第3章 实方式指令寻址与指令系统

- 3.1 指令的基本寻址方式
- 3.2 实方式32位指令寻址
- 3.3 实方式指令系统
- 3.4 字符设备I/O功能调用

3.1指令的基本寻址方式

- 8086 CPU的指令系统的基本指令包括:
- 数据传送类指令、算术运算类指令、位操作类指令、串操作类指令、控制转移类指令、处理机控制类指令等。后几节将分别介绍这些指令的语句格式和功能。

3.1.1指令的基本格式

- 大多数数据传送类指令、算术运算类指令、位操作类指令及串操作类指令，其操作数指令有相同的语句格式和操作规定。
- 指令一般由操作码OP 、寻址方式MOD和一个或多个操作数OD等字段组成。

操作码OP	寻址方式MOD	操作数OD

指令的表示形式

- PC微机的多数指令有一个或两个操作数，当然也可以没有操作数，常见的表示形式如下：

OP 操作数

OP 目的操作数 源操作数

OP

示例:

- 单操作数指令只需指定一个操作数。例如将寄存器AX的内容加1后，其和回送到AX，指令的符号表示形式为:
- INC　AX　；AX+1->AX

- 双操作数指令需要指定两个操作数.当源操作数和目的操作数进行运算后，多数指令将其结果回送到目的操作数的位置。例如寄存器BX与CX的内容相减后，其差送入BX的指令为:
- SUB　BX，CX ；BX-CX->BX

- 没有操作数指令，即隐含操作数指令，是指在OD位置虽然未明确给出操作数，但在指令OP中隐含有事先安排的操作数。例如:
- CLC　　；　0->CF

表示将标志寄存器中的CF位清0。

3.1.2　寻址方式

- 立即寻址
- 寄存器寻址
- 直接寻址
- 寄存器间接寻址
- 寄存器相对寻址
- 基址变址寻址
- 相对基址变址寻址

1. 立即寻址(Immediate Addressing)

- 立即寻址方式中，指令操作码和操作数都在存储器代码段中。
- 汇编格式：n（n为立即操作数，是用8位或16位二进制补码表示的有符号数）
- 功能：操作数存放在存储器，指令下一单元的内容为立即操作数n。
- 图形表示：

例如：

```
MOV   AX,2018H     ;2018H→AX,立即数(常数)只能用于源操作数
MOV   CL,96H       ;96H→CL
```

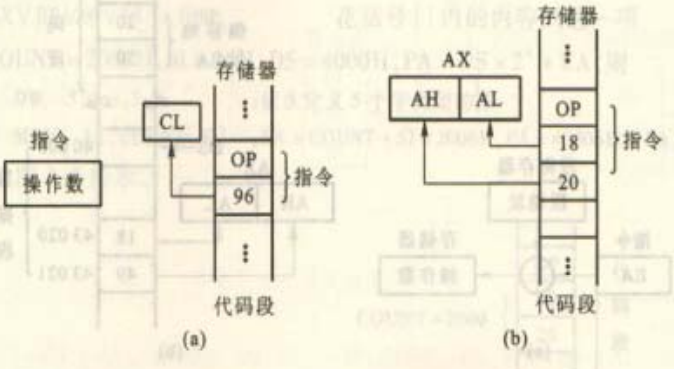


图 3.1 立即寻址示意图

2. 寄存器寻址(Register Addressing)

操作数在指令说明的寄存器中,运算速度较高。一个 16 位操作数寄存器可以是 AX、BX、CX、DX、SI、DI、SP 和 BP,对 8 位操作数的一个寄存器可以是 AL、AH、BL、BH、CL、CH、DL 和 DH。寻址如图 3.2 所示,例如:设 AX = 0180H,执行:

```
MOV   DX,AX        ;AX→DX,DX = 0180H
```

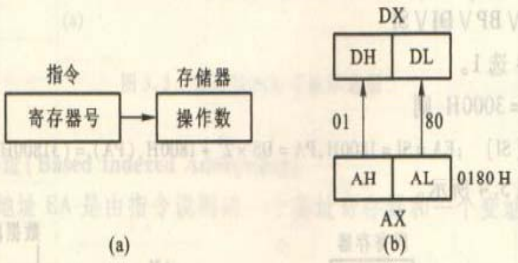


图 3.2 寄存器寻址示意图

3. 直接寻址(Direct Addressing)

指令操作码 OP 后直接给出操作数的 16 位偏移地址 EA。OP 与直接地址在代码段,操作数一般在数据段。

例如:设 DS = 4000H,EA = 3020H,PA = DS × 2^4 + EA = 43020H,(PA) = (43020H) = 4918H,则

```
MOV   AX,[3020H]   ;[3020H]表示地址,将 43020H 单元的内容→AX,AX = 4918H
```

寻址过程如图 3.3 所示。

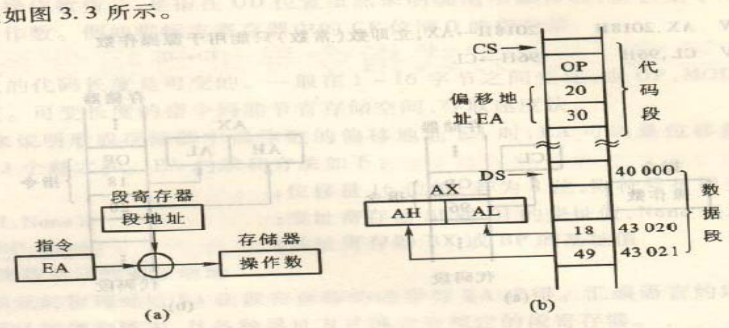


图 3.3 直接寻址示意图

又如

```
VAR   DW   86      ;定义 86 为字类型数,设 DS = 4000H,VAR 的 EA = 10H
      MOV  AX,VAR  ;PA = DS × 2^4 + EA = 40010H,(PA) = (VAR) = 86→AX
```

4. 寄存器间接寻址(Register Indirect Addressing)

操作数的偏移地址 EA 在指令说明的基址寄存器或变址寄存器中。即

$$EA = BX \vee BP \vee DI \vee SI$$

符号∨表示或者,4 选 1。

设 SI = 1800H,DS = 3000H,则

MOV AX,[SI] ;EA = SI = 1800H,PA = DS × 2^4 + 1800H,(PA) = (31800H) = 125AH→AX

寻址过程如图 3.4 所示。

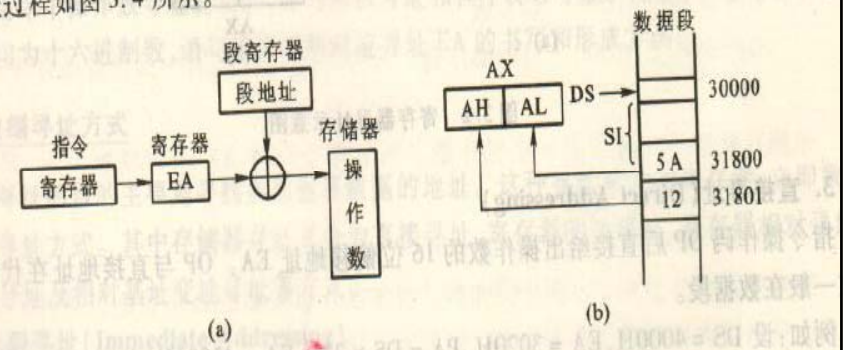


图 3.4 寄存器间接寻址示意图

5. 寄存器相对寻址(Register Relative Addressing)

操作数的偏移地址 EA 是指令中 8 位或 16 位位移量($DISP_{8,16}$)加上基址寄存器或变址寄存器的内容之和。即

$$EA = \{BX \vee BP \vee SI \vee DI\} + DISP_{8,16}$$

花括号{}内的内容只选一项

例如:设 COUNT = 2000H,SI = 06H,DS = 4000H,PA = DS × 2^4 + EA,则

```
COUNT   DW   5,6,3,7,8          ;依次定义 5 个字类型的数
        MOV  AX,COUNT[SI]       ;EA = COUNT + SI = 2006H,PA = 42006H,(PA) = 0007H→AX
```

寻址过程如图 3.5 所示。

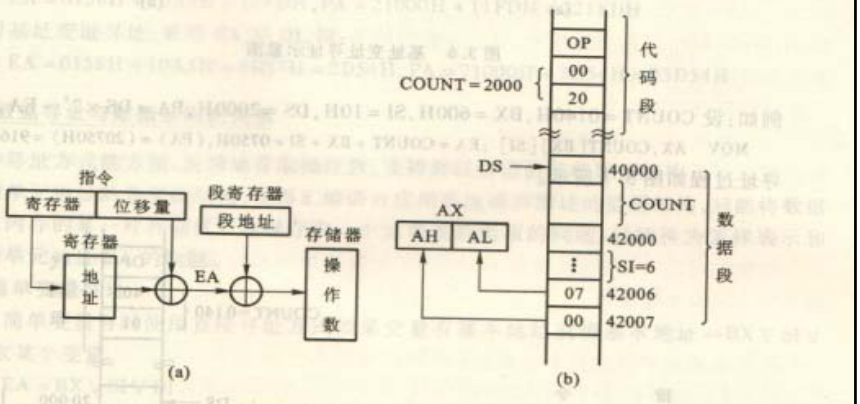


图 3.5 寄存器相对寻址示意图

6. 基址变址寻址(Based Indexed Addressing)

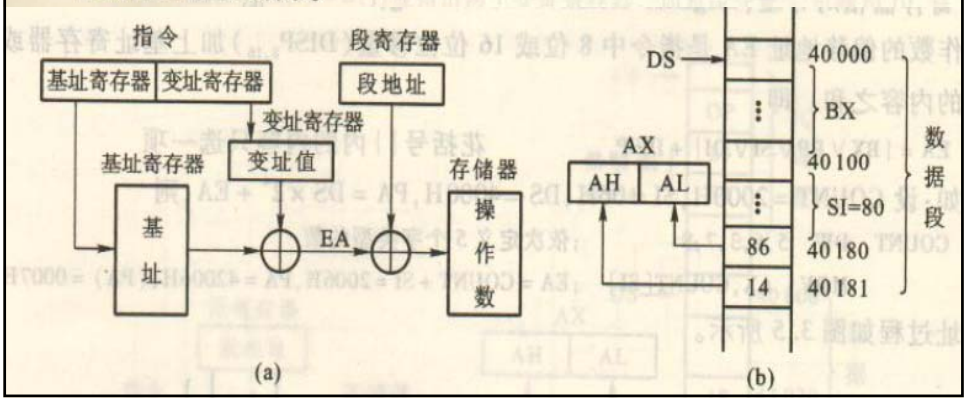
操作数的偏移地址 EA 是由指令说明的一个基址寄存器和一个变址寄存器的内容之和。即

$$EA = |BX \vee BP| + |SI \vee DI|$$

例如:设 BX = 100H,SI = 80H,DS = 4000H,PA = DS × 2^4 + EA,则

```
MOV  AX,[BX][SI]    ;EA = BX + SI = 180H,PA = 40180H,(PA) = 1486H→AX
```

寻址过程如图 3.6 所示。



7. 相对基址变址寻址(Relative Based Indexed Addressing)

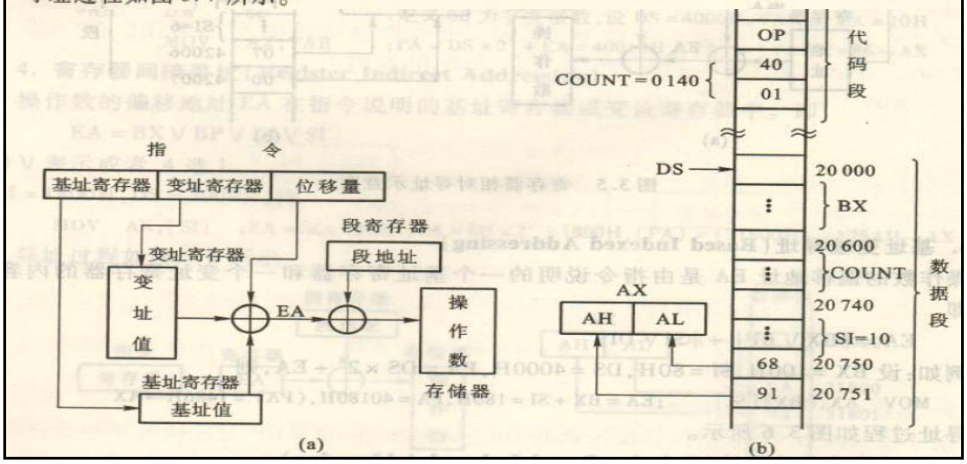
操作数的偏移地址 EA 是指令中一个 8 位或 16 位位移量($DISP_{8,16}$)加上基址寄存器和变址寄存器的内容之和。即

$$EA = |BX \vee BP| + |SI \vee DI| + |DISP_{8,16}|$$

例如:设 COUNT = 0140H,BX = 600H,SI = 10H,DS = 2000H,PA = DS × 2^4 + EA,则

```
MOV  AX,COUNT[BX][SI] ;EA = COUNT + BX + SI = 0750H,(PA) = (20750H) = 9168H→AX
```

寻址过程如图 3.7 所示。



- 寻址方式小结：
- 若BX=0158H,DI=10A5H,位移量DISP=1857H，数据段寄存器DS=2100H，则这些量形成EA和PA及各种不同寻址方式如下：
  - 直接寻址：EA=DISP=1857H
    PA=DS×2^4+EA=21000H+1857H=22B57H
  - 寄存器寻址：没有EA，数据在说明的寄存器。
  - 寄存器间接寻址：假定采用BX，则：
    EA=BX=0158H．PA=DS x2^4+EA=21000H+0158H=21158H
  - 寄存器相对寻址：假定采用BX．则：
    EA=BX十D1SP=01 58H+1857H=1CAFH
    PA=21000H+1CAFH=22CAFH
  - 基址变址寻址：若采用BX和DI，则：
    EA=0158H+10A5H=11FDH．PA=21000H+11FDH=221FDH
  - 相对基址变址寻址：采用BX和DI，则：
    EA=0158H+10A5H+1857H=2D54H
    PA=21000H+2D54H=23D54H

3.1.3数据寻址与数据结构的关系

- 多种寻址方式能方便、灵活的存取操作数，支持高级语言的某些数据结构。
- 1.简单变量寻址
- 2.数组或表格数据的寻址
- 3.记录型数组的寻址

1.简单变量寻址

存取简单变量有时使用直接寻址方式, 如果变量有基本地址就将基本地址->BX \/ SI \/ DI, 则存取某个变量 EA = BX \/ SI \/ DI

2.数组或表格数据的寻址

表格数据的存取也可当作是数组的存取,均可用寄存器间接、寄存器相对、基址变址和相对基址变址等寻址方式。

- (1)存取基本数组:数组的基本地址->BX,某个元素到数组基本地址的距离->SI \/ DI, 则:

  EA = BX + { SI \/ DI}

  或者位移量DISP=数组开始地址, BX \/ SI \/ DI = 数组元素到数组开始地址的距离,则:

  EA = BX + { SI \/ DI } + DISP

2.数组或表格数据的寻址（续）

- (2)对于赋值语句A(i) = B (j)应指出两个变址寄存器。如对应元素a_i可使用DI; 对应元素b_j用SI, 则两组元素的地址分别为:

  EA = SI + DISP

  EA = DI + DISP

- (3)存取二维数组A(i, j): 数组起始地址->BX, BX兼行元素相对数组起始地址变址,而列元素又相对行元素的变址量->SI \/ DI, 则存取某个元素应有:

  EA = {BX + SI} \/ {BX + DI}

- 若DISP = 数组元素的起始地址, BX = 行元素变址量, SI 或 DI = 列元素相对行元素的变址,则存取某个元素应有:

  EA = BX + {SI \/ DI} + DISP

3.记录型数组的寻址

- 数组和记录结合使用。 设雇员的记录有四个数据项(雇员、保险号、雇龄和工资),由多个雇员的记录组成一个数组,则任一雇员记录项地址的形式描述为 F=f(M, I, J)

其中,F为数组记录项的地址,M为数组的基本地址, I为数组的元素(记录), J为元素的数据项, f则表示记录型数组。 对此,存取任一记录中的数据项(如雇龄项)的操作数寻址对应描述为： 偏移地址 = f(基址,变址,位移)

3.记录型数组的寻址（续）

- 假定基址指针再**BX**,与数组元素相一致的变址值在**SI**,记录中项的位置用**DISP**表示,则

  EA = BX + SI + DISP。

- **4.堆栈数据结构寻址**
- 堆栈采用以**BP**为基址指针的寻址,方便数据结构相同而参数不同的数据存取。当存取堆栈中的简单变量时

  EA = BP + DISP

  若存取堆栈中的数据和记录,则

  EA = BP +{SI \/ DI} + DISP

3.1.4 程序转移寻址

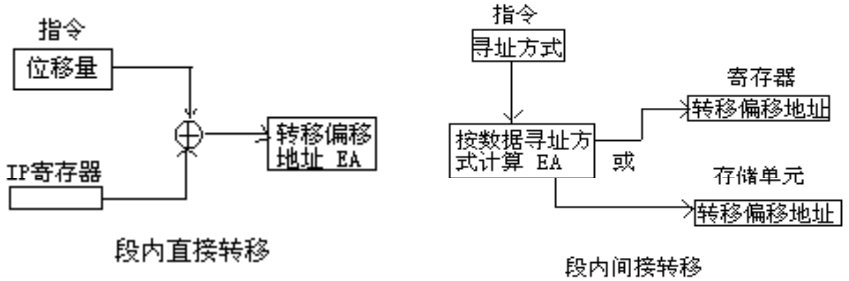
- 寻址方式对多数指令而言,是要计算出操作数的地址,但是也由少数指令是为了形成程序转移的地址(如无条件转移指令**JMP**、调用指令**CALL**等)。
- 程序正常顺序执行时,每取出一条指令执行**IP+n->IP**,其中**n**为取出指令的字节数;然后形成下一条指令的地址: **PA = CS * 2^4 + IP**
- 但是如果程序发生转移时,需要计算出转移偏移地址**EA**并修改**IP**,有时还需要修改**CS**的值。这种情况操作的对象是一个地址,地址的内容是要取出的指令,而不是上述讲到的操作数,因此称之为程序转移寻址。

程序转移方式：

- 段内转移是指程序在同一段代码内,仅改变IP的值而不改变CS的值所发生的转移。
- 而段间转移是程序要从一个代码段转移到另一个代码段,则不仅改变IP的值,同时也要改变CS的值。
- 无论是段内还是段间发生的转移都有直接和间接的形式,因此程序转移有四种寻址。

段内转移图示

- 段内直接转移和间接转移寻址如下图:



段内转移过程

- **1.段内直接寻址(Intrasegment Direct Addressing)**

  转移偏移地址EA是指令中8位或16位位移量($DISP_{8,16}$)与指令指针IP当前内容之和。即:

  $EA = IP + DISP_{8,16}$ -> IP

```
    JMP   NLAB ;近跳转      JMP  SHORT SLAB ;短跳转      JE    SLAB   ;短跳转
    :                       :                           :
NLAB:ADD   AX,BX        SLAB:MOV   DX,CX            SLAB:INC   SI
```

3字节（偏移量16位）　　2字节（偏移量8位）　　2字节（偏移量8位）

- 例1．已知JMP L1指令放在2000H的偏移地址中，L1标号的地址为1005H，求该指令的转移偏移量？
- 解：该指令为长转移指令，转移偏移量为16位补码，指令本身为3字节指令。
- 所以：2000H+3+ 转移偏移量 = 1005H
- 则：转移偏移量 =1005H-2003 H=1005H+DFFDH=F002（ -0FFE H）
- 即：向上转移
-
- 例2．已知JZ L1指令放在2000H的偏移地址中，L1标号的地址为2005H，求该指令的转移偏移量？
- 解：该指令为短转移指令，转移偏移量为8位补码，指令本身为2字节指令。
- 所以：2000H+2+ 转移偏移量 = 2005H
- 则：转移偏移量 = 2005H-2002H = 03H　向下转移。
-

段内转移过程

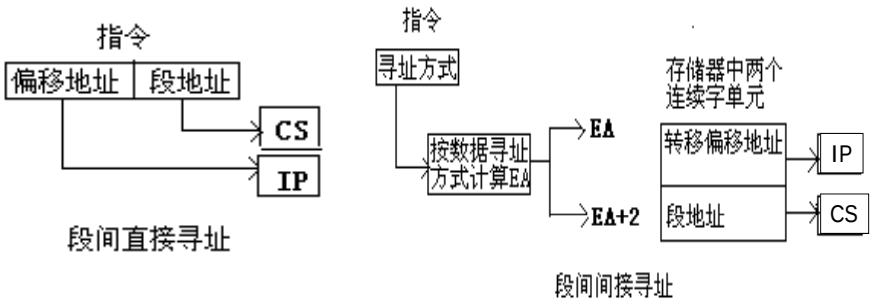
- **2.段内间接寻址(Intrasegment Indirect Addressing)**

转移偏移地址 **EA** 如果指定的是**16**位的寄存器,则将寄存器的内容**->IP**。 如果指定的是存储器中的一个字,则将该存储单元的内容**->IP**，例如：

```
JMP    BX              ;EA = BX, EA→IP
JMP    NLAB[BX]        ;EA = NLAB + BX, (EA)→IP
或 JMP WORD PTR NLAB[BX];EA = NLAB + BX, (EA)→IP, WORD PTR
```


段间转移图示

- 程序段间的直接或间接转移如下图:



段间转移过程

- 1.段间直接寻址(Intersegment Direct Addressing)
- 用指令中直接提供的转移偏移地址EA->IP,指令同时直接提供的转移段地址->CS, 实现从一个代码段转移到另一个代码段。
- 例如： CALL FAR PTR NEXTLAB;

- 2.段间间接寻址(Intersegment Indirect Addressing)
- 根据存储器的数据寻址方式获得EA,再将EA双字单元第一个字的内容作为转移偏移地址->IP,EA双字单元第二个字的内容作为转移段地址->CS,然后CS+IP形成指令的实际转移地址。

例如：

```
JMP  FAR PTR [BX]      ;EA=BX,(EA)→IP,(EA+2)→CS
CALL DWORD PTR [BX]    ;DWORD PTR 说明其后的地址单元是一个双字
```

注意：段内、段间的存储器的数据寻址获得 EA 后，还要加上合适段寄存器的内容，形成存储器的地址。转移地址 PA = CS 与 IP 的新内容之和。程序转移的四种寻址均适用指令 JMP 与 CALL。

为了说明间接转移怎样同一些数据寻址方式结合操作，假设 BX = 1256H，SI = 528FH，DISP = 20A1H，DS 的内容已知，则

用寄存器相对寻址（BX 寄存器），转移偏移地址 IP = (1256H + 20A1H + DS × 2^4)；

用 BX 和 SI 作为基址变址寻址，转移偏移地址 IP = (1256H + 528FH + DS × 2^4)。

- 设（DS）=2000H，（CS）=1000H，（BX）=1256H，（SI）=528EH，TABLE 的EA=20A1H，（232F7H）=3280H，（264E4H）=2450H，（264E6H）=3000H，求以下指令执行后的转移地址。
- （1） JMP BX
- （2） JMP WORD PTR TABLE[BX]
- （3） JMP DWORD PTR [BX][SI]
- 解：（1） JMP BX 执行后（BX）→IP，所以转移地址=1000：1256H
- PA=10000H+1256H=11256H
- （2） 有效地址的EA= TABLE +BX=20A1H+1256H = 32F7H
- PA= 20000H+ 32F7H=232F7H
- JMP WORD PTR TABLE[BX] 执行后，（232F7H）=3280H→IP
- 所以转移地址= 1000 (CS) ：3280H =13280H

- 设（DS）=2000H，（CS）=1000H，（BX）=1256H，（SI）=528EH，TABLE 的EA=20A1H，（232F7H）=3280H，（264E4H）=2450H，（264E6H）=3000H，求以下指令执行后的转移地址。
- （3） JMP DWORD PTR [BX][SI]
- 解： （3） 有效地址的EA= BX+SI = 1256H +0528E
- = 64E4H
- PA= 20000H+ 64E4H = 264E4H
- JMP DWORD PTR [BX][SI] 执行后，
- （264E4H）→IP=2450H，（264E6H）→CS=3000H
- 所以转移地址=3000：2450H =32450H

3.2* 实方式32位指令地址

- 实地址方式32位指令寻址，指在32位的PC机上使用16位的存储机制，执行32位的非保护方式及非虚拟方式的指令，达到直接存取32位寄存器和32位存储器操作数的目的。

3.2.1数据与地址类型

- 32位CPU可以处理的数据类型有字节、字、双字,也可访问字节地址、字地址、双字地。支持带符号或无符号的8位、16位、32位的整数和压缩BCD或非压缩BCD的数。

  实方式的近程指针:16位的段内偏移值,用于段内数据访问或段内转移;远程指针:32位的指针,由一个16位段值和一个16位的偏移值组成,用于段间数据访问或段间转移

  32位的CPU使用32位地址线,其物理存储器最大可达2^{32}字节。但是在实地址方式存储组织方式与16位机类似。

- 实方式地址类型既支持32位和16位的操作数,也支持位和32位的实方式寻址方式。

3.2.2 32位的指令寻址方式

- 偏移地址EA
- EA={基址}+{变址}×{比例因子}+{位移量}
- 基址寄存器可以是任意一个32位通用寄存器。
- 变址寄存器是指除了堆栈指针ESP以外的7个通用寄存器。
- 比例因子是1、2、4、8可以分别用于字节、字、双字、四字的变址。
- 常数位移量可以指8位或32位。

32位的寻址方式图示

$$\left\{\begin{array}{l}\text{无}\\EAX\\ECX\\EBX\\ESP\\EBP\\ESI\\EDI\end{array}\right\}+\left\{\begin{array}{l}\text{无}\\EAX\\ECX\\EDX\\EBX\\\text{无}ESP\\ESI\\EDI\end{array}\right\}*\left\{\begin{array}{l}1\\2\\4\\8\end{array}\right\}+\left\{\begin{array}{l}\text{无}\\8\text{位}\\32\text{位}\end{array}\right\}$$

- 如果使用ESP、EBP为基址寄存器，则SS是默认的段寄存器，也可用CS、DS、ES、FS、GS来替换SS；
- 使用EAX、EBX、ECX、EDX、ESI、EDI为基址寄存器，则DS是默认的段寄存器，同样也能用CS、SS、ES、FS、GS来越段替换DS

32位的指令寻址方式示例：

- MOV AX ,DS:[BP]
- MOV FS:[EBP],ECX；DS、FS分别替换了缺省段SS

32位的指令寻址方式示例（续）

- 为了取得指令代码，只能用CS；PUSH、POP等指令与堆栈有关，也只能用SS。此外，在指令代码32位的程序堆栈操作时，要确保ESP/SP的内容（地址）总是为4的倍数。
- 例如：
- PUSH 12345678H
- ;ESP/SP-4->ESP/SP,12345678->[ESP/SP]

POP EAX

; ([ESP/SP])->EAX, ESP/SP+4-> ESP/SP

3.2.3 实地址32位指令寻址

- 32位的指令寻址包括数据寻址和程序转移寻址。一般寻址的基本概念与16位汇编寻址概念相似，特殊的寻址方式如下叙述：

1、非存储器的数据寻址方式示例

- （1）立即寻址
- MOV EAX,19461201H; 19461201H->EAX
  与16位寻址相似。

（2） 寄存器寻址

- MOV EAX,ECX ;ECX->EAX
  与16位寻址相似

2、 存储器的数据寻址方式示例

- （1）直接寻址
- MOV EAX,[4612H] ;EA=4612,(EA)->EAX
  与16位寻址相似。
- （2）寄存器间接寻址
- MOV [ECX],EDX ;[ECX]间接指示存放操作数EA中，EA=ECX，EDX->EA，与16位寻址相似。

存储器的数据寻址方式示例（续）

- （3）寄存器相对寻址
- MOV ECX,[EAX+24] ； EA=EAX+24,(EA)->ECX

- （4）基址变址寻址
- MOV EAX ,[EBX] [ESI] ; EA=EBX+ESI,(EA)->EAX

- （5）相对基址变址寻址
- SUB EAX,[EBX+ESI+0FF0H]
  ;EA=EBX+ESI+0FF0H,EAX-(EA)->EAX

存储器的数据寻址方式示例（续）

（6）带比例因子的变址

- MOV ECX,[ESI*4]

;EA=ESI×4,(EA)->ECX

（7）基址与带比例因子的变址寻址

- MOV ECX,[EAX][EDX*8]

;EA=EAX+EDX×8,(EA)->ECX

（8）基址与带位移量及比例因子的变址寻址

- MOV EAX,LTAB[EDI*4][EBP+80]

;EA=LTAB+EDI×4+EBP+80,(EA)->EAX

3.程序转移寻址方式

- 有相对EIP的段内直接寻址，段内间接寻址、段间直接寻址和段间间接寻址。
- 4. 前缀代码66H或67H （不看）

3.3 实方式指令系统

- 指令系统是一台机器所有指令的集合。

  Pentium系列机指令系统庞大、类型多样,约有**300**多条指令,其中包括基本指令**100**多条。具有支持多进程、多任务、虚拟存储器和多媒体等功能的**32**位指令。

3.3.1常用指令类型集

- **1.**数据传送类指令
- **2.**算术运算类指令
- **3.**逻辑和移位操作类指令
- **4.**串操作与重复前缀类指令
- **5.**控制转移类指令
- **6.**处理机控制类指令
- **7.**其它指令

P61 注意:

- **OPD表示目的操作数; OPS表示源操作数; (OPS)表示OPS的内容; (OPD)表示OPD的内容; ->表示传送; R表示通用寄存器; Sr表示段寄存器; M表示主存储器;**
- **XXXX:XXXX 表示组合号, ":"表示其前后组成一个数; L表示操作数的长度; d表示立即数; B/W/D表示字节或字或双字。**

3.3.2数据传送类指令

1.传送指令MOV

语句格式：MOV　OPD，OPS

- 功能：将源操作数传送入目的地址，源地址内容不变。即（OPS）→OPD。
- 下图描述了MOV指令在传送数据时允许传送的路径及类型。

注意:

- (1)OPD,OPS必须同时是8/16/32位,否则会产生操作数类型不匹配的错误. 如：MOV AX,BL 是错的
- (2)OPS可以是R/M/Sr/d, OPD只能是R/M/Sr(除CS外).
- . 如：MOV 2000H,AX 是错的
- (3)例外:OPD和OPS不能同时为M,也不能同时为Sr;不能将d->Sr. MOV DS,ES 是错的
- 如：MOV [SI],[BX] 、 MOV DS,2000H 是错的
- (4)指令执行后不影响FLAGS的标志位的状态.

示例:存储器与寄存器间数据传送。

- MOV AL，BH ;
- MOV BX，DI;
- MOV AX，BUF[SI] ;
- MOV AX, 8 ;
- MOV CX，’A8’ ;

MOV DS，AX
MOV EAX，1975
MOV AX，ES
MOV AH，’B’
MOV BUF, CX

2. MOVSX和MOVZX指令

- 格式:MOVSX OPD, OPS

  MOVZX OPD, OPS

- 它们将OPS的内容->OPD,但是对于OPD左边空缺的位, MOVSX全部用OPS的符号填充(作符号延伸), 可以对有符号的数进行符号扩展;而MOVZX是全部以零(0)填充,可对无符号数进行0扩展.

例3.2

- MOV CL, 88H
  MOVZX AX, CL ;AX = 0088H
  MOVSX BX, CL ;BX = FF88H
- 另外也可如下书写指令:
- MOVSX CX, BL MOVSX EAX, BUF
- MOVZX AX, CL MOVSX EBX, AL
- MOVZX ESI, BUF MOVZX EDX,DI

3. 数据交换指令XCHG

- 语句格式：**XCHG OPD，OPS**
- 功能：将源地址与目的地址中的内容互换。即（**OPD**）→**OPS**，（**OPS**）→**OPD**。
- 寄存器与存储器之间数据交换。
- 例**3.3** 设 **ECX=39A5F034H，EDX=B218CD52H**，执行 **XCHG ECX，EDX**

  ；EDX=39A5F034H，ECX=B218CD52H

4. 堆栈操作指令

（1）进栈指令PUSH

- 语句格式：

```
PUSH  OPS  ; W/D, R/Sr/M/d
```

- 功能：将寄存器、段寄存器、立即数或存储器中的一个字数据压入堆栈顶部，指令视操作数长度为字(2字节)或双字(4字节)和地址为16位/32位,先将SP/ESP -2/4->SP/ESP,后将OPS->[SP]/[ESP]。

（2） 出栈指令POP

- 语句格式： POP OPD ；W/D, R/Sr/M/d
- 功能： 视OPD长度为字或双字,先将当前SP/ESP指向的内容->OPD,后将SP/ESP+2/4->SP/ESP 。
- 从POP指令功能可看出，该指令为PUSH指令的逆过程,一般成对使用。

- 例 3.4 SP/ESP=100H,执行下列指令后SP/ESP=?, EAX=?
- PUSH AX
- PUSH 20020418H
- POP EAX ；EAX=20020418H
- POP BX ；BX=原AX ； SP/ESP=100H

5. PUSHA/PUSHAD 和 POPA/POPAD 指令

```
格式:PUSHA   ;压入 8 个字通用寄存器。临时单元为 TEMP,块的大小 =2 ×8 =16
             ;字节
             ;TEMP = SP/ESP,SP/ESP - 16→SP/ESP
             ;依次将 AX,CX,DX,BX,TEMP,BP,SI,DI→[SP/ESP]
       POPA  ;弹出到 8 个 16 位通用寄存器
             ;([SP])/([ESP])依次→DI,SI,BP,TEMP,BX,DX,CX,AX(SP/
             ;ESP 除外)SP/ESP + 16→SP/ESP
    PUSHAD   ;压入 8 个双字通用寄存器。块的大小 =4 ×8 =32 字节
             ;TEMP = SP/ESP,SP/ESP - 32→SP/ESP
             ;依次将 EAX,ECX,EDX,EBX,TEMP,EBP,ESI,EDI→[SP]/[ESP]
     POPAD   ;弹出到 8 个 32 位通用寄存器
             ;([SP]/[ESP])依次→EDI,ESI,EBP,TEMP,EBX,EDX,ECX,EAX
             ;(SP/ESP 除外)SP/ESP +4 ×8→SP/ESP
```

PUSHA 和 POPA、PUSHAD 和 POPAD 常成对使用,操作不影响标志位。

6. 地址传送指令

- （1）传送偏移地址指令 LEA
- 语句格式：

```
LEA   OPD，OPS  ;OPS的EA->OPD
                ;OPD是16/32位的R,OPS或M
```

- 功能：主存按源地址的寻址方式计算偏移地址，将偏移地址送入指定寄存器。

- 倒3．5　设BUF的偏移地址为120H，BX=0A00H，S1=0010H，则执行指令：
- LEA DI, BUF ；EA=BUF的偏移地址=120H→DI
- LEA DX，[BX][SI]　；EA=BX+SI=0A00H+0010H=0A1OH→DX
-
- 例3．6 若EBX=00000034H，ESI=00000052H，DOLLAR=08H，执行指令：
- LEA ECX，[EBX][4*ESI]DOLLAR
- 则： ECX =EA = 34H+52H×4+8H=34H十148H+8=184H

7．LDS/LES/LFS/LGS/LSS指令

- 语句格式：

LDS/LES/LFS/LGS/LSS OPD，OPS ;EA = OPS

- 功能：地址传送LDS等指令中,OPD为R(16位),OPS为M(32位).操作对FLAGS无影响,执行后结果为:

(EA)->R　;取偏移地址

(EA+2)-> DS/ES/FS/GS/SS ；　取段地址

- 例3．7 设变量BUF的段地址=0CA0H，偏移地址=0100H
- BUF DW 1，2，3，4 ；定义4个字操作数
- D32_BUF DD BUF ；
- LDS SI，D32_BUF ；
- EA=0108H，
  (EA)=(0108H)=0100H→SI
- (EA+2)=0CA0H→DS

BUF（0100）	01
0101h	00
0102h	02
0103h	00
0104h	03
0105h	00
0106h	04
0107h	00
D32_BUF 0108H	00
	01
	A0
	0C

- 8.LAHF和SAHF指令
  格式:LAHF ;取标志, FLAGS低8位依次→AH
  SAHF 存标志,AH依次→ FLAGS低8位

  例3.8
  LAHF FLAGS低8位→AH
  OR AH,01000000; 逻辑或,将对应ZF的AH位置“1
  SAHF ; ZF=1,其余标志不变

- 9. PUSHF/POPF指令
  格式: PUSHF; FLAGS→堆栈
  POPF; 从当前堆栈→ FLAGS
  指令直接对16位标志寄存器 FLAGS操作，操作过程分别类似于PUSH和POP。
  10. PUSHFD/ POPFD指令
  格式: PUSHFD; SP/ESP-4→SP/ESP,32位 EFLAGS→SP/ESP
  POPFD;([SP/ESP])→ EFLAGS,SP/ESP+4→SP/ESP
  PUSHFD与 POPFD操作相反,它们都对堆栈进行双字操作。

3.3.3 逻辑运算和位操作类指令

- 1．求反指令NOT
- 2．逻辑乘指令AND
- 3．测试指令TEST
- 4．逻辑加指令OR
- 5．按位加指令XOR

1．求反指令NOT

- 语句格式：NOT　OPD
- 功能：将目的地址中的内容逐位取反后送入目的地址。

即（OPD）求反→OPD

【例】逻辑非运算。

```
MOV    AX，878AH；（AX）=878AH
NOT    AX          ；（AX）=7875H
```

2．逻辑乘指令AND

➢语句格式：AND　OPD，　OPS

➢功能：将目的操作数和源操作数进行逻辑乘运算，结果存目的地址。

即（OPD）∧（OPS）→OPD。

该指令用于清除目的操作数中与源操作数置0的对应位。说明：逻辑乘的运算法则为：1∧1=1，1∧0=0，0∧1=0，0∧0=0

【例】将AL中第3位和第7位清零。

```
MOV     AL，0FFH        MOV     AX，0F0F0H
AND     AL，77H         AND     AX,1234H
;AL=77H                 ;AX=1030H
```


3．测试指令TEST

➢语句格式：TEST　OPD，OPS

功能：源地址和目的地址的内容执行按位的逻辑乘运算，结果不送入目的地址。

即（OPD）∧（OPS）。

【例】测试AX中的第12位是否为0，为0则转L。

```
TEST    AX，1000H
JZ      L
```

测试AX中的低4位是否为0，不为0则转L。

```
TEST    AX，000FH
JNZ     L
```

4. 逻辑加指令OR

➢语句格式：OR　OPD，OPS

功能：将目的操作数和源操作数进行逻辑加运算，结果存目的地址。

即（OPD）∨（OPS）→OPD。

说明：逻辑加的运算法则为：1∨1=1，1∨0=1，0∨1=1，0∨0=0。

【例】将AL寄存器中第3位和第7位置1。

```
MOV    AL，0
OR     AL，88H      ;AL=88H
```


5. 按位加指令XOR

➢语句格式：　XOR　OPD，OPS

功能：目的操作数与源操作数做按位加运算，结果送入目的地址。

即（OPD）⊕(OPS) →OPD。

说明：按位加的运算法则为；1⊕1=0，1⊕0=1，0⊕1=1，0⊕0=0。

【例】按位加运算。

```
MOV    AL，45H      ；（AL）=45H
XOR    AL，31H      ；（AL）=74H
```

XOR指令可以将某些位求反

【例】将AL寄存器中低4位求反。

```
MOV    AL，05H
XOR    AL，0FH    ;AL=0AH

XOR AX，AX   ;    AX=0，CF=0,ZF=1
```


例 3.9

```
AND   AX,0FFF8H    ;AX∧0FFF8H→AX,将 AX 低 3 位置 0,高 13 位不变
OR    BX,3         ;BX∨03H→BX,BX 低 2 位置 1,其余位保持不变
XOR   EAX,EAX      ;0→EAX
NOT   M_BYTE       ;若(M_BYTE)=7FFFH,则指令执行后(M_BYTE)=8000H
```

例 3.10 测试 AL 中的位 7 和位 1 均为 1，转标号 EXIT。

```
MOV   AL,11100111B    ;B 表示二进制数的后缀
NOT   AL              ;将要测试的操作数求反,AL=00011000B
TEST  AL,10000010B    ;AL∧10000010B,结果为 0,ZF=1
JZ    EXIT            ;ZF=1,被测位均 1,转向 EXIT
```

2。一般移位指令

➢移位指令包括算术移位指令、逻辑移位指令和循环移位指令，分别进行左移和右移操作。这些指令均有统一的语句格式：

```
SAL/SHL OPD, OPS    ;左移
SAR     OPD, OPS    ;算术右移
SHR     OPD, OPS    ;逻辑右移
```

其中,OPD是8/16/32位的R/M;OPS是计数值(即移位次数),有三种情形:OPS即可是8位的立即数或CL的内容(移位前次数->CL)。

其功能为将目的操作数的所有位按操作符规定的方式移动1位或按寄存器CL规定的次数（0~255）移动，结果送入目的地址。目的操作数是8位（或16位）的寄存器数据或存储器数据。

移位指令

- SAL ；算术左移
- SAR ；算术右移
- SHL ；逻辑左移
- SHR ；逻辑右移

（1）算术左移和逻辑左移指令SAL（SHL）

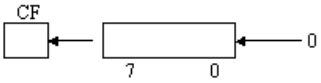
语句格式：SAL　OPD，1　或 SHL　OPD，1
　　SAL　OPD，CL 或 SHL　OPD，CL

功能：将（OPD）向左移动CL指定的次数，最低位补入相应的 0，CF的内容为最后移入位的值。

左移1位相当于乘以2

如：

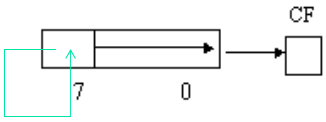
```
MOV　CL,4
SHL　AL,CL
```



（2）算术右移指令SAR

语句格式：SAR　OPD，1或SAR　OPD，CL

- CF功能：将（OPD）向右移动CL指定的次数且最高位保持不变；CF的内容为最后移入位的值。



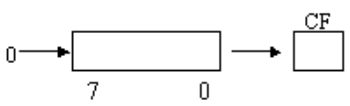
【例】算术右移运算

```
MOV   BH，0F4H    ；（BH）=0F4H        11110100
MOV   CL，  2     ；（CL）=2           11111101   0
SAR   BH，  CL    ；（BH）=0FDH，（CF）=0
```

该例语句“SAR BH，CL”实际上完成了（BH）/ 4 →BH的运算，所以，用SAR指令可以实现对有符号数除2^n的运算（n为移位次数）。

（3）逻辑右移指令SHR

- 语句格式：SHR OPD，1或SHR OPD，CL
- 功能：将（OPD）向右移动CL规定的次数，最高位补入相应个数的 0，CF的内容为最后移入位的值。
- 用SHR指令可以实现对无符号数除2^n的运算（n为移位次数）



【例】逻辑右移运算

```
MOV   BH，0F4H    ；（BH）=0F4H        11110100
MOV   CL，  2     ；（CL）=2           00111101   0
SHR   BH，  CL    ；（BH）=3DH，（CF）=0
```

该例语句“SHR BH，CL”实际上完成了（BH）/ 4 →BH的运算，所以，用SHR指令可以实现对无符号数除2^n的运算（n为移位次数）。

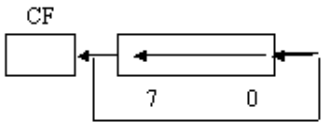

```
SAL  AX,1          ;设 AX 初值=8701H,移 1 位后,AX=0E02H,CF=1
SHR  M_DWORD,CL    ;设 M_DWORD 单元初值=8701H,CL=2,CF=1
                   ;移位后,M_DWORD 单元的值=21C0H,CF=0,CL=2
SHL  ECX,5         ;注意:当计数值≥2 时,实方式下才可以使用。
SAR  B_BYTE[BX][SI],2
```

- AX = 1000 0111 0000 0001
- AX= 000 0111 0000 00010 =0E02H, CF =1
- M_DWORD = 8701H= 1000 0111 0000 0001
- M_DWORD = 00 1000 0111 0000 00 =21C0H, CF=0

3. 循环移位指令

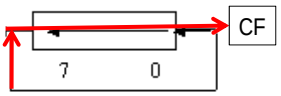
（1）循环左移指令ROL

- 语句格式：ROL　OPD，1或ROL　LPD，CL
- 功能：将目的操作数的最高位与最低位连成一个环，将环中的所有位一起向左移动CL规定的次数。CF的内容为最后移入位的值。



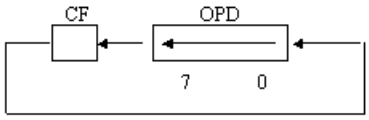
（2）循环右移指令ROR

- 语句格式：　ROR　OPD，1或ROR　OPD，CL
- 功能：将目的操作数的最高位与最低位连成一个环，将环中的所有位一起向右移动CL规定的次数，CF的内容为最后移入位的值。



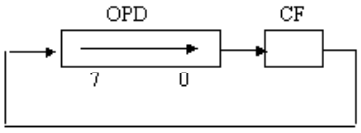
（3）带进位的循环左移指令RCL

- 语句格式： RCL OPD， 1 或 RCL OPD，CL
- 功能：将目的操作数连同CF标志一起向左循环移动CL规定的次数。



（4）带进位的循环右移指令RCR

- 语句格式：RCR OPD，1 或 RCR OD，CL
- 功能：将目的操作数连同CF标志一起向右循环移动所规定的次数。



循环移位指令

- L ；闭循环左移
- O
- R ；闭循环右移
- R
- L ；带进位循环左移
- C
- R ；带进位循环右移

示例：

```
ROL  AL,1
ROR  EBX,12
RCR  BX,CL
```

例 3.11 将存储单元中 4 个字节的内容连续左移一位。

```
A_WORD  DW      0FC85H
B_WORD  DW      4321H
        SHL     A_WORD,1        ;A_WORD 单元内容 = 0F90AH,CF = 1
        RCL     B_WORD,1        ;B_WORD 单元内容 = 8643H,CF = 0
```

- 1111 1100 1000 0101
- 111 1100 1000 0101 0 CF=1
- 1 0100 0011 0010 0001 0
- 0 100 0011 0010 0001 1

p68

4. 双精度移位指令

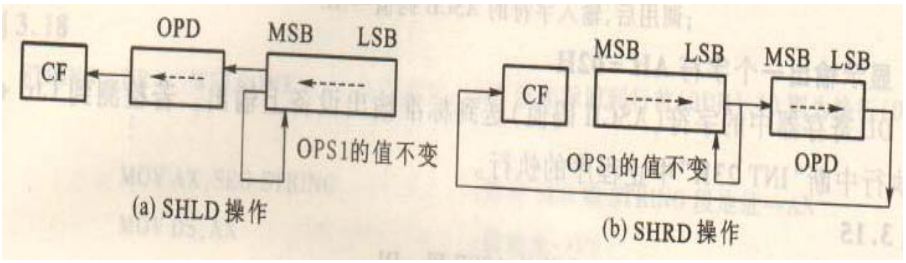
格式：SHLD OPD,OPS1,OPS2 ;双精度左移

SHRD OPD,OPS1,OPS2 ;双精度右移

双精度是指16位操作数长度下的32位精度，或者32位操作数长度下的64位精度。SHLD/SHRD使一串指定的二进制位OPS1移到OPD。其中：

OPD是R/M(16/32位)，OPS1是R，OPD和OPS1的长度应一致；

OPS2可以是8位立即数或CL的内容(移位前次数→CL)，移位后CL值不变。



例3.12

```
CLC                 ;CF=0
MOV   AX,0A000H
MOV   CX,8006H
SHLD  AX,CX,4       ;CF=0,AX=0008H,CX不变
```

例3.13

```
DBUF  DD    19461201H
      MOV   EAX,19471004H
      MOV   CL,4
      CLC
      SHRD  DBUF,EAX,CL  ;CF=0,EAX和CL不变,(DBUF)=41946120H
```

- （1）下列程序段执行后，BX寄存器中的内容是什么?
- MOV CL，3
- MOV BX，0B7H
- ROL BX，1
- ROR BX，CL ；C02DH
- （2）试写出执行下列指令后，BX寄存器的内容。
- MOV CL，7
- MOV BX，8D16H
- SHR BX，CL ；011AH

3.4 字符设备I/O功能调用

本节主要介绍DOS子程序调用方式、键盘输入、屏幕显示和打印机输出的DOS系统功能调用。

3.4.1 DOS子程序调用方式

- 调用之前：设置子程序的入口参数。
- 调用请求：执行“INT 21H”软中断指令调用。
- 调用之后：可能有出口参数，也可能无出口参数。如果有出口参数，可根据程序需要，判断本次调用是否成功或者分析执行情况。

1 从键盘输入一个字符AH=01H

从键盘读入一个字符，送到显示器输出，并将该字符的ASCII码值－>AL（出口参数）；如果检测到读入的字符是Ctrl＋Break，则中止程序执行。

例3.14：

- MOV AH,1
- INT 21H
- 等待键入一个字符调用后，输入字符的ASCII码值->AL

2、显示输出一个字符AH=02H

- 将DL寄存器重的字符（ASCII码值）送到标准输出设备上输出。若检测到Ctrl＋Break间，则执行中断“INT 23H”中止程序的执行。
- 例3.15

```
MOV DL,AL ;AL内容是字符的ASCII码->DL
MOV AH,2   ;功能号2->AH
INT 21H    ;调用显示一个字符
```

3.打印机输出一个字符AH=05H

- 将要打印的字符－>DL,然后调用就可以打印输出。如果有标准打印设备不输出，可在执行时用DOS命令键Ctrl＋Break进行帮助。
- 例3.16

```
MOV DL,AL ;AL的字符ASCII码->DL
MOV AH,5   ;功能号5->AH
INT 21H    ;调用打印一个字符
```


4、直接控制台输入AH=07H

- 输入时不回显，例如，从键盘输入一个字符，在屏幕上不显示，出口参数AL＝键入的字符。可用来设置保密口令。
- 例3.17

```
PASSWORD DB 10DUP(0)
        ...
        MOV CX,6
        MOV SI,0
AGAIN:  MOV AH,7
        INT 21H
        MOV  PASSWORD[SI],AL
        MOV DL,'*'
        MOV AH,2
        INT 21H    ; 这三条在书上加一下
        INC  SI
        LOOP AGAIN
```

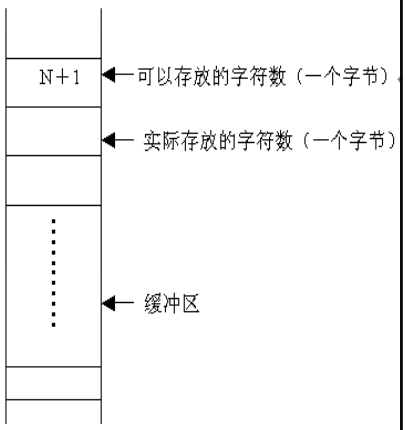
3.4.3 多字符输入显示输出

- 1、显示输出字符串AH=09H
- 例3.18 执行下语句后屏幕显示输出“WELCOME!”

```
STRING  DB 'WELCOME!',13,10,'$'
        …
        MOV AX,SEG STRING
        MOV DS,AX
        LEA DX,STRING
        MOV AH,9
        INT 21H
```


2.缓冲区键盘输入AH=0AH

- 如果需要键入最大的字符数位N，则由“DS：DX”指向的缓冲区可分为3个字段。
- 第一个字段：定义缓冲区可存放的字符数N+1，使机器自动控制检查，若键入实际的字符数个数超过N，则响铃报警。
- 第二个字段：定义缓冲区，当键盘输入调用退出后，系统自动计数并存放实际输入的字符。
- 第三个字段：定义可存放字符的缓冲区，调用退出后，存放实际输入的字符，最后一个字节总是回车符。



0AH调用示例

- 例3.19　编写在实地址方式（.586与USE16）下，定义一个能最多可输入20个字符到BUF缓冲区的程序。

```
.586
STACK    SEGMENT  USE16              ；定义16位段
         DB       256       DUP(0)
STACK    ENDS
DATAS    SEGMENT  USE16
N        EQU          20
BUF      DB     N+1         ；定义第一个字段，BUF位21
COUNT    DB   0                    ；定义第二个字段
CHAR     DB   N+1 DUP(0)           ；定义第三个字段
PROMPT   DB     'Please input:',13,10, '$'
DATAS    ENDS
```


0AH调用示例（续）

```
CODES   SEGMENT  USE16
        ASSUME  CS:CODES,DS:DATAS
START:    MOV    AX,DATAS
          MOV    DS,AX
          LEA  DX,prompt        ;取prompt的偏移地址－>dx
          MOV   AH,9
          INT    21H
          LEA    DX,BUF          ;取BUF的偏移地址－>dx
          MOV   AH,10
          INT    21H
          MOV  AH,4CH
          INT 21H
CODES    ENDS
         END  START
```

上机题：

- 首先显示提示信息“Please Input：”，然后从键盘上输入10个字符，存入keybuf缓冲区，再用9号DOS调用在屏幕上显示出来并在DEBUG下查该缓冲区的字符。

习题： 3.13、3.14、3.15、3.16

3.17 3.18

本章结束