微机原理与嵌入式系统

（内部讲义翻印必究）

中国科学技术大学信息科学与技术学院

微机原理与嵌入式系统课程组

2020 年 1 月

目 录

第 1 章 概述

电子计算机是 20 世纪人类最伟大的发明之一。伴随着电子科学、计算机和通信技术的飞速发展，人类已经开始步入了信息社会。为了学好本门课程，本章首先介绍电子计算机的发展历程，然后对计算机中数的表示方法展开讨论，接下来讲述计算机系统的组成、分类和特点，最后介绍当今应用最为广泛的嵌入式计算机的一些基本概念。

1.1 计算机发展简史

1.1.1 现代电子计算机的起源

人类文明从诞生之日起就面临计算问题。为了分配食物和生活物资，穴居在山洞里的古人类采用过在石壁上刻痕和结绳的方式进行计数和计算。中国古代先民使用过的算筹以及后来在华夏大地得到广泛应用的算盘，都是我们祖先寻求计算工具的光辉成就。

工业革命对计算工具提出了更高和更为迫切的需求。17 世纪中叶，法国科学家布莱兹·帕斯卡（Blaise Pasca）发明了基于齿轮结构的机械加减法器；稍后，德国数学家莱布尼茨（Gottfried Wilhelm Leibniz）于 1671 年设计了一架可以进行乘法、除法和自乘运算的机械计算器；19 世纪的英国科学家查尔斯·巴贝奇（Charles Babbage）为了研制基于齿轮结构的“差分机（Difference Engine）”和“分析机（Analytical Engine）”，耗费了大量金钱和毕生心血。以上这些都展现了人类在机械设计方面的高超技巧。

1936 年，英国科学家阿兰·图灵（Alan Turing，1912~1954）发表了现代计算机科学与技术领域的开山之作《论数字计算在决断难题中的应用》，从数学上给出了“可计算性”的严格定义，并提出了著名的“图灵机（Turing Machine）”的设想，指出一切可能的机械式计算都能够通过“图灵机”实现。尽管“图灵机”只是一个思想模型，但是阿兰·图灵证明了可以制造一种运算能力极强的通用计算装置，用来计算所有的、能想象的可计算函数。为了纪念阿兰·图灵这位伟大的科学家，国际计算机协会 ACM（Association for Computing Machinery）于 1966 年设立了“图灵奖”，该奖项现被公认为计算机科学与技术领域的诺贝尔奖。

1943 年，为了解决复杂的火炮弹道计算问题，美国军方委托美国宾州大学莫尔学院开始研制电子计算机。1946 年，在物理学博士莫克利（John Mauchly）和电气工程师艾克特（Eckert）所率领的团队的不懈努力下，世界上第一台数字式电子计算机 ENIAC（Electronic Numerical Integrator And Calculator，电子数字积分器和计算器）研制成功了。这台机器使用了 18 000 多个电子管，1 500 多个继电器，耗电量达 140 千瓦，总质量近 30 吨，占地面积约 170 平方米，可谓是庞然大物。ENIAC 采用字长为 10 位的 10 进制计数方式，通过接插线的方式进行编程，而运算速度仅有 5 000 次加法/秒。虽然 ENIAC 的性能远远不及今天的一台儿童玩具计算器，但 ENIAC 的确是人类科学技术史上一个划时代的伟大创新。为了表彰这两位科学家的杰出贡献，自 1979 年起，ACM 和 IEEE Computer Society 共同设立了 Eckert-Mauchly 奖，以奖励那些在计算机体系结构方面做出重要贡献的学者。

1944 年的一个夏天，著名的美籍匈牙利数学家冯·诺依曼（Von Neumanm）在一次偶然的邂逅中，获知了 ENIAC 项目的消息，他立即提出请求，希望有机会能够参观 ENIAC 的研制

现场。从他看到尚未完工的 ENIAC 第一眼起，冯·诺依曼就与现代电子计算机结下了不解之缘。虽然冯·诺依曼没有参加 ENIAC 的研制工作，但是仍受邀加入 ENIAC 项目组，参加一系列为完善和改进 ENIAC 的研讨会，对新型计算机的体系结构展开了研究。1945 年 6 月 30 日，莫尔学院在内部印发了一份由冯·诺依曼一人署名的研究报告《First Draft of a Report on the EDVAC》。在这份关于 EDVAC（The Electronic Discrete Variable Automatic Computer，离散变量自动电子计算机）的研究报告中，首次提出采用二进制计算和存储程序（Stored Program）的思想，在逻辑结构上将计算机划分为运算器、控制器、存储器、输入设备和输出设备 5 大部件，并描述了各部件的职能和相互间的联系，同时还设计了“条件转移”指令以改变程序执行顺序。但此后 EDVAC 的研制工作却遭遇了一些非技术层面的困难，直至 1952 年才宣告完工。

1946 年初，英国剑桥大学数学实验室主任莫里斯·威尔克斯（Maurice Wilkes）教授到宾大进行访问。在访问期间他参加了 ENIAC 团队主办的一系列讲座，并于 1946 年 5 月获得了一份 EDVAC 方案。回到英国后，威尔克斯以 EDVAC 方案为蓝本，设计和制造了世界上第一台存储程序式电子计算机 EDSAC（Electronic Delay[1] Storage Automatic Calculator）。1949 年 5 月 6 日，EDSAC 首次试运行获得成功。随后，该项目的投资商 Lyons 公司获得了 EDSAC 的批量生产权，这就是 1951 年正式投入市场的“LEO（Lyons Electronic Office）”计算机。

1946 年 3 月 31 日，莫奇利和埃克特向宾大校方递交了辞呈。1947 年两人离开了宾大，在费城的一个临街小楼里创立了埃克特-莫契利电脑公司（英文缩写为 EMCC，Eckert-Mauchly Computer Company）。这是世界上第一个以制造电脑为主业的公司。 EMCC 后因资金问题被老牌打印机制造商雷明顿·兰德（Remington Rand）公司收购。在莫奇利和埃克特两人的参与下，雷明顿·兰德公司于 1951 年研制出了更新式的通用自动计算机“UNIVAC”。分别诞生于英美两国的“LEO”和“UNIVAC”都宣称自己是世界上第一个商业化的计算机产品。

虽然 EDVAC 方案的提出到如今过去了 70 多年，计算机科学与技术发生了翻天覆地的变化，但是时至今日，冯·诺依曼体系结构仍然是许多现代电子计算机的基础。

1.1.2 现代电子计算机的发展历程

站在不同的视角，对计算机的发展史有不同的断代方法。如果按照计算机所使用的主要器件类型，计算机的发展历程大致可以划分为以下五个阶段。

第一阶段为 1946 至 20 世纪 50 年代中期，计算机中的有源器件都是电子管，因而计算机的体积庞大，功耗大，可靠性低，售价昂贵。当时一台计算机的价格与一架喷气式客机的价格大体相当，主要用于重要场合的科学计算和数据处理。

第二阶段为 1955 年至 20 世纪 60 年代中期，晶体管取代了电子管，内存采用快速磁芯存储器，外存采用磁带或者磁鼓，运算速度可达每秒几万次乃至几十万次。晶体管的使用减小了计算机的体积，提高了可靠性，降低了功耗和成本，计算机开始进入过程控制领域，出现了工业控制计算机。在这一阶段，面向过程的程序设计语言，如 FORTRAN、Algol 等高级语言相继面世。

第三阶段为 1965 年至 20 世纪 70 年代初期，计算机的主要部件开始采用中小规模集成电路，计算机的体积进一步缩小，可靠性进一步提高，成本进一步下降，运算速度提高到每秒几百万次。在这一阶段，操作系统逐渐成熟，计算机产业形成了种类多样化、产品系列化和

[1] EDSAC 名称中的 Delay 是因为使用水银延迟线作为程序存储器的缘故。

使用系统化，计算机的应用范围日益扩大，小型计算机开始出现。

第四阶段为 1972~1990 年，大规模集成电路 LSI（Large Scale Integration）和超大规模集成电路 VLSI（Very Large Scale Integration）出现和广泛应用，使得计算机的体积更进一步缩小，性能和可靠性得到更进一步的提高，成本更进一步降低。计算机内存普遍采用半导体存储器，外存采用磁盘、磁带和光盘。1981 年，IBM 公司推出了个人计算机（Personal Computer, PC 机），随后以 PC 机为代表的微型计算机迅速得到普及和广泛应用，深入到人类社会的各个领域。

第五阶段是从 1991 年开始至今，随着半导体工业和材料科学的进步，特大规模集成电路 ULSI（Ultra Large-Scale Integration）、巨大规模集成电路 GSI（Gigantic Scale Integration）、更高密度的存储器件和新型显示设备的广泛使用，计算机的体积愈加减小，速度不断取得突破。如今，普通个人计算机的运算速度都能达到每秒数十亿次。在这一时期，由一片巨大规模集成电路实现的单片计算机和嵌入式计算机的功能不断完善，性能日益提高，应用领域不断扩大，现已成为几乎无处不在和应用最为广泛的计算机。

总之，从 1946 年计算机诞生以来，大约每隔 5 年计算机的运算速度提高 10 倍，以平均故障间隔时间 MTBF（Mean Time Between Failure）为代表的可靠性提高 10 倍，成本降低 10 倍，体积缩小 10 倍。20 世纪 70 年代以来，计算机产量每年以超过 25%的速度递增。

从第三阶段开始，计算机的发展就与集成电路技术的进步密不可分。一块大规模集成电路芯片上的晶体管数量超过 1000 颗，而超大规模集成电路每个芯片上的晶体管数量超过了数百万颗，如今巨大规模集成电路（ULSI）芯片上可集成的晶体管数量超过了 10 亿颗。例如，美国 Intel 公司在 2012 年左右发布的第 4 代酷睿 4 核 CPU i7（研发代号 Haswell），采用 22nm 制程，在一块芯片上集成的晶体管数量超过了 14 亿颗。

1.1.3 摩尔定律

1965 年，Intel 的三位创始人之一的戈登·摩尔观察到芯片上的晶体管数量每年翻一番，1970 年这种态势减慢成每 18 个月翻一番，这就是人们所熟知的摩尔定律（Moore’s Law）。摩尔定律自提出后到 2015 年，基本上比较准确地描述了处理器在这段时间内所取得的令人难以置信的进步。图 1.1 是 1978~2018 年这 40 年来，处理器性能增长的总体态势。

在图 1.1 中，比较基准是第三方性能评测机构 SPEC（Standard Performance Evaluation Corporation，标准性能评估协会）给出的 DEC VAX 11/780 整数运算性能指标（SPEC int）。DEC 公司（Digital Equipment Corporation）曾经是美国著名的计算机公司之一，VAX 11/780 是其生产的一款十分畅销的小型机。VAX 11/780 每秒钟能够执行一百万条指令，亦即 1 MIPS（Million Instructions Per Second），在一段时间内，这款机器被当作是计算机性能指标比较的参照。

图 1.1 中不同灰度的阴影区反映了处理器性能增长的几个不同阶段。在 20 世纪 80 年代中期之前，在处理器设计和半导体技术进步的双轮驱动下，处理器的性能平均每年增长 25%左右。1986 年~2003 年，得益于计算机体系结构方面的进步，处理器性能开始出现飞速发展，年平均增长率高达 52%左右。到了 2003 年，受功耗和散热的限制，处理器性能的年平均增长率回落到 22%左右。2011 年以后，随着并行计算的“红利”逐渐消失之后，处理器性能增长率放慢到 12%以下。从 2015 年起，摩尔定律开始失效，年平均增长率仅有 3.5%。

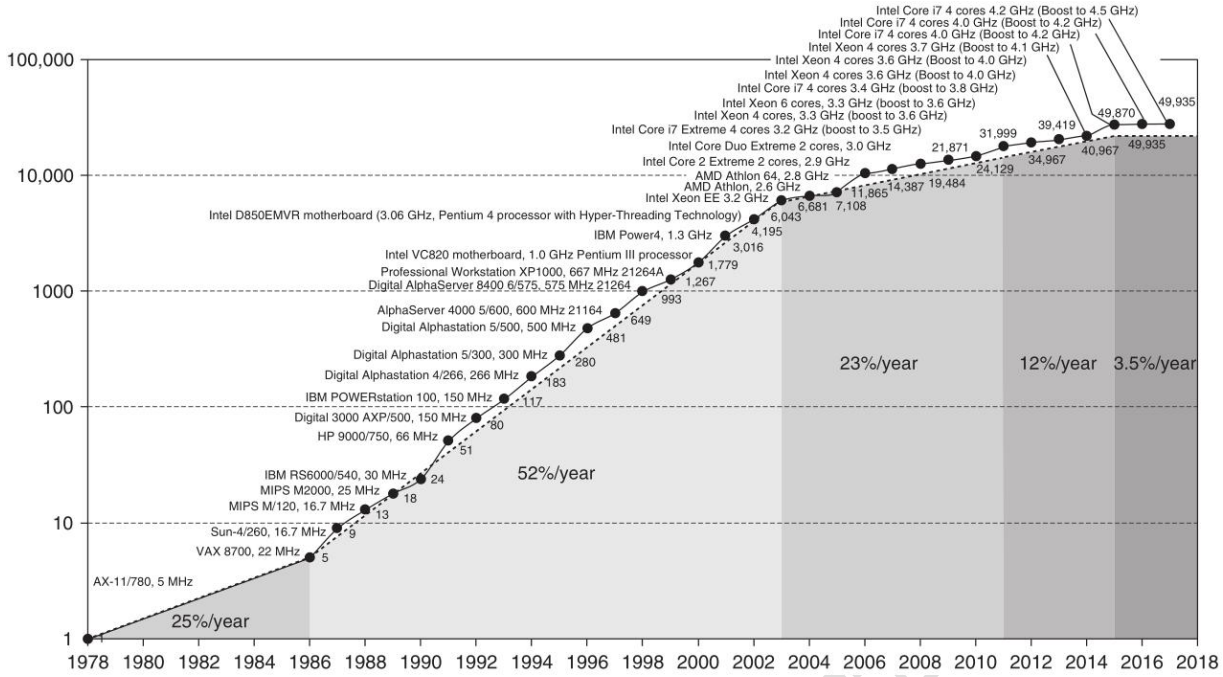

图 1.1　计算机性能 40 年间增长曲线

面向浮点运算的处理器性能也遵循同样的趋势，但是年增长率通常高出整数运算性能的 1%～2%。图 1.1 阴影区域上方的曲线反映了处理器浮点运算性能的增长情况。

随着处理器性能的提高，使用这些处理器制造的计算机整机性能也出现了突飞猛进的发展。在 20 世纪末，世界上运算速度最快的超级计算机，其价值可能高达数十亿人民币。20 年过去了，与今天一部售价不足 5000 元的智能手机相比，其计算能力大体相仿。

1.1.4　计算机的类型

经过 70 多年的高速发展，根据计算机的体系结构、使用目的、计算性能、形态大小和成本等诸多因素，可以将计算机分为多种差异较大的类型。如果按照其应用场合，现代计算机大致可以分为以下四大类型。

1. 微型计算机

微型计算机的主要代表是人们所熟知的个人计算机，即所谓的 PC 机。顾名思义，个人计算机主要面向个人使用，其应用场合遍及办公、科学研究、商业、教育和家庭。个人计算机支持通用计算、文档编制、表格处理、计算机辅助设计、视听娱乐、人际交流和互联网浏览等各种各样的应用。微型计算机根据其形态和主要用途又可以进一步划分为：

- ❒ 台式计算机（desktop computer）。可以满足一般的桌面应用需求，并占用较少的工作空间。
- ❒ 个人工作站（workstation）。为工程设计、图形和视频图像处理、科学计算等应用提供更高的计算能力和更强大的图形显示能力。
- ❒ 笔记本电脑（notebook computer），包括平板电脑（PAD）。提供个人计算机所具有的基本功能，可以使用内置电池工作，满足一定的便携性和移动性需求。

2. 服务器

服务器（server）是指具有较强大计算能力的计算机系统，可以通过网络为大量用户提供计算、信息处理和数据存储服务，一般用于大型企事业单位和政府机构的信息处理服务。

3. 嵌入式计算机

嵌入式计算机（embedded computer）是指集成（嵌入）到应用对象体系（设备或系统）中的一种专用计算机，用以自动监测和控制应用对象的物理过程。嵌入式计算机用于特定目的，而不是通用的任务处理。其典型应用包括工业自动化、智能家居、通信设备、数码产品和交通工具等。嵌入式计算机几乎无处不在，但是它们通常非常低调，用户往往觉察不到其身影，不了解它们在宿主系统中所发挥的作用。

4. 超级计算机

超级计算机（super computer）提供最高的计算性能，在复杂过程仿真、空气动力学计算、中长期天气预报、多光谱遥感图像处理、地质勘探、新型药物设计、基因测序和蛋白质结构分析等对计算能力要求极高的领域，超级计算机扮演着最重要的角色。超级计算机是最昂贵的也是物理上最大型的计算机。

鉴于高性能计算（High Performance Computer，HPC）的重要意义，美国、日本和欧盟等发达国家竞相研发超级计算机。可喜的是，改革开放以来，中国不仅成为世界上第二大经济体，在超级计算机研发领域同样取得了令世人瞩目的成就。在全球超级计算机 500 强（top500）排序中，2004 年中国的“曙光 4000A”跻身第 10 名，开始崭露头角；2009 年国产“星云号”位居排行榜第二；2010 年“天河 1 号”一举拔得头筹，接下来的几年，冠军称号一直由“天河 2 号”保持；2016 年，安装了 40 960 个中国自主研发的“申威 26010”众核处理器的“神威·太湖之光”登顶榜单首位。直至 2018 年 11 月，IBM 公司采用了 2 397 824 个处理器的“Summit”，以超过神威·太湖之光 60%的计算速度，时隔 9 年重新夺冠。Summit 的峰值计算速度达到 200 796T Flops/s，其中 95%的计算能力是由英伟达 Volta GPU 提供的。超级计算机领域中的竞争仍在继续，未来将更加精彩。

1.2 计算机系统的组成

计算机系统是由计算机硬件和软件两部分组成的，如图 1.2 所示。所谓计算机硬件是指构成计算机的物理部件，这些物理部件只有在计算机指令控制下才能运行。而计算机软件是由一系列按照特定顺序组织的计算机指令和数据的集合。

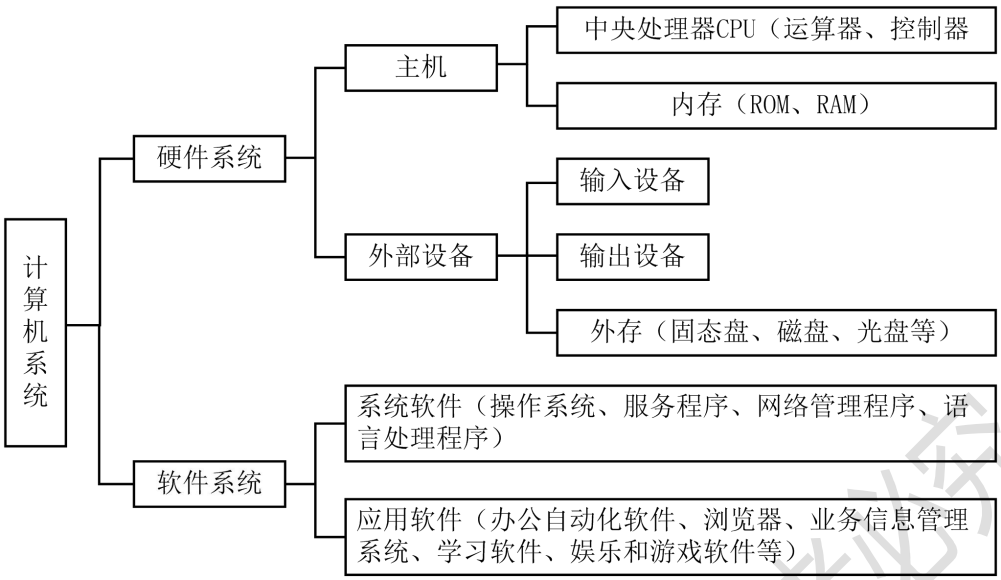

图 1.2 计算机系统的组成

1.2.1 计算机硬件

按照冯·诺依曼所划分的逻辑结构，计算机硬件由以下相对独立的 5 个主要功能部件所组成：存储器、运算器、控制器、输入设备和输出设备，这 5 个功能部件及相互之间的关系如图 1.3 所示。

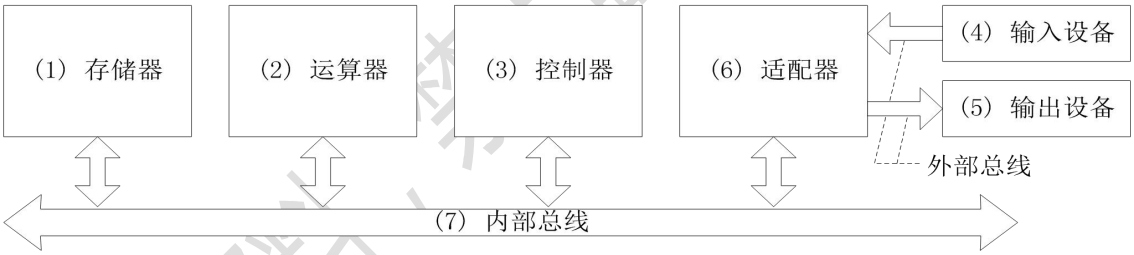

图 1.3 计算机系统主要硬件功能部件

1. 存储器

存储器的功能是存储程序和数据，可分为主存储器和辅助存储器两种。

1） 主存储器（primary memory）

主存储器简称为主存（main memory）。因为主存储器通常位于计算机内部，因此更多地被称作内存。现代计算机的主存都是由大量的半导体存储单元（cell）所构成的，这种最小的存储单元只能存储一位二进制信息，其单位是 1 个比特（1 bit，位）。这些最小存储单元很少被单独地读取或写入（统称为访问，access），一般是由 8 个最小存储单元（8 bit）组成的一个字节（1 Byte）。为了能方便地访问主存中的任何一个字节，每个字节都有一个唯一的物理地址（physical address）。字节是存储器存储和读取数据的基本单位。若干个字节又构成一个字（word），每个字所包含的位数称为计算机的字长（word length），典型的字长有 16 位（2 字节）、32 位（4 字节）和 64 位（8 字节）。

按照读写操作特性，半导体存储器可分为 ROM（Read Only Memory）和 RAM（Random

Access Memory）两大类。前者只能进行读操作，用于存放无需改变的程序和标准数据，例如计算机启动时的引导程序、固化的作业程序以及字符编码表等；后者可读可写，用于存放计算机正在执行的程序、处理过程中的临时数据以及需要与外界交换的信息等。

2） 辅助存储器（secondary storage）

虽然主存储器必不可少，但是主存的价格相对昂贵，其容量受到一定限制。主存还有一个缺点，即作为主存主要组成部分的 RAM，在断电后所存储的信息立刻消失。因此，为了存储大量的数据和程序，尤其是那些不经常访问的信息，就会使用价格低廉、容量较大、存储的信息不会因掉电而丢失（非易失性）的辅助存储器。尽管很多辅助存储器安装在计算机内部，但是相对于内存而言，辅助存储器通常也被称为外存。辅助存储器的容量大，但是读写速度相对较慢。常见的辅助存储设备包括：基于磁介质极化原理的磁盘（magnetic disk）和磁带，基于表面几何微观形状的光盘（optical disk，如 CD、DVD 和 BD），以及基于半导体的快闪存储器（flash memory，简称闪存）等。

2. 运算器

运算器主要完成各种数据的运算和处理，其核心是算术逻辑单元 ALU（Arithmetic and Logic Unit）以及由若干寄存器（register）构成的寄存器阵列。ALU 在控制信号的作用下完成任意的算术或逻辑运算，如加、减、乘、除、移位或比较大小。寄存器是运算器内部的高速存储单元，是整个计算机系统中读出和写入操作所需时间最小的存储部件。一个寄存器通常可以存储一个字的数据。受芯片面积的制约，寄存器的数量不会很多。运算器工作时，需要运算和处理的数据先被送到某个寄存器进行存储，运算过程中的临时数据或者处理后的结果也暂存在特定的寄存器中。

3. 控制器

控制器是计算机的指挥控制中心，它根据指令对计算机各个部件进行操作控制，协调计算机中各个部件有序地进行数据的读出、处理、存储、状态监测和输入或者输出。控制器是由指令寄存器（Instruction Register，IR）、指令译码器（Instruction Decoder，ID）和操作控制器（Operation Controller，OC）等构成的。

大规模集成电路出现之后，运算器和控制器都被集成在一个称为 **CPU**（Central Processing Unit，中央处理器单元）的芯片上。CPU 与主存是计算机的核心部分。

4. 输入设备

输入设备将接收到的信息进行编码后输入到计算机。键盘和鼠标就是最常用的输入设备。例如，当键盘上的一个键被按下时，这个键所代表的字母或数字就被转换成相应的二进制编码，然后再传送到计算机中。

除了键盘和鼠标以外，用于人机交互的输入设备还有触摸屏、操纵杆、轨迹球和麦克风和游戏机手柄等。常见的其他计算机输入设备还有扫描仪、证/照读卡器、光学符号阅读机 OCR/光学标记阅读机 OMR，摄像头、数字化仪、数据采集器等。计算机通信设备中的数据接收单元也可以看作是一种输入设备。

5. 输出设备

输出设备的功能是向外界输出计算机处理后的结果，最常见的输出设备是打印机、显示

器、扬声器和绘图仪等。打印机属于一种慢速输出设备。打印机的类型包括针式打印机、热敏打印机、激光打印机、喷墨打印机、3D 打印机以及高速行式打印机等。无论是哪种类型的打印机都离不开机械装置，所以打印机的速度与处理器相比是很慢的。

超高清图形显示器往往需要播放一些细节丰富、色彩逼真和画面快速变化的高质量的视频图像，因此可以看作一种高速的输出设备。

还有一些设备，例如带触摸屏功能的图形显示器，既能够显示文字与图形，实现输出功能；又能够通过触摸屏触摸操作提供输入功能。类似地，外部大容量存储器和数据通信设备等集输入和输出两种功能于一体。在很多情况下，这种具有双重功能的设备统一称作输入/输出设备。

6. 适配器

输入设备和输出设备都属于计算机的外围设备（简称外设）。这些设备既有高速的（如磁盘阵列），也有慢速的（如键盘、鼠标）；有机械电子结构的，也有全电子的（如数据通信终端设备）。外设的种类繁多，速度和信息编码格式各异，因此不宜与高速工作的运算器、控制器和存储器等直接相连，而应该通过适配器进行缓冲和转换，保证外设能够以计算机特性所要求的形式发送和接受信息，与计算机的主机并行协调地工作。这些在计算机与和外设之间起到桥接和匹配作用的适配器又被称为输入/输出接口（I/O Interface）。由于存在多种多样的外设，所以也有多种多样的输入/输出接口。

在计算机原理中，我们通常所说的外设主要是指上述输入/输出接口（简称外设接口），而不是特定的某一种外围设备物理装置。常见的外设接口有并行接口、串行接口、I^2C 接口和 USB 接口等。

7. 互联网络（总线）

图 1.3 中的连接各个部件的互联网络称为总线（Bus）。总线是计算机中各个部件实现互联的公共通道，用于各部件之间操作命令、控制信号、数据和各类状态信息的传输和交换。按照总线上所传送的信息种类，总线可以分为数据总线 DB（Date Bus）、地址总线 AB（Address Bus）和控制总线 CB（Control Bus）三类。数据总线用于传输数据，因而是双向的；地址总线用于确定参与数据传输过程的对象，例如某个特定的存储单元的读操作或者写操作对象。地址信号由 CPU 产生并输出，因而其传输方向是单向的。控制总线传送的是控制信号（例如读信号或者写信号）和一些状态信号，以实现对计算机的运行控制和状态检测。这些信号有些是源自 CPU，也有一些是源自其他部件发往 CPU；有些是单向，也有一些是双向。作为一个整体来说，控制总线属于双向信号线。

从硬件角度看，计算机就是通过上述三类总线，将 CPU 和各个部件进行互联的一个数字系统。为了提高计算机的处理效率，针对速度快慢不同的部件，可以使用不同类型的多套总线进行互联（类似于城市道路中的快车道、慢车道和非机动车道）。计算机系统也从最初的单总线系统，扩展出双总线、三总线以及多总线系统。

将总线划分为数据总线、地址总线和控制总线三部分，并将运算器和控制器合并为 CPU 之后，使用单颗 CPU 和采用单总线架构的计算机系统硬件逻辑结构如图 1.4 所示。

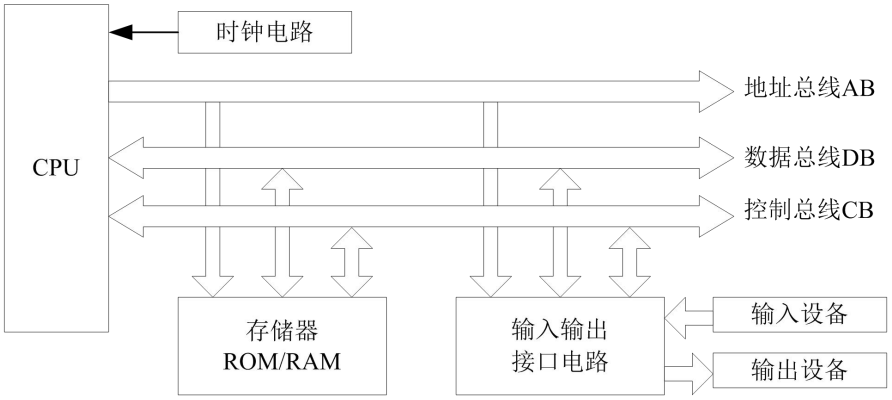

图 1.4 计算机系统硬件系统结构

1.2.2 计算机软件

在上一小节中，我们介绍了计算机中的控制器是根据指令实现对计算机各个部件的操作控制。事实上，利用电子计算机进行信息获取、存储、计算、状态监测和控制等各项操作，都需要使用能够完成多种操作的指令。这些指令以及相关的数据按照特定顺序排列后所组成的集合称为程序，凡是用于某台计算机的各种程序，统称为这台计算机的软件或者软件系统。在中国国家标准中对软件的定义是：与计算机系统操作有关的计算机程序、规程、规则，以及可能有的文件、文档及数据。

计算机软件一般分为计算机系统软件、应用软件以及介于两者之间的中间件软件。

1. 系统软件

系统软件是指能够控制计算机中各个部件协调有序工作，支持应用软件开发和运行，并且无需用户干预的各种程序集合。系统软件的主要功能是调度、监控和维护计算机的有效运行，管理计算机系统中各种部件，协调各种资源的有效利用，简化计算机的操作使用，提高计算机的工作效率。

系统软件的核心和基石是操作系统（Operating System，OS）。操作系统负责管理计算机硬件与软件资源，例如管理与配置内存、决定系统资源供需的优先次序、控制输入设备与输出设备、操作网络与管理文件系统等基本事务。操作系统同时提供了一个用户与系统交互的操作界面（User Interface，UI）。

此外，以下三类软件通常也被分类为系统软件：①用于计算机自检、故障诊断和排错等操作的服务性程序；②软件编程开发所需的汇编、编译、解释、链接和调试等程序；③数据库管理系统（Database Management System，DBMS）。

2. 应用软件

应用软件是利用计算机解决特定问题而编程开发的各种程序。例如：计算机辅助设计/辅助制造软件（CAD/CAM）、文字/电子表格处理软件、办公自动化软件、企业资源计划（Enterprise Resource Planning，ERP）软件、各类管理信息系统（Management Information System，MIS）、多媒体处理和播放软件、互联网浏览器、教育和游戏软件、网络空间安全软件等等。随着计算机应用的普及，应用软件的种类越来越多。

3. 中间件软件

中间件处于计算机系统（包括数据库系统）软件与用户应用软件之间，是分布式应用系统的基础软件。中间件为上层应用软件提供开发、集成和运行环境，并实现应用软件之间的互操作。中间件通过网络通信功能解决分布式环境下数据传输、数据访问、应用调度、系统构建、系统集成和流程管理等问题，是分布式环境下支撑应用开发、运行和集成的平台。

中间件软件的核心思想是抽取分布式系统对于数据传输、信息系统构建与集成等问题的共性要求，封装共性问题的解决方法，对外提供简单统一的接口，从而减少系统开发难度，优化系统结构和提高系统的开发效率。

4. 计算机软件的发展和演进过程

如同计算机硬件发展历程一样，计算机软件的发展也是一个循序渐进、逐步进步的过程，大体上可以分为以下几代。

1） 第一代（1946 年~1953 年）

在计算机发展的初期，工程师们直接用“0”和“1”组成的机器指令代码（机器语言）来编写程序。这种程序称为手编程序，计算机完全可以“识别”并能够执行，所以又称为目标程序或目的程序（object program）。显然，直接用“0”和“1”这种二进制符号编写程序是一件很烦琐的工作，不仅编写麻烦，难以阅读理解，而且还容易出错，调试时更是苦不堪言，这些弊端严重限制了计算机的使用和推广。

为了简化编程工作，提高程序开发效率，在此阶段的后期，工程师们使用了另外一种办法，用一些约定的文字、符号和数字按规定的格式来表示各种不同的指令，然后再用这些指令编写程序，这就是所谓的汇编语言（assembly language），亦称为符号语言。相对于机器语言，汇编语言简单直观、便于记忆，比二进制符号表示的机器语言方便了许多。但计算机只“认识”机器语言而不认识这些文字、数字、符号，为此还需要一个“翻译”程序，这就是汇编器（assembler）。用符号语言编写的程序经过汇编器翻译后，转换成用机器语言组成的目的程序，从而实现了程序设计工作的部分自动化。

汇编语言虽然比机器语言有了一定的进步，但是仍然属于一种初级语言，还存在着许多问题。首先是符号语言与数学语言的差异很大，使用汇编语言实现一个算法还是比较复杂；其次是不同计算机有不同的指令系统，使用汇编语言编写程序之前，必须先熟悉计算机的指令系统，才能使用汇编语言编写程序。

2） 第二代（1954 年~1964 年）

为了简化软件开发过程，提高编程效率，让不熟悉计算机指令系统的用户也能方便地编写各种应用软件，使用计算机解决实际问题，人们又创造了多种与数学语言相近的算法语言（algorithmic language）。

所谓算法是指为解决某个特定问题而采取的流程和步骤。而算法语言是用来表达算法的计算机程序设计语言，是算法的一种描述工具。算法语言实际上是一套利用约定的基本符号及由这套基本符号构成程序的规则，是一种接近数学语言并且与具体机器无关的通用语言，又称高级语言。算法语言直观通用，便于学习、理解和使用。

IBM 公司从 1954 年开始研制高级语言，同年 IBM 的计算机科学家约翰·巴克斯（John W.

Backus）发明了第一个用于科学与工程计算的 FORTRAN 语言；1958 年，麻省理工学院的麦卡锡（John Macarthy，1971 年图灵奖得主）发明了第一个用于人工智能的 LISP 语言；1959 年，宾州大学的霍普（Grace Hopper）发明了第一个用于商业应用程序设计的 COBOL 语言；1964 年达特茅斯学院的凯梅尼（John Kemeny）和卡茨（Thomas Kurt）发明了 BASIC 语言。

20 世纪 50 年代涌现的这些算法语言，大都是围绕单一体系结构（如 UNIVAC 和 IBM700 系列）计算机开发的，不同系统的用户之间交流还是比较困难。针对这种情况，ACM 于 1957 年成立程序设计语言委员会，由时任卡内基理工学院（卡内基·梅隆大学的前身）教授的艾伦·佩利（Alan Perlis）担任主席，于 1958 年开发了 Algol（Algorithmic language）语言，并于 1960 年 1 月修订为 Algol 60。Algol 60 是程序设计语言发展史上的一个里程碑，其特点包括局部性、动态性、递归性和严谨性，标志着程序设计语言由一种“技艺”转而成为一门“科学”，创立了程序设计语言这一研究领域，并为后来的软件自动化以及软件可靠性研究奠定了基础。1971 年出现的著名结构化编程语言 PASCAL 语言就是在 Algol 60 基础上加以扩充形成的。作为 Algol 语言以及计算机科学学科的催生者，艾伦·佩利当之无愧成为首届图灵奖的获得者。

用算法语言编写的程序称为源程序（source program）。但是，这种源程序如同汇编语言源程序一样，不能由机器直接识别和执行的，也必须给每种计算机配备一个既懂算法语言又懂机器语言的“翻译”，才能把源程序翻译为由机器语言组成的目标程序。实现上述翻译功能的程序称为编译器（compiler）。编译器的功能包括语法和语义分析，目标代码生成和优化等，有些还带有目标程序模拟运行环境和调试器等。从这一阶段开始，程序员也分化成系统程序员和应用程序员。前者负责编写诸如编译器这样的辅助工具，后者使用这些工具编写应用程序。

1956 年 IBM 公司的塞缪尔（A. M. Samuel），他也是达特茅斯会议[2]参加者，编写了一个跳棋程序。后来该程序经过不断完善，在 1959 年成为第一个有自学能力的跳棋程序。计算机从此在人工智能领域大展身手。

3） 第三代（1965 年~1970 年）

随着计算机性能的不断提高，以及计算机终端的诞生，多个用户可在终端上同时使用计算机。为了提高计算机中各个部件的运行效率，并满足多用户和多任务的应用需求，分时操作系统应运而生了。在这一阶段，计算机越来越多地用于数据处理和管理，又出现了数据库技术，以及对数据库进行统一管理的数据库管理系统 DBMS（Data Base Management System）。

1968 年，荷兰计算机科学家狄杰斯特拉（Edsgar W. Dijkstra，1972 年图灵奖获得者）发表了一篇论文，题为《GoTO 语句的害处》。文中指出，调试和修改程序的困难与程序中包含 GOTO 语句的数量成正比。从此，各种结构化程序设计理念逐渐确立起来。PASCAL 语言之父，瑞士计算机科学家尼古拉斯·沃斯（Niklaus Wirth）凭借一个著名公式：“算法+数据结构=程序（Algorithm + Data Structures = Programs）”，荣获了 1984 年的图灵奖。可见这个公式对计算机科学的影响足以和物理学中爱因斯坦的“$E=MC^2$”相媲美。

随着计算机应用的日益普及，软件数量急剧膨胀，在计算机软件的开发和维护过程中出现了一系列严重问题，出现了所谓的“软件危机”。为此，1968 年，北大西洋公约组织的计

[2] 1956 年 8 月，在美国达特茅斯学院，一群科学家聚在一起讨论了 2 个月，主题是如何让机器来模仿人类学习以及其他方面的智能。他们最终唯一达成的共识是 AI（Artificial Intelligence，人工智能）这个名词。1956 年也就成为了人工智能元年。

算机科学家在联邦德国召开会议，正式提出了“软件工程”这个名词，计算机软件工程开始成为一门独立的学科。

4） 第四代（1971 年~1989 年）

采用结构化程序设计技术不仅有 Pascal 语言，Basic 这种为第三代计算机设计的语言也被升级为具有结构化的版本。1973 年初，美国贝尔实验室的肯·汤普森（Ken L. Thompson）和丹尼斯·里奇（Dennis M. Ritchie）完成了 C 语言的主体部分的开发，并用 C 语言对他们所编写 UNIX 操作系统进行了完善。UNIX 因其所具有的开放性和可移植性，在工程应用和科学计算等领域获得了广泛应用。C 语言因其强大、高效和跨平台特性，成为应用最为广泛的底层软件开发工具，为此他们两人共同成为 1983 年图灵奖得主。

微软公司为 IBM PC 开发的 PC-DOS 以及为 IBM PC 兼容机开发的 MS-DOS，几乎成了微型计算机的标准操作系统。苹果公司在 Macintosh 计算机的操作系统中引入了鼠标的概念和点击式的图形界面，彻底改变了人机交互的方式。

20 世纪 80 年代，随着微电子和数字化声像技术的发展，在计算机应用程序中开始使用图像、声音等多媒体信息，出现了多媒体计算机。多媒体技术的发展使计算机的应用进入了一个新阶段。

在这个时期，面向没有任何计算机专业知识的用户，出现了许多运行在个人计算机上的应用软件。比较有代表性的有：第一个电子制表软件 Lotus 1-2-3，第一个文字处理软件 Word Perfect 和第一个桌面数据库管理软件 dBase II 等。

5） 第五代（1990 年~至今）

1990 年以后在计算机软件方面出现了三个著名事件，①微软公司的崛起，②面向对象（Object Oriented）的程序设计方法，③万维网 WWW（World Wide Web）的普及。

在这一阶段，微软公司的视窗操作系统 Windows 在微型计算机中取得了显著优势。微软公司的 Microsoft Office 套装软件绑定了文字处理软件 Word、电子制表软件 Excel、文稿演示软件 Power Point、数据库管理软件 Access 和其他应用程序，占据了全球办公自动化软件的大部分市场。近年来，我国金山公司自主研发的 WPS 办公套装软件，可以实现最常用的文字处理、表格制作、文稿演示等多种功能，并具有内存占用低、运行速度快、体积小巧、强大的插件平台支持、提供在线文档模板支持和海量云存储空间功能，首先在政府机关和企事业单位得到了广泛使用。

面向对象的程序设计方法因其适用于规模较大、具有高度交互性、能够反映现实世界动态内容的应用程序开发，从 20 世纪 90 年代起逐步取代了结构化的程序设计方法，成为当今最流行的程序设计技术。Java、C++、C#、Visual Basic、PowerBuild 和 Delphi 等都是面向对象程序设计语言。

万维网是建立在互联网 Internet 上的一种具有全球性、交互性、动态性和多平台性的分布式网络服务。其历史最早可追溯到 1957 年美国国防部组建的高等研究计划局（Advanced Research Projects Agency，ARPA）。1969 年，由 ARPA 资助的“阿帕网”（ARPANET）宣告建成。虽然最初只有 4 个节点，网络传输能力只有区区的 50Kbps，仅能传送纯文字信息，但“阿帕网”是世界上第一个的“Internet”。1972 年，麻省理工学院的雷·汤姆林森（ Ray Tomlinson）博士编制了一个名为 SNDMSG（即 Send Message）的电子邮件软件，email 从此成为了人们沟通与交流的有效工具。1983 年 1 月 1 日，所有连入“阿帕网”的节点开始采用

TCP/IP 协议。

1989 年夏天，英国科学家蒂姆·伯纳斯·李（Tim Berners-Lee）开发了世界上第一个 Web 服务器和客户机，并制定了一套技术规则以及创建格式化文档的 HTML 语言，能让用户通过浏览器访问遍布全球的各个站点的超文本信息。万维网技术赋予了 Internet 强大的生命力和靓丽的青春。

在这一阶段，应用软件的体系架构发生了重大转变，从集中式的主机+终端架构转变为分布式的客户机/服务器架构（Client/Server），或者浏览器/服务器模式（Browser/Server）架构。专家系统和人工智能软件也从实验室走出进入了实际应用。应用软件的普及和应用领域的不断拓展，改变了“计算机用户”这个概念的含义，以往计算机用户是拥有专业知识背景的程序员；而现在计算机用户可以是正在学习阅读的学龄前儿童，也可以是安度晚年的退休人员，是所有使用计算机的人。

随着计算机软硬件技术的蓬勃发展，基于“软件就是仪器”思想的虚拟仪器、软件定义网络 SDN（Software Defined Network）、软件定义存储 SDS（ Software Defined Storage）和软件定义数据中心 SDDC（Software Defined Data Center）等技术相继问世，以至于出现了“软件定义一切”思想。该思想的核心要义是用软件去定义系统功能，用软件给硬件赋能，实现系统运行能效的最大化。软件定义的实现基础是应用程序接口 API（Application Programming Interface）。API 解除了计算机软硬件之间的耦合关系，推动了应用软件朝着个性化方向发展，硬件资源朝着标准化方向发展，系统功能朝着智能化方向发展。故有人说：“API 之上，一切皆可编程；API 之下，‘如无必要、勿增实体’[3]。

不断完善的系统软件、日益丰富的系统开发工具、层出不穷的商品化应用程序，以及通信技术和计算机网络的飞速发展，使得人类社会正在步入“万物皆可互联、一切皆可编程”的新时代。

1.3 计算机中数的表示方法

1.3.1 进位计数制

进位计数制是指用一组固定的数字符号和特定规则来表示数的方法。我们最为熟悉的莫过于 10 进制，其原因正如亚里士多德所说，绝大多数人生下来都有十根手指头。但是在日常生活中，我们往往还遇到其他进制，例如：表示时间的时、分、秒之间是 60 进制，而小时与天之间为 24 进制。在计算机中，硬件电路普遍使用的是数字逻辑电路。而在数字逻辑电路中，只有晶体管的导通和截止、电平的高和低、开关的通和断等两种状态，因此，采用只有 0 和 1 两个数字的二进制计数制更为方便直观。使用二进制方式其他一些好处包括：运算规则简单，使用方便可靠。人们经常使用的字母、符号、图形和不同的语言文字等，在计算机中也一律用二进制编码来表示。但是二进制数的数位较长，不易书写和记忆，而人们又习惯于使用 10 进制数，因此在计算机领域里使用多种进位制来表示数，常用的有二进制、10 进制和 16 进制的表示方法。

[3] “如无必要、勿增实体”出自奥卡姆剃刀定律（Ockham's Razor）

1. 10 进制数（Decimal）

10 进制数具有 0~9 共 10 个不同的数字符号，其基数（模）为 10。10 进制数各位的权值为 10^i，其实际值可按权展开后相加获得。在 10 进制数字后面可以加后缀 D，表示该数是 10 进制数。但 D 通常省略不写。例如，10 进制数

$$456\mathrm{D} = 456 = 4\times10^2 + 5\times10^1 + 6\times10^0$$

2. 二进制数（Binary）

二进制表示法中只有 0 和 1 两个数字，其基数（模）为 2，各位的权值为 2^i，表示二进制数时，后面必须加后缀 B。例如，二进制数

$$10110\mathrm{B} = 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$$

由于二进制的数位较长，不便于阅读，所以本书在以后部分书写二进制数时，从最低位开始，每隔 4 位二进制数增加一个空格。但是程序中的二进制数不能出现空格，否则送入计算机时不能被识别。

3. 16 进制数（Hexadecimal）

由 0~9、A、B、C、D、E 和 F 共 10 个数字再加 6 个英文字母组成，其基数是 16，各位的权值为 16^i。A~F 分别表示 10 进制的 10~15。表示一个 16 位数时，后面必须加后缀 H 或者 h 以示区别。例如，

$$32\mathrm{AEh} = 3\times16^3 + 2\times16^2 + 10\times16^1 + 14\times16^0 = 12974$$

每个 16 进制数字都可以用 4 位二进制数表示（见表 1.1）。反过来，每 4 位二进制数也可以用 1 位 16 进制数来表示。由于 16 进制表示法大大缩短了数位长度，并且 16 进制与二进制之间转换方便，所以在计算机中普遍采用 16 进制来表示存储器地址以及所存储的数据。

表 1.1 10 进制、二进制和 16 进制的关系

10 进制数	二进制数	16 进制数	10 进制数	二进制数	16 进制数
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

4. 位、字节、字和字长

1） 位

更准确的称呼应该是比特，表示二进制位。位是计算机存储数据的最小单位，11010100 是一个 8 位二进制数。一个二进制位只能表示 0 和 1 两种状态（2^1）；两个二进制位可以表示 00、01、10、11 四种（2^2）状态；三位二进制数可表示八种状态（2^3）……。

2） 字节（Byte）

来自英文 Byte，音译为“拜特”，用大写的“B”表示。一个字节由 8 个二进制位构成，即一个字节等于 8 个比特（1 Byte = 8 bit）。字节是计算机中存储和读取数据的基本单位。

3） 字（word） 和字长

计算机进行数据处理时，一次存取、加工和传送的数据长度称为字。一个字通常由一个或多个（一般是字节的整数位）字节构成。例如 16 位计算机中一个字由 2 个字节组成，它的字长为 16 位；32 位计算机中的一个字由 4 个字节组成，它的字长为 32 位；64 位计算机中的一个字由 8 个字节组成，它的字长为 64 位。

1.3.2 有符号数的原码、反码和补码表示

在实际应用中，计算机需要处理的数值是有正负的，这些正数和负数在计算机中是如何表示的呢？这就是计算机中的数值编码问题。通过数值编码，可以在计算机中将正负数进行合理表示，以方便对其进行计算和处理。编码后，在计算机中以二进制形式表示的数据被称为“机器数”，机器数所代表的数值（或信息）被称为“真值”。

数值编码有多种方案，选择编码方案时既要考虑数据表示的简洁性，也要考虑计算机在处理数据时的便捷性。有符号数通常有原码、反码和补码三种编码表示方式。数学上可以证明，用二进制补码方式表示有符号数时，可用加法器实现减法运算，其加减运算处理最便捷，也最便于数字电路实现，所以在计算机中都使用补码表示有符号数。我们学习原码和反码的目的则是为了更好地探讨补码以及计算某个数的补码。

1. 原码（true form）

约定：数值 x 的原码记为$[x]_{原}$，若机器（处理器）字长为 n 位，那么数值 x 的原码定义如下：

$$[x]_{原}=\begin{cases} x & 0\le x\le 2^{n-1}-1 \\ 2^{n-1}+|x| & -(2^{n-1}-1)\le x\le 0 \end{cases} \tag{1.1}$$

原码的最高位（2^{n-1} 位）是符号位，对于正数该位为 0，对于负数该位为 1，其余各位表示这个数的绝对值，如图 1.5 所示，其中 S 位就是符号位。

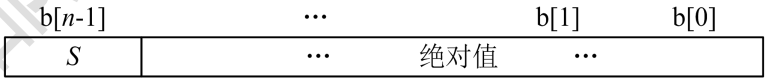

图 1.5 有符号数的原码位图

一个可存储 n bit 二进制数据的存储单元，可以看作是一个能够存放 n bit 二进制数据的容器。显见，n bit 数据容器可以存放的原码数值范围是 $-(2^{n-1}-1)\sim 2^{n-1}-1$。当 $n=8$ 时，原码可表示的数值范围是 -127～+127；当 $n=16$ 时，原码可表示的数值范围则是 -32 767～+32 767。

求某个有符号数的原码时，首先应根据目标机器数的字长 n，确认该数是否在可表示范围之内；然后再根据其符号确定 S 位（正数 $S=0$，负数 $S=1$）；最后将其绝对值转换成二进制数作为其余各位。

反之，求一个原码的真值时，首先根据其符号位判定其是正数还是负数，其余各位表示

其绝对值大小。

原码中的数值 0 有两种表示方式。以 $n=8$ 为例，数值 0 可以表示成：0000 0000（＋0）和 1000 0000（－0），但是人们习惯将 0 用＋0 表示。

例 1.1　$n=8$ 时，求 $[93]_{原}$ 和 $[-93]_{原}$。

解：93 = 5Dh = 101 1101b，　$[93]_{原}$= 0101 1101b，　$[-93]_{原}$= 1101 1101b。

原码编码方案简单，数值表示直观，与真值之间的转换方便。但是在做加/减法运算时会遇到很多麻烦，例如需要判断参加运算的两个数（操作数）的符号是否相同？是做加法还是做减法？哪个操作数的绝对值大？结果是正数还是负数？这些问题较难处理，电路实现比较麻烦，所以原码在计算机中没有得到应用。

2. 反码（inverse code）

约定：数值 x 的反码记为$[x]_{反}$，假设机器字长为 n 位，那么数值 x 的反码定义如下:

$$[x]_{反}=\begin{cases} x & 0\le x\le 2^{n-1}-1 \\ (2^n-1)-|x| & -(2^{n-1}-1)\le x\le 0 \end{cases} \qquad (1.2)$$

与原码表示相同，反码的最高位（2^{n-1} 位）是符号位，对于正数该位为 0，对于负数该位为 1，其余各位是数值位，如图 1.6 所示，其中 S 位就是符号位。同样，对于一个 n bit 容器，其可以表示的反码数值范围也是 $-(2^{n-1}-1)\sim 2^{n-1}-1$。例如，当 $n=8$ 时，可以表示的数值范围是－127～＋127。在反码中，数值 0 也有两种表示方式。例如在 $n=8$ 的反码中，0000 0000（＋0）和 1111 1111（－0），人们习惯用＋0 表示数值 0。

图 1.6 有符号数的反码位图

公式（1.2）看上去稍显复杂，但是实际转换时比较简单。求某个数的反码时，一般是先求出这个数的原码，对于正数，其反码与原码完全相同；对于负数，只需将符号位之后的“数值位”部分按位取反即可。

例 1.2　$n=8$ 时，求$[93]_{反}$和$[-93]_{反}$。

解：93 = 101 1101b，　$[93]_{原}$= 0101 1101b，　$[93]_{反}$= 0101 1101b，　$[-93]_{反}$= 1010 0010b。

求某个反码的真值也很简单，首先依据符号位（$S=0$ 或 $S=1$）确定正负号，如果是正数，反码与原码相同；如果是负数，将数值位按位取反将其转换成原码，再计算出其真值。

例 1.3　$n=8$ 时，分别求 $[0101\ 1101b]_{反}$和 $[1010\ 0010b]_{反}$的真值。

解：$[0101\ 1101b]_{反}$的最高位是“0”，表明这是一个正数，余下的 7 位数据位是 5Dh = 93，所以$[0101\ 1101b]_{反}$真值为＋93，即 93；

$[1010\ 0010b]_{反}$的最高码位是“1”，表明它的真值是负数，将 7 位数值位 010 0010b 按位取反得 101 1101b = 5Dh，所以$[1010\ 0010b]_{反}$的真值为－93。

如果使用反码表示数据，在运算时将会遇到与原码相同的问题，所以在计算机中也没有采用反码。

3. 补码（two's complement representation）

约定：数值 x 的补码记为$[x]_{补}$，假设机器字长为 n 位，那么数值 x 的补码定义如下：

$$[x]_{补}=\begin{cases} x & 0\le x\le 2^{n-1}-1 \\ 2^n-|x| & -2^{n-1}\le x\le 0 \end{cases} \qquad (1.3)$$

与原码和反码相同，补码的最高位（即 2^{n-1} 位）是符号位，对于正数该位为 0，负数则为 1，其余各位表示这个数的数值，如图 1.7 所示，其中的 *S* 位即为符号位。

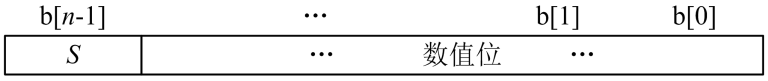

图 1.7 有符号数的补码位图

从约定公式（1.3）可以看出，正数的补码与原码完全相同；对于负数，则用模 2^n 的补数 $2^n-|x|$的二进制编码（补码）表示。*n* 位补码和原码以及反码之间的关系可以借用数轴来表示，如图 1.8 所示。

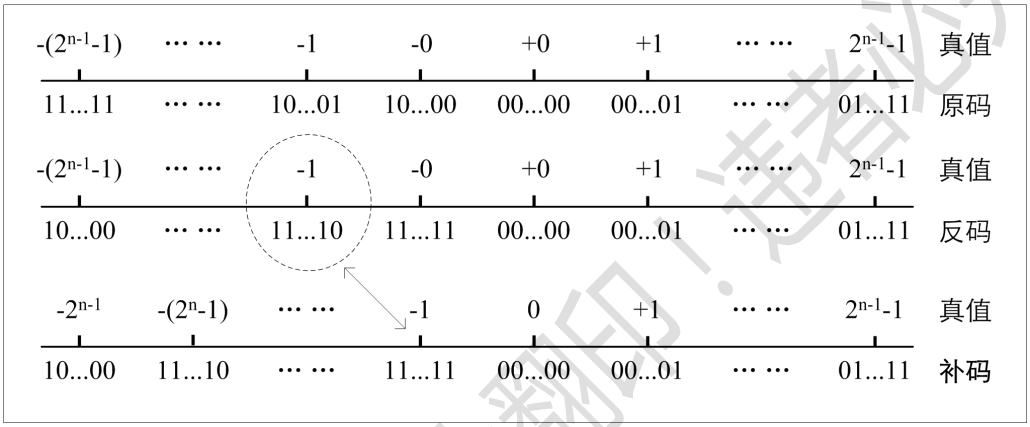

图 1.8 *n* 位补码和原码以及反码之间的关系

所谓“模（mod）”是指一个计量系统的计数范围。如时钟的计量范围是 0～11，模等于 12。计算机也可以看成一个计量机器，也有一个计量范围，即也存在一个“模”。

可以用一个带有指针的 12 小时制普通时钟为例，来帮助说明模、补数以及补码的概念。假如该时钟当前时针所指示的时间是 9 点，我们需要将其指向 2 点，我们应该怎么调整呢？一种方法是将该时针逆时针转动 7 格，9－7＝2；另外一种方法是顺时针转动 5 格，9＋5＝14，由于时钟转动一圈是 12 个小时，其中 12 在时钟上不会被显示而自动丢失，即 14－12＝2，两种方法的最终结果是一样的。

如果我们将时钟系统中的顺时针方向定义为正，逆时针方向定义为负，上例表明，当模为 12 时，－7 与＋5 等价。数学上将“模为 12”写作（mod 12），称－7 是＋5 以 12 为模的补数。记为－7＝＋5 (mod 12)。同样有：

－11＝＋1 (mod 12)；

－10＝＋2 (mod 12)；

－9＝＋3 (mod 12)；

－8＝＋4 (mod 12)；

－6＝＋6 (mod 12)；

以上表达式的一个共同特点是：一个数和它的补数的绝对值之和等于模。

从时钟的例子可以看出：

① “模”实质上是计量器产生“溢出”的量，它的值在计量器上表示不出来，计量器上只能表示出模的余数。

② 在模为 12 的时钟上，逆时针倒拨 7 格与顺时针正拨 5 格等价；或者说，减去一个数与加上这个数的补数，两者结果完全相同。这说明在任何有模的计量器，均可化减法为加法运算。

上述概念和方法完全适用于计算机。在一台字长为 n 的二进制系统中，所能表示的最大数为 2^n-1，若再加 1 则结果为 2^n，超过了数据容器（存储单元）的计量范围，最高位将会溢出从而丢失。所以 n 位二进制系统的模为 2^n。在这样的系统中，减法问题可以转化成加法问题，只需把减数用相应的补数表示即可。把补数用到计算机对数的处理上，就是补码。

对比反码和补码的约定公式（1.2）和（1.3），或者从图 1.8 所示的补码和原码以及反码关系中，我们不难看出某个负数的补码就等于它的反码加“1”。

从补码的约定公式（1.3）和图 1.8 中还可以看出，对于一个 n 位的二进制数据容器，其可表示的补码数值范围为 $-2^{n-1}\sim 2^{n-1}-1$。例如 $n=8$ 时，补码可表示的数值范围为 -128~127。

说明：补码中用 2^{n-1}（-0）约定为 -2^{n-1} 的补码。如 $n=8$ 时，用 1000 0000 表示 -128 的补码。这是一个边界特例，目的是为了充分利用编码空间。

例 1.4　$n=8$ 时，分别求出 93 和 -93 的补码。

解：93 的补码$[93]_{补}=[93]_{原}=5Dh=0101\ 1101b$

求 -93 的补码应先求$[-93]_{原}=1101\ 1101b$，再求$[-93]_{反}=1010\ 0010b$，然后$[-93]_{反}+1$，得到：$[-93]_{补}=1010\ 0011b$。

求一个负数的补码还有一种较为简单的方法（略去数学证明），具体步骤是：首先将该负数的绝对值用 n 比特字长的二进制表示，然后自右向左（从低位向高位）扫描该二进码，将遇到的首个 1 以及之前的所有的 0 保持不变，对之后的各位按位取反，结果就是该负数的补码。

例 1.5　$n=8$，求 -76 的补码。

解：首先求 -76 的绝对值 76 的二进制码，76 = 4Ch = 0100 1100b；再求 -76 的补码，按上述原则从最右边（第 0 位）开始向左扫描，在第 2 位遇到首个 1，第 2 位到第 0 位保持不变，第 7 位到第 3 位按位取反，得$[-76]_{补}=1011\ 0100b$。

若已知 p 的补码$[p]_{补}$，如何求其真值呢？根据补码（补数）的定义，有$[[p]_{补}]_{补}=[p]_{原}$。对于正数，p 的补码等于原码，计算方法不再赘述；对于负数，转换方法是保持符号位 S 不变，其余数值位按位取反后加 1，结果即为$[p]_{原}$。求负数原码的真值也不再赘述。

例 1.6　n=8 时，求$[1011\ 0100]_{补}$的真值。

解：$[1011\ 0100]_{补}$的最高位为 1，表明它的真值是负数，先求其原码；1011 0100 除最高位 1 保持不变之外，其余各位按位取反，得 1100 1011；然后再加 1，得$[1100\ 1100]_{原}$，其真值为 -4Ch = -76。

观察表 1.2 中的各项数据，进一步学习和领会补码的含义和计算过程。

表 1.2　求负数–76 的补码运算

数据	编码表示
数据 76 的二进制表示	0100 1100b
正数：$[76]_{补}=[76]_{反}=[76]_{原}$	0100 1100b
负数：-76 的补码即$[-76]_{补}$	1011 0100b
在字长为 8 位的机器中，$[-76]_{补}+[76]_{补}$	1 0000 0000b（最高位丢弃后结果正确）

本节最后，表 1.3 给出 8 bit 二进制机器数在原码、反码和补码编码体制中对应的真值，从中可以看出它们的最大表示范围。

表 1.3 8 bit 二进制机器数对应的原码、反码和补码真值

8 bit 二进制机器数	无符号二进制数(真值）	有符号二进制数（真值）		
		原码	反码	补码
0000 0000	0	＋0	＋0	0
0000 0001	1	＋1	＋1	＋1
0000 0010	2	＋2	＋2	＋2
0000 0011	3	＋3	＋3	＋3
…	…	…	…	…
0111 1110	126	＋126	＋126	＋126
0111 1111	127	＋127	＋127	＋127
1000 0000	128	－0	－127	－128
1000 0001	129	－1	－126	－127
1000 0010	130	－2	－125	－126
…	…	…	…	…
1111 1101	253	－125	－2	－3
1111 1110	254	－126	－1	－2
1111 1111	255	－127	－0	－1

1.3.3 补码的运算、溢出及判断方法

在计算机中，无论是正数或负数，无符号数或有符号数，数值或多媒体数据等等，CPU 在进行数据处理时，将所有数据均当作普通的二进制数，不去识别这些数据的具体类别和含义。如何看待和理解这些数据，则属于系统设计师和计算机使用者的职责。事实上，CPU 只是一个仅能对二进制数进行加、减等算术运算，或者进行与、或、非和左右移位等逻辑运算的二进制码处理器。即使是数据容器中的最高位 *S* 位，CPU 也不认为是符号位，也只是当作普通的二进制数参与运算。这样就带来一个问题，CPU 按照普通二进制码的运算规则对有符号数进行加减运算，怎样才能保证结果的正确性？补码正是为了满足上述要求而采用的一种编码体制，这也是我们学习和研究补码的一个主要原因。

1. 补码的加法运算

数学上可以证明，在编码长度为 n 的补码值域内，对于有符号数 p 和 q，有

$$[p+q]_{补}=[p]_{补}+[q]_{补} \tag{1.4}$$

亦即只要运算结果的值不超过补码可以表示的范围，两个有符号数之和的补码等于这两个数的补码之和，运算结果以补码形式表示。

以下通过几个示例说明上述结论，不做数学证明。

例 1.7　$n=8$ 时，$p=49$，$q=76$，分别计算 $[p+q]_{补}$ 和 $[p]_{补}+[q]_{补}$，并对比结果。

解：由于 $n=8$，补码的值域为 [－128，127]。

① 计算 $[p+q]_{补}$，先求和再求补码：

$$p+q = 49+76 = 125 \in [-128, 127]$$

$$[p+q]_{补} = [125]_{补} = [7Dh]_{补} = 0111\ 1101b。$$

② 计算$[p]_{补}+[q]_{补}$，先求补码再相加：

$$[p]_{补} = [49]_{补} = 0011\ 0001b$$

$$[q]_{补} = [76]_{补} = 0100\ 1100b$$

$$[p]_{补}+[q]_{补} = 0011\ 0001b+0100\ 1100b = 0111\ 1101b。$$

对比①和②，两种方法的计算结果完全相同。

例 1.8　$n=8$ 时，$p=49$，$q=-76$，分别计算 $[p+q]_{补}$ 和 $[p]_{补}+[q]_{补}$，并对比结果。

解：① 计算$[p+q]_{补}$，先求和再转换成补码：

$$p+q = 49+(-76) = -27 \in [-128, 127],$$

$$[p+q]_{补} = [-27]_{补} = 1110\ 0101b$$

求$[-27]_{补}$的过程不再赘述。

② 计算$[p]_{补}+[q]_{补}$，先求补码再相加：

$$[p]_{补} = [49]_{补} = 0011\ 0001b$$

$$[q]_{补} = [-76]_{补} = 1011\ 0100b$$

$$[p]_{补}+[q]_{补} = 0011\ 0001b+1011\ 0100b = 1110\ 0101b。$$

对比①和②，两种方法运算结果完全相同。

例 1.9　$n=8$ 时，$p=-49$，$q=-76$，分别计算 $[p+q]_{补}$和 $[p]_{补}+[q]_{补}$，并对比结果。

解：① 计算 $[p+q]_{补}$，先求和再转换成补码：

$$p+q = (-49)+(-76) = -125 \in [-128, 127],$$

$$[p+q]_{补} = [-125]_{补} = 1000\ 0011b$$

求$[-125]_{补}$的过程不再赘述。

② 计算$[p]_{补}+[q]_{补}$，先求补码再相加：

$$[p]_{补} = [-49]_{补} = 1100\ 1111b$$

$$[q]_{补} = [-76]_{补} = 1011\ 0100b$$

$$[p]_{补}+[q]_{补} = 1100\ 1111b+1011\ 0100b = 1\ 1000\ 0011b$$

运算结果的最高位 1 超出了 8 位数据容器的表示范围，溢出后数据容器保留的数据是 1000 0011b = $[-125]_{补}$，与方法①的结果相同。

2. 补码的减法运算

数学上可以证明，在编码长度为 n 的补码值域内，对于有符号数 p 和 q，有：

$$[p-q]_{补} = [p]_{补}-[q]_{补} = [p]_{补}+[-q]_{补} \qquad (1.5)$$

公式（1.5）表明，只要运算结果的值不超过补码可以表示的范围，两个有符号数之差的补码等于这两个数的补码的差，也等于被减数的补码加上减数的负数的补码之和，运算结果以补码形式表示。

以下通过几个示例说明上述结论，不做数学证明。

例 1.10　$n=8$ 时，$p=49$，$q=76$，分别计算$[p-q]_{补}$、$[p]_{补}-[q]_{补}$ 和 $[p]_{补}+[-q]_{补}$，并对比结果。

解：由于 $n=8$，补码的值域为$[-128, 127]$，$p-q = 49-76 = -27 \in [-128, 127]$。

① 计算$[p-q]_{补}$：

因为 $p-q=-27$，所以 $[p-q]_{补}=[-27]_{补}=[-1Bh]_{补}=1110\ 0101b$。

② 计算$[p]_{补}-[q]_{补}$：

$$[p]_{补}=[49]_{补}=0011\ 0001b$$

$$[q]_{补}=[76]_{补}=0100\ 1100b$$

按照二进制运算规则，

$$[p]_{补}-[q]_{补}=0011\ 0001b-0100\ 1100b=1110\ 0101b$$

其中最高位向前发生了借位。

③ 计算$[p]_{补}+[-q]_{补}$：

$$[p]_{补}=[49]_{补}=0011\ 0001b$$

$$[-q]_{补}=[-76]_{补}=1011\ 0100b$$

$$[p]_{补}+[-q]_{补}=0011\ 0001b+1011\ 0100b=1110\ 0101b。$$

对比三种计算方法结果，有：$[p-q]_{补}=[p]_{补}-[q]_{补}=[p]_{补}+[-q]_{补}$。

例 1.11　$n=8$ 时，$p=-49$，$q=76$，分别计算$[p-q]_{补}$、$[p]_{补}-[q]_{补}$和$[p]_{补}+[-q]_{补}$，并对比结果。

解：$p-q=-49-76=-125\in[-128，127]$

① 计算$[p-q]_{补}$：

因为 $p-q=-125$，所以 $[p-q]_{补}=[-125]_{补}=[-7Dh]_{补}=1000\ 0011b$。

② 计算$[p]_{补}-[q]_{补}$：

$$[p]_{补}=[-49]_{补}=1100\ 1111b$$

$$[q]_{补}=[76]_{补}=0100\ 1100b$$

按照二进制运算规则，

$$[p]_{补}-[q]_{补}=1100\ 1111b-0100\ 1100b=1000\ 0011b$$

③ 计算$[p]_{补}+[-q]_{补}$：

$$[p]_{补}=[-49]_{补}=1100\ 1111b$$

$$[-q]_{补}=[-76]_{补}=1011\ 0100b$$

$$[p]_{补}+[-q]_{补}=1100\ 1111b+1011\ 0100b=1\ 1000\ 0011b$$

结果超出 8 位数据容器的最大表示范围，最高位溢出后，保留的结果为 1000 0011b。

对比三种计算方法结果，有：$[p-q]_{补}=[p]_{补}-[q]_{补}=[p]_{补}+[-q]_{补}$。

以上示例表明，采用补码编码方式时，无论操作数是正数是负数，对于减法运算，只要运算结果在补码的值域范围内，都有：$[p-q]_{补}=[p]_{补}-[q]_{补}=[p]_{补}+[-q]_{补}$。

3. 补码运算的溢出问题

在以上所讨论的补码运算示例中，都有一个限制条件，运算结果不能超出补码的值域。如果不满足这个条件，结果又会怎样呢？

例 1.12　$n=8$ 时，$p=67$，$q=76$，计算$[67]_{补}+[76]_{补}$，并对运算结果进行分析。

解：两个操作数的真值相加：$p+q=67+76=143$，超出了 n 位补码的值域。

两个操作数的补码相加：$[67]_{补}+[76]_{补}=0100\ 0011b+0100\ 1100b=1000\ 1111b$。

运算结果的最高位符号位 $S=1$，是个负数，不是预期的 143。

为什么出现这样的结果呢？原因是运算结果 143 超出了 8 位补码所能表示的数值范围[-128，127]。对于运算结果超出了数据容器的表示范围，导致数据容器内的数据出现错误，这种现象称为溢出（overflow）。在原码和补码体制内同样存在溢出现象，例如，127+1 = 128，

当字长 $n = 8$ 时，原码运算为$[\ 127\ ]_{原}+[\ 1\ ]_{原}= 0111\ 1111b+0000\ 0001b = 1000\ 0000\ b = [\ -0\ ]_{原}$，结果显然是错误的，其原因也是运算结果超出了 8 位原码的值域。无论是原码、反码还是补码，只要运算结果出现溢出，结果就不能采用。为了避免出现这种情况，在计算机算法设计时必须进行特殊处理。

4. 补码运算溢出的判断方法

计算机在处理采用补码表示的有符号数时，如何判断运算结果是否发生了溢出？计算机并不是（也不能）根据溢出的定义去判断运算结果是否发生了溢出，而是根据运算过程中某些位是否发生进位及其组合情况进行判断的，并将判断结果用状态标志位表示，供程序设计者使用。

如前所述，计算机在处理数据时，将所有数据都不加区分地当作无符号二进制编码数据，并把减法运算转换成相应的加法运算。在运算过程中，假设数据容器的最高位（对应补码或原码或反码中的 S 位）出现了向前进位，我们用 $CP = 1$ 记录这一事件，否则 $CP = 0$。假设数据容器的次高位（对应补码或原码或反码中数值位的最高位）向最高位发生了进位，我们用 $CF = 1$ 记录这一事件，否则 $CF = 0$。

数学上已经证明，当且仅当 $CP \oplus CF = 1$ 时[4]，二进制数运算发生了溢出，如果 $CP \oplus CF = 0$，则没有发生溢出。

我们略去数学证明，仅给出 4 个示例来验证上述结论的正确性。

例 1.13　计算 (－1Fh)＋(－4Ah)。

解：$[\ -1Fh\ ]_{补}= 1110\ 0001b$，$[\ -4Ah\ ]_{补}= 1011\ 0110b$，列算式如下：

$$\begin{array}{r} 1110\ 0001 \\ +\ 1011\ 0110 \\ \hline 1\ 1001\ 0111 \end{array}$$

运算过程中，最高位向前发生了进位，$CP = 1$；次高位也向前发生了进位，$CF = 1$；因此 $CP \oplus CF = 0$。另外一方面，(－1Fh)＋(－4Ah) = (－69h) = －105 ∈ [－128，127]，没有发生溢出。

例 1.14　计算 (＋6Eh)＋(－7Ch)。

解：$[+6Eh\ ]_{补}= 0110\ 1110b$，$[\ -7Ch\ ]_{补}= 1000\ 0100b$，列算式如下：

$$\begin{array}{r} 0110\ 1110 \\ +\ 1000\ 0100 \\ \hline 1111\ 0010 \end{array}$$

运算过程中，最高位和次高位均没有发生进位，$CP = CF = 0$；$CP \oplus CF = 0$，另外一方面，(＋6Eh)＋(－7Ch) = (－0Eh) = －14 ∈[－128，127]，没有发生溢出。

例 1.15　计算 5Eh＋37h。

解：$[\ 5Eh\ ]_{补}= 0101\ 1110b$，$[\ 37h\ ]_{补}= 0011\ 0111b$，列算式如下：

$$\begin{array}{r} 0101\ 1110 \\ +\ 0011\ 0111 \\ \hline 1001\ 0101 \end{array}$$

运算过程中，最高位没有发生进位，$CP = 0$；次高位发生了进位，$CF = 1$；$CP \oplus CF = 1$，

[4] ⊕为异或运算

提示出现了溢出。事实上，5Eh 和 37h 是两个正数，二进制相加后最高位是 1，结果却成为了一个负数，其原因是 5Eh＋37h = 95h = 149 ∉ [－128，127]，所以发生了溢出。

例 1.16 计算 (－62h)＋(－3Bh)。

解：$[-62h]_{补}$= 1001 1110b，$[-3Bh]_{补}$= 1100 0101b，列算式如下：

$$\begin{array}{r} 1001\ 1110 \\ +\ 1100\ 0101 \\ \hline 1\ 0110\ 0011 \end{array}$$

运算过程中，最高位向前发生了进位，*CP* = 1；次高位向最高位没有发生进位，*CF* = 0；*CP* ⊕ *CF* = 1，提示出现了溢出。事实上，－62h 和－3Bh 是两个负数，二进制相加后容器里的最高位是 0，结果却成为了一个正数。其原因是 (－62h)＋(－3Bh) = (－9Dh) = －157 ∉ [－128，127]，所以发生了溢出。

CPU 在执行一条运算指令之后，将根据 *CP* ⊕ *CF* 的结果判断是否发生了溢出，并及时更新相关状态标志位。后续指令可以据此确定下一步程序执行的方向。例如，在 Intel 80x86 系列 CPU 中，若运算过程发生溢出，CPU 内部的状态标志位 OF = 1。如果参与运算的数被看作有符号数，运算结果就不能使用，必须要进行特别处理。

1.3.4 定点数和浮点数

在选择计算机的数据表示方式时，需要考虑如下几个问题：①需要表示的数据类型（小数、整数、实数和复数等）；②数的符号；③数值的范围；④数值的精确度。

计算机中常用的数据表示格式有两种：定点格式与浮点格式。采用定点格式的数据也称为定点数。定点数能够表示的数值范围较小，所需的硬件处理电路比较简单。采用浮点格式的数据称为浮点数。浮点数可以表示的数值范围很大，所需的硬件电路较为复杂。

1. 定点数的表示方法

所谓定点格式，就是在计算机中，所有数据的小数点位置固定不变。由于小数点位置在约定的固定位置，无需再使用符号“.”表示小数点。虽然在原理上小数点可以固定在任意位置，但是定点数通常将数据表示成纯小数或纯整数。在现代计算机中多采用定点纯整数，所以定点数运算又称为整数运算。

假设数 *x* 是一个 *n* bit 的二进制定点纯整数，则小数点位于最低位的右边。如果数 *x* 是无符号数，则数 *x* 的表示范围为：$0 \le x \le 2^{n}-1$。

2. 浮点数的表示方法

自然界中有许多量的数值相差巨大，例如电子的质量（9.11×10^{-23}g）和地球的质量（5.96×10^{27}g），在计算机中无法直接用定点数表示这类数值的范围。一种可行的方法是对它们取不同的比例因子，使其数值部分的绝对值小于 1，即

$$9.11 \times 10^{-28} = 0.911 \times 10^{-27}$$

$$5.96 \times 10^{27} = 0.596 \times 10^{28}$$

以上两个比例因子 10^{-27} 和 10^{28} 应分别存放在计算机中的某个存储单元，以便于在运算后恢复原有的表示方法。

这给了我们一个启发，在计算机中可以把一个数的有效数字和数的范围分别予以表示。

在这种表示方式中，一个数的小数点位置随着比例因子的不同而在一定范围内浮动，所以这种表示法称为浮点表示法。采用浮点表示法的数称为浮点数。

事实上，任意一个 10 进制数 N 可以表示为：$N = M \times 10^E$

同样，任意一个二进制数 N 也可以表示为：$N = M \times 2^e$

其中 M 称为浮点数的尾数，是一个纯小数。尾数给出了有效数字的位数，决定了浮点数的表示精度。E 或者 e 是比例因子的指数，是一个整数，也称为浮点数的阶码，对应小数点在数据中的位置，也决定了浮点数的表示范围。

在 IEEE（国际电气和电子工程师协会）-754 标准中，32 位浮点数的格式如图 1.9 所示。

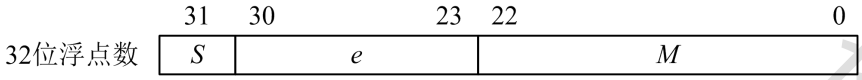

图 1.9 IEEE-754 标准中 32 位浮点数格式

图 1.9 中 S 位是 1 bit 符号位，e 是 8 bit 阶码（隐含基数为 2），M 是 23 bit 的有效位。可表示的数值范围（对应 10 进制数）为$\pm 10^{-38} \sim \pm 10^{38}$，能够满足大部分科学和工程计算的需要。

在 IEEE-754 标准中，还制定了 64 位浮点数的格式，如图 1.10 所示。

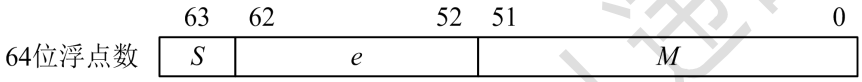

图 1.10 IEEE-754 标准中 64 位浮点数格式

64 位浮点数使用了更多的有效数值和阶码位数，从而可以得到更高的精度和更大的取指范围。所以 64 位浮点数又称为“双精度浮点数”，而 32 位浮点数被称为“单精度浮点数”。此外还有长度为 80 位的扩展双精度浮点数等。

1.3.5 计算机中的其他信息编码

计算机中的所有信息（指令、数据、文本、多媒体信息等）都是以二进制数据的形式表示的。如何使用二进制数据表示不同的信息以满足不同的应用需求（如信息的转换、传输、处理、存储、安全、容错和展现等），是信息编码学领域需要研究的内容。本小节仅介绍计算机中常用的 BCD 编码和 ASCII 编码。

1. BCD（Binary Coded Decimal）编码

由于大多数人都拥有 10 根手指，所以人们总是习惯于 10 进制计数。但是在计算机中如何表示 10 进制数呢？一种方法是将一个 10 进制数中的每一位（个位、十位、百位...）看作是一个字符，将 0～9 这些阿拉伯数字各用一个字节编码表示，然后用这些字节序列（字符串）表示这个 10 进制数，这就是稍后要介绍的 ASCII 编码方案。另外一种方法是使用二进制数表示一位 10 进制数 0～9，最常用的是使用 0000～1001 这 4 位二进制数，由于 4 位二进制数的最高位的权值是 $2^3 = 8$，次高位权值是 $2^2 = 4$，次低位权值是 $2^1 = 2$，最低位权值是 $2^0 = 1$，所以这种编码称为 8421 BCD 码，通常简称为 BCD 码。

对于任意的 10 进制数据，每一位都使用 4 位二进制数对其逐位编码。例如，10 进制数据 135，其 BCD 码为 0001 0011 0101。对于含有小数部分的 10 进制数同样处理。例如，10 进制数据 135.79，其 BCD 码为 0001 0011 0101.0111 1001。

BCD 码的优点是便于识别。BCD 码是以小数点为基准，向左每 4 位二进制数对应一位 10 进制数整数；向右每 4 位二进制数对应一位 10 进制数小数；二进制数与 10 进制数有固定的

对应关系。但是 BCD 码的缺点也很明显：①原本每 4 位二进制数可以表示 16 种状态，但是 BCD 码只使用了其中的 10 个，造成了二进制数状态空间利用率低。②计算机对所有数据进行运算时，都是按照二进制数的运算规则进行处理的。例如，2 个 10 进制 7 和 5 相加，一年级的小学生都知道结果是 12，但是按照 4 位二进制表示的 BCD 码计算，0111b＋0101b = 1100b，显然不是 12 的 BCD 码 0001 0010b。按照二进制运算法则计算 BCD 码是造成错误的原因。事实上，两个用 4 位二进制数表示的 BCD 码，按照二进制运算法则进行加法运算，只要相加结果大于 9 就会发生错误。减法运算也有类似问题。因此，BCD 码在运算后必须对结果进行调整或者修正，防止发生错误。有些处理器提供了 BCD 码的专用调整指令，如果参加运算的数是 BCD 码，在运算之后应立即调用相应的调整指令进行调整。

通常计算机中的存储器是按字节（8 bit）存放数据的，如果每个字节仅存放一位 BCD 码，该 BCD 码称为非压缩型 BCD 码。该字节的低 4 位存储 BCD 码数据，高 4 位是 0000。为了提高存储器的利用率，如果将一个字节的高 4 位和低 4 位分别存放两个 BCD 码数据，这样的 BCD 码被称为压缩 BCD 码，如图 1.11 所示。

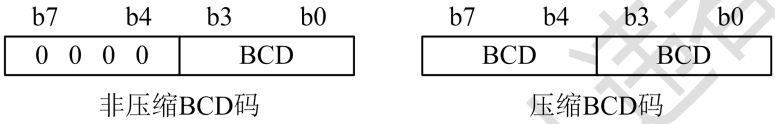

图 1.11　非压缩 BCD 码和压缩 BCD 码示意图

2. 字符的 ASCII 编码

除了数值数据以外，现代计算机还需要处理大量非数值信息，例如大小写英文字母、多民族语言文字、多媒体信息、常用的符号和控制码等。但是计算机只能处理二进制数据，因此，计算机在处理上述非数值信息之前，都必须将其编写成二进制代码。

对于大小写英文字母、阿拉伯数字和常用符号，国际上普遍采用的一种字符系统是 7 单位的 IRA 码（International Reference Alphabet，国际参考编码字符集），其美国版称为 ASCII 码（American Standard Code for Information Interchange，美国标准信息交换码）。表 1.4 列出了 7 位 ASCII 码字符编码表。

表 1.4　标准 ASCII 码编码表

$D_6D_5D_4$ / $D_3D_2D_1$	000	001	010	011	100	101	110	111
0000	NUL	DEL	SP	0	@	P	、	p
0001	SOH	DS1	!	1	A	Q	a	q
0010	STX	DS2	“	2	B	R	b	r
0011	ETX	DS3	#	3	C	S	c	s
0100	EOT	DS4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	Ω	n	~
1111	SR	US	/	?	O	_	o	DEL

在 ASCII 编码规范中，用 7 位二进制数 00 h~7Fh 表示 26 个大写英文字母、26 个小写英文字母、阿拉伯数字 0～9 和一些常用的可打印符号，共 95 个字符；再加上 33 个用于计算机控制和信息交换的控制字符（不可打印），总共 128 个元素。例如，阿拉伯数字 0～9 的 ASCII 码范围是 011 0000b～011 1001b，即 30h～39h；英文大写字符 A～Z 的 ASCII 码范围是 100 0001b～101 1010b，即 41h～5Ah；符号“＋”的 ASCⅡ编码为 010 1011b，即 2Bh。

在表 1.4 所示的 ASCII 编码规范中，如果把 20h 空格（space）也计入不可打印的控制字符，总共 34 个控制字符的含义如表 1.5 所示。

表 1.5 ASCII 码中不可打印的控制字符含义

HEX	英文以及缩写	含义	HEX	英文以及缩写	含义
0	NUL(null)	空字符	11	DC1 (dev. Cont. 1)	设备控制 1/Xon
1	SOH(start of headline)	标题开始	12	DC2 (dev. Cont. 2)	设备控制 2
2	STX (start of text)	正文开始	13	DC3 (dev. Cont. 3)	设备控制 3/Xoff
3	ETX (end of text)	正文结束	14	DC4 (dev. Cont. 4)	设备控制 4
4	EOT (end of trans.)	传输结束	15	NAK (negative ack.)	拒绝接收
5	ENQ (enquiry)	请求	16	SYN (sync. idle)	同步空闲
6	ACK (acknowledge)	收到通知	17	ETB (end of block)	传输块结束
7	BEL (bell)	响铃	18	CAN (cancel)	取消
8	BS (backspace)	退格	19	EM (end of medium)	介质中断
9	HT (horizontal tab)	水平制表符	1A	SUB (substitute)	替补
0A	LF (line feed)	换行键	1B	ESC (escape)	换码(溢出)
0B	VT (vertical tab)	垂直制表符	1C	FS (file separator)	文件分隔符
0C	FF (form feed)	换页键	1D	GS (group separator)	分组符
0D	CR (carriage return)	回车键	1E	RS (record separator)	记录分离符
0E	SO (shift out)	不用切换	1F	US (unit separator)	单元分隔符
0F	SI (shift in)	启用切换	20	SP (space)	空格

10	DLE (data link esc.)	数据链路转义	7F	DEL (delete)	删除

3. 字符串的表示方法

字符串是指连续的一串字符。在通常情况下，字符串在内存中占用连续多个字节，每个字节存放一个字符。当计算机的字长为 2 个字节或者 4 个字节时，同一个字可以存放 2 个或者 4 个字符。但是在不同类型的计算机中，同一个字中的字符存放顺序可能不同。有的是从高位字节向低位字节顺序存放，有些是从低位字节向高位字节顺序存放。例如在字长为 32 位计算机内存中，以下字符串有如图 1.12（a）和（b）两种存储方式。

#include<stdio.h>（回车符＋换行符）

int main...

其中，0Dh、0Ah 和 20h 分别是回车、换行和空格三个控制符对应的 ASCII 码，每个也需要占用一个字节内存。图 1.12（a）是一个字中的 4 个字符从高位字节向低位字节顺序存放；图 1.12（b）是一个字中的 4 个字符从低位字节向高位字节顺序存放。

高位			低位
#	i	n	c
l	u	d	e
<	s	t	d
i	o	.	h
>	(0DH)	(0AH)	i
n	t	(20H)	m
a	i	n	...

（a）

高位			低位
c	n	i	#
e	d	u	l
d	t	s	<
h	.	o	i
i	(0AH)	(0DH)	>
m	(20H)	t	n
...	n	i	a

（b）

图 1.12 字符串在存储器中的存放方式

1.4 嵌入式系统简介

1.4.1 嵌入式系统的基本概念

嵌入式系统（Embedded System）是嵌入式计算机系统的简称。在国际上，IEEE 给出的嵌入式系统定义是：嵌入式系统是用于控制、监视或者辅助设备、机器和车间运行的装置（An Embedded System is the devices used to control, monitor, or assist the operation of equipment, machinery or plants）。在中国国内，目前普遍被认同的嵌入式系统定义是：以应用为中心、以计算机技术为基础，软件硬件可裁剪，适应应用系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。

顾名思义，嵌入式系统最重要的特征就是嵌入性。所谓嵌入性就是嵌入到目标（对象）应用系统中以实现特定功能，这其中又包含两层含义：① 本系统是嵌入另一目标大系统中，成为目标系统的一个组成部分，并为实现目标系统的功能提供特定服务；② 提供特定服务的软件代码也嵌入目标系统中。

在体系结构和原理组成方面，嵌入式系统也是计算机系统，只不过是面向特定应用而特别设计的专用系统。嵌入性、专用性和计算机系统是嵌入式系统的三个基本要素或者三个主要关键字。

今天，嵌入式系统无处不在，已经渗透到了人们生活、学习和工作所使用的几乎所有的电器装置，如各种移动终端、家用电器、数码产品、汽车、通信和网络设备、视频监控和安全防范装置、工业自动化仪表和医疗仪器等等。

1.4.2 嵌入式系统的硬件

嵌入式系统硬件主要包括嵌入式处理器、存储器、嵌入式外围设备和 I/O 接口等。作为目标系统中的一个为完成特定功能而定制的专用计算机系统，嵌入式系统与一般的通用计算机系统存在显著差别，其外在形态也因目标系统的不同而呈现出千姿百态。嵌入式系统的处理器和存储系统也随不同的应用而各有特色。例如，有些嵌入式处理器着眼于高性能计算和实时处理，有些则面向低成本应用，还有许多追求的是超低功耗；大多数嵌入式系统没有硬盘或者光盘之类的大容量外部存储器，取而代之使用的是 E^2PROM 或快闪存储器等基于半导体材料的小型非易失存储器。

为了满足不同应用（目标）系统的具体需求，嵌入式系统的硬件大多是“量体裁衣”，追求高效设计，尽可能去除冗余，力争在相同的硅片面积上实现更高的性能。虽然目标系统的差异导致了嵌入式系统功能各异，在体系结构、外在形态和规格参数等方面存在许多差别，但是，所有嵌入式系统中必不可少的核心部件是嵌入式处理器。

1971 年 11 月 15 日，Intel 公司发布了其首款微处理器 Intel 4004，首次将一台计算机所需的各种元素融合在一颗单独的芯片上，这是全球最早出现的微处理器。如今，全球具有嵌入式功能和特点的处理器种类早已超过 1000 多种，包括目前仍在广泛应用的 8 位单片机（如 MCS-51、PIC8 等）、备受青睐的 32 位嵌入式微控制器（如基于 ARM 公司 32 位内核的众多产品），以及高性能的 64 位嵌入式 CPU 等。嵌入式处理器按照其功能可以分为以下四种类型。

1. 嵌入式微处理器

嵌入式微处理器 EMPU（Embedded Microprocessor Unit，也简称为 MPU）在原理和功能结构方面与通用微处理器相同。但是，为了使 EMPU 适应嵌入式系统特殊的运行环境，EMPU 在实现上进行了特别的“加固”处理。例如在工作温度范围、电压波动、电磁兼容（Electromagnetic Compatibility，EMC）、功耗和可靠性等方面，EMPU 比通用微处理器的标准要高出很多。

EMPU 内部没有存储器和外设接口，采用 EMPU 构建嵌入式系统时，需要另外选配主存储器和必要的 I/O 接口芯片，并通过总线进行互联。如果所有器件都安装在一块专门设计的主板上，就构成了一个单板机系统。与传统的工业控制计算机相比，采用 EMPU 的单板机具有集成度高、成本低、功耗小和可靠性高等优点。但在电路板上有多个通过总线互连的存储器芯片和 IO 接口芯片，系统结构呈裸露状态，技术保密性较差。较多的器件也导致可靠性无法进一步提高。目前典型的 EMPU 包括 AMD 公司的 Am 186/188、Intel 公司的 386 EX、IBM 公司的 Power PC、Motorola 公司的 68000、ARM 公司的 StrongARM 等。

2. 嵌入式微控制器

嵌入式微控制器 EMCU（Embedded Microcontroller Unit，也简称为 MCU）是将计算机的主要组成部分集成到一块芯片中，例如 ROM、RAM、总线控制逻辑和中断控制器，以及可能需要使用的定时/计数器、并行/串行接口、看门狗（Watch dog）、ADC 和 DAC 等各种部件和

外设接口，构成一个功能相对完整的计算机系统。因此，EMCU 又被称作单片机。EMCU 一般是以某款微处理器为核心，针对不同的应用需求，集成了不同的功能部件和外设接口，从而使得一个系列的 EMCU 具有多种衍生型号，旨在最大程度地匹配实际应用需求，降低系统功耗和成本。

与 EMPU 相比，EMCU 具有集成度高、产品开发周期短、成本低廉、功能丰富、功耗小和可靠性高等优点。因而 EMCU 成为嵌入式处理器中的主流，占嵌入式系统 70%以上的市场份额。比较有代表性的有 Intel 公司的 MCS-51、MCS-96/196/296 和 MCS-251 系列，Motorola 公司的 68K 系列，以及种类繁多的基于 ARM 内核的嵌入式微控制器。

虽然 MCU 和 MPU 有一定的区别，在现实中人们往往将两者混为一谈，但这并不会影响实际工作。因此，本书后续章节除了特别说明之外，对 MCP 和 MPU 也不进行严格区分。

3. 嵌入式 DSP 处理器

在通信、数字视音频、信号分析处理、模式识别、雷达和电子对抗等众多应用领域，需要进行大量的和高强度的数字信号处理计算，而通用结构设计的处理器无法满足实时性的要求。为此，专用的数字信号处理器（Digital Signal Processor，DSP）就应运而生了。DSP 内部设计了硬件乘法器和除法器等部件，采用流水线操作并提供特殊的 DSP 指令，可快速实现各种数字信号处理算法。DSP 在数字滤波、信号特征提取和分析、音视频信号编解码等方面获得了广泛应用。

DSP 在结构上分为非嵌入式和嵌入式两种，前者作为一个专用芯片单独存在，与通用处理器和其他部件通过系统总线互联，能够提供相对较高的处理性能。而后者作为一个功能部件，可以被集成在嵌入式 MCU 中。

DSP 处理器可以分为定点运算和浮点运算两大类，定点 DSP 产品已有 200 多种，其特点是功耗低、价格便宜，但是运算精度稍差；浮点 DSP 也有 100 多种，其特点是运算精度高、编程和调试方便，但是功耗较大。

DSP 的主要生产厂家有 TI 公司、Motorola 公司、ADI 公司、Lucent 公司等。目前占据市场主导地位的是 TI 公司，其三大主力 DSP 产品分别为：主要用于数字控制系统的 C2000 系列，主要用于低功耗和便携无线通信终端的 C5000 系列，以及主要用于高性能复杂通信系统的 C6000 系列。

需要指出的是，随着现场可编程逻辑门阵列 FPGA（Field Programmable Gate Array）器件技术的不断进步，其电路规模、内部硬件功能、处理速度、功耗和可靠性等指标不断提升，越来越多的场合开始使用基于 FPGA 的 DSP。这类 DSP 可以针对具体算法，由用户通过电子设计自动化 EDA（Electronics Design Automation）工具进行量身定制，不仅可以胜任以往只能依赖传统 DSP 才能完成的工作，而且可以与嵌入式 MCU 更好地进行集成。

4. 嵌入式片上系统

随着微电子技术的飞速发展，以及 EDA（Electronics Design Automation，电子设计自动化）工具的普及，把一个复杂系统集成在一块硅片上已经成为可能，这就是嵌入式片上系统 SOC（System On Chip）。

EDA 工具和硬件描述语言 HDL（Hardware Description Language）出现之后，工程师所设计的各种各样处理器、应用功能模块、外设接口和嵌入式外设等，均以不同层级的网表文件（netlist）形式呈现。这些网表文件可以构成标准部件库，直接用于最终的系统综合。全球有

许多 IC 设计公司开发了大量的和不同种类的功能部件。这些部件被称为 IP（Intellectual Property，知识产权核或知识产权模块）。其他芯片制造商、系统集成商甚至最终用户可以购买所需的 IP 授权，在系统设计和综合时直接使用这些 IP 核，从而大大简化了最终应用系统的开发工作，缩短了新产品的研制周期，并为 SOC 提供了可行性。

如今，SOC 设计师可以根据实际需求，定义所需的应用系统，按照上述方式完成系统综合、在线仿真和功能验证，然后交由芯片制造商进行流片（Tape out，属于小批量试生产），生产出 SOC 物理芯片。如果将设计后的系统下载到 FPGA 上，就形成 SOC 的另一种形式—SOPC（System On Programmable Chip）。

除了某些无法集成的器件以外，SOC 和 SOPC 将整个嵌入式系统集成到一块或几块芯片上，整个应用系统非常简洁，技术保密程度高，也有利于提高应用系统的工作可靠性和减小功耗。

1.4.3 嵌入式系统软件

嵌入式系统的软件分为嵌入式操作系统和嵌入式应用软件两部分。

1. 嵌入式操作系统

嵌入式操作系统是一个资源管理软件模块的集合，用于任务调度、存储器分配和管理、中断管理、定时器响应，以及任务之间的通信。与一般桌面操作系统（如微软公司各种版本的视窗操作系统）相比，嵌入式操作系统具有以下三个特点：

1） 实时性（Real-time）

所谓实时性，就是必须在规定的时间内准确地完成应该执行的操作。实时性的核心含义是时间的确定性，而不是单纯的速度快和时延低。

我们通常办公、学习和家庭娱乐所使用台式或者笔记本计算机，运行的都是桌面操作系统。这些计算机工作时，基本上都是根据用户通过键盘和鼠标输入的命令进行操作的。而人的反应和动作速度有快有慢，计算机工作时，对各种操作命令输入的时间没有严格的要求。

但是，许多执行控制任务的嵌入式系统对操作时间和时序有严格要求。例如在汽车、工业过程控制、核能发电和航天发射等应用中，用于控制的嵌入式系统所发出的各种指令不仅要求速度快，而且对时序也有严格的要求，否则就会酿成灾难性事故。在这类应用场景中，非实时的普通操作系统显然是无法胜任的。实时性是嵌入式系统的最大优点，在嵌入式软件中，最核心的莫过于嵌入式实时操作系统 RTOS（Real Time Operation System）。

2） 可靠性（Reliability）

可靠性也可以解读为“可依赖性”或者“可信任性”。嵌入式系统一旦投入运行，基本上无需过多的人工干预，很多情况下都是无人值守。这就要求担任系统管理和控制的嵌入式系统应具有极高的可靠性。

嵌入式操作系统应具有系统自诊断、自恢复、故障告警和故障隔离等功能。对于一个系统来说，其组成越简单，系统就越可靠；而组成越复杂，出现故障的概率就越大。嵌入式操作系统也是针对具体应用“量身定制”的，尽可能地去除了冗余的部分，人机接口多采用简单的人机会话式的字符界面。此外，由于硬件资源的限制，嵌入式操作系统小巧简洁，也有利于提高可靠性。

3） 可裁剪性

嵌入式系统是针对具体应用专门设计的，嵌入式操作系统也能够根据具体应用需求进行功能模块配置或者组态（configure），这也是嵌入式操作系统与普通操作系统的重要区别。功能模块化和配置可裁剪是嵌入式操作系统研发时必须遵循的设计原则。

2. 嵌入式应用软件

嵌入式应用软件是围绕特定的应用需求，在相应的嵌入式硬件平台上完成用户规定任务的计算机软件。复杂嵌入式系统的应用软件通常需要运行在嵌入式操作系统之上，由操作系统提供必要的支持。但是在许多简单的应用场合，也可以不需要操作系统，全部软件功能均由用户编程实现。多数嵌入式应用对成本十分敏感，除了努力降低嵌入式系统的硬件成本以外，嵌入式应用软件还要在确保逻辑准确性、时间确定性、运行可靠性等前提下，尽可能对软件结构和代码进行优化，减少所需的硬件资源开销。

1.4.4 嵌入式系统的发展概况

1. 嵌入式系统的发展历程

嵌入式系统的起源最早可以追溯到 1971 年 Intel 公司所推出的 Intel 4004。尽管 Intel 4004 仅仅集成了 2 250 个晶体管，位长只有 4 位，时钟频率 108KHz，每秒只能完成 6 万次运算，但在当时，凭借其所拥有的小型、价廉和高可靠性等特点，迅速在数字控制市场上“走红”。随后，以 Intel MCS-51 单片机为代表的嵌入式微控制器得到广泛应用，嵌入式系统开始步入繁荣发展和快速增长的时代。纵观嵌入式系统的发展历程，从系统构成的角度大致可以分为四个阶段。

第一阶段的产品主要是以嵌入式微处理器为核心的可编程控制器，其具有数据采集、处理、参数和状态显示以及伺服控制等功能，主要应用领域为工业过程控制。软件采用汇编语言编程对系统进行直接控制，没有使用操作系统。这一阶段的系统结构简单、功能单一、性能低下、存储容量较小、用户接口很少。

第二阶段的产品形态是以嵌入式微控制器为基础，以简单操作系统作为支撑的嵌入式系统。此阶段的主要特点是嵌入式微控制器种类繁多、系统运行效率高、操作系统有一定的兼容性和可扩展性。但是应用软件较为专业，通用性不强，用户接口也不够友好。

第三阶段的标志是系统普遍搭载了嵌入式操作系统。嵌入式操作系统已经较为成熟，内核精致，效率高，兼容性较好，能够运行在不同类型的微控制器上，并可提供文件管理、多任务支持、网络接口、图形用户界面（GUI）等功能，具有实时性、可扩展性、可裁剪性等特点。在这一阶段，不仅出现了大量的嵌入式应用软件，而且有丰富的应用软件开发接口（API），简化了嵌入式应用软件的开发工作。

第四阶段是以互联网和物联网为标志的嵌入式系统。在此阶段，嵌入式系统与 Internet 的深度融合，不仅能够进一步拉近人机之间的距离，还将打造一个万物互联的智能网络时代。

特别需要指出的，随着 FPGA 器件的快速发展，基于 FPGA 的 SOPC 是现阶段嵌入式系统发展的一个重要方向。SOPC 的硬件和软件均可在系统编程（in System Programming，iSP），并具有设计方式灵活、可裁剪、可扩充、可升级、高性能、低功耗和高可靠等诸多优点。

1.4.5 几种典型的嵌入式处理器

1. ARM 处理器

ARM 处理器是由总部设在英国剑桥的 ARM（Advanced RISC Machines）公司设计的。英国 ARM 公司是全球领先的半导体知识产权 (IP) 提供商，2016 年被日本软银集团收购。全世界超过 95%的智能手机和平板电脑都采用 ARM 架构。ARM 设计了大量高性价比、低耗能的 RISC（Reduced Instruction Set Computer，精简指令集计算机）处理器内核、相关技术及软件，但是 ARM 公司本身并不生产芯片，而是以相对较低技术转让费和版税将其技术授权给其他厂商。基于 ARM 架构的芯片已经占据了全球 80%以上的嵌入式处理器市场。据 ARM 公司统计，2015 年全球基于 ARM 架构的芯片年出货量约为 148 亿个，大约是 Intel x86 芯片出货量的 50 倍。ARM 公司对嵌入式系统的未来充满信心，宣称在今后的 20 年内，因物联网的推动，在全球范围内，基于 ARM 架构的芯片年出货量将达到 1 万亿颗。

ARM 处理器应用较为广泛的产品包括 ARM7 系列、ARM9 系列、ARM11 系列和 Cortex 系列。其中 ARM7 系列属于低功耗的低端产品，适用于对成本和功耗敏感的产品。ARM9 又分为适用于实时环境的 ARMTDMI、适用于开放平台的 ARM920T 以及适用于 DSP 运算及支持 Java 的 ARM9EJ 等多个产品（关于 ARM 公司的体系架构版本、产品名称以及后缀的命名规则详见本书第 5 章）。

从 ARMv7 体系结构版本发布之后，ARM 公司的产品改用 Cortex 来命名，并设计了三大分工明确的产品系列，分别是 Cortex-A、Cortex-R 和 Cortex-M，旨在满足各种不同的应用需求。A、R 和 M 正好契合公司的名称 ARM，这也挺有趣。

Cortex-A 系列中的“A”是指 Application Processor。该系列产品中基于 ARMv7 版本的处理器包括 Cortex-A5、Cortex-A7、Cortex-A8、Cortex-A9、Cortex-A12 和 Cortex-A15 等。Cortex-A 系列早期的目标市场是移动计算，后来扩展到高性能计算。Cortex-A 系列现在广泛应用于需要搭载复杂操作系统、提供交互媒体和图形体验的应用领域，如智能手机、平板电脑、数字电视以及各种服务器和网络控制器等。

Cortex-R 系列中的“R”是指 Real Time，主要面向有实时性和可靠性要求的应用，如汽车制动和安全驾驶辅助系统、防务电子设备、动力传动、大容量存储控制器和固态硬盘等。该系列产品中基于 ARMv7 版本的处理器有 Cortex-R4、Cortex-R5 和 Cortex-R7 等。

Cortex-M 系列中的“M”是指 MCU，即微控制器，其应用领域更加广泛。Cortex-M 系列主要针对成本和功耗敏感的应用，如智能测量、人机接口设备、汽车和工业控制系统、家用电器、消费性产品和医疗器械等。该系列产品中基于 ARMv7M 版本的处理器包括 Cortex-M0、Cortex-M0＋、Cortex-M1、Cortex-M3 和 Cortex-M4 等。

2012 年 ARM 公司发布了基于 64 位体系架构的 v8 版本，开始进入 64 位高性能计算领域，产品名称仍然沿用 Cortex。基于 ARMv8 版本的 ARM Cortex-A7x/A5x 系列内核已广泛应用于各种智能手机中。

全球许多著名的半导体生产厂商通过获取 ARM 公司授权，生产了众多基于 ARM 内核的商用处理器芯片，主要包括：

- ❒ 意法半导体（ST）公司的 STM32 系列微控制器；
- ❒ 恩智浦（NXP）公司的 i.MX 处理器；
- ❒ 三星（SAMSUNG）公司的 S3C44BO、S3C2400、S3C4510 和 S3C2410；

- ❒ 英特尔（Intel）公司的 StrongARM 系列和 XScale 系列；
- ❒ 德州仪器（TI）公司的 DSP＋ARM 处理器 OMAP 及 C5470/C5471。

2. MIPS 处理器

MIPS 技术公司是美国著名的芯片设计公司，致力于发展基于 RISC 体系结构的计算机系统。MIPS 技术公司所研发的 MIPS 处理器是一种有着广泛应用的 RISC 处理器。MIPS 的意思是“无内部互锁流水级的微处理器”(Microprocessor without Interlocked piped stages) ，其机制是尽量利用软件解决流水线中的数据相关问题。MIPS 设计理念较为独特，强调通过软硬件协同来提高性能，同时简化硬件设计。

在通用性方面，MIPS 公司的产品被美国硅图公司（SGI，Silicon Graphics）用于构建高性能图形工作站、服务器和超级计算机系统。在嵌入式系统方面，MIPS 是 1999 年以前全球销量最大的嵌入式处理器。如今 MIPS K 系列微处理器是目前仅次于 ARM 的用得最多的处理器，其应用领域覆盖游戏机、路由器、激光打印机、掌上电脑等各个方面。

习　　题

1.1 冯·诺依曼型计算机有哪几个主要部分？其主要设计思想是什么？

1.2 简述摩尔定律，并调研摩尔定律失效的主要原因。

1.3 什么是 CPU？什么是内存？什么是外存？什么是 I/O 接口?

1.4 计算机系统软件有哪些？请简述其主要用途。

1.5 将下列 16 进制数转换为 10 进制数，将 10 进制数转换成 16 进制数

789AH，0CEFH，1234D，7890D

1.6 将下列 10 进制数转换成 8 位二进制补码：

19，63，－1，－44，127，－127

1.7 已知：x=33，y=－64，请将 x 和 y 分别用原码和反码表示，并计算

1）$[x]_{补}+[y]_{补}$，　2）$[x]_{补}-[y]_{补}$，　3）$[x-y]_{补}$　　4）$[x+y]_{补}$

1.8 已知：

1）$[x_1]_{补}$=0000 0001，$[y_1]_{补}$=1111 1111

2）$[x_2]_{补}$=0101 1110，[y2]补=0011 0111

3）[x3]补=1001 1110，[y3]补=1100 0101

4）[x4]补=0110 1110，$[y_4]_{补}$=1000 0100

请计算$[x_i]_{补}+[y_i]_{补}$，($i=1\sim4$)，并判断结果是否出现溢出?

1.9 一个无符号的 4 位的 10 进制数，需要使用多少位二进制数才能表示？如果使用非压缩 BCD 码，则需要多少位二进制数才能表示？

1.10 在一台 32 位计算机的存储器中，从某个字的起始位置开始存放如下两行字符串：

It’s $19.9

OK！

请查表找出这两行字符串（包括空格、回车和换行符）所对应的 ASCII 码，并以表格形式标明这些 ASCII 码在存储器中两种不同的存放方式。

1.11 什么是嵌入式系统？嵌入式系统主要有哪些特点？

1.12 在你的生活和学习中，你遇到过或者听说过哪些嵌入式系统？

1.13 嵌入式 MCU 与嵌入式 MPU 主要区别是什么？

1.14 嵌入式操作系统有哪些主要特点？