

Лабораторная работа №2

Структура команд процессора

Цель работы: изучить структуру команд процессора, научиться составлять машинный код простейших команд.

Теоретические сведения

Машинная команда в памяти представляет собой последовательность байтов (от 1 до 15), в которых хранится информация о том, какую операцию необходимо выполнить процессору, над какими объектами необходимо выполнить операции, куда сохранить результат.

Операндами называют объекты, над которыми процессор выполняет какие-либо действия. Чаще всего это либо регистры процессора или сопроцессора, ячейки оперативной памяти или числа.

Операнды могут иметь следующий тип адресации:

1. *Непосредственная адресация.* Операнд задаётся в коде команды в виде числа, символа.

```
MOV EAX, 100h;      MOV AX, 111010001b
MOV AL, 'a'.
```

2. *Прямая адресация.* Операнд задаётся как абсолютный адрес ячейки памяти, по которой необходимо обратиться.

```
MOV EAX, [00000002h].
```

Обращение к переменным, объявленным в сегменте данных, также осуществляется непосредственно, к примеру, **MOV EAX, var1**. В процессе компиляции переменной **var1** будет сопоставлена ячейка памяти с конкретным адресом.

3. *Регистровая адресация.* Операнд в этом случае является регистром:

```
MOV EAX, ESP;      MOV AL, CL
MOV AX, CX.
```

4. *Базовая адресация* памяти имеет место, если эффективный адрес задаётся с помощью одного регистра:

```
MOV EAX, [EBX].
MOV AX, [EBP].
```

5. *Базово-индексная адресация.* Операнд содержит эффективный адрес, задаваемый в виде суммы содержимого двух регистров.

```
MOV EAX, [ESI+EDI].
```

6. *Базово-индексная адресация со смещением.*

```
MOV EAX, [ESI+EDI+10].
```

7. *Базово-индексная адресация с масштабированием.* Значение одного из регистров в эффективном адресе масштабируется на 2, 4, 8.

```
MOV EAX, [ESI*2+EDI].
MOV EAX, [ESI*4+EDX].
```

8. *Неявная адресация.* В этом случае объект, над которым выполняются действия, в описании команды явно не содержится. Команды

MUL BX; Умножение **AX** на **BX**. Результат в **DX:AX**

DIV CL; Деление **AX** на **CL**. Частное в **AL**, остаток в **AH**.

не явно используют регистры **EAX** и **EDX**.

Команды процессора имеют следующую структуру:

Пре-фикс	Код операции (КОП)						SIB			Смещение	Непосредственный операнд
		d	w	mod	reg/КОП	r/m	scale	index	base		

Код операции определяет вид операции, которую выполняет команда. *Коды операций команд ассемблера можно найти в учебнике В.И. Юрова «Assembler» в приложении, начиная со стр. 511.* Некоторые поля в коде команды могут отсутствовать. Обязательным является лишь поле кода операции. Бит **d** определяет порядок следования операндов. Если **d=0**, то первый операнд указывает на ячейку памяти или регистр, определяемые полем **r/m**. Второй операнд при этом определяется полем **reg**. Если **d=1**, то наоборот.

К примеру, для команды

MOV var1, ECX бит d = 0;

MOV ECX, var1 бит d = 1.

Если **w=0**, то размер данных, с которыми оперирует команда, равен байту. Если **w=1**, то размер данных – 16 или 32 бита. Если размер пересылаемых данных равен слову (16 бит), то к команде добавляется префикс 66h.

Поле **mod** определяет, какой размер в команде имеет поле смещения.

mod	Размер поля смещения	Пример команд
00	отсутствует	MOV [EBX+ESI], DX
01	один байт	MOV [EBX+ESI+2], DX
10	четыре байта (двойное слово)	MOV [EBX+ESI+2000], DX
11	Операндов в памяти нет	MOV EAX, EBX

Поле **reg** содержит информацию об операнде-регистре.

reg	w=0	w=1
000	AL	AX/EAX
001	CL	CX/ECX
010	DL	DX/EDX
011	BL	BX/EBX
100	AH	SP/ESP
101	CH	BP/EBP
110	DH	SI/ESI
111	BH	DI/EDI

Байт **SIB** необходим для кодирования эффективного адреса и присутствует в команде в том случае, если **r/m=100**. В поле **scale** располагается масштабный

множитель для индексного компонента **index**. Значение индексного регистра масштабируется на 1 (scale=00), 2 (scale=01), 4 (scale=10) либо 8 (scale=11). Примерами команд с масштабным множителем могут быть следующие:

```
MOV EAX, [ESI + EDI]           scale=00;
MOV EAX, [ESI*2 + ECX]         scale=01;
MOV EBX, [ESI*4 + EDX]         scale=10;
MOV EDX, [ECX*8 + 1023]        scale=11;
```

В поле индекса **index** содержится информация об индексном регистре аналогично полю **reg**, в поле **base** – о базовом регистре. К примеру, для команды **MOV EDX, [ESI*2+EDI]** индексным регистром будет **ESI**, базовым – **EDI**.

Задания для выполнения к работе

1. Ознакомиться с теоретическим материалом главы 2 учебника В.И. Юрова «Assembler» «Программно-аппаратная архитектура IA-32 процессоров Intel».
2. В соответствии с вариантом задания определить по символьному описанию команд их машинный код (для 5 команд), а также по машинному коду команд определить их символьное описание (для 2 машинных кодов).

Пример выполнения задания

Символьное описание команд на языке Assembler:

```
MOV EBX, 828
ADD EAX, [EBX*2+ESI+1010111011b]
SUB BYTE PTR [ESI], 4
MOV BX, [ECX]
CMP CL, CH
```

Машинные коды команд в 16 системе счисления:

```
BE 12000000
8BF0
```

Команда 1: **MOV EBX, 828**

Команда выполняет пересылку десятичного числа 828 в регистр EBX. Первый операнд имеет регистровую адресацию, второй является непосредственным операндом. Код операции данной команды **MOV КОП=1011**. Размер пересылаемых данных равен 4 байтам, значит **w=1**. Регистру EBX соответствует поле **reg=011**. Число 828 кодируется следующими четырьмя байтами. $828 = 33Ch = 1100111100b$. Байты числа представляются в памяти в обратном порядке, поэтому в коде команды первым будет младший байт $00111100b = 3Ch$, следующий – $00000011b = 03h$. Поля данной команды кодируются в следующей последовательности:

КОП	w	reg	828			
1011	1	011	00111100	00000011	00000000	00000000
BBh			3Ch	03h	00h	00h

Первые три поля **КОП**, **w**, **reg** образуют первый байт: $10111011b = BBh$. Непосредственный операнд кодируется следующими 4 байтами. Проанализировав команду **MOV EBX, 828** можно сделать вывод, что ей соответствует машинный код **BB3C030000h**. Длина команды – 5 байт.

Команда 2: ADD EAX, [EBX*2+ESI+1010111011b]

Команда выполняет сложение двойных слов из регистра EAX и из памяти по адресу DS: [EBX*2+ESI+1010111011b] и запись результата в регистр EAX. Первый операнд имеет регистровую адресацию, второй – базово-индексную со смещением и масштабированием.

Для данной команды **ADD КОП=000000**. **d=1**, т.к. данные пересылаются из поля **r/m** в поле **reg**. Поле **w=1** – пересылка двойного слова. Для кодирования смещения необходимо не менее двух байт, поэтому **mod=10**. Регистру **EAX** соответствует значение **reg=000**. **r/m = 100**, так как эффективный адрес задаётся в байте **SIB**, который добавляется к коду команды. Поля **SIB** имеют значения: **scale=01** (множитель 2), **index=011 (EBX)**, **base=110 (ESI)**. Смещение кодируется 4 байтами. Младший байт смещения 10111011b=BBh, второй – 00000010b=02h.

Поля данной команды кодируются в следующей последовательности:

КОП	d	w	mod	reg	r/m	scale	index	base	10 10111011b
000000	1	1	10	000	100	01	011	110	
03h				84h			5Eh		BB02h

Проанализировав команду **ADD EAX, [EBX+ESI]**, можно сделать вывод, что ей соответствует машинный код **03845EBB02**. Размер команды – 5 байт.

Команда 3: SUB BYTE PTR [ESI], 4

Команда выполняет вычитание десятичного числа 4 из ячейки памяти, адрес которой содержится в регистре **ESI**. Размер непосредственного операнда указан явно (**BYTE PTR**). Данной команде **SUB** соответствует **КОП=10000000/101**. Первый операнд имеет базовую адресацию, второй является непосредственным операндом. **mod=00**, так как поле смещения отсутствует. **r/m=110** – эффективный адрес равен значению в регистре **ESI**. Данная команда кодируется следующим образом:

КОП	mod	КОП	r/m	4
10000000	00	101	110	00000100
80h		2Eh		04h

Таким образом, машинный код данной команды **802E04**. Размер команды – 3 байта.

Команда 4: MOV BX, [ECX]

Команда выполняет пересылку слова из памяти по адресу DS: [ECX] в регистр **BX**. Первый операнд имеет регистровую адресацию, второй – базовую. Размер пересылаемых данных – 2 байта, значит **w=1**. Направление передачи – из памяти в регистр (**d=1**). Регистр **ECX** кодируется полем **r/m=001**, **BX** – полем **reg=011**. К командам, которые оперируют данными размером 2 байта, добавляется префикс 66h. Смещение отсутствует, поэтому **mod=00**. Построим машинный код данной команды:

Префикс	КОП	d	w	mod	reg	r/m
	100010	1	1	00	011	001
66h			8Bh			19h

Машинный код заданной команды: **66:8B19**. Размер команды – 3 байта.

Команда 5: CMP CL, CH

Команда выполняет сравнение 8-битных регистров **CL** и **CH**. Код операции данной команды **КОП**=001110. **w**=0, т.к. размер операндов – один байт, **d**=0. Регистр **CL** кодируется полем **r/m**=001, **CH** – полем **reg**=101. Операндов в памяти нет, поэтому **mod**=11. Построим машинный код данной команды:

КОП	d	w	mod	reg	r/m
001110	0	0	11	101	001
38h			E9h		

Таким образом машинный код данной команды **38E9**. Размер команды – 2 байта.

Команда 6: DIV R1

w = 0, если размер операнда **R1** равен одному байту, или **w**=1, если размер **R1** равен 2 или 4 байтам. Операндов в памяти нет, поэтому **mod**=11. **R1** – трёхбитное поле, которое кодирует номер регистра. Префикс имеется в том случае, если размер операнда **R1** равен 16 битам.

Префикс	КОП	w	mod	КОП	R1
66 ₁₆	1111011 ₂	w	11 ₂	110 ₂	III ₂

Машинный код 1: BE 12000000

Первый байт: **BEh**=10111110b. Код операции 1011 соответствует команде **MOV**, один из операндов которой имеет регистровую адресацию, второй – непосредственную. Разложим команду на части:

КОП	w	reg				
1011	1	110	00010010	00000000	00000000	00000000
BEh			12h	00h	00h	00h

Значение поля **reg**=110 соответствует регистру **ESI**. **w**=1, значит размер пересылаемых данных – 4 байта. Таким образом, искомая команда пересылает значение 12h=18 в регистр **ESI** и имеет вид: **MOV ESI, 18**.

Машинный код 2: 8BF0

Первый байт: **8Bh**=10001011b. **КОП**=100010 соответствует команде **MOV**, у которой операнды располагаются в памяти или в регистрах. Разложим команду на части:

КОП	d	w	mod	reg	r/m
100010	1	1	11	110	000
8Bh			F0h		

mod=11, значит оба операнда имеют регистровую адресацию. **d**=1, значит первый операнд закодирован в поле **reg**, а второй – в **r/m**. **w**=1 соответствует размеру двойного слова пересылаемых данных. **reg**=110 соответствует регистру **ESI**, **r/m**=000 – регистру **EAX**. Таким образом, машинный код **8BF0** соответствует команде **MOV ESI, EAX**.

Варианты заданий

- | | |
|--|--|
| 1. XOR BX, 100b
MOV DWORD PTR [EBX], 'b'
CMP [EBP+2], DL
SBB AX, DX
ADD EAX, [EBX*8+EDI+4Ah]

8A442E 02
B0 5A | 6. ADD DWORD PTR [EBX+200h], 3
MOV DI, 11010101b
XOR DX, [EBX*8+EDI+'A']
SUB BX, DX
SBB [ESI], AL

892B
8B5E 01 |
| 2. MOV AX, 1001b
ADD AX, [EBX]
CMP DI, AX
SUB EAX, [ECX*4+EAX+'Z']
AND WORD PTR [EDI], 12345h

02D8
8B56 07 | 7. AND AL, 'n'
MOV [EBX+200], AL
ADD BX, 1110101010101b
CMP DI, BX
TEST [ECX*2+EDI+761], 4567

8807
8B55 1F |
| 3. OR DI, 11110001b
MOV DWORD PTR [EBX], 'L'
TEST AX, [EDI+4Fh]
ADC CL, AL
CMP [EDX*4+ESI+7], ECX

8B5E 01
8AC3 | 8. OR AX, DX
MOV SI, 14789h
ADD AL, [ESI+8]
CMP BYTE PTR [EBP+4], 'j'
MOV AX, [EBX+EDI+17h]

BB 6400
B8 7800 |
| 4. MOV BP, '>'
ADC BX, [EBP]
MOV EAX, ECX
ADD BYTE PTR [EBX*4+7], 32
XOR [ESI*4+EDI+4], ECX

8A5C2E 0C
8953 02 | 9. MOV BP, 101111010101b
AND WORD PTR [EBP+2F2h], 23
ADD [EBX*8+EDI+'i'], EAX
SBB CX, AX
CMP DL, [EDI*4]

035B 02
BF 0400 |
| 5. CMP AX, 200
ADC BX, AX
MOV WORD PTR [EBX+4], 'y'
TEST CX, [ESI+12]
OR EDX, [EBP*8+EAX+14h]

8B56 07
2BDA | 10. CMP AX, [EBX+EDI+14Ah]
ADD BL, AL
TEST DX, [EBX*2]
MOV BYTE PTR [EDI], 'm'
XOR AX, [EBX+8]

136D00
895500 |

11. MOV ESI, 'c'
CMP BP, DI
ADD AL, [EBP+ESI+3]
SUB BYTE PTR [EBP], 45h
AND [ESI*2], CX

8B441F 0E
B05A

12. XOR BP, 30Dh
TEST SI, AX
MOV [EBP*2+ESI], EDX
SBB DWORD PTR [EBP], 10101b
ADD [EAX+'r'], DX

8BF0
0206

13. MOV AX, 120h
CMP DI, AX
SUB [EBP*8], AX
ADC CX, [ESI+101000101b]
OR BYTE PTR [EBP+EDI+8], 9

8BC8
BB 1200

14. ADD AX, 8080
AND BL, AL
TEST BYTE PTR [EBX+0E2h], 9
MOV DX, [ECX*4+34h]
OR [EAX], DX

8B441F 11
8B5D 02

15. CMP DI, 018h
ADC SI, AX
SBB [EBX*4], BP
MOV [EDX*2+EDI+2], 89799h
XOR CX, [EBX+EDI]

8955 00
BF 1400

16. OR DX, [EBP+ESI+3]
SUB CX, [EBP]
ADD EDX, ECX
MOV [EBP+2], DL
CMP AX, 10101010111b

8A542F 0A
02D8

17. MOV AX, [EBX+4]
SUB BX, AX
TEST [EBX+EAX+4], CX
ADD BL, CL
AND BYTE PTR [ESI], 'a'

8943 38
8BF0

18. CMP AL, 90h
MOV EAX, [ECX*4+ESI+'z']
OR [EBX], AX
ADC AX, DX
SUB DWORD PTR [EDX], 11123h

03D9
B8 C800

19. ADC AX, 'W'
SUB CL, DL
TEST BYTE PTR [ECX*8], 127
OR CX, [EDX*2+1000b]
MOV [EAX+1], EBX

8AD8
83E9 06

20. CMP ESP, 100
MOV BYTE PTR [EBP], 'Q'
ADD AX, [ESI]
XOR [EBX*2+ECX+2], EDX
SUB CX, AX

83E8 22
8BD8

21. AND SI, 48h
CMP AX, [ESI]
MOV SP, AX
SUB ECX, [EBP*4+EDI+200]
ADD WORD PTR [EBP], 'e'

890C1E
8A542F 0E

22. OR AX, 2002
MOV CX, AX
TEST AL, [EBX+'S']
SBB DWORD PTR [EDI], 1234
ADD EDX, [ECX*8+EAX+794h]

8B55 1F
8955 04

23. CMP AX, 20h
MOV [ESI], DI
AND ECX, [EBX*2+EDI+11110b]
SUB BYTE PTR [EAX+31], 'f'
ADD CL, DL

8B5E 01
03441F 04

24. TEST BP, 324
ADD EBP, ECX
SUB AL, [EDX]
OR [EBX*8+EBX+56], EAX
MOV WORD PTR [EBP+ESI], 'k'

8916
B8 C800

25. ADD BX, 700h
XOR CX, [EBX]
SBB BX, CX
CMP [EBP*4+EDI], ECX
MOV WORD PTR [ESI*8+1], 'a'

8BFB
03D1