〔荷兰〕A. S. 塔南鲍姆 著

分级式计算机的构成

科学出版社

分级式计算机的构成

〔荷兰〕A. S. 塔南鲍姆 著

汪成为 于士齐 译

张梓昌 校订

科学出版社

1982

本书从软、硬件相结合的角度出发，把近代的一些计算机系统分成多级式来分析和设计计算机系统，这种分级式的概念对于理解和学习目前计算机系统的组成是很有益的。书中还叙述了一些软、硬件在逻辑上等效的概念。

本书共分八章，前两章对计算机系统作了一般介绍，后六章依次为微程序级、传统机器级、操作系统级、汇编语言级、面向问题的高级语言级和虚拟机级。

本书的论述深入浅出，书中引用了大量的实例，每章后面附有习题。本书可供从事计算技术的专业人员、大专院校师生作参考书。

Andrew S. Tanenbaum
STRUCTURED COMPUTER ORGANIZATION
Prentice-Hall, Inc., 1976

分级式计算机的构成

〔荷兰〕A. S. 塔南鲍姆 著
汪成为　于士齐 译
张梓昌　校订
责任编辑　范铁夫

科学出版社 出版
北京朝阳门内大街 137 号
中国科学院印刷厂印刷
新华书店北京发行所发行　各地新华书店经售

1982 年 7 月第 一 版　开本：787×1092　1/16
1982 年 7 月第一次印刷　印张：22
印数：0001—7,200　字数：496,000
统一书号：15031·413
本社书号：2628·15—8

定价：3.40 元

译 者 的 话

这是一本很有特色的关于计算机系统构成和体系设计的书，自从出版以来已被欧美的一些大学作为教科书或参考书，某些著名学者对此书也给予了较高的评价.

本书作者把计算机软件和计算机硬件结合起来讨论，作者认为软件和硬件的功能在逻辑上是等效的，而且它们的界限也不是固定不变的. 今天的软件功能明天可能用硬件去实现，现在的硬件功能也许将来就要用软件去实现.

本书作者把一个现代的计算机系统看作是由若干个机器级所构成的，其最基础的一级是微程序级，依次往上为传统机器级、操作系统级、汇编语言级、面向问题的高级语言级和虚拟机级. 因此在研制一个计算机系统时，设计者应从这种多级机的概念出发来决定每一级的功能以及它们之间的关系. 当学习和使用一个新的计算机系统时，也应从这种多级机的概念出发，以便对庞杂的系统作出条理清晰的分析.

作者对本书的编写方法和选材作了周密推敲，对一些较抽象的概念作了深入浅出的说明，书中引用了大量的实例，它们大部分是选自我们比较熟悉的三个系列机，即 IBM 370、CDC Cyber 70 和 DEC PDP-11. 在每一章后还附有大量的习题.

译者认为，本书对于有一定计算机基础知识的读者来说是一本很好的参考书.

序　言

从前的计算机是很简单的，这些计算机执行少量的几条基本指令，用户直接使用这些原始的指令去编写程序. 这样的时代早已过去了. 现代的计算机是一个非常复杂的整体，它通常是由五、六个或者甚至更多个不同的级所构成的，在每一级上都能编制程序和进行研究. 事实上，在今天常常很难说清楚何处是"机器"的尽头，何处又是软件的开始.

很多大学在早期的课程中设有汇编语言程序设计和计算机结构的课. 在那时教会学生在一个具体的计算机上用汇编语言编制程序就认为是足够的了. 这样的时代也早已过去了. 这些课程必须跟上迅速变化着的学科，并且现在必须对有关计算机结构的各种广泛的课题进行介绍，而这些课题大部分也只是在最近几年内才刚刚被研究室以外的人有所了解. 分段式的虚拟存贮器、并行处理、竞态条件、微程序设计、可变体系结构的机器、存贮转发网络、自虚拟机和超高速缓冲存贮器等就是各种各样的课题中的几个例子.

本书打算作为有关汇编语言程序设计和计算机结构等入门课程的教科书. 其所需的预备知识仅仅是计算机科学的基础课程(或具有同等的实践经验)，其中包括用FORTRAN、COBOL、或PL/I等高级语言进行程序设计的初步知识. 本书不需要数学和工程的基础，目的是使本书适用于二年级的水平. 本书几乎包括了ACM课程68中的B2教程的全部材料，以及教程写好以后被认为重要的那些类似的材料. 本书也能作为"机器结构和机器语言程序设计"的COSINE教程之用. 本书的各章是独立的，可以把它们作为特定课题的参考之用.

本书的主题是计算机分级式的体系结构. 最底层的一级是实实在在的硬件，它的功能是执行称为微程序的解释程序. 被微程序所解释的语言就是大部分人所说的"机器语言"，也是在生产厂的机器参考手册上所描述的这种语言. 于是，这个"机器语言"比起直接由电路执行的语言来是更高的一个级了. 我们将最低一级称为"微程序级"(第四章). 而由它所支撑的这一级称为"传统机器级"(第三章).

大部分计算机都有一个运行在这个传统机器级上的操作系统. 这个操作系统提供给用户一个"扩大的"或"虚拟"的机器，此机器具有在传统机器级上所没有的指令和功能. 这些包括文件操作指令、并行处理(执行多个任务)的指令和虚拟存贮器. 为操作系统的用户提供的这些特性和指令可以认为定义了一个新的级，即"操作系统机器级"(第五章).

第4级是符号汇编语言级. 这一级和第2、3两级不同，第2、3级是由解释程序所支撑的，而这一级是由一个称为汇编程序的程序来实现的(第六章)，汇编程序是把第4级程序翻译到某一个低的级上去. 还有一些本书中尚未涉及到的更高的级，它们是由面向问题的语言所定义的.

第七章和第三—六章不同，第三—六章是每一章详尽地讨论一个级，而第七章是从整体上来考察多级计算机的构成和应用. 第八章是对进一步学习的引导.

IBM 370，CDC Cyber 70，和 DEC PDP-11 计算机是本书不断使用的例子. 援引这些机器只是为了举例说明问题，并不要求预先对它们都熟悉.

书中有少量的几个例题是用 PL/I 的简单语言书写的，这对任何一个熟悉 FORTRAN 或 ALGOL 的人都能看懂. 选用 PL/I 的原因是因为它应用广泛、又适合于结构式的程序设计. 我原先选用的是 ALGOL 68，但遗憾的是二年级学生几乎都不懂 ALGOL 68.

对本书作出过贡献的人很多. 有时，我感到我不是一个作者，而更象是一个编者. 有两个人，他们比其他各位更为突出. Jack Alanen 读了全部的第一次手稿，使得许多不好的概念不致于出现在打印稿上. 他还对内容和表达方法提出了大量的建议. 我特别感谢 Jack，因为他教了我大量有关教学的知识. Kim Gostelow 以极其仔细的态度检查了打印稿，并进行了大量的注释，改正差错. 还把我的平淡的文体改成很形象的英语，几乎在手稿的每一段上都注上了改进的意见. 我对他们两位表示深切的谢意.

Reind van de Riet 和 Mitchell Tanenbaum 也阅读和评论了全部手稿. 并提出了不同的观点. John W. Carr III, Arnie Falick, Dick Grune, Jim van Keulen, Bod Rosin, Carel Stillbroer 和 Wayne Wilner 等，每个人都提出了建议，并在一些个别的章节内进行了帮助. 我也感谢我的学生，特别是 Arie de Bruin, Wim Harmsen, Ad König, Sape Mullender, Johan Stevenson, 和 Hans van Vliet, 因为他们提供给我对本书的反应. Homburg-Knieper 夫人很好地完成了手稿的几次打字工作.

我还想对国际商业机器公司 (International Business Machines Corporation), 控制数据公司 (Control Data Corporation), 数字设备公司 (Digital Equipment Corparation) 和巴勒斯公司 (Burroughs Corporation) 表达我的谢意，感谢这些公司允许我采用它们版权所有的出版物，如：IBM 的《IBM 系统/370 操作原理手册》和 IBM 3125 的 CPU 的工程图，CDC 的《Control Data Cyber 70 的 72、73、74 型计算机系统参考手册》；DEC 的《PDP-11/45 处理器手册》和 PDP-11/40 的工程图；Burroughs 的《B1700 系统参考手册》. 在叙述这些机器时的任何差错均由本人负责.

最后，我还想对 Suzanne 表示谢意，在我准备这本书的过程中，是她给我以鼓励、支持和帮助.

A. S. 塔南鲍姆

致　教　师

汇编语言程序设计的练习应当是使用本书的任何一个课程的一个主要部分．在学期的一开始，就应该给学生一份有关用计算机经调试通过的汇编语言程序详尽文本表格，同时还应给学生一份关于计算机的简要说明和说明运行程序的管理细则，即到哪里去找到终端或键控穿孔机，需要什么样的作业控制卡，如何使用编辑程序等．学生必须进行穿孔和运行程序，使得他们今后不必在操作的细节上再花费时间了．

下一步是让他们在程序上作一些不很重要的变化．例如某个密码程序，它使用一种简单的替换密码就能对信息进行编码和译码，改变这些密码程序就能对数目和文字进行处理了．在对一个或者两个简单的程序逐渐作了一些困难的修改后，学生就应准备对付更加重要的程序了．

一个比较理想的题目是由教师用汇编语言提供图 4-15 的某种形式的解释程序．要求学生去改进由解释程序所支撑的目标机．在第四章末尾的习题中给出了几个建议，但应鼓励学生独立思考．这个练习是说明了“传统机器级”确实是由作为基础的解释程序(微程序)所确定的，而不是由硬件所确定的．

其他一些可能的练习可以是对某个计算机网络或是某个虚拟存贮器的性能作模拟，或是对某个简单的汇编程序作一些修改．把可以工作的程序提供给学生作为基础比让他们从头开始什么都由他们自己来写要好些．这样做可以免得迟钝的学生花费大量的时间在这样的问题上，如“你应在哪一列上设置操作码?”，同时也给较优秀的学生提供一种方便的途径去从事更富有想象的设计，如在汇编程序上加上一条宏指令等．

在小计算机上传授经验比用分时终端好，而用分时终端又比成批处理的方式好，但这种选择通常取决于有什么样的现成的设备．在教程中必须尽可能早地为学生安排专门阅读适当的生产厂的手册．

本书可以作为集中于一个学期内的课本，也可以辅加少量的课外读物作为比较从容的一年的课本．另一方面，作为一个学期的课本时可以集中钻研某些题目而省略另一些题目．例如，着重于计算机结构的课程，可使用第一章、第二章、第三章的一部分，然后是第四和第七章．着重于汇编语言程序设计的课程，则可使用第一、二、三和六章．而另外一种方法是所有各章都讲授，但某些概念较为高深和附加例题的段节，则留给能力较强而想多学一些的学生去学习．

目　　录

序言

致教师

第一章　引言 …… 1

1.1　语言、级和虚拟机 …… 2

1.2　现代的多级机 …… 3

1.3　多级机的历史演变 …… 5

1.4　硬件、软件和多级机 …… 7

1.5　进程 …… 8

1.6　本书的概要 …… 11

第二章　计算机系统的构成 …… 14

2.1　处理器 …… 14

2.1.1　指令的执行 …… 15

2.1.2　指令的并行执行 …… 16

2.2　存贮器 …… 18

2.2.1　位 …… 18

2.2.2　存贮器地址 …… 19

2.2.3　特征位 …… 20

2.2.4　辅助存贮器 …… 21

磁带 …… 21

磁盘 …… 22

磁鼓 …… 22

光存贮器 …… 24

2.3　输入/输出 …… 24

2.3.1　I/O 设备 …… 24

2.3.2　I/O 处理器 …… 25

2.3.3　字符编码 …… 25

2.3.4　误差校正码 …… 25

2.3.5　从频编码 …… 28

2.4　信息的传送 …… 31

2.4.1　数据通路 …… 31

2.4.2　远程通信 …… 32

调制 …… 33

异步和同步传输 …… 33

单向、半双向、全双向传输 …… 35

2.5　计算机网络 …… 35

2.6　分布式计算机 …… 37

第三章　传统机器级 …… 43
3.1　几个传统机器级的例子 …… 43
3.1.1　IBM360 和 370 系列 …… 43
3.1.2　CDC6000、Cyber70 和 Cyber170 …… 48
3.1.3　DEC PDP-11 …… 51
3.2　指令的格式 …… 54
3.2.1　指令格式的设计准则 …… 55
3.2.2　扩展操作码 …… 56
3.2.3　指令格式的几个例子 …… 57
3.3　定址法 …… 61
3.3.1　立即定址法 …… 62
3.3.2　直接定址法 …… 63
3.3.3　寄存器定址法 …… 63
3.3.4　间接定址法 …… 64
3.3.5　变址 …… 65
3.3.6　基地址寄存器 …… 66
3.3.7　堆栈定址法 …… 67
逆波兰法 …… 68
逆波兰法表达的公式的计算 …… 70
3.3.8　PDP-11 的定址方法 …… 72
3.3.9　关于定址方式的讨论 …… 75
3.4　指令的类型 …… 76
3.4.1　数据传送指令 …… 76
3.4.2　双操作数运算 …… 77
3.4.3　单操作数运算 …… 79
3.4.4　比较和条件转移 …… 81
3.4.5　转子指令 …… 82
3.4.6　循环的控制 …… 83
3.4.7　输入/输出 …… 84
3.5　数据的表示法 …… 87
3.5.1　整数 …… 87
3.5.2　浮点数 …… 87
3.5.3　布尔量 …… 87
3.5.4　字符 …… 88
3.5.5　字符串 …… 88
3.5.6　数组 …… 90
内情向量 …… 90
有界变址法 …… 92
3.6　流程的控制 …… 92
3.6.1　顺序流程的控制及转移 …… 92
3.6.2　子程序 …… 93
3.6.3　联立子程序 …… 99

3.6.4 陷井(中断) …… 101
3.6.5 中断 …… 102
第四章 微程序级 …… 111
4.1 处理器的部件 …… 111
4.1.1 寄存器 …… 112
4.1.2 总线 …… 112
4.1.3 门 …… 112
4.1.4 钟 …… 113
4.1.5 存贮器的接口 …… 113
4.1.6 算术逻辑部件 …… 114
4.1.7 处理器元件的组装 …… 115
4.2 基本操作 …… 116
4.2.1 寄存器间的传输 …… 116
4.2.2 存贮器的读和写 …… 117
4.2.3 位测试 …… 117
4.3 假想的目标机器级 …… 117
4.4 假想的宿主机器级 …… 119
4.4.1 宿主机器级的寄存器 …… 120
4.4.2 宿主级的 ALU …… 120
4.4.3 宿主级的门和数据通路 …… 120
4.5 开门顺序 …… 123
4.5.1 子周期 …… 123
4.5.2 ADD 指令的开门顺序 …… 124
4.6 微程序化的开门控制 …… 125
4.6.1 微指令 …… 125
4.6.2 微程序的执行 …… 126
4.6.3 一台两级的机器 …… 128
4.7 微程序设计语言 …… 128
4.7.1 GATE 微指令的表示方法 …… 128
4.7.2 TEST 微指令的表示方法 …… 129
4.8 目标机的解释程序 …… 130
4.8.1 乘法指令的解释 …… 132
4.8.2 除法指令的解释 …… 134
4.8.3 前景 …… 135
4.9 微程序级的设计 …… 136
4.9.1 编码字段 …… 136
4.9.2 横向与纵向结构 …… 137
4.9.3 存贮器周期的比值和重叠执行 …… 140
4.9.4 毫微存贮器 …… 142
4.9.5 通用的与专用的微程序级 …… 143
4.9.6 微程序级的结构的评价 …… 144
4.10 微程序设计的优、缺点 …… 145

4.11 IBM370/125 的微程序级 …… 147
4.11.1 IBM370/125 的微程序级的结构 …… 147
4.11.2 IBM3125 的微指令 …… 150
4.12 PDP-11/40 的微程序级 …… 151
4.12.1 PDP-11/40 的微程序级的结构 …… 151
4.12.2 单总线的操作 …… 154
4.12.3 PDP-11/40 的微指令 …… 155
4.13 BURROUGHS B1700 …… 158
4.13.1 B1700 的结构 …… 158
4.13.2 B1700 的指令系统 …… 162
第五章 操作系统机器级 …… 167
5.1 操作系统机器级的实现 …… 167
5.2 虚拟的 I/O 指令 …… 168
5.2.1 连贯的文件 …… 169
5.2.2 随机存取的文件 …… 171
5.2.3 虚拟 I/O 指令的实现 …… 172
5.2.4 IBM370 的虚拟 I/O …… 175
5.2.5 Cyber70 的虚拟 I/O …… 178
5.2.6 PDP-11 的虚拟 I/O …… 181
5.2.7 第 3 级 I/O 的比较 …… 183
5.3 用于并行处理的虚拟指令 …… 183
5.3.1 进程的产生和破坏 …… 184
5.3.2 竞态条件 …… 185
5.3.3 用信标使进程同步 …… 187
5.3.4 用于进程间通信的指令 …… 189
5.4 其他的 3 级指令 …… 190
5.4.1 目录管理指令 …… 190
5.4.2 3 级机的配置变换 …… 191
5.5 虚拟存贮器 …… 193
5.5.1 分页 …… 193
5.5.2 分页的实现 …… 195
5.5.3 请求式页面调度和工作集合模型 …… 198
5.5.4 页面替换策略 …… 200
5.5.5 涂改位 …… 201
5.5.6 硬件实现地址对照 …… 202
5.5.7 页面的大小和碎片 …… 202
5.5.8 超高速缓冲存贮器 …… 203
5.5.9 分段 …… 204
5.5.10 PDP-11 的虚拟存贮器 …… 207
棋盘残局 …… 209
5.5.11 MULTICS 的虚拟存贮器 …… 210
5.5.12 IBM370 的虚拟存贮器 …… 213

5.5.13 段式虚拟存贮器和文件 I/O ····· 214
5.6 作业控制语言 ····· 215
第六章 汇编语言级 ····· 224
6.1 汇编语言入门 ····· 224
6.1.1 什么是汇编语言 ····· 225
6.1.2 汇编语言语句的格式 ····· 225
6.1.3 汇编语言和 PL/1 的比较 ····· 227
6.1.4 程序调整 ····· 228
6.2 汇编过程 ····· 229
6.2.1 两遍汇编程序 ····· 230
6.2.2 第一遍 ····· 230
6.2.3 第二遍 ····· 234
6.3 查找和分类 ····· 235
6.3.1 查找 ····· 236
6.3.2 线性查找法 ····· 236
6.3.3 折半查找法 ····· 237
6.3.4 分类或排序 ····· 239
6.3.5 混列编码 ····· 240
6.3.6 混列函数和冲突 ····· 243
6.3.7 相联技术的比较 ····· 245
6.4 宏功能 ····· 246
6.4.1 宏定义、宏调用和宏扩展 ····· 246
6.4.2 带参数的宏功能 ····· 248
6.4.3 条件宏扩展 ····· 248
6.4.4 宏调用的嵌套 ····· 250
6.4.5 递归宏调用 ····· 251
6.4.6 嵌套的宏定义 ····· 253
6.4.7 在汇编程序内的宏功能的实现 ····· 253
6.5 连接和装入 ····· 254
6.5.1 由连接程序所完成的任务 ····· 255
6.5.2 目标模块的构成 ····· 258
6.5.3 地址固定时间和动态再定位 ····· 258
6.5.4 动态连接 ····· 260
第七章 多级机 ····· 268
7.1 实现新级的方法 ····· 268
7.1.1 解释 ····· 268
7.1.2 翻译 ····· 269
通用的宏处理程序 ····· 270
7.1.3 过程扩展 ····· 271
7.2 多级机的设计策略 ····· 272
7.2.1 自上而下的设计 ····· 272
7.2.2 由下而上的设计 ····· 273

7.2.3 中间向外的设计方法 …… 275
7.3 程序可移植性 …… 275
7.3.1 通用程序设计语言 …… 276
7.3.2 强制法 …… 277
7.3.3 UNOCL（面向语言的通用计算机） …… 278
7.3.4 自理虚拟机 …… 280
7.3.5 仿真 …… 282
7.3.6 网络 …… 283
7.4 自虚拟机 …… 283
7.4.1 IBM VM/370 系统 …… 284
7.4.2 自虚拟机的目的 …… 285
自虚拟机和分时 …… 286
操作系统测试 …… 286
机密数据的保护 …… 286
7.4.3 自虚拟机的实现 …… 287
异常和虚拟机失误 …… 288
虚拟机 I/O 的模拟 …… 289
自修改通道程序 …… 289
映影页表 …… 290
7.5 高级机体系结构 …… 292
7.5.1 定址和解说符 …… 293
7.5.2 高级机指令 …… 297
7.5.3 高级机的优点和缺点 …… 299
第八章 对进一步阅读材料的推荐和参考目录 …… 302
8.1 对进一步阅读材料的推荐 …… 302
8.1.1 定址和指令 …… 302
8.1.2 汇编程序和汇编语言程序设计 …… 302
8.1.3 二进制数和运算 …… 303
8.1.4 字符编码、冗余和非冗余 …… 304
8.1.5 计算机结构 …… 304
8.1.6 传统机器级 …… 305
8.1.7 死锁 …… 306
8.1.8 文件系统 …… 306
8.1.9 高级机 …… 307
8.1.10 输入/输出 …… 308
8.1.11 连接程序和加载程序 …… 308
8.1.12 宏功能 …… 308
8.1.13 微程序级 …… 309
8.1.14 多级计算机 …… 310
8.1.15 网络 …… 311
8.1.16 操作系统 …… 312
8.1.17 并行处理 …… 313

8.1.18 自虚拟机 …… 313
8.1.19 符号表 …… 314
8.1.20 远距离通信 …… 315
8.1.21 虚拟存贮器 …… 316
8.2 按字母顺序排列的参考目录 …… 317
附录 …… 321
A 有限精度运算和二进制数 …… 321
A.1 有限精度的数 …… 321
A.2 基和数系 …… 322
A.3 从一个基到另一个基的变换 …… 323
A.4 负的二进制数 …… 325
A.5 二进制运算 …… 326
B 浮点数 …… 328
C 布尔代数 …… 333

第一章 引 言

数字计算机是通过执行给定的指令来为人们求解问题的机器. 描述怎样完成某一任务的一串指令就称为程序. 每一个计算机的电路都能够识别和直接执行有限的一组简单指令. 所有的程序在执行以前都必须转化成这些简单指令. 这些基本指令诸如两数相加, 检查某一个数, 看它是否为 0, 把一小段数据从计算机存贮器的一部分传送到另一部分等. 很少有比这些更为复杂的了.

同时,计算机的基本指令构成了一种使人们能和计算机进行通信联系的语言,这种语言就称为*机器语言*.

人们凡是要设计一台新的计算机, 就必须决定在计算机的机器语言中应包括哪些指令. 通常,这些指令应和计算机的预期用途及性能要求相一致,并且为了减低所需电路的复杂性和价格, 总是试图使这些基本指令尽可能的简单. 但正因为大部分机器语言是如此简单,所以使得人们在使用它们时就感到困难和冗长了.

解决这一问题有两个主要的方法,它们都包括要设计一组新的指令,而且应比设在机器内的指令更便于使用. 同时, 正如同设在机器内的指令构成了一种称为 L1 的语言一样,这些新的指令也构成了一种称为 L2 的语言. 这两种方法的不同之处是计算机如何执行用 L2 编写的程序,而计算机归根结底只能执行用机器语言 L1 所编写的程序.

执行用 L2 编写的程序的方法之一是用等效的一串 L1 指令去替换每一条 L2 中的指令,其结果是产生了一个完全由 L1 指令所组成的程序. 计算机是执行新的 L1 程序, 而不是原来的 L2 程序了. 这样的一种方法称为*翻译*.

另一种方法是编写一段用 L1 语言的程序, 它把用 L2 写成的程序作为输入数据, 并且依次地考察 L2 的每一条指令, 直接执行与之对应的一串等效的 L1 指令. 这个方法不要求首先就要产生一个用 L1 编写的新程序. 这个方法称为*解释*, 实现解释的程序称为*解释程序*.

实际上, 翻译和解释是十分相似的. 在这两个方法中, 最终都是以执行一串等效的 L1 指令来实现 L2 的指令的. 其不同之处是: 在翻译中, 全部 L2 程序一开始就转换成 L1 程序, L2 程序就被置之不顾,而去执行新产生的 L1 程序. 在解释中,当每一条 L2 指令被译码和考察后就直接地被执行,不产生翻译出来的程序. 这两种方法都是被广泛使用的.

也可以不用翻译或解释的方法来考虑问题, 更为方便的是设想存在一个假想的计算机或是*虚拟机*, 它的机器语言就是 L2. 如果能够很便宜地构成这样一个机器, 那就不需要 L1 或者就根本不需要一个执行 L1 程序的机器. 人们就能简便地用 L2 编写程序, 计算机也能直接地执行它. 虽然要用电路去构成一个采用 L2 语言的虚拟机是十分昂贵的, 但人们能够为虚拟机编写程序, 然后用一个以 L1 编写的程序对它作解释或翻译, 而 L1 程序是能够在现有的机器上直接执行的. 换言之,人们能够为虚拟机编写程序,就好象真有一台这样的机器似的.

为了实现翻译和解释, 语言 L1 和 L2 的差别不能太大. 这一限制常常使 L2 虽然比

L1 强一些，但在大多数应用场合下还远不是理想的. 原先创造 L2 的企图是为了减轻程序人员由于应用了对机器很合适而对使用者并不合适的语言来表达问题引起的负担. 那么，由于 L2 并不理想就妨碍了原先的目的. 然而，事情也并非毫无希望.

办法之一是创造另一组指令，它与 L2 相比是更加面向使用者，而不是面向机器. 这第三组指令也构成一种语言，我们称之为 L3. 人们可以用 L3 编写程序，就好似真有一台其机器语言是 L3 的虚拟机. 这样的程序能够翻译成 L2，或者通过用 L2 写成的解释程序来执行它.

可以发明一系列语言，它们每一种都比它的前一种更方便些，直到最终得到一种满意的为止. 指出这样一点是很重要的，即每一种语言都用它的前者作为一个基础，所以我们可以把应用了这一方法的计算机看成有一系列的层次或级，如图 1-1 所示的那样，一级在另一级的顶上. 最底部的语言或级是最简单的，最高处的语言或级是最复杂的.

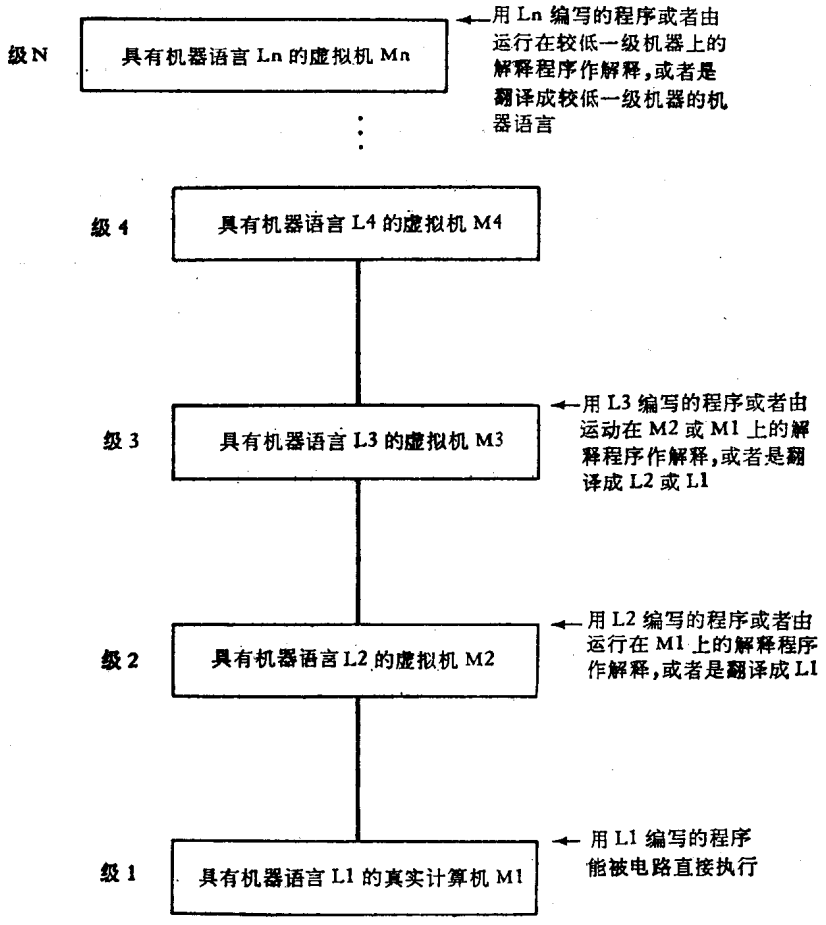

图 1-1 多级机

1.1 语言、级和虚拟机

语言和虚拟机之间有着重要的关系. 每一种机器都有某一种机器语言，它是由该机

器所能执行的全部指令所组成的．事实上，一种机器规定了一种语言．同样地，一种语言规定了一种机器，该机器能执行用该语言写成的全部程序．当然，由某一种确定的语言所规定的机器，如果直接用电路去构成它可能是十分复杂的，也是很昂贵的．但是我们仍然不妨这样来设想，一个把 PL/I、ALGOL68 或 COBOL 作为其机器语言的机器可能是一个十分复杂的怪物，但它肯定是可以想象的．也许在几年后要构成这样一个机器将是可行的．

一个具有N级的计算机可以看成是N个不同的机器语言的虚拟机．我们将可互换地使用“级”和“虚拟机”两个术语．只有用 L1 语言写出的程序才能直接地用电路来执行，不需要翻译或解释．而用 L2，L3，…，Ln 编写的程序就需要用一个在较低一级上运行的解释程序进行解释，或是翻译成另一种与较低一级相应的语言．

对于一个在第N级虚拟机上编写程序的人而言，他不必去了解作为基础的解释程序和翻译程序．他只知道能为第N级虚拟机编写程序，而且不管如何它总是要被执行的．他对于这些程序是由解释程序一步一步地执行的（解释程序本身又被另一个解释程序所执行）、还是由电路直接执行的并不感兴趣．他从这两种情况所得的结果是一样的，反正程序是被执行了．

对于绝大部分程序设计者来说，他使用一台N级的机器就只对最顶端的这一级感兴趣，它和最底端的机器语言之间只有很少的相似之处．然而，如果人们要了解计算机究竟是怎样工作的，那就必须对各级进行研究．凡致力于设计新的计算机或新的级(即新的虚拟机)的人，就必须了解全部各级，而不能只是顶端这一级．本书的主题是关于计算机怎样由一系列级构成的这样一些概念和方法，以及对一些重要的级作详细的说明．本书的书名来源于这样一个事实，即如把计算机看成是一系列的级，那么对于理解计算机是怎样构成的就提供了一个良好的结构或骨架．此外，按照级的概念来设计一个计算机也有助于保证所得的最终结果在结构上是良好的．

1.2 现代的多级机

大部分现代的计算机是由两个以上的级所组成的．如图 1-2 所示的五级系统也是不罕见的．市场上许多通行的计算机的最底端这一级在指令系统和一般结构上有很多共同之处．严格地说，两个具有不同的机器语言的计算机其最底端这一级也是不同的．但由于它们存在大量的相似之处，因此就允许我们对这样一级抽出其本质特性，并且就象已有明确定义似地对它们进行讨论．例如，在这一级中几乎没有一台机器的指令是超过 20 或 30 种．其中大部分指令含有这样的操作，即把数据从计算机的这一部分传送到另一部分，有些还含有简单的测试操作．这一级就称为*微程序级*，它所执行的解释程序就称为*微程序*．我们把它称为第一级是为了强调其位置处于最底端，其他各级都是建立在它的上面的．在这一级中的指令是由计算机的电路来实现的．

每一个 1 级机都有 1 个或数个能够在它上面运行的解释程序．每一个解释程序定义了一种 2 级语言（也定义了一个虚拟机，其机器语言就是该 2 级语言）．这些 2 级机也有许多相似之处，甚至不同厂家所出厂的机器也是大同小异的．由于没有一个公认的名称，在本书内我们把这一级称为*传统机器级*．