80X86汇编知识回顾

- 指令格式[...]
- 寻址方式
- 典型指令
- 调用与栈

一、寻址方式

操作数有三种可能的存放位置

（1）操作数直接在指令中，即跟随在指令操作码之后，指令的操作数部分就是操作数本身，这种操作数叫立即操作数。

（2）操作数存放在CPU的某个内部寄存器中，这时指令的操作数部分是CPU内部寄存器的一个编码，称为寄存器操作数。

（3）操作数存放在内存数据区中，这时指令的操作数部分包含此操作数所在的内存地址，称为存储器操作数。

一、寻址方式

寻址及寻址方式的概念

计算机的指令中通常要指定操作数的位置，即给出操作数的地址信息，在执行时需要根据这个地址信息找到需要的操作数，这种寻找操作数的过程称为寻址。寻址方式就是寻找操作数或操作数地址的方式。

不同机器的指令系统都规定了一些寻址方式以供编程时选择使用，根据给定的寻址方式，就可以方便地访问各类操作数。

一、寻址方式

1.立即数寻址方式

立即数寻址方式是指操作数直接存放在给定的指令中，紧跟在操作码之后。

立即数可以是8位或16位二进制数。例如，给定如下指令，采用立即数寻址：

```
MOV   AL, 10                ;   十进制数（D）
MOV   AL, 00100101B         ;   二进制数（B）
MOV   AX, 263AH             ;   十六进制数（H）
```

一、寻址方式

2. 寄存器寻址方式

寄存器寻址方式是在指令中直接给出寄存器名，寄存器中的内容即为所需操作数。在寄存器寻址方式下，操作数存在于指令规定的8位、16位寄存器中。寄存器可用来存放源操作数，也可用来存放目的操作数。

寄存器寻址方式是CPU内部的操作，不需要访问总线周期，因此**指令的执行速度比较快**。

对于16位操作数，寄存器可以是AX、BX、CX、DX、SI、DI、SP、BP等。

对于8位操作数，寄存器可以是AH、AL、BH、BL、CH、CL、DH、DL等。

一、寻址方式

例： INC SI
MOV AX，BX

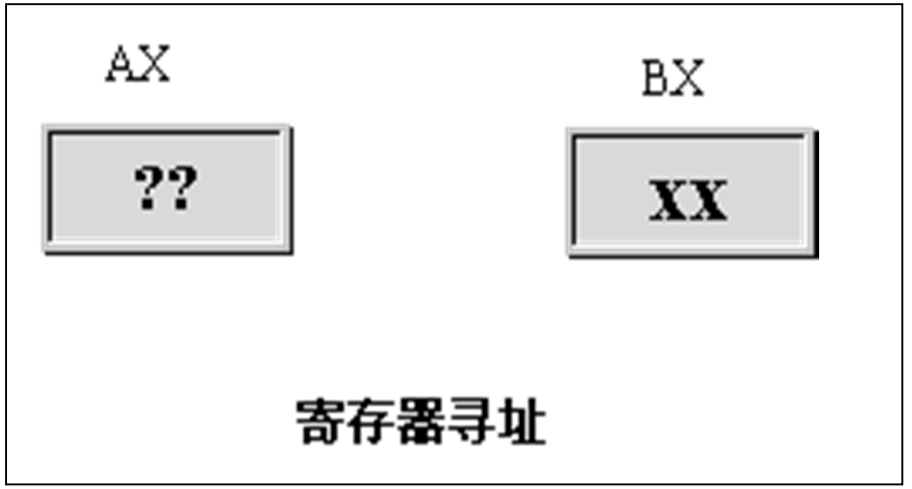
① Reg是任何CPU中的通用Reg

AX，BX，CX.....BP

AH，AL..... .....等

② 操作数在Reg中，无需访问存贮器，执行速度快。

③ 若选用AX，执行指令时间更短。



指令执行后，源操作数不变，目的操作数为源的内容。

一、寻址方式

3. 存储器寻址方式

计算机中访问内存的寻址方式有多种，不管哪一种寻址方式，最终都将得到存放操作数的**物理地址**。采用存储器寻址时，指令中需要给出操作数的地址信息。

存储器操作数的**有效地址**EA的计算方法和寻址方式有着密切地联系，而操作数**物理地址**PA的计算则和操作数的具体存放位置有关。

一、寻址方式

存储器寻址方式

(1). 直接寻址方式

直接寻址方式是一种针对内存的寻址方式。在这种寻址方式下，指令中给出的地址码即为操作数的有效地址EA，它是一个8位或16位的**位移量**。在默认方式下，操作数存放在**数据段**DS中，如果要对除DS段之外的其他段如CS、ES、SS中的数据寻址，应在指令中增加前缀，指出段寄存器名，这称为**段跨越**。

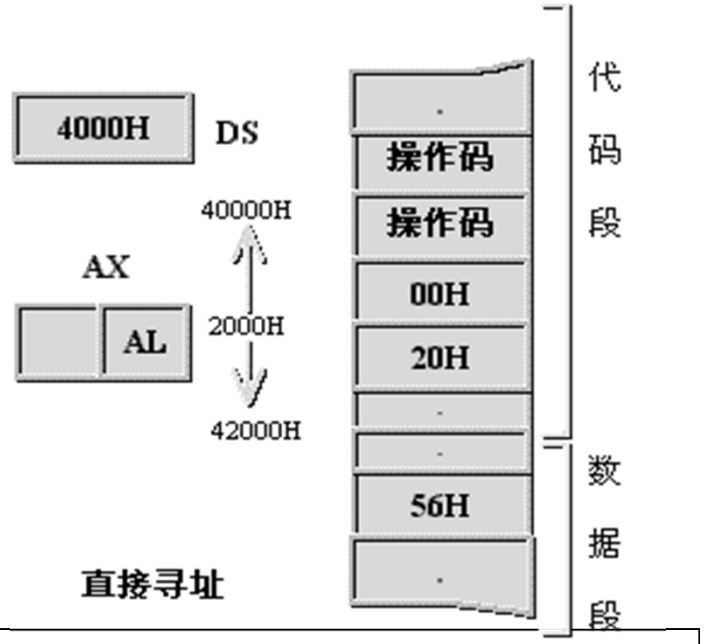
在直接寻址方式的指令中，操作数的有效地址EA已经给出，那么操作数的物理地址为：**PA＝（DS）×10H＋EA**。

例：MOV　　AL，[2000H]；

将DS段中2000H单元的内容送AL

若DS＝4000H，则物理地址为：

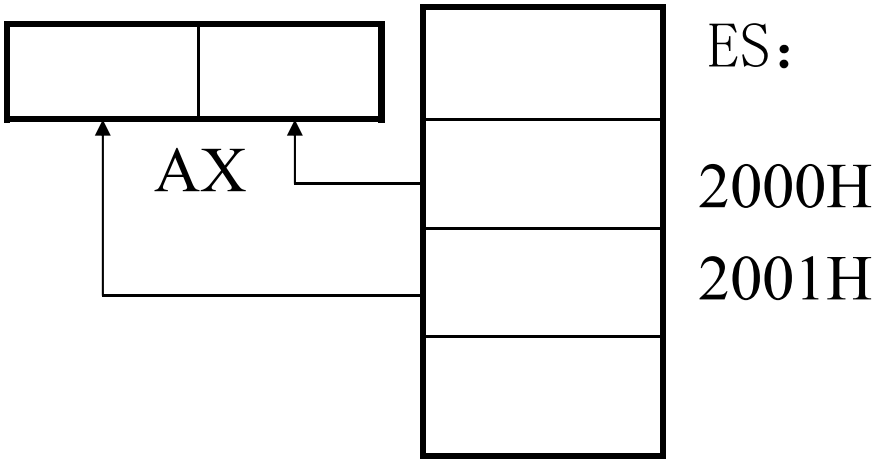
4000H×16+2000H＝42000H。见右图.



① 允许段超越

例：MOV　AX，ES:[2000H]

（或 ES:MOV　AX，[2000H]）



将ES段中2000H、2001H单元内容分别送AL，AH（低对低，高对高）

不管数据在那个段内存放，只要不是**DS**，一定要加段超越前缀。

② 直接寻址中EA可以以变量名的形式给出。

例：VALUE　DB　12H

　　MOV　AL，[VALUE]

或　　MOV　AL，VALUE　　;VALUE又称为符号地址

一、寻址方式

存储器寻址方式

(2). 寄存器间接寻址方式

寄存器间接寻址方式是指操作数的有效地址EA**在指定的寄存器**中，这种寻址方式是在指令中给出寄存器，寄存器中的内容为操作数的有效地址。

寄存器使用规定如下：

16位寻址时，EA在DI、SI、BX、BP中， 这时：

若以DI、SI、BX间接寻址，则默认操作数在数据段中.

操作数物理地址＝DS×16＋BX→（或SI、DI）

若以BP间接寻址，则默认操作数在堆栈段内.

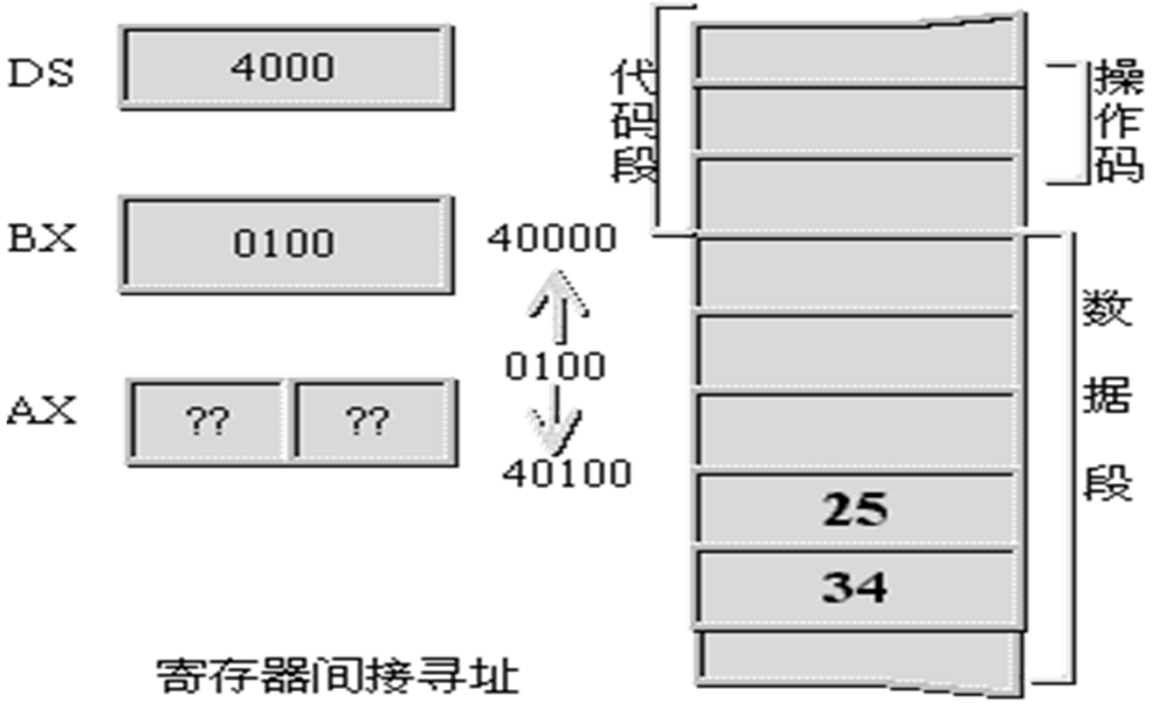
操作数物理地址＝SS×16＋BP

若操作数不在以上规定段内，则必须在指令中加上段超越前缀。

一、寻址方式

例：MOV　AX，[SI]　　;将DS段[SI][SI＋1]的内容送AL,AH
　　MOV　BH，[BP]　　;将SS段[BP]的内容送BH
　　MOV　CX，ES:[BX]　;将ES段[BX][BX＋1]内容送CL,CH

寄存器间接寻址示意图



例：MOV AX,[BX]
设DS＝4000H，
BX=0100H
寄存器间接寻址示意图如左。

一、寻址方式

存储器寻址方式

(3). 寄存器相对寻址方式

这种寻址方式是在指令中给定一个基址寄存器或变址寄存器和一个8位或16位的相对偏移量，**两者之和**作为操作数的**有效地址EA**。当选择间址寄存器BX、SI、DI时，指示的是数据段中的数据，选择BP作间址寄存器时，指示的是堆栈段中的数据。

有效地址计算为：EA＝（reg）＋8位或16位偏移量；其中reg为给定寄存器。

物理地址计算为:

PA＝（DS）×10H＋EA （使用BX、SI、DI间址寄存器）

PA＝（SS）×10H＋EA （使用BP作为间址寄存器）

例子 MOV AX,[BX+10H]

EA=(BX)+10H PA=(DS) ×10H+EA

一、寻址方式

存储器寻址方式

(4). 基址变址寻址方式

在基址变址寻址方式中，有效地址EA是基址寄存器加变址寄存器，即两个寄存器的**内容之和**为操作数的**有效地址**。在该寻址方式中，当基址寄存器和变址寄存器的默认段寄存器不同时，一般由基址寄存器来决定默认用哪一个段寄存器作为段基址指针。若在指令中规定了段跨越，则可以用其他寄存器作为段基地址。

基址变址寻址方式的物理地址计算公式为：

物理地址PA＝（DS）×10H＋（BX）＋（SI）

物理地址PA＝（SS）×10H＋（BP）＋（DI）

例子　　MOV AX,[BX+SI]

EA=(BX)+(SI)　　PA=(DS) ×**10H+EA**

一、寻址方式

存储器寻址方式

(5). 相对基址变址寻址方式

这种寻址方式是在指令中给出一个**基址寄存器**、一个**变址寄存器**和8位或16位的**偏移量**，**三者之和**作为操作数的**有效地址**。

基址寄存器可取BX或BP，变址寄存器可取SI或DI。

如果基址寄存器采用BX，则段寄存器使用DS；

如果基址寄存器采用BP，则段寄存器使用SS。

其物理地址计算为：

PA＝（DS）×10H＋（BX）＋（SI）或（DI）＋偏移量

PA＝（SS）×10H＋（BP）＋（SI）或（DI）＋偏移量

一、寻址方式

与I/O端口有关的寻址方式

(1) 直接端口寻址

(2) 寄存器间接端口寻址

二、典型指令

8086 CPU的指令系统

操作数可以是8位或16位，偏移地址是16位。

按功能可将指令分成六大类

- 数据传送类指令
- 算术运算类指令
- 逻辑运算与移位类指令
- 串操作类指令
- 控制转移类指令
- 处理器控制类指令

二、典型指令

使用MOV指令进行数据传送时要注意以下几点：

（1） 段寄存器CS及立即数不能作为目标操作数；

（2） 两个存储单元之间不允许直接传送数据；

（3）立即数不能直接传送到段寄存器；

（4）两个段寄存器之间不能直接传送数据；

（5）传送数据的类型必须匹配；

（6）MOV指令不影响标志位。

二、典型指令

例：

```
MOV  AL，CH                ；通用寄存器之间传送字节数据
MOV  DS，AX  ；通用寄存器→段寄存器（CS不能是目标）
MOV  AX，0FF3BH            ；立即数→通用寄存器
MOV  AL，BUFFER            ；存储器→通用寄存器
MOV  DAT[BP+DI]，ES        ；段寄存器→存储器
MOV  [1000H]，25H          ；立即数→存储器
MOV  CX，[1000H]           ；存储器→通用寄存器
```

二、典型指令

（2）堆栈操作指令PUSH/POP

进栈指令：PUSH　opr　　；SP←SP－2

出栈指令：POP　　opr　　；SP←SP+2

堆栈是存储器中的一个特殊区域，主要用于存入和取出数据，堆栈是以“先进后出”的方式进行数据操作的。在8086的堆栈组织中，堆栈从高地址向低地址方向生长，它只有一个出入口，堆栈指针寄存器SP始终指向堆栈的栈顶单元

二、典型指令

具体的入/出栈指令如下：

PUSH	reg16	；	POP	reg16
PUSH	Sreg	；	POP	Sreg
PUSH	mem16	；	POP	mem16

注：1. 程序中有一个PUSH，必有一个对应的POP。

2. 遵循后进先出原则。

3. 按字进行操作（PUSH　AH ；　POP　BL （错误））

4. PUSH　CS ；　POP　CS

（可以）　　（错误）

二、典型指令

3. 地址传送指令

8086的地址传送指令用于控制寻址机构，它可将存储器操作数的地址传送到16位目标寄存器中。

（1）有效地址送寄存器指令：LEA reg，src

LEA指令功能是将存储器操作数src的有效地址传送到16位的通用寄存器reg。

例：LEA AX，[2728H] ； AX=2728H
LEA BX，[BP+SI] ； BX=BP+SI的值。
LEA SP，[0482H] ； SP=0482H

注：MOV指令与LEA的不同：
前者传送操作数的内容，后者传送操作数的地址。

例：MOV DI ，TABLE ； DI← [TABLE]
LEA DI ，TABLE ； DI← TABLE所在单元的EA

二、典型指令

通常有 { MOV BX，OFFSET VARWORD ; LEA BX，VARWORD } 两者效果一致

例：比较 LEA BX，BUFFER
MOV BX， BUFFER
两指令的不同

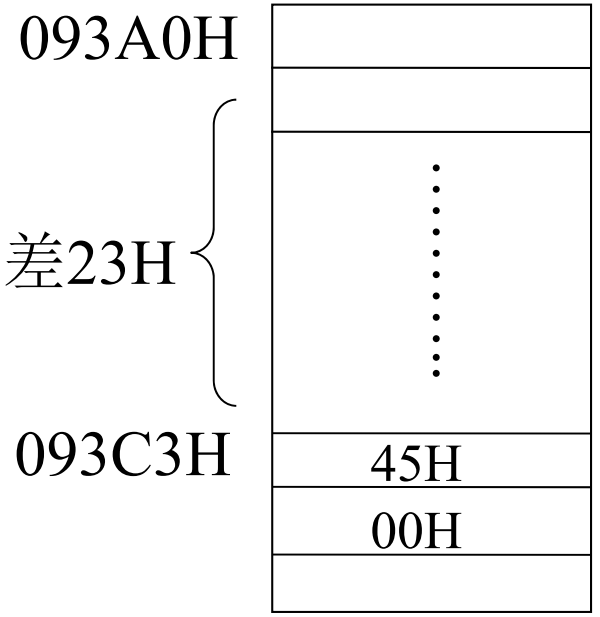
存储单元内容如右图所示。

DS:093AH，BUFFER 物理地址 093C3H

则

LEA BX，BUFFER后，BX=0023H

MOV BX，BUFFER后，BX=0045H



二、典型指令

1. 逻辑运算指令

有以下5条逻辑运算指令，它们可对8位或16位操作数按位进行逻辑运算。

（1）逻辑与指令：AND dst，src

（2）逻辑或指令：OR dst，src

（3）逻辑异或指令：XOR dst，src

（4）逻辑非指令：NOT dst

（5）测试指令：TEST dst，src

。

二、典型指令

2. 移位指令

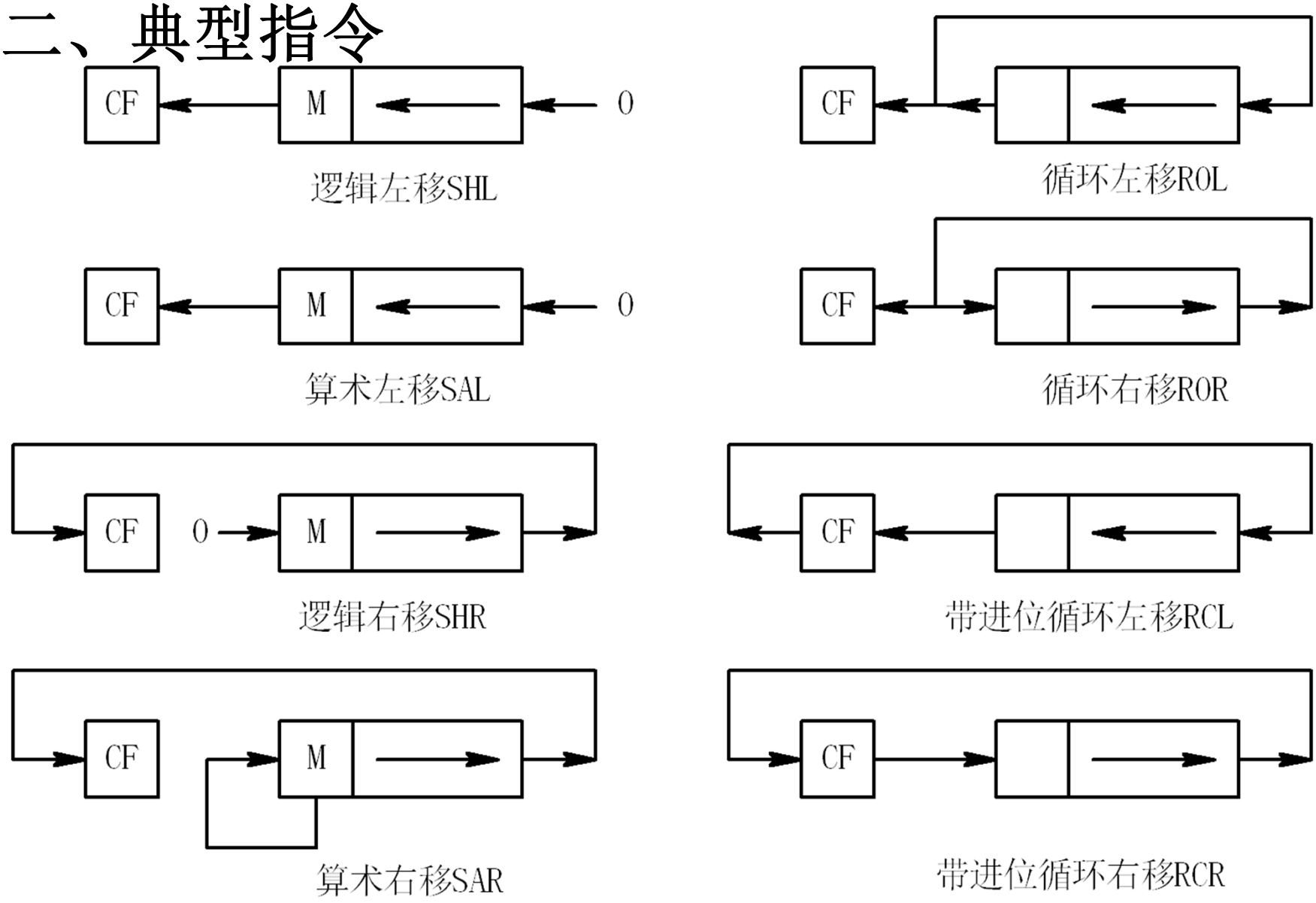
移位操作类指令可以对字节或字数据中的各位进行算术移位、逻辑移位或循环移位。

移位指令的格式为： SHL/SAL/SHR/SAR dst，1/ CL

循环移位指令的格式为：ROL/ ROR/ RCL/ RCR dst，1/ CL

上述指令分别对操作数进行逻辑左移SHL、算术左移SAL、逻辑右移SHR、算术右移SAR；循环左移ROL、循环右移ROR、带进位的循环左移RCL、带进位的循环右移RCR等操作。操作数可以是字节或字操作。

二、典型指令



（注：CF为进位位、M为符号位）

图 移位指令的操作功能示意

二、典型指令

控制转移类指令

程序的执行一般是按指令顺序逐条执行的，但有时需要改变程序的执行流程。控制转移类指令就是用来改变程序执行的方向，也就是修改IP和CS的值。通过控制转移指令可实现各种结构化程序设计，如分支结构程序、循环结构程序等。

（1）如果指令给出改变IP中内容的信息，转移的目标位置和转移指令在同一个代码段，则称为段内转移；

（2）如果指令给出改变IP中内容的信息，又给出改变CS中内容的信息，转移的目标位置和转移指令不在同一个代码段，则称为段间转移。

根据转移指令的功能，可分为无条件转移指令、条件转移指令、循环控制指令、子程序调用和返回指令4类。

二、典型指令

控制转移指令：

- 无条件转移指令

  JMP

- 条件转移指令

  JZ / JNZ 、 JE / JNE、 JS / JNS、 JO / JNO、

  JP / JNP、 JB / JNB、 JL / JNL、 JBE / JNBE、

  JLE / JNLE、 JCXZ

- 循环指令

  LOOP、LOOPZ / LOOPE、LOOPNZ / LOOPNE

- 子程序调用和返回指令

  CALL、RET

- 中断与中断返回指令

  INT、INTO、IRET

二、典型指令

4. 子程序调用和返回指令

（1）子程序调用指令

指令格式为：CALL NEAR PTR opr ；段内调用

CALL FAR PTR opr ；段间调用

（2）子程序返回指令RET

指令格式： RET

或 RET 表达式

二、典型指令

处理器控制类指令

这类指令主要用于修改状态标志位、控制CPU的功能如使CPU暂停、等待、空操作等。

- 标志处理指令

CLC、 STC、 CMC、

CLD、STD、

CLI、STI

- 其他处理机控制与杂项操作指令

NOP、HLT、WAIT、ESC、LOCK

二、典型指令

标志处理指令：

CLC　　CF ← 0
CMC　　CF ← ¬CF
STC　　CF ← 1

CLD　　DF ← 0
STD　　DF ← 1

CLI　　IF ← 0
STI　　IF ← 1

注意：　* 只影响本指令指定的标志

二、典型指令

其他处理机控制与杂项操作指令：

NOP 无操作（机器码占一个字节）

HLT 暂停机（等待一次外中断，之后继续执行程序）

WAIT 等待（等待外中断，之后仍继续等待）

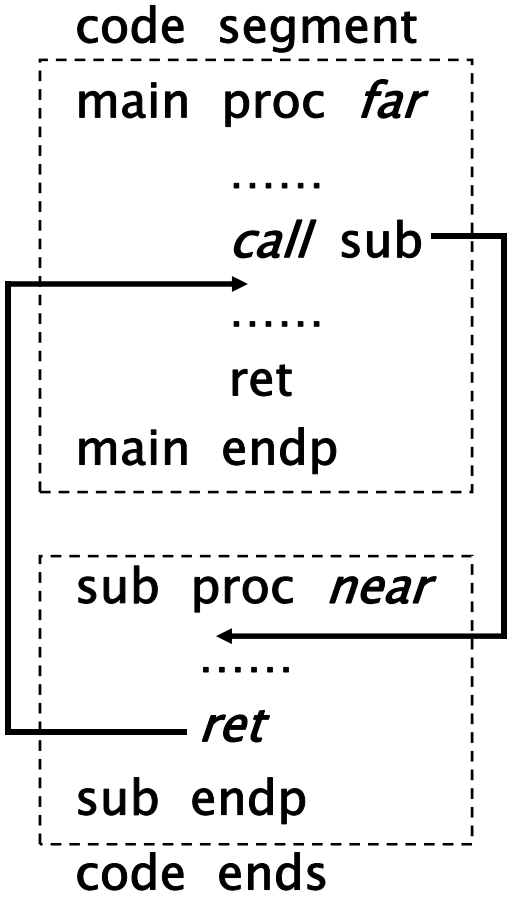
ESC 换码

LOCK 封锁（维持总线的锁存信号，直到其后的指令执行完）

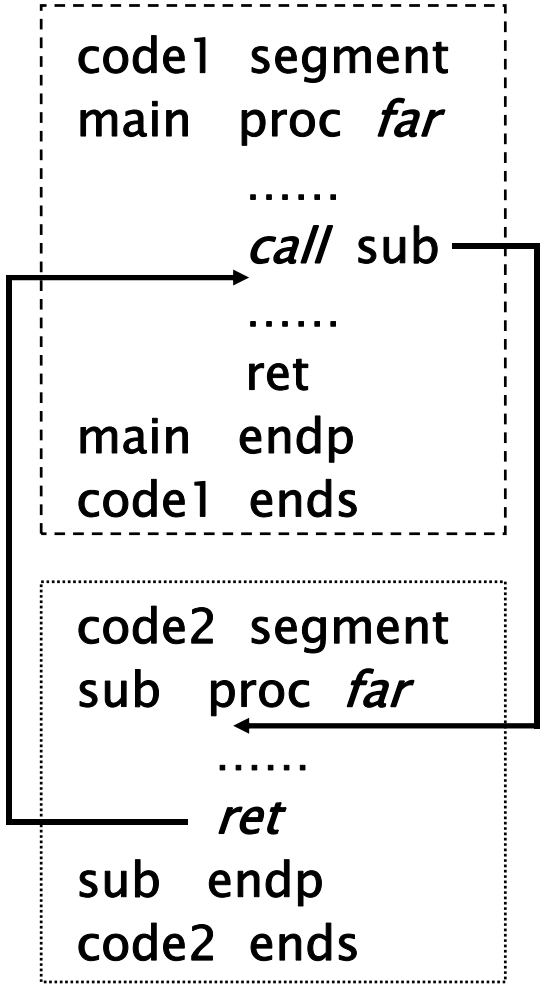
注意： * 不影响条件标志

三、调用和栈（典型指令）

子程序调用和返回指令：



段内调用和返回



段间调用和返回

三、调用和栈

CALL 调用指令

段内直接近调用：**CALL DST**

执行操作：**(SP) ← (SP) - 2**

((SP)+1,(SP)) ← (IP) **;IP入栈**

(IP) ← (IP) + 16位位移量 **;更新IP**

段内间接近调用：**CALL DST**

执行操作：**(SP) ← (SP) - 2**

((SP)+1,(SP)) ← (IP)

(IP) ← (EA)

三、调用和栈

段间直接远调用：**CALL DST**

执行操作：**(SP) ← (SP) - 2**

((SP)+1,(SP)) ← (CS)　;CS入栈

(SP) ← (SP) - 2

((SP)+1,(SP)) ← (IP)　;IP入栈

(IP) ← 偏移地址　;更新IP

(CS) ← 段地址　;更新CS

段间间接远调用：**CALL DST**

执行操作：**(SP) ← (SP) - 2**

((SP)+1,(SP)) ← (CS)

(SP) ← (SP) - 2

((SP)+1,(SP)) ← (IP)

(IP) ← (EA)

(CS) ← (EA+2)

三、调用和栈

RET 返回指令

段内近返回：**RET**

执行操作：**(IP) ← ((SP)+1,(SP))　;IP出栈**

(SP) ← (SP) + 2

段内带立即数近返回：**RET EXP　;IP出栈且清理栈**

段间远返回：**RET**

执行操作：**(IP) ← ((SP)+1,(SP))　;IP,CS出栈**

(SP) ← (SP) + 2

(CS) ← ((SP)+1,(SP))

(SP) ← (SP) + 2

段间带立即数远返回：**RET EXP ;IP,CS出栈且清理栈**

三、调用和栈

例：带立即数返回　;清理栈

```
code  segment
main  proc  far
      ……
      push  ax
      push  bx
      push  cx
      call  sub
      ……
      ret
main  endp

sub   proc  near
      ……
      ret  6
sub   endp
code  ends
```