**第 4周作业 常用指令和转移指令**

**1、阅读下面的程序，回答问题 每一小题 10分，总分 50分**

void func ()

{ unsigned short us = 65535;

unsigned int ui;

short s = -1;

int i;

int x = 0, y = 0; ; ①

ui = us;

i = s; ; ②

if (ui > 0) x = 1;

if (i > 0) y = 1; ; ③

ui = ui >> 1; ; >>1 右移一个二进制位

i = i >> 1; ; ④

ui = ~ui; ; 按位取反

i = -i; ; ⑤

}

**⑴ 执行完** ① 处语句后，

us = 0x \_FFFF\_\_\_\_\_\_ s = 0x \_FFFF\_\_\_\_\_\_\_（2个字节的16进制数）

**⑵ 执行完** ② 处语句后，

ui = 0x \_0000FFFF\_\_\_\_\_\_\_ i = 0x \_\_FFFFFFFF\_\_\_\_\_\_（4个字节的16进制数）

ui=us; 对应的机器指令： \_movzx\_\_\_\_ eax, us 、 mov ui, eax

i=s; 对应的机器指令： \_movsx\_\_\_\_\_ eax, s 、 mov i, eax

**⑶ 执行完 ③** 处语句后，

x = \_1\_\_\_\_\_\_\_\_\_ y = \_\_\_0\_\_\_\_\_\_\_

if (ui>0) x=1; 对应的机器指令： cmp ui,0、 \_\_jbe\_\_\_ lp1、 mov x,1、 lp1:…

if (i >0) y=1; 对应的机器指令： cmp i,0、 \_\_jle\_\_\_ lp2、 mov y,1、 lp2:…

**⑷ 执行完 ④** 处语句后，

ui = 0x \_00007FFF\_\_\_\_\_\_ i = 0x \_\_FFFFFFFF\_\_\_\_\_ （4个字节的16进制数）

ui = ui >> 1; 对应的机器指令： \_\_SHR\_\_\_\_\_ ui,1

i = i >> 1; 对应的机器指令： \_\_\_SAR\_\_\_\_ i, 1

**⑸**  **执行完**  ⑤ 处语句后，

ui = 0x \_\_FFFF8000\_\_\_\_\_\_ i = 0x \_00000001\_\_\_\_\_ （4个字节的16进制数）

ui = ~ui; 对应的机器指令： \_\_NOT\_\_\_\_\_ ui

i = -i; 对应的机器指令： \_\_\_NEG\_\_\_\_ i

**2、 设 esp = 0x 12345678 eax= 0x11223344 10分  
 执行 push eax 后， esp = 0x12345674**\_\_\_\_\_\_ **之后，再执行 pop ebx， esp = 0x12345678**\_\_\_ **ebx = 0x11223344**

**3、 设 esp = 0x 12345678， 执行 10分**

**mov dword ptr [esp], 12345566h  
pop eax**

**后， esp = 0x1234567C eax= 0x12345566**

**4、 设有机器指令 （指令地址为 0x 009A1230）： 10分  
 009A1230 75 07 ; 75 对应 jne**

**设执行该指令前， ZF =1，执行该指令后， eip = 0x 009A1232  
 若 ZF = 0， 执行该指令后， eip = 0x 009A1239 为什么？**

**在取指令、对指令进行译码后，eip = 0x 009A1232。当ZF=1，JNE的转移条件不成立，该指令执行结束，故 eip = 0x 009A1232；当ZF=0时，JNE的转移条件成立，该指令eip+位移量 ->eip, 故 eip = 0x 009A1232+07 = 0x 009A1239。**

**5、 C 语言中的 &、|、^、~、-（符号）分别对应什么指令操作符？ 10分**

**& AND**

**| OR**

**^ XOR**

**~ NOT**

* **NEG**

6、设有 int x,y,z; 写出 z = x && y; 对应的汇编语句段（只写完成该功能的程序段）**10分**  
 CMP x, 0

JZ Z\_ZERO

CMP y, 0  
 JZ Z\_ZERO  
 MOV Z, 1  
 JMP OVER

Z\_ZERO: MOV Z, 0  
 OVER:

另外的写法：

MOV Z, 0  
 CMP x, 0

JZ OVER

CMP y, 0  
 JZ OVER  
 MOV Z, 1  
 OVER: