# 高档微机开发指南

鲍明忠

H

中国科学院希望高级电
一九九〇年三

# 高档微机开发指南

鲍明忠

中国科学院希望高级电脑扌

一九九〇年三月

■北京新闻出版局

准印证号：891445

■订购单位：北京8721信箱资料部

■邮码：100080

■电话：2562329

■乘车：320、332、302路车至
海淀黄庄下车

■办公地点：希望公司大楼101房间

# 前　　言

本书是编者根据自己的开发实践，汇集许多软件、硬件资料的基础上编写的。其主主要目的是为了满足那些从事微机开发、应用工作的人之需要。

本书的编写方法是：力求把一般性的原理介绍与实际的应用程序例子相结合。一般性的原理介绍是为了便对于应用程序的理解，而实际应用实例则是为了对一般性原理的掌握。书中的许多例子都是实际应用程序的一部分。

由于汇编语言是最接近机器，也是与硬件联系最密切。为了便于读者从软件设计的角度掌握硬件，书中的大多数例子是以汇编语言的形式给出。由于书中的一部分对硬件编程的例子就是DOS中使用的，所以，读完本书，不但可以掌握有用的微机方面的资料，而且你还会更加了解DOS对硬件设备的管理方法。

虽然本书中的大多数例子是以IBM PC／XT，／AT系列机为基础的，但也有相当一部分与硬件不是直接相关的程序可用于其它型号的机器。

本书在编写、出版过程中得到希望公司的帮助，在此表示衷心地感谢。同时也向在编写本书过程中给予支持和关心的童家仙先生，宋锎女士表示感谢。

鲍明忠　　谭真

一九八九年十一月　于北京

# 目　　录


**第一章　80286 CPU结构及指令系统**

1.1　CPU的结构……（2）

1.2　指令系统……（5）

**第二章　总线**

2.1　Multibus总线……（18）

2.2　AT计算机的总线……（21）

2.3　微通道……（31）

**第三章　80286的虚地址保护模式**

3.1　保护的虚地址方式……（40）

3.2　DOS中虚地址模式的进入过程……（48）

3.3　虚地址模式下的程序设计问题……（52）

3.4　OS／2的虚拟存贮管理……（61）

**第四章　AT计算机的支持芯片**

4.1　82289总线仲裁器……（66）

4.2　总线控制器82288……（68）

4.3　时钟发生器82284……（72）

4.4　8259A中断控制器……（73）

4.5　DMA控制器8237A-5……（89）

4.6　RT／CMOS芯片……（105）

**第五章　键盘控制器**

5.1　键盘控制……（113）

5.2　组合键……（121）

5.3　双态键……（124）

5.4　编制自己的键盘中断处理程序……（126）

5.5　DOS键盘的扩展功能……（136）

5.6　一个键盘扩展程序……

**第六章　视频显示**

6.1　EGA显示的一般控制……（155）

6.2　字符显示……（162）

6.3　增强型图形适配器（EGA）……（172）

6.4　MCGA与VGA显示……（18

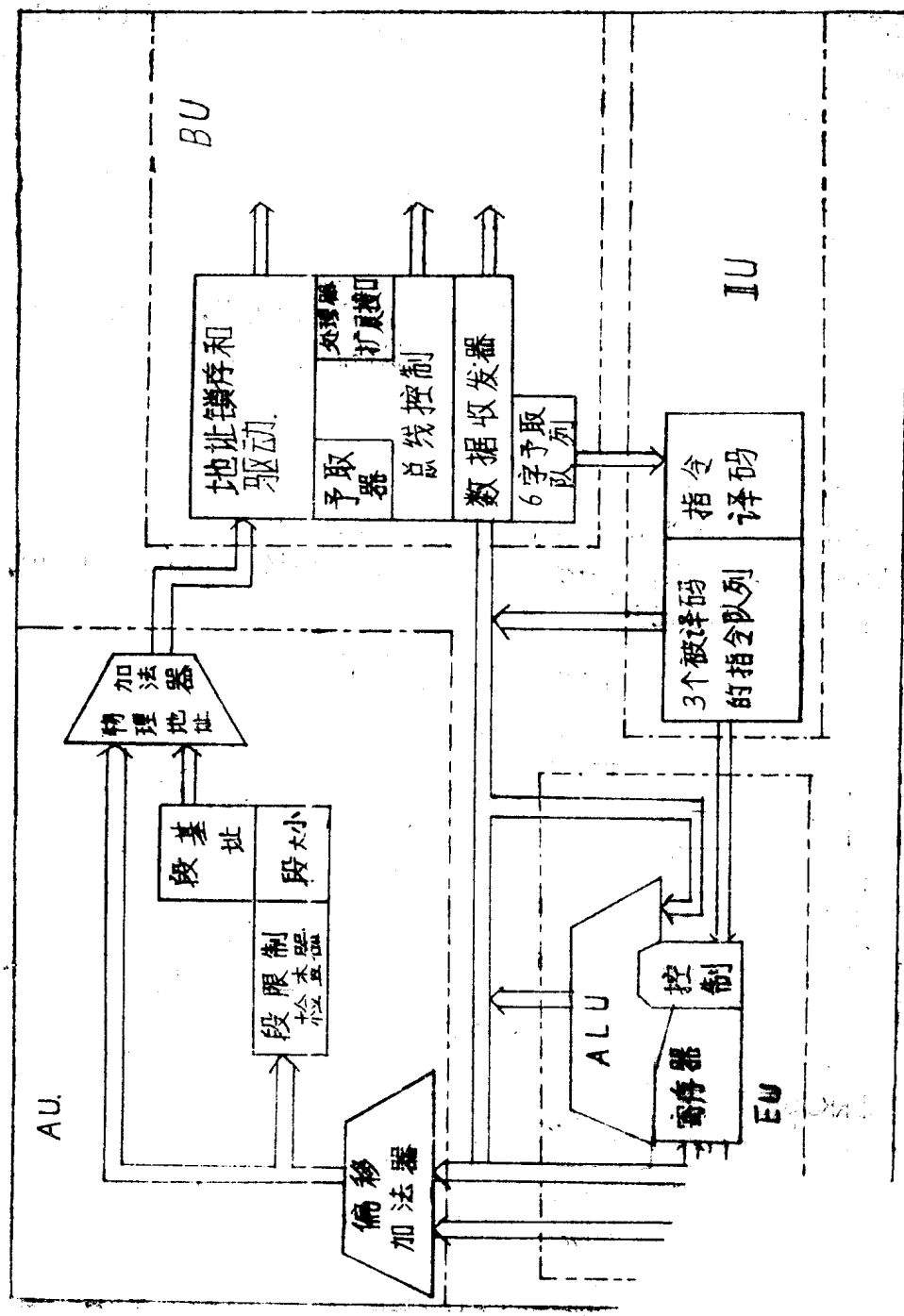
6.5　CGE显示……

6.6　窗口设计……

6.7 屏幕编辑……（215）
**第七章 磁盘控制**
7.1 软盘控制 ……（227）
7.2 硬盘控制 ……（242）
7.3 磁盘读／写、根记录读／写……（273）
7.4 DOS目录的内容 ……（277）
**第八章 磁盘文件**
8.1 DOS下的磁盘文件操作 ……（288）
8.2 高级磁盘文件操作 ……（296）
**第九章 软件开发工具的研究**
9.1 宏汇编 ……（316）
9.2 C语言的研究……（341）
**第十章 协处理器80287**
10.1 80287的结构……（356）
10.2 80287的指令集……（359）
10.3 协处理器的例外处理 ……（365）
10.4 80287的建立……（369）
**第十一章 80386简介及其支持芯片**
11.1 80386的结构……（373）
11.2 82384时钟发生器……（376）
11.3 高性能的32位DMA控制器 82380……（376）
11.4 32位高速缓冲控制器82385……（380）
11.5 操作系统结构……（382）

# 第一章　80286 CPU结构及指令系统

80286 cpu在功能上比8086／8088 CPU更强大。它有两种处理机模式，即实地址模式和虚地址保护模式。实地址模式就是一个增强的8086／8088，此时它最大寻址空间为




CPU结构简图

图1.1　80286 CPU的结构

1 兆；虚地址保护模式是一个新的处理机模式，此时，80286使用24位地址线，寻址空间为16兆。而且提供了存贮器的保护功能。在虚地址模式下，cpu内部操作方式将不同于实地址模式。在虚地址模式下，cpu提供的存贮器管理能力使微机系统的功能进入了多任务、多进程时代。而这在8086／8088系统上实现是很困难的。

本章将从cpu的结构、指令两个方面介绍80286 cpu所具有的强大的功能。

## 1.1 CPU的结构

BU (bus unit)

完成所有存贮器及 I/O 的读、写，当BU不用总线进行其它操作时，它予取指令字节，并把它们放入6字节予取队列里，当执行JMP或CALL指令时，BU将冲掉该队列，且用目标地址填入它，BU也控制与处理器扩展设备的数字传输，比如80287。

IU (instruction unit)

最多完成译三条予取指令，并把它们放入队列，EU从那里取出它们去执行。这是使用管道加速处理机的操作。

EU (excute unit)

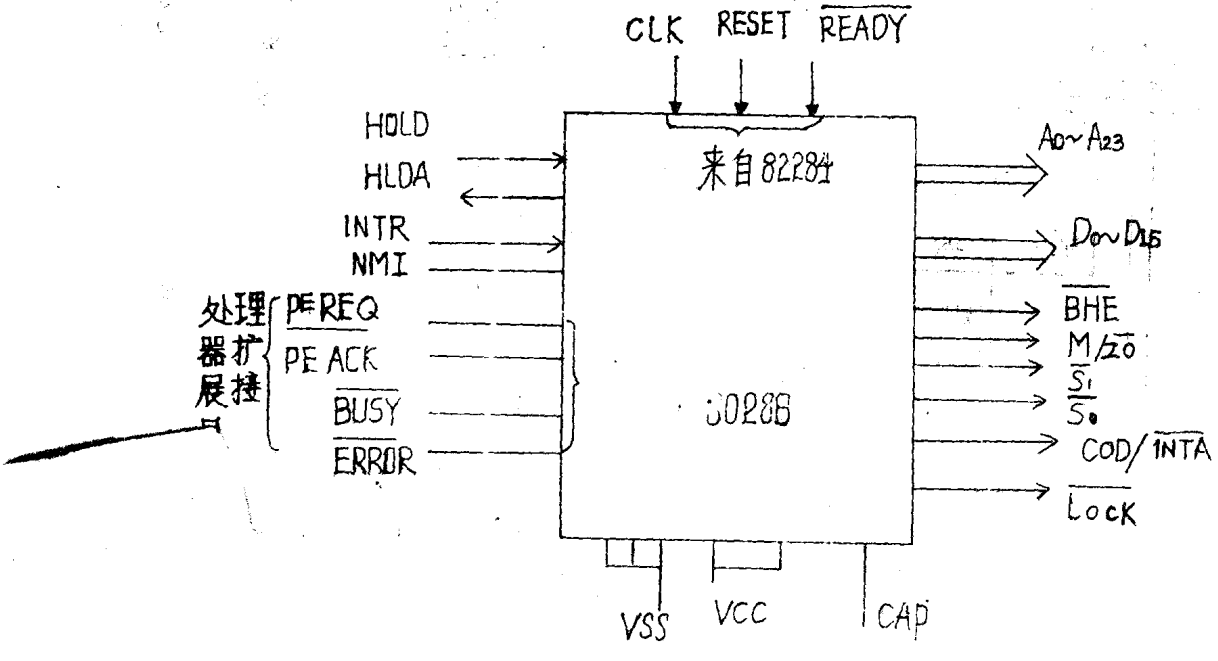
顺序执行指令，它含有一组与8086一致的索引，指针和通用寄存器，除了象8086的标志寄存器，80286还有一个16位机器状态寄存器，EU根据需要指示BU单元存取存贮器或I/O。

AU (address unit)

计算将由BU送到存贮器或I/O的地址。

当在实地址模式下时，完全类似于8086，这时寻址空间是1兆。

当在虚地址模式下时，AU类似于一个MMU，使用所有的24根地址线存取16兆物理存贮空间。此时提供虚拟和保护能力。




80286 CPU管脚图

管脚说明：

CLK　系统时钟，它为iAp×286系统提供基本的定时，在80286内部被二分频产生处理器时钟，内部分频电路通过Reset输入与外部时钟发生器同步。

$D_0$～$D_{15}$数据总线，在存贮器、I/O和中断应答读周期时，输入数据；在存贮器、I/O写周期时，输出数据。

$A_0$～$A_{23}$地址总线，输出物理存贮器和I／O口地址。$A_0$为低时，数据被传送到$D_0$～$D_7$，$A_{16}$～$A_{23}$在I／O传送时为低。高有效，总线保持应答期间（即HLDA为高）时，三态浮空。

$\overline{BHE}$　总线高使能。它指出数据或传输在数据总线的高字节，$D_8$～$D_{15}$。若8位设备被连到数据总线的高字节上，使用$\overline{BHE}$作芯片选择功能的条件为：

| $\overline{BHE}$ | $A_0$ | |
|---|---|---|
| 0 | 0 | 字传送 |
| 0 | 1 | 高半字传送（$D_8$～$D_{15}$） |
| 1 | 0 | 低半字传送（$D_0$～$D_7$） |
| 1 | 1 | 不会发生 |

$\overline{S_1}$，$\overline{S_0}$　总线周期状态。指出一个总线周期开始，与M／$\overline{IO}$，COD／$\overline{INTA}$一起定义总线周期类。不论何时，只要有1个或2个为低时，则总线处于Ts状态。

| COD／INTA | M／$\overline{IO}$ | $\overline{S_1}$ | $\overline{S_0}$ | 被初始的总线周期 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 中断应答 |
| 0 | 0 | 0 | 1 | ／ |
| 0 | 0 | 1 | 0 | ／ |
| 0 | 0 | 1 | 1 | 不是一个状态周期 |
| 0 | 1 | 0 | 0 | 若A＝1，则停机，否则shutdown |
| 0 | 1 | 0 | 1 | 存贮器读 |
| 0 | 1 | 1 | 0 | 存贮器写 |
| 0 | 1 | 1 | 1 | 不是一个状态周期 |
| 1 | 0 | 0 | 0 | ／ |
| 1 | 0 | 0 | 1 | I／O读 |
| 1 | 0 | 1 | 0 | I／O写 |
| 1 | 0 | 1 | 1 | 不是一个状态周期 |
| 1 | 1 | 0 | 0 | ／ |
| 1 | 1 | 0 | 1 | 存贮器指令读 |
| 1 | 1 | 1 | 0 | ／ |
| 1 | 1 | 1 | 1 | 不是一个 |

M／$\overline{IO}$　区别存贮器存取和I／O存取。

COD/$\overline{INTA}$(code/interrupt acknololedge)区别

$\overline{Lock}$　指出其它系统总线管理者不能在当前和

控制权。Lock可由Lock指令前缀激活，或在存贮器XGHG指令、中断应答或描述器表存取时，由80286的硬件自动激活。

READY （BUS Ready）终止一个总线周期。一个总线周期可无限制地扩展，直到由Ready变低为至。它是一个同步输入，故要求一定的建立时间和保持时间。

HOLD、HLDA、HOLD输入允许另一个总线管理者请求局部总线的控制权，当控制被允许时，80286将浮空它的总线驱动器、激活HLDA，然后，进入总线HLDA条件，请求者将保持控制权，直至HOLD变为无效、导致80286中止，HLDA再次获得总线控制权。

INTR 请求80286挂起当前程序的执行，服务外部请求。若标志字里的中断使能位被清除，则中断被屏蔽，当80286响应一个中断请求时，进行两个中断应答总线周期，读8位中断向量，它指明中断源，INTR在每个处理周期的开始被采样。

Vss 系统地 OV

Vcc 系统电源 +5V

RESET 请80286内部逻辑

80286的存贮器按奇堆、偶堆构成，非常类似于8086．$A_0=0$时，偶堆被选中；$\overline{BHE}$为低时（$\overline{BHE}=\phi$），奇堆被选中。为存取一个成行的字，$A_0$和$\overline{BHE}$同时为低。

从控制观点来看，80286类似8086在最大模式下的操作。状态信号$\overline{S_0}$，$\overline{S_1}$和M／IO由一个外部总线控制器82288译码产生控制总线、读、写和中断应答信号。

另外一些类似的脚是HOLD（为DMA操作请求使用总线）和HOLD应答(HLDA)（用于通知DMA控制器总线可以使用）。8259A优先级中断控制器被连到INTR中断输入，82288的中断应答输出$\overline{INTA}$，告诉8259A在数据线上发送希望中断型给处理机，NMI输入、$\overline{READY}$输入和$\overline{LOCK}$输出类似于8086。

CAP 为了以最大速度操作，在80286中，MOS设备基底带有负电压的偏移，这个负电荷是由80286上的一个偏压发生器产生的。连到CAP的外电容滤掉该偏压，注意电容极性。

COD／INTA是一种类型的状态信号输出，它与M／IO一起更早地产生控制总线信号。它对中断应答、存贮器读、写信号将断言为低，对I／O读、I／O写或存贮器指令读周期，它将是高。

**与80287接口的四个信号**

PEREQ (Processor Extension request) 它由协处理器设置通知80286为它从存贮器读或往存贮器里写数据。当80286完成一个这样的传送后，它置$\overline{PEACK}$ (Processor Extension acknowLedge) 给协处理器，让它知道该数据传送已经开始。这样数据传送由80286完成，所以，协处理器利用了80286中的mmu的虚存和保护能力。

$\overline{Busy}$ 与8086的功能一样，当80286执行一个WAIT指令时，它将停留在一个等待循环中

器的$\overline{Busy}$变为高。若协处理器在处理过程中发现某个错误

$\overline{ER}$，这将引起80286自动地进行一个类型16H的中断调用

错误条件作相应的反应。

## 1.2 指令系统

80286的指令是8088系统的超集，它包括所有的8088指令，并有一些指令的功能得到加强。如移位指令SHR AL，3。所有的8088指令都可在80286上运行。

此外还增加了一些新的指令，这些指令大多数是用于80286的虚地址保护模式。

80286的协处理器80287在功能上比8087更强，相应的80287的指令也有所增强，80287的指令将在另一章中介绍。

〈一〉算术运算指令

算术运算指令的操作数可以是字节、也可以是字。

1、加法指令

ADD、ADC、INC

```
ADD    AL，2     ；8位加法
ADD    AX，2     ；16位加法
```

若加长度超过16位的数，则可以同时使用ADD和带进位加法。

```
ADD  BX，L-WORD
ADC  AX，H-WORD
```

指针的加法就是这样实现的。AX：BX是指针的当前值，H-WORD：L-WORD则是移动指针的偏移量。

INC 加1运算的操作数是无符号的二进制数。它不会影响CF标志，而只影响AF、OF、PF、SF、EF

```
INC    BL    ；8位加1运算
INC    BX    ；16位加1运算
```

2、减法指令

SUB，SBB，DEC

减法是从目的操作数减去源操作数，并依据结果而影响标志位。

```
SUB    AL，DL    ；8位减法
SUB    AX，CX    ；16位减法
```

若减16位以上的数，则可用SBB、SUB组合进行。

例如进行指针减法

```
SUB    BX，OFFSET-L
SBB    AX，OFFSET-H
```

SBB带进位减法可以减去低字减的借位。

DEC减1运算与INC加1运算类似，只是DEC是把操作数减1。

3、乘法指令

MUL、IMUL

乘法有8位乘法、16位乘法。对8位乘法其结果放在AX中；对16位乘法，其结果的高16位在DX中，低16位在AX中。若是IMUL，其结果还可放入指定的寄存器中。

MUL是无符号乘法；IMUL是有符号乘法。

```
MUL  DL    ；8位无符号乘法
MUL  BX    ；16位无符号乘法
```

```
IMUL  BL      ; 8位有符号乘法
IMUL  BX      ; 16位有符号乘法
IMUL  BX, SI, 5 ; BX←SI×5
```

4、除法指令

```
DIV   IDIV
```

除法与乘法一样也有8位除、16除、有符号除，无符号除。

对16位除法，DX：AX是被除数，DX是高位字，AX是低位字。其除的结果商放在AX中，余数放在DX中。

对8位除法，AX中是被除数，AH是高字节，AL是低字节。商放在AL中，余数放在AH中。

使用除法可实现取整、取模。

```
DIV   BL
DIV   BX
IDIV  BL
IDIV  BX
```

〈二〉逻辑指令

6、逻辑指令包括布尔运算指令、旋转和移位指令等

1、布尔运算

AND、NOT、OR、XOR、NEG

AND、NOT、OR、XOR是按位执行逻辑运算；NEG对操作数的逻辑运算。它们都具有字节、字的操作数类型

```
AND   AL, Φ7
AND   AX, Φ7FH
NOT   AL
NOT   AX
OR    AL, Φ8ΦH
OR    AX, Φ8ΦΦΦH
XOR   AL, Φ1
XOR   AX, 1ΦH
NEG   AL
NEG   AX
```

XOR运算可实现特定位的取反运算。例

```
XOR   AL, 1
```

即将AL的最低取反。

OR 通常用于设置特定的位，如

```
      AL, 1
```

即

```
      1
```

，如

```
AND    AL, 0FEH
```

即将AL的最低位清0。

2、移位指令

SAL、SHL、SHR、SAR

SAL、SAR是算术移位指令，SAL在右边位补0，SAR（右移）则保留最高位，以保留操作数的正负号。

SHL、SHR是逻辑移位指令，不论左移，右移，空出的位都填0。

移位的次数有三种表示方法：

1、隐含在指令中，但只能移1位

2、使用一个立即数表示，这是286的功能。

3、把移位的次数放在CL中。

```
SAL    BL, 1
SAL    BL, 2
SAL    BL, CL
```

也可以是字操作数，它们常被用来作乘以或除以$2^n$运算

```
SHR    AX, 1      ; AX除以2
SHR    AX, 3      ; 指定移位次数
SHR    AX, CL
```

3、循环移位

ROL、ROR、RCL、RCR

循环移位分成两组，即不带进位循环移位，带进位的循环移位。

移位的次数同移位指令一样的方法被确定，也是三种方法。

循环移位的源操作数可以字节、也可以是字。与移位指令的区别是在移位过程中，不会有任何位被丢失。

通常使用这些指令及JC、JNC条件转换指令作条件转移。

```
        ⋮
        RCR    AX, 1
        JC     BIT1_S T
        ⋮
```

把AL中的低四位扩充成8位

```
        MOV    BL, AL
        MOV    CL, 4
Next_bit: ROR  AL, 1
        RCR    DL, 1
        ROR    BL, 1
        RCR    DL, 1
        Loop   NEXT_bit
```

〈三〉、类型转换、空操作

CBW、CWD、NOP

CBW将AL中带符号数扩充到AX中的16位数。CWD将AX中的16位有符号数扩充到DX：AX中的32位数。这两条指令通常是为了得到一个16位或32位的被除数。它们不影响任何标志位。

NOP 是空操作指令，通常是用它来延迟指令的执行。

〈四〉测试指令、比较指令、条件转移指令

测试、比较指令的共同点是它们不改变操作数的值，只是设置标志位。

测试指令是测试操作数中特定的位，或位集合。比较指令则测试整个操作数的值。

例如： TEST AL, 81H

则测试AL的第1位、第八位是否至少有一个不为0

而 AND AL, 81H

CMP AL, 81H

无符号数比较

| 助记符 | 条件 | 意义 |
|---|---|---|
| JA／JNBE | (CF或ZF) = 0 | 大于 |
| JAE／JNB | CF = 0 | 不小于 |
| JB／JNAE | CF = 1 | 小于 |
| JBE／JNA | (CF或ZF) = 1 | 不大于 |
| JC | CF = 1 | 进位 |
| JE／JE | EF = 1 | 等于 |
| JNC | CF = ① | 无进位 |
| JNE／JNE | EF = ① | 不相等 |
| JNP／JPO | PF = ① | 不同位 |
| JP／JPE | PF = 1 | 同位 |

有符号数

| 助记符 | 测试条件 | 意义 |
|---|---|---|
| JG／JNLE | (SF XOROF) orEF = ① | 大于 |
| JGE／JNL | SF XOR OF = ① | 不小于 |
| JL／JNGE | SF XOR OF = ① | 小于 |
| JLE／JNG | (SF ×OR OF)ORE F = 1 | 不大于 |
| JNO | OF = ① | 无溢出 |
| | SF = ① | 非负 |
| | OF = 1 | 溢出 |
| | SF = 1 | 负 |

则是测试AL的第1位、第八位是否都为1。

在程序设计中，TEST通常用来测试标识位，以确定是否进入指定状态，或指定条件是否发生，而它们是由开关位表示。

条件转移指令 INTO

INTO 若该指令执行时，OF = 1，则产生中断4。

〈五〉 无条件转移指令、软中断。

无条件转移指令包括JMP、CALL、RET、INT、IRET、JMP、CALL指令的操作数可以是一个16位地址，也可以是段地址加16位偏移地址。若控制是在同一个段内转移，则地址可以是16位偏移地址，也可以是段地址加偏移地址；若是段间转移，则必须是段地址加偏移地址。

```
JMP   INNER-ENTRy
JMP   SI
JMP   DWORD PTR   TABLE [BX]
```

JMP指令不保存返回地址，CALL指令保存返回地址。

RET

从子程序返回

INT n

根据n的值，执行指定的中断服务程序，它是执行操作系统程序的一种方法。

IRET

从中断服务程序返回被中断的程序，恢复标志寄存器。

RET n

该指令和RET指令相同，只是它还将SP的值加n，目的是取消调用程序放入堆栈的参数。n必须是偶数。

〈六〉 循环指令

循环指令用CX存放循环执行的次数。LOOP LOOPE LOOPZ LOOPNE每次执行时，先将CX减1，并检查CX是否为0，若为0，则结束循环。

LOOP指令仅测CX是否为0，为0则执行LOOP的下一条指令。

LOOPE与LOOPZ是相同的指令，它们既测试CX也测试ZF标志。

若CX≠0且ZF = 1，则程序转移到LOOPE／Z后指定的标号位置。

若CX = 0或ZF = 0则执行LOOPE／Z的下一条指令。

LOOPNE与LOOPE／Z一样，既检测CX，也检测ZF标。但不同的是，若CX≠0且ZF = 0时，执行循环体；而CX = 0或ZF = 1时，则执行LOOPNE的下一条指令。

JCXZ则是仅检查CX是否为0，它不改变CX的值。它与LOOP指令的差别是LOOP先执行CX减1操作，然后再测试CX是否为0。所以在程序循环中，通常这两条指令结合起来使用。

```
        ⋮
        JCXZ    OUT_LOOP
IN_LOOP: ⋮
```

```
        LOOP    IN_LOOP
OUT_LOOP: ⋮
```

保证CX = 0时，也保证结果是正确的，而避免CX = 0时，LOOP使CX变成0FFFFH的情况。

〈七〉 串处理指令

字串指令包括传送，比较、扫描字节、字字符指令，若加上前缀，还可有重复处理能力。

1、传送指令

MOVS/SB/SW其中MOVSB、MOVSW的操作数是隐含的。DS：SI为源地址，ES：DI为目标地址。每完成一次传送SI，DI按修正量进行修改。若DF = 0，则SI，DI各自加修正量；若DF = 1，则SI、DI各自减修正量。对MOVSB修正量是1；对MOVSW修正量是2。MOVS是操作数也是DI、SI；但它到底是传送字节，还是字则由操作数的类型指出。

```
LEA     SI, Source
LEA     DI, ES : destination
MOVS    destination, Source
```

若这些无前缀REP，则它们只执行一次。若加前缀REP，则传送重复执行，每执行一次CX减1，直到CX为0时结束。

```
        CLD
        MOV     S , Source
        MOV     DI, ES : destination
        MOV     CX, 50
    REP MOVSW
```

在视屏显示时，使用该指令可以快速地在屏幕显示信息。

2、比较指令

CMPS/SB/SW

操作数及其修改的方法与MOVS/SB/SW一致。但它仅比较源操作数（DS：SI）与目的操作数（ES：DI）的内容，并根据比较结果设置标志位。相当于用DS：SI所指单元内容减去ES：DI所指单元内容，并置标志位。但并不改操作数所指单元的内容，即DS：SI，ES：DI所指单元内容。

若无前缀，则指令仅执行一次。若加前缀则重复执行。前缀是REPE/Z或REPNE/NZ，与MOVS不同的是，比较是否中止受两个条件限制，即CX的内容和标志位ZF。

```
REPE/Z    CMPS/SB/SW
```

重复执行比较，直至CX = 0或ZF = 0，这可用来比较两个串是否相同。

```
        ⋮
        MOV     CX, 10
        MOV     SI, str1
        MOV     DI, str2
        REPE    CMPSW
```

```
        JNE     no_equal
        ⋮
```

则是比较两个长度为10的字数组是否相等，实际是比较长度20的二个字符串。

```
REPNE/NZ  CMPS/SB/SW
```

则可寻找两个串中第一个匹配字符的位置。

3、扫描字串指令

SCAS/SB/SW 比较AL或AX的内容与ES：DI所指单元的内容，并根据结果设置标志位。对ES：DI的修改与MOVS相同。实际上是用AL或AX中的内容减去(ES：DI)，并根据结果设置标志位。

该指令常用于寻找字符串中特定字符，或跳过特定字符，如滤除空格。

该指令也可有前缀REPE/Z和REPNE/NZ

```
        CLD
        MOV     AL,     ' '
        MOV     DI, str1
        REPE    SCASB
```

则滤除字符串str1的前导空格。

4、寄存器装入、写入指令

LODS/SB/SW、STOS/SB/SW

LODS/SB/SW将DS：SI所指单元的内容按字节或字取到AL，AX中，并相应修改DS：SI指针，修改的方法与MOVS一致。

STOS/SB/SW 将AL，AX的内容存入ES：DI所指单元，并相应修改DI的值，方法与MOVS一致。该指令常用于初始化内存单元到一指定的值。

```
        XOR     AX, AX
        MOV     50
        MOV     DI, destination
        REP     STOSW
```

则将目的单元清0。

5、查表指令

XLAT 操作数

该指令是为便于查表而设计的。由DS：BX作为基地址，AL作为无符号的下标，从DS：BX所指示的表中，查出一个字节、并将它放入AL中。

```
        LEA     BX, TABLE
        MOV     AL, INDEX
        XLAT    TABLE
```

该指令中的操作数是一个虚拟的操作数。有无都只产一个一字节操作码，它通常是为了增加程序的可读性，而无其它的意义。

〈八〉地址处理指令

LEA 取操作数的偏移地址