汇编语言程序设计
第三章 寻址方式与指令系统 III

张华平 副教授 博士

Email: kevinzhang@bit.edu.cn

Website: http://www.nlpir.org/

@ICTCLAS张华平博士

大数据搜索与挖掘实验室 (wSMS@BIT)

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

3.5 逻辑指令

逻辑指令提供了对二进制位的控制。该系统提供的逻辑指令包括：

逻辑运算指令

位测试指令

位扫描指令

基本移位指令

循环移位指令

双精度移位指令

一、逻辑运算指令

逻辑运算指令见下表。这些指令的操作数可以是8位、16位、32位,其寻址方式与MOV指令的限制相同。

名称	格 式	功 能	标 志
逻辑非	NOT DST	(DST)按位变反送DST	不影响
逻辑与	AND DST,SRC	(DST)∧(SRC)→DST	清0 CF和OF，影响SF、ZF及PF，AF不定
测 试	TEST OPR1,OPR2	OPR1∧OPR2	同AND指令
逻辑或	OR DST,SRC	(DST)∨(SRC)→DST	同AND指令
逻辑异或	XOR DST,SRC	(DST)⊻(SRC)→DST	同AND指令

表3-3 逻辑运算指令

NOT指令可用于把操作数的每一位均变反。

AND指令用于把某位清0（与0相与，也可称为屏蔽某位）、某位保持不变（与1相与）。

TEST指令可用于只测试其值而不改变操作数。

OR指令用于把某位置1(与1相或)、某位保持不变（与0相或）。

XOR指令用于把某位变反（与1相异或）、某位保持不变（与0相异或）。

XOR指令的用法：

0⊕0 = 0　1⊕1 = 0

1⊕0 = 1　0⊕1 = 1

X⊕key=X'　X'⊕key=X

1. 逻辑值相同，运算结果为0；

将某个寄存器清零操作；

2. 与1异或，结果相反；

将输入字符进行大小写转换；

加密解密

设数据的原始形式为A，密码为B。加密的操作为：

$$C = A \oplus B$$

加密后的数据为C。在不知道密码的情况下，从C不能推断出原始数据A，从而达到保密的目的。

而知道密码B后，从C可以求出A。如何解密？

不需要借助中间数，使用“异或”运算就能交换两个变量的值？

二、位测试指令

从386开始增加了位测试指令，它们包括BT、BTS、BTR和BTC。这些指令首先把指定位的值送给CF标志，然后对该位按照指令的要求操作。

名称	格 式	功 能	标 志
位测试	BT DST, SRC	测试由SRC指定的DST中的位	所选位值送CF,其它标志不定
位测试并置位	BTS DST, SRC	测试并置1由SRC指定的DST中的位	同上
位测试并复位	BTR DST, SRC	测试并清0由SRC指定的DST中的位	同上
位测试并取反	BTC DST, SRC	测试并取反由SRC指定的DST中的位	同上

表3-4　位测试指令

三、位扫描指令

从386开始增加了位扫描指令，它们包括BSF、BSR指令，可用于扫描操作数中第一个含1的位。

1. 顺向扫描指令 BSF

格式：BSF　DST,RSC

功能：从右向左扫描RSC操作数中第一个含1的位，并把扫描到的第一个含1的位号送DST操作数。若RSC＝0，则DST值不确定。

2. 逆向扫描指令 BSR

格式：BSR　DST,RSC

四、基本移位指令

这类指令实现对操作数移位，包括SHL、SAL、SHR和SAR指令。表3-5给出了这组指令。

名称	格式	功能	标志
逻辑左移	SHL DST, CNT		CF中总是最后移出的一位，ZF、SF、PF按结果设置，当CNT＝ 1时，移位使符号位变化置1 OF*，否则清0
算术左移	SAL DST, CNT		同上
逻辑右移	SHR DST, CNT		同上
算术右移	SAR DST, CNT		同上

注：当CNT＞1时，OF值不确定。
说明：DST可以是8位、16位或32位的寄存器或存储器操作数，CNT是移位位数。
对CNT的限定是：
当CNT＝1时，直接写在指令中；　　适用于8086／8088
当CNT＞1时，由CL寄存器给出；　　适用于80X86系列的所有型号
当CNT＞1时，由指令中的8位立即数给出；　　适用于80286以上
功能图中的符号表示：　CF　数据流向　操作数

表3-5 基本移位指令

可以用逻辑移位指令实现无符号数乘除法运算，只要移出位不含1：

SHL DST,n执行后是原数的2^n倍

SHR DST,n执行后是原数的$1/2^n$

可以用算术移位指令实现带符号数乘除法运算，只要移位操作不改变符号位：

SAL DST,n执行后是原数的2^n倍

SAR DST,n执行后是原数的$1/2^n$

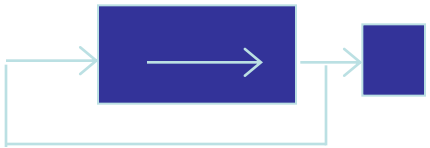
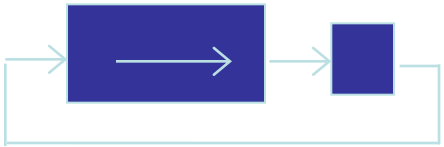
（只要移出位不含1）

例. 设无符号数X在AL中，用移位指令实现X×10的运算。

```
MOV AH,0  ;为了保证不溢出,将AL扩展为字
SHL AX,1  ;求得2X
MOV BX,AX  ;暂存2X
MOV CL,2  ;设置移位次数
SHL AX,CL  ;求得8X
ADD AX,BX  ;10X＝8X＋2X
```

五、循环移位指令

这类指令实现循环移位操作，包括ROL、ROR、RCL、RCR指令。表3-6给出了基本移位指令。

名称	格式	功能	标志
循环左移	ROL DST, CNT		CF中总是最后移进的位，当CNT＝1时，移位使符号位改变则置1 OF*，否则清0，不影响ZF、SF、PF
循环右移	ROR DST, CNT		同上
带进位循环左移	RCL DST, CNT		同上
带进位循环右移	RCR DST, CNT		同上

注：当CNT＞1时，OF值不确定。
说明：对DST和CNT的限定同基本移位指令。

表3-6 基本移位指令

例. 把CX:BX:AX一组寄存器中的48位数据左移一个二进制位。

```
SHL  AX,1
RCL  BX,1
RCL  CX,1
```

在没有溢出的情况下，以上程序实现了2×(CX:BX:AX)→CX:BX:AX的功能。

六、双精度移位指令（386以上）

1．双精度左移指令 SHLD

格式:SHLD　OPRD1,OPRD2,CNT

功能:把操作数OPRD1左移由CNT指定的位（设为n），空出的位用操作数OPRD2高端的n位填充，但OPRD2的内容不变，最后移出的位在进位标志CF中。

2．双精度右移指令 SHRD

格式：SHRD　OPRD1,OPRD2,CNT

3.6 程序控制指令

本节提供的指令可以改变程序执行的顺序，控制程序的流向。它们均不影响标志位。

转移指令；

循环指令；

子程序调用及返回指令；

中断调用及返回指令；

一、转移指令

这类指令包括无条件转移指令和条件转移指令。

1. 无条件转移指令 JMP

格式：JMP　DST

功能：无条件转移到DST所指向的地址

说明：DST为转移的目标地址（或称转向地址），使用与转移地址有关的寻址方式可以形成目标地址。

无条件转移指令 JMP

段内转移（IP）

段内直接短转移　　JMP　SHORT　LABEL

段内直接转移　JMP　NEAR　PTR　LABEL

段内间接转移　JMP　REG / M

段间转移（CS:IP）

段间直接转移　JMP　FAR　PTR　LABEL

段间间接转移　JMP　DWORD　PTR　M

① 段内直接短转移

格式：JMP　SHORT　LABEL

例.

```
        JMP SHORT B1    ;无条件转移到B1标号处
A1:     ADD AX,BX
B1:     ...
```

② 段内直接转移

格式：JMP　LABEL

或：　JMP　NEAR　PTR　LABEL

例.

```
    JMP   B2
A2:     ADD   AX,CX
    ...
B2:     SUB   AX,CX
```

③ 段内间接转移

格式：JMP　REG / M

例.

```
  LEA   BX, B2
  JMP   BX
A2:     ADD   AX, CX
  ...
B2:     SUB   AX, CX
```

动画演示

④ 段间直接转移

例.

```
CODE1       SEGMENT
            ...
            JMP   FAR   PTR
B3
            ...
CODE1       ENDS
CODE2       SEGMENT
            ...
B3:         SUB   AX,BX
            ...
CODE2       ENDS
```

⑤ 段间间接转移

例.

```
V    DD   B3
C1   SEGMENT
     ...
     JMP  DWORD PTR V
     ...
C1   ENDS
C2   SEGMENT
     ...
B3:  SUB  AX,CX
     ...
C2   ENDS
```

动画演示

2. 条件转移指令

执行这类指令时通过检测由前边指令已设置的标志位确定是否转移，所以它们通常是跟在影响标志的指令之后。这类指令本身并不影响标志。

条件转移指令的通用汇编格式：

J_{CC}　　LABEL

J_{CC} LABEL

功能：如果条件为真，则转向标号处，否则顺序执行下一条指令。

说明：其中cc为条件，LABEL是要转向的标号。在8086~80286中，该地址应在与当前IP值的－128~＋127范围之内，即只能使用与转移地址有关的寻址方式的段内短转移格式，其位移量占用一个字节。

(1) 检测单个标志位实现转移的条件转移指令

这组指令根据一个标志位的设置情况决定是否转移。

汇编格式	功　　能	测试条件
JC LABEL	有进位转移	CF＝1
JNC LABEL	无进位转移	CF＝0
JO LABEL	溢出转移	OF＝1
JNO LABEL	无溢出转移	OF＝0
JP/JPE LABEL	偶转移	PF＝1
JNP/JPO LABEL	奇转移	PF＝0
JS LABEL	负数转移	SF＝1
JNS LABEL	非负数转移	SF＝0
JZ/JE LABEL	结果为0/相等转移	ZF＝1
JNZ/JNE LABEL	结果不为0/不相等转移	ZF＝0

注：对实现同一功能但指令助记符有两种形式时，在程序中究竟选用哪一种视习惯或用途而定，例如对于指令JZ/JE　LABEL，当比较两数相等转移时常使用JZ助记符，当比较某数为0转移时常使用JE指令。下同。

表3-7　检测单个条件标志位转移指令

例. 比较AX和BX寄存器中的内容，若相等执行ACT1，不等执行ACT2。

```
CMP   AX,BX
JE    ACT1
ACT2:     .
          .
          .
ACT1:     ...
```

(2) 根据两个带符号数比较结果实现转移的条件转移指令

利用上表中提供的指令，可以实现两个带符号数的比较转移。

汇编格式	功　　能	测试条件
JG/JNLE LABEL	大于/不小于等于转移	ZF＝0　and SF＝OF
JNG/JLE LABEL	不大于/小于等于转移	ZF＝1　or SF≠OF
JL/JNGE LABEL	小于/不大于等于转移	SF≠OF
JNL/JGE LABEL	不小于/大于等于转移	SF＝OF
注1. G＝Greater, L＝Less, E＝Equel, N＝Not 注2. 显然，表3-7中的指令JZ/JE LABEL和 JNZ/JNE LABEL同样可以用于两个带符号数的比较转移。		

表3-8　根据两个带符号数比较结果实现转移的条件转移指令

(3) 根据两个无符号数比较结果实现转移的条件转移指令

利用上表中提供的指令，可以实现两个无符号数的比较转移。

汇编格式	功　能	测试条件
JA/JNBE LABEL	高于/不低于等于转移	CF＝0 and ZF＝0
JNA/JBE LABEL	不高于/低于等于转移	CF＝1 or ZF＝1
JB/JNAE/JC LABEL	低于/不高于等于转移	CF＝1
JNB/JAE/JNC LABEL	不低于/高于等于转移	CF＝0

注1. A＝Above, B＝Below, C＝Carry, E＝Equel, N＝Not可以看出，这里的高于相当于带符号数的大于，低于相当于带符号数的小于。

注2. 显然，表3-7中的指令JZ/JE LABEL和 JNZ/JNE LABEL同样可以用于两个无符号数的比较转移。

表3-9　根据两个无符号数比较结果实现转移的条件转移指令

3. 测试CX / ECX值为0转移指令

这类指令不同于以上介绍的条件转移指令，因为它们测试的是CX或ECX寄存器的内容是否为0，而不是测试标志位。这类指令只能使用与转移地址有关的寻址方式的段内短转移格式，即位移量只能是8位的。

格式： JCXZ LABEL;适用于16位操作数

JECXZ LABEL;适用于32位操作数

功能：测试CX(或ECX)寄存器的内容，当CX（或ECX）＝ 0时则转移，否则顺序执行。

说明：此指令经常用于在循环程序中判断循环计数的情况。

例. 设M＝(EDX:EAX),N＝(EBX:ECX),比较两个64位数，若M＞N，则转向DMAX，否则转向DMIN。

若两数为无符号数
则程序片断为：

```
CMP   EDX,EBX
JA    DMAX
JB    DMIN
CMP   EAX,ECX
JA    DMAX
DMIN:  ...

DMAX: ...
```

若两数为带符号数
则程序片断为：

```
CMP   EDX,EBX
JG    DMAX
JL    DMIN
CMP   EAX,ECX
JA    DMAX
DMIN: ...

DMAX: ...
```

二、循环指令

循环指令可以控制程序的循环。它们的特点是：

1. 用CX或ECX（操作数长度为32位时）作为循环次数计数器。
2. 不影响标志。

1．循环指令 LOOP

格式：LOOP　LABEL

功能：(CX)－1→CX，若(CX)≠0，则转向标号处执行循环体，否则顺序执行下一条指令。

说明：若操作数长度为32位，则其中的CX应为ECX。在LOOP指令前，应先把循环计数的初始值送给CX（或ECX）。

2．相等循环指令 LOOPE/LOOPZ

3．不等循环指令 LOOPNE/LOOPNZ

例. 用累加的方法实现M×N，并把结果保存到RESULT单元。

```
        MOV     AX,0          ;清0累加器
        MOV     BX,M
        CMP     BX,0
        JZ      TERM          ;被乘数为0转
        MOV     CX,N
        JCXZ    TERM          ;乘数为0转
L1:     ADD     AX,BX
        LOOP    L1
TERM:   MOV     RESULT,AX     ;保存结果
```

三、子程序调用与返回指令

子程序的作用；

子程序的执行流程；

主子程序关系示意图

1．子程序调用指令 CALL

格式：CALL　DST

功能：调用子程序。执行时先把返回地址压入堆栈，再形成子程序入口地址，最后把控制权交给子程序。

说明：其中DST为子程序名或子程序入口地址，其目标地址的形成与JMP指令有异同。

与JMP指令相同点：

段内直接／间接调用、

段间直接／间接调用、

只是不能使用段内直接寻址方式的SHORT格式。

指令的执行结果是无条件转到标号处

与JMP指令不同点：

CALL转移后要返回，所以要保存返回地址；

JMP转移后不再返回，所以不必保存返回地址。

(1) 段内调用

这类调用指令实现同一段内的子程序调用，它只改变IP值，不改变CS值。

执行操作：

把返回地址（CALL之后的那条指令地址的偏移量部分(当前IP值)）压入堆栈。

根据与转移地址有关的寻址方式形成子程序入口地址的IP值。

把控制无条件转向子程序，即执行CS:IP处的指令。

① 段内直接调用：

格式：CALL PROCEDURE

或：CALL NEAR PTR PROCEDURE

功能：调用PROCEDURE子程序。执行时先把返回地址压入堆栈，再使IP＝(IP)＋disp16，最后把控制权交给子程序。

② 段内间接调用：

格式：CALL REG／M

功能：调用子程序。执行时先把返回地址压入堆栈，再把指令指定的16位通用寄存器或内存单元的内容送给IP，最后把控制权交给子程序。

例. 可以把子程序入口地址的偏移量送给通用寄存器或内存单元，通过它们实现段内间接调用。

```
                CALL   WORD   PTR   BX
;子程序入口地址的偏移量在BX中
                CALL   WORD   PTR   [BX]
;子程序入口地址的偏移量在数据段的BX所指
;向的内存单元中
                CALL   WORD   PTR   [BX][SI]
;子程序入口地址的偏移量在数据段的BX＋SI
;所指向的内存单元中
```

(2) 段间调用

这类调用指令可以实现段间调用（FAR型调用），执行时即要改变IP值，也要改变CS值。

① 段间直接调用：

格式：CALL FAR PTR PROCEDURE

功能：调用PROCEDURE子程序。执行时先把返回地址(当前IP值和当前CS值)压入堆栈，再把指令中的偏移量部分送给IP，段基址部分送给CS，最后把控制权交给子程序。

② 段间间接调用：

格式：CALL M

功能：调用子程序。执行时先把返回地址(当前IP值和当前CS值)压入堆栈，再把M的低字送给IP，高字送给CS,最后把控制权交给子程序。

例. 若子程序B的入口地址(偏移量和段基址)放在变量VAR中,即可通过VAR实现段间间接调用。如下所示:

```
CALL  DWORD PTR VAR
;从VAR变量中得到子程序B的入口地址实现调用。
```

变量VAR的地址也可以通过寄存器间接寻址方式、基址变址寻址方式等存储器操作数寻址方式得到。

例. `CALL  DWORD PTR 8[BX][DI]`

1．子程序返回指令 RET

执行这组指令可以返回到被调用处。有两条返回指令，它们都不影响标志。以下介绍8086／8088的子程序调用指令。

(1) 返回指令 RET

格式：RET

功能：按照CALL指令入栈的逆序，从栈顶弹出返回地址（弹出一个字到IP，若子程序是FAR型还需再弹出一个字到CS），然后返回到主程序继续执行。

无论子程序是NEAR型还是FAR型，返回指令的汇编格式总是用RET表示。但经汇编后会产生不同的机器码。在DEBUG中，段间返回指令被反汇编成RETF。

(2) 带立即数的返回指令

格式：RET imm_{16}

功能：按照CALL指令入栈的逆序，从栈顶弹出返回地址（弹出一个字到IP，若子程序是FAR型还需再弹出一个字到CS），返回到主程序，并修改栈顶指针SP＝(SP)＋imm_{16}。

注：其中imm_{16}是16位的立即数，设通过堆栈给子程序传递了n个字型参数，则imm_{16}＝2n。

修改堆栈指针是为了废除堆栈中主程序传递给子程序的参数。

四、中断调用与返回指令

中断就是使计算机暂时挂起正在执行的进程而转去处理某种事件，处理完后再恢复执行原进程的过程。

对某事件的处理实际上就是去执行一段例行程序，该程序被称为中断处理例行程序或中断处理子程序，简称为中断子程序。

1. 中断向量

中断向量就是中断处理子程序的入口地址。在PC机中规定中断处理子程序为FAR型，所以每个中断向量占用4个字节，其中低两个字节为中断向量的偏移量部分，高两个字节为中断向量的段基址部分。

2. 中断类型号

IBM PC机共支持256种中断，相应编号为0~255，把这些编号称为中断类型号。

3. 中断向量表

256种中断有256个中断向量。把这些中断向量按照中断类型号由小到大的顺序排列，形成中断向量表。

表长为4×256＝ 1024字节，该表从内存的0000:0000地址开始存放，从内存最低端开始存放。

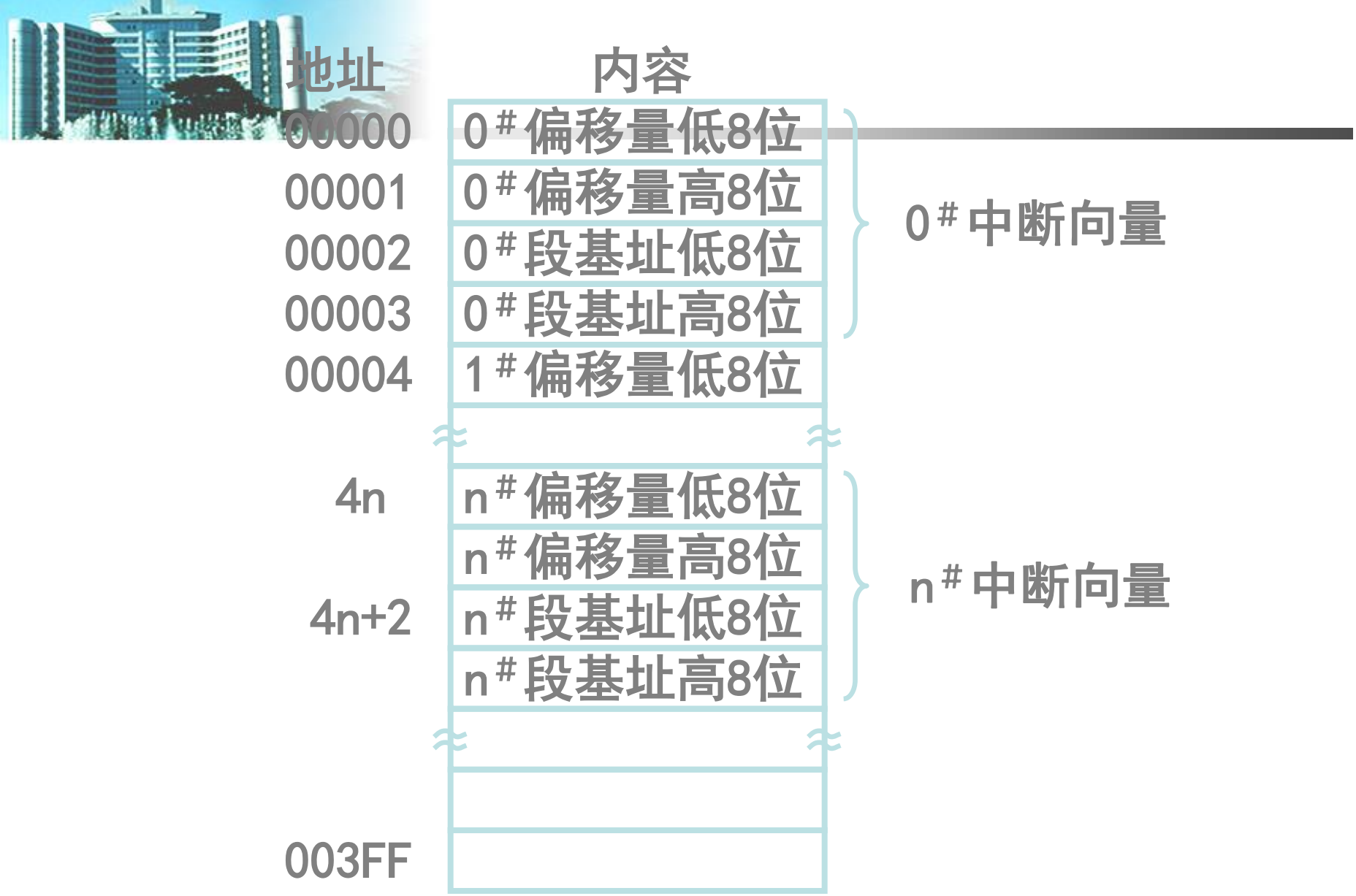


图3-19　中断向量表

4. 中断调用指令 INT

在8086/8088中，中断分为内中断(或称软中断)和外中断(或称硬中断)，本节只介绍内中断的中断调用指令。

格式：INT n ；n为中断类型号

功能：中断当前正在执行的程序，把当前的FLAGS、CS、IP值依次压入堆栈(保护断点)，并从中断向量表的4n处取出n类中断向量.

其中(4n)→IP，(4n+2)→CS，转去执行中断处理子程序。

例. INT 21H

21H为系统功能调用中断，执行时把当前的FLAGS、CS、IP值依次压入堆栈，并从中断向量表的84H处取出21H类中断向量，其中(84H)→IP，(86H)→CS，转去执行中断处理子程序。

动画演示

例. INT 3 ;断点中断

5. 中断返回指令 IRET

格式：IRET

功能：从栈顶弹出三个字分别送入IP、CS、FLAGS寄存器，把控制返回到原断点继续执行。

3.7 处理机控制指令

这组指令可以控制处理机状态以及对某些标志位进行操作。

一、标志操作指令

这组指令可以直接对CF、DF和IF标志位进行操作，它们只影响本指令所涉及的标志。

汇编格式	功　　能	影响标志
CLC(Clear Carry)	把进位标志CF清0	CF
STC(Set Carry)	把进位标志CF置1	CF
CMC(Complement Carry)	把进位标志CF取反	CF
CLD(Clear Direction)	把方向标志DF清0	DF
STD(Set Direction)	把方向标志DF置1	DF
CLI(Clear Interrupt)	把中断允许标志IF清0	IF
STI (Set Interrupt)	把中断允许标志IF置1	IF

表3-10　标志操作指令

二、其它处理机控制指令

这组指令可以控制处理机状态，它们均不影响标志，见下表。

名称	汇编格式	功 能	说 明
空操作	NOP (No Operation)	空操作	CPU不执行任何操作，其机器码占用一字节
停机	HLT(Halt)	使CPU处于停机状态	只有外中断或复位信号才能退出停机，继续执行
等待	WAIT(Wait)	使CPU处于等待状态	等待$\overline{\text{TEST}}$信号有效后，方可退出等待状态
锁定前缀	LOCK(Lock)	使总线锁定信号有效	LOCK是一个单字节前缀，在其后的指令执行期间，维持总线的锁存信号直至该指令执行结束

表3-11 其它处理机控制指令

3.8 串操作指令

利用串操作指令可以直接处理两个存储器操作数，方便地处理字符串或数据块。

一、串指令的特点

1．指令格式

串指令可以显式地带有操作数，也可以使用隐含格式。例如串传送指令MOVS，可以有以下几种格式：

显式：MOVS　DST,SRC

隐式：MOVSB　　;字节传送

　　MOVSW　　;字传送

　MOVSD　　;双字传送

经常使用隐含格式。操作数时应先建立地址指针。

2. 操作数

串指令可以处理寄存器操作数和存储器操作数。

若为寄存器操作数则只能放在累加器中；

对于存储器操作数应先建立地址指针：若为源操作数，则必须把源串首地址放入SI寄存器，缺省情况寻址DS所指向的段，允许使用段超越前缀；若为目标操作数，则必须把目标串首地址放入ES:DI寄存器，不允许使用段超越前缀。

3．地址指针的修改

串指令执行后系统自动修改地址指针SI（ESI）、DI（EDI）。若为字节型操作其修改量为1，若为字型操作其修改量为2,若为双字型操作其修改量为4。

4．方向标志

方向标志DF决定地址指针的增减方向。

DF＝0，则地址指针增量；

DF＝1，则地址指针减量。

可以用CLD和STD指令复位和置位DF。

5. 重复前缀

串指令前可以加重复前缀REPE / REPZ、 REP或REPNE / REPNZ,使后跟的串指令重复执行。重复次数应事先初始化在计数器CX中。

(1) 重复前缀 REP

REP可用在传送类串操作指令之前，其执行的操作为：

若(CX)＝0，则结束重复，顺序执行下一条指令。

若(CX)≠0，则执行后跟的串操作指令，然后修改计数器(CX)－1→CX，继续重复上述操作。

若地址长度为32位的，则使用ECX。

(2) 相等重复前缀 REPE / REPZ

REPE / REPZ可以用在比较类串操作指令之前，当比较相等且重复次数未到时重复执行后跟的串指令。其执行的操作为：

若(CX)＝0（计数到）或ZF＝0（不相等），则结束重复，顺序执行下一条指令。

否则执行后跟的串操作指令，修改计数器(CX)－1→CX。继续重复上述操作。

若地址长度为32位的，则使用ECX。

(3) 不等重复前缀 REPNE / REPNZ

二、串指令

1. 串传送指令 MOVS

显式格式：MOVS DST_m, SRC_m

隐含格式：MOVSB　　MOVSW　　MOVSD

功能：源→目标，即([SI])→ES:[DI]，且自动修改SI、DI指针。

标志：不影响标志位。

说明：若DF＝0，则MOVSB指令使SI、DI各加1；MOVSW指令使SI、DI各加2；MOVSD使SI、DI各加4。若DF＝1，则

MOVSB使SI、DI各减1；

MOVSW使SI、DI各减2；

MOVSD使SI、DI各减4。

若地址长度是32位的，则SI、DI相应为ESI、EDI。

2. 取串指令 LODS

显式格式：LODS　SRC_m

隐含格式：LODSB　LODSW　LODSD

功能：源→累加器，即([SI])→AL（或AX、EAX），且自动修改SI指针。

说明：若DF＝0,则LODSB(或LODSW、LODSD)使SI加1（或2、4）；若DF＝1，则LODSB（或LODSW、LODSD）使SI减1（或2、4）。若地址长度是32位的，则SI相应为ESI。

标志：不影响标志位。

3. 存串指令 STOS

显式格式：STOS　DST_m

隐含格式：STOSB　STOSW　STOSD

功能：累加器→目标，即(AL(或AX、EAX))

→ES:[DI]，且自动修改DI指针。

说明：若DF＝0,则STOSB(或STOSW、STOSD)使DI加1（或2、4）；若DF＝1，则STOSB（或STOSW、STOSD）使DI减1(或2、4)。若地址长度是32位的，则DI相应为EDI。

标志：不影响标志位。

4. 串输入指令 INS

显式格式：INS DST$_m$,DX

隐含格式：INSB INSW INSD

5. 串输出指令 OUTS

显式格式：OUTS DX,SRC$_m$

6. 串比较指令 CMPS

显式格式：CMPS DST$_m$,SRC$_m$

7. 串扫描指令 SCAS

显式格式：SCAS DST$_m$

例：把自AREA1开始的100个字数据传送到AREA2开始的区域中。

```
MOV     AX,SEG   AREA1
MOV     DS,AX
MOV     AX,SEG   AREA2
MOV     ES,AX
LEA     SI,AREA1
LEA     DI,AREA2
MOV     CX,100
CLD
REP     MOVSW
;100个字数据传送完毕后执行下一条指令
```

动画演示

例：把自NUM1开始的未压缩BCD码字符串转换成ASCII码，并放到NUM2中，字符串长度为8字节。设DS、ES已按要求设置。

```
        LEA     SI,NUM1
        LEA     DI,NUM2
        MOV     CX,8
        CLD
LOP:    LODSB
        OR      AL,30H
        STOSB
        LOOP    LOP
```

3.9 条件字节设置指令

80386以上的机器提供了一组条件字节设置指令，这些指令根据前边指令对标志位的设置而置1或清0目标字节。

格式：SET_{CC} DST

功能：如果条件为真，则置1 DST，否则清0 DST。

说明：cc表示条件，与条件转移指令中的含义相同。DST只能是8位寄存器或存储器操作数。

标志：所有条件字节设置指令均不影响标志。

条件字节设置指令见下表。

指令格式	功　　能	测试条件
SETZ/SETE　DST	等于0/相等时置DST为1	ZF＝1
SETNZ/SETNE　DST	不等于0/不相等时置DST为1	ZF＝0
SETS　DST	为负时置DST为1	SF＝1
SETNS　DST	非负时置DST为1	SF＝0
SETO　DST	溢出时置DST为1	OF＝1
SETNO　DST	无溢出时置DST为1	OF＝0
SETP/SETPE　DST	结果低8位有偶数个1时置DST为1	PF＝1
SETNP/SETPO　DST	结果低8位有奇数个1时置DST为1	PF＝0
SETG/SETNLE　DST	大于/不小于等于时置DST为1	ZF＝0 and SF＝OF
SETNG/SETLE　DST	不大于/小于等于时置DST为1	ZF＝1 or SF≠OF
SETL/SETNGE　DST	小于/不大于等于时置DST为1	SF≠OF
SETNL/SETGE　DST	不小于/大于等于时置DST为1	SF＝OF
SETA/SETNBE　DST	高于/不低于等于时置DST为1	CF＝0 and ZF＝0
SETNA/SETBE　DST	不高于/低于等于时置DST为1	CF＝1 or ZF＝1
SETB/SETNAE/SETC　DST	低于/不高于等于/有进位时DST置1	CF＝1
SETNB/SETAE/SETNC DST	不低于/高于等于/无进位时DST置1	CF＝0

条件字节设置指令

感谢关注聆听！

张华平
Email: kevinzhang@bit.edu.cn
微博：@ICTCLAS张华平博士
实验室官网：
http://www.nlpir.org

大数据千人会