第三章 8086/8088的寻址方式和指令系统

主讲：乔瑞萍

西安交通大学
信息与通信工程学院

学习要点

- 数据寻址的8种寻址方式，I/O端口寻址的2种寻址方式
- 程序转移地址的4种寻址方式
- 掌握8086指令（操作码助记符，可以使用的寻址方式）
- 了解80386新增寻址方式和指令
- 了解ARM64和x86寻址与指令系统的差异

3.1 计算机指令格式

1、汇编指令格式：

OP.C　　OP.D

（Operating Code、Operating Data）

注：OP.C指操作码助记符

OP.D指操作数部分，又称地址码。

（实际上的OP.C、OP.D都应该是二进制数，即机器码。）

例：MOV AH，02H；　B502H

2、根据OP.D中地址的个数，指令可分为：

1）零地址指令：只有OP.C，没有OP.D。NOP

2）一地址指令：单OP.D。INC CL

3）二地址指令：双OP.D。ADD AH,02H

4）三地址指令：三OP.D。SHLD AX,BX,7

指令操作数来源和去向:

- CPU内部的通用寄存器
- 输入输出设备（接口）的一个寄存器
- 主存储器的一个存储单元

3. 指令长度与字长的关系

指令长度主要取决于
- OP.C的长度
- OP.D地址的长度
- OP.D地址的个数

- 几个概念：
- 1)指令的长度指其机器码的长度，是字节的简单倍数；
- 2)现代计算机广泛采用变字长指令格式：并非所有指令的长度一致。
- 3)指令长度与其执行时间没有必然的联系，短指令也可能执行时间长。
- 4)指令字长与内存的编址单位及CPU的机器字长有关。

3.3 指令格式

1、80x86指令编码格式（P81~83）

指令码是指每条指令所对应的二进制编码，即机器码，这里只是要求大家了解一下汇编指令如何翻译成机器码的，即了解编译程序的工作,自学，不做要求。

3.2 80X86的寻址方式（addressing modes）

- 指令语句：由操作码和操作数两部分构成，P70
  - 操作码：表示计算机执行什么操作；
  - 操作数：可能指明了参与操作的数本身，或规定了操作数的**地址**。

寻址方式

清華大學 Tsinghua University

寻址方式（又称编址方式）指的是确定本条指令的操作数地址及下一条要执行的指令地址的方法。

不同的计算机系统，使用数目和功能不同的寻址方式，其实现的复杂程度和运行性能各不相同。有的计算机寻址方式较少，而有些计算机采用多种寻址方式。

通常需要在指令中为每一个操作数专设一个地址字段，用来表示数据的来源或去向的地址。在指令中给出的操作数（或指令）的地址被称为形式地址，使用形式地址信息并按一定规则计算出来或读操作得到的一个数值才是数据（或指令）的实际地址。在指令的操作数地址字段，可能要指出：

① 运算器中的累加器的编号或专用寄存器名称（编号）
② 输入/输出指令中用到的 I/O 设备的入出端口地址
③ 内存储器的一个存储单元（或一 I/O设备 ）的地址

有多种 基本寻址方式 和某些 复合寻址方式 ，简介如下。

8086/8088的寻址方式分为两类:

数据寻址方式

转移地址寻址方式

1）数据寻址方式（8种）

（以通用传送指令MOV　AX，SRC为例）

定义：指令中用以说明或形成操作数有效地址（Effective Adress）的方法，称为操作数的寻址方式。EA就是操作数的偏移地址。

MOV　DST，SRC

SRC：源操作数，指令执行过程中保持原值不变

P70

DST：目的操作数，操作数原值不保留，而将操作结果保留。

- 指令中的内容，包括指令操作码（指出指令完成的运算处理功能和数据类型）和 操作数或指令的地址（指明用到的数据或地址）两部分。
- 每一条指令必须指明它需要完成的功能，通常用几位指令操作码表示；还需要指明用到的数据、地址或设备，通常在地址字段给出，可能是:

(1) 寄存器编号， (2) 设备端口地址，

(3) 存储器的单元地址 (4) 数值等几种信息。

微机中OP.D存放位置有3种P70：

① OP.D包含在指令码中

（实际上存储在代码段中）

② OP.D存放在CPU的某个内部reg中

③ OP.D在内存的数据区中。

数据寻址方式

- 8086/8088中的数据寻址方式有8种：
  - 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；
  - 相对的基址加变址寻址； 隐含寻址。

返回

1、立即数寻址（Immediate addressing）

特点：所需的一个操作数在指令的地址字段部分直接给出。

例如：

MOV　　AX，1234H；

助记符　　立即数

- 适用于操作数固定的情况，取指同时取得操作数，指令执行阶段不必到存储器中取操作数，提高了指令执行速度；

注意：立即寻址方式只能用于源操作数字段，不能用于目的操作数字段。

3、直接寻址（Direct addressing）

特点：在指令的地址码字段，直接给出所需的操作数(或指令) 在存储器中的地址。

方法：先求操作数的物理地址，尔后再访问M取得操作数。

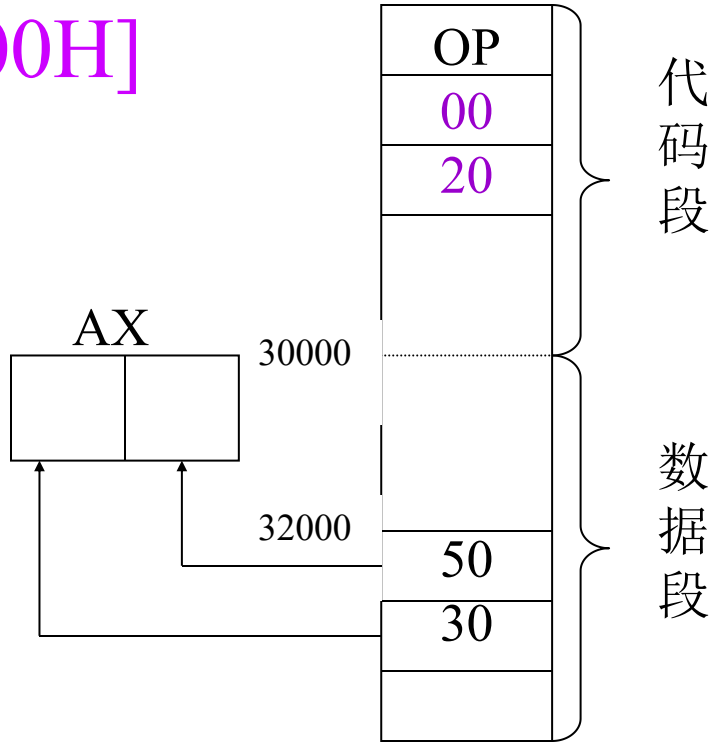
物理地址＝（DS ）×10H+ EA

- 直接寻址中的地址通常是标号

例如：

直接寻址例题

- MOV　AX，DS：[2000H]
- 如果（DS）=3000H，



执行结果为：3050H

例：Addr = 2000H ，是一个操作数的地址，若[2000H] =3050，则用2000H作地址，从内存储器单元中读出的操作数就是 3050H。

返回

2、寄存器寻址（Register addressing）

计算机的 CPU中设置有一定数量的通用寄存器，用于存放操作数、操作数地址或中间结果。

寄存器寻址: 操作数存放在指令规定的寄存器中.

说明：可用于寄存器寻址的寄存器为所有通用寄存器。

8位reg：AH～DH、AL～DL（数据寄存器）

16位reg：AX～DX（数据寄存器）

SI、DI、 SP、BP（指示器组）

- 执行时不需使用总线周期，因而可取得较高运算速度。

寄存器寻址举例

- 例如：

  MOV AX，BX；（AX）←——（BX）

- 若指令执行前：

  （AX）＝3064，（BX）＝1234，

  则执行后，

  （AX）＝1234，（BX）不变。

返回

4、寄存器间接寻址

（Register indirect addressing）

特点：寄存器中存放的是操作数在内存储器中所在单元的地址，操作数的EA在指令码指定的基址（BX）/变址（DI、SI）寄存器中，操作数本身在存储单元中。

有效地址可表示为：

$$EA=\begin{cases}(BX)\\(SI)\\(DI)\end{cases}$$

物理地址＝（DS）×10H+EA

指令中也可指定段跨越前缀来取得其他段中的数据。如：MOV　　AX，ES：[BX]

寄存器间接寻址举例

例：MOV　AX，[BX]

若（DS）=2000H，（BX）=1000H

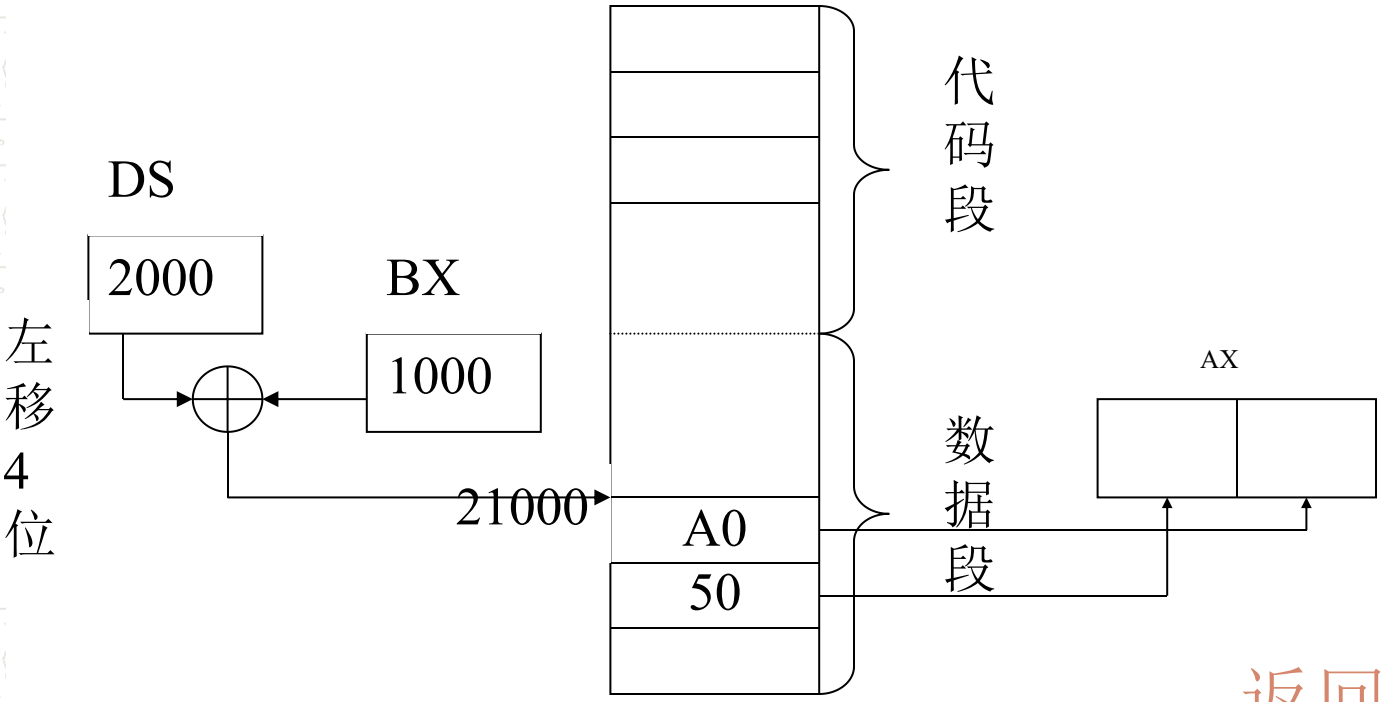
则物理地址=20000+1000=21000H

（AX）=50A0H（示意图在下页）

用途：这种寻址方式可以用于表格处理。执行完一条指令后，只需修改寄存器内容就可取出表格中的下一项。

寄存器间接寻址示意图

- 示意图：



返回

说明：

①当用BX、SI、BI作为间接寄存器时，默认的段寄存器是DS；

当用BP作为间接寄存器时，默认的段寄存器是SS。

②指令中也可指定段跨越前缀来取得其他段中的数据。

如：MOV AX，ES：[BX]

- 假设BX=1000H,MOV AX, BX;　MOV AX, [BX];
- 对寄存器寻址，操作数就是寄存器中的数值1000H
- 对寄存器间接寻址，从内存1000H单元读出来的数才是操作数

5、寄存器相对寻址

（Register relative addressing）

特点：操作数有效地址EA为8/16disp（相对寻址偏移量）与基址/变址reg内容之和，且这两部分在指令码中给出指明。操作数本身在存储单元中。

EA= （BX）
（BP）
（SI）
（DI） + [8bits位移量]

在代码段给出且为常量

在代码段指明

方法：物理地址=10H×（DS）+ {（BX）
（SI）
（DI）} + 8/16disp

物理地址=10H×（SS）+ （BP） + 8/16disp

在指令码中指明：DS：（BX） （SI） （DI）

SS：（BP）

16bit符号地址，即相对偏移量

例：MOV AX，COUNT[SI]

MOV AX，[COUNT+SI]

等效

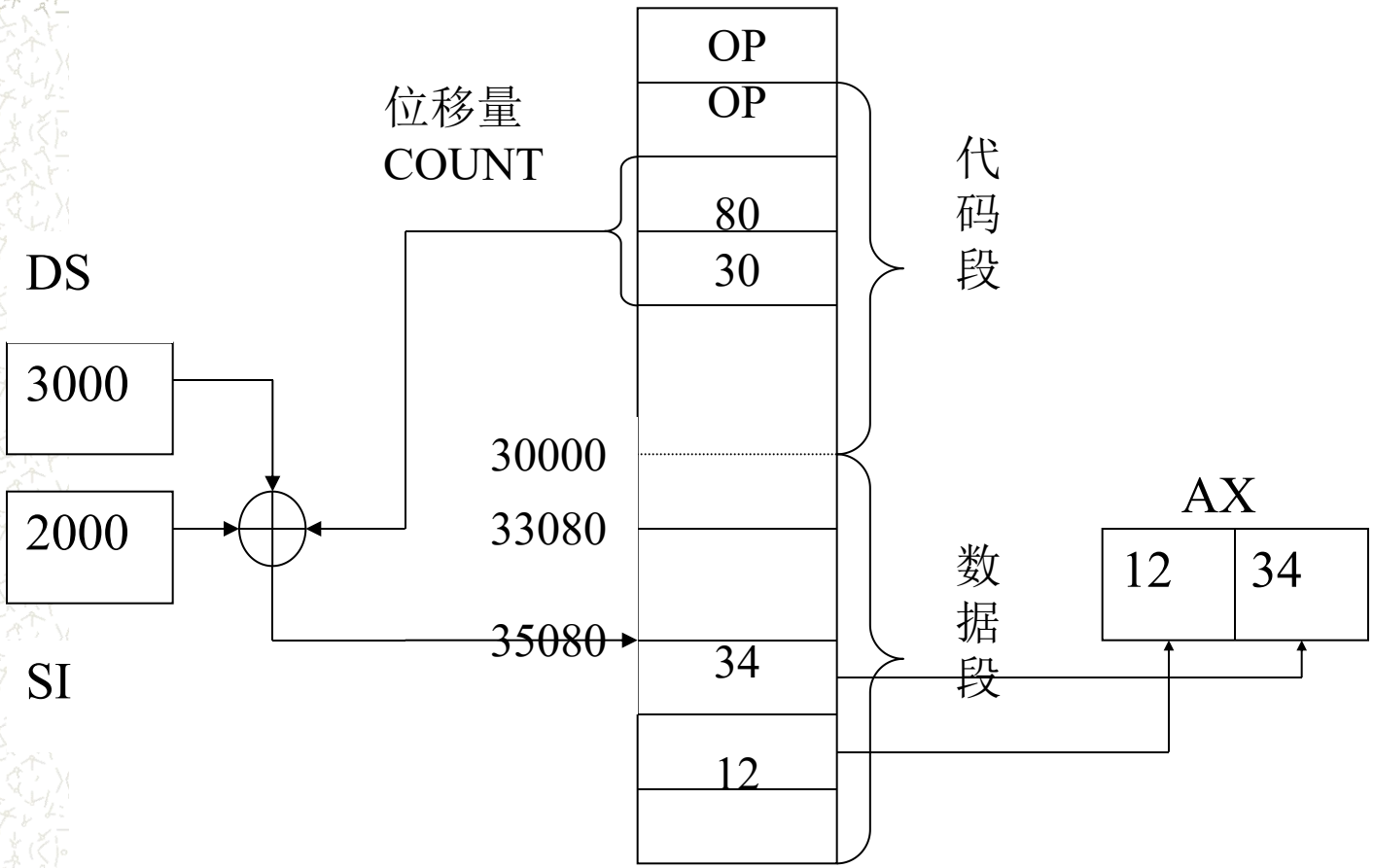
如果（DS）=3000H，COUNT=3080H

则 物理地址

=10H×（DS）+（SI）+（COUNT）

=30000+2000+3080=35080H

执行结果（AX）=1234H，如图


OP
OP
位移量
COUNT
80
30
代
码
段
DS
3000
2000
30000
33080
35080
SI
34
12
AX
12
34
数
据
段

说明：

① 这种方式也可用于表格处理，表格首址为COUNT，利用修改基址或变址reg的内容来取得表格中的值。便于读写数组中的元素。

② 也可用段跨越前缀

MOV DL，ES：STRING[SI]

③ 间接寻址寄存器与默认的段寄存器关系同上

6、基址变址寻址方式

（Based indexed addressing）P74

特点：操作数的EA为基址与变址reg的内容之和，操作数本身在存储单元中。

$$EA=\begin{bmatrix}(BX)\\(BP)\end{bmatrix}+\begin{bmatrix}(SI)\\(DI)\end{bmatrix}$$

方法：

物理地址=（DS）×10H+（BX）+（SI）或（DI）

物理地址=（SS）×10H+（BP）+（SI）或（DI）

例： MOV AX，[BX][DI]

MOV AX，[BX+DI]

等效

如果（DS）=2100H

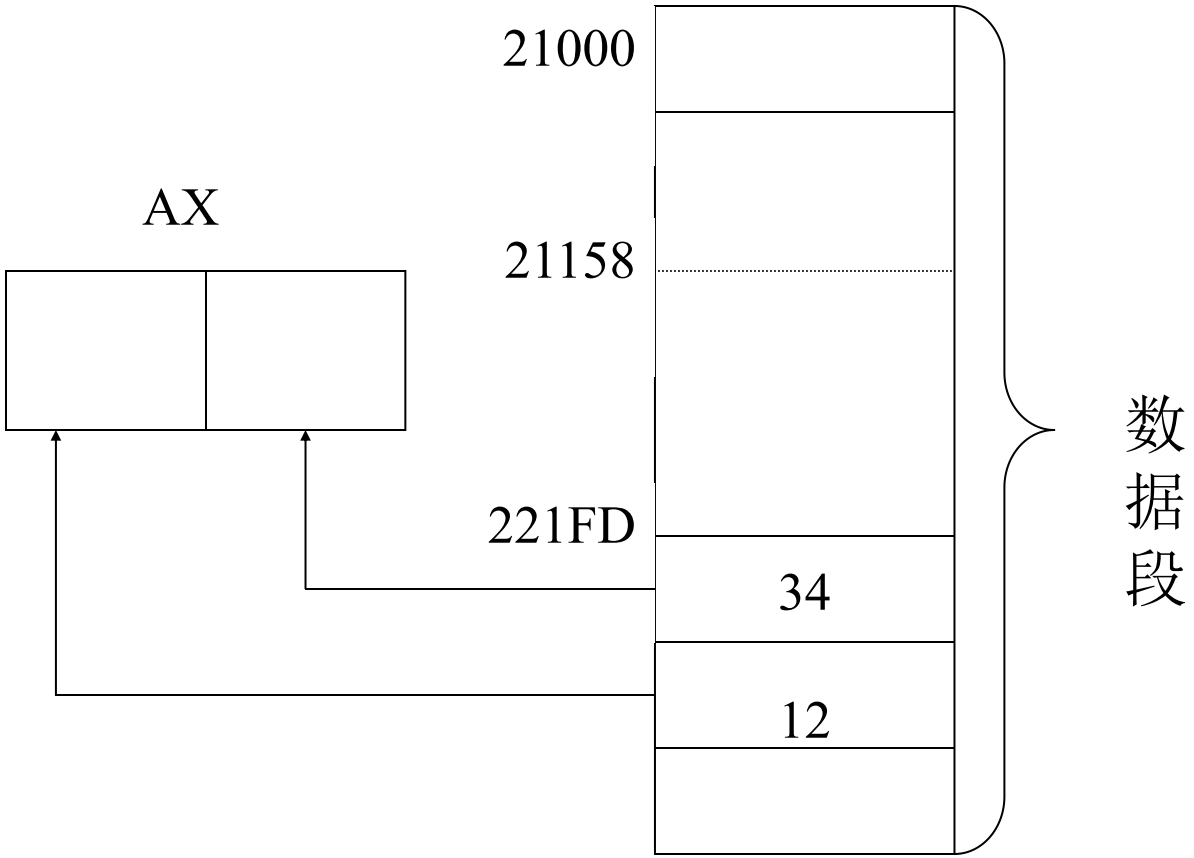
（BX）=0158H

（DI）=10A5H

则EA=0158+10A5=11FDH

物理地址=21000+11FD=221FDH

执行结果（AX）=1234H


21000
AX
21158
221FD
34
12
数据段

说明：

1、适用于数组或表格处理，首地址放基址reg中，变址reg访数组中各元素。因两reg都可修改，它比直接变址方式更灵活。

2、MOV　AX，ES：[BX][SI]

3、不能两个寄存器均为基址寄存器，也不能两个寄存器均为变址寄存器。

7、相对基址变址寻址方式

（Relative based indexed addressing）P75

特点：操作数在存储单元中，EA按下式形成并在指令中指明给出。

$$EA=\begin{Bmatrix}(BX)\\(BP)\end{Bmatrix}+\begin{Bmatrix}SI)\\DI)\end{Bmatrix}+8位disp/16disp$$

方法：

物理地址=

（DS）×10H+（BX）+（SI）或（DI）+8/16位disp

物理地址=

（SS）×10H+（BP）+（SI）或（DI）+8/16位disp

例：MOV　AX，COUNT[BX+DI]

MOV　AX，[COUNT+BX+DI]

MOV　AX，0250[BX][DI]

5[BX][SI]

1234 [BX][SI]

如果：（DS）=3000H，（BX）=2000H，

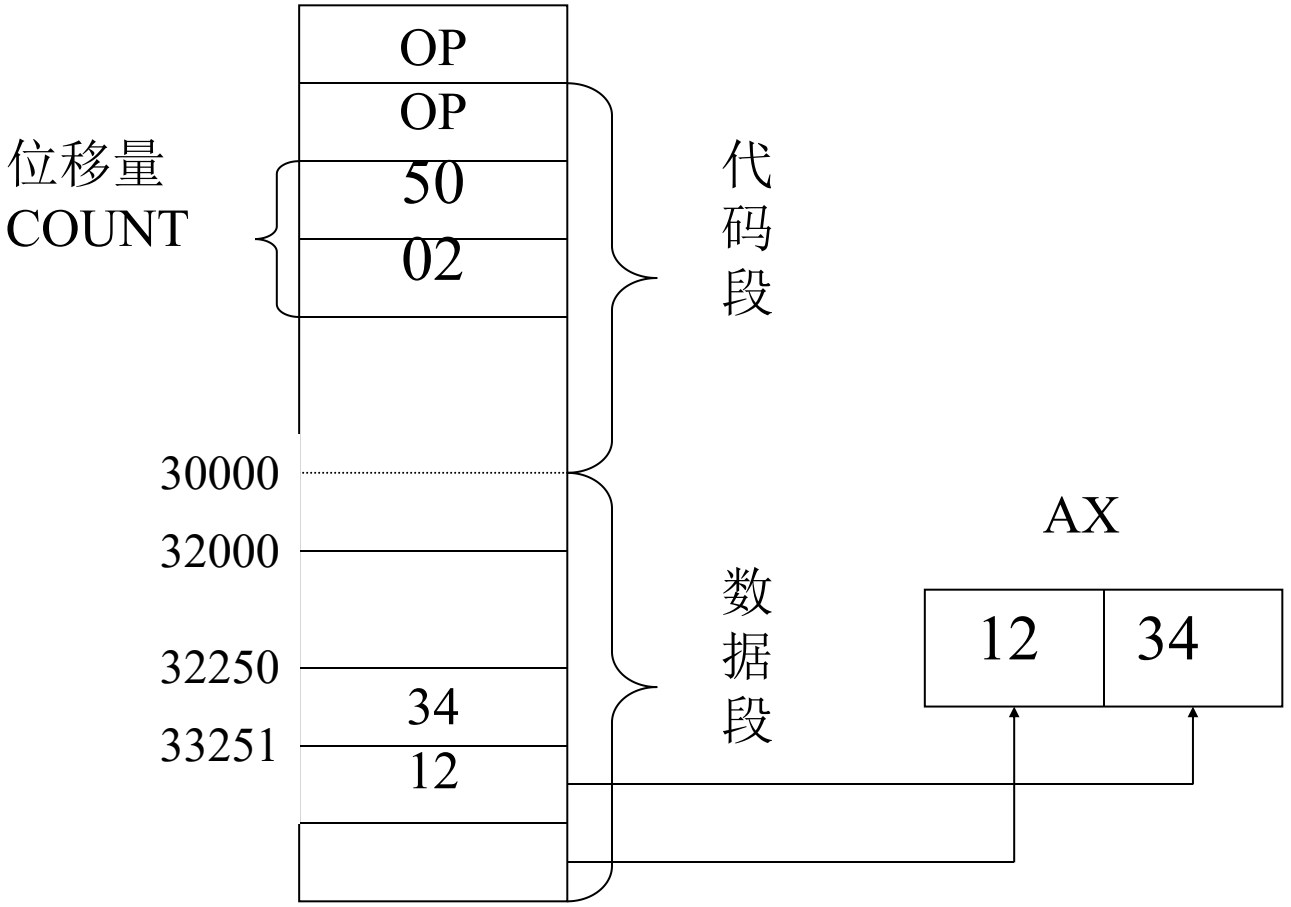
（SI）=1000H，COUNT=0250H

物理地址

=10H×（DS）+（BX）+（SI）+COUNT

=30000+2000+1000+0250=33250

执行结果（AX）=1234H


OP
OP
50
02
位移量
COUNT
代
码
段
30000
32000
32250
33251
34
12
数
据
段
AX
12
34

说明：这种寻址方式为堆栈处理提供了方便，一般（BP）可指向栈顶，从栈顶到数组的首地址可用位移量表示，变址reg可用访数组中的某个元素。

8、隐含寻址

特点：在指令码中无明确数位表示操作数地址，但指令操作码中隐含指明了操作数地址。

如字符串操作指令（SI，DI）

例：STOSB　　；存入串指令

[DI]←[AL]，

[DI]←[DI]+1

说明：若M与I/O端口统一编址，则以上寻址方式同样适于I/O端口

I/O端口寻址方式：

P76，两种寻址方式：

1、直接端口寻址方式（端口地址用8位表示）

```
IN    AL（AX），port8 ；
OUT   port8 ， AL（AX）；
```

例：OUT 0D6H ， AL

2、间接端口寻址方式（端口地址用16位表示）

```
IN    AL（AX），DX;
OUT   DX  ，  AL（AX）
```

说明：DX中是16位端口地址，范围0000H~FFFFH

注意：端口寻址方式中OPD形式表达上与存储器寻址方式中的不同。超出FFH的要用DX间接寻址。

2）转移地址寻址方式

（专对转移指令，确定转向地址）4种P77

- 段内直接寻址
- 段内间接寻址
- 段间直接寻址——目标地址在段间
- 段间间接寻址

前面我们介绍了CPU执行指令的过程，指令是按顺序存放在M中，而程序执行顺序是由CS和IP的内容决定。当程序执行到某一特定位置时，据程序设计要求，需脱离程序的正常顺序执行，而把它转移到指定的指令地址，程序转移指令通过改变IP和CS内容，就可改变程序执行顺序。

根据程序转移地址相对于当前程序地址的关系，可分为段内、段外；又根据转移地址是否直接出现在指令中，分为直接、间接，所以有四种程序转移寻址方式：

- 段内直接寻址
- 段内间接寻址

} 不改变CS，只改变IP；

- 段间直接寻址
- 段间间接寻址。

} 既改变IP，也改变CS；

1、直接寻址

（ direct addressing）(相对寻址)

- 定义：在指令的地址码字段，直接给出所需的操作数(或指令) 在存储器中的地址。

形式：JMP 转移地址（通常是地址标号）

说明：

（1）大多数的汇编编译器支持标号（指令在程序存储器中的地址），而不需要也不支持程序员在指令中给出转移的直接地址（数据形式）。

（2）当不指定标号的属性时，汇编器会自动选择最优的寻址方式，有三种：

(1)段内直接寻址

- ① 8位位移量的相对寻址（段内）

  当转移目标地址与当前地址很近时：

  —128~127,则选择8位位移量的相对寻址。

例1： JMP NEXT

相当于：JMP SHORT NEXT

CS：0035 EBF6 JMP 002D；F6=2D－37

CS：0037

例2： JMP NEXT

相当于：JMP SHORT NEXT

CS：0031 EB04 JMP 0037；

CS：0033 ；04=37－33

- 指令的地址由程序计数器 PC 的内容（即当前执行指令的地址）和指令的相对寻址偏移量相加得到。

- ②　16位位移量的相对寻址（段内）

当转移目标地址与当前地址超出：

—128~127,则选择16位位移量的相对寻址

例：JMP　NEXT

相当于：JMP　NEAR　PTR　NEXT

转向的EA=（IP）当前+ disp

8短程转移
16近程转移

→ $IP_{转向}$

指令中补码表示
的有符号数

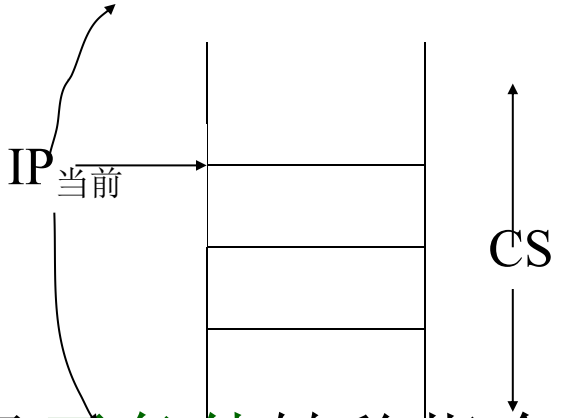
短程：JMP　SHORT　NEXT（8 disp）
近程：JMP　NEAR　PTR　NEXT（16 disp）

段内

类型

属性操作符
（已定义过的V/L）

标号表示位移量



适用于条件及无条件转移指令

仅允8disp　　若为8disp称短跳转

(2)段间直接寻址—目标地址在段间

当地址号在另一个代码段中定义时，汇编程序可能需要转移指令指出其转移范围。

指令中给出4个字节（32位）转向目标地址：

前两个字节为偏移地址IP；后两个为段地址CS。

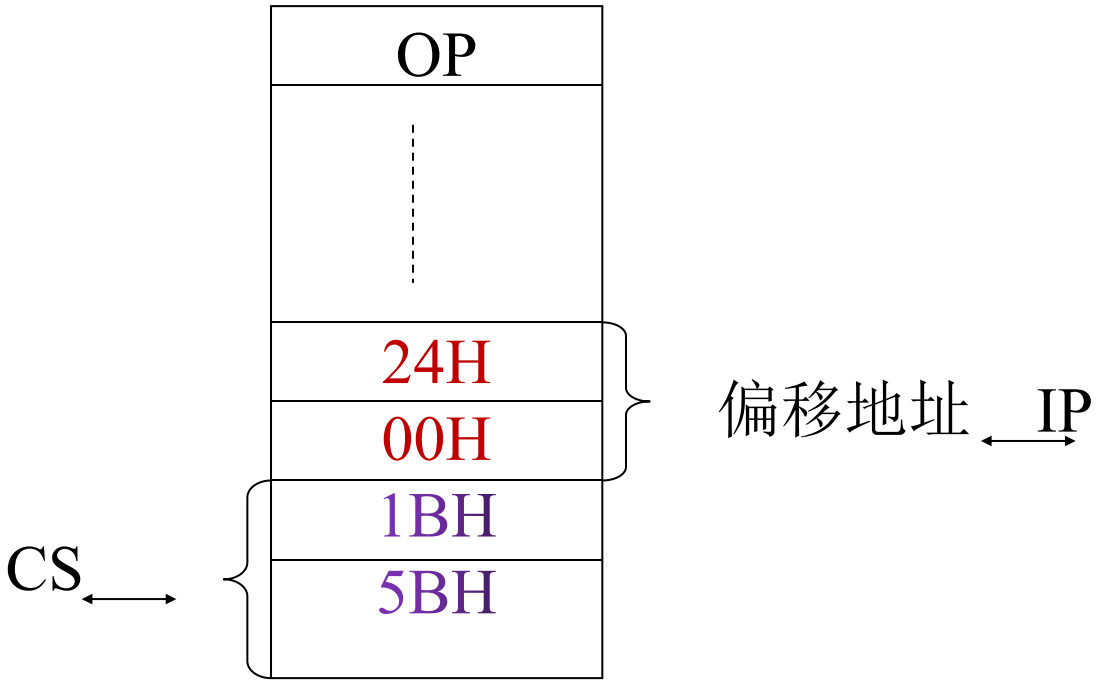
- 例如：JMP FAR PTR NEXT;

  等价于JMP 5B1BH:0024H

  段（CS）：偏移（IP）

  （后两字节）（前两字节）

- 其执行过程如下：



返回

2、间接寻址

- **定义**：指令中未直接指定转移地址，即操作数部分不是地址标号；转移地址的EA在寄存器或存储单元中。
- 即：指令的地址码字段给出的内容既不是操作数，也不是操作数的地址，而是操作数（或指令）地址的地址，这被称为间接寻址方式，多一次读内存储器的操作。

注意：存储单元寻址可以用数据寻址方式中的大多数寻址方式取得。（由汇编程 序限制）

（1）段内间接寻址：

- 同一CS段内，转移地址的EA在一个16位寄存器或2个连续的存储单元中。
- 同一CS段内，目标地址的16位EA在：

  Reg (各种寻址方式)
  M除立即数寻址外

  单元的内容中

- IP←——EA
- JMP REG
- JMP 存储单元
- 注：仅修改IP

例：JMP　TABLE[BX]　　；段内间接

寄存器相对

适用范围：不能用于条件转移指令

JMP　BX

JMP　WORD PTR [BP+TABLE]

物理地址=10H×（CS）+EA

注：数据寻址计算出EA即IP,再算物理地址。

下面举例说明在段内间接寻址方式的转移指令中，转移的EA计算方法

假设：（DS）=2000H，（BX）=1256H
（SI）=528FH，位移量=20A1H，
（232F7H）=3280H，（264E5H）=2450H

举例：

1）JMP　　BX

执行后：（IP）=1256H

2）JMP　　TABLE[BX]

执行后：（IP）=（10H×（DS）+（BX）+位移量）

=（20000+1256+20A1）

=（232F7）=3280H

3）JMP　　[BX][SI]

执行后：（IP）=（10H×（DS）+（BX）+（SI））

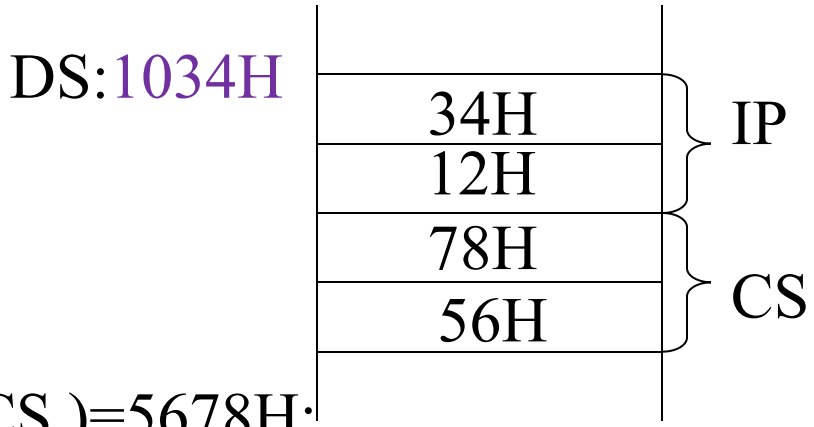
=（20000+1256+528F）

=（264E5）=2450H

（2）段间间接寻址

- 转向目标地址在存储单元（各种存储器操作数寻址方式确定计算）中连续四个单元。 例如：假定（BX)=1034H
- JMP DWORD PTR [BX]

双字操作符



指令执行后：(CS)=5678H;

(IP)=1234H

小结

指令由（ 操作码）和（操作数 ）构成

如何取得操作数——称为（寻址）

寻址方式可分为

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

寄存器间接、寄存器相对、基址变址、相对基址变址四种寻址方式的比较：

寻址方式	指令操作数形式

- 寄存器间接 —— 只有一个寄存器（BX/BP/SI/DI之一）
- 寄存器相对 —— 一个寄存器加上位移量
- 基址—变址 —— 两个不同类别的寄存器
- 相对基址-变址 —— 两个不同类别的寄存器加上位移量

例题

例1 设有关寄存器和存储单元的内容如下：

（DS）=2000H，（BX）=0100H，（SI）=0002H，

（20100H）=12H，（21200H）=2AH，

（20101H）=34H，（21201H）=4CH，

（20102H）=56H，（21202H）=0B7H，

（20103H）=78H，（21203H）=65H。

试说明下列各条指令执行之后，AX寄存器的内容？并指出源操作数的寻址方式。

（1）MOV AX，1200H；

（2）MOV AX，BX；

（3）MOV AX，[1200H]；

（4）MOV AX，[BX]；

（5）MOV AX，1100H[BX]；

（6）MOV AX，[BX][SI]；

（7）MOV AX，1100H[BX][SI]。

- 分析：DS给出数据段地址，因此，数据段的起始物理地址为

  （DS）×10H=20000H。

- 由BX、SI给出的偏移量，根据寻址方式（除立即、寄存器寻址外）求出有效地址EA，再加上20000H，求出物理地址，此地址（连续两字节）的内容就是AX的内容

（1）立即；　（AX）=1200H

（2）寄存器；（AX）=0100H

（3）直接寻址；　EA=1200H

物理地址=20000H+1200H=21200H，因为（21200H）=2AH，（21201H）=4CH，；（AX）=4C2AH

（4）寄存器间接；　　EA=（BX）=0100H

物理地址=20000H+0100H=20100H，因为（20100H）=12H，（20101H）=34H，；（AX）=3412H

（5）寄存器相对寻址；EA=1100H+0100H=1200H

物理地址=20000H+1200H=21200H

；（AX）=（21200H）=4C2AH

（6）基址变址；　EA=（BX）+（SI）=0100H+0002H=0102H

物理地址=20000H+0102H=20102H，因为（20102H）=56H，（20103H）=78H，

；（AX）=7856H

- （7）基址变址且相对寻址；

  EA=1100H+（BX）+（SI）=1100H+0100H+0002H=1202H

  物理地址=20000H+1202H=21202H，因为（21202H）=0B7H，（21203H）=65H，

  ；（AX）=65B7H

注意：

- 区别立即寻址方式和直接寻址方式。如：
  - （1）将数据1200H送入AX寄存器，
  - （3）将数据段中1200H单元的内容送AX。
- 使用寄存器间接寻址时应注意和寄存器寻址方式的区别。如：
  - （2）BX中的内容传送到AX，
  - （4）BX所指示的地址中的内容送AX。
- AX中存放的是一个字而不是字节数据。

80386及其后继机型的寻址方式（了解）

1、虚地址方式下的寻址方式

80386CPU向上具有虚地址方式。

存储器管理分段、页两级管理，每段最长是4GB（2^{32}个字节）。线性地址的构成为：段基地址（32位）+偏移地址（32）；段寄存器（16位）中的内容用于间接获取32位段基址的一个称为16位选择子（在段描述符表中选择对应的段描述符）。

在编程时都是用虚拟地址(逻辑地址)表示，即

选择器：有效地址

虚地址方式下的寻址方式共11种，与实方式下相比，不同的是：

①因为80386是32位CPU，因此寄存器有32位的，于是立即数可以有32位的。

②虚地址方式偏移地址EA是32位的，所以寻址MEM是32位地址。

③寄存器间接/基址寻址时，所有通用寄存器都可以作为间接/基址寄存器，包括ESP。

④具有比例因子的寻址，注意：比例因子只能是1、2、4、8。

80386的寻址方式分为三大类：

1）寄存器寻址方式

在这种寻址方式中，操作数放在32位、16位或8位的通用寄存器中。

2）立即数寻址方式

3）存贮器寻址方式

EA=基址+变址×比例因子+位移量

EA=基址+变址×比例因子+位移量

（1）基址：任何通用寄存器都可作为基址寄存器，其内容即为基址。

（2）位移量：在指令操作码后面的32位、16位或8位的数。

（3）变址：除了ESP寄存器外，任何通用寄存器都可以作为变址寄存器，其内容即为变址值。

（4）比例因子：变址寄存器的值可以乘以一个比例因子，根据操作数的长度可为1字节、2字节、4字节或8字节，比例因子相应地可为1，2，4或8。

按照这4个分量组合有效地址的不同方法，可以有9种存贮器寻址方式。此处仅对3种寻址方式加以说明。P77

①比例变址寻址(Scaled Indexed Addressing)：变址寄存器的内容乘以比例因子，再加位移量得到EA。

②基址比例变址寻址(Based Scaled Indexed Addressing)：变址寄存器的内容乘以比例因子，再加上基址寄存器的内容作为EA。

③相对基址比例变址寻址(Relative Based Scaled Indexed Addressing)：变址寄存器的内容乘以比例因子，再加上基址寄存器的内容，又加上位移量，形成EA。

以上各种寻址方式都是由CPU 自动完成的。

- 【例3. 15】 MOV AL，ARRAY [EBP + 2*EDI]
  有效地址EA = ARRAY + EBP + 2*EDI，
  将SS:EA中字节的内容送入AL中。

ARM寻址方式(补充了解）

ARM寻址方式(1)

- 寻址就是找到存储数据或指令的地址，寻址方式的方便与快捷是衡量CPU性能的一个重要方面，ARM处理器共有八种寻址方式：
  - 立即数寻址
  - 寄存器寻址
  - 寄存器间接寻址
  - 基址寻址
  - 多寄存器寻址
  - 堆栈寻址
  - 相对寻址
  - 寄存器移位寻址

ARM寻址与x86寻址异同

- ARM寻址形式上只多了一个多寄存器寻址，其他类同，区别仅在于寄存器号展现形式不同而已。

ARM寻址方式(2)

- 立即数寻址：
  - 立即数寻址指令中的地址码就是操作数本身，可以立即使用的操作数。其中，#0xFF000和#64都是立即数。如操作数是常量，用#表示常量；0x或&表示16进制数，否则表示十进制数。

例如：

MOV R0,#0xFF000

@指令省略了第1个操作数寄存器。将立即数0xFF000(第2操作数)装入R0寄存器

SUB R0,R0,#64

@R0减64，结果放入R0

ARM寻址方式(3)

- 寄存器寻址：
  - 操作数的值在寄存器中，指令执行时直接取出寄存器值来操作，寄存器寻址是根据寄存器编码获取寄存器内存储的操作数

例如：

MOV R1,R2

@将R2的值存入R1　在第1个操作数寄存器的位置存放R2编码

SUB R0,R1,R2

@将R1的值减去R2的值，结果保存到R0在第2操作数位置，存放的是寄存器R2的编码

ARM寻址方式(4)

- 寄存器间接寻址：
  - 操作数从寄存器所指向的内存中取出，寄存地存储的是内存地址

例如：

LDR R1,[R2]

@将R2指向的存储单元的数据读出，保存在R1中 R2相当于指针变量

STR R1,[R2]

@将R1的值写入到R2所指向的内存

SWP R1,R1,[R2]

@将寄存器R1的值和R2指定的存储单元的内容交换

[R2]表示寄存器所指向的内存

LDR 指令用于读取内存数据

STR 指令用于写入内存数据

ARM寻址方式(5)

- 基址变址寻址：
  - 基址寄存器的内容与指令中的偏移量相加，得到有效操作数的地址，然后访问该地址空间，基址变址寻址分为三种：
    - 前索引，例如：

      LDR R0, [R1,#4] @R1存的地址+4，访问新地址里面的值，放到R0；
    - 自动索引，例如：

      LDR R0, [R1,#4]! @在前索引的基础上，新地址回写进R1；(注：!表示回写地址)
    - 后索引，例如：

      LDR R0 [R1],#4 @R1存的地址的内容写进R0，R1存的地址+4再写进R1；

ARM寻址方式(6)

- 多寄存器寻址：
  - 一条指令完成多个寄存器的传送，最多16个寄存器，也称为块拷贝寻址

  例如：

  LDMIA R1!,{R2-R7,R12}

  @将R1指向的存储单元中的数据读写到R2～R7、R12中，然后R1自加1

  STMIA R1!,{R2-R7,R12}

  @将寄存器R2～R7、R12的值保存到R1指向的存储单元中，然后R1自加1

  注：基址寄存器不允许为R15，寄存器列表可以为R0～R15 的任意组合。这里R1没有写成[R1]!，是因为这个位不是操作数位，而是寄存器位

  LDMIA 和 STMIA 是块拷贝指令，LDMIA是从R1所指向的内存中读数据，STMIA是向R1所指向的内存写入数据
  - 执行这类指令要考虑如下几个问题：
    - 基址寄存器指向原始地址有没有放一个有效值
    - 寄存器列表哪个寄存器被最先传送
    - 存储器地址增长方向
    - 指令执行完成后，基址寄存器有没有指向一个有效值

ARM寻址方式(7)

- 寄存器堆栈寻址：
  - 是按特定顺序存取存储区，按后进先出原则，使用专门的寄存器SP（堆栈指针)指向一块存储区

例如：

LDMIA SP!,{R2-R7,R12}

@将栈内的数据，读写到R2～R7、R12中，然后下一个地址成为栈顶

STMIA SP!,{R2-R7,R12}

@将寄存器R2～R7、R12的值保存到SP指向的栈中，SP指向的是栈顶

ARM寻址方式(8)

- 相对寻址：
  - 即读取指令本身在内存中的地址。是相对于PC内指令地址偏移后的地址。由程序计数器PC提供基准地址，指令中的地址码字段作为偏移量，两者相加后得到的地址即为操作数的有效地址

例如：

```
BL   ROUTE1                    @调用到 ROUTE1 子程序
BEQ LOOP                       @条件跳转到 LOOP 标号处
…
LOOP MOV R2,#2
…
ROUTE1
…
```