第 1 章　计算机系统知识

计算机系统是由硬件和软件组成的，它们协同工作来运行程序。本章简要介绍计算机体系结构和存储系统以及安全性、可靠性与系统性能评测基础知识。

1.1　计算机硬件基础知识

计算机的基本硬件系统由运算器、控制器、存储器、输入设备和输出设备 5 大部件组成。运算器、控制器等部件被集成在一起统称为中央处理单元（Central Processing Unit，CPU）。CPU 是硬件系统的核心，用于数据的加工处理，能完成算术运算、逻辑运算及控制功能。存储器是计算机系统中的记忆设备，分为内部存储器和外部存储器。前者速度高、容量小，一般用于存储运行过程中的程序、数据及中间结果。而后者容量大、速度慢，可以长期保存程序和数据。输入设备和输出设备合称为外部设备（简称外设），输入设备用于输入原始数据及各种命令，而输出设备则用于输出计算机运行的结果。

1.1.1　中央处理单元

中央处理单元（CPU）是计算机系统的核心部件，它负责获取程序指令、对指令进行译码并加以执行。

1. CPU 的功能

CPU 的功能如下：

（1）程序控制。CPU 按照程序的安排来执行指令，保证程序指令严格按规定的顺序执行，通过执行程序控制计算机的行为。

（2）操作控制。一条指令功能的实现需要若干操作信号来完成，CPU 产生每条指令的操作信号并将操作信号送往不同的部件，控制相应的部件按指令的功能要求进行操作。

（3）时间控制。CPU 对每条指令的整个执行时间要进行严格控制。同时，指令执行过程中操作信号的出现时间、持续时间及出现的时间顺序都需要进行严格控制。

（4）数据处理。CPU 通过对数据进行算术运算及逻辑运算等方式进行加工处理，数据加工处理的结果为人们所使用。所以，对数据的加工处理是 CPU 最根本的任务。

此外，CPU 还需要对系统内部和外部的中断（异常）做出响应，进行相应的处理。

2. CPU 的组成

CPU 主要由运算器、控制器、寄存器组和内部总线等部件组成，如图 1-1 所示。

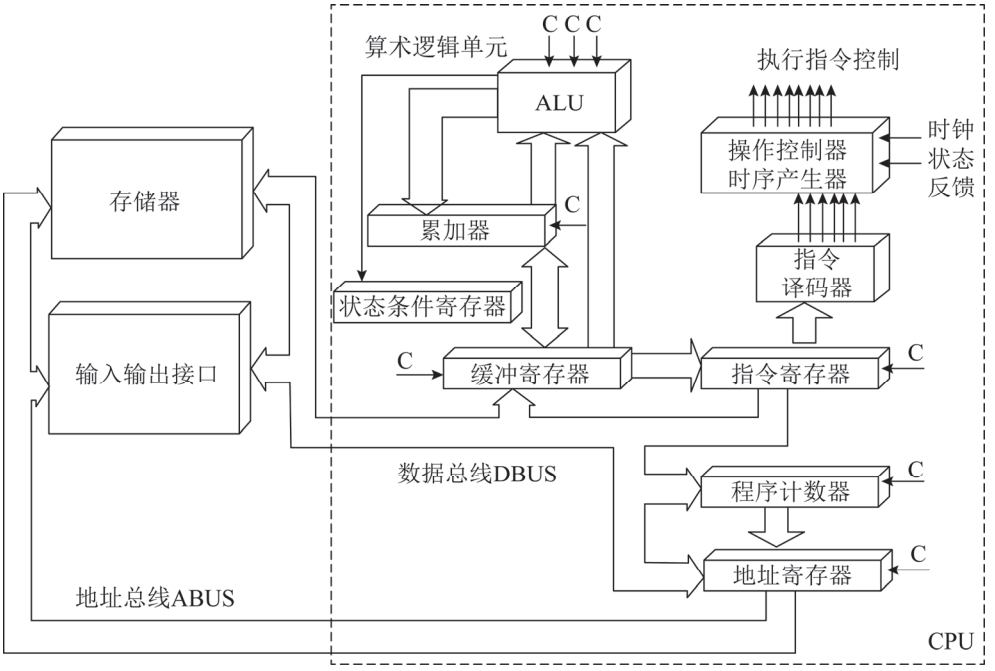

图 1-1　CPU 基本组成结构示意图

1）运算器

运算器包括算术逻辑单元（Arithmetic and Logic Unit，ALU）、累加寄存器、数据缓冲寄存器和状态条件寄存器等，它是数据加工处理部件，完成所规定的各种算术和逻辑运算。相对控制器而言，运算器接受控制器的命令而进行动作，即运算器所进行的全部操作都是由控制器发出的控制信号来指挥的，所以它是执行部件。运算器有如下两个主要功能：

（1）执行所有的算术运算，如加、减、乘、除等基本运算及附加运算。

（2）执行所有的逻辑运算并进行逻辑测试，如与、或、非、零值测试或两个值的比较等。

下面简要介绍运算器中各部件的组成和功能。

（1）算术逻辑单元（ALU）。ALU 是运算器的重要组成部件，负责处理数据，实现对数据的算术运算和逻辑运算。

（2）累加寄存器（AC）。AC 通常简称为累加器，它是一个通用寄存器。其功能是当运算器的算术逻辑单元执行算术或逻辑运算时，为 ALU 提供一个工作区。例如，在执行一个减法运算前，先将被减数暂存在 AC 中，再从内存储器中取出减数，然后与 AC 的内容相减，所得的结果送回 AC 暂存。

（3）数据缓冲寄存器（DR）。在对内存储器进行读写操作时，用 DR 暂时存放由内存储器读写的一条指令或一个数据字，将不同时间段内读写的数据隔离开来。DR 的主要作用为：作为 CPU 和内存、外部设备之间数据传送的中转站以及它们在操作速度上的缓冲；在单累加器结构的运算器中，数据缓冲寄存器还可兼作操作数寄存器。

（4）状态条件寄存器（PSW）。PSW 保存根据算术指令和逻辑指令运行或测试的结果建立的各种条件码内容，主要分为状态标志和控制标志，如运算结果进位标志（C）、运算结果溢出标志（V）、运算结果为 0 标志（Z）、运算结果为负标志（N）、中断标志（I）、方向标志（D）和单步标志等。这些标志通常分别由 1 位触发器保存，保存了当前指令执行完成之后的状态。通常，一个算术操作产生一个运算结果，而一个逻辑操作则产生一个判决。

2）控制器

控制器用于控制整个 CPU 的工作，它决定了计算机运行过程的自动化。它不仅要保证程序的正确执行，而且要能够处理异常事件。控制器一般包括指令控制逻辑、时序控制逻辑、总线控制逻辑和中断控制逻辑等几个部分。

指令控制逻辑要完成取指令、分析指令和执行指令的操作，其过程分为取指令、指令译码、按指令操作码执行、形成下一条指令地址等步骤。控制器在工作过程中主要使用下述几个部件：

（1）指令寄存器（IR）。当 CPU 执行一条指令时，先把它从内存储器取到缓冲寄存器中，再送入 IR 暂存，指令译码器根据 IR 的内容产生各种微操作指令，控制其他部件协调工作，完成指令的功能。

（2）程序计数器（PC）。PC 具有寄存信息和计数两种功能，又称为指令计数器。程序的执行分两种情况，一是顺序执行，二是转移执行。在程序开始执行前，将程序的起始地址送入 PC，该地址在程序加载到内存时确定，因此 PC 的内容即是程序第一条指令的地址。执行指令时，CPU 将自动修改 PC 的内容，以便使其保持的总是将要执行的下一条指令的地址。由于大多数指令都是按顺序来执行的，所以修改的过程通常只是简单地对 PC 加 1。当遇到转移指令时，后继指令的地址根据当前指令的地址加上一个向前或向后转移的位移量产生，或者根据转移指令给出的直接转移的地址产生，再送入 PC。

（3）地址寄存器（AR）。AR 保存当前 CPU 所访问的内存单元的地址。由于内存和 CPU 存在着操作速度上的差异，所以需要使用 AR 保持地址信息，直到内存的读/写操作完成为止。

（4）指令译码器（ID）。指令包含操作码和地址码两部分，为了能执行任何给定的指令，必须对操作码进行分析，以便识别要进行的操作。指令译码器就是对指令中的操作码字段进行分析解释，识别该指令规定的操作，向操作控制器发出具体的控制信号，控制各部件工作，完成所需的功能。

时序控制逻辑要为每条指令按时间顺序提供应有的控制信号。总线逻辑是为多个功能部件服务的信息通路的控制电路。中断控制逻辑用于控制各种中断请求，并根据优先级的高低对中断请求进行排队，逐个交给 CPU 处理。

3）寄存器组

寄存器组可分为专用寄存器和通用寄存器。运算器和控制器中的寄存器是专用寄存器，其作用是固定的。通用寄存器用途广泛并可由程序员规定其用途，其数目因处理器不同有所差异。

3. 多核 CPU

CPU 的核心又称为内核，是 CPU 最重要的组成部分。CPU 中心那块隆起的芯片就是核心，

是由单晶硅以一定的生产工艺制造出来的，CPU 所有的计算、接收/存储命令、处理数据都由核心执行。各种 CPU 核心都具有固定的逻辑结构，一级缓存、二级缓存、执行单元、指令级单元和总线接口等逻辑单元都需要合理的布局。

多核即在一个单芯片上面集成两个甚至更多个处理器内核，其中每个内核都有自己的逻辑单元、控制单元、中断处理器、运算单元，一级 Cache、二级 Cache 共享或独有，其部件的完整性和单核处理器内核相比完全一致。

起初，CPU 的主要厂商 AMD 和 Intel 的双核技术在物理结构上有很大不同。AMD 将两个内核做在一个 Die（晶元）上，通过直连架构连接起来，集成度更高。Intel 则是将放在不同核心上的两个内核封装在一起，因此有人将 Intel 的方案称为“双芯”，将 AMD 的方案称为“双核”。从用户端的角度来看，AMD 的方案能够使双核 CPU 的管脚、功耗等指标跟单核 CPU 保持一致，从单核升级到双核，不需要更换电源、芯片组、散热系统和主板，只需要刷新 BIOS 软件即可。

多核 CPU 系统最大的优点（也是开发的最主要目的）是可满足用户同时进行多任务处理等要求。

单核多线程 CPU 是交替地转换执行多个任务，只不过交替转换的时间很短，用户一般感觉不出来。如果同时执行的任务太多，就会感觉到“慢”或者“卡”。而多核在理论上则是在任何时间内每个核分别执行各自的任务，不存在交替问题。因此，单核多线程和多核（一般每核也是多线程的）虽然都可以执行多任务，但多核的速度更快。

虽然采用了 Intel 超线程技术的单核可以视为双核，4 核可以视为 8 核。然而，视为 8 核一般比不上实际是 8 核的 CPU 性能。

要发挥 CPU 的多核性能，就需要操作系统能够及时、合理地给各个核分配任务和资源（如缓存、总线、内存等），也需要应用软件在运行时可以把并行的线程同时交付给多个核心分别处理。

1.1.2 存储器

1. 存储器的分类

1）按存储器所处位置分类

按存储器所处的位置，可将其分为内存和外存。

（1）内存。内存也称为主存，设置在主机内（或主机板上），用来存放机器当前运行所需要的程序和数据，以便向 CPU 提供信息。相对于外存，其特点是容量小、速度快。

（2）外存。外存也称为辅存，如磁盘、磁带和光盘等，用来存放当前不参与运行的大量信息，必要时可把需要的信息调入内存。相对于内存，外存的容量大、速度慢。

2）按存储器的构成材料分类

按构成存储器的材料，可将其分为磁存储器、半导体存储器和光存储器。

（1）磁存储器。其是用磁性介质做成的，如磁芯、磁泡、磁膜、磁鼓、磁带及磁盘等。

（2）半导体存储器。根据所用元件又可分为双极型和 MOS 型；根据数据是否需要刷新，

又可分为静态（Static memory）和动态（Dynamic memory）两类。

（3）光存储器。如 CD-ROM、DVD-ROM 等光盘（Optical Disk）存储器。

3）按存储器的工作方式分类

按存储器的工作方式可将其分为读写存储器和只读存储器。

（1）读写存储器（Random Access Memory，RAM）。其是既能读取数据也能存入数据的存储器。

（2）只读存储器。根据数据的写入方式，这种存储器又可分为 ROM、PROM、EPROM 和 EEPROM 等类型。

①固定只读存储器（Read Only Memory，ROM）。这种存储器的内容是在厂家生产时就写好的，其内容只能读出，不能改变。一般用于存放系统程序 BIOS 以及用于微程序控制。

②可编程的只读存储器（Programmable Read Only Memory，PROM）。其中的内容可以由用户一次性地写入，写入后不能再修改。

③可擦除可编程的只读存储器（Erasable Programmable Read Only Memory，EPROM）。其中的内容既可以读出，也可以由用户写入，写入后还可以修改。改写的方法是写入之前先用紫外线照射 15～20 分钟以擦去所有信息，然后再用特殊的电子设备写入信息。

④电擦除可编程的只读存储器（Electrically Erasable Programmable Read Only Memory，EEPROM）。与 EPROM 相似，EEPROM 中的内容既可以读出，也可以进行改写。只不过这种存储器是用电擦除的方法进行数据的改写。

⑤闪速存储器（Flash Memory）。其简称闪存，闪存的特性介于 EPROM 和 EEPROM 之间，类似于 EEPROM，也可使用电信号进行信息的擦除操作。整块闪存可以在数秒内删除，速度远快于 EPROM。

4）按访问方式分类

存储器按访问方式可分为按地址访问的存储器和按内容访问的存储器。

5）按寻址方式分类

存储器按寻址方式可分为随机存储器、顺序存储器和直接存储器。

（1）随机存储器（Random Access Memory，RAM）。这种存储器可对任何存储单元存入或读取数据，访问任何一个存储单元所需的时间是相同的。

（2）顺序存储器（Sequentially Addressed Memory，SAM）。访问数据所需要的时间与数据所在的存储位置相关，磁带是典型的顺序存储器。

（3）直接存储器（Direct Addressed Memory，DAM）。介于随机存取和顺序存取之间的一种寻址方式。磁盘是一种直接存取存储器，它对磁道的寻址是随机的，而在一个磁道内则是顺序寻址。

2. 随机访问存储器

随机访问存储器（RAM）分为静态 RAM 和动态 RAM 两类。静态 RAM（SRAM）比动态 RAM（DRAM）更快，也更贵。SRAM 常用来做高速缓存存储器，DRAM 用来作为主存及图

形系统的帧缓冲存储区。

（1）SRAM。在 SRAM 中，将每个位存储在一个双稳态存储器单元中，每个单元是用一个六晶体管电路来实现的。只要供电，SRAM 存储单元的内容就保持不变。

（2）DRAM。在 DRAM 中，每个位由一个电容和一个晶体管组成，电容量很小（容量越小速度越快）。与 SRAM 不同，DRAM 存储器单元对干扰很敏感，电容的电压被扰乱之后也不能再自行恢复，也有其他原因会导致漏电，因此，必须在电容中的电荷漏掉之前进行补充，以保证信息不会丢失，这称为刷新。DRAM 必须周期性地进行刷新操作。

由于集成度高、价格低，DRAM 常用来构成主存储器，主要采用 SDRAM（Synchronous Dynamic Random Access Memory），发展出了 SDR SDRAM、DDR SDRAM、DDR2 SDRAM、DDR3 SDRAM、DDR4 SDRAM 等。

由 DRAM 芯片可组成所需容量要求的内存模块。例如，由 4 个 8M×8 位的 DRAM 芯片（DRAM 0、DRAM 1、DRAM 2、DRAM 3）构成 8M×32 位的内存区域，32 位字的 4 个字节分别由 4 个 DRAM 芯片的同一地址单元提供，DRAM 0 提供第 1 字节（最低字节），DRAM 1 提供第 2 字节，DRAM 2 提供第 3 字节，DRAM 3 提供第 4 字节（最高字节），如图 1-2 所示。

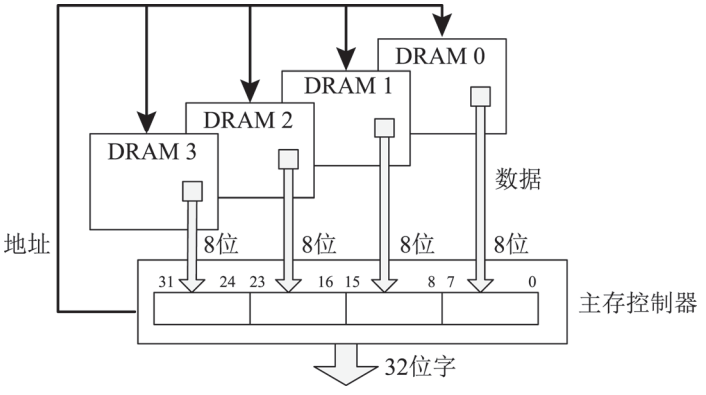

图 1-2　由 4 个 8M×8 位的 DRAM 芯片组成 8M×32 位的内存模块

3. 外存储器

外存储器用来存放暂时不用的程序和数据，并且以文件的形式存储。CPU 不能直接访问外存中的程序和数据，只有将其以文件为单位调入主存后才可访问。外存储器由磁表面存储器（如磁盘、磁带）及光盘存储器构成。下面介绍两种常用的外存储器。

1）磁盘存储器

在磁表面存储器中，磁盘的存取速度较快，且具有较大的存储容量，是目前广泛使用的外存储器。磁盘存储器由盘片、驱动器、控制器和接口组成。盘片用来存储信息。驱动器用于驱动磁头沿盘面径向运动以寻找目标磁道位置，同时驱动盘片以额定速率稳定旋转，并且控制数据的写入和读出。控制器接收主机发来的命令，将它转换成磁盘驱动器的控制命令，并实现主

机和驱动器之间数据格式的转换及数据传送，以控制驱动器的读/写操作。一个控制器可以控制一台或多台驱动器。接口是主机和磁盘存储器之间的连接逻辑。

硬盘是最常见的外存储器。一个硬盘驱动器内可装有多个盘片，组成盘片组，每个盘片都配有一个独立的磁头。所有记录面上相同序号的磁道构成一个圆柱面，其编号与磁道编号相同。文件存储在硬盘上时尽可能放在同一圆柱面上，或者放在相邻柱面上，这样可以缩短寻道时间。

为了正确存储信息，将盘片划成许多同心圆，称为磁道，从外到里编号，最外一圈为 0 道，往内道号依次增加。沿径向的单位距离的磁道数称为道密度，单位为 tpi（每英寸磁道数）。将一个磁道沿圆周等分为若干段，每段称为一个扇段或扇区，每个扇区内可存放一个固定长度的数据块，如 512B。磁道上单位距离可记录的位数称为位密度，单位为 bpi（每英寸位数）。因为每条磁道上的扇区数相同，而每个扇区的大小又一样，所以每条磁道都记录同样多的信息。又因为里圈磁道圆周比外圈磁道的圆周小，所以里圈磁道的位密度要比外圈磁道的位密度高。最内圈的位密度称为最大位密度。

硬盘的寻址信息由硬盘驱动号、圆柱面号、磁头号（记录面号）、数据块号（或扇区号）以及交换量组成。

磁盘容量有两种指标：一种是非格式化容量，它是指一个磁盘所能存储的总位数；另一种是格式化容量，它是指各扇区中数据区容量总和。计算公式分别如下：

非格式化容量＝面数×（磁道数/面）×内圆周长×最大位密度

格式化容量＝面数×（磁道数/面）×（扇区数/道）×（字节数/扇区）

按盘片是否固定、磁头是否移动等指标，硬盘可分为移动磁头固定盘片的磁盘存储器、固定磁头的磁盘存储器、移动磁头可换盘片的磁盘存储器和温彻斯特磁盘存储器（简称温盘）。

2）光盘存储器

光盘存储器是一种采用聚焦激光束在盘式介质上非接触地记录高密度信息的存储装置。

根据性能和用途，光盘存储器可分为只读型光盘（CD-ROM）、只写一次型光盘（WORM）和可擦除型光盘。只读型光盘是由生产厂家预先用激光在盘片上蚀刻不能再改写的各种信息，目前这类光盘使用很普遍。只写一次型光盘是指由用户一次写入、可多次读出但不能擦除的光盘，写入方法是利用聚焦激光束的热能，使光盘表面发生永久性变化而实现的。可擦除型光盘是读写性光盘，它是利用激光照射引起介质的可逆性物理变化来记录信息。

光盘存储器由光学、电学和机械部件等组成。其特点是记录密度高；存储容量大；采用非接触式读/写信息（光头距离光盘通常为 2mm）；信息可长期保存（其寿命达 10 年以上）；采用多通道记录时数据传送率可超过 200MB/s；制造成本低；对机械结构的精度要求不高；存取时间较长等。

1.1.3　总线

所谓总线（Bus），是指计算机设备和设备之间传输信息的公共数据通道。总线是连接计算机硬件系统内多种设备的通信线路，它的一个重要特征是由总线上的所有设备共享，因此可以将计算机系统内的多种设备连接到总线上。

1. 总线的分类

微机中的总线分为数据总线、地址总线和控制总线 3 类。不同型号的 CPU 芯片，其数据总线、地址总线和控制总线的条数可能不同。

数据总线（Data Bus，DB）用来传送数据信息，是双向的。CPU 既可通过 DB 从内存或输入设备读入数据，也可通过 DB 将内部数据送至内存或输出设备。DB 的宽度决定了 CPU 和计算机其他设备之间每次交换数据的位数。

地址总线（Address Bus，AB）用于传送 CPU 发出的地址信息，是单向的。传送地址信息的目的是指明与 CPU 交换信息的内存单元或 I/O 设备。存储器是按地址访问的，所以每个存储单元都有一个固定地址，要访问 1MB 存储器中的任一单元，需要给出 2^{20} 个地址，即需要 20 位地址（2^{20}=1M）。因此，地址总线的宽度决定了 CPU 的最大寻址能力。

控制总线（Control Bus，CB）用来传送控制信号、时序信号和状态信息等。其中有的信号是 CPU 向内存或外部设备发出的信息，有的是内存或外部设备向 CPU 发出的信息。显然，CB 中的每一条线的信息传送方向是单方向且确定的，但 CB 作为一个整体则是双向的。所以，在各种结构框图中，凡涉及控制总线 CB，均是以双向线表示。

总线的性能直接影响整机系统的性能，而且任何系统的研制和外围模块的开发都必须依从所采用的总线规范。总线技术随着微机结构的改进而不断发展与完善。

在计算机的概念模型中，CPU 通过系统总线和存储器之间直接进行通信。实际上在现代的计算机中，存在一个控制芯片的模块。CPU 需要和存储器、I/O 设备等进行交互，会有多种不同功能的控制芯片，称之为控制芯片组。对于目前的计算机结构来说，控制芯片集成在主板上，典型的有南北桥结构和单芯片结构。与芯片相连接的总线可以分为前端总线（FSB）、存储总线、I/O 总线、扩展总线等。

1）南北桥芯片结构

北桥芯片直接与 CPU、内存、显卡、南桥相连，控制着 CPU 的类型、主板的总线频率、内存控制器、显示核心等。前端总线（FSB）是将 CPU 连接到北桥芯片的总线。内存总线是将内存连接到北桥芯片的总线，用于和北桥之间的通信。显卡则通过 I/O 总线连接到北桥芯片。

南桥芯片主要负责外部设备接口与内部 CPU 的联系。其中，通过 I/O 总线将外部 I/O 设备连接到南桥，比如 USB 设备、ATA 和 SATA 设备以及一些扩展接口。扩展总线则是指主板上提供的一些 PCI、ISA 等插槽。

2）单芯片结构

单芯片组方式取消了北桥。由于 CPU 中内置了内存控制器，不再需要通过北桥来控制，这样就能提高内存控制器的频率，减少延迟。还有一些 CPU 集成了显示单元，使得显示芯片的频率更高，延迟更低。

2. 常见总线

常见总线包括：

（1） ISA 总线。ISA 是工业标准总线，只能支持 16 位的 I/O 设备，数据传输率大约是 16MB/s，也称为 AT 标准。

（2）EISA 总线。EISA 是在 ISA 总线的基础上发展起来的 32 位总线。该总线定义 32 位地址线、32 位数据线以及其他控制信号线、电源线、地线等共 196 个接点。总线传输速率达 33MB/s。

（3）PCI 总线。PCI 总线是目前微型机上广泛采用的内总线，采用并行传输方式。PCI 总线有适于 32 位机的 124 个信号的标准和适于 64 位机的 188 个信号的标准。PCI 总线的传输速率至少为 133MB/s，64 位 PCI 总线的传输速率为 266MB/s。PCI 总线的工作与 CPU 的工作是相互独立的，也就是说，PCI 总线时钟与处理器时钟是独立的、非同步的。PCI 总线上的设备是即插即用的。接在 PCI 总线上的设备均可以提出总线请求，通过 PCI 管理器中的仲裁机构允许该设备成为主控设备，主控设备与从属设备间可以进行点对点的数据传输。PCI 总线能够对所传输的地址和数据信号进行奇偶校验检测。

（4）PCI Express 总线。PCI Express 简称为 PCI-E，采用点对点串行连接，每个设备都有自己的专用连接，不需要向整个总线请求带宽，而且可以把数据传输率提高到一个很高的频率。相对于传统 PCI 总线在单一时间周期内只能实现单向传输，PCI Express 的双单工连接能提供更高的传输速率和质量。

PCI Express 的接口根据总线位宽不同而有所差异，包括 X1、X4、X8 以及 X16（X2 模式将用于内部接口而非插槽模式），其中 X1 的传输速度为 250MB/s，而 X16 就是等于 16 倍于 X1 的速度，即是 4GB/s。较短的 PCI Express 卡可以插入较长的 PCI Express 插槽中使用。PCI Express 接口能够支持热拔插。同时，PCI Express 总线支持双向传输模式，还可以运行全双工模式，它的双单工连接能提供更高的传输速率和质量，它们之间的差异与半双工和全双工类似。因此连接的每个装置都可以使用最大带宽。

（5）前端总线。微机系统中，前端总线（Front Side Bus，FSB）是将 CPU 连接到北桥芯片的总线。选购主板和 CPU 时，要注意两者的搭配问题，一般来说，如果 CPU 不超频，那么前端总线是由 CPU 决定的，如果主板不支持 CPU 所需要的前端总线，系统就无法工作。也就是说，需要主板和 CPU 都支持某个前端总线，系统才能工作。通常情况下，一个 CPU 默认的前端总线是唯一的。北桥芯片负责联系内存、显卡等数据吞吐量最大的部件，并与南桥芯片连接。CPU 通过前端总线（FSB）连接到北桥芯片，进而通过北桥芯片与内存、显卡交换数据。FSB 是 CPU 和外界交换数据的最主要通道，因此 FSB 的数据传输能力对计算机整体性能作用很大，如果没足够快的 FSB，再强的 CPU 也不能明显提高计算机整体速度。

（6）RS-232C。RS-232C 是一条串行外总线，其主要特点是所需传输线比较少，最少只需三条线（一条发、一条收、一条地线）即可实现全双工通信。传送距离远，用电平传送为 15m，电流环传送可达千米。有多种可供选择的传送速率。采用非归零码负逻辑工作，电平≤–3V 为逻辑 1，而电平≥+3V 为逻辑 0，具有较好的抗干扰性。

（7）SCSI 总线。小型计算机系统接口（SCSI）是一条并行外总线，广泛用于连接软硬磁盘、光盘、扫描仪等。其中，SCSI–1 是第一个 SCSI 标准，传输速率为 5MB/s；Ultra2 SCSI 的传输速率为 80MB/s；Ultra160 SCSI 也称 Ultra3 SCSI LVD，传输速率为 160MB/s；Ultra320 SCSI 也称 Ultra4 SCSI LVD，传输速率可高达 320MB/s。

（8）SATA。SATA 是 Serial ATA 的缩写，即串行 ATA。它主要用作主板和大量存储设备（如硬盘及光盘驱动器）之间的数据传输。SATA 总线使用嵌入式时钟信号，具备了更强的纠错能力，与以往相比其最大的区别在于能对传输指令（不仅仅是数据）进行检查，如果发现错误会自动矫正，这在很大程度上提高了数据传输的可靠性。串行接口还具有结构简单、支持热插拔的优点。

（9）USB。通用串行总线（USB）当前风头正劲，目前得到十分广泛的应用。USB 由 4 条信号线组成，其中两条用于传送数据，另外两条传送＋5V 容量为 500mA 的电源。可以经过集线器（Hub）进行树状连接，最多可达 5 层。该总线上可接 127 个设备。USB 1.0 有两种传送速率：低速为 1.5Mb/s，高速为 12Mb/s。USB 2.0 的传送速率为 480Mb/s。USB 3.0 的传送速率为 5Gb/s。USB 总线最大的优点还在于它支持即插即用，并支持热插拔。

（10）IEEE-1394。IEEE-1394 是高速串行外总线，近几年得到广泛应用。IEEE-1394 也支持外设热插拔，可为外设提供电源，省去了外设自带的电源，能连接多个不同设备，支持同步和异步数据传输。IEEE-1394 由 6 条信号线组成，其中两条用于传送数据，两条传送控制信号，另外两条传送 8～40V 容量为 1500mA 的电源，IEEE-1394 总线理论上可接 63 个设备。IEEE-1394 的传送速率从 400Mb/s、800Mb/s、1600Mb/s 直到 3.2Gb/s。

（11）IEEE-488 总线。IEEE-488 是并行总线接口标准。微计算机、数字电压表、数码显示器等设备及其他仪器仪表均可用 IEEE-488 总线连接装配，它按照位并行、字节串行双向异步方式传输信号，连接方式为总线方式，仪器设备不需中介单元直接并联于总线上。总线上最多可连接 15 台设备。最大传输距离为 20m，信号传输速率一般为 500KB/s，最大传输速率为 1MB/s。

1.1.4 输入输出控制

从硬件角度看，输入/输出（I/O）设备是电子芯片、导线、电源、电子控制设备、电机等组成的物理设备，从软件角度只关注输入/输出设备的编程接口。

1. I/O 设备概述

I/O 设备可分为块设备和字符设备两类。块设备把信息存放在固定大小的块中，每个块都有自己的地址，独立于其他块，可寻址。例如磁盘、USB 闪存、CD-ROM 等。字符设备以字符为单位接收或发送一个字符流，字符设备不可以寻址。例如打印机、网卡、鼠标键盘等。

I/O 设备一般都包含设备控制器，一般以芯片的形式出现，如南桥芯片。不同的控制器可以控制不同的设备。南桥芯片中包含了多种设备的控制器，如硬盘控制器、USB 控制器、网卡、声卡控制器等。I/O 设备通过总线以及卡槽与计算机其他部件进行连接，如 PCI、PCI-E、SATA、

USB 等。

不同设备控制器的操作控制通过专门的软件即驱动程序进行控制。每个控制器都有几个寄存器与 CPU 进行通信。通过写入这些寄存器，可以命令设备发送或接收数据，开启或关闭。通过读这些寄存器就能知道设备的状态。由于寄存器数量和大小是有限的，所以设备一般会有一个 RAM 性质的缓冲区，来存放一些数据。比如硬盘的读写缓存、显卡的显存等。一方面提供数据存放，另一方面是提高 I/O 操作的速度。

CPU 与 I/O 设备控制器中的寄存器或数据缓冲区如何进行通信？存在以下两个可选方案：

（1）为每个控制器分配一个 I/O 端口号，所有的控制器可以形成一个 I/O 端口空间，这些信息存放在内存中，一般程序不能访问，操作系统则通过特殊的指令和端口号来从设备读取或是写入数据。早期计算机基本都是这种方式，通常使用汇编语言进行操作。

（2）将所有控制器的寄存器映射到内存空间，于是每个设备的寄存器都有一个唯一的地址。这种称为内存映射 I/O。由于不需要特殊的指令控制，对待 I/O 设备和其他普通数据访问方式是相同的，因此可以使用 C 语言来编程。

也可以将上述两种方式相结合，例如，寄存器拥有 I/O 端口，而数据缓冲区则映射到内存空间。

CPU 无论是从内存还是 I/O 设备读取数据，都需要把地址放到地址总线上，然后向控制总线传递一个读信号，还要用一条信号线来表示是从内存还是 I/O 读取数据。

2. 程序控制方式

程序控制 I/O 是指外设数据的输入/输出过程是在 CPU 执行程序的控制下完成的。这种方式分为无条件传送和程序查询方式两种情况。

1）无条件传送

在此情况下，外设总是准备好的，它可以无条件地随时接收 CPU 发来的输出数据，也能够无条件地随时向 CPU 提供需要输入的数据。

2）程序查询方式

通过 CPU 执行程序来查询外设的状态，判断外设是否准备好接收数据或准备好了向 CPU 输入的数据。根据这种状态，CPU 有针对性地为外设的输入/输出服务。

通常，一个计算机系统中可以存在着多种不同的外设，如果这些外设是用查询方式工作，则 CPU 应对这些外设逐一进行查询，发现哪个外设准备就绪就对该外设服务。这种工作方式有两大缺点：一是降低了CPU 的效率，二是对外部的突发事件无法做出及时响应。

计算机系统中的 CPU 是稀缺资源，应尽量提高其利用率，减少等待 I/O 操作的时间。

3. 中断方式

在中断方式下，I/O 设备工作时 CPU 不再等待，而是进行其他的操作，当 I/O 设备完成后，通过一个硬件中断信号通知 CPU，CPU 再来处理接下来的工作。

利用中断方式完成数据的输入/输出过程为：当系统与外设交换数据时，CPU 无须等待也

第 2 章　程序语言基础知识

程序设计语言是为了书写计算机程序而人为设计的符号系统，用于对计算过程进行描述、组织和推导。程序设计语言的广泛使用始于 1957 年出现的 Fortran，程序语言仍然处于不断演化的过程中。

2.1　程序语言概述

本节主要介绍程序设计语言的基本概念、基本成分和一些有代表性的程序语言。

2.1.1　程序语言的基本概念

1. 低级语言和高级语言

计算机的硬件只能识别由 0、1 组成的机器指令序列，即机器指令程序，因此机器指令是最基本的计算机语言。由于机器指令是特定的计算机系统所固有的、面向机器的语言，所以用机器语言进行程序设计时，有较多不便之处，如效率低、程序可读性很差、难以理解、难以修改和维护等。因此，人们就用容易记忆的符号代替 0、1 序列来表示机器指令，例如，用 ADD 表示加法、SUB 表示减法等。用符号表示的指令称为汇编指令，汇编指令的集合被称为汇编语言。汇编语言与机器语言十分接近，其格式在很大程度上取决于特定计算机的机器指令，因此它仍然是一种面向机器的语言，人们称机器语言和汇编语言为低级语言。在此基础上，人们设计了功能更强、抽象级别更高的语言以支持程序设计，于是就产生了面向各类应用的程序语言，称为高级语言。常见的有 Java、C、C++、C#、Python、PHP、JavaScript 等。这类语言与人们使用的自然语言比较接近，大大提高了程序设计的效率。

2. 编译程序和解释程序

高级程序语言必须进行翻译才能为计算机硬件所理解，语言之间的翻译形式有多种，基本方式为汇编、解释和编译。

用某种高级语言或汇编语言编写的程序称为源程序，源程序不能直接在计算机上执行。如果源程序是用汇编语言编写的，则需要一个汇编程序将其翻译成目标程序后才能执行。如果源程序是用某种高级语言编写的，则需要对应的解释程序或编译程序对其进行翻译，然后在机器上运行。

解释程序也称为解释器，它或者直接解释执行源程序，或者将源程序翻译成某种中间代码后再加以执行；而编译程序（编译器）则是将源程序翻译成目标语言程序，然后在计算机上

运行目标程序。这两种语言处理程序的根本区别是：在编译方式下，机器上运行的是与源程序等价的目标程序，源程序和编译程序都不再参与目标程序的执行过程；而在解释方式下，解释程序和源程序（或其某种等价表示）要参与到程序的运行过程中，运行程序的控制权在解释程序。简单来说，在解释方式下，翻译源程序时不生成独立的目标程序，而编译器则将源程序翻译成独立保存的目标程序。

3. 程序设计语言的定义

程序设计语言的定义涉及语法、语义和语用等方面。

语法是指由程序语言的基本符号组成程序中的各个语言结构（包括程序）的一组规则，其中由基本字符构成的符号（单词）书写规则称为词法规则，由符号构成语法成分的规则称为语法规则。程序语言的语法可用形式语言进行描述。

语义是程序语言中按语法规则构成的各个语法成分的含义，可分为静态语义和动态语义。静态语义指编译时可以确定的语法成分的含义，而运行时才能确定的含义是动态语义。一个程序的执行结果说明了该程序的语义，它取决于构成程序的各个组成部分的语义。

语用表示构成语言的各个记号和使用者的关系，涉及符号的来源、使用和影响。语言的实现则有个语境问题。语境是指理解和实现程序设计语言的环境，包括编译环境和运行环境。

4. 程序设计语言的分类

程序语言有交流算法和计算机实现的双重目的，现在的程序语言种类繁多，它们在应用上各有不同的侧重面。若一种程序语言不依赖于机器硬件，则称为高级语言；若程序语言能够应用于范围广泛的问题求解领域，则称为通用的程序设计语言。

1）程序语言发展概述

各种程序语言都在不断地发展之中，许多新的语言也相继出现，各种开发工具在组件化和可视化方面进展迅速。

Fortran（Formula Translation）是第一个被广泛用来进行科学和工程计算的高级语言。一个 Fortran 程序由一个主程序和若干个子程序组成。主程序及每一个子程序都是独立的程序单位，称为一个程序模块。该语言自诞生以来广泛地应用于数值计算领域，积累了大量高效而可靠的源程序。Fortran 语言的最大特性是接近数学公式的自然描述，具有很高的执行效率，目前被广泛地应用于并行计算和高性能计算领域。

ALGOL（ALGOrithmic Language）诞生于晶体管计算机流行的年代，ALGOL 60 是程序设计语言发展史上的一个里程碑，主导了 20 世纪 60 年代程序语言的发展，并为后来软件自动化及软件可靠性的发展奠定了基础。ALGOL 60 有严格的公式化说明，即采用巴科斯范式 BNF 来描述语言的语法。ALGOL 60 引进了许多新的概念，如局部性概念、动态、递归等。

Pascal 是一种过程式、结构化程序设计语言，由瑞士苏黎世联邦工业大学的沃斯（N.Wirth）教授设计，于 1970 年发表。该语言是从 ALGOL60 衍生的，但功能更强且容易使用。Pascal 语言曾经在高校计算机软件教学中一直处于主导地位，其集成开发工具 Turbo Pascal 曾经非常流

行。1985 年发布了 Object Pascal。

C 语言是 20 世纪 70 年代初发展起来的一种通用程序设计语言，UNIX 操作系统及其上的许多软件都是用 C 编写的。它兼顾了高级语言和汇编语言的特点，提供了一个丰富的运算符集合以及比较紧凑的语句格式。由于 C 提供了高效的执行语句并且允许程序员直接访问操作系统和底层硬件，因此在系统级应用和实时处理应用开发中成为主要语言。

C++是在 C 语言的基础上于 20 世纪 80 年代发展起来的，与 C 兼容，但是比 C 多了封装和抽象，增加的类机制使 C++成为一种面向对象的程序设计语言。

C#（C Sharp）是由 Microsoft 公司所开发的一种面向对象的、运行于.NET Framework 的高级程序设计语言，相对于 C++，这个语言在许多方面进行了限制和增强。

Objective-C 是根据 C 语言所衍生出来的语言，继承了 C 语言的特性，是扩充 C 的面向对象编程语言，其与流行的编程语言风格差异较大。由于 GCC（GNU Compiler Collection，GNU 编译器套装）含 Objective-C 的编译器，因此可以在 GCC 运作的系统中编写和编译。该语言主要由 Apple 公司维护，是 MAC 系统下的主要开发语言。与 C#类似，Objective-C 仅支持单一父类继承，不支持多重继承。

Java 产生于 20 世纪 90 年代，其初始用途是开发网络浏览器的小应用程序，但是作为一种通用的程序设计语言，Java 得到非常广泛的应用。Java 保留了 C++的基本语法、类和继承等概念，删掉了 C++中一些不好的特征，因此与 C++相比，Java 更简单，其语法和语义更合理。

Ruby 是松本行弘（Yukihiro Matsumoto，常称为 Matz）大约在 1993 年设计的一种解释性、面向对象、动态类型的脚本语言。在 Ruby 语言中，任何东西都是对象，包括其他语言中的基本数据类型，比如整数；每个过程或函数都是方法；变量没有类型；任何东西都有值（不管是数学或者逻辑表达式还是一个语句，都会有值）等等。

PHP（Hypertext Preprocessor）是一种在服务器端执行的、嵌入 HTML 文档的脚本语言，其语言风格类似于 C 语言，由网站编程人员广泛运用。PHP 可以快速地执行动态网页，其语法混合了 C、Java、Perl 以及 PHP 自创的语法。由于在服务器端执行，PHP 能充分利用服务器的性能。另外，PHP 支持几乎所有流行的数据库以及操作系统。

Python是一种面向对象的解释型程序设计语言，可以用于编写独立程序、快速脚本和复杂应用的原型。Python 也是一种脚本语言，它支持对操作系统的底层访问，也可以将 Python 源程序翻译成字节码在 Python 虚拟机上运行。虽然 Python 的内核很小，但它提供了丰富的基本构建块，还可以用 C、C++和 Java 等进行扩展，因此可以用它开发任何类型的程序。

JavaScript 是一种脚本语言，被广泛用于 Web 应用开发，常用来为网页添加各式各样的动态功能，为用户提供更流畅美观的浏览效果。通常，将 JavaScript 脚本嵌入在 HTML 中来实现自身的功能。

Delphi 是一种可视化开发工具，在 Windows 环境下使用，其在 Linux 上的对应产品是 Kylix，其主要特性为基于窗体和面向对象的方法、高速的编译器、强大的数据库支持、与 Windows 编程紧密结合以及成熟的组件技术。它采用面向对象的编程语言 Object Pascal 和基于构件的开发结构框架。

Visual Basic.NET 是基于微软.NET Framework 的面向对象的编程语言。用.NET 语言（包括 VB.NET）开发的程序源代码不是直接编译成能够直接在操作系统上执行的二进制本地代码，而是被编译成为中间代码 MSIL（Microsoft Intermediate Language），然后通过.NET Framework 的通用语言运行时（CLR）来执行。程序执行时，.NET Framework 将中间代码翻译成为二进制机器码后，使它得以运行。因此，如果计算机上没有安装.NET Framework，这些程序将不能够被执行。

2）程序语言的分类

程序语言的分类没有统一的标准，这里根据设计程序的方法将程序语言大致分为命令式和结构化的程序设计语言、面向对象的程序设计语言、函数式程序设计语言和逻辑型程序设计语言等范型。

（1）命令式程序设计语言。命令式语言是基于动作的语言，在这种语言中，计算被看成是动作的序列。命令式语言族开始于 Fortran，Pascal 和 C 语言都可以体现命令式程序设计的关键思想。

通常所称的结构化程序设计语言属于命令式语言类，其结构特性主要反映在以下几个方面：一是用自顶向下逐步精化的方法编程；二是按模块组织的方法编程；三是程序只包含顺序、判定（分支）及循环构造，而且每种构造只允许单入口和单出口。结构化程序的结构简单清晰、模块化强，描述方式接近人们习惯的推理式思维方式，因此可读性强，在软件重用性、软件维护等方面都有所进步，在大型软件开发中曾发挥过重要的作用。目前仍有许多应用程序的开发采用结构化程序设计技术和方法。C、Pascal 等都是典型的结构化程序设计语言。

（2）面向对象的程序设计语言。程序设计语言的演化从最开始的机器语言到汇编语言到各种结构化高级语言，最后到支持面向对象技术的面向对象语言，反映的就是一条抽象机制不断提高的演化道路。

面向对象的程序设计在很大程度上应归功于从模拟领域发展起来的 Simula，其提出了对象和类的概念。C++、Java 和 Smalltalk 是面向对象程序设计语言的代表，它们都必须支持新的程序设计技术，如数据隐藏、数据抽象、用户定义类型、继承和多态等。

（3）函数式程序设计语言。函数式语言是一类以 λ-演算为基础的语言，其基本概念来自于 LISP，这是一个在 1958 年为了人工智能应用而设计的语言。函数是一种对应规则（映射），它使定义域中每个元素和值域中唯一的元素相对应。

函数定义 1：Square[x]:=x*x

函数定义 2：Plustwo[x]:=Plusone[Plusone[x]]

函数定义 3：fact[n]:=if n=0 then 1 else n*fact[n–1]

在函数定义 2 中，使用了函数复合，即将一个函数调用嵌套在另一个函数定义中。在函数定义 3 中，函数被递归定义。由此可见，函数可以看成是一种程序，其输入就是定义中左边括号中的量。它也可将输入组合起来产生一个规则，组合过程中可以使用其他函数或该函数本身。这种用函数和表达式建立程序的方法就是函数式程序设计。函数式程序设计语言的优点之一就是对表达式中出现的任何函数都可以用其他函数来代替，只要这些函数调用产生相同的值。

函数式语言的代表 LISP 在许多方面与其他语言不同，其中最为显著的是，其程序和数据的形式是等价的，这样数据结构就可以作为程序执行，程序也可以作为数据修改。在 LISP 中，大量地使用递归。常见的函数式语言有 Haskell、Scala、Scheme、APL 等。

（4）逻辑型程序设计语言。逻辑型语言是一类以形式逻辑为基础的语言，其代表是建立在关系理论和一阶谓词理论基础上的 Prolog。Prolog 代表 Programming in Logic。Prolog 程序是一系列事实、数据对象或事实间的具体关系和规则的集合。通过查询操作把事实和规则输入数据库。用户通过输入查询来执行程序。在 Prolog 中，关键操作是模式匹配，通过匹配一组变量与一个预先定义的模式并将该组变量赋给该模式来完成操作。以值集合 S 和 T 上的二元关系 R 为例，R 实现后，可以询问：

①已知 a 和 b，确定 $R(a,b)$是否成立。

②已知 a，求所有使 $R(a,y)$成立的 y。

③已知 b，求所有使 $R(x,b)$成立的 x。

④求所有使 $R(x,y)$成立的 x 和 y。

逻辑型程序设计具有与传统的命令型程序设计完全不同的风格。Prolog 数据库中的事实和规则是形式为“$P:\text{-}P_1, P_2,\cdots, P_n$”的 Hore 子句，其中 $n\geqslant 0$，P_i（$1\leqslant i\leqslant n$）为形如 $R_i(\cdots)$的断言，R_i 是关系名。该子句表示规则：若 P_1，P_2,…，P_n 均为真（成立），则 P 为真。当 $n=0$ 时，Hore 子句变成 P，这样的子句称为事实。

一旦有了事实与规则后，就可以提出询问。测试用户询问 A 是否成立时，采用归结方法。

①如果程序中包含事实 P，且 P 和 A 匹配，则 A 成立。

②如果程序中包含 Hore 子句“$P:\text{-}P_1, P_2,\cdots, P_n$”，且 P 和 A 匹配，则 Prolog 转而测试 $P_1, P_2,\cdots, P_n$。只有当 $P_1, P_2,\cdots, P_n$ 都成立时才能断言 P 成立。当求解某个 P_i 失败时，则返回到前面的某个成功点并尝试另一种选择，也就是进行回溯。

③只有当所有可能情况都已穷尽时，才能推导出 P 失败。

Prolog 有很强的推理功能，适用于书写自动定理证明、专家系统和自然语言理解等问题的程序。

2.1.2 程序语言的基本成分

程序语言的基本成分包括数据、运算、控制和传输。

1. 程序语言的数据成分

程序语言的数据成分指其程序中的数据对象。数据对象总是对应着应用系统中某些有意义的东西，数据表示则指示了程序中值的组织形式。数据类型用于描述数据对象，还用于在基础机器中完成对值的布局，同时还可用于检查表达式中对运算的应用是否正确。

数据是程序操作的对象，具有存储类别、类型、名称、作用域和生存期等属性，使用时要为它分配内存空间。数据名称由用户通过标识符命名，在一些语言中，标识符是由字母、数字和下画线“_”组成的标记；类型说明数据占用内存的大小和存放形式；存储类别说明数据在

内存中的位置和生存期；作用域则说明可以使用数据的代码范围；生存期说明数据占用内存的时间范围。从不同角度可将数据进行不同的划分。

1）常量和变量

按照程序运行时数据的值能否改变，将程序中的数据分为常量和变量。程序中的数据对象可以具有左值和（或）右值，左值指存储单元（或地址、容器），右值是值（或内容）。变量具有左值和右值，在程序运行过程中其右值可以改变；常量只有右值，在程序运行过程中其右值不能改变。

2）全局变量和局部变量

按数据的作用域范围，可将其分为全局量和局部量。系统为全局变量分配的存储空间在程序运行的过程中一般是不改变的，而为局部变量分配的存储单元是可以动态改变的。

3）数据类型

按照数据组织形式的不同可将数据分为基本类型、用户定义类型、构造类型及其他类型。以 C/C++为例，其数据类型如下：

（1）基本类型：整型（int）、字符型（char）、实型（float、double）和布尔类型（bool）。

（2）特殊类型：空类型（void）。

（3）用户定义类型：枚举类型（enum）。

（4）构造类型：数组、结构、联合。

（5）指针类型：type *。

（6）抽象数据类型：类类型。

其中，布尔类型和类类型由 C++语言提供。

2. 程序语言的运算成分

程序语言的运算成分指明允许使用的运算符号及运算规则。大多数高级程序语言的基本运算可以分成算术运算、关系运算和逻辑运算等，有些语言如 C/C++还提供位运算。运算符号的使用与数据类型密切相关。为了明确运算结果，运算符号要规定优先级和结合性，必要时还要使用圆括号。

3. 程序语言的控制成分

控制成分指明语言允许表述的控制结构，程序员使用控制成分来构造程序中的控制逻辑。理论上已经证明，可计算问题的程序都可以用顺序、选择和循环这三种控制结构来描述。

1）顺序结构

顺序结构用来表示一个计算操作序列。计算过程从所描述的第一个操作开始，按顺序依次执行后续的操作，直到序列的最后一个操作，如图 2-1 所示。顺序结构内也可以包含其他控制结构。

2）选择结构

选择结构提供了在两种或多种分支中选择其中一个的逻辑。基本的选择结构是指定一个条

件 P，然后根据条件的成立与否决定控制流走计算 A 还是计算 B，从两个分支中选择一个执行，如图 2-2（a）所示。选择结构中的计算 A 或计算 B 还可以包含顺序、选择和重复结构。程序语言中通常还提供简化了的选择结构，也就是没有计算 B 的分支结构，如图 2-2（b）所示。

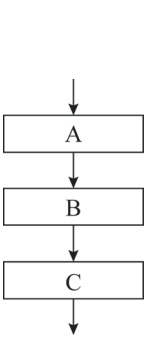

图 2-1　顺序结构示意图

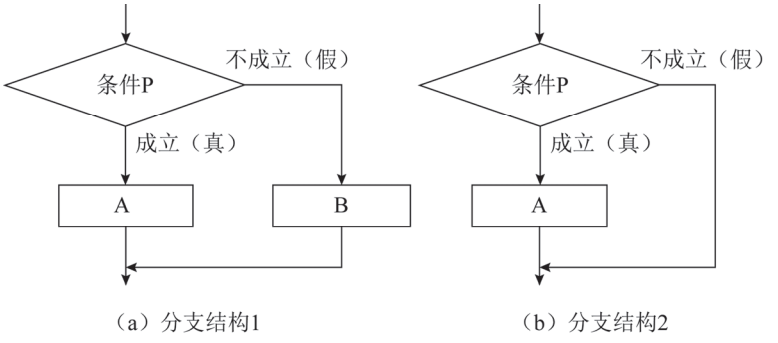

图 2-2　选择结构示意图

3）循环结构

循环结构描述了重复计算的过程，通常由三部分组成：初始化、循环体和循环条件。其中初始化部分有时在控制的逻辑结构中不进行显式的表示。循环结构主要有两种形式：while 型循环结构和 do-while 型循环结构。while 型结构的逻辑含义是先判断条件 P，若成立则执行循环体 A，然后再去判断条件 P；否则控制流就退出循环结构，如图 2-3（a）所示。do-while 型结构的逻辑含义是先执行循环体 A，然后再判断条件 P，若成立则继续执行 A 的过程并判断条件；否则控制流就退出循环结构，如图 2-3（b）所示。

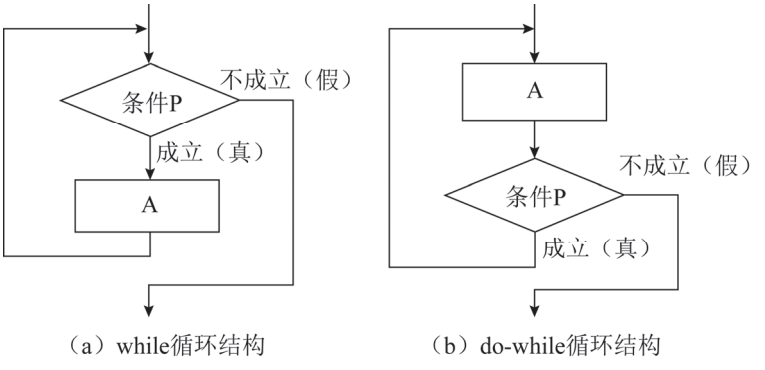

图 2-3　循环结构示意图

4）C/C++语言中的控制语句

（1）复合语句。复合语句用于描述顺序结构。复合语句是一系列用“｛”和“｝”括起来的声明和语句，其主要作用是将多条语句组成一个可执行单元。语法上能出现语句的地方都可以使用复合语句。复合语句是一个整体，要么全部执行，要么一条语句也不执行。

（2）if 语句和 switch 语句。

①if 语句实现的是双分支的选择结构，其一般形式为：

```
if (表达式) 语句1;else 语句2;
```

其中，语句 1 和语句 2 可以是任何合法的 C/C++语句，当语句 2 为空语句时，可以简化为：

```
if (表达式) 语句1;
```

使用 if 语句时，需要注意 if 和 else 的匹配关系。C/C++语言规定，else 总是与离它最近的尚没有 else 的 if 相匹配。

②switch 语句描述了多分支的选择结构，其一般形式为：

```
switch (表达式){
     case 常量表达式1: 语句1;
     case 常量表达式2: 语句2;
     ...
     case 常量表达式n: 语句n;
     default: 语句n+1;
}
```

执行 switch 语句时，首先计算表达式的值，然后用所得的值与列举的常量表达式值依次比较，若任一常量表达式都不能与所得的值相匹配，则执行 default的“语句 n+1”，然后结束 switch 语句。若表达式的值与常量表达式 i(i=1,2,…,n)的值相同，则执行“语句序 i”，当 case i 的语句 i 中无 break 语句时，则执行随后的语句 i+1，语句 i+2……直到执行完语句 n+1 后，才退出 switch 语句；或者遇到 break 时跳出 switch 语句。要使得程序在执行“语句 i”后结束整个 switch 语句，则语句 i 中应包含控制流能够到达的 break 语句。

常量表达式通常为字符型或整型。多个常量表达式可以共用一个语句组。

（3）循环语句。C/C++语言中有 while、do-while 和 for 三种循环语句，用于描述循环计算的控制结构。

①while 语句。while 语句描述了先判断条件再执行循环体的控制结构，其一般形式是：

```
while (条件表达式)  循环体语句;
```

其中，循环体语句多于一条时，应使用“{”和“}”括起来。执行 while 语句时，先计算条件表达式的值，当值为非 0 时，就执行循环体语句，然后重新计算条件表达式的值后再进行判断，否则就结束 while 语句的执行过程。

②do-while 语句。do-while 语句描述了先执行循环体再判断条件的控制结构，其一般格式是：

```
do
```

```
    循环体语句;
while (条件表达式);
```

执行 do-while 语句时，先执行其循环体语句，然后再计算条件表达式的值，若值为非 0，则再一次地执行循环体语句，计算条件表达式并进行判断，直到条件表达式的值为 0 时，才结束 do-while 语句的执行过程。

③for 语句。for 语句的基本格式是：

```
for(表达式 1;表达式 2;表达式 3)  循环体语句;
```

可用 while 语句等价地表示为：

```
表达式 1;
while(表达式 2){
      循环体语句;
      表达式 3;
}
```

for 语句的使用是很灵活的，其内部的三个表达式都可以省略，但用于分隔三个表达式的分号“;”不能遗漏。

C/C++语言中还有实现控制流跳转的 goto、break 和 continue 语句，由于使用 goto 有可能导致程序的逻辑结构不够清晰，因此不提倡使用。

程序语言的传输成分指明语言允许的数据传输方式，如赋值处理、数据的输入和输出等。

4. 函数

函数是程序模块的主要成分，它是一段具有独立功能的程序代码。C 程序由一个或多个函数组成，每个函数都有一个名字，其中有且仅有一个名字为 main 的函数，作为程序运行时的起点。函数的使用涉及三个概念：函数定义、函数声明和函数调用。

1）函数定义

函数的定义包括两部分：函数首部和函数体。函数的定义描述了函数做什么和怎么做。函数定义的一般格式是：

```
返回值的类型  函数名(形式参数表)   //函数首部
{
    函数体;
}
```

函数首部说明了函数返回值的数据类型、函数的名字和函数运行时所需的参数及类型。函数所实现的功能在函数体部分进行描述。

C/C++程序中所有函数的定义都是独立的。在一个函数的定义中不允许定义另外一个函数，也就是不允许函数的嵌套定义。

2）函数声明

对于函数，应该先声明后引用。如果程序中对一个函数的调用在该函数的定义之前进行，则应该在调用前对被调用函数进行声明。函数原型用于声明函数。函数声明的一般形式为：

```
返回值类型 函数名(参数类型表);
```

使用函数原型的目的是告诉编译器传递给函数的参数个数、类型以及函数返回值的类型，参数表中仅需要依次列出函数定义时参数的类型，以使编译器能更彻底地检查源程序中对函数的调用是否合适。

3）函数调用

当在一个函数（称为主调函数）中需要使用另一个函数（称为被调函数）实现的功能时，便以名字进行调用，称为函数调用。在使用一个函数时，只要知道如何调用就可以了，并不需要了解被调用函数的内部实现。因此，主调函数需要知道被调函数的名字、返回值和需要向被调函数传递的参数（个数、类型、顺序）等信息。

函数调用的一般形式为：

```
函数名(实参表);
```

在 C 程序的执行过程中，通过函数调用实现了函数定义时描述的功能。函数体中若调用自己，则称为递归调用。

调用函数和被调用函数之间交换信息的方法主要有两种：一种是由被调用函数把返回值返回给主调函数；另一种是通过参数带回信息。函数调用时实参与形参间交换信息的方法有值调用和引用调用两种。

（1）值调用（Call by value）。若实现函数调用时实参向形式参数传递相应类型的值（副本），则称为是传值调用。这种方式下形参不能向实参传递信息。

在 C 语言中，要实现被调用函数对实参的修改，必须用指针作形参。即调用时需要先对实参进行取地址运算，然后将实参的地址传递给指针形参。本质上仍属于值调用。这种方式实现了间接内存访问。

（2）引用调用（Call by Reference）。引用是 C++中增加的数据类型，当形参为引用类型时，形参名实际上是实参的别名，函数中对形参的访问和修改实际上就是针对相应实际参数所作的访问和改变。例如：

```
void swap(int &x, int &y){   /*交换 x 和 y*/
     int temp;
     temp=x;x=y;y=temp;
}
```

函数调用：swap(a,b);。

在实现调用 swap(a,b)时，x、y 就是 a、b 的别名，因此，函数调用完成后，交换了 a 和 b 的值。

2.2 程序语言翻译基础

语言翻译程序是系统软件的一种，其主要作用是将高级语言或汇编语言编写的程序翻译成某种机器语言程序，使程序可在计算机上运行。语言处理程序主要分为汇编程序、编译程序和解释程序三种基本类型。

2.2.1 汇编程序基本原理

1. 汇编语言

汇编语言是为特定的计算机或计算机系统设计的面向机器的符号化程序设计语言。用汇编语言编写的程序称为汇编语言源程序。因为计算机不能直接识别和运行符号语言程序，所以要用专门的翻译程序——汇编程序进行翻译。用汇编语言编写程序要遵循所用语言的规范和约定。

汇编语言源程序由若干条语句组成，一个程序中可以有三类语句：指令语句、伪指令语句和宏指令语句。

（1）指令语句。指令语句又称为机器指令语句，将其汇编后能产生相应的机器代码，这些代码能被 CPU 直接识别并执行相应的操作。常见的基本指令如 ADD、SUB 和 AND 等，书写指令语句时必须遵循指令的格式要求。

指令语句可分为传送指令、算术运算指令、逻辑运算指令、移位指令、转移指令和处理机控制指令等类型。

（2）伪指令语句。伪指令语句指示汇编程序在汇编源程序时完成某些工作，例如给变量分配存储单元地址，给某个符号赋值等。伪指令语句与指令语句的区别是：伪指令语句经汇编后不产生机器代码，而指令语句经汇编后要产生相应的机器代码。另外，伪指令语句所指示的操作是在源程序被汇编时完成的，而指令语句对应的操作必须在程序运行时完成。

（3）宏指令语句。在汇编语言中，还允许用户将多次重复使用的程序段定义为宏。宏的定义必须按照相应的规定进行，每个宏都有相应的宏名。在程序的任意位置，若需要使用这段程序，只要在相应的位置使用宏名，即相当于使用了这段程序。因此，宏指令语句就是宏的引用。

2. 汇编程序

汇编程序的功能是将汇编语言所编写的源程序翻译成机器指令程序。汇编程序的基本工作包括将每一条可执行汇编语句转换成对应的机器指令；处理源程序中出现的伪指令。由于汇编指令中，形成操作数地址的部分可能出现后面才会定义的符号，所以汇编程序一般需要两次扫