

北京理工大学 2010 汇编中期检测

(注：本次测验按%15 折算为期末成绩，超过 90 分可不做平时作业，且平时成绩按满分计,考试时间：90 分钟，若有抄袭：全部计 0 分)

一：选择题(每题 3 分，共 18 分)

1：以下寻址方式错误的是？ ()

A: mov eax,0000H

B:mov eax, 8[ebx]

C:mov eax,[ebp][ebp]

D:mov eax,[esp][esp]

2:JMP DWORD PTR [BX]属于以下哪种指令跳转方式？ ()

✓ A:段间间接寻址

B: 段内间接寻址

C:段内短转移

D: 段间直接转移

3: mov 指令使用错误的是？ ()

A:mov eax,ebx

B:mov cs,ax

C:mov ds,ax

D:mov eax,dword ptr LAB (LAB 为符

号地址)

4:下列说法错误的是？ ()

A:若使用 mul RSC 语句，则 RSC 不能为立即数

B:若使用 div ebx，则商存在 EAX 中，余数存在 EDX 中

C: 同一个数使用压缩 BCD 码，与非压缩 BCD 码，占用的位数不同

D: 16 位汇编程序中，若使用 `loop` 指令实现循环，设字节型变量 `A` 存储了循环次数，循环开始前使用语句 `mov CL,A` 就可以完全实现循环次数加载。

5: 下列说法正确的是?()

A: `JG` 实现带符号数比较转移

B: `JBE` 实现无符号数比较转移

C: 使用 `invoke` 指令时某些情况下可以带参数

D: `call` 指令与 `jmp` 指令有类似的地方，因此 `call` 指令不需要保存对应的返回地址

6: 以下说法错误的是? (D)

A: 调用操作系统中断，输入字符时，`AH` 的值设定为 `01H`

B: 调用操作系统中断，输入字符串时，`AH` 的值设定为 `0AH`

C: 16 位汇编程序中，段基址与偏移量的表示方式所能表示的最大值为 `1M`

D: 16 位汇编程序中通常只能对输入数据按单个字符处理。工作繁琐。

二: 判断题(每题 1 分，共 10 分)

(1)数据段中定义的变量属于全局变量。(T)

(2).`data`，`invoke` 与 `mov` 属于同一类指令。(F)

(3)局部变量位于堆栈，函数返回时，该局部变量的生存期结束。(F)

(4)16 位汇编程序中可以调用 `bios` 或操作系统中断进行数据输入，32

位汇编程序中同样可以调用这些中断进行数据输入与输出(X)

(5)a dw ? 声明了一个双字节的内存变量

(6)16 位与 32 位汇编程序都可以将一个立即数入栈

(7)32 位程序中程序计数器 eip 保存了当前正在执行的指令

(8)以 _cdecl 方式调用的函数采用 ret n 形式来平衡堆栈()

(9)bp,esp,ebp 默认的段寄存器为 ds()

(10)快速调用方式与标准调用方式,函数的参数传递方式相同,都是从右向左依次入栈

三：填空题(每空 2 分，共 16 分)

(1) 设变量对应的段基址为 D，偏移量为 E，若在 16 位中则该变量的实际地址是(),若在 32 位中该变量的实际地址是(D+E);

(2) 16 位汇编程序 ret 可以用语句 mov AX, (); int 21H 替换;

(3) 32 位汇编程序中使用 link 命令生成控制台子系统应用程序的关键参数是();

(4)设内存地址 0: 12H, 1:10H, 2:20H, 3:10H

从 0 号单元取得单字内容为(),双字内容为()

(5)windbg 时指定程序加载地址的伪寄存器的符号为(\$)

(6)当数 A, B 解释为()数且 CF=1 一定发生溢出。(带符号, 无符号)

四：简答题(5+5+6，共 16 分)

1: 解释什么是 32 位中内存的平坦模式，它与 16 位程序中的内存模

式存在哪些区别？(主要从程序内存分配方面回答)



书P140

2: 设双字节内存变量 a, b 使用至少三种方式实现两个变量的交换。



a、b分别放进寄存器直接换；XCHG；按字节挨个换

3: 简述机器语言，汇编语言，高级语言的区别。



五：综合应用题(10+10+20)

1: 观察:以下代码回答相关问题

Breakpoint 0 hit

eax=75c3ee0a ebx=7ffdb000 ecx=00000000 edx=0106100b esi=00000000
edi=00000000

eip=0106100b esp=002dfc78 ebp=002dfc80 iopl=0

cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000

0106100b be0a000000 mov esi,0Ah

0:000> p

eax=75c3ee0a ebx=7ffdb000 ecx=00000000 edx=0106100b esi=0000000a
edi=00000000

eip=01061010 esp=002dfc78 ebp=002dfc80 iopl=0

cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000

efl=00000246

01061010 bf14000000 mov edi,14h

0:000>

eax=75c3ee0a ebx=7ffdb000 ecx=00000000 edx=0106100b esi=0000000a
edi=00000014

eip=01061015 esp=002dfc78 ebp=002dfc80 iopl=0

cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000

efl=00000246

01061015 6a03 push 3

```
0:000> t
eax=75c3ee0a ebx=7ffdb000 ecx=00000000 edx=0106100b esi=0000000a
edi=00000014
eip=01061017 esp=002dfc74 ebp=002dfc80 iopl=0
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000
efl=00000246
```

```
01061017 6a02 push 2
```

```
0:000>
eax=75c3ee0a ebx=7ffdb000 ecx=00000000 edx=0106100b esi=0000000a
edi=00000014
eip=01061019 esp=002dfc70 ebp=002dfc80 iopl=0
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000
efl=00000246
```

```
01061019 6a01 push 1
```

```
0:000>
eax=75c3ee0a ebx=7ffdb000 ecx=00000000 edx=0106100b esi=0000000a
edi=00000014
eip=0106101b esp=002dfc6c ebp=002dfc80 iopl=0
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000
efl=00000246
```

```
0106101b e8e0ffffff call image01060000+0x1000
```

(01061000)(函数名: AddFunc)

```
0:000>
eax=75c3ee0a ebx=7ffdb000 ecx=00000000 edx=0106100b esi=0000000a
edi=00000014
eip=01061000 esp=002dfc68 ebp=002dfc80 iopl=0
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000
```

```
01061000 55 push ebp
```

```
0:000>
eax=75c3ee0a ebx=7ffdb000 ecx=00000000 edx=0106100b esi=0000000a
edi=00000014
eip=01061001 esp=002dfc64 ebp=002dfc80 iopl=0
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000
```

```
01061001 8bec mov ebp,esp
```

```
0:000>
eax=75c3ee0a ebx=7ffdb000 ecx=00000000 edx=0106100b esi=0000000a
edi=00000014
eip=01061003 esp=002dfc64 ebp=002dfc64 iopl=0
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000
efl=00000246
```

```
01061003 8bc6 mov eax,esi
```

```
0:000>
```

```

eax=0000000a ebx=7ffdb000 ecx=00000000 edx=0106100b esi=0000000a
edi=00000014
eip=01061005 esp=002dfc64 ebp=002dfc64 iopl=0
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000
efl=00000246
01061005 03c7          add     eax,edi
0:000>
eax=0000001e ebx=7ffdb000 ecx=00000000 edx=0106100b esi=0000000a
edi=00000014
eip=01061007 esp=002dfc64 ebp=002dfc64 iopl=0
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000
efl=00000206
01061007 c9          leave
0:000>
eax=0000001e ebx=7ffdb000 ecx=00000000 edx=0106100b esi=0000000a
edi=00000014
eip=01061008 esp=002dfc68 ebp=002dfc80 iopl=0
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000
efl=00000206
01061008 c20c00      ret     0Ch

```

(1)调用 AddFunc 时使用的调用方式是？并给出理由。(3 分)



(2)分析 leave 子句实际执行了那些操作。(3 分)



(4) 分析 ret 0ch 子句执行了哪些操作。(4 分)




二：程序填空(8 分)，并分析该程序的主要目的(2 分)，共 10 分

.386

.model flat,stdcall



Option casemap:none

 Msvcrt.lib

printf  c :dword :vararg

.data

dArray dword 20,15,70,30,32,89,12

ITEMS EQU  

szFmt byte 'dArray[%d]=%d',0dh,0ah,0

.code

Start:

 mov ecx,ITEMS-1

i10:

 xor esi,esi

i20:

 mov eax,dArray[esi*4]

mov ebx,dArray[esi*4+4]

cmp eax,ebx

jl 

mov dArray[esi*4],ebx

mov dArray[esi*4+4],eax

i30:

inc esi

cmp esi,ecx

jb 

loop i10

xor edi,edi

i40:

invoke printf,szFmt,edi,dArray[edi*4]

inc edi

cmp edi,ITEMS

JB 

Ret

End start



三：(开放题)请从汇编的角度，结合你的编程经验，描述你对以下问题的看法。(注：3 选 2 即可，全答按具体情况给分)

(1) 汇编语言的难移植性的原因。



(2) 通过上一题程序计数器 `eip` 的变化，你能得出哪些结论？



(3) 谈谈你对汇编学习过程中的一些收获

