
ЗАМЕН*	207	ЗАМЕН использована в программе асинхронного выхода, что не допустимо.
СВЯЗЬ* ЗАГР*	206	а) Программа потребителя использовала не выделенную ей память, в результате разрушена системная информация (нарушение связок в БУП загруженных программ); б) Не заданы ни название программы ни БУД; в) БУД не в границах памяти задачи.
ОТСЕЧ	206	Программа потребителя использовала не выделенную ей память, в результате разрушена системная информация (нарушение связок в БУП загруженных программ).
ПРГПРЕРВ	ССХ	Программное прерывание. Х - код программного прерывания.

Код	Причина программного прерывания
I	несуществующая операция
2	привилегированная операция
3	двойная подмена
4	нарушение защиты памяти
5	неправильная адресация
6	нарушение спецификации
7	неправильные данные
8	переполнение при фиксир.запятой
9	недопустимое деление при фиксир. запятой
A	десятичное переполнение
B	недопустимое деление десятичных чисел
C	положительное переполнение порядка
Д	отрицательное переполнение порядка
E	потеря значимости
F	недопустимое деление при плавающей запятой

*В таблице I4 приведены коды завершения программ "локатор", "загрузчик", работающих от имени этой макрокоманды.

Продолжение таблицы I3

I	2	3
ВНШПРЕРВ	3IC	Истек интервал времени, указанный в операторе ВПЛ
ЗВИ	3IC	Система неправильно сгенерирована (исчерпаны все элементы в таблице элементов таймера).
ВКОВСУ	IIC	Текущее задание отменено по указанию оператора (команда СТМЕНА)
ЗКО	EID	Адрес текста сообщения не кратен слову.
ЗКОС	EID	Потребитель используя макрорасширение ЗКОО задал адрес ответа не кратным слову..
	EID	Адрес ответа или адрес БУС задан вне границ области памяти задачи.
	IID	а)Длина текста сообщения больше I20 байтов; б) Длина заказанного текста ответа больше I20 байтов.
	2ID	Ошибка в ОС. Нет свободного элемента в таблице элементов запроса (система неправильно сгенерирована).
	3ID	Текст сообщения содержит одни пробелы.
ВВОД	OOI	В БУД в поле ВЫБОР указан код ЗШ и имеет место одна из нижеследующих ситуаций: а) Ошибка ввода-вывода (в БСК сигнал "неправильная длина" при чтении неблочных записей фиксированной длины); б) При чтении блочных записей фиксированной длины оказалось, что длина считанной физической записи не кратна длине логической записи; в) Длина физической записи равна нулю; г) При чтении записей переменной длины оказалось, что длина считанной физической записи не равна длине, указанной в "шапке" этой записи; д) Ошибка по циклическому контролю при чтении с перфоленты без контрольного.
ВЫВОД	OOI	Ошибка ввода-вывода при записи, и в БУД в поле ВЫБОР указан код ЗШ.

Продолжение таблицы I3

I	2	3
ВВОД ВЫВОД	002	Обнаружен конец файла при вводе или конец тома при выводе, а в БУДе не указан адрес подпрограммы потребителя "конец файла".
ВВОД ВЫВОД	003	Длина записи (буфера, блока) в БУДе не корректна, т.е. не выполнено условие ДЛБУФ ⩾ ДЛБЛОК ⩾ ДЛЗП ⩾ 0
ВВОД	004	В БУДе в поле ВЫБОР указан код ДП, а дальнейшая работа оказалась невозможной (только при вводе блочных записей).
ВЫВОД	004	Физическая запись, которая должна быть выведена на магнитную ленту содержит меньше 32 байтов.
ЗАКР	II4	Программой потребителя разрушены или адреса (адреса блоков вне границ области памяти задачи), или сами управляющие блоки.
	2I4	Ошибка ввода-вывода при операциях установки магнитной ленты.
	6I4	Ошибка ввода-вывода при записи метки или маркера на МЛ.
	7I4	Ошибка ввода-вывода при чтении метки на МЛ.
	8I4	При считывании метки не обнаружены признаки метки (EOF1 или HDR 1) или не совпадает порядковый номер файла ПНФ записанный в поле метки, с НПФ, указанным в ОД.
ОТКР	II8	а) Открывается БУД, который ранее не был закрыт; б) Нет соответствующего оператора ОД в ТВВЗ (в управляющих операторах задания пропущен соответствующий оператор ОД); в) В БУД и длина блока и длина буфера равны нулю; г) В БУД длина буфера меньше длины блока; д) Ввод с записывающего или вывод на считывающее устройство (например, в ОТКР для записи не указан код ЗП);

Продолжение таблицы I3

I	2	3
		в) Обратное чтение с устройства, отличного от магнитной ленты.
	2I3	а) Том магнитной ленты, ранее использовавшийся для работы с непомеченными файлами пытаются использовать для записи помеченных файлов. Необходимо перезаписать метку тома;
		б) При считывании метки файла на МЛ происходит несовпадение: -первые 4 байта в метке не совпадают с HDR I или EOFI; -полей метки с полями соответствующего оператора ОД (последовательный номер файла, номер тома) или задан помеченный файл с номером большим номера последнего, имеющегося файла на ленте.
ОТКР	3I3	а) Требуемый файл не опознан; б) При считывании метки считанная зона ≠ 80 байтам; в) При поиске файла вышли на аварийный конец ленты или на начало ленты.
	4I3	Попытка открыть файл на магнитной ленте при наличии на ней уже одного открытого.
	5I3	Программой потребителя разрушены или адреса (адреса блоков вне границ области памяти задачи), или сами управляющие блоки, построенные в области памяти потребителя.
	6I3	При чтении или записи метки файла - сигнал "ошибка в данных".
	7I3	На сообщение с ответом (в случае, когда файл защищен по времени) "пиши" либо "нет" оператор посылает ответ "нет".

Коды завершения программы "докатора" и "загрузчика"

Таблица I4

Библиотека		Л-формат		К-формат	
Код	Причина	Код	Причина	Код	Причина
I	2	3	4	5	6
			Д О К А Т О Р		
006	При поиске назад- не БОП или не метка При поиске вперед - не БОП (не та лента)	006	а) Неправильно задан оператор ОД; б) Ошибка при обращении к программе динамического вызова	006	В первой карте нет кода карты 02.
306	Нет программы в библиотеке а) Ошибка программиста при использовании макрокоманд динамического вызова (СВЯ ЗБ,ЗАГР,ЗАМЕН); б) Ошибочная карта ВПЛ; в) Не та лента	406	Имя программы, заданное в макрокоманде динамического вызова не совпадает с именем, указанным в БОП	506	В первой карте в поле "тип карты" не СВИ
				606	Первая карта СВИ не СТАРТ
806	Прочитана метка тома и оказалось, что это не СИСББЛ			406	В карте СТАРТ идентификатор не совпадает с названием, указанным в макрокоманде динамического вызова
407	На команде ПМВ вышли на аварийный конец ленты Лента СИСББЛ неправильно оформлена - нет 2-х маркеров в конце ленты				
I06	Ошибка ввода-вывода				

Продолжение таблицы I4

I	2	3	4	5	6

З А Г Р У З Ч И К

507	Адрес входной точки нечетный.
706	Входная точка вне программы (адрес входной точки больше длины программы)
807	Не хватает памяти (указана длина программы меньше, чем следует).
A06	Переполнение адреса при формировании адресной константы
C06	Адресная константа не в программе (сформированный адрес не относится к области программы).
906	Следующим блоком должен быть ТКТ, а считан не ТКТ.
707	Длина текста ТКТ равна нулю.
I07	Следующим блоком должен быть МОД, а считан не МОД.
607	Вместо ТКТ или МОД считан маркер
I06	Ошибка ввода-вывода
B06	Имеется оператор ВНЕШ (программа требует компоновки с другой программой).
D06	В поле кода карты код ≠ 02.
E06	Нарушен порядок следования карт (только для К-формата на перфокартах)
F06	I-я карта - не СВИ и не ТКТ.
007	Карта СВИ не ВХОД (СТАРТ или ВНЕШ).
307	Не определен тип карты

7. ТРАНСЛЯТОРНЫЕ КОМАНДЫ

7.1. Общие сведения

Подобно тому, как машинные команды используются для задания последовательности операций во время выполнения программы, трансляторные команды используются для задания определенных операций во время трансляции. Операторы трансляторных команд, в отличие от операторов машинных команд, не всегда приводят к созданию машинных команд в транслированной программе. Некоторые трансляторные команды, такие, как ОК (Определить константу) и ОП (Определить область памяти) не создают машинных команд, а резервируют области памяти для констант и другой информации. Другие трансляторные команды, как например, РАВНО и ПРПСК действуют только во время трансляции; они ничего не создают в транслированной программе и, следовательно, не изменяют счетчик адреса.

Трансляторных команд 16. Перечисленные ниже трансляторные команды (в скобках указана латинская мнемоника) в зависимости от своих функций могут быть разделены на такие группы:

Команда определения идентификатора:

РАВНО (EQV) - Определить идентификатор.

Команды определения данных:

ОК (DC) - Определить константу;

ОП (DS) - Определить область памяти;

УСК (CCW) - Определить управляющее слово канала.

Команды связи программ:

СТАРТ (START) - Начать трансляцию;

ВХОД (ENTRY) - Определить идентификатор входной точки;

ВНЕШ (EXTRN) - Определить внешний идентификатор.

Команды управления базированием:

ИСПЛЗ (USING) - Использовать базовый регистр

ОТМЕН (DROP) – Отменить базовый регистр.

Команды управления печатным документом:

НСТР (EJECT) – Начать новую страницу;

ПРПСК (SPACE) – Пропустить строку в печатном документе.

Команды обработки табличной информации:

ОШ (DCS) – Определить шаблон для таблицы;

Т (T) – Определить элементы строки таблицы по шаблону.

Команды управления транслятором:

ИЗМЕН (ORG) – Изменить счетчик адреса;

УНОП (CNOP) – Условное "нет операции";

КОНЕЦ (END) – Окончить трансляцию.

7.2. Команда определения идентификатора

РАВНО – определить идентификатор.

Оператор РАВНО используется для определения идентификатора путем присваивания ему значения и атрибута длины выражения, стоящего в поле операнда. Формат оператора РАВНО:

Название	Операция	Операнд
Идентификатор	РАВНО	Выражение

Выражение в поле операнда может быть абсолютным или перемещаемым. Все идентификаторы в этом выражении должны быть ранее определены.

Идентификатор в поле названия имеет такие же атрибуты длины и значения, как и выражение в поле операнда. Атрибутом длины идентификатора является атрибут длины крайнего левого терма выражения. В случае команды РАВНО со * или с самоопределенным термом в поле операнда атрибут длины равен I. Атрибутом значения этого идентификатора является значение выражения.

Команда РАВНО является средством присваивания идентификаторам номеров регистров, непосредственной информации и других произвольных значений. Следующие примеры показывают, как это можно сделать:

Название	Операция	Операнд
РЕГИСТР 2	РАВНО	2 (общий регистр)
ПРОВЕРКА	РАВНО	X'5F' (непосредственная информация)

Идентификатору РЕГИСТР 2 оператор РАВНО присваивает значение 2, а идентификатору ПРОВЕРКА - 5F.

Для сокращения времени программирования программист может сопоставить идентификатор с некоторым часто встречающимся выражением, а затем использовать этот идентификатор вместо выражения в поле операнда. Например:

Название	Операция	Операнд
ВРЕМЯ	РАВНО	АЛЬФА-БЕТА+ГАММА

ВРЕМЯ определяется как АЛЬФА-БЕТА+ГАММА и его можно использовать вместо выражения. Надо заметить, что АЛЬФА, БЕТА, ГАММА должны быть предварительно определены. Если окончательный результат выражения отрицательный, то он трактуется, как если бы он был положительный.

Если идентификатор в поле названия и (или) выражение в поле операнда отсутствуют или заданы неправильно, оператор РАВНО не выполняется и печатается сообщение об ошибке.

7.3. Команды определения данных

7.3.1. DC - определить константу. Оператор DC используется для размещения в памяти постоянной информации. Он может определять одну константу или группу констант, тем самым освобождая програм-

миста от необходимости ставить отдельный оператор для определения каждой константы.

Можно указывать константы следующих типов: с фиксированной запятой, с плавающей запятой, десятичные, шестнадцатеричные, символьные, адресные.

Константы данных называются просто константами, если они не создаются из адресов памяти. В противном случае их называют адресными константами.

Формат оператора ОК:

Название	операция	Операнд
Идентификатор или пробел	ОК	Операнд, описывающий константу в форме, приведенной ниже

Операнд состоит из четырех подполей: первые три описывают константу, а четвертое - это сама константа (или константы). Первое и третье подполя могут быть опущены, второе и четвертое - обязательны. Все константы, специфицированные таким образом, должны быть одного и того же типа. Описания подполей, которые предшествуют константам, относятся ко всем константам. Внутри любого из подполей не могут стоять пробелы (если пробелы не являются символами символьной константы или символьного самоопределенного терма). Пробелы не могут также стоять между подполями операнда.

Подполя каждого оператора ОК располагаются в следующей последовательности:

I	II	III	IV
Коэффициент кратности	Тип	Модификатор	Константа(ы) .

Атрибутом значения идентификатора, именующего оператор ОК, является адрес крайнего левого байта (после установки границ) первой или единственной константы. Атрибут длины зависит от двух обстоятельств: типа определяемой константы и наличия модификатора

длины. Если модификатор длины отсутствует, длина считается неявной. Если указано несколько констант, то атрибут длины - это длина в байтах первой константы.

Установка границ зависит от типа константы и от наличия модификатора длины. Некоторые типы констант размещаются только в границах байта. Другие константы размещаются в различных границах (полуслов, слов или двойных слов), если отсутствует модификатор длины. В случае явного задания длины установка границ не производится. Для того чтобы произвести любой вид установки границ может быть использована команда ОП (Определить область памяти). Это объясняется в пункте "ОП - определить область памяти".

Байты, которые надо пропустить для установки поля на нужную границу, не считаются частью константы. Другими словами, счетчику адреса дается приращение для настройки его на нужную границу (если это приращение необходимо). Таким образом, идентификатор, именующий константу, не получит значение адреса пропущенных байтов.

Все байты, пропущенные при размещении операторов, которые не являются транслируемой информацией, не очищаются. Байты, пропущенные для размещения оператора ОП, тоже не очищаются. Однако, байты, пропущенные для размещения оператора ОК - очищаются.

Коэффициент кратности, если не указан, вызывает повторение константы столько раз, каково его значение. Коэффициент кратности может быть десятичным самоопределенным термом без знака и применяется после того, как константа уже один раз транслирована.

Нулевой коэффициент кратности не допускается (исключение: типы С, Н, Р, D в операторе ОП). Коэффициент кратности может быть опущен, в этом случае будет создана только одна константа или одна группа констант.

Для адресной константы (т.е. для константы А-типа) коэффициент кратности не допускается.

Т и п. Подполе типа определяет тип константы. По указанию

типа транслятор решает, как следует истолковать константу и транслирует ее в соответствующий машинный формат.

Тип указывается кодом из одной буквы согласно следующей таблицы:

Код	Тип константы	Машинны формат
C	Символьная	8-разрядный код для каждого символа.
X	Шестнадцатеричная	4-разрядный код для каждой шестнадцатеричной цифры .
F	Фиксированная запятая	Двоичное со знаком; фиксированная запятая; слово.
H	Фиксированная запятая	Двоичное со знаком; фиксированная запятая; полуслово.
E	Плавающая запятая	Короткий формат; плавающая запятая, слово.
D	Плавающая запятая	Длинный формат; плавающая запятая, двойное слово.
P	Десятичная	Уплотненный десятичный формат
Z	Десятичная	Зонированный десятичный формат
A	Адресная	Значение адреса; слово.

Модификаторы, которые представляют третье подполе операнда, описывают длину константы в байтах (в отличие от неявной длины) и порядок константы. Если для описания константы используются оба модификатора, они записываются в следующей последовательности: длина, порядок.

Модификатор длины указывает явную длину константы в байтах и записывается в виде: Ln , где n – десятичный самоопределенный терм без знака. (Максимальные допустимые значения модификатора длины для различных типов констант будут приведены ниже).

Для констант типа H, E, D модификатор длины не допускается.

Если модификатор длины отсутствует, используется неявная длина константы:

Модификатор порядка записывается как En , где n – десятич-

ный самоопределенный терм со знаком или без знака. Если знак опущен, о подразумевается плюс. (Максимальные значения модификатора порядка также будут приведены ниже).

Модификатор порядка означает степень 10, на которую должна быть умножена константа перед преобразованием ее в соответствующий машинный формат, причем модификатор порядка допустим только для констант типа P, H, E, D .

Не следует смешивать модификатор порядка с порядком самой константы, который указывается как часть самой константы. Модификатор порядка одинаково воздействует на каждую константу в операнде, в то время как порядок константы относится только к данной константе.

Например, сама константа может иметь порядок +2 и ей предшествует модификатор порядка +5. В итоге эта константа имеет порядок +7.

Четвертое подполе операнда - константа (или константы), описанная предшествующими подполями. Константы данных заключаются в кавычки (одиночные), адресные константы - в круглые скобки. При указании двух или более констант они разделяются запятыми и в кавычки или скобки заключается вся последовательность констант.

Таким образом, для указания константы (констант) используется один из следующих форматов:

Одиночная константа	Групповые константы
'константа'	'константа,...,константа'
(константа)	(константа,...,константа)

Групповые константы не допускаются для символьных и шестнадцатеричных констант.

Все типы констант, кроме символьной (C), шестнадцатеричной (X), уплотненной десятичной (P), зонированной десятичной (Z) размещаются в соответствующих границах, если только не указан модификатор длины. При указании модификатора длины установка границ не производится. При групповом задании констант необходимая уста-

новка границ производится только для первой константы, остальные автоматически попадут в требуемые границы. Так, например, для операнда, который задает пять констант - слов, первая константа будет размещена в границе слова, а остальные автоматически попадут в границы слов.

Общий объем памяти, требуемый для размещения константы в памяти, равен произведению длины константы на количество их в операнде и на коэффициент кратности (если он задан) плюс байты, пропущенные для установки границы для первой константы.

Если адресная константа содержит обращение к счетчику адреса, то используемое значение счетчика адреса является адресом первого байта в памяти, который займет эта константа. Таким образом, если несколько адресных констант в одной и той же команде обращаются к счетчику адреса, то значение счетчика адреса меняется от константы к константе.

7.3.2. Символьная константа - С. В операнде может быть указана только одна символьная константа. Так как групповые константы в операнде разделяются запятыми, то попытка указать несколько символьных констант приведет к тому, что запятая, разделяющая константы , будет трактоваться как символ.

Каждая одиночная кавычка в символьной константе или знак & должны быть представлены парой кавычек или амперсендов, соответственно. В памяти будет записана только одна кавычка или один амперсенд.

Максимальная длина символьной константы - 256 байтов. Установка границ не производится. Каждый символ транслируется в один байт. Если модификатор длины отсутствует, длина символьной константы в байтах равна количеству символов в константе. Если же модификатор длины указан, длина константы зависит от следующего:

I. Если количество символов превышает указанную длину, то опускается столько крайних правых байтов, сколько необходимо.

2. Если количество символов в константе меньше указанной длины, лишние крайние правые байты заполняются пробелами.

В следующем примере атрибут длины идентификатора ПОЛЕ равен I4:

Название	Операция	Операнд
ПОЛЕ	DC	C'СУММА␣РАВНА␣I4'

Однако, в следующем примере задан модификатор длины, т.е. атрибут длины равен I7, и после трансляции в памяти справа от четверки будут записаны три пробела, а именно:

СУММА␣РАВНА␣I4␣␣␣

Название	Операция	Операнд
ПОЛЕ	DC	C I7'СУММА␣РАВНА␣I4'

В следующем примере атрибут длины ПОЛЕ равен I4, хотя в операнде I5 символов. Два знака & считаются как один символ.

Название	Операция	Операнд
ПОЛЕ	DC	C'РЕЗУЛЬТАТ␣-␣&&2'

В следующем примере указана длина 4, но в константе 5 символов:

Название	Операция	Операнд
ПОЛЕ	DC	3C L 4 'ABCDE'

Транслированная константа примет вид:

ABCDABCDABCD

Если же вместо 4 была указана длина 6, созданная константа имела бы вид:

ABCDE␣ABCDE␣ABCDE␣

Символьная константа может быть указана как литерал:

Название	Операция	Операнд
	ПП	ОБЛАСТЬ (I2,2),=3C'ABCD'

7.3.3. Шестнадцатеричная константа - X. Шестнадцатеричная константа состоит из одной или нескольких шестнадцатеричных цифр 0-9 и A-F. В операнде может быть указана только одна шестнадцатеричная константа. Максимальная длина шестнадцатеричной константы - 256 байтов (5I2 шестнадцатеричных цифр). Установка границ не производится.

Каждая пара шестнадцатеричных цифр, транслируется в один байт, Если указано нечетное число цифр, то в крайнем левом байте 4-е крайних левых разряда заполняются шестнадцатеричным нулем, в то время, как 4 крайних правых разряда содержат первую цифру.

Если модификатор длины не указан, неявная длина шестнадцатеричной константы вдвое меньше количества шестнадцатеричных цифр в константе (учитывая, что к нечетному числу цифр добавляется шестнадцатеричный нуль). Если же модификатор длины указан, константа обрабатывается следующим образом:

I. Если число пар шестнадцатеричных цифр превосходит указанную длину, то крайние левые цифры отбрасываются.

2. Если число пар шестнадцатеричных цифр меньше указанной длины, то слева дописываются шестнадцатеричные нули.

Шестнадцатеричная константа, состоящая из 8 шестнадцатеричных цифр удобна для представления слова. Константа в следующем примере устанавливает первый и третий байты слова в единицы.

Название	Операция	Операнд
СЧЕТЧИК	ОП ОК	ОР X'FF00FF00'

Так как в шестнадцатеричной константе установка границ не производится, а программист желает записать шестнадцатеричную константу в границе слова, то он должен перед константой поставить

оператор ОП с нулевым коэффициентом кратности.

В следующем примере цифра А при создании константы будет выброшена, т.к. указано 5 шестнадцатеричных цифр, а модификатор длины равен 2.

Название	Операция	Операнд
	DC	3XL2'A67FE'

Результирующая константа будет 67FE, что занимает указанные 2 байта. Затем она будет повторена три раза, согласно коэффициенту кратности. Если бы она просто была указана как X'A67FE', то результирующая константа имела бы шестнадцатеричный нуль в крайней левой позиции: 0A67FE.

В следующем примере шестнадцатеричная константа используется в качестве литерала и устанавливает в единицы 24-31 разряды регистра 5:

Название	Операция	Операнд
	SC	5,=X'FF'

7.3.4. Константы с фиксированной запятой P и H. Константа с фиксированной запятой записывается как десятичное число, за которым может следовать десятичный порядок. Это число может быть целым, правильной дробью или смешанной дробью.

Формат константы следующий:

1. Число записывается как десятичное значение со знаком или без знака. Десятичная точка может быть размещена перед, внутри или после числа, а также вообще отсутствовать - в этом случае подразумевается целое число. Если знак опущен, число считается положительным.

2. Порядок не обязателен. Если он указывается, то записывается непосредственно за числом как En, где n - десятичный самоопределенный терм со знаком или без знака, указывающий показатель

степени десяти. Порядок, перед преобразованием константы в двоичную форму, вызывает изменение значения константы в зависимости от указанной степени десяти.

Диапазон порядка: -85 ÷ +75. Порядок может превосходить допустимый диапазон при условии, что сумма порядка и модификатора порядка этот диапазон не превосходит.

После преобразования константы в двоичную форму производится округление константы, а затем она помещается в соответствующее поле согласно явной или неявной длине. Результирующее число будет отличаться от точного значения не более чем на единицу в последнем разряде. Если значение константы не помещается в поле, соответствующем явной или неявной длине, то знак числа теряется, крайние левые разряды усекаются до длины поля и затем это значение транслируется в данное поле.

Любой коэффициент кратности, если он есть, учитывается после того, как константа транслирована.

Отрицательное число представляется в дополнительном коде.

Для слова (F) подразумевается длина в 4 байта, для полуслова (H) - два байта.

Константа размещается в границах слова или полуслова, если длина задана неявно.

Для константы H-типа модификатор длины не допускается. Для констант F-типа с помощью модификатора длины можно указать любую длину до 8 байтов включительно. Причем, в этом случае установка границ не производится.

Максимальные и минимальные значения для констант с фиксированной запятой при заданных длинах следующие:

Длина	Максимальное	Минимальное
8	$2^{63}-1$	-2^{63}
4	$2^{31}-1$	-2^{31}
2	$2^{15}-1$	-2^{15}
1	$2^{7}-1$	-2^{7}

Оператор, приведенный ниже, создает поле из трех слов. Значением идентификатора СЛОВ^0 является адрес крайнего левого байта первого слова, а атрибутом длины – 4 (неявная длина для константы –слова с фиксированной запятой). Выражение СЛОВО+4 могло быть использовано для адресации второй константы (второго слова) в этом поле.

Название	Операция	Операнд
СЛОВО	ОК	ЗР'1592135'

Следующая константа перед преобразованием в двоичную форму умножается на 10 в степени +2:

Название	Операция	Операнд
ПОДУСЛВ	ОК	Н'3.52 Е+2'

Пример Н-константы как литерал-терма:

Название	Операция	Операнд
	СЛК	9,=Н'48 Е+5'

Последний пример указывает три константы. Модификатор порядка в этом случае используется для преобразования каждой константы:

Название	Операция	Операнд
ТРИКОНСТ	ОК	РЕ4'10,253.8,1452'

Перед преобразованием констант в машинный формат они примут

вид: IO 0000, 2538000, I4520000.

7.3.5. Константы с плавающей запятой - E и D . Константа с плавающей запятой записывается совершенно аналогично константе с фиксированной запятой.

Машинный формат чисел с плавающей запятой состоит из порядка, иногда называемого характеристикой, и дробной части, называемой мантиссой. Поэтому число, указанное как константа с плавающей запятой, должно быть преобразовано (транслятором) в дробное перед переводом его в соответствующий формат. Например, константа с плавающей запятой 27.35E2 представляет собой число 27.35, умноженное на IO в степени +2. Представленное как дробное, оно выглядит как 2735, умноженное на IO в степени 4, т.е. порядок константы отражает сдвиг десятичной точки.

Порядок константы переводится в двоичный эквивалент и дробная часть - в двоичное число. После преобразования константы в двоичную форму происходит округление дробной части в соответствии с неявной длиной и это число запоминается в соответствующем поле. Полученное число отличается от точного значения не более чем на единицу в младшем разряде.

Отрицательные дробные части представляются в прямом, а не в дополнительном двоичном коде.

Неявная длина для слова (E) и двойного слова (D) - 4 и 8 байтов, соответственно. Константа размещается в границах слова (E) или двойного слова (D). Модификатор длины для константы с плавающей запятой (E или D) не указывается.

Любой из следующих операторов может быть использован для указания 46.415 как положительной константы - слова с плавающей запятой. Отметим, что последние две константы содержат модификатор порядка. Последним стоит оператор машинной команды с операндом.

Название	Операция	Операнд
	ОК	E'46.4I5'
	ОК	E'464I5 E-3'
	ОК	E'+464.I5 E-I'
	ОК	E'+.464I5 E+2'
	ОК	EE2'.464I5'
	СЛН	4,=EE2'.464I5'

Каждая из записанных ниже констант представляется в машине двойным словом с плавающей запятой:

Название	Операция	Операнд
ПЛАВЗПТ	ОК	4'+46,-3.729,473'

7.3.6. Десятичные константы - P и Z . Десятичная константа записывается как десятичное число со знаком или без знака. Если знак опущен, то предполагается плюс.

Десятичная точка не обязательна. Модификатор порядка для десятичной константы не указывается. Максимальная длина I6 байтов. Установка границ не производится.

Положение десятичной точки не влияет на трансляцию константы, т.к. в отличие от констант с фиксированной и с плавающей запятой десятичная константа не преобразуется в двоичный эквивалент. Тот факт, что десятичная константа является целым, дробным или смешанным числом, не является существенным для ее трансляции. Более того десятичная точка не транслируется. Программист должен задавать данные с одинаковым положением запятой или программировать их так, чтобы десятичные точки выравнивались, так как десятичная точка не транслируется, программист должен помнить расположение ее.

Если указан зонированный формат (Z), каждая десятичная цифра транслируется в один байт. Крайний правый байт содержит знак и

крайнюю правую цифру.

8 разрядов		8 разрядов			8 разрядов	
Зона	Ц	Зона	Ц		Знак	Ц

4 разряда

Если указан уплотненный десятичный формат (Р), каждые две десятичные цифры транслируются в один байт. Крайняя правая цифра и знак транслируются в крайний правый байт.

8 разрядов		8 разрядов			8разрядов	
Ц	Ц	Ц	Ц		Ц	Знак

4 разряда.

Двоичное представление цифры такое же, как для шестнадцатеричных цифр 0-9. Для уплотненного и зонированного форматов знак плюс транслируется в шестнадцатеричную цифру А, знак минус - в цифру

Если в Р-константе указано четное число цифр, то одна цифра должна оставаться без пары, т.к. крайняя правая цифра образует пару со знаком. Таким образом, в крайнем левом байте крайние правые четыре разряда будут содержать левую цифру. Например,

Название	Операция	Операнд
	DC	P'+34.25'

После трансляции получим: 03425A.

Если модификатор длины отсутствует, неявная длина для любой константы равна числу байтов, занимаемых константой (с учетом формата, знака и возможности добавления нулей для уплотненного формата). Если же модификатор длины указан, константа обрабатывается следующим образом:

I. Если константа для своего представления требует меньше байтов, чем указано в модификаторе длины, то слева добавляется необходимое число нулей. Для зонированного десятичного формата в каждом добавленном байте помещается зонированный нуль. Для уплот-

ненного формата разряды каждого добавленного байта устанавливаются в нуль.

2. Если константа требует больше байтов, чем указано длиной, некоторое число крайних левых цифр или пар цифр (в зависимости от формата) теряется.

В общем случае, десятичная константа должна занимать не более I6 байтов.

Примеры десятичных констант:

Название	Операция	Операнд
	ОК	P'+I.25'
	ОК	Z'-843'
	ОК	Z'79.68'
	ОК	PL8'79.68,-3874,+2.3'
	ПЭФ	ВЫХОД,=PL 2 '+25'

После трансляции первая константа будет иметь вид I25A, вторая - 5354D3 и т.д.

Последний пример иллюстрирует использование уплотненного десятичного литерал-терма.

7.3.7. Адресные константы - А-типа. Адресная константа - это адрес ячейки, транслируемый в константу. Адресные константы обычно используются для загрузки базовых регистров допустимыми адресами памяти, что может использоваться для связи программ, транслированных независимо друг от друга.

Адресные константы, в отличие от констант других типов, заключаются в круглые скобки. Если в операнде указаны две или более адресные константы, они разделяются запятыми, и в круглые скобки заключается вся последовательность констант.

Адресная константа А-типа указывается как абсолютное или перемещаемое выражение, максимальное значение которого $2^{3I}-I$ (32

двоичных разряда /. Это значение в случае необходимости усекается слева в соответствии с явной или неявной длиной поля и помещается в крайние правые разряды поля. Для констант А-типа неявная длина - 4 байта; константа размещается в границах слова, если не указана длина. Если же длина указана, установка границ не производится. Указываемая длина зависит от типа выражения, используемого для константы. Для абсолютного выражения может использоваться длина I-4 байта, тогда когда для перемещаемого выражения - только в 3 или 4 байта. Ввиду того, что адресная константа, указанная как абсолютное выражение, не может использоваться в качестве адреса, тогда как адресная константа, указанная как перемещаемое выражение, задает адрес. В системе АСВТ адресация трехбайтная.

Коэффициент кратности не допускается.

В приведенных ниже примерах поле операнда оператора АКОНСТ содержит 4 константы, каждая из которых занимает 4 байта. В одной из констант имеется обращение к счетчику адреса: значением счетчика адреса будет адрес первого байта четвертой константы:

Название	Операция	Операнд
АКОНСТ	ОК	А /ЮЗ,ЦИКЛ,КОНЕЦ-НАЧАЛО, * + 4096/
	ЗГ	4,7= А/ЮЗ,ЦИКЛ,КОНЕЦ-НАЧАЛО * + 4096/

Второй оператор дает тот же выбор констант, указанных как литерал-термы /т.е. адресных литерал-констант/.

Обращение к счетчику адреса может производиться и в адресной литерал-константе /т.е.звездочка может использоваться в адресной константе, которая указывается в литерал-форме/. Значением счетчика адреса в этом случае будет адрес первого байта команды ЗГ. При перемещении программы в памяти все перемещаемые адресные

константы изменяются на так называемый коэффициент перемещения.

7.3.8. Кратные литерал-константы. Как уже было сказано, все литеральные константы располагаются в конце программы, начиная с границы двойного слова. Причем, программист при загрузке базового регистра должен предусмотреть некоторый объем памяти, требуемый транслятору для расположения этих констант.

Если в программе встречаются кратные литерал-константы, запоминается только одна из них. Литералы считаются кратными, если их спецификация тождественна.

Однако, если литерал-константа является константой А-типа, содержащей обращение к счетчику адреса, она запомнится, даже если она дублирует другую литерал-константу.

Следующий пример иллюстрирует, каким образом транслятор запоминает пары литералов:

X 'P4'	Хотя T в машине представляется как P4, обе
C 'T'	константы запоминаются.
X 3'0'	обе запоминаются
P 3'0'	
A(*+4)	обе запоминаются
A(*+4)	
X'FFFF'	тождественны; первая запоминается
X'FFFF'	

7.4. ОП - Определить область памяти

Оператор ОП используется для резервирования областей памяти и для наименования этих областей. Этот оператор может быть использован для символического определения рабочих областей памяти, областей ввода-вывода и т.д. Размер резервируемой области памяти ограничивается значением счетчика адреса.

Формат оператора ОП:

Название	Операция	Операнд
Идентификатор или пробел	ОП	Один операнд

Формат операнда ОП совпадает с форматом операнда ОК, т.е. используются те же подполя и в той же последовательности, за некоторыми исключениями. Во-первых, допустимы только следующие типы: C, H, P и D .

Во-вторых, допускается модификатор длины только для символьной константы. Максимальное значение модификатора длины - 65535.

В-третьих, указание самой константы недопустимо.

Значением идентификатора, стоящего в поле названия оператора ОП, является адрес крайнего левого байта зарезервированной области памяти. Атрибут длины его - длина (явная или неявная) указанного типа поля.

Символьное поле (C) имеет неявную длину в I байт, поэтому, чтобы зарезервировать несколько байтов, необходимо указать либо коэффициент кратности, либо модификатор длины.

Поля H, P и D имеют неявные длины в 2, 4, 8 байтов, соответственно. Следовательно, для резервирования большой области памяти нужно использовать коэффициент кратности. Байты, пропущенные для установки границ, не очищаются.

Для определения четырех IO-байтных полей и одного IOO-байтного поля, можно использовать следующие операторы ОП:

Название	Операция	Операнд
ПОЛЕ	ОП	4CL.IO
ПОЛЕIOO	ОП	CL.IOO

Хотя ПОЛЕ может быть указано как одно 40-байтное поле, приведенное определение операнда является более удобным при употреблении ПОЛЕ в качестве операнда машинной команды формата ПП.

Ниже приведены примеры ОП-операндов:

Название	Операция		Операнд
ПОЛЕ I	ОП	CL80	(одно 80-байтное поле; атрибут длины - 80)
ПОЛЕ 2	ОП	80C	(80-однобайтных полей; атрибут длины - I)
ПОЛЕ 3	ОП	6F	(шесть слов; атрибут длины - 4)
ПОЛЕ 4	ОП	D	(одно двойное слово; атрибут длины - 8)
ПОЛЕ 5	ОП	4H	(4 полуслова; атрибут длины - 2)

Примечание. Оператор ОП резервирует область памяти; но не очищает ее.

7.5. Особый случай употребления коэффициента кратности

При использовании соответствующего типа поля (D , F, H) с нулевым коэффициентом кратности можно установить счетчик адреса на границу двойного слова, слова, полуслова.

Например, нижеприведенные операторы настраивают счетчик адреса на границу двойного слова и резервируют область памяти в I28 байтов (адрес крайнего левого байта этого поля настроен на границу двойного слова):

Название	Операция	Операнд
	ОП	0D
ОБЛАСТЬ	ОП	CL I28

Оператор ОП с нулевым коэффициентом кратности может использоваться для наименования области памяти без ее резервирования. Дополнительные операторы ОП и (или) ОК могут затем зарезервировать область памяти и именовать поле внутри области (при использовании ОК - создавать константы). Предположим, например, что в какую-то

область памяти должна быть введена 80-символьная запись, имеющая следующий формат:

Позиции 5-10	- № платежной ведомости
Позиции 11-30	- фамилия служащего
Позиции 31-36	- дата
Позиции 47-54	- общая зарплата
Позиции 55-62	- вычеты.

Следующий пример иллюстрирует применение оператора ОП для наименования области записи, определения полей этой области и распределения памяти для них. Первый оператор именует всю область определением идентификатора ВЕДОМОСТ. Этот оператор присваивает ВЕДОМОСТ атрибут длины 80, но не резервирует область памяти. Аналогично, пятый оператор именует 6-байтную область определением идентификатора ДАТА; три следующие оператора определяют конкретные поля области ДАТА и распределяют память для них. Второй, девятый и последний операторы используются для указания пробелов и потому не именуются:

Название	Операция	Операнд
ВЕДОМОСТ	ОП	OCL80
	ОП	CL4
НОМЕРВЕД	ОП	CL6
Имя	ОП	CL20
ДАТА	ОП	OCL6
ДЕНЬ	ОП	CL2
МЕСЯЦ	ОП	CL2
ГОД	ОП	CL2
	ОП	CL10
ЗАРПЛАТА	ОП	CL8
ВЫЧЕТЫ	ОП	CL8
	ОП	CL18

СВОДНОЕ О КОНСТАНТАХ

Тип	Неявная длина в байтах	Размещение	Диапазон модификатора длины	Спецификация	Кол-во констант в операнде	Диапазон модификатора порядка	Отбрасываемая или заполняемая нулями сторона
C	требуемая	байт	I-256(I)	символ	одна	-	правая
X	требуемая	байт	I-256	I6-ричные цифры	одна	-	левая
P	4	слово	I-8	десятичные цифры	групповая	-85÷+75	левая
H	2	полуслово	-	десятичные цифры	групповая	-85÷+75	левая
E	4	слово	-	десятичные цифры	групповая	-85÷+75	правая
D	8	двойное слово	-	десятичные цифры	групповая	-85÷+75	правая
F	требуемая	байт	I-I6	десятичные цифры	групповая	-	левая
Z	требуемая	байт	I-I6	десятичные цифры	групповая	-	левая
A	4	слово	I-4	любые выражения	групповая	-	левая

(I) В операторе ОП константа С-типа может иметь модификатор длины равный 65535.

7.6. Определить управляющее слово канала (УСК)

Команда УСК дает удобный способ определения и образования 8-байтного управляющего слова канала, расположенного на границе двойного слова. Транслятор устанавливает байты, пропущенные для настройки счетчика адреса на границу, в нуль.

Машинный формат УСК следующий:

байты	Разряды	Назначение
I	2	3
I	0-7	Код команды
2 - 4	8-3I	Адрес данных
5	32-36	Флажки
	37-39	Должны быть нулями
6	40-47	Устанавливаются в нуль
7 - 8	48-63	Счетчик

Трансляторная команда УСК имеет следующий формат:

Название	Операция	Операнд
Идентификатор или пробел	УСК	Четыре операнда, разделенные запятыми, указывающие содержимое УСК в формате, описанном ниже

Должны быть указаны все четыре операнда. Они записываются слева направо следующим образом:

I. Абсолютное выражение, которое указывает код команды. Значение этого выражения помещается в I-й байт.

2. Выражение, указывающее адрес данных (это значение рассматривается как 3-байтная константа А-типа. Размещается в байтах 2-4).

3. Абсолютное выражение, указывающее флажки для 32-36 разрядов и нули для 37-39 разрядов. Располагается в 5-м байте (6-й байт - нулевой).

4. Абсолютное выражение, указывающее счетчик. Размещается в 7-8 байтах.

Название	Операция	Операнд
	УСК	2,АДРЕСДН,X'48',80

Заметьте, что форма третьего операнда устанавливает разряды 37-39 в нуль, что и требуется. Образец разрядов операнда:

32 - 35	36 - 39
0100	1000

Идентификатору в поле названия оператора УСК, если он есть, присваивается значение адреса крайнего левого байта, атрибут длины - 8.

7.7. Команды связи программ

7.7.I. Общие сведения. Команды связи программ дают возмож-

ность программисту на уровне идентификаторов объединять независимо транслированные программы, которые будут загружаться и выполняться совместно. Связь между программами реализуется при помощи идентификаторов, определенных в одной и используемых в качестве операндов в другой программе. Такие идентификаторы называются идентификаторами связи.

Идентификаторы связи в программе, где они определяются, называются входными точками; в той программе, где они используются в качестве операндов - это внешние идентификаторы.

Прежде, чем использовать внешний идентификатор или входную точку, программист должен сообщить транслятору, какой идентификатор является внешним, а какой - входной точкой. Для этих целей предусмотрены трансляторные команды ВНЕШ и ВХОД. В программе, где идентификатор объединения определяется (т.е. используется в качестве названия), он должен быть также указан транслятору с помощью трансляторной команды ВХОД. Аналогично, программа, которая использует идентификатор, определенный в некоторой другой программе, должна указать его трансляторной командой ВНЕШ.

Наименование программы, заданное в поле названия команды СТАРТ, также считается входной точкой. Для спецификации его как входной точки команда ВХОД не требуется.

7.7.2. СТАРТ -начать трансляцию. Оператор команды СТАРТ используется для наименования программы и для указания начального значения счетчика адреса.

Формат оператора СТАРТ:

Название	Операция	Операнд
Идентификатор или пробел	СТАРТ	Самоопределенный терм или пробел

Если идентификатор в поле названия СТАРТ отсутствует, программа называется неименованной. (В этом случае транслятор присвоит

программе название, состоящее из 8-и пробелов).

Если же программа именована, идентификатору присваивается значение самоопределенного терма (нулевое значение, если в поле операнда - пробел), заданного в поле операнда. Атрибут длины равен I.

Значение самоопределенного терма в поле операнда СТАРТ транслятор использует в качестве начального значения счетчика адреса. Если это значение не кратно 8, счетчик адреса будет установлен на границу следующего двойного слова. Значение самоопределенного терма не должно превышать максимальное значение, допустимое для счетчика адреса. Если поле операнда содержит недопустимую величину или пробел, то счетчик адреса устанавливается в нуль.

Оператор СТАРТ, если он присутствует в программе, должен быть первым оператором программы. Любое неправильное вхождение, СТАРТ не воспринимается.

Если СТАРТ вообще отсутствует, транслятор устанавливает счетчик адреса в нуль, программа будет считаться неименованной и транслироваться с нуля.

Оба оператора, приведенные ниже, могут быть использованы для присваивания программе наименования ПРОГРЗ и указания начального адреса трансляции 2040. Если операнд опущен, транслятор устанавливает начальное значение счетчика адреса программы в нуль.

Название	Операция	Операнд
ПРОГРЗ	СТАРТ	2040

Название	Операция	Операнд
ПРОГРЗ	СТАРТ	X'7F8'

7.7.3. ВХОД - определить идентификатор входной точки. Команда ВХОД указывает идентификатор связи, который определяется в одной программе, но может использоваться как входная точка в любой другой

программе. Функция оператора ВХОД заключается в передаче загрузчику того, что к идентификаторам будут обращаться в другой программе, загруженной вместе с ней.

Формат оператора ВХОД:

Название	Операция	Операнд
Пробел	ВХОД	Перемещаемый идентификатор или группа перемещаемых идентификаторов, разделенных запятыми

Идентификатор в поле названия не воспринимается.

Программа или несколько программ, предназначенных для совместного выполнения, могут содержать не более 100 входных точек.

В следующем примере идентификатор СИНУС специфицирован как входная точка.

Название	Операция	Операнд
ПОДПРОГР	СТАРТ	
	ВХОД	СИНУС
	.	
	.	
	.	
СИНУС	ЗПГ	1,10,АДРЕС

7.7.4. ВНЕШ - определить внешний идентификатор. Команда ВНЕШ указывает идентификатор связи, который используется в данной программе, а определен в другой программе. Функция оператора ВНЕШ состоит в сообщении загрузчику того, что к идентификаторам будут обращаться в этой программе, но определяться они будут в других, загружаемых в данный момент, программах. Они не обязательно должны определяться в одной и той же программе.

Атрибут длины внешнего идентификатора равен 1.

Внешний идентификатор всегда перемещаем.

Формат оператора ВНЕШ:

Название	Операция	Операнд
Пробел	ВНЕШ	Перемещаемый идентификатор или группа перемещаемых идентификаторов, разделенных запятыми

Идентификатор в поле названия не воспринимается.

Идентификатор, стоящий в поле операнда этого оператора, не должен появляться в поле названия оператора данной программы. Однако, он должен быть определен в другой программе и специфицироваться в ней командами СТАРТ или ВХОД.

Например, если операнд представляет собой входную точку другой программы, то обращающаяся к ней программа специфицирует внешний идентификатор следующим образом:

Название	Операция	Операнд
	ВНЕШ	ОПЕРАНД

Программа или несколько программ, предназначенных для совместного выполнения, могут содержать не более 255 внешних идентификаторов.

Правила использования внешних идентификаторов в программе описываются в следующем пункте.

7.7.5. Способы связи программ. Ограничения на способы связи программ. Единственная возможная процедура обращения к внешнему идентификатору состоит в том, чтобы, во-первых, определить его командой ВНЕШ, во-вторых, образовать адресную константу по этому внешнему идентификатору, в-третьих, загрузить константу в общий регистр и, в-четвертых, передать управление по содержимому регистра или использовать его в качестве базового адреса.

Например, для обращения к программе с наименованием СИНУС можно воспользоваться следующими командами:

Название	Операция	Операнд
	•	
	•	
	•	
	ВНЕШН	СИНУС
	•	
	•	
	•	
	З	4,АДСИНУС
	ПВР	I5,4
	•	
	•	
	•	
АДСИНУС	ОК	А(СИНУС)

В данном примере во время трансляции СИНУС получит нулевое значение, а в ячейке АДСИНУС будет записана нулевая константа из 4-х байтов. Компоновщик заменит четыре нулевых байта на фактический адрес, соответствующий идентификатору СИНУС. Этот адрес загружается в четвертый регистр и управление передается программе СИНУС.

Если программисту нужно обратиться на I2 байтов ниже СИНУС, то константа будет задаваться иначе.

Название	Операция	Операнд
	•	
	•	
	•	
	ВНЕШН	СИНУС
	•	
	•	
	•	
	З	4,АДСИНУС
	ПВР	I5,4
	•	
	•	
	•	
АДСИНУС	ОК	А(СИНУС+I2)

Компоновщик добавит I2 к фактическому адресу идентификатора СИНУС и сумму поместит в четырехбайтное поле АДСИНУС.

Другой способ обращения к СИНУС+I2 показан в следующем примере:

Название	Операция	Операнд
	•	
	•	
	ВНЕШ	СИНУС
	•	
	•	
	ИСПЛЗ	СИНУС,4
	З	4,АДСИНУС
	•	
	•	
	ПВ	I5,СИНУС+I2
	•	
	•	
АДСИНУС	ОК	А(СИНУС)
	•	
	•	
	•	

Здесь кроме операторов, используемых в предыдущем примере, еще употребляется оператор ИСПЛЗ. Этот оператор введен с целью возможности разбиения идентификатора СИНУС+I2 на базу и смещение.

Возврат из подпрограммы СИНУС можно осуществить через регистр не обращаясь к идентификаторам связи. Так, если вход в программу выполняется по команде:

Название	Операция	Операнд
	ПВР	I5,4

то возврат может быть осуществлен командой

Название	Операция	Операнд
	ПУР	X'F' ,I5

Порядок загрузки независимо транслированных программ определяет область возможных взаимных обращений этих программ. Программы, содержащие входные точки, должны загружаться раньше программы, по отношению к которым эти входные точки будут уже внешними иден-

тификаторами. Однако на наименование программы это ограничение не распространяется. Программа может обращаться к программам, загруженным после нее, по их наименованиям.

В примере, рассматриваемом ниже, должны совместно выполняться две независимо транслированные программы А и Б.

Если первой загружается программа А, она может обращаться к программе Б только по ее наименованию ПРОГРБ. Программа Б, однако, может обращаться к программе А как по ее наименованию ПРОГРА, так и с помощью входных точек ЦИКЛ и СВЯЗЬ.

При противоположном порядке загрузки программа Б может обращаться к программе А только по ее наименованию, в то время, как программа А может обращаться к программе Б и по наименованию и через входные точки СИНУС и КОСИНУС.

Программа А:

Название	Операция	Операнд
ПРОГРА	СТАРТ	
	ВХОД	ЦИКЛ
	ВХОД	СВЯЗЬ
	ВНЕШН	СИНУС
	ВНЕШН	КОСИНУС
	ВНЕШН	ПРОГРБ
ЦИКЛ	•	
СВЯЗЬ	•	
	•	
	•	
	•	
	•	
	•	
	•	
АСИНУС	ОК	А(СИНУС)
АКОСИНУС	ОК	А(КОСИНУС)
АПРОГРБ	ОК	А(ПРОГРБ)
	•	
	•	
	•	

Программа Б:

Название	Операция	Операнд
ПРОГРБ	СТАРТ	
	ВХОД	СИНУС
	ВХОД	КОСИНУС
	ВНЕШ	ЦИКЛ
	ВНЕШ	СВЯЗЬ
	ВНЕШ	ПРОГРА
	•	
	•	
СИНУС	•	
	•	
КОСИНУС	•	
АЦИКЛ	ОК	А(ЦИКЛ)
АСВЯЗЬ	ОК	А(СВЯЗЬ)
АПРОГРА	ОК	А(ПРОГРА)
	•	
	•	
	•	

Если две независимо транслированные программы должны пользоваться общей областью данных, то эту область также следует транслировать отдельно, а затем загружать первой, чтобы обе программы могли обращаться к ней без ограничений.

Программы, связанные друг с другом во время выполнения, перемещаемые. Поэтому программист не должен употреблять абсолютных выражений для обращений к областям, которые могут перемещаться.

7.8. Команды управления базированием

7.8.1. Общие сведения. Трансляторные команды ИСПЛЗ и ОТМЕН позволяют программистам использовать выражения из неявных адресов в качестве операндов операторов машинных команд. Разбиение неявного адреса на базу и смещение выполняется транслятором.

Для того чтобы использовать идентификаторы в поле операнда операторов машинных команд, программист должен указать транслятору с помощью оператора ИСПЛЗ, какие регистры должны использоваться в качестве базовых, что каждый из них содержит, и загрузить указанные регистры этими значениями.

I0.8.2. ИСПЛЗ - использовать базовый регистр. Оператор ИСПЛЗ указывает один или несколько общих регистров, которые могут использоваться в качестве базовых. Этот оператор указывает также значения базовых адресов, которые транслятор считает загруженными в регистры во время выполнения рабочей программы. Следует отметить, что оператор ИСПЛЗ не загружает указанные регистры. Программист должен следить, чтобы указанные значения базовых адресов были размещены в регистрах.

Формат оператора ИСПЛЗ:

Название	Операция	Операнд
Пробел	ИСПЛЗ	от 2 до I7 выражений в форме Б, Р1, Р2,...,Р16

Идентификатор в поле названия не воспринимается.

Операнд Б может быть абсолютным или перемещаемым выражением. Он может быть отрицательным числом, абсолютное значение которого не превышает 2^{24}. Литералы для задания операнда Б не допускаются.

Операнд Б указывает значение, которое транслятор может

использовать в качестве базового адреса. Остальные операнды должны быть абсолютными выражениями. Операнд P1 указывает общий регистр, который содержит базовый адрес, представленный операндом Б. Операнды P2, P3, P4,... указывают регистры, которые содержат Б+4096, Б+8192, Б+12288,..., соответственно. Значения операндов P1, P2,...,P16 должны быть в пределах от 0 до 15.

Например, оператор

Название	Операция	Операнд
	ИСПЛЗ	., 12, 13

сообщает транслятору, что текущее значение счетчика адреса во время выполнения рабочей программы будет находиться в общем регистре P12, а текущее значение счетчика адреса, увеличенное на 4096, - в общем регистре 13.

Если программист изменяет значение, которое используется в данный момент в базовом регистре, и хочет, чтобы транслятор вычислил смещение, исходя из этого значения, тогда транслятору должно быть сообщено новое значение при помощи другого оператора ИСПЛЗ.

В следующем примере транслятор вначале предполагает, что в 9 регистре находится значение A1.PHA. Затем второй оператор сообщает транслятору, что в регистре 9 находится значение A1.PHA + 1000

Название	Операция	Операнд
	ИСПЛЗ	A1.PHA, 9
	ИСПЛЗ	A1.PHA+1000,9

Оператор ИСПЛЗ может указать в качестве базового регистра регистр 0. Он должен быть операндом PI. В этом случае транслятор предполагает, что регистр 0 содержит нулевое значение, т.е. Б имеет значение, равное нулю. Последующие регистры, указанные в том же операторе, принимают значения 4096, 8192 и т.д. Поэтому транслятор размещает всю последовательность адресов, меньших чем 4096, в поле смещения и ставит 0 в поле базового регистра.

Примечание. Если регистр 0 указан в качестве базового, то программа является неперемещаемой. Однако, программист может сделать эту программу перемещаемой следующим образом:

1. Заменить регистр 0 в операторе ИСПЛЗ.
2. Загрузить регистр перемещаемым выражением.
3. Повторно транслировать программу.

7.8.3. ОТМЕН - отменить базовый регистр.

Оператор ОТМЕН указывает, что ранее доступный базовый регистр не может больше использоваться в качестве базового.

Формат оператора ОТМЕН:

Название	Операция	Операнд
Пробел	ОТМЕН	Абсолютные выражения в форме PI,...PI6

Идентификатор в поле названия не воспринимается.

Выражения в поле операнда указывают общие регистры, ранее упомянутые в операторе ИСПЛЗ, которые теперь недопустимы для базовой адресации.

Например, следующий оператор исключает возможность использования транслятором регистров 7 и II в качестве базовых.

Название	Операция	Операнд
	. ОТМЕН	7,II

Когда используемый базовый адрес изменяется оператором ИСПЛЗ, оператор ОТМЕН можно не употреблять.

Примечание. Если несколько программистов составляют отдельные части программы, которые должны быть транслированы совместно, целесообразно в начале своей программы сбросить все базовые регистры. Это выполняет следующий оператор:

Название	Операция	Операнд
	ОТМЕН	0,I,2,3,4,5,6,7,8,9,I0,II, I2,I3,I4,I5

Оператор ОТМЕН не обязательно ставить в конце исходной программы.

Регистр, ставший недоступным в результате применения команды ОТМЕН, снова может оказаться доступным в результате употребления последующей команды ИСПЛЗ.

Операторы ИСПЛЗ и ОТМЕН могут быть использованы в любом месте программы и так часто, как это необходимо.

Эти операторы дают транслятору информацию, необходимую для построения таблицы базовых регистров. Строки в этой таблице появляются, исчезают или заменяются при обработке транслятором каждой команды ИСПЛЗ или ОТМЕН.

Всякий раз, когда в машинной команде задан неявный адрес, транслятор просматривает эту таблицу и определяет, имеется ли

доступный регистр, в котором содержится подходящий базовый адрес.

Регистр считается доступным для перемещаемого адреса, если в операторе ИСПИЗ было указано, что он загружен перемещаемым значением. Регистр, содержащий абсолютное значение, доступен только для абсолютных адресов.

Базовый адрес будет подходящим, если разность между адресом, который нужно разбить на базу и смещение, и содержимым базового регистра не превышает 4095, а также эта разность не меньше 0.

В следующем примере Р2- базовый регистр, в котором находится значение адреса, равное 1400_{16}. Во время трансляции исходной программы идентификаторам ИМЯ1, ИМЯ2, ИМЯ3, ИМЯ4 были присвоены значения адресов 840_{16}, 1640_{16}, 1400_{16}, 2500_{16}, соответственно, которые транслятор должен разбить на базу и смещение.

1. $840_{16} - 1400_{16} < 0$
2. $1640_{16} - 1400_{16} = 240_{16}$
3. $1400_{16} - 1400_{16} = 0$
4. $2500_{16} - 1400_{16} = 1100_{16}$

В первом случае регистр 2 не может быть базовым, так как разность между адресами меньше 0.

Во втором и третьем случаях регистр 2 может быть базовым, так как разность между значениями адресов не превышает 4095.

В четвертом случае регистр 2 не может быть базовым, так как разность между значениями адресов (1100_{16}) превышает 4095.

В вычислениях, связанных с использованием базовых регистров, транслятор всегда будет пользоваться доступным регистром, дающим наименьшее смещение. Если два регистра содержат одинаковые значения, будет выбран регистр с большим номером.

7.9. Команды управления печатным документом.

7.9.1. НСТР - начать новую страницу.

Оператор НСТР помещает очередную строку печатного документа в начале новой страницы. Он обеспечивает удобный способ отделения подпрограмм друг от друга в печатном документе.

Формат оператора НСТР:

Название	Операция	Операнд
Пробел	НСТР	Пробел

Если оператор НСТР встречается в первой строке страницы, то пропускается вся страница, а если он появился после того, как отпечатана последняя строка страницы, то он не воспринимается.

Если две или более команды НСТР встречаются подряд, для каждой команды НСТР после первой пропускается целая страница, печать продолжается после выполнения последней команды НСТР.

Страница содержит 64 строки.

7.9.2. ПРПСК - пропустить строку в печатном документе.

Оператор ПРПСК используется для выделения одной или нескольких пустых строк в печатном документе программы.

Формат оператора ПРПСК:

Название	Операция	Операнд
Пробел	ПРПСК	Десятичное число или пробел

Десятичное значение в поле операнда оператора ПРПСК используется для указания количества пропускаемых строк в печатном документе.

Если поле операнда отсутствует, оператор ПРПСК вставляет

одну пустую строку. Если же значение операнда превышает количество оставшихся на странице строк, этот оператор равносилен оператору НСТР.

7.10. Команды обработки табличной информации

7.10.1. ОШ – определить шаблон для таблицы.

Операторы ОШ и Т используются для обработки информации, заданной в табличной форме.

Оператор ОШ задает шаблон, по которому транслируется поле операнда оператора Т для каждой таблицы, подлежащей трансляции.

Формат оператора ОШ:

Название	Операция	Операнд
Пробел	ОШ	Один или несколько операндов, разделенные запятыми, в форме, приведенной ниже.

Форма задания каждого операнда подобна заданию операнда оператора ОК. Существенное отличие состоит в том, что из четырех подполей, описывающих константу, отсутствует первое подполе (коэффициент кратности) и четвертое подполе (сама константа).

Допускаются все типы констант, кроме констант А-типа.

Другими словами, операнд оператора ОШ является описанием элемента таблицы, заданного в форме константы соответствующего типа. Для трансляции таблиц со строками переменной длины, количество операндов ОШ должно быть равно максимальному числу элементов строки таблицы, то есть количество операндов зависит от числа столбцов данной таблицы.

Для каждой таблицы создается один оператор ОШ.

7.10.2. Т- определить элемент таблицы по шаблону.

Оператор Т задает строку таблицы, которая будет транслироваться по шаблону, указанному оператором ОШ. Оператор Т создается отдельно для каждой строки таблицы.

Формат оператора Т:

Название	Операция	Операнд
Пробел	Т	Один операнд в форме, описанной ниже

Операнд, стоящий в поле операнда оператора Т, заключается в круглые скобки. Он представляет собой элементы соответствующей строки таблицы, разделенные запятыми.

Оператору Т в обязательном порядке должен предшествовать соответствующий оператор ОШ.

Если какой-нибудь элемент строки таблицы опущен, то запятая, отделяющая его от других элементов данного оператора Т, не опускается. Если же опускается последний элемент, запятая, отделяющая его от предыдущего элемента, тоже опускается.

Элементы строки таблицы могут состоять из любых допустимых символов, за исключением запятой, левой и правой круглых скобок.

Ниже приведены примеры операторов ОШ и Т.

Пример I.

Таблица для трансляции

43						
39	38					
I99	290	29I	200	I98	I96	Максимальная по длине строка

Оператор ОШ для этой таблицы будет выглядеть таким образом:

Название	Операция	Операнд
	ОШ	PL 2, PL 2, PL 2, PL 2, PL 2, PL 2

Максимальное число элементов в строке таблицы - 6, поэтому количество операндов в операторе ОШ - тоже 6.

Операторы Т для каждой строки имеют вид:

Название	Операция	Операнд
	T	(43)
	T	(39, 38)
	T	(I99,290,29I,200,I98,I96)

При обработке этих операторов транслятор, согласно заданному шаблону, сформирует из подполей операторов Т константы и дальнейшая обработка пойдет так, как если бы были заданы константы с помощью оператора ОК:

```
PL 2'43'
PL 2'39'
PL 2'38'
- - -
PL 2'200'
PL 2'I98'
PL 2'I96'
```

Пример 2.

Таблица для трансляции.

9 Актюбинск

8 Алма-Ата

5 Анапа

. . . . . .

I4 Днепропетровск

Эта таблица может быть представлена такой последовательностью операторов:

Название	Операция	Операнд
	СИ	X,C
	T	(9,АКТЮБИНСК)
	T	(8,АЛМА-АТА)
	T	(5,АНАПА)
	.	. . . . .
	T	(E,ДНЕПРОПЕТРОВСК)

Результат трансляции

09 АКТЮБИНСК

08 АЛМА-АТА

05 АНАПА

. . . . . .

0E ДНЕПРОПЕТРОВСК

Строки таблицы можно разделить каким-нибудь символом, например, пробелом. Для этого надо включить в поле операнда СИ описание этого разделяющего символа (первый или последний операнд), а в операторе T на месте, соответствующем заданному шаблону, поместить в качестве подполя операнда сам разделяющий символ.

Название	Операция	Операнд
	СИ	C,PL2, PL2, PL2, PL2, PL2, PL2
	T	(␣,43)
	T	(␣,89,88)
	.	. . . . .

Название	Операция	Операнд
	Т	(␣,I99,290,29I,200,I98,I96)
	.	. . . .
	Т	(␣)

Оператор Т (␣) используется для указания конца последней строки таблицы.

Разделение строк можно произвести еще и таким образом:

Название	Операция	Операнд
	ОШ	PL2,PL2,PL2, PL2, P L 2,C
	Т	(48,,,,,, ␣)
	Т	(39,38,,,,,, ␣)
	.	. . . .
	Т	(I99,290,29I,200,I98,I96, ␣)

7.II. Команды управления транслятором

7.II.I. ИЗМЕН - изменить счетчик адреса.

Оператор ИЗМЕН производит относительное изменение текущего значения счетчика адреса.

Формат оператора ИЗМЕН:

Название	Операция	Операнд
Пробел	ИЗМЕН	Перемещаемое выражение или пробел

Все символы в выражении должны быть ранее определенными. Счетчику адреса присваивается значение этого выражения и трансляция продолжается с полученным значением счетчика адреса.

Например:

Название	Операция	Операнд
	СТАРТ	I000
	.	. . . .
	ИЗМЕН	* + 500

Оператор ИЗМЕН не должен использоваться для указания адреса, который меньше начального адреса, установленного в команде СТАРТ.

Если желательно установить в счетчике адреса значение на один байт больше, чем к этому моменту уже присвоенное значение, надо воспользоваться следующим оператором:

Название	Операция	Операнд
	ИЗМЕН	

Оператор ИЗМЕН может быть, в частности, использован для того, чтобы зарезервировать некоторую область памяти (в тех случаях, когда команда ОП не удобна).

Пропущенные байты транслятором не очищаются.

7.II.2. УНОП - условное "нет операции".

Оператор УНОП позволяет программисту размещать команду в указанных границах полуслова, слова или двойного слова. Если счетчик адреса настроен должным образом, оператор УНОП не воспринимается. Если уже указанная установка счетчика адреса требует его увеличения, то вырабатывается от одной до трех неоперационных команд (ПУР 0,0), каждая из которых занимает два байта памяти.

Формат оператора УНОП:

Название	Операция	Операнд
Пробел	УНОП	Две десятичные цифры в форме: Б, С

Операнд Б указывает, на какой байт слова или двойного слова должен быть установлен счетчик адреса. Операнд Б может принимать значения 0, 2, 4 или 6. Операнд С указывает, задан ли байт Б в слове или двойном слове:

Допустимы следующие пары Б и С:

0,4 - начало слова;

2,4 - середина слова;

0,8 - начало двойного слова;

2,8 - II-е полуслово двойного слова;

4,8 - середина (III-е полуслово) двойного слова;

6,8 - IУ-е полуслово двойного слова.

Задание границ в операторе УНОП:

Двойное слово							
Слово				Слово			
Полуслово		Полуслово		Полуслово		Полуслово	
Байт	Байт	Байт	Байт	Байт	Байт	Байт	Байт
0,4		2,4		0,4		2,4	
0,8		2,8		4,8		6,8	

Здесь показана позиция в двойном слове, которую указывает каждая из этих пар. Пары 0,4 и 2,4 указывают по два адреса в двойном слове, а именно: пара 0,4 указывает адреса I-го и II-го слов двойного слова, а пара 2,4 указывает адреса II-го полуслова I-го слова и II-го полуслова II-го слова, из которых состоит двойное слово.

Предположим, что счетчик адреса установлен на границу двойного слова. Тогда оператор УНОП в следующих последовательностях не производит никакого действия.

Название	Операция	Операнд
	УНОП	0,8
	ПВР	2,I4

Название	Операция	Операнд
	УНОП	0,4
	ПВР	2,I4

Однако последовательность:

Название	Операция	Операнд
	УНОП	6,8
	ПВР	2,I4

вызывает создание трех команд ПУР (передача управления по условию, формат РР), размещая тем самым команду ПВР в IУ-м полуслове двойного слова, как показано ниже:

Название	Операция	Операнд
	ПУР	0,0
	ПУР	0,0
	ПУР	0,0
	ПВР	2,I4

7.II.3. КОНЕЦ – окончить трансляцию. Оператор КОНЕЦ оканчивает трансляцию. Он всегда должен быть последним оператором исходной программы. В операторе КОНЕЦ может быть указано место в данной программе, куда может быть передано управление после загрузки программы.

Формат оператора КОНЕЦ:

Название	Операция	Операнд
Пробел	КОНЕЦ	Перемещаемое выражение или пробел

Выражение в поле операнда обычно указывает на первую машинную команду программы. Например, в данной последовательности операторов идентификатор в поле операнда КОНЕЦ является наименованием оператора первой машинной команды:

Название	Операция	Операнд
	СТАРТ	2000
ОБЛАСТЬ	ОП	50Р
НАЧАЛО	ПВР	2,0
	ИСПЛЗ	№,2
	•	
	•	
	•	
	КОНЕЦ	НАЧАЛО

При обработке оператора КОНЕЦ транслятор выдает карту конца загрузки.

7.I2. Замечания по программированию.

7.I2.I. Основные правила по использованию базовых регистров. Особенностью программирования для АСВТ является использование базовых регистров. Использование базовых регистров позволяет свободно перемещать транслированные программы в оперативном запоминающем устройстве, благодаря чему достигается большая гибкость организации программ, поскольку отдельные ЗУ могут быть распределены в соответствии с требованиями конкретного набора программ. Кроме

того, регистры можно применять для индексирования.

Почти вся работа по определению базовых регистров и вычислению смещений относительно этих баз выполняется транслятором.

Для автоматического обозначения базовых регистров и автоматического вычисления смещений, программист должен сообщить транслятору с помощью трансляторной команды ИСПЛЗ следующую информацию:

1. Какие общие регистры использовать в качестве базовых.
2. Что каждый из них должен содержать во время выполнения рабочей программы.

Загрузку базовых регистров можно выполнить различными методами. Наиболее распространенным способом загрузки базового регистра является использование команды ПВР. Команда записывается так:

ПВР P1, P2

При выполнении этой команды в регистр P1 записывается адрес байта, следующего после этой команды. В регистре P2 указывается адрес перехода, если только P2≠0. Если P2=0, то управление передается следующей команде.

В качестве базового регистра может быть использован любой регистр, за исключением нулевого, 1-го, 13-го, 14-го и 15-го, которые служат для связей между программами одной задачи и для связей программы потребителя с управляющей программой. Обычно используются в качестве базовых регистры со старшими номерами.

Рассмотрим пример 3. В команде СТАРТ указывается, что программа должна быть транслирована с адреса $256_{10} = 100_{16}$. Следующая команда ЗПГ размещается по этому адресу (все адреса рабочей программы, напечатанной слева, представлены в шестнадцатеричной системе). С помощью команды ЗПГ 14, 12, 12(13)

Производится запоминание содержимого регистров, начиная с I4 по I2 включительно в области сохранения, адрес которой содержит I3-й регистр. Команда ПВР I2,0 указывает, что регистр I2 должен быть загружен адресом следующей машинной команды. Затем выполняется следующая команда, так как P2=0.

Пример 8

```
                                        ПРОГ    СТАРТ      256
000I000     90    EC    D00C            НАЧАЛО  ЗПГ        I4,I2,I2(I3)
000I04      05    C0                            ПВР        I2,0
000I06                                          ИСПИЗ      м,I2
000I06      58    20    C02C                    З          2,ДАН
000I0A      5A    20    C034                    СЛ         2,ДЕС
000I0E      8B    20    000I                    СЛА        2,I
000II2      5B    20    C030                    В          2,ДАН+4
000II6      50    20    C038                    ЗП         2,РЕЗУЛЬТ
000IIA      58    60    C03C                    З          6,КОНСТI
000IIE      5A    60    C040                    СЛ         6,КОНСТ2
000I22      4E    60    C044                    ПДС        6,ДВОЙНОЕ
000I26      98    C0    D0I4                    ЗГ         I2,0,20(I3)
000I2A      58    E0    D00C                    З          I4,I2(I3)
000I2E      IB    FF                            ВР         I5,I5
000I30      07    FE                            ПУР        I5,I4
000I32      00    00    00I9      ДАН     ОК      Р'25'
000I36      00    00    000F              ОК      Р'I5'
000I3A      00    00    000A      ДЕС     ОК      Р'I0'
000I3E                                  РЕЗУЛЬТ ОП      Р
000I42      00    00    000C      КОНСТI  ОК      Р'I2'
000I46      00    00    004E      КОНСТ2  ОК      Г'78'
000I4A                                  ДВОЙНОЕ ОП      D
                                                КОНЕЦ   НАЧАЛО
```

Команда ИСПЛЗ указывает, что регистр I2 будет содержать адрес следующей команды и служит базовым регистром для данной программы.

Команда ПВР занимает два байта, а предыдущая команда-четыре байта, поэтому следующая команда 3 имеет адрес I06.

Команда ИСПЛЗ указывает, что именно это значение транслятор поместил в регистр I2.

Рассмотрим теперь, как транслятор обрабатывает команду 3.

Читая слева направо оттранслированный машинный код этой команды, мы видим, что ее код операции 58, регистр, который загружается из ЗУ имеет номер 2, индексация не используется. Базовый регистр имеет $C_{16}=I2_{10}$, а смещение равно 02С. Если базовый регистр содержит I06, а смещение равно 02С, то исполнительный адрес будет I32. Действительно, мы видим, что абсолютный адрес переменной ДАН равен I32, как и должно быть.

Команда СЛ похожа на предыдущую. Используя базовый регистр I2, мы имеем базовый адрес I06 и со смещением 034 получаем исполнительный адрес I3А, который является абсолютным адресом переменной ДЕС.

Команда СЛА несколько отлична от предыдущей. Все команды сдвига имеют формат РП. Поле индекса не используется, но базовый регистр должен быть указан. В этой команде исполнительный адрес никогда не используется для обращения к памяти. Он указывает необходимое число сдвигов и поэтому, изменяя содержимое базового регистра, можно указывать сдвиг на различное число разрядов. Однако, в данной программе эта возможность не используется, и поэтому необходимо указывать базовый регистр 0. Исполнительный адрес тогда равен единице. Относительные адреса переменных, фигурирующих в последующих командах В, ЗП, З, СП, ПДС вычисляются транслятором аналогичным образом. Группа команд ЗГ, З, ВР и ПУР осуществляют завершение

выполнения программы и восстановление содержимого регистров, запомненных в самом начале программы.

Таким образом, мы рассмотрели некоторые основные способы использования базовых регистров.

Как всегда, необходимо точно указывать, что делается во время трансляции и что во время выполнения рабочей программы.

В примере 3 во время трансляции записан (где это необходимо) номер базового регистра и вычислены смещения. Этот номер базового регистра и смещения вошли в состав машинных команд, написанных слева. Для выполнения этой работы транслятор должен знать, какие базовые регистры мы желаем использовать и что мы собираемся поместить в них. Эта информация была бы ему сообщена командой ИСПЛЗ.

Транслятор не может загрузить в базовый регистр содержимое, необходимое для выполнения данной программы. Это может быть сделано во время выполнения самой программы. Для этой цели служит команда ПВР, которая во время выполнения программы записывает в указанный регистр адрес следующей команды, что обеспечивает формирование в дальнейшем правильных исполнительных адресов.

Следует принять во внимание, что существует еще третий момент размещения программы - загрузка программы. С помощью команды СТАРТ мы указали, что первый байт программы должен иметь адрес $256_{10}=100_{16}$. Вся программа составлена именно в этом предположении.

Предположим, что после того, как программа уже транслирована, ее по каким-либо причинам нужно поместить, начиная с адреса 1200_{16}.

С помощью соответствующей управляющей карты мы указываем программе-загрузчику, что первая команда программы должна быть помещена в байт 1200_{16}. Что же нужно изменить в самой программе, чтобы она работала правильно на новом месте? Оказывается, что не требуется никаких изменений.

Когда программа загружена, она начинает работать с выполнения команды ЗПГ. После этого выполняется команда ПВР, которая загрузит базовый регистр адресом следующей команды. Но адрес команды, следующей после ПВР равен I206. Эта величина записывается в регистр I2 и становится базовым адресом программы. Смещения в машинных командах остались, конечно, прежними. При этом исполнительный адрес в команде 3 стал теперь равным I206+02C=I232. При новом начале программы I232 как раз является абсолютным адресом переменной ДАН.

Исполнительный адрес переменной ДЕС оказывается теперь равным I23A. "Адрес" в команде сдвига не изменяется, так как в команде не используется базовый регистр.

В случае большой программы, желательно разбить ее на мелкие куски, что само по себе навязывает использование нескольких базовых регистров. Иначе обстоит дело при наличии больших массивов данных. Часто можно использовать один базовый регистр для программы, другой - для данных.

Пусть программа использует больше чем 4096 байтов для команд и для данных вместе. Дополнительно покажем, как используется команда транслятора ИЗМЕН, продвигающая счетчик адреса (см.пример 4).

Пример 4.

```
                                          СТАРТ    256
000100   90   EC   D000   НАЧАЛО          ЗПГ      I4,I2,I2(I8)
000104   05   C0                          ПВР      I2,0
000106                                    ИСПИЗ    БРЕГ,I2,II
000106   58   B0   C008   БРЕГ            З        II,БАЗА
00010A   47   C0   C00C                   ПУ       I2,НЕРВЫЙ
00010E   00   00   1106   БАЗА            ОК       A(БРЕГ+4096)
00012   58   20   СРРА   ПЕРВЫЙ           З        2,ДАН
```

```
000II6   5A   30   B00A                    СЛ      3,ДЕС
000II8   47   F0   CFFE                    ПУ      I5,ВТОРОЙ
000IIA                                     ИЗМЕН   *+4096
000IIE
00II00   00   00   007B   ДАН    ВТОРОЙ    ОК      Р'I23'
00II04   58   30   CFFA          ВТОРОЙ    З       З,ДАН
00II08   5A   30   B00A                    СЛ      З,ДЕС
00II0C   47   F0   C00C                    ПУ      I5,ПЕРВЫЙ
00II10   00   00   000A   ДЕС              ОК      Р'I0'
                                           КОНЕЦ   НАЧАЛО
```

В этом примере команда ИСПЛЗ задана по другому. Вместо использования звездочки для обозначения адреса следующей команды мы обозначаем эту команду идентификатором БРЕГ и используем значение этого идентификатора. Это производит тот же самый эффект, что и в предыдущем случае, но позволяет обозначать содержимое регистра I2 с помощью идентификатора, что нам потребуется при загрузке регистра II.

Команда ИСПЛЗ БРЕГ, I2, II

сообщает транслятору следующую информацию:

- в качестве базовых регистров можно использовать РI2 и РII;
- РI2 содержит значение идентификатора БРЕГ;
- РII содержит величину на 4096 большую, то есть БРЕГ+4096.

Первый базовый регистр мы загрузили с помощью команды ПВР I2,0. Второй базовый регистр этим способом уже нельзя загрузить, так как мы должны загрузить в него величину на 4096 байтов больше той, что находится в регистре I2.

Для этой цели мы используем адресную константу

БАЗА = БРЕГ + 4096

Здесь следует отметить следующее: эта константа может находиться только в той части области памяти, которая доступна с помощью регистра I2, так как во время выполнения рабочей программы, пока только РI2 содержит базовый адрес. Регистр II, хотя и указан в качестве базового, пока (т.е. к моменту указания базовых регистров) не загружен нужной величиной. Поэтому мы помещаем команду загрузки базового регистра II сразу после ИСПЛЗ. Транслятор при трансляции этой команды выберет в качестве базового РI2.

Адрес константы БАЗА равен IOE.

В качестве базовых можно использовать как РI2, так и РII. Но если бы в качестве базового был выбран РII, то смещение было бы отрицательным.

IOE - II06 = -FF8

Поэтому при трансляции команды

БРЕГ 3 II, БАЗА

в качестве базового будет взят регистр РI2 и смещение будет равно IOE -I06= 008. Исполнительный адрес, таким образом, равен IOE, что, как видно, является абсолютным адресом константы БАЗА.

После константы БАЗА следуют две рабочие команды, а затем управление передается команде в конце программы.

Специально для иллюстрации в данном примере показано, что конец программы отдален от начала больше чем на 4096 байтов. Это было достигнуто с помощью трансляторной команды ИЗМЕН, которая продвигает счетчик адреса в данном случае на 4066.

Число 4066 произвольно и выбрано таким образом, чтобы поместить переменную ДАН в конце области в 4096 байтов, доступную с помощью регистра I2. Следующие команды и данные могут быть выбраны только с помощью регистра II.

Посмотрим теперь, как транслятор обозначает базовые регистры и вычисляет смещения.

Первая команды ПУ имеет исполнительный адрес переменной ПЕРВЫЙ, вычисляемый с помощью базового регистра I2. Если бы бы указан регистр II, то смещение было бы отрицательным.

Команда 3, наименованная идентификатором ПЕРВЫЙ, загружает переменную ДАН. В качестве базового взят регистр I2, а смещение равно РРА.

В команде СЛ указан адрес, смещенный более чем на 4096 байтов от начала программы, поэтому базовый регистр I2 не может быть использован. Мы видим, что указан регистр II и смещение ОСА.

Во второй команде ПУ осуществляется переход к адресу II04, который на 2 меньше, чем содержимое регистра II, поэтому в качестве базового регистра указан РI2 и смещение РРЕ. Транслированная команда ПУ перехода к переменной ПЕРВЫЙ в точности такая, как рассмотренная уже команда, стоящая непосредственно перед константой БАЗА.

Существенно то, что транслятор обозначает базовый регистр таким образом, чтобы обеспечить смещение меньше, чем 4096. Если программа написана так, что имеется два регистра и более, содержимое которых удовлетворяет этому правилу, то транслятор выбирает тот из них, который даст меньшее смещение.

Очень часто требуются раздельные базовые регистры для команд и для данных, даже если все вместе занимают меньше, чем 4096 байтов. Предположим, что имеется в памяти шесть массивов, каждый по 80 символов. Массивы расположены подряд. Первый из 480 байтов имеет символический адрес ДАН. Внутри каждого массива имеется восемь полей по девять символов каждый, обозначенных ПОЛЕ1, ПОЛЕ2 и т.д.. Каждое поле представляет собой упакованное десятичное число. Требуется сложить ПОЛЕ1 и ПОЛЕ2 и поместить в ПОЛЕ3.

Остальные пять областей в этой программе не используются. Эта процедура должна быть проделана для каждого из шести массивов с помощью циклической программы. Вопрос состоит в том, как организовать цикл.

Можно, конечно, изменять смещение в каждой команде, относящейся к массивам, однако, есть лучший способ. Предлагается изменять содержимое базового регистра так, чтобы обходить массивы в нужном порядке. Это означает, что в каждом цикле необходимо добавлять 80 к содержимому в базовом регистре. Но теперь возникает другая проблема: если используется только один базовый регистр, то как можно модифицировать его содержимое и в то же время указывать правильную базу для команд перехода и для обращения к программным константам?

Необходимо использовать два базовых регистра. Причем, второй только для данных, обрабатываемых в цикле (см.пример 5).

Загрузка базовых регистров такая же, только Р8 загружается адресом соответствующей переменной ДАН, который может и не превышать содержимое регистра I2 на 4096 байтов.

Действительно, оказывается, что РI2 содержит I06, а Р8 – I38. Это означает, что первый байт области, обозначенный ДАН, может быть получен добавлением смещения С32 к содержимому РI2 или добавлением смещения ООО к содержимому Р8.

Мы видим, что в нашей программе адресная константа для загрузки Р8 помещена не внутри программы, а в ее конце. Это допустимо поскольку известно, что конец программы отстоит от начала меньше, чем на 4096 байтов.

Предположим, что исходные данные подготавливаются другой программой, а результаты нашей программы используются последующими. Поэтому мы поместим данные с помощью команды ОП, которая

никакой области не резервирует, а только присваивает наименование нашему массиву данных, непосредственно следующему за командой ОП.

Пример 5

```
                                                СТАРТ    256
000100   90   EC   D00C   НАЧАЛО   ЗПГ      I4,I2,I2(I3)
000104   05   C0                   ПВР      I2,0
000106                             ИСПЛЗ    ж,I2
000106   58   80   C026   МЕТКА1   З        8,БАЗА
00010A                             ИСПЛЗ    ДАН,8
00010A   D2   09   80I4   8000МЕТКА2 ШП      ПОЛЕ3,ПОЛЕ1
000110   FA   99   80I4   800A     СЛД      ПОЛЕ3,ПОЛЕ2
000116   5A   80   C02A             СЛ       8,ШАГ
00011A   59   80   C02E             СР       8,ТЕСТ
00011E   47   70   C004                      7,МЕТКА2
000120   98   C0   D0I4             ЗГ       I2,0,20(I3)
000124   58   E0   D00C             З        I4,I2(I3)
000128   IB   FF                   ВР       I5,I5
00012A   07   FE                   ПУР      I5,4
00012C   00   00   0I38   БАЗА     ОК       А(ДАН)
000130   00   00   0050   ШАГ      ОК       Р'80'
000134   00   00   03I8   ТЕСТ     ОК       А(ДАН+480)
00138                              ДАН      ОП       ОР
00138                              ПОЛЕ1    ОП       СЬIO
00142                              ПОЛЕ2    ОП       СЬIO
0014C                              ПОЛЕ3    ОП       СЬIO
00156                              ПОЛЕ4    ОП       СЬIO
```

```
000160          ПОЛЕ5   ОП      СL10
00016A          ПОЛЕ6   ОП      СL10
000174          ПОЛЕ7   ОП      СL10
00017E          ПОЛЕ8   ОП      СL10
000188                  ОП      СL80
000ID8                  ОП      СL80
000228                  ОП      СL80
000278                  ОП      СL80
0002C8                  ОП      СL80
000318          КОНЕЦ   НАЧАЛО
```

Обозначение переменной ДАН означает следующее - только адрес обращения к символу. Никакой действительной константы здесь не находится, поскольку указан 0-й коэффициент кратности.

Таким образом, ДАН и ПОЛЕ1 имеют один и тот же адрес. Это делается для того, чтобы присвоить наименование ДАН для всего массива входной информации объемом в 480 байтов и, кроме того, использовать некоторые наименования для отдельных курсов этого массива.

Можно было бы использовать наименование ДАН для первого куска, ДАН+10 для второго, ДАН+20 для третьего и т.д. Однако, отказ от смысловых наименований нежелателен.

Третий способ заключается в том, чтобы отказаться от наименований ДАН и использовать ПОЛЕ1 всюду, где раньше использовалось наименование ДАН. При этом обозначения также становятся смысловыми.

Команда ПП, помеченная идентификатором МЕТКА2, пересылает ПОЛЕ1 в область ПОЛЕ3.

Читаем соответствующую машинную команду:

Код операции - D2

Код длины - 09

Базовый регистр для I-го операнда - P8

Смещение - 0I4

Базовый регистр для 2-го операнда - P8

Смещение - 000.

Код длины 09 соответствует длине I0 ПОЛЕI. Базовый адрес I38 и смещение 0I4 дают исполнительный адрес I4C, что является абсолютным адресом для ПОЛЕ3. Базовый адрес I38 плюс смещение 000 дают адрес ПОЛЕI.

Команда СЛД выполняет сложение. Эта команда имеет два кода длины. Оба в нашем случае равны 09. Смещение 00A вместе с базовым адресом I38 дают адрес I42 для ПОЛЕ2. Адресация ПОЛЕ3 такая же, как и раньше.

Теперь можно добавить 80 к базовому регистру, связанному с переменной ДАН и повторить процесс. Мы добавляем 80 к базовому регистру P9 и сравниваем результат с адресной константой, чтобы обеспечить проверку окончания цикла. Эта константа должна быть равной 480. Команда ПУ с 7 в качестве маски является командой "Переход, если не равно". Если переход не выполнен, мы должны закончить вычисление, и поэтому следующими идут команды восстановления содержимого общих регистров.

Если бы программа была написана с использованием только одного базового регистра, возникли бы трудности с адресом команды ПУ. Транслятор при вычислении смещения этой команды использовал бы некоторое содержимое базового регистра. Однако это содержимое изменилось бы при модификации и адрес перехода стал бы неверным уже после одного цикла. Поэтому стоит заботиться о том, чтобы этот P8 не использовался нигде, кроме переменной ДАН. Тогда не возникнет никаких нарушений, хотя содержимое базового регистра меняется по отношению к тому, что было указано транслятору. Указанное транс-

лятору значение точно соответствует тому, что необходимо для обработки первого массива. Во время выполнения программы содержимое этого регистра изменяется, но транслятор уже не участвует в работе.

Иногда бывает необходимо обрабатывать не один, а несколько блоков по шесть массивов. В таком случае нужно восстанавливать в Р8 его начальное содержимое. Это выполняется простым повторением команды "Загрузить", помеченной идентификатором МЕТКА1.

Если эту программу нужно переместить по памяти, то изменение только одного начального адреса программы явно недостаточно. Необходимо при загрузке изменять ее адресные константы БАЗА и ТЕСТ.

Для того чтобы программа - загрузчик знала, какие адресные константы подлежат изменению, эти константы автоматически помечаются при трансляции.

Предположим, что после оператора, помеченного ДАР, необходимо поместить еще несколько дополнительных команд. Управление им передается с помощью команды перехода. Пусть внутри этой группы команд имеется команда перехода к некоторым из этих же команд. При этом транслятор выберет в качестве базы регистр 8, дающий меньшее смещение, чем Р12. Но Р8 используется только для адресации массива данных, а для команд нужно использовать Р12. Чтобы указать это транслятору, необходимо перед началом новой группы команд использовать команду транслятора ОТМЕН. Эта команда укажет транслятору, что Р8 не может быть использован в качестве базы. Остается выбрать в качестве такого только Р12, что и сделает транслятор.

При программировании часто приходится использовать стандартные программы и подпрограммы. Они используются для уменьшения времени и труда программирования, когда некоторая уже существующая программа включается в новую программу.

Пусть некоторая подпрограмма первоначально была транслирована для загрузки в ту же область памяти, что и основная программа. Это приводит к необходимости перемещения подпрограммы, что возможно благодаря использованию базовых регистров.

Далее рассмотрим вопрос, какие действия необходимы для того, чтобы позволить одной программе нормально функционировать после перемещения?

Из рассмотренных примеров 3 и 4 было видно, что адреса, в которых используется базовый регистр, не зависят от перемещения программы.

Но не все адреса программы формируются с помощью базового регистра. Это адреса в адресных константах А-типа. Решение заключается в следующем: к каждой адресной константе добавить разность между новым адресом загрузки и адресом, присвоенным программе при трансляции. Это число называется константой перемещения. Таким образом, необходим список всех адресов констант, которые должны быть модифицированы. Имея такой список, загрузчик может прибавить к каждой адресной константе константу перемещения. Этот список загрузчик получает из карт списка перемещаемых идентификаторов, которые ему представляются транслятором.

7.12.2. Внутренняя связь с подпрограммами.

Команда "Переход с возвратом" (ПВ, ПВР) позволяет эффективно перейти на подпрограмму. Например:

```
ПЕРЕХОД     ПВ     РI, АДПЕР
```

АДПЕР - адрес перехода. 32 правых разряда ССП, которые в данный момент представляют собой адрес обрабатываемой команды, запоминаются в общем регистре РI, что дает возможность для продолжения выполнения следующей команды после возврата из подпрограммы.

В команде ПВ подразумевается, что АДПЕР представляет собой адрес, состоящий из базы, сложенной со смещением.

ПВР позволяет перейти на подпрограмму, которая не может быть адресована вышеприведенным способом. Последовательность для перехода на подпрограмму принимает вид:

```
            З       Р2,=А(АДПЕР)
ПЕРЕХ       ПВР     РI,Р2
```

Общий регистр, представленный Р2, содержит адрес перехода, а РI дает связывающую информацию для возврата.

Так как вычисление адреса предшествует запоминанию связующей информации, то в вышеприведенном примере может быть использован РI для выполнения обеих функций.

Последовательность запишется в виде:

```
            З       РI,А=(АДПЕР)
ПЕРЕХ       ПВР     РI,РI
```

При использовании этой формы необходимо следить, чтобы была обеспечена база для программы.

```
АДПЕР       ПВР     Р4,0
            ИСПЛЗ   ж,Р4
```

В чередующейся последовательности ПВ может использовать индексацию:

```
            З       Р4, = А(АДПЕР)
ПЕРЕХ       ПВ      РI, 0(Р4)
```

Такая запись не является неправильной, но правильнее выполнять это путем загрузки адресной константы и использования ПВР. Полезным оказывается использование ПВ с индексным регистром в случае многократного входа (или входа ниже ячейки, помеченной именем подпрограммы). Когда управление удерживается в пределах величин возрастания к желаемой входной точке, то последовательность принимает вид:

```
            З       РI, =А (АДПЕР)
            ПВ      Р2, КОНСТ(РI)
КОНСТ       РАВНО   N
```

Место для константы может быть зарезервировано там, где оно не является результатом вычисления. Можно использовать следующий вариант вышеприведенной последовательности перехода:

```
         З      РI,  = А(АДПЕР)
         ЗА     Р2,N
         СПР    РI,Р2
ПЕРЕХ    ПВР    Р2,РI
```

Обычно эта форма употребляется тогда, когда должно быть использовано действительное возрастание (N) константы.

Объединение аргументов. Большинство подпрограмм содержат в себе передачу вводимых значений или вывод результатов, или, наконец, какой-то способ переработки информации, иногда они действительно являются аргументами, в другой раз они являются адресами аргументов. Объединение аргументов между главной программой и подпрограммами может быть выполнено многими способами.

Одним подходом предполагается определение места внутри самой подпрограммы для параметров. Этот метод требует, чтобы данные, которые будут обрабатываться подпрограммой, были перемещены в это место до вызова.

Последовательность, использующая этот метод, представлена на примере ниже. Выходные значения должны быть переданы симметричными операциями, следующими за возвратом.

Пример 6.

```
         З      Р2,=А(АДПЕР)
         З      РI,ПАРАМI
         ЗП     РI,0(Р2)
         З      РI,ПАРАМ2
         ЗП     РI,4(Р2)
         З      РI,ПАРАМ3
```

```
            ЗП      PI,8(P2)
ПЕРЕХ       ПВ      PI,I2(P2)
```

Такая пересылка расточительна по времени выполнения и объему памяти, если ее повторять много раз. С другой стороны, можно повысить относительно эффективность, когда большинство параметров являются статистическими и только один из них изменяется от вызова к вызову.

Данные с плавающей запятой должны адресоваться регистрами с плавающей запятой. Эти регистры не могут быть использованы, как общие регистры для получения адреса или запоминания возврата.

В определенных случаях подпрограмма может вызываться для резервирования места в памяти, используя одну ячейку для ввода и вывода.

Программа, которая предполагает использование массива без самой подпрограммы в числе других подпрограмм, могла бы исключить полную расточительность памяти посредством передвижения аргументов в массиве общих параметров раньше, чем произойдет вызов каждой подпрограммы. Здесь появляются новые предположения, а именно: подпрограмма должна быть снабжена адресом массива параметров. Если не принимать во внимание наличие регистров, то адресную константу для массива параметров можно будет загрузить в регистр раньше перехода. Вызов подпрограммы тогда должен принять вид:

```
            З       P2,=A(ПАРАМ)
            З       PI,=A(АДПЕР)
            ПВР     P3,PI
```

Подпрограмма должна использовать P2 для сборки параметров. Первые операции должны принять вид, показанный в примере 7.

Пример 7.

```
АДПЕР       ИСПЛЗ       *,PI
            ЗПГ         PI,PI6,СОХРОБЛ
```

```
                ЗПН     Р2,ПАРАМ1
                ЗПН     Р4,ПАРАМ2
                З       Р3,0(Р2)
                ЗП      Р3,ПАР
ССХРОБЛ         ОП      I6Р
ПАРАМ1          ОП      ID
ПАРАМ2          ОП      ID
ПАР             ОП      IPD
```

Чтобы недостаток регистров не сделал такую процедуру невозможной, адресная константа должна стоять в основной программе за командов перехода для восстановления подпрограммы. Такой вызов должен быть:

```
                З       2,=А(АДПЕР)
                УНОП    2,4
                ПВР     Р1,Р2
                ОК      А(ПАРАМ)
```

Подпрограмма должна подобрать параметр, загружая его адрес в общие регистры для использования как базы. Таким образом, первые операторы программы должны принять вид, показанный в примере 8.

Пример 8.

```
АДПЕР           ИСПЛЗ   АДПЕР2
                ЗПГ     Р1,Р16,ССХРОБЛ
                ЗПВ     Р2,ПАРАМ1
                З       Р3,0(Р1)
                ЗВ      Р2,0(Р3)
                ЗПВ     Р2,АРГ
ССХРОБЛ         ОП      I6Р
ПАРАМ1          ОП      ID
АРГ             ОП      ID
```

Данные, которые будут использоваться подпрограммой, должны быть размещены в массиве параметров перед вызовом, а выводимые результаты должны выбираться из массива параметров после возврата.

Таким образом, там, где серия вызовов использует вывод из подпрограммы, как ввод в другую, можно сэкономить время и память, должным образом размещая параметры в ячейках.

Тем способом, которым адресная константа помещается в регистр или в строку программы, следующую за переходом, можно вводить действительные данные. Этот способ является наиболее выгодным, когда имеется несколько вводимых и выводимых значений. Например, он оказывается очень выгодным при многократной работе.

Вызов подпрограммы, использующий пересылку регистра с плавающей запятой, запишется в виде:

```
          ЗВ      ПЗР4,ВВОД
          З       Р2,=А(АДПЕР)
          ПВР     РI,Р2
          .
          .
          .
ВВОД      ОК      D'22,26'
```

Вероятно, что наилучшим методом является использование четырех регистров для различных задач объединения:

```
          ЗА      РI,СОХРОБЛ
          ЗА      РI3,ПАРАМ
          З       PI5,A(АДПЕР)
          ПВР     PI4,PI5
          .
          .
          .
СОХРОБЛ   ОП      I6Р
ПАРАМ     ОК      А(АРГI)
          ОК      А(АРГ2)
          .
          .
          .
          ОК      А(АРГ N )
```

Это разрешает находиться "накапливаемому массиву" в вызывающей программе, а не в вызываемой, что более соответствует сегодняшним воззрениям на управление программой.

7.12.3. Временные массивы подпрограмм. Подпрограмма должна иметь резерв для сохранения информации о программе.

Регистры, используемые для вычисления в подпрограмме, должны быть запомнены, если по некоторой договоренности они не оставлены свободными для программ.

Сохраняющие массивы должны соответствовать длине регистров, которые будут запоминаться (заметим, что регистры с плавающей запятой имеют двойную длину).

Если ячейки, выделенные для запоминания регистров программы, находятся в начале подпрограммы, то входная точка должна быть установлена или небольшим приращением смещения, или использованием индивидуальных адресных констант.

При использовании подпрограмм должны быть приняты меры для сохранения базы вызывающей программы, чтобы обеспечить надлежащий возврат.

Вызовы более низкого уровня, происходящие внутри подпрограммы, имеют здесь большую важность и поэтому желательно принять меры предосторожности для предовращения разрушения базы, так как от нее зависит возврат.

В частности, подпрограммы, которые используют внутри другие подпрограммы, несмотря на первоначальные определения программы, должны обеспечить место для сохранения базовых регистров.

Возврат. Когда выполняется переход с возвратом, регистр, обозначенный как PI, содержит 32 самых правых разрядов ССП, которые определяют адрес обрабатываемой инструкции. Этим обеспечивается

связующая информация для возврата из подпрограммы.

Если регистр связи заполнен в регистре 2 на вводе, он перезагружается адресом вызывающей программы для возврата. Если переход на подпрограмму является последней инструкцией в вызывающей последовательности, то возврат будет простым переходом вида:

```
            З       Р2,СВЯЗЬ
ВОЗВР       ПВР     РI5,Р2
СВЯЗЬ       ОП      Р
```

Если параметры или константы появляются после перехода, то достаточно добавить смещение, равное их длине, к адресу обрабатываемой инструкции.

В этом случае приведенная выше последовательность примет вид:

```
            З       Р2,СВЯЗЬ
ВОВВР       ПУ      I5,4(2)
СВЯЗЬ       ОП      Р
```

7.I3. Некоторые замечания по составлению программ

Операционная система АСВТ дает программисту больше возможности в проектировании и подготовке программы.

Разделение программ является одной из важных фаз программирования. Наличие в операционной системе АСВТ обрабатывающих программ (компоновщик, загрузчик, и др.) в некоторой степени облегчает эту часть программирующих действий.

Рассмотрим на примере, какие трудности могут возникнуть при разделении программы на подпрограммы.

Предположим, что программа (см.рис.I) вместе с данными и таблицами занимает 4095 байтов и для адресации этой программы используется только один базовый регистр.

База	программа (4095 байтов)
База	данные
База	таблицы

Рис.I

Очевидно, что этот базовый регистр не может перекрыть дополнительные модификации программы. Если нет в наличии свободного общего регистра, который можно было бы выделить дополнительно в качестве базового, то программист был бы вынужден создать так называемую программу модификации,! дав ей временную базу.

После модификации программа будет выглядеть следующим образом (см.рис.2).

Если программа модифицируется несколько раз, то вышеописанную процедуру приходится повторять столько раз, сколько раз подвергается изменениям программа. При этом возникает большое число изменений связи таких, как регистры и ликвидация нарушений (переполнение, адресация и т.п.).

База	программа модификации (небольшая подпрограмма)
База	данные
База	подпрограмма
	таблицы

Рис.2

Эти изменения можно произвести путем "вставки", то есть программист должен сначала загрузить регистр (предполагая, что таковой имеется), который будет использоваться как база "вставка", конечно, адресные константы для загрузки должны достигаться посредством настоящего базового регистра. Переход к "вставке" осуществляется с помощью нового регистра, как базового. Если нет свобод-

ного регистра, или, если адресная константа находится вне области, адресуемой при помощи настоящего базового регистра, то "вставку" сделать невозможно.

Этого можно избежать, если программист будет следовать простой методике проектирования программы. Сначала программист должен разделить программу на главную часть (вызывающая программа) и ряд вызываемых подпрограмм.

Главная часть должна содержать общие константы и все "вставки" программы. Главная часть должна перекрываться одним базовым регистром, а каждая подпрограмма – другим. Связь (возврат) требует загрузки адресной константы, прежде чем произойдет переход на подпрограмму при помощи команды ПВР.

На рис.3 показан этот тип разбиения.

Основная программа Общие константы Малые подгруппы	База
Вспомогательная область	
Подпрограмма I	База
Подпрограмма 2	База
Подпрограмма 3	База

Рис.3

Такая связь допускается для любого типа модификации, то есть добавляется новая подпрограмма и обеспечивается связь с нею. В этом случае массив "вставка" всегда перекрывается базовым регистром главной части программы.

Контролируя базовый регистр подпрограммы, программист может отметить, какая подпрограмма была вызвана последней.

Распределение общих регистров в операционной системе рас-

смотрено в части II. Программист должен соблюдать при составлении программы установленные там соглашения относительно регистров.

Распределение общих регистров для конкретной программы зависит от размера, сложности и количества подпрограмм в ней. Одной из основных функций распределения регистров является установление нужного количества базовых регистров, загрузка их определенными величинами и сообщение об этом транслятору посредством трансляторной команды ИСПЛЗ.

В большинстве случаев разбиение программы на подпрограммы и выделение для каждой подпрограммы отдельного базового регистра, рекомендованное в предыдущем пункте, является пригодным, если не стремиться к большой экономии памяти и регистров.

Различие между базовым и индексным регистрами обычно является искусственным. В одной команде некоторый регистр используется как база, например, в то время как в другой он может использоваться в качестве индексного регистра (с абсолютным значением, не превосходящим 4095).

Простейшим способом выделения базовых регистров из числа всех общих регистров является указание таковых командой ИСПЛЗ и загрузка их перемещаемым индентификатором. Индексный регистр обычно содержит абсолютную величину, которая растет в цикле.

Очень важно при распределении регистров заранее договориться о регистрах возврата для связей с подпрограммами. Тут могут быть полезными следующие рекомендации:

I. Для многоуровневых связей хранение и перезапоминание регистров неизбежно. При этом нужно пытаться использовать смежные регистры для связей так, чтобы запоминание и восстановление их было возможно осуществить при помощи команд ЗНГ и ЗГ.

2. Согласованная договорность о связи является особенно важной, когда одна и та же подпрограмма вызывается несколькими программами.

3. Необходимо избегать постоянного распределения регистров для данной программы, то есть сделать временным, насколько это возможно, использование регистров на протяжении всей программы.

Следует отметить еще несколько особенностей по использованию общих регистров с малыми номерами.

Общий регистр 0 не может использоваться как базовый или индексный. Но он может быть успешно использован для других целей, например, для связи при пересылке значений или адресов, при умножении или делении с фиксированной запятой как счетчик или накопитель (т.е. сумматор).

Регистры 1 и 2 также не должны использоваться для базирования и индексации, так как они могут быть изменены действиями отдельных команд (например, ПКА и РДО). Обычно это не выводит их из работы, так как они дают большие возможности и многосторонность этим командам. Кроме того, регистры 1 и 2 могут быть использованы для тех же целей; что и регистр 0.

Вообще, программист может исключить противоречия в использовании общих регистров, должным образом планируя и устанавливая соглашение на их использование.

Разделение программы и распределение регистров - две главные функции планирования.

Разделение программы является третьей очень важной функцией при подготовке программы, так как определение данных непосредственно сказывается на применении системы команд. Следовательно, программист должен точно определить, с каким типом данных и какой системой команд он будет оперировать.

Следующая таблица показывает, какие данные обычно связываются и с какими командами или системной команд.

Типы данных	Команды
F	логические (слово) и с фикс.зап.(слово)
H	с фиксированной запятой (полуслова)
E	с плавающей запятой (низкая точность)
P	с фиксированной запятой (упакованный)
	с фиксированной запятой (зонированный)
C	логические (символ)
X	логические (символ)
A	адресные
D	с плавающей запятой (высокая точность)

Конечно, данные могут использоваться различными способами. Однако, если типы данных сочетаются с соответствующими командами, то программист будет значительно экономить время.

Оглавление

В разработке учебно-методического пособия принимали участие:

проф. Кузьмин И.В.
доц. Канарский В.Ф.
доц. Филиппенко И.Г.
к.т.н. Салыга В.И.
к.т.н. Бекиров Ф.А.
инж. Денищук Н.Ф.
Гринченко А.Д.
Трофимова Е.Ю.
Коршиков В.И.
Кундай А.В.
Шумеев В.В.
Кузьменко В.М.
Пикова И.И.

Ответственный за выпуск Денищук Н.Ф.

БЦ № 20412 от 16. XI. 1978 г., Заказ № 666
Тираж 200 экз.
Объем 20 усл.п.л.
Цена 2р.50 коп.

Ротапринт ХИРЭ, г. Харьков,
пр. Ленина, 14.