计算机系统结构和RISC设计

洪志良 编译

复旦大学出版社

计算机系统结构和RISC设计

洪志良　编译

复旦大学出版社

（沪）新登字202号

责任编辑　林溪波
责任校对　马金宝

计算机系统结构和RISC设计
洪志良　编译
复旦大学出版社出版
（上海国权路579号）
新华书店上海发行所发行　江苏东台印刷厂印刷
开本850×1168　1/32　印张15.25　插页0　字数352,000
1994年7月第1版　1994年7月第1次印刷
印数1—5,000
ISBN 7-309-01341-7/T·107
定价14.00元

内 容 提 要

本书是介绍计算机体系结构的一本著作。

全书共分十一章。第一章介绍四十多年来特别是最近几年计算机的发展，从中引出 Amdahl 定律。第二章给出计算机的性能方程和计算机性能测试方程。第三章用四维空间来分析指令集。第四章给出了历史上成功了的例子：IBM 360/370，DEC VAX，Intel 80 X 86 系列和 RISC 机的指令集和定量测试结果。第五章单独介绍存贮器分层管理系统、虚存、高速缓冲区、主存和地址变化页表。第六章主要介绍微码执行技术。指令的流水线操作在第七章中介绍，流水线操作的结构、数据和控制冲突及解决办法是本章的重点。主要的计算机输入和输出：显示器、磁盘、总线、网络等安排在第八章中叙述。第九章简单介绍了矢量机的原理和性能估算。第十章介绍采用先进流水线技术：记分牌。最后一章给出与 SPARC 指令兼容的 RISC 设计实例。

前　言

计算机体系结构的发展远远领先于计算机书籍内容的更新。为使广大读者能从书中学习最近几年发展起来的计算机体系结构和了解这种趋势，这就是作者编写本书的目的。

自1946年在美国普林斯顿大学第一台电子计算机问世以来，随着微电子技术的发展，计算机经历了电子管、晶体管、集成电路、VLSI和联网工作站五个时代，现已进入多重机的研究、开发和应用时期。

计算机的性能是多方面的，广为人知的是计算机的计算速度。1993年3月在德国海诺威计算机博览会上展出的瑞士苏黎世高工的63重处理机MUSIC已经达到每秒380亿次浮点计算能力。计算机的性能还包括带外围设备的能力，这个能力是可以扩充的。体积、重量和功耗上的改进也同样反映计算机的发展。

初期计算机不仅操作速度慢，而且只能执行几条简单指令的操作。随着计算机性能的提高，计算机的指令集愈来愈复杂，指令种类越来越多，其中有些指令操作相当复杂。在这种情况下，和半导体只读存贮器发展息息相关的微码执行技术也发展起来。八十年代初期RISC机的出现打破了计算机指令集愈来愈复杂的趋向。采用操作速度快和最常用的操作指令，来代替不经常用到的繁琐的指令是RISC机在机器性能上获益的关键。联网工作站使计算机资源共享，数据通信容易。目前正在发展的多重机系统，使计算机处理能力更强，它使卫星探测的数据分析，气象卫星的图像处理和机器人的实时处理成为可能。

和计算机CPU并行发展的存贮器，在过去40年中，存贮器的容量提高了6个数量级。二级存贮系统中，半导体存贮器占据着第一层高速缓冲区和主存，16兆位动态存贮器已经在商业上应用。磁盘垄断着第二层存贮系统。五十年代直径为121.92cm(48英寸)存贮量为几百BSI的磁盘现在已发展到直径为6.35 cm(2.5英寸)容量为150百万BSI；更小的磁盘（其容量达到20亿BSI）也已问世。可靠、便宜和可以携带的存贮器是磁带和软盘。为适应由以CPU为中心的系统到以网络为中心的系统的过渡，二级存贮器管理系统正向多级存贮器管理系统发展，这也带动了网络的发展。

平行处理是提高计算机性能的关键。流水线操作在执行指令上的应用使机器同时平行处理几条指令。程序的作业并行操作使机器向矢量机过渡。几台甚至几十台处理器分工执行是多重机发展的本意。处理好平行处理中的冲突，更加有机地、合理地安排好计算机系统的平行处理是计算机体系结构设计的目标。

学习如何设计计算机体系结构存在两种途径。过去是通过定义和理论的学习来设计计算机。美国斯坦福大学John L. Hennessy教授和加州大学贝克莱分校David A. Patterson教授另辟蹊径，希望学生通过对历史上成功体系结构例子的学习和性能的定量分析来学习和掌握计算机体系结构的设计。

本书在第一章引言中，引入了基本的Amdahl定律。第二章介绍了计算机的性能方程和测试性能的步骤。第三章的内容是指令集设计。第四章是指令集的测试。在以上两章中，列举了历史上成功了的计算机体系结构的许多例子：IBM 360/370，DEC VAX，Intel 8086

系列和 RISC 机。存贮器的分层管理和各层次的存贮系统，如高速缓冲区、主存、虚存在第五章中介绍。第六章重点介绍六十年代和七十年代盛行的微码执行技术。指令的流水线执行过程及其冲突在第七章中介绍。第八章介绍输入和输出设备。第九章和第十章，对矢量机和先进的流水线操作作了简单的介绍。第十一章阐述了 SUN SPARC 指令集和能执行该指令集的设计实例。计算机领域的宏大世界不可能全包括在本书中。本书重点介绍体系结构的设计和实例，令人遗憾的是多重机结构没有编入本书中。本书可以作为高年级本科生和研究生的教材，也可供计算机设计人员作参考。由于作者的水平和时间的限制，书中一定存在不少错误和不足之处，请来函指正。

本书是根据美国斯坦福大学 John L. Hennessy 教授和贝克莱分校 David A.Patterson 教授 1989 年的讲义并结合设计 RISC 机的经验编写而成的。在编写过程中，得到作者的支持和鼓励，Patterson 教授还把他的新著“计算机体系结构：一个定量的设计方法”(Computer Architecture: A Quantitative Approach) 寄赠给我。衷心地感谢 John L. Hennessy 教授和 David A. Patterson 教授和他们的著作给我的支持和鼓励。

感谢上海市政府在 1988 年至 1990 年中支持我们的“RISC 技术研究和应用”项目，使我们有机会接触计算机体系结构的设计。1989 年美国加州大学贝克莱分校 P. R. Gray 教授邀请我到贝克莱客座研究一学期，使我有机会看到 Patterson 教授的教学录像，并结识了早期 RISC 研究者 Sequin 教授，HP 精确结构精确设计者 Ruby B. lee 博士和 IBM Vojin G. Oklobdzija，他们的书籍和报告给我以后的教学和研究都有很大帮助。在此表示感谢。

本书得以出版是与复旦大学出版社林溟波先生的长期努力分不开的。他审阅了全稿，不论在书的内容，还是在文字上都给了我很多指正。在此向他表示深切谢意。

最后我要特别感谢我的同事任俊彦、程志宏、晁英伟和曾晓军等同志，他们为本书的出版做了大量的工作，并为 RISC 机的设计任务的完成起了主要的作用。本书的内容在研究生教学中试用两届，教学上的成功对我也是一种鼓舞。本书绝大部分是业余时间完成的，感谢我夫人蒋凤仙的支持和理解，她还帮助我作文字上的审阅工作。

复旦大学电子工程系　洪志良

1993.10

目　录

第一章 引 论

§1.1 计算机的分类和发展

自半导体集成电路问世之后，计算机总是随着半导体技术的发展而发展的。图 1.1 表示了 1966～1986 年各类计算机的发展情况。

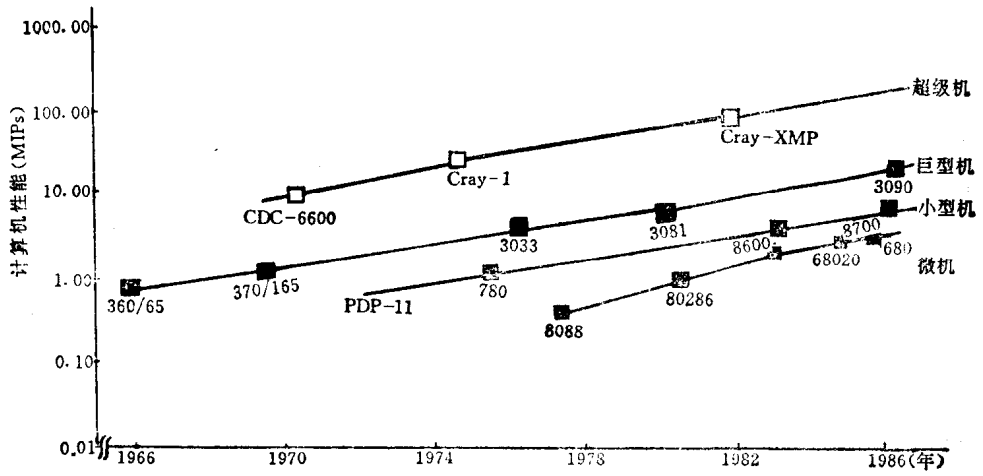

图 1.1 各类计算机性能增长曲线

在 1966～1986 年中，各类计算机性能增长的速率分别如下：

超级机：每年 19%

巨型机：每年 16%

小型机：每年 20%

微 机：每年 26%

其中巨型机的增长速率比其他类型计算机对工艺技术的依赖更大。超级机的发展既依赖于工艺技术，又依赖于结构。小型机的发展首先是结构改进。微机发展速度最快，其中一个原因是它直接利用了工艺技术改进的优势。当然微机的发展在很大程度上决定于结构的改进和新思想的引入。

事实上，八十年代已经产生了 RISC 结 构，它首先为计算机性能的改进提供了最有利的条件。而这种改进又恰恰是与工艺无关的。

图 1.2 表示了自 1984 年引入商用 RISC 后各类计算机(包括 RISC) 的性能增长速率。在这段时间内，计算机的性能每两年翻一番。当时的主要背景为：计算机市 场 大 幅 度增长，使结构创新更加容易。同时高级语言减少了对特殊目标码结构的依赖。标准化的与用户独立的操作系统降低了推出新结构的成本和冒险性，例如 UNIX 操作系统。

本书共有十章。第一章是引论，从中可以明确学习本书的目的和可能达到的结果。重

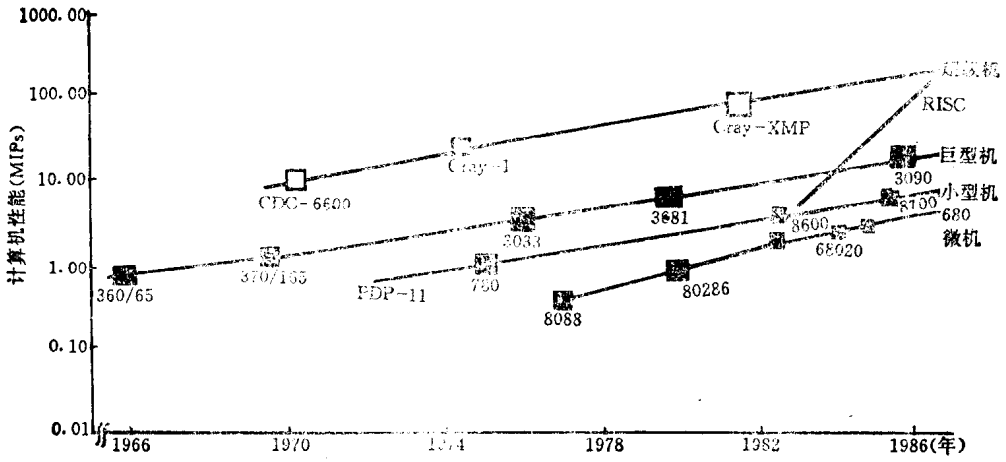

图 1.3 包括 RISC 在内的各类计算机性能增长速率

要的 Amdahl 定律也将在第一章中给出。第二章讲述性能方程。第三章讲述指令集设计。第四章介绍当前几种主要计算机的体系结构，同时也介绍了 RISC 结构。第五章讲述存贮器。第六章介绍执行过程。第七章介绍流水线操作。第八章介绍计算机的输入和输出。最后两章的内容是矢量机和高级流水线。尽管本书面很广，但尚有很多内容不能包括在内，它们是算法和数据结构，互相相关网络，故障容差、可靠性、专用机、协理器和多重机等等。

为了更好地理解概念，本书很多地方是通过实际的机器作为例子来讲清楚原理的，整本书贯穿着 RISC 机的设计。

本书的宗旨是要对现代计算机系统设计过程中的主要论题进行定量的研究。我们将集中精力在那些已经验证过了的概念，而不是把精力分散在尚未成功或已证明了只有有限生命力的那些概念上。我们讲述的绝大多数概念已经在商业机器中检验过。一旦某个思想出现，总是会在更新的实际系统中反映出来。

每章主要章节是叙述、展开关键的概念。这些概念的编排，使读者能最大可能地掌握这些概念。在最后给出每一章的部分参考文献。

§1.2 计算机设计者的任务

学习计算机体系结构，目的是要理解计算机的原理，从前人那里学习设计计算机的经验、设计出速度更快、性能更好的计算机。

计算机设计不仅仅是一门艺术，它是一门定量的科学。计算机设计不仅仅是描一张图，设计的机器应该能运行程序。计算机科学是一门牵涉面很广的科学，和计算机设计直接有关的有编译器、操作系统、逻辑、存贮器和封装等等。计算机设计者要从模拟建立体系结构，由实验、测试、分析到合成，来回反复进行。设计的机器的功能应该符合要求，除了软件的兼容性外，还要注意到操作系统和高级语言、地址大小对语言的要求、动态地址的再分配、保护和共享、多重处理、虚拟系统及虚址大于物理地址的事实、中断、故障及陷阱等等。

计算机设计要规范化，规范化的机器兼容性强。如标准接口能和 IEEE 的浮点处理机连接使用，对输入和输出设备应该有标准接口、标准总线，结构上力求其兼容性强，在市场上有竞争能力。确定一台计算机的好坏关键还是性能价格比，尽可能低的价格和尽可能高的性能，使性能价格比优越于其他机器。机器的性能价格比决定于机器的开发时间，市场大小和机器在市场上的生命力。

计算机的资源包括硬件和软件两部分。要密切注意硬件、软件和微码的发展动向。在某些地方，硬件和软件都能实现所要求的功能。在这种情况下，应该如何处理呢？要权衡速度、体积、成本诸方面的因素。一句话，看它的性能价格比。在这里平衡和协调是非常重要的。

§1.3 计算机体系结构

众所周知，标准包括价格和性能两方面。尽管性能价格比是一个公共的标尺（或许性能和价格作为标尺更好一些），性能和价格本身可以单独使用。在其他情况下，只有价格或性能目标指定之后，其他因素才能优化。系统整体必须满足功能上的要求和实现最高的性能。具备完善的各级硬件和软件是实现这个标准的必要条件。在应用和基本硬件之间不同级的资源有应用工具、编译器、汇编、微码和基本硬件。图 1.3 表示了各级层次中的资源。

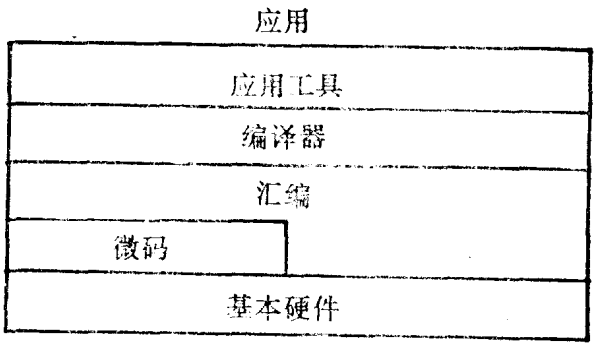

图 1.3　不同级可执行的资源

这些资源是在硬件中实现，还是在软件中实现，各有各的优点。用软件实现的优点是：错误代价低，软件中排错和改变更容易。用软件实现简单（需要较少时间），这是因为软件提供了更好的工具，使它更容易用抽象的东西（例如数据和过程抽象）。另外用软件实现协调也较简单。

用硬件实现的优点是性能较好。但这并不是说，用硬件实现总是意味着有更高的性能。若采用平行处理也能够提高机器的性能。

为达到要求的功能，设法选择最好的实现方法，其方法上常常用重复序列：实验/分析/合成/测量。每个步骤是：

(1) 运行一个实验（或许通过一个模拟器运行一个程序）；

(2) 分析结果；

(3) 根据实验合成设计；

(4) 测试效果，进入下一个实验。

紧跟当前形势进行设计，特别注意下列领域的发展：

技术（存贮器，IC 密度，大存贮）

性能增长(时钟速率等)

地址空间消耗(每年长$\frac{1}{2}$～1 位 VM)

Amdahl 规律

Rent 规律

软件发展趋势

汇编到高级语言

可以兼容端口和标准化的 OS

新的编程语言模型

更好的编译技术

设计计算机的体系结构，包括很多方面，需要考虑到：

IC 密度：每年单片集成的晶体管数增加一倍。基本器件的速度提高要稍微慢一些。

存贮器技术：RAM 密度每 2 年增加 1 倍。

磁盘技术：每10年容量增大 10 倍。

另外，像封装这样的因素也影响设计时组织的选择。

回顾一下过去，可以看到，用户消耗地址空间以每年多$\frac{1}{2}$位到 1 位虚拟空间地址的速率增长。当然，这是通过降低存贮器的价格和扩大单位面积的应用来达到的。忽略地址空间常常是指令集结构一败涂地的主要原因。Gordon Bell 曾经说过："地址空间大小是体系结构上难以修复的错误。"

设计者常常面临的问题是如何平衡他们的系统：多大存贮器、多大磁盘和多快的 CPU 相匹配。Amdahl 的回答是"1 MIPS 速度（每秒执行 1 百万条指令），1M 字节大小存贮器和每小时 1 M 位输入/输出的外围相匹配"。

计算机系统结构既要考虑硬件技术和计算机应用趋向，也要考虑软件的重要倾向。或许过去 20 年中，最重要的趋向是汇编语言由高级语言作为主要编程语言所代替。这已经成为编译器的主要角色，结构趋向将支持编译器。和这一个倾向相平行的是用高级语言写操作系统。这些可以兼容的操作系统已经发展成为标准的操作系统，与具体用户无关。

编译技术也在不断改进，体系结构必须理解新的编译技术，因为编译器是用户与机器之间的主要接口。

比编程技术发展更慢的是新编程模型的产生和出现。我们能期待它的编程语言在功能上和抽象性上不断增加，但迅速的改变恐怕不会出现。

计算机设计基础：

响应与产量之间的矛盾；

Amdahl 定律（减少返回，机器的性能受最慢部分限制）；

90%～10%规则（程序定位）；

命题、反命题和发明的合成模型。

对性能的理解，有两种不同的观点。对计算机用户来说，希望更快得到响应，计算机性能就是响应时间。对计算机管理员来说，希望每天完成更多的任务，即产量。

在计算机设计中，产量和响应是一对经典的矛盾。这里不仅有性能问题，而且也有硬件的复杂性问题。典型情况下，是按照某一基准测量其响应，用这样简单的方式来评估机器。但在某些环境下，我们应该更多地测量产量，例如交互处理环境中计数是每秒钟转换，一些商业基准也确实是测量这个参数。

现在来看一看下面的情况是属于响应、产量或是两者兼有之。

高速本地存贮器——应该是两者。

高速时钟——应该是两者。

单独执行任务的多重处理器——产量（在这一级是交换处理）。

科学问题的平行处理——应该是两者。

非公正的编排——如在多重处理器系统中一个处理器对总线享受最高优先权——响应。

在文件处理过程中磁盘控制器或磁盘性能——应该是两者。

计算机设计的另一个基础是 Amdahl 定律。它的基本说法可以概括为：计算机的性能受其操作最慢部分限制。Amdahl 原先在平行处理中叙述这个定律如下：从这一点可以得出一个重要的结论，就是扩大实现平行处理率，除非在串行处理时能由相近速率来完成。解决的办法是减少返回，除非一个机器是整体一致加速的。图 1.4 以形象的方式说明了 Amdahl 定律。

假设用两种方式进行计算（快的和慢的）。我们感兴趣的是快的方式，并想知道采用了快的方式之后加速了多少。这里加速的意义是同时采用快的和慢的两种方法的速率与只采用慢方法的速率进行比较，并假设整个过程完全采用快速是不可能的。其结果将告诉我们采用快速方式的性能明显地会提高。

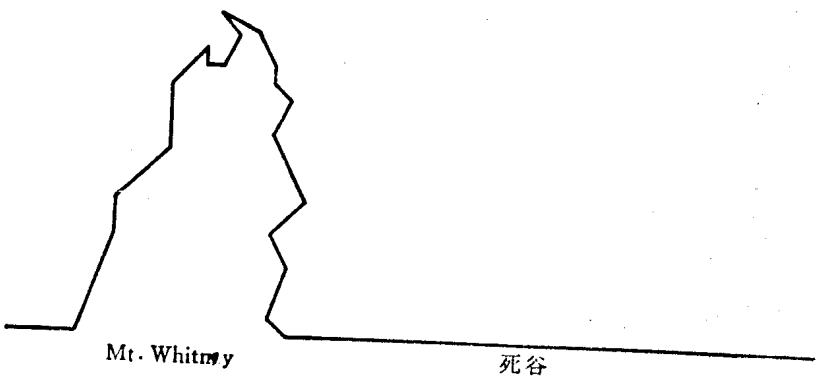

图 1.4 Amdahl 定律示意图

爬越 Mt. Whitney	穿过死谷	穿越死谷加速	总加速
爬越 Mt. Whitney 花 40 h	步行：60 h	1	1
	自行车：20 h	3×	1.7
	吉普车：3 h	20×	2.3

美纳年：	30 min	120×	2.5

如果我们假设在慢方式下时间为 1，那么在快速、慢速兼用的情况下的加速应为

$$s=\frac{1}{(1-f)+\frac{f}{k}}$$

式中 s 为加速率；f 为在快速时花的时间因子；k 为快速对慢速的比例。总时间中有 f 这部分是被加速了。

下面举一个具体的例子来说明问题。

[例 1.1] 假设采用快速方式的速率是采用慢速方式的 10 倍，但只有一半时间可以在快速方式下工作，由此总过程加速率为

$$s=\frac{1}{(1-0.5)+\frac{0.5}{10}}=1.8$$

Amdahl 定律告诉我们：如果计算机快速操作只是程序的一部分，不管我们如何加速，也不能加速到快速因子的倒数。如果采用快速方式下无限地快又会如何呢？

由程序的性质产生一些重要的基础，最重要的程序性质是地址再分配。有几个不同的方式来叙述这一原理。其中 90%～10%规则是由 Knuth 叙述的：程序花 90% 的时间用在 10%的编码上。这隐含着我们能相当准确地预示程序那一部分会接着执行。这就是，数据最近碰到的那部分相当可能不久会再遇到。

§1.4 计算机体系结构设计初步

在计算机设计中反复强调的几个方面是：

使频繁出现的情况快；

充分吸取程序行为的优点；

转移出现的频率和转移的距离；

吸取地址分配的优点——存贮器层次工作；

建立平衡系统；

如果要成功地设计一台计算机，它必须是适当地平衡——至少它瞄准了应用市场。不平衡机器不可避免地性能价格比要下降。根据 Amdahl 定律，最弱的环节会有致命的影响。但是，机器其他部分价格的增加意味着性能价格比向负的方向变化。

[例 1.2] 设我们用 5 倍价格使 CPU 速度提高 5 倍。并假设 CPU 占用 50%的时间，CPU 占基本价格的 1/3。请问这是不是一个好的思想？

首先看一台 Neumann 机，因为它用于传输主存和处理器之间的数据，也用于传输二级存贮器与主存之间的数据，总线带宽和由存贮器提供的带宽及存贮器决定了 CPU 能看见的数据和指令的速率。

增加存贮器带宽和减少存贮器存取的响应时间，这两者的要求是这样的苛刻，以至于许多人集中精力来改进结构。愈小的存贮器，也是愈快的存贮器。实际上，如果存贮器小的话，就需要更昂贵的技术来制造它。程序一般有位置属性，这里有这样的可能性，如果某一项最近被访问过，那么程序将访问已经给出的指令和数据的可能性更高些。

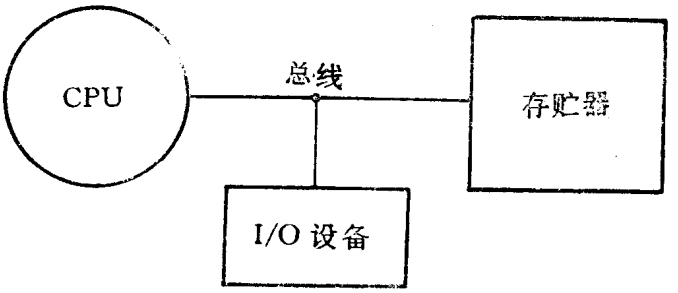

图 1.5　Neumann 机结构原理图

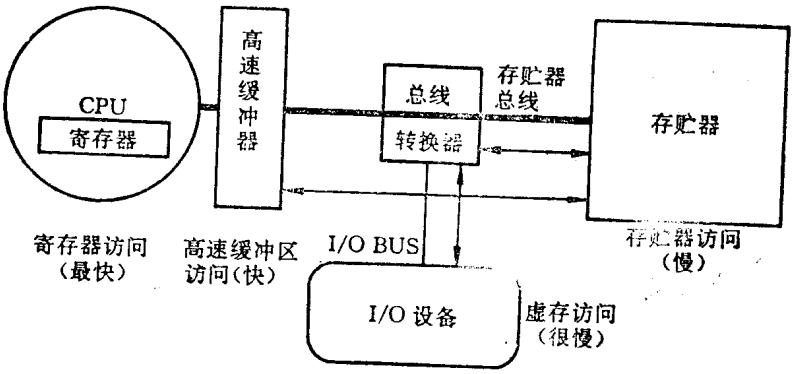

图 1.6　典型的存贮器分层体系

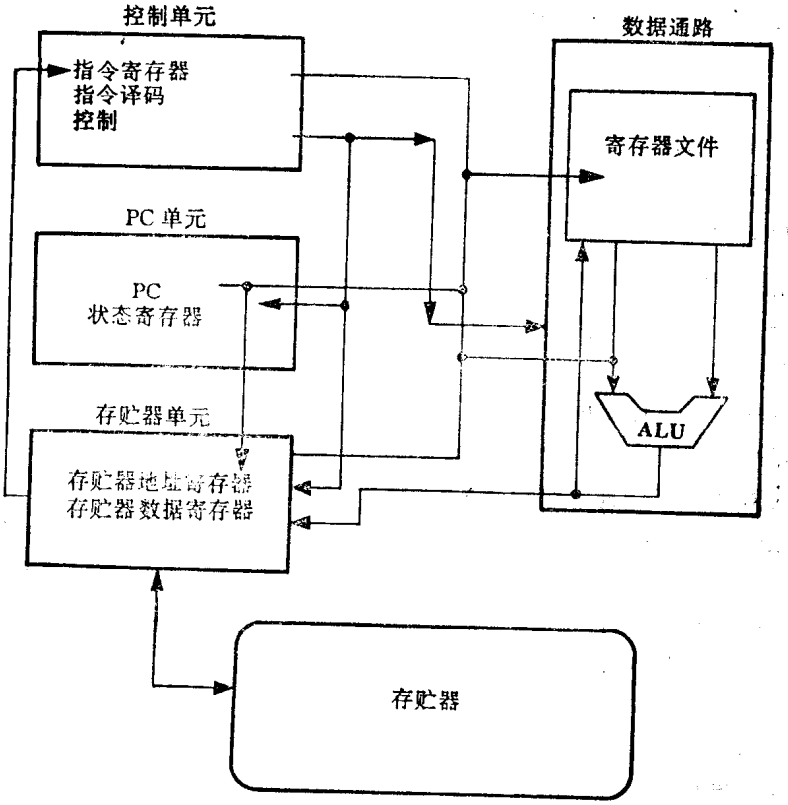

图 1.7　典型的 CPU 组织

图 1.6 表示一个典型的存贮器分层管理系统。在这个系统中，共有四级存贮器，它们的关系如下：

层次顺序	存贮器名字	它由谁来分配	它由谁支撑
(1)	寄存器	编译器	高速缓冲区
(2)	高速缓冲区	硬件	主　存
(3)	实际存贮器	操作系统	磁　盘
(4)	磁盘存贮器	操作系统	磁　带

如果存贮器系统每层的产值与响应发生变化，则会影响整体速度和机器的其他组织。

图 1.7 表示了 CPU 的整体组织。对每一条具体的指令来说，典型的执行顺序为：

(1) 取指令和指令译码；

(2) 操作数译码与取操作数；

(3) 执行；

(4) 修改 PC。

让我们运行一条典型的指令来看执行过程中的每循环的功能。首先看一条指令：Add R1, R2, R3。

在取指令循环中的作用是：

(1) 送 PC 到存贮器单元，并存入存贮器地址寄存器。

(2) 开始存贮器循环，将数据返回到存贮器数据寄存器中。

(3) 将指令由存贮器数据寄存器移到指令寄存器。

(4) 指令译码，产生 Add 的控制信号。

在执行循环的操作步骤是：

(1) 由寄存器文件和在 ALU 中取寄存器操作数。

(2) 执行 Add。

(3) 把结果存回到寄存器文件。

现在考虑一下更复杂的指令：

Load R2, 48(R1)

取操作数和译码循环作用如下：

(1) 由寄存器文件读 R1，并将 48 装入 ALU 锁存器。

(2) 对有效地址部分进行加的运算。

(3) 把地址送到存贮器地址寄存器。

(4) 存贮器操作。

(5) 将存贮器数据和寄存器内容送入寄存器文件。

最后让我们考虑 R1＝0 时的转移指令：

BZ R1, 60(PC)

这是一个与 PC 相关的转移。在操作数译码的那个循环，计算 60＋PC；并将它临时锁存起来。在执行循环将 R1 与 0 进行比较。如果 R1〈〉0，那么 PC 如通常那样递增。否则，由有效地址计算出来的值用来代替 PC。