МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение
высшего профессионального образования
«Оренбургский государственный университет»

Кафедра вычислительной техники

Е.В. БУРЬКОВА

МОДЕЛИРОВАНИЕ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНОМУ ПРАКТИКУМУ

Рекомендовано к изданию Редакционно-издательским советом
государственного образовательного учреждения
высшего профессионального образования
«Оренбургский государственный университет»

Оренбург 2007

УДК 681.3(07)
ББК 32.973.26-04 я 73
Б 91

Рецензент
доктор технических наук, профессор Т.З. Аралбаев

Бурькова Е.В.
Б 91 Моделирование микропроцессорных систем: методические указания к лабораторному практикуму / Е.В. Бурькова – Оренбург, ГОУ ОГУ, 2007. – 39 с.

В методических указаниях представлены теоретические сведения об архитектуре микроконтроллера Intel 8051, дана характеристика современных средств разработки и моделирования микропроцессорных систем, описание интегрированной среды проектирования – симулятора UMPS. Методические указания содержат материалы для проведения лабораторного практикума по дисциплине «Микропроцессорные системы» с применением программного симулятора UMPS. Методические указания предназначены для студентов специальностей направления 230100 «Информатика и вычислительная техника», а также могут быть полезными для инженеров, занятых проектированием вычислительных систем на базе микроконтроллеров.

ББК 32.973.26-04 я 73

Содержание

Введение

Микропроцессорная техника постепенно замещает и вытесняет традиционную цифровую на «жесткой логике». Универсальность, гибкость, простота проектирования аппаратуры, практически неограниченные возможности по усложнению алгоритмов обработки информации – всё это объясняет столь широкое использование микропроцессорных систем в современном производстве.

Большое место в микропроцессорной технике занимают микроконтроллеры. В настоящее время происходит настоящая революция, оказавшая значительное влияние на каждого из нас – это автоматизация практически всей окружающей нас среды с помощью дешёвых и мощных микроконтроллеров. Микроконтроллер – это самостоятельная компьютерная система, которая содержит процессор, память, вспомогательные схемы и устройства ввода-вывода данных, размещенные в общем корпусе. Микроконтроллеры, используемые в различных устройствах, выполняют функции интерпретации данных, поступающих с клавиатуры пользователя или от датчиков, определяющих параметры окружающей среды, обеспечивают связь между различными устройствами системы и передают данные другим приборам. Применение микроконтроллеров позволяет значительно снизить количество и стоимость используемых материалов и комплектующих изделий, что обеспечит снижение себестоимости конечной продукции. Использование микроконтроллеров может существенно увеличить привлекательность продукции для потребителя благодаря реализации «дружественного интерфейса» при относительно небольших дополнительных затратах. Обеспечивается также возможность расширения области применения выпускаемой продукции путем использования одних и тех же аппаратных средств с разнообразным программным обеспечением, специализированным для реализации различных функций.

Проектирование любой сложной системы начинается с создания математической модели и исследования ее на ЭВМ. В настоящее время при проектировании микропроцессорных систем широко применяются методы аналитического и имитационного моделирования с использованием автоматизированных систем. Одним из плодотворных подходов к оценке важнейших конструктивных показателей служит вероятностное моделирование, которому посвящены исследования таких ученых как: О.И. Авен, И.Н. Альянах, В.Н. Тарасов. Наличие в микропроцессорной системе, как аппаратных, так и программных средств обуславливает ряд специфических особенностей, присущих процессу ее создания. Он существенно отличается от проектирования традиционных электронных устройств, не предполагающих программное обеспечение. В отличие от традиционного подхода, когда все функции, возлагаемые на устройство, достигаются чисто аппаратными средствами и другой альтернативы просто не существует, при аппаратно-программной реализации выполняемые функции оптимально располагаются между программными и аппаратными средствами микропроцессорной системы.

Идея единства программного и аппаратного обеспечения систем на базе микроконтроллеров является очень важной и получила свое воплощение в интегрированных средах разработки. Важным преимуществом интегрированной среды при проектировании микропроцессорной системы является объединение инструментальных средств разработки программного обеспечения с инструментальными средствами разработки аппаратного обеспечения.

Первичным инструментом для отладки приложений является симулятор. Это программа, которая выполняет программный код, используя модель микроконтроллера, и позволяет при этом пользователю наблюдать за ходом выполнения программы. Симулятор является частью интегрированной среды программирования, где отладка происходит на уровне исходного текста.

Часто проблема использования симулятора заключается в изобилии предоставляемых возможностей: различные контрольные точки, возможные варианты вывода, формирование входных тестовых комбинаций, вывод на дисплей содержимого регистров и обрабатываемых данных, окна модификации и многое другое. Самый эффективный способ познакомиться с симулятором - это использовать его при разработке конкретной программы.

Важное преимущество симуляторов — возможность многократного воспроизведения рабочих ситуаций. Если надо понять, почему участок программы работает некорректно, можно повторять этот участок снова и снова до тех пор, пока ошибка не будет обнаружена.

Данные методические указания посвящены моделированию микропроцессорных систем на базе микроконтроллеров Intel 8051 с использованием симулятора UMPS (Universal Microprocessor Program Simulator). UMPS использует уникальный метод симуляции приложений. Вместо разработки специального тестового файла UMPS позволяет создать виртуальную схему, с которой взаимодействует приложение. В симуляторе помимо программирования, компиляции и отладки программы предоставляется возможность подключения к микроконтроллеру виртуальных ресурсов: светодиодов, кнопок, семисегментных индикаторов, LCD-дисплея и др.

Применяя интегрированную среду UMPS, студенты имеют возможность моделировать микропроцессорное устройство от начального этапа (постановки задачи) до программной реализации всех возможных режимов работы на модели устройства с отладкой управляющей программы. Результатом работы является полностью собранная и отлаженная виртуальная аппаратно-программная модель микропроцессорной системы.

Использование интегрированных сред разработки вычислительных систем для организации учебного процесса по курсу «Микропроцессорные системы» позволяет значительно повысить готовность студентов к решению практико-ориентированных задач высокого уровня сложности, способствует вовлечению студентов в профессиональную деятельность.

1 Основные сведения о микроконтроллерах

Микроконтроллер (МК) – это самостоятельная компьютерная система, которая содержит процессор, устройства памяти, вспомогательные схемы и устройства ввода/ вывода данных, размещенные в общем корпусе.

Основные типы:

- встраиваемые 8-разрядные МК;
- 16 и 32-разрядные МК;
- цифровые сигнальные процессоры (DSP).

Встраиваемые МК. Промышленностью выпускается очень широкая номенклатура встраиваемых (embedded) МК. В этих МК все необходимые ресурсы (память, устройства ввода/вывода и т.д.) располагаются на одном кристалле с процессорным ядром. Всё, что необходимо сделать – это подать питание и тактовые сигналы. Встраиваемые микроконтроллеры могут базироваться на существующем микропроцессорном ядре или на процессоре, разработанном специально для данного микроконтроллера.

Основное назначение встраиваемых МК – обеспечить с помощью недорогих средств гибкое программируемое управление объектами и связь с внешними устройствами. Эти МК не предназначены для реализации комплекса сложных функций, но они способны обеспечить эффективное управление во многих областях применения.

Встраиваемые МК содержат значительное число вспомогательных устройств, благодаря чему обеспечивается их включение в реализуемую систему с использованием минимального количества дополнительных компонентов. В состав этих МК обычно входят:

- схема начального запуска процессора (RESET);
- генератор тактовых импульсов;
- центральный процессор;
- память программ (E (E) P)ROM и программный интерфейс;
- память данных RAM;
- средства ввода-вывода данных;
- таймеры, фиксирующие число командных циклов.

Общая структура микроконтроллера показана на рисунке 1.1.

Более сложные встраиваемые микроконтроллеры могут дополнительно реализовать следующие возможности:

- встроенный монитор/отладчик программ;
- внутренние средства программирования памяти программ (ROM);
- обработка прерываний от различных источников;
- аналоговый ввод – вывод;
- последовательный ввод – вывод (синхронный и асинхронный);
- параллельный ввод – вывод (включая интерфейс с компьютером);
- подключение внешней памяти (микропроцессорный режим).

Рисунок 1.1 – Общая структура микроконтроллера

Все эти возможности значительно увеличивают гибкость применения МК и делают более простым процесс разработки систем на их основе. Но для реализации этих возможностей требуется расширение функций внешних выводов. Типичные значения max частоты тактовых сигналов составляют для различных микроконтроллеров 10-20 МГц. Главным фактором, ограничивающим их скорость, является время доступа к памяти, применяемой в МК.

Микроконтроллеры с внешней памятью. Структура микроконтроллера с внешней памятью показана на рисунке 1.2.

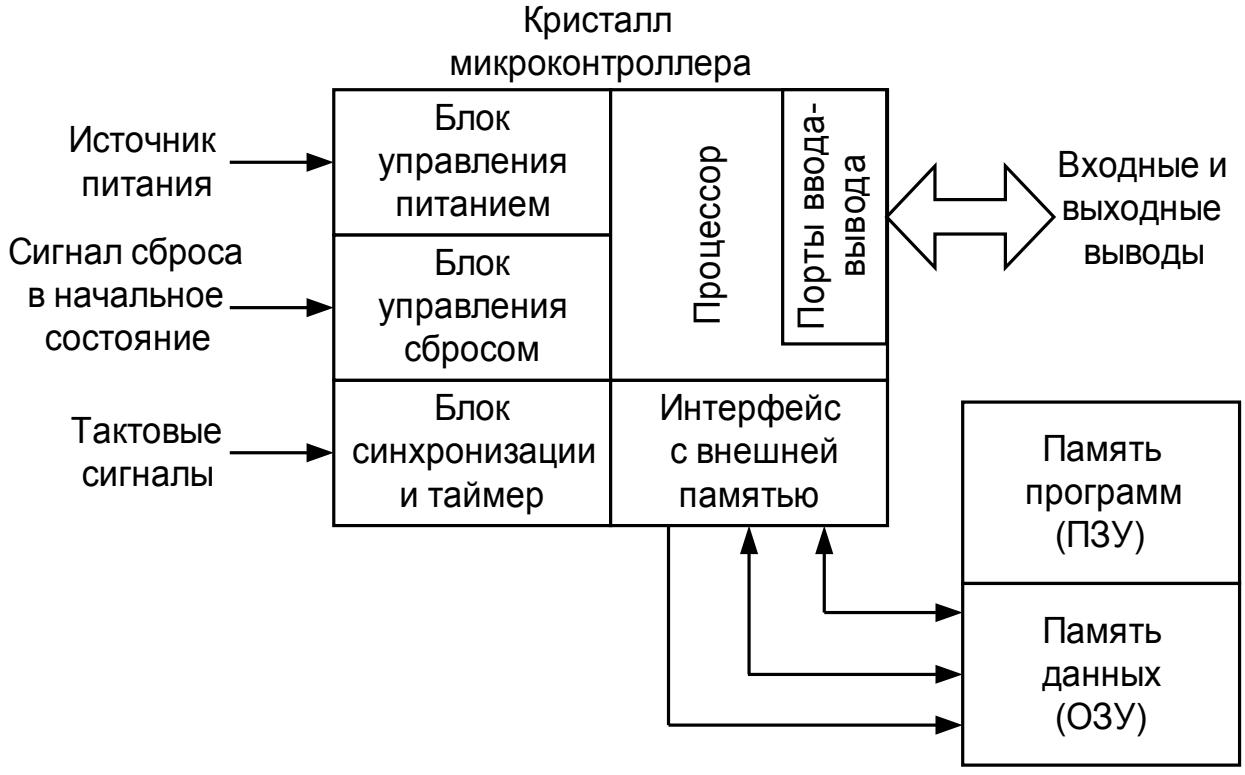

Рисунок 1.2 - Структура микроконтроллера с внешней памятью

Некоторые МК (особенно 16- и 32-разрядные) используют только внешнюю память, которая включает в себя как память программ (ROM), так и некоторый объём памяти данных (RAM), требуемый для данного применения.

Классическим примером такого МК является Intel 80188. По существу он представляет собой микропроцессор 8088, который использовался в компьютерах IBM PC, интегрированный на общем кристалле с дополнительными схемами, реализующими ряд стандартных функций, таких как прерывания и прямой доступ к памяти (DMA). Цель создания 80188 состояла в том, чтобы объединить в одном корпусе все устройства, необходимые для реализации микропроцессорных систем.

Микроконтроллеры с внешней памятью предназначены для других применений, нежели встраиваемые микроконтроллеры. Эти применения обычно требуют большого объёма памяти (RAM) и небольшого количества устройств (портов) ввода-вывода. Для МК с внешней памятью наиболее подходящими являются приложения, в которых критическим ресурсом является память, а не число входов-выходов общего назначения, тогда как для встраиваемых микроконтроллеров имеет место противоположная ситуация.

Цифровые сигнальные процессоры. Цифровые сигнальные процессоры (DSP) – относительно новая категория процессоров. Назначение DSP состоит в том, чтобы получать текущие данные от аналоговой системы и формировать соответствующий отклик. DSP и их арифметико-логическое устройство (ALU) работают с очень высокой скоростью, что позволяет осуществить обработку данных в реальном масштабе времени. DSP часто используют в активных шумоподавляющих микрофонах, которые устанавливаются в самолётах или для подавления раздвоения изображения в телевизионных сигналах.

В разнообразных DSP можно найти особенности, присущие как встраиваемым микроконтроллерам, так МК с внешней памятью. DSP не предназначены для автономного применения, обычно они входят в состав систем, используясь в качестве устройств управления внешним оборудованием, а также для обработки входных сигналов.

2 Микроконтроллер Intel 8051

2.1 Общие сведения

Несмотря на непрерывное развитие и появление все новых и новых 16- и 32-разрядных МК, наибольшая доля мирового микропроцессорного рынка остается за 8-разрядными устройствами. В 2002 году общий мировой объем продаж МК всех типов составил $ 15,4 млрд., при этом 48,6% пришлось на долю 8-разрядных кристаллов. Это в 2,5 раза больше объема продаж ближайших конкурентов: 16-разрядных МК ($ 2,1 млрд.) и DSP ($ 2,4 млрд.). За последние пять лет лидирующее положение 8-разрядных МК на мировом рынке сохраняется.

Первые МК 8051 были выпущены Intel в 1980 г. Удачный выбор периферийных устройств, возможность гибкого выбора внешней и внутренней памяти программ и приемлемая цена обеспечили этому МК успех на рынке. Важную роль в достижении высокой популярности семейства 8051 сыграла открытая политика фирмы Intel, направленная на широкое распространение лицензий на ядро 8051 среди ведущих полупроводниковых компаний мира. Каждая фирма производила усовершенствования 8051 с целью расширения функциональных возможностей или повышения скорости. В результате на сегодняшний день существует более 200 модификаций МК семейства 8051, выпускаемых почти 20 компаниями. Эти модификации включают в себя кристаллы с широким спектром периферии: от простых 20-выводных устройств с одним таймером и 1 Кбайтом программной памяти до сложнейших 100-выводных кристаллов с 10-разрядными АЦП, массивами таймеров-счетчиков, аппаратными 16-разрядными умножителями и 64 Кбайтами программной памяти на кристалле. Основные направления развития: увеличение быстродействия, снижение напряжения питания и энергопотребления, увеличение объема ОЗУ и FLASH-памяти на кристалле с возможностью внутрисхемного программирования и т. д.

Основными производителями клонов семейства 8051 являются фирмы: Philips, Siemens, Intel, Atmel, Dallas, Temic, Oki, AMD, MHS, Gold Star, Winbond, Silicon Systems, Dallas Semiconductor и др.

В России аналоги 8051 – 1816ВЕ31, 51 (Воронеж), 1830ВЕ31, 51 (Киев), 1834ВЕ31 (Минск), 1850ВЕ31 (Новосибирск).

2.2 Базовая архитектура микроконтроллера 8051

Структурная схема микроконтроллера 8051 приведена на рисунке 2.1

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.

К адресному пространству памяти данных примыкает адресное пространство регистров специальных функций SFR (Special Function Register) (таблица 2.1).

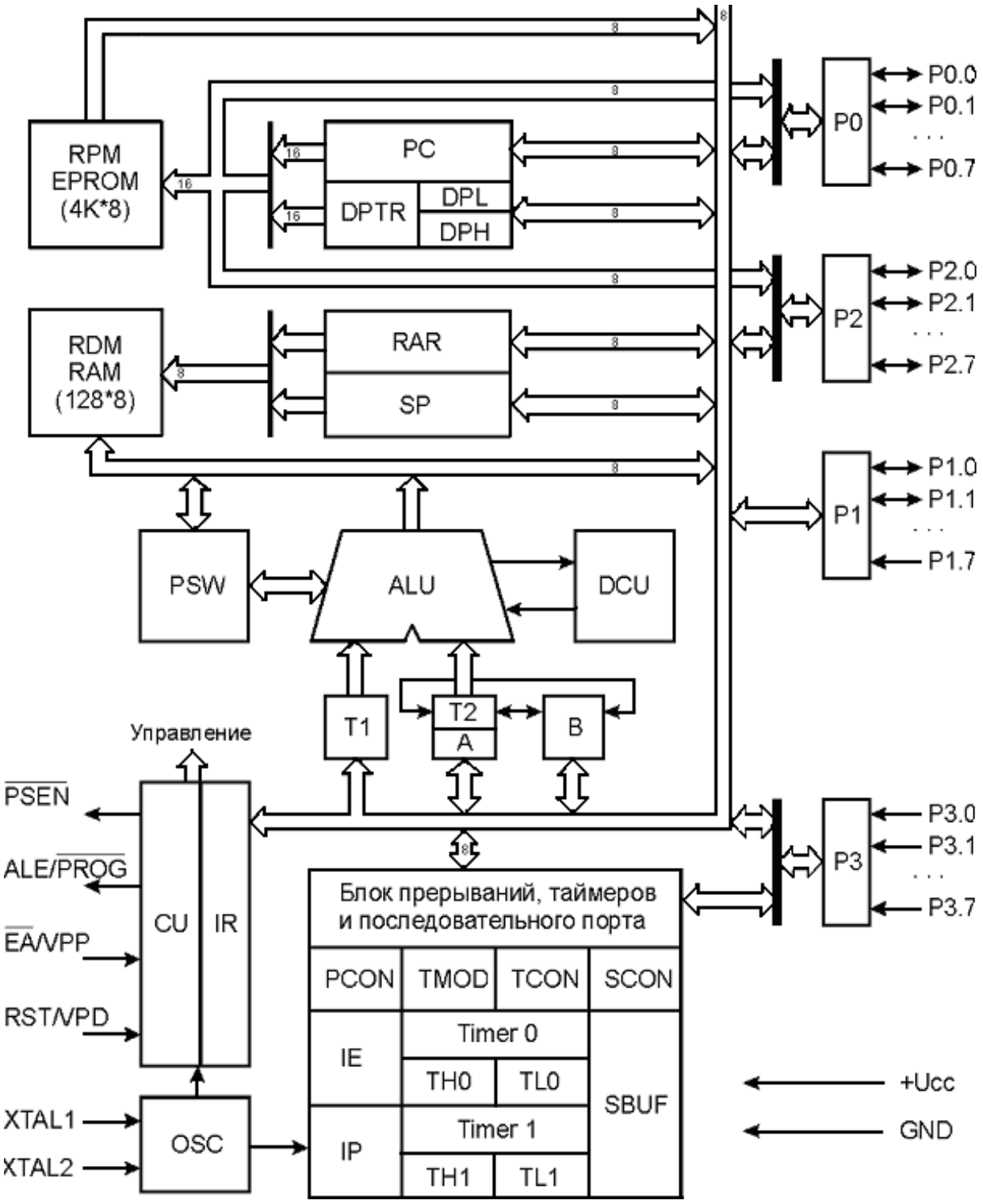

Рисунок 2.1 – Структурная схема микроконтроллера 8051

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**.

Таблица 2.1 – Регистры специальных функций Intel 8051

Адрес	Символ	Наименование
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)

Для управления режимами энергопотребления МК используется регистр **PCON**. Регистры **IP** и **IE** управляют работой системы прерываний МК, регистры **SBUF** и **SCON** — работой приемопередатчика последовательного порта. Регистр-указатель стека **SP** в МК рассматриваемого семейства — восьмибитный. Он может адресовать любую область внутренней памяти данных. В отличие от микропроцессора КР580ВМ80, у микроконтроллера семейства 8051 стек «растет вверх», т.е. перед выполнением команды PUSH или CALL содержимое SP инкрементируется, после чего производится запись информации в стек. Соответственно при извлечении информации из стека регистр SP декрементируется после извлечения информации. В процессе инициализации МК после сигнала сброса или при включении питающего напряжения в SP заносится

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

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

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

Базовая архитектура микроконтроллера 8051 представлена на рисунке 2.2. Объем расположенной на кристалле памяти данных – 128 байт.

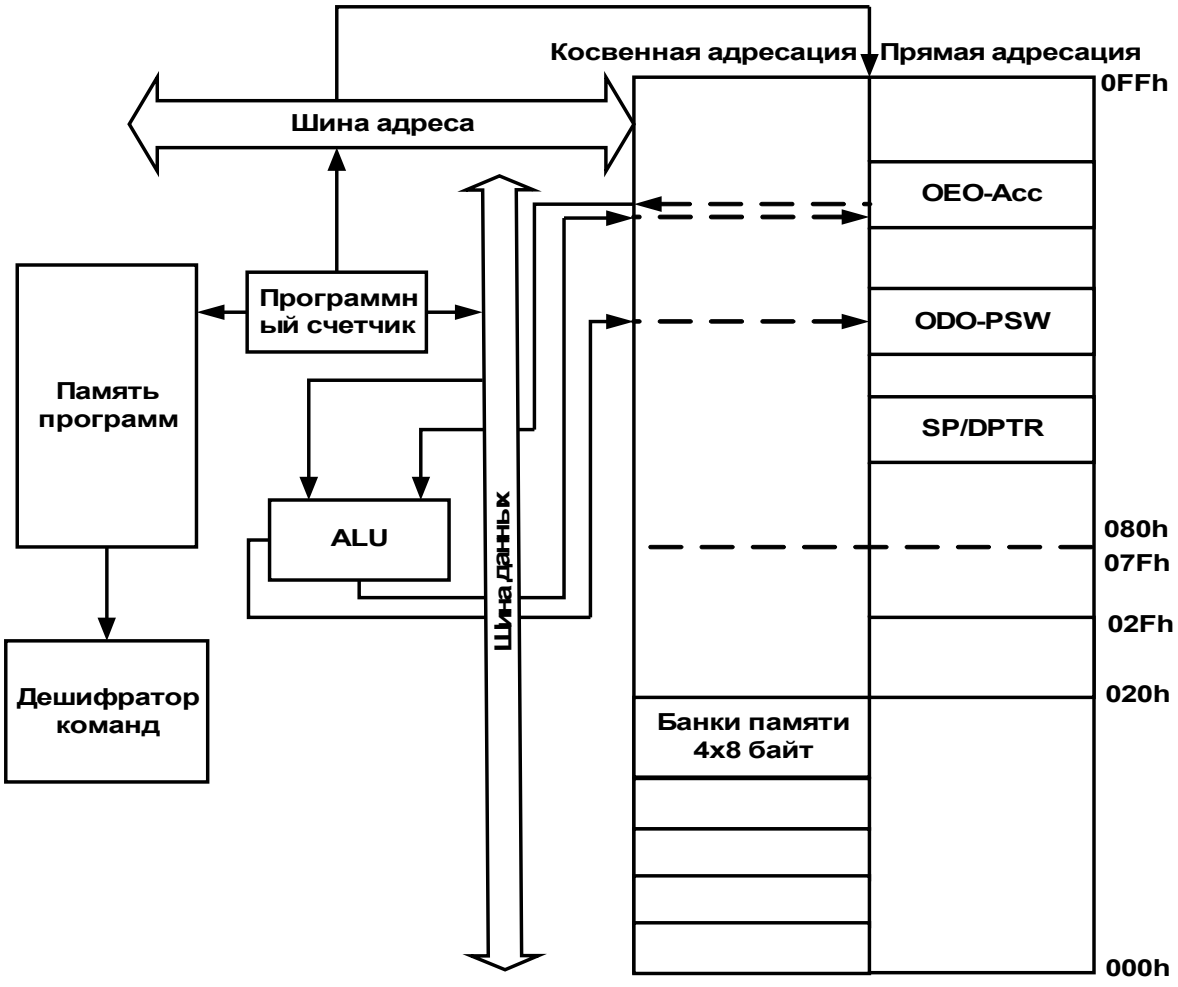

Рисунок 2.2 - Базовая архитектура микроконтроллера Intel 8051

Первые 32 байта организованы в 4 банка регистров общего назначения (РОН) - банк 0 – банк 3. Каждый банк состоит из 8 регистров R0-R7. В любой момент программе доступен только один банк. Следующая часть ОЗУ – побитно адресуемая память RAM 020 – 02F. Эти 16 байт могут адресоваться побитно и побайтно, доступны только при прямой адресации. К адресному пространству памяти данных примыкает адресное пространство регистров специальных функций. В диапазоне адресов с 080 по 0FF расположены служебные регистры. Эта область регистров зарезервирована для процессора и периферийных устройств. Прямая адресация памяти данных в диапазоне адресов 80-FF вызовет обращение к служебным регистрам. В стандартный набор этих регистров входят: указатель стека, индексный указатель DPTR, регистры портов и т.д. Эти регистры допускают адресацию отдельных бит при использовании команд операций над битами.

Отметим, что регистры занимают только часть 128-байтового адресного пространства. Те ячейки памяти с адресами 80H-0FFH, которые не заняты регистрами, физически отсутствуют, на кристаллах микроконтроллеров семейства 8051 при обращении к ним можно прочитать лишь код команды возврата.

2.3 Регистр флагов (PSW)

Перечень флагов, их символические имена и условия формирования приведены в таблице 2.2.

Таблица 2.2 – 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 приоритета. 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>
</table>

C	PSW.7	Флаг переноса. Устанавливается и сбрасывается как аппаратурно, так и программным путем

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.

2.4 Устройство управления и синхронизации

Кварцевый резонатор, подключаемый к внешним выводам микроконтроллера, управляет работой внутреннего генератора, который в свою очередь формирует сигналы синхронизации. Устройство управления CU (см рисунок 2.1) на основе сигналов синхронизации формирует машинный цикл фиксированной длительности, равной 12 периодам резонатора. Большинство команд микроконтроллера выполняется за один машинный цикл. Некоторые команды, оперирующие с 2-байтными словами или связанные с обращением к внешней памяти, выполняются за два машинных цикла. Только команды деления и умножения требуют четырех машинных циклов. На основе этих особенностей работы устройства управления производится расчёт времени исполнения прикладных программ. На схеме микроконтроллера к устройству управления примыкает регистр команд IR (см рисунок 2.1). В его функцию входит хранение кода выполняемой команды.

Входные и выходные сигналы устройства управления и синхронизации:

PSEN – разрешение программной памяти,

ALE – выходной сигнал разрешения фиксации адреса,

PROG – сигнал программирования,

EA – блокировка работы с внутренней памятью,

VPP – напряжение программирования,

RST – сигнал общего сброса,
VPD – вывод резервного питания памяти от внешнего источника,
XTAL – входы подключения кварцевого резонатора.

2.5 Способы адресации микроконтроллера Intel 8051

Способы адресации в микроконтроллере 8051 реализованы с учетом приоритета доступа к определенным массивам данных. В большинстве приложений часто обращаться приходится к очень небольшому количеству переменных, к некоторым переменным обращение происходит время от времени, а большая часть переменных выбирается достаточно редко. При проектировании 8051 компания Intel определила для доступа к данным однобайтный цикл выполнения команд, который содержит 12 тактов. Доступ к_регистрам текущего банка выполняется с помощью однобайтных команд за 12 тактов. Имеются также двухбайтные команды, выполняющие обращение к первым 256 адресам памяти за 24 такта (2 цикла), и трехбайтные команды для доступа к ячейкам памяти за пределами первых 256 байт, для выполнения которых требуется несколько циклов. Графически это показано на рисунке 2.3, который полезно помнить при разработке сложных прикладных программ на языке ассемблера. При разработке программ следует стремиться оптимизировать размещение переменных в памяти данных, чтобы обращение к ним производилось за меньшее число циклов и требовало меньшего числа байт, занимаемых в памяти программ.

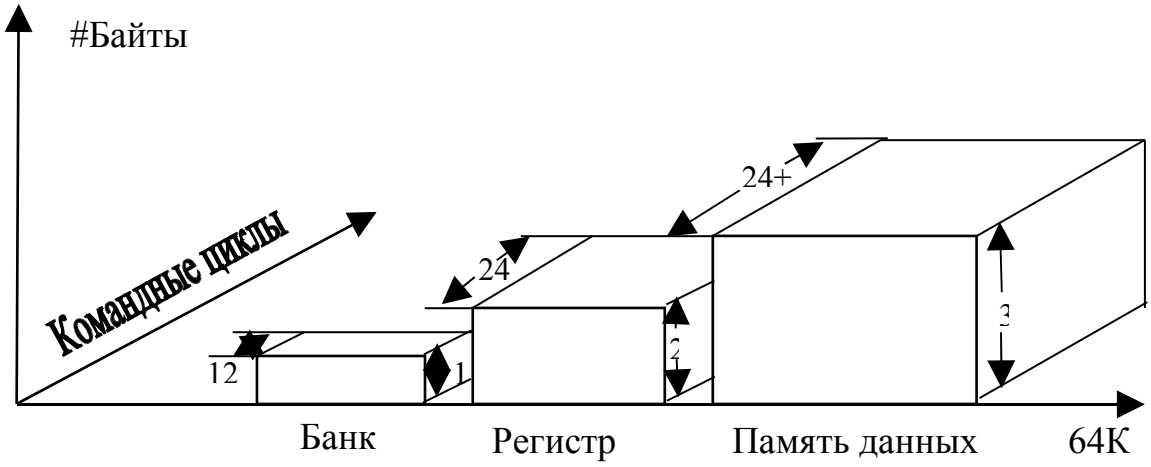

Рисунок 2.3 - Сравнение способов адресации 8051

Первый из рассматриваемых способов - **непосредственная адресация** не требует обращения к регистрам или памяти данных. При непосредственной адресации на обработку поступает операнд, который является частью команды. Значение непосредственного операнда обозначается символом #.

addA, #77 - добавить 77 к содержимому аккумулятора, где 77 - десятичное число.

Адресация к регистровым банкам обеспечивает доступ к одному из 8 байтов, размещенных в текущем банке. Это наиболее эффективный способ доступа к данным с точки зрения уменьшения объема памяти программ и сокра-

щения числа требуемых командных циклов. Большинство команд, работающих с регистровыми банками, выполняются за один цикл и занимают только один байт. Регистры банка имеют имена, начиная с R0 и по R7.

Прямая адресация отличается от регистровой тем, что можно получить доступ к любому байту в первых 256 ячейках памяти, указав 8-разрядный адрес. Ячейки с адресами от 080h no 0FFh не могут быть выбраны при помощи прямой адресации. Если написать команду: mov A, 088h, то в аккумулятор загрузится содержимое служебного регистра TCON (регистр управления таймером), а не содержимое ячейки памяти. Не следует путать прямую адресацию с непосредственной. Если в команде с непосредственной адресацией забыть поставить символ # перед операндом, то будет выполняться команда с прямой адресацией. Это является одной из распространенных синтаксических ошибок при программировании микроконтроллеров 8051. Следует быть особенно внимательным при разработке и отладке прикладных программ, чтобы быть уверенным, что реализуется требуемый способ адресации.

Косвенно-регистровая адресация осуществляется с помощью регистров R0 или R1 текущего банка. В этом случае содержимое регистра R0 или R1 используется как 8-разрядный адрес для обращения к первым 256 байтам памяти данных: Orl A, @ R0. На выполнение косвенно-регистровой адресации указывает символ @ перед именем регистра R0 или R1. Использование других регистров банка приведет к ошибке.

Регистр DPTR используется в качестве 16-разрядного индексного регистра. При этом можно указать смещение, которое добавляется к содержимому DPTR для формирования адреса операнда.

Такой способ адресации, который называется **косвенно-регистровая со смещением**, удобно использовать для доступа к отдельным элементам в структурах данных.

Когда необходимо получить доступ к данным, хранящимся в памяти программ, можно воспользоваться командой «move A, @A+PC». Эта команда адресует в памяти программ байт, который расположен со смещением относительно адреса текущей команды. Значение смещения, которое размещается в аккумуляторе A, должно быть заранее вычислено перед выполнением команды.

2.6 Организация портов ввода вывода

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 (см таблицу 2.1). 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 2.3.

Таблица 2.3 – 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.

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

2.7 Таймеры / счетчики микроконтроллеров семейства 8051

В базовых моделях семейства 8051 имеются два программируемых 16-битных таймера/счетчика (T/C0 и T/C1), которые могут быть использованы как в качестве таймеров, так и в качестве счетчиков внешних событий. В первом случае содержимое соответствующего таймера/счетчика (далее для краткости T/C) инкрементируется в каждом машинном цикле, т.е. через каждые 12 периодов колебаний кварцевого резонатора, во втором оно инкрементируется под воздействием перехода из «1» в «0» внешнего входного сигнала, подаваемого на соответствующий (T0,T1) вывод микроконтроллера 8051. Для управления режимами работы T/C и для организации их взаимодействия с системой прерываний используются два регистра специальных функций (TMOD и TCON), описание которых приведено в таблицах 2.4, 2.5.

Так как на распознавание периода требуются два машинных цикла, максимальная частота подсчета входных сигналов равна 1/24 частоты резонатора. На длительность периода входных сигналов ограничений сверху нет. Для гарантированного прочтения входной сигнал должен удерживать значение 1, как минимум, в течение одного машинного цикла микроконтроллера.

Четыре различных режима работы реконфигурируют структуру таймера для выполнения разнообразных задач. В этой структуре TLn и THn представляют 8 младших и 8 старших разрядов каждого таймера, где n – его номер (например, TL0 – младший 8-разрядный регистр таймера 0).

Источником синхросигналов, переключающих таймер, могут быть тактовые импульсы, частота которых уменьшается при помощи делителя, или внеш-

ний сигнал. Выбор источника определяется битом C/T в регистре TMOD таймера. Биты выбора режима в регистре TMOD определяют режим работы таймера.

Таблица 2.4 - Регистр режима работы таймера/счетчика TMOD

<table>
<tr><th>Символ</th><th>Позиция</th><th colspan="3">Имя и назначение</th></tr>
<tr><td>GATE</td><td>TMOD.7 для T/C1; TMOD.3 для T/CO</td><td colspan="3">Управление блокировкой. Если бит установлен, то таймер /счетчик "x" разрешен до тех пор, пока на входе "INTx" высокий уровень и бит управления "TRx" установлен. Если бит сброшен, то T/C разрешается, как только бит управления "TRx" устанавливается</td></tr>
<tr><td>C/T</td><td>TMOD.6 для T/C1; TMOD.2 для T/CO</td><td colspan="3">Бит выбора режима таймера или счетчика событий. Если бит сброшен, то работает таймер от внутреннего источника сигналов синхронизации. Если; установлен, то работает счетчик от внешних сигналов на входе "Tx"</td></tr>
<tr><td rowspan="5">M1
M0</td><td rowspan="5">TMOD.5 для T/C1; TMOD.1 для T/CO
и
TMOD.4 для T/C1 и TMOD.0 для T/CO</td><td colspan="3">Режим работы</td></tr>
<tr><td>M1</td><td>M0</td><td></td></tr>
<tr><td>0</td><td>0</td><td>Таймер BE48. "TLx" работает как 5-битный предделитель</td></tr>
<tr><td>0</td><td>1</td><td>16 битный таймер/счетчик. "THx" и "TLx" включен последовательно</td></tr>
<tr><td>1</td><td>0</td><td>8-битный авто перезагружаемый таймер/счетчик. "THx" хранит значение, которое должно быть перезагружено в "TLx" каждый раз по переполнению</td></tr>
<tr><td>1</td><td>1</td><td>Таймер/счетчик 1 останавливается. Таймер/счетчик 0: TLO работает как 8-битный таймер/счетчик, и его режим определяется управляющими битами таймера 0. TH0 работает только как 8 битный таймер, и его режим определяется управляющими битами таймера 1</td></tr>
</table>

Таблица 2.5 - Регистр управления/статуса таймера TCON

Символ	Позиция	Имя и назначение
TF1	TCON.7	**Флаг переполнения таймера 1.** Устанавливается аппаратно при переполнении таймера/счетчика. Сбрасывается при обслуживании прерывания аппаратно
TR1	TCON.6	**Бит управления таймера 1.** Устанавливается, / сбрасывается программой для пуска/останова
TF0	TCON.5	**Флаг переполнения таймера 0.** Устанавливается аппаратурно. Сбрасывается при обслуживании прерывания
TR0	TCON.4	**Бит управления таймера 0.** Устанавливается / сбрасывается программой для пуска/останова таймера/счетчика
IE1	TCON.3	**Флаг фронта прерывания 1.** Устанавливается аппаратно, когда детектируется срез внешнего сигнала INT1. Сбрасывается при обслуживании прерывания
IT1	TCON.2	**Бит управления типом прерывания 1.** Устанавливается / сбрасывается программно для спецификации запроса INT1 (срез/низкий уровень)
IE0	TCON.1	**Флаг фронта прерывания 0.** Устанавливается по срезу сигнала INT0. Сбрасывается при обслуживании прерывания
IT1	TCON .0	**Бит управления типом прерывания 0.** Устанавливается / сбрасывается программно для спецификации запроса INT0 (срез/низкий

		уровень)

В режиме 0 таймер конфигурируется как 13-разрядный счетчик, переполнение которого фиксируется по возникновению переноса из разряда 4 регистра TLn. В режиме 1 таймер работает как 16-разрядный счетчик.

Режим 2 обычно используется для задания скорости обмена встроенного последовательного порта. Для счета используется только счетчик TLn. Когда содержимое этого счетчика превышает OFFh, то в него загружается значение, которое содержится в регистре THп, и счет возобновляется. Последовательный порт может использовать сигнал переполнения, чтобы задать скорость обмена.

В режиме 3 регистры TL0 и THO работают как два независимых таймера. Этот режим обеспечивается только в таймером 0, так как для его реализации требуются как внешний так и внутренний синхросигналы, а таймер 1 не может переключаться внешним сигналом. Таймер THO переключается внешними сигналами, а TL0 — внутренними тактовыми импульсами. Режим 3 может применяться для реализации тахометра, когда счетчик THO определяет количество событий (например, оборотов вала), а TL0 — промежуток времени, в течение которого они произошли.

2.8 Система прерываний микроконтроллера Intel 8051

В блоке регистров специальных функций есть два регистра, предназначенных для управления режимом прерываний IE и уровнями приоритета IP. Возможность программной установки/сброса любого управляющего бита в этих двух регистрах делает систему прерываний 8051 исключительно гибкой.

В таблицах 2.6 и 2.7 описаны назначения битов регистров IE и IP.

Таблица 2.6 – 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 2.7 – 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.

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.

3 Характеристика современных средств разработки микропроцессорных систем

3.1 Основные инструментальные средства разработки приложений на базе микроконтроллеров

Наличие в микропроцессорной системе, как аппаратных, так и программных средств обуславливает ряд специфических особенностей, присущих процессу ее создания. Он существенно отличается от проектирования традиционных электронных устройств, не предполагающих программное обеспечение. В отличие от традиционного подхода, когда все функции, возлагаемые на устройство, достигаются чисто аппаратными средствами и другой альтернативы просто не существует, при аппаратно-программной реализации выполняемые функции оптимально располагаются между программными и аппаратными средствами микропроцессорной системы.

Идея единства программного и аппаратного обеспечения систем на базе микроконтроллеров является очень важной. Объединение инструментальных средств разработки программного обеспечения с инструментальными средствами разработки аппаратного обеспечения может стать важным преимуществом при разработке устройства. Существуют пять различных инструментов, которые используются для разработки приложений на базе микроконтроллеров, и объединение их функций может существенно облегчить процесс проектирования:

- редактор исходных текстов;
- компилятор/ассемблер;
- программный симулятор;
- аппаратный эмулятор;
- программатор.

Хотя не все из этих инструментов являются необходимыми, и каждый из них может исполняться в отдельности, но их совместное использование упрощает разработку и отладку приложения.

Редактор используется для создания исходного кода программы. Существует множество самых разнообразных редакторов от простых, которые копируют код, вводимый с клавиатуры, в файл, до специализированных редакторов, реакция которых на нажатие определенных клавиш может программироваться пользователем. Такая реакция редактора избавляет разработчика от необходимости заботится о правильном синтаксисе оператора.

Компилятор/ассемблер используется для преобразования исходного текста в машинные коды микроконтроллера, т.е. в формат, который может быть загружен память программ.

Симуляторы — это программы, которые выполняют откомпилированный программный код в инструментальном компьютере. Это позволяет осуществлять наблюдение за программой и реакцией микроконтроллера на различные события. Симулятор может быть неоценимым инструментом в процессе разработки программного обеспечения, позволяя исследовать различные ситуации, которые трудно воспроизвести на реальной аппаратуре.

Важное преимущество симуляторов — возможность многократного воспроизведения рабочих ситуаций. Если надо понять, почему участок программы работает некорректно, можно повторять этот участок снова и снова до тех пор, пока ошибка не будет обнаружена. Воспроизводимость может быть расширена путем использования специальных файлов входных воздействий. Эти файлы служат для того, чтобы задать симулятору различные комбинации входных потоков данных и формы сигналов. Чтобы имитировать внешние условия и ситуации, обычно используется специальный файл входных воздействий. Этот файл задает последовательность входных сигналов, поступающих на моделируемое устройство. Разработка такого файла может потребовать много времени и больших усилий. Но для понимания того, как работают микроконтроллер и программа в определенных ситуациях, использование симулятора и файла входных воздействий является наилучшим методом. В большинстве случаев следует использовать симуляцию перед сборкой и включением реальной схемы. Если устройство не работает ожидаемым образом, то следует изменить файл входных воздействий и попытаться понять, в чем состоит проблема, используя для этого симулятор, который позволяет наблюдать за процессом выполнения программы в отличие от реальной аппаратуры, где можно увидеть только конечные результаты.

Эмуляторы. Наиболее сложным и дорогим инструментом для отладки приложения и электрических интерфейсов является эмулятор. Эмулятор — это устройство, которое заменяет микроконтроллер в схеме и выполняет программу под Вашим управлением. Эмулятор является превосходным инструментом для разработки большинства приложений, хотя имеются некоторые разногласия по вопросу их использования в процессе разработки.

Обычно эмулятор содержит специальный эмуляторный кристалл, который подсоединяется в ведущему компьютеру или рабочей станции. Эмуляторный кристалл - это обычный микроконтроллер, помещенный нестандартный корпус с дополнительными выводами, которые подключаются к шине памяти программ и управляющим сигналам процессора. Эти дополнительные выводы позволяют соединять микроконтроллер с внешней памятью программ, которая подключена к ведущему компьютеру. Такой интерфейс позволяет легко загружать тестовые программы в эмуляторный кристалл. Выводы эмуляторного кристалла соединяются с разъемом (эмуляторная вилка), который подключается к отлаживаемому устройству, заменяя в нем микроконтроллер.

Взаимодействие с эмуляторами очень похоже на работу с симулятором. Разница состоит в том, что эмулятор не воспринимает файл входных воздействий, так как его выводы подключены к реальному устройству. Эмулятор предназначен для отладки приложений на реальном оборудовании.

Программатор. Последний инструмент разработчика - это программатор памяти программ микроконтроллера. Хотя некоторые производители микроконтроллеров предпочитают выпускать их с масочнопрограммируемой памятью программ, они обычно выпускают также аналогичные версии микроконтроллеров с E(E)PROM памятью для разработки приложений. Это значит, что существует возможность непосредственного программирования микроконтроллера при разработке приложения.

Для некоторых микроконтроллеров требуется специальный программатор, но чаще всего используются возможности внутрисистемного программирования ISP. В таком случае программатор является частью проектируемого устройства. Некоторые программаторы реализуют функции схемного эмулятора. При этом установленный в программаторе микроконтроллер подключается к отлаживаемой системе и управляет ее работой аналогично тому, как это выполняется в эмуляторе.

3.2 Краткое описание программного симулятора UMPS

UMPS (Universal Microprocessor Program Simulato) – интегрированная среда проектирования приложений на базе микроконтроллеров. UMPS – универсальный симулятор микроконтроллеров. Работает под управлением операционных систем Microsoft Windows 3.11, 95/98, 2000, NT.

Программа-симулятор предназначена для моделирования микропроцессорных систем, в основе которых лежит микроконтроллер, а также для изучения особенностей работы микроконтроллеров, приобретения навыков в написании и отладки программ. Симулятор позволяет выполнять следующие действия:

- создавать или открывать файлы с исходным кодом на ассемблере микроконтроллера;
- сохранить исходный код программы под старым или новым именем;
- выполнить компиляцию программы;
- запустить программу в непрерывном режиме;
- запустить программу в пошаговом режиме;
- запустить программу в режиме трассировки;
- прервать выполнение программы в любой момент времени;
- сбросить содержимое регистров, перевести ресурсы в выключенное состояние и перейти к выполнению программы с начального адреса;
- отобразить код микроконтроллера;
- отобразить память программ микроконтроллера;
- отобразить ресурсы;
- добавить ресурсы;
- отобразить регистры;
- добавить регистры;

- загрузить модель микроконтроллера;
- подключать дополнительные ресурсы отладки: светодиоды, семисегментные индикаторы, счетчики и т.д.

Имеется возможность создания собственных ресурсов на языке Си или Pascal. Кроме наличия редактора и ассемблера UMPS обеспечивает возможность моделирования работы микроконтроллера совместно с внешними устройствами.

3.3 Порядок работы симулятора UMPS

Начать работу в симуляторе UMPS можно с загрузки модели микроконтроллера, а можно сразу приступить к написанию управляющей программы. Рабочее окно симулятора в начальный момент работы имеет вид, показанный на рисунке 3.1.

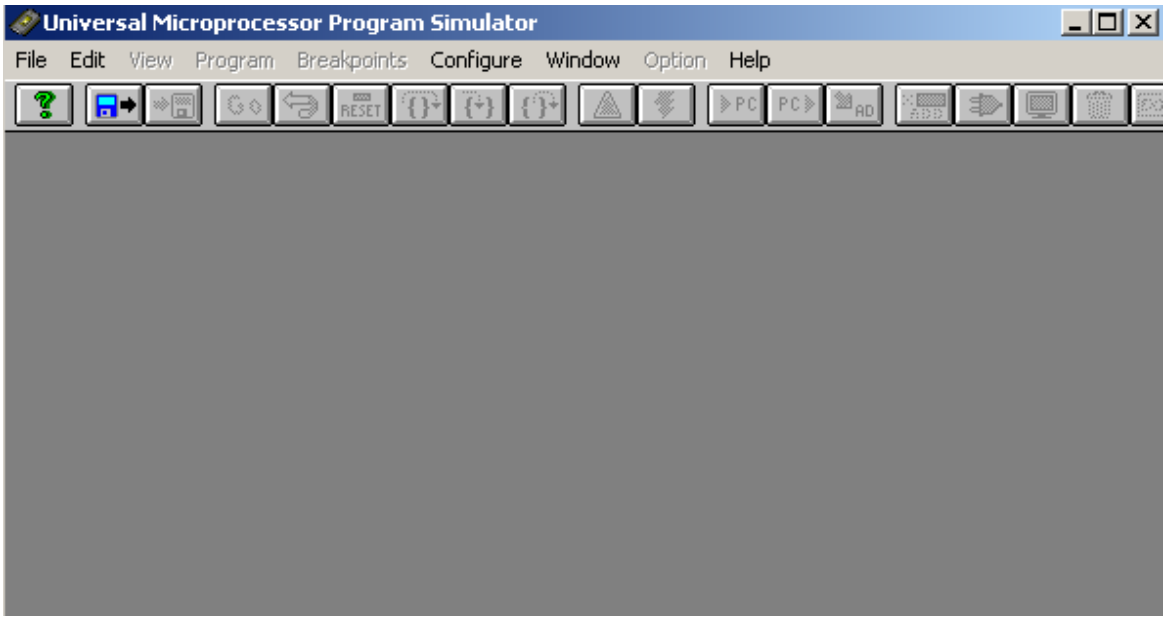

Рисунок 3.1 - Рабочее окно симулятора UMPS

Для загрузки типа используемого микроконтроллера необходимо выбрать пункт меню «Configure», а затем «Load CPU…». Из предложенного списка микроконтроллеров выберите DS80C320 (это микроконтроллер с ядром 8051).

Для написания программы используют редактор, необходимо в меню «File» выбрать «New». По мере ввода программного кода комментарии отображаются курсивом синего цвета. Директивы ассемблера помечаются красным, а команды выделяются жирным шрифтом. Пример программы на языке ассемблера микроконтроллера 8051 показан на рисунке 3.2.

Следующим шагом после написания программы является компиляция. Откомпилировать код можно путем выбора команды «Compile» в меню «Program» или нажатием на кнопку с изображением желтого знака, напоминающего молнию на панели инструментов (рисунок 3.3). Вам будет задан вопрос об име-

ни исходного файла и предложено сохранить текст перед компиляцией. После окончания компиляции появится окно с результатами. Ассемблер UMPS прекращает компиляцию при обнаружении первой ошибки. Строка с ошибкой будет выделена красным цветом.

```
14
15   org 0
16    ajmp    Mainline
17
18   org 0Bh                        ;  Timer0 Interrupt
19    cpl     P1.7                  ;  Flip Bit 7
20    reti
21
22   org 020h                       ;  Program Mainline
23  Mainline:
24
25    clr     CKCON                 ;  Use Internal /12 Clock for Timer0
26    mov     TMOD,#%00000001       ;  Timer0 - Uses Internal Clock
27                                  ;         - Run in Mode 1
28    mov     TCON,#%00010000       ;  Start Timer0 running
29
30    mov     IE,#%10000010         ;  Enable the Timer 0 Interrupt
31
32    mov     P1,#081h              ;  Make LED Bits High
33    mov     P2,#1                 ;  Make All P2 Bits Low except for Button
```

Рисунок 3.2 - Пример программы в редакторе UMPS

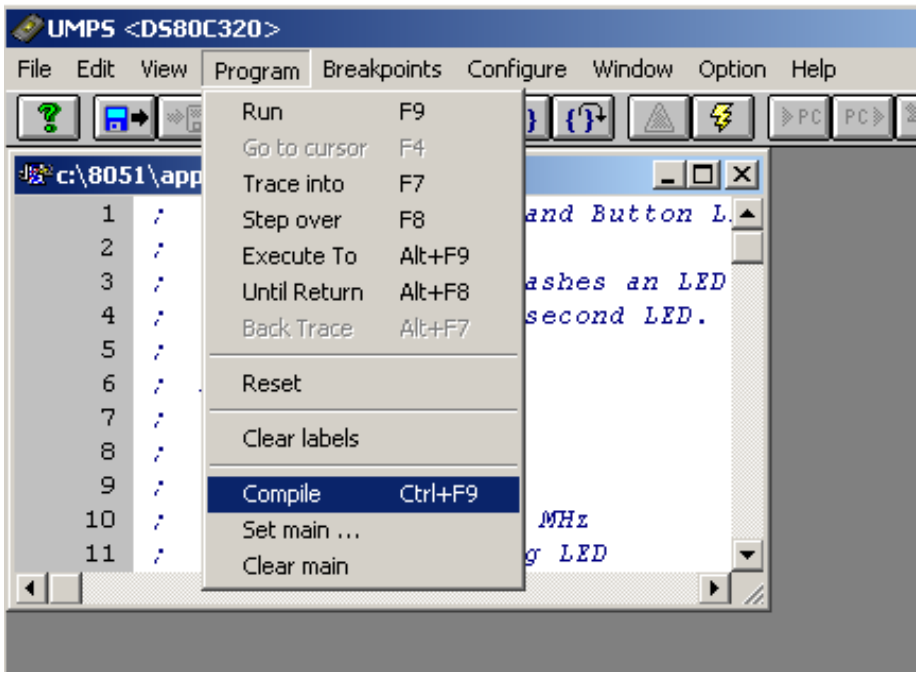

Рисунок 3.3 – Компиляция программы

После того, как программа успешно откомпилирована, можно приступать к симуляции. UMPS позволяет осуществит подключение виртуальных ресурсов. Для выбора и подключения ресурсов необходимо выбрать позицию «Re-

sources» в меню «Configure». На экране появится окно. На панели инструментов нужно выбрать «ADD» и поместить в окно выбранные ресурсы (светодиоды, кнопки и т.д.). Окно подключения ресурсов показано на рисунке 3.4.

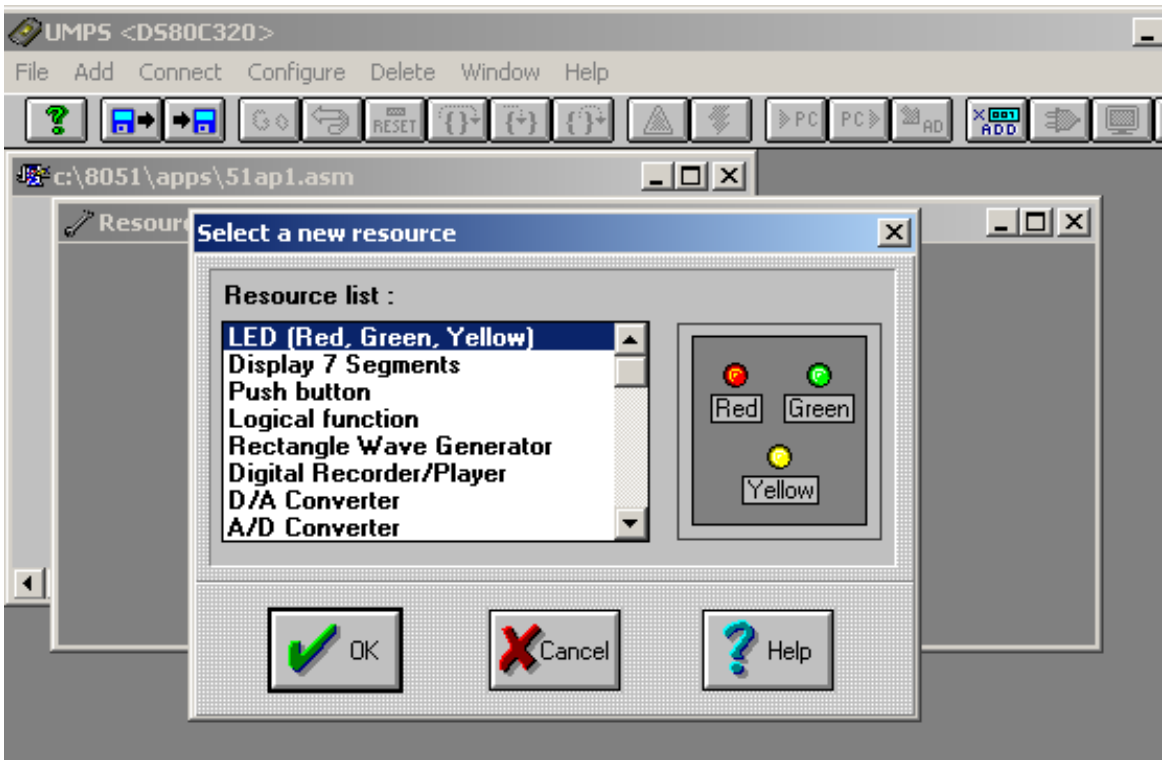

Рисунок 3.4 – Окно подключения ресурсов

Далее окно можно уменьшить и поместить в левый нижний угол экрана. Чтобы подключить введенные элементы, например светодиод, выберите светодиод в окне ресурсов и нажмите «Connect». Затем выберите «Anode» и «Always 1» и нажмите «Connect», затем нужно подключить катод, чаще всего его подключают к одному из разрядов порта. Для зажигания светодиода необходимо создать разность потенциалов между его анодом и катодом, т.е. если на аноде всегда «1», то светодиод зажжется при подаче на катод «0». Симулятор предоставляет возможность выбрать цвет светодиода, обозначить каждый светодиод по своему усмотрению.

Для подключения других ресурсов необходимо изучить информацию, представленную в самой программе UMPS, а также в книге М. Предко «Руководство по микроконтроллерам», том 1. Для каждого элемента в симуляторе имеется информация по его структуре, электрическая схема и т.д. Для ознакомления необходимо дважды щелкнуть мышью по установленному в рабочее окно элементу. После подключения всех необходимых ресурсов следует выйти из окна ресурсов, выбрав на панели инструментов желтую кнопку «Exit».

После подключения внешних элементов можно вывести на экран содержимое всех используемых регистров микроконтроллера, чтобы наблюдать за выполнением программы. Для этого следует выбрать пункт «CPU Registers» из меню «Configure». Для добавления аккумулятора нажмите «ADD» и выберите

«ACC». Эти действия необходимо повторить для каждого требуемого регистра. В результате этих действий на экране симулятора будут отображены три окна: окно с текстом программы, окно с ресурсами, окно с регистрами микроконтроллера (рисунок 3.5).

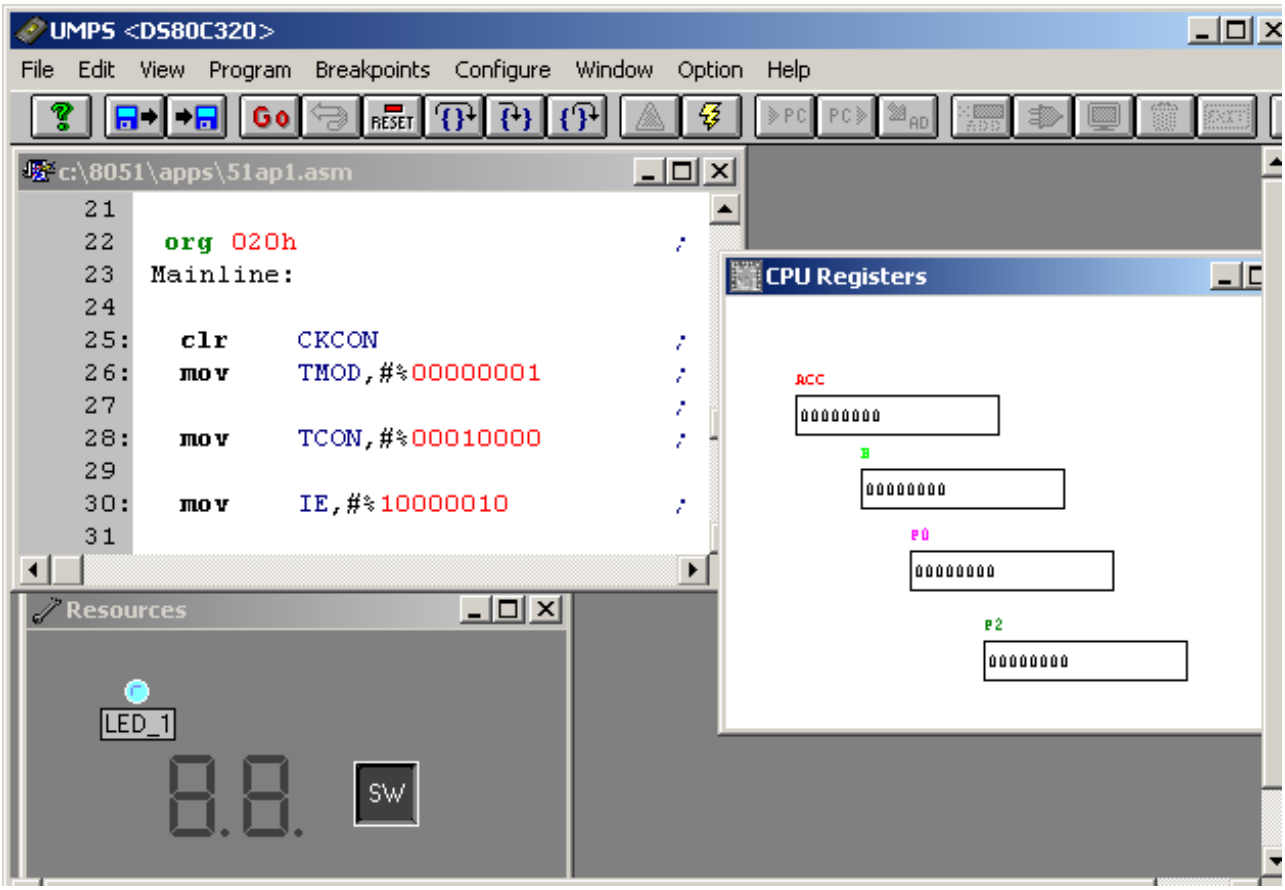

Рисунок 3.5 – Экран симулятора с отображением трех окон

Теперь можно выполнить программу, выбрав в пункте меню «Program» необходимый режим: «Run» - прогон или «Step over» - пошаговый. Пошаговый режим полезен в том случае, когда необходимо отследить изменения содержимого выведенных на экран регистров процессора, тем самым определить правильность работы программы. Когда программа полностью отлажена, в режиме «Run» будут видны конечные результаты, отображающиеся с помощью ресурсов: зажигание светодиодов, отображение информации на семисегментных индикаторах и т.д. Имеется возможность установить точки останова в определенных адресах программы, что позволит реализовать более сложное взаимодействие программной и аппаратной частей разработанной микропроцессорной системы.

Одним из распространенных подключаемых элементов является семисегментный индикатор, который применяют для отображения временных интервалов в различных микроконтроллерных приложениях. Ниже приводится описание семисегментного индикатора, реализованного в интегрированной среде UMPS.

3.4 Семисегментный светодиодный индикатор

Вероятно, самый простой способ вывода числовых десятичных и шестнадцатеричных данных — это использование семисегментного светодиодного индикатора. Такие индикаторы были очень популярны в 70-х годах, но впоследствии их место заняли жидкокристаллические индикаторы. Но светодиодные индикаторы до сих пор являются полезными приборами, которые могут быть включены в схему без больших усилий для создания программного обеспечения. Включая определенные светодиоды (зажигая сегменты), можно выводить десятичные числа (рисунок 3.6).

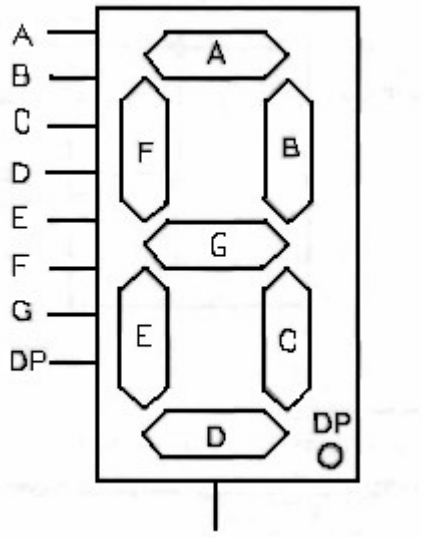

Рисунок 3.6 - Семисегментный светодиодный индикатор

Каждый светодиод в индикаторе имеет свой буквенный идентификатор (A, B, C, D, E, F, G), и один из выводов светодиода подключен к соответствующему внешнему выводу. Вторые выводы всех светодиодов соединены вместе и подключены к общему выводу. Этот общий вывод определяет тип индикатора: с общим катодом или с общим анодом. Подключение индикатора к микроконтроллеру осуществляется весьма просто: обычно индикатор подключают как семь или восемь (если используется десятичная точка) независимых светодиодов.

Наиболее важной частью работы при подключении к микроконтроллеру нескольких семисегментных индикаторов является назначение линий ввода-вывода для каждого светодиода. Решение этой задачи в начале выполнения проекта упростит монтаж разводки и отладку устройства в дальнейшем. Типичный способ подключения нескольких индикаторов состоит в том, чтобы включить их параллельно и затем управлять протеканием тока через общие выводы отдельных индикаторов. Так как величина этого тока обычно превышает допустимое значение выходного тока микроконтроллера, то для управления током включаются дополнительные транзисторы, которые выбирают, какой из индикаторов будет находиться в активном состоянии.

На рисунке 3.7 показано подключение к микроконтроллеру четырех семисегментных индикаторов.

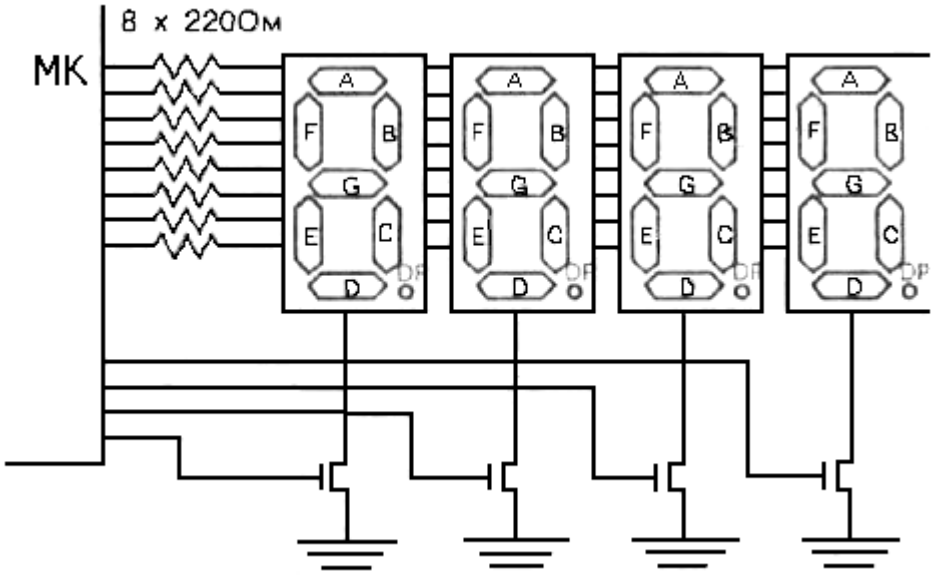

Рисунок 3.7 –Подключение четырех семисегментных индикаторов

В этой схеме микроконтроллер выдает данные для индикации, последовательно переходя от одного индикатора к другому. Каждая цифра будет высвечиваться в течение очень короткого интервала времени. Это обычно выполняется с помощью подпрограммы обслуживания прерываний таймера. Эта подпрограмма будет циклически производить вывод цифры на каждый индикатор, разрешая протекание тока через транзистор, подключенный к его общему выводу. Чтобы избежать мерцания изображения, подпрограмма должна выполняться со скоростью, обеспечивающей включение индикатора (свечение каждой цифры) по крайней мере 50 раз в секунду. Чем больше цифр, тем чаще должны следовать прерывания от таймера. Например, при восьми индикаторах цифры должны выводиться со скоростью 400 раз в секунду, то есть в два раза быстрее, чем для четырех индикаторов.

В некоторых ситуациях может оказаться, что выделение каждому светодиоду индикатора отдельного вывода микроконтроллера слишком расточительно. Можно использовать демультиплексор с высоким выходным током, например, ТТЛ-микросхему типа 74S138, вместо дискретных транзисторов. Когда на выбранном выходе демультиплексора устанавливается низкий уровень, то он пропускает ток подключенного к нему индикатора, обеспечивая вывод цифры. При этом разводка монтажных соединений в устройстве оказывается более простой. Следует обратить внимание на то, чтобы используемый де-мультиплексор был способен пропускать максимальный ток 140мА, который протекает через общий катод индикатора. Наряду с семисегментными индикаторами существуют 14- и 16-сегментные индикаторы, которые позволяют выводить символы алфавита («A»-«Z» и «0»-«9»).

4 Лабораторный практикум

Использование симулятора UMPS при выполнении лабораторных работ является эффективным способом получения умений и навыков моделирования микропроцессорных систем на базе микроконтроллеров. Для решения поставленных задач необходимо освоить архитектуру микроконтроллеров с ядром 8051, изучить систему команд, способы адресации, научиться работать в интегрированной среде проектирования UMPS.

Первый шаг цикла моделирования микропроцессорной системы (МПС) включает в себя определение набора требований пользователя и создания вытекающей из них **функциональной спецификации**, а также формулирование системных требований к МПС. В качестве требований пользователя выступает задание на проектирование МПС. Функциональная спецификация МПС определяет, какие функции должны выполняться для удовлетворения требований пользователя и обеспечения интерфейса (связи) между системой и ее внешним окружением (исполнительными устройствами, датчиками и т.д.). Последнее определяет наличие и количество индикационных элементов, клавиатуры, входов и выходов МПС. На этапе формулирования системных требований детализируется функциональная спецификация с точки зрения выполнения системных функций (системная функция ввода-вывода дискретной, аналоговой информации, обслуживание клавиатуры и индикации и др.).

Следующим этапом является **разбиение МПС на аппаратную и программную части**. Необходимо, во-первых, определить аппаратную и программную конфигурации; во-вторых – какие из функций функциональной спецификации будут выполняться аппаратной частью МПС, а какие - программной. На данном этапе, называемым системно-алгоритмическим проектированием МПС, помимо разбиения МПС на программную и аппаратную части, разрабатывается также ее общая структура и алгоритмы функционирования. Последнее выполняется с учетом разделения аппаратно-реализуемых и программно-реализуемых функций.

После принятия компромиссного решения о разделении на аппаратную и программную части дальнейшая разработка МПС проводится параллельно для аппаратных и программных средств. При этом необходимо тщательно учитывать особенности, достоинства и недостатка реализации функций каждой частью МПС.

Для выполнения работы необходимо разработать структурную схему микропроцессорной системы с учетом всех требований, предъявляемых в задании, разработать алгоритм программы. Затем написать программу на языке ассемблера 8051, оптимизировать ее, провести компиляцию. Выбрать виртуальные ресурсы и осуществить их подключение. Далее необходимо провести отладку программы в пошаговом режиме, отслеживая правильность работы подключенных ресурсов. После выполнения работы следует оформить отчет, содержащий следующие пункты:

1) цель работы;
2) задание;
3) функциональная спецификация;
4) разбиение МПС на аппаратную и программную части;
5) структурная схема;
6) таблица распределения подключаемых ресурсов к разрядам портов;
7) схема подключения ресурсов;
8) схема алгоритма программы;
9) текст программы;
10) экранные формы, отображающие выполнение микропроцессорной системой заданных функций;
11)список использованной литературы.

4.1 Лабораторная работа 1. Подключение светодиодов и кнопки

Цель работы: Изучение архитектуры микроконтроллера Intel 8051. Освоение системы команд. Получение навыков работы с UMPS.

4.1.1 Задание

Разработать схему подключения светодиодов и кнопки к микроконтроллеру DS80C320. Написать управляющую программу, обеспечивающую выполнение следующих задач.

Задача А

1 Загрузить в регистры R0-R3 банка 1 операнды: 23; 142; 231; 203, представив их в разных кодах (B; D; H).
2 Сложить попарно числа в регистрах R0+R1; R2+R3. Результаты сложения сохранить в портах P0 и P1 соответственно.
3 К портам P0, P1 подключить по восемь светодиодов. Зажигая их отобразить полученные результаты сложения в B-коде.
4 Реализовать гашение светодиодов при нажатии кнопки.
5 Вывести на экран содержимое всех используемых регистров.

Задача В

1 Ввести четыре 8-разрядных числа (A, B, X, Y);
2 Перемножить числа попарно A*B и X*Y;
3 Сравнить результаты операций;
4 При нажатии кнопки вывести через порты результаты сравнения на три светодиода. Использовать светодиоды разного цвета и обозначить их «>», «<», «=»;

5 Вывести на экран содержимое всех используемых регистров.

4.1.2 Контрольные вопросы

1 Что входит в число пяти основных инструментов разработчика программного и аппаратного обеспечения систем на базе микроконтроллеров? Пояснить назначение каждого из них.
2 Что такое симулятор? Какие возможности он предоставляет? Каково важное преимущество симуляторов? В каких случаях симулятор не применим?
3 Что такое эмулятор? В чем состоит его отличие от симулятора? Что такое гибрид эмулятора и симулятора? Каков основной недостаток применения эмулятора?
4 В чем заключается стратегия отладки прикладных программ для микроконтроллеров? Что такое «восходящее» проектирование?
5 Что такое программаторы? Какие возможности они предоставляют? Что такое программирование в составе системы?
6 Какова базовая архитектура микроконтроллера 8051? Назвать типовые характеристики микроконтроллера.
7 Показать распределение адресов памяти данных микроконтроллера 8051. Пояснить назначение и способы обращения к каждой области памяти данных микроконтроллера.
8 Назвать служебные регистры микроконтроллера 8051? Каково назначение регистров процессора?

4.2 Лабораторная работа 2. Подключение семисегментных индикаторов

Цель работы: Изучение системы ввода/вывода микроконтроллера 8051, программная реализация таблиц.

4.2.1 Задание

Разработать схему подключения кнопок клавиатуры и семисегментных индикаторов к микроконтроллеру DS80C320. Написать управляющую программу, обеспечивающую выполнение следующих задач.

Задача А

1 Ввести 8-разрядное число в Н-коде в порт микроконтроллера. Предусмотреть возможность введения любых 8-разрядных чисел.
2 Преобразовать введённое число в семисегментный код: каждая цифра числа в его десятичном формате преобразуется в соответствующий

семисегментный код. Например, цифре 1 соответствует код: 0110000 (см рисунок 4.1)

A	B	C	D	E	F	G
0	1	1	0	0	0	0

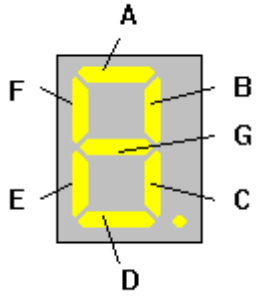

Рисунок 4.1 – Семисегментный индикатор

Максимальное 8-разрядное число в Н-коде равно FF, его десятичный формат – 256. Значит, для вывода десятичного эквивалента 8-разрядного числа потребуется 3 семисегментных индикатора.

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

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

5 Предусмотреть минимальный объем программы.

Задача В

1 Ввести в регистры два 8-разрядных числа X и Y в Н-коде;

2 Подключить к портам микроконтроллера кнопки с обозначением цифр от 0 до 9, образуя, таким образом, клавиатуру;

3 При нажатии кнопок полученной клавиатуры обеспечить отображение задаваемых 8-разрядных чисел X и Y на четырех семисегментных индикаторах в десятичном коде (использовать фрагмент программы преобразования чисел из Н-кода в D- код - **задание А**);

4 Осуществить отображение результатов вычислений над 8-разрядными числами X и Y, вводимыми с клавиатуры. Вычисления провести в соответствии с вариантами формул, представленных в таблице 4.1.

Таблица 4.1 – Варианты формул для вычислений

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

4.2.2 Контрольные вопросы

1 Каковы достоинства и недостатки Принстонской и Гарвардской архитектур микроконтроллеров?

2 Описать структуру регистра слова состояния процессора – PSW микроконтроллера 8051, объяснить назначение каждого бита.

3 Пояснить - какова организация стека микроконтроллера 8051. Какую область памяти может адресовать указатель стека микроконтроллера 8051? Какое начальное значение регистра SP удобно установить и почему?

4 Что представляет собой регистр DPTR? Для чего он предназначен? С помощью какого способа адресации можно обратиться к этому регистру?

5 Сколько тактов содержит однобайтный цикл выполнения команд? Почему необходимо стремиться оптимизировать размещение переменных в памяти данных при разработке программ?

6 Какие вы знаете способы адресации микроконтроллера 8051. Показать примеры команд, соответствующие каждому способу. Произведите сравнение способов адресации 8051.

7 Перечислить основные группы команд микроконтроллера 8051.

8 Как организовать программным путем работу с таблицами? Какие регистры используются при этом?

4.3Лабораторная работа 3. Разработка охранной сигнализации автомобиля

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

4.3.1 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 капо-

том, 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.

4.3.2 Контрольные вопросы

1 Входят ли порты ввода/ вывода в состав ядра микроконтроллера? Перечислить модули, входящие в состав ядра.
2 Какие порты ввода/вывода и в каком количестве входят в состав базовой модели микроконтроллера 8051? Объясните работу типовой схемы подключения внешнего вывода порта, схемы подключения вывода с открытым коллектором.
3 Какие функции выполняют параллельные порты микроконтроллера 8051? Рассмотреть каждый порт. Какие вы знаете альтернативные функции параллельных портов?
4 Как программным путем реализовать обращение к портам? Есть ли у них адреса? С помощью какого способа адресации можно обратиться к портам микроконтроллера?
5 Где расположена область памяти для размещения программ обработки прерываний? Сколько байт памяти выделено для каждого вида прерываний в контроллере 8051?
6 Какие регистры специальных функций микроконтроллера 8051 предназначены для управления режимом прерываний? Опишите их.
7 Какие условия блокируют аппаратный вызов прерывания?
8 Каков порядок работы микроконтроллера при обслуживании прерывания?

4.4Лабораторная работа 4. Измерение временного интервала

Цель работы: Изучение режимов работы таймеров.

4.4.1 Задание

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

1 Измерить длительность временного интервала с точностью до 0,01 секунды. Использовать таймер, входящий в состав микроконтроллера;
2 Начало и окончание интервала задать нажатием кнопки;
3 Отобразить в десятичной системе счисления измеренную длительность интервала на четырехпозиционном индикаторе (два знака после запятой);
4 Сравнить измеренное время с эталоном, имеющим два знака после запятой. Обеспечить возможность изменять значение эталона перед запуском программы;
5 Реализовать индикацию результата сравнения с помощью светодиодов с обозначением «>»; «<»; «=».

4.4.2 Контрольные вопросы

1 Входят ли таймеры в состав ядра микроконтроллера? Перечислить модули, входящие в состав ядра.
2 Сколько таймеров входит в состав микроконтроллера 8051? Назовите и опишите их.
3 Опишите режимы работы таймеров.
4 Для чего предназначен сторожевой таймер? В чем заключается принцип его работы?
5 Опишите структуру таймера в микроконтроллере. Как реализовать задержку с помощью таймера.
6 Какие вы знаете команды передачи управления микроконтроллера 8051?
7 В каких командах и для чего используются регистры DPTR и PC?
8 Назовите команды побитовых операций, входящих в систему команд микроконтроллера 8051.

4.3 Лабораторная работа 5. Разработка микропроцессорной системы управления светофором

Цель работы: Приобретение навыков разработки микропроцессорной системы с подключением разнообразных ресурсов

4.5.1 Задание

Разработать контроллер светофора на перекрестке главной улицы и второстепенной в соответствии с рисунком 4.2. Имеются четыре кнопки К для пешеходов на второстепенной улице.

1 Реализовать последовательное зажигание лампочек светофора главной улицы, причем желтый свет включается на 10 секунд; зеленый и красный - на 30 секунд;

2 Светофор второстепенной улицы работает синхронно со светофором главной улицы: когда на главной улице горит красный, на второстепенной – зеленый;

3 При любом свете главного светофора обеспечить зажигание зеленого света для второстепенной улицы при наличии сигнала с кнопки пешехода через 10 секунд после нажатия кнопки;

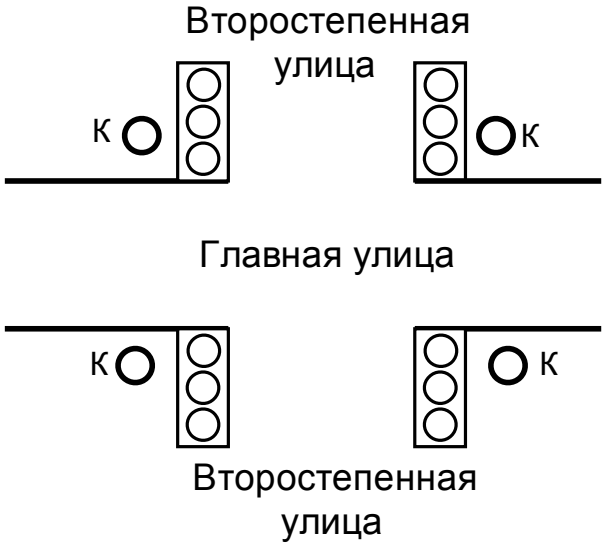

Рисунок 4.2 Схема перекрестка

4 Отображать на индикаторе время, которое осталось до окончания горения зеленого света на главной дороге.

4.5.2 Контрольные вопросы

1 Назовите отличительные признаки современных 8-разрядных микроконтроллеров. В чем заключается модульная организация микроконтроллеров.

2 Какие вы знаете типы памяти микроконтроллеров? Как классифицируются микроконтроллеры по типу организации памяти?

3 Как осуществляется тактирование и синхронизация микроконтроллеров?

4 Дайте описание схемы питания микроконтроллера. Что такое потребляемая мощность микроконтроллера?

5 Назовите средства разработки микроконтроллеров.

6 Какие вы знаете языки программирования микроконтроллеров?

7 Какие группы входят в систему команд микроконтроллеров?

8 Назовите отличия системы команд CISC-процессоров от системы команд RISC-процессоров.

Список использованных источников

1. **Бродин В.Б.** Системы на микроконтроллерах и БИС программируемой логики / В.Б. Бродин, А.В. Калинин– М.: ЭКОМ, 2002. – 431 с.
2. **Бурькова Е.В.** Освоение микропроцессорной техники в формировании информационной компетентности студентов университета: Учебное пособие / Е.В. Бурькова – Челябинск: Изд-во Южно-Уральского отделения РАО, 2005. – 209 с.
3. **Васильев В.Н.** Электронные промышленные устройства / В.Н. Васильев, Ю.М. Гусев – М.: Высшая школа, 1988. – 303с.
4. **Каган Б.Н.** Основы проектирования микропроцессорных устройств автоматики / Б.Н. Каган, В.В. Сташин – М.: Энергоатомиздат, 1987. – 304с.
5. **Корнеев В.В.** Современные микропроцессоры / В.В.Корнеев, А.В. Киселёв– М.: Нолидж, 2000. – 320 с.
6. **Новиков Ю.В.** Основы микропроцессорной техники / Ю.В. Новиков, П.К. Скоробогатов - М.: ИНТУИТ.РУ. «Интернет-Университет Информационных технологий», 2003.-440с.
7. **Новиков Ю.В.** Основы цифровой схемотехники. Базовые элементы и схемы. Методы проектирования / Ю.В. Новиков –М.: Мир 2001.–379 с.
8. **Предко М.** Руководство по микроконтроллерам. Том I / М. Предко - М.: Постмаркет, 2001. - 416 с.
9. **Предко М.** Руководство по микроконтроллерам. Том II / М. Предко - М.: Постмаркет, 2001. - 488 с.
10. **Пухальский Г.И.** Проектирование микропроцессорных устройств: Учебное пособие для вузов / Г.И. Пухальский - СПб.: Политехника, 2001-544с.
11. **Пятибратов А.П.** Вычислительные системы, сети и телекоммуникации / А.П. Пятибратов, К.П. Гудыко – М.: Высшая школа, 2000.
12. Сайт фирмы Intel [электронный ресурс]: информационный сервер. – Режим доступа: http://www.intel.com.
13. Сайт фирмы Dallas Semiconductor [электронный ресурс]: информационный сервер. – Режим доступа: http:// www.dalsemi.com.
14. Сайт фирмы Siemens [электронный ресурс]: информационный сервер. – Режим доступа: http://www.siemens.com.