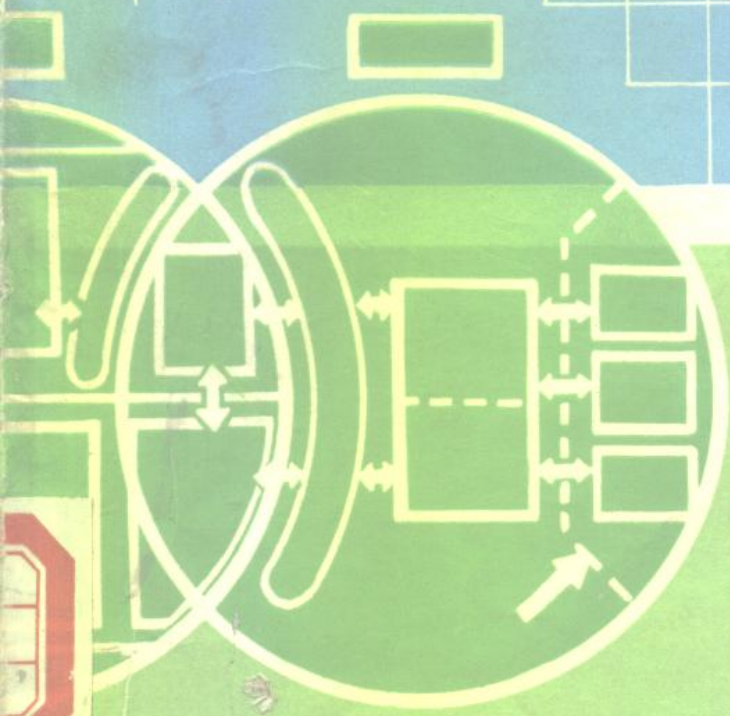


计算机 系统结构和组织

王绪宜 陈华生 骆光武 等编著



国防工业出版社

73.87.2.01
125

计算机系统结构和组织

王绪宜 陈华生 骆光武 等编著

国防工业出版社

内 容 简 介

本书讲述计算机系统结构和计算机组织的基本概念、基本原理以及发展趋向。

全书共分十二章。第一章讲述了多层结构概念和冯·诺依曼结构特点，并介绍了较早突破冯·诺依曼结构的例子；第二章描述了DJS200系列的系统结构；第三章到第七章分别讲述了快速运算方法，微程序设计的方法及应用，多层存储结构，输入输出通道和文件存储器；第八章评述了计算机系统结构的发展趋向；第九章到第十二章分别描述了DJS200系列四个机型的特点。

本书可供有关科技人员及大专院校计算机专业师生参考。

计算机系统结构和组织

王崇宜 陈华生 骆光武 等编著

*

国防工业出版社出版

新华书店北京发行所发行 各地新华书店经售

国防工业出版社印刷厂印装

*

787×1092¹/₁₆ 印张29 665千字

1985年10月第一版 1985年10月第一次印刷 印数：00,001—11,400册

统一书号：15034·2807 定价：5.90元

前 言

本书采用把计算机系统结构和计算机组织结合起来进行叙述的方法。从六十年代中期第三代计算机确立以来,软、硬件工作者都普遍重视对计算机系统结构的研究。这是因为,它已经形成研究软、硬件之间的功能分担,并考虑如何从软、硬件两方面实现一个最佳的计算机系统的学科。对机器设计者来说,从计算机系统结构的角度去分析研究计算机各功能部件的特性,更易于观察和理解它们之间的内在联系。

计算机系统结构是指一个计算机物理系统中程序员所能看到的概念性结构和功能特性。计算机系统结构的内容包括:指令系统、数据类型和格式、输入输出系统的连接方式、文件存储系统等内容。计算机组织的内容一般包括计算机各部件的逻辑功能和部件间的功能联系,诸如算术逻辑结构、总线结构、控制方式(如先行控制、流水线等)、微程序设计、多层存储结构等。当然,与实现逻辑和算术功能有关的指令系统和数据格式等也是重点讨论的内容。

由于计算机系统结构和计算机组织在概念和原理上有很多密切相关的内容,因此,只有结合起来讨论,才能使问题得到深入了解。但是,在本书中,一般不再涉及元件的物理特性、电路原理以及其他工艺问题。因为我们相信,它所包含的基本概念和基本原理是软、硬件工作者均应掌握的基本内容。

本书力求用通俗易懂的描述来表达计算机的各种重要概念,因此,只要读者具有“计算机基础”的知识(例如:布尔代数;数的机器表示及二进制运算方法;一般逻辑元件和部件;计算机的基本组成等),则学习本书的基本内容就不会有多大困难。通过本书的学习,可以掌握计算机系统结构和组织的基本概念、原理和分析方法。对于硬件专业的读者,则可为进一步学习系统结构的有关课程(例如“分布式并行处理计算机结构”、“计算机网络”、“可靠性设计技术”等)打下基础。

本书主要结合一些中、大型计算机系统来讲述,因为它们一般代表着计算机系统结构和组织的发展特点。第一章至第七章是概念和原理性的论述,其后面都附有一定的习题和复习题可作教学和学习中的参考。目录中标有*的章节可以作为选学的内容,因此,若用作教材,则可不受其章节顺序的限制。

本书各章由下列同志提供初稿:第一章,王绪宜;第二章,陈华生;第三章,陈允济;第四章,蔡士杰;第五章,陈世福;第六章,江邦人;第七章,高玉寰;第八章,周根林和吴春谔;第九章,骆光武;第十章,刘秀芳;第十一章,张秉麟;第十二章,张国良。全稿由南京大学王绪宜、陈华生、骆光武三人分别进行整理和补充,最后由王绪宜对全稿进行修改和统编。

在本书的编写过程中,得到了电子工业部计算机工业管理局总工程师陈力为同志的具体指导。华北计算机研究所陈炳从同志对全书的编写做了大量的工作,提出了许多宝贵的意见。在此一并表示衷心的感谢。

本书文稿虽经多次修改,并通过了教学实践,但由于水平所限,可能仍会有错误和不妥之处,恳请读者批评指正。

目 录

第一章 绪论	1
1.1 引言	1
1.2 计算机系统的多层结构概念	2
1.3 冯·诺依曼结构	5
1.4 IBM370 系统结构简介	10
1.5 突破冯·诺依曼结构的机器实例	17
1.5.1 B-5000 计算机的堆栈结构	17
1.5.2 CDC6600 的并行操作	22
1.5.3 ILLIAC IV 的处理单元阵列	26
1.6 本书概要	30
习题和复习题	31
参考文献	32
第二章 DJS200 系列的系统结构	33
2.1 引言	33
2.2 DJS200 系列的基本结构格式	33
2.2.1 系列统一规定和分型概况	33
2.2.2 界限保护和寻址方式	36
2.2.3 数据格式	40
2.2.4 指令格式	43
2.3 系统控制功能	45
2.3.1 程序状态字	46
2.3.2 控制寄存器	49
2.3.3 中断系统	49
2.3.4 监视	56
2.3.5 系统的时钟	57
2.3.6 控制台	57
2.3.7 输入输出系统	58
2.4 系统的处理功能	59
2.4.1 DJS200 系列的指令系统	59
2.4.2 传送型指令	60
2.4.3 算术运算指令	63
2.4.4 逻辑运算指令	69
2.4.5 转移指令	71
2.4.6 控制与管态指令	73
2.4.7 程序设计示例	73
习题和复习题	74
参考文献	74
第三章 快速运算方法	75
3.1 引言	75
3.2 加法	76

3.2.1 成组跳跃进位加法器的逻辑结构	79
3.2.2 十进制数的加减法运算	83
3.3 乘法	88
3.3.1 跳“0”跳“1”乘法	88
3.3.2 多位一乘的乘法	90
3.3.3 快速并行乘法器	99
3.4 除法	107
3.4.1 跳“0”跳“1”除法	107
3.4.2 迭代除法	110
习题和复习题	124
参考文献	125
第四章 微程序设计	126
4.1 引言	126
4.2 微程序控制	127
4.2.1 计算机的数据通路	127
4.2.2 常规机器的控制原理	128
4.2.3 微程序控制的基本原理	131
4.2.4 控制存储器	134
4.3 微程序设计方法	136
4.3.1 基本设计原则	136
4.3.2 微指令字结构格式的设计	138
4.3.3 微指令的顺序控制	142
4.3.4 微指令的执行	144
4.3.5 微中断与强制转移	147
4.3.6 软件对微程序设计的支援	148
4.3.7 微程序设计实例	149
4.3.8 毫微程序设计概念	162
4.3.9 微程序设计的优缺点	163
4.4 微程序设计的应用	165
4.4.1 微诊断	165
4.4.2 用户微程序设计	168
4.4.3 仿真	169
习题和复习题	171
参考文献	171
第五章 存储器组织	173
5.1 引言	173
5.2 多层存储结构概念	175
5.3 主存储器的存储控制	178
5.3.1 单一存储控制	178
5.3.2 重叠与交叉存取控制	179
5.3.3 存储器总线结构	180
5.3.4 实例	181
5.4 高速缓冲存储器	184
5.4.1 存储器的缓冲	184
5.4.2 高速缓冲存储器的结构	184
5.4.3 缓冲存储器性能参数的选择	187
5.5 相联存储器	188

5.5.1 基本概念	188
5.5.2 硬件结构	189
5.5.3 逻辑功能	190
5.5.4 相联存储器的应用	192
5.6 虚拟存储器	193
5.6.1 虚拟存储器的概念	193
5.6.2 虚拟存储器的基本结构	194
5.6.3 虚拟存储器的实现方法	197
习题和复习题	204
参考文献	204
第六章 输入输出组织	205
6.1 引言	205
6.2 输入输出设备控制器	212
6.2.1 输入输出设备地址	212
6.2.2 输入输出设备命令	212
6.2.3 输入输出设备状态字节	213
6.2.4 设备断定字节	214
6.3 输入输出标准接口	214
6.3.1 标准接口描述	214
6.3.2 接口逻辑操作	217
6.4 输入输出过程	222
6.4.1 输入输出指令	223
6.4.2 通道指令	225
6.4.3 通道程序	226
6.4.4 通道地址字	228
6.4.5 设备的选择和启动	229
6.4.6 数据传送期	229
6.4.7 输入输出操作结束处理	230
6.4.8 多重输入输出操作	232
6.4.9 程序实例	233
6.4.10 原始程序输入	235
6.5 通道设计	236
6.5.1 输入输出状态	236
6.5.2 输入输出状态与条件码	237
6.5.3 微子程序	237
6.5.4 五条输入输出指令的处理流程	248
习题和复习题	251
参考文献	252
第七章 文件存储器	253
7.1 引言	253
7.1.1 顺序存取设备	254
7.1.2 直接存取设备	255
7.2 磁盘存储器	258
7.2.1 硬磁盘磁道数据记录格式	258
7.2.2 硬磁盘存储器操作命令	261
7.2.3 硬磁盘存储器编址方式	263
7.2.4 硬磁盘存储器驱动程序举例	263

7.2.5 软磁盘存储器简介	265
7.3 磁带存储器	270
7.3.1 磁带记录信息的格式	270
7.3.2 磁带文件编址与寻址	272
7.3.3 驱动程序举例	272
习题和复习题	273
参考文献	273
第八章 计算机系统结构的进展——分布处理技术	274
8.1 计算机系统结构进展概况	274
8.1.1 新一代计算机系统结构发展的背景	275
8.1.2 计算机复合系统结构现状	276
8.1.3 分布处理技术的确立	282
8.2 分布式计算机	282
8.2.1 分布式计算机的一般概念	282
8.2.2 负荷分布式计算机	284
8.2.3 功能分布式计算机	287
8.3 分布式计算机网络	292
8.3.1 分布式网络的一般概念	292
8.3.2 分布式网络的技术课题	294
8.3.3 分布式网络的结构	296
8.4 数据库系统	300
8.4.1 数据库系统的一般概念	300
8.4.2 数据库计算机	301
8.4.3 分布式数据库	304
8.5 数据流计算机	306
8.5.1 数据流计算机的基本原理	306
8.5.2 MIT 的DFM	308
8.5.3 LAU 数据流计算机	313
8.5.4 数据流计算机的发展	318
8.6 第五代计算机	320
8.6.1 背景	320
8.6.2 基本概念	320
8.6.3 基本结构	321
第九章 DJS200-10 型计算机	324
9.1 引言	324
9.1.1 主要性能	324
9.1.2 时钟	326
9.1.3 总体框图	327
9.2 数据通路	329
9.2.1 加法器	329
9.2.2 总线	331
9.2.3 B 寄存器	332
9.2.4 存储控制电路	332
9.3 微程序设计	335
9.3.1 微程序控制器	335
9.3.2 微指令字格式	336
9.3.3 微程序格式	338

9.3.4 微程序设计步骤	339
9.4 输入输出通道	339
9.4.1 输入输出系统概况	339
9.4.2 10型的子通道	341
9.4.3 输入输出系统的数据通路	343
9.4.4 通道微程序设计	344
9.5 RAS 技术	347
9.5.1 海明校验	347
9.5.2 通道指令复执	351
9.5.3 检查用微子程序	351
9.5.4 诊断和故障定位检测	351
第十章 DJS200-20 型计算机	352
10.1 引言	352
10.1.1 20型计算机的组成及其特点	352
10.1.2 20型计算机的主要性能	352
10.2 中央处理机	356
10.2.1 中央处理机的数据通路	356
10.2.2 20型计算机的时序脉冲系统	361
10.3 主存储器	362
10.3.1 主存储器的组成及有关数据通路	362
10.3.2 主存储器的时序	363
10.3.3 主存储器的保护性措施	364
10.4 微程序设计	364
10.4.1 微指令字的结构格式	364
10.4.2 微指令的并行和重叠	369
10.4.3 微程序控制器的结构	369
10.5 输入输出系统和结合型通道	371
10.5.1 结合型通道的组成	371
10.5.2 通道微中断	371
10.5.3 通道操作的微程序控制	374
10.6 RAS 技术	377
10.6.1 主存储器的可靠性措施	377
10.6.2 数据通路的校验	379
10.6.3 指令复执	380
10.6.4 双工系统	381
10.6.5 故障诊断	382
第十一章 DJS200-40 型计算机	388
11.1 引言	388
11.1.1 基本性能	388
11.1.2 主要特点	390
11.2 40型的数据通路	391
11.2.1 单机系统的组成	391
11.2.2 双工系统的组成	392
11.2.3 中央处理机的数据通路	395
11.2.4 主存储器	405
11.2.5 集中控制通道的数据通路	407
11.2.6 专用外围设备控制器	412

11.3	40型的微程序设计	413
11.3.1	40型微程序设计概况	413
11.3.2	运算器和分析器的微程序设计	415
11.3.3	诊断部件的微程序设计	421
11.4	40型的中断系统	422
11.5	RAS 技术	423
11.5.1	40型的故障检测	423
11.5.2	40型的故障处理	425
11.5.3	40型的诊断系统	426
第十二章	DJS200-60型计算机	429
12.1	引言	429
12.2	数据通路	430
12.2.1	单处理机系统的数据通路	430
12.2.2	双处理机的数据通路	436
12.3	控制通路	437
12.3.1	先行控制	437
12.3.2	流水线技术	438
12.3.3	微程序控制	439
12.4	中断系统	441
12.5	RAS 技术	442
12.5.1	错误检测	443
12.5.2	指令复执	443
12.5.3	诊断系统	444
12.5.4	双工方式和系统重构	446

第一章 绪 论

1.1 引 言

从第一台电子计算机问世至今,已经三十多年。人们习惯上常说,它已经经历了电子管、晶体管、集成电路和大规模集成电路等四代,并正酝酿着第五代。实际上,无论从计算机本身的速度、体积、可靠性和成本来看,还是从计算机应用的广度和深度着眼,都可以说明:电子器件的工艺革新和换代引起了电子计算机的迅速发展。

但是,电子计算机的元器件的换代,只是其发展的一个突出标志,而不是唯一的标志。很明显,如果对早期计算机的结构,用集成电路组装起来,人们并不会承认它就是第三代的计算机。因为第三代计算机在系统结构上(同样,在系统软件上)较之早期的计算机已经有了很大的改进和发展。因此,系统结构方面的特点同样被看作是计算机划代的重要标志。

计算机系统结构(Computer Architecture)一词虽然早就见诸文献,但现在通行的含义却是阿姆达尔(Amdahl)等人在1964年提出的。他们把系统结构定义为:由程序员所看到的一个计算机系统的概念性结构和功能特性,它不同于这个系统的另外一些因素,诸如基本数据流及其控制,逻辑设计和物理实现等等。

梅厄斯(G. J. Myers)在用多层结构概念分析了计算机系统以后,进一步指出:阿姆达尔等所指的程序员应为机器程序员或编译程序员。并且认为,计算机系统结构的重要任务是确定软件和硬件的功能界面,研究哪一些系统功能由软件实现最好,哪一些系统功能由硬件实现最好。这就是说,软、硬件之间的功能界面并不是固定的,而是浮动的。事实上,一方面,随着计算机功能的日益增强,许多由程序实现的功能都由硬件通过一条指令来实现了(例如,向量乘法、快速傅里叶变换,褶积运算等等相当复杂的运算,现在都可在诸如阵列处理机之类的专用设备上用硬件直接实现);另一方面,在一些廉价的小型机上又常常连乘、除法和浮点操作都由软件实现。这些事例不仅说明,软、硬件的功能界面的浮动幅度是相当大的,而且也说明,软、硬件在功能上是等效的。因此,研究计算机系统中软、硬件的功能分担确实应是计算机系统结构的一个重要任务。

计算机系统结构的另一个重要任务是,在分析传统机器系统结构的特点和局限性的基础上,探讨突破这些限制的途径。传统机器结构乃指冯·诺依曼结构(1.2节将作专门讨论)。虽然计算机的系统结构已经获得了很大的发展,但从一些基本点来看,迄今为止的绝大多数计算机仍然属于冯·诺依曼结构。而这个传统机器结构,随着计算机的发展,日益暴露出它本身的一些主要缺点,特别是在提高实现高级语言的效率和并行处理功能等方面更为明显。在提高实现高级语言的效率方面所遇到的问题,在很大程度上是由于计算机系统结构和程序设计语言在语义上的严重脱节所致。例如在数据结构上,高级语言所需数组(Array)和字符串(String)等数据类型,在大多数现代计算机系统

8710325

结构中都不具备。这样就增加了编译的复杂性,导致程序量过大甚至加大了程序设计语言的失真度和软件的不可靠性等。在实现并行处理问题上所受到的限制,是由于传统机器的集中的、顺序的控制性质所造成的。可以认为,许多类型的多处理机结构就是在这一点上对传统结构有了突破。

为了进行上述研究工作,首先应该对现代计算机的典型结构有比较深入的了解。因此,有必要对下述两方面的内容作详细介绍:

(1) 现代计算机的概念性结构(即阿姆达尔所定义的计算机系统结构)的内容,主要包括:

- ① 数据类型和数据格式;
- ② 指令格式和指令系统;
- ③ 编址和寻址方式;
- ④ 中断系统;
- ⑤ 中央处理机和输入输出设备的交接处理等;

(2) 实现上述概念性结构的设计方法、计算机系统的各个功能部件,以及它们之间的逻辑联系,诸如:

- ① 快速运算方法及其逻辑结构(功能框图);
- ② 微程序设计;
- ③ 先行控制和流水线(本书通过具体机型来介绍);
- ④ 存储器层次结构;
- ⑤ 通道组织和文件存储器。

尽管在这些内容中,许多是程序员所看不到的,也就是它们对程序员来说具有透明性(Transparency),因而它们并不属于上述计算机系统结构所定义的范围。但是,这些内容与计算机系统结构有密切的关系,而且只有把两者结合起来才能使讨论深化,把问题讲清楚。通常把它们归属为计算机组织(Computer Organization)。梅厄斯则称之为处理机系统结构(Processing Architecture)。我们在本书中也沿用这个术语,以便在一般情况下把计算机系统结构和计算机组织称为系统结构,只在必要时才把它们区分开来。

在对现代计算机典型结构较为详细了解的基础上,本书用一定篇幅介绍了计算机系统结构的发展趋向,供进一步研究时参考。

本章首先介绍计算机的多层结构概念,它有助于了解系统结构在研究计算机系统时的地位和作用。然后,讨论冯·诺依曼结构的基本特点和发展概况,并进而以IBM370系统为例介绍现代计算机的系统结构。最后,叙述较早突破冯·诺依曼结构的几个例子。虽然这些机器的系统结构被其后衍生的计算机改进和发展了,但由于它们在设计原理上具有的明显特点,故一向被人们作为典型的例子进行讨论。

1.2 计算机系统的多层结构概念

现代计算机是一个复杂的系统。它面对着各种不同的应用,包含了复杂的系统软件和硬件。由于软、硬件的设计者和使用者从不同的角度、以各种不同的语言来对待一个计算机系统,因此,他们各自看到的计算机系统的属性以及对它提出的要求和作出的改

进也不一致。这样，一个计算机系统对不同的对象来说，就成为实现不同语言的，具有不同属性的机器。如果在软、硬件之间，设计者和使用者之间，不能密切配合，则会随着计算机学科各个领域的发展，在它们之间出现越来越严重的脱节现象，从而影响计算机的性能和效率的进一步提高。

计算机系统的多层结构概念正是为了研究上述不同现象从各个角度所看到的机器之间的有机联系，分清相互之间的界面，明确各自的要求，以便构成最有效的计算机系统结构。

对于计算机多层结构的分层方式，目前并不统一，我们在这里采用了比较常用的一种，如图 1-1 所示。

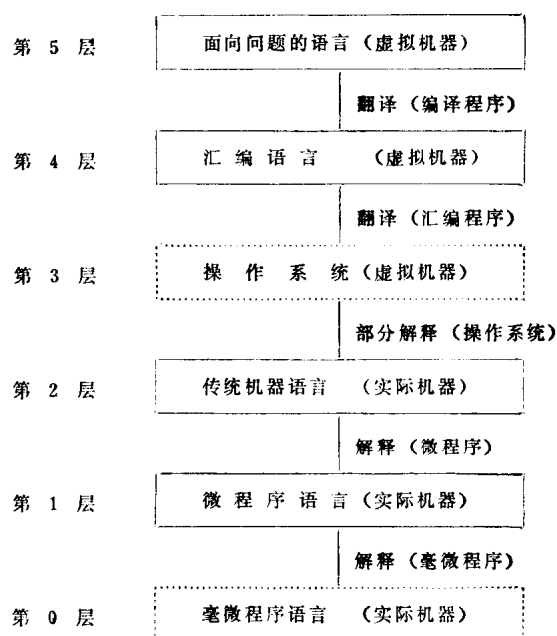


图1-1 计算机系统的多层结构

这样一个多层结构并不是凭空想象的，而是逐步发展而形成的。现在就首先来回顾一下多层机器的演变过程。

我们先从图 1-1 所示的传统机器（第 2 层）说起。所谓传统机器即指实现机器语言（指令系统）的机器。早期的计算机系统，由于没有任何其他语言，没有操作系统，而是直接用机器语言（二进制代码表示）来编程序，并且由电子线路直接实现，因此，它只有传统机器这一层。现代计算机虽然较之复杂多了，但就这一层来说，至今仍然保留着早期计算机的基本形态（冯·诺依曼机器结构），所以称它为传统机器层。

1951年，英国数学家威尔克斯（M·V·Wilkes）首次提出微程序设计的概念。在根据这种概念所设计的机器中，每条机器指令不是由电子线路直接实现，而是由一组微指令（微程序）来实现。这样，微程序控制的计算机就由两层机器组成：一层是实现机器语言的传统机器；另一层是实现微程序设计语言的微程序机器（图 1-1 中的第 1 层）。其中，传统机器程序是逐条指令地由一组确定的微程序来分析执行的。这个过程通常称

为解释。由于在微程序控制的机器中，微程序一般存放在一个只读存储器（ROM）里，而ROM及其有关电路可作为单独的产品出售，所以常把微程序称为固件。微程序机器较之用组合逻辑控制的传统机器有规整性、灵活性以及易于采用新工艺等优点，因而从六十年代中期开始便得到了广泛的应用。但也有很多机器，特别是超大型计算机，由于速度要求仍采用硬联组合逻辑控制，此外，工厂总是把硬件说明、工作原理手册等规定在传统机器这一层。因此，我们把这两层用硬件和固件实现的机器称为实际机器，统归为硬件的范围。

近年来为了改进微程序设计，进一步提高其效率和灵活性，提出了毫微程序设计的方法（见第四章），从而把微程序机器层又一分为二，在微程序机器层之下出现了毫微程序机器层（图中第0层）。其中，每一条微指令由毫微程序解释执行。目前毫微程序设计的机器还处于实验阶段，未得到广泛应用，故在图中用虚线画出，但借此可以说明，多层结构还可从微程序机器层向下延伸。

计算机系统多层结构概念更重要的是用于从传统机器层向上的延伸。

早期计算机采用机器语言手编程序进行解题的原始状况并未持续多久，就出现了汇编语言。汇编语言用助记符号（英文缩写）代替了机器语言的二进制代码，由于其可读性较好，也较少出错，所以用以编写程序的效率大为提高。不过用汇编语言写的程序（源程序）不能直接在机器上运行，必须使用一个汇编程序把它转换成机器语言的程序（目标程序）才行，这个过程称为翻译。汇编程序就是一种翻译程序。由于汇编语言机器的功能是由机器语言实现的，所以它在多层结构中处于传统机器之上（第4层）。

在程序设计语言中，汇编语言仍属低级语言。因为它与传统机器语言基本上是一一对应的，所以写出来的程序仍然很长，从而与解题的要求相差很远。因此，很快就出现了面向问题的高级语言，如：FORTRAN，ALGOL和COBOL等。它们实现起来也像汇编一样，需要把用高级语言写的源程序翻译成机器语言程序，这样才能在实际机器上执行。翻译的过程一般是先把高级语言源程序经编译程序翻译成汇编语言程序（或某种中间形式的语言程序），而后再翻译成机器语言程序（或由机器语言解释执行）。因此，把它置于汇编语言机器层之上，列为第5层。在第5层之上还可能有进一步的延伸，有面向各种应用的应用语言机器层等。

现在返回来说明一下操作系统机器层（图中第3层）。在这个多层结构中，操作系统的情况有些特殊。一方面，它可以看作一个混合层。因为操作系统的大部分语言是与机器指令系统相同的。对于这一部分，汇编语言（或其他高级语言）在翻译成机器语言时，相当于穿过操作系统这一层，直接用机器语言实现；对于另一部分，则是操作系统专用的。如控制输入输出等操作的一些特权指令，以及象访管指令之类的广义指令等。遇到用这些指令时，则必须由操作系统来实现。由于操作系统包括这样两部分，故称为混合层。此外，由于操作系统也是用机器语言描述的程序，所以操作系统机器层置于传统机器层与汇编机器层之间。另一方面，也应该注意操作系统的主要作用是提高整个计算机系统的效率，扩大它的功能，控制它的整个工作（包括各级语言的翻译或解释，以及在机器上的运行等等）。因此，不能简单地把它看作多层结构中的一个机器层。

可以看出，从第3层以上的机器层，它们的语言必须通过汇编程序，编译程序（有时也兼用解释程序）翻译成传统机器语言才能得以实现。我们把这些必须通过软件（程

序) 的转换才能在实际机器上实现其语言的机器层称为虚拟机器层。

通过以上说明可知, 计算机系统可以看成是一个有机的多层结构。在这个多层结构中, 各层机器都有自己的语言, 而每一层语言通过翻译或解释由低一层的机器实现, 显然, 由于各自的语言工具不同, 每一层的设计者(程序设计者和机器设计者)对整个系统的理解和要求是不同的。因此, 他们在研究或设计中, 相互之间, 特别是相邻层次之间, 必须协同拟定明确的界面, 研究功能分担和各种约定, 否则就会造成功能、格式、语义等方面的脱节。从原则上讲, 各层之间都应注意这个问题。但目前主要的脱节现象还不在于实际机器(第2层以下)之间和虚拟机器(第3层以上)之间, 而在于实际机器与虚拟机器之间, 即第2层与第3层或第4层(考虑到上述操作系统的特殊性)之间, 亦即软件和硬件之间。前已指出, 系统结构的一个重要任务, 正是研究软、硬件之间的功能界面问题。

软、硬件之间, 特别是程序设计工作者与系统结构工作者之间的脱节, 其原因是多方面的。有传统的历史原因, 也有软、硬件之间缺少沟通, 尤其是系统结构工作者对软件缺乏了解所形成的原因。

在电子计算机发展的初期, 由于当时技术条件的限制, 机器设计者主要关注的是硬件的成本和可靠性, 他们仅着眼于一个具有最基本功能的计算机, 而把大量的问题留给程序设计者解决。随着工艺技术的改革, 计算机的功能日益增强, 结构日益复杂, 但是, 早期的设计准则仍在一定程度上起着作用, 机器设计者依然希望以软件的代价换取最低的硬件成本。实际上, 现今的计算机系统的软、硬件成本比例已经发生倒置, 早期的设计准则所带来的经济性已经没有什么意义。

另外, 由于软、硬件之间的脱节, 一些系统结构的改进却往往给程序设计者带来新的麻烦。下面举两个例子来说明。

一个例子是流水线结构(见第12章)。这个概念可以说是硬件方面的重要发明。但是由于流水线机器是以多个指令并行操作的, 当某一指令出错时, 将会造成不精确的指令执行断点, 因此将使程序员难以得到确定的出错信息。

另一个例子是采用通用寄存器。这个方法也是系统结构的重要改进, 在现代计算机上得到了普遍的采用。但是, 它却给系统程序增加了麻烦。由于通用寄存器的数目很少(例如IBM370系统只有16个), 因而编译程序必须产生管理寄存器的代码, 并且要优化它的使用。

这两个例子说明了系统结构上的改进是与软件密切关联着的。每一项改进都应从软、硬件两方面权衡利弊, 决定取舍。从一定意义上讲, 计算机系统结构正是从硬件中提取出来的与软件直接有关的各种因素。在计算机系统设计时, 不是仅从硬件方面, 而是从软、硬结合的观点, 即从整个系统的观点来分析和确定这些因素的。

1.3 冯·诺依曼结构

在第一台存储程序的电子计算机(EDSAC, 1949年)问世之前, 冯·诺依曼(J. Von Neumann)等人在1946年就对它的基本原理进行了较为详细的描述, 他们归纳了电子计算机的如下基本性质:

(1) 计算机必须有一个存储器。执行计算的程序和数据都以二进制代码的形式存

于存储器中。由于从形式上看,指令和数据没有区别,因此,程序在其执行过程中也和数据一样可以处理和修改。

(2) 必须有一个控制器。在它的控制下,指令依次从存储器中取出,然后对其解释和执行。每条指令都由操作码、操作数地址和(或)运算结果地址组成。

(3) 必须有一个运算器。它用来完成所需要的算术运算和逻辑运算。

(4) 必须有输入和输出设备。它用来进行人机之间的通信。同时可以认为,人机之间的指令、数据以及运算结果的交换是没有限制的。

冯·诺依曼等人根据上述原则制成了早期的计算机EDVAC和IAS等,它们的基本组成如图1-2所示。他们所做的所有这些工作为尔后的计算机的迅速发展奠定了基础。

在这里我们应该指出:冯·诺依曼等人的主要贡献并不在于确立了计算机的五个组成部分,这是因为,早在一百年以前,在英国的巴贝奇(C. Babbage)所制造的机械计算机中,就已经具备了这

五个组成部分;也不在于实现了数的二进制表示和运算,这是因为,虽然冯·诺依曼等对二进制数的四则运算进行了更为详细地分析和研究,但是二进制运算方法在他们以前的机电计算机Z₃

[德国人雷色(Komrad Zuse)于1941年制成]上就

已经采用。冯·诺依曼等人的主要贡献在于“存储程序概念”的提出及其实现。在此以前的计算机,包括第一台电子计算机ENIAC在内,都是把计算程序预先在调度板上编制好然后再执行的。因此,程序的装入和变更是极其麻烦的事。只有在确定了存储程序的概念以后,才真正实现了计算机的连续自动计算原理。现代计算机都是建立在这一基础之上的。

图1-2所示的结构是电子计算机的最初结构形式,它是冯·诺依曼机器的基本组成。这种结构的机器往往称之为古典计算机。古典计算机其所以简单和初级,自然是由于历史条件形成的。在当时,主要解决的是电子计算机的能行性问题,因此只要求实现最基本的功能。另外,无论是在使用经验方面还是在工艺技术方面,当时都还没有可能提供成熟的条件来实现一个复杂的系统结构。因此,古典计算机在系统结构方面存在着明显的缺点。特别是慢速的输入输出操作和快速的运算控制器的串行操作两者大大影响了计算机速率的充分发挥。在计算机的应用很快进入商业领域以后,由于数据处理要求频繁的输入输出操作,故上述缺点显得更加突出。于是,随着计算机应用的日益推广,计算机的系统结构也不断得到改进。许多系统结构的重要概念在五十年代就已经提出并且得到了实现。现概要列举如下:

(1) 变址寄存器概念。早在1949年,就在英国曼彻斯特大学的计算机EDSAC上首次采用了变址寄存器的概念。1953年作为正式产品的Datatran计算机又进一步采用了它。变址寄存器方法在数组计算及其他循环重复的计算中,为程序设计提供了很大的方

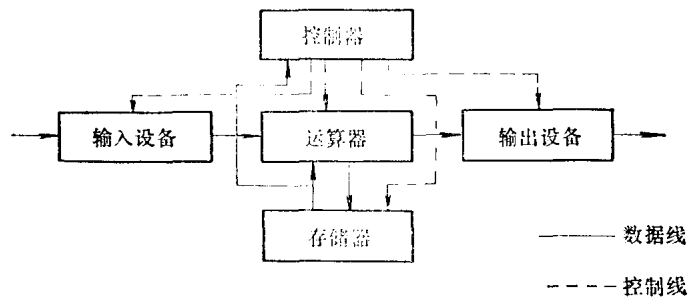


图1-2 早期计算机结构

便，所以在其后的计算机上不断得到改进和推广。

(2) 通用寄存器概念。它开始出现在 Pegasus 计算机 (Ferranti 有限公司, 1956 年) 上。其作用是把变址器和累加器结合起来使用, 从而提供了多累加器结构。这个概念在以后的很多重要计算机系统 (如 IBM360 和 370 系统) 中得到了采用。

(3) 浮点数据表示。由于它把操作数表示为两部分: 阶和尾数, 因此充分扩大了机器上数的表示范围, 从而大大减少了程序中经常调整比例因子的麻烦。IBM 704 和 IBM NORC 是最早采用这个概念的机器 (1954 年)。

(4) 程序中断概念。这一概念是指, 程序在执行过程中, 被外部事件或程序执行中发生的事件所中断, 系统转而处理所发生的事件, 待处理完毕, 再返回到被中断的程序。这一概念最早在 1954 年问世的 Univac 1103 上得以采用。

(5) 输入输出通道概念。通道是输入输出操作的控制部件, 它提供了输入输出系统与中央处理机二者的并行操作能力, 从而引起计算机系统结构的很大变化。输入输出操作由执行专用程序 (即通道程序) 的通道来控制。相对于输入输出处理机 (即通道) 而言, 原来的运算器和控制器就有了一个新的名称: 中央处理机 (CPU)。由于 CPU 执行的常规程序和通道执行的通道程序可以并行操作, 分时共享主存储器周期, 因此大大提高了计算机的使用效率。早期的通道又称为交换器 (详见第六章), 最初出现在 LARC 计算机上 (Remington Rand, 1956 年)。

(6) 间接寻址概念。在间接寻址方式中, 根据指令中操作数地址部分从主存储器中所取出的数据, 仍然作为操作数地址来继续访问主存储器。这种间接寻址方式还可以是多重的。利用间接寻址可以在指令本身不变的情况下, 灵活地改变操作数地址, 从而提高程序的通用性。间接寻址的概念较早是在 1958 年问世的 IBM 709 上采用的。

(7) 虚拟存储器概念。这个概念发端于 1959 年问世的 Atlas 计算机 (英国曼彻斯特大学)。它采用了一级存储器的设计方法, 使主存储器具有磁鼓存储器的容量和接近于磁心存储器的速度。它最早引进了分页和动态地址变换的概念, 从而奠定了尔后虚拟存储器的基础。

以上列举的计算机系统结构方面的各种新概念, 在其以后的应用中都得到了进一步的发展和完善。待到六十年代中期, IBM 公司发表 IBM 360 系统的时候, 这些概念便作为第三代计算机系统结构的特点肯定下来。

IBM 360 系统具有一个功能很强的庞大的指令系统和比较完善的中断系统。因此, 它可以胜任科学计算、数据处理以及实时控制等多方面的应用。在系统结构上还有三个应该强调的重要特点, 即系列化、微程序设计, 以及输入输出通道的分级连接结构。现就这三个特点说明如下:

1. 系列化

系列化的概念是与计算机系统结构的含义密切联系着的。当初, 阿姆达尔正是为了说明系列机概念而使用“计算机系统结构”一词的。一个计算机系列中可以包括许多机型。尽管面对不同的用户, 它们具有不同的价格、速度和性能指标, 但它们却具有一个共同的计算机系统结构。因此, 对程序员来说, 他们看不到这些机器之间的差别。不管是用高级语言所写的程序还是用机器语言、汇编语言所写的程序, 都可以在各个机型上执行。这就是所谓“兼容性”。有时, 由于系列中的低档机的存储容量小、速度慢以及