Рек. к 5 от 29.04.15.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

УТВЕРЖДАЮ
Директор ИК

_____А.А. Захарова
«30» 04 2015 г.

В.Л. Ким, Е.А. Мыцко, М.Л. Иванов

МИКРОПРОЦЕССОРЫ И МИКРОКОНТРОЛЕРЫ

Методические указания к выполнению лабораторных работ
по курсу «Микропроцессоры и микроконтроллеры» для студентов IV курса, обучающихся по направлению 09.03.01
«Информатика и вычислительная техника»

Издательство
Томского политехнического университета
2015

УДК 004.31(075.8)
ББК 32.973.26-04я73
К40

Ким В.Л.

К40 Микропроцессоры и микроконтроллеры : методические указания к выполнению лабораторных работ по курсу «Микропроцессоры и микроконтроллеры » для студентов IV курса, обучающихся по направлению 09.03.01 «Информатика и вычислительная техника» / В.Л. Ким, Е.А. Мыцко, М.Л. Иванов ; Томский политехнический университет. – Томск : Изд-во Томского политехнического университета, 2015. – 31 с.

УДК 004.31(075.8)
ББК 32.973.26-04я73

Методические указания рассмотрены и рекомендованы к изданию методическим семинаром кафедры вычислительной техники ИК
« 23 » апреля 2015 г.

Зав. кафедрой вычислительной техники
Доктор технических наук *Н.Г.Марков*

Рецензент

Доктор технических наук, зав. кафедрой электроники и автоматики физических установок ТПУ
А.Г. Горюнов

Введение

Данные учебно-методические указания предназначены для выполнения лабораторных работ по курсу «Микропроцессоры и микроконтроллеры» студентов направления 09.03.01 «Информатика и вычислительная техника». Данные указания содержат пять лабораторных работ, каждая из которых посвящена изучению и разработке программ для учебно-лабораторного стенда SDK-1.1. Для каждой работы приведены необходимые теоретические сведения и задачи для практической реализации.

Темы для лабораторных работ подобраны таким образом, дать практическое применение знаниям, полученным в теоретической части курса «Микропроцессоры и микроконтроллеры».

Лабораторная работа 1.

Изучение лабораторного комплекса SDK – 1.1

Цель работы: изучение методов архитектуры и методов проектирования систем на базе микропроцессоров, однокристальных микроЭВМ, встраиваемых контроллеров, систем сбора данных, периферийных блоков вычислительных систем, подсистем ввода-вывода встраиваемых систем.

Методические указания

1. Структура аппаратной части

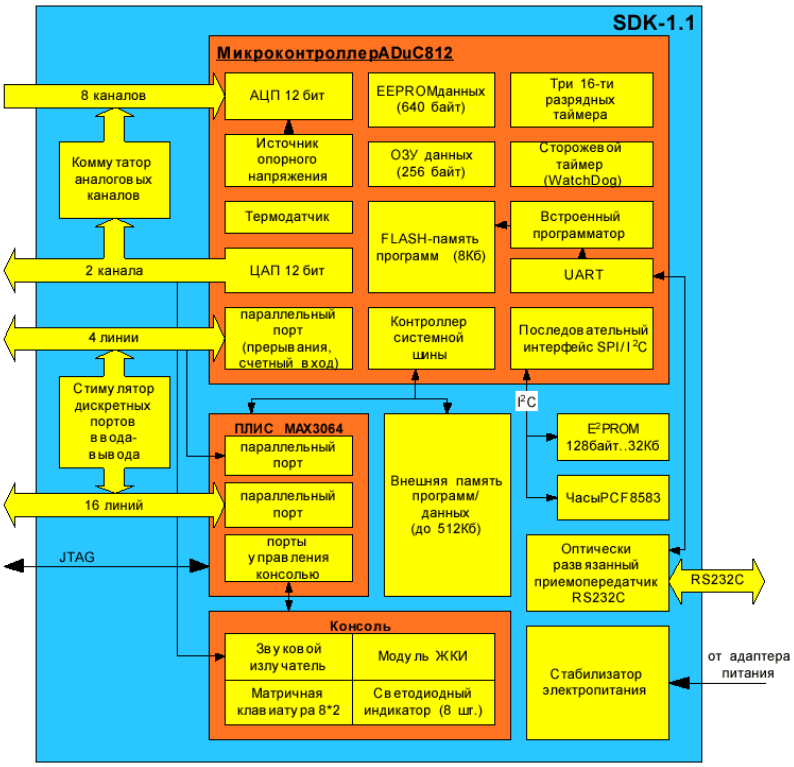


Рис. 1. Структура аппаратной части учебного стенда SDK-1.1

В состав учебного стенда SDK-1.1 входят (рис. 1):

- Микроконтроллер ADuC812BS;
- Внешняя E^2PROM объёмом 256 байт;
- Клавиатура AK1604A-WWB фирмы ACCORD;

- Жидкокристаллический индикатор (ЖКИ) WH1602B-YGK-CP фирмы Winstar Display;
- Часы реального времени PCF8583;
- 128K внешней SRAM с возможностью расширения до 512K;
- Набор сигнальных светодиодов (8 шт.).

2. Распределение памяти в SDK-1.1

Карта распределения памяти в SDK-1.1 представлена на рис. 2.

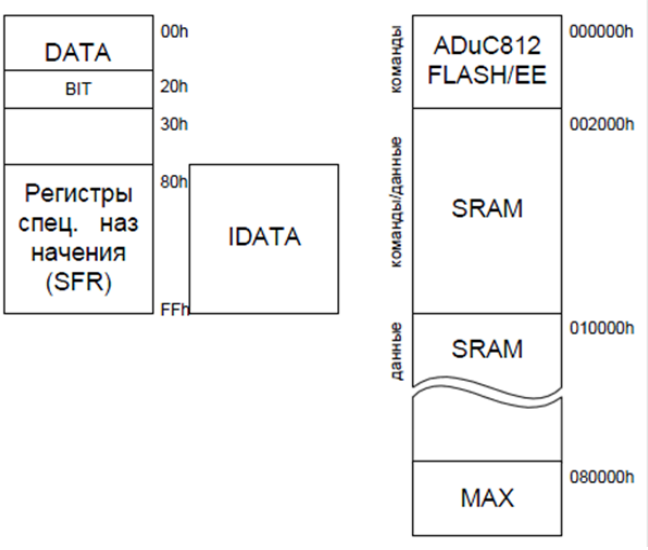


Рис. 2. Карта памяти SDK-1.1

Стандартная для архитектуры 8051 структура внутренней памяти представлена четырьмя банками по 8 регистров общего назначения (диапазоны адресов 00h-07h, 08h-0Fh, 10h-17h, 1Fh-20h), битовым сегментом (20h-2Fh), свободным участком 30h-7Fh, областью размещения SFR (регистров специального назначения) 80h-FFh, доступной при прямой адресации, и свободной областью 80h-FFh, доступной при косвенной адресации.

Внешняя память SDK-1.1 разбита на следующие области:

- **ADuC812 Flash/EE**. Это область, в которой располагается таблица векторов и резидентный загрузчик файлов в формате HEX в память SRAM;
- **SRAM.** Статическая память SRAM в SDK-1.1 имеет страничную организацию (максимум 8 страниц по 64 К) и условно разделяется на две области. Первая занимает младшие 64 Кбайт (страница 0) и доступна для выборки команд микроконтроллером ADuC812. Таким образом,

программы могут располагаться только в этих младших 64 К адресного пространства. Остальные страницы доступны только для размещения данных. Для адресации ячейки памяти определённой страницы необходимо записать номер страницы в регистр специального назначения DPP ADuC812 (адрес 84h);

- **MAX**. В младших адресах восьмой страницы адресного пространства (080000h-080007h) располагается 8 ячеек-регистров ПЛИС MAX8064 (MAX8128). Эта область предназначена для взаимодействия с периферийными устройствами.

3. Регистры ПЛИС

В табл. 1 представлен список регистров ПЛИС.

Табл. 1. Перечень регистров ПЛИС

Адрес	Регистр	Доступ	Назначение
080000H	KB	R/W	Регистр клавиатуры.
080001H	DATA_IND	R/W	Регистр шины данных ЖКИ.
080002H	EXT_LO	R/W	Регистр данных параллельного порта (разряды 0..7).
080003H	EXT_HI	R/W	Регистр данных параллельного порта (разряды 8..15).
080004H	ENA	W	Регистр управления портами ввода-вывода, звуком и сигналом INT0.
080006H	C_IND	W	Регистр управления ЖКИ.
080007H	SV	W	Регистр управления светодиодами.

В данной лабораторной работе понадобится регистр управления светодиодами SV. Его адрес 080007H. Значение после сброса 00000000b. Поля регистра и их назначения представлены в табл. 2 и 3.

Табл. 2. Регистр управления светодиодами SV

7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W
D7	D6	D5	D4	D3	D2	D1	D0

Табл. 3. Назначение битов регистра SV

Биты	Поле	Описание
0..7	D0..D7	Биты управления светодиодами. Подача логической единицы зажигает светодиоды.

4. Пример программирования

Рассмотрим примеры программирования под микроконтроллер. В данном курсе лабораторных работ программы для SDK-1.1 необходимо писать на языке ASM51. Полный перечень команд языка находится в приложении А.

Для начала рассмотрим программу, реализующую зажигание 1, 3, 5 и 7 светодиода (листинг 1).

```
DPP EQU 84h          ;подключаем дополнительную память
    ORG 2000h ;адрес расположения нашего кода
                     ;(адрес SRAM)
    LJMP main ;переход на метку main
main:
    MOV DPP,#08h     ;активизируем ПЛИС
    MOV DPTR,#07h    ;загружаем команду в регистр
                          ;управления светодиодами
    MOV A,#0AAh      ;значение 10101010 записываем
                     ;в аккумулятор
    MOVX @DPTR,A     ;выводим значение из аккумулятора на
                     ;светодиоды (записываем в адрес
                     ;регистра)
    SJMP $           ;зацикливаем программу
                     ;(совершает прыжок на эту же строку)
end                  ;конец
```

Листинг 1. Простая программа управления светодиодами

Далее, рассмотрим пример, в котором реализовано поочерёдное зажигание чётных и нечётных светодиодов с использованием программной задержки, видимой для человеческого взгляда (листинг 2).

```
DPP EQU 84h
    ORG 2000h
    LJMP main0
main0:
    MOV R1,# 0AAh    ;задаем значение R1 #0AAh
main:
    MOV DPP,#08h
    MOV DPTR,#07h
    MOV R2,#0FFh     ;задаем значение R2 #0FFh
    MOV R3,#0FFh     ;задаем значение R3 #0FFh
decry:
    NOP                ;пустая команда
    NOP                ;пустая команда
    NOP                ;пустая команда
    DJNZ R3,decry    ;отнимаем 1 от R3 до тех пор,
                     ;пока R3 не ноль и переходим на decrx
```

```
decrx:
     DEC R2          ;уменьшаем значение R2 на 1
     MOV R3,#0FFh    ;задаем значение R3 #0FFh
     CJNE R2,#0,decry    ;если R2!=0 переходим на decry
     MOV A,R1        ;из R1 загрузить в акк-р
     MOVX @DPTR,A ;выводим значение на светодиоды
     CPL A           ;инверсия A
     MOV R1,A        ;из R1 загрузить в акк-р
     LJMP main ;переходим на main
end
```

Листинг 2. Программная реализация задержки

Рассчитаем задержку, которая была получена в примере. Микроконтроллер ADuC812BS работает на частоте 11059200 Гц. Машинный цикл выполняется за 12 тактов. Следовательно, частота одного машинного цикла – 11059200 / 12 = 921600 Гц, что в микросекундах составляет 1 / 921600 = 1.085 мкс. Длительность операций: «NOP» – 1 машинный цикл, «DJNZ R3, decry» - 2 машинных цикла, «DEC R2» – 1 машинный цикл, «MOV R3, #0FFh» составляет 1 машинный цикл и «CJNE R2,#0,decry» - 2 машинных цикла. В программе реализован двойной цикл по $FF_{16}=255_{10}$ итераций. Следовательно, с учётом выполняемых команд, задержка составит: 255 * ((1+1+1+2) * 255 +1 + 1 + 2) = 326 145 машинных цикла, или 326 145 * 1.085 мкс = 0.35 с. Человеческий глаз различает мерцание с частотой менее 50 Гц, что эквивалентно 0.20 с. Это составляет 0.20 с./ 1.085 мкс = 184331 машинных циклов. Следовательно, полученная нами задержка будет видна для человеческого глаза.

Теперь, рассмотрим использование аппаратной задержки. Следующая программа выводит на диоды анимацию из таблицы, задержка реализуется с помощью таймеров микроконтроллера AduC812 (см. листинг 3). Для получения задержки в 500 мс нужно аппаратно, при помощи счётчика, реализовать задержку в 50 мс и запустить её в цикл на 10 раз.

```
DPP EQU 84h
          ORG 2000h
          LJMP main
          ORG 200bh   ;вектор - прерываний для таймера
          LJMP clock
          ORG 2100h
main:
          MOV DPP,#8h
          MOV DPTR,#7h
```

```
        MOV R1,#0
        MOV TH0,#HIGH(15535) ;вр.задер.=65535-15535=
        MOV TL0,#LOW(15535) ; =50000 мкс = 50 мс
        MOV TMOD,#00000001b
        MOV IE, #10000010b  ;маска прерываний
        SETB TR0         ;запуск таймера
        SJMP $
clock:
        MOV A,R1
        INC R1
        CJNE R1,#9, now
        MOV R1,#0
        RETI
now:
        MOV DPTR, #table1
        MOVC A, @A+DPTR
        MOV DPTR,#7h
        MOVX @DPTR,A
        CLR TR0        ;остановка таймера
        MOV TH0,#HIGH(15535)
        MOV TL0,#LOW(15535)
        SETB TR0
        RETI
table1:         ;таблица анимации
      DB 10000000b
      DB 01000000b
      DB 00100000b
      DB 00010000b
      DB 00001000b
      DB 00000100b
      DB 00000010b
      DB 00000001b
end
```

Листинг 3. Аппаратная реализация задержки

Таким образом, были рассмотрены простейшие примеры управления светодиодами, использование программной и аппаратной задержки, работа с таблицами.

Порядок выполнения работы

1) Изучить структуру учебного лабораторного комплекса SDK-1.1;
2) Организовать зажигание светодиодов в соответствии со своим вариантом (табл. 4) с использованием программной задержки. Время задержки вычисляется по формуле $T_{задерж}=0.5+0.1*№_{вар}$ (с).
3) Модифицировать программу с реализацией аппаратной задержки;

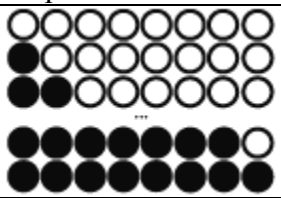
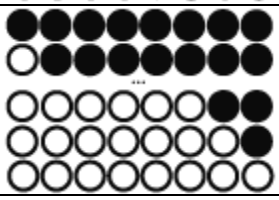
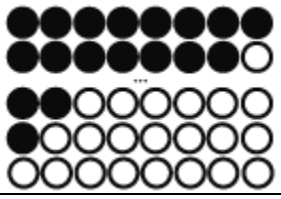
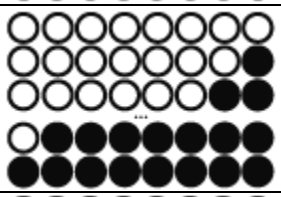
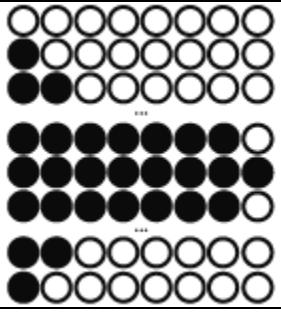
4) Написать отчёт о проделанной работе.

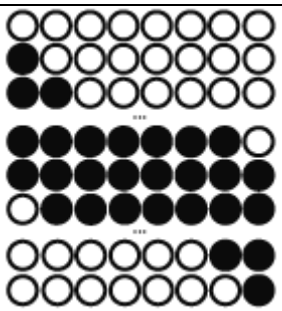
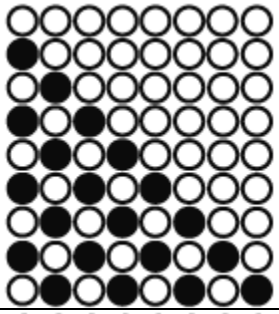
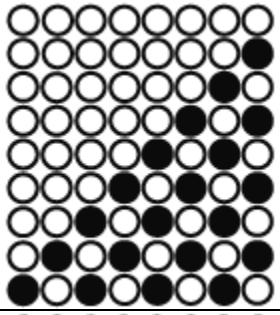
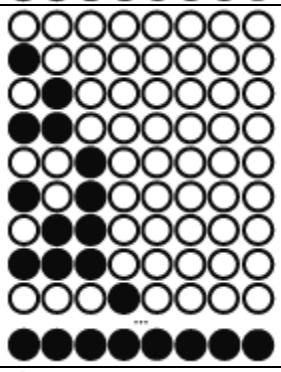
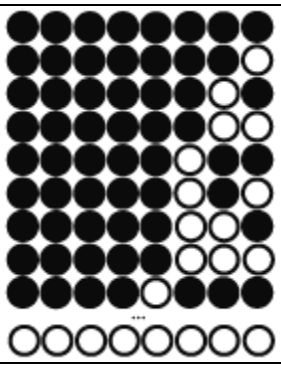
Содержание отчёта

Отчёт по лабораторной работе должен содержать следующее:

1) Цель работы;
2) Постановку задачи;
3) Листинг обеих программ;
4) Выводы.

Табл. 4. Варианты задания для лабораторной работы №1

Вариант №	Порядок зажигания
1	
2	
3	
4	
5	

6	
7	
8	
9	
10	

Лабораторная работа 2.
Матричная клавиатура

Цель работы: написать программу для макета SDK 1.1, которая позволяет проверить комбинацию нажатых клавиш с матричной клавиатуры, подавлять дребезг контактов, если он присутствует. Также программа должна сигнализировать на светодиодах об неправильно нажатых кнопках, основываясь на заданной комбинации.

Методические указания

1. Описание работы матричной клавиатуры.

В качестве устройств ввода информации, в МК-системах получили распространение цифровые, алфавитно-цифровые и специальные клавиатуры. По способам аппаратурной реализации различают два типа клавиатур: кодирующую и некодирующую. В клавиатурах первого типа аппаратным путем на выходе формируется код, соответствующий нажатой клавише. К такому типу можно отнести клавиатуру персонального компьютера. Такие клавиатуры в МК-системах применяются редко.

Значительно более широкое распространение получили некодирующие (матричные) клавиатуры, которые представляют собой простую матрицу двоичных переключателей (требуемой размерности), включенных на пересечении строк и колонок матрицы. Идентификация (кодирование) нажатой клавиши в таких клавиатурах выполняется программой. На рис.3 представлена структурная схема матричной клавиатуры 4x4, входящей в состав лабораторного комплекса SDK 1.1.

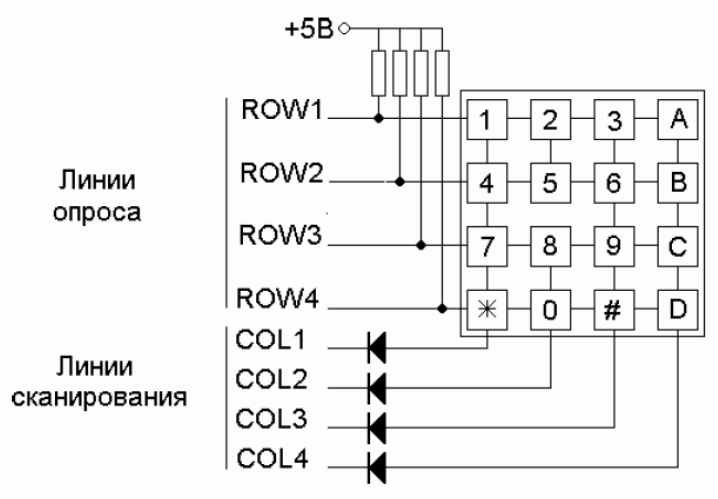


Рис. 3 Схема матричной клавиатуры

Сигналы COL1 – COL4 предназначены для сканирования клавиатуры (колонки матрицы). Сканирование производится посредством установки в «0» в одном из разрядов поля.

Сигналы ROW1 – ROW4 используются для считывания данных с клавиатурной матрицы (строки матрицы). Если не одна из кнопок в строке на нажата, то на всех линиях опроса (сигналы ROW1 – ROW4) высокий уровень (логическая «1»). Если кнопка нажата и на ее колонку подан «0» , то на линии ROW тоже появиться 0.

Доступ к колонкам и строкам организован как чтение/запись определенного байта внешней памяти - 4 бита соответствуют 4 колонкам, другие 4 бита – рядам. (См. табл. 5).

Адрес 080000H. Значение после сброса 00000000B.

Табл. 5. Доступ к колонкам и строкам регистра клавиатуры

7	6	5	4	3	2	1	0
чтение	чтение	чтение	чтение	запись	запись	запись	запись
ROW4 – ROW1				COL4 – COL1			

Табл. 6. Назначение битов регистра клавиатуры

Биты	Поле	Описание
0..3	COL	Поле предназначено для сканирования клавиатуры (колонки матрицы). Сканирование производится посредством записи логического «0» в один из разрядов поля.
4..7	ROW	Поле предназначено для считывания данных с клавиатурной

		матрицы (строки). Если ни одна из кнопок в строке не нажата, все биты поля ROW содержат логические «1». Если кнопка нажата и на ее колонку подан логический «0», то в поле ROW также появится логический «0».

2. Дребезг контактов.

При работе с любыми механическими переключателями необходимо помнить о таком явлении как «дребезг контактов». Это явление возникает при замыкании/размыкании механического контакта и выражается в виде серии импульсов различной длинны (рис. 4). Продолжительность дребезга контактов может достигать до 40…100 мс.

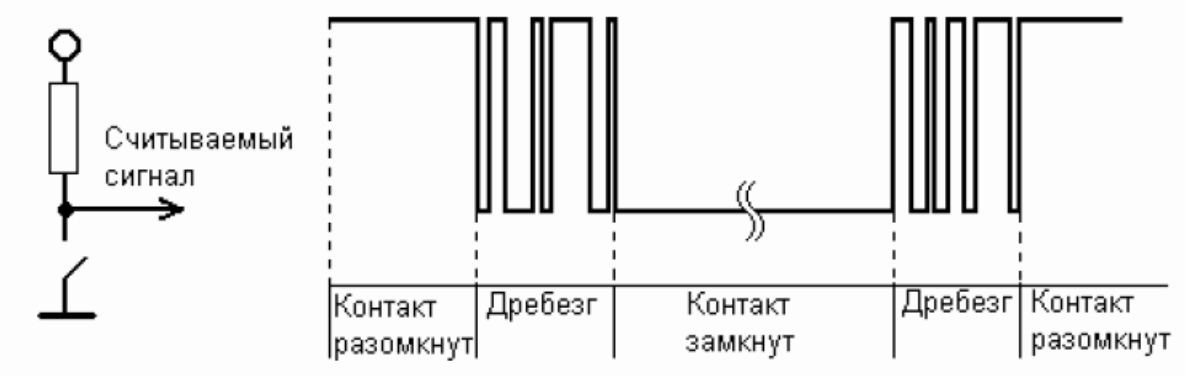


Рис.4 Дребезг контактов

Для предотвращения возможности считывания неверного значения сигнала, необходимо подавлять дребезг контактов. Это можно сделать либо аппаратно, используя специальные фильтрующие схемы, либо программно. Сущность программного метода заключается в многократном считывании значения сигнала через определенный промежуток времени и последующем сравнении полученных значений. При совпадении, - значение сигнала принимается, при несовпадении отвергается.

3. Пример программирования.

Рассмотрим пример программы, использующую матричную клавиатуру входящей в состав лабораторного комплекса SDK 1.1. При выполнении программы производится сканирование первого столбца клавиатуры, в зависимости от

нажатия/не нажатия кнопок в первом столбце будет определяться результирующий код, который выводится на светодиоды.

```
      DPP EQU 84h
      ORG 2000h
      LJMP main
      main:
            MOV DPP,#08
            MOV R1, #11111110b ;для скан. 1го столбца
      main1:
            MOV DPTR,#0
            MOV A,R1
            MOVX @DPTR,A ;сканирование клав.
            MOVX A,@DPTR ;получение  скан. кода
            MOV DPTR,#7h
            MOVX @DPTR,A
            LJMP main1
end
```

Листинг 4. Программа сканирования 1го столбца матр. клавиатуры

Как видно из программы значение «11111110» позволяет сканировать только первый столбец, для того что бы сканировать 2ой столбец надо в разряд выше записать 0, что будет соответствовать коду «11111101», также делается для 3 и 4 столбцов. Все скан-коды нажатых клавиш представлены в таблице 7.

Табл. 7. Скан-коды нажатых клавиш

Клавиша	Скан-код	
	Двоичная форма	Шестнадцатеричная форма
0	01111101	7D
1	11101110	EE
2	11101101	ED
3	11101011	EB
4	11011110	DE
5	11011101	DD
6	11011011	DB
7	10111110	BE
8	10111101	BD
9	10111011	BB
A	11100111	E7
B	11010111	D7
C	10110111	B7
D	01110111	77
*	01111110	7E
#	01111011	7B

Порядок выполнения работ

1) Изучить принцип работы матричной клавиатуры лабораторного комплекса SDK-1.1;
2) Написать программу, которая с помощью клавиатуры осуществляет проверку ввода комбинации символов в соответствии с вариантом (табл. 8);
3) Проверка кода должна происходить следующим образом:

   а) Перед началом ввода, должен гореть 8ой светодиод;

   б) Как только нажата одна из кнопок, светодиод потухает;

   в) После ввода комбинации, проверяется ее правильность: если код верный, то все светодиоды горят, если есть неверный символ, то определенные светодиоды моргают (1,2 светодиоды относятся к первому символу, 3,4 ко второму, 5,6 к третьему, 7,8 к четвертому соответственно)
4) Написать отчёт о проделанной работе.

Содержание отчёта

Отчёт по лабораторной работе должен содержать следующее:

1) Цель работы;
2) Постановку задачи;
3) Листинг программы;
4) Выводы.

Табл. 8. Варианты задания для лабораторной работы №2

Вариант №	Комбинация символов
1	7 0 9 B
2	3 9 7 1
3	D C 8 *
4	A * 6 C
5	2 B 3 #
6	0 5 4 A
7	8 4 D 1
8	5 B 7 C
9	1 4 8 D
10	C 9 A *

Лабораторная работа 3.

Жидкокристаллический индикатор

Цель работы: написать программу для макета SDK 1.1, которая позволяет вывести на жидкокристаллический индикатор ряд стандартных символов и собственно нарисованный символ, используя память данных ЖКИ DDRAM и SGRAM.

Методические указания

1. **Общее описание жидкокристаллического индикатора.**

Основные характеристики приведены в таблице 9, а внешний вид представлен на рисунке 5.

Рис. 5 Внешний вид ЖКИ

Табл. 9. Основные характеристики ЖКИ

Элемент	Размерность	Единицы
Число символов	16 символов * 2 строки	
Размер модуля	80.0 * 36.0 * 13.2(MAX)	мм
Область изображения	66.0 * 16.0	мм
Активная область	56.21 * 11.5	мм
Размер точки	0.56 * 0.66	мм
Расстояние между точками	0.60 * 0.70	мм
Размер символа	2.96 * 5.56	мм
Тип ЖКИ	STN, положительный, полупрозрачный, серый	
Производительность	1/16	
Направление луча	Стрелка показывает на 12 часов	
Тип подсветки	Желто-зеленая	

2. **Описание функций ЖКИ**

Модуль ЖКИ встроен в контроллер (БИС) и имеет два 8-битовых регистра: регистр команд (IR) и регистр данных (DR). Регистр команд хранит коды таких операций, как очистка дисплея, перемещение курсора, а также информацию об адресах памяти отображаемых данных (DDRAM) и генератора символов (CGRAM). В регистр команд можно только записывать информацию из микропроцессора. Регистр данных временно хранит данные, предназначенные

для записи или чтения из DDRAM или CGRAM. Когда адресная информация записывается в регистр команд, данные из DDRAM или CGRAM сохраняются в регистре данных. Эти два регистра можно выбрать с помощью регистрового переключателя (RS). Перечень команд приведен в таблице 10.

Табл. 10. Таблица команд ЖКИ

Команды (время выполн.)	**Код операции**										**Описание**
	RS	**R/W**	**DB7**	**DB6**	**DB5**	**DB4**	**DB3**	**DB2**	**DB1**	**DB0**	
Отчистка экрана (1.53 мс)	0	0	0	0	0	0	0	0	0	1	Запись “00H” в DDRAM и установка адреса DDRAM на “00H” из AC.
Возврат в начало строки (1.53 мс)	0	0	0	0	0	0	0	0	1	*	Установка адреса DDRAM на “00H” из AC и возврат курсора в начало строки, если он был смещен. Содержимое DDRAM не меняется.
Начальные установки (39 мс)	0	0	0	0	0	0	0	1	I/D	SH	Задает направление перемещения курсора и разрешает сдвиг сразу всех символов
Дисплей ON/OFF (39 мс)	0	0	0	0	0	0	1	D	C	B	Устанавливает / отключает биты, отвечающие за режим дисплея (D), отображение курсора (C), мерцание курсора (B).
Передвижение курсора по экрану (39 мс)	0	0	0	0	0	1	S/C	R/L	*	*	Установка бита движения курсора и смещения всех символов, указание направления смещения без изменения данных в DDRAM.
Функц. установки (39 мс)	0	0	0	0	1	DL	N	F	*	*	Установка длины данных (DL:8-бит/4-бита), количества строк на дисплее (N:2-строки или 1) и размера символов (F:5Ч11 точек/5Ч8 точек)
Установка адреса CGRAM (39 мс)	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Установка адреса CGRAM в счетчик адреса
Установка адреса DDRAM (39 мс)	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Установка адреса DDRAM в счетчик адреса.
Чтение флага занятости и адреса (0 мс)	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Прочитав флаг занятости, можно определить, занят ли контроллер выполнением внутренних операций. Также можно прочесть содержимое счетчика адреса.
Записать данные в	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Запись данных во внутреннюю память

память (43 мс)										(DDRAM/CGRAM).
Чтение данных из памяти (43 мс)	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Чтение данных из внутренней памяти (DDRAM/CGRAM)

3. Использование команд для работы с ЖКИ

Для работы с ЖКИ требуется выполнение последовательности следующих команд:

1) Работа с регистром шины данных ЖКИ, адрес 080001H (табл. 11):
   -Выбор типа памяти либо DDRAM, либо CGRAM.
2) Работа с регистром управления ЖКИ, адрес 080006H (табл. 13):
   -Установка данных в регистре команд (работа с командами);
   -Фиксация данных регистре команд (работа с командами).
3) Работа с регистром шины данных ЖКИ:
   -Выбор символа в таблице 15 и запись его кода в регистр.
4) Работа с регистром управления ЖКИ:
   -Установка данных в регистре команд (работа с данными);
   -Фиксация данных регистре команд (работа с данными).

Для выполнения каждой из команд требуется определенной время, поэтому после любых обращениях к регистрам ЖКИ следует осуществлять задержку. В примере программы задержка реализована тремя командами «nop».

Табл. 11. Регистр шины данных ЖКИ DATA_IND

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
D7	D6	D5	D4	D3	D2	D1	D0

Табл. 12. Назначение битов Регистра DATA_IND

Биты	Поле	Описание
0..7	D0..D7	Регистр DATA_IND позволяет устанавливать данные на шине данных ЖКИ и считывать их оттуда. Для организации взаимодействия с ЖКИ (формирования временных диаграмм чтения и записи) необходимо использование регистра C_IND.

Табл. 13. Регистр управления ЖКИ C_IND

7	6	5	4	3	2	1	0
-	-	-	-	-	W	W	W
-	-	-	-	-	RS	RW	E

Табл. 14. Назначение битов регистра C_IND

Биты	Поле	Описание
0	E	Бит управления входом «E» ЖКИ. Наличие положительного импульса на входе «E» позволяет зафиксировать данные на шине ЖКИ (данные, сигналы RW и RS к этому моменту должны быть уже установлены).
1	RW	Бит переключения шины данных ЖКИ на чтение или запись. 0 – запись, 1 – чтение.
2	RS	Бит переключения режимов команды/данные ЖКИ. 1 – данные, 0 – команды.

4. Пример программирования.

Рассмотрим пример работы программы, которая выводит на дисплей символ «$».

```
     DPP EQU 84h
ORG 2000h
LJMP begin
begin:
     MOV  DPP, #08h
;//////////выбор типа памяти/////////////////
     mov  A,#10000000b  ;#10000000b - для DDRAM/ #01000000b -
для CGRAM
     mov  dptr,#01h  ;регистр данных  ЖКИ
     movx @dptr,a    ;уст DDRAM  шаг 1 (определение с какой
памятью работать)
     lcall     sleep
     mov  dptr,#06h  ;регистр команд ЖКИ
     mov  A,#00000001b ; 00000XYZb, x:0-команда,1-
данные(x=0);
                                    ;y:0-режим записи,1-режим
чтения(y=0);
                                    ;z: смена 1 на 0-фиксация
установок(z=1)
     movx @dptr,a            ;уст  DDRAM  шаг 2 (установка
данных в регистре команд)
     lcall     sleep
     mov  A,#00000000b   ; x = 0 ,y = 0, z = 0
     movx @dptr,a       ;уст  DDRAM  шаг 3 (фиксация данных в
регистре команд)
     lcall     sleep
     ;///////////////работа с таблицей//////////////////
     mov     R2,#00100100b ; символ "$"
     mov  dptr,#01h
     mov      A, R2
     movx @dptr,a
     lcall     sleep
     mov  dptr,#06h
     mov  A,#00000101b  ; x=1 - работа с данными, y = 0, z =
1
     movx @dptr,a
     lcall     sleep
     mov  A,#00000100b  ; z = 0
     movx @dptr,a
     ;///////////////работа с таблицей//////////////////
     sjmp $
sleep:
nop
nop
nop
     ret
```

```
end
```

Листинг 5. Программа вывода символов «$» на ЖКИ

Табл. 15. Таблица ПЗУ знакогенератора

Ст. тетр. / Мл. тетр.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	CG RAM (1)															
0001	CG RAM (2)															
0010	CG RAM (3)															
0011	CG RAM (4)															
0100	CG RAM (5)															
0101	CG RAM (6)															
0110	CG RAM (7)															
0111	CG RAM (8)															
1000	CG RAM (1)															
1001	CG RAM (2)															
1010	CG RAM (3)															
1011	CG RAM (4)															
1100	CG RAM (5)															
1101	CG RAM (6)															
1110	CG RAM (7)															
1111	CG RAM (8)															

Содержание отчёта

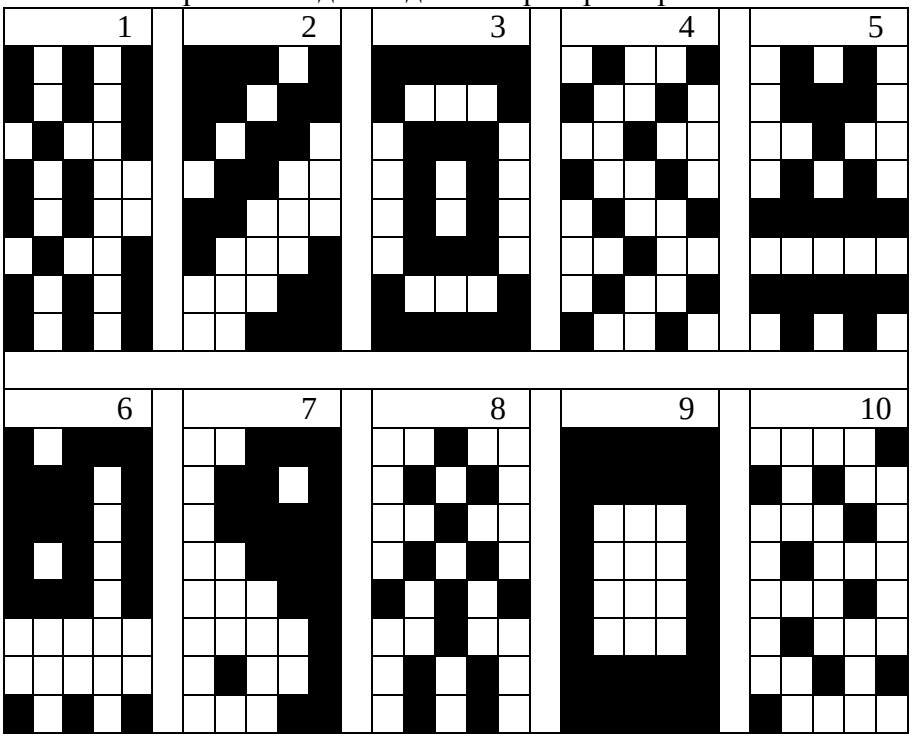
Отчёт по лабораторной работе должен содержать следующее:

5) Цель работы;
6) Постановку задачи;
7) Листинг программы;
8) Выводы.

Табл. 16. Варианты задания для лабораторной работы №3

Вариант №	Фамилия студента	Символ из таблицы 17
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Табл. 17. Варианты задания для лабораторной работы №3



Лабораторная работа 4.

Последовательный интерфейс RS-232. UART

Цель работы: написать программу для макета SDK 1.1, которая осуществляет обмен данных между ПК и последовательным портом микроконтроллера.

Методические указания

1. Описание работы последовательного порта в макете SDK 1.1.

Через универсальный асинхронный приемопередатчик UART (Universal Asynchronous Receiver-Transmitter) осуществляются прием и передача информации, представленной последовательным кодом (младшими битами вперед), в полном дуплексном режиме обмена. В состав приемопередатчика, называемого часто последовательным портом входят принимающий и передающий сдвигающие регистры, а также специальный буферный регистр (SBUF) приемопередатчика.

Кроме того, работой последовательного порта управляют два служебных регистра -

- Регистр управления/статуса приемопередатчика SCON
- Бит SMOD регистра управления мощностью PCON

Запись байта в буфер приводит к автоматической переписи байта в сдвигающий регистр передатчика и инициирует начало передачи байта. Наличие буферного регистра приемника позволяет совмещать операцию чтения ранее принятого байта с приемом очередного. Но если к моменту окончания приема байта предыдущий не был считан из SBUF, то он будет потерян.

Последовательный порт 8051 может работать в четырех различных режимах.

- **Режим 0.** Информация и передается, и принимается через вывод входа приемника (RXi TXi). Принимаются или передается 8 бит данных. Через вывод выхода передатчика (TXD; выдаются импульсы сдвига, которые сопровождают каждый бит. Частота передачи бита информации равна 1/12 частоты кварцевого резонатора
- **Режим 1.** В этом режиме передаются через вывод TXD или принимаются через RXD 10 бит информации: старт-бит (0), 8 бит данных и стоп-бит (1) при приеме информации в бит RB8 регистра управления/статуса приемопередатчика SCON заносятся стоп-бит Скорость приема/передачи — величина переменная и задается таймером.
- **Режим 2.** В этом режиме через вывод TXD передаются или через RXD принимаются 11 бит информации: старт-бит, 8 бит данных, программируемый девятый бит и стоп-бит. При передаче девятый бит данных может принимать значение 0 или 1 или, например, для повышения достоверности передачи путем контроля по четности в него может быть помещено значение признака паритета из слова состояния программы (PSW.0). При приеме девятый бит данных помещается в бит RB8 SCON, а стоп-бит, в отличие от режима 1, теряется. Частота приема/передачи выбирается программой и может быть равна либо 1/32, либо 1/64 частоты резонатора в зависимости от управляющего бита SMOD.

- **Режим 3.** совпадает с режимом 2 во всех деталях, за исключением частоты приема/передачи, которая является величиной переменной и задается таймером.

Во всех случаях передача инициализируется инструкцией, в которой данные перемещаются в SBUF. Прием инициализируется при обнаружении перепада из 1 в 0 на входе приемника. При этом в режиме 0 этот переход должен сопровождаться выполнением условий R1 = 0 и REN= 1 (см. табл. 8), а для остальных режимов - REN = 1.

3.5.Регистр управления/статуса приемопередатчика SCON.

Управление режимом работы приемопередатчика осуществляется через специальный регистр с символическим именем SCON. Этот регистр содержит не только управляющие биты, определяющие режим работы последовательного порта, но и девятый бит принимаемых или передаваемых данных (RB8 и TB8) и биты прерывания приемопередатчика (R1 и T1).

Прикладная программа путем загрузки в старшие биты регистра SCON двухбитного кода определяет режим работы приемопередатчика. Во всех четырех режимах работы передача инициализируется любой командой, в которой буферный регистр SBUF указан как получатель байта. Как уже отмечалось, прием в режиме 0 осуществляется при условии, что R1 = 0 и REN = 1, в остальных режимах - при условии, что REN = 1.

В бите TB8 программно устанавливается значение девятого бита данных, который будет передан 8 режиме 2 или 3. В бите RB8 в этих режимах фиксируется девятый принимаемый бит данных. В режиме 1 в бит RB8 заносится стоп-бит. В режиме 0 бит RB8 не используется.

Флаг прерывания передатчика TI устанавливается аппаратно в конце периода передачи стоп-бита во всех режимах. Соответствующая подпрограмма обслуживания прерывания должна сбрасывать бит TL.

Флаг прерывания приемника RI устанавливается аппаратно в конце периода приема восьмого бита данных в режиме 0 и в середине периода приема стоп-бита в режимах 1, 2 и 3. Подпрограмма обслуживания прерывания должна сбрасывать бит RI.

Табл. 18.Описание регистра SCON

<table>
<tr><th>Символ</th><th>Позиция</th><th>Имя и назначение</th></tr>
<tr><td>SM0</td><td>SCON.7</td><td rowspan="2">Биты управления режимом работы приемопередатчика. Устанавливаются/сбрасываются программно см. примечание 1
<table><tr><th>SM0</th><th>SM1</th><th>Режим работы приемопередатчика</th></tr><tr><td>0</td><td>0</td><td>Сдвигающий регистр расширения ввода/вывода</td></tr><tr><td>0</td><td>1</td><td>8 битовый приемопередатчик, изменяемая скорость передачи</td></tr><tr><td>1</td><td>0</td><td>9 битовый приемопередатчик. Фиксированная скорость передачи</td></tr><tr><td>1</td><td>1</td><td>9 битовый приемопередатчик, изменяемая скорость передачи</td></tr></table></td></tr>
<tr><td>SM1</td><td>SCON.6</td></tr>
<tr><td>SM2</td><td>SCON.5</td><td>Бит управления режимом приемопередатчика. Устанавливается программно для запрета приема сообщения, в котором девятый бит имеет значение 0</td></tr>
<tr><td>REN</td><td>SCON.4</td><td>Бит разрешения приема. Устанавливается/сбрасывается программно для разрешения/запрета приема последовательных данных</td></tr>
<tr><td>TB8</td><td>SCON. 3</td><td>Передача бита 8. Устанавливается/сбрасывается программно для задания девятого передаваемого бита в режиме 9-битового передатчика</td></tr>
<tr><td>RB8</td><td>SCON.2</td><td>Прием бита 8. Устанавливается/сбрасывается аппаратно для фиксации девятого принимаемого бита в режиме 9-битового приемника</td></tr>
<tr><td>TI</td><td>SCON. 1</td><td>Флаг прерывания передатчика. Устанавливается аппаратно при окончании передачи байта. Сбрасывается программно после обслуживания прерывания</td></tr>
<tr><td>RI</td><td>SCON.0</td><td>Флаг прерывания приемника. Устанавливается аппаратно при приеме байта. Сбрасывается программно после обслуживания прерывания</td></tr>
</table>

2. Пример программирования.

Рассмотрим пример работы программы, которая считывает данные из буфера, инвертирует их и отправляет обратно на ПК.

```
DPP EQU 84H
ORG 2000H
```

```
     LJMP MAIN
     ORG 2023H   ;вектор прерывания для посл. порта
     LJMP RS_INTERR
     ORG 2100H
     MAIN:
          MOV DPP, #8H
          MOV TMOD, #00100000b
          MOV TL1,  #0FDH ; установка значения для скорости
9600 бит/сек
          MOV TH1,  #0FDH
          SETB TR1
          MOV SCON, #01010000b ; установка режима посл. порта
          SETB EA ; снятие блокировки прерываний
          SETB ES
          SJMP $
     RS_INTERR:
          JB TI, CLEAR
          MOV  A,     SBUF
          MOV  DPTR,  #7H
          MOVX @DPTR, A
          CPL  A
          CLR RI        ; очистка флага приема
          MOV  SBUF, A
          LJMP EXIT
     CLEAR:
          CLR TI        ; очистка флага передачи
     EXIT:
          RETI
END
```

Листинг 5. Программа для обмена данным с ПК

Содержание отчёта

Отчёт по лабораторной работе должен содержать следующее:

9) Цель работы;
10) Постановку задачи;
11) Листинг программы;
12) Выводы.

Задание: Передать последовательно **два числа** с ПК на макет и выполнить с ними следующие **операции**:

№	Задание
1	Инкрементировать первое число и сложить со вторым, умноженным на 2
2	Декрементировать первое число и вычесть из него инвертированное второе
3	Сдвинуть первое число влево на 1 бит и умножить на второе, с обменянными тетраэдрами.
4	Сдвинуть первое число вправо на 1 бит и разделить на второе инкрементированное
5	Сделать инверсию первого числа и операцию «логическое И» со вторым, сложенным с константой 5.
6	В обоих числах обменять местами тетраэдры и выполнить операцию «логическое ИЛИ»
7	Умножить первое число на 2 и выполнить операцию «исключающее ИЛИ» со вторым числом, умноженным на 3
8	В первом числе обменять местами тетраэдры и сложить со декрементированным вторым.
9	Инвертировать оба числа и сложить
10	В первом числе обменять местами тетраэдры, во втором сделать инверсию и выполнить между ними операцию «исключающее ИЛИ»

Приложение А. Перечень команд микроконтроллера 8051

Таблица.7.*Команды передачи 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	$(A_{0...3}) \leftrightarrow ((Ri)_{0...3})$

Таблица.8.*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)

Таблица.9.*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	$(A_{0...3}) \leftrightarrow (A_{4...7})$

Таблица.10.*Операции с 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)

Таблица.11.*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)=I, 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, re	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,re	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

Учебное издание

КИМ Валерий Львович
МЫЦКО Евгений Алексеевич
ИВАНОВ Максим Леонидович

МИКРОПРОЦЕССОРЫ И МИКРОКОНТРОЛЛЕРЫ

Методические указания к выполнению лабораторных работ по курсу «Микропроцессоры и микроконтроллеры» для студентов IV курса, обучающихся по направлению 09.03.01 «Информатика и вычислительная техника»

Отпечатано в Издательстве ТПУ в полном соответствии с качеством предоставленного оригинал-макета

Подписано к печати __.__.2015. Формат 60x84/16. Бумага «Снегурочка».
Печать XEROX. Усл. печ. л. __. Уч.-изд. л. __,__.
Заказ ___-__. Тираж 20 экз.

Национальный исследовательский Томский политехнический университет
Система менеджмента качества
Издательства Томского политехнического университета
сертифицирована в соответствии с требованиями ISO 9001:2008

ИЗДАТЕЛЬСТВО ТПУ. 634050, г. Томск, пр. Ленина, 30
Тел./факс: 8(3822)56-35-35, www.tpu.ru