Kpi-best

Міністерство освіти та науки України

Національний технічний університет України «Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Розрахунково-графічна робота

з дисципліни

«Архітектура комп’ютера»

Виконав:

Студент Шийка В.В.

Група ІО-01

ФІОТ  
Залікова книжка № 129

Номер технічного завдання 1000001

Прийняв:

Викладач Ткаченко В.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ 2012 г.

Опис альбому

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *N п.п.* | *Формат* | *Позначення* | | | *Найменування* | *К. сторін* | | *№ экз.* | | *Примітки* | | |
|  |  |  | | | Документація загальна |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  | А4 | ІАЛЦ 463617.001 ОА | | | Мікропроцесорна система | 1 | |  | |  | | |
|  |  |  | | | *Опис альбому* |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  | А4 | ІАЛЦ 463617002 ТЗ | | | Мікропроцесорна система | 1 | |  | |  | | |
|  |  |  | | | *Технічне завдання* |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  | А3 | ІАЛЦ 463617.004 Е2 | | | Мікропроцесорна система | 1 | |  | |  | | |
|  |  |  | | | *Схема структурна* |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  | А4 | ІАЛЦ 463617.003ПЗ | | | Пояснювальна записка | 35 | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  |  |  | *ІАЛЦ 463617.001 ОА* | | | | | | | |
|  |  |  |  |  |
| *Изм.* | *Лист* | *№ докум.* | *Підпис* | *Дата* |
| *Виконав* | | *Шийка В.В.* |  |  | Мікропроцесорна система  *Опис альбому* | | *Лит.* | | | | *Лист* | *Листів* |
| *Перевірив* | | *Ткаченко В.В.* |  |  |  | |  |  | *1* | *1* |
|  | |  |  |  | *НТУУ «КПІ» ФІОТ*  *Гр. ІО-01* | | | | | |
| *Н. контр.* | |  |  |  |
| *Зав. каф.* | |  |  |  |

Технічне завдання

Пояснювальна записка

Зміст

[ВСТУП 2](#_Toc280296568)

[1. Структура мікроконтролера 3](#_Toc280296570)

[1.1 Акумулятор 3](#_Toc280296573)

[1.2 Арифметико-Логічний пристрій 3](#_Toc280296574)

[1.3 Лічильник команд 4](#_Toc280296575)

[1.4 Пам’ять програм 4](#_Toc280296576)

[1.4.1 Режим роботи з резидентною пам’яттю програм 5](#_Toc280296581)

[1.4.2 Режим роботи з зовнішньою пам’яттю програм 5](#_Toc280296581)

[1.5 Пам’ять данных 6](#_Toc280296576)

[1.5.1 Режим роботи з зовнішньою пам’яттю даних 7](#_Toc280296581)

[1.6 Приклади програм 9](#_Toc280296576)

[1.7 Система команд 10](#_Toc280296580)

[1.7.1 Формат команд 10](#_Toc280296581)

[1.7.2 Система команд 10](#_Toc280296582)

[1.7.3 Команди пересилки даних 11](#_Toc280296583)

[1.7.4 Команди передачі управління 13](#_Toc280296583)

[1.7.5 Команди управління режимами роботи 14](#_Toc280296583)

[1.7.6 Приклади виконання 14](#_Toc280296583)

[1.8 КПП і КПДП. Реалізація переривань. 22](#_Toc280296580)

[2. Програмна частина 26](#_Toc280296585)

[2.1 Підпрограма множення 26](#_Toc280296586)

[2.1.1 Блок-схема алгоритма 26](#_Toc280296587)

[2.2 Блок-схема алгоритму обчислення основної функції 27](#_Toc280296590)

[2.3 Лістинг програми 28](#_Toc280296591)

[ВИсновок 34](#_Toc280296592)

[ЛіТЕРАТУРА 35](#_Toc280296593)

# ВСТУП

Мікроконтролер - мікросхема, призначена для керування електронними пристроями. Типовий мікроконтролер поєднує в собі функції процесора і периферійних пристроїв, може містити ОЗП і ПЗП. Велика частина процесорів, що випускається в сучасному світі - мікроконтролери.

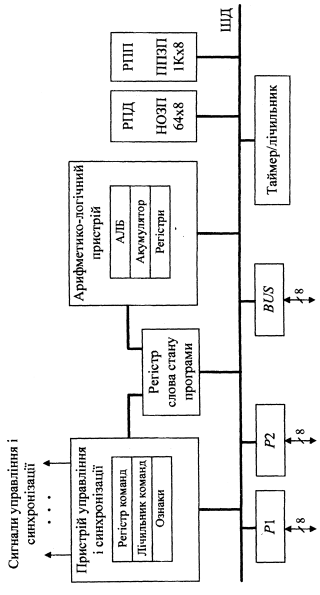
Основною класифікаційною ознакою мікроконтролерів є розрядність даних, які оброблюються арифметико-логічним пристроєм (АЛП). За цією ознакою вони поділяються на 4 -, 8 -, 16 -, 32 - і 64-розрядні.

При проектуванні мікроконтролерів доводиться дотримувати баланс між розмірами і вартістю з одного боку і гнучкістю і продуктивністю з іншого. Для різних додатків оптимальне співвідношення цих та інших параметрів може розрізнятися дуже сильно. Тому існує величезна кількість типів мікроконтролерів, що відрізняються архітектурою процесорного модуля, розміром і типом вбудованої пам'яті, набором периферійних пристроїв, типом корпусу і т. д.

Неповний список периферії, яка може бути присутня в мікроконтролерах, включає в себе:

* універсальні цифрові порти
* різні інтерфейси введення-виведення
* аналого-цифрові і цифро-аналогові перетворювачі
* широтно-імпульсні модулятори
* таймери, вбудований тактовий генератор і таймер
* контролери без колекторних двигунів
* контролери дисплеїв і клавіатур
* радіочастотні приймачі та передавачі
* масиви вбудованої флеш-пам'яті

Програмування мікроконтролерів зазвичай здійснюється на мові асемблера або Сі. Для налагодження програм використовуються програмні симулятори (спеціальні програми для персональних комп'ютерів, що імітують роботу мікроконтролера) внутрішньосхемний емулятор і інтерфейс JTAG.

1. **Структура мікроконтролера 1816ВЕ48**

Структурна схема МК48 наведена на рис.1 Мікроконтролер містить резиденту пам’ять програм (РПП), резидентну пам’ять даних(РПД); пристрій управління і синхронізації, до складу якого входять: лічильник команд, регістр команд і регістри ознак; арифметико-логічний пристрій, до складу якого входить АЛБ, акумулятор і регістри; регістр слова стану програми; таймер/лічильник. Обмін даними здійснюється через порти *Р*1, *Р2* і *BUS.*

**Рис.1 Структурна схема мікроконтролера КР1816ВЕ48**

***1.1 Акумулятор***

Акумулятор *А* являється восьмирозрядним регістром, який крім вказівника адреси, використовується в якості приймача або джерела операнда.

***1.2 Арифметико-логічний пристрій***

До складу АЛП входять наступні блоки: комбінаційна схема обробки байтів, регістри Г, регістр-акумулятор А, схема десяткового коректора і схема формування ознак .

Акумулятор використовується як регістр операнда і регістра ре­зультату. Регістр тимчасового зберігання операнда Т1 програмно недоступний і використовується для тимчасового зберігання другого операнда при виконанні двооперандних команд. Комбінаційна схема АЛП може виконувати наступні операції: складання байтів з перене­сенням або без нього; логічні операції І, АБО та ВИКЛЮЧНЕ АБО; інкремент, декремент, інверсія, циклічний зсув вліво і вправо із встановленням (або без встановлення) ознаки переносу, обмін тетрадами в байті; десяткова корекція вмісту акумулятора.

Під час виконання команди ЛЗЬ (де *Ь* = 0,7 ) в АЛБ формується ознака *ВЬ,* яка відповідає значенню відповідного розряду акумуля­тора. Під час виконання команд JZ, DJNZ формуєть­ся ознака і нульового вмісту акумулятора або зазначеного у команді регістру. За значеннями ознак *ВЬ* і *z* здійснюється розгалуження програм. Тригери для зберігання ознак *ВЬ* і  *z* відсутні тобто після виконання команд ознаки не запам’ятовуються.

При виконанні операцій обробки даних в АЛБ виробляються ознаки, які формуються на комбінаційній схемі і не фіксуються на тригерах, за винятком ознаки переносу С.

Ознака додаткового переносу *АС* встановлюється під час пере­носу з молодшої тетради в старшу. Ознаки переносу і допоміжного переносу фіксуються на тригерах, що входять до складу регістра слова стану програми *PSW.* Окрім перерахованих ознак логіка умовних переходів МК оперує прапорами *F0* і *F1,* функціональне призначення яких визна­чається розробником; прапором переповнювання таймера *TF,* сигна­лами на входах T0 і T1. Програмістом можуть бути також викорис­тані ознаки робочого банку регістрів *RB* і вибраного банку зовніш­ньої пам’яті програм MB. Крім того, логікою переходів після закінчення кожного машинного циклу обпитується ще одна ознака, а са­ме ознака дозволу/заборони переривання.

***1.3 Лічильник команд***

Лічильник команд PC має довжину дванадцять розрядів. Після вибірки чергової команди вміст *PC* збільшується на одиницю. Пере­нос при цьому розповсюджується тільки від нульового до десятого розряду.

Старший розряд РС[ 11] виконує спеціальну функцію і визнача­ється ознакою банку пам’яті програм MB, яка встановлюється про­грамно, за допомогою команд SEL МВ0 і SEL МВ1. Програмні за­соби для перевірки ознаки банка пам’яті MB відсутні.

***1.4 Пам’ять програм***

Пам’ять пробам і пам’ять даних в МК48 розділені.

Пам’ять програм розділяється на резиденту, розташовану всередині ІС, та зовнішню, для реалізації якої необхідні додаткові ІС пам’яті. Максимальний адресний простір пам’яті програм складає -1 Кб (рис. 2). Резидента пам’ять програм (РПП) представляє со­бою перепрограмоване ПЗУ(ППЗУ) об’ємом 1 Кб (адреса від 0 до 1023).

Адреси 0, 0003 та 0007 мають спеціальне призначення. З адреси 0 починається виконання програми за системним скиданням. Комір­ка 0003 призначена для зберігання початкової адреси підпрограми обслуговування зовнішнього переривання, а комірка 0007 - збері­гання початкової адреси підпрограми обробки переривання від тай­мера/лічильника.

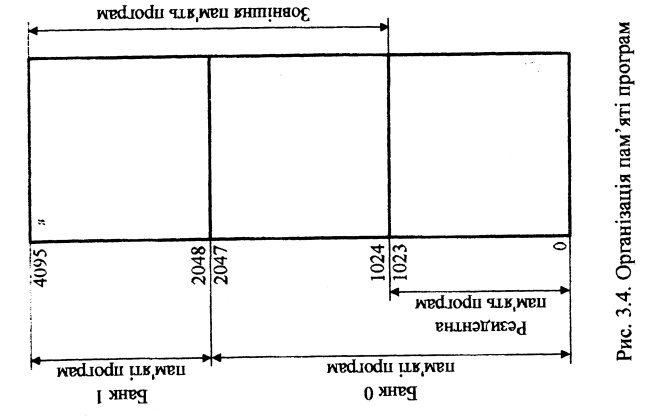
Пам’ять програм розглядається як два банки - нульовий банк програм і перший банк програм. Якщо встановлюється розряд лічи­льника команд РС[11] = 0, то вибір слів здійснюється із нульового банку пам’яті (адреси від 0 до 2047), і, якщоРС[11] = 1 із першого банку пам’яті (адреси від 2047 до 4095). Вибір банку пам’яті здійс­нюється командами SEL MB0 та SEL МВ1, які встановлюють озна­ку MB вибору банку пам’яті.

Окрім розділення на банки програм, пам’ять програм поділя­ється на сторінки об’ємом 256 байт. Це пов’язано з тим, що команди умовних переходів модифікують тільки вісім молодших розрядів адреси, тобто забезпечують перехід всередині сторінки. При перехо­ді до підпрограм обслуговування переривань автоматично встанов­люється в нуль розрядРС[11] лічильника команд. У зв’язку з цим підпрограми обслуговування переривань повинні розміщуватися в нульовому банку пам’яті.

Способи адресації операндів в пам’яті програм:

* безпосередня;
* непряма з використанням акумулятора.

В першому випадку операнд розміщується в байті, наступному за кодом команди. За непрямої адресації, в якості покажчика адреси операнда застосовується акумулятор *А.* Акумулятор містить адресу чергової сторінки пам’яті (MOVP А, @А), або третьої сторінці (MOV3P А, @А).



**Рис.2 Організація пам'яті програм**

***1.4.1 Режим роботи з резидентною пам’яттю програм***

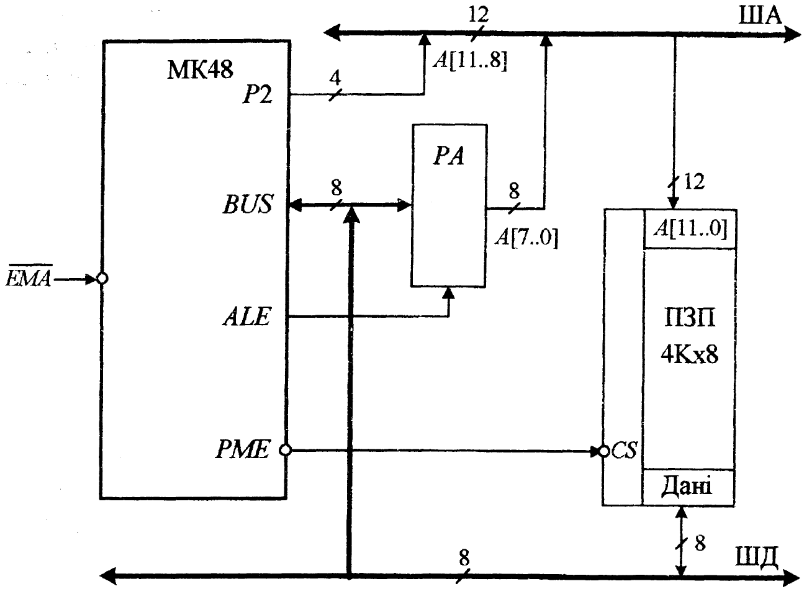
У режимі роботи МК48 з резидентною пам’яттю програм (для ВІС ВЕ48 і ВІС ВЕ49 за *ЕМА* = 0) для зв’язку з об’єктом управління використовуються три порти (*Р1*, *Р2, ВUS).* При цьому кожний розряд будь-якого порту може бути запрограмований на ввід чи вивід інформації. За необхідності між виходами МК48 і входами/виходами об’єкту управління включаються відповідні елементи, що забезпечують формування сигналів за різними параметрами. Під час звернення резидентної пам’яті програм зовнішні управляючи сигнали, окрім сигналу ALE, не формуються.

СигналALE за необхідності можна використовувати для синхронізації роботи внутрішніх пристроїв. Якщо адреса перевищує допустиме для резидентної пам’яті програм значення – 1023, то МК48 автоматично переходить в режим роботи з зовнішньою пам’яттю програм.

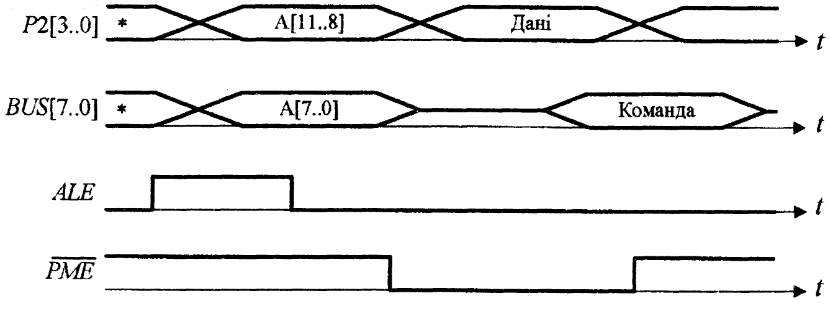
***1.4.2 Режим роботи з зовнішньою пам’яттю програм***

Режим роботи МК48 з зовнішньою пам’яттю програм можливий за застосування додаткових мікросхем ПЗП. Якщо не використовувати сторінкову адресацію, об’єм пам’яті програм можна розширити до 4К байт. За встановлення сигналу *ЕМА* = 1 доступні всі 4К байта зовнішньої пам’яті. Якщо сигнал *ЕМА* = 0, то адресація комірок зовнішньої пам’яті розпочинаються з адреси 1024. При цьому область пам’яті з адресами від 0 до 1023 належать резидентній пам’яті програм. Схема підключення зовнішньої пам’яті програм об’ємом 4К наведена на рис. 3, а часова діаграма читання байта (команди або даних) - на рис. 4.

Для підключення зовнішньої пам’яті програм використовуються виходи портів BUS[7..0] та P2[3..0]. Для зберігання адреси звернення до пам’яті використовується зовнішній регістр адреси РА.



**Рис.3 Схема підключення зовнішньої пам’яті програм**

****

**Рис.4 Часова діаграма вибору інформації з зовнішньої пам’яті програм**

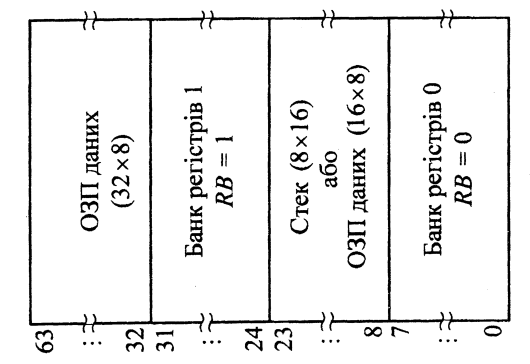
В процесі звернення до зовнішньої пам'яті програм МК48 формує дванадцятирозрядну адресу, де старші розряди адреси *А[* 11..8] формуються на виходах Р2[3..0] мікросхеми, а молодші розряди *А[7..0]* - на виходах *BUS[7..0].* Запис адреси в регістр адреси РА стробується сигналом *ALE.* Після цього виходи порту *BUS* перемикаються на ввід інформації та видається сигнал дозволу читання пам'яті РМЕ. За цим сигналом на виходах даних мікросхеми пам'яті формується байт інформації (команди або даних), який за шиною *BUS* приймається в МК48.

За випадку, коли резидентна пам'ять програм відсутня (ЕМА = 0), необхідно передбачити логіку відключення резидентної пам'яті під час звернення до адрес 0 - 1023. Наприклад, якщо вибір зовнішньої пам'яті здійснюється низьким рівнем сигналу на вході CS, то на цей вхід можна подати значення логічної функції *NOT(P2[3]OR P2[2]).*

***1.5 Пам’ять даних***

Пам’ять даних розділяється на внутрішню і зовнішню. Внутрі­шня пам’ять даних представляє собою ОЗП ємністю 64 байти (рис. 5). Пам’ять даних містить два банки регістрів загального при­значення. Банк регістрів 0 включає регістри *R0-R7* з адресами 0 - 0007, а банк регістрів 1 - регістри *R0 - R1*, які мають адреси 0024 - 0031. Вибір банку регістрів здійснюється командами SEL RB0 та SEL RB1, які встановлюють ознаку *RB,* що знаходиться в четвертому розряді *PSW.*

Спеціальна команда для перевірки *RB* відсутня, але ознаку мо­жна проаналізувати, перенісши вміст *PSW* в *А* і виконати перехід за ознакою В4, яка перевіряється командою JB4. Комірки з адресами 0008 - 0023 можуть використовуватися як восьмирівневий стек шістнадцятирозряднйх слів або як комірки ОЗП даних задовільним доступом.



**Рис.5 Карта розподілу адрес внутрішньої пам'яті даних**

*Покажчик стеку SP* входить до складу регістра *PSW* (розряди S2 - S0). При переході до підпрограми в стек записується вміст *PC* і чотирьох старших розрядів *PSW*. Частина внутрішньої пам’яті з адресами 0032 - 0063 використовується тільки в якості ОЗП даних.

*Способи адресації* для доступу до внутрішньої пам’яті даних:

* пряма регістрова;
* непряма регістрова.

Пряма регістрова адресація використовується для звернення до регістрів загального призначення *R0-R7* банку регістрів 0, якщо *RB* = 0, та *R0 - R1* банку регістрів 1, якщо *RB* = 1. В цьому випадку трирозрядна адреса регістра присутня в коді команди.

За допомогою непрямої регістрової адресації можна звернутися до будь-якого байту внутрішньої пам’яті даних ( в тому числі до регістрових банків і стеку). В якості покажчика адресу операції в та­кому випадку використовуються регістри R0, R1*,* відповідно до обраного банку регістрів (нульового або першого). Для обміну даними застосовуються наступні команди MOV A, @Rr, MOV @Rr, А, (де r = 1,0).

***1.5.1 Режим роботи з зовнішньою пам’яттю даних***

В режимі роботи МК48 з зовнішньою пам'яттю даних використовуються додаткові мікросхеми ОЗП об'ємом 256 байт.

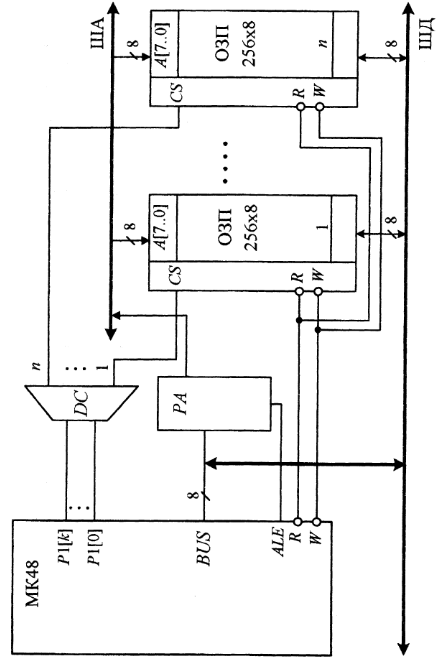
Якщо адресний простір має об'єм більш ніж 256 байт, то необхідна сторінкова організація зовнішньої пам'яті даних. При роботі з адресами в межах однієї сторінки застосовуються команди **MOVX A, @Rr, MOVX @Rr, А** (де r =1,0). Обмін інформацією здійснюється між акумулятором А і коміркою ОЗП, яка непрямо адресується через регістрR0 або R1. Переключення між сторінками потребує використання додаткових команд вибору сторінки пам'яті даних.

На рис.6 показаний спосіб підключення до МК48 n сторінок зовнішньої пам'яті даних з використанням *к* виходів портуР1 (де *к* =]log2n[) і дешифратора *DC.*

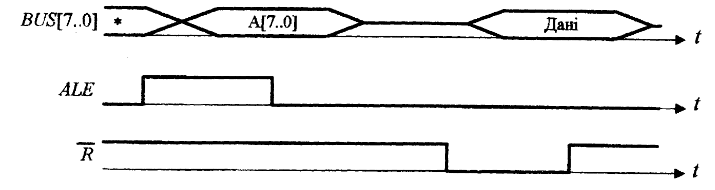
За наявності одночасно зовнішньої пам'яті програм і даних використовується один і той самий зовнішній регістр адреси.

Часові діаграми циклів звернення до зовнішньої пам'яті даних наведені на рис.7.рис.8

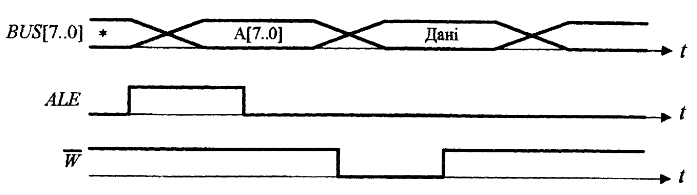
Команди виконуються за два цикли. В процесі читання пам'яті спочатку на шину *BUS* встановлюється адреса, яка фіксується у зовнішньому регістрі РА за встановлення сигналу *ALE.* Після чого виходи порту *BUS* перемикаються для вводу інформації і формується сигнал читання даних *R.* За цим сигналом ЗПД виставляє дані на ШД, які через порт *BUS* приймаються в МК48. На протязі циклу запису інформація, що передається, встановлюється на шині *BUS.* Запис в пам'ять стробується сигналом W.

****

**Рис.6 Схема підключення зовнішньої пам’яті даних**

****

**Рис.7 Часова діаграма обміну даними з зовнішньою пам’яттю даних, читання даних**

****

**Рис.8 Часова діаграма обміну даними з зовнішньою пам’яттю даних, запис даних**

* 1. **Приклад передачі масиву даних із зовнішньої пам'яті програм в банк регістрів**

;Пересилання масиву із 7 байт з РВ в RB0

MOV R7, #7 ; запис константи 7 в регістр R7, для пересилання масиву

CALL TOOL1 ; виклик підпрограми TOOL1, першого інструменту для пересилання

SEL RB0 ; підключення нульового банку регістрів

MOV R0, #C1h ; підготовка адреси порту РВ

MOV R1, #00h ; підготовка адреси РПД

LOOP1: ; мітка LOOP1

MOVX A, @R0 ; читання байту з РВ

MOV @R1,A ; пересилання з РВ в RB0

INC R1 ; збільшення покажчика на 1

DJNZ R7, LOOP1 ; декремент R7 та повернення на початок циклу LOOP1 поки R7!=0

;Пересилання масиву із 9 байт з PA в RB1

MOV R7, #9 ; запис константи 9 в регістр R7, для пересилання масиву

CALL TOOL2 ; виклик підпрограми TOOL2, другого інструменту для пересилання

SEL RB1 ; підключення першого банку регістрів

MOV R0, #C0h ; підготовка адреси порту РА

MOV R1, #18h ; підготовка адреси РПД

LOOP2: ; мітка LOOP2

MOVX A, @R0 ; читання байту з РА

MOV @R1,A ; пересилання з РА в RB1

INC R1 ; збільшення покажчика на 1

DJNZ R7, LOOP2 ; декремент R7 та повернення на початок циклу LOOP2 поки R7!=0

JMP ENDED ; стрибок до мітки ENDED

TOOL1: ; мітка для входження в підпрограмму TOOL1

MOV R0, #C3h ; адреса РУС

MOV A, #81h ; ініціалізація ППА(вивід із РВ)

MOVX @R0, A ; завантаження управляючого слова в РУС

RET ; повернення з підпрограми TOOL1

TOOL2: ; мітка для входження в підпрограмму TOOL2

MOV R0, #C3h ; адреса РУС

MOV A, #80h ; ініціалізація ППА(вивід із РА)

MOVX @R0, A ; завантаження управляючого слова в РУС

RET ; повернення з підпрограми TOOL2

ENDED: ; мітка ENDED

END. ; кінець програми

; Переслати вміст регістрів R2, R3, R4 нульового банку регістрів в ЗПД, починаючи з комірки А0h другої сторінки пам'яті

SEL RB0 ; вибір нульового банку регістрів

MOV R2, #32h ; запис в регістр R2 константи

MOV R3, #13h ; запис в регістр R3 константи

MOV R4, #8h ; запис в регістр R4 константи

ANL P1, #0 ; встановлення порту Р1 в нуль

ORL P1, #4h ; вибір другої сторінки ЗПД

MOV A, R2 ; копіювання вмісту R2 в акумулятор, A=R2

MOV R0, #A0h ; запис в регістр R0 константи A0h, для використання в якості адреси

MOVX @R0, A ; пересилка вмісту акумулятора(вмісту R2) в комірку з адресою A0h

INC R0 ; збільшення вказівника адреси на 1

MOV A, R3 ; копіювання вмісту R3 в акумулятор, A=R3

MOVX @R0, A ; пересилка вмісту акумулятора(вмісту R3) в комірку з адресою A1h

INC R0 ; збільшення вказівника адреси на 1

MOV A, R4 ; копіювання вмісту R4 в акумулятор, A=R4

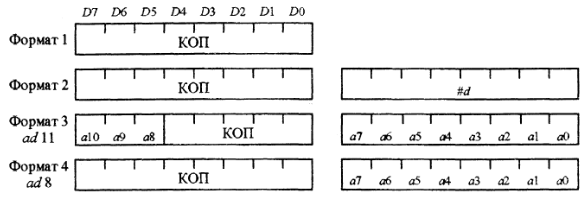
MOVX @R0, A ; пересилка вмісту акумулятора(вмісту R4) в комірку з адресою A2h

END. ; кінець програми

**1.7. Система команд ( приклади виконання деяких програм, а саме, привести три лабораторні роботи по варіантам).**

**1.7.1 Формат команд**

Всі команди МК48 мають формат 1 чи 2 байта і виконуються за один чи два машинні цикли. Кожний цикл виконується за 5 тактів. Частота синхронізації тактів складає F/3 а циклів – F/15. За два машинні цикли виконуються всі команди з безпосереднім операндом, команди введення-виведення, команди передачі управління і роботи з підпрограмами, а також команди пе­ресилок МОVХ, МОVР, МОVР3. Решта команд виконуються за один машинний цикл. В МК48 передбачена можливість суміщення вико­нання однієї команди і вибірки наступної, що може зменшити час виконання команди. Мікроконтролер оперує з командами чотирьох типів (рис. 9).

****

**Рис.9 Типи команд МК48**

**1.7.2 Система команд**

У МК48 використовуються чотири способи адресації:

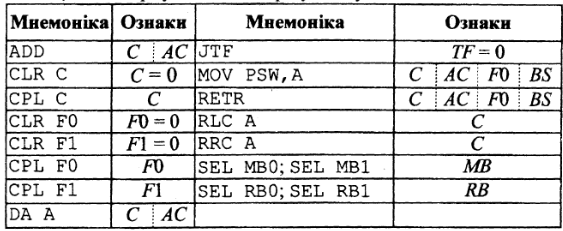
* пряма,
* безпосередня,
* непряма,
* неявна.

До переваг системи команд МК48 можна віднести: ефективний ввід/вивід, включаючи маскування і можливість управління окремими бітами портів; реалізацію розгалуження за значенням окремих бітів; обробку як двійкових, так і десяткових двійково-кодованих чисел.

Під час виконання команд використовуються ознаки, що формуються в регістрі слова стану програми RSW, і ознаки користувача.

Команди, в результаті виконання яких модифікуються ознаки, наведені в табл. 1.

**Таблиця 1. Формування ознак результату**

****

Для опису команд використовуються мнемокоди мови асемблера МК48. Під час запису символічного коду команд застосовуються наступні позначення:

А - акумулятор;

Т - таймер;

r - номер регістру;

Rr - регістр з номером r;

b - номер біту;

р - номер порту вводу/виводу;

Рр - порт з номером р;

а - адреса;

d - безпосередньо операнд;

#d - безпосередньо операнд (восьмирозрядне двійкове число);

@Rr- операнд, що адресується непрямо через Rr;

@А - операнд, що адресується непрямо через А

Rr - робочий регістр.

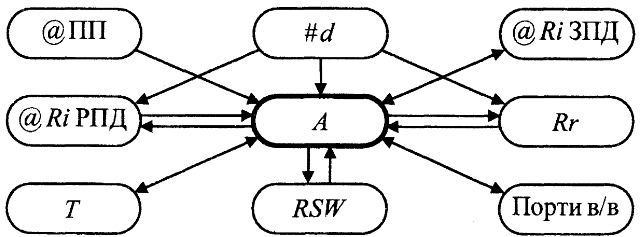
Для запису операцій використовуються мова мікрооперацій із застосуванням, символічних імен і скорочень. Номери розрядів регістрів подаються у квадратних дужках [ ]. Операнд за непрямої адресації подається у дужках ( ). Наприклад, запис *А* := *(Rr)* означає, що в акумуляторі *А* фіксується число, що зчитане з внутрішньої пам'яті даних за адресою, записаною в регістрі Rr. Для запису операндів часто використовуються складені слова, що записані у вигляді двох слів, розділених крапкою. Наприклад, запис *РС[11..8].А* подає дванадцятирозрядне двійкове слово, вісім молодших розрядів якого є вмістом акумулятора А, а чотири старші розряди - вмістом бітів (11..8) лічильника команд *РС.*

Всю множину команд асемблеру МК48 можна розбити на чоти­ри основні групи:

* команди пересилки даних,
* команди основної групи:
* виконання арифметичних операцій;
* виконання логічних операцій;
* команди передачі управління;
* команди управління режимами роботи.

***1.7.3 Команди пересилки даних***

На рис. 10 представлений граф, що ілюструє можливі операції пересилки даних в МК48. Операнди розрізняються за місцем розташування і за способом адресації. В операціях пересилки приймають участь такі операнди: акумулятор, РЗП, PSW, таймер, порти вводу/виводу, безпосередньо операнд, ЗПД, РПД, ПП.



**Рис.10 Граф обміну даними у МК48**

Всі операції пересилки даних в МК48 виконуються з застосуванням акумулятора А.

Під час пересилки даних між *А* та регістрами загального призначення РЗП банків регістрів, використовується пряма адресація, коли адреса операнда вміщується в тілі команди. Номер регістра, вміст якого пересилається в акумулятор вказується в трьох молодших бітах коду операції (рис. 9).

Обмін даними між *А* та комірками РПД або ЗПД здійснюється з використанням непрямої адресації. При цьому, покажчики адреси розміщуються в регістрах *R0* або *R1* вибраного банку регістрів.

За неявної адресації у коді команди неявно указується один з операндів. Найчастіше таким операндом є акумулятор. За безпосередньої адресації у тілі команди в якості другого байту вказується безпосередній операнд (константа), який пересилається за місцем призначення, визначеним першим операндом.

До пам'яті програм здійснюється доступ лише у напряму читання даних.

Всі команди, окрім MOV PSW, А, не впливають на встановлення ознак.

Більшість команд виконує пересилку восьмибітних даних. Декілька команд оперують з чотирибітними операндами (теградами) і застосовуються для звернення до чотирибітних портів вводу/виводу Р4, *Р5, Р6, P7.*

В МК48 передача даних відбувається в двох режимах: пересилки (завантаження) і обміну. Під час пересилки дані передаються від джерела до приймача, при цьому джерело даних не змінює свого вмісту. Обмін припускає одночасну передачу даних в обох напрямах при цьому змінюються значення обох операндів, що приймають участь в обміні.

Команди пересилки даних всередині МК48 виконуються за один машинний цикл, а обмін даними з ЗПД потребує двох машинних циклів.

*Приклади команд:*

|  |  |  |
| --- | --- | --- |
| MOV | A,Rr | *(А) :=Rr*, *г* = (7 - 0); пряма адресація. |
| MOV | A, #d | *А :=d;* безпосередня адресація. |
| MOV | А, #05 | *А :=5* ; безпосередня адресація. |
| MOV | Rr, #d | *(Rr):=d, r* = (7 - 0). |
| MOV | А, PSW | *(А) :=PSW.* |
| MOV | А, Т | *(А) := Т.* |
| MOV | A, @R0 | *(А) := ((Rr))*, *r* = 0,1; непряма адресація. |

До команд основної групи належать команди виконання арифметичних та логічних операцій. В МК48 виконуються наступні операції над восьмибітними цілими двійковими числами без знаку: двійкове додавання, двійкове додавання з урахуванням переносу, операції десяткової корекції, інкременту, декременту, зсуву, кон'юнкції, диз'юнкції тощо.

Дві логічні команди скидання CLR та інверсія CPL дозволяють виконувати операції з бітами.

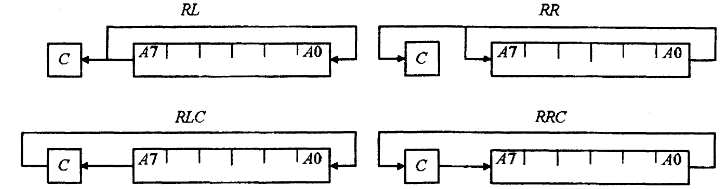
Під час додавання застосовується неявна адресація джерела першого операнду і місця розташування результату, в якості яких використовується акумулятор. До вмісту акумулятора можна додавати вміст регістру РЗП, константу, вміст комірки РПД. Під час підсумовування формується ознака переносу С, що фіксується у відповідному розряді регістру RSW. Команда підсумовування АDDС з урахуванням переносу дозволяє виконувати підсумовування багатобайтних чисел.

В МК48 відсутня безпосередньо операція віднімання, при цьому віднімання реалізується за наступної послідовності дій: отримання додаткового коду другого операнду, додавання його до вмісту А, де зберігається перший операнд, та подання результату в доповнювальному коді.

Складні арифметичні операції ділення, множення, піднесення до ступеня, тощо виконуються за підпрограмами.

Команди виконання арифметичних операцій змінюють відповідні ознаки в регістрі слова стану програми (табл. 1).

Принцип виконання зсувів командами RL, RLC, RR, RRC ілюструє схема, що показана на рис. 11.



**Рис.11. Схема виконання операцій зсуву**

***1.7.4 Команди передачі управління***

В групу команд передачі управління входять дві команди безумовного переходу, команди умовних переходів, команда виклику підпрограми і дві команди повернення з підпрограм.

Виконання команд передачі управління і роботи з підпрограмами має свої особливості і залежить від розподілу програм і даних за банками і сторінками пам'яті.

Більшість команд передачі управління застосовують *пряму адресацію*, при цьому в тілі команди вказується адреса переходу.

Команди *безумовного переходу* JMP і CALL вміщують одинадцятибітну адресу переходу і дозволяють здійснити перехід до будь-якої комірки ЗПП тільки в межах одного банку пам'яті програм (2048 адресів). Номер банку пам'яті програм визначається ознакою MB, значення якого копіюється у старший біт лічильника команд *(PC*11) під час виконання команд JMP або CALL.

Перехід до іншого банку пам'яті програм можливий тільки за використання команд із групи команд управління режимами роботи SEL МВО та SEL МВ1. До тих пір поки не буде обрано інший блок пам'яті, всі переходи здійснюються в межах поточного банку пам'яті.

Інші команди є *командами умовних переходів* (окрім команд повернення), вміщують восьмибітну адресу переходу і дозволяють здійснювати передачу управління лише в межах поточної сторінки (256 байт). Така адресація називається *короткою прямою адресацією*, вона обумовлює певні обмеження на розміщення програм у пам'яті.

За застосування команд умовних переходів ознаки, що аналізуються, за виключенням С і F0 не фіксуються в спеціальних регістрах ознак, а подаються короткотривалими значеннями сигналів на виходах АЛЛ або відповідних входах МК48.

За застосування команд умовних переходів можна перевіряти не тільки внутрішні ознаки, але і деякі сигнали на зовнішніх входах МК48. Це дозволяє виконувати ефективне розгалуження в програ­мах без попереднього вводу і аналізу.

*Непряма адресація* реалізована командою **JMPP,** яка здійснює перехід за адресою, що зберігається у вказаній комірці ПП. Покажчиком адреси є акумулятор. Перехід також здійснюється у межах однієї сторінки ПП.

Для організації програмних циклів використовується команда **DINZ** (декремент регістру і перехід, якщо не нуль), яка дозволяє використовувати будь-який із восьми робочих регістрів в якості лічильника і здійснює перехід в межах поточної сторінки пам'яті програм.

Для роботи з підпрограмами використовуються команди **CALL** та **RET.** Команда **CALL,** дозволяє звернутись до любої комірки поточного банку ПП. Під час виклику підпрограми в стеку за­пам'ятовується адреса повернення і частина регістру *RSW.* Кількість вкладань обмежується глибиною стеку (16 байтів) и не повинна перевищувати восьми. Під час повернення з підпрограми в лічильнику команд поновлюється адреса повернення. Команда **RETR** використовується для повернення з підпрограми обробки переривань, окрім адреси повернення поновлює вміст регістру *RSW* і встановлює дозвіл переривання від даного джерела.

***1.7.5 Команди управління режимами роботи***

В групу команд управління режимами роботи входять команди управління таймером/лічильником, перериваннями и ознаками переключення банків регістрів і банків пам'яті програм.

Вище були розглянуті команди обміну інформацією між таймером і акумулятором (MOV А, Т і MOV Т, А), за виконання яких може бути прочитано вміст таймера після зупинки підрахунку або безпосередньо під час підрахунку, а також за необхідністю перезавантажено вміст таймера. Окрім цього в МК48 виконуються спеціальні команди управління режимом роботи таймера.

Таймер, залежно від застосованої команди, може бути використаний як лічильник тактів від внутрішнього джерела сигналів або як лічильник подій від зовнішнього джерела сигналів.

Система команд МК48 має в своєму розпорядженні засоби дозволу або заборони переривання від таймера. Спеціальною командою ENT0 на вивід Т0 дозволяється передача імпульсів з частотою тактового синхросигналу, діленою на три. Видача цього сигналу може бути відключена тільки сигналом загального скидання. Синхросигнал на виході Т0 використовується для загальної синхронізації зовнішніх пристроїв, узгоджених з МК48 за частотою роботи.

*Приклади команд:*

|  |  |  |
| --- | --- | --- |
| МОV | Т, А | ; завантаження таймеру. |
| STRT | Т | ; запуск таймеру. |
| STRT | CNT | ; запуск лічильника. |
| EN | TCNTI | ; дозвіл переривання від таймеру. |
| DIS | TCNTI | ; заборона переривань від таймеру. |
|  |  |  |

**1.7.6 Приклади виконання деяких програм, а саме, привести три лабораторні роботи по варіантам**

**Лабораторна робота №1**

;Lab1

;ORG 900000h

SEL RB0

;;;;;;;;;;;;;;;;;

; INPUT X1

IN A, P1

MOV R1, A

; INPUT X2

IN A, P1

MOV R2, A

; (R3:R4) = X1 + X2

MOV A, R1

ADD A, R2

; low bits

MOV R4, A

; high bits

;CLR A

MOV A, #0

ADDC A, #0

MOV R3, A

;;;;;;;;;;;;;;;;;

;INPUT X3

IN A, P1

MOV R1, A

;INPUT X4

IN A, P1

MOV R2, A

; (R5:R6) = X3 - X4 - 1

MOV A, R2

CPL A

ADD A, R1

; low bits

MOV R6, A

; high bits

MOV A, #FFh

ADDC A, #0

MOV R5, A

; (R3:R4) = (X1 + X2) & (X3 - X4 - 1)

; low bits

MOV A, R4

ANL A, R6

MOV R4, A

; high bits

MOV A, R3

ANL A, R5

MOV R3, A

; (R3:R4) = 2\*((X1 + X2) & (X3 - X4 - 1))

; shift

MOV A, R4

RLC A

MOV R4, A

MOV A, R3

RLC A

MOV R3, A

;;;;;;;;;;;;;;;;;

;INPUT X5

IN A, P1

MOV R1, A

;INPUT X6

IN A, P1

MOV R2, A

MOV A, R1

ORL A, R2

MOV R6, A

CLR C

MOV R5, #0

; shift

MOV A, R6

RLC A

MOV R6, A

MOV A, R5

RLC A

MOV R5, A

CLR C

; shift

MOV A, R6

RLC A

MOV R6, A

MOV A, R5

RLC A

MOV R5, A

CLR C

; shift

MOV A, R6

RLC A

MOV R6, A

MOV A, R5

RLC A

MOV R5, A

;;;;;;;;;;;;;;;;;;;;;;;;;

; (R3:R4) = (R3:R4) + not(R5:R6)

; low bits

MOV A, R6

CPL A

ADD A, R4

MOV R4, A

; high bits

MOV A, R5

CPL A

ADDC A, R3

MOV R3, A

; (R3:R4) = (R3:R4) + 1

; low bits

MOV A, R4

ADD A, #1

MOV R4, A

; high bits

MOV A, R3

ADDC A, #0

MOV R3, A

END

**Лабораторна робота №2**

12910=10000012

SEL RB0

MOV R7, 4

; INPUT X0

IN A, P1

MOV R0, A

; INPUT X1

IN A, P1

MOV R1, A

; INPUT X2

IN A, P1

MOV R2, A

; INPUT X3

IN A, P1

MOV R3, A

; INPUT X4

IN A, P1

MOV R4, A

; INPUT X5

IN A, P1

MOV R5, A

; INPUT X6

IN A, P1

MOV R6, A

; R2 = (R3 - 1) >> 1

MOV A, R3

DEC A

CLR C

RRC A

MOV R2, A

; R4 = R2 & R3

ANL A, R3

MOV R4, A

ITERA:

; R1 = (R4 + 1) << 1

;; R1l = R4 + 1

MOV A, R4

INC A

MOV R1, A

SEL RB1

;; R1b = C

MOV A, 0h

ADDC A, #0

MOV R1, A

SEL RB0

;; R1l = R1l << 1

MOV A, R1

CLR C

RLC A

MOV R1, A

SEL RB1

;; R1b = C

MOV A, 0h

ADDC A, #0

MOV R1, A

;;;

JC NEXT

CALL MPP

NEXT:

SEL RB0

DJNZ R7, ITERA

SEL RB0

; R7 = (R6 - R5 - 1) >> 2

;; R7l = R6l + not R5l

MOV A, R5

CPL A

ADD A, R6

MOV R7, A

SEL RB1

;; R7b = not R5b + R6b + C

MOV A, R5

CPL A

ADDC A, R6

MOV R7, A

SEL RB1

;; R7b = R7b >> 1

MOV A, R7

CLR C

RRC A

MOV R7, A

SEL RB0

;; R7l = C

MOV A, R7

RRC A

MOV R7, A

;;;

JMP TEHEND

MPP:

SEL RB0

; R6 = R5 + 77h

;; R6l = R5l + 77h

MOV A, R5

ADD A, 77h

MOV R6, A

SEL RB1

;; R6b = R5b + C

MOV A, 0h

ADDC A, R5

MOV R6, A

;;;

JF0 NICE

JMP LOL

NICE:

SEL RB0

; R0 = (R6 - 1) << 2

;; R0l = R6l + FFh

MOV A, R6

ADD A, #FFh

MOV R0, A

SEL RB1

;; R0b = R6b + FFh + C

MOV A, R6

ADDC A, #FFh

MOV R0, A

SEL RB0

;; R0l = R0l << 1

MOV A, R0

CLR C

RLC A

MOV R0, A

SEL RB1

;; R0b = R0b << 1

MOV A, R0

RLC A

MOV R0, A

SEL RB0

;; R0l = R0l << 1

MOV A, R0

CLR C

RLC A

MOV R0, A

SEL RB1

;; R0b = R0b << 1

MOV A, R0

RLC A

MOV R0, A

;;;

LOL:

SEL RB0

; R5 = (R4 - 1) << 1

;; R5l = R4 + FFh

MOV A, R4

ADD A, #FFh

MOV R5, A

SEL RB1

;; R5b = FFh + C

MOV A, 0h

ADDC A, #FFh

MOV R5, A

SEL RB0

;; R5l = R5l << 1

MOV A, R5

CLR C

RLC A

MOV R5, A

SEL RB1

;; R5b = R5b << 1

MOV A, R5

RLC A

MOV R5, A

;;;

RET

TEHEND:

END

**Лабораторна робота №3**

; [X2, X1, Y5, Y4, Y3, Y2, Y1, NULL]

CLR F0

ANL P1, #C0h

IN A, P1

JB6 X1

; BLOCK 1 (y1, y2)

ORL P1, #6 ; (activate)

NOP

NOP

NOP

NOP

NOP

ANL P1, #C2h ; +2.5 (deactivate y2)

MOV A, #FEh ; +2.5

CALL PAUSEB ; +2\*80 +10

NOP

NOP

NOP

NOP

NOP

NOP

ANL P1, #C0h (deactivate y1)

;;;;;;;;;

BLOCK2:

; BLOCK 2 (y1, y3)

ORL P1, #10 ; (activate)

MOV A, #FEh ; +2.5

CALL PAUSEB ; +2\*80 +10

MOV R0, #3h

CALL PAUSEA

NOP

ANL P1, #C8h ; +2.5 (deactivate y1)

MOV A, #FBh ; +2.5

CALL PAUSEB ; +6\*80 +10

MOV R0, #Ah

CALL PAUSEA

ANL P1, #C0h (deactivate y3)

JF0 IFX2

BLOCK3:

; BLOCK 3 (y4, y5)

ORL P1, #48 ; (activate)

MOV A, #FBh ; +2.5

CALL PAUSEB ; +5\*80 +10

MOV R0, #11

CALL PAUSEA

ANL P1, #E0h ; +2.5 (deactivate y4)

MOV R0, #8 ; +2.5

CALL PAUSEA ; +11\*2.5 +10

ANL P1, #C0h (deactivate y5)

;;;;;;;;;

JMP TEHEND

BLOCK4:

; BLOCK 4 (y3)

ORL P1, #8 ; (activate)

MOV A, #F8h ; +2.5

CALL PAUSEB ; +9\*80 +10

MOV R0, #7 ; +2.5

CALL PAUSEA ; +11\*2.5 +10

ANL P1, #C0h (deactivate y3)

;;;;;;;;;

JMP TEHEND

X1:

CLR F0

CPL F0

JMP BLOCK2

IFX2:

ANL P1, #C0h

IN A, P1

JB7 BLOCK4

JMP BLOCK3

PAUSEA:

LL1: DJNZ R0, LL1 ; +R0\*2.5

RET

PAUSEB:

MOV T, A ; +2.5

STRT T ; +2.5

ISJTF: JTF FINE ; +A\*80

JMP ISJTF ; +2.5

FINE:

RET

TEHEND:

END

**1.8 КПП і КПДП. Реалізація переривань. . Реалізація обміну інформацією із зовнішніми пристроями в режимі прямого доступу до пам'яті. Привести рисунки, які ілюструють роботу вибраного блоку по варіанту (централізований або децентралізований).**

Під перериваннями розуміють тимчасове припинення виконан­ня програми і перехід на підпрограму з можливістю повернення на перервану програму.

Розрізняють внутрішні і зовнішні переривання. Внутрішні пе­реривання у свою чергу поділяються на програмні і апаратні. Про­грамні переривання є ефективним засобам для виклику стандартних підпрограм базової системи вводу-виводу, дозволяють спростити процес налагодження програм взаємодії процесора із зовнішніми пристроями, які, можливо, розробляються паралельно з програмним забезпеченням і таке інше. Внутрішні апаратні переривання дозволяють процесору відреагувати на непередбачені ситуації (наприклад, переповнювання розрядної сітки, спрацьовування внутрішнього таймера, збій при зверненні до магістралі) і скоректувати обчислю­вальний процес, змінити режим роботи або, принаймні, провести безпечне зупинення системи.

Існують векторні і безвекторні зовнішні переривання. Запити на безвекторні переривання поступають на спеціальні входи процесо­рів. Ці запити мають більший пріоритет, ніж запити на векторні пе­реривання. Механізм обробки безвекторних переривань закладений в процесорі на мікропрограмному або апаратному рівні.

У цілому, механізм забезпечує перехід до підпрограми за визначеної адреси (із запам'ятовуванням адреси повернення і стану перерваної програми). Безвекторні переривання, як правило, використовуються процесо­ром на його локальній магістралі, причому, входи запитів мають ви­значене призначення (відключення живлення, спрацьовування зов­нішнього таймера і таке інше).

Зовнішні векторні переривання є важливим засобом синхроні­зації процесів в мікропроцесорній системі. За допомогою сигналів зовнішніх переривань пристрої системи оповіщають один одного про готовність до передачі даних або про змінення режимів роботи, тобто обмінюються управляючою інформацією. Якщо для прийому сигналів переривань в процесорах передбачені спеціальні входи, то для формування сигналів вимоги переривання можуть знадобитися додаткові засоби.

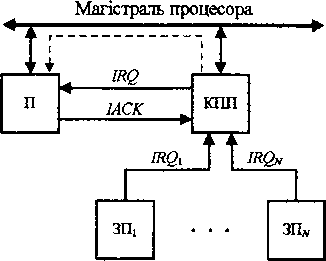
Зовнішні векторні переривання реалізуються за допомогою спе­ціального контролера наступним чином.

По запитах від зовнішніх пристроїв контролер переривань із урахуванням системи пріоритетів видає на процесор сигнал вимоги переривання IRQ. Умовою формування такого сигналу є готовність пристроїв до взаємодії з процесором і відсутність маскування запи­тів з боку процесора.

Отримавши сигнал IRQ, процесор завершує до кінця виконання чергової команди, видає на контролер сигнал підтвердження пере­ривання ІАСК і зчитує з шини даних вектор, який виставляє на шину контролер переривання.

Процесор зберігає (зазвичай в стеку) адреси повернення і стан програми, що тимчасово перервана, обчислює адресу переходу, ви­користовуючи для цього отриманий вектор переривання й після цьо­го здійснює перехід на першу команду підпрограми обробки пере­ривання. Процесор виконує підпрограму. Остання команда підпрог­рами є спеціальною командою повернення з переривання, яка відно­влює стан перерваної основної програми і передає ш управління.

Схема підключення централізованого контролера пріоритетних переривань (КПП) показана на рис. 12. Зовнішні пристрої (ЗП), в числі яких можуть бути і інші процесори, формують запити на пере­ривання IRQ. Особливість такої схеми полягає в тому, що підклю­чення ЗП до магістралі процесора не є обов’язковим. Це пояснюється тим, що вектор V в процесор передає КПП, а не зовнішні пристрої. Завдяки цьому через переривання можуть взаємодіяти процесори, підключені до різних магістралей.



**Рис. 12. Система с централізованим контролером пріоритетних переривань**

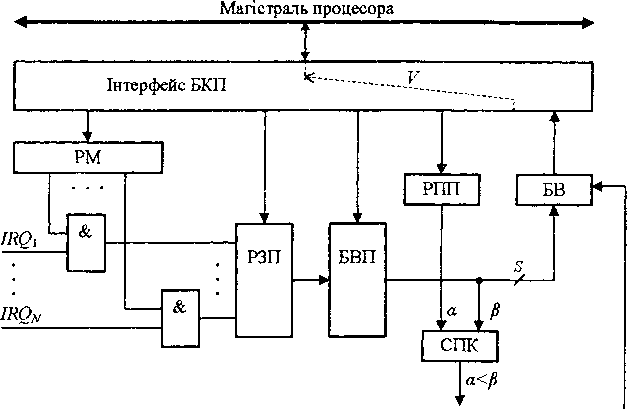


Рис. 13. Централізований КПП

Можливий варіант побудови централізованого КПП показаний на рис. 13. Контролер підключений до магістралі процесора через інтерфейс, який забезпечує процесору доступ до регістру маски (РМ) і регістру поточного пріоритету (РПП). Адреси вказаних регіс­трів включені в адресний простір процесора. Записом слова маски в регістр РМ процесор може дозволити або заборонити переривання від певних джерел. Незамасковані запити IRQ, записуються в регістр запитів переривань (РЗП). Блок вибору пріоритету (БВП) формує код самого старшого пріоритету, який в схемі порівняння кодів (СПК) порівнюється з поточним пріоритетом - пріоритетом викону­ваної процесором програми. Якщо запрошений пріоритет вище по­точного, то формується сигнал IRQ. Далі, у відповідь сигналу ІАСК, через буфер вектора (БВ) і інтерфейс в магістраль видається вектор, який приймається процесором. У регістр РПП записується новий код пріоритету. У даному контролері роль вектора виконує код номера запиту.

До достоїнств централізованих контролерів слід віднести на­ступне:

* можливість динамічно змінювати стратегію обслуговування заявок;
* швидке вибіркове маскування запитів на переривання.

Централізований КПП потенційно дозволяє забезпечувати різні дисципліни обслуговування заявок, оскільки всі заявки поступають в один пристрій. Для забезпечення пріоритетного обслуговування застосовується блок вибору пріоритету. У простому випадку в якості БВП використовується пріоритетний шифратор (ПШ), який формує код старшого рівня пріоритету. При цьому всі запити мають фіксо­вані пріоритети, рівень яких визначається номером входу КПП. За рахунок ускладнення схеми можна забезпечити інші дисципліни об­слуговування заявок. Найчастіше застосовують циклічну зміну рів­нів пріоритетів, що забезпечує гарантоване обслуговування будь- якого запиту на певному проміжку часі, причому, незалежно від ін­тенсивності запитів.

Оскільки регістр маски включений в адресний простір процесо­ра, то за умови, що число запитів не перевищує розрядності шини даних (що зазвичай виконується), процесор може за одне звернення до цього регістра записати в нього будь-яку маску.

До недоліків централізованих КПП слід віднести:

* велику кількість ліній запитів в шині управління (що дорів­нює кількості ЗП);
* обмеження на максимальне число джерел переривань;
* можлива неоднорідність процесорних модулів.

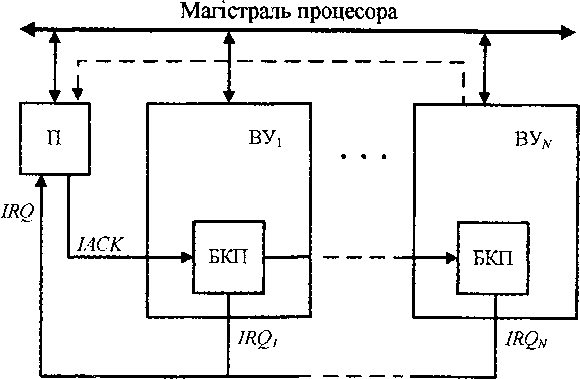
Останній з вказаних недоліків обумовлений тим, що в централі­зованих системах приймачем переривань зазвичай є управляючий процесор. Отже, за необхідності використання в системі резерву апа­ратури на рівні модулів буде потрібно різні типи резерву.

Вказані недоліки обмежують можливості застосування цього способу при побудові однорідної модульної МПС.

Під час використання розподіленого контролера переривань (рис. 14) все ЗП повинні бути підключені до магістралі процесора. Це обумовлено тим, що вектор переривання на шину даних в даному випадку видає сам активний ЗП.

До складу кожного ЗП включений блок контролера переривань БКП, який видає сигнал запиту IRQ, на загальну лінію IRQ. Техно­логічні особливості елементної бази повинні допускати таке об'єд­нання виходів елементів (наприклад, використовуються елементи з відкритим колектором, а сигнали запитів мають активний низький рівень). У відповідь сигнал процесора ІАСК розповсюджується по­слідовно через елементи БКП, створюючи так званий пріоритетний ланцюжок або "гірлянду" (daisy chain). Елементи ланцюжка в кож- ному БКП пропускають сигнал ІАСК або розривають ланцюжок. Пріоритетний ланцюжок розривається на першому (по шляху розпо­всюдження сигналу) активному ЗП, який виставив сигнал запиту IRQ. Даний активний ЗП видає на шину даних вектор переривання, який приймається процесором.

Приклад побудови БКП показаний на рис. 15.



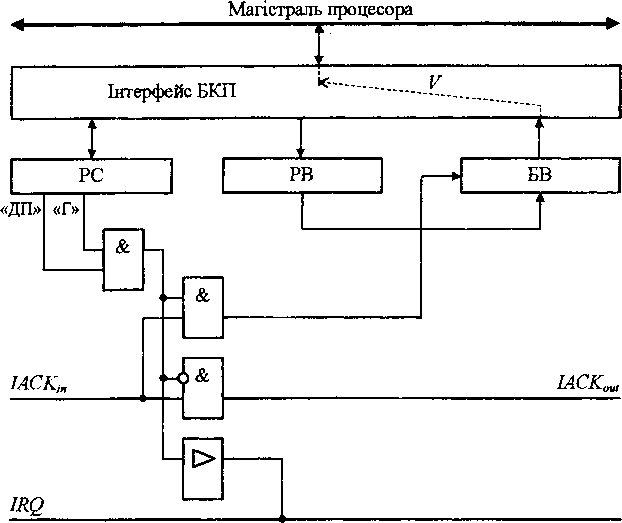
**Рис. 14. Система с децентралізованим КПП**

Блок БКП містить доступні для процесора регістр стану (РС) і регістр вектора (РВ). Під час ініціалізації режиму роботи системи процесор записує в регістр РВ вектор переривання, а в регістр РС - біт дозволу переривання (ДП). Якщо ЗП готовий до взаємодії з про­цесором, то в регістрі РС встановлюється біт готовності «Г». Це встановлення виконується засобами внутрішнього управління ЗП. За збігу сигналів «Г» і «ДП» формується запит IRQ, який через елемент узгодження поступає на лінію Вхідний для кожного блоку сигнал ІАСКіп передається на вихід ІАСКоut або забезпечує видачу век­тора через буфер вектора (БВ), що визначається значенням IRQ.

До достоїнств розподілених КПП можна віднести:

* невелику кількість ліній зв'язку в шині управління;
* простоту нарощування числа ЗП.

Завдяки вказаним достоїнствам розподілені контролери нахо­дять широке застосування в МПС системах.



**Рис. 15. Блок контролера переривань розподіленого КПП**

Недоліками розподілених КПП є:

* велика кількість звернень процесора до магістралі під час іні­ціалізації системи;
* використання фіксованих рівнів пріоритетів запитів, які не можна змінювати динамічно.

Перший недолік обумовлений необхідністю запису біта дозволу переривання в кожен блок окремо. Коли ініціалізація режимів здійс­нюється рідко, цей недолік можна вважати неістотним.

Використання фіксованих рівнів пріоритетів не забезпечує га­рантованого обслуговування заявок на певному відрізку часу. Заявки з низьким рівнем пріоритету за великої інтенсивності заявок з висо­кими пріоритетами можуть не виконуватися тривалий час. Ця обста­вина може привести до уповільнення обчислювального процесу, а іноді - до тупикової ситуації.

Обидва розглянуті підходи до реалізації зовнішніх векторних переривань можуть поєднуватися. Наприклад, декілька пристроїв об’єднуються ланцюжком «гірлянди», а декілька ланцюжків, у свою чергу, замикаються на централізованому контролері.

1. **Програмна частина**

## Підпрограма множення

### Блок-схема алгоритму



## Блок-схема алгоритму обчислення основної функції

Початок

Введення Х1, Х2, X3, X4

X1 – X2

X3 \* X4

X1 – X2 + X3 \* X4

(X7+X8)

X1 – X2 + X3 \* X4 + (X7+X8)^2

Кінець

(X7+X8)^2

## Лістинг програми

; X = X1 – X2 + X3 \* X4 + (X7+X8)^2

; спосіб множення - 4, розрядність операндів - 16

; операнди подані в ПК

;

; Результат обчислення функції розміщується за адресою F0h.F1h.F2h.F3h

; X2^2

SEL RB1

MOV R1, X2\_h

MOV R2, X2\_h

SEL RB0

MOV R1, X2\_l

MOV R2, X2\_l

CALL multiply

; запис результату в F0h.F1h.F2h.F3h

SEL RB1

MOV R0, #0F0h

MOV A, R4

MOV @R0, A

INC R0

SEL RB0

MOV A, R4

SEL RB1

MOV @R0, A

SEL RB1

INC R0

MOV A, R5

MOV @R0, A

INC R0

SEL RB0

MOV A, R5

SEL RB1

MOV @R0, A

; X3 \* X4

SEL RB1

MOV R1, X3\_h

MOV R2, X4\_h

SEL RB0

MOV R1, X3\_l

MOV R2, X4\_l

CALL multiply

; запис результату в F4h.F5h.F6h.F7h

SEL RB1

MOV R0, #0F4h

MOV A, R4

MOV @R0, A

INC R0

SEL RB0

MOV A, R4

SEL RB1

MOV @R0, A

SEL RB1

INC R0

MOV A, R5

MOV @R0, A

INC R0

SEL RB0

MOV A, R5

SEL RB1

MOV @R0, A

CALL adding

SEL RB1

MOV R1, X5\_h

MOV R2, X5\_h

SEL RB0

MOV R1, X5\_l

MOV R2, X5\_l

CALL multiply

; запис результату в F4h.F5h.F6h.F7h

SEL RB1

MOV R0, #0F4h

MOV A, R4

MOV @R0, A

INC R0

SEL RB0

MOV A, R4

SEL RB1

MOV @R0, A

SEL RB1

INC R0

MOV A, R5

MOV @R0, A

INC R0

SEL RB0

MOV A, R5

SEL RB1

MOV @R0, A

CALL adding

SEL RB1

MOV R0, #0F4h

MOV A, #0

MOV @R0, A

INC R0

SEL RB0

MOV A, #0

SEL RB1

MOV @R0, A

SEL RB1

INC R0

MOV A, X6\_h

MOV @R0, A

INC R0

SEL RB0

MOV A, X6\_l

SEL RB1

MOV @R0, A

MOV R0, #0F6h

MOV A, @R0

JB7 add\_minus

JMP add\_plus

add\_minus:

ANL A, #7Fh

MOV @R0, A

MOV R0, #0F4h

MOV A, @R0

ORL A, #80h

MOV @R0, A

add\_plus:

CALL adding

JMP ending

multiply:

; Очищення регістрів

SEL RB0

MOV A, R3

XRL A, R3

MOV R3, A

MOV A, R4

XRL A, R4

MOV R4, A

MOV A, R5

XRL A, R5

MOV R5, A

SEL RB1

MOV A, R3

XRL A, R3

MOV R3, A

MOV A, R4

XRL A, R4

MOV R4, A

MOV A, R5

XRL A, R5

MOV R5, A

SEL RB1

CLR F1

MOV A, R1

XRL A, R2

JB7 sign\_minus

JMP sign\_plus

sign\_minus:

CPL F1 ; встановлення ознаки F1

sign\_plus:

MOV A, R1

ANL A, #7Fh

MOV R1, A

MOV A, R2

ANL A, #7Fh

MOV R2, A

CLR C

SEL RB1

MOV A, R2

RRC A

MOV R2, A

SEL RB0

MOV A, R2

RRC A

MOV R2, A

SEL RB1

MOV A, R3

RRC A

MOV R3, A

loop:

SEL RB1

MOV A, R1

CPL A

JB7 zero

; Z: R4\_RB1.R4\_RB0.R5\_RB1.R5\_RB0

; Y: R2\_RB1.R2\_RB0.R3\_RB1.R3\_RB0

SEL RB0

CLR C

MOV A, R5

ADDC A, R3

MOV R5, A

SEL RB1

MOV A, R5

ADDC A, R3

MOV R5, A

SEL RB0

MOV A, R4

ADDC A, R2

MOV R4, A

SEL RB1

MOV A, R4

ADDC A, R2

MOV R4, A

zero:

CLR C

SEL RB0

MOV A, R1

RLC A

MOV R1, A

SEL RB1

MOV A, R1

RLC A

MOV R1, A

CLR C

SEL RB1

MOV A, R2

RRC A

MOV R2, A

SEL RB0

MOV A, R2

RRC A

MOV R2, A

SEL RB1

MOV A, R3

RRC A

MOV R3, A

SEL RB0

MOV A, R3

RRC A

MOV R3, A

SEL RB1

MOV A, R1

JNZ loop

JF1 rez\_minus

JMP rez\_plus

rez\_minus:

SEL RB1

MOV A, R4

ORL A, #80h

MOV R4, A

rez\_plus:

RET

adding:

SEL RB0

MOV R0, #0F0h

MOV A, @R0

JB7 sign\_add\_first\_minus

JMP sign\_add\_first\_plus

sign\_add\_first\_minus:

ANL A, #7Fh

MOV @R0, A

MOV R0, #0F3h

MOV A, @R0

CPL A

MOV @R0, A

DEC R0

MOV A, @R0

CPL A

MOV @R0, A

DEC R0

MOV A, @R0

CPL A

MOV @R0, A

DEC R0

MOV A, @R0

CPL A

MOV @R0, A

CLR C

MOV R0, #0F3h

MOV A, @R0

ADDC A, #1

MOV @R0, A

DEC R0

MOV A, @R0

ADDC A, #0

MOV @R0, A

DEC R0

MOV A, @R0

ADDC A, #0

MOV @R0, A

DEC R0

MOV A, @R0

ADDC A, #0

MOV @R0, A

sign\_add\_first\_plus:

MOV R0, #0F4h

MOV A, @R0

JB7 sign\_add\_second\_minus

JMP sign\_add\_second\_plus

sign\_add\_second\_minus:

ANL A, #7Fh

MOV @R0, A

MOV R0, #0F7h

MOV A, @R0

CPL A

MOV @R0, A

DEC R0

MOV A, @R0

CPL A

MOV @R0, A

DEC R0

MOV A, @R0

CPL A

MOV @R0, A

DEC R0

MOV A, @R0

CPL A

MOV @R0, A

CLR C

MOV R0, #0F7h

MOV A, @R0

ADDC A, #1

MOV @R0, A

DEC R0

MOV A, @R0

ADDC A, #0

MOV @R0, A

DEC R0

MOV A, @R0

ADDC A, #0

MOV @R0, A

DEC R0

MOV A, @R0

ADDC A, #0

MOV @R0, A

sign\_add\_second\_plus:

CLR C

SEL RB1

MOV R0, #0F3h

MOV R1, #27h

MOV R2, #4h

add\_loop:

MOV A, @R0

ADDC A, @R1

MOV @R0, A

DEC R0

DEC R1

DJNZ R2, add\_loop

RET

;--------------------------------------------------------

ending:

END

# ВИСНОВОК:

Розроблена мікропроцесорна система (МПС), ядром якої є процесор 1816ВЕ48. Виконали оцінку ефективності прийнятих технічних рішень.

У склад розробленої МПС входять процесор (П), основна пам’ять (ОП), яка містить ОЗП і ПЗП, ППА (ВВ55), а також зовнішні пристрої (ЗП) зовнішньої пам’яті даних, зовнішньої пам’яті програми, периферійного адаптера, зовнішніх пристроїв, контролер переривання і прямого доступу до пам’яті.

# 

# ЛІТЕРАТУРА

1. <http://digteh.ru/MCS48/pam_cntr.php>
2. <http://www.kosmos.mk.ua/usmtu/mpk/k1816/k1816org0.html>
3. Жабін В.І., І.А. Жуков, В.В. Ткаченко, І.А. Клименко. Мікропроцесорні системи. СПД: Гуральник О.Ю. ,2009.-492с.
4. Самофалов К.Г., Корнійчук В.И., Тарасенко В.П., Жабін В.И. Цифрові ЕОМ. Практикум.- К.: Вища.шк. 1989. - 124 с.