
面向计算思维的 大学计算机基础

曹庆华 艾明晶 主编

万寒 孙青 欧阳元新 李莹 傅翠娇 刘禹 编

北京航空航天大学

2020 年 9 月

内 容 简 介

本书借鉴运用计算思维解决科学问题的思路，构建了一个贯穿计算思维的大学计算机课程体系。针对学生使用计算机语言解决实际问题无从下手的窘境，从问题抽象建模开始，通过从自然语言描述的问题，到形式语言描述的模型，到算法和程序实现的逐层映射，以计算思维为核心建立学生解决实际问题的导航图，通过系列案例建模、典型算法设计的讲解和分析，打开一扇利用计算机解决实际问题的科学艺术之门。全书包括：计算思维与计算机模型、问题抽象与建模、程序设计基础与数据结构、算法设计与优化、科学计算与数据处理等五章内容。

本书采用案例驱动的启发式教学方法：先通过一个实例提出问题；再给出设计思路，重点讲解设计技巧和难点。通过分析计算机解题的思路和方法，着重讲解如何运用知识将实际问题转化成机器语言的思考过程，如提炼问题、转换问题、构建模型、设计算法、用合适的程序语言描述以用计算机解决问题，以促进学生计算思维抽象和自动化本质特征的理解，掌握计算思维面向典型问题的问题求解方法。通过系列案例教学，逐层递进，从案例的分析到具体实现，促进学生对计算思维概念的理解和程序设计能力的提升。

作者结合各个知识点和重要概念，精心设计了具有趣味性和实用性、能够激发学生思考的若干教学案例，使抽象的概念具体化，从而启发和开拓学生的思维。书中大部分案例，均采用 Python 语言编程实现，使学生更好地理解运用“计算思维”求解问题的思想，掌握其方法。书中淡化操作，强调的是分析解决问题的思路和方法。同时，本书具有内容先进、层次清晰、详略得当、突出应用、图文并茂等特色。

通过本书的学习，读者将系统了解计算机基础知识，逐步理解计算思维含义及其思维方法；掌握常用数据建模方法及程序设计的基本概念、设计思路和方法，能够使用 Python 语言实现典型数据结构、进行基本的程序设计；理解算法的概念，掌握常用的算法设计思路和方法，能够采用经典算法或者自行设计算法解决实际问题；能够运用 Python 语言及其常用计算工具库进行基本的科学计算和数据处理；能够运用计算思维的一般方法分析问题和解决实际问题，为各专业的后续计算机能力和素养的需求提供必要的计算思维和能力储备，为专业领域的创新活动奠定基础。

本书既可以作为高等院校非计算机专业学生的大学计算机基础教材，也可以作为计算机基础知识及应用的自学和培训教材。

前 言

1、计算思维培养国内外现状

2006年，时任美国卡内基·梅隆大学（CMU）计算机科学系主任的周以真教授（Jeannette M. Wing）首次明确提出计算思维（Computational Thinking, CT）的概念。她指出，计算思维是运用计算机科学的基础概念进行问题求解、系统设计以及人类行为理解的涵盖计算机科学之广度的一系列思维活动；计算思维的本质是抽象（Abstraction）和自动化（Automation）。计算思维无处不在，它在无形中深刻地影响着人们的生活、学习和工作等各个方面。人们逐渐认识到计算思维独特的重要作用，以及计算思维能力培养的重要性和迫切性。

近十几年以来，国际一流大学已经对计算思维能力的培养有了充分的认识和行动。他们纷纷将计算思维纳入各自的专业与非专业基础教学中，如麻省理工学院（MIT）、加州大学伯克利分校、斯坦福大学、卡内基·梅隆大学等高校均面向全校开设了以计算思维为核心的基础课程。

国内一些高校敏感地注意到国际一流大学在计算机教育上教学理念和模式的转变，从2010年开始，陆续开展以计算思维为导向的课程改革研究，把“计算思维能力的培养”作为计算机基础教学的核心任务。经过多年的探索实践，大学计算机课程的教学体系和教学内容已发生了很大变化，课程大多都引入了计算思维，并逐渐从单纯的计算机知识传授转向有意识地培养计算思维的养成。

然而，思维毕竟是人类特有的一种精神活动，看不见摸不着。如果老师只是单纯地讲解计算思维的概念，既抽象又枯燥，就如同隔靴搔痒，学生不容易理解，更无法运用计算思维来解决实际问题——因此，计算思维能力培养如何落地？这是改革的关键所在，也是许多高校普遍感到棘手和困惑的问题。

2、基于计算思维的大学计算机课程改革

《大学计算机基础》作为高等院校非计算机专业学生的第一门计算机基础课程，在培养学生的计算机知识、素质和应用能力方面具有基础性和先导性的重要作用。在当今以计算思维培养为核心的大学计算机教育阶段，《大学计算机基础》应重新进行课程定位：《大学计算机基础》既不是一门单纯的计算机知识课程，更不是一门程序设计课程或者算法课程，而是一门思维方式和工程方法的训练课程。

《大学计算机基础》课程的教学目标是，基于OBE（Outcome based on Education，以成果为导向）教育理念，以学生为中心，坚持知识、能力、素质有机融合，培养学生复杂性思维、复杂问题求解能力等高级能力。通过课程的学习，使学生理解计算思维的基本概念和主要思维方法（如约简、抽象、分解、建模、递归等）；掌握必要的计算机基础知识；通过理论学习和实验环节，初步形成解决问题的计算思维，并且具备一定的程序设计能力。有助于学生在后续课程学习和将来的工作中，当遇到本专业领域的问题时，能够运用计算思维的思维方法以及计算机知识和技能来解决问题。因此，必须从课程体系的顶层设计入手，借鉴运用计算思维解决科学问题的思路，按照问题求解的过程来构建一个贯穿计算思维的课程体系；同时，选择合适的编程语言作为落实计算思维概念的载体，构建一个循序渐进的实验体系。通过理论学习和配套的实践环节，使学生边学习边实践，从而逐步形成解决实际问题的计算思维。

北京航空航天大学《大学计算机基础》课程团队从2014年开始，对《大学计算机基础》从课程体系和实验体系上进行了重大改革。他们通过深入分析计算思维的本质特征，研究体现计算思维的核心教学内容，参照MIT、加州大学伯克利分校和卡内基·梅隆三所大学的课程教学大纲和教学内容，确定按照计算思维基本思想→问题抽象与建模→数据结构与程序设计基础→算法设计与优化→人机交互设计→工程思维及系统设计方法的顺序，循序渐进、抽丝剥茧地讲解利用计算思维进行问题求解的方法和过程；并选择了Python语言作为落实计算思维概念的编程语言。他们制订了课程新版教学大纲，研究开发了全新

的教学内容，以Python为载体构建了一个循序渐进的、层次化实验体系。经过五年的教学实施、不断改进和完善，取得了很好的教学效果，受到学生的一致好评。

3、计算机知识与计算思维有机融合的理论教材

依据构建的以计算思维能力培养为主线的教学体系和教学内容，课程团队编写了计算机知识与计算思维有机融合的理论教材。2016年秋季，在校内印刷使用。之后，结合教学实际效果和学生的反馈，每年对教材进行进一步的修改和补充，经过几个回合的反复修改，今天终于完稿了。全书包括：计算思维与计算机模型、问题抽象与建模、程序设计基础与数据结构、算法设计与优化、科学计算与数据处理等五章内容。

第1章介绍计算思维的基本概念和主要思维方法，计算机的理论模型（图灵机）与物理实现（冯·诺伊曼计算机）；同时介绍一些必要的计算机基础知识，如信息在计算机中的表示；计算机系统基本结构。最后通过三个计算思维主要思维方法的案例，增强读者对计算思维的理解，如通信、协作、抽象、分解、关注点分离、计算效率等。

从第2章开始，各章内容循序渐进，一步步向学生揭示——针对一个实际问题，怎样运用计算思维的一般方法（抽象、建模、算法、程序、效率、工程）求解。从而使学生了解，当遇到一个实际问题时，如何对问题进行抽象（找出待解决问题中的本质），然后建立合适的模型（采用数学符号对问题的本质进行模型描述）。接下来，如果用计算机描述问题，应考虑怎样组织数据，常用的数据结构有哪些，如何用Python语言来实现这些数据结构。接着，应如何选择合适的算法（解决问题的方法和步骤）来求解问题（采用流程图、伪代码等计算机符号对模型的实现步骤进行描述）；在算法实现过程中应该如何考虑算法的效率，如何对算法进行优化。进一步，如何对数据进行处理，如何将求解的结果显示出来，如何进行人机交互，设计GUI（图形用户界面）。

第1章使学生掌握计算思维的基本概念以及在计算机科学中的体现。重点是：理论思维、实验思维和计算思维的区别与联系；计算思维的本质和主要思维方法。为什么计算机能够计算。

第2章帮助学生建立视野，学会将模糊的问题表述转化成可计算的表达方式。重点是：利用计算机进行问题求解的过程；数学建模的基本过程和基本方法。

第3章使学生掌握Python程序设计的基本语法和方法，能够对数据进行描述和存储。重点是：Python的基本语法，包括结构化数据类型，函数的定义及调用，文件操作；程序的选择结构和循环结构；数据结构的定义，主要的数据逻辑结构；利用Python实现典型数据结构的方法。

第4章通过经典算法及案例的讲解，使学生掌握一系列有用的算法和算法设计技巧。重点是：算法的特征和主要描述方法；用典型算法（穷举、递归、动态规划及贪心法等）求解问题的方法和过程；算法的时间效率和空间效率的评价方法；求解查找问题、排序问题的几种不同方法。

第5章使学生了解如何利用计算工具，包括科学计算、数值计算和绘图工具来模型化和理解数据。重点是：使用Matplotlib绘图库绘制柱状图、折线图和饼图等常用图形的方法；常用的数据处理方法——数据拟合和函数插值；图形用户界面（GUI）设计的一般方法和Tkinter主要组件的使用方法。

4、教材主要特色

（1）以应用案例引导知识、技术和方法

本教材的编写本着以应用案例引导知识、技术和方法的原则，在讲解关键技术原理的同时注重分析问题和解决问题方法的引导。我们将枯燥晦涩的原理性介绍改为以案例教学驱动，设计和开发了既联系课程内容又紧贴实际应用的案例，将抽象枯燥的理论讲解转化为一个个形象生动的案例分析，引导学生

对问题的分析思路和解决方法进行思考，从而激发学生的学习兴趣 and 探究问题的求知欲，逐渐养成学生的计算思维能力。

例如，在第1章“计算思维与计算机模型”的最后一节，通过三个典型的案例，分析了计算思维的主要方法的实际应用：通过网络协议的一个经典问题——两军问题，来分析其中蕴含的通信和协作的思维，并引入网络通信、通信协议等概念。通过计算机网络从客户端到服务端的访问过程，来分析计算机网络及其模型蕴含的抽象、分解、关注点分离、建模等计算思维重要方法。最后一个案例从RSA算法的安全性来分析计算的效率问题，使学生理解为什么说RSA算法理论上是不可攻破的？并思考RSA密钥长度是不是越大越好，对计算效率有何影响。在第4章中，针对该实例进一步讨论，在进行加密和解密时，如何通过优化算法来提高加密和解密的计算效率。

在第3章“程序设计基础与数据结构”中，在讲授最基本的线性结构——队列时，以一个有趣的儿童游戏——烫手的山芋为例，讲解一个具体问题是如何抽象成队列这种数据结构的，程序是如何巧妙使用队列的删除与插入操作实现拿着山芋的孩子在队列中的移动，从而模拟山芋怎样经过规定次数的传递后，手上拿着山芋的孩子退出游戏的。通过数据结构+演示+程序，使学生很好地理解此案例，自然就容易理解队列的性质和操作了。

(2) 内容组织深入浅出、循序渐进

我们期望在有限的篇幅内，使学生学到他们最需要、最希望了解和掌握的知识，更重要的是，通过本教材教给他们解决实际问题的方法或者说有效的途径，从而培养提高学生的自学能力和应用能力。所以既要避免面面俱到，但浅尝辄止、言之无物；又要避免就事论事、过于详细，以留给学生一些自学和自由发挥的空间；同时考虑用通俗易懂的语言，把一些计算机的基本原理和一些最新技术讲明白。

我们力图兼顾计算机基础理论的连贯性和计算机技术的实用性，所以整个教材在内容组织上注重深入浅出、循序渐进。在第1章中介绍计算机基础知识，如图灵机模型、信息在计算机中的表示、计算机系统基本结构时，尽量用通俗的语言讲清楚其原理，并通过一些小例子使学生加深理解。而在第2章~第5章介绍一些专业性和实践性较强的知识，如数学建模、算法设计、程序设计、数据处理时，则结合大量贴切、生动的实例，来讲解思路、方法和技巧。例如，在第4章中，主要介绍了枚举法、迭代法、递归法、分治、贪心法和动态规划等六种算法。这些知识对于非计算机专业的学生来讲是个难点，本书通过不同的案例介绍了这些常用算法的基本思想和实现方法，使学生通过具体实例的学习，首先了解每种算法的适用条件，即何时、在何种条件下可以使用某种算法；其次，从实例中学习基本的算法设计方法，掌握如何描述自己的问题以及如何高效、快速地解决问题；进一步，能够从时间效率、空间效率对算法复杂度进行分析，从而掌握提升算法效率的方法。

在各章中，需要特别注意的地方或是实用性很强的小技巧，本书中都会用“注意”或“提示”醒目表示，便于学生掌握重点。在每章的最后都配有相应的习题，便于学生对所学内容进行总结、思考和练习。

(3) 以计算思维为主线，各章内容前后呼应

教材编写从顶层设计入手，精心设计具有实际应用场景的案例，并使同一个案例在多章的不同章节中出现，从而达到知识的连贯和呼应。

例如，“2.4 建模的综合案例分析”中的三个案例（【案例 2.6】地铁自动购票机找零，【案例 2.7】学生选课，【案例 2.8】导航地图中的最短路径），在第2章中只讲解到如何建立数学模型；而如何求解则在第3章或4章中介绍。

比如，在“3.3 程序控制结构”的最后，详细分析了针对【案例 2.6】所建立的**线性模型**，如何通过枚举的方法逐一列举出凑够11元所有可能的组合，再从所列举的组合中选择出硬币个数最少的那个组合，从而使读者对采用循环控制结构求解线性模型有更深刻的体会；在“4.2.5 动态规划”中，则详细

分析了针对【案例 2.6】所建立的**动态规划模型**，如何将问题划分为凑 0 元、1 元、2 元、……、10 元、11 元这一系列的子问题，在子问题中使用枚举方法进行计算，通过求子问题的最优解从而计算出该问题的整体最优解。通过此案例的求解，加深读者对动态规划思想的理解。

针对【案例 2.7】所建立的模型，则通过编程实现，使读者理解，对于包含多个约束条件的模型，可以对每个判断条件定义一个函数，再采用 `while True` 语句结合 `if` 语句调用函数来求解问题。

对于【案例 2.8】所建立的模型，则在“4.2.4 贪心法”的最后，介绍了如何使用求解“最短路径问题”的经典算法——Dijkstra 算法来求解该问题，从而使读者进一步体会**贪心法**的思想和求解思路。

(4) 以Python为载体使计算思维培养落到实处

第 2 章~第 5 章的案例，均利用 Python 编程语言设计实现，使学生直观感受计算机的问题求解过程和结果。同时，课程团队参照 MIT 和加州大学伯克利分校教学内容，以 Python 为载体，围绕程序设计、数据结构、算法、数据处理、人机交互这几个课程核心内容，构建了一个层次化、循序渐进的实验体系。通过以问题求解为导向的 Python 编程实践，使学生更好地理解和运用计算思维求解问题的思想和方法，提高学生应用计算思维方法求解问题的兴趣。

参与编写本教材的，都是北京航空航天大学计算机学院从事计算机基础教育多年、有着丰富教学经验和科研经历的老师。主编曹庆华教授为北京市教学名师，北航《大学计算机基础》课程负责人。他提出了基于计算思维改革《大学计算机基础》的思路，带领课程教学团队，深入分析研究国外著名高校在计算思维培养上的举措，对标国际一流大学教学方案，大刀阔斧，对课程体系和实验体系进行了重大改进。主编艾明晶副教授为北京市高等教育学会计算机教育研究会常务理事，全国高等院校计算机基础教育研究会在线教育专业委员会常务委员，北航《大学计算机基础》课程主要负责人。她多年来潜心课程教学改革研究，带领课程团队不断改进教学内容和教学方法，努力创新，先后主编出版了 4 本《大学计算机基础》理论教材和实验教材。

本书第 1 章由艾明晶副教授编写，第 2 章由李莹副教授编写，第 3 章由万寒副教授和傅翠娇博士编写，第 4 章由孙青博士和欧阳元新副教授编写，第 5 章由艾明晶副教授和刘禹副教授编写。全书由艾明晶统稿，由曹庆华和艾明晶审稿。此外，本书在编写过程中，参考了许多优秀教材的内容，在此表示深深的谢意。

因时间仓促，加之编者水平有限，所以尽管经过了多次反复修改，但书中仍难免有疏漏和不足之处，在此恳请专家、教师和广大读者批评指正，以便于今后本教材的修订。

2020 年 7 月于北京

目 录

第1章 计算思维与计算机模型	1
1.1 理论思维，实验思维和计算思维	2
1.1.1 理论思维、实验思维、计算思维三种思维的概念	2
1.1.2 计算思维的本质和主要方法	7
1.2 计算的基础	8
1.2.1 信息的概念	8
1.2.2 什么是计算	9
1.2.3 逻辑代数与逻辑运算	10
1.2.4 数据的0和1表示与物理实现	14
1.3 计算机的理论模型与物理实现	20
1.3.1 图灵机模型	20
1.3.2 冯·诺伊曼计算机	25
1.4 信息在计算机中的表示	30
1.4.1 计算机中的数据及其单位	31
1.4.2 进位计数制及其转换	36
1.4.3 字符的编码	40
1.5 计算机系统基本结构（自学材料）	44
1.5.1 计算机硬件系统	44
1.5.2 计算机软件系统	49
1.6 计算思维方法的案例	54
1.6.1 从两军问题看通信与协作的思维	54
1.6.2 计算机网络访问过程蕴含的计算思维	57
1.6.3 从RSA算法看计算效率的思维	68
本章小结	71
习 题	73

第 1 章 计算思维与计算机模型

本章主要使读者掌握理论思维、实验思维和计算思维的区别与联系；把握计算思维的本质，了解其主要思维方法；了解为什么计算机能够计算，掌握数据的 0 和 1 表示与物理实现；了解图灵机模型及其计算思想；了解冯·诺依曼计算机的基本思想、组成及其特点；了解 7 个计算原理的核心概念。并通过一些计算思维主要思维方法的案例，比如通信、协作、抽象、分解、关注点分离、建模、计算效率等，增强读者对计算思维的理解。

1.1 理论思维，实验思维和计算思维

在本节中，主要介绍关于理论思维、实验思维和计算思维的基本概念，以及计算思维的本质和主要方法。

1.1.1 理论思维、实验思维、计算思维三种思维的概念

在开始本章的学习之前，首先来回顾一下发生在 2016 年 3 月间的一件轰动全球的大事：3 月 9 日，一场举世瞩目的围棋“人机世纪大战”在韩国首尔上演。比赛一方为谷歌公司研制的人工智能程序 AlphaGo，另一方则是围棋世界冠军、韩国名将李世乭九段。经过 3 个半小时的鏖战，李世乭九段投子认输，输掉了这五番棋中的第一场。随后在 3 月 10 日、12 日、13 日和 15 日又进行了 4 场对弈。最终，AlphaGo 以 4:1 赢得这场“人机战争”。如图 1-1 所示，为李世乭九段与 AlphaGo 对弈的照片。



图 1-1 2016 年 3 月 10 日，围棋世界冠军李世乭九段与谷歌 AlphaGo 对抗赛第二局

提示：人工智能（Artificial Intelligence, AI）是研究、开发用于模拟、延伸及扩展人的智能的理论、方法、技术及应用系统的一门新的技术科学。人工智能学科研究的主要内容包括：知识表示、自动推理和搜索方法、机器学习和知识获取、知识处理系统、自然语言理解、计算机视觉、智能机器人以及自动程序设计等方面。

人们不禁要问：围棋在对弈过程中具有海量的弈棋变化和复杂的可能性，AlphaGo 究竟是靠什么打败李世乭的呢？难道计算机也具有智能吗？

卡耐基梅隆大学机器人系博士、Facebook 公司的智能围棋 DarkForest 系统的负责人田渊栋指出，AlphaGo 主要由四部分组成：

1) 走棋网络 (Policy Network)，给定当前局面，预测/采样下一步的走棋。

走棋网络把当前局面作为输入，预测/采样下一步的走棋。它的预测不只给出最强的一手，而是对棋盘上所有可能的下一着给出一个分数。

2) 快速走子 (Fast rollout)，其目标与走棋网络一样，但在适当牺牲走棋质量的条件下，速度要比走棋网络快 1000 倍。

由于走棋网络的运行速度比较慢 (约 3 毫秒)，为快速确定下一步走棋，AlphaGo 的研制者为其设计了快速走子算法 (速度为几微秒)。快速走子还可以用来迅速评估盘面。由于天文数字般的可能局面数，围棋的搜索是毫无希望走到底的，搜索到一定程度就要对现有局面进行估分。为了寻求一个质量高且速度快的走子策略，快速走子使用了传统的局部特征匹配 (Local Pattern Matching) 加线性回归 (Logistic Regression) 的方法，达到了 2 微秒的走子速度和 24.2% 的走子准确率。


3) 估值网络 (Value Network)，给定当前局面，估计是白胜还是黑胜。

估值网络与快速走子对盘面估计是互补的，在棋局一开始时，估值网络可能起主要作用；但在面临复杂的死活或是对杀时，通过快速走子来估计盘面就变得尤为重要。

4) 蒙特卡罗树搜索 (Monte Carlo Tree Search, MCTS)，将以上这三个部分连起来，形成一个完整的系统。

MCTS 算法，对于给定的当前根节点，通过计算机模拟推演从当前根节点出发的各种可能的走法，配合高效的“剪枝”算法来控制搜索空间大小，并用演算到最后一步的结果来反过来影响当前根节点下一步棋的选择。MCTS 算法是一个多轮迭代算法，每一轮迭代都会依次经历四个阶段：选择 (Selection)，展开 (Expansion)，模拟 (Simulation) 和反向传播 (Back Propagation)。经历过足够多轮的迭代之后 (或者限定时间耗尽)，迭代结束。这时候，会从当前根节点的所有探索过的子节点中，选择一个得分最高的子节点，作为最终的下一步走法。

AlphaGo 通过深度学习 (Deep Learning) 算法，更有效地模拟了人类的下棋策略。通过不间断地自我学习，即采用强化学习 (Reinforcement Learning) 的方法自己跟自己对弈，不断优化这个策略。通过走棋网络和估值网络共同筛选出 MCTS 搜索中下一步最应该优先探索节点，AlphaGo 才能在有限的计算资源下，更快速准确地找到最佳下一步。

 **提示：**深度学习是机器学习研究中的一个新领域，其动机在于建立、模拟人脑进行分析学习的神经网络，它模仿人脑的机制来解释数据，例如图像，声音和文本。

2017 年 5 月 23~27 日，中国乌镇·围棋峰会在嘉兴桐乡市乌镇举行。AlphaGo 再次挑战人类。此次 AlphaGo 2.0 的技术原理与一年之前有着很大不同，它摒弃了人类棋谱，只靠深度学习的方式成长。以前从零训练一个 AlphaGo 要三个月，现在只需要一个星期。与 AlphaGo 1.0 相比，AlphaGo 2.0 在围棋策略和战术上都有极大的进步。

峰会期间，AlphaGo 与世界排名第一的中国九段棋手柯洁进行了三番棋对弈，最终柯洁以 0:3 落败。5 月 26 日，人机大战第二季的团队赛展开角逐，人类五位世界冠军周睿羊、聿昱廷、陈耀烨、时越、唐韦星合力抗衡 AlphaGo。但 AlphaGo 开局不久就掌握了棋局主动权，随后抗住人类战队的反击，战至 254 手，AlphaGo 执白中盘胜出。

人类所创造的工具击溃了人类，并且是在人类引以为骄傲的智慧领域，由此引发了有关人类创造物与人类自身关系的深层思考和讨论：作为人类发明的工具，人工智能机器的智力真的会超过人类吗？人工智能对人类真正的威胁是什么？机器会控制人类吗？……

AlphaGo 先后战胜李世乭、柯洁以及五位世界冠军，与其说是机器的胜利，不如说是机器与人类的共同胜利。因为，正是人类，使机器具有了这样的“智力”。为什么计算机会上围棋？归根结底，是人编写了“智能”的程序，让计算机能够通过各种先进的算法快速、正确地决策下一步棋怎么走；同时发挥了计算机自身的优势——永远不会疲劳，不会受情绪的影响，而且拥有强大的计算资源……

正如 DeepMind CEO 哈萨比斯在中国乌镇·围棋峰会开幕式的演讲中所说，“**我们最终的目的是探索新的领域，而最终的胜利属于人类**”。AlphaGo 的出现，开启了人与机器共存的新时代，也带给我们更多关于人类未来、关于智能、关于人类的“存在”的深度思考。我们要做好准备，勇敢迎接一个人机共存、人与机器共同进步和进化的时代。

下面就来了解，计算机都能够做些什么，人们又是如何运用一种独特的思维方法，利用计算机对遇到的实际问题进行求解的。

1. 科学研究的重要方法—理论和实验

为了了解和认识这个客观世界，从中发现规律，并利用客观规律造福人类，完善自我，人类一直在开展各种各样的科学研究（即通过适当的手段和途径，对事物进行深入研究，发现其本质规律的认识过程）。而**科学方法是人们在科学研究过程中采取的各种手段和途径。**

在计算机出现之前，人们从事科学研究，大多采用两种方法。一种是理论研究，一种是进行实验。它们相辅相成、取长补短，并共同完善、改进和充实着科学知识系统，在科学研究中起着不可或缺的重要作用。

随着技术的进步和发展，计算机的问世使得科学研究又产生了新的方法——计算。计算是在将现实世界映射到数字世界的过程中，形成的一种独特的方法。计算包含大问题的复杂性、效率、演化、按空间排序、按时间排序，计算的表示、表示的转换、状态和状态转换，可计算性、计算复杂性理论等核心概念。

科学思维是人脑对科学信息的加工活动，是人类思维中运用于科学认识活动的部分，是对感性认识材料进行加工处理的方式与途径的理论体系。

与理论、实验和计算相对应的科学思维是理论思维、实验思维和计算思维(Computational Thinking)。

(1) 理论及理论思维

理论是客观世界在人类意识中的反映和用于改造现实的知识系统。理论是人们按照已知的知识或认知，结合经验，经由一般化与演绎推理等方法，最终获得的合乎逻辑的推论性总结。

在数学、物理及生物学等学科，人们逐渐形成了各种理论。例如，数学学科有集合论、混沌理论、图论、数论和概率论等；物理学科有相对论、弦理论、声学理论和万物理论等；生物学有进化论等。

理论研究方法的优点是，所求得的问题的解是“精确”的。

但其缺点是，实际问题往往比较复杂，其精确解很难得到。为此，人们逐渐摸索出实验的研究方法。

理论思维是人们在认识过程中借助于概念、判断和推理等思维形式能动地反映客观现实的理性认识过程，其以数学学科为典型代表。

理论思维支撑着所有的学科领域。正如数学一样，定义是理论思维的灵魂，定理和证明是它的精髓。公理化方法是最重要的理论思维方法。

理论思维又称推理思维，它以推理和演绎为特征，通过构建分析模型和理论推导来预测和发现规律。它强调结论的严密性。

(2) 实验及实验思维

实验是人们根据一定的科学研究目的，运用科学仪器、设备等手段，在人为控制或模拟研究对象的条件下，使自然过程以纯粹、典型的形式表现出来，以便进行观察、研究，从而获取科学事实的方法。

例如，意大利数学家、物理学家和天文学家伽利略通过细心的观察和逻辑推理，大胆质疑古希腊著名的思想家和哲学家亚里士多德的论点“物体下落的快慢由它们的重量决定”是错误的，提出了“物体下落速度是均匀变化的”这一假说；并在对自由落体运动的研究中，采用以实验检验猜想和假设的科学方法，最终得出自由落体运动规律：自由落体运动是 $v_0=0$ 的匀加速直线运动，且所有物体下落的加速度都相同。

实验研究方法具有以下几个优点：可以简化和纯化研究对象；可以强化和激化研究对象；能够重复或再现研究过程和结果。

但它也存在一些限制：有的实验需要花费高昂的成本（如人力、物力、财力），比如汽车碰撞实验、蛋白质与晶体结构研究等；有的实验非常危险，具有高风险性，比如核实验，稍有疏忽就可能发生核辐射、核泄漏甚至核爆炸。

实验思维是通过观察和总结自然现象来发现规律。它强调逻辑自洽，结果可被重现，甚至可预见新的现象。其以物理学科为典型代表。

实验思维又称实证思维，它以实验、观察和总结为特征，通过直接观察获取数据、对数据进行分析进而发现规律。

意大利科学家伽利略是实验思维的先驱，他被人们誉为“近代科学之父”。

与理论思维不同，实验思维往往需要借助于某些特定的设备（科学工具），并用它们来获取数据以供以后的分析。例如，伽利略不仅设计和演示了许多实验，而且还亲自研制出不少技术精湛的实验仪器，如温度计、望远镜和显微镜等。

伽利略的实验思维方法可以分为以下三个步骤：

- ① 先提取出从现象中获得的直观认识的主要部分，用最简单的数学形式表示出来，以建立量的概念；
- ② 再由此式用数学方法推导出另一易于实验证实的数量关系；
- ③ 然后通过实验证实这种数量关系。

2. 计算与计算思维

1998年1月31日，美国副总统戈尔发表了题为“数字地球——认识21世纪我们这颗星球”的演讲。他说，“在计算机出现之前，实验和理论这两种创造知识的方法一直受到限制。实验科学家面对的研究对象太困难，不是太大就是太小，不是太快就是太慢。”“纯理论不能预报雷雨或飞机上天的空气流动之类复杂的自然现象。随着高速计算机的使用，我们才能模拟那些不容易观察到的现象。计算科学突破了实验和理论科学的局限性。”

计算是一种将单一或复数之输入值转换为单一或复数之结果的思考过程。通俗地讲，计算是指“数据”在运算符的操作下，按“规则”进行的数据变换。

计算最初指“数值计算”，它研究如何借助计算工具求得数值解答。

从20世纪40年代起，电子计算机的发明使人的计算能力得到极大提高。应用计算机处理科学研究和工程技术中数值计算的方法被称为计算科学（或科学计算）。计算机和计算方法是科学计算必不可少的两个因素。

计算手段与各学科的结合就形成了计算科学，如计算物理学、计算化学、计算生物学和计算经济学等。计算科学是一个与数学模型构建、定量分析方法以及利用计算机来分析和解决科学问题相关的研究领域。对各个科学学科中的问题，进行计算机模拟和其他形式的计算。

作为新的科学研究方法，计算已渗入各个领域的方方面面。例如，天文学家利用计算机来分析太空脉冲、星位移动；生物学家利用计算机来模拟蛋白质的折叠过程，发现基因组的奥秘；药物学家利用计算机研制治愈癌症或抑制各类细菌与病毒的药物；数学家利用计算机计算最大的质数和圆周率的更精确值；经济学家利用计算机建立国家宏观经济最优控制模型……

计算这个专门的数学概念已泛化到人类的整个知识领域，什么都要计算，计算方法已成为人们认识事物、研究问题的一种新视角、新观念和新方法。

2003年，著名计算机科学家、美国ACM前主席Peter J. Denning（彼得J.丹宁）在其论文“Great Principles of Computing（伟大的计算原理）”中指出，计算原理可以被归为7个类别，每个类别都从一个独特的视角去看待计算本身。根据Denning的观点，7个计算原理分别是：计算、通信、协作、记忆、自动化、评估和设计。

计算（Computation）是执行一个算法的过程。从一个包含算法本身的初始状态开始，输入数据，然后经过一系列中间级状态，直到达到最终也即目标状态。

通信（Communication）是指信息从一个过程或者对象传输到另一个过程或者对象。

协作（Coordination）是为确保多方参与的计算过程（如多人会话）最终能够得到确切的结论而对整

个过程中各步骤序列先后顺序进行的时序控制。

记忆 (Recollection) 是指通过实现有效搜索数据的方法或者执行其他操作对数据进行编码和组织。计算思维表述体系中的记忆是人们讨论大数据背后的原理之所在，没有“记忆”，大数据就是空谈。

自动化 (Automation) 是计算在物理系统自身运作过程中的表现形式（镜像）。什么能被（有效地）自动化是计算学科的根本问题。这里的“什么”通常是指人工任务，尤其是认知任务，可以用计算来执行的任务。计算机能够下棋吗？计算机能够解决数学问题吗？我们给出关键字能够在因特网上迅速搜索到我们想要的东西吗？计算机能够实时、准确实现汉语和英语互译吗？

评估 (Evaluation) 是对数据进行统计分析、数值分析或实验分析。

设计 (Design) 是利用学科中的抽象、模块化、聚合和分解等方法对一个系统、程序或对象等进行组织。

2006 年，时任美国卡内基·梅隆大学 (CMU) 计算机科学系主任的周以真 (Jeannette M. Wing) 教授在美国计算机权威期刊《Communications of the ACM》杂志上发表论文“Computational Thinking”，首次明确提出计算思维的概念。她指出，计算思维是运用计算机科学的基础概念进行问题求解、系统设计以及人类行为理解等涵盖计算机科学之广度的一系列思维活动。她认为，计算思维不仅仅是计算机科学家应具备，而是每个人都应该具备的基本能力。计算思维的本质是抽象和自动化。计算是抽象的自动化。自动化意味着需要某种计算机来解释抽象。这种计算机是一个具有处理、存储和通信能力的设备。

计算思维包括约简、嵌入、转化、仿真、递归、并行、抽象、分解、建模、预防、保护、恢复、冗余、容错、纠错、启发式推理、规划、学习、调度等一系列概念。

什么是计算机科学呢？计算机科学是研究计算机和可计算系统的理论方面的学科。如软件、硬件等计算系统的设计和建造，发现并提出新的问题求解策略和算法，在硬件、软件、互联网方面发现并设计使用计算机的新方式和新方法等。简单而言，计算机科学围绕“构造各种计算机器”和“应用各种计算机器”进行研究。

结合周以真教授对计算思维的定义和 Peter J. Denning 提出的 7 个计算原理，计算思维的完整表述体系主要包括计算、抽象、自动化和设计等 8 个方面，如表 1-1 所示。

表 1-1 计算思维的完整表述体系

分类	关注点	核心概念
计算	什么能计算，什么不能计算	计算的表示、表示的转换、状态和状态转换、按空间排序、按时间排序；可计算性、计算的复杂性
抽象	对象的本质特征	概念模型与形式模型、抽象层次、抽象结构、虚拟机
自动化	信息处理的算法发现	形式化方法、程序、算法、迭代、递归、搜索、推理；强人工智能、弱人工智能
设计	可靠和可信系统的构建	模块化、信息隐藏与封装、层次聚集；一致性和完备性、重用、安全性、可靠性、折中与结论
通信	不同位置间的可靠信息移动	信息及其表示、信息量、编码与解码、信息压缩、信息加密；校验与纠错
协作	多个自主计算机的有效使用	同步、并发、死锁、仲裁；事件及其处理、流和共享依赖，协同策略与机制
记忆	媒体信息的表示、存储和恢复	存储体系、对象与存储的动态绑定、层次命名、检索与索引；局部性与缓存、抖动 (trashing)
评估	复杂系统的性能评价	模型方法、模拟方法、基准测试；可视化建模与仿真、预测与评价、服务网络模型；负载、吞吐率、反应时间、瓶颈、容量规划

计算思维又称为构造思维，以计算机学科为典型代表，以设计和构造为特征，通过建立仿真的分析模型和有效的算法，利用计算工具来预测和发现规律。计算思维强调抽象和构造、可解，强调用自动方式逐步求解。

总之，理论思维、实验思维和计算思维是人类科学思维中的重要思维方式。但它们各有特点，理论思维强调推理，实验思维强调归纳，计算思维则强调自动求解。

计算思维与理论思维有相似之处，都是要建立分析模型。但不同之处在于，理论思维还需要通过理论推导来预测和发现规律。而计算思维则需要设计算法，告诉计算机怎么做，然后利用计算机自动求解，

来预测和发现规律。

1.1.2 计算思维的本质和主要方法

1. 计算思维的本质

周以真教授指出，计算思维的本质是抽象（Abstraction）与自动化（Automation）。**抽象**是指将要求解的问题形式化地表示为计算机所能理解的符号模型，进而用程序语言描述该模型。**自动化**是指计算机自动执行程序，实现问题求解。抽象是由人来完成的，自动化则是由计算机完成的。

计算思维中的抽象完全超越物理的时空观，并完全用符号来表示。而数学抽象只是其中的一类特例。

抽象层次是计算思维中的一个重要概念，它使人们可以根据不同的抽象层次，有选择地忽视某些细节，最终控制系统的复杂性；在分析问题时代，计算思维将注意力集中在感兴趣的抽象层次，或其上层、下层；因此，还应当了解各抽象层次之间的关系。

2. 计算思维的主要方法

周以真教授归纳了计算思维的七类方法，这实际上也是计算思维的核心概念。在人们利用计算机解决实际问题的过程中，常常会用到其中的一种或者几种思维方法。计算思维的七类方法描述如下：

（1）通过约简、嵌入、转化和仿真等方法，把一个看起来困难的问题重新阐释成一个我们知道问题怎样解决的方法。

（2）计算思维是一种递归思维，是一种并行处理，是一种把代码译成数据又能把数据译成代码，是一种多维分析推广的类型检查方法。

递归的思维方法是指在函数的定义中调用函数自身的方法。例如求阶乘 $F(n)=n!$ ，就可以采用递归的方法。其递归函数可以这样定义：

递归基础： $F(1)=1$ ；

递归步骤： $F(n)=n \times F(n-1)$ ， $n=2, 3, \dots$ 。

根据 $F(1)=1$ ，可以计算得到 $F(2)=1 \times F(1)=2$ ；根据 $F(2)$ ，可以得到 $F(3)=3 \times F(2)=6$ 。以此类推，依据上一个数的阶乘可以很容易计算当前数的阶乘。由此可见，递归使得一个复杂问题的解决变得简单化。

并行处理的例子：CPU 采用多核处理器，每个核可以同时工作，并行处理不同的任务，这样能够有效提高计算机的处理速度。

（3）计算思维是一种采用抽象和分解来控制庞杂的任务或进行巨大复杂系统设计的方法，是基于关注分离的方法。

（4）计算思维是一种选择合适的方式去陈述一个问题，或对一个问题的相关方面建模使其易于处理的思维方法。

建模即对问题本质采用数学符号进行模型描述。利用计算机求解问题时，一般需要先建立数学模型。因此，数学建模是读者需要学习的一个重要方法。

（5）计算思维是按照预防、保护及通过冗余、容错、纠错的方式，并从最坏情况进行系统恢复的一种思维方法。

（6）计算思维是利用启发式推理寻求解答，也即在不确定情况下的规划、学习和调度的思维方法。

（7）是利用海量数据来加快计算，在时间和空间之间、在处理能力和存储容量之间进行折衷的思维方法。

例如，现代计算机采用 Cache（高速缓冲存储器，存取速度快，但容量小）、主存（内存，存取速度较快，但容量较小，价格贵）和辅存（外存，如硬盘；存取速度较慢，但容量大、价格低）的层次化存储体系，通过组合优化不同性能的存储资源，来达到速度、容量、价格之间的一个平衡，以保证计算机整体性能。这体现了计算思维“在时间和空间之间、在处理能力和存储容量之间进行折中”的思维方法。

利用计算机自动求解，一般都要经过抽象→建模→选择或设计数据结构→选择或设计算法→编程实

现这样一个过程，而这些要素，正是本书在后续各章中要介绍的计算思维的核心方法。

1.2 计算的基础

人们发明计算机的初衷，是为了解决繁琐耗时的数学计算问题。那么，为什么计算机能够思考？或者说，为什么计算机能够计算？

计算机要实现自动计算，必须解决以下几个问题：

- (1) 数据的表示
- (2) 数据的存储及自动存储
- (3) 计算规则的表示
- (4) 计算规则的自动执行
- (5) 计算机的所有资源的管理和调度

在本节中，主要了解，计算机科学中的信息（Information）指什么，在计算机中数据是如何表示的，物理上采用什么器件实现数据的表示和存储；计算机实现计算的物理基础是什么，即组成计算机的基本元器件是什么，以及如何由基本元器件构造出更复杂的电路，进而构造出计算机。

1.2.1 信息的概念

在现实世界中，每天都产生、传播着各种各样的信息。究竟什么是信息呢？

信息指音讯、消息、通讯系统传输和处理的对象，泛指人类社会传播的一切内容。许多研究者从各自的研究领域出发，对信息给出了不同的定义。美国数学家、信息论的创始人克劳德·艾尔伍德·香农（Claude Elwood Shannon）认为，“信息是用来消除随机不确定性的东西”，这一定义被人们看作是信息的经典性定义并加以引用。控制论创始人维纳（Norbert Wiener）认为，“信息是人们在适应外部世界、并使这种适应反作用于外部世界的过程中，同外部世界进行互相交换的内容和名称”。经济管理学家则认为，“信息是提供决策的有效数据”。

科学的信息概念可以概括如下：信息是对客观世界中各种事物的运动状态和变化的反映，是客观事物之间相互联系和相互作用的表征，表现的是客观事物运动状态和变化的实质内容。人类通过获得、识别自然界和社会的不同信息来区别不同事物，认识和改造世界。

计算机科学中的**信息**通常被认为是能够用计算机处理的有意义的内容或消息，它们以数据的形式出现，如：文本（数字、字符）、声音、图形、图像、视频等。数据是信息的载体。

实际上，在现实世界中，一切信息都可以用 0 和 1 的组合来表示，也就是将现实世界的（事物）语义用特定的符号来表达，进而就可以进行基于符号的计算。

我国古代的《易经》就是将语义符号化的典型事例。太昊伏羲氏在天水卦台山始画八卦，一画开天。八卦即八个卦相，它表示事物自身变化的阴阳系统，用“—”代表阳，用“- -”代表阴。用 3 个这样的符号，按照大自然的阴阳变化平行组合，组成 8 种不同形式，可以表示 8 种语义，故称为八卦。

每卦代表自然空间中的一个事物：乾代表天，坤代表地，巽（xùn）代表风，震代表雷，坎代表水，离代表火，艮（gèn）代表山，兑代表泽。《易经》中的八卦如图 1-2 所示。

八卦可以描述世界吗？三画阴阳共有 8 种组合方式，可以表示 8 种语义；六画阴阳的组合就可以表示 64 种语义（六十四卦）。试想一下，如果采用更多的符号组合，是不是就可以表示更多的语义或事物？假如这里的阴和阳用 0 和 1 来表示，那么，世上万物都可以用 0 和 1 的组合表示出来。



图 1-2 《易经》中的八卦

1.2.2 什么是计算

如何理解计算呢？数学学科中的计算指将单一或复数之输入值转换为单一或复数之结果的思考过程。人们从小学习的算术运算就是一种简单计算，它是“数据”在运算符的操作下，按“规则”进行的数据变换。如：4+5=9，4×5=20 等。人们不断学习和训练运算符的“规则”及其组合应用，以通过计算得到正确的结果。

一个函数就是一次计算，比如对于函数 $f(x) = ax^2 + bx + c$ ，假定 a 不等于 0，可以求得其零点 $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ 。对于这类计算，人们通常是学习计算规则及其简化计算方法，应用规则来求解问题。

但是，对于有些复杂计算，人虽然知道规则，却无法得到计算结果（因为超出了人的计算能力），怎么办呢？有两种解决办法，一种方法是研究其简化的等效计算方法（数学），使人可以计算。另一种方法则是设计简单的规则，让机械代替人按照“规则”自动计算。

例如：判断丢番图方程（Diophantine Equation） $a_1x_1^{b_1} + a_2x_2^{b_2} + \dots + a_nx_n^{b_n} = c$ 是否有整数解？其中系数均为整系数。显然这是一个不定方程（未知数的个数多于方程个数）。最直接的解法是，采用枚举的方法，即一一列举出该方程所有可能的解；再将每一组可能解代入方程中，检验每个可能的解是否是可行解。但过去只能通过手工计算，比较繁琐，尤其当 n 很大时，计算量很大，难以迅速求出可行解。因此，人们逐步研究出许多方法，如因式分解法、同余法、不等式估计法等来求解不定方程。而现在，因为有了计算机，人们可以设计规则，让计算机依据规则自动进行计算，迅速求解。

广义地讲，计算是一种符号串的转换：是将输入符号串转换为输出符号串的过程。本书所提到的计算实际是指计算科学（或科学计算），是计算这个专门的数学名词被泛化后的一个概念。

计算机科学和计算科学的诞生和发展促使人们思考：

（1）什么能够被有效地自动计算？

现实世界中的诸多问题，哪些问题是自动计算的，哪些问题是可以在有限时间、有限空间内自动计算的？——这就出现了计算及计算复杂性问题。人类在寻找求解复杂问题的有效规则过程中，就出现了算法及算法设计与分析。例如人类通过研究生物行为规律而提出的各种仿生学算法，如遗传算法、蚁群算法、蜂群算法等。

（2）如何低成本、高效地实现自动计算？

如何构建一个高效的计算系统？使得能够自动计算并能在尽可能短的时间内得到计算结果？这就促进了计算机器的构建和软件系统的构建。

（3）如何方便有效地利用计算系统进行计算？

如何利用已有的计算系统，面向各行各业进行问题求解？这就促进了计算在各个领域的研究和应用。例如，在大型水坝的设计或卫星气象预报中，利用计算机的高速运算能力，来求解几十阶微分方程组、几百个线性联立方程组以及大型矩阵等，解决人工计算无法完成的复杂计算问题。利用计算机进行基因

组研究，运用大规模高效的理论模型和数值计算来识别基因组序列中代表蛋白质的编码区，破译隐藏在核酸序列中的遗传语言规律。

简单来说，计算就是对输入到输出的转换。要利用计算机进行计算，就必须给定输入和输出的描述；必须给定转换的算法；还必须考虑转换的效率。

1.2.3 逻辑代数与逻辑运算

本节将介绍与计算机密切相关的逻辑基础，包括逻辑代数相关概念，二进制数的逻辑运算。

1.2.3.1 逻辑代数的相关概念

逻辑代数是计算机重要的学科基础。因此，有必要了解与逻辑代数相关的重要概念。

1. 逻辑和逻辑代数

所谓“**逻辑**”（或**逻辑关系**），指事物间的因果关系，或者说条件与结果的关系。“因”是条件，条件之间用基本逻辑关系进行组合，根据不同的条件进行运算得到“结果”。

常见的逻辑关系有逻辑与、逻辑或、逻辑非等。

只有决定事件结果的全部条件（输入）同时具备时，结果（输出）才发生，这种因果关系叫做逻辑与（也称为逻辑乘）。例如，“小张学习成绩好，并且小提琴也拉得很好”就表达了逻辑与关系。只有这两句话都为真，整个话才为真；只要其中一句话为假，则整个话为假。

在决定事件结果的诸多条件中只要有任意一个满足，结果就会发生，这种因果关系叫做逻辑或（也称为逻辑加）。例如，“小李或者是班长，或者是团支书”就表达了逻辑或关系。只要这两句话至少有一句话为真，整个话就为真；只有两句话都为假，则整个话才为假。

只要条件具备了，结果便不会发生；而条件不具备时，结果一定发生，这种因果关系叫做**逻辑非**（也称为逻辑反）。例如，“并非闪光的东西都是金子”就表达了逻辑非关系，它是对命题“闪光的东西都是金子”的否定，也称为负命题。负命题的真假与被否定命题的真假是相反的。当被否定的命题“闪光的东西都是金子”为真时，负命题为假；当被否定的命题为假时，负命题为真。

逻辑代数是英国数学家和逻辑学家乔治·布尔（George Boole）（1815~1864）建立的，它是利用代数的方法来研究推理、证明等逻辑问题，又称布尔代数（Boolean algebra）。

逻辑代数（布尔代数）是捕获了集合运算和逻辑运算二者的根本性质的一个代数系统。它处理集合运算交集、并集、补集以及逻辑运算与、或、非。逻辑代数不仅是研究逻辑学的数学基础，也是分析和设计逻辑电路的数学基础。

逻辑代数与普通代数的相似之处在于，它们都是由变量、常量及各种运算符组成的代数系统。逻辑代数与普通代数的不同之处在于：

- ① 逻辑代数表达的是电路输入与输出间的逻辑关系，而不是数量关系。
- ② 逻辑代数中的变量和常量只能取值为 0 或 1，这里的 0 或 1 不是表示数值的大小，而是表示两种对立的关系。
- ③ 逻辑代数的基本运算为与、或、非；普通代数的基本运算为加、减、乘、除。

2. 逻辑常量与逻辑变量

当两个二进制数码表示不同的逻辑状态（通常用 1 表示真，用 0 表示假）时，它们之间可以按照指定的某种因果关系进行推理运算，这种运算称为逻辑运算。如前面提到的基本逻辑运算——逻辑与、逻辑或、逻辑非等。

在逻辑运算中其值不会改变的量称为逻辑常量。最基本的逻辑常量是 0 和 1（还有高阻 z、未知 x）。用 1 和 0 表示一个事物的两种不同逻辑状态，如一件事情的是和非、真和假、有和无、好和坏，电平的高和低，电流的有和无，灯的亮和灭，开关的闭合和断开等。这种只有两种对立逻辑状态的逻辑关系称

为二值逻辑。

在逻辑运算中其值会发生改变的量称为**逻辑变量**。逻辑变量由字母或字母加数字组成。分为原变量和反变量两种表示形式。如果原变量为： A, B, C, A_1 ，则反变量写为： $\bar{A}, \bar{B}, \bar{C}, \bar{A}_1$ 。

原变量与反变量的关系是“互非”或“互补”的，也就是说，原变量中的值与反变量中的值总是相反的。

3. 表示逻辑关系的不同方法

在逻辑代数中，可以用真值表、逻辑函数表达式、逻辑符号等不同的方法来表示逻辑关系。

假定 A, B 是两个输入逻辑变量， Y 是输出逻辑变量， A 和 B 进行逻辑与运算，得到 Y 。则这种逻辑关系可以分别描述如下。

真值表（True Table）是用 0 和 1 表示逻辑输入与输出之间全部关系的表格。

罗列出输入变量 A, B 的所有取值情况，分别计算它们执行逻辑与运算的输出 Y ，填入表格，即得到逻辑与的真值表，如表 1-2 所示。

表 1-2 逻辑与的真值表

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

用逻辑运算符把各个输入连接起来，得到输出所形成的函数表达式称为**逻辑函数表达式**。

逻辑与的逻辑函数表达式如式（1-1）所示。其中符号“ \cdot ”为逻辑与的运算符，也可以省略。

$$Y = A \cdot B = AB = A \& B \quad (1-1)$$

此外，还可以将与、或、非等各种逻辑关系用特定的图形符号来表示，这种图形符号称为**逻辑符号**。

在我国国家标准、部标准以及 IEEE 国际标准中，逻辑与的逻辑符号如图 1-3 所示。

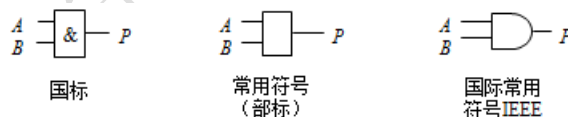


图 1-3 逻辑与的逻辑符号

1.2.3.2 逻辑运算

逻辑代数的逻辑运算分为基本逻辑运算和复合逻辑运算。

基本逻辑运算包括逻辑与、逻辑或、逻辑非三种运算。实际的逻辑问题往往比基本逻辑复杂，但都可以用与、或、非组合成的复合逻辑来实现，如与非、或非、与或非、异或、同或等。

各种编程语言中的逻辑运算符如表 1-3 所示。

表 1-3 各种编程语言中的逻辑运算符

逻辑运算	C	Python	Pascal	Java
与	&&	and	and	&&
或		or	or	
非	!	not	not	!

1. 逻辑与运算

可以用如图 1-4 所示的指示灯控制电路来形象地说明逻辑与运算。

用两个开关 A 和 B 串联来控制指示灯 P 的亮灭。规定：

输入条件（开关 A、B）：闭合表示 1，断开表示为 0；

输出结果（灯 P）：亮表示为 1，灭表示为 0。

显而易见，只有当开关 A、B 同时闭合时，指示灯 P 才会亮。

逻辑与的函数表达式为 $P = A \cdot B = AB = A \& B$ 。

逻辑与运算通常用符号“ \wedge ”或“ \cdot ”来表示。由于逻辑与的运算符借用了普通代数中的乘号“ \cdot ”，所以，逻辑与又称为逻辑乘。乘号“ \cdot ”也可以省略。



注意：这里的例子只有 2 个输入变量，但实际上，逻辑与可以有 2 个以上的输入，比如 $P = ABC$ 。

逻辑与的运算规则：只要输入中有一个 0，输出就为 0；只有输入全为 1 时，输出才为 1。用式子表示即为：

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

2. 逻辑或运算

逻辑或运算电路如图 1-5 所示。与“逻辑与”运算电路不同，图中开关 A 和 B 是并联的。只要开关 A、B 有一个闭合，指示灯 P 就会亮。

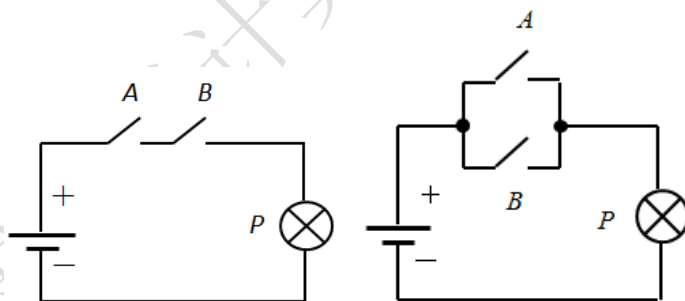


图 1-4 逻辑与运算电路

图 1-5 逻辑或运算电路

逻辑或的函数表达式为 $P = A + B$ 。

逻辑或的运算符号是“ \vee ”或“ $+$ ”。由于借用了普通代数中的加号，所以逻辑或又称为逻辑加。

逻辑或也可以有 2 个以上的输入，比如 $P = A + B + C + D$ 。

逻辑或的运算规则：

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 1$$

3. 逻辑非运算

逻辑非对单一的逻辑变量进行求反运算，它将一个二进制数据的 0 变为 1，1 变为 0。

同样，用一个指示灯控制电路来形象地说明逻辑非的概念，如图 1-6 所示。开关 A 和灯 P 并联。开关 A 闭合时，指示灯 P 不亮；开关 A 断开时，灯反而亮。由于条件满足与结果发生是相反的关系，所以称为逻辑非。

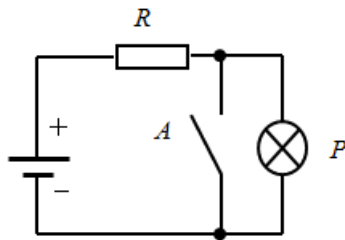



图 1-6 逻辑非运算电路

逻辑非的运算符号是在逻辑变量前加 “ \neg ” 或在逻辑变量的上方加 “ $\bar{}$ ”。


逻辑非的函数表达式为 $P = \bar{A}$ 。

 **注意：**逻辑与、逻辑或都允许有 2 个或 2 个以上输入变量，而逻辑非只有 1 个输入变量。

逻辑非的运算规则：


$\neg 0 = 1$ 或 $\bar{0} = 1$

$\neg 1 = 0$ 或 $\bar{1} = 0$

 **提示：**读者如果想系统了解逻辑代数的运算法则，可以进一步学习离散数学、数字电路（数字逻辑）等课程。

4. 逻辑运算的用途

逻辑运算通常用来测试各种逻辑关系的真假值。当我们学会编程后，经常就会用到条件语句。而逻辑运算就常用作条件语句中的条件。最常见到的逻辑运算就是循环的处理，逻辑运算的结果用作判断是否执行循环的条件，即用来判断是否该离开循环或继续执行循环内的指令。

 **提示：**在第 3 章学习程序结构时将接触到循环结构。循环结构主要用于重复执行相同的语句序列（被称为循环体），如果循环条件为真，则执行循环体；直到循环条件为假时才可终止执行循环体。循环条件可以是某个变量，可以是逻辑函数表达式，也可以是关系运算表达式。

下面的实例就是利用逻辑与、逻辑非和逻辑或的组合逻辑关系，来作为判断某年是否为闰年的条件。

【微实例 1.1】逻辑运算示例：闰年判断。

对于阳历 1900 年~2100 年之间的任何一年，写出判断其是否为闰年（leap year）的逻辑函数表达式。并以此判断 2014 年、2016 年、2100 年是否为闰年。

分析：

一回归年的时间 365 天 5 时 48 分 46 秒。阳历把一年定为 365 天，所余的时间约每 4 年积累成一天，加在二月里。这样的方法，在历法上叫做闰。

阳历四年一闰，在二月末加一天，这一天叫做闰日（leap day）。阳历有闰日的一年叫做闰年，这年有 366 天。所以阳历平常年份每年 365 天，二月为 28 天；闰年为 366 天，二月为 29 天。因此，每 400

年中有 97 个闰年。2000 年~2099 年间有 25 个闰年。

解：根据以上分析，满足以下两个条件之一的阳历年为闰年：

- ① 能被 4 整除、且不能被 100 整除的年为闰年（如 1904 年，2004 年）；
- ② 能被 400 整除的年为闰年（如 2000 年），不能被 400 整除的年则为平年（如 1900 年）。

整除可以用求模（即求余）运算符（%）来表示，若 x 与 y 的求模运算结果为 0，说明 x 能被 y 整除。假定要判断的年用变量 year 表示。

如果使用 Python 语言编程，则对于第一个条件，可以用下面的逻辑函数表达式来描述：

$$(year \% 4 == 0) \text{ and } (\text{not}(year \% 100 == 0)) \quad (1-2)$$

式中“==”是等值运算符，若等式成立，则运算结果为 1，否则为 0。

对于第二个条件，可以用下面的逻辑函数表达式来描述：

$$year \% 400 == 0 \quad (1-3)$$

整合后的逻辑函数表达式为：

$$(year \% 4 == 0) \text{ and } (\text{not}(year \% 100 == 0)) \text{ or } (year \% 400 == 0) \quad (1-4)$$

若该表达式的值为 1，则该年为闰年；若该表达式的值为 0，则该年不是闰年。

要判断 2014 年是否为闰年，只需将 2014 代入（1-4）式中，计算该式的值是否为 1：

$$(2014 \% 4 == 0) \text{ and } (\text{not}(2014 \% 100 == 0)) \text{ or } (2014 \% 400 == 0)$$

$$= 0 \text{ and } (\text{not}(0)) \text{ or } 0$$

$$= 0 \text{ and } 1 \text{ or } 0$$

$$= 0$$

该式的值为 0，说明 2014 年不是闰年。

请读者自行根据（1-4）式计算 2016 年、2100 年是否为闰年。

1.2.4 数据的 0 和 1 表示与物理实现

1.2.4.1 数据的 0 和 1 表示

在计算机内部，无论何种信息，都是用一串 0 和 1 字符串来表示的。

计算机中的数据又分为数值型数据和非数值型数据。**数值型数据**是表示数量、可以进行数值运算的数据类型。比如年龄 18 岁，身高 1.75m，气温 29℃，高考成绩总分 750 分等。这些数据在这里是用十进制来描述的。十进制就是采用 10 个基本符号（0，1，2，…，9）表示数值，其进位原则是“逢十进一”。

但在计算机中，其实是采用一串二进制数来表示数据的。比如整数 18 用 8 位二进制数表示就是 0001 0010。二进制就是采用 2 个基本符号（0 和 1）表示数值，其进位原则是“逢二进一”。

非数值型数据是表示字符（英文字母、汉字、标点符号等）、声音、图形、图像和视频等信息，不能进行数值运算的数据类型。例如汉字，尽管在计算机内部使用一串二进制数来表示，但它是不可进行数值运算的，而且运算也没有意义。如汉字“中”在计算机中存储时表示为 1101 0110 1101 0000，在程序或文档中为方便书写，一般写成十六进制 D6D0H。

1.2.4.2 0 和 1 的物理实现

数据在计算机中如何存储呢？换句话说，采用什么器件能够方便地存储 0 和 1 这两种状态？

可以用继电器开关（用闭合和断开表示 1 和 0）、灯泡（用亮和灭表示 1 和 0）等来表示二进制的两种状态。在计算机外部，通常使用磁盘、光盘或优盘（U 盘，闪盘）存储数据。磁盘是根据电磁学原理，使得涂敷在磁盘片上的磁性材料被磁化时具有两种极性来记录二进制信息的。光盘则利用激光原理，通过激光照射染料层，呈现结晶和非结晶两种状态，来记录二进制信息。优盘是一种移动存储设备，它利用 EEPROM（Electrically-Erasable Programmable Read-Only Memory，电擦除可编程只读存储器）技术来

记录数据，使用 USB 接口与计算机连接。

而在计算机内部，是采用电子技术来实现 0 和 1 的，即采用半导体器件来存储二进制的这两种状态。

导电能力介于导体和绝缘体之间的物体称为半导体。在纯净的半导体中掺入微量杂质，可以使导电能力剧增（几十万~几百万倍），从而制做成半导体器件。常见的半导体器件有晶体二极管、三极管和 MOS（Metal Oxide Semiconductor Field Effect Transistor，金属氧化物半导体场效应管）管等。

半导体器件具有典型的开关特性：它们有导通和截止两种状态，导通状态下允许电信号通过，截止状态下禁止电信号通过。导通和截止这两种状态正好可以用来表示 1 和 0。下面简要介绍二极管和三极管的开关特性。

1. 二极管的稳态开关特性

二极管稳态开关特性是指二极管稳定在正向导通和反向截止两种状态下的特性。与此相对应，还有二极管的瞬态开关特性：二极管由导通到截止、或者由截止到导通瞬变状态下表现出来的特性。对于后者这里不做介绍。

半导体二极管又称晶体二极管，或简称二极管。二极管是将 PN 结封装后引出两个金属电极制成的。因为有两个电极（阳极和阴极），所以称为二极管。二极管一般记作 D，其符号如图 1-3(a)所示。

二极管具有单向导电性，它相当于一个受外加电压极性控制的开关，其开关特性是正向导通，反向截止。也就是说，当在其两端加上正向电压时，其上的电阻很小，有很大的电流流过，二极管导通，相当于开关闭合；加反向电压时，其上的电阻为无穷大，几乎没有电流流过，二极管截止，相当于开关断开。理想二极管的开关特性电路示意图如图 1-7(b)和(c)所示。

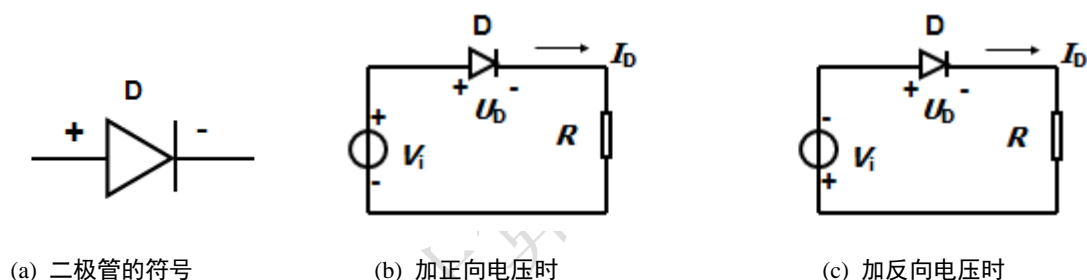


图 1-7 二极管的符号和理想二极管的开关特性


2. 三极管的稳态开关特性

三极管的稳态开关特性是指三极管稳定在截止和饱和导通两种状态下的特性。同样，晶体三极管具有瞬态开关特性，即指晶体三极管由导通过渡到截止、或者由截止过渡到导通表现出的特性。对于后者这里不做介绍。

半导体三极管又称晶体（三极）管，或简称三极管。一般记作 T。在模拟电路中，三极管主要作为线性放大器件和非线性器件；在数字电路中，三极管主要作为开关器件（即具有开关特性的器件）。

三极管具有放大、饱和（导通）、截止三种状态。当三极管作为放大器件时，主要工作在放大区；作为开关器件时，主要工作在截止区和饱和区。

在数字电路中，三极管相当于一个受输入电压 V_i 控制的开关。只要在输入端加上两种不同幅值（高电平如 3.0V，低电平如 0.3V）的信号，就可以控制三极管的导通或截止。

 **提示：**关于半导体器件这里不做详细介绍，感兴趣的读者可以通过学习《数字电路》或《数字逻辑》等课程来深入了解半导体器件的内部结构、工作原理和开关特性等。

1.2.4.3 计算机实现计算的物理基础

1. 门电路的概念

从前面的介绍可知，在计算机中，一切信息都是由 0 和 1 表示的。0 和 1 在物理上是利用半导体器件（二极管、三极管）来存储的。

如果把若干个半导体器件通过线路连接，就可以构成门电路。**门电路**是实现某种逻辑关系的电路，它是数字电路的基本逻辑单元电路。之所以叫做门电路，是因为它有开、关操作，正好能表示逻辑电路的两种状态。门电路是构造计算机或数字电路的基本元器件，计算机中的主要部件都是由各种门电路组合而成的。

门电路按照其实现的逻辑功能不同，分为基本逻辑门（实现基本逻辑关系的门电路）和复合逻辑门（由基本逻辑门组合而成、实现复杂逻辑关系的电路）。基本逻辑门有与门、或门以及非门，复合逻辑门有与非门、或非门、与或非门以及异或门等。

门电路按照其构成，主要分为两类：一类是分立元件门，即由电阻、二极管或三极管等分立元器件构成的门电路；一类是集成门，即把构成门电路的基本元器件制作在一小片半导体芯片上而形成的门电路。集成门相比分立元件门，具有体积小、耗电省、重量轻以及可靠性高等优点，因此，集成门出现后，即得到了广泛的使用，迅速取代了分立元件门。

通常一片芯片上集成有多个相同的逻辑门，如过去很流行的 74 系列芯片产品 74LS00 是 2 输入四与非门，即在一块芯片上集成了 4 个 2 输入的与非门。

2. 常用的门电路

各种常见的门电路有与门、或门、非门、与非门、或非门等。

（1）与门

实现逻辑与运算的电路称为与门。早期的与门是由若干个二极管组成的，二极管与门电路如图 1-8 所示。

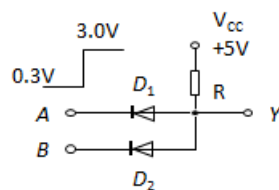


图 1-8 二极管与门电路

图中 A、B 为输入端，Y 为输出端。在输入端施加不同电位的电压，如低电平 0.3V，或者高电平 3.0V，则电路经过逻辑运算，在输出端会出现相应的变化，或者为高电平 3.7V，或者为低电平 1V。其工作原理分析如下：

当 A、B 均为低电平 0.3V 时， D_1 、 D_2 均导通；由于二极管导通后的钳位电压为 0.7V，则输出 $Y=0.3+0.7=1.0V$ 。

当 A 为 0.3V、B 为 3.0V 时， D_1 优先导通，输出 $Y=0.3+0.7=1.0V$ ；由于 B 为 3.0V，大于 Y (1.0V)，则 D_2 被施以反向电压，故为截止状态。

当 A、B 均为高电平 3.0V 时， D_1 、 D_2 均导通，则输出 $Y=3+0.7=3.7V$ 。

若用 0 代表低电平（输入 0.3V，输出 1.0V），用 1 代表高电平（输入 3.0V，输出 3.7V），则可以得到真值表，如 1.2.3.1 节中表 1-2 所示。

由表 1-2 可以看出，只要输入有一个是低电平，则输出是低电平；只有当输入都是高电平时，输出才是高电平。输出与输入之间符合“与”逻辑关系，故这样的电路称为与门。

根据真值表，可以推导得出与门的逻辑函数表达式为 $Y = A \cdot B$ 。

（2）或门

实现逻辑或运算的电路称为或门。二极管或门电路及其逻辑符号如图 1-9 所示。

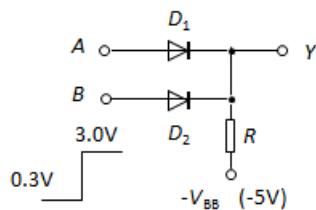


图 1-9 二极管或门电路

由于篇幅所限，对二极管或门的工作原理分析略去。或门的真值表如表 1-5 所示。

表 1-4 或门的真值表

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

根据真值表，可以推导出或门的逻辑函数表达式为： $Y = A + B$ 。

(3) 非门

实现逻辑非运算的电路称为非门（反相器）。最早非门是由三极管加上若干电阻实现的。由于篇幅所限，三极管非门电路及其工作原理分析略去。非门的真值表如表 1-5 所示。

表 1-5 非门的真值表

A	Y
0	1
1	0

非门的逻辑函数表达式为： $Y = \overline{A}$ 。

(4) 与非门

利用基本逻辑门可以构成更复杂的门电路，即复合逻辑门，如与非门、或非门、异或门等。与非门是由二极管与门和三极管非门复合而成的。与非门的逻辑函数表达式为： $Y = \overline{AB}$ 。

(5) 或非门

或非门由二极管或门和三极管非门复合而成。或非门的逻辑函数表达式为： $Y = \overline{A + B}$ 。

3. 由门电路构造加法器

由上面的分析可以看到，由基本逻辑门如与门、或门、非门进行各种组合，可以构成复合逻辑门，如与非门、或非门、异或门等。再由各种门电路，可以组合成更复杂的逻辑电路；将具有一定功能的电路集成在一个硅片上，可以形成规模更大、功能更强的集成电路。

下面以加法器为例说明由各种门电路构造出更复杂逻辑电路的一般方法。

能对两个 1 位二进制数进行相加并考虑低位来的进位、求得和并向高位进位的逻辑电路称为全加器。怎样设计出 1 位全加器呢？一般来说，逻辑电路的手工设计方法包括以下几个步骤：

(1) 进行逻辑抽象。即确定输入变量和输出变量，分析因果关系，列出真值表。

(2) 写出逻辑函数表达式。根据真值表写出逻辑函数的标准表达式。

(3) 进行逻辑化简。采用逻辑代数的公式化简法或卡诺图化简法将逻辑函数的标准表达式化简为最简逻辑函数表达式。

(4) 绘逻辑图。即根据最简逻辑函数表达式画出原理图。

例如，可以按照上面步骤采用各种逻辑门搭建出一个 1 位全加器。再由 4 个 1 位全加器扩展成一个 4 位加法器。

因篇幅有限，这里不给出详细设计步骤，感兴趣的读者可以进一步学习数字电路或数字逻辑课程。

根据输出的逻辑函数表达式，即可画出相应的逻辑电路。1 位全加器电路如图 1-10(a)所示。可以看出，全加器的电路结构包括 2 个异或门及 3 个与非门。相应的逻辑符号如图 1-10(b)所示。

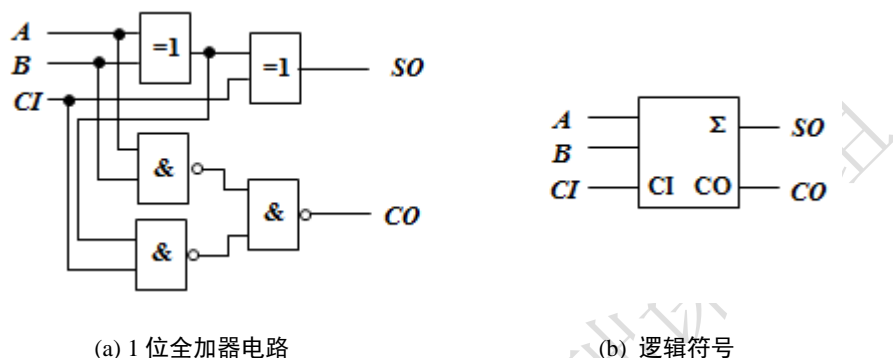


图 1-10 1 位全加器电路及其逻辑符号

进一步地，由 4 个 1 位全加器可以扩展成 4 位串行进位加法器，如图 1-11 所示。之所以称作串行进位，是因为进位是从最低位向高位逐位串行完成的。它依次将低位全加器的进位输出端 CO 接到高位全加器的进位输入端 CI 。加法从低位开始，高位全加必须等低位进位来到后才能进行，因此完成加法的时间较长。

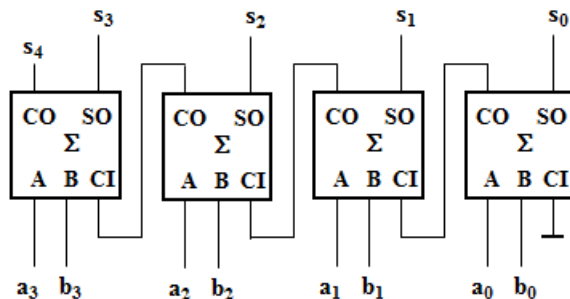


图 1-11 4 位串行进位加法器

实际上，通过特定的技术，可以将二进制数之间的减、乘、除算术运算，都转化为若干步的加法运算来进行。故实现了加法器，就能实现所有的二进制算术运算。因此，加法器是构成算术运算电路的基本单元电路。

由本节分析可知，在现实世界中，一切信息均可用 0 和 1 的组合来表示；用 0 和 1 可以方便地进行算术运算和逻辑运算；0 和 1 可以用半导体器件来存储；半导体二极管、半导体三极管、场效应晶体管等器件是数字电路（逻辑电路）中的主要开关器件，由它们构成了基本逻辑门电路。门电路是构造计算机或数字电路的基本元器件，计算机中的各大部件都是由各种门电路组合而成的。再把具有一定功能的部件集成在一个硅片上，就能形成更大规模、更复杂、功能更强的电路。比如，处理器就是把运算器和控制器集成在一起制成的。再把处理器、存储器、各种输入设备和输出设备的接口等放在一块电路板上，它们之间通过导线连接，就像搭积木一样，最终就构建出了计算机。计算机硬件系统的具体组成将在 1.5.1 节中详细介绍。信息被计算机存储和处理的原理如图 1-12 所示。

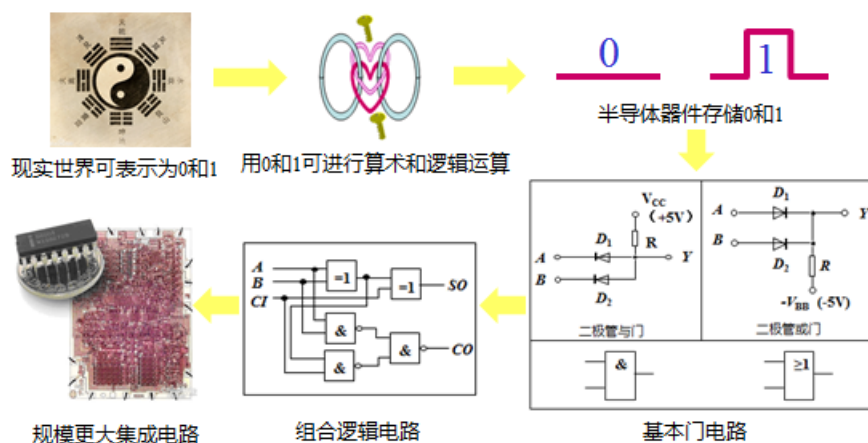


图 1-12 信息被计算机存储和处理的原理

1.2.4.4 为什么计算机中采用二进制而不是十进制

在日常生活和工作中，人们都习惯使用十进制来表示数的大小。然而，计算机内部却采用二进制来表示一切数据，这是为什么呢？概括起来，主要有以下几个方面的原因：

(1) 二进制运算规则简单

众所周知，十进制包括 0~9 共 10 个数码，遵循进位原则“逢 10 进 1，借 1 当 10”。而二进制只包括“0”和“1”两个数码，进位原则为“逢 2 进 1，借 1 当 2”。

二进制的加法运算只有四条规则：

$$\begin{aligned} 0+0 &= 0 \\ 0+1 &= 1 \\ 1+0 &= 1 \\ 1+1 &= (1)0 \end{aligned}$$

乘法运算也只有四条规则：

$$\begin{aligned} 0*0 &= 0 \\ 0*1 &= 0 \\ 1*0 &= 0 \\ 1*1 &= 1 \end{aligned}$$

而十进制数的运算规则则复杂得多。比如乘法运算，除 0 以外的每一个数码与 1~9 数字的乘法运算规则一共 $9*9=81$ 条，要全部记住也不是一件容易的事。显而易见，比起十进制来，二进制运算规则更简单。

(2) 采用半导体器件表示二进制的两种状态具有诸多优势

在计算机中，使用半导体器件如晶体二极管、三极管、MOS 管来表示二进制的两种状态，具有诸多优势：

① **半导体器件具有开关特性。**即在不同的输入条件下，有两个完全不一样的状态（导通和截止），正好对应二进制的 0 和 1。

② **抗干扰能力强。**只要电信号在一定范围之内，就能够可靠地区分高电平和低电平两种状态。

③ **从一种状态转换为另一种状态很方便。**比如硅二极管，只要加在其两端的电压大于 0.7V，它就导通；如果小于 0.7V，它就截止。对于三极管，只要在输入端加上两种不同幅值（高电平和低电平）的信号，就可以控制它的导通或截止。

④ **状态转换速度非常快。**即开关速度非常快，这是开关电路的重要性能指标，它决定了计算机的运算速度。

⑤ **体积小。**随着微电子技术和集成电路制造技术的进步，晶体管的尺寸越来越小，在一片硅片上，可以集成几万、几十万甚至上亿个晶体管，使得集成电路的集成度越来越高，相应地，整个计算机的体积减小，可靠性更高。

⑥ **工作时消耗的电能低，即功耗（能耗）低。**则整个计算机的功耗很小。功耗是计算机的一项重

要指标，因为能耗会导致计算机中的芯片发热，极大地影响芯片的集成度，从而限制计算机的运行速度。

(3) 二进制算术运算与逻辑运算能够统一起来

可以用逻辑运算实现算术运算（可以采用移位、逻辑与、逻辑或等运算进行加、减、乘、除运算）。在计算机中，正是将具有逻辑与、逻辑或、逻辑非等功能的门电路组合起来，实现加法运算的。而对于减法、乘法、除法等运算，计算机都可以利用特殊的技术，将其转化为加法来实现，这样就使得运算器的设计变得简单了。

(4) 二进制数据便于存储

采用二进制表示数据，物理上容易实现数据的存储。二进制数据除了可以利用半导体器件来存储外，还可以利用磁盘和光盘等来存储。

1.3 计算机的理论模型与物理实现

在本节中，首先了解计算机的理论模型——图灵机模型，包括其组成、工作原理及其蕴含的计算思想。然后学习计算机的物理实现——冯·诺依曼计算机，包括其组成、特点等，并理解冯·诺伊曼机的基本思想。

请带着以下问题学习本节内容：

- (1) 图灵机模型由哪几部分组成？图灵机模型是如何抽象的？
- (2) 图灵机的思想是什么？
- (3) 冯·诺依曼机的基本思想是什么？冯·诺依曼机有哪些特点？
- (4) 冯·诺依曼机包括哪五大部件？其作用分别是什么？为什么计算机能够像人一样进行加减乘除各种算术运算，甚至能够进行各种逻辑运算？为什么计算机能够自动工作？
- (5) 以存储器为中心的结构与以运算器为中心的结构有何不同？

1.3.1 图灵机模型

计算机之所以能够计算，或者说具备智能，是因为其思维的过程可以与人脑类比：

- ✓ 语言单元是由人脑的神经元单元来感知的；在计算机中，则表示为数字及其编码。
- ✓ 语言的关系在人脑中表现为神经元的连接；在计算机中，则表现为数据结构、数据关系。
- ✓ 思维在人脑中体现为神经网络的自动连接；而在计算机中，则体现为数据的模型和算法。
- ✓ 人的感知和动作是由人的器官和肢体来感知和响应的；而在计算机中，则体现为输入和输出。

如果有一台机器，可以存储数据，可以进行运算，可以输入数据和输出运算结果，那么，它就可以具备计算的能力。下面就来看看这个抽象模型——图灵机模型的建立。

1. 图灵与图灵机缘起

阿兰·麦席森·图灵（1912~1954），英国数学家、逻辑学家和密码学家，计算机逻辑的奠基者，曾提出“图灵机（Turing Machine）”和“图灵测试”等重要概念，被誉为“计算机科学之父”和“人工智能之父”。

图灵机的由来是因德国数学家希尔伯特在他所提出的“23个数学难题”中，提出了逻辑的完备性问题，即是否所有数学问题原则上都可解。1936年，图灵向伦敦权威的数学杂志投了一篇题为“论数字计算在决断难题中的应用（On Computable Numbers With an Application to the Entscheidungs Problem）”的论文。在这篇开创性的论文中，图灵给“可计算性”下了一个严格的数学定义，并提出著名的图灵机的设想：凡是能够用“图灵机”的自动机理论模型表达的问题，就是可解的问题。图灵机不是一种具体的机器，而是一种思想模型，可制造一种十分简单但运算能力极强的计算装置，用来计算所有能想象到的可计算函数。

图灵机与后来的冯·诺伊曼机齐名，被永远载入计算机的发展史中。

1950 年 10 月，图灵又发表了另一篇题为“机器能思考吗”的论文，该文成为了人工智能的开山之作。也正是这篇文章，为图灵赢得了“人工智能之父”的桂冠。

为纪念图灵在计算机领域的卓越贡献，计算机界于 1966 年以其名字命名，设立了计算机领域的最高荣誉奖：ACM 图灵奖，人们称之为计算机界的诺贝尔奖。

2. 图灵机模型的组成

人们用纸笔进行数学计算的过程基本包括 5 个步骤：

- (1) 根据计算需求在纸上写下相应的公式或符号；
- (2) 根据眼睛看到的符号，在脑中思考相应的计算方法；
- (3) 在纸上写上或擦除某个符号；
- (4) 把视线从纸的一个位置移动到另一个位置；
- (5) 转到第 (2) 步，重复以上步骤，直到认为计算停止为止。

图灵通过仔细的观察和研究发现，在每个阶段，人要决定下一步的动作，都依赖于：

- (a) 人当前所关注的纸上某个位置的符号
- (b) 人当前思维的状态

为了模拟人的这种数学计算过程，图灵构造出一台假想的机器——图灵机。图灵机模型如图 1- 13 所示。

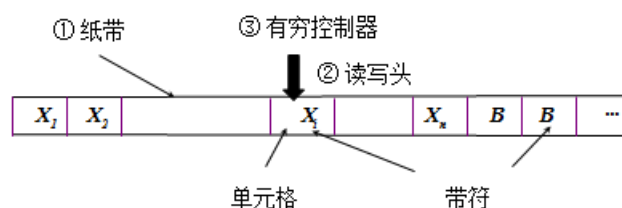


图 1- 13 图灵机模型

图灵机模型由 3 个部件组成：

① 无穷纸带（符号集合）

纸带两端可无限延长。纸带被分为一系列均匀的方格，每个方格中可填写一个来自有限字母表的符号。字母表中有一个特殊的符号表示空白。

② 读写头（读、改写、左移、右移）

读写头可沿带子方向左右移动（一次只能移动一格）或停留在原地，并可以在当前方格上进行读写。

③ 有穷控制器（有限状态机）

控制器是一个有限状态自动机，拥有预定的有限个互不相同的状态，并能根据输入改变自身的状态，但任何时候它只能处于这些状态中的一种；由控制器来控制读写头左右移动并读写。

尽管纸带可以无限长，但写进方格的符号不能无限多，通常是一个有穷的字母表，可设为 $\{C_0, C_1, C_2, \dots, C_n\}$ 。控制器的状态用集合 $\{S_0, S_1, S_2, \dots, S_m\}$ 来表示，控制器的状态也就是图灵机的状态。

该模型具有以下两个性质：

- (a) 模型的每个过程都是有穷可描述的；
- (b) 过程必须是由离散的、可以机械执行的步骤组成。

图灵机的计算就是由控制器控制执行的一系列动作。

3. 图灵机模型是如何抽象的？

图灵机模型将所有具体的输入、输出、转换细节抛弃，定义了一个五元组 $(K, \Sigma, \delta, s, H)$ ，其中：

K 是有穷（有限）个状态的集合；

Σ 是字母表，即符号的集合；

δ 是转移函数，即控制器的规则集合（所有操作命令的集合）；

$s \in K$ ，是初始状态；

$H \in K$ ，是停机状态的集合，当控制器内部状态为停机状态时图灵机结束计算。

4、图灵机的工作原理

图灵机的计算，就是由控制器控制执行的一系列动作——读符号、改写符号、读写头左移、读写头右移等。图灵机从纸带上的某个起始点出发，动作完全由初始状态和指令组决定；其计算结果是从图灵机停止时纸带上的信息得到的。

【案例 1.1】图灵机模型案例：设计计算“ $x+1$ ”的图灵机。

利用二进制设计一个专门计算“ $x+1$ ”的图灵机，要求计算完成时，读写头回归原位。 x 是一个二进制变量，由一系列的 0、1 串组成，“*”为 x 的分隔符和界定符（当纸带上有多个数时，用*将它们隔开）。

解：

（1）对问题进行抽象，确定图灵机五元组的内容

要设计一个计算“ $x+1$ ”的图灵机，首先必须进行抽象，即确定图灵机五元组的内容：

① 状态集合 K : {start, add, carry, noncarry, overflow, return, halt}

状态集合 K 共有 7 个状态，start: 开始，add: 加法，carry: 进位，noncarry: 不进位，overflow: 溢出，return: 返回，halt: 停机。

② 字母表 Σ : {0, 1, *}

③ 规则集合 δ

④ 初始状态 s : start

⑤ 停机状态集合 H : {halt}

规则的一般表述是：如果 A 那么 B，其形式化表示为： $A \rightarrow B$ 。

图灵机控制器的规则其形式化表示为：

（控制器当前状态，读写头当前位置的符号） \rightarrow （读写头要写入的新符号，读写头移动动作指示，控制器新状态）

上面形式化表示的含义是：根据控制器当前状态和读写头当前位置的符号确定读写头将写入的新符号（如果新符号与原来符号一致，表示不改写）、读写头移动动作指示以及控制器新状态。

（2）设计规则

分析计算“ $x+1$ ”时所有可能发生的情况，可以设计出规则集合 δ ，如表 1-6 所示。

如果把一行中的 5 列值放在一起，用圆括号括起来，就可以把它看成是一条指令。对于计算“ $5+1$ ”，共使用了 8 条指令。例如在初始状态下，指令为（start, *, *, left, add）。

有的指令需要重复执行几次。如当计算得到 110 后，由于读写头要回归原位，有两次当前状态都是 return，当前符号都是 1，所以（return, 1, 1, right, return）要执行 2 次。

实际上，规则集合是图灵机所有操作命令的集合，我们称之为指令集。

下面以“ $5+1$ ”的计算过程为例，来分析图灵机的工作过程。

图灵机初始状态如图 1-14 所示。 $x=5$ ，用二进制表示为 101，存储在两个*号之间。计算过程从 101 右边的分隔符“*”开始，根据当前状态和读入的数据决定是否改写数据，并向左移动读写头。

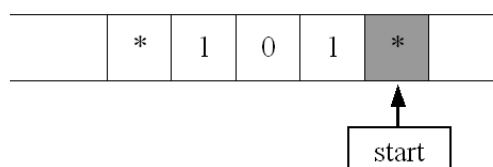


图 1-14 计算“ $x+1$ ”时图灵机初始状态

表 1-6 “x+1”的图灵机规则集合

输 入		响 应		
当前状态	当前符号	新符号	读写头移动	新状态
start	*	*	left	add
add	0	1	left	noncarry
add	1	0	left	carry
add	*	*	right	halt
carry	0	1	left	noncarry
carry	1	0	left	carry
carry	*	1	left	overflow
noncarry	0	0	left	noncarry
noncarry	1	1	left	noncarry
noncarry	*	*	right	return
overflow	0 或 1	*	right	return
return	0	0	right	return
return	1	1	right	return
return	*	*	stay	halt

第 1 步：根据规则集中规则 1，当前状态为初始状态 start，当前符号为“*”。则图灵机响应输入，新符号也是“*”，不变；读写头左移一格；新状态变为“add”（加法）。

第 1 步完成后状态如图 1-15 所示。

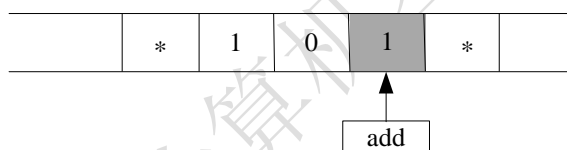


图 1-15 计算“x+1”第 1 步完成后状态

第 2 步：第 1 步完成后，读写头已左移至最低位“1”。按照规则 3，新符号从 1 变成 0，然后读写头左移一格，新状态为“carry”（进位）。第 2 步完成后状态如图 1-16 所示。

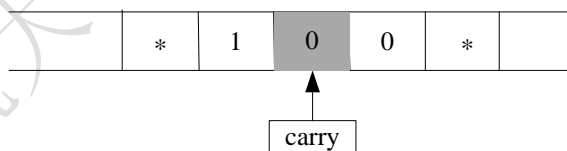


图 1-16 计算“x+1”第 2 步完成后状态

第 3 步：第 2 步完成后，读写头已左移至中间一位“0”。按照规则 5，新符号从 0 变成 1，然后读写头左移一格，新状态为“noncarry”（不进位）。

第 3 步完成后状态如图 1-17 所示。

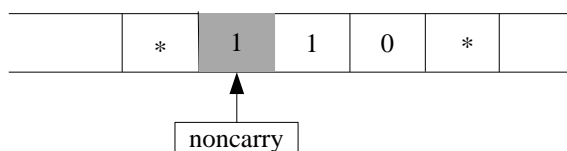


图 1-17 计算“x+1”第 3 步完成后状态

第 4 步：第 4 步完成后，得到 6（110），并把读写头移到 110 的左边（*的位置）。

第 5 步：读写头右移一位，至输入数据的最高位“1”；新状态变为 return（返回）。

第 6~8 步：读写头逐步右移，直到移至最右边的“*”位置。

当第 8 步完成时，读写头移回至最右边的 “*” 位置，状态为 return。

第 9 步：停机（停止计算）。此时，根据规则集合，新符号仍为 “*”，读写头停留在原地不动（stay），新状态变为 “halt”（停机）。如图 1- 18 所示。

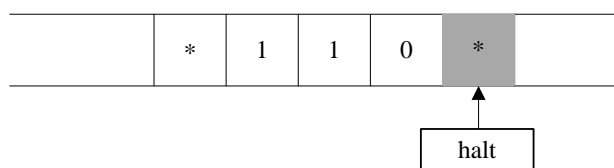


图 1- 18 计算 “ $x+1$ ” 第 9 步完成后状态（停机状态）

可见，经过一系列的操作（有限个步骤），如改写当前符号、读写头左移或右移，最后，读写头仍然回到原位。此时，图灵机停止操作，输出结果为二进制的 110，即完成了 $5+1=6$ 的计算。

在本例中，一共使用了 9 个步骤实现 “ $5+1$ ” 的运算。第 1 步~第 4 步完成 $5+1$ 的计算，得到 110，并把读写头移到 110 的左边（*的位置）；第 5 步~第 9 步将读写头移回到其原来的位置。

上例中计算 “ $x+1$ ” 的图灵机其功能是固定的，如果要实现更加复杂的计算呢？比如 $x+y$ 、 $x*y$ 、 y/x 等，只需制定不同的规则集合，就可以实现不同的运算。

图灵机的本质是进行字符串的处理。图灵机输入是一个字符串，输出也是一个字符串，如果将图灵机的有限个内部状态与读写头的有限个动作也用字符串表示，那么每条转换规则也可以用一個字符串表示：（当前状态，当前符号，新符号，动作，新状态）。则图灵机可以由一个较长字符串完全表示，这就是通用图灵机。

满足这样一个有穷规则、有限状态的可以对字符串进行处理的机器，就是计算机；而可以描述成这样的科学问题，就是可计算的；这种机器的具体实现，则是下一小节将学习的冯·诺伊曼机。

5、图灵机的思想

图灵机不是具体的机器，而是一种思想模型，其基本思想是用机器来模拟人们用纸笔进行数学运算的过程。

在图灵看来，这台机器只用保留一些最简单的指令，一个复杂的工作只需把它分解为这几个最简单的操作就可以实现了。

所谓**计算**，就是计算者（人或机器）对一条两端可无限延长的纸带上的一串 0 或 1（输入符号串），执行指令一步一步地改变纸带上的 0 或 1，经过有限步骤，最后得到一个满足预先规定的输出符号串的变换过程。可以用如图 1- 19 所示的图来形象地描述图灵机的思想。

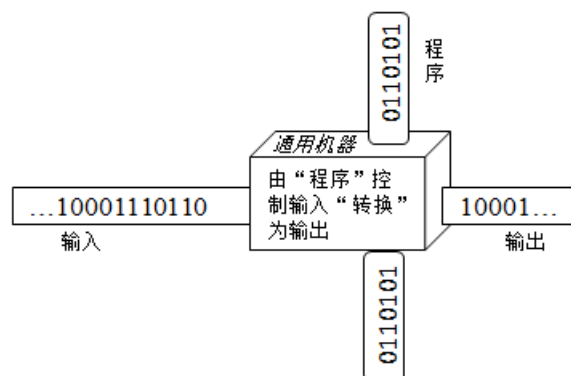


图 1- 19 图灵机的思想示意图

图灵机是从过程这一角度来表现计算的本质的。其结构简单，操作运算规则较少，易于被人们所理解。

图灵机的思想是数据、指令、程序及程序/指令自动执行的基本思想。

输入被制成一串 0 和 1 的纸带，送入机器中——这就是数据。如 00010000100011……。

机器可对输入纸带执行的基本动作包括：“翻转 0 为 1”，或“翻转 1 为 0”，“左移一位”，“右移一位”，“停止”。

对基本动作的控制是由指令完成的，机器是按照指令的控制选择执行哪一个动作。指令也可以用 0 和 1 来表示：如 001 表示“翻转 0 为 1”(当输入为 1 时不变)，010 表示“翻转 1 为 0”(当输入 0 时不变)，011 表示“左移一位”，100 表示“右移一位”，000 表示“停止”。

输入如何变为输出的控制可以用指令编写一个程序来完成，如：011110110111011100……。

机器能够读取程序，按程序中的指令顺序读取指令，读一条指令执行一条指令，由此实现自动计算。

1.3.2 冯·诺伊曼计算机

图灵机模型直观形象地说明了通用计算机的工作机理，建立了指令、程序及通用机器执行程序的理论模型，奠定了计算理论的基础，这是图灵对人类的最大贡献。

但是，图灵机毕竟不是实际的计算机，后人在图灵机理论模型基础上，通过研究一系列的关键技术、实现方法和制造技术，最终才发明了现代计算机。

本节将从冯·诺伊曼计算机的基本思想出发，介绍现代计算机模型的产生、组成和特点。

1.3.2.1 冯·诺伊曼思想的提出

1. 冯·诺伊曼其人

回顾 20 世纪科学技术的辉煌发展，不能不提及 20 世纪最杰出的数学家之一——约翰·冯·诺依曼 (John Von Neuman, 1903—1957)。正是冯·诺依曼，在 1945 年提出了“程序内存式”计算机的这一卓越的设计思想，从而为电子计算机的逻辑结构设计奠定了基础。由于冯·诺依曼在计算机逻辑结构设计上的伟大贡献，他被誉为“现代电子计算机之父”。

冯·诺依曼是美籍匈牙利人。他是普林斯顿大学、宾夕法尼亚大学、哈佛大学、伊斯坦堡大学、马里兰大学、哥伦比亚大学和慕尼黑高等技术学院等校的荣誉博士，也是美国国家科学院、秘鲁国立自然科学院和意大利国立林且学院等院的院士。1951 年~1953 年任美国数学会主席；1954 年任美国原子能委员会委员。

冯·诺伊曼不仅是个数学天才，在数学的诸多领域都有很深的造诣并取得了重要成果，而且在原子物理学、化学和经济学等领域也颇有建树。他对人类的最大贡献是对计算机科学、计算机技术、数值分析和经济学中的博弈论的开拓性工作。

2. 冯·诺伊曼思想的提出

1946 年 2 月 14 日，世界第二台电子计算机 ENIAC (Electronic Numerical Integrator And Calculator, 电子数字积分计算机)，在美国宾夕法尼亚大学诞生。研制小组由四位科学家和工程师埃克特 (John Presper Eckert)、约翰·莫克利 (John Mauchley)、赫尔曼·戈尔斯坦 (Herman Goldstein)、亚瑟·博克斯 (Arthur Burks) 组成。ENIAC 能在一秒钟内进行 5000 次加法运算或 400 次乘法运算，运算速度是当时最快的继电器计算机的 1000 倍。而且它还具有按事先编好的程序自动执行算术运算、逻辑运算和存储数据的功能。ENIAC 宣告了一个新时代的开始，从此科学计算的大门被打开了。

不过，ENIAC 机本身存在两大缺陷：(1) 采用十进制进行运算，逻辑元件多，结构复杂，可靠性低；(2) 没有存储器，操纵运算的指令分散存储在许多电路部件内，计算题目之前必须预先编写指令，再按指令手工连接好控制线路，然后启动它才能自动运行。每次计算一个题目都必须重复以上工作，重新连线，有时这种准备工作要花费几小时甚至几天时间，因此在很大程度上抵消了 ENIAC 的计算速度。ENIAC 机研制小组意识到了这个问题，在 ENIAC 尚未竣工之前，就着手计划研制一个结构全新的电子计算机。

1944 年夏，冯·诺依曼应戈尔斯坦中尉之邀加入了 ENIAC 研制小组，担任顾问。

1945 年 6 月底，ENIAC 研制小组发表了一个全新的“存储程序通用电子计算机方案”——EDVAC (Electronic Discrete Variable Automatic Computer, 离散变量自动电子计算机)。冯·诺伊曼起草了长达 101 页的总结报告“关于 EDVAC 的报告草案”。报告全面而具体地介绍了制造电子计算机和程序设计的新思想，明确规定了电子计算机由运算器、逻辑控制装置、存储器、输入设备和输出设备等五大部分构成的基本结构形式，并描述了这五部分的职能和相互关系。报告中，冯·诺伊曼对 EDVAC 中的两大设计思想作了进一步的论证。

EDVAC 的设计思想之一是**二进制**。冯·诺伊曼根据电子元件双稳工作的特点，建议在电子计算机中采用二进制，并预言，二进制的采用将极大简化计算机的逻辑线路。

EDVAC 的设计思想之二是**存储程序** (Stored Program)。针对 ENIAC 没有存储器、指令需要手工连线接入的问题，冯·诺伊曼提出了存储程序的思想：把运算程序和数据都存储在机器的存储器中，程序设计员只需要在存储器中寻找运算指令，机器就会自行计算。这样，不仅可以使计算机的结构大大简化，而且为实现运算控制自动化和提高运算速度提供了良好的条件。

1946 年 7~8 月间，冯·诺依曼和戈尔斯坦、博克斯在 EDVAC 方案的基础上，为普林斯顿大学高等研究院研制 IAS (Institute for Advanced Study) 计算机时，又提出了一个更加完善的设计报告“电子计算机逻辑设计初探”。

EDVAC 于 1949 年 8 月交付给弹道研究实验室，1951 年才开始运行。它的运算速度与 ENIAC 相似，而使用的电子管却只有 5900 多个，比 ENIAC 少得多。EDVAC 的诞生，使计算机技术出现了一个新的飞跃。它奠定了现代电子计算机的基本结构，标志着电子计算机时代的真正开始。

至此，冯·诺伊曼建立了现代电子数字计算机的基本结构体系 and 设计原则。这些设计思想和方案很快就被成功运用到计算机的设计中，并被沿用至今。因此这种体系结构的计算机被称为“冯·诺伊曼计算机”。它的综合设计思想，便是著名的“冯·诺伊曼思想”。

1.3.2.2 冯·诺伊曼计算机的组成及其特点

1. 冯·诺伊曼计算机的组成

如图 1- 20 所示，冯·诺伊曼计算机由五大部分组成：输入数据和程序的输入设备，记忆程序和数据存储器，完成数据加工处理的运算器，控制程序执行的控制器，输出处理结果的输出设备。各部分相互连接，并各司其职，使得计算机能够自动、协调一致地工作。下面简要分析各部分的作用。

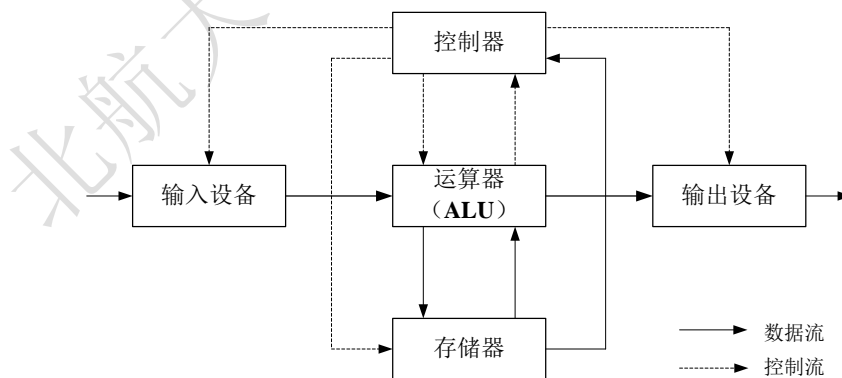


图 1- 20 冯·诺伊曼计算机的组成

(1) 运算器

在本小节的开始，曾提出一个问题：为什么计算机能够像人一样进行加减乘除各种算术运算，甚至能够进行各种逻辑运算呢？这是因为，它有一个称为运算器 (Arithmetic Unit) 的部件。

运算器是计算机中执行各种算术运算（加、减、乘、除）、逻辑运算（与、或、非、异或）及移位操作（左移、右移）的部件。运算器处理的数据来自存储器；处理后的结果数据通常送回存储器，或暂

时寄存在运算器的寄存器中。

运算器主要由算术逻辑单元（Arithmetic and Logic Unit, ALU）、寄存器和控制电路 3 个部分构成。

ALU：是具体完成算术与逻辑运算的运算单元，是运算器的核心，由加法器和若干门电路组成。主要完成对二进制信息的算术运算、逻辑运算和各种移位操作。

寄存器：包括通用寄存器和状态寄存器，**通用寄存器**用于接收、暂存和记录操作数，保存运算结果。**状态寄存器**用来记录算术、逻辑运算或测试操作的结果状态。程序设计中，这些状态通常用作条件转移指令的判断条件，所以又称为**条件码寄存器**。

控制电路：按照一定时间顺序发出不同的控制信号，使数据经过相应的门电路进入寄存器或加法器，完成规定操作。

运算器的操作和操作种类由控制器决定。

运算器能执行多少种操作和其操作速度，标志着运算器能力的强弱，甚至标志着计算机本身的能力。前面曾介绍过，为了简化 CPU 的设计，计算机的各种算术运算，其实都是以加法为基础的。运算器最基本的算术运算是加法。减法采用补码，转换为加法运算；乘法采用移位和加法；除法运算通常是将除法转换成若干次“加、减、移位”循环，然后通过硬件或软件来实现。

运算器的主要性能指标有：

字长：运算器所能并行处理的二进制数的位数，一般为字节的整数倍，8 位、16 位、32 位到 64 位不等。

运算速度：一般用平均速度，即在单位时间内平均能执行的指令条数表示。运算速度也是计算机的主要指标之一。

（2）控制器

计算机能够自动、高速、精确地完成信息处理和其他工作。它为什么能够自动工作呢？实际上就是因为它具有控制器这个关键部件。**控制器**是指挥计算机的各个部件按照指令的功能要求协调工作的部件。它是计算机的神经中枢和指挥中心，就好比人的大脑一样。

控制器由指令寄存器（Instruction Register, IR）、指令译码器（Instruction Decoder, ID）、程序计数器（Program Counter, PC）和操作控制器（Operation Controller, OC）4 个部件组成。控制器结构简图如图 1-21 所示。

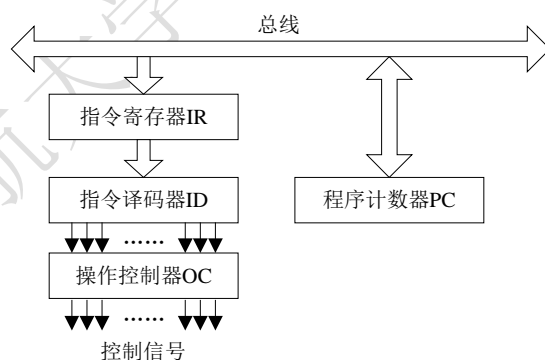


图 1-21 控制器结构简图

IR 是保存当前执行或即将执行的指令的一种寄存器。


ID 是对指令的操作码进行译码的译码器。

PC 是指明程序中下一次要执行的指令地址的一种计数器，又称**指令计数器**。

OC 根据指令操作码和时序信号，产生各种操作控制信号，以便正确地建立数据通路，从而完成取指令和执行指令的控制。

概括来说，控制器的功能就是读取指令、分析指令并执行指令。执行程序时，控制器先向存储器发送第 1 条指令地址，按地址从存储器中读取指令，并分析指令；然后执行指令，即向存储器发送操作数地址，从存储器中取出该数据；再向存储器发送下一条指令地址，重复进行读取指令、分析指令和执行

指令的操作，直至程序执行完毕。

 **提示：**在最初的冯·诺依曼计算机中，计算机由运算器、控制器、存储器、输入设备和输出设备五大部分组成。然而，随着集成电路的出现，为提高可靠性和减小体积，人们便把运算器和控制器制作到一个硅片上，称之为中央处理器（Central Processing Unit，CPU）。

（3）存储器

冯·诺伊曼计算机的核心思想是“存储程序”的思想：指令和数据以同等地位事先存于存储器中，可按地址寻访，机器从存储器中读取指令和数据，实现程序的连续自动执行。

什么是存储器呢？**存储器（Memory）**是计算机中具有记忆功能的部件，用来存放程序和数据，能在计算机运行过程中高速、自动地完成程序或数据的存取。

计算机中的全部信息，包括输入的原始数据、计算机程序、中间运行结果和最终运行结果等都保存在存储器中。存储器根据控制器指定的位置存入和取出信息。有了存储器，计算机才有记忆功能，才能保证正常工作。

存储器内部结构如图 1- 22 所示。存储器由若干个存储单元组成，而存储单元由若干个存储元（存储位）组成。

存储单元是可以被存储器一次并行读出或写入的独立访问单元。一般是 8 个存储元组成一个存储单元。**存储元**是存放一个二进制数位（0 或 1）的记忆单元，是存储器最小的存储单位。



图 1- 22 存储器内部结构图

每个存储单元的位置都有一个编号，称为**存储地址**。按地址访问存储单元的内容是存储器的基本特性，存储单元的地址按二进制编码，存储单元的内容也按二进制存储。

地址线的位数决定了存储容量。比如有 10 位地址线，则存储容量为 $2^{10}=1024$ ，即存储器共有 1024 个存储单元。

在 1.5.1 节中，将详细介绍存储器的不同分类以及现代计算机系统的层次化存储体系。

（4）输入设备

输入设备是将程序和指令等用户信息变换为计算机能识别和处理的二进制信息形式输入计算机中的设备。

（5）输出设备

输出设备是将计算机处理的结果（二进制信息）变换为用户所需要的信息形式输出的设备。

2. 冯·诺伊曼计算机的特点

根据冯·诺依曼提出的原理制造的计算机被称为**冯·诺依曼结构计算机**或**冯·诺依曼计算机**，也称为**存储程序计算机**。现代计算机虽然结构更加复杂，计算能力更加强大，但仍然是基于冯·诺伊曼思想设计的。归纳起来，冯·诺依曼计算机在体系结构上的主要特点是二进制、程序存储、顺序执行、五大部件组成、共享数据。具体而言，具有如下几个显著特点：

（1）**二进制：**指令和数据都用二进制代码表示；

(2) **存储程序方式**：指令和数据不加区别以同等地位事先混合存于存储器中，可按地址寻访，连续自动执行；

(3) **存储器的地址和位数**：存储器是按地址访问的线性编址的一维结构，每个存储单元的位数是固定的；

(4) **指令的形式**：指令由操作码和地址码组成，操作码指明指令所要完成操作的性质和功能，地址码指明操作数及其在存储器中的位置；

(5) **指令的执行**：指令按照执行的顺序依次存放在存储器中，由指令计数器指明要执行的指令所在存储单元的地址；

(6) **由五大部件构成**：运算器、控制器、存储器、输入设备和输出设备；

(7) **以运算器为中心**：输入设备、输出设备与存储器之间的数据传送都要经过运算器，控制器负责解释指令，运算器负责执行指令。

3. 现代计算机的演化

冯·诺依曼计算机早期的结构是以运算器为中心的，输入、输出数据或程序都要经过运算器，运算也要通过运算器来进行。两种操作要争夺运算器资源，二者不能同时进行，即输入或输出时不能计算，计算时不能输入或输出。这样势必影响计算机的工作效率。

为此，人们对计算机的结构进行改进，现代计算机一般采用以存储器为中心的体系结构，如图 1- 23 所示。输入的数据或程序直接存储在存储器中，不经过运算器；运算器只负责运算；运算结果直接从存储器中取出，送给输出设备，也不经过运算器。这样，存储器可支持运算器与输入设备或输出设备的并行工作，即当存储器的一部分存储单元在进行输入、输出时，另一部分存储单元可为运算器提供数据存取服务。因此有效提高了计算机的工作效率。

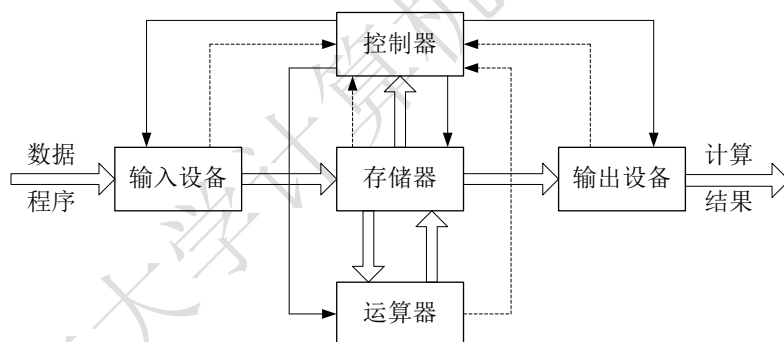


图 1- 23 以存储器为中心的现代计算机机构图

【微实例 1.2】冯·诺依曼计算机原理示例：分析计算 $x*a$ 的工作过程。

结合冯·诺依曼计算机的结构，分析计算 $x*a$ 的工作过程。其中 x 和 a 为操作数。

解：冯·诺依曼计算机计算 $x*a$ 的工作过程示意图如图 1- 24 所示。这里程序和数据都事先存储于存储器中。

计算 $x*a$ 的工作过程如下：

- (1) 启动控制器工作；
- (2) 控制器发送第 1 条指令地址给存储器；
- (3) 从存储器中取出指令返给控制器，并分析指令；
- (4) 执行指令：控制器发送操作数 x 所在地址给存储器；
- (5) 执行指令：从存储器中取出操作数 x ；
- (6) 控制器发送下一条指令地址给存储器；
- (7) 从存储器中取出指令，并分析指令；

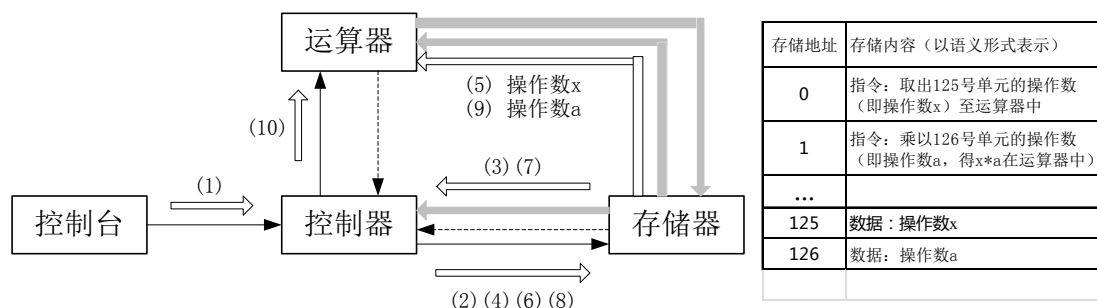


图 1-24 计算 $x*a$ 的工作过程示意图

- (8) 执行指令：发送操作数 a 所在地址；
- (9) 执行指令：从存储器中取出操作数 a ；
- (10) 执行指令：控制器通知运算器，计算 $x*a$ ；
- (11) 继续后续指令的取指、执行……

可见，控制器与存储器、运算器分工协作，协调工作，从而完成计算 $x*a$ 的整个过程。

1.4 信息在计算机中的表示

通过 1.2 节的学习可知：在计算机内部，无论何种信息，包括数字、字符、声音、图形、图像、视频等信息，都是用二进制数来表示的。进而了解了：计算机的主要部件，实际是由各种门电路逐级搭建而成的，而半导体二极管、三极管、场效应管又是组成门电路的主要开关器件。还了解了计算机的硬件基础知识——逻辑（布尔）代数以及逻辑运算。

但是，读者心中可能还是存有一些疑惑，例如：

- (1) 在计算机中，数字采用二进制表示，那么实数的小数点怎么表示？负数的负号又如何表示？
- (2) 如果在编程时，数字也用二进制来表示。当一个数很大时，岂不要写很长一串的二进制串？有没有更简单的表示方法？
- (3) 计算机可以进行算术运算和逻辑运算，甚至还能进行关系运算。是计算机本身具有进行这些运算的部件吗？如果不是，它是通过什么方法实现这些运算的？
- (4) 怎样把一个十进制数写成二进制数？又怎样把一个二进制数写成十进制数？
- (5) 在计算机中，文字、声音、图形和图像等也是用二进制表示的吗？如果是，它们是如何输入到计算机中，又是怎样被计算机识别、处理和存储，进而输出的？
- (6) 当需要输入汉字时，为什么在键盘上敲拼音，计算机就能识别是什么汉字，又能在屏幕上显示出这个汉字？

这些问题，正是本节要解答的内容。

现实世界中的信息究竟是如何传输到计算机中，又是怎样被计算机识别、处理和存储，进而输出的呢？实际上，各种信息的数据首先需要经过输入设备的转换，变成计算机能够识别的一种形式，即**数字信号**（在数值上和时间上的变化均不连续的信号，用 0 和 1 两种状态来表示信息），再送入计算机中进行处理和存储；然后再由输出设备，将数字信号转换为人们习惯和需要的形式输出。各类数据在计算机中的转换过程如图 1-25 所示。

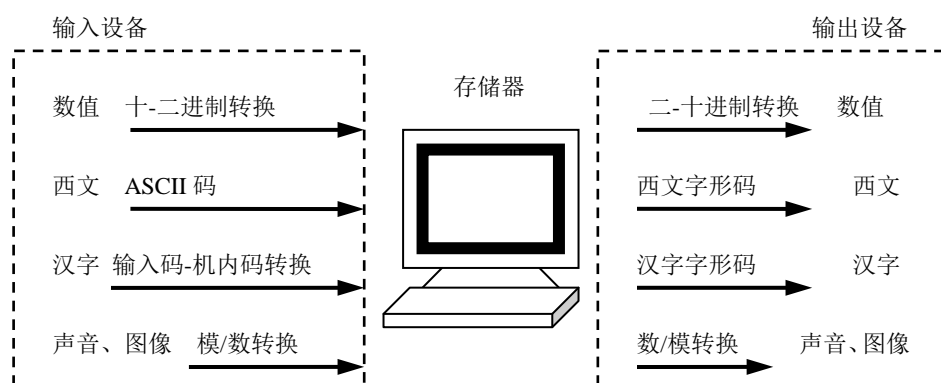


图 1-25 各类数据在计算机中的转换过程

例如，各种声音被麦克风（输入设备）接收，生成的电信号为**模拟信号**（在时间和幅值上的变化均连续的信号），必须经过一种称为模/数（A/D）转换器的器件，将其转换为数字信号，再送入计算机中进行处理和存储；然后通过一种称为数/模（D/A）转换器的器件，将处理结果的数字信号转换为模拟信号，人们通过扬声器（输出设备）听到的，才是连续的正常的声音。

1.4.1 计算机中的数据及其单位

在计算机内部，一切数据均采用二进制数表示。从 1.2.4.1 中可知，计算机中的数据又分为数值型数据和非数值型数据。

数值型数据有整数、实数之分，还有正数、负数之分。那么，在计算机内部，如何表示实数中的小数点以及正负数的正号和负号呢？

1.4.1.1 数值型数据的机器内部表示法

1. 数的长度和单位

二进制只有 0 和 1 两个数符。但对于实际的一个数，往往需要用多位二进制数码才能表示出来，其中的每一个数码称为 1 **位**（bit，比特）。比如对于正整数 77，可以用 7 位二进制数码 1001101 来表示，即该数的长度为 7bit。

位是计算机中数据的最小单位。但是，对于一个较大的数，若用 bit 来表示可能就太长了。于是，在 1956 年，IBM 公司在设计其第一台超级计算机 STRETCH 时，提出了字节（Byte）的概念，8 个二进制位称为 1 个**字节**。字节是现代计算机中数据存储和处理的基本单位。此外还有 KB、MB、GB、TB、PB、EB 等。

一个字节由 8 位二进制数字组成（即 1Byte = 8bit）。存储器的容量统一以字节（Byte，B）为单位。

- ✓ K 字节：1KB = 1024B = 2^{10} B
- ✓ M 字节：1MB = 1024KB = 2^{20} B
- ✓ G 字节：1GB = 1024MB = 2^{30} B
- ✓ T 字节：1TB = 1024GB = 2^{40} B
- ✓ P 字节：1PB = 1024TB = 2^{50} B
- ✓ E 字节：1EB = 1024PB = 2^{60} B

计算机一次能够并行处理的二进制数称为该机器的**字长**。随着电子技术的发展，计算机的并行处理能力越来越强，计算机的字长从 8 位、16 位、32 位，发展到今天微型机的 64 位，大型机已达 128 位。

字长是计算机的一个重要指标，直接反映一台计算机的计算能力和精度。字长越长，计算机的数据处理速度越快，精度越高。

2. 正负号的表示

正整数 77 可以表示为二进制数 1001101。但如果是负整数-77 呢？

在计算机中采取一种约定的方法解决这个问题：在数的前面增加一位符号位，该位为 0 表示正数，为 1 表示负数。假定一个数占 8 位，则十进制数+77 可以写作 0 1001101，十进制数-77 写作 1 1001101，这里最高位为符号位。

通常，用 0 或 1 表示正负号的数叫做计算机的“**机器数**”，机器数原来的数值形式叫做机器数的“**真值**”，也有的称做**尾数**。例如：真值 $(-1001101)_B$ ，其机器数为 1 1001101，存放在计算机中。机器数的表示方法如图 1-26 所示。



图 1-26 机器数

如果是实数 77.5 呢？在计算机内部，小数点怎样表示呢？

实际上，计算机内部没有专门设置小数点，它是通过默认小数点在什么位置，来解决实数的表示的。也就是说，小数点并未出现在数值中，其位置是隐含的。小数点具体位于什么位置，则与数的表示方法有关。一般地，计算机中的实数有两种表示格式：定点数表示法和浮点数表示法。

3. 定点数表示法

定点数是指小数点位置固定的数。定点数的长度也是固定的。

定点数分为两种：定点纯小数和定点纯整数。定点纯小数的小数点固定隐含在数值部分最高位的左边；定点纯整数的小数点固定隐含在数值部分最低位的右边。

定点纯小数如图 1-27 所示，定点纯小数的绝对值一定小于 1。例如，定点纯小数 1.1011101_B 表示小数 $-0.1011101_B = -0.7265625_D$ 。

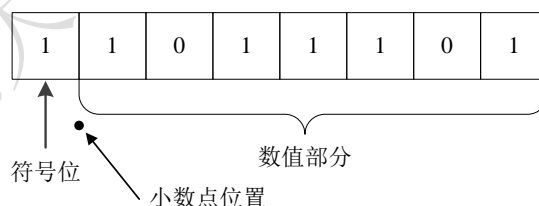


图 1-27 定点纯小数表示

定点纯整数如图 1-28 所示。例如，定点纯整数 11011101_B 表示整数 $-1011101_B = -93_D$ 。

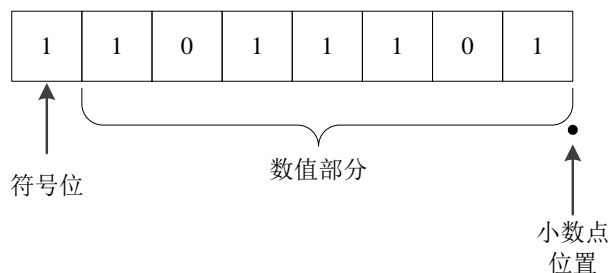


图 1-28 定点纯整数表示

早期的计算机只有定点数，没有浮点数。这种计算机的优点是，采用定点数表示的数是精确的，且计算机硬件结构简单。但是，它存在如下几个缺点：

其一，由于定点数的长度是固定的，所以数的表示范围有限。

比如，一台字长为 8 位的计算机，只能表示 -127~+127 之间的整数。对于字长为 m 位的计算机，其所能表示的整数范围为 $|N| \leq 2^{m-1}-1$ 。对于绝对值大于该范围的数，如果直接采用定点纯整数格式将会产生“溢出”（指在计算机中，当要表示的数据超出计算机所使用的数据表示范围时，产生的数据溢出），应根据实际需要适当地选择一个“比例因子”进行调整，使所表示的数据在规定的范围之内。进行运算后，再对运算结果按比例因子扩大。

其二，数据存储单元的利用率较低。

如果采用定点纯小数表示法，则所有参与运算的数据，必须除以其中最大的数，以转换为纯小数。但这样势必造成很多数据有大量的前置 0，从而导致大量数据存储单元的浪费。

其三，编程繁琐。

为了采用定点纯整数或定点纯小数表示法，编程人员对于所有参与运算的数据，必须根据计算机的小数点位置按一个“比例因子”进行扩大或缩小，以便将小数点统一对齐到该位置。运算后还要对运算结果进行相反的处理，以恢复正确的数值。这样势必给程序设计人员编程带来不便。

正是为了克服定点数的这些缺点，人们又发明了浮点数。

4. 浮点数表示法

浮点数是属于有理数中某特定子集的数的数字表示，在计算机中用以近似表示任意一个实数。所谓浮点就是指小数点在逻辑上是不固定的，即小数点的位置是浮动的。浮点数表示法其思想来源于基数为 10 的科学计数法，即用指数表示一个特大或特小的数。

例如：十进制数 234.5 的科学计数法为 0.2345×10^3 ，其中 0.2345 称为**尾数**，“ 10^3 ”中的 3 称为**阶码**。

在计算机中，一个浮点数包括阶码和尾数两部分，阶码和尾数都有符号位，称为**阶符**和**数符**。浮点数在计算机内的存储形式如图 1-29 所示。

阶符	阶码	数符	尾数
----	----	----	----

图 1-29 浮点数表示法

一般规定，阶码为定点整数，其小数点约定在阶码最右面；尾数为定点纯小数，其小数点约定在数符与尾数之间。因此，浮点数是定点整数和定点小数的混合。对于一个 32 位的浮点数，阶符和数符各占 1 位，阶码长度为 7 位，尾数长度为 23 位。

【微实例 1.3】浮点数表示法示例：将实数表示为浮点数。

将实数 234.5 写成 32 位的浮点数格式。

解：首先将十进制数转换为二进制数，具体方法见“1.4.2 进位计数制及其转换”。

$$234.5 = (11101010.1)_B$$

然后，将二进制数写为 2 的幂次方的形式：

$$(11101010.1)_B = (0.111010101)_B \times 2^8 = (0.111010101)_B \times 2^{(1000)}_B$$

可见，其阶码长度为 7 位，为 000 1000；尾数为长度为 23 位，为 111 0101 0100 0000 0000 0000。

则实数 234.5 的浮点数格式如图 1-30 所示。

在浮点数表示法中，阶码可取不同的数值，阶码的大小和正负决定了小数点的位置；尾数的大小和正负决定了数的有效数字，反映了数据的精度。可见，小数点的位置随阶码的变化而浮动，这正是“浮点”一词的由来。例如，234.5 的浮点数还可以写为 $(11.1010101)_B \times 2^6$ ， $(11101010.1)_B \times 2^0$ ， $(0.00111010101)_B \times 2^{10}$ 等。

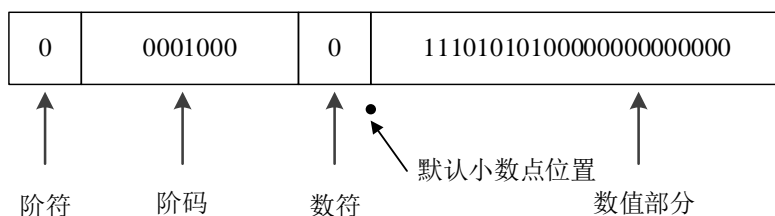


图 1-30 实数 234.5 的浮点数格式

浮点数表示法与定点数表示法的比较：

① **数值的表示范围**。浮点数表示法所能表示的数值范围远远大于定点表示法。

② **精度**。对于字长相同的定点数与浮点数来说，浮点数虽然扩大了数的表示范围，但这是以降低精度为代价的，也就是数轴上各点的排列更稀疏了。

③ **溢出处理**。定点运算时，当运算结果超出数的表示范围，就会发生溢出；而在浮点运算时，运算结果超出尾数的表示范围却并不一定发生溢出，只有当阶码也超出所能表示的范围时，才发生溢出。

④ **浮点数运算比定点数运算复杂**。例如浮点数的加法运算，首先要保证参加运算的数是规格化数；然后再“对阶”，即通过移动尾数使两个数的阶码相同；之后尾数才能相加；最后还要对运算结果进行规格化处理。为了加快浮点数的运算，在计算机硬件结构中专门设计了浮点运算部件。

1.4.1.2 原码、反码和补码

通过前面的学习可知，计算机不仅可以进行加法运算，而且可以进行减法、乘法、除法等算术运算，是计算机本身具有进行这些运算的部件吗？当然不是！因为如果是那样，计算机可就太复杂了。事实上，计算机的“大脑”——CPU 的核心部件就是加法器，而其他所有算术运算，都是转化为加法来实现的。那么，计算机究竟是通过什么方法，巧妙地实现这些转换的呢？

这就要引入原码、反码和补码这几个概念。原码、反码和补码是机器数在计算机中的不同编码方法。

1. 原码

我们知道，在计算机内的数用 0 和 1 来表示正号和负号，称为机器数。

原码是一种计算机对数值数据的二进制定点表示方法。其最高位为符号位（0 表示正数，1 表示负数），其余位表示数值的绝对值大小。通常用 $[X]_{\text{原}}$ 表示数据 X 的原码。

例如：

$$[+1]_{\text{原}} = 00000001 \quad [+127]_{\text{原}} = 01111111$$

$$[-1]_{\text{原}} = 10000001 \quad [-127]_{\text{原}} = 11111111$$

由上可知，8 位带符号定点整数原码表示的数的最大值为 127，最小值为 -127，表示数的范围为 -127 ~ 127。

【微实例 1.4】原码示例：求机器数的原码。

已知 $X = -0.1011011$ ， $Y = -1101001$ ，求 $[X]_{\text{原}}$ 和 $[Y]_{\text{原}}$ 。

解： $[X]_{\text{原}} = 1.1011011$ ， $[Y]_{\text{原}} = 1\ 1101001$



注意：0 的原码有两种表示形式，如采用 8 位带符号整数，有 $[+0]_{\text{原}} = 00000000$ ， $[-0]_{\text{原}} = 10000000$ ；如采用 8 位带符号纯小数，有 $[+0]_{\text{原}} = 0.0000000$ ， $[-0]_{\text{原}} = 1.0000000$ 。也就是说，0 的原码不是唯一的。

原码加法的运算规则是：

原码中符号位不参加运算；同符号数相加做加法；不同符号数相加做减法：较大数的数值部分减去较小数的数值部分，符号位取较大数的符号。

【微实例 1.5】原码加法示例：求两个机器数的原码加法。

已知 $X = +1001101$, $Y = -1101011$, 求 $[X+Y]_{\text{原}}$ 。

解: $[X]_{\text{原}}$ 、 $[Y]_{\text{原}}$ 的数值部分用 $|X|$ 、 $|Y|$ 表示:

$$|X| = 1001101, |Y| = 1101011,$$

显然 $|X| < |Y|$,

$$\text{则求出 } R = |Y| - |X| = 0011110,$$

$$\text{故 } [X+Y]_{\text{原}} = 1\ 0011110。$$

原码的优点是直观、简单易懂, 与其真值的转换方便。

但原码存在如下两个缺点:

其一, 运算过程复杂。进行异号原码的加法运算时, 需先判两数的大小, 然后从较大数中减去较小数, 最后还要判定结果的符号位。

其二, 运算器电路结构复杂。需要使用数值比较电路和减法运算电路, 而且增大了运算时间。因此, 在计算机中原码很少被采用。

2. 反码

其实反码在计算机中也很少使用, 但是它是一种编码方法, 主要用作求补码的中间码。


反码的表示方法是, 正数的反码与其原码相同; 负数的反码是将原码的数值部分各位取反 (0 变 1, 1 变 0), 符号位为 1。

例如:

$$[+1]_{\text{反}} = 0\ 0000001 \quad [+127]_{\text{反}} = 0\ 1111111$$

$$[-1]_{\text{反}} = 1\ 1111110 \quad [-127]_{\text{反}} = 1\ 0000000$$

同原码一样, 8 位带符号定点整数反码表示的数的最大值为 127, 最小值为 -127, 表示数的范围为 -127~127。

 **注意:** 0 的反码也有两种表示形式, 即 0 的反码也不是唯一的。如采用 8 位带符号纯小数, $[+0]_{\text{反}} = 0.0000000$, $[-0]_{\text{反}} = 1.1111111$; 如采用 8 位带符号整数 $[+0]_{\text{反}} = 0\ 0000000$, $[-0]_{\text{反}} = 1\ 1111111$ 。

【微实例 1.6】反码示例: 求机器数的反码。

已知 $X_1 = +0.1001010$, $X_2 = -0.1011011$, $X_3 = +1101001$, $X_4 = -1101001$, 求 X_1 、 X_2 、 X_3 、 X_4 的反码。

解: $[X_1]_{\text{反}} = 0.1001010$, $[X_2]_{\text{反}} = 1.0100100$,

$$[X_3]_{\text{反}} = 0\ 1101001, [X_4]_{\text{反}} = 1\ 0010110。$$


3. 补码

补码的表示方法是, 正数的补码与其原码相同; 负数的补码由在其反码的最低有效位上加 1 获得。

例如:

$$X = +1001101, [X]_{\text{原}} = 0\ 1001101, [X]_{\text{反}} = 0\ 1001101, [X]_{\text{补}} = 0\ 1001101$$

$$Y = -1001101, [Y]_{\text{原}} = 1\ 1001101, [Y]_{\text{反}} = 1\ 0110010, [Y]_{\text{补}} = [Y]_{\text{反}} + 1 = 1\ 0110011$$

 **注意:** 0 的补码只有一种形式, 即 0 在补码表示中是唯一的。如采用 8 位带符号纯小数, $[+0]_{\text{补}} = [-0]_{\text{补}} = 0.0000000$; 如采用 8 位带符号整数, $[+0]_{\text{补}} = [-0]_{\text{补}} = 0\ 0000000$ 。

【微实例 1.7】补码示例: 求机器数的补码。

已知 $X_1 = +0.1001010$, $X_2 = -0.1011011$, $X_3 = +1101001$, $X_4 = -1101001$, 求 X_1 、 X_2 、 X_3 、 X_4 的补码。

解: $[X_1]_{\text{反}} = 0.1001010$, 则 $[X_1]_{\text{补}} = 0.1001010$

$$[X_2]_{\text{反}} = 1.0100100, \text{则 } [X_2]_{\text{补}} = 1.0100101$$

$$[X_3]_{\text{反}} = 0\ 1101001, \text{则 } [X_3]_{\text{补}} = 0\ 1101001$$

$$[X_4]_{\text{反}} = 1\ 0010110, \text{则 } [X_4]_{\text{补}} = 1\ 0010111$$

【微实例 1.8】补码示例：求十进制数的补码。

采用 8 位带符号数补码表示下列各数： $(0.25)_D$ 、 $(-0.8125)_D$ 、 $(228)_D$ 、 $(-12)_D$ 。

解： $X_1 = (0.25)_D = (0.01)_B$ ， $[X_1]_{反} = 0.0100000$ ，则 $[X_1]_{补} = 0.0100000$ 。

$X_2 = (-0.8125)_D = (-0.1101)_B$ ， $[X_2]_{反} = 1.0010111$ ，则 $[X_2]_{补} = 1.0011000$ 。

$X_3 = (228)_D = (11100100)_B$ ，由于 8 位带符号整数补码的表示范围为 $-128 \leq X \leq +127$ ，而 228 超出了此范围，所以无法用 8 位带符号整数补码表示。

$X_4 = (-12)_D = (-1100)_B$ ， $[X_4]_{反} = 1\ 1110011$ ，则 $[X_4]_{补} = 1\ 1110100$ 。

利用补码可以方便地进行运算。一般地，进行补码加法运算的二进制数有效位数应比原二进制数的位数 n 多 1 位，才能确保任何一个 n 位二进制数的补码加法运算的结果不致溢出。再加上 1 位符号位，则采用 $(n+2)$ 位的二进制补码进行运算。

补码加法的运算规则是：补码的符号位和数值一起参加运算。若符号位产生了进位，则将进位舍弃。

假定 X 、 Y 是真值，则 $[X+Y]_{补} = [X]_{补} + [Y]_{补}$



注意：这里 X 、 Y 既可以是正数，也可以是负数。

引入补码的意义究竟是什么呢？下面通过一个实例来说明。

【微实例 1.9】补码加法示例：将减法运算用补码加法实现。

采用 8 位补码，计算 $13-7$ 。

解： $[13]_{原} = 0\ 0001101$ ， $[13]_{反} = 0\ 0001101$ ， $[13]_{补} = 0\ 0001101$ 。

$[-7]_{原} = 1\ 0000111$ ， $[-7]_{反} = 1\ 1111000$ ， $[-7]_{补} = 1\ 1111001$ 。

根据补码的运算规则，有：

$$\begin{array}{r} 0\ 0001101 \quad [13]_{补} \\ + 1\ 1111001 \quad [-7]_{补} \\ \hline 1\ 00000110 \quad [6]_{补} \end{array}$$

即： $[13]_{补} + [-7]_{补} = 0\ 0001101 + 1\ 1111001 = 1\ 00000110$ 。

可见 8 位补码进行加法运算后，结果为 9 位，其中最高位为进位。如果舍弃向最高位的进位“1”，则结果就是 000000110 ，为十进制数 6 的补码，与 $(13-7)$ 的运算结果是一致的。

这说明，可以采用补码加法，来代替减法运算。若要计算 $13-7$ ，只需求 $13+(-7)$ 的补码即可。

由此可见，利用补码可以方便地实现正数、负数的加法运算。补码规则简单，在数的有效存储范围内，符号位如同数值一样参加运算，无需单独处理，也允许产生最高位的进位（被丢弃），且最后的结果符号位仍然有效，所以补码被广泛地应用于数字系统和计算机中。

引入补码的意义在于：

其一，补码加法比原码加法运算简单。无需使用数值比较电路和减法运算电路，在很大程度上简化了运算器的电路结构，提高了运算速度。

其二，可将减法运算用补码加法实现。而乘法可以转化为加法，除法可以转化为减法，这样，加减乘除算术运算都只需用加法器实现，因而大大简化了 CPU 的设计。

通过引入补码，计算机就巧妙地实现了将所有算术运算都转化为用加法器完成。这种设计思想，体现了“通过约简、嵌入、转化和仿真等方法，求解问题”的计算思维方法。

1.4.2 进位计数制及其转换

在计算机内部，一切数据都是用二进制表示的。然而，在日常生活中，人们一般习惯采用十进制来表示数据，或者采用十二进制、六十进制等。编程或撰写文档时，为了书写方便，有时也采用八进制或

十六进制形式表示二进制数据。

那么，到底什么是进制呢？进制之间为什么需要转换？

1.4.2.1 进位计数制的概念

进位计数制（简称**进位制**，**进制**或**数制**）是用数码和带有权值的数位来表示有大小关系的数值型信息的表示方法。

如果采用 R 个基本符号（例如：0, 1, 2, ..., $R-1$ ）表示数值，则称 R 进制， R 称为该数制的**基数**（Radix），它是某种进位制所包含的数字符号的个数。而数制中表示基本数值大小的基本符号，称为**数码**。数制中某一位上的“1”所表示数值的大小（数码所处位置的价值）称为**位权**（**权值**）。

数码、基数和位权是进位制中的三个要素。处于不同位置的数码代表的值不同，与它所在位置的“权”值有关。例如，十进制数“3”如果放在个位，就代表3；如果放在十位，则代表30。

若某进制的基数为 R ，则第 i 位的位权为基数的 i 次方（ R^i ）。例如，十进制数“795”，其个位数5的位权是 10^0 ，十位数9的位权是 10^1 ，百位数7的位权是 10^2 。

任意一个 R 进制数 D 均可展开为：

$$(D)_R = \sum_{i=-m}^{n-1} k_i \times R^i \quad (1-5)$$

其中 R 为计数的基数； k_i 为第 i 位的系数，可以为 0, 1, 2, ..., $R-1$ 中的任何一个； R^i 称为第 i 位的权。

1.4.2.2 常用的进位制及表示方法

表 1-7 给出了二进制、八进制等几种常用的进位制。

表 1-7 几种常用的进位制的表示

进位制	基数	数码	权	形式表示
二进制	2	0, 1	2^i	B
八进制	8	0, 1, 2, 3, 4, 5, 6, 7	8^i	O
十进制	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	10^i	D
十六进制	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	16^i	H

表 1-7 中十六进制的数字符号除了十进制中的 10 个数字符号以外，还使用了 6 个英文字母：A, B, C, D, E, F，它们分别等于十进制的 10, 11, 12, 13, 14, 15。

在数字电路和计算机中，可以用括弧加数制基数（或该进制英文单词首字母）下标的方式，表示不同数制的数，如 $(25)_{10}$ 、 $(1101.101)_2$ 、 $(37F.5B9)_{16}$ ，或者表示为 $(25)_D$ 、 $(1101.101)_B$ 、 $(37F.5B9)_H$ 。

或在数的后面加上该进制英文单词的首字母，如 25D、1101.101B、37F.5B9H。

表 1-8 是十进制数 0~15 与等值二进制、八进制、十六进制数的对照表。

可以看出，采用不同的数制表示同一个数时，基数越大，则使用的位数越少。比如十进制数 15，需要 4 位二进制数来表示，只需要两位八进制数来表示，只需要 1 位十六进制数来表示。这也是为什么在程序或文档的书写中一般采用八进制或十六进制表示二进制数据的原因。


 **提示：**在数制中有一个规则，就是 N 进制一定遵循“逢 N 进一”的进位规则。如十进制就是“逢十进一”，二进制就是“逢二进一”，十六进制就是“逢十六进一”。

表 1-8 不同进制数的对照表

十进制	二进制	八进制	十六进制
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

1.4.2.3 不同进制间的转换方法

在数字电路和计算机中，采用二进制表示数据，而人们一般习惯于用十进制数，因此需要把十进制数转换为二进制数后，再送给计算机进行处理；而计算机处理后的二进制数结果也需要转换成十进制数或其他进制数显示，以便于人们识别。而人们在编写程序或撰写文档时，为方便二进制数据的书写，也需要将二进制数转换为八进制或十六进制数来表示。因此，需要掌握这些常用进制之间的转换方法。

1. R 进制数转换为十进制数

按照式 (1-5)，将 R 进制数按权展开求和即可得到相应的十进制数。

【微实例 1.10】R 进制数转换为十进制数示例。

分别将 $(1101.011)_2$ 、 $(376.65)_8$ 、 $(1FD.6C)_{16}$ 转换为十进制数。

$$\begin{aligned}
 \text{解: } (1101.011)_2 &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} \\
 &= 8 + 4 + 0 + 1 + 0.0 + 0.25 + 0.125 = (13.375)_{10} \\
 (376.65)_8 &= 3 \times 8^2 + 7 \times 8^1 + 6 \times 8^0 + 6 \times 8^{-1} + 5 \times 8^{-2} \\
 &= 192 + 56 + 6 + 0.75 + 0.078125 = (254.828125)_{10} \\
 (1FD.6C)_{16} &= 1 \times 16^2 + 15 \times 16^1 + 13 \times 16^0 + 6 \times 16^{-1} + 12 \times 16^{-2} \\
 &= (509.421875)_{10}
 \end{aligned}$$

2. 十进制数转换为 R 进制数

将十进制数转换为 R 进制数时，可将十进制数分成整数与小数两部分分别转换，然后再把结果拼接起来即可。方法如下：

① 整数部分转换采用“**除基取余法**”，直到商为零；每次相除所得余数为对应的 R 进制整数的各位数码，余数从右到左排列，首次取得的余数排在最右边。

② 小数部分转换采用“**乘基取整法**”，直到乘积的小数部分为零，或达到所要求的位数（当小数部分永不可能为零时）。每次相乘所得整数从小数点之后自左往右排列，取有效精度，首次取得的整数排在最左边。

【微实例 1.11】十进制数转换为二进制数示例。

将 $(62.625)_{10}$ 转换为二进制数。

解：如图 1-31 所示，整数部分和小数部分分别进行转换：

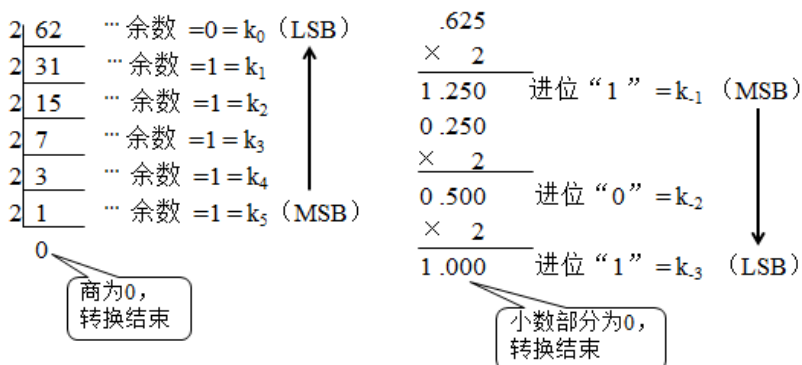


图 1-31 十进制数转换为二进制数的转换过程

将整数部分 62 逐次除以 2，所得余数写在旁边，直到商为 0，则结束转换。首次取得的余数为转换后整数部分的最低位，最后取得的余数为最高位。

将小数部分 .625 逐次乘以 2，所得进位（整数部分）写在旁边，每次对剩下的小数部分乘以 2，直到乘积的小数部分为零，或达到所要求的位数，则结束转换。首次取得的整数为转换后小数部分的最高位，最后取得的整数为最低位。

最后将两部分结果拼接起来，得到转换后的二进制数：

$$(62.625)_{10} = (111110.101)_2$$

【微实例 1.12】十进制数转换为十六进制数示例。

将十进制数 $(197.734375)_{10}$ 转换成十六进制数。

解：如图 1-32 所示，整数部分和小数部分分别进行转换：

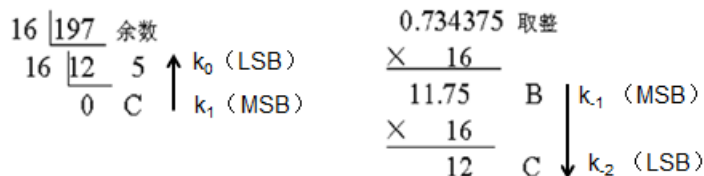


图 1-32 十进制数转换为十六进制数的转换过程

转换后的十六进制数： $(197.734375)_D = (C5.BC)_H$

3. 二进制数与八进制数或十六进制数的相互转换

当一个数很大时，若用二进制表示，位数太多，既不便于书写，也不便于阅读和记忆。这时可以用八进制或十六进制表示。因此需要将二进制数转换为八进制数或十六进制数。

$2^3=8$ ，即 3 位二进制数对应 1 位八进制数，而 1 位八进制数对应 3 位二进制数。故二进制数与八进制数之间的转换方法比较简单。

二进制数转换成八进制数的方法是，以小数点为界，向左（小数点之前）或向右（小数点之后），每 3 位二进制数用相应的一位八进制数取代（两头不足 3 位的二进制数先用 0 补足）。

八进制数转换成二进制数的方法正好与之相反，以小数点为界，向左或向右，每一位八进制数用相应的 3 位二进制数取代。

二进制数与十六进制数之间的转换方法跟二进制数与八进制数之间的转换类似。这里不再赘述。

4. 八进制数与十六进制数相互转换

八进制数转换为十六进制数的方法是，以二进制为桥梁，首先将八进制数转换为二进制数，再将二进制数转换为十六进制数。

同样，十六进制数转换为八进制数也要以二进制为桥梁。

1.4.3 字符的编码

在 1.4.1 中，介绍了数值型数据在计算机中的表示方法。那么非数值型数据在计算机中如何表示呢？不同类型（如字符、图形、图像、音频、视频等）的信息，有不同的表示方法，被称之为编码。

编码即是以若干位数码或符号的不同组合来表示非数值型信息的方法。编码具有唯一性和规律性。唯一性指每种编码组合都有唯一的含义，比如，每个公民都有唯一的身份证号码，以把每个人区别开来。规律性指编码具有一定的规律和特定的编码规则，比如某大学本科生的学号由 8 位数字组成，如 13061101，前 2 位表示入学年份，第 3、4 位表示院系代号，第 5、6 位表示班级，第 7、8 位表示序号。如果是硕士生，则在编码的最前面添加 SY 加以区分，博士生则在编码的最前面加 BY 加以区分。



提示：字符是计算机中最常见的一种数据形式，因此，本小节只介绍字符的编码方法。至于对图形、图像、音频、视频等的编码方法，有兴趣的读者请学习《多媒体技术》、《计算机图形学》和《视频编码》等课程。

字符包括西文字符（数字、英文字母、各种符号）、汉字以及其他国家的文字和符号等非数值型数据。由于计算机是以二进制的形式存储和处理数据的，因此在计算机内部，字符也必须统一采用二进制代码表示才能被计算机识别。用一组二进制代码来表示特定信息的方法称为二进制编码。用以表示字符的二进制编码称为字符编码。

字符编码的方法很简单，首先确定需要编码的字符总数，然后将每一个字符按顺序确定顺序编号，编号值的大小无意义，仅作为识别与使用这些字符的依据。字符形式的多少涉及编码的位数。对西文字符和中文字符，由于形式不同，使用不同的编码。

1.4.3.1 西文字符的编码

在计算机内部，西文字符的编码采用的是 ASCII (American Standard Code for Information Interchange, 美国信息交换标准交换代码)。ASCII 是单字节编码系统，1967 年由美国国家标准学会 (American National Standard Institute, ANSI) 制定，后来被国际标准化组织 (International Organization for Standardization, ISO) 指定为国际标准，称为 ISO/IEC 646 标准，适用于所有拉丁文字字母。它是不同计算机在相互通信时共同遵守的西文字符编码标准。

ASCII 码有 7 位码和 8 位码两种版本。国际通用的是 7 位 ASCII 码，称为**基础 ASCII 码**，即用 7 位二进制数表示一个字符的编码。由于 $2^7=128$ ，相应可以表示 128 个不同字符，包括 26 个大写英文字母、26 个小写英文字母、阿拉伯数字 0~9、通用运算符（+、-、×、÷ 等）及标点符号等共 32 个、控制字符或通信专用字符 34 个，如表 1-9 所示。8 位 ASCII 码称为**扩展 ASCII 码**，它允许将第 8 位二进制位用于确定附加的 128 个特殊符号字符、外来语字母和图形符号。许多基于 x86 的系统都支持使用扩展 ASCII 码。但扩展 ASCII 码不是国际标准。



提示：控制字符或通信专用字符又称为非图形字符，因为它们并没有特定的图形显示，但是会依不同的应用程序，而对文本显示产生不同的影响。控制字符有：BEL（响铃）、BS（退格）、LF（换行）、FF（换页）、CR（回车）、DEL（删除）等；通信专用字符有：SOH（文头）、EOT（文尾）、ACK（确认）等。

表 1-9 7 位 ASCII 代码表


$\begin{matrix} \backslash & b_6b_5b_4 \\ b_3b_2b_1b_0 \end{matrix}$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	.	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	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	↑	n	~
1111	SI	US	/	?	O	↓	o	DEL

表 1-9 中每个字符都对应一个数值，称为该字符的 **ASCII 码值**。其排列次序为 $b_6b_5b_4b_3b_2b_1b_0$ ， b_6 为最高位， b_0 为最低位，编码为 000 0000~111 1111。

34 个非图形字符以外的其余 94 个可打印字符，也称为**图形字符**。

其实 ASCII 码的编排是有一定规律的。从表中可以发现，0~9、A~Z、a~z 都是按顺序编排的。而且小写字母比大写字母的码值大 32，即对同一个字母，大写字母的位值 b_5 为 0，小写字母的位值 b_5 为 1，其余位的值分别相同。比如 A 的 ASCII 码为 100 0001，ASCII 码值为 65；a 的 ASCII 码为 110 0001，ASCII 码值为 97。这种规律有利于大写字母与小写字母之间的编码转换。

ASCII 码是一组二进制代码的组合，因此 ASCII 码值有大小之分。比较空格、数字、大写字母和小写字母的 ASCII 码可以发现，空格的 ASCII 码为 100 0000，ASCII 码值为 64，值最小；字母 z 的 ASCII 码为 111 1010，ASCII 码值为 122，值最大。利用这个特点可以对一些符号组合（如国家名、姓名、时间）进行排序。比如在 Windows 操作系统的资源管理器中，可以按文件名或修改日期对一个文件夹中的若干文件或子文件夹进行排序，实际上就是利用了 ASCII 码值有大小之分的特点。

 **注意：**计算机内部用一个字节（8 位）存放一个 7 位 ASCII 码，最高位为 0。

ASCII 码有什么用途呢？它是计算机与外部设备交换信息的字符编码。也就是说，通过键盘输入的信息，首先要转换成 ASCII 码，再由计算机进行处理和存储；计算机处理好的信息，也是用 ASCII 码输出给显示器或打印机的。

1.4.3.2 汉字的编码

ASCII 码只对西文字符进行了编码。为了使计算机能够处理、显示、打印和交换汉字字符，也需要对汉字进行编码。

当需要输入汉字时，为什么在键盘上敲拼音，计算机就能识别是什么汉字，又能在屏幕上显示出这

个汉字？实际上，计算机对汉字信息的处理过程就是各种汉字编码间的转换过程——那么，对汉字究竟是如何编码的呢？

在不同的场合，对汉字的编码方法不同。也就是说，汉字从输入，到存储，再到输出，计算机对汉字要进行一系列不同的处理。概括起来，汉字编码包括国标码、汉字机内码、汉字输入码、汉字字形码和汉字地址码。

- ✓ 国标码是我国汉字编码的国家标准。它使用两个字节表示一个汉字，每个字节的最高位为 0。
- ✓ 汉字机内码是计算机内部处理和存储汉字所使用的统一编码。把国标码每个字节的最高位置 1 即得到汉字机内码。
- ✓ 汉字输入码是利用键盘上按键的不同组合编码汉字，以便进行汉字输入的一种编码。它主要分为三类：数字编码、拼音编码和字形编码。
- ✓ 汉字字形码是用 0 和 1 编码无亮点和有亮点像素，形成汉字字形，以输出汉字的一种编码。
- ✓ 汉字地址码是访问存储汉字字形信息的汉字库的地址码。

【案例 1.2】汉字的编码过程案例：在键盘上输入汉字“大”，并在屏幕上显示。

下面针对此实例，逐步揭示计算机在不同汉字编码间的转换过程。

1. 国标码

1980 年，我国颁布了汉字编码的国家标准 GB 2312-80，其全称是“信息交换用汉字编码字符集—基本集”（简称**国标码**或**GB 码**）。其中共收录了 6763 个常用汉字（一级汉字 3755 个，二级汉字 3008 个）和 682 个图形符号。

但一个字节最多只能表示 256 种编码，是不足以表示 6763 个汉字的，因此，一个国标码使用两个字节来表示一个汉字，每个字节的最高位为 0。比如，“天”的国标码为 01001100_01101100，为便于书写，一般写成十六进制，即(4C6C)_H。

2. 汉字机内码（内码）

前面介绍了，计算机内部用一个字节存放一个 7 位 ASCII 码，最高位为 0。但是，计算机如何区分两个字节的数究竟是一个汉字的国标码还是两个西文字符的 ASCII 码呢？这就引出了汉字机内码的概念。

汉字机内码简称**内码**，是计算机内部处理和存储汉字所使用的统一编码。为了避免 ASCII 码和国标码同时使用时产生歧义，汉字机内码将国标码每个字节最高位置“1”，作为汉字机内码的标识。这样既解决了汉字机内码与西文机内码之间的二义性，又使汉字机内码与国标码具有极简单的对应关系。即：

$$\text{汉字的内码} = \text{汉字的国标码} + 8080_{\text{H}} \quad (1-6)$$

【微实例 1.13】汉字机内码示例：通过国标码求汉字机内码。

已知“天”字的国标码为(4C6C)_H，求其内码。

解：根据式 (1-6) 得：

$$\text{“天”字的内码} = \text{“天”字的国标码}(4\text{C}6\text{C})_{\text{H}} + (8080)_{\text{H}} = (\text{CCEC})_{\text{H}}$$

3. 汉字输入码（外码）

汉字机内码解决了汉字在计算机中的存储问题，但是，汉字又怎样输入到计算机中呢？换句话说，怎么让计算机知道用户想要输入的是什么汉字？

用于输入字符的键盘都是西文键盘，上面主要包括 26 个英文字母，数字 0~9，以及常用的标点符号和各种快捷键、控制字符等。如何利用西文键盘输入汉字呢？于是，人们发明了各种汉字输入码，来解决汉字的输入问题。

汉字输入码又称**外码**，它是利用计算机标准键盘上按键（字母和符号）的不同排列组合编码汉字，以进行汉字输入的一种编码。

汉字输入码主要分为三类：数字编码、拼音编码和字形编码。人们希望输入码既便于记忆，重码少，

输入速度又快。选择不同的输入码方案，则输入汉字的方法、按键次数和输入速度均有所不同。

数字编码就是用数字串代表一个汉字的输入，国标区位码和沿用多年的电报码都可以作为汉字输入码。它们编码用的字符是阿拉伯数字 0~9，每个汉字的码长为 4 位十进制数。其优点是无重码，缺点是代码难以记忆。**区位码**是用汉字在国标码中的位置信息编码汉字的一种方法。它将 GB 2312-80 中的 6763 个汉字分为 94 行（每行称为一个区）、94 列（每列称为一位），由区号和位号的 4 位十进制数字构成区位码，前 2 位为区号，后 2 位为位号。例如汉字“中”的区位码为 54 48，即它位于第 54 行、第 48 列。

拼音编码是以汉语读音为基础的输入方法。如智能 ABC、微软拼音输入法、紫光拼音、谷歌拼音、QQ 拼音、搜狗拼音输入法等。由于汉字同音字太多，输入重码率很高，因此，按拼音输入后还必须进行同音字选择，从而影响了输入速度。相比之下，搜狗拼音输入法由于具有全拼联想、庞大词库、简拼等诸多功能，所以输入效率较高。

字形编码是基于汉字的结构和笔画编码汉字的方法。全部汉字的部件和笔划是有限的，把汉字的笔划部件用字母或数字进行编码，按笔划书写的顺序依次输入，就能表示一个汉字。1983 年推出的五笔字型输入法就是典型的一种字形码。该方法输入速度快，但缺点是需要强背字根、入门较难。

4. 汉字字形码

汉字输入码解决了汉字输入到计算机中的问题。但是，存储在计算机中的汉字，只是一串二进制符号。那么汉字又是怎样从计算机中输出，在显示器屏幕上显示或者通过打印机打印出来的呢？

人们发现，同样可以用 0 和 1 的组合来表示汉字字形。比如，在一个 16×16 的网格中用点描出一个汉字，如“次”字，整个网格分为 16 行 16 列，每个小格用 1 位二进制数表示，有字形点的用 1 表示，没有字形点的用 0 表示，即用点阵来表示字形。这样就产生了汉字字形码，又称为**汉字字模**。**汉字字形码**是用 0 和 1 编码无亮点和有亮点像素，形成汉字字形的一种编码。如图 1-33 所示为“次”字的 16×16 字形点阵和编码。

根据输出汉字的精度要求不同，点阵的规模也不同。如简易型汉字为 16×16 点阵，普通型汉字为 24×24 点阵，提高型汉字为 32×32 点阵、48×48 点阵。点阵规模愈大，字形愈清晰美观，但所占存储空间也愈大。由于一个点用 1 位二进制数表示，8 位为一个字节，则 16×16 点阵的汉字需要 $16 \times 16 / 8 = 32$ 个字节的存储空间，48×48 点阵的汉字需要 $48 \times 48 / 8 = 288$ 个字节的存储空间。

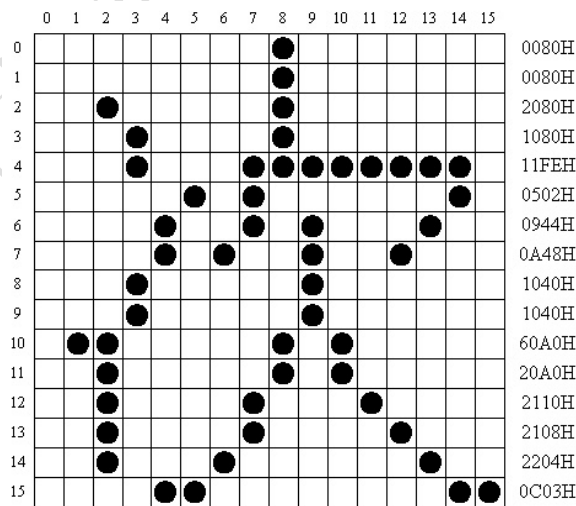


图 1-33 “次”字的字形点阵和编码

5. 汉字地址码

实际上，汉字字形信息是按一定顺序（大多数按国标码中汉字的排列顺序）连续存放在存储介质上的，这片连续的区域称为**汉字库**。汉字库中存储了每个汉字的点阵代码。一般对应不同的字体（如宋体、

黑体、楷体等)，有不同的字库。

当需要向输出设备输出汉字时，必须通过汉字地址码访问汉字库。**汉字地址码**是指汉字库中存储汉字字形信息的逻辑地址码。它大多连续有序，且与汉字内码间有着简单的对应关系，以简化汉字内码到汉字地址码的转换。

至此，学习了国标码、汉字机内码、汉字输入码、汉字字形码和汉字地址码。这些编码之间的关系前面已做介绍。那么，汉字从通过键盘输入到计算机中，直至显示或打印输出，都经过了哪些处理呢？

现在可以对本节开始的【案例 1.2】做出解答了。计算机对汉字信息的处理过程实际上是各种汉字编码间的转换过程，如图 1-34 所示。

(1) 首先通过键盘对每个汉字输入汉字输入码。对于“大”字，可以输入拼音输入码“da”，然后在弹出的可选汉字列表中选择“大”。则计算机自动根据“大”字的汉字输入码查到其对应的国标码“0 0110100_0 1110111”，再将国标码加上 $(8080)_{\text{H}}$ ，转换为机内码“1 0110100_1 1110111”，保存在计算机中。

(2) 输出汉字时，计算机先将汉字的机内码通过简单的对应关系转换为相应的汉字地址码；然后通过汉字地址码访问汉字库，从汉字库中提取汉字的字形码，最后根据字形数据在屏幕上显示汉字或通过打印机打印出汉字。

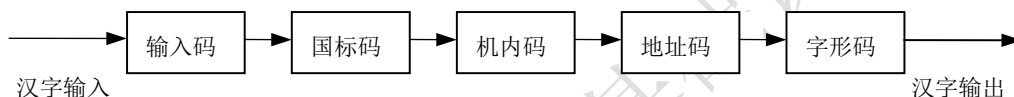


图 1-34 汉字信息的编码处理过程

1.5 计算机系统基本结构（自学材料）

现代计算机从冯·诺伊曼计算机发展至今，不断丰富、不断完善，功能不断扩大，其复杂程度已远远超过了当时定义的模式。计算机不再是单纯用于数值计算，而是可以应用于数据处理、过程控制、计算机辅助、网络通信、人工智能、多媒体应用、嵌入式系统等多个领域。人们的工作、生产、学习、生活和娱乐，已经离不开计算机，科学研究更是离不开计算机的帮助。

现代计算机不单纯是一台机器，它其实是一个复杂的系统，即计算机系统（Computer System）。因此，本节介绍现代计算机系统的组成以及其中蕴含的计算思维。在学习过程中，请思考以下问题：

- (1) 用户能不能直接操作计算机硬件？计算机的五大部件是靠什么协调工作的？
- (2) 是不是编程人员编写了程序，将它输入到计算机中，计算机就能够直接识别并执行？计算机到底能够理解与执行什么？
- (3) 程序设计语言分为哪三类？各有何特点？
- (4) 什么是操作系统？操作系统有哪四大功能？

1.5.1 计算机硬件系统

本小节从使用的角度，介绍现代计算机系统的构成、硬件组成，并重点介绍存储器以及不同性能的存储器如何优化组合成一个层次化的存储体系。

1.5.1.1 现代计算机系统的构成

简单地说，计算机由计算机硬件和软件组成，但严谨地说，现代计算机系统是由硬件（Hardware）、

软件（Software）和数据（Data）构成的。

硬件是构成计算机系统的物理实体，是看得见摸得着的实物。现代计算机的硬件结构大多遵循冯·诺依曼计算机的体系结构，即由运算器、控制器、存储器、输入设备和输出设备五大部分组成。

软件是控制硬件按指定要求进行工作的由有序命令构成的程序以及可能有的文件、文档和数据的集合，是系统的灵魂。没有安装软件的主机称为裸机，是不能按照人的要求自动工作的。


数据是计算机软件 and 硬件处理的对象。数据是对客观事物的逻辑归纳，用符号、字母等方式对客观事物进行直观描述，是表达知识的字符的集合。数据是信息的载体，比如数字、文字、符号、声音、图形、图像等都是数据。

1.5.1.2 计算机硬件的组成

（1）主机和外部设备

计算机硬件由主机和外部设备两大部分构成。

主机的前面板上有光盘驱动器（简称光驱）、多个 USB 口；后面板上各种外接端口（如打印机并口、VGA 显示器接口、网卡接口、PS2 键盘/鼠标接口、USB 口）。主机箱内部插有一块主电路板（简称主板），主板上插有内存条、CPU，还有若干插槽，可以插各种外设的接口电路板。

 **提示：**现在打印机、键盘和鼠标大多使用 USB 接口，故现在的主机上一般已不再设置并行端口、PS2 键盘/鼠标接口。

主机箱的外部是各种外部设备，外部设备包括输入设备和输出设备，它们通过不同的信号线与接口电路板相连接。输入设备和输出设备是计算机与外部世界进行信息交换的中介，是人与计算机联系的桥梁。

输入设备是将外界信息输入到计算机中的设备。它实现信息格式转换和传输速率的统一。即将人类理解和可读的各种信息（输入命令、程序、数据、文本、图形、图像、音频和视频等）转换为计算机能够识别的二进制代码输入计算机，供计算机处理。同时，它解决了输入信息的传输速率和计算机工作速率不匹配问题，提高了计算机的工作效率。常见的输入设备有键盘、鼠标、摄像机、照相机、扫描仪、光笔、触摸屏、游戏杆、语音输入装置、模/数转换器等。

输出设备是将计算机处理的结果以人们容易识别的形式输出的设备。常见的输出设备有显示器、打印机、绘图仪、语音输出装置、数/模转换器等。

有的设备既有输入功能，又有输出功能，例如，应用越来越普遍的触摸屏就是一种典型的输入/输出设备。它的屏是输出设备；在屏的表面安装了一种能感应手指或其他物体触摸的透明膜，将感应信息作为输入信号输入到计算机中。

计算机软件是一个复杂的系统，将在 1.5.2 节中专门介绍。

（2）CPU

CPU 是计算机的核心部件之一，它决定了计算机的速度和性能。CPU 的主要性能指标包括主频、字长、数据总线的位数、地址总线位数、内存寻址能力、包含的晶体管个数等。

主频是 CPU 内核工作时的时钟频率。一般说来，一个时钟周期内完成的指令数是固定的，主频越高，CPU 的速度也就越快，即计算机的运算速度越快。目前 CPU 主频最高达 4.7GHz。

字长是 CPU 一次能并行处理的二进制数的位数。字长的大小决定了计算机运算的精度和速度，字长越长，所能处理的数的范围就越大，运算精度越高；计算机处理数据的速度越快。CPU 的字长从最初的 4 位发展到 8 位、16 位、32 位直至现在的 64 位。

数据总线是一组用来在存储器、运算器、控制器和 I/O 部件之间传输数据信号的公共通路，是双向三态总线。数据总线位数是计算机的一个重要指标，它体现了传输数据的能力，通常与 CPU 的字长一致。

地址总线是 CPU 向主存储器和 I/O 接口传送地址信息的公共通路。地址总线位数决定了 CPU 可直接寻址的内存空间大小，简单地说就是 CPU 能够使用多大容量的内存。地址总线位数从最初的 16 位发展到 20 位、24 位、32 位，直至现在的 36 位。

内存寻址能力是 CPU 可直接寻址的内存空间大小。例如，8 位微机的地址总线为 16 位，则其最大可寻址空间为 $2^{16}=64\text{KB}$ ，16 位微型机的地址总线为 20 位，则其可最大寻址空间为 $2^{20}=1\text{MB}$ 。因此，随着 CPU 地址总线位数的扩展，CPU 的内存寻址能力也从 64KB 逐步发展到 1MB、16MB、4GB 直至现在的 64GB。

包含的晶体管个数：一片 CPU 上包含的晶体管个数越多，CPU 集成度就越高，处理数据的能力就越强。

过去的计算机一般只有一片 CPU，但随着微电子技术的发展和制造工艺的进步，多核（Core）处理器芯片开始出现，如双核或 4 核 CPU，这里的核一般指在一片处理器芯片上集成有多个 CPU。目前家用台式计算机的 CPU 最高有 8 核 CPU，工作站最高有 18 核 CPU。而在大型计算机系统中，往往包含多个处理器芯片，每个 CPU 都是多核心的。多核 CPU 使得多个任务可以并行执行，从而大大提高了计算机的运行效率。

1.5.1.3 存储器及层次化存储体系

（1）存储器及其分类方法

存储器是计算机系统中的记忆设备，是计算机的另一核心部件。存储器有不同的分类方法，比如，按存储介质分为：半导体存储器，采用半导体材料制成；磁表面存储器，采用磁性材料制成；光盘，利用光学原理制成。

按读写功能分为：只读存储器（Read Only Memory，ROM），存储的内容是固定不变的，是工作时只能读出而不能写入的半导体存储器；随机读写存储器（Random Access Memory，RAM），是工作时既能读出又能写入数据的半导体存储器。

按照存储器与 CPU 的相近程度，以及它们所处的位置，存储器可分为两大类：主存储器或内存储器（简称主存或内存），外部存储器或辅助存储器（简称外存或辅存）。

① ROM 和 RAM

ROM 由半导体材料制作而成，使用中（指正常工作状态下）只读不写。与 RAM 一同管理，可按地址访问。其优点是电路结构简单；掉电后信息不会丢失。缺点是不能修改或重新写入数据，只适用于存储固定数据。因此，ROM 一般用于存放启动计算机所需的少量程序和参数信息，如监控程序、基本输入/输出系统模块 BIOS（Basic Input Output System）等。计算机出厂前由厂家固化，用户无法修改。

RAM 又分为静态 RAM（Static RAM，SRAM）和动态 RAM（Dynamic RAM，DRAM）。SRAM 采用触发器作为存储单元，使用中可读可写，不需要刷新。存取速度比 DRAM 快，但制造工艺比较复杂，集成度不如 DRAM 高，价格比 DRAM 贵。后面要介绍的 Cache 就是使用的 SRAM。

DRAM 采用电容存储信息，使用中可读可写。DRAM 中“动态”的含义是指每隔一个固定的时间必须对存储信息刷新一次。因为随着时间的推移，电容上的电荷会逐渐减少，为保持其内容必须周期性地对其进行刷新（对电容充电）。其结构非常简单，集成度远高于 SRAM，价格比 SRAM 便宜；但存取速度不如 SRAM 快。后面要介绍的内存就是使用的 DRAM。

② 内存和外存

内存由半导体存储器 RAM 组成，用来暂存当前正在使用的数据、正在执行的程序以及运算结果，断电后其上所存储的信息将全部丢失。内存位于计算机的主板上，直接与 CPU 互连。内存的主要特点有：可以被 CPU 直接访问，容量小（MB/GB 级），存取速度快（访问一个存储单元的时间在 ns 级）；可读可写，掉电则数据消失。

内存一般采用 DRAM。PC 机中的内存将多片 RAM 制作在一个条形电路板上，因此俗称**内存条**。

内存条插于主板上的内存插槽中，这样便于内存容量的扩展。单条内存条的容量有 2GB、4GB、8GB、16GB、32GB 等。


外存一般由磁性材料或光学材料制成，用来存放各种数据文件和程序文件等需要长期保存的信息。CPU 不能直接存取外存中的程序和数据。外存中的程序和数据必须先调入内存，才能被 CPU 读取。外存的主要特点有：容量大（GB/TB 级），存取速度慢（读取磁盘一次的时间在 ms/s 级），断电后所保存内容不会丢失。

常见的外存有利用磁粒子极性记录数据的硬盘、利用激光照射技术记录数据的光盘、利用 EEPROM 技术记录数据的快闪存储器（Flash Memory，简称闪存）等。过去常用的软盘，在很长一段时期内，是主要的一种外存，但由于其容量小、易损坏，现在已被淘汰，取而代之的是光盘和 U 盘（闪存的一种，移动数据存储设备）。

硬盘（Hard Disk）是 PC 机主要的外部存储设备，一般采用磁性材料制作，可永久保存信息。操作系统、应用程序和用户的数据文件一般都保存在硬盘上。硬盘是根据电磁学原理，使得涂敷在磁盘片上的磁性材料被磁化时具有两种极性来记录二进制信息的。这种极化状态不会受到断电的影响。

硬盘的性能指标主要包括硬盘容量和转速。硬盘容量指硬盘的总存储容量，是由单碟容量和碟片数量决定的。目前有 320GB、500GB、750GB、1TB、1.5TB、2TB、3TB、4TB、6TB、8TB 甚至 10TB 等。不同的容量，价格不同。

硬盘转速是指硬盘内电机主轴的旋转速度，即硬盘盘片在一分钟内旋转的最大转数，单位为 RPM（Revolutions PerMinute，转/每分钟）。目前有 3600、4500、5400、5900、7200 甚至 10000 RPM。

 **提示：**上面介绍的硬盘为机械硬盘，还有一种硬盘称为固态硬盘（Solid State Disk，SSD），它是用

固态电子存储芯片阵列制成的硬盘，采用闪存作为存储介质，没有机械部件，无需专门的机械驱动供电。与传统硬盘相比，具有快速、轻便、耐震、省电和无噪音等优点，因此逐渐受到人们的青睐。它特别适于作为移动数据存储设备，比如笔记本硬盘、微硬盘、存储卡、U 盘等。基于闪存的固态硬盘是目前主流产品。

但与机械硬盘相比，固态硬盘也有其缺点：容量不够大（目前市场上能够稳定运行的民用级 SSD 其最大容量是 2TB，企业级 SSD 最大容量为 16TB，即三星的 PM1633a。随着技术的不断进步，近期一些闪存厂商又将 SSD 整合容量提高到了 32TB 甚至 60TB），价格贵，写入寿命有限，数据损坏后难以恢复。

（2）层次化存储体系

人们希望存储器满足以下几个要求：

- ① 存储器工作速度应与 CPU 匹配，否则 CPU 工作效率不高，浪费资源；
- ② 存储器容量越大越好，但价格越便宜越好；
- ③ 存储器应能够永久保存程序和数据。

首先来看速度匹配的问题。在冯·诺依曼计算机结构中，CPU 直接与内存互连，程序和数据都保存在内存中。CPU 需要不断从内存中读取指令、读取数据，处理的结果也需要保存到内存中。但因为内存的速度（目前主频最高 4266MHz）比 CPU（目前主频最高达 5GHz）低，造成 CPU 不得不经常停下来，等待指令和数据，使得 CPU 的高速处理能力不能充分发挥。因此，人们采用高速缓冲存储器（简称**缓存**）Cache 来解决 CPU 与内存之间速度不匹配的问题。

Cache 采用速度比 DRAM 快、容量小于 DRAM、价格更高的 SRAM。Cache 容量小于内存，但速度比内存高得多，接近于 CPU 的速度。

Cache 的理论依据是程序局部性原理。其一，时间的局部性（Temporal Locality）：最近被访问的内存内容（指令或数据）很快还会被访问；其二，空间的局部性（Spatial Locality）：靠近当前正在被访问内存的内存内容很快也会被访问。

根据程序局部性原理，当 CPU 存取内存某一单元时，计算机硬件自动将包括该单元在内的那一组单元内容调入 Cache，即 Cache 在任何时候其内容都只是内存中一部分内容的拷贝。CPU 从内存中读取指令或数据时，同时访问 Cache：若所需内容在 Cache 中，立即获取；若没有，再从内存中读取。Cache 的存取速度比内存快得多，这样就有效提高了计算机的处理速度。

其次，来看存储器的容量与价格。内存采用 RAM 存储器，其容量在 GB 级。容量越大，价格越高。故内存容量不可能无限制地增大。而且，内存也不能满足第 3 个要求，它只是暂存程序和数据，一旦计算机关机或掉电，则内存中存储的信息全部丢失。

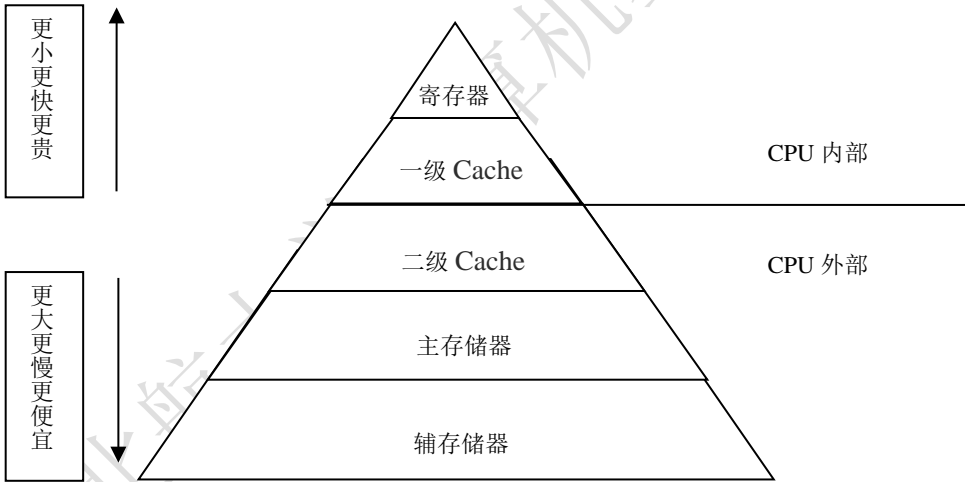
为此，人们发明了辅助存储器（外存），来弥补内存的不足。相比内存，外存容量大得多，价格也低得多。目前机械式硬盘容量最大已达到 10TB。更重要的是，外存可以永久保存程序和数据，即使掉电，所存储信息也不会丢失。

现代计算机通常采用 Cache、主存（内存）和辅存（外存）这种层次化的存储体系，通过优化组合不同性能的资源，来达到存取速度、存储容量、保存时间、价格之间的一个平衡，从而提高计算机的整体性能。在计算机操作系统的管理和协调下，三个层次的存储器之间有条不紊地传递着数据（程序和数据平时存储在辅存中，需要时被调入主存，CPU 从主存中读取指令或数据），不仅解决了速度和容量的问题，也提高了计算机的性价比。这种设计思想，体现了计算思维“在时间和空间之间、在处理能力和存储容量之间进行折中”的思维方法。

现代计算机的三级存储系统如图 1-35 所示。

为进一步改善存储系统的性能，人们将 Cache 分为一级 Cache 和二级 Cache。一级 Cache 可以以 CPU 的速度访问，位于 CPU 内；二级 Cache 采用 SRAM，速度稍逊于一级 Cache，位于 CPU 外。

“Cache—主存”解决了 CPU 和主存速度不匹配问题。“主存—辅存”解决了存储器系统的容量问题，大量的数据和文件平时存储在辅存里，需要运行的程序和使用的数据临时调入主存，一旦程序退出，则释放主存。



将图 1-35 中不同级别的存储器的存取速度和容量做一对比，如表 1-10 所示。

表 1-10 不同级别的存储器性能对比

存储器	存取速度	存储容量
寄存器	最快	最低
一级 Cache	一次存取时间约 10ns	4~16MB
二级 Cache	一次存取时间约 20~30ns	128~512MB
主存储器（内存）	一次存取时间约 60ns	1~32GB
硬盘（外存）	读取磁盘一次的时间约 12ms	500GB~10TB

1.5.2 计算机软件系统

硬件是计算机系统的物质基础，而软件则是计算机系统的灵魂。软件是用户与硬件之间的接口，用户通过软件使用计算机硬件资源。如果仅有硬件而没有软件，计算机（此时称为裸机）是无法工作的。因此，软件与硬件同等重要。

1.5.2.1 软件系统的构成

国标中对**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。

计算机软件系统分为系统软件（System Software）和应用软件两大类，如图 1-36 所示。

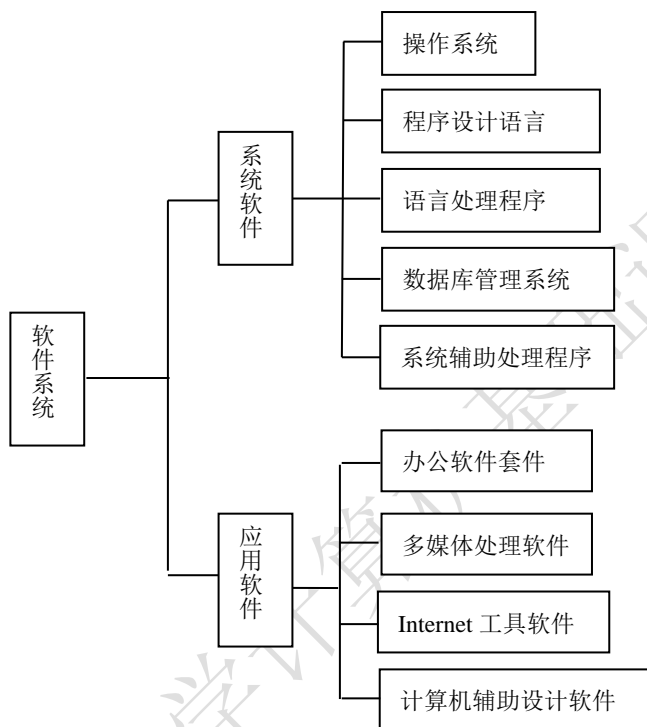


图 1-36 计算机软件系统的组成

1. 系统软件

系统软件是指管理、控制和协调计算机及外部设备，支持应用软件开发和运行的软件。系统软件的主要功能是调度、监控和维护计算机系统；管理计算机系统中各独立硬件，使它们能够协调工作。系统软件使得底层硬件对计算机用户是透明的，用户在使用计算机时无需了解硬件的工作过程。

系统软件的主要特征是：与硬件有很强的交互性；能对资源共享进行调度管理；能解决并发操作处理中存在的协调问题；数据结构复杂，外部接口多样化，便于用户反复使用。

系统软件是软件的基础，所有应用软件都要在系统软件上运行。系统软件中最重要的是操作系统（Operating System）；除此之外，还有程序设计语言、各种语言处理程序、数据库管理系统以及帮助用户开发软件的各种软件工具等。

（1）操作系统

操作系统是管理、控制和监督计算机软件及硬件资源协调运行的程序系统，由一系列具有不同控制和管理功能的程序组成。它是直接运行在计算机硬件上的、最底层的软件，是计算机裸机与应用程序及用户之间的桥梁。没有它，用户无法使用其他软件或程序。常用的操作系统有 Windows、Linux、DOS、Unix、Mac OS 等。

当初冯·诺依曼等人提出冯·诺依曼机模型时，并没有操作系统的概念。机器的整个执行过程完全由

人来掌控，即单一控制终端、单一操作员模式。但是随着计算机越来越复杂、功能越来越多，人已经没有办法直接掌控计算机。于是，人们编写操作系统来代替人掌控计算机，将人从日益复杂繁重的任务中解脱出来。

操作系统具体的功能将在“2、操作系统的功能”中介绍。

（2）程序设计语言

在现实世界中，人类通过自然语言进行交流。自然语言由字、词、句、段、篇等构成。而人使用程序设计语言（Programming Language）与计算机沟通。**程序设计语言**是用于书写计算机程序的语言，一般由单词、语句、函数和程序文件等组成。它是软件的基础和组成。

自 20 世纪 60 年代以来，世界上公布的程序设计语言已有上千种之多，但是只有很小一部分得到了广泛的应用。从程序设计语言的发展历程来看，可以分成机器语言、汇编语言、高级语言三大类。

① 机器语言

想一想，是不是我们编写了程序，将它输入到计算机中，计算机就能够直接识别并执行呢？问题并没有那么简单。计算机到底能够理解与执行什么呢？

在所有程序设计语言中，机器语言是唯一能被计算机硬件系统理解和执行的语言。**机器语言**是直接由二进制代码指令表达的计算机语言。例如“10110000 00000011”表示将数 3 送入寄存器 AL 中。

机器语言的优点是处理效率最高，执行速度最快，且无需“翻译”。

但是，机器语言并不便于人们使用，既难以记忆，书写起来又很繁琐、易出错，调试、修改、移植和维护都非常繁琐。如何克服机器语言的缺点呢？

② 汇编语言

如果将二进制编码的指令用便于记忆和书写的符号如英文单词或缩写来表示，并用这些符号编写程序，再设法将程序翻译成机器语言程序，问题便迎刃而解。于是就出现了汇编语言。

汇编语言是用助记符号编写程序的语言。如“MOV”表示传送指令，“ADD”表示加法指令等。用汇编语言编写的程序就称为**汇编语言源程序**。

但计算机无法识别和自动执行汇编语言源程序，还必须使用语言处理软件将汇编语言翻译成机器语言，计算机才能够理解和自动执行。将汇编语言源程序翻译成机器语言程序的程序称为**汇编程序**，又称**汇编语言编译器**。

例如，对于传送数据 3 到寄存器 AL 中的机器语言“10110000 00000011”，可以用汇编语言“MOV AL, 3”来代替。

③ 高级语言

尽管用汇编语言编写程序比用机器语言编写程序方便了许多，但仍存在一些问题。比如，指令必须逐条书写，不易编写大程序，不利于复杂算法的实现；编程人员需要理解硬件的结构和操作的细节。

能不能像写数学公式一样编写程序呢？即先按类似于自然语言方式书写程序，然后再将其自动翻译成机器语言呢？

于是，人们又设计了这样一套规范/书写标准，以语句为单位编写程序，并开发了一个翻译程序，实现了将语句程序自动翻译成机器语言程序的功能。

高级语言就是用类似算术语言和自然语言的语句编写程序的语言，由单词、语句、函数和程序文件等组成。例如，在 C 语言中，采用“+”、“-”、“*”、“/”表示加、减、乘和除运算，用“if”和“else”作为条件语句的关键字，用 fopen 函数实现打开文件，用 fclose 函数实现关闭文件等。非常直观，便于书写、理解和记忆。按照高级语言的发展历史，高级语言有 FORTRAN、Pascal、C、C++、Java、C#等。

用高级语言编写的程序即称为**高级语言源程序**。同样，计算机无法识别和自动执行高级语言源程序，也必须先经过翻译。将高级语言源程序翻译成等价的机器语言程序的程序称为**编译程序**。

高级语言是面向用户的、基本上独立于计算机种类和结构的语言。其最大的优点是：形式上接近于算术语言和自然语言，概念上接近于人们通常使用的概念。高级语言的一条语句可以代替几条、几十条甚至几百条汇编语言的指令。因此，高级语言易学易用，通用性强。与机器语言和汇编语言相比，用高级语言编写的程序最简洁，最易于理解。

（3）语言处理程序

如前所述，计算机只能直接识别和执行机器语言。如果要在计算机上运行高级语言程序，就必须配备程序语言翻译程序，即语言处理程序。翻译程序本身是一组程序，不同的高级语言都有相应的翻译程序。语言处理程序包括汇编程序、编译程序和链接程序等。

人们编写的高级语言程序，是如何被计算机识别并执行的呢？

人先用高级语言编制高级语言源程序；然后编译程序首先将其翻译成汇编语言程序；再由汇编程序将其翻译成机器语言程序，则计算机识别它并自动执行。

（4）数据库管理系统

在信息社会里，社会和生产活动产生的信息越来越多，对信息的管理和共享也越来越重要。人们希望借助计算机对信息进行搜集、存储、处理和使用。于是，出现了数据库（Database，DB）和数据库管理系统（Data Base Management System，DBMS）。

数据库是指按照一定联系长期储存在计算机内的、有组织的、可共享的数据集合。比如一个单位的员工档案、学校图书馆的图书信息、物流公司的物资信息都可以创建数据库来进行管理。**数据库管理系统**则是工作在操作系统之上、能够对数据库进行创建和管理的大型软件。其主要功能包括建立、删除、维护数据库及对库中数据进行各种操作（如检索、修改、统计、排序和合并等）。常用的数据库管理系统有 Oracle、DB2、Access、SQL Server、Sybase 和 Foxpro 等。

（5）系统辅助处理程序

系统辅助处理程序也称为“软件研制开发工具”、“支持软件”、“软件工具”，主要有编辑程序、调试程序、装备和连接程序。

2. 应用软件

应用软件是为解决各类实际问题、进行业务工作或者与学习、生活、娱乐相关而设计的程序系统，包括办公软件套件、多媒体处理软件、Internet 工具软件、计算机辅助设计软件、数值计算类软件以及游戏软件等。

办公软件是日常办公需要的一些软件，它一般包括文字处理软件、电子表格处理软件、演示文稿制作软件、个人数据库、个人信息管理软件等。

多媒体技术的迅速发展，极大地改变了人们的工作和生活方式，各种各样功能强大的多媒体处理软件也不断涌现，主要包括：图形处理软件，图像处理软件，动画制作软件，视频编辑工具，桌面排版软件等。

随着计算机网络技术的发展和 Internet 的普及，涌现了许许多多基于 Internet 环境的应用软件。如各种 Web 服务软件，文件传送工具 FTP，远程访问工具 Telnet，Web 浏览器，即时通讯（Instant messaging，IM）软件，下载工具等。

1.5.2.2 操作系统的功能

操作系统控制和管理计算机的所有资源，包括硬件资源、软件资源和信息资源等，调度计算机各部件协调工作，从而高效、自动地执行程序。它是最基本、最重要的系统软件，也是对计算机硬件功能进行扩展的软件系统。

操作系统是计算机系统的控制和管理中心，从资源角度来看，操作系统通常包括四大功能模块：处理机管理，内存管理，文件管理，设备管理。它们是操作系统的组件，本质上就是程序。

1. 处理机管理（进程管理）

处理机管理指管理 CPU 资源，当多个程序同时运行时，解决 CPU 时间的分配问题。处理机管理也称为**进程管理**，换句话说，当有多个进程要执行时，操作系统根据一定的调度算法调度 CPU 执行某个进程。

计算机可以同时做多件事情吗？比如，人们可以边听音乐，边使用 Word 写论文，边使用 PowerPoint 做演示文稿……

可以的！这就是由操作系统的处理机管理实现的，即一个 CPU 可以执行多个进程。那么，怎样让所有进程（及进程相关的用户）都感觉到其是在独占 CPU 呢？这是通过分时调度策略实现的。

分时调度策略的思想是，把 CPU 的被占时间划分成若干段，每段间隔特别小，CPU 按照时间段轮流执行每个进程，使得每个进程都感觉其在独占 CPU。分时调度策略有效解决了多任务共享使用单一资源的问题。如图 1-37 所示为单 CPU 分时调度策略示意图。

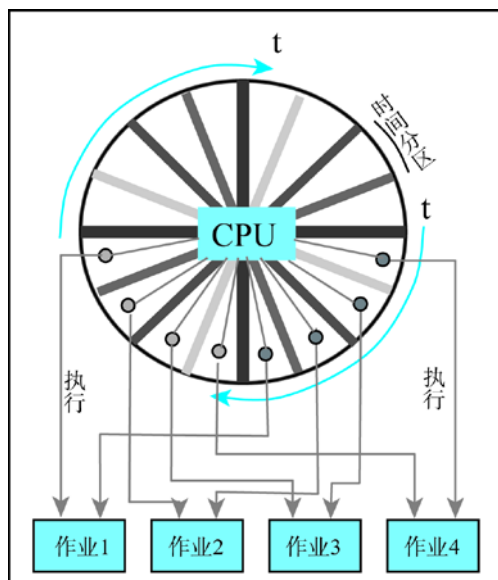


图 1-37 单 CPU 分时调度策略

提示：什么是进程（process）？进程是指进行中的程序（program）。它是程序的一次执行过程，是系统调度和资源（内存、CPU 时间片等）分配的基本单位。当程序存放在磁盘上时，它只是程序。一个程序被加载到内存，系统就创建一个进程，程序执行结束后，该进程随即消亡。程序和进程的关系犹如剧本和演出的关系。其中，程序是静态的，而进程是动态的。只有被装入内存的程序和数据，才有可能被 CPU 执行和处理。简单理解，进程即内存中的可执行程序。

2. 内存管理


有时需要装入内存的程序较多，CPU 需要处理的数据量也比较大，程序和数据不能一次性装入。怎样充分利用有限的内存空间，使得多个程序能够协调执行，CPU 的使用效率又足够高呢？这就需要合理对内存进行管理。

内存管理指管理内存资源，为各个程序及其使用的数据分配存储空间，有效地管理、分配、利用和回收内存，以提高内存的利用率和 CPU 的使用效率。其主要功能有：

- ① 管理内存的存储空间：记录内存空间是已分配还是空闲；
- ② 内存分配：按一定的策略为每道正在处理的程序和数据分配内存空间；
- ③ 内存与外存数据的自动交换：当内存空间不够时，通过腾挪手段腾出空间；
- ④ 内存空间的回收：当某个程序工作结束时，及时回收其占用的内存空间以供其他程序使用；

⑤ 内存保护：建立内存保护机制，保证各程序在自己的内存区域内运行而不相互干扰；

⑥ 内存扩充：借助虚拟存储（Virtual Memory）技术（内存和外存联合起来统一使用）实现增加内存的效果。

 **提示：**虚拟存储技术是指用硬盘空间模拟内存，在硬盘上为用户开辟一个比实际内存大得多的内存空间，并按内存的结构进行组织。在计算机运行过程中，部分进程保留在内存；其他暂时不在 CPU 运行的进程驻留在外存，当需要调入内存时，操作系统直接进行映射操作，避免了磁盘数据格式向内存存储数据格式的转换，从而提高了运行效率。

3. 文件管理

文件管理指操作系统对文件的存储、检索、共享和保护，为用户提供文件操作的方便。

计算机中的**文件（File）**是按一定格式建立在存储设备上的一批信息的有序集合，并标记以一个名字。各种信息被操作系统组织成文件，文件是操作系统管理信息的基本单位。

文件名以字母和数字的组合唯一标识一个文件，不同操作系统的文件命名规则不同。在 Windows 9x 及其后版本中，文件名为 1~255 个字符（包括空格），其后通常具有 3~4 个字母的文件扩展名，用于指示文件类型，如声音文件（.wav）、图像文件（.jpg）、位图文件（.bmp）、Word 文档文件（.docx）、文本文件（.txt）、C 语言源程序文件（.c）、Python 语言源程序文件（.py）等。文件扩展名与文件名之间用“.”分隔，因此扩展名也叫做“文件后缀”或“后缀名”。

 **提示：**应用程序的可执行文件后缀一般为.COM 或.EXE，双击带有这样后缀的文件名，就可以立即执行该应用程序。

用户不必关心文件在磁盘上是如何存取的，只需关注文件名和文件内容即可。文件在磁盘上的存取是由操作系统自动实现的。

4. 设备管理

设备管理用于管理接入计算机系统的所有外部设备，为用户进程分配其所需的 I/O 设备，提高 CPU 和 I/O 设备的利用率；提高 I/O 速度；方便用户使用 I/O 设备。由于管理的是 I/O 设备，所以设备管理也称为 I/O 管理。


设备管理的主要任务包括：

① 设备分配：接收用户进程提出的 I/O 请求，根据用户进程的 I/O 请求、系统的现有资源情况以及某种设备分配策略，为之分配其所需的设备；

② 设备传输控制：启动设备、中断处理、结束处理等；

③ 确保设备独立性（设备无关性）：使用户程序与实际使用的物理设备无关；

④ 缓冲管理：为解决 CPU 运行的高速性与 I/O 低速性之间的矛盾，在现代计算机系统中，都在内存中设置了缓冲区，并通过增加缓冲区容量的方法，来改善系统的性能。有效地缓和 CPU 与 I/O 设备速度不匹配的矛盾，提高 CPU 的利用率，进而提高系统吞吐量。

 **提示：**关于设备驱动程序。用户和应用程序并不能直接使用设备，而是通过操作系统来使用设备。

操作系统实际是通过管理设备的设备驱动程序来间接操控设备的。对于每个物理设备，厂家都会提供相应的驱动程序，所有与设备相关的操作代码都写在驱动程序中。同类设备如果系统可以屏蔽其差别，则可以共用同一个驱动程序。

使用设备前，必须安装设备驱动程序，例如打印机。当然，对于鼠标、键盘和显示器这些标

准设备，操作系统已提供了标准的驱动程序，用户不必自己安装。

在层次化存储体系环境下，程序是如何被 CPU 执行的？当用户请求执行一个程序后，操作系统的处理机管理（进程管理）、内存管理、文件管理等分工合作完成此任务。它体现了计算思维“采用抽象和分解的方法来控制庞杂的任务”的思维方法。因篇幅有限，这里不展开论述。有兴趣的读者请查阅《操作系统》课程相关资料。

1.6 计算思维方法的案例

在本节中，通过三个典型的案例，分析计算思维主要方法在实际中的应用，如通信、协作、抽象、分解、关注点分离、建模、计算效率。

1.6.1 从两军问题看通信与协作的思维

下面通过网络协议的一个经典问题：两军问题来了解通信和协作的作用。

【案例 1.3】通信与协作思维方法案例：两军问题。

一支白军被围困在一个山谷中，山谷的两侧是蓝军。困在山谷中的白军人数多于山谷两侧的任一支蓝军，而少于两支蓝军的总和。两支蓝军协同作战可战胜白军，单独攻击则会失败。两支蓝军的信息需派士兵经过白军驻扎的山谷传递，蓝军 1 拟于次日拂晓发起攻击，需通知蓝军 2。问：如何保证协同攻击能够成功发起并且取胜？

分析：本案例包括信息可靠传输和信息安全传输两个方面。**信息可靠传输**是指，采用一系列技术或方法来保障信息在发送方和接收方之间准确、精确的传输。**信息安全传输**是指，传送的信息内容不被别人知晓。在蓝军传递信息的过程中，信息可能丢失，因为在穿越山谷时，通讯兵有可能被俘，从而造成信息不能传送到对方那里，这就无法保证信息可靠传输。另一方面，蓝军传递的信息可能被白军截获，造成信息泄露，这就不能保证信息安全传输。那么应该如何通信，才能使蓝军必胜？

求解思路 1：采用两次握手协议传递信息，如图 1-38 所示。

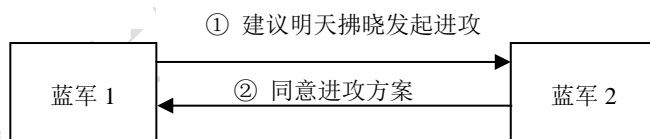


图 1-38 两军问题采用两次握手协议

① 蓝军 1 的指挥官发出消息：“我建议在明天拂晓发起进攻，请确认”，消息由通讯士兵带到了蓝军 2；

② 蓝军 2 的指挥官同意这一建议，并回信“同意进攻方案”，该回信由通讯士兵安全送到了蓝军 1 的指挥官处。

那么，联合进攻会发生吗？

不会！因为蓝军 2 的指挥官无法知道他的回信是否安全送到了，所以，他不能贸然发起进攻。

求解思路 2：既然两次握手协议不能获得该问题求解，则改进协议，将两次握手协议改为三次握手协议，如图 1-39 所示。

① 蓝军 1 的指挥官发出消息：“我建议在明天拂晓发起进攻，请确认”，消息由通讯士兵带到了蓝军 2；

② 蓝军 2 的指挥官同意这一建议，并回信“同意进攻方案”，该回信由通讯士兵安全送到了蓝军 1 的指挥官处；

③ 蓝军 1 指挥官收到蓝军 2 的确认消息，则他须告诉对方（蓝军 2 指挥官）“你的同意信息我已经

知晓了”，从而完成三次握手协议。

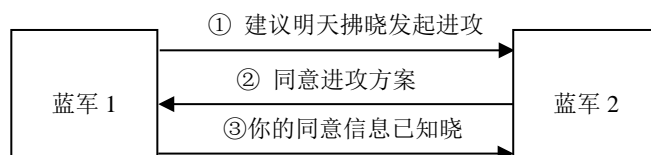


图 1-39 两军问题采用三次握手协议

这回联合进攻会发生吗？

还是不会！因为蓝军 1 指挥官无法知道他的确认信息“你的同意信息我已经知晓了”是否安全送到了蓝军 2，所以，他不能贸然发起进攻。可见三次握手协议还是不能保证协同攻击成功发起。

显然，在这种通信方式下不存在使蓝军必胜的通信约定（协议）。这是因为，永远存在一个需要确认的信息。因此，没有一个有效的经典协议可以使得蓝军获胜。

两军问题的抽象：

两军问题是现实生活中的一个关于信息传送的典型示例。随着信息技术的发展，两军问题在不断演化，传递信息的方式在不断变化，最初由通讯士兵传递消息，古代人们用烽火狼烟、飞鸽传书，二次工业革命之后人们发明了电话和无线电来方便地传递信息，直到当今的信息时代，人们普遍使用网络媒介来迅捷传递信息……

抽象之后的两军问题实际上就是研究如何使用非可靠的通信信道来传递可靠保密的信息，即密码学的**保密通信**：在军事通信上，使传送信息的载体（如文本、无线电报等）即使在被截获的情况下也不会让截获者得知其中信息内容的通信方法或技术。

如何解决两军问题？

可以从协作（Coordination）的角度来思考解决此问题。协作即是为确保多方参与的计算过程（如多人会话）最终能够得到确切的结论而对整个过程中各步骤序列先后顺序进行的时序控制。

对于两军问题，过去那种采用通讯士兵传递消息的方式是无解的。由于不能确定对方是否接收到消息，双方就会无休止地建立连接。联合进攻不会发起，蓝军不可能取胜！

但可以考虑采用其他通信方式，如飞鸽传书，同时采用古典加密方法如**代换密码**（将明文字母替换成其他字母、数字或符号）或**置换密码**（不改变明文的内容，而只是改变明文出现的次序，即重新排列消息中的字母）等对信息进行加密，将明文转换成密文，这样即使信息被敌人截获，也不会泄密。这里**明文**指人们要传送的以通用语言明确表达的文字内容。没有进行加密，能够直接代表原文含义的信息。**密文**是由明文经变换而形成的、用于密码通信的一系列看上去杂乱无章的符号。密文是经过加密处理之后，隐藏原文含义的信息。

或者采用电话传递信息，当然，为避免被窃听，最好使用隐语。

还可以采用无线电通信，同样，为避免被截获，最好使用加密技术。

在当今信息时代，完全可以采用网络通信技术来进行**密码通信**，即发送者把表达信息的意义明确的文字符号，用通信双方事先所约定的变换规则，变换为另一串看似毫无意义的符号，作为通信的文本发送给收信者；收信者根据事先约定的变换规则，把它恢复成原来意思的文字。这涉及到通信协议、加密算法和解密算法等。

网络通信示意图如图 1-40 所示。**网络通信**是将不同位置的两台或多台计算机连接起来，实现信息的发送、接收和转换。信息的发送者称为**信源**；信息的接收者称为**信宿**；传送信息的媒介称为**信道**（或载体）。

信源通过信道将信息传输到信宿。信道可以有线的，如利用各种电缆（双绞线、同轴电缆、光纤）进行传输；也可以是无线的（如无线电波、微波和红外线）。

为保障信息的机密性，不要在公共信道上以明文方式传输需要保密的信息。可以在发送端通过密钥

（**密钥**是密码算法中的控制参数）对明文加密；然后将密文在信道中传输，到达接收端后，再利用密钥对密文解密，得到明文。

信源（信息的发送者）

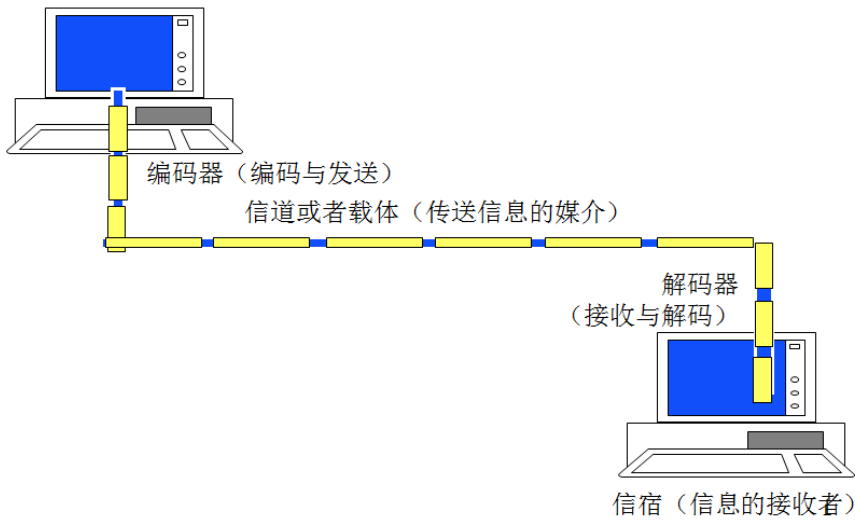


图 1-40 网络通信示意图

在现代作战中，可以利用网络通信解决两军问题。简单来说，就是蓝军 1 和蓝军 2 采用 TCP（Transmission Control Protocol，传输控制协议）协议，建立可靠的连接；然后蓝军 1 利用公共信道传送协同作战消息（加密传输）；随后释放连接；蓝军 2 将收到的协同作战消息进行解密，得到明文。同理，蓝军 2 采用相同步骤将确认消息发送给蓝军 1。

这里**协议**是指为使交流信息的双方能够正确实现信息交流而建立的一组规则、标准或约定。为进行网络中各节点和计算机之间的数据交换而建立的规则、标准或约定就称为网络协议。网络协议明确规定了所交换数据的格式以及有关的同步问题。

TCP 连接包括三个阶段：连接建立、数据传送和连接释放。在正式发送数据前，发送方必须与接收方建立可靠的连接。

一个 TCP 连接必须要经过三次“对话”（一般称为**握手**）才能建立起来。三次“对话”的目的是使数据包的发送和接收同步，经过三次“对话”之后，主机 A 才向主机 B 正式发送数据。用三次握手建立 TCP 连接其示意图如图 1-41 所示。

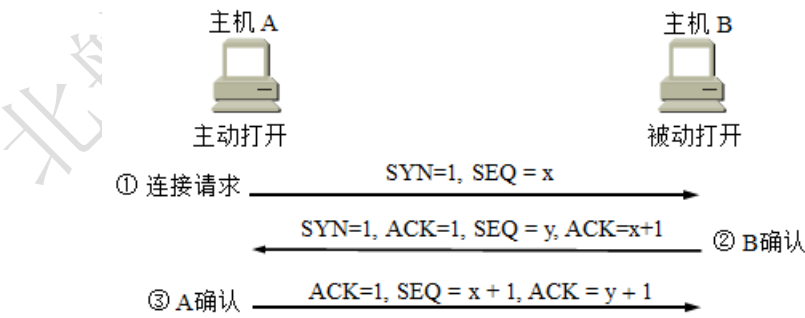


图 1-41 用三次握手建立 TCP 连接示意图

① A 发送连接请求。首先主机 A 的 TCP 向主机 B 发出连接请求报文段，其首部中的同步比特 SYN 应置为 1，并选择序号 $SEQ=x$ ，表明传送数据时的第一个数据字节的序号是 x 。

② B 确认收到此请求。当 B 的 TCP 收到连接请求报文段后，如同意，则发送确认报文。B 在确认报文段中应将同步比特 SYN 和确认比特 ACK 置为 1，其确认号 ACK 应为 $x+1$ ，同时也为

自己选择序号 $SEQ=y$ 。

③ A 确认收到了 B 的确认信息。A 收到此报文段后，向 B 给出确认，其确认号 ACK 应为 $y+1$ 。

同时，TCP 还专门设计了重传机制，即 TCP 每发送一个报文，就对该报文设置一次计时器，如果在计时器设置的超时重传时间内没有收到确认报文，就会自动重传该报文，多次重传失败会自动断开连接，并通知上层应用进程。以确保信息的可靠传输。


这样，在规定的时间内，A 收到了相应的三次握手的确认报文，A 的 TCP 就会通知上层应用进程：连接已经建立。同时，运行服务器进程的主机 B 的 TCP 收到主机 A 的确认后，也通知其上层应用进程：连接已经建立。

三次握手协议不仅可以保证协议双方的可靠通信，也避免了双方无休止地建立连接。完成三次握手后，客户端与服务器开始传送数据。当信息传送完毕，需要释放连接时，TCP 也需要互相确认才可以断开连接，采用四次握手释放一个连接。因篇幅有限，这里不做详细介绍。

1.6.2 计算机网络访问过程蕴含的计算思维

【案例 1.4】计算机网络访问过程蕴含的计算思维。

计算机网络非常复杂，包括许多“构件”，如主机、路由器、各种媒体的链路、应用、协议、硬件和软件等。如何将它们组织成一个完整的体系？从一个客户端到服务端的访问是如何完成的？分析在计算机网络及其模型中，蕴含了哪些计算思维？

 **提示：**客户端（Client）或称为用户端，是指与服务端相对应，为客户提供本地服务的程序。一般

安装在普通的客户机上，需要与服务端互相配合运行。常用的客户端如万维网使用的网页浏览器，收寄电子邮件时的电子邮件客户端，以及即时通讯的客户端软件等。对于这一类应用程序，需要网络中有服务器和服务程序（称为服务端）来提供相应的服务，如数据库服务，电子邮件服务等。这样在客户端和服务端，需要建立特定的通信连接，来保证应用程序的正常运行。

服务端（Server）是一种有针对性的服务程序，是为客户端服务的，如向客户端提供资源，保存客户端数据等。**服务器**是指网络中能对其它机器提供某些服务的计算机系统，在其上安装了服务端程序。

1、OSI 参考模型与 TCP/IP 参考模型

（1）计算机网络体系结构

计算机网络是以共享资源（硬件、软件和数据）为目的，利用某种传输媒介，将不同地点的独立自治计算机系统或外部设备连接起来形成的系统。

为了完成计算机间的通信合作，按照计算机互连的不同功能，将计算机网络划分成定义明确的若干个层次，并规定同层次进程通信的协议及相邻层之间的接口和服务。这样的层次结构模型和通信协议统称为网络体系结构。

计算机网络采用这种层次化结构具有如下优点：

① **各层之间功能相互独立。**高层不需要知道低层是如何实现的，而仅知道该层通过层间的接口所提供的服务即可。

② **提供标准化接口，灵活性好。**只要层间接口不变，则某层的改变不会影响其上下层。

③ **采用模块化结构。**结构上可分割开，各层都可以采用最合适的技术来实现，系统易于维护。

（2）OSI 参考模型

从 20 世纪 70 年代起，许多著名的计算机公司都建立了自己的网络协议和网络体系结构，但它们都属于各公司专用，彼此不兼容。由于缺乏统一的标准，各计算机公司的计算机网络之间很难进行通信。

为使不同计算机厂家的计算机能够相互通信，进而在更大的范围内建立计算机网络，有必要建立一

个国际范围的网络体系结构标准。1985 年，为实现开放系统环境中的互连性、互操作性和应用的可移植性，ISO（International Standard Organization，国际标准化组织）推出了一个标准的网络体系结构，被称为**开放系统互联（Open System Interconnection）模型**，简称 **OSI 参考模型**。

OSI 参考模型采用分层结构技术，定义了网络互联的 7 层框架，详细规定了每一层的功能，定义了每一层的协议（协议是某层同远方一个对等层通信所使用的一套规则和约定）。每一层使用与之相邻的下层所提供的服务，并向相邻上层提供一套功能更强大的服务。每一层实体为相邻的上一层实体提供的通信功能称为服务。用于实现某层功能的活动元素称为**实体**。

OSI 参考模型划分的 7 个层次由低到高依次为：物理层（Physical Layer）、数据链路层（Data Link Layer）、网络层（Network Layer）、传输层（Transport Layer）、会话层（Session Layer）、表示层（Presentation Layer）、应用层（Application Layer），各层之间功能相对独立。如图 1- 42 所示。

上三层（应用层、表示层、会话层）统称为**应用层**，主要与网络应用相关，负责对应用程序提供接口和网络服务、对用户数据进行编码以及管理会话等操作。下四层（传输层、网络层、数据链路层和物理层）统称为**数据流层**，主要负责网络通信，负责将用户的数据传递到目的地。

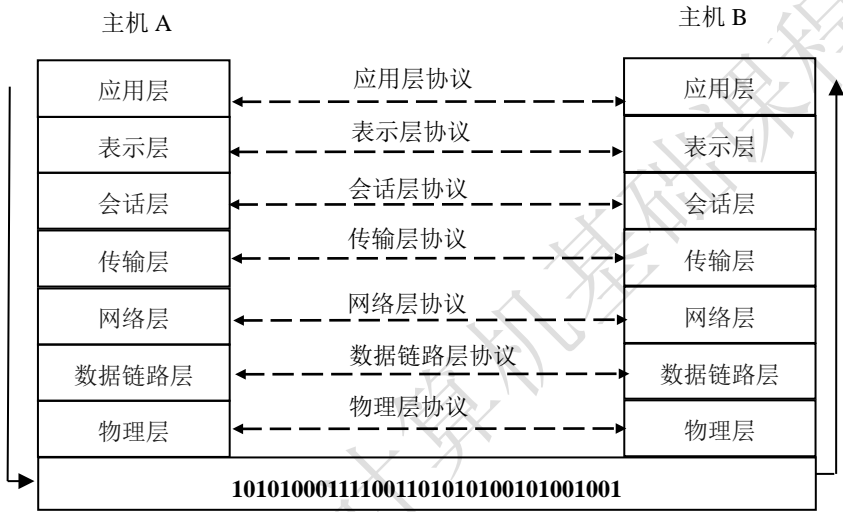


图 1- 42 OSI 开放系统互联参考模型

每一层遵守一定的协议。OSI 高层协议（应用层协议、表示层协议和会话层协议）是由软件实现的，面向应用，面向用户；低层协议（传输层协议、网络层协议、数据链路层协议、物理层协议）一般由硬件实现，面向物理信号的传输与处理。

不同主机之间的相同层次被称为**对等层（Peer）**。对等层之间存在协议关系。即对等实体之间互相通信需要遵守一定的规则，如通信的内容、通信的方式等。这种对等实体之间交换数据或通信时必须遵守的规则称为**对等层协议（Peer Protocol）**。除了物理层外，OSI 参考模型中的其他 6 个对等层都存在相应的对等层协议，如数据链路层协议、网络层协议等。对等实体利用对等层协议进行通信，从而向高层提供服务。

① OSI 参考模型第一层：物理层

主要功能是利用物理传输介质为数据链路层提供物理连接，以便透明地传送比特流。数据传输单位为**比特（bit）**。该层描述网络硬件接口的电气、机械、功能、过程等特性，如：电压、物理数据速率、最大传输距离、物理连接器和其他的类似特性。

② OSI 参考模型第二层：数据链路层

主要解决在物理连接的基础上如何保证两个相邻节点之间数据可靠传输的问题。数据传输单位为**帧（Frame）**。主要功能是在通信的实体之间建立数据链路连接，传送数据；为网络层提供差错控制、流量控制服务，使有差错的物理线路变成无差错的数据链路。

③ OSI 参考模型第三层：网络层

主要解决寻找信息接收方、选择最佳传输路径和建立维护网络连接的问题。数据传输单位为**分组（包，**

Packet)。主要任务是通过路由选择算法，为分组选择最佳路径；实现分组的分片与重组以及网络互连等功能。

④ OSI 参考模型第四层：传输层

主要为用户提供 End—to—End（端到端）服务，处理数据报错误、数据包次序等传输问题，为上一层提供可靠或不可靠的传输服务。数据传输单位为段（Segment）。传输层向高层屏蔽了下层网络数据通信的细节，是计算机网络体系结构中承上启下的关键一层。

⑤ OSI 参考模型第五层：会话层

主要任务是建立、维护和终止两个远程系统进程之间的连接会话，并管理数据的交换。会话层通过对话控制来决定使用全双工通信或半双工通信方式。会话层通过会话层协议对请求与应答进行协调。

⑥ OSI 参考模型第六层：表示层

主要任务是处理两个通信系统中交换信息的表示方式，包括数据格式变换、数据加密与解密、数据压缩与解压等。表示层为应用层提供的服务包括语法转换、语法选择和联接管理。

⑦ OSI 参考模型第七层：应用层

应用层是 OSI 模型中的最高层，直接面向用户。应用层提供了许多应用层的协议，程序开发者基于这些协议可以开发出各种应用程序。例如 IE 浏览器，使用的是应用层的 HTTP 协议。应用层直接为应用程序提供网络资源的访问服务，如：远程登录、电子邮件、文件传输等。

OSI 模型精确定义了开放系统互联的框架，便于有关标准和协议在框架内开发和相互配合，不会因为以后技术的发展而必须重新作出修改。

OSI 参考模型的这种层次化结构具有如下优点：

- ① 开放的标准化接口。有利于应用程序开发和网络设备的设计。
- ② 多厂商使用统一结构，不同厂商的同类产品互相兼容。便于组建更大的计算机网络，便于网络之间的互相通信。
- ③ 易于理解、学习和更新协议标准。当更新某一层的协议时，不会干扰到其他层。
- ④ 模块化设计、流程化操作。程序员开发程序时，思路更清晰，编程更容易。
- ⑤ 网络分层。就像一个筛子，能快速定位故障发生的地方，更便于故障排除。

（3）TCP/IP 参考模型

1969 年，美国国防部高级研究计划署 ARPA(Advanced Research Project Agency)出于军事研究目的，开发出 ARPANET（阿帕网）——世界上第一个运营的封包交换网络，全球互联网的始祖。ARPANET 在技术上的一个重大贡献是 TCP/IP（Transmission Control Protocol/Internet Protocol，传输控制协议/因特网互联协议）协议簇的开发和利用。

在 ARPANET 基础上发展起来的 Internet（因特网，又称国际互联网）并没有使用 OSI 模型来定义网络体系结构，而是使用了 TCP/IP 协议簇。随着 Internet 规模的不断扩大，网络厂商纷纷支持 TCP/IP 协议，使得 TCP/IP 协议簇成为了“事实上的国际标准”，被称为 **TCP/IP 参考模型**。

TCP/IP 参考模型最多将网络结构划分为 5 个层次，分别是物理层、数据链路层、网际层（对应 OSI 参考模型的网络层，也称为网络互连层或网络层）、传输层和应用层。它将所有与应用程序相关的内容都归为一层，即把 OSI 参考模型中的高三层（会话层、表示层和应用层）合并为一层，统称应用层，主要处理高层协议、有关表达、编码和会话控制。

有些情况下，TCP/IP 参考模型只有 4 个层次，它将物理层、数据链路层统称为网络接口层，整个体系结构包括网络接口层、网际层、传输层和应用层。

OSI 参考模型与 TCP/IP 参考模型的体系结构如图 1- 43 所示。可以看出，TCP/IP 参考模型比 OSI 参考模型的结构更简化。

网络接口层：也称**网络接入层**或**主机到网络层**（Host-to-Network）。与 OSI 参考模型中的物理层和数据链路层相对应。主要负责监视数据在主机与网络之间的交换。事实上，TCP/IP 本身并未定义该层的

协议，而是允许参与互连的各个网络使用自己的物理层和数据链路层协议，然后与 TCP/IP 的网络接口层进行连接。

网际层：对应于 OSI 参考模型的网络层。主要运送数据包，将数据从任何相连的网络上送到目的地，最佳路径选定和数据包交换都发生在这层。其最重要的协议是 IP（Internet Protocol，因特网互联协议）协议，它为每台计算机规定了唯一的 Internet 地址，并提供一个不可靠、无连接的数据报传递服务。此外还有 ARP（Address Resolution Protocol，地址转换协议），RARP（Reverse ARP，反向地址转换协议），ICMP（Internet Control Message Protocol，控制报文协议）等协议。

传输层：对应于 OSI 参考模型的传输层。为应用层实体提供端到端的通信功能，保证数据包的顺序传送及数据的完整性。该层定义了两个主要的协议：TCP 协议、UDP（User Datagram Protocol，用户数据报协议）。TCP 协议提供一种可靠的、面向连接的数据传输服务；而 UDP 协议则提供的是不可靠的、无连接的数据传输服务。

应用层：对应于 OSI 参考模型的高层。为用户提供所需要的各种服务，例如：SMTP（Simple Mail Transfer Protocol，简单邮件传输协议）、FTP（File Transfer Protocol，文件传输协议）、HTTP（Hyper Text Transfer Protocol，超文本传输协议）、DNS（Domain Name System，域名解析系统）等。

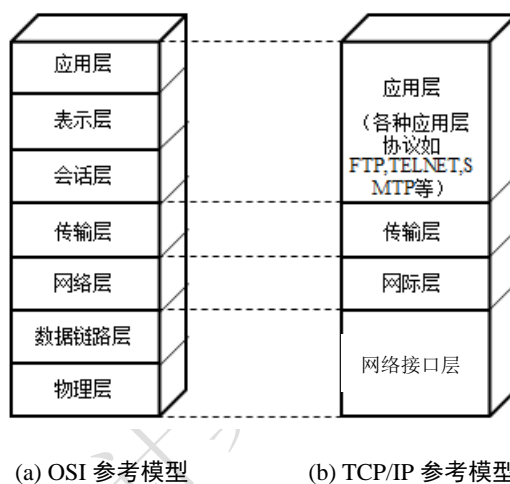


图 1-43 OSI 参考模型与 TCP/IP 参考模型的体系结构

注意：尽管 TCP/IP 字面上代表了两个协议：TCP（传输控制协议）和 IP（因特网互联协议或网际协议），但 TCP/IP 协议并不是 TCP 和 IP 这两个协议的合称，而是一组协议的集合，称为 TCP/IP 协议簇。其中每种协议负责网络传输中的一部分服务。

TCP/IP 参考模型与 OSI 参考模型各自层与层之间关系相似，也都是对等的层间通信。其功能与 OSI 参考模型大致相同，在两个模型中，传输层及以上的各层都是为通信的进程提供点到点、与网络无关的传输服务；但 TCP/IP 参考模型比 OSI 参考模型提供更好的网络管理功能。

TCP/IP 参考模型的数据传输原理也与 OSI 参考模型相同。

（4）OSI 参考模型与 TCP/IP 参考模型比较

OSI 参考模型与 TCP/IP 参考模型的共同之处在于：

- ① 采用分层体系结构，将庞大而复杂的问题转化为若干个较小且易于处理的子问题。
- ② 分工合作，责任明确。将性质相似的工作划分在同一层，性质相异的工作则划分到不同层。当两个主机间通信出现问题时，易于查找故障所在。
- ③ 基于独立的协议栈的概念。每层定义有相应的通信协议，不同主机之间的相同层次依据该层的协议交换数据或通信。
- ④ 逐层处理，层层负责。当第 N 层收到从第 N-1 层传送的数据时，按照本层定义的功能对数据进

行处理，再将数据向上传送给第 N+1 层；如果第 N 层收到第 N+1 层传下来的数据，也是对数据进行相应处理，完成后才向下传给第 N-1 层。每一层只负责完成该层应该完成的工作。

OSI 参考模型与 TCP/IP 参考模型的主要区别在于：

① OSI 参考模型与 TCP/IP 参考模型的制定初衷不同

OSI 参考模型有 3 个主要明确概念：服务、接口、协议，这是 OSI 参考模型最大的贡献。而 TCP/IP 参考模型在最初提出时并没有明确区分这三者。

② 构建模型与制定协议的顺序不同

OSI 参考模型先提出层次化的模型概念，再在各层内补充协议。而 TCP/IP 型是先制定协议，再构建层次化结构。

③ 在是否考虑多种异构网的互联问题上不同

OSI 是理想化的网络模型结构，最初只考虑到使用一种标准的公用数据网将各种不同的系统互联在一起，但并没有考虑异构网的互联问题。TCP/IP 参考模型一开始就考虑到多种异构网的互联问题，并将网际协议 IP 作为 TCP/IP 的重要组成部分。

由于 ISO 在制定 OSI 参考模型过程中总是着眼于通信模型所必需的功能，理想化地期待政府行为来统一各种网络协议，所以在制定过程中忽略了互联网协议的重要性。当考虑到这一点时，却由于功能复杂难以实现等原因，失去了市场。

而 TCP/IP 参考模型在现有的协议基础上，充分考虑了“将协议实际安装到计算机中如何进行编程最好”这个涉及实际应用的问题，使得模型容易实现，得到了广大用户和各大厂商的支持。同时，由于它一开始就着眼于通用连接，使得 TCP/IP 参考模型及其协议，可以在任何互连的网络集合中进行通信。因此，TCP/IP 参考模型得到了迅速发展和应用，成为事实上的网络体系结构标准。

尽管 OSI 参考模型是对发生在网络设备间的信息传输过程的一种理论化的描述，它仅仅是一种模型，并没有定义如何通过硬件和软件实现每一层功能，但是，它可以有效地帮助我们理解网络的层次化结构以及数据传输的过程。

2、万维网及其工作原理

当人们需要从浩瀚的 Internet 上查找某个信息时，Internet 是如何迅速返回人们想要的结果的呢？这里以万维网（World Wide Web，WWW）为例，简单分析从一个客户端到服务端的访问是如何完成的。

（1）万维网相关概念

万维网即环球信息网，又称为环球网，常简称为 **Web**。Web 是 Internet（因特网）发展的产物，它是 Internet 所提供的众多服务中的一个。它实质上是一个基于超文本的文件信息服务系统，用户可以通过浏览器搜索和浏览文字、图片、声音和视频等信息；可以在线播放音频或视频文件；还可以在上面发布信息、发表观点；或查找去往某个地点的最佳路线；或进行电子购物等。

Internet 即国际互联网，它是一个世界范围的网络，为人们提供了便捷的通信和丰富的资源。常用的 Internet 服务包括远程登录、文件传输、电子邮件、万维网、即时通讯和电子商务等。

“超文本”是超级文本的中文缩写。超文本是用超链接的方法，将各种不同空间的文字信息组织在一起的网状文本。超文本由若干个互连的文本块组成，这些文本块被称为**节点（node）**。超文本普遍以电子文档方式存在，其中的文字包含有可以链接到其他位置或者文档的指针，允许从当前阅读位置直接切换到超文本连结所指向的位置。因此，与普通文本不同，超文本不仅包含自身的信息文本，还包含指向其他文档的链接。超文本打破了传统的文本线性阅读方式，人们可以在任何一个节点上停下来，通过点击链接进入另一重文本，然后再点击链接、进入又一重文本……。从而，原先的单一的文本变成了无限延伸和扩展的超级文本、立体文本。

事实上，随着多媒体技术的发展，构成超文本的节点除文本外还有图形、图像、音频和视频文件等，此时的超文本便称为“**超媒体**”。

超链接是超级链接的简称。超链接是某超文本文档中的元素，与另一个超文本、文件或脚本的不同

元素之间的连接。用户可以通过点击被链接的元素来激活这些链接，通常在被链接的元素下使用下划线或者以不同的颜色（如蓝色）显示来表示一个可以被执行的链接。当移动鼠标到超链接时，鼠标会变成一只小手的形状。

例如，在浏览器中查找“颐和园”，会弹出有关颐和园的介绍文档，如图 1-44 所示。其中有许多蓝色的文字，这就是超链接。单击这样的文字，可以迅速跳转到该文字所指向的另一个文档。比如，单击“乾隆皇帝”，则弹出关于“爱新觉罗·弘历”的介绍，在这篇超文本中，又包含了许多超链接，读者可以单击任意一个超链接，从而去阅读自己希望了解的其他内容。



图 1-44 超文本与超链接示意

(2) Web 的组成

Web 是 Internet 上的分布式多媒体信息系统，整个系统由 Web 客户端 (Client) 和 Web 服务器 (Web Server) 组成。Internet 上有许多主机向 Web 提供信息，这些主机称为 Web 站点 (Web Site)，在其上运行着 Web 服务器程序。用户在客户机上使用 Web 浏览器访问浏览 Web 服务器上的页面，以检索、阅读和下载 Web 信息。

Web 客户端主要指 Web 浏览器 (Browser)，它安装在客户机上，将用户向服务器请求的 Web 资源呈现出来，显示在浏览器窗口中。浏览器是一种应用软件，用于在电脑上显示通过万维网或局域网传送的文字、图像及其他信息，以及用户与这些信息的交互操作。

主流的浏览器包括 Microsoft (微软) 公司的 IE (Internet Explorer)、Mozilla 公司的 Firefox、Apple

(苹果)公司的 Safari、Google(谷歌)公司的 Chrome 以及挪威 Opera Software ASA 公司的 Opera 等几大类。

Web 服务器是一台与 Internet 相连,执行传送网页和其他相关文件(如与网页相连的图像文件)的计算机,主要为客户端提供网上信息浏览服务。

Web 服务器也指驻留于因特网上某种类型计算机的程序,其主要功能有:响应浏览器的请求,向 Web 客户端提供文档或其它资源;存放网站文件,供身处世界各地的用户浏览;存放数据文件,供任何人下载;执行 Web 服务器端程序,即利用 CGI、ASP、PHP 和 JSP 等动态网页技术语言编写的 Web 服务器端运行程序。

目前最主流的三个 Web 服务器是 Apache 软件基金会的 Apache(阿帕奇)、伊戈尔·赛索耶夫开发的 Nginx(engine x)、Microsoft 公司的 IIS(Internet Information Services,互联网信息服务)。在 UNIX 和 LINUX 平台下使用最广泛的免费 HTTP 服务器是 Apache 和 Nginx 服务器,而 Windows 平台下使用 IIS 服务器。

Apache 是世界使用排名第一的 Web 服务器软件,其市场占有率达 60%左右。其优势在于源代码开放,支持跨平台的应用(可以运行在几乎所有的 Unix、Windows、Linux 系统平台上)以及可移植性强等方面。

Nginx 是一个高性能、轻量级的 HTTP/反向代理 Web 服务器及电子邮件(IMAP/POP3/SMTP)代理服务器,是由伊戈尔·赛索耶夫为俄罗斯访问量第二的 Rambler.ru 站点开发的。其特点是占有内存少,并发能力强。我国使用 Nginx 网站的著名企业有:百度、京东、新浪、网易、腾讯、淘宝等。

IIS 是一种 Web 服务组件,它包括 Web 服务器、FTP 服务器、NNTP 服务器和 SMTP 服务器,分别用于网页浏览、文件传输、新闻服务和邮件发送等方面,使得在网络(包括互联网和局域网)上发布信息成为一件很容易的事。IIS 提供了一个图形界面的管理工具——Internet 服务管理器,可用于监视配置和控制 Internet 服务。

(3) Web 工作原理

万维网要提供因特网范围内的信息浏览功能,必须解决以下 4 个问题:

① 如何标志分布在因特网上的 Web 文档?

使用统一资源定位器(Universal Resource Locator, URL)来标志万维网上的各种文档。使每一个文档在整个因特网的范围内具有唯一的标识符 URL。

URL 由三部分组成:协议、欲访问机器的 IP 地址或域名、在该机器下的目录及文件名。其格式如下:

<协议>://<主机>[:<端口>]/<路径>

其中,“**协议**”指 Web 服务器提供的 Internet 服务对应的协议。利用 Web 浏览器可以访问大多数的 Internet 服务,如文件传输 FTP(File Transfer Protocol)、电子邮件 E-mail、远程访问 Telnet、Web 服务等,在使用这些服务时需要使用相应的协议。“**主机**”指存放资源的主机在因特网中的 IP 地址或域名。“**端口**”指 HTTP 的端口号,默认是 80,通常可省略。“**路径**”指要访问的资源在 Web 服务器中的具体路径(目录及文件名),若省略文件的<路径>项,则 URL 就指到因特网上欲访问网站的主页(home page)。

IP 地址是给每一个连接在 Internet 上的主机所分配的一个唯一的 32 位二进制数的地址。每个地址由网络号和主机号组成,表明接入的是哪个网络的哪个主机。一般将 IP 地址书写成用“.”分隔的四个十进制数。但即便如此,数字还是难以记忆,因此,人们往往使用以“.”分隔的一系列字母来表示 IP 地址,这种表示计算机 IP 地址的符号名字被称为**域名**。

例如,百度搜索引擎的 Web 服务器的 IP 地址是 61.135.169.125,其域名是 www.baidu.com,则百度搜索引擎网站的主页的 URL 为 <http://61.135.169.125/>或 <http://www.baidu.com/>。http 表示采用的是 HTTP 协议。https 表示采用的是 HTTPS(Hyper Text Transfer Protocol over SecureSocket Layer,超文本传输安全协议)协议。

 **提示:**HTTPS 协议是由 HTTP 协议加上 TLS(Transport Layer Security,传输层安全性)和 SSL(Secure

Sockets Layer, 安全套接层) 协议所构建的可以进行加密传输、身份认证的网络协议, 主要通过数字证书、加密算法、非对称密钥等技术完成互联网数据传输加密, 实现互联网传输安全保护。它克服了 HTTP 协议由于数据的明文传送和缺乏消息完整性检测所造成的安全缺陷。

若要使用域名表示 IP 地址, 则需要有一个**域名系统** (Domain Name System, DNS) 将域名转换成计算机的 IP 地址。

② Web 客户端与 Web 服务器之间采用何种协议进行多媒体通信?

Web 客户端 (浏览器) 与 Web 服务器之间使用 HTTP 协议进行通信。HTTP 是一个应用层协议, 它是属于浏览器与 Web 服务器之间的通信协议, 建立在 TCP/IP 基础之上, 用于传输浏览器到服务器之间的 HTTP 请求和响应。它不仅需要保证传输网络文档的正确性, 同时还需要确定文档显示的先后顺序。

③ 怎样使各种 Web 文档都能在因特网上的各种计算机上显示出来, 同时使用户清楚地知道在什么地方存在着超链接?

在 Web 服务器上, 主要以网页 (Web Page) 的形式来发布多媒体信息。网页是采用超文本标记语言 HTML (Hyper Text Markup Language) 编写的。一个网页就是一个 HTML 文件, 它由两部分组成: 标记和文本。**文本**是指文件本身的内容, 为纯文本信息, 或者为图像、声音文件等; 而**标记**用于指明文件内容的性质、格式和链接等。为与文本区分, 标记是用 <> 尖括号括起来的, 如 <HEAD>、<HTML>、<BODY> 等为标记。

网页的设计者采用 HTML, 可以很方便地用一个超链接从本页面的某个位置链接到因特网上的任何一个 Web 页面。当浏览器软件连接到 Web 服务器并获取所链接的网页后, 通过对网页 HTML 文档的解释执行, 将该网页所包含的信息显示在用户的计算机屏幕上。

④ 如何使用户方便、迅速地找到所需的信息?

为了在万维网上方便迅速地查找信息, 用户可使用各种搜索工具 (即搜索引擎)。搜索引擎是根据一定的策略、运用特定的计算机程序从互联网上采集信息, 并对信息进行组织和处理, 从而为用户提供检索服务, 将检索的相关信息展示给用户的系统。搜索引擎依托多种技术, 如网络爬虫技术、检索排序技术、网页处理技术、大数据处理技术、自然语言处理技术等, 为信息检索用户提供快速、高相关性的信息服务。

目前常用的搜索引擎主要有:

百度 (www.baidu.com), 百度公司研发, 全球最大的中文搜索引擎。


搜狗搜索 (www.sogou.com), 2004 年由搜狐公司推出的第三代互动式搜索引擎, 随着 2013 年腾讯公司的搜索引擎 SOSO 的并入, 搜狗搜索成为中国第二大搜索引擎。

必应 (Bing) (cn.bing.com), 微软 (Microsoft) 公司于 2009 年 5 月推出的全新搜索引擎服务。

有道 (www.youdao.com)。网易公司自主研发的全新中文搜索引擎, 2007 年 12 月推出正式版。

360 搜索 (www.so.com), 全文搜索引擎, 奇虎 360 公司开发的基于机器学习技术的第三代搜索引擎, 具备“自学习、自进化”能力, 能够发现用户最需要的搜索结果。为 360 浏览器的默认搜索引擎。

Google (www.google.com), 谷歌 (Google) 公司研发, 全球最大的搜索引擎。但已退出中国, 国内用户无法访问。

 **提示:** 浏览器与搜索引擎有何区别? 浏览器是一个专门的程序软件, 需要安装在用户的电脑上。它

是用户访问因特网的入口, 通过这个程序可以链接因特网, 从而使用户可以浏览网页, 获取服务器上的资料。

而搜索引擎是在因特网上执行信息搜索的一种专用工具, 它通过一定的算法, 对因特网内的众多网页进行分类和检索。搜索引擎是一个网站, 用户无需将其安装到电脑上, 只需在浏览器的地址栏中键入搜索引擎的网址, 就可以访问该搜索引擎, 进而搜索自己希望得到的信息或资料。现在的浏览器除了提供地址栏用于输入网站的地址、分门别类地提供各种网页的链接以外, 大多会提供一个搜索框。这个搜索框是搜索引擎与浏览器相结合的产物, 它使得用户可以方便地

在浏览器上直接使用搜索引擎。

假如现在希望了解北京航空航天大学计算机学院的学生培养和科学研究等各方面情况，应该怎么做呢？一般是先登录一个浏览器，比如谷歌公司的 Chrome 浏览器。然后在地址栏中输入北京航空航天大学官网的网址（<https://www.buaa.edu.cn/>），回车，则屏幕上会显示北京航空航天大学官网的主页。单击【机构设置】栏目下的【院系设置】，则进入“院系设置”页面；将鼠标移至其中的“计算机学院”，当鼠标形状变成小手时，单击该超链接，则进入“北京航空航天大学计算机学院”网页，其中包括首页、学院简介、本科生培养、研究生培养、师资队伍、科学研究等栏目，用户可以根据需要，选择相应的栏目进行浏览。

下面介绍，当用户在浏览器中单击某个网页中的超链接（如院系设置-计算机学院），万维网是如何找到相应网页并返回给客户端的。整个工作过程分为 4 个步骤：建立连接，浏览器发出 HTTP 请求，Web 服务器应答，关闭连接。Web 工作原理示意图如图 1-45 所示。

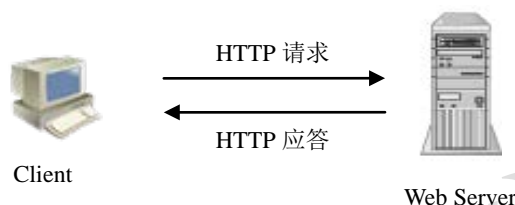


图 1-45 Web 工作原理

第一步：建立连接。

Web 服务器主机中运行着的服务器进程，始终等待客户端的连接请求。

当用户在“院系设置”页面中点击超链接“计算机学院”时，浏览器分析该超链接所指向页面的 URL（<http://scse.buaa.edu.cn/>）。浏览器向 DNS 请求解析域名（scse.buaa.edu.cn）的 IP 地址。DNS 解析出 Web 服务器的 IP 地址。浏览器与 Web 服务器建立 TCP 连接。

第二步：浏览器发出 HTTP 请求。

Web 浏览器向要访问资源的 Web 服务器（scse.buaa.edu.cn）发送包含网页地址的 HTTP 请求。

第三步：Web 服务器应答。

Web 服务器接收到 HTTP 请求后，按照网页地址寻找所请求的 Web 页面。如果找到就将所请求的网页（.htm 文件）作为应答回送给 Web 浏览器。

Web 浏览器接收到所请求的网页文件后，解释网页中所含的格式命令，并按照格式将网页元素正确显示在用户屏幕上，这时用户就看到了北航计算机学院主页的整个页面。

第四步：关闭连接。

TCP 连接释放，Web 服务器与浏览器之间断开连接。

3、数据在各层之间的传输过程

一般网络体系结构实际采用的是 TCP/IP 参考模型。那么，当用户浏览信息时，数据在各层之间是怎样传输的呢？

（1）协议数据单元与数据封装等相关概念

在了解数据传输过程之前，先介绍几个重要概念：协议数据单元（Protocol Data Unit, PDU），封装（Encapsulation），数据封装，数据解封装。

① 协议数据单元

在 OSI 和 TCP/IP 参考模型中，对等实体间按协议进行通信，上下层实体间按服务进行通信。对等实体之间为实现该层协议所交换的信息单元，称为**协议数据单元**。某一层的实体可能根据该层协议，将需要传送的数据划分为多个 PDU。PDU 包括控制信息和用户数据。比如，应用层的 PDU 包括应用层的控制信息和用户原始数据。

通常在某层的 PDU 前面增加一个代表该层的单字母的前缀，来表示是哪一层数据。例如，OSI 参考模型的会话层通过传送 SPDU 与对等的会话层进行通信。

应用层数据称为**应用层协议数据单元**（Application Protocol Data Unit, APDU），表示层数据称为**表示层协议数据单元**（Presentation Protocol Data Unit, PPDU），会话层数据称为**会话层协议数据单元**（Session Protocol Data Unit, SPDU）。

通常人们把传输层数据称为**段**（Segment），网络层数据称为**包**（Packet），数据链路层数据称为**帧**（Frame），物理层数据称为**比特流**（Bit）。

② 数据封装

封装（Encapsulation）是一个计算机术语。它指隐藏对象的属性和实现的细节，仅对外公开接口，控制在程序中属性的读和修改的访问级别。封装的原则是，把尽可能多的东西藏起来，只对外提供简捷的接口。

数据封装：网络通信中，在发送端，数据需要从高层一层一层地向下传送。将某层控制信息（称为报头或首部）添加到该层一个 PDU 的过程称作**数据封装**。当某个层收到 PDU 时，它为该 PDU 添加一个头或尾（即该层控制信息），并将封装后的 PDU 传送到下一层。添加到 PDU 上的控制信息将被远端设备的相同层所解读。

OSI 和 TCP/IP 参考模型的每一层都对数据进行封装，以保证数据能够正确无误地到达目的地，被终端主理解，执行。

③ 数据解封装

数据解封装是数据封装的逆过程。在接收端，数据需要从最低层一层一层地向上传送，从物理层到数据链路层，逐层去掉各层的报文头部，最终将数据传递给应用程序执行。将控制信息从 PDU 剥离的过程称作**数据解封装**。

（2）TCP/IP 参考模型中的数据传输

TCP/IP 参考模型中数据在各层之间的传输过程如图 1-46 所示。

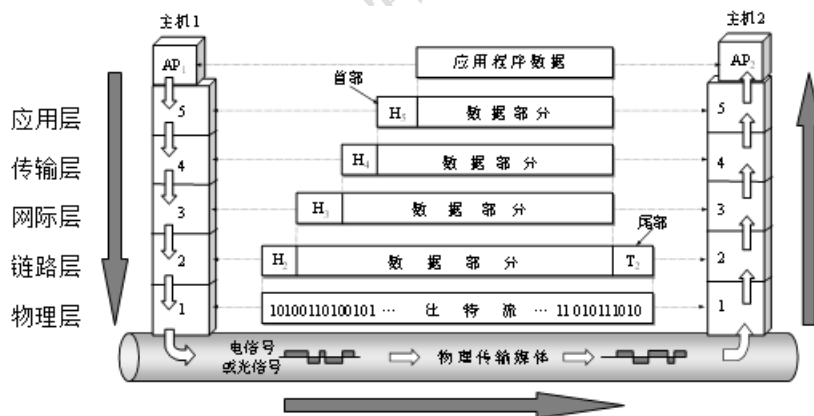


图 1-46 TCP/IP 参考模型中数据在各层之间的传输过程

发送方：

当主机 1 需要传送用户的数据（DATA）时，数据首先通过应用层的接口进入应用层。在应用层，用户的数据被加上应用层的报头 H₅（也称**首部**），形成应用层协议数据单元 APDU，然后通过应用层与传输层的接口数据单元，递交到传输层。

传输层并不“关心”应用层的数据格式，而是把整个应用层递交的数据报看作一个整体进行封装，即加上传输层的报头 H₄，然后递交到网际层。同样，网际层、数据链路层也都要分别给上层递交下来的数据加上自己的报头，即网际层报头 H₃ 和数据链路层报头 H₂。其中，数据链路层还要给网际层递交的数据加上数据链路层报尾 T₂，数据变成了由各层控制信息（H₂、H₃、H₄、H₅）加应用数据组成的数据，形成最终的一帧数据。

从应用层开始，将用户数据由高层向低层逐层传递，每经过一层，加上该层的控制信息（称为**报头**或**首部**），直到最低层（物理层），然后直接通过物理传输媒体（如双绞线电缆、同轴电缆或光纤）传输到目的方。可见在客户端，数据传输的过程是逐层封装的过程，简单来说，就是在不同的层次对数据打上相应的标识。

接收方：

主机 2 将收到的数据由最低层（物理层）向高层逐层传递，每经过一层，去掉该层的控制信息，直到最高层（应用层），恢复为用户数据。例如，数据比特流从物理层上传至链路层时，链路层依据链路层协议，识别报头和报尾，并将它们去掉，再上传给网际层。

可见在服务端，数据传输的过程是逐层解封装的过程，简单来说，就是在不同的层次对数据去掉相应的标识。

每层传递的数据分为首部字段和有效载荷字段两部分。有效载荷是相邻上层传下来的数据。

4、计算机网络及其模型蕴含的计算思维

计算机网络采用层次化结构，蕴含了抽象、分解、关注点分离和建模等计算思维方法。

周以真教授指出，计算思维是一种采用抽象和分解来控制庞杂的任务或进行巨大复杂系统设计的方法，是基于关注点分离的方法。计算思维是一种选择合适的方式去陈述一个问题，或对一个问题的相关方面建模使其易于处理的思维方法。

（1）抽象

在设计计算机网络方案时，人们通常把计算机等物理设备抽象成点，通信介质抽象成线，这样计算机网络的连接结构就变成了由点和线组成的几何图形，这种几何图形就是网络拓扑结构。抽象以后，无论多么复杂的网络，都可以清晰地用拓扑结构来描述。计算机网络中的通信则抽象为对数据的处理和相应的协议。

（2）分解

实现计算机网络通信需要解决如何表示信息和发送信息、如何确定信息接收者、如何保证信息的正确传输和识别等诸多复杂的问题。

对于一个复杂而庞大的系统，采用分层结构，将系统进行功能的划分，每一层仅实现一种相对独立、明确且简单的功能；一个难以处理的复杂问题通过多层次的分解，最终可转换为容易处理的问题从而得到解决，这是进行复杂系统设计的一种重要方法。

计算机网络体系结构采用了分解的方法，将复杂的网络转化为若干功能相对独立的层，以及层间的交互，从而使得系统层次清晰，易于实现，便于管理、维护和查找故障。

（3）关注点分离

关注点分离（Separation of Concerns, SoC）是日常生活和生产中广泛使用的解决复杂问题的一种系统思维方法。其思路是，先将复杂问题做合理的分解，再分别仔细研究问题的不同侧面（关注点），最后综合各方面的结果，合成整体的解决方案。在计算思维中，SoC 是极为重要的方法。

计算机网络体系结构就采用了 SoC 方法，它将网络分成不同层次，每个层次只关注各层本身的功能，而不必知道下面一层功能的实现细节，只需要知道下一层可以提供什么服务、通过什么接口提供服务以及本层要向上一层提供什么服务即可，使得复杂问题局部化；而层次之间关注对应的协议，从而使复杂的网络可以分别设计、整体架构。

（4）建模

计算机通信网是由许多具有信息交换和处理能力的节点互连而成的。要使整个网络有条不紊地工作，就要求每个节点必须遵守一些事先约定好的有关数据格式及时序等的规则。

计算机网络系统体系结构分为多个层次，那么各层之间如何通信呢？

为确保计算机网络各层能够可靠通信，人们专门定义了各层的**网络协议**以及每层的**转换规则**，即建

立了相关的**模型**。在计算机网络中为进行各节点和计算机之间的数据交换而建立的规则、标准或约定称为**网络协议**。

网络体系结构采取的是分层结构，如 TCP/IP 参考模型将网络分为 4 层或 5 层，则网络协议也被分为多个层次，不同的层具有不同的协议。

TCP/IP 协议簇包含了至少几十个协议。例如，网际层主要有 IP、ARP、RARP、ICMP 协议，传输层主要有 TCP、UDP 协议，应用层主要有 Telnet、SMTP、FTP 协议等。

计算机网络通过建立模型，即定义各层通信协议及每层的转换规则，使得对数据的处理和传递变得简单。

1.6.3 从 RSA 算法看计算效率的思维

【案例 1.5】计算效率思维方法案例：RSA 算法。

若 Helen 有明文“public key encryptions”，试设计 RSA 算法，将明文加密后发送给 Peter。为什么说 RSA 算法理论上是不可攻破的？RSA 密钥长度是不是越大越好？对计算效率有何影响？

1. 密码体制相关概念

在分析此案例之前，先介绍密码学的几个基本概念。

(1) 公钥密码体制

公钥密码体制是在密码通信中应用广泛的一种密码体制（或称**密码系统**，Cryptosystem），也称**公钥加密技术**。其基本思想是：若用户 A 有加密密钥 PK（公开）和解密密钥 SK（保密），要求 PK 的公开不影响 SK 的安全。若 B 要向 A 保密送去明文 m，可查 A 的公开密钥 PK_A ，用 PK_A 加密明文 m 得到密文 c；A 收到密文 c 后，用只有 A 自己才掌握的解密密钥 SK_A 对 c 进行解密得到明文 m。公钥密码体制示意图如图 1-47 所示。

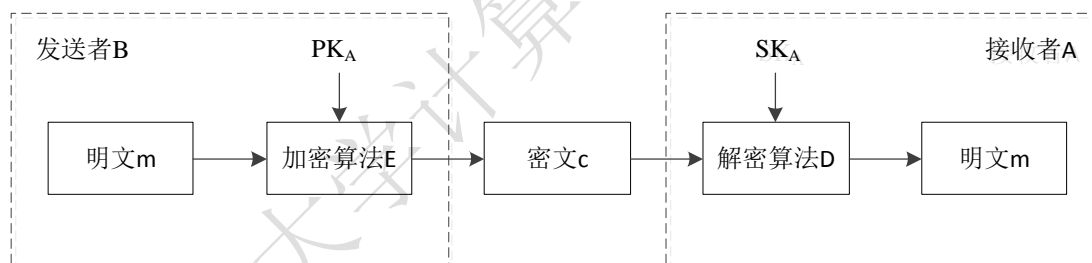


图 1-47 公钥密码体制示意图

密码体制是能完整地解决信息安全中的机密性、数据完整性、认证、身份识别、可控性及不可抵赖性等问题中的一个或几个的一个系统。其基本要素是密码算法和密钥。**密码算法**指信息在加密/解密过程中采用的变换函数。**密钥**是密码算法中的控制参数。尽人皆知的密钥叫做**公开密钥**（public key，简称**公钥**）。只有密钥拥有者才知道的密钥叫做**私有密钥**（private key，简称**私钥**）。这两种密钥合在一起称为**密钥对**。

(2) RSA 公钥密码算法

RSA（Rivest-Shamir-Adleman）是一种基于公钥密码体制的优秀加密算法。RSA 公钥密码算法由 MIT 的 3 位学者 Ron Rivest（李维斯特）、Adi Shamir（沙米尔）和 Leonard Adleman（艾德曼）提出，故算法以它们姓氏的首字母命名。该方案于 1978 年首次发表，ISO 在 1992 年颁布的国际标准 X.509 中，将 RSA 算法正式纳入国际标准。迄今为止，RSA 算法是被使用最多的公钥密码方案，其在公开密钥加密和电子商业中被广泛使用。

RSA 算法的安全性（保密强度）基于“具有大素数因子的合数，其因子分解是困难的”这一数学难题。即给定大整数 n，将 n 分解为两个素数因子 p 与 q，在数学上已证明是难题。因子分解不同位数的合数所

需时间比较如表 1- 11 所示。

从表 1- 11 可以看到，因子分解一个 200 位长的合数，需要计算约 38 亿年！

表 1- 11 因子分解不同位数的合数所需时间比较

合数 n 的十进制位数	因子分解的运算次数	所需计算时间 (每微秒一次)
50	1.4×10^{10}	3.9 小时
75	9.0×10^{12}	104 天
100	2.3×10^{15}	74 年
200	1.2×10^{23}	3.8×10^9 年
300	1.5×10^{29}	4.0×10^{15} 年
500	1.3×10^{39}	4.2×10^{25} 年

因为公钥和私钥是一对大素数 (p 和 q , 是秘密的) 的函数, 所以从一个公钥和密文中恢复出明文的难度等价于分解两个大素数之积 n ($n=p \times q$, 是公开的)。故 RSA 算法在理论上是不可攻破的。

素数即质数 (Prime Number), 指在大于 1 的自然数中, 除了 1 和此数自身外, 不能被其他自然数整除的数。比 1 大但不是素数的自然数称为**合数**。合数是由若干个质数相乘而得来的。

RSA 加密原理基于单向函数, 非法接收者利用公开密钥不可能在有限时间内推算出私有密钥, 从而无法对密文解密。给定一个函数 f , 若对任意给定的 x , 计算 y , 使得 $y=f(x)$ 是容易的; 但对任意给定的 y , 计算 x , 使得 $f(x)=y$ 是难解的, 即计算 $f^{-1}(y)$ 是困难的。则称 f 为**单向函数**。例如: $f(x)=a^x (x, a \in GF(q))$ 就是单向函数。

RSA 公钥密码算法描述如下:

(1) 设计密钥

- ① 选择一对不同的、足够大的素数 p 和 q 。
- ② 计算 $n=p \times q$ 。 n 称为 RSA 算法的模数。
- ③ 计算 n 的欧拉函数值 $f(n)=(p-1) \times (q-1)$, 同时对 p 和 q 严加保密, 不让任何人知道。
- ④ 找一个与 $f(n)$ 互质的数 e (公开指数), 且 $1 < e < f(n)$ 。
- ⑤ 计算秘密解密指数 d , 使得 $d \times e \equiv 1 \pmod{f(n)}$ 。称 d 是 e 关于模 f 的乘法逆元。

乘法逆元的定义: 如果 $ax \equiv 1 \pmod{b}$, 且 $\gcd(a, b)=1$ (a 与 b 互质), 则称 a 关于模 b 的乘法逆元为 x 。


- ⑥ 公开公钥 $KU=(e, n)$, 保密私钥 $KR=(d, n)$ 。

(2) 加密

RSA 算法属于分组加密方案, 也即明文以分组为单位加密。先将明文 (英文) 转换成 $0 \sim n-1$ 之间的一个整数 M 。若明文较长, 可先分割成适当的组, 然后再进行变换。

设密文为 C , 则用公钥 (e, n) 加密明文 M :


$$C = E_{PK}(M) = M^e \pmod{n} \quad (1-7)$$

 **提示:** 明文 (或明文分组) 所变换后的整数一定要小于 n 。即 n 必须足够大, 应大于数字化明文 M , 才能保证每个数字化明文的密文 C 是唯一的。否则, 如果 $M > n$, 由于 $C = M^e \pmod{n}$, 就可能会造成两个不同的 M 对应同一个 C , 则解密后的明文就会发生错误。

(3) 解密

对于密文 C , 用私钥 (d, n) 解密可得到明文 M :

$$M = D_{SK}(C) = C^d \pmod{n} \quad (1-8)$$

 提示：“ \equiv ”是数论中表示同余的符号。在 $d \times e \equiv 1 \pmod{f(n)}$ 中，“ \equiv ”符号的左边必须与符号右边同余，也就是两边以 $f(n)$ 为模的模运算结果相同。因此， $d \times e \equiv 1 \pmod{f(n)}$ 可以写为： $(d \times e) \pmod{f(n)} = 1$ ，即 $(d \times e)$ 除以 $f(n)$ ，余数为 1。

2、求解思路

RSA 公钥密码算法包括设计密钥、加密和解密 3 个步骤。对于本例：


步骤 1：先将明文分块为两个一组（此处为简化计算考虑）

public key encryption

步骤 2：将明文数字化

因为一般明文由若干英文字母组成，而 RSA 算法是对自然数进行加密和解密，所以，首先必须将明文信息数字化。令 a、b...、z 分别为 00、01、...、25，得数字化后的明文为：

1520 0111 0802 1004 2404 1302 1724 1519 0814 1318

 提示：也可以直接利用字符 a~z 的 ASCII 码值作为 a~z 的编码值（Python 中的内置函数 ord(<‘字符’>) 可以获取某个字符对应的 ASCII 码值），得到数字化后的明文；然后进行以下步骤 3~步骤 5 的各项操作：设计密钥、利用加密算法得到密文、利用解密算法对密文解密；最后，对解密后的全部数字化的明文求每个 ASCII 码值对应的字符（Python 中的内置函数 chr(<ASCII 码值>) 可以获取某 ASCII 码值对应的字符），即得到原始明文。


步骤 3：Peter 设计密钥

(1) 选取一对素数 $p=43$ ， $q=59$ ，则模数 $n=p \times q=43 \times 59=2537$

(2) 计算 n 的欧拉函数值 $f(n)=(p-1) \times (q-1)=42 \times 58=2436$

(3) 取公开指数 $e=13$ (e 满足小于 n 且与 $f(n)$ 互质的条件)

(4) 采用扩展欧几里德算法计算 e 的乘法逆元 d

 提示：公元前 300 年，伟大的数学家欧几里德在其著作《几何原本》(Elements) 中提出了欧几里德算法（辗转相除法），用于求两个正整数的最大公约数。该算法经过变形（称为扩展的欧几里德算法），可用于求乘法逆元。已知整数 a 、 b ，扩展欧几里德算法可以在求得 a 、 b 的最大公约数的同时，找到整数 x 、 y （其中一个很可能是负数），使它们满足贝祖等式 $ax + by = \gcd(a, b)$ 。根据乘法逆元的定义： $ax \equiv 1 \pmod{b}$ ，且 $\gcd(a, b)=1$ (a 与 b 互质)，即利用扩展欧几里德算法可以求解等式 $ax + by = 1$ 。令 $a=e$ ， $b=f(n)$ ，求得的 x 即为 e 的乘法逆元 d 。

要解方程 $d \times e \equiv 1 \pmod{2436}$ ，即求解下式：

$$d \times 13 \equiv 1 \pmod{2436} \quad (1-9)$$

(a) 式 (1-9) 可以表示成 $13 \times d - 2436 \times k = 1$ （其中 k 为正整数，即 $13 \times d$ 除以 2436 得到的商）；

(b) 根据欧几里德算法，2436 除以 13 后得到的余数 5 作为新的除数，仍然有 $13 \times d \pmod{5} = 1$ ，故将 2436 对 13 取模得到的余数 5 代替 2436，则式 (1-9) 变为 $13 \times d - 5 \times k = 1$ ；

(c) 同理，上式中将 13 对 5 取模得到的余数 3 代替 13，则变为 $3 \times d - 5 \times k = 1$ ；

(d) 同理，上式中将 5 对 3 取模得到的余数 2 代替 5，则变为 $3 \times d - 2 \times k = 1$ ；

(e) 同理，上式中将 3 对 2 取模得到的余数 1 代替 3，则变为 $d - 2 \times k = 1$ 。

注意：上面各式中的 k 代表进行相应除法后所得的商，并不是同一个值。

当 d 的系数最后化为 1 时，

令 $k=0$ ，代入 (e) 式中，得 $d=1$ ；

将 $d=1$ 代入 (d) 式，得 $k=1$ ；

将 $k=1$ 代入 (c) 式，得 $d=2$ ；

将 $d=2$ 代入 (b) 式，得 $k=5$ ；

将 $k=5$ 代入 (a) 式, 得 $d=937$, 此即为我们要求的秘密解密指数 d 的最终值。

步骤 4: Helen 利用加密算法可得密文

对于第一个分组明文 $M_1=1520$, 有:

$$C_1 = M_1^e \bmod n = 1520^{13} \bmod 2537 = 95$$

同理可以求得其他分组明文的密文, 最后得到全部密文:

0095 1648 1410 1299 1365 1379 2333 2132 1751 1324

步骤 5: Peter 利用解密算法对密文解密后得到恢复的明文

对于步骤 4 中得到的密文 C_1 解密:

$$M_1 = C_1^d \bmod n = 95^{937} \bmod 2537 = 1520 \text{ (明文)}$$

同理可以求得其他分组密文的明文, 最后得到全部 (数字化的) 明文:

1520 0111 0802 1004 2404 1302 1724 1519 0814 1318

可以看到, 利用私钥解密后恢复的明文与步骤 2 中数字化后的明文是完全相同的。而且只有知道私钥的 Peter 能够对密文解密, 这样就保证了信息传输的安全性。

3、RSA 算法安全性与计算效率的平衡

RSA 算法的安全性依赖于大数的因子分解, 但并没有从理论上证明破译 RSA 的难度与大数分解难度等价。目前人们已能分解 140 多个十进制位的大素数, 这就迫使人们使用更长的密钥来确保安全性。RSA 算法密钥长度 (指模数 n 的二进制数的位数) 要求在 1024~2048bit 之间才有安全保障。在实际应用中, RSA 算法的密钥长度至少是 1024bit, 即 $n=2^{1024} \approx 2^{1000} = (2^{10})^{100} = (1024)^{100} \approx (1000)^{100} = 10^{300}$, n 的大小相当于约 300 位以上十进制数。而在安全要求比较高的政府等部门, 需要采用 2048bit 长的密钥。

由于 RSA 算法中使用的模数 n 以及 p 与 q 都是大整数, 所以无论是用硬件实现还是软件实现, 计算效率都比较低。密钥长度的增加虽然提高了安全性, 但另一方面却影响了算法的计算效率。

因此, 在选择 RSA 密钥时, 不能只考虑安全性, 单纯扩大 RSA 密钥长度, 应在系统的安全性和计算效率之间找到一个平衡点。

在第 4 章中, 将通过实例, 进一步讨论, 在进行加密和解密时, 如何通过优化算法来提高加密和解密的计算效率。

本章小结

本章共分为 6 节, 主要围绕计算思维和计算机模型, 一步步揭示计算机模型的建立和发展, 以及蕴含在其中的计算思维。

在 1.1 节中, 介绍了理论思维、实验思维、计算思维这三种思维的概念和特点, 着重介绍了计算思维的概念、特征、方法和本质。随后在 1.2 节中, 介绍了计算的基础, 包括计算的概念, 数据的 0 和 1 表示与物理实现, 计算机必要的逻辑基础——逻辑代数与逻辑运算等。接着在 1.3 节中, 介绍了计算机的理论模型——图灵机模型, 及其物理实现——冯·诺伊曼计算机的组成和特点。在 1.4 节中, 介绍了信息在计算机中的表示方法, 包括计算机中的数据及其单位, 进制及其转换方法, 字符的编码。在 1.5 节中, 从使用的角度介绍现代计算机系统, 包括计算机硬件系统和软件系统, 着重介绍存储数据的关键部件——存储器, 以及层次化的存储体系, 软件系统中的核心软件——控制和管理计算机所有资源的操作系统。在 1.6 节中, 通过三个典型的案例, 分析了计算思维的主要方法的实际应用, 如通信、协作、抽象、分解、关注点分离、建模及计算效率。

计算机之所以能够自动、高速、精确地完成信息处理和其他工作, 是因为, 它很好地解决了以下几个问题:

(1) 数据的表示

在计算机内部，一切数据均采用二进制数表示，即采用 0 和 1 的组合来表示。这是因为，二进制运算规则简单，采用半导体器件表示二进制的两种状态具有诸多优势，二进制算术运算与逻辑运算能够统一起来，二进制数据便于存储。

计算机中的数据又分为数值型数据和非数值型数据。对于数值型数据，为了表示负数，在数的前面增加一位符号位，该位为 0 表示正数，为 1 表示负数。这种带符号位的数称为机器数。计算机内部没有专门设置小数点，它是通过默认小数点在什么位置，来解决实数的表示的。计算机中的实数有两种表示格式：定点数表示法和浮点数表示法。

机器数在计算机中有不同的编码方法——原码、反码和补码。如果采用补码，则减法运算可以用补码加法实现，而乘法可以转化为加法、除法可以转化为减法，这样，加、减、乘、除算术运算都只需用加法器实现，因而大大简化了 CPU 的设计。

对于非数值型数据，不同类型的信息（文本、图形、图像、音频、视频等）采用不同的编码方法。西文字符的编码采用 ASCII 码，国际通用使用 7 位二进制数表示一个字符的编码。汉字编码包括汉字输入码、国标码、机内码、汉字地址码和字形码，计算机对汉字信息的处理过程实际上是各种汉字编码间的转换过程。

（2）数据的存储及自动存取

计算机中的全部信息，包括输入的原始数据、计算机程序、中间运行结果和最终运行结果等都保存在存储器中。存储器是由控制器控制来实现自动存取的。

按照存储器与 CPU 的相近程度，以及它们所处的位置，存储器分为两大类：内存（主存）和外存（辅存）。内存由半导体存储器 RAM 组成，用来暂存当前正在执行的数据、程序和结果。内存可以被 CPU 直接访问，容量小，存取速度快；可读可写，具有掉电易失性。外存一般由磁性材料或光学材料制成，用来存放各种数据文件和程序文件等需要长期保存的信息。CPU 不能直接访问外存，外存中的程序和数据必须先调入内存，才能被 CPU 读取。外存容量大，但存取速度慢，具有掉电非易失性。

现代计算机通常采用 Cache、主存和辅存这种层次化的存储体系，通过优化组合不同性能的资源，来达到存取速度、存储容量、保存时间及价格之间的一个平衡，从而提高计算机的整体性能。

（3）计算规则的表达

由于计算机采用二进制，所以计算规则简单，其加法运算只有四条规则，乘法运算也只有四条规则。进位原则为“逢 2 进 1，借 1 当 2”。

（4）计算规则的自动执行

计算规则是由运算器来执行的，运算器是计算机中执行各种算术运算、逻辑运算及移位操作的部件。而运算器的操作和操作种类又是由控制器决定的。运算器最基本的算术运算是加法。加法器是构成算术运算电路的基本单元电路。由于二进制数之间的减、乘、除算术运算，都可以转化为若干步的加法运算来进行，所以，实现了加法器，就能实现所有的二进制算术运算。

（5）计算机内所有资源的管理和调度

计算机系统是由硬件、软件和数据构成的。没有软件的计算机称为裸机，它是不可能自动工作的。

人们之所以能够方便地使用计算机，归根结底，是因为在计算机上安装了操作系统这一核心软件。操作系统控制和管理着计算机的所有资源，调度计算机各部件协调工作，从而高效、自动地执行程序。正是因为有了操作系统，计算机的硬件功能才得以充分地发挥。

（6）运算速度和精度

计算机的速度和性能主要是由其核心部件——CPU 决定的。CPU 有两个关键性能指标——主频和字长。主频是 CPU 内核工作时的时钟频率。一般说来，一个时钟周期内完成的指令数是固定的，主频越高，CPU 的速度也就越快，即计算机的运算速度越快。字长是 CPU 一次能并行处理的二进制数的位数。字长的大小决定了计算机运算的精度和速度。字长越长，计算机所能处理的数的范围就越大，运算精度越高；计算机处理数据的速度越快。CPU 的字长已从最初的 4 位，发展到现在的 64 位。

计算机科学家设计和构造计算机的过程，以及计算机的体系结构、特点、工作机制，体现了计算思维的一系列方法。例如，在机器数表示中引入补码，则将减法转化为加法运算；乘法采用移位和加法；除法通过若干次“加、减、移位”循环实现，使得各种算术运算均以加法器为基础，从而简化 CPU 的设计，这种设计思想，体现了计算思维“通过约简、嵌入、转化和仿真等方法，求解问题”的思维方法。现代计算机采用 Cache、主存和辅存的层次化存储体系，通过组合优化不同性能的存储资源，来达到速度、容量、价格之间的一个平衡，以保证计算机整体性能，体现了计算思维“在时间和空间之间、在处理能力和存储容量之间进行折中”的思维方法。程序从外存调入内存并被 CPU 执行，是由操作系统的处理机管理（进程管理）、内存管理、文件管理等分工合作完成的，体现了操作系统化整为零、分工合作、协同求解的思维，即计算思维“采用抽象和分解的方法来控制庞杂的任务”的思维方法等。

习 题

1. 简答题

(1) “计算”的含义是什么？计算学科中的计算与小学、中学乃至大学所学的计算有什么差异？若要实现“自动计算”，需要解决哪几个问题？计算机科学家是怎样一步步研究自动计算问题的？

(2) 计算思维的定义和本质是什么？查阅资料，阐述为什么说计算思维是如同所有人都具备“读、写、算”（Reading, wRiting, and aRithmetic，简称 3R）能力一样，都必须具备的思维能力。

(3) 参考教材中关于计算思维主要思维方法的案例的分析，查阅资料，分析你所知道的事物、系统或者解决问题的方法中蕴含了哪些计算思维。你认为学习计算思维对我们日常生活、学习和工作将会有何影响？

(4) 为什么冯·诺伊曼机要采用二进制数而不是十进制数表示指令和代码？用什么器件可以很方便地表示二进制的 0 和 1？

(5) 计算机为什么能像人一样进行加减乘除各种算术运算？逻辑运算又是什么运算？它与算术运算有何区别？逻辑运算有什么用途？

(6) 构造计算机或者数字电路（逻辑电路）的基本元器件是什么？数字电路中的主要开关器件有哪些？

(7) 在计算机中，数字采用二进制表示，那么实数的小数点怎么表示？负数的负号又如何表示？

(8) 什么是定点数？什么是浮点数？浮点数表示法与定点数表示法有何区别？

(9) 总结原码、反码和补码的表示方法。补码加法是怎样进行运算的？为什么在计算机中大多采用补码？

(10) R 进制转换为十进制的方法是什么？十进制转换为 R 进制的方法是什么？

(11) 八进制与十六进制如何实现相互转换？

(12) 说出位、字节和字长的含义。存储容量有哪些单位？现在计算机的硬盘、优盘的最大容量可达多少？

(13) 什么是编码？其主要特征有哪些？什么是字符编码？

(14) 什么是国标码和汉字机内码？它们各有何特点？举例说明国标码和汉字机内码的关系。

(15) 什么是 ASCII 码？计算机如何区分两个字节的数究竟是一个汉字的国标码还是两个西文字符的 ASCII 码？

(16) 什么是汉字输入码？它有哪些编码方法？各有何优缺点？你喜欢用哪种输入码？为什么？

(17) 汉字从通过键盘输入到计算机中，直至显示或打印输出，都经过了哪些处理？

(18) 叙述现代计算机系统的构成。主机主要包括哪些部件？

(19) 什么是 CPU？CPU 的主要性能指标有哪些？

(20) 什么是内存？什么是外存？各有何特点？

(21) 分析现代计算机的存储体系下，不同性能资源的组合优化思维。三级存储系统的结构和工作原理是什么？怎样解决存储速度和存储容量的矛盾？

2. 计算题

(1) 进行下列数的数制转换。

① $(213)_D = ()_B = ()_H = ()_O$

② $(69.625)_D = ()_B = ()_H = ()_O$

③ $(127)_D = ()_B = ()_H = ()_O$

④ $(3E1)_H = ()_B = ()_D$

⑤ $(10A)_H = ()_O = ()_D$

⑥ $(3E1)_H = ()_B = ()_D$

⑦ $(10A)_H = ()_O = ()_D$

⑧ $(670)_O = ()_B = ()_D$

⑨ $(10110101101011)_B = ()_H = ()_O = ()_D$

⑩ $(11111111000011)_B = ()_H = ()_O = ()_D$

(2) 完成二进制数与八进制数和十六进制数之间的转换。

① 将二进制数 11010011.1101101 转换成八进制数。

② 将八进制数 174.536 转换成二进制数。

③ 将二进制数 1101 0011.1101 101 转换成十六进制数。

④ 将十六进制数 17C.5F 转换成二进制数。

(3) 八进制数转换成十六进制数的方法是什么？试将八进制数 2731.62 转换成十六进制数，写出转换过程。

(4) 假定某台计算机的机器数占 8 位，试写出十进制数 $(-67)_D$ 的原码、反码和补码。

(5) 写出字符 B、b、3 和空格的 ASCII 码值。

(6) 试进行两个二进制数 1001 和 0101 的算术运算（加、减、乘、除）。

(7) 将十进制数 33.628 转换成等值的二进制数和十六进制数，要求写出转换过程，且二进制数保留小数点以后 4 位有效位。

3. 单选题

(1) 世界上第一台计算机诞生于_____。

A. 1945 年

B. 1956 年

C. 1935 年

D. 1946 年

(2) 下列叙述中，正确的一条是_____。

A. 二进制正数原码的补码是原码本身

-
- B. 所有十进制小数都能准确地转换为有限的二进制小数
C. 存储器中存储的信息即使断电也不会丢失
D. 汉字的机内码就是汉字的输入码

(3) 下列字符中, ASCII 码值最大的是_____。

- A. H B. 5 C. c D. v

(4) 微型计算机中, 普遍使用的字符编码是_____。

- A. 原码 B. ASCII 码 C. 补码 D. 汉字编码

(5) 一个汉字的内码与其国标码之间的差是_____。

- A. 2020H B. 4040H C. 8080H D. A0A0H

(6) 汉字在计算机中输出的形式是_____。

- A. 内码 B. 字形码 C. 国标码 D. 外码

(7) 下列 4 个数中, 最大的数是_____。

- A. $(1101101)_2$ B. $(98)_{10}$ C. $(137)_8$ D. $(67)_{16}$

(8) 存储 32×32 点阵的一个汉字信息, 需要的字节数是_____。

- A. 12800 B. 3200 C. 32×3200 D. 128K

(9) 一个汉字的国标码需用 2 字节存储, 其每个字节的最高二进制位的值分别为_____。

- A. 0, 0 B. 1, 0 C. 0, 1 D. 1, 1