第5章 计算机组件

在电视剧《三体》中，上万人组成的人列计算机运行场景宏大而震撼。每个人都在其中扮演了一个小零件，或举起手中的红、白灯，或御马沿道奔驰，或端坐疾书；而无数红、白灯此起彼落，无数的骏马多次往返，无数的纸笔记录信息，组成一个巨大的计算机器。

实际上，人列计算机的设计和现实中的计算机无比相似。士兵举起手中的红灯或白灯对应着数字计算机中的 1 和 0，骏马奔驰的道路对应着计算机中的总线，而记录数据的文书则是对应计算机的存储器；同时，士兵主要以方阵形式分布排列，每一块区域的士兵负责不同的工作，对应着计算机组件的不同功能，如 CPU、外存、显示屏等。人列计算机的主要结构和功能与现实计算机大致一致。计算机组件的组成和连接是计算机科学中极为重要的一部分。了解了计算机的组成，才能更好地理解计算机的运行方式。

本章将对计算机组件进行介绍，包括中央处理器的基本结构、计算机的存储器分类、计算机各个组件之间的连接方式，以及计算机的运行周期等。通过相关理论以及应用的介绍，为读者呈现计算机的静态结构以及动态运行逻辑，同时通过一些具体的示例来加深对计算机组件和运行的理解。

本章的主要内容如下：

- 计算机存储器的概念和分类
- 计算机各个组件之间的连接方式
- 计算机的机器周期
- 计算机的并行架构

5.1 冯·诺依曼体系

冯·诺依曼模型(von neumann architecture)是计算机体系结构的一种基本设计原理，也被称为冯·诺依曼体系结构。第 1 章已经介绍了冯·诺依曼模型，本章将进行更详细的学习。

如图 5.1 所示，冯·诺依曼模型主要包含以下几个关键组件：

中央处理器(central processing unit，CPU)负责执行计算机指令、控制数据的流动和进行算术逻辑运算。中央处理器主要包括**控制单元(control unit，CU)**和**算术逻辑单元(arithmetic logic unit，ALU)**，控制单元负责指令的解码和控制信号的生成，而算术逻辑单元执行算术和逻辑运算。可以说，中央处理器是计算机的核心，负责计算机程序的运行和信息处理。中央处理器对于计算机可以类比于大脑对于人类，人依靠大脑思考，决定下一步的行动，而计算机依靠中央处理器处理数据，控制下一步的操作。总而言之，中央处理器能够决定计算机的每一步操作，负责计算机运行中所有数字和逻辑的计算。可以说一台计算机的性能在一定程度上主要取决于中央处理器的性能。

存储器(memory)负责存放程序和数据，日常所说的内存属于存储器的一种。计算机存储器可以存储数据和程序，存储数据是计算机从设计之初就具有的功能，而存储程序

是由冯·诺依曼提出，这也是冯·诺依曼模型最重要的特点之一，当代所有计算机都具有这一特点。在冯·诺依曼模型中，程序和数据都以二进制形式存储在同一种存储介质中，并且可以根据需要进行读取和写入操作。

总线(bus)用于不同组件之间的数据传输和通信。中央处理器和存储器处在计算机内部，二者通过总线进行互通。总线可以分为数据总线、地址总线和控制总线，分别用于传输数据、地址和控制信号。总线的具体概念和功能在后面会详细讲解，这里可以暂时理解为两个结构之间负责传输数据的数据线。

输入/输出设备(I/O device)通过接口与计算机内部进行连通，实现信息的输入和输出，如生活中外部设备最经常使用的接口是 USB 接口。输入设备(input device)用于将外部数据和指令输入到计算机中，如键盘、鼠标、扫描仪等。输出设备(output device)用于将计算机处理的结果输出给用户，如显示器、打印机、音频设备等。计算机输入/输出设备属于外部设备，其他常见的外部设备还有外部存储设备(外存)，如磁盘、U 盘等。

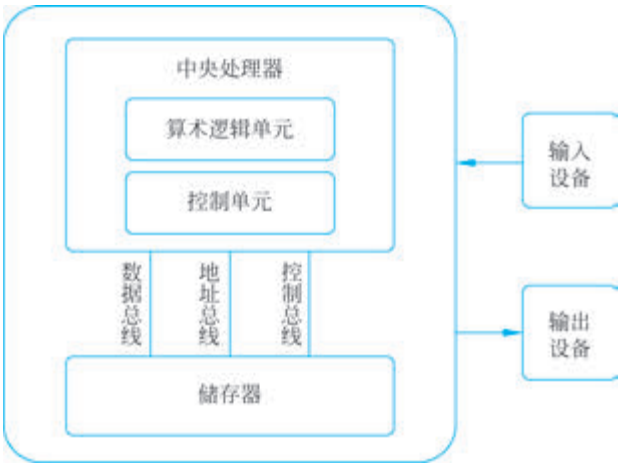

图 5.1　冯·诺依曼体系结构

冯·诺依曼模型有两个重要特点：存储程序和指令按顺序执行。如果读者有运行代码的经历，就能够更好地理解这两个特点的含义：存储程序就是计算机将程序员所写的代码以计算机能够理解的语言存储到内部存储器中；按顺序执行则是从上至下地运行代码中的每一条代码所代表的指令。这两个特点的提出奠定了现代计算机系统的基础，无论后来的计算机如何发展，最终都无法脱离冯·诺依曼模型。在后面的学习中也能够发现，计算机各组件之间的联系和程序的运行都脱离不了冯·诺依曼模型，而存储程序和指令按顺序运行这两个特点也在程序运行中起着重要作用。

5.2 计算机组件的介绍

本节主要介绍计算机各个组件的功能和简单的工作原理，同时将各个组件进行联系，建立较为完整的计算机体系，了解计算机的运行基础，为后面学习机器周期做准备。本书主要包括下列要点：

- 中央处理器的组成及功能
- 存储器的分类
- 总线的概念和功能
- 计算机外部设备

5.2.1 中央处理器

中央处理器(CPU)是计算机的运算和控制中心,计算机中所有关于数字逻辑的运算都在中央处理器中完成,计算机执行的操作也由中央处理器控制。更具体地,中央处理器由三个结构组成:算术逻辑单元、控制单元和一系列寄存器,这三个结构分别具有运算功能、控制功能和存储功能,三者之间相互联系和配合,实现程序的运行和信息的处理。中央处理器的大致结构如图 5.2 所示。接下来,将对中央处理器中的三大结构进行进一步介绍。

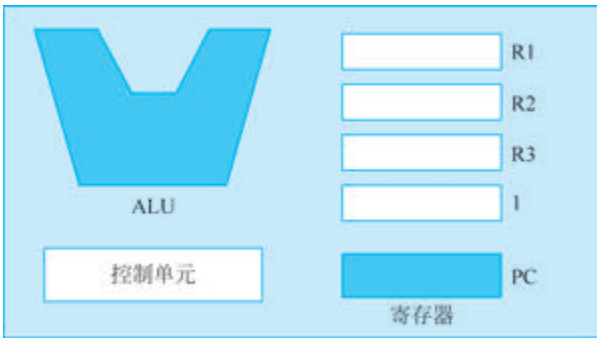

图 5.2 中央处理器的大致结构

1. 算术逻辑单元

算术逻辑单元(ALU)是计算机进行算术和逻辑运算的组件,由算术单元和逻辑单元组成。逻辑单元主要负责逻辑运算,如与运算、或运算、非运算等;算术单元进行算术计算,如加法、减法、乘法、除法以及更高级的数学运算。其实,根据我们已有的知识可以知道,算术运算在底层层面也是通过逻辑运算来实现的,如之前在第 4 章所学习的半加器就可以进行两位二进制数的计算,而半加器实现算术运算又是通过逻辑门实现的,所以也可以说,计算机中所有的运算都是通过逻辑运算实现的。由此可见,算术逻辑单元由大量逻辑门组成,功能越强大的算术逻辑单元,所需要的逻辑门越多。例如,简单的算术逻辑单元往往不能直接进行乘法运算,而是通过多次相加才能得到最后的结果,如恒温器、电视遥控器等只进行简单运算的机器中的算术逻辑单元只能进行加减法运算;而手机、计算机等需要进行复杂运算的机器则有专门进行乘法运算的算术逻辑单元。可以想象,能进行乘法运算的算术逻辑单元会比只能进行加法运算的算术逻辑单元结构复杂得多。

算术逻辑单元进行运算时,操作单元是一个**字(word)**,其长度由**机器字长**决定。机器字长取决于中央处理器芯片,往往指算术逻辑单元可以进行运算操作数据的长度。现在经常使用的 64 位中央处理器中,每个字的长度就是 8 字节。机器字长不同的中央处

理器进行计算的速度也不同，在大部分情况下，机器字长越长的计算机，计算速度越快。举例来说，若机器字长为 8 位，进行一次两个 8 位数的加法运算时，算术逻辑单元只需要进行一次加法计算；而进行一次两个 16 位数的加法运算时，算术逻辑单元则需要进行两次加法计算，分别对低八位和高八位进行计算，才能得到结果。从这个例子可以看出，机器字长较长的机器，进行数据较大的数据运算时，比机器字长较短的机器计算速度快，但在进行数据较小的数据运算时，就无法充分利用算术逻辑单元的运算功能，造成空间的浪费。

2. 控制单元

控制单元(CU)能够通过发送指令来控制其他单元的运行。指令就是要求计算机进行的一定操作的命令，常见的指令有“加”“减”“无操作”等，指令越多，计算机可以直接进行的操作越多。每一条指令都对应一个编码串，可以理解为每一条指令都有一个独一无二的代号，编码串和指令的对应关系在机器出厂前就会写在硬件上，计算机通过识别编码串来执行相应的指令，如“0000”表示不做操作，“0001”表示加一等。可以看出，计算机可以执行的指令种类越多，指令的编码串就越长。在计算机运行中，控制单元发送需要执行指令的编码串，计算机根据控制单元发送的编码串来执行相应的指令。在后面机器周期的学习中，控制单元在计算机的取址-执行周期(fetch-execute cycle)中具有重要作用。

中央处理器可以进行的所有的指令称为**指令集(instruction set)**，不同计算机的指令集不同。在计算机进行运算时，一些简单计算可以通过指令集中的指令就可以直接完成；但在进行一些复杂的运算时，部分计算机的指令集中就没有相对应的指令，此时计算机就需要通过对指令集中的其他指令进行组合，才能完成复杂计算。举个例子，假设有两台计算机，一台计算机的指令集中有“加”和“乘”的指令，另一台计算机的指令集中只有“加”的指令，当两台计算机同时进行一个乘法运算时，前者就可以直接相乘得到结果，而后者只能通过多次相加得到结果。从这个例子可以看出，指令集越简单，编程过程越复杂，因为需要将繁杂的任务化简到少数指令中去执行。当然，这一过程并不需要计算机的使用者完成，而是通过程序员制作的编译器进行编译来实现，即在代码完成后，需要编译器对代码进行编译，计算机才能够运行程序。指令集的设计需要考虑多个方面，包括指令的种类和功能、指令的格式和编码、寻址方式、寄存器的使用等。指令集的设计会影响计算机的性能、指令执行的效率以及编程的方便性。

根据指令集的种类可以将计算机分为两类：**复杂指令集计算机(complex instruction set computer，CISC)**和**精简指令集计算机(reduced instruction set computer，RISC)**。复杂指令集计算机有 100～200 个指令，可以直接进行复杂计算；而精简指令集计算机只有 30～40 个指令，只能进行简单操作，通过编译器将复杂操作化简为多个简单操作来完成。生活中常见的两个计算机系列，英特尔系列和苹果系列，就分别支持 CISC 和 RISC。

3. 寄存器

寄存器(register)是在中央处理器中快速存储的单位，存储空间很小，只能短暂地存储信息，断电后寄存器所存储的信息就会丢失。算术逻辑单元进行运算时，需要先把所

需要的数据从主存储器提取到寄存器中，再进行运算；计算得到的结果，也会先存入寄存器，再转存到主存储器中。同时，寄存器也可以存储指令和地址。

根据存储内容的种类，寄存器可以分为三种：数据寄存器、指令寄存器和程序计数器。

数据寄存器(data registers)负责存储数据，如数字、字符等，这些数据都以二进制的形式存储在数据寄存器中。数据寄存器的大小取决于算术逻辑单元的操作单元的大小，即机器字长的大小；换而言之，数据存储器可以存储一个字。

指令寄存器(instruction registers，IR)负责存储指令。当从主存储器中抓取指令时，指令会被存储到指令寄存器中供 CPU 执行。指令寄存器的大小取决于指令的长度，即能够存储的指令位数。这里要说明，在计算机科学中，指令往往有两种意思，一种是指令集中的指令，如 Add、Subtract 等，它仅表示一种操作码；还有一种是后面会提到的计算机完整指令，如 Load 1001 R1(其中 1001 是地址，R1 是寄存器的名称，这条指令的意思是将 1001 存储单元中的数据存放在 R1，这些内容会在后面详细介绍)。指令寄存器中存储的是后者。

程序计数器(program counter，PC)存储下一个指令的地址，即每一次操作结束后，中央处理器会根据程序计数器所寄存的地址到主存储器中抓取指令，寄存到指令寄存器后再由算术逻辑单元执行；每次抓取指令后，PC 会自动加一个数，这个数的大小由程序决定。这样，CPU 就可以按照程序的顺序依次执行指令序列。程序计数器的大小取决于地址的位数，即能够寻址的内存空间大小。有关地址的概念会在后面存储器板块进行学习。寄存器之间的相互配合和联系在后面机器周期的学习中会进一步展现。

寄存器在计算机运行过程中起到了关键的作用，它们作为临时存储单元，可以快速存取和操作数据、指令和地址。通过寄存器，CPU 可以高效地进行数据处理和指令执行，从而提高计算机的运行效率。

5.2.2 存储器

1. 主存储器与地址空间的概念

主存储器(main memory)主要用于存储数据和程序。生活中，人们通常称主存储器为内存，但其实主存储器只是计算机内部存储器中的一种，计算机中还存在其他的内部存储器，如寄存器和高速缓冲存储器。

主存储器是一定数量的存储单元(memory unit)的联结，每个存储单元可以存储一个字，字的长度取决于**存储字长**，存储字长由计算机的型号决定，不同的计算机存储字长可能不同。这里要说明，存储字长和前面提到的机器字长并不一样，两者的长度有可能相同，也有可能不相同，但都是在计算机出厂前就已决定。一个存储单元对应一个唯一的地址，计算机可以通过地址访问到相应存储单元中的数据；存储单元无论存储什么内容，最终都会转换成二进制形式进行存储。

如图 5.3 所示为主存储器示意图，每个小格子就是一个存储单元，其中八位二进制数就是这个存储单元存储的信息，这台机器的存储字长是 8 位。当然，生活中很多数据

都是一个存储单元无法存储的，这时，这个数据就会由相邻的多个存储单元进行存储。例如，在 C++ 中，一个 int 类型的变量需要四个存储单元进行存储，一个 char 类型的变量需要两个存储单元进行存储。

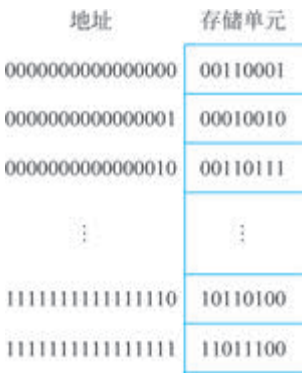

图 5.3　主存储器示意图

地址(address)是数据存储位置的编码，计算机通过地址访问相应的数据，地址与存储单元一一对应，每一个地址都独一无二。地址通常以无符号二进制整数表示，例如，“0101001110101101”就可以表示一个特定存储单元的地址。地址的长度取决于存储单元的数量，存储单元越多，地址长度越长，这可以类比为房屋住宅，当房屋的数量增多时，存在的房屋编号就变大，房屋编号的位数也会越来越长。一般情况下，存储单元是 2^n 个，这时地址用 $0\sim2^n-1$ 的 n 值二进制形式表示。在图 5.3 中，存储器有 2^{16} 个存储单元，地址由 16 位二进制表示。

思考题：如果一台计算机内存大小为 128MB，且这台计算机的存储字长是 8B。那么这台计算机的地址长度是多少？

解答：计算机的存储单元的数量为

$$\frac{128\text{MB}}{8\text{B}}=16\times2^{20}=2^{24}$$

因此这台计算机的地址长度为 24bit＝3B。

2. 高速缓冲存储器

高速缓冲存储器(cache memory)位于主存储器与中央处理器之间，用来存储主存储器中的部分常用数据。相较于主存储器，高速缓冲存储器的存储空间较小，但获取和存储数据的速度更快，这可以理解为离中央处理器越近，获取和存储数据的速度更快。有研究表明，大部分计算机在 80％的时间只会处理 20％的数据，这称为 80-20 法则。所以，如果这些数据存储在中央处理器中，计算机的运行速度会大大提高，但由于寄存器的存储空间极小，中央处理器无法存储过多数据，因此，科学家选择使用高速缓冲存储器来提高计算机的运行效率。我们可以这样理解：高速缓冲存储器相较于主存储器，距离中央处理器更近，获取和存储数据的速度更快，即使稍慢于寄存器，但也几乎相近。在程序运行时，中央处理器会预先复制一部分数据到高速缓冲存储器中，每次调取数据时会先到高速缓冲存储器中搜索，如果目标数据不在高速缓冲存储器中才会到主存储器中获取，当然，这一概率也只有 20％，这样一来，程序的运行速度会大大提升。

至此，计算机中三类可以存储数据的结构已经叙述完毕。根据获取数据的速度，可以将这三类存储器由快到慢作一个排序：寄存器＞高速缓冲存储器＞主存储器。这三类存储器都属于计算机内部存储器，都位于冯・诺依曼体系的核心，因此三者虽然有快慢之分，但都远远快过外部存储器。

3. 随机存取存储器与只读存储器

同时，存储器还可以分为随机存取存储器和只读存储器两类。这两种存储器都是半导体存储介质，属于计算机的内部存储器，但它们在功能和特性上有一些重要区别。

随机存取存储器（random access memory，RAM）又称为读/写存储器，顾名思义，它既可以进行读取的操作，也可以进行撰写的操作。RAM 用于临时存储数据和程序，具有快速的读写速度。但 RAM 是易失性（volatile）存储器，意味着当计算机断电时，RAM 中存储的数据会被清除，因此需要持续供电来保持数据的保存。RAM 的存储容量可以根据计算机的需求进行扩展或缩减，现代计算机通常具有多种层次的 RAM，包括主存储器和高速缓冲存储器。

只读存储器（read only memory，ROM）只能读取，不能撰写。ROM 用于存储永久性的程序和数据，其内容通常由制造商在制造过程中编程，用户只能读取其中的数据，而无法修改或擦除。与 RAM 不同的是，ROM 中存储的数据在断电后仍然保留，因此具有非易失性，不需要持续供电来保持数据。由于以上特性，ROM 常用于存放永久性的程序和数据，如初始引导程序、监控程序、基本输入/输出管理程序（BIOS）等。

早期的 ROM 由于受技术的限制，只能进行读取工作，而不能将其中的内容擦除。如今，随着技术的发展，科学家在 ROM 的基础上发明了新的半导体存储介质——EPROM 和 EEPROM。这两种存储器不像 ROM 那样只能进行只读操作，而是可以进行读写操作。其实这已经不符合 ROM 的命名规则，但由于二者都是由 ROM 技术衍生出来的，所以采用了一部分 ROM 的叫法。

5.2.3 总线

计算机是一个整体，各个部分都需要信息的交流和沟通，所以，计算机需要一个连接各部分的结构，这个结构就是总线。

总线（bus）是连接计算机各子系统用于传送信息的公共通道，是一类信号线的集合。总线的英文又可以译作“公交车”，这也是对总线功能形象的比喻。任何人都可以到公交车路线的站点乘坐公交车，而总线就相当于计算机中的公交车，所有数据都可以通过相应总线传送到计算机的指定位置，如中央处理器（CPU）从存储器中获取数据或将输出数据存储到存储器中，如图 5.4 所示。

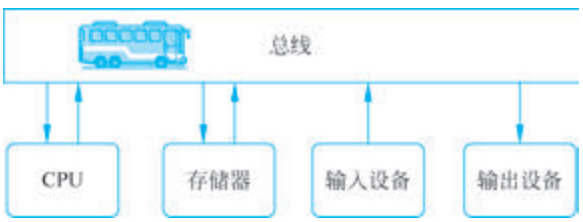

图 5.4 计算机组件通过总线连接

根据总线传输信息的内容，总线可以分为三种：数据总线、地址总线和控制总线。

数据总线（data bus）就是用于传送数据的线路，宽度取决于机器字长，如一个 16 位

的数据总线可以同时传输 16 位的数据。

地址总线(address bus)就是用于传送地址的线路,以指定读取或写入数据的位置。它的宽度取决于地址空间的长度。

控制总线(control bus)用于传送计算机指令集中的指令(注意,这里的指令仅指操作码,而不是计算机中的一条完整指令)。控制总线的宽度取决于指令集的大小,中央处理器获取程序中的指令就是通过控制总线进行传输的。

总线的速度和宽度对计算机系统的性能和数据传输速度有重要影响。总线越宽,每次能传送的数据、地址或指令越多。而较快的总线速度可以提高数据的传输速度。

接下来,我们通过一道思考题对所学的知识进行进一步的学习。该题是对本节目前所学内容的综合应用,许多容易混淆、忽略的知识点在此例题中都有所体现,希望读者能认真梳理和体会。

思考题:假设一台虚拟计算机有四个数据寄存器(记为 R0～R3),主存储器中有 1024 个存储单元,该机器有 16 种不同的操作指令(Add、Subtract 等)。一条完整的指令是如下格式:Add 565 R2(注意,565 在这里是计算机的一个地址),试回答:

(1) 一条指令的最小长度是多少位?

(2) 这台计算机每个指令寄存器的大小是多少?

(3) 如果此计算机中央处理器的数据字长等于指令字长,那么每个数据寄存器的大小是多少?

(4) 程序计数器的大小是多少?

(5) 数据总线的大小是多少?

(6) 地址总线的大小是多少?

(7) 控制总线的大小是多少?

解答:

(1) 指令的操作有 $16=2^4$ 种,至少要用 4 位表示;存储单元有 $1024=2^{10}$ 个,至少要用 10 位的地址空间表示;寄存器有 $4=2^2$ 个,至少要用 2 位表示。故一条指令的最小长度是(4+10+2)位=16 位。

(2) 由于指令寄存器用于存储完整指令,故大小等于一条完整指令的大小。故每个指令寄存器的大小是 16 位。

(3) 数据寄存器寄存数据,由题意可得数据大小等于指令大小。故数据寄存器的大小为 16 位。

(4) 程序计数器存储下一个指令的地址,因此大小等于地址长度的大小。故程序计数器的大小为 10 位。

(5) 数据总线传输数据,此计算机数据大小为 16 位,故数据总线的大小为 16 位。

(6) 地址总线传输地址,故地址总线的大小为 10 位。

(7) 控制总线传输指令,注意,控制总线传输的指令不是题目中的完整指令,而是指令集中的 Add、Subtract 等操作,因此,控制总线的大小为 4 位。

总之，总线在计算机系统中起到了数据传输和通信的桥梁作用，它可以连接多个硬件组件，包括中央处理器、存储器、输入/输出设备和其他外部设备。通过总线，这些设备可以共享数据和控制信息，以实现计算机的协调工作。我们会在后面的机器周期小节给出一个有关不同计算机硬件组件通过总线进行协调工作的动态实例。

知识闲谈

实际中经常遇到有人会说，这台机器是“64 位”的，这个“64 位”指代的是什么？

其实，这个问题没有标准答案。通常情况下，当提到一台机器是“64 位”时，意味着这台机器的中央处理器的数据寄存器的长度为“64 位”，即算术逻辑单元能够处理操作数的长度，就是机器字长。这意味着该处理器可以一次性处理 64 位的数据。这对于进行复杂的计算和处理大量数据的任务是非常有利的。

但也有人用“64 位”代表地址长度或存储字长。地址长度是指处理器能够直接寻址的内存空间大小或地址总线的宽度，而存储字长是指存储单元的宽度或数据总线的宽度。在这种情况下，“64 位”表示这台机器的地址长度或存储字长为 64 位，意味着它可以支持更大的内存空间或处理更大的数据块。

因此，对于“64 位”的具体含义，需要结合上下文和使用场景来理解，以确保准确传达和理解信息。

5.2.4 计算机外部设备

根据各自的功能，计算机外部设备可以分为输入/输出设备、外部存储设备和网络设备等。前面提到，总线是计算机内部各个组件之间进行数据传输的通道，因此它也扮演着计算机与外部设备之间数据传输的桥梁角色。但值得注意的是，由于主机和输入/输出等外部设备的特性（介质、速度等）不同，计算机需要一种中间媒介来处理这种不同，这个中间媒介称为**接口（interface）**。接口能够处理外部设备和计算机兼容性的问题，确保它们能够正确地进行通信和交互。计算机各组件连接示意图如图 5.5 所示。

常见的接口包括：USB（universal serial bus）接口，用于连接许多外部设备，如键盘、鼠标、打印机、存储设备等；音频接口，用于连接音频设备，如扬声器、麦克风、耳机等；HDMI（high-definition multimedia interface）接口，用于音视频传输；Ethernet 接口，用于网络连接等。

1. 外部存储设备

计算机外部存储设备用于扩展计算机的存储容量，提供额外的存储空间来保存数据和文件。常见的计算机外部存储设备包括：磁性存储设备（如磁盘）、光学存储设备（如光盘）、闪存等，下面分别进行介绍。

磁盘（magnetic disk）是计算机常见的外部存储设备，常见的磁盘种类有硬盘和软盘。软盘是最早出现的可移动存储器，体积较小，可拆卸随身携带，自身带有电源，通过软盘存储器读取存储的信息；可存储空间小，最大存储空间为 1.44 MB，且存储读取信息的速度慢，自从 U 盘出现之后软盘的使用就逐渐减少。而硬盘则体积大，使用时需要连

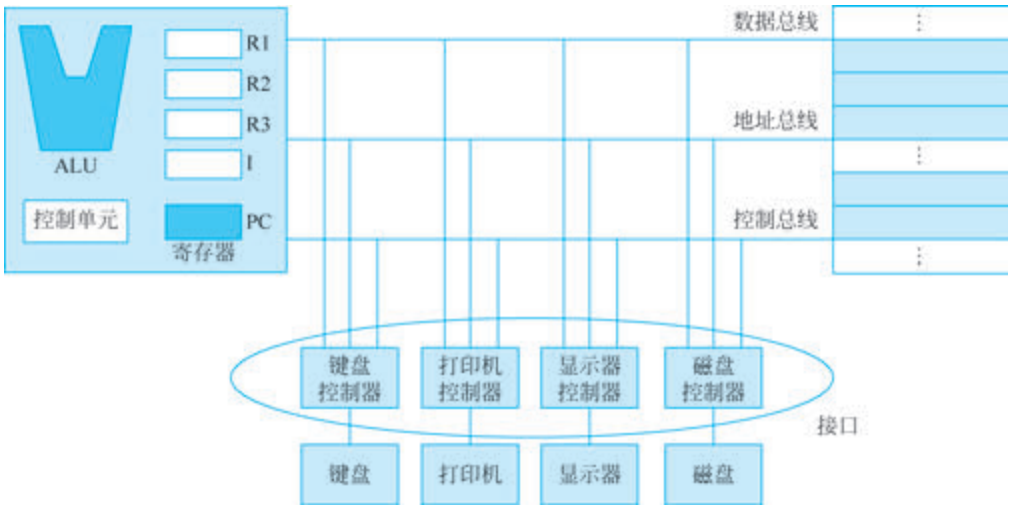

图 5.5　计算机各组件连接示意图

接电源，不方便携带，通常安装在计算机的硬盘驱动器中。硬盘存储空间很大，且存储读取信息速度较快。下面具体介绍硬盘中磁盘的使用。

从形状上来看，磁盘是一块圆形扁平的盘状物，磁盘可以通过多个同圆心且大小不同的圆以及多条均匀分布的直线划分为不同区域，如图 5.6(a)所示。图中圆环称为**轨道(track)**，直径所分割出的扇形称为**扇区(sector)**，一个扇区和一个轨道相交的区域称为磁盘的一个**数据块(block)**。数据块是磁盘的读取单元，每个数据块都对应着一个轨道号和扇区号，即每个数据块都有唯一的标号。在读取信息磁盘时，计算机通过控制磁盘自身旋转来确定轨道，通过控制读取头的移动来确定扇区，两者配合读取相应数据块中存储的内容。

实际上，现在的计算机使用硬盘时往往不是只使用一个磁盘，而是多个磁盘堆叠在一起组成一个**阵列(RAID)**。如图 5.6(b)所示，这些磁盘在一个固定转轴上转动，并通过多个固定在同一机械臂上的读写头的移动来读取有关信息。此时，一个数据块的地址只由轨道号和扇区号已经不能确定，还需要另一个编号来确定数据块是在第几个盘上。而且由于在同一时刻，各磁盘的读取头所在的轨道号相同，所以某一时刻读取头所在的轨道可以形成一个**柱面(cylinder)**，轨道号又可以称为柱面号。此时，每个数据块的地址可以由柱面号、盘号和扇区号决定。

通过寻道时间、转速和传送时间可以评判磁盘质量的性能。寻道时间是读写头移动到相应轨道所需要的时间；转速是磁盘的转动速度，即磁盘旋转到相应扇区所具有的速度；传送时间是磁盘和计算机之间的传送时间。磁盘的读取数据速度就由这三者决定，速度越快，磁盘的性能越好。

此外，磁盘上数据的存储和获取遵循一定的规则：读取头一次只可以读取一个数据块；每个数据块都是随机存储，既可以读取，也可以撰写，但相较于 RAM，磁盘上的存储数据是不易丢失的，断电后仍旧存在。

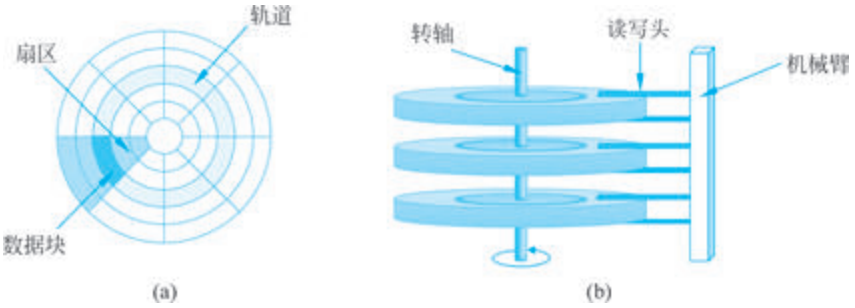

图 5.6　磁盘的模型

光盘(optical disc)是一种常见的外部存储介质,它利用光学技术的物理原理。在读取信息时,光盘主要通过激光反射的方向和角度来读取所存储的内容。光盘的盘面有着凹凸不平的纹路,光盘可以将二进制形式的信息存储在这些凹凸不平的纹路中,通常,凹面代表 1,凸面代表 0。当激光照射到光盘盘面上的纹路时,不同的纹路反射回来的激光不同,接收器在不同角度接收反射回来的激光,就可以知晓激光所照射的纹路属于凸面还是凹面,进而读取光盘中的信息。

生活中最常见的光盘有 CD(compact disc)、DVD(digital versatile disc)两种,二者在存储容量、数据传输速率和使用技术上有所区别。通常,CD 存储空间较小,大约为 700 MB,常用于存储数据和音频; DVD 存储空间较大,约为 4.7 GB,常用于存储数据或视频。因此,人们通常将 CD 用于存放音乐,将 DVD 用于存储电影。

最早发明出来的光盘是不可擦写光盘,后面又发明出可擦写光盘,二者最大的区别在于存储信息后是否能够进行修改。不可擦写光盘有 CD-R、DVD-R 等,“R”代表此光盘只能进行一次刻录,刻录完后就不可以修改。这种光盘经常用于存储一些需要长期保存、不会改变的信息,如存储歌曲的 CD 和存储影片的 DVD。可擦写光盘包括 CD-RW、DVD-RW 等,“RW”代表光盘在刻录信息后,如果想进行修改,可以擦除原本的信息后再重新存入信息,但是擦除次数是有限制的,理论上,大部分可擦写光盘的擦除次数能够达到 1000 次,但由于存放方式不当、使用出现磨损等原因,实际可擦除次数往往无法达到 1000 次。

根据光盘刻录次数,光盘又可以分为一次刻录和多次刻录。一次刻录是指光盘刻录一次后进行封口,意思是无论光盘还剩下多少存储空间没有使用,都不能够再刻录其他内容; 因此,在使用一次刻录光盘时,往往要确定光盘存储空间都能使用完后,再进行刻录,否则会造成存储空间的浪费。多次刻录则是指光盘刻录一次后还可以再次进行刻录,添加其他数据,这比一次刻录更便于日常使用。但多次刻录存在一个风险,反复刻录会造成光盘中的数据杂乱,如果出现一个区域多次刻录的情况,很可能造成数据丢失、文件被破坏的结果。由此可见,一次刻录的光盘存储数据安全,但日常使用不便; 而多次刻录的光盘日常使用方便,但有数据丢失的风险。

这里要说明,一次刻录、多次刻录与可擦写、不可擦写既有相似之处,也有很大差异。下面以比喻的形式进一步解释它们之间的联系和区别: 一次刻录可以比喻成考试,交卷

以后就不能继续修改和添加；多次刻录可以比喻成做笔记，每上完一节课还可以在后面添加新的文字；不可擦写就是用钢笔写字，写上后就不能删除或修改；可擦写则是用铅笔写字，事后还可以用橡皮擦掉重写。总而言之，多次刻录意味着在光盘上添加新的内容，可擦写则是对光盘上已有的内容进行删除；一次刻录光盘是不可擦写光盘，而多次刻录光盘既可以是可擦写光盘，又可以是不可擦写光盘。

举个例子，无论是 CD-R 、DVD-R，还是 CD-RW、DVD-RW，都可以进行多次刻录，在需要的时候，使用者可以在原有的基础上对光盘进行再次刻录，但前者刻录完后就不可以修改光盘中的数据，意味着其只能添加新的数据和文件，但不能对原有的数据进行修改和删除；而后者可以将光盘内存储的数据全部擦除后再进行重新刻录，这意味着光盘可以添加新的数据和文件，并且在必要时还可以对光盘进行“格式化”并重写。

光盘具有较大的存储容量、长时间的数据保存能力以及良好的便携性，作为一种外部存储介质，在过去几十年中起到了重要的作用，它广泛应用于数据存储、音视频媒体、软件发布和备份等领域。然而，随着存储技术的进步，光盘逐渐被闪存驱动器和云存储等新型存储介质所取代，但在某些特定的应用场景仍然有重要作用。

闪存(flash memory)是一种常见的非易失性存储器技术，基于电子存储技术，能够长时间地保存数据，即使是在设备断电之后。闪存具有许多优点，使其成为流行的存储解决方案之一。首先，它具有较快的读取和写入速度，使得数据的传输和存储更加高效。其次，闪存具有较小的体积和轻巧的特点，使得它成为移动设备中理想的存储介质。此外，闪存还具有耗电量较低、耐用性高等优势。

闪存的工作原理是利用了半导体器件中的晶体管和电荷存储单元。它使用了一种特殊的电荷存储机制，可以在没有电源的情况下持久地保存数据。闪存的存储单元被分为称为“位”的小区域，每个位可以存储一个或多个二进制数据。通过控制晶体管的通断状态，可以对位进行编程(写入数据)和擦除(删除数据)操作。此外，闪存分为两种主要形式：NAND 闪存和 NOR 闪存。NAND 闪存是最常见的类型，它以数据块(block)为单位进行读写操作，适用于大容量存储和高速数据传输。NOR 闪存则以字节为单位进行读写操作，适用于存储那些容量不大但需要快速启动的代码。

作为一种电学设备，闪存的内部电路与计算机的内部电路十分兼容，但由于闪存仍属于计算机外部设备，需要通过接口挂载在总线上，闪存的数据读取速度在一定程度上受接口的速度影响，因此，通过改善接口设计可以在一定程度上提高闪存的读取速度。例如，最常见的闪存种类就是 U 盘，全称是 USB 闪存驱动器(USB flash drive)，它通过 USB 接口接入计算机内部。USB 有多个版本，其中最常见的是 USB 2.0 和 USB 3.0：USB 2.0 是较早的版本，它的传输速度相对较低，最高传输速率为 480 Mb/s(兆位每秒)；USB 3.0 是较新的版本，也称为 superspeed USB，它引入了新的技术和改进方案，传输速度比 USB 2.0 快得多，最高传输速度可达到 5 Gb/s(千兆位每秒)，约为 USB 2.0 传输速度的 10 倍。

当前另一种非常常见的闪存设备是固态硬盘(solid state drive，SSD)，已显现出逐渐用于替代传统机械硬盘(hard disk drive，HDD)的趋势。SSD 采用闪存存储芯片，通常是

NAND 闪存来存储数据，而不需要机械磁盘和移动读写头。这使得固态硬盘相比传统机械硬盘具有许多优势，例如：固态硬盘可以直接访问存储芯片中的数据，而不需要等待磁盘马达旋转和移动读写头寻址，因此具有更快的数据读取和写入速度；由于没有机械运动和移动部件，对振动、冲击和温度变化等外部因素具有抗干扰性，具有较高的耐用性，可以承受更多的数据写入操作；固态硬盘还具有较小的体积和较轻的重量，使其非常适合于便携设备和薄型计算机等场景。然而，相比于传统机械硬盘，固态硬盘的存储容量较小、价格相对较高。

2. 输入/输出设备

计算机的输入/输出设备在日常生活中扮演着至关重要的角色。它们使得人们能够与计算机进行交互，并将计算机处理的数据和信息呈现出来。最常见且基本的输入/输出设备包括鼠标、键盘和显示器。

随着不同工作和娱乐需求的增加，更高级的输入/输出设备应运而生。例如，针对特定领域的输入需求，出现了游戏控制器、绘图板和数字画板等外部输入设备。它们为游戏和图形设计等领域提供了更加精准和直观的输入方式。而对于输出需求，投影仪、显示屏幕和音响系统等外部输出设备能够将计算机中的内容以更大的尺寸和更高的音质输出。这样可以满足更广泛的展示和传播需求，使得观众能够更好地体验和享受多媒体内容。

3. 其他外部设备

还有其他类型的外部设备在计算机系统中起着重要的作用。例如，网络设备是连接计算机与互联网或局域网的关键组件。调制解调器、路由器和以太网适配器等是常见的网络设备。调制解调器用于将计算机信号转换为可通过电话线路传输的数字信号，以便连接到互联网。路由器用于在局域网和广域网之间进行数据包转发和网络流量管理。以太网适配器则负责将计算机连接到以太网局域网，并实现网络通信。更多关于网络设备的介绍请参考第 6 章。

扩展卡是一种用于扩展计算机功能和性能的设备。它们通常以插入计算机主板上的扩展槽中的形式添加到系统中。常见的扩展卡包括显卡、声卡和网卡。显卡用于处理计算机图形输出，提供高质量的图像和视频显示。声卡则负责处理计算机的音频输入和输出，提供高保真的音效体验。网卡用于连接计算机到网络，使其能够进行网络通信和访问互联网。

在本节的最后，我们要了解冯·诺依曼模型的所有结构都会连接在计算机**主板（motherboard）**上。主板也是计算机系统中最重要的基础组件之一，它承担着多项关键功能。主板是一个电路板，提供了各种连接和插槽，用于安装和连接计算机的核心组件，包括中央处理器、内存、外存、扩展卡、输入/输出设备等。主板内部包含着许多关键的电路和芯片，用于控制和协调各个组件的工作。主板上的插槽和连接器允许各种组件通过电缆、插针等方式与主板进行物理连接。这些插槽和连接器提供了各种接口和协议，使不同的组件能够相互配合工作，并通过总线系统实现数据的传输和交换。需要指出的是，计算机主板的设计和规格会根据不同的计算机类型和用途而有所差异。不同的主板具

有不同的芯片组、插槽类型和扩展功能。因此，在搭建一台高性能和可靠的计算机系统时，选择适合自己计算机需求的主板是一个重要的步骤。

5.3 机器周期

计算机组件的学习过程是静态地对计算机运行进行学习，接下来，将介绍计算机如何动态地执行指令的获取和执行。本节首先给出计算机机器周期的概念，随后用一个机器周期的示例来介绍计算机各个组件之间是如何配合工作的。

1. 计算机的机器周期

计算机的机器周期又称为**取址-执行周期(the fetch-execute cycle)**，是指计算机完成一次操作所需要的一系列步骤，这些步骤需要各个计算机组件之间相互配合才能完成。一个机器周期大致可以分为以下四个步骤。

(1) **取指(fetch)**：计算机的控制单元首先从主存储器中的指定地址获取下一条指令，并将其存储到指令寄存器中。这一步骤涉及程序计数器(PC)，它存储着下一条指令的地址。控制单元通过地址总线将程序计数器中的地址传送到主存储器，并将获取的指令通过数据总线传送到指令寄存器中。然后，程序计数器自动加上一个数，其数值由执行的程序决定。

(2) **解码(decode)**：在指令寄存器中获取的指令需要通过控制单元进行解码，即将指令转换成计算机能够理解的机器语言。解码后的指令确定了计算机下一步要执行的操作，如获取输入设备数据、进行数值计算或将数据输出到设备。

(3) **取值(get)**：在执行指令过程中，如果需要使用相关的数据，算术逻辑单元会将相关地址发送到主存储器，获取需要的数据，并将其复制到内部数据寄存器中。

(4) **执行(execute)**：算术逻辑单元执行指令所规定的操作。根据指令的类型，可能需要对数据进行计算、存储或输出等操作。如果在上一步骤中获取了有关数据，这些数据将参与指令的执行过程。

执行指令后，一个机器周期完成，并进入下一个机器周期。可以认为，上述步骤(1)和(2)属于取指周期，由 CPU 中的控制单元主要负责，步骤(3)和(4)为执行周期，由 CPU 中的算术逻辑单元主要负责，如图 5.7 所示。

总体来说，计算机的机器周期是一个由多个步骤组成的循环过程，通过控制单元、寄存器、算术逻辑单元(ALU)和主存储器等组件的协调工作，实现了指令的获取、解码和执行。这些步骤的顺序和操作的细节会根据计算机体系结构和具体指令集的设计而有所不同。了解机器周期有助于理解计算机的工作原理和指令执行的基本过程。

2. 机器周期的示例

首先，先回顾一下冯·诺依曼模型的两个特点：一个是存储程序，另一个是指令按顺序执行。这两个特点极大地影响了计算机机器周期的步骤。在现在的计算机中，主存储器往往会分为多个区域，其中专门有不同的区域负责存储指令和数据，由于计算机是按顺序执行程序中的指令的，所以计算机程序的执行其实就是一个个指令的执行周期。下

面以图 5.8 为例，介绍如何使用计算机执行一段程序。

图 5.7　机器周期

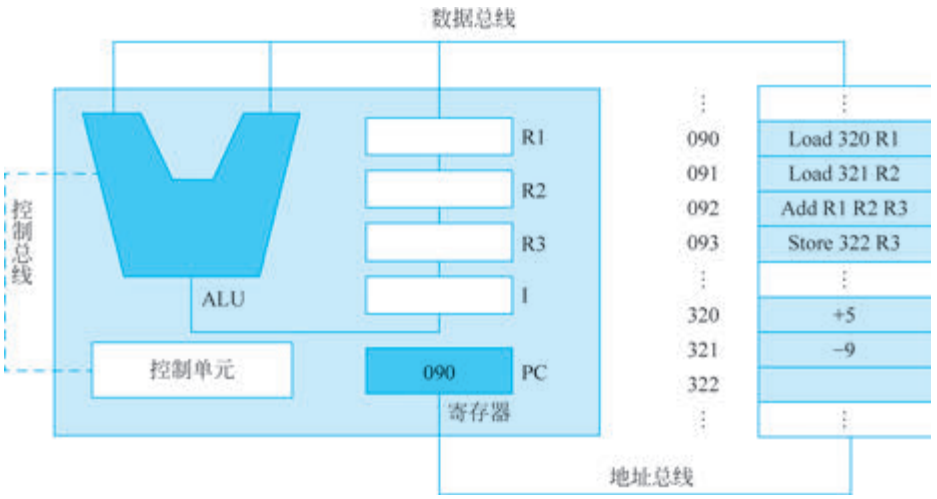

图 5.8　机器周期的示例

在图 5.8 中，R1、R2、R3 是数据寄存器，I 是指令寄存器，PC 是程序计数器。在最开始时，PC 中已经存储了下一条指令的地址。在此示例中，PC 存储的地址为 090，因此，地址 090 首先通过地址总线输送到主存储器，并将地址为 090 的存储单元中的指令通过数据总线传送回中央处理器的指令寄存器 I 中。此时，由于程序是按顺序执行的，PC 中的数据就会自动加 1。这里要注意，PC 不是在所有情况下都加 1，在不同的情况下可能所加的数不同，在该示例中，由于一条完整的指令刚好占据一个存储单元，PC 每执行完一次自动加 1，以跳转到下一条指令的地址。完成以上步骤后，算术逻辑单元开始执行 I 中的指令 Load 320 R1，将地址为 320 的存储单元中的数据＋5 放入 R1 中，本轮机器周期结束，最后的结果如图 5.9 所示。

接下来的过程和以上相似。在第二个机器周期，地址 091 中的指令被存入指令寄存器 I，PC 加 1，算术逻辑单元执行 I 中的指令 Load 321 R2，将地址 321 中的数据－9 存入

R2 中，本轮机器周期结束，最后的结果如图 5.10 所示。

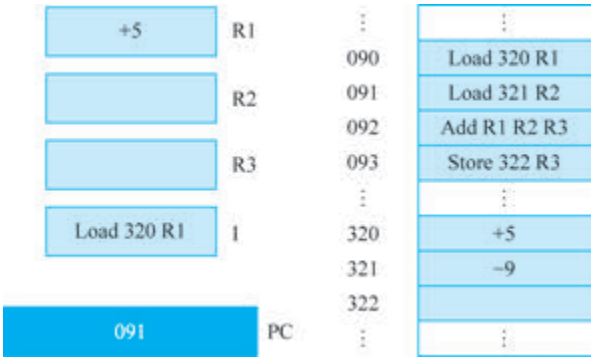

图 5.9　执行 Load 320 R1 的结果

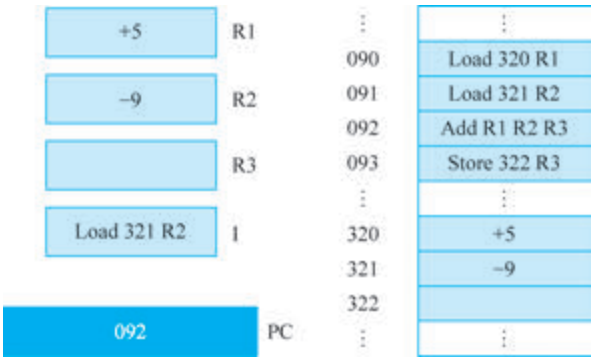

图 5.10　执行 Load 321 R2 的结果

在第三个机器周期，地址 092 中的指令被存入指令寄存器 I，PC 加 1，算术逻辑单元执行 I 中的指令 Add R1 R2 R3，将 R1 中的数据＋5 和 R2 中的数据－9 相加，得到结果－4，将其存入 R3 中，本轮机器周期结束，最后的结果如图 5.11 所示。

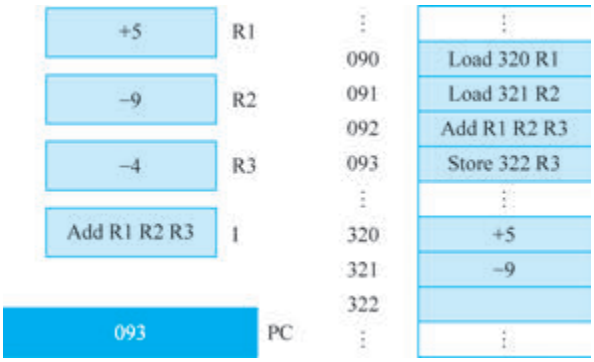

图 5.11　执行 Add R1 R2 R3 的结果

在第四个机器周期，地址 093 中的指令被存入指令寄存器 I，PC 加 1，算术逻辑单元执行 I 中的指令 Store 322 R3，将 R3 中的数据存入地址为 322 的存储单元，此次机器周

期结束。四个机器周期结束后，加法程序运行完毕，最终的结果如图 5.12 所示。

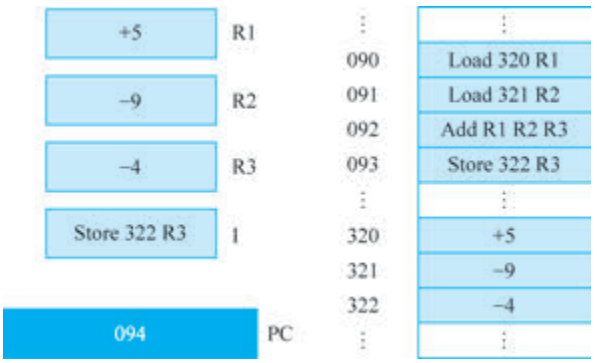

图 5.12　加法程序结果

可以看出，这就是一个简单的加法程序，计算 5＋(－9)＝－4，计算结果存在存储单元 322。在 C++编程中，如果已经创建两个变量 i 和 j，并令 i 等于＋5，j 等于－9，那么让计算机执行 $i+j$ 这一程序时，计算机必须循环以上四个机器周期才能完成。

5.4 并行架构

前面所学习的计算机结构都是一些比较基本的结构，最明显的特征就是一台计算机中只有一个处理器(类似于只有一个算术逻辑单元或一个控制单元)，但在实际生活应用中，只有一个处理器的计算机往往不能够满足人们的需求。因此，为了提高运行和计算的效率，现在的计算机往往会采取并行架构，即一个计算机中具有多个处理器，如此一来，一台计算机在同一时间可以处理多个任务。

并行架构(parallel architectures)的提出目的是在同一时间一台计算机可以执行多个任务，以此来提高计算机的运行速度。拥有并行结构的计算机一般有多个处理器，可以同时执行多个任务或处理多个数据，这大大减少了机器执行任务时所需要执行的机器周期，加快了计算机的计算和操作。最常见的并行架构如图 5.13 所示。可以看到，这台机器上有多个处理器，每个处理器有自己的本地存储空间，这些存储空间之间都是相互独立、互不干扰的；同时，这些处理器一定会有一个共享内存，实现相互之间的沟通。在大部分时间内，各个处理器都在相互独立的工作，将运行得到的数据结果存储到自身独立的本地存储空间中；必要时，各个处理器之间会通过共享内存进行数据的共享和通信，这也称为并行构架的共享内存模式。

就并行的力度而言，并行架构可以分为四种：比特级别的并行，指令级别的并行，数据级别的并行以及任务级别的并行。

(1) **比特级别的并行(bit-level parallelism)**是一种极为常见的并行架构，并且几乎所有的机器都在采用。这种并行模式主要是通过增加计算机的机器字长来实现并行的，最终目的是减少计算机运行时的机器周期的数目。对较大的数据进行计算时，对于没有采用比特级别的并行的计算机，它需要将这个数据分成多份，进行多次机器周期才能够得到最终结果；但对于采用比特级别的并行的计算机，它可能就不需要把数据拆分，或将数

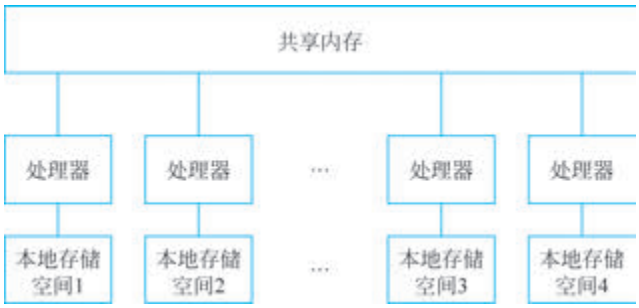

图 5.13　共享内存模式的并行架构

据分为较少的份，进行一次或少数次机器周期就能得到最终结果。这样一来，计算机所要执行的机器周期数就大大减少。举个例子，如果是机器字长为 8 位的计算机，那么该计算机对两个 16 位的数据进行加法运算时，就需要将两个数据分别分成两份，并按先后顺序进行两次加法运算、执行两个机器周期才能得到结果；但如果把该计算机的机器字长增长到 16 位，那么计算机只要进行一次加法运算、执行一个机器周期就能得到最终结果，如此就可以达到减少机器周期循环数的目的。

(2) **指令级别的并行(instruction-level parallelism)**是指对程序中的部分指令同时执行。前面所提到的程序都是按先后顺序一个个执行其中的指令，但其实很多程序中存在着大量相互独立、彼此之间运行结果互不影响的指令，也就是说，在这些指令之间，无论是谁先运行得到结果，或谁没有运行且结果未知，对其他指令都没有影响，所以，这些指令完全可以同时运行，从而大大减少程序运行的时间，这就是指令级别的并行运行的原理。举个例子，计算下述三个算式：

$$a = 1 + 2 \tag{5.1}$$

$$b = 3 + 4 \tag{5.2}$$

$$c = a + b \tag{5.3}$$

如果按照先前惯例，按顺序进行式(5.1)～式(5.3)的计算，则需要 3 个机器周期；但其实不难发现，式(5.1)和式(5.2)是相互独立的，二者无论谁先进行计算都不会影响另一个算式的计算，因此，这两个算式完全可以同时计算，这样一来计算机的计算速度就得到了提高，只需要 2 个机器周期。现在有一种称为超标量机(superscalar)的处理器，能够自动识别可以同时进行计算的场景，并把这些运算分配给不同的算术逻辑单元进行计算。

(3) **数据级别的并行(data-level parallelism)**通常称为 SIMD(single instructions, multiple data)，这种并行是对不同的数据执行同一组指令集操作，即通过不同的处理器对大量数据进行相同的操作。矩阵的乘法可以采取这种并行架构，以下面的矩阵乘法为例：

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 3 & 2 \end{pmatrix} \begin{pmatrix} 10 & 11 \\ 7 & 5 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} 1\times10+2\times7+3\times2 & 1\times11+2\times5+3\times4 \\ 4\times10+5\times7+6\times2 & 4\times11+5\times5+6\times4 \\ 1\times10+3\times7+2\times2 & 1\times11+3\times5+2\times4 \end{pmatrix}$$

如果串行执行，最终结果要通过六次向量点乘计算才可以得到。但这六步是一样的操作，而且它们之间互不影响，完全可以通过六个运算单元把它们同时计算出来，大大提高了计算速度。现在的图形处理器单元（graphics processing unit，GPU）采用了 SIMD 结构，使用大量简单的处理单元同时执行相同的指令，如对图形中每个像素点都进行调高亮度的操作。

知识闲谈

图形处理单元（GPU）是一种具有强大的 SIMD 计算能力的处理器，它能够同时执行相同指令的多个线程，每个线程操作不同的数据。

在 GPU 中，SIMD 被广泛应用于图形渲染和通用计算任务。GPU 中的流处理器（也称为 CUDA 核心）以 SIMD 方式工作，它们组成了 GPU 中的处理单元数组。每个流处理器可以同时执行相同的指令，但在不同的数据上进行操作，实现了数据级并行性。

GPU 中的 SIMD 架构允许同时处理多个像素、顶点或其他图形数据元素。在图形渲染中，如绘制三角形时，GPU 可以通过并行处理每个顶点的坐标变换、光照计算和纹理映射等操作来加快图形渲染的速度。通过在单个指令中同时处理多个数据元素，GPU 能够实现高效的并行计算，并在实时图形渲染中提供流畅的帧率。

除了图形渲染，GPU 还广泛应用于通用计算领域。通过将通用计算任务转换为 SIMD 形式，GPU 能够在处理大规模数据集时提供显著的加速效果。例如，在机器学习和数据科学中，GPU 可用于加速矩阵运算、向量操作和神经网络计算等密集的数值计算任务。GPU 的并行计算能力使得它在处理大规模数据集和复杂计算任务时能够显著提高计算效率。

为了利用 GPU 的 SIMD 计算能力，开发人员使用并行计算框架（如 CUDA 和 OpenCL）编写代码，并将计算任务划分为适合 SIMD 并行处理的任务单元。通过合理地设计算法和数据结构，以及利用 GPU 的并行计算特性，开发人员能够实现高性能的并行计算，并将任务分发到 GPU 的多个流处理器上同时执行。

（4）**任务级别的并行（task-level parallelism）**是通过不同的处理器，对相同或不同的数据执行不同的任务。任务级别的并行类似于流水线。工厂在进行零件加工时，一个零件往往要经过多个阶段的加工才能成为成品，每个零件由一个阶段的工人进行加工后交给下一个阶段的工人进行加工，每一个阶段的工人都在进行各自阶段的零件加工工作，并且处理的零件都是由上一个阶段的工人加工过的。任务级别的并行就类似这样一个过程。最开始由第一个处理器处理一个任务，并将处理得到的结果交给第二个处理器进行处理，当第二个处理器处理由第一个处理器传递过来的结果时，第一个处理器将会处理下一个任务。因此，在每个时间段，不同的处理器都在执行不同的任务，处理来自上一个处理器得到的结果。

现代计算机系统通常采用多种并行架构的组合，以充分利用不同的硬件资源和优势。这种综合使用并行架构的方式可以提供更高的计算性能、更好的资源利用率和更广

泛的应用适用性。

当机器采用并行计算时，需要考虑一系列的问题，其中一个重要问题是数据竞争。如图 5.14 所示，例如，假设处理器 A 和处理器 B 分别进行两个进程，两个处理器同时需要文件 1 和文件 2；如果处理器 A 先分配到文件 1，再申请文件 2，而处理器 B 先分配到文件 2，再申请文件 1，在两者互不相让的情况下，谁都无法得到全部需要的文件，也无法进行下一步操作，这时就会出现死锁，此时，计算机运行的程序会发生堵塞，无法继续正常进行。这就好比警匪两方对峙，罪犯劫持一名无辜群众作为人质，而警方也抓获了罪犯的同伙，此时双方都要求对方先释放手中的人质，如果没有一个协商和妥协的方式来解决困局，双方只能长时间僵持。数据竞争可能导致不确定的结果或程序崩溃。为了解决这个问题，需要使用同步机制，如锁、互斥量或原子操作来保护共享数据的访问。

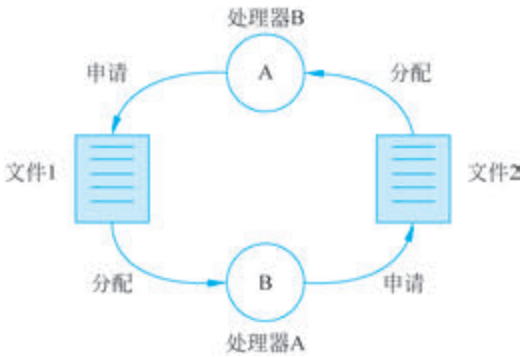

图 5.14　并行计算中的数据竞争问题

最后，我们简单讲述**分布式计算（distributed computing）**的概念。分布式计算的出现主要是为了解决处理大规模数据和复杂计算问题时的挑战。随着数据量的不断增长和计算需求的日益复杂，单个计算机系统的处理能力可能会受限。分布式计算利用多个计算节点或计算机集群中的资源来共同完成大规模的计算任务。在分布式计算中，任务被分解成多个子任务，这些子任务可以同时在不同的计算节点上进行并行处理，然后将计算结果进行合并以得到最终的结果。在分布式计算中，计算节点可以是一组相互连接的计算机，也可以是云计算平台上的虚拟机实例。这些节点通过网络进行通信和协作，彼此之间可以传输数据、共享计算资源和进行任务调度。分布式计算系统通常包括任务调度器、数据分发器、通信协议和容错机制等组件，以确保任务能够正确地分发、执行和完成。

分布式计算与并行计算的概念有一些重叠的地方，但它们具有本质的区别，即是否共享内存。在并行计算中，多个处理单元共享同一块内存，它们可以直接读取和写入内存中的数据，从而实现数据的共享和通信。这种共享内存的模型可以更方便地进行数据交换和协作计算，适用于需要频繁地共享数据和通信的任务。而分布式计算则强调在不同的计算节点之间进行任务的分布和协作，每个节点拥有自己的独立内存，并通过网络进行通信和数据交换。每个节点可以独立地执行任务，相互之间并不直接共享内存。分布式计算通常用于解决大规模计算和数据处理问题，可以通过增加计算节点来扩展计算

能力。因此，可以将并行计算看作是在共享内存的体系结构中进行的计算，而分布式计算则是在分布式环境下进行的计算。并行计算更注重于内存共享和通信的效率，而分布式计算则更注重于任务的划分和分布、节点间的通信和数据传输的效率。

注意，并行计算和分布式计算并不是互斥的概念，实际上很多并行计算系统也可以通过网络连接形成分布式计算环境。此外，并行计算和分布式计算可以结合使用，如在一个大规模的分布式集群系统中，每个节点可以是一个并行计算系统，从而实现更高的计算性能和可扩展性。

本章小结

本章对计算机中各个组件的结构和功能进行了静态学习，并在机器周期的动态运行中进一步了解了各个组件的功能以及它们之间的相互联系和配合。以下是本章内容的总结：

首先，深入学习了冯·诺依曼模型的关键组件，包括中央处理器、存储器和输入/输出设备。中央处理器由控制单元和算术逻辑单元组成，是计算机的核心。冯·诺依曼模型的特点是存储程序和指令按顺序执行。

其次，详细介绍了计算机主要组件的结构和功能。将计算机的主要组件分为四部分进行了讲解，包括中央处理器、存储器、总线和外部设备。中央处理器由三个主要结构组成，其中算术逻辑单元负责核心计算，控制单元负责控制和协调各个部件，而寄存器则是中央处理器中临时存放数据的地方。除了寄存器，计算机内部的存储器还包括主存储器和高速缓冲存储器，它们的存储空间大小和数据获取速度各不相同。此外，存储器还可以细分为随机存取存储器(RAM)和只读存储器(ROM)。总线作为计算机内部信息传输的通道，根据传输信息的种类分为数据总线、地址总线和控制总线，分别用于传送数据、地址和指令。外部设备有许多种类，它们通过接口连接到总线上。本章重点介绍的外部存储器有磁盘、光盘和闪存等。

接下来，对机器周期进行了详细学习，了解了一个机器周期的四个基本步骤，包括取指、解码、取值和执行。通过这些步骤的循环执行，计算机能够实现程序的执行。

最后，进一步扩展了单处理器计算机的概念，介绍了多处理器计算机和并行架构。初步了解了比特级别的并行、指令级别的并行、数据级别的并行和任务级别的并行这四种并行架构的运行原理，并简要介绍了分布式计算的概念。分布式计算是基于多台计算机通过网络相互协作完成任务的一种计算模式，实现了资源共享和任务分担。

通过本章的学习，我们对计算机的结构和运行原理有了更深入的了解。我们理解了计算机各个组件的功能和相互关系，掌握了机器周期的运行过程以及并行计算的原理和应用。这些知识对于理解计算机的工作方式和优化程序性能等方面非常重要。

思考与练习

1. 中央处理器中用于存储地址的寄存器称为_____。

2. _____是计算机中传送信息的公共通道。

3. 下列存储器中，(　　)获取数据的速度最快。

A. 寄存器　　B. 高速缓冲存储器

C. 闪存　　D. 主存储器

4. (　　)通过增加字长实现减少机器周期的目的。

A. 比特级别的并行　　B. 数据级别的并行

C. 指令级别的并行　　D. 任务级别的并行

5. 下列不属于计算机外部存储设备的是(　　)。

A. CD　　B. U 盘　　C. 硬盘　　D. 主存储器

6. RAM 中存储的数据是易丢失的，这句话的描述是(　　)的。

A. 正确　　B. 错误

7. 如果一台计算机的内存大小为 512 MB，且这台计算机的存储字长是 8 B。那么这台计算机的地址长度是多少？

8. 假设一台虚拟计算机有四个数据寄存器(记为 R0～R3)，主存储器中有 65536 个存储单元，64 种不同的指令(Add、Subtract 等)。一条典型的指令是如下格式：Add 565 R2(注意，565 在这里是计算机的一个地址)，试回答：

(1) 一条指令的最小长度是多少位？

(2) 这台计算机每个指令寄存器的大小是多少？

(3) 如果流入和流出算术逻辑单元的信息长度为 16 位，那么每个数据寄存器的大小是多少？

(4) 程序计数器的大小是多少？

9. 请简述一个机器周期的主要步骤。

10. 说出冯·诺依曼体系结构各个组件的名称和功能。