一、本章的学习内容

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

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!

二、本章的学习重点

(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

三、本章的学习难点

(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

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。

一个高质量的程序不仅要满足设计的要求，而且还应尽可能实现以下几点：

- 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

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）根据具体问题，确定输入输出数据的格式。

（3）分配存贮区并给变量命名(包括分配寄存器)。

（4）绘制程序的流程图，即将解题方法和步骤用程序流程图的形式表示出来。

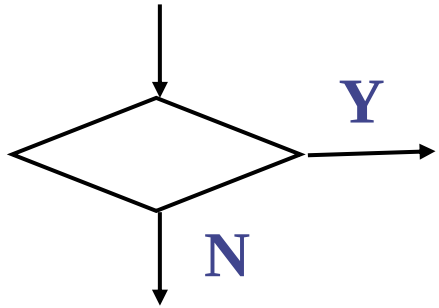
（5）根据流程图编写程序。

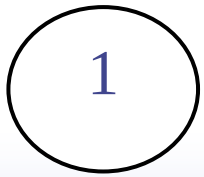
（6）静态检查与动态调试

4.1 概述

华中科技大学

几种框图符号



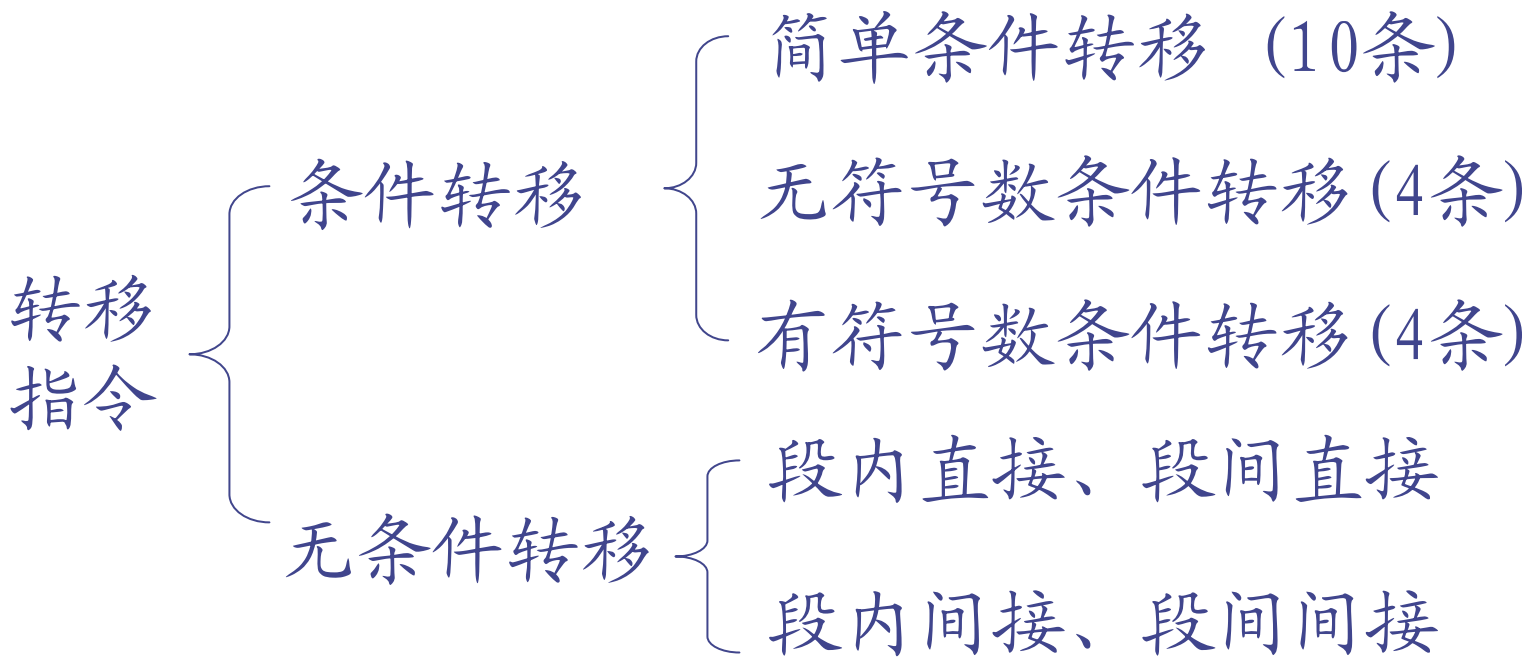


4.2 顺序程序设计

书 P145，作业 4.1

第二次上机题：
目的：复习、巩固第三章学习的指令

4.3 分支程序设计


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

4.3.1 简单条件转移

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

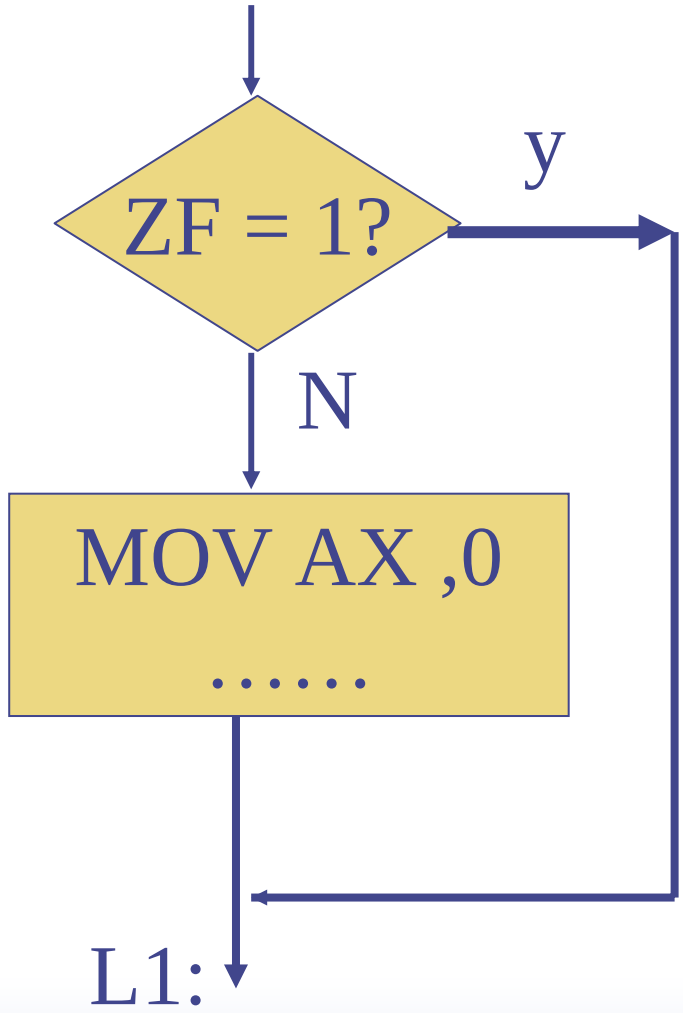
短标号是一个标号。该标号代表条件成立时，想转移到的目的地址。若该目的地址与当前(IP)之间的字节距离在-128 和 127之间，则称该标号为短标号。

4.3.1 简单条件转移

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

4.3.1 简单条件转移

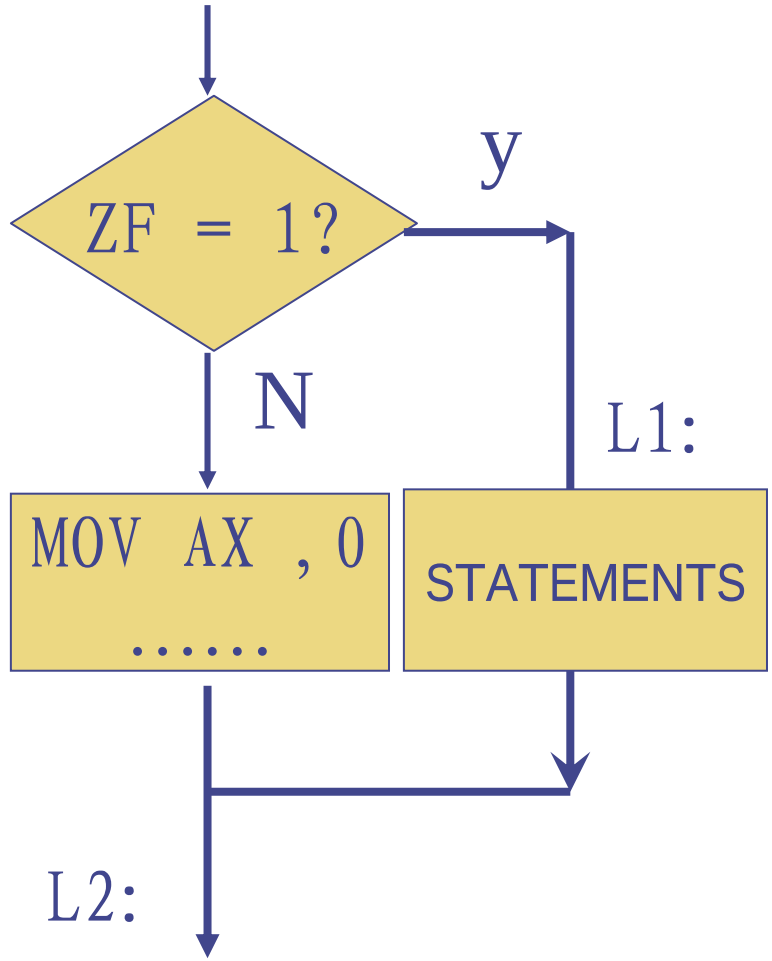
```
      JZ    L1
      MOV   AX , 0
      ......
L1:
```



指令与流程图
的对应关系

4.3.1 简单条件转移

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



指令与流程图的对应关系

4.3.2 无符号数条件转移指令

JA / JNBE　　短标号

当 CF=0 且 ZF=0时，转移

JAE / JNB　　短标号

当 CF=0 或者 ZF=1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 或者 ZF=1时，转移

4.3.2 无符号数条件转移指令

```
        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

4.3.3 有符号数条件转移指令

JG / JNLE　短标号

当 SF=OF 且 ZF=0时，转移

JGE / JNL　短标号

当 SF=OF 或者 ZF=1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 或者 ZF=1时，转移

4.3.3 有符号数条件转移指令

有符号数条件转移指令的理解

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

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

4.3.3 有符号数条件转移指令

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

4.3.4 分支程序设计示例

(1) 选择合适的转移指令;

(2) 为每个分支安排出口;

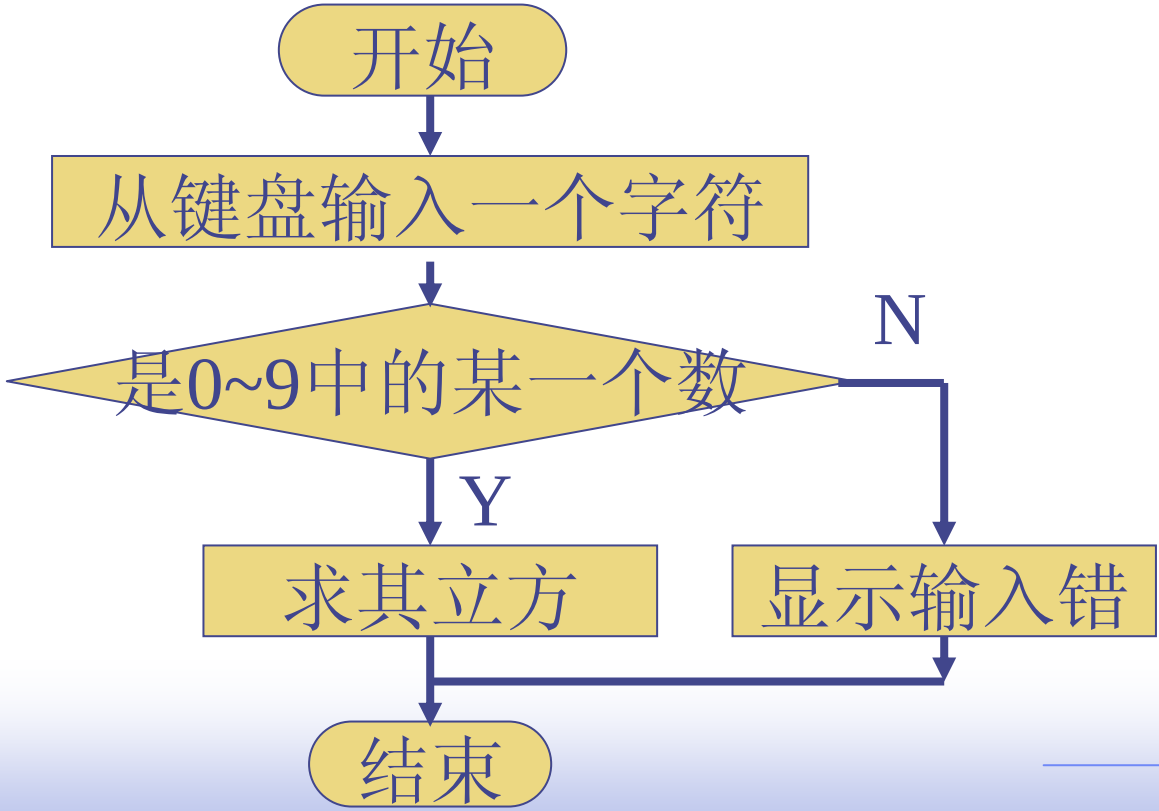
(3) 将分支中的公共部分尽量放到分支前或分支后的公共程序段中;

(4) 流程图、程序对应

(5) 调试时，逐分支检查

4.3.4 分支程序设计示例

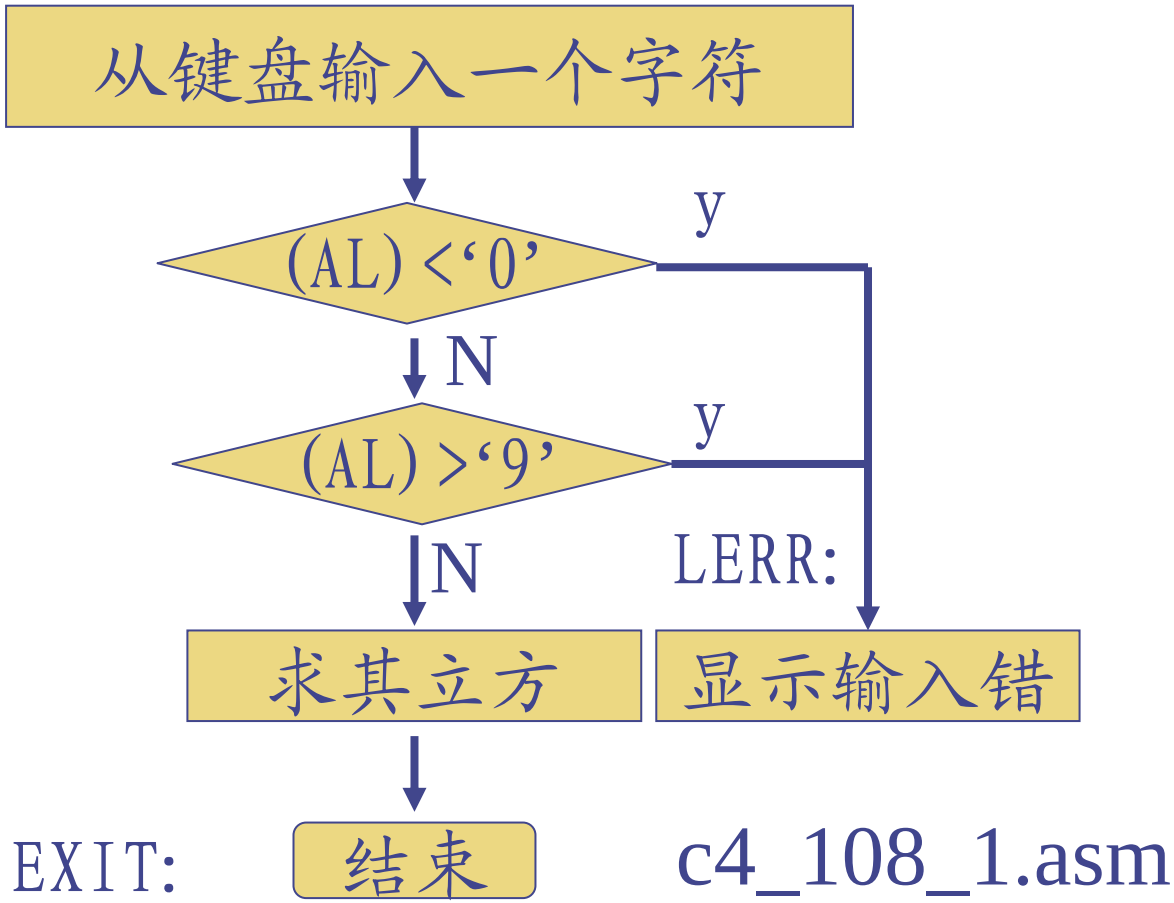
例1：从键盘输入0~9中任一自然数，求其立方值。若输入的字符不是0~9中的数字，则显示“Input Error!”



4.3.4 分支程序设计示例

条件细化

加标号



c4_108_1.asm

4.3.4 分支程序设计示例

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？

见程序 ： C4_108_2.asm

4.3.4 分支程序设计示例

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

4.3.4 分支程序设计举例

例4：判断 x 中的内容，

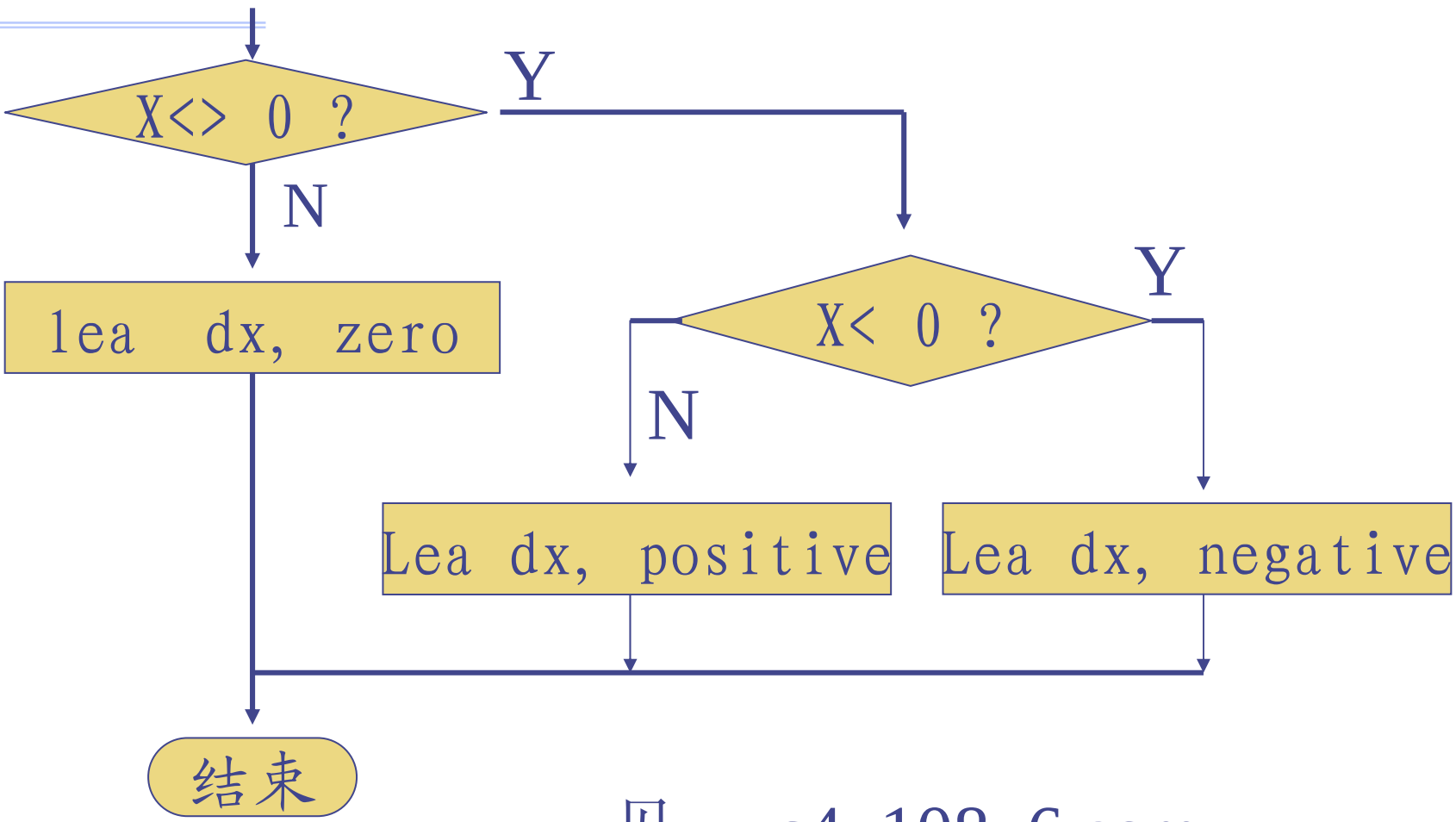
为正，显示 positive > 0 ；

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

4.3.4 分支程序设计举例



见： c4_108_6.asm

4.3.4 分支程序设计举例

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

4.3.5 无条件转移指令

格式	名称	功能
JMP 标号	段内直接	(IP/EIP)+位移量 → IP/EIP
JMP OPD	段内间接	(OPD) → IP/EIP
JMP 标号	段间直接	标号的EA → IP/EIP 段首址 → CS
JMP OPD	段间间接	(OPD) → IP/EIP (OPD+2/4) → CS

4.3.5 无条件转移指令

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

4.3.5 无条件转移指令

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
L1:   .....             ;   B
L0:   .....             ;   A
```

4.3.5 无条件转移指令

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

若程序段A,B的长度均大于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。

4.3.5 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！

见程序：c4_108_4.asm

4.3.5 无条件转移指令

例5：段间直接转移指令　（c4_108_5.asm）

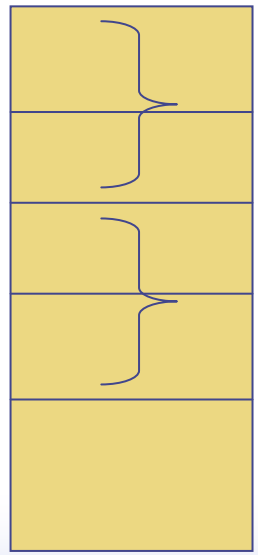
例6：段间间接转移指令：

```
BUF    DD    LP    ;    LP为标号

JMP    BUF
```

(BUF) → IP

(BUF+2) → CS



4.3.6 条件转移伪指令

条件控制流伪指令　P323

```
.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
```

C4_108_7.ASM

条件表达式:

(1) 关系运算

==

!=

>

>=

<

<=

(2) 逻辑运算

&&

||

4.3.6 条件转移伪指令


```
X    db    -5
```

为什么结果不正确？

观察目标程序，发现其用的是无符号数比较转移指令。即masm6 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
```

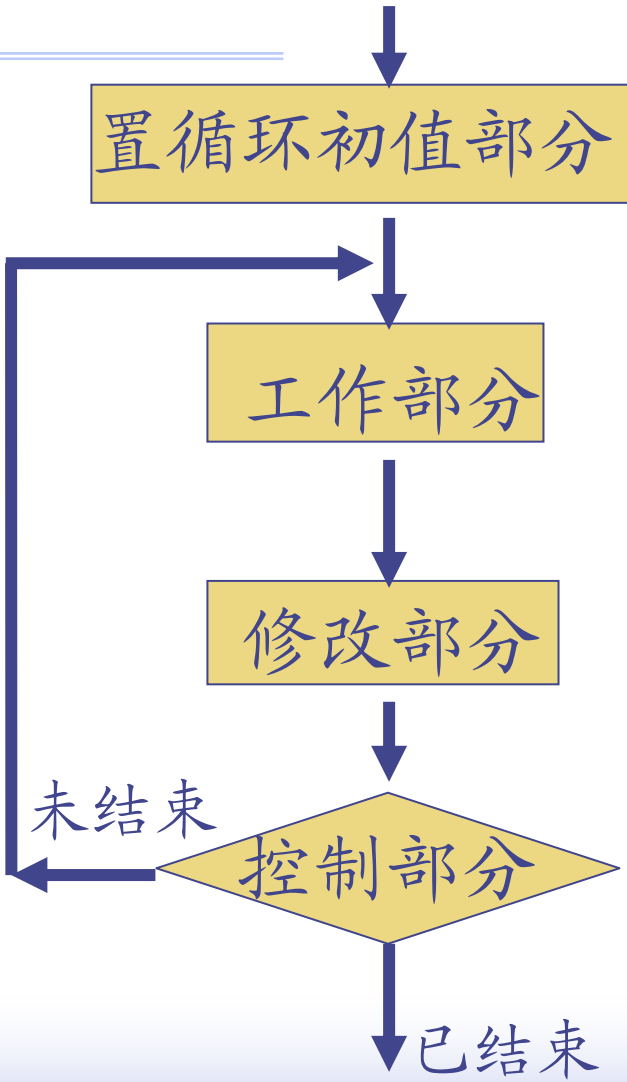
同样，有SWORD, SDWORD……

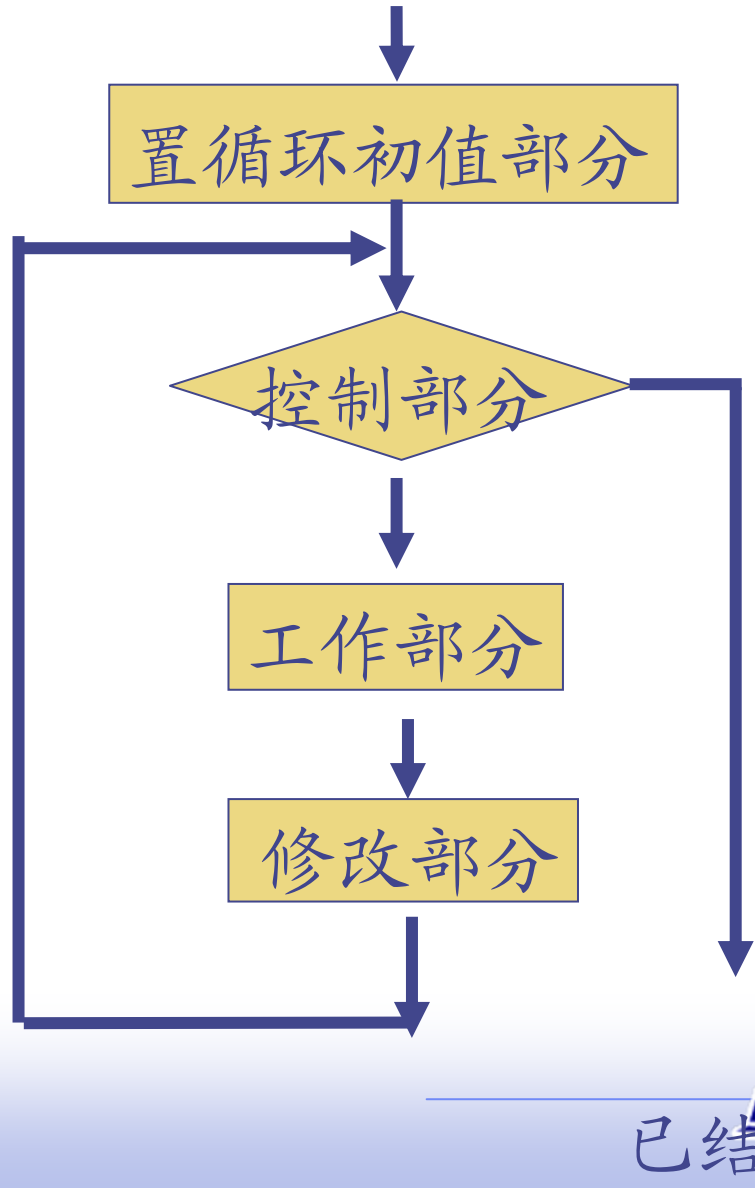
4.4 循环程序设计

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





4.4.1 循环程序的结构和控制方法

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

（接上页）

```
        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:.....
```

若BUF中存放的字数据，程序应作何修改？

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

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

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

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

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

(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)

4.4.1 循环程序的结构和控制方法

(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。

(等于或为0循环转移指令, 本指令对标志位无影响)

例: 判断以BUF为首址的10个字节中是否有非0字节。 C4_115J.asm

4.4.1 循环程序的结构和控制方法

(3) 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。 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

(4) LOOPNE /LOOPNZ 标号

(CX / ECX) -1 → CX / ECX

若CX / 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。 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

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

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

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
```

4.4.2 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

循环 n次;

有些同学认为在循环体中，每次访问下一个单元，有:

```
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

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

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

（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

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

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

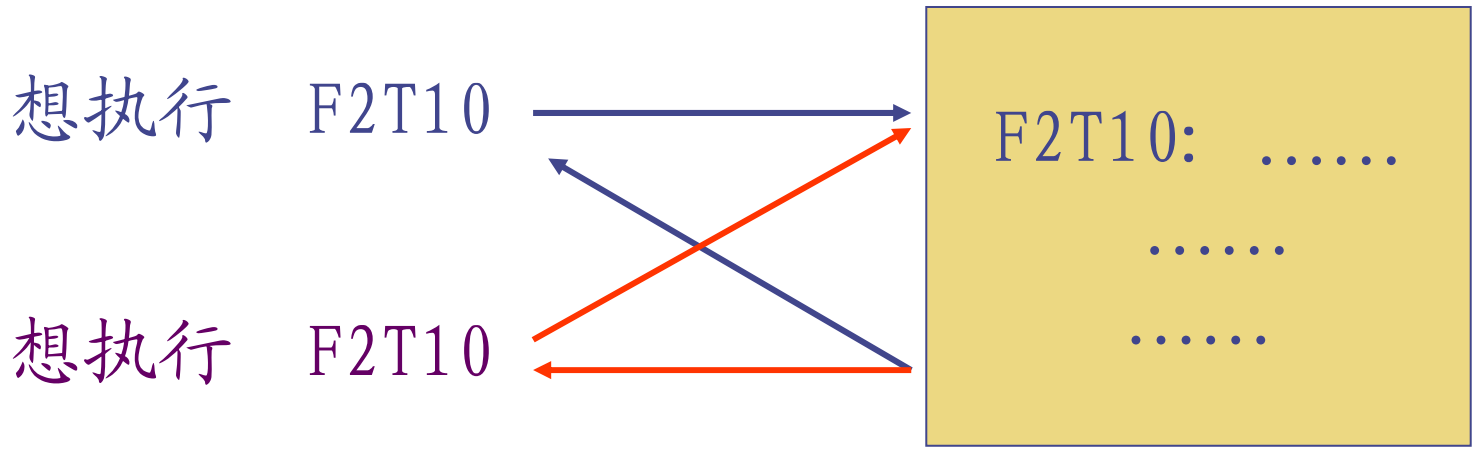
4.5 子程序设计

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

在编写程序时，常常出现如下情况:

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?



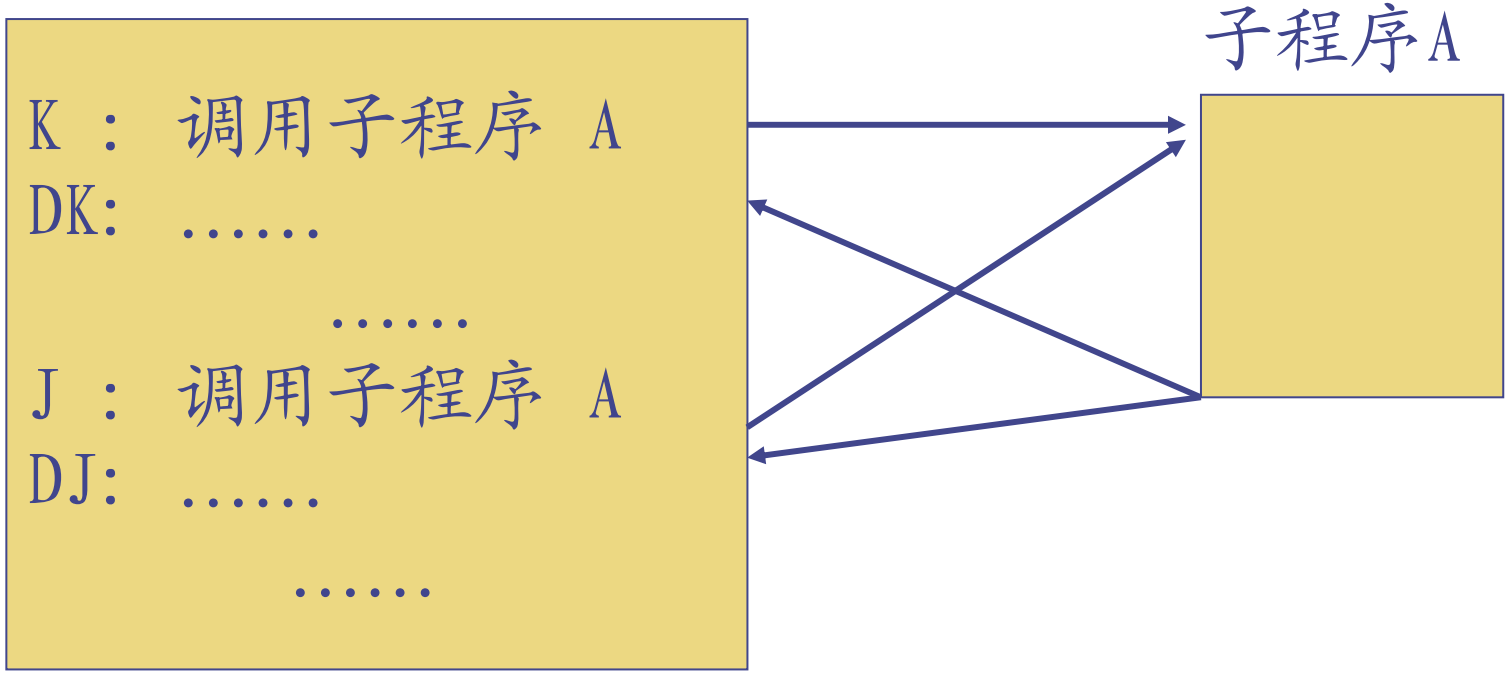
在执行完F2T10后，希望回到调用处继续执行

4.5.1 子程序的概念

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



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

4.5.2 子程序的定义


```
子程序名    PROC   NEAR 或者 FAR
            ……
            ……
子程序名    ENDP
```

4.5.3 子程序的调用与返回

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

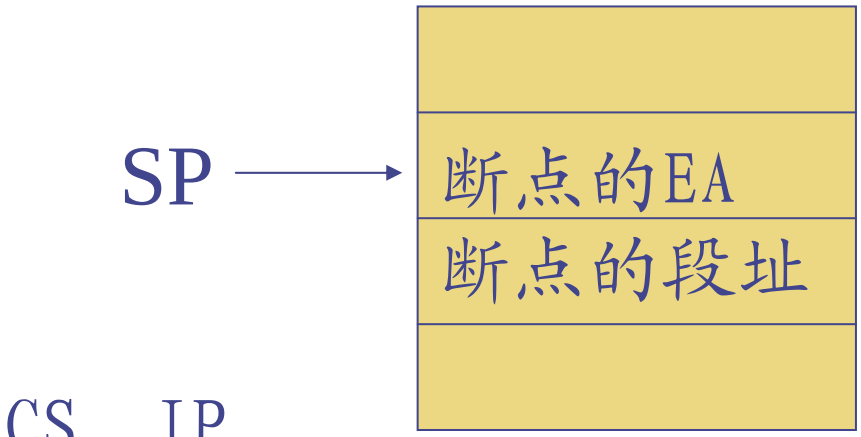
直接: (IP / EIP) → ↓ (SP / ESP)

间接: (CS) → ↓ (SP / ESP)

(IP / EIP) → ↓ (SP / ESP)

4.5.3 子程序的调用与返回

(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

返回指令: (1) RET
(2) RET n

返回指令的执行过程:

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

段内返回: ↑(SP) → IP /EIP

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

4.5.4 子程序调用现场的保护方法

现场：执行到某一条指令时，各寄存器的值。
存储单元中的内容等等。

为什么要保护现场？

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。

如何保护和恢复现场？

4.5.6 主程序与子程序间的参数传递

- 使用寄存器传递参数

  例1: 求一串数据的和.

  c4_134_1.asm

- 约定单元传递参数

  例2: 求一串数据的和.

  c4_134_2.asm

- 使用堆栈传递参数

  例3: 求一串数据的和.

  c4_134_3.asm

4.5.6 主程序与子程序间的参数传递

例4：对于两组数据分别进行排序

C4_134_5.asm

例5：输入数字串,转换成 BCD码串

C4_134_6.asm

4.5.6 主程序与子程序间的参数传递

华中科技大学

补充内容:

结构定义伪指令及用法

程序: c4_s_k1.asm

c4_s_k2.asm

c4_s_k3.asm

使用结构+堆栈传递参数

c4_s_k4.asm

4.5.6 子程序的递归

使用递归子程序 求 N! 以十进制形式显示结果

c4_141.asm

与 C4_digui.c 比较

比较c语句编译后的汇编代码

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

第四章 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
- 地址表法
- 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