Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Оренбургский государственный университет»

Кафедра вычислительной техники и защиты информации

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ

Методические указания

Составитель:
Е.В. Бурькова

Рекомендовано к изданию редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего образования «Оренбургский государственный университет» для обучающихся по образовательной программе высшего образования по направлению подготовки 09.03.01 Информатика и вычислительная техника

Оренбург
2020

УДК 681.3(07)
ББК 32.973.26-04 я 73
О64

Рецензент – кандидат педагогических наук А.А. Рычкова

О64 **Организация ЭВМ и систем**: методические указания / составитель Е.В. Бурькова; – Оренбургский гос. ун-т. – Оренбург: ОГУ, 2020.– 72 с.

Методические указания содержат теоретические сведения и задания к практическим работам по дисциплине «Организация ЭВМ и систем».

Методические указания предназначены для обучающихся по образовательной программе высшего образования по направлению подготовки 09.03.01 Информатика и вычислительная техника.

УДК 681.3(07)
ББК 32.973.26-04 я 73

Содержание

Введение

Учебная дисциплина «Организация ЭВМ и систем» относится к базовой части дисциплин и предназначения для обучающихся направления подготовки 09.03.01 «Информатика и вычислительная техника». Основная цель дисциплины «Организация электронно-вычислительных машин и систем» заключается в формировании знаний основ организации, принципов построения и функционирования аппаратных и программных средств ЭВМ, умений применять эти знания при решении профессиональных задач для объектов профессиональной деятельности.

Задачи:

– изучить основы организации аппаратных и программных средств ЭВМ, принципы функционирования ЭВМ и их составных частей;

– сформировать умение выбирать оптимальные конфигурации вычислительных систем для решения конкретных практических задач;

– обучить студентов использовать программные средства при решении задач профессиональной деятельности.

Данные методические указания содержат теоретические сведения, задания к лабораторным работам по дисциплине «Организация ЭВМ и систем» и контрольные вопросы для самопроверки усвоения материала.

Методические указания содержат 9 лабораторных работ в соответствии с утвержденной рабочей программой данной дисциплины и охватывают все разделы. Для выполнения лабораторных работ рекомендуется изучить теоретический материал, представленный в методических указаниях, а также в источниках, перечисленных в списке литературы.

1 Практическая работа 1. Изучение архитектуры микроконтроллера 8051

Цель:

— изучение структурной схемы микроконтроллера 8051;
— изучение основных регистров.

1.1 Краткие теоретические сведения

МК 8051 представляет сбой законченное устройство с большим объемом встроенной памяти программ и данных, портами ввода/вывода и возможностью подключения внешней памяти. Типовые характеристики:

- тактовая частота 24 МГц;
- командный цикл 12 тактов;
- объем памяти программ 4 Кб;
- объем памяти данных 128 байт;
- число линий ввода/вывода – 32;
- два 8/16-разрядных таймера;
- множество внутренних и внешних источников прерываний;
- программируемый последовательный порт;
- интерфейс с внешней памятью объемом до 128 Кб.

МК 8051 имеет Гарвардскую архитектуру, то есть память программ и память данных являются самостоятельными и независимыми друг от друга устройствами.

Микроконтроллер выполнен на основе высокоуровневой n-МОП технологии. Через четыре программируемых параллельных порта ввода/вывода и один последовательный порт микроконтроллер взаимодействует с внешними устройствами. Структурная схема микроконтроллера 8051 приведена на рисунке 1.1.

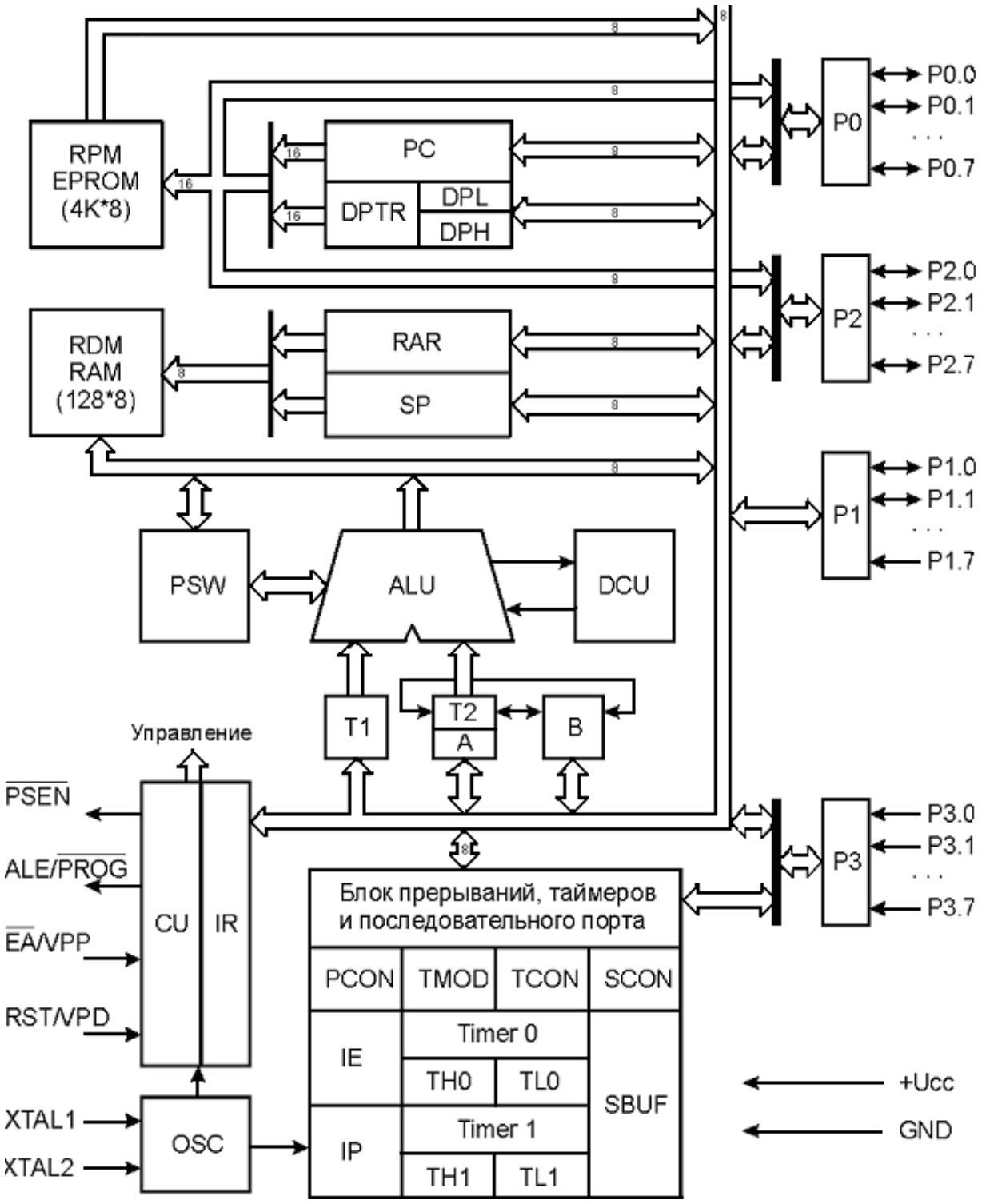

Рисунок 1.1 – Структурная схема микроконтроллера 8051

Основу структурной схемы образует внутренняя двунаправленная 8-битная шина, которая связывает между собой основные узлы и устройства микроконтроллера: резидентную память программ (RPM), резидентную память

данных (RDM), арифметико-логическое устройство (ALU), блок регистров специальных функций, устройство управления (CU) и порты ввода/вывода (P0-P3). 8-битное арифметико-логическое устройство (ALU) может выполнять арифметические операции сложения, вычитания, умножения и деления; логические операции И, ИЛИ, исключающее ИЛИ, а также операции циклического сдвига, сброса, инвертирования и т.п. К входам подключены программно-недоступные регистры T1 и T2, предназначенные для временного хранения операндов, схема десятичной коррекции (DCU) и схема формирования признаков результата операции (PSW).

Важной особенностью ALU является его способность оперировать не только байтами, но и битами. Отдельные программно-доступные биты могут быть установлены, сброшены, инвертированы, переданы, проверены и использованы в логических операциях. Эта способность достаточно важна, поскольку для управления объектами часто применяются алгоритмы, содержащие операции над входными и выходными булевыми переменными, реализация которых средствами обычных микропроцессоров сопряжена с определенными трудностями.

Таким образом, ALU может оперировать четырьмя типами информационных объектов: булевыми (1 бит), цифровыми (4 бита), байтными (8 бит) и адресными (16 бит). В ALU выполняется 51 различная операция пересылки или преобразования этих данных. Так как используется 11 режимов адресации (7 для данных и 4 для адресов), то путем комбинирования операции и режима адресации базовое число команд 111 расширяется до 255 из 256 возможных при однобайтном коде операции.

Условное графическое обозначение микроконтроллера 8051 приведено на рисунке 1.2. Показаны внешние выводы микросхемы микроконтроллера. Далее приводится описание назначения всех внешних выводов микроконтроллера.

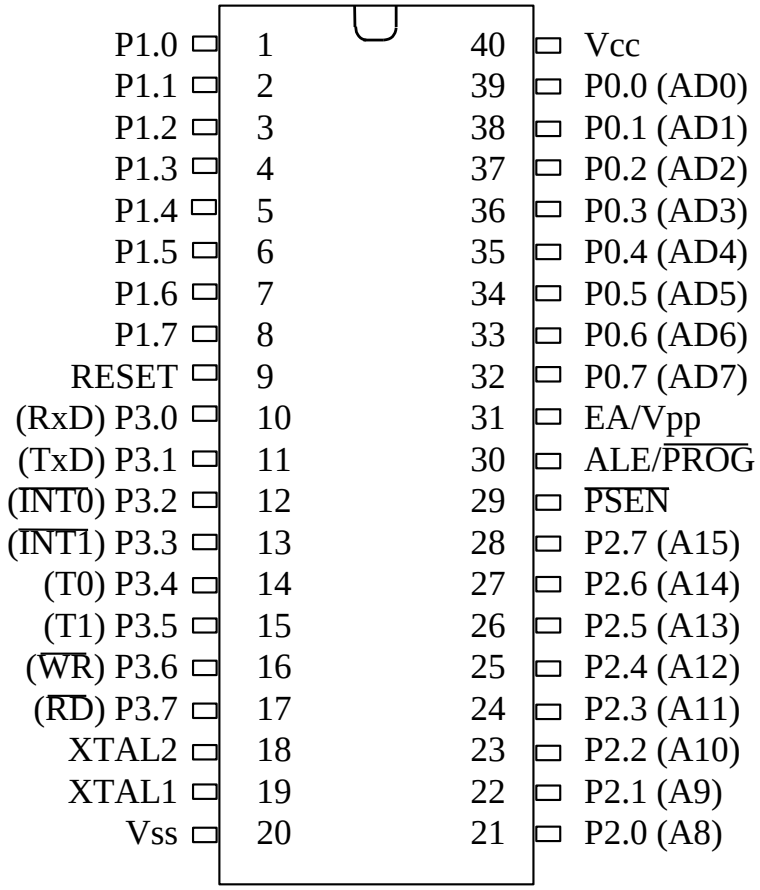

Рисунок 1.2 – Условное графическое обозначение микроконтроллера 8051

Назначение выводов микроконтроллера 8051.

U_{ss} — потенциал общего провода («земля»);

U_{cc} — основное напряжение литания +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 память; уровень 0 на этом входе заставляет микроконтроллер выполнять программу только внешнее ПЗУ; игнорируя внутреннее (если последнее имеется);

P1 — восьмибитный двунаправленный 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 двунаправленный 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 последней). Выводы порта используются при программировании 8751 для ввода в микроконтроллер старших разрядов адреса:

P3 — восьмибитный двунаправленный 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 — восьмибитный 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.

Назначение регистров микроконтроллера 8051.

Регистры разделены на две группы: регистры общего назначения (РОН) и регистры специальных функций. 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** управляют 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, у микро-ЭВМ 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 микро-ЭВМ 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. В распоряжении пользователя имеются 8 регистров общего назначения R0–R7 одного из четырёх возможных банков. 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;

— при выполнении ряда команд в арифметико-логическом устройстве (АЛУ) формируются признаки операций — флаги, которые фиксируются в регистре **PSW**.

Регистр флагов (PSW).

Перечень флагов, их символические имена и условия формирования приведены в таблице 1.1.

Таблица 1.1 – Регистр флагов Intel 8051

<table>
<tr><th>Символ</th><th>Позиция</th><th>Имя и назначение</th></tr>
<tr><td>P</td><td>PSW.0</td><td>Флаг паритета. Устанавливается и сбрасывается аппаратурно в каждом цикле команды и фиксирует нечетное/четное число единичных бит в аккумуляторе</td></tr>
<tr><td>-</td><td>PSW.1</td><td>Не используется</td></tr>
<tr><td>OV</td><td>PSW.2</td><td>Флаг переполнения. Устанавливается и сбрасывается аппаратурно при выполнении арифметических операций</td></tr>
<tr><td>RS0 - RS1</td><td>PSW.3 - PSW.4</td><td>Биты выбора используемого банка регистров. Могут быть изменены программным путем
<table>
<tr><th>RS0</th><th>RS1</th><th>Банк</th><th>Границы адресов ОЗУ</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>00H - 07H</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>08H - 0FH</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>10H - 17H</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>18H - 1FH</td></tr>
</table></td></tr>
<tr><td>F0</td><td>PSW.5</td><td>Флаг пользователя. Может быть установлен, сброшен или проверен программой пользователя</td></tr>
<tr><td>AC</td><td>PSW.6</td><td>Флаг вспомогательного переноса. Устанавливается и сбрасывается только аппаратными средствами при выполнении команд сложения и вычитания и сигнализирует о переносе или заеме в бите 3 аккумулятора</td></tr>
<tr><td>C</td><td>PSW.7</td><td>Флаг переноса. Устанавливается и сбрасывается как аппаратурно, так и программным путем</td></tr>
</table>

Наиболее активным флагом PSW является флаг переноса, который принимает участие и модифицируется в процессе выполнения множества

операций, включая сложение, вычитание и сдвиги. Кроме того, флаг переноса (C) выполняет функции «булева аккумулятора» в командах, манипулирующих с битами.

Флаг переполнения (OV) фиксирует арифметическое переполнение при операциях над целыми числами со знаком и делает возможным использование арифметики в дополнительных кодах. ALU не управляет флагами селекции банка регистров (RS0, RS1), их значение полностью определяется прикладной программой и используется для выбора одного из четырёх регистровых банков.

В микропроцессорах, архитектура которых опирается на аккумулятор, большинство команд работают с ним, используя неявную адресацию. В Intel 8051 дело обстоит иначе. Хотя процессор имеет в своей основе аккумулятор, он может выполнять множество команд и без его участия. Например, данные могут быть переданы из любой ячейки памяти в любой регистр, любой регистр может быть загружен непосредственным операндом и т.д. Многие логические операции могут быть выполнены без участия аккумулятора. Кроме того, переменные могут быть инкрементированы, декрементированы и проверены без использования аккумулятора. Флаги и управляющие биты могут быть проверены и изменены аналогично.

Принстонская и Гарвардская архитектуры.

В 60-х годах 20 века правительство Соединенных Штатов дало задание Гарвардскому и Принстонскому университетам разработать архитектуру компьютера для военно-морской артиллерии. Принстонский университет разработал ЭВМ, которая имела общую память для хранения программ и данных. Такая архитектура компьютеров больше известна как архитектура Фон-Неймана по имени научного руководителя этой разработки. Структура ЭВМ с Принстонской архитектурой показана на рисунке 1.3.

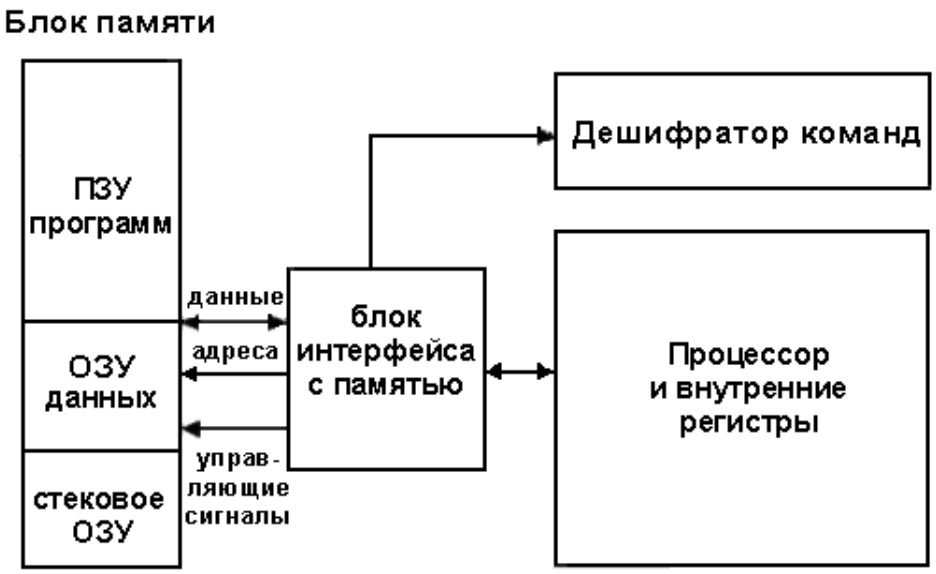

Рисунок 1.3 - Структура ЭВМ с Принстонской архитектурой

В этой архитектуре блок интерфейса с памятью выполняет арбитраж запросов к памяти, обеспечивая выборку команд, чтение и запись данных, размещаемых в памяти или внутренних регистрах. Может показаться, что блок интерфейса является наиболее узким местом между процессором и памятью, так как одновременно с данными требуется выбирать из памяти очередную команду. Однако во многих процессорах с Принстонской архитектурой эта проблема решается путем выборки следующей команды во время выполнения предыдущей. Такая операция называется предварительной выборкой («предвыборка»), и она реализуется в большинстве процессоров с такой архитектурой

Гарвардский университет представил разработку компьютера, в котором для хранения программ, данных и стека использовались отдельные банки памяти (рисунок 1.4). Принстонская архитектура выиграла соревнование, так как она больше соответствовала уровню технологии того времени. Использование общей памяти оказалось более предпочтительным из-за ненадежности ламповой электроники (это было до широкого распространения транзисторов) - при этом возникало меньше отказов.

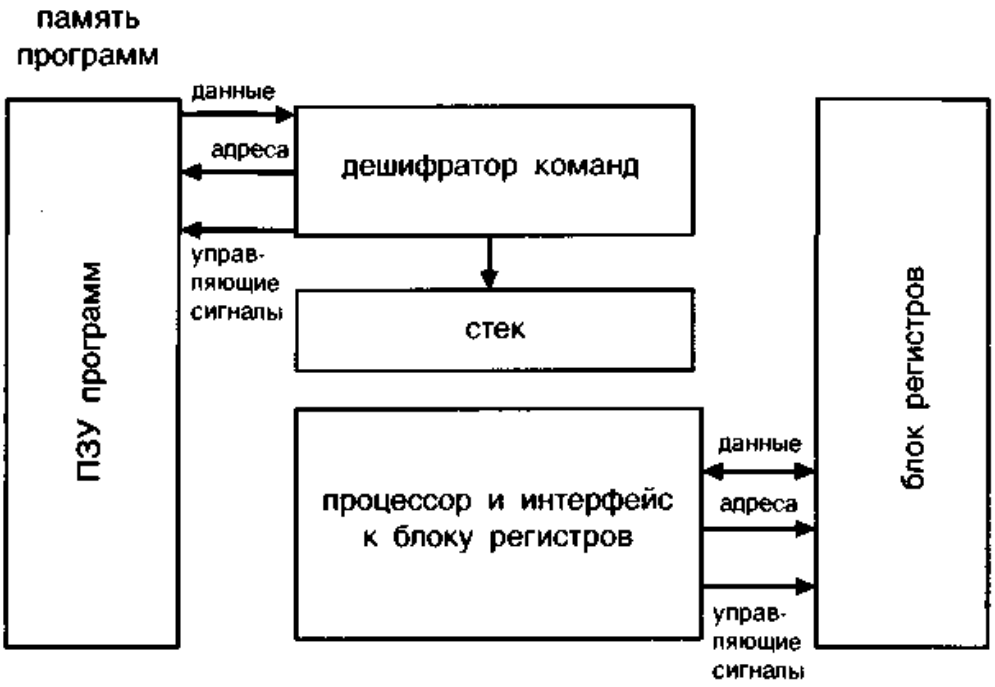

Рисунок 1.4 - Структура компьютера с Гарвардской архитектурой

Гарвардская архитектура почти не использовалась до конца 70-х годов, когда производители микроконтроллеров поняли, что эта архитектура дает преимущества устройствам, которые они разрабатывали.

Основным преимуществом архитектуры Фон Неймана является то, что она упрощает устройство микропроцессора, так как реализует обращение только к одной общей памяти. Для микропроцессоров самым важным является то, что содержимое ОЗУ (RAM Random Access Memory) может быть использовано как для хранения данных, так и для хранения программ. В некоторых приложениях программе необходимо иметь доступ к содержимому стека. Все это предоставляет большую гибкость для разработчика программного обеспечения в области операционных систем реального времени.

Гарвардская архитектура выполняет команды за меньшее количество тактов, чем архитектура Фон Неймана. Это обусловлено тем, что в Гарвардской архитектуре больше возможностей для реализации параллельных операций. Выборка следующей команды может происходить одновременно с выполнением предыдущей команды, и нет необходимости останавливать процессор на время

выборки команды. В Гарвардской архитектуре, обеспечивающей более высокую степень параллелизма операций, выполнение текущей операции может совмещаться с выборкой следующей команды. Команда также выполняется за два цикла, но выборка очередной команды производится одновременно с выполнением предыдущей. Таким образом, команда выполняется всего за один цикл (во время чтения следующей команды). Этот метод реализации операций («параллелизм») позволяет командам выполняться за одинаковое число тактов, что дает возможность более просто определить время выполнения циклов и критических участков программы. Это обстоятельство является особенно важным при выборе микроконтроллера для приложений, где требуется строгое обеспечение заданного времени выполнения. Следует отметить, что такие общие способы сравнения производительности не следует использовать для всех процессоров и микроконтроллеров, в которых реализуются эти две архитектуры. Сравнение лучше проводить применительно к конкретному приложению. Различные архитектуры и устройства имеют свои специфические особенности, которые позволяют наилучшим образом реализовать те или иные приложения. В некоторых случаях конкретное приложение может быть выполнено только с использованием определенной архитектуры и специфических особенностей микроконтроллера.

1.2 Задание

1.2.1 Изучить структурную схему микроконтроллера Intel 8051.

1.2.2 Изучить назначение всех регистров микроконтроллера Intel 8051.

1.2.3 В программной среде построить электрическую схему подключения микроконтроллера к напряжению питания и тактовому генератору. Схема подключения микроконтроллера приведена на рисунке 1.5.

1.2.4 Вывести на поле схемы окно с регистрами микроконтроллера.

1.2.5 Вывести на поле схемы окна резидентной памяти данных и программ.

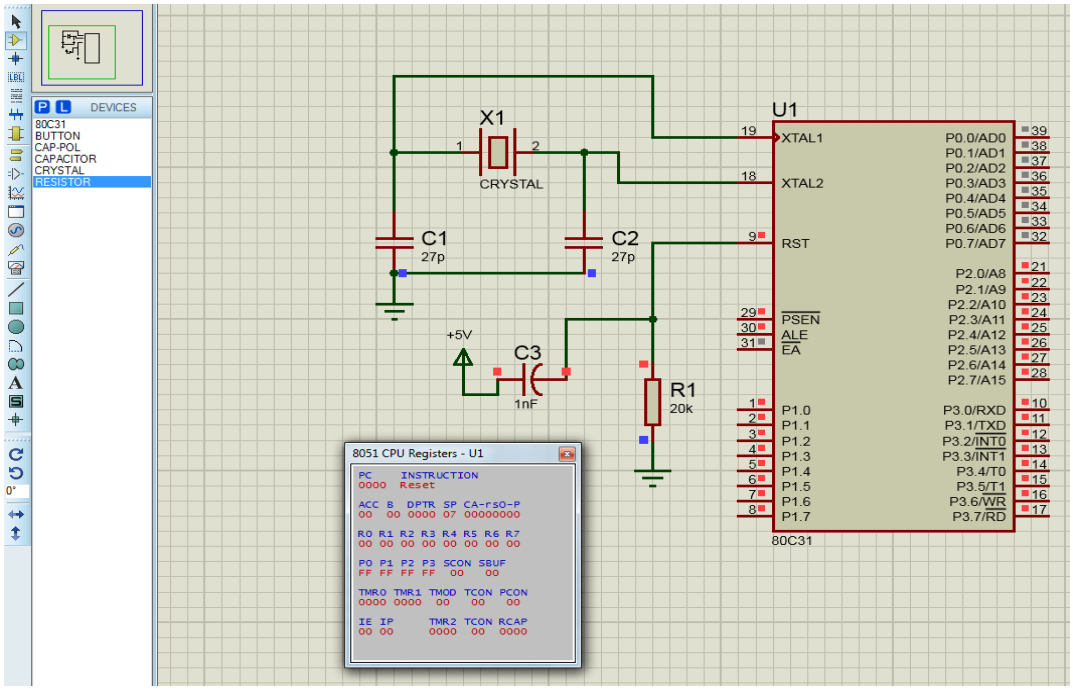

Рисунок 1.5 - Схема подключения микроконтроллера Intel 8051

Для того, чтобы вывести окна регистров и памяти необходимо войти в режим паузы, выбрать в меню Debug/8051CPU и далее в отрывающемся окне выбрать Registers SFR Memory, что показано на рисунке 1.6.

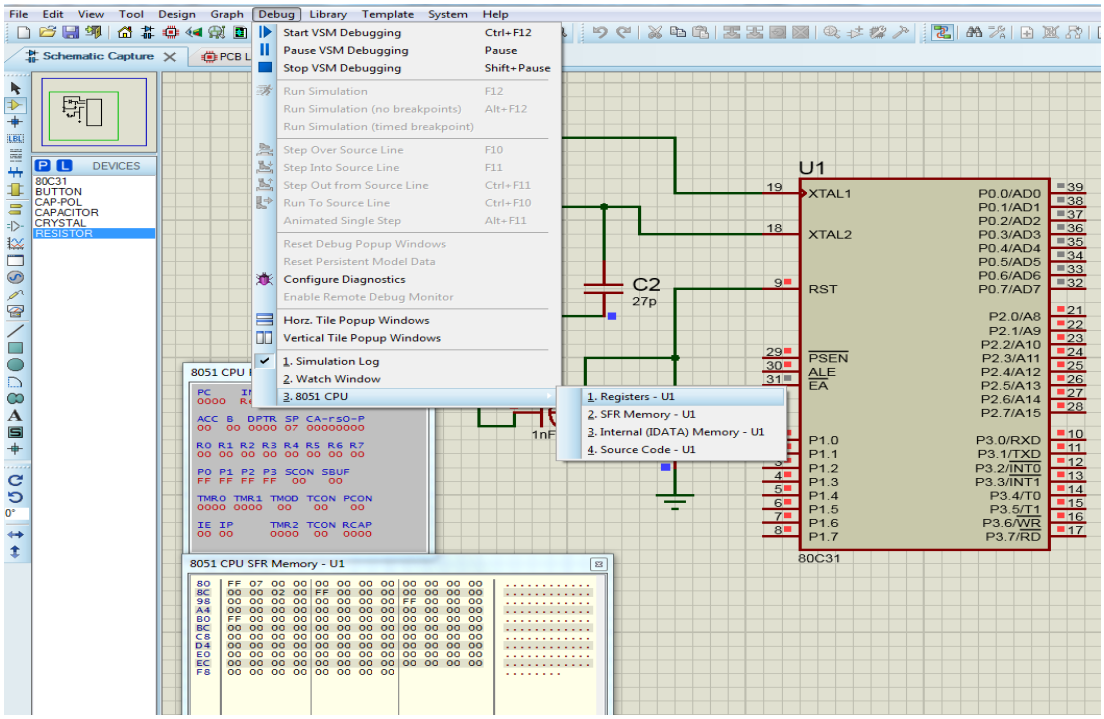

Рисунок 1.6 – Вывод окон регистров и резидентной памяти Intel 8051

1.3 Контрольные вопросы

1 Дать определение микропроцессора. Представить структурную схему и пояснить назначение основных элементов схемы.

2 Описать классификацию микропроцессоров по каждому признаку.

3 Дать характеристику регистров микропроцессора, их назначения.

4 Пояснить принцип построения микро-ЭВМ, назначение всех шин и узлов.

5 Дать определение микроконтроллера. Описать классы микроконтроллеров.

6 Назвать типовые характеристики микроконтроллера 8051.

7 В чем заключается суть Гарвардской архитектуры и в чем отличие от Принстонской? Пояснить на структурных схемах.

8 Пояснить назначение регистров PC, SP.

9 Дать характеристику назначения выводов микроконтроллера 8051.

10 Пояснить формат и назначение регистра DPTR.

11 Какие регистры управляют работой таймера? Перечислить и пояснить назначение каждого из них.

12 Пояснить назначение регистров аккумуляторов. Их связь с АЛУ и регистром PSW.

13 Сколько портов входит в базовую архитектуру микроконтроллера? Пояснить их структуру и функции.

14 Пояснить назначение регистров прерываний. Указать виды и количество прерываний.

15 Регистр словосостояния процессора PSW, структура, назначение всех битов. В каких командах используется этот регистр?

16 Назначение АЛУ. Связь АЛУ с другими модулями микроконтроллера.

17 Назначение внутренней двунаправленной шины микроконтроллера.

18 какой области памяти отображаются регистры общего назначения, пояснить их организацию и указать адреса.

2 Практическая работа 2. Изучение системы команд МК 8051

Цель: Освоение системы команд микроконтроллера Intel 8051.

2.1 Краткие теоретические сведения

Микро-ЭВМ рассматриваемого семейства являются типичными микропроцессорными устройствами с архитектурой SISC - со стандартным набором команд. Поэтому их система команд довольно обширна и включает в себя 111 основных команд. Их длина – один, два или три байта, причем большинство из них (94%) – одно- или двухбайтные. Все команды выполняются за один или два машинных цикла (соответственно 1 или 2 мкс при тактовой частоте 12 МГц), исключение – команды умножения и деления, которые выполняются за четыре машинных цикла (4 мкс). Микро-ЭВМ семейства 8051 используют прямую, непосредственную, косвенную и неявную, адресацию данных

В качестве операндов команд микро-ЭВМ семейства 8051 могут использовать отдельные биты, четырехбитные цифры, байты и двухбайтные слова.

Все эти черты обычны для набора команд любого SISC-процессора и по сравнению с RISC набором команд обеспечивает большую компактность программного кода и увеличение быстродействия при выполнении сложных операций.

В то же время, набор команд семейства 8051 имеет несколько особенностей, связанных с типичными функциями выполняемыми микроконтроллерами - управлением, для которого типичным является оперирование с одноразрядными двоичными сигналами, большое число операций ввода выводи и ветвлений программы. Наиболее существенная особенность системы команд рассматриваемых микро ЭВМ это возможность адресации отдельных бит в

резидентной памяти данных. Кроме того, как отмечалось, некоторые регистры блока регистров специальных функций также допускают адресацию отдельных бит. Карты адресов отдельных бит в резидентной памяти данных и в блоке регистров специальных функций.

Типы команд.

Всего микро-ЭВМ выполняют 13 типов команд, они приведены в таблице 2.1. Как следует из нее, первый байт команды всегда содержит код операции (КОП), а второй и третий (если они присутствуют в команде) – адреса операндов или их непосредственные значения.

Таблица 2.1 – Типы команд

Тип команды	Первый байт D7...D0	Второй байт D7...D0	Третий байт D7...D0
тип 1	коп		
тип 2	коп	#d	
тип 3	коп	ad	
тип 4	коп	bit	
тип 5	коп	rel	
тип 6	коп	a7...a0	
тип 7	коп	ad	#d
тип 8	коп	ad	rel
тип 9	коп	ads	add
тип 10	коп	#d	rel
тип 11	коп	bit	rel
тип 12	коп	ad16h	ad16l
тип 13	коп	#d16h	#d16l

Типы операндов.

Состав операндов включает в себя операнды четырёх типов: биты, 4-битные цифры, байты и 16-битные слова.

Микроконтроллер имеет 128 программно-управляемых флагов пользователя. Имеется также возможность адресации отдельных битов блока регистров специальных функций и портов. Для адресации битов используется прямой 8-битный адрес (bit). Косвенная адресация битов невозможна. Карты адресов отдельных битов представлены на рисунке 2.1.

Адреса	D_7)							D_0)
7FH								
2FH	7F	7E	7D	7C	7B	7A	79	78
2EH	77	76	75	74	73	72	71	70
2DH	6F	6E	6D	6C	6B	6A	69	68
2CH	67	66	65	64	63	62	61	60
2BH	5F	5E	5D	5C	5B	5A	59	58
2AH	57	56	55	54	53	52	51	50
29H	4F	4E	4D	4C	4B	4A	49	48
28H	47	46	45	44	43	42	41	40
27H	3F	3E	3D	3C	3B	3A	39	38
26H	37	36	35	34	33	32	31	30
25H	2F	2E	2D	2C	2B	2A	29	28
24H	27	26	25	24	23	22	21	20
23H	1F	1E	1D	1C	1B	1A	19	18
22H	17	16	15	14	13	12	11	10
21H	0F	0E	0D	0C	0B	0A	09	08
20H	07	06	05	04	03	02	01	00
1FH								
	Банк 3							
18H								
17H								
	Банк 2							
10H								
0FH								
	Банк 1							
08H								
07H								
	Банк 0							
00H								

Рисунок 2.1 - Карта адресуемых битов в резидентной памяти данных

Четырехбитные операнды используются только при операциях обмена SWAP и XCHD. Восьмибитным операндом может быть ячейка памяти программ (ПП) или данных (резидентной (РПД) или внешней (ВПД)), константа (непосредственный операнд), регистры специальных функций, а также порты ввода/вывода. Порты и регистры специальных функций адресуются только прямым способом. Байты памяти могут адресоваться также и косвенным образом через адресные регистры R0, R1, DPTR и PC.

Двухбайтные операнды - это константы и прямые адреса, для представления которых используются второй и третий байты команды.

Группы команд.

Система команд семейства MCS-51 содержит 111 базовых команд, которые по функциональному признаку можно подразделить на пять:

- пересылки данных;
- арифметических операций;
- логических операций;
- операций над битами;
- передачи управления.

Формат команд - одно-, двух- и трехбайтовый, причем большинство команд (94) имеют формат один или два байта. Первый байт любых типа и формата всегда содержит код операции, второй и третий байты содержат либо адреса операндов, либо непосредственные операнды.

Состав операндов включает в себя операнды четырех типов: биты, ниблы (4 разряда), байты и 16-битные слова. Время исполнения команд составляет 1, 2 или 4 машинных цикла. При тактовой частоте 12 мГц длительность машинного цикла составляет 1 мкс, при этом 64 команды исполняются за 1 мкс, 45 команд - за 2 мкс и 2 команды (умножение и деление) - за 4 мкс.

Способы адресации.

*1.Прямая адресация (Direct Addressing).*Операнд определяется 8-битным адресом в инструкции. Эта адресация используется только для внутренней памяти данных и регистров SFR.

*2.Косвенная адресация (Indirect Addressing).*В этом случае инструкция адресует регистр, содержащий адрес операнда. Данный вид адресации может применяться при обращении как к внутреннему, так и внешнему ОЗУ. Для указания 8-битных адресов могут использоваться регистры R0 и R1 выбранного регистрового банка или указатель стека SP.

Для 16-битной адресации используется только регистр "указатель данных" (DPTR - Data Pointer).

*3.Регистровая адресация (Register Instruction).*Данная адресация применяется для доступа к регистрам R0+R7 выбранного банка. Команды с регистровой адресацией содержат в байте кода операции трехбитовое поле, определяющее номер регистра. Выбор одного из четырех регистровых банков осуществляется программированием битов селектора банка (RS1, RS0) в PSW.

*4.Непосредственная адресация (Immediate constants).*Операнд содержится непосредственно в поле команды вслед за кодом операции и может занимать один или два байта ($data_8$, $data_{16}$).

5.Индексная адресация (Indexed Addressing). Индексная адресация используется при обращении к памяти программ и только при чтении. В этом режиме осуществляется просмотр таблиц в памяти программ. 16-битовый регистр (DPTR или PC) указывает базовый адрес требуемой таблицы, а аккумулятор указывает на точку входа в нее. Адрес элемента таблицы находится сложением базы с индексом (содержимым аккумулятора).

Другой тип индексной адресации применяется в командах "перехода по выбору" (Case Jump). При этом адрес перехода вычисляется как сумма указателя базы и аккумулятора.

*Неявная адресация (Register-Specific Instructions).*Некоторые инструкции используют индивидуальные регистры (например, операции с аккумулятором, DPTR), при этом данные регистры не имеют адреса, указывающего на них; это заложено в код операции

Обозначения, используемые при описании команд.

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,

Команды пересылки данных микроконтроллера 8051.

Эта группа представлена 28 командами, их краткое описание приведено в таблице 2.2, где также указаны тип команды (Т) в соответствии с таблицей, ее длина в байтах (Б) и время выполнения в машинных циклах (Ц). В таблице показан также код операции и место хранения результата операции.

Таблица 2.2 - Команды пересылки и загрузки

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))
Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
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	$(A_{0\ldots3})$ ↔$((Ri)_{0\ldots3})$

По команде MOV выполняется пересылка данных из второго операнда в первый. Эта команда не имеет доступа ни к внешней памяти данных, ни к памяти программ. Для этих целей предназначены команды M0VX и MOVC соответственно. Первая из них обеспечивает чтение/запись байт из внешней памяти данных, вторая – чтение байт из памяти программ.

По команде XCH выполняется обмен байтами между аккумулятором и ячейкой РПД, а по команде XCHD – обмен младшими тетрадами (битами 0 – 3).

Команды PUSH и POP предназначены соответственно для записи данных в стек и их чтения из стека. Размер стека ограничен лишь размером резидентной памяти данных. В процессе инициализации микро-ЭВМ 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.

Группа команд пересылки микроконтроллера имеет следующую особенность - в ней нет специальных команд для работы со специальными регистрами: PSW, таймером, портами ввода-вывода. Доступ к ним, как и к другим регистрам специальных функций, осуществляется заданием соответствующего прямого адреса, т.е. это команды обычных пересылок, в которых вместо адреса можно ставить название соответствующего регистра. Например, чтение PSW в аккумулятор может быть выполнено командой:

MOV A, PSW, которая преобразуется Ассемблером к виду:

MOV A, 0D0h (E5 D0),

где E5 – код операции, а D0 – операнд (адрес PSW).

Кроме того, следует отметить, что в микро-ЭВМ аккумулятор имеет два различных имени в зависимости от способа адресации: A – при неявной адресации (например, MOV A, R0) и ACC – при использовании прямого адреса. Первый способ предпочтительнее, однако, не всегда применим.

2.2 Задание

1 Построить электрическую схему в программной среде, используя микросхему 80C51. Подключить схему тактирования и синхронизации.

2 Вывести на экран окна резидентной памяти программ и данных микроконтроллера. Изучить их структуру.

3 Изучить команды пересылки и загрузки. Составить программу по вариантам.

4 Вывести на экран содержимое регистров, окно с программным кодом.

5 Изучить пошаговый режим выполнения программы.

6 Составить отчет о выполненной работе.

Таблица 2.1 – Варианты заданий

Команда, № варианта	1	2	3	4	5	6
Загрузка в регистр константы	R0, 34h	R1,43h	R2,01b	R3,0101b	R4,34d	R5,25h
Пересылка в аккумулятор из регистра	R0	R1	R2	R3	R4	R5
Пересылка в аккумулятор прямоадресуемого байта (ad)	02h	23h	48h	32h	12h	26h
Пересылка в аккумулятор байта из РПД (i=0,1)	23h	41h	12h	07h	03h	04h
Загрузка в аккумулятор константы	12h	15d	0101b	42h	18d	10101b
Пересылка в регистр из аккумулятора	R0	R1	R2	R3	R4	R5
Пересылка в регистр прямоадресуемого байта	02h	23h	48h	32h	12h	26h
Пересылка по прямому адресу аккумулятора	32h	43h	18h	32h	14h	16h
Пересылка по прямому адресу регистра	R0	R1	R2	R3	R4	R5
Пересылка по прямому адресу константы	32h	43h	18h	32h	14h	16h
Загрузка указателя данных	12h	15d	0101b	42h	18d	10101b
Пересылка в аккумулятор байта из ПП	00h	01h	02h	03h	04h	05h
Обмен аккумулятора с регистром	R0	R1	R2	R3	R4	R5
Обмен аккумулятора с прямоадресуемым байтом	02h	23h	48h	32h	12h	26h
Обмен младших тетрад аккумулятора и байта РПД	32h	43h	18h	32h	14h	16h

Продолжение таблицы 2.1

Команда, № варианта	7	8	9	10	11	12
Загрузка в регистр константы	R0, 10h	R1,03h	R2,010b	R3,1101b	R4,64d	R5,220h
Пересылка в аккумулятор из регистра	R6	R7	R8	R3	R4	R5
Пересылка в аккумулятор прямоадресуемого байта (ad)	0Ah	63h	10h	39h	42h	23h
Пересылка в аккумулятор байта из РПД (i=0,1)	0ABh	35h	82h	37h	0Fh	0A4h
Загрузка в аккумулятор константы	11h	150d	1101b	0B2h	208d	10111b
Пересылка в регистр из аккумулятора	R8	R7	R5	R2	R1	R4
Пересылка в регистр прямоадресуемого байта	025h	0B3h	0EFh	72h	1Ch	2Eh
Пересылка по прямому адресу аккумулятора	0ABh	35h	82h	0D2h	0BCh	0EAh
Пересылка по прямому адресу регистра	R6	R7	R8	R3	R4	R5
Пересылка по прямому адресу константы	45h	6Fh	2Ah	0D2h	1Ah	1Eh
Загрузка указателя данных	15h	18d	1101b	4Bh	180d	10111b
Пересылка в аккумулятор байта из ПП	00h	01h	02h	03h	04h	05h
Обмен аккумулятора с регистром	R8	R1	R5	R3	R2	R1
Обмен аккумулятора с прямоадресуемым байтом	0Bh	63h	10h	3Eh	42h	23h
Обмен младших тетрад аккумулятора и байта РПД	02h	53h	17h	3Ah	1Ch	1Eh

2.3 Контрольные вопросы

1 К какому типу архитектуры команд относится система команд микроконтроллера 8051? Каковы отличия CISC и RISC – архитектур команд?

2 Перечислите и поясните группы команд микроконтроллера 8051. Привести примеры команд из каждой группы.

3 Перечислите и опишите способы адресации микроконтроллера 8051. Привести пример команды на каждый способ адресации.

4 В чем отличие прямой адресации от регистровой? Пояснить на примере.

5 Что такое аккумулятор, какие функции он выполняет в микроконтроллере 8051? Сколько аккумуляторов в микроконтроллере 8051, какие

функции у второго аккумулятора? Привести примеры команд, использующие регистры-аккумуляторы.

6 Какие команды существуют для обмена данными между регистрами? Привести примеры команд обмена данными.

7 Сколько байт содержат команды пересылки и загрузки? От чего зависит длина команды? Какие команды имеют самую короткую длину?

8 Какова структура команды? Какие поля входят в формат команды? Пояснить на примере.

9 Какую особенность имеет группа команд пересылок микроконтроллера?

10 Для каких операций используется регистр указатель данных?

11 Какие вы знаете типы операндов? Привести примеры.

12 Назовите флаги регистра словосостояния процессора.

13 Какие флаги могут быть сформированы в результате выполнения команд пересылки и загрузки? Привести примеры.

14 Какие флаги могут быть сформированы в результате выполнения арифметических команд? Привести примеры.

15 Какие команды входят в состав группы логических команд?

16 Какие команды входят в состав группы команд передачи управления?

17 Какие команды входят в состав группы команд побитовой обработки?

18 Запишите три способа выбора банка № 3, используя различные команды.

3 Практическая работа 3. Изучение портов ввода-вывода микроконтроллера Intel 8051

Цель:

- Освоение навыков построения схемы подключения светодиодов;
- Изучение команд обращения к портам микроконтроллера.

3.1 Краткие теоретические сведения

Модульная организация микроконтроллеров.

Современные 8-разрядные МК обладают, как правило, рядом отличительных признаков. Перечислим основные из них:

− модульная организация, при которой на базе одного процессорного ядра (центрального процессора) проектируется ряд (линейка) МК, различающихся объемом и типом памяти программ, объемом памяти данных, набором периферийных модулей, частотой синхронизации;

− использование закрытой архитектуры МК, которая характеризуется отсутствием линий магистралей адреса и данных на выводах корпуса МК. Таким образом, МК представляет собой законченную систему обработки данных, наращивание возможностей которой с использованием параллельных магистралей адреса и данных не предполагается;

− использование типовых функциональных периферийных модулей (таймеры, процессоры событий, контроллеры последовательных интерфейсов, аналого-цифровые преобразователи и др.), имеющих незначительные отличия в алгоритмах работы в МК различных производителей;

− расширение числа режимов работы периферийных модулей, которые задаются в процессе инициализации регистров специальных функций МК.

При модульном принципе построения все МК одного семейства содержат процессорное ядро, одинаковое для всех МК данного семейства, и изменяемый

функциональный блок, который отличает МК разных моделей. Структура модульного МК приведена на рисунке 3.1.

Процессорное ядро включает в себя:

- центральный процессор;
- внутреннюю контроллерную магистраль (ВКМ) в составе шин адреса, данных и управления;
- схему синхронизации МК;
- схему управления режимами работы МК, включая поддержку режимов пониженного энергопотребления, начального запуска (сброса) и т.д.

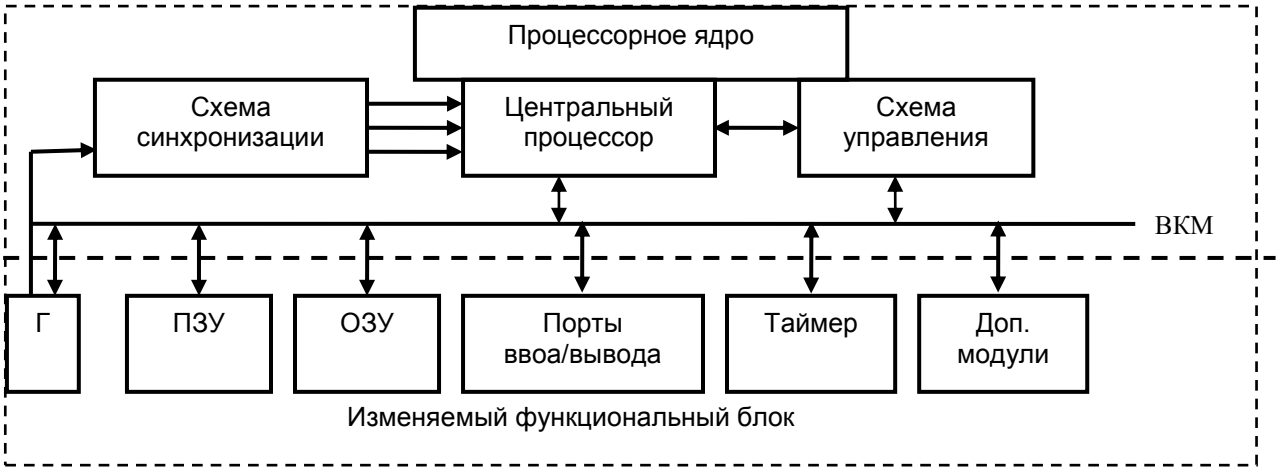

Рисунок 3.1 - Модульная организация МК

Изменяемый функциональный блок включает в себя модули памяти различного типа и объема, порты ввода/вывода, модули тактовых генераторов (Г), таймеры. В относительно простых МК модуль обработки прерываний входит в состав процессорного ядра. В более сложных МК он представляет собой отдельный модуль с развитыми возможностями.

В состав изменяемого функционального блока могут входить и такие дополнительные модули как компараторы напряжения, аналого-цифровые преобразователи (АЦП) и другие. Каждый модуль проектируется для работы в составе МК с учетом протокола ВКМ. Данный подход позволяет создавать разнообразные по структуре МК в пределах одного семейства.

Базовая архитектура микроконтроллера 8051 содержит четыре параллельных порта ввода/вывода - P0 - P3. Они адресуются как регистры специальных функций и имеют фиксированные адреса в памяти данных микроконтроллера, что показано в таблице 3.1.

Таблица 3.1 – Адреса регистров специальных функций Intel 8051

Адрес	Символ	Наименование
1	2	3
0E0H	*ACC	Аккумулятор (Accumulator)
0F0H	*B	Регистр расширитель аккумулятора (Multiplication Register)
0D0H	*PSW	Слово состояния программы (Program Status Word)
080H	*P0	Порт 0 (SFR P0)
090H	*P1	Порт 1 (SFR P1)
0A0H	*P2	Порт 2 (SFR P2)
0B0H	*P3	Порт 3 (SFR P3)
081H	SP	Регистр указатель стека (Stack Pointer)
083H	DPH	Старший байт регистра указателя данных DPTR (Data Pointer High)
082H	DPL	Младший байт регистра указателя данных DPTR (Data Pointer Low)
08CH	TH0	Старший байт таймера 0 ()
08AH	TL0	Младший байт таймера 0 ()
08DH	TH1	Старший байт таймера 1 ()
08BH	TL1	Младший байт таймера 1 ()
089H	TMOD	Регистр режимов таймеров счетчиков (Timer/Counter Mode Control Register)
088H	*TCON	Регистр управления статуса таймеров (Timer/Counter Control Register)
0B8H	*IP	Регистр приоритетов (Interrupt Priority Control Register)
0A8H	*IE	Регистр маски прерывания (Interrupt Enable Register)
087H	PCON	Регистр управления мощностью (Power Control Register)
098H	*SCON	Регистр управления приемопередатчиком (Serial Port Control Register)
099H	SBUF	Буфер приемопередатчика (Serial Data Buffer)

Разрядность портов - 8 с возможностью побитной адресации разрядов. Направление обмена информацией через порты - все порты двунаправленные, причем имеется возможность в каждом порту часть разрядов использовать для ввода данных, а часть для вывода.

Альтернативные функции. Из-за ограниченного количества выводов корпуса интегральной микросхемы микроконтроллера, большинство выводов используется для выполнения двух функций - в качестве линий портов и для альтернативных функций.

Порты P0 и P2 используются при обращении к внешней памяти. При этом на выходах порта P0 младший байт адреса внешней памяти мультиплексируется с вводимым/выводимым байтом. Выходы порта 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 единицы во все биты. Информация в разрядах порта 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. Альтернативные функции порта P3 указаны в таблице 3.2.

Таблица 3.2 – Альтернативные функции 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

Альтернативные функции могут быть активированы только в том случае, если в соответствующие биты порта P3 предварительно занесены «1». Неиспользуемые альтернативным образом разряды могут работать как обычно. 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 процессорного элемента (CPU). С прямого выхода D-триггера информация мажет быть выведена на внутреннюю шину по сигналу «Чтение SFR Px» от CPU, а с вывода микросхемы («из внешнего мира») по сигналу «Чтение выводов Px». Одни команды активизируют сигнал «Чтение SFR PI», другие - «Чтение выводов PI».

Особенности электрических характеристик портов. Выходные каскады триггеров SFR портов P1 — P3 выполнены на полевых транзисторах с внутренней нагрузкой, в то время как аналогичные каскады триггеров SFR P0—на транзисторах с открытым стоком. Каждая линия любого из портов может независимо использоваться как для ввода, так и для вывода информации (для линий портов P0 и P2 это справедливо тогда, когда они не используются для обращения к внешней памяти).

Для перевода любой линии портов P1 — P3 в режим ввода информации необходимо в соответствующий разряд SFR занести «1». При этом выходной полевой транзистор отключается. Внутренний нагрузочный резистор как бы «подтягивает» потенциал вывода к напряжению питания, в то время как внешняя нагрузка может сделать его нулевым. Выходные каскады порта P0 имеют иную структуру. Нагрузочный полевой транзистор линии порта включен только тогда, когда порт выводит «1» при обращении к внешней памяти. В остальных случаях нагрузочный транзистор отключен. Таким образом, при работе в режиме обычного ввода-вывода информации (как, например, порт P1) выходные каскады порт» P0 представляют собой ступени на транзисторах с открытым стоком. Запись «1» в соответствующий бит SFR отключает и второй транзистор, что приводит к тому, что вывод БИС оказывается под «плавающим» потенциалом. Это позволяет использовать линии порта P0 как выводы с высокоимпедансным состоянием. Схема подключения подгрузочных резисторов к порту P0 показана на рисунке 3.2.

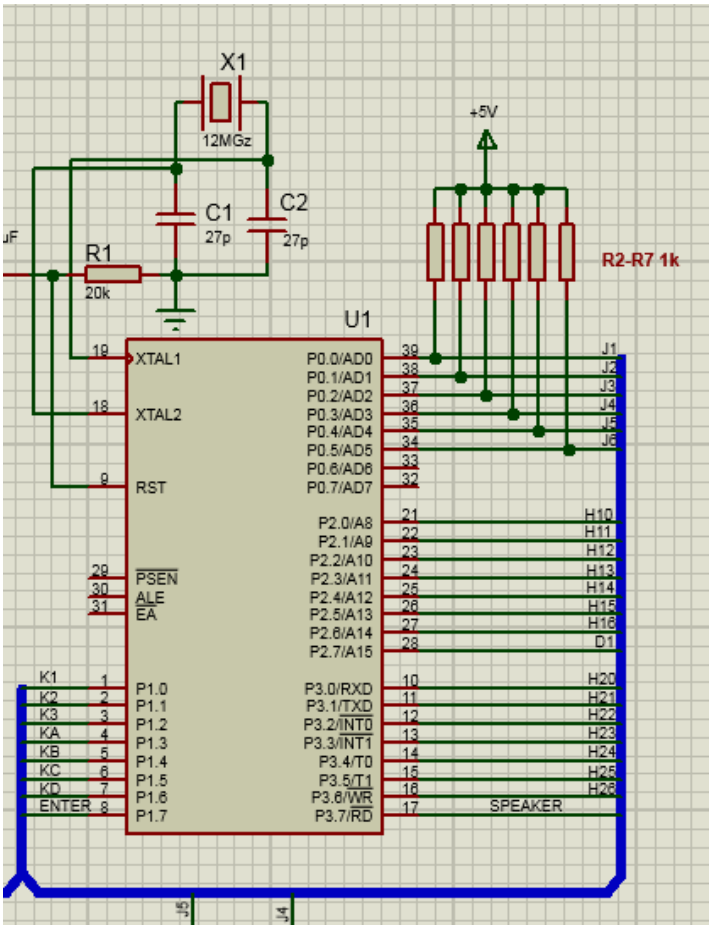

Рисунок 3.2 - Схема подключения подгрузочных резисторов к порту P0

Поскольку выходные каскады портов P1 — P3 имеют внутреннюю нагрузку, при переводе в режим ввода информации они становятся источниками тока для микросхемы или транзистора.

Процедура задержки ставится после зажигания каждого светодиода.

```
dely:
        mov  R6,#255
dd0:    mov  R7,#255
        djnz  R7,$
        djnz  R6,dd0
        ret
```

Для примера возьмем задачу поочередного зажигания светодиодов и при нажатии на кнопку произвести инвертирование результата. Проанализировав поставленную задачу можно составить как минимум два алгоритма решения. Первый заключается в том, что все комбинации зажигания светодиодов заносятся в таблицу, а затем в цикле поочередно извлекаются, второй – в том, что можно

выявить некоторую закономерность комбинаций и составить последовательность действий для реализации этой закономерности. У каждого из алгоритмов могут быть свои преимущества и недостатки, а именно такие как простота реализации и размер программного кода.

Для реализации первого метода главным достоинством является использование команды для работы с таблицами movc A,@A+DPTR. Перед выполнением команды в регистр указатель DPTR заносится адрес таблицы в памяти программ, а в регистр аккумулятор A – индекс элемента. После выполнения команды в регистр A помещается элемент таблицы размером в 1 байт. Составленный листинг программы, реализующий первый метод, приведен ниже.

```
 N equ 8 ; число элементов в таблице
;--------------------------------
Start:
        mov     DPTR,#Table ; В рег. указатель заносится адрес таблицы
init:   mov     B,#0        ; В рег. B хранится индекс на элемент таблицы
cycle:          mov     A,B             ; Копируем индекс из рег.A в рег.B
        cjne    A,#N,NoInc  ; Если индекс указывает на N+1 элемент
        jmp     init        ; то переход на обнуление индекса
                            ; иначе прыжок на метку NoInc
NoInc:          movc    A,@A+DPTR       ; в рег.A загружаем элемент, из таблицы
        push    ACC         ; Сохраняем аккумулятор в стеке (на случай, если
                            ; произойдет инвертирование)
        jb      P2.0, NoInvrt ; Если на входе P2.0 лог.едиинца (не нажали кнопку),
                            ; то переход на метку NoInvrt
        cpl     A           ; иначе инвертируем биты в аккумуляторе
NoInvrt:
        mov     P1, A       ; Записываем значение рег. A в порт P1
        pop     ACC         ; Восстанавливаем аккумулятор
        call    delay_140msec       ; Вызываем процедуру задержки
        inc     B           ; Инкрементируем индекс
        jmp     cycle       ; Повторяем цикл

;--------------------------------
; процедура задержки на 140 мс (при f=11059 кГц)
delay_140msec:
        mov     R6,#254             ;
dd1:    mov     R7,#254             ;
        djnz    R7,$        ; (Знак "$" - текущий адрес)
        djnz    R6,dd1
        ret

;--------------------------------
; Таблица для линейки светодиодов
```

```
Table: db     10000000b
      db      01000000b
      db      00100000b
      db      00010000b
      db      00001000b
      db      00000100b
      db      00000010b
      db      00000001b
END
```

Для реализации второго метода можно выявить закономерность зажигания светодиодов – циклический сдвиг разрядов от старшего к младшему. Данную операцию выполняет команда циклического сдвига содержимого аккумулятора вправо rr A. Ниже приведен листинг программы реализующий второй метод.

```
Name lab_1_b
;--------------------------------
Start:
      mov   A,#10000000b        ; Устанавливаем  начальное значение
cycle:
      push  ACC                 ; Сохраняем аккумулятор в стеке (на случай, если
                                ; произойдет инвертирование)
      jb    P2.0, NoInvrt  ; Если на входе P2.0 лог.едиинца (не нажади кнопку),
                                ; то переход на метку NoInvrt
      cpl   A                   ; иначе инвертируем биты в аккумуляторе
NoInvrt:
      mov   P1, A               ; Записываем значение рег. A в порт P1
      pop   ACC                 ; Восстанавливаем аккумулятор
      rr    A                   ; Циклический сдвиг вправо
      call  delay_140msec       ; Вызываем процедуру задержки
      jmp   cycle               ; Повторяем цикл
;--------------------------------
;процедура задержки на 140 мс (при f=11059 кГц)
delay_140msec:
      mov   R6,#254             ;
dd1:  mov   R7,#254             ;
      djnz  R7,$                ; (Знак "$" - текущий адрес)
      djnz  R6,dd1
      ret
END
```

Проанализировав два листинга программного текста, можно сделать вывод, что при разработке первого метода текст программы составил 45 строчек и 48 байт программного кода против 25 строчек текста и 27 байт кода во втором методе. Второй метод оказался более оптимальным.

3.2 Задание

1 Построить схему в программной среде по образцу схемы на рисунке 3.3.

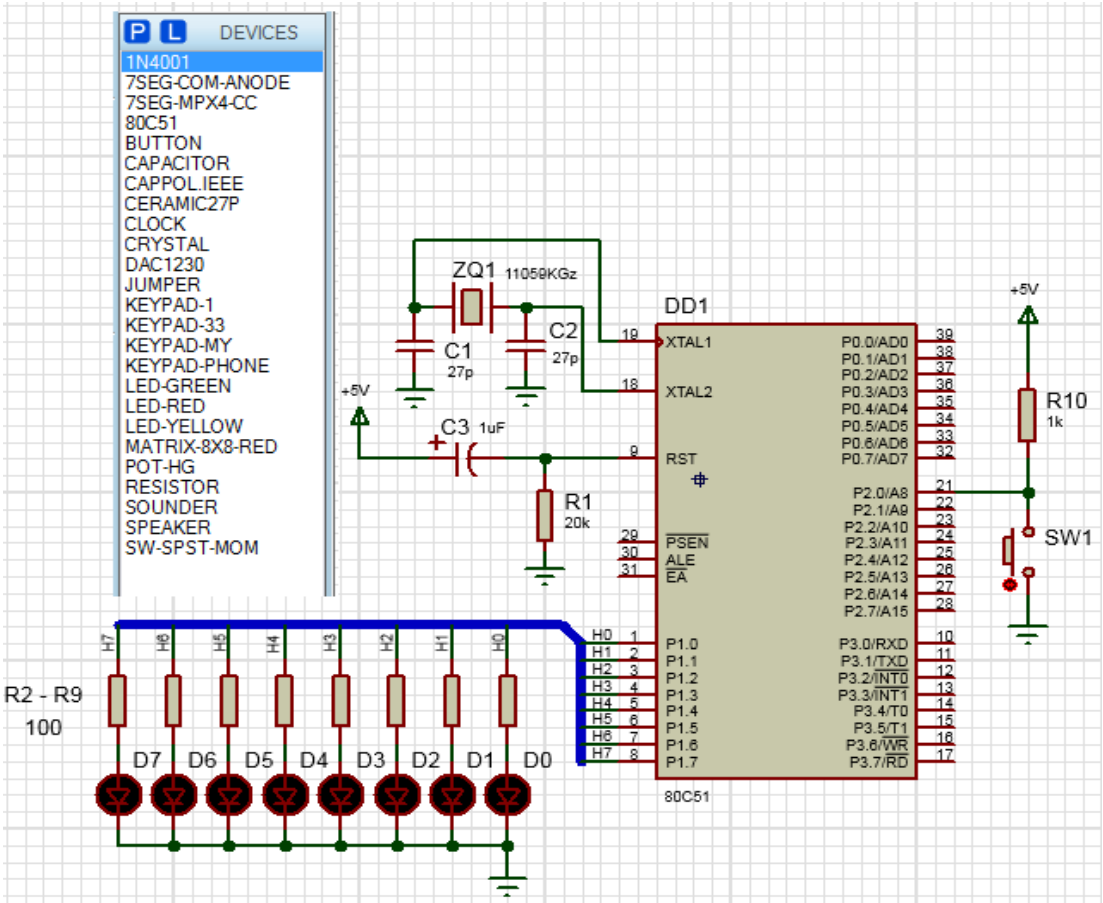

Рисунок 3.3 – Схема подключения светодиодов

2 Написать программу, реализующую поочередное зажигание светодиодов по вариантам, указанным в таблице 3.3. Организовать гашение светодиодов при нажатии на кнопку.

3 Вывести на экран окно регистров микроконтроллера. Запустив программу в пошаговом режиме, наблюдать за изменением состояния регистров портов.

4 Составить отчет о выполненной работе.

Таблица 3.3 – Варианты заданий

№ варианта	Таблица битов включения светодиодов. 1 – включен, 0 – выключен	№ варианта	Таблица битов включения светодиодов. 1 – включен, 0 – выключен	№ варианта	Таблица битов включения светодиодов. 1 – включен, 0 – выключен
1	10000000 01000000 00100000 00010000 00001000 00000100 00000010 00000001 00000010 00000100 00001000 00010000 00100000 01000000	5	00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000 01000000 00100000 00010000 00001000 00000100 00000010	9	10000000 11000000 11100000 11110000 11111000 11111100 11111110 11111111 11111110 11111100 11111000 11110000 11100000 11000000
2	11100000 01110000 00111000 00011100 00001110 00000111 10000011 11000001	6	00000111 00001110 00011100 00111000 01110000 11100000 11000001 10000011	10	11110000 01111000 00111100 00011110 00001111 00011110 00111100 01111000
3	10000001 01000010 00100100 00011000 00100100 01000010 10000001 00000000	7	00011000 00111100 01111110 11111111 01111110 00111100 00011000 00000000	11	10000001 11000011 11100111 11111111 11100111 11000011 10000001 00000000
4	10000000 11000000 11100000 11110000 11111000 11111100 11111110 11111111 01111111 00111111 00011111 00001111 00000111 00000011	8	00000001 00000011 00000111 00001111 00011111 00111111 01111111 11111111 11111110 11111100 11111000 11110000 11100000 11000000	12	00000000 00000001 00000011 00000111 00001111 00011111 00111111 01111111 11111111 01111111 00111111 00011111 00001111 00000111

3.3 Контрольные вопросы

1 Что такое закрытая архитектура микроконтроллера? В чем выражается это понятие для микроконтроллера 8051.

2 Что подразумевает модульная организация микроконтроллера? Пояснить схему. Входят ли порты ввода вывода в состав ядра микроконтроллера? Перечислить модули, входящие в состав ядра.

3 Какие порты ввода/вывода и в каком количестве входят в состав базовой модели микроконтроллера 8051?

4 Какие функции выполняют параллельные порты микроконтроллера 8051? Рассмотреть каждый порт.

5 Какие вы знаете альтернативные функции параллельных портов? Назвать и охарактеризовать эти функции.

6 Как программным путем реализовать обращение к портам? Есть ли у них адреса? С помощью какого способа адресации можно обратиться к портам микроконтроллера? Привести примеры команд.

7 В чем заключается отличие порта P0 от других портов микроконтроллера? Каким образом осуществляется схемотехническое подключение к порту P0.

8 Каким образом подключается кнопка к порту микроконтроллера, как осуществить опрос нажатия кнопки?

9 В каком состоянии находятся порты микроконтроллера по умолчанию до программирования?

10 Как осуществить программно опрос портов микроконтроллера? Как программно осуществить сброс портов в нулевое состояние?

11 Каким образом необходимо подключать светодиоды к микроконтроллеру? Как осуществить программно их зажигание и гашение?

4 Практическая работа 4. Изучение арифметических и логических команд микроконтроллера Intel 8051

Цель: Освоение группы арифметических и логических команд.

4.1 Краткие теоретические сведения

Команды арифметических операций 8051.

В данную группу входят 24 команды, краткое описание которых приведено в таблице 4.1. Из нее следует, что микроЭВМ выполняет достаточно широкий набор команд для организации обработки целочисленных данных, включая команды умножения и деления.

В таблице также указаны тип команды (Т) в соответствии с таблицей, ее длина в байтах (Б) и время выполнения в машинных циклах (Ц).

Таблица 4.1- Команды арифметических операций

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
1	2	3	4	5	6	7
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	0010011	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	0011011	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

Продолжение таблицы 4.1

1	2	3	4	5	6	7
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	1001011	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	0000011	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	0001011	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)

По результату выполнения команд ADD, ADDC, SUBB, MUL и DIV устанавливаются флаги PSW, структура которых приведена в таблице.

Флаг C устанавливается при переносе из разряда D7, т. е. в случае, если результат не помещается в восемь разрядов; флаг AC устанавливается при переносе из разряда D3 в командах сложения и вычитания и служит для реализации десятичной арифметики. Этот признак используется командой DAA.

Флаг OV устанавливается при переносе из разряда D6, т. е. в случае, если результат не помещается в семь разрядов и восьмой не может быть интерпретирован как знаковый. Этот признак служит для организации обработки чисел со знаком.

Наконец, флаг P устанавливается и сбрасывается аппаратно. Если число единичных бит в аккумуляторе нечетно, то P = 1, в противном случае P = 0.

Команды логических операций микроконтроллера 8051.

В этой группе 25 команд, их краткое описание приведено в таблице 4.2. Нетрудно видеть, что эти команды позволяют выполнять операции над байтами: логическое И (/\ ), логическое ИЛИ (\/), исключающее ИЛИ ((+)), инверсию (NOT), сброс в нулевое значение и сдвиг. В таблице также указаны тип команды (Т) в соответствии с таблицей, ее длина в байтах (Б) и время выполнения в машинных циклах (Ц).

Таблица 4.2 – Команды логических операций

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
1	2	3	4	5	6	7
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

Продолжение таблицы 4.2

1	2	3	4	5	6	7
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	$(A_{0...3}) \leftrightarrow (A_{4...7})$

Команды операций над битами микроконтроллера 8051.

Группа состоит из 12 команд, краткое описание которых приведено в таблице 4.3. Эти команды позволяют выполнять операции над отдельными битами: сброс, установку, инверсию бита, а также логические И (/\) и ИЛИ (\/). В качестве "логического" аккумулятора, участвующего во всех операциях с двумя операндами, выступает признак переноса C (разряд D7 PSW), в качестве операндов могут использоваться 128 бит из резидентной памяти данных и регистры специальных функций, допускающие адресацию отдельных бит.

В таблице также указаны тип команды (Т) в соответствии с таблицей, ее длина в байтах (Б) и время выполнения в машинных циклах (Ц).

Таблица 4.3 - 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)

4.2 Задание

1 Построить схему подключения светодиодов и кнопки к микроконтроллеру 80C51. Восемь светодиодов подключить к порту P1, другие восемь светодиодов подключить к порту P2. Кнопку подключить к любому разряду порта P3. Схема показана на рисунке 4.1.

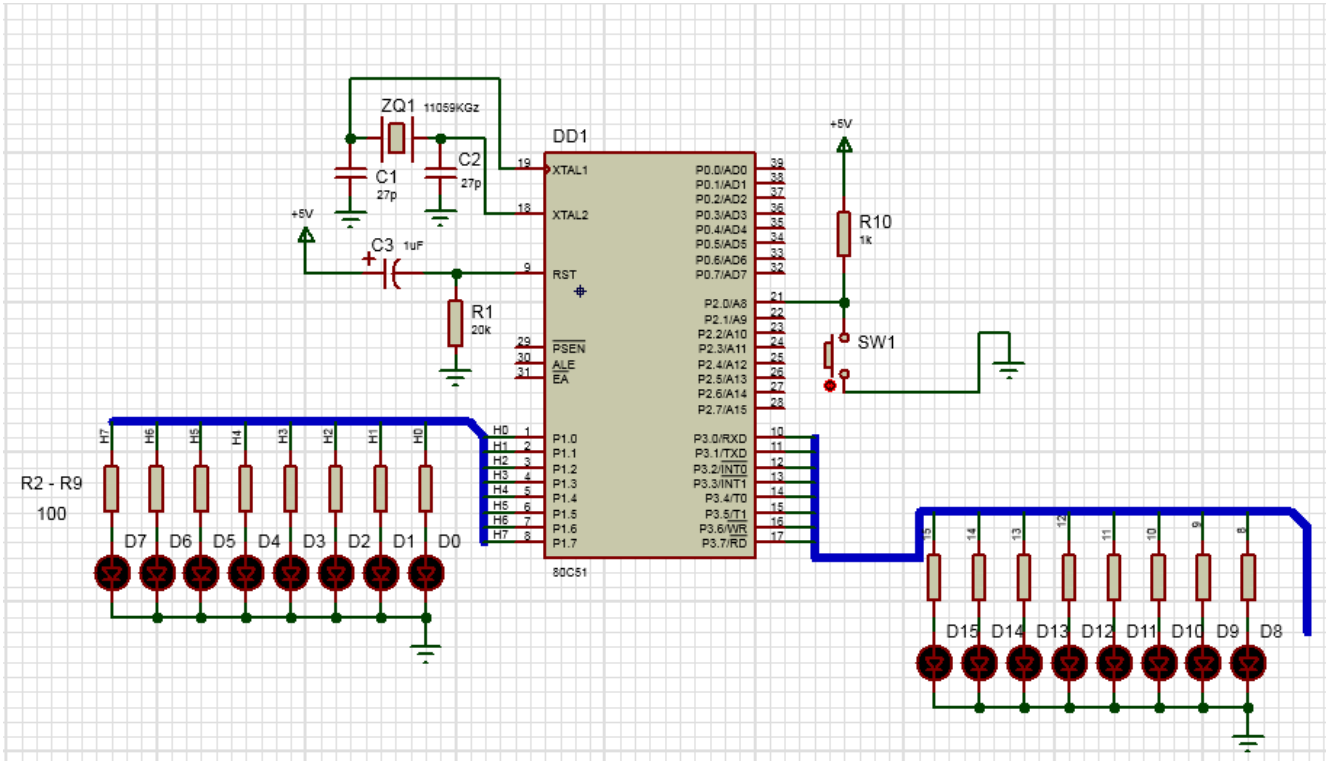

Рисунок 4.1 – Схема подключения светодиодов и кнопки

2 Составить программу, решающую следующие задачи:

– выбрать банк регистров 1.

– загрузить в регистры R0-R3 банка 1 операнды: 23; 142; 231; 203, представив их в разных кодах (B; D; H).

– сложить попарно числа в регистрах R0+R1; R2+R3. Результаты сложения сохранить в портах P1 и P2 соответственно.

– реализовать гашение светодиодов при нажатии кнопки.

3 Составить программу умножения чисел R0*R1; R2*R3. Результаты умножения сохранить в портах P1 и P2 соответственно.

4 Провести отладку программы в среде Proteus.

5 Вывести на экран содержимое всех используемых регистров.

4.3 Контрольные вопросы

1 Каким образом осуществляется выбор банка регистров общего назначения? Привести два способа программной реализации выбора банка.

2 Привести примеры команд арифметического сложения. Каким образом можно учесть возникновение переполнения разрядной сетки при сложении.

3 Какие флаги формируются при выполнении арифметических операций?

4 Привести примеры команд арифметического умножения. Каким образом можно учесть возникновение переполнения разрядной сетки при умножении.

5 В каких регистрах сохраняются результаты умножения? Где хранится старший байт результата?

6 Приведите пример команд деления. В каких регистрах хранится результат деления?

7 Как программным способом определить, что в результате умножения получилось двухбайтное число?

8 Как программным способом определить, что в результате деления получилось число с остатком, и где остаток сохраняется?

5 Практическая работа 5. Изучение группы команд передачи управления

Цель: Освоение навыков построения разветвлений программы с использованием команд переходов и передачи управления

5.1 Краткие теоретические сведения

Группа команд переходов и передачи управления представлена командами безусловного и условного переходов, командами вызова подпрограмм и командами возврата из подпрограмм.

Команда безусловного перехода LJMP (L – long – длинный) осуществляет переход по абсолютному 16-битному адресу, указанному в теле команды, т. е. команда обеспечивает переход в любую точку памяти программ.

Действие команды AJMP (A – absolute – абсолютный) аналогично команде LJMP, однако в теле команды указаны лишь 11 младших разрядов адреса. Поэтому переход осуществляется в пределах страницы размером 2 Кбайт, при этом надо иметь в виду, что сначала содержимое счетчика команд увеличивается на 2 и только потом заменяются 11 разрядов адреса.

В отличие от предыдущих команд, в команде SJMP (S – short – короткий) указан не абсолютный, а относительный адрес перехода. Величина смещения reI рассматривается как число со знаком, а, следовательно, переход возможен в пределах – 128...+127 байт относительно адреса команды, следующей за командой SJMP.

В таблице 5.1 представлены все команды группы переходов и передачи управления, также указаны тип команды (Т) в соответствии с таблицей, ее длина в байтах (Б) и время выполнения в машинных циклах (Ц).

Таблица 5.1 - Команды передачи управления

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
1	2	3	4	5	6	7
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_8$00001	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

Продолжение таблицы 5.1

1	2	3	4	5	6	7
Сравнение байта в РПД с константой и переход, если не равно	CJNE @Ri,#d,rel	1011011i	10	3	2	(PC) ← (PC) + 3,если ((Ri)) ≠ #d, то (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_8$10001	6	2	2	(PC) ← (PC) + 2, (SP) ← (SP) + 1, ((SP)) ← ($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

Команда косвенного перехода JMP @A+DPTR позволяет вычислять адрес перехода в процессе выполнения самой программы.

В качестве условий в командах условного перехода используются флаги операций: флаг нуля Z, флаг переноса C, флаг бита B, флаг четности P. Командами условного перехода можно проверять следующие условия:

- JZ — аккумулятор содержит нулевое значение;
- JNZ — аккумулятор содержит не нулевое значение
- JC — бит переноса C установлен;
- JNC — бит переноса C не установлен;
- JB — прямо адресуемый бит равен 1;
- JNB — прямо адресуемый бит равен 0;

– JBC — прямо адресуемый бит равен 1 и сбрасывается в нулевое значение при выполнении команды.

Все команды условного перехода рассматриваемых микро-ЭВМ содержат короткий относительный адрес, т. е. переход может осуществляться в пределах— 128... +127 байт относительно следующей команды.

Возможно производить декремент регистра с одновременной проверкой этого регистра на ноль. Команда DJNZ предназначена для организации программных циклов. Регистр Rn или байт по адресу ad, указанные в теле команды, содержат счетчик повторений цикла, а смещение rel — относительный адрес перехода к началу цикла. При выполнении команды содержимое счетчика уменьшается на 1 и проверяется на 0. Если значение содержимого счетчика не равно 0, то осуществляется переход на начало цикла, в противном случае выполняется следующая команда.

Команда CJN удобна для реализации процедур ожидания внешних событий. В теле команды указаны "координаты" двух байт и относительный адрес перехода rel. В качестве двух байт могут быть использованы, например, значения содержимого аккумулятора и прямо адресуемого байта или косвенно адресуемого байта и константы. При выполнении команды значения указанных двух байт сравниваются и в случае, если они не одинаковы, осуществляется переход. Например, команда:

WAIT: CJNE A, P0, WAIT

будет выполняться до тех пор, пока значения на линиях порта P0 не совпадут со значениями содержимого аккумулятора.

Действие команд вызова процедур полностью аналогично действию команд безусловного перехода. Единственное отличие состоит в том, что они сохраняют в стеке адрес возврата.

Команда возврата из подпрограммы RET восстанавливает из стека значение содержимого счетчика команд, а команда возврата из процедуры обработки прерывания RETI, кроме того, разрешает прерывание обслуженного уровня. Команды RET и RETI не различают, какой командой – LCALL или ACALL – была

вызвана подпрограмма, так как и в том, и в другом случае в стеке сохраняется полный 16-разрядный адрес возврата.

Следует отметить, что большинство Ассемблеров допускают обобщенную мнемонику JMP – для команд безусловного перехода и CALL – для команд вызова подпрограмм. Конкретный тип команды определяется Ассемблером, исходя из "длины" перехода или вызова.

В качестве примера применения команд переходов рассмотрим задачу проверки одиночного бита порта. Пусть кнопка подсоединена одним зажимом к «0», а другим к разряду 3 порта P0. Если кнопка не нажата, то на P0.3 логическая «1», а если нажата, то на втором ее зажиме будет «0», т.е. бит P0.3 = 0. Значит в программе надо проверять бит P0.3. Схема алгоритма программы приведена на рисунке 5.1.

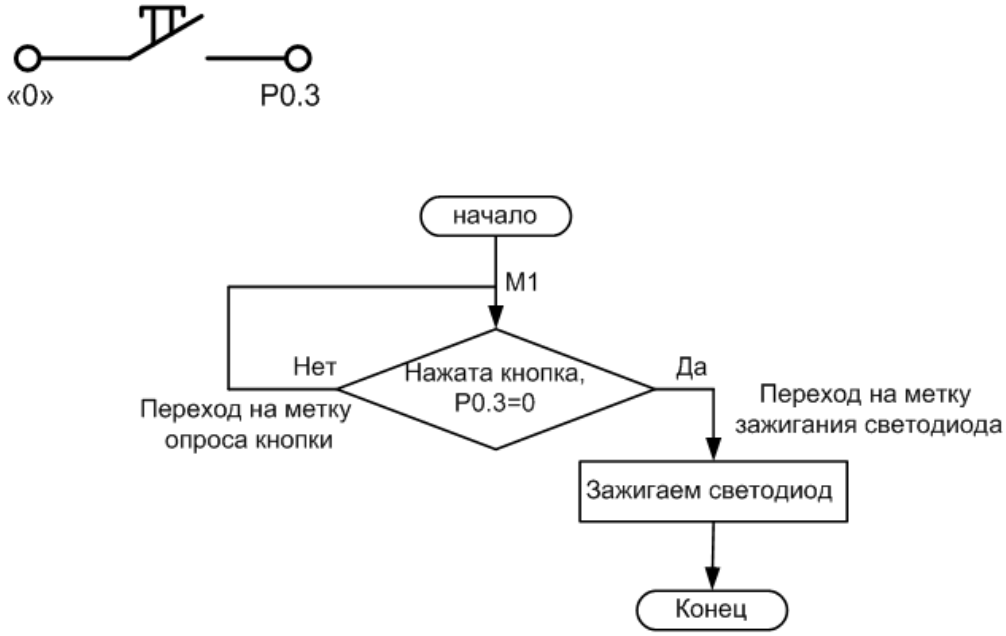

Рисунок 5.1 - Схема алгоритма программы

Программа проверки:

M1: JB P0.3, M1; если P0.3 = 1, то перейти по метке M1, а если нет,

то выполняется следующая команда

MOV P2,#1; зажигаем светодиод.

END.

Пример схемы алгоритма сравнения однобайтовых чисел приведен на рисунке 5.2.

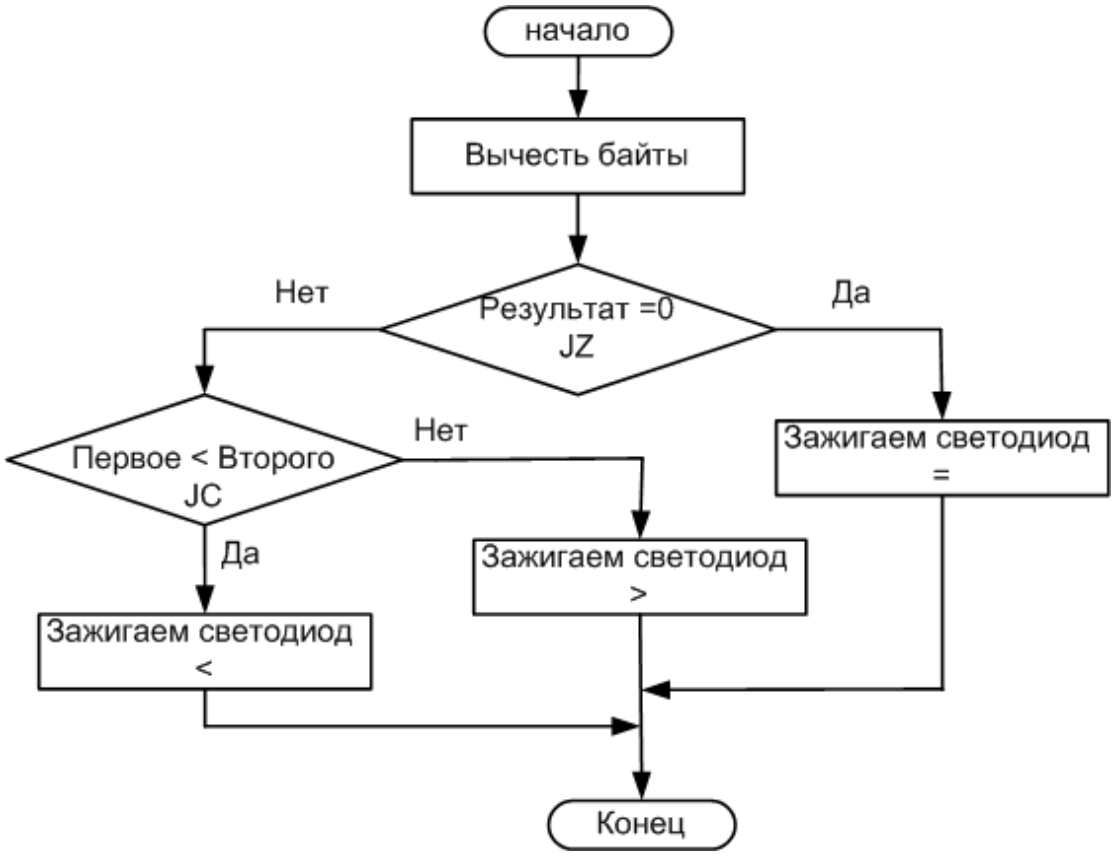

Рисунок 5.2 - Схема алгоритма программы сравнения однобайтовых чисел

Пример программы:

```
SUBB A, R1          - вычесть из А содержимое R1
      JZ РАВНЫ      - если результат вычитания =0, то перейти по метке РАВНЫ
      JC МЕНЬШЕ     - если первое число меньше второго, то перейти по метке МЕНЬШЕ
      MOV P1, #0    - если не меньше, то больше и зажечь светодиод «БОЛЬШЕ»
      JMP КОНЕЦ     - переход на метку «конец программы»
РАВНЫ:
      MOV P2, #0    - зажечь светодиод «РАВНЫ»
      JMP КОНЕЦ     - переход на метку «конец программы»
МЕНЬШЕ:
      MOV P3, #0    -  зажечь светодиод «МЕНЬШЕ»
      JMP КОНЕЦ     - безусловный переход на метку «конец программы»
КОНЕЦ:
            END.
```

5.2 Задание

1 Построить схему подключения трех светодиодов и кнопки к микроконтроллеру 80C51. Три светодиода подключить к порту P1. Кнопку подключить к любому разряду порта P3.

2 Составить схему алгоритма и программу для следующих задач:

− загрузить в регистры R0, R2, R3, R4 операнды;

− перемножить числа попарно R0*R1; R2*R3. Результаты умножения сохранить в регистрах R4, R5 и R6, R7 соответственно. **Необходимо учесть,** что результаты умножения могут быть 2-байтными числами, значит, для сохранения результатов понадобятся по 2 регистра;

− сравнить результаты умножения, предусмотреть сравнение 2-байтных чисел;

− при нажатии кнопки вывести через порт P3 результаты сравнения на три светодиода. Использовать светодиоды разного цвета и обозначить их «>», «<», «=», или словами.

Пример схемы приведен на рисунке 5.3.

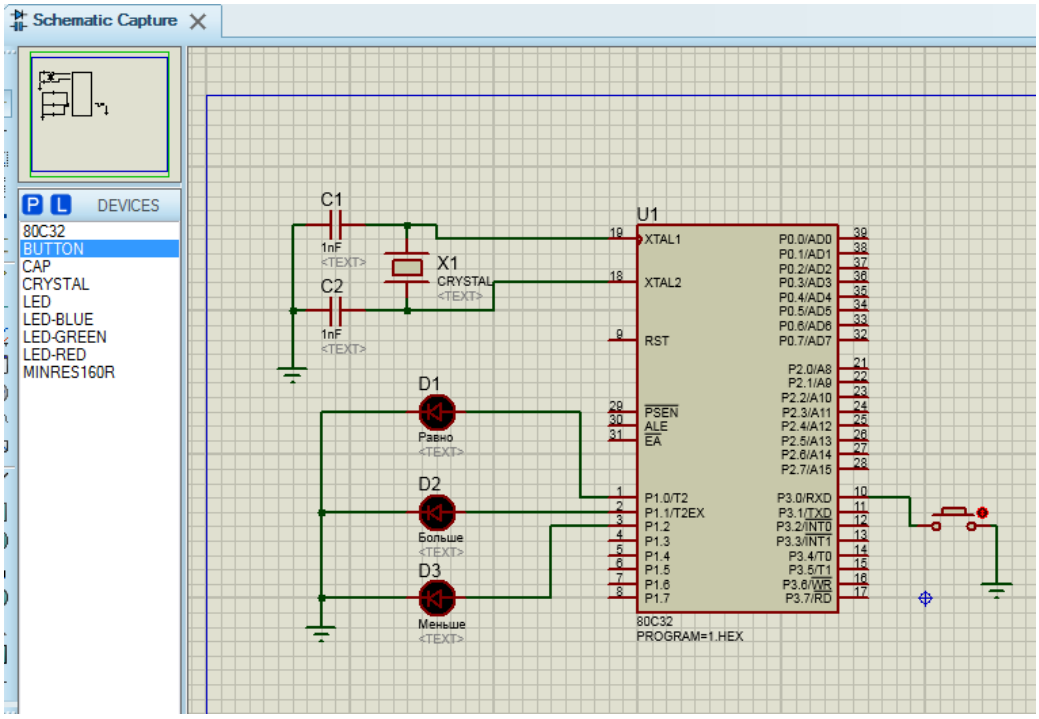

Рисунок 5.3 - Пример схемы

5.2 Контрольные вопросы

1 Назовите команды переходов, приведите примеры записи. По каким признакам осуществляются условные переходы, назвать флаги, привести примеры.

2 Какие виды безусловных переходов используются в системе команд микроконтроллера Intel 8051? Привести пример использования таких команд.

3 С помощью каких команд можно провести сравнение двух чисел? Как осуществить сравнение содержимого регистра с константой? Привести примеры.

4 В регистре R1 организовать счетчик событий, количество которых равно 5. Событием является умножение аккумулятора на 2. Реализовать повторение данного события 5 раз, уменьшая счетчик событий после каждого умножения. Проверить счетчик на ноль. Вывести результат умножения в порт P1. Использовать команду DJNZ.

5 Провести сравнение содержимого регистра R2 с константами: 1; 2; 3. Если R2=1, то установить бит порта P1.1=1. Если R2=2, то установить бит порта P1.2=1. Если R2=3, то установить бит порта P1.3=1.

6 Провести проверку результата умножения двух чисел. Если результат однобайтный, то сбросить бит порта P1.2=0. Если результат двухбайтный, то сбросить бит порта P1.3=0.

7 Провести проверку битов порта P1. Организовать в регистре R0 счетчик количества нулевых битов порта P1.

8 Провести проверку битов порта P1. Организовать в регистре R0 счетчик количества битов порта P1, равных единице.

6 Практическая работа 6. Подключение семисегментных индикаторов к микроконтроллеру

Цель: Освоение навыков аппаратно-программного обеспечения подключения семисегментных индикаторов к микроконтроллеру.

6.1 Краткие теоретические сведения

Семисегментный светодиодный индикатор - это устройство отображения цифровой информации, наиболее простая реализация индикатора, который может отображать цифры и буквы, состоит из семи элементов индикации (сегментов), включающихся и выключающихся по отдельности. Сегменты обозначаются буквами от A до G; восьмой сегмент — десятичная точка (decimal point, DP), предназначенная для отображения дробных чисел. Условное графическое изображение семисегментного индикатора приведено на рисунке 6.1.

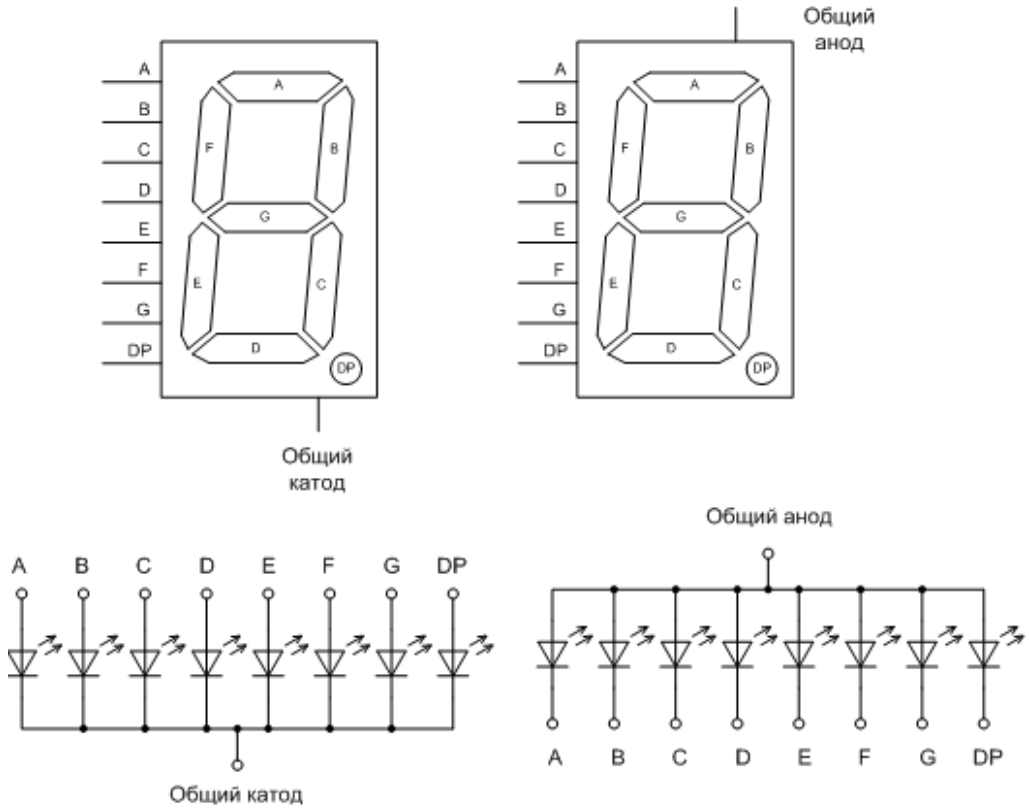

Рисунок 6.1 – Условное графическое изображение семисегментного индикатора

Семисегментные коды цифр для семисегментного индикатора зависят от модели индикатора, их можно подобрать на практике.

Перевод десятичного числа в семисегментный код.

Максимальное 8-разрядное число равняется 255. Для перевода 8-разрядного десятичного числа в семисегментный код надо найти отдельно каждую из трех цифр, значит вычислить сколько в этом числе сотен, десятков и единиц. Чтобы узнать, сколько в числе сотен, надо разделить это число **на 100**. В ассемблере есть команда деления **DIV AB**. В результате деления в регистр **A** попадает **целая часть (число сотен)**, в регистр **B** – **остаток**. Т.е. если исходное число **255**, то после деления **A=2, B=55**. Следующий шаг: выделить количество десятков. Для этого надо разделить остаток на **10**. Число **55:10** получим **A=5, B=5**. Значит в регистре **A** хранится количество десятков, а в регистре **B** – количество единиц. Результаты делений сохранять в регистрах.

После вычисления количества сотен, десятков и единиц, надо определить какая цифра получилась, и какой код ей соответствует. Предварительно составить таблицу соответствия:

11111100B - 0
01100000B - 1
11011010B - 2 и т.д.

Реализация таблиц.

Таблицы могут быть легко реализованы при помощи команды

movc A,@A+PC.

Здесь использована индексная адресация, в качестве индексного регистра служит регистр - программный счетчик **PC.**

Таблицы применяются, когда надо выборочно считывать из них данные.

Например, при преобразовании чисел в семисегментный код, коды цифр можно расположить в таблице. Пример программы:

```
mov    A, R0;       - загрузить в A число из регистра R0.
call   readtable;   - вызов прерывания по адресу readtable.
mov    P1,A;        - после возврата из прерывания загрузить  A в P3.
mov    A, R1;       - загрузить в A число из регистра R0.
call   readtable;   - вызов прерывания по адресу readtable.
mov    P2,A
mov    A, R2;       - загрузить в A число из регистра R2.
```

```
        call    readtable
        mov     P3,A
        jmp e
    Readtable:
        add     A,#(Table-GetTable);
        movc    A,@A+PC;  - в A  число, расположенное по  адресу «начало таблицы + цифра в А
    GetTable:
        ret
    Table:
        db 11111100B; 0                 Если в R2 была «1», то в А
        db 01100000B; 1                 попадет число 01100000B,
        db 11011010B; 2                 соответствующее  1 в таблице
        db 11110010B; 3
    e:
    loop:
    jmp loop
    end
```

Схема подключения семисегментного индикатора к микроконтроллеру представлена на рисунке 6.2.

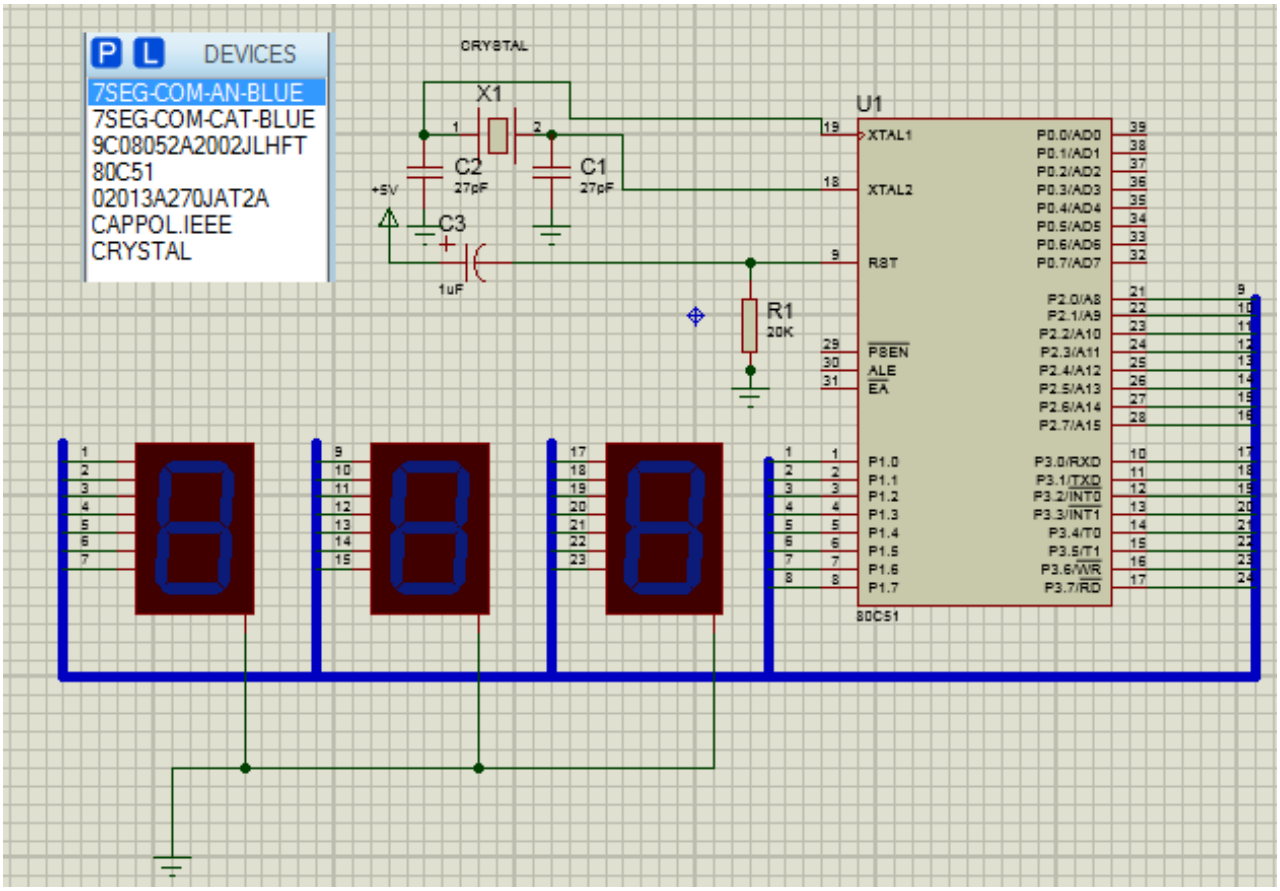

Рисунок 6.2 – Схема подключения семисегментного индикатора

6.2 Задание

1 Построить схему подключения трех семисегментных индикатора к портам P1, P2, P3 микроконтроллера 80C51.

2 Разработать программу, выполняющую следующие задачи:

− создать таблицу соответствия двоичного кода и семисегментного кода для цифр от 0 до 9;

− программно записать в регистр R0 число в Н-коде;

− выделить из введенного числа цифры по разрядам: сотни, десятки, единицы, сохранить их в регистрах R1, R2, R3;

− преобразовать каждую цифру введенного числа в семисементный код, сохранить все цифры числа в регистрах;

− вывести полученные коды на семисегментные индикаторы;

− предусмотреть возможность введения и изменение любых 8-разрядных чисел.

3 Использовать в программе работу с таблицами.

4 Каждую цифру преобразованного числа вывести в отдельный порт. К портам присоединить семисегментные индикаторы.

5 В результате выполнения программы на экране должны быть зажжены индикаторы, показывающие десятичный эквивалент первоначально введённого в Н-коде числа.

6 Предусмотреть минимальный объем программы.

6.3 Контрольные вопросы

1 Пояснить, что такое индексная адресация, привести пример использования.

2 Какие виды семисегментных индикаторов существуют, пояснить различия в их подключении.

3 Пояснить правила работы с таблицами в Ассемблере 8051.

7 Практическая работа 7. Подключение к микроконтроллеру устройств ввода и отображения информации

Цель: Изучение периферийного оборудования микроконтроллеров и способов аппаратно-программного обеспечения их работы.

7.1 Краткие теоретические сведения

Практически ни одно микропроцессорное устройство не обходится без кнопок и простейших датчиков на основе обычных контактов. С помощью этого вида периферийных элементов в микроконтроллер поступает различная информация, которая используется для изменения алгоритма работы программы. Моделью датчика может служить простой контакт, замыкание которого может рассматриваться как срабатывание. Схема подключения датчика к микроконтроллеру приведена на рисунке 7.1.

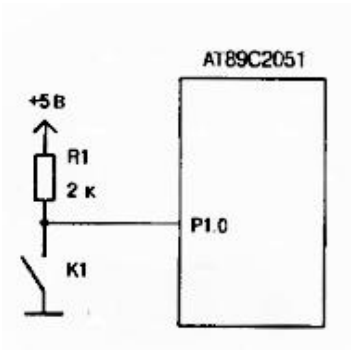

Рисунок 7.1 - Схема подключения датчика

В приведенном примере датчик подключен к линии P1.0 порта P1 микроконтроллера. Через этот вход микроконтроллер считывает состояние датчика. Принцип работы следующий: в исходном состоянии контакты датчика разомкнуты, на вход микроконтроллера через резистор R1 подается напряжение +5 В, это воспринимается микроконтроллером как логическая единица. При срабатывании датчика контакты замыкаются и на вход P1.0 подается логический

ноль от заземления. Резистор R1 служит токоограничивающим элементом, предотвращая короткое замыкание между шиной питания и общим проводом.

Пример программы для считывания состояния датчика:

```
M1:  Setb P1.0        ; записываем «1» в разряд порта P1.0.
     Mov C, P1.0      ; считываем состояние датчика в битовый аккумулятор C.
     JC M2            ; если контакты датчика разомкнуты, перейти к M2.
     Call proc1       ; если контакты замкнуты, перейти к процедуре 1.
     Jmp M1           ; возврат к началу (следующий цикл считывания).
M2:  Call proc2       ; вызов процедуры обработки размыкания контакта.
     Jmp M1           ; возврат к началу (следующий цикл считывания).
```

Этот же принцип считывания состояния контактов лежит в основе обслуживания клавиатуры. Клавиатуру представляют в виде матрицы клавиш, как показано на рисунке 7.2.

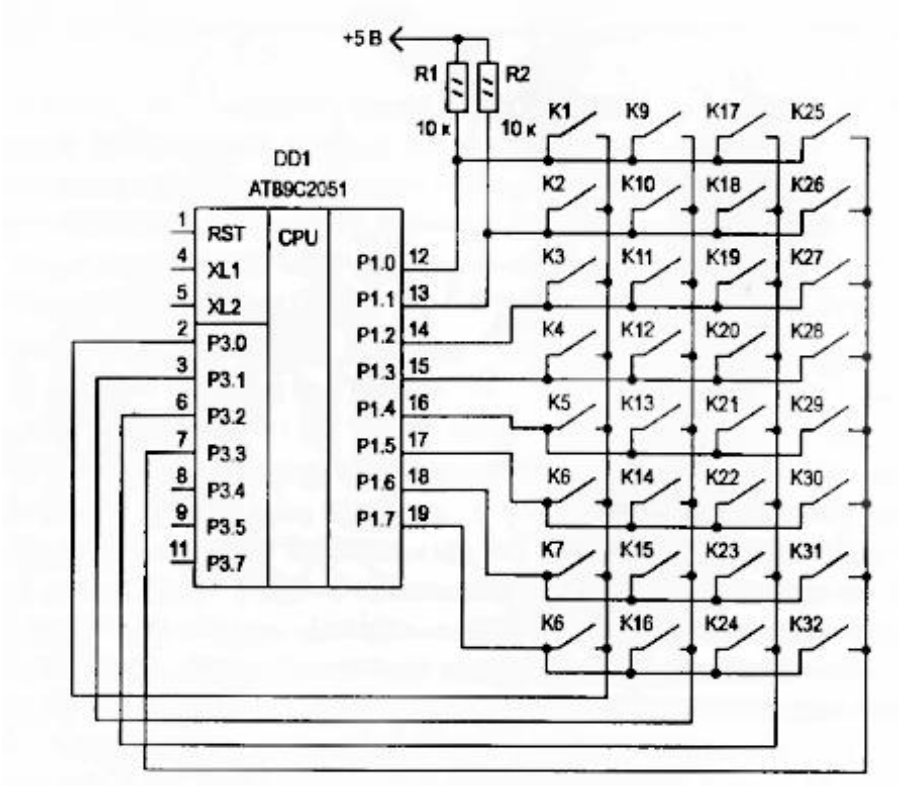

Рисунок 7.2 - Схема подключения кнопок к микроконтроллеру

В данной схеме для подключения 32-х клавиш используется 12 выводов микроконтроллера. Порт P1 работает на ввод, четыре младших разряда порта P3 работают на вывод. в исходном состоянии на выводы P3.0…P3.3 подается сигнал логической «1». На все выводы порта P1 также поданы единицы. Контроллер периодически опрашивает состояние клавиш путем изменения сигналов на выходах P3.0…P3.3 и считывания сигнала из порта P1. В случае обнаружения замыкания контактов одной из клавиш, программа выполняет закрепленные за этой клавишей действия. Если нажато несколько клавишей одновременно, это рассматривается как ошибка.

Процедура опроса клавиш поочередно переводит одну из линий P3.0…P3.3 в нулевое состояние. Сначала переводится в ноль P3.0, сразу после этого производится чтение порта P1. Если ни одна из клавиш не нажата. То все разряды порта P1 будут равны «1». Если одна из клавиш нажата. То считанное число будет отличаться от OFFH. Предположим, что нажали клавишу K1, то сигнал логического нуля поступит на вход P1.0, и младший разряд считанного числа окажется равным нулю. Число, считанное из порта P1 будет 0FEH. При нажатии других клавиш, будут другие коды. Аналогичным образом контроллер опрашивает другие вертикали клавиш.

7.2 Задание

1 Собрать схему для реализации задания по примеру рисунка 7.3.

2 Программно ввести в регистры любые два 8-разрядных числа X и Y в H-коде;

3 Подключить к портам микроконтроллера кнопки с обозначением цифр от 0 до 9, образуя, таким образом, клавиатуру; можно использовать телефонную клавиатуру, как показано на схеме рисунка 7.3.

4 Программно реализовать ввод двух чисел с помощью клавиатуры. Сохранить их в регистрах.

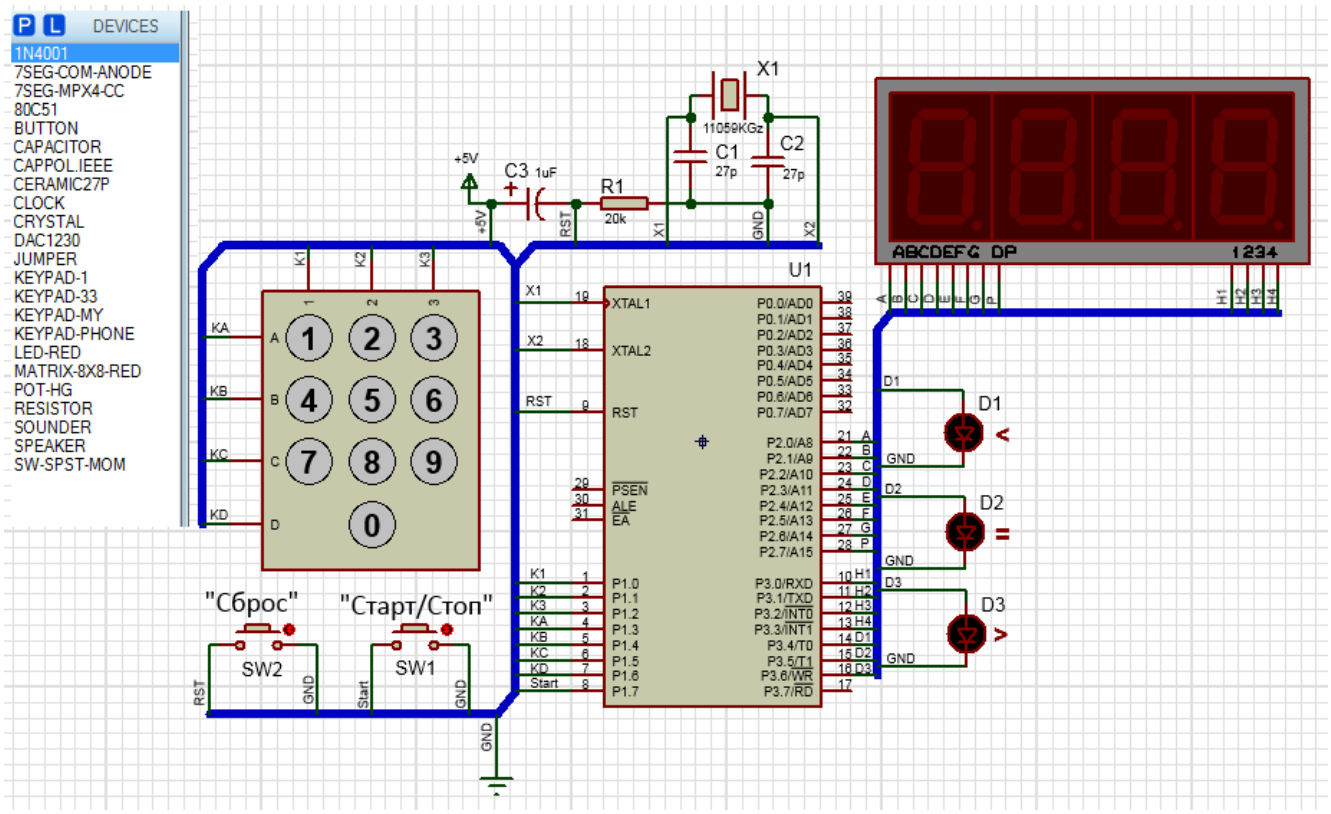

Рисунок 7.1 – Схема реализации

5 Отобразить вводимые числа на дисплее, также можно использовать семисегментные индикаторы.

6 Составить программу вычисления результата по формуле варианта, представленных в таблице 4.1.

7 Осуществить отображение результатов вычислений над 8-разрядными числами X и Y, вводимыми с клавиатуры, на дисплее или индикаторах. Вычисления провести в соответствии с вариантами формул,

Таблица 7.1 – Варианты формул для вычислений

№	1	2	3	4	5	6	7	8
	$X^2 - Y$	$\frac{X^2}{Y}$	$X^2 - Y^2$	$\frac{X+Y}{2}$	$X^2 + Y^2$	$\frac{X-Y}{X+Y}$	$X + Y^2$	$\frac{X+Y}{Y^2}$

8 Практическая работа 8. Изучение режимов работы таймеров

Цель: освоение навыков программно-аппаратной реализации работы микроконтроллера с таймерами

8.1 Краткие теоретические сведения

Программное формирование временной задержки. Временная задержка малой длительности. Процедура реализации временной задержки использует метод программных циклов. При этом в некоторый рабочий регистр загружается число, которое затем в каждом проходе цикла уменьшается на 1. Так продолжается до тех пор, пока содержимое регистра не станет равным нулю, что интерпретируется программой как момент выхода из цикла. Время задержки при этом определяется числом, загруженным в рабочий регистр, и временем выполнения команд, образующих программный цикл.

Предположим, что в управляющей программе необходимо реализовать временную задержку 99 мкс. Фрагмент программы (DELAY) оформим в виде подпрограммы, так как предполагается, что основная управляющая программа будет производить к ней многократные обращения для формирования выходных импульсных сигналов, длительность которых кратна 99 мкс:

```
DELAY:  MOV R2, #X          ; загрузка числа X
COUNT:  DJNZ R2, COUNT      ; декремент R2 и цикл, если не равно 0
RET                         ; возврат
```

Для получения требуемой временной задержки необходимо определить число X, загружаемое в рабочий регистр. Определение числа X выполняется на основе расчёта времени выполнения команд, образующих данную подпрограмму. При этом необходимо учитывать, что команды MOV и RET выполняются однократно, а число повторений команды DJNZ равно числу X. Кроме того,

обращение к подпрограмме временной задержки осуществляется по команде CALL DELAY, время исполнения которой также необходимо учитывать при подсчете временной задержки.

В описании команд микроконтроллера указывается, за сколько машинных циклов (МЦ) исполняется каждая команда: CALL - 2 МЦ, MOV - 1 МЦ, DJNZ - 2 МЦ, RET - 2 МЦ. На основании этих данных определяется суммарное число машинных циклов в подпрограмме. При тактовой частоте 12 МГц каждый машинный цикл выполняется за 1 мкс. Таким образом, подпрограмма выполняется за время 2 + 1 + 2X + 2 = 5 + 2X мкс. Для реализации временной задержки 99 мкс число X = (99 - 5)/2 = 47.

В данном случае при загрузке в регистр R2 числа 47 требуемая временная задержка (99 мкс) реализуется точно. Если число X получается дробным, то временную задержку можно реализовать лишь приблизительно. Для более точной подстройки в подпрограмму могут быть включены команды NOP, время выполнения каждой из которых равно 1 мкс.

Минимальная временная задержка, реализуемая подпрограммой DELAY, составляет 7 мкс (X = 1). Временную задержку меньшей длительности программным путем можно реализовать, включая в программу цепочки команд NOP. Максимальная длительность задержки, реализуемая подпрограммой DELAY, составляет 515 мкс (X = 255). Для реализации задержки большей длительности можно рекомендовать увеличить тело цикла включением дополнительных команд или использовать метод вложенных циклов. Так, например, если в подпрограмму DELAY перед командой DJNZ вставить дополнительно две команды NOP, то максимальная задержка составит 5 + X(2 + 1 + 1) = 5 + 4*255 = 1025 мкс (т. е. почти в два раза больше).

Временная задержка большой длительности. Задержка большой длительности может быть реализована методом вложенных циклов. Числа X и Y выбираются из соотношения T = 2 + 1 + X(1 + 2Y + 2) + 2, где T – реализуемый временной интервал в микросекундах. Максимальный временной интервал,

реализуемый таким способом, при X = Y = 255 составляет 130,82 мс, т. е. приблизительно 0,13 с.

В качестве примера рассмотрим подпрограмму (DLY100), реализующую временную задержку 100 мс.

Здесь два вложенных цикла реализуют временную задержку длительностью 5 + 195(3 + 2*254 = 99 650 мкс, а дополнительный цикл LOOPAD и команда NOP реализует задержку 350 мкс и тем самым обеспечивает точную настройку временного интервала.

```
DLY100: MOV  R1, #195       ;загрузка X
LOOPEX: MOV  R2, #254       ;загрузка Y
LOOPIN: DJNZ R2, LOOPIN     ;декремент R2 и внутренний цикл,
                            ;если (R2) не равно 0
        DJNZ R1, LOOPEX     ;декремент R1 и внешний цикл,
                            ;если (R1) не равно 0
        MOV  R3, #174       ;точная подстройка
LOOPAD: DJNZ R3, LOOPAD     ;временной
        NOP                 ;задержки
        RET                 ;возврат
```

Временная задержка длительностью 1 с. Из предыдущего примера видно, что секунда является очень большим интервалом времени по сравнению с тактовой частотой микроконтроллера. Такие задержки сложно реализовать методом вложенных циклов, поэтому их обычно набирают из точно подстроенных задержек меньшей длительности. Например, задержку в 1 с можно реализовать десятикратным вызовом подпрограммы, реализующей задержку 100 мс:

```
DLY1:    MOV R4, #10        ;загрузка в R4 числа вызовов DELAY
LOOP:    CALL DLY100        ;задержка 100 мс
         DJNZ R4, LOOP      ;R4 – 1 и цикл, если (R4) не равно 0
```

Погрешность подпрограммы DLY1 составляет 21 мкс. Для очень многих при-менений это достаточно высокая точность, хотя реализованные на основе этой программы часы астрономического времени за сутки «убегут» примерно на 1,8 с.

Формирование временной задержки таймером.

Недостатком программного способа реализации временной задержки является нерациональное использование ресурсов микроконтроллера: во время формирования задержки он практически простаивает, так как не может решать никаких задач управления объектом. В то же время аппаратные средства позволяют реализовать временные задержки на фоне работы основной программы.

На вход таймера/счетчика (Т/С) могут поступать сигналы синхронизации с частотой 1 МГц (Т/С в режиме таймера) или сигналы от внешнего источника (Т/С в режиме счетчика). Оба эти режима могут быть использованы для формирования задержек. Если использовать Т/С в режиме таймера полного формата (16 бит), то можно получить задержки в диапазоне 1 – 65 536 мкс.

В качестве примера рассмотрим организацию временной задержки 50 мс (TIMER). Предполагается, что бит IE.7 установлен, т. е. снята блокировка всех прерываний. Обратите внимание, что здесь использована команда перевода микроконтроллера в режим холостого хода, который прекращается после истечения 50 мс. Этот режим реализован в микроконтроллере Intel 80C51.

```
;организация перехода к метке NEXT при переполнении T/C0
        ORG  0BH        ;адрес вектора прерывания от T/C0
        CLR  TCON.4     ;останов T/C0
        RETI            ;выход из обработчика прерывания

        ORG  30         ;начальный адрес программы
TIMER:  MOV  TMOD, #01H                 ;настройка T/C0
        MOV  TL0, #LOW(NOT(50000-1))    ;загрузка таймера для
        MOV  TH0, #HIGH(NOT(50000-1))   ;задержки 50 мс
        SETB TCON.4     ;старт T/C0
        SETB IE.1       ;разрешение прерывания от T/C0
        SETB PCON.0     ;перевод в режим холостого хода
NEXT:   ...
```

8.2 Задание

1 Разработать модель микропроцессорного устройства для измерения временного интервала, длительность которого не превышает 10 секунд.

2 Задать длительность интервала с помощью клавиатуры.

3 Начало и окончание интервала задать нажатием кнопки.

4 Измерить длительность временного интервала с точностью до 0,01 секунды. Использовать таймер, входящий в состав микроконтроллера.

5 Отобразить на дисплее ход изменения времени в десятичной системе счисления на четырехпозиционном индикаторе (два знака после запятой).

6 Сравнить измеренное время с эталоном, имеющим два знака после запятой. Обеспечить возможность изменять значение эталона перед запуском программы.

7 Реализовать индикацию результата сравнения с помощью светодиодов с обозначением «>»; «<»; «=».

Пример схемы приведен на рисунке 8.1

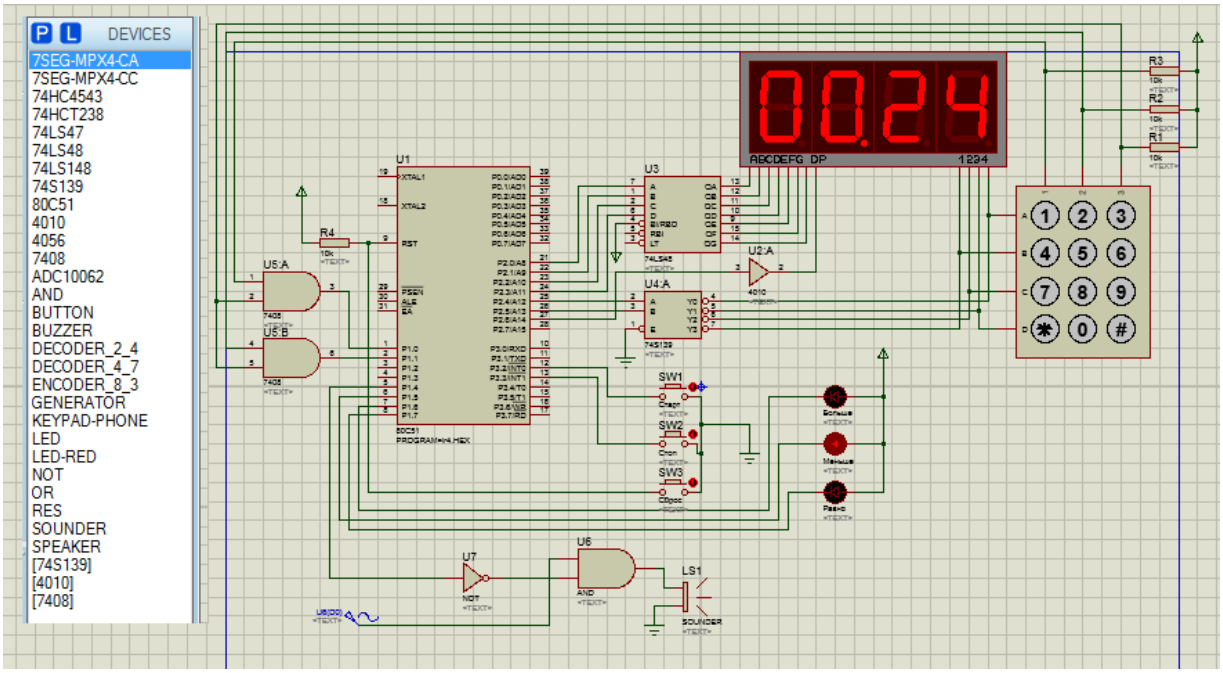

Рисунок 8.1 – Пример реализации

9 Практическая работа 9. Разработка охранной сигнализации автомобиля на основе микроконтроллера

Цель: Освоение работы с прерываниями, программная реализация временных задержек.

9.1 Краткие теоретические сведения

Система прерываний микроконтроллера Intel 8051

Источников прерываний для микроконтроллера 8051 пять. Упрощенная схема прерываний микро-ЭВМ 8051 показана на рисунке 9.1.

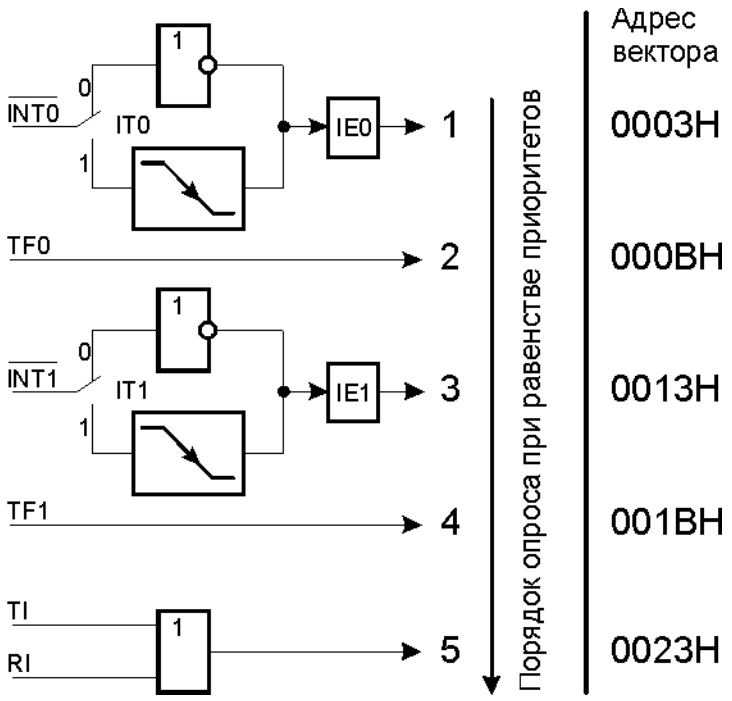

Рисунок 9.1 - Упрощенная схема прерываний микро-ЭВМ 8051

Внешние прерывания INT 0 и INT 1 могут быть вызваны либо уровнем, либо переходом сигнала из 1 в 0 на входах 8051 в зависимости от значений управляющих бит IT0 и IT1 в регистре TCON.

От внешних прерываний устанавливаются флаги IE0 и IE1 в регистре TCON, которые инициируют вызов соответствующей программы обслуживания прерывания. Сброс этих флагов выполняется аппаратно только в том случае, если прерывание было вызвано по переходу (срезу) сигнала. Если же прерывание вызвано уровнем входного сигнала, то сбросом флага I должна управлять соответствующая подпрограмма обслуживания прерывания путем воздействия на источник прерывания с целью снятия им запроса.

Флаги запросов прерывания от таймеров TF0 и TF1 сбрасываются автоматически при передаче управления подпрограмме обслуживания. Флаги запросов прерывания RI и TI устанавливаются блоком управления приемопередатчика аппаратно, но сбрасываться должны программным путем.

Прерывания могут быть вызваны или отменены программой, так как все названные флаги программно доступны и могут быть установлены/ сброшены программой с тем же результатом, как если бы они были установлены/сброшены аппаратными средствами.

В блоке регистров специальных функций есть два регистра, предназначенных для управления режимом прерываний IE и уровнями приоритета IP. В таблицах 9.1 и 9.2 описаны назначения битов регистров IE и IP.

Таблица 9.1 – Регистр масок прерывания (IE)

Символ	Позиция	Имя и назначение
1	2	3
EA	IE.7	Снятие блокировки прерывания. Сбрасывается, программно для запрета всех прерываний независимо от состояний IE.4 - IE.0
	IE.6	Не используется
	lE.5	Не используется
ES	IE.4	Бит разрешения прерывания, от приемопередатчика Установка/сброс программой для разрешения/запрета прерываний от флагов TI или RI .
ET1	IE.3	Бит разрешения прерывания от таймера. Установка/сброс программой для разрешения/запрета прерываний от таймера 1
EX1	IE.2	Бит разрешения внешнего прерывания 1. Установка/сброс программой для разрешения/запрета прерывания 1
ET0	IE.1	Бит разрешения прерывания от таймера 0. Установка/сброс программой для разрешения/запрета прерываний от таймера 0
EX0	IE.0	Бит разрешения внешнего прерывания 0. Установка/сброс программой для разрешения/запрета прерывания 0

Таблица 9.2 – Регистр приоритетов прерываний (IP)

Символ	Позиция	Имя и назначение
-	IP.7 - IP.5	Не используется
PS	IP.4	Бит приоритета приемопередатчика. Установка/сброс программой для присваивания прерыванию от приемопередатчика высшего/низшего приоритета.
PT1	IP.3	Бит приоритета таймера 1. Установка/сброс программой для присваивания прерыванию от таймера 1 высшего/низшего приоритета.
PX1	IP.2	Бит приоритета внешнего прерывания 1. Установка/сброс программой для присваивания высшего/низшего приоритета внешнему прерыванию INT1.
PT0	IP.1	Бит приоритета таймера 0. Установка/сброс программой для присваивания прерыванию от таймера 0 высшего/низшего приоритета.
PX0	IP.0	Бит приоритета внешнего прерывания 0. Установка/сброс программой для присваивания высшего/низшего приоритета внешнему прерыванию INT0.

Возможность программной установки/сброса любого управляющего бита в этих двух регистрах делает систему прерываний 8051 исключительно гибкой.

Система прерываний формирует аппаратный вызов (LCALL) соответствующей подпрограммы обслуживания, если она не заблокирована одним из следующих условий:

- в данный момент обслуживается запрос прерывания равного или высокого уровня приоритета;
- текущий машинный цикл - не последний в цикле выполняемой команды;
- выполняется команда RETI или любая команда, связанная с обращением к регистрам IE или IP.

Отметим, что если флаг прерывания был установлен, но по одному из указанных выше условий не получил обслуживания и к моменту окончания блокировки уже сброшен, то запрос прерывания теряется и нигде не запоминается. По аппаратно сформированному коду LCALL система прерывания помещает в стек только содержимое счетчика команд (PC) и загружает в него адрес вектора соответствующей подпрограммы обслуживания. По адресу вектора должна быть расположена команда безусловной передачи управления (JMP) к начальному адресу подпрограммы обслуживания прерывания. В случае необходимости она должна начинаться командами записи в стек (PUSH) слова

состояния программы (PSW), аккумулятора, расширителя, указателя данных и т.д. и должна заканчиваться командами восстановления из стека (POP). Подпрограммы обслуживания прерывания должны завершаться командой RETI, по которой в счетчик команд перезагружается из стека сохраненный адрес возврата в основную программу. Команда RET также возвращает управление прерванной основной программе, но при этом не снимут блокировку прерываний, что приводит к необходимости иметь программный механизм анализа окончания процедуры обслуживания данного прерывания.

9.2 Задание

Разработать модель микропроцессорного устройства охранной сигнализации автомобиля (УОС) на базе микроконтроллера 8051, удовлетворяющую следующим требованиям. Устройство должно включить тревожную сигнализацию в том случае, если открывается капот, крышка багажника или любая из трех дверей, за исключением двери водителя. При открывании двери водителя обеспечить 10-секундную задержку, и если за это время на двухпозиционном цифронабирателе не будет набрана секретная комбинация цифр (например 27) и после этого не будет нажата кнопка ввода, то устройство должно включить тревожную сигнализацию. Тревожная сигнализация включается и в том случае, если на цифронабирателе установлен ошибочный код. Включение и отключение УОС осуществляется кнопкой подачи электропитания на микроконтроллер. Устройство должно обеспечить охрану автомобиля после включения электропитания микроконтроллера и закрывания дверей. Для этого под капотом, крышкой багажника и в проемах всех дверей установлены концевые выключатели, сигнализирующие о состоянии объекта (открыто/закрыто). Тревожная сигнализация реализуется миганием двух светодиодов разного цвета.

Реализовать программную обработку прерываний по переполнению таймера (при отсчете 10 с) и от внешних событий (открытие дверей и т.д.).

Использовать следующие ресурсы: кнопка включения электропитания микроконтроллера; кнопка ввода после набора секретного кода; ресурсы,

имитирующие открытые/закрытые двери, капот, багажник (могут быть использованы кнопки, логические элементы); цифронабиратель для ввода двухзначного кода либо клавиатура с цифрами; два светодиода разного цвета.

Представить подробную схему подключения ресурсов, алгоритм и текст управляющей программы.

Пример схемы реализации представлен на рисунке 9.2.

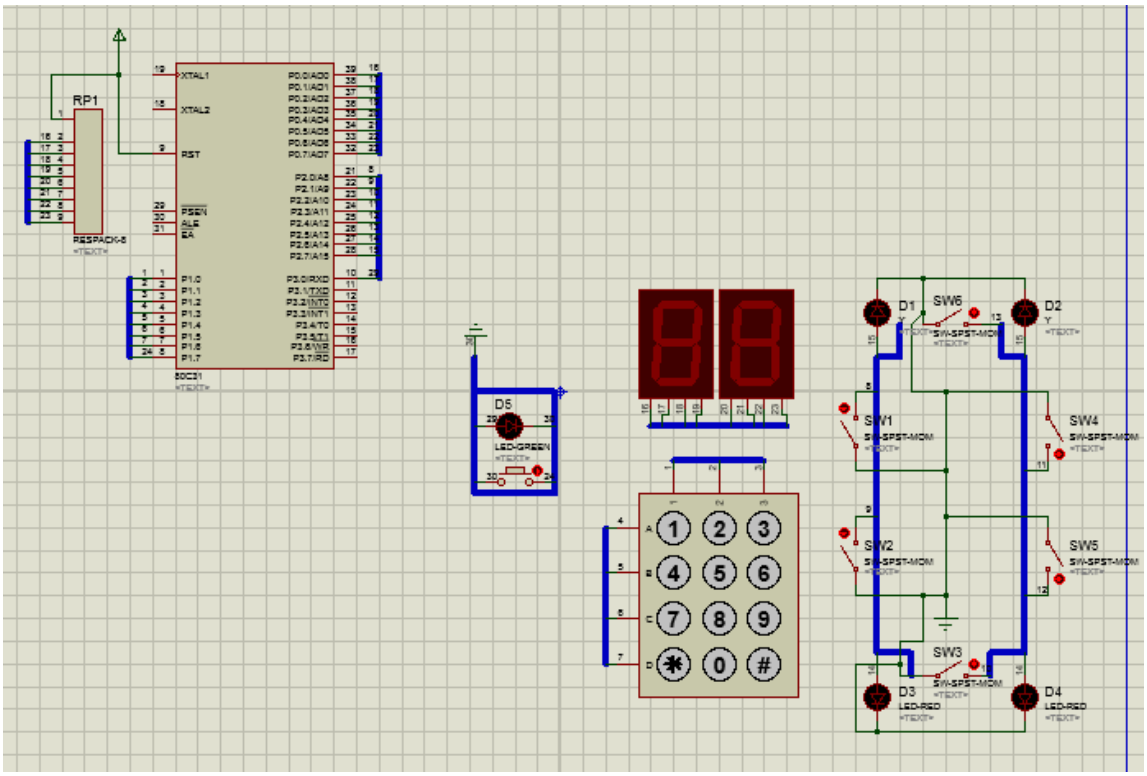

Рисунок 9.2 - Пример схемы реализации

9.3 Контрольные вопросы

1 Где расположена область памяти для размещения программ обработки прерываний? Сколько байт памяти выделено для каждого вида прерываний в контроллере 8051?

2 Какие регистры специальных функций микроконтроллера 8051 предназначены для управления режимом прерываний? Опишите их.

3 Какие условия блокируют аппаратный вызов прерывания?

4 Каков порядок работы микроконтроллера при обслуживании прерывания?

Список использованных источников

1 Белов, А.В. Конструирование устройств на микроконтроллерах / А.В. Белов. - Спб.: Наука и техника, 2005. – 256 с.

2 Бурькова Е.В. Характеристика учебно-исследовательских стендов по курсам «Схемотехника» и «Микропроцессорные системы» / Е.В. Бурькова, Р.Р. Кадыров. Материалы XI Всероссийской научно-практической конференции «Современные информационные технологии в науке, образовании и практике». – Оренбург, 2014. – С. 125-128. Издательство: Издательско-полиграфический комплекс "Университет" (Оренбург)

4 Гусев, В.Г. Электроника и микропроцессорная техника / В.Г. Гусев, Ю.М. Гусев. - М.: КноРус, 2015. - 1247 с.

5 Магда, Ю.С. Микроконтроллеры серии 8051: практический подход / Ю.С. Магда. – М.: ДМК Пресс, 2008. – 228 с.

6 Мелехин, В.Ф. Вычислительные машины, системы и сети: учебник для вузов / В.Ф. Мелехин, Е.Г. Павловский. - 2-е изд., стер. – М.: Академия, 2007. - 560с.

7 Микроконтроллеры 8051, PIC, AVR и ARM: отличия и особенности. Режим доступа: http://digitrode.ru/computing-devices/mcu_cpu/1253-mikrokontrollery-8051-pic-avr-i-arm-otlichiya-i-osobennosti.html

8 Угрюмов, Е.П. Цифровая схемотехника: учеб. пособие / Е. П. Угрюмов .- 2-е изд., перераб. и доп. - СПб. : БХВ-Петербург, 2007. - 800 с.

9 Хартов, В.Я. Микропроцессорные системы: Учебное пособие / В.Я. Хартов. - М.: Academia, 2017. - 320 с.

10 Цилькер, Б.Я. Организация ЭВМ и систем: учебник / Б.Я. Цилькер, С.А. Орлов. – СПб.: Питер, 2006. - 668 с.

11 Proteus in Education. Режим доступа: https://www.labcenter.com/