<<计算机组成与设计>>

图书基本信息

书名：<<计算机组成与设计>>

13位ISBN编号：9787111353058

10位ISBN编号：7111353056

出版时间：2012-1

出版时间：机械工业出版社

作者：（美）David A.Patterson,（美）John L.Hennessy

译者：康继昌,樊晓桠,安建峰

<<计算机组成与设计>>

内容概要

本书是计算机组成的经典教材。
全书着眼于当前计算机设计中最基本的概念，展示了软硬件间的关系，并全面介绍当代计算机系统发展的主流技术和最新成就。

同以往版本一样，本书采用MIPS处理器作为展示计算机硬件技术、汇编语言、计算机算术、流水线、存储器层次结构以及I/O等基本功能的核心。
书中强调了计算机从串行到并行的最新革新，在每章中都纳入了并行硬件和软件的主题，以软硬件协同设计发挥多核性能为最终目标。

本书适合作为高等院校相关专业的本科生和研究生教材，对广大技术人员也有很高的参考价值。

作者简介

作者：(美国)帕特森 (David A.Patterson) (美国)亨尼斯 (John L.Hennessy) 译者：康继昌 樊晓桠 安建峰等David A.Patterson，加州大学伯克利分校计算机科学系教授，美国国家工程研究院院士，IEEE和ACM会士，曾因成功的启发式教育方法被IEEE授予James H.Muiiigan.Jr教育奖章。
他因为对RISC技术的贡献而荣获1995年IEEE技术成就奖，而在RAID技术方面的成就为他赢得了1999年IEEE。
Reynold Johnson信息存储奖。
2000年他和John L.Hennessy分享了Johnvon Neumann奖。
John L.Hennessy斯坦福大学接长。
IEEE和ACM会士，美国国家工程研究院院士及美国科学艺术研究院院士。
Hennessy教授因为在RISC技术方面做出了突出贡献而荣获2001年的Eckert-Mauchly奖章，他也是2001年Seymour Cray计算机工程奖得主，并且和David A.Pafferson分享了2000年Johnvon Neumann奖。

<<计算机组成与设计>>

书籍目录

出版者的话
译者序
前言
第1章　计算机概要与技术
1.1　引言
1.1.1　计算应用的分类及其特性
1.1.2　你能从本书学到什么
1.2　程序概念入门
1.3　硬件概念入门
1.3.1　剖析鼠标
1.3.2　显示器
1.3.3　打开机箱
1.3.4　数据安全
1.3.5　与其他计算机通信
1.3.6　处理器和存储器制造技术
1.4　性能
1.4.1　性能的定义
1.4.2　性能的测量
1.4.3　CPU性能及其因素
1.4.4　指令的性能
1.4.5　经典的CPU性能公式
1.5　功耗墙
1.6　沧海巨变：从单处理器向多处理器转变
1.7　实例：制造以及AMD Opteron X4基准
1.7.1　SPEC CPU基准测试程序
1.7.2　SPEC功耗基准测试程序
1.8　谬误与陷阱
1.9　本章小结
1.10　拓展阅读
1.11　练习题
第2章　指令：计算机的语言
2.1　引言
2.2　计算机硬件的操作
2.3　计算机硬件的操作数
2.3.1　存储器操作数
2.3.2　常数或立即数操作数
2.4　有符号和无符号数
2.5　计算机中指令的表示
2.6　逻辑操作
2.7　决策指令
2.7.1　循环
2.7.2　case/switch语句
2.8　计算机硬件对过程的支持
2.8.1　使用更多的寄存器
2.8.2　嵌套过程

2.8.3 在栈中为新数据分配空间
2.8.4 在堆中为新数据分配空间
2.9 人机交互
2.10 MIPS中32位立即数和地址的寻址
2.10.1 32位立即数
2.10.2 分支和跳转中的寻址
2.10.3 MIPS寻址模式总结
2.10.4 机器语言解码
2.11 并行与指令：同步
2.12 翻译并执行程序
2.12.1 编译器
2.12.2 汇编器
2.12.3 链接器
2.12.4 加载器
2.12.5 动态链接库
2.12.6 启动一个Java程序
2.13 以一个C排序程序为例
2.13.1 swap过程
2.13.2 sort过程
2.14 数组与指针
2.14.1 用数组实现clear
2.14.2 用指针实现clear
2.14.3 比较两个版本的clear
2.15 高级内容：编译C语言和解释Java语言
2.16 实例：ARM指令集
2.16.1 寻址模式
2.16.2 比较和条件分支
2.16.3 ARM的特色
2.17 实例：x86指令集
2.17.1 Intel x86的改进
2.17.2 x86寄存器和数据寻址模式
2.17.3 x86整数操作
2.17.4 x86指令编码
2.17.5 x86总结
2.18 谬误与陷阱
2.19 本章小结
2.20 拓展阅读
2.21 练习题
第3章 计算机的算术运算
3.1 引言
3.2 加法和减法
3.2.1 多媒体算术运算
3.2.2 小结
3.3 乘法
3.3.1 顺序的乘法算法和硬件
3.3.2 有符号乘法
3.3.3 更快速的乘法

3.3.4 MIPS中的乘法
3.3.5 小结
3.4 除法
3.4.1 除法算法及其硬件结构
3.4.2 有符号除法
3.4.3 更快速的除法
3.4.4 MIPS中的除法
3.4.5 小结
3.5 浮点运算
3.5.1 浮点表示
3.5.2 浮点加法
3.5.3 浮点乘法
3.5.4 MIPS中的浮点指令
3.5.5 算术精确性
3.5.6 小结
3.6 并行性和计算机算术：结合律
3.7 实例：x86的浮点
3.7.1 x86浮点体系结构
3.7.2 Intel SIMD流扩展2（SSE2）浮点体系结构
3.8 谬误与陷阱
3.9 本章小结
3.10 拓展阅读
3.11 练习题
第4章 处理器
4.1 引言
4.1.1 一个基本的MIPS实现
4.1.2 实现方式概述
4.2 逻辑设计惯例
4.3 建立数据通路
4.4 一个简单的实现机制
4.4.1 ALU控制
4.4.2 主控制单元的设计
4.4.3 数据通路的操作
4.4.4 控制的结束
4.4.5 为什么不使用单周期实现方式
4.5 流水线概述
4.5.1 面向流水线的指令集设计
4.5.2 流水线冒险
4.5.3 对流水线概述的小结
4.6 流水线数据通路及其控制
4.6.1 图形化表示的流水线
4.6.2 流水线控制
4.7 数据冒险：转发与阻塞
4.8 控制冒险
4.8.1 假定分支不发生
4.8.2 缩短分支的延迟
4.8.3 动态分支预测

4.8.4 流水线小结
4.9 异常
4.9.1 异常在MIPS体系结构中的处理
4.9.2 在流水线实现中的异常
4.10 并行和高级指令级并行
4.10.1 推测的概念
4.10.2 静态多发射处理器
4.10.3 动态多发射处理器
4.11 实例：AMD Opteron X4（Barcelona）流水线
4.12 高级主题：通过硬件设计语言描述和建模流水线来介绍数字设计以及更多流水线示例
4.13 谬误与陷阱
4.14 本章小结
4.15 拓展阅读
4.16 练习题
第5章 大容量和高速度：开发存储器层次结构
5.1 引言
5.2 cache的基本原理
5.2.1 cache访问
5.2.2 cache缺失处理
5.2.3 写操作处理
5.2.4 一个cache的例子:内置FastMATH处理器
5.2.5 设计支持cache的存储系统
5.2.6 小结
5.3 cache性能的评估和改进
5.3.1 通过更灵活地放置块来减少cache缺失
5.3.2 在cache中查找一个块
5.3.3 替换块的选择
5.3.4 使用多级cache结构减少缺失代价
5.3.5 小结
5.4 虚拟存储器
5.4.1 页的存放和查找
5.4.2 缺页
5.4.3 关于写
5.4.4 加快地址转换：TLB
5.4.5 集成虚拟存储器、TLB和cache
5.4.6 虚拟存储器中的保护
5.4.7 处理TLB缺失和缺页
5.4.8 小结
5.5 存储器层次结构的一般架构
5.5.1 问题1：一个块可以被放在何处
5.5.2 问题2：如何找到一个块
5.5.3 问题3：当cache缺失时替换哪一块
5.5.4 问题4：写操作如何处理
5.5.5 3C：一种理解存储器层次结构行为的直观模型
5.6 虚拟机
5.6.1 虚拟机监视器的必备条件
5.6.2 指令集系统结构（缺乏）对虚拟机的支持

5.6.3 保护和指令集系统结构
5.7 使用有限状态机来控制简单的cache
5.7.1 一个简单的cache
5.7.2 有限状态机
5.7.3 一个简单的cache控制器的有限状态机
5.8 并行与存储器层次结构：cache一致性
5.8.1 实现一致性的基本方案
5.8.2 监听协议
5.9 高级内容：实现cache控制器
5.10 实例：AMD Opteron X4(Barcelona)和Intel Nehalem的存储器层次结构
5.10.1 Nehalem和Opteron的存储器层次结构
5.10.2 减少缺失代价的技术
5.11 谬误和陷阱
5.12 本章小结
5.13 拓展阅读
5.14 练习题
第6章 存储器和其他I/O主题
6.1 引言
6.2 可信度、可靠性和可用性
6.3 磁盘存储器
6.4 快闪式存储器
6.5 连接处理器、内存以及I/O设备
6.5.1 互联基础
6.5.2 x86处理器的I/O互联
6.6 为处理器、内存和操作系统提供I/O设备接口
6.6.1 给I/O设备发送指令
6.6.2 与处理器通信
6.6.3 中断优先级
6.6.4 在设备与内存之间传输数据
6.6.5 直接存储器访问和内存系统
6.7 I/O性能度量：磁盘和文件系统的例子
6.7.1 事务处理I/O基准程序
6.7.2 文件系统和Web I/O的基准程序
6.8 设计I/O系统
6.9 并行性与I/O：廉价磁盘冗余阵列
6.9.1 无冗余(RAID 0)
6.9.2 镜像（RAID 1）
6.9.3 错误检测和纠错码（RAID 2）
6.9.4 位交叉奇偶校验（RAID 3）
6.9.5 块交叉奇偶校验（RAID 4）
6.9.6 分布式块交叉奇偶校验（RAID 5）
6.9.7 P+Q冗余（RAID 6）
6.9.8 RAID小结
6.10 实例：Sun Fire x4150服务器
6.11 高级主题：网络
6.12 谬误与陷阱
6.13 本章小结

6.14 拓展阅读
6.15 练习题
第7章 多核、多处理器和集群
7.1 引言
7.2 创建并行处理程序的难点
7.3 共享存储多处理器
7.4 集群和其他消息传递多处理器
7.5 硬件多线程
7.6 SISD、MIMD、SIMD、SPMD和向量机
7.6.1 在x86中的SIMD：多媒体扩展
7.6.2 向量机
7.6.3 向量与标量的对比
7.6.4 向量与多媒体扩展的对比
7.7 图形处理单元简介
7.7.1 NVIDIA GPU体系结构简介
7.7.2 深入理解GPU
7.8 多处理器网络拓扑简介
7.9 多处理器基准测试程序
7.10 Roofline：一个简单的性能模型
7.10.1 Roofline模型
7.10.2 两代Opteron的比较
7.11 实例：使用屋顶线模型评估四种多核处理器
7.11.1 4个多核系统
7.11.2 稀疏矩阵
7.11.3 结构化网格
7.11.4 生产率
7.12 谬误与陷阱
7.13 本章小结
7.14 拓展阅读
7.15 练习题
附录A 图形和计算GPU
A.1 引言
A.1.1 GPU发展简史
A.1.2 异构系统
A.1.3 GPU发展成了可扩展的并行处理器
A.1.4 为什么使用CUDA和GPU计算
A.1.5 GPU统一了图形和计算
A.1.6 GPU可视化计算的应用
A.2 GPU系统架构
A.2.1 异构CPU-GPU系统架构
A.2.2 GPU接口和驱动
A.2.3 图形逻辑流水线
A.2.4 将图形流水线映射到统一的GPU处理器
A.2.5 基本的统一GPU结构
A.3 可编程GPU
A.3.1 为实时图形编程
A.3.2 逻辑图形流水线

A.3.3 图形渲染程序
A.3.4 像素渲染示例
A.3.5 并行计算应用编程
A.3.6 使用CUDA进行可扩展并行编程
A.3.7 一些限制
A.3.8 体系结构隐含的问题
A.4 多线程的多处理器架构
A.4.1 大规模多线程
A.4.2 多处理器体系结构
A.4.3 单指令多线程（SIMT）
A.4.4 SIMT warp执行和分支
A.4.5 管理线程和线程块
A.4.6 线程指令
A.4.7 指令集架构（ISA）
A.4.8 流处理器（SP）
A.4.9 特殊功能单元（SFU）
A.4.10 与其他多处理器的比较
A.4.11 多线程多处理器总结
A.5 并行存储系统
A.5.1 DRAM的考虑
A.5.2 cache
A.5.3 MMU
A.5.4 存储器空间
A.5.5 全局存储器
A.5.6 共享存储器
A.5.7 局部存储器
A.5.8 常量存储器
A.5.9 纹理存储器
A.5.10 表面
A.5.11 load/store访问
A.5.12 ROP
A.6 浮点算术
A.6.1 支持的格式
A.6.2 基本算术
A.6.3 专用算术
A.6.4 性能
A.6.5 双精度
A.7 资料：NVIDIA GeForce 8800
A.7.1 流处理器阵列（SPA）
A.7.2 纹理/处理器簇（TPC）
A.7.3 流多处理器（SM）
A.7.4 指令集
A.7.5 流处理器（SP）
A.7.6 特殊功能单元（SFU）
A.7.7 光栅化
A.7.8 光栅操作处理器（ROP）和存储系统
A.7.9 可扩展性

A.7.10 性能
A.7.11 密集线性代数性能
A.7.12 FFT性能
A.7.13 排序性能
A.8 资料：将应用映射到GPU
A.8.1 稀疏矩阵
A.8.2 在共享存储器中进行缓存
A.8.3 扫描和归约
A.8.4 基数排序
A.8.5 GPU上的N-Body应用
A.9 谬误与陷阱
A.10 小结
A.11 拓展阅读
附录B 汇编器、链接器和SPIM仿真器
B.1 引言
B.1.1 什么时候使用汇编语言
B.1.2 汇编语言的缺点
B.2 汇编器
B.2.1 目标文件的格式
B.2.2 附加工具
B.3 链接器
B.4 加载
B.5 内存的使用
B.6 过程调用规范
B.6.1 过程调用
B.6.2 过程调用举例
B.6.3 另外一个过程调用的例子
B.7 异常和中断
B.8 输入和输出
B.9 SPIM
B.10 MIPS R2000汇编语言
B.10.1 寻址方式
B.10.2 汇编语法
B.10.3 MIPS指令编码
B.10.4 指令格式
B.10.5 常数操作指令
B.10.6 比较指令
B.10.7 分支指令
B.10.8 跳转指令
B.10.9 陷阱指令
B.10.10 取数指令
B.10.11 保存指令
B.10.12 数据传送指令
B.10.13 浮点运算指令
B.10.14 异常和中断指令
B.11 小结
B.12 参考文献

B.13　练习题

章节摘录

版权页：插图：存储器层次结构可以由多层构成，但是数据每次只能在相邻的两个层次之间进行复制。
因此我们将注意力重点集中在两个层次上。
高层的存储器靠近处理器，比低层存储器容量小且访问速度更快，这是因为它采用了成本更高的技术来实现的。
如图5-2中所示，我们将一个两级层次结构中存储信息的最小单元称为块（block）或行（line），就像在图书馆中，一个信息块就是一本书。
存储系统采用层次结构后，用户对于存储器的认识就是：它的容量和层次结构和容量最大的那层存储器相同，而访问速度和最快的那层存储器相当。
在很多嵌入式系统中，闪存已经代替了磁盘，对于台式计算机和服务器来说可能会在存储层次中引入新的一层；见6.4 节。
被认为一个是高层次，一个是低层次在每一层中，那些存储信息的最小单元被称为块或者行。
通常在层次之间复制时按整块进行传输。
如果处理器需要的数据存放在高层存储器中的某个块中，则称为一次命中（这就好像正好从书桌上的一本书中找到所需的信息一样）。
如果在高层存储器中没有找到所需的数据，这次数据请求则称为一次缺失。
随后访问低层存储器来寻找包含所需数据的那一块（如同从书桌旁走到书架前去寻找所需的书籍）。
命中率，或命中比率，是在高层存储器中找到数据的存储访问比例，通常被当成存储器层次结构性能的一个衡量标准。
缺失率（1-命中率）则是数据在高层存储器中没有找到的存储访问比例。
追求高性能是我们使用存储器层次结构的主要目的，因而命中和缺失的执行时间就显得尤为重要。
命中时间是指访问存储器层次结构中的高层存储器所需要的时间，包括了判断当前访问是命中还是缺失所需的时间（浏览书桌上书籍所花费的时间）。
缺失代价是将相应的块从低层存储器替换到高层存储器中，以及将该信息块传送给处理器的时间之和（从书架上取另一本书并将它放到桌上的时间），由于较高存储层次容量较小并且使用了快速的存储器部件，因此比起对存储层次中较低层的访问，命中时间要少得多，这也是缺失代价的主要组成部分。
（同样，查找书桌上书籍的时间比站起来到书架前查找一本新书所需的时间要少得多）。
在这一章中我们也将看到，用来构建存储器层次结构的这些概念也将影响到一台计算机的许多其他方面，包括操作系统如何管理存储器和I/O，编译器如何产生代码，甚至对应用程序如何使用计算机也产生一定影响。
当然，由于所有程序花费大量时间访问存储器，因而存储系统必然成为评估机器性能的一个主要指标。
利用存储器层次结构来达到性能的提升意味着，在过去程序员可以把存储器看成是一个平台随机访问存储设备，而现在必须理解存储层次结构如何工作才能获得良好的性能。
稍后我们将举例来说明其重要性（见图5-18）。

编辑推荐

《计算机组成与设计:硬件、软件接口(原书第4版)》特色：涵盖从串行计算到并行计算的革命性变革，新增了关于并行化的一章，并且每章中还有一些强调并行硬件和软件主题的小节。
新增一个由NVIDIA的首席科学家和架构主管撰写的附录，介绍了现代GPU的出现和重要性，首次详细描述了这个针对可视计算进行了优化的高度并行化、多线程、多核的处理器。
描述一种度量多核性能的独特方法——Roofline模型，自带AMDOpteronX4、IntelXeon5000、SunUltraSPARCT2和IBM Cell的基准测试和分析。
涵盖一些关于闪存和虚拟机的新内容。
提供了大量富有启发性的练习题。
将AMDOpteronX4和Inter Nehalem作为贯穿《计算机组成与设计:硬件、软件接口(原书第4版)》的实例。
用SPECCPU2006组件更新了所有处理器性能实例。
这本最畅销的计算机组成书籍经过全面更新，关注现今发生在计算机体系结构领域的革命性变革二从单处理器发展到多核微处理器，从串行发展到并行。
与前几版一样，《计算机组成与设计:硬件、软件接口(原书第4版)》采用了MIPS处理器来展示计算机硬件技术、汇编语言、计算机算术、流水线、存储器层次结构以及I／O等基本功能。
此外，奉书还包括一些关于ARM和x86体系结构的介绍。

<<计算机组成与设计>>

版权说明