计算机组成与结构
—指令系统3

计算机科学与技术学院

温故 —关于指令系统

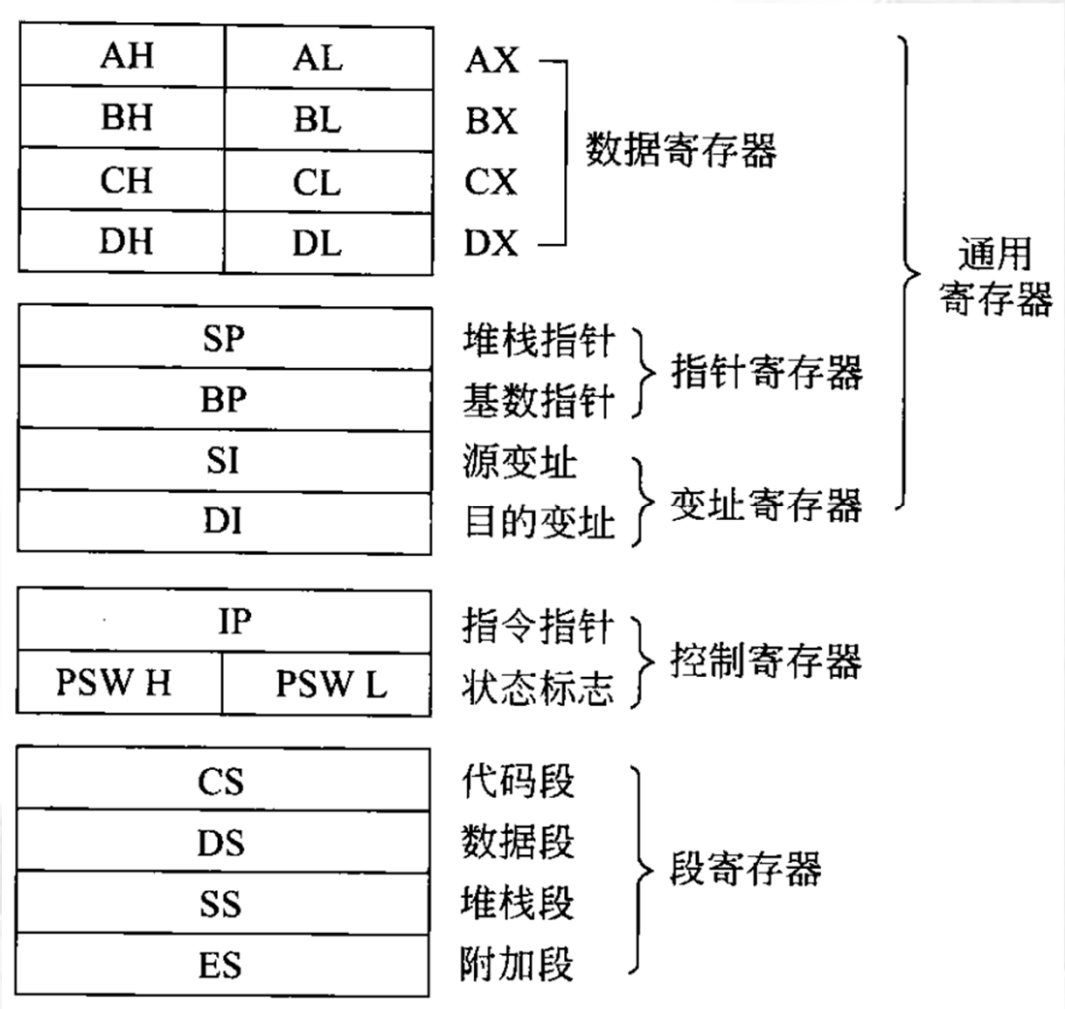
- Huffman方式指令编码的优劣势?
  - 优点:
  - 让高频指令的opcode更短，平均长度较短，减少冗余量
  - 缺点:
    - 长度不规整，译码困难
    - 与地址码组成定长指令困难
- 扩展式指令编码原则?
  - 短码不能是长码的前缀

温故 —关于指令系统

- 15-15-15 如何规划字段的?
  - 0000-1110为4位短码，1111（未用）作为前缀
  - 1111 + 4位短码 = 8位码， 1111 1111（未用）
  - 1111 + 1111 + 4位短码 = 12位码
- 8-64-512如何规划字段的?
  - 0XXX为4位短码，1XXX（未用）作为前缀
  - 1XXX + 0XXX = 8位码， 1XXX 1XXX （未用）
  - 1XXX + 1XXX + 0XXX = 12位码

温故 —关于指令系统

- **简述8086内寄存器分几类，各有什么功能?**
  - **数据寄存器**
  - **指针寄存器**
  - **变址寄存器**
  - **控制寄存器**
  - **段寄存器**



寻址方式

运算，
需要数据，
需要取数据，
需要知道用哪种方式去取数据，
还需要知道从哪里取数据，
—— 寻址方式
Addressing Mode！

但在此之前，
必须仔细学习一个汇编指令…

- **MOV指令**

格式：　　**MOV　OPRD1，　　OPRD2**

　　目的　　源

功能：　　**（OPRD2）→（OPRD1）**

其中：　目的操作**OPRD1**　　源操作数　**OPRD2**

存储器	**mem**	立即数	**im**
通用寄存器	**r**	存储器	**mem**
段寄存器	**SEG（除CS/IP）**	通用寄存器	**r**
		段寄存器	**SEG（含CS）**

但也有4种例外情况：

- 目的操作数不能为立即数，或 CS、IP；
- 内存单元之间不能进行数据直接传送；
- 立即数不能直接传送到段寄存器中；
- 段寄存器之间的不能进行数据直接传送；

Done ！

寻址方式Addressing Mode

- Immediate 立即
- Direct 直接
- Register 寄存器
- Register + Indirect 寄存器间接 (2种)
- Register + Indexing 寄存器相对
- Base + Displacement 基址变址 (2种)
- Base + Displacement + Indexing 基址+变址+相对
- Implicit 隐式

寻址方式 — 立即寻址

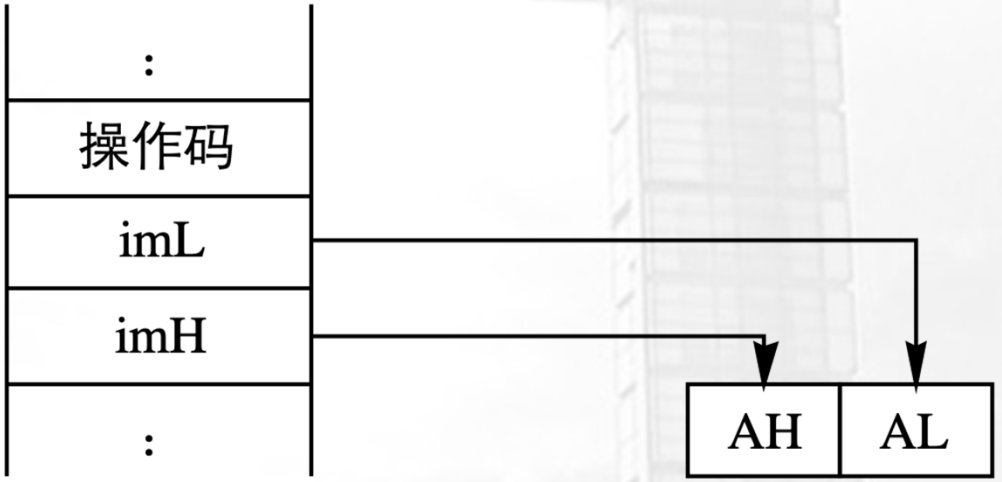
- **操作数直接包含在指令中！ (跟在操作码后面)**
- **操作数是在代码段！！**

例如:

MOV AL, 27H

ADD AL, 35H

MOV AX, 2000H



注意！！ 地址低位去AL，高位去AH !!!!

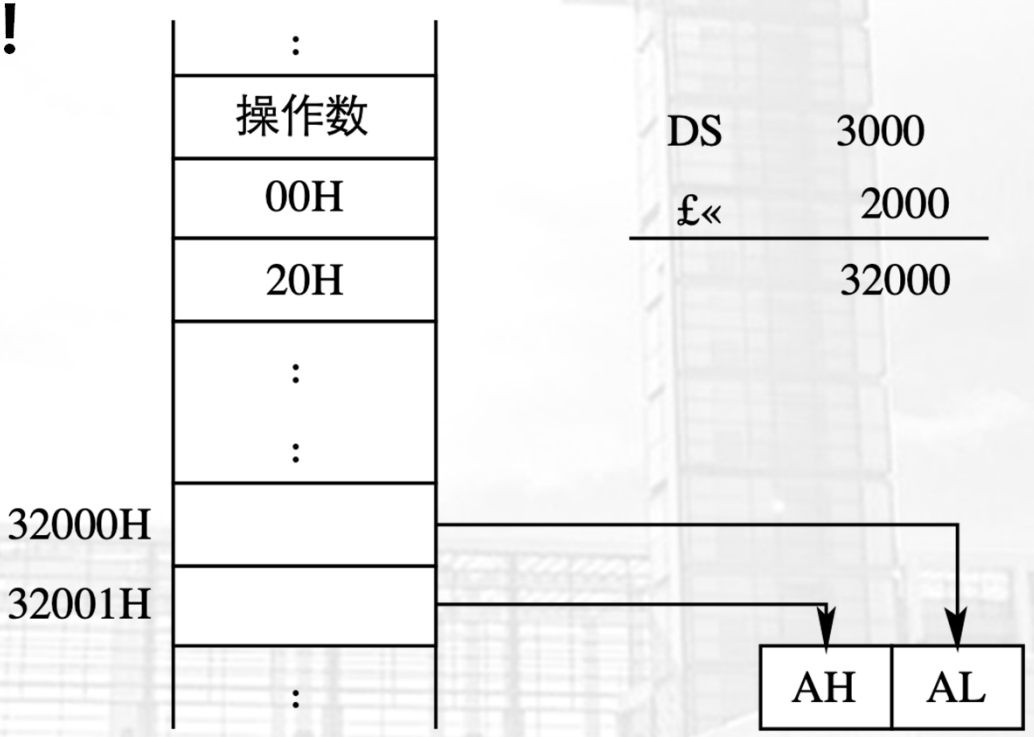
寻址方式 — 直接寻址

- **操作数的段内偏移地址直接包含在指令中！（跟在操作码后面）**
- **操作数是在数据段！！**

例如：

MOV AX, [2000H]

设DS=3000H



寻址方式 — 寄存器寻址

- **操作数在某个寄存器中！**

例如:

MOV DS, AX

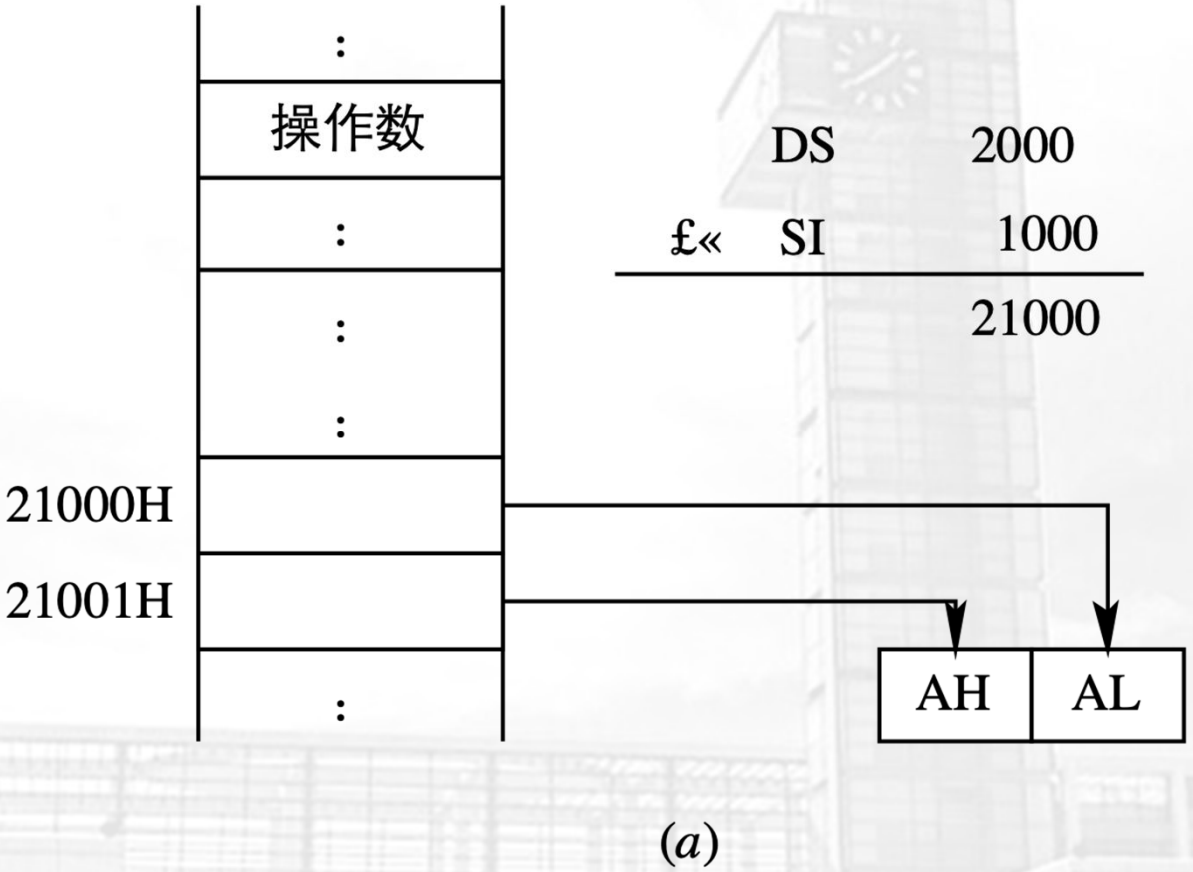
寻址方式 — 寄存器间接寻址

- 操作数**的段内偏移地址**放在SI/DI/BP/BX其中一个寄存器中!
- 分两种情况:
  - 如果是SI/DI/BX　<—搭配DS在数据段找操作数
  - 如果是BP　<—搭配SS在堆栈段找操作数

寻址方式 — 寄存器间接寻址

- **例如：**

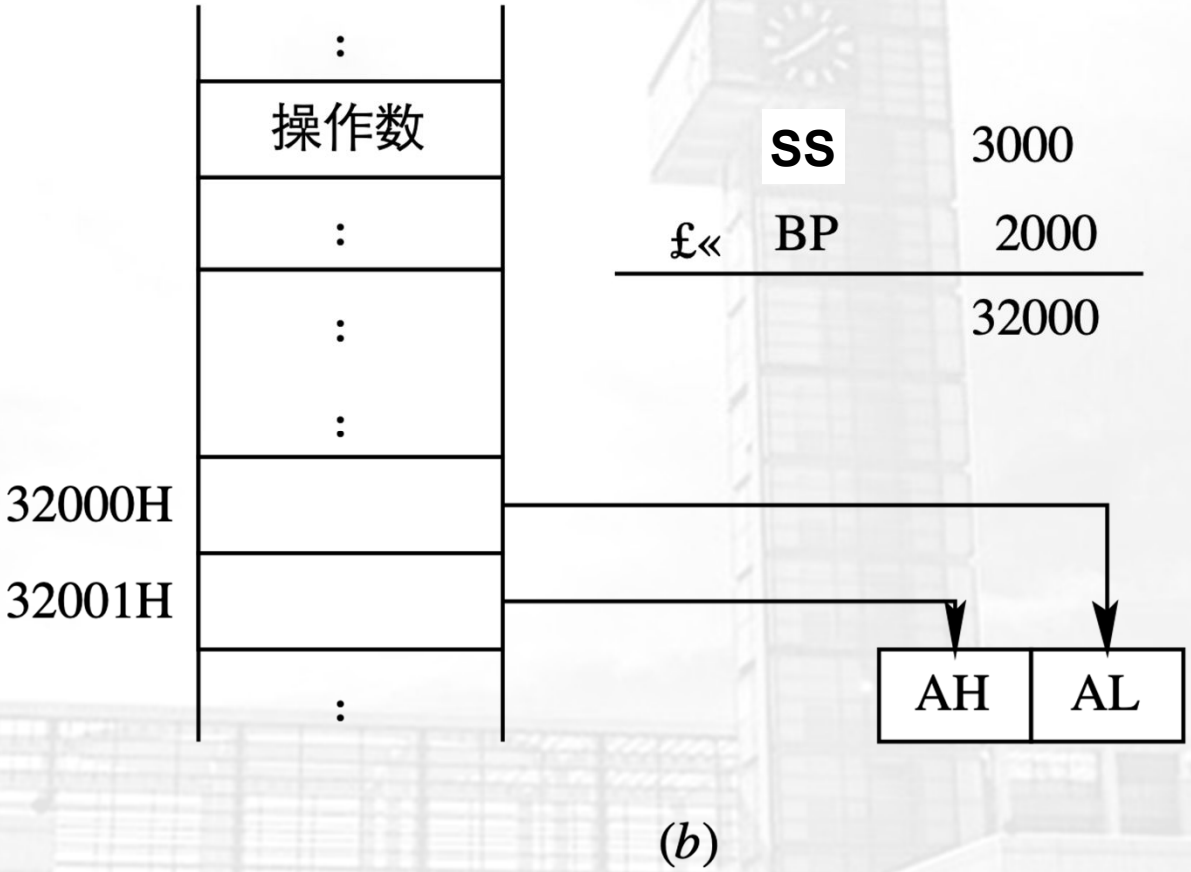
```
MOV AX, 2000H
MOV DS,  AX
MOV SI,  1000H
MOV AX,  [SI]
```



寻址方式 — 寄存器间接寻址

- **例如：**

```
MOV AX, 3000H
MOV SS,  AX
MOV BP,   2000H
MOV AX,  [BP]
```



寻址方式 — 寄存器间接寻址

- **例:假设有指令：MOV BX,[DI]，在执行时，(DS)=1000H，(DI)=2345H，存储单元12345H的内容是4354H。问执行指令后，BX的值是什么?**

- **解:**

根据寄存器间接寻址方式的规则，在执行本例指令时，寄存器DI的值不是操作数，而是操作数的地址。该操作数的物理地址(Physical Address，PA)应由DS和DI的值形成，即:

PA=(DS)*16+DI=1000H*16+2345H=12345H。

所以，该指令的执行效果是：把从物理地址为12345H开始的一个字的值传送给BX。

寻址方式 — 寄存器间接寻址

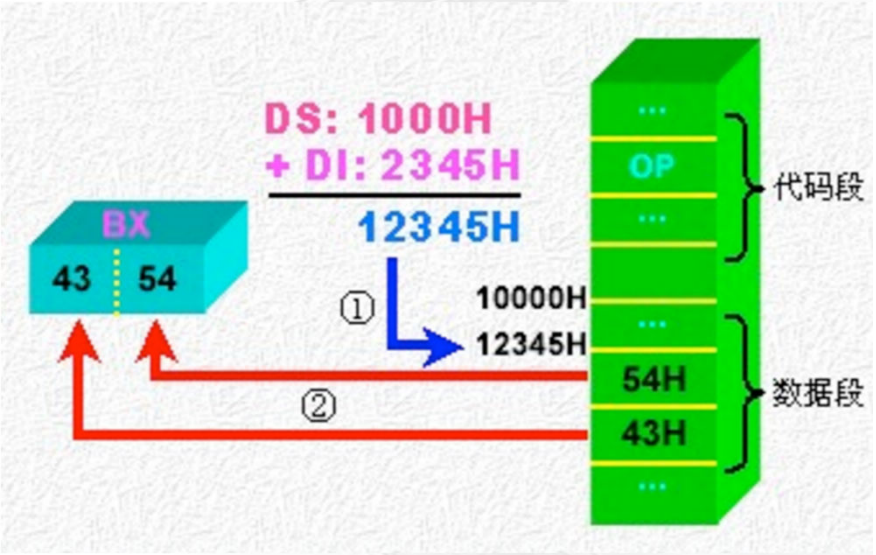
- **例:假设有指令：MOV BX,[DI]，在执行时，(DS)=1000H，(DI)=2345H，存储单元12345H的内容是4354H。问执行指令后，BX的值是什么?**

- **解:**

根据寄存器间接寻址方式的规则，在执行本例指令时，寄存器DI的值不是操作数，而是操作数的地址。该操作数的物理地址(Physical Address, PA)应由DS和DI的值形成，即:

PA=(DS)*16+DI=1000H*16+2345H=12345H。

所以，该指令的执行效果是：把从物理地址为12345H开始的一个字的值传送给BX。



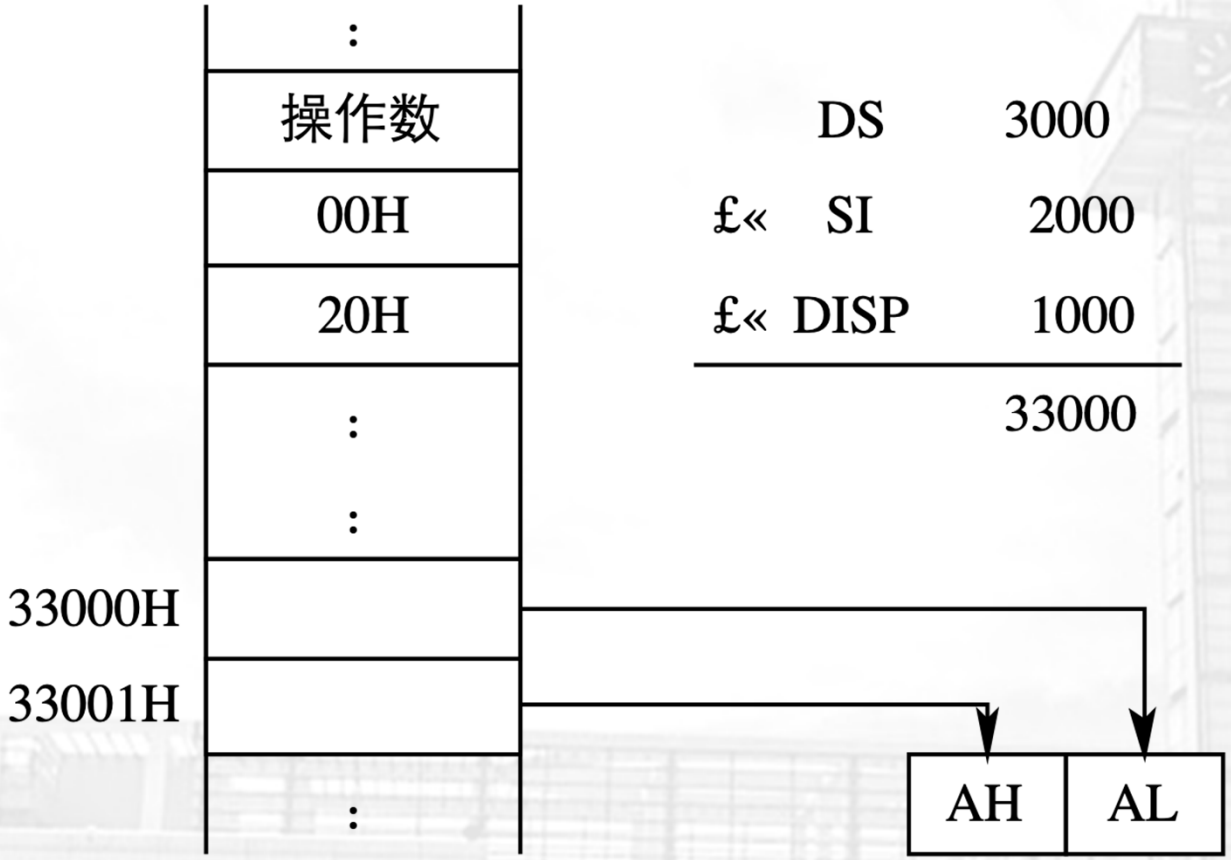
寻址方式 — 寄存器相对寻址

- 本质是寄存器间接 + 相对寻址：在寄存器间接的基础上再加上地址偏移值DISP；
- DISP跟着操作码后面，在代码段！！！
- 同样分两种情况：
  - 如果是SI/DI/BX　<—搭配DS在数据段找操作数
  - 如果是BP　<—搭配SS在堆栈段找操作数

例：MOV AX，DISP［SI］

寻址方式 — 寄存器相对寻址

例：MOV AX，DISP［SI］； DISP=1000H ; DS = 3000H; SI = 2000H



寻址方式 — 寄存器相对寻址

例: 假设指令：MOV BX, [SI+100H]，在执行它时，(DS)=1000H，(SI)=2345H，内存单元12445H的内容为2715H，问该指令执行后，BX的值是什么?

解：根据寄存器相对寻址方式的规则，在执行本例指令时，源操作数的有效地址EA为：

EA=(SI)+100H=2345H+100H=2445H

该操作数的物理地址应由DS和EA的值形成，即：

PA=(DS)*16+EA=1000H*16+2445H=12445H。

所以，该指令的执行效果是：把从物理地址为12445H开始的一个字的值传送给BX。

寻址方式 — 寄存器相对寻址

例: 假设指令：MOV BX, [SI+100H]，在执行它时，(DS)=1000H，(SI)=2345H，内存单元12445H的内容为2715H，问该指令执行后，BX的值是什么？

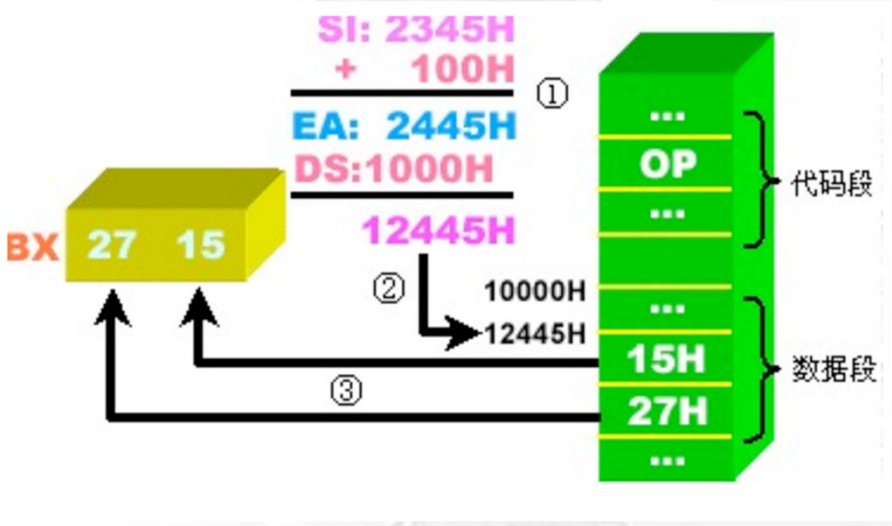
解：根据寄存器相对寻址方式的规则，在执行本例指令时，源操作数的有效地址EA为：

EA=(SI)+100H=2345H+100H=2445H

该操作数的物理地址应由DS和EA的值形成，即：

PA=(DS)*16+EA=1000H*16+2445H=12445H。

所以，该指令的执行效果是：把从物理地址为12445H开始的一个字的值传送给BX。



数据寻址方式 — 总结

- **加起来！！！ 把右边的地址值全加起来！！！**
- **SI/DI/BX找DS！！！！**
- **BP找SS！！！！**
- **“直接” v.s “间接”**
- **“相对” v.s “变址”**

源操作数	指令的变形	源操作数的寻址方式
只有偏移量	MOV AX, [100H]	直接寻址方式
只有一个寄存器	MOV AX, [BX] 或 MOV AX, [SI]	寄存器间接寻址方式
有一个寄存器和偏移量	MOV AX, [BX+100H] 或 MOV AX, [SI+100H]	寄存器相对寻址方式
有二个寄存器	MOV AX, [BX+SI]	基址加变址寻址方式
有二个寄存器和偏移量	MOV AX, [BX+SI+100H]	相对基址加变址寻址方式

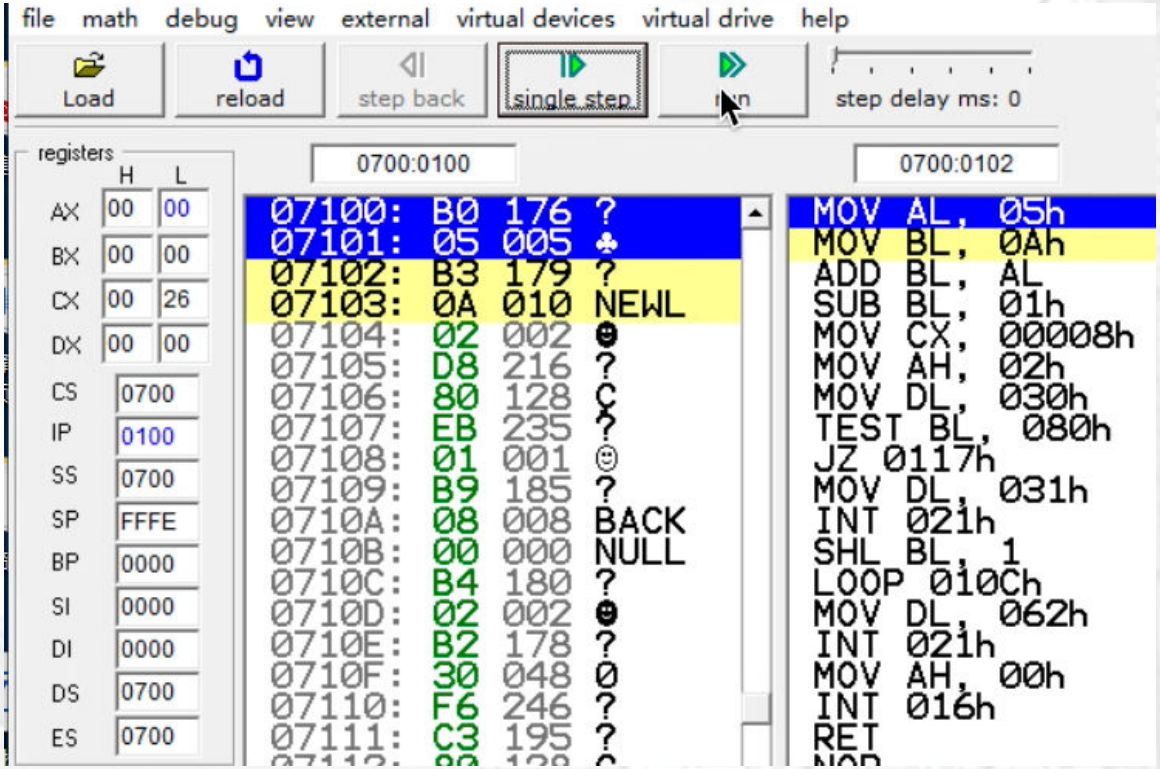
地址，
除了数据，
还有指令本身

指令寻址

- **默认寻址**
- **段内转移**
- **段内间接转移**
- **段间转移**
- **段间间接转移**

指令寻址

1．默认寻址：顺序执行指令时，PC(IP)寄存器总是指向下一条指令的地址



指令寻址

段内，指哪个段?

数据寻址时，“直接”指什么?

2．段内直接寻址：转向的有效地址是当前IP内容和指令指定的8位或16位位移量之和。

- **例如： JMP 2000H**

指令寻址

段内，指哪个段?

数据寻址时，“间接”指什么?

2．段内间接寻址方式

- 段内间接寻址方式：用寄存器或存储单元中的内容修改IP

- 例如： JMP BX

  JMP WORD PTR [SI + 2000H]

  (JMP WORD PTR:段内转移)

指令寻址

数据寻址时，“直接”指什么?

3．段间直接寻址方式

■段间直接寻址方式: 直接给出了转向的段基址和偏移地址，16位的段基址用来更新CS，16位的偏移地址用来更新IP

■如 JMP FAR PTR DS:0x00001234

(JMP FAR PTR)段间转移

指令寻址

数据寻址时，“间接”指什么?

4．段间间接寻址方式

- 段间间接寻址方式：低位字单元取代IP的内容，高位字单元取代CS，从而实现段间程序转移。
- 例：JMP DWORD PTR DS:[EBX], 当DS=0x440, EBX=0x00006200, DS:0x00006200=0x00003500
  - JMP DWORD PTR DS:[EBX] => JMP 0x440:0x00003500

4.3 8086的指令系统

8086的指令系统

- 数据传送指令
- 算术运算指令
- 程序控制指令
- 逻辑运算和移位循环指令
- 串操作指令
- 处理器控制指令
- 输入/输出指令

数据传送指令

- **MOV指令**

格式：　　**MOV　OPRD1，　OPRD2**

目的　　源

功能：　　**（OPRD2）→（OPRD1）**

其中：　目的操作**OPRD1**　　源操作数　**OPRD2**

存储器	**mem**	立即数	**im**
通用寄存器	**r**	存储器	**mem**
段寄存器	**SEG（除CS）**	通用寄存器	**r**
		段寄存器	**SEG（含CS）**

但也有4种例外情况：

数据传送指令

- 目的操作数不能为立即数，或 CS、IP；
- 内存单元之间不能进行数据直接传送；
- 立即数不能直接传送到段寄存器中；
- 段寄存器之间的不能进行数据直接传送；

数据传送指令

- **在CPU各内部寄存器间传送数据**

  MOV AL, BL

  MOV AX, DX

- **立即数传送至CPU的内部通用寄存器，给寄存器赋值**

  MOV CL，4

  MOV AX，03FFH

- **CPU内部寄存器与存储器间的数据传送**
  - 在CPU的通用寄存器与存储器之间传送数据。

  MOV AL，BUFFER

  MOV AX，[SI]

数据传送指令

- 在CPU寄存器与存储器之间的传送数据

MOV DS，DATA[SI+BX]

MOV DEST[BP+DI], ES

MOV指令不能在两个存储器单元之间进行数据直接传送。为了实现两个存储器单元之间的数据传送，必须用内容寄存器作为中介。

MOV AL，AREA1

MOV AREA2，AL

数据传送指令

2. 交换指令

格式：　　XCHG　OPRD1，　　OPRD2

目的　　源

功能：　（OPRD2）←→ (OPRD1)

其中：这种交换能在通用寄存器与累加器之间、通用寄存器之间、通用寄存器与存储器之间进行。

3. 地址传送指令

格式：　　LEA　OPRD1，　　OPRD2

目的　　源

功能：　OPRD2→ (OPRD1)

其中：OPRD1为16位通用寄存器，OPRD2为内存操作数。

LEA BX, BUFR

数据传送指令

4. 堆栈指令

格式：　PUSH　OPRD ；　r、 mem、 SEG（含CS）
　　POP　　OPRD ；　r、 mem、 SEG（除CS)

```
MOV AX，8000H
MOV SS，AX
MOV SP，2000H
MOV DX，3E4AH
PUSH DX
PUSH AX
```

数据传送指令

5.字节、字转换指令

CBW指令，将字节转换成一个字。
CWD指令，将字转换成双字。

6.标志寄存器传送指令

LAHF，将标志寄存器的低字节传送到AH中。
(Load Flags into AH)
SAHF，将AH的内容传送到标志寄存器的低字节中。 (Store AH into Flags)
PUSHF，将标志寄存器的内容压入堆栈。
POPF，由堆栈弹出一个字放入标志寄存器。

算术运算指令

- 加、减、乘、除
- 定点有符号整数、浮点数、压缩十进制数
- 单操作数指令
  - 有符号数的绝对值（Absolute）
  - 求操作数的相反数（Negate）
  - 递增操作数（Increment）
  - 递减操作数（Decrement）
- 由ALU完成

4.3.2 算术运算指令

1．加法指令

ADD OPRD1，OPRD2 ;(add)

功能：（OPRD1）+（OPRD2）→(OPRD1)

ADC OPRD1，OPRD2 ;(add with carry)

功能：（OPRD1）+（OPRD2）+*CF*→(OPRD1)

INC OPRD ;(increment)

功能：（OPRD）+1→(OPRD)

- 影响标志位AF、 CF、 OF、 PF、 SF、ZF（INC不影响CF）。

算术运算指令

运算结果标志位

1、进位标志CF(Carry Flag)
2、奇偶标志PF(Parity Flag)
3、辅助进位标志AF(Auxiliary Carry Flag)
4、零标志ZF(Zero Flag)
5、符号标志SF(Sign Flag)
6、溢出标志OF(Overflow Flag)

状态控制标志位

1、追踪标志TF(Trap Flag)
2、中断允许标志IF(Interrupt-enable Flag)
3、方向标志DF(Direction Flag)

算术运算指令

■ 例:

```
ADD  AL，30
ADD  BX，3000H
ADD  AX，SI
ADD  DX，DATA[BX＋SI]
ADD  [SI]，100
ADD  [BX]，AX
```

■ 例:

INC常用于在循环程序中修改地址指针和循环次数等。

```
INC  AL
INC  [SI]
```

算术运算指令

■ 例：ADC指令用于多字节运算

（DX、AX）= 0002，F365H

\+ （BX、CX）= 0005，E024H

---------- ***双精度加法***

ADD AX，CX

ADC DX，BX

第一条指令，

```
      F365
    + E024
    ------
   ↙1, D389
```

CF=1　　结果 （AX）= D389H，CF=1，SF=1，ZF=0，　OF=0

第二条指令，

```
      0002
      0005
    +    1（CF）
    ------
      0008
```

CF=0　　结果 （DX）= 0008H，CF=0，SF=0，ZF=0，　OF=0

算术运算指令

2．减法指令

SUB OPRD1，OPRD2 ;(减！)

功能：（OPRD1）-（OPRD2）→（OPRD1）

SBB OPRD1，OPRD2 ;(借位减！)

功能：（OPRD1）-（OPRD2）- CF →(OPRD1)

DEC OPRD ;(自减！)

功能：（OPRD）-1 → (OPRD)

NEG OPRD ;(negate) 取补指令

功能：FFFFH -（OPRD）+1 → (OPRD)

CMP OPRD1，OPRD2 ;(compare) 比较指令

功能：（OPRD1）-（OPRD2）

■影响标志位AF、CF、OF、PF、SF、ZF（DEC不影响CF）。

算术运算指令

3．乘法指令

(1) 无符号数乘法指令MUL

MUL OPRD ;(unsigned multiple)

功能： 若OPRD为8位，（AL）*（OPRD）→ （AX）

若OPRD为16位，（AX）*（OPRD）→ （DX、AX）

(2) 带符号数乘法指令IMUL

IMUL OPRD ;(signed multiple)

功能： 若OPRD为8位，（AL）*（OPRD）→ （AX）

若OPRD为16位，（AX）*（OPRD）→ （DX、AX）

算术运算指令

4．除法指令

(1) 无符号数除法指令DIV

DIV OPRD ;(unsigned divide)

功能：若OPRD为8位，(AX) ÷ (OPRD) 商 → (AL)
余数 → (AH)

若OPRD为16位，(DX，AX) ÷ (OPRD) 商 → (AX)
余数 → (DX)

(2) 带符号除法指令ID1V

IDIV OPRD ;(signed divide)

功能：若OPRD为8位，(AX) ÷ (OPRD) 商 → (AL)
余数 → (AH)

若OPRD为16位，(DX，AX) ÷ (OPRD) 商 → (AX)
余数 → (DX)

■采用MUL或IMUL，DIV或IDIV由运算数的格式决定。

逻辑运算与移位

1. 逻辑运算

NOT OPRD

功能：NOT（OPRD）→(OPRD)　　；按位取反

AND OPRD1，OPRD2

功能：（OPRD1）AND（OPRD2）→(OPRD1)；按位与

OR OPRD1，OPRD2

功能：（OPRD1）OR（OPRD2）→(OPRD1)；按位或

XOR OPRD1，OPRD2

功能：（OPRD1）XOR（OPRD2）→(OPRD1)；按位异或

TEST OPRD1，OPRD2

功能：（OPRD1）AND（OPRD2）　　；按位与，结果不保存

逻辑运算与移位

例：逻辑指令的使用

AND 指令使操作数的某些位不变，其它位为0。

OR 指令使操作数的某些位不变，其它位为1。

XOR 指令使操作数的某些位不变，其它位取反。

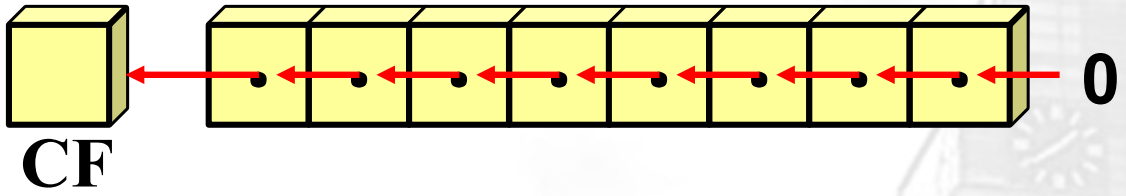
TEST指令在不操作数的前提下，用来检测某几位的状态。

A	B	NOT A	A AND B	A OR B	A XOR B	A==B
0	0	1	0	0	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	0
1	1	0	1	1	0	1

逻辑运算与移位

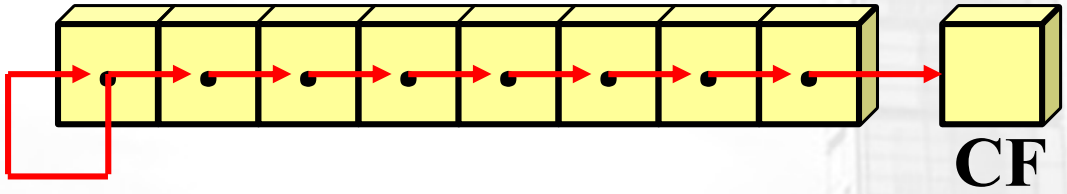
SAL / SHL OPRD，m. (Shift Arithmetic / Logic Left)

逻辑左移/
算数左移：



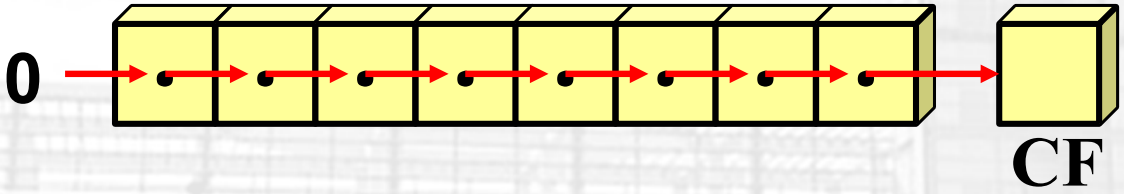
SAR OPRD，m (Shift Arithmetic Right)

算数右移：



SHR OPRD，m (Shift Logic Right)

逻辑右移：

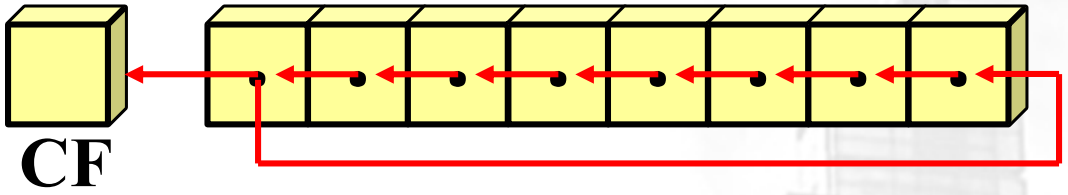


● OPRD为通用寄存器 或 内存操作数。
M为立即数或CL，决定移位次数。

逻辑运算与移位

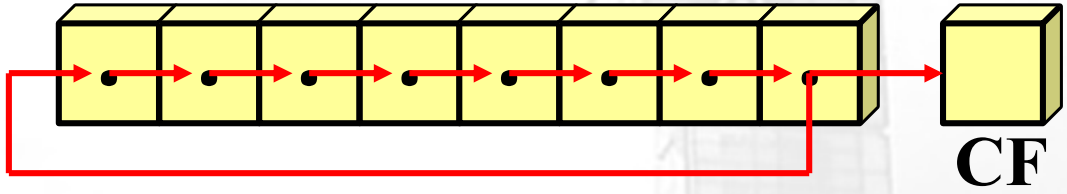
ROL OPRD，m (Rotate Left)

不带进位循环左移：



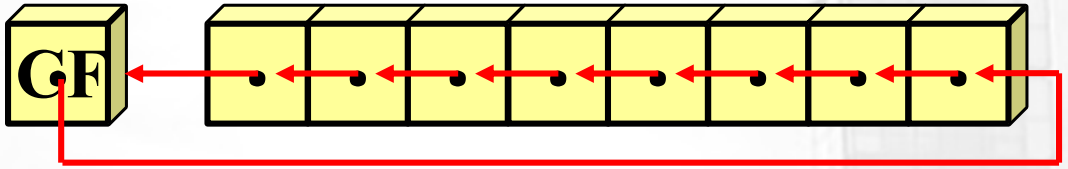
ROR OPRD，m (Rotate Right)

不带进位循环右移：



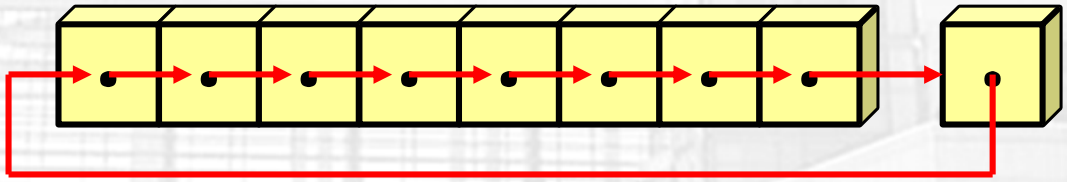
RCL OPRD，m (Rotate Left with Carry)

带进位循环左移：



RCR OPRD，m (Rotate Right with Carry)

带进位循环右移：



- OPRD为通用寄存器 或 内存操作数。M为立即数或CL，决定循环移位次数。

串操作指令

- **串操作**：某个**连续的内存区**中存放的一串字或字节，可以是BIN数，也可以是BCD码或ASCII码。若**对其中的每个字或字节均作同样的操作**，就称为串操作。
- **串操作指令**：完成串操作功能的指令称为字符串操作指令，简称为串操作指令。
- **串操作指令中的默认寻址**：

DS:SI	源串
ES:DI	目的串
CX	计数器
AL/AX	累加器
DF	方向标志，DF=0，正向；DF=1，负向

串操作指令

1. MOVS、STOS、LODS与REP配合

■MOVS 串传送指令

MOVS DST，SRC ；SRC 源串，DST目的串

MOVSB ；字节传送

MOVSW ；字传送

功能: (DS：SI) → (ES：DI)

MOVSW	Move word from string to string	if (DF==0)
		(word)DI++ = *(word*)SI++;
		else
		(word)DI-- = *(word*)SI--;

串操作指令

■ STOS 串存贮指令（store to string)

STOS DST；DST目的串

STOSB ；字节操作

STOSW ；字操作

功能:

■ LODS 串装入指令 (load from string)

LODS SRC； SRC源串

LODSB ；字节操作

LODSW ；字操作

串操作指令

■ REP 串指令前缀

REP MOVS

REP STOS

REP LODS

功能：重复串指令执行，重复次数为（CX）。

过程如下：

1.若（CX）=0，则退出，否则执行2；

2.（CX）-1 → （CX）；

3.执行串指令；

4.重复1 – 3。

串操作指令

例：1000H：0100H为首地址，长度为1000字节的串，传送到2000H：0000H开始的串。

```
MOV  AX，1000H
MOV  DS，AX
MOV  SI，0100H
MOV  AX，2000H
MOV  ES，AX
MOV  DI，0000H
MOV  CX，1000
CLD          ；DF=0 (Clear Direction Flag)
REP MOVSB (Repeat MOVSB for 1000 times)
```

程序控制指令

- 程序控制指令主要是指程序转移指令。
- 转移类指令通过改变CS与IP的值或仅改变IP的值，以改变指令执行的顺序。

1.无条件转移指令 JMP

该指令分直接转移和间接转移两种。

直接转移又可分短程(SHORT)、近程(NEAR)和远程(FAR) 3种形式。

- 短程转移 JMP SHORT OPRD
- 近程转移 JMP NEAR OPRD
- 远程转移 JMP FAR OPRD

程序控制指令

2.条件转移指令

条件转移指令将上一条指令所设置的某些标志位的状态作为测试条件，条件满足则转向指令中所指示的目的地址。

条件转移指令只能为短程(SHORT) 1种形式。

(1) 以单个标志位为条件

```
JO      OPRD        ；溢出转移  (Overflow)
JNO     OPRD        ；不溢出转移 (Not Overflow)
JS      OPRD        ；结果为负转移 (Sign)
JNS     OPRD        ；结果为正转移 (No Sign)
JC      OPRD        ；进位转移 (Complement)
JNC     OPRD        ；无进位则转移 (Not Complement)
JE / JZ OPRD        ；等于或为零转移(Equal / Zero)
JNZ     OPRD        ；不为零转移(Not Zero)
```

程序控制指令

(2) 无符号数比较

JA / JNBE OPRD ; 高于或不低于等于转移 (Above / Not Below nor Equal)
JAE / JNB OPRD ; 高于等于或不低于转移 (Above or Equal / Not Below)
JB / JNAE OPRD ; 低于或不高于等于转移 (Below / Not Above nor Equal)
JBE / JNA OPRD ; 低于等于或不高于转移 (Below or Equal / Not Above)

(3) 带符号数比较

JG / JNLE OPRD ; 大于或不小于等于转移 (Greater / Not Less or Equal)
JGE / JNL OPRD ; 大于等于或不小于转移 (Greater or Equal / Not Less)
JL / JNGE OPRD ; 小于或不大于等于转移 (Less / Not Greater nor Equal)
JLE / JNG OPRD ; 小于等于或不大于转移 (Less or Equal / Not Greater)

3.循环控制指令 LOOP

LOOP OPRD ; CX-1, if CX≠0循环