第7章 循环程序设计

華中科技大學

一、学习内容

循环程序设计方法

二、学习重点

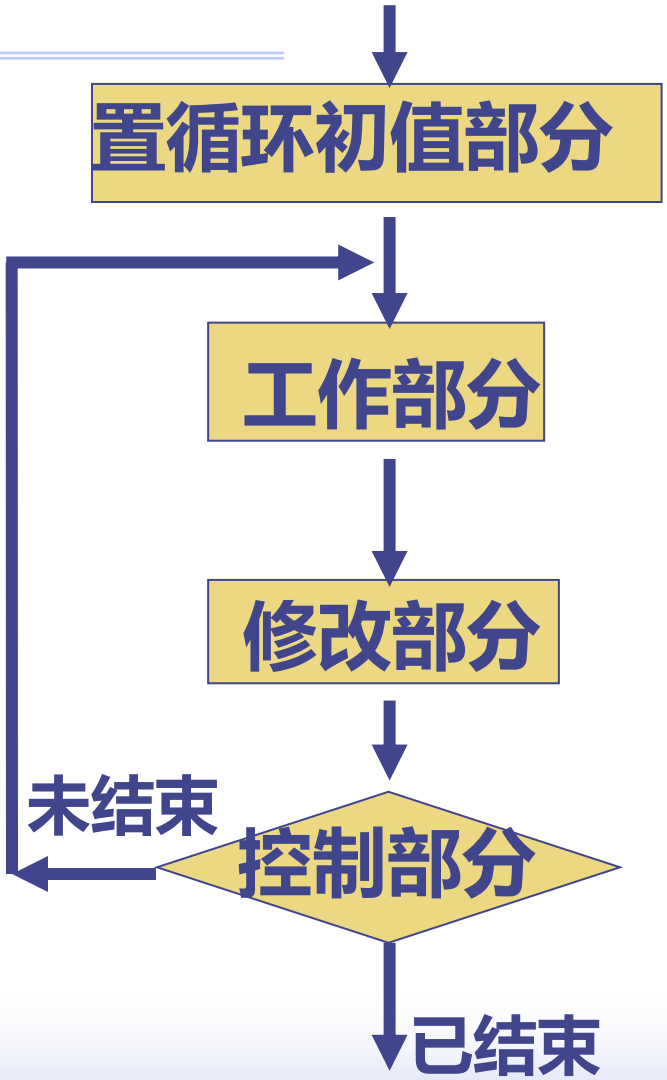
循环指令、循环结构

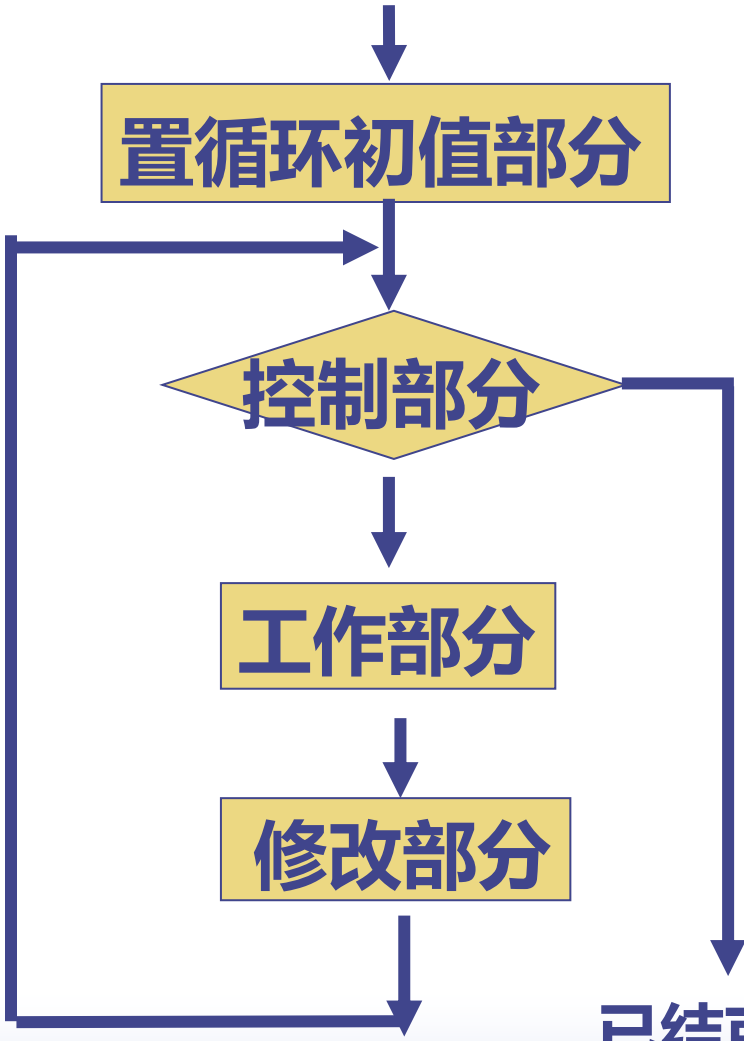
三、学习难点

综合应用前几章的内容，编写和调试程序

第7章 循环程序设计

7.1 循环程序的结构





7.1 循环程序的结构

例：设以BUF为首址的一片单元中，存放了N个有符号字节数据，找出其中的最大数，存放到AL中。

```
BUF   DB   1, -10, 20,-25, 25,50, …
N     =    $ - BUF
```

将BUF视为数组

```
AL ← BUF[0]
FOR (i=1; i<N; i++)
   if (AL < BUF[i])  AL←BUF[i];
```

将EBX与i对应，(EBX)=1,…,N-1

BUF[i] --- BUF[EBX]

7.1 循环程序的结构


```
        MOV   AL, BUF
        MOV   EBX, 1
L1:     CMP   EBX, N
        JAE   EXIT
        CMP   AL, BUF[EBX]
        JGE   L2
        MOV   AL, BUF[EBX]
L2:     INC   EBX
        JMP   L1
EXIT:……
```

```
AL ← BUF[0]
FOR (EBX=1; EBX<N; EBX++)
{
  if (AL < BUF[EBX])
      AL←BUF[EBX];
}
```

最大数放在AL中

7.1 循环程序的结构

例：若BUF中存放一串字数据，求最大值

```
        MOV   AX,     BUF
        MOV   EBX,    1
L1:     CMP   EBX,    N
        JAE   EXIT
        CMP   AX,  BUF[EBX*2]
        JGE   L2
        MOV   AX,  BUF[EBX*2]
L2:     INC   EBX
        JMP   L1
EXIT:……
```

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 ！ ）

(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
```

7.1 循环程序的结构

(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 单重循环程序设计

- 构思好算法
- 用C或者伪代码表达算法思想
- 理清算法和变量空间分配

- 分配好寄存器的用途，建立与变量的对应关系
- 按循环语句的执行过程，写出汇编语言程序

7.2 单重循环程序设计

例1：已知 有n个元素存放在以BUF为首址的字节存储区中，试统计其中负数的个数

例2：以BUF为首址的字节存储区中，存放以0作结束标志的字符串。要求：将串中的小写字母转换成大写字母。

7.2 单重循环程序设计

例3：输入一个数字串， 将其转换成字数据（即二进制形式），以16进制形式显示出来。

（不考虑符号，不考虑溢出）

串数转换 127 -> 31H 32H 37H 00 -> 007FH

(AX) 存放转换的结果，初始为0。

(ESI) 输入缓冲区指针，指向待转换字符

从串左到右依次读入各字符，一边读入一边转换，直到串结束。

设某时刻读到的新数码为 X （字符-30H ），则： (AX) × 10 ＋ X → AX 。

Q：有符号数字串，又如何转换？

7.2 单重循环程序设计

例4：将一个无符号字数转换成10进制形式显示。

例5：将一个有符号字数转换成10进制形式显示。

7.3 多重循环程序设计

- 从外层循环到内层循环一层一层地进行
- 在设计外层循环时，仅把内层循环看成一个处理粗框
- 当内层循环设计完之后，用其替代外层循环体中被视为一个处理粗框的对应部分，构成了一个多重循环。

7.3 多重循环程序设计

例：设以buf为首址的存储区中存放着n个有符号双字数，将其中的数按从小到大的顺序排列。

数组定义为 int buf[n];

```
for  (i=0;i<n-1;i++) {
     将数组中第i小的数，排在 buf[i]的位置。
}
```

在排第i小的数时，buf[0],…,buf[i-1]已经排好，第i小的数只从buf[i]到buf[n-1]中找。

7.3 多重循环程序设计

例：设以buf为首址的存储区中存放着n个有符号双字数，将其中的数按从小到大的顺序排列。

```
for  (i=0;i<n-1;i++) {
     // 从buf[i]到buf[n-1]中找最小的数，
     // 将其排在 buf[i]的位置。
     for (j=i+1;j<n;j++)
         if (buf[i]>buf[j])
              交换 buf[i] 和buf[j]
}
```

7.3 多重循环程序设计


```
for  (i=0;i<n-1;i++) {          // i -> ESI
  for (j=i+1;j<n;j++)           // j -> EDI
       if (buf[i]>buf[j])
            交换 buf[i] 和buf[j]
}
```

分配寄存器的用途:

esi 对应 i，外层循环次数控制

edi 对应 j，内层循环次数控制

eax 表示中间读到的数据

7.3 多重循环程序设计

```
for  (i=0;i<n-1;i++) {        // i -> ESI
  for (j=i+1;j<n;j++)         // j -> EDI
       if (buf[i]>buf[j])   交换 buf[i] 和buf[j]
}
```

```
    MOV  ESI, 0
OUT_LOOP:
    CMP  ESI, n-1   ; n 是符号常量    n=……
    JGE  EXIT
      ……          ; 内循环
    INC  ESI
    JMP  OUT_LOOP
EXIT:
```

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，试统计其中负数的个数存放到变量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？

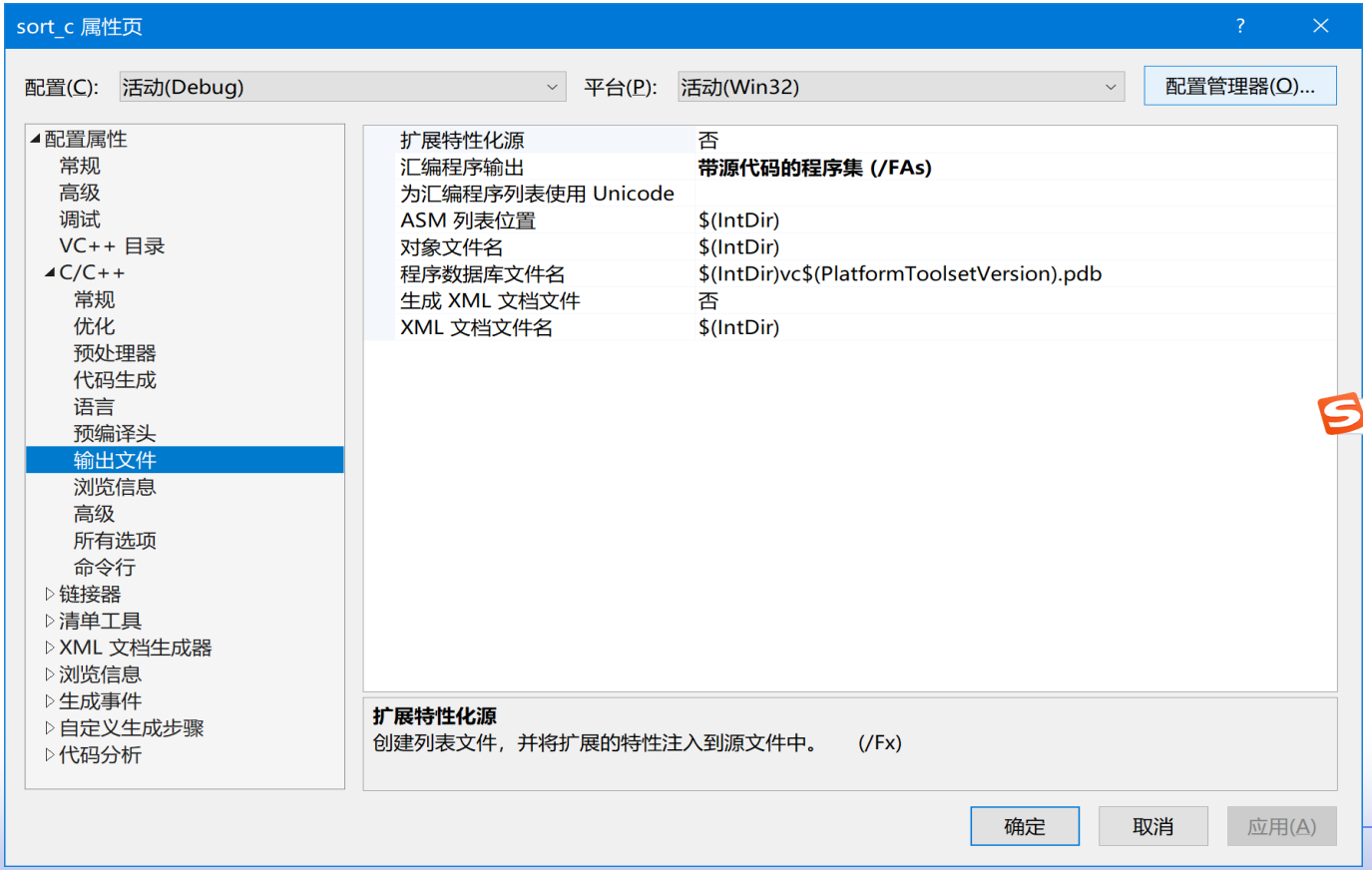
Q：将第(1)行写到lopa:之下，运行结果如何？

Q：交换 (9)-(10)行的顺序，运行结果如何？

7.5 与C循环程序反汇编的比较

不同的编译开关下生成的机器语言程序是不同的。

Debug VS Release；优化开关；编译开关



7.5 与C循环程序反汇编的比较

```
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？
但完成循环，又要执行多少条语句？

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？

7.5 与C循环程序反汇编的比较


```
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] ;
```

7.5 与C循环程序反汇编的比较


```
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！

7.5 与C循环程序反汇编的比较

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;
}
```

7.5 与C循环程序反汇编的比较

華中科技大學

```
        scanf("%s", buf1);
00251090  lea           eax,[buf1]
00251093  push          eax
00251094  push          offset string "%s" (0252100h)
00251099  call          scanf (0251050h)
```

7.5 与C循环程序反汇编的比较

Release 编译优化

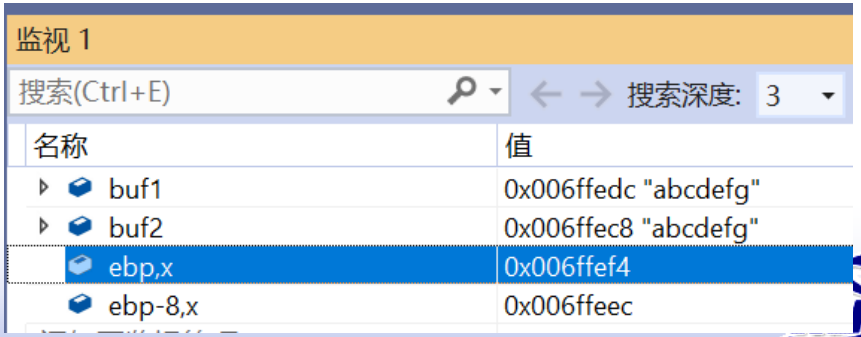
```
0025109E   mov          eax,dword ptr [ebp-8]
002510A1   movups       xmm0,xmmword ptr [buf1]
002510A5   mov          dword ptr [ebp-1Ch],eax
        for (i = 0;i < 20;i++)
                buf2[i] = buf1[i];
        printf("%s\n", buf2);
002510A8   lea          eax,[buf2]
002510AB   push        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



7.5 与C循环程序反汇编的比较

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
```

7.5 与C循环程序反汇编的比较


```
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]
003D10A7  mov           byte ptr [ebp+eax-2Ch],cl
003D10AB  inc           eax
003D10AC  cmp           eax,14h
003D10AF  jl            003D10A3

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

7.6 循环控制伪指令

循环执行伪指令

```
.while   条件表达式1
            语句序列1
   [.break    [.if   条件表达式2]]
   [.continue   [.if   条件表达式3]]
            语句序列2
.endw
```

中断循环伪指令　.break

继续循环伪指令　.continue

7.6 循环控制伪指令

重复执行伪指令

```
.repeat
        语句序列
.until 条件表达式

.repeat
        语句序列
.untilcxz [条件表达式]
```

第7章 循环程序设计

華中科技大學

循环程序的结构

循环控制方法

循环程序设计的基本方法

编写循环程序的注意细节

要求能够熟练编写算法简单的循环程序

C语言循环语句的编译和优化