Министерство общего и профессионального образования
Российской Федерации
уральский государственный технический университет

СИСТЕМА КОМАНД МИКРОКОНТРОЛЛЕРА INTEL 8051

Методические указания к лабораторной работе №2
по курсу “Цифровые устройства и микропроцессоры”
для студентов всех форм обучения специальностей
200700 – Радиотехника;
201500 – Бытовая радиоэлектронная аппаратура

Екатеринбург 1999

УДК 681.322

Составители В.А.Добряк, В.К.Рагозин

Научный редактор доц., канд. техн. наук В.И.Елфимов

СИСТЕМА КОМАНД МИКРОКОНТРОЛЛЕРА INTEL 8051: Методические указания к лабораторной работе №2 по курсу “Цифровые устройства и микропроцессоры”/ В.А.Добряк, В.К.Рагозин. Екатеринбург: Изд-во УГТУ, 1999. 32 с.

Методические указания предназначены для использования при выполнении лабораторного практикума. Содержат описание системы команд микроконтроллера семейства Intel 8051, порядок выполнения домашнего и лабораторного заданий, контрольные вопросы и задания для самостоятельной работы.

Библиогр.: 4 назв. Рис. 19. Табл. 3. Прил. 3.

Подготовлено кафедрой радиоприёмных устройств.

ОГЛАВЛЕНИЕ

1. ЦЕЛЬ И СОДЕРЖАНИЕ РАБОТЫ

Целью работы является изучение системы команд микроконтроллеров популярного семейства Intel 8051, а также продолжение начатого в лабораторной работе №1 [3] изучения интегрированной среды ProView фирмы Franklin Software Inc., которая предназначена для разработки программного обеспечения этого семейства. Работа рассчитана на 4 часа домашней подготовки и 4 часа занятий в лаборатории.

При подготовке к работе изучается система команд одного из клонов семейства – микроконтроллера КМ1816ВЕ51, группы команд и особенности их выполнения. Выполняется домашнее задание, которое состоит в формировании наборов команд, выполняющих определенные в задании операции с заданными типами адресации операндов. Разрабатывается простейшая программа поразрядной обработки данных.

Перед началом лабораторной работы проводится коллоквиум. Студенты, успешно ответившие на поставленные вопросы, допускаются к лабораторной части работы.

В лаборатории выполняются несколько заданий, иллюстрирующих особенности системы команд. Правильность выбора команд в домашнем задании проверяется путем контроля содержимого регистров и памяти микроконтроллера при выполнении каждой команды. Выполняется отладка программы поразрядной обработки.

Затем оформляется отчёт с указанным ниже содержанием.

2. ЗАДАНИЯ ДЛЯ ДОМАШНЕЙ ПОДГОТОВКИ

2.1. Изучите систему команд микроконтроллеров семейства Intel 8051

Общие сведения о системе. Форматы команд, типы операндов. Способы адресации: регистровая, прямая, непосредственная, косвенная и неявная. Флаги результата. Символические имена регистров специальных функций и портов.

Группа команд передачи данных. Типы операндов и структура информационных связей. Обращение к аккумулятору и внешней памяти данных.

Арифметические команды. Операции сложения, вычитания, умножения и деления, десятичной коррекции, инкремента/декремента.

Команды логических операций. Источники и приёмники операндов.

Команды операций с битами.

Группа команд передачи управления. Длинный, абсолютный, относительный и косвенный переходы. Условные переходы. Подпрограммы. Работа со стеком [1, 2, 4].

2.2. Составьте набор из 26 команд

Для выполнения лабораторного задания составьте набор из 26 команд, выполняющих заданные в табл. 1 операции. Для некоторых из этих команд задан также способ адресации хотя бы одного из операндов.

Для каждой команды из набора необходимо определить источники операндов и приемник результата. Далее следует задать начальные численные значения (ненулевые) для регистров и ячеек памяти, участвующих в этой операции. Для некоторых команд необходимо определить и флаги.

По смыслу команд определить численные значения результатов операций. Реальные конечные значения, получаемые в результате выполнения операции, должны фиксироваться при выполнении лабораторного задания.

Таблица 1

Операции и способы адресации

№ команды	Операция	Способ адресации
1	Пересылка данных	Регистровая
2	Пересылка данных	Прямая
3	Пересылка данных	Косвенная
4	Пересылка данных	Непосредственная
5	Загрузка в стек	
6	Извлечение из стека	
7	Обмен байтами	Аккумулятор ↔ регистр
8	Обмен байтами	Аккумулятор ↔ память
9	Обмен младшими тетрадами	Аккумулятор ↔ резидентная память данных
10	Сложение	Регистровая
11	Сложение с переносом	Косвенная
12	Вычитание	Прямая
13	Вычитание	Непосредственная
14	Умножение	
15	Деление	
16	Логическое И	Регистровая
17	Логическое ИЛИ	Непосредственная
18	Исключающее ИЛИ	Прямая для первого операнда (приемника)
19	Циклический сдвиг влево	
20	Сдвиг вправо через бит переноса C	
21	Безусловный переход в пределах страницы	
22	Условный переход по флагу C	
23	Условный переход по произвольному биту (флаг пользователя)	
24	Сравнение и переход, если “не равно”	Непосредственная для второго операнда
25	Декремент и переход, если ≠ 0	Регистровая
26	Передача управления подпрограмме	

2.3. Составьте программу поразрядной обработки

Программа должна установить нулевой разряд числа в регистре R5 в 1, сбросить четвёртый разряд в 0 и инвертировать шестой разряд. Используйте команды логических операций числа с масками. Составьте контрольный пример для отладки программы.

2.4. Контрольные вопросы

1. По каким функциональным группам можно классифицировать команды микроконтроллера?
2. Какой формат может иметь команда?
3. Как длительность машинного цикла микроконтроллера соотносится с его тактовой частотой?
4. Как определить время выполнения команды?
5. С какими типами данных может оперировать микроконтроллер?
6. Для чего используются четырёхбитные операнды?
7. Какие команды работают с четырёхбитными операндами?
8. Для чего используются двухбайтные операнды?
9. Как косвенно адресуются байты памяти?
10. Укажите назначение флагов слова состояния программы PSW.
11. Сформулируйте условия установки флага OV.
12. Каково назначение регистров указателей?
13. Может ли порт одновременно являться источником операнда и приемником результата операции?
14. Какие способы адресации используются в микроконтроллере?
15. Можно ли адресовать порты и регистры специальных функций косвенно?
16. Приведите примеры команд передачи данных с различными способами адресации.
17. Расшифруйте команду MOVC A, @A+DPTR.
18. Приведите примеры команд доступа к 256 байтам резидентной памяти данных, к внешней памяти данных.
19. Приведите примеры логических и арифметических команд.
20. Как выполнить вычитание многобайтных операндов?
21. Перечислите команды операций с битами.
22. Как инвертировать отдельные биты портов?
23. Можно ли адресовать биты косвенно?
24. Какие переходы возможны в командах управления?
25. Для чего используются косвенные переходы в программах?
26. Поясните отличия длинного, абсолютного и относительного переходов в программах.
27. Как организовать процедуру ожидания с помощью одной команды?
28. Какие команды используются при организации подпрограмм?
29. Какие команды модифицируют флаги результата?
30. Укажите, какие из регистров специальных функций допускают битовую адресацию.
31. Какие флаги используются командами условных переходов?
32. Чем отличаются команды RET и RETI?

3. СИСТЕМА КОМАНД МИКРОКОНТРОЛЛЕРОВ INTEL 8051

3.1. Общие сведения

Определение и ассемблерная мнемоника команд, их тип в соответствии с рис. 1 (Т), число байтов в командах (Б), а также продолжительность исполнения команд в циклах (Ц) даны в прил. 1. Система содержит 111 базовых команд, которые по функциональному признаку могут быть разделены на пять групп:

- команды передачи данных,
- арифметические операции,
- логические операции,
- операции с битами,
- команды передачи управления.

Большинство команд имеют формат в один или два байта и выполняются за один или два машинных цикла. При тактовой частоте 12 МГц длительность машинного цикла составляет 1 мкс. На рис. 1 показаны 13 типов команд.

Первый байт команды любых типа и формата всегда содержит код операции (КОП). Второй и третий байты содержат либо адреса операндов, либо непосредственные операнды.

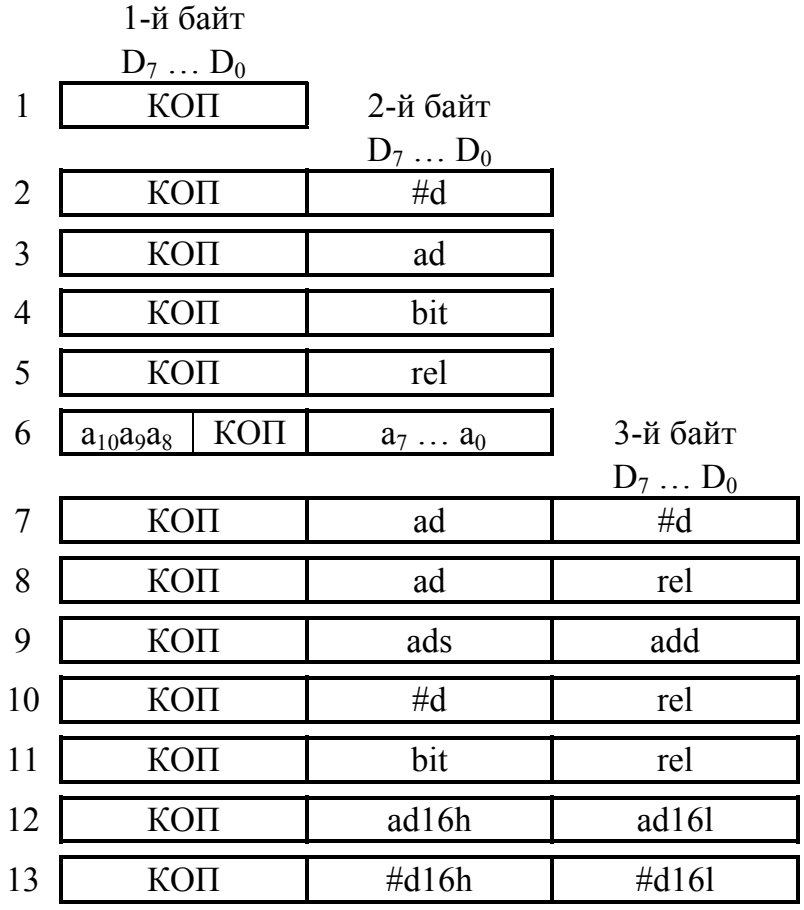


Рис. 1. *Типы команд*

3.1.1. Типы операндов

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

Микроконтроллер имеет 128 программно-управляемых флагов пользователя. Имеется также возможность адресации отдельных битов блока регистров специальных функций и портов. Для адресации битов используется прямой 8-битный адрес (bit). Косвенная адресация битов невозможна. Карты адресов отдельных битов представлены на рис. 2 и 3.

Четырёхбитные операнды используются только при операциях обмена SWAP и XCHD.

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

Рис. 2. *Карта адресуемых битов в резидентной памяти данных*

Прямой адрес бита	(D_7)							(D_0)	
0FFH									Имя регистра
0F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
0E0H	E7	E6	E5	E4	E3	E2	E1	E0	A
0D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
0B8H	-	-	-	BC	BB	BA	B9	B8	IP
0B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
0A8H	AF	-	-	AC	AB	AA	A9	A8	IE
0A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
90H	97	96	95	94	93	92	91	90	P1
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
80H	87	86	85	84	83	82	81	80	P0

Рис. 3. *Карта адресуемых битов в блоке регистров специальных функций*

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

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

3.1.2. Способы адресации данных

В микроконтроллере используются следующие способы адресации данных: прямая, непосредственная, косвенная и неявная.

При косвенном способе адресации резидентной памяти данных используются все восемь битов адресных регистров R0 и R1.

3.1.3. Флаги результата

Слово состояния программы PSW включает себя четыре флага: C - перенос, AC - вспомогательный перенос (полуперенос), OV - переполнение и P - паритет.

Флаг паритета напрямую зависит от текущего значения аккумулятора. Если число единичных битов аккумулятора нечётное, то флаг P устанавливается, а если чётное - сбрасывается. Все попытки изменить флаг P, присваивая ему новое значение, бесполезны, если содержимое аккумулятора при этом останется неизменным.

Флаг AC устанавливается, если при выполнении операции сложения или вычитания между тетрадами байта (полубайтами) возник перенос или заем.

Флаг C устанавливается, если в старшем бите результата возникает перенос или заем. При выполнении операций умножения и деления флаг C сбрасывается.

Флаг OV устанавливается, если результат операции сложения или вычитания не укладывается в семи битах и старший (восьмой) бит результата не может интерпретироваться как знаковый. При выполнении операции деления флаг OV сбрасывается, а в случае деления на нуль устанавливается. При умножении флаг OV устанавливается, если результат больше 255.

В табл. 2 перечисляются команды, при выполнении которых модифицируются флаги результата. В таблице отсутствует флаг паритета, так как его значение изменяется всеми командами, которые изменяют содержимое аккумулятора. Кроме команд, приведенных в таблице, флаги модифицируются командами, в которых местом назначения результата определены PSW или его отдельные биты, а также командами операций над битами.

Таблица 2

Команды, модифицирующие флаги результата

Команды	Флаги	Команды	Флаги
ADD	C, OV, AC	CLR C	C = 0
ADDC	C, OV, AC	CPL C	C = NOT(C)
SUBB	C, OV, AC	ANL C, b	C
MUL	C = 0, OV	ANL C, /b	C
DIV	C = 0, OV	ORL C, b	C
DA	C	ORL C, /b	C
RRC	C	MOV C, b	C
RLC	C	CJNE	C
SETB C	C = 1		

3.1.4. Символическая адресация

При использовании ассемблера для получения объектных кодов программ допускается применение в программах символических имен регистров специальных функций, портов и их отдельных битов (рис. 3).

Для адресации отдельных битов и портов (такая возможность имеется не у всех регистров специальных функций) можно использовать символическое имя бита следующей структуры: <имя регистра или порта>.<номер бита>.

Например, символическое имя пятого бита аккумулятора будет следующим: ACC.5. Символические имена являются зарезервированными словами, и их не надо определять с помощью директив ассемблера.

3.2. Команды передачи данных

Большую часть команд данной группы (табл. П.1.1) составляют команды передачи и обмена байтов. Команды пересылки битов представлены в группе команд битовых операций. Все команды данной группы не модифицируют флаги результата, за исключением команд загрузки PSW и аккумулятора (флаг паритета).

3.2.1. Структура информационных связей

В зависимости от способа адресации и места расположения операнда можно выделить девять типов операндов, между которыми возможен информационный обмен. Граф возможных операций передачи данных показан на рис. 4. Передачи данных могут выполняться без участия аккумулятора.

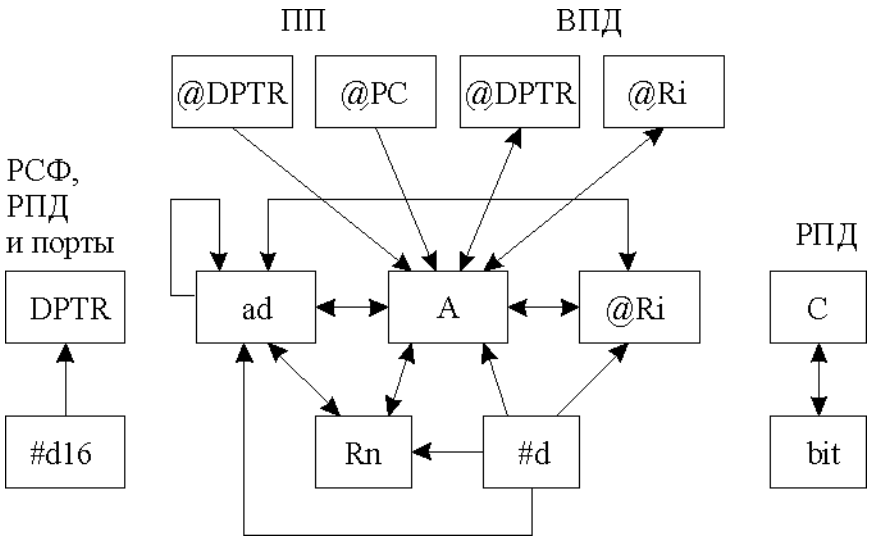


Рис. 4. *Граф путей передачи данных*

3.2.2. Обращение к аккумулятору

Обращение к аккумулятору может быть выполнено с использованием неявной и прямой адресации. В зависимости от способа адресации аккумулятора применяется одно из символических имен: A или ACC (прямой адрес). При прямой адресации обращение к аккумулятору производится как к одному из регистров специальных функций, и его адрес указывается во втором байте команды. Использование неявной адресации аккумулятора предпочтительнее, но не всегда возможно, например, при обращении к отдельным битам аккумулятора.

3.2.3. Обращение к внешней памяти данных

При использовании команд MOVX @Ri обеспечивается доступ к 256 байтам внешней памяти данных. Существует также режим обращения к расширенной внешней памяти данных, когда для доступа используется 16-битный адрес, храня-

щийся в регистре-указателе данных DPTR. Команды MOVX @DPTR обеспечивают доступ к 65536 байтам внешней памяти данных.

3.3. Арифметические операции

Данную группу образуют 24 команды (табл. П.1.2), выполняющие операции сложения, десятичной коррекции, инкремента/декремента байтов. Имеются команды вычитания, умножения и деления байтов.

Команды ADD и ADDC допускают сложение аккумулятора с большим числом операндов. Аналогично командам ADDC существуют четыре команды SUBB, что позволяет достаточно просто производить вычитание байтов и многобайтных двоичных чисел.

В микроконтроллере реализуется расширенный список команд инкремента/декремента байтов, команда инкремента 16-битного регистра-указателя данных.

3.4. Логические операции

Данную группу образуют 25 команд (табл. П.1.3), реализующих функционально полную систему логических операции над байтами. В микроконтроллере расширено число типов операндов, участвующих в операциях.

Имеется возможность производить операцию “исключающее ИЛИ” с содержимым портов. Команда XRL может быть эффективно использована для инверсии отдельных битов портов.

3.5. Команды передачи управления

К данной группе команд (табл. П.1.4) относятся команды, условного и безусловного ветвления, вызова подпрограмм и возврата из них, а также команда пустой операции NOP. В большинстве команд используется прямая адресация, т.е. адрес перехода целиком (или его часть) содержится в самой команде передачи управления. Можно выделить три разновидности команд ветвления по разрядности указываемого адреса перехода.

3.5.1. Длинный переход

Переход по всему адресному пространству памяти программ. В команде содержится полный 16-битный адрес перехода (ad16). Трехбайтные команды длинного перехода содержат в мнемокоде букву L (Long). Всего существует две такие команды: LJMP - длинный переход и LCALL - длинный вызов подпрограммы. На практике редко возникает необходимость перехода в пределах всего адресного пространства, и чаще используются укороченные команды перехода, занимающие меньше места в памяти.

3.5.2. Абсолютный переход

Переход в пределах одной страницы памяти программ размером 2048 байтов. Такие команды содержат только 11 младших битов адреса перехода (ad11). Команды абсолютного перехода имеют формат 2 байта. Начальная буква мнемокода - A (Absolute). При выполнении команды в вычисленном адресе следующей по порядку команды ((PC) = (PC) + 2) 11 младших битов заменяются на ad11 из тела команды абсолютного перехода.

3.5.3. Относительный переход

Короткий относительный переход позволяет передать управление в пределах от – 128 до +127 байт относительно адреса следующей команды (команды, следующей по порядку за командой относительного перехода). Существует одна команда короткого безусловного перехода SJMP (Short). Все команды условного перехода используют данный метод адресации. Относительный адрес перехода (rel) содержится во втором байте команды.

3.5.4. Косвенный переход

Команда JMP @A + DPTR позволяет передавать управление по косвенному адресу. Эта команда удобна тем, что предоставляет возможность организации перехода по адресу, вычисляемому самой программой и неизвестному при написании исходного текста программы.

3.5.5. Условные переходы

Система условных переходов предоставляет возможность осуществлять ветвление по следующим условиям: аккумулятор содержит нуль (JZ), содержимое аккумулятора не равно нулю (JNZ), перенос равен единице (JC), перенос равен нулю (JNC), адресуемый бит равен единице (JB), адресуемый бит равен нулю (JNB).

Для организации программных циклов удобно пользоваться командой DJNZ. В качестве счётчика циклов может использоваться не только регистр, но и прямоадресуемый байт (например, ячейка резидентной памяти данных).

Команда CJNE эффективно используется в процедурах ожидания какого-либо события. Например, команда

```
WAIT: CJNE A, P0, WAIT
```

будет выполняться до тех пор, пока на линиях порта 0 не установится информация, совпадающая с содержимым аккумулятора.

Все команды данной группы, за исключением CJNE и JBC, не оказывают воздействия на флаги. Команда CJNE устанавливает флаг C, если первый операнд оказывается меньше второго. Команда JBC сбрасывает флаг C в случае перехода.

3.5.6. Подпрограммы

Для обращения к подпрограммам необходимо использовать команды вызова подпрограмм LCALL и ACALL. Эти команды в отличие от команд перехода LJMP и AJMP сохраняют в стеке адрес возврата в основную программу. Для возврата из подпрограммы необходимо выполнить команду RET. Команда RETI отличается от команды RET тем, что разрешает прерывания обслуженного уровня.

3.6. Операции с битами

Отличительной особенностью данной группы команд (табл. П.1.5) является то, что они оперируют с однобитными операндами. В качестве таких операндов могут выступать отдельные биты некоторых регистров специальных функций и портов, а также 128 программных флагов пользователя.

Существуют команды сброса (CLR), установки (SETB) и инверсии (CPL) битов, а также конъюнкции и дизъюнкции бита и флага переноса. Для адресации битов используется прямой восьмиразрядный адрес (bit). Косвенная адресация битов невозможна.

4. ЛАБОРАТОРНЫЕ ЗАДАНИЯ

4.1. Выполнение примеров программ

В пошаговом режиме работы ProView выполните исследование приведённых ниже примеров программ. Первый пример содержит подробное описание последовательности действий. Остальные примеры исследуются самостоятельно.

4.1.1. Пример использования команд передачи данных

Пример 1. Запись данных. Записать в резидентную память данных по адресам 41 и 42 число 1C3FH:

```
LOAD:     MOV  R0,#41H   ;загрузка в R0 указателя данных
          MOV  @R0,#1CH  ;запись в память числа 1CH
          INC  R0        ;инкремент указателя
          MOV  @R0,#3FH  ;запись в память числа 3FH
```

Дополним программу директивами и командами ассемблера, обеспечивающими отладку и тестирование:

```
START:    JMP LOAD  ;переход к программе
          ORG  30H  ;директива размещения программы с адреса 30
LOAD:     MOV  R0,#41H   ;загрузка в R0 указателя данных
          MOV  @R0,#1CH  ;запись в память числа 1CH
          INC  R0        ;инкремент указателя
          MOV  @R0,#3FH  ;запись в память числа 3FH
          JMP START      ;зацикливание программы
          END            ;директива окончания трансляции
```

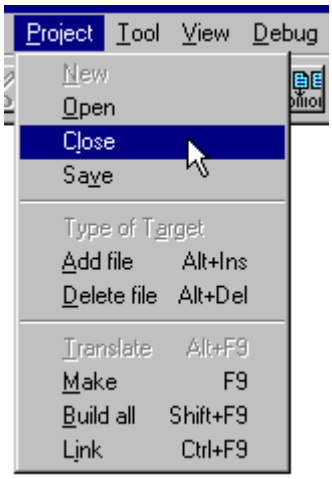


Рис. 5. *Закрытие проекта*

Переходим к созданию проекта и тестированию программы в среде ProView. Большинство шагов Вам уже знакомо по лабораторной работе №1 [3]. Специфика состоит только в том, что текст программы написан на языке ассемблера.

Шаг 1. Загрузить программу ProView. Если после загрузки программы открывается предыдущий проект, с которым производилась работа, то необходимо закрыть его через меню (рис. 5).

Шаг 2. Создать новый проект (рис.6).

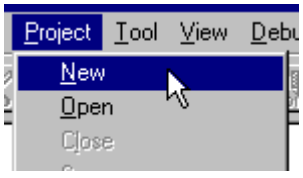


Рис. 6. *Создание нового проекта*

Укажите имя файла проекта в вашей личной папке (рис. 7).

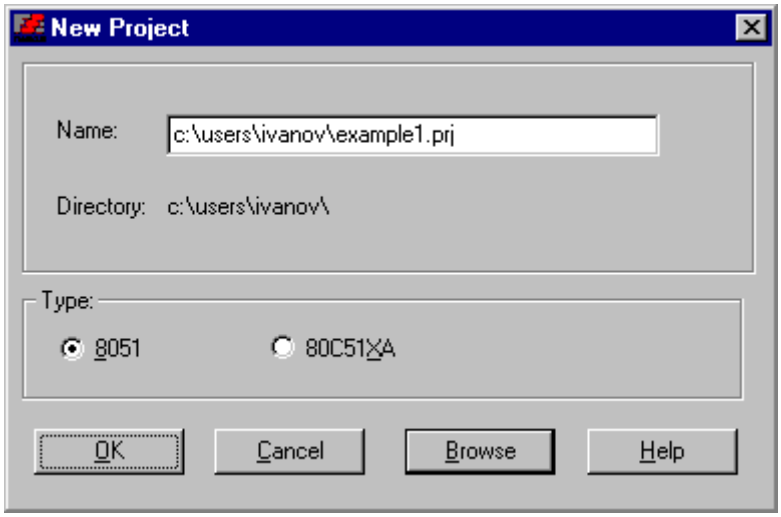


Рис. 7. *Имя и путь файла проекта*

Шаг 3. Создать новый файл для последующего ввода текста программы (рис. 8) и указать тип файла (рис. 9).

Рис. 8. *Новый файл*

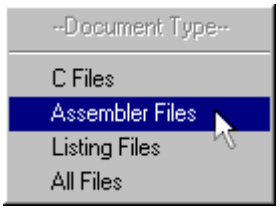


Рис. 9. *Тип файла*

Откроется пустое окно для ввода ассемблерного текста программы. Записать этот файл (рис. 10).

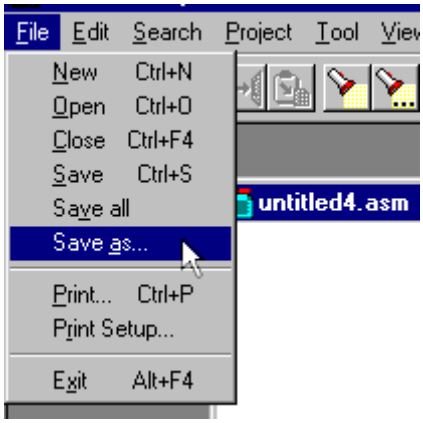


Рис. 10. *Запись файла*

Укажите имя и расширение имени файла (рис. 11).

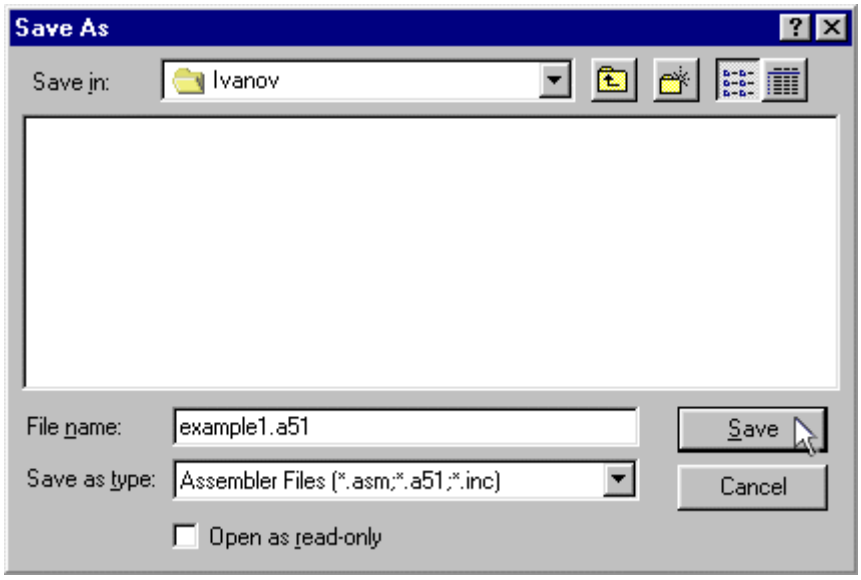


Рис. 11. *Имя файла*

Шаг 4. Ввести текст программы:

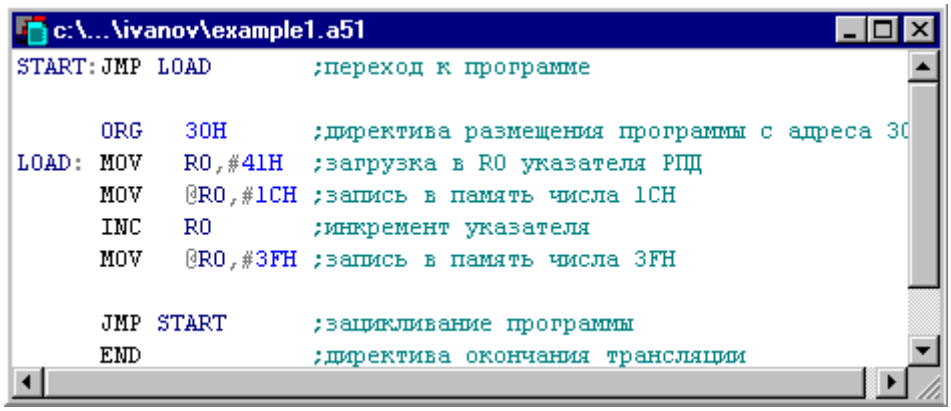
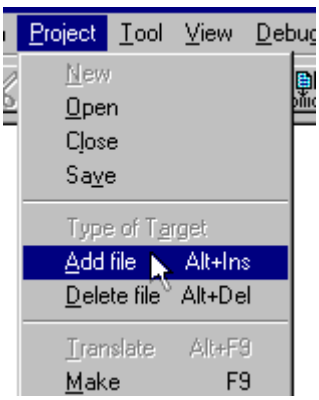


Рис. 12. *Текст программы*



Шаг 5. Подключить этот файл к проекту, предварительно активизировав окно проекта (рис. 13, 14).

Рис. 13. *Подключение файла к проекту*

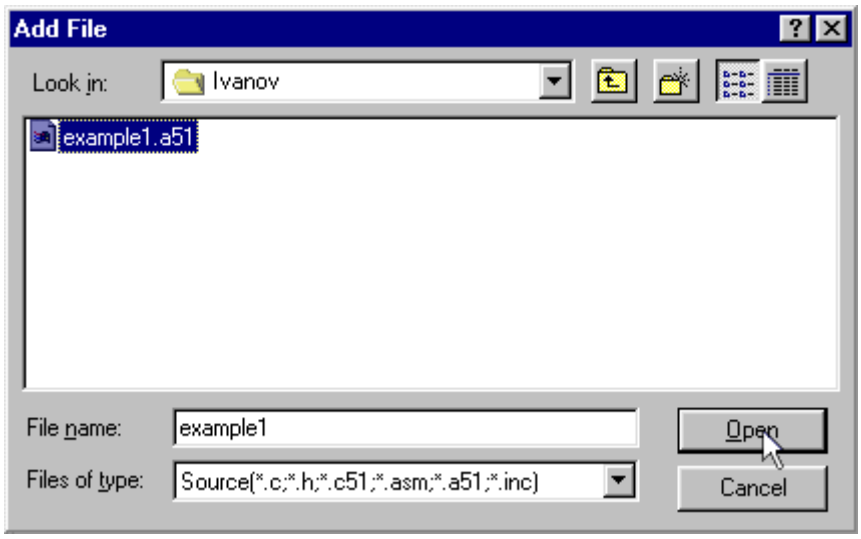
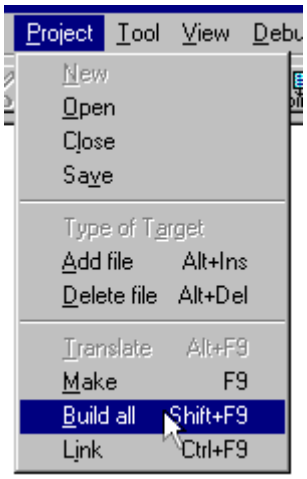


Рис. 14. *Выбор файла для подключения*

Шаг 6. Выполнить компиляцию проекта (рис. 15).

Шаг 7. Запустить программу на выполнение с помощью команды Start из меню Debug, указав необходимые опции отладчика (рис. 16).



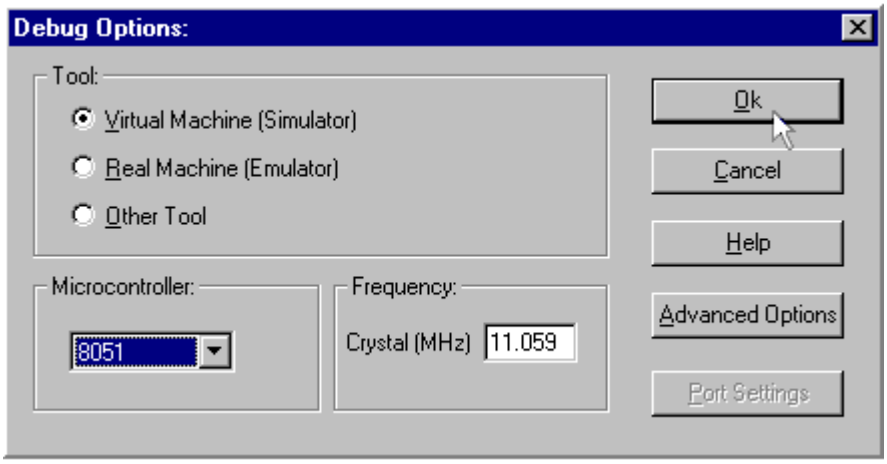
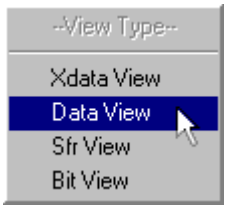


Рис. 15. *Компиляция*

Рис. 16. *Окно опций отладчика*



Шаг 8. С помощью команды Data dump из меню View (рис. 17) открыть окно резидентной памяти данных (рис. 18).

Рис. 17. *Выбор окна резидентной памяти данных*

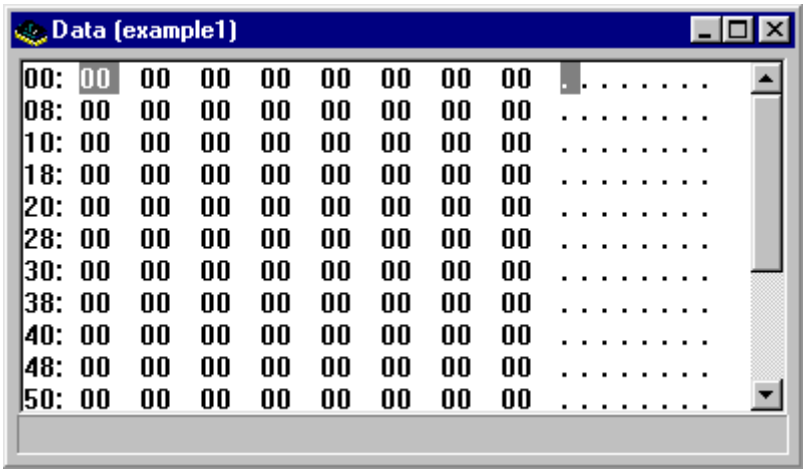


Рис. 18. *Окно резидентной памяти данных*

Дальнейшие шаги. Выполните программу в пошаговом режиме. Наблюдая изменение содержимого окна резидентной памяти данных и окна Main Registers, убедитесь в правильности работы программы. Определите время выполнения программы.

4.1.2. Примеры использования команд арифметических операций

Пример 2. Сложение. Сложить два двоичных многобайтных числа. Слагаемые располагаются в резидентной памяти данных, начиная с младшего байта. Начальные адреса слагаемых заданы в R0 и R1, формат слагаемых в байтах - в R2:

```
        CLR  C          ;сброс переноса
LOOP:   MOV  A, @R0     ;загрузка в A текущего байта первого
                        ;слагаемого
        ADDC A, @R1     ;сложение байтов с учетом переноса
        MOV  @R0, A     ;размещение байта результата
        INC  R0         ;продвижение указателей
        INC  R1
        DJNZ R2, LOOP   ;цикл, если не все байты просуммированы
```

При сложении чисел без знака на переполнение укажет флаг C, а в случае сложения чисел со знаком - флаг OV.

Дополните программу сложения командами, обеспечивающими её тестирование, составьте контрольный пример и выполните отладку в ProView. Определите время вычисления в зависимости от формата исходных чисел.

Пример 3. Умножение. Команда MUL вычисляет произведение двух целых беззнаковых чисел, хранящихся в регистрах A и B. Младшая часть произведения размещается в A, а старшая - в регистре-расширителе B. Если содержимое B оказывается равным нулю, то флаг OV сбрасывается, иначе - устанавливается. Флаг переноса всегда сбрасывается. Например, если аккумулятор содержал число 200 (0C8H), а расширитель 160 (0A0H), то в результате выполнения команды MUL AB

получится произведение 32000 (7D00H). Аккумулятор будет содержать нуль, а расширитель – 7DH, флаг OV будет установлен, а флаг C - сброшен.

Пусть требуется умножить целое двоичное число произвольного формата на константу 73. Исходное число размешается в резидентной памяти данных, адрес младшего байта находится в регистре R0. Формат числа в байтах хранится в R1:

```
        MOV  A,#0       ;сброс аккумулятора
LOOP:   ADD  A,@R0      ;загрузка множимого
        MOV  B,#73      ;загрузка множителя
        MUL  AB         ;умножение
        MOV  @R0,A      ;запись младшего байта частичного
                        ;произведения
        INC  R0         ;приращение адреса
        MOV  A,B        ;пересылка старшего байта частичного
                        ;произведения в аккумулятор
        XCH  A,@R0      ;предварительное формирование
                        ;очередного байта произведения
        DJNZ R1,LOOP    ;цикл, если не все байты исходного
                        ;числа умножены на константу
```

Полученное произведение размещается на месте исходного числа и занимает в памяти на один байт больше.

Разберитесь в алгоритме умножения. Дополните программу командами, обеспечивающими её тестирование, составьте контрольный пример и выполните отладку в ProView. Определите время вычисления в зависимости от формата исходного числа.

Пример 4. Деление. Команда DIV производит деление содержимого аккумулятора на содержимое регистра-расширителя. После деления аккумулятор содержит целую часть частного, а расширитель - остаток. Флаги C и OV сбрасываются. При делении на нуль устанавливается флаг переполнения, а частное остается неопределенным. Команда деления может быть использована для быстрого преобразования двоичных чисел в десятичные двоично-кодированные (BCD-числа).

В качестве примера рассмотрим программу, которая переводит двоичное число, содержащееся в аккумуляторе, в BCD-код. При таком преобразовании может получиться трёхразрядное BCD-число. Старшая цифра (число сотен) будет размещена в регистре R0, а две младшие в аккумуляторе:

```
MOV  B,#100     ;загрузка 100 для вычисления количества сотен
DIV  AB         ;аккумулятор содержит число сотен (старшую цифру)
MOV  R0,A       ;пересылка в R0 старшей цифры
XCH  A,B        ;пересылка остатка исходного числа в аккумулятор
MOV  B,#10      ;загрузка 10 для вычисления количества десятков
DIV  AB         ;A содержит число десятков, B - число единиц
SWAP A          ;размещение числа десятков в старшей тетраде A
ADD  A,B        ;подсуммирование остатка (числа единиц),
                ;теперь аккумулятор содержит две младшие цифры
```

Разберитесь в алгоритме перевода. Дополните программу командами, обеспечивающими её тестирование, составьте контрольный пример и выполните отладку в ProView. Определите время вычисления.

4.1.3. Пример использования команд логических операций

Пример 5. Логические операции. Выполните отладку программы поразрядной обработки, которая была подготовлена при выполнении домашнего задания. **Убедитесь в её работоспособности на контрольных примерах. Определите время вычисления.**

4.1.4. Пример использования команд передачи управления и работы со стеком

Пример 6. Операции со стеком. Перед загрузкой в стек содержимое регистра-указателя стека SP инкрементируется, а после извлечения из стека декрементируется.

По сигналу системного сброса в SP заносится начальное значение 07H. Для переопределения SP можно воспользоваться командой MOV SP, #d.

Таким образом, стек может располагаться в любом месте резидентной памяти данных. Стек используется для организации обращений к подпрограммам и при обработке прерываний, может быть использован для передачи параметров подпрограммам и для временного хранения содержимого регистров специальных функций.

Подпрограмма должна сохранить в стеке содержимое тех регистров, которые она сама будет использовать, а перед возвратом в прерванную программу должна восстановить их значения.

Подпрограмма с дополнениями для её тестирования может, например, иметь следующую структуру:

```
START:    MOV  R1,#02H   ;загрузка регистров
          MOV  A,#30H
          MOV  R2,#00H
          LCALL SUB      ;переход на подпрограмму
          SJMP START

SUB:      PUSH PSW  ;сохранение в стеке PSW
          PUSH ACC  ;сохранение аккумулятора
          PUSH B    ;сохранение расширителя аккумулятора B
          ADD  A,R1 ;собственно обработка данных
          MOV  R2,A
          POP  B    ;восстановление B
          POP  ACC  ;восстановление аккумулятора
          POP  PSW  ;восстановление PSW
          RET       ;возврат

          END
```

Если предположить, что SP перед возникновением прерывания содержал значение 1FH, то размещение регистров в стеке после входа в подпрограмму обработки будет таким, как на рис. 19.

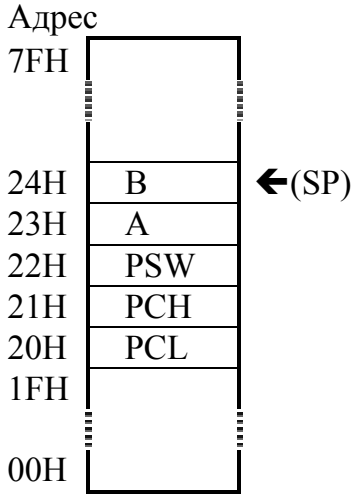


Рис. 19. *Содержимое стека после выполнения команды CALL и серии команд PUSH*

Исследуйте процесс выполнения команд вызова и возврата из подпрограммы, а также команд работы со стеком. Для этого запустите программу в пошаговом режиме.

Замените команду POP PSW на NOP и проследите, как будет выполняться программа. Объясните происшедшие изменения, сформулируйте выводы для отчёта.

4.2. Выполнение домашнего набора команд

Запустите на выполнение команды из набора, сформированного при выполнении домашнего задания. Сделать это можно различными способами, например:

```
      JMP  M1
      ORG  30H
M1:   <исследуемая команда>
      END
```

Команда из набора записывается во второй строке программы за меткой M1 и будет размещена по адресу 30H в резидентной памяти программ. Далее осуществляется ассемблирование и загрузка программы. Перед пуском программы необходимо с помощью мыши и клавиатуры задать начальные значения (операнды) для тех регистров и ячеек памяти, которые участвуют в выполнении соответствующей команды. Затем осуществляется пуск программы, после чего необходимо зафиксировать результаты операции.

Далее описанный процесс повторяется для следующей команды и т.д. Для команд, модифицирующих флаги слова состояния программы PSW, необходимо также зафиксировать изменения состояния битов C, OV, AC.

Результаты выполнения операций сводятся в табл. 3, которую следует включить в отчёт о лабораторной работе.

Таблица 3

Результаты выполнения команд

№	Команда	Код	Выполняемая операция	Содержимое регистров и памяти до и после выполнения		Пояснение
				До	После	
1	MOV A,R0	E8	Пересылка байта данных из регистра R0 в аккумулятор A	A/00 R0/F2	A/F2 R0/F2	
2	…	…	…	…	…	…
…	…	…	…	…	…	…
26	…	…	…	…	…	…

Проанализировать соответствие результатов определению выполняемых операций. Выделить команды, модифицирующие флаги слова состояния программы PSW, и пояснить состояния флагов после выполнения этих команд.

5. СОДЕРЖАНИЕ ОТЧЁТА

Отчёт о лабораторной работе должен содержать:

- титульный лист;
- цель и задачи работы;
- исходные тексты программ с дополнениями, обеспечивающими тестирование и отладку, результаты решения контрольных примеров;
- выводы по работе.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. М.: Энергоатомиздат, 1990. 224 с.
2. Однокристальные микроЭВМ/ А.В.Боборыкин, Г.П.Липовецкий, Г.В.Литвинский и др. М.: МИКАП, 1994. 400 с.
3. Ваша первая программа для микроконтроллера Intel 8051: Методические указания к лабораторной работе №1 по курсу “Микропроцессоры и вычислительные устройства”/ Добряк В.А. Екатеринбург: УГТУ, 1999. 32 с.
4. Микропроцессоры. В 3-х кн. Кн. 1. Архитектура и проектирование микро-ЭВМ. Организация вычислительных процессов: Учебник для втузов/ П.В.Нестеров, В.Ф.Шаньгин, В.Л.Горбунов и др.; Под ред. Л.Н.Преснухина. М.: Высшая школа, 1986. 495 с.

Приложение 2
Система команд в алфавитном порядке

Мнемокод	КОП	Мнемокод	КОП	Мнемокод	КОП
ACALL 0xxH	11	ANL A, R4	5C	DJNZ R1, rel	D9
ACALL 1xxH	31	ANL A, R5	5D	DJNZ R2, rel	DA
ACALL 2xxH	51	ANL A, R6	5E	DJNZ R3, rel	DB
ACALL 3xxH	71	ANL A, R7	5F	DJNZ R4, rel	DC
ACALL 4xxH	91	ANL A, @R0	56	DJNZ R5, rel	DD
ACALL 5xxH	B1	ANL A, @R1	57	DJNZ R6, rel	DE
ACALL 6xxH	D1	ANL A, #d	54	DJNZ R7, rel	DF
ACALL 7xxH	F1	ANL ad, A	52	INC A	04
ADD A, ad	25	ANL ad, #d	53	INC ad	05
ADD A, R0	28	ANL C, bit	82	INC DPTR	A3
ADD A, R1	29	ANL C, /bit	B0	INC R0	08
ADD A, R2	2A	CJNE A, ad, rel	B5	INC R1	09
ADD A, R3	2B	CJNE A, #d, rel	B4	INC R2	0A
ADD A, R4	2C	CJNE R0, #d, rel	B8	INC R3	0B
ADD A, R5	2D	CJNE R1, #d, rel	B9	INC R4	0C
ADD A, R6	2E	CJNE R2, #d, rel	BA	INC R5	0D
ADD A, R7	2F	CJNE R3, #d, rel	BB	INC R6	0E
ADD A, @R0	26	CJNE R4, #d, rel	BC	INC R7	0F
ADD A, @R1	27	CJNE R5, #d, rel	BD	INC @R0	06
ADD A, #d	24	CJNE R6, #d, rel	BE	INC @R1	07
ADDC A, ad	35	CJNE R7, #d, rel	BF	JB bit, rel	20
ADDC A, R0	38	CJNE @R0, #d, rel	B6	JBC bit, rel	10
ADDC A, R1	39	CJNE @R1, #d, rel	B7	JC rel	40
ADDC A, R2	3A	CLR A	E4	JMP @A + DPTR	73
ADDC A, R3	3B	CLR bit	C2	JNB bit, rel	30
ADDC A, R4	3C	CLR C	C3	JNC rel	50
ADDC A, R5	3D	CPL A	F4	JNZ rel	70
ADDC A, R6	3E	CPL bit	B2	JZ rel	60
ADDC A, R7	3F	CPL C	B3	LCALL ad16	12
ADDC A, @R0	36	DA A	D4	LJMP ad16	02
ADDC A, @R1	37	DEC A	14	MOV A, ad	E5
ADDC A, #d	34	DEC ad	15	MOV A, R0	E8
AJMP 0XXH	01	DEC R0	18	MOV A, R1	E9
AJMP 1XXH	21	DEC R1	19	MOV A, R2	EA
AJMP 2XXH	41	DEC R2	1A	MOV A, R3	EB
AJMP 3XXH	61	DEC R3	1B	MOV A, R4	EC
AJMP 4XXH	81	DEC R4	1C	MOV A, R5	ED
AJMP 5XXH	A1	DEC R5	1D	MOV A, R6	EE
AJMP 6XXH	C1	DEC R6	1E	MOV A, R7	EF
AJMP 7XXH	E1	DEC R7	1F	MOV A, @R0	E6
ANL A, ad	55	DEC @R0	16	MOV A, @R1	E7
ANL A, R0	58	DEC @R1	17	MOV A, #d	74
ANL A, R1	59	DIV AB	84	MOV ad, A	F5
ANL A, R2	5A	DJNZ ad, rel	D5	MOV ad, R0	88
ANL A, R3	5B	DJNZ R0, rel	D8	MOV ad, R1	89

Система команд в алфавитном порядке (окончание)

Мнемокод	КОП	Мнемокод	КОП	Мнемокод	КОП
MOV ad, R2	8A	MOV @R1, A	F7	SUBB A, ad	95
MOV ad, R3	8B	MOV @R1, ad	A7	SUBB A, R0	98
MOV ad, R4	8C	MOV @R1, #d	77	SUBB A, R1	99
MOV ad, R5	8D	MOVC A, @ + DPTR	93	SUBB A, R2	9A
MOV ad, R6	8E	MOVC A, @ + PC	83	SUBB A, R3	9B
MOV ad, R7	8F	MOVX A, @DPTR	E0	SUBB A, R4	9C
MOV ad, @R0	86	MOVX A, @R0	E2	SUBB A, R5	9D
MOV ad, @R1	87	MOVX A, @R1	E3	SUBB A, R6	9E
MOV ad, #d	75	MOVX @DPTR, A	F0	SUBB A, R7	9F
MOV add, ads	85	MOVX @R0, A	F2	SUBB A, @R0	96
MOV bit, C	92	MOVX @R1, A	F3	SUBB A, @R1	97
MOV C, bit	A2	MUL AB	A4	SUBB A, #d	94
MOV DPTR, #16	90	NOP	00	SWAP A	C4
MOV R0, A	F8	ORL A, ad	45	XCH A, ad	C5
MOV R0, ad	A8	ORL A, R0	48	XCH A, R0	C8
MOV R0, #d	78	ORL A, R1	49	XCH A, R1	C9
MOV R1, A	F9	ORL A, R2	4A	XCH A, R2	CA
MOV R1, ad	A9	ORL A, R3	4B	XCH A, R3	CB
MOV R1, #d	79	ORL A, R4	4C	XCH A, R4	CC
MOV R2, A	FA	ORL A, R5	4D	XCH A, R5	CD
MOV R2, ad	AA	ORL A, R6	4E	XCH A, R6	CE
MOV R2, #d	7A	ORL A, R7	4F	XCH A, R7	CF
MOV R3, A	FB	ORL A, @R0	46	XCH A, @R0	C6
MOV R3, ad	AB	ORL A, @R1	47	XCH A, @R1	C7
MOV R3, #d	7B	ORL A, #d	44	XCHD A, @R0	D6
MOV R4, A	FC	ORL ad, A	42	XCHD A, @R1	D7
MOV R4, ad	AC	ORL ad, #d	43	XRL A, ad	65
MOV R4, #d	7C	ORL C, bit	72	XRL A, R0	68
MOV R5, A	FD	ORL C, /bit	A0	XRL A, R1	69
MOV R5, ad	AD	POP ad	D0	XRL A, R2	6A
MOV R5, #d	7D	PUSH ad	C0	XRL A, R3	6B
MOV R6, A	FE	RET	22	XRL A, R4	6C
MOV R6, ad	AE	RETI	32	XRL A, R5	6D
MOV R6, #d	7E	RL A	23	XRL A, R6	6E
MOV R7, A	FF	RLC A	33	XRL A, R7	6F
MOV R7, ad	AF	RR A	03	XRL A, @R0	66
MOV R7, #d	7F	RRC A	13	XRL A, @R1	67
MOV @R0, A	F6	SETB bit	D2	XRL A, #d	64
MOV @R0, ad	A6	SETB C	D3	XRL ad, A	62
MOV @R0, #d	76	SJMP rel	80	XRL ad, #d	63

Старшая цифра	Младшая							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	NOP	AJMP 0XXH	LJMP ad16	RR A	INC A	INC ad	INC @R0	INC @R1
1	JBC bit, rel	ACALL 0XXH	LCALL ad16	RRC A	DEC A	DEC ad	DEC @R0	DEC @R1
2	JB bit, rel	AJMP 1XXH	RET	RL A	ADD A, #d	ADD A, ad	ADD A, @R0	ADD A, @R1
3	JNB bit, rel	ACALL 1XXH	RETI	RLC A	ADDC A, #d	ADDC A, ad	ADDC A, @R0	ADDC A, @R1
4	JC rel	AJMP 2XXH	ORL ad, A	ORL ad, #d	ORL A, #d	ORL A, ad	ORL A, @R0	ORL A, @R1
5	JNC rel	ACALL 2XXH	ANL ad, A	ANL ad, #d	ANL A, #d	ANL A, ad	ANL A, @R0	ANL A, @R1
6	JZ rel	AJMP 3XXH	XRL ad, A	XRL ad, #d	XRL A, #d	XRL A, ad	XRL A, @R0	XRL A, @R1
7	JNZ rel	ACALL 3XXH	ORL C, bit	JMP @A+DPTR	MOV A, #d	MOV ad, #d	MOV @R0, #d	MOV @R1, #d
8	SJMP rel	AJMP 4XXH	ANL C, bit	MOVC A, @A+PC	DIV AB	MOV add, ads	MOV ad, @R0	MOV ad, @R1
9	MOV DPTR, #d16	ACALL 4XXH	MOV bit, C	MOVC A,@A+DPTR	SUBB A, #d	SUBB A, ad	SUBB A, @R0	SUBB A, @R1
A	ORL C, /bit	AJMP 5XXH	MOV C, bit	INC DPTR	MUL AB		MOV @R0, ad	MOV @R1, ad
B	ANL C, /bit	ACALL 5XXH	CPL bit	CPL C	CJNE A, #d, rel	CJNE A, ad, rel	CJNE @R0,#d, rel	CJNE @R1,#d, rel
C	PUSH ad	AJMP 6XXH	CLR bit	CLR C	SWAP A	XCH A, ad	XCH A, @R0	XCH A, @R1
D	POP ad	ACALL 6XXH	SETB bit	SETB C	DA A	DJNZ ad, rel	XCHD A, @R0	XCHD A, @R1
E	MOVX A, @DPTR	AJMP 7XXH	MOVX A, @R0	MOVX A, @R1	CLR A	MOV A, ad	MOV A, @R0	MOV A, @R1
F	MOVX @DPTR, A	ACALL 7XXH	MOVX @R0, A	MOVX @R1, A	CPL A	MOV ad, A	MOV @R0, A	MOV @R1, A

X – шестнадцатиричная цифра.

Приложение 3
Шестнадцатиричные коды команд КМ1816ВЕ51

цифра								**Старшая цифра**
8	**9**	**A**	**B**	**C**	**D**	**E**	**F**	
INC R0	INC R1	INC R2	INC R3	INC R4	INC R5	INC R6	INC R7	**0**
DEC R0	DEC R1	DEC R2	DEC R3	DEC R4	DEC R5	DEC R6	DEC R7	**1**
ADD A, R0	ADD A, R1	ADD A, R2	ADD A, R3	ADD A, R4	ADD A, R5	ADD A, R6	ADD A, R7	**2**
ADDC A, R0	ADDC A, R1	ADDC A, R2	ADDC A, R3	ADDC A, R4	ADDC A, R5	ADDC A, R6	ADDC A, R7	**3**
ORL A, R0	ORL A, R1	ORL A, R2	ORL A, R3	ORL A, R4	ORL A, R5	ORL A, R6	ORL A, R7	**4**
ANL A, R0	ANL A, R1	ANL A, R2	ANL A, R3	ANL A, R4	ANL A, R5	ANL A, R6	ANL A, R7	**5**
XRL A, R0	XRL A, R1	XRL A, R2	XRL A, R3	XRL A, R4	XRL A, R5	XRL A, R6	XRL A, R7	**6**
MOV R0, #d	MOV R1, #d	MOV R2, #d	MOV R3, #d	MOV R4, #d	MOV R5, #d	MOV R6, #d	MOV R7, #d	**7**
MOV ad, R0	MOV ad, R1	MOV ad, R2	MOV ad, R3	MOV ad, R4	MOV ad, R5	MOV ad, R6	MOV ad, R7	**8**
SUBB A, R0	SUBB A, R1	SUBB A, R2	SUBB A, R3	SUBB A, R4	SUBB A, R5	SUBB A, R6	SUBB A, R7	**9**
MOV R0, ad	MOV R1, ad	MOV R2, ad	MOV R3, ad	MOV R4, ad	MOV R5, ad	MOV R6, ad	MOV R7, ad	**A**
CJNE R0, #d, rel	CJNE R1, #d, rel	CJNE R2, #d, rel	CJNE R3, #d, rel	CJNE R4, #d, rel	CJNE R5, #d, rel	CJNE R6, #d, rel	CJNE R7, #d, rel	**B**
XCH A, R0	XCH A, R1	XCH A, R2	XCH A, R3	XCH A, R4	XCH A, R5	XCH A, R6	XCH A, R7	**C**
DJNZ R0, rel	DJNZ R1, rel	DJNZ R2, rel	DJNZ R3, rel	DJNZ R4, rel	DJNZ R5, rel	DJNZ R6, rel	DJNZ R7, rel	**D**
MOV A, R0	MOV A, R1	MOV A, R2	MOV A, R3	MOV A, R4	MOV A, R5	MOV A, R6	MOV A, R7	**E**
MOV R0, A	MOV R1, A	MOV R2, A	MOV R3, A	MOV R4, A	MOV R5, A	MOV R6, A	MOV R7, A	**F**

СИСТЕМА КОМАНД
МИКРОКОНТРОЛЛЕРА INTEL 8051

Составители Добряк Вадим Алексеевич
Рагозин Владимир Константинович

Редактор Н.П.Кубыщенко

Подписано в печать 08.02.99 Формат 60x84 1/16
Бумага типографская Офсетная печать Усл. п. л. 1,86
Уч.-изд. л. 1,78 Тираж 100 Заказ 39 Цена “С”

Издательство УГТУ
620002, Екатеринбург, Мира, 19
ЗАО УМЦ УПИ. 620002, Екатеринбург, Мира, 17