4.2 8086/8088 指令系统

2. 减法指令

8086/8088共有5条减法指令

①不带CF的减法指令（SUB）

格式：SUB DST , SRC

功能：(1) DST ← (DST) - (SRC)

(2) 根据差设置6个状态标志

②带CF的减法指令（SBB）

格式：SBB　DST，SRC

功能：　(1) DST ← (DST)-(SRC)-CF

(2) 根据差设置6个状态标志

4.2 8086/8088 指令系统

③减1指令（DEC）

格式：DEC　DST

功能：　(1) DST ← (DST)-1,

(2)根据差设置除CF以外的6个状态标志，对CF无影响

4.2 8086/8088 指令系统

④取负指令（NEG）

格式：NEG　DST

功能：　DST ←0-(DST)，

NEG指令完成的是特殊的减法操作，它的被减数一定为零，因此它是一条单操作数指令。

4.2 8086/8088 指令系统

实际它是将目的操作数的值取负。若操作数的原值为一正数，那么，执行该指令后，其值变为该数的负数的补码；而若操作数的原值为一负数（补码表示），那么，执行该指令后，其值变为该数所对应的正数。该指令将正常影响各标志位，并且CF表示最高位产生的借位。指令中的目的操作数可以采用除立即数以外的各种寻址方式。

4.2 8086/8088 指令系统

例1:　　若(AL)=03H，则CPU执行

NEG　AL 指令后，

-3的补码

(AL)=0FDH

```
      0 0 0 0 0 0 0 0 B
  -   0 0 0 0 0 0 1 1 B
  ---------------------
    [1]1 1 1 1 1 1 0 1 B
```

结果为-3的补码

AF=1
PF=0
ZF=0
SF=1
CF=1
OF=0

4.2 8086/8088 指令系统

若(AL)=0FDH，则CPU执行

NEG AL 指令后，

+3的补码

(AL)=03H

```
      0 0 0 0 0 0 0 0 B
  -   1 1 1 1 1 1 0 1 B
  ---------------------
[1]0 0 0 0 0 0 1 1 B
```

结果为+3的原码(补码)

应用场合：常常用于求某数的绝对值的场合

4.2 8086/8088 指令系统

⑤比较指令（CMP）

格式：CMP　DST，SRC

功能：　(1) (DST) —(DST)

根据差设置6个状态标志位

4.2 8086/8088 指令系统

与减法指令不同的是所产生的两数之差并不取代目的操作数，因而指令执行后，仅仅改变了标志寄存器的内容，两操作数的值保持不变。

4.2 8086/8088 指令系统

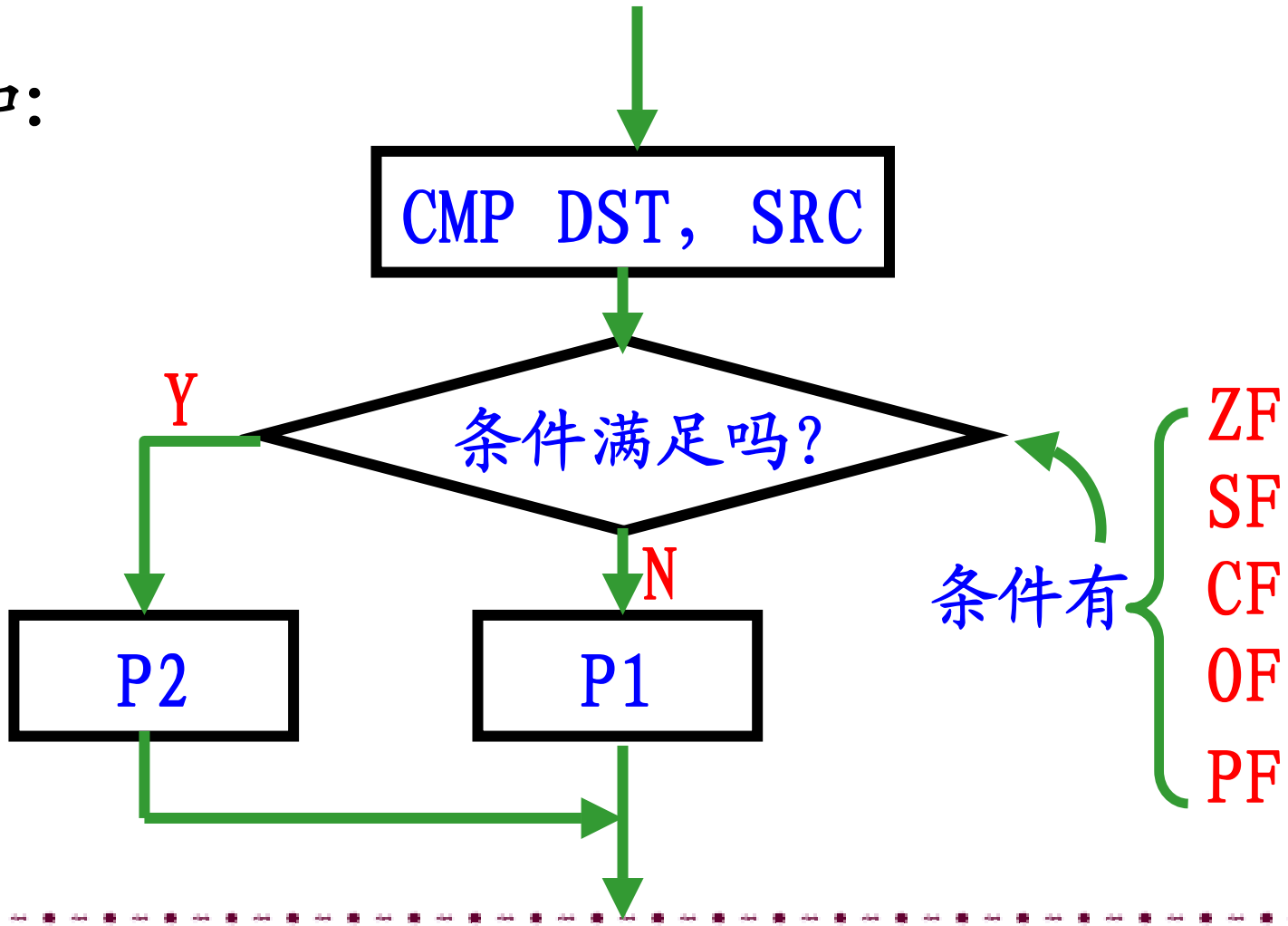
应用场合：

- 两操作数参加比较，根据比较结果确定两操作数之间的关系（如等于、大于、小于等）

- 在分支程序设计中，用来产生条件，其后往往紧跟的是一条条件转移指令。

4.2 8086/8088 指令系统

如：



4.2 8086/8088 指令系统

程序中的表现形式为：

●
●
●

```
    CMP  AL,0
    JZ  L1
```

●
●
●

```
L1:
```

●
●
●
●

4.2 8086/8088 指令系统

在此，结合CMP指令的应用，简要介绍一下8086指令系统中的条件转移指令。（见教材P109-111）

4.2 8086/8088 指令系统

例.有两数α和β，则CPU执行下列指令组

```
MOV   AL，α
CMP   AL，β ；（AL）－β，根据差
            ；设置6个状态标志
```

后，要实现条件转移，根据不同的条件，有不同的条件转移指令。

4.2 8086/8088 指令系统

- 根据单个条件标志实现转移

ZF
- 1, JZ/JE 标号　测试条件：ZF=1
- 0, JNZ/JNE 标号　测试条件：ZF=0

SF
- 1, JS 标号　测试条件：SF=1
- 0, JNS 标号　测试条件：SF=1

4.2 8086/8088 指令系统

OF
- 1, JO 标号 测试条件：OF=1
- 0, JNO 标号 测试条件：OF=1

PF
- 1, JP 标号 测试条件：PF=1
- 0, JNP 标号 测试条件：PF=1

CF
- 1, JC 标号 测试条件：CF=1
- 0, JNC 标号 测试条件：CF=1

4.2 8086/8088 指令系统

- 若α、β为两个无符号数，根据比较结果实现转移

α =β JE/JZ　　标号　　测试条件：　　ZF=1

α低于β（或α不高于或不等于β） JB/JC/JNAE 标号 测试条件：　　CF=1

4.2 8086/8088 指令系统

α 不低于β（或高于或等于）

JNB/JAE/JNC 标号 测试条件：CF=0

α 高于β（或不低于或不等于）

JA/JNBE 标号 测试条件：CF∨ZF=0

α 不高于β（或低于或等于）

JNA/JBE 标号 测试条件：CF∨ZF=1

4.2 8086/8088 指令系统

- 若α、β为两个带符号数，根据比较结果实现转移

α =β

JE/JZ　　标号　　测试条件：　ZF=1

α小于β（或α不大于或不等于β）

JL/JNGE 标号 测试条件：SF ⊻OF=1

4.2 8086/8088 指令系统

α不小于β（大于或等于）

JNL/JGE 标号 测试条件：SF ⊻OF=0

α大于β（不小于或不等于）

JG/JNLE 标号 测试条件：(SF ⊻OF) ∨ ZF=0

α不大于β（小于或等于）

JNG/JLE 标号 测试条件：(SF ⊻OF) ∨ ZF=1

4.2 8086/8088 指令系统

- 测试CX的值为0则转移

JCXZ　标号　测试条件：(CX)=0

下面举例说明比较指令（CMP）和条件转移指令的用法。

4.2 8086/8088 指令系统

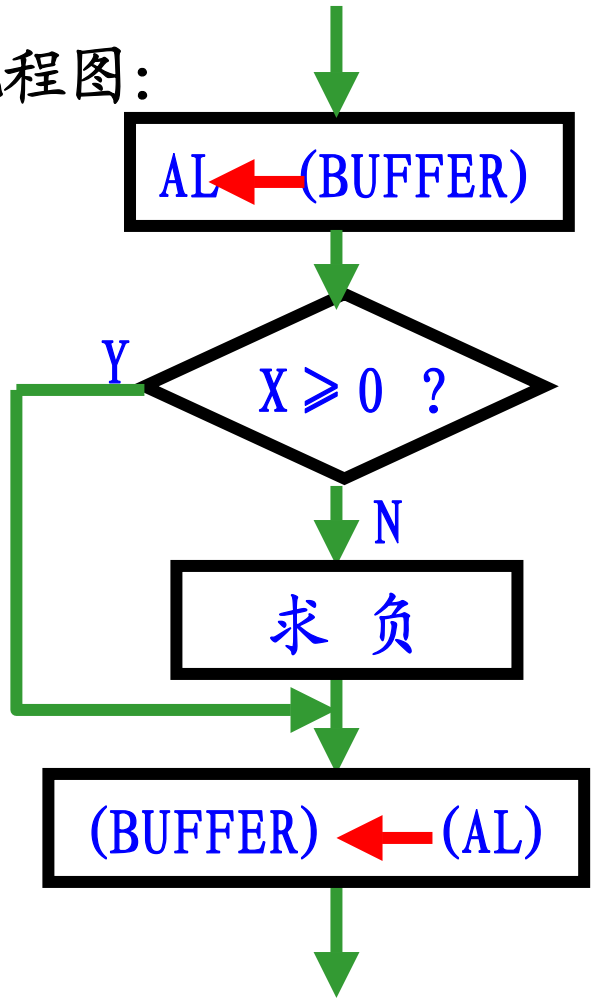
例1. 设在存储器BUFFER单元存放着一个字节的带符号数X，求X的绝对值，并将结果放回原处。（P111例4.3.3）

算法：

$$|X| = \begin{cases} X, ; & \text{当 } X \geqslant 0 \text{时} \\ -X, ; & \text{当 } X < 0 \text{时} \end{cases}$$

4.2 8086/8088 指令系统

流程图：



程序：

```
        MOV  AL, BUFFER
        CMP  AL, 0
        JNS  NONEG
        NEG  AL
NONEG:  MOV  BUFFER, AL
```

4.2 8086/8088 指令系统

例2：符号函数的处理

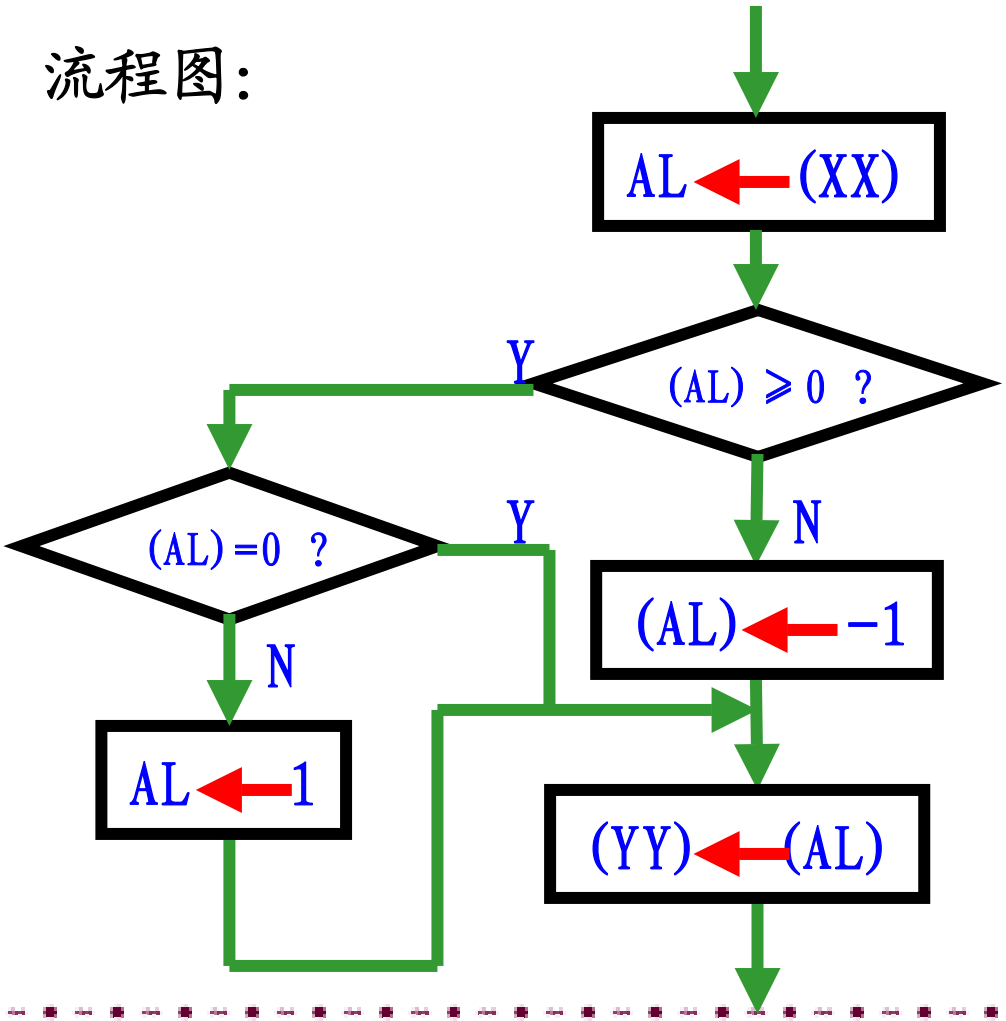
有一符号函数：

$$Y=\begin{cases}1, & \text{当 } X>0 \text{时} \\ 0, & \text{当 } X=0 \text{时} \\ -1, & \text{当 } X<0 \text{时}\end{cases} \quad (-128 \leqslant X \leqslant +127)$$

设给定值X存放在XX单元，函数Y值存放到YY单元。

4.2 8086/8088 指令系统

流程图：



程序：

```
        MOV  AL, XX
        CMP  AL, 0
        JGE  BIGR
        MOV  AL, -1
        JMP  EQUL
BIGR:   JE   EQUL
        MOV  AL, 1
EQUL:   MOV  YY, AL
```

4.2 8086/8088 指令系统

3. 乘法指令（MUL/IMUL）

乘法指令分带符号运算和不带符号运算两种。带符号运算时，操作数和结果均以补码表示，结果的符号按一般的运算规则确定。

无符号数乘法指令（MUL）

格式：MUL　SRC

不能为立即数

4.2 8086/8088 指令系统

功能：

DST(被乘数)　　SRC(乘数)

字节乘：AX ← 积 (AL) * (SRC)

字乘：DX:AX ← 积 (AX) * (SRC)

操作类型：用SRC的类型确定。

乘法指令只影响CF和OF标志，其余标志均无意义。

其中，

CF=0
OF=0
} 表明结果的高位字节（或字）为全0。

CF=1
OF=1
} 表明结果的高位字节（或字）有有效积。

4.2 8086/8088 指令系统

带符号数乘法指令（IMUL）

格式：IMUL　SRC

不能为立即数

4.2 8086/8088 指令系统

功能：

DST(被乘数)　　SRC(乘数)

字节乘：AX ← 积 (AL) * (SRC)

字乘：DX:AX ← 积 (AX) * (SRC)

操作类型：用SRC的类型确定。

乘法指令只影响CF和OF标志，其余标志均无意义。

4.2 8086/8088 指令系统

其中，

CF=0
OF=0
} 表明结果的高位字节（或字）仅仅是低位字节（或字）的符号扩展。

CF=1
OF=1
} 表明结果的高位字节（或字）有效积。

4.2 8086/8088 指令系统

总之，乘法指令中的目的操作数一定为AL（8位数相乘）或AX（16位数相乘），源操作数不能为立即数，但可采用其它寻址方式，指令对字节运算还是字运算由源操作数的类型确定。

4.2 8086/8088 指令系统

例1. 下列指令是合法的。

MUL CX ; (DX: AX) ← (AX) * (CX)，为字操作

IMUL CL ; (AX) ← (AL) * (CL)，为字节操作

下列指令是非法的。

MUL AL, BL ✗ DST(被乘数)应为隐含寻址

IMUL 05H ✗ SRC(乘数)不能为立即数寻址

4.2 8086/8088 指令系统

例2.若（AL）=05H,(BL)=0FDH

则CPU执行:

```
MUL   BL   ; 无符号数乘法
           ; （AL）*（BL）积→AX
           ; ∴（AX）=04F1H
```

4.2 8086/8088 指令系统

则CPU执行:

```
IMUL  BL    ;带符号数乘法
            ;   (AL) * (BL) 积→ AX
            ;   05H*0FDH=0FFF1H
            ;   ∴  (AX) =0FFF1H
```

[-3]补

(即[-15]补)

4.2 8086/8088 指令系统

——除法指令

4. 除法指令（DIV/IDIV）

除法指令和乘法指令一样，分无符号除法和带符号除法。操作类型有字节除法和字除法，其操作类型取决于SRC(除数)的类型。

无符号数除法指令（DIV）

格式：DIV　SRC

4.2 8086/8088 指令系统

——除法指令

带符号数除法指令（IDIV）

格式：IDIV　SRC

其中，DST为隐含寻址（作被除数）

字节除时，被除数一定在AX中

字除时，被除数一定在DX:AX中

SRC（除数）不能为立即数

4.2 8086/8088 指令系统

——除法指令

功能:

字节除法: (AX) / (SRC) → 商 → AL；余数 → AH

字除法: (DX: AX) / (SRC) → 商 → AX；余数 → DX

4.2 8086/8088 指令系统

——除法指令

除法运算后，标志位没有意义。除法不允许出现除数为0或商溢出，若发生除数为0或商溢出则其结果没有意义，并引起中断(关于中断的概念以后再作介绍）。

4.2 8086/8088 指令系统

——除法指令

例1. 下列指令是合法的。

```
DIV    BL
IDIV   CX
DIV    WORD PTR [BX][SI]
```

下列指令是非法的。

DIV 12 ✗ SRC(除数)不能为立即数寻址

DIV [SI]+02H ✗ SRC类型不明确

IDIV AX,BL ✗ DST(被除数)应为隐含寻址

4.2 8086/8088指令系统

——除法指令

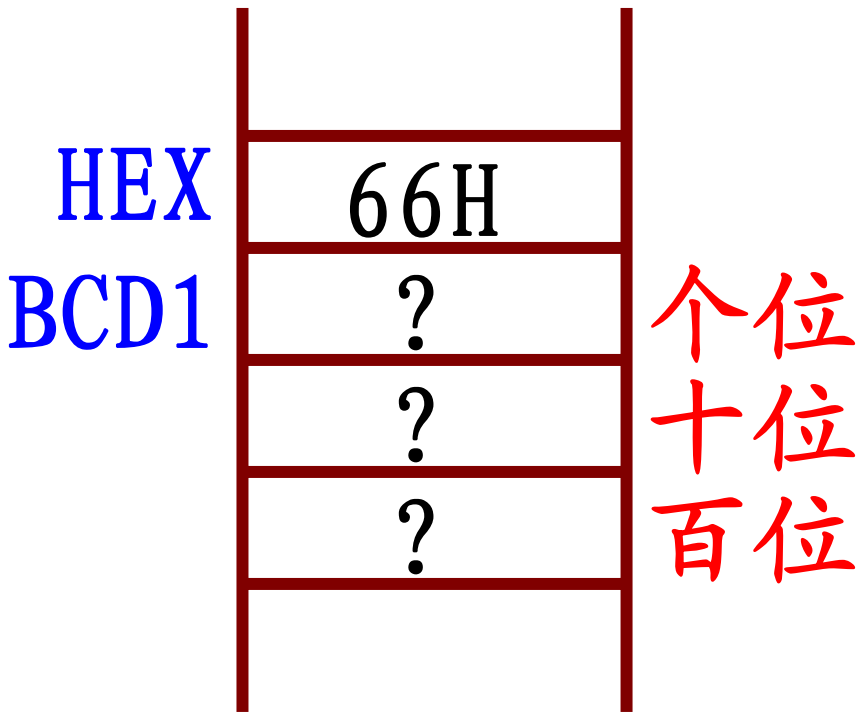
例2.

在存储器HEX单元存放着一个字节的无符号二进制数，试将其转换成十进制数以分离BCD数形式存到BCD1以下存储单元。（设个位存在低地址单元）

4.2 8086/8088 指令系统

——除法指令

分析题目



4.2 8086/8088 指令系统

——除法指令

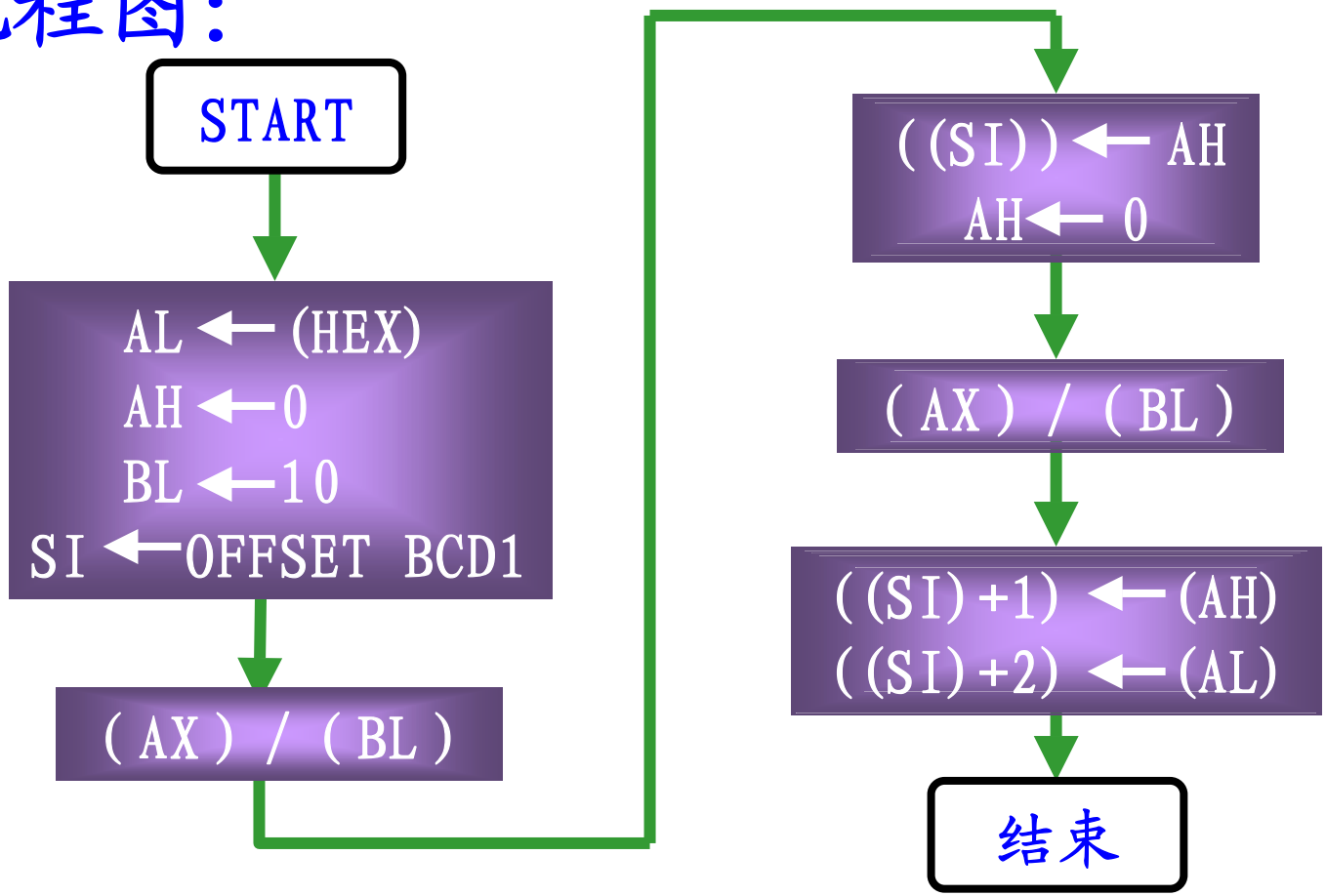
确定算法：

用除10取余法。即用被转换的数除以10，第一次得到的余数为转换后的十进制数的个位数，用中间商再除以10，得到的余数为转换后十进制数的十位数，最后的商为十进制数的百位数。

4.2 8086/8088 指令系统

——除法指令

流程图：



4.2 8086/8088 指令系统

——除法指令

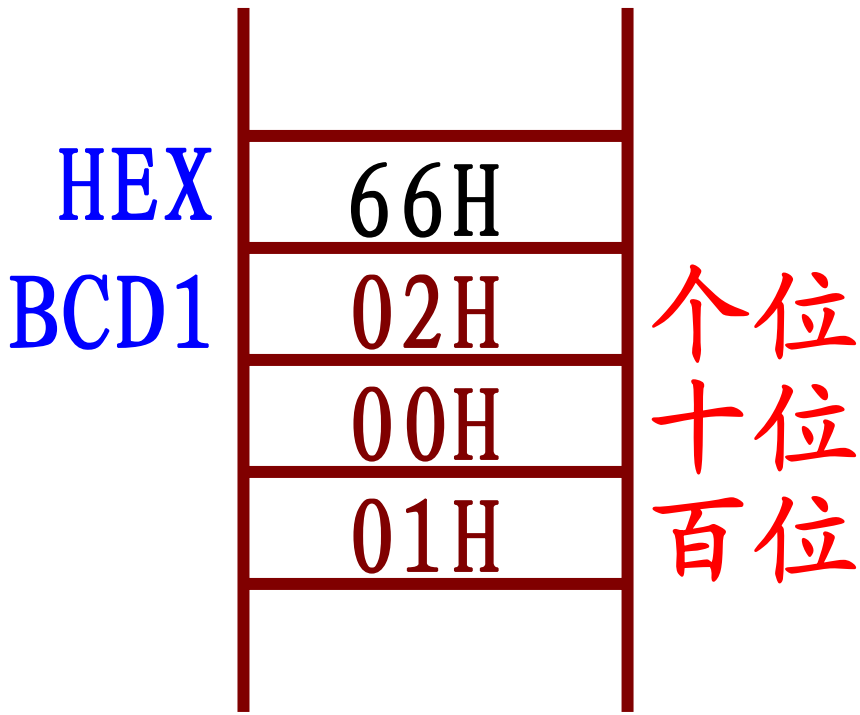
程序：

```
; 在数据段定义变量如下:
  HEX DB 66H
  BCD1 DB 3 DUP (?)
; 在代码段编写程序
  MOV AH,0
  MOV AL,HEX
  MOV BL,10
  MOV SI,OFFSET BCD1
  DIV BL
  MOV [SI],AH
  MOV AH,0
  DIV BL
  MOV [SI]+1,AH
  MOV [SI]+2,AL
        .
        .
        .
```

4.2 8086/8088 指令系统

——除法指令

程序执行后存储器示意图



4.2 8086/8088指令系统
——符号扩展指令

5.符号扩展指令（CBW/CWD）

- 格式：　CBW

- 功能：将AL中的符号扩展到AH中，即将一个字节的带符号数扩展成一个字。

4.2 8086/8088 指令系统

——符号扩展指令

- 格式：　CWD

- 功能：将AX中的符号扩展到DX中，将一个字的带符号数扩展成双字。

4.2 8086/8088 指令系统

——符号扩展指令

应用场合:

在算术运算中，有时会遇到两个长度不等的数进行加、减运算，此时，应将长度短的数的位数扩展，以使两数的长度一致，只有这样，才能保证参加运算的两个操作数的类型是一致的。对于一个无符号数来说，这种扩展是简单的，只要将其高位补“0”就可以；但对一个带符号数来说就不一样了，高位扩展时，补“0”还是补“1”就取决于该数的符号位。

4.2 8086/8088 指令系统

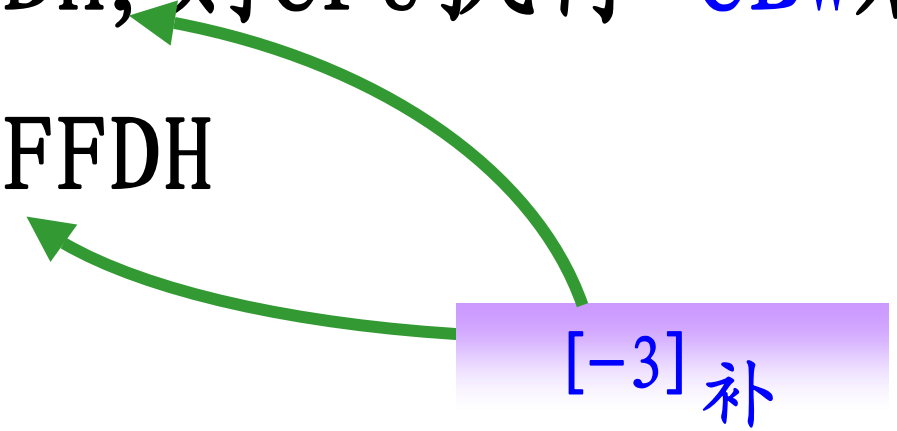
——符号扩展指令

例1. 若(AL)=05H, 则CPU执行 CBW后，

(AL)=0005H。

若(AL)=0FDH, 则CPU执行 CBW后，

(AX)=0FFFDH



4.2 8086/8088指令系统

——符号扩展指令

例2.写出计算Y=a*b+c-18的程序

（P103 例4.3.1）

```
;  在数据段定义变量如下:
   DAT1  DB   34H   ;34H为变量a的一个设定值
   DAT2  DB   56H   ;56H为变量b的一个设定值
   DAT3  DB   0E7H  ;0E7H为变量c的一个设定值
   DATY  DW  ?      ;DATY单元存放结果
```

4.2 8086/8088 指令系统

——符号扩展指令

```
;  在代码段编写程序
   MOV   AL, DAT1                ;取a
   MOV   BL, DAT2                ;取b
   IMUL  BL                      ; (AL)*(BL)  →(AX)
   MOV   BX, AX                  ; (AX) → (BX)
   MOV   AL, DAT3                ;取c
   CBW                           ;扩展AL →AX
   ADD   AX, BX                  ; (AX)+(BX) → (AX)
   SUB   AX, 18
   MOV   DATY, AX
           ⋮
```

Y=a*b+c-18

4.2 8086/8088 指令系统

——BCD数调整指令

6. BCD数调整指令

● 我们知道，BCD数是用4位二进制码表示的一位十进制数。在计算机中，所有运算均以二进制运算为基础，要实现对BCD数进行十进制运算，通常分两步进行，首先对BCD数按二进制进行运算，然后对运算结果进行相应的调整，才能得到正确的BCD数结果。

4.2 8086/8088指令系统

——BCD数调整指令

- 为什么必须调整呢？这是因为4位二进制码有16种不同的状态，而一位十进制数只有10种不同的数字，当BCD数按二进制运算时，不可避免地会出现6种不用的状态，加减运算的十进制调整指令的基本思想就是在必要的时候让运算的结果跳过6种不用的状态。

4.2 8086/8088指令系统

——BCD数调整指令

- BCD数由4位二进制码组成，因此一个字节中可以存放一位BCD数，也可以存放两位BCD数，通常将前者称为分离BCD数，后者称为组合BCD数。

4.2 8086/8088 指令系统

——BCD数调整指令

BCD数调整指令共有6条，如下所示：

助记符格式	功　能　说　明
AAA	加法分离BCD调整
AAS	减法分离BCD调整
DAA	加法组合BCD调整
DAS	减法组合BCD调整
AAM	乘法分离BCD调整
AAD	除法分离BCD调整

4.2 8086/8088指令系统

——BCD数调整指令

在BCD数运算时，应注意以下几个问题：

①BCD数加、减法运算，可以按分离BCD数运算，也可以按组合BCD数运算。而BCD数乘、除法运算，只能按分离BCD数运算。

4.2 8086/8088 指令系统

——BCD数调整指令

②被调整的数隐含的存放在AL中（乘法时在AX中）。

③加法、减法、乘法都是先运算后调整，而除法是先调整后运算。

4.2 8086/8088 指令系统

——BCD数调整指令

下面通过例子说明BCD数的调整规则

例1. 用组合BCD数求1945+1271的运算。

设加数及和的低字节放在低地址单元中。加数及和的高字节放在高地址单元中。

4.2 8086/8088 指令系统

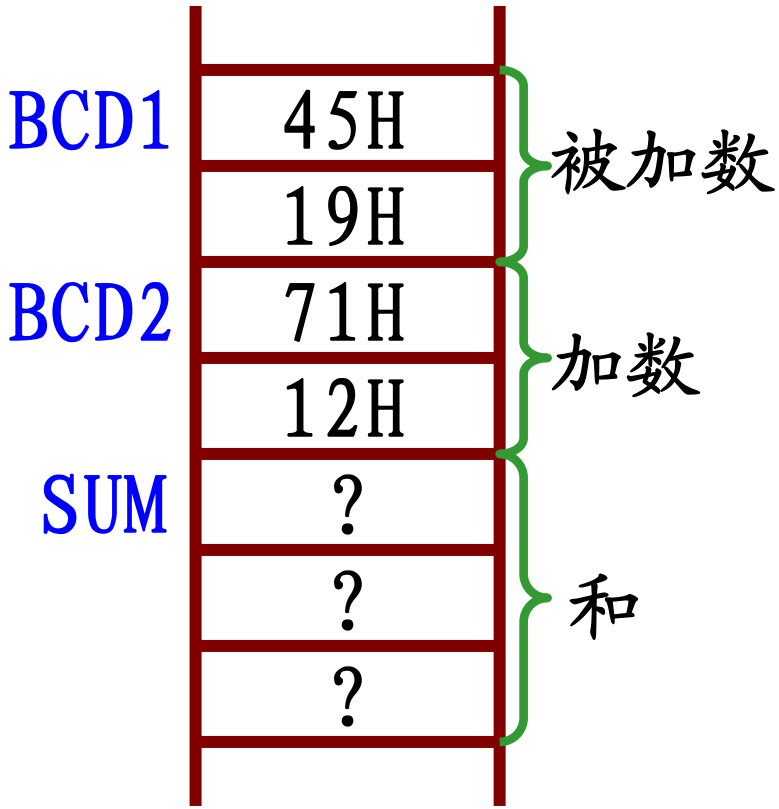
——BCD数调整指令

- 在数据段定义变量如下：

```
BCD1  DB  45H, 19H

BCD2  DB  71H, 12H

SUM   DB  3   DUP(?)
```



4.2 8086/8088 指令系统

——BCD数调整指令

- 人要做的运算为

```
  1 9 4 5
+ 1 2 7 1
---------
  3 2 1 6
```

CPU要做的运算为

```
  1 9 4 5 H
+ 1 2 7 1 H
-----------
  2 B B 6 H
```

ADC加法 ADD加法

结果不是BCD数结果，需用DAA调整。

4.2 8086/8088 指令系统

——BCD数调整指令

- 在程序段编写程序如下：

变量定义

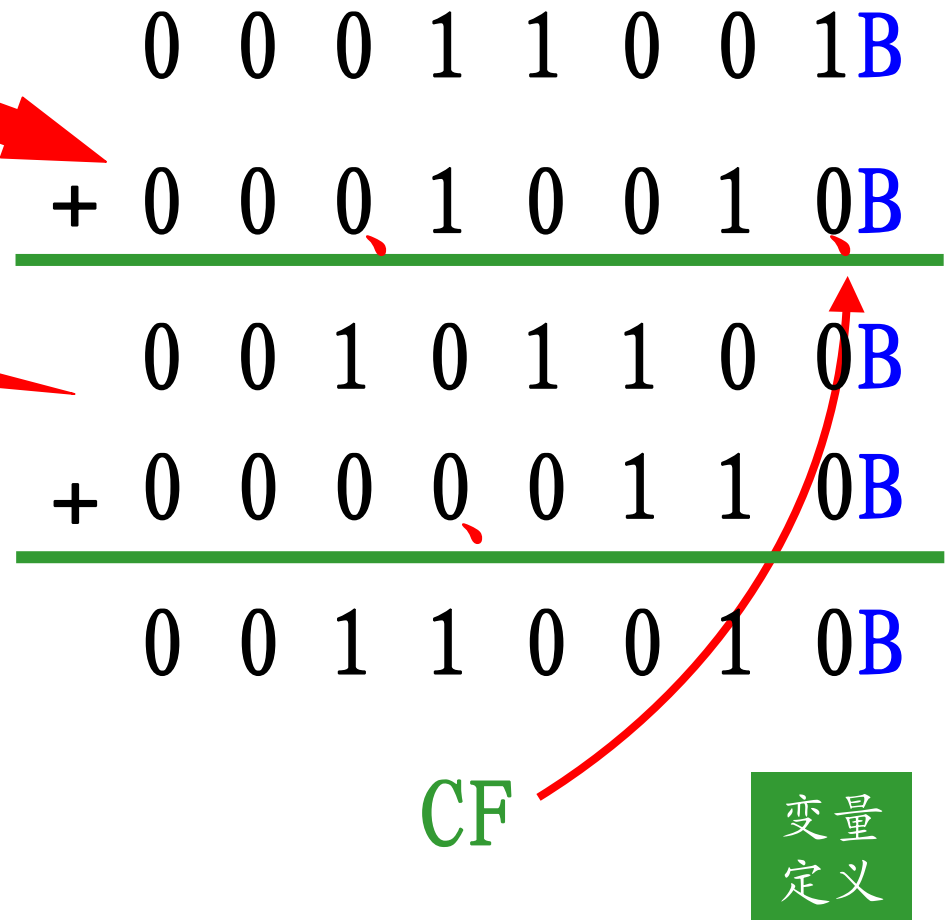
```
MOV   AL, BCD1
ADD   AL, BCD2
DAA
MOV    SUM, AL
MOV   AL, BCD1+1
```

```
  0 1 0 0 0 1 0 1B
+ 0 1 1 1 0 0 0 1B
-----------------
  1 0 1 1 0 1 1 0B
+ 0 1 1 0 0 0 0 0B
-----------------
[1]0 0 0 1 0 1 1 0B
```

4.2 8086/8088 指令系统

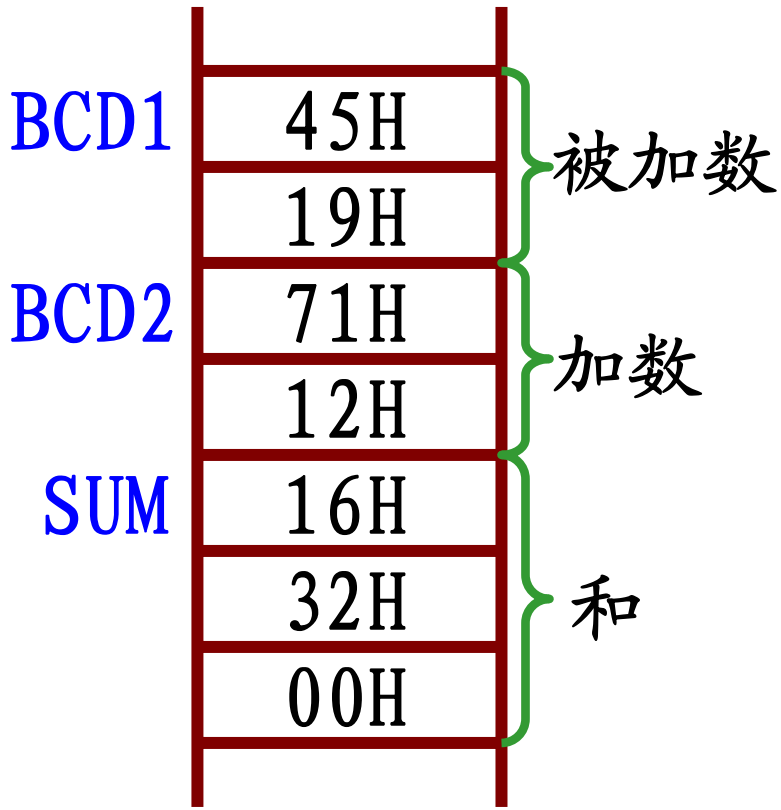
——BCD数调整指令

```
ADC   AL, BCD2+1
DAA
MOV   SUM+1, AL
MOV   AL, 0
ADC   AL , 0
MOV   SUM+2, AL
```



4.2 8086/8088 指令系统

——BCD数调整指令



$$
\begin{array}{r}
1\ 9\ 4\ 5 \\
+\ 1\ 2\ 7\ 1 \\
\hline
3\ 2\ 1\ 6
\end{array}
$$

4.2 8086/8088 指令系统

——BCD数调整指令

例2. 十进制（BCD）数乘法 5*9运算

```
MOV   AL, 05H
MOV   BL, 09H
MUL   BL      ;
AAM           ;
```

(AL) * (BL) → AX

即45

∴ (AX) =002DH

(AX) /10 → 商 → AH (十位)

余数 → AL (个位)

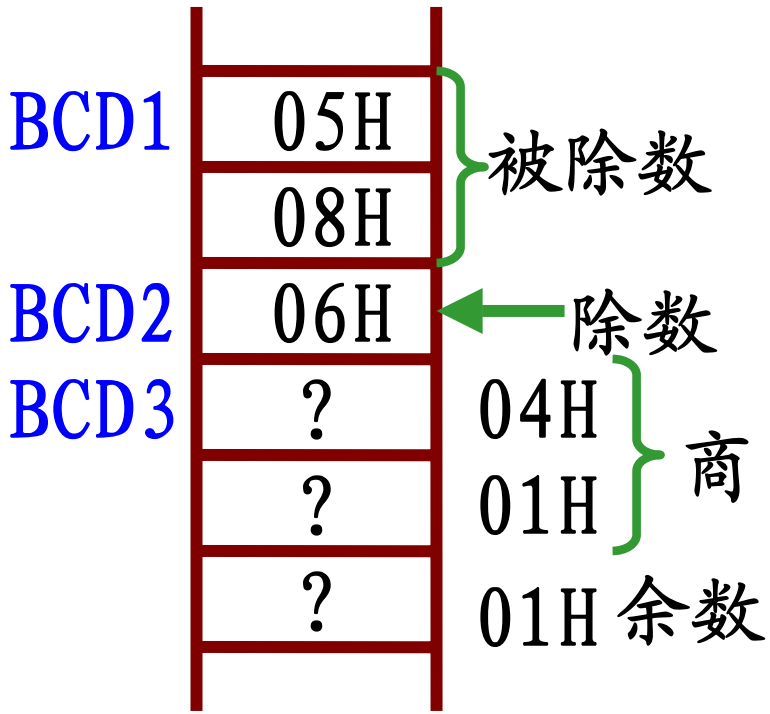
∴ (AX) =0405H

4.2 8086/8088 指令系统

——BCD数调整指令

例3. BCD数除法运算

设有两数X, Y, 且X=85, Y=6，用分离BCD数完成X/Y。




```
    1 4
  ------
6 | 8 5
    6
  ----
    2 5
    2 4
  ----
      1
```

程序上　程序下

4.2 8086/8088 指令系统

——BCD数调整指令

我们有如下程序：

```
BCD1   DB  05H,  08H ;  高位在高字节
BCD2   DB  06H
BCD3   DB  3 DUP(?)
   :
MOV    AH,  0
MOV    AL,  BCD1+1   ;  (AL)=08
MOV    BL,  BCD2     ;  (BL)=06
```

被除数在AX中

除数放在BL中

变量定义

4.2 8086/8088 指令系统

——BCD数调整指令

```
DIV    BL          ; (AX) / (BL) → (AL)=01, (AH)=02
MOV    BCD3+1, AL  ; 商的高位送BCD3+1
MOV    AL, BCD1    ; (AL)=05
AAD                ; (AL)=(AH)*10+(AL)=25, 0→(AH)
DIV    BL          ; (AX) / (BL) → (AL)=04, (AH)=01
MOV    BCD3, AL    ; 商的低位送BCD3
MOV    BCD3+2, AH  ; 余数送BCD3+2
```

变量定义

4.2 8086/8088 指令系统

——逻辑运算类指令

三. 逻辑运算类指令（5条）

5条逻辑运算类指令书写格式与完成的功能如下表所示。

助记符格式			功 能 说 明
与	AND	DST，SRC	(DST) ← (DST) ∧ (SRC)
或	OR	DST，SRC	(DST) ← (DST) ∨ (SRC)
异或	XOR	DST，SRC	(DST) ← (DST) ∀ (SRC)
测试	TEST	DST，SRC	(DST) ∧ (SRC)置各标志位
非	NOT	DST	(DST)中各位取反

4.2 8086/8088 指令系统

——逻辑运算类指令

指令的运算都是按位进行的。

NOT指令是将操作数的各位取反，其它指令是两个操作数的对应位实行相应的逻辑运算。指令可以有字节操作，也可以有字操作。

4.2 8086/8088 指令系统

——逻辑运算类指令

➡ 指令执行后对标志位的影响情况:

除NOT指令对标志位不产生影响外，其余指令将使CF、OF置0，并以正常规则设置SF、ZF和PF的状态。

4.2 8086/8088 指令系统
——逻辑运算类指令

应用场合

- AND指令经常用在使DST的某些位清零，其余位不变的场合。

例如：使AL寄存器的高4位清零，低4位不变，方法为：

AND AL, 0FH

4.2 8086/8088 指令系统
——逻辑运算类指令

- OR指令经常用在使DST的某些位置1，其余位不变的场合。

例如：使BX寄存器的低4位置1，其余位不变，方法为：

```
OR BX, 000FH
```

4.2 8086/8088 指令系统
——逻辑运算类指令

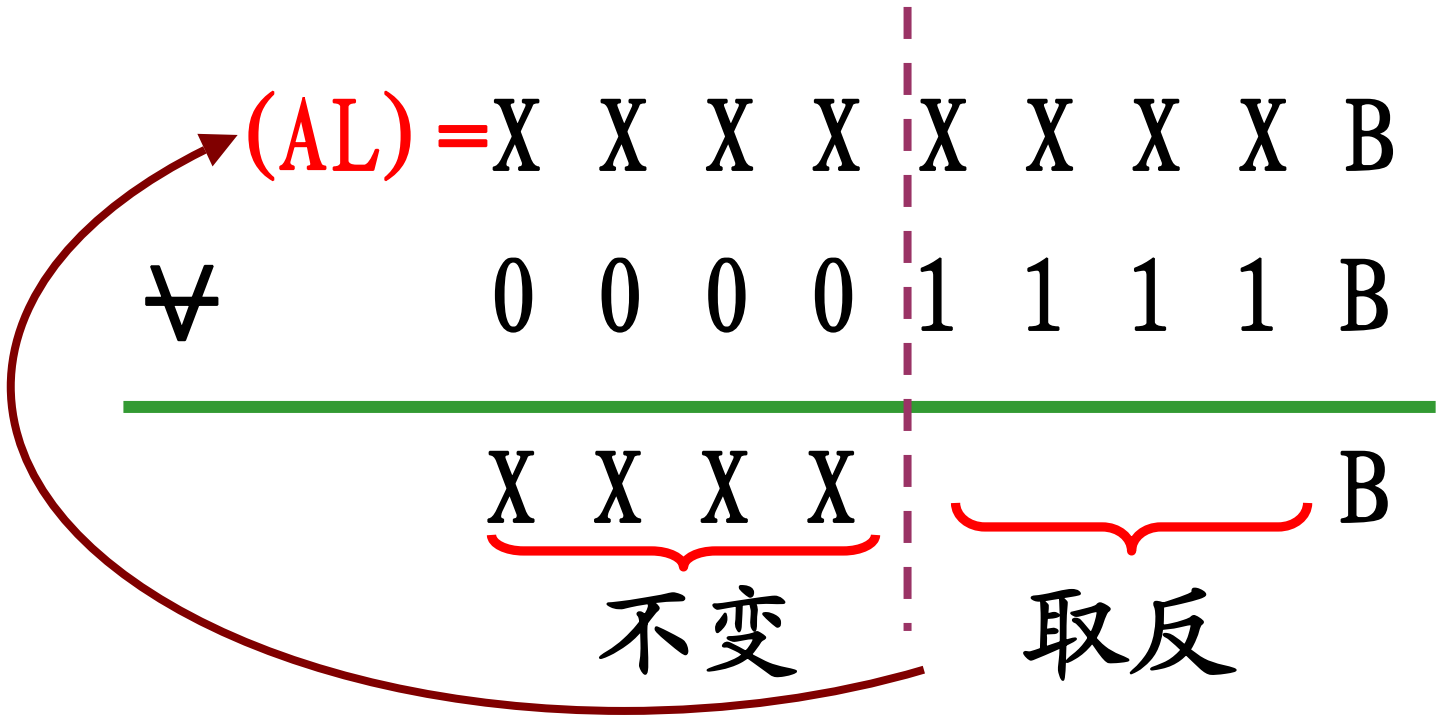
- XOR指令经常用在使DST的某些位取反，其余位不变的场合。

例如：使AL寄存器的低4位取反，其余位不变，方法为：

```
XOR AL, 0FH
```

4.2 8086/8088 指令系统

——逻辑运算类指令



4.2 8086/8088 指令系统

——逻辑运算类指令

- NOT指令是将DST的各些位取反。

例如：若（AL)=00H,则CPU执行

NOT　AL　指令后，

（AL）=0FFH

4.2 8086/8088 指令系统
——逻辑运算类指令

- TEST指令主要应用在条件分支程序设计中。

例如：测试BX的D_3、D_2位，若同时为0，则AL置1，否则AL清零。

4.2 8086/8088 指令系统

——逻辑运算类指令

```
        TEST   BX,  000CH
        JZ     SET1
        MOV    AL,  0
        JMP    EXIT
SET1:   MOV    AL,  0FFH
EXIT:   ...
```

4.2 8086/8088 指令系统
——逻辑运算类指令

★例：使AX清0可用以下方法

- MOV AX, 0000H
- XOR AX, AX
- AND AX, 0000H
- SUB AX, AX
- MOV CL, 16
  SHL AX, CL

4.2 8086/8088 指令系统

——移位和循环移位指令

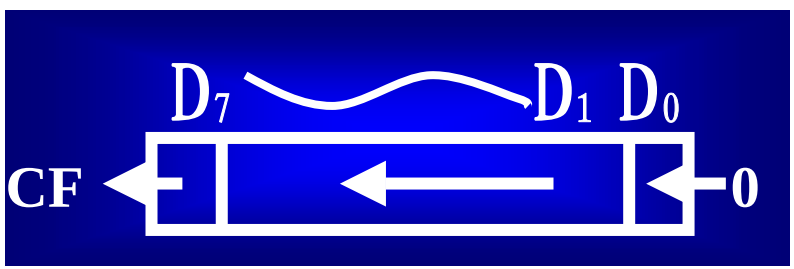
四. 移位和循环移位指令

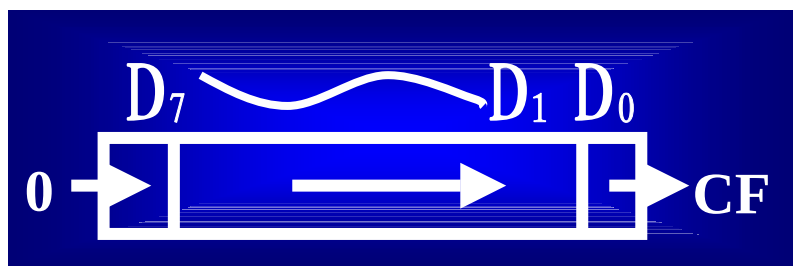
1. 移位指令

移位次数

①逻辑移位

左移　SHL DST, CNT　　右移　SHR DST, CNT





4.2 8086/8088指令系统

——移位和循环移位指令

指令中的目的操作数DST可以为除立即数和CS以外的其它各种寻址方式，CNT则可以为立即数和寄存器CL。当CNT为寄存器CL时，移位次数存放在CL中。

注意：当CNT为立即数时，其值只能为1。

4.2 8086/8088 指令系统

——移位和循环移位指令

例.

```
SHR  AX,  1  ;将AX的内容右移1位，最
             ;高位补0
SHL  AL,  CL ;将AL的内容左移CL中指
             ;定的次数，且每次移位
             ;最低位补0
```

4.2 8086/8088 指令系统

——移位和循环移位指令

SHL DAT1[SI],CL;将内存某单元的

;内容左移CL中所

;中指定的次数

SHR BL, 2 ✗ 当移位次数>1时，必

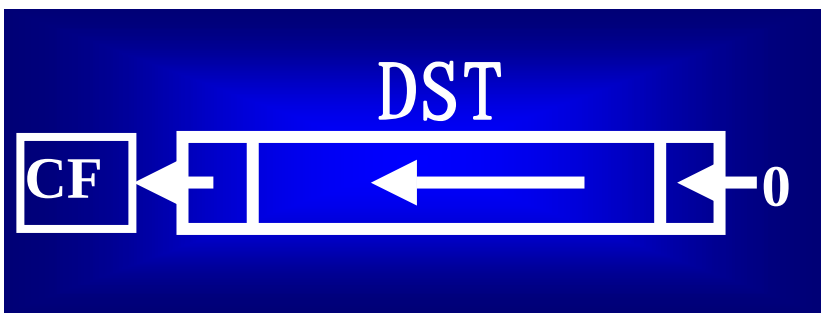
须用CL提供移位次数

4.2 8086/8088 指令系统

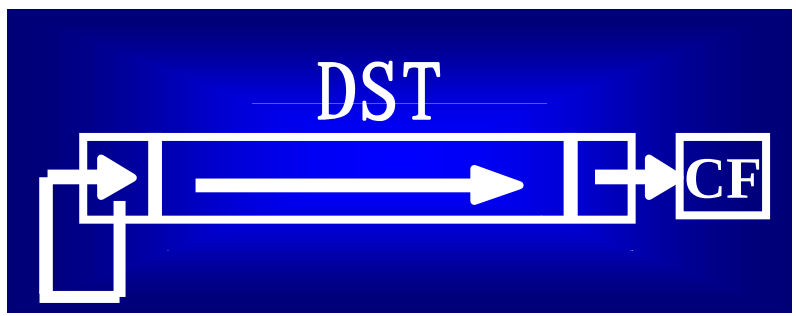
——移位和循环移位指令

②算术移位

左移　SAL DST, CNT



右移　SAR DST, CNT



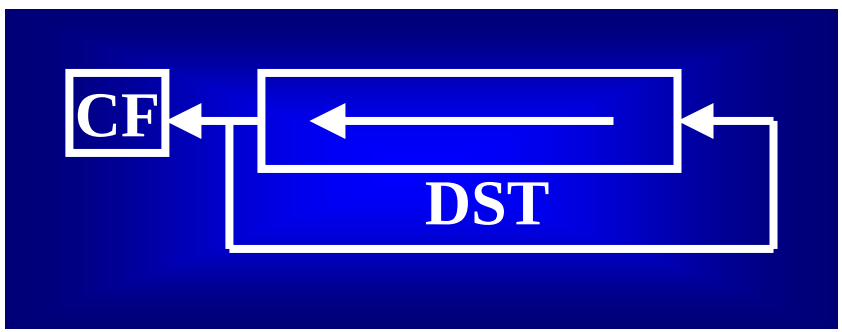
4.2 8086/8088 指令系统

——移位和循环移位指令

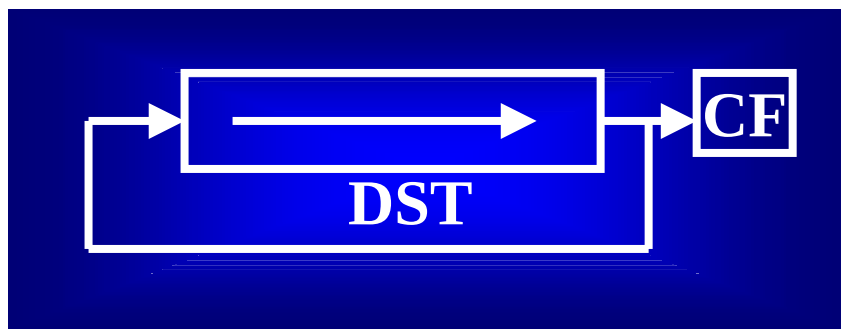
2. 循环移位指令

①不带CF的循环移位

左移　ROL DST, CNT

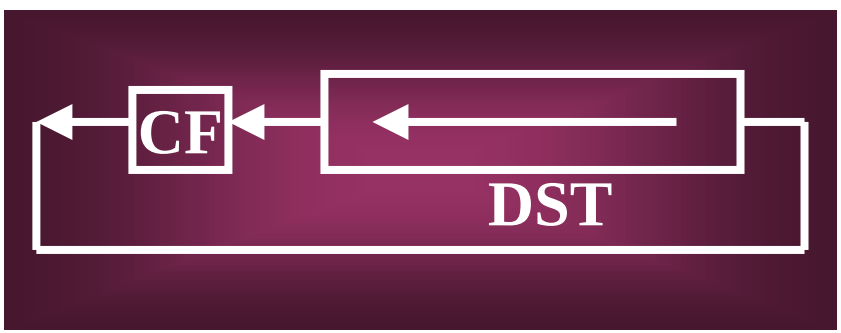


右移　ROR DST, CNT

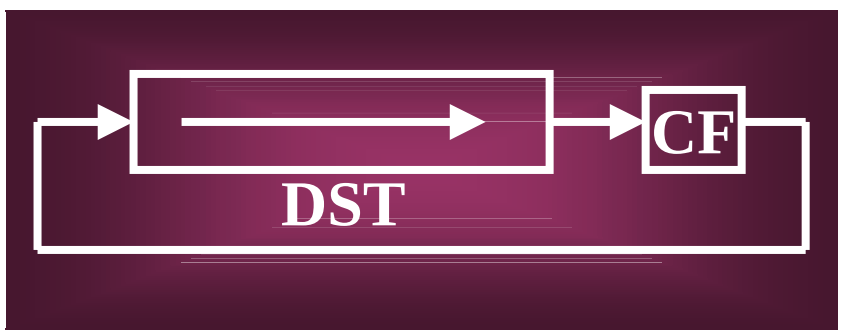


②带CF的循环移位

左移　RCL DST, CNT



右移　RCR DST, CNT



4.2 8086/8088 指令系统

——移位和循环移位指令

以上所有移位指令和循环移位指令对DST和CNT的要求同逻辑移位指令规则，即：

➡ 指令中的目的操作数DST可以为除立即数和CS以外的其它各种寻址方式，CNT则可以为立即数和寄存器CL。当CNT为寄存器CL时，移位次数存放在CL中。

注意：当CNT为立即数时，其值只能为1。

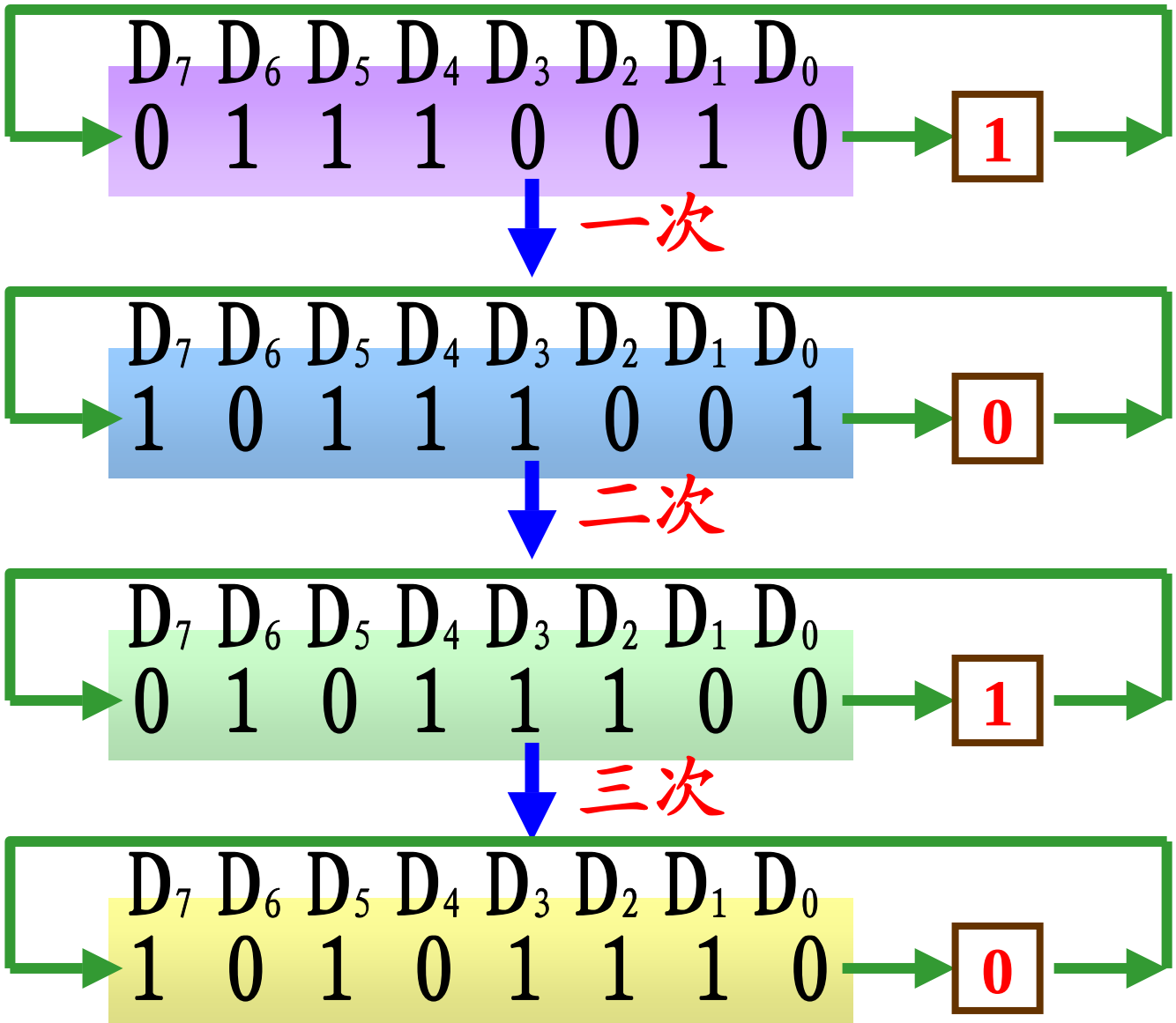
4.2 8086/8088 指令系统

——移位和循环移位指令

例1. 若(BL)=01110010B, (CL)=3, CF=1, 则CPU执行:

RCR BL, CL 指令后,

(BL)=10101110B=0AEH


D7 D6 D5 D4 D3 D2 D1 D0
0 1 1 1 0 0 1 0
1
一次
D7 D6 D5 D4 D3 D2 D1 D0
1 0 1 1 1 0 0 1
0
二次
D7 D6 D5 D4 D3 D2 D1 D0
0 1 0 1 1 1 0 0
1
三次
D7 D6 D5 D4 D3 D2 D1 D0
1 0 1 0 1 1 1 0
0

4.2 8086/8088 指令系统

——移位和循环移位指令

★例2. 编程序实现5*10运算。
（不能用MUL指令）

算法：

5*10

=5*（2+8）

=5*2+5*8

4.2 8086/8088 指令系统

——移位和循环移位指令

```
MOV   AL,  05H
SHL   AL,  1   ; (AL)=5*2
MOV   BL,  AL  ; (BL)=5*2
MOV   CL,  2
SHL   AL,  CL  ; (AL)=5*8
ADD   AL,  BL  ; (AL)=5*10
```

4.2 8086/8088 指令系统

——处理器控制指令与标志处理指令

五. 处理器控制指令与标志处理指令

助记符格式	功能说明	助记符格式	功能说明
CLC	0 → CF	CLI	0 → IF
STC	1 → CF	STI	1 → IF
CMC	CF → CF	NOP	空操作
CLD	0 → DF	HLT	暂停
STD	1 → DF		

4.2 8086/8088指令系统

——移位和循环移位指令

- NOP指令

CPU执行该指令时不完成任何具体功能也不影响标志位，只占用机器的3个时钟周期。所以，也称为空操作指令。

4.2 8086/8088 指令系统

——移位和循环移位指令

- HLT指令

该指令是使CPU进入暂停状态。只有当下面三种情况之一发生时，CPU才退出暂停状态。这三种情况是： CPU的复位输入端RESET线上有复位信号；非屏蔽中断请求输入端NMI线上出现请求信号；可屏蔽中断输入端INTR线上出现请求信号且标志寄存器的中断标志IF=1。