第3章
指令系统（Ⅲ）

3.5 逻辑运算和移位指令

逻辑运算		算术逻辑移位	
NOT	取反	SHL/SAL	逻辑/算术左移
AND	逻辑乘（与）	SHR	逻辑右移
OR	逻辑加（或）	SAR	算术右移
XOR	异或	循环移位	
TEST	测试	ROL	循环左移
		ROR	循环右移
		RCL	通过进位的循环左移
		RCR	通过进位的循环右移

双操作数指令

3.5.1 单操作数逻辑指令

1、NOT取反指令

- 指令格式：NOT　目的
- 指令功能：目的←目的取反
- 操作数要求：

①可以是8位或16位寄存器、存储器。

②对于存储器操作，需指明是字还是字节。

注意：指令执行后，对标志位无影响。

【例】NOT AL　　；若(AL) = 10001111B,

2、移位指令SHL/SAL/SHR/SAR

这类指令可对寄存器、存储器中的字或字节的各位进行算术移位或逻辑移位，移位的次数由指令中的计数值决定。

- 无符号数×2可使用SHL指令，左移1位；
- 无符号数÷2可使用SHR指令，右移1位；
- 符号数×2可使用SAL指令，左移1位；
- 符号数÷2可使用SAR指令，右移1位。

SAL算术左移指令(Shift Arithmetic Left)

❑ 指令格式：SAL 目的，计数值

SHL逻辑左移指令(Shift Logic Left)

❑ 指令格式：SHL 目的，计数值

指令功能：以上两条指令的功能完全相同，均将寄存器或存储器中的目的操作数的各位左移；

每移一次，最低有效位LSB补0，而最高有效位MSB进入标志位CF。

移动一次，相当于将目的操作数乘以2。

注意：

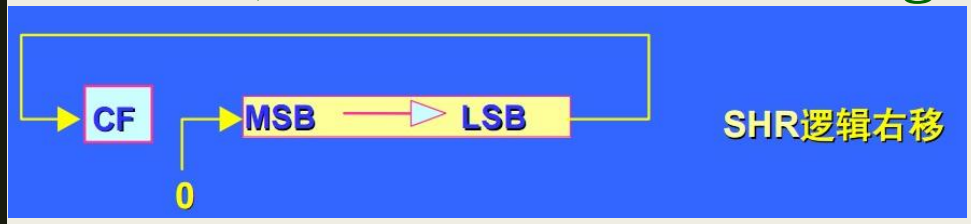
移动一次，相当于将目的操作数乘以2；计数值就是所要移位的次数。

- 若移位一次，直接将计数值置1；
- 移位次数大于1，将移位次数送进CL，再把CL放在计数值位置上。

对标志位的影响：

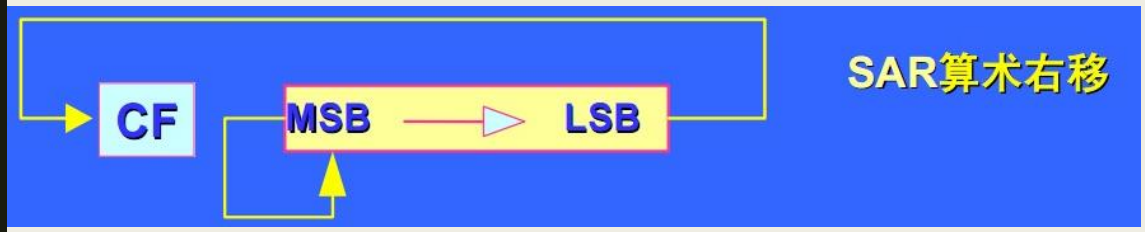
- 移位次数为1时，若移位后最高位的值被改变，OF置1，否则OF清0。
- 多次移位时，OF的值不确定。
- 不论移多少次，CF总是等于目的操作数最后被移出去的值。SF和ZF 将根据指令执行后目的操作数的状态来决定，PF只有当目的操作数在AL中时才有效，AF不定。

逻辑右移指令（Shift Logic Right）



- 指令格式：SHR　目的，计数值
- 指令功能：各位进行右移。每移一次，低位进入CF，最高位补0 。右移次数由计数值决定。
- 注意：若目的操作数为无符号数，每右移一次，目的操作数除以2，余数被丢掉。

SAR算术右移指令(Shift Arithmetic Right)



- 指令格式：SAR　目的，计数值
- 指令功能：各位右移。每移位一次，最低位进入CF，但最高位(即符号位)保持不变。
- 注意：

每移一次，相当于对带符号数进行除2操作。

【例】若AH和AL分别存有分离的BCD数，则用下列指令可转换成组合的BCD数，并存于AL中

解：
```
MOV   CL，4
SHL   AL，CL
SHR   AX，CL
```

3、循环移位指令

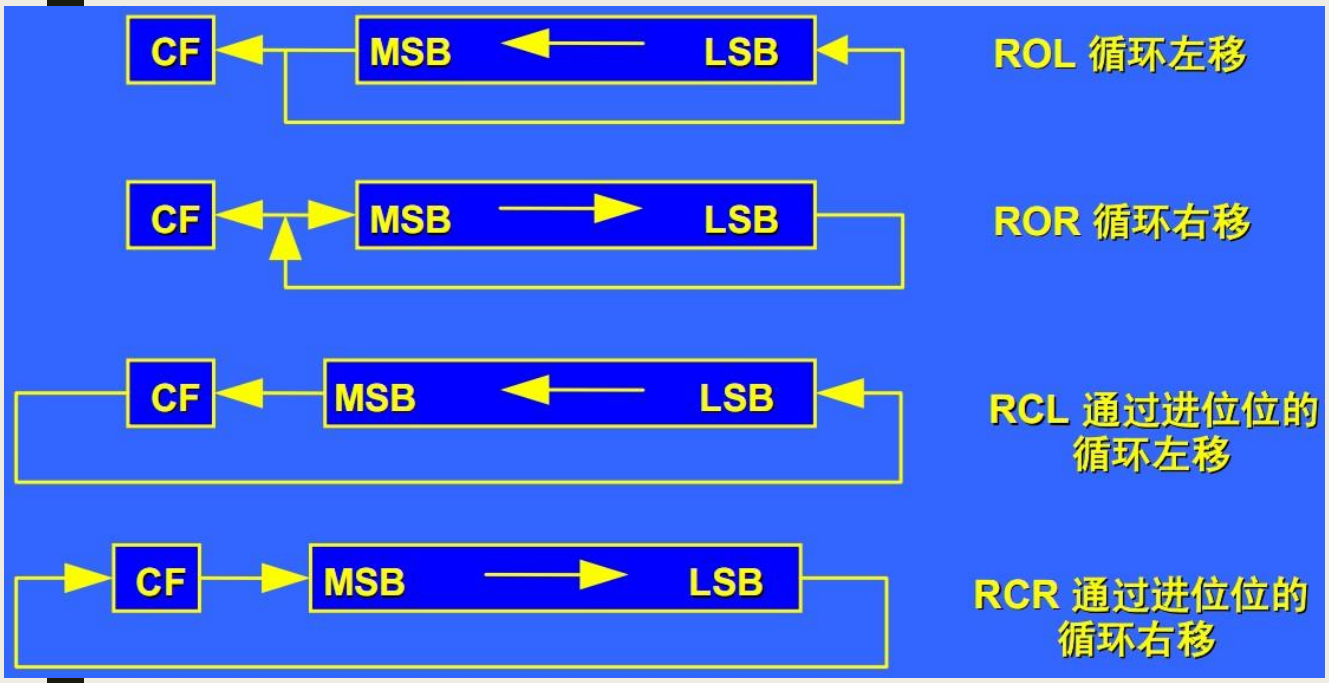
算术逻辑移位指令，移出操作数的数位均被丢失；循环移位指令把操作数从一端移到另一端，移出的位不丢失。循环移位指令共4条：

ROL循环左移 (Rotate Left)、ROR循环右移(Rotate Right)、RCL通过进位位循环左移 (Rotate through Carry Left)、RCR通过进位位循环右移 (Rotate through Carry Right)指令

指令格式：ROL/ROR/RCL/RCR　目的，计数值

- 循环移位后：结果→目的操作数
- 目的操作数可以是8/16位的寄存器、存储器，移位的次数可以是1，或由CL寄存器的值指定。

关于标志位：



①ROL和ROR指令没有把进位标志CF包含在循环中；而RCL和RCR指令把CF作为整个循环的一部分，一起参加循环移位。

②CF的值总是由最后一次被移出的值决定。

③OF位只有在移位次数为1的时候才有效。在移位后最高有效位发生变化(由1变0或由0变1)时，则OF置1，否则OF置0。在多位循环移位时，OF值不确定。

【课后习题10】若有一个四字节数，放在寄存器DX与AX中(DX中放高16位)，要求这个4字节数整个左移一位如何实现？右移一位又如何实现？

逻辑变量的表示和运算

A	B	A ∧ B	A ∨ B	A⊕B	$\bar{A}$
0	0	0	0	0	1
0	1	0	1	1	
1	0	0	1	1	0
1	1	1	1	0	

3.5.2 双操作数逻辑指令

1、AND逻辑与指令 (Logical AND)

指令格式：AND 目的，源

指令功能：目的 ← 目的 ∧ 源

主要用于使操作数的某些位保留 (和“1”相与)，而使某些位清除(和“0”相与)。

2、OR逻辑或指令 （Logical OR）

指令格式：OR 目的，源

指令功能：目的 ← 目的 ∨ 源

主要用于使操作数的某些位保留(和“0”相或)，而使某些位置位(和“1”相或)。

【例】

```
MOV   AL,  48H
MOV  AH,  AL
AND  AL,  0FH
MOV  CL,  4
SHR  AH,  CL
MOV   NUM,  AX
```

【例】试分析下面程序完成什么任务？设(DL)=12H，(BL)=34H

```
MOV  CL,  4
SHL  DL, CL
SHR  BL,  CL
OR  DL,  BL
```

【课后习题3】AX中有一个负数，欲求其绝对值，若该数为补码，则用什么指令？若该数为原码，又用什么指令？

3、XOR异或操作指令（Exclusive OR）

- 指令格式：XOR　目的，源
- 指令功能：目的 ← 目的 ∨ 源

主要用于使操作数的某些位保留（和“0”相异或），而使某些位取反（和“1”相异或）

【例】

```
XOR AX，AX          ；清累加器
```

【课后习题6】若CX=6700H，DX=78FFH，CF=1。求分别执行指令后，CX与DX寄存器中的内容，并指出

(1)ADD CX，DX
(2)ADC CX，DX
(3)SUB CX，DX
(4)SBB CX，DX
(5)AND CX，DX
(6)OR CX，DX
(7)XOR CX，DX

【课后习题16】编一程序使：

（1）AX寄存器低4位清零
（2）BX寄存器低4位置1
（3）CX寄存器低4位变反
（4）DX寄存器高3位不变，其余位清零

4、TEST测试指令（Test）

指令格式：TEST　目的，源

指令功能：目的∧源

逻辑与操作，并修改标志位，**但不回送结果，**两个操作数都不变。

常用于在要检测某些条件是否满足，但又不希望改变原有操作数的情况下。紧跟在这条指令后面的往往是一条**条件转移**指令，根据测试结果产生分支，转向不同的处理程序。

- 源操作数可以是8位或16位立即数、寄存器、存储器；
- 目的操作数只能是寄存器、存储器，两操作数**不能同时为存储器**；
- 指令执行后，均将CF和OF清零，ZF、SF和PF反映操作结果，AF未定义。

【课后习题19】

```
MOV AX，2280H
MOV CX，0FF00H
MOV DS，AX
MOV SI，CX
ADD CX，AX
MOV [SI]，CX
ADC[SI]，AL
DEC BYTE [SI]
MOV AX，06H
ADC AX，08H
AAA
ADD AL，59H
```

```
DAA
AND AL，0FH
MOV BX，-8
NEG BX
MOV DL，06H
MUL DL
OR AX，0FF00H
CWD
IDIV BX
MOV AH，4CH
INT 21
```

3.6 字符串操作指令

串操作

- 字符串：一系列存放在存储器中的字或字节数据，不管它们是不是ASCII码。字符串长度可达64K字节。
- 字符串元素：组成字符串的字节或字。每种字符串指令对字符串元素只进行同一种操作。
- 字符串操作指令：对字符串进行的传送、比较、搜索、装入及填充等5种操作。

串操作指令就是用一条指令实现对一串字符数据的操作；是唯一的一组源操作数和目的操作数都在存储单元的指令。

字符串操作指令的类型和格式

指令名称		字节操作	字操作
字符串传送 MOVS	目的串，源串	MOVSB	MOVSW
字符串比较 CMPS	目的串，源串	CMPSB	CMPSW
字符串搜索 SCAS	目的串	SCASB	SCASW
字符串装入 LODS	源串	LODSB	LODSW
字符串填充 STOS	目的串	STOSB	STOSW

每种指令都有3种格式。有两种方法用以说明是字节操作还是字操作。

- 方法一：用指令中的源串和目的串名(即操作数)来表明是字节还是字；
- 方法二：在指令助记符后加B说明是字节，加W说明是字操作。

❖ 串操作指令的隐含约定：

①源串：用寄存器SI指向源串偏移首地址，DS作为源串的段地址，即DS：SI 。源串允许使用段超越前缀来修改段地址。

②目的串：用寄存器DI指向目的串偏移首地址，ES作为目的串段地址，即ES：DI。目的串不允许使用段超越前缀修改ES。如果要在同一段内进行串运算，必须使DS和ES指向同一段。

③指针：每执行一次字符串指令，指针SI和DI会自动修改，以指向下一个待操作单元。

④用方向标志位DF的值决定串处理方向；

DF＝0为递增方向。DS：SI指向源串首地址；

DF＝1为递减方向。DS：SI指向源串末地址。

⑤字节串操作时，地址指针SI/DI分别加/减1

字串操作时，地址指针SI/DI分别加/减2；

STD使DF置1，CLD将DF清0。

⑥串长度：在串操作指令前面可以加重复前缀指令前缀REP、REPZ/REPE、REPNZ/REPNE，能重复执行该指令。固定用**CX指定字符串个数**。

关于重复前缀:

在基本指令前加重复前缀，可加快串运算指令的执行速度。每重复执行一次，地址指针SI和DI按方向标志自动修改，CX的值自动减1。

类型与格式:

- REP：无条件重复(Repeat)，常与(MOVS)连用，连续传送字符串。直到传送完毕，即CX＝0为止。

- REPE/REPZ：相等/结果为零则重复(Repeat while Equal/Zero)，常与(CMPS)连用，连续比较字符串。当两个字符串对应字符相等(ZF＝1)和CX≠0时，则重复进行比较，直到ZF＝0或CX＝0为止。
- REPNE/REPNZ：不相等/结果非零则重复(Repeat while Not Equal/Not Zero)，常与(SCAS)连用，当结果非0(ZF＝0)和CX≠0时，重复进行扫描，直到ZF＝1或CX＝0为止。
- 带有重复前缀的串指令执行过程中允许有中断进入。

1．MOVS字符串传送指令（Move String）

- 指令格式：MOVS　目的串，源串
- 指令功能：将DS段SI寻址的一个字节或字传送到ES段DI寻址的单元中，并修改SI和DI使指向下一个元素。
- 应用：
  - 解决MOV指令不能直接在存储单元间进行数据传送的问题。
  - 若使用重复前缀，还可以利用一条指令传送一批数据。

【课堂练习：课后习题11】

若把1K字节的数据块从偏移地址为1000H开始的单元传送到偏移地址为1400H开始的缓冲区。试用串传送操作指令和一般传送指令两种方法各编一程序实现。

2、CMPS字符串比较指令（Compare Sting）

- 指令格式：CMPS　目的串，源串
- 指令功能：把DS段由SI作指针的源串减去ES段由DI作指针的目的串数据，结果反映在标志位上；两个数据串的原始值不变；源串和目的串指针自动修改，指向下一对待比较的串。
- 加重复前缀：

▲REPE/REPZ　CMPS；直至CX＝0(比完了)或ZF＝0(两串不相等)时停止操作。

▲REPNE/REPNZ CMP；直至CX＝0(比完了)或ZF＝1(两串相等)时停止比较。

3、SCAS字符串扫描指令（Scan String）

- 指令格式：SCAS　目的串
- 指令功能：AL/AX(字节/字操作)内容减去ES:DI中的串元素，结果反映在标志位上，源操作数不变，操作后目的串指针会自动修改，指向下一个待搜索的串元素。
- 应用：
  - 利用SCAS指令，可在内存中搜索关键字。指令执行前，必须事先将关键字存在AL(字节)或AX(字)中，才能用SCAS指令进行搜索。
  - SCAS指令可以加重复前缀。

4、LODS数据串装入指令（Load String）

- 指令格式：LOAD 源串
- 指令功能：DS:SI中的串元素加载到AL/AX(字节/字操作)中，修改SI，以指向串中的下一个元素，修改量遵守隐含约定(4)。
- 注：

该指令加重复前缀意义不大，因为重复传送只能保留最后写入的数据。

5、STOS数据串填充指令（Store string）

- 指令格式：STOS　目的串
- 指令功能：AL/AX(字节/字操作)的一个字节或字传送到ES:DI所指的目的串；修改DI，以指向串中的下一个单元。
- 关于重复前缀：

“REP STOS”，可用累加器中的常数，对数据串初始化。如初始化为全0串

3.7 程序控制类指令

无条件转移和过程调用指令	
JMP	无条件转移
CALL	过程调用
RET	过程返回
条件转移	
JZ / JE (10条指令)	直接标志转移
JA / JNBE (8条指令)	间接标志转移
条件循环控制	
LOOP	CX≠0 则循环
LOOPE / LOOPZ	CX≠0和ZF=1 则循环
LOOPNE / LOOPNZ	CX≠0和ZF=0 则循环
JCXZ	CX=0 则转移
中 断	
INT	中断
INTO	溢出中断
IRET	中断返回

通过改变IP和CS值，实现程序执行顺序的改变。

3.7.1 调用、转移与返回指令

1、JMP无条件转移指令(Jump)

- 指令格式：JMP　目的
- 指令功能：程序无条件地转移到指定的目的地址去执行。
- 两种转移类型：段内转移或近(NEAR)转移

转移指令的目的地址和JMP指令在同一代码段中，转移时，仅改变IP寄存器的内容，段地址CS的值不变。

段间转移，又称为远(FAR)转移

转移指令的目的地址和JMP指令不在同一段中，转移时，CS和IP的值都要改变，程序转到另一代码段去执行。

❖ 与转移地址有关的4种寻址方式

类型	方式	寻址目标		指令举例
段内转移	直接	立即短转移（8位）	①	JMP SHORT PROG_S
	直接	立即近转移（16位）		JMP NEAR PTR ROG_N
	间接	寄存器（16位）	②	JMP BX
	间接	存储器（16位）		JMP WORD PTR 5[BX]
段间转移	直接	立即转移（32位）	③	JMP FAR PTR PROG_F
	间接	存储器（32位）	④	JMP DWORD PTR [DI]

①段内直接寻址：转移的偏移地址是当前IP值和指令中指定的8位或16位位移量之和；

②段内间接寻址：转移的偏移地址是一个寄存器或一个存储单元的内容；

③段间直接寻址：指令中直接提供了转向的段地址和偏移地址；

④段间间接寻址：用存储器中的两个相继字内容取代IP和CS中的内容；

2、调用指令(CALL)

CALL指令用来调用一个过程或子程序。由于过程或子程序有段间(即远程**FAR**)和段内调用(即近程**NEAR**)之分。

近过程调用：调用指令CALL和被调用的过程在同一代码段中。

远过程调用：两者在不同代码段中。

近调用的操作：SP←SP-2，IP入栈；

远调用的操作：SP←SP-2，CS入栈；SP←SP-2，IP入栈

【课后习题14】若DS=1100H，SI=1000H，CS=1200H，IP=100H，[12000H]=2000H，[12002H]=1500H，求执行指令

(1)CALL FAR[SI]后CS与IP的内容；

(2)CALL [SI]后CS与IP的内容。

3、RET指令

从栈中弹出返回地址，使程序返回主程序继续执行。

这里分为两种情况：

如果从**近过程**返回：

从栈中弹出一个字→IP，并且使SP←SP+2

如果从**远过程**返回：

先从栈中弹出一个字→IP，并且使SP←SP+2；

再从个字栈中弹出一个字→CS，并使SP←SP+2。

3.7.2 条件转移指令

根据上一条指令执行后的状态标志作为判别测试条件来决定是否转移。

注意:

条件转移均为段内短转移。

目的地址＝当前IP值＋8位相对位移量。

它与转移指令后那条指令的距离，允许为-128~+127 字节;

8位偏移量用符号扩展法扩展到16位与IP相加。

指令格式: 条件操作符 标号

条件转移指令共有18条，可以为两类:

直接标志转移指令

间接标志转移指令

1、直接标志转移指令(10条)

以CF，ZF，SF，OF和PF等5个标志的10种状态为判断的条件。

指令助记符	测试条件	指令功能
JC	CF=1	有进位 转移
JNC	CF=0	无进位 转移
JZ/JE	ZF=1	结果为零/相等 转移
JNZ/JNE	ZF=0	不为零/相等 转移
JS	SF=1	符号为负 转移
JNS	SF=0	符号为正 转移
JO	OF=1	溢出 转移
JNO	OF=0	无溢出 转移
JP/JPE	PF=1	奇偶位为1/为偶 转移
JNP/JPO	PF=0	奇偶位为0/为奇 转移

2、间接标志转移

指令的助记符中不直接给出标志状态位的测试条件，而是标志的状态组合作为测试的条件。

指令应用：通常放在比较指令CMP之后，以比较两个数的大小。

- 无符号数比较测试指令中，指令助记符中的“A”是英文Above的缩写，表示“高于”之意，“B”是英文Below的缩写，表示“低于”之意；
- 带符号数比较测试指令中，指令助记符中的“G”(Great than)表示大于，“L”(Less than)表示小于。

2、间接标志转移

类别	指令助记符	测试条件	指令功能
无符号数比较测试	**JA/JNBE**	CF ∨ ZF =0	高于/不低于等于　转移
	JAE/JNB	CF=0	高于等于/不低于　转移
	JB/JNAE	CF=1	低于/不高于等于　转移
	JBE/JNA	CF ∨ ZF=1	低于等于/不高于　转移
带符号数比较测试	**JG/JNLE**	(SF ⊕ OF) ∨ZF=0	大于/不小于等于　转移
	JGE/JNL	SF ⊕ OF =0	大于等于/不小于　转移
	JL/JNGE	SF ⊕ OF =1	小于/不大于等于　转移
	JLE/JNG	(SF ⊕ OF) ∨ ZF=1	小于等于/不大于　转移

3.7.3 循环指令

特点：

- 是一组增强型的条件转移指令，控制程序段的重复执行，重复次数存于CX寄存器中。
- 指令的字节数均为2，第一字节是操作码，第二字节是8位偏移量，转移的目标都是短标号。
- 循环指令中的偏移量都是负值。
- 循环控制指令均不影响任何标志。

这类指令共有4条。

1、LOOP循环指令(Loop)

- 指令格式：LOOP 短标号
- 指令功能：控制重复执行一系列指令，重复次数放在CX寄存器中，每执行一次LOOP指令，CX自动减1。如果减1后CX≠0，则转移到指令中所给定的标号处继续循环；若自动减1后CX＝0，则结束循环。
- 一条LOOP指令相当于执行以下两条指令的功能：

```
DEC  CX
JNZ  标号
```

2、 LOOPE/LOOPZ相等或结果为零时循环 (Loop If Equal / Zero)

- 指令格式：LOOPE 标号 或 LOOPZ 标号
- 指令功能：

LOOPE是相等时循环；

LOOPZ是结果为零时循环；

指令执行前，先将重复次数送到CX中，每执行一次指令，CX 自动减1，若**减1后CX≠0和ZF＝1**，则转到指令所指定的标号处**重复执行**；若**CX＝0或ZF＝0**，便**退出循环**。

3、LOOPNE/LOOPNZ不相等或结果不为零循环(Loop If Not Equal/Not Zero)

- 指令格式：LOOPNE 标号 或 LOOPNZ 标号
- 指令功能：

LOOPNE是不相等时循环

LOOPNZ是结果不为零循环

指令执行前，应将重复次数送入CX，每执行一次，CX自动减1，若**减1后CX≠0和ZF＝0**，则转移到标号所指定的地方**重复执行**；若**CX＝0或ZF＝1**，则退出循环，顺序**执行下一条指令**。

4、JCXZ若CX为0跳转(Jump If CX Zero)

- 指令格式：JCXZ　标号
- 指令功能：

若CX寄存器为零，则转移到指令中标号所指定的地址处，它不对CX寄存器进行自动减1的操作。

主要用在循环程序开始处，为跳过循环，把CX寄存器清零。

中断概念

- 定义：计算机暂时中止当前程序的运行，执行中断服务程序去为临时发生的事件服务，执行完毕后，返回正常程序继续运行，这个过程称为中断。

3.7.4 中断控制指令

中断过程:

- 断点(CS，IP值)、标志寄存器的值入栈保护
- 取入口地址(中断类型×4得到中断向量的地址)
- 执行中断服务程序
- 执行中断返回指令IRET，从堆栈中恢复中断前CPU的状态和断点。

中断向量表:

- 低2字节存放中断服务程序入口地址的偏移量IP;
- 高2字节存放中断服务程序入口地址的段地址CS; 中断服务程序入口地址＝中断类型×4;

1、软件中断指令(Interrupt)

INT n

也称陷阱中断。n为中断类型号，范围0～255。

CPU执行INT n指令的过程：

- 标志寄存器的内容入栈;
- 当前断点的段基地址CS、偏移地址IP入栈;
- 清除中断标志IF、单步标志TF ；
- 将中断类型号n乘以4，找到中断服务程序的入口地址表的表头地址，获得中断服务程序的入口地址，置入CS和IP寄存器转到相应中断服务程序。

2、溢出中断指令(Interrupt On overflow)

INTO

- 带符号数进行算术运算时，若OF＝1，则溢出中断指令INTO产生类型为4的中断。
- 可在带符号加减法运算之后安排一条INTO指令。

3、中断返回指令(Interrupt Return)

IRET

实现中断返回。

安排在中断服务程序的出口处。

执行后：

- 首先从堆栈中依次弹出程序断点，送到IP和CS寄存器中；
- 接着弹出标志寄存器的内容，送回标志寄存器；
- CPU按CS：IP的值返回断点，继续执行原来被中断的程序。

【课后习题17】下面程序段，在什么情况下执行结果是AL = 0?

```
BEGIN:    IN AL，5FH
          TEST AL，80H
          JZ BRCH1
          XOR AX，AX
          JMP STOP
BRCHI:    MOV AL，0FFH
STOP:     MOV AH，4CH
          INT 21H
```

3.8 CPU控制指令

1、标志操作指令(CF、DF、IF位操作)

指令助记符	操作	指令名称
CLC	CF←0	进位标志清0 （Clear Crray）
CMC	CF←/CF	进位标志求反 (Complement Carry)
STC	CF←1	进位标志置1 （Set Carry）
CLD	DF←0	方向标志清0 （Clear Direction）
STD	DF←1	方向标志置1 （Set Direction）
CLI	IF←0	中断标志清0 （Clear Interrupt）
STI	IF←1	中断标志置1 （Set Interrupt）

2、NOP空操作或无操作指令(No Operation)

单字节指令，执行时需耗费3个时钟周期的时间，但不完成任何操作。

常在循环等操作中增加延时，或在调试程序时使用空操作指令。

3、HLT处理器暂停(Processor Halt)

CPU进入暂停状态，不进行任何操作。

下列情况发生时，CPU才脱离暂停状态：

- 在RESET线上加复位信号；
- 在NMI引脚上出现中断请求信号；
- 在允许中断的情况下，在INTR引脚上出现中断请求信号；

4．WAIT等待指令(Wait)

通常用在CPU执行ESC指令后，表示8086CPU正处于等待状态，它不断检测8086的测试引脚TEST，每隔5个时钟周期检测一次，若此脚为高电平，则重复执行WAIT指令，处理器处于等待状态。一旦TEST引脚的信号变为低电平，便退出等待状态，执行下条指令。

5、ESC换码指令(Escape)

- 指令格式：ESC 外部操作码，源操作数
- 指令功能：换码指令用来实现8086对8087协处理器的控制。

6、LOCK封锁总线指令(Lock Bus)

是一种前缀，加在指令的前端，用来维持8086的总线封锁信号LOCK有效，带有LOCK前缀的指令在执行过程中，禁止其它协处理器使用总线。

以上三个是使CPU与其它协处理器同步工作的指令，用于多处理机；指令执行后均不影响标志位。