**第二章 8088/8086微处理器的指令系统**

**2.1 8088/8086的寻址方式**

指令是微处理器执行某种操作的命令，微处理器全部指令的集合称为指令系统。

指令的兼容性：同一系列机的指令都是兼容的。

一条指令应包含的信息：

1.运算数据的来源

2.运算结果的去向

3.执行的操作

指令有两种书写格式：机器指令和符号指令。

符号指令是用规定的助记符和规定的书写格式书写的指令。符号指令的书写格式为：

操作码助记符 操作数助记符

操作码 【操作数】（目标操作数，一定是一个地址），【操作数】（源操作数）

MOV AL, 1

10110000 00000001

**1) 操作码**

指明CPU要执行什么样的操作。

是一条指令必不可少的部分，用助记符表示。

数据传送

算术运算

逻辑运算

串操作

控制转移

处理机控制

**2) 操作数**

指明参与操作的数据或数据所在的地方。

了解操作数的来源、个数、类型。

1. 立即操作数：立即数本身是参加操作的数据，8/16位，不能作为目标操作数
2. 寄存及操作数：参加运算的数存放在指令给出的寄存器中，运行所用时间最短
3. 存储器操作数参加运算的数存放在存储器的某一个或某两个单元中，表现形式【单元的偏移地址】，最慢的

操作数个数

按指令格式中，操作数个数的多少分为四类：

无操作数: 指令只有一个操作码，没有操作数

单操作数: 指令中给出一个操作数

双操作数: 指令中给出两个操作数。

三操作数：指令中给出三个操作数。

**寻址方式**

寻址方式：寻找指令中操作数地址的方式。

操作数有三种可能的存放方式：

直接包含在指令中

立即数

立即数寻址

包含在某个寄存器中

寄存器操作数

寄存器寻址

在内存中

存储器操作数（内存操作数）

存储器寻址

段内偏移地址可以由如下四个部分组成（称为偏移地址四元素）：

基址寄存器内容

变址寄存器内容

比例因子（ Pentium处理器才有,8086为1）

位移量

由四元素组合形成的偏移地址称为有效地址EA:

EA=基址+(变址比例因子)+位移量

EA8088/8086=基址+变址+位移量

**1. 立即寻址**

操作数作为立即数直接存在指令中，紧跟在操作码后，放在代码段。立即寻址方式只能出现在源操作数的寻址中，目的地操作数不能采用此种方式。（赋初值）

MOV AX, 5678H

**2. 寄存器寻址**

这种寻址方式的操作数在CPU的内部寄存器中。

MOV AX, BX

**3. 直接寻址**

指令中的操作数部分直接给出操作数的有效地址EA【】，操作数是16位整数，操作数默认在DS段中。

MOV AX, [3000H]

如果操作数在DS以外的其他段（CS,SS,ES）中，指令中必须指明段寄存器，这称段跨越。

MOV AX, ES:[3000H]

**4. 寄存器间接寻址**

操作数地址的有效地址EA存放在寄存器中

由寄存器间接给出操作数的偏移地址

16位偏移地址放在SI,DI,BP,BX中（【】中只能是这四个间址寄存器）

以SI,DI, BX间接寻址，默认操作数在DS段中，数据段

MOV AX, [SI]

以BP间接寻址，默认操作数在SS段中，堆栈段

MOV AX, [BP]

**5. 基址寻址**

EA=[基址寄存器]+位移量

方括号表示寄存器中的内容是偏移地址。

BP,BX为基址寄存器

BX， DS数据段为默认段寄存器

BP， SS堆栈段为默认段寄存器

1. **变址寻址**

EA=[变址寄存器]+位移量

SI,DI为变址寄存器, DS为默认段寄存器

变址寻址适用于对一维数组的元素进行操作

**7. 基址加变址寻址**

EA=[基址寄存器]+[变址寄存器]+偏移量

通常把BX和 BP作为基址寄存器，把SI和DI作为变址寄存器，一共有四种组合。

BX DS为默认段寄存器

BP SS为默认段寄存器

MOV AX, COUNT[BX+SI]

MOV AX, [BP+SI]

**2.2 8088/8086的指令系统**

1.数据传送（Data transfer）指令 (14)

2.算术运算（Arithmetic）指令 (3+5+2+2+2+6=20)

3.逻辑运算（Logic）指令 (4+8=12)

4.串操作（String manipulation）指令 (5\*2=10)

5.控制转移（Control transfer）指令 (1+8+10+2+2+2=23)

6.处理器控制（Processor control）指令

以下几个方面来掌握一条指令:

指令的助记符

指令的格式：操作数的个数、类型（B，W，DW）

执行的操作：指令执行后的结果

包括: 哪些寄存器、内存单元的值发生了变化

对标志位有无影响，哪些受影响

注意三： 书写指令注意事项：

不区分字母的大小写。

下列写法表示同一条指令：

MOV AX， 1ABDH

mov ax, 1abdh

不添加指令系统没有的指令，即不自创助记符。

将 MOV AL , 0 写成 MOVE AL，0

JMP lable 写成 JUMP lable

注意四：注意操作数的范围

对字节操作指令 0 ~ FFH 0 ~ 255

对字操作指令 0 ~ FFFFH 0 ~ 65535

八位有符号的补码 80H~~7FH -128~~+127

十六位有符号数 8000H~~7FFFH -32768~~+32767

MOV AL ， 260

MOV AX ，70000

MOV AL, 1FFH

MOV AL, 2ABCDH

**注意五：注意操作数的格式**

对无操作数指令，不添加操作数。

STC AL

对单操作数指令，操作数不能是立即数。

IMUL 6

对双操作数指令

① 不能两个同为存储器操作数

MOV [ DI ] , [ SI ]

② 目的操作数不能是立即数

ADD 3 , AL

③ 两个操作数的类型应相同

SUB AX , BL

若 value 定义为字类型存储器变量：

MOV CL, value [ BX ]

一致性原则：跟随原则不明确的操作数，跟随类型明确的而确定。

1. B、C、D、E、F开头的十六进制数前面加0，与H结尾的标识符区别。

如 寄存器名： AH、BH、CH、 DH

变量名 ： abcdH 等

例 mov AL, 0AH

mov AL, AH

mov BX, 0abcdH

**注意六：个别寄存器的特殊性**

CS 和 IP的值只在控制转移指令中修改。

对非控制转移指令，取完指令后IP值自动 指向下条指令。

段寄存器CS的值，只在MOV、PUSH中可作操作数，

且这两条指令执行结果不改变CS值。

MOV AX, CS

PUSH CS

IP、PSW两个寄存器不作为操作数在指令中出现。

mov IP , 1234H

mov PSW , 0F0FH

PSW状态寄存器的值由指令执行后确定，

不同的指令对各标志的影响不同。

**一、数据传送指令**

数据传送是最基本、最重要的一种操作

实际程序中，使用的比例最高

1. **寄存器 寄存器**
2. **寄存器 内存单元**
3. **寄存器 I/O端口**
4. **设置寄存器、内存单元的初始值**

执行后不影响标志位，源操作数不变，有四类：

通用传送指令 地址传送指令

累加器专用传送指令 标志传送指令

**（一）通用数据传送指令（该类所有指令不影响标志位）**

包括MOV PUSH POP XCHG

MOV 目标操作数（OPRD1)， 源操作数(OPRD2)

**两个操作数字长必须相同**

**两操作数不允许同时为存储器操作数**

**两个操作数不允许同时为段寄存器**

**在源操作数是立即数时，目标操作数不能是段寄存器**

1. 源操作数可以是8/16位的立即数、寄存器操作数、内存操作数。目标操作数不允许为立即数，其余同源操作数。
2. 源、目不能同时为内存操作数。

源、目操作数类型必须匹配，属性明确

MOV BYTE PTR [BX], 12H

3) 不能向段寄存器写立即数

MOV DS,2000 (错误） MOV AX, 2000

MOV DS, AX

4) 以CS和IP为目的操作数的一切传送指令都是非法的

PUSH 源操作数 （操作数必须为16位，不能是立即数）

**堆栈操作指令**

**16位操作数，不能是立即数**

堆栈是按照先进后出原则组织的一段内存区，存在于堆栈段中，SP在任何时候都指向栈顶。

通常用于存放一些重要数据,

如：程序的地址、或是需要恢复的数据。

为方便数据的存放和恢复，

设置专门的指针，指向堆栈中要操作的单元。

段值由 SS 给出，偏移地址由 SP 给出

SS → 堆栈段寄存器 (stack segment)

SP → 堆栈指针寄存器 (stack point)

**PUSH 源操作数**

进栈指令，先调整堆栈指针，再把源操作数压栈

（Ⅰ）SP SP–1

（Ⅱ）OPRDH (SP)

（Ⅲ）SP SP –1

（Ⅳ）OPRDL (SP)

PUSH AX

PUSH WORD PTR [SI+5]

**POP 目标操作数**

出栈指令，先将栈顶2字节送目标操作数，再调整堆栈指针

（Ⅰ）(SP) OPRDL

（Ⅱ） SP SP+1

（Ⅲ） (SP) OPRDH

（Ⅳ） SP SP+1

POP AX

POP WORD PTR [SI+5]

注意：堆栈操作只能对十六位的操作数进行操作，且为寄存器或存储器操作数。如：PUSH AL 是错误的。

**交换指令**

**XCHG 目标操作数, 源操作数**

将源操作数与目的操作数的内容互换

XCHG AL, BL

XCHG [2500H], DX

1）可以是字节交换也可以是字交换

2）可以是寄存器与寄存器之间进行交换

3）可以是寄存器与存储器之间进行交换

4）不可以是存储器与存储器之间交换

5）CS和IP不能用来进行交换

1. **累加器专用传送指令**

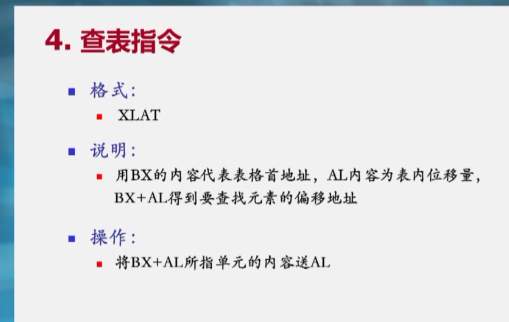
**1、查表转换指令**

•XLAT 转换表名（即转换表首地址）

从转换表中查找出一个字节的内容，用其取代AL寄存器中的内容。

XLAT TABLE

转换表最长为256个字节，是由用户设计的。执行查表指令前，BX应指向转换表的起点。AL的内容被用作查表时索引，即被查找数在表中的位置。



**2. 输入/输出指令**

完成**累加器**和**I/O端口**之间的数据传送

**IN** 累加器，端口号

**OUT** 端口号,累加器

端口号为8位时，直接寻址,最多可访问256个端口。

IN AL, PORT（端口地址）

OUT PORT,AL/AX

端口地址为16位时，间接寻址，端口地址必须放在DX寄存器中，最多可访问65536个端口。

IN AL, DX

OUT DX, AL

**三） 地址传送指令**

**•LEA 目标寄存器， 源操作数**

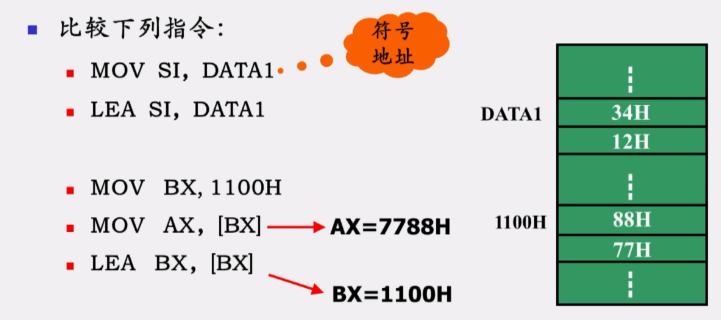
有效地址传送指令，源操作数为内存操作数，将内存单元的有效地址（而不是内容）传送到目标寄存器，即将目的操作数的偏移地址送寄存器。

LEA R, SRC ; SRC代表源操作数，R代表间址寄存器

例：将TABLE的偏移地址送SI

LEA SI, TABLE

与MOV SI, OFFSET TABLE等效。



**LDS/LES 目标寄存器， 源操作数**

指针传送指令，将一个存放在4个存储单元中共计32位的目标指针（段地址和偏移量）传送到两个目的寄存器。

LDS DI, [2130H]

把2132H，2133H中的内容送DS，把2130H，2131H的内容送DI。

LES与LDS基本相同，区别只是把目的段地址送ES寄存器。

**（四）标志传送指令**

LAHF （PSW低8位——》AH）

把标志寄存器中SF、ZF、AF、PF、CF五个标志位传到AH的第7、6、4、2、0，第5、3、1没定义。

SAHF

作用与LAHF相反，将AH中的内容送至标志寄存器中。

PUSHF

把标志寄存器压栈。

POPF

将标志寄存器退栈。

**二、 算术运算指令**

加、减、乘、除，运算对象8/16位有符号/无符号整数，以及BCD码，影响标志位。

**1. 加法指令**

ADD 目标操作数， 源操作数

源操作数+目标操作数——>目标操作数

ADC 目标操作数， 源操作数

源操作数+目标操作数+CF——>目标操作数

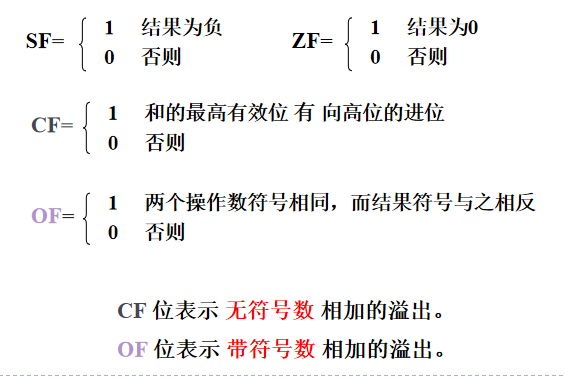
影响A,C,O,P,S,Z 6个标志位

INC 目标操作数

目标操作数+1——>目标操作数

影响A,O,P,S,Z 5个标志位

加法指令对条件标志位（CF/OF/ZF/SF）的影响：



1. **减法指令**

SUB 目标操作数， 源操作数

目标操作数-源操作数─>目标操作数

SBB 目标操作数， 源操作数

目标操作数-源操作数-CF─>目标操作数

DEC 目标操作数

目标操作数-1─>目标操作数

NEG 目标操作数

0-目标操作数─>目标操作数，即对给出的字节

或字操作数求补

影响A,C,O,P,S,Z6个标志位

CMP 目标操作数， 源操作数

比较指令，执行两数相减操作，但不送回相减结果

只是影响标志位。

（1）两数为无符号数的比较

如果CF为0则表示无借位，被减数大于减数；如果

CF为1，则表示有借位，被减数小于减数。

（2）两数为有符号数的比较

如果OF和SF相等，则表示被减数大于减数；如果

OF和SF不相等（相异），则表示被减数小于减数。

**3. 乘法指令**

MUL 乘数

无符号乘法

IMUL 乘数

有符号乘法

（1）字节乘，被乘数放在AL中，乘积结果的低8位放在AL中，高8位放在AH中。

（2）字乘，被乘数放在AX中，乘积结果的低16位放在AX中，高16位放在DX中。

**4. 除法指令**

DIV 除数

无符号除法

IDIV 除数

有符号除法

注意：除数必须为被除数的一半字长

（1）字节除，被除数放在AX, 则商放在AL中，余数放在AH中。

（2）字除，被除数放在DX AX中, 则商放在AX中，余数放在DX中。

**5. BCD码运算指令**

十进制调整指令

BCD码：用二进制编码的十进制数，又称二--十进制数

组合的BCD码：用 4 位二进制数表示 1 位十进制数

例：(59)10 ＝(0101 1001)BCD

非组合的BCD码：用 8 位二进制数表示 1 位十进制数

例：(59)10 ＝(0000 0101 0000 1001)BCD

组合的BCD码运算调整指令 DAA

两个组合的BCD码相加，结果在AL中，执行该指令后将结果调整为十进制，放在AL中。

未组合的BCD码运算调整指令 AAA

个未组合的BCD码相加，结果在AL中，执行该指令后将结果调整为十进制，放在AX中。

**三 逻辑运算指令**

**位操作**

AND 目标操作数，源操作数

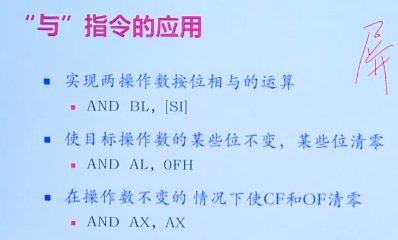
OR 目标操作数，源操作数

XOR 目标操作数，源操作数

NOT 目标操作数

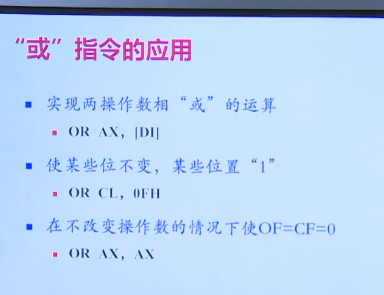
**AND** 按位相与，主要用于将二进制数的某些位

清0。



**OR** 按位 相或，主要用于将二进制数的某些位

置1。



**XOR** 按位相异或，主要用于将二进制数的某些

位求反。

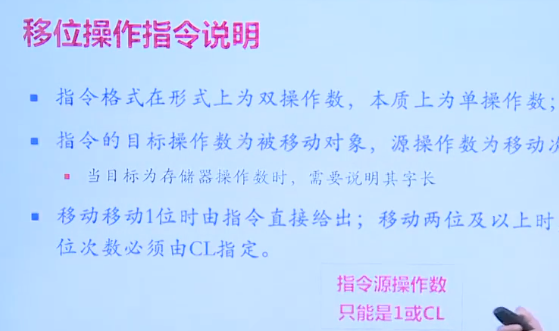
**TEST**与AND一样将两个操作数按位相与，但

结果不回送，只影响标志位。

当被测试为为0时，ZF置位（ZF=1）;

当被测试为为1时，ZF复位（ZF=0）

**移位操作**



逻辑移位：无符号数 （SHL,SHR)

算术移位：有符号数 (SAL,SAR)

指令格式：

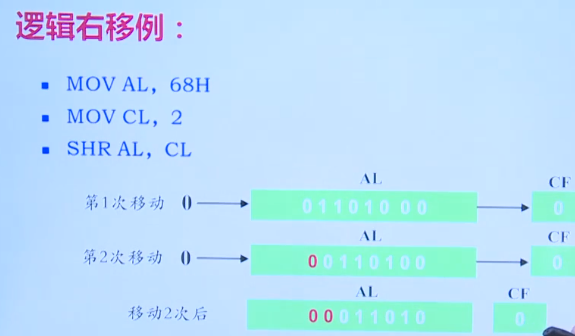
SHL 目标操作数，计数

移1位时，计数值可以为立即数1

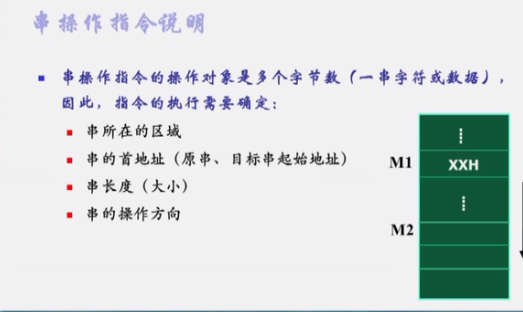
移多位时，计数值必须先存入CL寄存器

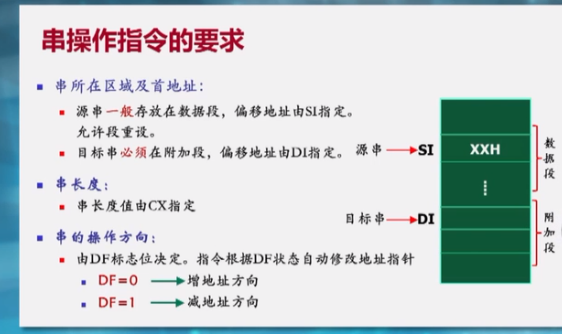
左移，最高位移到CF，最低位补0





1. **串操作指令**

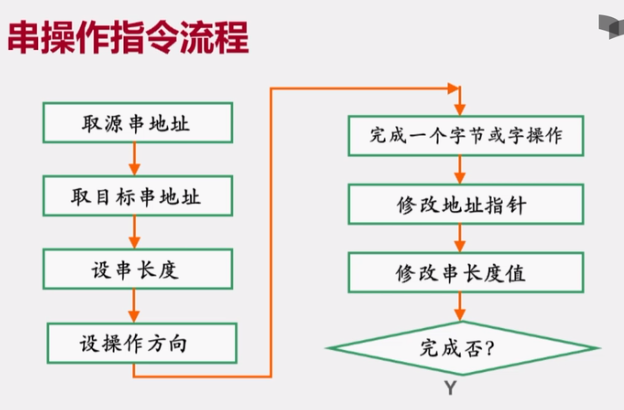


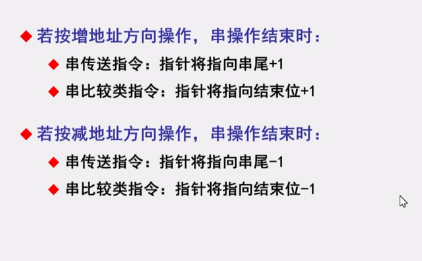


串的基本概念

顺序存放在内存中的一组数据，称为串。

用串的首（末）地址、元素类型、串的长度表示。



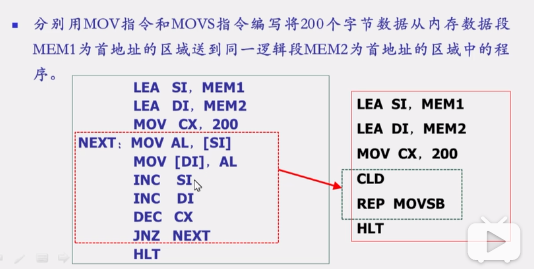


**串传送**

1. **将一个字节/字从**
2. **DS:SI ES:DI**

MOVSB

MOVSW



完成操作后自动修改SI、DI，使其指向串的下一个元素

串操作方向由CLD和STD指令设置

CLD 地址递增方向（DF=0）

STD 地址递减方向（DF=1）

需要先将串的长度存入CX寄存器

每处理完一个元素自动使CX-1,直到CX=0才结束串传送——完成整个串的传送

**串比较**

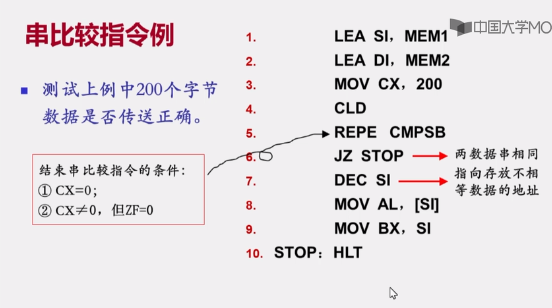
CMPSB CMPSW

**会影响标志位**

比较地址为DS:SI、ES:DI的两个字节/字，同时修改SI和DI指向下一个元素。

指令前通常加重复前缀REPZ/REPE, REPNZ/REPNE ，从而当发现两个串不同(或相同）时结束比较,这样就可以找到两个串中第一个不相等元素或第一个相等的元素。





目标串搜索指令

SCASB

SCASW

在首地址为ES:DI的串中搜索某个元素（字节/字），同时修改DI指向下一个元素。

事先要将待搜索的元素存入AL/AX

指令前通常加重复前缀REPNZ/REPNE，从而当发现待搜索的元素时结束搜索。

**串搜索**

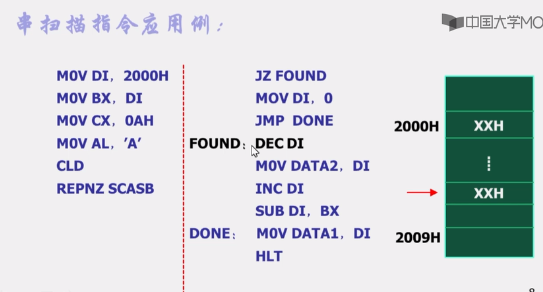
MOV DI, OFFSET String

MOV CX, xx

MOV AL, ‘h’

REPNZ SCASB

JNZ Not\_Found



**串装载**

LODSB

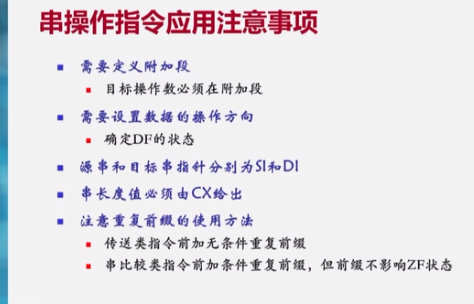
LODSW

将地址为DS:SI的一个字节/字装入AL/AX，同时修改SI指向下一个元素。

**串存储**

将AL/AX的值存入地址为ES:DI的内存单元，并修改DI

指向下一个元素。



**五、 程序控制指令**

**控制程序的流向：**

无条件转移

条件转移

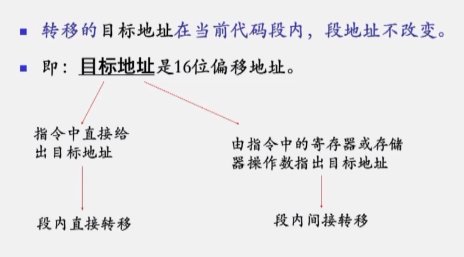
循环控制

过程调用与返回

中断指令

**无条件转移**

(1)段内直接转移 JMP 2000H ; IP=2000H



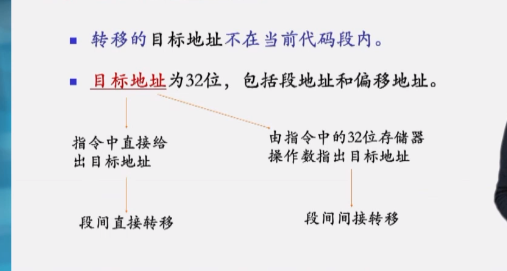
1. 段内间接转移 JMP AX ; IP=(AX)

(3)段间直接转移 JMP 2500H:0100H

;CS=2500H,IP=0100H

(4)段间间接转移 JMP DWORD PTR[SI]

;IP和CS的内容用内存中2个连续的字来替代。



**条件转移**

根据执行上一指令后标志寄存器的状态而决定是否转移

**1、判断无符号数大小的条件转移**

A/JNBE ; > ,CF v ZF=0

JAE/JNB ; >=,CF=0

JB/JNAE ;< ,CF=1

JBE/JNA ;<=,CF v ZF=1

**2、判断有符号数大小的条件转移**

JG/JNLE ;> ,ZF=0且SF xor OF=0

JGE/JNL ;>= ,SF xor OF=0

JL/JNGE ;< ,SF xor OF=1

JLE/JNG ;<= ,ZF=1或Sf xor OF=1

**3、单标志位条件转移**

JZ/JE ;ZF=1

JNZ/JNE ;ZF=0

JC ;CF=1

JNC ;CF=0

JO ;OF=1

JNO ;OF=0

**循环控制**

循环控制指令所控制的目的地址都用标号表示，

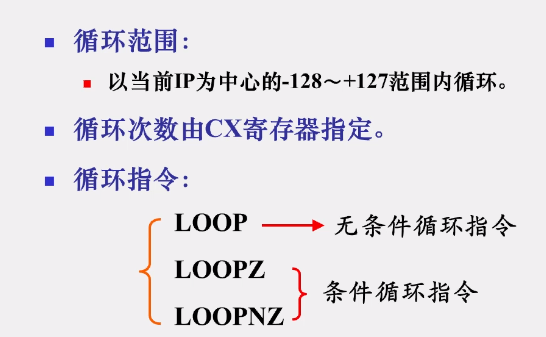
该标号都在距当前IP：-128~+127范围，且与CX配合使用，CX存放循环次数。

结束条件：

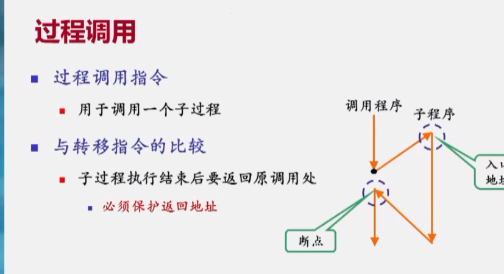
LOOP ；CX=0

LOOPE/LOOPZ ；CX=0或者ZF=0

LOOPNE/LOOPNZ ; CX=0或者ZF=1



**子程序调用和返回指令**



段内直接调用：CALL 1000H ;IP=1000H

CALL LABLE；IP=LABLE地址

段内间接调用：CALL AX ;IP=(AX)

段间直接调用：CALL 2500:1000H

;IP=1000H ，CS=2500H

段间间接调用：CALL DWORD PTR [SI]

;IP在SI指向的单元中，CS 在IP+2指向的单元中

返回指令： RET ;弹出IP和CS，返回主程序

**中断指令INT n，INTO和IRET**

执行INT n时，会引起CPU转入一个中断服务程序。其过程为：

（Ⅰ）首先标志位入堆栈

（Ⅱ）清除中断允许标志IF和单步标志TF

（Ⅲ）保护断点，即断点地址入栈

（Ⅳ）将n×4得到的中断向量送IP和CS

INTO 为溢出中断指令，当OF=1是，INTO的中断处理程序会给出出错标志。

INTO=INT 4

IRET 中断返回指令，和中断指令配套使用，用以退出中断过程，返回到主程序。

**处理器控制指令**

**1、标志操作指令**

CLC ;清CF=0

CMC ;使CF取反

STC ;置CF=1

CLD ;清DF=0

STD ;置DF=1

CLI ;清IF=0

STI ;置IF=1

**2、处理器控制指令**

.处理器暂停指令HLT

以下三种情况可使8088脱离暂停状态:

(1)在RESET上有复位信号

(2)在NMI线上有非屏蔽中断请求

(3)中断允许（IF=1）时，INTR上有请求

**3.处理器交权指令ESC**

用于向外部处理机提供了从8088获得操作码和存储器操作数的手段，用于多处理机中，即8088的最大工作模式。

.等待指令WAIT

使CPU进入等待状态，直至TEST线上的信号有效为止，用于与外部接口电路的同步。

**5.总线封锁指令LOCK**

可以放在任何一条指令前，是一个指令前缀。用于最大工作模式。在执行紧跟在LOCK前缀之后的那条指令期间，发出总线封锁信号，并使该信号保持到该指令执行完。LOCK信号有效期间其它处理机不得占用总线。

**6.空操作指令NOP**

NOP使CPU不做任何工作，它不影响标志，主要用于时序配合。