第6、7章 顺序和分支程序设计

一、学习内容

(1) 程序设计的一般步骤

(2) 顺序程序设计

(3) 分支程序设计

(4) 循环程序设计方法

要求掌握汇编语言程序设计的基本技术，能够熟练的编写、调试汇编语言程序。

多动手！

第6章 顺序和分支程序设计

6.2.2 存储模型说明伪指令

- .model 存储模型 [,语言类型]
- 对于Win32程序，存储模型选择为 flat
  代码和数据全部放在同一个4G空间内；
- “语言类型”指定了函数参数的传递方法和释放参数所占空间的方法；
  语言类型为 stdcall 或者 c
  .函数原型描述中最右边的参数最先入栈、最左边的参数最后入栈；
  .stdcall 被调用者（即函数内）在返回时释放参数占用的堆栈空间；
  .c 在调用函数中释放参数所占的空间。

6.3 转移控制指令

```
if  (x ==y )
{
      Statements 1
      .........
}
else
{
      Statements 2
      ......
}
```

```
    MOV   AX, X
    CMP   AX, Y
    JNE   L1
    Statements 1
      ......
    JMP   L2
L1:
    Statements 2
      ......
L2:
```

Q: c 程序中分支语句的执行流程是什么？与机器指令有何对应关系？

6.3.1 转移指令概述

- 转移指令
  - 条件转移
    - 简单条件转移（10条）
    - 无符号数条件转移(4条)
    - 有符号数条件转移(4条)
  - 无条件转移
    - 直接转移
    - 间接转移

JMP **

6.3.2 简单条件转移指令

根据单个标志位 CF、ZF、SF、OF、PF的值确定是否转移。

语句格式：　[标号:]　操作符　标号

如果转移条件满足，则(EIP)+ 位移量 → EIP，否则，执行紧跟转移指令之后的那条指令。

6.3.2 简单条件转移指令

地址(A): main(void)

查看选项

- [x] 显示代码字节
- [x] 显示地址
- [x] 显示源代码
- [x] 显示符号名
- [ ] 显示行号

```
00B3842A 83 FB 64             cmp         ebx,64h
                    ;   if (ebx>100) goto exit
    jg    exit
00B3842D 7F 05                jg          lp+0Ah (0B38434h)
    add   eax, ebx  ;   (eax) = (eax) + (ebx)
00B3842F 03 C3                add         eax,ebx
    inc   ebx       ;   (ebx) = (ebx) +1
00B38431 43                   inc         ebx
    jmp   lp        ;   goto lp
00B38432 EB F6                jmp         lp (0B3842Ah)
exit:
   mov   eax, offset lpFmt
00B38434 B8 00 B0 BA 00       mov         eax,offset lpFmt (0
   mov   ebx, len1
```

6.3.2 简单条件转移指令

JZ / JE	ZF=1时，转移
JNZ / JNE	ZF=0时，转移
JS	SF=1时，转移
JNS	SF=0时，转移
JO	OF=1时，转移
JNO	OF=0时，转移
JC	CF=1时，转移
JNC	CF=0时，转移
JP / JPE	PF=1时，转移
JNP / JPO	PF=0时，转移

6.3.3 无符号条件转移指令

JA / JNBE　　标号　　(CF=0 且 ZF=0，转移)

JAE / JNB　　标号　　(CF=0 或 ZF=1，转移)

JB / JNAE　　标号　　(CF=1 且 ZF=0，转移)

JBE / JNA　　标号　　(CF=1 或 ZF=1，转移)

6.3.3 无符号条件转移指令

无符号数条件转移指令的理解

```
      CMP    AX，BX
      JA     L1
        ……          CF=0 且 ZF=0，转移
L1:
```

将(AX),(BX)中的数据当成无符号数，
若 (AX)>(BX)，执行 (AX)-(BX)，
则 CF一定会为0，ZF=0

什么时候使用无符号数条件转移指令？

例1：(AX)= 1234H,　(BX)=0234H

例2：(AX)= 0A234H,　(BX)=0234H

例3：(AX)= 0A234H,　(BX)=09234H

6.3.4 有符号条件转移指令

JG / JNLE　　标号

当 SF=OF 且 ZF=0时，转移

JGE / JNL　　标号

当 SF=OF 或者 ZF=1时，转移

JL / JNGE　　标号

当 SF≠OF 且 ZF=0时，转移

JLE / JNG　　标号

当 SF≠OF 或者 ZF=1时，转移

6.3.4 有符号条件转移指令


```
    CMP    AX，BX
    JG     L1
      ……
L1:
```

有符号数条件转移指令的理解

将(AX),(BX)中的数据当成有符号数，
若 (AX)>(BX)，执行 (AX) - (BX)，
则 SF、OF会相等，ZF=0。

例1: (AX)= 1234H, (BX)=0234H

SF=0、OF=0, ZF=0, CF=0

不论使用 JA 还是 JG ，转移的条件均成立

6.3.4 有符号条件转移指令

例2：(AX)= 0A234H, (BX) =0234H

执行(AX) - (BX)后：

SF = 1, ZF=0, CF=0, OF = 0

对于 JA ，条件成立 （CF=0 ， ZF=0）

对于 JG ，条件不成立 （因为SF≠OF）

例3：(AX)= 0A234H, (BX) =09234H

SF = 0 ， ZF=0, CF=0, OF=0

对于JA、JG，条件均成立

什么时候使用有符号转移指令，什么时候使用无符号数转移指令？

6.3.5 无条件转移指令

格式	名称	功能
JMP 标号	直接	(EIP)+位移量 → EIP
JMP OPD	间接	(OPD) → EIP

6.3.5 无条件转移指令

间接转移方式中，除了立即数寻址方式外，其它方式均可以使用。

```
BUF     DD    L1     ; L1为标号

(1)   JMP    L1
(2)   JMP    BUF
(3)   LEA    EBX , BUF
      JMP    DWORD PTR [EBX]
(4)   MOV    EBX , BUF
      JMP    EBX
```

功能等价的转移指令

6.3.5 无条件转移指令

例4：根据不同的输入，执行不同的程序片段。

构造指令地址列表

输入1，执行程序段 LP1 ：
输入2，执行程序段 LP2 ：
输入3，执行程序段 LP3 ：
……

JMP LP1
……
JMP LP2
……
JMP LP3

如果分支很多，每个分支均使用JMP 标号，程序难看，臃肿！

6.3.5 无条件转移指令

例4：根据不同的输入，执行不同的程序片段。

构造指令地址列表

```
FUNCTAB   DD   LP1, LP2, LP3

JMP   FUNCTAB[EBX*4]
```

(EBX)=0, 跳转到 LP1处
(EBX)=1, 跳转到 LP2处

6.4 简单分支程序设计

(1) 选择合适的转移指令；

(2) 为每个分支安排出口；

(3) 将分支中的公共部分尽量放到分支前或分支后的公共程序段中；

(4) 流程图、程序对应

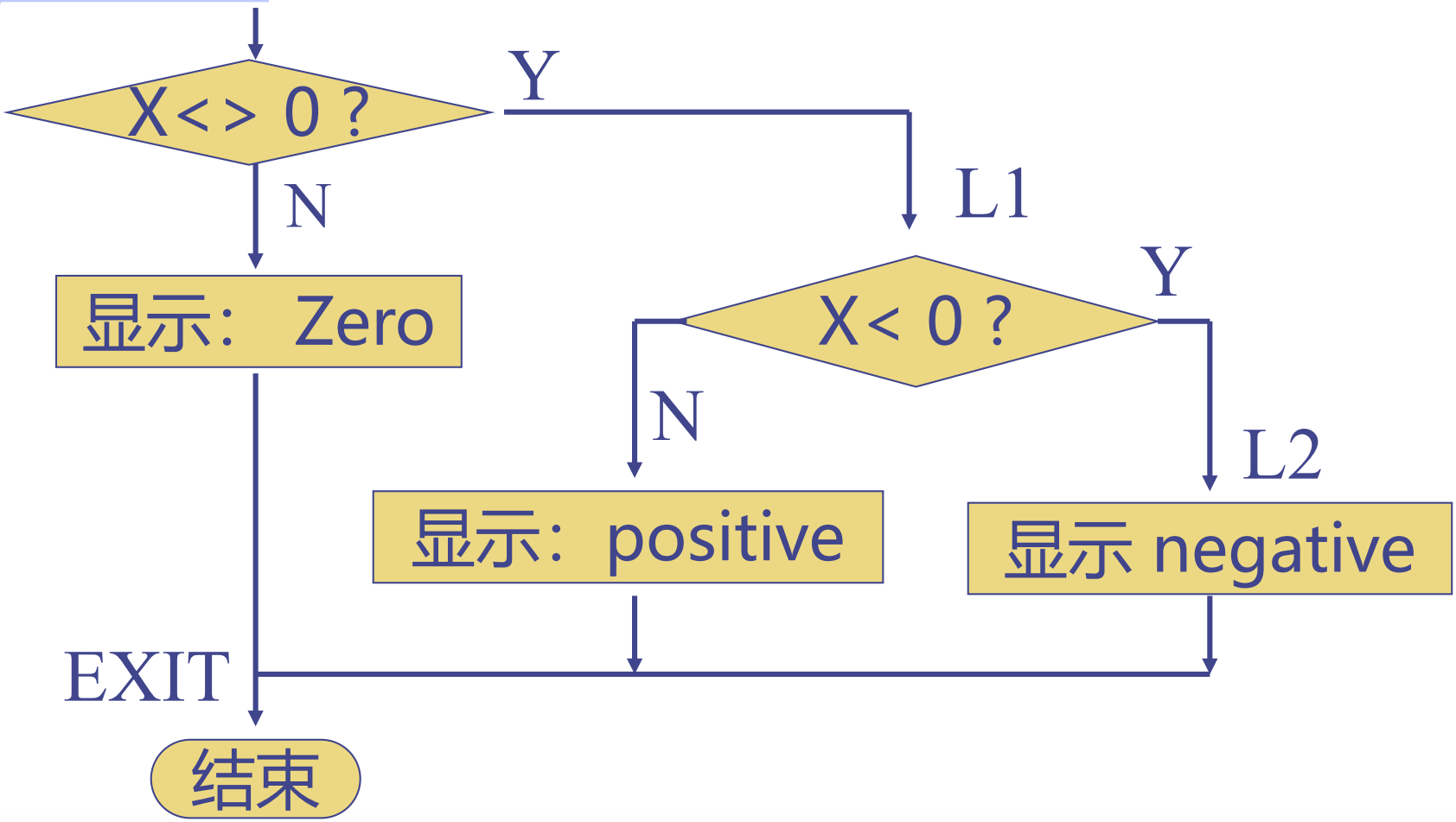
(5) 调试时，逐分支检查

6.4 简单分支程序设计

例：判断 x 中的内容,
为正,显示 positive > 0 ；
为负,显示 < 0;
为0, 显示 =0

实验：使用不同转移指令后的结果比较合分析

6.4 简单分支程序设计



6.4 简单分支程序设计

例：判断 x 中的内容，
为正,显示 positive > 0 ；
为负,显示 < 0;
为0, 显示 =0

实验：使用不同转移指令后的结果比较合分析

```
.data
lpFmt    db"%s",0ah, 0dh, 0
x        dw  -5
pos      db  'positive > 0', 0
neg1     db  '< 0',0
zero1    db  '= 0',0
```

6.4 简单分支程序设计


```
cmp   x, 0
jz    zero1
js    neg1——可以改成哪些转移语句?
invoke printf, offset lpFmt, offset pos
jmp   l1
zero1:invoke printf, offset lpFmt, offset zero1
jmp   l1
neg1:invoke printf, offset lpFmt, offset neg1
l1:invoke ExitProcess, 0
```

6.4 简单分支程序设计

TIPS：

- 不要随意摆放各个分支、不要随意跳转；
- 最好是将两个分支紧靠在一起，使得整个条件语句像一个小模块，有一个起始点和一个终止点；
- 从起始点到终止点之间的语句全部都是该条件语句的组成部分，而不含有其他语句。

```
    MOV   AX,  X
    CMP   AX,  Y
    JNE    L1
    Statements 1
       ......
    JMP    L2
L1:
   Statements 2
       ......
L2:
```

6.5 多分支程序设计

6.5.1 多分支向无分支的转化

例：统计一个字符串中各个字母出现的次数。

定义两张字母表，再用XLAT指令：

```
alb   db  'abcd…'
albh  db  'ABCD'
```

例：当x==1时，显示‘Hello，One’；
当x==2时，显示‘Two’；
当x==3时，显示‘Welcome,Three’,……,
即x为不同的值，显示不同的串。

定义地址表　TAB　DD　L1, L2, L3;

使用寻址方式：　TAB[EBX*4]

——获取每个串的首地址

6.6 条件控制流伪指令

条件控制流伪指令

```
.IF   条件表达式
      语句序列
.ENDIF

.IF   条件表达式
      语句序列
.ELSE
      语句序列
.ENDIF
```

条件表达式：

(1)关系运算

==

!=

>

>=

<

<=

(2) 逻辑运算

&&

||

6.6 条件控制流伪指令

条件控制流伪指令

```
.IF    条件表达式
       语句序列
.ELSEIF    条件表达式
           语句序列
……
.ELSE
       语句序列
.ENDIF
```

6.6 条件控制流伪指令

例： 判断字节单元 x 中的内容

为正, 显示 positive > 0

为负, 显示 < 0

为0, 显示 zero

6.6 条件控制流伪指令


```
.data
x       db   -5
bufp  db  'positive > 0`,0
bufn  db  ' < 0 `,0
zero  db  'zero `,0
Fmt   db "%s",0ah,0dh, 0

.code ……
```

```
.if  x==0
    lea ebx, zero
.elseif  x>0
    lea ebx, bufp
.else
    lea ebx, bufn
.endif
invoke  printf,
        offset Fmt, ebx
```

```
invoke  printf, offset Fmt, offset bufp
```

注意：显示一个串，就要给出串的首地址，串以0结束.

6.6 条件控制流伪指令

实践：结果不正确，为什么？

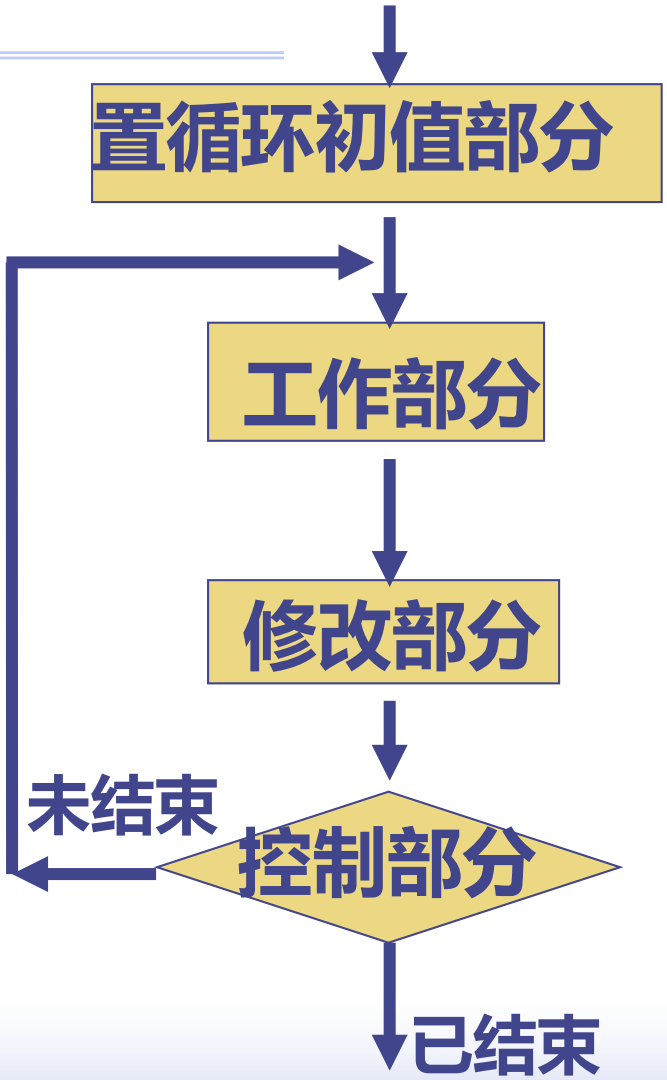
分析原因：观察目标程序，发现其用的是无符号数比较转移指令。即masm 将X当作无符号数来翻译的。

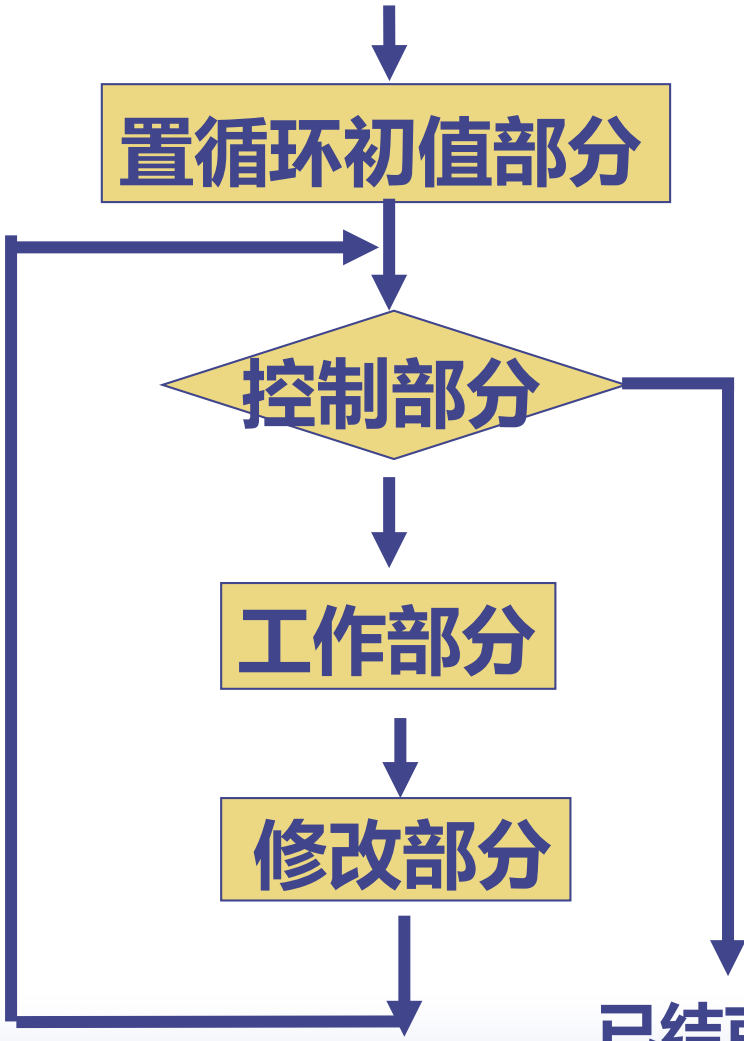
若要将其当作有符号数，定义形式是：

```
X    SBYTE    -5
```

同样，有SWORD, SDWORD……

7.1 循环程序的结构





7.1 循环程序的结构

循环控制方法

计数控制：循环次数已知时常用

（1） 倒计数

```
        ……
        MOV   ECX，循环次数
LOOPA：  ……
        ……
        DEC    ECX
        JNE    LOOPA
```

- 循环次数n → 循环计数器
- 每循环一次，计数器减1
- 直到计数器值为0时，结束循环

7.1 循环程序的结构

循环控制指令

X86提供的四种计数控制循环转移指令

```
LOOP     标号
LOOPE    标号
LOOPNE   标号
JCXZ     标号
```

(1) LOOP 标号

(ECX) -1 → ECX

若 (ECX) 不为0, 则转标号处执行。

基本等价于：

```
DEC   ECX
JNZ   标号
```

（LOOP指令对标志位无影响 ！ ）

7.1 循环程序的结构

循环控制方法

计数控制：循环次数已知时常用

（2） 正计数

```
        ……
        MOV   ECX，0
LOOPA：
        ……
        INC   ECX
        CMP   ECX,   n
        JNE   LOOPA
```

7.1 循环程序的结构

循环控制方法

条件控制：循环次数不固定

通过指令来测试条件是否成立，
决定继续循环还是结束循环。

例：求一个以0为结束符的字符串的长度

7.1 循环程序的结构

循环控制方法

阅读程序段，指出其功能：

```
      MOV   CL, 0
L:    AND   AX , AX
      JZ    EXIT
      SAL   AX , 1
      JNC   L
      INC   CL
      JMP   L
EXIT:
```

7.1 循环程序的结构

循环控制方法

阅读程序段，指出其功能：

```
        MOV    CL,   0
        MOV    BX,  16
L:      SAL    AX ,  1
        JNC    NEXT
        INC    CL
NEXT:   DEC    BX
        JNZ    L
```

7.4 循环程序中的细节分析

有n个元素存放在以buf为首址的双字存储区中，试统计其中负元素的个数存放到变量r中。

```
    lea   ebx, buf                ; ebx : 待访问数据的地址
    mov   ecx, n                  ; ecx : 循环次数
    xor   eax, eax                ; eax : 负数个数
lopa:
    cmp   dword ptr [ebx],0       ;工作部分（循环体）
    jge   next
    inc   eax
next:
    add   ebx, 4                  ;修改部分
    dec   ecx
    jnz   lopa                    ;控制部分
```

7.4 循环程序中的细节分析


```
    lea   ebx, buf      ;  (1)
    mov   ecx, n        ;  (2)
    xor   eax, eax      ;  (3)
lopa:                   ;  (4)
    cmp   dword ptr [ebx],0
    jge   next          ;  (6)
    inc   eax           ;  (7)
next:                   ;  (8)
    add   ebx, 4        ;  (9)
    dec   ecx           ;  (10)
    jnz   lopa          ;  (11)
```

Q：能否交换 (1)-(3) 行的顺序？

Q：将lopa写到第（1）行，运行结果如何？

Q：将第(1)行写到lopa:之下，运行结果如何？

Q：交换 (9)-(10)行的顺序，运行结果如何？