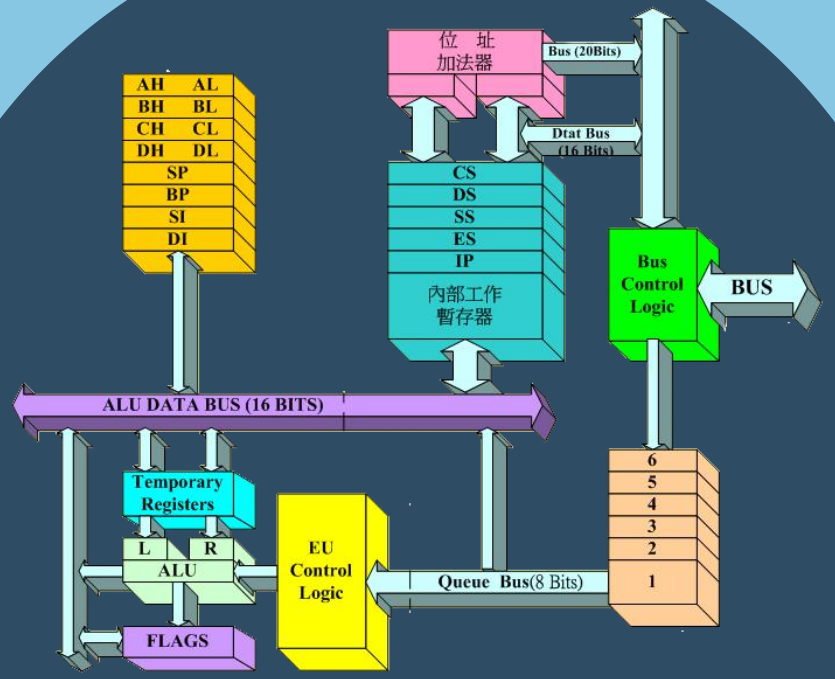
标志寄存器

贺利坚　主讲



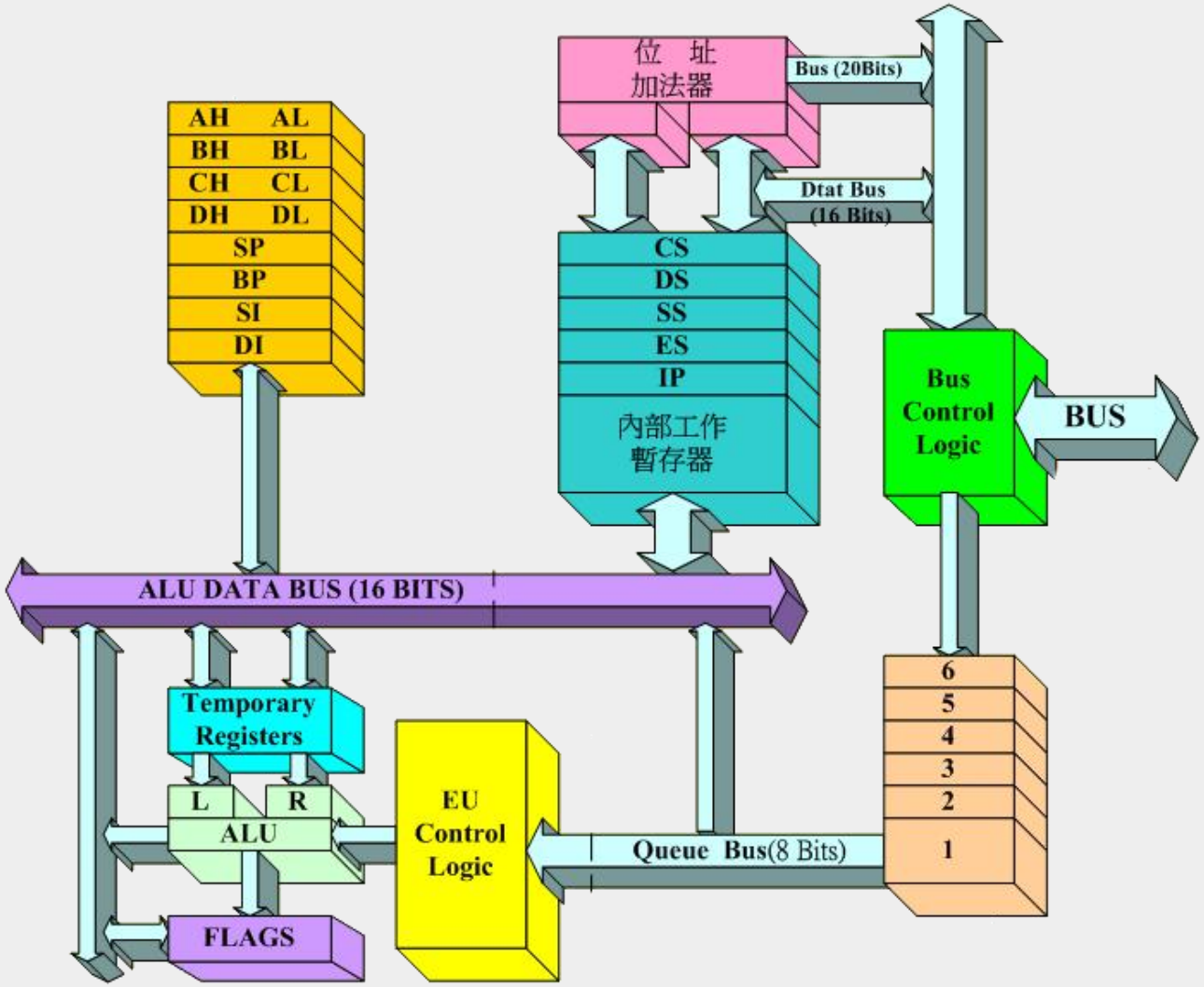
汇编语言程序设计

Assembly Language

标志寄存器

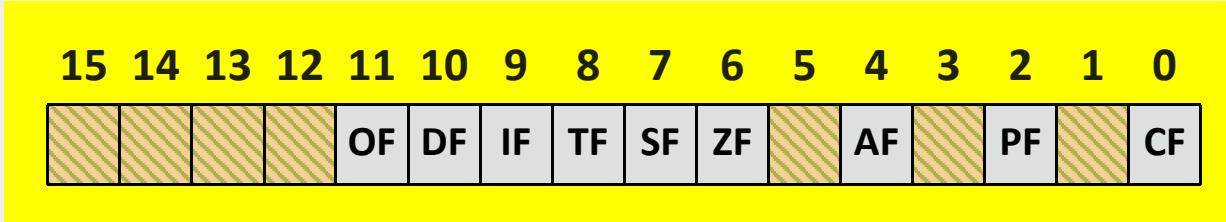
- 8086CPU有14个寄存器：
  - 通用寄存器：AX、 BX、 CX、 DX
  - 变址寄存器：SI、 DI
  - 指针寄存器：SP、 BP
  - 指令指针寄存器： IP
  - 段寄存器：CS、 SS、 DS、 ES
  - 标志(flag)寄存器：PSW/FLAGS

别称：程序状态字



认识标志寄存器的特殊之处

- 标志寄存器的结构
  - flag寄存器是按位起作用的，也就是说，它的每一位都有专门的含义，记录特定的信息。
  - 8086CPU中没有使用flag的1、3、5、12、13、14、15位，这些位不具有任何含义。
- 标志寄存器的作用
  - 用来存储相关指令的某些执行结果
  - 用来为CPU执行相关指令提供行为依据
  - 用来控制CPU的相关工作方式



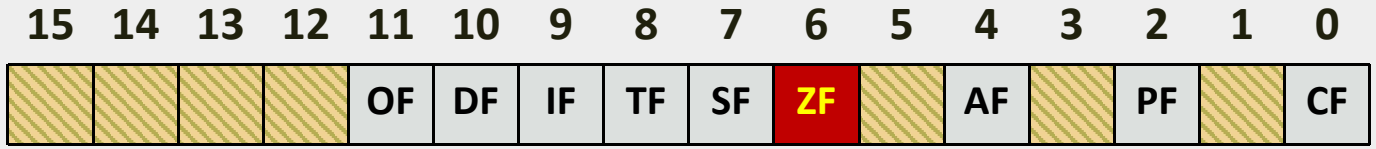
- 观察寄存器的值

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=**** ES=**** SS=**** CS=**** IP=0100  NV UP EI PL NZ NA PO NC
                                        ↑  ↑     ↑  ↑     ↑  ↑
                                        OF DF    SF ZF    PF CF
```

- 直接访问标志寄存器的方法
  - pushf ：将标志寄存器的值压栈；
  - popf ：从栈中弹出数据，送入标志寄存器中。

	标志	值为1	值为0	意义	
Overflow	OF	OV	NV	溢出	
Direction	DF	DN	UP	方向	
Sign	SF	NG	PL	符号	Positive /negative
Zero	ZF	ZR	NZ	零值	
Parity	PF	PE	PO	奇偶	odd/even
Carry	CF	CY	NC	进位	

ZF-零标志(Zero Flag)

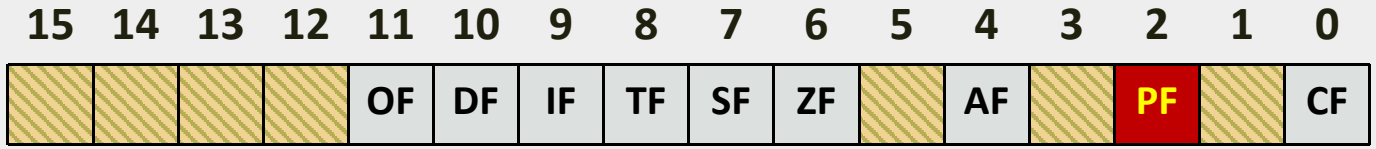


- ZF标记相关指令的计算结果是否为0
  - ZF=1，表示“结果是0”，1表示“逻辑真”
  - ZF=0，表示“结果不是0”，0表示“逻辑假”
- 示例

指令	执行结果
mov ax,1 and ax,0	ZF=1， 表示“结果是0”
mov ax,1 or ax,0	ZF=0， 表示“结果非0”

- 在8086CPU的指令集中，有的指令的执行是影响标志寄存器的，比如：add、sub、mul、div、inc、or、and等，它们大都是运算指令，进行逻辑或算术运算；
- 有的指令的执行对标志寄存器没有影响，比如：mov、push、pop等，它们大都是传送指令。
- 使用一条指令的时候，要注意这条指令的全部功能，其中包括执行结果对标记寄存器的哪些标志位造成影响。

PF-奇偶标志(Parity Flag)



- PF记录指令执行后，结果的所有二进制位中1的个数：
  - 1的个数为偶数，PF = 1；
  - 1的个数为奇数，PF = 0。
- 示例

指令	执行结果
mov al,1 add al,10	结果为0000 1011B = 0000 0001B + 0000 1010B 其中有3（奇数）个1，则PF=0；
mov al,1 or al,2	结果为00000011B = 0000 0001B or 0000 0010B 其中有2（偶数）个1，则PF=1；

SF-符号标志(Sign Flag)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

- SF记录指令执行后，将结果视为有符号数
  - 结果为负，SF = 1；
  - 结果为非负，SF = 0。
- 示例

指令	执行结果
mov al,10000001B add al,1	结果al 为10000010B，为负数，则SF=1；
sub ax, ax	结果ax为0，为非负数，故SF=0；

1000 0010B作为有符号数对应-111 1110B，即-126D

1000 0010B作为无符号数对应+1000 0010B，即+130D

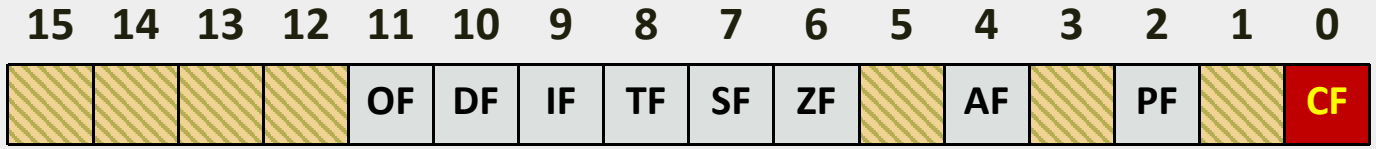
1000 0010B究竟算正数还是负数？

基础：有符号数与补码

- 计算机中有符号数一律用补码来表示和存储。
- 正整数的补码是其二进制表示，与原码相同
  - 例：+9的补码是00001001
- 负整数的补码，将其对应正数二进制的所有位取反（包括符号位，0变1，1变0）后加1
  - 例：-5的补码
    - -5对应正数5（00000101）→所有位取反（11111010）→加1(11111011)
    - 所以-5的补码是11111011。

SF 标志是CPU对有符号数运算结果的一种记录。
将数据当作有符号数来运算的时候，通过SF可知结果的正负；将数据当作无符号数来运算，SF的值则没有意义，虽然相关的指令影响了它的值。

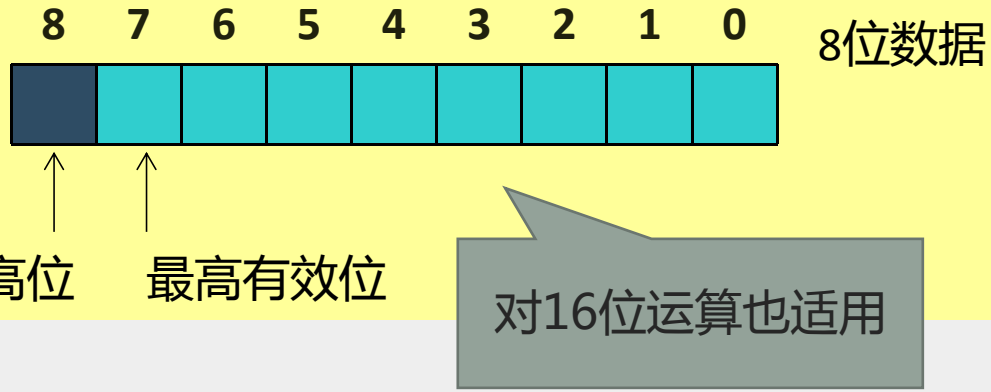
CF-进位标志(Carry Flag)



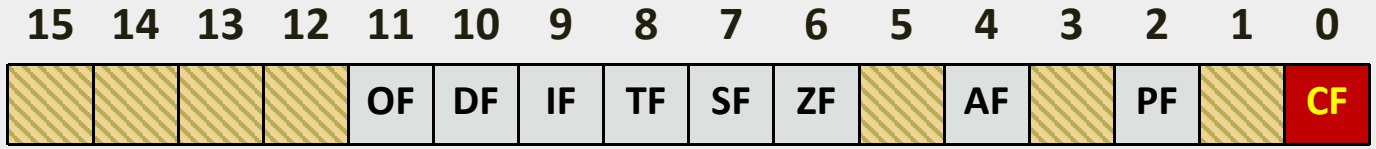
- 在进行无符号数运算的时候，CF记录了运算结果的最高有效位向更高位的进位值，或从更高位的借位值。
- CF记录指令执行后，
  - 有进位或借位，CF = 1
  - 无进位或借位，CF = 0
- 示例

指令	执行结果
mov al,98H add al,al	(al)=30H，CF=1，CF记录了最高有效位向更高位的进位值
add al,al	(al)=60H，CF=0，CF记录了最高有效位向更高位的进位值
sub al,98H	(al)=C8H，CF=1，CF记录了向更高位的借位值

- 对于位数为N的无符号数来说，其对应的二进制信息的最高位即第N-1位，是最高有效位
- 假想存在的第N位，就是相对最高有效位的更高位。



OF-溢出标志(Overflow Flag)



- 在进行有符号数运算的时候，如结果超过了机器所能表示的范围称为**溢出**。
- OF记录有符号数操作指令执行后，
  - 有溢出，OF = 1
  - 无溢出，OF = 0
- 示例

指令	执行结果
mov al,98 add al,99	(al)=197，超出了8位有符号数的范围(-128~127)，OF=1
mov al,0F0H add al,88H	(al)=(-16)+(-120)=-136，有溢出，OF=1

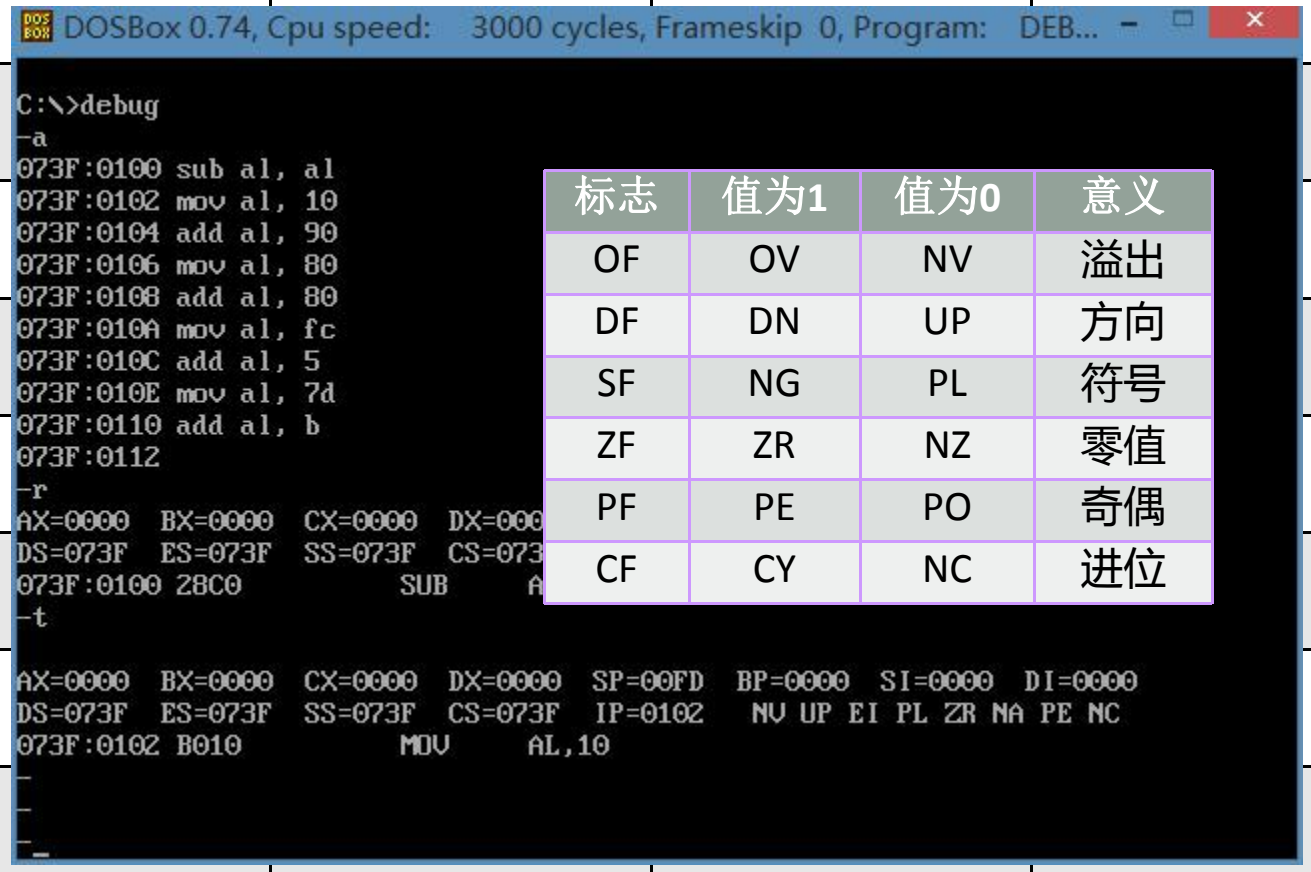
- 机器所能表达的范围
  - 以8位运算为例，结果用 8 位寄存器或内存单元来存放，机器所能表示的范围就是-128~127。
  - 同理，对于16 位有符号数，机器所能表示的范围是-32768~32767。
- 注意，此处溢出只是对有符号数运算而言。

- CF和OF的区别
  - CF是对无符号数运算有意义的进/借位标志位
  - OF是对有符号数运算有意义的溢出标志位
- 应用

指令	执行结果
mov al,0F0H add al,88H	CF=1, OF=1，当无符号数运算有进位，当有符号数运算有溢出

综合：一条指令会带来多个标志寄存器的变化

指令	CF	OF	SF	ZF	PF
sub al, al					
mov al, 10h					
add al, 90h					
mov al, 80h					
add al, 80h					
mov al, 0FCh					
add al, 05h					
mov al, 7Dh					
add al, 0Bh					



标志	值为1	值为0	意义
OF	OV	NV	溢出
DF	DN	UP	方向
SF	NG	PL	符号
ZF	ZR	NZ	零值
PF	PE	PO	奇偶
CF	CY	NC	进位

综合：一条指令会带来多个标志寄存器的变化

指令	CF	OF	SF	ZF	PF
sub al, al	0	0	0	1	1
mov al, 10h					
add al,90h					
mov al, 80h					
add al, 80h					
mov al, 0FCh					
add al, 05h					
mov al 7Dh					
add al, 0Bh					