第一章　Power PC 概念简介

本章包含以下内容：

- 计算机体系结构，尤其是 RISC 体系结构的基本概念。
- 计算机体系结构在个人计算机处理中扮演角色的概述。
- 一些本书使用的非正式术语定义。

在本世纪的大部分时期内，技术领域的竞争反映了科技的进步。而在第一艘宇宙飞船超越太阳系的同时，人类的兴趣也突破了原来的范围。今天，计算机功能的增强和体积的缩小成为衡量技术工艺的尺度。在 20 多年的时间里，个人计算机已经是人们日常生活中必不可少的一部分了。

1.1　计算机简史

在不长的发展历史中，实际上个人计算机中每一部分的发展都是极其迅速的。价格持续下降，但性能不断增强——技术的进步好像从来没有减慢过，特别是越来越多的应用程序和集成环境持续不断地增加各种选项。

虽然微处理器设计和这些发展不无关系，但是基本的体系结构——制约所有的软件和硬件设计——早在调制解调器、引导、计算机窃贼、向后兼容等词汇为人们熟知之前已经定义了。体系结构按照当时的技术设计，并且不断地影响现行个人计算机各主要方面，它要求纷繁芜杂的处理器保持兼容。

当今的个人计算机依然沿用 70 年代的体系结构，但这并不是因为计算机体系结构跟不上技术的发展。在 80 年代初期，出现了一种微处理器设计的新方法，称为简化指令集处理（或 RISC），它迅速发展并统治了新处理器的设计。——为功能异常强大（而且价格昂贵）的工作站提供其所需的速度、效率和精度，这个市场整个由 RISC 体系结构控制。也许对 RISC 体系结构最令人信服的重要证据是在过去的五年中，所有的新处理器设计都采用了 RISC 技术。原来唯一的复杂指令集处理（或 CISC）处理器设计的作用已经改变了，它只适用于更宽的总线结构以及配合浮点运算（运算协处理器）。

注：

在个人计算机处理与电视和其他形式的电子通信设备紧密结合之后，人们对微处理器的要求越来越高。问题不再是个人计算机是否转向 RISC 设计，而是什么时间开始。

然而，虽然在今天的社会中年轻人和 TV 商品广告已使用 Super VGA、波特率及闪速 ROM 等用语，但是计算机体系结构的概念仍然令人感到陌生和难以理解。而其主体，处理器实际上仍然是一个使用户感到束手无策的神秘黑盒子。

1.2 术语和定义

虽然每台个人计算机中都有一片微处理器，但是大多数用户却无法清楚地看到微处理器的内部工作情况或者了解微处理器设计所遵循的体系结构定义。

因此要理解体系结构变革的意义，则了解用于描述微处理器，特别是 RISC 微处理器各部分的专用语言是很有用的。

1.2.1 体系结构

术语体系结构在计算机产业中有非常丰富的涵义。对于微处理器体系结构而言，它是指微处理器或者微处理器系列的设计规范。体系结构包括指令集(最重要的)、编程模式(包括寄存器组和存储器约定)、异常模式以及表示兼容处理器特性的其他定义。

体系结构的指令集部分通常称为指令集结构，或者称为 ISA。ISA 定义指令的功能、指令格式以及寻址方式。

1.2.2 简化指令集处理

简化指令集处理(RISC)包含的指令将其操作最大限度地简化，从而使大多数指令严格地在相同的时间内执行(时间一般相当短)，指令机器码的长度基本固定，而且指令使用简单的寻址方式。这样处理器就可以轻松地对这些指令进行译码，反过来 RISC 又加快了指令的执行速度并简化了处理器设计。

RISC 处理器中最显著的特点是其运算指令读取源操作数和写入结果的场所是寄存器而不是系统内存；内存仅由意义明确的 load/store 指令存取。因此，RISC 体系结构有时被称作寄存器——寄存器体系结构，或者 load/store 体系结构。RISC 体系结构没有子集。

然而 RISC 的简单定义并未说明实现符合上述标准的指令集的优越性。关于其优越性的讨论需要引入一些和 RISC 处理器紧密相关的术语，这些术语——诸如流水线操作(pipelining)、超级标量、并发执行等和 RISC 的定义很难截然分开。因而这些技术得益于规范的 RISC 指令集，由于大多数 RISC 体系结构和某些 CISC 处理器一样实现了流水线技术、并发处理、超级标量分派。

1.2.3 流水线技术

此技术有多种形式，例如总线流水线技术和指令流水线技术。简言之，流水线技术就是

在前一个任务结束时开始另一个任务的能力。

指令流水线操作策略使得指令处理任务被分解为简单的几个阶段——例如读取、译码、分派、执行和回存。指令一个阶段接一个阶段地执行，当它在流水线操作中一步步移动时，它留下空间供后继指令逐个阶段地执行。这就好比是在同一个工作间同时完成各种工作的装配线。当一个阶段的工作完成后，产品就进入下一个阶段。

很明显，这样的安排有很多好处。在指令执行过程中，资源不用闲置在一边。由于一条指令执行需要三个、四个或更多时钟周期才能完成时，一旦管道中满载指令，那么在每一个时钟周期都有指令完成执行。

1.2.4 超级标量

超级标量处理器中包含多个执行单元，同时可以分派和执行多条指令，术语超级标量(superscalar)的产生是由于它介于标量计算机(计算单个数值或标量的常规计算机)和矢量计算机(用于科研和工程矢量计算的高性能计算机)之间。

1.2.5 执行单元

执行单元是执行指令的单元。常见的例子有整数(或定点)单元、分支处理单元和浮点单元。

1.2.6 浮点单元

浮点单元是完成浮点运算的执行单元。浮点运算使用非常精确的数值进行数学运算，用于科研、工程以及图形计算。浮点单元就是安装主板上的数学协处理器芯片，通常缩写为FPU。

1.2.7 整数单元

整数单元是完成整数运算的执行单元。整数运算进行整型数值(更准确地讲，是定点数值)的数学运算，其精度不如浮点运算，通常缩写为IU。

1.2.8 指令并行性

指令并行性是指同时执行多条指令的能力，这是超级标量处理器的一个特征。例如，某操作的实现需要一个整数单元和一个浮点单元，那么两种指令可同时分派到相应的执行单元，并可同时执行。指令并行处理的好处在于在给定的时间内可执行多条指令。并发执行的优点对于只有一个收银员值班时在杂货商店结帐排队的人们而言就明显。

1.2.9 执行时间

执行时间(latency)为完成特定任务或任务组(如指令执行或一个内存访问)所需的时间。

1.2.10 通过量

通过量是一个给定时间内完成执行及记录结果的指令数目。对于超级标量或流水线技术处理器而言,通过量比执行时间更适于作为衡量处理器指令执行效率的标准。

1.2.11 正交性

正交性是指指令集中不包含重复功能。RISC 指令集是典型的正交集,它只完成相互不重复的基本任务。

1.2.12 保留区

保留区是当分派的指令在无法进入执行单元的流水线时提供它们在执行单元前端排队的缓冲区。指令无法进入流水线的主要原因是管道忙,或者在同一个时钟周期内为同一个执行单元分派了多条指令。

1.2.13 实现

在本书中,名词"实现"涉及一个芯片,尤指根据 Power PC 体系结构设计的微处理器。嵌入式的控制器也可认为是一个实现。

1.3 小 结

本节简要描述了处理器体系结构及其在个人计算机设计中所处的地位。同时还介绍了一些和 RISC 体系结构有关的最基本概念,它们对于本书后面展开的讨论将大有帮助。

下一章讨论体系结构设计的方法,主要着眼于 RISC 体系结构的特征和有关技术以及 RISC 体系结构为个人计算机产业带来的影响。

第二章　体系结构的定义及原因

在本章中，将学习以下内容：

■RISC 和 CISC 的基本特征

■RISC 处理的有关基本概念以及其如何提供性能

■优化技术的例子，例如流水线技术和指令并行性

如果在波士顿旅行过，就可以理解为什么需要 RISC 体系结构。在波士顿，街道狭窄，道路曲折多弯，就像一个迷宫，游人很容易迷路。与此类似，当今个人计算机中的微处理器在原有设计不断提高的同时也受到许多相似的瓶颈困扰。

波士顿交通问题的原因是很容易想到的。在 18 世纪，当时的聪明人把街道和便道设计在人们经常行走的地方是有道理的，以后这些马路和小路就变成了现在的街道和便道。这里很少发生交通事故，各种交通工具都很慢，马车也很小，因此狭窄的街道、看不清的拐角、很急的转弯以及错综复杂的路口不会产生什么问题。

波士顿的城市建设者没有预计到科学技术和人口的增长。现在成千上万的轿车、卡车和公共汽车在伟大的美国思想家曾经漫步和骑马的狭窄的街道中艰难地向前挤。虽然波士顿仍不失富于创造、充满音乐、拥有丰富文化和美味的名城，但是它同时为以后的城市规划者提供了一个很好的反面教材。

微处理器设计同样会遇到类似于波士顿城市规划的难题(哈佛体系结构除外)——拥挤的交通竞争狭窄的数据通道，技术的惊人发展以及相应的复杂技巧的应用越发造成了这种局面的出现。再者用户继续为保持向后兼容能力而付出重大代价。

2.1　CISC 和 RISC 处理的对比

在 CISC(复杂指令集计算)体系中，增加了许多复杂指令以满足特定的需求，这就像城市的改造需要增加新的街道，城市逐渐扩大融合了更小的乡镇并且边界变得无法分辨了，这又正如扩充的芯片形成了浮点单元(FPU)、高速缓存和内存管理单元。

RISC(简化指令集计算)体系采取了另一种微处理器设计方法——借鉴生产流水线的思想，这是导致工业革命的一项重大革新。在生产流水线出现之前，操作工人必须了解全部生产过程。只有了解全部生产过程的人才知道生产对象。这对于生产的安全性很可靠，然而要想生产批量产品就困难，而且还会因骨干人员生病等原因而导致整个生产的停顿。

有了生产流水，作为一项任务的操作被分解为许多简单的步骤，每一个步骤完成的时间

都是相等的。生产人员只需接受一个生产阶段的训练。他们接过上一阶段的产品，完成本职的工作(他们对此是精通的)，然后再传向下一阶段。在这种方式下，流水线上的十个经验较少的工人可能比十个熟练手艺工人在一天中生产出更多的产品，而且关键的过程是完全不变的，于是性能可以得到优化。

在处理器设计中，生产流水线以流水线操作代替。在完全优化的 RISC 处理器中，每一阶段的工作在同一段时间内完成(一时钟周期)。

表 2.1 展示了一个包含四个阶段的流水线操作示例——读取、译码/分派、执行和写回。

表 2.1 简单的流水线操作示例

时钟周期	读取	译码/分派	执行	写回
1	Inst1			
2	Inst2	Inst1		
3	Inst3	Inst2	Inst1	
4	Inst4	Inst3	Inst2	Inst1
5	Inst5	Inst4	Inst3	Inst2

如表 2.1 所示，在第一个时钟周期中，第一个指令(Inst 1)从存储器中读取。

在时钟周期 2 中，Inst 1 进入译码/分派阶段，处理器分析组成指令的每一位以确定后续的操作，然后把指令送入执行单元。因为 Inst 1 退出了读取阶段，所以 Inst 2 可以被读取了。

在时钟周期 3 中，Inst 1 被执行(执行包含算术逻辑运算、读出或存入操作、或者其他任务)。Inst 2 被译码，Inst 3 被读取。

在时钟周期 4 中，Inst 1 写回结果，处理完成，其余的所有指令向前进入下一阶段。此时，整个流水线全满了，虽然处理每条指令花费了四个时钟周期，但是从现在开始(至少在这个虚构模型中)，每个时钟周期可以执行一条指令。

每个时钟周期可以执行的周期数称为通过量(throughput)，它提供了一个比指令执行时间更恰当的关于流水线处理器的量度。指令执行时间(instruction latency)说明执行一条指令所需的时钟周期。

吞吐量

每个时钟周期可以执行的周期数。

指令执行时间

一条指令执行完成所需的时间。

在流水线执行单元中，执行时间对应于阶段的数目；但是，按照通常的看法，流水线中有多少阶段是无关紧要的。在流水线充满之后，假定指令不受任何干扰，那么这个流水线的通过量达到每时钟周期一条指令。因此，在流水线模型中，执行时间主要是何时充满流水线的主要因素。

此处应该重申许多 CISC 处理器也采用流水线技术，有的还广泛应用，但是 RISC 处理

器的设计最大限度地利用了该技术。

流水线技术

在第一个任务完成之前开始第二个任务的能力。指令流水线技术把指令处理简化为简单的阶段(典型的为读取、译码、分派、执行和写回)。在指令通过流水线时,它退出一个阶段并进入下一个阶段。

指令通过量

说明给定时间内可以执行多少指令的比率,通常表述为每个内部处理器时钟周期的指令数。

然而流水线技术仅仅是妙招的一半。几个流水线还可以并发执行。对于相互没有影响的独立任务,芯片设计者(或者生产流水线设计者)可以建立另一个并发执行的流水线,并在下一个阶段把结果综合在一起。在一个流水线处理浮点数学运算时,另一个流水线可以处理整型数学运算,而第三个流水线可以处理读出和存入指令,等等。值得注意的是,必须有一种机制保证指令顺序写回(虽然具体顺序要求并不严格)。随着指令流水线数目的增加,维持顺序的机制变得越来越复杂了。

在另一种情况下,如果对一个流水线的结果需求很高,以致它无法跟上其他流水线的速度(也就是说,它成为关键的过程),设计者可以建立另一个类似的流水线,以双倍的速度产生结果。许多处理器具有多个整型单元,以便同时执行两个整型指令。

注:

同时执行多条指令的能力称为指令并行性(instruction parallelism),实现多流水线的处理称为超级标量处理器。

如果指令的设计和处理正确,而且执行这些指令的硬件容许指令并存,那么很容易看出超级标量流水线机器可以进行优化,每个时钟周期执行多条指令。和流水线技术一样,CISC 处理器也采用指令并行,但是因为 CISC 指令集的复杂性,所以其实现相当复杂,从而导致芯片非常大。可以看出,RISC 体系专为流水线技术和指令并行设计,这就说明了 Power PC 芯片如何以较小的芯片和较低的电力消耗来提供更高的效能。

注:

RISC 体系结构在 70 年代中期首次公布。最早的例子是用作高速电话交换系统的处理器设计。其指令集包括处理器中存储器和寄存器组之间传送数据的简单指令。虽然这个方案没有完成,但是其基本原理被 IBM 和一些研究部门深入发掘。80 年代初,商用 RISC 处理器出现,主要用于强大的工作站计算机和专用系统。

2.2 价格和性能的对比

所有处理器、计算机和外设的设计目标是性能价格比——最佳的性能对应尽可能低的价格。CISC 处理器试图通过增加微代码(microcode)执行更复杂的汇编语言指令的方法获得

更高的性能。

微代码

指 CISC 处理器中执行一条指令的微指令。

提高性能的方法是采用汇编语言(直接跟处理器或体系结构打交道的基础语言)。汇编语言比那些指令基本相同的高级语言(例如 C)更加有效。缩小称为汇编语言和高级语言之间的"语义代沟"可以使性能得到提高。然而,随着指令变得越来越复杂,芯片上要完成译码和执行指令的微代码也变得越来越复杂了。

因此 CISC 指令集总是包含用于完成不常见任务的各种指令。例如,VAX 具有处理多项式的汇编语言指令。虽然一些科技方面的应用程序需要此类运算,但是对于大多数应用程序则没有必要。

要求长序列微指令的指令执行时间也较长。这就使得难以充分利用流水线和指令并行的优点。在流水线中,所有的后续指令,无论它们需要多少工作,都不得不在前面的耗时指令后面等待。

在杂货店中处理同样问题的方法是建立高速便道,通过模拟,处理器可以实现另一类指令并行,把耗时的指令送入一个流水线,而把快速指令送入另一个流水线。但是,因为这些指令有可能相互依赖,所以这些快速指令仍有可能必须等待那些耗时的复杂指令执行完成。

在过去的 30 年中,晶体管已经变得越来越便宜了,而且工艺处理不断地使它们更小、更快了。存储器、电力要求、晶体管限制和其他大多数物理限制对芯片设计者的要求比 1/4 世纪以前宽松得多了。因此选择倾向于使用这些经济、小巧的晶体管,再加上较复杂的指令,或者认识到存储结构和逻辑电路现在不再昂贵且易于集中,因而可以回头重新考虑整个设计方案。

我们不能处处拿波士顿举例,但处理器体系结构是另外一个教训。不断增长的费用只能获得不太大的收益——包括消耗费用和芯片大小和电力要求——成为导致 RISC 设计声名鹊起,令人信服的原因。

从头开始的主要缺陷是为原有的处理器编写软件方面的巨大投资,但是 RISC 处理器具有强大的性能,它可以模拟执行为原系统编写的旧软件。

因此不在复杂指令上使用晶体管——它们很少用到——RISC 处理器把它们用在扩大流水线的芯片逻辑上,从而更容易地实现并行和超级标量设计。这样的硬件优化了通过量,使许多指令共同受益。芯片上的许多部件同时工作。读取、分派、执行和前置机制越复杂,芯片上的存储部件,诸如高速缓存(cache)、TLB 和寄存器占用的空间越大。然而,这样的硬件比用于独立的复杂指令处理的相同晶体管工作更加有效。

高速缓存

处理器直接访问外部存储器的条件下读写指令和/或数据的专门存储。

2.3 RISC 设计基础

为了获得较高的通过量，RISC 设计坚持几个相互关联的基本原则：

■连续编码的均匀长度指令

■基本的简单指令。复杂的任务通常要求可以流水线和并发执行的多条指令。

■基本寻址方式下的寄存器—寄存器(读出/写入)体系结构。

■允许独立访问各寄存器的大型寄存器文件。

这些规则的好处在下面的章节中讨论。

2.3.1 均匀长度的指令

使所有的指令具有相同的长度简化了指令读取、译码和分派的设计，使流水线技术和寻址计算更加灵活。

例如，如果所有指令的长度为一个字(32 位)，那么读取部件可以认为指令以 32 位为其界限。如果指令缓存于指令队列(如第一台 Power PC 处理器 Power PC 601 中)，则高速缓存中的指令存取可以容易地按照指令读取数进行检索。

Power PC 601 处理器

第一台 Power PC 处理器，于 1993 年中期面世，基于基于 Power PC 的 IBM 和 Apple 系统。

类似的，601 中的指令队列长度为 8 个字(指令)，和高速缓冲块的大小一致。当一块指令读入缓存时，不可能出现指令位于两个高速缓存块的情况。

指令均长还使分支指令的标识和处理得到简化。通过事先探出分支指令，可以提前知道分支去向(指令流中的分支指令可以用预定目标指令取代，意味着需要执行的指令更少)，而且预定目标指令可以事先验证其是正确的，还是不正确(需要处理)的。有些处理(例如 601 和 603)甚至在分支处理之前就已开始虚拟执行来自目标流中的指令。由处理分支指令所需的时间引出了许多技术，它们都因均长的指令而得以简化。

2.3.2 简化的指令任务

把复杂操作分解为简化指令使指令摆脱了竞争而造成阻塞及数据依赖。例如，如果读出或写入操作在存取高速缓存或系统内存时遇到延迟，那么整型单元(integer unit，IU)或浮点单元(floating unit，FPU)不会被强制保持空闲。相反地，它可以执行下一条指令。这就使指令容易通过流水线。

整型单元(IU)

完成整型,或者较精确的定点计算。整型运算不要求浮点运算所需的精度。

浮点单元(FPU)

执行浮点运算,以及要求高精度的算术运算。

具有可预定执行时间的简化指令有可能达到每时钟周期一条指条的通过量。601 整型单元中实现的大多数指令要求在一个周期内执行并具有每时钟周期一条指令的通过量。类似地,虽然 FPU 中有三个阶段,然而一旦流水线充满,则可以得到每时钟周期一条指令的通过量及每两个时钟周期一条指令的双精度通过量。

2.3.3 寄存器—寄存器结构

寄存器—寄存器操作使流水线操作更容易实现,因为大多数指令可以设计为在一个周期内执行。

算术指令使用寄存器文件保存源操作数和目的操作数,而不是直接存取存储器(高速缓存或系统内存)。单独的读出和写入指令用于从存储器传送源数据和结果数据。通过使这些任务从其他任务中分离出来,ALU 被解放出来并且在写入指令向存储器写上一个结果的同时开始计算。

类似地,编译技术的进步使编译也受益于指令独立,这样就使高级语言程序员受益于改进的性能,而不受到它的纠缠。

典型地,这两条简化指令可以实现占用一个周期,从而简化了流水线技术。

建立附加路径(称为前置路径)可使结果数据在写回寄存器文件之前作为后续相关指令的操作数。它避免了流水线阻塞。

前置

减少数据依赖导致的指令执行时间的技术。

2.3.4 大型寄存器文件

通过提供大型寄存器文件,多条指令在寄存器资源竞争激烈的情况下可以同步并发地执行,而且可以实现更多的设计规划,诸如更名寄存器和前置(缩短由于数据依赖或者寄存器竞争引起的执行时间)。

增加寄存器资源强调的是再利用而不是丢失操作数。大多数 RISC 算术指令的语法为三操作数——两个源操作数和一个目的操作数。目的寄存器不作为源寄存器,这样避免了数据丢弃并有利于数据再利用。

第五章"Power PC 体系结构的起源"回顾了 RISC 体系结构的历史并说明了 RISC 设计原则是如何从 1975 年中 RISC 处理器第一次尝试中发展起来的。

2.4 实现的广度

早些时候，为个人计算机设计的处理器不得不受限于当时的技术——电力要求、大小、晶体管精度、存储费用和外设的速度。虽然大型机和超级计算机的体系结构中实现了一些高级技术，但是没有微处理器大小、费用和电力限制的定论。

因为所有限制都加诸于处理器设计，看来个人计算机不会威胁大型机统治的看法不大可靠。这已经被证明是错误的。

然而，功能强大的中央大型机在技术上仍扮演着举足轻重的角色。即使 PC 已经取代了终端和信关(它们曾经是专用的、单用途的设备)，大型机体系结构和操作系统仍然为安于现状的当前微处理器体系结构设置了一个无法逾越的鸿沟。

注：

现有的微处理器体系结构侧重于特定类型的实现。许多第一代 RISC 芯片适用于高端科技、图形工作站和小型机，而新一代 CISC 芯片仍然最适用于桌面系统、服务器和低端工作站。

目前大型机实现、工作站、桌面和便携计算机有理由共享相同的体系结构，而且 Power PC 体系结构可以有意设计为提供广泛应用的通用基础——从电池供电、手持设备到与主型机竞争的多处理器系统。

前四个 Power PC 处理器表明了 Power PC 体系结构适合的实现广度。这四个处理描述如下：

- 601 处理器针对桌面和工作站系统。它也可用于和主型机计算能力相抗衡的多处理器系统。它采取 32 位寻址。Power PC 601 处理器如图 2.1 所示。
- 603 处理器也采取 32 位寻址，为低耗能系统，诸如方便使用的笔记本和手持计算机，以及其他电池供电的设备。
  Power PC RISC 体系结构要加简化这一事实使其特别适于空间、散热及电力消耗要求严格的实现。Power PC 603 处理器如图 2.2 所示。
- 604 处理器是今年推出的，也意在桌面市场，也采取 32 位寻址，但设计上与 601 不同，它具有更快的速度。
- 620 处理器是第一个实现 64 位寻址的 Power PC 芯片，针对高端工作站和小型计算机实现。

注：

第七章“Power PC 处理器系列简介”概述了各种 Power PC 处理器。尽管每一种芯片设计都有明显的不同(它证明体系结构的多面性)，但是所有这四种处理器支持相同的软件和操作系统。第四章“操作系统和 Power PC 体系结构”讲述操作系统支持方面的详细信息。

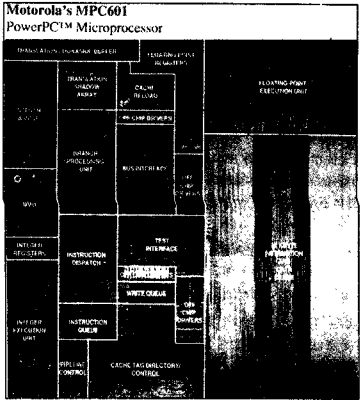

图 2.4 Power PC 601 处理器

2.5 对于用户而言所有这些意味着什么

至此，我们已经回顾了微处理器内部的有关问题，介绍了一些概念并定义了一些 RISC 使用的术语。至于用户，不可能测量这些特殊概念在 PCI 总线、面向对象操作系统、EISA 接口和语音识别系统领域中的重要性。其证据在于计算机制造商如何利用这个速度。从全部迹象上看，需要做的不只是制作用于商业应用程序的高端计算机。相反地，IBM、Apple、以及其他公司正在建立使用户计算机运行更快的系统，并使用户计算机完成以前甚至未曾想到的工作。

再也不会出现像成千上万的波士顿司机争抢着通过狭窄的格林汉姆大街冲向洛根国际机场的情景了。

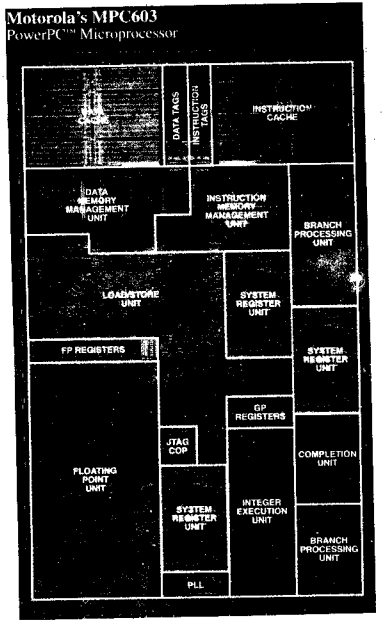

图 2.2 PowerPC 603 处理器

2.6 小　结

本章回顾了 RISC 体系结构的一些基本概念和原理，从当前 CISC 设计存在的问题以及通过简化实现来避免这些问题的方法开始，然后介绍了固定长度的指令以及允许计算指令从芯片寄存器，而不是从物理存储器中存取数据和记录结果。

下一章简要介绍筹划基于 RISC 的新型 Power PC 体系结构的 IBM、Apple 和 Motorola 同盟，以及高性能计算机系统的新家族。

第三章　Power PC 同盟

本章回顾了 Power PC 同盟头两年中发生的下列事件：

■Power PC 处理器的宣布
■基于 Power PC 系统的宣布
■Power Open 联盟的重大事件
■不同硬件和软件适配器的宣布

在 1991 年 10 月，IBM、Apple 和 Motorola 联合声明他们共同设计一种新的体系结构，以形成下一代 IBM 和 Macintosh 个人计算机系统的基础。在合力投资了上亿美元之后，这三家公司在得克萨斯的奥斯汀创建了"大变革设计中心"(Somerset Design Center)。

3.1　同盟的目标

对于 IBM 和 Macintosh 而言，汇融 Power PC 系统的能力是最有吸引力的。Power PC 同盟为兼容性提供了更大的便利以及更加有效的混合网络环境。这次合作的成果是 IBM 认证的 Apple 令牌环卡。相互之间的关系对其他公司也产生了相应的影响：

■Kaleida Labs 是在 IBM 和 Apple 共同投资的基础上出现的新公司，为迅速成长的多媒体产品建立了通用标准。
■Taligent 是一家新公司，也是在 IBM 和 Apple 合资的基础上产生的，它开发了一个面向对象的操作系统，于 90 年代中期完成。
■Power Open 是在联合基础上的产物，旨在开发一个 UNIX 操作系统的新版本，综合了 IBM 的 AIX 和 Apple 的 A/UX 操作系统的特点。这个开放系统平台允许用户访问 Macintosh 和基于 AIX 的应用程序。

3.2　Power PC 同盟的历史

下列是围绕着 Power PC 同盟的大事件编年史：

■1991 年 10 月，Apple、IBM 和 Motorola 联合声明成立一个前所未有的同盟。随声明一起首次发布了 Power PC 体系结构和 Power Open 环境开发的细节。

■1992 年 1 月，Bull HN Information System，Inc. 宣布他们的服务器和工作站系统采用了 Power PC 体系结构。

■1992 年 4 月，Thomson－CSF 的子公司 CETIA 宣布了和 IBM 共同开发基于 Power PC 体系结构的产品。

■1992 年 5 月，Motorola、IBM 和 Apple 在得克萨斯的奥斯汀落成"大变革设计中心"。

■1992 年 10 月，Motorola、IBM 和 Apple 推出第一台 Power PC 处理器——相当子"第一块可控硅式"的 Power PC 601。

■1992 年 11 月，Harris 公司宣布和 IBM 共同开发基于 Power PC 体系结构的实时工作站。

■1992 年 12 月，Tadpole Technology PLC 宣布和 IBM 共同生产基于 Power PC 体系结构的笔记本计算机。

■1993 年 1 月，Thomson-CSF CETIA 公布 Power PC VME 系统和 Lynx 实时软件。

■1993 年 3 月，Apple、Bull、Harris、Motorola、Tadpole Technology 和 Thomson-CSF 宣布正式成立旨在提供开放系统和兼容性的 Power Open 联盟。

■1993 年 4 月，SunSoft 宣布准备支持基于 Power PC 系统的 Solaris。

■1993 年 4 月，Motorola 公布 Power PC 601 的公众调查报告。

■1993 年 5 月，Motorola 和 IBM 宣布提供 Power PC 601 工具类。

■1993 年 5 月，Ford 宣布其下一代 Power Train Electronic Controller (PTEC)将基于 Power PC 体系结构。

■1993 年 6 月，Kaleida Labs、Scientific－Atlanta 和 Motorola 宣布计划交互式多媒体设备开发采用 Power PC 处理器。

■1993 年 9 月，IBM 公布其 RISC System/6000 系列中的第一批基于 Power PC 的系统，POWERstation 25T、POWERstation 25W、POWERstation 25S 和 POWERstation/POWERserver 250。

■1993 年 10 月，公布 80MHz 601。

■1993 年 10 月，联合宣布 Power PC 603 达到"第一块可控硅"阶段。

■1993 年 10 月，Thomson-CSF 子公司 CETIA 推出基子 Power PC 601 处理器的 VME 单板计算机和工作站。

■1993 年 10 月，Bull 公布其第一个基于 Power PC 的系统——Bull DPX/20 150 型。这个采用 Power PC 601 的系统由下列三种配置构成——150，紧凑模式台式服务器；150S，桌面服务器；155W，单用户工作站。

■1993 年 10 月，Motorola 的 RISC 微处理器部门公布优化 Power PC 603 性能的五个开发软件包——第一套专为 603 设计的工具。包括 C 和 FORTRAN 编辑器、体系仿真器、603 定时仿真器和源代码级调试器。

■1993 年 10 月，Apple 宣布接受其他七个软件开发商的委托——Artwork System、N. V.、Canto Software，Inc.、Fractal Design Corp.、Graphisoft、Great Plains Software、ITEDO Software GmbH 和 Wolfram Research，Inc. 另外还有 Adobe System Inc.、ACIUS Inc.、Aldus Corporation、Claris Corporation、Deneba Software、Frame Technol-

ogy、Insignia Solutions、Microsoft Corporation、Quark、Inc.、Specular International 和 WordPerfect Corporation。同时，他们宣布计划同时把升级产品移植到第一个 Power PC 系统。这份公告包括 Apple Workgroup Server Series 的升级选择。

- 1993 年 11 月，Microsoft 和 Motorola 宣布 Windows NT 的一个端口和 IBM 的 Power Personal System 部联合开发。Windows NT 的端口符合 Power PC 交叉平台的要求，以短端(little-endian)模式操作，而且它是第一个利用 Power PC 双端(bi-endian)特性的系统。
- 1993 年 11 月，IBM 公布 Power PC 交叉平台，这是 IBM Power Personal Systems 一部制订的非专有标准。Power PC 交叉平台(或 PReP)支持的操作系统包括 AIX、Workspace OS、Windows NT、Solaris 和 Taligent，据此为 OS/2、AIX、DOS、Windows、Win32、UNIX 和 Taligent 提供支持，并且支持 Power Open 下的 Macintosh 应用程序。
- 1993 年 11 月，计算机子系统制造商群体宣布支持 Power PC 交叉平台——包括 Austek Microsystems、Cirrus Logic、Matrox、National Semiconductor、NCR、S3 和 SMC Weitek。
- 1993 年 11 月，Motorola 的 Semiconductor Products Sector、Kaleida Labs 和 Scientific-Atlanta 宣布计划采用 Mailbu－多媒体显示加速芯片，由 Kaleida 设计，Motorola 生产。该芯片提供增强的二维、三维图形、动画以及显示高分辨率文本的功能。它将用于 Scientific-Atlanta 的高起点设计，通过宽带电视网络为在家中的网络用户提供交互式多媒体服务。
- 1994 年 2 月，Insignia Solutions 和 Apple 达成协议，在所选的基于 Power PC 的 Macintoshes 配置中包含 SoftWindows。

3.3 小　结

本章简单讨论了 Power PC 同盟在最初三年中的积极行动，说明了广大硬件和软件开发商的急切接受的态度。

因为 Power PC 体系结构适用于广大范围的应用程序，从低耗能的便携机到高端工作站和服务器无一例外，它为各种各样的操作系统提供了合适的平台。下一章研讨基于 Power PC 的系统和 Power PC 体系结构可以支持的操作系统之间的关系。