Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра радиоэлектронных средств

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ И ИХ ПРИМЕНЕНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ

для студентов специальности
I-38 02 03 «Техническое обеспечение безопасности»
заочной формы обучения

Минск 2007

УДК 004.31(075.8)
ББК 32.973.26–04 я 73
М 59

Р е ц е н з е н т
проф. кафедры ЭВС БГУИР,
канд. техн. наук И. М. Русак

С о с т а в и т е л и :
В. М. Логин, И. Н. Цырельчук

М 59 **Микропроцессорные** системы и их применение : метод. указания к выполнению контр. работы для студ. спец. I-38 02 03 «Техническое обеспечение безопасности» заоч. формы обуч. / сост. В. М. Логин, И. Н. Цырельчук. – Минск : БГУИР, 2007. – 32 с. : ил.

Изложены требования к содержанию, объёму и оформлению контрольной работы, а также дана литература по дисциплине. Приводятся краткие теоретические сведения и методические указания по программированию 8-разрядных микроконтроллеров семейства MC68HC11 фирмы Motorola. Содержатся контрольные вопросы по следующим темам: методы адресации и команды пересылки данных, арифметические команды, логические команды, команды работы с битовыми полями и команды сдвигов, команды передачи управления и специальные команды.

УДК 004.31(075.8)
ББК 32.973.26–04 я 73

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Предмет дисциплины «Микропроцессорные системы и их применение» – принципы организации микропроцессорных систем различной сложности, алгоритмы их функционирования и методы проектирования.

Цель – изучение основ микропроцессорной техники, ее технической реализации и применения.

Содержание – основные сведения о функционировании процессора, организации микропроцессоров, микроконтроллеров и персональных компьютеров, проектировании устройств на микропроцессорных системах.

Дисциплина базируется на знаниях общеобразовательных (физики, математики) и специальных дисциплин (основы радиоэлектронных средств, проектирование устройств цифровой обработки информации, языки программирования) и курсов, связанных с цифровыми электронными системами.

В соответствии с учебным планом подготовки специалистов по заочной форме обучения специальности I-38 02 03 «Техническое обеспечение безопасности» и рабочей программой по дисциплине «Микропроцессорные системы и их применение» каждый студент должен в межсессионный период выполнить контрольную работу.

Выполнение контрольной работы является важным звеном в обучении студентов-заочников и преследует следующие цели:

– привить навыки и оказать помощь в организации самостоятельной работы в межсессионный период;

– указать правильную последовательность в изучении учебной дисциплины;

– систематизировать учебный материал;

– привить навыки самостоятельного изучения материала, применения теоретических знаний для решения практических задач;

– выработать умение анализировать достоинства и недостатки отдельных технических решений;

– проверить правильность понимания теоретических вопросов;

– научить студента грамотно и лаконично излагать материал;

– проверить и оценить работу студента-заочника в межсессионный период по изучению данной дисциплины.

В контрольной работе студент должен продемонстрировать понимание предложенных вопросов и показать знание теории предмета.

Задание по контрольной работе выдаётся на установочной сессии и выполняется по указанию преподавателя.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ

Приступая к выполнению контрольной работы, студент должен выписать из каждого раздела данных методических указаний контрольные вопросы своего варианта, уяснив, какого ответа требуют предлагаемые вопросы. Затем изучить учебную дисциплину по предложенному теоретическому материалу и рекомендованной литературе, руководствуясь учебной программой, с тем, чтобы иметь общее представление обо всем материале учебной дисциплины и понять взаимосвязь предложенных в контрольной работе вопросов с другими вопросами дисциплины. После этого можно приступать к более глубокому изучению материала по тем вопросам, которые заданы в контрольной работе, и подготовке ответа на них.

Отрабатывать вопросы контрольной работы следует по нескольким рекомендованным пособиям. При этом можно привлекать и другие источники, не приведённые в списке рекомендованной литературы, в том числе и из Интернета, делая обязательные ссылки на источники. Невыполнение этого условия может квалифицироваться как нарушение авторских прав.

Контрольную работу следует писать самостоятельно, не допуская компиляции и плагиата, иначе она может быть возвращена на доработку со сменой задания.

Контрольная работа выполняется машинописным способом на стандартных листах бумаги формата А4 и должна включать титульный лист, лист задания, содержание, содержательную часть и список использованной литературы.

Лист задания должен включать перечисление вопросов заданного варианта.

Содержание должно включать вопросы заданного варианта с названием разделов, подразделов, пунктов и подпунктов с указанием страниц контрольной работы, с которых начинаются ответы на эти вопросы.

В содержательной части даются ответы на поставленные вопросы, причём ответ на вопрос должен начинаться с формулировки вопроса. Каждый вопрос и ответ на него должен начинаться с новой страницы. Ответ должен быть лаконичным, полным, точным и по существу вопроса.

После ответа на все вопросы контрольной работы приводится полный список литературы, которая была использована при выполнении работы. Литература, на которую нет ссылок в тексте, в список использованной литературы не включается.

В конце контрольной работы оставляется чистый лист для рецензии.

Работа должна быть подписана автором с указанием даты её отправки в университет.

Работа должна быть представлена в деканат не позже указанного деканатом срока.

1. МЕТОДЫ АДРЕСАЦИИ. КОМАНДЫ ПЕРЕСЫЛКИ ДАННЫХ

1.1. Методы адресации

Микроконтроллеры семейства MC68HC11 выполняют обработку 8- и 16-разрядных операндов и реализуют набор из 108 команд. Они содержат два 8-разрядных аккумулятора A и B, которые при выполнении ряда команд используются как 16-разрядный регистр D, два 16-разрядных индексных регистра X и Y, регистр условий CCR, 16-разрядные регистр-указатель стека SP и программный счетчик PC.

Регистр CCR (табл. 1.1) содержит значения признаков переноса C, переполнения V, нулевого результата Z, знака N, запрещения прерывания I, переноса между тетрадами H.

Таблица 1.1

Формат содержимого регистра условий CCR

Биты	7	6	5	4	3	2	1	0
Признаки	S	X	H	I	N	Z	V	C

Микроконтроллеры семейства MC68HC11 имеют следующие типы адресации: неявная, непосредственная, прямая, расширенная, индексная и относительная.

Рассмотрим каждый из видов адресации подробнее.

Неявная адресация используется в том случае, когда в качестве операндов используются либо регистры (например, COMA, CLI), либо фиксированная ячейка памяти (SWI). Другими словами можно сказать, что неявная адресация не требует отдельного битового поля для указания операнда. В большинстве случаев такие команды однобайтные.

```
43          COMA
53          COMB
```

Исключение составляют команды, взаимодействующие с регистром Y:

```
18  35      TYS
18  3A      ABY
```

В случае использования ***непосредственной адресации*** операнд (или один из операндов) включен непосредственно в код команды. Длина таких команд может составлять от 2 до 4 байтов. При записи команд, использующих непосредственную адресацию, операнд предваряется символом «решетка» ('#').

```
86    03          LDAA    #3
CE    80  00      LDX     #32768
18    8C  56  78  CPY     #$5678
```

Прямая адресация используется для доступа к данным, расположенным в первых 256 байтах памяти. При этом младший байт адреса операнда расположен непосредственно за кодом команды. Применение этой группы команд позволяет сократить объем программы, а также время выполнения на выборке операнда из памяти.

```
96    3F          LDAA
63    DA  FF      GRAB    $FF
```

Использование ***расширенной адресации*** позволяет осуществить доступ к любой ячейке памяти в пределах адресного пространства контроллера. При этом 2 байта, следующие непосредственно за кодом команды, представляют собой абсолютный адрес операнда.

```
B6    40  00      LDAA    $4000
7E    78  12      JMP     $7812
```

Как правило, ассемблер автоматически выбирает наиболее оптимальный из двух вышеописанных методов адресации.

Для доступа к массивам данных удобно использовать ***индексную адресацию***. В микроконтроллерах семейства MC68HC11 используется так *называемая индексная адресация с 8-разрядным смещением*. При этом в индексный регистр X или Y заносится 16-разрядный адрес, а следующий за кодом команды байт содержит 8-разрядное смещение. Абсолютный адрес при этом вычисляется простым суммированием содержимого индексного регистра с байтом смещения.

```
A6    07          LDAA    $07,X
18    AD  00      JSR     0,Y
```

Команды работы со стеком также принято относить к командам с индексной адресацией.

```
32                PULA
37                PSHB
```

Эти команды используют *индексную адресацию без смещения*.

Относительная адресация используется в командах передачи управления. При этом абсолютный адрес перехода вычисляется путем сложения содержимого программного счетчика со смещением, представляющим собой 8-разрядное знаковое число. Таким образом, используя относительную адреса-

цию можно осуществить переход на адрес, лежащий в пределах от –128 до +127, относительно адреса следующего за командой перехода.

```
8D    00          BSR     *+$2
24    FF          BCC     *-125
```

Заметим, что для наглядности здесь использован символ «звездочка» ('*'), который заменяется ассемблером на адрес текущей команды. Программы, использующие только относительную и неявную адресацию, принято называть *позиционно-независимыми программами*. Это объясняется тем, что при перемещении кода из одной области памяти в другую работоспособность программы сохраняется.

1.2. Команды пересылки данных

Простейшими командами являются команды пересылки данных. Список этих команд приведен в табл. 1.2. Рассмотрим каждую из команд подробнее на простых примерах.

Таблица 1.2

Команды пересылки данных

TSTA TSTB TST*	CLRA CLRB CLR*	TAB TBA TAP	PSHA PULA PSHB
LDAA** LDAB** LDD** LDX** LDY** LDS**	STAA*** STAB*** STD*** STX*** STY*** STS***	TPA TSX TXS TSY TYS	PULB PSHX PULX PSHY PULY
		XGDX XGDY	

Примечания:
* – команды, использующие расширенную и индексную адресацию;
** – команды, использующие непосредственную, прямую, расширенную и индексную адресацию;
*** – команды, использующие прямую, расширенную и индексную адресацию.

TSTA, TSTB, TST (opr)

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	0	0

LDAA (opr), LDAB (opr),
LDD (opr), LDS (opr),
LDX (opr), LDY (opr)

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	0	-

Команды TSTA, TSTB и TST служат для установки регистра статуса в соответствии с содержимым регистра A, B или ячейки памяти соответственно. Далее результат может быть использован в командах условного перехода. Занесите в регистр A значение $00 и выполните в пошаговом режиме команду TSTA. Теперь посмотрите на содержимое регистра статуса: должен быть установлен флаг нуля (Z) и сброшены флаг отрицательного результата (M),

переноса (C) и переполнения (V). Проведите подобный опыт при других значениях регистра A, обращая внимание на различное состояние регистра статуса.

Рассмотрим команды загрузки в регистр содержимого ячейки памяти:

```
org   $8000
ldab  $56        ;   загрузить в регистр B содержимое ячейки $56,
                 ;   используя прямую  адресацию
ldy   $c800      ;   загрузить в регистр Y данные, расположенные по адресу
                 ;   $c800 (предыдущую команду)
ldx   #$1f00     ;   установить регистр X
ldaa  $03,x      ;   считать информацию
```

CLRA, CLRB, CLR (opr)

S	X	H	I	N	Z	V	C
-	-	-	-	0	1	0	0

Работа команд очистки регистров A и B и ячейки памяти может быть проиллюстрирована на примере следующей простой программы:

```
org   $8000
clrb             ;   очистить регистр B
ldx   #$1f00     ;   установить регистр X
clr   $04,x      ;   очистить
```

STAA (opr), STAB (opr), STD (opr), STS (opr), STX (opr), STY (opr)

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	0	-

Теперь рассмотрим работу команд модификации ячеек памяти. Для этого введем следующую программу:

```
org   $8000
ldd   #$AA55     ;   установить в регистре D значение $AA55
ldx   #$1f00     ;   установить регистр X
clr   $04,x      ;   очистить
staa  $04,x      ;   записать
stab  $04,x      ;   записать
ldaa  $03,x      ;   считать информацию
staa  $04,x      ;   записать
```

В результате выполнения команды TAB значение аккумулятора A будет присвоено аккумулятору B. Команда TBA имеет противоположный эффект. Следует отметить, что регистр статуса принимает состояние, подобное выполнению команд STAA, STAB.

Команда TPA осуществляет перенос содержимого регистра CCR в аккумулятор A. Это удобно, если после выполнения какой-либо подпрограммы необходимо сохранить состояние регистра статуса (см. также TAP).

TAB, TBA

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	0	-

TPA, TSX, TSY, TXS, TYS, XGDX, XGDY

S	X	H	I	N	Z	V	C
-	-	-	-	-	-	-	-

Группа команд работы с регистром стека имеет одну особенность: при переносе числа из индексного регистра регистр стека получает на единицу меньшее значение, при обратной пересылке происходит увеличение индексного регистра. Рассмотрим эти команды подробнее:

```
org    $8000
ldx    #$220        ; занести в регистр X адрес $220
xgdx                ; обмен содержимого регистров X и D
clrb                ; очистить младший байт регистра D
xgdx                ; X = $200
txs                 ; SP = $1ff
tsy                 ; Y = $200
```

Обмен содержимого индексного регистра и регистра D, как правило, используется при арифметических операциях (так как арифметические команды работы с регистром D более развиты) или в случае необходимости 8-разрядного доступа к содержимому индексного регистра, что может быть полезно, например, для организации кольцевого буфера.

TAP

S	X	H	I	N	Z	V	C
?	?*	?	?	?	?	?	?

* - значение может быть изменено только из 1 в 0.

Команда TAP осуществляет перенос значения регистра A в соответствующие биты регистра статуса CCR. При этом содержимое регистра A остается неизменным. Флаг X, служащий для маскирования прерывания XIRQ, в результате выполнения этой команды может быть сброшен, но он не может быть установлен, если до выполнения команды флаг был сброшен.

```
org    $8000
ldaa   #$47         ;  занести в регистр A новое содержимое регистра статуса
tap                 ;  установить новое значение регистра статуса: заметьте,
                    ;  что флаг X не будет установлен
```

Команды работы со стеком, как правило, используются в подпрограммах для того, чтобы сохранить значение одного или более регистров.

Алгоритм работы команд PSH таков:

PSHA, PSHB, PSHX, PSHY, PULA, PULB, PULX, PULY

S	X	H	I	N	Z	V	C
-	-	-	-	-	-	-	-

1) в ячейку памяти, на которую указывает регистр SP, записывается (младший) байт регистра-операнда;

2) значение регистра SP уменьшается на 1, указывая на следующую свободную ячейку в области стека;

3) в случае 2-байтного операнда последовательность (1–2) повторяется со старшим байтом операнда.

Команды группы PUL выполняют данную последовательность в обратном порядке, увеличивая значение регистра SP.

Следующая программа демонстрирует, каким образом можно сохранить неизменными все внутренние регистры ОЭВМ (рекомендуется также обратить внимание на содержимое стека):

```
org    $8000
psha          ; последовательно сохраняем регистры в стеке: A, B, X, Y, CCR
pshb
pshx
pshy
tpa
psha
ldaa   #$20   ; выполняем какие-либо действия, в результате которых изменяется
ldx    $12    ; содержимое регистров
ldy    $1f03
clrb
xgdy
pula          ;  восстанавливаем регистры:  CCR,  Y, X,  B, A
tap
puly
pulx
pulb
pula
```

1.3. Контрольные вопросы

1. Какие методы адресации вам известны? Дайте краткую характеристику каждого из них.
2. Какие методы адресации могут быть использованы в командах LDAA, STAA?
3. На какие флаги влияет выполнение команды TSTA?
4. Как формируется абсолютный адрес перехода в командах, использующих индексную адресацию?
5. Укажите на неточности (если они есть) в написании команд:

```
ldaa    #20
staa    #$50
ldab    #$500
tax
xgdy
```

6. Какие из изученных в данном разделе команд влияют на содержимое регистра SP?

7. Что такое позиционно-независимая программа?

8. Какие методы адресации используют приведенные ниже команды:

```
ldaa    #20
staa    $20
psha
coma
pulb
```

9. Каково значение регистров X и D в результате выполнения программы:

```
ldaa    #30
ldx     #$4020
tab
psha
psha
xgdx
pulx
```

10. Какие особенности имеет команда TAP?

11. Какое применение находит команда XGDX?

12. Каково значение регистра SP в результате выполнения фрагмента программы:

```
ldx     #$200
txs
pshx
pula
```

13. Как формируется абсолютный адрес перехода в командах, использующих относительную адресацию?

14. Какая логическая ошибка допущена при написании данного фрагмента программы:

```
ldx     #$20
pula
ldaa    0,x
staa    5,x
ldaa    3,x
staa    $22
psha
```

15. Каково значение регистра Y в результате выполнения программы:

```
ldx     #$4644
stx     $20
ldaa    #$20
tab
std     $21
ldy     $20
```

2. АРИФМЕТИЧЕСКИЕ КОМАНДЫ

2.1. Арифметические команды

Список арифметических команд приведен в табл. 2.1.

Таблица 2.1

Арифметические команды

INCA	DECA	NEGA	CMPA*	SUBA*	ADDA*	ADCA*	DAA
INCB INC**	DECB DEC**	NEGB NEG**	CMPB* CPD*	SUBB* SUBD*	ADDB* ADDD*	ADCB*	MUL
INX INY	DEX DEY		CPX* CPY*	SBCA* SBCB*	ABA ABX		FDIV
INS	DES		CBA	SBA	ABY		IDIV

Примечания:
* – команды, использующие непосредственную, прямую, расширенную и индексную адресацию;
** – команды, использующие расширенную и индексную адресацию.

INCA, INCB, INC (opr)
DECA, DECB, DEC (opr)

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	?	-

INS, DES

S	X	H	I	N	Z	V	C
-	-	-	-	-	-	-	-

INX, INY, DEX, DEY

S	X	H	I	N	Z	V	C
-	-	-	-	-	?	-	-

Приведем примеры использования этих команд в порядке увеличения сложности.

Команды инкремента и декремента являются простейшими арифметическими операциями и служат соответственно для увеличения и уменьшения на единицу значения регистра ОЭВМ или ячейки памяти.

В зависимости от типа операнда значение регистра статуса после выполнения команд может принимать различные значения. При работе с 8-разрядным операндом команды инкремента и декремента влияют на флаги отрицательного результата (N), нуля (Z) и переполнения (V). В случае если операндом является указатель стека, значение регистра статуса остается неизменным. При операциях с индексными регистрами команды инкремента и декремента влияют только на флаг нуля (Z).

Как правило, команды INC и DEC используются для организации циклов. Тот факт, что эти команды не изменяют флаг переноса, используется при арифметических операциях над многобайтными числами.

Следующий простой пример иллюстрирует работу этих команд:

```
org    $8000
ldaa   #$10        ;     поместить в регистр А значение $10
inca               ;     увеличить на 1
```

```
tab             ;       поместить в регистр B
decb            ;       уменьшить B на 1
std   $10       ;       сохранить регистры A и B в ячейках $10 и $11
ldx   $10       ;       загрузить в регистр X
inx             ;       инкрементировать регистр X
des             ;       указателя стека
```

NEGA, NEGB,
NEG (opr)

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	?	?

Команда NEG замещает операнд его двоичным дополнением. Другими словами можно сказать, что результатом операции является изменение знака числа, представленного в дополнительном коде.

Продемонстрируем на примере эмуляцию команды INC через NEG и DEC:

```
org   $8000
nega            ; изменить знак числа
deca            ; увеличить на 1
nega            ; изменить знак числа
```

CMPA (opr), CMPB (opr),
CPX (opr), CPY (opr),
CPD (opr), CBA

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	?	?

Команды сравнения используются при сравнении значения регистра со значением ячейки памяти или регистра. Фактически происходит операция вычитания ячейки памяти, указанной в качестве операнда, или регистра B (в случае команды CBA) из соответствующего регистра МК. Команды не оказывают влияния на операнды, изменяется лишь регистр статуса. В дальнейшем результат обычно используется командами перехода.

```
org   $8000
ldaa  #$10      ; инициализация регистров A и B
ldab  #$50
cba             ; сравнение регистров A и B
stab  $01
cmpb $01        ; сравнение содержимого регистра B с ячейкой $01
```

ABA,
ADCA (opr), ADCB (opr),
ADDA (opr), ADDB (opr)

S	X	H	I	N	Z	V	C
-	-	?	-	?	?	?	?

При выполнении команд сложения происходит суммирование содержимого регистра-приемника с непосредственно заданным значением, ячейкой памяти или другим регистром. В командах ADC к результату дополнительно прибавляется значение флага переноса. Результат сложения аккумуляторов командой ABA заносится в регистр A, результат сложения регистра B с индексным регистром – в соответствующий индексный регистр.

При выполнении команд вычитания происходит вычитание из регистра-

приемника второго операнда (в случае SBA происходит вычитание регистра B из регистра A). Команды SBC дополнительно вычитают из регистра-приемника значение флага переноса.

ADDD (opr), SUBD (opr),
SBA,
SBCA (opr), SBCB (opr),
SUBA (opr), SUBB (opr)

ABX, ABY

S	X	H	I	N	Z	V	C
-	-	-	-	-	-	-	-

Команды, учитывающие флаг переноса, как правило, используются при операциях над многобайтными числами. Ниже приводится пример сложения и вычитания двух 3-байтных чисел, расположенных в ячейках $0 ... $2 и $3 ... $5 соответственно.

```
org   $8000
ldx   #01
ldd   0,x          ; сложение младших 2-х байтов
addd  3,x
std   0,x
dex                ; переход к 3-му байту
ldaa  0,x
adca  3,x          ; сложение с учетом переноса
staa  0,x          ; записываем результат
ldx   #$2
ldaa  0,x          ; вычитание с использованием SBA
ldab  3,x
sba
dex                ; переход к следующему байту
ldaa  0,x
sbca  3,x
staa  0,x
dex                ; переход к последнему байту
ldaa  0,x
sbca  3,x
staa  0,x          ; результат получен, записываем последний байт
```

Двоично-десятичная коррекция после сложения командами ABA, ADDA и ADCA обеспечивает суммирование двух чисел, представленных в двоично-десятичном формате. При этом флаг переноса используется в качестве старшего бита, обеспечивая получение корректного двоично-десятичного значения.

DAA

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	?*	?

*-значение не определено

Фактически команда DAA после команд сложения действует следующим образом:

1) если содержимое младшей тетрады аккумулятора больше 9 или флаг полупереноса H установлен в «1», то к аккумулятору добавляется число 6;

2) если содержимое старшей тетрады аккумулятора стало после этого более 9 или установлен флаг переноса, то число 6 добавляется и к старшей тетраде аккумулятора.

```
org    $8000
ldaa   #$99          ; 99 в двоично-десятичном коде
ldab   #$20
aba                  ; результат равен B9
daa                  ; коррекция до двоично-десятичного значения: C = 1, A = $19
tab
ldaa   #0            ; использование clra не допустимо, т.к. будет сброшен флаг
                     ; переноса
adca   #0            ; D = $0119
```

MUL

S	X	H	I	N	Z	V	C
-	-	-	-	-	-	-	?

Команда умножения производит беззнаковое умножение двух чисел, представленных в 8-разрядных аккумуляторах. Результат помещается в 16-разрядный аккумулятор D. Флаг переноса при этом устанавливается таким образом, что при выполнении команды ADCA #0 происходит округление старшего байта.

```
org    $8000
ldaa   #$10
ldab   #$68
mul                  ;       $10 * $68 = $0680
adca   #0            ;       A = $7
```

IDIV

S	X	H	I	N	Z	V	C
-	-	-	-	-	?	0	?

Команда IDIV производит целочисленное деление аккумулятора D на индексный регистр X. После выполнения в регистр X заносится частное, а в регистр D – остаток от деления. При выполнении команды IDIV делимое обычно больше делителя.

FDIV

S	X	H	I	N	Z	V	C
-	-	-	-	-	?	?	?

Команда FDFV производит операцию дробного деления тех же аргументов. Фактически FDIV может быть представлен как умножение регистра D на 2^{16} с последующим выполнением команды IDIV, поэтому при выполнении этой команды делитель обычно больше делимого. Эти две команды очень редко используются на практике.

```
org    $8000
ldd    #1020         ;       D = 1020 ($3fc)
ldx    #512          ;       X = 512 ($200)
idiv                 ;       D = 1 ($1), X = 508 ($1fc)
fdiv                 ;       D = 129 ($81), X = 4 ($4)
```

2.2. Контрольные вопросы

1. Какие команды сложения вы знаете?
2. Какие методы адресации используют команды ABA, ADDA, ABY?
3. Какие команды вычитания вам известны?
4. Каким образом используется бит переноса в операции вычитания?
5. Над какими операндами могут выполняться команды INC, DEC?
6. Объясните отличие в выполнении команд ADD и ADC.
7. Где располагаются результаты команды FDIV и что они собой представляют?
8. Что может служить операндом команды ADCA?
9. Какой флаг устанавливается, если результат операции сложения превышает $FF?
10. Объясните, по какому принципу устанавливаются флаги переноса, нуля и переполнения в регистре статуса CCR при выполнении арифметических команд сложения и вычитания.
11. Объясните логику работы команд сложения/вычитания с учетом переноса/заёма при обработке многобайтовых чисел.
12. Объясните логику работы команды DAA.
13. Чем отличаются команды FDIV и IDIV?

3. ЛОГИЧЕСКИЕ КОМАНДЫ. КОМАНДЫ РАБОТЫ С БИТОВЫМИ ПОЛЯМИ. КОМАНДЫ СДВИГОВ

3.1. Логические команды

Логические команды включают в себя действия булевой алгебры над аккумулятором или в случае команды COM над ячейкой памяти, заданной при помощи расширенной или индексной адресации. Команды BITA и BITB по принципу работы схожи с командами ANDA и ANDB, но не изменяют содержимого аккумулятора (сравните CMPA и SUBA). Команды приведены в табл. 3.1.

Таблица 3.1

Логические команды

COMA COMB COM	ANDA ANDB	BITA BITB	ORAA ORAB	EORA EORB

COMA, COMB,
COM (opr)

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	0	1

ANDA (opr), ANDB (opr),
BITA (opr), BITB (opr),
ORAA (opr), ORAB (opr),
EORA (opr), EORB (opr)

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	0	–

Обычно логические команды используются для выборочной установки, обнуления, дополнения и тестирования бит, что часто используется при работе с периферийными устройствами. Следующий пример показывает, каким образом можно перенести содержимое старшего и младшего бит порта C в старший и младший биты порта B. При этом младший бит порта C переносится с инверсией. (Программа написана таким образом, чтобы задействовать максимальное число логических команд, но не оптимизирована на скорость выполнения.)

```
org    $8000
ldaa   $1f03          ;    считать состояние порта C
ldab   $1f04          ;    считать состояние порта B
andb   #%01111110     ;    выделить неизменную часть
oraa   #%01111110     ;    установить все неиспользуемые биты  в «1»
coma                  ;    инвертировать значение аккумулятора
eora   #%10000000     ;    инвертировать значение старшего бита
aba                   ;    совместить результат (заметьте, что мы
                      ;    заблаговременно маскировали неиспользуемые
                      ;    биты, чтобы сейчас совместить два числа простым
                      ;    суммированием)
staa   $1f04          ;    вывести результат
```

3.2. Команды работы с битовыми полями

Команды работы с битовыми полями позволяют изменять указанные биты приемника (ячейки памяти или регистра статуса CCR), оставляя незадействованные биты нетронутыми. Список команд приведен в табл. 3.2.

Таблица 3.2

Команды работы с битовыми полями

SEC	SEI	SEV	BSET*
CLC	CLI	CLV	BCLR*

Примечание.
* – команды, использующие прямую или индексную адресацию в качестве первого параметра и непосредственную – в качестве второго.

Команды, представленные в первых трех столбцах таблицы, устанавливают (SE?) или сбрасывают (CL?) отдельные флаги в регистре статуса, на которые указывает третья буква в мнемонике команды (C – флаг переноса, I – маскирование прерываний, V – флаг переполнения).

Приведем простой пример, показывающий один из способов занесения числа 1 в аккумулятор:

```
org   $8000
clra              ;  очистить аккумулятор
sec               ; установить флаг переноса
adca  #0          ;  прибавить его к аккумулятору
```

В реализации отладчика имеется одна особенность – если трассируемая команда запрещает прерывания, то выполнение программы будет продолжаться до тех пор, пока либо прерывания не будут разрешены, либо не произойдет прерывания по неправильному коду команды. В частности это относится к командам SEI и SWI (эта команда будет рассмотрена в следующей главе). Также следует отметить, что если выполнение программы затянется, то отладчик выдаст сообщение об истечении времени ожидания ответа.

Почти в каждой программе требуется возможность манипуляции отдельными битами ячейки памяти. Так, блок регистров представляет собой по большей части битовые поля.

BCLR (opr), BSET (opr)

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	0	-

Команды BCLR и BSET в качестве первого операнда получают ячейку памяти, в которой соответственно сбрасываются или устанавливаются биты, указанные в маске, заданной непосредственно вторым параметром.

Приведем пример, демонстрирующий работу этих команд:

```
org   $8000
ldx   #$1f00          ;       настроить регистр X
```

```
bset   4,x,#$AA      ;   установить в «1» через один бит,  оставив
                     ;   незадействованные биты в прежнем состоянии
bclr   4,x,#$55      ;   установить в «0» остальные биты
```

3.3. Команды сдвигов

Список команд сдвигов представлен в табл. 3.3. Эти команды позволяют адресоваться к аккумулятору или ячейке памяти.

Команды сдвигов обычно подразделяют на три группы:

- арифметические сдвиги;
- логические сдвиги;
- циклические сдвиги.

Таблица 3.3

Команды сдвигов

ASIA/ LSLA ASLB/ LSLB ASLD/ LSLD ASL*/ LSL*	ASRA ASRB ASR*	LSRA LSRB LSRD LSR*	ROLA ROLB ROL*	RORA RORB ROR*

Примечание.

* – команды, использующие расширенную или индексную адресацию.

ASLA, ASLB, ASL (opr),
ASLD, ASRA, ASRB,
ASR (opr)

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	?	?

При арифметическом сдвиге происходит сохранение знака первоначального операнда при выполнении сдвига. При выполнении команды ASR происходит расширение знакового разряда. Это позволяет использовать команду для деления знакового числа на 2^N. Однако для нечетных чисел деление не всегда является корректным (разница в результате может составлять 1). При выполнении команды арифметического сдвига влево всякий раз при смене знакового бита устанавливается флаг V, а освободившиеся разряды заполняются 0. Таким образом, становится возможным при помощи команд ASR производить знаковое умножение числа на 2^N.

Флаг переноса устанавливается в соответствии с отбрасываемым битом. Приведем пример, демонстрирующий работу этих команд:

```
org    $8000
ldaa   #%00101001
ldx    #$1f00
staa   4,x       ;   вывести содержимое аккумулятора
asl    4,x       ;   умножить на два
asl    4,x       ;   еще раз умножить на два  (происходит переполнение)
asr    4,x       ;   разделить на два
asr    4,x       ;   разделить на два
```

LSLA, LSLB, LSL (opr),
LSLD, LSRA, LSRB,
LSR (opr), LSRD

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	?	?

Логические сдвиги производят сдвиг содержимого аккумулятора или ячейки памяти влево (LSL) или вправо (LSR). При этом освободившиеся разряды всегда заполняются нулями. Команды групп ASL и LSL выполняют в точности одинаковые действия и имеют одинаковые коды операций, поэтому покажем лишь отличие команд ASR от команд LSR:

```
org   $8000
ldaa  #%10101010
staa  $1f04       ; установить в «1» через один бит
asr   $1f04       ; арифметический сдвиг вправо:
asr   $1f04       ; старший бит сохраняется
lsr   $1f04       ; логический сдвиг:
lsr   $1f04       ; старший бит заполняется «0»
```

ROLA,ROLB, ROL (opr),
RORA,RORB, ROR (opr)

S	X	H	I	N	Z	V	C
-	-	-	-	?	?	?	?

Команды циклического сдвига позволяют осуществить операцию логического сдвига над многобайтными числами. Отличие этих операций от операций логического сдвига состоит в том, что освободившийся разряд заполняется не нулем, а состоянием флага переноса C. Рассмотрим пример использования этих команд для сдвига 4-байтного числа, расположенного в ячейках 0 ... 3, влево:

```
org   $8000
ldx   #0
lsl   3,x
rol   2,x
rol   1,x
rol   0,x
```

3.4. Контрольные вопросы

1. Каков результат выполнения программы

```
sec
clra
adda  #0
```

2. Какие методы адресации применимы к командам циклического сдвига?
3. Расскажите о командах работы с битами регистра CCR.
4. Какие особенности работы с отладчиком следует учитывать при отладке программ, запрещающих прерывания?
5. В чем разность команд ASR и LSR?
6. Можно ли использовать команду ROLA вместо команд ASLA, LSLA?

7. Какие логические команды вы знаете?

8. Каким образом реализуется команда ASRD (сдвиг регистра D на 1 байт вправо)?

9. На какие группы можно подразделить команды сдвигов?

10. Дайте определение команд логического сдвига.

11. Чем отличается команда COM от команды NEG?

12. Каким образом можно сэмулировать команду COM, пользуясь командами, изученными в данном разделе?

13. Какие параметры имеет команда BSET?

14. Чем отличается команда ORAA от команды EORA?

4. КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ. СПЕЦИАЛЬНЫЕ КОМАНДЫ

4.1. Команды передачи управления

Команды передачи управления можно разделить на 4 группы:

- команды безусловного перехода (JMP, BRA, BRN, NOP);
- команды работы с подпрограммами (JSR, BSR, RTS);
- команды условного перехода (BEQ, BNE, BMI, BPL, BCS/BLO, BCC/BHS, BVS, BVC, BGT, BGE, BLT, BLE, BLS, BHI, BRSET, BRCLR);
- команды работы с прерываниями (SWI, RTI).

Список команд передачи управления представлен в табл. 4.1. Все команды передачи управления не оказывают влияния на состояние регистра статуса.

Таблица 4.1

Команды передачи управления

JMP*	BEQ**	BCS/BLO**	BOT**	BLS**	JSR*****	SWI***
BRA**	BNE**	BCC/BHS**	BGE**	BHI**	BSR**	RTI***
BRN**	BMI**	BVS**	BLT**	BRSET****	RTS***	
NOP***	BPL**	BVC**	BLE**	BRCLR****		

Примечания:

* – команды, использующие расширенную и индексную адресацию;

** – команды, использующие относительную адресацию;

*** – команды, использующие неявную адресацию;

**** – команды, использующие смешанную адресацию: первый операнд использует либо прямую, либо индексную адресацию, второй – относительную;

***** – команды, использующие прямую, расширенную и индексную адресацию.

Команды ***безусловного перехода*** служат для передачи управления другому участку программы независимо от состояния регистра статуса микроконтроллера и содержимого ячеек памяти. Рассмотрим работу этих команд более подробно на следующем примере:

JMP, BRA, BRN, NOP

S	X	H	I	N	Z	V	C
-	-	-	-	-	-	-	-

```
org     $8000
        ldab    #$02        ;   выбор варианта ветвления программы
        ldx     #ways       ;   занести в регистр X адрес таблицы переходов
p0      ldy     0,x         ;   считать значение в регистр Y
        jmp     0,y         ;   вызвать подпрограмму по адресу, находящемуся в
                            ;   регистре Y
p1      nop                 ;   задержка в 2 такта
        inx                 ;   увеличить регистр X на 2 для выборки
                            ;   следующего адреса из таблицы переходов
```

```
        bra   p0          ;     перейти на метку p0
p2      bra   p1          ;     перейти на метку p1
p3      brn   *           ;     задержка в 3 такта
ways    fdb   p1,p2,p3
```

Выполните программу в пошаговом режиме. Обратите внимание, что команда «jmp 0,x» выполнится два раза, при этом в первом случае переход будет осуществлен на метку p2, а во втором – p3.

JSR, BSR, RTS

S	X	H	I	N	Z	V	C
-	-	-	-	-	-	-	-

Команды работы с подпрограммами позволяют выделять часто используемую последовательность действий в подпрограмму. При переходе к подпрограмме (JSR, BSR) в стеке сохраняется адрес следующей за текущей команды и регистр PC изменяется по правилам команд безусловного перехода. При выходе из подпрограммы по команде RTS происходит выборка из стека адреса возврата. Работу этих команд можно проследить на примере программы, размещающей адрес своей второй команды в ASCII формате в ячейках $0 ... 3:

```
            org   $8000
            bsr   p1          ;     переход на следующую команду
p1          pulx              ;     получить в регистр X адрес p1 ($8002)
            xgdx              ;     расписать адрес в формате ASCII
            ldy    #0         ;     в ячейках $0 ... $3
            bsr   bin2ascii   ;     вызвать подпрограмму, сохраняющую
                              ;     регистр A в формате ASCII в ячейках
                              ;     (y), (y+1)
            tba               ;     переписать регистр B в регистр A
            iny               ;     и расписать его в ячейках $2, $3
            iny
            bsr   bin2ascii
            bra   *
bin2ascii   pshb              ;     сохранить в стеке регистр B
            tab               ;     скопировать в него регистр A
            lsra              ;     выделить в A старшую тетраду
            lsra
            lsra
            lsra
            bsr   hex2ascii   ;     преобразовать число 0 ... f в ASCII код
            staa  0,y
            tba               ;     повторить для младшей тетрады
            anda  #$f
            bsr   hex2ascii
            staa  l,y
            pulb
```

```
            rts
hex2ascii   adda  #0          ;     преобразование числа из диапазона
            daa
            adda  #$f0
            adca  #$40
            rts
```

Команды условного перехода служат для передачи управления в зависимости от состояния регистра CCR ОЭВМ или значения ячейки памяти (BRSET и BRCLR).

Иногда команды условного перехода, выполняющие передачу управления, в зависимости от состояния регистра статуса подразделяют на три группы:

- знаковые;
- беззнаковые;
- простые (табл. 4.2).

Таблица 4.2

Команды условного перехода

Условие	Логическая функция	Мнемоника	Противоположное действие		Тип
r>m	Z+(N⊕V)=0	BGT	r≤m	BLE	Знаковый
r≥m	N⊕V=0	BGE	r<m	BLT	Знаковый
r=m	Z=l	BEQ	r≠m	BNE	Знаковый
r≤m	Z+(N⊕V)=1	BLE	r>m	BGT	Знаковый
r<m	N⊕V=1	BLT	r≥m	BGE	Знаковый
r>m	C+Z=0	BHI	r≤m	BLS	Беззнак.
r≥m	C=0	BCC/BHS	r<m	BCS/BLO	Беззнак.
r=m	Z=l	BEQ	r≠m	BNE	Беззнак.
r≤m	C+Z=1	BLS	r>m	BHI	Беззнак.
r<m	C=l	BCS/BLO	r≥m	BCC/BHS	Беззнак.
перенос	C=l	BCS/BLO	нет пер.	BCC/BHS	Простой
отрицат.	N=1	BMI	полож.	BPL	Простой
переп.	V=l	BVS	нет пер.	BVC	Простой
r=0	Z=l	BEQ	r≠0	BNE	Простой

Покажем один из способов организации циклов с помощью команд условного перехода на примере сложения двух 4-байтных чисел:

```
            org   $8000
            ldx   #$3         ;     конечный адрес первого числа
            ldy   #$7         ;     конечный адрес второго числа
            ldab  #$4         ;     размер числа
            clc               ;     сброс флага переноса
loop        ldaa  0,x         ;     сложить два байта
            adca  0,y
            staa  0,x
```

```
        dex             ;       перейти к следующему байту
        dey
        decb            ;       уменьшить число обрабатываемых байтов
        bne   loop      ;       на 1 и повторить, если не 0
```

Другой пример показывает использование команд переходов при проверке правильности даты, записанной в виде BCD числа в ячейках $0...$3 в формате ДДММГГГГ (т.е. в ячейке $0 хранится день, в ячейке $1 – месяц, а в ячейках $2 и $3 – столетие и год в столетии соответственно):

```
        org   $8000
        ldx   #0        ;       установить указатель на начало даты
        ldd   0,x       ;       загрузить в A и B день и месяц
        tstb            ;       если день или месяц равен 0 или
        beq   invalid   ;       месяц больше 13, то дата не корректна
        tsta
        beq   invalid
        cmpb  #$13
        bhs   invalid
        cmpb  #$2       ;       если это не февраль, то переход
        bne   lab2      ;       на выборку числа дней из таблицы
        ldab  3,x       ;       проверка високосного года
        bsr   bcd2bin
        andb  $03       ;       получение остатка от деления на 4
        bne   lab2      ;       год невисокосный, если не делится нацело
                        ;       на 4
        ldab  #$29
        bra   lab1
lab2    bsr   bcd2bin   ;       преобразование месяца в двоичный код
        decb
        ldy   #dpm      ;       настроить указатель на список дней в
                        ;       месяце
        aby
        ldab  0,y       ;       занести в B число дней в месяце
lab1    cba             ;       если число дней больше вычисленного,
        bhi   invalid   ;       то дата не корректна
valid   clc             ;       год может быть любым, поэтому проверка
        bra   done      ;       не производится и возвращается признак
                        ;       успешной проверки
invalid sec             ;       установка флага ошибки
done    bra   *
bcd2bin psha            ;       преобразование числа в формате BCD,
        tba             ;       заданного в регистре B в двоичный
        anda  #$0F      ;       формат
```

```
            lsrb            ;       выделение старшей тетрады
            lsrb
            lsrb
            lsrb
            lslb            ;       умножение на 10 и сложение
            aba             ;       с регистром А
            lslb
            lslb
            aba
            tab
            pula
            rts
dpm         fcb     $31,$28,$31,$30,$31,$30,$31,$31,$30,$31,$30,$31
```

BRSET (opr),
BRCLR (opr)

S	X	H	I	N	Z	V	C
-	-	-	-	-	-	-	-

Команды условного перехода, выполняющие передачу управления в зависимости от значения ячейки памяти, используются в циклах ожидания изменения какого-либо регистра управления, опросе внешних линий данных или работе с другими битовыми данными. Простой пример показывает, каким образом можно обеспечить отображение информации:

```
            org     $8000
            ldx     #$1f00      ;       установить регистр X на базовый адрес
            brset   3,x,#$01,p1 ;       если младший бит установлен, тогда
                                ;       перейти к программе ожидания сброса
p0          bsr     display     ;       отобразить информацию
            brclr   3,x,#$01,*  ;       ждать установки бита
                                ;       * – адрес текущей команды
p1          bsr     display
            brset   3,x,#$01,*  ;       ждать сброса бита
            bra     p0          ;       следующий цикл
display     ldaa    3,x         ;       отобразить состояние
            staa    4,x
            rts
```

Команды работы с прерываниями предназначены для входа или выхода из прерывания.

Иногда бывает необходимо выполнить программную реализацию прерывания. Конечно, это можно реализовать через подпрограмму, в начале которой выполняется сохранение регистров в стеке, однако это снижает эффективность работы программы в целом. Для выполнения этой задачи служит команда генерации программного прерывания SWI.

Она выполняет последовательное сохранение в стеке регистров PC+1, Y, X,

SWI

S	X	H	I	N	Z	V	C
-	-	-	1	-	-	-	-

RTI

S	X	H	I	N	Z	V	C
?	?*	?	?	?	?	?	?

* - значение может быть изменено только из 1 в 0.

D, CCR, запрещает маскируемые прерывания (устанавливает бит I в регистре CCR) и передает управление на подпрограмму, адрес которой находится в таблице векторов прерывания. Адрес обработчика прерывания SWI должен быть расположен по адресу $fff6, если работа происходит в нормальном режиме работы, или по адресу $bff6, если в специальном. В режиме bootstrap по адресам $bf40 ... $bfff находится bootstrap ПЗУ, в котором векторы прерываний указывают на ячейки памяти внутреннего ОЗУ. Таким образом, в исследуемых в данном разделе экспериментах для задания окончательного адреса перехода будем использовать адреса $f4 ... $f6, в которых будет расположена команда безусловного перехода JMP.

Для возврата из прерывания используется команда RTI. Она восстанавливает значения регистров из запомненных в стеке, тем самым осуществляется возврат к прерванной программе с сохранением состояния регистров.

Небольшая программа демонстрирует работу этих двух команд:

```
            org   $00f4       ;     установка вектора обработчика
            jmp   ih          ;     прерывания SWI
            org   $8000
gen_int     ldaa  #$55
            swi               ;     вызов прерывания
            coma
            swi
            coma
            swi
            coma
            swi
done        bra   *
ih          ldx   #$1f04      ;     обработчик прерывания осуществляет
            staa  0,x         ;     отображение числа из регистра A
            ldy   #$500       ;     и задержку ≈10 мс
delay       dey
            bne   delay
            rti
```

Попробуйте выполнить эту программу в пошаговом режиме (как было указано в разд. 3, обработчик прерывания будет выполняться за 1 шаг) и с установкой точек останова на метке ih. Посмотрите содержимое области данных выше указателя стека (стекового фрейма) и убедитесь, что все регистры процессора были сохранены.

4.2. Специальные команды

Как отмечалось ранее, к специальным командам относятся команды, переводящие контроллер в режим низкого потребления энергии.

Команда WAI переводит контроллер в режим ожидания первого немаскированного прерывания. При этом сохранение регистров происходит в момент выполнения команды WAI, а не в момент обнаружения прерывания.

Команда STOP выполняет остановку всех внутренних генераторов микроконтроллера и перевод системы в режим минимального энергопотребления. В случае если установлен бит S регистра CCR, то команда STOP выполняется как команда NOP. Восстановление системы из режима минимального энергопотребления может произойти при появлении прерываний от RESET, XIRQ или немаскированного прерывания IRQ. В случае когда установлен бит X регистра CCR, маскирующий прерывание XIRQ и происходит это прерывание, то выполнение программы происходит со следующей за STOP команды. В некоторых масках семейства MC68HC11 была допущена ошибка, приводящая к неправильному интерпретированию кода команды в некоторых особых случаях, поэтому фирма Motorola рекомендует перед командой STOP помещать команду NOP, таким образом исключая эту ошибку.

4.3. Контрольные вопросы

1. Каково различие между командами JMP и BRA?
2. Объясните различие между командами WAI и STOP.
3. Каким образом можно реализовать переход к подпрограмме, не используя команд BSR и JSR?
4. Произойдет ли вызов программного прерывания при установленном флаге I в регистре CCR?
5. Какие команды относятся к знаковым командам условного перехода?
6. Какие виды переходов вам известны?
7. Каково назначение команд BLE, BSR, BCS, BRCLR?
8. Сколько и какие операнды используются командами условного перехода по состоянию бита?
9. Реализуйте (примерно) команды BRCLR и BRSET через другие команды.
10. Каков результат выполнения фрагмента программы:

```
        ldaa   #34
        ldab   #$34
        cba
        bmi    p2
        beq    p3
        bra    done
p2      ldaa   #45
        bra    done
p3      ldaa   #$23
done    clrb
```

11. Можно ли выполнить переход, аналогичный переходу по команде BCS, используя команды BNE и BLE?

12. Каким образом можно осуществить корректный выход из подпрограммы, не используя команду RTS?

13. Какие команды относятся к беззнаковым командам условного перехода?

14. Для какой цели используются команды WAI и STOP?

15. Каково назначение команд BLE, RTI, JSR, BEQ?

ЛИТЕРАТУРА

1. Белевич, А. Ю. Микроконтроллеры семейства 68300 фирмы Motorola / А. Ю. Белевич, О. В. Сердюков, Г. Ю. Костин // Chip News. – 1996. – №2. – С. 32–36.

2. Бродин, В. Б. Технология проектирования микропроцессорных контроллеров / В. Б. Бродин // Электроника и компоненты. – 1997. – №1. – С. 8–9, №2. – С. 7–9.

3. Бродин, В. Б. Эмуляторы 8-разрядных ОЭВМ к IBM PC : сб. статей «Библиотека информационной технологии» / В. Б. Бродин, А. В. Калинин; под ред. Г. Р. Громова. – М. : ИнфоАрт, 1991. – Вып. 3. – С. 222–229.

4. Кобахидзе, Ш. Средства разработки и отладки для однокристальных микроконтроллеров / Ш. Кобахидзе, А. Тамазов // Chip News. – 1996. – №2. – С. 37–43.

5. Корнеев, В. Современные микропроцессоры / В. Корнеев. – 3-е изд.– СПб. : БХВ-СПб, 2003. – 448 с.

6. Коффрон, Дж. Технические средства микропроцессорных систем : практический курс / Дж. Коффрон; пер. с англ. – М. : Мир, 1983. – 344 с.

7. Куприянов, М. С. Коммуникационные контроллеры фирмы Motorola / М. С. Куприянов, О. Е. Мартынов, Д. И. Панфилов. – СПб. : БХВ-Петербург, 2001. – 560 с.

8. Левенталь, Л. Введение в микропроцессоры: программное обеспечение, аппаратные средства, программирование / Л. Левенталь; пер. с англ. – М. : Энергоатомиздат, 1983. – 464 с.

9. Морисита, И. Аппаратные средства микроЭВМ / И. Морисита; пер. с япон. – М. : Мир, 1988. – 280 с.

10. Рафикузаман, М. Микропроцессоры и машинное проектирование микропроцессорных систем : в 2 кн. Кн. 1 / М. Рафикузаман; пер. с англ. – М. : Мир, 1988. – 312 с.

11. Токхайм, Р. Микропроцессоры : курс и упражнения / Р. Токхайм; пер. с англ.; под. ред. В. Н. Грасевича. – М. : Энергоатомиздат, 1987. – 336 с.

12. Фрир, Дж. Построение вычислительных средств на базе перспективных микропроцессоров / Дж. Фрир; пер. с англ. – М. : Мир, 1990. – 413 с.

13. Шагурин, И. И. Микропроцессоры и микроконтроллеры фирмы Motorola : справ. пособ. / И. И. Шагурин. – М. : Радио и связь, 1998. – 560 с.

14. Программно-аппаратные комплексы для проектирования и отладки систем на базе микроконтроллеров Motorola. / И. И. Шагурин [и др.] // Chip News. – 1998. – №1. – С. 22–27.

15. Шагурин, И. И. Современные микроконтроллеры и микропроцессоры Motorola : справочник / И. И. Шагурин. – М. : Горячая линия – Телеком, 2004. – 952 с.

Св. план 2008, поз. 7

Учебное издание

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ И ИХ ПРИМЕНЕНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ
для студентов специальности
I-38 02 03 «Техническое обеспечение безопасности»
заочной формы обучения

Составители:

Логин Владимир Михайлович
Цырельчук Игорь Николаевич

Редактор Н. В. Гриневич
Корректор М. В. Тезина
Компьютерная верстка Е. Г. Реут

Подписано в печать 23.10.2007. Формат 60x84 1/16. Бумага офсетная.
Гарнитура «Таймс». Печать ризографическая. Усл. печ. л. 1,98.
Уч.-изд. л. 1,8. Тираж 100 экз. Заказ 562.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0056964 от 01.04.2004. ЛП №02330/0131666 от 30.04.2004.
220013, Минск, П. Бровки, 6