**第三章、指令系统及寻址方式**

## 3.1 8086寻址方式

### 3.1.1指令格式

### 3.1.2寻址方式

立即寻址

寄存器寻址

存储器寻址

直接寻址

基址寻址

变址寻址

基址--变址寻址

## 3.2 8086指令系统

### 3.2.1数据传输指令

MOV指令

MOV指令使用规则

XCHG 指令

堆栈指令

PUSH指令

POP指令

地址传送指令LEA

符号位扩展指令 CBW CWD

### 3.2.2算术运算指令

加法/减法指令

ADD/SUB

ADC/SBB

INC/DEC

求补指令NEG（Negate）

比较指令CMP

乘法/除法指令

十进制调整指令

### 3.2.3逻辑运算和移位类指令

逻辑运算类指令

AND

OR

NOT

XOR

TEST

移位类指令

非循环（开环）移位指令

循环移位类指令

### 3.2.4串指令

串操作指令共同点

字符串传送指令

### 3.2.5控制转移指令

转移指令的概念

无条件转移指令

条件转移指令

比较和测试指令

单标志位条件转移指令

无符号数比较转移指令

有符号数比较转移指令

循环指令

## 3.1 8086寻址方式

### 3.1.1指令格式

指令中应包含的信息：

* 运算数据的来源（源操作数）
* 运算结果的去向（目的操作数）
* 执行的操作  
  操作码 [目的操作数]，[源操作数] ；注释

### 3.1.2寻址方式

立即寻址：操作数直接出现在指令中 MOV AX , 3EC5H

寄存器寻址：操作数在寄存器中 MOV BX , AX

存储器寻址：操作数在内存 MOV AX , [SI]

速度：存储器寻址 > 寄存器寻址 > 立即寻址

#### 立即寻址

操作数（常数）直接由指令给出，其作为指令的一部分存放在代码段中。  
【注意】

* 立即寻址只能用于源操作数
* 立即数和目的操作数的长度要一致

MOV CX, 100 ;正确,CX赋值为100

MOV 2A00H , AX ;错误,目的操作数不能是立即数

MOV BL , 8000H ;错误,BL长度为1个字节,8000H长度为2个字节

#### 寄存器寻址

操作数存放在CPU的内部寄存器中,寄存器名表示其内容（操作数）。  
【注意】

* 源操作数与目的操作数字长要相同
* 寄存器寻址与段地址无关

MOV CL , AL ;正确,将AL中的值传送给CL

MOV AX , BL ;错误,AX为2个字节,BL为1个字节

MOV CX , ES:AX ;错误,源操作数不能是段地址

#### 存储器寻址

存储器操作数的表现形式（中括号）：[立即数或寄存器]  
[ ]中的内容是存放所寻找数据的单元的**偏移地址**  
【注意】

* 在默认的、或用段超越前缀指定的段寄存器中
* 指令中只需给出操作数的逻辑地址（有效地址EA）

MOV [SI] , AX ;默认在DS段,DS:[SI] ← AX,把AX中的值赋给DS段,偏移地址为SI的存储单元

MOV ES:[SI] , AX ;使用段超越前缀,ES:[SI] ← AX,把AX中的值赋给ES段,偏移地址为SI的存储单元

4种存储器寻址方式

##### 直接寻址

指令中直接给出操作数的偏移地址

* 默认在数据段（DS）
* 允许段超越（重设）

MOV AX , [1200H] ;将数据段DS中,偏移地址为1200H的数据赋给AX

MOV AX , ES:[1200H] ;将附加段ES中,偏移地址为1200H的数据赋给AX

##### 基址寻址

* 有效地址存放在基址寄存器BX或BP中
* 或有效地址 EA = [BX/BP] + [8位/16位 偏移量]  
  中括号里面只能是BX或BP
* 对应BX寄存器默认是DS ，对应BP寄存器默认是SS， 可使用段超越前缀改变
* 主要用于一维数组

MOV AX , [BX] ;AX ← DS:[BX]

MOV AX , [BX+06H] ;AX ← DS:[BX+06H]

MOV AX , 06H[BX] ;AX ← DS:[BX+06H]两种写法一样

##### 变址寻址

* 有效地址存放在变址寄存器SI或DI中
* 或有效地址 EA = [SI/DI] + [8位/16位 偏移量]  
  中括号里面只能是SI或DI
* 默认段是DS ，可使用段超越前缀改变

MOV AX , [DI] ;AX ← DS:[DI]

MOV AX , [SI+06H] ;AX ← DS:[SI+06H]

MOV AX , 06H[SI] ;AX ← DS:[SI+06H]两种写法一样

##### 基址--变址寻址

* 主要用于二维数组操作
* 有效地址 EA = [BX/BP] + [SI/DI]
* 或有效地址EA = [BX/BP] + [SI/DI] + [8位/16位 偏移量]
* BX对应的段地址寄存器默认是DS，BP的段地址默认是SS；可用段超越前缀改变
* 注意：同一组内的寄存器不能同时出现，即不能出现[BX][BP]或者[SI][DI]

MOV AX , [BX+SI] ;AX ← DS:[BX+SI]

MOV AX , [BX][SI] ;AX ← DS:[BX+SI]两种写法一样

MOV AX , [BX+DI+6] ;AX ← DS:[BX+DI+6]

MOV AX , 6[BX+DI] ;AX ← DS:[BX+DI+6]

MOV AX , 6[BX][DI] ;AX ← DS:[BX+DI+6]三种写法一

例：假定DS=8000H, BX=2000H, SI=10H, (82010H)=22H, (82011H)=11H

执行MOV AX，[BX][SI]

结果AX=？

解答：BX+SI = 2010H,DS:[BX+SI] = 80000+2010H = 82010H

​ 即从偏移地址2010H处取2个字节赋值给AX，则AL = (82010H) = 22H，AH = (82011H) = 11H

​ 低地址给低字节，高地址给高字节，AX = 1122H

## 3.2 8086指令系统

### 3.2.1数据传输指令

* 通用数据传送指令 MOV、 XCHG
* 堆栈指令 PUSH 、 POP
* 地址传送指令 LEA 、LDS
* 输入输出指令 IN 、 OUT

#### MOV指令

* 把一个字节或字的操作数从源地址传送至目的地址
* 指令格式

MOV dest , src ;dest ← src

* 可实现8位或16位数据的传送
* 不影响状态位
* 把数据从一个位置传送到另一个位置

##### MOV指令使用规则

1. 不允许段寄存器之间的传送
2. 不允许立即数到段寄存器的传送
3. 不允许内存到内存的传送
4. 不允许数据长度类型不同的传送（但位数前可补0）
5. 传送时数据类型需明确

MOV CL , DL ;8位(1字节)传送

MOV AX , [BX] ;2字节传送

MOV [SI] , CL ;1字节传送

MOV byte ptr [BX+5] , 0A2H ;1字节传送

;立即数传给存储器，需要用ptr指明存储格式(byte还是word)

MOV word ptr [BX+5] , 0A2H ;2字节位传送

MOV ES , DS ;错误,(附加)段寄存器不能到(数据)段寄存器

MOV DS , 1000H ;错误,立即数不能到段寄存器

MOV [BX] , [1000H] ;错误,内存操作数(存储器)不能到内存(存储器)

MOV AL , 050AH ;错误,长度不匹配

#### XCHG 指令

指令格式

XCHG dest , src ;两操作数内容交换

遵循指令的通用规则

* 两操作数中必须有一个是寄存器
* 操作数不能是段寄存器或立即数

XCHG AX , BX ;正确

XCHG [2000H] , CL ;正确

XCHG ES , AX ;错误,操作数不能是段寄存器

XCHG AX , 1234H ;错误,操作数不能是立即数

#### 堆栈指令

指令格式

PUSH OPRD ;压栈

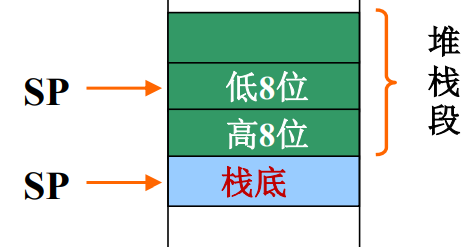
POP OPRD ;弹栈

* 是“后进先出”（或“先进后出”） 的操作，位于堆栈段中
* SS段寄存器记录堆栈段地址
* 以字为单位（每次操作影响一个字的长度，即两字节）
* 不能对立即数做堆栈操作

##### PUSH指令

PUSH指令执行过程

1. SP - 2 → SP
2. 操作数高字节 → SP+1
3. 操作数低字节 → SP  
   先减指针、再压数据、先高后低

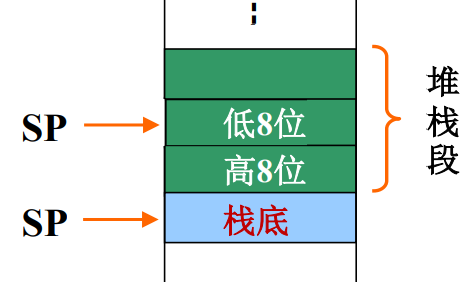


栈顶指针SP一开始位于栈底，压栈时，先将栈顶指针上移（减小）2个字节，再将操作数赋值给栈顶指针对应的位置

##### POP指令

POP指令执行过程

1. SP → 操作数低字节
2. SP+1→操作数高字节
3. SP+2→ SP  
   先出数据、再加指针、先低后高



弹栈时，先将栈顶指针SP指向的数据取出来（每次操作取2个字节，低地址对应低字节，高地址对应高字节），再将栈顶指针下移（增加）2个字节

PUSH AX ;正确

PUSH WORD PTR [1000H] ;正确

PUSH 1234H ;错误,操作数不能是立即数

PUSH AL ;错误,操作数要2个字节

POP BX ;正确

POP AL ;错误

堆栈段的作用

* 临时存放数据
* 传递参数
* 保存和恢复寄存器

#### 地址传送指令LEA

将存储器操作数的16位偏移地址送到指定的寄存器  
指令格式

LEA reg , mem

* 源操作数必须是一个存储器操作数，目的操作数 须是一个16位的通用寄存器（最好是间址寄存器）
* 常用于初始化程序，使一个寄存器成为指针

LEA BX , [SI+10H] ;若SI=8000H，则执行该指令后，BX=8010H

**LEA指令和MOV指令的区别**

;假定BUF指向地址0100H处,且地址0100H处存放数据1234H

LEA BX , BUF ;BX ← 0100H,LEA指令取BUF的地址给BX

MOV BX , BUF ;BX ← 1234H,直接把BUF的值给BX

MOV BX , OFFSET BUF ;BX ← 0100H,OFFSET先取BUF的地址作为一个立即数赋值给BX

#### 符号位扩展指令 CBW CWD

x86asm

* 这两条指令用于将8（16）位有符号数扩展为16（32） 位数据。
* 符号扩展虽然使数据位数加长，但数据大小并没有改变，扩展的高部分仅是低部分的符号扩展。
* 常用于有符号数的除法运算中
* 隐含寻址，寄存器AX（AL）  
  指令格式

CBW ;把AL的符号位复制到AH

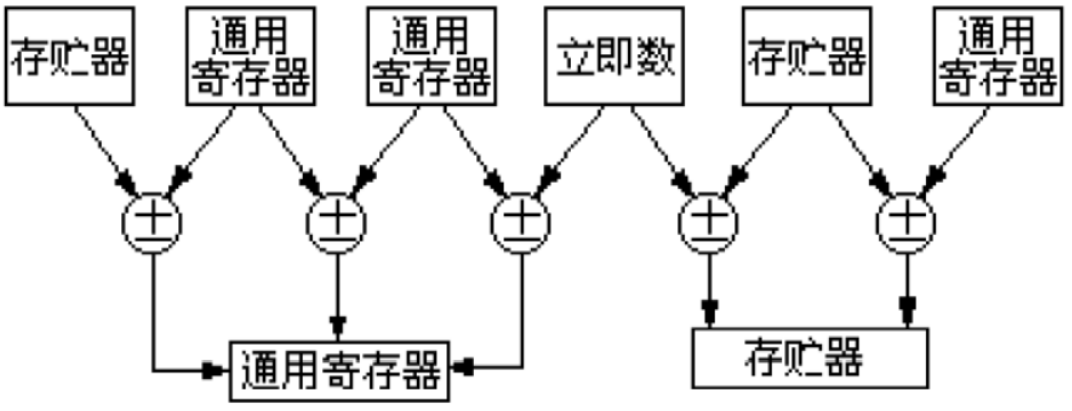
CWD ;把AX的符号位复制到DX

MOV AL , 91H ;AL = 1001 0001符号位为1

CBW ;AX = 1111 1111 1001 0001,把AL的符号位填入AH全部位置

### 3.2.2算术运算指令

* 执行二进制的算术运算：加减乘除
* 可用于字或字节的运算
* 操作数不能是段寄存器
* 关注对状态标志的影响



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **分类** | **名称** | **格式** | **功能** | **O S Z A P C** |
| 加法指令 | 加法指令 | ADD DST , SRC | 加法（字、字节） | O S Z A P C |
| 带进位加法指令 | ADC DST , SRC | 带进位加法（字、字节） | O S Z A P C |
| 加1指令 | INC OPRD | 加1（字、字节） | O S Z A P |
| 减法指令 | 减法指令 | SUB DST , SRC | 减法（字、字节） | O S Z A P C |
| 带借位减法指令 | SBB DST , SRC | 带借位减法（字、字节） | O S Z A P C |
| 减1指令 | DEC OPRD | 减1（字、字节） | O S Z A P |
| 比较指令 | CMP DST , SRC | 比较（字、字节） | O S Z A P C |
| 求补指令 | NEG OPRD | 求补码（字、字节） | O S Z A P C |
| 乘法指令 | 无符号数乘法 | MUL SRC | 不带符号数乘法（字、字节） | O C |
| 带符号数乘法 | IMUL SRC | 带符号数乘法（字、字节） | O C |
| 除法指令 | 无符号数除法 | DIV SRC | 不带符号数除法（字、字节） | 没有定义 |
| 带符号数除法 | IDIV SRC | 带符号数除法（字、字节） | 没有定义 |

#### 加法/减法指令

##### ADD/SUB

* 不带进位/借位的加法/减法
* 影响6个状态标志位
* 指令格式

ADD OPRD1 , OPRD2 ;OPRD1 ← OPRD1 + OPRD2

SUB OPRD1 , OPRD2 ;OPRD1 ← OPRD1 - OPRD2

##### ADC/SBB

* 带进位/借位的加法/减法
* 用于长字节数的加法/减法，在ADD/SUB指令后使用
* 指令格式

ADC OPRD1 , OPRD2 ;OPRD1 ← OPRD1 + OPRD2 + CF

SBB OPRD1 , OPRD2 ;OPRD1 ← OPRD1 - OPRD2 – CF

##### INC/DEC

* 二进制加1/减1指令
* 不能是段寄存器或立即数
* 常用于在程序中修改地址指针
* 不影响CF标志
* 指令格式

INC reg/mem ;reg/mem ← reg/mem + 1

DEC reg/mem ;reg/mem ← reg/mem – 1

##### 求补指令NEG（Negate）

* 求补运算，即用零减去操作数，结果返回操作数
* 用于求一个数的（有符号）相反数
* 对标志的影响与SUB指令一样
* 对非零数求补时，CF=1
* 指令格式

NEG reg/mem ;reg/mem ← 0-reg/mem

##### 比较指令CMP

* 将两个操作数相减，但结果不送目的操作数
* 影响全部6个标志状态，可作为条件转移指令转移的条件
* 两数相等ZF=1
* 指令格式

CMP OPRD1 , OPRD2

#### 乘法/除法指令

* 指令中的乘数、除数必须是寄存器或存储器操作数
* 乘法指令中不能出现 AL，AX寄存器
* 除法指令中不能出现AX，DX
* 运算前应先送被乘数/被除数到累加器AX（AL）/AX(DX:AX)
* 指令格式

MUL OPRD ;AX ← AL \* OPRD或DX:AX← AX \* OPRD

DIV OPRD ;AX(16位)/OPRD(8位) = AL(商)……AH(余数)

;DX:AX(32位)/OPRD(16位) = AX(商)……DX(余数)

#### 十进制调整指令

* 压缩BCD码加法调整指令DAA（Decimal Adjustment after Addition）、减法调整指令 DAS
* 非压缩BCD码加法调整指令AAA 、减法调整指令 AAS
* 调整指令在算数运算指令后使用
* 加/减法调整指令默认操作对象是AL

;AL=28H,BL=68H

ADD AL , BL ;AL=90H,AF=1

DAA ;AL=96H

;若低位>9或AF=1，则AL加6H，且AF=1

;若高位>9或CF=1，则AL加60H，且CF=1

### 3.2.3逻辑运算和移位类指令

#### 逻辑运算类指令

##### AND

* 段寄存器不能参加逻辑运算
* 用途：保留操作数的某几位，其他位清零
* 指令格式

AND AL , 0FH ;留AL中低4位，高4位清0

AND AL , 11011111B ;将AL中的字符 ’a’～’z’转换成大写

##### OR

* 用途：把操作数的某几位置1，其他位不变
* 指令格式

OR AL , 00100000B ;把AL的 bit5 置1

OR AL , 30H ;把AL中的非压缩BCD码转换成ASCII码

##### NOT

* 按位取反再送回原地址

##### XOR

* 指令格式

;BL=01110101B

XOR BL , 80H ;把操作数的某几位变反（与1异或）

;BL ← 0111 0101 XOR 1000 0000 = 1111 0101

XOR AX , AX ;把寄存器清零（与自身异或）AX = 0

##### TEST

* 执行“与”运算，但结果不送回目标地址
* 常用于测试某些位的状态，用‘1’测试，用ZF标志位判断结果。后面跟转移指令。

;测试AL的内容是否为负数

TEST AL , 80H ;检查AL的最高位是否为1,若为1,ZF=1,否则ZF=0

#### 移位类指令

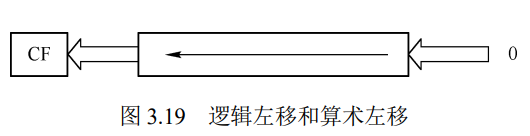
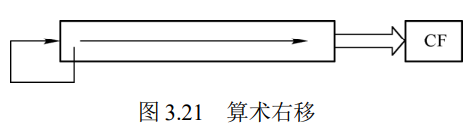
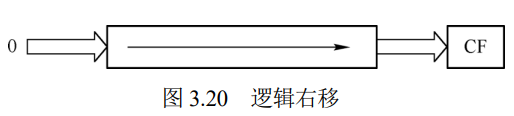
【注】

* 移动1位时由指令直接给出
* 移动两位及以上，则移位次数由CL指定

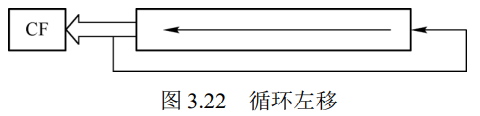
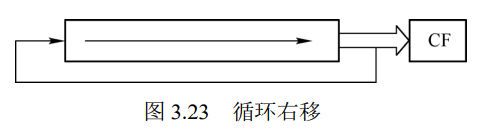
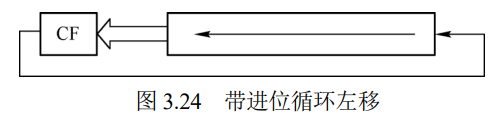
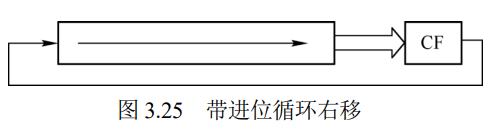
SAL reg/mem , 1 ;移位位数=1时

SAL reg/mem , CL ;移位位数>1时

##### 非循环（开环）移位指令

* 算术左移指令 SAL(Shift Arithmetic Left)和逻辑左移指令 SHL(Shift Left)  
  
* 算术右移指令 SAR(Shift Arithmetic Right)  
  
* 逻辑右移指令 SHR(Shift Right)  
  
* 左移一位可实现×2运算，右移一位可实现÷2运算
* 无符号数的乘除用逻辑移位，有符号数用算数移位

##### 循环移位类指令

* 不含进位位的循环左移指令 ROL(Rotate Left)  
  
* 不含进位位的循环右移指令 ROR(Rotate Right)  
  
* 含进位位的循环左移指令 RCL(Rotate Left with Carry)  
  
* 含进位位的循环右移指令 RCR(Rotate Right with Carry)  
  
* 可实现多字节移位

;将DX:AX中32位数值左移一位

SHL AX , 1

RCL DX , 1

### 3.2.4串指令

* 针对内存中一个连续区域（数据块或字符串）的操作
* 可实现存储器到存储器的数据传送
* 8086指令系统提供了5条基本的串操作指令和3类重复前缀
  + 传送数据串：MOVS，STOS，LODS
  + 检测数据串：CMPS，SCAS
  + 重复前缀：REP，REPZ，REPNZ

#### 串操作指令共同点

1. 寻址方式均为隐含寻址
   * 源操作数 DS:[SI]
   * 目的操作数 ES:[DI]
2. CX作为串计数器
3. 指针SI、DI自动修正，方向取决 于DF
   * DF=0，增量
   * DF=1，减量
4. DF控制指令
   * CLD指令 ,DF=0
   * STD指令 , DF=1

#### 字符串传送指令

* 指令格式

REP MOVSB ;字节操作

REP MOVSW ;字操作

REP MOVSD ;双字操作

* 传送操作：实现了内存到内存的传送

|  |  |  |  |
| --- | --- | --- | --- |
|  | **字节操作** | **字操作** | **双字操作** |
| DF=0为增址型 | SI ← SI + 1 | SI ← SI + 2 | SI ← SI + 4 |
| DI ← DI + 1 | DI ← DI + 2 | DI ← DI + 4 |
| DF=1为减址型 | SI ← SI - 1 | SI ← SI - 2 | SI ← SI - 4 |
| DI ← DI - 1 | DI ← DI - 2 | DI ← DI - 4 |

执行 REP MOVSB之前，需要

* DS:SI, ES:DI 赋值
* D标志置0/置1
* CX 赋值  
  若CX=0，则退出串处理循环， 执行下一条指令  
  若CX≠ 0，则执行基本串操作， CX ← CX - 1  
  先对CX进行判断，再执行串操作

例：试编程将0000:1000H开始的100个字节传送到 2000:0000H开始的单元中去

MOV AX , 0000H

MOV DS , AX ;DS赋值

MOV AX , 2000H

MOV ES , AX ;ES赋值

MOV SI , 1000H

MOV DI , 0000H

MOV CX , 64H ;CX赋值

CLD ;CLD指令,DF置0,增址型

REP MOVSB

### 3.2.5控制转移指令

* 控制转移指令按应用分
  + 转移指令
  + 循环控制
  + 过程调用
  + 中断控制
* 按转移条件分：无条件转移和有条件转移
* 按转移范围分：段内转移和段间转移
* 按获取转移地址的方法分：直接转移和间接转移
* 控制转移指令不影响标志位

#### 转移指令的概念

* 通过改变CS和指令指针IP，改变指令执行的顺序。
* 根据程序转移地址的范围不同，分为：
  + 段内转移：只改变IP内容
  + 段间转移：改变IP和CS的内容
* 条件转移指令只能是段内转移
* 控制转移指令根据获取转移地址的方法不同，分为：
  + 直接转移：指令中通过立即数（标号）给出转移地址
  + 间接转移：指令中通过寄存器或存储器给出转移地址

|  | **段内** | **段间** |
| --- | --- | --- |
| 直接转移 | 通过立即数（标号）直接给出16位地址来修改IP | 通过立即数直接给出32位地址来修改CS和IP |
| 间接转移 | 通过寄存器操作数或内存操作数给出16位地址来修改IP | 通过内存操作数给出32位地址来修改CS和IP |

#### 无条件转移指令

指令格式

JMP OPRD

#### 条件转移指令

* 在满足一定条件下，程序转移到目标地址，可实现分支结构
* 条件转移指令均为段内短转移
* 30个操作码助记符隐含了转移条件

##### 比较和测试指令

| **指令操作码** | **指令格式** | **功能描述** |
| --- | --- | --- |
| CMP | CMP dest , src | (dest) - (src)，不存结果 |
| TEST | TEST dest , src | (dest) ∧∧ (src)，不存结果 |

如果只是想比较两个数，而不想改变它的值，选用CMP  
如果只是想判断个别二进制位，而不想改变它的值，选用TEST

##### 单标志位条件转移指令

判断5个标志位的值

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **转移指令** | | | | | |
| 单位标志 | JZ | JS | JO | JP | JC |
| JNZ | JNS | JNO | JNP | JNC |

##### 无符号数比较转移指令

在cmp指令后使用

CMP N1 , N2

无符号数条件转移指令

| **指令格式** | **转移条件** |  |
| --- | --- | --- |
| JA XYZ | N1 > N2 转 | A（Above，高于） |
| JAE XYZ | N1 ≥ N2 转 |  |
| JB XYZ | N1 < N2 转 | B（Below，低于） |
| JBE XYZ | N1 ≤ N2 转 |  |
| JE XYZ | N1 = N2 转 | E (Equal，等于) |

##### 有符号数比较转移指令

在cmp指令后使用

CMP N1 , N2

有符号数条件转移指令

| **指令格式** | **转移条件** |  |
| --- | --- | --- |
| JG XYZ | N1 > N2 转 | G (Greater，大于) |
| JGE XYZ | N1 ≥ N2 转 |  |
| JL XYZ | N1 < N2 转 | L (Less，小于) |
| JLE XYZ | N1 ≤ N2 转 |  |
| JE XYZ | N1 = N2 转 | E (Equal，等于) |

例：将AX，BX中的大数放在AX和变量WMAX中

CMP AX , BX

JAE NEXT

XCHG AX , BX

NEXT: MOV WMAX , AX

#### 循环指令

* 指令格式

L1: ················ ;循环体

;L1与：间不能有空格

;参数修正

LOOP L1

* 循环次数由CX指定
* 首先CX←CX－1
* 若CX≠0，转到目标地址
* 循环条件：CX ≠ 0

例：将数据段中定义的100字 的数组ARY逆序传送到另 一个数组DEST

LEA SI , ARY

LEA DI , DEST

ADD DI , 198

MOV CX , 100

L1: MOV AX , [SI]

MOV [DI] , AX

ADD SI , 2

SUB DI , 2

LOOP L1