第7章 循环程序设计

7.1 循环程序的结构

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
```

循环控制方法

Q：能否用其他寄存器或者变量控制循环次数？

- 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:
```

（AX）中 1 出现的次数 -> CL

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
```

（AX）中 1 出现的次数 -> CL

7.1 循环程序的结构

80X86提供的四种计数控制循环转移指令

循环控制指令

```
LOOP     标号
LOOPE    标号
LOOPNE   标号
JECXZ    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

(2) LOOPE /LOOPZ 标号

(ECX) - 1 → ECX

若(ECX)不为0, 且ZF=1,则转标号处执行。

(等于或为0循环转移指令, 本指令对标志位无影响)

32位段用 ECX, 16位段用 CX

(2) LOOPE /LOOPZ 标号

例：判断以BUF为首址的10个字节中是否有非0字节。
有，则置ZF为0，否则ZF置为1。

```
        MOV     ECX, 10
        MOV     EBX, OFFSET BUF -1
L3 :    INC     EBX
        CMP     BYTE PTR [EBX], 0
        LOOPE   L3
```

(3) LOOPNE /LOOPNZ 标号

(ECX) -1 → ECX

若(ECX)≠0, 且ZF=0,则转标号处执行。

例：判断以MSG为首址的10个字节中的串中是否有空格字符。无空格字符，置ZF为0，否则为1。

```
        MOV     ECX,   10
        MOV     EBX,   OFFSET MSG -1
L4 :    INC     EBX
        CMP     BYTE PTR [EBX], ‘ ’
        LOOPNE  L4
```

(4) JECXZ 标号

若 （ECX）为0，则转标号处执行。

（先判断，后执行循环体时，可用此语句，

标号为循环结束处）

7.2 C循环语句的反汇编


```
int  i = 0, sum = 0, a[5];
    ......
for  (i = 0; i < 5; i++)      sum += a[i];
```

```
00D71750   mov    dword ptr [i],0
00D71757   jmp    f+62h (0D71762h)
00D71759   mov    eax,dword ptr [i]
00D7175C   add    eax,1
00D7175F   mov    dword ptr [i],eax
00D71762   cmp    dword ptr [i],5
00D71766   jge    f+77h (0D71777h)
00D71768   mov    eax,dword ptr [i]
00D7176B   mov    ecx,dword ptr [sum]
00D7176E   add    ecx,dword ptr a[eax*4]
00D71772   mov    dword ptr [sum],ecx
00D71775   jmp    f+59h (0D71759h)
00D71777   // 循环结束
```

Debug 版本

Q：程序段有多少条语句？
完成整个循环，需要要执行多少条语句？

程序段：12条指令
循环执行指令数 50余条
(10*5+2+…)

Q：可以做哪些优化？

7.2 C循环语句的反汇编


```
int  i = 0, sum = 0, a[5];
    ......
for  (i = 0;i < 5;i++)      sum += a[i];
```

Release 编译优化

```
mov   eax,dword ptr [ebp-8]
add   eax,dword ptr [ebp-0Ch]
add   eax,dword ptr [ebp-10h]
add   eax,dword ptr [ebp-14h]
add   eax,dword ptr [ebp-18h]
mov   sum,  eax
```

Q：如果循环次数 5 改为一个变量，又如何优化 ？

```
for(i = 0; i < n; i++) sum += a[i] ;
```

7.2 C循环语句的反汇编


```
int  i = 0, sum = 0, a[5];
    ......
for  (i = 0; i < n; i++)      sum += a[i];
```

Release 编译优化

```
            mov   edi, sum   ; edi来存放 和
            xor   eax, eax   ; eax 对应 i
            mov   edx, n     ; edx 对应 n
            jmp   main+0C5h (05E1145h)
005E1140    add   edi,dword ptr [ebp+eax*4-18h]
005E1144    inc   eax
005E1145    cmp   eax,edx
005E1147    jl    main+0C0h (05E1140h)
```

变量与寄存器绑定，语句数量大幅减少！

循环部分：由 10 条语句减为 4条语句！

7.2 C循环语句的反汇编

Release 编译优化

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

void main()
{
    char  buf1[20];
    char  buf2[20];
    int   i;
    scanf("%s", buf1);
    for (i = 0;i < 20;i++)
        buf2[i] = buf1[i];
    printf("%s\n", buf2);
    return;
}
```

7.2 C循环语句的反汇编

Release 编译优化

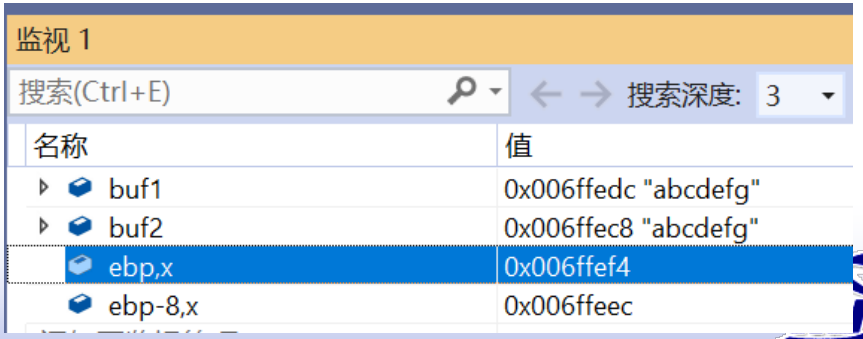
```
        for (i = 0;i < 20;i++)
                buf2[i] = buf1[i];
        printf("%s\n", buf2);
0025109E  mov         eax,dword ptr [ebp-8]
002510A1  movups      xmm0,xmmword ptr [buf1]
002510A5  mov         dword ptr [ebp-1Ch],eax
002510A8  lea         eax,[buf2]
002510AB  push        eax
002510AC  push        offset string "%s\n" (0252104h)
002510B1  movups      xmmword ptr [buf2],xmm0
002510B5  call        printf (0251020h)
```

Q：这段代码如何解读？

buf1 的前16个字节拷贝到 xmm0

后4个字节拷贝到 eax

再分别送到 buf2 相应位置



7.2 C循环语句的反汇编

Q: 能否写一个C程序，能实现 buf1 中的内容拷贝到 buf2 中，但Release又不好优化?

```
void main()
{
    char  buf1[20];
    char  buf2[20];
    int   i;
    scanf("%s", buf1);
    ……
    printf("%s\n", buf2);
    return;
}
```

```
void fcopy(char* dst, char* src)
{
    int i;
    for (i = 0;i < 20;i++)
    {
        *dst = *src;
        dst++;
        src++;
    }
}

fcopy(buf2,buf1);
```

```
fcopy(buf1-20, buf1);    // 可能存在数据相关，未优化
```

7.2 C循环语句的反汇编

華中科技大學

```
scanf("%s", buf1);
003D1090  lea           eax,[ebp-18h]
003D1093  push          eax
003D1094  push          3D2100h
003D1099  call          003D1050
003D109E  add           esp,8
003D10A1  xor           eax,eax
fcopy(buf1-20, buf1);
003D10A3  mov           cl,byte ptr [ebp+eax-18h]   ;(EAX)=0
003D10A7  mov           byte ptr [ebp+eax-2Ch],cl   ;buf1=[ebp-18h]
003D10AB  inc           eax                         ;buf1-20
003D10AC  cmp           eax,14h                     =[ebp-2Ch]
003D10AF  jl            003D10A3

printf("%s\n", buf2);
```

总结

- 用分支转移指令，实现循环；
- 有专门的循环指令：LOOP、LOOPE、LOOPNE、JECXZ

- 编译优化

  循环展开：消除了循环

  与寄存器绑定：减少访存操作，减少指令

  用XMM寄存器、成组运算等，减少指令