Практическая работа №1

Подключение внешней памяти и ее тестирование

Цель работы: изучение схемы подключения микроконтроллера с внешней памятью и протестировать память.

1. Микроконтроллер i8051

Семейство MCS-51 включает более 200 модификаций 8-разрядных микроконтроллеров с широким спектром периферии, которые имеют общую систему команд. Наличие дополнительного оборудования влияет только на количество регистров специального назначения.

Основными характеристиками микроконтроллеров i8051 являются:

- внутренне ОЗУ объемом 128 байт;
- четыре двунаправленных побитно настраиваемых восьмиразрядных порта ввода-вывода;
- два 16-разрядных таймера-счетчика;
- встроенный тактовый генератор;
- адресация 64 Кбайт памяти программ и 64 Кбайт памяти данных;
- две линии запросов на прерывание от внешних устройств;
- интерфейс для последовательного обмена информацией с другими микроконтроллерами или персональными компьютерами.

На рис. 1 приведено назначение выводов МК 8051.

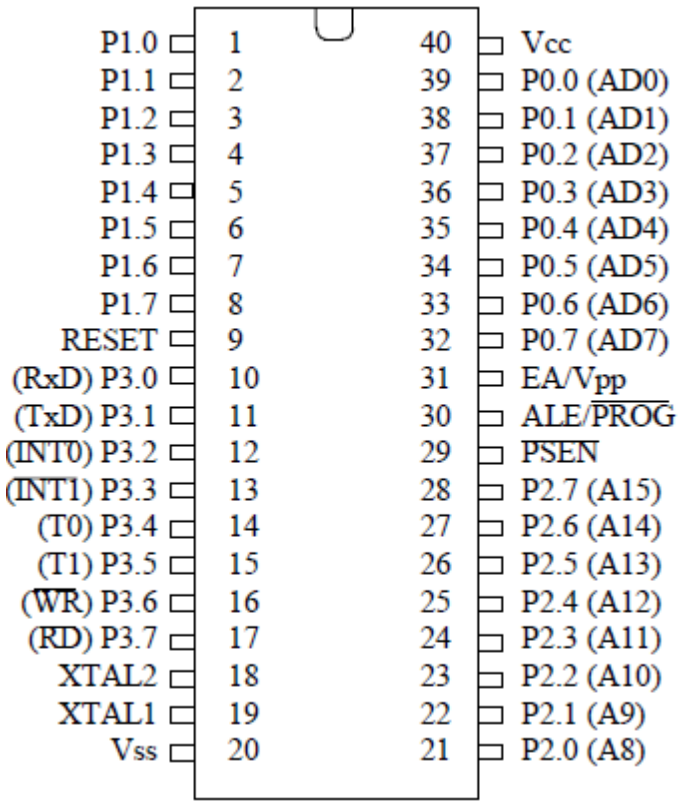

Рис. 1. Назначение выводов 8051

Uss — потенциал общего провода ("земли");

Ucc — основное напряжение питания +5 В;

X1,X2 — выводы для подключения кварцевого резонатора;

RST — вход общего сброса микроконтроллера;

PSEN — разрешение внешней памяти программ; выдается только при обращении к внешнему ПЗУ;

ALE — строб адреса внешней памяти;

EA — отключение внутренней программной памяти; уровень 0 на этом входе заставляет микроконтроллер выполнять программу только внешнее ПЗУ; игнорируя внутреннее (если последнее имеется);

P1 — восьми битный квазидвунаправленный порт ввода/вывода: каждый разряд порта может быть запрограммирован как на ввод, так и на вывод информации, независимо от состояния других разрядов;

P2 — восьми битный квазидвунаправленный порт, аналогичный P1; кроме того, выводы этого порта используются для выдачи адресной

информации при обращении к внешней памяти программ или данных (если используется 16-битовая адресация);

P3 — восьми битный квазидвунаправленный порт, аналогичный P1; кроме того, выводы этого порта могут выполнять ряд альтернативных функций, которые используются при работе таймеров, порта последовательного ввода-вывода, контроллера прерываний, и внешней памяти программ и данных;

P0 — восьми битный двунаправленный порт ввода-вывода информации: при работе с внешними ОЗУ и ПЗУ по линиям порта в режиме временного мультиплексирования выдается адрес внешней памяти, после чего осуществляется передача или прием данных.

Все порты МК имеют возможность побитовой адресации разрядов и являются двунаправленными, причем имеется возможность в каждом порту часть разрядов использовать для ввода данных, а часть для вывода.

Каждый из портов содержит регистр-защелку (SFR P0 — SFR P3), выходную цепь и входной буфер. Каждый из разрядов регистра-защелки SFR является D-триггером, информация в который заносится с внутренней шины данных микроконтроллера по сигналу «Запись в SFR Px» (x= 0, 1, 2, 3) от центрального процессора (ЦП). С прямого выхода D-триггера информация может быть выведена на внутреннюю шину по сигналу «Чтение SFR Px» от ЦП, а с вывода микросхемы по сигналу «Чтение выводов Px». Одни команды активизируют сигнал «Чтение SFR PI», другие - «Чтение выводов PI».

Для перевода любой линии портов P1 — P3 в режим ввода информации необходимо в соответствующий разряд SFR занести 1.

Корпус МК имеет небольшое количество выводов, поэтому помимо основных функций они выполняют еще ряд альтернативных функций. Порты P0 и P2 используются при обращении к внешней памяти. При этом на выходах P0 младший байт адреса внешней памяти мультиплексируется с вводимым/выводимым байтом. Выходы P2 содержат старший байт адреса внешней памяти, если адрес 16-разрядный. При использовании

восьмиразрядного адреса портом P2 можно пользоваться для ввода-вывода информации обычным образом. При обращении к внешней памяти в P0 автоматически заносятся 1 во все биты. Информация в P2 при этом остается неизменной. Порт P3 помимо обычного ввода и вывода информации используется для формирования и приема специальных управляющих и информационных сигналов. Разряды порта (все или частично) при этом могут выполнять альтернативные функции приведенные в табл. 1.

Таблица 1

Функции выводов порта P3

Вывод порта	Альтернативная функция
P3.0	RXD - вход последовательного порта
P3.1	TXD - выход последовательного порта
P3.2	INT0 - внешнее прерывание 0
P3.3	INT1 - внешнее прерывание 1
P3.4	T0 - вход таймера-счетчика 0
P3.5	T1 - вход таймера-счетчика 1
P3.6	WR - строб записи во внешнюю память данных
P3.7	RD - строб чтения из внешней памяти данных

У МК 8051 память программ и память данных являются самостоятельными и независимыми друг от друга устройствами, адресуемыми различными командами и управляющими сигналами. Объем встроенной памяти программ – 4 Кбайт, расположенной на кристалле памяти данных – 128 байт. Помимо встроенной памяти имеется возможность подключить внешнюю память программ и данных объемом 64 Кбайт.

Первые 32 байта адресного пространства данных организованы в виде четырех банков регистров общего назначения, каждый из которых состоит из восьми регистров R0-R7. В любой момент программе доступен только один банк регистров. Выбор банка регистров осуществляется установкой соответствующих бит слова состояния программы PSW.

К адресному пространству памяти данных примыкает адресное пространство регистров специальных функций SFR (Special Function Register). Доступные регистры SFR приведены в табл. 2.

Таблица 2

Регистры SFR

Символ	Наименование
*ACC	Аккумулятор (Accumulator)
*B	Регистр расширитель аккумулятора (Multiplication Register)
*PSW	Слово состояния программы (Program Status Word)
*P0	Порт 0 (SFR P0)
*P1	Порт 1 (SFR P1)
*P2	Порт 2 (SFR P2)
*P3	Порт 3 (SFR P3)
SP	Регистр указатель стека (Stack Pointer)
DPH	Старший байт регистра указателя данных DPTR (Data Pointer High)
DPL	Младший байт регистра указателя данных DPTR (Data Pointer Low)
TH0	Старший байт таймера 0 ()
TL0	Младший байт таймера 0 ()
TH1	Старший байт таймера 1 ()
TL1	Младший байт таймера 1 ()
TMOD	Регистр режимов таймеров счетчиков (Timer/Counter Mode Control Register)
*TCON	Регистр управления статуса таймеров (Timer/Counter Control Register)
*IP	Регистр приоритетов (Interrupt Priority Control Register)

Окончание табл. 2

*IE	Регистр маски прерывания (Interrupt Enable Register)
PCON	Регистр управления мощностью (Power Control Register)
*SCON	Регистр управления приемопередатчиком (Serial Port Control Register)
SBUF	Буфер приемопередатчика (Serial Data Buffer)

Регистры специальных функций управляют работой блоков, входящих в микроконтроллер.

Регистры-защелки SFR параллельных портов P0...P3 - служат для ввода-вывода информации.

Две регистровые пары с именами TH0, TL0 и TH1, TL1 представляют собой регистры, двух программно-управляемых 16-битных таймеров-счетчиков.

Режимы таймеров-счетчиков задаются с использованием регистра TMOD, а управление ими осуществляется с помощью регистра TCON.

Для управления режимами энергопотребления микро-ЭВМ используется регистр PCON.

Регистры IP и IE управляют работой системы прерываний микро-ЭВМ, регистры SBUF и SCON — работой приемопередатчика последовательного порта.

Регистр-указатель стека SP в микро-ЭВМ рассматриваемого семейства — восьмибитный. Он может адресовать любую область внутренней памяти данных. В отличие от микропроцессора КР580ВМ80, МК семейства 8051 стек «растет вверх», т.е. перед выполнением команды PUSH или CALL содержимое SP инкрементируется, после чего производится запись информации в стек. Соответственно при извлечении информации из стека регистр SP декрементируется после извлечения информации. В процессе инициализации МК после сигнала сброса или при включении питающего

напряжения в SP заносится код 07H. Это означает, что первый элемент стека будет располагаться в ячейке памяти с адресом 08H.

Регистр-указатель данных DPTR чаще всего используют для фиксации 16-битного адреса в операциях обращения к внешней памяти программ и данных. С точки зрения программиста он может выступать как в виде одного 16-битного регистра, так и в виде двух независимых регистров DPL и DPH

Аккумулятор (ACC) является источником операнда и местом фиксации результата при выполнении арифметических, логических операций и ряда операций передачи данных. Кроме того, только с использованием аккумулятора могут быть выполнены операции сдвигов, проверка на нуль, формирование флага паритета и т.п.

При выполнении многих команд в АЛУ формируется ряд признаков операции (флагов), которые фиксируются в регистре PSW.

Регистр B используется как источник и как приемник при операциях умножения и деления, обращение к нему, как к регистру SFR, производится аналогично аккумулятору.

2. Особенности подключения к МК внешней памяти и периферийных устройств

МК MCS-51 может работать с внешней памятью данных емкостью до 64 Кбайт, построенной на одной или нескольких микросхемах статической памяти.

Микросхемы RAM с байтовой организацией HM1-65642-883 и HM116A120 имеют размер 8Кх8 и 2Кх8.

Такие микросхемы имеют 8 двунаправленных выводов данных (D0-D7), 11 или 13 адресных входов (A0-A10 или A0-A12). Вход WE (W) определяет характер обращения: если на нем установлена 1, то осуществляется чтение из выбранной ячейки, при WE=0 в ячейку будет записана информация. Вход CS (E1,E2) активирует микросхему памяти: когда на входе CS установлена 1, она выключена, при CS=0 допускается

любое обращение к памяти. Нулевой сигнал на входе OE (G) разрешает работу выходной шины данных микросхемы.

Микросхемы ПЗУ ROM (32Kx8) 27C256-12L, EPROM (8Kx8, 16Kx8) 27C128-17P, 27C64E350-883. Такие микросхемы в рабочем режиме допускают только считывание информации. Выводы микросхем аналогичны микросхемам RAM, кроме вывода PGM, отвечающего за программирование.

В МК MCS-51 существует 4 многофункциональных 8-битовых порта ввода/вывода P0, P1, P2, P3, предназначенных для обмена информацией с разными внешними устройствами и для выполнения специализированных функций, таких как подключение внешней памяти программ, данных, программирование внутреннего ПЗУ и др.

Каждый порт может адресоваться как побайтно, так и побитно, по конкретным физическим адресам. При подключении к МК внешней памяти через порт P0 выводится младший байт адреса, а также передается и принимается в микроконтроллер байт данных (в мультиплексированном режиме). В 1 и 2 тактах машинного цикла при обращении к внешней памяти на линиях P0 активизируется адресная информация A0-A7 при высоком уровне сигнала ALE, а затем на этих же линиях появляется сигнал D0-D7 (при низком уровне сигнала ALE). Для фиксации байта адреса в течение всего машинного цикла используются регистры-защелки, например, 74LS373N, информация в которых фиксируется по спаду сигнала на его входе ENG.

Через порт P2 выводится старший байт адреса (разряды A8-A15) внешней памяти программ и данных. Для каждого из битов порта P3 имеется ряд альтернативных функций. Сигналы стробов записи (WR#) и чтения (RD#) внешней памяти формируются на линиях P3.6 и P3.7 соответственно. Альтернативные функции всех портов реализуются только в том случае, если в соответствующий разряд фиксатора-защелки порта записана логическая 1, иначе на соответствующем выводе будет присутствовать 0.

Каждый вывод портов P0-P3 может использоваться как вход или выход независимо от других. Для настройки линии порта на ввод информации необходимо в соответствующий разряд порта записать 1, а для использования в качестве выхода – 0. При системном сбросе в регистрах защелках всех портов устанавливается значение FFh.

3. Порядок выполнения практической работы

3.1. Создание проекта

Создайте новый проект и разместите на рабочем поле МК MCS-51, микросхему памяти емкостью 2 Кбайта, регистр-защелку (например, 4037ВР), землю и питание. Соедините элементы между собой как показано на рис. 2.

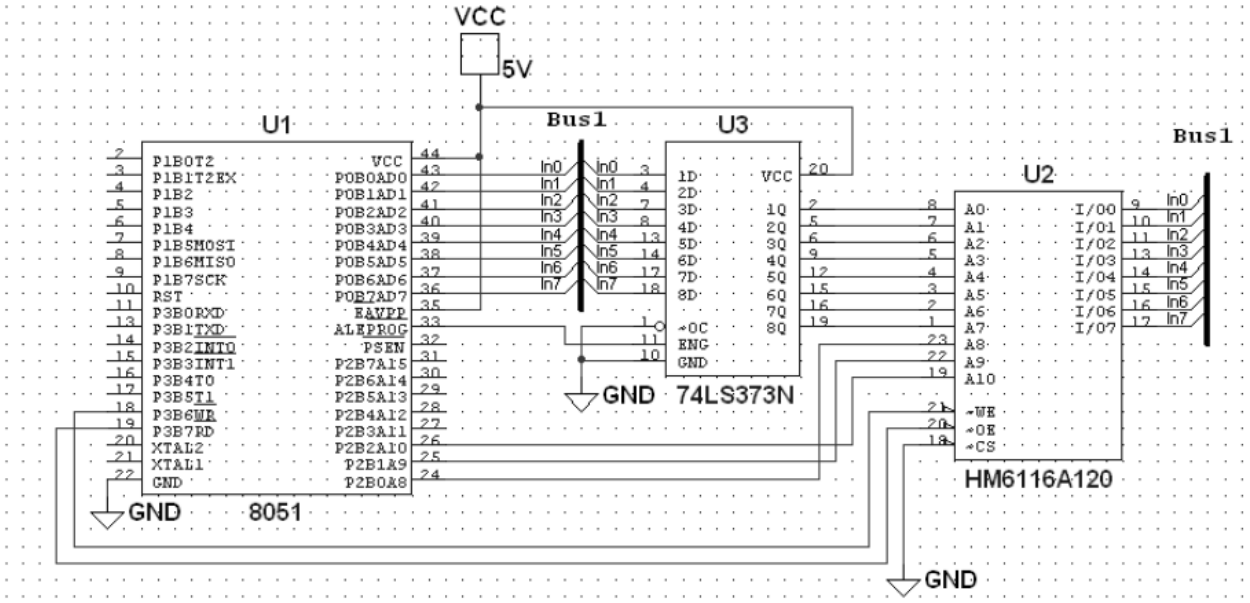

Рис. 2. Схема подключения памяти.

Всем соединениям присваивается автоматически номер по порядку, начиная с 1. Чтобы изменить нумерацию соединения или присвоить ему логическое имя, необходимо дважды кликнуть по соединительному проводнику.

В Multisim имеется функция автоматического соединения выводов между собой и с проводниками. При добавлении компонента к существующей сети соединений необходимо, чтобы выводы компонента

касались существующей сети, либо пометить компонент параллельно соединению.

Для размещения соединяющей шины в схемном проекте необходимо выбрать пункт меню Place-Bus или соответствующий значок на панели компонентов.

Для соединения элемента с шиной в контекстном меню элемента, соединяемого с шиной, выбирается пункт Bus Vector Connect. Слева окна соединений представлены необходимые данные о компоненте (обозначение, положение выводов, выводы), справа – о шине (обозначение, шинные линии). Далее необходимо выбрать выводы компонента для подключения к шине, имя шины и сформировать вектора (линии шины). Multisim предлагает для этого два способа: ручной и автоматический.

Для формирования векторов вручную необходимо нажать кнопку Add buslines для задания векторов, появляется окно, в котором указывается префикс (метка), начальное значение вектора, инкремент и количество векторов. После нажатия на кнопку Ok полученные линии шины (вектора) отразятся в предыдущем окне. Количество линий шины должно быть эквивалентно выбранным для подключения к шине выводам. Далее необходимо выделить полученные линии шины и нажать кнопку Ok.

При автоматическом формировании векторов, они вводятся автоматически при нажатии кнопки Auto-assign.

Для объединения двух шин необходимо выбрать в контекстном меню одной из шин пункт Properties. В открывшемся окне нажать кнопку Merge. В следующем открывшемся окне слева указана первая шина, предназначенная для объединения, а справа – вторая. В нижнем поле задается имя объединенной шины, и ее вектора. Затем необходимо нажать кнопку Merge.

В свойствах МК на вкладке Value в поле Built-in External RAM необходимо указать объем подключенной внешней памяти.

3.2. Разработка программного файла

Программный файл отображается на соответствующей закладке Code, которую можно открыть, выбрав программный файл в окне разработки.

Пример программы тестирования внешней памяти, которая осуществляет проверку 255 байт внешней памяти, начиная с ячейки 00h, используя тестовый набор 055h.

Ассемблерный файл программы:

```
    $MOD51              ;подключение МК-51
    org 00h             ;начинаем программу с адреса 00h
    mov dptr,#00h           ;загрузить в регистр-указатель
число 00h
    mov r2,#0ffh        ;загрузить в регистр R2 число 0ffh
(счетчик цикла)
    mov r1,#055h        ;загрузить в регистр R1 число 55h
    test:
    mov a,r1            ;загрузить     аккумулятор     ACC
операндом из регистра R1
    movx @dptr,a        ;переслать в ячейку внешней памяти
XRAM содержимое аккумулятора ACC
    movx a,@dptr;           ;   считать   в   аккумулятор
содержимое текущей ячейки внешней памяти
    xrl a,#055h;            ;операция XOR считанного и
изначального операнда, если 0 в Акк, то ячейка работает
нормально
    jnz error               ;ошибка – выход из программы
    inc dptr            ;инкремент содержимого регистра
DPTR – переход к следующему адресу
    djnz r2,test            ;вычесть 1 из содержимого
регистра R2 и перейти по метке, если в ячейке не 0.
    error:
    END
```

Чтобы откомпилировать и отредактировать программный файл необходимо выбрать пункт Build. В окне сообщений на экране отражается информация об ошибках и предупреждениях. Если имеются ошибки, в окне сообщений указываются номера строк, в которых они находятся. Для отображения нумерации строк необходимо выбрать меню MCU – Show lines numbers.

Моделирование работы схемы производится через меню Simulate – Run, или через соответствующую кнопку на панели Моделирование. При этом открывается окно отладчика. Для пошагового моделирования используется пункт Step into в меню MCU.

При нажатии кнопки Pause моделирование приостанавливается и возможно посмотреть, на каком этапе находится симуляция. Кнопка Stop останавливает выполнение моделирования.

Просмотр памяти МК активируется через меню MCU-MCU 8051 U1 – Memory view, также можно использовать пункт MCU Windows, где во всплывающем окне можно отметить необходимые для работы окна атрибуты. При необходимости выполнения программы до определенной инструкции в Multisim предусмотрены точки останова, установить которые возможно через меню MCU – Toggle breakpoint. Кнопка Remove all breakpoints – удаляет все точки останова.

Посмотрите состояние памяти после выполнения программой инструкции djnz r2, test. В ACC должно быть записано число 00h, в R1 – 55h, в R2 – FEh, в первой ячейке внешней памяти XRAM – 55h.

Посмотрите состояние памяти после завершения моделирования.

Работа с С-файлом аналогична. Ниже приведен вариант тестирования внешней памяти с адресами от 400h до 7FFh.

```
    #include <8051.h>              //подключение
заголовочного файла
    void main()                 //главная  функция,  точка
входа в программу
    {
    int i;
    char xdata *ptr;            //указатель    на    адрес
ячейки памяти внешнего ОЗУ
    char test, nabor;
    nabor = 0x55;               //тестовый набор
    ptr = (char xdata *) 0x400;   //начальный      адрес
внешней памяти
    for(i=0; i<1024;i++)        //проверка    1024    ячеек
внешней памяти
    {
```

```
    *ptr=nabor;                 //в каждую ячейку памяти
посылается значение тестового набора
    test=*ptr;                  //переменная          test
принимает значение содержимого ячейки памяти
    if(test!=nabor)
    {
    break;                      //в случае ошибки цикл
завершается
    }
    ptr++;
    }
    }
```

3.3. Подключение осциллографа

Разместите на рабочей области виртуальны осциллограф. Символ осциллографа имеет два канала: Channel A и Channel B. Подсоедините «+» канала А к выводу ALE/PROG МК, «-» - к земле. На панели прибора необходимо указать деление временной шкалы, равное одному машинному циклу, то есть 1 мкс. Выберите по оси у цену деления для канала А равной 5В. После запуска моделирования посмотрите процесс генерации сигнала ALE, дважды за машинный цикл.

3.4. Самостоятельное задание

Измените проект таким образом, чтобы произвести тестирование области памяти 1 Кбайт внешнего ОЗУ емкостью N Кбайт, используя тестовый набор XX, начиная с адреса ZZZ. Включить светоиндикатор, если записанный и считанный из ячейки памяти наборы не совпадают.

Таблица 1. Варианты заданий

Параметры	1	2	3	4	5	6	7	8
N, Кбайт	8	2	8	8	8	2	2	8
XX	0AAh	55h	33h	0FFh	55h	0AAh	33h	0FFh
ZZZ	800h	100h	0C00h	1000h	1400h	400h	240h	1200h

Продолжение таблицы 8

Параметры	9	10	11	12	13	14	15	16
N, Кбайт	8	2	8	8	8	8	8	8
XX	55h	AAh	0FFh	33h	0FFh	0AAh	33h	55h
ZZZ	0A20h	300h	0800h	540h	1800h	680h	1000h	0B00h

Параметры	17	18	19	20	21	22	23	24	25
N, Кбайт	2	2	8	8	2	2	2	2	8
XX	0AAh	33h	AAh	0FFh	55h	0AAh	33h	33h	0AAh
ZZZ	800h	140h	00h	8C0h	1000h	64h	156h	32h	1600h

Приложение 1

(справочное)

Система команд MCS-51

Обозначения:

Rn (n = 0, 1,..., 7) – регистр общего назначения в выбранном банке регистров;

@Ri(i= 0, 1) – регистр общего назначения в выбранном банке регистров, используемый в качестве регистра косвенного адреса;

ad – адрес прямоадресуемого байта;

ads – адрес прямо адресуемого байта-источника;

add – адрес прямо адресуемого байта-получателя;

ad11 – 11-разрядный абсолютный адрес перехода;

ad16 – 16-разрядный абсолютный адрес перехода;

rel – относительный адрес перехода;

#d – непосредственный операнд;

#d16 – непосредственный операнд (2 байта);

bit – адрес прямо адресуемого бита;

/bit – инверсия прямо адресуемого бита;

A - аккумулятор;

PC – счетчик команд;

DPTR – регистр указатель данных;

( ) – содержимое ячейки памяти или регистра.

1. Команды пересылки данных

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Пересылка в аккумулятор из регистра (n=0÷7)	MOV A, Rn	11101rrr	1	1	1	(A) ← (Rn)
Пересылка в аккумулятор прямоадресуемого байта	MOV A, ad	11100101	3	2	1	(A) ← (ad)
Пересылка в аккумулятор байта из РПД (i=0,1)	MOV A, @Ri	1110011i	1	1	1	(A) ← ((Ri))
Загрузка в аккумулятор константы	MOV A, #d	01110100	2	2	1	(A) ← #d
Пересылка в регистр из аккумулятора	MOV Rn, A	11111rrr	1	1	1	(Rn) ← (A)
Пересылка в регистр прямоадресуемого байта	MOV Rn, ad	10101rrr	3	2	2	(Rn) ← (ad)
Загрузка в регистр константы	MOV Rn, #d	01111rrr	2	2	1	(Rn) ← #d
Пересылка по прямому адресу аккумулятора	MOV ad, A	11110101	3	2	1	(ad) ← (A)
Пересылка по прямому адресу регистра	MOV ad, Rn	10001rrr	3	2	2	(ad) ← (Rn)
Пересылка прямоадресуемого байта по прямому адресу	MOV add, ads	10000101	9	3	2	(add) ← (ads)
Пересылка байта из РПД по прямому адресу	MOV ad, @Ri	1000011i	3	2	2	(ad) ← ((Ri))
Пересылка по прямому адресу константы	MOV ad, #d	01110101	7	3	2	(ad) ← #d
Пересылка в РПД из аккумулятора	MOV @Ri, A	1111011i	1	1	1	((Ri)) ← (A)
Пересылка в РПД прямоадресуемого байта	MOV @Ri, ad	0110011i	3	2	2	((Ri)) ← (ad)
Пересылка в РПД константы	MOV @Ri, #d	0111011i	2	2	1	((Ri)) ← #d
Загрузка указателя данных	MOV DPTR, #d16	10010000	13	3	2	(DPTR) ← #d16
Пересылка в аккумулятор байта из ПП	MOVC A, @A+DPTR	10010011	1	1	2	← ((A) +(DPTR))
Пересылка в аккумулятор байта из ПП	MOVC A, @A+PC	10000011	1	1	2	(PC) ← (PC)+1, (A) ← ((A)+(PC))
Пересылка в аккумулятор байта из ВПД	MOVX A, @Ri	1110001i	1	1	2	(A) ← ((Ri))
Пересылка в аккумулятор байта из расширенной ВПД	MOVX A,@DPTR	11100000	1	1	2	(A) ← ((DPTR))
Пересылка в ВПД из аккумулятора	MOVX @Ri, A	1111001i	1	1	2	((Ri)) ← (A)
Пересылка в расширенную ВПД из аккумулятора	MOVX @DPTR,A	11110000	1	1	2	((DPTR)) ← (A)
Загрузка в стек	PUSH ad	11000000	3	2	2	(SP) ← (SP) + 1, ((SP)) ← (ad)
Извлечение из стека	POP ad	11010000	3	2	2	(ad) ← (SP), (SP) ← (SP) - 1
Обмен аккумулятора с регистром	XCH A, Rn	11001rrr	1	1	1	(A) ↔ (Rn)
Обмен аккумулятора с прямоадресуемым байтом	XCH A, ad	11000101	3	2	1	(A) ↔ (ad)
Обмен аккумулятора с байтом из РПД	XCH A, @Ri	1100011i	1	1	1	(A) ↔ ((Ri))
Обмен младших тетрад аккумулятора и байта РПД	XCHD A, @Ri	1101011i	1	1	1	$(A_{0...3}) \leftrightarrow ((Ri)_{0...3})$

2. Арифметические операции

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Сложение аккумулятора с регистром (n=0÷7)	ADD A, Rn	00l01rrr	1	1	1	(A) ← (A) + (Rn)
Сложение аккумулятора с прямоадресуемым байтом	ADD A, ad	00100101	3	2	1	(A) ← (A) + (ad)
Сложение аккумулятора с байтом из РПД (i = 0,1)	ADD A, @Ri	0010011i	1	1	1	(A) ← (A) + ((Ri))
Сложение аккумулятора с константой	ADD A, #d	00100100	2	2	1	(A) ← (A) + #d
Сложение аккумулятора с регистром и переносом	ADDC A, Rn	00111rrr	1	1	1	(A) ← (A) + (Rn) + (C)
Сложение аккумулятора с прямоадресуемым байтом и переносом	ADDC A, ad	00110101	3	2	1	(A) ← (A) + (ad) + (C)
Сложение аккумулятора с байтом из РПД и переносом	ADDC A, @Ri	0011011i	1	1	1	(A) ← (A) + ((Ri)) + (C)
Сложение аккумулятора с константой и переносом	ADDC A, #d	00110100	2	2	1	(A) ← (A) + # d + (C)
Десятичная коррекция аккумулятора	DA A	11010100	1	1	1	Если $(A_{0...3})$>9 или ((AC)=1),то$(A_{0...3})$← $(A_{0...3})$ + 6, затем если $(A_{4...7})$ >9 или ((C)=1), то $(A_{4...7})$ ← $(A_{4...7})$ + 6
Вычитание из аккумулятора регистра и заёма	SUBB A, Rn	10011rrr	1	1	1	(A) ← (A) - (C) - (Rn)
Вычитание из аккумулятора прямоадресуемого байта и заема	SUBB A, ad	10010101	3	2	1	(A) ← (A) - (C) - ((ad))
Вычитание из аккумулятора байта РПД и заема	SUBB A, @Ri	1001011i	1	1	1	(A) ← (A) - (C) - ((Ri))
Вычитание из аккумулятора константы и заема	SUBB A, d	10010100	2	2	1	(A) ← (A) - (C) - #d
Инкремент аккумулятора	INC A	00000100	1	1	1	(A) ← (A) + 1
Инкремент регистра	INC Rn	00001rrr	1	1	1	(Rn) ← (Rn) + 1
Инкремент прямоадресуемого байта	INC ad	00000101	3	2	1	(ad) ← (ad) + 1
Инкремент байта в РПД	INC @Ri	0000011i	1	1	1	((Ri)) ← ((Ri)) + 1
Инкремент указателя данных	INC DPTR	10100011	1	1	2	(DPTR) ← (DPTR) + 1
Декремент аккумулятора	DEC A	00010100	1	1	1	(A) ← (A) – 1
Декремент регистра	DEC Rn	00011rrr	1	1	1	(Rn) ← (Rn) – 1
Декремент прямоадресуемого байта	DEC ad	00010101	3	2	1	(ad) ← (ad) – 1
Декремент байта в РПД	DEC @Ri	0001011i	1	1	1	((Ri)) ← ((Ri)) - 1
Умножение аккумулятора на регистр B	MUL AB	10100100	1	1	4	(B)(A) ← (A)*(B)
Деление аккумулятора на регистр B	DIV AB	10000100	1	1	4	(B).(A) ← (A)/(B)

3. Логические операции

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Логическое И аккумулятора и регистра	ANL A, Rn	01011rrr	1	1	1	(A) ← (A) AND (Rn)
Логическое И аккумулятора и прямоадресуемого байта	ANL A, ad	01010101	3	2	1	(A) ← (A) AND (ad)
Логическое И аккумулятора и байта из РПД	ANL A, @Ri	0101011i	1	1	1	(A) ← (A) AND ((Ri))
Логическое И аккумулятора и константы	ANL A, #d	01010100	2	2	1	(A) ← (A) AND #d
Логическое И прямоадресуемого байта и аккумулятора	ANL ad, A	01010010	3	2	1	(ad) ← (ad) AND (A)
Логическое И прямоадресуемого байта и константы	ANL ad, #d	01010011	7	3	2	(ad) ← (ad) AND #d
Логическое ИЛИ аккумулятора и регистра	ORL A, Rn	01001rrr	1	1	1	(A) ← (A) OR (Rn)
Логическое ИЛИ аккумулятора и прямоадресуемого байта	ORL A, ad	01000101	3	2	1	(A) ← (A) OR (ad)
Логическое ИЛИ аккумулятора и байта из РПД	ORL A, @Ri	0100011i	1	1	1	(A) ← (A) OR ((Ri))
Логическое ИЛИ аккумулятора и константы	ORL A, #d	01000100	2	2	1	(A) ← (A) OR #d
Логическое ИЛИ прямоадресуемого байта и аккумулятора	ORL ad, A	01000010	3	2	1	(ad) ← (ad) OR (A)
Логическое ИЛИ прямоадресуемого байта и константы	ORL ad, #d	01000011	7	3	2	(ad) ← (ad) OR #d
Исключающее ИЛИ аккумулятора и регистра	XRL A, Rn	01101rrr	1	1	1	(A) ← (A) XOR (Rn)
Исключающее ИЛИ аккумулятора и прямоадресуемого байта	XRL A, ad	01100101	3	2	1	(A) ← (A) XOR (ad)
Исключающее ИЛИ аккумулятора и байта из РПД	XRL A, @Ri	0110011i	1	1	1	(A) ← (A) XOR ((Ri))
Исключающее ИЛИ аккумулятора и константы	XRL A, #d	01100100	2	2	1	(A) ← (A) XOR #d
Исключающее ИЛИ прямоадресуемого байта и аккумулятора	XRL ad, A	01100010	3	2	1	(ad) ← (ad) XOR (A)
Исключающее ИЛИ прямоадресуемого байта и константы	XRL ad, #d	01100011	7	3	2	(ad) ← (ad) XOR #d
Сброс аккумулятора	CLR A	11100100	1	1	1	(A) ← 0
Инверсия аккумулятора	CPL A	11110100	1	1	1	(A) ← NOT(A)
Сдвиг аккумулятора влево циклический	RL A	00100011	1	1	1	$(A_{n+1}) \leftarrow (A_n)$, n=0÷6, $(A_0) \leftarrow (A_7)$
Сдвиг аккумулятора влево через перенос	RLC A	00110011	1	1	1	$(A_{n+1}) \leftarrow (A_n)$, n=0÷6 $(A_0) \leftarrow$ (C), (C) $\leftarrow (A_7)$
Сдвиг аккумулятора вправо циклический	RR A	00000011	1	1	1	$(A_n) \leftarrow (A_{n+1})$, n=0÷6, $(A_7) \leftarrow (A_0)$
Сдвиг аккумулятора вправо через перенос	RRC A	00010011	1	1	1	$(A_n) \leftarrow (A_{n+1})$, n=0÷6 $(A_7) \leftarrow$ (C), (C) $\leftarrow (A_0)$
Обмен местами тетрад в аккумуляторе	SWAP A	11000100	1	1	1	$(A_{0...3}) \leftrightarrow (A_{4...7})$

4. Команды операций над битами

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Сброс переноса	CLR C	11000011	1	1	1	(C) ← 0
Сброс бита	CLR bit	11000010	4	2	1	(b) ← 0
Установка переноса	SETB C	11010011	1	1	1	(C) ← 1
Установка бита	SETB bit	11010010	4	2	1	(b) ← 1
Инверсия переноса	CPL C	10110011	1	1	1	(C) ← NOT(C)
Инверсия бита	CPL bit	10110010	4	2	1	(b) ← NOT(b)
Логическое И бита и переноса	ANL C, bit	10000010	4	2	2	(C) ← (C) AND (b)
Логическое И инверсии бита и переноса	ANL C, /bit	10110000	4	2	2	(C) ← (C) AND (NOT(b))
Логическое ИЛИ бита и переноса	ORL C, bit	01110010	4	2	2	(C) ← (C) OR (b)
Логическое ИЛИ инверсии бита и переноса	ORL C, /bit	10100000	4	2	2	(C) ← (C) OR (NOT(b))
Пересылка бита в перенос	MOV C, bit	10100010	4	2	1	(C) ← (b)
Пересылка переноса в бит	MOV bit, C	10010010	4	2	2	(b) ← (C)

5. Команды передачи управления

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Длинный переход в полном объеме ПП	LJMP ad16	00000010	12	3	2	(PC) ← ad16
Абсолютный переход внутри страницы в 2 Кб	AJMP ad11	$a_{10}a_9a_800001$	6	2	2	(PC) ← (PC) + 2, (PC_{0-10}) ← ad11
Короткий относительный переход внутри страницы в 256 байт	SJMP rel	10000000	5	2	2	(PC) ← (PC) + 2, (PC) ← (PC) +rel
Косвенный относительный переход	JMP @A+DPTR	01110011	1	1	2	(PC) ← (A) + (DPTR)
Переход, если аккумулятор равен нулю	JZ rel	01100000	5	2	2	(PC)←(PC)+2, если (A)=0, то (PC)←(PC)+rel
Переход, если аккумулятор не равен нулю	JNZ rel	01110000	5	2	2	(PC)←(PC)+2, если (A)≠0, то (PC)←(PC)+rel
Переход, если перенос равен единице	JC rel	01000000	5	2	2	(PC)←(PC)+2, если (C)=1, то (PC)←(PC)+rel
Переход, если перенос равен нулю	JNC rel	01010000	5	2	2	(PC)←(PC)+2, если (C)=0, то (PC)←(PC)+rel
Переход, если бит равен единице	JB bit, rel	00100000	11	3	2	(PC)←(PC)+3, если (b)=l, то (PC)←(PC)+rel
Переход, если бит равен нулю	JNB bit, rel	00110000	11	3	2	(PC)←(PC)+3, если (b)=0, то (PC)←(PC)+rel
Переход, если бит установлен, с последующим сбросом бита	JBC bit, rel	00010000	11	3	2	(PC) ← (PC) + 3, если (b)=1, то (b) ← 0 и (PC)← (PC) + rel
Декремент регистра и переход, если не нуль	DJNZ Rn, rel	11011rrr	5	2	2	(PC) ← (PC) + 2, (Rn) ← (Rn) - 1, если (Rn) ≠ 0, то (PC) ← (PC) + rel
Декремент прямоадресуемого байта и переход, если не нуль	DJNZ ad, rel	11010101	8	3	2	(PC) ← (PC) + 2, (ad) ← (ad) - 1, если (ad) ≠ 0, то (PC) ← (PC) + rel
Сравнение аккумулятора с прямоадресуемым байтом и переход, если не равно	CJNE A, ad, rel	10110101	8	3	2	(PC) ← (PC) + 3,если (A) ≠ (ad), то (PC) ← (PC) + rel,если (A) < (ad), то (C) ← 1, иначе (C) ← 0
Сравнение аккумулятора с константой и переход, если не равно	CJNE A, #d, rel	10110100	10	3	2	(PC) ← (PC) + 3,если (A) ≠ #d, то (PC) ← (PC) + rel,если (A) < #d, то (C) ← 1, иначе (C) ← 0
Сравнение регистра с константой и переход, если не равно	CJNE Rn, #d, rel	10111rrr	10	3	2	(PC) ← (PC) + 3,если (Rn) ≠ #d, то (PC) ← (PC) + rel,если (Rn) < #d, то (C) ← 1, иначе (C) ← 0
Сравнение байта в РПД с константой и	CJNE @Ri,#d,rel	1011011i	10	3	2	(PC) ← (PC) + 3,если ((Ri)) ≠ #d, то
переход, если не равно						(PC) ← (PC) + rel,если ((Ri)) < #d, то (C) ← 1, иначе (C) ← 0
Длинный вызов подпрограммы	LCALL adl6	00010010	12	3	2	(PC) ← (PC) + 3, (SP) ← (SP) +1, ((SP)) ← ($PC_{0...7}$), (SP) ← (SP) + 1, ((SP)) ← ($PC_{8...15}$), (PC) ← ad16
Абсолютный вызов подпрограммы в пределах страницы в 2 Кб	ACALL ad11	$a_{10}a_9a_810001$	6	2	2	(PC) ← (PC) + 2, (SP) ← (SP) + 1, ((SP)) ← ($PC_{0...7}$), (SP) ← (SP) + 1, ((SP)) ← ($PC_{8...15}$), (PC_{0-10}) ← ad11
Возврат из подпрограммы	RET	00100010	1	1	2	($PC_{8...15}$) ← ((SP)), (SP) ← (SP) - 1, ($PC_{0...7}$) ← ((SP)), (SP) ← (SP) – 1
Возврат из подпрограммы обработки прерывания	RETI	00110010	1	1	2	($PC_{8...15}$) ← ((SP)), (SP) ← (SP) - 1, ($PC_{0...7}$) ← ((SP)), (SP) ← (SP) – 1
Пустая операция	NOP	00000000	1	1	1	(PC) ← (PC) + 1