80X86汇编语言程序设计

80X86 Assembly Language Programming

第4章 程序设计的基本方法

金良海 教授

计算机学院 医学图像信息研究中心

Email: lianghaijin@hust.edu.cn

本章内容

汇编语言的程序设计的基本技术：

(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

(5)子程序设计

学习重点

(1) 转移指令、分支程序的设计

(2) 循环指令、循环结构

(3) 子程序的定义、调用、返回；
主程序与子程序的参数传递

学习难点

(1)无条件转移指令的灵活运用

(2)条件转移指令的正确选择

(3)分支出口的安排与汇合

(4)循环程序的结构和控制循环的方法

(5)CALL与RET指令的执行过程，堆栈操作

(6)综合应用前几章的内容，编写和调试程序

4.1 概述 (1)

设计一个程序要点：

- 认真分析问题的需求，选择好解决方法；
- 针对选定的算法，编写高质量的程序。

高质量的程序：

- 满足设计的要求。
- 结构清晰、简明、易读、易调试。
- 执行速度快。
- 占用存储空间少。

4.1 概述 (2)

汇编语言程序设计的一般步骤:

（1）分析问题，选择合适的解题方法。

（2）分配寄存器和存储区(变量命名)。

（3）绘制程序的流程图。

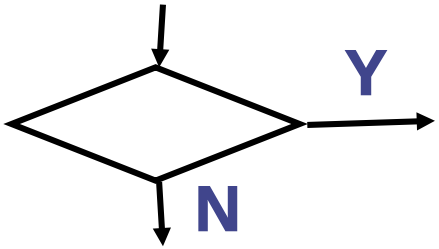
（4）根据流程图编写程序。

（5）静态检查与动态调试。

4.1 概述 (3)

几种框图符号

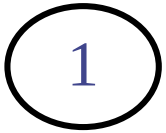
起始、终止框



判断框

处理说明框

子程序调用框



连接框

流向线

4.2 顺序程序设计

華中科技大學

例：从键盘键入0至9中的任一自然数，求其立方值。

```
DATA    SEGMENT
INPUT   DB  'PLEASE INPUT X(0~9): $'
TAB     DW  0, 1, 8, 27, 64, 125,
            216, 343, 512, 729
X       DB  ?
XXX     DW  ?
DATA    ENDS
STACK   SEGEMNT  STACK
        DB  200  DUP(0)
STACK   ENDS
CODE    SEGMENT
        ASSUME CS:CODE,
        DS:DATA, SS:STACK
BEGIB:  MOV  AX, DATA
        MOV  DS, AX
        LEA DX, INPUT
        MOV  AH, 9
        INT    21H
        MOV   AH, 1
        INT    21H
        ADD    AL, 0FH
        MOV   X, AL
        XOR    EBX, EBX
        MOV   BL, AL
        MOV   AX, TAB[2*EBX]
        MOV   XXX, AX
        MOV   AX, 4C00H
        INT    21H
CODE    ENDS
        END  BEGIN
```

4.3 分支程序设计 (1)


```
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: 程序的执行流程是什么？

4.3 分支程序设计 (2)

- 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
    - 段内直接、段间直接
    - 段内间接、段间间接

JMP **

4.3.1 转移指令

1.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 短标号

短标号代表的目的地址与当前(IP)之间的字节距离在-128 和 127之间。

2.无条件转移指令

语句格式： [标号:] JMP 地址表达式

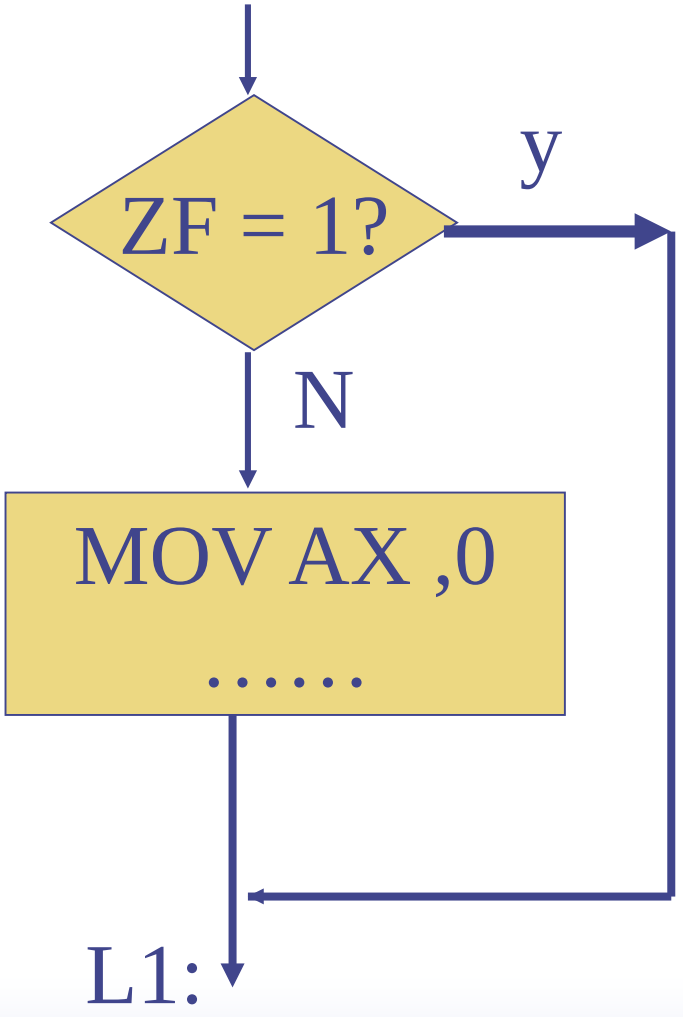
可以在段内和段间转移。

(4.3.1) 1.简单条件转移指令(1)

JZ / JE　　ZF=1时，转移　A == B
JNZ / JNE　　ZF=0时，转移　A != B
JS　　SF=1时，转移　A < 0
JNS　　SF=0时，转移　A > 0
JO　　OF=1时，转移　溢出，转
JNO　　OF=0时，转移　不溢出，转
JC　　CF=1时，转移　进位/借位
JNC　　CF=0时，转移　无进位/借位
JP / JPE　　PF=1时，转移　1的个数为偶数
JNP / JPO　　PF=0时，转移　1的个数为奇数

(4.3.1) 1.简单条件转移指令(2)

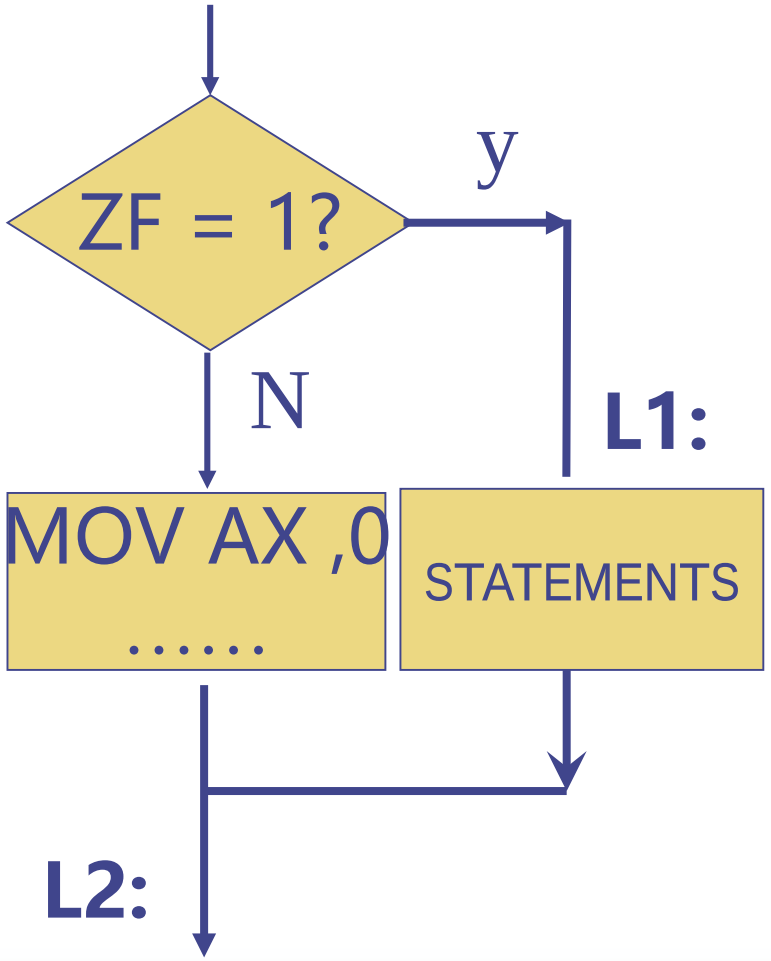

```
    JZ   L1
    MOV  AX , 0
    ……
L1:
```



指令与流程图的对应关系

(4.3.1) 1.简单条件转移指令(3)


```
    JZ   L1
    MOV  AX , 0
    ……
    JMP  L2
L1: STATEMENTS
L2:
```



指令与流程图的对应关系

(4.3.1) 2.无符号数条件转移指令(1)

JA / JNBE　短标号　　**> / !<=**

当 CF=0 且 ZF=0 时，转移

JAE / JNB　短标号　　**>= / !<**

当 CF=0 时，转移

JB / JNAE　短标号　　**< / !>=**

当 CF=1 且 ZF=0时，转移

JBE / JNA　短标号　　**<= / !>**

当 CF=1 时，转移

(4.3.1) 2.无符号数条件转移指令(2)


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

无符号数条件转移指令的理解

将(AX),(BX)中的数据当成无符号数，执行 (AX) – (BX)。若 (AX)>(BX), 则CF一定会为0，ZF=0，转移到L1处。

例1: (AX)= 1234H,　(BX)=0234H

例2: (AX)= 0A234H,　(BX)=0234H

例3: (AX)= 0A234H,　(BX)=09234H

(4.3.1) 3.有符号数条件转移指令(1)

JG / JNLE　短标号　　**> / !<=**

当 SF=OF 且 ZF=0 时，转移

JGE / JNL　短标号　　**>= / !<**

当 SF=OF 时，转移

JL / JNGE　短标号　　**< / !>=**

当 SF≠OF 且 ZF=0时，转移

JLE / JNG　短标号　　**<= / !>**

当 SF≠OF 时，转移

(4.3.1) 3.有符号数条件转移指令(2)


```
    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，执行 (AX) – (BX)。若 (AX)>(BX), 则SF、OF会相等，ZF=0，转移到L1处。

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

(4.3.1) 3.有符号数条件转移指令(3)

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

(4.3.1) 4.无条件转移指令(1)

格式	名称	功能
JMP 标号	段内直接	(IP/EIP)+位移量 → IP/EIP, 立即寻址
JMP OPD	段内间接	(OPD) → IP/EIP 寄存器寻址, 地址表达式
JMP 标号	段间直接	标号的EA → IP/EIP 段首址 → CS, 立即寻址
JMP OPD	段间间接	(OPD) → IP/EIP (OPD+2/4) → CS, 地址

(4.3.1) 4.无条件转移指令(2)

无条件转移指令中，若是间接方式，除了立即数寻址方式外，其它方式均可以使用。

设在数据段中有：

```
BUF   DW   L1   ; L1为标号
```

(1) JMP L1

(2) JMP BUF

(3) LEA BX , BUF
JMP WORD PTR [BX]

(4) MOV BX , BUF
JMP BX

功能等价的转移指令

(4.3.1) 4.无条件转移指令(3)

设有如下程序段：

```
    JZ   L1
    A  ……    ；ZF=0,对应的程序段A
L1: B  ……    ；ZF=1,对应的程序段B
```

若程序段A的长度大于128个字节，怎么办？

```
    JNZ  L0
    JMP  L1
L0: …..       ；A
L1: …..       ；B
```

无条件转移指令的转移范围不受限制。

(4.3.1) 4.无条件转移指令(4)

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！

见程序：c4_108_4.asm

(4.3.1) 4.无条件转移指令(5)

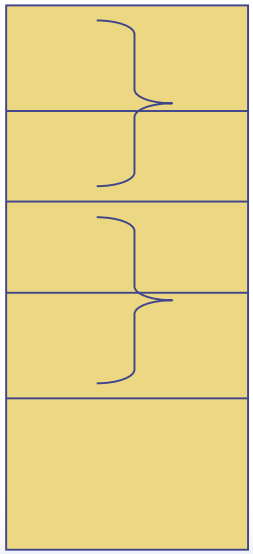
例5：段间直接转移指令 (c4_108_5.asm)

例6：段间间接转移指令：

BUF DD LP ；LP为标号

JMP BUF BUF

(BUF) → IP
(BUF+2) → CS



LP的EA

LP的段址

4.3.2 分支程序设计示例(1)

(1) 选择合适的转移指令；

(2) 为每个分支安排出口;

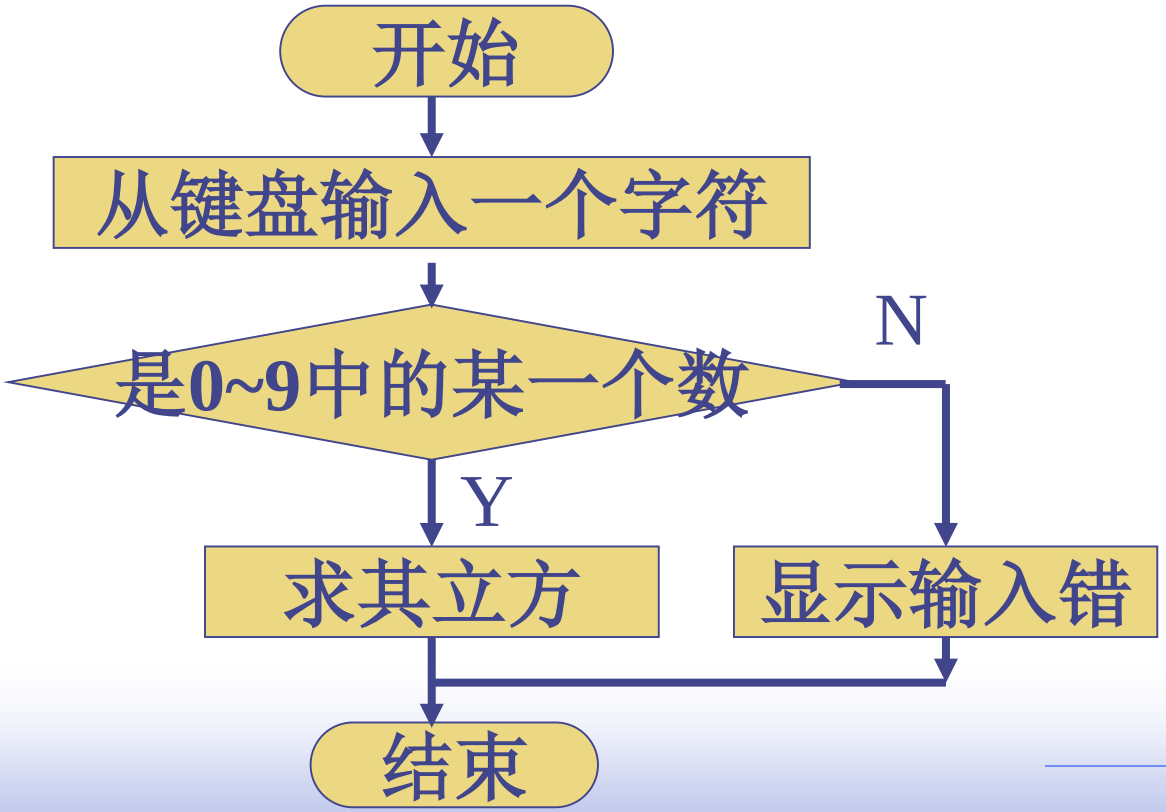
(3) 将分支中的公共部分尽量放到分支前或分支后的公共程序段中；

(4) 流程图、程序对应

(5) 调试时，逐分支检查

4.3.2 分支程序设计示例(2)

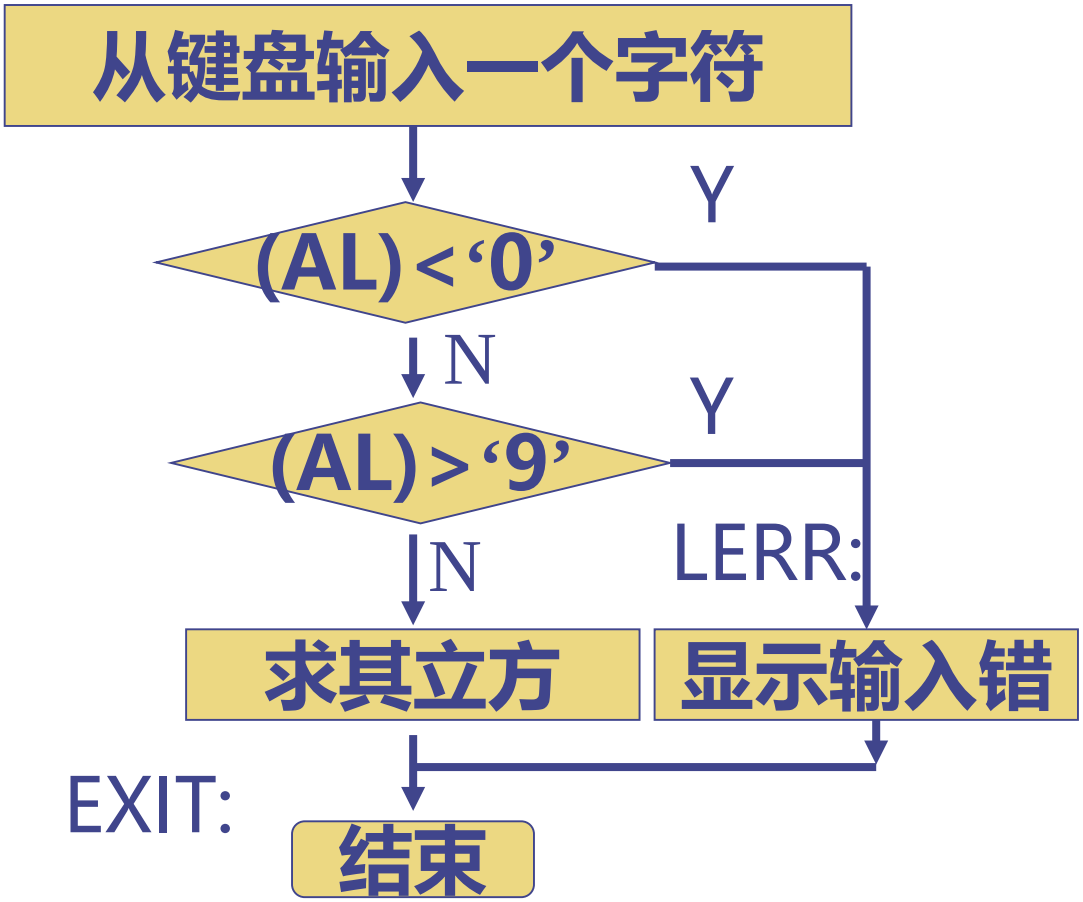
例1：从键盘输入0～9中任一自然数，求其立方值。若输入的字符不是0～9中的数字，则显示“Input Error!”



4.3.2 分支程序设计示例(3)

条件细化

加标号




```
TAB  DW 0, 1, 8, 27, 64, 125, 216, 343, 512, 729
```

c4_108_1.asm

4.3.2 分支程序设计示例(4)

例2：在例1的基础上显示出立方值。

显示立方值，可以使用“输出一个串”调用。
构造一个串表，存放各立方值对应的ASCII串。

如何构造？
如何找到待显示串的起始位置？

```
MSG DB  '  0$',' 1$',' 8$',' 27$',' 64$ '
    DB  '125$','216$','343$','512$','729$'
```

见程序 ：C4_108_2.asm

4.3.2 分支程序设计示例(5)

例3：根据输入的数字，显示对应的信息.

0 ： zero

1 ： first ……

9 ： nine

其它： error

对于不同的输入，输出的串长度不同。

程序的关键：如何根据输入，将对应的待显示的串首址送DX。

见程序: C4_108_3.asm

4.3.2 分支程序设计举例(6)

例4：判断 x 中的内容,

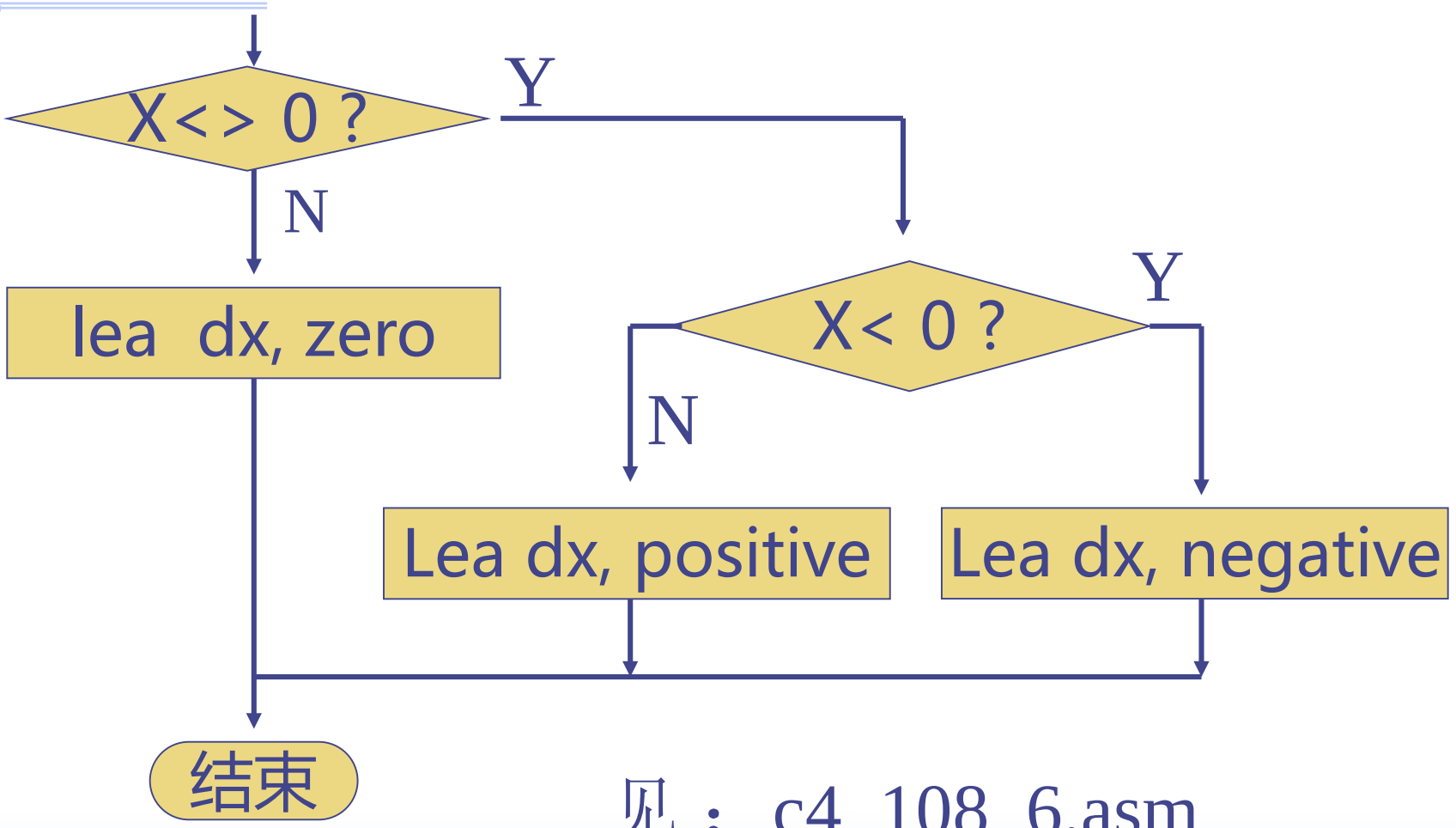
为正,显示 positive > 0 ；

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

4.3.2 分支程序设计举例(7)



见： c4_108_6.asm

4.3.2 分支程序设计举例(8)

例5：有一段程序，实现|(AX)|+(BX)→ Y，试改写之。

```
        OR   AX, AX
        JS   L1
        ADD  AX, BX
        MOV  Y,  AX
        JMP  EXIT
L1:     NEG  AX
        ADD  AX, BX
        MOV  Y,  AX
        JMP  EXIT
EXIT:
```

```
        OR   AX, AX
        JNS  L1
        NEG  AX
L1:     ADD  AX, BX
        MOV  Y,  AX
EXIT:
```

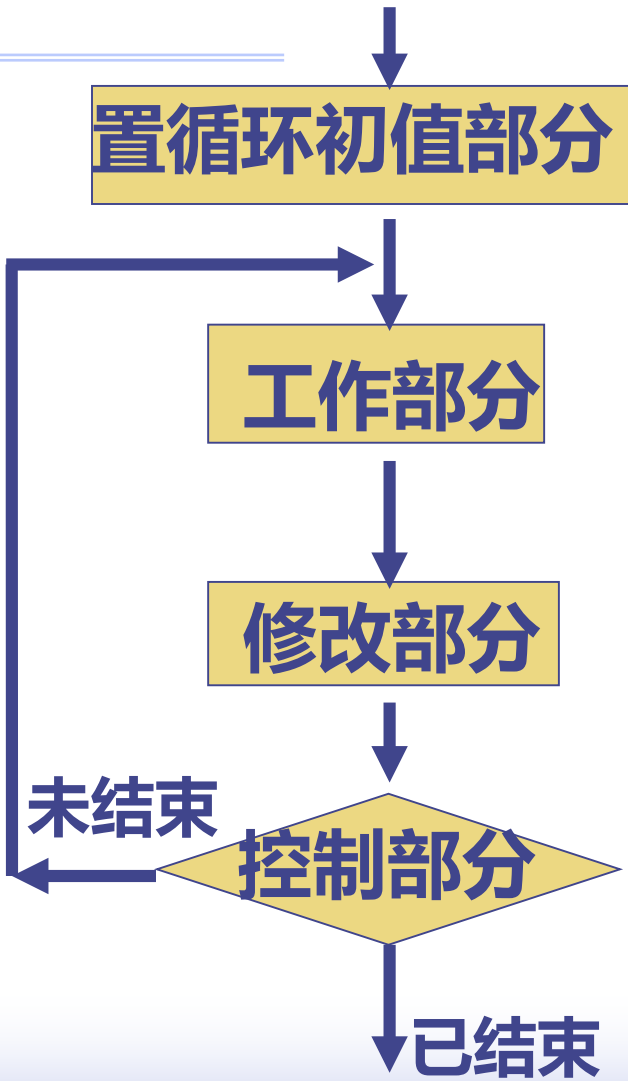
公共部分的简化

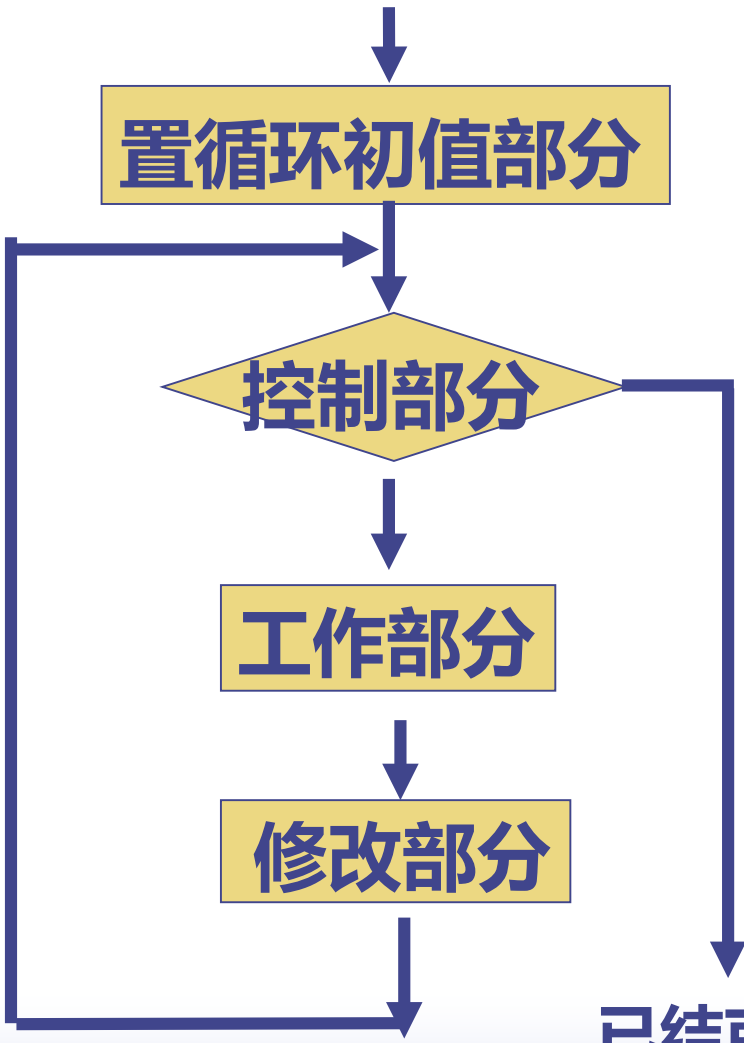
4.4 循环程序设计

1. 循环程序的结构
2. 循环控制方法

   计数控制、条件控制
3. 单重循环程序设计
4. 多重循环程序设计

4.4.1 循环程序的结构和控制方法(1)





4.4.1 循环程序的结构和控制方法(2)

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BX与i对应，(BX)=1,…,N-1

BUF[i] --- BUF[BX]

4.4.1 循环程序的结构和控制方法(3)

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

最大数放在AL中

程序 C4_113J1.asm

4.4.1 循环程序的结构和控制方法(4)

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

程序C4_113J2.asm

4.4.1 循环程序的结构和控制方法(5)

计数控制：循环次数已知时常用

（1）倒计数

将循环次数n ,送入一循环计数器中，每循环一次，计数器减1，直到其值为0

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

4.4.1 循环程序的结构和控制方法(6)

计数控制：循环次数已知时常用

（2）正计数

循环次数n 。 0送入一循环计数器中，每循环一次，计数器加1，直到其值为n。

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

4.4.1 循环程序的结构和控制方法(7)

80X86提供的四种**计数控制**循环转移指令

(1) LOOP 标号

(CX / ECX) - 1 → CX / ECX

若（CX / ECX）不为0, 则转标号处执行。

基本等价于： DEC CX / ECX

JNZ 标号

（LOOP指令对标志位无影响！） C4_115J.asm

执行右边的指令后，

(AX)=? (CX)=?

ZF=? CF=?

```
    MOV AX, 5
    SUB CX, CX
L1: INC AX
    LOOP
    L1
```

4.4.1 循环程序的结构和控制方法(8)

(2) JCXZ 标号 / JECXZ 标号

若 (CX / 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)

(3) LOOPE / LOOPZ 标号

(CX / ECX) -1 → CX / ECX

若 (CX / ECX) 不为0, 且ZF=1,
则转标号处执行。

(本指令对标志位无影响)

4.4.1 循环程序的结构和控制方法(9)

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

C4_115J.asm

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

4.4.1 循环程序的结构和控制方法(10)

(4) LOOPNE /LOOPNZ 标号

(CX / ECX) -1 → CX / ECX

若(CX / ECX)≠0, 且ZF=0,则转标号处执行。

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。

C4_115J.asm

```
        MOV   CX,  10
        MOV   BX,  OFFSET MSG -1
L4 :    INC     BX
        CMP   BYTE PTR [BX], ' '
        LOOPNE   L4
```

4.4.1 循环程序的结构和控制方法(11)

条件控制：

循环次数未知，比较所要求的循环条件是否满足，满足继续循环，否则结束循环。

4.4.1 循环程序的结构和控制方法(12)

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:
```

4.4.1 循环程序的结构和控制方法(13)

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
EXIT:
```

4.4.2 循环程序设计(1)

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

```
L1: ....
    INC   BUF          ; 将 BUF的地址加1
    DEC   CX
    JNZ   L1
```

C4_117.asm

4.4.2 循环程序设计(2)

例2：以BUF为首址的字节存储区中，存放以'$'作结束标志的字符串。显示该串，并要求将其中的小写字母转换成大写字母显示。

见程序： c4_118.asm

4.4.2 循环程序设计(3)

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

（输入的串最长为5个字符，不考虑符号）

(1)输入缓冲区的定义

c4_121_1.asm

(2)转换方法

(AX) 存放转换的结果，初始为0。

(SI) 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。设新字符为 X, 则：

(AX) × 10 + X → AX 。

即读入X后的结果。Try ‘123’ 的转换。

4.4.2 循环程序设计(4)

例4：将一个无符号字节数转换成10进制形式显示。
C4_121_2.asm

例5：将一个有符号字节数转换成10进制形式显示。
C4_121_3.asm

4.4.2 循环程序设计(5)

循环程序设计中应该注意的问题：

比较不同指令次序，程序的运行结果
C4_121_4.asm

4.5 子程序设计

華中科技大學

1. 子程序的概念

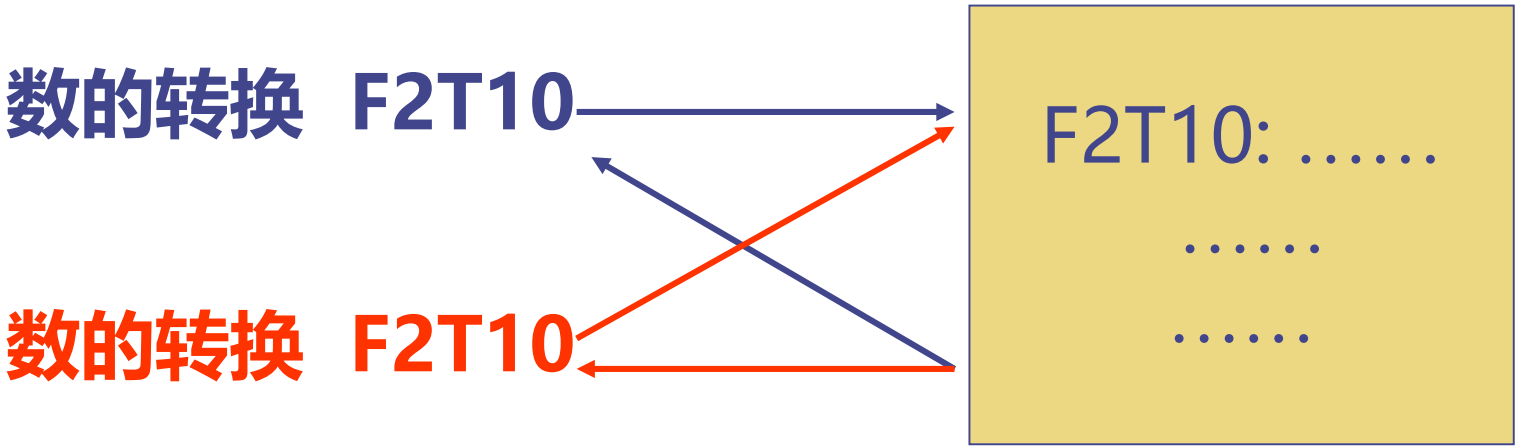
2. 子程序的调用与返回

3. 子程序的定义格式及现场保护方法

4. 主程序与子程序之间的参数传递

4.5.1 子程序的概念(1)

一段程序要被多次使用。例如“将一个二进制数转换成十进制的形式显示出来”。使用该段程序，也没有什么规律，怎么办？



在执行完F2T10后，希望回到调用处继续执行

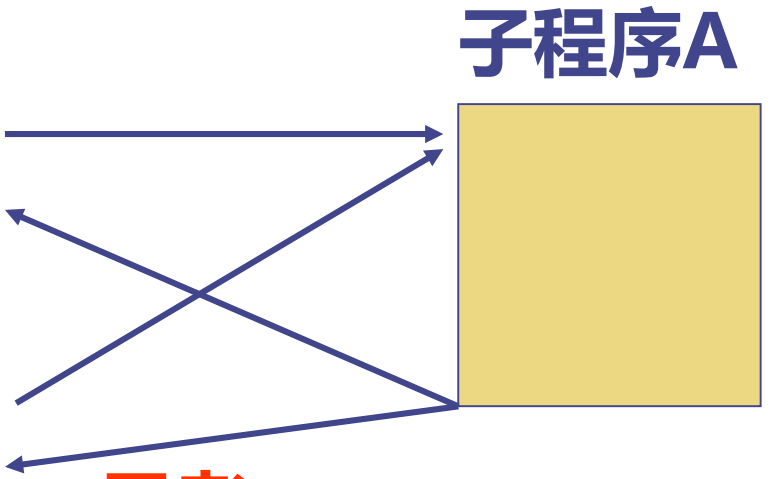
4.5.1 子程序的概念(2)

将经常要使用的或者重复的程序段设计成可供反复调用的独立程序段，在需要时，用控制指令调用它，在执行完后，再返回调用它的程序中继续执行。这样的独立程序段称为**子程序**。

调用子程序的程序称为**主程序** (或称调用程序)

4.5.1 子程序的概念(3)

K : 调用子程序 A
DK: ……
……
J : 调用子程序 A
DJ: ……
……



思考：

转移的本质是什么？

改变CS:IP

断点：转子指令的直接后继指令的地址。

子程序执行完毕，返回主程序的断点处继续执行

如何改变CS:IP,使其按照预定的路线图前进？

4.5.2 子程序的定义

子程序名 PROC NEAR 或者 FAR

……

……

RET [n]

子程序名 ENDP

4.5.3 子程序的调用与返回(1)

调用指令

段内直接　CALL　过程名

段间直接　CALL　FAR PTR 过程名

段内间接　CALL　WORD PTR OPD

段间间接　CALL　DWORD PTR OPD

CPU 要做的工作：

(1) 保存断点

段内：(IP / EIP) →↓ (SP / ESP)

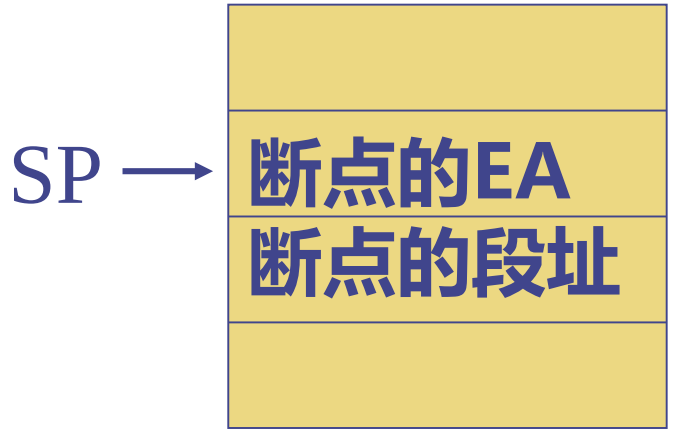
段间：(CS) →↓(SP / ESP)

(IP / EIP) →↓(SP / ESP)

4.5.3 子程序的调用与返回(2)

(1)保存断点

(2)将子程序的地址送 CS, IP



段内：子程序的入口偏移地址→ IP / EIP

段间：子程序的入口偏移地址→ IP /EIP
子程序的段地址→ CS

注意：段间间接调用是如何取地址的。

4.5.3 子程序的调用与返回(3)

返回指令：RET

返回指令的执行过程：

段内返回：↑(SP/ESP) → IP /EIP

段间返回：(1) ↑(SP/ESP) → IP / EIP
(2) ↑(SP/ESP) → CS

特别注意：栈顶必须是主程序的断点地址，否则，运行会出现问题

4.5.3 子程序的调用与返回(4)

返回指令：RET n

消除不再使用的入口参数对堆栈空间的占用

在16位段中，n是2的倍数。

FIRST STEP

段内返回：↑(SP/ESP) → IP /EIP

段间返回：(1) ↑(SP/ESP) → IP / EIP
(2) ↑(SP/ESP) → CS

SECOND STEP

(SP/ESP)+n →SP /ESP

4.5.4 子程序调用现场的保护方法

现场：执行到某一条指令时，各寄存器(包括状态寄存器)的值，存储单元中的内容等。

现场是否需要保护？为什么？

如何保护和恢复现场？

何时保护和恢复现场？

保护和恢复现场可以在主程序中完成，也可在子程序中完成。一般在子程序中完成。

4.5.5 主程序与子程序间的参数传递

例：求一串数据的和.

涉及到三个参数：

数据的起始地址、数据的个数，存放结果的地址

- 使用寄存器传递参数
  例1: c4_134_1.asm
- 约定单元传递参数
  例2: c4_134_2.asm
- 使用堆栈传递参数
  例3: c4_134_3.asm

4.5.6 子程序的嵌套和递归 (1)

使用递归子程序 求 N! 以十进制形式显示结果

c4_141.asm

F(N)=N*F(N-1)

F(1) = 1

与 C4_digui.c 比较

比较c语句编译后的汇编代码

4.5.6 子程序的嵌套和递归 (2)


```
f_jiechen  proc
        cmp  bx,1
        jg      LP
        mov  ax,1
        mov  dx,0
        ret
LP:  dec   bx
        call    f_jiechen
        inc     bx
        mul   bx
        ret
f_jiechen endp
```

(bx)：求阶乘的数

(ax)：计算结果
(dx)

4.5.7 子程序举例 (1)

(1) NUMSTR_1

16位无符号二进制数转换为N (N<=16) 进制的数字串。

参数传递方式：寄存器参数

(AX) = 16位无符号数NUM

(BX) = N

(DS:SI) = 用于保存转换结果的缓冲区.

返回: (AX)=字符个数

算法：NUM除N, 余数是低位, 直到商为0

例：57B H -> 35H 37H 42H (N=16)

-> 31H 34H 30H 33H (N=10, 1403)

-> 32H 35H 37H 33H (N=8, 2573)

-> 31H 30H 31H …… (N=2, 101 0111 1011)

4.5.7 子程序举例 (2)


```
NUMSTR_1   PROC
           MOV    CX , 0
L1:        MOV    DX , 0
           DIV    BX
           PUSH   DX   ;余数压栈
           INC    CX
           CMP    AX , 0
           JNZ    L1
           MOV    BX, CX  ;字符个数
L2:        POP    AX
           CMP    AL, 0AH
           JB     L3
           SUB    AL, 0AH
           ADD    AL, 'A'
           JMP    L4
L3:        ADD    AL, 30H
L4:        MOV    [SI] , AL
           INC    SI
           DEC    CX
           JNZ    L2
           MOV    AX, BX ;字符个数
           RET
NUMSTR_1   ENDP
```

4.5.7 子程序举例 (3)


```
DATA    SEGMENT
BUF     DB  20  DUP(0)
NUM     DW   057BH
DATA    ENDS
CODE    SEGMENT
        ASSUME  CS:CODE,
        SS:STACK,  DS:DATA
BEGIN:  MOV   AX, DATA
        MOV   DS,  AX
        LEA    SI,   BUF
        MOV   BX,  10
        MOV   AX,  NUM
        CALL  NUMSTR_1
        MOV   BX,  AX
        MOV   BUF[BX],  '$'
        MOV   AH, 9
        LEA    DX,  BUF
        INT     21H
        ;;
        MOV   AX, 4C00H
        INT    21H
CODE    ENDS
        END    BEGIN
```

4.5.7 子程序举例 (4)

(1) NUMSTR_2

16位无符号二进制数转换为N (N<=16) 进制的数字串。

参数传递方式：堆栈传递参数

类似C语言的函数说明：

int NUMSTR_2(short int num, short int N, char buf[])

例：57B H -> 35H 37H 42H (N=16)

-> 31H 34H 30H 33H (N=10, 1403)

-> 32H 35H 37H 33H (N=8, 2573)

-> 31H 30H 31H …… (N=2, 101 0111 1011)

4.5.7 子程序举例 (5)


```
DATA    SEGMENT
BUF     DB  20  DUP(0)
NUM     DW   057BH
DATA    ENDS
CODE    SEGMENT
        ASSUME  CS:CODE,
        SS:STACK,  DS:DATA
BEGIN:  MOV    AX, DATA
        MOV    DS,  AX
        PUSH   AX
        PUSH   OFFSET  BUF
        PUSH   0016H
        PUSH   NUM
        CALL  FAR PTR  NUMSTR_2
        ADD    SP,   8
        MOV   BX,  AX
        MOV   BUF[BX],  '$'
        MOV   DX,  OFFSET BUF
        MOV   AH,  9
        INT     21H
        ;;
        MOV   AX,  4C00H
        INT    21H
CODE    ENDS
        END    BEGIN
```

4.5.7 子程序举例 (6)


```
NUMSTR_2    PROC  FAR
        PUSH    BP
        MOV     BP, SP
        MOV     AX,  [BP+0CH]
        MOV     DS,  AX
        MOV     SI,   [BP+0AH]
        MOV     BX,  [BP+08H]
        MOV     AX,  [BP+06H]
        ;;
        MOV     CX , 0
L1:     MOV     DX , 0
        DIV     BX
        PUSH    DX   ;余数压栈
        INC     CX
        CMP     AX , 0
        JNZ     L1
        MOV   BX,  CX  ;字符个数
        INC   BX   L2:   POP   AX
        CMP   AL,  0AH
        JB    L3
        SUB   AL, 0AH
        ADD   AL, 'A'
        JMP   L4
L3:     ADD   AL，30H
L4:     MOV   [SI] ，AL
        INC   SI
        DEC   CX
        JNZ   L2
        MOV   AX, BX   ;字符个数
        POP   BP
        RET
NUMSTR_2    ENDP
```

4.5.7 子程序举例 (7)

(3) 数制转换综合例子

从键盘输入一个10进制字符串, 将它按N进制显示出来。

设计步骤：

(1) 用10号功能调用从键盘输入一个10进制字符串；
(2) 设计并调用子程序 STRNUM_10 将该10进制字符串转换为2进制数;
(3) 调用子程序 NUMSTR_2 将这个2进制数转换为N进制字符串；
(4) 显示转换好的字符串。

举例：N=16, 从键盘输入的10进制串为 39608：
33 39 36 30 38 => 9AB8 H (39608) => 39 41 42 38 => 显示

4.5.7 子程序举例 (8)


```
DATA    SEGMENT
BUF     DB   5, 0, 10 DUP(0)
N       DW   16
DATA    ENDS
CODE    SEGMENT
        ASSUME CS:CODE,
        SS:STACK, DS:DATA
BEGIN:  MOV  AX, DATA
        MOV  DS, AX
        MOV  AH, 10
        LEA  DX, BUF
        INT  21H
        MOV  CL, BUF+1
        CMP  CL, 0
        JZ   EXIT
        LEA  SI, BUF+2
        CALL STRNUM_10
        PUSH DS
        PUSH OFFSET BUF
        PUSH N
        PUSH AX
        CALL FAR PTR NUMSTR_2
        ADD  SP, 8
        MOV  BX, AX
        MOV  BUF[BX], '$'
        MOV  DX, OFFSET BUF
        MOV  AH, 9
        INT  21H
        ;;
EXIT:   MOV  AX, 4C00H
        INT  21H
CODE    ENDS
        END  BEGIN
```

4.5.7 子程序举例 (9)

将10进制字符串转换为16位的2进制数. 使用寄存器传递参数:

(DS:SI) = 10进制字符串**的首地址**
(CX) = 字符串中字符的个数
返回: (AX) = 转换结果

算法演示:

31H 32H 33H =>
31 => 01 = (AX)
32 => AX * 10 + 2 => (AX) = 12
33 => AX * 10 + 3 => (AX) = 123

如果输入的是N进制数字符串, 如何转换为2进制数?

```
STRNUM_10   PROC
            MOV   AX , 0
            MOV   BX, 10
L1:         MUL   BX        ;DX, AX
            MOV   DL, [SI]
            SUB   DL, '0'
            MOV   DH, 0
            ADD   AX, DX
            INC   SI
            LOOP  L1
            RET
STRNUM_10   ENDP
```

4.5.7 子程序举例 (10)

(4) 趣味子程序

编写子程序，使其能够显示CALL指令下面的字符串。

```
.386
STACK    SEGMENT USE16 STACK
         DB 200 DUP(0)
STACK    ENDS
CODE     SEGMENT USE16
         ASSUME  CS:CODE, SS:STACK
BEGIN:   CALL  DISPLAY
         DB     'HELLO', 0DH, 0AH, 0
         CALL  DISPLAY
         DB    'VERY GOOD', 0DH, 0AH, 0
         MOV  AH, 4CH
         INT   21H

DISPLAY  PROC
        ……
DISPLAY  ENDP

CODE     ENDS
         END  BEGIN
```

4.5.7 子程序举例 (11)


```
DISPLAY   PROC
          POP     BX
LOOPA:    MOV     DL, CS:[BX]
          CMP     DL, 0
          JZ      EXIT
          MOV     AH, 2
          INT     21H
          INC     BX
          JMP     LOOPA
EXIT:     INC     BX
          PUSH    BX
          RET
DISPLAY   ENDP
```

4.5.7 子程序举例 (12)


```
STACK   SEGMENT USE16 STACK
        DB 200 DUP(0)
STACK   ENDS
CODE    SEGMENT USE16
        ASSUME  CS:CODE,SS:STACK
BEGIN:  CALL  DISPLAY
        DB    'HELLO', 0DH, 0AH, 0
        CALL  DISPLAY
        DB    'VERY GOOD', 0DH, 0AH, 0
DISPLAY PROC
        ……
DISPLAY ENDP
        MOV   AH, 4CH
        INT    21H
CODE    ENDS
        END   BEGIN
```

分析程序的执行过程，指出其问题.

C4_141_3.asm

4.5.7 子程序举例 (13)

(5) 机器码程序　　C4_ADD_1.ASM

```
CODE  SEGMENT  USE16
    ASSUME   CS:CODE
    MSG  DB   48H, 45H, 4CH, 4CH, 4FH, 20H    ;HELLO WORLD$
         DB   57H, 4FH, 52H, 4CH, 44H, 24H
START:
    DB   8CH,  0C8H              ; MOV  AX, CS
    DB   8EH,  0D8H              ; MOV  DS, AX
    DB   0BAH, 00H, 00H          ; LEA  DX, MSG
    DB   0B4H, 09H               ; MOV  AH, 9
    DB   0CDH, 21H               ; INT  21H
    DB   0B4H, 4CH               ; MOV  AH, 4CH
    DB   0CDH, 21H               ; INT  21H
CODE  ENDS
    END START
```

4.5.7 子程序举例 (14)

(6) 怪异程序-1　　C4_ADD_3.ASM

```
CODE    SEGMENT  USE16
        ASSUME  CS:CODE
MSG     DB  48H, 45H, 4CH, 4CH, 4FH, 20H
        DB  57H, 4FH, 52H, 4CH, 44H, 24H
CODE1   DB  8CH, 0C8H, 8EH, 0D8H, 0BAH, 00H, 00H
        DB  0B4H, 09H, 0CDH, 21H, 0B4H, 4CH, 0CDH, 21H
LEN  =  $ - CODE1
START:  PUSH   CS
        LEA     BX, CODE1
        PUSH   BX
        RETF
CODE    ENDS
        END START
```

4.5.7 子程序举例 (15)

(6) 怪异程序-2　　C4_ADD_2.ASM

```
CODE    SEGMENT  USE16
        ASSUME  CS:CODE
MSG     DB  48H, 45H, 4CH, 4CH, 4FH, 20H
        DB  57H, 4FH, 52H, 4CH, 44H, 24H
CODE1   DB  8CH, 0C8H, 8EH, 0D8H, 0BAH, 00H, 00H
        DB  0B4H, 09H, 0CDH, 21H, 0B4H, 4CH, 0CDH, 21H
LEN = $-CODE1
START:  MOV   AX,  CS
        MOV   DS,  AX
        MOV   ES,  AX
        MOV   CX,  LEN
        LEA    SI,  CODE1
        LEA    DI,  CODE2
        CLD
        REP   MOVSB
CODE2:  DB    10H  DUP(0)
CODE    ENDS
        END  START
```

4.6 程序设计中的注意事项

- 正确地分配数据存储器
- 合理的分配寄存器
- 设计逻辑明晰的算法
- 选用合适的指令与寻址方式
- 合理安排程序的格式

第4章 总结

- 汇编语言程序设计的一般步骤
- 算法框图的画法
- 无条件转移指令、简单条件转移指令、比较转移指令
- 分支程序设计的方法
- 循环控制指令、循环程序设计
- 子程序设计