**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
Запорізький національний технічний університет

# МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни «Вбудовані комп’ютерні системи» у середовищі MultiSim, LabView для студентів усіх форм навчання спеціальності 8.091503«Спеціалізовані комп'ютерні системи» усіх форм навчання (частина ІІ)

2019

Методичні вказівки до виконання лабораторних робіт з дисципліни «Вбудовані комп'ютерні системи» у середовищі MultiSim, LabView для студентів усіх форм навчання спеціальності 8.091503 «Спеціалізовані комп'ютерні системи» усіх форм навчання (частина II) / Укладач: Проскурін М.П. - Запоріжжя: ЗНТУ, 2019- 80 с.

Укладач: М.П. Проскурін, к.т.н., доцент

Рецензент: І.Я. Зеленьова, к.т.н., доцент

Відповідальний за випуск: М.П. Проскурін, к.т.н., доцент.

Висловлюємо окрему щиру подяку керівництву фірми National Instruments (NI), представництву NI в Україні (м. Київ) та особисто його керівнику Гладкову М.В. за сприяння в отриманні учбових версій ліцензійних програм MultiSim v.12, LabView v.12, AWR для ознайомлення і навчання студентів із сучасними САПР та проведення курса лабораторних, дослідницьких робіт по навчальних дисциплінах на II і V курсах в ІІРЕ ЗНТУ в період 01.2017-12.2018рр.

Затверджено
на засіданні кафедри
«Комп'ютерні системи і мережі»
Протокол №10
від « 22 » квітня2019р.

Рекомендовано до видання
НМК факультету
Протокол № 9
від « 26 » квітня2019р.

Зміст

**Перелік використовуваних скорочень**

АЦП - аналого-цифровий перетворювач
БД - база даних
ВКС - вбудовані комп`ютерні системи
ВП - віртуальні пристрої
ЗМР - значення молодшого розряду
КМОН - комплементарна логіка на транзисторах метал-оксид-напівпровідник
ЛР - лабораторная работа
ЛКМ - ліва кнопка миші
МК - мікроконтролери
МП - мікропроцесорні пристрої
ОЗП - оперативне запам'ятовуючий пристрій
ПЗ - програмне забезпечення
ПЗП - постійне запам'ятовуючий пристрій
ПК - персональний комп'ютер
ПКМ - права кнопка миші
ППП - пакет прикладних програм
ТЗ - технічне завдання
ТТЛ(Ш) - транзисторних-транзисторна логіка (Шоткі)
САПР - система автоматичного проектування
ФФК - формат з фіксованою комою
ФПК - формат з плаваючою комою
ЦАП - цифро-аналоговий перетворювач
ЦА - цифровий автомат
ЕОЗ - електронно-обчислювальні засоби
NI - National Instruments
EWB - Electronics WorkBench – «робочий стіл» електронщика
LV - LabView
MS - MultiSim
SFR - registr of special functions) - регістр спеціальних функції

## 5. Лабораторна робота № 4. Підключення зовнішньої пам'яті до МК і її тестування

**Мета роботи**: Розробити схеми підключення МК до зовнішньої пам'яті двох типів і протестувати ІС пам'яті в MS і LV.

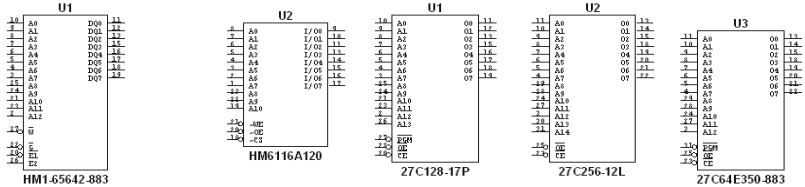
### 5.1 Загальні відомості про типи зовнішньої пам`яті

Зовніння пам`ять МК використовується для збільшення їх ефективності, можливостей і/або для розширення кількості і якості алгоритмів виконання завдань. В основному МК можуть доповнюватись схемами RAM, ROM (EPROM) або їх комбінаціями. Їх функціональна, фізична і байтова організація повинна співпадати з можливостями МК організувати обмін даними по інтерфейсу.

#### 5.1.1 Особливості підключення МК до зовнішньої пам'яті і / або периферійних пристроїв

МК 8051 може працювати із зовнішньою пам'яттю даних ємністю до 64 КБ, построеннной на одній або декількох мікросхемах статичної пам'яті. В БД MS є ІС для організації Random Access Memory (RAM або ОЗП) з байтовою організацією обсягом 2Кх8 і 8Кх8 біт (рис. 5.1а, б).

Такі мікросхеми мають 8 двонаправлених виводів даних (D0-D7), 11 або 13 адресних входів (A0-A10 або A0-A12). Вхід WE(W) визначає характер звернення: якщо на ньому встановлена 1, то здійснюється читання з вибраної комірки, при WE = 0 в комірку буде записана інформація. Вхід CS (E1, E2) активізує ІС памяті: коли на вході CS встановлена 1, вона виключена, при CS = 0 допускається будь-яке звернення до пам'яті. Нульовий сигнал на вході OE (G) дозволяє роботу вихідний шини даних мікросхеми. В БД Multisim також представлені ІС для організації Read Only Memory (ROM або ПЗУ) з організацією 32Кх8, 8Кх8, 16Кх8, рис. 5.1в - д. Ці ІС в робочому режимі допускають тільки читання інформації і їх виводи аналогічні мікросхем RAM, крім виведення PGM, що відповідає за програмування. Робота ІС ОЗУ, ПЗУ розглянута, наприклад, в [8].




а б в г д

Рис. 5.1 - ІС RAM (ОЗУ) 8Кх8 -а; 2Кх8 -б; EPROM (ППЗУ) 16Кх8 - в, ROM (ПЗУ) 32Кх8 - г; EPROM (ППЗУ) 8Кх8 - д

В МК-51 існує чотири багатофункціональних 8-бітових порти введення/виведення P0, P1, P2, P3, призначених для обміну інформацією з різними зовнішніми пристроями (ЗП) і для виконання спеціалізованих функцій, таких як підключення зовнішньої пам'яті програм, даних, програмування внутрішнього ПЗП та ін. Кожен порт може адресуватися як побайтово, так і побітно, за конкретними фізичними адресами. При підключенні до МК зовнішньої пам'яті через порт P0 виводиться молодший байт адреси, а також передається і приймається в мікроконтролер байт даних (у мультиплексованому режимі). У 1 і 2 тактах машинного циклу при звернені до зовнішньої пам'яті на лініях P0 активізується адресна інформація A0-A7 при високому рівні сигналу ALE, а потім на цих же лініях з'являється сигнал D0-D7 (при низькому рівні сигналу ALE). Для фіксації байту адреси протягом усього машинного циклу використовуються регістри-засувки, наприклад, 74LS373N, інформація в яких фіксується по спаду сигналу на його вході ENG.

Через порт P2 виводиться старший байт адреси (розряди A8-A15) зовнішньої пам'яті програм і даних. Для кожного з бітів порту P3 є ряд альтернативних функцій. Сигнали спроб запису (WR#) і читання (RD#) зовнішньої пам'яті формуються на лініях P3.6 і P3.7 відповідно. Альтернативні функції всіх портів реалізуються тільки в тому випадку, якщо у відповідний розряд фіксатора-засувки порту записана логічна «1», інакше на відповідному виводі буде присутній «0».

Кожен висновок портів P0-P3 може використовуватися як вхід або вихід незалежно від інших. Для налаштування лінії порту на введення інформації необхідно в відповідний розряд порту записати «1», а для використання в якості виходу - «0». При системному скиданні в регістрах засувках всіх портів встановлюється значення FFh.

### 5.1.2 Порядок виконання лабораторної роботи

#### 5.1.2.1 Створення схемного проекту

Відкриваємо і зберігаємо новий схемний проект, наприклад Circuit2. Розміщуємо на робочому полі МК-51, ІС пам'яті ємністю 2 (8 ... 16 Кбайта (Place - Component - MCU-RAM), регістр-засувку, наприклад 4037ВР, яку можна знайти в групі CMOS в сімействі CMOS_5V, землю і живлення (Place - Component - Sources - POWER_SOURCES).

Після вибору компонентів з БД і розміщення їх на схемі, необхідно з'єднати компоненти між собою. У програмі Multisim дія миші на схемі залежить від положення курсора. Зовнішній вид курсора змінюється в залежності від того, на який об'єкт він наведений (рис. 5.2 ) Для того щоб провести з'єднувальний провід між елементами, необхідно клікнути ЛКМ по виводу одного елемента, потім, не відпускаючи кнопку, провести

з'єднання до виводу іншого елемента. Після появи з'єднувального провідника між елементами MS автоматично присвоїть йому номер. Нумерація ліній збільшується послідовно, починаючи з 1.

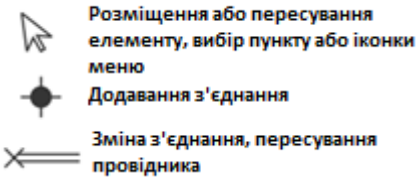



Рис. 5.2 - Види курсора

Заземляючі дроти завжди мають номер 0, це вимога пов'язана з роботою прихованого емулятора SPICE. Щоб змінити нумерацію з'єднання або привласнити йому логічне ім'я, необхідно двічі клікнути по сполучному провіднику (рис. 5.3).

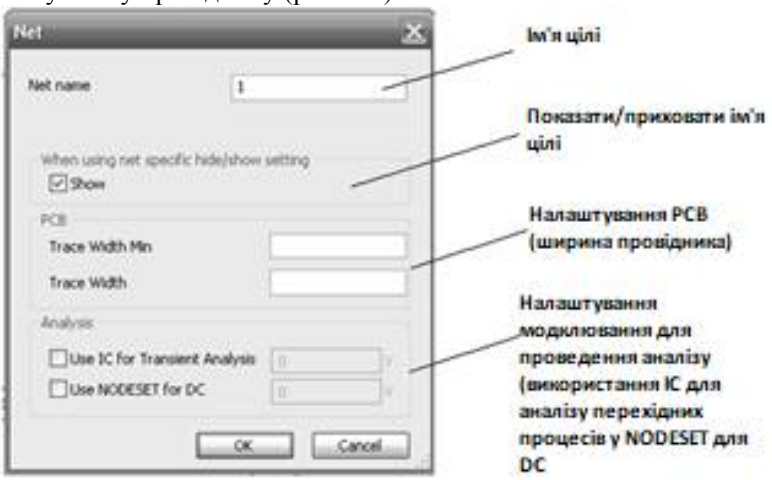



Рис. 5.3 - Встановлення ланцюга

У MS є функція автоматичного з'єднання виводів між собою і з провідниками. При додаванні компонента до існуючої мережі з'єднань необхідно, щоб виводи компонента торкались існуючої мережі, або помістити компонент паралельно з'єднанню.

Для розміщення з'єднуючої шини в схемном проекті необхідно вибрати пункт меню Place - Bus або значок на панелі компонентів або гарячу клавішу Ctrl-U.

Розглянемо підключення необхідних компонентів до шини на прикладі компонента U3 (рис. 5.8). За допомогою ЛФМ розміщуємо шину навпроти виводів, необхідних для підключення компонентів, наприклад, навпроти виводів D0-D7 елемента U3, потім фіксуємо положення шини ПКМ. Далі в контекстному меню компонента U3, що з'єднується з шиною, натискаємо Bus Vector Connect (рис. 5.4).

Зліва вікна з'єднань представлені необхідні дані про компоненти (позначення, положення виводів, виводи), праворуч - про шину (позначення, шинні лінії). Вибираємо в підміню Pins виводи компонента 1D-8D для підключення до шини, утримуючи клавішу Shift і за допомогою стрілок відправляємо їх в нижнє віконце. Справа вікна з'єднань вибираємо в поле Name шину, в нашому випадку Bus1. Далі необхідно **сформувати вектори** (лінії шини), Multisim пропонує для цього два способи: ручний і автоматичний. **Формування векторів вручну.** Натискаємо кнопку Add buslines (рис.5.4) для завдання векторів, з'являється відповідне вікно (рис. 5.5).

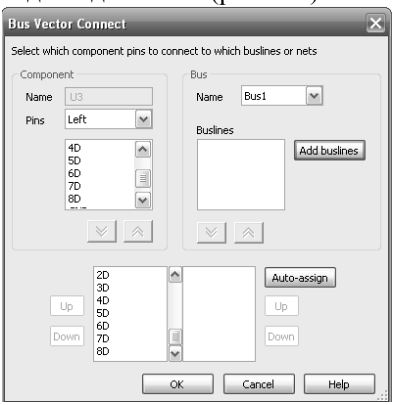



Рис. 5.4 - Вікно з'єднання компонента з шиною

Тут вказуються префікс (мітка), наприклад, *ln*, початкове значення вектора, інкремент і кількість векторів (в нашому випадку їх 8). Після натискання на кнопку ОК (рис. 5.5) отримані лінії шини (вектори) відобразяться в попередньому вікні. Кількість ліній шини має бути еквівалентним обраним для підключення до шини виводів. Далі виділяємо отримані лінії шини і натискаємо кнопку ОК (рис. 5.4).

**Автоматичне формування векторів**. Вектори вводяться автоматично при натисканні кнопки Auto-assign (рис. 5.4).

На завершення натискаємо кнопку ОК. У такому ж порядку підключаємо до шини виводи МК51.

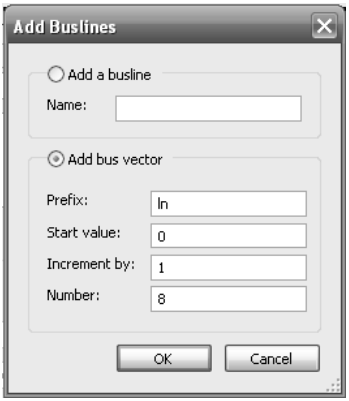



Рис. 5.5 - Вікно заддання ліній шин вручну

При підключенні зовнішньої пам'яті аналогічним чином малюємо шину навпроти необхідних виводів елемента U2 і фіксуємо її. Ця шина автоматично буде названа Bus2. Для об'єднання шини Bus2 до шини Bus1 необхідно вибрати в контекстному меню Bus2 пункт Properties (рис. 5.6).

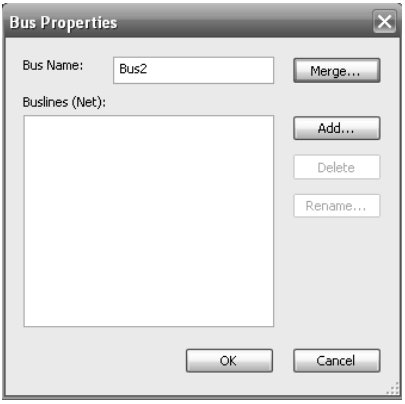



Рис. 5.6 - Вікно властивостей шини

У вікні, натискаємо кнопку Merge. У наступному вікні, що відкрилося (рис. 5.7) зліва вказана перша шина, призначена для об'єднання Bus1, а справа друга - Bus2. У вікні об'єднування шин в поле Name другої шини вибираємо Bus1, після чого у віконці Buslines з'являться задані раніше вектори Bus1. Нижче вказана об'єднана шина з її векторами (рис. 5.8).

Далі у вікні об'єднання шин натискаємо кнопку Merge і у вікні властивостей шини кнопку ОК. Як і раніше, вибираються виводи компонента і вектори шини, тільки в цьому випадку при натисканні кнопки ОК ППП MS попереджає, що фрагмент шини, до якої необхідно зробити підключення, вибирається подвійним кліком ЛКМ. У вікні розробки вказана вся структура створеного проекту: готова схема представлена на рис. 5.8.

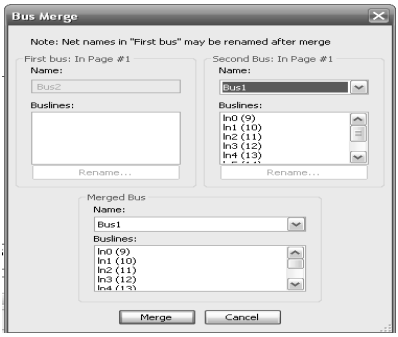



Рис. 5.7 - Вікно об'єднання шин

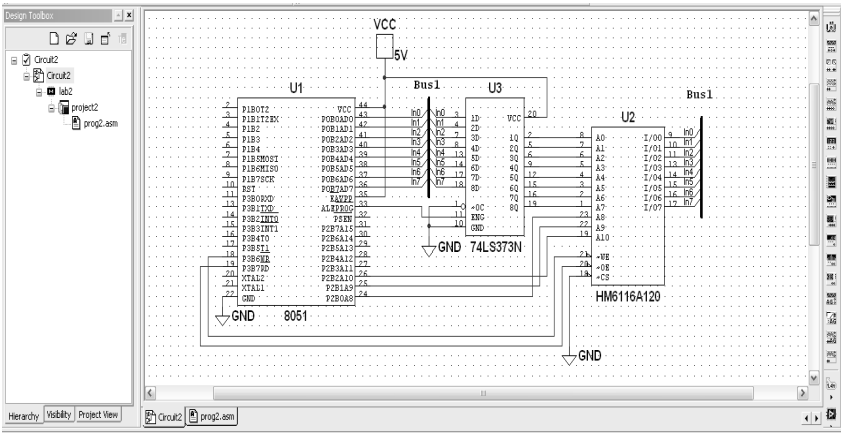



Рис. 5.8 - Схема підключення зовнішньої пам'яті до МК-51

У властивостях МК на вкладці Value в поле Built-in External RAM необхідно вказати обсяг підключеної зовнішньої пам'яті (рис. 5.9).

*Розробка програмного файлу.* Активуємо закладку програмного файлу

prog2.asm (prog2.c), клацнувши по ній ЛКМ або вибравши програмний файл у вікні розробки.
\

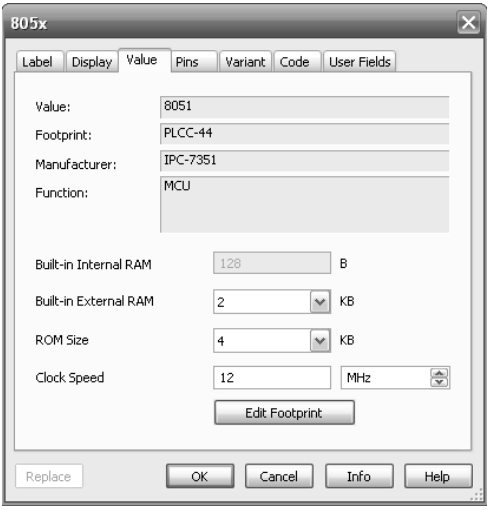



Рис. 5.9 - Вікно властивостей МК, вкладка Value

Якщо є готова програма, розробка якої виконана, наприклад, в середовищі PRO View [1], то вставляємо отриманий асемблерний (або С) програмний код і зберігаємо його.

Покажемо програму тестування зовнішньої пам'яті на наступному прикладі: потрібно перевірити 255 байт зовнішньої пам'яті, починаючи з комірки 00h, використовуючи тестовий набір 055h.

Асемблерний файл програми:

```
$ MOD51          ; Підключення МК-51
org 00h          ; починаємо програму з адреси 00h
mov dptr, # 00h; завантажити в регістр-покажчик число 00h
mov r2, # 0ffh   ; завантажити в регістр R2 число 0ffh (лічильник циклу)
mov r1, # 055h   ; завантажити в регістр R1 число 55h
test:
mov a, r1        ; завантажити акумулятор ACC операндом з регістра R1
movx @ dptr, a ; переслати в комірку зовнішньої пам'яті XRAM вміст
                 акумулятора ACC
movx a, @ dptr ; зчитати в акумулятор вміст поточної комірки зовнішньої
                 пам'яті
xrl a, # 055h    ; операція XOR ліченого і початкового операнду, якщо 0
```

```
                в Акк, то осередок працює нормально
 jnz error      ; помилка - вихід з програми
 inc dptr       ; інкремент  вмісту регістра DPTR - перехід до наступної
                адреси
 djnz r2, test  ; відняти 1 з вмісту регістра R2 і перейти по мітці, якщо
                в комірці не 0
 error:
END
```

Вибираємо в меню MCU: MCU 8051 U1, пункт Build, який дозволяє відкомпілювати і відредагувати програмний файл. У вікні повідомлень на екрані відображається інформація про помилки та попередження. Якщо є помилки, у вікні повідомлень вказані номери рядків, в яких вони знаходяться. Для відображення нумерації рядків необхідно вибрати меню MCU - Show lines numbers.

Моделювання роботи схеми проводиться через меню Simulate - Run, або гарячу клавішу F5 або іконку на панелі Моделювання - . При цьому відкривається вікно відладчика. Для покрокового моделювання використовується пункт Step into в меню MCU, гаряча клавіша F11 або іконка на панелі Моделювання.

Кнопка Pause або гаряча клавіша F6 ініціюють паузу моделювання, при натисканні якої можливо подивитися, на якому етапі перебуває симуляція.

кнопка Stop зупиняє виконання моделювання. Перегляд пам'яті мікроконтролера активується через меню

MCU - MCU 8051 U1 - Memory view, також можна використовувати пункт MCU Windows, де у спливаючому вікні можна відзначити галочкою необхідні для роботи вікна атрибути. При необхідності виконання програми до певної інструкції в Multisim передбачені точки зупину, встановити які можна купити через меню MCU - Toggle breakpoint або через панель Моделювання.

Кнопка Remove all breakpoints видаляє всі точки зупинки. Після виконання програмою інструкції djnz r2, test переглянемо результати записи в зовнішню пам'ять (рис. 5.10). У ACC записано число 00h, в R1 - 55h, в R2 - FEh, в першій клітинці зовнішньої пам'яті XRAM - 55h

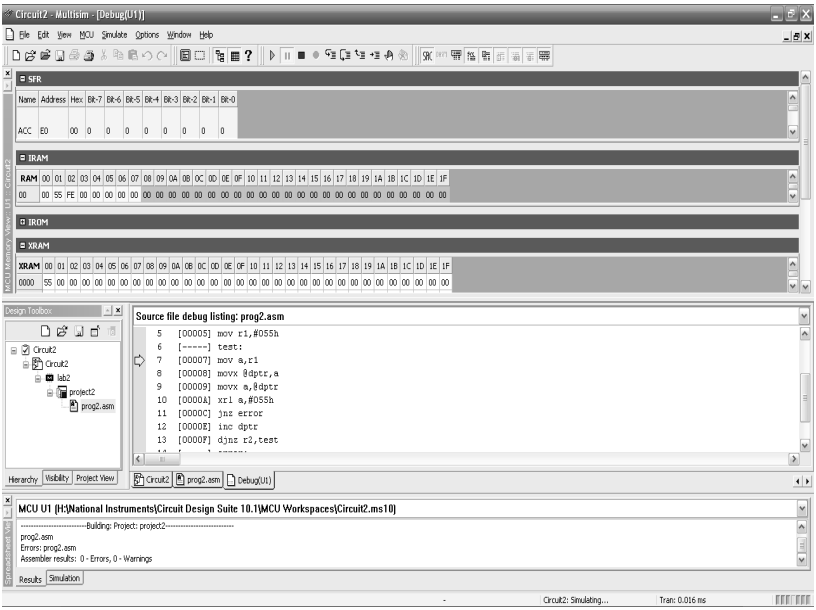



Рис. 5.10 - Результати моделювання після інструкції djnz r2, test

Після завершення моделювання в кожному комірку зовнішньої пам'яті записано число 55h, а регістр R2 матиме нульове значення.

Робота з С-файлом аналогічна. Нижче розглядається варіант тестування зовнішньої пам'яті з адресами від 400h до 7FFh.

```
#include <8051.h> // підключаємо заголовки
void main () // головна функція, точка входу в про- граму
        {
        int i;                          // оголошуємо змінні
        char xdata * ptr;  // змінна ptr - вказівник на адресу комірки пам'яті
                                зовнішнього ОЗП
// символьні змінні test, nabor (байтові змінні)
        char test, nabor;
        nabor = 0x55;                   // тестовий набір

        ptr = (char xdata *) 0x400; // початкова адреса зовнішньої пам'яті
        for (i = 0; i <1024; i ++)      // будуть перевірені 1024 комірку пам'яті
        {
// в кожну клітинку пам'яті надсилається значення тестового набору
        *ptr = nabor;
// змінна test приймає значення вмісту комірки пам'яті test = * ptr;
```

```
// якщо test ≠ nabor, то кінець циклу (помилка), інакше перевіряється
наступна комірка пам'яті;
    if (test! = nabor)
    {
    break;
    }
    ptr ++;
    }
    }
```

З порівняння програмних кодів видно, що тестування великих масивів даних (> 255 байт) легше організувати на С, ніж на асемблері, так як в асемблерній програмі буде потрібна організація подвійних циклів.

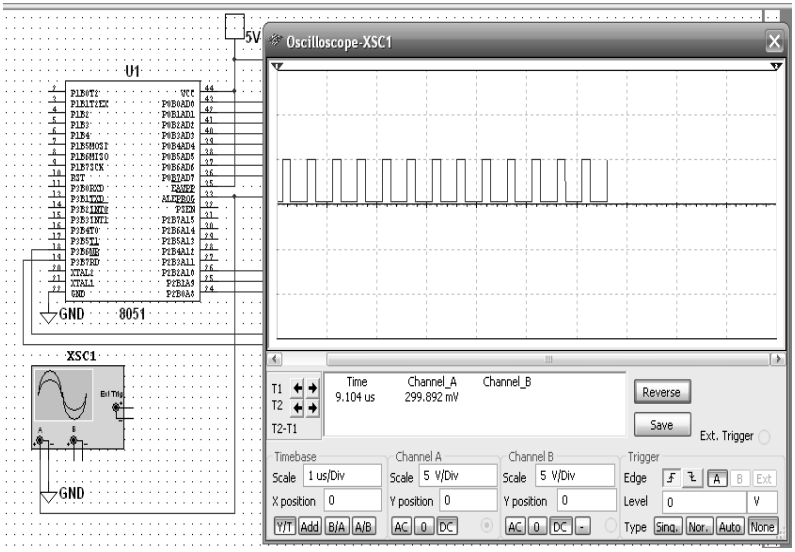



Рис. 5.11 - Підключення осцилографа в MS і його інструментальна панель під час моделювання

На закінчення ЛР використовуємо віртуальний осцилограф/аналізатор. Зробимо, наприклад, підключення осцилографа до даної схеми. Вибираємо на панелі інструментів іконку Oscilloscope і встановлюємо на робочій області. Символ осцилографа має два канали: Channel A і Channel B. Під'єднуємо «+» каналу A до виводу ALE / PROG мікроконтролера, «-» до землі. На панелі приладу необхідно

вказати розподіл тимчасової шкали, що дорівнює одному машинному циклу, тобто 1 мкс. Вибираємо по осі у ціну ділення для каналу А, приймемо її рівною 5,0 В. Після запуску моделювання видно процес генерації сигналу ALE, двічі за машинний цикл (рис. 5.11).

### 5.1.3 Завдання для ЛР і хід її виконання

1. Підключити до МК зовнішнє ОЗУ ємністю N Кбайт (табл. 5.1) і використовуючи тестовий набір XX, зробити тестування області пам'яті 1 Кбайт, починаючи з адреси ZZZ. Включити світлоіндикатор, якщо записаний і лічений з комірки пам'яті набори не збігаються. Підключити осцилограф зняти осцилограму з виведення 0 ALE МК.

Таблиця 5.1

Завдання 1

| параметри | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| N, кбайт | 8 | 2 | 8 | 8 | 8 | 2 |
| XX | 0AAh | 55h | 00h | 0FFh | 55h | 0AAh |
| ZZZ | 800h | 100h | 0C0h | F00h | 140h | 400h |

2. Підключити до МК зовнішнє ПЗУ ємністю N Кбайт (табл. 5.2). Підрахувати контрольну суму області пам'яті ємністю N Кбайт, починаючи з адреси ZZZ і порівняти її з константою, що знаходиться в резидентному ПЗУ. Якщо порівнювані величини співпадають, тестування ПЗУ пройшло успішно, інакше включити індикатор. Підключити осцилограф, зняти осцилограму з виведення ALE МК.

Таблиця 5.2

Завдання 2

| параметри | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|
| *N*, Кбайт | 8 | 16 | 32 | 8 | 16 | 32 |
| *K*, байт | 100 | 300 | 500 | 150 | 350 | 650 |
| *ZZZ* | 800h | 1000h | 2000 | 0C00h | 1000h | 400h |

**Для виконання ЛР№4 необхідно виконати наступне:**

- завантажити ППП MS і LV;
- реалізувати схему рис. 5.8 і її зв'язки між елементами (МК, ІС, живлення, заземлення, індикатори, ін.) на робочому столі MS, підключити вхідні і вихідні прилади (генератор, осцилограф, ін.);
- завантажити перевірену (откомпільовану) програму для тестування ОЗП, ПЗП (відповідно до варіанта завдання, табл.5.1, 5.2);

- отримати діаграми моделювання тесту, для чого підключити ВП осцилограф/цифровий аналізатор;

- використовуючи вихідну схему рис.5.11 (MS), створити її варіант/аналог в середовищі LV з ВП типу осцилограф/логічний аналізатор (ЛР1-3, ч.1), перевірити функціонування (див.способи 1 -4, п.2.1)

### 5.1.4 Зміст звіту

1. Найменування і мета ЛР.
2. Опис МК і ІС зовнішньої пам'яті (ОЗП, ПЗП), призначення виводів, короткі характеристики МК, ІС.
3. Копія програмного файлу (на асемблері і/або на С) з детальними коментарями.
4. Копія схемного файлу в MS і LV, моделювання із зазначенням режимів, позиційних позначень елементів і шин.
5. Отримані результати (діаграми, графіки, значення, ін.), Виводи по ЛР.

### 5.1.5 Питання для самоконтролю

1. Які функції виконує регістр - засувка при підключенні до МК зовнішньої пам'яті?
2. Назвіть альтернативну функцію, виконувану портом Р2.
3. Поясніть призначення сигналу PSEN (37 ніжка МК-51).
4. Поясніть, у чому відмінності в роботі МК при зверненні до зовнішньої пам'яті даних в порівнянні з внутрішньою?
5. Яке призначення виходу EA/Vpp МК-51?
6. Яким чином можна використовувати зовнішнє ОЗП/ПЗП для звернення до даних і/або до програмного коду?
7. Як підраховується контрольна сума заданої області пам'яті?
8. Поясніть призначення виводів CE(CS), OE мікросхем пам'яті.
9. Які конструктивно-технологічні відмінності однократно-програмованих мікросхем ПЗП (PROM) в порівнянні з репрограмованими ПЗП (EPROM- з ультрафіолетовим стиранням)?
10. Поясніть особливості тестування ІС ОЗП в порівнянні з ПЗП.
11. В чому схожість/особливість моделювання схем в MS і LV?

## 6. Лабораторна робота № 6. Організація генерації імпульсів і підрахунку заданих інтервалів часу

**Мета роботи:** навчитися реалізовувати необхідні часові інтервали для генерації і/або підрахунку внутрішніх/зовнішніх імпульсів (завдання типу конвеєр) на основі використання двох вбудованих таймерів/лічильників МК-51

### 6.1Основи налаштування і використання таймерів МК-51

До складу МК-51 входять два 16-розрядних таймера/лічильника Т/С0, Т/С1. Стан таймерів/лічильників відображається в програмно-доступних регістрах (TH0, TL0), (TH1, TL1), які розміщені в просторі SFR за адресами (8Ch, 8Ah), (8Dh, 8Bh).

Таймери/лічильники Т/С0 і Т/С1 можуть бути запрограмовані для роботи або в якості таймера, або в якості лічильника. Функція таймера полягає в рахунку числа машинних циклів, що слідують з частотою FOSC/12. Функція лічильника полягає у відстеженні числа переходів з 1 в 0 на відповідних входах Т0, Т1.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| GATE | C/T | M1 | M0 | GATE | C/T | M1 | M0 |

Рис. 6.1 - Структура регістра TMOD

Управління режимами роботи пристроїв Т/С0, Т/С1 здійснюється регістром TMOD (Timer/Counter Mode), який розташований за адресою 89h, до нього можливе звернення тільки байтом даних.

Регістр розбитий на два 4-розрядних підрегістра T0MOD і T1MOD (рис. 6.1)., Які відповідальні за управління Т/С0 (біти 0-3) і Т/С1 (біти 4-7):

TMOD.0 - M0. Молодший біт поля управління режимом Т/С0.

TMOD.1 - M1. Старший біт поля управління режимом Т/С0.

При цьому режим таймера Т/С0 програмується відповідно до значень M0 і M1 наступним чином:

| M0 | M1 | режим | M0 | M1 | режим |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 2 |
| 1 | 0 | 1 | 1 | 1 | 3 |

TMOD.2 - C/T. Вибір функції таймера або лічильника Т/С0.

При C/T = 0 вибирається функція таймера, інакше-лічильника.

TMOD.3 - GATE. Прапорець управління роботою Ст0. При GATE = = 1 робота дозволяється, якщо вхід INT0 = 1 і біт TR0 = 1 (Рис. 6.2).

При: GATE = 0 робота лічильника залежить тільки від стану TR0.

TMOD.4 - M0. Те ж, але для Т/С1.

TMOD.5 - M1. Те ж, але для Т/С1.

TMOD.6 - C/T. Те ж, але для Т/С1.

TMOD.7 - GATE. Те ж, але для Т/С1.

За керування таймерами Т/С0, Т/С1 також відповідальний регістр TCON (Timer/Counter Control), розташований за адресою 88h, який може управлятися побитно (рис. 6.2).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| F1 | R1 | F0 | R0 | E1 | T1 | E0 | T0 |

Рис.6.2 - Структура регістра TCON

Молодша половина регістра використовується для управління входами запиту на переривання INT0 і INT1, старша - для управління безпосередньо таймерами Т/С0, Т/С1:

TCON.0 - IT0. Управління типом входу запиту на переривання INT0. При IT0 = 1 програмується динамічний по зрізу тип входу, в іншому випадку - статичний.

TCON.1 - IE0. Прапорець запиту переривання INT0 при динамічному вході. При підтвердженні переривання скидається.

TCON.2 - IT1. Те ж, що і IT0, але для входу INT1.

TCON.3 - IE1. Те ж, що і IE0, але для INT1.

TCON.4 - TR0. Прапорець програмного запуску/зупинки Т/С0.

TCON.5 - TF0. Прапорець переповнення Т/С0, який викликає запит переривання. При підтвердженні переривання скидається.

TCON.6 - TR1. Те ж, що і TR0, але для Т/С1.

TCON.7 - TF1. Те ж, що і TF0, але для Т/С1.

МК-51 має чотири режими роботи вбудованих таймерів, визначалися установкою відповідних бітів регістра TMOD. Режими роботи таймерів 0, 1, 2 збігаються для обох таймерів. Установка в режим 3 таймера-лічильника Т/С0 впливає на режим роботи Т/С1.

### 6.1.1 Режими роботи таймерів-лічильників.

**Режим 0:** таймери/лічильники є пристроями на базі 13-розрядних регістрів, утворених відповідними регістрами THX (x дорівнює 0 або 1) і 5-ю молодшими розрядами регістрів TLx (три старших розряду TL0 і TL1 є незначущими). Регістри TLx виконують в таймерах функцію дільника на 32. Таймери починають рахувати при установці біта TRx регістра TCON в стан«1». Установка в «1» біта GATEx регістра TMOD дозволяє керування таймерами/лічильниками ззовні (по входах INT0, INT1). Встановлення в «0» біта TRx регістра TCON забороняє рахунок незалежно від стану інших бітів. Біт C/Tx визначає роботу Т/С як таймерів ( «0»), так і лічильників ( «1»).

**Режим 1:** даний режим відрізняється від попереднього тільки тим, що в лічильнику використовується 16-розрядний регістр.

**Режим 2:** в цьому режимі використовуються 8-розрядні регістри TL0 (в Т/С0) і TL1 (в Т/С1). При переповненні цих регістрів в процесі рахунку відбувається перезавантаження їх значенням, заданим, регістрами TH0 або TH1. Регістри TH0 (TH1) завантажуються програмно,

і процес перезавантаження їх вмісту в TL0 (TL1) не змінює їх значення.

**Режим 3:** в цьому режимі працює тільки T/C0. Лічильник T/C1 заблокований і зберігає вміст своїх регістрів TL1 і TH1 (як при TR1 = 0). Лічильник T/C0 представлений двома незалежними 8-розрядними регістрами TH0 і TL0. Пристрій на базі TH0 може працювати тільки як таймер і використовує деякі керуючі біти і прапори T/C1, наприклад, при переповненні TH0 відбувається установка TF1, а для включення використовується біт TR1. Решта керуючі біти лічильника T/C1 з роботою таймера TH0 не пов'язані. Установка T/C0 в режим роботи 3 призводить до позбавлення T/C1 керуючого біта TR1. З цієї причини під час налаштування пристрою T/C0 в 3 режим, пристрій T/C1, що працює в режимах 0, 1, 2 і при GATE1 = 0, завжди включено. При переповненні в режимах 0 і 1 таймер T/C1 обнуляється, а в 2-му режимі - перезавантажується, не встановлюючи прапор TR1.

Лічильники/таймери T/C0, T/C1 є внутрішніми джерелом переривань і використовують для вироблення запитів переривань прапори переповнення TF0, TF1, представлені в регістрі управління TCON. Обробка переривань (вектору переривання) для T/C0 і T/C1 починається з адрес 000Bh і 001Bh відповідно.

### 6.1.2 Короткі теоретичні відомості

Зазвичай в керуючих програмах для ВКС виникає необхідність очікування ланцюжка подій, що представляється послідовністю імпульсних сигналів від датчиків і формування відповіді на них. Розглянемо дві типові процедури: підрахунок числа імпульсів між двома подіями, підрахунок числа імпульсів за заданий інтервал часу- задача типу «конвейер».

### 6.1.3 Підрахунок числа імпульсів між двома подіями

Цю типову процедуру зручно проілюструвати на конкретному прикладі. Припустимо, що необхідно підрахувати число деталей, які зійшли з конвеєра від моменту його включення до моменту вимикання. Факт того, що деталь зійшла з конвеєра фіксується фотоелементом, на виході якого формується імпульсний сигнал, який св свою чергу фіксується МК. Особливість процедури очікування імпульсного сигналу полягає в тому, що повинен виявити не тільки факт появи, але й факт закінчення сигналу (відсутність цієї події може вказувати на те, що деталь «застрягла» на конвеєрі, відповідно видати сигнал «зупинки конвеєра» з фіксацією часу події в регістрі).

По закінченню виконання процедури в акумуляторі фіксується число деталей, представлене у двійково-десятковому, 16-му, або ін. коді. Для простоти реалізації програми вважаємо, що загальна кількість деталей не перевищує 250, в акумуляторі фіксується число деталей, представлене у

двійковому коді (максимальна кількість деталей 255). На початку роботи конвеєра слід подати сигнал «включити конвеєр», а перед опитуванням лічильника «виключити конвеєр».

```
                                ; версія для МК-51
mov tmod,#01000000b        ; настроювання лічильника 1
mov th1,#0                  ; скидання лічильника деталей
waito: jb p3.4,waito       ; очікування включення конвеєра
setb tcon.6                 ; пуск лічильника 1
wait1: jnb p3.4,wait1      ; очікування вимикання конвеєра
clr tcon.6                  ; останов лічильника 1
mov a,th1                   ; (акумулятор) <-- число деталей
exit: ...                   ; вихід із процедури
```

### 6.1.4 Підрахунок числа імпульсів за заданий проміжок часу

При виконанні завдання, перетворення число-імпульсного коду у двійковий код, а також у ряді інших завдань може виникнути необхідність підрахунку числа імпульсів за заданий інтервал часу. Ця процедура може бути реалізовано чотирма різними способами:

- **програмною** реалізацією часового інтервалу і підрахунку числа імпульсів на вході МК;
- **програмною** реалізацією часового інтервалу і **апаратурним** підрахунком числа імпульсів (на внутрішньому таймері/лічильнику);
- **апаратурною** реалізацією часового інтервалу і **програмним** підрахунком числа імпульсів;
- **апаратурною** реалізацією часового інтервалу і **апаратурним** підрахунком числа імпульсів.

Для МК-51, що має у своєму складі два таймери/лічильника, можливий апаратний спосіб реалізації обох процесів: формування тимчасового інтервалу на Т/Л0 і підрахунок числа імпульсів на Т/Л1. Датчик імпульсів повинен бути підключений до входу Т1:

```
                                ; версія для МК-51
time equ not(10000)+1       ; визначення константи для
                                ; відліку інтервалу 10 мс
mov tmod,#01010001b         ; настроювання т/л, 16 біт
                                ; 1 - лічильник
                                ; 0 - таймер
clr a                       ; скидання акумулятора
mov th1,a                   ; скидання т/л1
mov th0,#high(time)         ; завантаження в т/л0
mov tl0,#low(time)          ; константи time
orl tcon,#50h               ; пуск т/л1 і т/л0
```

```
wait:jbc tcon.5,exit        ; перевірка переповнення т/л0
sjmp wait                   ; цикл, якщо tf=0
exit:mov b,th1              ; (b) (a)<--число імпульсів за 10 мс
mov a,tl1
exit:...                    ; вихід із процедури
```

### 6.1.5 Формування вихідних динамічних керуючих впливів

Сформувати на одному з виводів порту Р2 МК-51 періодичний імпульсний сигнал, період якого і шпаруватість задаються вмістом і-го регістру (див. варіант завдання).

Принцип роботи програми: зовнішня програма у випадкові моменти часу встановлює в регістри Ri мікроконтролера значення, що визначають тривалість імпульсу, період проходження сигналу і його шпаруватість. Підпрограма, що розроблюється повинна використовувати ці значення для формування сигналів на виходах портів мікроконтролера.

**Генерація періодичного керуючого впливу (меандру).** Для генерації меандру зручно скористатися процедурою видачі періодичного імпульсного керуючого впливу (**програмно**).

; **версія для МК- 51**

```
…
meandr: xcor:
cpl p1.3 acall
dlyx sjmp xcor
```

Нескінченний (або кінцевий за умовою припинення події MEANDR) періодичний сигнал формується в лінії 3 порту 1. На ін. лініях порту 1 сигнали залишаються незмінними. Як варіант, можливо використати генератор імпульсів з панелі MS «Інструменти» (апаратурна реалізація).

### 6.1.6 Формування вихідних статичних сигналів керування

**Формування статичних сигналів:** для керування виконавчим механізмом, що працюють за принципом «ввімкнути» і « вимкнути», на відповідній вхідній лінії порту МК необхідно сформувати **статичний сигнал** 0 («вимкнути») або 1 («ввімкнути»), що реалізується командами виводу безпосереднього операнду (містить у необхідному біті значення 0 або 1).

У випадку паралельного керування групою автономних виконавчих механізмів, підключених до вихідного порту, формується не двійковий керуючий вплив, а **вихідне управляюче слово** (ВУС), що має формат байта, кожному розряду якого ставиться у відповідність 1 або 0 залежно від того, які виконавчі механізми повинні бути включені, а які виключені. Керуючі слова зручно формувати командами логічних операцій над умістом порту. Команда ANL використовується для скидання тих біт ВУС, які в операнді (масці) задані нулем. Команда ORL використовується для установки біт ВУС.

Командою XRL здійснюється інверсія біт відповідно до вираження x + 1 = `x (x інвертоване).

**Приклад:** до виводів портів P1 і P2 підключені виконавчі механізми. Поточний стан виконавчих механізмів зафіксовано в поточному слові стану системи. Зовнішня програма може змінити в довільні моменти часу еталонне слово стану системи.

**Принцип роботи алгоритму може бути наступний:**

- періодично, із заданим періодом, програма виконує запит еталонного слова стану (те, яке потрібно встановити) системи і порівнює його з поточним словом стану системи;
- при виявленні розбіжності змісту еталонного і поточного слів стану, програма змінює стан відповідних ліній порту, починаючи з молодших;
- кожна зміна стану виконавчих механізмів фіксується в поточнім слові стану системи;
- період опитування – 10…100 ms.

Програма повинна бути оформлена у вигляді підпрограми, запит до якої здійснюється командою CALL. Для формування складних послідовностей ВУС зручно користуватися табличним методом, при якому усі можливі ВУС упаковані в таблицю, а прикладна програма МК обчислює адресу необхідного еталонного ВУС, вибирає його з таблиці і передає в порт виводу.

**Формування аперіодичного керуючого сигналу:** послідовність імпульсних сигналів з довільною тривалістю і шпаруватістю може бути отримана аналогічним чином, тобто шляхом чергування процедур видачі змінюваного значення сигналу (0 або 1) і виклику підпрограм тимчасових затримок заданої тривалості.

У МК-51 на вхід таймера/лічильника (Т/Л) можуть надходити сигнали синхронізації із частотою 1 Мгц (Т/Л у режимі таймера) або сигнали від зовнішнього джерела (Т/Л у режимі лічильника). Обидва ці режими можуть бути використані для формування затримок. Якщо використовувати Т/Л у режимі таймера повного формату (16 біт), то можна одержати затримки в діапазоні 1 - 65 536 мкс. Як приклад, розглянемо організацію тимчасової затримки тривалістю 50 мс у МК-51. Передбачено, що біт IE.7 встановлений.
-організація переходу до мітки next при переповненніТ/С0;

**версія для МК-51**

```
…
org 0bh              ; адреса вектора переривання від Т/С0
clr tcon.4                    ; зупинка Т/С0
reti                          ; вихід з підпрограми
                              ; обробки переривання
org 100h                      ; початкова адреса програми
```

```
mov tmod,#01h                    ; настроювання T/C0
mov tl0,#low(not(50000-1))       ; завантаження таймера
mov th0,#high(not(50000-1))
setb tcon.4                      ; старт T/C0
setb ie.1          ; дозвіл переривання від T/C0
setb pcon.0        ; перехід МК-51 в режим холостого ходу
next: . . . . .
```

Керуючий вплив типу "одиночний імпульс" можна одержати послідовною видачею сигналів «ввімкнути» і «вимкнути» із проміжним викликом підпрограми тимчасової затримки. Тривалість імпульсу визначається тимчасовою затримкою, реалізованою підпрограмою DELAY.

```
puls:                            ;видача імпульсу в лінію 3 порту 1

….
on: anl p1,#11110111b ; включення виконавчого механізму call
delay
;тимчасова затримка
off: orl p1,#00001000b       ; відключення виконавчого механізму
```

### 6.1.7 Завдання до ЛР№5

Завдання має два розділи (непарний/парний варіант № підгрупи):

- підрахунок числа вхідних імпульсів між двома подіями (непарний-1, 3) або за заданий проміжок часу (парний-2, 4);

- формування вихідних динамічних або статичних сигналів керування – збільшені вдвічі тривалість і період імпульсів попереднього завдання.

**Варіанти завдань по групах розділів 1-4**

1) тривалість імпульсу 100 (200) мкс, період проходження сигналу-500(1000)мкс, шпаруватість 1:5.

2) проміжок часу- тривалість імпульсу 200(400) мкс, період проходження 400(800) мкс, шпаруватість 1:2, використати довільну затримку

3) тривалість імпульсу 250(500) мкс, період проходження сигналу – 1000(2000)мкс, шпаруватість 1:4.

4) проміжок часу- тривалість імпульсу 150 (300)мкс, період проходження 1200(2400) мкс, шпаруватість 1:8, використати довільну затримку

### 6.1.8 Порядок виконання ЛР№5

На занятті студент повинен побудувати на «робочому столі» в MS - електричну схему пристрою на основі МК-51, випробувати і перевірити її функціонування і тестові програми, які наведені та їх варіанти. При підготовці до роботи студент повинен написати і відкоригувати власні варіанти програми і, відповідно до завдання, виконати необхідні розрахунки, корегування і провести моделювання, перевірити вірність функціонування.

Сформовані діаграми імпульсів відобразити на ВП типу осцилограф/ логічний аналізатор (панель «Інструменти») в ППП LV у вигляді діаграм (використати один із способів 1 ... 4, см. п.2.1)

### 6.1.9 Зміст звіту.

1. Найменування і мета ЛР.
2. Опис МК-51 і його лічільників/таймерів, їх призначення, режими роботи.
3. Копія програмного файлу (на асемблері і/або на С) з коментарями.
4. Копія схемного файлу в MS і LV з ВП, моделювання із зазначенням режимів, позиційних позначень елементів і шин.
5. Отримані результати (діаграми, графіки, значення, ін.), пояснення до них.
6. Висновки по ЛР.

### 6.1.10 Контрольні питання

1. Які особливості архітектури має МК-51?
2. Які є підходи до формалізації функцій конвеєра при його створенні?
3. Від чого залежить нормальна робота конвеєра? Як можливо вдосконалити алгоритм роботи будь-якого конвеєра з іншими типами МК і використанням сучасних цифрових технологій?
4. Як використано в ЛР таймери-лічільники? Вкажіть їх режими роботи.
5. Які суттєві недоліки і/або переваги є у МК-51?

## 7. Лабораторна робота № 6. Проведення основних арифметичних операцій

**Мета роботи:** освоїти основні методи аналізу вхідних даних і виконання типових алгоритмів арифметичних операцій з ними у форматі з фіксованої комою (ФФК) на основі МК-51 у ППП MS. Учбове завдання: розробка програм для отримання алгоритмів чотирьох арифметичних операцій для 8 /16-и розрядних операндів та контроль результатів.

### 7.1 Особливості МК-51 при виконанні арифметичних операцій

(Деякі алгоритми їх виконання наведено в додатку до цієї ЛР у додатку А.1)

**Порти** (0,1,2,3) МК-51 служать для побайтного введення - виводу даних. Крім того, порти 0 і 2 служать для виводу адреси. Виводи порту 3 можуть бути використані для реалізації альтернативних функцій. Порт 0 є двонаправленим, порти 1, 2, 3 – квазідвонаправ-леними: кожна лінія порту може бути налаштована для введення або виводу даних. Навантажувальна здатність портів 1, 2, 3: один ТТЛ- вхід, а порту 0- два ТТЛ входу.

**Таймер-лічільник**: два 16-розрядних програмно доступних.

**Акумулятор** (А) служить для зберігання як операнду, так і результату операції. Але МК-51 може виконувати ряд команд і без участі акумулятора. Виконання багатьох команд проводиться в арифметико-логічному пристрої, а ряд ознак операцій фіксується в регістрі слова стану програми (PSW).

**Арифметико-логічний пристрій** (АЛП, ALU 8- бітний) може виконувати арифметичні операції додавання, віднімання, множення й ділення; логічні операції nI, nАБО, виключне nАБО, а також операції циклічного зрушення, скидання, інвертування і т.д. В АЛП є програмно недоступні регістри Т1 і Т2, призначені для тимчасового зберігання операндів, схеми десяткової корекції і формування ознак.

**Восьмирозрядний покажчик стека** (SP) може адресувати будь-яку область пам'яті даних. Вміст його зменшується на одиницю в ході виконання команд PUSH і CALL і збільшується на одиницю при виконанні команд POP і RET. При скиданні в SP автоматичнозавантажується початковий код 07 Н. 16-розрядний регістр-покажчик даних (DPTR) служить для фіксації адреси при звертанні до зовнішньої пам'яті. Він може працювати як два незалежні 8-розрядних регістри DPH і DPL.

**Команди пересилання даних (Додаток 1).** Як було розглянуто раніше, арифметичні s логічні команди МК можуть бути виконані тільки над вмістом регістру акумулятора, тому винятково важливе значення в системі команд здобувають команди пересилання даних. За допомогою цих команд можна скопіювати вміст будь-якої комірки пам'яті в регістр-акумулятор або навпаки скопіювати вміст акумулятора в будь-яку комірку пам'яті. Тому що в

мікроконтролері присутні три незалежні області пам'яті, то для звертання до них уведені різні команди:
- копіювання даних у внутрішньому ОЗП: MOV;
- обмін даними акумулятора із внутрішнім ОЗП: XCH, XCHD;
- копіювання із зовнішньої пам'яті даних: MOVX;
- копіювання даних з пам'яті програм: MOVC.

Будь-яке гніздо 256-байтового блоку внутрішнього ОЗП даних може бути обрана з використанням непрямо-регістрової адресації через регістри покажчики R0 і R1 (обраного банку робочих регістрів). Команди пересилання із прямою адресацією між комірками пам'яті дозволяють затягати вміст порту в гніздо внутрішнього ОЗП або обмінюватися вмістом гнізд внутрішнього ОЗП між собою без використання акумулятора. Таблиці символів (кодів), записані в ПЗП програми можуть бути скопійовані в акумулятор за допомогою команд передачі даних з непрямою адресацією. Гніздо адресного простору 64 Кбайт зовнішнього ОЗП також може бути обрана з використанням непрямо-регістрової адресації через регістр покажчик даних DPTR. Вміст акумулятора може бути виміняне із вмістом робочих регістрів обраного банку.

**Команди розгалуження і передачі керування.** Команди розгалуження дозволяють реалізовувати умовні оператори й оператори циклів. У мікроконтролерах сімейства MCS-51 доступні наступні команди:
- безумовний перехід: LJMP, AJMP, SJMP;
- виклик і повернення з підпрограми: LCALL, ACALL, RET, RETI;
- перевірка вмісту акумулятора: JZ, JNZ, CJNE, JMP;
- перевірка прапора переносу З: JC, JNC; перевірка вмісту будь-якого біта в бітовому просторі: JB, JNB, JBC.

**Команди** 16-розрядних безумовних переходів і викликів підпрограм дозволяють здійснити перехід у будь-яку точку адресного простору пам'яті програм обсягом до 64 Кбайт. Приклади команд:

**Команди** 11-розрядних переходів і викликів підпрограм дозволяють скоротити обсяг програми, але при цьому забезпечують переходи тільки усередині програмного модуля.

**Команди умовних і безумовних переходів** щодо початкової адреси наступної команди в межах від (PC)-127 до (PC)+127.

**Команди перевірки** вмісту акумулятора і прапора переносу С можуть бути використані для реалізації перевірки різних умов. При цьому вміст не змінюється, тобто якщо потрібно зробити кілька перевірок однієї й тієї ж змінної, то повторно переносити значення цієї змінної в акумулятор не потрібно.

**Команди типу непрямий перехід** JMP @A+DPTR у системі команд МК-51 забезпечує розгалуження програми по вмісту акумулятора А. Це

дозволяє реалізовувати операцію переходу за заданим кодом, еквівалентну операторові case у мові програмування pascal, але набагато швидше (за два машинні цикли). Використання в цій команді покажчика даних DPTR дозволяє розміщати таблицю переходів у будь-якому місці пам'яті програм. Як видно з наведених прикладів, команди переходів цього МК дозволяють реалізувати більш ефективні по кількості команд програми в порівнянні з іншими МК (наприклад МС-48).

**Арифметичні команди.** У наборі команд є наступні:
- додавання ADD;
- додавання з урахуванням прапора переносу ADDC;
- віднімання з запозиченням SUBB;
- інкрементування (збільшення на 1) INC;
- декрементування (зменшення на 1) DEC;
- десяткова корекція DA;
- множення MUL;
- ділення DIV.

Дії проводяться над цілими числами без знака. При операції множення вміст акумулятора А множиться на вміст регістру В, і результат розташовується таким способом: молодший байт у регістрі В, старший у регістрі А. При виконанні операції ділення, ціле від ділення поміщується в акумулятор А, залишок у регістр В.

## 7.2 Завдання до лабораторної роботи

### 7.2.1 Складання двох 8-розрядних чисел

Скласти (X+Y) і помістити результат згідно з варіантом табл.7.1-7.2.

Таблиця 7.1 - Додати два 8-розрядних числа X і Y (на ДСДК)

| Варіант | X | Y |
|---|---|---|
| I | 1EH | 24H |
| II | 1FH | 35H |
| III | 5DH | 20H |
| IV | 25H | 7BH |
| V | 8AH | CBH |

Таблиця 7.2 - Результат розмістити:

| Варіант | Внутрішнє ОЗП | Внутрішнє ОЗП | Порт |
|---|---|---|---|
| I | 06H | 20H | P2 |
| II | 4BH | 68H | P0 |
| III | 72H | 2AH | P3 |
| IV | 3EH | B5H | P1 |
| V | 54H | D0H | P2 |

### 7.2.2 Множення 8-ми розрядних чисел

Помножити два 8-ми розрядних числа (X x Y) і помістити результат згідно з варіантом табл.7.3-7.4.

Таблиця 7.3 - Помножити два 8-розрядних числа X и Y

| Варіант | X | Y |
|---|---|---|
| I | A0H | 09H |
| II | 2BH | 7AH |
| III | 1FH | 6DH |
| IV | 83H | E4H |
| V | 37H | FAH |

Таблиця 7.4 - Результат помістити

| Варіант | Внутрішнє ОЗП | Порти |
|---|---|---|
| I | 18H | P2,P3 |
| II | 0FH | P0, P1 |
| III | 3AH | P1, P3 |
| IV | 6DH | P2, P3 |
| V | 7AH | P0, P1 |

### 7.2.3 Віднімання 16-х чисел

Провести віднімання для двох чисел (X – Y) та помістити результат згідно з варіантом табл.7.5-7.6.

Таблиця 7.5 - Зробити віднімання (на ДСДК) 16-их чисел X, Y

| Варіант | X | Y |
|---|---|---|
| I | 201FH | 0145H |
| II | 7A04H | 150AH |
| III | 9F1DH | 70DEH |
| IV | 660BH | 3A72H |
| V | 7845H | D976H |

Таблиця 7.6 - Результат помістити

| Варіант | Внутрішнє ОЗП | Порти |
|---|---|---|
| I | 6DH | P0, P3 |
| II | 27H | P1, P0 |
| III | 4FH | P2, P1 |
| IV | 2EH | P3, P0 |
| V | 2AH | P1, P2 |

*В кінці ЛР№7 наведені деякі приклади схем алгоритмів (рис.А.1-А.3): множення цілих чисел зі знаком на двійковому суматорі прямого коду (ДСПК) (7.1); складання цілих чисел зі знаком на ДСДК (7.2); перевод прямого коду в доповняльний код (7.3). Інші алгоритми та їх особливості обговорити з викладачем і /або найти в Internet.

### 7.2.4 Ділення 16-х чисел

Розділити два 16-розрядних числа (X / Y) та помістити результат згідно з варіантом табл. 7.775.8.

Таблиця 7.7 - Розділити два 16-розрядних числа A і B

| Варіант | X | Y |
|---|---|---|
| I | 3A06H | 2005H |
| II | 101FH | 01A5H |
| III | E02AH | A10DH |
| IV | A101H | 040AH |
| V | F486H | DABAH |

Таблиця 7.8 - Результат помістити

| Варіант | Внутрішнє ОЗП | Порт |
|---|---|---|
| I | 20H | P0, P1 |
| II | 4BH | P3, P0 |
| III | 5FH | P0, P2 |
| IV | 28H | P1, P3 |
| V | 56H | P2, P3 |

## 7.3 Зміст звіту

Звіт з ЛР повинен мати: теоретичну частину, варіант завдання, розрахунки, програму виконання, покроковий алгоритм виконання арифметичних операцій з поясненнями, діаграми (фото) результатів, висновки. На занятті студент повинен отримати у викладача варіант, одержати допуск до виконання ЛР на ПК, ознайомитись з підходами до виконання завдання, налагодити програму на ПК. При необхідності, пояснити викладачеві роботу програми (в т.ч. покроково), її частин і обґрунтувати правильність її функціонування відповідно до варіанту завдання. Після цього робота вважається виконаною, викладач контролює її електронний вигляд і підписує чернетку ЛР. До наступного заняття студент повинен оформити звіт по виконаній ЛР (на основі чорнетки) і захистити його у викладача. Позитивна оцінка і підпис звіту викладачем означає, що ЛР здана. Рекомендується підписані звіти з ЛР зберігати до кінця навчання за курсом. На іспиті студентам дозволяється користуватися своїми звітами при відповідях на запитання і розв'язання завдань.

## 7.4 Контрольні питання

1. Вказати типи команд, які треба використати у ЛР для виконання арифметичних операцій (згідно варіанту для цілих чисел: додавання, множення, віднімання, ділення, їх комбінація) на МК 80C51.

2. В чому полягає різниця між алгоритмами виконання чотирьох арифметичних операцій для 8-ми і 16-ти розрядних цілих операндів?
3. Які частини алгоритмів арифметичних операцій співпадають і їх можливо поєднати для скорочення процесів розрахунків?
4. Які складові операндів можуть впливати на час проведення арифметичних операцій?
5. Як обчислити середній час виконання арифметичної операції? Запропонуйте блок схему такого обчислення.
6. Які ресурси МК додатково потрібно задіяти для виконання чотирьох арифметичних операцій у форматі з плаваючою комою (ФПК) у порівнянні з форматом з фіксованою комою (ФФК).

## 7.5 Додаткові данні про виконання арифметичних операцій +, -, x, /.

### Арифметичні операції

Цю групу утворюють 24 команди (табл. П1.2), які виконують операції додавання, десяткової корекції, інкремента/декремента байтів. Є команди віднімання, множення та ділення байтів. Команди ADD та ADDC допускають складення(додавання) акумулятора з великою кількістю операндів. Аналогічно командам ADDC існуютьчотири команди SUBB, що дозволяє достатньо просто виконувати віднімання байтів та багатобайтних двійкових чисел. В мікроконтролері реалізується розширений список (перелік) команд інкремента/декремента байтів, команда інкремента 16-бітного регістра-вказівника (покажчика) даних.

**Деякі приклади використання команд для виконання арифметичних операцій.**

**Приклад 1.** Додавання (віднімання*). Додати два двійкові двобайтні числа. Доданки розміщуються в резидентній пам'яті даних, починаючи з молодшого байту. Початкові адреса доданків - в R0, R1; формат доданків у байтах- в R2:

```
CLR C
      LOOP:
ADDC A, @R1
MOV @R0, A
INC R0 INC R1
DJNZ R2, LOOP      ; цикл, якщо не всі байти підсумовані
```

При додаванні чисел без знаку, на переповнення вкаже прапор С, а у випадку додавання чисел зі знаком - прапор OV.

***Приклади* алгоритмів додавання (віднімання) на двійковому суматорі додаткового коду (ДСДК): операнди А і В є цілі числа (а) або дрообові (б) в форматі з фіксованою комою (ФФК). Отримання результату С можливо з урахуванням особливостей провелення операції додавання (віднімання) з урахуванням знаку другого операнду В***

Доповніть програму додавання командами, які забезпечують її тестування, створіть контрольний приклад і виконайте налагодження в ProView. Визначте час обчислення в залежності від формату вхідних чисел.

**Приклад 2.** Множення. Команда MUL обчислює добуток двох цілих без знакових чисел, які зберігаються в регістрах A і B. Молодша частина добутку розміщається в A, а старша в регістрі-розширювачі B. Якщо вміст B виявляється рівним нулю, то прапор OV скидається, інакше – встановлюється. Прапор переносу завжди скидається. Наприклад, якщо акумулятор містив число 200 (0C8H), а розширювач 160 (0A0H), то в результаті виконання команди MUL AB отримаємо добуток 32000 (7D00H). Акумулятор буде містити нуль, а розширювач
– 7DH, прапор OV буде встановлений, а прапор C – скинутий.
Нехай потрібно помножити ціле двійкове число довільного формату на константу 73. Вихідне число розміщається в резидентній пам'яті даних, адреса молодшого байту знаходиться в регістрі R0. Формат числа в байтах зберігається в R1:

```
MOV A, #0          ; скидання акумулятора
LOOP: ADD A, @R0   ; завантаження множеного
MOV B, #73         ; завантаження множника
MUL AB             ; множення
MOV @R0, A         ; запис молодшого байту часткового
                           добутку
INC R0             ; приріст адреси
MOV A, B           ; пересилання старшого байту
                   часткового добутку в акумулятор
XCH A, @R0         ; попереднє формування чергового
                           байту добутку
                   ; цикл, якщо не всі байти вихідного
DJNZ R1, LOOP        числа помножені на константу
```

Отриманий добуток розміщається на місці вихідного числа і займає в пам'яті на один байт більше.
Розберіться в алгоритмі множення. Доповніть програму командами, які забезпечують її тестування, складіть контрольний приклад і виконайте налагодження.

***Приклади* схем алгоритмів множення на двійковому суматорі прямого коду (ДСПК): множене і множник - цілі числа - а; множене-ціле число, множник - дробове число – б; або навпаки. Знак одержуваного добутку $Sg_C$ визначається до його проведення по формулі: $Sg_C=Sg_A \square Sg_B$, де $Sg_A \square\square Sg_B$ - значення знакової частини операндів А і В, а ☐☐☐☐знак додавання функції «Σmod2».***

**Приклад 3.** Ділення цілих чисел. Команда DIV виконує ділення вмісту акумулятора на вміст регістра – розширювача. Після ділення

акумулятор містить цілу частину частки, а розширювач – залишок. Прапори C і OV скидаються. При діленні на нуль встановлюється прапор переповнення, а частка залишається невизначеною. Команда ділення може бути використана для швидкого перетворення двійкових чисел в десяткові (через бінарно кодовані десяткові – BCD- числа) .

У якості приклада розглянемо програмку, яка переводить двійкове число, що міститься в акумуляторі, в BCD – код. При такому перетворенні може вийти трирозрядне BCD – число. Старша цифра (число сотень) буде розміщена в регістрі R0, а дві молодші в акумуляторі:

```
MOV B, #100   ; завантаження 100 для обчислення кількості сотень ;
              акумулятор містить число сотень (старшу цифру)
DIV AB
MOV R0, A     ; пересилання в R0 старшої цифри
XCH A, B      ; пересилання залишку вихідного числа в акумулятор
MOV B, #10    ; завантаження 10 для обчислення кількості десятків
DIV AB        ; A містить число десятків, B – число одиниць
SWAP A ADD    ; розміщення числа десятків в старшому тетраді A
A, B          ; підсумовування залишку (числа одиниць), тепер
              акумулятор містить дві молодші цифри
```

**Приклад 4.** Ділення цілих чисел. Алгоритм, який виконується на 8-ми розрядних МК, вважається швидким для двобайтних чисел: Припустимо, що маємо два числа: ділене A = 0010111001101010 і дільник B = 0000000000101000. Означемо ідею алгоритму, додаткові дані і формальні «блоки» мікрооперацій:

(1) заведемо змінну index рівну 0000000000000001 і result = 0

(2) зрушуємо дільник і index вліво поки старший біт дільника НЕ порівняється зі старшим бітом діленого, тобто отримуємо дільник = 0010100000000000, index = 0000000100000000

(3) віднімаємо з діленого отриманий дільник:

```
  0010111001101010
_ 0010100000000000
  0000011001101010
```

(4) якщо повчань число <0, то зрушуємо дільник і index вправо на 1 розряд і переходимо до 3-му пункту. А якщо> 0 (як у нас), відразу йдемо далі.

(5) Число отримане при відніманні використовуємо в якості нового діленого (залишок).

(6) result = result + index (0000000100000000)

(7) зрушуємо дільник і index на 1 розряд вправо

(8) goto (3) поки не отримаємо негативний результат віднімання, тоді у змінній result буде частка від цілочисельного ділення


Рисунок 7.1 - Множення двох цілих чисел на ДСПК (знак результату C = A x B) вираховується згідно формули $Sg_C = Sg_A \square Sg_B$;

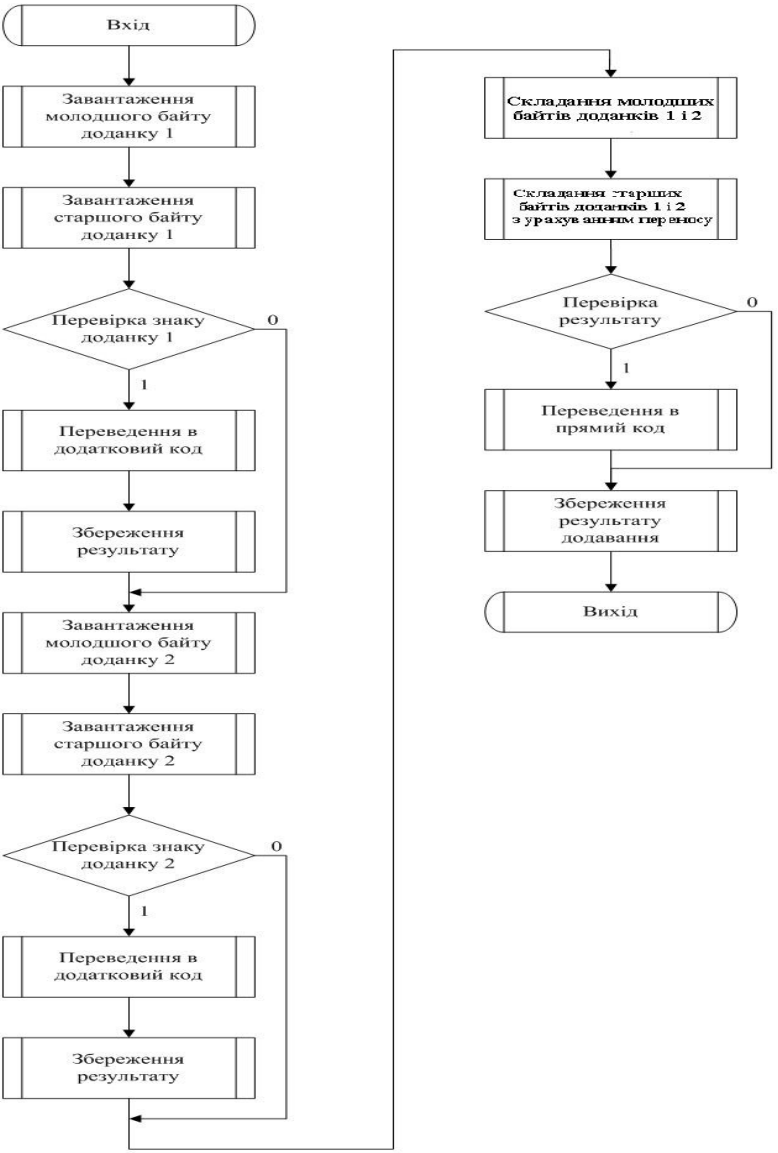



Рисунок А.2 - Складання цілих чисел зі знаком на ДСДК


Рисунок 7.3 - Переклад чисел: з прямого коду в доповняльний.

**Зауваження**. Розберіться в алгоритмах переводу. Доповніть програму командами, які забезпечать її тестування, складідь контрольний приклад і виконайте налагодження, визнайте приблизний час обчислень.

## Додаток А. Особливості МК51

### А.1 Відомості : однокристальні МК сімейства МК51, їх характеристики

Сімейство однокристальних МК51 включає ряд моделей, модифікації яких варіюються залежно від обсягу і характеру обчислювальних ресурсів (пам'яті програм і даних, тактової частоти).

Сімейство вітчизняних МК51 включає 5 основних модифікацій серій КР1816, КР1830, що відрізняються по реалізації резидентної пам'яті програм і потужності споживанні (Мікросхема КР1835ВЕ51* є одним з модифікованих варіантів КР1830ВЕ51).

У таблиці А.1 наведено основні моделі МК (далі МК51).

Таблиця А.1 - Сімейство МК8051 (MSC8051) і їх аналоги

| Модель | Аналог | Обсяг резидент-ної пам'яті програм, Кбайт | Обсяг резидент-ної пам'яті даних, байт | Тактова частота, МГц | Струм спожи-вання, мА |
|---|---|---|---|---|---|
| КР1816ВЕ31 | 8031АН (8031) | зовнішня | 128 | 12 | 150 |
| КР18І6ВЕ51 | 8051АН (8051) | 4 К | 128 | 12 | 150 |
| КМ1816ВЕ751 | 875ІАН (8751) | 4 К | 128 | 12 | 220 |
| КР1830ВЕ31 | 80С3ІВН (80С31) | зовнішня | 128 | 12 | 18 |
| КР1830ВЕ51 | 80С51ВН (80С51) | 4 К | 128 | 12 | 18 |
| КР1835ВЕ51* | 80С51ВU (80С51) | 4 К | 128 | 12 | 18 |

З таблиці А.1 видно, що найбільш економічними є великі інтегральні схеми (ВІС) серії КР1830 (1835) при однакових значеннях основних технічних характеристик.

Їхнім недоліком є менший температурний діапазон експлуатації (-10 - +70 С), у той час як закордонний фірми Intel, Siemens роблять модифікації контролерів, розраховані на застосування в діапазонах: 0 - +70 С, -40 - +85 С, - 40 - +110 С і -40 - +125С [5].

КР1816ВЕ51, у відмінності від КР1816ВЕ31, містить внутрішнє ППЗП ємністю 4 Кбайт із ультрафіолетовим стиранням.

Слід зазначити, що у всіх моделей МК51 за рахунок використання зовнішньої пам'яті, ємність пам'яті програм і даних може бути розширена до 64 Кбайт.

Приведемо деякі особливості моделей КМ1830ВЕ751 і КМ1830ВЕ753 (аналог 8753Н фірми AMD). Остання відрізняється наявністю перепрограмовувального запам'ятовувального пристрою (ППЗП) з ультрафіолетовим стиранням на 8 Кбайт.

Мікросхеми мають особливості:

- спеціальний режим експлуатації;
- додаткові засоби захисту пам'яті програм, розташованої на кристалі, два біти захисту пам'яті і шифрувальну таблицю;
- алгоритм програмування вкороченими імпульсами.

Восьмирозрядні однокристальні мікроконтролери сімейства МК51 виконані по n-МОП технології (серія 1816) і КМОП технології (серія 1830). Кожний МК розглянутого сімейства містить вбудоване ОЗП пам'яті даних ємністю 128 байт із можливістю розширення загального обсягу оперативної пам'яті даних до 64 Кбайт за рахунок використання зовнішніх мікросхем ЗППВ.

Загальний обсяг пам'яті МК сімейства МК51 може досягати 128 Кбайт: 64 Кбайт - пам'ять програм і 64 Кбайт - пам'ять даних.

При розробці на базі МК більш складних систем можуть бути використані стандартні ІС із байтовой організацією, наприклад, серії КР580.

МК містять усі вузли, необхідні для автономної роботи:

- центральний восьмирозрядний процесор;
- пам'ять програм обсягом 4 Кбайт (тільки КМ1816ВЕ751, КР1816ВЕ51 і КР1830ВЕ51);
- пам'ять даних обсягом 128 байт;
- чотири восьмирозрядних програмувальних каналів введення-виводу (порти Р0, Р1, Р2, Р3);
- два 16-бітових багаторежимних таймера/лічильника;
- система переривань із п'ятьома векторами і двома рівнями;

- послідовний інтерфейс;
- тактовий генератор.

Система переривань, блок послідовного інтерфейсу і таймери об'єднані в один блок.

Використання мікроконтролера сімейства МК51 (у порівнянні із МК48) забезпечує збільшення обсягу пам'яті команд і пам'яті даних. Нові можливості введення-виводу і периферійних пристроїв розширюють діапазон застосування і знижують загальні витрати системи. Залежно від умов використання швидкодія системи збільшується мінімум у два з половиною раза і максимум вдесятеро.

МК КМ1816ВЕ751 містить ППЗП ємністю 4096 байт зі стиранням ультрафіолетовим випромінюванням і зручний на етапі розробки системи при налагодженні програм, а також при виробництві невеликими партіями або при створенні систем, що вимагають у процесі експлуатації періодичного підстроювання. За рахунок використання зовнішніх мікросхем пам'яті загальний обсяг пам'яті програм може бути розширений до 64 Кбайт. МК КР1816ВЕ31 і КР1830ВЕ31 не містять вбудованої пам'яті програм і можуть використовувати до 64 Кбайт зовнішньої постійної (або перепрограмовувальної пам'яті програм) та ефективно використовуватися в системах, що вимагають суттєво більшого за обсягом ПЗП пам'яті програм (на кристалі тільки 4 Кбайт).

МК51 має:

**- Пам'ять програм у вигляді постійного запам`ятовуючого пристрою (ПЗП):** призначена для зберігання програм і має окреме від пам'яті даних адресний простір обсягом до 64 Кбайт, причому для мікросхем КР1816ВЕ51, КМ1816ВЕ751 і для КР1830ВЕ51 частина пам'яті програм з адресами 0000Н -0FFFH розташована на кристалі МК. Пам'ять програм, розташована на кристалі, складається з 12-розрядного дешифратора і ПЗП ємністю 4 К*8 біт для мікросхем КР1816ВЕ51, КР1830ВЕ51 або перепрограмуємого ПЗП (ППЗП) з ультрафіолетовим стиранням ємністю 4 К*8 біт для КМ1816ВЕ751. Запис програм у ПЗП відбувається під час виготовлення кристалів.

**-Регістри особливого призначення (РОП) -** 32 РОП;

**-Програмно-керовані «прапори» (ПКП) -** 128 обумовлених користувачем;

**-Набір регістрів спеціальних функцій (РСФ).**

Таблиця А.2 – Набір регістрів спеціальних функцій (РСФ)

| Символ | Найменування | Адреса |
|---|---|---|
| * ACC | Акумулятор | 0E0H |
| * B | Регістр-Розширник акумулятора | 0F0H |
| * PSW | Слово стану програми | 0D0H |
| SP | Регістр-Покажчик стека | 81H |
| DPTR | Регістр-Покажчик даних (DPH)/(DPL) | 83H/82H |
| * P0 | Порт 0 | 80H |
| * P1 | Порт 1 | 90H |
| * P2 | Порт 2 | 0A0H |
| * P3 | Порт 3 | 0B0H |
| * IP | Регістр пріоритетів | 0B8H |
| * IE | Регістр маски переривань | 0A8H |
| TMOD | Регістр режиму таймера/лічильника | 89H |
| * TCON | Регістр керування/статус таймера | 88H |
| TH0 | Таймер 0 (старший байт) | 8CH |
| TL0 | Таймер 0 (молодший байт) | 8AH |
| TH1 | Таймер 1 (старший байт) | 8DH |
| TL1 | Таймер 1 (молодший байт) | 8BH |
| * SCON | Регістр керування приймача | 98H |
| SBUF | Буфер приймача | 99H |
| PCON | Регістр керування потужністю | 87H |
| Примітка. Регістри, які відзначені *, допускають адресацію окремих біт. | | |

(РОП і обумовлені користувачем ПКП розташовані в адресному просторі **внутрішнього** ОЗП даних (ВОЗП). Регістри спеціальних функцій РСФ (SFR, SPECIAL FUNCTION REGISTERS) із вказівкою їх адрес наведено в табл. 1.2).

**-Акумулятор.** ACC- регістр акумулятора. Команди, призначені для роботи з акумулятором, використовують мнемоніку "A", наприклад, MOV A, P2. Мнемоніка "ACC" використовується, приміром, при побітовій адресації акумулятора. Так, символічне ім'я п'ятого біта акумулятора при використанні асемблера ASM51 буде наступним: ACC.5.

**-Регістр B.** Використовується під час операцій множення і ділення. Для інших інструкцій регістру може розглядатися як додатковий надшвидкий регістр.

**-Регістр стану програми.** Регістр PSW містить інформацію про стан програми.

**-Покажчик стека SP**. 8-бітовий регістр, уміст якого інкременується перед записом даних у стек при виконанні команд PUSH і CALL. При початковому скиданні покажчик стека встановлюється в 07H, а область стека в ОЗП даних починається з адреси 08H. При необхідності шляхом перевизначення покажчика стека область стека може бути розташована в будь-якім місці внутрішнього ОЗП даних МК.

**-Покажчик даних.** Покажчик даних (DPTR) складається зі старшого байта (DPH) і молодшого байта (DPL). Містить 16-бітову адресу при звертанні до зовнішньої пам'яті. Може використовуватися як 16-бітовий регістр або як два незалежні восьми бітові регістри.

**-Порт0 -Порт3**. Регістрами спеціальних функцій P0, P1, P2, P3 є регістри-"засувки" відповідно портів P0, P1, P2, P3.

**-Буфер** послідовного порту. SBUF являє собою два окремі регістри: буфер передавача і буфер приймача. Коли дані записуються в SBUF, вони надходять у буфер передавача, причому запис байта в SBUF автоматично ініціює його передачу через послідовний порт. Коли дані читаються з SBUF, вони вибираються з буфера приймача.

**-Регістри таймера.** Регістрові пари (TH0.TL0) і (TH1.TL1) утворюють 16-бітові рахункові регістри відповідно таймера/лічильника 0 і таймера/лічильника 1.

**-Регістри керування.** Регістри спеціальних функцій IP, IE, TMOD, TCON, SCON і PCON містять біти керування і біти стану системи переривань, таймерів/лічильників і послідовного порту. МК при функціонуванні забезпечує:

1) мінімальний час виконання команд додавання - 1 мкс;

2) апаратне множення/ділення з мінімальним часом виконання команд множення/ділення - 4 мкс.

В МК передбачена можливість завдання частоти внутрішнього генератора за допомогою кварцу, LC - ланцюжка або зовнішнього генератора.

Архітектура сімейства МК51 незважаючи на те, що вона заснована на архітектурі сімейства МК48, все-таки не є повністю сумісною з нею. У новім сімействі є ряд нових режимів адресації, додаткові інструкції, розширений адресний простір і ряд інших апаратних відмінностей. Розширена система команд забезпечує побайтовую і побітову адресацію, двійкову і двоїчно-десяткову арифметику, індикацію переповнення і визначення

парності/непарності, можливість реалізації логічного процесора. Найважливішою і відмітною рисою архітектури сімейства МК51 є те, що АЛП може, поряд з виконанням операцій над 8-розрядними типами даних, маніпулювати однорозрядними даними. Окремі програмно-доступні біти можуть бути встановлені, скинуті або замінені їхнім доповненням, можуть пересилатися, перевірятися і використовуватися в логічних обчисленнях. Тоді як підтримка простих типів даних (при існуючій тенденції до збільшення довжини слова) може з першого погляду здатися кроком назад, ця якість робить МК сімейства МК51 особливо зручними для застосувань, у яких використовуються контролери. Алгоритми роботи останніх по своїй суті припускають наявність вхідних і вихідних булевих змінних, які складно реалізувати за допомогою стандартних мікропроцесорів. Усі ці властивості в цілому називаються булевим процесором сімейства МК51. Завдяки такому потужному АЛП набір інструкцій МК сімейства МК51 однаково добре підходить як для застосувань керування в реальному масштабі часу, так і для алгоритмів з більшим обсягом даних.

### А.2 Про деякі особливості функціонування МК51

Якщо на вивід МК EA/NPP подана напруга живлення Vcc то звертання до зовнішньої пам'яті програм відбувається автоматично при виробітку лічильником команд адреси, що перевищує 0FFFH.

Якщо адреса перебуває в межах 0000H - 0FFFH, обіг відбувається до пам'яті програм, розташованої на кристалі (внутрішньої пам'яті програм).

Якщо на вивід EA/NPP поданий «0», внутрішня пам'ять програм відключається і, починаючи з адреси 0000H, усі звернення виконуються до зовнішньої пам'яті програм.

Якщо МК не має внутрішньої пам'яті програм, її вивід EA/NPP повинен бути підключений до загальної шини.

Читання із зовнішньої пам'яті програм стробується сигналом PSEN. При роботі із внутрішньою пам'яттю програм сигнал PSEN не формується. МК не мають інструкцій і апаратних засобів для програмного запису на згадку програм.

При звертаннях до зовнішньої пам'яті програм завжди формується 16-розрядна адреса, молодший байт якого видається через порт P0, а старший - через порт P2. При цьому байт адреси,

видаваний через порт P0, повинен бути зафіксований у зовнішньому регістрі по спаду сигналу АЛП ALE, тому що надалі лінії порту P0 використовуються в якості шини даних, по якій байт із зовнішньої пам'яті програм уводиться в МК.

### А.3 Функціональна схема включення МК51 із зовнішнім ППЗП програм

**Порт P0** працює як мультиплексорна шина адреса/даних: видає молодший байт лічильника команд, а потім переходить у високоімпедансне становище і очікує приходу байта із ППЗП програм. Коли молодший байт адреси перебуває на виходах порту P0, сигнал ALE залишає його в адресному регістрі RG. Старший байт адреси перебуває на виходах порту P2 протягом усього часу звертання до ППЗП. Сигнал PME дозволяє вибірку байта із ППЗП, після чого обраний байт надходить на порт P0 МК51 і вводиться в МК.

**Пам'ять даних** призначена для приймання, зберігання і видачі інформації, використаної в процесі виконання програми. Пам'ять даних, розташована на кристалі МК, складається з регістру адреси ОЗП, дешифратора, ОЗП і покажчика стека.

**Регістр адреси ОЗП** призначений для приймання і зберігання адреси обраної за допомогою дешифратора комірки пам'яті, яка може містити як біт, так і байт інформації.

**ОЗП являє собою** 128 восьмирозрядних регістрів, призначених для приймання, зберігання і видачі різної інформації.

**Покажчик стека являє собою** восьмирозрядний регістр, призначений для приймання і зберігання адреси гнізда стека, до якої було останнє звернення. При виконанні команд LCALL, ACALL вміст покажчика стека збільшується на 2. При виконанні команд RET, RETI вміст покажчика стека зменшується на 2. При виконанні команди PUSH direct вміст покажчика стека збільшується на 1. При виконанні команди POP direct вміст покажчика стека зменшується на 1. Після скидання в покажчику стека встановлюється адреса 07H, що відповідає початку стека з адресою 08H.

У МК передбачена можливість розширення пам'яті даних шляхом підключення зовнішніх пристроїв ємністю до 64 Кбайт. При цьому звертання до зовнішньої пам'яті даних можливо тільки за допомогою команд MOVX.

Команди MOVX ©ri, A і MOVX A, @Ri формують восьмирозрядну адресу, видану через порт P0. Команди MOVX @DPTR, A і MOVX @A, DPTR формують 16-розрядну адресу, молодший байт якої видається через порт P0, а старший- через порт P2.

Байт адреси, виданий через порт P0, повинен бути зафіксований у зовнішньому регістрі по спаду сигналу АЛП ALE, тому що надалі лінії порту P0 використовуються як шина даних, через яку байт даних приймається з пам'яті при читанні або видається в пам`ять даних при записі. При цьому читання стробується сигналом RD, а запис – сигналом WR. При роботі із внутрішньою пам'яттю даних сигнали RD і WR не формуються.

**Порти P0, P1, P2, P3** є двонаправлені порти введення-виводу і призначені для забезпечення обміну інформацією МК із зовнішніми пристроями, утворюючи 32 лінії вводу-виводу. Кожний з портів містить фіксатор-засувку, який являє собою восьмирозрядний регістр, що має байтову і бітову адресацію для установки (скидання) розрядів за допомогою програмного забезпечення.

Фізичні адреси P0, P1, P2, P3 становлять для:

- P0 – 80H, при бітовій адресації 80H -87H; P2 – A0H, при бітовій адресації A0H – A7H;
- P1 – 90H, при бітовій адресації 90H -97H; P3 – B0H, при бітовій адресації B0H – B7H.

Крім роботи як звичайних портів введення/виводу лінії портів P0 – P3 можуть виконувати ряд додаткових функцій, описаних нижче.

**Структура мікроконтролера КР1816ВЕ51 (МК51).**

Умовне графічне позначення мікроконтролера МК51 показано на рис. А.1. МК51 працює від внутрішнього генератора, який має зовнішні виводи для підключення кварцового резонатора.

Призначення виводів наведено в таблиці 1.3:

Uss - потенціал загального дрота ("землі"); Ucc

- основна напруга живлення +5 В;

X1, X2 - виводи для підключення кварцового резонатора;

RST/VPD - вхід загального скидання/підключення резервного джерела живлення;

PSEN - дозвіл зовнішньої пам'яті програм; видається тільки при звертанні до зовнішнього ПЗП;

ALE - строб адреси зовнішньої пам'яті;

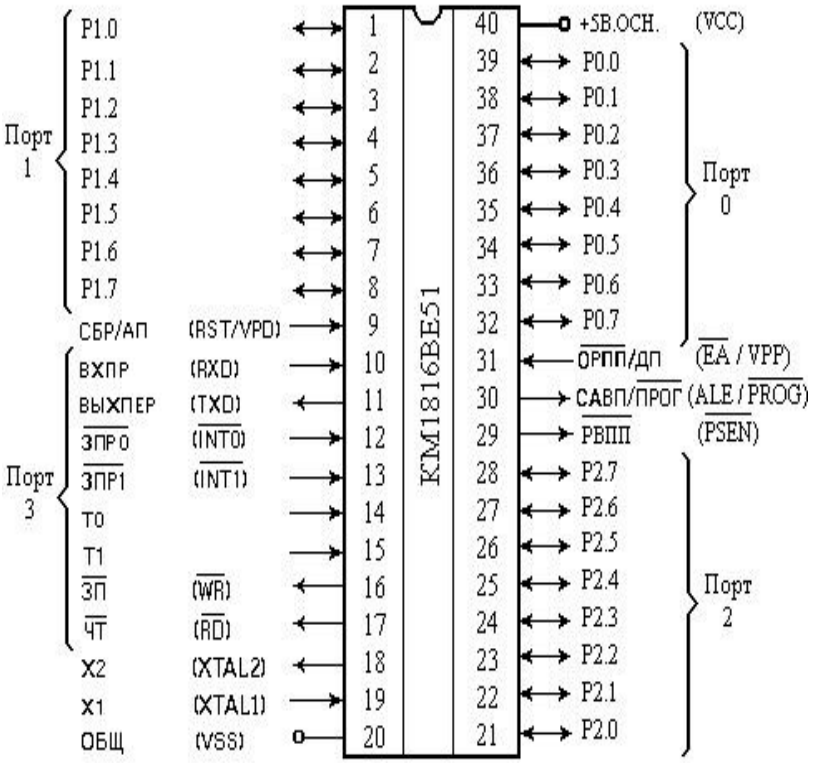



Рисунок А.1 - Умовне графічне позначення мікроконтролера МК51

EA - відключення внутрішньої програмної пам'яті; рівень 0 на цьому вході змушує мікроконтролер виконувати програму тільки зовнішньої ПЗП; ігноруючи внутрішню (якщо остання є);

P0 - восьмибітний двонаправлений порт введення-виводу інформації: при роботі із зовнішніми ОЗП і ПЗП по лініях порту в режимі тимчасового мультиплексування видається адреса зовнішньої пам'яті, після чого здійснюється передача або приймання даних;

P1 - восьмибітний квазідвонаправлений порт введення/виводу: кожний розряд порту може бути запрограмований як на введення, так і на вивід інформації, незалежно від стану інших розрядів;

P2 - восьмибітний квазідвонаправлений порт, аналогічний P1; крім того, виводи цього порту використовуються для видачі адресної інформації (старшого байта) при звертанні до зовнішньої пам'яті

програм або даних (якщо використовується 16-бітова адресація останньої). Виводи порту використовуються так само для підключення і програмування розширника ( К580ВР43);

Р3 - восьмибітний квазідвонаправлений порт, аналогічний Р1; крім того, виводи цього порту можуть виконувати ряд альтернативних функцій, які використовуються при роботі таймерів, порту послідовного введення-виводу, контролера переривань і зовнішньої пам'яті програм і даних.

Альтернативні функції порту Р3:

Rxd – вхід приймача послідовного порту в режимі УАПП. Введення/вивід даних у режимі регістру, що зрушує;

Txd – вихід передавача послідовного порту в режимі УАПП. Вихід синхронізації в режимі регістру, що зрушує;

INT0 (інв), INT1 (інв) – входи запитів на переривання (по низькому рівню або зрізі);

Т0, Т1 – входи таймерів-лічильників або тест-входи;

RD (інв), WR (інв) – сигнали керування читання і запису (формуються при звертанні до зовнішньої пам'яті).

Структурна організація і система команд МК51 добре описана в [1] і наведена на рисунку 1.2. Відзначимо, що МК можна представити у вигляді наступних основних вузлів:

- арифметико-логічний пристрій (АЛП);
- пам'ять даних і програм;
- акумулятор;
- регістр слова стану (регістр ознак);
- 8 - бітний регістр - покажчик стека;
- 16 - бітний регістр - покажчик даних;
- блок таймерів - лічильників;
- послідовний порт;
- регістри спеціальних функцій;
- пристрою керування і синхронізації;
- порти введення - виводу.

МК51 працює від внутрішнього генератора, який має зовнішні виводи для підключення кварцового резонатора.

Таблиця А. 3- Призначення виводів МК51

| N виводу | Позначення | Призначення | Тип |
|---|---|---|---|
| 1-8 | P1.0-P1.7 | 8-розрядний двонаправлений порт Р1. Вхід адреси А 0-А7 при перевірці внутрішнього ПЗП (РПЗП). | вхід/ вихід |
| 9 | RST | Сигнал загального скидання. Вивід резервного живлення ОЗП від зовнішнього джерела (для 1816). | вхід |
| 10-17 | P3.0-P3.7 | 8-розрядний двонаправлений порт Р3 з додатковими функціями: | вхід/ вихід |
| | P3.0 | Послідовні дані приймача - Rxd | вхід |
| | P3.1 | Послідовні дані передавача - Txd | вихід |
| | P3.2 | Вхід зовнішнього переривання 0- $\overline{INT\ 0}$ | вхід |
| | P3.3 | Вхід зовнішнього переривання 1- $\overline{INT\ 1}$ | вхід |
| | P3.4 | Вхід таймера/лічильника 0: - Т0 | вхід |
| | P3.5 | Вхід таймера/лічильника 1: - Т1 | вхід |
| | P3.6 | Вихід стробуючого сигналу АЛП при записі в зовнішню пам'ять даних: - $\overline{WR}$ | вихід |
| | P3.7 | Вихід стробуючого сигналу АЛП при читанні із зовнішньої пам'яті даних - $\overline{RD}$ | вихід |
| 18<br>19 | BQ2<br>BQI | Виводи для підключення кварцового резонатора. | вихід<br>вхід |
| 20 | 0 В | Загальний вивід | |
| 21-28 | P2.0-P2.7 | 8-розрядний двонаправлений порт Р2. Вихід адреси А 8-А15 у режимі роботи із зовнішньою пам'яттю. У режимі перевірки внутрішнього ПЗП виводи Р2.0 - Р2.6 використовуються як вхід адреси А8-А1 +. Вивід Р2.7 - дозвіл читання ПЗП: - Е | вхід/ вихід |
| 29 | $\overline{PME}$ | Дозвіл програмної пам'яті | вихід |
| 30 | ALE | Вихідний сигнал дозволу фіксації адреси. При програмуванні РПЗП сигнал: - PROG | вхід/ вихід |

Продовження таблиці А.3

| N виводу | Позначення | Призначення | Тип |
|---|---|---|---|
| 31 | DEM A | Блокування роботи із внутрішньою пам'яттю. При програмуванні РПЗП подається сигнал UPR | вхід/ вихід |
| 32-39 | P0.7-P0.0 | 8-розрядний двонаправлений порт P0. Шина адреси даних при роботі із зовнішньою пам'яттю. Вихід даних D7-D0 у режимі перевірки внутрішнього ПЗП (РПЗ). | вхід/ вихід |
| 40 | Ucc | Вивід живлення від джерела напруги +5 В. | |

**Порти** (0, 1, 2, 3) МК51 служать для побайтного введення - виводу даних. Крім того, порти 0 і 2 служать для виводу адреси. Виводи порту 3 можуть бути використані для реалізації альтернативних функцій. Порт 0 є двонаправленним, порти 1, 2, 3 - квазідвунаправлені: кожна лінія порту може бути налаштована для введення або виводу даних. Навантажувальна здатність портів 1, 2, 3 - один ТТЛ - вхід, а порту 0 - два ТТЛ входу. PC - 16-розрядний програмний лічильник.

**Акумулятор** (А) служить для зберігання як операнда, так і результату операції. Але МК51 може виконувати ряд команд і без участі акумулятора. Виконання багатьох команд проводиться в арифметико-логічному пристрої АЛП, а ряд ознак операцій фіксується в регістрі слова стану програми (PSW)- РССП.

**Арифметико-логічний пристрій** (АЛП): 8-бітний і може виконувати арифметичні операції додавання, вирахування, множення і ділення; логічні операції І, АБО, що виключає АБО, а також операції циклічного зрушення, скидання, інвертування і т.п. В АЛП є програмно недоступні регістри Т1 і Т2, призначені для тимчасового зберігання операндів, схеми десяткової корекції і формування ознак.

**Восьмирозрядний покажчик стека** (SP) може адресувати будь-яку область пам'яті даних. Уміст його зменшується на одиницю в ході виконання команд PUSH і CALL і збільшується на одиницю при виконанні команд POP і RET. При скиданні в SP автоматично завантажується початковий код 07 Н.

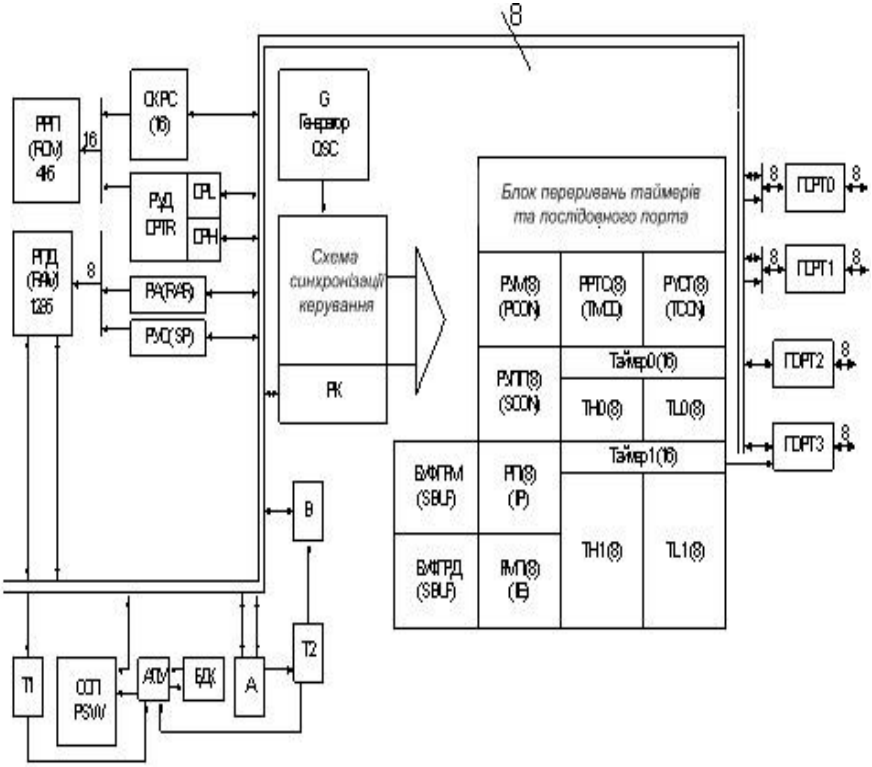



Рисунок А.2 – Структурна схема мікроконтролера КР1816ВЕ51 (МК51)

**16-розрядний регістр-покажчик даних** (DPTR) служить для фіксації адреси при звертанні до зовнішньої пам'яті. Він може працювати як два незалежні 8-розрядних регістри DPH і DPL.

МК51 має два незалежні програмувальні 16-розрядних таймера/лічильника, представлених у вигляді двох регістрових пар: TH0, TL0 і TH1, TL1, буфер послідовного порту SBUF - два незалежні 8 - розрядних регістру: регістр приймача і регістр передавача, регістри спеціальних функцій з іменами IP, IE, TMOD, TCON, SCON, PCON, що дозволяють програмувати схему переривання, режими роботи таймерів-лічильників і прийомопередавача.

**Пам'ять програм і пам'ять даних**, розміщені на кристалі МК51, фізично і логічно розділені, мають різні механізми адресації, працюють під керуванням різних сигналів і виконують різні функції.

**Пам'ять програм**: **постійний запам`ятовуючий пристрій** (ПЗП) має ємність 4 Кбайта і призначений для зберігання: команд, констант, що управляють словами ініціалізації (при включенні), таблиць перекодування вхідних і вихідних змінних і т.п. **Регістр** ПЗП має 16-бітну шину адреси, через яку забезпечується доступ з лічильника команд або з **регістру-покажчика даних** (РПД). Останній виконує функції базового регістру при непрямих переходах по програмі або використовується в командах, що оперують із таблицями. Оперативний запам`ятовуючий пристрій (ОЗП) потрібен для зберігання змінних у процесі виконання прикладної програми, адресується одним байтом і має ємність 128 байт. Крім того, до адресного простору РПД примикають адреси **регістрів спеціальних функцій** (РСФ), які перераховано в таблиці 1.2.

### 1.4 Арифметико-логічний пристрій (АЛП). Регістр PSW

АЛП являє собою паралельний восьмирозрядний пристрій, що забезпечує виконання арифметичних і логічних операцій, а також операції логічного зрушення, скидання, установки і т.д..

АЛП складається з регістру акумулятора, регістру тимчасового зберігання, ПЗУ констант, суматора, додаткового регістру (регістру В), акумулятора, регістру стану програми.

**Регістри акумулятора і тимчасового зберігання** — восьмирозрядні регістри, призначені для приймання і зберігання операндів на час виконання операцій над ними. Програмно не доступні.

**ПЗУ констант** забезпечує створення коригувального коду при двоїчно-десятковому представленні даних, коду маски при бітових операціях і коду констант.

**Паралельний восьмирозрядний суматор -** являє собою схему комбінаційного типу з послідовним переносом, призначену для виконання арифметичних операцій додавання, вирахування і логічних операцій додавання, множення, нерівнозначності і тотожності.

**Регістр В -** восьмирозрядний регістр, використовуємий під час операцій множення і ділення. Для інших інструкцій він може розглядатися як додатковий надоперативний регістр.

**Акумулятор -** являє собою восьмирозрядний регістр, призначений для приймання і зберігання результату, отриманого при виконанні арифметико-логічних операцій або операцій пересилання.

**Регістр стану програми (PSW) -** призначений для зберігання інформації про стан АЛП при виконанні програми. Позначення розрядів регістру PSW і призначення розрядів наведені в табл.А.4.

Таблиця А.4 - Формат слова стану програми (ФССП)

| Символ | Позиція /№ біта | Ім'я і призначення |
|---|---|---|
| CY | PSW.7 | Прапор переносу. Встановлюється і скидається при виконанні арифметичних і логічних операцій |
| AC | PSW.6 | Прапор допоміжного переносу. Встановлюється і скидається при виконанні команд додавання і вирахування і сигналізує про перенос або позику в біті 3 |
| F0 | PSW.5 | Прапор 0. Може бути встановлений, скинутий або перевірений програмою як прапор, специфікуємий користувачем |
| RS1<br>RS0 | PSW.4<br>PSW.3 | Вибір банку регістрів. Встановлюється і скидається програмою для вибору робочого банку регістрів |
| 0V | PSW.2 | Прапор переповнення. Встановлюється і скидається при виконанні арифметичних операцій |
| - | PSW.1 | Не використовується |
| P | PSW.0 | Прапор паритету. Встановлюється і скидається в кожному циклі команди і фіксує непарне/парне число одиничних біт в акумуляторі |

| Примітка | RS1 | RS0 | Банк | Границі адрес |
|---|---|---|---|---|
| | 0 | 0 | 0 | 00H-07H |
| | 0 | 1 | 1 | 08H-0FH |
| | 1 | 0 | 2 | 10H-17H |
| | 1 | 1 | 3 | 18H-1FH |

**Прапор переносу** CY може встановлюватися і скидатися як апаратними, так і програмними засобами. Прапор CY може бути програмно прочитаний. Апаратними засобами прапор CY встановлюється, якщо в старшому біті результату виникає перенос або позика. При виконанні операцій множення і розподілу прапор CY скидається. Крім того, прапор CY виконує функції "булевого акумулятора" у командах, що працюють із бітами.

**Прапор додаткового переносу** AC програмно доступний по запису ("0" і "1") і читанню.

Прапори F0, RS1, RS0 програмно доступні по запису ("0" і "1") і читанню. Прапор переповнення OV програмно доступний по запису ("0" і "1") і читанню. Встановлюється апаратно, якщо результат операції додавання/вирахування не укладається в семи бітах і старший (восьмий) біт результату не може інтерпретуватися як знаковий. При виконанні операції розподілу прапор OV апаратно скидається, а у випадку розподілу на нуль встановлюється. При множенні прапор OV апаратно встановлюється, якщо результат більше 255.

**Прапор P** є доповненням вмісту акумулятора до парності. В 9-розрядному слові, що полягає з 8 розрядів акумулятора і біта P, завжди втримується парне число одиничних бітів. У випадку, якщо в акумуляторі всі розряди встановлені в "0", прапор P прийме нульове значення. Програмно доступний тільки по читанню.

У таблиці 1.5 приводиться перелік прапорів ФССП, даються їхні символічні імена і описуються умови їх формування. Особливості програмування регістрів наведено в [1,3].

### А.5 Блок таймерів/лічильників. Регістри TMOD і TCON

Таймери/лічильники (Т/Л) призначені для підрахунку зовнішніх подій, для одержання програмно керованих тимчасових затримок і виконання часових функцій МК . До складу блоку Т/Л входять:

- два 16-розрядних регістри Т/Л 0 і Т/Л 1;
- восьмирозрядний регістр режимів Т/Л (TMOD);
- восьмирозрядний регістр керування (TCON);
- схема інкремента;
- схема фіксації $\overline{INT}$ 0, $\overline{INT}$ 1, T0, T1;
- схема керування прапорами;
- логіка керування Т/С.

**Два 16-розрядних регістри Т/Л0 і Т/Л1** виконують функцію зберігання вмісту рахунку. Кожний з них складається з пари восьмирозрядних регістрів, відповідно TH0, TL0 і TH1, TL1. Причому регістри TH0, TH1- старші, а регістри TL0, TL1- молодші 8 розрядів. Кожний з восьмирозрядних регістрів має свою адресу і може бути використаний як РОН, якщо Т/Л не використовуються (біт TR0 для Т/Л0 і біт TR1 для Т/Л1 у регістрі керування TCON рівні "0").

Позначення розрядів регістру TMOD наведено в табл.А.5.

Таблиця А.5 - Регістр **TMOD**

| Біти | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Позначен. | GATE1 | C/Tl | Ml.l | M0.1 | GATE0 | C/T0 | M1.0 | M0.0 |

Призначення розрядів регістру TMOD наведено в таблиці А.6.

Таблиця А.6 - Регістр **TMOD**

<table>
<tr><th>Біти</th><th>Найменування</th><th colspan="3">Призначення бітів</th><th>Примітка</th></tr>
<tr><td rowspan="6">1, 4, 5</td><td rowspan="6">M0-M1</td><td colspan="3">Визначають один з 4-х режимів роботи, окремо для Т/Л1 і Т/Л0</td><td rowspan="6">Усі біти встановлюються програмно; біти 0-З визначають режим роботи Т/Л0, біти 4-7 визначають режим роботи Т/Л1.</td></tr>
<tr><td>M1</td><td>M0</td><td>Режим</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>2</td></tr>
<tr><td>1</td><td>1</td><td>3</td></tr>
<tr><td>2, 6</td><td>C/T0<br>C/T1</td><td colspan="3">Визначають роботу в якості:<br>З/Т 0, З/Т 1= 0 - таймера<br>С/Т 0, З/Т 1= 1 - лічильника</td><td></td></tr>
<tr><td>3, 7</td><td>GATE</td><td colspan="3">Дозволяє управляти таймером від зовнішнього виводу ( 0- $\overline{INT}$ для Т/Л0, $\overline{INT}$ 1- для Т/Л1).<br>GATE=0- керування заборонене<br>GATE= 1- керування дозволене</td><td></td></tr>
</table>

Код величини початкового рахунку заноситься в регістри Т/Л програмно. У процесі рахунку вміст регістрів Т/Л інкрементується. Ознакою закінчення рахунку, як правило, є переповнення регістру Т/Л, тобто перехід його вмісту зі стану "усі одиниці" у стан "усі нулі". Усі регістри TH0, TH1, TL0, TL1 доступні по читанню, і при необхідності, контроль досягнення необхідної величини рахунку може виконуватися програмно.

Регістр режимів Т/Л (TMOD) призначений для приймання і зберігання коду, що визначає:

- один з 4-х можливих режимів роботи кожного Т/Л;

- роботу як таймери або лічильника;
- керування Т/Л від зовнішнього виводу.

При роботі в якості таймера вміст регістру Т/Л інкрементується в кожному машинному циклі, тобто Т/Л є лічильником машинних циклів МК. Оскільки машинний цикл складається з 12 періодів частоти синхронізації МК f, то частота рахунку в цьому випадку рівна f/12.

При роботі Т/Л у якості лічильника зовнішніх подій уміст регістру Т/Л інкрементується у відповідь на перехід з "1" в "0" сигнал АЛП на вході лічильнику МК (вивід Т0 для Т/Л0 і вивід Т1 для Т/Л1). Входи лічильника апаратно перевіряються у фазі S5P2 кожного машинного циклу. Коли перевірки показують високий рівень на вході лічильника в одному машинному циклі і низький рівень в іншому машинному циклі, регістр Т/Л інкрементується. Нове (інкрементоване) значення заноситься в регістр Т/Л у фазі S3PI машинного циклу, що безпосередньо випливає за тим, у якім був виявлений перехід з "1" в "0" на рахунковому вході МК. Так як для розпізнавання такого переходу потрібно два машинні цикли (24 періоду частоти синхронізації МК f), та максимальна частота рахунку Т/Л у режимі лічильника рівна f/24.

Щоб рівень сигналу АЛП на рахунковому вході був гарантовано зафіксований, він повинен залишатися незмінним протягом як мінімум одного машинного циклу.

Регістр керування (TCON) призначений для приймання і зберігання коду керуючого слова. Позначення розрядів регістру TCON наведене в табл. А.7. Призначення розрядів регістру TCON наведене в табл. А.8.

Таблиця А.7 - Регістр **TCON**

| Біти | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Позначення | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

Таблиця А.8 - Регістр **TCON**

| Біти | Найменування | Призначення бітів | Примітка |
|---|---|---|---|
| 6<br>4 | TR1<br>TR0 | Біти вимикання Т/З, окремо для Т/Л0 і Т/Л1.<br>TR=0 - виключений,<br>TR=1 - включений. | Біти встановлюються й скидаються програмно. Доступні по читанню. |
| 7<br>5 | TF1<br>TF0 | Прапори переповнення Т/С. | Біти скидаються й встановлюються апаратно і програмно. Доступні по читанню. |
| 2<br>0 | IT1<br>1T0 | Біти, що визначають вид переривання по входах $\overline{INT}$ 0, $\overline{INT}$ 1:<br>IT=0 - переривання за рівнем (низькому)<br>IT=1 - переривання по фронту<br>(перехід з "1" в "0") | Біти встановлюються і скидаються програмно Доступні по читанню. |
| 3<br>1 | IE1<br>1E0 | Прапори запиту зовнішніх переривань по входах $\overline{INT}$ 0, $\overline{INT}$ 1 | Біти скидаються і встановлюються апаратно і програмно. Доступні по читанню. |
| | | | Біти 4,5 ставляться до Т/Л0; біти 6,7- до Т/Л1. Біти 0,1 визначають зовнішні переривання по входу $\overline{INT}$ 0, біти 2.3 - по входу $\overline{INT}$ 1. |

Прапори переповнення TF0 і TF1 встановлюються апаратно при переповненні відповідних Т/Л (перехід Т/Л із стану "усі одиниці" у стан "усі нулі"). Якщо при цьому переривання від відповідного Т/Л дозволене, то установка прапора TF викличе переривання. Прапори TF0 і TF1 скидаються апаратно при передачі керування програми обробки відповідного переривання.

Прапори TF0 і TF1 програмно доступні і можуть бути встановлені/скинуті програмою. Використовуючи цей механізм, переривання по TF0 і TF1 можуть бути викликані (установка TF) і скасовані (скидання TF) програмою.

Прапори 1Е0 і 1Е1 встановлюються апаратно від зовнішніх переривань (відповідно входи МК $\overline{INT}$ 0 і $\overline{INT}$ 1) або програмно і ініціюють виклик програми обробки відповідного переривання. Скидання цих прапорів виконується апаратно при обслуговуванні переривання тільки в тому випадку, коли переривання було викликано по фронту сигналу АЛП. Якщо переривання було викликано рівнем сигналу АЛП на вході $\overline{INT}$ 0 ( $\overline{INT}$ 1), то скидання прапора 1Е повинна виконувати програма обслуговування переривання, впливаючи на джерело переривання для зняття їм запиту. Схема інкремента призначена:

- для збільшення на 1 у кожному машинному циклі вмісту регістрів Т/Л0, Т/Л1, для яких встановлений режим таймера і рахунок дозволений;

- для збільшення на 1 вмісту регістрів Т/Л0, Т/Л1, для яких установлений режим лічильника, рахунок дозволений і на відповідному вході МК (Т0 для Т/Л0 і Т1 для Т/Л1) зафіксований рахунковий імпульс.

**Схема фіксації** $\overline{INT}$ 0, $\overline{INT}$ 1, Т0, Т1 являє собою чотири тригери. У кожному машинному циклі в момент S5P2 у них запам'ятовується інформація з виводів МК INT0, $\overline{INT}$ 1, Т0, Т1. **Схема керування прапорами** виробляє і знімає прапори переповнення Т/С і прапори запитів зовнішніх переривань.

Логіка керування Т/Із синхронізує роботу регістрів Т/Л0 і Т/Л1 відповідно до запрограмованих режимів роботи і синхронізує роботу блоку Т/С з роботою МК .

**Режим роботи кожного Т/Л** визначається значенням бітів М0, М1 у регістрі TMOD. Т/Л0 і Т/Л1 мають чотири режими роботи. Режими роботи 0, 1, 2 однакові для обох Т/Л; Т/Л0 і Т/Л1 у цих режимах повністю незалежні друг від друга. Робота Т/Л0 і Т/Л1 у режимі 3 різна. При цьому установка режиму 3 у Т/Л0 впливає на режими роботи Т/Л1.

**Установка бітів М0=0, М1=0 визначає режим роботи 0.** Т/С у режимі 0 являє собою пристрій на основі 13-розрядного регістру.

13-розрядний регістр полягає для Т/Л0 з 8 розрядів регістру TH0 і 5 молодших розрядів регістру TL0, а для Т/Л1 - з 8 розрядів регістру TH1 і 5 молодших розрядів регістру TL1.

У цьому режимі функцію дільника на 32 виконують регістри TL0, TL1. Вони є програмно доступними, але треба пам'ятати, що значущими в режимі 0 є тільки п'ять молодших розрядів регістрів TL0, TL1. Логіка роботи в режимі 0 на прикладі Т/Л1 показана на рис. 1.3. Для Т/Л0 логіка роботи аналогічна. На рис. А.3-А.6 OSC — джерело синхронізації МК (внутрішній або зовнішній). На виході OSC — частота f. Біт З/Т регістру TMOD визначає роботу Т/Л або в якості таймера (З/Т=0), або в якості лічильника (З/Т=1). Рахунок починається при установці біта TR регістру TCON у стан "1". При необхідності керування рахунком ззовні біт GATE регістру TMOD встановлюється в стан "1". Тоді при TR=1 рахунок буде дозволений,

якщо на вході $\overline{INT}$ 0 ( для Т/Л0) або $\overline{INT}$ 1 ( для Т/Л1) встановлений стан "1" і буде заборонений, якщо встановлений стан "0". Установка біта TR0 для Т/Л0 і TR1 для Т/Л1 у стан "0" виключає Т/Л незалежно від стану інших бітів.

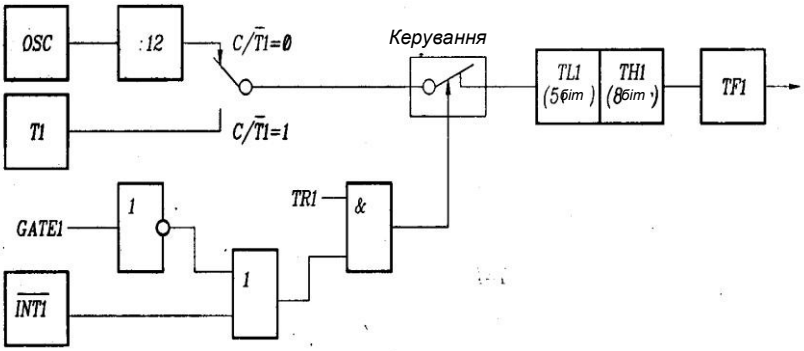



Рисунок А.3 - Логіка роботи Т/Л1 у режимі 0

При переповненні Т/Л(перехід умісту регістру Т/Із зі стану "усі одиниці" у стан "усі нулі") встановлюється прапор TF0 для Т/Л0 або TF1 для Т/Л1 у регістрі TCON.

Установка бітів M1=0, M0=1 визначає режим роботи 1. Режим 1 аналогічний режиму 0. Відмінність полягає в тому, що установка режиму 1 перетворює Т/С у пристрій на основі 16-розрядного регістру. Для Т/Л0 регістр складається із програмно доступних пар TL0, TH0, для Т/Л1 із програмно доступних пар TL1, TH1. Логіка роботи в режимі 1 на прикладі Т/Л1 показана на рис. А.4.

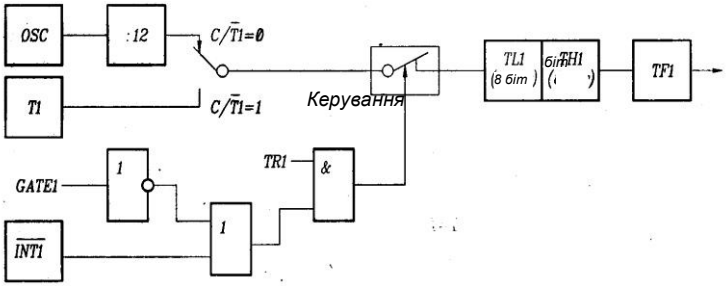



Рисунок А.4 - Логіка роботи Т/Л1 у режимі 1

Установка бітів М1=1, М0=0 визначає режим 2. У цьому режимі Т/Лявляє собою пристрій на основі восьмирозрядного регістру TL0 для Т/Л0 і TL1 для Т/Л1. При кожнім переповненні TL0 крім установки в регістрі TCON прапора TF0 відбувається автоматично перезавантаження вмісту із TH0 в TL0. Відповідно для Т/Л1 при переповненні TL1 у регістрі TCON установлюється прапор TF1 і відбувається перезавантаження TL1 із TH1. Регістри TH0 і TH1 завантажуються програмно. Перезавантаження TL0 із TH0 і TL1 із TH1 не впливає на вміст регістрів TH0 і TH1. Логіка роботи Т/Л1 у режимі 2 показана на рис. 1.5. Логіка роботи Т/Л0 у режимі 2 аналогічна. Призначення бітів керування TR0, TR1, GATE0, GATE1, З/Т 0, З/Т 1 таке ж як режимі 0.

**Установка бітів М1=1, М0=1 визначає режим 3.** Т/Л1 у режимі 3 заблокований і просто зберігає свій рахунок (значення коду в регістрі Т/З). Ефект такий же, як при установці TR1=0.

Т/Л0 у режимі 3 являє собою два незалежні пристрої на основі восьмирозрядних регістрів TL0 і TH0. Пристрій на основі регістру TL0 може працювати в режимі таймера і у режимі лічильника. За ним зберігаються всі біти керування Т/Л0, воно реагує на впливи по входах Т0, $\overline{INT}$ 0. При переповненні TL0 встановлюється прапор TF0. Пристрій на основі регістру TH0 може працювати тільки в режимі таймера. Воно використовує біт включення TR1, при переповненні TH0 виставляє прапор TF1. Інших бітів керування пристрій на основі TH0 у цьому режимі не має. Логіка роботи Т/Л0 у режимі 3 показана на рис. А.6.

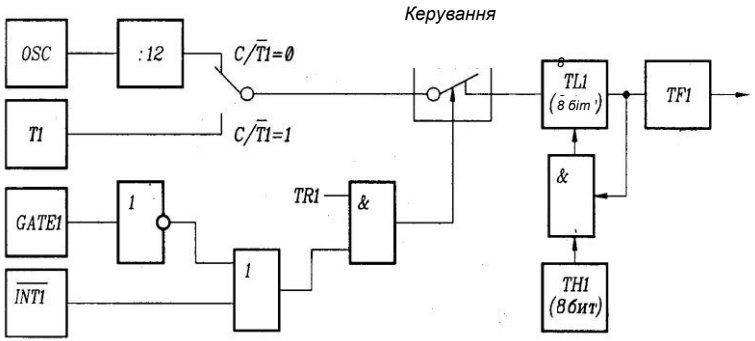



Рисунок А.5 - Логіка роботи Т/Л1 у режимі 2

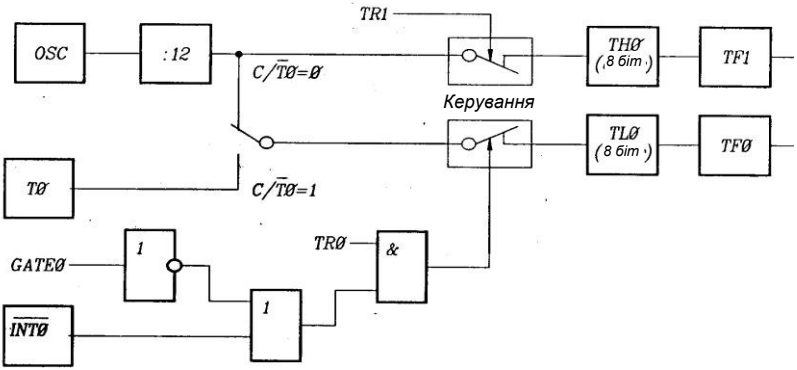



Рисунок А.6 - Логіка роботи Т/Л0 у режимі 3

Установка Т/Л0 у режим 3 позбавляє Т/Л1 біта включення TR1. Тому Т/Л1 у режимах 0, 1, 2 при GATE1=0 завжди включений і при переповненні в режимах 0 і 1 Т/Л1 скидається, а в режимі 2 перезавантажується не встановлюючи прапор, якщо Т/Л0 перебуває в режимі 3. Керування від входів $\overline{INT}$ 1, Т1, біти керування С/Т1, GATE1 для Т/Л1 не залежать від режиму Т/Л0.

Т/Л1 апаратно пов'язаний із блоком синхронізації послідовного інтерфейсу (ПІ). При роботі в режимах 0, 1, 2 при переповненні Т/Л1 завжди виробляє тактовий імпульс ПІ. Тому 3-й режим Т/Л0 зручно застосовувати тоді, коли потрібна робота ПІ і двох таймерів або ПІ, таймера і лічильника.

Коли Т/Л0 переведено в режим 3, Т/Л1 можна виключити, перевівши його також у режим 3, використовувати з послідовним портом для виробітку імпульсів тактировки або в будь-які інших додатках переривання, що не вимагають.

**Додаткова інформація.** Вимикання Т/Л за допомогою бітів TR0, TR1 (скидання цих бітів в"0") або за допомогою входів МК $\overline{INT}$0, $\overline{INT}$1 (установка на цих входах логічного "0" при GATE=1) не спотворює код, що перебуває в регістрі Т/Л. Т/Л можна виключити, через довільне, час знову включити і рахунок почнеться з тієї величини, яка була в регістрі Т/Л на момент вимикання (якщо, звичайно, після вимикання регістр Т/Л не перезаписувався)

Нове завантаження Т/Л відразу ж означає нову величину рахунку, а стара губиться. Якщо завантаження зроблене при включеному Т/З, рахунок продовжиться з нової величини. Черговість завантаження регістрів TL0, TH0, TL1, TH1 — довільна.

У всіх режимах, крім режиму 2, після переповнення Т/Л рахунок триває з величини 00H, якщо Т/Лне виключити за допомогою бітів TR0, TR1 або входів $\overline{INT}$0, $\overline{INT}$1.

**1.6 Блок послідовного інтерфейсу і переривань. Регістри SCON, IP, IE**

Блок послідовного інтерфейсу і переривань ( ПІП) призначений для організації введення-виводу послідовних потоків інформації і організації системи переривання програм.

До складу блоку ПІП входять: буфер ПІП, логіка керування ПІП, регістр керування, буфер передавача, буфер приймача, приймач/передавач послідовного порту, регістр пріоритетів переривань, регістр дозволу переривань, логіка обробки прапорів переривань і схема виробітку вектора.

**Буфер ПІП** забезпечує побайтовий обмін інформацією між внутрішньою магістраллю даних і шиною ПІП.

**Логіка керування ПІП** призначена для виробітку сигналів керування, що забезпечують чотири режими роботи послідовного інтерфейсу, і організації переривання програм.

Послідовний інтерфейс (послідовний порт) МК51 може працювати в наступних чотирьох режимах:

**Режим 0.** Інформація передається і ухвалюється через вхід приймача Rxd (вивід P3.0 МК). Через вихід передавача Txd (вивід P3.1

МК) видаються імпульси синхронізації, що стробують кожний переданий або прийнятий біт інформація. Формат посилки — 8 біт. Частота приймання і передачі — f/12, де f — тактова частота МК.

**Режим 1.** Інформація передається через вихід передавача Txd, а ухвалюється через вхід приймача Rxd. Формат посилки — 10 біт: старт-біт (нуль), вісім біт даних і стоп-біт (одиниця). Частота приймання і передачі задається Т/Л1.

**Режим 2.** Інформація передається через вихід передавача Txd, а ухвалюється через вхід приймача Rxd. Формат посилки — 11 біт: старт-біт (нуль), вісім біт даних, програмувальний дев'ятий біт і стоп-біт (одиниця). Переданий дев'ятий біт даних ухвалює значення біта TB8 з регістру спеціальних функцій SCON. Біт TB8 у регістрі SCON може бути програмно встановлений в " ПРО" або в "1", або в нього, приміром, можна помістити значення біта P з регістру PSW для підвищення вірогідності прийнятої інформації (контроль по паритету). При прийманні дев'ятий біт даних прийнятої посилки надходить у біт RB8 регістру SCON. Частота приймання і передачі в режимі 2 задається програмно і може бути рівна f/32 або f/64.

**Режим 3**. Режим 3 повністю ідентичний режиму 2 за винятком частоти приймання і передачі, яка в режимі 3 задається Т/Л1. Докладно робота послідовного порту описано в розділі 2.3.3.

**Регістр керування** (SCON) призначений для приймання і зберігання коду 8-бітних слів, що управляють послідовним інтерфейсом. Позначення розрядів регістру SCON наведено в табл. А.9 (усі розряди SCON програмно доступні по запису ("0"і"1") і читанню).

Розряди SM0, SMI визначають режим роботи ППП, як зазначено в табл. А.10.

Інші біти регістру мають наступне призначення:

SM2 - дозвіл багатопроцесорної роботи. У режимах 2 і 3 при SM2=1 прапор RI не активізується, якщо дев'ятий прийнятий біт даних рівний "0". У режимі 1 при SM2=1 прапор RI не активізується, якщо не прийнятий стоп-біт, рівний "1". У режимі 0 біт SM2 повинен бути встановлений в "0".

REN - дозвіл приймання послідовних даних. Встановлюється і скидається програмним забезпеченням відповідно для дозволу і заборони приймання.

Таблиця А.9 - Регістр керування послідовним інтерфейсом **SCON**

| Біти | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Позначення | SM0 | SMI | SM2 | REN | TB8 | RB8 | T1 | RI |

Таблиця А.10 - Регістр керування послідовним інтерфейсом **SCON**

| SM0 | SMI | Режим | Найменування | Швидкість передачі |
|---|---|---|---|---|
| 0 | 0 | 0 | Регістр зсуву | f/12 |
| 0 | 1 | 1 | 8-бітовий універсальний асинхронний приймач/передавач (УАПП) | змінна, задається Т/Л1 |
| 1 | 0 | 2 | 9-бітовий (УАПП) | f/64 або f /32 |
| 1 | 1 | 3 | 9-бітовий (УАПП) | змінна, задається Т/Л1 |

TB8 - дев'ятий біт переданих даних у режимах 2 і 3. Встановлюється і скидається програмним забезпеченням.

RB8 - дев'ятий біт прийнятих даних у режимах 2 і 3. У режимі 1, якщо SM2=0, RB8 є прийнятим стоп-бітом. У режимі 0 біт RB8 не використовується.

TI - прапор переривання передавача. Встановлюється апаратно наприкінці часу видачі 8-го біта в режимі 0 або на початку стоп-біта в інших режимах. Скидається програмним забезпеченням.

RI - прапор переривання приймача. Встановлюється апаратно наприкінці часу приймання 8-го біта в режимі 0 або через половину інтервал біту в режимах 1, 2, 3 при SM2=0. При SM2=1 див. опис для біта SM2.

**Буфер передавача** призначений для приймання із шини ППП паралельної інформації і видачі її у вигляді послідовного потоку символів на передавач послідовного порту.

**Буфер приймача** призначений для приймання даних у вигляді послідовного потоку символів з послідовного порту, перетворення їх у паралельний вид, зберігання і видачі в паралельному виді на внутрішню шину ППП.

Буфер приймача і буфер передавача при програмному доступі мають однакове ім'я (SBUF) і адреса (99H). Якщо команда використовує SBUF як регістр джерела, то обіг відбувається до буфера приймача. Якщо команда використовує SBUF як регістр призначення, то обіг відбувається до буфера передавача.

У всіх чотирьох режимах роботи послідовного порту передача ініціюється будь-якою командою, яка використовує SBUF як регістр призначення. Приймання в режимі 0 ініціюється умовою Rl=0 і REN=1. В інших режимах приймання ініціюється приходом старт-біта, якщо REN=1.

**Приймач/передавач** послідовного порту призначений для приймання послідовного потоку символів із входу послідовного порту, виділення даних і видачі їх у буфер приймача, а також для приймання послідовних даних з буфера передавача, перетворення їх у послідовний потік символів і видачі його на вихід послідовного порту.

**Регістр пріоритетів переривань (IP)** призначений для установки рівня пріоритету переривання для кожного з п'яти джерел переривань. Позначення розрядів регістру IP показано в табл. А.11, а їх призначення зазначене нижче:

-PX0 - установка рівня пріоритету переривання від зовнішнього джерела $\overline{INT}0$;

-PT0 - установка рівня пріоритету переривання від Т/Л0.

-PX1 - установка рівня пріоритету переривання від зовнішнього джерела $\overline{INT}1$;

-PT1 - установка рівня пріоритету переривання від Т/Л1.

-PS - установка рівня пріоритету переривання від послідовного порту;

X - резервний розряд.

Таблиця А.11 - Регістр пріоритетів переривань IP

| Біти | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Позначення | X | X | X | PS | PT1 | PX1 | PT0 | PX0 |

Наявність у розряді IP "1" встановлює для відповідного джерела високий рівень пріоритету, а наявність у розряді IP "0" — низький рівень пріоритету. При читанні резервних розрядів відповідні лінії магістралі даних не визначені. Користувач не повинен записувати "1" у резервні розряди, тому що вони зарезервовані під подальше розширення сімейства МК51.

**Регістр дозволу переривань (IE)** призначений для дозволу або заборони переривань від відповідних джерел. Позначення розрядів регістру IE показане, у табл.А.12, а їх призначення зазначене нижче.

X - резервний розряд.

EA - керування всіма джерелами переривань одночасно. Якщо EA=0, то переривання заборонені. Якщо EA=1,то переривання можуть бути дозволені індивідуальними дозволами EX0, ET0, EX1, ET1, ES.

ES - керування перериванням від послідовного порту. ES=1 - дозвіл. ES=0 - заборона.

ET1 - керування перериванням від Т/Л1. ET1=1 - дозвіл. ET1=0 - заборона.

EX1 - керування перериванням від зовнішнього джерела $\overline{INT}$ 1. EX1=1 - дозвіл. EX1=0 - заборона.

ET0 - керування перериванням від Т/Л0. ET0=1 - дозвіл. ET0=0 - заборона.

EX0 - керування перериванням від зовнішнього джерела $\overline{INT}$ 0. EX0=1 - дозвіл. EX0=0 - заборона.

При читанні резервних розрядів відповідні лінії магістралі не визначені. Користувач не повинен записувати "1" у резервні розряди - вони зарезервовані під подальше розширення сімейства МК51.

Таблиця А.12 - Регістр дозволу переривань IE

| Біти | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Позначення | EA | X | X | ES | ET1 | EX1 | ET0 | EX0 |

**Логіка обробки прапорів переривань** здійснює пріоритетний вибір запиту переривання, скидання його прапора і ініціює виробіток апаратно реалізованої команди переходу на підпрограму обслуговування переривання.

**Схема виробітку вектора переривання** виробляє двохбайтові адреси підпрограм обслуговування переривання залежно від джерела переривань, які наведено в табл.А.13. Докладно система переривань описана в розділі 2.3.4.

Таблиця А.13 - Двохбайтові адреси підпрограм обслуговування переривання

| Джерело переривання | Вектор переривання |
|---|---|
| Зовнішнє переривання $\overline{INT}$ 0 | 0003H |
| Таймер/лічильник Т/Л0 | 000BH |
| Зовнішнє переривання $\overline{INT}$ 1 | 0013H |
| Таймер/лічильник Т/Л1 | 001BH |
| Послідовний порт | 0023H |

### А.7 Порти

Порти Р0, Р1, Р2, Р3 є двонаправленими портами введення-виводу і призначені для забезпечення обміну інформацією МК із зовнішніми пристроями, утворюючи 32 лінії вводу-виводу. Кожний з портів містить фіксатор-засувку, який являє собою восьмирозрядный регістр, що має байтовую і бітову адресацію для установки (скидання) розрядів за допомогою програмного забезпечення.

Фізичні адреси фіксаторів Р0, Р1, Р2, Р3 становлять для:

- Р0 — 80Н, при бітовій адресації 80 Н-Н-87Н;
- Р1 — 90Н, при бітовій адресації 90 Н-Н-97Н;
- Р2 — А0Н, при бітовій адресації А0 Н-А7Н;
- Р3 — В0Н, при бітовій адресації В0 Н-В7Н.

Крім роботи як звичайних портів введення/виводу лінії портів Р0-Р3 можуть виконувати ряд додаткових функцій, описаних нижче.

**Через порт Р0:** виводиться молодший байт адреси А0-А7 при роботі із зовнішньою пам'яттю програм і зовнішньою пам'яттю даних. Видається з МК і ухвалюється в МК байт даних при роботі із зовнішньою пам'яттю (при цьому обмін байтом даних і вивід молодшого байта адреси зовнішньої пам'яті мультиплексовані в часі);

Задаються дані при програмуванні внутрішнього ППЗУ і читається вміст внутрішньої пам'яті програм.

**Через порт Р1:** задається молодший байт адреси при програмуванні внутрішнього ППЗУ і при читанні внутрішньої пам'яті програм.

**Через порт Р2:** виводиться старший байт адреси А8-А15 при роботі із зовнішньою пам'яттю програм і зовнішньою пам'яттю даних (для зовнішньої пам'яті даних — тільки при використанні команд MOVX A, @DPTR і MOVX @DPTR, A, які виробляють 16-у адресу);

Задається старший байт (розряди А8-А14) адреси при програмуванні внутрішнього ППЗП і при читанні внутрішньої пам'яті програм.

**Кожна лінія порту Р3 має індивідуальну альтернативну функцію:**

-Р3.0 - Rxd, вхід послідовного порту, призначений для введення послідовних даних у приймач послідовного порту;

-Р3. 1 — Txd, вихід послідовного порту, призначений для виводу послідовних даних з передавача послідовного порту;

-Р3. 2 - $\overline{INT}$ 0, використовується як вхід 0 зовнішнього запиту переривання;

-Р3.3 - $\overline{INT}$ 1, використовується як вхід 1 зовнішнього запиту переривання;

-Р3.4 - Т0, використовується як вхід лічильника зовнішніх подій Т/Л0;

-Р3.5 - Т1, використовується як вхід лічильника зовнішніх подій Т/Л1;

-Р3.6 -$\overline{WR}$ , строб запису в зовнішню пам'ять даних, вихідний сигнал, що супроводжує вивід даних через порт Р0 при використанні команд MOVX @Ri, A і MOVX @DPTR, A.

-Р3.7 - $\overline{RD}$ , строб читання із зовнішньої пам'яті даних, вихідний сигнал, що супроводжує введення даних через порт Р0 при використанні команд MOVX A, @Ri і MOVX A, @DPTR.

Альтернативна функція кожної з ліній порту Р3 реалізується тільки в тому випадку, якщо у відповідному цій лінії розряді фіксатора-засувки втримується "1". А якщо ні, то на лінії порту Р3 буде присутній "0".

### А.8 Пам'ять даних

Пам'ять даних призначена для приймання, зберігання і видачі інформації, використовуваної в процесі виконання програми. Пам'ять даних, розташована на кристалі МК, складається з регістру адреси ОЗУ, дешифратора, ОЗП, покажчика стека.

Регістр адреси ОЗП призначений для приймання і зберігання адреси обираної за допомогою дешифратора комірки пам'яті, яка може містити як біт, так і байт інформації. ОЗУ являє собою 128-м 8-ми розрядних регістрів, призначених для прийому, зберігання і видачі D.

Покажчик стека являє собою 8-ми розрядний регістр, призначений для приймання і зберігання адреси гнізда стека, до якої був останній обіг. При виконанні команд LCALL, ACALL вміст покажчика стека збільшується на 2. При виконанні команд RET, RETI вміст покажчика стека зменшується на 2. При виконанні команди PUSH direct уміст покажчика стека збільшується на 1. При виконанні команди POP direct уміст покажчика стека зменшується на 1. Після скидання в покажчику стека встановлюється адресу 07Н, що відповідає початку стека з адресою 08Н.

В МК передбачена можливість розширення пам'яті даних шляхом підключення зовнішніх пристроїв ємністю до 64 Кбайт. При цьому звертання до зовнішньої пам'яті даних можливо тільки за допомогою команд MOVX.

Команди MOVX @Ri, A і MOVX A, @Ri формують восьмирозрядну адресу, видавану через порт P0. Команди MOVX @DPTR, A и MOVX @A, DPTR формують 16-розрядну адресу, молодший байт які видається через порт P0, а старший - через порт P2.

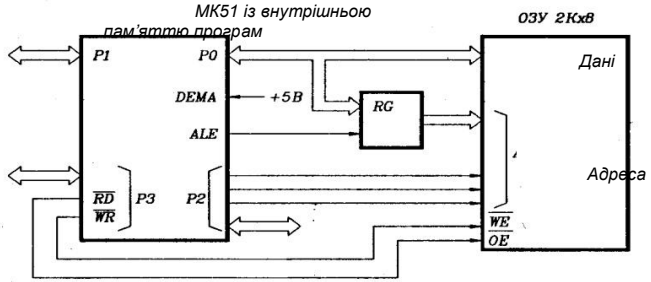



Рисунок А.7 - Сторінкова організація зовнішньої пам'яті даних

Байт адреси, видаваний через порт P0, повинен бути зафіксований у зовнішньому регістрі по спаду сигнал АЛП ALE, тому що надалі лінії порту P0 використовуються як шина даних, через яку байт даних ухвалюється з пам'яті при читанні або видається на згадку даних при записі. При цьому читання стробується сигналом МК $\overline{RD}$, а запис — сигналом МК $\overline{WR}$. При роботі із внутрішньою пам'яттю даних, сигнали $\overline{RD}$ і $\overline{WR}$ не формуються.

На рис. А.7 показана сторінкова організація зовнішньої пам'яті даних. Наведена схема дозволяє працювати з пам'яттю даних ємністю 2 Кбайт, використовуючи команди типу MOVX @Ri. Порт P0 при цьому працює як мультиплексована шина адресу/даних, а три лінії порту P2 адресують сторінки зовнішнього ОЗУ. Інші 5 ліній порту P2 можуть використовуватися в якості ліній введення/виводу. Докладно організація пам'яті даних описано в розділі А.9.

На рис. А.8 відповідно наведені діаграми циклів читання і запису при роботі МК із зовнішньою пам'яттю даних (Z — високоімпедансний стан).

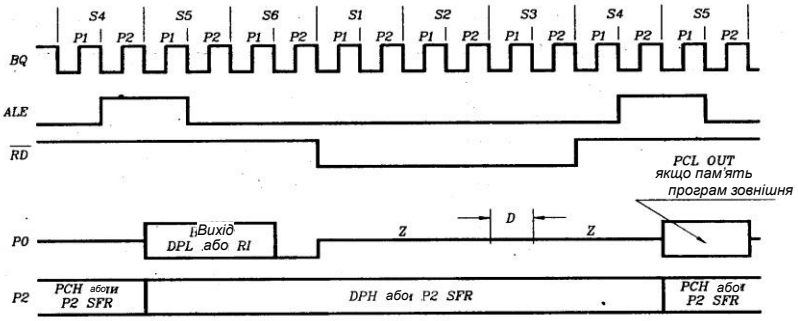



Рисунок А.8 - Цикл читання із зовнішньої пам'яті даних

PCL OUT - видача молодшого байта лічильника команд PC;

PCH - старший байт лічильника команд PC;

DPL, DPH - відповідно молодший і старший байти регістру покажчика даних DPTR, який використовується як регістр непрямої адреси в командах MOVX A, @DPTR і MOVX @DPTR, A;

P2 SFR - засувки порту P2;

RI - регістри P0 і R1, які використовуються в якості регістрів непрямої адреси в командах MOVX A, @RI і MOVX @P1, A;

D - період, протягом якого дані з P0 уводяться в МК.

### А.9 Пам'ять програм

Пам'ять програм призначена для зберігання програм і має окреме від пам'яті даних адресний простір обсягом до 64 Кбайт, причому, для мікросхем КР1816ВЕ51, КМ1816ВЕ751 і для КР1830ВЕ51 частина пам'яті програм з адресами 0000 Н-Н—0FFFH розташована на кристалі МК. Пам'ять програм, розташована на кристалі, складається з 12-розрядного дешифратора і ПЗП ємністю 4 до *8 біт для мікросхем КР1816ВЕ51, КР183ОВЕ51 або ППЗП з ультрафіолетовим стиранням ємністю 4К**х**8 біт для КМ1816ВЕ751.

Запис програм у ПЗП відбувається під час виготовлення кристалів.

Якщо на вивід МК DEMA подана напруга живлення UCC, то звертання до зовнішньої пам'яті програм відбувається автоматично при виробітку лічильником команд адреси, що перевищує 0FFFH. Якщо адреса перебуває в межах 0000 H-H-0FFFH, обіг відбувається до пам'яті програм, розташованої на кристалі (внутрішньої пам'яті програм).

Якщо на вивід МК DEMA поданий "0", внутрішня пам'ять програм відключається і починаючи з адреси 0000H усі обіги виконуються до зовнішньої пам'яті програм.

PCL OUT - видача молодшого байта лічильника команд PC;

PCH - старший байт лічильника команд PC;

DPL, DPH - відповідно молодший і старший байти регістру покажчика даних DPTR, який використовується як регістр непрямої адреси в командах MOVX A, @DPTR і MOVX @DPTR.A;

P2 SFR - засувки порту P2;

Ri - регістри R0 і R1, які використовуються в якості регістрів непрямої адреси в командах MOVX A, @Ri і MOVX @Ri, A.

Якщо МК не має внутрішньої пам'яті програм, її вивід DEMA повинен бути підключений до шини ОВ.

Читання із зовнішньої пам'яті програм стробується сигналом МК $\overline{PME}$. При роботі із внутрішньою пам'яттю програм сигнал не формується. МК не мають інструкцій і апаратних засобів для програмного запису на згадку програм.

При звертаннях до зовнішньої пам'яті програм завжди формується 16-розрядна адреса, молодший байт якого видається через порт P0, а старший — через порт P2. При цьому байт адреси, видаваний через порт P0, повинен бути зафіксований у зовнішньому регістрі по спаду сигналу ALE, тому що надалі лінії порту P0 використовуються в якості шини даних, по якій байт із зовнішньої пам'яті програм уводиться в МК.

На рис. А.9 показана функціональна схема включення МК51 із зовнішнім ППЗУ програм. Порт P0 працює як мультиплексована шина адресу/даних: видає молодший байт лічильника команд, а потім переходить у высокоимпедансное стан і очікує приходу байта із ППЗУ програм. Коли молодший байт адреси перебуває на виходах порту P0, сигнал ALE записує його в адресному регістрі RG. Старший байт адреси перебуває на виходах порту P2 протягом усього часу звертання

до ППЗУ. Сигнал дозволяє вибірку байта із ППЗУ, після чого обраний байт надходить на порт P0 МК51 і вводиться в МК.

На рис. А.10 і А.11 наведені діаграми, що показують формування відповідних сигналів при роботі МК із зовнішньою пам'яттю програм.

Як видно з діаграм, сигнал $\overline{PME}$ формується двічі в кожному машинному циклі незалежно від кількості байт у команді. Якщо другий обраний байт у поточній команді не використовується, він ігнорується МК. Далі при переході до виконання наступної команди цей байт буде введений вдруге.

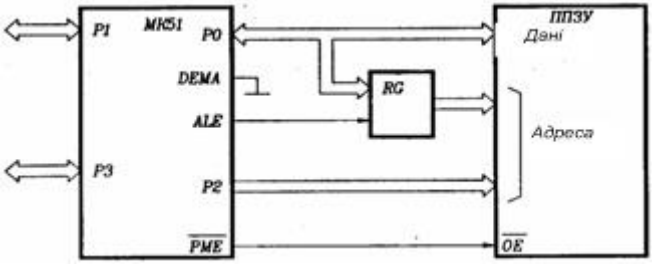



Рисунок А.9 - Схема включення МК51 із зовнішнім ППЗУ програм

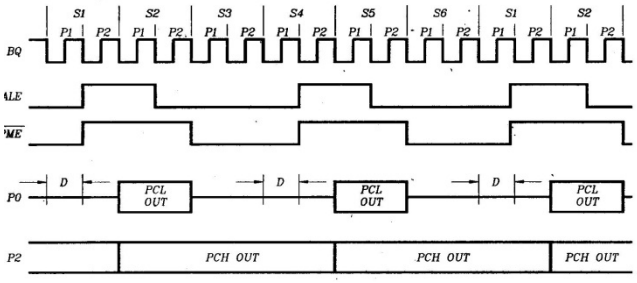



Рисунок А.10 - Робота із зовнішньою пам'яттю програм: PCL OUT - видача молодшого байту лічильника команд; PCH OUT - видача старшого байту лічильника команд; *D* - періоди, під час яких дані з P0 вводятся в ОМЕВМ.

Якщо виконується команда MOVX (рис. А.11, діаграма В), два сигнали $\overline{PME}$ не формуються, тому що порт P0 звільняється для адресації і обміну даними із зовнішньою пам'яттю даних.

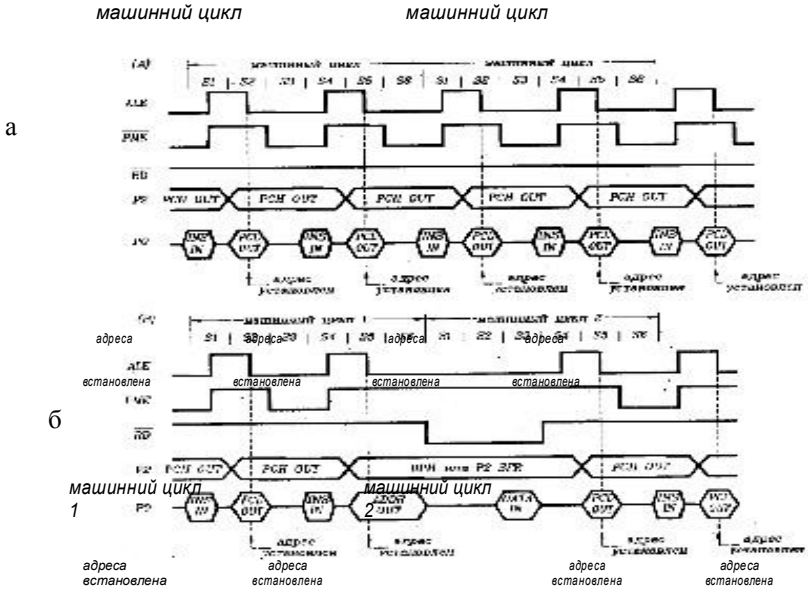



Рисунок А.11 - Цикли роботи із зовнішньою пам'яттю програм

Коли МК працює із внутрішньою пам'яттю програм, $\overline{PME}$ не формується і адреса на портах Р0 і Р2 не видається. Проте, сигнал ALE буде формуватися двічі в кожному машинному циклі завжди за винятком випадку команди MOVX (у цьому випадку один сигнал ALE пропускається). Таким чином, якщо не використовуються команди MOVX, сигнал ALE може бути задіяний у якості вихідного синхросигналу. На рис.А1.11 подані цикли роботи із зовнішньою пам'яттю програм:

(а) - без виконання команди MOVX;

(б) – з виконанням команди MOVX;

PCL OUT - видача молодшого байту лічильника команд PC;

PCH OUT - видача старшого байту лічильника команд PC; DPH - старший байт регістру покажчика данних DPTR, який використовується у якості регістру непрямого адресу в командах MOVX A, @DPTR і MOVX @DPTR, A;

P2 SFR - засувки порту P2;

INS IN- ввод байту інструкції з пам’яті програм;

ADDR OUT - видача молодшого байту адреси зовнішньої пам'яті даних з регістрів R0, R1 або регістру DPL. Задній фронт ALE стробує адресу в порті P0. В цей час адреса гарантовано встановлена.

### А.10 Блок керування. Синхронізація МК. Регістр PCON. Режими зменшеного енергоспоживання

Блок керування призначений для виробітку синхронізуючих і керуючих сигналів, що забезпечують координацію спільної роботи блоків МК у всіх припустимих режимах його роботи.

До складу блоку входять: пристрій виробітку тимчасових інтервалів, логіка введення-виводу, регістр команд, регістр керування споживанням, дешифратор команд, ПЛМ і логіка керування ЕОМ.

**Пристрій виробітку часових інтервалів** призначений для формування і видачі внутрішніх синхросигналів фаз, тактів і циклів. Кількість машинних циклів визначає тривалість виконання команд. Практично всі команди МК виконуються за один або два машинні цикли, крім команд множення MUL A, У і розподілу DIV A, B, тривалість виконання яких становить чотири машинні цикли. Машинний цикл має фіксовану тривалість і містить шість станів S1-S6, кожне з яких по тривалості відповідає такту, і, у свою чергу, складається із двох часових інтервалів, обумовлених фазами P1 і P2. Тривалість фази дорівнює періоду проходження зовнішнього сигналу BQ, що є первинним сигналом синхронізації МК. Сигнал BQ виробляється або вбудованим тактовим генератором МК при підключенні до її виводів 18 (BQ2) і 19 (BQ1) кварцового резонатора або LC-ланцюжка, або зовнішнім джерелом тактових сигналів.

Схема підключення кварцового резонатора і LC-Ланцюжка до виводів МК BQ2 і BQ1 показана на рис. А.12. Схема підключення зовнішнього джерела тактових сигналів показана на рис. А.13. Джерело тактових сигналів повинен забезпечувати наступні характеристики зовнішнього синхросигналу МК:

- тривалістьнизького рівня сигнал АЛП - не менше 20 нс;
- тривалість високого рівня сигнал АЛП - не менше 20 нс;
- часи фронтів наростання і спаду сигнал АЛП - не більше 20 нс.

Як видно з рис. А.13, зовнішній синхросигнал для n-МОН МК (серія 1816) подається на вихід 18 (BQ2), а для КМОН МК (серія 1830) — на вихід 19 (BQ1). При цьому необхідно забезпечувати необхідні рівні напруги синхросигналу.

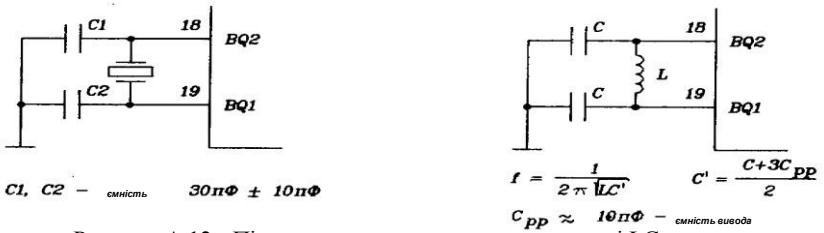



Рисунок А.12 - Підключення кварцового резонатора і LC-ланцюжка

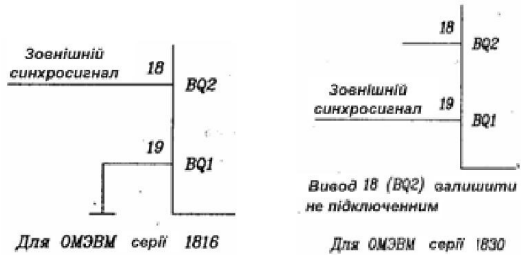



Рисунок А.13 - Підключення зовнішнього джерела тактових сигналів

На рис. А.14 показані в загальному виді внутрішні генератори n-моп і КМОН МК сімейства МК51. BQ1 і BQ2 є відповідно входом і виходом підсилювача, що інвертує, який може бути включений у режим генератора (при підключенні до виводів BQ1 і BQ2 резонатора або LC-ланцюжка рис. А.12), при чому сигнал PD задається установкою однойменного біта в регістрі PCON для перекладу МК у режим мікроспоживання (рис. А.14).

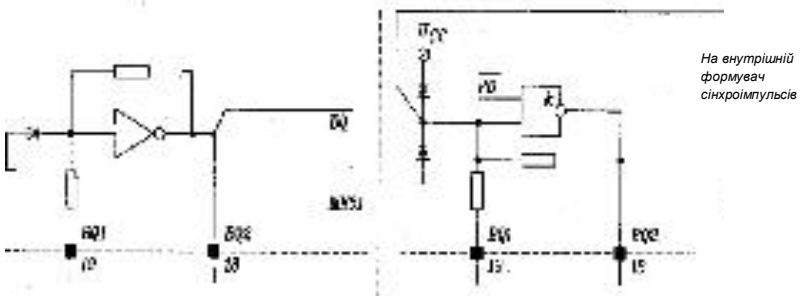



Рисунок А.14 - Внутрішні генератори МК51

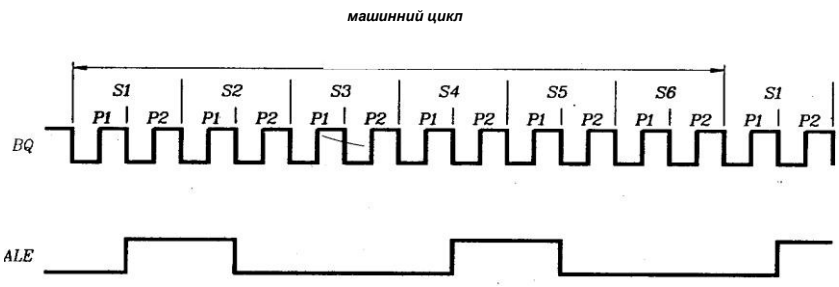



Рисунок А.15 - Діаграма формування машинних циклів МК51

Рис. А.15 ілюструє формування машинних циклів МК. Усі машинні цикли МК однакові, складаються з 12 періодів сигналу BQ, починаються фазою S1P1 і закінчуються фазою S6P2. Двічі за один машинний цикл формується сигнал ALE. На рис. А.16 показана діаграма зовнішнього синхросигналу МК.

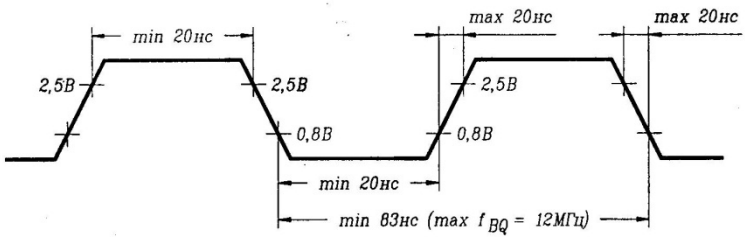



Рисунок А.16 - Діаграма зовнішнього синхросигналу МК

**Логіка введення-виводу** призначена для приймання і видачі сигналів, що забезпечують обмін інформацією МК із зовнішніми пристроями через порти введення-виводу Р0-Р3.

**Регістр команд** призначений для запису і зберігання 8-ми розрядного коду операції виконуваної команди, який за допомогою дешифратора команд перетворюється в 24-х розрядний код для ПЛМ, за допомогою якої виробляється набір мікрооперацій відповідно до мікропрограми виконання команди. Регістр команд програмно не доступний.

**Конструкція регістру керування споживанням (PCON)** визначається технологією виготовлення МК: n-МОН або КМОН. Усі біти регістру PCON програмно доступні по запису ("0","1") і читанню.

Для варіанта виготовлення за технологією n- МОН (серія 1816) регістр PCON має всього 1 біт, керуючий швидкістю передачі послідовного порту SMOD.

Для варіанта виготовлення за технологією К-МОН (серія 1830) позначення розрядів регістру PCON наведено в таблиці А.14, а призначення розрядів у таблиці А.15. Для n-МОН і К-МОН МК розташування і призначення розряду SMOD ідентичні.

**Функції біта SMOD** докладно розглянуті при описі роботи послідовного порту.

Таблиця А.14 - Регістр **PCON** серії 1830 (КМОП)

| Біти | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Позначення | SMOD | - | - | - | GF1 | GF0 | PD | IDL |

Біти PCON з номерами 4…6 зарезервовані для подальшого розширення сімейства МК51. При читанні значення цих розрядів не визначене. Програміст не повинен записувати "1" у ці біти, тому що вони можуть використовуватися в майбутніх розробках МК сімейства МК51 для завдання нових функцій. У цьому випадку пасивне значення бітів 4…6 буде "0", а активне - "1". Біти GF1 і GF0 користувач може задіяти за своїм розсудом. В МК сімейства МК51, виконаних по КМОН технології, є два режими зменшеного енергоспоживання: режим холостого ходу і режим мікроспоживання. Джерелом живлення в цих режимах є вивід Ucc.

Режими зменшеного споживання для КМОП МК ініціюються установкою бітів PD і IDL у регістрі PCON. Вплив цих бітів на апаратуру МК ілюстровано на рис. А.17.

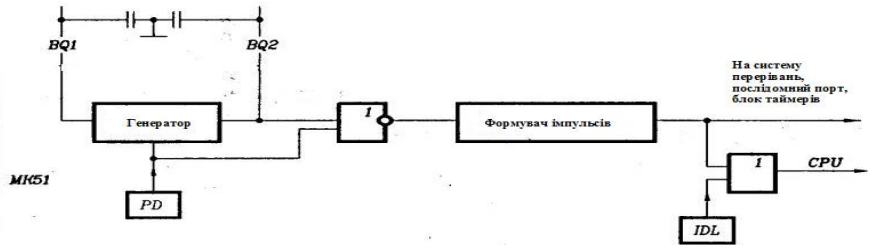



Рисунок А.17 - Дія бітів PD і IDL регістру PCON

Таблиця А.15 - Регістр **PCON** серії 1830 (КМОП)

| Біти | Наймен.. | Призначення бітів | Примітка |
|---|---|---|---|
| 7 | SMOD | Біт подвоєння швидкості передачі: при установці в "1"- швидкість передачі подвоюється | При роботі послідовного порту |
| 6 | - | Резервний | |
| 5 | - | Резервний | |
| 4 | - | Резервний | |
| 3 | GF1 | Прапор загального призначення | |
| 2 | GF0 | Прапор загального призначення | |
| 1 | PD | Біт включення режиму мікроспоживання "1" - режим мікроспоживання | Якщо в PD і IDL одночасно записана "1", перевага має PD |
| 0 | IDL | Біт холостого ходу "1"- режим холостого ходу | |

**Режим холостого ходу:** інструкція, яка встановлює PCON.0=1 (IDL), є останньою інструкцією, виконуваної перед переходом у режим холостого ходу. У цьому режимі блокуються функціональні вузли центрального процесора (CPU), що і зменшує енергоспоживання. Зберігаються стани покажчика стека, програмного лічильника, PSW, акумулятора і усіх інших регістрів, а також внутрішнього ОЗП даних.

Для закінчення режиму холостого ходу є два способи. Активізація будь-якого дозволеного переривання автоматично приведе до установки PCON. 0=0, закінчуючи режим холостого ходу. Після виконання команди RETI (вихід з підпрограми обслуговування переривання) буде виконана команда, яка іде за командою, яка переводе МК у режим холостого ходу.

Біти GF0 і GF1 зручно використовувати для індикації режиму, у якому була викликана програма обробки переривання: відбулося це при нормальній роботі МК або в режимі холостого ходу. Приміром, команда, що викликає режим холостого ходу, може також установлювати один або кілька прапорів (GF0, GF1 або яких-небудь інших). Програма обробки переривання, перевіряючи ці прапори, може визначити передісторію свого виклику.

Іншим способом закінчення режиму холостого ходу є апаратне скидання по входу RST тривалістю не менш двох машинних циклів.

Активний сигнал скидання на виводі RST асинхронно скидає біт IDL (PCON.0). Оскільки тактовий генератор працює, МК відразу після скидання IDL починає виконувати програму з команди, що випливає за командою, що викликала режим холостого ходу. Між скиданням біта IDL і моментом, коли ввімкнеться внутрішній алгоритм скидання, може пройти до двох машинних циклів виконання програми. Внутрішні апаратні засоби МК блокують доступ до внутрішньої пам'яті даних протягом зазначеного часу, але не блокують доступ до портів. Якщо при цьому зміна інформації на портах небажана, то необхідно стежити, щоб за командою, яка встановлює біт IDL, не випливала безпосередньо команда, що записує інформацію в порт або в зовнішню пам'ять даних.

**Режим мікроспоживання:** інструкція, яка встановлює PCON.1=1 (PD), є останньою виконуваною командою перед переходом у режим мікроспоживання. У цьому режимі генератор, що задає **f**, вимикається, припиняючи тим самим роботу всіх вузлів МК і зберігається тільки вміст ОЗП.

Єдиним виходом із цього стану є апаратне скидання RST. У цьому режимі роботи напруга Ucc може бути зменшена до 2 У і повинна бути відновлена до номінального перед виходом з режиму мікроспоживання.

Скидання слід утримувати в активному стані не менш 10 мс (час відновлення роботи задаючого генератора). При записі IDL=1 і PD=1 перевага має біт PD.

У таблиці А.16 представлені стани виводів МК у режимах холостого ходу і мікроспоживання.

Таблиця А.16 - Стану виводів МК у режимах холостого ходу і мікроспоживання

| Режим | Пам'ять програм | ALE | $\overline{PME}$ | Порт P0 | Порт P1 | Порт P2 | Порт P3 |
|---|---|---|---|---|---|---|---|
| Холостого ходу | Внутрішня | 1 | 1 | Дані | Дані | Дані | Дані |
| Холостого ходу | Зовнішня | 1 | 1 | Z | Дані | Адреса | Дані |
| Мікроспоживання | Внутрішня | 0 | 0 | Дані | Дані | Дані | Дані |
| Мікроспоживання | Зовнішня | 0 | 0 | Z | Дані | Дані | Дані |

**Режим зниженого споживання для МК серії 1816 ( n-МОН)**

Під час нормальної роботи внутрішня ОЗП живиться від Ucc. Однак для МК серії 1816 сімейства МК51, якщо напруга на виводі RST перевищує Ucc, воно стає джерелом живлення для ОЗП. Це реалізовано за допомогою двох внутрішніх діодів, з катодів яких береться живлення ОЗП, а аноди підключені відповідно до входу RST

і до виводу живлення МК $U_{cc}$. Підкреслимо, що при К-МОН МК даний режим відсутній.

**Логіка керування ЕОМ** залежно від режиму роботи МК виробляє необхідний набір керуючих сигналів.

### А.11 Система команд МК51

Мікроконтролер має 111 базових команд передачі даних, арифметичних і побітних операцій, передачі керування і операцій з бітами. Більшість команд виконуються за 1 або 2 машинних циклів при $f_{так}$=12МГц і тривалості машинного циклу 1 мкс. Перший байт команди завжди містить код операції, другий і третій байти або адреси операндів, або безпосередні операнди. Перелік команд МК51 приводится в таблиці А.18.

Таблиця А.18 - Система команд

| Мнемокод | КОП | Мнемокод | КОП | Мнемокод | КОП |
|---|---|---|---|---|---|
| ACALL 0xxH | 11 | AJMP 5XXH | A1 | DA A | D4 |
| ACALL 1xxH | 31 | AJMP 6XXH | C1 | DEC A | 14 |
| ACALL 2xxH | 51 | AJMP 7XXH | El | DEC ad | 15 |
| ACALL 3xxH | 71 | ANL A, ad | 55 | DEC R0 | 18 |
| ACALL 4xxH | 91 | ANL A, R0 | 58 | DEC R1 | 19 |
| ACALL 5xxH | B1 | ANL A, R1 | 59 | DEC R2 | 1A |
| ACAlL 6xxH | D1 | ANL A, R2 | 5A | DEC R3 | 1B |
| ACALL 7xxH | FI | ANL A, R3 | SB | DEC R4 | 1C |
| ADD A, ad | 25 | ANL A, R4 | 5C | DEC R5 | 1D |
| ADD A, R0 | 28 | ANL A, R5 | 5D | DEC R6 | 1E |
| ADD A , R1 | 29 | ANL A, R6 | 5E | DEC R7 | 1F |
| ADD A, R2 | 2A | ANL A, R7 | 5F | DEC @R0 | 16 |
| ADD A, R3 | 2B | ANL A, @R0 | 56 | DEC @R1 | 17 |

| | | | | | |
|---|---|---|---|---|---|
| ADD A, R4 | 2C | ANL A, @R1 | 57 | DIV AB | 84 |
| ADD A, R5 | 2D | ANL A, #d | 54 | DJNZ ad, rel | D5 |
| ADD A, R6 | 2E | ANL ad, A | 52 | DJNZ R0, rel | D8 |
| ADDA, R7 | 2F | ANL ad, #d | S3 | DJNZ R1, rel | D9 |
| ADD A, @R0 | 26 | ANL C, bit | 82 | DJNZ R2, rel | DA |
| ADD A, @R1 | 27 | ANL C, /bit | BO | DJNZ R3, rel | DB |
| ADD A, #d | 34 | CJNE A, ad, | B5 | DJNZ R4, rel | DC |
| ADDC A, ad | 35 | CJNE A, #d, | B4 | DJNZ R5, rel | DD |
| ADDC A, R0 | 38 | CJNE R0, #d, | B8 | DJNZ R6, rel | DE |
| ADDC A, R0 | 39 | CJNE R1, #d, | B9 | DJNZ R7, rel | DF |
| ADDC A, RO | 3A | CJNE R2, #d, | BA | INC a | 04 |
| ADDC A , R0 | 3B | CJNE R3, #d, | BB | INC ad | 05 |
| ADDC A, R0 | 3C | CJNE R4, #d, | BC | INC DPTR | A3 |
| ADDC A, R0 | 3D | CJNE R5, #d, | BD | INC R0 | 08 |
| ADDC A, R0 | 3E | CJNE R6, #d , | BE | INC R1 | 09 |
| ADDC A , R0 | 3F | CJNE R7 , #d, | BF | INC R2 | OA |
| ADDC A, @R0 | 36 | CJNE @R0, | B6 | INC R3 | OB |
| ADDC A, @R1 | 37 | CJNE @R1, | B7 | INCR4 | 0C |
| ADDC A, #d | 24 | CLR A | E4 | INC R5 | 0D |
| AJMP 0XXH | 01 | CLR bit | C2 | INC R6 | 0E |
| AJMP 1XXH | 21 | CLR C | C3 | INC R7 | OF |
| AJMP 2XXH | 41 | CPL A | F4 | INC @R0 | 06 |
| AJMP 3XXH | 61 | CPL bit | B2 | INC @R1 | |
| AJMP 4XXH | 81 | CPL C | B3 | JB bit, rel | |
| | | | | JBC bit, rel | |
| JC rel | 40 | MOV ad , @R0 | 86 | MOV R7 , ad | AF |
| JMP@A+DPTR | 73 | MOV ad, @R1 | 87 | MOV R7, #d | 7F |
| JNB bit , rel | 30 | MOV ad, #d | 75 | MOV @R0 , A | F6 |
| JNC rel | 50 | MOV ad, ads | 85 | MOV@R0, ad | A6 |
| JNZrel | 70 | MOV bit, C | 92 | MOV@R0, #d | 76 |
| JZ rel | 60 | MOV C, bit | A2 | MOV @R1 , A | F7 |
| LCALL ad16 | 12 | MOV DPTR, | 90 | MOV@R1, ad | A7 |
| LJMP ad 16 | 02 | MOV R0, A | F8 | MOV @R1, #d | 77 |
| MOV A , ad | E5 | MOV R0, ad | A8 | MOVC A, @+DPTR | 93 |

| | | | | | |
|---|---|---|---|---|---|
| MOV A , RO | E8 | MOV R0, #d | 78 | MOVC A , @+PC | 83 |
| MOV A, R1 | E9 | MOV R1 , A | F9 | MOVX A , @DPTR | EO |
| MOV A , R2 | EA | MOV R1 , ad | A9 | MOVX A, @R0 | E2 |
| MOV A , R3 | EB | MOV Rl , #d | 79 | MOVX A, @R1 | E3 |
| MOV A , R4 | EC | MOV R2, A | FA | MOVX @DPTR, A | F0 |
| MOV A , R5 | ED | MOV R2, ad | AA | MOVX @R0 , A | F2 |
| MOV A , R6 | EE | MOV R2, #d | 7A | MOVX @R1, A | F3 |
| MOV A , R7 | EF | MOV R3 , A | FB | MUL AB | A4 |
| MOV A , @R0 | E6 | MOV R3 , ad | AB | NOP | 00 |
| MOV A, @R1 | E7 | MOV R3 , #d | 7B | ORL A , ad | 45 |
| MOV a , #d | 74 | MOV R4, A | FC | ORL A , R0 | 48 |
| MOV ad , A | F5 | MOV R4 , ad | AC | ORL A, R1 | 49 |
| MOV ad , R0 | 88 | MOV R4, #d | 7C | ORL A, R2 | 4A |
| MOV ad , R1 | 89 | MOV R5, A | FD | ORL A , R3 | 4B |
| MOV ad , R2 | 8A | MOV R5 , ad | AD | ORL A, R4 | 4C |
| MOV ad, R3 | 8B | MOV R5 , #d | 7D | ORL A, R5 | 4D |
| MOV ad , R4 | 8C | MOV R6 , A | FE | ORL A, R6 | 4E |
| MOV ad , R5 | 8D | MOV R6, ad | AE | ORL A, R7 | 4F |
| MOV ad , R6 | 8E | MOV R6, #d | 7E | ORL A, @R0 | 46 |
| MOV ad , R7 | 8F | MOV R7 , A | FF | ORL A , @R0 | 47 |
| ORL A, #d | 44 | RRC A | 13 | SUBB A , R7 | 9F |
| ORL ad , A | 42 | SETB bit | D2 | SUBB A , @R0 | |
| ORL ad , #d | 43 | setb c | D3 | SUBB A, @R1 | 97 |
| ORL C , bit | 72 | SJMP rel | 80 | SWAP A | C4 |
| ORL C, /bit | AO | SUBB A, ad | 95 | XCH A , ad | C5 |
| POP ad | DO | SUBB A, R0 | 98 | XCH A, R0 | C8 |
| PUSH ad | CO | SUBB A, R1 | 99 | XCH A, R1 | C9 |
| RET | 22 | SUBB A , R2 | 9A | XCH A , R2 | CA |
| RETI | 32 | SUBB A , R3 | 9B | XCH A , R3 | CB |
| RL A | 23 | SUBB A, R4 | 9C | XCH A , R4 | CC |
| RLC A | 33 | SUliB A , R5 | 9D | XCH A , R5 | CD |
| RR A | 03 | SUBB A, R6 | 9E | XCH A , R6 | CE |
| XCH A, R7 | CF | XRL A, R1 | 69 | XRL A , R7 | 6F |
| XCH A, @R0 | 06 | XRL A, | 6A | XRL A , @)R0 | 66 |

| | | | | | |
|---|---|---|---|---|---|
| XCH A, @Rl | C7 | XRL A, R3 | 6B | XRL A, @R1 | 67 |
| XCHD A, @R0 | D6 | XRL A, R4 | 6C | XRL A, #d | 64 |
| XCHD A, @R1 | D7 | XRL A, R5 | 6D | XRL ad , A | 62 |
| XRL A, ad | 65 | XRL A , R6 | 6E | XRL ad, #d | 63 |
| XRL A, R0 | 68 | | | | |

## ЛІТЕРАТУРА


1. Шегал А. А. Применение программного комплекса Multisim для проектирования устройств на микроконтроллерах: лабораторный практикум / А. А. Шегал. - Екатеринбург : Изд-во Урал. ун-та, 2014. - 114с.
2. Каспер Э. Программирование на языке Ассемблера для микроконтроллеров семейства i8051 / Э. Каспер - М.: Горячая линия-Телеком, 2004. - 192 с.
3. Магда Ю. С. Микроконтроллеры серии 8051: практический подход / Ю. С. Магда - М.: ДМК Пресс, 2008. - 228 с.
4. Баррет С. Ф. Встраиваемые системы. Проектирование приложений на микроконтроллерах семейства 68HC12/HCS12 с применением языка С / С. Ф. Барретт, Д. Дж. Пак. М. : ДМК- пресс, 2007. 640 с.
5. Микушин А. В. Занимательно о микроконтроллерах / А. В. Микушин. С-Пб. : БХВ-Петербург, 2006. 432 с.
6. Хернитер М. Е. Multisim 7 Современная система компьютерного моделирования и анализа схем электронных устройств / М. Е. Хернитер – М.: «ДМК - Пресс», 2006. – 488 с.
7. Загидуллин Р. Ш. Multisim, LabView, Signal Express /Р. Ш. Загидуллин. М.: Горячая линия-Телеком, 2009. 366 с.
8. Марченко А. Л. Лабораторный практикум по электротехнике и электронике в среде Multisim: учебное пособие для вузов / А. М. Марченко, С. В. Освальд. М. : ДМК Пресс, 2010. 448 с.
9. Белов А. В. Конструирование устройств на микроконтроллерах / А. В. Белов. СПб. : Наука и Техника, 2005. 256 с.
10. Угрюмов Е. П. Цифровая схемотехника: учебное пособие / Е. П. Угрюмов. СПб. БХВ. С-Пб. : 2010. 528 с.
11. Волович Г. И. Схемотехника аналоговых и аналого-цифровых электронных устройств / Г. И. Волович. М. : Издательский дом «Додэка-XXI», 2005. 528 с.