7.3.1　指令寻址

指令寻址比较简单，它分为顺序寻址和跳跃寻址两种。

顺序寻址可通过程序计数器 PC 加 1，自动形成下一条指令的地址；跳跃寻址则通过转移类指令实现。图 7.6 示意了指令寻址过程。

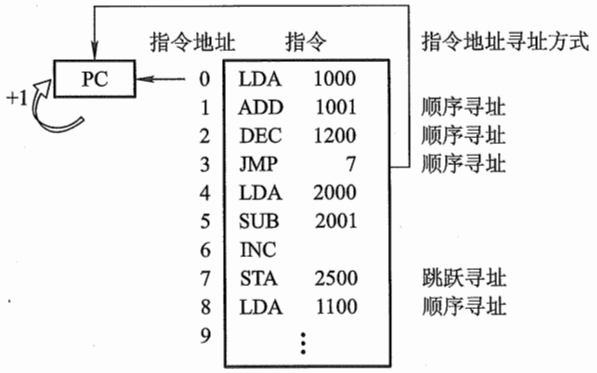


图 7.6　指令的寻址方式示意图

如果程序的首地址为 0，只要先将 0 送至程序计数器 PC 中，启动机器运行后，程序便按 0，1，2，3，7，8，9，…顺序执行。其中第 1、2、3 号指令地址均由 PC 自动形成。因第 3 号地址指令为“JMP 7”，故执行完第 3 号指令后，便无条件将 7 送至 PC，因此，此刻指令地址跳过 4、5、6 三条，直接执行第 7 条指令，接着又顺序执行第 8 条、第 9 条等指令。

关于跳跃寻址的转移地址形成方式，将在 7.3.2 节的直接寻址和相对寻址中做介绍。

7.3.2　数据寻址

数据寻址方式种类较多，在指令字中必须设一字段来指明属于哪一种寻址方式。指令的地址码字段通常都不代表操作数的真实地址，故把它称为形式地址，记作 A。操作数的真实地址称为有效地址，记作 EA，它是由寻址方式和形式地址共同来确定的。由此可得指令的格式应如图 7.7 所示。

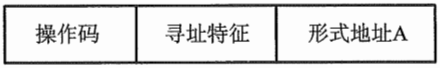


图 7.7　一种一地址指令的格式

为了便于分析研究各类寻址方式，假设指令字长、存储字长、机器字长均相同。

1. 立即寻址

立即寻址的特点是操作数本身设在指令字内，即形式地址 A 不是操作数的地址，而是操作数本身，又称之为立即数。数据是采用补码形式存放的，如图 7.8 所示，图中“#”表示立即寻址特

征标记。

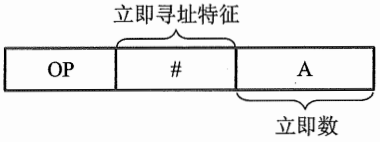


图 7.8 立即寻址示意图

可见,它的优点在于只要取出指令,便可立即获得操作数,这种指令在执行阶段不必再访问存储器。显然 A 的位数限制了这类指令所能表述的立即数的范围。

2. 直接寻址

直接寻址的特点是,指令字中的形式地址 A 就是操作数的真实地址 EA,即

$$EA = A$$

图 7.9 示意了直接寻址。

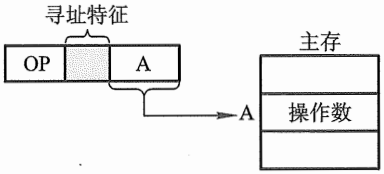


图 7.9 直接寻址示意图

它的优点是寻找操作数比较简单,也不需要专门计算操作数的地址,在指令执行阶段对主存只访问一次。它的缺点在于 A 的位数限制了操作数的寻址范围,而且必须修改 A 的值,才能修改操作数的地址。

3. 隐含寻址

隐含寻址是指指令字中不明显地给出操作数的地址,其操作数的地址隐含在操作码或某个寄存器中。例如,一地址格式的加法指令只给出一个操作数的地址,另一个操作数隐含在累加器 ACC 中,这样累加器 ACC 成了另一个数的地址。图 7.10 示意了隐含寻址。

又如 IBM PC(Intel 8086)中的乘法指令,被乘数隐含在寄存器 AX(16 位)或寄存器 AL(8 位)中,可见 AX(或 AL)就是被乘数的地址。又如字符串传送指令 MOVS,其源操作数的地址隐含在 SI 寄存器中(即操作数在 SI 指明的存储单元中),目的操作数的地址隐含在 DI 寄存器中。

由于隐含寻址在指令字中少了一个地址,因此,这种寻址方式的指令有利于缩短指令字长。

4. 间接寻址

倘若指令字中的形式地址不直接指出操作数的地址,而是指出操作数有效地址所在的存储单元地址,也就是说,有效地址是由形式地址间接提供的,即为间接寻址,即 EA=(A),如图 7.11 所示。

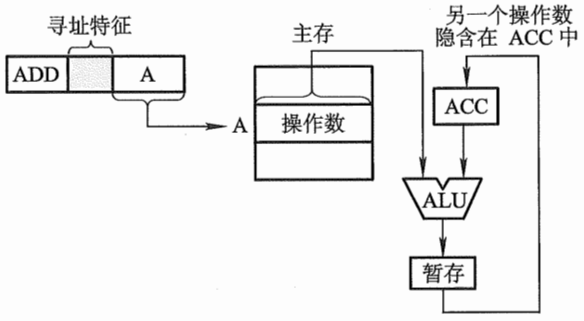
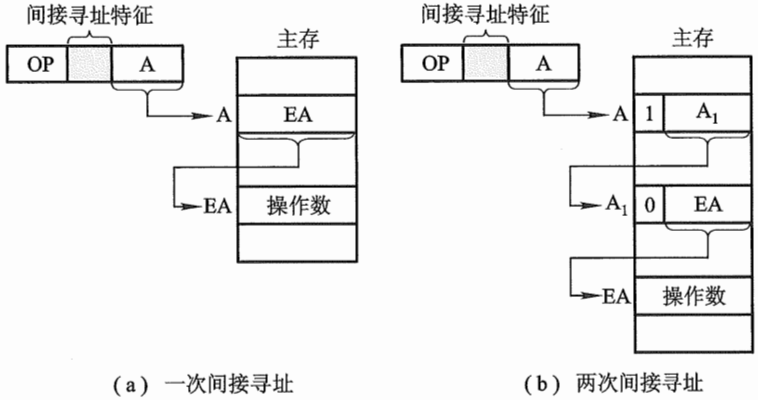


图 7.10　隐含寻址示意图



(a) 一次间接寻址　　(b) 两次间接寻址

图 7.11　间接寻址示意图

图 7.11(a)为一次间接寻址,即 A 地址单元的内容 EA 是操作数的有效地址;图 7.11(b)为两次间接寻址,即 A 地址单元的内容 A_1 还不是有效地址,而由 A_1 所指单元的内容 EA 才是有效地址。

这种寻址方式与直接寻址相比,它扩大了操作数的寻址范围,因为 A 的位数通常小于指令字长,而存储字长可与指令字长相等。若设指令字长和存储字长均为 16 位,A 为 8 位,显然直接寻址范围为 2^8,一次间接寻址的寻址范围可达 2^{16}。当多次间接寻址时,可用存储字的首位来标志间接寻址是否结束。如图 7.11(b)中,当存储字首位为“1”时,标明还需继续访存寻址;当存储字首位为“0”时,标明该存储字即为 EA。由此可见,存储字首位不能作为 EA 的组成部分,因此,它的寻址范围为 2^{15}。

间接寻址的第二个优点在于它便于编制程序。例如,用间接寻址可以很方便地完成子程序返回,图 7.12 示意了用于子程序返回的间址过程。

图中表示两次调用子程序,只要在调用前先将返回地址存入子程序最末条指令的形式地址 A 的存储单元内,便可准确返回到原程序断点。例如,第一次调用前,使[A] = 81,第二次调用

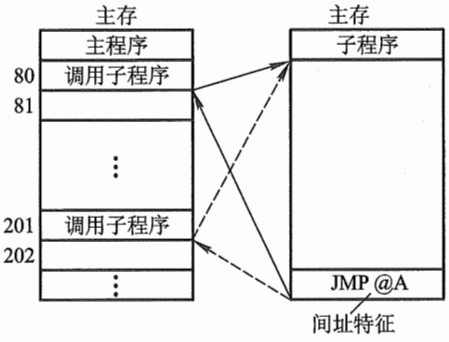


图 7.12 用于子程序返回的间址过程的示意图

前,使[A]=202。这样,当第一次子程序执行到最末条指令"JMP @A"(@为间址特征位),便可无条件转至81号单元。同理,第二次执行完子程序后,便可返回到202号单元。

间接寻址的缺点在于指令的执行阶段需要访存两次(一次间接寻址)或多次(多次间接寻址),致使指令执行时间延长。

5. 寄存器寻址

在寄存器寻址的指令字中,地址码字段直接指出了寄存器的编号,即 $EA=R_i$,如图7.13所示。其操作数在由 R_i 所指的寄存器内。由于操作数不在主存中,故寄存器寻址在指令执行阶段无须访存,减少了执行时间。由于地址字段只需指明寄存器编号(计算机中寄存器数有限),故指令字较短,节省了存储空间,因此寄存器寻址在计算机中得到广泛应用。

6. 寄存器间接寻址

图7.14示意了寄存器间接寻址过程。

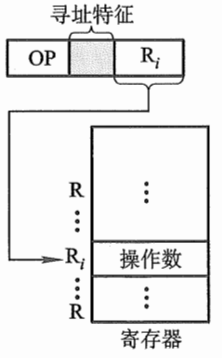


图 7.13 寄存器寻址示意图

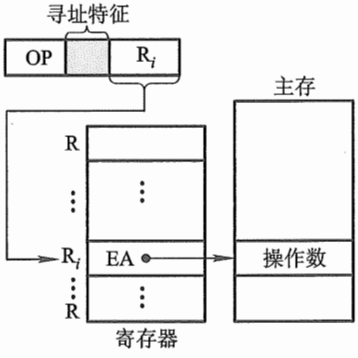


图 7.14 寄存器间接寻址示意图

图中 R_i 中的内容不是操作数,而是操作数所在主存单元的地址号,即有效地址 $EA=(R_i)$。与寄存器寻址相比,指令的执行阶段还需访问主存。与图7.11(a)相比,因有效地址不是存放在

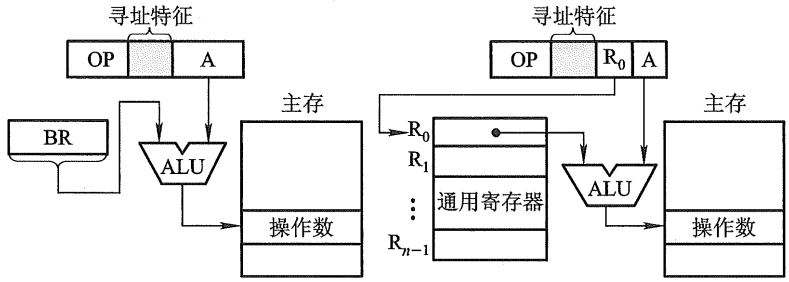
存储单元中，而是存放在寄存器中，故称其为寄存器间接寻址，它比间接寻址少访存一次。

7. 基址寻址

基址寻址需设有基址寄存器 BR，其操作数的有效地址 EA 等于指令字中的形式地址与基址寄存器中的内容（称为基地址）相加，即

$$EA = A + (BR)$$

图 7.15 示意了基址寻址过程。



（a）专用基址寄存器 BR　　（b）通用寄存器作基址寄存器

图 7.15　基址寻址示意图

基址寄存器可采用隐式的和显式的两种。所谓隐式，是在计算机内专门设有一个基址寄存器 BR，使用时用户不必明显指出该基址寄存器，只需由指令的寻址特征位反映出基址寻址即可。显式是在一组通用寄存器里，由用户明确指出哪个寄存器用作基址寄存器，存放基地址。例如，IBM 370 计算机中设有 16 个通用寄存器，用户可任意选中某个寄存器作为基址寄存器。对应图 7.15（a）为隐式基址寻址，图 7.15（b）为显式基址寻址。

基址寻址可以扩大操作数的寻址范围，因基址寄存器的位数可以大于形式地址 A 的位数。当主存容量较大时，若采用直接寻址，因受 A 的位数限制，无法对主存所有单元进行访问，但采用基址寻址便可实现对主存空间的更大范围寻访。例如，将主存空间分为若干段，每段首地址存于基址寄存器中，段内的位移量由指令字中形式地址 A 指出，这样操作数的有效地址就等于基址寄存器内容与段内位移量之和，只要对基址寄存器的内容做修改，便可访问主存的任一单元。

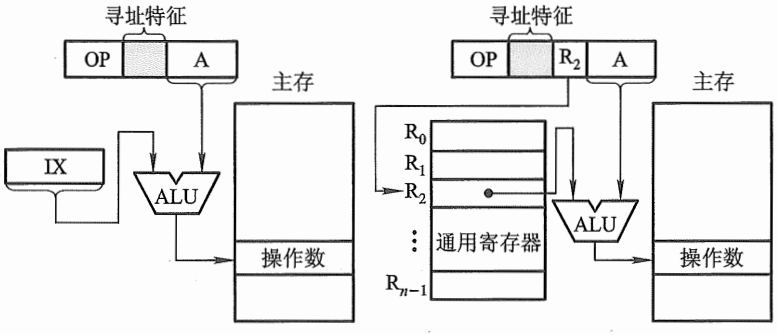
基址寻址在多道程序中极为有用。用户可不必考虑自己的程序存于主存的哪一空间区域，完全可由操作系统或管理程序根据主存的使用状况，赋予基址寄存器内一个初始值（即基地址），便可将用户程序的逻辑地址转化为主存的物理地址（实际地址），把用户程序安置于主存的某一空间区域。例如，对于一个具有多个寄存器的机器来说，用户只需指出哪一个寄存器作为基址寄存器即可，至于这个基址寄存器应赋予何值，完全由操作系统或管理程序根据主存空间状况来确定。在程序执行过程中，用户不知道自己的程序在主存的哪个空间，用户也不可修改基址寄存器的内容，以确保系统安全可靠地运行。

8. 变址寻址

变址寻址与基址寻址极为相似。其有效地址 EA 等于指令字中的形式地址 A 与变址寄存器 IX 的内容相加之和，即

$$EA = A + (IX)$$

显然只要变址寄存器位数足够，也可扩大操作数的寻址范围，其寻址过程如图 7.16 所示。



(a) 专用变址寄存器 IX　　(b) 通用寄存器作变址寄存器

图 7.16　变址寻址示意图

图 7.16(a)、(b)与图 7.15(a)、(b)相比，显见变址寻址与基址寻址的有效地址形成过程极为相似。由于两者的应用场合不同，因此从本质来认识，它们还是有较大的区别。基址寻址主要用于为程序或数据分配存储空间，故基址寄存器的内容通常由操作系统或管理程序确定，在程序的执行过程中其值是不可变的，而指令字中的 A 是可变的。在变址寻址中，变址寄存器的内容是由用户设定的，在程序执行过程中其值可变，而指令字中的 A 是不可变的。变址寻址主要用于处理数组问题，在数组处理过程中，可设定 A 为数组的首地址，不断改变变址寄存器 IX 的内容，便可很容易形成数组中任一数据的地址，特别适合编制循环程序。例如，某数组有 N 个数存放在以 D 为首地址的主存一段空间内。如果求 N 个数的平均值，则用直接寻址方式很容易完成程序的编制。表 7.1 列出了用直接寻址求 N 个数平均值的程序。

表 7.1　直接寻址求 N 个数的平均值程序

程　　序	说　　明
LDA D	[D]→ACC
ADD D+1	[ACC]+[D+1]→ACC
ADD D+2	[ACC]+[D+2]→ACC
⋮	⋮
ADD D+(N−1)	[ACC]+[D+(N−1)]→ACC
DIV #N	[ACC]÷N→ACC
STA ANS	[ACC]→ANS 单元(ANS 为主存某单元地址)

显然，当 $N=100$ 时，该程序用了 102 条指令，除数据外，共占用 102 个存储单元存放指令。而且随 N 的增加，程序所用的指令数也增加（共 $N+2$ 条）。

若用变址寻址，则只要改变变址寄存器的内容，而保持指令“ADD X，D”（X 为变址寄存器，D 为形式地址）不变，便可依次完成 N 个数相加。用变址寻址编制的程序如表 7.2 所示。

表 7.2　变址寻址求 N 个数的平均值程序

程　序	说　明
LDA #0	0→ACC
LDX #0	0→X（X 为变址寄存器）
M ADD X，D	[ACC] +[D +(X)]→ACC（D 为形式地址，X 为变址寄存器）
INX	[X] +1→X
CPX #N	[X] −N，并建立 Z 的状态，结果为“0”，Z = 1；结果非“0”，Z = 0
BNE M	当 Z = 1 时，按顺序执行；当 Z = 0 时，转至 M
DIV #N	[ACC] ÷N→ACC
STA ANS	[ACC]→ANS（ANS 为主存某单元地址）

该程序仅用了 8 条指令，而且随 N 的增加，指令数不变，指令所占的存储单元大大减少。

有的机器（如 Intel 8086、VAX-11）的变址寻址具有自动变址的功能，即每存取一个数据，根据数据长度（即所占字节数），变址寄存器能自动增量或减量，以便形成下一个数据的地址。

变址寻址还可以与其他寻址方式结合使用。例如，变址寻址可与基址寻址合用，此时有效地址 EA 等于指令字中的形式地址 A 和变址寄存器 IX 的内容（IX）及基址寄存器 BR 中的内容（BR）相加之和，即

$$EA=A+(IX)+(BR)$$

变址寻址还可与间接寻址合用，形成先变址后间址或先间址再变址等寻址方式，读者在使用各类机器时可注意分析。

9. 相对寻址

相对寻址的有效地址是将程序计数器 PC 的内容（即当前指令的地址）与指令字中的形式地址 A 相加而成，即

$$EA=(PC)+A$$

图 7.17 示意了相对寻址的过程，由图中可见，操作数的位置与当前指令的位置有一段距离 A。

相对寻址常被用于转移类指令，转移后的目标地址与当前指令有一段距离，称为相对位移量，它由指令字的形式地址 A 给出，故 A 又称位移量。位移量 A 可正可负，通常用补码表示。倘若位移量为 8 位，则指令的寻址范围在（PC）+127～（PC）−128 之间。

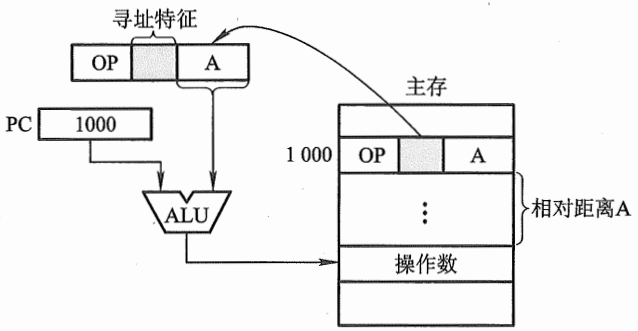


图 7.17　相对寻址示意图

相对寻址的最大特点是转移地址不固定,它可随 PC 值的变化而变,因此,无论程序在主存的哪段区域,都可正确运行,对于编写浮动程序特别有利。例如,表 7.2 中有一条转移指令“BNE M”,它存于 M+3 单元内,也即

```
⋮
M        ADD   X,D
M+1      INX
M+2      CPX   #N
M+3      BNE   M
```

显然,随程序首地址改变,M 也改变。如果采用相对寻址,将“BNE M”改写为“BNE ＊-3”(＊为相对寻址特征),就可使该程序浮动至任一地址空间都能正常运行。因为从第 M+3 条指令转至第 M 条指令,其相对位移量为-3,故当执行第 M+3 条指令“BNE ＊-3”时,其有效地址为

$$EA=(PC)+(-3)=M+3-3=M$$

直接指向了转移后的目标地址。

相对寻址也可与间接寻址配合使用。

例 7.2　设相对寻址的转移指令占 3 个字节,第一字节为操作码,第二、三字节为相对位移量(补码表示),而且数据在存储器中采用以低字节地址为字地址的存放方式。每当 CPU 从存储器取出一个字节时,即自动完成(PC)+1→PC。

(1) 若 PC 当前值为 240(十进制),要求转移到 290(十进制),则转移指令的第二、三字节的机器代码是什么?

(2) 若 PC 当前值为 240(十进制),要求转移到 200(十进制),则转移指令的第二、三字节的机器代码是什么?

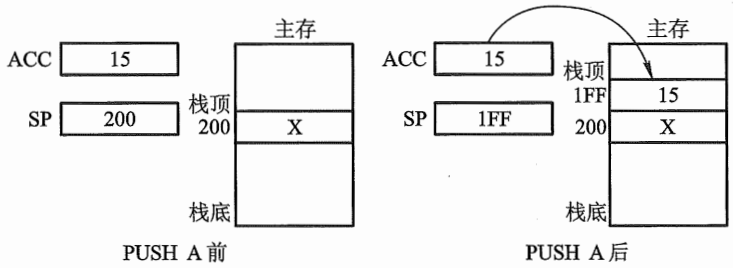
解:(1) PC 当前值为 240,该指令取出后 PC 值为 243,要求转移到 290,即相对位移量为 290-243=47,转换成补码为 2FH。由于数据在存储器中采用以低字节地址为字地址的存放方

式,故该转移指令的第二字节为 2FH,第三字节为 00H。

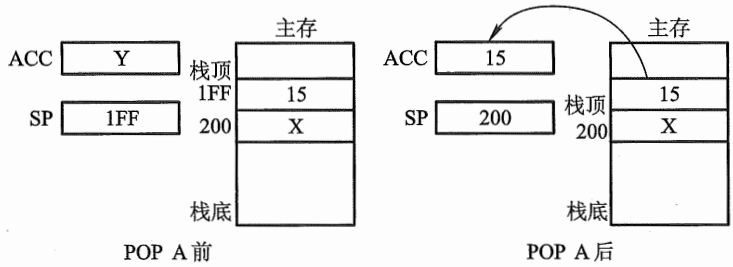
(2) PC 当前值为 240,该指令取出后 PC 值为 243,要求转移到 200,即相对位移量为 200-243=-43,转换成补码为 D5H,由于数据在存储器中采用以低字节地址为字地址的存放方式,故该转移指令的第二字节为 D5H,第三字节为 FFH。

10. 堆栈寻址

堆栈寻址要求计算机中设有堆栈。堆栈既可用寄存器组(称为硬堆栈)来实现,也可利用主存的一部分空间作堆栈(称为软堆栈)。堆栈的运行方式为先进后出或先进先出两种,先进后出型堆栈的操作数只能从一个口进行读或写。以软堆栈为例,可用堆栈指针 SP(Stack Point)指出栈顶地址,也可用 CPU 中一个或两个寄存器作为 SP。操作数只能从栈顶地址指示的存储单元存或取。可见堆栈寻址也可视为一种隐含寻址,其操作数的地址总被隐含在 SP 中。堆栈寻址就其本质也可视为寄存器间接寻址,因 SP 可视为寄存器,它存放着操作数的有效地址。图 7.18 示意了堆栈寻址过程。



(a) PUSH A 示意



(b) POP A 示意

图 7.18　堆栈寻址示意图

图 7.18(a)、(b)分别表示进栈“PUSH A”和出栈“POP A”的过程。

由于 SP 始终指示着栈顶地址,因此不论是执行进栈(PUSH),还是出栈(POP),SP 的内容都需要发生变化。若栈底地址大于栈顶地址,则每次进栈(SP)-Δ→SP;每次出栈(SP)+Δ→SP。Δ 取值与主存编址方式有关。若按字编址,则 Δ 取 1(如图 7.18 所示);若按字节编址,则需根据存储字长是几个字节构成才能确定 Δ,例如字长为 16 位,则 Δ=2,字

长为32位，Δ=4。

例7.3 一条双字长直接寻址的子程序调用指令，其第一个字为操作码和寻址特征，第二个字为地址码5000H。假设PC当前值为2000H，SP的内容为0100H，栈顶内容为2746H，存储器按字节编址，而且进栈操作是先执行(SP)-Δ→SP，后存入数据。试回答下列几种情况下，PC、SP及栈顶内容各为多少？

(1) CALL指令被读取前。

(2) CALL指令被执行后。

(3) 子程序返回后。

解：(1) CALL指令被读取前，PC=2000H，SP=0100H，栈顶内容为2746H。

(2) CALL指令被执行后，由于存储器按字节编址，CALL指令共占4个字节，故程序断点2004H进栈，此时SP=(SP)-2=00FEH，栈顶内容为2004H，PC被更新为子程序入口地址5000H。

(3) 子程序返回后，程序断点出栈，PC=2004H，SP被修改为0100H，栈顶内容为2746H。

由于当前计算机种类繁多，各类机器的寻址方式均有各自的特点，还有些机器的寻址方式可能本书并未提到，故读者在使用时需自行分析，以利于编程。

从高级语言角度考虑问题，机器指令的寻址方式对用户无关紧要，但一旦采用汇编语言编程，用户只有了解并掌握机器的寻址方式，才能正确编程，否则程序将无法正常运行。如果读者参与机器的指令系统设计，则了解寻址方式对确定机器指令格式是不可缺少的。从另一角度来看，倘若透彻了解了机器指令的寻址方式，将会使读者进一步加深对机器内信息流程及整机工作概念的理解。

7.4 指令格式举例

指令格式不仅体现了指令系统的各种功能，而且也突出地反映了机器的硬件结构特点。设计指令格式时必须从诸多方面综合考虑，并经一段模拟运行后，最后确定。

7.4.1 设计指令格式应考虑的各种因素

指令系统集中反映了机器的性能，又是程序员编程的依据。用户在编程时既希望指令系统很丰富，便于用户选择，同时还要求机器执行程序时速度快、占用主存空间少，实现高效运行。此外，为了继承已有的软件，必须考虑新机器的指令系统与同一系列机器指令系统的兼容性，即高档机必须能兼容低档机的程序运行，称之为“向上兼容”。

指令格式集中体现了指令系统的功能，为此，在确定指令格式时，必须从以下几个方面综合考虑。

① 操作类型：包括指令数及操作的难易程度。

② 数据类型：确定哪些数据类型可以参与操作。

③ 指令格式：包括指令字长、操作码位数、地址码位数、地址个数、寻址方式类型，以及指令字长和操作码位数是否可变等。

④ 寻址方式：包括指令和操作数具体有哪些寻址方式。

⑤ 寄存器个数：寄存器的多少直接影响指令的执行时间。

7.4.2　指令格式举例

不同机器的指令格式可以有很大的差别，本书不可能将各种机器的指令格式都做介绍，只能列举几种较为典型的格式供读者学习。

1. PDP-8

PDP-8 的指令字长统一为 12 位，CPU 内只设一个通用寄存器，即累加器 ACC，其主存被划分为若干个容量相等的存储空间（每个相同的空间被称为一页）。该机的指令格式可分为三大类，如图 7.19 所示。

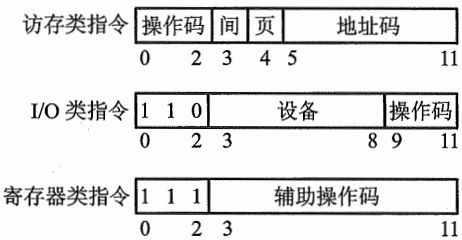


图 7.19　PDP-8 指令格式

访存类指令属一地址指令。0～2 位为操作码（只定义了 000～101 六种基本操作）；3、4 两位为寻址特征位，其中 3 位表示是否间接寻址，4 位表示是当前页面（即 PC 指示的页面）还是 0 页面；5～11 位为地址码。

为了扩大操作种类，对应操作码“111”又配置了辅助操作码，构成了寄存器类指令，这类指令主要对 ACC 进行各种操作，如清 A、对 A 取反、对 A 移位、对 A 加 1、根据 A 的结果是否跳转等。辅助操作码的每一位都有一个明确的操作。

第三类指令是 I/O 类，用 0～2 位为 110 作标志，其具体操作内容由 9～11 位反映，3～8 位表示设备号，总共可选 64 种设备。

PDP-8 指令格式支持间接寻址、变址寻址、相对寻址。加上操作码扩展技术，共有 35 条指令。

2. PDP-11

PDP-11 机器字长为 16 位，CPU 内设 8 个 16 位通用寄存器，其中两个通用寄存器有特殊作用，一个用作堆栈指针 SP，一个用作程序计数器 PC。

PDP-11 指令字长有 16 位、32 位和 48 位三种，采用操作码扩展技术，使操作码位数不固定，指令字的地址格式有零地址、一地址、二地址等共有 13 类指令格式，图 7.20 列出了其中五种。

图中（a）为零地址格式；（b）为一地址格式，其中 6 位目的地址码中的 3 位为寻址特征位，另外 3 位表示 8 个寄存器中的任一个；（c）、（d）、（e）均为二地址格式指令，但操作数来源不同，有

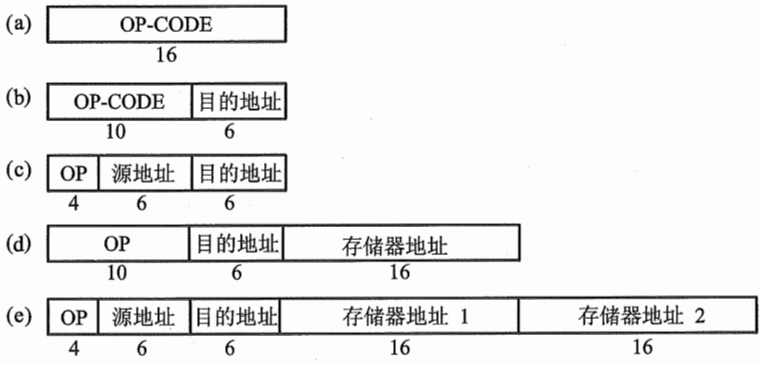


图 7.20　PDP-11 五种指令格式

寄存器-寄存器型、寄存器-存储器型和存储器-存储器型。

PDP-11 指令系统和寻址方式比较复杂，既增加了硬件的价格，又增加了编程的复杂度，但好处是能编出非常高效的程序。

3. IBM 360

IBM 360 属于系列机。所谓系列机，是指其基本指令系统相同，基本体系结构相同的一系列计算机。IBM 370 对 IBM 360 是完全向上兼容的。所以 IBM 370 可看作 IBM 360 的扩展、延伸或改进。

IBM 360 是 32 位机器，按字节寻址，并可支持多种数据类型，如字节、半字、字、双字（双精度实数）、压缩十进制数、字符串等。在 CPU 中有 16 个 32 位通用寄存器（用户可选定任一个寄存器作为基址寄存器 BR 或变址寄存器 IX），4 个双精度（64 位）浮点寄存器。指令字长有 16 位、32 位、48 位三种，如图 7.21 所示。

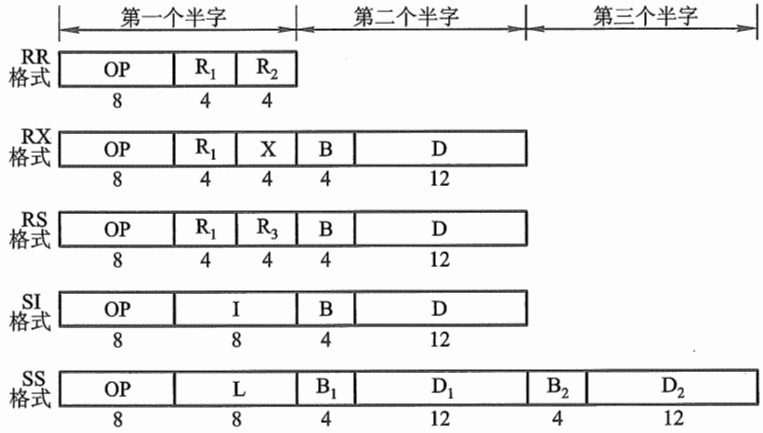


图 7.21　IBM 360/370 指令格式

图中共画出了五种指令格式,它们的操作码位数均为8位。RR格式是寄存器-寄存器格式,两个操作数均在寄存器中,完成 (R_1) OP $(R_2)\rightarrow R_1$ 的操作。RX是二地址格式的寄存器-存储器型指令,一个操作数在寄存器中,另一个操作数在存储器中,其有效地址由变址(X)和基址(B)寻址方式求得,可以完成 (R_1) OP M $[(X)+(B)+D]\rightarrow R_1$ 的操作。RS格式是三地址格式的寄存器-存储器型指令,完成 (R_3) OP M $[(B)+D]\rightarrow R_1$ 操作。SI格式中的I为立即数,它完成立即数→M[(B)+D]的操作。SS格式是存储器-存储器型指令,两个操作数均在存储器,这类指令用于十进制运算和字符串处理,数据长度字段L可定义一个长度(1~256个字符)或两个长度(每一个为1~16个十进制数),它完成 M $[(B_1)+D_1]$ OP M $[(B_2)+D_2]$ $\rightarrow$M$[(B_1)+D_1]$ 的操作。

4. Intel 8086/80486系列机

Intel 8086/80486系列微型计算机的指令字长为1~6个字节,即不定长。例如,零地址格式的空操作指令NOP只占一个字节;一地址格式的CALL指令可以是3字节(段内调用)或5字节(段间调用);二地址格式指令中的两个操作数既可以是寄存器-寄存器型、寄存器-存储器型,也可以是寄存器-立即数型或存储器-立即数型,它们所占的字节数分别为2、2~4、2~3、3~6个字节。有关该系列机的指令格式,读者可以查阅有关资料自行分析。

7.4.3 指令格式设计举例

例7.4 某机字长16位,存储器直接寻址空间为128字,变址时的位移量为-64~+63,16个通用寄存器均可作为变址寄存器。设计一套指令系统格式,满足下列寻址类型的要求。

(1) 直接寻址的二地址指令3条。

(2) 变址寻址的一地址指令6条。

(3) 寄存器寻址的二地址指令8条。

(4) 直接寻址的一地址指令12条。

(5) 零地址指令32条。

试问还有多少种代码未用?若安排寄存器寻址的一地址指令,还能容纳多少条?

解:(1) 在直接寻址的二地址指令中,根据题目给出直接寻址空间为128字,则每个地址码为7位,其格式如图7.22(a)所示。3条这种指令的操作码为00、01和10,剩下的11可作为下一种格式指令的操作码扩展用。

(2) 在变址寻址的一地址指令中,根据变址时的位移量为-64~+63,形式地址A取7位。根据16个通用寄存器可作为变址寄存器,取4位作为变址寄存器 R_x 的编号。剩下的5位可作操作码,其格式如图7.22(b)所示。6条这种指令的操作码为11000~11101,剩下的两个编码11110和11111可作为扩展用。

(3) 在寄存器寻址的二地址指令中,两个寄存器地址 R_i 和 R_j 共8位,剩下的8位可作操作码,比格式(b)的操作码扩展了3位,其格式如图7.22(c)所示。8条这种指令的操作码为

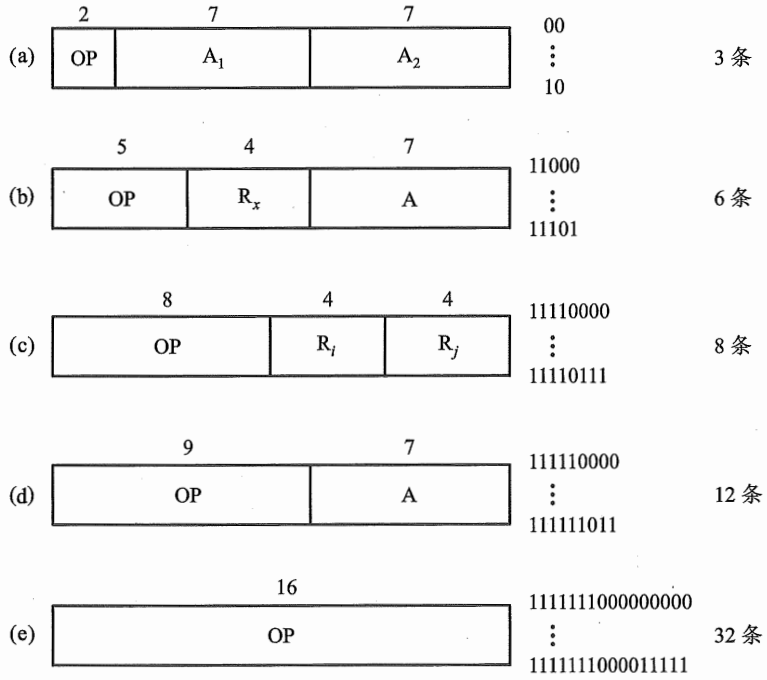


图 7.22 例 7.4 五种指令格式

11110000~11110111。剩下的 11111000~11111111 这 8 个编码可作为扩展用。

(4) 在直接寻址的一地址指令中,除去 7 位的地址码外,可有 9 位操作码,比格式(c)的操作码扩展了 1 位,与格式(c)剩下的 8 个编码组合,可构成 16 个 9 位编码。以 11111 作为格式(d)指令的操作码特征位,12 条这种指令的操作码为 111110000~111111011,如图 7.22(d)所示。剩下的 111111100~111111111 可作为扩展用。

(5) 在零地址指令中,指令的 16 位都作为操作码,比格式(d)的操作码扩展了 7 位,与上述剩下的 4 个操作码组合后,共可构成 4×2^7 条指令的操作码。32 条这种指令的操作码可取 1111111000000000~1111111000011111,如图 7.22(e)所示。

还有 $2^9-32=480$ 种代码未用,若安排寄存器寻址的一地址指令,除去末 4 位为寄存器地址外,还可容纳 30 条这类指令。

例 7.5 设某机配有基址寄存器和变址寄存器,采用一地址格式的指令系统,允许直接和间接寻址,且指令字长、机器字长和存储字长均为 16 位。

(1) 若采用单字长指令,共能完成 105 种操作,则指令可直接寻址的范围是多少?一次间接寻址的寻址范围是多少?画出其指令格式并说明各字段的含义。

(2) 若存储字长不变,可采用什么方法直接访问容量为 16 MB 的主存?

解:(1) 在单字长指令中,根据能完成 105 种操作,取操作码 7 位。因允许直接和间接寻址,

且有基址寄存器和变址寄存器，故取 2 位寻址特征位，其指令格式如下：

7	2	7
OP	M	AD

其中，OP 为操作码，可完成 105 种操作；M 为寻址特征，可反映四种寻址方式；AD 为形式地址。

这种指令格式可直接寻址 $2^7=128$，一次间接寻址的寻址范围是 $2^{16}=65\ 536$。

（2）容量为 16 MB 的存储器，正好与存储字长为 16 位的 8 M 存储器容量相等，即 16 MB = 8 M×16 位。欲使指令直接访问 16 MB 的主存，可采用双字长指令，其操作码和寻址特征位均不变，其格式如下：

7	2	7
OP	M	AD_1
AD_2		

其中，形式地址为 $AD_1 // AD_2$，共 7+16=23 位。$2^{23}=8$ M，即可直接访问主存的任一位置。

例 7.6　某模型机共有 64 种操作，操作码位数固定，且具有以下特点。

（1）采用一地址或二地址格式。

（2）有寄存器寻址、直接寻址和相对寻址（位移量为-128～+127）三种寻址方式。

（3）有 16 个通用寄存器，算术运算和逻辑运算的操作数均在寄存器中，结果也在寄存器中。

（4）取数/存数指令在通用寄存器和存储器之间传送数据。

（5）存储器容量为 1 MB，按字节编址。

要求设计算逻指令、取数/存数指令和相对转移指令的格式，并简述理由。

解：（1）算逻指令格式为寄存器-寄存器型，取单字长 16 位。

6	2	4	4
OP	M	R_i	R_j

其中，OP 为操作码，6 位，可实现 64 种操作；M 为寻址模式，2 位，可反映寄存器寻址、直接寻址、相对寻址；R_i 和 R_j 各取 4 位，指出源操作数和目的操作数的寄存器（共 16 个）编号。

（2）取数/存数指令格式为寄存器-存储器型，取双字长 32 位，格式如下：

6	2	4	4
OP	M	R_i	A_1
A_2			

其中，OP 为操作码，6 位不变；M 为寻址模式，2 位不变；R_i 为 4 位，源操作数地址（存数指令）或目的操作数地址（取数指令）；A_1 和 A_2 共 20 位，为存储器地址，可直接访问按字节编址的 1 MB 存储器。

（3）相对转移指令为一地址格式，取单字长 16 位，格式如下：