Chapter 5

第5章　指令系统

计算机的指令是程序人员和计算机交往的最基本的界面，是机器指令的简称。机器指令是人们指示计算机执行各种基本操作的命令。一般来说，指令包括操作码及地址码两部分，用二进制编码表示。操作码用来表示各种不同的操作，如加、减、乘、除等。不同指令的操作码是不同的。操作码的位数，反映一台机器允许的不同指令的个数。如5位操作码，可以有$2^5=32$个编码，最多允许设置32种指令；7位操作码，最多有$2^7=128$个编码，最多允许设置128种指令。指令种类的多少，反映机器功能的强弱，但增加指令种类伴随而来的是机器结构愈加复杂。地址码指出被运算数据在存储器中存放的位置。被运算数据，一般都放在计算机的主存储器中。因此，指令地址码通常给出的是被运算数据的主存地址编号。地址位数反映CPU可直接访问的主存的容量，如地址码是11位二进制数，则可直接访问的主存最大容量是$2^{11}=2048$个地址单元，简称2K个单元；若地址码是15位，最大的主存容量是$2^{15}=32\ 768$个单元，简称32K个单元。

因为指令字的位数有限，通常不直接给出要访问的主存单元地址，而是通过地址变换技术，扩充可以访问的主存容量。通过指令中给出的地址码及地址变换方式字段，寻找操作数在主存中存放的真实地址，这种方法叫寻址方式。寻址方式不但扩大了可以访问的存储器容量，而且还可以为满足用户提出的各种方便编写程序的需要。

指令系统是一台机器所能执行的全部指令。

确定机器指令系统是计算机设计时的大事，一方面机器设计要满足用户的需要，方便编制程序；另一方面要节省设备，简化硬件结构，这对于机器的造价、维护、可靠性等方面都有好处。

由于计算机发展很快，各种型号的计算机层出不穷，其性能有了很大提高，但是由于指令系统不兼容给用户更新机器带来很大不便。因此提出了计算机升级换代的系列化问题，特别是指令系统兼容问题，这在计算机的推广应用及设计生产时都值得特别重视。

5.1　指令格式

计算机能把计算程序存放在存储器中，并按规定顺序执行有关操作，是近代计算机广泛应用的关键，这种“存储程序”的概念是冯·诺依曼1945年提出的。

实现“存储程序”的关键是用二进制代码表示指令。其中,一部分二进制代码表示操作种类的性质,另一部分二进制代码表示参加操作的数据在存储器中存放的位置。

从主存看,指令字表面上与数据字没有什么不同,但作为指令的二进制代码与数据字有着根本不同的含义。主存单元作为指令字或是数据字是由指令计数器及指令中地址码分别给出的。由指令计数器给出地址,从主存单元中读出指令字,读出的内容送给控制器中的指令寄存器进行分析。根据指令中地址码决定的主存有效地址,从主存单元读出操作数,读出的内容送到运算器中的数据寄存器中等待运算器处理。

5.1.1 指令字

指令字的位数叫指令字的长度。它通常与数据字的长度有关,也就是说与每个主存单元存放的数据位数有关。每个主存单元,可以用来存放一个操作数,也可用来存放一条指令,选取指令字长与数据字长相同,可以充分利用主存单元每个信息位,另外也便于存放。如果指令字长大于数据字长,则取一条指令,要访问几次主存,造成控制线路复杂,也降低了执行一条指令的速度。因此大多数机器选取二者等长。但在8位字长的微机中,除8位指令外,还设有多倍字长的指令,满足长指令字的需要。

在一些大型计算机中,数据字较长,如48位、64位等,而指令字较短,若一个主存单元中存放两条指令或三条指令,可以充分利用主存单元中每个信息位。读指令时,一次读出多条指令,供控制器依次分析处理,这样做还可以提高指令的执行速度。大型计算机指令系统复杂,指令功能差别较大。不同类型的指令字长是不同的。这样可以更好提高主存单元利用率。例如IBM 360/370计算机字长是32位,两个操作数的指令字长有16位、32位、48位三种,分别对应于寄存器——寄存器型指令,寄存器——存储器型指令,以及存储器——存储器型指令。

一般指令中应包括以下信息:

(1) 操作的种类和性质,称为操作码。

(2) 操作数的存放地址,在双操作数运算中,如加、减、乘、除、逻辑乘、逻辑加的运算中都需要指定两个操作数,给出二个操作数地址。

(3) 操作结果存放地址。

(4) 下条指令存放地址。这样可以保证程序能连续不断地执行下去,直到程序结束。

指令中用不同的代码段表示上述不同信息,这种代码段的划分和含义,就是指令的编码方式,又叫指令格式,通常一条指令中包括操作码字段和若干个地址码字段。

操作码字段规定操作的种类、性质、指令字长度、操作数个数和指令格式等,一般放在指令的第一个字段。

存放操作数位置的字段,称为操作数地址码,一般操作数可以放在通用寄存器或主存中,因此,操作数地址码可以是寄存器号或主存单元地址。同理,存放操作结果位置的字段,也可以是寄存器号或主存单元地址。

一般情况下,程序中各条指令是顺序执行的。下条指令放在当前指令后一单元中,也

就是下条指令地址是当前指令地址加1,多数计算机中都设置程序计数器PC,指明本条指令的主存存放地址,本条指令结束时,PC+1给出下条指令地址。这样在指令中可以省去一个地址字段,缩短指令字长度。如果当前指令占用多个存储单元,则下条指令地址为PC加上当前指令占用的主存单元数。

典型的指令格式如图5-1所示。

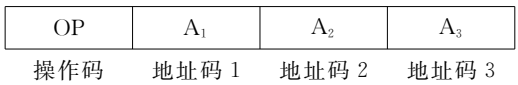

图5-1 典型指令格式

这种指令操作的含义是:主存地址A_1单元中的数据与主存地址A_2单元中的数据,执行OP规定的操作,运算结果放到主存地址A_3的单元中。

根据指令中包含地址码的数目,指令可分为以下几种格式。

(1) 三地址指令,指令格式与图5-1相同。

OP为操作码,并且表示按三地址指令格式解释指令;

A_1 为第一源操作数地址;

A_2 为第二源操作数地址;

A_3 为运算结果存放地址。

该指令的含义是:$(A_1)OP(A_2)\rightarrow A_3$。

这里需要说明:A_1表示主存单元的地址,(A_1)表示主存单元存放的数据,是A_1单元的内容。

三地址指令字长较长,程序在主存中占用单元数较多,并且执行一条指令最多要访问存储器四次,即取指令,取第一源操作数,取第二源操作数,结果送往主存单元。因此,执行一条指令的时间也较长。

(2) 二地址指令,指令格式如图5-2所示。

为了缩短指令字长度,可把存放运算结果的地址码省去,规定第二源操作数地址一身二用,即不但作为第二源操作数地址,还兼作运算结果存放的主存地址。

二地址指令含义:$(A_1)OP(A_2)\rightarrow A_2$。

二地址指令表达意义清楚,指令字长较短,但执行结果删去了一个操作数,有时也带来不便。

(3) 一地址指令,指令格式如图5-3所示。

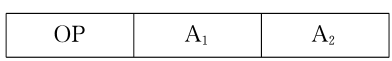

图5-2 二地址指令格式

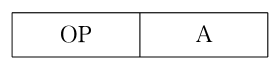

图5-3 一地址指令格式

一地址指令只给出一个操作数地址,另外一个操作数地址,隐含规定为累加器AC。在单累加器结构的计算机中,都这样处理。

一地址指令的含义是:$(AC)OP(A)\rightarrow AC$。

累加器的名称也由此而来。每一条指令运算结果都放在累加器中,当然累加器也提

供一个操作数。这种指令的优点是指令字长短，访问主存次数少，指令执行时间短应用非常广泛。缺点是上条指令运算结果还有用时，需另外保存。

(4) 寄存器型指令　指令中地址码不是主存地址，而是运算器中数据寄存器编号。

这种指令的优点是指令字长短、访问主存次数更少、指令执行时间更短，因此应用很广，特别是二个操作数都在寄存器中，除了取指令外，一个时钟周期即可得到运算结果。在新型的精简指令计算机系统中，广泛采用寄存器型指令。

(5) 有些指令不需要指出操作数地址，如堆栈指令，由栈顶两个单元的数相操作，操作结果仍放在堆栈中，这种指令格式称为零地址指令。

还有一类指令，根本不需要操作数，常用于控制操作，如停机、空操作等。

在一些大型机中，还设置向量指令、字符串处理指令等，它们所需的地址字段会更多一些。

5.1.2 指令操作码及其扩展技术

指令操作码有两种编码格式，常用的是固定格式，操作码长度固定不变，放在一个字段中，若操作码为 k 位二进制数，则它最多只能有 2^k 种不同指令。这种格式译码方便，控制简单，在大中型计算机中采用。缺点是扩充性差、不灵活。如 IBM 360/370 计算机、VAX-11 系列计算机字长为 32 位，操作码字段占 8 位，可表示 256 种不同指令。

另一种格式为可变长度操作码格式，各种指令的操作码位数不同，能够在不长的指令字中灵活设置很多种指令，满足不同类型指令的需要，因此能够有效缩短操作码的平均长度，增加新的指令也比较方便。但这种操作码译码不方便，而且指令格式也不规整。

【例 5-1】 设某台机器有指令 128 种，用二种操作码编码方案：(1)用固定长度操作码方案设计其操作码编码；(2)如果在 128 种指令中常用指令有 8 种，使用概率达到 80%，其余指令使用概率为 20%，采用可变长操作码编码方案设计其编码，并求出其操作码平均长度。

解：

(1) 采用固定长操作码编码方案，需要 7 位操作码，$2^7=128$，可有 128 种编码，每种编码表示一种指令。

(2) 采用可变长操作码编码方案，用 4 位代码表示 8 种常用指令，用 8 位代码表示 120 种不常用指令，具体操作码分配如下：

0000 表示指令 0 的操作码
0001 表示指令 1 的操作码
⋮
0111 表示指令 7 的操作码
1000 0000 表示指令 8 的操作码
1000 0001 表示指令 9 的操作码
⋮
1000 1111 表示指令 23 的操作码

1001 0000 表示指令 24 的操作码

⋮

1001 1111 表示指令 39 的操作码

⋮

1110 1111 表示指令 119 的操作码

1111 0000 表示指令 120 的操作码

⋮

1111 0111 表示指令 127 的操作码

指令操作码的平均长度为

$$4\text{ 位}\times 80\%+8\text{ 位}\times 20\%=4.8\text{ 位}$$

可见使用可变长操作码可以有效缩短操作码平均长度。从指令的扩展性看,可变长操作码提供这种支持,特别是系列计算机设计中,为了保持软件的兼容性,希望保留先前机器指令的编码,而新增指令的操作码与以前指令的操作码又不相同。在上例中,7 位固定长操作码已经不能扩展新的指令,而 8 位操作码方案剩下的编码种类也是有限的,如利用可变字长操作码方案,把操作码扩展为 12 位,显然 $8\times 2^4=128$,则又可扩展 128 种指令。例如 PDP-11 计算机指令操作码采用可变长度方案,其字长为 16 位,操作码长度有 4 位、7 位、8 位、13 位和 16 位几种。

5.1.3 地址码与数据字长

地址码用来指定操作数的地址。地址码可以是存储器地址,也可以是通用寄存器号。地址码的编码方法很多,形成操作数有效地址的方法各不相同,我们把寻找操作数有效地址的方式称为寻址方式,在 5.2 节中专门介绍。

指令中需要的操作数的个数不同,根据操作数的个数,可以设置不同数目的地址码。操作码的设计应与地址码的设计相配合。例如地址码数目较多的指令中可安排较短的操作码,或者操作码较长的指令中安排的地址码是寄存器号等。

计算机中处理的数据字长,有时并不是固定的。会根据指令的不同要求而改变。如 IBM 360/370 计算机单字长指令处理数据是 32 位,双字长指令处理的数据是 64 位,有时还要处理 16 位半字数据,有时又要处理 8 位数据。字长为 8 位的数据对应 8 位的 ASCII 编码,在字符处理中非常有用。通常称为 1 字节。为了能读、写 1 个字节数据,必须对每一个字节单元进行编址,因此在 IBM 360/370 系列机以及 PDP-11 系列机中,都是对字节进行编址。但是 CPU 处理数据以及与主存交换数据,大多数情况都是以单字长形式进行的,每次访问主存,必须保证能够读出一个单字长数据,因此不同类型的数据在主存中的存放有专门规定。如 IBM 360/370 系列机器规定:主存的最小寻址单位为字节,每个字节有一个地址编号,字节地址是任意的二进制数。2 个字节组成 16 位的半字,半字的地址为 2 的倍数,即地址的最低位为“0”,由本地址指出的字节与下一地址字节(本字节地址+1)组成 16 位半字,也可称双字节数据。

单字长数据由 4 个字节组成,共 32 位。其地址为 4 的倍数,即字地址的最低两位为

“00”,本地址字节与紧跟着的 3 个字节一起组成一个 32 位的单字长数据。

双字长数据由 8 个字节组成,共 64 位,其地址为 8 的倍数,即地址的末 3 位为“000”,显然本地址字节与紧跟的 7 个字节组成一个双字,字节地址末 3 位为 000～111,共 8 个字节。

这种多字节数据地址编排格式称为数据边界对齐,其基本要求是一个数据字存放在一个完整的数据单元中,一次访存操作能够读出一个完整的数据字。或者说用最少的访存次数读取一个多字节数据。这就要求一个 4 字节长的数据必须放在一个主存的字单元中;一个 8 字节长的数据必须放在 2 个主存的字单元中;一个双字节数据必须放在 1 个字单元中,且必须放在该字单元的高 2 个字节或低 2 个字节,保证每个字单元可以存放 2 个双字节数据,如图 5-4 所示。

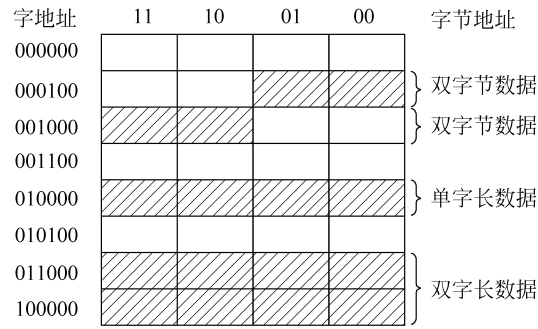

图 5-4　存储数据边界对齐示意图

当一个数据字或一个双字长数据字在主存单元中存放时,可以把低位字节数据放在字节地址小的字节内;也可按照相反次序存放,低位字节数据放在字节地址大的字节内,高位字节放在字节地址小的字节内。前者称小数端数据存储方式,后者称大数端数据存储方式。

5.2　寻址方式

在指令中,由于指令字长有限,且主存地址位数太多,通常在指令的地址码中不直接给出被运算数在主存中存放的地址,而是给出寻找操作数有效地址的编码方法和位移量,位移量有时又叫形式地址。寻找操作数有效地址的方式叫寻址方式。

设置不同寻址方式的目的,除了扩展可以访问的主存容量外,也是为了在程序中更加灵活地指定操作数的存放位置,特别是在重复执行的程序段中以及对某些数据结构类型进行操作时,可以不修改程序就完成对不同数据的操作。

5.2.1　存储器寻址方式

操作数在存储器中存放。经常使用的寻址方式有如下几种。

1. 直接寻址方式

指令地址码中直接给出操作数存放的有效地址,这种寻址方式叫做直接寻址方式。直接寻址中地址码的位数决定了可以访问的主存容量。这种方式适于访问固定的主存单元,如图 5-5 所示。

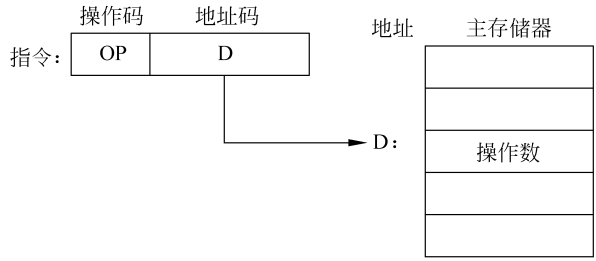

图 5-5 直接寻址方式

存放操作数的有效地址是 EA=D,D 单元的内容是操作数。

2. 间接寻址方式

指令地址码中给出的既不是操作数,也不是操作数地址,而是存放操作数地址的主存单元地址,访问一次主存后,才可得到操作数有效地址,这种方式称为一次间接寻址方式。存放操作数地址的寄存器或主存单元又叫地址指针。

指令中必须给出间接寻址的标志,以便与直接寻址等方式相区分。

其指令格式如图 5-6 格式。

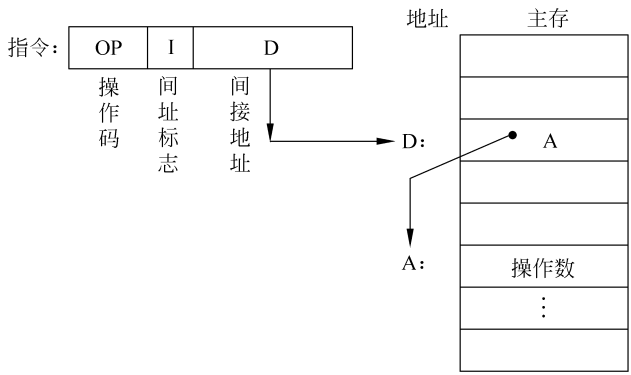

图 5-6 间接寻址方式

存放操作数的有效地址是 EA=(D)=A,即指令地址码指定单元 D 的内容是 EA,而 EA 单元的内容(主存 A 单元的内容)是操作数。

3. 变址寻址方式

指令中的地址码位数有限,为了扩大访存容量,常常采用变址寻址方式。规定指令中的地址码与另外一个寄存器的内容相加,得到的结果才是操作数的有效地址。显然操作数有效地址的位数取决于相加 2 个数中位数最长的数。如地址码 8 位,寄存器存放的数为 16 位,两数相加其和为 16 位数,有效地址位数应为 16 位数,显然扩大了可访问的存储

器的空间。

变址寻址方式的指令中应有变址寻址标志，指出有效地址码如何形成，同时应该设有专门的变址寄存器，其内容称为变址量，其指令格式如图 5-7 所示，其中

变址寄存器 R_X 中数据 K 称变址量；

操作数的有效地址：EA＝(R_X)＋D＝K＋D＝E，E 单元的内容是操作数。

有时变址寄存器可以有多个，由指令指定具体使用哪一个变址寄存器。

变址寻址方式适合于对一组数据进行访问，每访问一个数据元素后，只要改变变址值，就可用原来的指令访问下一个数据元素。

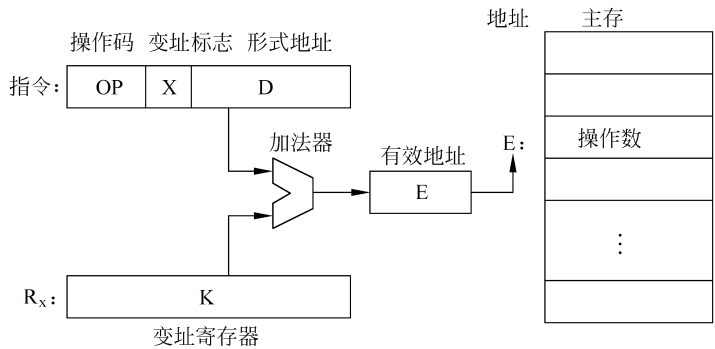

图 5-7　变址寻址方式

4. 相对寻址方式

相对寻址方式是一种特殊的变址寻址方式。当变址寄存器指定为程序计数器 PC 时的寻址方式称为相对寻址方式。

PC 的内容是当前执行指令的地址。相对寻址方式格式如图 5-8 所示。

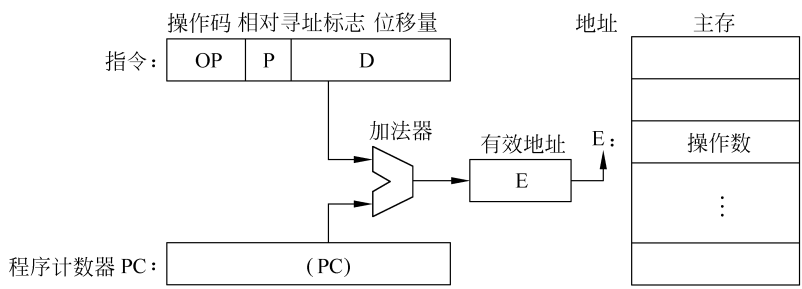

图 5-8　相对寻址方式

存放操作数有效地址是 EA＝(PC)＋D＝E，EA 单元的内容(即 E 单元的内容)是操作数。

指令中地址码部分 D 给出的叫位移量，它是一个固定的数值。当前指令的地址是(PC)单元，而指令使用的操作数放在距本条指令地址间隔为 D 的那个单元中。这种指令适合于程序搬家再定位时使用，因为不管指令搬到何处，其使用的操作数永远放在距本条指令地址为 D 的那个单元中。

5. 基址寻址方式

在多用户的计算机中,为每一个用户分配一个存储空间,每个用户按自己的逻辑地址编程。在程序装入机器中运行时,为每个用户指定一个基地址,用户程序实际存放的地址为基地址与其逻辑地址之和,不同用户使用不同的基地址,占用不同的存储空间。这种寻址方式称为基址寻址方式。在计算机中需设置专门的基址寄存器,其内容由系统指定。

基址寻址指令格式如图 5-9 所示。

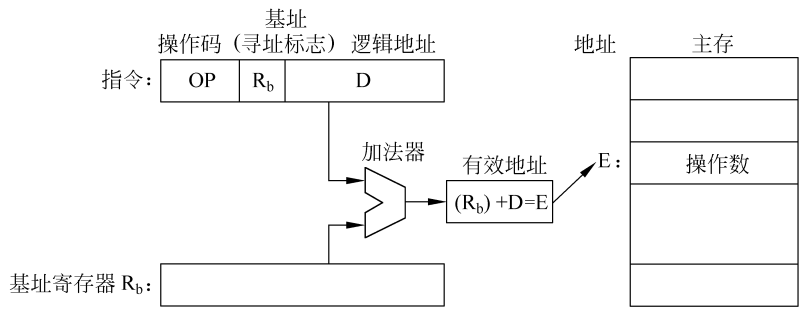

图 5-9 基址寻址

存放操作数的有效地址是 $EA=(R_b)+D=E$,其中 R_b 为基址寄存器。

5.2.2 寄存器寻址方式

在计算机中,运算器中的数据寄存器数目越多,运算越方便,还可以减少访存次数。例如,本条指令的结果下条指令不用,但再下条指令使用,这时运算器中数据寄存器数目少时,必须将本条指令结果先存入存储器,再下条指令使用时,还需由该存储单元取出来送入运算器,多访问了两次存储器;如果运算器的数据寄存器多,就可将运算结果存放到一个暂时不用的寄存器中,不必存到存储器中。

运算器中的数据寄存器称为通用寄存器,各个寄存器的作用,地位都是一样的。

计算机执行程序时,操作数在通用寄存器中存放,取数时不必访问存储器,这样可以加快程序的执行速度。

在相应的指令地址码中,只要给出寄存器号即可。由于寄存器数目不会太多,一般是 4 个、8 个、16 个或 32 个。因此,寄存器的编号为 2～5 位二进制数。例如:有 4 个通用寄存器,则用 00、01、10、11 分别表示 4 个通用寄存器的编号,4 个寄存器也可分别用 R_0、R_1、R_2、R_3 表示,称为寄存器地址。

寄存器地址位数少,指令字长较短,一条指令中可以设置多个寄存器地址。在寄存器寻址的指令中,一般给出两个寄存器地址,相当于二地址指令,指令格式中,除了操作码外,还给出源寄存器编号和目的寄存器编号,分别称为 RS、RD,为使用通用寄存器存放操作数的指令带来很多方便。

1. 寄存器直接寻址

操作数在寄存器中存放，指令的地址码只需给出寄存器号即可。由于寄存器号的位数不多，指令字长较短，另外取操作数不需访问主存，因而指令执行速度较快。

在一地址指令中若使用寄存器寻址，指令中给出一个源寄存器号 R_i，另一个操作数隐含指定放在累加寄存器 AC 中，且 AC 还用来存放运算结果。寄存器直接寻址指令格式如图 5-10 所示，执行：(AC)OP(R_i)→AC。

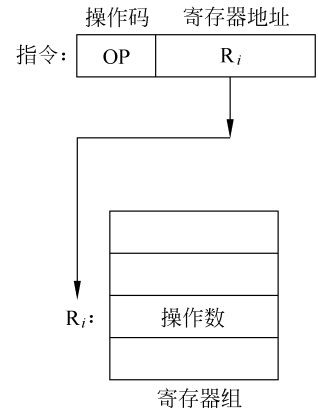

图 5-10　寄存器直接寻址方式

2. 寄存器间接寻址

如果在指令地址码中给出的不是操作数的地址，而是存放操作数地址的主存单元地址时，称为间接寻址。同理，寄存器寻址时，指令中指定的寄存器中存放的不是操作数，而是操作数的地址时，称为寄存器间接寻址。

寄存器间接寻址方式是指令中指定的寄存器含有操作数的有效地址，而不是操作数本身。此种方式，要读操作数必须访问主存，但找操作数有效地址，不访问主存。这种指令的字长也较短，其指令格式如图 5-11 所示。

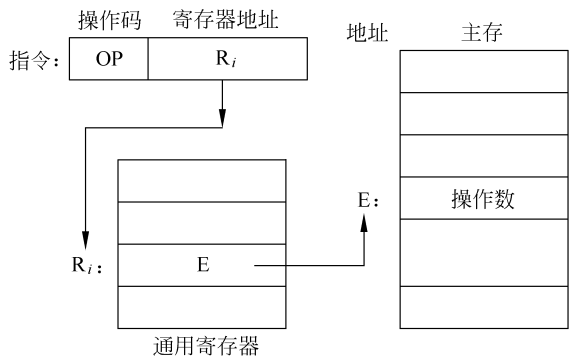

图 5-11　寄存器间接寻址方式

寄存器间接寻址方式操作数有效地址：EA＝(R_i)＝E。

指令执行(AC)OP(((R_i))→AC。

特别强调，R_i 的内容不是操作数，而是操作数的地址。要找到操作数，还需访主存。操作数在存储器中存放，从寻找操作数的角度看，应属于存储器型寻址。但指令格式属于寄存器型寻址，指令中给出的是寄存器编号。

该类指令是寄存器直接寻址，还是寄存器间接寻址，一般由操作码指定，或是由地址码字段的寻址方式位来决定。

有些计算机的指令是二地址指令，一个地址是寄存器号，另一个地址是存储器地址。