1. 通用数据传送指令

MOV 传送字或字节.

MOVSX 先符号扩展,再传送.

MOVZX 先零扩展,再传送.

PUSH 把字压入堆栈.

POP 把字弹出堆栈.

PUSHA 把 AX,CX,DX,BX,SP,BP,SI,DI 依次压入堆栈.

POPA 把 DI,SI,BP,SP,BX,DX,CX,AX 依次弹出堆栈.

PUSHAD 把 EAX,ECX,EDX,EBX,ESP,EBP,ESI,EDI 依次压入堆栈.

POPAD 把 EDI,ESI,EBP,ESP,EBX,EDX,ECX,EAX 依次弹出堆栈.

BSWAP 交换 32 位寄存器里字节的顺序

XCHG 交换字或字节.(至少有一个操作数为寄存器,段寄存器不可作为操作数)

CMPXCHG 比较并交换操作数.(第二个操作数必须为累加器 AL/AX/EAX)

XADD 先交换再累加.(结果在第一个操作数里)

XLAT 字节查表转换.

—— BX 指向一张 256 字节的表的起点, AL 为表的索引值 (0-255,即 0-FFH); 返回 AL 为查表结果. ([BX+AL]->AL)

2. 输入输出端口传送指令.

IN I/O 端口输入. (语法: IN 累加器, {端口号 | DX})

OUT I/O 端口输出. (语法: OUT {端口号 | DX},累加器)

输入输出端口由立即方式指定时, 其范围是 0-255; 由寄存器 DX 指定时, 其范围是 0-65535.

3. 目的地址传送指令.

LEA 装入有效地址. 例: LEA DX,string ;把偏移地址存到 DX.

LDS 传送目标指针,把指针内容装入 DS. 例: LDS SI,string ;把段地址:偏移地址存到 DS:SI.

LES 传送目标指针,把指针内容装入 ES. 例: LES DI,string ;把段地址:偏移地址存到 ES:DI.

LFS 传送目标指针,把指针内容装入 FS. 例: LFS DI,string ;把段地址:偏移地址存到 FS:DI.

LGS 传送目标指针,把指针内容装入 GS. 例: LGS DI,string ;把段地址:偏移地址存到 GS:DI.

LSS 传送目标指针,把指针内容装入 SS. 例: LSS DI,string ;把段地址:偏移地址存到 SS:DI.

4. 标志传送指令.

LAHF 标志寄存器传送,把标志装入 AH.

SAHF 标志寄存器传送,把 AH 内容装入标志寄存器.

PUSHF 标志入栈.

POPF 标志出栈.

PUSHD 32 位标志入栈.

POPD 32 位标志出栈.

折叠算术运算指令

ADD 加法.

ADC 带进位加法.

INC 加 1.

AAA 加法的 ASCII 码调整.

DAA 加法的十进制调整.

SUB 减法.

SBB 带借位减法.

DEC 减 1.

NEC 求反(以 0 减之).

CMP 比较.(两操作数作减法,仅修改标志位,不回送结果).

AAS 减法的 ASCII 码调整.

DAS 减法的十进制调整.

MUL 无符号乘法.

IMUL 整数乘法.

以上两条,结果回送 AH 和 AL(字节运算),或 DX 和 AX(字运算),

AAM 乘法的 ASCII 码调整.

DIV 无符号除法.

IDIV 整数除法.

以上两条,结果回送:商回送 AL,余数回送 AH, (字节运算);或 商回送 AX,余数回送 DX, (字运算).

AAD 除法的 ASCII 码调整.

CBW 字节转换为字. (把 AL 中字节的符号扩展到 AH 中去)

CWD 字转换为双字. (把 AX 中的字的符号扩展到 DX 中去)

CWDE 字转换为双字. (把 AX 中的字符号扩展到 EAX 中去)

CDQ 双字扩展. (把 EAX 中的字的符号扩展到 EDX 中去)

折叠逻辑运算指令

AND 与运算.

or 或运算.

XOR 异或运算.

NOT 取反.

TEST 测试.(两操作数作与运算,仅修改标志位,不回送结果).

SHL 逻辑左移.

SAL 算术左移.(=SHL)

SHR 逻辑右移.

SAR 算术右移.(=SHR)

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.

以上八种移位指令,其移位次数可达 **255** 次.

移位一次时，可直接用操作码. 如 SHL AX,1.

移位>1 次时，则由寄存器 CL 给出移位次数.

如 MOV CL,04

SHL AX,CL

折叠串指令

DS:SI 源串段寄存器 :源串变址.

ES:DI 目标串段寄存器:目标串变址.

CX 重复次数计数器.

AL/AX 扫描值.

D 标志 0 表示重复操作中 SI 和 DI 应自动增量; 1 表示应自动减量.

Z 标志 用来控制扫描或比较操作的结束.

MOVS 串传送.

(MOVSB 传送字符. MOVSW 传送字. MOVSD 传送双字.)

CMPS 串比较.

(CMPSB 比较字符. CMPSW 比较字.)

SCAS 串扫描.

把 AL 或 AX 的内容与目标串作比较,比较结果反映在标志位.

LODS 装入串.

把源串中的元素(字或字节)逐一装入 AL 或 AX 中.

(LODSB 传送字符. LODSW 传送字. LODSD 传送双字.)

STOS 保存串.

是 LODS 的逆过程.

REP 当 CX/ECX<>0 时重复.

REPE/REPZ 当 ZF=1 或比较结果相等,且 CX/ECX<>0 时重复.

REPNE/REPNZ 当 ZF=0 或比较结果不相等,且 CX/ECX<>0 时重复.

REPC 当 CF=1 且 CX/ECX<>0 时重复.

REPNC 当 CF=0 且 CX/ECX<>0 时重复.

折叠程序转移指令

1>无条件转移指令 (长转移)

JMP 无条件转移指令

CALL 过程调用

RET/RETF 过程返回.

2>条件转移指令 (短转移,-128 到+127 的距离内)

(当且仅当(SF XOR OF)=1 时,OP1<OP2)

JA/JNBE 不小于或不等于时转移.

JAE/JNB 大于或等于转移.

JB/JNAE 小于转移.

JBE/JNA 小于或等于转移.

以上四条,测试无符号整数运算的结果(标志 C 和 Z).

JG/JNLE 大于转移.

JGE/JNL 大于或等于转移.

JL/JNGE 小于转移.

JLE/JNG 小于或等于转移.

以上四条,测试带符号整数运算的结果(标志 S,O 和 Z).

JE/JZ 等于转移.

JNE/JNZ 不等于时转移.

JC 有进位时转移.

JNC 无进位时转移.

JNO 不溢出时转移.

JNP/JPO 奇偶性为奇数时转移.

JNS 符号位为 "0" 时转移.

JO 溢出转移.

JP/JPE 奇偶性为偶数时转移.

JS 符号位为 "1" 时转移.

3>循环控制指令(短转移)

LOOP CX 不为零时循环.

LOOPE/LOOPZ CX 不为零且标志 Z=1 时循环.

LOOPNE/LOOPNZ CX 不为零且标志 Z=0 时循环.

JCXZ CX 为零时转移.

JECXZ ECX 为零时转移.

4>中断指令

INT 中断指令

INTO 溢出中断

IRET 中断返回

5>处理器控制指令

HLT 处理器暂停, 直到出现中断或复位信号才继续.

WAIT 当芯片引线 TEST 为高电平时使 CPU 进入等待状态.

ESC 转换到外处理器.

LOCK 封锁总线.

NOP 空操作.

STC 置进位标志位.

CLC 清进位标志位.

CMC 进位标志取反.

STD 置方向标志位.

CLD 清方向标志位.

STI 置中断允许位.

CLI 清中断允许位.

折叠伪指令

DW 定义字(2 字节).

PROC 定义过程.

ENDP 过程结束.

SEGMENT 定义段.

ASSUME 建立段寄存器寻址.

ENDS 段结束.

END 程序结束.

处理机控制指令：

标志处理指令 CLC（进位位置 0 指令）

CMC（进位位求反指令）

STC（进位位置为 1 指令）

CLD（方向标志置 0 指令）

STD（方向标志位置 1 指令）

CLI（中断标志置 0 指令）

STI（中断标志置 1 指令）

NOP（无操作）

HLT（停机）

WAIT（等待）

ESC（换码）

LOCK（封锁）

详细说明

1.MOV（传送）
指令写法：MOV target，source
功能描述：将源操作数 source 的值复制到 target 中去，source 值不变
注意事项：1）target 不能是 CS（代码段寄存器），我的理解是代码段不可写，只可读，所以相应这地方也不能对 CS 执行复制操作。2）target 和 source 不能同时为内存数、段寄存器（CS\DS\ES\SS\FS\GS）3)不能将立即数传送给段寄存器 4）target 和 source 必须类型匹配，比如，要么都是字节，要么都是字或者都是双字等。4）由于立即数没有明确的类型，所以将立即数传送到 target 时，系统会自动将立即数零扩展到与 target 数的位数相同，再进行传送。有时，需要用 BYTE PTR 、WORD PTR、 DWORD PTR 明确指出立即数的位数
写法示例：MOV dl,01H;MOV eax,[bp]; eax =ss:[bp] 双字传送。
2、 XCHG(交换)
指令写法：XCHG object1，object2
功能描述：交换 object1 与 object2 的值
注意事项：1）不能直接交换两个内存数的值 2）类型必须匹配 3）两个操作数任何一个都不能是段寄存器【看来段寄存器的写入的限制非常的严格，MOV 指令也不能对段寄存器进行写入】，4）必须是通用寄存器（ax、bx、cx、dx、si、di）或内存数
写法示例：XCHG ax，[bx][si]; XCHG ax,bx;
3、 LEA(装入有效地址)
指令写法：LEZ reg16，mem
功能描述：将有效地址 MEM 的值装入到 16 位的通用寄存器中。
写法示例：假定 bx=5678H，EAX=1,EDX=2

```
Lea si,2[bx]            ;si=567AH
Lea di,2[eax][edx]      ;di=5
```

注意，这里装入的是有效地址，并不是实际的内存中的数值，如果要想取内存中该地址对应的数值，还需要加上段地址才行，而段地址有可能保存在 DS 中，也有可能保存在 SS 或者 CS 中哦:>不知道我的理解可正确。。。。
4、 LDS\LES\LGS\LSS（注意，与 LEA 不同的是，这里是装入的值，而不是有效地址）
这几个指令，名称不同，作用差不多。
写法：LDS reg16，mem32
功能描述：reg16 等于 mem32 的低字，而 DS 对应于 mem32 的高字（当为 LES 时，这里就是 ES 对应于 mem32 的高字）
用来给一个段寄存器和一个 16 位通用寄存器同时复制。
注意事项：第一个操作数必须是 16 位通用寄存器

在接着往下说之前，先熟悉下堆栈的概念。堆栈，位于内存的堆栈段中，是内存的一部分，具有“先进后出”的特点，堆栈只有一个入口，即当前栈顶，当堆栈为空时，栈顶和栈底指向同一内存地址，在 WINDOWS 中，可以把堆栈理解成一个倒着的啤酒瓶，上面的地址大，下面的地址小，当从瓶口往啤酒瓶塞啤酒时（进栈），栈顶就会往瓶口下移动，也就是往低地址方向移动，同理，出栈时，正好相反，把啤酒给倒出来，栈顶向高地址方向移动。这就是所谓的堆栈，哼哼，很 Easy 吧。
在汇编语言中，堆栈操作的最小单位是字，也就是说，只能以字或双字为单位，同时，SS：SP 指向栈顶（SS 为堆栈段寄存器，SP 为堆栈指针，二者一相加，就构成了堆栈栈顶的内存地址）。
5、 PUSH（进栈）
写法：PUSH reg16（32）/seg/mem16（32）/imm
功能描述：将通用寄存器/段寄存器/内存数/立即数的值压入栈中，即：

SP=SP-2 SS:[SP]=16 位数值（当将 32 位数值压入栈中时，SP=SP-4，SS:[SP]=32 为数值）

6、 POP（出栈）

写法：POP reg16（32）/seg/mem16（32）【不能出栈到 CS 中】

功能描述：将堆栈口的 16（32）位数据推出到通用寄存器/段寄存器/内存中，即：

寄存器/段寄存器/内存= SS:[SP] SP=SP+2（当将 32 位数值出栈时，SP=SP+4）（注意，不能出栈给立即数哦，常量不可变嘛）

7、 PUSHA、PUSHAD、POPA、POPAD

作用：将所有 16/32 位通用寄存器进栈/出栈

如：PUSHA ;将 AX、CX、DX、BX、原 SP、BP、SI、DI 依次进栈。POPA 出栈顺序正好相反，但要注意的是，弹出到 SP 的值被丢弃，SP 通过增加 16 位来恢复（当然嘛，不然栈顶地址就被修改了，就会出息不对齐的情况，就有可能乱套了）

POPAD PUSHAD 一样，只不过是 32 位的罢了。

8、 PUSHF、PUSHFD、POPF、POPFD

功能描述：标志寄存器 FLAGS（EFLAGS）进栈或出栈

如：PUSHF ； FLAGS 进栈 POPF； 栈顶字出栈到 FLAGS

总结下，POP 和 PUSH 通常可以用来交换两个寄存器的值，也可以用来保护寄存器的值，如下：

交换 ax 与 cx 的值：push ax； push cx； pop ax； pop cx；

保护寄存器：push ax； push cx； ….中间有很多执行的代码…pop cx;pop ax;

9、LAHF\SAHF（标志寄存器传送指令）

写法：lahf；

作用：AH=FLAGS 的低 8 位

写法：sahf；

作用：FLAGS 的低 8 位=AH

10、符号扩展和零扩展指令

CBW；AL 符号扩展为 AX

CWD；AX 符号扩展为 32 位数 DX:AX

CWDE;AX 符号扩展为 EAX；

CDQ：EAX 符号扩展为 64 位数 EDX:EAX

MOVSX（符号扩展指令的一般形式）

写法：MOVSX reg16\32，reg8\reg16\mem8\mem16

作用：用来将 8 位符号扩展到 16 位，或者 16 位符号扩展到 32 位

MOVZX（零扩展指令）

写法：MOVZX reg16\32，reg8\reg16\mem8\mem16

零扩展，就是高位补 0 进行扩展。通常用在将数据复制到一个不同的寄存器中，如 AL 零扩展为 EBX。相同寄存器的零扩展，可以使用 MOV 高位，0 来实现。

11、BSWAP（字节交换）

写法：bswap reg32

作用：将 reg32 的第 0 与第 3 个字节，第 1 与第 2 个字节进行交换。

示例：设 EAX=12345678h

执行 bswap eax；后，eax=78563412H

12、XLAT（换码）

写法：XLAT；

作用：AL=DS:[bx+AL]

将 DS:BX 所指内存中的由 AL 指定位移处的一个字节赋值给 AL。（貌似这是一个方便偷懒的指令哦。。），原来它的主要用途是查表。注意可以给它提供操作数，用来指定使用哪个段地址，如：

XLAT ES：table；使用 ES 来作为段地址，table 不起作用。

XLAT table ；使用 table 所在段对应的段寄存器作为段地址。

----------数据传送指令结束----------

----------算术指令----------

13、ADD（加法）

写法：ADD reg/mem reg/mem/imm

作用：将后面的操作数加到前面的操作数中

注意：两个操作数必须类型匹配，并且不能同时是内存操作数

ADC （带进位加法）

写法：ADC reg/mem, reg/mem/imm ；

作用：dest=dest+src+cf

当 CF=0 时 ADD 与 ADC 的作用是相同的。

示例：实现 64 位数 EDX:EAX 与 ECX:EBX 的加法：

Add EAX,EBX；

ADC EDX,ECX;

14、INC（自加一）

写法：INC reg/mem；

作用：dest=dest+1；

15、XADD（交换加）

写法：XADD reg/mem， reg

作用：先将两个数交换，然将二者之和送给第一个数

16、SUB（减法）

写法：SUB reg/mem， reg/mem/imm；

作用：dest=dest-src；
SBB（带借位减法）
写法：SBB reg/mem， reg/mem/imm
作用：dest=dest-src-cf；
注意：两个操作数必须类型匹配，且不能同时是内存数
17、DEC（自减 1）
写法：DEC reg/mem；
作用：dest=dest-1；
18、CMP（比较）
写法：CMP reg/mem， reg/mem/imm
作用：dest-src
注意：这里并不将结果存入 dest 中，而仅仅是执行相减的运算，达到依据运算结果去影响 EFLAG 标志位的效果
19、NEG（求补）
写法：NEG reg/mem
作用：求补就是求相反数，即：dest=0-dest；
20、CMPXCHG(比较交换)
写法：CMPXCHG reg/mem， reg；
作用：AL/AX/EAX-oprd1，如果等于 0，则 oprd1=oprd2，否则，AL/AX/EAX=oprd1；
即：比较 AL/AX/EAX 与第一个操作数，如果相等，则置 ZF=1,并复制第二个操作数给第一个操作数；否则，置 ZF=0，并复制第一个操作数给 AL/AX/EAX。
说明：CMPXCHG 主要为实现原子操作提供支持
CMPXCHG8B（8 字节比较交换指令）
写法：CMPXCHG8B MEM64;
功能：将 EDX:EAX 中的 64 位数与内存的 64 位数进行比较，如果相等，则置 ZF=1，并存储 ECX:EBX 到 mem64 指定的内存地址；否则，置 ZF=0，并设置 EDX:EAX 为 mem64 的 8 字节内容
21、MUL（无符号乘法）
写法：MUL reg/mem；
作用：当操作数为 8 位时，AX=AL*src；
当操作数为 16 位时，DX:AX=AX*src；
当操作数为 32 位时，EDX:EAX=EAX*src；
22、IMUL(带符号位乘法)
写法：IMUL reg/mem；（作用同上）
IMUL reg16，reg16/mem16，imm16；
IMUL reg32，reg32/mem32，imm32；
IMUL reg16，imm16/reg16/imm16；
IMUL reg32，reg32/mem32/imm32；
注意：没有两个操作数均为 8 位的多操作数乘法。
对于同一个二进制数，采用 MUL 和 IMUL 执行的结果可能不同，设 AL=0FF，BL=1，分别执行下面的指令，会得到不同的结果：
Mul bl； AX=0FFH(255);
Imul bl； AX=0FFFFH（-1）（高一半为低一半的扩展）
23、DIV（无符号除法 ）/IDIV(带符号数除法)
写法：DIV reg/mem；/IDIC reg/mem
作用：如果操作数是 8 位，AX%SRC，结果商在 AL、余数在 AH 中；
如果操作数是 16 位，DX:AX%SRC，结果商在 AX,余数在 DX 中；
如果操作数是 32 位，EDX:EAX%SRC，结果商在 EAX,余数在 EDX 中；
注意：不能直接实现 8 位数除 8 位数、16 位数除 16 位数、32 除 32，若需要这样，则必须先把除数符号扩展或零扩展到 16、32、64 位，然后用除法指令。
对于 IDIV，余数和被除数符号相同，如：-5 IDIV 2 = 商 -2，余数：-1；
在下列情况下，会使 CPU 产生中断：一：除数为 0 ；二：由于商太大，导致 EAX\AX 或 AL 不能容纳，从而产生了溢出。
----------BCD 码调整指令（十进制调整指令）待补充----------
24、关于 BCD 码：BCD 码就是一种十进制数的二进制编码表示，分为压缩 BCD 码和非压缩 BCD 码，压缩 BCD 码用 4 个二进制位表示一个十进制位，即用 0000B~1001B 表示十进制 0~9，如 0110 0100 0010 1001B 表示 6429
用 8 位二进制来表示一个十进制叫非压缩 BCD 码，其中，低四位与压缩 BCD 码相同，高四位无意义。
压缩 BCD 码调整指令包括 DAA(加法的压缩 BCD 码调整)和 DAS（减法的压缩 BCD 码调整）
写法：
DAA;
作用：调整 AL 中的和为压缩 BCD 码。
功能：使用 DAA 指令时，通常先执行 ADD/ADC 指令，将两个压缩 BCD 码相加，结果存放在 AL 中，然后使用该指令将 AL 调整为压缩 BCD 码格式。
DAA 的调整[算法]：
IF(AL 低 4 位>9 或 AF=1)
THEN
AL=AL+6;
AF=1;
ENDIF
IF(AL 高 4 位>9 或 CF=1)
THEN
AL=AL+60H;
CF=1;

ENDIF
说明：CF 反映压缩 BCD 码相加的进位。
DAS;
作用：调整 AL 中的差为压缩 BCD 码。
功能：使用 DAS 指令时，通常先执行 SUB/SBB 指令，将两个压缩 BCD 码相减，结果存放在 AL 中，然后使用该指令将 AL 调整为压缩 BCD 码格式。
DAS 的调整算法：
IF(AL 低 4 位>9 或 AF=1)
THEN
AL=AL-6;
AF=1;
ENDIF
IF(AL 高 4 位>9 或 CF=1)
THEN
AL=AL-60H;
CF=1;
ENDIF
说明：CF 反映压缩 BCD 码相减的借位。
特别注意，如果使用 DAA 或 DAS 指令，则参加加法或减法运算的操作数应该是压缩 BCD 码，如果将任意两个二进制数相加或相减，然后调整，则得不到正确的结果。
关键是调整的规则，其中 AF 标志位就是专门为 BCD 码调整设计的，当低四位有向高四位进位或借位时，值为 1。而 CF 就是最高位有进位或者借位时，为 1.
非压缩 BCD 码调整指令，包括 AAA,AAS,AAM,AAD。
写法：AAA ;
作用：调整 AL 中的和为非压缩 BCD 码；调整后，AL 高 4 位等于 0，AH=AH+产生的 CF
功能：使用 AAA 指令时，通常先执行 ADD/ADC 指令，以 AL 为目的操作数，将两个非压缩 BCD 码（与高位无关）相加，然后使用 AAA 将 AL 调整为非压缩 BCD 码格式，且高 4 位等于 0，同时，将调整产生的进位加到 AH 中。
AAA 调整算法：
IF(AL 低 4 位>9 或者 AF=1)
THEN
AL=AL+6;
AH=AH+1;
AF=1;
CF=1;
ELSE
AF=0;CF=0;
ENDIF
AL=AL AND OFH;;AL 高 4 位清 0
写法：AAS ;
作用：调整 AL 中的差为非压缩 BCD 码；调整后，AL 高 4 位等于 0，AH=AH-产生的 CF
功能：使用 AAS 指令时，通常先执行 SUB/SBB 指令，以 AL 为目的操作数，将两个非压缩 BCD 码（与高位无关）相减，然后使用 AAS 将 AL 调整为非压缩 BCD 码格式，且高 4 位等于 0，同时，将调整产生的借位从 AH 中减去。
AAA 调整算法：
IF(AL 低 4 位>9 或者 AF=1)
THEN
AL=AL-6；
AH=AH-1；
AF=1;
CF=1;
ELSE
AF=0;CF=0;
ENDIF
AL=AL AND OFH;;AL 高 4 位清 0
写法：AAM；
作用：AH=AX DIV 10, AL=AX MOD 10;
功能：使用 AAM 时，通常先执行 MUL/IMUL 指令，将两个一字节非压缩 BCD 码（高四位必须为 0）相乘，结果存入 AX.然后使用 AAM 指令将 AX（AH=0）调整为两字节压缩 BUC 码格式。
写法：AAD;
作用：AL=AH*10+AL,AH=0;
功能：使用 AAD 时，通常先执行该指令，将 AX 中的两字节非压缩 BCD 码（AH 与 AL 的高 4 位必须为 0）调整为相应的二进制表示，然后使用 DIV/IDIV 指令，除以一个一字节的非压缩 BCD 码（高四位必须为 0），可得到非压缩 BCD 码的除法结果。
特别注意，参加非压缩 BCD 码乘法或除法的操作数高 4 位必须为 0。
----------算术指令结束----------
----------位操作指令----------
25、AND\OR\XOR\NOT\TEST
写法：
AND reg/mem,reg/mem/imm;
OR reg/mem,reg/mem/imm;

XOR reg/mem,reg/mem/imm;
NOT reg/mem;
TEST reg/mem,reg/mem/imm;
作用：AND\TEST\OR\XOR，两个操作数必须类型匹配，而且不能同时是内存操作数。
XOR 通常用来将寄存器清 0，如 XOR AX,AX;
TEST 与 AND 的关系类似于 CMP 与 SUB。TEST 的典型用法是检查某位是否为 1，如：
TEST DX,109H；
若 DX 的第 0，3，8 位至少有一位为 1，则 ZF=0，否则 ZF=1；
26、移位指令
SHL(逻辑左移)
写法：SHL REG\mem，1\CL ;
作用：将 dest 的各个二进制位向左移动 1（CL）位，并将 DEST 的最高位移出到 CF，最低位移入 0。
SAL（算术左移）
写法：SAL REG\mem，1\CL ;
作用：将 dest 的各个二进制位向左移动 1（CL）位，并将 DEST 的最高位移出到 CF，最低位移入 0（同 SHL）。
SHR（逻辑右移）
写法：SHR REG\mem，1\CL ;
作用：将 dest 的各个二进制位向左移动 1（CL）位，并将 DEST 的最低位移出到 CF，最高位移入 0。
SAR(算术右移)
写法：SAR REG\mem，1\CL ;
作用：将 dest 的各个二进制位向左移动 1（CL）位，并将 DEST 的最低位移出到 CF，最高位不变。
SHLD(双精度左移)
写法：SHLD REG16/REG32/MEM16/MEM32, REG16/REG32, IMM8/CL;(类型须匹配)
作用：将 OPRD1 的各二进制左移，并将 oprd1 的最高位移到 CF,oprd2 的最高位移到 oprd1 的最低位，但是，oprd2 的值不变。
SHRD（双精度右移）
写法与作用与双精度左移类似。注意移动方向为右移。
以上位移指令对标志位的影响：
若移位后符号位发生了变化，则 OF=1，否则 OF=0;CF 为最后移入位；按一般规则影响 ZF 与 SF。然而，若移位次数为 0，则不影响标志位；若移位次数大于 1，则 OF 无定义。
27、循环移位指令
ROL(循环左移)
写法：ROL REG\MEM, 1\CL；或 ROL REG/MEM,IMM8;(类型可不匹配)
作用：将 DEST 的各二进制位向左移动，并将最高位移出到 CF,并同时移入最低位。
ROR(循环右移)
写法：ROR REG\MEM, 1\CL；或 ROR REG/MEM,IMM8;(类型可不匹配)
作用：将 DEST 的各二进制位向右移动，并将最低位移出到 CF,并同时移入最高位。
RCL(带进位循环左移)
写法：RCL REG\MEM, 1\CL；或 RCL REG/MEM,IMM8;(类型可不匹配)
作用：将 DEST 的各二进制位向左移动，并将最高位移出到 CF，原 CF 移入最低位。
RCR(带进位循环右移)
写法：RCR REG\MEM, 1\CL；或 RCR REG/MEM,IMM8;(类型可不匹配)
作用：将 DEST 的各二进制位向右移动，并将最低位移出到 CF，原 CF 移入最高位。
28、位测试指令
BT（位测试）
写法：BT REG16/MEM16,REG16/IMM8;或 BT REG32/MEM32,REG32/IMM8;
作用：CF=DEST 的第 index 位，dest 不变。
BTS（位测试并置位）
写法：BTS REG16/MEM16,REG16/IMM8;或 BTS REG32/MEM32,REG32/IMM8;
作用：CF=DEST 的第 index 位，dest 的第 index 位=1；
BTR(位测试并复位)
写法：BTR REG16/MEM16,REG16/IMM8;或 BTR REG32/MEM32,REG32/IMM8;
作用：CF=DEST 的第 index 位，dest 的第 index 位=0；
BTC(位测试并复位)
写法：BTC REG16/MEM16,REG16/IMM8;或 BTC REG32/MEM32,REG32/IMM8;
作用：CF=DEST 的第 index 位，dest 的第 index 位取反；
说明：若 dest 为寄存器，则以 index 除以 16（dest 为 reg16）或 32（dest 为 reg32）的余数作为测试位。当然，index 最好不要超出操作数的位数。若 dest 为内存操作数，则无论其类型为字或双字，测试位为相对于起始地址的位移，例如,设 BX=50，X 为字类型的变量，则执行指令 BT X,BX；后，CF=X+6 单元的第 2 位，因为 50%8=6 余 2.
BTS、BTC、BTR 指令可用于并发程序设计。
29、位扫描指令
BSF(前向位扫描)
写法：BSF reg16/reg32， reg16/reg32/mem16/mem32；（类型须匹配）
作用：dest=src 中值为 1 的最低位编号（从低位向高位搜索）
BSR(后向位扫描)
写法：BSR reg16/reg32， reg16/reg32/mem16/mem32；（类型须匹配）
作用：dest=src 中值为 1 的最高位编号（从高位向低位搜索）
说明：BSF 和 BSR 搜索 SRC 操作数中首次出现 1 的位置，BSF 从低位向高位搜索，BSR 反之。若找到一个 1，则置 ZF=0，并存储位编号到 DEST 操

作数中。若 SRC=0，即没有 1 出现，则置 ZF=1，且 dest 的值不确定。
比如，有如下二进制数 0111 1111 1010 0100
执行 bsf 后，位编号为 2，执行 bsr 后，位编号为 14.
30、条件置位指令
通用写法：SETcc reg8/mem8
作用：若条件 cc 成立，则 dest=1，否则，dest=0；
SETcc 有很多种命令形式，这里的 cc 只是一个描述符，具体的参见下面的三个表，其中，E（Equal）表示相等，G（Greatet）表示带符号大于，L（Less）表示带符号小于，A（Above）表示无符号大于，B（Below）表示无符号小于。
表一：测试单个标志位的 SETcc 指令：

SETcc 指令	描述	置 1 条件
SETC,SETB,SETNAE	有进位时置 1	CF=1
SETNC,SETNB,SETAE	无进位时置 1	CF=0
SETZ,SETE	为 0（相等）时置 1	ZF=1
SETNA,SETNE	非 0（不等）时置 1	ZF=0
SETS	为负时置 1	SF=1
SETNS	为正时置 1	SF=0
SET0	溢出时置 1	OF=1
SETNO	不溢出时置 1	OF=0
SETP,SETPE	‘1’的个数为偶数时置 1	PF=1
SETNP,SETPO	‘1’的个数为奇数时置 1	PF=0

表二：用于带符号数比较的 SETcc 指令，这些指令常用在 CMP 指令之后，以判断带符号数的大小：

SETcc 指令	描述	置 1 条件
SETG,SETNLE	大于（不小于等于）时置 1	SF=OF 且 ZF=0
SETGE,SETNL	大于等于（不小于）时置 1	SF=OF
SETL,SETNGE	小于（不大于等于）时置 1	SF≠OF
SETLE,SETNG	小于等于（不大于）时置 1	SF≠OF 或 ZF=1

表三：用于无符号数比较的 SETcc 指令，常用在 CMP 指令之后，用来判断无符号数的大小：

SETcc 指令	描述	置 1 条件
SETA,SETNBE	大于（不小于等于）时置 1	CF=0 且 ZF=0
SETAE,SETNB,SETNC	大于等于（不小于）时置 1	CF=0
SETB,SETNAE,SETC	小于（不大于等于）时置 1	CF=1
SETBE,SETNA	小于等于（不大于）时置 1	CF=1 或 ZF=1

----------位操作指令结束----------
汇编语言指令详讲(2011-05-13 17:31:32)

标签： **it**	分类： 汇编-C-C-java-VB 编程

AAA	未组合的十进制加法调整指令 AAA(ASCII Adgust for Addition) 格式: AAA 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.	说明: 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. 当一个节(8 位)中存放一位 BCD 码,且放在字节的低 4 位, 高 4 位为时称为未组合的 BCD 码. 2. AAA 的调整操作 若(AL) and 0FH>9 或 AF=1,则调整如下: (AL)<--(AL)+6,(AH)<--(AH)+1,AF=1,CF<--AF,(AL)<--(AL) and 0FH
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).
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	说明: 1. 实际上是两个未组合的十进制数字节相乘,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)<--(AL)/10 (AL)<--(AL)MOD10 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).

	十进制数存在 AL 中.	
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 0FH, 否则(AL)<--(AL) and 0FH
ADC	带进位加法指令 ADC(Addition Carry) 格式: ADC OPRD1,OPRD2 功能: OPRD1<--OPRD1 + OPRD2 + CF	说明: 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. 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. 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. 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. 4. 该指令对标志位的影响同 ADD 指令.
ADD	加法指令 ADD(Addition) 格式: ADD OPRD1,OPRD2 功能: 两数相加	说明: 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. 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. 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. 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.
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
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.
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
CLC	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.
CLD	处理器控制指令－标志位操作指令 格式: CLC ;置 CF=0 STC ;置 CF=1	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.

	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.	
CLI	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.
CMC	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.
CMP	比效指令 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后,实现条件转移的指令.
CMPS	字符串比较指令 格式: CMPS OPRD1,OPRD2 CMPSB CMPSW 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.	说明: 1. 其中 OPRD2 为源串符号地址,OPRD1 为目的串符号地址. 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. 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.
CMPSB	字符串比较指令 格式: CMPS OPRD1,OPRD2 CMPSB CMPSW 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.	说明: 1. 其中 OPRD2 为源串符号地址,OPRD1 为目的串符号地址. 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. 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.

	同时 SI,DI 将自动调整.	
CMPSW	字符串比较指令 格式: CMPS OPRD1,OPRD2 CMPSB CMPSW 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.	说明: 1. 其中 OPRD2 为源串符号地址,OPRD1 为目的串符号地址. 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. 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.
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。
DAA	组合的十进制加法调整指令 DAA(Decimal Adjust for Addition) 格式: DAA 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.	说明: 1. 调整操作如下 (1) 若(AL) and 0FH>9 或 AF=1,则(AL)<--(AL)+6,AF<--1,对低四位的调整. (2) 若(AL) and 0F0H>90H 或 CF=1,则(AL)<--(AL)+60H,CF<--1. 2. 示例: (AL)=18H,(BL)=06H ADD AL,BL ; (AL)<--(AL)+(BL) ; (AL)=1EH DAA ; (AL)
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
DEC	减一指令 DEC(Decrement by 1) 格式: DEC OPRD 功能: OPRD<--OPRD-1	说明: 1. OPRD 为寄存器或存储器操作数. 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. 3. 示例 DEC AX DEC CL DEC WORD PTR[DI] DEC ALFA[DI+BX]
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)<--(AX)/OPRD,(AH)<--(AX)MOD OPRD 字除法: (AX)<--(DX)(AX)/OPRD,(DX)<--(DX)(AX) MOD OPRD
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.
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.
IDIV	带符号数除法指定	说明:

	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.	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.
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.
IN	输入指令 IN 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	说明: 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.
INC	加 1 指令 INC(INCrement by 1) 格式: INC OPRD 功能: OPRD<--OPRD+1	说明: 1. OPRD 为寄存器或存储器操作数. 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. 3. 示例: INC SI;(SI)<--(SI)+1 INC WORD PTR[BX] INC BYTE PTR[BX+DI] INC CL;(CL)<--(CL)+1 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.
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.
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.
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.
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
JAE	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.

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
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
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
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
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
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
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
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
JMP	无条件转移指令 JMP 格式: JMP OPRD 功能: JMP 指令将无条件地控制程序转移到目的地址去执行.当目的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址,在段内转移时,指令只要能提供目的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址的段地址及段内偏移地址值.	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.
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
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
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.

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
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
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
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
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
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
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
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
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
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
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
JO	条件转移指令 JO 格式: JO 标号 功能: OF＝1,转至标号处执行	说明: JO 是根据溢出标志位 OF 进行转移的指令
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
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
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
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

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
LAHF	标志传送指令 LAHF 格式: LAHF 功能: 取 FLAG 标志寄存器低 8 位至 AH 寄存器.(AH)<--(FLAG)7~0	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.

LDS	从存储器取出 32 位地址的指令 LDS
	格式: LDS OPRD1,OPRD2
	功能: 从存储器取出 32 位地址的指令.
		说明:
		OPRD1 为任意一个 16 位的寄存器.
		OPRD2 为 32 位的存储器地址.
		示例: LDS SI,ABCD
		LDS BX,FAST[SI]
		LDS DI,[BX]
		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 DI.

LEA	有效地址传送指令 LEA 格式: LEA OPRD1,OPRD2 功能: 将源操作数给出的有效地址传送到指定的的寄存器中.	说明: 1. OPRD1 为目的操作数,可为任意一个 16 位的通用寄存器. OPRD2 为源操作数,可为变量名、标号或地址表达式. 示例: LEA BX,DATA1 LEA DX,BETA[BX+SI] LEA BX BX,[BP],[DI] 2. 本指令对标志位无影响。
LES	从存储器取出 32 位地址的指令 LES 格式: LES OPRD1,OPRD2 功能: 从存储器取出 32 位地址的指令.	说明: OPRD1 为任意一个 16 位的寄存器. OPRD2 为 32 位的存储器地址. 示例: LES SI,ABCD LES BX,FAST[SI] LES DI,[BX] 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.
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!
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.
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.
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.
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.
LOOPNZ	循环控制指令 LOOPNZ/LOOPNE	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.

	格式: LOOPNZ/LOOPNE 标号 功 能 : (CX)<--(CX)-1,(CX)<>0 且 ZF＝0 时,转至标号处循环	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.
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.
MOVE	数据传送指令 MOV 格式: MOV OPRD1,OPRD2 功能: 本指令将一个源操作数送到目的操作数中,即 OPRD1<--OPRD2.	说明: 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. 2. MOV 指令以分为以下四种情况: < 1> 寄存器与寄存器之间的数据传送指令 < 2> 立即数到通用寄存器数据传送指令 < 3> 寄存器与存储器之间的数据传送指令 < 4> 立即数到存储器的数据传送 3. 本指令不影响状态标志位
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.
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.
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.
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)<--(AL)*OPRD,当结果的高位字节(AH)不等于 0 时,则 CF＝1、OF＝1.
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.
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.
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),结果关回原操作数.	说明: 1. 其中 OPRD 可为任一通用寄存器或存储器操作数. 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.
OR	逻辑或指令 OR	说明:

	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.	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.
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.
POP	堆栈操作指令 PUSH 和 POP 格式: PUSH OPRD POP OPRD 功能: 实现压入操作的指令是 PUSH 指令;实现弹出操作的指令是 POP 指令.	说明: 1. OPRD 为 16 位(字)操作数,可以是寄存器或存储器操作数. 2. POP 指令的操作过程是: POP OPRD:OPRD<--((SP)),(SP)<--(SP)+2 它与压入操作相反,是先弹出栈顶的数顶,然后再修改指针 SP 的内容. 3. 示例: POP AX POP DS POP DATA1 POP ALFA[BX][DI] 4. PUSH 和 POP 指令对状态标志位没有影响.
POPF	标志传送指令 POPF 格式: POPF 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.	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.
PUSH	堆栈操作指令 PUSH 和 POP 格式: PUSH OPRD POP OPRD 功能: 实现压入操作的指令是 PUSH 指令;实现弹出操作的指令是 POP 指令.	说明: 1. OPRD 为 16 位(字)操作数,可以是寄存器或存储器操作数. 2. PUSH 的操作过程是: (SP)<--(SP)-2,((sp))<--OPRD 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. 此处的((SP))<--OPRD,也可以理解为: [(SS)*16+(SP)]<--OPRD 或 [SS:SP]<--OPRD
PUSHF	标志传送指令 PUSHF 格式: PUSHF 功能: 本指令可以把标志寄存器的内容保存到堆栈中去	
RCL	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.
RCR	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.

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.
REPE	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. 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 指令结合使用,可以完成在一个字符串中搜索一个关键字. 3. 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. 4. REPNZ/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.
REPNE	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 指令结合使用,可以完成在一个字符串中搜索一个关键字. 3. 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. 4. REPNZ/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.
REPNZ	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 指令结合使用,可以完成在一个字符串中搜索一个关键字. 3. 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. 4. REPNZ/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.
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 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. 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 指令结合使用,可以完成在一个字符串中搜索一个关键字. 3. 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. 4. REPNZ/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.
RET	返回指令 RET 格式: RET 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.	说明: 由于在过程定义时,已指明其近(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.

ROL	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.
ROR	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.
SAHF	标志传送指令 SAHF 格式: SAHF 功能: 将 AH 存至 FLAG 低 8 位	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.
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.
SAR	算术右移指令 SAR 格式: SAR OPRD1,COUNT 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.	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.
SBB	带借位减去指令 SBB(SuBtraction with Borrow) 格式: SBB OPRD1,OPRD2 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<--OPRD1-OPRD2-CF,其结果放在 OPDR1 中.	说明: 示例 SBB DX,CX SBB AX,DATA1 SBB BX,2000H SBB ALFA[BX+SI],SI SBB BETAP[DI,030AH
SCAS	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.

SCASB	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.
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.
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
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.
STC	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.
STD	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	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.

	外部中断 功能: 完成对标志位的置位、复位等操作.	
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.
STOS	字符串存储指令 STOS 格式: STOS OPRD 功能: 把 AL(字节)或 AX(字)中的数据存储到 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.

SUB	减法指令 SUB(SUBtract)
	格式: SUB OPRD1,OPRD2
	功能: 两个操作数的相减,即从 OPRD1 中减去 OPRD2,其结果放在 OPDR1 中.
		说明:
		示例 SUB DX,CX
		SUB [BX+25],AX
		SUB DI,ALFA[SI]
		SUB CL,20
		SUB DATA1[DI][BX],20A5H

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.
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.
XCHG	数据交换指令 XCHG 格式: XCHG OPRD1,OPRD2 其中的 OPRD1 为目的操作数,OPRD2 为源操作数 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.	说明: 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. 2. 段寄存器内容不能用 XCHG 指令来交换. 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 PUSH DATA2 POP DATA1 POP DATA2 4. 本指令不影响状态标志位.
XLAT	查表指令 XLAT 格式: XLAT TABLE 其中 TABLE	说明: 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 (AL)<--((BX)+(AL)).

	为一待查表格的首地址. 功能: 把待查表格的一个字节内容送到 AL 累加器中.	2. 本指令不影响状态标位,表格长度不超过 256 字节.
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,对标志位的影响与与 AND 指令相同. 2. 相异为真,相同为假.