

汇编语言程序设计

进位和溢出标志



状态标志

➤ 状态标志是处理器最基本的标志

- ▶ 一方面：作为加减运算和逻辑运算的辅助结果
- ▶ 另一方面：构成各种条件，实现程序分支

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

8086的标志



进位标志CF（Carry Flag）

- 当加减运算结果的最高有效位有进位（加法）或借位（减法）时，进位标志置1，即**CF=1**；否则**CF=0**
- 针对无符号整数，判断加减结果是否超出表达范围

无符号整数的表达范围

N位	8位	16位	32位
$0 \sim 2^N - 1$	$0 \sim 255$	$0 \sim 65535$	$0 \sim 2^{32} - 1$



进位标志CF举例—1

8位二进制

$$\begin{array}{r} 00111010 \\ + 01111100 \\ \hline 10110110 \end{array}$$

十六进制

$$\begin{array}{r} 3A \\ + 7C \\ \hline B6 \end{array}$$

十进制

$$\begin{array}{r} 58 \\ + 124 \\ \hline 182 \end{array}$$

$0 < 182 < 255$



结果: 182, 无进位 $CF=0$



进位标志CF举例—2

8位二进制

$$\begin{array}{r} 10101010 \\ + 01111100 \\ \hline 100100110 \end{array}$$

十六进制

$$\begin{array}{r} AA \\ + 7C \\ \hline 126 \end{array}$$

十进制

$$\begin{array}{r} 170 \\ + 124 \\ \hline 294 = 256 + 38 \end{array}$$

255 < 294

结果：38，有进位CF=1

低位部分



溢出标志OF（Overflow Flag）

- 有符号数加减结果有溢出，则**OF=1**；否则**OF=0**
- 针对有符号整数，判断加减结果是否超出表达范围

有符号整数（补码）的表达范围

N位	8位	16位	32位
$-2^{N-1} \sim 2^{N-1}-1$	$-128 \sim 127$	$-32768 \sim 32767$	$-2^{31} \sim 2^{31}-1$

杯中水已满，再加就溢出！



溢出标志OF举例—1

8位二进制

$$\begin{array}{r} 00111010 \\ + 01111100 \\ \hline 10110110 \end{array}$$

十六进制

$$\begin{array}{r} 3A \\ + 7C \\ \hline B6 \end{array}$$

十进制

$$\begin{array}{r} 58 \\ + 124 \\ \hline 182 \end{array}$$

127 < 182

结果: -74, 超范围 OF=1

错误



溢出标志OF举例—2

8位二进制	十六进制	十进制
$\begin{array}{r} 10101010 \\ + 01111100 \\ \hline 100100110 \end{array}$	$\begin{array}{r} AA \\ + 7C \\ \hline 126 \end{array}$	$\begin{array}{r} -86 \\ + 124 \\ \hline 38 \end{array}$

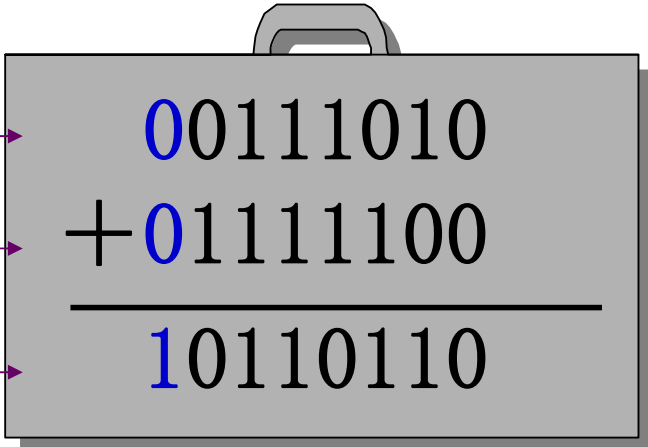
$$-128 < 38 < 127$$

结果: 38, 范围内 OF=0



溢出标志的人工判断

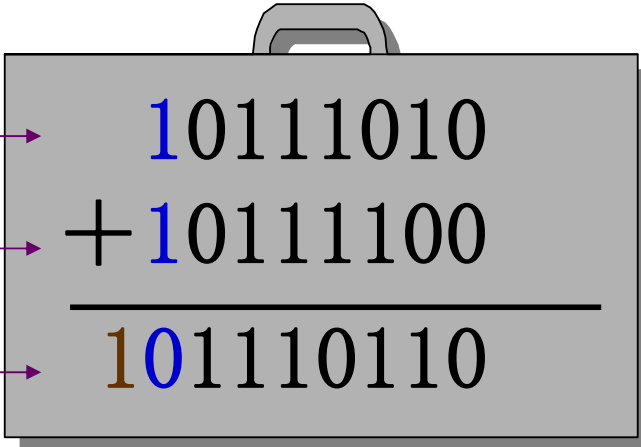
- 只有当两个相同符号数相加(含两个不同符号数相减)
- 而运算结果的符号与原数据符号相反时，产生溢出



正数 → 00111010
正数 → +01111100

负数 → 10110110

This diagram shows a briefcase containing a binary addition. The first operand is 00111010, labeled '正数' (positive). The second operand is +01111100, also labeled '正数'. The result is 10110110, labeled '负数' (negative). The result's sign is opposite to the operands' signs, indicating an overflow.



负数 → 10111010
负数 → +10111100

正数 → 101110110

This diagram shows a briefcase containing a binary addition. The first operand is 10111010, labeled '负数' (negative). The second operand is +10111100, also labeled '负数'. The result is 101110110, labeled '正数' (positive). The result's sign is opposite to the operands' signs, indicating an overflow.

- 其他情况下，不会产生溢出



进位和溢出的区别

- 进位标志反映无符号整数运算结果是否超出范围
 - ▶ 有进位，加上进位或借位后运算结果仍然正确
- 溢出标志反映有符号整数运算结果是否超出范围
 - ▶ 有溢出，运算结果已经不正确
- 处理器按照无符号整数求得结果
 - ▶ 设置进位标志**CF**
 - ▶ 设置溢出标志**OF**

程序员决定

无符号数，关心进位
有符号数，注意溢出

汇编语言程序设计

零标志、符号标志和奇偶标志



状态标志

➤ 状态标志是处理器最基本的标志

- ▶ 一方面：作为加减运算和逻辑运算的辅助结果
- ▶ 另一方面：构成各种条件，实现程序分支

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

8086的标志



零标志ZF (Zero Flag)

➤ 运算结果为0，则**ZF=1**，否则**ZF=0**

结果是0，ZF标志不是0！

举例

➤ 8位二进制数相加：结果不是0，**ZF=0**

$$00111010 + 01111100 = 10110110$$

➤ 8位二进制数相加：结果是0，**ZF=1**

$$10000100 + 01111100 = [1]00000000$$

进位

结果

符号标志SF (Sign Flag)

- 运算结果最高位为**1**，则**SF=1**；否则**SF=0**

最高位 = 符号位 = SF

举例

- 8位二进制数相加：最高位=**1**，**SF=1**

$$00111010 + 01111100 = 10110110$$

- 8位二进制数相加：最高位=**0**，**SF=0**

$$10000100 + 01111100 = [1]00000000$$

进位

结果



奇偶标志PF（Parity Flag）

- 当运算结果最低字节中“1”的个数为零或偶数时，
PF=1；否则**PF=0**

仅最低8位“1”的个数

举例

- 8位二进制数相加：“1”的个数为5个，**PF=0**

$$00111010 + 01111100 = 10110110$$

- 8位二进制数相加：“1”的个数为0个，**PF=1**

$$10000100 + 01111100 = [1]00000000$$

进位

结果



加减运算结果同时影响状态标志

8位加法运算及结果	CF	OF	ZF	SF	PF
$00111010 + 01111100 = [0]10110110$	0	1	0	1	0
$10101010 + 01111100 = [1]00100110$	1	0	0	0	0
$10000100 + 01111100 = [1]00000000$	1	0	1	0	1



影响状态标志的指令

- 需要关注对标志影响的主要指令：
加减运算指令、逻辑运算指令、移位指令等
- 只用于影响标志的特殊指令1：比较指令CMP
 - ▶ 进行减法运算
 - ▶ 用于判断两个数据大小、是否相等
- 只用于影响标志的特殊指令2：测试指令TEST
 - ▶ 进行逻辑与运算
 - ▶ 用于判断某位为0或为1等



汇编语言程序设计

ADD指令



算术运算类指令

- 算术运算对数据进行加减乘除
- 这是基本的数据处理方法
- 注意，加减运算有“和”或“差”的结果外，
还有进借位、溢出等状态标志，也是结果的一部分



加法指令

- 加法指令 **ADD**
 - 带进位加法指令 **ADC**
 - 增量指令 **INC**
 - ▶ 除**INC**不影响进位标志**CF**外
 - ▶ 其他指令按定义影响全部状态标志位
- 即，按照运算结果相应设置各个状态标志为**0**或为**1**



加法指令ADD

- 目的操作数加上源操作数，和送到目的操作数

ADD reg, imm/reg/mem

;reg←reg+imm/reg/mem

ADD mem, imm/reg

;mem←mem+imm/reg

- ▶ 按照定义影响状态标志位
- ▶ 支持8位(字节)、16位(字)和32位(双字)数据运算



ADD指令：8位加法

mov eax,0aaff7348h

add al,27h

;8位加法

;最高位是D₇

加法之前EAX

AAFF7348H

8位加法

+27H

加法之后EAX

AAFF736FH

0110 1111

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



ADD指令：16位加法

add ax,3fffh

;16位加法

;最高位是D₁₅

加法之前EAX	AAFF736FH
16位加法	+3FFFH
加法之后EAX	AAFFB36EH

1011 0011 0110 1110

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



ADD指令：32位加法

add eax,88000000h

;32位加法

;最高位是D₃₁

加法之前EAX	AAFFB36EH
32位加法	+88000000H
加法之后EAX	32FFB36EH

0011 0010 1111 1111 1011 0011 0110 1110

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



大写字母转换为小写可以用ADD

;确认是大写字母， 加**20H**为小写字母

add al,20h ;20H='a' - 'A'

;20H=' '(空格字符)

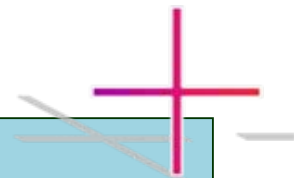
小写 = 大写 + 20H

大写 = 小写 - 20H

字母	ASCII值
'A'	41H
'B'	42H
...	...
'a'	61H
'b'	62H
...	...

本讲总结

- **ADD**指令是最基本的加法指令
- 支持**8**、**16**和**32**位加法运算
 - ▶ 并按照**8**、**16**和**32**位相应影响状态标志
 - ▶ 但**PF**标志只利用低**8**位结果



```
ADD dest, src  
; dest ← dest + src
```



汇编语言程序设计

SUB指令



算术运算类指令

- 算术运算对数据进行加减乘除
- 这是基本的数据处理方法
- 注意，加减运算有“和”或“差”的结果外，
还有进借位、溢出等状态标志，也是结果的一部分



减法指令

- 减法指令 **SUB**
- 带借位减法指令 **SBB**
- 减量指令 **DEC**
- 求补指令 **NEG**
- 比较指令 **CMP**
 - ▶ 除**DEC**不影响**CF**标志外
 - ▶ 其他按定义影响全部状态标志位



减法指令SUB (subtract)

- 目的操作数减去源操作数，差送到目的操作数

SUB reg,imm/reg/mem

;reg←reg—imm/reg/mem

SUB mem,imm/reg

;mem←mem—imm/reg



- ▶ 按照定义影响状态标志位
- ▶ 支持**8位(字节)**、**16位(字)**和**32位(双字)**数据运算



SUB指令：8位减法

mov eax,0aaff7348h

sub al,27h

;8位减法

;最高位是D₇

减法之前EAX

AAFF7348H

8位减法

-27H

减法之后EAX

AAFF7321H

0010 0001

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



SUB指令：16位减法

sub ax,3fffh

;16位减法

;最高位是D₁₅

减法之前EAX

AAFF7321H

16位减法

-3FFFH

减法之后EAX

AAFF3322H

0011 0011 0010 0010

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



SUB指令：32位减法

sub eax,0bb000000h

;**32位减法**

;**最高位是D₃₁**

减法之前EAX	AAFF3322H
32位减法	-BB000000H
减法之后EAX	EFFF3322H

1110 1111 1111 1111 0011 0011 0010 0010

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



小写字母转换为大写可以用SUB

;确认是小写字母， 减**20H**为大写字母

sub al,20h ;20H='a' - 'A'

;20H=' '(空格字符)

大写 = 小写 - 20H

小写 = 大写 + 20H

字母

ASCII值

'A'

41H

'B'

42H

...

...

'a'

61H

'b'

62H

...

...



本讲总结

- **SUB**指令是最基本的减法指令
- 支持**8**、**16**和**32**位减法运算
 - ▶ 并按照**8**、**16**和**32**位相应影响状态标志
 - ▶ 但**PF**标志只利用低**8**位结果

SUB dest, src
; dest ← dest - src



汇编语言程序设计

INC、DEC和NEG指令



增量指令INC (increment)

- 只有一个操作数：寄存器或存储单元
- 对操作数加1（增量）再将结果返回原处

i++

INC reg/mem ;加1: $\text{reg/mem} \leftarrow \text{reg/mem} + 1$

- 用于计数器和地址指针的调整
 - ▶ 不影响进位**CF**标志，影响其他状态标志位

```
inc ecx
```

```
inc dword ptr [ebx]
```

```
inc wvar
```

```
inc bl
```

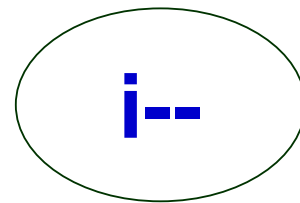
```
inc word ptr [esi]
```

```
inc wvar[edi]
```



减量指令DEC (decrement)

- 只有一个操作数：寄存器或存储单元
- 对操作数减1（减量）再将结果返回原处



DEC reg/mem ;减1: $\text{reg/mem} \leftarrow \text{reg/mem} - 1$

- 用于计数器和地址指针的调整
 - ▶ 不影响进位**CF**标志，影响其他状态标志位

dec cx

dec byte ptr [ebx]

dec wvar

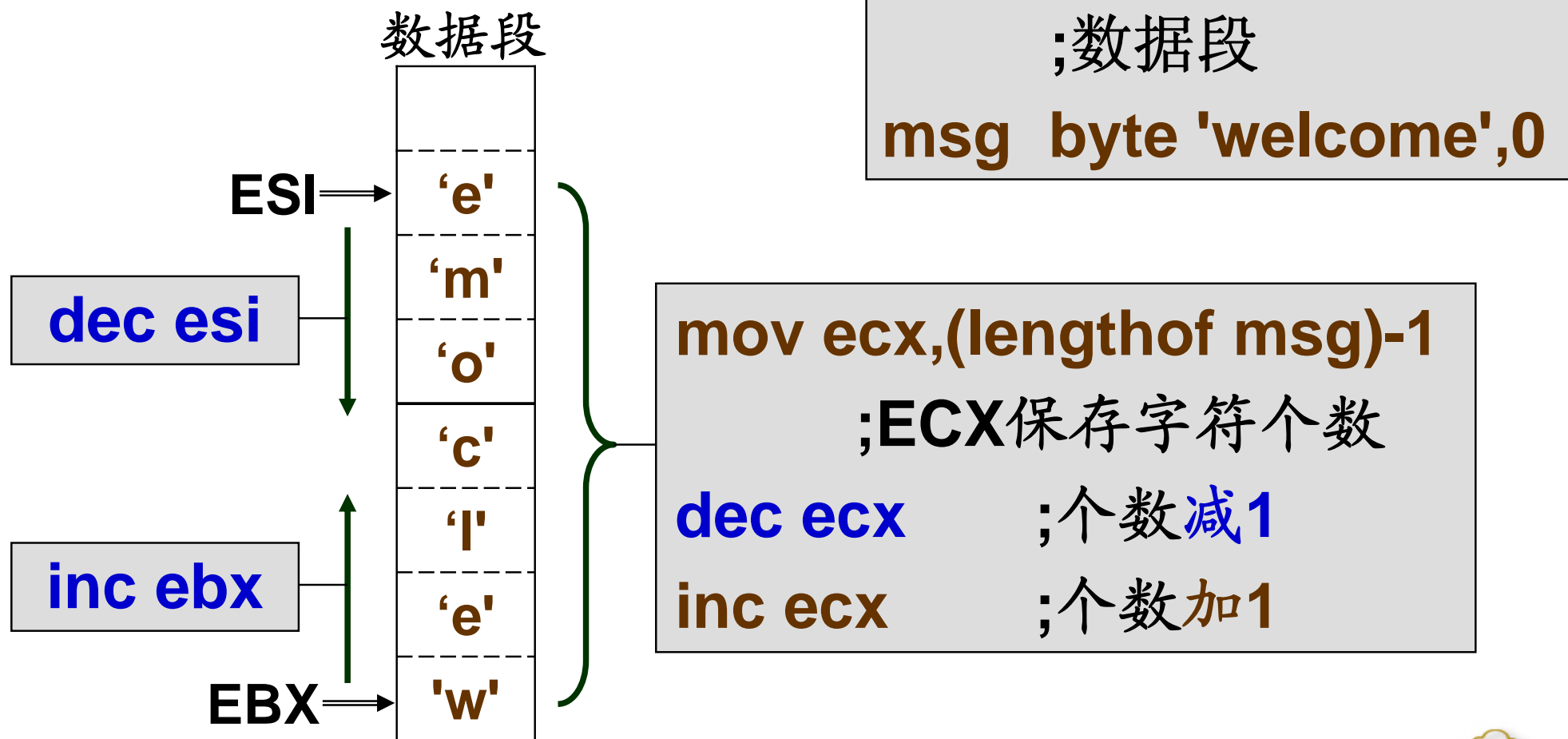
dec bl

dec word ptr [esi]

dec wvar[edi]



INC和DEC用于指针调整和计数增减



求补指令NEG (negative)

- 对操作数执行求补运算，即用零减去操作数
NEG reg/mem ; reg/mem $\leftarrow 0 - \text{reg/mem}$
- 对标志的影响与用零作减法的**SUB**指令一样
- 可用于对负数求补码或由补码求其绝对值

0-i

```
neg al  
neg byte ptr [ebx]  
neg wvar[esi]
```

```
neg ax  
neg word ptr [ebx]  
neg wvar[edi]
```



NEG 用于负数的求补运算

;已知100的8位编码, 求-100的8位补码

mov al,64h ;AL=64H =100

neg al ;AL=0-64H =9CH= -100

;OF=0, SF=1, ZF=0, PF=1, **CF=1**

;已知-100的32位补码, 求其绝对值 (即100)

mov eax,0ffffff9ch ;EAX=FFFFFFFF9CH= -100

neg eax ;EAX=0-FFFFFFFF9CH =64H =100

;OF=0, SF=0, ZF=0, PF=0, **CF=1**



本讲总结

- **INC、DEC和NEG**是加减运算的辅助指令
 - ▶ **INC**实现指针加1（与**ADD**加1功能相同）、不影响**CF**
 - ▶ **DEC**实现指针减1（与**SUB**减1功能相同）、不影响**CF**
 - ▶ **NEG**进行数据求补（使用**0**减功能实现）
- 这些指令都只给出一个操作数位置
 - ▶ 既是源操作数，也是目的操作数

