

汇编语言程序设计


Assembly Language Programming

第四章

基本汇编语言程序设计

4.1. 顺序程序设计

- 
- ♥ 1. 从键盘读入一个2位的十进制的正整数，存入BL中。



```
MOV AH,1
INT 21H      ; 等待输入十位
AND AL,0FH   ; 取数
MOV BL,10
MUL BL       ; 乘以10
MOV BL,AL    ; 暂存
MOV AH,1     ; 等待输入个位
INT 21H
AND AL,0FH   ; 取数
ADD BL,AL    ; 相加
```


4.2 分支程序设计

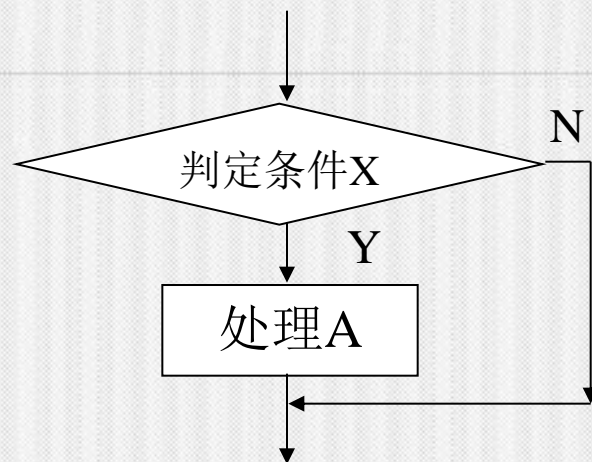
♥ 分支程序

- ❖ 使用条件转移指令来完成分支。一个可以产生两分支。JMP不会产生分支。

♥ 分支程序基本结构

- ❖ 单分支、双分支、多分支

1 单分支结构



```
JX Next
JMP Done
Next:
    ; Handle A
Done:
    ; Switch has done
```

```
JNX Next
    ; Handle A
Next:
    ; Switch has done
```


; 计算AX的绝对值

例1 求绝对值P91

☀ Good

cmp ax, 0

jns nonneg ;分支条件: $AX \geq 0$

neg ax ;条件不满足, 求补

nonneg: mov result, ax ;条件满足

; 计算AX的绝对值

☀ Bad

cmp ax, 0

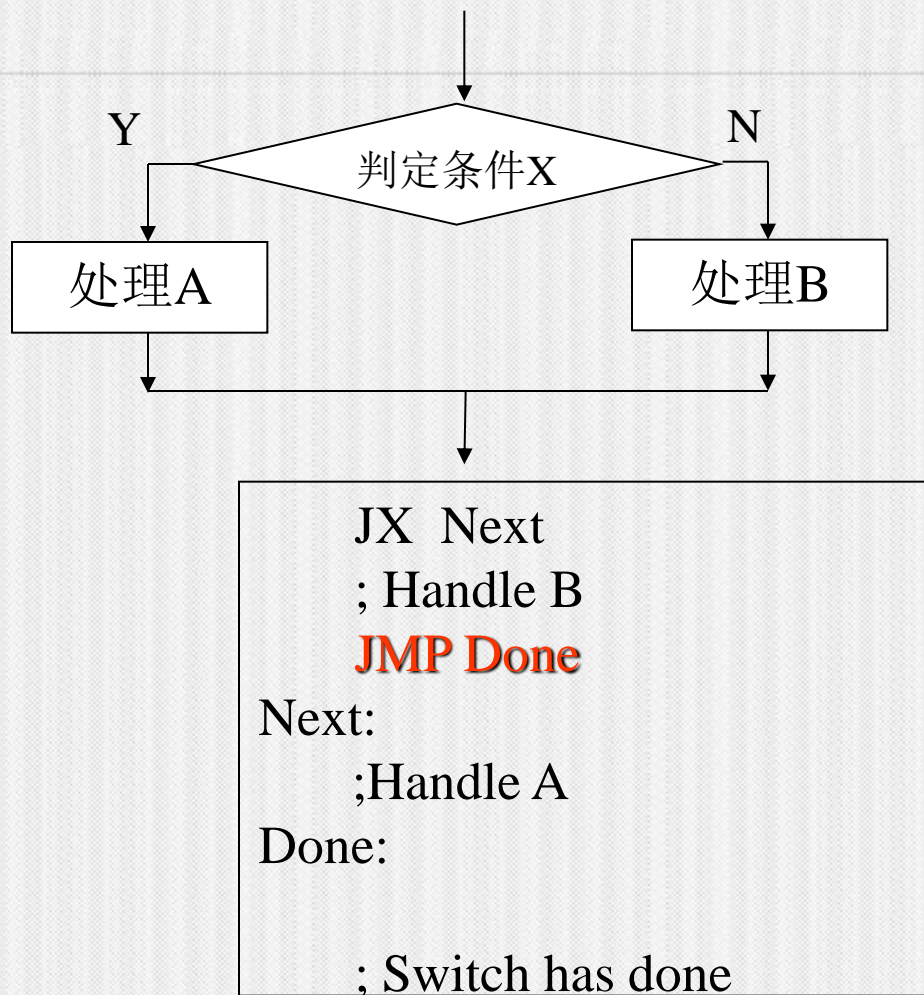
j1 yesneg ;分支条件: $AX < 0$

jmp nonneg

yesneg: neg ax ;条件不满足, 求补

nonneg: mov result, ax ;条件满足

2 双分支结构



例2 显示BX最高位 (P92)

```
shl bx, 1      ;BX最高位移入CF
jnc one        ;CF = 0, 即最高位为0, 转移
mov dl, '1'
;CF = 1, 即最高位为1, DL ← '1'
jmp two        ;一定要跳过另一个分支体
one: mov dl, '0' ;DL ← '0'
two: mov ah, 2
int 21h        ;显示
```


例3 判断2次方程有无实根 (P93)

;a、b、c均为字节变量: -127~127

```
mov al, _b
```

```
imul al
```

```
mov bx, ax      ;BX中为 $b^2$ 
```

```
mov al, _a
```

```
imul _c
```

```
mov cx, 4
```

```
imul cx      ;AX中为 $4ac$  (DX无有效数据?)
```


例3 判断有无实根—2/2

`cmp bx, ax` ;比较二者大小

`jge yes` ;条件满足?

`mov tag, 0`

;第一分支体: 条件不满足, $tag \leftarrow 0$

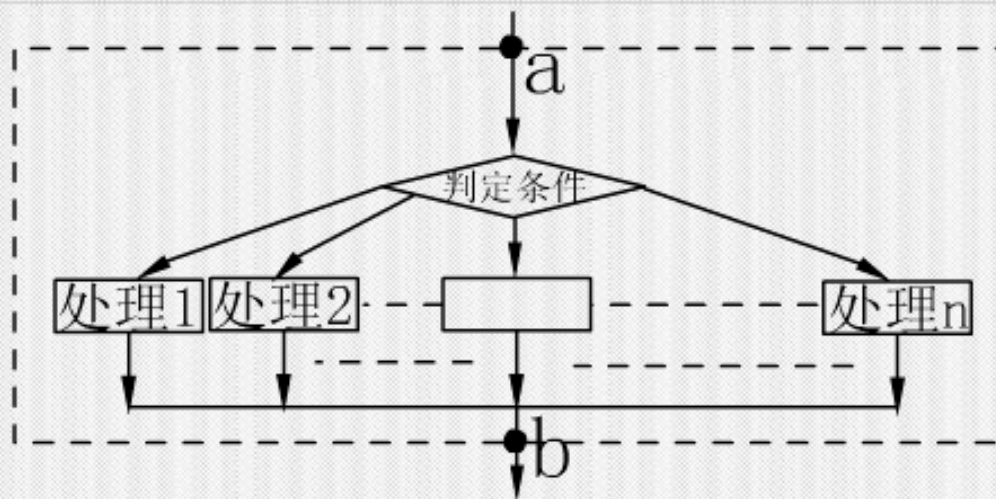
`jmp done` ;跳过第二个分支体

`yes: mov tag, 1`

;第二分支体: 条件满足, $tag \leftarrow 1$

`done: .exit 0`

3 多分支结构



♥ 多分支程序处理方法：


- ❖ 1. 多条件转移指令实现 (if ... else if ... else if ...)
- ❖ 2. 地址表 (Switch ... Case...)

例 4 (if ... else if ...else if ...)

♥ 在内存Score缓冲区中存放有100个学生的成绩数据，为无符号字节数。分别统计各个分数段的人数，分别存储在NOTPASSED、PASSED、GOOD、BETTER、BEST。

```
CMP SCORE[BX],90
JB NEXT
INC BEST
    ;if >= 90 , Best!
JMP DONE
NEXT:
    CMP SCORE[BX],80
    ;If got here, must <90!
JB NEXT1
INC BETTER
    ;if >=80 , Better
JMP DONE
...
```


Switch case



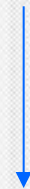
```
switch(表达式)
{
case:常量1: do sth1;break;
.....
case:常量n: do sthn;break;
}
```

分析问题



多分支条件

转化为表达式



离散常量

地址表 (Switch case)

♥ `JMP WORD PTR[2]`

❖ DS:2中的字内容, 是跳转的位置

♥ `JMP X`

❖ X是字变量, 则等价于`JMP WORD PTR[X地址如2]`

♥ `X DW 200, 202`

❖ 则`JMP X`跳到偏移地址是200的位置

❖ 则`JMP X+2`跳到偏移地址是202的位置

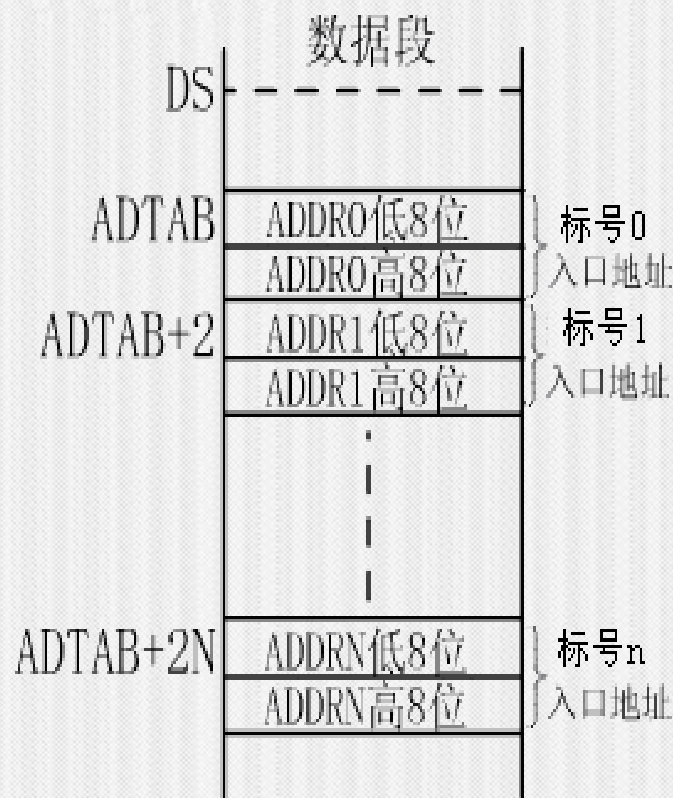
♥ `X DW L0, L1`

❖ 则`JMP X`跳到标号L0的位置

❖ 则`JMP X+2`跳到到标号L1的位置

地址表 (Switch case)

- ♥ 首先,设计分支条件,使第n个分支映射为数n
- ♥ 然后,在存储器的数据段中定义一张入口地址表
 - ❖ AddressTable DW L0,L1,L2,.....
- ♥ 最后, 根据条件转入n分支。
 n号分支地址 =
 [入口地址表首地址 + $n \times 2$]
JMP AddressTable[2*n]



例 5（地址表）

♥ 根据输入的数字1 - 7，分别显示相应的英文星期名。

Data segment

```
ADDRTABLE DW L0,L1,L2,L3 ;  
S0 DB 'MONDAY$'  
S1 DB 'TUESDAY$'  
S2 DB 'WEDNESDAY$'  
S3 DB 'THURSDAY$'  
;.....
```

Data ends

```
.....  
MOV AH,1  
INT 21H  
SUB AL,31H  
SHL AL,1  
MOV AH,0  
MOV BX,AX  
JMP ADDRTABLE[BX]  
L0:  
LEA DX,S0 .....
```


例4.4 根据键盘输入的1~8数字转向不同的处理程序

start1: mov dx, offset msg ;提示输入数字

mov ah, 9

int 21h

mov ah, 1 ;等待按键

int 21h

cmp al, '1' ;数字 < 1?

jb start1

cmp al, '8' ;数字 > 8?

ja start1

and ax, 000fh ;将ASCII码转换成数值



```
dec ax
```

```
shl ax, 1 ;等效于add ax, ax
```

```
mov bx, ax
```

```
jmp table[bx]
```

```
; (段内) 间接转移: IP←[table+bx]
```

```
start2: mov ah, 9
```

```
int 21h
```

```
.exit 0
```

```
displ: mov dx, offset msg1 ;处理程序1
```

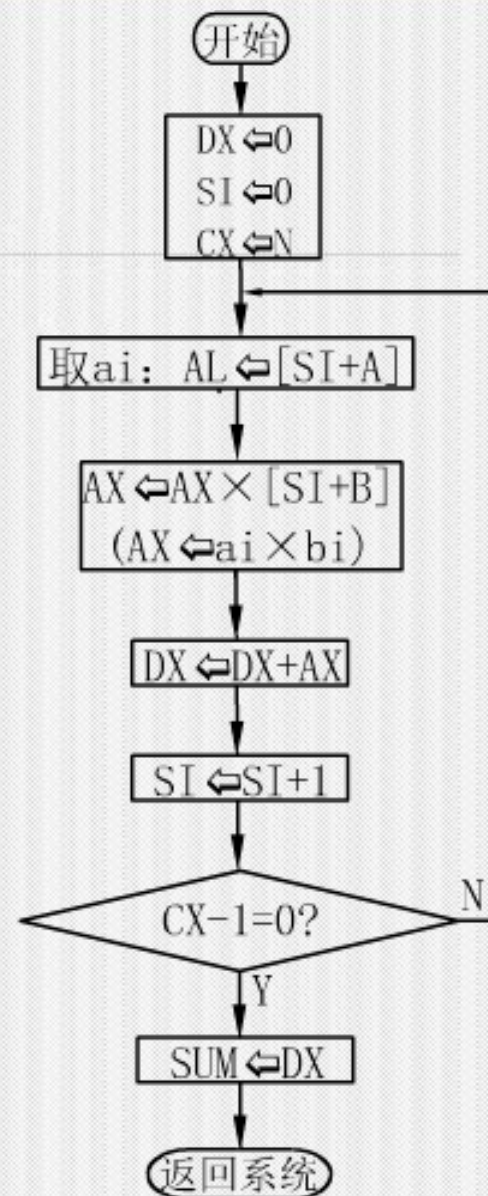
```
jmp start2
```

```
...
```


4.3 循环程序设计

♥ 例6：设数据段中有两个有符号数字字节数组A和B，编程计算：

$$SUM = \sum_{i=1}^{10} A_i B_i = A_1 B_1 + A_2 B_2 + \dots + A_{10} B_{10}$$



例4.7 将一个字符串中所有大写字母改为小写

```

mov bx, offset string
again:  mov al, [bx]      ;取一个字符
        or al, al        ;是否为结尾符0
        jz done          ;是, 退出循环
        cmp al, 'A'      ;是否为大写A~Z
        jb next          ;小于B, 即小写字母, 直接跳至next
        cmp al, 'Z'      ;大于Z, 即小写字母, 直接跳至next
        ja next          ;大于A, 即小写字母, 直接跳至next
        or al, 20h       ;是, 转换为小写字母 (使D5=1)
        mov [bx], al     ;仍保存在原位置
next:   inc bx
        jmp again        ;继续循环
done:   .exit 0

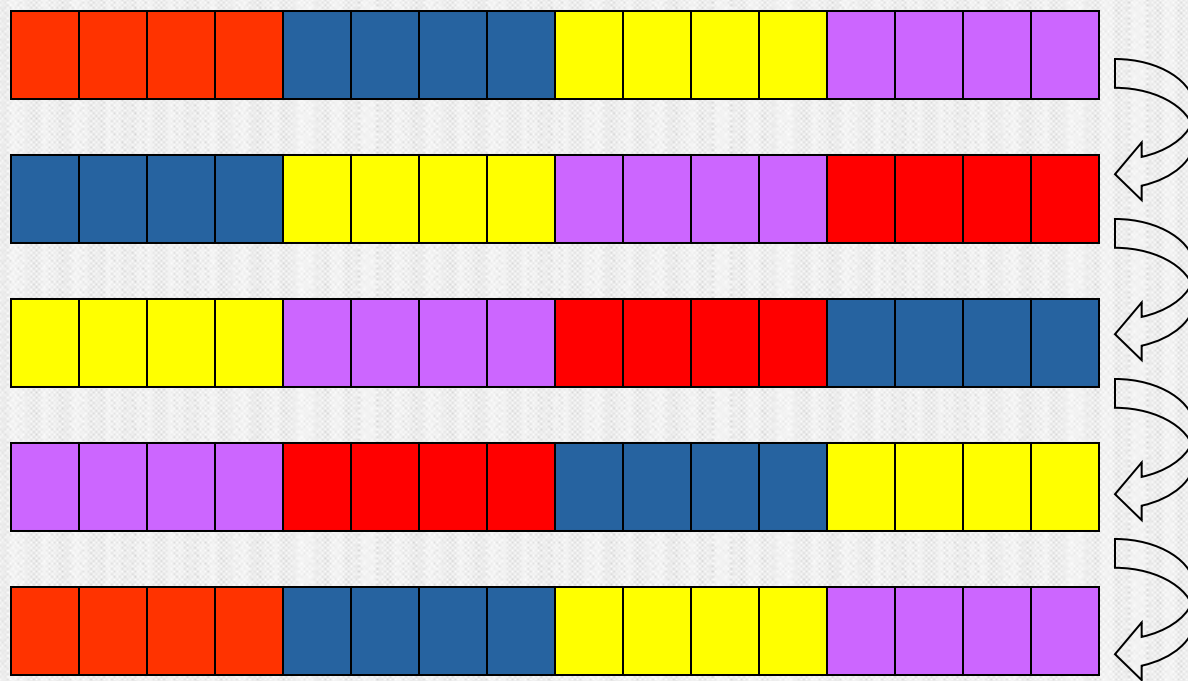
```

条件控制循环
利用标志退出

大小写字母仅 D5位不同

例7 把BX中的二进制数以十六进制的形式显示在屏幕上。

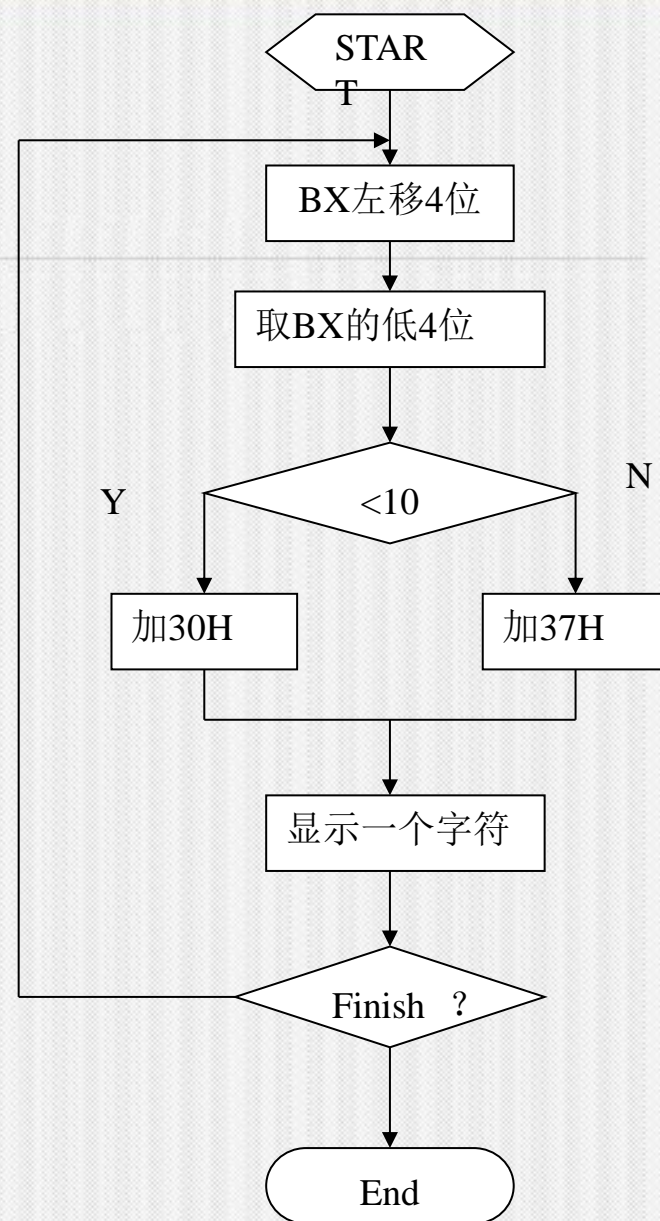
BX



rotate:

```
mov ch, 4
mov cl, 4
rol bx, cl
mov al, bl
and al, 0fh
```

```
cmp al, 0ah
jl next
add al, 37h ; 'A'-'F'
jmp done
add al, 30h ; '0'-'9'
mov dl, al
mov ah, 2
int 21h
dec ch
jnz rotate
```

next:
done:

♥例8：数据段中Score缓冲区中有5个学生的成绩（字节型）。将各自的名次算出填充到Rank缓冲区中。

```
MOV CL,5
MOV SI,0
```

AGAIN:

```
MOV AL,SCORE[SI]
MOV DI,0
MOV CH,5
```

GOON:

```
CMP AL,SCORE[DI]
JAE NEXT
INC RANK[SI]
```

NEXT:

```
INC DI
DEC CH
JNZ GOON
```

```
INC SI
DEC CL
JNZ AGAIN
```

