第三章

2018302070001沈思源

3.13

3.13　若执行“MOV　CX,0F081H”后,再执行“MOVZX　EAX,CX”和“MOVSX　EBX,CX”,问 EAX =? EBX =?

解

首先`MOV,MOVSX,MOVZX`都是数据传送类指令，格式均为:

OP OPD , OPS　;OPD表示目的操作数，OPS表示源操作数

执行`MOV CX 0F081H`，注意到后缀是H，表示源操作数数字是16进制，16进制的数如果开头是A-F，那么开始数字前加上0，所以16位通用寄存器CX将数据`F081H`送到CX寄存器地址中，(CX)=F081H.

接着执行

`MOVZX EAX,CX` `MOVZX`指令表示的是将OPS内容-->OPD,对于OPD左边高位空缺的位采用**Zero填充**

`MOVSX EBX,CX`指令与之类似，但是对于OPD高位采用OPS的**符号填充**

因此，32位寄存器EAX,EBX分别对应如下:

```
MOV CX 0F081H
-> (CX)= 1111 0000 1000 0001
MOVZX EAX CX ;Zero
-> (EAX)= 0000 0000 0000 0000 1111 0000 1000 0001 = 0000F081H
MOVSX EBX CX ;Symbol
-> (EBX)= 1111 1111 1111 1111 1111 0000 1000 0001 = FFFFF081H
```

3.14

3.14　设 SP/ESP = 180H,AX = 198H,其余通用寄存器的值均为 0,问执行语句序列

PUSHA
PUSHAD
⋮
POPA

后,SP/ESP =? AX =?

解

首先明确指令对`PUSHA/POPA`和`PUSHAD/POPAD`

`PUSHA`指令表示压入8个字类型通用寄存器，依次顺序是:

AX,CX,DX,BX,TEMP(指令执行前的SP),BP,SI,DI

与之对应的`POPA`指令的出栈顺序就是:

DI,SI,BP,TEMP,BX,DX,CX,AX

注意，TEMP称作临时单元；PUSHA指令执行时发生以下操作:

TEMP=SP/ESP，SP/ESP-16 -> SP/ESP

之所以栈指针寄存器-16是因为目前的计算机体系中栈向下增长(堆相反)，而本处理机体系1字2字节，一共压入8字16字节，因此下移16字节地址

`PUSHAD`指令表示的是压入8个双字通用寄存器(32bit)，因此块的大小是2x2 x8=32bytes

TEMP=SP/ESP, SP/ESP-32->SP/ESP

依次将EAX,ECX,EDX,EBX,TEMP(指令执行前的ESP),EBP,ESI,EDI压入[SP]/[ESP]

因此，解答如下

```
PUSHA
-> SP/ESP = SP/ESP - 16  STACK:DI SI BP...AX
PUSHAD
-> SP/ESP = SP/ESP - 32  STACK:EDI ESI...EAX DI SI...AX
...
POPA
->SP/ESP =SP/ESP + 16    STACK:弹出16字节数据，即弹出EDI,ESI,EBP,TEMP给DI,SI,BP...AX

SP/ESP=180H-32=160H
而TEMP是指令执行前的ESP，TEMP的高16位给了CX，低16位(也就是SP)给了AX，指令执行前的SP/ESP有
SP/ESP=180H-16D=170H
= 0000 0000 0000 0000 0000 0001 0111 0000
                        ____________________
                                SP
因此AX=170H

```

3.15

3.15 若 EDX = 1034H, ESI = 15H, BUF = 289H, 问执行指令

LEA EBX, BUF[EDX][4 * ESI]

后, EBX = ? ESI = ?

解

先明确`LEA`指令: `LEA OPD,OPS`,指令将OPS的有效地址/偏移地址EA传送给OPD

首先看题干指令OPD部分，**BUF[EDX] [4*ESI]**，这是**相对基址变址寻址**寻址类型，因此:

```
OPD : EA=BUF+EDX+4*ESI=289H+1034H+4*15H
        =289H+1034H+60H
        =131DH
```

3.16

3.16 若 CX = 6, EDX = 0, EAX = 0F1501980H, CF = 1, 问执行指令

SHLD EDX, EAX, CL

后, EDX = ? EAX = ? CX = ? CF = ?

解

先明确`SHLD`指令.

S表示这是逻辑和移位指令中的**一般移位指令**，**H**表示逻辑移位，**R**表示右移，逻辑左移和算术左移(SAL,SHL)在物理上一样，但是右移不同。

逻辑右移： 将各位向右移动，而当前的最低有效位移到CF，将0补到最高有效位MSB上

算术右移： 将各位向右移动，而当前的最低有效位移到CF，但是MSB位保持不变

`SHLD` 指令是**双精度移位指令**,L表示这是双精度左移，格式如下

SHLD OPD,OPS1,OPS2

OPS1(寄存器)存放待移入的OPD值，移位时，OPD根据OPS2的值进行移位，移出的位-->CF(16位标志寄存器EFALGS的进位标志CF),空出位用OPS1的高位(SHLD向左移时)或者低位(SHRD右移时)来填补.

移位后，OPS1的值保持不变，CF等于最后一次移出的那个位值。

注意，移位数是指的是二进制数的移位数，比如如果是4，那么对应的16进制数就是移位1

```
SHLD EDX,EAX,CL ;EDX=0,EAX=0F1501980H
->注意到，CX=6=0000 0000 0000 0110
                          ---------
                              CL
  因此CL=6.
EAX由于16进制MSB位为字母表示，有前缀0，实际值为F150 1980
EAX = 1111 0001 0101 0000 0001 1001 1000 0000
EDX = 0
EDX左移6位,低位由EAX高位进行填补,因此移位后的EDX有
EDX=  0000 0000 0000 0000 0000 0000 0011 1100
CF则对应EDX最后一次移出的那个位值，显然是0
EAX作为OPS1，保持不变.
CX在ECX中，未受影响，保持不变

综上,移位后
EDX=3CH,EAX=0 F1501980H,CX=6,CF=0;
```