王爽汇编第11章,标志寄存器

1. 标志寄存器概述

CPU内部的寄存器中，有一种特殊的寄存器（对于不同的处理机，个数和结构都可能不同）具有以下3种作用。

- 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。

这种特殊的寄存器在8086CPU中，`被称为标志寄存器(flag)`。

8086CPU的标志寄存器有16位，其中存储的信息通常被称为程序状态字(PSW Program Status Word)。

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 的 flag 寄存器的结构如图 11.1 所示。

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

图 11.1 flag 寄存器各位示意图

flag的1、3、5、12、13、14、15位在8086CPU中没有使用，不具有任何含义

其他每一位都有特殊含义

在8086CPU中，有的指令是影响标志寄存器的，比如`add、sub、mul、div、inc、or、and(运算指令)`,而向`mov、push、pop(传送指令)`都是对flag寄存器没有影响的。

2. 标志位

2.1. ZF(零标志位)|JE JZ

零标志位 `(Zero Flag)` 。它记录相关指令执行后， `其结果是否为0` 。

如果结果为0，那么zf = 1(表示结果是0)；如果结果不为0，那么zf = 0。

```
;比如判断函数返回值是否为0
call (某函数)

cmp eax,0 ;判断eax是否为0
------------------------
test eax,eax;判断eax是否位0

;如果eax为0   zf标志位 = 1
je xxx (Jump if Equals)
jz xxx (Jump if Zero)

;相反
jne xxx (Jump if not Equals)
jnz xxx (Jump if not Zero)
```

2.2. PF(奇偶标志位)|JP JNP

奇偶标志位 `(Parity Flag)` 。它记录相关指令执行后，其结果的所有bit位中1的个数是否为偶数。

如果1的个数为偶数，pf = 1，如果为奇数，那么pf = 0。

```
mov al, 1
add al, 10 ;执行后，结果为00001011B，其中有3（奇数）个1，则pf = 0;

mov al, 1
or al, 2  ;执行后，结果为00000011B，其中有2（偶数）个1，则pf = 1;
jp xxx (Jump if Parity)  ;偶数为跳转
jnp xxx (Jump if Parity) ;奇数为跳转
```

2.3. SF(符号标志位)|JS JNS

符号标志位 `(Symbol Flag)` 。它记录相关指令执行后，其结果是否为负。

如果 `结果为负，sf = 1` ；如果 `非负，sf = 0` 。

计算机中通常用补码来表示有符号数据。计算机中的一个数据可以看作是有符号数，也可以看成是无符号数。

00000001B，可以看作为无符号数1，或有符号数+1；
10000001B，可以看作为无符号数129，也可以看作有符号数-127。

CPU在执行add等指令的时候，就包含了两种含义:可以将add指令进行的运算当作无符号数的运算，也可以将add指令进行的运算当作有符号数的运算。

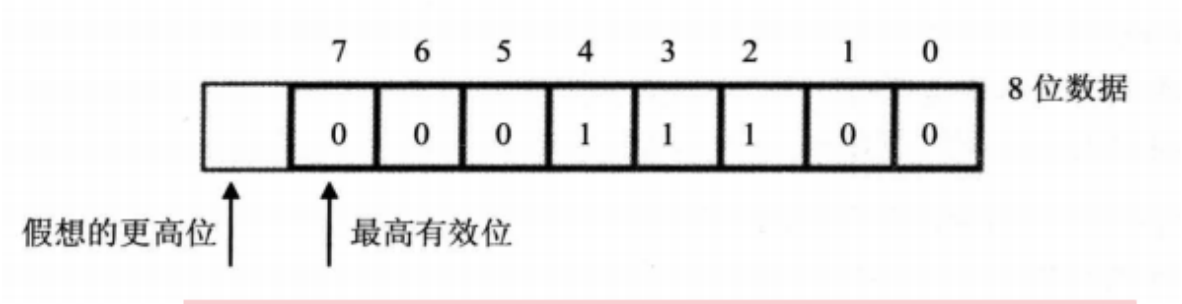
SF标志，就是CPU对有符号数运算结果的一种记录，它记录数据的正负。在我们将数据当作有符号数来运算的时候，可以通过它来得知结果的正负。如果我们将数据当作无符号数来运算，SF的值则没有意义，虽然相关的指令影响了它的值。

```
;负数
mov al, 10000001B
add al, 1   ;执行后，结果为10000010B，sf = 1，表示：如果指令进行的是有符号数运算，那么结果为负;
js  xxx (Jump if Symbol ) ;为负则跳转


;非负数
mov al, 10000001B
add al, 01111111B   ;执行后，结果为0，sf = 0，表示：如果指令进行的是有符号数运算，那么结果为非负
jns  xxx (Jump if not Symbol );不为负则跳转
```

2.4. CF(进位标志位)|JC JNC

进位标志位 (Carry Flag) 。一般情况下，在进行无符号数运算的时候，它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



两个8位数据：98H+98H，将产生进位1。

两个8位数据：97H-98H，将产生借位1。

8086CPU就用flag的CF位来记录这个进位/借位值


```
97H - 98H 产生借位  CF = 1
(al) = 197H - 98H = FFH

;CF = 1
jc  xxx(Jump if  Carry );进位则跳转
;CF = 0
jnc xxx(Jump if not Carry );不进位则跳转
```

2.5. OF(溢出标志位)|JO JNO

溢出标志位 (Overflow Flag) 。一般情况下，OF记录了有符号数运算的结果是否发生了溢出。

如果发生溢出，OF = 1；如果没有，OF = 0。

CF和OF的区别： CF是对无符号 数运算有意义的标志位，而 OF是对有符号 数运算有意义的标志位。

CPU在执行add等指令的时候，就包含了两种含义：无符号数运算和有符号数运算。

- 对于 无符号 数运算，CPU用 CF 位来记录是否产生了进位；
- 对于 有符号 数运算，CPU用 OF 位来记录是否产生了溢出，当然，还要用 SF位来记录结果 的符号。

```
;OF = 1,CF = 0
mov al, 98
add al, 99   ;执行后将产生溢出。因为进行的"有符号数"运算是:
; (al) = (al) + 99 = 197 = C5H 为-59的补码
;而结果197超出了机器所能表示的8位有符号数的范围: -128-127。
                        ;通过SP来得知是否为有符号数
jo xxx (Jmp if Overflow);溢出则跳转
jno xxx (Jmp if not Overflow);不溢出则跳转

;OF = 1,CF = 1
mov al, 0F0H  ;F0H, 为有符号数-16的补码    -Not(F0 - 1)
add al, 088H  ;88H, 为有符号数-120的补码    -Not(88- 1)
;执行后, 将产生溢出。因为add al, 088H进行的有符号数运算结果是:
; (al) = -136
;而结果-136超出了机器所能表示的8位有符号数的范围: -128-127。
;add 指令执行后: 无符号运算有进位CF=1, 有符号运算溢出OF=1
```

2.6. DF(串标志位)|串操作

方向标志位。在串处理指令中，控制每次操作后si、di的增减。

- df = 0每次操作后si、di递增；
- df = 1每次操作后si、di递减。

3. 标志指令

3.1. adc和sbb指令

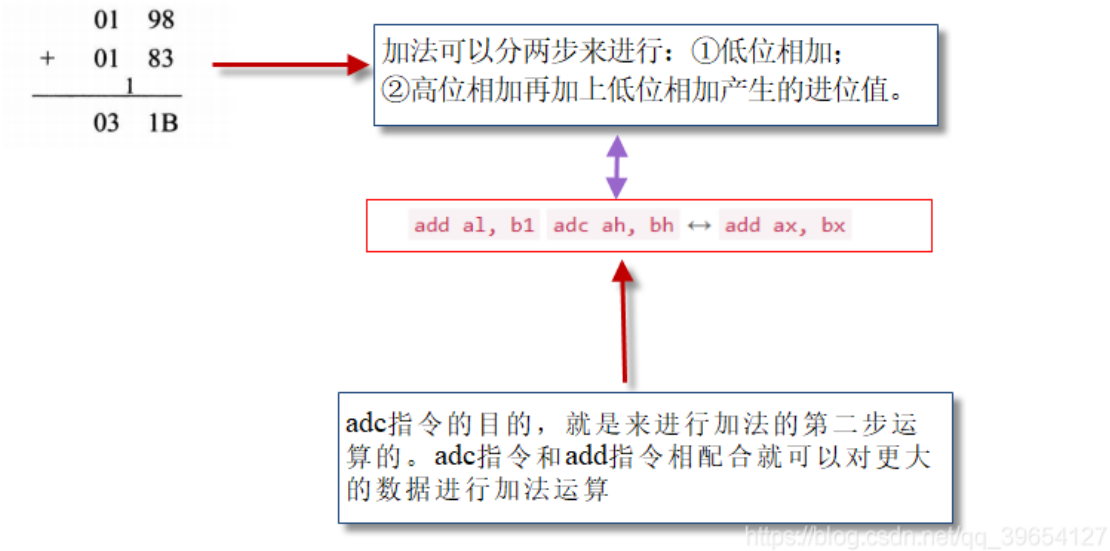
3.1.1. adc指令

adc是带进位加法指令，它利用了CF位上记录的进位值。

指令格式： `adc 操作对象1，操作对象2`

功能：操作对象1 = 操作对象1 + 操作对象2 + CF

```
mov ax, 2
mov bx, 1
sub bx, ax  ;无符号运算借位CF=1，有符号运算OF = 0
adc ax, 1   ;执行后，(ax) = 4。adc执行时，相当于计算：(ax)+1+CF = 2+1+1 = 4。
```



3.1.2. sbb指令

sbb是带借位减法指令，它利用了CF位上记录的借位值。

指令格式： `sbb 操作对象1，操作对象2`

功能：操作对象1 = 操作对象1 - 操作对象2 - CF

```
;计算 003E1000H - 00202000H，结果放在ax，bx中，程序如下：
mov bx, 1000H
mov ax, 003EH
sub bx, 2000H
sbb ax, 0020H
```

3.2. cmp指令

cmp是比较指令，cmp的功能相当于减法指令，只是不保存结果。cmp指令执行后，将对标志寄存器产生影响。

cmp ax, bx	无符号比较时
(ax) = (bx)	zf = 1
(ax) ≠ (bx)	zf = 0
(ax) < (bx)	cf = 1
(ax) ≥ (bx)	cf = 0
(ax) > (bx)	cf = 0 且 zf = 0
(ax) ≤ (bx)	cf = 1 且 zf = 1

3.3. 比较条件转移指令

指令	含义	检测的相关标志位
je	等于则转移	zf = 1
jne	不等于则转移	zf = 0
jb	低于则转移	cf = 1
jnb	不低于则转移	cf = 0
ja	高于则转移	cf = 0 且 zf = 0
jna	不高于则转移	cf = 1 且 zf = 1

3.4. 串传送指令

- 格式： `movsb`
  功能：将ds:si指向的内存单元中的字节送入es:di中，然后根据标志寄存器df位的值，将si和di递增或递减
- 格式： `movsw`
  功能：将ds:si指向的内存字单元中的字送入es:di中，然后根据标志寄存器df位的值，将si和di递增2或递减2。
- 格式： `rep movsb`
  movsb和movsw进行的是串传送操作中的一个步骤，一般来说，movsb和movsw都和rep配合使用，
  功能：rep的作用是根据cx的值， 重复执行 后面的串传送指令

8086CPU提供下面两条指令对df位进行设置。

- `cld` 指令：将标志寄存器的df位置0
- `std` 指令：将标志寄存器的df位置1

```
;将data段中的第一个字符串复制到它后面的空间中。
data segment
    db 'Welcome to masm!'
    db 16 dup (0)
data ends

mov ax, data
mov ds, ax
mov si, 0   ;ds:si 指向data:0
mov es, ax
mov di, 16  ;es:di指向data:0010

mov cx, 16  ; (cx) =16, rep循环16次
cld  ;设置df=0, 正向传送
rep movsb
```

3.5. pushf 和 popf

为直接访问标志寄存器提供了一种方法。

```
pushf ;标志寄存器压栈
popf  ;从栈中弹出数据，送入标志寄存器中
```

4. 跳转功能表

```
JE    ;等于则跳转
JNE   ;不等于则跳转

JZ    ;为 0 则跳转
JNZ   ;不为 0 则跳转

JS    ;为负则跳转
JNS   ;不为负则跳转

JC    ;进位则跳转
JNC   ;不进位则跳转

JO    ;溢出则跳转
JNO   ;不溢出则跳转

JA    ;无符号大于则跳转
JNA   ;无符号不大于则跳转
JAE   ;无符号大于等于则跳转
JNAE  ;无符号不大于等于则跳转

JG    ;有符号大于则跳转
JNG   ;有符号不大于则跳转
JGE   ;有符号大于等于则跳转
JNGE  ;有符号不大于等于则跳转

JB    ;无符号小于则跳转
JNB   ;无符号不小于则跳转
JBE   ;无符号小于等于则跳转
JNBE  ;无符号不小于等于则跳转

JL    ;有符号小于则跳转
JNL   ;有符号不小于则跳转
JLE   ;有符号小于等于则跳转
JNLE  ;有符号不小于等于则跳转

JP    ;奇偶位置位则跳转
```

```
JNP  ;奇偶位清除则跳转
JPE  ;奇偶位相等则跳转
JPO  ;奇偶位不等则跳转
```

图格式：

```
JE   ;等于则跳转
JNE  ;不等于则跳转

JZ   ;为 0 则跳转
JNZ  ;不为 0 则跳转

JS   ;为负则跳转
JNS  ;不为负则跳转

JC   ;进位则跳转
JNC  ;不进位则跳转

JO   ;溢出则跳转
JNO  ;不溢出则跳转

JA   ;无符号大于则跳转
JNA  ;无符号不大于则跳转
JAE  ;无符号大于等于则跳转
JNAE ;无符号不大于等于则跳转

JG   ;有符号大于则跳转
JNG  ;有符号不大于则跳转
JGE  ;有符号大于等于则跳转
JNGE ;有符号不大于等于则跳转

JB   ;无符号小于则跳转
JNB  ;无符号不小于则跳转
JBE  ;无符号小于等于则跳转
JNBE ;无符号不小于等于则跳转

JL   ;有符号小于则跳转
JNL  ;有符号不小于则跳转
JLE  ;有符号小于等于则跳转
JNLE ;有符号不小于等于则跳转

JP   ;奇偶位置位则跳转
JNP  ;奇偶位清除则跳转
JPE  ;奇偶位相等则跳转
JPO  ;奇偶位不等则跳转
```

> 参考文章：
> https://blog.csdn.net/qq_39654127/article/details/88698911 《王爽《汇编语言》笔记（详细）》
> https://blog.csdn.net/u010326355/article/details/12762345 汇编语言—跳转指令：JMP、JECXZ、JA、JB、JG、JL、JE、JZ、JS、JC、JO、JP
> http://c.biancheng.net/view/3567.html 《C语言学习网》
> http://www.xumenger.com/asm-cmp-je-jne-jmp-20180205/ 《8086汇编语言中的比较和跳转指令》

学逆向的话看完这十一章差不多了，我是先学的逆向，当时没有学汇编 才发现原来汇编对逆向是那么的重要，了解汇编后 再回头来看OD动态调试 好多以前不懂的东西 豁然开朗了。

有兴趣的小伙伴可以加群：一起讨论逆向、PWN甚至是Web安全 群内有web大佬 ai大佬 pwn大佬，只有我这个re菜鸡哈哈

群名称:Pwn菜鸡学习小分队
群　号:1145528880