刘武克 郑 建 编

浙江大学出版社

VAX 汇编语言程序设计

VAX汇编语言程序设计

刘武克　　郑　建编

浙江大学出版社

1989年·杭州

内 容 简 介

用汇编语言进行程序设计是计算机专业人员最重要的基础训练之一。本书选用具有代表性和先进性的VAX计算机作为实例，阐述汇编语言程序设计的基本概念和实用技术。内容翔实、笔墨精练、由浅入深、实例丰富是本书的特色。在内容编排上，既介绍汇编语言程序设计的一般原理和方法，又给出了许多涉及VAX系统特色的感兴趣又较深入且实用的程序设计技术技巧。

本书可作为大专院校计算机有关专业的教材，也可供有关的科技人员阅读参考。

VAX汇编语言程序设计

刘武克　郑　建编

*　　*　　*

浙江大学出版社出版

浙江大学印刷厂印刷

浙江省新华书店经销

*　　*　　*

开本787×1092　1／16　印张13.5　字数328千

1989年6月第1版　1989年6月第1次印刷

印数：0001—4000

ISBN　7-308-00169-5

TP·015 定价：4.90元

前　言

汇编语言是现代各种计算机必备的基本软件。用汇编语言进行程序设计是计算机专业人员最重要的基础训练之一。学习汇编语言，除了能够掌握这一独具特色的程序设计工具外，还有助于了解计算机的体系结构和内部工作机制，以及高级语言的翻译和执行过程，进而能够更充分、直接和有效地使用和开发计算机的各种资源。

汇编语言是面向机器的。本书之所以选用VAX计算机，是基于该机种的普及性和代表性。VAX在体系结构方面树立了权威和标准，是举世公认的优秀超级小型计算机系列，在国内外得到广泛应用，有很大的装机比例和良好的发展前景。而从汇编语言入手，正是深入了解和掌握该机的最低要求和最佳途径。

迄今我国大多数院校的计算机专业仍沿用VAX的前身，即16位的PDP-11小型机，作为汇编语言课程的实例。有的院校虽已配备VAX机，但由于缺乏教材，不得不利用VAX的兼容工作方式来学习PDP-11汇编语言。本书就是意在改变这种状况。我们知道，16位小型机过去在同高性能的32位超级小型机和价格低廉的微型机的争夺中已经注定了被淘汰的趋势。现今超级微型机的发展更促其走入末路。VAX与PDP-11在汇编语言上虽有形式上的相似之处，却具有大为先进和发展了的体系结构。VAX除了大大扩展的虚存地址空间外，还具有更加丰富的寻址方式、更加广泛的数据类型、远为完备的指令系统，以及更为有力而易用的系统支撑功能。通过学习VAX计算机的体系结构和汇编语言，将有助于读者在汇编语言一级迅速了解和掌握所遇到的任何新机器。

学习汇编语言，首先必须熟悉特定计算机的体系结构，亦即必须知晓所有机器硬件的属性。然后，要了解特定汇编语言的语法规则以及特殊的功能和约定。只有在逐步通晓机器提供和支持的指令系统、寻址方式和数据类型等的意义和用法的基础上，才能正确地用汇编语言实际编程了。也只有通过对各种系统支撑功能和设施的进一步掌握，才能在汇编语言一级得心应手、灵活高效地操纵和使用计算机各项资源。

按照上述学习语言的思路，本书在编排上从汇编语言程序设计的必备知识开始，使读者逐步对一个具体的计算机体系结构有一个完整的了解，进而掌握各种程序设计的基本方法、技巧，以及基本数据结构的使用和实现原理，以便能编写出水准较高而且规模相当的VAX汇编语言程序。

书中用大量程序实例帮助读者对各种概念加深理解，同时也使读者在运用汇编语言描

述计算机硬件和算法，以及在程序设计风格方面具备良好的素养。此外，书中还介绍了一些涉及VAX系统特色的令人感兴趣的实用程序设计技术与技巧。这对于专业人员也不乏参考价值。

本书附有习题，以供读者练习之用。为便于初次使用VAX机的学生上机实习，在附录中还编入了简明的VAX汇编语言上机操作指南。

本书承蒙金连甫同志审阅，谨此致谢。

书中的疏漏乃至谬误之处，恳请读者指正。

编者

1988年10月

目　　录

1 绪　言

1.1 汇编语言

每一台计算机都有它自己的指令系统。指令系统中的每一条指令都称为机器指令。计算机能够理解机器指令在机内的代码和执行这些特定代码所规定的操作。也就是说，每一条机器指令在计算机中都有对应的硬件线路或微操作来实现。指令系统定义了计算机唯一能够直接接受和理解的语言，我们把它称为机器语言。机器语言的形式是由0和1组成的二进制代码。其他计算机语言，不管如何高级，最后都要翻译为机器语言才能被计算机所执行。

为了使计算机工作，我们必须明确地告诉计算机硬件要进行的操作、操作的对象及操作对象的地址。用机器语言编写的程序是二进制代码形式的机器指令序列。如果在计算机内存中放入若干条机器指令代码，计算机就能依次将每条指令按照编码的不同进行解释，转换为实际的操作序列，完成人们给定的预期任务。

使用二进制代码形式的指令，对计算机实现来说，自然十分方便，但对人来说却是很不方便的，机器语言难认难记，这就使得直接用二进制代码编制程序十分困难，要修改和调试这些机器语言程序就更为困难。

为此，人们把机器指令符号化，用指令助记符来表示操作，用符号地址来表示操作对象。我们把这种符号化了的机器语言称为汇编语言。汇编语言在发展充实过程中，除了包含符号化的机器指令外，还包含了伪指令、宏指令等软设施，从而成为一种独具特色的程序设计工具。

由于计算机只认识机器语言，所以用汇编语言编制的程序要经过一次转换，翻译为等效的机器语言程序后才能投入运行。这个翻译过程称为汇编，它也是由计算机完成的。

汇编语言比机器语言易读、易记、易查、易改，这就方便了程序的编制、修改、阅读和交流。同时，汇编语言又保持了机器语言的优点，例如执行速度快，能够充分利用特定计算机的硬件特点，有效使用各种硬件资源，从而可以编出高质量高性能的汇编语言程序来。因此，计算机系统中有许多软件，特别是操作系统中的大多核心程序都采用汇编语言编写。某些高级语言编译程序的目标程序往往也采用汇编语言的形式。汇编语言程序在实时应用和微机开发应用领域被广泛采用。在现代计算机系统中，不论规模大小，汇编程序都是必备的基本软件之一。汇编语言在计算机系统中占有重要地位。

1.2 学习汇编语言的目的

学习和使用汇编语言有以下两个目的。

(1) 充分利用计算机的功能

汇编语言能够精确描述具体计算机的硬件特性，直接控制硬件机构的操作，形成紧凑高效的程序。

一般而言，高级语言的编译系统总是以一定的标准化方式工作，在设计它们时并未把各种具体机型的特性考虑进去，并且编译程序所产生的代码在执行效率上一般都逊于人工仔细编写的汇编语言程序。所以，当我们需要充分发挥机器的某些特性时，或者在那些执行时间和节省内存是重要指标的场合，我们自然会选择使用汇编语言。

(2) 更进一步了解计算机的结构与原理

计算机专业人员必须清楚地了解计算机的结构与原理，这对于使用高级语言也是非常重要的，因为这样能更有效地利用计算机。为了要清楚地了解计算机的体系结构特性和内部工作机制，或者了解计算机的新思想或新概念，最佳的做法就是学习和掌握计算机的机器语言或汇编语言。

通过学习使用汇编语言编制程序，我们可以了解计算机的结构，指令集，指令的解码执行，数据的机内表示，访问数据的方法，各种程序控制结构的实现机制，参数在过程或子例程之间的传递过程，各种数据结构的实际构造和处理手段，各种输入输出设备的控制驱动途径，等等。

汇编语言课程中所涉及的这些内容，将有助于我们通晓计算机实际工作过程，有助于理解高级语言程序的编译与执行，最终，有助于我们进行更高质量的程序设计，更自如、更有效、更高水准地开发和使用计算机。

1.3 计算机系统的一般概念

各种数字计算机，不论其外观、规模、功能和性能诸方面的差异，其基本结构仍属于冯·诺依曼型的范畴，即其基本工作方式都是按照冯·诺依曼在1946年正式提出并论证的“存储程序”原理进行的。本节将就计算机的基本工作原理，以及计算机各组成部分的基本功能、工作特点及相互关系作一般性讨论，以期使读者对计算机系统建立起一个总体的概念。

1.3.1 程序和指令

计算机的工作就是执行程序。所谓程序是为完成特定任务而按一定顺序排列起来的一连串指令。

每台计算机在设计时就制定了一定数量的基本指令。每一条指令都有规定的格式和具体的含义，并有相应的线路实现。计算机能够识别这些指令在机器中的二进制代码，并执行每一条指令所规定的基本操作。指令通常由操作码和地址码两部分组成，操作码规定机器进行操作的种类，地址码指出参与操作的数以及所要操作的数的地址。

一台计算机指令的集合就是该计算机的指令系统。计算机所拥有指令的数目、种类、内容和格式彼此有很大差异。指令系统是计算机体系结构的一个重要特性，在很大程度上决定了计算机在数据运算和数据处理方面的能力。计算机指令系统功能越强，人们编制程序就越方便，程序性能就越好，但随之机器结构也就越复杂。

一台计算机的指令是有限的，但是如把指令按照不同顺序排列，操作于不同的数据对

象，就能够完成各种任务，创造出各种奇观。

在确立体系结构时，除了设计指令系统外，还必须为机器码制定一套访问操作数的模式，我们称这一模式为寻址方式。不同计算机提供的寻址能力和手段是不同的。只有掌握了寻址方式的定义和规律，才能灵活、合理地使用每一条指令。

指令系统和寻址方式相结合就构成了特定计算机的机器语言。机器语言程序用二进制代码给出指令并规定寻址方式，这是机器硬件直接能接受的程序形式。人们采用便于记忆和书写的符号来给出指令和寻址方式，从而编制出各种所需程序。

1.3.2 存储程序工作原理

程序是指令的有序排列。计算机工作时按顺序执行每条指令，最后实现人们预想的功能。但计算机要实现自动连续工作，不能由人送一条指令才去执行一条指令。为此，首先要求在计算机开始工作前，必须把人预先编好的程序，即指令序列和数据，通过一定方式送入并存储在计算机内部。其次，要求计算机工作时，机器的控制部件知道在什么时间到什么地方取哪条指令，每执行完一条指令后又自动去取下一条指令。这样，当计算机开始工作时，只要知道程序中第一条指令放在什么地方，它就能自动地按程序中规定的顺序依次取出要执行的每一条指令，加以识别和执行，从而实现程序的连续、自动执行。

把指令和数据预先存储在机器中，使机器自己能取出指令和数据进行操作，这就是存储程序工作原理。它是电子计算机实现自动计算的根本保证。也正是通过这一原理，确定了构成计算机的各个必要成分。

1.3.3 计算机的基本组成

就计算机硬件系统而言，从本质上讲，几乎所有现代通用数字计算机都由三个单元组成，即存储器、中央处理机（CPU）和输入输出单元，即具有如图 1-1 所示的基本结构。

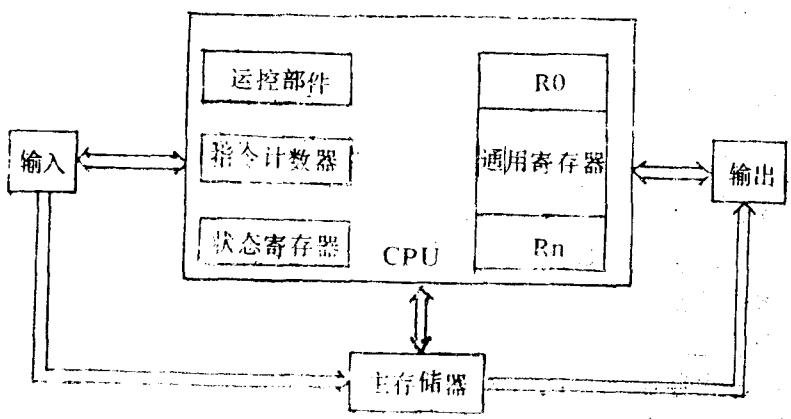

图1-1 计算机基本结构

(1) 存储器

要实现存储程序，计算机必须设有存储信息的记忆装置，即存储器。存储器的主要功能是用来存储指令和数据。存储器一般由电子或电磁器件实现，这些器件能够保持两种或多种不同状态。这些状态能够被读出和改写。因此，每一器件，譬如一个触发器或者一个磁心，能够用来表示一个二进制位。

大多数计算机（包括VAX），把二进制位逻辑地组成8位一个单元，称为一个字节。为了区分不同的字节，把所有字节按一定顺序编号。习惯上把这种编号称为字节的地址。从概念上说，存储器可以看成是由固定大小的存储单元（字节）组成的，按地址顺序编号的一维线性结构。

在这些计算机上，字节是最小的可寻址信息单位。但为了便于处理多种数据类型，往往还以多种方式把若干个连续的字节组合起来，以提供较大的存储单元。每一个这样的存储单元也都有一个存储器中的地址，即它的第一个字节的地址。

计算机存入或读取数据时，首先要给出存储单元的地址，才能进行数据的存取。在这里，我们必须区别存储单元的内容和地址这两个不同的概念。实际上，地址是表征特定存储单元在存储器中相对位置的编号，而内容是存储器特定位置上实际存储的数据。

在存储器中，指令和数据同等对待，均以二进制代码出现，故从它们本身不能区分。从表面上看，存储器中存放着许多以字节或其他存储单元为单位的二进制代码，似乎杂乱无章。实际上由程序计数器（PC）指向的存储器地址上的信息被解释为指令，而由指令所确定的操作对象都作为操作数来使用。指令也可以看作是数据，它们同数据一样可运算，即由指令组成的程序是可修改的。只有这样，才能使指令和数据一样保存在存储器中，实现存储程序。

计算机工作时，中央处理机要逐条从存储器中取出指令加以识别、执行，存储器要保存运算的中间结果和最后结果，还往往要同机器的输入输出设备直接交换数据。因此，计算机的速度与存储器的存取速度以及访问存储器的次数关系很大。随着应用水平的提高，人们对存储器的存取速度和存储容量两者都提出越来越高的要求。然而在实现上，速度和容量存在着尖锐的矛盾。因此几乎所有的计算机系统都具有不止一种形式的存储器，形成存储器的多级结构。最典型的是如下的三级结构：

主存储器 它在计算机内部，所以又称内存。内存直接与CPU以及计算机内其他装置交换数据。它的速度高，但容量小。因此，有时只能把某个程序正在运行的那一部分放在里面。

辅助存储器 它们设置在主机外部，所以又称外存。常用的外存设备有磁盘和磁带。外存的容量比内存大得多，而且价格便宜，但存取速度较慢，通常外存不直接和CPU以及其他外部设备发生联系，而只与内存交换数据。并且不是按单个存储单元存取，而是以成批数据进行交换。计算机系统把内存中暂时不用的代码记入外存，让内存空出来存放当前要用的代码。外存上的代码在需要时再一批批地调入内存。在早期的计算机系统上，一个大的用户程序由于内存容量的限制，往往要分割为若干能够单独运行的模块，逐块地调入内存运行。程序员往往要在很大的精力来解决这种程序覆盖问题。随着硬件技术和操作系统的不断深入和发展，现在的计算机普遍采用虚拟存储器技术，从而实现了程序的自动覆盖。

超高缓存储器 为了进一步提高存取速度，现代的计算机系统往往还在内存与中央处理机之间，增加一个容量不大但速度极高的双极型存储器，称为超高缓存储器（Cache）。当CPU要访问内存时，它首先检查数据是否在Cache中。如果在（称为命中）就能够很快取出数据而不需使用内存。如果不在，则像通常一样访向内存，同时将与现行指令有关的

一批指令和数据送入Cache中。事实上，CPU在一小段时间里所要处理的指令和数据只占一个不大的存储空间，因此Cache的命中率通常在90%以上。也就是说，CPU的访内操作绝大多数是在Cache中进行的，因而附加Cache显著地提高了机器的运算速度。

(2) 中央处理器

中央处理器负责从存储器取出指令，并加以解码和执行，其内容包括确定要执行何种操作、计算操作数地址、按指令的要求产生各种相应的控制电平和工作脉冲，完成指令规定的微操作，如取操作数，运算，存数、转移等，从而保证指令的正确执行。

CPU内部通常含有以下功能部件：

算术和逻辑运算部件(ALU) 用于实施指令规定的各种算术和逻辑运算操作。

若干个通用寄存器 用作累加器、数据的高速临时存放区，以及用于存储器寻址。

程序计数器(PC) 总是存放要执行的下一条指令的内存地址。

一个或多个状态寄存器 用于描述中央处理器的状态、指令操作的结果，以及所发生的任何特殊条件。

若干专用寄存器 CPU和操作系统通过这些专用寄存器执行输入输出操作和控制输入输出数据的传送。这类专用寄存器大多不能被用户访问。

CPU的主要作用是使整个计算机能够自动地执行存储器中存贮的程序。程序在存储器中按顺序存储。CPU用程序计数器PC保持跟踪在所执行程序中的位置。当程序装入存储器时，须随之把程序的启动地址，即程序中要执行的第一条指令所在的内存地址置入PC。CPU用PC作为指针，从内存取出指令加以解码执行。每取出一条指令，PC的内容都被修改为指向下一条指令。这样，处理器就能自动顺序地取出并执行程序中的指令。遇到需要改变这个顺序时，例如转移指令、子程序调用或返回指令等，通过改变PC的内容，即可改变控制流。

(3) 输入/输出单元

与CPU和内存相连接的输入/输出设备使计算机能与人和外部世界进行对话，交换信息。输入设备接受人发出的命令和外界输入的数据，输出设备把经过计算机计算和处理的结果显示或打印出来，乃至直接画图、发音、操纵机器等。常见的I/O设备有终端机、打印机、电传打字机、x-y绘图仪、图形/图像显示器、图片文字扫描仪、图形数字化仪，等等。

CPU和内存组成计算机的主机，输入/输出设备以及磁盘、磁带等辅助存储设备又合称计算机的外部设备。

CPU、内存和外部设备之间是通过总线连接。总线由一组传输线组成，其中包括地址线、控制线和数据线。计算机系统的总线，从数量、配置、数据传输协议和寻址机构上均有差异。但不论何种类型的总线，其作用都是相同的，即在连接于总线上的各种设备之间传输地址、数据和控制信号。地址指定设备，控制信号规定要对数据施行的操作。任何连接到总线上的设备都可发送信息到总线上，也可从总线上读取信息。总线为挂在它上面的各功能部件之间的信息传输提供了有效的通路和手段。

计算机的输入/输出是相当复杂的操作，比CPU执行的算术和逻辑运算复杂得多。这是由于它涉及与各种计算机外部设备的通讯和数据交换。计算机可以与大量的不同设备相

连，而每种设备都有它自己的数据格式、传送速率和控制规约。因此，现代计算机的物理级I/O操作都由用户可以访问的操作系统中的例程来执行。关于外部设备的物理特性、数据的实际存放格式和位置等均由操作系统负责处理，对用户透明。事实上，用户一般既无特权，也无必要直接编程I/O。

1.3.4 操作系统

计算机主机和外部设备这些实际的物理装置构成计算机的硬件系统。但是只有硬件的计算机只会空转，在人们把编制好的程序送入计算机存储器之前，计算机简直一无所为。各种程序，包括计算机厂商和软件制造商提供的操作系统、程序设计语言、信息管理系统及网络通讯系统等各种系统程序，以及用户自己编写的应用程序，通通称为软件。计算机技术发展到今天，我们已不可能仅同由硬件构成的课机打交道，而是与由硬件和软件共同组成的整个计算机系统交往。硬件和软件，缺一不可。许多软件，特别是操作系统已经成为计算机系统不可分割的组成部分。

操作系统是随计算机一同交付用户使用的一套必备基本程序，它是整个计算机系统全部资源的管理系统。操作系统由许多各具控制、管理、服务及命令处理等功能的子例程组成。这些子例程互相配合，共同完成对计算机各种硬、软件资源的分配、调度、控制和协调，同时为用户提供使用和操作计算机的便利环境。

操作系统是用户和计算机之间的接口。任何一个用户都是通过操作系统来使用计算机的。任何一个程序的执行都是在机器硬件和操作系统例程的共同支持和控制下完成的。计算机配上操作系统，就会提高计算机的使用效率，使各种资源的管理更加合理可靠，用户使用更加方便灵活。

1.4 计算机运算基础

计算机的基本功能就是数据计算和数据处理。在计算机中，除了人们日常习用的十进制数据表示法之外，还广泛使用二进制、十六进制及八进制等数据表示法。对于字符文字等非数字信息，也约定了若干标准和通用的数字编码表示法。

1.4.1 数系

任何数据信息在计算机内部均以二进制代码出现，实行二进制运算。由于计算机元件的固有二进制特性，采用二进制对计算机的实现来说是自然而且几乎是唯一的选择。然而不管二进制对计算机多么便利，对使用计算机的人来说却是非常麻烦的。在考察存储器内的数据或指令时，无论阅读或书写这些冗长的由0和1组成的二进制数节，都是既费力又易出错。因此在计算中通常使用十六进制或八进制数据表示法作为二进制数的简写形式，并作为在二进制数和人们惯用的十进制数之间相互转换的中间过渡形式。十六进制和八进制两种表示法均与二进制表示法有简单直接的转换关系，但是它们的基数大，阅读和书写都远为简短直观。

考察十进制、二进制、十六进制和八进制的数制，就可以发现它们有两个共同点。其一，是采用进位计数制。例如，逢10进1，逢2进1，逢16进1。而10，2，16恰是各种数制中表示一位数所需要的符号数目，数学上称之为基数。其二，是采用位置表示法。也

就是说，一个数字所代表的值取决于它所处的位置。例如 6 在365中代表六十，而6在6931中代表六千。各个数位所表示的值在数学上称为权。容易看出，权的值恰为其基数的某次幂。（不具上述特点的计数制的一个例子是罗马数制，例如罗马数字X 不管在什么位置上都总是表示十或负十。）

对任何一种进位计数制和位置表示法表示的数都可以写出其按权展开的多项式之和。设某数制的一个正整数N写为

$$d_n d_{n-1} \cdots d_2 d_1 d_0$$

其中d_i是表示某位数的数符。若用r表示该数制的基数，则N的十进制值为

$$N = k_n \times r^n + k_{n-1} \times r^{n-1} + \cdots + k_2 \times r^2 + k_1 \times r^1 + k_0 \times r^0 \qquad (1\text{-}1)$$

其中k_i为r进制数字d_i的十进制值，且有$0 \leqslant k_0 \leqslant r-1$。例如十进制数373用不同的进位 制可以展开为

$$373 = 3\times10^2 + 7\times10^1 + 3\times10^0 \qquad \text{(十进制)}$$

$$= 1\times2^8 + 0\times2^7 + 1\times2^6 + 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 \text{(二进制)}$$

$$= 1\times16^2 + 7\times16^1 + 5\times16^0 \qquad \text{(十六进制)}$$

$$= 5\times8^2 + 6\times8^1 + 5\times8^0 \qquad \text{(八进制)}$$

换句话说，有

$$(373)_{10} = (101110101)_2 = (175)_{16} = (565)_8$$

这里我们采用习惯的作法，即在一个数后用一个下标来表示这个数的数制基。

二进制的有效数字只有0和1。十六进制需要16个数符，即 0～9 再加上A、B、C、D、E和F。字母A到F分别用于表示数值10到15。八进制的有效数字是0～7。表 1-1给出部分数的对照表。

表1-1 常用进位制数的对应关系

十进制	十六进制	八进制	二进制	十进制	十六进制	八进制	二进制
0	0	0	0	11	B	13	1011
1	1	1	1	12	C	14	1100
2	2	2	10	13	D	15	1101
3	3	3	11	14	E	16	1110
4	4	4	100	15	F	17	1111
5	5	5	101	16	10	20	10000
6	6	6	110	32	20	40	100000
7	7	7	111	255	FF	377	11111111
8	8	10	1000	256	100	400	100000000
9	9	11	1001	1024	400	2000	10000000000
10	A	12	1010	4096	1000	10000	1000000000000

1.4.2 数制间的转换

(1) 十六进制数、八进制数与二进制数间的相互转换

这种转换十分方便。把二进制数转换为十六进制或八进制表示时，只需把二进制数从最低位开始，向左每4位（十六进制）或每3位（八进制）分为一组，不够的补足4位或3位，然后把每一组二进制数用它对应的十六进制或八进制数字（参考1-1)写出即可。例如，对于二进制数10101101110011，有

$$(0010\ 1011\ 0111\ 0011)_2 = (2B73)_{16}$$

和

$$(010\ 101\ 101\ 110\ 011)_2 = (25563)_8$$

反之，把十六进制或八进制数转换为二进制数，只要把每位数字用所对应的四位或三位二进制数写出就完成了转换工作。

(2) 十六进制或二进制表示转换为十进制表示。

利用(1-1)式能够把任何一种进位制表示的数转换为十进制数。例如

$$\begin{aligned}(A2806)_{16} &= 10\times16^4+2\times16^3+8\times16^2+0\times16^1+6\times16^0\\ &= 665360+8192+2048+6\\ &= 665606\end{aligned}$$

$$\begin{aligned}(11001001)_2 &= 1\times2^7+1\times2^6+1\times2^3+1\times2^0\\ &= 128+64+8+1\\ &= 201\end{aligned}$$

这里的计算都是按熟知的十进制计数系统的运算法则进行的。如果我们手头有一张数制基各次幂的表，或者我们已经记熟数制基若干次幂的值，那么这种转换就更加轻而易举了。

实用中对于二进制数更常用的转换方法是先把它转换为十六进制数，再由十六进制数转换为十进制数。例如

$$\begin{aligned}(11001001)_2 &= (C9)_{16}\\ &= 12\times16^1+9\\ &= 201\end{aligned}$$

(3) 十进制数转换为十六进制数

从十进制到其他进位制的转换要用到除法而不是乘法。注意到(1-1)式中除了最后一项k_0外，其余项均可被基数r整除。由于$0\leqslant k_0\leqslant r-1$，当该数被基数r除时，$k_0$为余数，因此用十进制数N除以基数r，其余数$k_0$就是基r记数制的最低位数字。所得商为

$$k_n\times r^{n-1}+k_{n-1}\times r^{n-2}+\cdots+k_2\times r^1+k_1$$

其中除了k_1外所有项均可被r整除。这样如果我们用第一次除法所得的商再除以基数r，取其余数k_2就是基r记数制的倒数第二位数。继续这一过程，直到最后一次除得商为0时，相应的余数就是基r记数制的最高位数字。例如将十进制数5678转换为十六进制数，有

16|5678

16|354　　余14所以$k_0 = E$

16|22　　余2所以$k_1 = 2$

16|1　　余6所以$k_2 = 6$

0　　余1所以$k_3 = 1$

结果为$(162E)_{16}$。

类似地用除2取余、除8取余也能够实现十进制到二进制或八进制的转换。

人们所以从一开始就习惯于十进制计数，大概是由于人有十个手指头的缘故。我们将发现，许多使用十进制计数系统的场合，几乎能同样方便地使用十六进制。因此，我们实际上并不必经常在十六进制和十进制间进行数制转换。

1.4.3 机器数的补码表示

数在机器中的表示形式称为机器数。无符号整数的二进制形式和它的机器数形式可以是一致的，但是负数在机内如何表示呢？由于计算机中只能用数字化信息，显然只能把符号数值化，采用某种把符号位和数值一起编码的表示方法。有三种方法可以用来在机内表示负的二进制数：原码、一的补码（通常称反码）和二的补码（通常称补码）表示。

原码表示是把一个机器数的最高有效位处理为符号位，而其余位处理为整数的二进制绝对值，并且规定符号位为0表示正号，符号位为1表示负号。例如±92的8位机器数原码表示是

92　　01011100

−92　　11011100

原码表示使实现机器运算变得复杂。加减运算除了要考虑两个数值外，还要对符号位作测试和判别。为此引入了机器数的补码表示。

补码表示仍把机器数的最高有效位作符号位，但数值部分的表示法对正数和负数是不同的。正数的补码表示与原码是一样的；至于负数，一的补码表示是把该负数的二进制绝对值的所有位逐位取反（0变1，1变0）而形成的；二的补码表示是把其一的补码表示再加1而形成的。例如±92的八位机器数补码表示是

92　　01011100

−92　　10100011　　一的补码形式

−92　　10100100　　二的补码形式

采用补码表示的用意是使得符号位无需单独处理，可以和数值一道参加运算。尽管一的补码比二的补码更易实现，但是一的补码有一个乖张之处。试对数字$(00000000)_2$取一的补码，我们得到$(11111111)_2$。即用一的补码表示时，−0和+0有不同的表示法。但在数学上−0 = +0。这种对0的两种表示使得运算中频繁用到的零测试变得困难。因此，现代的计算机，包括VAX，普遍采用二的补码来表示负数。实际上，二的补码是一的补码的一种变化，它排除了后者的负零表示。