**Методичні вказівки**

**до виконання самостійної роботи з дисципліни**

**«Мікропроцесорна техніка»**

**для студентів спеціальностей**

**153 – «Мікро- та наносистемна техніка»**

**та 171 – «Електроніка»**

Міністерство освіти і науки України
Вінницький національний технічний університет

# Методичні вказівки
# до виконання самостійної роботи з дисципліни
# «Мікропроцесорна техніка»
# для студентів спеціальностей
# 153 – «Мікро- та наносистемна техніка»
# та 171 – «Електроніка»

Вінниця
ВНТУ
2019

Рекомендовано до друку Методичною Радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 3 від 22.11.2018 р.)

Рецензенти:
**С. Т. Барась,** кандидат технічних наук, професор
**О. С. Городецька,** кандидат технічних наук, доцент

Методичні вказівки до виконання самостійної роботи з дисципліни «Мікропроцесорна техніка» для студентів спеціальностей 153 – «Мікро- та наносистемна техніка» та 171 – «Електроніка» / Уклад. Б. П. Книш. – Вінниця : ВНТУ, 2019. – 28 с.

У методичних вказівках наведено основні теоретичні дані до виконання самостійної роботи з дисципліни «Мікропроцесорна техніка» та рекомендовану літературу. Методичні вказівки розроблено відповідно до навчальної програми дисципліни «Мікропроцесорна техніка»

---

*Навчальне видання*

**Методичні вказівки до виконання самостійної роботи з дисципліни ««Мікропроцесорна техніка» для студентів спеціальностей 153 – «Мікро- та наносистемна техніка» та 171 – «Електроніка»**

Укладач *Богдан Петрович Книш*

Рукопис оформив *Б. Книш*

Редактор *Т. Старічек*

Оригінал-макет виготовив *О. Ткачук*

Підписано до друку 31.01.2019 р. Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman. Друк різографічний. Ум. друк. арк. 1,68.
Наклад 40 (1-й запуск 1–21) пр. Зам. № 2019-015.

Видавець та виготовлювач
Вінницький національний технічний університет,
інформаційний редакційно-видавничий центр.
ВНТУ, ГНК, к. 114. Хмельницьке шосе, 95, м. Вінниця, 21021. Тел. (0432) 65-18-06.
**press.vntu.edu.ua**; *E-mail*: kivc.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

**ЗМІСТ**

# ТЕМА 1. ПОНЯТТЯ СИСТЕМНОЇ ШИНИ. ШИНИ АДРЕСИ, ШИНА КОМАНД, ШИНА ДАНИХ

**Мета:** отримання знань щодо системних шин, їх особливостей та різновидів, зокрема шин адрес, команд, даних.

В системну магістраль (системну шину) мікропроцесорної системи входить три основні інформаційні шини: адреси, даних і управління.

Шина даних – це основна шина, заради якої і створюється вся система. Кількість її розрядів (ліній зв'язку) визначає швидкість і ефективність інформаційного обміну, а також максимально можливу кількість команд.

Шина даних завжди двонаправлена, оскільки припускає передачу інформації в обох напрямах. Тип вихідного каскаду, що часто зустрічається, для ліній цієї шини – вихід з трьома станами.

Звичайно шина даних має 8, 16, 32 або 64 розряди. Зрозуміло, що за один цикл обміну по 64-розрядній шині може передаватися 8 байтів інформації, а по 8-розрядній – тільки один байт. Розрядність шини даних визначає і розрядність всієї магістралі. Наприклад, коли говорять про 32-розрядну системну магістраль, мається на увазі, що вона має 32-розрядну шину даних.

Шина адреси – друга за важливістю шина, яка визначає максимально можливу складність мікропроцесорної системи, тобто допустимий обсяг пам'яті і, отже, максимально можливий розмір програми і максимально можливий обсяг даних, що запам'ятовуються. Кількість адрес, забезпечуваних шиною адреси, визначається як $2N$, де $N$ – кількість розрядів. Наприклад, 16-розрядна шина адреси забезпечує 65 536 адрес. Розрядність шини адреси зазвичай кратна 4 і може досягати 32 і навіть 64. Шина адреси може бути однонаправленою (коли магістраллю завжди управляє тільки процесор) або двонаправленою (коли процесор може тимчасово передавати управління магістраллю іншому пристрою, наприклад контролеру). Найбільш часто використовуються типи вихідних каскадів з трьома станами або звичайні ТТЛ (з двома станами).

Як в шині даних, так і в шині адреси може використовуватися позитивна логіка або негативна логіка. При позитивній логіці високий рівень напруги відповідає логічній одиниці на відповідній лінії зв'язку, низький – логічному нулю. При негативній логіці – навпаки. В більшості випадків рівні сигналів на шинах – ТТЛ.

*Під час підготовки до цієї теми необхідно розглянути:*

– шина адреси;

– шина команд;

– шина даних.

*Рекомендована література*

1. Мікропроцесори та мікроконтролери : електронний підручник / [Т. О. Терещенко, Ю. С. Петергеря, В. Я. Жуйков та ін.]. – К. : НТУ «КПІ», 2009.

2. Гусев В. Г. Электроника и микропроцессорная техника : учебник для вузов / Гусев В. Г. – СПб. : БХВ-Петербург, 2005. – 312 с.

## ТЕМА 2. ПЕРЕВЕДЕННЯ ЧИСЕЛ З ОДНІЄЇ СИСТЕМИ В ІНШУ

**Мета:** отримання знань щодо правил та принципів переведення чисел в десятковій, двійковій, вісімковій та шістнадцятковій системах числення.

Система числення – символічний метод запису чисел, подання чисел за допомогою письмових знаків. Система числення дає подання безлічі чисел (цілих та/або дійсних), дає кожному числу унікальне подання (або, принаймні, стандартний вигляд), відображає алгебраїчну і арифметичну структуру чисел. До систем числення висуваються певні вимоги, серед яких найбільш важливими є вимоги однозначного кодування невід'ємних чисел 0, 1,... з деякої їх скінченної множини – діапазону $P$ за скінченне число кроків і можливості виконання відносно чисел арифметичних і логічних операцій. Від вдалого чи невдалого вибору системи числення залежить ефективність розв’язання зазначених задач і її використання на практиці.

Системи числення поділяються на: позиційні, непозиційні, змішані.

У позиційних системах числення один і той самий числовий знак (цифра) в запису числа має різні значення залежно від того місця (розряду), де він розташований. Під позиційною системою числення зазвичай розуміється $b$-ва система числення, яка визначається цілим числом $b>1$, так званою основою системи числення. У позиційних системах чим більше основа системи, тим менша кількість розрядів (тобто цифр) потрібна для записування числа.

У непозиційних системах числення величина, яку позначає цифра, не залежить від положення в числі. При цьому система може накладати обмеження на положення цифр, наприклад, щоб вони були розташовані в порядку спадання, зменшення. Канонічним прикладом майже непозиційної системи числення є римська, в якій як цифри використовуються латинські літери.

У нумізматиці особливо велику вагу мають десяткова система, дванадцяткова (дуодецимальна), четвертна та шісткова системи. У інформаційних технологіях застосовуються двійкова, десяткова, вісімкова, та шістнадцяткова системи.

*Під час підготовки до цієї теми необхідно розглянути:*

– арифметичні операції з двійковими числами;

– арифметичні операції з вісімковими числами;

– арифметичні операції з шістнадцятковими числами.

*Рекомендована література*

1. Гусев В. Г. Электроника и микропроцессорная техника : учебник для вузов / Гусев В. Г. – СПб. : БХВ-Петербург, 2005. – 312 с.

2. Жаворонков М. А. Микропроцессорная техника : учебник для вузов / М. А. Жаворонков, А. В. Кузин. – М. : Академия, 2008. – 304 с.

## ТЕМА 3. ПРИНЦИП СЕГМЕНТНОЇ ОРГАНІЗАЦІЇ ПАМ'ЯТІ

**Мета:** отримання знань щодо організації даних в пам'яті, зокрема сегментної організації пам'яті.

Організація пам'яті передбачає, що віртуальна пам'ять – це безперервний масив з наскрізною нумерацією слів, що не завжди є оптимальним. Зазвичай програма складається з декількох частин – кодової, інформаційної та стекової. Оскільки заздалегідь невідомі довжини цих складових, то зручно, щоб під час програмування кожна з них мала власну нумерацію слів, що починається з нуля. Для цього організовують систему сегментованої пам'яті, виділяючи у віртуальному просторі незалежні лінійні простори змінної довжини, що називаються сегментами. Кожен сегмент є окремою логічною одиницею інформації, що містить сукупність даних або програмний код і розташована в адресному просторі користувача. В кожному сегменті встановлюється своя власна нумерація

слів, починаючи з нуля. Віртуальна пам’ять також розбивається на сегменти, з незалежною адресацією слів усередині сегмента. Кожній складовій програми виділяється сегмент пам’яті. Логічна адреса визначається номером сегмента і адресою всередині сегмента. Для перетворення логічної адреси у фізичну використовується спеціальна сегментна таблиця.

Недоліком такого підходу є те, що неоднаковий розмір сегментів приводить до неефективного використання основної пам’яті. Так, якщо основна пам’ять заповнена, то у разі заміщення одного з сегментів потрібно видалити такий, розмір якого дорівнює або більший розміру нового. У разі багатократного повтору подібних дій в основній пам’яті залишається багато вільних ділянок, недостатніх за розміром для завантаження повного сегмента. Вирішенням проблеми слугує сегментно-сторінкова організація пам’яті. В ній розмір сегмента вибирається не довільно, задається кратним розміру сторінки.

Сегмент може містити те або інше, але обов’язково ціле число сторінок, навіть якщо одна із сторінок заповнена частково. Виникає певна ієрархія в організації доступу до даних, що складається з трьох ступенів: сегмент > сторінка > слово. Цій структурі відповідає ієрархія таблиць, які слугують для перекладу логічних адрес у фізичні. В сегментній таблиці програми перераховуються всі сегменти цієї програми з вказанням початкових адрес сторінкових таблиць, які відносяться до кожного сегмента. Кількість сторінкових таблиць дорівнює числу сегментів і будь-яка з них визначає розташування кожної із сторінок сегмента в пам’яті, які можуть розташовуватися не послідовно – частина сторінок може знаходитися в основній пам’яті, інші – у зовнішній пам’яті

Для отримання фізичної адреси необхідний доступ до сегментної та однієї із сторінкових таблиць, тому перетворення адреси може займати багато часу.

*Під час підготовки до цієї теми необхідно розглянути:*

– технологія сегментації пам’яті;

– сегментна адресація в реальному режимі.

*Рекомендована література*

1. Тарарака В. Д. Архітектура комп'ютерних систем : навчальний посібник / Тарарака В. Д. – Житомир : ЖДТУ, 2018. – 383 с.

2. Жаворонков М. А. Микропроцессорная техника : учебник для вузов / М. А. Жаворонков, А. В. Кузин. – М. : Академия, 2008. – 304 с.

3. Гилмор Ч. Введение в микропроцессорную технику / Гилмор Ч. – М Мир, 2002. – 314 с.

## ТЕМА 4. ПРИНЦИП ПОБУДОВИ МІКРОПРОЦЕСОРНОЇ СИСТЕМИ

**Мета:** отримання знань щодо мікропроцесорних систем (МПС), їх видів та принципів побудови.

В основу побудови МПС покладено три принципи: магістральності, модульності, мікропрограмного керування.

Принцип магістральності визначає характер зв’язків між функціональними блоками МПС – усі блоки з’єднуються з єдиною системною шиною.

Принцип модульності полягає в тому, що система будується на основі обмеженої кількості типів конструктивно і функціонально завершених модулів. Кожний модуль МПС системи має вхід керування третім (високоімпедансним) станом. Цей вхід називається CS (Chip Select) – вибір кристала або OE (Output Enable) – дозвіл виходу.

Принцип мікропрограмного керування полягає у можливості здійснення елементарних операцій – мікрокоманд (зсуву, пересилання інформації, логічних операцій). Певною комбінацією мікрокоманд можна створити набір команд, який максимально відповідатиме призначенню системи, тобто створити технологічну мову. У секційних процесорах набір мікрокоманд можна змінити, використовуючи інші мікросхеми пам’яті мікрокоманд.

*Під час підготовки до цієї теми необхідно розглянути:*

– принципи побудови МПС;

– типова структура МПС та призначення її функціональних модулів;

– призначення входу керування третім станом.

*Рекомендована література*

1. Гилмор Ч. Введение в микропроцессорную технику / Гилмор Ч. – М. : Мир, 2002. – 314 с.

2. Бродин В. Б. Системы на микроконтроллерах и БИС программируемой логики / В. Б. Бродин, А. В. Калинин. – М. : ЭКОМ, 2002. – 398 с.

## ТЕМА 5. ФОН-НЕЙМАНІВСЬКА ТА ГАРВАРДСЬКА АРХІТЕКТУРИ

**Мета:** отримання знань щодо поняття архітектура, її видів, а саме фон Нейманівська та Гарвардська архітектури.

Головними особливостями фон Нейманівської архітектури є:

1. Інформація в ЕОМ ділиться на команди і дані;
2. Команди вказують ЕОМ, які дії і над якими операндами виконувати;
3. Послідовність команд, за якою виконується алгоритм вирішення задачі, називають програмою;
4. Весь набір виконуваних ЕОМ команд називають системою команд ЕОМ;
5. Дані – це числа і закодовані символи, які використовуються командами як операнди. Одні команди для інших також можуть бути операндами;
6. Команди і дані представлено двійковим кодом;
7. Немає відмінностей в представленні команд і даних;
8. Команди і дані зберігаються в одній пам’яті;
9. Команди і дані зберігаються в пам’яті за відповідними адресами;
10. Пам’ять має довільну адресацію, тобто в кожному такті можна звернутися до довільної її комірки;
11. Пам’ять є лінійною. Її адресу кодують двійковим кодом.

Гарвардська архітектура передбачає розділення пам’яті на пам’ять даних і пам’ять команд. Тим самим розділяються шини передачі керівної і оброблюваної інформації. При цьому підвищується продуктивність комп’ютера за рахунок суміщення в часі пересилання та обробки даних і команд.

*Під час підготовки до цієї теми необхідно розглянути:*

– особливості Фон-Нейманівської архітектури;

– особливості Гарвардської архітектури.

*Рекомендована література*

1. Бродин В. Б. Системы на микроконтроллерах и БИС программируемой логики / В. Б. Бродин, А. В. Калинин. – М. : ЭКОМ, 2002. – 398 с.

2. Калашников О. А. Ассемблер – это просто. Учимся программировать / Калашников О. А. – СПб. : БХВ-Петербург, 2012. – 336 с.

## ТЕМА 6. ПЕРЕВЕДЕННЯ ЛОГІЧНИХ АДРЕС У ФІЗИЧНІ ТА НАВПАКИ

**Мета:** отримання знань щодо фізичних і логічних адрес та правил й принципів переведення однієї в іншу й навпаки.

При використанні обчислювальних машин характерною є ситуація, коли розміщення всієї програми в основній пам’яті неможливо через її великий розмір. У цьому, проте, і немає принципової необхідності, оскільки в кожен момент часу «увага» машини концентрується на певних порівняно невеликих ділянках програми. Таким чином, в основній пам’яті досить зберігати тільки частини програм, що використовуються в цей період, а решта частин може розташовуватися на зовнішніх запам’ятовувальних пристроях. Складність подібного підходу в тому, що процеси звернення до основної пам’яті і зовнішнього запам’ятовуючого пристрою істотно розрізняються, і це ускладнює завдання програміста. Виходом з такої ситуації була поява в 1959 році ідеї віртуалізації пам’яті, під якою розуміється метод автоматичного управління ієрархічною пам’яттю, при якому програмістові здається, що він має справу з єдиною пам’яттю великої ємкості і високої швидкодії. Цю пам’ять називають віртуальною пам’яттю. За своєю суттю віртуалізація пам’яті є способом апаратної і програмної реалізації концепції ієрархічної пам’яті.

У рамках ідеї віртуалізації основної пам’яті розглядається як лінійний простір N адрес, названий фізичним простором пам’яті. Для завдань, де потрібно більш ніж N комірок, надається значно більший простір адрес (зазвичай такий, що дорівнює загальній ємності всіх видів пам’яті), названий віртуальним простором, у загальному випадку не обов’язково лінійний. Адреси віртуального простору називають логічними, а адреси фізичного простору – фізичними.

Програма пишеться у логічних адресах, але оскільки для її виконання потрібно, щоб оброблювані команди і дані знаходилися в основній пам’яті, потрібно, щоб кожній логічній адресі відповідала фізична. Таким чином, у процесі обчислень необхідно, перш за все, переписати з зовнішнього запам’ятовувального пристрою в основну пам’ять ту частину інформації, на яку вказує логічна адреса (відобразити віртуальний простір на фізичний), після чого перетворити логічну адресу у фізичну.

Серед систем віртуальної пам’яті можна виділити два класи: системи з фіксованим розміром блоків (сторінкова організація) і системи із змінним розміром блоків (сегментна організація).

*Під час підготовки до цієї теми необхідно розглянути:*

– логічні адреси;

– фізичні адреси;

– переведення логічних адрес у фізичні;

– переведення фізичних адрес у логічні.

*Рекомендована література*

1. Калашников О. А. Ассемблер – это просто. Учимся программировать / Калашников О. А. – СПб. : БХВ-Петербург, 2012. – 336 с.

2. Тарарака В. Д. Архітектура комп'ютерних систем : навчальний посібник / Тарарака В. Д. – Житомир : ЖДТУ, 2018. – 383 с.

3. Цилькер Б. Я. Организация ЭВМ и систем : учебник для вузов / Б. Я. Цилькер, С. А. Орлов. – СПб. : Питер, 2004. – 668 с.

## ТЕМА 7. РЕГІСТР ПРАПОРЦІВ ТА ЙОГО ВИКОРИСТАННЯ

**Мета:** отримання знань щодо поняття прапорець, видів прапорців та регістрів, які відповідають за їх використання.

Важливий регістр, що використовується при виконанні більшості команд – регістр прапорців. В цьому регістрі кожен біт встановлюється в 1 чи 0 при виконанні певних умов. Всі біти, розташовані в старшому слові регістра прапорців, мають відношення до управління захищеним режимом.

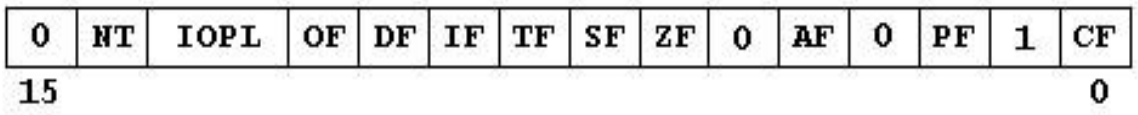


Рисунок 7.1 – Регістр прапорців

Молодші 16 бітів цього регістра називаються:

- CF (Carry Flag) – біт переносу. Встановлюється в 1, якщо при попередній операції відбувся перенос з старшого біта або якщо потрібно позичити біт (при відніманні), інакше встановлюється в 0.
- PF (Parity Flag) – біт парності. Встановлюється в 1, якщо молодший байт результату попередньої команди містить парне число бітів, які дорівнюють одиниці, 1 і в 0, якщо число бітів непарне.
- AF (Adjust Flag) – біт напівпереносу або додаткового переносу. Встановлюється в 1, якщо в результаті попередньої операції відбувся перенос чи позичання з третього біта в четвертий. Цей прапорець автоматично використовується командами двійково-десяткової корекції.
- ZF (Zero Flag) – біт нуля. Встановлюється в 1, якщо результат попередньої команди рівний нулю.
- SF (Sign Flag) – біт знаку. Він завжди рівний старшому біту результату.
- TF (Trap Flag) – біт пастки. Він був передбачений для роботи відлагоджувачів. Встановлення його в 1 призводить до того, що після виконання кожної команди програми управління передається відлагоджувачу.
- IF (Interrupt enable Flag) – біт переривань. Встановлення його в 1 призводить до того, що процесор перестає обробляти зовнішні переривання. Зазвичай його встановлюють на короткий проміжок часу при виконання критичних ділянок коду.
- DF (Direction Flag) – біт напряму. Цей прапорець контролює поведінку команд обробки рядків – коли він встановлений в 1, вони обробляються в бік зменшення адрес, а коли в 0 – то навпаки.
- OF (Overflow Flag) – біт переповнення. Цей прапорець встановлюється в 1, якщо результат попередньої арифметичної операції над числом зі знаком виходить за допустимі для нього межі.

*Під час підготовки до цієї теми необхідно розглянути:*

– біт переносу;

– біт парності;

– біт напівпереносу або додаткового переносу;
– біт нуля;
– біт знаку;
– біт пастки;
– біт переривань;
– біт напряму;
– біт переповнення.

*Рекомендована література:*

1. Регістр прапорців [Сайт]. Режим доступу: https://helpiks.org/5-63427.html (дата звернення 15.01.2019). – Назва з екрану.

## ТЕМА 8. ЗАДАЧІ, ЩО ВИРІШУЮТЬСЯ СИГНАЛЬНИМИ МІКРОПРОЦЕСОРАМИ

**Мета:** отримання знань щодо сигнальних процесорів та задач, які вони вирішують.

Для цифрової обробки сигналів використовуються сигнальні мікропроцесори. До їх особливостей відносяться малорозрядна (40 і менше розрядів) обробка чисел з плаваючою точкою, переважне використання чисел з фіксованою точкою розрядності 32 і менше, а також орієнтація на нескладну обробку великих масивів даних. Відмінною особливістю задач цифрової обробки сигналів є потоковий характер обробки великих обсягів даних у реальному режимі часу, що потребує високої продуктивності процесора і забезпечення можливості інтенсивного обміну з зовнішніми пристроями. Відповідність цим вимогам досягається нині завдяки специфічній архітектурі сигнальних процесорів і проблемно-орієнтованій системі команд.

Сигнальні процесори мають високий ступінь спеціалізації. У них широко використовуються методи скорочення тривалості командного такту (характерні і для універсальних RISC-процесорів), такі як конвеєризація на рівні окремих мікрокоманд і команд, розміщення операндів більшості команд в регістрах, використання прихованих (тіньових) регістрів для збереження стану обчислень при перемиканні контексту, поділ пам’яті команд і даних. У той же час для сигнальних процесорів характерною є наявність апаратного помножувача, що дозволяє

виконувати множення двох чисел за один такт. В універсальних процесорах множення зазвичай реалізується протягом кількох тактів як послідовність операцій зсуву та додавання. Іншою особливістю сигнальних процесорів є віднесення до системи команд таких операцій як множення з накопиченням, реверсування порядку розташування бітів адреси, операції над бітами. У сигнальних процесорах реалізується апаратна підтримка програмних циклів, кільцевих буферів, обробки переривань.

*Під час підготовки до цієї теми необхідно розглянути:*

– особливості та застосування сигнальних мікропроцесорів;

– процес конвеєризації.

*Рекомендована література*

1. Рожков П. П. Мікропроцесорна техніка : конспект лекцій / П. П. Рожков, С. Е. Рожкова. – Харків : ХНАМГ, 2008. – 116 с.

## ТЕМА 9. ПРИНЦИПИ ПОБУДОВИ НЕЙРОМЕРЕЖ

**Мета:** отримання знань щодо поняття нейромережа та принципи їх побудови.

Згідно з загальноприйнятим уявленням найбільш загальними принципами, характерними для сучасних нейромереж, є: коннекціонізм, нелінійність активаційної функції, локальність і паралелізм обчислень, навчання замість програмування, оптимальність навчальних алгоритмів.

Принцип коннекціонізму означає, що кожен нейрон нейромережі, як правило, пов’заний з усіма нейронами попереднього шару обробки даних.

Нелінійність вихідної функції активації принципова. Щоб відобразити суть біологічних нейронних систем, визначення штучного нейрона відбувається таким чином. Він отримує вхідні сигнали (вихідні дані або вихідні сигнали інших нейронів нейронної мережі) через кілька вхідних каналів. Кожен вхідний сигнал проходить через з’єднання, що має певну інтенсивність (або вагу); ця вага відповідає синаптичній активності біологічного нейрона. З кожним нейроном пов'язано певне порогове значення. Обчислюється зважена сума входів, з неї віднімається порогове значення і отримуємо величину активації нейрона.

Сигнал активації перетворюється за допомогою функції активації (або передаточної функції) і як результат отримуємо вихідний сигнал нейрона.

Нелінійність руйнує лінійну суперпозицію і призводить до значного розширення можливостей нейромереж.

Масовий паралелізм нейрообчислень, необхідний для ефективної обробки образів, забезпечується локальністю обробки інформації в нейромережах. Кожен нейрон реагує лише на локальну інформацію, що надходить до нього в певний момент від пов'язаних з ним таких самих нейронів, без апеляції до загального плану обчислень, звичайної для універсальних ЕОМ. Таким чином, нейромережеві алгоритми локальні, і нейрони здатні функціонувати паралельно.

Штучна нейромережа, як і природна біологічна, може навчатися вирішенню завдань: вона містить внутрішні адаптивні параметри нейронів і своєї структури та, змінюючи їх, може змінювати свою поведінку. Місце програмування займає навчання, тренування нейронної мережі: для вирішення завдання не потрібно програмувати алгоритм – потрібно взяти універсальний нейромережевий інструмент, створити і навчити нейромережу.

Привабливою рисою нейрокомп'ютингу є єдиний принцип навчання нейромереж – мінімізація емпіричної помилки. Функція помилки, що оцінює цю конфігурацію мережі, задається ззовні – залежно від того, яку мету переслідує навчання. Але далі мережа починає поступово модифікувати свою конфігурацію – стан всіх своїх синаптичних ваг – таким чином, щоб мінімізувати цю помилку. Насамкінець, в процесі навчання мережа все краще справляється з покладеним на неї завданням.

Базовою ідеєю всіх алгоритмів навчання є врахування локального градієнта в просторі конфігурацій для вибору траєкторії якнайшвидшого спуску по функції помилки. Функція помилки, однак, може мати безліч локальних мінімумів, що є субоптимальними рішеннями.

*Під час підготовки до цієї теми необхідно розглянути:*

– коннекціонізм;
– нелінійність активаційної функції;
– локальність і паралелізм обчислень;
– навчання замість програмування;
– оптимальність навчальних алгоритмів.

*Рекомендована література*

1. Новотарський М. А. Штучні нейронні мережі: обчислення / М. А. Новотарський, Б. Б. Нестеренко. – К. : Ін-т математики НАН України, 2004. – 408 с.

## ТЕМА 10. ЗАДАЧІ, ЩО ВИРІШУЮТЬСЯ НЕЙРООБЧИСЛЮВАЧАМИ

**Мета:** отримання знань щодо поняття нейрообчислювач та задач, які вони вирішують.

Нейрообчислювач – пристрій переробки інформації з урахуванням засад роботи природних нейронних систем. Ці принципи було формалізовано, що дозволило говорити про теорії штучних нейронних мереж. Проблематика нейрокомп'ютерів у побудові реальних фізичних пристроїв, що дозволить моделювати штучні нейронні мережі на звичайному комп'ютері, а також змінити принципи роботи комп'ютера.

Основною проблемою сучасних нейрообчислювачів є їхня доступність. Вони випускаються у складі спеціалізованих пристроїв та досить дорогі. На розробку витрачається чимало часу, протягом якого програмні реалізації на останніх комп'ютерах виявляються на порядок менш продуктивними, що робить використання нейропроцесорів нерентабельним. Проте аналогічна проблема раніше виникала і для звичайних комп'ютерів, тож потрібно очікувати, що нейрообчислювачі стануть доступнішими.

*Під час підготовки до цієї теми необхідно розглянути:*

– нейрообчислювач;

– задачі, що вирішуються нейрообчислювачами;

– основна проблема нейрообчислювачів.

*Рекомендована література*

1. Мікропроцесори та мікроконтролери : електронний підручник / [Т. О. Терещенко, Ю. С. Петергеря, В. Я. Жуйков та ін.]. – К. : НТУ «КПІ», 2009.

2. Гусев В. Г. Электроника и микропроцессорная техника : учебник для вузов / Гусев В. Г. – СПб. : БХВ-Петербург, 2005. – 312 с.

3. Новотарський М. А. Штучні нейронні мережі: обчислення / М. А. Новотарський, Б. Б. Нестеренко. – К. : Ін-т математики НАН України, 2004. – 408 с..

## ТЕМА 11. ПРИНЦИПИ АДРЕСАЦІЇ ПОРТІВ ВВЕДЕННЯ-ВИВЕДЕННЯ

**Мета:** отримання знань щодо портів комп'ютера, зокрема портів введення-виведення та принципів їх адресації.

Кожен мікропроцесор має деяку кількість ліній введення-виведення, що об'єднані в багаторозрядні (частіше 8-розрядні) паралельні порти введення-виведення. У пам'яті мікропроцесора кожному порту введення-виведення відповідає своя адреса регістра даних. Звертання до регістра даних порту введення-виведення відбувається тими самими командами, що і звертання до пам'яті даних. Крім того, у багатьох мікропроцесорів окремі розряди портів можуть бути опитані або встановлені командами бітового процесора.

Залежно від реалізованих функцій розрізняють такі типи паралельних портів:

- однонапрямлені порти, призначені тільки для введення або тільки для виведення інформації;
- двонапрямлені порти, напрямок передачі яких (введення або виведення) визначається в процесі ініціалізації мікропроцесора;
- порти з альтернативною функцією (мультиплексовані порти);
- порти з програмно керованою схемотехнікою вхідного-вихідного буфера.

Порти виконують роль пристроїв часового узгодження функціонування мікропроцесора і об'єкта управління, які, в загальному випадку, працюють асинхронно. Розрізняють три типи алгоритмів обміну інформацією між мікропроцесором і зовнішнім пристроєм через паралельні порти введення-виведення:

- режим простого програмного введення-виведення;
- режим введення-виведення зі стробом;
- режим введення-виведення з повним набором сигналів підтвердження обміну.

*Під час підготовки до цієї теми необхідно розглянути:*

– типи паралельних портів;

– типи алгоритмів обміну інформацією між мікропроцесором і зовнішнім пристроєм.

*Рекомендована література*

1. Бродин В. Б. Системы на микроконтроллерах и БИС программируемой логики / В. Б. Бродин, А. В. Калинин. – М. : ЭКОМ, 2002. – 400 с.

2. Рожков П. П. Мікропроцесорна техніка : конспект лекцій / П. П. Рожков, С. Е. Рожкова – Харків : ХНАМГ, 2008. – 116 с.

## ТЕМА 12. ОРГАНІЗАЦІЯ ЦИКЛІВ

**Мета:** отримання знань щодо поняття цикл та принципів його організації.

Циклічними називаються обчислювальні процеси, в яких неодноразово виконуються одні й ті ж дії, але з різними даними. Тіло циклу складається з операторів, що повторюються у програмі. Для організації циклу необхідно задати початкове значення змінної, яка буде змінюватися у циклі, її кінцевого значення та крок її зміни. Треба контролювати значення цієї змінної для перевірки умови виходу з циклу. Умовою може бути: перевищення параметром циклу кінцевого значення, виконання заданого числа повторень, досягнення заданої точності обчислення.

Цикли бувають арифметичні та ітеративні. В арифметичних циклах число повторень визначається на основі зміни параметра циклу; в ітеративних циклах – цикл повторюється доти, доки не буде виконана умова виходу з циклу.

*Під час підготовки до цієї теми необхідно розглянути:*

– поняття циклу;

– арифметичні цикли;

– ітеративні цикли.

*Рекомендована література*

1. Гусев В. Г. Электроника и микропроцессорная техника : учебник для вузов / Гусев В. Г. – СПб. : БХВ-Петербург, 2005. – 312 с.

2. Рожков П. П. Конспект лекцій з дисципліни «Мікропроцесорна техніка» [Сайт]. Режим доступу: http://eprints.kname.edu.ua/5522/1/MPRKN_1.pdf (дата звернення 01.11.2018). – Назва з екрану.

3. Opticstoday – Каталог статей. Научные статьи и публикации [Сайт]. Режим доступу: http://opticstoday.com/katalog-statej/stati-na-ukrainskom/mikroprocesori (дата звернення 01.11.2018). – Назва з екрану.

## ТЕМА 13. ЦИКЛИ З ПЕРЕДУМОВОЮ

**Мета:** отримання знань щодо поняття цикл з передумовою та принципів його організації.

Цикл з передумовою – це цикл, у якому тіло циклу виконується тільки у разі виконання умови, заданої перед тілом циклу. Якщо умова не виконується, то робота циклу припиняється і керування передається оператору, наступному за оператором циклу.

У більшості процедурних мов програмування здійснюється за допомогою інструкції while, звідси його друга назва — while-цикл.

Мовою Паскаль цикл з передумовою має такий вигляд:

```
while <умова> do
begin
  <тіло циклу>
end;
```

Мовою Сі:

```
while(<умова>)
{
  <тіло циклу>
}
```

*Під час підготовки до цієї теми необхідно розглянути:*

– поняття циклу з передумовою;

– цикл з передумовою мовою Паскаль;

– цикл з передумовою мовою Сі.

*Рекомендована література*

1. Гусев В. Г. Электроника и микропроцессорная техника : учебник для вузов / Гусев В. Г. – СПб. : БХВ-Петербург, 2005. – 312 с.

2. Цилькер Б. Я. Организация ЭВМ и систем : учебник для вузов / Б. Я. Цилькер, С. А. Орлов. – СПб. : Питер, 2004. – 668 с.

## ТЕМА 14. ЦИКЛИ З ПІСЛЯУМОВОЮ

**Мета:** отримання знань щодо поняття цикл з післяумовою та принципів його організації.

Цикл з післяумовою – це цикл, у якому тіло циклу виконується доти, доки умова, задана після тіла циклу, не стане правильною. Якщо умова стає правильною, то робота циклу припиняється й управління передається оператору, наступному за оператором циклу.

У мові Паскаль такий цикл здійснює інструкція repeat … until; у Сі — do … while.

Мовою Паскаль цикл з післяумовою має такий вигляд:

```
repeat
   <тіло циклу>
until <умова>
```

Мовою Сі:

```
do
{
   <тіло циклу>
}
while(<умова>)
```

*Під час підготовки до цієї теми необхідно розглянути:*

– поняття циклу з післяумовою;

– цикл з післяумовою мовою Паскаль;

– цикл з післяумовою мовою Сі.

*Рекомендована література*

1. Гусев В. Г. Электроника и микропроцессорная техника : учебник для вузов / Гусев В. Г.. – СПб. : БХВ-Петербург, 2005. – 312 с.

2. Цилькер Б. Я. Организация ЭВМ и систем : учебник для вузов / Б. Я. Цилькер, С. А. Орлов. – СПб. : Питер, 2004. – 668 с.

## ТЕМА 15. ОРГАНІЗАЦІЯ ПЕРЕРИВАНЬ

**Мета:** отримання знань щодо поняття переривання та правил і принципів організації переривань.

Під час роботи комп'ютера можуть виникати події у зовнішніх пристроях або у програмі, що виконується в цей момент. Сигнал про ці події направляється до процесора. Він призупиняє роботу команди, що виконується в цей момент та передає управління обробнику переривань. Обробник переривань – це спеціальна процедура, яка починає свою роботу за сигналом процесора і вирішує проблему, що виникла. Після обробки переривання обробник передає управління назад призупиненій програмі і та продовжує своє виконання. В ПК основні обробники переривань знаходяться у BIOS. Під час завантаження ПК ОС замінює їх своїми драйверами пристроїв.

Обробники переривань бувають високопріоритетні (ВОП) та низькопріоритетні (НОП). Обробку переривання починає високопріоритетний обробник ВОП, який швидко обслуговує переривання, збирає необхідну інформацію і підключає низькопріоритетний обробник переривання НОП. НОП закінчує обробку переривання і передає управління призупиненій програмі.

*Під час підготовки до цієї теми необхідно розглянути:*

– переривання;

– обробник переривань.

*Рекомендована література*

1. Бродин В. Б. Системы на микроконтроллерах и БИС программируемой логики / В. Б. Бродин, А. В. Калинин. – М. : ЭКОМ, 2002. – 400 с.

## ТЕМА 16. ДИРЕКТИВИ ВИЗНАЧЕННЯ СЕГМЕНТІВ

**Мета:** отримання знань щодо поняття «директива» та основних директив визначення сегментів.

Сегмент даних призначений для визначення констант, робочих полів і областей для введення-виведення. Згідно з існуючими директивами в асемблері дозволено визначення даних різної довжини: наприклад, директива DB визначає байт, а директива DW визначає слово.

Елемент даних може містити безпосереднє значення або константу, визначену як символьний рядок або як числове значення. Іншим способом визначення константи є безпосереднє значення, тобто вказане прямо в команді, наприклад: MOV AL,20H. У цьому випадку шістнадцяткове число 20 стає частиною машинного об’єктного коду. Безпосереднє значення обмежене одним байтом або одним словом, але таке використання є більш ефективним, аніж застосування константи.

Асемблер забезпечує два способи визначення даних: по-перше, через задання довжини даних і, по-друге, за їх значенням.

Розглянемо основний формат визначення даних:

[Ім’я] Dn вираз

Ім’я елемента даних не є обов’язковим, але якщо в програмі є посилання на деякий елемент, те це робиться за допомогою імені. Для визначення сегментів даних є такі директиви:

- DB (байт);
- DW (слово – 2 байти);
- DD (подвійне слово – 4 байти);
- DQ (8 байтів);
- DT (10 байтів).

*Під час підготовки до цієї теми необхідно розглянути:*

– формат визначення даних;

– директиви для визначення сегментів даних.

*Рекомендована література*

1. Гилмор Ч. Введение в микропроцессорную технику / Гилмор Ч. – М. : Мир, 2002. – 314 с.

## Тестові завдання

1. Які є основні інформаційні шини?
(!) адреси, даних, управління;
(?) системи, даних, управління;
(?) системи, адреси, даних.

2. Якою є шина даних?
(?) однонаправлена;
(!) двонаправлена;
(?) тринаправлена.

3. Кількість адрес, які забезпечуються шиною адреси, визначається як:
(?) $N$;
(!) $2N$;
(?) $2^N$.

4. Якою є шина адрес?
(?) однонаправлена;
(?) двонаправлена;
(!) однонаправлена або двонаправлена.

5. При негативній логіці високий рівень напруги відповідає:
(?) логічній одиниці;
(!) логічному нулю;
(?) логічній одиниці або логічному нулю.

6. Системи числення бувають:
(?) позиційні;
(?) непозиційні;
(!) позиційні, непозиційні, змішані.

7. У позиційних системах числення величина, яку позначає цифра:
(!) залежить від положення в числі;
(?) не залежить від положення в числі;
(?) може залежати, а може не залежати від положення в числі.

8. У непозиційних системах числення величина, яку позначає цифра:
(?) залежить від положення в числі;
(!) не залежить від положення в числі;
(?) може залежати, а може не залежати від положення в числі.

9. В основу побудови МПС покладено:
(?) магістральність;
(?) модульність;
(!) магістральність, модульність, мікропрограмне керування.

10. Головною особливістю фон Нейманівської архітектури є:
(!) команди і дані зберігаються в одній пам'яті;
(?) розділення пам'яті на пам'ять даних і пам'ять команд;
(?) розділення пам'яті на пам'ять даних, пам'ять команд і пам'ять адрес.

11. Головною особливістю Гарвардської архітектури є:
(?) команди і дані зберігаються в одній пам'яті;
(!) розділення пам'яті на пам'ять даних і пам'ять команд;
(?) розділення пам'яті на пам'ять даних, пам'ять команд і пам'ять адрес.

12. CF – це:
(!) біт переносу;
(?) біт парності;
(?) біт напівпереносу або додаткового переносу.

13. PF – це:
(?) біт переносу;
(!) біт парності;
(?) біт напівпереносу або додаткового переносу.

14. AF – це:
(?) біт переносу;
(?) біт парності;
(!) біт напівпереносу або додаткового переносу.

15. ZF – це:
(!) біт нуля;
(?) біт знака;

(?) біт пастки.

16. SF – це:
(?) біт нуля;
(!) біт знака;
(?) біт пастки.

17. TF – це:
(?) біт нуля;
(?) біт знака;
(!) біт пастки.

18. IF – це:
(!) біт переривань;
(?) біт напряму;
(?) біт переповнення.

19. DF – це:
(?) біт переривань;
(!) біт напряму;
(?) біт переповнення.

20. OF – це:
(?) біт переривань;
(?) біт напряму;
(!) біт переповнення.

21. Для цифрової обробки сигналів використовуються:
(?) цифрові мікропроцесори;
(!) сигнальні мікропроцесори;
(?) центральні мікропроцесори.

22. В чому полягає принцип коннекціонізму?
(!) кожен нейрон нейромережі пов’язаний з усіма нейронами попереднього шару обробки даних;
(?) кожен нейрон нейромережі пов’язаний з усіма нейронами всіх шарів обробки даних;

(?) кожен нейрон нейромережі пов’язаний з деякими нейронами попереднього шару обробки даних.

23. Рисою нейрокомп’ютингу є:
(?) максимізація емпіричної помилки;
(!) мінімізація емпіричної помилки;
(?) усунення помилок.

24. Типи паралельних портів:
(?) однонапрямлені порти та двонапрямлені порти;
(?) однонапрямлені порти двонапрямлені порти та мультиплексовані порти;
(!) однонапрямлені порти, двонапрямлені порти, мультиплексовані порти та порти з програмно керованою схемотехнікою вхідного-вихідного буфера.

25. Типи алгоритмів обміну інформацією між мікропроцесором і зовнішнім пристроєм:
(?) режим простого програмного введення-виведення та режим введення-виведення зі стробом;
(?) режим введення-виведення зі стробом та режим введення-виведення з повним набором сигналів підтвердження обміну;
(!) режим простого програмного введення-виведення, режим введення-виведення зі стробом та режим введення-виведення з повним набором сигналів підтвердження обміну.

26. Циклічними називаються обчислювальні процеси, в яких неодноразово виконуються:
(!) одні й ті самі дії, але з різними даними;
(?) одні й ті самі дії, але з однаковими даними;
(?) різні дії, але з різними даними.

27. Цикли є:
(?) арифметичні;
(?) ітеративні;
(!) арифметичні та ітеративні.

28. While-цикл – це:
(!) цикл з передумовою;
(?) цикл з післяумовою;
(?) цикл з передумовою та цикл з післяумовою.

29. Обробники переривань є:
(?) високопріорітетні;
(?) низькопріорітетні;
(!) високопріорітетні та низькопріорітетні.

30. Асемблер забезпечує визначення даних:
(?) через задання довжини даних;
(?) за значенням даних;
(!) через задання довжини даних і за їх значенням.