Міністерство освіти і науки України
Національний університет «Запорізька політехніка»

С.К. Корнієнко

«КОМП'ЮТЕРНА СХЕМОТЕХНІКА ТА АРХІТЕКТУРА КОМП'ЮТЕРА»

конспект лекцій

для студентів спеціальності 122 «Комп'ютерні науки»

Запоріжжя, 2019

Корнієнко С.К. «Комп’ютерна схемотехніка та архітектура комп’ютера»: Конспект лекцій для студентів спеціальності 122 «Комп’ютерні науки» / С.К. Корнієнко. – Запоріжжя: НУ «Запорізька політехніка», 2019. – 90 с.

Рецензенти: Степаненко О.О., доцент, к.т.н.
Онищенко В.Ф., доцент, к.ф.-м.н.

Затверджено
на засіданні кафедри
"Програмні засоби"

Протокол № 1 від 25.08.2019 р.

ЗМІСТ

СЛОВНИК СКОРОЧЕНЬ

BIOS (Basic Input/Output System) - базова система вводу/виводу;

DRAM (Dynamic RAM) – динамічна оперативна пам'ять;

FIFO (First In, First Out) – «першим ввійшов - першим вийшов», один із способів організації ОЗП з послідовним доступом;

HDD (Hard Disk Drive) – зовнішній накопичувач інформації на жорсткому диску (вінчестер);

INT (Interrupt) – переривання, вектор переривання;

LIFO (Last In, First Out) – тип оперативної пам'яті з принципом роботи: «останній ввійшов – перший вийшов»;

PC (Program Counter) – лічильник команд;

PSW (Processor Status Word) – слово стану процесора, код у внутрішньому регістрі стану процесора;

RAM (Random Access Memory) – оперативна пам'ять, ОЗП;

ROM (Read-Only Memory) – постійна пам'ять, ПЗП;

SP (Stack Pointer) – покажчик стека;

USB (Universal Serial Bus**)** – стандартний послідовний інтерфейс з високою швидкістю передач;

АЛП – арифметико-логічний пристрій;

АЦП – аналого-цифровий перетворювач%

ЗП – запам’ятовуючий пристрій;

МП – мікропроцесор;

МПС – мікропроцесорна система;

ОЗП – оперативний запам’ятовуючий пристрій;

ОС – операційна система;

РЗП – регістри загального призначення;

ЦАП – цифро-аналоговий перетворювач.

1 ЛОГІЧНІ ЕЛЕМЕНТИ ТА ВУЗЛИ КОМП'ЮТЕРА

1.1 Логічні елементи та їх класифікація

Логічний елемент – це електронний пристрій, що реалізує одну з логічних операцій. В цих пристроях інформація, що обробляється, кодується у вигляді двійкових чисел за допомогою сигналів високого та низького рівнів.

Логічні елементи являють собою електронні пристрої, у яких оброблювана інформація закодована у вигляді двійкових чисел, відображуваних напругою (сигналом).

Термін «логічні» прийшов в електроніку з алгебри логіки, яка оперує зі змінними величинами та їхніми функціями, що можуть приймати тільки два значення: «істинно» чи «хибно».

Для позначення істинності чи хибності висловлень використовують відповідно символи 1 чи 0. Кожна логічна перемінна може приймати тільки одне значення: 1 чи 0. Ці двійкові змінні та функції від них називаються логічними змінними й логічними функціями. Пристрої, що реалізують логічні функції, називаються логічними або цифровими пристроями.

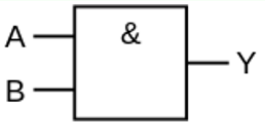
Логічний елемент (логічний вентиль) ***І*** реалізує операцію ***кон'юнкція***. Активний сигнал («логічна 1», «істина») на виході цього вентиля присутній лише тоді, коли на обох його входах присутній активний сигнал. Якщо ж хоча б на одному із входів сигнал пасивний («логічний 0», «хиба»), на виході також буде пасивний сигнал. На рис. 1.1 наведено позначення елемента на схемах і його таблицю істинності, що описує логічну функцію в двійковому коді у вигляді станів вхідних і вихідних змінних.

В нотації алгебри логіки дія цього вентиля записується формулою: $Y=A \wedge B$.

Логічний елемент (логічний вентиль) ***АБО*** реалізує операцію ***диз'юнкція***. Активний сигнал («логічна 1», «істина») на виході цього вентиля присутній тоді, коли на хоча б одному вході присутній активний сигнал. Лише коли на обох його входах сигнал пасивний («логічний 0», «хиба»), на виході також буде пасивний сигнал. Опис вентиля наведено на рис. 1.2.

I (AND)

Позначення на схемах



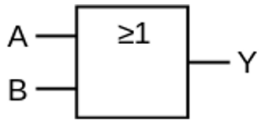
Таблиця істинності

Входи		Вихід
0	0	0
0	1	0
1	0	0
1	1	1

Рисунок 1.1 – Логічний елемент І

АБО (OR)

Позначення на схемах



Таблиця істинності

Входи		Вихід
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

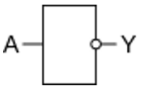
Рисунок 1.2 – Логічний елемент АБО

В нотації алгебри логіки дія цього вентиля записується формулою Y=A ∨ B.

Логічний елемент НІ (інвертор) реалізує операцію ***заперечення***. Опис елемента наведено на рис. 1.3.

Інвертор (NOT)

Позначення на схемах



Таблиця істинності

Вхід	Вихід
0	1
1	0

Рисунок 1.3 – Логічний елемент НІ

В нотації алгебри логіки дія цього елемента записується формулою $Y=\overline{A}$.

Будь-яка функція алгебри логіки може бути задана формулою за допомогою диз'юнкції, кон'юнкції й заперечення. Із цього випливає, що система функцій {І, АБО, НІ} є функціонально повною.

1.2 Тригери

Логічні елементи служать основою для створення більш складних цифрових пристроїв, одним з яких є тригер. Тригер – це цілий клас електронних пристроїв, які можуть тривалий час перебувати в одному з двох стійких станів після припинення сигналу, що змінює стан. Стан виходу тригера визначається не тільки сигналами на його входах, а й попереднім станом пристрою. Таким чином, тригер є найпростішою однобітною чарункою пам'яті.

Найпростіший тригер можна отримати з двох логічних елементів АБО-НІ, або з двох логічних елементів І-НІ. Схемотехнічне позначення RS-тригера наведено на рис. 1.4.

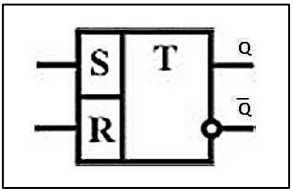


Рисунок 1.4 – Схемотехнічне позначення RS-тригера

Така схема являє собою асинхронний RS-тригер. Він має два входи: S (set) – встановлення, R (reset) – скидання, та два виходи: Q (прямий) та $\overline{Q}$ (інверсний).

При подачі одиниці на вхід **S** вихідний стан стає рівним логічної одиниці. А при подачі одиниці на вхід **R** вихідний стан стає рівним логічному нулю. Стан, при якому на обидва входи **R** і **S** одночасно подані логічні одиниці, в найпростіших реалізаціях є забороненим.

1.3 Вузли комп'ютера

Вузлом комп'ютера називається сукупність функціонально пов'язаних елементів, призначених для виконання певних операцій над двійковими словами. Вузли є основними елементами реалізації апаратних функцій комп'ютера (перетворення, передача, зберігання та управління інформацією). Вони забезпечують перетворення кодів, підрахунок імпульсів, порівняння кодів, зсув двійкових слів.

За функціями, що виконуються, вузли поділяються на регістри, суматори (накопичувального типу), лічильники, дешифратори, шифратори, мультиплексори, демультиплексори, схеми порівняння кодів, програмовані логічні матриці (ПЛМ), аналого-цифрові та цифро-аналогові перетворювачі (АЦП і ЦАП) тощо.

1.3.1 Регістри

Регістр – вузол комп'ютера, призначений для зберігання двійкових слів і виконання над ними деяких логічних операцій. Регістр є сукупністю тригерів, число яких відповідає числу розрядів в слові, та допоміжних схем, що забезпечують виконання деяких операцій, таких як:

– установка регістра в 0 (скидання);

– прийом слова;
– видача слова;
– зсув слова вліво або вправо на необхідну кількість розрядів;
– перетворення послідовного коду в паралельний і навпаки;
– розрядні логічні операції.

Головною класифікаційною ознакою регістрів є спосіб запису інформації до нього. Відповідно до цієї ознаки регістри поділяються на три групи:

– паралельні (регістри зберігання або пам'яті);
– послідовні (регістри зсуву);
– послідовно-паралельні (універсальні регістри).

У ***паралельні регістри*** запис коду відбувається одночасно в усі розряди регістра. Паралельні регістри можуть бути побудовані з будь-яких типів тригерів.

У послідовних регістрах запис інформації, що надходить у послідовному коді, відбувається побітно, у відповідності до її надходження. При цьому інформація просувається від розряду до розряду, відповідно до надходження сигналів синхронізації (запису інформації).

У відповідності до того, в якому напряму відбувається зсув інформації, розрізнюють регістри зі зсувом уліво або вправо. При зсуві вліво інформація переміщується від молодшого розряду до старшого (зсув старшим розрядом уперед), як показано на рис. 1.5. На цьому рисунку показано, як відбувається зсув чотирьох розрядного інформації у межах байту. Зсув відбувається порозрядно у відповідності до надходження тактового сигналу.

Інформація, яка висувається з біту D7, якщо не прийняти спеціальних заходів, безповоротно втрачається. У біт D0 може вводитися будь-який символ (0 чи 1), що на рисунку показано символами «х».

Аналогічно будується схема регістра зі зсувом управо. Відміни будуть полягати лише у позначенні усіх виходів розрядів регістру. Якщо розглядати чотирьохрозрядний регістр зі зсувом управо, то вихід першого розряду буде мати позначення DO_3, наступний – DO_2 і так далі. Вхідний сигнал у такий регістр буде подаватися починаючи зі старшого розряду.

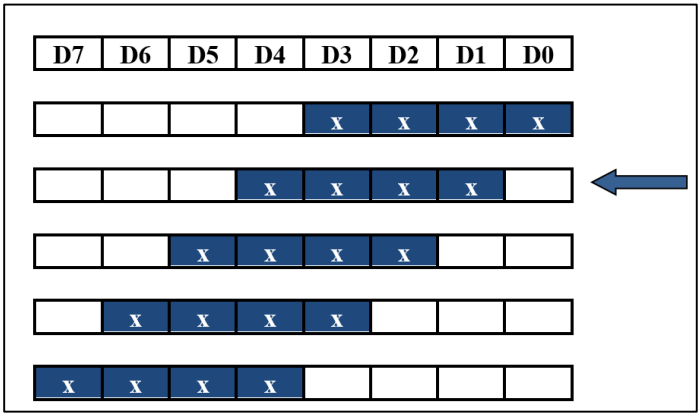


Рисунок 1.5 – Процес зсуву інформації уліво

Послідовно-паралельні регістри забезпечують можливість організації схеми регістра будь-якого типу. Для цього вони доповнюються логічними схемами, які забезпечують необхідну організацію подавання сигналів на входи тригерів регістра відповідно до сигналів керування цими логічними схемами.

В цифровій техніці також використовуються регістри в яких можливо змінювати напрям зсуву у процесі роботи. Такі регістри будуються аналогічно розглянутому вище і мають назву реверсивних. В них з'єднання розрядів між собою виконується за допомогою логічних схем, які здійснюють комутацію відповідно до вхідних сигналів керування напрямом зсуву.

Універсальні регістри дозволяють виконувати як функції перетворення кодів, так і зсув інформації.

Умовне графічне позначення універсального чотирьохрозрядного регістру показано на рис. 1.6, на якому виводи регістра мають такі позначення:

SL – сигнал керування зсувом уліво. Він повинен мати активний рівень впродовж всього циклу зсуву інформації;

DL – вхід даних при зсуві вліво;

SR – сигнал керування зсувом управо. Він повинен мати активний рівень впродовж всього циклу зсуву інформації;

DR – вхід даних при зсуві управо;

C – вхід синхронізації;

DI_0 …DI_3 – входи даних;
DO_0 …DO_3 – виходи даних.

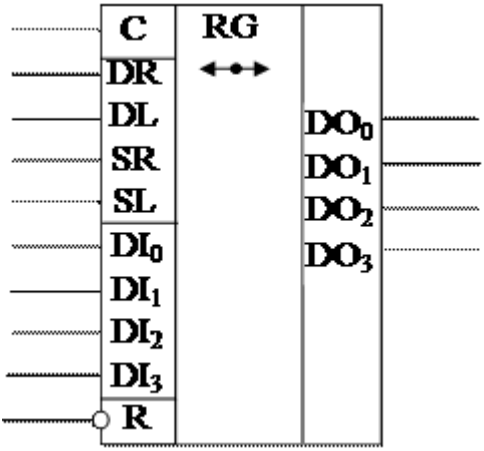


Рисунок 1.6 – Умовне графічне позначення універсального чотирьохрозрядного регістру

1.3.2 Лічильники

Лічильник – накопичувальний вузол комп'ютера, призначений для підрахунку числа імпульсів, що надійшли на його вхід. Вхідні імпульси можуть поступати на лічильник як періодично, так і довільно розподіленими в часі. Лічильник є одним з функціональних основних вузлів комп'ютера, а також різних цифрових керуючих і інформаційно-вимірювальних систем.

Список мікрооперацій лічильника включає попередню установку в початковий стан, інкремент або декремент слова, що зберігається, видачу слів паралельним кодом тощо.

Розрядність лічильника ***n*** рівна числу Т-тригерів. Кожен вхідний імпульс змінює стан лічильника, який зберігається до надходження наступного сигналу. Значення виходів тригерів лічильника Q_n Q_{n-1},…,Q_1 відображають результат рахунку в прийнятій системі числення.

У залежності від алгоритму реалізації виділяють лічильники:

– прямого рахунку, у яких код збільшується на одиницю після надходження на вхід кожного імпульсу;

– зворотного рахунку, у яких код відповідно зменшується після надходження на вхід кожного імпульсу;

– реверсивні, які в залежності від наявності сигналу додавання або віднімання ведуть рахунок в прямому або зворотному напрямках, тобто працюють у тому або іншому режимі за зовнішньою командою;

– з попередньою установкою, які дозволяють спочатку переслати деякий код до лічильника, а потім продовжити перерваний рахунок, починаючи з цього записаного коду.

Коефіцієнт (або модуль) перерахунку M визначає кількість станів лічильника або кількість імпульсів, що надійшли на вхід лічильника, які переводять його в початковий стан. Між числом розрядів лічильника ***n*** і коефіцієнтом перерахунку ***M*** існує співвідношення

$$M=2^n+1$$

На рис. 1.7 наведено умовне графічне позначення реверсивного чотирьохрозрядного лічильника. Літерами *CT (counter)* позначається логічна функція лічильника.

Виводи лічильника мають таке призначення:

+1 – для подачі тактових імпульсів при прямому рахунку;

-1 – для подачі тактових імпульсів при зворотному рахунку;

C – для запису інформації до лічильника у паралельному коді;

R – вхід скидання лічильника;

D1, D2, D4, D8 – для вводу паралельного коду;

1, 2, 4, 8 – виводи результату;

≥15 – індикація переповнення лічильника при прямому рахунку;

≤0 – індикація переповнення лічильника при зворотному рахунку.

1.3.3 Суматори

Суматор – вузол комп'ютера, призначений для додавання двох n-розрядних слів (чисел). Операція віднімання замінюється додаванням слів в оберненому або додатковому кодах.

Суматор є важливою частиною арифметико-логічного пристрою. Функція суматора позначається літерами SM або Σ. Суматор складається з окремих схем, які називаються однорозрядними суматорами; вони виконують усі дії з додавання значень однойменних

розрядів двох чисел (операндів). Багаторозрядні суматори будуються як сукупність одноразрядних.

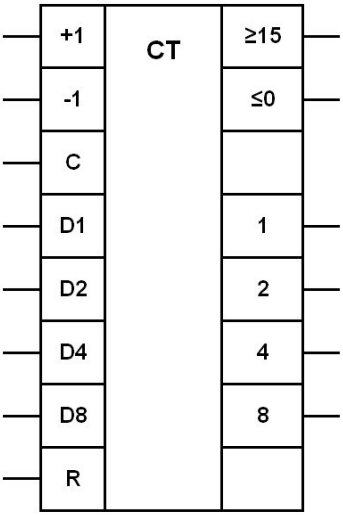


Рисунок 1.7 – Умовне графічне позначення реверсивного чотирьохрозрядного лічильника

Розрізняють комбінаційні та накопичувальні суматори.

У *комбінаційних* суматорах обидва операнда подаються одночасно. При цьому на виходах суматора сума фіксується, доти на входах присутні операнди.

У *накопичувальних* суматорах спочатку подається перший операнд, який запам'ятовується суматором. Після подачі другого операнда в суматорі утворюється сума, яка теж запам'ятовується.

Залежно від способів обробки розрядів операндів розрізняють суматори:

– послідовної дії (розряди обробляються послідовно один за іншим, починаючи з молодшого);

– паралельної дії (всі складові обробляються одночасно, як правило, за один робочий такт);

– послідовно-паралельної дії (одночасно обробляється група розрядів, а між групами обробка йде послідовно).

По числу входів розрізняють два типи однорозрядних комбінаційних суматорів:

– напівсуматор що, має два входи a і b (рис. 1.8, а);

– повний суматор, що має три входи a, b, p (рис. 1.8, 6). Цей пристрій призначений для додавання трьох однорозрядних двійкових чисел a, b, Pi -1 де Pi -1 - сигнал переносу з попереднього молодшого розряду. Подібно до напівсуматора повний суматор має два виходи Si, (сума) і Pi (перенос).

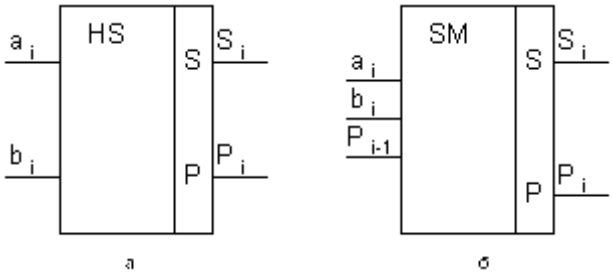


Рисунок 1.8 -- Умовне графічне позначення однорозрядного напівсуматора (а) і повного суматора (б)

1.3.4 Дешифратори та шифратори

Дешифратор – комбінаційний вузол, який призначений для перетворення двійкового коду на вході в керуючий сигнал на одному з виходів. Якщо входів *n,* то вихідних шин повинно бути $N = 2^n$. При подачі на вхід дешифратора двійкового коду на одному з виходів виробляється сигнал «1», а на інших виходах зберігається «О» (дешифратор перетворює код на входах в унітарний код на виходах).

На рис. 1.9 наведено умовне графічне позначення дешифратора «три на вісім», де $\mathbf{a_i}$ – розряди вхідного двійкового коду, $\mathbf{y_i}$, **E** – сигнал дозволу на обробку.

Шифратор - це вузол комп'ютера з декількома входами та виходами, що перетворює сигнал на одному з входів (унарний сигнал) в двійковий код цього входу (рис. 1.10*)*. Шифратор виконує функцію, зворотну щодо дешифратора. Прикладом шифратора є клавіатура, що перетворює сигнали клавіш в код цієї клавіші.

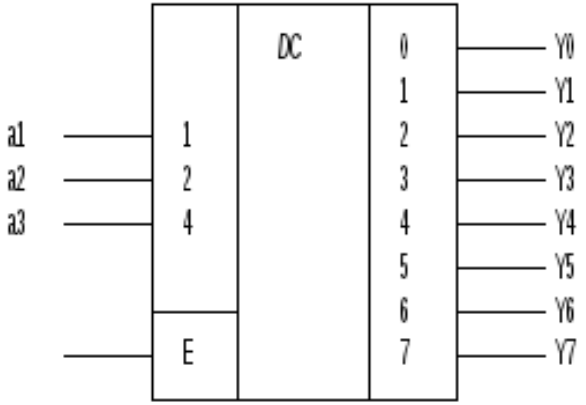


Рисунок 1.9 – Умовне графічне позначення дешифратора

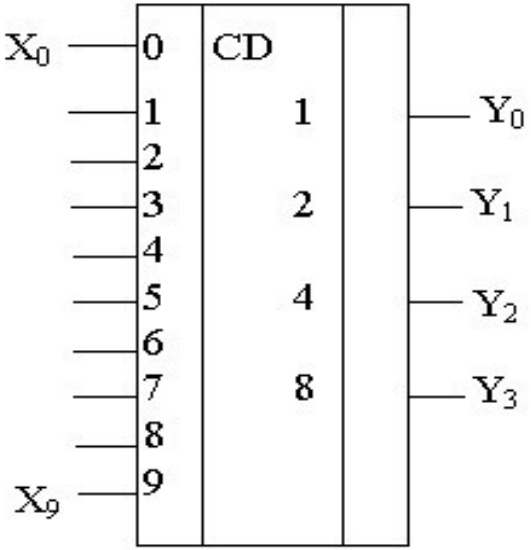


Рисунок 1.10 – Умовне графічне позначення шифратора

1.3.5 Мультиплексори, демультиплексори

Мультиплексором називається функціональний вузол комп'ютера, призначений для почергової комутації (перемикання) інформації від одного з n входів на загальний вихід.

Номер конкретної вхідної лінії, що підключається до виходу в кожний такт машинного часу, визначається адресним кодом **A0, A1, ..., Am-1**. Зв'язок між числом інформаційних **n** і адресних **m** входів визначається співвідношенням $\mathbf{n=2^m}$. Таким чином, мультиплексор реалізує керовану передачу даних від кількох вхідних ліній в одну вихідну.

Мультиплексор забезпечує тимчасове об'єднання каналів і є основним вузлом, який реалізують апаратну функцію передачі даних (рис. 1.11).

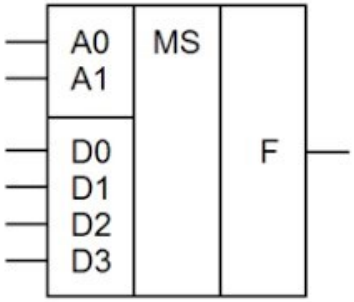


Рисунок 1.11 – Умовне графічне позначення мультиплексора

Демультиплексор виконує функцію, зворотну функції мультиплексора. Він має один інформаційний вхід і кілька виходів. Він являє собою пристрій, який здійснює комутацію входу до одного з виходів, який має задану адресу (номер). На рис. 1.12 показано символічне зображення демультиплексора з чотирма виходами.

Використання демультиплексора може істотно спростити побудову логічного пристрою, що має кілька виходів, на яких формуються різні логічні функції одних і тих же змінних.

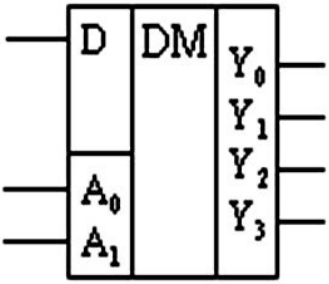


Рисунок 1.12 – Умовне графічне позначення демультиплексора

2 АРХІТЕКТУРА КОМП'ЮТЕРА

2.1 Класична структура комп'ютера

Комп'ютер **–** це універсальний пристрій для обробки дискретних (відокремлених) даних. Комп'ютер виконує операції введення інформації, зберігання та обробки її за певною програмою, виведення одержаних результатів у формі, придатній для сприйняття людиною.

Структура комп'ютера — це сукупність його функціональних елементів і зв'язків між ними. До основних блоків комп'ютера відносяться (рис. 2.1):

- пристрій введення,
- запам'ятовуючий пристрій,
- центральний процесор,
- пристрій виведення.

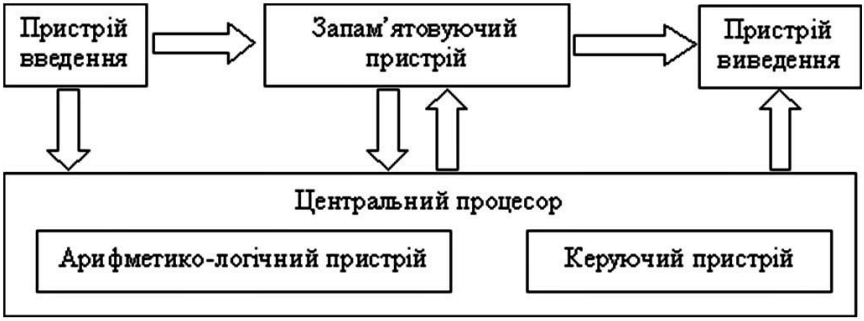


Рис. 2.1 – Загальна структура комп'ютера

Центральний процесор (***мікропроцесор***) — основний блок, призначений для виконання всіх обчислень і забезпечення загального керування роботою ПК.

Пам'ять – функціональна складова, призначена для збереження інформації (програм і даних).

Пристрої введення-виведення (периферія) = забезпечують взаємодію ПК з навколишнім середовищем (користувачами, об'єктами керування та іншими ПК).

Сукупність функціональних блоків (пристроїв) комп'ютера і

зв’язків між ними називають *функціональною структурою* комп'ютера.

Базові принципи побудови та функціонування цифрового комп'ютера сформульовані американським вченим Джоном фон Нейманом (1945-46 рр.).

2.2 Поняття архітектури комп’ютера

Структура комп'ютера визначає конкретний його склад на деякому рівні деталізації (пристрої, блоки, вузли, тощо) і описує всі внутрішні зв'язки. На відміну від структури архітектура визначає правила взаємодії складових частин комп’ютера.

Поняття **архітектури ЕОМ** вперше було успішно застосовано при проектуванні серії обчислювальних машин IBM System/360, серії універсальних ЕОМ загального призначення, кожна з яких мала різну швидкодію та конструктивні особливості, але всі вони були ***програмно сумісними***.

Така сумісність означала можливість виконувати програми без необхідності їх додаткової адаптації до різних моделей серії, що було певною мірою революційним. Справа в тому, що в той час практично всі ЕОМ випускались, як би ми зараз сказали, з унікальною архітектурою і необхідні були суттєві витрати для адаптації існуючого програмного забезпечення до нових моделей обчислювальної техніки. І якщо для спеціалізованих обчислювачів це було платою за високі показники швидкодії, то для класу універсальних ЕОМ така ситуація була неприпустимою.

Спеціалісти фірми IBM при створенні System/360 (S/360) зробили архітектуру єдиною для всіх машин серії, але реалізували її в кожній машині по-різному. В 1964 році було анонсовано відразу 6 моделей S/360.

Архітектура S/360 саме завдяки такій сумісності моделей мала надзвичайний комерційний успіх і отримала свій розвиток в наступний серіях.

Архітектура комп'ютера – це сукупність логічних і фізичних принципів організації комп'ютера (рис. 2.2), які визначають функціональні можливості апаратних засобів, котрі використовуються для представлення програм та даних, а також для керуванням процесом обчислювання. Іншими словами, це представлення комп'ютера з точки зору користувача.

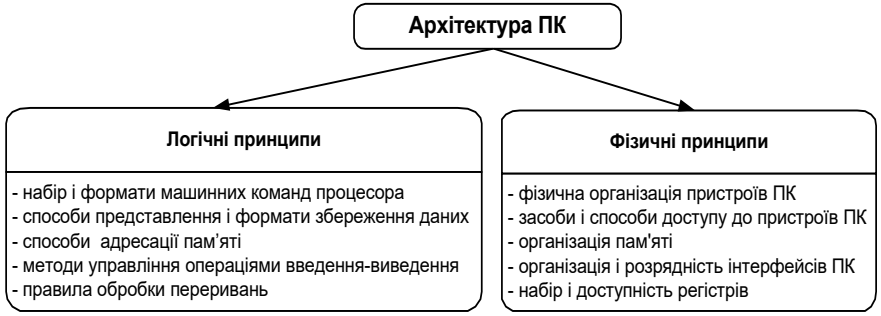


Рисунок 2.2– Основні компоненти архітектури комп'ютера

Зараз найбільшого поширення отримали два типи архітектури: фоннеймановська (або принстонська), запропонована американським вченим Джоном фон Нейманом наприкінці сорокових років 20-го століття, та гарвардська. Обидві вони виділяють два основних вузли: центральний процесор і пам'ять комп'ютера.

У фоннеймановській архітектурі для зберігання програм і даних використовується один простір пам'яті, і нема ніяких ознак, які вказують на тип інформації, що зберігається в чарунці пам'яті. Інформація передається в процесор одним каналом Зміст чарунки інтерпретується при цьому самим мікропроцесором (рис. 2.3,а).

У гарвардській архітектурі пам'ять програм та пам'ять даних поділені і мають свої власні адресні простори та засоби доступу до них (рис. 2.3,б).

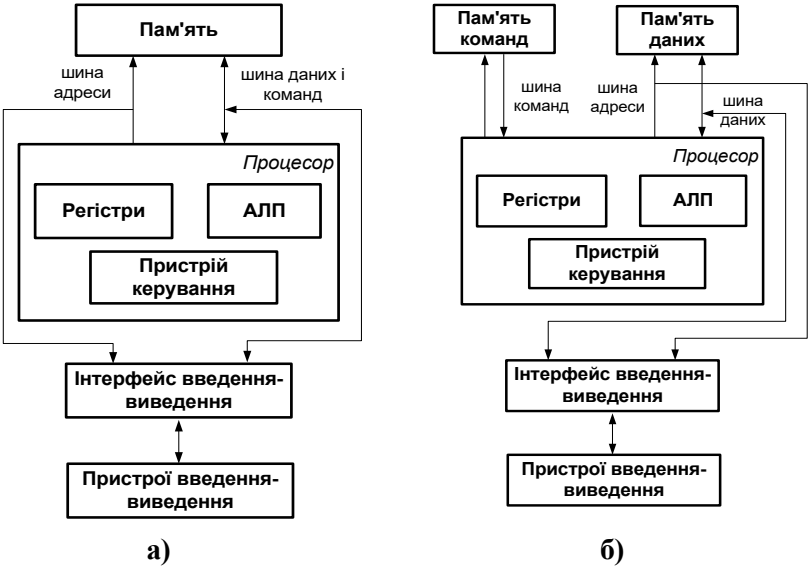


Рисунок 2.3– Основні типи архітектури комп'ютера

Переваги фон-нейманівської архітектури архітектури:

– спрощення устрою МП, так як реалізується звертання тільки до загальної ОП;

– використання єдиної області пам'яті дозволяє оперативно перерозподіляти ресурси між областями програм і даних, що істотно підвищує гнучкість МП.

Переваги гарвардської архітектури:

– застосування невеликих за обсягом пам'яті даних сприяє прискоренню пошуку інформації в пам'яті і збільшує швидкодію МП;

– наявність окремих шини даних і шини команд також дозволяє підвищити швидкодію МП;

– з’являється можливість організувати паралельне виконання програм (кожна пам'ять з'єднується з процесором окремої шиною , що дозволяє одночасно з читанням - записом даних при виконанні поточної команди робити вибірку і декодування наступної команди).

Недоліком гарвардської архітектури є ускладнення архітектури МП і необхідність генерації додаткових керуючих сигналів для пам'яті команд і пам'яті даних.

Сучасний підхід передбачає, що в основі побудови однопроцесорних комп'ютерів, як і раніше, лежать принципи фон-Неймана, хоча і значно модифіковані. Багатопроцесорні ж комп'ютерні системи, які здатні здійснювати паралельні обчислення, базуються на Гарвардській архітектурі.

До докладнішого опису, що визначає конкретну архітектуру, також входять:

– структурна схема комп'ютера;
– засоби і способи доступу до елементів цієї структурної схеми;
– організація і розрядність інтерфейсів;
– набір і доступність регістрів;
– організація пам'яті та способи її адресації;
– набір і формат машинних команд процесора;
– способи представлення та формати даних;
– правила обробки переривань тощо.

За перерахованими ознаками та їх поєднаннями серед архітектур виділяють:

– за розрядністю інтерфейсів і машинних слів: 8-, 16-, 32-, 64-розрядні (ряд комп'ютерів мають інші розрядності);

– за особливостями набору регістрів, формату команд і даних: CISC, RISC, VLIW;

– за кількістю центральних процесорів: однопроцесорні, багатопроцесорні, суперскалярні;

– багатопроцесорні за принципом взаємодії з пам'яттю: симетричні багатопроцесорні (SMP), масивно-паралельні (MPP), розподілені.

2.3 Архітектура мікропроцесора

2.3.1 Призначення та класифікація

Мікропроцесор (МП)– центральний блок персонального комп'ютера, призначений для управління роботою всіх інших блоків і виконання арифметичних і логічних операцій над інформацією.

Основними функціями МП є:

– читання та дешифрування команд із основної пам'яті;
– читання даних з основної пам'яті та зовнішніх пристроїв;
– прийом та обробку запитів і команд від зовнішніх пристроїв

на їхнє обслуговування;

– обробку даних і їх запис до основної пам'яті або зовнішніх пристроїв;

– генерація сигналів управління для всіх інших вузлів і блоків комп'ютера.

Мікропроцесори поділяють на окремі класи відповідно до їх архітектури, структури і функціонального призначення. За функціональним призначенням вони поділяються на МП загального призначення та спеціалізовані. В свою чергу, останні поділяються на мікроконтролери та цифрові процесори сигналів.

Мікропроцесори загального призначення використовуються для вирішення широкого кола завдань обробки різноманітної інформації. Їх основною областю використання є персональні комп'ютери, робочі станції, сервери та інші цифрові системи масового застосування.

Спеціалізовані мікропроцесори орієнтовані на вирішення специфічних завдань управління різними об'єктами. Містять додаткові мікросхеми (інтерфейси), що забезпечують спеціалізоване використання. Мають особливу конструкцію, підвищену надійність.

Мікроконтролери є спеціалізованими мікропроцесорами, які орієнтовані на реалізацію пристроїв керування, вбудованих у різноманітну апаратуру. Характерною особливістю структури мікроконтролерів є розміщення на одному кристалі з центральним процесором внутрішньої пам'яті і великого набору периферійних пристроїв.

Цифрові процесори сигналів (ЦПС) представляють клас спеціалізованих мікропроцесорів, орієнтованих на цифрову обробку вхідних аналогових сигналів. Специфічною особливістю алгоритмів обробки аналогових сигналів є необхідність послідовного виконання ряду команд множення-підсумовування з накопиченням проміжного результату в регістрі-акумуляторі. Тому архітектура ЦПС орієнтована на реалізацію швидкого виконання операцій такого роду. Набір команд цих процесорів містить спеціальні команди MAC (Multiplication with Accumlation), які реалізують ці операції.

2.3.2 Структура мікропроцесора

До складу мікропроцесора зазвичай входять три основні блоки:

- блок керування та синхронізації.
- арифметико логічний пристрій (АЛП);
- блок регістрів;

Для передачі даних між цими блоками використовується внутрішня шина даних. Внутрішня шина даних безпосередньо приєднана до шини даних мікропроцесорної системи.

Мікропроцесорна система (МПС) – це програмно-апаратний комплекс, призначений для реалізації функцій обробки даних, контролю або управління та побудований на одному або декількох мікропроцесорах.

Таким чином, МП – це лише частина обчислювального пристрою, для нормальної організації якого потрібні пам'ять та пристрої вводу-виводу інформації.

Загальні принципи роботи МП доцільно розглядати на прикладі реального МП, в якості якого був обраний МП і8080, архітектура якого фактично стала класичною (рис.2.4).

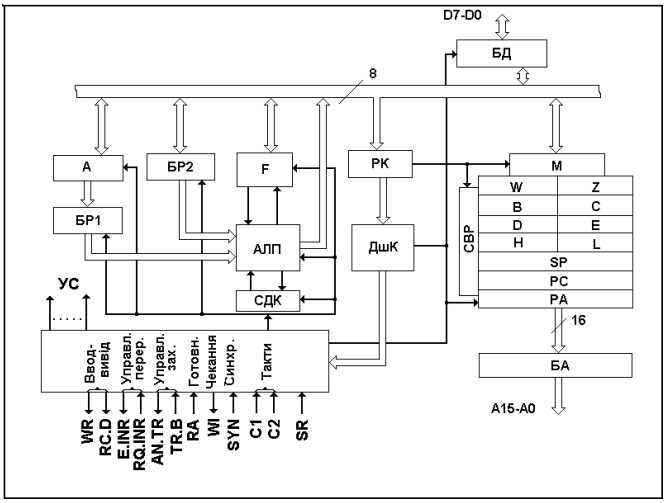


Рисунок 2.4 - Структурна схема мікропроцесора

2.3.3 Блок регістрів

Блок регістрів містить шість 16-розрядних регістрів. Три з них можуть використовуватися як шість окремих 8-розрядних програмно-доступних регістрів загального призначення ***(РЗП) B, C, D, E, H, L*** для зберігання операндів або як три 16-розрядні програмно-доступні регістрові пари ***BC, DE, HL*** для зберігання адрес або двобайтових операндів (рис. 2.5).

Рисунок 2.5 - Блок регістрів мікропроцесора

При виконанні арифметичних та логічних операцій з регістровою адресацією в РЗП зберігаються 8-разрядні операнди, які передаються в блок АЛП для участі в операції. Другий операнд і результат операції зберігаються в блоці АЛП.

Пари регістрів ***BC***, ***DE*** і ***HL*** зазвичай використовують в якості регістрів-вказівників опосередкованої адресації. При цьому основним

регістром-вказівником є пара ***HL***.

Крім того, при складанні двох 16-розрядних чисел в регістрових парах можуть зберігатися 16-розрядні операнди. В цьому випадку вміст будь-якої регістрової пари складається в блоці ***АЛП*** зі змістом пари ***HL***, і результат записується в регістрову пару ***HL***.

Вміст кожного із регістрів можливо переслати в блок АЛП або пам'ять через 8-розрядну внутрішню шину даних. Аналогічно відбувається завантажування регістрів.

Крім того, вміст кожного регістра (пари регістрів) можливо програмно збільшити або зменшити на одиницю. Ця зміна здійснюється за допомогою адресної логіки.

PC (Program Counter) – програмний лічильник (***лічильник команд***) використовується для зберігання адреси наступного байту команди.

Оскільки програма - це послідовність команд, які зберігаються в пам'яті ***мікро-ЕОМ***, для її коректного виконання необхідно, щоб команди знаходились у суворо визначеній послідовності. При цьому саме на лічильник команд покладається функція стеження за тим, яка команда виконується, а яка підлягає виконанню наступною.

Перед виконанням програми в лічильник команд завантажується адреса першої команди програми. Вміст ***PC*** після вибірки кожного байта команди автоматично збільшується схемою адресної логіки. Вміст ***PC*** може бути змінено під впливом самої програми. В результаті відбувається примусова передача керування тій області пам'яті, в якій міститься наступна ділянка програми.

Завантаження та видача змісту ***PC*** відбуваються через мультиплексори та внутрішню магістраль даних. Після включення живлення або виконання команди вміст програмного лічильника обнуляється.

SP (Stack Pointer) – покажчик стека - використовується при запису до стека або читання з нього.

Стек - пам'ять зі спрощеною формою адресації. Це оперативна пам'ять магазинного типу, в якій запис або вибірка слова виконується за принципом: останній записаний елемент вибирається з пам'яті першим ***(LIFO - 1ast in, first out)***.

В I8080 стек організується наступним чином (рис. 2.6). В ОЗП команди розміщуються в чарунках з молодшими адресами, які послідовно зростають. Стек використовує чарунки зі старшими адресами. По мірі

заповнення стека займаються чарунки з адресами, які послідовно убувають. Таким чином, адреси цих двох областей пам'яті змінюються назустріч одна одній.

Покажчик стека містить так названу адресу входу до стека (вершина стека), тобто. адрес чарунки області пам'яті, до якої було зроблене останнє звернення. Зміст SP автоматично за допомогою адресної логіки зменшується на 1 перед кожним занесенням слова до стека та збільшується на 1 після кожного витягання слова зі стека.

Покажчик стека спрощує адресацію пам'яті, але виключає можливість звернення до будь якої чарунки пам'яті. Крім цього, з огляду відсутності у коді команди запису до стеку ***(PUSH)*** та читання із стеку ***(POP)*** адресного поля, зменшується розрядність цих команд та час їх виконання.

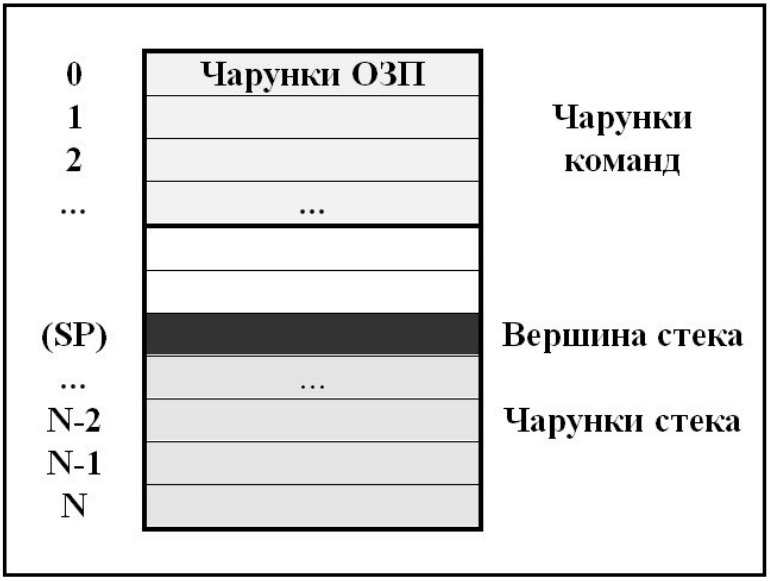


Рисунок 2.6 - Організація стека

Покажчик стека встановлюється програмним шляхом (по команді ***LXI SP*** в старший байт ***SP*** заноситься третій байт команди, в молодший байт ***SP*** - другий байт команди).

Пара регістрів WZ використовується для тимчасового зберігання другого та третього байтів команди, а також для обміну інформацією між регістровими парами (команда ***XCHG***). Регістри програмно недоступні.

Адресна логіка призначена для зберігання та видачі на шину адреси даних та команд. Вона містить:

- буферний регістр адреси (***РА***);
- логічну схему інкремента-декремента (***СІД***);
- адресний буфер (***БА***).

Регістр адреси приймає та зберігає адресу з будь якого 16-бітового регістра. Його вихід зв'язано зі входами ***СІД*** та ***БА***.

Схема інкремента-декремента представляє собою схему швидкого переносу (займу), за допомогою якої є можливість змінювати на 1 зміст будь якої регістрової пари.

Адресний буфер представляє собою 16 вихідних формувачів з трьома станами та презначений для видачі адреси на вихід адресної магістралі ***А15-А0***. Наявність третього (високоомного) стану в ***БА*** дозволяє безпосередньо підключати ***МП*** до системної адресної магістралі ***мікро-ЕОМ.***

2.3.4 Арифметико-логічний пристрій

Арифметико-логічний пристрій (АЛП) призначений для виконання найпростіших арифметичних та логічних операцій над 8-розрядними двійковими числами (складання, віднімання, зсув, порівняння, логічне множення тощо). Більш складні операції, наприклад, множення, виконуються програмно.

Крім того, ***АЛП*** може працювати і з 16-розрядними двійковими числами. При цьому операції виконуються в два етапи: спочатку виконуються операції над молодшими, а потім над старшими байтами. На другому етапі виконання команди в операції бере участь перенос, отриманий в операції з молодшими байтами та збережений у тригері переносу ***С***.

Інформація обробляється в ***АЛП*** з використанням акумулятора (***А***), регістра ознак (***F***), регістра тимчасового зберігання ***БР2*** та регістра акумулятора ***БР1***. При виконанні операції один з операндів пересилається з акумулятора до регістра ***БР1***; інший - надходить з пам'яті або ***РЗП*** через внутрішню магістраль даних до регістру ***БР2***,

звідки надходить до ***АЛП*** через кодоперетворювач у прямому або додатковому коді в залежності від операції, яка виконується.

Регістр тимчасового зберігання ***БР1*** дозволяє запобігти виникненню "гонок", коли акумулятор використовується в одній операції в якості регістра-операнда та в якості регістра-результата.

Результат операції передається через внутрішню магістраль даних до акумулятора, а ознака результату записується в регістр ознак ***F***.

Регістр ознак (F) (або прапорців) (рис. 2.7) призначений для зберігання двійкових ознак, або прапорців:

Рисунок 2.7 – Формат регістру ознак

Двійкові ознаки мають сенс:

C (***Carry***)	– ознака переносу, установлюється в "1" у випадку переносу 1 зі старшого розряду регістра ***A*** при виконанні деяких операцій над даними. При відсутності переносу ознака ***C*** скидується (фактично 1 переносу зі старшого розряду акумулятора запам'ятовується в тригері переносу).
P (***parity***)	– прапорець парності, який приймає значення 1, якщо сумарне число одиничних розрядів у регістрі ***A*** являється парним
AC ***(Auxiliary Carry)-***	– ознака додаткового переносу з молодшого напівбайта числа до старшого. Використовується у командах десяткової арифметики
Z (***zero***)	– ознака нульового результату

S (***sign***) – ознака знака, яка приймає значення одиниці, як що зміст ***A*** стає негативним у результаті виконання деякої операції

Слово стану програми PSW (***Program Status Word***) - це 16-розрядне слово, старший байт якого дорівнює змісту акумулятора, а молодший - змісту регістра ознак ***F***:

$$PSW = [A] + [F]$$

За допомогою регістра ***F*** здійснюється перехід в програмах (порушення природної послідовності виконання команд).

Схема десяткової корекції (***СДК***) призначена для перетворення двійково-кодованого результату в двійково-десятковий код при обробці десяткових чисел.

2.3.5 Блок керування та синхронізації

Блок керування та синхронізації (БКС) відповідно до кодів команд і зовнішніх керуючих сигналів та сигналів синхронізації формує керуючі сигнали для всіх блоків структурної схеми МП, а також керує обміном інформацією між МП, пам'яттю і ПВВ.

Він містить регістр команд (***РК***). дешифратор команд (***ДшК***) та пристрій керування (***ПКр***).

РК призначений для приймання першого байта команди в якому міститься код операції. Код операції після дешифрації використовується для формування керуючих сигналів, під впливом яких виконуються мікрооперації в окремих вузлах МП.

ПКр містить виконану на програмованій логічній матриці управляючу пам’ять, в якій зберігаються мікропрограми окремих операцій. Користувач не може змінити зміст управляючої пам’яті, а значить, і склад команд. Приймання та видача синхронізуючих та керуючих сигналів здійснюється через виводи МП.

БКС реалізує такі функції: початкового встановлення МП, синхронізації, переривань, узгодження швидкодії модулів МП системи.

Функція початкового встановлення МП. Зовнішній сигнал початкового встановлення процесора RESET формується під час ввімкнення джерела живлення МП або під час натискання кнопки RESET. Після появи цього сигналу пристрій керування забезпечує

завантаження нульового значення до програмного лічильника, що ініціює вибирання з пам'яті байта команди з нульовою адресою. Наприкінці вибирання вміст лічильника команд збільшується на одиницю, після чого вибирається байт команд з наступною адресою. Так виконується вся записана у пам'яті програма.

Функція синхронізації. Згідно із зовнішніми керуючими сигналами і сигналами синхронізації БУС синхронізує роботу всіх блоків МП.

Функція переривань. З надходженням сигналу переривання пристрій керування ініціює роботу підпрограми обробки відповідного переривання. Потреба у реалізації функцій переривань виникає тоді, коли під час виконання основної програми треба перевести МП на розв'язання іншої задачі, наприклад оброблення аварійної ситуації або роботи з ПВВ.

Функція узгодження швидкодії модулів мікропроцесорної системи. Під час обслуговування пам'яті та ПВВ із значно меншою швидкодією, ніж МП, узгодження швидкодії вирішується генерацією тактів очікування МП, а під час обслуговування пристроїв з більшою швидкодією, ніж МП, використовується режим ***прямого доступу до пам'яті (ПДП).***

Більшість пристроїв керування реалізовані за принципом **мікропрограмного керування**. Кожна команда, що входить до системи команд мікропроцесора, виконується не миттєво, а поступово такт за тактом в суворій послідовності, яка визначається кодом команди та синхронізується в часі сигналами зовнішнього тактового генератора.

Послідовність певних дій, що виконуються за кожним тактом при виконанні певної команди, визначається **мікропрограмою** виконання команди. Мікропрограма складається з **мікрокоманд**. Виконання мікрокоманди призводить до утворення певного керуючого сигналу, який надходить до задіяного в цій мікрокоманді блоку МП або до шини керування МП-системи.

2.3.6 Виконання команд мікропроцесором

Машинні цикли та такти

Всі дії МП синхронізовані вкладеними циклами трьох рівнів: командними, машинними та мікротактами.

Командний цикл складається з декількох (від 1 до 5) машинних циклів, які позначаються ***М1 - М5*** та реалізують фазу вибірки та фазу виконання команди.

Машинний цикл - це час, потрібний МП для одного звернення до пам'яті або зовнішнього пристрою для вибірки або запису одного байта інформації.

Цикл ***М1*** - єдиний цикл, який реалізує фазу вибірки коду операції. Реалізація машинного циклу покладається на пристрій керування.

Машинний цикл складається з ***машинних тактів*** (мікротактів – МТ). ***МТ*** - це інтервал часу, який дорівнює періоду слідування синхроімпульсів фази ***С1***. При частоті імпульсів ***С1***, яка дорівнює 2 МГц, тривалість ***МТ*** дорівнює 0,5 мс. Один машинний цикл включає 3-5 машинних тактів ***Т1-Т5***. Протягом одного ***МТ*** виконується один мікронаказ пристрою керування.

Зазвичай цикл ***М1*** триває 4-5 тактів, цикли ***М2-М5*** містять 3 такти. Сигнал ***SYN*** видається в першому такті кожного машинного циклу.

Виконання команд

Як зазначалося раніше, для виконання програми необхідно, щоб її команди та дані знаходилися в основній пам'яті (ОП). Функціональну схему взаємодії мікропроцесора та ОП при виконанні програми наведено на рис. 2.8.

Виконання команди здійснюється в такій послідовності:

1) на початку першого машинного циклу за адресою, яка задається вмістом програмного лічильника, з ОЗП зчитується перший байт команди, який містить код операції (КОп). КОп надходить до регістру команд мікропроцесора, що входить до складу його пристрою керування;

2) КОп пересилається до дешифратора команд. При дешифрації коду команди (коду операції та коду адресації операндів) визначається:

– вид операції та адреси необхідних операндів;
– необхідне число машинних циклів для виконання команди.

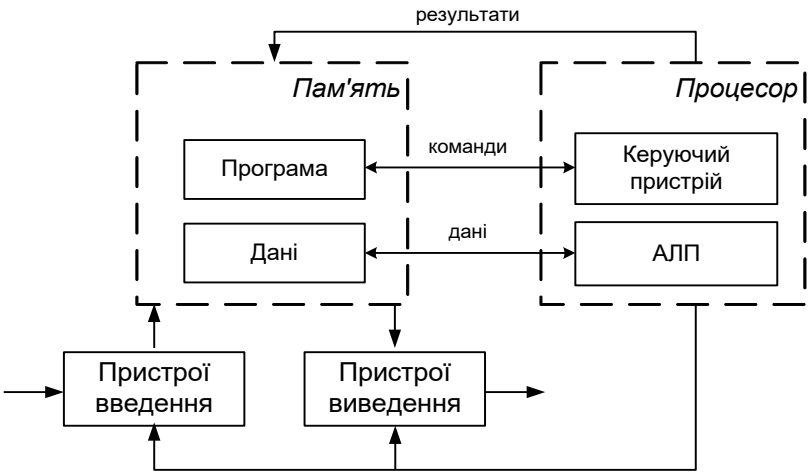


Рисунок 2.8 – Схема взаємодії мікропроцесора і ОП

Якщо для виконання команди не потрібне зчитування операндів з пам'яті (зовнішніх пристроїв) або запис до пам'яті чи зовнішнього пристрою результатів операції, то така команда виконується за один цикл. В іншому випадку виконуються додаткові цикли читання/запису або вводу/ виводу.

3) після зчитування коду поточної команди вміст програмного лічильника РС автоматично збільшується на одиницю, формуючи адресу наступної команди програми. При реалізації безумовних або умовних переходів (розгалужень) або інших змін послідовності виконання команд відбувається завантаження в програмний лічильник нового вмісту, в результаті чого здійснюється перехід до іншої гілки програми. Слід зауважити, що вимога саме послідовного, в порядку зміни адрес в програмному лічильнику, виконання команд є принциповою. Архітектури, які не додержуються такого принципу, взагалі не вважаються фон-нейманівськими;

4) відповідно до операції, що виконується, пристрій керування формує необхідні сигнали для реалізації машинних циклів і необхідну послідовність мікрокоманд в кожному циклі. Слід зауважити, що вимога саме послідовного, в порядку зміни адрес в ЛК, виконання команд є принциповою. Архітектури, які не додержуються такого принципу, взагалі не вважаються фон-нейманівськими.

2.3.7 Формат команд

Мікропроцесор I8080 – одноадресний. Команди подаються одним, двома або трьома байтами (рис. 2.9). Однобайтовий формат використовується для кодування команд звертання до регістрів, непрямого адресування ЗУ, при роботі зі стеком. Двох- та трьохбайтовий формати використовуються для команд з безпосередньою та прямою адресацією.

У другому байті двобайтового формату вказується байт даних або 8-розрядна адреса зовнішнього пристрою, а в трьохбайтовому форматі у другому та третьому байтах вказується двобайтове слово даних або 16-розрядна адреса ЗУ.

Багатобайтові команди зберігаються у сусідніх чарунках пам'яті та адресуються за першим байтом, причому двобайтові слова даних та адрес розташовуються у порядку зростання.

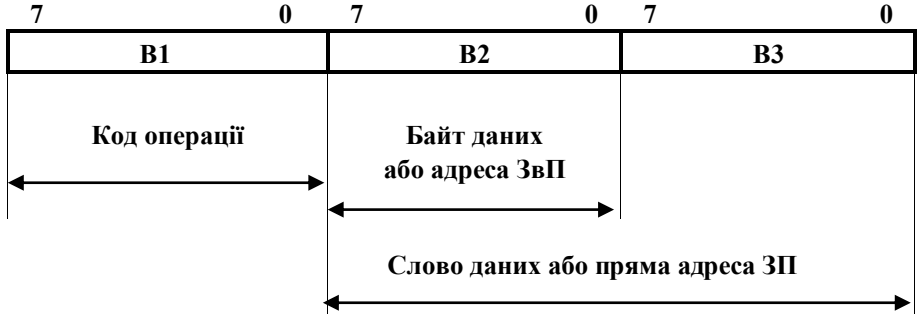


Рисунок 2.9 - Формати команд МП I8080

2.3.8 Методи адресації

В МП I8080 використовуються такі методи адресації:

– неявна;
– пряма;
– безпосередня;
– опосередкована.

Неявна адресація

При цьому методі адресації адреса модулю МПС, з яким працює дана команда, ніде не згадується та визначається кодом операції.

Неявна адресація використовується для :

– звертання до акумулятору. Наприклад, команда **CMA** (інвертувати зміст акумулятору);

– звертання до бітів регістру ознак, наприклад, **STC** (встановити біт переносу);

– для адресації тригеру переривань (команди заборони та дозволяння переривань).

Пряма адресація

При цьому методі адресації адреса елементу МПС, який використовується підчас виконання операції, задається у команді. Пряма адресація використовується для адресування:

1) пам'яті; в цьому випадку виконавча адреса записується у другому та третьому байтах команди, наприклад, **STA 1000H** (завантажити чарунку з адресою 1000H вмістом акумулятора);

2) регістрів зовнішніх пристроїв; у цьому випадку виконавча адреса записується у другому байті команди, наприклад, **OUT 05H** (завантажити вміст акумулятора до зовнішнього пристрою номер 5);

3) регістрів і регістрових пар МП; у цьому випадку виконавча адреса записується у полі коду операції, наприклад, **MOV A, B** (передати вміст регістру **B** до акумулятора).

Безпосередня адресація

При цьому методі адресації адреса елементу, який використовується під час виконання операції МП, вказана безпосередньо у команді.

Безпосередня адресація використовується в операціях безпосереднього завантаження регістрів та регістрових пар, наприклад, **MVI B,05H, LXI H,0016H**.

Опосередкована адресація

При цьому методі адресації виконавча адреса попередньо формується в одній з регістрових пар B, D, H. Наприклад, **STAX D** (записати вміст акумулятора до чарунки пам’яті, з адресою,

розміщеною в регістровій парі **D**.

Різновидом непрямого адресації є адресація з автоіндексуванням, при якому значення адреси, яка зберігається у вказівнику стека **SP**, автоматично збільшується або зменшується на 2 після звертання до стека, наприклад, **PUSH PSW**.

Систему команд МП І8080 наведено у додатку А.

2.3.9 Розширення мікропроцесорної системи

До мікропроцесора та системної шині поряд з типовими зовнішніми пристроями можуть бути підключені й додаткові плати з інтегральними мікросхемами, що розширюють і поліпшують функціональні можливості мікропроцесора. До них відносяться математичний співпроцесор, контролер прямого доступу до пам'яті, співпроцесор вводу/виводу, контролер переривань тощо

Математичний співпроцесор використовується для прискорення виконання операцій над двійковими числами з плаваючою комою, над кодованими десятковими числами, для обчислення тригонометричних функцій. Математичний співпроцесор має свою систему команд і працює паралельно з основним мікропроцесором, але під управлінням останнього. В результаті відбувається прискорення виконання операцій в десятки разів. Моделі мікропроцесора, починаючи з МП 80486 DX, включають математичний співпроцесор в свою структуру.

Контролер прямого доступу до пам'яті звільняє мікропроцесор від прямого управління накопичувачами на магнітних дисках, що істотно підвищує ефективну швидкодію комп'ютера.

Співпроцесор вводу/виводу за рахунок паралельної роботи з мікропроцесором значно прискорює виконання процедур вводу/виводу при обслуговуванні декількох зовнішніх пристроїв, звільняє мікропроцесор від обробки процедур вводу/виводу, в тому числі реалізує режим прямого доступу до пам'яті.

Переривання – це тимчасовий зупинка виконання однієї програми з метою оперативного виконання іншої, в даний момент більш важливої. ***Контролер переривань*** обслуговує процедури переривання, приймає запит на переривання від зовнішніх пристроїв, визначає рівень пріоритету цього запиту і видає сигнал переривання в мікропроцесор.

2.3.10 Режими процесора

Режими процесора призначені для виконання програм у різних середовищах. В різних режимах можливості МП неоднакові, тому що команди виконуються по-різному. В залежності від режиму процесора змінюється схема управління пам'яттю системи та завданнями.

Процесори можуть працювати в трьох режимах: реальному, захищеному та віртуальному реальному режимі (реальному всередині захищеного).

Реальний режим

У першому персональному комп'ютері IBM PC використовувався процесор I8088, який міг виконувати 16-розрядні команди, застосовуючи 16-розрядні внутрішні регістри, та адресувати тільки 1 Мбайт пам'яті, використовуючи 20 розрядні адресні регістри.

Все програмне забезпечення PC спочатку було призначено для цього процесора; воно було розроблено на основі 16-розрядної системи команд і моделі пам'яті об'ємом 1 Мбайт. Наприклад, DOS, все програмне забезпечення DOS, Windows від 1.x до 3.x і всі застосунки для Windows від 1.x до 3.x написані в розрахунку на 16-розрядні команди. Ці 16-розрядні операційні системи та застосунки були розроблені для виконання на цьому процесорі.

Всі програми, що виконуються в реальному режимі, повинні використовувати тільки 16-розрядні команди, 20-розрядні адреси і підтримуватися архітектурою пам'яті, розрахованою на ємність до 1 Мбайт.

Для програмного забезпечення цього типу зазвичай використовується однозадачний режим, тобто одночасно може виконуватися тільки одна програма. Немає ніякого вбудованого захисту для запобігання перезапису елементів пам'яті однієї програми або навіть операційної системи іншою програмою. Це означає, що при виконанні декількох програм можуть бути зіпсовані дані або код однієї з них, а це може привести всю систему до зупинки або краху.

Захищений режим

Першим 32-розрядним процесором, призначеним для PC, був I386. Цей чіп міг виконувати абсолютно нову 32-розрядну систему команд. Щоб повністю використовувати перевагу 32-розрядної системи команд, були необхідні 32-розрядна операційна система та 32-

розрядні застосунки.

Цей новий режим називався захищеним, тому що програми, що працюють у ньому, захищені від перезапису своїх областей пам'яті іншими програмами. Такий захист робить систему більш надійною, оскільки жодна програма з помилками вже не зможе так легко пошкодити інші програми або операційну систему. Крім того, програму, "потерпілу крах", можна досить просто завершити без збитку для всієї системи.

Захищений режим має багато переваг:

– доступна вся системна пам'ять (не існує межі 1 Мбайт);

– операційна система може організувати одночасне виконання декількох завдань (багатозадачність);

– підтримується віртуальна пам'ять, тобто операційна система при необхідності може використовувати жорсткий диск в якості розширення оперативної пам'яті.

– здійснюється швидкий (32/64-розрядний) доступ до пам'яті і підтримується робота 32-розрядних операцій вводу-виводу.

Віртуальний реальний режим

Для зворотної сумісності 32-розрядна система Windows використовує третій режим в процесорі – віртуальний реальний режим. Віртуальний реальний, по суті, є режимом виконання 16-розрядного середовища (реальний режим), яке реалізовано всередині 32-розрядного захищеного режиму (тобто віртуально, а не реально).

Виконуючи команди у вікні підказки DOS усередині Windows, ви створюєте віртуальний сеанс реального режиму. Оскільки захищений режим є справді багатозадачним, фактично можна виконувати декілька сеансів реального режиму, причому в кожному сеансі власне програмне забезпечення працює на віртуальному комп'ютері. І всі ці додатки можуть виконуватися одночасно, навіть під час роботи інших 32-розрядних програм.

Віртуальне реальне вікно повністю імітує середовище процесора 8088, і, якщо не враховувати швидкодію, програмне забезпечення буде виконуватися так, як воно виконувалося першим PC в реальному режимі. Кожна віртуальна машина отримує власний 1 Мбайт адресного простору і власний екземпляр реальних апаратних підпрограм управління апаратурою (базову систему вводу-виводу), причому при цьому емулюються всі регістри та можливості реального режиму.

Віртуальний реальний режим використовується при виконанні програм у вікні DOS, а також при виконанні 16-розрядних програм, написаних для DOS або Windows. При запуску програми DOS операційна система Windows створює віртуальну машину DOS, на якій цей додаток може виконуватися.

Важливо відзначити, що всі процесори Intel (а також Intel-сумісні AMD і Cyrix) при включенні живлення починають працювати в реальному режимі. При завантаженні 32-розрядна операційна система автоматично перемикає процесор в 32-розрядний режим і управляє ним в цьому режимі.

2.4 Система пам'яті комп'ютера

2.4.1 Класифікація запам'ятовуючих пристроїв

За функціональним призначенням всі запам'ятовуючі пристрої (ЗП), які використовуються в комп'ютері, підрозділяються на такі групи:

– **понадоперативні ЗП**, які уявляють собою набір регістрів МП, вміст яких безпосередньо використовується при обробці інформації в МП;

– **оперативні ЗП**, які зберігають оперативну інформацію (операнди, частини програми), потрібну в процесі роботи;

– **постійні ЗП**, призначені для тривалого зберігання незмінної в процесі роботи комп'ютера інформації (програми, константи);

– **зовнішні ЗП**, призначені для довготривалого зберігання великих обсягів інформації з невеликою питомою вартістю біта інформації. Зовнішня пам'ять недоступна процесору для безпосереднього звернення

– **кеш-пам'ять** – буферна, недоступна для користувача швидкодіюча пам'ять, яка автоматично використовується комп'ютером для прискорення операцій з інформацією, що зберігається в більш повільно діючих запам'ятовуючих пристроях. Наприклад, для прискорення операцій з основною пам'яттю організується регістрова кеш-пам'ять усередині мікропроцесора (кеш-пам'ять першого та другого рівнів) або поза мікропроцесора на материнській платі; для прискорення операцій з дисковою пам'яттю.

ОЗП і ПЗП утворюють ***основну пам'ять МПС***, яка реалізується

на основі напівпровідникових запам'ятовуючих елементів, або чарунок, які зберігають двійкову інформацію.

Основна пам’ять відноситься до класу ***ЗП*** з довільною вибіркою ***(ЗПДВ)***. На відміну від пам’яті з послідовним доступом у ***ЗПДВ*** до інформації можна звертатися в довільному порядку, і час звернення до чарунки пам’яті не залежить від її адреси.

В послідовній пам’яті дані можливо зчитувати тільки у тому порядку, в якому вони записувалися (дисципліна обслуговування FIFO – First IN First Out), або зворотному (дисципліна обслуговування LIFO – Last IN First Out).

2.4.2 Основні характеристики запам'ятовуючих пристроїв

Запам'ятовуючі пристрої характеризуються сукупністю якісних показників:

– ***ємність*** ЗП визначається максимально можливою кількістю бітів інформації, що зберігається.

– ***питома ємність*** – кількістю чарунок пам’яті (або бітів) на кристал. Важливою характеристикою *ЗП* є інформаційна організація кристала пам’яті M×N, де M – кількість слів, N – розрядність слова. Наприклад, кристал ємністю 16Кбіт може мати різноманітну організацію: 16Кх1, 4Кх4, 2Кх8.

– ***ширина вибірки*** пов’язана з інформаційною організацією та визначається кількістю інформації, яка записується до ЗП або зчитується з нього за одне звернення. При однаковому часі звернення ЗП з більшою шириною вибірки має більшу інформаційну ефективність.

– ***швидкодія*** ЗП характеризується двома параметрами: *часом вибірки Тчв*, який представляє собою інтервал часу між моментом подачі сигналу вибірки та появою зчитаних даних на виході, та *циклом запису Тцз*, який визначається мінімально допустимим часом між моментом подачі сигналу вибірки при запису та моментом, коли припустиме наступне звернення до пам’яті.

– ***швидкість обміну інформацією*** між ЗП та іншими пристроями, яка визначається кількістю біт (байт), які передаються в одиницю часу. Вимірюється в бітах за секунду або в бодах (1біт/1 сек = 1 бод);

– ***питома вартість***, яка визначається співвідношенням

вартості ЗП до інформаційної ємності, тобто. вартість біта інформації, що зберігається .

– ***надійність*** та ***потужність***, що споживається.

– ***здібність зберігання інформації*** при відключенні джерела живлення. Розрізняють енергонезалежну пам'ять, в якій при збоях у роботі системи живлення інформація не руйнується, та енергозалежну, в якій руйнується.

2.4.3 Оперативні запам'ятовуючі пристрої

Оперативні запам'ятовуючі пристрої (ОЗП) призначені для тимчасового зберігання даних і команд, необхідних процесору для виконання операцій в поточному сеансі роботи. Крім того, доступ до даних оперативної пам'яті відбувається набагато швидше, ніж до даних зовнішньої пам'яті, наприклад жорсткого диска, тому вона й називається оперативною.

В англійській мові ОЗП відповідає термін ***RAM*** (***Random Access Memory*** – пам'ять довільного доступу), що відображає властивість ОЗП надавати з однаковою швидкістю доступ до будь-якої чарунки пам'яті незалежно від її адреси.

Робота комп'ютера – це, перш за все, робота процесора з оперативною пам'яттю. При включенні комп'ютера в оперативну пам'ять завантажуються з диска програми й дані для роботи операційної системи та роботи окремих пристроїв, а потім прикладні програми, які відкриває користувач.

Оперативна пам'ять зберігає дані тільки на час, поки комп'ютер включений, тому вона тимчасова (на час сеансу роботи) та енергозалежна . Дані в пам'яті втрачаються при виключенні комп'ютера або перезавантаження операційної системи.

Процесор виконує обчислення за програмою, розміщеною в оперативній пам'яті, обмінюється з пам'яттю даними, відправляє дані з пам'яті до зовнішніх ЗП або мережі.

Об'єм оперативної пам'яті визначає, наскільки великі програми можуть виконуватися, скільки даних буде підготовлено їм для доступу, а також скільки програм можуть виконуватися одночасно, що дуже важливо для швидкодії.

За принципом зберігання інформації напівпровідникові ОЗП

підрозділяються на два основні класи:

– динамічні (**DRAM** – Dynamic RAM);
– статичні (**SRAM** – Static Random Access Memory).

В ***динамічних ОЗП*** стан біта пам’яті запам'ятовується у вигляді наявності або відсутності заряду на конденсаторі, в якості якого виступає вхідна ємність MOS-транзистора.

Це найбільш розповсюджений і економічно доступний тип пам'яті. Недоліки цього типу пам’яті пов'язані, по-перше, з тим, що як заряджання, так і розряджання конденсаторів потребує певного часу, тобто записування даних відбувається порівняно повільно. По-друге, через наявність струмів відтоку заряди конденсаторів мають властивість розсіюватися в просторі, причому дуже швидко. Тому їх необхідно з визначеною періодичністю регенерувати (оновлювати). Під час регенерації запис нової інформації повинен бути заборонений, що викликає непродуктивну витрату ресурсів обчислювальної системи.

Для регенерації пам’яті потрібен спеціальний блок, який автоматично виконує періодичний перезапис вмісту усіх чарунок пам’яті (приблизно кожну мілісекунду).

В ***статичних ОЗП*** кожний біт пам’яті уявляє собою тригер, стан якого визначає стан відповідного біта пам’яті. Оскільки у тригері зберігається не заряд, а стан (ввімкнено/вимкнено), цей тип пам'яті забезпечує більш високу швидкодію, хоч технологічно вона складніша і відповідно дорожча.

Динамічна пам’ять у зрівнянні зі статичною має більшу питому ємність, меншу вартість, але більше енергоспоживання та меншу швидкодію.

Вважаючи характеристики типів ОЗП, у якості ОЗП комп’ютера зараз використовується динамічна пам’ять. Статична пам’ять використовується в основному для побудови кеш-пам'яті.

Оперативна пам'ять у комп'ютері розміщується на стандартних панельках, так званих модулях пам’яті. Модулі оперативної пам'яті встановлюються у відповідні роз'єми на материнській платі.

2.4.4 Постійні запам’ятовуючі пристрої

Постійна пам'ять (ПЗП або ROM – англ. Read Only Memory – пам’ять тільки для читання) – швидкодіюча енергонезалежна пам'ять,

призначена для зберігання інформації, що не змінюється під час виконання програм. Ця пам’ять невелика за ємністю (кілька сотень кілобайтів) і містить програму тестування пристроїв комп’ютера при ввімкненні – POST (англ. Power-On Self Test – самоперевірка при ввімкненні енергії) та базову систему введення-виведення – BIOS (англ. Basic Input/Output System).

BIOS використовується для завантаження операційної системи або установки нової. Крім того, програма BIOS встановлює потік даних між операційною системою комп'ютера та приєднаними пристроями: жорстким диском, клавіатурою, мишею, принтером, відеосистемою, керує енергоспоживанням комп’ютера.

При включенні комп'ютера процесор звертається до ПЗУ, зчитує програму BIOS, починає її виконувати та здійснює тестування основних пристроїв: клавіатури, оперативної пам'яті, дисководів тощо. Якщо не знайдено жодного пристрою або якийсь пристрій не працює, BIOS повідомляє про помилки звуковими сигналами або текстом па екрані. В разі нормального проходження початкового тестування встановлюється зв'язок системної плати з пристроями, підключаються клавіатура, жорсткий диск, і починається процес завантаження операційної системи.

Після успішного завантаження операційної системи до оперативної пам'яті подальше управління комп'ютером бере на себе операційна система, яка в подальшому виконує завантаження та управління прикладними програмами.

Система BIOS зберігає програму установки Setup *(англ.* Set up - встановити). Програма дозволяє користувачеві встановити клавішами клавіатури деякі настройки BIOS*,* які записуються в окрему постійну CMOS -пам'ять, що живиться від акумуляторної батареї. CMOS-пам'ять (Complimentary Metal Oxide Semiconductor Memory) представляє собою пам'ять для зберігання конфігурації комп'ютера. Вона має низьке енергоспоживання та не змінюється при відключенні живлення.

Ця пам'ять розташовується на контролері периферії, для електроживлення якого використовуються спеціальні акумулятори. У *CMOS* зберігаються деякі настройки системи, поточна дата й час (їх можна налаштувати також за допомогою операційної системи), пароль на вхід до комп'ютера.

Постійна пам’ять виготовляється у вигляді спеціальної

мікросхеми, яку розміщують на системній платі. Крім основної системи BIOS на окремих мікросхемах ПЗП, розміщених на платах розширення (відеокартах, мережевих адаптерах), зберігаються BIOS цих плат.

Починаючи з 1996 року у всіх комп'ютерах BIOS записується на мікросхему Flash ROM. Програми в цій мікросхемі можна стирати і перепрограмовувати без програматора. Використовування Flash ROM дозволяє записати нову версію BIOS без видалення й заміни мікросхеми. Оновлення програм BIOS завантажуються з WEB-сервера виробника системної плати.

2.4.5 Кеш-пам'ять

При організації швидкодіючої буферної пам'яті часто використовують так звані ***асоціативні ЗП***.

Асоціативна пам'ять (АП) є особливим видом машинної пам'яті, що використовується для дуже швидкого пошуку. Відома також як *пам'ять, що адресується за вмістом* (англ. *Content-addressable memory*, *CAM*).

При використанні звичайного ОЗП користувач задає адресу пам'яті та отримує слово даних, що зберігається за цією адресою. При використанні АП користувач вказував слово даних, а АП шукає його у всій пам'яті, щоби з'ясувати, чи зберігається воно де-небудь. Якщо слово даних знайдено, АП повертає список однієї або більше адрес зберігання, де слово було знайдено.

Кеш (від англ. cache – схованка) – особлива швидкісна буферна пам'ять невеликої ємності, яка зберігає вміст і адресу даних, до яких часто звертається процесор. Під час чергового звертання процесора до адреси пам'яті, перевіряється наявність цієї адреси у кеші. Якщо відповідні дані наявні, вони передаються процесору з кеша. Це дозволяє скоротити тривалість обміну, оскільки швидкодія кеша більша за швидкодію звичайної пам'яті. Вибір даних зі звичайної пам'яті здійснюється лише тоді, коли потрібні процесору дані в кеші відсутні.

Найчастіше кеш залишається прозорим для програміста, тому що система команд процесора, зазвичай, не містить команд роботи з кешем

Основна пам'ять ***M*** (рис.2.10) поділена на ***N*** сторінок (зазвичай обсягом 1024 байт). Кеш містить ***n*** сторінок такого ж обсягу. При цьому ***N>>n.*** Будь яка сторінка кеша може заповнюватися інформацією з ***M*** всього за кілька тактів. В кожній сторінці кеша розташовані данні зі суміжними адресами, різні ж сторінки кеша можуть не стикуватися за адресами.

При зверненні до *M* за допомогою асоціативної пам'яті виконується перевірка наявності потрібної інформації в кеші. Якщо інформація мається, робиться звернення до кеша.

Якщо потрібна сторінка в кеші відсутня, то з ***M*** до кеша робиться запис нової сторінки.

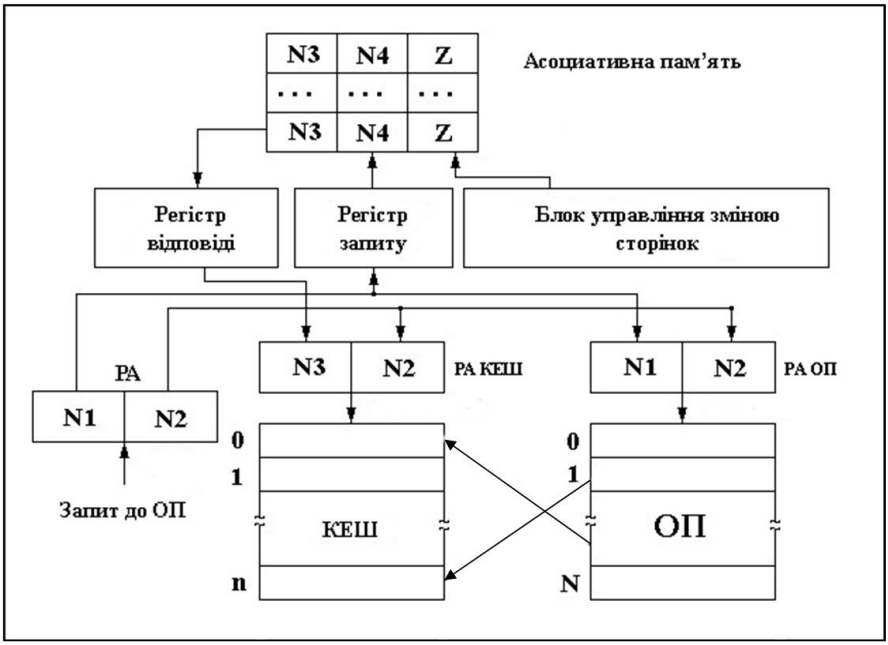


Рисунок 2.10 - Структура кеш-пам'яті

На рисунку 2.10 використовуються такі умовні позначення:

N1 і **N2** – номери сторінки та слова, вказані в запиту до ОП;

N3 – номер сторінки кеша;

N4 – номер сторінки ОП, розміщеної в кеші.

При всякому звернені до ***M*** за допомогою асоціативної пам'яті робиться перевірка наявності даної сторінки у кеші. Якщо вона є, то адреса в кеші буде:

A=N3 (із асоціативної пам'яті) + ***N2***(із команди звернення).

До поля ***Z*** періодично заноситься деяке число. При кожному звернені до сторінки кеша із цього числа відраховується одиниця. Коли в кеші не виявляється необхідної інформації, потребується замінити сторінку. Для цього проглядаються поля ***Z***. Там, де кількість найбільша, сторінка вважається такою, що рідко використовується. Замість неї з ***M*** вводиться нова сторінка, а асоціативна пам'ять відповідно коректується.

Ефективність кеша залежить також від характеру розподілу даних в оперативній пам'яті. Нераціональний розподіл даних призводить до частої зміни сторінок.

Однією з проблем є фундаментальна проблема балансу між затримками кеша та інтенсивністю влучень. Великі кеші мають більш високий відсоток влучень але, разом з тим, і велику затримку. Щоб послабити протиріччя між цими двома параметрами, більшість комп'ютерів використовує кілька рівнів кеша, коли після маленьких і швидких кешей знаходяться більш повільні великі кеші. Зазвичай використовується до 3 рівнів в ієрархії кеша. В деяких системах реалізують 4 рівні кеш-пам'яті.

Багаторівневі кеші зазвичай працюють в послідовності від менших кешей до великих. Спочатку відбувається перевірка найменшого і, в той же час, найшвидшого кеша першого рівня (L1), у разі попадання процесор продовжує роботу на високій швидкості. Якщо менший кеш дав промах, перевіряється наступний, трохи більший і більш повільний кеш другого рівня (L2), і так далі, поки не буде запиту до основного ОЗП.

2.4.6 Накопичувачі на магнітних дисках. Фізичний формат

На жорсткому диску (англ. – hard disk drive (***HDD***)), або накопичувачу на магнітних дисках (НМД), дані записуються та зчитуються універсальними головками читання/запису з поверхні

магнітних дисків, що постійно обертаються. Під час роботи пристрою між головкою та поверхнею диска утворюється дуже малий повітряний зазор (повітряна подушка), і головки читання/запису не торкаються дисків. При зупинці головки переміщуються за межі поверхні дисків, тому механічний контакт між головками та пластинами практично виключений. Така конструкція забезпечує довговічність запам'ятовуючих пристроїв цього типу.

Інформація розміщується в секторах на концентричних ***доріжках*** поверхні диска (рис. 2.11). Зазвичай НМД мають декілька дисків, і дані записуються на обох сторонах кожного з них. Однойменні доріжки декількох поверхней утворюють ***циліндр***. Кількість робочих поверхней та циліндрів є характеристикою дисководу.

Кожна доріжка розбивається на сектори**.** ***Сектори*** є елементами даних диска, що мінімально адресуються. Обмін інформацією між ОЗП та дисками здійснюється тільки секторами. Кожний сектор складається з поля даних та поля службової інформації.

Традиційно розмір сектора становить 512 байт. В сучасних HDD, виконаних за технологією ***Advanced Format*** (розширений формат), використовуються сектори розміром 4 кілобайт. Це дозволяє значно підвищити корисний об'єм області зберігання даних диска, поліпшити його функціональні якості (знизити час читання/запису та доступу, знизити гучність, нагрів, знос механіки диска).

Кількість циліндрів, секторів на доріжці та розмір сектора встановлюється при форматуванні (розмітці) диска.

Практично у всіх сучасних накопичувачах використовується так званий ***зонний запис*** зі змінною кількістю секторів на доріжці. Доріжки, більш віддалені від центру (а значить, і більш довгі), містять більше секторів, ніж доріжки, розташовані близько до центру (рис. 2.12).

При зонному запису циліндри розбиваються на групи, які називаються зонами, причому в міру просування до зовнішнього краю диска доріжки розбиваються на все більше число секторів. У всіх циліндрах, що відносяться до однієї зоні, кількість секторів на доріжках однакова. Можлива кількість зон залежить від типу накопичувача; в більшості пристроїв їх буває 10 і більше

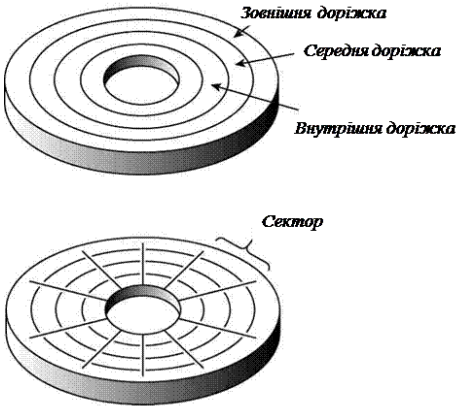


Рисунок 2.11 – Доріжки та сектори

Ще одна властивість зонного запису полягає в тому, що швидкість обміну даними з накопичувачем може змінюватися і залежить від зони, в якій в конкретний момент розташовуються головки. Відбувається це тому, що секторів в зовнішніх зонах більше, а кутова швидкість обертання диска постійна.

Метод зонного запису дозволив виробникам підвищити ємність пристроїв на 20-50% в порівнянні з накопичувачами, в яких число секторів на доріжці фіксоване.

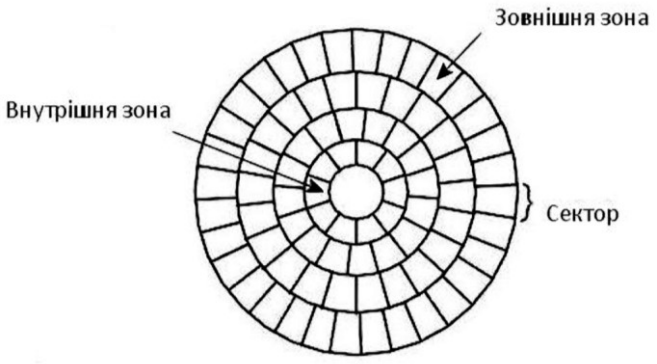


Рисунок 2.12 – Зонне форматування

Фізична адреса сектора визначається тріадою **[*c*-*h*-*s*],** в якій

– ***c*** – номер циліндра (доріжки). Доріжки нумеруються, починаючи з нуля. Нульовий номер має сама крайня зовнішня доріжка на диску;

– ***h*** – номер поверхні диска (магнітної головки). Нумеруються, починаючи з нуля;

– ***s*** – номер сектора на доріжці, починаючи з одиниці.

Наприклад, тріада [1-0-2] адресує сектор 2 доріжки (циліндра) 1 нульової поверхні.

Для жорсткого диска нумерація секторів на доріжці не відповідає послідовності їх фізичного розташування. Висока швидкість обертання диска не дозволяє передати до комп'ютеру зміст зчитаного сектора до підводу до головок фізично наступного сектора.

Ось чому сектор з номером, на одиницю більшим попереднього, розташовується за ним через певну кількість секторів (коефіцієнт черегування **n**). При **n=3** нумерується кожний третій сектор, два пропускаються. При цьому уся доріжка зчитується за **n** оборотів замість **S** (кількості секторів).

Така нумерація встановлюється за допомогою процедури, яка має назву ***форматування нижнього рівня.***

Швидкість роботи того або іншого жорсткого диска залежить від частоти його обертання, швидкості переміщення системи головок і кількості секторів на доріжці.

Швидкість 5400 об/хв в основному використовуються в ноутбуках, 7200 об/хв – в настільних ПК (90% всіх жорстких дисків). Швидкості 10 000 і 15 000 об/хв – у серверах і високопродуктивних робочих станціях. Слід зазначити, що надійність HDD обернено пропорційна швидкості обертання диску.

2.4.7 Логічна структура жорсткого диску

З апаратної точки зору будь-який жорсткий диск можна уявити як сукупність секторів, адресованих тим чи іншим способом (CHS або LBA), і кожен сектор може бути записаний або зчитаний незалежно від інших. Але для більшості прикладних програм представляє інтерес не звернення до окремих секторів, а можливість звернення до файлів, які

можуть займати довільну, в тому числі і не цілу, кількість секторів.

Для полегшення звернення до файлів і впорядкування використання простору секторів диска до складу будь-якої операційної системи входить файлова система, тісно пов'язана з логічною структурою диска. Логічна структура диска організована як система розділів. Це один з найбільш важливих елементів дискової підсистеми. Її стандарт не залежить від файлових і операційних систем.

Жорсткий диск розбивається на декілька ***розділів***, які використовуються різними ***ОС*** (максимальна кількість розділів – 4):

I – ***первинний розділ DOS*** (системний логічний диск С);

II, III – ***розділи не-DOS*** (наприклад, ***Linux, Unix*** тощо). Ці розділи ***DOS*** не доступні;

IV – ***розширений розділ DOS*** (логічні диски).

Створення більше одного розділу має такі переваги:

– відділення операційної системи (ОС) і програмних файли від файлів користувача. Це дозволяє робити резервні копії (клони) тільки операційної системи і встановленого програмного забезпечення;

– один фізичний диск може вміщувати декілька різних ОС, розташованих в різних розділах диску. Під час початкового завантаження можна вказати розділ диску, з якого повинна завантажуватися ОС;

– захист або ізоляція файлів, щоб зробити простіше та швидше відновлювання пошкодженої файлової системи або установку операційної системи. Якщо один розділ пошкоджений, інші файлові системи можуть бути не порушені;

– можливість організації контрольованого доступу до окремих розділів різним групам користувачів тощо.

Операційна система представляє зовнішню пам'ять у виді набору ***логічних дисків*** (logical drive). *Логічний диск* — це сукупність секторів з номерами, що послідовно наростають.

Логічний дисковий простір будь якого логічного диска поділяється на дві області: ***системну*** та ***даних***.

Системна область створюється та ініціалізується при форматуванні, а в подальшому обновлюється при маніпулюванні файловою структурою.

Область даних містить файли та каталоги, які підпорядковані кореневому. На відміну від системної доступна через інтерфейс користувача. Системна область логічного диску включає такі

компоненти:

– завантажувальній запис операційної системи (***Boot Record, BR***);

– таблиця розміщення файлів (***File Allocation Table, FAT***);

– кореневий каталог (***Root Directory, RDir***).

Boot Record містить:

– програму завантаження операційної системи (***System Bootstrap, SB***);

– блок параметрів диску ***(Disk Parameter Block, DPB***).

SB завантажує систему, якщо диск є системним.

DPB містить інформацію о кількості та розмірі секторів, розмірі кластера, розмірі FAT тощо, тобто служить для ідентифікації фізичних та логічних форматів логічного диску.

Сектори диску об'єднуються в більш крупні структури – ***кластери***.

Кластер – це група суміжних секторів, яка є мінімальною одиницею дискової пам'яті, котра виділяється файлу (або некореневому каталогу).

При запису файла на диск він розбивається на групу кластерів, які можуть розташовуватися, де завгодно на диску. Послідовність при організації файла не потребується, оскільки програма бачить тільки весь файл цілком

Кожний з файлів займає цілу кількість кластерів. При цьому останній кластер може бути задіяний не повністю.

File Allocation Table представляє собою карту (образ) області даних, в якій описується стан кожного кластера та зв'язуються у ланцюжок кластери, які належать одному файлу (некореневому каталогу).

Кількість чарунок FAT відповідає числу кластерів на диску. Кожна клітинка таблиці зберігається номер кластера. У каталозі зберігаються записи про файли, в яких поряд з іншими характеристиками файлу вказано номер його першого кластера К1. Тому при організації доступу до файлу спочатку звертаються до чарунки FAT, номер якої зазначений в записі каталогу.

В чарунці з номером першого кластера К1 файлу зберігається

номер другого кластера К2, в осередку з номером другого кластера К2 файлу зберігається номер третього кластера К3 тощо. Таким чином, створюється ланцюжок кластерів, які займає файл (рис. 2.13). До чарунки з номером останнього кластера файлу заноситься код FFF або FFFF для вказівки кінця ланцюжка.

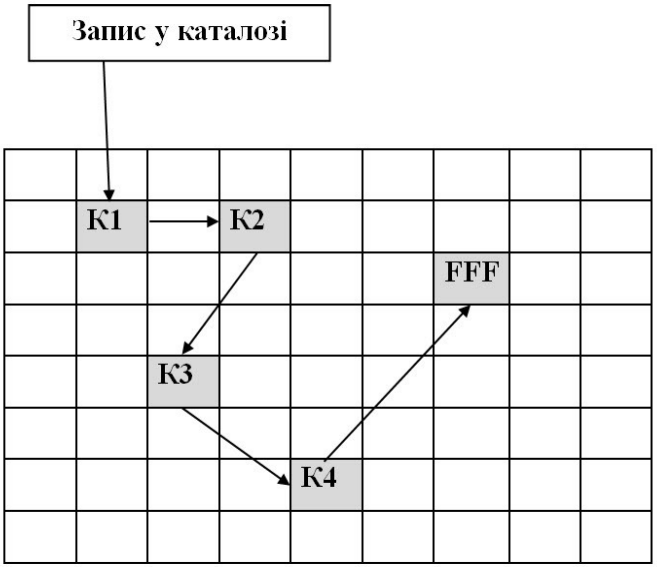


Рисунок 2.13 – Організація таблиці розміщення файлів

Кластери ланцюжка можуть перебувати в різних місцях диска, вільних на момент запису файлу на диск. У цьому випадку говорять, що файл фрагментований, так як зберігається на диску у вигляді окремих фрагментів. Процедура дефрагментації файлів, що виконується за допомогою спеціальних програм, дозволяє розмістити файли в сусідніх кластерах, що скорочує час доступу до кожного файлу.

Логічне розбиття області даних на кластери замість використання одиничних секторів має наступний сенс:

– зменшується можлива фрагментація файлів;

– зменшується розмір FAT, а значить, і обсяг системної області

логічного диска;

–прискорюється доступ до файлу, тобто в декілька разів скорочується довжина ланцюжків фрагментів дискового простору.

Але дуже великий розмір кластера веде до неефективного використання області даних, особливо у випадку великої кількості маленьких файлів.

З метою безпеки ***FAT*** зберігається у двох ідентичних примірниках. Оновлюються копії одночасно. Використовується тільки перший примірник. У випадку його пошкодження звернення йде до іншого примірника.

Кореневий каталог (***Root Directory, RD***) містить інформацію про файл, в том числі номер першого кластера кожного файла. ***RD*** розташовується на диску зразу після ***FAT***. Розмір каталога залежить від формату диска.

Підкаталоги розташовуються у просторі даних диска так само, як і звичайні файли. Розмір підкаталогу, як і звичайного файла не обмежений.

Головний завантажувальний запис (***Master Boot Record, MBR***) – код і дані, необхідні для подальшого завантаження операційної системи і розташовані в перших фізичних секторах (найчастіше в найпершому) на жорсткому диску або іншому пристрої для збереження інформації.

Функція MBR – «перехід» в той розділ жорсткого диска, з якого слід завантажувати операційну систему.

MBR містить позасистемний завантажувач (***NonSystem Bootstrap, NSB***) та таблицю розділів (***Partition Table***). Таким чином, в стартовому секторі фізичного жорсткого диска знаходиться не ***BR***, а ***MBR***.

NSB здійснює завантажування за фіксованою адресою в оперативній пам'яті системного завантажувача (System Bootstrap) із завантажувального запису логічного диска в активному розділі та передачу йому керування. Завантажувальний запис продовжує процес завантаження операційної системи.

Таблиця розділів – це 64-байтовая структура, яка використовується для визначення типу та місця розташування розділів на жорсткому диску. Вміст цієї структури уніфікований і не залежить від операційної системи. Інформація про кожний розділ займає 16 байт – таким чином, на диску може бути не більше чотирьох розділів.

2.4.8 Твердотільні накопичувачі

SSD (solid state-drive) або твердотільний накопичувач – запам'ятовуючий пристрій відносно нового типу, який працює на основі використання мікросхем пам'яті та контролера керування ними й не містить рухомих частин на відміну від HDD.

Розрізняють два види твердотільних накопичувачів: SSD на основі пам’яті, подібної до оперативної пам’яті комп’ютерів, і SSD на основі флеш-пам’яті.

Цей тип пристроїв порівняно з HDD має ряд переваг: відсутність будь-якої вібрації та шуму, низьке енергоспоживання, більш висока швидкість роботи при невеликих розмірах, стійкість до температурних коливань і механічного впливу тощо.

Найбільшими недоліками SSD є їх висока вартість і швидкість зношування (зазвичай близько 10 тис. циклів перезапису, у більш дорогих виробах – до 100 тис.). Останнє обов'язково повинне враховуватися при їх експлуатації. Не рекомендується проводити дефрагментацію таких носіїв (це ніяк не прискорить пошук інформації), розміщувати на них файл підкачки, а також здійснювати інші дії, пов'язані з їх «невиправданим» використанням.

Нині твердотільні накопичувачі використовуються в компактних пристроях: ноутбуках, нетбуках, комунікаторах і смартфонах. Деякі відомі виробники переорієнтувались на випуск твердотільних накопичувачів вже повністю. Наприклад Samsung продав бізнес з виробництва жорстких дисків компанії Seagate.

Існують і так звані гібридні тверді диски, що поєднують в одному пристрої як HDD, в якості основного, так і відносно невеликого обсягу SSD (4-8 ГБ) в ролі кешу (для збільшення продуктивності, швидкого холодного запуску системи, зниження енергоспоживання). Такі диски використовуються, в основному, в переносних пристроях (ноутбуках) і там, де продуктивність має більше значення, ніж ціна.

2.5 Організація зв'язків між функціональними вузлами комп'ютера

2.5.1 Шини системи

Для забезпечення роботи комп'ютера його функціональні вузли повинні бути відповідним чином з'єднані. Обмін інформацією між вузлами комп'ютера проводиться за допомогою шин.

Шина – це сукупність ліній передачі інформації, які об'єднані за функціональним призначенням.

Магістраль – це сукупність усіх шин ***МПС***. Часто терміни *"шина" та "магістраль"* використовуються як синоніми.

Всі модулі системи постійно підключені до магістралі, але в кожний момент часу обмін інформацією може відбуватися тільки між двома модулями за рахунок використання спеціальних буферних пристроїв з трьома стійкими станами.

Найважливішою характеристикою шини є її розрядність, яка визначає кількість даних, переданих по шині одночасно (за один такт). Зрозуміло, що чим більше розрядність шини, тим більше її продуктивність.

За призначенням шини можна розділити на три категорії:

- шина даних (DB – Data Bus);
- шина адреси (AD – Address Bus);
- шина управління (Control Bus).

Шина даних

По цій шині відбувається обмін даними між процесором, картами розширення та пам'яттю. Ця шина даних завжди двоспрямована, оскільки припускає передачу інформації в обох напрямах. Розрядність шини даних може становити 8 біт, 16 біт, 32 біта тощо. Розрядність шини даних визначає й розрядність всієї магістралі. Наприклад, коли говорять про 32-розрядну системну магістраль, мається на увазі, що вона має 32-розрядну шину даних.

Шина адреси

Ця шина визначає допустимий об'єм пам'яті, тобто максимально можливий розмір програми та максимально можливий об'єм даних, які запам'ятовуються. Кількість адрес, які підтримуються розрядністю

шини адреси, визначається як 2^N, де N – кількість розрядів. Наприклад, 16-розрядна шина адреси забезпечує 65 536 адрес, тобто 64 Кб, 20 розрядів – 1Мб, 32 розряди – 4 Гб. Шина адреси може бути односпрямованою (коли магістраллю завжди управляє тільки процесор) або двоспрямованою (коли процесор може тимчасово передавати управління магістраллю іншому пристрою, наприклад, контролеру ПДП).

Для зменшення загальної кількості ліній зв'язку часто застосовується мультиплексування шин адреси та даних. Тобто одні й ті ж лінії зв'язку використовуються в різні моменти часу для передачі як адреси, так і даних (на початку циклу – адреса, в кінці циклу – дані).

Зрозуміло, що мультиплексування шин адреси/даних забезпечує меншу швидкість обміну, вимагає більш тривалого циклу обміну.

Шина управління

Ця шина є допоміжною. Сигнали керування на ній визначають тип поточного циклу та фіксують моменти часу, відповідні різним частинам або стадіям циклу. Крім того, сигнали керування забезпечують узгодження роботи процесора (або іншого господаря магістралі, задавача) з роботою пам'яті або пристрою вводу/виводу (пристрою-виконавця). Сигнали керування також обслуговують запит і надання переривань, запит і надання прямого доступу.

Пристрої, що підключаються до шини, розділяються на два основних типи: ***bus masters (BM)*** і ***bus slaves (BS)***. Bus masters – це пристрої, здатні керувати роботою шини, тобто ініціювати запис / читання тощо. Bus slaves – це, відповідно, пристрої, які можуть тільки відповідати на запити.

2.5.2 Поняття інтерфейсу

Інтерфейс – сукупність засобів і методів взаємодії між елементами системи, у яких всі фізичні та логічні параметри погоджуються між собою. Крім того, під інтерфейсом розуміють і правила (протокол) взаємодії цих елементів. Термін «інтерфейс» зазвичай трактується як синонім слова «сполучення».

Поняття інтерфейсу використається практично у всіх галузях науки й техніки. Його значення ставиться до будь-якого сполучення

взаємодіючих сутностей. Якщо інтерфейс є загальноприйнятим, наприклад, затвердженим на рівні міжнародних угод, то він називається стандартним.

В обчислювальній техніці під ***стандартним інтерфейсом*** розуміється сукупність уніфікованих апаратних, програмних і конструктивних засобів, необхідних для реалізації взаємодії окремих функціональних елементів (ФЕ).

Основним призначенням інтерфейсу є уніфікація внутрисистемних та міжсистемних зв'язків і пристроїв спряження. Основні функції інтерфейсу містяться в забезпеченні інформаційної, електричної та конструктивної сумісності між ФЕ системи.

Інформаційна сумісність - це узгодження взаємодій ФЕ системи у відповідності зі сукупністю логічних умов.

Логічні умови визначають: структуру та склад шин; засіб кодування і формати інформації; часові співвідношення між керуючими сигналами. Логічні умови визначають функціональну та структурну організацію інтерфейсу.

Електрична сумісність - це узгодження параметрів електричних сигналів, тобто рівні напруги (струму), тривалість імпульсів.

Конструктивна сумісність - це узгодження конструктивних елементів інтерфейсу: типи та геометрія роз'ємних з'єднань, кількість і розташування контактів тощо.

Кожен з функціональних елементів (пам'ять, монітор або інший пристрій) пов'язаний з шиною певного типа – адресною, управляючою або шиною даних. Для узгодження інтерфейсів периферійні пристрої підключаються до шини не безпосередньо, а через свої контролери (адаптери) і порти (рис. 3.4):

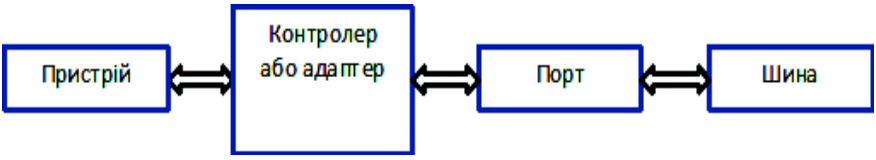


Рисунок 2.13 – Схема підключення приладу до шини

Контролери *(адаптерами)* – це набори електронних ланцюгів,

якими забезпечуються пристрої комп'ютера з метою сумісності їх інтерфейсів. Контролери, окрім цього, здійснюють безпосереднє управління периферійними пристроями за запитами мікропроцесора.

Портами – це електронні пристрої, що містять один або декілька регістрів вводу/виводу і дозволяють підключати периферійні пристрої комп'ютера до зовнішніх шин мікропроцесора.

Портами також називають пристрої стандартного інтерфейсу: послідовний, паралельний та ігровий порти (або інтерфейси). Послідовний порт обмінюється даними з процесором побайтно, а із зовнішніми пристроями – побітно. Паралельний порт отримує та посилає дані побайтно.

До послідовного порту зазвичай під'єднують ті пристрої, що повільно діють або віддалені пристрої, такі, як миша і модем. До паралельного порту під'єднують «швидкі» пристрої – принтер і сканер. Через ігровий порт під'єднується джойстик. Клавіатура і монітор підключаються до своїх спеціалізованих портів.

Основні електронні компоненти, що визначають архітектуру процесора, розміщуються на основній платі комп'ютера, яка називається системною або *материнською* (Motherboard). Контролери та адаптери додаткових пристроїв, або самі ці пристрої, виконуються у вигляді плат розширення (Daughterboard – дочірня плата) і підключаються до шини за допомогою роз'ємів розширення, званих також слотами розширення (англ. slot – щілина, паз).

2.5.3 Апаратний склад інтерфейсу

Шинні формувачі

Як відомо, навантажувальна здатність виходів МП обмежена. Тому щоб уникнути струмового перевантаження мікропроцесора проводять буферизацію шин, що дозволяє суттєво збільшити їх навантажувальну здатність. Особливо часто така ситуація виникає, якщо система, що розробляється, має великі розміри та утворює значне ємкісне навантаження із-за кабелів великої довжини.

Для підвищення навантажувальної здатності виводів МП використовуються спеціальні шинні посилювачі (шинні формувачі), які представляють собою 8-розрядні паралельні приймально-передавачі з трьома стійкими станами (рис. 2.14).

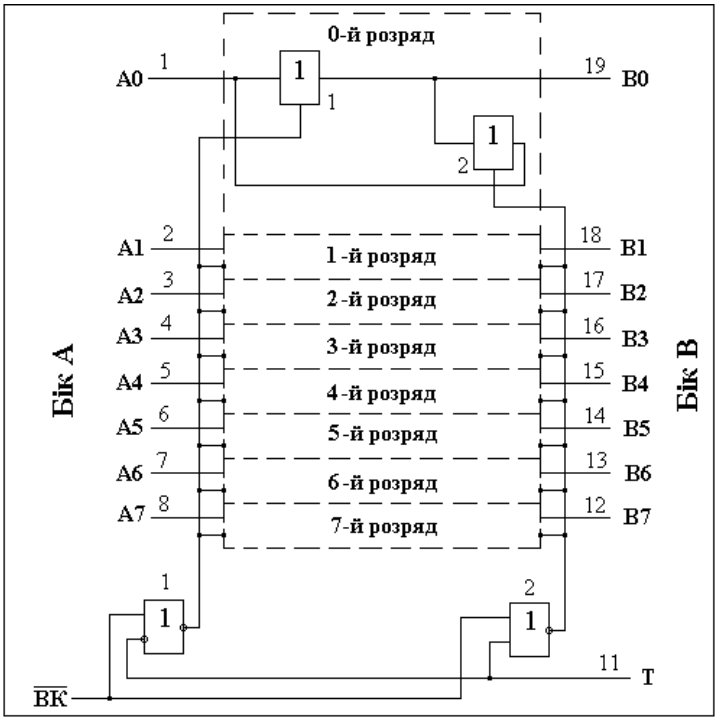


Рисунок 2.14 - Шинний формувач

Кожний прийомопередавач складається з двох повторювачів, які включені назустріч один одному та мають три стійких стану. При цьому якщо один з повторювачів знаходиться в увімкнутому стані, другий буде знаходитися у вимкненому (третьому) стані. Таким чином, передача інформації може здійснюватися тільки в одному напрямку: або від виводів ***A*** до ***B***, або від ***B*** до ***A.***

Вхід ***T*** - вхід управління напрямком передачі, ***ВК*** - вхід дозволу на передачу, виводи ***A*** та ***B*** - інформаційні. До виводів ***A*** підключаються виходи МП, а до виводів ***B*** - системні шини адреси або даних.

Якщо на вхід ***ВК*** подається сигнал високого рівня (наприклад сигнал "***Підтвердження Захоплення***"), то всі інформаційні входи та виходи мікросхеми переходять в високоомний стан, та МП відключається від системної шини. При подачі на цей вхід сигналу низького рівня відбувається передача даних у напрямку, який

визначається сигналом на вході ***T*** (наприклад, "***Запис***"): при ***0*** - від ***B*** до ***A***, ***1*** - від ***A*** до ***B***.

Порти вводу-виводу

Портом вводу виводу називається комплекс засобів, презначений для підключення периферійних пристроїв до мікро-ЕОМ. Порти вводу-виводу виконують наступні функції:

– вибір потрібного модуля системи, тобто селекцію необхідного зовнішнього пристрою;

– підключення обраного модуля до шини даних по відповідному сигналу або відключення від неї (роль буфера);

– зберігання рівнів бінарних сигналів під час їх читання або запису незалежно від тривалості цих операцій, тобто роль фіксатора рівнів сигналів.

У найпростішому випадку порт вводу-виводу є комбінацією шинного формувача та дешифратора (рисунок 3.8), у якій дешифратор служить для селекції необхідного зовнішнього пристрою. При цьому на виході дешифратора сигнал з'являється тільки при наявності на його вході суворо визначеного коду, відповідного конкретному ПВВ.

Крім розшифрування коду зовнішнього пристрою дешифратор також повинен розшифровувати сигнал обраного ПВВ ("***Читання***" або "***Запис***").

У відзнаку від пам'яті зовнішні пристрої не мають безпосереднього контакту з системними шинами адреси та даних. Зв'язок ***ПВВ*** з шинами відбувається через порти вводу-виводу.

Код вибору зовнішнього пристрою міститься в команді вводу-виводу та по ША передається на усі контролери периферійних пристроїв. Кожний з них має свій власний "замок", який називається селектором пристрою. Код вибірки представляє собою своєрідний ключ до конкретного пристрою.

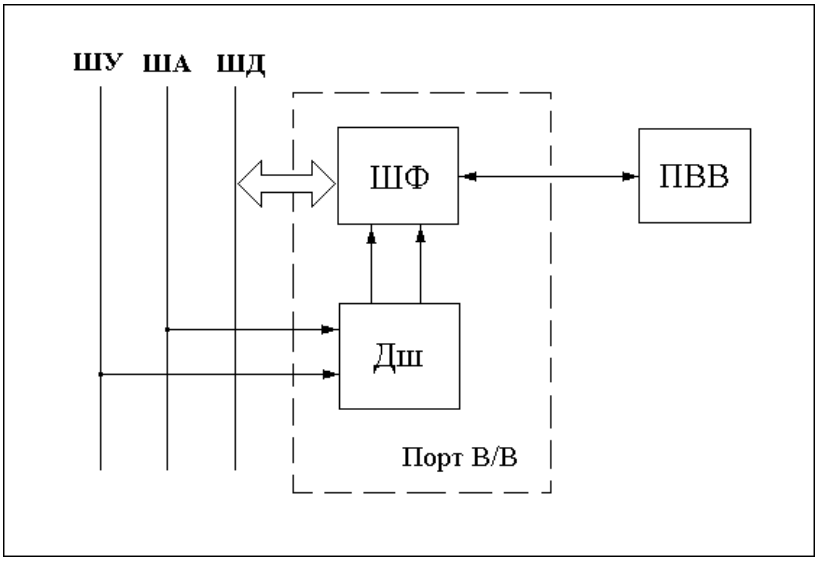


Рисунок 2.15 - Порт вводу-виводу

З контролера вводу-виводу сигнал передається на вхід "Вибір модулю" (***ВМ***) відповідного порту вводу або виводу. При наявності такого сигналу до ***МП*** у процесі роботи системи може підключатися тільки один периферійний пристрій. Іншими словами, у кожний момент часу у системі є тільки один модуль, який бере участь у обміні даними. Входи та виходи усіх інших периферійних пристроїв при цьому лишаються у високоімпедансному стані, тобто практично відключеними від системних шин.

2.6 Режими обміну інформацією

Між МП та периферійними пристроями (**ПП**) відбувається обмін інформацією двох типів: службовою та власно даними. Службова інформація від МП ініціює дії, пов’язані з обміном даних, та передається за допомогою управляючих слів ***CW (Control Word)***.

Службові повідомлення від ПП, які інформують систему про

поточний стан ПП, мають назву ***слів стану SW (Status Word)***.

Дані передаються за допомогою ***слів даних DW (Data Word)***.

Обсяг службової інформації, якою обмінюються ПП та МПС, а також її інтерпретація залежать від типу ПП. Для найбільш простих пристроїв (наприклад, світлодіодних матриць) службова інформація не потрібна. Для інших пристроїв, наприклад НГМД, керуюча інформація та дані про стан ПП можуть мати значний обсяг.

2.6.1 Протоколи обміну. Арбітраж

Під ***протоколом обміну*** розуміється сукупність правил, які встановлюють єдині принципи взаємодії підсистем. Протокол є основою для створення *драйвера* періферійного пристрою. Драйвер являє собою набір підпрограм, що обслуговують обмін ПП з мікропроцесором.

Основними протоколами обміну є:

– ***синхронний*** (тактуємий) обмін, при якому передача інформації по магістралям МПС здійснюється за один суворо заданий по тривалості проміжок часу (такт);

– ***асинхронний –*** нетактуємий обмін;

– ***напівсинхронний,*** при якому сам процес обміну тактується в часі, але займає декілька тактів.

Задавач - пристрій, який керує роботою магістралей при роботі з іншим пристроєм - виконавцем.

Виконавець - пристрій, який керується задавачем при обміні інформацією.

<u>Синхронний обмін</u>

Обмін інформацією між задавачем та виконавцем відбувається на інтервалі дії імпульсу синхронізації.

Перевага синхронного обміну міститься в простоті його реалізації та максимальній швидкодії. Але у випадку наявності в МПС виконавців з різною швидкодією, мінімальну тактову частоту синхроімпульсів необхідно знижувати, сходячи з надійного

забезпечення обміну інформацією з самим повільним виконавцем.

Асинхронний обмін

При наявності в МПС виконавців з різною швидкодією оптимальним є режим, при якому обмін з кожним виконавцем здійснюється з тією швидкістю, на яку він розрахований. В цьому випадку виконавці отримують магістралі МПС на необхідний їм час.

Арбітраж

Арбітр - схема, яка керує у відповідності з деяким пріоритетом черговостю захоплення задавачами управління магістралями.

Арбітраж - процедура розгляду запитів задавачів на управління магістралями.

Основна функція арбітражу міститься у розподілі часу користування магістралями МПС між задавачами з метою усунення конфлікту, пов'язаного з участю у обміні декількох задатчиків та виконавців. Якщо в системі є один задавач, необхідність в арбітражі відпадає.

2.6.2 Програмний обмін інформацією

Даний режим характеризується тим, що всі дії по обміну даними реалізуються командами прикладної програми. Найбільш простими ці дії виявляються для "завжди готових" периферійних пристроїв, наприклад, індикаторів на світлодіодах.

При необхідності обміну у відповідному місті програми використовуються команди ***IN*** або ***OUT***. Така передача даних називається ***синхронним***, прямим або безумовним обміном (рисунок 2.16,а).

Для більшості периферійних пристроїв до виконання операції необхідно переконатися в їх готовності до цих операцій, тобто обмін стає ***асинхронним.***

Загальний стан периферійних пристроїв характеризуються прапорцем готовності ***READY,*** який входить до слова стану цього пристрою. МП перевіряє прапорець готовності, та якщо він

встановлений, ініціює власно ввід або вивід даних.

Існує два типи умовного обміну: з заняттям циклу (рисунок 2.16,б) та сполучений (рисунок 2.16,в). У першому випадку МП зависає на циклі чекання готовності, витрачаючи на це весь машинний час. В другому випадку, якщо ПП не готовий до обміну, МП повертається до основної задачі без виконання операції ВВ.

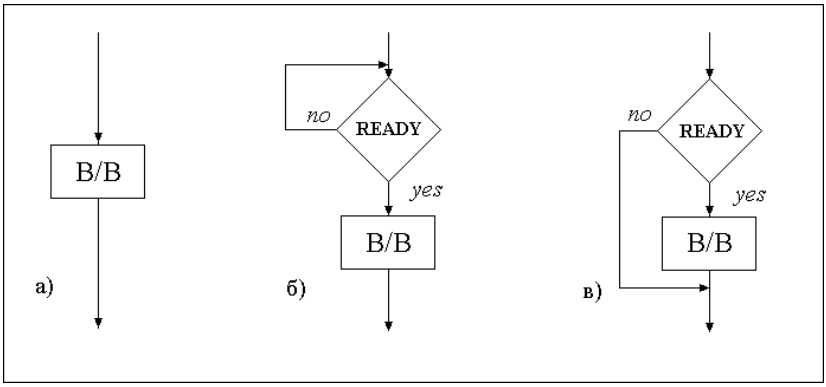


Рисунок 2.16 - Програмний ввід/вивід

Після завершення операції обміну сигнал готовності ***READY*** повинен бути знятий та виставлений знову тільки при новій готовності до обміну. З цією метою периферійний пристрій слід проінформувати про закінчення операції, для чого використовується включений до одного з управляючих слів ***CW*** сигнал підтвердження ***ACK(Acknowledgment).***

Протокол обміну службовою інформацією такого типу має назву ***квитування***. Він забезпечує надійну асинхронну передачу даних зі швидкостями, які визначає ПП.

Основний недолік програмного обміну пов’язаний з непродуктивними витратами часу процесора в циклах чекання, в яких він не виконує ніякої корисної праці.

Перевагою програмного ВВ є простота його реалізації, яка не потребує додаткових апаратних витрат.

2.6.3 Обмін інформацією в режимі переривань

Більш ефективним є обмін даними з перериванням програми. ***Переривання*** **(***interrupt)* **–** тимчасове припинення виконання основної програми з метою реалізації більш термінових на поточний момент дій (навіть безпосередньо не пов'язаних з основною програмою), що відбувається в довільний момент часу за запитом зовнішнього пристрою (ЗП) чи у випадку особливої ситуації в процесорі.

Система обробки переривань – це комплекс апаратних засобів у складі обчислювальної системи та програмно-логічних принципів для обробки асинхронних подій.

Сигнал переривання повідомляє процесор про настання події, яка потребує невідкладної уваги. При цьому виконання поточної послідовності команд призупиняється і керування передається ***обробнику переривання***, який реагує на подію та обслуговує її, після чого повертає управління перерваний програмі.

Залежно від джерела виникнення сигналу переривання поділяються на:

– асинхронні або зовнішні (апаратні) – події, що створені зовнішніми джерелами (наприклад, периферійними пристроями) та можуть відбутися в довільний момент: сигнал від таймера, мережевої карти або дискового накопичувача, натискання клавіш клавіатури, рух миші;

– синхронні або внутрішні – виняткові ситуації у самому процесорі як результат порушення якихось умов при виконанні машинного коду: ділення на нуль або переповнення, звернення до неприпустимих адрес або неприпустимий код операції;

– програмні (частковий випадок внутрішнього переривання) — ініціюються виконанням спеціальної інструкції в коді програми. Програмні переривання, як правило використовуються для звернення до функцій вбудованого програмного забезпечення (firmware), драйверів і операційної системи.

Сигнал запиту переривання ***INT*** (*Interrupt*) з'являється в довільні моменти часу, асинхронно до дій МП (рис. 2.18). Відповідно, керувати його появою програма не може. Тому, реагуючи на сигнал ***INT***, МП

після виконання поточної команди повинен припинити виконання основної програми та виставити сигнал ***INTE*** («Дозвіл переривання»). При цьому вміст програмного лічильника, ***PSW***, а також усіх або частини ***РЗП*** записуються до ***ОЗП***. Після цього виконується відповідна підпрограма обробки переривання . По закінченні виконання процедури повинно бути реалізовано повернення до перерваної програми. Цей режим називають також апаратним викликом підпрограм

Важливим є те, що вся робота, як і у випадку програмного режиму, здійснюється самим процесором, зовнішня подія просто тимчасово відволікає його від виконання програми. Реакція на зовнішню подію по перериванню, в загальному випадку, повільніша, ніж при програмному режимі. Як і у випадку програмного обміну, тут усі сигнали на магістралі виставляються процесором, тобто він цілком контролює магістраль.

Для обслуговування переривань у систему вводиться спеціальний модуль контролера переривань, але він в обміні інформацією участі не бере. Його завдання полягає в тому, щоб спростити роботу процесора з зовнішніми запитами переривань. Цей контролер зазвичай програмно керується процесором за допомогою системної магістралі.

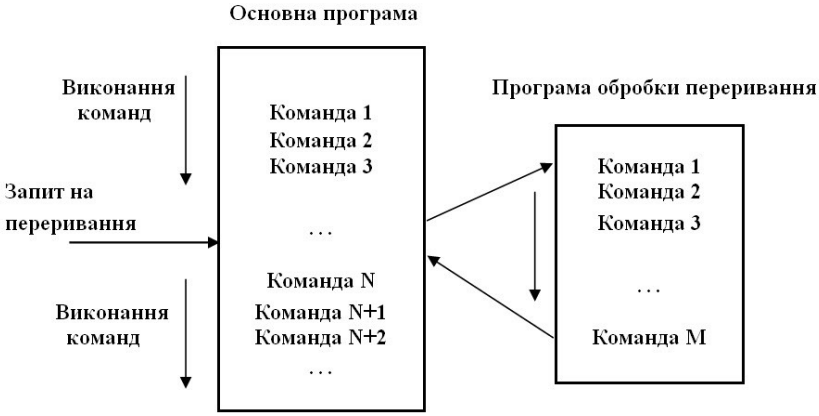


Рисунок 2.18 – Схема обслуговування переривання

Природно, ніякого прискорення роботи системи переривання не дає. Його застосування дозволяє тільки відмовитися від постійного опитування прапорця зовнішньої події та тимчасово, до її настання, зайняти процесор виконанням якихось інших задач.

У випадку одночасної подачі запитів на переривання від кількох зовнішніх пристроїв вступає в силу система пріоритетів, яка дозволяє встановити черговість їх обслуговування. Найбільш часто це вирішується двома способами: послідовним опитом (система ***полінга***) або перериванням за вектором (***система пріоритетних переривань***), які також називаються, відповідно, *програмним* та *апаратним* способами ідентифікації джерела переривання

Програмний спосіб базується на використанні головної програми обробки переривань, яка після надходження запиту на переривання виконує послідовне програмне опитування кожного зовнішнього пристрою до тих пір, поки не виявить пристрій, який надіслав запит. Такий спосіб ідентифікації ЗП називають *опитуванням* або *полінгом* (Polling). Зазвичай він використовується при об'єднанні запитів від декількох джерел за допомогою елемента АБО і передачі їх по одній лінії на вхід процесора.

Після надходження сигналу запиту на переривання процесор завершує поточний цикл виконання команди та виробляє сигнал дозволу переривання, який надходить через системний контролер послідовно до всіх зовнішніх пристроїв. Цей сигнал відсилається спочатку до пристрою з найвищим пріоритетом. Якщо цей пристрій не запитував переривання, то сигнал передається наступному пристрою (з більш низьким пріоритетом). Якщо ж пристрій запитував переривання, то він перериває подальше поширення сигналу.

Полінг доцільно використовувати при невеликому числі джерел переривання. Її недолік – затримка початку обслуговування ЗП, пов'язана з опитуванням.

При використанні другого способу перехід на обслуговування переривання йде безпосередньо по запиту конкретного зовнішнього пристрою, який передає МП також інформацію про свою адресу (адресу відповідної підпрограми), в наслідок чого відпадає необхідність опиту інших ЗП.

Система пріоритетних переривань дозволяє ідентифікувати відповідний ЗП за допомогою сукупності апаратних засобів, які при одночасному надходженні кількох запитів надають обслуговування

тільки одному пристрою, який має старший пріоритет. Завжди самий вищий пріоритет присвоюється найменш швидкодіючому пристрою.

У випадку, якщо деяка програма не може бути перервана з будь-яких міркувань, використовується особливий стан системи, який має назву ***маскування***. Запити на переривання, ініційовані замаскованим пристроєм, не приймаються системою.

Замаскувати можливо всі без виключення запити на переривання або тільки запити, які мають пріоритет не вище завданого. Зокрема при виконанні програми, початої на основі запита на переривання з деяким пріоритетом, усі запити на переривання, які мають більш низький пріоритет, маскуються автоматично.

Обробники переривань зазвичай пишуться таким чином, щоб час їх обробки був якомога меншим, оскільки під час їх роботи можуть не оброблятися інші переривання, а якщо їх буде багато (особливо від одного джерела), то вони можуть губитися.

2.6.4 Прямий доступ до пам'яті

Більшість пристроїв комп'ютера потребують періодичний обмін даними не тільки з центральним процесором, а й з оперативною пам'яттю. Звичайний обмін даними між зовнішньою та основною пам'яттю потребує спочатку прийняти дані від джерела до процесора, а потім видати їх з процесора приймачеві, тобто реалізується за два командних цикли.

У разі необхідності передачі великих масивів інформації доцільно робити обмін даними між зовнішніми ЗП та ОЗП безпосередньо, без використання робочих регістрів МП, зокрема акумулятора.

Такий режим роботи отримав назву прямого доступу до пам'яті (ПДП) (англ. Direct Memory Access, DMA). ПДП реалізується за допомогою спеціальних апаратних засобів, які мають назву контролера ПДП.

Контролер ПДП (КПДП) здійснює повноцінний обмін по системній магістралі без жодної участі процесора. Причому процесор заздалегідь повинен повідомити КПДП, звідки йому слід брати інформацію та/або куди її слід направляти. КПДП може вважатися спеціалізованим процесором, який відрізняється тим, що сам не бере участь в обміні, тобто не приймає інформацію та не видає її (рис. 2.19).

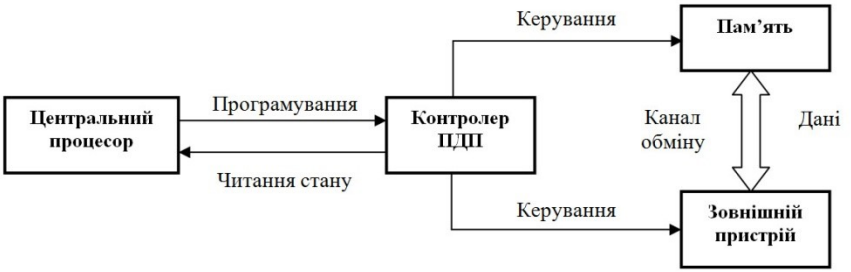


Рисунок 2.19 – Інформаційні потоки в режимі ПДП

Для прямого доступу до пам'яті управління шинами МПС передається ЗП, який і виконує захоплення шин, призупиняючи при цьому виконання програми МП. Після отримання сигналу «Захоплення шин» МП, закінчує виконання поточного машинного циклу, припиняє виконання програми, відключається від системних шин і встановлює сигнал логічної одиниці на лінії «Підтвердження захоплення».

Отримавши цей сигнал, ЗП здійснює обмін з пам'яттю. По завершенні обміну ЗП встановлює на лінії "Захоплення шин" логічний нуль, і управління шинами передається МП.

При виконанні ПДП вміст внутрішніх регістрів МП не модифікується, тому його не треба запам'ятовувати в пам'яті; а потім відновлювати, як при обробці переривань. Зазвичай передача даних у режимі ПДП має пріоритет перед іншими видами обміну.

Інтерфейс каналу ПДП по складності перевищує інші типи інтерфейсів. Хоча його структура може бути спрощена за рахунок програмної реалізації частини функцій, це не може бути доцільним, тому що основна ціль використання режиму ПДП – забезпечення вищої швидкості передачі даних.

3 ПРОГРАМУВАННЯ МІКРОПРОЦЕСОРНИХ СИСТЕМ

3.1 Елементарні засоби програмування

Нижче приведені приклади застосування окремих команд та складання невеликих програм.

Приклад 1. Результат дії операції **DCR A**, якщо **(A)=00H**, складає **(A)=FFH**.

Приклад 2. Для якої мети можна використовувати команду **XRA**?
Відповідь :
а) для установки акумулятора в нульовий стан **(XRA A)** ;
б) для інвертування вмісту регістру **(MVI A, FFH; XRA B)**.

***Приклад 3*.** Встановити прапорці регістру ознаки згідно із значенням числа, яке зберігається у чарунці **0456H:**

```
LDA A, 0456H
ANA A
```

***Приклад 4*.** Отримати додатковий код числа, яке зберігається в акумуляторі.

```
CMA
INR A
```

Приклад 5. Перевести число 13 у двійково-десятичну форму

```
MVI A,13
DAA
```

Приклад 6. Обчислити арифметичний вираз (-17+47) за допомогою додаткового коду.

```
MVI A,10001B
CMA
INR A
ADI 101111B
```

Приклад 7. Замінити число, яке зберігається у чарунці з адресою **0200H**, на те ж саме число у допоміжному коді :

```
LXI H,0200H
MOV A,M
CMA
INR A
MOV M,A
```

Приклад 8. Обнулити 0-й та 5-й розряди вмісту регістру **B :**

```
MOV A,B
ANI, 11011110B; ( ANI DEH)
MOV B,A
```

Приклад 9. Встановити в одиницю 2-й, 3-й та 7-й розряду вмісту регістру **D**:

```
MOV A,D
ORI, 01001100B; (  ANI, 4CH)
MOV D,A
```

Приклад 10. Перевірити стан нульового біту регістру **C**:

```
а) MVI A, 00000001B
   ANA C
```

Якщо нульовий розряд регістру **C** дорівнює нулю, то прапорець **Z** регістру ознак встановлюється в одиничний стан і до потрібного фрагменту програми можна перейти, наприклад, по команді **JZ**.

```
б) MOV A,C
   RAR
```

В результаті нульовий розряд буде циклічно зсунутий праворуч до тригеру С регістра ознак. Після цього його можна тестувати за допомогою команд умовних переходів **JC** або **JNC**.

Приклад 11. Змінити вміст чарунки пам'яті з адресою **0400H** у розряді **D5**:

```
LXI H,0400H
MVI A,00100000B
XRA M
MOV M,A
```

Приклад 12. Завантажити у регістр ознак (тригер **C**) значення розряду **D5** регістру **D**:

Спосіб 1:
```
MOV A,D
RLC
RLC
RLC
```

Спосіб 2:
```
MOV A,D
RRC 4
RRC 3
RRC 2
RRC1
RRC 0
RRC C
```

Приклад 13. Відняти вміст чарунки пам'ять з адресою **AF9CH** із вмісту чарунки пам'яті з адресою **B9FH** та здійснити перехід до чарунки пам'яті з адресою **ADR**, якщо результат виявиться від'ємним:

```
LXI H, AF9CH
LXI D, B9FH
MOV B,M
LDAX D
SUB B
JM ADR
```

Приклад 14. Прийняти інформацію з порту з адресою **PORT1** і записати її в порт з адресою **PORT2**:

```
IN PORT1
OUT PORT2
```

Приклад 15. Перевірити значення розряду **D3** порту **05H** та перейти до програми **A** у тому випадку, якщо **D3=0**, та до програми **B**, якщо **D3=1**:

```
IN 05H
ANI 08H     ; виділення розряду D3
CZ A
CALL B
```

Приклад 16. З`ясувати, чи є число, отримане з порту 00h, парним. Якщо так – вивести на порт 00h одиницю, інакше – нуль.

```
        IN 00H
        RAR
        JNC L1
        MVI A,0
        OUT 0
        HLT
L1:
        MVI A,1
        OUT 0
        HLT
```

Приклад 17. Порівняти число, отримане з порту 0 із значенням регістру D.

1. Якщо значення однакові — вивести на порт 1 число 1.
2. Якщо не однакові — вивести на порт 0 їх різницю.
3. Якщо отримано від'ємне число — вивести його модуль на порт 0.

```
        IN 0
        SUB D
        JZ L2
        JP L1
        CMA
        INR A
        OUT 0
        JMP L3
L1:     OUT 0
        JMP L3
L2:     MVI A,1
        OUT 1
L3:     продовження програми
```

3.2 Організація циклів

Цикл організується за наступною схемою :

	MVI r,N	; ініціалізація циклу, де **N** – кількість повторень тіла циклу
LABEL:	**...**	; тіло циклу
	DCR R	; зменшення лічильника на одиницю
	JNZ LABEL	; перевірка виходу із циклу

Якщо кількість повторень більше 256 D, належить організувати вкладені цикли або використати для організації лічильника регістрову пару **rp**.

Увага!!!

*В останньому випадку після команди **DCX rp** неможна одразу використати команду тестування регістру ознак, оскільки команда **DCX rp** не модифікує прапори.*

3.3 Програмна обробка масивів

Приклад 1. Маємо масив чисел з адресою 04FFH. Потрібно реалізувати програму складання цих чисел одне з одним до появи ознаки перенесення. Адреса останнього числа, яка приймає участь у складанні, повинна бути видана на порти з адресами 04H та 05H.

```
        LXI H,04FFH
        MOV A,M
AGAIN:  INX H
        ADD M
        JNC AGAIN
        MOV A,H
        OUT 04H
        MOV A,L
        OUT 05H
        HLT
```

Приклад 2. Маємо масив чисел з адресою 13ECH довжиною 25D. Підрахувати кількість парних елементів у масиві.

```
        LXI H,13ECH   ; завдання початкової адреси масиву
        MVI C,19H     ; ініціалізація циклу
        MVI B,0H      ; ініціалізація лічильника парних
                        елементів
LAB:    MOV A,M       ; вибірка числа із масива
        ANI 01H       ; виділення 0-го біту
        JNZ L1        ; перехід на вибір наступного числа
        INR B         ; збільшення лічильника парних
                        елементів
L1:     INX H         ; формування адреси наступного
                        елементу масиву
        DCR C         ; перевірка закінчення циклу
        JNZ LAB
        HLT
```

Приклад 3. Підрахувати кількість чисел у масиві (адрес першого елемента FFh, кількість елементів – 100), більших за отримане з порту 00h, та вивести результат на порт 00h:

```
        IN 0
        MVI B,0
        LXI H,00FFH
        MVI C,100
LABEL1:
        CMP M
        JNC LABEL2
        INR B
LABEL2:
        INX H
        DCR C
        JNZ LABEL1
        MOV A,B
        OUT 0
        HLT
```

Приклад 4. У масиві всі від'ємні елементи замінити їх модулями. Кількість елементів в масиві – 10, адреса першого елемента A100h

```
        MVI B,10
        LXI H,A100H
ITERATION:
        MOV A,M
        ANA A
        JP SKIP
        CMA
        INR A
        MOV M,A
SKIP:
        INX H
        DCR B
        JNZ ITERATION
        MOV A,B
        OUT 00H
        HLT
```

Приклад 5. Масив складається з 50 елементів. Підрахувати елементи масиву, що дорівнюють 37.

```
        MVI C,50
        MVI A,37
        MVI B,0
        LXI H,1000H
 ITERATION:
        CMP M
        JNZ SKIP
        INR B
SKIP:   INX H
        DCR C
        JNZ  ITERATION
        MOV A,B
        OUT 0
        HLT
```

Приклад 6. Масив складається з 50 елементів, перший з яких має адресу 32h. Скопіювати усі його додатні елементи в масив, що починається з адреси FAh. Кількість додатних елементів вивести на порт 0.

```
        MVI B,50
        LXI H,0032H
        LXI D,00FAH
        MVI C,0
S:
        MOV A,M
        ANA A
        JM GH
        INR C
        STAX D
        INX D
GH:
        INX H
        DCR B
        JNZ S
        MOV A,C
        OUT 0
        HLT
```

Приклад 7. Масив складається з 100 чисел. Перше знаходиться у чарунці пам'яті з адресою 100h. Скопіювати числа в масив, що починається з чарунки 200h, помножуючи непарні на 2

```
        MVI B,64H
        LXI H,100H
        LXI D,200H
S:
        MOV A,M
        ANA A
        RAR
        JC G;   перевірка парності
        RAL;    повернення попереднього значення числа
        RAL;    множення на 2
```

```
        STAX D
        INX D
G:
        INX H
        DCR B
        JNZ S
        HLT
```

Приклад 8. Масив складається з 70 елементів. Адреса першого елемента: 64h. Скласти молодші 4 розряди усіх додатних елементів, та отриману суму вивести на порт 0.

```
        MVI B,70
        LXI H,0064H
        MVI D,00H
START:
        MOV A,M
        ANA A
        JM NEXT
        ANI 00001111B
        ADD D
        MOV D,A
NEXT:
        INX H
        DCR B
        JNZ START
        MOV A,D
        OUT 0
        HLT
```

Приклад 9. В масиві замість його елементів записати значення їх молодших полубайтів. Значення старших полу байтів записати в інший масив. Перший масив починається з адреси 100h, другий масив – з адреси 200h, кількість елементів – 50.

```
        LXI H,100H
        LXI D,200H
```

```
        MVI B,50
LABEL1:
        MOV A,M
        MOV C,A
        ANI 11110000B
        RRC
        RRC
        RRC
        RRC
        STAX D
        MOV A,C
        ANI 00001111B
        MOV M,A
        INX H
        INX D
        DCR B
        JNZ LABEL1
        HLT
```

3.4 Робота з підпрограмами

Організація підпрограм ставить деякі проблеми використовування внутрішніх регістрів мікропроцесору. Правильно складена підпрограма не повинна змінювати жодного із регістрів, які будуть потрібні головній програмі після виконання підпрограми.

Таким чином, вміст регістрів, виконаних підпрограмою, необхідно тимчасово запам'ятати, тим, щоб після виконання підпорами його можна було відновити без змін. Найбільш зручною пам'ятаю для тимчасового запам'ятовування вмісту регістрів є стек. Запам'ятати вміст регістрів можна у головній програмі перед викликом підпрограми, або у самій підпрограмі.

Практика програмування показує, що доцільно завжди запам'ятовувати вміст регістрів на початку і відновити його наприкінці підпрограм.

Наприклад, якщо програма **SUBR** використовує регістри **A**, **B** і **C**, вона повинна мати такий вигляд :

```
SUBR:
PUSH PSW        ; Запам'ятовування вмісту регістрів до стеку
PUSH B
    ...         ; Команди підпрограми
    ...
    ...
POP B           ; Відновлення колишнього вмісту регістрів
POP PSW
RET             ; Повернення з підпрограми
```

У деяких підпрограмах зустрічаються команди умовного повернення. При явності команд **PUSH** і **POP** такі підпрограми повинні бути трохи модифіковані. Нехай, наприклад, заключний фрагмент підпрограми має такий вигляд :

```
...
        DCR A
        RZ
        INX B
        RET
```

Коли у підпрограмі вимагається тимчасове запам'ятовування та відновлення вмісту регістрів, цей фрагмент трохи змінюється:

```
SUBR:   PUSH PSW
        PUSH H
          ...
        DCR A
        JZ NOT
        INX B
NOT:POP H
        POP PSW
        RET
```

3.5 Організація переривань

Обслуговування вводу/виводу по перериванням є альтернативною програмно-керованому обміну. Якщо при чисто програмному керуванні як початок процедури, так і безпосередньо її виконання знаходиться під керуванням програми, то обслуговування по перериванням ініціюється апаратними засобами. Сукупність цих засобів, команд і програм їх обслуговування зветься системою переривань.

Процес обробки запиту на переривання значно подібний процесам виклику і повернення із підпрограм. Але у випадку переривань виклик здійснюється командою **CALL,** яка формується та підставляється у загальну команду послідовність за допомогою апаратури системи переривань. З цієї причини запити на переривання часто називають апаратними викликами підпрограм.

Усі запити на переривання можуть бути заборонені або дозволені одночасно за допомогою команд **DI** та **EI**, відповідно.

Для прискорення процедури обробки переривань застосовується спеціальний скорочений варіант команди **CALL adr m- RST n**, де **adr=8n, n=0-7.**

Використання команди **RST n** припускає резервування перших 64 (8×8) байтів пам'яті під таблицею входів у підпрограми обслуговування переривань (таблиця).

Звичайно за адресами **8n, n = 0-7**, розташовані команди **JMP**, передаючи керування на підпрограми обслуговування переривань. Модифікування адресної частини команд **JMP** дозволяє оперативно здійснювати входи у підпрограми.

Таблиця 2.1 - Розподіл адрес входів до підпрограм обслуговування переривань

Номер області ЗУ	Початкова адреса	Номер області ОЗУ	Початкова адреса
0	0000h	4	0020h
1	0008h	5	0028h
2	0010h	6	0030h
3	0018h	7	0038h

При виконанні команди **RST** також, як і у команді **CALL,**

адреса повернення запам'ятовується у стеку. Стан програмно-приступних регістрів може бути збережено у пам'яті, а потім відновлено безпосередньо перед поверненням у перервану програму. Цей процес називається контекстним перемиканням і виконується як програмними, так і апаратними засобами.

Повернення із підпрограм обслуговування переривань звичайно виконується за допомогою командної послідовності

EI ; **INTE<= 1** (дозволяння переривання)
RET

ЛІТЕРАТУРА

1. Колонтаевський Ю.П. Електроніка і мікросхемотехніка / Ю.П. Колонтаевський, А.Г. Сосков. – К.: Каравела, 2009. – 416 с.

2. Електроніка та мікросхемотехніка: підручник / О.М.Воробйова, І.П. Панфілов, М.П. Савицька, Ю.В. Флейта. – Одеса: ОНАЗ ім. О.С. Попова, 2015. – 298 с.

3. Мартин Р. Чиста архітектура / Р. Мартин. – К.: Фабула, 2019. – 416с.

4. Колонтаевський Ю.П., Сосков А.Г. Електроніка і мікросхемотехніка. – К.: Каравела, 2009. – 416 с.

5. Електроніка та мікросхемотехніка: підручник / О.М. Воробйова, І.П. Панфілов, М.П. Савицька, Ю.В. Флейта. – Одеса: ОНАЗ ім. О.С. Попова, 2015. – 298 с.

6. Матвієнко М., Розен В.П. Комп'ютерна схемотехніка. – К.: Ліра-К, 2014. – 192 с.

7. Таненбаум Эндрю, Остин Тодд. Архитектура компьютера. – СПб, Питер, 2013. – 816 с.

8. Мартин Р. Чиста архітектура. К.: Фабула, 2019. – 416с.

9. Матвієнко М., Розен В.П., Закладний О.М. Архітектура комп'ютера. – К.: Ліра-К, 2013. – 264 с.

10.Паттерсон Д., Хеннесси Дж. Архитектура компьютера. – СПб, Питер, 2012. – 784 с.

11.Дэвид М. Хэррис, Сарра Л. Хэррис Цифровая схемотехника и архитектура компьютера.—СПб.:ДМК, 2018.- 792 с.

Додаток А Система команд мікропроцесора I8080

Система команд однокриштального мікропроцесора I8080 має команди трьох форматів. Перший байт команди містить інформацію про формат команди, код операції, вид адресації та про регістри або регістрові пари, якщо вони приймають участь у виконанні операцій.

У двобайтових командах другий байт (***B2***) містить 8-розрядний операнд або 8-розрядну адресу пристрою вводу чи виводу.

У трибайтових командах другий та третій байти (***B2***, ***B3***) містять 16-розрядні адреси (у командах із прямою адресацією пам'яті) або 16-розрядні операнди (у командах завантаження регістрових пар або покажчика стека). Другий байт трибайтової команди містить молодший байт числа, третій – старший.

У таблиці А.1 наведені мнемонічні позначення та опис команд мікропроцесора i8080. При цьому використовуються наступні умовні позначення :

r - регістр загального призначення;
rp - пара регістрів загального призначення;
PC - програмний лічильник (лічильник команд);
SP - покажчик стека;
M - чарунка пам'яті;
B2,B3 - другий та третій байти команд

Таблиця А.1 - Система команд мікропроцесора I8080

Команда	Довжина команди, байт	Число тактів	Опис команди	Ознаки
1	2	3	4	5
1 Команди пересилання даних				
MOV r1,r2	1	5	Пересилання даних із регістра **r**2 до регістра **r**1	-
MOV M,r (MOV r,M)	1	7	Пересилання даних із регістра **r** до пам'яті за адресою, що зберігається у регістровій парі **H-L** (із пам'яті до регістру **r**)	-
XCHG	1	4	Обмін даними між парами регістрів **H-L** і **D-E**	-

Продовження таблиці А.1

1	2	3	4	5
MVI r,<B2> **(MVI M,<B2>)**	2	7 (10)	Занесення байта даних до регістру **r** (до пам'яті)	-
LXI rp **<2байта>**	3	10	Занесення двох байтів даних у пару регістрів **(B-C, D-E, H-L, SP**). Третій байт команди заноситься в старший регістр, а другий - у молодший	-
LDAX rp	1	7	Завантаження в накопичувач вмісту чарунки, яка опосередковано адресується парою регістрів **rp (B-C, D-E)**	-
LDA <адреса>	3	13	Завантаження накопичувача вмістом чарунки за вказаною адресою. 2-й байт команди - молодший байт адреси, 3-й байт - старший	-
STAX rp	1	7	Занесення вмісту накопичувача до чарунки, яка опосередковано адресується парою **rp**	-
STA, <адреса>	3	13	Занесення вмісту накопичувача до чарунки за вказаною адресою	-
LHLD <адреса>	3	16	Завантаження регістра **L** вмістом чарунки за вказаною адресою, а регістр **H** - чарунки з адресою на одиницю більше	-
SHLD <адреса>	3	16	Занесення вмісту регістрів **H** і **L** до пам'яті (аналогічно команді **LHLD**)	-
2 Арифметичні команди				
ADD r **(ADD M)**	1	4 (7)	Складання змісту регістра **r** (чарунки пам'яті) і накопичувача	Z,S,P,C
ADC r **(ADC M)**	1	4 (7)	Складання змісту регістра **r** (чарунки пам'яті) і накопичувача з бітом перенесення	Z,S,P, C,AC

Продовження таблиці А.1

1	2	3	4	5
SUB r (SUB M)	1	4 (7)	Віднімання змісту регістра **r** (чарунки пам'яті) від змісту накопичувача	Z,S,P, C[1],AC[2]
SBB r (SBB M)	1	4 (7)	Віднімання змісту регістра **r** (чарунки пам'яті) та біта перенесення від змісту накопичувача	Z,S,P, C[1],AC[2]
ADI ,<байт>	2	7	Складання байта зі змістом накопичувача	Z,S,P,C, AC
ACI ,<байт>	2	7	Складання байта зі змістом накопичувача та бітом перенесення	Z,S,P,C, AC
SUI,<байт>	2	7	Віднімання байта із змісту накопичувача	Z,S,P, C[1],AC[2]
SBI,<байт>	2	7	Віднімання байта команди і біта перенесення від змісту накопичувача	Z,S,P, C[1],AC[2]
DAD rp	1	10	Складання змісту пари регістрів **rp** (**B-C,D-E,H-L,SP**) зі змістом пари регістрів **H-L**	C
INR r (INR M)	1	5 (10)	Збільшення змісту регістра **r** (чарунки пам'яті) на одиницю	Z,S,P, AC
DCR r (DCR M)	1	5 (10)	Зменішення змісту регістра **r** (чарунки пам'яті) на одиницю	Z,S,P, AC[2]
INX rp (DCX rp)	1	5	Збільшення (зменьшення) змісту пари регістрів **rp** (**B-C,D-E,H-L,SP**) на оди-ницю	-
DAA	1	4	Перетворення змісту накопичувача в двійково-десятичний код	Z,S,P,C, AC
3 Логічні команди				
ANA r (ANA M)	1	4 (7)	Порозрядне 'І' над змістом регістра **r** (чарунки пам'яті) і накопичувача	Z,S,P, C=0, AC=0
XRA r (XRA M)	1	4 (7)	Порозрядне виключаюче 'АБО' над змістом регістра **r** (чарунки пам'яті) і накопичувача	Z,S,P, C=0, AC=0

Продовження таблиці А.1

1	2	3	4	5
ORA r (ORA M)	1	4 (7)	Порозрядне ‘АБО’ над змістом регістра **r** (чарунки пам’яті) і накопичувача	Z,S,P, C=0, AC=0
CMP r (CMP M)	1	4 (7)	Порівняння змісту регістра **r** (чарунки пам’яті) і накопичувача	(Z,S,P, C,AC)[3]
ANI,<байт>	2	7	Порозрядне ‘І’ над змістом накопичувача і байтом	Z,S,P, C=0, AC=0
XRI,<байт>	2	7	Порозрядне виключаюче ‘АБО’ над змістом накопичувача і байтом	Z,S,P, C=0, AC=0
ORI,<байт>	2	7	Порозрядне ‘АБО’ над змістом накопичувача і байтом	Z,S,P, C=0, AC=0
CPI,<байт>	2	7	Порівняння байта зі змістом накопичувача	(Z,S,P, C,AC)[3]
RLC (RRC)	1	4	Циклічний зсув змісту накопичувача вліво (вправо)	C[4]
RAL (RAR)	1	4	Циклічний зсув змісту накопичувача вліво (вправо) через перенесення	C[4]
CMA	1	4	Порозрядне інвертування накопичувача	-
STC	1	4	Встановлення ознаки перенесення в одиницю	C=1
CMC	1	4	Інвертування ознаки перенесення C	$C=\overline{C}$
4 Команди переходів				
PCHL	1	5	Занесення змісту регістрів **H**, **L** в лічильник команд (зміст **H** – в старший байт, **L** – в молодший)	-
JMP,<адреса>	3	10	Безумовний перехід по вказаній адресі	-
JC/(JNC), <адреса>	3	10	Перехід при наявності (відсутності) перенесення	-
JZ/(JNZ), <адреса>	3	10	Перехід при наявності (відсутності) нуля	-
JP/(JM), <адреса>	3	10	Перехід при плюсі (мінусі)	-

Продовження таблиці А.1

1	2	3	4	5
JPE/(JPO), <адреса>	3	10	Перехід при парності (непарності)	-
CALL,<адреса>	3	17	Виклик підпрограми	-
CC/(CNC), <адреса>	3	11 (17)	Виклик підпрограми при наявності (відсутності) перенесення	-
CZ/(CNZ), <адреса>	3	11 (17)	Виклик підпрограми при наявності (відсутності) нуля	-
CP/(CM), <адреса>	3	11 (17)	Виклик підпрограми при плюсі (мінусі)	-
CPE/(CPO), <адреса>	3	11 (17)	Виклик підпрограми при парності (непарності)	-
RET	1	10	Повернення з підпрограми	-
RC/(RNC)	1	5 (11)	Повернення при наявності (відсутності) перенесення	-
RZ/(RNZ)	1	5 (11)	Повернення при наявності (відсутності) нуля	-
RP/(RM)	1	5 (11)	Повернення при плюсі (мінусі)	-
RPE/(RPO)	1	5 (11)	Повернення при парності (непарності)	-
RST,<номер>	1	11	Повторне запущення з адреси 8 х номер (0,8,…,56)	-
5 Команди вводу-виводу та управління				
IN,<порт>	2	10	Ввод даних із вказаного порту до накопичувача	-
OUT,<порт>	2	10	Вивід даних із накопичувача до вказаного порту	-
PUSH rp	1	11	Занесення змісту пари регістрів **rp** (**B-C,D-E,H-L, PSW**) до стека	-
POP rp	1	10	Видання даних зі стека в пару регістрів **rp** (**B-C,D-E, H-L,PSW**)	(Z,S,P,CAC)[6]
XTHL	1	18	Обмін даними між верхівкою стека та парою регістрів **H-L**	-
SPHL	1	5	Занесення в покажчик стека змісту регістрів **H-L**	-
DI/EI	1	5	Заборонити/дозволити переривання	-
NOP	1	4	Порожня операція	-
HLT	1	7	Зупин	-

Примітки:

1 встановлюється при наявності займу до старшого розряду, у протилежному випадку скидається;

2 встановлюється при наявності займу зі старших чотирьох розрядів в молодші, у протилежному випадку скидається;

3

– **Z** встановлюється, якщо зміст регістра та байта даних дорівнює змісту накопичувача ;

– **S,C,** якщо зміст регістра або байта даних більше змісту накопичувача;

– **AC,** якщо зміст молодших чотирьох розрядів регістра й байта даних більше змісту молодших чотирьох розрядів накопичувача;

– **P,** якщо байт різниці між змістом накопичувала та змістом регістра або байта даних містить парне число одиниць;

4 стан ознаки дорівнює значенню висунутого з накопичувача двійкового розряду;

5 у знаменнику дробу вказано кількість тактів при виконанні умов, в чисельнику – при невиконанні;

6 за командою POP PSW ознаки установлюються відповідно до значення розрядів слова, яке занесено до стека, при інших значеннях rp ознаки не змінюються