第 3 章　16 位/32 位微处理器指令系统

所谓指令，就是要求计算机执行各种特定操作的命令。微机能够识别和执行的全部指令集合称为该微机的指令系统。不同的微处理器所对应的指令系统也不相同。

按功能分，指令系统的指令一般分为数据传送类指令、算术运算类指令、逻辑运算与移位类指令、字符串处理类指令、控制转移类指令、处理器控制类指令等。本章首先给出指令的一般格式，然后介绍指令中操作数的寻址方式，最后详细分析各类指令的功能及其使用方法。


视频讲解

3.1　指令的基本格式

3.1.1　指令的构成

指令包括两部分内容：一部分是表示操作性质或类型的编码——操作码；另一部分是操作对象——操作数。具体构成与微处理器有关。

现以 8086/8088 机器指令为例，说明它的指令构成规则。该指令系统的指令由 1～6 字节组成，如图 3.1 所示。一般来说，第一字节（即图 3.1 中的字节 1，下同）表示操作码，第二字节表示寻址方式，第三、四字节表示操作数在内存的位移量或者是立即数（在指令中没有位移量时），第五、六字节表示立即数。

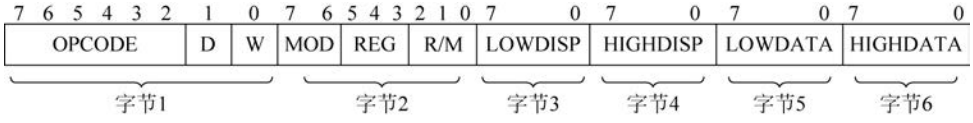

图 3.1　8086/8088 指令构成

指令第一字节的 7～2 位为操作码，由该字段规定指令的操作类型。第一字节的 1 位为 D 位，由其指定操作数的传输方向。D＝0，表示 REG 字段（第二字节的 5～3 位）给出的寄存器为源操作数寄存器，操作数应从这个寄存器中取得。D＝1，表示 REG 字段给出的寄存器为目的操作数寄存器，运算结果应存放这个寄存器中。第一个字节的 0 位为 W 位。W 位指出参加运算的操作数是字（16 位）还是字节（8 位）。W＝0，表示参加运算的数是字节操作数；W＝1，表示参加运算的数是字操作数。

指令的第二字节给出指令中使用的两个操作数存放在什么地方和存储器操作数的有效地址如何求得。第二字节的 7、6 位为 MOD（方式）字段。由 MOD 给出指令中使用的两个操作数都是寄存器操作数，还是一个是寄存器操作数，而另一个是存储器操作数，如表 3.1 所示。

表 3.1 MOD 字段编码

MOD	说　明
00	存储器方式,没有位移量
01	存储器方式,有 8 位位移量
10	存储器方式,有 16 位位移量
11	寄存器方式(无位移量)

指令第二字节的 5～3 位为 REG 字段。REG 字段给出指令中使用的一个操作数所在寄存器的编码。表 3.2 为寄存器编码。

表 3.2 REG 字段编码

REG	W=0	W=1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

第二字节的 2～0 位称为 R/M(存储器/寄存器)字段。R/M 字段和 MOD 字段共同确定寻址方式。当 MOD=11 时为寄存器方式,由 R/M 字段给出第二个操作数所在的寄存器编码。当 MOD≠11 时为存储器寻址方式。R/M 字段和 MOD 字段用来指出应如何计算存储器操作数的有效地址,如表 3.3 所示。

表 3.3 R/M 和 MOD 字段编码及有效地址计算方法

存储器寻址方式 MOD≠11				寄存器寻址方式 MOD=11		
R/M	MOD=00	MOD=01	MOD=10	R/M	W=0	W=1
000	(BX)+(SI)	(BX)+(SI)+D8	(BX)+(SI)+D16	000	AL	AX
001	(BX)+(DI)	(BX)+(DI)+D8	(BX)+(DI)+D16	001	CL	CX
010	(BP)+(SI)	(BP)+(SI)+D8	(BP)+(SI)+D16	010	DL	DX
011	(BP)+(DI)	(BP)+(DI)+D8	(BP)+(DI)+D16	011	BL	BX
100	(SI)	(SI)+D8	(SI)+D16	100	AH	SP
101	(DI)	(DI)+D8	(DI)+D16	101	CH	BP
110	16 位直接寻址	(BP)+D8	(BP)+D16	110	DH	SI
111	(BX)	(BX)+D8	(BX)+D16	111	BH	DI

指令中的第三～六字节根据不同的指令有不同的安排。一般由它们给出存储器操作数地址的位移量或立即操作数的值。指令中给出的位移量,可以是 8 位的也可以是 16 位的。一条指令中有一字节还是两字节的位移量由指令码中的 MOD 字段定义。8086 系统规定,如果位移量为两字节,那么位移量的高位字节存放在高地址,低位字节存放在低地址。如果指令中有立即操作数,那么立即操作数位于位移量的后面。也就是如果三、四字节有位移量(LOWDISP、HIGHDISP),立即数就位于五、六字节(LOWDATA、HIGHDATA)。若指令中无位移量,立即数位于指令码的三、四字节。总之,指令中缺少的项要由指令后面存在的

项向前顶替，以减少指令的长度。

上述介绍的是 8086/8088 指令构成的一般情况，在个别指令中也有例外的情况。

3.1.2 8086/8088 的指令格式

指令格式是指令在源程序中的书写格式，其基本格式为：

```
[标号:]  操作码助记符  目的操作数,源操作数  [;注释]
[标号:]  操作码助记符  目的操作数            [;注释]
```

第一条格式为双操作数指令格式，第二条为单操作数指令格式；方括号部分为可选项，可有可无。

其中，标号为该条指令所在内存单元的符号地址，后面要跟冒号。标号一般由字母开头，后跟字母、数字或特殊字符。注意，不允许使用保留字，例如 MOV、SEGMENT、END 等。

操作码助记符指示 CPU 执行什么样的操作。

操作数分目的操作数和源操作数两种，目的操作数是指令结果存放的位置，源操作数是指令操作的对象。目的操作数应放在源操作数前，并以逗号隔开。

注释用来说明本条指令或一段程序的功能，使程序可读性强。注释由分号(;)开始，汇编程序对其不进行处理。

3.2 8086/8088 的寻址方式

一条汇编语言指令有两个问题需要关注：一个是该条指令将进行什么操作；另一个是操作的对象和操作后结果的存放位置。操作数的寻址方式就是指寻找操作数位置的方式。

1. 立即寻址

在 8086 指令系统中，有一部分指令所用的 8 位或 16 位操作数直接在指令中给出，故称为立即寻址。以这种寻址方式表示的操作数也简称立即数。

例如：

```
MOV AL, 80H          ;将十六进制数 80H 送入 AL
MOV AX, 306AH        ;将十六进制数 306AH 送入 AX,其中 AH 中为 30H
                     ;AL 中为 6AH
```

立即数可以为 8 位，也可以为 16 位。但规定立即数只能作为源操作数。

立即寻址方式常用于给寄存器赋值，操作数直接在指令中取得，不需要使用另外的总线周期，故执行时间短、速度快。

2. 寄存器寻址

如果操作数在 CPU 的内部寄存器中，那么可在指令中指出该寄存器名，这种寻址方式称为寄存器寻址。

对于 8 位操作数来说，寄存器可以是 AH、AL、BH、BL、CH、CL、DH、DL。

对于 16 位操作数来说，寄存器可以是 AX、BX、CX、DX、SI、DI、SP、BP 等。

例如：

```
MOV  AL, BL          ;将 BL 的内容传送到 AL 中
```

```
MOV  BX, AX          ;将 AX 的内容传送到 BX 中
```

采用寄存器寻址的指令在执行时,操作数就在 CPU 中,不需要访问存储器来取得操作数,执行速度快。另外寄存器名比内存地址短,指令所占内存空间少。

3. 直接寻址

通常把操作数的偏移地址称为有效地址(Effective Address,EA),EA 可通过不同的寻址方式得到。

在直接寻址方式中,指令中直接给出操作数的有效地址。注意,如果用数字形式提供有效地址,可在指令中直接给出 16 位有效地址并加方括号([]),但必须加上段前缀,以便和立即数相区分。

例如:

```
MOV AX, DS:[1000H]   ;若 (DS) = 2000H,则将数据段 21000H
                     ;21001H 两个单元的内容送到 AX 中
```

如果有效地址是以符号地址形式提供的,则可不加方括号。

例如:

```
MOV AX,BUFA
```

BUFA 为符号地址,这时可不加跨段前缀,默认为 DS 数据段。如果 BUFA 变量在附加段中,就必须书写如下:

```
MOV AX,ES: BUFA
```

需要说明的是,有些宏汇编程序规定,指令中直接给出的是 16 位常量的有效地址时也默认操作数在 DS 数据段,这时可以不加段前缀 DS。

例如,MOV AX,DS:[1000H]可写为 MOV AX,[1000H]。

4. 寄存器间接寻址

操作数在存储器中,操作数有效地址存放在指定寄存器中,被用来存放操作数的有效地址的寄存器称为间址寄存器。间址寄存器必须是 BX、BP、SI 和 DI 之一。

书写时,该寄存器必须加方括号。如果没有加方括号,则为寄存器寻址。

当指令中使用 BX、SI 和 DI 作间址寄存器时,默认操作数在数据段中;如果使用 BP 做间址寄存器,则默认操作数在堆栈段中,否则应该加段前缀。

例如:

```
MOV  AX, [BX]        ;若(DS) = 2000H,(BX) = 1000H,则将数据段 21000H
                     ;21001H 两个单元的内容送到 AX 中
MOV  CX, [BP]        ;若(SS) = 4000H,(BP) = 1000H,则将堆栈段 41000H
                     ;41001H 两个单元的内容送到 CX 中
MOV  AX, ES:[SI]     ;若(ES) = 3000H,(SI) = 1000H,则将附加段 31000H
                     ;31001H 两个单元的内容送到 AX 中
```

5. 基址寻址和变址寻址

操作数在存储器中,操作数的有效地址是基址寄存器或变址寄存器的内容与指令中指定的 8 位或 16 位位移量之和。可以选用 BX、BP 作为基址寄存器,选用 SI、DI 作为变址寄存器。指令中使用 BX、SI 和 DI 时,默认操作数在数据段中;指令中使用 BP 寄存器时,则默认操作数在堆栈段中。可以用段前缀来改变默认段基址。

例 3.1 MOV AX,[SI+3000H]的执行结果。

设(DS)=4000H,(SI)=2000H,内存单元(45000H)=34H,(45001H)=12H。

首先求操作数的物理地址:

物理地址=(DS)×16+(SI)+3000H=40000H+2000H+3000H=45000H

然后根据该物理地址从内存得到操作数,也即内存单元 45000H、45001H 的内容。将该内容送入 AX,故(AX)=1234H。

6. 基址变址寻址

操作数在存储器中,操作数的有效地址是一个基址寄存器和一个变址寄存器内容以及 8 位或 16 位位移量三者之和,两个寄存器均由指令指定。可以选用 BX、BP 作为基址寄存器;选用 SI、DI 作为变址寄存器。如果选用 BX 做变址寄存器,则默认操作数在数据段中,如果选用 BP 做变址寄存器,则默认操作数在堆栈段中。也可以用段前缀来改变默认段基址。

例如:

```
MOV   AX, 8[BX + SI]          ;默认操作数在数据段中
MOV   BX, -6[BP + DI]         ;默认操作数在堆栈段中
MOV   BX, ES:[BP + DI]        ;操作数在附加段中
```

基址变址寻址方式适用于表格或数组的存取,可以将表格或数组的首地址放入基址寄存器中,将表格或数组中的数据的相对位移放在变址寄存器中,灵活实现表格或数组中数据的存取。

7. 固定寻址

固定寻址又称隐含寻址,指令码中不包含指明操作数地址的部分,而其操作码本身隐含地指明了操作数地址。例如,十进制数调整指令 DAA,该指令的功能是 AL 寄存器的内容进行十进制调整,调整后的内容仍放入 AL 中。DAA 指令没有书面指明 AL,而是将 AL 隐含在操作码中,也就是说只要是 DAA 指令,它的操作数就一定在 AL 中。这样的指令还有很多,例如乘法指令、除法指令和换码指令等。

3.3 8086/8088 的指令系统

8086/8088 指令系统中包含 133 条基本指令,如果考虑其中寻址方式的组合,以及数据类型(字节或字),则可构成上千条指令。

8086/8088 的指令分以下几类:①数据传送类指令;②算术运算类指令;③逻辑运算与移位类指令;④串操作类指令;⑤控制转移类指令;⑥处理器控制类指令。

在学习过程中,要正确理解每条指令的助记符、寻址方式、数据类型,并理解指令对标志位的影响,掌握每条指令的功能及其使用注意事项,奠定第 4 章汇编语言程序设计的基础。

视频讲解

3.3.1 数据传送类指令

数据传送指令包括通用数据传送指令、标志寄存器传送指令、目标地址传送指令、输入输出指令等。它们的功能是用于控制数据在计算机各部分之间的传送。具体可实现寄存器与存储器之间、寄存器与外设端口之间以及寄存器与寄存器之间的数据传送等。它们的共

同特点是不影响标志寄存器的内容。

为了方便叙述和学习,本章符号约定如下。

OPD:表示目的操作数(8/16位)。

OPS:表示源操作数(8/16位)。

1. 通用数据传送指令

通用数据传送指令包括MOV指令、XCHG指令、XLAT指令、PUSH指令与POP指令。

1) MOV指令

格式:MOV OPD,OPS

MOV指令是形式简单但使用最多的指令,它可以完成CPU内寄存器之间、寄存器与存储器之间的数据传送,还可以将立即数送入寄存器或内存。

例如:

```
MOV  AL, BL          ;将寄存器BL的内容传送到寄存器AL中
MOV  [DI], AX        ;将寄存器AX的内容传送到DI和DI+1所
                     ;指的内存字单元中
MOV  CX, DS:[1000H]  ;将数据段中偏移地址1000H和1001H单元的
                     ;内容送到CX中
MOV  BL, 40          ;将立即数40传送到寄存器BL中
```

MOV指令在使用时应该注意以下几点。

(1) 立即数、CS和IP不能作为目的操作数,CS和IP的内容不能任意改变,程序是通过转移指令来修改CS和IP的内容。

(2) 两个段寄存器之间不能相互传送数据。确要传送,可以参考下面方法:

```
MOV  AX,DS
MOV  ES,AX
```

(3) 两个存储单元之间不能直接传送。确要传送,可以参考下面方法:

```
MOV  AX,[SI]
MOV  [DI],AX
```

(4) 不能将立即数直接传送到段寄存器。

如MOV DS,2000H为错误指令,可以通过以下两条指令实现:

```
MOV  AX, 2000H
MOV  DS, AX
```

(5) MOV指令的两个操作数的类型和长度必须一致。例如下列指令是错误的,书写时应该注意:

```
MOV  AX, BL          ;类型不一致
MOV  AL, 256         ;长度不一致
```

例3.2 试用MOV指令实现数据段中偏移地址2035H和2045H两内存字节单元内容的交换。

(1) 用直接寻址方式实现。

```
MOV BL, DS:[2035H]
MOV CL, DS:[2045H]
```

```
MOV DS:[2045H], BL
MOV DS:[2035H], CL
```

(2) 用寄存器间接寻址方式实现。

```
MOV SI, 2035H
MOV DI, 2045H
MOV AH, [SI]
MOV AL, [DI]
MOV DS:[2035H], AL
MOV DS:[2045H], AH
```

2) XCHG 指令

格式：XCHG OPD,OPS

功能：交换操作数 OPD、OPS 的值，即 OPD⇔OPS，操作数的类型可以为字节或字。交换只能在通用寄存器之间、通用寄存器与存储器之间进行。

例如：

```
XCHG  AX,BX          ;AX 和 BX 寄存器的内容互换
XCHG  AX,[SI]        ;AX 和数据段中 SI 所指字单元的内容互换
```

3) XLAT 指令

格式：XLAT　或　XLAT 表首址

功能：使 AL 中的值变换为内存表格中的对应值。它是一条隐含寻址方式的指令，将数据段内有效地址为(BX)+(AL)的内存字节单元中数据送入 AL。该指令常用来查表，即将表头地址赋予 BX，再将需求的表内位移地址赋予 AL，最后运用 XLAT 指令即可以将该地址处的表值送到 AL。

例 3.3　在内存数据段中偏移地址 1000H 处存放了如图 3.2 所示的字节类型的数据表，则换码指令可如下列使用方法。

```
…                    ;省略
MOV  BX, 1000H       ;BX 中填写表格的起始地址
MOV  AL, 03H         ;表内位移地址赋予 AL
XLAT
```

上述指令执行后结果为：(AL)=0CH。

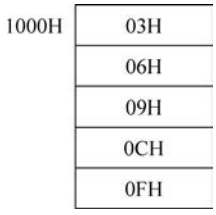

图 3.2　字节类型的数据表

4) PUSH 指令与 POP 指令

这两条指令是堆栈操作指令。所谓堆栈，是以先进后出管理的一段存储区域，常用于子程序调用和中断处理过程。为了能够在子程序返回或中断返回后，程序继续使用子程序调用前或中断发生前各通用寄存器的内容，就必须在子程序或中断程序开始处保护子程序执行过程中要改变的寄存器(包括标志寄存器)的值(称为保护现场)，然后在子程序结束前或中断返回前再恢复这些寄存器的值(称为恢复现场)。

堆栈的存取操作都发生在栈顶，而堆栈指针 SP 一直指向堆栈的栈顶。每执行一个 PUSH 指令，SP 的内容减 2；每执行一个 POP 指令，SP 的内容加 2。

(1) 入栈指令 PUSH。

格式：PUSH OPS

功能：修改指针，(SP)－2→SP，将OPS指明的寄存器、段寄存器或存储器中的一个字数据压入堆栈的顶部。

例3.4 假设在指令执行前，(SP)＝1000H；(AX)＝1234H。

```
PUSH  AX
```

指令执行后，12H被压入堆栈段偏移地址为0FFFH单元，34H被压入堆栈段偏移地址为0FFEH单元，栈顶指针(SP)＝0FFEH。

PUSH指令在使用时应该注意：

① 源操作数只能是16位的，而不能是8位的，例如PUSH　AL就是错误的。

② 源操作数不能为立即数。

(2) 出栈指令POP。

格式：POP OPD

功能：将栈顶的一个字数据送至OPD指明的寄存器、段寄存器(CS除外)或存储器中；修改指针，(SP)＋2→SP。

例3.5 假设在指令执行前，(SP)＝0FFEH，(BX)＝2004H，并且堆栈段栈顶(0FFEH)单元内存放34H，(0FFFH)单元内存放12H。

```
POP  BX
```

指令执行后，1234H被弹出放入BX中，栈顶指针(SP)＝1000H。

POP指令在使用时应该注意：

① 目的操作数只能是16位的，而不能是8位的。

② 立即数、CS不能作为目的操作数。

2. 标志寄存器传送指令

1) 取标志寄存器指令

格式：LAHF

功能：将标志寄存器FR的低8位送入AH中。

2) 设置标志寄存器指令

格式：SAHF

功能：将AH内容送入标志寄存器FR的低8位。

3) 标志进栈指令

格式：PUSHF

功能：将16位标志寄存器FR内容进栈保存。

4) 标志出栈指令

格式：POPF

功能：将当前栈顶两字节内容弹出送到标志寄存器FR中。

3. 目标地址传送指令

1) 取偏移地址指令LEA

格式：LEA　OPD,OPS

功能：将源操作数的偏移地址送到目的操作数。该指令不影响标志位，源操作数必须是存储器操作数，目的操作数必须是16位通用寄存器。

例如：LEA　SI,TABLE　; TABLE 为存储器操作数的符号地址

该指令等效于 MOV SI,OFFSET TABLE(OFFSET 取偏移地址算符,详见第 4 章)。

例如：LEA　AX,[SI]

该指令等效于 MOV AX,SI 指令,与 MOV AX,[SI] 指令的效果不同,注意区别。

2) 传送偏移地址及数据段首址指令 LDS

格式：LDS　OPD,OPS

功能：从源操作数所指定的存储单元中取出某变量的地址指针(共 4 字节),将低地址两字节(偏移量)送到目的操作数,将高地址两字节(段首址)送到 DS 中。该指令对标志位不影响,源操作数是双字类型存储器操作数,目的操作数必须是 16 位通用寄存器。

例 3.6　设数据段中某双字存储单元的偏移地址为 3000H,双字数据为 12345678H。

```
LDS  SI,DS:[3000H]
```

上述指令执行后,(DS)＝1234H,(SI)＝5678H。

3) 传送偏移地址及附加段首址指令 LES

格式：LES　OPD,OPS

功能：从源操作数所指定的存储单元取出某变量的地址指针(共 4 字节),将低地址两字节(偏移量)送到目的操作数,将高地址两字节(变量的段首址)送到 ES 中。

4. 输入输出指令

输入输出指令用于完成输入输出端口与累加器(AL/AX)之间的数据传送,指令中给出输入输出端口的地址。

1) 输入指令 IN

格式：IN　OPD,OPS

功能：从端口 OPS(地址为 n 或在 DX 中)输入 8 位数据到 AL 或输入 16 位数据到 AX。

例如：

```
IN    AL, 40H          ;从 40H 端口读入一字节送入 AL
IN    AX, 80H          ;从 80H 端口读入一字节送入 AL,从 81H 端口
                       ;读入一字节送入 AH
MOV   DX, 8F00H        ;将端口地址 8F00H 送入 DX
IN    AL, DX           ;从 8F00H 端口读入一字节送入 AL
```

2) 输出指令 OUT

格式：OUT　OPD,OPS

功能：从 AL 输出 8 位数据或从 AX 输出 16 位数据到端口(地址为 n 或在 DX 中)。

例如：

```
OUT  40H, AL           ;将 AL 内容送入 40H 端口
OUT  80H, AX           ;将 AL 内容送入 80H 端口,将 AH 的内容送入 81H 端口
MOV  DX, 8F00H         ;将端口地址 8F00H 送入 DX
OUT   DX, AL           ;将 AL 内容送入 8F00H 端口
```

输入输出指令在使用应该注意以下两点。

(1) 输入输出指令对标志寄存器没有影响。

(2) 端口地址大于 255 时,必须用 DX 指定端口地址。

3.3.2 算术运算类指令

在8086/8088指令系统中,具有完备的加、减、乘和除运算,可以完成有符号和无符号8位/16位二进制数的算术运算,以及BCD码表示的十进制数的算术运算。

视频讲解

1. 加法指令

1) 不带进位加法指令ADD

格式:ADD OPD,OPS

功能:将源操作数和目的操作数相加,结果存入目的操作数,即OPS+OPD→OPD。

影响标志位CF、AF、PF、SF、OF和ZF。

2) 带进位加法指令ADC

格式:ADC OPD,OPS

功能:将源操作数和目的操作数相加,并加上CF的值,结果存入目的操作数,即OPS+OPD+CF→OPD。影响标志位CF、AF、PF、SF、OF和ZF。

ADC指令常用于多字的加法。

例3.7 将X1(56781234H)、X2(9F887AB4H)相加,结果存入SUM(低字在前,高字在后)。

```
DSEG  SEGMENT
  X1  DW 1234H,5678H
  X2  DW 7AB4H,9F88H
 SUM  DW 0,0,0
DSEG  ENDS
  …
  MOV  AX, X1
  ADD  AX, X2              ;低字加用ADD指令
  MOV  SUM, AX
  MOV  AX, X1 + 2
  ADC  AX, X2 + 2          ;高字加用ADC指令
  MOV  SUM + 2, AX
  MOV  AX, 0
  ADC   AX, 0              ;处理相加发生进位情况
  MOV  SUM + 4, AX
```

3) 加1指令

格式:INC OPD

功能:将目的操作数OPD的内容加1,并将结果回送到目的操作数。影响标志位AF、PF、SF、OF和ZF,但不影响CF标志。

例如:

```
INC  AX                    ;AX中内容加1,结果送回AX
INC  BL                    ;BL中内容加1,结果送回BL
INC  BYTE PTR [SI]         ;将SI所指内存字节单元内容加1,并回存
```

2. 减法指令

1) 不带借位的减法指令SUB

格式:SUB OPD,OPS

功能:目的操作数减去源操作数,结果回送到目的操作数,即OPD−OPS→OPD,结果

影响标志位 CF、AF、PF、SF、OF 和 ZF。

例如：SUB BX,CX ; BX 的内容减去 CX 的内容,结果送入 BX

2）带借位的减法指令 SBB

格式：SBB OPD,OPS

功能：目的操作数减去源操作数,再减去 CF 的值,结果回送到目的操作数,即 OPD－OPS－CF→OPD,结果影响标志位 CF、AF、PF、SF、OF 和 ZF。

例如：

```
SBB [SI], AL              ;SI 所指字节单元内容减 AL 的值,再减 CF
                          ;结果存回原内存单元
```

3）减 1 指令

格式：DEC OPD

功能：将目的操作数 OPD 的内容减 1,并将结果回送到目的操作数。影响标志位 AF、PF、SF、OF 和 ZF,但不影响 CF 标志。

例如：

```
DEC AX                    ;AX 的内容减 1,结果送回 AX
DEC CL                    ;CL 的内容减 1,结果送回 CL
```

4）比较指令 CMP

格式：CMP OPD,OPS

功能：目的操作数 OPD 减去源操作数 OPS,结果不送回。但影响标志位 CF、AF、PF、SF、OF 和 ZF。

例如：CMP AL,09H ;将 AL 的内容和 09H 比较,结果影响标志位

可以根据标志位的变化,来判断比较结果。

通过 ZF 的值来判断两数是否相等。若 ZF＝1,则说明两数相等；否则,两数不等。

通过 CF、OF 和 SF 的变化来判断无符号数或有符号数的大小：

（1）对于无符号数,如果 CF＝0,则目的操作数比源操作数大；如果 CF＝1,则目的操作数比源操作数小。

（2）对于有符号数,如果 OF＝SF,则目的操作数比源操作数大；如果 OF≠SF,则目的操作数比源操作数小。

5）求补指令

格式：NEG OPD

功能：由目的操作数 OPD 求补,将其结果送回目的操作数。需要注意的是,它实际做 0－OPD→OPD 运算。影响标志位 CF、AF、PF、SF、OF 和 ZF。

例 3.8

```
MOV  AL, 05H
NEG  AL          ;(AL) = 0FBH,CF = 1
MOV  AL, -05H
NEG  AL          ;(AL) = 05H,CF = 1
```

3. 乘法指令

1）无符号数乘法指令

格式：MUL OPS

功能：字节乘法，(AL)×(OPS)→AX。

字乘法，(AX)×(OPS)→(DX,AX)。

例如：MUL BL

指令执行前，(AL)=0B4H=180，(BL)=11H=17。

指令执行后，(AX)=0BF4H=3060，CF=1，OF=1。

MUL 指令在使用时应该注意：

① OPS 不能是立即数。例如，MUL 20H 为错误指令。

② MUL 指令影响 CF、OF 标志，不影响 SF、ZF、AF 和 PF。对于字节乘法，若 AH≠0，也即 AH 存在有效位，则 CF=1，OF=1；否则 CF=0，OF=0。对于字乘法，若 DX≠0，也即 DX 存在有效位，则 CF=1，OF=1；否则 CF=0，OF=0。

2) 有符号数乘法指令

格式：IMUL OPS

功能：字节乘法：(AL)×(OPS)→AX。

字乘法：(AX)×(OPS)→(DX,AX)。

例如：IMUL BL

指令执行前，(AL)=0B4H=－76，(BL)=11H=17。

指令执行后，(AX)=0FAF4H =－1292，CF=1，OF=1。

IMUL 指令在使用时应该注意的内容参见 MUL 指令。

4. 符号扩展指令

1) 字节扩展为字指令

格式：CBW

功能：将 AL 的内容从字节扩展为字，存放到 AX 中。若 AL 中数据的最高位为 0，则(AH)=00H；若 AL 中数据的最高位为 1，则(AH)=0FFH。该指令不影响标志位。

例 3.9
```
MOV  AL, -7
CBW
```

执行后，(AX)=0FFF9H。

2) 扩展为双字指令

格式：CWD

功能：将 AX 的内容从字扩展为双字，存放到 DX、AX 中。若 AX 中数据的最高位为 0，则(DX)=0000H；若 AX 中数据的最高位为 1，则(DX)=0FFFFH。该指令不影响标志位。

例 3.10
```
MOV DX, 0
MOV AX, 0FFABH
CWD
```

指令执行后，(DX)=0FFFFH；(AX)=0FFABH。

5. 除运算指令

1) 无符号数除法指令

格式：DIV OPS

功能：字节除法，(AX)/(OPS)→AL(商)，AH(余数)。

字除法，(DX,AX)/(OPS)→AX(商)，DX(余数)。

例 3.11
```
DSEG  SEGMENT
  A   DW  0400H
  B   DW  00B4H
  C   DW  ?
  D   DW  ?
DSEG  ENDS
```

在以上数据段定义的基础上，求无符号数 A/B，商放入 C，余数放入 D。

```
  …
MOV  AX, A
MOV  DX, 0
DIV   B
MOV  C, AX
MOV  D, DX
```

DIV 指令在使用时应该注意：

① OPS 不能是立即数。

② 除法指令不影响标志位，除 0 会导致结果溢出，产生溢出中断。

2）有符号数除法指令

格式：IDIV OPS

功能：字节除法，(AX)/(OPS)→AL(商)，AH(余数)。

字除法，(DX,AX)/(OPS)→AX(商)，DX(余数)。

例 3.12 以下指令序列实现了有符号除法－4001H/4。

```
MOV  AX, -4001H
CWD
MOV  CX, 4
IDIV  CX
```

结果：(AX)＝0F000H；(DX)＝0FFFFH。

6. 十进制调整指令

上述算术运算指令都是二进制运算指令，如何利用它们来进行 BCD 码十进制运算？一般方法是：首先对 BCD 码表示的十进制数进行二进制运算，然后再使用调整指令对运算结果进行调整，得出正确的 BCD 码表示的十进制运算结果。下面分别介绍这些调整指令。

1）加法的 BCD 码调整指令

(1) 压缩的 BCD 码调整指令。

格式：DAA

功能：将 AL 中二进制加法运算的结果调整为两位压缩 BCD 码，结果仍保留在 AL 中。

调整的方法：若 AL 的低 4 位大于 9，则 AL 的内容加 06H，并 AF 位置 1；若 AL 的高 4 位大于 9，则 AL 的内容加 60H，并 CF 位置 1；若 AF＝1，则低 4 位要加 6；若 CF＝1，则高 4 位要加 6。

例 3.13
```
MOV  BL, 35H
MOV  AL, 85H
ADD  AL, BL
DAA
```

上述指令序列执行后，(AL)＝20H，CF＝1，AF＝1。

(2) 非压缩的 BCD 码调整指令。

格式：AAA

功能：将 AL 中二进制加法运算结果调整为一位非压缩 BCD 码，调整后的结果仍保留在 AL 中，如果向高位有进位(AF=1,CF=1)，AH 的内容加 1。

调整的方法：若 AL 的低 4 位大于 9 或 AF=1，则自动将 AL 的内容加 06H，AH 内容加 1 并置 AF=CF=1，将 AL 的高 4 位清 0；若 AL 的低 4 位小于或等于 9，则仅将 AL 的高 4 位清 0，并且 AF→CF。

例 3.14 两个 BCD 码数(AX)=0807H、(BX)=0905H，要求将(AX)+(BX)的结果存放到 BCDBUF 开始的单元(低字节在前)。

```
PUSH   AX                  ;暂存 AX
MOV    AH, 0               ;AH 清 0
ADD    AL, BL              ;低字节直接相加
AAA                        ;调整
MOV    BCDBUF, AL          ;存放和的低字节
POP    AX                  ;恢复 AX
MOV    AL, AH
MOV    AH, 0               ;AH 清 0
ADC    AL, BH              ;带进位的加法
AAA                        ;调整
MOV    BCDBUF + 1, AL      ;存放和的高字节
MOV    BCDBUF + 2, AH      ;存放和的高字节的进位
```

DAA、AAA 指令在使用时应该注意：

① DAA、AAA 指令一般是紧跟在 ADD 或 ADC 指令后使用，单独使用没有意义。

② 调整指令只对 AL 的内容进行调整，故在调整前，务必保证待调整结果出现在 AL 中。

2) 减法的 BCD 码调整指令

(1) 压缩的 BCD 码调整指令。

格式：DAS

功能：将 AL 中二进制减法运算的结果调整为两位压缩 BCD 码，结果仍保留在 AL 中。

调整的方法：若 AF=1 或 AL 的低 4 位大于 9，则自动(AL)−06H→AL,1→AF；若 CF=1 或 AL 的高 4 位大于 9，则自动(AL)−60H→AL,1→CF。

例 3.15
```
SUB   AL, BL       ;先使用二进制减运算指令
DAS                ;在使用调整指令,可得 BCD 码运算结果
```

(2) 非压缩的 BCD 码调整指令。

格式：AAS

功能：将 AL 中二进制减法运算结果调整为一位非压缩 BCD 码，如果有借位，则保留在 CF 中。

调整的方法：若 AL 的低 4 位大于 9 或 AF=1，则自动将 AL 的内容加 06H，AH 内容减 1 并置 AF=CF=1，将 AL 的高 4 位清 0；若 AL 的低 4 位小于或等于 9，则仅将 AL 的高 4 位清 0，并且 AF→CF。

DAS、AAS 指令在使用时应注意点可参考加法调整指令。

3）乘法的非压缩 BCD 码调整指令

格式：AAM

功能：将 AL 中二进制乘法运算结果调整为两位非压缩 BCD 码，高位放在 AH，低位放在 AL。影响标志位 PF、SF 和 ZF。该指令必须紧跟在 MUL 之后，且被乘数和乘数必须用非压缩的 BCD 码表示。

例 3.16
```
AND  AL, 0FH          ;确保被乘数为非压缩的 BCD 码
AND  BL, 0FH          ;确保乘数为非压缩的 BCD 码
MUL  BL
AAM
```

4）除法的非压缩 BCD 码调整指令

格式：AAD

功能：用在两位非压缩的 BCD 码相除之前，将 AX 内容调整为二进制数。

例 3.17
```
MOV  CL, 08H          ;送非压缩 BCD 码除数，由于二进制数也是 08H
                      ;没有用调整指令，否则也要调整
MOV  AX, 0309H        ;送非压缩 BCD 码被除数
AAD                   ;调整被除数为二进制数
DIV  CL               ;结果商和余数为二进制数
```

3.3.3 逻辑运算与移位类指令


视频讲解

1. 逻辑运算指令

逻辑运算指令包括“非”“与”“测试”“或”“异或”指令，可以对字或字节按位进行逻辑运算。

1）非运算指令

格式：NOT OPD

功能：将目的操作数的内容按位取反后，再送回目的操作数。该指令不影响标志位。

例如：NOT AX

执行前：(AX)＝0AAAAH。

执行后：(AX)＝5555H。

2）与运算指令

格式：AND OPD,OPS

功能：将目的操作数的内容与源操作数按位相与，结果送回目的操作数。影响标志位 SF、ZF、PF，使 OF＝0，CF＝0，对 AF 无定义。

例如：AND AL,0FH

执行前：(AL)＝39H。

执行后：(AL)＝09H。

AND 指令常用于屏蔽不需要的位，上例中将 AL 高 4 位屏蔽，取得低 4 位。

3）测试指令

格式：TEST OPD,OPS

功能：将目的操作数的内容与源操作数按位相与，但结果不送回目的操作数。影响标志位 SF、ZF、PF，使 OF＝0，CF＝0，对 AF 无定义。

例如：TEST AL,80H

执行前：(AL)＝39H。

执行后：ZF＝1。

该指令可以用于判断目的操作数的某个数位是否为1，上例中可以根据ZF＝1，判断出AL内容的最高位为0。

4) 或运算指令

格式：OR　OPD,OPS

功能：将目的操作数的内容与源操作数按位相或，结果送回目的操作数。影响标志位SF、ZF、PF，使OF＝0，CF＝0，对AF无定义。

例如：OR　AX,55H

执行前：(AX)＝0749H。

执行后：(AX)＝075DH。

该指令常用来将目的操作数的某一位或几位置1。

5) 异或运算指令

格式：XOR　OPD,OPS

功能：将目的操作数的内容与源操作数按位异或，结果送回目的操作数。影响标志位SF、ZF、PF，使OF＝0，CF＝0，对AF无定义。

例如：XOR　AX,0101H

执行前：(AX)＝0749H。

执行后：(AX)＝0648H。

由于某个操作数和同一个数异或结果为0，故异或运算常被用来比较两数是否相等或初始化某数为0。

2. 移位指令

这组指令可以对字节或字中的各位进行算术移位和逻辑移位。移位次数可以是1，也可以大于1。若移位次数大于1，必须将次数预先放入CL。也就是说，移位指令中的移位次数的源操作数只能是1或CL。这组指令影响除AF以外的各个标志位。

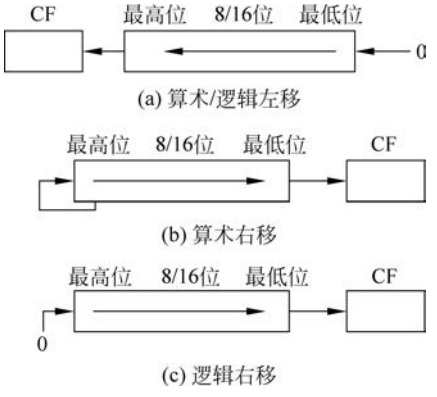

图3.3　移位指令操作过程

1) 算术左移指令

格式：SAL　OPD,OPS

功能：根据源操作数OPS中的移位次数，将目的操作数的内容连续进行左移操作，每次高位进入CF，最低位补0，如图3.3(a)所示。

例3.18
```
MOV　CL, 3
SAL　AL, CL
```

执行前：(AL)＝01H。

执行后：(AL)＝08H，CF＝0。

无符号数的算术左移一位相当于目的操作数乘以2。

2) 逻辑左移指令

格式：SHL　OPD,OPS

功能：与算术左移指令SAL完全相同。

3）算术右移指令

格式：SAR　OPD,OPS

功能：根据源操作数 OPS 中的移位次数，将目的操作数的内容连续进行右移操作，每次低位进入 CF，最高位用移位前的值填补，如图 3.3(b)所示。

例如：SAR　BH,CL

执行前：(BH)＝84H,(CL)＝2。

执行后：(BH)＝0E1H,CF＝0。

4）逻辑右移指令

格式：SHR　OPD,OPS

功能：根据源操作数 OPS 中的移位次数，将目的操作数的内容连续进行右移操作，每次低位进入 CF，最高位补 0，如图 3.3(c)所示。

例如：SHR　AL,CL

执行前：(AL)＝9AH,(CL)＝4。

执行后：(AL)＝09H,CF＝1。

3. 循环移位指令

8086/8088 指令系统中有 4 条循环移位指令，其中 2 条为不带进位的循环移位指令，另 2 条为带进位的循环移位指令。源操作数 OPS 的循环移位次数的设置和移位指令相同。这组指令只影响 CF、OF 标志位。

1）循环左移指令

格式：ROL　OPD,OPS

功能：根据源操作数 OPS 中的移位次数，将目的操作数的内容连续进行循环左移操作，如图 3.4(a)所示。

例如：ROL　DL,CL

执行前：(DL)＝0FAH,(CL)＝4。

执行后：(DL)＝0AFH,CF＝1。

2）循环右移指令

格式：ROR　OPD,OPS

功能：根据源操作数 OPS 中的移位次数，将目的操作数的内容连续进行循环右移操作，如图 3.4(b)所示。

例如：ROR　AL,1

执行前：(AL)＝0FAH。

执行后：(AL)＝07DH,CF＝0。

3）带进位的循环左移指令

格式：RCL　OPD,OPS

功能：根据源操作数 OPS 中的移位次数，连续对目的操作数的内容带 CF 循环左移操作，如图 3.4(c)所示。

4）带进位的循环右移指令

格式：RCR　OPD,OPS

功能：根据源操作数 OPS 中的移位次数，连续对目的操作数的内容带 CF 循环右移操

作,如图3.4(d)所示。

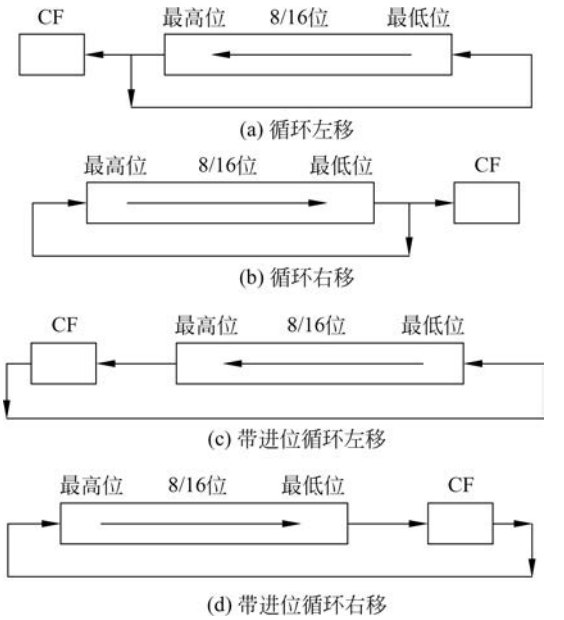

图3.4 循环移位指令的操作过程

例3.19 有一无符号32位二进制数存放在DX、AX中,其高16位在DX中,低16位在AX中,以下指令序列实现对该数的除2操作。

```
SHR   DX, 1
RCR   AX, 1
```

循环移位指令常用于按位检查某单元的内容或实现某单元的半字或半字节互换等。

视频讲解

3.3.4 串操作类指令

字符串是指存储器中顺序存放、类型相同的字节或字的序列。串操作是指对串中每个元素都执行同样的操作,如串传送、比较以及查找等。规定一个字符串的长度最长不能超过64KB,字符串常设置在数据段或附加段中。

串操作指令约定源串存放在数据段,并必须用SI提供源串的偏移地址;目的串存放在附加段,并必须用DI提供目的串的偏移地址;当方向标志DF=0,地址指针SI、DI自动增加1(字节串)或2(字串),当方向标志DF=1,地址指针SI、DI自动减少1(字节串)或2(字串)。串操作指令前不加重复前缀,串操作只执行一次。如果重复执行串操作,可以用CX存放重复的次数,每重复执行一次,CX内容减1。当CX内容减为0时,串操作停止。

在介绍串操作指令前,首先介绍和串操作指令配合使用的几个重复前缀指令。

1. 重复指令前缀

1) 无条件重复前缀

格式:REP

功能:用于一个串操作指令的前缀,每重复执行一次串操作指令,CX的内容减1,直到(CX)=0为止。

执行步骤如下：

(1) 先判断 CX 的内容，如(CX)＝0，则串操作停止，否则执行第(2)步；

(2) (CX)－1→CX；

(3) 执行其后的串操作指令，转第(1)步。

2) 相等/为零重复前缀

格式：REPE

功能：用于一个串操作指令的前缀，每重复执行一次串操作指令，CX 的内容减 1，直到(CX)＝0 或 ZF＝0 为止。

执行步骤如下：

(1) 先判断 CX 的内容，如(CX)＝0 或 ZF＝0，则串操作停止，否则执行第(2)步；

(2) (CX)－1→CX；

(3) 执行其后的串操作指令，转第(1)步。

3) 不相等/不为零重复前缀

格式：REPNE

功能：用于一个串操作指令的前缀，每重复执行一次串操作指令，CX 的内容减 1，直到(CX)＝0 或 ZF＝1 为止。

执行步骤如下：

(1) 先判断 CX 的内容，如(CX)＝0 或 ZF＝1，则串操作停止，否则执行第(2)步；

(2) (CX)－1→CX；

(3) 执行其后的串操作指令，转第(1)步。

2. 数据字节串/字串传送指令

格式：MOVSB/MOVSW

功能：将数据段中由(DS:SI)指向的源串的一字节(字)传送到附加段由(ES:DI)指向的目的串中，且相应修改地址指针，使其指向下一字节(字)。

字节串：当 DF＝0 时，(SI)＋1→SI，(DI)＋1→DI；当 DF＝1 时，(SI)－1→SI，(DI)－1→DI。

字串：当 DF＝0 时，(SI)＋2→SI，(DI)＋2→DI；当 DF＝1 时，(SI)－2→SI，(DI)－2→DI。

例 3.20 将内存单元首地址 3100H 起的 100 字节传送到首地址 2800H 的内存单元。

```
CLD
MOV  SI, 3100H
MOV  DI, 2800H
MOV  CX, 100
REP   MOVSB
```

3. 数据字节串/字串比较指令

格式：CMPSB/CMPSW

功能：将数据段中由(DS:SI)指向源串的一字节(字)减去附加段由(ES:DI)指向的目的串的一字节(字)，不回送结果，只根据结果影响标志位，并相应修改地址指针，使其指向下一字节(字)。

字节串：当 DF＝0 时，(SI)＋1→SI，(DI)＋1→DI；当 DF＝1 时，(SI)－1→SI，(DI)－1→DI。

字串：当 DF＝0 时，(SI)＋2→SI，(DI)＋2→DI；当 DF＝1 时，(SI)－2→SI，(DI)－2→DI。

例 3.21 检查内存单元首地址 2200H 起的 50 字节与首地址 3200H 起的 50 字节是否对应相等，如果相等，则 BX＝0；如果不相等，BX 指向第一个不相等的字节单元。AL 存放第一个不相等的源串内容。

```
    CLD
    MOV  SI, 2200H
    MOV  DI, 3200H
    MOV  CX, 50
    REPE CMPSB
    JZ   LP1
    DEC  SI
    MOV  BX, SI
    MOV  AL, [SI]
    JMP  LP2
LP1:MOV  BX, 0
LP2:
```

4. 数据字节串/字串检索指令

格式：SCASB/SCASW

功能：将 AL(AX)的内容减去附加段由(ES:DI)指向的目的串的一字节(字)，不回送结果，只根据结果影响标志位，并相应修改地址指针，使其指向下一字节(字)。

字节串：当 DF＝0 时，(DI)＋1→DI；当 DF＝1 时，(DI)－1→DI。

字串：当 DF＝0 时，(DI)＋2→DI；当 DF＝1 时，(DI)－2→DI。

例 3.22 在内存附加段首地址为 4300H 起的 100 字节中，查找是否有 *，如果有，则将偏移地址送入 BX，否则 BX＝0。

```
    CLD
    MOV    DI, 4300H
    MOV    AL, '*'
    REPNZ  SCASB
    JNZ    LP1
    DEC    DI
    MOV    BX, DI                ;找到"*",偏移地址送入 BX
    JMP    LP2
LP1:MOV    BX, 0                 ;未找到
LP2:
```

5. 数据字节串/字串读出指令

格式：LODSB/LODSW

功能：将数据段中由(DS：SI)指向源串的一字节(字)读出，放入 AL(AX)中，并相应修改地址指针，使其指向下一字节(字)。

字节串：当 DF＝0 时，(SI)＋1→SI；当 DF＝1 时，(SI)－1→SI。

字串：当 DF＝0 时，(SI)＋2→SI；当 DF＝1 时，(SI)－2→SI。

6. 数据字节串/字串写入指令

格式：STOSB/STOSW

功能：将 AL(AX)的内容写入附加段中由(ES:DI)指向的目的串一字节(字)中，并相应修改地址指针，使其指向下一字节(字)。

字节串：当 DF＝0 时，(DI)＋1→DI；当 DF＝1 时，(DI)－1→DI。

字串：当 DF＝0 时，(DI)＋2→DI；当 DF＝1 时，(DI)－2→DI。

例 3.23 将内存数据段首地址为 1800H 起的 100 字节清 0。

```
CLD
MOV  DI, 1800H
MOV  CX, 100
XOR  AL, AL
REP  STOSB
```

3.3.5 控制转移类指令


视频讲解

一般情况下指令是顺序地逐条执行的，但实际上程序不可能总是顺序执行，而经常要改变程序的执行流程。这里介绍的控制转移指令就是用来控制程序的执行流程的。程序执行顺序的改变实际是通过修改代码段寄存器 CS 和指令指针 IP 的内容来实现的。

8086 有 5 种类型的转移指令，它们是无条件转移、条件转移、循环控制、子程序调用及返回和与中断有关的指令。

1. 无条件转移指令

无条件地转移到指令指定的地址去执行从该地址开始的指令。无条件转移可以分为段内转移和段间转移。具体有以下 5 种基本格式。

1) 段内直接短转移

格式：JMP SHORT 目标标号

功能：无条件地转移到标号所指定的目标地址去执行程序。短转移时，目标地址与 JMP 指令的下一条指令地址之差为－128～＋127 字节。

2) 段内直接转移

格式：JMP 目标标号
　　JMP NEAR PTR 目标标号

功能：无条件地转移到标号所指定的目标地址去执行程序，但转移的范围扩大到－32 768～＋32 767 字节，也即跳转地址的范围为 16 位带符号二进制数的范围。

3) 段内间接转移

格式：JMP WORD PTR OPD

功能：无条件转移到 OPD 所指定的目标地址去执行程序。OPD 只能是 16 位寄存器或两个连续存储的内存字节单元。转移范围为 64KB。

例如：JMP DX

执行前：(DX)＝2000H，(IP)＝1260H，(CS)＝3000H。

执行后：(IP)＝2000H，CPU 转到地址为 32000H 的单元执行程序。

4) 段间直接转移

格式：JMP FAR PTR 目标标号

功能：无条件转移到目标标号所指定地址去执行程序。将目标标号所在的段基址送入CS,将目标标号相对所在段的段内偏移地址送入IP。可以转移范围为1MB。

5) 段间间接转移

格式：JMP DWORD PTR OPD

功能：无条件转移到目的操作数OPD所指定地址去执行程序。目的操作数为双字,将目的操作数的第一个字送入IP,将目的操作数的第二个字送入CS。可以转移范围为1MB。

例如：JMP DWORD PTR[BX]

执行前：(BX)=2000H,(DS)=5000H,(52000H)=0200H,(52002H)=0400H。

执行后：(IP)=0200H,(CS)=0400H,CPU转到地址为04200H的单元执行程序。

2. 条件转移指令

测试上一条指令对标志位的影响,从而决定程序执行的流程。若满足条件则转移；若不满足条件则顺序执行。注意,转移范围都只有－128～＋127字节,如果需要转移的地址超出了该范围,可以将条件转移指令与无条件转移指令配合使用。所有条件转移指令对标志位均无影响。

1) 单标志位转移指令

单标志位条件转移指令如表3.4所示。

表3.4 单标志位条件转移指令

指 令	测试条件	含 义	指 令	测试条件	含 义
JZ/JE	ZF=1	0/相等则转移	JP/JPE	PF=1	低8位中1的个数为偶数则转移
JNZ/JNE	ZF=0	非0/不相等则转移			
JS	SF=1	结果为负则转移	JNP/JPO	PF=0	低8位中1的个数为奇数则转移
JNS	SF=0	结果非负则转移			
JO	OF=1	结果溢出则转移	JC	CF=1	有进位则转移
JNO	OF=0	结果不溢出则转移	JNC	CF=0	无进位则转移

例3.24 运用条件转移指令,判断AX的内容,如果(AX)=04H,则设置CX为0,否则设置CX为0FFFFH。

```
      MOV   BX, 04H
      CMP   AX, BX
      JZ    LP1
LP2:  MOV   CX, 0FFFFH
      JMP   LP3
LP1:  MOV   CX, 0
LP3:
```

2) 无符号数的条件转移指令

该组转移指令用于无符号数的比较,并根据比较的结果进行转移。具体无符号数的条件转移指令如表3.5所示。

表3.5 无符号数的条件转移指令

指 令	测 试 条 件	说明(*A*,*B* 两数关系)
JA/JNBE	CF=0 ∧ ZF=0	高于/不低于且不等于(即$A>B$)则转移
JAE/JNB	CF=0 ∨ ZF=1	高于等于/不低于(即$A\geqslant B$)则转移

续表

指　令	测试条件	说明(*A*,*B* 两数关系)
JB/JNAE	CF=1 ∧ ZF=0	低于/不高于且不等于(即 $A<B$)则转移
JBE/JNA	CF=1 ∨ ZF=1	低于等于/不高于(即 $A\leqslant B$)则转移

例 3.25　以下指令完成判断 AL 的内容,如果低于 60,则进行 AL 内容加 5 的操作。

```
    CMP   AL, 60
    JAE   LP1
    ADD   AL, 5
LP1:
```

3) 有符号数的条件转移指令

该组转移指令用于有符号数的比较,并根据比较的结果进行转移。具体转移指令如表 3.6 所示。

表 3.6　有符号数的条件转移指令

指　令	测试条件	说明(*A*,*B* 两数关系)
JG/JNLE	SF=OF ∧ ZF=0	大于/不小于且不等于(即 $A>B$)则转移
JGE/JNL	SF=OF ∨ ZF=1	大于或等于/不小于(即 $A\geqslant B$)则转移
JL/JNGE	SF≠OF ∧ ZF=0	小于/不大于且不等于(即 $A<B$)则转移
JLE/JNG	SF≠OF∨ ZF=1	小于或等于/不大于(即 $A\leqslant B$)则转移

例 3.26

```
        MOV  AL, -40H       ;  0C0H→AL
        CMP  AL, 50H
        JG   LP1            ; -40H<50H  不转移
        MOV  CX, 0FFFFH
        JMP  LP2
LP1:    MOV  CX, 0
LP2:
```

以上指令序列执行后,(CX)=0FFFFH。注意,指令序列中的 JG 若改为 JA,CX 的内容则为 0。

4) 测试 CX 条件转移指令

格式:JCXZ　目标标号

功能:若(CX)=0,则转移到目标标号所指定地址去执行程序。

3. 循环控制指令

8086/8088 指令系统有三条循环控制指令,一般用它们来实现程序循环,循环的次数必须放在 CX 寄存器中,这组指令也不影响标志位。

1) 计数循环指令

格式:LOOP　标号

功能:每执行一次 LOOP 指令,CX 的内容减 1,若 CX≠0,则循环转移到标号所指定的目标地址去重复执行程序,直到 CX=0,退出循环,接着执行 LOOP 指令的下一条指令。

例 3.27　以下为使用 LOOP 指令编写的延时程序段。

```
      MOV  CX, 0100H            ;设置循环次数
DELAY:LOOP DELAY
```

LOOP 指令执行转移时,用 9 个时钟周期,结束循环指向下一条指令时,用 5 个时钟周

期,程序员可以设置循环次数来控制延迟的时间。

2) 相等/为零计数循环指令

格式:LOOPE/LOOPZ 标号

功能:每执行一次循环指令,CX的内容减1,若CX≠0且ZF=1,则循环转移到标号所指定的目标地址去重复执行程序,否则执行循环指令的下一条指令。

3) 不相等/不为零计数循环指令

格式:LOOPNE/LOOPNZ 标号

功能:每执行一次循环指令,CX的内容减1,若CX≠0且ZF=0,则循环转移到标号所指定的目标地址去重复执行程序,否则执行循环指令的下一条指令。

4. 子程序调用和返回指令

在8086/8088指令系统中,调用子程序或从子程序返回的指令分别为CALL和RET。

1) 子程序调用指令

(1) 段内直接调用。

格式:CALL 标号

功能:首先将返回地址(CALL指令的下一条指令,16位偏移地址)压入堆栈,然后将标号所指的子程序在本段中的偏移地址送入IP,转子程序执行。

(2) 段内间接调用。

格式:CALL WORD PTR OPD

功能:首先将返回地址(CALL指令的下一条指令,16位偏移地址)压入堆栈,然后将目的操作数的内容送入IP,转至同一段内的子程序执行。

(3) 段间直接调用。

格式:CALL FAR PTR 标号

功能:首先将断点地址CS、IP顺序压入堆栈(共4字节),然后将标号所在的段基址送入CS,将标号相对所在段的偏移地址送入IP,转子程序执行。

(4) 段间间接调用。

格式:CALL DWORD PTR OPD

功能:首先将断点地址CS、IP顺序压入堆栈(共4字节),然后将有效地址指定的4字节送入IP、CS,低地址的两字节送入IP,高地址的两字节送入CS,转子程序执行。

2) 返回指令

子程序的最后一条指令为返回指令,用来控制程序返回断点地址处(相应CALL指令的下一条指令)继续执行下去。

(1) 返回指令。

格式:RET

功能:把断点地址从堆栈弹出送入IP或IP、CS。如果该子程序为FAR类型,首先从堆栈弹出一个字送入IP(SP+2→SP),再从堆栈弹出一个字送入CS(SP+2→SP);如果该子程序为NEAR类型,从堆栈弹出一个字送入IP(SP+2→SP),从而返回主程序。

(2) 带弹出值的返回指令。

格式:RET n

功能:n为偶数,在执行RET指令后,再修改指针SP+n→SP,也即先从堆栈弹出断点

地址送 IP 或 IP、CS，再废除栈顶的 n 字节。

CALL 和 RET 指令在使用应该注意以下几方面。

① CALL 和 RET 指令不影响标志位。

② CALL 和 RET 指令必须成对使用，与无条件转移指令的不同之处，在于它含有将断点地址入栈和出栈的操作。

5. 中断和中断返回指令

中断和中断返回指令能使 CPU 暂停执行后续指令，而转去执行相应的中断服务程序或从中断服务程序返回主程序。

1）软中断指令

格式：INT n

功能：n 为中断类型码，可以取 0～0FFH 的 256 个值。每个中断类型码在中断矢量表中占 4 字节，前 2 字节用来存放中断服务程序入口地址的偏移地址，后 2 字节用来存放段基址。CPU 执行 INT 指令时，首先将标志寄存器 FR 压栈，接着清除 IF、TF，然后将当前程序断点的段基址和偏移地址入栈保护，最后将中断矢量表中与中断类型码对应的 4 字节内容先后送入 IP、CS，这样 CPU 转去执行中断服务程序。

2）中断返回指令

格式：IRET

功能：放在中断服务程序的出口处，由它从堆栈中弹出程序断点分别送 IP、CS，并弹出一个字送入标志寄存器 FR，以退出中断，返回到断点处执行后续程序。

中断服务程序的最后一条指令必须是 IRET。

3）溢出中断指令

格式：INTO

功能：该指令为单字节指令，中断类型码为 4，放在有符号的算术运算指令之后，仅当运算产生溢出(OF＝1)时，才向 CPU 发出溢出中断请求。

3.3.6 处理器控制类指令

1. 标志位操作指令

1）进位位清 0 指令

格式：CLC

功能：置 CF＝0。

2）进位位求反指令

格式：CMC

功能：$CF \leftarrow \overline{CF}$。

3）进位位置 1 指令

格式：STC

功能：置 CF＝1。

4）关中断指令

格式：CLI

功能：置 IF＝0，禁止外部可屏蔽中断。

5) 开中断指令

格式：STI

功能：置 IF=1,允许外部可屏蔽中断。

6) 方向标志清 0 指令

格式：CLD

功能：置 DF=0。

7) 方向标志置 1 指令

格式：STD

功能：置 DF=1。

2. 外部同步指令

1) 空操作指令

格式：NOP

功能：不执行任何操作,其机器码占 1 字节。

2) 暂停指令

格式：HLT

功能：该指令执行后,使机器暂停工作,使处理器处于停机状态,以等待一次外部中断到来,中断结束后,程序继续执行,处理器继续工作。

3) 交权指令

格式：ESC

功能：协处理器在系统加电工作后,就不断检测 CPU 是否需要协助工作,当发现 ESC 指令时,被选定的协处理器便开始工作。

4) 等待指令

格式：WAIT

功能：该指令每隔 5 个时钟周期就测试一次 $\overline{\text{TEST}}$ 信号,若该信号为高电平,CPU 则继续执行 WAIT 指令,进入等待状态；否则结束等待,执行后续指令。

3.4 80386 的寻址方式和指令系统

3.4.1 80386 的寻址方式

80386 的寻址方式和 8086 一样,包括立即寻址、寄存器寻址、直接寻址和寄存器间接寻址。但是由于 80386 的存储器组织方式和 8086 有差别,故寄存器间接寻址有别于 8086。

按照 80386 系统的存储器组织方式,逻辑地址由段选择子和偏移量组成。偏移量由以下 4 个分量计算得到。

① 基址。任何通用寄存器都可以作为基址寄存器,其内容即为基址。

② 位移量。在指令操作码后面的 32 位、16 位或 8 位数。

③ 变址。除了 ESP 寄存器外,任何通用寄存器都可以作为变址寄存器,其内容即为变址。

④ 比例因子。变址寄存器的值可以乘以一个比例因子,根据操作数的长度可以为 1 字节、2 字节、4 字节或 8 字节,比例因子相应可以为 1、2、4 或 8。

由上面 4 个分量计算偏移量的方法为：

偏移量＝基址＋变址×比例因子＋位移量

按照 4 个分量组合偏移量的不同方法，可以有 9 种存储器寻址方式，其中 8 种属于寄存器间接寻址。

(1) 直接寻址方式。

位移量就是操作数的有效地址，此位移量包含在指令中。

例如：DEC　WORD PTR [200]　　;有效地址为 200

(2) 寄存器间接寻址方式。

基址寄存器的内容为操作数的有效地址。

例如：MOV　[EBX],EAX　　;有效地址在 EBX 中

(3) 基址寻址方式。

基址寄存器的内容和位移量相加形成有效地址。

例如：MOV　[EBX＋100],EAX　　;有效地址为 EBX 的内容加 100

(4) 变址寻址方式。

变址寄存器的内容和位移量相加形成有效地址。

例如：SUB　EAX,[ESI],20　　;有效地址为 ESI 的内容加 20

(5) 带比例因子的变址寻址方式。

变址寄存器的内容乘以比例因子，在加位移量相加形成有效地址。

例如：SUB　EAX,[ESI＊8],7　　;有效地址为 ESI 的内容乘以 8 再加 7

(6) 基址变址寻址方式。

基址寄存器的内容加上变址寄存器的内容组成有效地址。

例如：SUB　EAX,[ESI][EBX]　　;有效地址为 EBX 的内容加 ESI 的内容

(7) 基址加带比例因子的变址寻址方式。

变址寄存器的内容乘以比例因子再加上基址寄存器的内容组成有效地址。

例如：MOV　ECX,[EDI＊2][EBX]　　;有效地址为 EDI 的内容乘以 2 再加 EBX 的内容

(8) 带位移量的基址加变址寻址方式。

基址寄存器的内容加位移量，再加上变址寄存器的内容组成有效地址。

例如：MOV　EDX,[ESI][EBP＋200H]　　;有效地址为 EBP 的内容加 200H 加 EBX 的内容

(9) 带位移量的基址加带比例因子的变址寻址方式。

变址寄存器的内容乘以比例因子，再加上基址寄存器内容与位移量之和组成有效地址。

例如：MOV　ECX,[EDI＊2][EBX＋20]　　;有效地址为 EDI 的内容乘以 2，再加 EBX 的内容再加 20

3.4.2　80386 的指令系统

80386 的指令系统是在 8086 指令系统基础上设计的，并完全兼容 8086 指令系统，主要差别是 80386 指令系统扩展了数据宽度，对存储器寻址方式也进行了扩充，另外还增加了少量指令。本节仅对两者的差别和使用的注意点做简单的介绍，8086 已有的指令参见 3.3 节。

1. 数据传送类指令

数据传送指令包括通用数据传送指令、标志寄存器传送指令、地址传送指令、数据转换

指令等。

1) 通用数据传送指令

使用 MOV 指令时,两个操作数的位数必须相同,如果不同,则可以选用新增的 MOVZX 和 MOVSX 指令。例如:

```
MOVZX  AX,BL
```

该指令将 BL 的内容带符号扩展为一个字送入 AX。

另外,PUSH 指令的功能有所扩展,其源操作数可以是立即数。例如:

```
PUSH  0204H
```

而这样的指令在 8086 系统中不被允许。另外 80386 指令系统还提供了 PUSHA 指令,可以将全部 16 位寄存器一次压入堆栈,提供的 PUSHAD 指令可以全部 32 位寄存器一次压入堆栈。

2) 标志寄存器传送指令

在 LAHF、SAHF、PUSHF 和 POPF 的基础上,增加了两条指令,即

```
PUSHFD                          ;将标志寄存器的内容作为双字压入堆栈
POPFD                           ;从堆栈弹出双字送入标志寄存器
```

3) 地址传送指令

80386 的地址传送指令实现 6 字节的地址指针传送。地址指针来自存储单元,目的地址为两个寄存器,其中一个为段寄存器,另一个为双字的通用寄存器。例如:

```
LDS  EBX,DATA                   ;将 DATA 开始的指针送入 DS、EBX 寄存器
LSS  ESP,DATA                   ;将 DATA 开始的指针送入 SS、ESP 寄存器
```

这些指令适合 32 位微机系统的多任务操作,因为在任务切换时,需要同时改变段寄存器和偏移量指针的值。

4) 数据转换指令

在 8086 的 CBW、CWD 指令的基础上,增加了两条指令。即

```
CWDE                            ;将 AX 的内容扩展为双字送入 EAX
CDQ                             ;将 EAX 的内容扩展为四字送入 EDX、EAX
```

输入输出指令和 8086 完全相同,端口地址可以在指令中给出,也可以由 DX 寄存器给出。

2. 算术运算类指令

算术运算类指令的用法和 8086 中基本一致,只是在 80386 指令系统中,运算支持 32 位。例如:

```
ADD  EAX,0FF200A0H              ;将 EAX 内容加 0FF200A0H 送回 EAX 寄存器
SUB  EAX,EBX                    ;将 EAX 内容减去 EBX 内容送回 EAX 寄存器
```

在 80386 指令系统中,对 IMUL 指令给出两种扩充形式。

例如:`IMUL  DX,BX,100` ;将 BX 的内容乘以 100,结果送入 DX 寄存器

这类指令是用一个立即数乘以一个放在寄存器或存储器中的数,结果存入指定的寄存器。

例如:`IMUL  EDX,ECX` ;将 EDX 的内容乘以 ECX 内容,结果送入 EDX

这类指令是一个寄存器操作数乘以一个同样长度的放在寄存器或存储器的数，结果存入该寄存器。但由于这两类指令的乘积和被乘数、乘数的长度一样，有时会产生溢出，如果溢出，OF 标志则被自动置 1。

3. 逻辑运算类指令

逻辑运算类指令包括逻辑运算和移位指令。这两组指令的用法和 8086 中基本一致，只是在 80386 指令系统中，运算和移位支持 32 位。

```
例如：AND  EAX,EBX              ;将 EAX 内容与 EBX 内容相与,结果送回 EAX
      ROL  EBX,CL               ;按 CL 指定的次数将 EAX 内容循环左移
```

在 80386 指令系统中，还增加了两条专用的双精度移位指令，即双精度左移指令 SHLD 和双精度右移指令 SHRD。

```
例如：SHRD  EAX,EBX,10
```

这条指令将 EAX 的内容右移 10 位，高 10 位由 EBX 的低 10 位来补充，而 EBX 的内容不变。

4. 串操作指令

80386 的串操作指令和 8086 中基本一致，包括 MOVS、CMPS、SACS、LODS 和 STOS。此外，80386 指令系统还增加了字符串输入指令 INS 和字符串输出指令 OUTS，来处理输入输出端口的数据块的读写。

INS 指令可以从一个输入输出端口读入数据送到一串连续的存储单元；OUTS 指令可以将一串连续的存储单元的内容顺序输出到一个输入输出端口。

INS 指令使用时有 INSB、INSW 和 INSD 3 种形式，分别对应字节串、字串和双字串。OUTS 与之类似。

5. 转移、循环和调用指令

80386 的转移指令在形式和意义上和 8086 相同，唯一不同的是条件转移的地址不再受范围的限制，可以到达存储空间的任何地方。

循环指令 LOOP、LOOPE、LOOPNE 等与 8086 完全相同，转移范围仍然为－128～＋127。

调用指令 CALL 和返回指令 RET 在用法和含义上与 8086 类同，只是在 80386 中，由于 EIP 是 32 位，故在堆栈操作时，对于 EIP 的操作是 4 字节。

6. 条件设置指令

条件设置指令是 80386 中新增加的，用来根据测试标志的结果设置目的操作数的值。

```
例如：SETZ  AL              ;若 ZF = 1,则 AL 设置为 1,否则设置为 0
      CMP   AX,1000
      SETBE BX              ;若 AX 低于或等于 1000,则 BX 设置为 1,否则设置为 0
```

7. 系统设置和测试指令

这组指令也是 80386 中新增加的，它们一般出现在操作系统中，用来对系统设置和测试。具体如下。

(1) CLTS：清 TS 标志指令。

(2) SGDT/SLDT/SIDT。

这 3 条指令将全局描述符表寄存器、局部描述符表寄存器或中断描述符表寄存器的内容送到存储器。

例如：

```
SGDT        MEM1          ;其中 MEM1 为 6 字节存储器操作数
SLDT        MEM2          ;其中 MEM2 为 2 字节存储器操作数
SIDT        MEM3          ;其中 MEM3 为 6 字节存储器操作数
```

(3) LTR：装入任务寄存器指令。

这条指令一般用在多任务操作系统中，它将内存中 2 字节装入任务寄存器 TR，执行该指令后，相应的任务状态段 TSS 标上“忙”标志。

例如：`LTR  MEM2          ;其中 MEM2 为 2 字节存储器操作数`

(4) STR：存储任务寄存器指令。

这条指令一般用在多任务操作系统中，它将任务寄存器 TR 的内容送入内存中 2 字节。

例如：`STR  MEM2          ;其中 MEM2 为 2 字节存储器操作数`

(5) LAR：装入访问权指令。

本指令将 2 字节段选择子中的访问权字节送入目的寄存器。

例如：`LAR  AX,SELECT      ;将段选择子中的访问权字节送入 AH,AL 清 0`

(6) LSL：装入段界限值指令。

本指令将描述符中的段界限值送入目的寄存器。在指令中，由段选择子来指出段描述符。

例如：`LSL  BX,SELECT2     ;将 SELECT2 段选择子所对应的描述符中的段界限值送入 BX`

(7) LGDT/LLDT/LIDT。

这 3 条指令分别将存储器的内容送全局描述符表寄存器、局部描述符表寄存器或中断描述符表寄存器的内容。

```
例如：LGDT  MEM1          ;其中 MEM1 为 6 字节存储器操作数
      LLDT  MEM2          ;其中 MEM2 为 2 字节存储器操作数
      LIDT  MEM3          ;其中 MEM3 为 6 字节存储器操作数
```

(8) VERR/VERW：检测段类型指令。

VERR 指令检测段选择子所对应的段是否可读，VERW 指令则检测一个段选择子所对应的段是否可写。

```
例如：VERR  SELECT1       ;SELECT1 段选择子所对应的段是否可读
      VERW  SELECT2       ;SELECT2 段选择子所对应的段是否可写
```

(9) LMSW：装入机器状态字指令。

本指令将存储器中的 2 字节送入机器状态字 MSW。通过这种方式，可以使 CPU 切换到保护方式。检测段选择子所对应的段是否可读，VERW 指令则检测一个段选择子所对应的段是否可写。

例如：`VMSW  [SP]          ;将堆栈指针 SP 所指出的 2 字节送入 MSW`

(10) SMSW：存储机器状态字指令。

本指令将 2 字节机器状态字 MSW 存入内存中。

例如：`SMSW  MEM2          ;其中 MEM2 为 2 字节存储器操作数`

(11) ARPL：调整请求权级指令。

本指令可以调整段选择子的 RPL 字段，由此常用来阻止应用程序访问操作系统中涉及安全的高级别的子程序。ARPL 的第一个操作数可由存储器或寄存器给出，第二个操作数

必须是寄存器。如前者的 RPL(最后 2 位)小于后者的 RPL,则 ZF 置 1,且将前者的 RPL 增值,使其等于后者的 RPL;否则,ZF 清 0,并不改变前者的 RPL。

例如:`ARPL  MEM_WORD,BX`

3.5 Pentium 新增加的指令

Pentium 一共增加了 3 条处理器专用指令和 5 条系统控制指令,具体如下。

1. 处理器专用指令

1) CMPXCHG8B m:8 字节比较指令

本指令将 EDX:EAX 中的 8 字节与 m 所指的存储器中的 8 字节比较,若相等,则 ZF 置 1,并将 EDX:EAX 中的 8 字节送入目的存储单元;否则 ZF 清 0,并将目的存储单元中的 8 字节送入 EDX:EAX 中。

2) RDSTC:读时钟周期数指令

本指令读取 CPU 中用来记录时钟周期数的 64 位计数器的值,并将读取的值送入 EDX:EAX,供一些应用软件通过两次执行该指令来确定某段程序需要多少时钟周期。

3) CPUID:读 CPU 的标识等有关信息

本指令用来获得 Pentium 处理器的类型等有关信息。在执行此指令前,EAX 中如果为 0,则指令执行后,EAX、EBX、ECX、EDX 的内容合起来即为 Intel 产品的标识字符串,如果此前 EAX 中为 1,则指令执行后,在 EAX、EBX、ECX、EDX 中得到 CPU 级别、工作模式、可设置的断点数等。

2. 系统控制指令

(1) RDMSR:读取模式专用寄存器指令。

本指令用来读出 Pentium 处理器的模式专用寄存器中的值。在执行该指令前,在 ECX 中设置寄存器号,可以为 0～14H,指令执行后,读取的内容存放在 EDX:EAX 中。

(2) WRMSR:写入模式专用寄存器指令。

本指令将 EDX:EAX 中的 64 位数写入模式专用寄存器。同样在执行该指令前,必须在 ECX 中设置寄存器号,可以为 0～14H。

(3) RSM:复位到系统管理模式。

(4) MOV CR4,R32:将 32 位寄存器的内容送入控制寄存器 CR。

(5) MOV R32,CR4:将控制寄存器 CR 内容送入 32 位寄存器。