НАВЧАЛЬНО-МЕТОДИЧНА ЛІТЕРАТУРА

Тернопільський національний технічний університет імені Івана Пулюя

Кафедра автоматизації технологічних процесів і виробництв

Лабораторна робота № 2 з курсу ”Мікропроцесорні та програмні засоби автоматизації”

Дослідження структури та програмування ОМЕОМ МК48 з використанням програмного емулятора Single-Chip Machine

Тернопіль 2018

Методичні вказівки до лабораторної роботи №2 “Дослідження структури та програмування ОМЕОМ МК48 з використанням програмного емулятора Single-Chip Machine” з курсу «Мікропроцесорні та програмні засоби автоматизації».. Медвідь В.Р., Пісьціо В.П., - Тернопіль: ТНТУ, 2018 - 21 с.

Відповідальні за випуск доцент, к.т.н. Медвідь В.Р.,
асистент Пісьціо В.П.

Для студентів напряму: 151 "Автоматизація та комп'ютерно-інтегровані технології"

Лабораторна робота №2

Дослідження структури та програмування ОМЕОМ МК48 з використанням програмного емулятора Single-Chip Machine

Мета роботи: дослідження структури та програмування ОМЕОМ МК48.

1. Структура МК48. Короткі теоретичні відомості

Однокристальна мікро-ЕОМ МК48 використовується як мікроконтролер і містить (рис. 1):

- центральний процесор **ЦП** (8 Біт);
- пам'ять програм **РПЗП** (ємністю 1 Кбайт);
- пам'ять даних **ОЗП** (64 Байти);
- багатоканальний інтерфейс вводу-виводу **P0, P1, P2** (27 ліній);
- програмований 8-бітний таймер-лічильник **ТЛ**;
- регістр адреси **РА**;
- пристрій управління та синхронізації;
- дешифратор команд **ДК**;
- регістр команд **РК.**

Передбачено розширення пам'яті РПЗП до 4 КБайт, ОЗП до 256 Байт, а також збільшення кількості ліній вводу-виводу за рахунок під'єднання зовнішніх ПЗП, ОЗП та інтерфейсів вводу-виводу.

Акумулятор (А) – 8-розрядний регістр для зберігання операнду та результату операції.

Регістр тимчасового зберігання (РТЗ) – призначений для прийому та тимчасового зберігання другого та третього байтів команд переходів, що передаються з внутрішньої магістралі даних в лічильник команд.

Схема десяткового коректора (СДК) .

Схема формування ознак (СФО) – формує ознаки, причина яких не фіксується в РССП (ознаки нульового вмісту А та ознака наявності 1 в селектованому розряді акумулятора).

Регістр словА-стану програми (РССП).

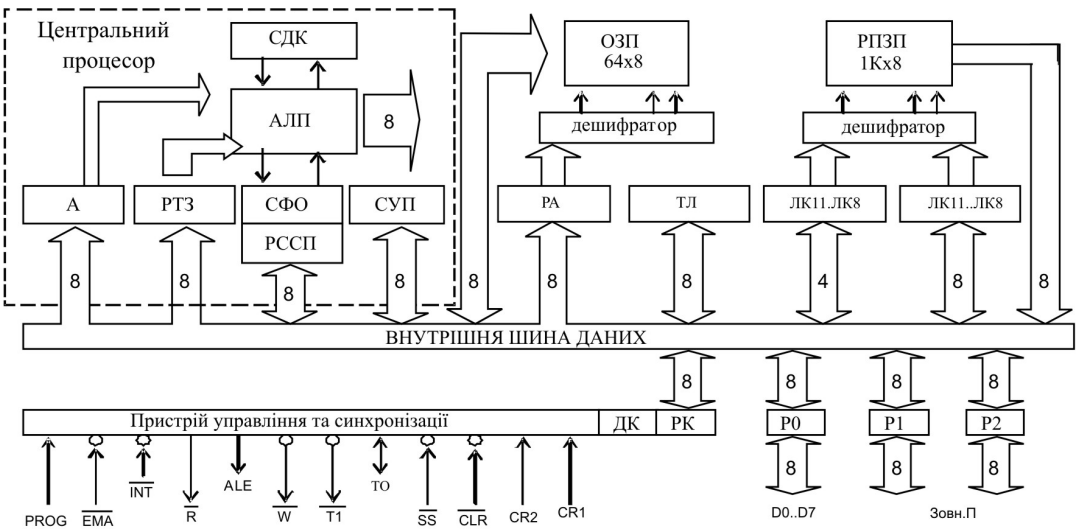

Рис. 1 Структурна схема

Регістр слова-стану програми (РССП)

D7	D6	D5	D4	D3	D2	D1	D0
C	**AC**	**F0**	**BS**	*	**S2**	**S1**	**S1**
Зберігання в стеку					Вказівник стеку		

За сигналом **"Запит переривання"** (**INT**) вміст РССП (D7..D4) завантажується в стек а при поверненні з програми переривання вміст цих розрядів відновлюється.

При поданні сигналу **"Скидання" (CLR)** управління передається адресі "0", за якою знаходиться команда безумовного переходу на початок програми.

Переключаються банки програмно.

Сигнал **“Фіксації адреси”** (**ALE**) заднім фронтом фіксує діючу адресу зовнішнього пристрою (Зовн.П). Сигнал **PME** (зчитування з зовнішньої пам’яті програм **ПЗП**) стробує вибірку байту з зовнішньої пам’яті програм на шину ШД.

ОЗП – пам’ять даних, служить для запиту, зберігання та зчитування даних в процесі обробки інформації.

64 (128) комірки ОЗП розбиті на два **банки BS0, BS1** регістрів загального призначення (РЗП) з адресами **00-7 та 24-31** по вісім регістрів в кожному (8х8, рис. 3).

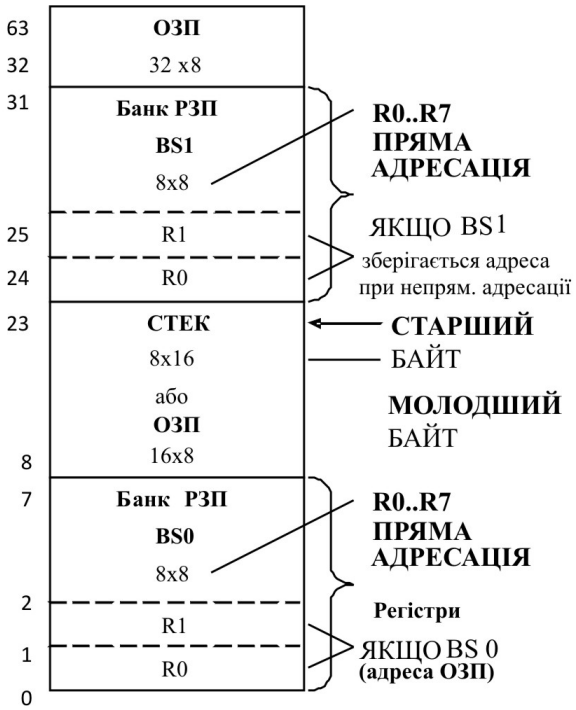

Рис**.** 2 Схема адресного простору ОЗП

Восьмирівневий 16-розрядний стек (адреси 8…23) та ОЗП (32..63), що використовується тільки як ОЗП даних.

Комірки ОЗП адресуються вказівником стеку (**S2, S1, S0**) з **РССП** (рис. 2). Обмін інформацією із зовнішною пам’ятю даних **ОЗП** стробується сигналом **R** (читання даних зовнішнього ОЗП) та **W** (запис даних в зовнішнє ОЗП).

2.3. Організація вводу-виводу

Багатоканальний інтерфейс вводу-виводу призначений для обміну інформацією ОМЕОМ з периферійними пристроями.

ОМЕОМ має 27 ліній вводу-виводу, 24 з яких об’єднані в три 8-розрядних канали **P2, P1, P0** (рис. 3).

Канали **P1, P2** мають можливість **фіксації даних**. Ці дані присутні на виводах каналу і можуть бути змінені тільки новою командою. P1, P2 можуть працювати на ввід або на роботу з двонаправленою лінією передачі.

Команда* Ошибка! Ошибка внедренного объекта. *встановлює канал в стан високого рівня, що дозволяє роботу на прийом даних.

Особливість каналів: при вводі інформації виконується операція **“І”** над даними, що вводяться, і даними, що були присутні перед цим. Тому необхідно, щоб **до вводу інформації на лініях зберігались “1”.**

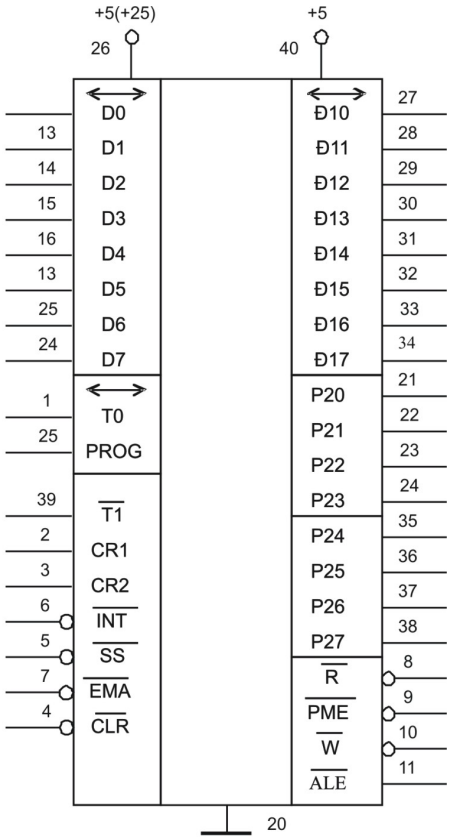

Рис. 3 Позначення МК48 на схемі

Канал P_0 – 8-розрядний двонаправлений регістр з трьома станами, який можна використовувати **для статично фіксованого виходу двонаправленого каналу даних та виводу молодших розрядів адреси при використанні зовнішньої пам'яті.**

1) Якщо порт P_0 використовується для **двонаправлених** передач, то обмін інформацією через нього використовується командами **MOVX.**

При виводі байту генерується стробуючий імпульс **W**, а сам байт, що виводиться, фіксується в вихідному регістрі (буферному).

При вводі байта генерується стробуючий імпульс **R**, а байт в регістрі **не фіксується.**

У відсутності передач порт P_0 за своїми виходами знаходиться у високоімпедансному стані.

2) Якщо порт використовується як однонаправлений, то вивід через нього виконується командою **OUT**L, а ввід - **INS**.

При прямій адресації три молодших біти визначають адресу РЗП (регістри R0..R7).

Функціональне призначення виводів

VSS – земля;

+5 В ОСН (VCC) – живлення в режимах роботи та програмування;

+5 В ОСН (VDD) – під час роботи **живиться тільки ОЗП.** В режимі програмування передається +25 В;

PGOG – вхід для подачі **програмуючого імпульсу +25** В при завантаженні РПЗП і як вихід для збільшення каналів вводу виводу;

CR1, CR2 – входи під'єднання **кварцевого резонатора;**

CLR – сигнал очистки (**скидання)** при запуску МК;

SS – сигнал, що разом із ALE при відлагодженні дозволяє виконувати програму з **зупинкою після виконання чергової команди;**

PME – видається тільки при звертанні **до зовнішньої пам'яті програм** (строб зчитування);

ALE – строб адреси зовнішньої пам’яті (для прийому та фіксації адреси зовнішньої пам’яті на зовнішньому регістрі);

R – строб при **читанні із зовнішньої пам’яті даних**;

W – строб при **запису в зовнішню пам’ять даних**;

T0 (вхід-вихід) – сигнал, що формується командами умовного переходу JT0 та JNT0, ***Вихід тактового сигналу*;**

T1 (вхід) – сигнал, запитуваний командами умовного переходу JT1 та JNT1. ***Вхід лічильника зовнішніх подій*;**

INT - **запит переривання** від зовнішнього джерела (викликає команду обслуговування переривання. **Ошибка! Ошибка внедренного объекта.** забороняє виконання цієї команди);

EMA – **відключення РПЗП.** “1” на цьому вході змушує МК виконувати вибірку команд тільки із зовнішньої пам’яті;

Порт 1 (P1) – 8-бітний **квазідвонаправлений** порт вводу-виводу, кожен розряд якого може бути запрограмований на ввод або вивід;

Порт 2 (P2) – 8-бітний **квазідвонаправлений** порт вводу-виводу**,** біти P2 0-3 під час читання із ВПП міняють старші розряди чотири біти лічильника команд ЛК 8-11;

Порт 3 (P3) – 8-бітний **двонаправлений** порт вводу-виводу інформації:

а) може під’єднуватися до навантаження;

б) може виконувати прийом та видачу байтів за сигналами **W, R.**

2. Програмування однокристальної мікро-ЕОМ МК48 з використанням програмного емулятора Single-Chip Machine

Система моделювання Single-Chip Machine є програмним емулятором для мікроконтролера MCS48 і призначена для:

- моделювання роботи мікро-ЕОМ MCS48 в сукупності з мікросхемою-розширником портів вводу- виводу КР580ВР43 і блоком зовнішньої пам'яті даних об'ємом 256 байт;
- розробки та налагодження програм для мікроконтролерів серії МК48;
- дослідження поведінки внутрішніх і зовнішніх сигналів зазначених мікросхем.

Інтерфейс емулятора зображено на рис 4.

Для завантаження емулятора потрібно з каталогу дистрибутива запустити файл setup.exe. Далі необхідно ознайомитися і слідувати вказівкам майстра установки.

Для запуску програми необхідно натиснути кнопку «Пуск», після чого в меню «Програми» вибрати папку з заданим при інсталяції ім'ям (SCM за замовчуванням).

У цій папці знаходяться два ярлики, а саме:

- SCM Help для виклику довідкової системи і
- Single-Chip Machine 1.xx для виконуваного модуля.

Для запуску програми необхідно вибрати ярлик Single-Chip Machine 1.xx і натиснути на ньому лівою кнопкою мишки.

SCM включає засоби налагодження і редагування програм на асемблері з вбудованим інтерпретатором, що робить введення програм набагато зручнішим і ефективнішим, ніж в інших емуляторах подібного класу.

Виконання програми користувача здійснюється з максимальним наближенням до дійсності за допомогою імітаційної моделі, рівень деталізації якої дорівнює одному такту (1т = 0.5 mks).

Можна налаштувати режим моделювання:

- на один такт вперед;
- на один машинний цикл вперед:
- на один крок вперед;
- виконання кроку до зміни регістра адреси мікроконтролера;
- виконання до найближчої точки зупину;
- виконання до кінця програми;
- виконання до першої порожньої комірки пам'яті;

- на один машинний цикл назад;
- на один такт назад.

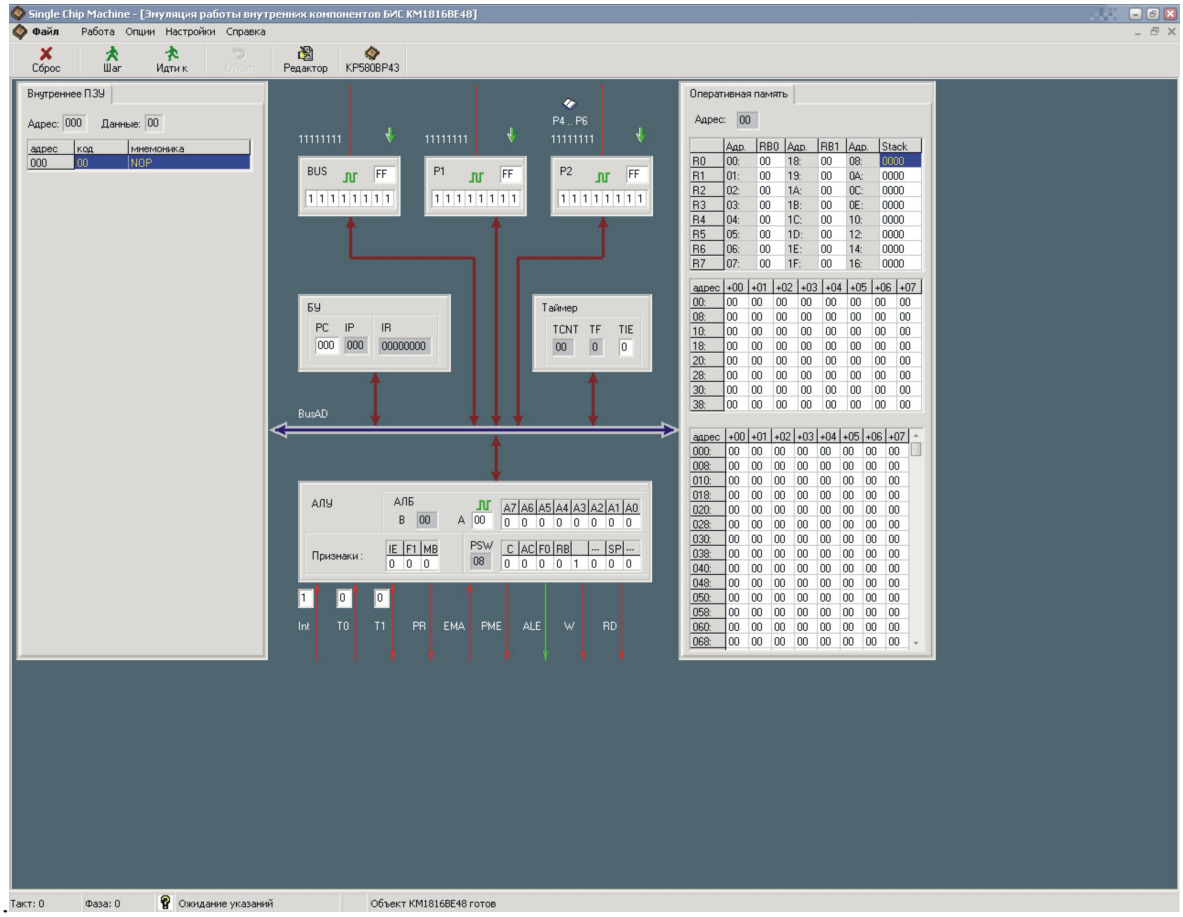

Рис. 4

Крім того, користувачеві надаються такі можливості як:

- часові діаграми внутрішніх і зовнішніх сигналів;
- імітація зовнішніх сигналів з відображенням змін на умовно-графічному відображенні мікросхем;
- можливість зміни значень вузлів мікроЕОМ в процесі роботи моделі та ін.

Вбудований редактор-компілятор дозволяє набирати програми на асемблері МК48, а потім за допомогою кнопки «компіляція» перевести текст програм в машинні коди і записати його, як в файл ПЗП з розширенням ".MPM"', так і в ПЗП мікроконтролера для відображення в відлагоджувачі.

Крім формату ПЗП, існує ще один формат пам'яті програм - формат «hex», який підтримується програмним забезпеченням всіх моделей програматорів.

Таким чином, SCM повністю сумісний з промисловими емуляторами.

Налаштування емулятора можна змінити двома шляхами.

Перший - за допомогою меню «Налаштування» і вибором відповідного пункту.

Другий - за допомогою самостійного редагування файлу SCMF.CFG, текстового файлу конфігурації програми SCN.

Якщо програму завантажити неможливо, то буде видано відповідне повідомлення, в якому пояснюється причину.

Інтерфейс емулятора містить:

у центральній частині:

- зображення трьох портів BUS, P1, P2 (молодший біт P0 – справа);

- програмний лічильник PC;
- молодші одинадцять розрядів програмного лічильника IP;
- вміст таймера-лічильника TCNT;
- тригер таймера-лічильника TF;
- тригер заборони переривання по таймеру-лічильнику TIE;
- акумулятор A;
- регістр для зберігання другого операнду (константи) B;
- тригер заборони переривання IE;
- значення старшого розряду лічильника команд MB, яке встановлюється командами Sel MB0 (логічний «0») та Sel MB1 (логічна «1»);
- флажок користувача F1;
- регістр слова-стану програми PSW: C – флажок перенесення, AC – флажок додаткового перенесення, F0 – фляжок користувача, RB – селектор банків регістрів, SP – вказівник стеку (три молодші розряди).

В лівій верхній частині інтерфейсу висвічується адреси та дані внутрішньої пам'яті програм (РПЗП).

В правій верхній частині розташована резидентна пам'ять даних (РПД), початкову область якої займають регістри R0…R7 банків RB0 та RB1, а також стек.

Під ними розташовується резидентна ОЗП, а ще нижче – зовнішня пам'ять даних (ЗПД).

Для завантаження вже існуючого файлу необхідно натиснути курсором на верхню ліву закладку «Файл», і далі – «Відкрити». З папки «Sourse» вибрати для завантаження файл на асемблері, з папки «Work» - об'єктний файл (з розширенням hex).

Для створення нового файлу потрібно курсором натиснути на опцію «Редактор» (знаходить в верхній частині інтерфейсу на панелі управління). З'явиться поле редактора (рис. 5), в якому записуються команди **асемблера** Вашої робочої програми.

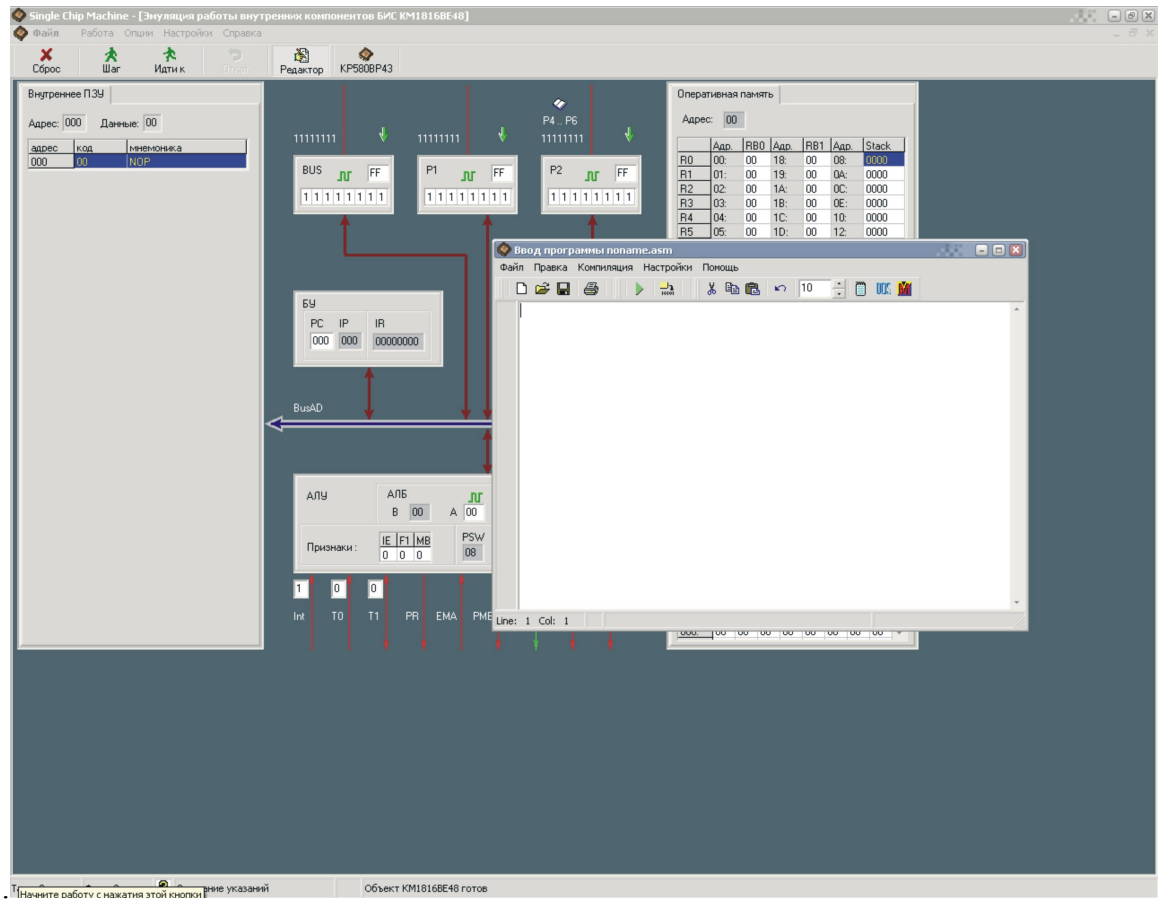

Рис. 5

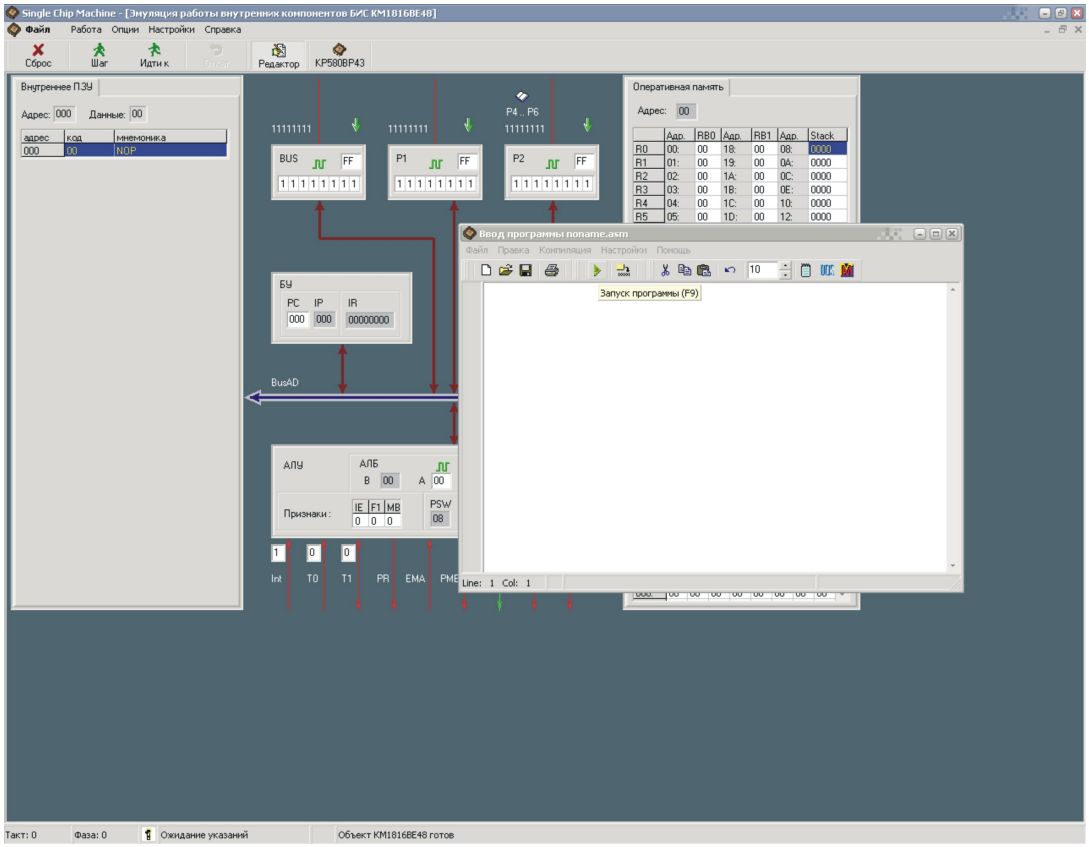
Single Chip Machine - [Эмуляция работы внутренних компонентов БИС КМ1816ВЕ48]
Файл
Работа
Опции
Настройки
Справка
Сброс
Шаг
Идти к
Редактор
КР580ВР43
Внутреннее ПЗУ
Адрес: 000
Данные: 00
адрес
код
мнемоника
000
00
NOP
11111111
P4 .. P6
BUS
P1
P2
FF
Оперативная память
Адр.
RB0
RB1
Stack
R0
R1
R2
R3
R4
R5
Ввод программы noname.asm
Файл
Правка
Компиляция
Настройки
Помощь
Запуск программы (F9)
Line: 1 Col: 1
БУ
PC
IP
IR
BusAD
АЛУ
АЛБ
B
A
Признаки :
IE
F1
MB
PSW
Int
T0
T1
PR
EMA
PME
Такт: 0
Фаза: 0
Ожидание указаний
Объект КМ1816ВЕ48 готов

Рис. 6

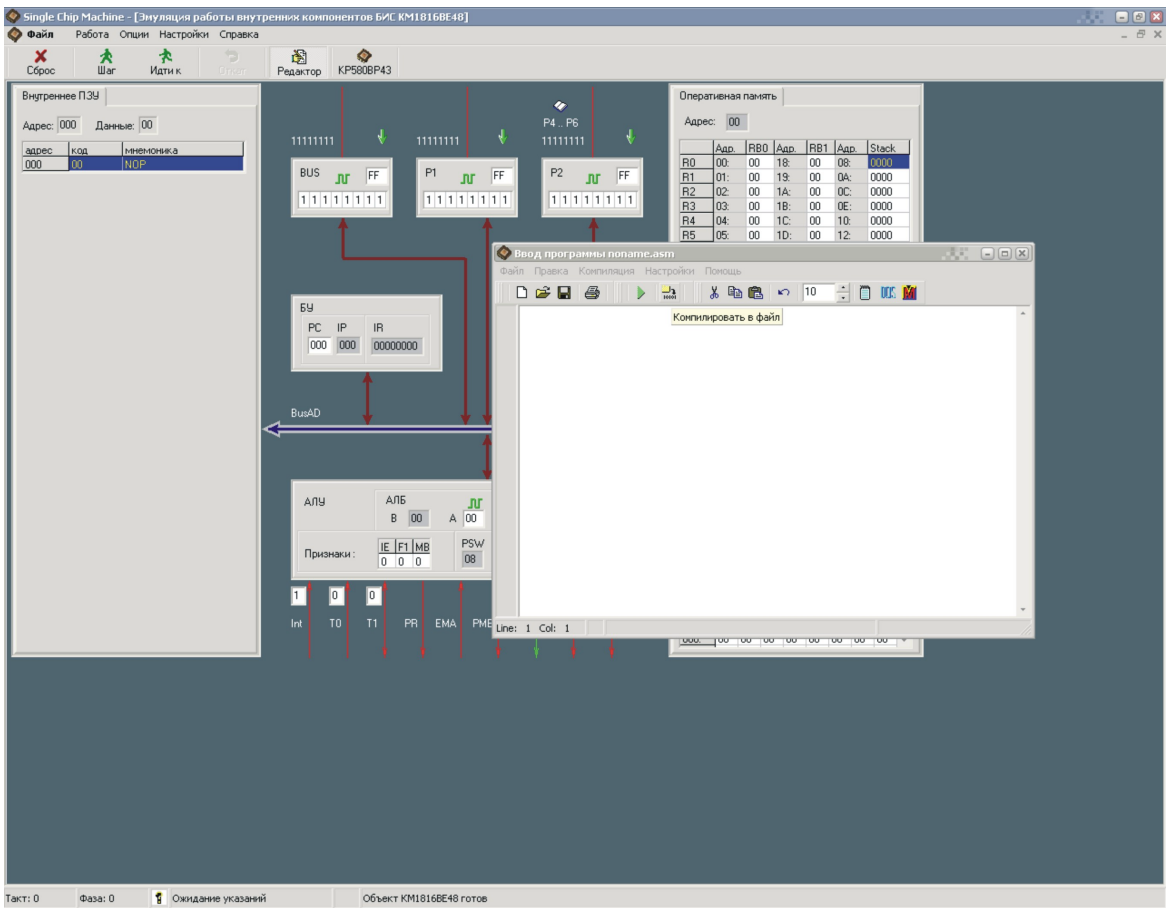
Single Chip Machine - [Эмуляция работы внутренних компонентов БИС КМ1816ВЕ48]
Файл
Работа
Опции
Настройки
Справка
Сброс
Шаг
Идти к
Редактор
КР580ВР43
Внутреннее ПЗУ
Адрес: 000
Данные: 00
адрес
код
мнемоника
000
00
NOP
11111111
P4 .. P6
BUS
P1
P2
FF
Оперативная память
Адр.
RB0
RB1
Stack
R0
R1
R2
R3
R4
R5
Ввод программы noname.asm
Файл
Правка
Компиляция
Настройки
Помощь
Компилировать в файл
Line: 1 Col: 1
БУ
PC
IP
IR
BusAD
АЛУ
АЛБ
B
A
Признаки :
IE
F1
MB
PSW
Int
T0
T1
PR
EMA
PME
Такт: 0
Фаза: 0
Ожидание указаний
Объект КМ1816ВЕ48 готов

Рис. 7

Для запуску програми на виконання потрібно натиснути на зображення трикутника зеленого кольору («Запуск редактора») в верхній частині вікна редактора (рис. 6).

Для компілювання програми необхідно натиснути курсором на кнопку справа від трикутника («Компіляція»), після чого відкомпільована програма з'явиться на лівому полі емулятора (поле ПЗП) (рис. 7).

Для запуску програми на виконання можна скористатися закладкою «Крок» на панелі управління емулятора (кроковий режим) або закладкою «Йти до» (програма виконується до заздалегідь позначеної команди).

Повернутися на початок програми можна закладкою «Скидання».

Приклад програми для роботи з емулятором контролера

```
; Демонстраційна програма можливостей
; компілятора і моделі однокристальної мікроЕОМ
; МК48 разом з зовнішними портами
; КР580ВР43 і блоком зовнішної пам'яті даних
          Org 0
          Jmp Program_start       ; перехід на початок програми
          Org 3
          Jmp External_inthandler ; обробка зовнішного переривання
          Org 7
          Jmp Timer_inthandler    ; обробка переривання таймера

External_inthandler:
           Anl    A, #7FH   ; Скинути 7-ий біт
           Dis    I         ; заборонити зовнішнє переривання
           Retr
Timer_inthandler:
           Clr    A          ; По перериванню таймера обнулити акумулятор
           Retr              ; повернення в програму

Program_start:               ; початок програми
           Outl   Bus, A
           Ins    A, Bus
           En     Tcnti      ; дозвіл переривання таймера

; Формування низькорівневого сигналу на всіх виводах порту P1
; тривалістю ~1000 мкс з використанням переривання таймера
           Clr    A          ; A=0
           Outl   P1, A      ; записати A в P1
           Mov    A, #0h-#12 ; 12*80=960 мкс.
           Mov    T, A       ; завантажити значение в регістр таймера
           Strt   T          ; запустити таймер
L1:
           Jb7    L1         ; Чекаємо завершення обробки переривання
                             ;(при A.7=0)
           Cpl    A          ; інвертуємо A (A=#0FFh)
           Outl   P1, A      ; підготуємо порт для читання
           Dis    Tcnti      ; заборонимо переривання від таймера
```

3. ЗАВДАННЯ для виконання лабораторної роботи

3.1. Використання команд пересилання даних

Варіант 1.

1. ***Записати в РПД в комірки з адресами 41 і 42 число 1C3FH:***

```
MOV  R0,#41          ; Завантаження в R0 покажчика РПД
 MOV  @R0,#1CH       ; Запис у РПД числа 1CH
 INC   R0            ; Просування покажчика адреси РПД
 MOV  @R0,#3FH       ; Запис у РПД числа 3FH
```

2. ***Передати вміст регістрів банку 0 у ЗПД, починаючи з адреси 50H:***

```
        MOV   A,#10000B    ; Вибір банку регістрів 1
        MOV   PSW,A
        MOV   R0,#50H      ; Визначення початкової адреси ЗПД
        MOV   R1,#0        ; Визначення початкової адреси банку регістрів
        MOV   R2,#8        ; Лічильник регістрів (циклів) <= 8
LOOP:   MOV   A,@R1        ; Пересилання байту з регістра у ЗПД
        MOVX  @R0,A        ; через акумулятор
        INC   R0           ; Просування покажчиків
        INC   R1
        DJNZ  R2,LOOP      ; Продовжити, якщо передані не всі регістри
```

Варіант 2.

1. ***Ввести байт із порту 1 і передати його в порт 2:***

```
        MOV   A,#0FFH      ; Налаштування порту 1 на ввід
        OUTL  P1,A
        IN A, P1           ; Ввід байту з порту 1
        OUTL  P2,A         ; Вивід байту в порт 2
```

2. Формування сигналу (#0C0h) через порт P2 тривалістю ~350 мкс з використанням таймера:

```
L2:   JTF   L2             ; Флжок TF скидається лише командою JTF
      MOV   A, #0C0H       ; Завантажуємо значення сигналу в акумулятор
      OUTl  P2, A          ; Виводимо значення в порт
      MOV   A, #0h-#4      ; 4*80=320 мкс.
      MOV   T, A           ; запустити таймер
      MOV   A, #0FFH       ; акумулятор прирівнюємо до початкового значення порту
      STRT  T              ; запустити таймер
L3:   JTF   L4
      JMP   L3
L4:   NOP                  ; Затримка на 2.5 мкс
      NOP                  ; Затримка на 2.5 мкс
      OUTl  P2, A          ; Відновлення порту P2
```

3.2. Використання команд арифметичних операцій

Варіант 3.

1. ***Додати вміст регістра R7 і комірки РПД з адресою 60H:***

```
        MOV  R0,#60H       ; Завантаження в R0 адреси РПД
        MOV  A,R7          ; Завантаження операнду в акумулятор
        ADD  A,@R0         ; Додавання
```

Результат операції додавання фіксується в акумуляторі, встановлення прапора перенесення буде свідчити про переповнення.

2. ***Інкрементувати вміст комірок РПД за адресами 10-18:***

```
        MOV  R0,#10        ; Завантаження в R0 початкової адреси
        MOV  R3,#(18-10+1) ; Завантаження в R3 числа комірок пам'яті
LOOP:   INC  @R0           ; Інкремент комірок РПД
        INC  R0            ; Просування покажчика адреси
        DJNZ R3,LOOP       ; Декремент R3 і повтор, поки R3 не дорівнює нулю
```

3. ***Віднімання байтів***. Операція віднімання може бути виконана двома способами:

- переведенням від'ємного числа, що віднімається, у додатковий код з наступним додаванням;
- переведенням зменшуваного в зворотний код з наступною інверсією суми.

1. ***Відняти від A вміст регістра R6.***

Вирахування виконати відповідно до виразу (A) <= не ((не (A)) +(R6))

```
MOV   R6,#10          ; Завантаження в R0 першого числа
MOV A,#15             ; Завантаження в A другого числа
CPL  A                ; Інверсія акумулятора
ADD A, R6             ; Додавання
CPL  A                ; Одержання різниці
```

Встановлення флажка C після виконання додавання буде свідчити про від'ємне переповнення.

2. ***Одержати різницю 2-байтових чисел без знаку***.

Операнди розташовуються у РПД. Адреса зменшуваного зберігається в R1, а від'ємника - у R0. Результат зписати на місце зменшуваного:

```
; Обчислити Z=X-Y
; X,Y - РПД
; R0 - Адреса Y
; R1 - Адреса X
; Результат на місце X
MOV  A,@R0          ; Завантаження молодшого байту Y
CPL    A            ; Одержання додаткового коду Y
INC    A
ADD    A,@R1        ; Вирахування молодших байтів
MOV   @R0,A         ; Запам'ятовування молодшого байту різниці
INC    R0           ; Перехід до старших байтів X і Y
INC    R1
MOV   A,@R0         ; Завантаження старшого байту Y
CPL    A            ; Зворотний код Y
ADDC A,@R1          ; Вирахування старших байтів
MOV   @R0,A         ; Запам'ятовування результату
```

3.4. Використання команд логічних операцій

Варіант 4.

1. ***Маскування при вводі. Ввести в регістр R7 інформацію з ліній 0,1,3,4 і 7 порту P1:***

```
IN     A,P1         ; Ввід байту з порту P1
ANL   A,#10011011B  ; Маскування
MOV  R7,A           ; Передача
```

2. ***Ввести в акумулятор дані з порту P2 і виділити необхідні біти маскою, що знаходиться в R0:***

```
IN     A,P2         ; Ввід з порту P2
ANL   A,R0          ; Маскування
```

3. ***Виконати арифметичний зсув двобайтового слова (R2) (A) вправо:***

```
MOV   R2,#10        ; Завантаження в R2 числа
CLR  C              ; Скидання флажка перенесення
CPL   C             ; Встановлення флажка перенесення
XCH  A,R2           ; Обмін байтами
RRC  A              ; Зсув флажка перенесення в розширник
XCH  A,R2           ; Обмін
RRC  A              ; Зсув молодшого байту
```

Варіант 5.

1. ***Видати в лінію 0 порту 4 значення четвертого біту акумулятора:***

```
MOV   R6,#13        ; Завантаження в A числа
SWAP  A             ; Обмін бітів 0 і 4 акумулятора
```

```
        ANL   A,#1          ; Виділення біту A.0
        ORLD  P4,A          ; Встановлення P4., якщо A.0=1
        ORL  A,#0EH         ; Встановлення бітів 1...3 акумулятора
        ANLD P4,A           ; Скидання P4.0, якщо A.0=0
```

2. ***Визначити парність кількості одиниць в акумуляторі:***

```
        MOV  R6,#13         ; Завантаження в R6 числа
        CLR  F0             ; Скидання F0
        MOV  R7,#8          ; Кількість повторів
LOOP:   RRC  A              ; Пересилання біту A.0 через перенесення
        JNC  NEXT           ; Пропустити, якщо біт дорівнює 0
        CPL  F0             ; Підрахунок паритету
NEXT:   DJNZ R7,LOOP        ; Повторити 8 разів
```

Варіант 6.

1. ***Видати вміст акумулятора в послідовному коді через нульову лінію порту 1, залишаючи без зміни інші біти порту.*** Передачу вести, починаючи з молодшого біту:

```
        MOV  R1,#8          ; Лічильник бітів
LOOP:   JB0  ONE            ; Перехід, якщо біт A.0 дорівнює 1
        ANL  P1,#(NOT 1)    ; Скидання P1.0
        JMP  NEXT
ONE:    ORL  P1,#1          ; Встановлення P1.0
        JMP  NEXT           ; Надлишкова команда для вирівнювання
                            ; часу передачі 0 і 1
NEXT:   RR   A              ; Зсув акумулятора вправо (підготовка
        DJNZ R1, LOOOP      ; до передачі чергового біту)
```

2. ***Запрограмувати біти 0-3 порту P1 не ввід:***

```
        ORL P1, #0FH        ; Встановлення бітів P1.0... P1.3
```

Очистити біти 4-7 порту 2:

```
        ANL P2,#0FH         ; Скидання бітів P2.4...P2.7
```

3. ***Видати в лінію 0 порту 4 значення четвертого біту акумулятора:***

```
        SWAP  A             ; Обмін бітів 0 і 4 акумулятора
        ANL   A,#1          ; Виділення біту A.0
        ORLD  P4,A          ; Встановлення P4.0, якщо A.0=1
        ORL  A,#0EH         ; Встановлення бітів 1...3 акумулятора
        ANLD P4,A           ; Скидання P4.0, якщо A.0=0
```

3.5. Використання команд передачі керування і команд керування режимом МК48

Варіант 7.

1. Передати керування мітці LL, якщо перемикач банків регістрів (біт PSW.4) встановлений:

```
        MOV  A,PSW          ; Передача PSW в акумулятор
        JB4  LL             ; Перехід, якщо A.4=1
LL:     MOV  R0,#13         ; Завантаження в R0 константи
```

Варіант 8.

1. ***Передати керування мітці LABEL, якщо лічильник подій досягне стану 64:***

```
        MOV A,1             ; Пересилання вмісту лічильника в акумулятор
        JB6 LABEL           ; Перехід за міткою, якщо A.6=1
```

```
LABEL:   MOV R0,A          ; Переслати вміст акумулятора в регістр R0
         INC R0            ; Збільшення на 1 вмісту регістра
```

2. ***Здійснити перехід з нульового банку ПП до програми з ім'ям ROUT, розташованої в першому банку ПП***:

```
         SEL  MB1          ; Встановлення флажка DBF
         JMP  ROUNT        ; Перехід до програми ROUNT
ROUNT:   MOV A,#0EH        ; Завантажити константу в акумулятор
```

Варіант 9.

1. ***При надходження на вхід T0 послідовності з восьми нульових імпульсів встановити вихід P2.7:***

```
         MOV  R7,#8        ; Завантаження в R7 числа імпульсів
ONE:     JT0  ONE          ; Чекання сигналу 0 на вході T0
ZERO:    JT0  SKIP         ; Чекання сигналу 1 на вході T0
         JMP  ZERO
SKIP:    DJNZ R7,ONE       ; Повторювати, поки не надійде восьмий імпульс
         ORL  P2,#80H      ; Встановлення одиниці на вході 7 порту 2
```

Тривалість нуля й одиниці на вході пристрою повинна бути не менш чотирьох машинних циклів, тобто 10 мкс.

2. ***Дочекатися надходження на вхід T0 100 імпульсів і перейти за міткою*** **PULSE:**

```
         MOV  A,#156       ; (A)  <== (256 - 100)
         MOV  T,A          ; Попереднє встановлення лічильника
         STRT CNT          ; Запуск лічильника
WAIT:    JTF  PULSE        ; Перехід, якщо прийшло 100 імпульсів
         JMP  WAIT
PULSE:   MOV  A, #FFh      ; Завантаження в акумулятор константи
```

Варіант 10.

1. ***Заборонити переривання від таймера, але дозволити переривання після восьми сигналів переповнення таймера.*** При переході до процедури обробки переривання зупинити таймер. Сигнали переповнення підраховувати в регістрі R5:

```
         DIS   TCNT1       ; Заборона переривання від таймера
         CLR   A           ; Скидання акумулятора
         MOV   T,A         ; Скидання таймера
         MOV   R5,A        ; Скидання регістра R5
         STRT  T           ; Запуск таймера
M1:      JTF   COUNT       ; Якщо TF=1, то перехід до COUNT і скидання TF
         JMP   M1          ; Цикл
COUNT:   INC   R5          ; Інкремент регістра R5
         MOV   A,R5        ; Пересилання вмісту R5 в акумулятор
         JB3   INT         ; Перехід до підпрограми обслуговування
                           ; переривання INT, якщо біт A.3 дорівнює 1
         JMP   M1          ; Перехід, якщо біт A.3 не дорівнює 1
INT:  STOP TCNT            ; Зупинка таймера
      JMP  07H             ; Перехід до комірки 7  (вектор переривання
                           ; від  лічильника подій)
```

Для кожного з варіантів завдання після виконання кожної з команд записати вміст регістрів, які змінюються в процесі виконання програми, в табл.1.

Таблиця 1

Регістр	PC	BUS	P1	P2	A	PSW	RB0	RB1	C	FC	F0	RB	SP	TIE
Команда 1														
Команда 2														
……..														
Команда n														

Література

1. Проектирование цифровых устройств на однокристальных микроконтроллерах / В. В. Сташин и др. - М.: Энергоатомиздат, 1990. – 224 с.
2. Микропроцессоры / Под ред. Преснухина Л.Н., т. 1, 2, 3. - М.: Высшая школа, 1986.
3. Бойко Н.П., Стеклов В.К. Системы автоматического управления на базе микро-ЭВМ. - К.: Техника, 1989. - 182 с.

Додаток1

Команди МК48

1. Команди, що модифікують РССП та ознаки користувача

Усі команди, через які модифікуються ознаки, наведені у таблиці 1. Логічні команди не змінюють регістр ознак.

Таблиця 1. Команди модифікації ознак

Команди	Ознаки	Опис команди
ADD	C, AC	Команда додавання до акумулятора
ADDC	C, AC	Команда додавання до акумулятора з урахуванням перенесення
DA A	C, AC	Команда десяткової корекції після додавання
CLR C	C = 0	Скидання ознаки перенесення
CPL C	C	Інверсія ознаки перенесення
CLR F0	F0	Скидання ознаки користувача F0
CLR F1	F1	Скидання ознаки користувача F1
CPL F0	F0	Інверсія ознаки користувача F0
CPL F1	F1	Інверсія ознаки користувача F1
JTF	TF = 0	Перехід, якщо біт TF рівний 1. Команда скидає біт переповнення таймера TF
MOV PSW, A	C,AC,F0,BS	Запис даних у РССП
RETR	C,AC,F0,BS, GEI	Повернення з підпрограми або переривання
RLC	C	Лівий циклічний зсув бітів через біт C
RRC	C	Правий циклічний зсув бітів через ознаку перенесення
SEL MB0 SEL MB1	MB	Вибір банків пам'яті програм. Команда задає нульовий чи перший банк пам'яті програм
SEL RB0 SEL RB1	BS	Вибір банків регістрів. Команда активізує нульовий або перший банк регістрів загального призначення

2. Група команд пересилання даних

Усі команди (крім MOV PSW, A) не впливають на ознаки. Команди пересилання даних усередині МК виконуються за один машинний цикл, обмін із зовнішньою пам'яттю і портами вимагає двох машинних циклів.

Більшість команд пересилає 8-бітні (байтові) операнди. Існують кілька команд, що оперують з 4-бітними операндами (тетрадами). Команди пересилання тетрад використовують при звертанні до 4-бітних портів зовнішньої схеми розширювача вводу-виводу (P4÷P7).

Формат, дія та назва кожної команди пересилання даних наведені у таблиці 2.

У таблиці використані позначення: Т – тип команди, Б – кількість байтів у коді команди, Ц – тривалість виконання команди у машинних циклах.

Таблиця 2. Команди пересилання даних

Позначення	Код	Т	Б	Ц	Назва та дія команди
MOV A, Rn	11111rrr	1	1	1	Пересилання регістра в акумулятор: A = Rn
MOV A, @Ri	111000i	1	1	1	Пересилання байта з ВПД в аккумулятор, адреса комірки внутрішньої пам'яті даних знаходиться у регістрі R0 (R1): A = IRAM[Ri]
MOV A, #d	00100011 $d_{7\div0}$	2	2	2	Пересилання безпосереднього операнда в акумулятор: A = #d
MOV Rn, A	10101rrr	1	1	1	Пересилання акумулятора в регістр: Rn = A
MOV Rn, #d	10111rrr $d_{7\div0}$	2	2	2	Пересилання безпосереднього операнда в регістр: Rn = #d
MOV @Ri, A	1010000i	1	1	1	Пересилання акумулятора у внутрішню пам'ять даних, адреса комірки пам'яті знаходиться у регістрі R0 чи R1: IRAM[Ri] = A

Таблиця 2. Команди пересилання даних

Позначення	Код	Т	Б	Ц	Назва та дія команди
MOV @Ri, #d	1011000i $d_{7\div0}$	2	2	2	Пересилання безпосереднього операнда у внутрішню пам'ять даних, адреса комірки знаходиться у регістрі R0 чи R1: IRAM[Ri] = #d
MOV A, PSW	11000111	1	1	1	Пересилання слова стана програми в акумулятор: A = PSW
MOV PSW, A	11010111	1	1	1	Пересилання акумулятора у РССП: PSW = A
MOV A, T	10000100	1	1	1	Пересилання вмісту лічильника-таймера в акумулятор: A = T
MOV T, A	01100010	1	1	1	Пересилання акумулятора в лічильник-таймер: T = A
MOVX A,@Ri	1000000i	1	1	2	Пересилання байта із зовнішньої пам'яті даних в акумулятор, адреса комірки пам'яті даних знаходиться у регістрі R0 чи R1: A = ERAM[Ri]
MOVX @Ri,A	1001000i	1	1	2	Пересилання акумулятора у зовнішню пам'ять даних, адреса комірки пам'яті знаходиться у регістрі R0 чи R1: ERAM[Ri] = A
MOVP A,@A	10100011	1	1	2	Пересилання байта з поточної сторінки програмної пам'яті в акумулятор: A = ROM[$PC_{11\div8}$:A]
MOVP3 A,@A	11100011	1	1	2	Пересилання байта з третьої сторінки програмної пам'яті в акумулятор: A = ROM[0011:A]
XCH A,Rn	00101rrr	1	1	1	Обмін регістра з акумулятором: A ↔ Rn
XCH A,@Ri	0010000i	1	1	1	Обмін акумулятора з внутрішньою пам'ятю даних, адреса комірки пам'яті даних знаходиться у регістрі R0 чи R1: A ↔ IRAM[Ri]
XCHD A,@Ri	0011000i	1	1	1	Обмін молодших тетрад акумулятора та з внутрішньою пам’яттю даних, адреса комірки пам'яті знаходиться у регістрі R0 чи R1: $A_{3...0}$ ↔ $IRAM[Ri]_{3...0}$
IN A,Pp	000010pp	1	1	2	Пересилання даних з портів вводу-виводу Pp (p =1,2) в акумулятор: A = Pp
INS A,BUS	00001000	1	1	2	Стробоване введення даних з P0: A = BUS
SWAP A	01000111	1	1	1	Обмін тетрад акумулятора: $A_{3...0}$ ↔ $A_{7...4}$
OUTL Pp,A	001110pp	1	1	2	Пересилання A у порти P1 та P2 (Pp p = 1, 2): Pp = A
OUTL BUS, A	00000010	1	1	2	Стробований вивід даних з A в порт BUS: BUS = A
MOVD A,Pp	000011pp	1	1	2	Ввід тетради з порту Pp (p = 4 – 7) схеми розширення: $A_{3...0}$ = Pp
MOVD Pp,A	001111pp	1	1	2	Вивід тетради в порт Pp (p = 4 – 7) схеми розширення: Pp = $A_{3...0}$

3. Група команд арифметичні операції

Група складається з 12 команд і дозволяє виконувати такі операції над 8-бітними цілими двійковими числами без знака: двійкове додавання (ADD), двійкове додавання з урахуванням перенесення (ADDC), десяткова корекція (DAA), інкрементування (INC) і декрементування (DEC) регістрів MCS-48. Опис команд наведений у таблиці 3.

Таблиця 3. Група команд арифметичних операцій

Позначення	Код	Т*)	Б*)	Ц*)	Назва та дія команди
ADD A,Rn	01101rrr	1	1	1	Додавання регістра R0 – R7 до акумулятора: A = A + Rn
ADD A,@Ri	0110000i	1	1	1	Додавання до акумулятора байта із внутрішньої пам'яті даних. Адреса комірки пам'яті знаходиться у регістрі R0 чи

Таблиця 3. Група команд арифметичних операцій

Позначення	Код	Т*)	Б*)	Ц*)	Назва та дія команди
					R1: A = A + IRAM[Ri]
ADD A,#d	00000011 $d_{7\div0}$	2	2	2	Додавання до акумулятора константи: A = A + #d
ADDC A,Rn	01111rrr	1	1	1	Додавання до акумулятора регістра R0 – R7 та ознаки перенесення: A = A + Rn + C
ADDC A,@d	00010011 $d_{7\div0}$	2	2	2	Додавання до акумулятора константи та ознаки перенесення: A = A + #d
ADDC A,@Ri	0111000i	1	1	1	Додавання до акумулятора байта із внутрішньої пам'яті даних та ознаки перенесення. Адреса комірки пам'яті знаходиться у регістрі R0 чи R1: A = A + IRAM[Ri] + C
DAA	01010111	1	1	1	Десяткова корекція акумулятора. Команда виправляє результат додавання за типовими правилами
INC A	00010111	1	1	1	Інкремент акумулятора: A = A + 1
INC Rn	00011rrr	1	1	1	Інкремент регістра: Rn = Rn+l
INC @Ri	0001000i	1	1	1	Інкремент байта у пам'яті. Адреса комірки пам'яті знаходиться у регістрі R0 чи R1: IRAM[Ri] = IRAM[Ri] + 1
DEC A	00000111	1	1	1	Декремент акумулятора: A = A – l
DEC Rn	11001rrr	1	1	1	Декремент регістра R0 – R7: Rn = Rn – 1

Команда додавання ADD додає до акумулятора інший операнд. Вміст акумулятора А можна просумувати з регістром загального призначення, з константою, або коміркою внутрішньої пам'яті даних. Старший (дев'ятий) біт результату фіксується у біті С регістра РССП.

Команда додавання з урахуванням перенесення ADDC працює аналогічно команді додавання ADD, але до суми додає попереднє значення ознаки перенесення. Команда призначена для сумування багатобайтових чисел.

У системі команд відсутні команди віднімання, множення, ділення. Операції множення та ділення слід оформляти підпрограмами.. У процесі написання програми команди віднімання необхідно замінювати на команди додавання з числом записаним у додатковому коді.

4. Група команд роботи з бітами

Група складається із 27 команд і дозволяє виконувати такі операції: логічне І, логічне АБО, виключне АБО, інверсію, скидання і зсув. Дві команди (скидання й інверсія) дозволяють виконувати операції над бітами. Формат та характеристики команд наведені у таблиці 4.

Таблиця 4. Група команд роботи з бітами

Позначення	Код	Т*)	Б*)	Ц*)	Назва та дія команди
ANL A,Rn	01011rrr	1	1	1	Логічне І акумулятора й регістра R0 ÷ R7: A = A /\ Rn
ANL A,@Ri	0101000i	1	1	1	Логічне І байта акумулятора та комірки внутрішньої пам'яті даних. Адреса комірки пам'яті знаходиться у регістрі R0 чи R1: A = IRAM[Ri] /\ A
ANL A,#d	01010011 $d_{7\div0}$	2	2	2	Логічне І константи та акумулятора: A = A /\ #d
ORL A,Rn	01001rrr	1	1	1	Логічне АБО акумулятора та регістра R0 ÷ R7: A = A \/ Rn
ORL A,@Ri	0100000i	1	1	1	Логічне АБО акумулятора та комірки внутрішньої пам'яті даних. Адреса комірки пам'яті знаходиться у регістрі R0 чи R1:

Таблиця 4. Група команд роботи з бітами

Позначення	Код	Т*)	Б*)	Ц*)	Назва та дія команди
					A = IRAM[Ri] ∨ A
ORL A,#d	01000011 $d_{7\div0}$	2	2	2	Логічне АБО акумулятора та константи: A = A ∨ #d
XRL A,Rn	11011rrr	1	1	1	Виключне АБО акумулятора й регістра R0 ÷ R7: A = A ⊕ Rn
XRL A,@Ri	1101100i	1	1	1	Виключне АБО акумулятора та комірки внутрішньої пам'яті даних. Адреса комірки пам'яті знаходиться у регістрі R0 чи R1: A = A ⊕ IRAM[Ri]
XRL A,#d	11010011 $d_{7\div0}$	2	2	2	Виключне АБО константи й акумулятора: A = A ⊕ #d
CPA	00110111	1	1	1	Інверсія акумулятора:
CLR A	00100111	1	1	1	Скидання акумулятора: A = 0
RL A	11100111	1	1	1	Циклічний зсув вліво вмісту акумулятора: $A_{n+1} = A_n$, n = 0 ÷ 6, $A_0 = A_7$
RLC A	11110111	1	1	1	Зсув вліво акумулятора через ознаку перенесення: $A_{n+1} = A_n$, n = 0...6, A_0 = C, C = A_7
RR A	01110111	1	1	1	Циклічний зсув вправо акумулятора: $A_n = A_{n+l}$, n = 0 ÷ 6, $A_7 = A_0$
RRC A	01100111	1	1	1	Зсув вправо акумулятора через перенесення: $A_n = A_{n+l}$, n = 0 ÷ 6, A_7 = C; C = A_0
ANL Pp,#d	100110pp $d_{7\div0}$	2	2	2	Логічне І константи та даних з порта Pp: Pp = Pp /\ #d, p = 1, 2
ANL BUS,#d	10011000 $d_{7\div0}$	2	2	2	Логічне І константи та даних з порта BUS: BUS =BUS /\ #d
ANLD Pp,A	100111pp	1	1	2	Логічне І акумулятора та даних з порта Pp (p = 7 ÷ 4): Pp = Pp /\ A_{3-0}
ORL BUS,#d	10001000 $d_{7\div0}$	2	2	2	Логічне АБО константи та даних BUS: BUS = BUS ∨ #d
ORLD Pp,A	100011pp	1	1	2	Логічне АБО акумулятора і порту Pp (p = 7 ÷ 4): Pp = Pp ∨ $A_{3...0}$
CLR C	10010111	1	1	1	Скидання ознаки перенесення C = 0
CLR F0	10000101	1	1	1	Скидання ознаки користувача F0 (F0 = 0)
CLR F1	10100101	1	1	1	Скидання ознаки користувача F1 (F1 = 0)
CPL C	10100111	1	1	1	Інверсія ознаки перенесення
CPL F0	10010101	1	1	1	Інверсія ознаки користувача F0 ()
CPL F1	10110101	1	1	1	Інверсія ознаки користувача F1 ()

Команди виконання логічних операцій, котрі працюють з портами P4…P7, виконує розширювач вводу-виводуК580ВВ43: мікроконтролер видає через лінії P2.3…2.0 код команди та необхідні дані, а дії над даними виконує К580ВВ43.

5. Група команд передачі керування

Групу утворюють 19 команд передавання керування, з них дві команди безумовного переходу, 14 команд умовного переходу, команда виклику підпрограм і дві команди повернення з підпрограм.

Команди умовних переходів здійснюють перехід лише у випадку виконання певної умови. Якщо ж умова не справджується здійснюється перехід до наступної команди.

Команди CALL, RET та RETR виконують дії зі стеком.

У більшості команд прямо вказана адреса переходу. У тілі команди при цьому зберігаються 8 ($ad_{7\div0}$) чи 11 ($ad_{10\div0}$) бітів адреси переходу. Команда JMP дозволяє передати керування у будь-яке місце

2048-байтного банку пам'яті програм (ПП). Номер банку ПП визначається ознакою DBF, значення якого копіюється у старший біт лічильника команд (PC11) при виконанні команди JMP чи CALL.

Для переходу з нульового банку ПП у перший недостатньо тільки встановити ознаку DBF, необхідно також виконати команду JMP або CALL, що змінить значення старшого біта лічильника команд.

Таблиця 5. Команди передавання керування

Позначення	Код	Т*)	Б*)	Ц*)	Назва та дія команди
1	2	3	4	5	6
JMP ad11	$a_{10}a_9a_8$00100 $a_{7\div0}$	3	2	2	Безумовний перехід: PC_{11} = DBF, $PC_{10\div0} = a_{10\div0}$
JMPP @A	10110011	1	1	2	Непрямий перехід у середині сторінки: $PC_{7\div0}$ = A
DJNZ Rn, ad	11101rrr $ad_{7\div0}$	4	2	2	Декремент регістра та перехід, якщо результат ненульовий. Здійснюється перехід на адресу задану другим байтом у середині сторінки пам'яті програм
JC ad	11110110 $ad_{7\div0}$	4	2	2	Умовний перехід, за перенесенням. Якщо ознака C = 1, то $PC_{7\text{-}0} = ad_{7\div0}$
JNC ad	11100110 $ad_{7\div0}$	4	2	2	Умовний перехід, у випадку відсутності перенесення. Якщо C=0, то $PC_{7\div0} = ad_{7\div0}$
JZ ad	11000110 $ad_{7\div0}$	4	2	2	Умовний перехід, за нульовим значенням акумулятора. Якщо A = 0, то $PC_{7\div0} = ad_{7\div0}$
JNZ ad	10010110 $ad_{7\div0}$	4	2	2	Перехід, якщо значення акумулятора ненульове. Якщо $A \neq 0$, $PC_{7\div0} = ad_{7\div0}$
JT0 ad	00110110 $ad_{7\div0}$	4	2	2	Умовний перехід за високим рівнем на вході T0. Якщо T0 = 1, то $PC_{7\div0} = ad_{7\div0}$
JNT0 ad	00100110 $ad_{7\div0}$	4	2	2	Умовний перехід за низьким рівнем на вході T0. Якщо T0 = 0, то $PC_{7\div0} = ad_{7\div0}$
JT1 ad	01010110 $ad_{7\div0}$	4	2	2	Умовний перехід за високим рівнем на вході T1. Якщо T1 = 1, то $PC_{7\div0} = ad_{7\div0}$
JNT1 ad	01000110 $ad_{7\div0}$	4	2	2	Умовний перехід за низьким рівнем на вході T1. Якщо T1 = 0, то $PC_{7\div0} = ad_{7\div0}$
JF0 ad	10110110 $ad_{7\div0}$	4	2	2	Умовний перехід за станом ознаки F0. Якщо F0 = 1, то $PC_{7\div0} = ad_{7\div0}$
JF1 ad	01110110 $ad_{7\div0}$	4	2	2	Умовний перехід за станом ознаки F1. Якщо F1 = 1, то $PC_{7\div0} = ad_{7\div0}$
JTF ad	00010110 $ad_{7\div0}$	4	2	2	Умовний перехід за станом таймера. Якщо таймер з моменту попереднього виконання команди переповнювався (ознака TF = 1), то $PC_{7\div0} = ad_{7\div0}$, а TF обнулюється
JNI ad	10000110 $ad_{7\div0}$	4	2	2	Умовний перехід за станом лінії INT. Якщо на виводі INT логічний 0, то $PC_{7\div0} = ad_{7\div0}$
JBb ad	$b_2b_1b_0$10010 $ad_{7\div0}$	4	2	2	Умовний перехід, у випадку рівності 1 бітів акумулятора. Якщо біт регістра A, номер котрого заданий кодом $b_{2\div0}$ дорівнює одиниці, то $PC_{7\div0} = ad_{7\div0}$
CALL ad11	$a_{10}a_9a_8$10100 $ad_{7\div0}$	3	2	2	Виклик підпрограми. Адреса команди, що йде за командою виклика записується у стек: Stack[S] = ($PSW_{7\div4}$, PC+2), S = S + 1, PC_{11} = DBF, $PC_{10\div0} = ad_{10\div0}$
RET	10000011	1	1	2	Повернення з підпрограми зі збереженням слова стану: S = S – 1, PC = Stack[S]
RETR	10010011	1	1	2	Повернення з підпрограми з відновленням слова стану: S = S – 1, ($PSW_{7\div4}$, PC) = Stack[S]

6. Команди керування режимом роботи лічильника-таймера

У групу входять команди завантаження, зчитування, керування лічильником-таймером та перериванням від лічильника-таймера. У табл. 6 наведені формати команд цієї групи.

Таблиця 6**.** Група команд керування лічильником-таймером

Позначення	Код	Т*)	Б*)	Ц*)	Назва та дія команди
MOV A, T	10000100	1	1	1	Пересилання вмісту лічильника-таймера в акумулятор: A = T
MOV T, A	01100010	1	1	1	Пересилання акумулятора в лічильник-таймер: T = A
STRT T	01010101	1	1	1	Запуск таймера
STRT CNT	01000101	1	1	1	Запуск лічильника
STOP TCNT	01100101	1	1	1	Зупинка лічильника-таймера
TCNTI	00100101	1	1	1	Дозвіл переривання від лічильника-таймера
DIS TCNTI	00110101	1	1	1	Заборона переривання від лічильника-таймера

7. Команди перемикання банків регістрів та пам'яті програм

Перемикання банків пам'яті програм, зміна старшого біта лічильника команд відбувається у момент виконання команди виклику підпрограми. Наявність команд перемикання банків регістрів дозволяє при виклику підпрограм та виконанні підпрограм переривань ефективно використовувати другий банк регістрів як робочий, зберігаючи параметри обчислювального процесу не в стеку, а у вихідному банку регістрів.

Програмуючи процедури переривань, можна перемикати чи не перемикати банки регістрів. У випадку, коли банки регістрів перемикаються, повернення до вихідного банку регістрів буде виконано автоматично, якщо підпрограма переривань закінчується командою повернення з відновленням РССП (RETR).

Таблиця 7. Група команд керування контролером

Позначення	Код	Т*)	Б*)	Ц*)	Назва та дія команди
EN I	00000101	1	1	1	Дозвіл зовнішнього переривання
DIS I	00010101	1	1	1	Заборона зовнішнього переривання
SEL RB0	11000101	1	1	1	Вибір нульового банку регістрів: BS = 0
SEL RB1	11010101	1	1	1	Вибір першого банку регістрів: BS = 1
SEL MB0	11100101	1	1	1	Вибір нульового банку пам'яті програм: DBF = 0
SEL MB1	11110101	1	1	1	Вибір нульового банку пам'яті програм: DBF = 1
ENT0 CLC	01110101	1	1	1	Дозвіл видавання сигналу на вихід T0
NOP	00000000	1	1	1	Команда "немає операції"