

零、状态寄存器

1	<b>MOVE</b>	<p>数据传送指令 MOV</p> <p>格式：MOV OPRD1, OPRD2</p> <p>功能：本指令将一个源操作数送到目的操作数中, 即<math>OPRD1 \leftarrow OPRD2</math>.</p>	<p>说明：</p> <p>1. OPRD1 为目的操作数, 可以是寄存器、存储器、累加器. OPRD2 为源操作数, 可以是寄存器、存储器、累加器和立即数.</p> <p>2. MOV 指令以分为以下四种情况： &lt;1&gt; 寄存器与寄存器之间的数据传送指令 &lt;2&gt; 立即数到通用寄存器数据传送指令 &lt;3&gt; 寄存器与存储器之间的数据传送指令 &lt;4&gt; 立即数到存储器的数据传送</p> <p>3. 本指令不影响状态标志位</p>
2	<b>PUSH</b>	<p>堆栈操作指令 PUSH和POP</p> <p>格式：PUSH OPRD</p> <p>POP OPRD</p> <p>功能：实现压入操作的指令是PUSH指令;实现弹出的指令是POP指令.</p>	<p>说明：</p> <p>1. OPRD为16位(字)操作数, 可以是寄存器或存储器操作数.</p> <p>2. PUSH的操作过程是：<math>(SP) \leftarrow (SP) - 2</math>, <math>((sp)) \leftarrow OPRD</math> 即先修改堆栈指针SP(压入时为自动减2), 然后, 将指定的操作数送入新的栈顶位置.</p> <p>此处的 <math>((SP)) \leftarrow OPRD</math>, 也可以理解为：<math>[(SS) * 16 + (SP)] \leftarrow OPRD</math> 或 <math>[SS:SP] \leftarrow OPRD</math></p>
3	<b>POP</b>	<p>堆栈操作指令 PUSH和POP</p> <p>格式：PUSH OPRD</p> <p>POP OPRD</p> <p>功能：实现压入操作的指令是PUSH指令;实现弹出的指令是POP指令.</p>	<p>说明：</p> <p>1. OPRD为16位(字)操作数, 可以是寄存器或存储器操作数.</p> <p>2. POP指令的操作过程是：POP OPRD: <math>OPRD \leftarrow ((SP))</math>, <math>(SP) \leftarrow (SP) + 2</math></p> <p>它与压入操作相反, 是先弹出栈顶的数项, 然后再修改指针SP的内容.</p> <p>3. 示例：POP AX POP DS POP DATA1 POP ALFA[BX][DI]</p> <p>4. PUSH和POP指令对状态标志位没有影响.</p>
4	<b>XCHG</b>	<p>数据交换指令 XCHG</p> <p>格式：XCHG OPRD1, OPRD2 其中的OPRD1为目的操作数, OPRD2为源操作数</p> <p>功能：将两个操作数相互交换位置, 该指令把源操作数OPRD2与目的操数OPRD1交换.</p>	<p>说明：</p> <p>1. OPRD1及OPRD2可为通用寄存器或存储器, 但是两个存储器之间是不能用XCHG指令实现的.</p> <p>2. 段寄存器内容不能用XCHG指令来交换.</p> <p>3. 若要实现两个存储器操作数DATA1及DATA2的交换, 可用以下指令实现： 示例：PUSH DATA1</p>

			<div>PUSH DATA2 POP DATA1 POP DATA2</div> <div>4. 本指令不影响状态标志位.</div>
5	<b>XLAT</b>	<div>查表指令 XLAT</div> <div>格式: XLAT TABLE其中TABLE为一待查表格的首地址.</div> <div>功能: 把待查表格的一个字节内容送到AL累加器中.</div>	<div>说明:</div> <div>1. 在执行该指令前, 应将TABLE先送至BX寄存器中, 然后将待查字节与在表格中距表首地址位移量送AL, 即 <math>(AL) \leftarrow ((BX) + (AL))</math>.</div> <div>2. 本指令不影响状态标位, 表格长度不超过256字节.</div>
6	<b>LAHF</b>	<div>标志传送指令 LAHF</div> <div>格式: LAHF</div> <div>功能: 取FLAG标志寄存器低8位至AH寄存器. <math>(AH) \leftarrow (FLAG) 7 \sim 0</math></div>	<div>说明: 该指令不影响FLAG的原来内容, AH只是复制了原FLAG的低8位内容.</div>
7	<b>SAHF</b>	<div>标志传送指令 SAHF</div> <div>格式: SAHF</div> <div>功能: 将AH存至FLAG低8位</div>	<div>说明: 本指令将用AH的内容改写FLAG标志寄存器中的SF、ZF、AF、PF、和CF标志, 从而改变原来的标志位.</div>
8	<b>PUSHF</b>	<div>标志传送指令 PUSHF</div> <div>格式: PUSHF</div> <div>功能: 本指令可以把标志寄存器的内容保存到堆栈中去</div>	
9	<b>POPF</b>	<div>标志传送指令 POPF</div> <div>格式: POPF</div> <div>功能: 本指令的功能与PUSHF相反, 在子程序调用和中断服务程序中, 往往用PUSHF指令保护FLAG的内容, 用POPF指令将保护的FLAG内容恢复.</div>	<div>说明: 如果对堆栈中的原FLAG内容进行修改, 如对TF等标志位进行修改, 然后再弹回标志位寄存器FLAG. 这是通过指令修改TF标志的唯一方法.</div>
10	<b>LEA</b>	<div>有效地址传送指令 LEA</div> <div>格式: LEA OPRD1, OPRD2</div> <div>功能: 将源操作数给出的有效地址传送到指定的寄存器中.</div>	<div>说明:</div> <div>1. OPRD1 为目的操作数, 可为任意一个16位的通用寄存器.</div> <div>OPRD2 为源操作数, 可为变量名、标号或地址表达式.</div> <div>示例: LEA BX, DATA1 LEA DX, BETA[BX+SI] LEA BX BX, [BP], [DI]</div> <div>2. 本指令对标志位无影响。</div>
11	<b>LDS</b>	<div>从存储器取出32位地址的指令 LDS</div> <div>格式: LDS OPRD1, OPRD2</div> <div>功能: 从存储器取出32位地址的指令.</div>	<div>说明:</div> <div>OPRD1 为任意一个16位的寄存器.</div> <div>OPRD2 为32位的存储器地址.</div> <div>示例: LDS SI, ABCD LDS BX, FAST[SI] LDS DI, [BX]</div> <div>注意: 上面LDS DI, [BX]指令的功能是把BX所指的32位地址指针的段地址送入DS, 偏移地址送入</div>

			DI.
12	LES	<p>从存储器取出32位地址的指令 LES</p> <p>格式: LES OPRD1, OPRD2</p> <p>功能: 从存储器取出32位地址的指令.</p>	<p>说明:</p> <p>OPRD1 为任意一个16位的寄存器.</p> <p>OPRD2 为32位的存储器地址.</p> <p>示例: LES SI, ABCD LES BX, FAST[SI] LES DI, [BX]</p> <p>注意: 上面LES DI, [BX]指令的功能是把BX所指的32位地址指针的段地址送入ES, 偏移地址送入DI.</p>
13	ADD	<p>加法指令 ADD (Addition)</p> <p>格式: ADD OPRD1, OPRD2</p> <p>功能: 两数相加</p>	<p>说明:</p> <p>1. OPRD1为任一通用寄存器或存储器操作数, 可以是任意一个通用寄存器, 而且还可以是任意一个存储器操作数.</p> <p>OPRD2为立即数, 也可以是任意一个通用寄存器操作数. 立即数只能用于源操作数.</p> <p>2. OPRD1和OPRD2均为寄存器是允许的, 一个为寄存器而另一个为存储器也是允许的, 但不允许两个都是存储器操作数.</p> <p>3. 加法指令运算的结果对CF、SF、OF、PF、ZF、AF都会有影响. 以上标志也称为结果标志. 加法指令适用于无符号数或有符号数的加法运算.</p>
14	ADC	<p>带进位加法指令 ADC (Addition Carry)</p> <p>格式: ADC OPRD1, OPRD2</p> <p>功能: <math>OPRD1 \leftarrow OPRD1 + OPRD2 + CF</math></p>	<p>说明:</p> <p>1. OPRD1为任一通用寄存器或存储器操作数, 可以是任意一个通用寄存器, 而且还可以是任意一个存储器操作数.</p> <p>OPRD2为立即数, 也可以是任意一个通用寄存器操作数. 立即数只能用于源操作数.</p> <p>2. OPRD1和OPRD2均为寄存器是允许的, 一个为寄存器而另一个为存储器也是允许的, 但不允许两个都是存储器操作数.</p> <p>3. 加法指令运算的结果对CF、SF、OF、PF、ZF、AF都会有影响. 以上标志也称为结果标志.</p> <p>4. 该指令对标志位的影响同ADD指令.</p>
15	INC	<p>加1指令 INC (INCrement by 1)</p> <p>格式: INC OPRD</p> <p>功能: <math>OPRD \leftarrow OPRD + 1</math></p>	<p>说明:</p> <p>1. OPRD 为寄存器或存储器操作数.</p> <p>2. 这条指令执行结果影响AF、OF、PF、SF、ZF标志位, 但不影响CF标志位.</p>

			<p>3. 示例:</p> <pre>INC SI; (SI) &lt;-- (SI) + 1 INC WORD PTR[BX] INC BYTE PTR[BX+DI] INC CL; (CL) &lt;-- (CL) + 1</pre> <p>注意: 上述第二, 三两条指令, 是对存储字及存储字节的内容加1以替代原来的内容.</p>
16	AAA	<p>未组合的十进制加法调整指令 AAA (ASCII Adjust for Addition)</p> <p>格式: AAA</p> <p>功能: 对两个组合的十进制数相加运算(存在AL中)的结果进行调整, 产生一个未组合的十进制数放在AX中.</p>	<p>说明:</p> <p>1. 组合的十进制数和未组合的十进制数: 在计算中, 十进制数可用四位二进制数编码, 称为BCD码.</p> <p>当一个字节(8位)中存放一位BCD码, 且放在字节的低4位, 高4位为0时称为未组合的BCD码.</p> <p>2. AAA的调整操作</p> <p>若 (AL) and 0FH &gt; 9 或 AF=1, 则调整如下:</p> <p>(AL) &lt;-- (AL) + 6, (AH) &lt;-- (AH) + 1, AF=1, CF &lt;-- AF, (AL) &lt;-- (AL) and 0FH</p>
17	DAA	<p>组合的十进制加法调整指令 DAA (Decimal Adjust for Addition)</p> <p>格式: DAA</p> <p>功能: 对AL中的两个组合进制数相加的结果进行调整, 调整结果仍放在AL中, 进位标志放在CF中.</p>	<p>说明:</p> <p>1. 调整操作如下</p> <p>(1) 若 (AL) and 0FH &gt; 9 或 AF=1, 则 (AL) &lt;-- (AL) + 6, AF &lt;-- 1, 对低四位的调整.</p> <p>(2) 若 (AL) and 0F0H &gt; 90H 或 CF=1, 则 (AL) &lt;-- (AL) + 60H, CF &lt;-- 1.</p> <p>2. 示例: (AL)=18H, (BL)=06H</p> <pre>ADD AL, BL ; (AL) &lt;-- (AL) + (BL) ; (AL)=1EH DAA ; (AL)</pre>
299	SUB	<p>减法指令 SUB (SUBtract)</p> <p>格式: SUB OPRD1, OPRD2</p> <p>功能: 两个操作数的相减, 即从OPRD1中减去OPRD2, 其结果放在OPDR1中.</p>	<p>说明:</p> <p>示例</p> <pre>SUB DX, CX SUB [BX+25], AX SUB DI, ALFA[SI] SUB CL, 20 SUB DATA1[DI][BX], 20A5H</pre>
300	SBB	<p>带借位减去指令 SBB (SuBtraction with Borrow)</p> <p>格式: SBB OPRD1, OPRD2</p> <p>功能: 是进行两个操作数的相减再减去CF进位标志位, 即从OPRD1 &lt;-- OPRD1 - OPRD2 - CF, 其结果放在OPDR1中.</p>	<p>说明:</p> <p>示例</p> <pre>SBB DX, CX SBB AX, DATA1 SBB BX, 2000H SBB ALFA[BX+SI], SI SBB BETAP[DI], 030AH</pre>
301	DEC	<p>减一指令 DEC (Decrement by 1)</p> <p>格式: DEC OPRD</p> <p>功能: OPRD &lt;-- OPRD - 1</p>	<p>说明:</p> <p>1. OPRD 为寄存器或存储器操作数.</p> <p>2. 这条指令执行结果影响AF、OF、PF、SF、ZF标志位, 但不影响CF标志位.</p>



			3. 示例 DEC AX DEC CL DEC WORD PTR[DI] DEC ALFA[DI+BX]
302	NEG	取补指令 NEG (NEGate)  格式: NEG OPRD  功能: 对操作数OPRD进行取补操作, 然后将结果送回OPRD. 取补操作也叫作求补操作, 就是求一个数的相反数的补码.	说明:  1. OPRD为任意通用寄存器或存储器操作数. 2. 示例: (AL)=44H, 取补后, (AL)=0BCH(-44H).  3. 本指令影响标志位CF、OF、SF、PF、ZF及AF.
303	CMP	比较指令 CMP (CoMPare)  格式: CMP OPRD1, OPRD2  功能: 对两数进行相减, 进行比较.	说明:  1. OPRD1为任意通用寄存器或存储器操作数.  OPRD2为任意通用寄存器或存储器操作数, 立即数也可用作源操作数OPRD2.  2. 对标志位的影响同SUB指令, 完成的操作与SUB指令类似, 唯一的区别是不将OPRD1-OPRD2的结果送回OPRD1, 而只是比较.  3. 在8088/8086指令系统中, 专门提供了一组根据带符号数比较大小时, 实现条件转移的指令.
304	AAS	未组合十进制减法调整指令 AAS (ASCII Adjust for Subtraction)  格式: AAS  功能: 对两个未组合十进制数相减后存于AL中的结果进行调整, 调整后产生一个未组合的十进制数且仍存于AL中.	说明:  1. 本指令影响标志位CF及AF. 2. 调整操作 若 (AL) and 0FH > 9 或 AF=1 则 (AL)<-- (AL)-6, (AH)<-- (AH)-1, CF<--AF, (AL)<-- (AL) and 0 FH,  否则 (AL)<-- (AL) and 0FH
305	DAS	组合十进制减法调整指令 DAS (Decimal Adjust for Subtraction)  格式: DAS  功能: 对两个组合十进制数相减后存于AL中的结果进行调整, 调整后产生一个组合的十进制数且仍存于AL中.	说明:  调整操作 若 (AL) and 0FH > 9 或 AF=1, 则 (AL)<-- (AL)-6, AF=1  若 (AL) and 0F0H > 90H 或 CF=1, 则 (AL)<-- (AL)-60, CF=1
306	MUL	无符号数乘法指令 MUL (MULtiply)  格式: MUL OPRD  功能: 乘法操作.	说明:  1. OPRD为通用寄存器或存储器操作数. 2. OPRD为源操作数, 即作乘数. 目的操作数是隐含的, 即被乘数总是指定为累加器AX或AL的内容.  3. 16位乘法时, AX中为被乘数. 8位乘法时, AL为被乘数. 当16位乘法时, 32位的乘积存于DX及AX中;

			8位乘法的16位乘积存于AX中。  4. 操作过程：字节相乘： $(AX) \leftarrow (AL) * OPRD$ , 当结果的高位字节(AH)不等于0时, 则 $CF=1$ 、 $OF=1$ 。 .
307	IMUL	带符号数乘法指令 IMUL(Integer MULtiply)  格式：IMUL OPRD  功能：完成两个带符号数的相乘	说明：  1. 其中OPRD为任一通用寄存器或存储器操作数。  2. MUL指令对带符号相乘时, 不能得到正确的结果。  例如：(AL)=255 (CL)=255 MUL CL (AX)=65025 注意：这对无符号数讲, 结果是正确的, 但对带符号数讲, 相当于 $(-1)*(-1)$ 结果应为+1, 而65025对应的带符号数为-511, 显然是不正确的。
308	AAM	未组合十进制数乘法调整指令 AAM(ASCII Adjust MULtiply)  格式：AAM  功能：对两个未组合的十进制数相乘后存于AX中的结果进行调整, 产生一个未组合的十进制数存在AL中。	说明：  1. 实际上是两个未组合的十进制数字字节相乘, 一个0~9的数与另一个0~9的数相乘其积最大为81. 为了得到正确的结果, 应进行如下调整：  乘积： $(AH) \leftarrow (AL) / 10$ $(AL) \leftarrow (AL) \bmod 10$  2. 本指令应跟在MUL指令后使用, 乘积的两位十进制结果, 高位放在AH中, 低位放在AL中. AH内容是MUL指令的结果被10除的商, 即 $(AL) / 10$ , 而最后的AL内容是乘积被10整除的余数(即个位数)。
309	DIV	无符号数除法指令 DIV(DIVision)  格式：DIV OPRD  功能：实现两个无符号二进制数除法运算。	说明：  1. 其中OPRD为任一个通用寄存器或存储器操作数。  2. 字节相除, 被除数在AX中; 字相除, 被除数在DX, AX中, 除数在OPRD中。  字节除法： $(AL) \leftarrow (AX) / OPRD$ , $(AH) \leftarrow (AX) \bmod OPRD$  字除法： $(AX) \leftarrow (DX)$ $(AX) / OPRD$ , $(DX) \leftarrow (DX) (AX) \bmod OPRD$
310	IDIV	带符号数除法指定 IDIV(Interger DIVision)  格式：IDIV OPRD  功能：这实现两个带符号数的二进制除法运算。	说明：  1. 其中OPRD为任一通用寄存器或存储器操作数。  2. 理由与IMUL相同, 只有IDIV指令, 才能得到符号数相除的正确结果。  3. 当被除数为8位, 在进行字节除法前, 应把AL的符号位扩充至

			AH中. 在16位除法时, 若被除数为16位, 则应将AX中的符号位扩到DX中.
311	CBW	字节扩展指令 CBW(Convert Byte to Word)  格式: CBW  功能: 将字节扩展为字, 即把AL寄存器的符号位扩展到AH中.	说明:  1. 两个字节相除时, 先使用本指令形成一个双字节长的被除数.  2. 本指令不影响标志位.  3. 示例: MOV AL, 25 CBW IDIV BYTE PTR DATA1
312	CWD	字扩展指令 CWD(Convert Word to Double Word)  格式: CWD  功能: 将字扩展为双字长, 即把AX寄存器的符号位扩展到DX中.	说明:  1. 两个字或字节相除时, 先用本指令形成一个双字长的被除数.  2. 本指令不影响标志位.  3. 示例: 在B1、B2、B3字节类型变量中, 分别存有8们带符号数a、b、c, 实现(a*b+c)/a运算。
313	AAD	未组合十进制数除法调整指令 AAD(ASCII Adjust for Division)  格式: AAD  功能: 在除法指令前对AX中的两个未组合十进制数进行调整, 以便能用DIV指令实现两个未组合的十进制数的除法运算, 其结果为未组合的十进制数, 商(在AL中)和余数(在AH中).	说明:  1. AAD指令是在执行除法DIV之前使用的, 以便得到二进制结果存于AL中, 然后除以OPRD, 得到的商在AL中, 余数在AH中.  2. 示例: MOV BL, 5 MOV AX, 0308H AAD ; (AL)<--1EH+08H=26H, (AH)<--0 DIV BL ; 商=07H-->(AL), 余数=03H-->(AH).
314	AND	逻辑与运算指令 AND  格式: AND OPRD1, OPRD2  功能: 对两个操作数实现按位逻辑与运算, 结果送至目的操作数. 本指令可以进行字节或字的‘与’运算,  OPRD1<--OPRD1 and OPRD2.	说明:  1. 目的操作数OPRD1为任一通用寄存器或存储器操作数. 源操作数OPRD2为立即数, 任一通用寄存器或存储器操作数.  2. 示例: AND AL, 0FH ; (AL)<--(AL) AND 0FH AND AX, BX ; (AX)<--(AX) AND (BX) AND DX, BUFFER[SI+BX] AND BETA[BX], 00FFH 注意: 两数相与, 有一个数假则值为假
315	OR	逻辑或指令 OR  格式: OR OPRD1, OPRD2  功能: OR指令完成对两个操作数按位的‘或’运算, 结果送至目的操作数中, 本指令可以进行字节或字的‘或’运算.  OPRD1<--OPRD1 OR OPRD2.	说明:  1. 其中OPRD1, OPRD2含义与AND指令相同, 对标志位的影响也与AND指令相同.  2. 两数相或, 有一个数为真则值为真.
316	NOT	逻辑非运算指令 NOT  格式: NOT OPRD  功能: 完成对操作数按位求反运算(即0变1, 1变0),	说明:  1. 其中OPRD可为任一通用寄存器或存储器操作数.

		结果关回原操作数.	2. 本指梳令可以进行字或字节‘非’运算. 3. 本指令不影响标志位.
317	XOR	逻辑异或运算指令 XOR  格式: XOR OPRD1, OPRD2  功能: 实现两个操作数按位‘异或’运算, 结果送至目的操作数中.  OPRD1<--OPRD1 XOR OPRD2	说明:  1. 其在OPRD1、OPRD2的含义与AND指令相同, 对标志位的影响与AND指令相同.  2. 相异为真, 相同为假.
318	TEST	测试指令 TEST  格式: TEST OPRD1, OPRD2  功能: 其中OPRD1、OPRD2的含义同AND指令一样, 也是对两个操作数进行按位的‘与’运算, 唯一不同之处是不将‘与’的结果送目的操作数, 即本指令对两个操作数 的内容均不进行修改, 仅是在逻辑与操作后, 对标志位重新置位.	说明: TEST与AND指令的关系, 有点类似于CMP与SUB指令之间的关系.
319	SHL	逻辑左移指令 SHL (Shift logical left)  格式: SHL OPRD1, COUNT  功能: 对给定的目的操作数左移COUNT次, 每次移位时最高位移入标志位CF中, 最低位补零.	说明:  1. 其中OPRD1为目的操作数, 可以是通用寄存器或存储器操作数.  2. COUNT代表移位的次数(或位数). 移位一次, COUNT=1; 移位多于1次时, COUNT=(CL), (CL)中为移位的次数.  3. 例如: SHL AL, 1 SHL CX, 1 SHL ALFA[DI] 或者: MOV CL, 3 SHL DX, CL SHL ALFA[DI], CL
320	SHR	逻辑右移指令 SHR  格式: SHR OPRD1, COUNT  功能: 本指令实现由COUNT决定次数的逻辑右移操作, 每次移位时, 最高位补0, 最低位移至标志位CF中.	说明:  1. 其中OPRD1为目的操作数, 可以是通用寄存器或存储器操作数.  2. COUNT代表移位的次数(或位数). 移位一次, COUNT=1; 移位多于1次时, COUNT=(CL), (CL)中为移位的次数.  3. 影响标志位OF, PF, SF, ZF, CF.
321	SAL	算术左移指令 SAL (Shift Arithmetic Left)  格式: SAL OPRD1, COUNT  功能: 其中OPRD1, COUNT与指令SHL相同. 本指令与SHL的功能也完全相同, 这是因为逻辑左移指令与算术左移指令所要完成的操作是一样的.	说明:  1. 其中OPRD1为目的操作数, 可以是通用寄存器或存储器操作数.  2. COUNT代表移位的次数(或位数). 移位一次, COUNT=1; 移位多于1次时, COUNT=(CL), (CL)中为移位的次数.
322	SAR	算术右移指令 SAR  格式: SAR OPRD1, COUNT	说明:  1. 其中OPRD1为目的操作数, 可以是通用寄存器或存储器操作



		功能：本指令通常用于对带符号数减半的运算中，因而在每次右移时，保持最高位(符号位)不变，最低位右移至CF中。	数。 2. COUNT代表移位的次数(或位数)。移位一次, COUNT=1; 移位多于1次时, COUNT=(CL), (CL)中为移位的次数。
323	<b>ROL</b>	循环移位指令  格式：ROL OPRD1, COUNT ;不含进位标志位CF在循环中的左循环移位指令。  ROR OPRD1, COUNT ;不含进位示志位CF在循环中的右循环移位指令。  RCL OPRD1, COUNT ;带进位的左循环移位指令。  RCR OPRD1, COUNT ;带进位的右循环移位指令。	说明：  1. 本指令组只影响标志CF、OF。OF由移入CF的内容决定, OF取决于移位一次后符号位是否改变, 如改变, 则OF=1。  2. 由于是循环移位, 所以对字节移位8次; 对字移位16次, 就可恢复为原操作数。由于带CF的循环移位, 可以将CF的内容移入, 所以可以利用它实现多字节的循环。
324	<b>ROR</b>	循环移位指令  格式： ROL OPRD1, COUNT ;不含进位标志位CF在循环中的左循环移位指令。 ROR OPRD1, COUNT ;不含进位示志位CF在循环中的右循环移位指令。 RCL OPRD1, COUNT ;带进位的左循环移位指令。 RCR OPRD1, COUNT ;带进位的右循环移位指令。	说明：  1. 本指令组只影响标志CF、OF。OF由移入CF的内容决定, OF取决于移位一次后符号位是否改变, 如改变, 则OF=1。  2. 由于循环移位, 所以对字节移位8次; 对字移位16次, 可恢复为原操作数。
325	<b>RCL</b>	循环移位指令  格式：ROL OPRD1, COUNT ;不含进位标志位CF在循环中的左循环移位指令。 ROR OPRD1, COUNT ;不含进位示志位CF在循环中的右循环移位指令。 RCL OPRD1, COUNT ;带进位的左循环移位指令。 RCR OPRD1, COUNT ;带进位的右循环移位指令。	说明：  1. 本指令组只影响标志CF、OF。OF由移入CF的内容决定, OF取决于移位一次后符号位是否改变, 如改变, 则OF=1。  2. 由于是循环移位, 所以对字节移位8次; 对字移位16次, 就可恢复为原操作数。由于带CF的循环移位, 可以将CF的内容移入,  所以可以利用它实现多字节的循环。
326	<b>RCR</b>	循环移位指令  格式：ROL OPRD1, COUNT ;不含进位标志位CF在循环中的左循环移位指令。  ROR OPRD1, COUNT ;不含进位示志位CF在循环中的右循环移位指令。  RCL OPRD1, COUNT ;带进位的左循环移位指令。  RCR OPRD1, COUNT ;带进位的右循环移位指令。	说明：  1. 本指令组只影响标志CF、OF。OF由移入CF的内容决定, OF取决于移位一次后符号位是否改变, 如改变, 则OF=1。  2. 由于是循环移位, 所以对字节移位8次; 对字移位16次, 就可恢复为原操作数。由于带CF的循环移位, 可以将CF的内容移入, 所以可以利用它实现多字节的循环。  注意：以上程序中的指令SHR AL, CL如改为SAR AL, CL, 虽然最高4位可移入低4位, 但最高位不为0, 故应加入一条指令AND AL, 0FH。否则, 若最高位不为0时, 将得到错误结果。
327	<b>JMP</b>	无条件转移指令JMP  格式：JMP OPRD	说明：  1. 其中OPRD为转移的目的地址。程序转移到目的地址所指向的指

		功能：JMP指令将无条件地控制程序转移到目的地地址去执行. 当目的地地址仍在同一个代码段内, 称为段内转移; 当目标地址不在同一个代码段内, 则称为段间转移. 这两种情况都将产生不同的指令代码, 以便能正确地生成目的地地址, 在 段内转移时, 指令只要能提供目的地地址的段 内偏移量即够了; 而在段间转移时, 指令应能提供目的地地址的段地址及段内偏移地址值.	令继续往下执行. 2. 本组指令对标志位无影响. 3. <1> 段内直接转移指令: JMP NEAR 标号  <2> 段内间接转移指令: JMP OPRD  <3> 段间直接转移指令: JMP FAR 标号  <4> 段间间接转移指令: JMP OPRD其中的OPRD为存储器双字操作数. 段间间接转移只能通过存储器操作数来实现.
328	JC	条件转移指令 JC  格式: JC 标号  功能: CF=1, 转至标号处执行	说明: JC为根据标志位CF进行转移的指令
329	JNC	条件转移指令JNC  格式: JNC标号  功能: CF=0, 转至标号处执行	说明: JNC为根据标志位CF进行转移的指令
330	JE	条件转移指令JE/JZ  格式: JE/JZ标号  功能: ZF=1, 转至标号处执	说明: 1. 指令JE与JZ等价, 它们是根据标志位ZF进行转移的指令 2. JE, JZ均为一条指令的两种助记符表示方法
331	JZ	条件转移指令JE/JZ  格式: JE/JZ标号  功能: ZF=1, 转至标号处执	说明: 1. 指令JE与JZ等价, 它们是根据标志位ZF进行转移的指令 2. JE, JZ均为一条指令的两种助记符表示方法
332	JNE	条件转移指令JNE/JNZ  格式: JNE/JNZ 标号  功能: ZF=0, 转至标号处执行	说明: 1. 指令JNE与JNZ等价, 它们是根据标志位ZF进行转移的指令 2. JNE, JNZ均为一条指令的两种助记符表示方法
333	JNZ	条件转移指令JNE/JNZ  格式: JNE/JNZ 标号  功能: ZF=0, 转至标号处执行	说明: 1. 指令JNE与JNZ等价, 它们是根据标志位ZF进行转移的指令 2. JNE, JNZ均为一条指令的两种助记符表示方法
334	JS	条件转移指令JS  格式: JS 标号  功能: SF=1, 转至标号处执行	说明: JS是根据符号标志位SF进行转移的指令
335	JNS	条件转移指令JNS  格式: JNS 标号  功能: SF=0, 转至标号处执行	说明: JNS是根据符号标志位SF进行转移的指令

336	J0	条件转移指令J0  格式：J0 标号  功能：OF=1, 转至标号处执行	说明：J0是根据溢出标志位OF进行转移的指令
337	JNO	条件转移指令JNO  格式：JNO 标号  功能：OF=0, 转至标号处执行	说明：JNO是根据溢出标志位OF进行转移的指令
338	JP	条件转移指令JP/JPE  格式：JP/JPE 标号  功能：PF=1, 转至标号处执行	说明：  1. 指令JP与JPE, 它们是根据奇偶标志位PF进行转移的指令  2. JP, JPE均为一条指令的两种助记符表示方法
339	JPE	条件转移指令JP/JPE  格式：JP/JPE 标号  功能：PF=1, 转至标号处执行	说明：  1. 指令JP与JPE, 它们是根据奇偶标志位PF进行转移的指令  2. JP, JPE均为一条指令的两种助记符表示方法
340	JNP	条件转移指令JNP/JPO  格式：JNP/JPO 标号  功能：PF=0, 转至标号处执行	说明：  1. 指令JNP与JPO, 它们是根据奇偶标志位PF进行转移的指令  2. JNP, JPO均为一条指令的两种助记符表示方法
341	JPO	条件转移指令JNP/JPO  格式：JNP/JPO 标号  功能：PF=0, 转至标号处执行	说明：  1. 指令JNP与JPO, 它们是根据奇偶标志位PF进行转移的指令  2. JNP, JPO均为一条指令的两种助记符表示方法
342	JA	条件转移指令JA/JNBE  格式：JA/JNBE标号  功能：为高于/不低于等于的转移指令	说明：  1. 例如两个符号数a, b比较时, a>b(即CF=0, ZF=0)时转移. 因为单一标志位CF=0, 只表示a>=b.  2. JA/JNBE是同一条指令的两种不同的助记符.  3. 该指令用于无符号数进行条件转移
343	JNBE	条件转移指令JA/JNBE  格式：JA/JNBE标号  功能：为高于/不低于等于的转移指令	说明：  1. 例如两个符号数a, b比较时, a>b(即CF=0, ZF=0)时转移. 因为单一标志位CF=0, 只表示a>=b.  2. JA/JNBE是同一条指令的两种不同的助记符.  3. 该指令用于无符号数进行条件转移
344	JAE	条件转移指令JAE/JNB  格式：JAE/JNB 标号	说明：  1. JAE/JNB是同一条指令的两种不同的助记符.

		功能：为高于等于/不低于的转移指令	2. 该指令用于无符号数进行条件转移.
345	JNB	条件转移指令JAE/JNB  格式：JAE/JNB 标号  功能：为高于等于/不低于的转移指令	说明：  1. JAE/JNB是同一条指令的两种不同的助记符.  2. 该指令用于无符号数进行条件转移.
346	JB	条件转移指令JB/JNAE  格式：JB/JNAE 标号  功能：低于/不高于等于时转移	说明：该指令用于无符号数的条件转移
347	JNAE	条件转移指令JB/JNAE  格式：JB/JNAE 标号  功能：低于/不高于等于时转移	说明：该指令用于无符号数的条件转移
348	JBE	条件转移指令JBE/JNA  格式：JBE/JNA 标号  功能：低于等于/不高于时转移	说明：该指令用于无符号数的条件转移
349	JNA	条件转移指令JBE/JNA  格式：JBE/JNA 标号  功能：低于等于/不高于时转移	说明：该指令用于无符号数的条件转移
350	JG	条件转移指令JG/JNLE  格式：JG/JNLE 标号  功能：大于/不小于等于时转移	说明：用于带符号数的条件转移指令
351	JNLE	条件转移指令JG/JNLE  格式：JG/JNLE 标号  功能：大于/不小于等于时转移	说明：用于带符号数的条件转移指令
352	JGE	条件转移指令JGE/JNL  格式：JGE/JNL标号  功能：大于等于/不小于时转移	说明：用于带符号数的条件转移指令
353	JNL	条件转移指令JGE/JNL  格式：JGE/JNL标号  功能：大于等于/不小于时转移	说明：用于带符号数的条件转移指令
354	JL	条件转移指令JL/JNGE  格式：JL/JNGE标号  功能：小于/不大于等于时转移	说明：用于带符号数的条件转移指令
355	JNGE	条件转移指令JL/JNGE  格式：JL/JNGE标号  功能：小于/不大于等于时转移	说明：用于带符号数的条件转移指令
356	JLE	条件转移指令JLE/JNG  格式：JLE/JNG 标号  功能：小于等于/不大于时转移	说明：用于带符号数的条件转移指令



357	JNG	条件转移指令JLE/JNG  格式：JLE/JNG 标号  功能：小于等于/不大于时转移	说明：用于带符号数的条件转移指令
358	LOOP	循环控制指令LOOP  格式：LOOP 标号  功能：(CX)←-(CX)-1, (CX)≠0, 则转移至标号处循环执行，直至(CX)=0, 继续执行后继指令。	说明：  1. 本指令是用CX寄存器作为计数器, 来控制程序的循环。  2. 它属于段内SHORT短类型转移, 目的地址必须距本指令在-128到+127个字节的范围内。
359	LOOPZ	循环控制指令LOOPZ/LOOPE  格式：LOOPZ/LOOPE 标号  功能：(CX)←-(CX)-1, (CX)≠0 且ZF=1时, 转至标号处循环	说明：  1. 本指令是用CX寄存器作为计数器, 来控制程序的循环。  2. 它属于段内SHORT短类型转移, 目的地址必须距本指令在-128到+127个字节的范围内。  3. 以上两种助记符等价。
360	LOOPE	循环控制指令LOOPZ/LOOPE  格式：LOOPZ/LOOPE 标号  功能：(CX)←-(CX)-1, (CX)≠0 且ZF=1时, 转至标号处循环	说明：  1. 本指令是用CX寄存器作为计数器, 来控制程序的循环。  2. 它属于段内SHORT短类型转移, 目的地址必须距本指令在-128到+127个字节的范围内。  3. 以上两种助记符等价。
361	LOOPNZ	循环控制指令LOOPNZ/LOOPNE  格式：LOOPNZ/LOOPNE 标号  功能：(CX)←-(CX)-1, (CX)≠0 且ZF=0时, 转至标号处循环	说明：  1. 本指令是用CX寄存器作为计数器, 来控制程序的循环。  2. 它属于段内SHORT短类型转移, 目的地址必须距本指令在-128到+127个字节的范围内。  3. 以上两种助记符等价。
362	LOOPNE	循环控制指令LOOPNZ/LOOPNE  格式：LOOPNZ/LOOPNE 标号  功能：(CX)←-(CX)-1, (CX)≠0 且ZF=0时, 转至标号处循环	说明：  1. 本指令是用CX寄存器作为计数器, 来控制程序的循环。  2. 它属于段内SHORT短类型转移, 目的地址必须距本指令在-128到+127个字节的范围内。  3. 以上两种助记符等价。
363	CALL	过程调用指令 CALL  格式：CALL OPRD  功能：过程调用指令	说明：  1. 其中OPRD为过程的目的地址。  2. 过程调用可分为段内调用和段间调用两种. 寻址方式也可以分为直接寻址和间接寻址两种。  3. 本指令不影响标志位。
364	RET	返回指令 RET  格式：RET	说明：  由于在过程定义时, 已指明其近

		功能：当调用的过程结束后实现从过程返回至原调用程序的下一条指令, 本指令不影响标志位.	(NEAR) 或远 (FAR) 的属性, 所以 RET指令根据段内调用与段间调用, 执行不同的操作  对段内调用：返回时, 由堆栈弹出一个字的返回地址的段内偏移量至IP.  对段外调用：返回时, 由堆栈弹出的第一个字为返回地址的段内偏移量, 将其送入IP中, 由堆栈弹出第二个字为返回地址的段基址, 将其送入CS中.
365	MOVS	字符串传送指令 MOVS  格式：MOVS OPRD1, OPRD2 MOVSB MOVSW  功能：OPRD1←--OPRD2.	说明：  1. 其中OPRD2为源串符号地址, OPRD1为目的串符号地址.  2. 字节串操作：若DF=0, 则作加, 若DF=1, 则作减.  3. 对字串操作时：若DF=0, 则作加, 若DF=1, 则作减, .  4. 在指令中不出现操作数时, 字节串传送格式为MOVSB、字串传送格式为MOVSW.  5. 本指令不影响标志位.
366	MOVSB	字符串传送指令 MOVS  格式：MOVS OPRD1, OPRD2 MOVSB MOVSW  功能：OPRD1←--OPRD2.	说明：  1. 其中OPRD2为源串符号地址, OPRD1为目的串符号地址.  2. 字节串操作：若DF=0, 则作加, 若DF=1, 则作减.  3. 对字串操作时：若DF=0, 则作加, 若DF=1, 则作减, .  4. 在指令中不出现操作数时, 字节串传送格式为MOVSB、字串传送格式为MOVSW.  5. 本指令不影响标志位.
367	MOVSW	字符串传送指令 MOVS  格式：MOVS OPRD1, OPRD2 MOVSB MOVSW  功能：OPRD1←--OPRD2.	说明：  1. 其中OPRD2为源串符号地址, OPRD1为目的串符号地址.  2. 字节串操作：若DF=0, 则作加, 若DF=1, 则作减.  3. 对字串操作时：若DF=0, 则作加, 若DF=1, 则作减, .  4. 在指令中不出现操作数时, 字节串传送格式为MOVSB、字串传送格式为MOVSW.  5. 本指令不影响标志位.
368	CMPS	字符串比较指令  格式：CMPS OPRD1, OPRD2 CMPSB CMPSW	说明：  1. 其中OPRD2为源串符号地址, OPRD1为目的串符号地址.

		<p>功能：由SI寻址的源串中数据与由DI寻址的目的串中数据进行比较, 比较结果送标志位, 而不改变操作数本身。 同时SI, DI将自动调整。</p>	<p>2. 本指令影响标志位AF、CF、OF、SF、PF、ZF. 本指令可用来检查二个字符串是否相同, 可以使用循环控制方法对整串进行比较。</p> <p>3. 与MOVS相似, CMPS指令也可以不使用操作数, 此时可用指令CMPSB或CMPSW分别表示字节串比较或字串比较。</p>
369	CMPSB	<p>字符串比较指令</p> <p>格式：CMPS OPRD1, OPRD2 CMPSB CMPSW</p> <p>功能：由SI寻址的源串中数据与由DI寻址的目的串中数据进行比较, 比较结果送标志位, 而不改变操作数本身。 同时SI, DI将自动调整。</p>	<p>说明：</p> <p>1. 其中OPRD2为源串符号地址, OPRD1为目的串符号地址。</p> <p>2. 本指令影响标志位AF、CF、OF、SF、PF、ZF. 本指令可用来检查二个字符串是否相同, 可以使用循环控制方法对整串进行比较。</p> <p>3. 与MOVS相似, CMPS指令也可以不使用操作数, 此时可用指令CMPSB或CMPSW分别表示字节串比较或字串比较。</p>
370	CMPSW	<p>字符串比较指令</p> <p>格式：CMPS OPRD1, OPRD2 CMPSB CMPSW</p> <p>功能：由SI寻址的源串中数据与由DI寻址的目的串中数据进行比较, 比较结果送标志位, 而不改变操作数本身。 同时SI, DI将自动调整。</p>	<p>说明：</p> <p>1. 其中OPRD2为源串符号地址, OPRD1为目的串符号地址。</p> <p>2. 本指令影响标志位AF、CF、OF、SF、PF、ZF. 本指令可用来检查二个字符串是否相同, 可以使用循环控制方法对整串进行比较。</p> <p>3. 与MOVS相似, CMPS指令也可以不使用操作数, 此时可用指令CMPSB或CMPSW分别表示字节串比较或字串比较。</p>
371	SCAS	<p>字符串搜索指令 SCAS</p> <p>格式：SCAS OPRD SCASB SCASW</p> <p>功能：把AL(字节串)或AX(字串)的内容与由DI寄存器寻址的目的串中的数据相减, 结果置标志位, 但不改变任一操作数本身。</p> <p>地址指针DI自动调整。</p>	<p>说明：</p> <p>1. 其中OPRD为目的串符号地址。</p> <p>2. 本指令影响标志AF、CF、OF、PF、SF、ZF. 该指令可查找字符串中的一个关键字, 只需在本指令执行前,</p> <p>把关键字放在AL(字节)或AX(字串)中, 用重复前缀可在整串中查找。</p> <p>指令中不使用操作数时, 可用指令格式SCASB, SCASW, 分别表示字节串或字串搜索指令。</p>
372	SCASB	<p>字符串搜索指令 SCAS</p> <p>格式：SCAS OPRD SCASB SCASW</p> <p>功能：把AL(字节串)或AX(字串)的内容与由DI寄存器寻址的目的串中的数据相减, 结果置标志位, 但不改变任一操作数本身。</p>	<p>说明：</p> <p>1. 其中OPRD为目的串符号地址。</p> <p>2. 本指令影响标志AF、CF、OF、PF、SF、ZF. 该指令可查找字符串中的一个关键字, 只需在本指令执行前,</p>

		地址指针DI自动调整.	把关键字放在AL(字节)或AX(字串)中,用重复前缀可在整串中查找.  指令中不使用操作数时,可用指令格式SCASB, SCASW, 分别表示字节串或字串搜索指令.
373	SCASW	字符串搜索指令 SCAS  格式: SCAS OPRD SCASB SCASW  功能: 把AL(字节串)或AX(字串)的内容与由DI寄存器寻址的目的串中的数据相减,结果置标志位,但不改变任一操作数本身.  地址指针DI自动调整.	说明:  1. 其中OPRD为目的串符号地址.  2. 本指令影响标志AF、CF、OF、PF、SF、ZF. 该指令可查找字符串中的一个关键字,只需在本指令执行前,  把关键字放在AL(字节)或AX(字串)中,用重复前缀可在整串中查找.  指令中不使用操作数时,可用指令格式SCASB, SCASW, 分别表示字节串或字串搜索指令.
374	LODS	取字符串元素指令 LODS  格式: LODS OPRD 其中OPRD为源字符串符号地址.  功能: 把SI寻址的源串的数据字节送AL或数据字送AX中去, 并根据DF的值修改地址指针SI进行自动调整.	说明:  1. 本指令不影响标志位.  2. 当不使用操作数时,可用LODS(字节串)或LODSW(字串)指令.
375	STOS	字符串存储指令 STOS  格式: STOS OPRD  功能: 把AL(字节)或AX(字)中的数据存储在DI为目的串地址指针所寻址的存储器单元中去. 指针DI将根据DF的值进行自动调整.	说明:  1. 其中OPRD为目的串符号地址.  2. 本指令不影响标志位. 当不使用操作数时,可用STOSB或STOSW分别表示字节串或字串的操作.
376	REP	重复前缀的说明  格式: REP ;CX<>0 重复执行字符串指令 REPZ/REPE ;CX<>0 且ZF=1重复执行字符串指令 REPNZ/REPNE ;CX<>0 且ZF=0重复执行字符串指令  功能: 在串操作指令前加上重复前缀,可以对字符串进重复处理. 由于加上重复前缀后,对应的指令代码是不同的,所以指令的功能便具有重复处理的功能,重复的次数存放在CX寄存器中.	说明:  1. REP与MOVS或STOS串操作指令相结合使用,完成一组字符的传送或建立一组相同数据的字符串.  2. REPZ/REPE常用与CMPS串操作指令结合使用,可以完成两组字符串的比较.  3. REPZ/REPE常与SCAS指令结合使用,可以完成在一个字符串中搜索一个关键字.  4. REPNZ/REPNE与CMPS指令结合使用,表示当串未结束(CX=1)且当对应串元素不相同(ZF=0)时,继续重复执行串比较指令.
377	REPZ	重复前缀的说明  格式: REP ;CX<>0 重复执行字符串指令 REPZ/REPE ;CX<>0 且ZF=1重复执行字符串指令 REPNZ/REPNE ;CX<>0 且ZF=0重复执行字符串指令	说明:  1. REPZ/REPE常用与CMPS串操作指令结合使用,可以完成两组字符串的比较.  2. REPZ/REPE常与SCAS指令结合



		<p>功能：在串操作指令前加上重复前缀,可以对字符串进重复处理. 由于加上重复前缀后,对应的指令代码是不同的,所以指令的功能便具有重复处理的功能,重复的次数存放在CX寄存器中.</p>	<p>使用,可以完成在一个字符串中搜索一个关键字.</p> <p>3. REPZ/REPNE与CMPS指令结合使用,表示当串未结束(CX=1)且当对应串元素不相同(ZF=0)时,继续重复执行串比较指令.</p> <p>4. REPZ/REPNE与SCAS指令结合使用,表示串未结束(CX=1)且当关键字与串元素不相同(ZF=0)时,继续重复执行串搜索指令.</p>
378	REPE	<p>重复前缀的说明</p> <p>格式: REP ;CX&lt;&gt;0 重复执行字符串指令 REPZ/REPE ;CX&lt;&gt;0 且ZF=1重复执行字符串指令 REPZ/REPNE ;CX&lt;&gt;0 且ZF=0重复执行字符串指令</p> <p>功能：在串操作指令前加上重复前缀,可以对字符串进重复处理. 由于加上重复前缀后,对应的指令代码是不同的,所以指令的功能便具有重复处理的功能,重复的次数存放在CX寄存器中.</p>	<p>说明:</p> <p>1. REPZ/REPE常用与CMPS串操作指令结合使用,可以完成两组字符串的比较.</p> <p>2. REPZ/REPE常与SCAS指令结合使用,可以完成在一个字符串中搜索一个关键字.</p> <p>3. REPZ/REPNE与CMPS指令结合使用,表示当串未结束(CX=1)且当对应串元素不相同(ZF=0)时,继续重复执行串比较指令.</p> <p>4. REPZ/REPNE与SCAS指令结合使用,表示串未结束(CX=1)且当关键字与串元素不相同(ZF=0)时,继续重复执行串搜索指令.</p>
379	REPZ	<p>重复前缀的说明</p> <p>格式: REP ;CX&lt;&gt;0 重复执行字符串指令 REPZ/REPE ;CX&lt;&gt;0 且ZF=1重复执行字符串指令 REPZ/REPNE ;CX&lt;&gt;0 且ZF=0重复执行字符串指令</p>	<p>说明:</p> <p>1. REPZ/REPE常用与CMPS串操作指令结合使用,可以完成两组字符串的比较.</p> <p>2. REPZ/REPE常与SCAS指令结合使用,可以完成在一个字符串中搜索一个关键字.</p> <p>3. REPZ/REPNE与CMPS指令结合使用,表示当串未结束(CX=1)且当对应串元素不相同(ZF=0)时,继续重复执行串比较指令.</p> <p>4. REPZ/REPNE与SCAS指令结合使用,表示串未结束(CX=1)且当关键字与串元素不相同(ZF=0)时,继续重复执行串搜索指令.</p>
380	REPNE	<p>重复前缀的说明</p> <p>格式: REP ;CX&lt;&gt;0 重复执行字符串指令 REPZ/REPE ;CX&lt;&gt;0 且ZF=1重复执行字符串指令 REPZ/REPNE ;CX&lt;&gt;0 且ZF=0重复执行字符串指令</p>	<p>说明:</p> <p>1. REPZ/REPE常用与CMPS串操作指令结合使用,可以完成两组字符串的比较.</p> <p>2. REPZ/REPE常与SCAS指令结合使用,可以完成在一个字符串中搜索一个关键字.</p> <p>3. REPZ/REPNE与CMPS指令结合使用,表示当串未结束(CX=1)且当对应串元素不相同(ZF=0)时,继续重复执行串比较指令.</p> <p>4. REPZ/REPNE与SCAS指令结合</p>

			使用, 表示串未结束(CX=1)且当关键字与串元素不相同(ZF=0)时, 继续重复执行串搜索指令.
381	CLC	<p>处理器控制指令—标志位操作指令</p> <p>格式:</p> <p>CLC ;置CF=0</p> <p>STC ;置CF=1</p> <p>CMC ;置CF=(Not CF)进位标志求反</p> <p>CLD ;置DF=0</p> <p>STD ;置DF=1</p> <p>CLI ;置IF=0, CPU禁止响应外部中断</p> <p>STI ;置IF=1, 使CPU允许向应外部中断</p> <p>功能: 完成对标志位的置位、复位等操作.</p>	说明: 例如串操作中的程序, 经常用CLD指令清方向标志使DF=0, 在串操作指令执行时, 按增量的方式修改吕指针.
382	STC	<p>处理器控制指令—标志位操作指令</p> <p>格式:</p> <p>CLC ;置CF=0</p> <p>STC ;置CF=1</p> <p>CMC ;置CF=(Not CF)进位标志求反</p> <p>CLD ;置DF=0</p> <p>STD ;置DF=1</p> <p>CLI ;置IF=0, CPU禁止响应外部中断</p> <p>STI ;置IF=1, 使CPU允许向应外部中断</p> <p>功能: 完成对标志位的置位、复位等操作.</p>	说明: 例如串操作中的程序, 经常用CLD指令清方向标志使DF=0, 在串操作指令执行时, 按增量的方式修改吕指针.
383	CMC	<p>处理器控制指令—标志位操作指令</p> <p>格式:</p> <p>CLC ;置CF=0</p> <p>STC ;置CF=1</p> <p>CMC ;置CF=(Not CF)进位标志求反</p> <p>CLD ;置DF=0</p> <p>STD ;置DF=1</p> <p>CLI ;置IF=0, CPU禁止响应外部中断</p> <p>STI ;置IF=1, 使CPU允许向应外部中断</p> <p>功能: 完成对标志位的置位、复位等操作.</p>	说明: 例如串操作中的程序, 经常用CLD指令清方向标志使DF=0, 在串操作指令执行时, 按增量的方式修改吕指针.
384	CLD	<p>处理器控制指令—标志位操作指令</p> <p>格式:</p> <p>CLC ;置CF=0</p> <p>STC ;置CF=1</p> <p>CMC ;置CF=(Not CF)进位标志求反</p> <p>CLD ;置DF=0</p> <p>STD ;置DF=1</p> <p>CLI ;置IF=0, CPU禁止响应外部中断</p> <p>STI ;置IF=1, 使CPU允许向应外部中断</p> <p>功能: 完成对标志位的置位、复位等操作.</p>	说明: 例如串操作中的程序, 经常用CLD指令清方向标志使DF=0, 在串操作指令执行时, 按增量的方式修改吕指针.
385	STD	<p>处理器控制指令—标志位操作指令</p> <p>格式:</p> <p>CLC ;置CF=0</p> <p>STC ;置CF=1</p> <p>CMC ;置CF=(Not CF)进位标志求反</p> <p>CLD ;置DF=0</p> <p>STD ;置DF=1</p> <p>CLI ;置IF=0, CPU禁止响应外部中断</p> <p>STI ;置IF=1, 使CPU允许向应外部中断</p> <p>功能: 完成对标志位的置位、复位等操作.</p>	说明: 例如串操作中的程序, 经常用CLD指令清方向标志使DF=0, 在串操作指令执行时, 按增量的方式修改吕指针.
386	CLI	<p>处理器控制指令—标志位操作指令</p> <p>格式:</p>	说明: 例如串操作中的程序, 经常用CLD指令清方向标志使DF=0, 在串操作指令执行时, 按增量的

		CLC ;置CF=0 STC ;置CF=1 CMC ;置CF=(Not CF)进位标志求反 CLD ;置DF=0 STD ;置DF=1 CLI ;置IF=0, CPU禁止响应外部中断 STI ;置IF=1, 使CPU允许向应外部中断  功能: 完成对标志位的置位、复位等操作.	方式修改吕指针.
387	STI	处理器控制指令—标志位操作指令  格式: CLC ;置CF=0 STC ;置CF=1 CMC ;置CF=(Not CF)进位标志求反 CLD ;置DF=0 STD ;置DF=1 CLI ;置IF=0, CPU禁止响应外部中断 STI ;置IF=1, 使CPU允许向应外部中断  功能: 完成对标志位的置位、复位等操作.	说明: 例如串操作中的程序, 经常用CLD指令清方向标志使DF=0, 在串操作指令执行时, 按增量的方式修改吕指针.
388	HLT	处理器暂停指令 HLT  格式: HLT  功能: 使处理器处于暂时停机状态.	说明:  1. 本指令不影响标志位.  2. 由执行HLT引起的暂停, 只有RESET(复位)、NMI(非屏蔽中断请求)、INTR(可屏蔽的外部中断请求)信号可以使  其退出暂停状态. 它可用于等待中断的到来或多机系统的同步操作.
389	WAIT	处理器等待指令 WAIT  格式: WAIT  功能: 本指令将使处理器检测TEST端脚, 当TEST有效时, 则退出等待状态执行下条指令, 否则处理器处于等待状态, 直到TEST有效.	说明: 本指令不影响标志位.
390	ESC	处理器交权指令 ESC  格式: ESC EXTOPRD, OPRD  功能: 使用本指令可以实现协处理器出放在ESC指令代码中的6位常数, 该常数指明协处理器要完成的功能.  当源操作数为存储器变量时, 则取出该存储器操作数传送给协处理器.	说明:  1. 其中EXTOPRD为外部操作码, OPRD为源操作数.  2. 本指不影响标志位.
391	NOP	空操作指令 NOP  格式: NOP  功能: 本指令不产生任何结果, 仅消耗几个时钟周期的时间, 接着执行后续指令, 常用于程序的延时等.	说明: 本指令不影响标志位.
392	LOCK	封锁总线指令 LOCK  格式: LOCK  功能: 指令是一个前缀, 可放在指令的前面, 告诉CPU在执行该指令时, 不允许其它设备对总线进行访问.	无可用信息!用户可自行添加!
393	IN	输入指令 IN	说明:

		格式：IN AL, n ; (AL)←-(n) IN AX, n ; (AX)←-(n+1), (n) IN AL, DX ; (AL)←-[(DX)] IN AX, DX ; (AX)←-[(DX)+1], [(DX)]  功能：输入指令	1. 其中n为8位的端口地址, 当字节输入时, 将端口地址n+1的内容送至AH中, 端口地址n的内容送AL中.  2. 端口地址也可以是16位的, 但必须将16位的端口地址送入DX中. 当字节寻址时, 由DX内容作端口地址的内容送至AL中;  当输入数据字时, [(DX)+1]送AH, [(DX)]送AL中, 用符号: (AX)←-[(DX)+1], [(DX)]表示.
394	OUT	输出指令 OUT  格式：OUT n, AL ; (n)←-(AL)  功能：输出指令	说明：  1. OUT n, AX ; (n+1), (n)←-(AX) OUT DX, AL ; [(DX)]←-(AL) OUT DX, AX ; [(DX)+1], [(DX)]←-(AX)  2. 输入指令及输出指令对标志位都不影响.
395	INTO	溢出中断指令 INTO (INTerrupt if Overflow)  格式：INTO  功能：本指令检测OF标志位, 当OF=1时, 说明已发生溢出, 立即产生一个中断类型4的中断, 当OF=0时, 本指令不起作用.	说明：  1. 本指令影响标志位IF及TF.  2. 本指令可用于溢出处理, 当OF=1时, 产生一个类型4的软中断. 在中断处理程序中完成溢出的处理操作.
396	INT	软中断指令 INT  格式：INT n 其中n为软中断的类型号.  功能：本指令将产生一个软中断, 把控制转向一个类型号为n的软中断, 该中断处理程序入口地址在中断向量表的n*4地址  处的二个存储器字(4个单元)中.	说明：操作过程与INTO指令雷同, 只需将10H改为n*4即可. 所以, 本指令也将影响标志位IF及TF.
397	IRET	中断返回指令 IRET  格式：IRET  功能：用于中断处理程序中, 从中断程序的断点处返回, 继续执行原程序.	说明：  1. 本指令将影响所有标志位.  2. 无论是软中断, 还是硬中断, 本指令均可使其返回到中断程序的断点处继续执行原程序.

一、状态寄存器

PSW (Program Flag)程序状态字寄存器，是一个16位寄存器，由条件码标志（flag）和控制标志构成，如下所示：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

- 条件码：
- ①OF (Overflow Flag)溢出标志。溢出时为1, 否则置0.
  - ②SF (Sign Flag) 符号标志。结果为负时置1, 否则置0.
  - ③ZF (Zero Flag)零标志，运算结果为0时ZF位置1, 否则置0.
  - ④CF (Carry Flag)进位标志，进位时置1, 否则置0.
  - ⑤AF (Auxiliary carry Flag) 辅助进位标志，记录运算时第3位（半个字节）产生的进位置。有进位时1, 否则置0.
  - ⑥PF (Parity Flag) 奇偶标志。结果操作数中1的个数为偶数时置1, 否则置0.

控制标志位：



- ⑦DF（Direction Flag）方向标志，在串处理指令中控制信息的方向。
- ⑧IF（Interrupt Flag）中断标志。
- ⑨TF（Trap Flag）陷井标志。

二、 直接标志转移（8位寻址）

指令格式	机器码	测试条件	如... 则转移	指令格式	机器码	测试条件	如... 则转移
JC	72	C=1	有进位	JNS	79	S=0	正号
JNC	73	C=0	无进位	J0	70	0=1	有溢出
JZ/JE	74	Z=1	零/等于	JNO	71	0=0	无溢出
JNZ/JNE	75	Z=0	不为零/不等于	JP/JPE	7A	P=1	奇偶位为偶
JS	78	S=1	负号	JNP/IPO	7B	P=0	奇偶位为奇

三、间接标志转移（8位寻址）

指令格式	机器码	测试格式	如... 则转移
JA/JNBE(比较无符号数)	77	C或Z=0	> 高于/不低于或等于
JAE/JNB(比较无符号数)	73	C=0	>= 高于或等于/不低于
JB/JNAE(比较无符号数)	72	C=1	< 低于/不高于或等于
JBE/JNA(比较无符号数)	76	C或Z=1	<= 低于或等于/不高于
JG/JNLE(比较带符号数)	7F	(S异或0) 或Z=0	> 大于/不小于或等于
JGE/JNL(比较带符号数)	7D	S异或0=0	>= 大于或等于/不小于
JL/JNGE(比较带符号数)	7C	S异或0=1	< 小于/不大于或等于
JLE/JNG(比较带符号数)	7E	(S异或0) 或Z=1	<= 小于或等于/不大于

四、无条件转移指令

操作码	伪码指令	含义
EB cb	JMP rel8	相对短跳转（8位），使rel8处的代码位下一条指令
E9 cw	JMP rel16	相对跳转（16位），使rel16处的代码位下一条指令
FF /4	JMP r/m16	绝对跳转（16位），下一指令地址在r/m16中给出
FF /4	JMP r/m32	绝对跳转（32位），下一指令地址在r/m32中给出
EA cb	JMP ptr16:16	远距离绝对跳转，下一指令地址在操作数中
EA cb	JMP ptr16:32	远距离绝对跳转，下一指令地址在操作数中
FF /5	JMP m16:16	远距离绝对跳转，下一指令地址在内存m16:16中
FF /5	JMP m16:32	远距离绝对跳转，下一指令地址在内存m16:32中

五、16位/32位寻址方式

操作码	伪码指令	跳转含义	跳转类型	跳转的条件（标志位）
0F 87 cw/cd	JA rel16/32	大于	near	(CF=0 and ZF=0)
0F 83 cw/cd	JAE rel16/32	大于等于	near	(CF=0)
0F 82 cw/cd	JB rel16/32	小于	near	(CF=1)
0F 86 cw/cd	JBE rel16/32	小于等于	near	(CF=1 or ZF=1)
0F 82 cw/cd	JC rel16/32	进位	near	(CF=1)
0F 84 cw/cd	JE rel16/32	等于	near	(ZF=1)
0F 84 cw/cd	JZ rel16/32	为0	near	(ZF=1)
0F 8F cw/cd	JG rel16/32	大于	near	(ZF=0 and SF=0F)
0F 8D cw/cd	JGE rel16/32	大于等于	near	(SF=0F)
0F 8C cw/cd	JL rel16/32	小于	near	(SF<>0F)
0F 8E cw/cd	JLE rel16/32	小于等于	near	(ZF=1 or SF<>0F)
0F 86 cw/cd	JNA rel16/32	不大于	near	(CF=1 or ZF=1)
0F 82 cw/cd	JNAE rel16/32	不大于等于	near	(CF=1)
0F 83 cw/cd	JNB rel16/32	不小于	near	(CF=0)
0F 87 cw/cd	JNBE rel16/32	不小于等于	near	(CF=0 and ZF=0)
0F 83 cw/cd	JNC rel16/32	不进位	near	(CF=0)
0F 85 cw/cd	JNE rel16/32	不等于	near	(ZF=0)
0F 8E cw/cd	JNG rel16/32	不大于	near	(ZF=1 or SF<>0F)
0F 8C cw/cd	JNGE rel16/32	不大于等于	near	(SF<>0F)

0F 8D	cw/cd	JNL rel16/32	不小于	near	(SF=0F)
0F 8F	cw/cd	JNLE rel16/32	不小于等于	near	(ZF=0 and SF=0F)
0F 81	cw/cd	JNO rel16/32	未溢出	near	(OF=0)
0F 8B	cw/cd	JNP rel16/32	不是偶数	near	(PF=0)
0F 89	cw/cd	JNS rel16/32	非负数	near	(SF=0)
0F 85	cw/cd	JNZ rel16/32	非零（不等于）	near	(ZF=0)
0F 80	cw/cd	J0 rel16/32	溢出	near	(OF=1)
0F 8A	cw/cd	JP rel16/32	偶数	near	(PF=1)
0F 8A	cw/cd	JPE rel16/32	偶数	near	(PF=1)
0F 8B	cw/cd	JPO rel16/32	奇数	near	(PF=0)
0F 88	cw/cd	JS rel16/32	负数	near	(SF=1)
0F 84	cw/cd	JZ rel16/32	为零（等于）	near	(ZF=1)

注：一些指令操作数的含义说明：

rel8 表示 8 位相对地址

rel16 表示 16 位相对地址

rel16/32 表示 16或32 位相对地址

r/m16 表示16位寄存器

r/m32 表示32位寄存器

test逻辑与运算结果为零,就把ZF(零标志)置1;

cmp 算术减法运算结果为零,就把ZF(零标志)置1