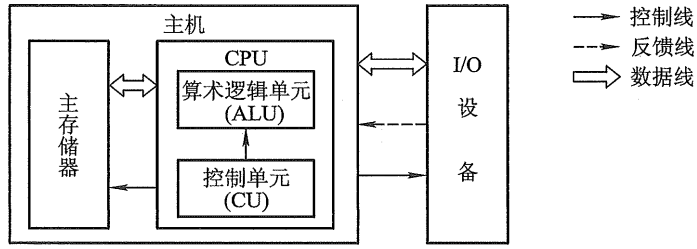

图 1.9 现代计算机的组成框图

图 1.9 中的主存储器是存储器子系统中的一类,用来存放程序和数据,可以直接与 CPU 交换信息。另一类称为辅助存储器,简称辅存,又称外存,其功能参阅 4.4 节。

算术逻辑单元(Arithmetic Logic Unit,ALU)简称算逻部件,用来完成算术逻辑运算。控制单元(Control Unit,CU)用来解释存储器中的指令,并发出各种操作命令来执行指令。ALU 和 CU 是 CPU 的核心部件。

I/O 设备也受 CU 控制,用来完成相应的输入、输出操作。

可见,计算机有条不紊地自动工作都是在控制器统一指挥下完成的。

1.2.3 计算机的工作步骤

用计算机解决一个实际问题通常包含两大步骤。一个是上机前的各种准备,另一个是上机运行。

1. 上机前的准备

在许多科学技术的实际问题中,往往会遇到许多复杂的数学方程组,而数字计算机通常只能执行加、减、乘、除四则运算,这就要求在上机解题前,先由人工完成一些必要的准备工作。这些工作大致可归纳为:建立数学模型、确定计算方法和编制解题程序 3 个步骤。

(1) 建立数学模型

有许多科技问题很难直接用物理模型来模拟被研究对象的变化规律,如地球大气环流、原子反应堆的核裂变过程、航天飞行速度对飞行器的影响等。不过,通过大量的实验和分析,总能找到一系列反映研究对象变化规律的数学方程组。通常,将这类方程组称为被研究对象变化规律的数学模型。一旦建立了数学模型,研究对象的变化规律就变成了解一系列方程组的数学问题,这便可通过计算机来求解。因此,建立数学模型是用计算机解题的第一步。

(2) 确定计算方法

由于数学模型中的数学方程式往往是很复杂的,欲将其变成适合计算机运算的加、减、乘、除四则运算,还必须确定对应的计算方法。

例如,欲求 $\sin x$ 的值,只能采用近似计算方法,用四则运算的式子来求得(因计算机内部没

有直接完成三角函数运算的部件)。

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots$$

又如,计算机不能直接求解开方 x,但可用迭代公式:

$$y_{n+1} = \sqrt{x} = \frac{1}{2}\left(y_n + \frac{x}{y_n}\right) \quad (n = 0, 1, 2, \cdots)$$

通过多次迭代,便可求得相应精度的$\sqrt{x}$值。

(3) 编制解题程序

程序是适合于机器运算的全部步骤,编制解题程序就是将运算步骤用一一对应的机器指令描述。

例如,计算 ax^2+bx+c 可分解为以下步骤。

① 将 x 取至运算器中。

② 乘以 x,得 x^2,存于运算器中。

③ 再乘以 a,得 ax^2,存于运算器中。

④ 将 ax^2 送至存储器中。

⑤ 取 b 至运算器中。

⑥ 乘以 x,得 bx,存于运算器中。

⑦ 将 ax^2 从存储器中取出与 bx 相加,得 ax^2+bx,存于运算器中。

⑧ 再取 c 与 ax^2+bx 相加,得 ax^2+bx+c,存于运算器中。

可见,不包括停机、输出打印共需 8 步。若将上式改写成:$(ax+b)x+c$,则其步骤可简化为以下 5 步。

① 将 x 取至运算器中。

② 乘以 a,得 ax,存于运算器中。

③ 加 b,得 $ax+b$,存于运算器中。

④ 乘以 x,得$(ax+b)x$,存于运算器中。

⑤ 加 c,得$(ax+b)x+c$,存于运算器中。

将上述运算步骤写成某计算机一一对应的机器指令,就完成了运算程序的编写。

设某机的指令字长为 16 位,其中操作码占 6 位,地址码占 10 位,如图 1.10 所示。

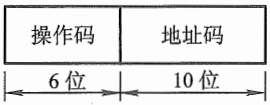

图 1.10　某机器指令格式

操作码表示机器所执行的各种操作,如取数、存数、加、减、乘、除、停机、打印等。地址码表示参加运算的数在存储器内的位置。机器指令的操作码和地址码都采用 0、1 代码的组合来表示。表 1.1 列出了某机与上例有关的各条机器指令的操作码及其操作性质的对应关系。

表 1.1 操作码与操作性质的对应表

操作码	操作性质	具体内容
000001	取数	将指令地址码指示的存储单元中的操作数取到运算器的累加器 ACC 中
000010	存数	将 ACC 中的数存至指令地址码指示的存储单元中
000011	加	将 ACC 中的数与指令地址码指示的存储单元中的数相加,结果存于 ACC 中
000100	乘	将 ACC 中的数与指令地址码指示的存储单元中的数相乘,结果存于 ACC 中
000101	打印	将指令地址码指示的存储单元中的操作数打印输出
000110	停机	

此例中所用到的数 a、b、c、x,事先需存入存储器的相应单元内。

按 ax^2+bx+c 的运算分解,可用上述机器指令编写出一份运算的程序清单,如表 1.2 所列。

表 1.2 计算 ax^2+bx+c 程序清单

指令和数据存于主存单元的地址	指令		注释
	操作码	地址码	
0	000001	0000001000	取数 x 至 ACC
1	000100	0000001001	乘 a 得 ax,存于 ACC 中
2	000011	0000001010	加 b 得 $ax+b$,存于 ACC 中
3	000100	0000001000	乘 x 得 $(ax+b)x$,存于 ACC 中
4	000011	0000001011	加 c 得 ax^2+bx+c,存于 ACC 中
5	000010	0000001100	存数,将 ax^2+bx+c 存于主存单元
6	000101	0000001100	打印
7	000110		停机
8	x		原始数据 x
9	a		原始数据 a
10	b		原始数据 b
11	c		原始数据 c
12			存放结果

以上程序编完后,便可进入下一步上机。

2. 计算机的工作过程

为了比较形象地了解计算机的工作过程,首先分析一个比图 1.9 更细化的计算机组成框图,如图 1.11 所示。

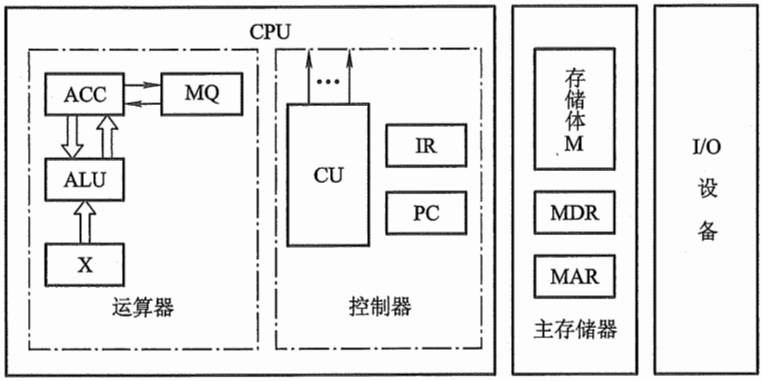

图 1.11　细化的计算机组成框图

(1) 主存储器

主存储器(简称主存或内存)包括存储体 M、各种逻辑部件及控制电路等。存储体由许多存储单元组成,每个存储单元又包含若干个存储元件(或称存储基元、存储元),每个存储元件能寄存一位二进制代码“0”或“1”。可见,一个存储单元可存储一串二进制代码,称这串二进制代码为一个存储字,这串二进制代码的位数称为存储字长。存储字长可以是 8 位、16 位或 32 位等。一个存储字可代表一个二进制数,也可代表一串字符,如存储字为 0011011001111101,既可表示为由十六进制字符组成的 367DH(有关十六进制数制详见附录 6A),又可代表 16 位的二进制数,此值对应十进制数为 13 949,还可代表两个 ASCII 码:“6”和“}”(参见附录 5A ASCII 编码表)。一个存储字还可代表一条指令(参阅表 1.2)。

如果把一个存储体看作一幢大楼,那么每个存储单元可看作大楼中的每个房间,每个存储元可看作每个房间中的一张床位,床位有人相当于“1”,无人相当于“0”。床位数相当于存储字长。显然,每个房间都需要有一个房间编号,同样可以赋予每个存储单元一个编号,称为存储单元的地址号。

主存的工作方式就是按存储单元的地址号来实现对存储字各位的存(写入)、取(读出)。这种存取方式称为按地址存取方式,即按地址访问存储器(简称访存)。存储器的这种工作性质对计算机的组成和操作是十分有利的。例如,人们只要事先将编好的程序按顺序存入主存各单元,当运行程序时,先给出该程序在主存的首地址,然后采用程序计数器加 1 的方法,自动形成下一条指令所在存储单元的地址,机器便可自动完成整个程序的操作。又如,由于数据和指令都存放在存储体内各自所占用的不同单元中,因此,当需要反复使用某个数据或某条指令时,只要指出其相应的单元地址号即可,而不必占用更多的存储单元重复存放同一个数据或同一条指令,大大提高了存储空间

的利用率。此外,由于指令和数据都由存储单元地址号来反映,因此,取一条指令和取一个数据的操作完全可视为是相同的,这样就可使用一套控制线路来完成两种截然不同的操作。

为了能实现按地址访问的方式,主存中还必须配置两个寄存器 MAR 和 MDR。MAR(Memory Address Register)是存储器地址寄存器,用来存放欲访问的存储单元的地址,其位数对应存储单元的个数(如 MAR 为 10 位,则有 $2^{10}=1\ 024$ 个存储单元,记为 1 K)。MDR(Memory Data Register)是存储器数据寄存器,用来存放从存储体某单元取出的代码或者准备往某存储单元存入的代码,其位数与存储字长相等。当然,要想完整地完成一个取或存操作,CPU 还得给主存加以各种控制信号,如读命令、写命令和地址译码驱动信号等。随着硬件技术的发展,主存都制成大规模集成电路的芯片,而将 MAR 和 MDR 集成在 CPU 芯片中(参阅图 4.5)。

早期计算机的存储字长一般和机器的指令字长与数据字长相等,故访问一次主存便可取一条指令或一个数据。随着计算机应用范围的不断扩大,解题精度的不断提高,往往要求指令字长是可变的,数据字长也要求可变。为了适应指令和数据字长的可变性,其长度不由存储字长来确定,而由字节的个数来表示。1 个字节(Byte)被定义为由 8 位(bit)二进制代码组成。例如,4 字节数据就是 32 位二进制代码;2 字节构成的指令字长是 16 位二进制代码。当然,此时存储字长、指令字长、数据字长三者可各不相同,但它们必须是字节的整数倍。

(2) 运算器

运算器最少包括 3 个寄存器(现代计算机内部往往设有通用寄存器组)和一个算术逻辑单元(ALU)。其中 ACC(Accumulator)为累加器,MQ(Multiplier-Quotient Register)为乘商寄存器,X 为操作数寄存器。这 3 个寄存器在完成不同运算时,所存放的操作数类别也各不相同。表 1.3 列出了寄存器存放不同类别操作数的情况。

表 1.3 各寄存器所存放的各类操作数

运算 / 操作数 / 寄存器	加法	减法	乘法	除法
ACC	被加数及和	被减数及差	乘积高位	被除数及余数
MQ			乘数及乘积低位	商
X	加数	减数	被乘数	除数

不同机器的运算器结构是不同的。图 1.11 所示的运算器可将运算结果从 ACC 送至存储器中的 MDR;而存储器的操作数也可从 MDR 送至运算器中的 ACC、MQ 或 X。有的机器用 MDR 取代 X 寄存器。

下面简要地分析一下这种结构的运算器加、减、乘、除四则运算的操作过程。

设:M 表示存储器的任一地址号,[M]表示对应 M 地址号单元中的内容;X 表示 X 寄存器,[X]表示 X 寄存器中的内容;ACC 表示累加器,[ACC]表示累加器中的内容;MQ 表示乘商寄存

器，[MQ]表示乘商寄存器中的内容。

假设 ACC 中已存有前一时刻的运算结果，并作为下述运算中的一个操作数，则

- 加法操作过程为

[M]→X

[ACC]+[X]→ACC

即将[ACC]看作被加数，先从主存中取一个存放在 M 地址号单元内的加数[M]，送至运算器的 X 寄存器中，然后将被加数 [ACC] 与加数 [X] 相加，结果(和)保留在 ACC 中。

- 减法操作过程为

[M]→X

[ACC]-[X]→ACC

即将[ACC]看作被减数，先取出存放在主存 M 地址号单元中的减数 [M] 并送入 X，然后[ACC]-[X]，结果(差)保留在 ACC 中。

- 乘法操作过程为

[M]→MQ

[ACC]→X

0→ACC

[X]×[MQ]→ACC//MQ[①]

即将[ACC]看作被乘数，先取出存放在主存 M 号地址单元中的乘数 [M] 并送入乘商寄存器 MQ，再把被乘数送入 X 寄存器，并将 ACC 清“0”，然后 [X] 和 [MQ] 相乘，结果(积)的高位保留在 ACC 中，低位保留在 MQ 中。

- 除法操作过程为

[M]→X

[ACC]÷[X]→MQ

余数 R 在 ACC 中

即将 [ACC]看作被除数，先取出存放在主存 M 号地址单元内的除数 [M] 并送至 X 寄存器，然后 [ACC] 除以 [X]，结果(商)暂留于 MQ，[ACC] 为余数 R。若需要将商保留在 ACC 中，只需做一步 [MQ]→ACC 即可。

(3) 控制器

控制器是计算机的神经中枢，由它指挥各部件自动、协调地工作。具体而言，它首先要命令存储器读出一条指令，称为取指过程(也称取指阶段)。接着，它要对这条指令进行分析，指出该指令要完成什么样的操作，并按寻址特征指明操作数的地址，称为分析过程(也称分析阶段)。最后根据操作数所在的地址以及指令的操作码完成某种操作，称为执行过程(也称执行阶段)。

① //表示两个寄存器串接。

以上就是通常所说的完成一条指令操作的取指、分析和执行3个阶段。

控制器由程序计数器(Program Counter,PC)、指令寄存器(Instruction Register,IR)以及控制单元(CU)组成。PC用来存放当前欲执行指令的地址,它与主存的MAR之间有一条直接通路,且具有自动加1的功能,即可自动形成下一条指令的地址。IR用来存放当前的指令,IR的内容来自主存的MDR。IR中的操作码(OP(IR))送至CU,记作OP(IR)→CU,用来分析指令;其地址码(Ad(IR))作为操作数的地址送至存储器的MAR,记作Ad(IR)→MAR。CU用来分析当前指令所需完成的操作,并发出各种微操作命令序列,用以控制所有被控对象。

(4) I/O

I/O子系统包括各种I/O设备及其相应的接口。每一种I/O设备都由I/O接口与主机联系,它接收CU发出的各种控制命令,并完成相应的操作。例如,键盘(输入设备)由键盘接口电路与主机联系;打印机(输出设备)由打印机接口电路与主机联系。

下面结合图1.11进一步深入领会计算机工作的全过程。

首先按表1.2所列的有序指令和数据,通过键盘输入到主存第0号至第12号单元中,并置PC的初值为0(令程序的首地址为0)。启动机器后,计算机便自动按存储器中所存放的指令顺序有序地逐条完成取指令、分析指令和执行指令,直至执行到程序的最后一条指令为止。

例如,启动机器后,控制器立即将PC的内容送至主存的MAR(记作PC→MAR),并命令存储器做读操作,此刻主存"0"号单元的内容"0000010000001000"(表1.2所列程序的第一条指令)便被送入MDR内。然后由MDR送至控制器的IR(记作MDR→IR),完成了一条指令的取指过程。经CU分析(记作OP(IR)→CU),操作码"000001"为取数指令,于是CU又将IR中的地址码"0000001000"送至MAR(记作Ad(IR)→MAR),并命令存储器做读操作,将该地址单元中的操作数x送至MDR,再由MDR送至运算器的ACC(记作MDR→ACC),完成此指令的执行过程。此刻,也即完成了第一条取数指令的全过程,即将操作数x送至运算器ACC中。与此同时,PC完成自动加1的操作,形成下一条指令的地址"1"号。同上所述,由PC将第二条指令的地址送至MAR,命令存储器做读操作,将"0001000000001001"送入MDR,又由MDR送至IR。接着CU分析操作码"000100"为乘法指令,故CU向存储器发出读命令,取出对应地址为"0000001001"单元中的操作数a,经MDR送至运算器MQ,CU再向运算器发送乘法操作命令,完成ax的运算,并把运算结果ax存放在ACC中。同时PC又完成一次(PC)+1→PC,形成下一条指令的地址"2"号。依次类推,逐条取指、分析、执行,直至打印出结果。最后执行完停机指令后,机器便自动停机。

1.3 计算机硬件的主要技术指标

衡量一台计算机性能的优劣是根据多项技术指标综合确定的。其中,既包含硬件的各种性能指标,又包括软件的各种功能。这里主要讨论硬件的技术指标。

1.3.1　机器字长

机器字长是指 CPU 一次能处理数据的位数，通常与 CPU 的寄存器位数有关。字长越长，数的表示范围越大，精度也越高。机器的字长也会影响机器的运算速度。倘若 CPU 字长较短，又要运算位数较多的数据，那么需要经过两次或多次的运算才能完成，这样势必影响机器的运算速度。

机器字长对硬件的造价也有较大的影响。它将直接影响加法器（或 ALU）、数据总线以及存储字长的位数。所以机器字长的确定不能单从精度和数的表示范围来考虑。

1.3.2　存储容量

存储器的容量应该包括主存容量和辅存容量。

主存容量是指主存中存放二进制代码的总位数。即

存储容量=存储单元个数×存储字长

图 1.11 中 MAR 的位数反映了存储单元的个数，MDR 的位数反映了存储字长。例如，MAR 为 16 位，根据 $2^{16}=65\ 536$，表示此存储体内有 65 536 个存储单元（即 64 K 个存储字，1 K=1 024=2^{10}）；而 MDR 为 32 位，表示存储容量为 $2^{16}\times32=2^{21}=2$ M 位（1 M=2^{20}）。

现代计算机中常以字节数来描述容量的大小，因一个字节已被定义为 8 位二进制代码，故用字节数便能反映主存容量。例如，上述存储容量为 2 M 位，也可用 2^{18} 字节表示，记作 2^{18} B 或 256 KB（B 用来表示一个字节）。

辅存容量通常用字节数来表示，例如，某机辅存（如硬盘）容量为 80 GB（1 G=1 024 M=$2^{10}\times2^{20}=2^{30}$）。

1.3.3　运算速度

计算机的运算速度与许多因素有关，如机器的主频、执行什么样的操作、主存本身的速度（主存速度快，取指、取数就快）等都有关。早期用完成一次加法或乘法所需的时间来衡量运算速度，即普通法，显然是很不合理的。后来采用吉普森（Gibson）法，它综合考虑每条指令的执行时间以及它们在全部操作中所占的百分比，即

$$T_M = \sum_{i=1}^{n} f_i t_i$$

其中，T_M 为机器运行速度；f_i 为第 i 种指令占全部操作的百分比数；t_i 为第 i 种指令的执行时间。

现在机器的运算速度普遍采用单位时间内执行指令的平均条数来衡量，并用 MIPS（Million Instruction Per Second，百万条指令每秒）作为计量单位。例如，某机每秒能执行 200 万条指令，则记作 2 MIPS。也可以用 CPI（Cycle Per Instruction）即执行一条指令所需的时钟周期（机器主频的倒数）

数,或用 FLOPS(Floating Point Operation Per Second,浮点运算次数每秒)来衡量运算速度。

1.4 本书结构

本书介绍计算机组成原理,其内容安排如下:

第 1 篇:概论,介绍计算机系统的基本组成、应用与发展。

第 2 篇:计算机系统的硬件结构,引导读者自顶向下了解计算机系统的硬件结构,包括中央处理器、存储器、I/O 等主要部件以及连接它们的系统总线。其中,除中央处理器比较复杂放在第 3 篇单独讲述外,其他各部件均在此篇介绍。

第 3 篇:中央处理器(CPU),本篇讲述 CPU 的功能和结构,并对影响 CPU 特性、结构和功能的算逻单元及其运算方法、指令系统、指令流水、中断系统等进行详细分析。有关控制单元(CU)在第 4 篇单独介绍。

第 4 篇:控制单元(CU),本篇在详细分析时序系统以及微操作命令节拍安排的基础上,分别介绍如何用组合逻辑控制及微程序控制两种方法设计和实现控制单元。

总之,全书按自顶向下、由表及里的层次结构,向读者展示计算机的组成及其工作原理,目的是使读者能先从整体上对计算机有一个粗略的认识,然后,逐步深入到机器内核,从而更容易形成计算机的整体概念。图 1.12 形象地描述了上述各章节之间的联系。

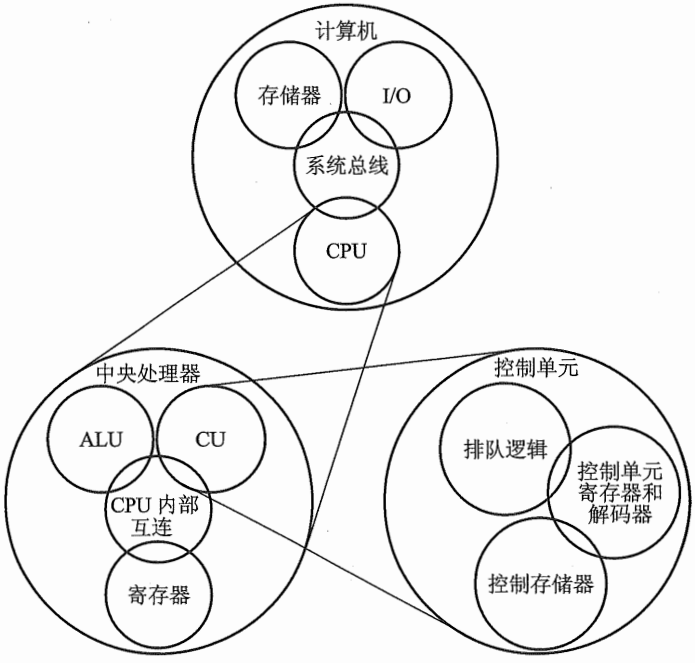

图 1.12 全书各章节之间的关系

思考题与习题

1.1　什么是计算机系统、计算机硬件和计算机软件？硬件和软件哪个更重要？

1.2　如何理解计算机系统的层次结构？

1.3　说明高级语言、汇编语言和机器语言的差别及其联系。

1.4　如何理解计算机组成和计算机体系结构？

1.5　冯·诺依曼计算机的特点是什么？

1.6　画出计算机硬件组成框图，说明各部件的作用及计算机硬件的主要技术指标。

1.7　解释概念：主机、CPU、主存、存储单元、存储元件、存储基元、存储元、存储字、存储字长、存储容量、机器字长、指令字长。

1.8　解释英文代号：CPU、PC、IR、CU、ALU、ACC、MQ、X、MAR、MDR、I/O、MIPS、CPI、FLOPS。

1.9　画出主机框图，分别以存数指令“STA M”和加法指令“ADD M”（M 均为主存地址）为例，在图中按序标出完成该指令（包括取指阶段）的信息流程（如$\xrightarrow{①}$）。假设主存容量为 256 M×32 位，在指令字长、存储字长、机器字长相等的条件下，指出图中各寄存器的位数。

1.10　根据迭代公式$\sqrt{x}=\frac{1}{2}\left(y_n+\frac{x}{y_n}\right)$，设初态 $y_0=1$，要求精度为 ε，试编制求$\sqrt{x}$的解题程序（指令系统自定），并结合所编程序简述计算机的解题过程。

1.11　指令和数据都存于存储器中，计算机如何区分它们？

1.12　什么是指令？什么是程序？

第 2 章　计算机的发展及应用

本章简要介绍计算机的发展史以及它的应用领域，旨在使读者对计算机有一个感性的认识。最后，本章展望计算机的未来。

2.1　计算机的发展史

谁也不曾想到，当初只是当作军事计算工具应用的电子计算机，在半个世纪中竟然会成为改变社会结构，乃至促使人们的工作和生活方式发生惊人变化的宠儿，真可谓 20 世纪下半世纪科技发展最有影响的发明，并且它还将继续影响着未来世界的变化，使数千年人类文明史中曾有过的各种神话般的幻想逐渐变为现实。

2.1.1　计算机的产生和发展

1. 第一代电子管计算机

1943 年，第二次世界大战进入后期，因战争的需要，美国国防部批准了由 Pennsyivania 大学 John Mauchly 教授和 John Presper Eckert 工程师提出的建造一台用电子管组成的电子数字积分机和计算机（Electronic Numerical Integrator And Computer，ENIAC）的计划，用它来解决当时国防部弹道研究实验室（BRL）开发新武器的射程和检测模拟运算表的难题。当时，由于运算能力不足，该实验室无法在规定的时间内拿出准确的运算表，严重影响了新武器的制作。

ENIAC 于 1946 年交付使用，其首要任务就是完成了一系列测定氢弹可靠性的复杂运算。ENIAC 采用十进制运算，电路结构十分复杂，使用 18 000 多个电子管，运行时耗电量达 150 千瓦，体积庞大，重量达 30 吨，占地面积为 1 500 平方英尺，而且需用手工搬动开关和拔、插电缆来编制程序，使用极不方便，但它比任何机械计算机快得多，每秒可进行 5 000 多次加法运算。

ENIAC 的出现不但实现了制造一台通用计算机的目标，而且标志计算工具进入了一个崭新的时代，是人类文明发展史中的一个里程碑。仅仅半个世纪，计算机已经使人类社会从工业化社会发展到了信息化社会。虽然 ENIAC 于 1955 年正式退役，并陈列于美国国立博物馆供人们参观，但它的丰功伟绩将永远记载在人类的文明史册中。

1945 年，ENIAC 的顾问、数学家冯·诺依曼在为一台新的计算机 EDVAC（电子离散变量计算机）所制定的计划中首次提出了存储程序的概念，即将程序和数据一起存放在存储器中，使编

程更加方便。这个思想几乎同时被科学家图灵(Turing)想到了。

1946 年,冯·诺依曼与他的同行们在 Princeton Institute 进行高级研究时,设计了一台存储程序的计算机 IAS,可惜因种种原因直到 1952 年 IAS 也未能问世。但 IAS 的总体结构从此得到了认可,并成为后来通用计算机的原型,图 2.1 就是 IAS 计算机的总体结构示意图。它由几部分组成:一个同时存放指令和数据的主存储器、一个二进制的算逻运算部件、一个解释存储器中的指令并能控制指令执行的程序控制部件以及由控制部件操作的 I/O 设备。

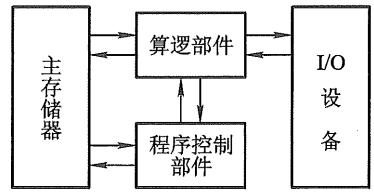

图 2.1　IAS 计算机结构

20 世纪 50 年代,美国出现了 Sperry 和 IBM 两大制造计算机的公司,后来又从 Sperry 公司分离出了 UNIVAC 子公司,他们控制着计算机市场。

1947 年,Eckert 和 Mauchly 共同建立了生产商用计算机的计算机公司,他们第一个成功的产品是 UNIVAC I(Universal Automatic Computer),后来 Eckert-Mauchly 公司成为从 Sperry-Rand 公司分离出来的 UNIVAC 子公司,并继续制造了一系列产品,如 UNIVAC Ⅱ 及 UNIVAC 1100 系列产品,它们成为科学和商用计算机的主流产品。同时 IBM 公司在 1953 年推出了首台存储程序的计算机 701 机,1955 年又推出了 702 机,使之更适用于科学计算和商业应用,后来形成了 700/7000 系列,使 IBM 成为计算机制造商的绝对权威。

自从 ENIAC 问世后,人类为提高电子计算机性能的欲望从未减退过,并在 20 世纪 50 年代初,除美国外,英、法、苏联、日本、意大利等国都相继研制出本国的第一台电子计算机,我国也于 1958 年研制成自己的第一台电子计算机。可是在这十多年的时间里,计算机的性能并未出现奇迹般的提高,它的运算速度每秒仅在数千次至上万次左右,其体积虽然不像 ENIAC 那样庞大,但也占了相当大的空间,耗电量也很大。直到 20 世纪 50 年代末,计算机技术迎来了第一次大飞跃的发展机遇,其性能出现了数十倍以至几百倍的提高,这就是用晶体管替代电子管的重大变革。

2. 第二代晶体管计算机

1947 年在贝尔实验室成功地用半导体硅作为基片,制成了第一个晶体管,它的小体积、低耗电以及载流子高速运行的特点,使真空管望尘莫及。进入 20 世纪 50 年代后,全球出现了一场以晶体管替代电子管的革命,计算机的性能有了很大的提高。以 IBM 700/7000 系列为例,晶体管机 7094(1964 年)与电子管机 701(1952 年)相比,其主存容量从 2 K 字增加到 32 K 字;存储周期从 30 μs 下降到 1.4 μs;指令操作码数从 24 增加到 185;运算速度从每秒上万次提高到每秒 50 万次。7094 机还采用了数据通道和多路转换器等在当时看来是最新的技术。

尽管用晶体管代替电子管已经使电子计算机的面貌焕然一新,但是随着对计算机性能越来越高的追求,新的计算机所包含的晶体管个数已从一万个左右骤增到数十万个,人们需要把晶体管、电阻、电容等一个个元件都焊接到一块电路板上,再由一块块电路板通过导线连接成一台计算机。其复杂的工艺不仅严重影响制造计算机的生产效率,更严重的是,由几十万个元件产生几百万个焊点导致计算机工作的可靠性不高。

随着1958年微电子学的深入研究,特别是新的光刻技术和设备的成熟,为计算机的发展又开辟了一个崭新时代——集成电路时代。

3. 第三代集成电路计算机

仔细分析就会发现,计算机的数据存储、数据处理、数据传送以及各类控制功能基本上都是由具有布尔逻辑功能的各类门电路完成的,而大量的门电路又都是由晶体管、电阻、电容等搭接而成,因此,当集成电路制造技术出现后,可以利用光刻技术把晶体管、电阻、电容等构成的单个电路制作在一块极小(如几个平方微米)的硅片上。进一步发展,实现了将成百上千个这样的门电路全部制作在一块极小(如几个平方毫米)的硅片上,并引出与外部连接的引线,这样,一次便能制作成成百上千个相同的门电路,又一次大大地缩小了计算机的体积,大幅度下降了耗电量,极大地提高了计算机的可靠性。这就是人们称为小规模集成电路(SSI)和中等规模集成电路(MSI)的第三代计算机,其典型代表为IBM的System/360和DEC的PDP-8。

1964年,IBM推出了一个新的计算机系列System/360,打破了7000系列在体系结构方面的一些约束。为了推动集成电路技术,改进原来的结构,IBM投入了大量的人力和物力进行技术开发,作为回报,它最终占领了大约70%的市场份额,成为计算机制造的最大制造商。

System/360系列中有不同的机型,但它们又都是互相兼容的,即在某种机型上运行的程序可以在这一系列中的另一种机型上运行。它们具有类似或相同的指令系统(该系列中低档机的指令系统可以是高档机指令系统的一个子集),各机型有类似或相同的操作系统,而且随着机器档次的提高,机器的速度、存储器的容量、I/O端口的数量以及价格都有所增长。

另一种有代表性的机器是DEC的PDP-8,它采用总线结构,有迷你机之称。它以低价格、小体积吸引了不少用户,售价仅16 000美元,而当时System/360大型机的售价为数十万美元。PDP-8使DEC迅速发展起来,使其成为继IBM之后的第二大计算机制造商。

从1946年的ENIAC到1964年的IBM System/360,历时不到20年,计算机的发展经历了电子管—晶体管—集成电路3个阶段,通常称为计算机的3代。显然,早期计算机的更新换代主要集中体现在组成计算机基本电路的元器件(电子管、晶体管、集成电路)上。

第三代计算机之后,人们没有达成定义新一代计算机的一致意见。

表2.1列出了硬件技术对计算机更新换代的影响。

表2.1 硬件技术对计算机更新换代的影响

发展阶段	时间	硬件技术	速度/(次/秒)
一	1946—1957	电子管	40 000
二	1958—1964	晶体管	200 000
三	1965—1971	中、小规模集成电路	1 000 000
四	1972—1977	大规模集成电路	10 000 000
五	1978年到现在	超大规模集成电路	100 000 000

进入到20世纪70年代后,把计算机当作高级计算工具的狭隘观念已被人们逐渐摒弃,计算

机成为一门独立的学科而迅猛发展,并且影响、改变着人类的生活方式,这是由于微处理器的出现(采用大规模和超大规模集成电路)、软件技术的完善及应用范围的不断拓宽所带来的必然结果。

2.1.2　微型计算机的出现和发展

集成电路技术把计算机的控制单元和算逻单元集成到一个芯片上,制成了微处理器芯片。1971 年,美国 Intel 公司 31 岁的工程师霍夫研制成世界上第一个 4 位的微处理器芯片 4004,集成了 2 300 个晶体管。随后,微处理器经历了 4 位、8 位、16 位、32 位和 64 位几个阶段的发展,芯片的集成度和速度都有很大的提高。与此同时,半导体存储器的研制也正在进行,1970 年,Fairchild 制作了第一个存储芯片,该芯片大约只有一个磁心这么大,却能保存 256 位二进制信息,但是每位的价格高于磁心。1974 年后,随着半导体存储器价格的迅速下降,位密度的不断提高,存储芯片的容量经历了 1 K 位,4 K 位,16 K 位,64 K 位,256 K 位,1 M 位,4 M 位,16 M 位,64 M 位,…,1 G 位这几个阶段,每个新的阶段都比过去提高到 4 倍的容量,而价格和访问时间都有所下降。

总之,芯片集成度不断提高,从在一个芯片上集成成百上千个晶体管的中、小规模集成电路,逐渐发展到能集成成千上万个晶体管的大规模集成电路(LSI)和能容纳百万个以上晶体管的超大规模集成电路(VLSI)。微芯片集成晶体管的数目验证了 Intel 公司的缔造者之一 Gordon Moore 提出的“微芯片上集成的晶体管数目每 3 年翻两番”的规律,这就是人们常说的 Moore (摩尔)定律。

微处理器芯片和存储器芯片出现后,微型计算机也随之问世。例如,1971 年用 4004 微处理器制成了 MCS-4 微型计算机。20 世纪 70 年代中期,8 位微处理器 8008、8080、R6502、M6800、Z80 等相继出现,并用 R6502 制成了 Apple Ⅱ微型计算机,用 Z80 制成了 CROMEMCO 80 微型计算机等。

最值得一提的是世界上第一大微处理器的制造商 Intel,其典型产品如下。

- 8080:世界上第一个 8 位通用的微处理器,1974 年问世。
- 8086:16 位,2.9 万个晶体管,地址 20 位,采用 6 个字节指令队列,指令系统与 8088 完全兼容,1978 年问世。
- 8088:集成度达 2.9 万个晶体管,主频 4.77 MHz,字长 16 位(外部 8 位),又称准 16 位,地址 20 位,采用 4 个字节指令队列,被 IBM 首台微型计算机(IBM PC)选用,1979 年问世。
- 80286:16 位,13.4 万个晶体管,6 MHz,地址 24 位,可用实际内存 16 MB 和虚拟内存 1 GB,1982 年问世。
- 80386:32 位,27.5 万个晶体管,12.5 MHz、33 MHz,地址 32 位,4 GB 实际内存,64 TB (1 TB = 2^{40} B)虚拟内存,其性能可与几年前推出的小型机和大型机相比,1985 年问世。
- 80486:32 位,120 万个晶体管,25 MHz、33 MHz、50 MHz,4 GB 实际内存,64 TB 虚拟内

存,引用更加复杂的 Cache 技术和指令流水技术,速度比 80386 快一倍,性能指标比 80386 高出 3~4倍,1989 年问世。

- Pentium:32 位,310 万个晶体管,66 MHz、100 MHz,4 GB 实际内存,64 TB 虚拟内存,采用超标量技术,使多条指令可并行执行,速度比 80486 高出 6~8 倍,1993 年问世。
- Pentium Pro:64 位,550 万个晶体管,133 MHz、150 MHz、200 MHz,64 GB 实际内存,64 TB 虚拟内存,采用动态执行 RISC/CISC 技术、分支预测、指令流分析、推理性执行和二级 Cache 等技术,1995 年问世。
- Pentium Ⅱ:64 位,750 万个晶体管,200~300 MHz,64 GB 实际内存,64 TB 虚拟内存,融入了专门用于有效处理视频、音频和图形数据的 Intel MMX 技术,1997 年问世。
- Pentium Ⅲ:64 位,950 万个晶体管,450~600 MHz,64 GB 实际内存,64 TB 虚拟内存,融入了新的浮点指令,以支持三维图形软件,1999 年问世。
- Pentium 4:64 位,4 200 万个晶体管,1.3~1.8 GHz,64 GB 实际内存,64 TB 虚拟内存,包括另外的浮点和其他多媒体应用的增强,2000 年问世。

显然,从 20 世纪 70 年代初至今,微型计算机的发展在很大程度上取决于微处理器的发展,而微处理器的发展又依赖于芯片集成度和处理器主频的提高。从 2000 年 Intel Pentium 4 问世至今的发展历程看,处理器的架构变化不大,主要从提高处理器的主频、增加扩展指令集、增加流水线、提高生产工艺水平(晶体管的线宽从 180 nm→130 nm→90 nm→65 nm)等几方面来不断改进处理器的性能。但制造工艺的缺陷,导致了处理器功耗持续上升。大量研究表明,每推出一代新型处理器,它的功耗是上一代处理器功耗的 2 倍,倘若芯片集成度达 10 亿个,处理器的自身功耗将会使人们一筹莫展。可见,有效解决微处理器的功耗和散热问题已成为当务之急。事实上一味追求微芯片集成度的提高,除了引发功耗、散热问题外,还会出现更多的问题,如线延迟问题、软误码率现象等。

为了提高计算机的性能,除了提高微处理器的性能外,人们还努力通过开发指令级并行性来实现。可是在指令级并行性应用中,又受到数据预测精度有限、指令窗口不能过大以及顺序程序固有特性的限制等,使得依靠开发指令级并行性来提高计算机的性能又有很大的局限性。

虽然很多因素阻碍了微型计算机性能的不断提高,可是随着计算机的广泛应用,尤其是网络技术的迅猛发展,人们依然在追求着机器性能的完美。例如,当前网络的环境基本上是让计算机处于桌面固定的状态,而人们更希望机器能围绕人们的需求转,越来越方便地使用计算机,不希望机器局限于固定的桌面式应用,让机器以手持式或穿戴式以及其他形式,使之更具人性化,能和谐地融合于人们的生活和工作之中。与此相适应的移动计算技术便应运而生。

移动计算模式迫切要求微处理器具有响应实时性、处理流式数据类型的能力、支持数据级和线程级并行性、更高的存储和 I/O 带宽、低功耗、低设计复杂性和设计的可伸缩性。

当前主流商用处理器大部分都是超标量结构,是一种在一个时钟周期内同时发射多条标量指令到多个功能部件以提高处理器性能的体系结构,若每周期发送 4 条指令,已不能满足日渐庞大的应用程序对高性能的需求。而继续开发更大发射带宽的超标量结构将会导致处理器的逻辑