第5章 chapter 5

指令系统

本章详细内容参见图 5-0。

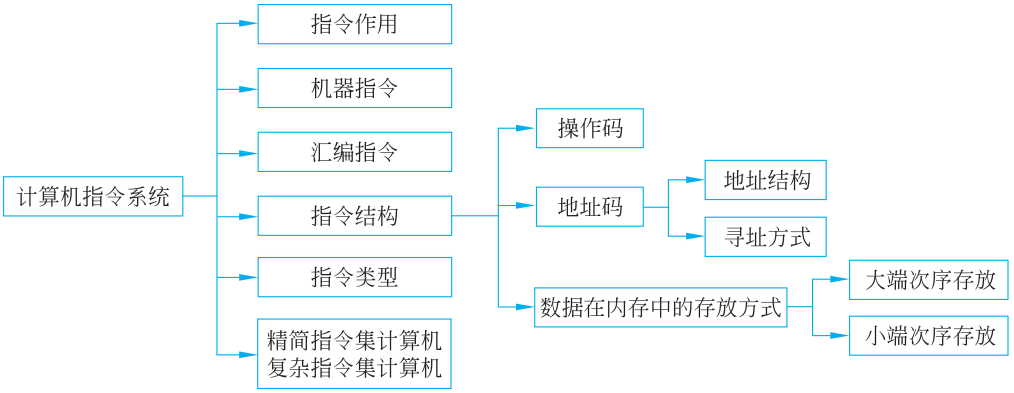


图 5-0　知识点思维图

课程思政材料

要编写程序必须使用指令或语句(语句最终必须经过编译转换为机器指令),指令又称为机器指令,是规定计算机执行某种操作的指示或命令。计算机的工作基本上体现在执行指令上,指令是用户使用计算机与计算机本身运行的最小功能单位。计算机的指令有微指令、机器指令和宏指令之分。微指令是微程序级的命令,它属于硬件,将在第 6 章讨论控制器的设计时介绍。宏指令是由若干条机器指令组成的软件指令,它属于软件,在汇编程序设计课程中介绍。机器指令介于微指令与宏指令之间,通常简称为指令,每一条指令可以完成一个独立的算术运算或逻辑运算。

本章主要讨论以下问题。

(1) 什么是指令? 指令的作用是什么?

(2) 什么是指令系统?

(3) 指令的格式,即一条指令应该包含哪些部分? 这些部分如何组织?

(4) 如何在内存中找到要执行的指令和要处理的数据,即什么是指令的寻址方式?

(5) 指令系统中应该包含哪些类型的指令?

(6) 什么是精简指令集? 什么是复杂指令集? 各有什么特点?

5.1 概 述


视频：指令系统概述

用户要用计算机解决问题就必须编写程序，编写程序必须使用指令（用高级语言编写的程序最终也要经过编译转换成机器指令）。一台计算机的所有指令的集合构成该计算机的指令系统。指令系统的设计是计算机系统设计最有影响的一方面，是计算机设计人员和计算机编程人员能看到的同一计算机的分界面，是表征一台计算机性能的重要因素。它表明一台计算机所具有的硬件功能。指令的格式与功能不仅直接影响计算机的硬件结构，也直接影响系统软件，影响计算机的适用范围，是设计一台计算机的起始点和基本依据。20 世纪五六十年代，由于受器件限制，计算机的硬件结构比较简单，所支持的指令系统只有定点数加减法、逻辑运算、数据传送等几十条指令。20 世纪 60 年代后期，随着集成电路的出现，硬件功能不断增强，指令系统越来越丰富，并新设置了乘除运算、浮点运算和多媒体等指令，指令数多达一二百条，寻址方式也日趋多样化。

随着集成电路的发展和计算机应用领域的不断扩大，从 20 世纪 60 年代开始出现了系列计算机，所谓系列计算机，是指基本指令系统和基本体系结构都相同的一系列计算机。系列计算机解决了各机种的软件兼容问题，同一系列的各机种有共同的指令集，并且新推出的机种指令系统一定包含旧机种的全部指令。

20 世纪 70 年代末期，计算机硬件结构随着 VLSIC 技术的飞速发展而越来越复杂化，大多数计算机的指令系统有多达几百条指令，有几十种寻址方式，这样的计算机称为复杂指令集计算机（complex instruction set computer，CISC）。但如此庞杂的指令系统不但使计算机的研制周期变长，不易调试维护，并且非常浪费硬件资源，特别是用于智能家电领域（如冰箱、空调、微波炉、洗衣机）的微控制芯片（单片机），并不需要进行复杂的数据运算，需要的是低成本，这样就将一些不用的指令从系统指令集中剔去，只保留一些最基本的指令，提出了便于 VLSIC 技术实现的精简指令集计算机（reduced instruction set computer，RISC）。随着行业的细分，CPU 设计和制造技术的成熟，专门为各自领域设计的 CPU 芯片也越来越多。

指令系统的性能如何，决定了计算机的基本功能。一个完善的指令系统应满足下面 4 个要求。

1. 完备性

指令系统的完备性是指用指令系统中的指令编制各种程序时，指令系统直接提供的指令足够使用，而不必用软件实现。换句话说，要求指令系统内容丰富、功能齐全、使用方便。在指令系统中有一部分指令是基本的、必不可少的，如数据传送指令、加法指令等。指令系统中还有一部分指令（如乘法指令、除法指令、浮点运算指令等）既可以用硬件实现，由指令系统直接提供这类指令，也可以用其他指令编程实现，例如乘法可以采用加法指令和移位操作指令实现，但这两种不同的实现方法在程序的执行时间和编写程序的难易程度上差别很大。

2. 有效性

指令系统的有效性是指该指令系统所编制的程序能够高效率地运行。高效率主要体现在使用该指令系统提供的指令编制的程序静态占存储空间小、动态执行速度快两方面。

3. 规整性

指令系统的规整性包括指令系统的对称性、匀齐性和指令格式与数据格式的一致性。指令系统的对称性是指在指令系统中，所有的寄存器和存储器单元都可同等对待，所有的指令都可使用各种寻址方式，不需要根据指令来选择系统资源，指令的这一性质对于简化汇编程序设计，提高程序的可读性非常有用。

指令系统的匀齐性是某种操作性质的指令可以支持各种数据类型，如算术运算指令既支持字节、字和双字整数运算，也支持十进制数运算和单、双精度浮点运算等。指令的这种性质，可以使汇编程序设计和高级语言编译程序无须考虑数据类型而选用指令，可提高编程的效率。

指令格式与数据格式的一致性是指指令长度与数据长度有一定的关系，通常指令长度与数据长度均为字节的整数倍。

4. 兼容性

兼容性是指某计算机上运行的软件可以不加任何修改在另一台计算机上正确运行。例如，在采用 Intel 公司设计的 CPU 制造的计算机上开发的程序，在采用 AMD 公司设计的 CPU 制造的计算机上一样可以运行。指令系统的兼容性可以使大量已有的软件得到继承，减少软件的开发费用，使新机种一出现就能继承老机种的丰富软件，深受新老用户的欢迎。

视频：指令格式和数据的存储方式

5.2 机器指令

5.2.1 机器指令格式

指令格式是指令在计算机中的二进制表示的结构形式。一条指令一般应提供两方面信息：一是操作码，规定 CPU 执行什么操作；二是地址码，指出被处理的数据从哪里取，结果送到什么地方以及下一条指令存放在何处。由于计算机中指令是用二进制形式编码表示的，因此，这两部分分别称为操作码和操作数地址码，操作数地址码可简称为地址码（或操作数）。这些代码被划分成几个字段，字段的结构和组合形式称为指令格式。

指令格式的设计是一个比较复杂且技巧性较强的问题，一般涉及指令字长度、操作码结构和地址码结构。图 5-1 所示为指令的基本格式。

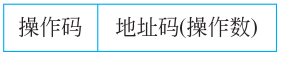


图 5-1 指令的基本格式

操作码：指令系统中包含有许多指令，为了区别这些指

令，每条指令都用唯一的代码表示其操作性质，这就是指令的操作码，用 OP 表示。例如，用四位二进制代码表示操作码，0001 表示执行加法操作，0010 表示执行减法操作……1111 表示执行停机操作等。计算机可以通过译码电路识别它们，并进行相应的处理。

地址码：指令中的地址码字段用来指出操作数的地址。操作数可分为源操作数和目的操作数两种。源操作数只表示处理的对象来自何处（数据来源），它不存放处理的结果，用 OPS 表示。目的操作数用来指明指令的处理结果置于何处（存放在什么地方，目的地），用 OPD 表示。

在大多数情况下，待取的下一条指令紧跟当前指令之后，此时下一条指令的地址由程序计数器（PC）给出，指令中不必提供此地址，只有当程序发生转移时需要在指令中给出下一条指令的地址。

1. 指令字长度

指令字长度是指一条指令字中包含的二进制代码的位数。指令字长度与存储器尺寸、存储器组织、总线结构、CPU 复杂程度和 CPU 速度等相互影响。

为了编程方便，编程人员希望指令有更多的二进制位。指令所包含的二进制位越多，所能表示的操作信息和地址信息就越多，指令功能就越丰富，编程人员可以用较短的程序、更灵活的方法、更大的寻址空间来完成给定的任务。但是，指令字越长，占用的存储空间就越大，读取指令的时间也会增加，执行指令的时间也就越长，而且对于功能简单的指令，过长的指令长度也是浪费。

2. 操作码的结构

操作码是指令中表示机器操作类型的部分，其长度（二进制码位数）决定了指令系统中完成不同操作的指令的条数。操作码位数越多，所能表示的操作种类越多。

操作码的长度取决于计算机指令系统的规模，指令系统越大，包含的操作越多，操作码的长度相应就要长些，反之，操作码的长度可以短一些。通常，一个含有 n 位长度的操作码，最多能表示 2^n 条指令。例如，设计具有 32 条指令的计算机，操作码的长度至少需要 5 位（$2^5=32$）才可以满足需要。

1）固定长度操作码

固定长度操作码的长度固定，且集中放在指令字的一个字段中，这种结构的优点是有利于简化硬件译码电路，减少指令的译码时间，而且便于扩充操作种类。

2）可变长操作码

可变长操作码是操作码的长度允许有几种不同的选择，不再是固定长度。当指令长度较长时，可以利用某些类型指令中地址位数的减少扩充操作码的位数，所以又称为扩充操作码。在扩充操作码时，首先分析指令系统中的地址结构，即不同类型指令所需要的地址数，按指令中给出的地址数，可将指令分成三地址指令、二地址指令、单地址指令和零地址指令。地址数较多的指令其地址段所需要的位数较多，允许操作码占有的位数就较少。地址数少的指令，其地址段位数一般也较少，允许将地址段减少的位数分配给

操作码使用，当然也可以不考虑地址码而直接扩充操作码的位数。

早期的计算机都采用单一固定长度（所有的指令长度都是相同的）的指令，称为定长指令格式，现代计算机大多采用变长指令格式。例如，8086 的指令为 1～6 字节，80386/80486 最长的指令达 15 字节，而 Pentium 最长的指令达 16 字节。变长指令使用灵活，执行效率高。通常，指令的长度都是字节的整数倍。

3. 地址码的结构

指令中地址结构包括在指令的地址码字段中给出几个地址，以及地址如何给出等问题（寻址方式问题）。按照指令中地址码部分给出的地址个数的不同，可将指令分为三地址指令、二地址指令、单地址指令和零地址指令等。

1）三地址指令

三地址指令的格式为：

OP	A1	A2	A3

功能：(A1)OP(A2)→A3。

三地址指令在操作完成后两个操作数都不被破坏，用户使用方便。

2）二地址指令

二地址指令的格式为：

OP	A1	A2

功能：(A1)OP(A2)→A2。

由于两个操作数在执行完相应操作后，有一个操作数不需要保留，如数据传输指令，所以，可以将一个操作数的地址作为存放运算结果操作数的地址，这就形成了二地址指令。通常把兼作存放运算结果操作数的地址称为目的操作数地址，另一个操作数地址称为源操作数地址。

3）单地址指令

单地址指令的格式为：

OP	A

功能：OP(A)→A。

对只有目的操作数的单操作数指令（如加 1、减 1，求补等指令），A 既是源操作数地址，又是目的操作数地址。

另一种情况是虽然有两个操作数，一个操作数由 A 给出，另一个操作数（目的操作数）隐含在累加器 Acc 中（或其他特殊功能寄存器中），(A)OP(Acc)→Acc。

4）零地址指令

零地址指令的格式为：

指令中只有操作码，而没有操作数地址。这种指令有两种可能：一是指令不需要任何操作数，如空操作指令 NOP、停机指令 HLT 等；二是所需要的操作数默认在堆栈中，由堆栈指针 SP 隐含给出，结果仍然放回到堆栈中，如子程序返回指令 RET。

以上几种指令格式，从编制程序长度、用户使用方便程度、增加基本操作的并行性等方面来看，三地址指令与二地址指令格式较好。从减少程序占用空间、减少访存次数、简化计算机硬件设计等方面来看，单地址指令格式较好。当然，到底采用哪种指令，首先要根据指令操作功能来确定。

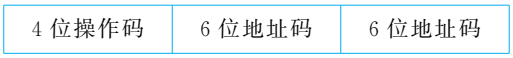
指令中所需要指明的操作数可能在存储器中，也可能在 CPU 的寄存器中。因此指令中给出的地址可以是存储单元地址，也可以是寄存器地址（寄存器地址通常采用寄存器编号来表示）。

【例 5-1】 某计算机指令字长为 16 位，有双操作数指令、单操作数指令和无操作数指令三种格式。每个操作数均为 6 位二进制。该指令系统中共含 m（$m<16$）条双操作数指令，并且存在无操作数指令，若采用扩充操作码技术，那么最多还可以设计_____条单操作数指令。

A. 2^6　　B. $(2^4-m)\times 2^6-1$

C. $(2^4-m)\times 2^6$　　D. $(2^4-m)\times(2^6-1)$

解： 假设指令中全是二地址指令，则指令格式为：

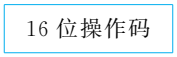


最多 2^4 条双操作数指令。

假设指令全是单地址指令，则指令格式为：

最多可以有 2^{10} 条单操作数指令。

假设指令全是零地址指令，则指令格式为：



最多可以有 2^{16} 条零操作数指令。

所以，双操作指令最多有 16 条（2^4）。每减少 1 条双操作数指令，就多出 6 位地址码，就可以增加 2^6 条单操作数指令。每减少 1 条单操作数指令就可以增加 2^6 条无操作数指令。由于题目中告诉我们有 m 条双操作数指令，说明减少了（2^4-m）条双操作数指令，最多有（2^4-m）$\times 2^6$ 条单操作数指令。题目中说有无操作数指令，并没有说有多少条无操作数指令，但要求单操作数指令最多，所以只要减少一条单操作数指令就可以有 2^6 条无操作数指令，满足“有”无操作数的条件，同时满足单操作数指令最多。所以最多有（2^4-m）$\times 2^6-1$ 单操作数指令，答案选 B。

5.2.2 操作数类型和存储方式

1. 操作数类型

机器指令可对不同类型的操作数进行操作。操作数的类型主要有地址、数值数据和非数值数据三种类型。

2. 操作数的存储方式

操作数的存储方式是指操作数在存储器中的存放顺序，由于计算机是按照字节对存储器进行编址的，如果一个操作数长度占多个字节单元，就必然会遇到操作数的存放顺序问题，通常有两种存放顺序：大端次序存放和小端次序存放。

1）大端次序

大端次序的数据存放方式是最高有效字节存储在最小地址位置单元，最低有效字节存储在最大地址位置单元。例如，一个占 4 个字节数操作数 87654321H，大端次序存放示意图如图 5-2(b)所示。

图 5-2 大、小端次序存放示意图

采用大端次序优点如下。

- 字符串排序方便。因为字符串与整型数据的字节顺序一致，在对大端次序字符串进行比较时，整型 ALU 可以同时比较多个字符（比较大小是从高位进行比较的）。
- 进制数及字符串的显示方便。所有的数值可直接以从左到右的顺序显示（从高位到低位）而不会产生混淆。
- 顺序一致性。大端次序计算机采用相同的顺序存储其整型数和字符串。

2）小端次序

小端次序的数据存放方式是最低有效字节存储在最小地址位置单元，最高有效字节存储在最大地址位置单元。一个占 4 个字节数操作数 87654321H 的小端次序存放如图 5-2(a)所示。

小端次序优点是适合于超长数据的算术运算。对小端次序存储的数据进行高精度算术运算时，不需要先寻找最低字数据，然后反向移动到高位。

Intel 80x 系列计算机采用的是小端次序。

【例 5-2】 在一个按字节编址的计算机系统中，数据在存储器中以小端次序存放。假设 int 型变量 i 的地址为 08000000H，i 的机器数为 01234567H，地址 08000000H 单元中的内容是_____。

A. 01H　　B. 23H　　C. 45H　　D. 67H

解：按照字节编址，变量 i=01234567H 占 4 个字节单元，单元地址为 08000000H～08000003H，采用小端次序存放，变量的最低有效数据 67H 存放在低地址。所以地址 08000000H 单元中的内容为 67H，答案为 D。

5.2.3 指令类型


视频：指令类型

不同的计算机系统设置的指令类型是不完全相同的，但通常都包括一些基本类型指令，这些基本类型指令一般可以归纳为以下几类。

1. 数据传送指令

数据传送指令是计算机中最基本的指令，也是数量最多、使用频率最高的一类指令。这类指令用来完成计算机系统内部各功能部件之间的数据传送，包括：

- CPU 内部各寄存器之间的数据传送(R↔R)。
- CPU 内部寄存器与内存之间的数据传送(R↔M)。
- CPU 与外部设备之间的数据传送(R↔I/O)。

2. 算术逻辑运算指令

算术逻辑运算类指令主要完成数值数据的加、减、乘、除运算和非数值数据的与、或、非等逻辑运算。有的计算机还设计了浮点运算指令和十进制运算指令。

3. 控制转移类指令

计算机在执行程序时，通常是按照指令地址的存放顺序执行。如果遇到执行结果出现多种情况，并且根据不同结果执行不同的分支程序，就需要分支跳转指令来实现。控制转移类指令的操作就是将转移地址送到 PC 中，用来实现程序的分支跳转。

转移指令分无条件转移(绝对跳转)指令和条件转移指令。使用无条件转移指令时，转移不需要任何条件。使用条件转移指令时，程序执行的流向取决于上条指令执行的结果或其他的一些标志。

除了各种转移指令外，控制转移类指令还包括子程序调用与返回、中断及中断返回指令。

4. 程序控制类指令

程序控制类指令只完成某种控制功能，因此它们都是无操作数指令，如暂停指令、空操作指令、开中断指令、设置标志位及清除标志位指令等。

5. 输入输出指令

输入输出指令是计算机与外部设备进行数据传输的一类指令。不同的计算机系统采用的方法不同，有的计算机系统专门设置输入输出指令来访问外部设备（如 Intel 80x 系列 CPU）；有的计算机系统不设置输入输出指令，而是把外部设备等同于主存储器，外部设备与主存统一编址，直接用访问主存的指令来访问外部设备（如 MCS51 单片机的 CPU）。

以上仅仅是对指令种类简单的分类介绍，更详细的情况可以参考各 CPU 的指令系统手册。

5.2.4 指令助记符

在计算机系统中，无论是数据还是指令都必须数字化，用 0、1 的不同组合来编码表示，即机器指令。这种表示方法不便于程序编写、阅读和修改，为此引入了助记符和汇编指令及汇编的概念。通常为了方便记忆与理解，将机器指令用一组英文字母来表示，使用的英文字母通常是与机器指令语意一致的英语单词缩写。例如，数据传送类指令用 MOV（move）表示，跳转指令用 JMP（jump）表示，加法指令用 ADD（add）表示，加 1 指令用 INC（increment）表示，与指令用 AND（and）表示，输入/输出指令用 IN/OUT（in/out）表示，有一点英语基础的人看到了助记符就可以猜到该指令的大概含义。用助记符表示的指令称为汇编指令，用汇编指令编写的程序称为汇编源程序。汇编源程序必须通过汇编程序（相当于高级语言的编译器）汇编成（翻译成）机器指令，计算机才能执行，这部分内容在汇编语言程序设计课程中详细介绍。


视频：寻址方式

5.3 寻 址 方 式

寻址方式是指产生寻找指令和操作数实际（有效）地址的方法。

5.3.1 指令的寻址方式

形成指令地址的方式称为指令的寻址方式。通常指令在主存中按地址顺序存放，下一条要执行指令的地址存放在程序计数器中。当程序顺序执行时，在取指令周期根据 PC 中的内容从内存中取一条指令到 CPU 内部的指令寄存器 IR 中，同时将 PC＋Δn→PC（Δn 取决于该条指令占内存的字节单元数）形成下一条指令的地址。当程序发生转移时将转移的入口地址送 PC，下一条要执行指令的地址始终在 PC 中，所以，一般不讨论指令地址的寻址方式。

5.3.2 操作数的寻址方式

所谓操作数的寻址方式，就是确定参加运算的操作数有效地址的方法。操作数通常存放在主存中，或在 CPU 的内部某一寄存器中。在指令的地址码（操作数）字段，一般给出的是确定该操作数地址的方法，而不是直接给出操作数。这是因为计算机处理的操作

数一般都是变量，程序员开始并不知道这个变量的确切值，只能在存储器中提供一个存储单元来保存该变量，这个存储单元的地址就是该操作数的有效地址。计算机在执行指令时，根据操作数字段提供的寻址方式获得操作数的地址，并根据该地址得到所需要的操作数。指令的地址码字段一般由两部分组成，一个是寻址方式字段，其位数决定寻址方式种类多少；另外一个字段用字母 A 表示，其含义取决于寻址方式字段。地址码格式如图 5-3 所示。

后文中使用符号的含义：A＝指令中地址字段的内容；EA＝有效地址；(X)＝X 地址单元中存放的内容；R＝寄存器。

1. 立即寻址

立即寻址方式由指令中的操作数地址 A 段直接给出操作数(立即数)，在取出指令的同时就取出操作数，所以，又称立即数寻址。图 5-4 所示为立即寻址。

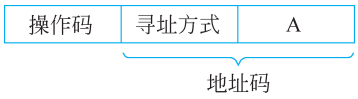


图 5-3 地址码格式

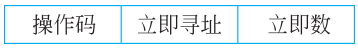


图 5-4 立即寻址

立即寻址一般是给变量置初值，优点是获取操作数不需要另外访问存储器。

2. 直接寻址

直接寻址是指指令的操作数地址字段 A 含有操作数的有效地址，根据该地址访问内存可以直接读取操作数。图 5-5 所示为直接寻址。

3. 间接寻址

直接寻址的寻址范围有限，如给出 20 位的地址也只能在 1M 的地址空间范围寻址，如果要用直接寻址的方法在 CPU 提供的整个地址空间范围寻址，必须保证直接寻址的地址字段 A 的位数同 CPU 提供的地址线位数完全相同，这将使该指令的长度非常长。解决的方法是让地址字段指示一个存储器单元地址，而此地址对应的单元中存放操作数的全长度地址，这样的寻址方式就是间接寻址。图 5-6 所示为间接寻址。

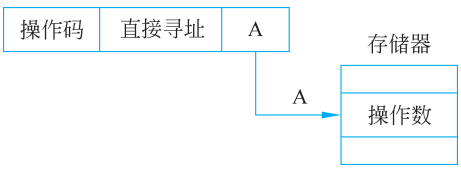


图 5-5 直接寻址

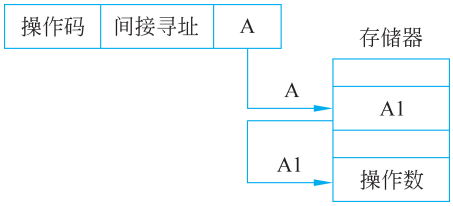


图 5-6 间接寻址

间接寻址的缺点是获取一个操作数，指令执行两次访存，第一次获得操作数地址，第二次获得操作数，指令执行时间就相对较长。