第 3 章　指令系统

每种 CPU 芯片都配置有相应的指令系统，供用户编程使用。本章从指令格式、寻址的概念着手，具体讨论 8086 系统中采用的寻址方式，分析 8086 指令系统中各类指令的功能、特点及应用，并引申到 Pentium 微处理器新增指令和寻址方式的特点。

通过本章的学习，重点理解和掌握以下内容:

- 指令格式及寻址的有关概念
- 8086 指令系统的寻址方式及其应用
- 8086 各类指令的表示、具体功能、特点及其应用
- Pentium 微处理器新增指令和寻址方式介绍

3.1　指令格式及寻址

计算机在解决计算或处理信息等问题时，需由人们事先把各类问题转换为计算机能识别和执行的操作命令。这种能被计算机执行的各种操作用命令形式写下来，就成为计算机指令。通常一条指令对应一种基本操作，如加减运算、数据传送和移位处理等。目前，一般小型或微型计算机系统可以包括几十种或百余种指令。

3.1.1　指令系统与指令格式

1. 指令系统的概念

计算机中的指令以二进制编码形式存放在存储器中，用二进制编码形式表示的指令称为机器指令，CPU 可直接识别机器指令。

对于使用者来说，机器指令的理解、记忆和阅读都比较困难，也容易出错，为此，人们采用一些助记符——通常是指令功能的英文单词的缩写，如数据传送指令用助记符 MOV（MOVE 的缩写）表示，这样表示的指令称为符号指令，也称为汇编指令。

汇编指令具有直观、易理解、好记忆的特点。在计算机中，汇编指令与机器指令具有一一对应的关系。

不同的 CPU 赋予的指令助记符不同，而且各自的指令系统中包含的操作类型也有不同。每种 CPU 指令系统的指令都有几十条、上百条之多。8086 指令系统对 Intel 公司后继机型具有很好的向上兼容性，用户编写的各种汇编语言源程序可以在其环境下运行。

指令系统是计算机系统结构中非常重要的组成部分。从计算机组成层次结构来说，计算机指令有机器指令、伪指令和宏指令之分。指令系统是计算机硬件和软件之间的桥梁，是汇编语言程序设计的基础。

2. 指令格式

计算机中的汇编指令由操作码字段和操作数字段两部分组成。

（1）操作码字段。操作码表示计算机要执行的某种指令功能，由它来规定指令的操作类型，说明计算机要执行的具体操作，例如传送、运算、移位、跳转等操作。同时还指出操作数的类型、操作数的传送方向、寄存器编码或符号扩展等，是指令中必不可少的组成部分。

计算机执行指令时，首先将操作码从指令队列取入执行部件中的控制单元，经指令译码器识别后，产生执行本指令操作所需的时序控制信号，控制计算机完成规定的操作。

（2）操作数字段。操作数表示计算机在操作中所需要的数据，或者所需数据的存放位置（也称为地址码），还可以是指向操作数的地址指针或其他有关操作数据的信息。

操作数字段可以有一个、二个或三个，分别称为单地址指令、双地址指令和三地址指令。单地址指令操作只需一个操作数，如加 1 指令“INC　AX”。大多数运算型指令都需两个操作数，如加法指令“ADD　AX,BX”中，AX 为被加数，BX 为加数，运算结果送到 AX 中，因此，AX 称为目的操作数，BX 称为源操作数。对于三地址指令则是在二地址指令的基础上再指定存放运算结果的地址。

计算机通过执行指令来处理各类信息，为了指出信息的来源、操作结果的去向以及所执行的操作，事先要规定好指令格式，每条指令中一般要包含操作码和操作数等字段。

8086CPU 的指令格式如图 3-1 所示，指令的长度范围是 1～6 个字节。其中，操作码字段为 1～2 个字节（B_1、B_2），操作数字段为 0～4 个字节（B_3～B_6）。每条具体指令的长度将根据指令的操作功能和操作数的形式而定。

B1	B2	B3	B4	B5	B6
OP code	OP code	low disp 或data	high disp 或data	low data	high data

图 3-1　8086CPU 的指令格式

3.1.2　操作数类别与寻址

计算机的指令中通常要指定操作数的位置，即给出操作数的地址信息，在执行时需要根据这个地址信息找到需要的操作数，这种寻找操作数的过程称为寻址。寻址方式就是指寻找操作数或操作数地址的方式。

不同机器的指令系统都规定了一些寻址方式以供编程时选择使用，根据给定的寻址方式，就可以方便地访问各类操作数。

一般来说，8086CPU 指令中的操作数有以下三种：

（1）立即操作数。操作数直接在指令中，即跟随在指令操作码之后，指令的操作数部分就是操作数本身。

（2）寄存器操作数。操作数存放在 CPU 的某个内部寄存器中，这时指令的操作数部分是 CPU 内部寄存器的一个编码。

（3）存储器操作数。操作数存放在内存储器的数据区中，这时指令的操作数部分包含此操作数所在的内存地址。

为了简便和易于理解，我们以 8086 为参考机型，分析 7 种基本的数据寻址方式，它们是：

立即数寻址、寄存器寻址、直接寻址、寄存器间接寻址、寄存器相对寻址、基址变址寻址、相对基址变址寻址方式。

此外，还有一种在指令中无操作数的固定寻址方式，即操作对象是固定的，在指令助记符中已经隐含了操作对象。如 DAA 指令，该指令中没有操作数，采用固定寻址方式，它对寄存器 AL 的内容进行操作，结果存于 AL 寄存器中。

3.2　8086 寻址方式及其应用

8086CPU 提供了与操作数有关的立即数寻址、寄存器寻址、直接寻址、寄存器间接寻址、寄存器相对寻址、基址变址寻址和相对基址变址寻址等 7 种寻址方式。此外，还规定了与输入/输出有关的端口地址寻址方式。

3.2.1　立即数寻址

立即数寻址方式是指操作数直接存放在给定的指令中，紧跟在操作码之后。立即数总是和操作码一起被取入 CPU 的指令队列，在指令执行时不需要访问存储器。

立即数可以是 8 位或 16 位二进制数。如果是 16 位操作数，则低位字节存放在低地址单元中，高位字节存放在高地址单元中。这种方式通常用于给寄存器或存储单元赋初值，该数可以是数值型常数，也可以是字符型常数。在指令格式中，它只能用于源操作数字段，不能用于目的操作数字段。

立即数可采用二进制数、十进制数以及十六进制数来表示。在非十进制的立即数末尾需要使用字母加以标识，一般情况下十进制数不需要加标识。

立即数寻址因操作数直接从指令中取得，不执行总线周期，所以该寻址方式的显著特点是执行速度快。

【例 3.1】分析以下立即数寻址方式的指令操作功能。

```
MOV  AL,25H              ;将十六进制数 25H 送 AL
MOV  AX,263AH            ;将十六进制数 263AH 送 AX
MOV  CL,50               ;将十进制数 50 送 CL
MOV  AL,10100110B        ;将二进制数 10100110B 送 AL
```

3.2.2　寄存器寻址

寄存器寻址方式是在指令中直接给出寄存器名，寄存器中的内容即为所需操作数。在寄存器寻址方式下，操作数存在于指令规定的 8 位、16 位寄存器中。寄存器可用来存放源操作数，也可用来存放目的操作数。

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

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

【例 3.2】分析以下寄存器寻址方式的指令操作功能。

```
MOV  AL,BL               ;将 BL 中保存的源操作数传送到目的操作数 AL
ADD  AX,BX               ;两个 16 位寄存器操作数相加，结果放在 AX
INC  CX                  ;对寄存器 CX 中的内容进行加 1 处理
```

寄存器寻址方式属于 CPU 内部的操作，不需要访问总线周期，因此指令的执行速度比较快。

3.2.3 存储器寻址

立即数寻址和寄存器寻址两种寻址方式中，操作数是从指令或寄存器中直接获得的，而在实际的程序运行中，大多数操作数需要从内存中获得。

用存储器寻址的指令，其操作数一般位于代码段之外的数据段、堆栈段或附加段的存储器中，指令中给出的是存储器单元的地址或产生存储器单元地址的信息。

执行这类指令时，CPU 先根据操作数字段提供的地址信息，由执行部件计算出有效地址 EA，再由总线接口部件根据公式计算出物理地址 PA，执行总线周期访问存储器取得操作数，最后再执行指令规定的基本操作。

注意：采用存储器寻址的指令中只能有一个存储器操作数，或者是源操作数，或者是目的操作数，且指令书写时将存储器操作数的地址放在方括号[]之中。

常见有以下 5 种存储器寻址方式。

1. 直接寻址方式

直接寻址方式是一种针对内存的寻址方式。该寻址方式下，指令中给出的地址码即为操作数的有效地址 EA，它是一个 8 位或 16 位的位移量。在默认方式下，操作数存放在数据段 DS 中，如果要对除 DS 段之外的其他段如 CS、ES、SS 中的数据寻址，应在指令中增加前缀，指出段寄存器名，这称为段跨越。

直接寻址方式的指令中，操作数的有效地址 EA 已经给出，则操作数的物理地址为：

PA=(DS)×10H+EA

【例 3.3】给定指令：MOV AX,[1002H]，当（DS）=2000H 时，操作数在内存中的存储形式如图 3-2 所示，计算操作数的物理地址并分析指令的执行结果。

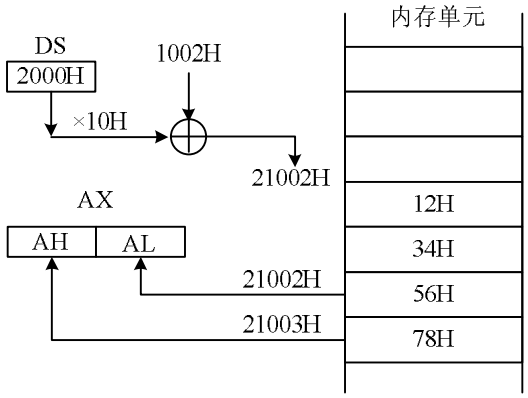


图 3-2 例 3.3 直接寻址分析

解：该指令为直接寻址，有效地址为 16 位，指令的操作数要占两个字节，其存储方式应该按“字”来存放。根据指令中给定的有效地址 EA=1002H 和物理地址的计算公式，可以得到该指令对应的操作数物理地址为：

PA=(DS)×10H+EA

=2000H×10H+1002H

=21002H

由于指令中目的操作数是 16 位寄存器，因此要取出 21002H 单元的字节数据 56H 送 AL 寄存器，21003H 单元的字节数据 78H 送 AH 寄存器。

指令执行后寄存器的内容为：(AX)=7856H。

8086 指令系统中规定，存储器直接寻址方式如果不加说明，操作数一定在数据段。

若操作数在规定段以外的其他段，则必须在地址前加以说明。

例如：MOV　AL,ES:[0002H]

该指令指明源操作数存放于附加段寄存器 ES 中，则物理地址 PA=(ES)×10H+EA。

2. 寄存器间接寻址方式

寄存器间接寻址方式是指操作数的有效地址 EA 在指定的寄存器中，这种寻址方式是在指令中给出寄存器，寄存器中的内容为操作数的有效地址。

16 位操作数寻址时，EA 放在基址寄存器 BX、BP 或变址寄存器 SI、DI 中，所以该方式下操作数的物理地址计算公式有以下几个：

PA=(DS)×10H+(BX)

PA=(DS)×10H+(DI)

PA=(DS)×10H+(SI)

PA=(SS)×10H+(BP)

前三个式子表示操作数在数据段，最后一个式子表示操作数在堆栈段。

【例 3.4】已知指令为：MOV　AX,[BX]，给定(DS)=2000H，(BX)=1000H，数据在内存中的存放位置如图 3-3 所示，计算该指令中操作数的物理地址并分析指令的执行结果。

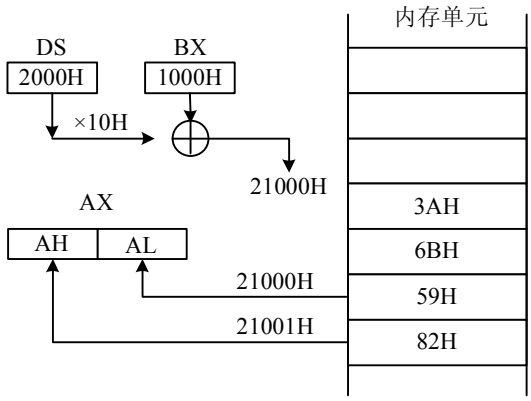


图 3-3　例 3.4 寄存器间接寻址分析

解：该指令采用寄存器间接寻址方式，指令中操作数的物理地址计算如下：

PA=(DS)×10H+(BX)

=2000H×10H+1000H

=21000H

由于目的操作数为 16 位寄存器，根据图 3-2 所示，指令执行时从内存 21000H 单元取出字节数据 59H 送 AL，从 21001H 单元取出字节数据 82H 送 AH，最后组合成 1 个字数据 8259H 传送到寄存器 AX 中。

即指令执行后的结果为：(AX)=8259H。

在该寻址方式下，当指令中指定的寄存器为 BX、SI、DI 时，操作数存放在数据段中，则段地址是数据段寄存器 DS 中的内容；若指令中指定的寄存器为 BP 时，操作数存放在堆栈段中，段地址是堆栈段寄存器 SS 中的内容；若指令中指定了跨越前缀，则可以从指定的段中获得操作数。

执行直接寻址时，操作数的有效地址 EA 在指令中，它是一个常量；执行间接寻址时，操作数的有效地址 EA 在寄存器中，寄存器的内容由它之前的指令确定，因而是一个变量。

3. 寄存器相对寻址方式

这种寻址方式是在指令中给定一个基址寄存器或变址寄存器和一个 8 位或 16 位的相对偏移量，两者之和作为操作数的有效地址。当选择间址寄存器 BX、SI、DI 时，指示的是数据段中的数据，选择 BP 作间址寄存器时，指示的是堆栈段中的数据。

有效地址为：EA=(reg)+8 位或 16 位偏移量；其中 reg 为给定寄存器。

物理地址为：PA=(DS)×10H+EA　　（使用 BX、SI、DI 间址寄存器）

PA=(SS)×10H+EA　　（使用 BP 作为间址寄存器）

【例 3.5】已知指令 MOV AX,[BX+0010H]，给定寄存器内容为(BX)=1200H，(DS)=2000H，存储单元内容为(21210H)=34H，(21211H)=12H；分析指令执行后的结果和计算操作数的有效地址及物理地址。

解：该题的功能是传送指定地址中的字数据到累加器中，属于寄存器相对寻址方式。

则：操作数的有效地址 EA=(BX)+0010H

=1210H

操作数的物理地址 PA=(DS)×10H+EA

=21210H

指令执行后，从内存 21210H 单元取出字节数据 34H 送 AL 寄存器，从内存 21211H 单元中取出字节数据 12H 送寄存器 AH。

最后的结果为：(AX)=1234H。

4. 基址变址寻址方式

在基址变址寻址方式中，有效地址 EA 是基址寄存器加变址寄存器，即两个寄存器的内容之和为操作数的有效地址。在该寻址方式中，当基址寄存器和变址寄存器的默认段寄存器不同时，一般由基址寄存器来决定默认用哪一个段寄存器作为段基址指针。若在指令中规定了段跨越，则可以用其他寄存器作为段基地址。

基址变址寻址方式的物理地址计算公式为：

物理地址 PA=(DS)×10H+(BX)+(SI)

物理地址 PA=(SS)×10H+(BP)+(DI)

【例 3.6】给定指令：MOV AX,[BX+SI]，寄存器保存的内容为(DS)=2000H，(BX)=1000H，(SI)=0050H，内存中的数据存放如图 3-4 所示，计算操作数地址并分析指令的执行情况。

解：该题为基址变址寻址方式，将指定的内存单元内容传送到寄存器 AX 中。

操作数的物理地址为：

PA=(DS)×10H+(BX)+(SI)

=2000H×10H+1000H+0050H

=21050H

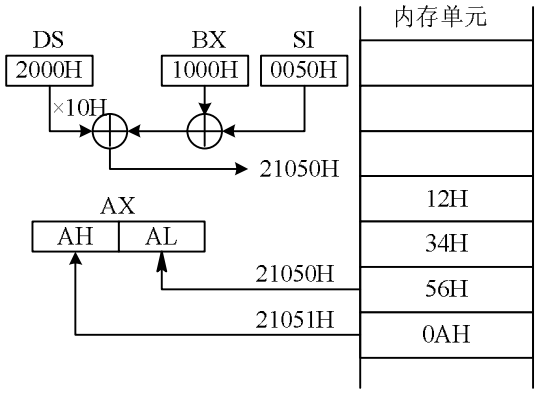


图 3-4　例 3.6 基址变址寻址分析

指令的执行结果是将内存 21050H 单元的数据传送到寄存器 AL 中，将内存 21051H 单元的数据传送到寄存器 AH 中。

指令执行完毕后：(AX)=0A56H

在基址变址寻址方式中，也可以使用段跨越前缀标识操作数所在的段。

如：MOV　AX,ES:[BX+DI]

则物理地址计算为：PA= (ES)×10H+(BX)+(DI)

在基址变址寻址方式中，基址寄存器可取 BX 或 BP，变址寄存器可取 SI 或 DI，但指令中不能同时出现两个基址寄存器或两个变址寄存器。

若基址寄存器为 BX，则段寄存器使用 DS；若基址寄存器用 BP，则段寄存器使用 SS。

5. 相对基址变址寻址方式

这种寻址方式是在指令中给出一个基址寄存器、一个变址寄存器和 8 位或 16 位的偏移量，三者之和作为操作数的有效地址。

基址寄存器可取 BX 或 BP，变址寄存器可取 SI 或 DI。

若基址寄存器采用 BX，则段寄存器使用 DS；

若基址寄存器采用 BP，则段寄存器使用 SS。

其物理地址为：PA=(DS)×10H+(BX)+(SI)或(DI)+偏移量

PA=(SS)×10H+(BP)+(SI)或(DI)+偏移量

【例 3.7】给定指令：MOV　AX,[1100H+BX+SI]，已知寄存器内容(DS)=2000H，(BX)=0100H，(SI)=0002H，存储单元内容(21202H)=B5H，(21203H)=37H，试分析指令执行后，AX 寄存器中的内容。

解：该指令为相对基址变址寻址方式，采用基址寄存器 BX 和变址寄存器 SI，给定的 16 位的偏移量为 1100H，则操作数的物理地址计算如下：

PA=(DS)×10H+(BX)+(SI)+偏移量

=2000H×10H+0100H+0002H+1100H

=21202H

指令的执行结果是将内存 21202H 单元的字节数据 B5H 传送到寄存器 AL 中，将内存 21203H 单元的字节数据 37H 传送到寄存器 AH 中。

最后的执行结果为：(AX)=37B5H

3.2.4 I/O 端口寻址

由于 8086CPU 的 I/O 端口采用独立编址方式，可有 64K 个字节端口或 32K 个字端口。指令系统中设有专门的输入指令 IN 和输出指令 OUT 来进行访问。

I/O 端口的寻址方式有直接端口寻址和寄存器间接端口寻址两种。

1. 直接端口寻址

直接端口寻址是在指令中直接给出要访问的端口地址，一般采用 2 位十六进制数表示，可访问的端口数为 0～255 个。

例如：

```
IN  AL，30H  ;表示从 I/O 端口地址为 30H 的端口中取出字节数据送到 8 位寄存器 AL 中
IN  AX，50H  ;表示从 I/O 端口地址为 50H 和 51H 的两个相邻端口中取出字数据送到 16 位寄
             存器 AX 中
```

OUT 指令和 IN 指令一样，提供了字节或字两种使用方式，由端口的宽度来决定，当端口宽度只有 8 位时，只能用字节指令。

直接端口地址也可以用符号地址表示，例如：

```
OUT  PORT，AL    ;通过符号地址 PORT 表示的端口进行字节输出
OUT  PORT，AX    ;通过符号地址 PORT 表示的端口进行字输出
```

2. 寄存器间接端口寻址

当访问的端口地址数≥256 时，直接端口寻址不能满足要求，要采用 I/O 端口的间接寻址方式。它是把 I/O 端口的地址先送到寄存器 DX 中，用 16 位的 DX 作为间接寻址寄存器。此种方式可访问的端口数为 0～65535 个。

例如：

```
MOV  DX，283H     ;将端口地址 283H 送到 DX 寄存器
OUT  DX，AL       ;将 AL 中的内容输出到 DX 所指定的端口中
```

又如：

```
MOV  DX，280H     ;将端口地址 280H 送到 DX 寄存器中
IN  AX，DX        ;从(DX)和(DX)+1 所指的两个端口输入一个字，低地址端口输入到 AL，
                  高地址端口输入到 AH
```

3.3 8086 指令系统

8086 指令系统是 80X86/Pentium 微处理器的基本指令集。指令的操作数可以是 8 位或 16 位，偏移地址是 16 位。按功能可将指令分成数据传送、算术运算、逻辑运算与移位、串操作、控制转移和处理器控制等六大类指令。

本节在介绍指令的基本情况基础上，着重于对指令功能的理解和应用。在学习指令时，要注意掌握各类指令的助记符、书写格式、操作功能、寻址方式、指令对标志位的影响等方面，再通过实验操作来全面而准确地理解每条指令的功能和用法，为编制汇编语言源程序打下牢固基础。

3.3.1 数据传送类指令

数据传送类指令的基本功能是把操作数或操作数的地址传送到指定的寄存器或存储单

元中。

数据传送类指令共有 14 条，根据传送的内容可分成以下 4 组：

（1）通用数据传送指令。

（2）累加器专用传送指令。

（3）地址传送指令。

（4）标志寄存器传送指令。

数据传送类指令是计算机中最基本、最常用、最重要的一类操作。可用于在寄存器与存储器、寄存器与寄存器、累加器与 I/O 端口之间传送数据、地址等信息，也可以将立即数传送到寄存器或存储器中。为此，指令中必须指明数据起始存放的源地址和数据传送的目标地址。源操作数可以是累加器、寄存器、存储器操作数和立即数；而目的操作数可以是累加器、寄存器和存储器。

1. 通用数据传送指令

（1）传送指令 MOV。

格式：MOV　dst,src

MOV 指令的功能是把源操作数 src 传送至目的操作数 dst，执行后源操作数内容不变，目的操作数内容与源操作数内容相同。

源操作数可以是通用寄存器、段寄存器、存储器以及立即数，目标操作数可以是通用寄存器、段寄存器（CS 除外）或存储器。

采用 MOV 指令时，各类数据之间的传送关系如图 3-5 所示。

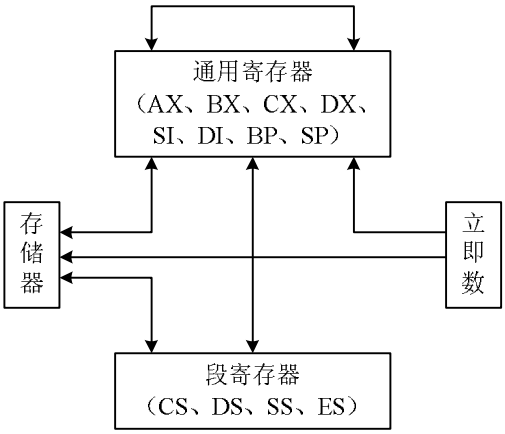


图 3-5　数据之间的传送关系

使用 MOV 指令进行数据传送时要注意以下几点：

- 段寄存器 CS 及立即数不能作为目标操作数。
- 两个存储单元之间不允许直接传送数据。
- 立即数不能直接传送到段寄存器。
- 两个段寄存器之间不能直接传送数据。
- 传送数据的类型必须匹配。
- MOV 指令不影响标志位。

【例 3.8】给定如下 MOV 指令的形式，分析其操作功能。

```
MOV  AL,34H              ;8 位立即数 34H 送 AL 寄存器
MOV  BL,'A'              ;字符'A'的 ASCII 码送 BL 寄存器
MOV  SI,COUNT            ;COUNT 为一个符号常数，其值送 SI 寄存器
MOV  DX,2175H            ;16 位立即数 2175H 送 DX 寄存器
MOV  AX,BX               ;16 位寄存器 BX 内容传送到累加器 AX
MOV  DH,BH               ;8 位寄存器 BH 内容传送到 DH
MOV  AX,[2365H]          ;将指定存储单元 2365H 中的数据传送到 AX 寄存器中
MOV  [3200H],SI          ;将 SI 寄存器中的内容传送到指定的存储单元 3200H 中
MOV  ARRAY,21H           ;将立即数 21H 送指定符号地址的内存单元 ARRAY
MOV  DS,AX               ;将累加器 AX 中的内容传送到数据段寄存器 DS 中
MOV  [SI],DS             ;将 DS 中的内容传送到 SI 所指示的字单元中
MOV  ES,[BX]             ;将 BX 所指示的存储单元中的内容传送到 ES 中
```

（2）堆栈操作指令 PUSH/POP。

进栈指令：PUSH　opr　　;SP←SP-2，将源操作数 opr 压入堆栈

出栈指令：POP　 opr　　;栈顶弹出字数据到目标操作数 opr，SP←SP+2

前面已经讨论过，堆栈是存储器中的一个特殊区域，主要用于存入和取出数据，堆栈是以“先进后出”的工作方式进行数据操作的。在 8086 堆栈组织中，堆栈从高地址向低地址方向生长，它只有一个出入口，堆栈指针寄存器 SP 始终指向堆栈的栈顶单元。

堆栈操作时栈顶的位置将发生变化，即堆栈指针寄存器 SP 的内容会被修改，并始终指向的是栈顶位置。操作时按照字数据进行，即每次入栈或出栈都按 2 个字节单元来处理，堆栈的示意如图 3-6 所示。

执行进栈指令 PUSH 时，使 SP←SP-2，然后将 16 位源操作数压入堆栈，先高位后低位。源操作数可以是通用寄存器、段寄存器和存储器。

操作过程是：首先 SP 内容减 1，将操作数的高位字节送入当前 SP 所指示的单元中；然后 SP 中的内容再减 1，将操作数的低位字节送入当前 SP 所指示的单元中。

出栈指令 POP 的执行过程与 PUSH 相反，它从当前栈顶弹出 16 位操作数到目标操作数，同时 SP←SP+2，使 SP 指向新的栈顶。目标操作数可以是通用寄存器、段寄存器（CS 除外）或存储器。

操作过程是：首先将 SP 所指示的栈顶单元内容送入操作数低位字节单元，SP 的内容加 1，然后将 SP 所指栈顶单元内容送入操作数的高位字节单元，SP 的内容再加 1。

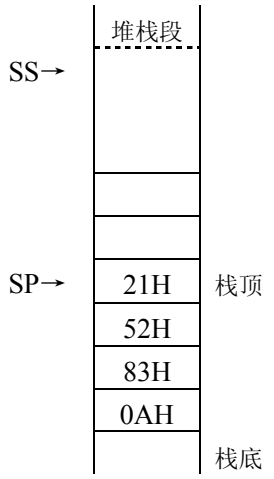


图 3-6　堆栈示意图

【例 3.9】给定堆栈操作指令和环境如下，分析其操作功能。

1）已知(SP)=00F8H，(SS)=2000H，(AX)=3485H，执行指令 PUSH　AX。

2）已知(SS)=2000H，(SP)=0100H，(BX)=1234H，(20100H)=53H，(20101H)=2AH，执行指令 POP　BX。

解：1）执行进栈指令 PUSH　AX。

堆栈指针变化为(SP)←(SP)-2

即(SP)=00F8H-2=00F6H

目的操作数的物理地址为：PA=(SS)×10H+(SP)

=2000H×10H+00F6H

=200F6H

该指令将累加器 AX 中的字数据 3485H 压入堆栈区域 200F6H 单元开始的字存储区，执行后，(200F6H)=85H，(200F7H)=34H。

2）执行出栈指令 POP　BX。

堆栈区数据的物理地址为：PA=(SS)×10H+(SP)

=2000H×10H+0100H

=20100H

该指令将堆栈区域 20100H 单元开始的字存储区数据 2A53H 弹出到寄存器 BX 中，即(BX)=2A53H。此时堆栈指针变化为 SP←SP+2，即(SP)=0100H+2=0102H。

堆栈在子程序调用或中断处理时常用于保护当前的断点地址和现场数据，以便子程序执行完毕后正确返回到主程序。

堆栈操作时，PUSH 和 POP 指令不影响标志位。

（3）交换指令 XCHG。

XCHG 指令用来将源操作数和目的操作数的内容进行交换。它可以实现字节数据交换，也可以实现字数据交换。

该指令的操作数必须有一个是在寄存器中，即可以在两个通用寄存器之间或寄存器与存储器之间交换数据，但不能在两个存储器之间交换数据。指令执行结果不影响标志位。

例如：若给定(AX)=1234H，(BX)=5678H，执行指令：XCHG　AX,BX，该指令将寄存器 AX 的内容与寄存器 BX 的内容互相交换位置，则指令执行后：(AX)=5678H，(BX)=1234H。

2. 累加器专用传送指令

8086 指令系统中将累加器 AX 作为数据传输的核心，系统的输入/输出指令 IN/OUT 和换码指令 XLAT 就是专门通过累加器来执行的，称之为累加器专用传送指令。

下面分析相关指令的功能。

输入指令：IN　Acc,src　　;Acc 为 8 位或 16 位累加器

输出指令：OUT　dst,Acc

使用 I/O 指令时需注意：

（1）IN 和 OUT 指令只能用累加器进行输入和输出数据，不能采用其他寄存器。

（2）直接端口寻址的 I/O 指令端口范围为 0～FFH，在一些规模较小的微机系统已经够用了。

（3）在一些功能较强的微机系统会使用大于 FFH 的端口数，这时需通过 DX 采用寄存器间接端口寻址方式。

例如，将 12 位 A/D 转换器所得数字量输入。A/D 转换器使用一个字端口，地址设为 2F0H。输入数据程序段为：

```
MOV  DX，02F0H
IN    AX，DX
```

换码指令：XLAT

XLAT 指令的功能是根据累加器 AL 中的一个值去查内存表格，将查到某一个值送回 AL 中。

使用 XLAT 指令之前，要先将表格的首地址送入 BX 寄存器，将待查的值放入 AL 中，用

它来表示表中某一项与表首址的距离。执行时，BX 和 AL 的内容相加得到一个地址，再将该地址单元的值取到 AL 中，即为查表转换的结果。换码指令通常用于无规律代码之间的转换，又称为查表转换指令。

【例 3.10】给定累加器专用传送指令如下，分析其操作功能。

```
IN   AL,20H          ;将 20H 端口中的数据读入到 AL 中
IN   AX,25H          ;将 25H 端口数据读入到 AL 中，将 26H 端口数据读入到 AH 中
OUT  DX,AL           ;AL 中的内容输出到 DX 所指示的字节端口
XLAT                 ;执行操作 AL←(BX+AL)
```

3. 地址传送指令

8086 的地址传送指令用于控制寻址机构，它可将存储器操作数的地址传送到 16 位目标寄存器中。

这类指令有以下 3 种形式。

（1）有效地址送寄存器指令：LEA　reg,src。

LEA 指令功能是将存储器操作数 src 的有效地址传送到 16 位的通用寄存器 reg。

例如：LEA　BX,[SI+BP]

该指令的功能是将(SI)+(BP)的值送到 BX 中。

（2）地址指针送寄存器和 DS 指令：LDS　reg,src。

该指令完成一个 32 位的地址指针传送，地址指针包括段地址和偏移地址两部分。执行的操作是将存储器操作数 src 指定的 4 个字节地址指针传送到两个目标寄存器，其中，地址指针的前两个字节单元的内容送入指令所指定的 16 位通用寄存器中，src＋2 所指示的两个字节单元的内容送入寄存器 DS 中。

【例 3.11】若给定(SI)=0010H，(DS)=2000H，(BX)=3572H，(20130H)=80H，(20131H)=12H，(20132H)=42H，(20133H)=21H。

执行指令：LDS　BX,0120H[SI]，分析其操作功能。

解：该指令执行后，由指令的寻址方式计算出操作数的起始物理地址为：

PA=(DS)×10H+(SI)+0120H

=2000H×10H+0010H+0120H

=20130H

指令从 20130H 单元开始取出 4 个字节的内容，分别送入规定的寄存器中。

执行结果为：(BX)=1280H，(DS)=2142H。

（3）地址指针送寄存器和 ES 指令：LES　reg,src。

LES 指令执行的操作与 LDS 指令相似，不同之处是以 ES 代替 DS。

【例 3.12】若给定(DS)=2000H，(BX)=0010H，(ES)=4000H，(20010H)=25H，(20011H)=31H，(20012H)=00H，(20013H)=25H。

执行指令：LES　DI,[BX]，分析其操作功能。

解：该指令执行后，由指令的寻址方式计算出操作数的起始物理地址为：

PA=(DS)×10H+(BX)

=2000H×10H+0010H

=20010H

指令从 20010H 单元开始取出 4 个字节的内容，分别送入规定的寄存器中。

则指令执行后：(DI)=3125H，(ES)=2500H。

4. 标志寄存器传送指令

8086 可通过这类指令读出当前标志寄存器中各标志位的内容，也可以重新设置各标志位的值。标志寄存器的传送指令共有 4 条，均为单字节指令，指令的操作数以隐含形式出现，隐含为 AH 寄存器。

（1）取标志指令 LAHF。

该指令将 16 位标志寄存器 FLAG 中的低 8 位取到 AH 中，即将 SF、ZF、AF、PF、CF 这 5 个状态标志位分别取出传送到 AH 的对应位，如图 3-7 所示。

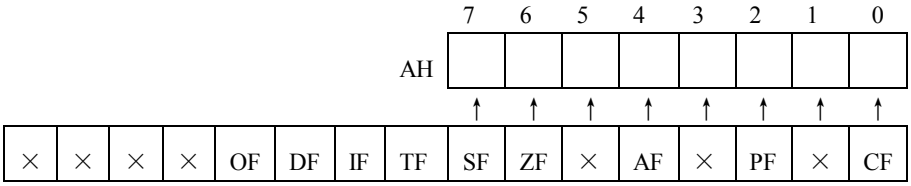


图3-7　LAHF指令执行过程示意图

（2）置标志位指令 SAHF。

SAHF 指令的操作与 LAHF 正好相反，它是将 AH 寄存器中的内容分别传送到标志寄存器的低 8 位。FLAG 中的 SF、ZF、AF、PF 和 CF 将被修改成 AH 寄存器所对应位的值，但 OF、DF、IF 和 TF 等 4 个标志位不受影响。

（3）标志寄存器入栈指令 PUSHF。

PUSHF 指令先将堆栈指针 SP 减 2，然后将 16 位 FLAG 中的内容压入堆栈中。指令执行后，标志寄存器的内容不变。

（4）标志寄存器出栈指令 POPF。

POPF 指令的操作与 PUSHF 指令正好相反，它将堆栈顶部的一个字弹出到标志寄存器 FLAG，然后修改堆栈指针 SP 加 2。该指令执行后，将改变标志寄存器的内容。

置标志位指令 SAHF 和标志寄存器出栈指令 POPF 指令对状态标志位有影响，其他两条指令执行后对标志位没有影响。

PUSHF 和 POPF 指令一般用在子程序和中断服务程序中，可保护或恢复主程序的标志位的值。

为方便使用，我们将数据传送类指令的格式和功能归纳如表 3-1 所示。表中 dst 表示目的操作数，src 表示源操作数。

表 3-1　数据传送类指令的格式及功能

指令类型	指令格式	指令功能
通用数据传送	MOV　dst,src	字节或字传送
	PUSH　src	字压入堆栈
	POP　dst	字弹出堆栈
	XCHG　dst,src	字节或字交换

续表

指令类型	指令格式	指令功能
累加器专用传送	IN　Acc,src	从指定端口将数据送累加器
	OUT　dst,Acc	将累加器中的数据送指定端口
	XLAT	换码指令
地址传送	LEA　dst,src	装入有效地址
	LDS　dst,src	装入 DS 寄存器
	LES　dst,src	装入 ES 寄存器
标志寄存器传送	LAHF	将 FLAG 低字节装入 AH 寄存器
	SAHF	将 AH 内容装入 FLAG 低字节
	PUSHF	将 FLAG 内容压栈
	POPF	从堆栈中弹出一个字给 FLAG

3.3.2　算术运算类指令

8086 的算术运算类指令包括加、减、乘、除 4 种基本运算指令，以及为进行 BCD 码十进制数运算而设置的各种较正指令。

8086 的基本算术运算指令中，除加 1 和减 1 指令外，其余均为双操作数指令，两个操作数中除了源操作数可为立即数外，必须有一个操作数在寄存器中，而单操作数指令则不允许采用立即数方式。

算术运算指令涉及的操作数从数据形式来分有 8 位和 16 位的操作数两种，这些操作数从类型来分又有无符号数和带符号数两种类型数据。在进行加减运算时可采取同一套指令，而乘除运算则各自有不同的指令。

加减法运算在执行过程中会产生溢出，无符号数运算时，如果加法运算最高位向前产生进位或减法运算最高位向前有借位，则表示出现溢出，采用标志位 CF=1 来表示；带符号数采用补码运算时，符号位也参与运算，出现溢出则表示运算结果发生了错误，采用标志位 OF=1 来表示。

算术运算指令除加 1 指令 INC 不影响 CF 标志外，其余指令对 CF、OF、ZF、SF、PF、AF 等 6 个标志位均可产生影响，其规则如下：

- 无符号数运算产生溢出时，CF=1。
- 带符号数运算产生溢出时，OF=1。
- 当运算结果为 0 时，ZF=1。
- 当运算结果为负数时，SF=1。
- 当运算结果中有偶数个 1 时，PF=1。
- 当操作数为 BCD 码，低 4 位出现进位 1 时，AF=1。

下面分别介绍 6 种基本算术运算指令的功能和应用特点。

1. 加法指令

（1）不带进位加法指令：ADD　dst,src。

指令功能为：(dst)←(dst)+(src)

使用时要注意两个操作数类型保持一致，而且不能对两个存储器操作数直接相加，段寄存器不能参加运算。

ADD 指令可采用的格式如下：

```
ADD  AL,BL                    ;两个寄存器的字节数据相加
ADD  AL,[0123H]               ;内存单元与寄存器的字节数据相加
ADD  [SI],AX                  ;寄存器与内存单元的字数据相加
ADD  BYTE  PTR[SI],24H        ;立即数与内存单元的字节数据相加
```

（2）带进位的加法指令：ADC　dst,src。

指令功能为：(dst)←(dst)+(src)+CF

与 ADD 指令不同的是要加进位标志 CF 的值， ADC 指令主要用于多字节或多字的加法运算中。

（3）加 1 指令：INC　opr。

指令功能为：(opr)←(opr)+1

opr 只能为通用寄存器或存储器，不能对立即数或段寄存器加 1。该指令也叫增量指令，通常在循环程序中用作循环计数器。

INC 指令的用法如下：

```
INC  AL                       ;对 8 位寄存器 AL 中的内容加 1
INC  CX                       ;对 16 位寄存器 CX 中的内容加 1
INC  BYTE  PTR  [SI][BX]      ;对指定内存的字节单元的内容加 1
```

2. 减法指令

（1）不带借位的减法指令：SUB　dst,src。

指令功能为：(dst)←(dst)−(src)

例如：

```
SUB  AX,BX        ;将 AX 内容减去 BX 内容，结果送到 AX 中
```

（2）带借位的减法指令：SBB　dst,src。

指令功能为：(dst)←(dst)−(src)−CF

与 SUB 不同之处在于还要减去借位 CF 的值，SBB 指令通常用于多精度数的减法运算。

例如：

```
SBB  DX,CX        ;执行 (DX)-(CX)-CF，结果送到 DX 中
```

（3）减 1 指令：DEC　opr

指令功能为：(opr)←(opr)−1

此指令通常用于循环程序中修改指针和循环次数。

DEC 指令使用形式如下：

```
DEC  CX                   ;将 CX 中的内容减 1 后结果再送回 CX
DEC  BYTE  PTR[SI]        ;将 SI 所指示字节单元中的内容减 1 后结果送回该单元
```

（4）求补指令：NEG　opr。

该指令将 opr 中的内容取 2 的补码，相当于将 opr 中的内容按位取反后末位加 1。

（5）比较指令：CMP　opr1,opr2。

指令功能为：(opr1)−(opr2)

该指令与 SUB 指令一样进行减法操作，但不保存结果，指令执行后两个操作数的内容不会改变。CMP 指令通常根据操作的结果来设置标志位，按比较结果使程序产生条件转移。

3. 乘法运算指令

乘法指令包括无符号数和带符号数相乘的指令，指令中只给出乘数，被乘数隐含给出。两个 8 位数相乘时被乘数放入 AL 中，16 位数的乘积存放到 AX 中；两个 16 位数相乘时被乘数先放入 AX 寄存器中，32 位数的乘积放到 DX 和 AX 两个寄存器中，规定 DX 中存放高 16 位，AX 中存放低 16 位。

（1）无符号数乘法指令：MUL　src。

若 src 为字节数据，执行 AX←(AL)×(src)；

若 src 为字数据，执行 DX、AX←(AX)×(src)。

src 可采用寄存器和存储器，但不能使用立即数和段寄存器。

指令使用形式如下：

```
MUL  AL          ;完成(AL)×(AL)操作，结果送 AX
MUL  BX          ;完成(AX)×(BX)操作，结果分别送 DX 和 AX
```

（2）带符号数乘法指令：IMUL　src。

该指令的执行功能与 MUL 相同，此处不再重复。

4. 除法运算指令

除法指令可用来实现两个无符号数或带符号数的除法运算，包括字和字节两种操作，该指令隐含使用 AX 和 DX 作为一个操作数，指令中给出的源操作数为除数。

（1）无符号数除法指令：DIV　src。

DIV 指令的被除数、除数、商和余数全部为无符号数。

（2）带符号数除法指令：IDIV　src。

IDIV 指令的被除数、除数、商和余数均为带符号数，且余数的符号位同被除数。

两条指令执行的操作功能如下：

当除数 src 为字节数据时，用 AX 除以 src，得到的 8 位商保存在 AL 中，8 位余数保存在 AH 中。

当除数 src 为字数据时，用 DX、AX 除以 src，得到的 16 位商保存在 AX 中，16 位余数保存在 DX 中。

这里需要注意的是，DIV 指令在除数为 0，或者字节操作时其商超过 8 位，字操作时其商超过 16 位，会产生除法溢出；同样，IDIV 指令在除数为 0，或者字节操作时其商超过-128～+127 范围，字操作时其商超过-32727～+32728 范围，也会产生除法溢出。

5. 符号扩展指令

符号扩展指令是指用一个操作数的符号位形成另一个操作数，后一个操作数的各位是全 0（正数）或全 1（负数），符号扩展指令虽然使数据位数加长，但数据的大小并没有改变。该指令的执行不影响标志位。

（1）字节转换为字指令 CBW。

该指令的功能是将 AL 中的符号位 D_7 扩展到 AH 中。

如果 AL 中的最高位 D_7=0，则转换后(AH)=00H；如果 AL 中的 D_7=1，则转换后(AH)=FFH，AL 的值不变。

（2）字转换为双字指令 CWD。

该指令的功能是将 AX 中的符号位扩展到 DX 中。

如果 AX 中的最高位 D_{15}=0，则转换后(DX)=0000H，如果 AX 中的 D_{15}=1，则转换后(DX)=FFFFH。

符号扩展指令常用来获得带符号数的倍长数据，而无符号数通常采用直接使高 8 位或高 16 位清 0 的方法来获得倍长数据。

6. 十进制调整指令

十进制数在计算机中是采用二进制数来表示的，这就是 BCD 码，要对十进制的 BCD 码进行算术运算，必须对得到的结果进行调整，否则结果无意义。

8086 指令系统提供了以下两类十进制调整指令。

（1）组合 BCD 码加法、减法调整指令。

```
DAA   ;组合 BCD 码加法调整指令，将 AL 中的和调整为组合 BCD 码
DAS   ;组合 BCD 码减法调整指令，将 AL 中的差调整为组合 BCD 码
```

DAA 和 DAS 分别用于加法指令（ADD、ADC）或减法指令（SUB、SBB）之后，执行时先对 AL 中保存的结果进行测试，若结果中的低 4 位＞09H，或者标志位 AF=1，则进行 AL←(AL)±06H 修正；如果 AL 寄存器中所保存结果的高 4 位＞09H，或者标志位 CF=1，则进行 AL←(AL)±60H 修正。该指令会影响 OF 以外的其他状态标志位。

【例 3.13】给定寄存器的保存内容为：(AL)=28H，(BL)=69H。

执行指令：

```
ADD  AL,BL
DAA
```

分析指令的操作结果。

解：执行 ADD 指令后，(AL)=91H，AF=1。可见，AL 中保存的结果不符合组合 BCD 码要求，出现了误差。

执行 DAA 调整指令时，由于 AF=1，要作 AL←(AL)+06H 的调整操作，调整后 AL 中的内容为 97H，符合要求；而 AL 的高 4 位≤09H，则不必进行调整。

【例 3.14】给定寄存器的保存内容为：(AL)=97H，(AH)=39H

执行指令：

```
SUB  AL,AH
DAS
```

分析指令的操作结果。

解：执行 SUB 指令后，(AL)=5EH，AF=1，可见，AL 中的内容不是组合 BCD 码格式，需要对其进行调整。

执行 DAS 指令，完成 AL←(AL)-06H 后，(AL)=58H，CF=0 且 AL 中高 4 位≤09H，不必进行调整。

（2）非组合 BCD 加法、减法调整指令。

```
AAA  ;非组合 BCD 加法调整指令，将 AL 中的和调整为非组合 BCD 码
AAS  ;非组合 BCD 减法调整指令，将 AL 中的差调整为非组合 BCD 码
```

AAA 和 AAS 分别用于加法指令（ADD、ADC）或减法指令（SUB、SBB）之后，执行时对 AL 进行测试，若 AL 中的低 4 位＞09H，或 AF=1，则进行 AL←(AL)±06H 修正；AL 的高 4 位为 0，同时 AH←(AH)±1；CF=AF=1。调整后的结果放在 AX 中。

【例 3.15】给定寄存器的保存内容为：(AX)=0505H，(BL)=09H。

执行指令：

```
ADD  AL,BL
AAA
```

分析指令的执行结果。

解：执行 ADD 指令后，(AL)=0EH，可见 AL 的低 4 位＞09H，该结果不是非组合 BCD 码，要进行调整。

执行 AAA 指令，进行 AL←(AL)+06H 的修正，最终结果为(AX)=0104H，是 14 的非组合 BCD 码表示。

各类算术运算指令的格式、功能及对标志位的影响归纳如表 3-2 所示。

表 3-2　算术运算指令格式、功能及对标志位的影响

类别	指令书写格式（助记符）	指令名称	状态标志					
			OF	SF	ZF	AF	PF	CF
加法	ADD dst,src	加法（字节/字）	¤	¤	¤	¤	¤	¤
	ADC dst,src	带进位加法（字节/字）	¤	¤	¤	¤	¤	¤
	INC dst	加 1（字节/字）	¤	¤	¤	¤	¤	—
减法	SUB dst,src	减法（字节/字）	¤	¤	¤	¤	¤	¤
	SBB dst,src	带借位减法（字节/字）	¤	¤	¤	¤	¤	¤
	DEC dst	减 1	¤	¤	¤	¤	¤	—
	NEC dst	求补	¤	¤	¤	¤	¤	¤
	CMP dst,src	比较	¤	¤	¤	¤	¤	¤
乘法	MUL src	不带符号乘法（字节/字）	¤	※	※	※	※	¤
	IMUL src	带符号乘法（字节/字）	¤	※	※	※	※	¤
除法	DIV src	不带符号除法（字节/字）	※	※	※	※	※	※
	IDIV src	带符号除法（字节/字）	※	※	※	※	※	※
符号扩展	CBW	字节扩展	—	—	—	—	—	—
	CWD	字扩展	—	—	—	—	—	—
十进制调整	AAA	非组合 BCD 码加法调整	※	※	※	¤	※	¤
	DAA	组合 BCD 码加法调整	※	¤	¤	¤	¤	¤
	AAS	非组合 BCD 码减法调整	※	※	※	¤	※	¤
	DAS	组合 BCD 码减法调整	※	¤	¤	¤	¤	¤

注：表中“¤”表示运算结果影响标志位；“—”表示运算结果不影响标志位；“※”表示标志位为任意值。

3.3.3　逻辑运算与移位类指令

1. 逻辑运算指令

有以下 5 条逻辑运算指令，它们可对 8 位或 16 位操作数按位进行逻辑运算。

（1）逻辑与指令：AND　dst,src。

该指令将源操作数 src 和目的操作数 dst 按位进行逻辑“与”运算，运算结果送回 dst。指

令执行结果会影响 CF、OF、SF、PF 和 ZF 标志位，使 CF=0，OF=0，其他标志位按结果进行设置。

利用 AND 指令可将操作数中的某些位保持不变，而使其他一些特定位清 0，称为屏蔽。

例如：

```
AND  AL,0FH
```

如果给定(AL)=52H，则指令执行后，(AL)=02H，屏蔽了 AL 中的高 4 位。

（2）逻辑或指令：OR　dst,src。

实现 src 和 dst 按位进行逻辑“或”运算，运算结果送回 dst。指令执行结果会影响状态标志位，与 AND 指令相同。

利用 OR 指令可将操作数中的某些位保持不变，而使其他一些位置 1。

例如：

```
OR  AL,F0H
```

如果给定(AL)=43H，则指令执行后，(AL)=F3H，达到将字节的高 4 位置 1 的目的。

（3）逻辑异或指令：XOR　dst,src。

实现 src 和 dst 按位进行逻辑“异或”运算，将运算结果送回 dst。指令执行结果会影响标志位，与 AND 指令相同。

例如：

```
XOR  AX,AX        ;该指令执行后将累加器清 0，同时 CF=0。
```

（4）逻辑非指令：NOT　dst。

为单操作数指令，对给定的操作数 dst 逐位取反，结果送回 dst。该指令执行后不影响任何标志位。

例如：

```
NOT  AL
```

若给定(AL)=01111000B，则指令执行后，(AL)=10000111B。

（5）测试指令：TEST　dst,src。

两个操作数执行“与”操作，结果不回送 dst，只影响标志位。该指令常用来检测操作数的某一位或某几位是“0”还是“1”。

例如：

```
TEST  AL,80H  ;检测 AL 中的数据是正数还是负数，当 D7=0 时为正数，ZF=1；否则为负数，
              ZF=0
```

2. 移位指令

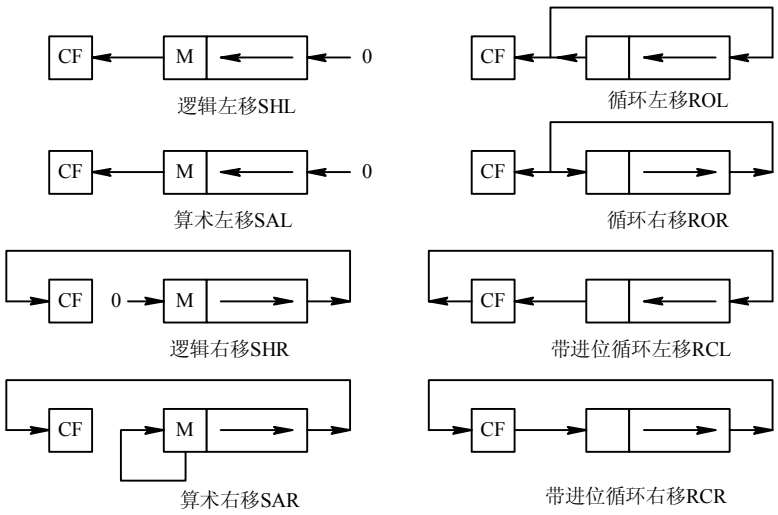
移位操作类指令可以对字节或字数据中的各位进行算术移位、逻辑移位或循环移位。

移位指令的格式为：　SHL/SAL/SHR/SAR　dst,1/ CL

循环移位指令的格式为：ROL/ ROR/ RCL/ RCR　dst,1/ CL

上述指令分别对操作数进行逻辑左移 SHL、算术左移 SAL、逻辑右移 SHR、算术右移 SAR、循环左移 ROL、循环右移 ROR、带进位的循环左移 RCL、带进位的循环右移 RCR 等操作。操作数可以是字节或字操作。

图 3-8 所示为各种移位操作的功能示意。指令中操作数可由任何寻址方式获得，位移次数可以取 1，也可以将位移的次数送到 CL 寄存器中。



（注：CF为进位位、M为符号位）

图 3-8　移位指令的操作功能示意

移位指令对各标志位的影响如下：

（1）CF 标志位要根据各种移位指令而定。OF 标志位可表示移位后的符号位与移位前是否相同，即当位移为 1，移位后的最高有效位的值发生变化时，OF 置“1”，否则清“0”。

（2）循环移位指令均影响 CF、OF、SF、ZF、PF 标志位。

（3）移位指令根据移位后的结果设置 SF、ZF、PF 标志位，AF 标志位无定义。

在程序设计中，常常用逻辑左移和逻辑右移指令来实现将无符号操作数乘以 2 或除以 2 的操作。要进行将带符号的数乘以 2 或除以 2 的运算，可以通过算术左移和算术右移指令来实现。

【例 3.16】分析以下移位指令的操作功能。

```
SHL   AL,1          ;AL 中内容向左移动 1 位，执行 (AL)×2 操作
MOV   CL,4          ;移位次数送 CL 保存
SHL   AL,CL         ;AL 中内容向左移动 CL 中指定的 4 位，空出位补 0
SHR   AX,1          ;AX 中内容向右移动 1 位，执行 (AX)/2 操作
```

注意：用左、右移位指令实现乘、除运算要比用乘、除法指令实现所需时间短得多。此外，循环移位指令可用来检测寄存器或存储单元中数据含 1 或 0 的个数，因为用循环移位指令循环 8 次，数据又恢复了，只要对 CF 进行检测，就可计算出 1 或 0 的个数。

3.3.4　串操作类指令

数据串是存储器中的一串字节或字的数据序列。8086 指令系统中设置了串操作指令，其操作对象是内存中地址连续的字节串或字串。在每次操作后能够自动修改地址指针，为下一次操作作准备。

基本串操作指令有串传送（MOVS）、串比较（CMPS）、串扫描（SCAS）、串存取（LODS、STOS）等。任何一个基本串操作指令的前面都可以加一个重复操作前缀，使指令操作重复，这样在处理长数据串时要比用循环程序速度快得多。

串操作指令具有以下几个共同的特点：

（1）约定以 DS:SI 寻址源串，以 ES:DI 寻址目标串。指令中不必指明操作数。其中源串

的段寄存器 DS 可通过加段超越前缀而改变，但目标串的段寄存器 ES 不能超越。源操作数常用在现行的数据段，隐含段寄存器 DS；目的操作数总是在现行的附加段，隐含段寄存器 ES。

（2）采用方向标志规定串处理方向。若方向标志 DF＝0，则从低地址向高地址方向处理，地址指针增量，字节操作时地址指针加 1，字操作时地址指针加 2；若 DF＝1，则处理方向相反，地址指针减量，字节操作时地址指针减 1，字操作时地址指针减 2。每一次操作以后修改地址指针，源串、目标串的两个地址指针 SI 和 DI 都将根据方向标志 DF 的值自动增量或减量，以指向串中下一项。

（3）可在串操作指令前加重复前缀用来对一个以上的串数据进行操作。这时必须用 CX 作为重复次数计数器，存放被处理数据串的元素个数（字节个数或字个数）。串操作指令每执行一次，CX 值自动减 1，直至减为 0 则停止串操作。

（4）重复的数据串处理过程可以被中断。CPU 在处理数据串中的下一元素之前识别中断并转入中断服务程序。在中断返回以后，重复过程从中断点继续执行下去。

（5）若串操作指令的基本操作影响标志 ZF（如 CMPS、SCAS），则可加重复前缀 REPE/REPZ 或 REPNE/REPNZ，此时操作重复进行的条件不仅要求（CX）≠0，而且同时要求 ZF 的值满足重复前缀中的规定。

（6）除了串比较指令和串搜索指令外，其余串操作指令均不影响标志位。

1. 串传送指令

```
指令格式：MOVS   dst，src     ;用于字串或字节串传送，由给定的数据类型确定
          MOVSB                ;字节串传送，SI和DI的内容±1
          MOVSW                ;字串传送，SI和DI的内容±2
```

该指令的操作功能为：将位于 DS 段的由 SI 所指示的存储单元的内容传送到位于 ES 段由 DI 所指示的存储单元，然后修改地址指针 SI 和 DI，以指向串中的下一个元素。

串传送指令在执行前，都必须把 SI 指向源操作数，DI 指向目的操作数，并将 DF 置 1 或清 0。

2. 串存储指令

```
指令格式：STOS   dst           ;用于字串或字节串存储，由给定的数据类型确定
          STOSB                ;字节存储
          STOSW                ;字存储
```

该指令的操作功能为：把 AL 或 AX 中的内容存入由 DI 指示的 ES 段中的字节数据或字数据，并根据 DF 的值及数据类型来修改 DI 中的内容。在该指令执行前，要将存入的内容预先放到 AL 或 AX 中，并设置 DF、DI 初始值。

STOS 指令的执行不影响标志位。这条指令和 REP 指令配合使用，可用来将存储区中的某一连续区域放入相同的内容。

3. 取串指令

```
指令格式：LODS   src           ;用于字串或字节串的取出，由给定的数据类型确定
          LODSB                ;取字节串
          LODSW                ;取字串
```

该指令的操作功能为：把 SI 指示的 DS 段中的字节数据或字数据传送至 AL 或 AX，并根据 DF 的值及数据类型来调整 SI 中的内容。该指令的执行不影响标志位。

4. 串比较指令

指令格式：CMPS dst,src ;用于字串或字节串比较，由给定的数据类型确定
CMPSB ;字节串比较
CMPSW ;字串比较

该指令的操作功能为：完成两个字节数据或字数据的相减，结果不回送，只影响状态标志位，并根据DF的值及数据类型来修改DI的内容。设置SI指向被减数，DI指向减数，并设置DF值。

5. 串搜索指令

指令格式：SCAS dst,src ;用于字串或字节串搜索，由给定的数据类型确定
SCASB ;字节串搜索
SCASW ;字串搜索

该指令的操作功能为：将AL或AX中的内容减去字节数据或字数据，结果不回送，只影响状态标志位，并根据DF的值及数据类型来调整DI的内容。在该指令执行前，AL或AX中设置被搜索的内容，DI指向被搜索的字符串的首单元，并设置DF值。

6. 方向标志清除、设置指令

指令格式：CLD ;方向标志清除指令，使 DF=0，可使串操作地址自动增量
STD ;方向标志设置指令，使 DF=1，可使串操作地址自动减量

7. 重复操作前缀

指令格式：REP ;重复操作前缀
REPE/REPZ ;相等/为零时重复操作前缀
REPNE/REPNZ ;不相等/不为零时重复操作前缀

指令的操作功能为：

REP 用在 MOVS、STOS、LODS 指令之前，重复次数预先送入 CX 中，每执行一次串操作指令，CX 中的内容自动减 1，一直重复到(CX)=0，操作结束。

REPE/REPZ 用在 CMPS、SCAS 指令之前，每执行一次串操作指令，CX 中的内容自动减 1，并判断 ZF 是否为 0，当(CX)=0 或 ZF=0 时，重复操作结束。

REPNE/REPNZ 用在 CMPS、SCAS 指令之前，每执行一次串操作指令，CX 中的内容自动减 1，并判断 ZF 是否为 1，当(CX)=0 或 ZF=1 时，重复操作结束。

【例 3.17】已知两个字节串 STR1 和 STR2 存放在内存中，设串的长度为 10，试比较 STR1 和 STR2 是否相等，如果相等，将标志单元 DL 置“全 1”，否则置“全 0”。

程序段设计如下：

```
        LEA  SI,STR1          ;将源字符串首地址送 SI 指针寄存器
        LEA  DI,STR2          ;将目标字符串首地址送 DI 指针寄存器
        MOV  CX,10            ;字符串长度送 CX
        CLD                   ;将 DF 标志位清 0，按递增方式进行
        REPE  CMPSB           ;按字节重复进行比较
        JNZ   NEXT1           ;如果两个字符串不相等，则转至 NEXT1
        MOV  DL,FFH           ;否则，字符串相等，将 DL 单元置全 1
        JMP  NEXT2            ;跳转到 NEXT2
NEXT1:MOV  DL，00H            ;字符串不相等，将 DL 单元置全 0
NEXT2:HLT                     ;暂停
```

【例 3.18】将内存区域 BUF1 开始存储的 100 个字节数据传送到从 BUF2 开始的存储区中。

（1）采用串传送指令的程序段如下：

```
        LEA  SI,BUF1                ;源数据区首址送 SI
        LEA  DI,BUF2                ;目标数据区首址送 DI
        MOV CX,100                  ;串长度送 CX
        CLD                         ;清方向标志，按正向传送
  NEXT:MOVSB                        ;串传送一个字节
        DEC  CX                     ;计数器减 1
        JNZ  NEXT                   ;判断是否传送完毕，没完则继续
  DONE:HLT                          ;暂停
```

（2）采用重复传送指令的程序段如下：

```
        LEA  SI,BUF1                ;源数据区首址送 SI
        LEA  DI,UF2                 ;目标数据区首址送 DI
        MOV CX,100                  ;串长度送 CX
        CLD                         ;清方向标志，按正向传送
        REP  MOVSB                  ;重复传送至（CX）=0 结束
        HLT                         ;暂停
```

3.3.5　控制转移类指令

程序的执行一般是按指令顺序逐条执行的，但有时需要改变程序的执行流程。控制转移类指令就是用来改变程序执行的方向，也就是修改 IP 和 CS 的值。通过控制转移指令可实现各种结构化程序设计，如分支结构程序、循环结构程序等。

按程序的转移位置可将转移指令分为段内转移和段间转移：

（1）如果指令给出改变 IP 中内容的信息，转移的目标位置和转移指令在同一个代码段，则称为段内转移；

（2）如果指令给出改变 IP 中内容的信息，又给出改变 CS 中内容的信息，转移的目标位置和转移指令不在同一个代码段，则称为段间转移。

根据转移指令的功能，可分为无条件转移指令、条件转移指令、循环控制指令、子程序调用和返回指令 4 类。下面分别进行讨论。

1. 无条件转移指令

无条件转移指令 JMP 用来控制程序转移到指定的位置去执行，指令中要给出转移位置的目标地址，通常有以下 5 种形式，如表 3-3 所示。

表 3-3　无条件转移指令及其功能

指令助记符	指令名称	指令操作功能
JMP　SHORT opr	段内直接短转移	无条件转移到指定的目标地址 opr。opr 为当前的 IP 值与指令中给定的 8 位偏移量之和，在-128～+127 范围内转移，SHORT 为属性运算符
JMP　NEAR　PTR opr	段内直接近转移	无条件地转移到指定的目标地址 opr。该地址为当前 IP 值与指令中给定的 16 位偏移量之和，在-32768～+32767 范围内转移，NEAR PTR 是类型说明符
JMP　WORD　PTR opr	段内间接转移	无条件转移到指定的目标地址。寄存器寻址时，将寄存器中的内容送到 IP 中；存储器寻址时，按寻址方式计算出有效地址和物理地址，用物理地址去读取内存中的数据送给 IP 指针

续表

指令助记符	指令名称	指令操作功能
JMP FAR PTR opr	段间直接转移	转移到指定段内的目标地址。由操作数决定的段地址送 CS，段内偏移地址送 IP。汇编时 opr 所对应的偏移量和所在代码段的段地址放在操作码之后，需要 4 个字节的存储单元
JMP DWORD PTR opr	段间间接转移	完成段间转移，由 opr 的寻址方式计算出有效地址和物理地址，通过物理地址去读取内存中连续的两个字数据，其中低位字送给 IP，高位字送给 CS

【例 3.19】给定(IP)=0012H，(BX)=0110H，(DS)=2000H，(20110H)=50H，(20111H)=01H。

执行指令：JMP WORD PTR[BX]

分析指令的操作功能。

解：该指令为段内间接转移方式，目标地址为存储器寻址方式。

操作数的有效地址为：EA=(BX)=0110H

物理地址为：PA=DS×10H+EA=20110H

指令执行后：(IP)=0150H，即该指令将跳转到以 IP 指针为 0150H 单元的目标地址开始执行。

【例 3.20】给定(CS)=3000H，(IP)=0032H，(BX)=0100H，(DS)=2000H，(20120H)=80H，(20121H)=10H，(20122H)=20H，(20123H)=40H。

执行指令：JMP DWORD PTR[BX+0020H]，分析指令的操作功能。

解：该指令为段间间接转移方式，目标地址为存储器寻址方式。

操作数的有效地址为：EA=(BX)+0020H=0120H

物理地址：PA=DS×10H+EA=20120H

指令执行后，从 20120H 单元开始取出连续的 4 个字节数据，前两个单元的数据送 IP，后两个单元的数据送 CS。最后结果为：(IP)=1080H，(CS)=4020H。

2. 条件转移指令

8086 指令系统具有一系列的条件转移指令，以某些标志位的状态或有关标志位的逻辑运算结果作为依据来决定是否发生转移。条件转移指令是根据上一条指令所设置的条件码来测试，被测试的内容为状态标志位。满足测试条件则转移到指令中指定的位置去执行，如果不满足条件则顺序执行下一条指令。

条件转移指令都为短转移，即转移的相对地址位移范围在-128～+127。当满足转移条件时，将位移量与当前的指令寄存器 IP 的内容相加，由此形成所需的程序地址并开始执行。

条件转移指令的执行不会影响标志位。

条件转移指令根据判断的标志位不同，通常可以归纳为 3 类，即判断单个标志位状态、比较无符号数高低和比较带符号数大小。这 3 类指令在使用之前，应该有比较 CMP、测试 TEST、加减或逻辑运算等指令。

各类条件转移指令的助记符、指令名称及转移条件等列于表 3-4 中。

表 3-4　条件转移指令

指令助记符	转移条件	指令含义	指令助记符	转移条件	指令含义
JZ/JE	ZF＝1	等于零/相等	JC/JB/JNAE	CF=1	进位/低于/不高于等于
JNZ/JNE	ZF＝0	不等于零/不相等	JNC/JNB/JAE	CF=0	无进位/不低于/高于等于
JS	SF＝1	符号为负	JBE/JNA	CF=1 或 ZF=1	低于等于/不高于
JNS	SF＝0	符号为正	JNBE/JA	CF=0 且 ZF=0	不低于等于/高于
JO	OF＝1	结果有溢出	JL/JNGE	SF≠OF	小于/不大于等于
JNO	OF＝0	结果无溢出	JNL/JGE	SF=OF	不小于/大于等于
JP/JPE	PF＝1	有偶数个“1”	JLE/JNG	ZF≠OF 或 ZF=1	小于等于/不大于
JNP/JPO	PF＝0	有奇数个“1”	JNLE/JG	SF=OF 且 ZF=0	不小于等于/大于

从表 3-4 中可知，判断转移条件共有 16 种，其中，用于判断单个标志位状态的指令有 8 种，它们是根据某一个状态标志是 0 或 1 来决定是否转移；用于比较无符号数高低的指令有 4 种，它们需要利用 CF 标志来确定高低、利用 ZF 标志来确定相等；用于比较带符号数大小的指令有 4 种，它们需要组合 OF、SF 标志、并利用 ZF 标志来确定相等与否。

【例 3.21】已知在内存中有两个无符号字节数据 NUM1 和 NUM2，找出其中的大数送到 MAX 单元保存。

程序段如下：

```
        MOV  AL,NUM1          ;取出数据 NUM1 送到 AL 中
        CMP  AL,NUM2          ;和数据 NUM2 进行比较
        JA   NEXT             ;NUM1＞NUM2 时，转到 NEXT
        MOV  AL,NUM2          ;否则取出 NUM2 数据送到 AL 中
NEXT:MOV  MAX,AL              ;将 AL 中保存的大数送到 MAX 单元
        HLT                   ;暂停
```

本题若改为两个带符号数的比较，则程序中条件转移指令应为 JG。

此外，在条件转移指令中，有时还会专门对 CX 寄存器的值进行测试，当(CX)=0 时产生转移。该指令的格式为：JCXZ　opr ；测试条件是：若(CX)=0，则转移到指定位置。JCXZ 指令常用于循环程序中对循环次数进行控制。

3. 循环控制指令

将一段代码程序执行多次操作即为循环，采用循环控制指令实现。循环控制转向的目的地址是在以当前 IP 内容为中心的-128～+127 的范围内，指令采用 CX 作为计数器，每执行一次循环，CX 内容减 1，直到为零后循环结束。

循环控制指令是根据标志位状态进行控制操作的，指令本身不影响标志位。8086 指令系统中有以下三种循环控制语句。

（1）循环控制指令 LOOP。

指令格式：LOOP　opr

LOOP 指令用在循环次数固定的循环结构中，循环次数送入 CX，语句标号 opr 为循环体的入口。该指令是以 CX 的内容作为计数控制，作(CX)←(CX)-1 的操作，并进行判断，当 CX≠0 时，转移到由操作数指示的目的地址，即(IP)←(IP)+位移量，进行循环处理；当 CX=0 时，

结束循环。

一条 LOOP 指令相当于下面两条指令的作用：

```
DEC  CX          ;(CX)-1
JNZ   NEXT       ;不为零转移
```

（2）为零或相等时循环控制指令 LOOPZ/LOOPE。

指令格式：LOOPZ/LOOPE　opr

LOOPZ/LOOPE 指令可完成当 ZF=1 且 CX≠0 条件下的循环操作。在 LOOPZ 或 LOOPE 所做的控制循环操作过程中，除了进行(CX)←(CX) -1 的操作，还要判断 CX 是否为零。此外，还将判断标志位 ZF 的值。

（3）不为零或不相等时循环控制指令 LOOPNZ/LOOPNE。

指令格式：LOOPNZ/LOOPNE　opr

LOOPNZ 或 LOOPNE 指令可完成当 ZF=0 且(CX)≠0 的条件下控制循环操作。其操作过程类似于 LOOPZ 或 LOOPE 指令。

【例 3.22】用循环程序来实现 S=1+2+3+…+100 计算。

程序段如下：

```
      MOV    CX,100              ;数据长度送 CX 计数器
      XOR    AL,AL               ;将 AL 寄存器清零
      MOV    BL,1                ;对 BL 赋初值为 1
NEXT:ADD    AL,BL               ;（AL）←（AL）+（BL）
      INC    BL                  ;BL 加 1
      LOOP   NEXT                ;（CX）-1≠0 转 NEXT 位置继续
       HLT                       ; 否则，累加完毕，结果保存在 AL 中，程序暂停
```

【例 3.23】在内存中有一个具有 N 个字节的数据串，首单元地址为 DATA-BUF，找出第一个不为 0 的数据的地址送到 ADDR 单元中。

程序段如下：

```
      LEA    SI,DATA-BUF         ;取数据串的首单元地址
      MOV    CX,N                ;数据串长度送 CX 计数器
      MOV    AL,0                ;对 AL 清零
      DEC    SI                  ;循环初始化
NEXT:INC    SI                  ;地址指针加 1
      CMP    AL,[SI]             ;将内存中的数据和 AL 中的内容进行比较
      LOOPZ  NEXT                ;为 0 且未比较到末尾转 NEXT 位置继续
      JZ    EXIT                 ;判断 ZF=1，条件成立转 EXIT
      MOV   ADDR,SI              ;ZF=0，将 SI 中的内容送 ADDR 单元
EXIT:HLT                         ;程序暂停
```

本例中，有两种情况可以使循环结束：一种是经比较找到了第一个不为 0 的数据；另一种是 N 个数据全部比较结束后未找到数据 0，所以退出循环时要判断 ZF 是否为 1。

4. 子程序调用和返回指令

在复杂程序的设计过程中，通常把系统的总体功能分解为若干个小的功能模块。每一个小功能模块对应一个过程。在汇编语言中，过程又称为子程序。程序中可以由调用程序（称之为主程序）来调用这些子程序，子程序执行完毕后要返回主程序调用处继续执行下一条指令。子程序调用及返回指令是程序设计中常用的指令，在程序的执行过程中，它们可对某一个具有

独立功能子程序进行多次调用操作，由此可实现模块化的程序设计。

8086 指令系统提供了子程序调用 CALL 和返回指令 RET。

（1）子程序调用指令。

指令格式为：CALL　NEAR　PTR　opr　;段内调用

CALL　FAR　PTR　opr　　;段间调用

其中，opr 为子程序名（即子程序第一条指令的符号地址）。

为了保证调用之后正确地返回，需要把 CALL 指令的下一条指令的地址（称为断点）压入堆栈进行保护。

下面分别讨论段内和段间的子程序调用指令所完成的操作：

- 对于段内的直接调用指令，其指令中的目的地址为一个 16 位目的地址的相对位移量。CALL 指令的操作可完成(SP)←(SP)-2，并将指令指针 IP 压入堆栈，然后修改 IP 的内容，即(IP)←(IP)+相对位移量。
- 对于段内的间接调用指令，指令中所指定的 16 位通用寄存器或存储单元的内容为目的地址的位移量。CALL 指令的操作可完成(SP)←(SP)-2，将指令寄存器 IP 压入堆栈，然后取出目的地址位移量送入 IP。
- 对于段间的直接调用指令，其目的地址不仅包括位移量还包括段地址，它们由指令直接给出。因此 CALL 指令的操作可完成(SP)←(SP)-2，将现行指令的段地址（CS 的内容）压入堆栈，然后作(SP)←(SP)-2，将现行的位移量（IP 的内容）压入堆栈。最后将指令中所指示的段地址及位移量分别送入 CS 及 IP 中。
- 对于段间的间接调用指令，其目的地址由指令的寻址方式所决定。将现行地址压入堆栈的操作同段间直接调用指令。段地址及段内位移量送入 CS 及 IP 将由寻址方式来决定。

（2）子程序返回指令 RET。

指令格式：　RET

或　　RET　表达式

RET 指令为子程序的最后一条指令。子程序操作完成之后，RET 指令使其返回主程序，该指令所完成的操作是从堆栈中弹出返回地址，送入指令寄存器 IP 和段寄存器 CS。

由于子程序调用分为段内调用和段间调用，因此返回指令也可分为段内返回和段间返回两种。

- 段内返回是指将 SP 所指示的堆栈顶部弹出一个字的返回地址，送入 IP 中。
- 段间返回是指从堆栈顶部弹出的返回地址为 2 个字的内容，其中一个字送入 IP，另一个字送入 CS 中，以表示不同的段。

此外，对于段内和段间返回都可带立即数，如 RET　0100H。

由于主程序在调用子程序之前利用堆栈进行参数传递，因此，利用带立即数的返回指令可以对堆栈指针 SP 进行调整，即(SP)←(SP)+立即数，使堆栈指针寄存器 SP 所指示的位置为调用之前的位置。

子程序调用和返回指令对标志位无影响。

【例 3.24】在主程序中执行一条子程序段内调用语句。

调用格式如下：

```
MAIN  PROC  FAR                    ;定义主程序
      MOV   AX,DATA                ;DS 初始化
      MOV   DS,AX
        ……
      CALL  SUB1                   ;调用子程序 SUB1
        ……
SUB1  PROC  NEAR                   ;定义子程序
      PUSH   AX                    ;保护现场
      PUSH   BX
       ……
      RET                          ;子程序返回
```

3.3.6 处理器控制类指令

这类指令主要用于修改状态标志位、控制 CPU 的功能，如使 CPU 暂停、等待、空操作等。其指令表示符号和功能如表 3-5 所示。

表 3-5 处理器控制类指令

指令名称	助记符	指令功能	指令名称	助记符	指令功能
进位标志设置指令	CLC	CF 位清 0	暂停指令	HLT	CPU 进入暂停状态，不进行任何操作。
	STC	CF 位置 1			
	CMC	CF 位求反	等待指令	WAIT	CPU 进入等待状态
方向标志设置指令	CLD	DF 位清 0	空操作指令	NOP	CPU 空耗一个指令周期
	STD	DF 位置 1	封锁指令	LOCK	CPU 执行指令时封锁总线
中断允许控制标志设置指令	CLI	IF 位清 0	交权指令	ESC	指令将处理器的控制权交给协处理器
	STI	IF 位置 1			

（1）对于各标志位的设置，可按照程序的要求选择相关指令进行处理。

（2）HLT 暂停指令可使机器暂停工作，处理器处于停机状态，以便等待一次外部中断到来，中断结束后，退出暂停继续执行后续程序。对系统进行复位操作也会使 CPU 退出暂停状态。

（3）WAIT 等待指令使处理器处于空转状态，也可用来等待外部中断发生，但中断结束后仍返回 WAIT 指令继续等待。

（4）NOP 空操作指令不执行任何操作，其机器码占一个字节单元，在调试程序时往往用这种指令占一定数量的存储单元，以便在正式运行时用其他指令取代；执行该指令花 3 个时钟周期，也可用在延时程序中拼凑时间。

（5）LOCK 是一个一字节的前缀，可放在任何指令的前面。执行时，使引脚 LOCK 有效，在多处理器具有共享资源的系统中可用来实现对共享资源的存取控制，即通过对标志位进行测试，进行交互封锁。根据标志位状态，在 LOCK 有效期间，禁止其他的总线控制器对系统总线进行存取。当存储器和寄存器进行信息交换时，LOCK 前缀指令非常有用。

【例 3.25】利用 LOCK 封锁指令，在多处理器系统中，实现对共享资源存取的控制。

解：根据题目要求，有如下的程序段。

```
CHECK:MOV   AL,1                   ;AL 置 1（隐含封锁）
      LOCK  SEMA,AL                ;测试并建立封锁
```

```
TEST  AL,AL                ;由 AL 设置标志
JNZ   CHECK                ;封锁建立则重复
MOV   SEMA,0               ;完成，清除封锁
```

（6）执行 ESC 指令时，协处理器可监视系统总线，并且能取得这个操作码。ESC 和 LOCK 指令用在 8086 最大工作方式中，分别处理主机和协处理器以及多处理器间的同步关系。

3.4　Pentium 微处理器新增指令和寻址方式

Pentium 微处理器以最先进的技术将 PC 机推向了一个崭新的发展阶段，Pentium 拥有全新的结构与功能，它采用了超标量体系结构，具有动态转移预测、流水线浮点部件、片内超高速缓冲存储器、较强的错误检测和报告功能、测试挂钩等新技术。

下面简要分析 Pentium 微处理器与 8086、80X86 系列芯片在指令及寻址方式等方面中的不同和特点。

3.4.1　Pentium 微处理器寻址方式

1. Pentium 微处理器内部寄存器和指令格式

由于 Pentium 微处理器采用 32 位指令，它的内部寄存器和指令格式与 16 位微处理器存在不同，主要有以下几方面：

（1）指令的操作数可以是 8 位、16 位或 32 位。

（2）根据指令的不同操作数字段可以是 0～3 个，三操作数时，最左边的操作数为目的操作数，右边两个操作数均为源操作数。

（3）在部分不存放结果的单操作数指令中，可以采用立即数作为操作数。

（4）部分指令对操作数的数据类型不是简单地要求一致，而是要有不同的匹配关系。

（5）立即数寻址方式中，操作数可以是 32 位的立即数；寄存器寻址方式中，操作数可以是 32 位通用寄存器。

（6）存储器操作数寻址方式中，操作数可达 32 位，寻址方式既可采用 16 位的地址寻址方式也可采用 32 位的扩展地址寻址方式。

（7）16 位微处理器原有的 4 个通用数据寄存器扩展为 32 位，更名为 EAX、EBX、ECX 和 EDX。

（8）原有的 4 个用于内存寻址的通用地址寄存器同样扩展为 32 位，更名为 ESI、EDI、EBP、ESP。

（9）指令指针寄存器扩展为 32 位，更名为 EIP，实地址方式下仍然可以使用它的低 16 位 IP。

（10）在原有的 4 个段寄存器基础上，增加了 2 个新的段寄存器 FS 和 GS，段寄存器长度仍然为 16 位，但是它存放的不再是“段基址”，而是代表这个段编号的 13 位二进制数，称为“段选择字”。

（11）32 位微处理器增加了 4 个系统地址寄存器，它们分别是存放“全局段描述符表”首地址的 GDTR，存放“局部段描述符表”选择字的 LDTR，存放“中断描述符表”首地址的 IDTR，存放“任务段”选择字的“任务寄存器”TR。

（12）标志寄存器也扩展为 32 位，更名为 EFLAGS，除了原有的状态、控制标志外，还

增加了 2 位，表示 IO 操作特权级别的 IOPL，表示进入虚拟 8086 方式的 VM 标志等。

（13）新增加了 5 个 32 位的控制寄存器，命名为 CR0～CR4，CR0 寄存器的 PE=1 表示目前系统运行在“保护模式”，PG=1 表示允许进行分页操作。CR3 寄存器存放“页目录表”的基地址。

（14）新增了 8 个用于调试的寄存器 DR0～DR7，2 个用于测试的寄存器 TR6 和 TR7。

2. Pentium 微处理器新增寻址方式

如前所述，8086 微处理器有固定寻址、立即数寻址、寄存器寻址、直接寻址、寄存器间接寻址、寄存器相对寻址、基址变址寻址和相对基址变址寻址等 8 种寻址方式，而 Pentium 微处理器有 11 种寻址方式。与 8086 相比新增加的 3 种寻址方式分别是比例变址寻址方式、基址加比例变址寻址方式和带位移量的比例变址寻址方式。

（1）比例变址寻址方式。

其有效地址为：EA=[变址寄存器]×比例因子+位移量

该方式下操作数有效地址是变址寄存器的内容乘以指令中指定的比例因子再加上位移量之和，所以有效地址由三种成分组成。乘比例因子的操作是在 CPU 内部由硬件完成的。

这种寻址方式与寄存器相对寻址相比，增加了比例因子，其优点在于：对于元素大小为 2、4、8 字节的数组，可以在变址寄存器中给出数组元素下标，而由寻址方式控制直接用比例因子把下标转换为变址值。

例如：

```
MOV  EAX,COUNT[ESI×4]
```

（2）基址加比例变址寻址方式。

其有效地址为：EA=[基址寄存器]+[变址寄存器]×比例因子

该方式下 EA 是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之和，这种寻址方式与基址变址寻址方式相比，增加了比例因子。

例如：

```
MOV  ECX,[EAX][EDX×8]
```

（3）带位移量的基址加比例变址寻址方式。

其有效地址为：EA=[基址寄存器]+[变址寄存器]×比例因子+位移量

操作数的有效地址是变址寄存器的内容乘以比例因子，加上基址寄存器的内容，再加上位移量之和，所以有效地址由四种成分组成。在寻址过程中，变址寄存器内容乘以比例因子的操作也是在 CPU 内部由硬件来完成。

这种寻址方式比相对基址变址寻址方式增加了比例因子，便于对元素为 2、4、8 字节的二维数组的处理。

例如：

```
MOV  EAX,TABLE[EBP][EDI×4]
```

需要注意的是：

- Pentium 微处理器在实地址方式下，一个段的最大长度仍然为 64KB，段基址是 16 的倍数，用段寄存器存放段基址。
- Pentium 有 6 个段寄存器，在寻址内存操作数时，指令给出的内存操作数的地址均为有效地址。

- 内存操作数有效地址可由一个 32 位基址寄存器、一个可乘上比例因子 1、2、4、8 的 32 位变址寄存器和一个不超过 32 位的常数偏移量组成。
- 32 位寻址情况下，8 个 32 位通用寄存器均可作基址寄存器，其中 ESP、EBP 以 SS 为默认段寄存器，其余 6 个通用寄存器均以 DS 为默认段寄存器。
- 基址字段、变址字段、偏移量字段可任意省略其一或其二。
- 比例因子与变址字段联合使用，如省略了变址字段，则比例因子不能独立存在。

3.4.2　Pentium 系列微处理器专用指令

Pentium 系列处理器的指令集是向上兼容的，它保留了 8086 和 80X86 微处理器系列的所有指令，因此，所有早期的软件可直接在奔腾机上运行。

从微处理器的指令系统中可看出，自 1985 年 Intel 公司推出 32 位微处理器 80386 以来，始终使用着几乎一样的指令系统，只是每提高一代便追加很少几条指令。Pentium 微处理器的指令集与 80486 相比变化不大，Pentium 的主要特色是拥有能使系统程序员实现多路处理 Cache 一致性协议的新指令，以及一条 8 字节比较交换指令和一条微处理器识别指令。

Pentium 处理器指令集中新增加了以下 3 条专用指令。

1. 比较和交换 8 字节数据指令 CMPXCHG8B

指令格式：CMPXCHG8B　opr1,opr2

该指令执行 64 位数据的比较和交换操作。执行时将存放在 opr1（64 位存储器）中的目的操作数与累加器 EDX：EAX 的内容进行比较，如果相等，则 ZF=1，并将源操作数 opr2（规定为 EDX：EAX）的内容送入 opr1；否则 ZF=0，并将 opr1 送到相应的累加器。

例如：

```
CMPXCHG8B  mem,ECX:EBX
```

指令执行后，如果 EDX:EAX=[mem]，则 ECX:EBX→[mem]，ZF=1

否则[mem]→EDX:EAX 且 ZF=0

2. CPU 标识指令 CPUID

指令格式：CPUID

该指令执行后可以将有关 Pentium 处理器的型号和特点等系列信息返回到 EAX 中。在执行 CPUID 指令前，EAX 寄存器必须设置为 0 或 1，根据 EAX 中设置值的不同，软件会得到不同的标志信息。

3. 读时间标记计数器指令 RDTSC

指令格式：RDTSC

在 Pentium 处理器中有一个片内 64 位计数器，称为时间标记计数器 TSC。计数器的值在每个时钟周期都自动加 1，执行 RDTSC 指令可以读出计数器 TSC 中的值，并送入寄存器 EDX:EAX 中，EDX 保存 64 位计数器中的高 32 位，EAX 保存低 32 位。

一些应用软件需要确定某个事件已执行了多少个时钟周期，在执行该事件之前和之后分别读出时钟标志计数器的值，计算两次值的差就可得出时钟周期数。

3.4.3　Pentium 系列微处理器控制指令

Pentium 处理器指令集中新增加了以下 3 条系统控制指令。

1. 读专用模式寄存器指令 RDMSR

RDMSR 指令使软件可访问专用模式寄存器的内容，执行指令时在访问的模式专用寄存器与寄存器组 EDX:EAX 之间进行 64 位的读操作。

2. 写专用模式寄存器指令 WRMSR

WRMSR 指令执行时在访问的模式专用寄存器与寄存器组 EDX:EAX 之间进行 64 位的写操作。

Pentium 处理器有两个专用模式寄存器，即机器地址检查寄存器（MCA）和机器类型检查寄存器（MCT）。

如果要访问机器地址检查寄存器 MCA，指令执行前需将 ECX 置为 0；而为了访问机器类型检查寄存器 MCT，需要将 ECX 置为 1。

3. 恢复系统管理模式指令 RSM

Pentium 处理器有一种称为系统管理模式（SMM）的操作模式，这种模式主要用于执行系统电源管理功能。外部硬件的中断请求使系统进入 SMM 模式，执行 RSM 指令后返回原来的实模式或保护模式。

8086 指令系统和寻址方式是汇编语言程序设计的基础，应熟练掌握其具体内容、特点和应用场合。

8086CPU 指令按操作数类别分隐含操作数、单操作数和双操作数 3 种类型；按操作数存放位置分立即数、寄存器操作数、存储器操作数和 I/O 端口操作数 4 种类型。需要进行频繁访问的数据可放在寄存器中，这样能够保证程序执行的速度更快。批量数据的处理可使用存储器操作数。

在指令中寻找操作数有效地址的方式称为寻址方式，寻址的目的是为了得到操作数。8086 系统有立即数寻址、寄存器寻址、直接寻址、寄存器间接寻址、寄存器相对寻址、基址变址寻址、相对基址变址寻址 7 种基本寻址方式。I/O 端口的寻址方式有直接端口寻址和寄存器间接端口寻址两种。在学习时，要弄清各类寻址方式的区别和特点，结合 8086 存储器分段，重点掌握和理解存储器寻址方式中有效地址 EA 和物理地址 PA 的计算方法。

8086 指令系统按功能分为数据传送类、算术运算类、逻辑运算类、串操作类、控制转移类、处理器控制类等指令。状态标志是 CPU 进行条件判断和控制程序执行流程的依据，大多数指令的执行不影响标志位，某些指令的执行会按照规则影响标志位，还有一些指令会按特定方式影响标志位。

实际应用中，要正确理解和运用各种指令格式、功能和注意事项。有关 8086CPU 指令集可参见本书的附录 A。

一、单项选择题

1．寄存器间接寻址方式中，要寻找的操作数位于（　　）中。

A．通用寄存器　　B．段寄存器　　C．内存单元　　D．堆栈区

2．下列传送指令中正确的是（　　）。

A．MOV　AL,BX　　B．MOV　CS,AX　　C．MOV　AL,CL　　D．MOV　[BX],[SI]

3．下列 4 个寄存器中，不允许用传送指令赋值的寄存器是（　　）。

A．CS　　B．DS　　C．ES　　D．SS

4．将 AX 清零并使 CF 位清零，下面指令错误的是（　　）。

A．SUB　AX,BX　　B．XOR　AX,AX　　C．MOV　AX,0　　D．AND　AX,0000H

5．指令 MOV　[SI+BP]，AX；其目的操作数的隐含段为（　　）。

A．数据段　　B．堆栈段　　C．代码段　　D．附加段

6．设(SP)=1010H，执行 PUSH　AX 后，SP 中的内容为（　　）。

A．1011H　　B．1012H　　C．100EH　　D．100FH

7．对两个带符号整数 A 和 B 进行比较，要判断 A 是否大于 B，应采用指令（　　）。

A．JA　　B．JG　　C．JNB　　D．JNA

8．已知(AL)=80H，(CL)=02H，执行指令 SHR　AL,CL 后的结果是（　　）。

A．(AL)=40H　　B．(AL)=20H　　C．(AL)=C0H　　D．(AL)=E0H

9．当执行完下列指令序列后，标志位 CF 和 OF 的值是（　　）。

```
MOV　AH,85H
SUB　 AH,32H
```

A．0,0　　B．0,1　　C．1,0　　D．1,1

10．JMP BX 的目标地址偏移量是（　　）。

A．SI 的内容　　B．SI 指向内存字单元的内容

C．IP+SI 的内容　　D．IP+[SI]

二、填空题

1．计算机指令通常由_____和_____两部分组成；指令对数据操作时，按照数据的存放位置可分为_____。

2．寻址的含义是_____；8086 指令系统的寻址方式按照操作数的存放位置可分为_____；其中寻址速度最快的是_____。

3．若指令操作数保存在存储器中，操作数的段地址隐含放在_____中；可以采用的寻址方式有_____。

4．指令 MOV　AX,ES:[BX+0200H] 中，源操作数位于_____；读取的是_____段的存储单元内容。

5．堆栈是一个特殊的_____、_____，其操作是以_____为单位按照_____原则来处理；采用_____来指向栈顶地址，入栈时地址变化为_____。

6．I/O 端口的寻址有_____、_____两种方式；采用 8 位数时，可访问的端口地址为_____；采用 16 位数时，可访问的端口地址为_____。

三、判断题

1．各种 CPU 的指令系统是相同的。　　（　　）

2．在指令中，寻址的目的是找到操作数。　　（　　）

3．指令 MOV　AX,CX 采用的是寄存器间接寻址方式。（　　）

4．条件转移指令可以实现段间转移。（　　）

5．串操作指令只处理一系列字符组成的字符串数据。（　　）

6．LOOP 指令执行时，先判断 CX 是否为 0，如果为 0 则不再循环。（　　）

四、分析设计题

1．设(DS)=2000H，(ES)= 2100H，(SS)= 1500H，(SI)= 00A0H，(BX)= 0100H，(BP)= 0010H，数据变量 VAL 的偏移地址为 0050H，请指出下列指令的源操作数字段是什么寻址方式？它的物理地址是多少？

(1) MOV　AX,21H　　(2) MOV　AX,BX　　(3) MOV　AX,[1000H]

(4) MOV　AX,VAL　　(5) MOV　AX,[BX]　　(6) MOV　AX,ES:[BX]

(7) MOV　AX,[BP]　　(8) MOV　AX,[SI]　　(9) MOV　AX,[BX+10]

(10) MOV　AX,VAL[BX]　　(11) MOV　AX,[BX][SI]　　(12) MOV　AX,VAL[BX][SI]

2．给定寄存器及存储单元的内容为：(DS) = 2000H，(BX) = 0100H，(SI) = 0002H，(20100) = 32H，(20101) = 51H，(20102) = 26H，(20103) = 83H，(21200) = 1AH，(21201) = B6H，(21202) = D1H，(21203) = 29H。试说明下列各条指令执行完后，AX 寄存器中保存的内容是什么？

(1) MOV　AX,1200H　　(2) MOV　AX,BX　　(3) MOV　AX,[1200H]

(4) MOV　AX,[BX]　　(5) MOV　AX,1100[BX]　　(6) MOV　AX,[BX][SI]

3．分析下列指令的正误，对于错误的指令要说明原因并加以改正。

(1) MOV　AH,BX　　(2) MOV　[BX],[SI]

(3) MOV　AX,[SI][DI]　　(4) MOV　MYDAT[BX][SI],ES:AX

(5) MOV　BYTE　PTR[BX],1000　　(6) MOV　BX,OFFSET MAYDAT[SI]

(7) MOV　CS,AX　　(8) MOV　DS,BP

4．设 VAR1、VAR2 为字变量，LAB 为标号，分析下列指令的错误之处并加以改正。

(1)　ADD　　VAR1,VAR2　　(2)　MOV　　AL,VAR2

(3)　SUB　　AL,VAR1　　(4)　JMP　　LAB[SI]

(5)　JNZ　　VAR1　　(6)　JMP　　NEAR LAB

5．写出能够完成下列操作的 8086CPU 指令。

(1) 把 4629H 传送给 AX 寄存器。

(2) 从 AX 寄存器中减去 3218H。

(3) 把 BUF 的偏移地址送入 BX 中。

6．根据以下要求写出相应的汇编语言指令。

（1）把 BX 和 DX 寄存器的内容相加，结果存入 DX 寄存器中。

（2）用 BX 和 SI 的基址变址寻址方式，把存储器中的一个字节与 AL 内容相加，并保存在 AL 寄存器中。

（3）用寄存器 BX 和位移量 21B5H 的变址寻址方式把存储器中的一个字和 CX 相加，并把结果送回存储器单元中。

（4）用位移量 2158H 的直接寻址方式把存储器中的一个字与数 3160H 相加，并把结果送回该寄存器中。

（5）把数 25H 与 AL 相加，结果送回寄存器 AL 中。

7．写出将首地址为 BLOCK 的字数组的第 6 个字送到 CX 寄存器的指令序列，要求分别使用以下几种寻址方式：

（1）以 BX 的寄存器间接寻址。

（2）以 BX 的寄存器相对寻址。

（3）以 BX、SI 的基址变址寻址。