普通高等教育“十一五”国家级规划教材

计算机课程设计与综合实践规划教材

计算机系统综合课程设计

杨全胜 主编
邢汉承 主审

清华大学出版社

计 算 机 课 程 设 计 与 综 合 实 践 规 划 教 材

计算机系统综合课程设计

杨全胜 主编
邢汉承 主审
杨全胜 王晓蔚 翟玉庆 张志政 吴 强 编著

清华大学出版社
北京

内 容 简 介

本书立足系统，软硬结合，鼓励创新，注重实践，以一个实际的 SoC（片上系统）系统的设计为例，介绍了如何进行软硬件协同设计。具体叙述了一个带有可执行 31 条 MIPS 指令的 CPU 和若干接口部件所组成的 SoC 芯片 MiniSys 的设计过程，以及在该芯片上运行的 BIOS 与汇编器的设计方法。读者通过本教材的学习，不仅学习了实际的简单嵌入式 SoC 系统 MiniSys 从硬件到软件的整个开发过程，还能加深对计算机系统的原理与设计方法的理解。

本书可作为高等院校计算机专业计算机系统综合课程设计的教材，对工程技术人员也具有参考价值。

图书在版编目（CIP）数据

计算机系统综合课程设计/杨全胜主编．—北京：清华大学出版社，2008.10
（计算机课程设计与综合实践规划教材）
ISBN 978-7-302-18558-1

Ⅰ．计…　Ⅱ．杨…　Ⅲ．计算机体系结构－课程设计－高等学校－教材
Ⅳ．TP303

中国版本图书馆 CIP 数据核字（2008）第 140970 号

责任编辑：袁勤勇　李玮琪
封面设计：李建庄
责任印制：王秀菊

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京市昌平环球印刷厂
装 订 者：三河市李旗庄少明装订厂
经　　销：全国新华书店
开　　本：185×260　　**印　张**：18.75　　**字　数**：430 千字
版　　次：2008 年 10 月第 1 版　　**印　次**：2008 年 10 月第 1 次印刷
印　　数：1～3000
定　　价：28.00 元

FOREWORD

前 言

当前，加强实践环节，培养创新人才已经成为全国高校本科生培养的大方向。从计算机专业的学科特点来看，我们更强调课程体系整体优化，立足系统，软硬结合，加强实践，注重创新和发展学生个性。

如何利用实践课程切实做到提高学生综合设计能力和实践动手能力，培养学生创新思维与锐意进取的精神，是我们多年来一直探索的课题。从 2000 年开始，我们就针对计算机专业所有硬件课程进行了整合与实践环节建设。2003 年开始，开设了包含软件与硬件相结合的计算机系统综合课程设计。经过 5 年的建设，该课程得到了学生和同行的认可。在 2007 年高校计算机系主任论坛和 2008 年高校计算机实践课程论坛上，与会同行对该课程表示出浓厚的兴趣，给予了积极的评价。本书就是对我们 5 年来进行该课程建设的一个总结。

多年的教学实践使我们感觉到，计算机专业的学生除了加强计算机科学理论知识的学习，加强实践能力的培养外，还需要加强计算机系统综合分析和综合设计能力的培养。当前是嵌入式系统迅猛发展的时期，作为一类特殊的计算机应用系统，嵌入式系统的开发更需要大量具有系统层面分析与设计能力的技术人员，因此在高校计算机专业高年级本科生中培养这方面的能力是非常有必要的。

立足系统，软硬结合，鼓励创新，注重实践是“计算机综合课程设计”课程的总目标，也是本书编写的主要指导思想。本书以一个实际的 SoC（片上系统）设计为例，介绍了如何进行软硬件协同设计。具体叙述了一个带有可执行 31 条 MIPS 指令的 CPU 和若干接口部件所组成的 SoC 芯片 MiniSys 的设计过程，以及在该芯片上运行的 BIOS 与汇编器的设计方法。读者通过本书的学习，不仅学习了实际的简单嵌入式 SoC 系统 MiniSys 从硬件到软件的整个开发过程，还能加深对计算机系统的原理与设计方法的理解。每章后的思考题能够引导读者在一个更高的层面上考虑设计更复杂的系统。

全书内容安排如下。

第 1 章　概述。本章从原理的角度，分析了设计一个具体系统应该注意的问题，尤其是强调了嵌入式系统设计中，软硬件协同设计的有关问题。

第 2 章　MiniSys CPU 设计。描述了 MiniSys CPU 的体系结构、寄存器组定义和指令集，分析了指令执行的过程，最后给出了大部分部件设计的细节。

第 3 章　MiniSys 接口部件设计。从系统角度描述了总线控制、中断、各类接口部件

的设计方法，并给出了详细的设计。

第4章　BIOS设计。解释了BIOS应该具备的功能以及BIOS开发中应该注意的问题，并给出了几个BIOS模块设计的例子。本章还讨论了有关堆栈设计和实现的方法。

第5章　汇编器的设计。详细讨论了MiniSys系统汇编语言规范的设计方法，从编译原理的角度介绍了MiniSys汇编器的设计方法。

第6章　综合测试与验证。详细描述了如何用仿真工具对设计的系统进行验证，如何编写测试程序以及如何测试软件功能。

第7章　高级设计讨论。为能力较强的读者提供了更高层次的设计方案，包括在硬件上如何实现流水线处理器，如何增加浮点运算单元和乘除运算器以及软件层面上如何设计Mini C编译器。

第8章　Verilog HDL语言基础。本章相对独立，主要是针对没有学过Verilog HDL语言的读者提供一个快速入门的简单教程，但还是希望读者至少有C语言的基础。

通过上述8章的介绍和章后的思考题，读者最终能够完成一个简单的SoC系统的软硬件设计，并能在Altera的Quartus Ⅱ平台上使用Cyclone芯片实现。本书中的Verilog HDL代码在Quartus Ⅱ 7.0下实现，汇编器代码也在Turbo C 2.0以及VC++ 6.0中调试通过。读者也可以使用C++、C#、Java等语言实现。

本书在编写过程中充分考虑了不同层次学生的具体情况，因此在使用本书作为实践教材的时候，建议教师要注重因材施教，合理选择。一般，对学生的最基本要求是将教材前6章中提供的源码补充完整，并完成思考题中不带*号的题目。对于基础好，能力较强的学生，要求他们选择完成思考题中带*号的问题，*号越多，题目难度越大。第7章的内容也是为学有余力的同学准备的，鼓励他们完成更高的设计。

我们建议使用该教材开设“计算机系统综合设计”类的实践课程课时数为68学时，其中20学时左右介绍Verilog HDL语言和相关的理论知识，侧重介绍工程设计理念和软硬协同设计方面的理论知识。46学时实验，建议每5人一组，分别完成CPU、接口部件、BIOS、汇编器(或编译器)的设计。每组2个学时的时间进行验收，主要验收测试程序和仿真波形。另外，本书作为实践课程教材，并未对理论部分做详细阐述，所以希望读者最好具备“数字逻辑电路”、“计算机组成原理”、“微机原理与接口技术”、“编译原理”、“C语言程序设计”等课程的知识。

全书由杨全胜、王晓蔚、翟玉庆、张志政和吴强老师编写，杨全胜老师负责全书内容的修改和最终定稿。东南大学计算机科学与工程学院博士生导师邢汉承教授与朱怡健副教授审核了全书，并提出了宝贵意见。

本书在编写过程中还得到了东南大学计算机科学与工程学院领导和系统结构教研组王健、任国林、徐造林等老师的大力支持。另外，东南大学计算机科学与工程学院2004届到2008届本科生在本课程建设期间，用自己的实践不仅验证了本课程内容的正确性，还帮助我们完善了课程内容。其中，张玉健、张冠群、万乾坤、崔涛、李正兴、苏凯、谢川、邱栋等同学的实践成果丰富了我们教材中的实例。邹峰、崔涛、闫升、李正兴同学制作了本课程的网站和网络课件，参与研发了本课程实验设备。在此，我们对所有帮助我们的同事和同学表示衷心的感谢。另外，我们还要特别感谢清华大学出版社的编辑们在本教材申报

国家十一五规划教材期间，以及书稿编辑期间给予我们的极大帮助。

本书网站地址：http://cse.seu.edu.cn/people/yangqs/declareWB1/Index.htm，作者邮件地址：yangqs@seu.edu.cn。

由于实践类教材的编写对于我们来说是第一次，再加上我们水平有限，难免会有不足之处，殷切希望得到广大同仁和读者的批评指正。

编　者

于南京江宁九龙湖畔

2008年8月

CONTENTS

目 录

CHAPTER 1

第1章 概 述

1.1 计算机系统概述

1946年,第一台电子计算机诞生。经过60多年的发展,历经了电子管计算机、晶体管计算机、集成电路计算机和大规模及超大规模集成电路计算机四代。计算机性能不断提高,性能价格比不断上升,应用领域也越来越广。当前的计算机既向巨型化发展,也同时向微型化发展。即出现了由131 072个CPU组成的目前世界上排名第一的eServer Blue Gene Solution计算机;也出现了专用领域的嵌入式系统,甚至是在一个芯片上实现的计算机系统SoC。但不论是巨型机还是SoC,它们在基本原理上都有共同之处。本节主要介绍计算机系统的基本知识和一些将要用到的概念。

1.1.1 计算机系统的组成

1945年6月30日,普林斯顿大学数学教授冯·诺依曼(Von Neumann)发表了EDVAC(Electronic Discrete Variable Computer,离散变量自动电子计算机)方案,确立了现代计算机的基本结构,提出计算机应具有五个基本组成部分(运算器、控制器、存储器、输入设备和输出设备),描述了这五大部分的功能和相互关系,并提出"采用二进制"和"存储程序"这两个重要的基本思想。迄今为止,大部分计算机仍基本上遵循冯·诺依曼结构。当我们要计算机完成某项工作的时候,例如一个财务管理工作或者一个文字处理工作等,必须先设计解决问题的算法,然后根据算法,编写有关的程序,准备所需要的数据。"存储程序"就是把这些事先编写好的程序和数据存储到存储器中保留起来。系统会根据给出的程序中第一条指令的存储地址,取出第一条指令,然后控制器就可以依据存储程序中的指令顺序周而复始地取指令、分析指令和执行指令,直到完成全部的指令操作。

计算机系统由硬件系统和软件系统组成,如图1.1所示。

1. 计算机系统的硬件系统

从图1.1可以看出,计算机系统的硬件系统包括了作为计算机核心的中央处理单元(CPU)、存放数据和程序的存储器以及能够为程序提供各种数据的输入设备和将运算结果显示出来的输出设备。

(1) 中央处理单元(CPU)

CPU是计算机的运算和指挥控制中心。它包含了运算器、控制器和寄存器组。运算

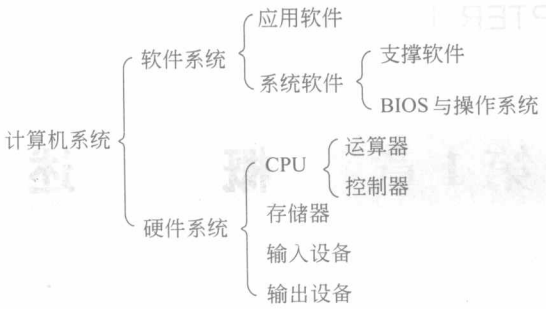

图 1.1　计算机系统的组成

器负责算术和逻辑运算，控制器负责协调 CPU 各部件的工作和根据指令发出有关的控制信号。在现代微处理器中，控制器一般由指令预取单元、指令译码单元、控制单元和结果回写单元组成。

- 指令预取单元根据指令指针寄存器中存放的地址从存储器中取出指令，放入指令寄存器中。
- 指令译码单元分析和解释指令寄存器中的指令，并根据需要从存储器或者 I/O 端口中取出指令所需的数据。如果需要，译码单元会将数据送到运算逻辑单元进行运算。
- 控制单元按照指令的要求，对计算机各部件发出相应的控制信号，协调它们的工作。
- 结果回写单元负责将运算的结果写回存储器或者送到 I/O 端口输出。

寄存器组包含多个寄存器，它们用来存放 CPU 经常使用或正在使用的数据及暂存计算结果。这些寄存器包括通用寄存器、控制与状态寄存器、指令指针寄存器等。

(2) 存储器

存储器是计算机的存储和记忆装置，用来存放数据和程序，它包括内存和外存(如硬盘)，通常它以 8 个二进制位为一个单元，每个单元规定一个唯一的物理地址。对于内存，CPU 可直接对它进行访问，因为其读写速度快于外存，所以主要存放当前正在使用的数据和程序。但由于当前 CPU 主频提高很快，内存读写速度相对来说不能适应 CPU 的高速读写要求，因此，在当前的计算机系统的内存子系统设计中，往往会在 CPU 和内存之间加上快速存储的高速缓存(Cache)作为过渡。

在计算机中，通常 8 个二进制位称为一个字节(Byte)，1024(2^{10})字节记做 1KB，2^{20} 字节记做 1MB，2^{30} 字节记做 1GB，2^{40} 字节记做 1TB。

(3) 输入输出设备

输入/输出(I/O)设备是那些为计算机提供数据或信息的输入设备(如扫描仪、键盘、鼠标等)和那些接收从计算机中输出的信息或数据的输出设备(如打印机、显示器等)。但无论哪种输入/输出设备，都不能直接和 CPU 相连，因为设备的类型繁多，需要的信号类型也各有差异，时序上也千差万别，因此，必须通过各种 I/O 接口电路进行相应的转换以后，再和 CPU 相连。I/O 接口电路是计算机与 I/O 设备之间的桥梁，是数据进出计算机

的通道，也是微机与I/O设备工作的协调者。

(4) 计算机总线

上述谈到的各种部件并不是独立存在的，它们之间需要密切配合，互通信息，而将这些部件联系到一起的就是计算机总线。计算机总线是在部件和部件之间或设备与设备之间的一组进行互连和传输信息的信号线，其传输的信息包括指令、数据和地址。对于连接到总线上的多个设备而言，任何一个设备发出的信号可以被连接到总线上的所有其他设备接收。但在同一时间段内，连接到同一条总线上的多个设备中只能有一个设备主动进行信号的传输，其他设备只能处于被动接收的状态。

当前计算机系统中，通常采用单总线或双总线结构。单总线中，CPU、内存和I/O设备均挂在一条总线上，这种结构简单，但内存和I/O设备会因总线冲突而降低传输效率。双总线结构中，CPU与内存之间以及CPU与I/O之间是通过两条总线相连，这将提高计算机系统的运行速度。

每条总线都是由多个传输线组成，这些传输线又被分为地址总线、数据总线和控制总线。地址总线传送CPU发出的访问内存或外部设备接口的地址信息。对于内存的访问，地址总线的宽度(总线条数)决定了系统能访问的最大内存容量。数据总线既可以从CPU传送数据信息到外设和内存，也可以从内存和外设向CPU传送数据。数据总线的宽度决定了一次可以传送的二进制数据的位数。控制总线传送控制信息、时序信息和状态信息。这些信息控制数据总线、地址总线的使用。

2. 计算机系统的软件系统

硬件系统只是计算机系统的物理基础，必须配备各种软件才能做人们想要它们做的事情。计算机系统的软件系统包括为了运行、管理和维护计算机而编制的各种程序的总和，它分为系统软件和应用软件。系统软件包括基本输入输出系统(BIOS)、操作系统和支撑软件。其中BIOS在开机的时候完成硬件自检、启动操作系统的功能，并负责向上层软件提供控制硬件的简单接口。操作系统负责管理和保护计算机系统的各种资源，它通过进程管理、作业管理、内存管理、设备管理、文件管理等几大模块不仅有效地管理、利用和保护了系统资源，还向用户或程序员提供了便捷的操作界面和编程接口。另外，现代操作系统充分利用处理器的资源，通过各项虚拟技术为用户提供了一个比实际裸机更为强大的虚拟计算机，例如多任务系统中，单处理器微机被虚拟成多个处理器，而请求页式、请求段式存储管理，使得虚拟存储的容量也远远大于实际内部存储器的容量。

计算机系统可以采用二进制机器指令码直接编程，这样写出的程序执行效率较高，而且代码量小。但是这种方法不容易记住指令码，也很难在今后进行代码维护。为了方便程序员编程，逐渐形成了带有指令助记符的汇编语言和各种更接近自然语言的高级语言，如BASIC、C、C++、Delphi等。这些语言并不能被机器自动识别，必须有专门的软件将其翻译成机器能懂的机器码，这就需要编译系统。除此以外，还有帮助编程人员的调试软件与文字编辑软件、管理大量数据的数据库管理系统软件以及为了扩大计算机的功能而事先编好的各种标准子程序所组成的程序库、中间件等。所有这些就组成了系统软件中的支撑软件。

应用软件是指用户为解决各种问题而利用计算机及其系统软件编写的软件。

不过不是所有的计算机系统都具有完整的软件体系，在一些简单的系统（如使用单片机作为处理器的工业系统中）就可能没有基本输入输出系统和操作系统，而直接由应用程序对硬件进行控制。

3. 计算机系统的软硬件工作过程

从程序员的角度看，计算机系统是通过程序（软件）的执行来完成各种任务，软件是一条条指令的集合。而从系统结构人员的角度，计算机系统完成任务最终又是通过各种电子信号（数据、控制等）的配合来实现。

程序可以是用汇编语言写的，而更多的是用高级语言编写的。一个用高级语言描述的程序需要经过编译、连接、执行，才能最终变成一个个电子的数据信号、地址信号或控制信号，完成所需的工作。图 1.2 给出了一个高级语言或汇编语言程序如何转换电子信号的过程。

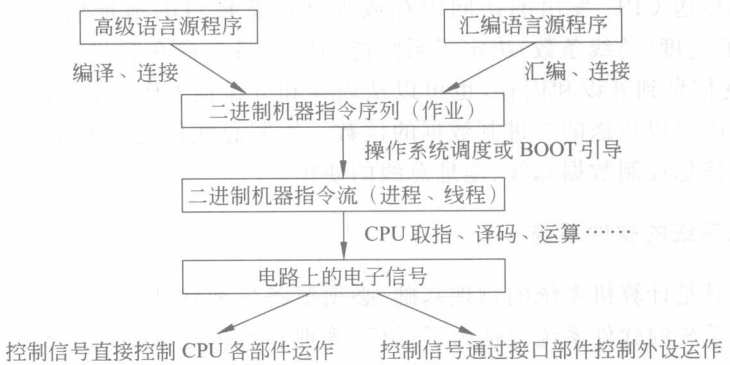

图 1.2　从软件到电子信号的大致转换过程

高级语言或汇编语言编写的程序是机器无法识别的，因此，必须通过编译系统将其翻译成机器能识别的二进制形式的机器指令。编译系统通过词法分析和语法分析、中间代码生成等过程，最后根据不同的指令系统生成相关的二进制机器指令序列。在生成机器指令序列的时候，好的编译系统除了常规优化外，可以根据硬件结构的特点对指令序列进行特殊的优化，以便该程序能够充分利用硬件资源。例如，在超标量结构中，尽量将不相关的指令连续存放，以便几条指令能同时进入几条流水中。有时候，编译系统甚至需要通过库程序来弥补硬件的不足。如早期 PC 的 CPU 没有浮点运算单元，编译器就需要提供一个软件的浮点运算仿真库来解决这个问题。

二进制的机器指令序列仅仅是存放在外存中的可执行文件，并不能产生任何的信号，必须要调入到内存中实际运行起来。通常可执行文件是由操作系统的作业调度程序调度到内存中成为进程，再由进程调度程序调度进 CPU 运行环境中才正式运行，并产生各类信号。而操作系统本身的核心部分则是由 BIOS 的 BOOT 引导程序引导进内存执行。

进入到内存的二进制机器指令流经过 CPU 的取指单元取出、通过译码单元对指令进行分析和解释，为下一步的运算作数据准备，而控制单元根据指令的要求，发出各种控

制信号，这些控制信号会协调各部件工作。运算单元计算出来的结果由回写单元写回存储器或I/O端口。

系统的BIOS程序是直接将编译过后生成的二进制机器指令序列固化在ROM、E^2PROM或FLASH中，系统启动时直接被CPU取指执行。在有些结构的系统中，没有BIOS和操作系统，其应用程序也是采用这种方法执行。

由上面的步骤可以看到，无论是高级语言程序还是汇编程序，最终都要转换成机器能识别的机器指令，这些机器指令再在CPU的工作下转换成各类电子信号。

1.1.2 计算机存储结构类型

在计算机的存储结构上，存在着冯·诺依曼结构和哈佛结构两种。

冯·诺依曼结构的计算机在存储结构上采用了一种将指令和数据以二进制形式存储在同一个存储器的结构。指令存储地址和数据存储地址指向同一个存储器的不同物理位置，因此指令和数据的宽度相同。目前使用冯·诺依曼存储结构的中央处理器和微控制器有很多，例如Intel公司的x86系列、ARM公司的ARM7、MIPS公司的MIPS处理器等。

这种存储结构有一个缺陷，那就是指令和数据存放在一个存储体内，在流水线处理器中，当一条指令取指，而上一条指令需要取数据的时候，就会产生访存的冲突，从而降低系统性能。

另外，数据和指令都以二进制形式不加区分的存储在一个存储空间中，这样数据取出后，也可以当作指令执行。换句话说，一个可执行的程序可以被当作数据存放。这给一些病毒程序提供了一个漏洞，曾经就有一种蠕虫病毒将自己程序的一部分作为文本文件存放在硬盘中，然后通过指针调用该文本执行，因此躲过了大部分杀毒软件的查杀。

另一种存储结构是将指令和数据分别存储在指令存储器和数据存储器中，中央处理器首先到指令存储器中读取程序指令内容，解码后得到数据地址，再到相应的数据存储器中读取数据，并进行下一步的操作(通常是执行)。这种存储结构称为哈佛结构。指令存储器和数据存储器分开，可以使指令和数据有不同的数据宽度。

由于指令和数据分开组织和存储，这样在读取数据的时候同时取下一条指令，不会产生访存冲突，因此哈佛结构的微处理器通常具有较高的执行效率。摩托罗拉公司的MC68系列、Zilog公司的Z8系列、ATMEL公司的AVR系列、ARM公司的ARM9、ARM10和ARM11系列都采用了哈佛结构，MCS-51单片机广义上也属于哈佛结构。

哈佛结构的缺点是两套存储体机制下，如果采用处理器外扩展存储体方式时，处理器需要的引脚数会比冯·诺依曼结构多很多。另外，如果程序中的常量定义在指令存储器中，则对它的读取要比读取数据存储器中的数据复杂。

实际上，现在的处理器设计已经开始出现融合两种结构的设计方案，例如Intel的Pentium系列和酷睿系列，总的架构是冯·诺依曼结构，但在L1 Cache的设计上，就借鉴了哈佛结构，单独设计了L1指令Cache和L1数据Cache。

1.1.3 计算机指令集类型

一个计算机系统的指令集是该计算机系统能够执行的所有指令的总和。不同的计算机系统会有不同的指令集，而不同的指令集往往会影响到整个系统的体系结构设计。尽管目前计算机的指令集很多，但大致上可以分成两类，复杂指令集和精简指令集。根据这两类指令集，计算机的体系结构也被相应分成两种。

第一种是复杂指令集计算机(complex instruction set computer，CISC)，早期的计算机都是采用的 CISC 结构，这种结构是希望指令功能非常强大，因此指令格式比较复杂，通常采用不等长指令设计，指令的寻址方式丰富，绝大多数指令的执行需要多个时钟周期。x86 就属于 CISC 型处理器。CISC 技术有其一定的优点，其指令系统与高级语言的语义相近，因此降低了编译程序的复杂程度，而且使得编译后的程序较小，节省存储空间。

但是，CISC 也有其固有的缺点，首先随着计算机结构的不断改进，指令的功能和指令条数不断增加，指令系统变得异常庞大，有些系统的指令数目甚至多达 300 多条。而且经过研究发现，只有 20%的指令是最常用的，在程序中占了 80%。其次，复杂的指令格式和众多的寻址方式使得硬布线逻辑电路设计更为复杂，尽管采用微程序设计技术可以降低电路复杂性，但也在一定程度下影响了指令的执行速度。第三，复杂不规整的指令会降低流水线的性能。第四，随着指令条数的增加，完成同一任务的指令组合变多，编译系统在最后优化的时候分析就变得更加困难。

第二种是精简指令集计算机(reduced instruction set computer，RISC)。通过简化指令，使得计算机的结构变得简单、合理，从而提高 CPU 的执行速度。RISC 的主要特点如下。

(1) 优化指令系统，只选用使用频率高的指令，减少指令条数。

(2) 采用简单的指令格式和寻址方式，指令的长度固定，大多数指令能在一个时钟周期内完成。

(3) 除了 Load/Store 指令能访问存储器外，其他任何指令的操作数或为立即数或存放在寄存器中，因此，进行的是寄存器与寄存器之间的操作。通常 RISC 处理器设计了大量的寄存器临时存放数据。

(4) 由于计算机结构简单，所以主要采用硬布线逻辑，较少使用或者不用微程序控制。

RISC 体系结构更加适合流水线、超标量的 CPU 设计。但它也有其自身的缺点，那就是简化了硬件设计，却使得编译程序变的复杂，因为要完成同一个任务，RISC 程序通常要比 CISC 程序需要更多的指令，这就使得编译器的优化更为重要。

目前运行的很多处理器都综合采用了 CISC 和 RISC 体系结构，例如传统的 CISC 处理器系列中的 Pentium、Pentium 4、Core 2 Duo 也采用了 RISC 设计思想，而 RISC 处理器中的 Power 4 处理器则采用了一些复杂指令。

1.1.4 单周期、多周期和流水线处理器

计算机中的最核心部分是处理器，目前的处理器至少要完成取指、译码和执行等工

作。单周期的CPU会在一个时钟周期内完成所有的工作，即从指令取出到得到结果，全部在一个时钟之内完成。单周期CPU的设计比较适合不很复杂的电路，因为当电路复杂后，整个处理电路的最长路径延迟会严重阻碍处理器的工作频率的提高。

多周期CPU的设计是将整个CPU的执行过程分成几个阶段，每个阶段用一个时钟去完成。如上述所说的分成取指、译码、执行等阶段，如果每个阶段都用一个时钟周期完成的话，则整个指令的取指到得到结果的过程需要3个时钟才能完成。尽管多周期的设计比单周期的设计在完成一条指令的执行上要花费更多的时钟周期，但由于每个阶段的电路复杂性降低，各阶段的电路最长路径延迟必定会比整个任务执行时间短，而多周期的处理器的时钟周期是由具有最大阶段电路延迟决定的，因此，时钟周期会大大缩短，进而时钟频率会大步提升。例如，假设原来单周期的CPU完成一次完整操作需要10ms，则处理器最高频率为100Hz，如果将其执行分成5个阶段，每个阶段完成的时间分别为1ms、2ms、2ms、4ms、1ms，则多周期下的最高频率可以接近1000/4＝250Hz。多周期CPU的设计不仅能提高CPU的工作频率，还为组成指令流水线提供了基础。

在多周期CPU设计的基础上，利用各阶段电路间可并行执行的特点，让各个阶段的执行在时间上重叠起来，这种技术就是流水线技术。假设一条指令的执行分为取指(F)、译码(D)和执行(E)三个阶段，每个阶段都花费一个时钟周期。如果采用顺序执行，则三条指令的执行过程如图1.3(a)所示，但如果采用流水线技术，同样三条指令执行过程如图1.3(b)所示，从图中可以看到采用流水线技术后，三条指令执行总周期数从9个减少到了5个。这就在一定程度上提高了单位时间的执行指令数。

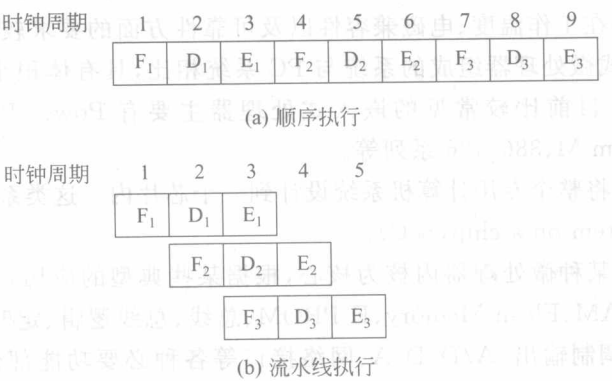

图1.3　指令流水线的基本思想

CPU的性能指标可以用式(1-1)计算。

$$\text{CPU性能}=\text{CPU主频}\times\text{IPC}/\text{指令数} \tag{1-1}$$

式中，IPC是每时钟执行的指令条数。作为分母的指令数由指令系统结构(ISA)、编译器性能和操作系统性能决定，ISA中每条指令完成的工作量越多，则完成特定功能的指令数越少；编译器对程序优化的越好，指令数也会越少；而应用程序对操作系统中的功能调用会增加指令数。显然，从单周期发展到多周期是为了提高CPU的主频，而流水线处理器是为了在多周期的情况下提高IPC，为了进一步提高IPC，现代微处理器中甚至采用多流