Министерство образования и науки Российской Федерации

Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых

Кафедра управления и информатики в технических
и экономических системах

Программирование однокристальных микроЭВМ семейства 8051

Методические указания к лабораторным работам по дисциплине

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА

Составители:
Кочуров О. М.
Кокорин С. А.

Владимир 2011

УДК 004.31

Рецензент

доцент Владимирского государственного университета
В. С. Грибакин

Программирование однокристальных микроЭВМ семейства 8051**:** Методические указания к лабораторным работам по дисциплине «Микропроцессорная техника» / Составители: Кочуров О. М., Кокорин С. А. — Владимир: ВлГУ — 2011, 28 с.

Приведены описания лабораторных работ, посвященных программированию распространенных микроконтроллеров семейства 8051. Рассматривается архитектура, система команд микроконтроллеров, основные принципы разработки и отладки программ.

Предназначены для студентов специальности — «Управление и информатика в технических системах».

Ил. 7, Табл. 6. Библиогр.: 3 назв.

УДК 004.31

Введение

В последние десятилетия цифровая электроника стремительно развивается. Совершенствование технологий, рост степени интеграции микросхем, увеличение быстродействия привело к появлению дешевых микропроцессоров. Вскоре после появления из них выделился класс однокристальных микро-ЭВМ или микроконтроллеров. Эти микросхемы фактически представляют собой готовую микропроцессорную систему, все элементы которой кроме устройств сопряжения, размещены на одном кристалле.

Кристалл микроконтроллера содержит следующие элементы:

1) микропроцессорное ядро: арифметико-логическое устройство, другие операционные узлы, устройство управления, блок регистров общего назначения;

2) память для хранения программ и данных;

3) генератор тактовых импульсов;

4) множество цифровых периферийных устройств, таких как параллельные и последовательные порты, счетчики;

5) устройства ввода и вывода аналоговых сигналов (аналого-цифровые и цифро-аналоговые преобразователи).

Микроконтроллеры нашли широкое применение в системах автоматического управления, как универсальное средство для ввода, программной обработки и вывода цифровых и аналоговых сигналов.

Традиционно микроконтроллеры строятся по гарвардской архитектуре, то есть используют разные запоминающие устройства для хранения программ и данных; обладают системой команд ориентированной на задачи сравнительно простой арифметической и логической обработки.

Однако в последние годы следует отметить рост многообразия существующих микроконтроллеров, разрушающий традиционную классификацию микропроцессорной техники.

Так, фирма AMD распространяет микроконтроллер с ядром классического процессора Intel 80188. Завоевывающее большую популярность семейство ARM использует единое адресное пространство для хранения программ и данных.

32-разрядные микроконтроллеры ARM с быстродействием около 100 миллионов операций в секунду вряд ли можно называть пригодными лишь для решения простейших задач.

Считавшийся самостоятельным класс сигнальных процессоров в настоящее время также фактически сливается с микроконтроллерами.

Многие производители стали выпускать микроконтроллеры с аппаратной поддержкой цифровой обработки сигналов.

Настоящий курс посвящен изучению наиболее популярной архитектуры 8-разрядных микроконтроллеров 8051. Подобная микросхема впервые была выпущена Intel в 1980 г. Архитектура получила чрезвычайно широкое распространение благодаря своей открытости. Она была поддержана десятками производителей. Среди наиболее известных NXP, Atmel, Winbond, Texas Instruments. Существует и отечественный аналог — микросхема КР1816ВЕ51.

Укажем на двух представителей семейства, представляющих особый интерес. Крупнейшим разработчиком аналоговой техники Analog Devices ядро 8051 было совмещено с прецизионным аналого-цифровым преобразователем (ADUC8xx). Наиболее развитым набором периферийных устройств обладают микросхемы фирмы Silicon Laboratories (8051Fxxx). Их высокое быстродействие (до 100 МГц) и сравнительно большие объемы встроенной памяти позволяют разрабатывать программы на языке С.

Таким образом, несмотря на почти тридцатилетнюю историю, благодаря непрекращающемуся развитию и сегодня семейство 8051 остается одним из самых популярных. Семейство поддерживается благодаря широкой известности, доступной документации и разнообразию отладочных средств.

1 Архитектура микроконтроллеров семейства 8051

Схемы программистской модели и организации памяти МК семейства 8051 изображены на рисунке 1.

Программистская модель (на рисунке в центре) включает аккумулятор A (Acc); расширитель аккумулятора B; восемь регистров общего назначения R0–R7, слово состояния процессора PSW, указатель данных DPTR, указатель стека SP.

Архитектуру 8051 можно назвать ориентированной на аккумулятор. Большинство инструкций используют его в качестве одного из операндов, поскольку не могут воздействовать непосредственно на регистры общего назначения и ячейки памяти. Это значит, что прежде чем произвести ту или иную операцию часто приходится сперва записать данные в аккумулятор, обработать, а затем выгрузить результат в память данных.

Расширитель аккумулятора B используется командами умножения и деления для хранения одного из операндов и части результата.

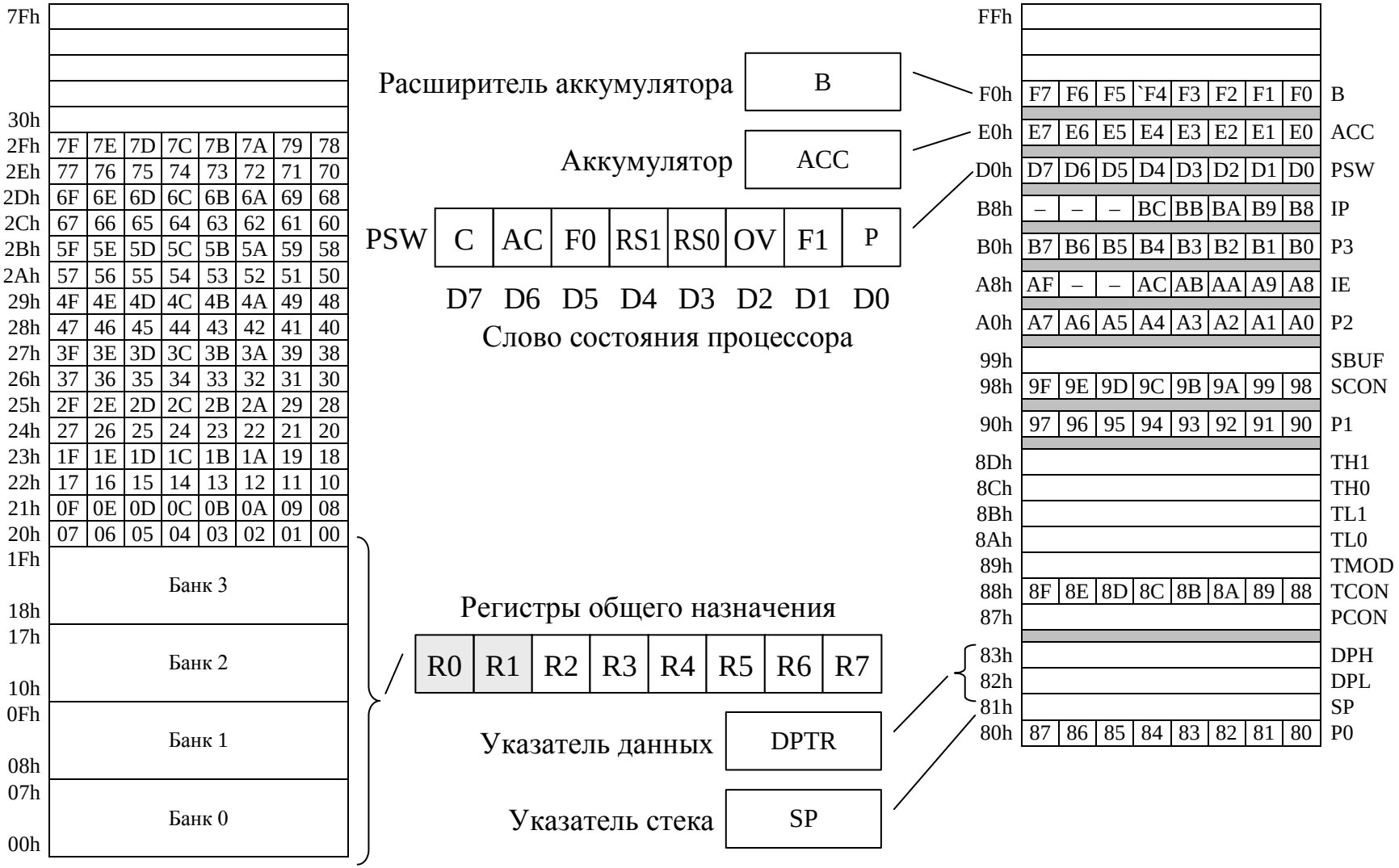

Рисунок 1 – Архитектура микроконтроллера семейства 8051

Регистры общего назначения предназначены для сокращения кода программы за счет уменьшения разрядности поля адреса. Адрес регистра занимает 3 бита, в то время как адрес ячейки памяти — восемь. Регистры R0 и R1, кроме того, используются для хранения исполнительного адреса при косвенной адресации ячеек памяти данных.

Регистр-указатель DPTR — 16-разрядный. Он также используется для хранения исполнительного адреса. Высокая разрядность необходима при обращении к внешней памяти данных или памяти программ.

Особенностью 8051 можно считать удобные команды пересылки. Одной командой возможна пересылка данных из любого источника в любой приемник: аккумулятор, регистр общего назначения, ячейка памяти данных, адресуемая прямо или косвенно, или константа.

В то же время команды арифметических и логических операций требуют размещения одного из аргументов в аккумуляторе. Приемником результата арифметических операций также является аккумулятор.

Рассмотрим организацию памяти данных микроконтроллера 8051. Младшие 128 адресов (00h–7Fh) соответствуют памяти данных (см. рисунок 5 слева). Регистры общего назначения (R0–R7) отображаются на один из банков памяти, расположенных по адресам 00h–07h, 08h–0Fh, 10h–17h, 18h–1Fh. Выбор банка памяти осуществляется через биты RS1:RS0 слова состояния процессора PSW.

Регистрам специальных функций (рисунок 5 справа) соответствуют старшие 128 адресов (80h–FFh). Обращение к ним возможно лишь с применением прямого метода адресации. Аккумулятор A, расширитель аккумулятора B, регистр DPTR, слово состояния процессора PSW и указатель стека SP включены в пространство адресов регистров специальных функций.

В микроконтроллерах 8052 встроенная память данных расширена до 256 байт. Адреса верхних 128 байт совпадают с адресами регистров специальных функций. Обращение к регистрам специальных функций производится прямым методом, а к расширенной памяти — косвенным.

Память данных микроконтроллеров 8051 может быть расширена при помощи внешних микросхем. Обращение к внешней памяти данных осуществляется при помощи специальной команды **`movx`**. Адресация внешней памяти производится только косвенно через регистр DPTR. С учетом разрядности этого регистра можно адресовать 64 кбайта памяти данных. Существуют микроконтроллеры, в которых «внешняя» память данных размещена на кристалле микроконтроллера, поэтому является фак-

тически внутренней, но для программиста обращение к ней производится, как к внешней.

К некоторым ячейкам памяти данных можно обращаться в битовом режиме. Для этого предназначены специальные команды. Биты имеют «сквозную» нумерацию от младшего бита ячейки памяти 20h (бит 00h) до старшего бита ячейки 2Fh (бит 7Fh). Многие из регистров специальных функций также могут быть изменены в битовом режиме (адреса битов 80h – FFh).

Слово состояния процессора содержит шесть флагов и два управляющих бита. Перечислим их и укажем назначение.

C — флаг переноса, устанавливается в единицу при переносе из старшего (седьмого) разряда АЛУ;

AC — дополнительный флаг переноса, устанавливается в единицу при переносе из третьего разряда АЛУ;

OV — флаг переполнения, устанавливается в единицу при переносе из шестого разряда АЛУ; означает потерю знака;

F0, F1 — определяются пользователем;

RS0, RS1 — биты выбора банка для регистров общего назначения.

P — бит четности; устанавливается в единицу, если аккумулятор содержит нечетное число единиц.

Приведенное описание передает лишь основные назначения флагов. Подробное описание влияния команд на слово состояния процессора приведено в книге [1]. Перечень команд, влияющих на регистр PSW, приведен в таблице 1.

Таблица 1 – Команды, влияющие на флаги состояния процессора

Команда	Флаги	Команда	Флаги	Команда	Флаги
ADD	C, OV, AC	DA	C	CPL C	C=C′
ADDC	C, OV, AC	RRC	C	ANL C	C
SUBB	C, OV, AC	RLC	C	ORL C	C
MUL	C=0, OV	SETB C	C=1	MOV C	C
DIV	C=0, OV	CLR C	C=0	CJNE	C

В стеке сохраняются адреса возврата при вызове подпрограмм и обработке прерываний. В микроконтроллерах 8051 стек организуется в ячейках памяти данных. Указателем вершины стека служит регистр SP. Указатель стека получает приращение автоматически перед записью в стек. По умолчанию он ссылается на ячейку памяти с адресом 07h, то есть стек начинается с адреса 08h.

Таблица 2 – Система команд микроконтроллеров семейства 8051

Мнемоническое обозначение	Описание		Методы адресации				Время
			Прям.	Косв.	Рег.	Неп.	
Группа команд пересылки							
MOV A, *Источник*	A = *Источник*		✓	✓	✓	✓	1
MOV *Приемник*, A	*Приемник* = A		✓	✓	✓		1
MOV *Приемник*, *Источник*	*Приемник* = *Источник*		✓	✓	✓	✓	2
MOV DPTR, #*const16*	DPTR = *const16*					✓	2
PUSH *Источник*	SP = SP + 1, [SP] = *Приемник*		✓				2
POP *Приемник*	*Приемник* = [SP], SP = SP – 1		✓				2
XCH A, *Байт*	A ⇔ *Байт*		✓	✓	✓		1
XCHD A, *Байт*	A<3:0> ⇔ *Байт*<3:0>			✓			1
MOVX A, *Источник*	A = *Источник*	Обращение к внешней памяти данных		✓			2
MOVX *Приемник*, A	*Приемник* = A			✓			2
MOVX A, @DPTR	A = [DPTR]		Косвенная через указатель				2
MOVX @DPTR, A	[DPTR] = A		Косвенная через указатель				2
MOVC A, @A+DPTR	A = [A + DPTR]	Чтение из памяти программ	Косвенная DPTR и аккум.				2
MOVC A, @A+PC	A = [A + PC]		Косв. через PC и аккум.				2
Группа команд арифметических операций							
ADD A, *Байт*	A = A + *Байт*		✓	✓	✓	✓	1
ADDC A, *Байт*	A = A + *Байт* + C		✓	✓	✓	✓	1
SUBB A, *Байт*	A = A – *Байт* – C		✓	✓	✓	✓	1
INC A	A = A + 1		Только аккумулятор				1
INC *Байт*	*Байт* = *Байт* + 1		✓	✓	✓		1
INC DPTR	DPTR = DPTR + 1		Только указатель				2
DEC A	A = A – 1		Только аккумулятор				1

Таблица 2 – Система команд микроконтроллеров семейства 8051 (продолжение)

Мнемоническое обозначение	Описание	Методы адресации				Время
		Прям.	Косв.	Рег.	Неп.	
Группа команд арифметических операций (продолжение)						
DEC *Байт*	*Байт* = *Байт* – 1	✓	✓	✓		1
MUL AB	B:A = B × A	Только аккумулятор				4
DIV AB	A = int(A/B)[1]; B = mod(A/B)[2]	Только аккумулятор				4
DA A	Десятичная коррекция A	Только аккумулятор				1
Группа команд логической обработки						
ANL A, *Байт*	A = A ∧ *Байт*	✓	✓	✓	✓	1
ANL *Байт*, A	*Байт* = *Байт* ∧ A	✓	✓	✓	✓	1
ANL *Байт*, #*const8*	*Байт* = *Байт* ∧ *const8*	✓				
ORL A, *Байт*	A = A ∨ *Байт*	✓	✓	✓	✓	1
ORL *Байт*, A	*Байт* = *Байт* ∨ A	✓	✓	✓	✓	1
ORL *Байт*, #*const8*	*Байт* = *Байт* ∨ *const8*	✓				
XRL A, *Байт*	A = A ⊕ *Байт*	✓	✓	✓	✓	1
XRL *Байт*, A	*Байт* = *Байт* ⊕ A	✓	✓	✓	✓	1
XRL *Байт*, #*const8*	*Байт* = *Байт* ⊕ *const8*	✓				
CLR A	A = 0	Только аккумулятор				1
CPL A	A = A′	Только аккумулятор				1
RL A	Сдвиг A влево	Только аккумулятор				1
RLC A	Сдвиг A влево через флаг переноса	Только аккумулятор				1
RR A	Сдвиг A вправо	Только аккумулятор				1
RRC A	Сдвиг A вправо через флаг переноса	Только аккумулятор				1
SWAP A	A<3:0> ⇔ A<7:4>	Только аккумулятор				1

Таблица 2 – Система команд микроконтроллеров семейства 8051 (продолжение)

Мнемоническое обозначение	Описание	Методы адресации				Время
		Прям.	Косв.	Рег.	Неп.	
Группа команд операций с битами						
ANL C, *Бит*	C = C ∧ *Бит*	✓				2
ANL C, /*Бит*	C = C ∧ *Бит′*	✓				2
ORL C, *Бит*	C = C ∨ *Бит*	✓				2
ORL C, /*Бит*	C = C ∨ *Бит′*	✓				2
MOV C, *Бит*	C = *Бит*	✓				1
MOV *Бит*, C	*Бит* = C	✓				1
CLR C	C = 0	✓				1
CLR *Бит*	*Бит* = 0	✓				1
CPL C	C = C′	✓				1
CPL *Бит*	*Бит* = *Бит′*	✓				1
SETB C	C = 1	✓				1
SETB *Бит*	*Бит* = 1	✓				1
Группа команд безусловных переходов						
JMP *Адрес*	PC = *Адрес*				✓	2
JMP @A+DPTR	PC = [A + DPTR]		✓			2
CALL *Адрес*	[SP] = PC, PC = *Адрес*, SP = SP + 1				✓	2
RET	PC = [SP], SP = SP − 1					2
RETI	Возврат из прерывания					2
Группа команд условных переходов						
JZ *Адрес*	PC = *Адрес*, если A = 0	Только аккумулятор				2
JNZ *Адрес*	PC = *Адрес*, если A ≠ 0	Только аккумулятор				2
DJNZ *Байт*, *Адрес*	*Байт* = *Байт*−1; PC = *Адрес*, если *Байт* ≠ 0	✓		✓		2

Таблица 2 – Система команд микроконтроллеров семейства 8051 (продолжение)

Мнемоническое обозначение	Описание	Методы адресации				Время
		Прям.	Косв.	Рег.	Неп.	
Группа команд условных переходов (продолжение)						
`CJNE A,` *Байт*`,` *Адрес*	`PC =` *Адрес*, если `A ≠` *Байт*	✓			✓	2
`CJNE` *Байт*`, #`*const8*`,` *Адрес*	`PC =` *Адрес*, если *Байт* ≠ *const8*		✓	✓		2
`JC` *Адрес*	`PC =` *Адрес*, если `C = 1`	✓				2
`JNC` *Адрес*	`PC =` *Адрес*, если `C = 0`	✓				2
`JB` *Бит*`,` *Адрес*	`PC =` *Адрес*, если *Бит* `= 1`	✓				2
`JNB` *Бит*`,` *Адрес*	`PC =` *Адрес*, если *Бит* `= 0`	✓				2
`JBC` *Бит*`,` *Адрес*	`PC =` *Адрес* и *Бит* `= 0`, если *Бит* `= 1`	✓				2

Примеры команд

```
        jmp   Label1    ; Записать в счетчик команд адрес, обозначенный меткой
        …
Label1: clr   23h       ; Сбросить бит с адресом 23h
        mov   A, 7Fh    ; Прямая адресация. Записать в аккумулятор содержимое
                        ; ячейки памяти с шестнадцатеричным адресом 7Fh
        add   A, R3     ; Регистровая адресация. Записать в аккумулятор
                        ; содержимое регистра общего назначения R3
        anl   A, @R0    ; Косвенная адресация. Выполнить логическое умножение
                        ; аккумулятора и ячейки памяти, адрес которой хранится в R0
        subb  A, #-12   ; Непосредственная адресация. Вычесть из аккумулятора
                        ; десятичное число -12
```

[1] `int(A/B)` — целая часть от деления `A/B`

[2] `mod(A/B)` — остаток от деления `A/B`

Остальные регистры специальных функций предназначены для управления периферийными устройствами, встроенными в МК. В базовой версии — это четыре порта ввода-вывода, два 16-разрядных таймера, последовательный приемопередатчик. В версии 8052 к ним добавлен третий таймер и массив программируемых таймеров. Описание этих регистров выходит за рамки методических указаний. Для изучения обращайтесь к [1].

2 Изучение основных приемов работы с интегрированной средой Keil µVision 3

Все операции, связанные с программированием микроконтроллеров выполняются с помощью популярной среды разработки Keil µVision 3 Среда включает текстовый редактор, ассемблер, СИ-компилятор и отладчик. В лабораторных работах используется демонстрационная версия программного обеспечения. Единственным ограничением является объем исполнимого кода, формируемого ассемблером — он не может превышать 2 кбайт.

Основное окно программы при работе в режиме редактора и отладчика показано на рисунках 2 и 3.

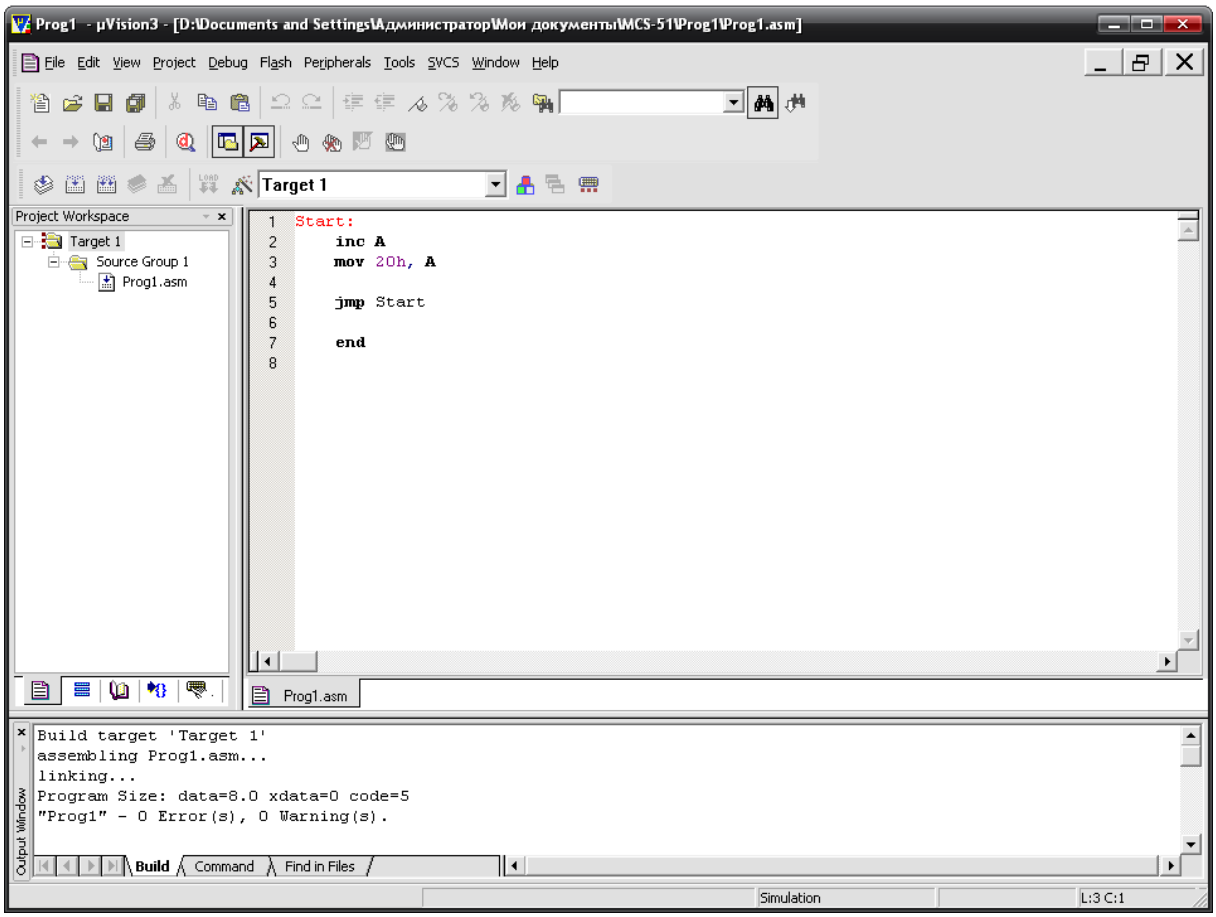

Рисунок 2 – Основное окно Keil µVision 3 в режиме программирования

Создание проекта

В современных средах разработки все файлы, относящиеся к создаваемой программе, объединяются в проекты. Фактически, проект — это файл, хранящий информацию о размещении на диске всех компонентов программы, параметры настройки среды разработки и т. п. Все файлы проекта должны быть размещены в одном каталоге.

Перед началом работы требуется создать рабочий каталог на жестком диске, в котором будут размещены файл с исходным текстом программы, объектный файл и другие файлы, создаваемые средой разработки.

Для создания нового проекта необходимо:

а) Воспользоваться пунктом меню **Project ⇒ New µVision Project**. При этом откроется стандартное окно сохранения файла.

б) Перейти в папку проекта и сохранить проект.

в) Выбрать тип имитируемого контроллера (см. рисунок 4). В нашем случае — **Generic ⇒ 8052 (All Variants)**. На вопрос «**Copy Standard 8051 Startup Code to Project Folder and Add File to Project**?» следует ответить «**Нет**».

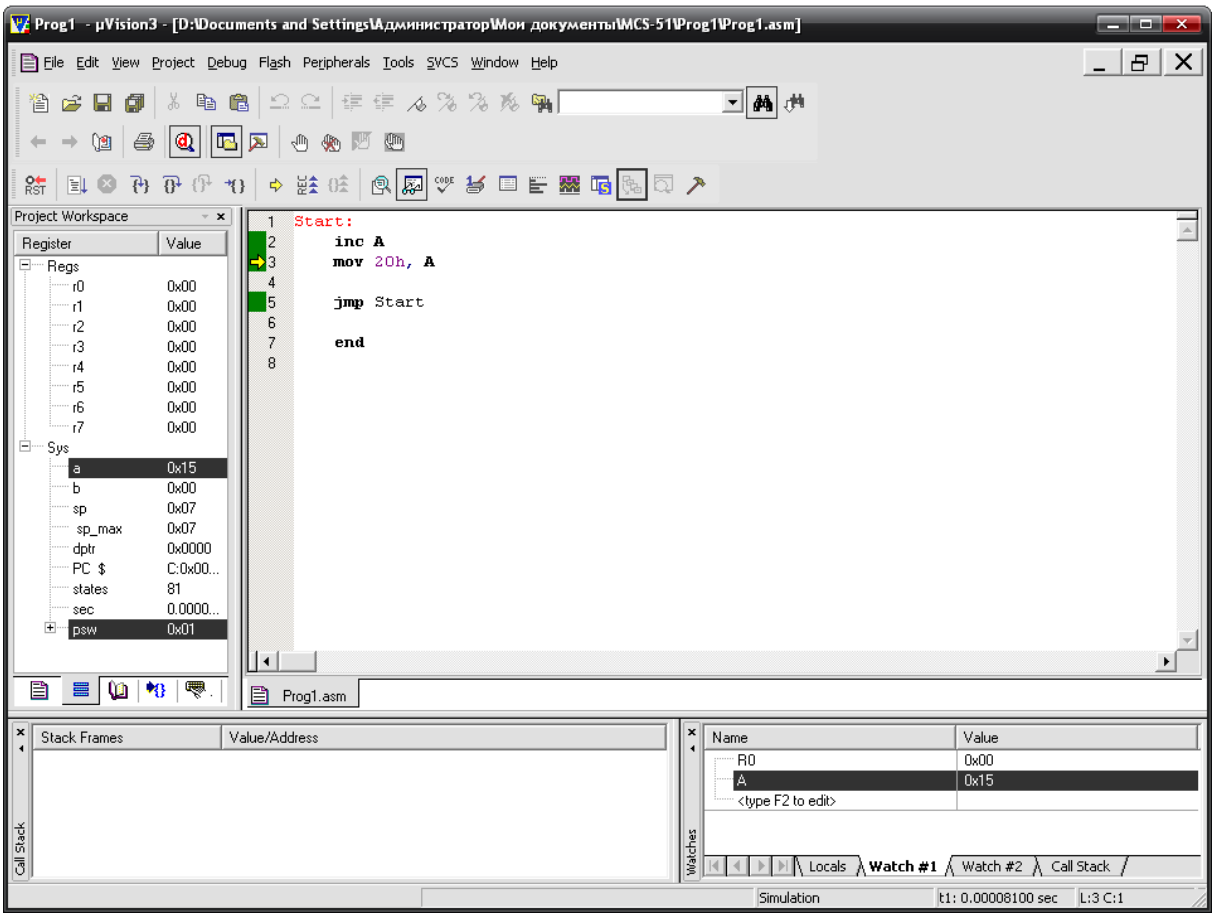

Рисунок 3 – Основное окно среды Keil µVision 3 в режиме отладчика

Создание файла программы

Файл программы представляет собой обычной текстовый файл с расширением *.ASM. Для создания файла необходимо выполнить следующие операции:

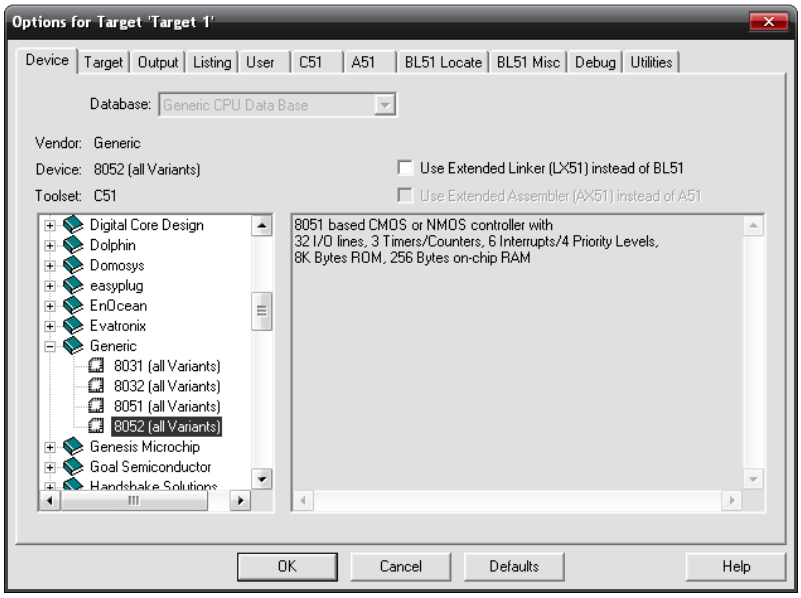

Рисунок 4 – Окно выбора микроконтроллера

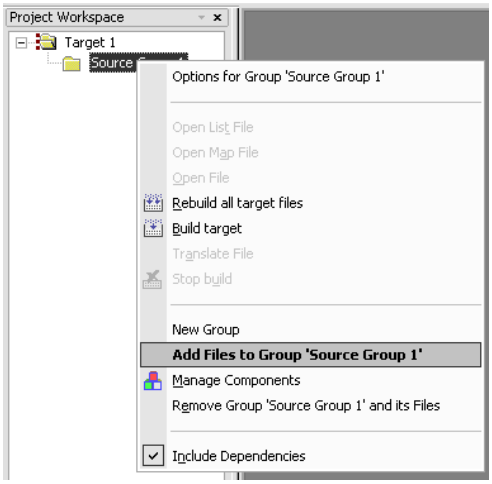

Рисунок 5

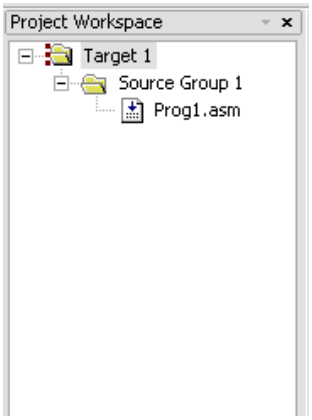

Рисунок 6

а) Создать новый файл командой меню **File** ⇒ **New**.

б) Сохранить пока еще пустой файл командой **File** ⇒ **Save**. Сохранять файл необходимо в папку проекта.

в) Щелчком правой кнопки мыши по пункту **Source Group 1** окна проекта (см. рисунок 5) открыть его контекстное меню.

г) Воспользоваться пунктом **Add Files to Group…** и выбрать файл программы (*.ASM) в открывшемся диалоговом окне. Имя файла программы должно появиться в окне проекта (см. рисунок 6).

Набор текста программы

Текст программы на ассемблере принято оформлять в четыре колонки. Каждая строка состоит из четырех полей: меток, операторов, операндов, комментариев. Поля отделяются друг от друга символом «табуляция» (кнопка TAB).

Листинг простейшей программы приведен ниже.

```
Start:
        inc     A           ; Инкремент аккумулятора
        mov     20h, A      ; Скопировать содержимое
                            ; A в ячейку 20h
        jmp     Start       ; Перейти на начало
        end
```

Ассемблирование программы

На данном этапе будет создан машинный код, готовый к записи в память программ микроконтроллера. Ассемблирование выполняется нажатием кнопки F7. При этом формируется отчет об ассемблировании, который можно просмотреть щелчком по вкладке **Output Window** в нижней части окна программы. В отчете, прежде всего, следует обратить внимание на число ошибок (**Errors**) и предупреждений (**Warnings**) (см. рисунок 7).

При наличии ошибок их необходимо исправить. Строка отчета, информирующая об ошибке, содержит имя файла программы, номера́ ошибочных строк, коды и описания ошибок. Двойным щелчком мыши по сообщению об ошибке можно быстро переместиться на строку программы, в которой эта ошибка содержится.

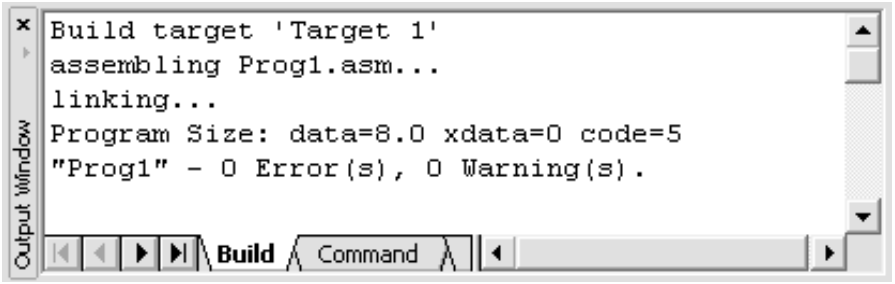

Рисунок 7 – Отчет об ассемблировании программы

Отладка программы

На этапе отладки моделируется работа программы. Среда разработки полностью имитирует поведение реального контролера. Можно наблюдать за пошаговым выполнением программы, за состоянием регистров и течением модельного времени. Переключение между режимом редактирования и отладки производится комбинацией клавиш Ctrl+F5.

Назначение кнопок наиболее часто используемых в режиме отладки приведены в таблице 3. Способы вызова диалоговых окон, необходимых для слежения за состоянием микроконтроллера и его периферийных устройств приведены в таблице 4.

Таблица 3 – Основные команды отладчика Keil μVision 3

Запуск программы	F5
Остановка программы	Debug ⇒ Stop Running
Системный сброс	Peripherals ⇒ Reset CPU
Пошаговое выполнение программы	F10
Пошаговое выполнение с заходом в подпрограммы	F11
Точка останова	Insert/Remove Breakpoint (в контекстном меню)

Таблица 4 – Основные диалоговые окна отладчика Keil μVision 3

Ячейки внутренней памяти данных	View ⇒ Memory Window
Содержимое памяти программ	View ⇒ Disassembly Window
Состояние объявленных переменных в памяти данных	View ⇒ Watch & Call Stack Window
Виртуальный терминал асинхронного приемопередатчика	View ⇒ Serial Window
Состояние встроенных периферийных устройств	Peripherals ⇒ …
Хронометраж	View ⇒ Performance Analyzer Window

3 Лабораторная работа №1. Изучение группы команд арифметических операций и слова состояния процессора

Цель работы. Освоить программирование основных арифметических операций с 8- и 16-разрядными числами. Познакомиться с назначением флагов состояния процессора. Продемонстрировать влияние команд арифметических операций на слово состояния процессора.

Подготовка к работе

Перед началом работы изучить разделы 3.8.1, 3.8.3 в [1].

Выполнить операции в соответствии с заданием. Для выполнения каждого задания (кроме последнего) потребуется всего две команды. Рекомендуется все команды размещать в одной программе, которая будет наращиваться по ходу выполнения работы. Отлаженные блоки можно вре-

менно отключать, превращая в комментарии символом «точка с запятой» в начале строки.

Перед выполнением и после выполнения каждой исследуемой команды фиксировать состояние процессора. Записывать

а) значение счетчика команд;

б) содержимое всех задействованных регистров и ячеек памяти в шестнадцатеричном формате;

в) слово состояния процессора PSW в двоичном формате; можно ограничиться значениями флагов CY, AC, OV, P.

Состояние процессора рекомендуется фиксировать, всякий раз заполняя таблицу, подобную таблице 5. В таблицу следует включать только регистры и ячейки памяти, на которые оказывает влияние (или может повлиять) исследуемая команда.

Фиксируя слово состояния процессора можно пользоваться двоичным форматом, шестнадцатеричным форматом с расшифровкой значений флагов CY, AC, OV, P. Можно для краткости указывать лишь флаги, равные единице. Остальные будут считаться нулевыми.

Таблица 5 – Пример записи результатов эксперимента

	До выполнения команды	После выполнения команды
PC	36h	39h
SP	70h	71h
PSW	00h	41h (P=1, AC=1)
A	0	12h
R0	10h	10h
71h	FFh	FEh

Задание

При выполнении всех пунктов следить, чтобы оба операнда и результат принадлежали диапазону представления чисел [–128; 127], если специально не оговорены другие условия.

1. *Сложение положительных чисел*. Выполнить сложение двух положительных чисел. Для этого загрузить командой **mov** в аккумулятор первое слагаемое, командой **add** прибавить к аккумулятору второе. Ниже приведены равнозначные примеры применения команды **mov**.

```
mov     A, #00001010b     ; Двоичное число
mov     A, #0A            ; Шестнадцатеричные
mov     A, #0xA           ; числа
mov     A, #10            ; Десятичное число
```

2. *Дополнительный перенос*. Загрузить в аккумулятор число 15. Прибавить к нему любое положительное число.

3. *Перенос*. Аналогично выполнить сложение отрицательного и положительного числа. Числа подобрать так, чтобы положительное число было больше по модулю.

4. *Вычитание с положительным результатом*. Вычесть из большего числа меньшее (командой **subb**).

5. *Вычитание с отрицательным результатом*. Вычесть из меньшего числа большее. Затем здесь же из результата вычесть ноль.

6. *Инкремент*. Загрузить в аккумулятор число –1. Командой **inc** произвести инкремент аккумулятора.

7. *Переполнение*. Сложить в аккумуляторе два числа таких, чтобы сумма оказалась больше 127 или меньше –128.

8. *Умножение*. Выполнить умножение (команда **mul**) любых двух чисел больше 100.

9. *Деление*. Выполнить деление (команда **div**) двух чисел. Делимое должно быть больше делителя и не должно быть кратным делителя.

10. *Наращивание разрядности*. Составить подпрограмму сложения двух 16-разрядных чисел. Сложение производится побайтно. Сначала складываются младшие байты, затем старшие. Для старших байтов необходимо использовать команду сложения с переносом (**addc**).

В отчет включить текст программы. Сопроводить его описанием состояния процессора до и после выполнения команд.

4 Лабораторная работа №2. Изучение организации памяти, методов адресации и группы команд пересылки данных

Цель работы. Изучить организацию памяти и методы адресации МК семейства 8051. Познакомиться с командами пересылки данных.

Подготовка к работе

Перед началом работы изучить раздел 3.8.2 в [1].

Программу необходимо начать с объявления переменного словаря. Следует ввести три имени, соответствующие адресам ячеек памяти данных. Создание пользовательских имен выполняется при помощи директивы **data**. Формат директивы следующий:

Имя **data** *Адрес*

После этого при ассемблировании программы вместо символического *Имени* будет подставлена числовая константа *Адрес*. Разумеется, в

программе можно использовать адреса непосредственно, но замена их заранее объявленными символическими именами удобна.

Итак, следует создать три имени для ячеек памяти, расположенных по адресам 20h–7Fh.

После выполнения каждой команды необходимо зафиксировать содержимое всех регистров общего назначения и ячеек памяти данных, на которые оказала влияние данная команда.

Просмотр содержимого ячеек памяти данных, адреса которых были введены директивой **`data`** удобно с помощью окна **View ⇒ Watch & Call Stack Window**. Символические имена интересующих ячеек вводятся в левом столбце на вкладке **Watch #1** (см. рисунок 3).

Задание

1. *Прямая адресация*. Записать в ячейку памяти данных любую числовую константу (команда **`mov`**). Использовать одну из ячеек памяти, объявленных в начале программы.

2. *Регистровая адресация*. Записать в любой регистр общего назначения (R0-R7) числовую константу. Проследить и зафиксировать адрес ячейки памяти данных, которая при этом изменится.

Изменить банк регистров общего назначения по умолчанию, установив его номер через биты RS0, RS1 слова состояния процессора. Напоминаем, что пара битов RS1:RS0 служит номером банка памяти данных, на которую отображаются регистры общего назначения. Например, если RS1:RS0=01, то выбран первый банк, и регистрам общего назначения соответствуют адреса 08h–0Fh.

Изменение битов слова состояния процессора рекомендуется производить командами **`setb`** и **`clr`**. Адреса битов RS0, RS1 см. на рисунке 1.

Повторить пункт 2 для нового банка регистров общего назначения. Зафиксировать значение номера банка в регистре PSW.

3. *Косвенная адресация*. Переслать данные из одной ячейки памяти в другую. Одну из ячеек памяти адресовать прямо, другую косвенно. Для косвенной адресации необходимо сперва загрузить исполнительный адрес в один из регистров R0 или R1, а затем использовать регистр в команде **`mov`** в качестве приемника, поставив перед его именем символ **`@`**.

4. *Чтение памяти программ через регистр-указатель*. Разместить в конце программы константу в памяти программ. Константа в память программ может быть «прошита» директивой **`db`**. Формат директивы

Метка`:` **`db`** *Константа*

Метка позволяет ссылаться на адрес константы в памяти программ.

Далее организовать загрузку константы из памяти программ в аккумулятор при помощи команды **movc**, предварительно загрузив адрес константы (то есть *Метку*) в регистр-указатель DPTR и очистив аккумулятор. При этом DPTR служит базовым адресом, а аккумулятор — смещением.

Зафиксировать адрес константы в памяти программ, значение аккумулятора и регистра DPTR, число, считанное из памяти программ.

Для просмотра содержимого памяти данных программ служит окно дизассемблера **View ⇒ Disassembly Window**.

5. *Чтение памяти программ через счетчик команд*. Считать ту же константу, используя в качестве базового адреса счетчик команд. Предварительно в аккумулятор следует загрузить смещение, которое вычисляется по формуле:

$$Acc = B - C - 1,$$

где *B* — адрес константы в памяти программ, *C* — адрес команды **movc**. Вместо *B* и *C* следует использовать соответствующие метки.

Зафиксировать адрес константы в памяти программ, адрес команды **movc**, число в аккумуляторе, число, считанное из памяти программ.

6. *Работа со стеком*. Установить указатель стека SP на любой из адресов памяти данных 20h–7Fh. Записать в стек содержимое одной из ячеек памяти данных командой **push**. Проследить и зафиксировать значение указателя стека до и после выполнения команды, адрес ячейки памяти, в которую фактически произошла запись.

Здесь удобно воспользоваться окном просмотра памяти данных **View ⇒ Memory Window**. Для вывода содержимого памяти данных в строке **Address** следует указать **D:0** (символ «D» указывает на память данных, число 0 задает начальный адрес для просмотра).

Далее считать содержимое стека в какую-либо ячейку памяти командой **poph**. Записать состояние процессора.

5 Лабораторная работа №3. Изучение группы команд передачи управления

Цель работы. Изучить группу команд передачи управления, научиться построению основных конструкций управления, таких как ветвления, циклы, подпрограммы.

Подготовка к работе

Перед началом работы изучить раздел 3.8.6 в [1].

По аналогии с заданием к лабораторной работе № 2 при помощи директивы **data** объявить имена двух произвольных ячеек памяти данных.

Задание

1. *Вызов подпрограммы*. Проинициализировать указатель стека, записав в РСФ SP адрес вершины стека. SP должен ссылаться на область старших адресов памяти, например 70h, чтобы не перекрыть хранимые в памяти данные. Разместить в конце программы подпрограмму. Подпрограмма должна начинаться с метки и заканчиваться командой возврата **ret**. Подпрограмма может быть пустой или включать произвольные команды, например, **nop**.

Организовать вызов подпрограммы из основной программы командой **call**.

До и после выполнения подпрограммы зафиксировать состояние указателя стека, вершины стека, счетчика команд.

2. *Вычисляемый переход*. Организовать вычисляемый переход, то есть пропуск заданного числа адресов памяти программ. Для этого служит команда **jmp @A+DPTR**. Предварительно необходимо записать в аккумулятор адрес команды **jmp** (при помощи метки), а в DPTR — индекс смещения, то есть шаг вычисляемого перехода. Рекомендуется после **jmp** разместить ряд пустых команд **nop**, на которые и будет передаваться управление.

3. *Цикл со счетчиком в аккумуляторе*. Записать в аккумулятор число итераций, после чего организовать цикл. Тело цикла начать с метки. В нем должна присутствовать команда декремента аккумулятора. Заканчивается цикл командой **jnz**, которая обеспечит проверку условия окончания цикла и передачу управления на начало тела цикла.

4. *Цикл со счетчиком в регистре*. Аналогично предыдущему заданию организовать цикл со счетчиком в одной из ячеек памяти данных или регистре общего назначения (на усмотрение студента). Использовать команду **djnz**, которая помимо всего прочего обеспечит и декремент счетчика.

5. *Цикл с опросом флага*. Организовать бесконечный цикл, который прерывается появлением единицы в одном из разрядов произвольной ячейки памяти данных. Использовать команду **jnb**. При тестировании вручную устанавливать единицу в нужном бите, пользуясь средствами симулятора. Адреса битов см. рисунок 1.

6. *Ветвление*. Организовать ветвление со сравнением двух ячеек памяти при помощи команды **cjne**. Предварительно поместить в программе две метки, обозначающие адреса ветвей и загрузить значениями сравниваемые ячейки памяти.

6. Лабораторная работа №4. Изучение системы прерываний

Цель работы. Изучить процесс обработки прерываний в микроконтроллерах семейства 8051.

Перед началом работы изучить раздел 3.6 в [1].

Задание

1. Разместить в начале программы команду безусловного перехода **jmp** на вектор входа. Например,

```
jmp   Start
```

2. Заполнить адреса векторов прерываний в соответствии с таблицей 6. Например, для вектора внешнего прерывания 0 вставить следующие строки:

```
org   03h
jmp   Ext0
```

Заполнить все адреса. Процедурам обработки прерываний должны быть присвоены индивидуальные метки.

Таблица 6 – Адреса векторов прерываний МК 8052

Источник	Адрес вектора прерывания
Внешнее 0	003h
Таймер 0	00Bh
Внешнее 1	013h
Таймер 1	01Bh
Последовательный порт	023h
Таймер 2	02Bh

3. Основную программу начать с метки, обозначающей точку входа. В примере выше использована метка **Start:**

4. Поместить команды настройки регистра прерываний IE и регистра приоритетов прерываний IP. Включить все прерывания установкой единиц во всех разрядах регистра IP. Назначить всем прерываниям низший (первый) приоритет путем записи нулей во все разряды регистра IP.

5. Разместить в конце основной программы вечный цикл.

6. Разместить в конце программы процедуры обработки всех прерываний. Каждая процедура должна начинаться с метки и заканчиваться командой возврата из прерывания (**reti**).

7. Проверить работу системы прерываний на примере прерывания от таймера 0 или 1. Для этого перевести симулятор в режим отладки; довести выполнение программы до вечного цикла. Вручную установить флаг прерывания от таймера 0 или 1 средствами симулятора (см. [1]). Убедить-

ся, что выполнение цикла прерывается переходом на адрес, соответствующий прерыванию таймера.

Зафиксировать изменения счетчика команд, стека, регистра управления таймером TCON.

8. Выполнить те же операции, имитируя прерывание от последовательного порта. Убедиться, что прерывание вызывается циклически.

9. Дополнить процедуру обработки прерывания от последовательного порта командами сброса соответствующего флага прерывания (**`clr`**). Снова протестировать работу программы, фиксируя изменения в счетчике команд и стеке.

10. Повысить приоритет процедуре прерывания от таймера 1. Создать ситуацию, при которой прерывание с низшим приоритетом происходит внутри процедуры обработки с высшим приоритетом, затем наоборот.

7 Лабораторная работа №5. Разработка простейшей программы генератора сигнала прямоугольной формы

Цель работы. Познакомиться с применением таймера и системы прерываний для решения чрезвычайно распространенной практической задачи формирования прямоугольного напряжения заданной частоты.

Задание

Перед началом работы изучить разделы 3.4, 3.6 в [1].

Разработать программу генератора прямоугольного сигнала с частотой 50 Гц (при тактовой частоте 12 МГц). В программе задействован один таймер, используется прерывание по переполнению таймера.

Общие указания по разработке программы

1. Программа должна начинаться с команды безусловного перехода **`jmp`** на точку входа (обход векторов прерываний).

2. По адресу, соответствующему вектору прерывания от таймера [1] необходимо разместить команду безусловного перехода на процедуру обработки прерываний. Процедура будет расположена в конце программы. Нужный адрес задается директивой **`org`**, предшествующей команде перехода. Формат директивы:

`org` *Адрес*

3. Основную программу начать с метки точки входа. Программа должна содержать команды настройки таймера и системы прерываний.

Настройка таймера включает выбор источника счетных импульсов (внешний/внутренний), задание номера режима 0–3 (регистр TMOD). Включение таймера (регистр TCON).

Настройка системы прерываний производится через регистр IE и состоит в установке глобального разрешающего бита и бита прерывания от таймера.

4. После того как настройка узлов микроконтроллера завершена, перейти к выполнению вечного цикла.

5. Процедура обработки прерываний должна включать команду останова таймера, присваивания таймеру начального значения, задающего частоту вызова прерывания, запуск таймера. Здесь же разместить команду инверсии произвольного разряда одного из портов ввода-вывода микроконтроллера (команды **`xrl`**). Процедура должна завершаться командой возврата **`reti`**.

6. Начальное значение таймера может быть рассчитано по формуле:

$$N = 2^{16} - \frac{T/2}{T_0} + N_0 = 65536 - \frac{T/2}{12/F_0} + N_0,$$

где T — период формируемого сигнала; T_0 — длительность машинного цикла; F_0 — тактовая частота (по умолчанию 12 МГц); N_0 — поправка, учитывающая затраты времени на выполнение переходов и команд процедуры обработки прерываний (подбирается экспериментально).

7. Протестировать программу. В ходе отладки рекомендуется придерживаться следующего порядка действий.

а) В пошаговом режиме (F10, F11) проследить за переходом на основную программу, выполнением команд и «зацикливанием» программы.

б) Убедиться, что таймер 0 запускается и инкрементируется в каждом машинном цикле (**Peripherals** ⇒ **Timer** ⇒ **Timer 0**).

в) Уставить точку останова в начале процедуры обработки прерываний от таймера. Убедиться, что процедура обработки прерываний запускается.

г) Пронаблюдать присваивание начальных значений регистрам таймера (**Peripherals** ⇒ **Timer** ⇒ **Timer 0**).

д) Проследить за изменением портовой линии P0.0 (**Peripherals** ⇒ **I/O-Ports** ⇒ **Port X**).

ж) Проконтролировать корректный возврат из подпрограммы обработки прерывания.

з) Измерить период вызова процедуры обработки прерываний; сравнить с желаемым временем (**View** ⇒ **Performance Analyzer Window**).

Дополнительное задание

Модифицировать программу генератора с использованием таймера 2 микроконтроллера 8052. Таймер 2 имеет программируемый модуль счета, то есть переполнение таймера может происходить при достижении произвольного значения.

1. Объявить регистры специальных функций микроконтроллера 8052 директивами

```
t2con   equ     0C8h
t2mod   equ     0C9h
rcap2l  equ     0CAh
rcap2h  equ     0CBh
```

2. В основной программе настроить таймер 2 через регистры T2MOD и T2CON. Задать модуль счета через пару регистров RCAP2H:RCAP2L.

3. Команды останова, запуска и перезагрузки таймера из процедуры обработки прерываний исключить.

4. Флаги прерывания от таймера 2 не снимаются аппаратно, поэтому в процедуру обработки прерывания необходимо включить команды очистки двух старших битов (6 и 7) регистра T2CON.

5. Протестировать программу.

8 Контрольные вопросы и задания

Для защиты лабораторных работ студент должен быть знаком с основами архитектуры МК 8051 по крайней мере в объеме параграфа 2 методических указаний.

Следует знать организацию памяти данных; структуру слова состояния процессора и значения флагов; назначение регистров специальных функций, к которым приходилось обращаться в ходе работ; назначение использованных команд.

Требуется знание представления положительных и отрицательных чисел в памяти микроконтроллера.

Необходимо владение основными приемами работы с интегрированной средой разработки.

Для самостоятельного контроля знания рекомендуется следующий перечень вопросов.

1. В чем особенность Гарвардской архитектуры ЭВМ?

2. Какие периферийные устройства встроены в базовый вариант микроконтроллера семейства 8051?

3. Поясните назначение следующих регистров специальных функций: A, B, PSW, SP, DPTR.

4. В чем отличие директив ассемблера от мнемонических команд?

5. Объясните назначение директив ассемблера DATA, ORG, DB.

6. Объясните назначение всех флагов состояния процессора. Опишите условия, при которых каждый из флагов изменяется, приведите примеры.

7. Что необходимо предпринимать для обработки данных разрядностью больше восьми?

8. Изобразите упрощенную схему организации памяти микроконтроллера семейства 8051. Обозначьте на ней область регистров общего назначения, адресов памяти данных, область памяти данных с битовой адресацией, область регистров специальных функций.

9. Какие методы адресации поддерживаются микроконтроллером семейства 8051? Объясните суть каждого метода адресации. Что служит исполнительным адресом, где он хранится (для разных методов)?

10. Какие методы адресации применяются для доступа к регистрам специальных функций, для доступа к внешней памяти данных, чтения памяти программ?

11. Каков максимальный объем внешней памяти данных?

12. Что такое стек? Для чего он предназначен?

13. Опишите процесс обработки прерывания в микроконтроллере. В каком случае инициируется этот процесс? Что происходит при возврате из прерывания?

14. Что такое приоритеты прерываний? Для чего предназначена поддержка многоприоритетных прерываний?

15. При помощи блок схемы изобразите структуру программы, использующей прерывания. Покажите на схеме основную программу, векторы прерывания и процедуры обработки прерываний.

16. Объясните назначение таймеров микроконтроллера. Расскажите об основных режимах их работы.

Список литературы

1. Сташин В. В. и др., Проектирование цифровых устройств на однокристальных микроконтроллерах. — М.: Энергоатомиздат, 1990. — 224 с.

2. Каган Б. М., Электронно-вычислительные машины и системы: Учеб. пособие для вузов — 3-е изд. перераб. и доп. — М.: Энергоатомиздат, 1991. — 592 с.

3. Микропроцессоры, в 3-х кн., под ред. Л. Н. Преснухина, кн. 1, Архитектура и проектирование микро-ЭВМ: Учеб. для вузов. — М.: Высш. шк., 1986. — 495 с.

Содержание