1 МЕТОДЫ АДРЕСАЦИИ. КОМАНДЫ ПЕРЕСЫЛКИ ДАННЫХ

1.1. Методы адресации

Микроконтроллеры семейства MC68HC11 выполняют обработку 8и 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-азрядное 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 адресацию 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

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;

еранда 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.

непосредственная, прямая, расширенная, индексная

2. Какие методы адресации могут быть использованы в командах LDAA, STAA? прямая, расширенная, индексная

Z - нулевой результат
V - переполнение
N - знак
C - перенос

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    тут нельзя использовать непосредственную адресацию
ldab    #$500   переполнение
tax     такой команды нету
xgdy
```

6. Какие из изученных в данном разделе команд влияют на содержимое регистра SP? psh, pul - работа со стеком
txs, tys - пересылка в SP

7. Что такое позиционно-независимая программа?

8. Какие методы адресации используют приведенные ниже команды:

```
ldaa   #20    непосредственная
staa   $20    прямая
psha   индексная
coma   неявная
pulb   индексная
```

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

```
ldaa   #30        [$1E1E  $4020]
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     [0200]
txs              [01FF]
pshx             [01FD]
pula             [01FE]
```

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   [4644]
stx    $20      → [46 44 (20 21)]
ldaa   #$20     [20]
tab             [20]
std    $21      → [4620 (20 21)]
ldy    $20      [Y = $4620]
```

1.4 Задания

1. Напишите программу, заполняющую ячейки $8200…$8205 значением $55, используя индексную адресацию.

2. Перезаписать регистр A в регистр B таким образом, чтобы значение регистра флагов осталось неизменным.

3. Занести $AA и $55 в регистры A и B соответственно. Перенести значение этих регистров в регистр X таким образом, чтобы в регистре X оказалось значение $55AA.

4. Заполнить 10 ячеек стека значением ячеек памяти, начиная с $8000.

5. Произвести обмен регистров X и Y тремя различными способами.

6. Занести в регистр X число $1F0. Используя только рассмотренные в этой лабораторной работе команды, уменьшить это число на 3.

7. Произвести обмен содержимого младшего байта регистра X с регистром A.

8. Изменить порядок следования байтов в регистре X, не используя команду XGDX.

9. Занести значение регистра стека в регистр D.

10. Изменить порядок следования байтов в регистре Y, используя только неявную адресацию.

11. Сохранить текущее значение регистра стека в стеке.

12. Установить регистр флагов в соответствии с содержимым младшего байта регистра SP.

13. Переписать содержимое регистра A в регистры B, X и Y.

14. Сохранить все регистры ОЭВМ в ячейках памяти $8100 ... $8108. При этом содержимое данных ячеек памяти должно соответствовать значению регистров при входе в программу.

В приложении А представлена система команд, а пример программы – в приложении Б.

Примечание. При написании программ в случае необходимости следует предварительно записать значения в ячейки памяти в соответствии с заданием.