第 1 章　绪　　论

计算机系统设计的旅程正式开始。不可否认，整个过程将充满艰辛和挑战，但每一阶段你都会有突破自我的巨大成就感。虽然道路崎岖，但会收获颇丰，让我们揭开计算机系统设计的神秘面纱，最后大家会发现其实我们每个人都可以实现一台属于自己的计算机！

现在我们先从了解计算机系统入手。在本章中，首先，对计算机系统的基本内容进行介绍，包括计算机系统的层次结构、计算机硬件系统的组成和计算机软件系统的组成；然后，从 3 个方面，即性能、成本和功耗去评价一个计算机系统的优劣；接着，针对计算机系统的核心部件——处理器，引出有关指令集体系结构的概念，并重点介绍 MIPS 指令集体系结构；最后，对本书所设计的目标处理器 MiniMIPS32 及原型系统 MiniMIPS32_SYS 进行大体介绍。

1.1　计算机系统概述

究竟什么是计算机系统？大多数人可能认为计算机就是桌面电脑，但是实际上，远不止如此。除了大家熟知的个人电脑、超级计算机和服务器等通用计算机外，像手机、平板电脑、数码照相机、数字电视、游戏机、打印机、路由器等设备也都属于计算机的范畴。可见，计算机已经深入到我们生活的方方面面。总而言之，计算机可以很大，大到需要建一栋专门的大楼放置，并建设一个电站专门为其供电（如数据中心，超级计算机神威、天河等）；也可以很小，小到可以放到口袋里，充电几小时就能用一周（如手机）。但无论计算机的规模多么大，种类多么繁多，其本质无非就是一台能够满足人们各种不同计算需求的自动化计算设备而已，因此各类计算机系统具有很多共性。

1.1.1　计算机系统的层次结构

任何计算机系统都可以被看成图 1-1 所示的层次结构。计算机系统划分为**应用程序、操作系统、硬件系统和晶体管** 4 个层次，而应用程序和操作系统又可以统一为**软件系统**。这 4 个层次可通过 3 个抽象界面相互联系，分别是**应用程序编程接口（Application Programming Interface，API）**、**指令集体系结构**（也称为指令系统，Instruction Set Architecture，ISA）和**工艺模型**。采用抽象界面的原因是，计算机系统的多级层次结构使得其设计需要多方面人员的参与，包括应用工程师、系统工程师、硬件设计前端工程师、硬件设计后端工程师等，这样通过抽象层可以屏蔽下层实现细节，提取上层调用接口，使得处于各个层次的工程师更加关注本层的开发，降低设计难度，提高设计效率。

API 位于应用程序和操作系统之间，是高级语言的编程接口，可便于应用程序设计人员更快速、更便捷地进行应用程序开发。常见的 API 包括 C/C++语言、Java 语言、Python 语言、JavaScript 脚本语言接口及 OpenGL 图形编程接口和 Socket 网络编程接口等。使用一种 API 编写的应用程序，经过编译后可以在支持该 API 的不同计算机上运行。可以说，API 的优劣直接决定了应用程序的质量。

ISA 位于操作系统和硬件系统之间。它是对计算机底层硬件细节的抽象，包括指令集、寄存器及系统状态切换、地址空间划分、中断异常处理等运行时环境。可以说，ISA 是系统级程序员所能看到的虚拟计算机。这样，软件设计只需遵循某一 ISA，而不用了解底层硬件的具体细节，从而所开发的软件就可以运行在支持该 ISA 的各种不同的计算机上，保证了软件的兼容性。常见的 ISA 包括 x86、ARM、MIPS 等。

工艺模型位于硬件系统和物理器件之间，是芯片生产厂家提供给芯片设计者的界面。除了表示晶体管和连线等基本参数的模型之外，它还包括该工艺所能提供的各种宏单元及 IP（Intellectual Property）核，如实现 PCIE 接口的物理层（简称 PHY）等。

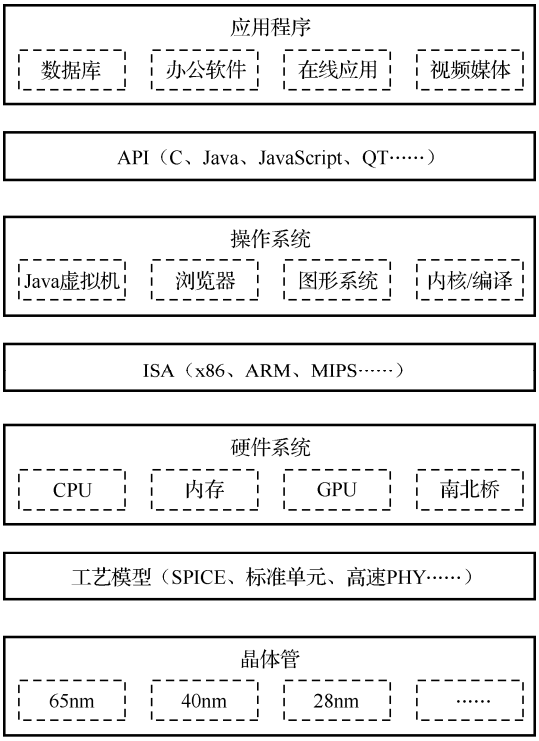

图 1-1　典型计算机系统的层次结构

1.1.2　计算机硬件系统的组成

从 1946 年第一台通用计算机 ENIAC 面世到现在，各种类型的计算机层出不穷，绝大多数的计算机硬件系统仍然沿用了 1945 年由美籍匈牙利数学家冯·诺依曼提出的“**冯·诺依曼计算机结构**”，该结构包含 5 个部分，分别是运算器、控制器、存储器、输入设备和输出设备。其中，运算器和控制器又合称为中央处理器（Central Processing Processor，CPU），简称处理器。冯·诺依曼计算机结构如图 1-2 所示。

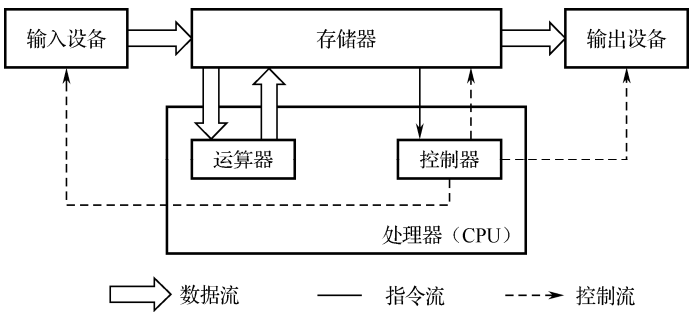

图 1-2　冯·诺依曼计算机结构

冯·诺依曼计算机结构的特点：①计算机处理的数据和指令一律用二进制表示；②存储器按地址访问，且每个存储单元的位数固定，并且按线性一维编址；③采用存储程序方式，指令和数据不加区别混合存储在同一个存储器中；④指令在存储器中按顺序存放，通常，指令按顺序执行，但执行顺序也会根据运算结果或指定的条件而改变；⑤以运算器为中心，输入/输出设备和存储器之间的数据传送都要通过运算器。概括起来，**冯·诺依曼计算机最重要的特点是存储程序和指令驱动执行**。

1．处理器（CPU）

处理器由运算器和控制器两个部件组成，是计算机系统中最核心的运算和控制部件。运算器在计算机中负责计算，而且大部分的运算器只能做简单算术运算（加、减、乘、除，有些低端计算机中的运算器甚至不支持除法）、逻辑运算和移位运算。其他复杂运算可通过这些简单运算组合而成。控制器则根据指令发出控制命令以协调计算机各部件自动工作。控制器具有的功能包括：根据程序计数器中存放的地址从存储器中取出指令，简称为“取指”；对取出的指令进行分析和解释，并根据解释的结果从存储器或输入设备取出指令所需的数据，简称为“译码”；控制器还要向计算机各部件发出控制信号，以协调它们的工作。例如，对于运算指令，则要将数据和操作类型送到运算器；如果是访存指令，则要向存储器发出相应的读写命令；如果要与输入/输出设备进行交互，则除了发送相应的读写命令外，还要负责中断处理。随着集成电路的发展，芯片的集成度不断提高，现代的处理器除了运算器和控制器还集成了很多其他部件，如高速缓存 Cache、主存控制器等，已有别于传统处理器了。

2．存储器

存储器是计算机中的存储和记忆部件，负责保存程序、初始数据和中间结果。在程序执行过程中，要求存储器可以按地址进行随机访问，并且能够以 CPU 处理数据的粒度（4 字节或 8 字节）进行访问。目前，在计算机系统中被广泛采用的存储器按照存储介质可分为以下几类。

1）磁性存储器：如硬盘，其优势是存储密度极高、成本很低、具有非易失性（即断电仍可保存信息），但缺点是访问速度很慢，通常在毫秒级。

2）闪存：如固态硬盘，属于非易失性存储设备，相比磁盘，其访问速度更快，但成本更高、容量较小。

3）动态随机存储器（Dynamic Random Access Memory, DRAM）：如 SDRAM、DDR、DDR2 等，每个存储单元为 1T1C（1 个晶体管 1 个电容）结构，其优势是存储密度较高、访问速度较快（一般在几十纳秒级），其最大缺点是数据具有易失性。

4）静态随机存储器（Static Random Access Memory, SRAM）：每个存储单元为 6T（6 个晶体管）结构，其优势是速度极快（一般在纳秒级），但存储密度较低，且成本高。

在使用计算机时，人们总是希望用最便宜的价格（低成本）买到大容量（存放更多的程序和数据）且访问速度快（处理器可以更快地获取数据）的存储器。显然，上述任何一种存储器都无法满足这个要求。一般来说，速度越快，存储介质的成本越高，而成本越低，存储介质的速度越慢。因此，在现代计算机中通常引入了多层存储体系的概念，以缓解这一矛盾，如图 1-3 所示。从左向右依次为**寄存器堆**（采用 D 触发器实现，位于处理器内部）、**高速缓存 Cache**（采用 SRAM 实现，通常位于处理器内部）、**主存储器**（简称主存，采用 DRAM 实现）和**磁盘**（采用磁介质或闪存实现），在物理距离上离 CPU 越来越远，其速度越来越慢，容量越来越大，价格越来越便宜。

程序的局部性表现为两个方面：时间局部性和空间局部性。时间局部性指如果一个指令或数据被访问，那么其在不久的将来很可能被再次访问；空间局部性指如果一个指令或数据被访问，那么地址与其邻近的指令或数据也可能被访问。利用局部性原理，将近期会用到指令或数据存放在靠近 CPU 的、

访问速度较快的存储层次，把近期用不到的指令或数据存放在远离 CPU、访问速度较慢但存储容量较大的存储层次，从而在整体上达到一个大容量高速单级存储器的效果。

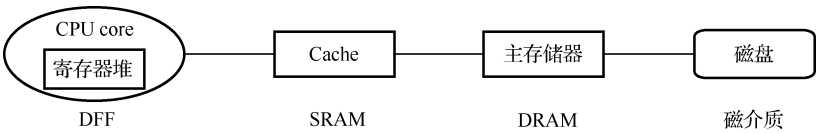

图 1-3　多层存储体系

多层存储体系之所以能够缓解访问速度、存储容量和成本之间的矛盾，得益于程序访问的局部性。

3．输入/输出设备

输入/输出设备（I/O 设备）用于计算机系统与外界之间进行信息交互。输入设备是为计算机提供程序、数据的设备。常用的输入设备包括键盘、鼠标、扫描仪等。输出设备是从计算机中将运算结果传送出来的设备。常用的输出设备包括打印机、显示器、扬声器等。I/O 设备的种类十分繁多，其信号类型和时序也千差万别，通常需要通过标准接口控制器与 CPU 相连，如 GPIO、USB、UART 串口、PS/2 接口、PCIE 接口等。CPU 对 I/O 设备进行访问既可通过专用 I/O 指令（如 x86 系统），也可通过访存指令（如大多数嵌入式系统）。

1.1.3　计算机软件系统的组成

硬件系统为计算机提供了基本的运算能力、控制机制、存储空间及与外界的交互能力，而软件系统构建于硬件系统之上，控制和管理各个硬件模块实现各种具体计算任务。软件系统可以分为**系统软件**和**应用软件**，而系统软件又可进一步细分为**操作系统**和**支撑软件**。

操作系统是一种管理和保护各类软硬件资源、控制程序执行、改善人机界面、合理组织计算机工作流程和为用户使用计算机提供良好运行环境的系统软件，通常由进程管理、线程管理、存储管理、设备管理、文件管理等几大模块组成。另外，现代操作系统通过虚拟化技术为用户提供一个比实际更为强大的计算机，例如，将单处理器虚拟化为多个处理器以适应多任务的需求，而基于段页式的虚拟存储器为程序员提供了一个远大于物理存储空间的编程空间，使其从繁重的存储管理中解放出来。

支撑软件包括用于处理软件语言的语言处理系统（如编译器等）、用于支持数据管理和存取的软件（如数据库）、用于用户与计算机系统之间进行信息交互的软件系统，以及其他事先编写好的各种标准子程序组成的函数库、中间件等。

应用软件是指用户为解决各种问题而利用计算机及其他系统软件编写的软件。

并不是所有的计算机都需要具备这些软件，一些简单的计算机系统，如用于工控的计算机系统，就可能没有操作系统和支撑软件，而直接由应用程序对硬件进行控制。

1.2　计算机系统的评价指标

评价一个计算机优劣的指标有很多，其中性能、成本和功耗是较为重要的 3 个指标。

1．性能

通常说一台计算机速度很快，这个“快”就是指计算机的性能。那如何评价这个“快”呢？对于不同的对象，评价方式可能不同。对于普通计算机用户而言，速度快指的是执行一个程序运行的时间短，即响应速度快。例如，一台 Core i7 的机器和一台 Pentium Pro 的机器相比，对一个大文件进行压

缩，前者完成的时间更短。而对于亚马逊、淘宝、京东这样的电子商务网站，评价其背后的数据中心性能的指标通常是每秒可以完成的交易事务，即吞吐率。

现在，我们仅考虑将计算机的性能定义为“完成一个任务所需的时间”，即计算机系统的执行时间。这个执行时间由多方面因素决定，包括 CPU 时间、磁盘的访问、主存的访问、输入/输出和操作系统开销等。在理想情况下，计算机执行一个任务的时间可以等同于 CPU 时间，即仅考虑处理器的计算时间，而忽略访存时间和等待 I/O 的时间。这样就可以得出 CPU 时间的计算公式。一个程序的 CPU 时间可以描述为

CPU 时间=指令数目×平均每条指令的时钟周期数×时钟周期

CPU 时间越短，说明相应的计算机系统速度越快，性能越高。从 CPU 时间公式可以看出，CPU 的性能由 3 个参数决定。这 3 个参数之间相互依赖，很难只改变一个参数而不影响其他两个参数。时钟周期与工艺技术及计算机组织有关；平均每条指令的时钟周期数（Clock cycles Per Instructions，CPI）则与计算机组织及指令集体系结构相关；程序所含的指令数目则与指令集体系结构及编译技术相关。例如，相比 RISC 指令集，CISC 指令集的指令格式更为复杂，因此 CPI 会较高，但由于 CISC 指令集的功能更强大，因此相比 RISC，其实现同一程序所花费的指令数更少，因此计算机的性能需要综合评价。

除了上述公式外，还可以通过 CPI、每秒执行百万条指令数（MIPS）、每秒执行百万次浮点运算数目（MFLOPS）、每秒执行的事务数（TPS）等多种指标来评价计算机的性能。

2．成本

计算机的成本和芯片的成本直接相关。芯片的成本包括制造成本和一次性成本（如研发成本）的分摊部分。因此，芯片的产量直接影响最终的成本。随着集成电路工艺水平的逐步提升，生产成本可以持续地降低。此外，随着工艺技术的发展，实现相同功能所需要的晶片面积呈指数级降低，从而单个晶片的成本也呈指数级降低。但成本降到一定程度就不再下降，甚至还会缓慢地上升，这是因为厂家为了保持利润不再生产和销售该产品，转而生产和销售升级产品。

3．功耗

目前，对于各种类型的计算机系统，功耗显得越来越重要。例如，手机等移动设备需要电池供电，想延长电池的工作时间，低功耗显得十分重要。再如，位于天津的天河-1A 超级计算机，满负荷运转时，每天总耗电近 10 万千瓦时，费用高达数十万元，再次凸显低功耗的重要性。此外，近年来，由于峰值功耗等约束，芯片上能够同时工作的晶体管数目呈指数趋势减少，芯片上的多个处理器无法按照设计频率满负荷运转，不得已必须关闭一些处理器或降低工作主频，那些不能工作的晶体管就形成了所谓的“暗硅”。这说明当前的计算机设计不能再一味追求性能，而需要着重考虑性能功耗比（Performance per Watt），即每瓦的性能。

芯片功耗是计算机功耗的重要组成部分，主要由 CMOS 晶体管产生，分为动态功耗和静态功耗。动态功耗主要是由电路的翻转（0→1/1→0）产生的。静态功耗主要是指漏电功耗，它是指 MOS 管不能严格关闭而发生漏电产生的功耗。随着晶体管特征尺寸的减小，处理器内的漏电流持续增加。

升级工艺是降低动态功耗的有效方法，因为工艺升级可以降低电容和电压，从而成倍地降低动态功耗。通过选择低功耗工艺可以降低芯片的静态功耗，芯片生产一般会提供高性能工艺和低功耗工艺，低功耗工艺速度稍慢，漏电功耗呈数量级降低。

1.3 处理器概述

处理器是计算机系统中最核心的运算和控制部件。它负责计算机内部主要的计算任务，并根据指令发出控制命令以保证计算机各部件协调、自动地工作。**本书的核心任务就是设计一款基于 MIPS 指令集的处理器**。

1.3.1 指令集体系结构和微体系结构

处理器依靠执行指令来控制计算机的运行，不同的处理器具有不同的指令集，就好比不同国家的人拥有不同的语言一样。这就造成针对某一个处理器编写的程序，不能直接运行于另一个处理器上，必须重新编写，并重新编译后才能运行，大大降低了软件的可复用性和可移植性。

IBM 为了避免上述重新编写软件的问题，使相同的软件可以不经任何修改就能运行在其出品的各种计算机之上，在它的 System/360 计算机中首次引入了**指令集体系结构（ISA）**的概念，这是世界上首个指令集**可兼容**的计算机，具有跨时代的意义。从图 1-1 所示的计算机层次结构中可以看出，ISA 是软件系统和硬件系统的分界面，它将程序员编程时所需要了解的硬件细节从硬件系统中抽象出来，这样程序员只要面向 ISA 进行编程，开发出的软件就能运行在符合该 ISA 的所有计算机之上。而对于处理器设计者，只需要依照 ISA 进行硬件设计，就可保证符合该 ISA 的各种软件能够正常运行。具体而言，从程序员的角度来看，**ISA 主要包括指令集结构、基本数据类型、一组编程规范、寄存器、寻址模式、存储管理、异常和中断处理、运行时环境、运行级别控制等**，涉及软硬件交互的各个方面。

微体系结构（microarchitecture，简称微结构）是 ISA 的一个具体硬件实现。具有相同 ISA 的处理器，可能具有不同微结构，从而表现出不同性能。例如，Intel 酷睿系列处理器遵循 IA-32 或 IA-64 的 ISA，但微结构可能有多种，如 Sandy Bridge、Ivy Bridge、Haswell 等。

1.3.2 CISC 和 RISC

当前的 ISA 可以分为两类：**复杂指令集计算机（Complex Instruction Set Computer, CISC）**和**精简指令集计算机（Reduced Instruction Set Computer, RISC）**。CISC 的指令长度可变，而 RISC 的指令长度固定。

早期的处理器全部是 CISC 架构，倾向于使用功能强大的指令集，一条指令能够完成很多功能，也就是说，其设计的目的就是用最少的机器指令来完成所需的计算任务。采用 CISC 架构的处理器包括 Intel 的 80x86 和 Motorola 的 68K 系列。这种指令集结构的出现与当时的时代特点有关，早期的处理器昂贵且处理速度慢，设计者不得不加入越来越多的复杂指令来提高执行速度，部分复杂指令甚至与高级语言中的操作直接对应。此外，那个时代的主存容量有限，访问速度较慢且价格昂贵，CISC 采用变长指令字，节约了存储空间，而复杂的指令也减少了对主存的访问次数，从而降低了缓慢的访问速度对程序性能的影响。总地来说，**CISC 指令集简化了软件和编译器的设计难度，但也提高了硬件复杂度**。

随着计算机结构的不断改进，指令的功能和数目不断增加，CISC 指令集变得异常庞大，有些处理器的指令数目甚至达到 300 多条。1975 年，IBM Thomas J. Watson 研究中心在研究指令系统的合理性时发现，日趋庞杂的指令系统不但不易实现，而且还可能降低系统性能。1979 年，美国加州大学伯克利分校的 David Patterson 教授研究发现，在 CISC 指令集中，各种指令的使用率相差悬殊，只有 20%的指令被经常使用，而这些指令在程序中占到指令总数的 80%，这就是著名的“20%～80%”准则，可见消耗了大量精力的复杂设计只带来了很少的回报。因此，研究人员开始对指令集和处理器进行重

新设计，只保留常用的简单指令，使得处理器的结构更简单，也更合理，从而提升 CPU 的处理速度，这就是 RISC 指令集。RISC 指令集的特点包括：

- 减少指令集中的指令种类，编译器或程序员通过几条指令完成一个复杂的操作。
- 采用简单的指令格式和寻址方式，指令长度固定，大多数指令能在一个时钟周期内完成。
- 相比 CISC 指令集，拥有更多的通用寄存器，例如，MIPS 拥有 32 个通用寄存器。
- 除了 Load/Store 指令可以访问存储器，其他指令都不能访问存储器，操作数要么来自寄存器，要么来自立即数。
- 硬件结构简单，采用硬布线逻辑，可通过流水线、多发射等技术提高处理器的主频和效率。

虽然 RISC 指令集有很多优点，但其自身也有缺点。例如，RISC 简化了硬件设计，但增加了编译器设计的复杂度。完成同样的任务，基于 RISC 指令集的程序相比 CISC 程序需要更多的指令，使得编译器的优化变得更为重要。

现代处理器对 CISC 和 RISC 进行了融合。例如，Intel 处理器先将复杂指令分解为若干类似于 RISC 指令的微操作，然后采用 RISC 结构实现这些微操作。而一些 RISC 处理器，如 PowerPC，也加入了一些功能强大的专用指令（如向量指令、多媒体指令等）。

1.3.3　指令集体系结构中的“五朵金花”

在当今计算机系统中，处理器的种类成千上万，但 ISA 相对固定。其中 5 种 ISA 较为常见，分别是 x86、ARM、POWER、SPARC 和 MIPS。除了 x86 是 CISC ISA 外，其他都是 RISC ISA。

1．x86

x86 是史上最成功、最赚钱的指令集体系结构。目前，绝大多数个人计算机都使用基于 x86 指令集的处理器。

1978 年，Intel 推出了 8086、8088 处理器，IBM-PC 采用 8088 作为其计算机的核心。1982 年，Intel 推出了 80286，被 IBM PC/AT 所采用。从此以后，x86 成为个人计算机的标准平台。后来，由于 IBM 在选择供应商时为了减少风险，要求至少两家公司同时提供产品，因此 Intel 将 x86 架构开放给了 AMD。为了与 AMD 相区别，Intel 后来采用 IA（Intel Architecture）来代替 x86，即 32 位指令集 IA-32 和 64 位指令集 IA-64。

Intel 处理器之所以成功，得益于其著名的 tick-tock（工艺年-架构年）战略。两年为一周期，第一年提高工艺，晶体管变小，第二年在维持相同工艺的前提下，推出新的处理器微结构。工艺和架构交替改善，既避免了设计风险，也加快了新产品的发布周期，提升了产品的竞争力。

2．ARM

ARM 指令集体系结构属于 ARM Holding，这是一家总部位于英国剑桥的公司，在 1990 年由 Acorn Computers、Apple、VLSI Technology 3 家公司合资组建。早在 1985 年 Acorn 公司就设计了代号为 Acorn RISC 的 32 位处理器，由 VLSI 负责生产，这就是 ARM1。

与 x86 更加关注处理器的性能不同，ARM 更侧重于低功耗、低成本，主要面向嵌入式市场，如手机市场的 90%采用的是 ARM 指令集的处理器。因此，ARM 指令集也是当今销量最大的指令集。

ARM 公司本身不生产芯片，而是向半导体公司提供指令集授权、内核授权，其他公司使用 ARM 内核设计自己的处理器芯片，如三星、高通及我国的飞腾等。目前，ARM 的指令集体系结构包括 ARMv4、ARMv5、ARMv6、ARMv7 及 ARMv8，前 4 种是 32 位指令集，最后一种是 64 位指令集。

3．POWER

最早提出 RISC 思想的 IBM 公司，在 1990 年提出了高性能的 POWER（Performance Optimized With Enhanced RISC）系列处理器，并一直用于 IBM 公司的服务器之上。

1991 年，IBM 与 Apple、Motorola 一起成立 AIM 联盟，进军 PC 市场，并对 POWER 处理器进行改进，形成了 PowerPC。到了 2004 年，Motorola 将其半导体部门分拆，成立了 Freescale，继续对 PowerPC 进行支持。IBM 生产的 POWER 和 PowerPC 处理器侧重于服务器和游戏机领域，如 Sony（索尼）、Microsoft（微软）的游戏机，Freescale 的 PowerPC 更侧重嵌入式市场，如通信、汽车电子等。2004 年 IBM 发起了 Power.org 联盟，发布了统一的指令集体系结构，将 POWER 和 PowerPC 体系结构统一到新架构中。目前，IBM 开放了 POWER8 指令集，并开始向外提供内核授权。

4．SPARC

SPARC（Scalable Processor ARChitecture，可扩展处理器架构）源自美国加州大学伯克利分校 20 世纪 80 年代的研究，并由 Sun 公司在 1985 年首先提出，于 1989 年成为商用架构。Sun 公司将 SPARC 系列处理器用在了高性能工作站和服务器之上，指令集包括 v8、v9 等。

SPARC 指令集架构完全开放，在此基础上出现了一些开放源代码的处理器，如 Sun 公司的 UltraSPAC T1、LEON 等。其中，LEON 是一种 SPARC v8 架构的处理器，至今已发布到了 LEON4，是一种计划用在航天器上的处理器。

5．MIPS

MIPS（Microprocessor without Interlocked Piped Stages，无内部互锁流水级处理器）是最经典的 RISC 处理器，被视为处理器教科书的典范。

MIPS 是由 John L. Hennessy 领导的研究小组在 1981 年开始设计的。其设计理念是使用相对简单的指令，结合优秀的编译器及采用流水线技术执行指令，从而使用更少的晶体管生产出更快的处理器。这一理念在 20 世纪 80 年代取得了巨大的成功，于是 1984 年成立了 MIPS 计算机系统公司，开始对 MIPS 架构进行商业化。随后，基于 MIPS 指令集架构的处理器在工作站、服务器系统中得到了广泛的应用。MIPS 指令集也从 MIPS Ⅰ、MIPS Ⅱ、MIPS Ⅲ、MIPS Ⅳ、MIPS Ⅴ、MIPS32 发展到了 MIPS64。目前，MIPS 公司已经被英国的 Imagination 公司收购。

虽然，在商业上 MIPS 远不如 Intel、ARM 公司成功，不过它的学术地位很高，很多处理器吸收了其中的设计思想，而且 MIPS 架构中的指令专利已经过期，可以自由使用。我国的龙芯系列处理器采用的就是 MIPS 架构。**本书设计的就是基于 MIPS 指令集架构的处理器**。

需要特别注意的是，在计算机发展的几十年间，曾经出现过数量庞大的指令集，但最终被人们所接受的屈指可数。其根本原因并不是那些指令集不够出色，而是缺乏一套完善的生态体系，包括成熟的编译器、操作系统、虚拟机及大量的应用软件。这就好比一个人当然可以发明、使用自己的语言，但是如何与别人交流才是真正的问题。如果无法交流，再优秀的语言也注定会失败。因此，重新定义一套指令集并不难，难的是与之配套的编译器、操作系统、各种应用软件都需要重新编写，也就是重新构建生态体系，这样的工作量和难度都是巨大的。如果没有生态体系，再优秀的指令集体系结构也不会有人使用。这也是为什么本书选择 MIPS 指令集作为目标指令集的原因。

1.3.4 MIPS 指令集体系结构的发展

自 20 世纪 80 年代 MIPS 指令集体系结构出现后，其不断地更新换代，从最初的 MIPS Ⅰ到 MIPS Ⅴ，发展到目前可支持扩展模块的 MPS32 和 MIPS64 系列。每一代都兼容前一代指令集。各代 MIPS

指令集体系结构的概况如下所示。

1．MIPS Ⅰ

MIPS Ⅰ提供加载/存储、计算、跳转、分支、协处理器及其他特殊指令，最初用于 MIPS 早期的 R2000 和 R3000 处理器，其中 R2000 是首款支持 MIPS 指令集的处理器。

2．MIPS Ⅱ

MIPS Ⅱ添加了自陷指令、链接载入指令、条件存储指令、同步指令、可能分支指令、平方根指令，为后来 MIPS32 指令集体系结构奠定了基础。

3．MIPS Ⅲ

MIPS Ⅲ提供了 32 位指令集，同时支持 64 位指令集，最初用于首次添加了浮点处理单元的 MIPS R4000 处理器。

4．MIPS Ⅳ

MIPS Ⅳ增加了条件移动指令、预取指令及一些浮点指令，最初用于 MIPS 处理器 R8000，后来应用于 R5000/R10000。其中，R5000 采用的是经典的 5 级流水线、顺序运行，R10000 采用的是乱序运行。

5．MIPS Ⅴ

添加了能够提高代码生产效率和数据转移效率的指令。可是没有任何一款处理器是基于该指令集体系结构的。MIPS Ⅴ为后来的 MIPS64 指令集体系结构奠定了基础

6．MIPS32 和 MIPS64

MIPS32 和 MPS64 指令集体系结构是定位于高性能、低功耗的 MIPS 指令集，第一版于 1998 年提出，称为 MIPS32/64 Release 1。MIPS32 以 MIPS Ⅱ架构为基础，选择性地增加了 MIPS Ⅲ、MIPS Ⅳ、MIPS Ⅴ中的指令。MIPS64 以 MIPS Ⅴ架构为基础，同时兼容 MIPS32。该 ISA 第一次包括了被称为协处理器 0（CP0）的“CPU 控制”功能。2003 年。MIPS32/64 指令集体系结构的第二版（Release 2）被公布，也称为 MIPS32/64 R2。目前，最新版本是第五版（Release 5），也称为 MIPS32/64 R5。采用 MIPS32 的处理器包括 MIPS32 4K、MIPS32 4KE、MIPS32 24K 等。采用 MIPS64 的处理器包括 MIPS64 5K、MIPS64 20Kc 等。

7．microMIPS32/64

microMIPS32/64 指令集体系结构集成了 16 位和 32 位优化指令的高性能代码压缩技术，保持了 98%的 MIPS32 性能，同时减少了至少 30%的代码体积，从而减少芯片成本，也有助于减少系统功耗。MIPS M14K 处理器内核是 2009 年公布的首款遵循 microMIPS 指令集架构的 MIPS32 兼容内核。

整个 MIPS 指令集体系结构的发展过程如图 1-4 所示。

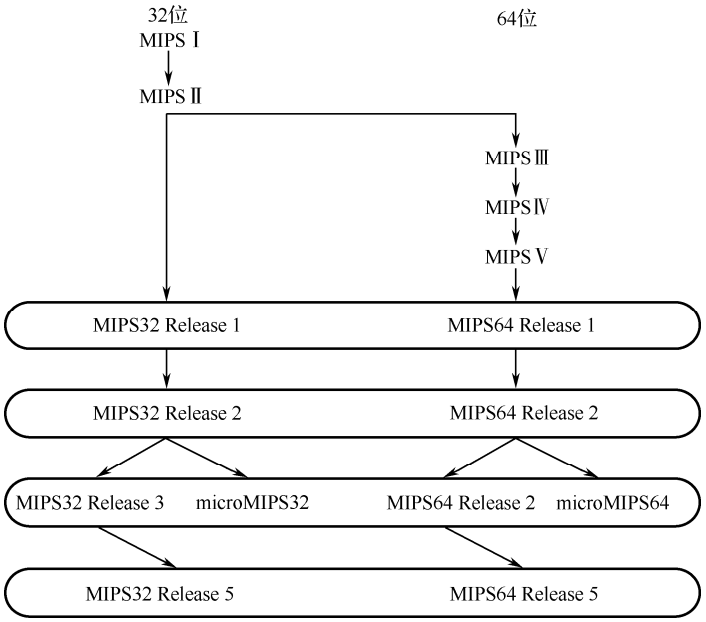

图 1-4　MIPS 指令集体系结构的发展

1.4　本书的主要内容

本书将基于 MIPS32 Release 1 指令集，设计一款采用经典 5 级流水线的 32 位 RISC 处理器——MiniMIPS32，并基于所设计的处理器、存储器和简单外设搭建一个简易原型系统 MiniMIPS32_SYS 来对处理器的功能进行验证和评估。

1.4.1　目标处理器 MiniMIPS32

MiniMIPS32 处理器支持 MIPS32 Release 1 指令集的一个子集，共计 56 条整数指令。基于这 56 条指令的 MiniMIPS32 可运行很多常见应用，以及简单操作系统内核。MiniMIPS32 采用小端模式，数据线和地址线均为 32 位。MiniMIPS32 采用经典 5 级流水线结构，支持定向前推、分支延迟及暂停等常见流水线技术。内部具有 32 个通用寄存器、2 个特殊寄存器及 CP0 协处理器中的部分寄存器，并具有硬件乘法器和除法器。MiniMIPS32 还支持 6 种异常和 6 个外部中断，并且实现了精确异常处理流程。此外，本书所实现的 MiniMIPS32 不支持 MMU 和虚实地址变换，处理器不支持特权级别切换，始终工作在内核模式。

1.4.2　原型系统 MiniMIPS32_SYS

原型系统 MiniMIPS32_SYS 的整体架构如图 1-5 所示，由 1 个 MiniMIPS32 处理器、2 个存储器、若干 I/O 设备、1 个设备接口模块和 1 个 I/O 设备译码模块组成。

MiniMIPS32_SYS 采用哈佛结构，指令存储器和数据存储器分开，其中指令存储器采用 ROM，数据存储器采用 RAM，并支持字节写使能。两个存储器都按字节进行编址。由于存储器都将基于 Xilinx FPGA 内部的块存储器进行构建，因此读写操作都是同步的，其中读指令存储器和读数据存储器都是以 32 位（4 字节）为单位与处理器进行数据交互，而写数据存储器则是根据字节使能位完成的。

MiniMIPS32_SYS 目前仅支持两类 I/O 设备：GPIO 和定时器。其中 GPIO 包括 16 个 LED 灯、4

个七段数码管和 2 个三色 LED，用于显示程序执行结果。定时器为 32 位，可用于对程序的执行时间进行计时。I/O 地址空间采用和存储器统一编址的方式，这样可以通过和访问存储器相同的加载/存储指令去访问 I/O 设备。

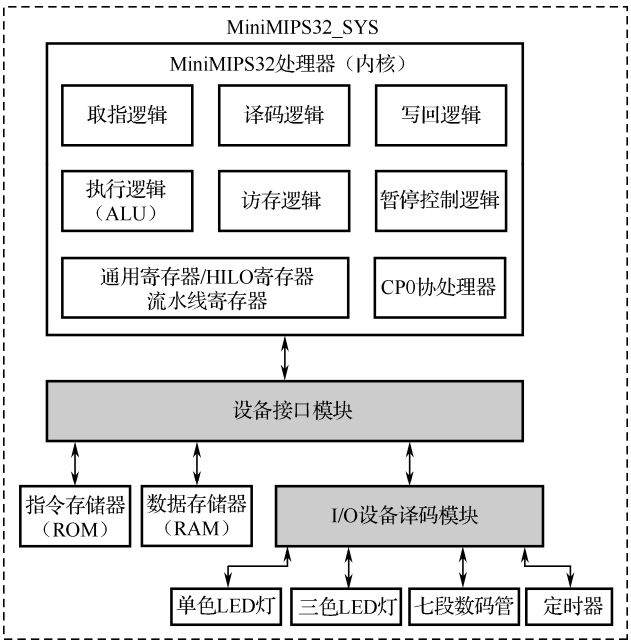

图 1-5　原型系统 MiniMIPS32_SYS 的整体架构

设备接口模块类似于总线，负责处理器与存储器和 I/O 设备进行信息交互。其最主要的功能是对处理器发送来的访问地址进行翻译，并将其转发到相应的设备上。I/O 设备译码模块则是对 I/O 地址进行译码，确定所需要访问的具体 I/O 设备。

在软件工具方面，由于 MIPS32 指令集兼容 MiniMIPS32 指令集，因此，可直接选用现有的 MIPS 交叉编译器编译程序。整个 MiniMIPS32_SYS 硬件系统将在 Xilinx Vivado 集成开发环境中，采用 Verilog HDL 进行描述和仿真，并部署到 Diglient 公司的 Nexys4 DDR FPGA 开发平台上进行实现和验证。为了保证系统功能的正确性，本书采用多种方法进行测试。首先，进行 MiniMIPS32 处理器设计时，将采用手工编写汇编程序，利用 Vivado 进行软件仿真，并结合指令集仿真器 QtSpim 进行协同验证。搭建完成 MiniMIPS32_SYS 系统后，我们将提供具有较高覆盖率的随机功能点测试和常见 C 程序测试。