

## 第5章 算术和逻辑运算指令

罗文坚  
中国科大 计算机学院

<http://staff.ustc.edu.cn/~wjluo/mcps/>

1

## 本章内容

- 加法、减法和比较指令
- 乘法和除法指令
- **BCD**码和**ASCII**码算术运算指令
- 基本逻辑运算指令
- 移位和循环移位指令
- 串比较指令

2

## 加法指令

加法	格式	ADD REG/MEM, REG/MEM/IMM
	功能	源操作数、目的操作数相加，结果存入目的操作数
	标志	所有状态标志（ZF、CF、PF、AF、SF及OF）都受影响

带进位加	格式	ADC REG/MEM, REG/MEM/IMM
	功能	源操作数、目的操作数以及进位标志CF相加，结果存入目的操作数
	标志	所有状态标志（ZF、CF、PF、AF、SF及OF）都受影响

3

## 加法指令

加1	格式	INC REG/MEM
	功能	目的操作数加1
	标志	除CF标志位，其余状态标志都受影响

交换并相加	格式	XADD REG/MEM, REG
	功能	（80486以上）源操作数和目的操作数相交换，并将两者之和存入目的操作数
	标志	所有状态标志都受影响，根据加法结果设置

4

## 加法指令

- 加法指令注意事项：
  1. 源操作数和目的操作数不能同时为内存单元（MEM）。
  2. 不允许与段寄存器（SREG）相关的加法。
  3. XADD指令的源操作数在寄存器（REG）中。
  4. 标志寄存器中状态位随运算结果而变化，但INC指令不影响CF标志。
  5. 指令中操作数是带符号数还是无符号数由程序员解释。

□注意：第4章的数据传送指令不改变状态标志。

5

## Example 1

- 例、试用加法指令对两个8位16进制数5EH和3CH求和，并分析加法运算指令执行后对标志位的影响。
- 解：

MOV AL, 5EH ;AL=5EH (94)	0101 1110
MOV BL, 3CH ;BL=3CH (50)	+ 0011 1100
ADD AL, BL ;结果AL=9AH	1001 1010

运算后标志：ZF=0, AF=1, CF=0, SF=1, PF=1, OF=1。
- 若程序员认为两个加数是无符号数，则运算结果位9AH，即154。
  - ✓ 此时，SF标志和OF标志没有意义。
- 若程序员认为两个加数是有符号数，则运算溢出，结果无效。
  - ✓ 此时，CF标志没有意义。

6

## CF标志和OF标志

- 当加减运算结果的最高有效位有进位（加法）或借位（减法）时，CF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。
  - N个二进制位表达无符号整数的范围： $0 \sim 2^N - 1$
- 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。
  - N个二进制位表达有符号整数的范围： $-2^{N-1} \sim 2^{N-1} - 1$

7

## 进位标志CF：举例

- 8位二进制数相加：  
 $00111010 + 01111100 = 10110110$
  - 十六进制表达： $3A + 7C = B6$
  - 转换成十进制数： $58 + 124 = 182$
  - 没有产生进位：CF=0
- $0 < 182 < 255$
- 
- 8位二进制数相加：  
 $10101010 + 01111100 = [1]00100110$
  - 十六进制表达： $AA + 7C = [1]26$
  - 转换成十进制数： $170 + 124 = 294 = 256 + 38$
  - 产生进位：CF=1
- 进位1表达256

8

## 溢出标志OF：举例

- 8位二进制数相加：  
 $00111010 + 01111100 = 10110110$
  - 十六进制表达： $3A + 7C = B6$
  - 转换成十进制数： $58 + 124 = 182$
  - 超出范围：OF=1
- $182 > 127$
- 
- 8位二进制数相加：  
 $10101010 + 01111100 = [1]00100110$
  - 十六进制表达： $AA + 7C = [1]26$
  - 转换成十进制数： $-86 + 124 = 38$
  - 没有超出范围：OF=0
- 补码AAH表达-86

9

## Example 2

- 设一个学生的三门课的成绩分别为60、65、90，入学分数线为总分256分，判断该学生是否取得入学资格。  
；采用无符号数表示  
MOV AL, 60  
ADD AL, 65  
ADD AL, 90  
JC PASS; 超过256分?  
.....  
PASS;  
.....; 取得入学资格
- 设张三在海拔60米的地点，他先往上走了65米，然后又往上走了90米，请问他现在所在地点的海拔高度？  
；为便于表示低于海平面的情况，采用有符号数表示  
MOV AL, 60  
ADD AL, 65  
JO ERROR; (AL)=01111101  
ADD AL, 90  
JO ERROR; (AL)=11010111  
.....  
ERROR;  
.....; 错误处理

10

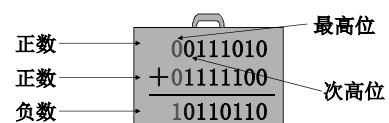
## 进位和溢出的区别

- 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
  - 在设置进位标志CF的同时，根据是否超出有符号整数的范围设置溢出标志OF。
- 应该利用哪个标志，由程序员决定！
  - 操作数是无符号数，关心进位
  - 操作数是有符号数，注意溢出

11

## 溢出标志的判断

- 处理器硬件判断规则
  - 最高位和次高位同时有进位或同时无进位，无溢出；最高位和次高位进位状态不同，有溢出
- 人工判断的简单规则
  - 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；其他情况下，不会产生溢出



12

## 奇偶标志PF (Parity Flag)

- 当运算结果最低8位中“1”的个数为零或偶数时，PF=1；否则PF=0

举例

- 8位二进制数相加：  
 $00111010 + 01111100 = 10110110$   
“1”的个数为5个：PF=0
- 8位二进制数相加：  
 $10000100 + 01111100 = [1]00000000$   
“1”的个数为0个：PF=1

进位 结果

13

## 零标志ZF (Zero Flag)

- 运算结果为0，则ZF=1，否则ZF=0

举例

- 8位二进制数相加：  
 $00111010 + 01111100 = 10110110$   
结果不是0，ZF=0
- 8位二进制数相加：  
 $10000100 + 01111100 = [1]00000000$   
结果是0，ZF=1

结果是0，  
ZF标志不是0！

进位 结果

14

## 符号标志SF (Sign Flag)

- 运算结果最高位为1，则SF=1；否则SF=0。

举例

- 8位二进制数相加：  
 $00111010 + 01111100 = 10110110$   
最高位=1：SF=1
- 8位二进制数相加：  
 $10000100 + 01111100 = [1]00000000$   
最高位=0：SF=0

进位 结果

15

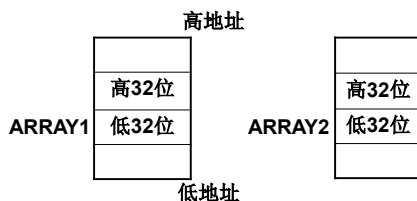
## 辅助进位标志 (Auxiliary Carry)

- AF (Auxiliary Carry)：辅助进位标志。
  - 用于标志D3向D4位之间的进位（加法运算）或借位（减法运算）的状态。
  - AF标志供DAA和DAS指令使用，以便在BCD码的加法或减法之后对AL中的结果值进行十进制调整。

16

## Example 3

- 例、32位CPU中，实现64位加法。
  - 把ARRAY1地址开始的四个字(低字在前)和ARRAY2地址开始的四个字相加，和存放在ARRAY1开始处。



17

## 程序段

- MOV ESI, OFFSET ARRAY1；取第一个数的首地址
- MOV EAX, [ESI]；将第一个数的低32位送AX
- MOV EDI, OFFSET ARRAY2；取第二个数的首地址
- ADD EAX, [EDI]；第一个数的低32位和第2个数的低32位相加（不加CF，但此条指令的执行影响CF）
- MOV [ESI], EAX；存低32位相加结果
- MOV EAX, [ESI+4]；
- ADC EAX, [EDI+4]；两个高32位连同CF（低32位相加形成的）相加。
- MOV [ESI+4], EAX；存高32位相加结果。

思考：128位整数相加/减？

18

## Example 4

- MOV BL, 12H
- MOV DL, 02H
- XADD BL, DL; BL=14H, DL=12H

19

## 减法指令

减法	格式	SUB REG/MEM, REG/MEM/IMM
	功能	目的操作数—源操作数，结果存入目的操作数
	标志	所有状态标志（ZF、CF、PF、AF、SF及OF）都受影响
带借位减	格式	SBB REG/MEM, REG/MEM/IMM
	功能	目的操作数—源操作数—进位标志CF，结果存入目的操作数
	标志	所有状态标志都受影响
减1	格式	DEC REG/MEM
	功能	目的操作数减1
	标志	除CF标志位，其余状态标志都受影响

20

## 减法指令

- 减法指令注意事项与加法指令类似：
  1. 源操作数和目的操作数不能同时为内存单元（MEM）。
  2. 不允许与段寄存器（SREG）相关的加法。
  3. 标志寄存器中状态位随运算结果而变化，但DEC指令不影响CF标志。
  4. 指令中操作数是带符号数还是无符号数由程序员解释。

21

## Example

- SUB CL, BL
- SUB DH, 4FH
- SUB AX, SP
- SUB DI, TEMP[ESI]
- DEC QWORD PTR [RSI]
- SBB BYTE PTR [DI], 3

22

## 比较指令

比较	格式	CMP REG/MEM, REG/MEM/IMM
	功能	目的操作数减去源操作数 源操作数、目的操作数相减不能同时为内存单元。
	标志	影响 ZF、CF、PF、AF、SF 及 OF。
比较并交换	格式	CMPXCHG REG/MEM, REG
	功能	(80486以上) 比较目的操作数与AL、AX、EAX或RAX寄存器中的值。如果两个值相等，则将源操作数加载到目的操作数。否则，将目标操作数加载到AL、AX、EAX或RAX。
	标志	如果目标操作数与AL、AX、EAX或RAX中的值相等，则置ZF标志，否则清除此标志。CF、PF、AF、SF及OF标志根据比较操作的结果设置。

23

## 比较指令

比较并交换8字节	格式	CMPXCHG8B MEM64
	功能	(Pentium以上) 比较目的操作数和EDX:EAX中的64位值。如果这两个值相等，则将ECX:EBX中的64位值存储到目的操作数。否则，将目标操作数的值加载到EDX:EAX。目标操作数是8字节内存位置。EDX与ECX包含64位值的32个高位，EAX与EBX包含32个低位。
	标志	如果两值相等，ZF=1，否则ZF=0；CF、PF、AF、SF及OF标志不受影响。
比较并交换16字节	格式	CMPXCHG16B MEM128
	功能	与CMPXCHG8B类似，但寄存器为RDX:RAX，RCX:RBX。但是，要求MEM128是16-byte对齐。
	标志	与CMPXCHG16B一样。

24

## Example

- **CMP CL, BL**
- **CMP EBP, ESI**
- **CMP RDI, RSI**
- **CMP AX, 2000H**
- **CMP R10W, 12H**
- **CMP [DI], CH**
- **CMP DI, TEMP[BX]**
- **CMPCHG CX, DX**
- **CMPXCHG8B TEMP**

25

## 本章内容

- 加法、减法和比较指令
- 乘法和除法指令
- **BCD码和ASCII码算术运算指令**
- 基本逻辑运算指令
- 移位和循环移位指令
- 串比较指令

26

## 乘法指令

无符号乘法	格式	<b>MUL REG/MEM</b>
	功能	8位：源操作数与AL相乘，乘积在AX 16位：源操作数与AX相乘，乘积在DX:AX 32位：源操作数与EAX相乘，乘积在EDX:EAX 64位：源操作数与RAX相乘，乘积在RDX:RAX
	标志	影响 OF 及 CF，其余状态标志ZF、PF、AF、SF 不确定。 当结果（乘积）的高半部分=0时，CF←0，OF←0，表示高半部分是无效数字；否则CF←1，OF←1。
	注意	SRC可以是寄存器或存储单元，但不能是立即数。

27

## 乘法指令

有符号乘法	格式	<b>1个操作数：IMUL REG/MEM</b>
	功能	8位：源操作数与AL相乘，乘积在AX 16位：源操作数与AX相乘，乘积在DX:AX 32位：源操作数与EAX相乘，乘积在EDX:EAX 64位：源操作数与RAX相乘，乘积在RDX:RAX
	标志	影响 OF 及 CF，其余状态标志ZF、PF、AF、SF 不确定。 如果乘积的高半部分不是低半部分的符号扩展（不是全0或全1），则视高半部分为有效位，置CF=1，OF=1； 如果结果的高半部分是全0或全1，表明它仅包含符号位，置CF=0，OF=0；

28

## 乘法指令

有符号乘法	格式	<b>2个操作数：IMUL REG, REG/MEM/IMM</b>
	功能	目的操作数和源操作数相乘，乘积放在目的操作数。
	标志	• 影响 OF 及 CF，其余状态标志ZF、PF、AF、SF 不确定。 • 当结果必须截断以放在目的操作数时，OF和CF为1，否则为0。
	注意	• 8086不支持 • 目的操作数不能是8位寄存器。 • 立即数不能是64位。 • 源操作数和目的操作数长度相同。当IMM的长度不足时，进行符号扩展。

29

## 乘法指令

有符号乘法	格式	<b>3个操作数：IMUL REG, REG/MEM, IMM</b>
	功能	第2个操作数与IMM相乘，乘积在第1个操作数中。
	标志	• 影响 OF 及 CF，其余状态标志ZF、PF、AF、SF 不确定。 • 当结果必须截断以放在目的操作数时，OF和CF为1，否则为0。
	注意	• 8086不支持 • 第1个操作数不能是8位寄存器。 • 立即数不能是64位。 • 三个操作数长度相同。当IMM的长度不足时，进行符号扩展。

30

## Example

- MUL CL
- IMUL BYTE PTR [BX]
- IMUL TEMP
- IMUL WORD PTR [SI]
- IMUL BX, NUMBER, 1000H
- IMUL RCX

31

## Example

- 例1、设AL=55H, BL=14H, 计算它们的乘积。
  - MUL BL
  - 结果: AX=06A4H。由于AH=06H, 不为0, 则CF=1, OF=1。
- 例2、AL=-28H, BL=59H, 计算它们的乘积。
  - IMUL BL
  - 结果: AX=0F98CH, CF=1, OF=1。

32

## MUL与IMUL

- 能否用MUL做带符号数的乘法?
- 例、尝试用MUL计算FFH×FFH。
  - 用二进制进行计算, 可表示为:

```

      1111 1111
    × 1111 1111
    ──────────
    1111 1110 0000 0001
  
```

- 若为无符号数, 则相当于 $255 \times 255 = 65025$ 的运算, 结果正确。
- 若为有符号数, 则上面的结果表示为 $(-1) \times (-1) = -511$ , 结果不正确。

33

## MUL与IMUL

- IMUL指令采用什么算法来实现其功能?
  - 有多种方案。
    - 可以直接采用补码相乘。
    - 也可以先将参加运算的操作数恢复成原码, 数位当成符号数相乘, 然后给乘积赋予正确的符号。
  - 这些工作由微处理器自动完成。

34

## 除法指令

无符号乘法	格式	DIV REG/MEM
	功能	<ul style="list-style-type: none"> <li>• 8位: AX除以源操作数, 商在AL, 余数在AH</li> <li>• 16位: DX:AX除以源操作数, 商在AX, 余数在DX</li> <li>• 32位: EDX:EAX除以源操作数, 商在EAX, 余数在EDX</li> <li>• 64位: RDX:RAX除以源操作数, 商在RAX, 余数在RDX</li> </ul>
	标志	所有状态标志OF、CF、ZF、PF、AF、SF均不确定。

35

## 除法指令

有符号除法	格式	IDIV REG/MEM
	功能	<ul style="list-style-type: none"> <li>• 8位: AX除以源操作数, 商在AL, 余数在AH</li> <li>• 16位: DX:AX除以源操作数, 商在AX, 余数在DX</li> <li>• 32位: EDX:EAX除以源操作数, 商在EAX, 余数在EDX</li> <li>• 64位: RDX:RAX除以源操作数, 商在RAX, 余数在RDX</li> </ul>
	标志	所有状态标志OF、CF、ZF、PF、AF、SF均不确定。
	注意	商的符号符合一般代数符号规则, 余数的符号与被除数相同。

36

## 除法指令

- 除法指令可能发生两种错误
  - 除数为0
  - 除法溢出
    - 除法溢出：在被除数很大，而除数很小时，会发生除法溢出。
      - 例如，AX=3000，除数BL=2，此时商在AL=1500使得除法溢出。
- 这两种错误都会使得微处理器产生中断。此时所得的商和余数都不确定。
- ❖ 在任何微处理器中，都不存在立即数除法指令。

37

## 乘法指令、除法指令与标志位

- 标志：O D I T S Z A P C
 

IMUL	×	---	U	U	U	U	X
MUL	×	---	U	U	U	U	X
IDIV	U	---	U	U	U	U	U
DIV	U	---	U	U	U	U	U
- X：根据结果设置。当结果（乘积）的高半部分=0时，CF←0，OF←0，表示高半部分无有效数字；否则，CF←1，OF←1。
- U：无定义。
- ：不影响。

38

## Example

- 例、给定无符号数7A86H和04H，求7A86H÷04H=?。
  - 若用DIV指令进行计算，即
 

```
MOV AX, 7A86H
MOV BL, 04H
DIV BL ; 7A86H ÷ 04H的商为1EA1H > FFH
```
- 由于BL中的除数04H为字节，被除数为字，商1EA1H大于AL中能存放的最大无符号数FFH，结果将产生除法出错中断。

39

## 符号扩展指令

- 除法指令中，被除数常常需要进行符号扩展或零扩展。

将字节扩展成字	格式	CBW
	功能	将AL的符号位扩展到AH
将字扩展成双字	格式	CWD
	功能	将AX的符号位扩展到DX
将双字扩展成四字	格式	CDQ
	功能	(386以上)EAX符号位扩展到EDX

- 例：设AX=379AH。
  - ✓ 若执行CBW指令，则AX=FF9AH；
  - ✓ 若执行的是CWD指令，则DX=0000H，AX=379AH。
- 注意：80386以上CPU还有MOVSX指令和MOVZX指令。

40

## Example

- 二进制四则混合算术运算程序段
- 试计算：
 
$$AX = (V - (X * Y + Z - 540)) / X \text{ 之商,}$$

$$DX = \text{余数,}$$
 其中，X，Y，Z，V均为字变量、有符号数。

41

## Program

```
; ( V - (X*Y+Z-540) ) / X
MOV AX, X;
IMUL Y;          X*Y,结果在DX:AX中
MOV CX, AX;
MOV BX, DX;      将乘积存在BX:CX中
MOV AX, Z;
CWD;             将符号扩展后的Z加到BX:CX中的乘积上去
ADD CX, AX;
ADC BX, DX;
SUB CX, 540;
SBB BX, 0;       从BX:CX中减去540
MOV AX, V;
CWD;
SUB AX, CX;      从符号扩展后的V中减去(BX:CX)并
SBB DX, BX;      除以X,商在AX中,余数在DX中。
IDIV X;
```

42

## 关于余数

- 可以根据实际应用的需求来处理余数。
  - 四舍五入
  - 截断
- 如果是无符号数除法，采取四舍五入方式时，可将余数与除数的一半进行比较，以决定余数是加入到商，还是舍去。
  - 例，AX除以BL，无符号数，结果四舍五入

```
DIV BL
ADD AH, AH
CMP AH, BL
JB NEXT
INC AL
NEXT:
```

43

## 本章内容

- 加法、减法和比较指令
- 乘法和除法指令
- **BCD码和ASCII码算术运算指令**
- 基本逻辑运算指令
- 移位和循环移位指令
- 串比较指令

44

## BCD算术运算指令

- **BCD算术运算指令不能用于64位模式。**
- **DAA**，加法的十进制修正
  - **Decimal Adjust AL after Addition**
- **DAS**，减法的十进制修正
  - **Decimal Adjust AL after Subtraction**

45

## BCD数

- **BCD数**：二进制编码的十进制数（Binary Coded Decimal）
  - 用4位二进制码表示一位十进制数；
  - **0000~1001**是合法BCD码；**1010~1111**是非法BCD码。
- **压缩BCD数**：用一个字节表示2位BCD数。
  - 例：37
- **非压缩BCD数**：用一个字节的低4位表示一位BCD数，高4位为0。
  - 例：37

0011	0111
------	------

0000	0011
------	------

0000	0111
------	------

46

## 压缩BCD数十进制调整原理(1)

• 例1：18 + 7 = 25

0001	1000	-----	18	
+	0000	0111	-----	7
<hr/>				
0001	1111	-----	?	

(1111是非法BCD码)

- 需要对结果进行变换(调整)，方法：“加6调整”。

0001	1111	
+	0000	0110
<hr/>		
0010	0101	-----

25(正确结果)

- 注意：此时，第3位向第4位(低半字节向高半字节)有进位，AF=1。

47

## 压缩BCD数十进制调整原理(2)

例2：19 + 8 = 27

0001	1001	-----	19	
+	0000	1000	-----	8
<hr/>				
0010	0001	-----	21(结果不对)	

- 运算时，低位数字向高位数字产生了进位(AF=1或CF=1)，实际上是“满16进一”，但进到高位，当成了10，“少6”，需“加6调整”。

0010	0001	
+	0000	0110
<hr/>		
0010	0111	-----

27(结果正确)

- 可见，在BCD数运算时，若AF=1(或CF=1)就需在低4位(或高4位)上进行“加6调整”。

48



### 压缩BCD数十进制调整原理(3)

- 加法器实际上是按二进制运算  
“满16进一”
- 但对于BCD数，应当按10进制算  
“满10进一”，“进1当10”
- 在BCD码结果中，
  - 若某一位BCD数字所对应的二进制码超过9（1010~1111），应“加上6”，产生进位，进行调整。
  - 若低位数字向高位数字产生了进位（AF=1或CF=1），应“加上6”，补上少加的“6”，进行调整。
  - 这可由软件(调整指令)来完成。

49

### 压缩BCD数十进制调整原理(4)

- 压缩BCD数加法十进制调整规则：
  - 如果两个BCD数字相加的结果是一个在1010~1111之间的二进制数，或者有向高一位数字的进位（AF=1或CF=1），则应在现行数字上加6（0110B）调整。
- 压缩BCD数减法十进制调整规则：
  - (1) AF=1，或运算结果的低位是一个在1010~1111之间的二进制数，则在低位上要进行“−6”调整。
  - (2) CF=1，或运算结果的高位是一个在1010~1111之间的二进制数，则在高位上要进行“−6”调整。

50

### 压缩BCD数十进制修正指令(1)

- 加法的十进制修正指令：  
格式：DAA（必须紧跟在ADD、ADC指令后）  
操作：AL ← AL中的和数调整到压缩BCD格式  
修正规律：AL的低4位>9或AF=1，则AL ← AL+06H，AF ← 1  
AL的高4位>9或CF=1，则AL ← AL+60H，CF ← 1  
注意：状态位OF不确定，其余状态位随运算结果而变。

DAA——Decimal Adjust for Addition

标志：O D I T S Z A P C  
U ———× × × × ×

51

### 压缩BCD数十进制修正指令(2)

- 减法的十进制修正指令：  
格式：DAS（必须紧跟在SUB、SBB指令后）  
操作：AL ← AL中的差数调整到压缩BCD格式。  
修正规律：AL的低4位>9或AF=1，则AL ← AL−06H，AF ← 1  
AL的高4位>9或CF=1，则AL ← AL−60H，CF ← 1  
注意事项：状态位OF不确定，其余状态位随运算结果而变。

DAS——Decimal Adjust for Subtraction

标志：O D I T S Z A P C  
U ———× × × × ×

52

### 例：计算BCD3 = BCD1+BCD2

- 例、设BCD1，BCD2，BCD3定义为字变量，可分别存放4位数字的组合BCD数。假定字变量BCD1的值为1834，字变量BCD2的值为2789。要求计算BCD3 = BCD1+BCD2，并指出执行每条指令的操作及执行指令后AL，AF，CF的内容。
- 程序段附后。

BCD1+1	1	8	BCD2+1	2	7		4	6
BCD1	3	4	BCD2	8	9	BCD3	2	3

53

### 例：计算BCD3 = BCD1+BCD2

- 计算：1834+2789=4623

指令	操作	AL	CF	AF
MOV AL, BYTE PTR BCD1	AL ← 34	34	—	—
ADD AL, BYTE PTR BCD2	AL ← 34+89	BDH	0	0
DAA	调整	23 <sub>BCD</sub>	1	1
MOV BYTE PTR BCD3, AL	(BCD3) ← 23	23 <sub>BCD</sub>	1	1
MOV AL, Byte Ptr BCD1+1	AL ← 18	18	1	1
ADC AL, Byte Ptr BCD2+1	AL ← 18+27+CF	40H	0	1
DAA	调整	46 <sub>BCD</sub>	0	1
MOV Byte Ptr BCD3+1, AL	(BCD3+1) ← 46	46 <sub>BCD</sub>	0	1

54

## 压缩BCD数的乘除法

- 没有压缩BCD数的乘法和除法调整指令。
  - 主要原因是相应的调整算法比较复杂，所以不支持压缩BCD数的乘除法运算。
- 如果要处理压缩BCD数的乘除法问题，可以把操作数（压缩BCD数）转换成相等的二进制数，然后用二进制算法进行运算，运算完成后再将结果转换成BCD数。

55

## ASCII算术运算指令

- AAA，加法的ASCII修正
  - ASCII Adjust After Addition
- AAS，减法的ASCII修正
  - ASCII Adjust AL After Subtraction
- AAM，乘法的ASCII修正
  - ASCII Adjust AX After Multiply
- AAD，除法的ASCII修正
  - ASCII Adjust AX Before Division
- 均不能用于64位模式。

56

## ASCII算术运算指令

- AAA，加法的ASCII修正
  - 格式：AAA（必须紧跟在ADD，ADC指令后）
  - 功能：对AL中的非压缩BCD数（或十进制的ASCII码）的加法结果进行修正。
  - 修正规律：
    - AL的低4位>9或AF=1，则 $AL \leftarrow AL + 06H$ ， $AF \leftarrow 1$ ， $AH \leftarrow AH + 1$ ， $AL \leftarrow AL \wedge 0FH$ ， $CF \leftarrow AF$ ，其中 $AH = AH + 1$ 用来实现低位BCD数向高位的进位。
    - AL的低4位<9且AF=0， $AL \leftarrow AL \wedge 0FH$ ， $CF \leftarrow AF$ 。
  - 注意：状态位AF、CF随操作数结果变化；其余状态位都是不确定的。

57

## ASCII算术运算指令

- AAS，减法的ASCII修正
  - 格式：AAS（必须紧跟在SUB，SBB指令后）
  - 功能：对AL中的非压缩BCD数（或十进制的ASCII码）的减法结果进行修正。
  - 修正规律：
    - AL的低4位>9或AF=1，则 $AL \leftarrow AL - 06H$ ， $AF \leftarrow 1$ ， $AH \leftarrow AH - 1$ ， $AL \leftarrow AL \wedge 0FH$ ， $CF \leftarrow AF$ ，其中 $AH = AH - 1$ 用来实现低位BCD数向高位的借位。
    - AL的低4位<9且AF=0， $AL \leftarrow AL \wedge 0FH$ ， $CF \leftarrow AF$ 。
  - 注意：状态位AF、CF随操作数结果变化；其余状态位都是不确定的。

58

## ASCII算术运算指令

- AAM，乘法的ASCII修正
  - 格式：AAM（必须紧跟在MUL指令后）
  - 功能：操作数为累加器AX，对AL中的非压缩BCD数的乘法结果进行修正。
  - 修正规律： $AH = AL / 10$ 的商（高位非压缩BCD数）， $AL = AL / 10$ 的余数（低位非压缩BCD数）。
  - 注意：状态位SF、ZF、PF随操作结果变化；其余状态位都是不确定的。

59

## ASCII算术运算指令

- AAD，除法的ASCII修正
  - 格式：AAD（必须紧跟在DIV指令前）
  - 功能：操作数为累加器AX，AX的内容为两位非压缩的BCD数；在做除法前，对AX中的非压缩BCD数进行修正。
  - 修正规律： $AL = AH \times 10 + AL$ ， $AH = 0$ 。
    - 本质：把BCD码转换成二进制数。
  - 注意：状态位SF、ZF、PF随操作结果变化；其余状态位都是不确定的。

60

### Example1

- 例、求两个非压缩十进制数09和06之乘积，可用下列指令实现。

```
MOV AL, 09H ;置初值
MOV BL, 06H
MUL BL ;计算乘积，得AL=36H
AAM ;调整后得AH=05H（十位），AL=04H（个位）
```

61

### Example2

- 例、求BCD数72除以9。

```
MOV BL, 09H
MOV AX, 702H ;置初值
AAD ;调整，AX=0048H
DIV BL ;计算乘积，得AL=09H
```

62

### 本章内容

- 加法、减法和比较指令
- 乘法和除法指令
- BCD码和ASCII码算术运算指令
- 基本逻辑运算指令
- 移位和循环移位指令
- 串比较指令

63

### 逻辑运算指令

名称	格式	操作
与	AND DST, SRC	$DST \leftarrow DST \wedge SRC$
或	OR DST, SRC	$DST \leftarrow DST \vee SRC$
非	NOT DST	$DST \leftarrow \overline{DST}$
异或	XOR DST, SRC	$DST \leftarrow DST \oplus SRC$
测试	TEST DST, SRC	$DST \wedge SRC$

- DST可以是reg, mem;
- SRC可以是reg, mem, 或imm。
- 注意：AND, OR, NOT, XOR, TEST是按位进行运算。

64

### 逻辑运算指令

- AND DST, SRC, OR DST, DST和XOR DST, SRC
  - DST和SRC不能同时为mem。
  - 状态位SF、ZF和PF随运算结果而变化， $CF \leftarrow 0$ ， $OF \leftarrow 0$ ，而AF不确定。
- NOT DST
  - 注意：NOT不影响标志位。
- TEST DST, SRC
  - 执行 $DST \wedge SRC$ 操作后，两个操作数内容不变。
  - DST和SRC不能同时为mem。
  - 状态SF、ZF和PF随运算结果而变化， $CF \leftarrow 0$ ， $OF \leftarrow 0$ ，而AF不确定。

65

### Example

- 设AX=3538H
  - AND AX, 0F0FH; AX←0508H
- 设AX=0508H
  - OR AX, 3030H; AX←3538H
- NOT BYTE PTR [BX]; 对存储单元的内容按位取反
- TEST BX, 3; 合法
  - TEST 3, BX; MASM5.0中不合法
  - MASM6.15中，编译器会把TEST 3, BL转换成TEST BL, 3

66

## 基本逻辑运算指令

- 80386以上CPU新增加了一些测试单一的位测试指令。

位测试	格式	BT REG/MEM, REG/IMM8
	功能	测试目的操作数中的某一位，测试结果放入CF标志
位测试并取反	格式	BTC REG/MEM, REG/IMM8
	功能	测试目的操作数中的某一位，测试结果放入CF标志，并将测试位取反
位测试并清零	格式	BTR REG/MEM, REG/IMM8
	功能	测试目的操作数中的某一位，测试结果放入CF标志，并将测试位清零
位测试并置位	格式	BTS REG/MEM, REG/IMM8
	功能	测试目的操作数中的某一位，测试结果放入CF标志，并将测试位置位

67

## Example

- BT AX, 4; 如果第4位为1，则CF=1，否则CF=0
- BTC AX, 4
- BTR AX, 4
- BTS AX, 4

68

## 基本逻辑运算指令

求补（变负）指令	格式	NEG REG/MEM
	功能	把操作数当成一个带符号数， <ul style="list-style-type: none"> <li>如果原操作数是正数，NEG指令则将其变成负数（用补码表示）；</li> <li>如果原操作数是负数（用补码表示），NEG指令则将其变成正数。</li> </ul>

- 方法：“各位（包括符号位）求反，末位加1”。
- 对比：由原码求补码？由补码求原码？

69

## Example

- 若AL=00010001B=+17，
  - 执行NEG AL后，AL=11101111B=[-17]<sub>补</sub>
- 若AL=11010001B=[-47]<sub>补</sub>，
  - 执行NEG AL后，AL=00101111B=+47

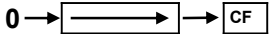
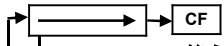
70

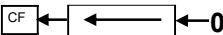
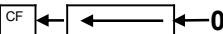
## 本章内容

- 加法、减法和比较指令
- 乘法和除法指令
- BCD码和ASCII码算术运算指令
- 基本逻辑运算指令
- 移位和循环移位指令
- 串比较指令

71

## 移位指令

- 逻辑右移 SHR REG/MEM, CL/IMM8  

- 算术右移 SAR REG/MEM, CL/IMM8  


✓算术右移时，符号位保持不变。
- 逻辑左移 SHL REG/MEM, CL/IMM8  

- 算术左移 SAL REG/MEM, CL/IMM8  


✓ SHL和SAL功能一样。

72

## 移位指令

1. 8086CPU中，IMM8只能是1。
2. 8086CPU不对移位次数取模。32位CPU对移位次数取32的模。对于64位目的操作数，取64的模。
3. 状态位SF、ZF和PF随运算结果变化。AF不确定。
4. 状态位CF：
  - 最后一次移入到CF中的值。
  - 对于SHL和SHR，当移位次数超过目的操作数长度时，CF值不确定。
5. 状态位OF：
  - 左移1位时，结果的最高位(即符号位)与CF一致，则OF=0，否则为1。SAR右移，OF=0；SHR右移，OF=源操作数的最高有效位。
  - 左移/右移多位时，OF值不确定。

73

## 移位指令

- 逻辑移位：把操作数作为无符号数进行移位。  
右移时，最高位补0；  
左移时，最低位补0。
- 算术移位：把操作数作为有符号数进行移位。  
右移时，最高位保持不变；  
左移时，最低位补0。

74

## Example

- 例，AX的内容乘以5。

```
MOV BX, AX
SHL AX, 2
ADD AX, BX
```

75

## 双精度移位指令

- 80386以上的CPU包含2条双精度移位指令：
  - SHLD（左移），SHRD（右移）

双精度 左移	格式	SHLD REG/MEM, REG, CL/IMM8
	功能	第一操作数向左移，高位依次移入CF，其“空出”的低位由第二操作数的高位来填补，但第二操作数自己不移动、不改变。
	标志	CF、OF、PF、SF和ZF受影响，AF无定义。
双精度 右移	格式	SHRD REG/MEM, REG, CL/IMM8
	功能	第一操作数向右移，低位依次移入CF，其“空出”的高位由第二操作数的低位来填补，但第二操作数自己也不移动、不改变。
	标志	CF、OF、PF、SF和ZF受影响，AF无定义。

76

## 循环移位指令

- 不带进位的循环左移：ROL REG/MEM, CL/IMM8
- 不带进位的循环右移：ROR REG/MEM, CL/IMM8
- 带进位的循环左移：RCL REG/MEM, CL/IMM8
- 带进位的循环右移：RCR REG/MEM, CL/IMM8

77

## 循环移位指令

1. 8086CPU中，IMM8只能是1。
2. 8086CPU不对移位次数取模。32位CPU对移位次数取32的模。对于64位目的操作数，移位次数取64的模。
3. 状态位SF、ZF、AF不受影响。
4. 状态位CF：最后一次移入到CF中的值。
5. 状态位OF：
  - 左移1位时，结果的最高位(即符号位)与CF一致，则OF=0，否则为1。
  - 右移1位时，结果的最高2位的异或值。
  - 左移/右移多位时，OF值不确定。

78

## Example

- 例，将DX、BX和AX中的48位数据向左移移位。

```
SHL AX, 1
RCL BX, 1
RCL DX, 1
```

79

## 位扫描指令

- 80386+包含了两条位扫描指令：
  - BSF: Bit scan forward, 向前位扫描指令
  - BSR: Bit scan reverse, 向后位扫描指令

向前位扫描	格式	BSF REG, REG/MEM
功能		在源操作数中搜索值为1的最低位。如果找到，则将位索引存储到目标操作数。位索引是从0算起的无符号偏移量。如果源操作数的内容为0，则目标操作数的内容未定义。
标志		如果源操作数的所有位都是0，则ZF=1；否则ZF=0。CF、OF、SF、AF及PF标志未定义。

80

## 位扫描指令

向后位扫描	格式	BSR REG, REG/MEM
功能		在源操作数中搜索值为1的最高位。如果找到，则将位索引存储到目标操作数。位索引是从0算起的无符号偏移量。如果源操作数的内容为0，则目标操作数的内容未定义。
标志		如果源操作数的所有位都是0，则ZF=1；否则ZF=0。CF、OF、SF、AF及PF标志未定义。

- BSF和BSR的用途：为位操作指令寻址值为1的位。
- 例，设EAX=60000000H。
  - 如果执行BSF EBX, EAX，则EBX=29，ZF=0。
  - 如果执行BSR EBX, EAX，则EBX=30，ZF=0。
  - 注意：教材的中英文版不一致。英文版写错了。

81

## 本章内容

- 加法、减法和比较指令
- 乘法和除法指令
- BCD码和ASCII码算术运算指令
- 基本逻辑运算指令
- 移位和循环移位指令
- 串比较指令

82

## 串比较指令

- SCAS: String Scan, 串扫描
- CMPS: String Compare, 串比较

83

## SCAS指令

串扫描	格式	SCAS MEM 或 SCASB / SCASW / SCASD / SCASQ
功能		比较ES:DI或ES:EDI或RDI指定的字节、字或双字与AL、AX或EAX或RAX中的值，并根据结果设置状态标志。比较之后，根据DF标志，DI、EDI或RDI自动递增或递减1、2、4或8。ES段不能使用跨段前缀覆盖。
标志		OF、SF、ZF、AF PF及CF标志根据比较的临时结果设置。

- 通过REP前缀，SCAS、SCASB、SCASW及SCASD指令可用于整块比较CX个字节、字或双字或四字。

84

## Example

- 例，假定从BLOCK开始的存储区域长为100字节，要求测试该存储区域，查看哪个单元有00H。

```
MOV DI, OFFSET BLOCK
CLD
MOV CX, 100
XOR AL, AL
REPNE SCASB
```

85

## CMPS指令

串比较	格式	CMPS MEM, MEM 或 CMPSB / CMPSW / CMPSD / CMPSQ
	功能	比较DS:SI与ES:DI（或DS:ESI与ES:EDI，RSI与RDI）指定的字节、字、双字或四字的值，并根据结果设置状态标志。比较之后，根据DF标志，DI、EDI或RDI自动递增或递减1、2、4或8。ES段不能使用跨段前缀覆盖。
	标志	OF、SF、ZF、AF PF及CF标志根据比较的临时结果设置。

- 在CMPS、CMPSB、CMPSW、CMPSD及CMPSQ前面增加REP前缀，可整块比较CX个字节、字或双字或四字。

86

## Example

- 例，假定LINE和TABLE分别指向两段存储区域，要求检查它们的内容是否相同。

```
MOV SI, OFFSET LINE
MOV DI, OFFSET TABLE
CLD
MOV CX, 10
REPE CMPSB
```

87

## REP前缀

- REP前缀指令
  - 常用格式：REP MOVSB/LODSB/STOSB
  - 若CX≠0，重复执行，每执行一次CX=CX-1
  - 若CX=0，则退出重复，结束串操作。
- REPE/REPZ前缀指令
  - 常用格式：REPE/REPZ CMPSB/SCASB
  - CX≠0且ZF=1，重复执行，每执行一次CX=CX-1
  - CX=0或ZF=0，则停止重复执行。
- REPNE/REPNZ前缀指令
  - 常用格式：REPNE/REPNZ CMPSB/SCASB
  - CX≠0且ZF=0重复执行，每执行一次CX=CX-1
  - CX=0或ZF=1，则停止重复执行。

88

## REP前缀

- 前缀指令本身不影响状态位。
- 关于CX寄存器：
  - 80386以上微处理器使用ECX。
  - Pentium 4在64位模式下，使用RCX。
- REP MOVSB指令：先检查CX是否等于0，然后再执行MOVSB。
  - 若CX=0，则一次都不执行MOVSB，也不会执行CX=CX-1的操作。

89

## 本章小结

- 加法、减法和比较指令
- 乘法和除法指令
- BCD码和ASCII码算术运算指令
- 基本逻辑运算指令
- 移位和循环移位指令
- 串比较指令
- 熟记指令的格式和功能。

90

## 作业（1）

- 习题5、习题13、习题19、系统37、习题55。
- （补充题1）指出下列指令中哪些是错误的，错在什么地方？
  - （1）ADD AL, AX
  - （2）ADD 8650H, AX
  - （3）ADD DS, 0200H
  - （4）ADD [BX], [1200H]
  - （5）ADD IP, 0FFH
  - （6）ADD [BX+SI+3], IP
  - （8）INC [BX]

91

## 作业（2）

- （补充题2）写一个短指令序列，要求计算BL和CL中的数据的数据的平方和；在计算开始前，将5和6分别装入BL和CL寄存器；结果存放在DL寄存器中。
- （补充题3）设计短指令序列，将AL中奇数位的值均为1，偶数位的值均为0，并将AH中的位取反。

92