计算机组成 Lecture Notes 7

本文主要探讨一些比较新的技术，如多核、并行、分布式计算。

多处理器简介

至少拥有两个处理器的计算机系统称为**多处理器系统（Multiprocessor）**。与之对应的为单处理器。

在一个多处理器的微处理器芯片上，一个处理器被称为**核（Core）**，因此"多处理器微处理器"可称为**多核微处理器（Multicore Microprocessor）**。核的数量预计以摩尔定律增长。

> 微处理器是 CPU 的一种单芯片的实现。也就是说，多核微处理器是将多个处理器集成到了一个芯片上。

并行和并发

以例子说明串行（Serial）、并行、顺序（Sequential）、并发（Concurrent）的区别。

		软件	
		顺序	并发
硬件	串行	在 Intel Pentium4 上运行的使用 Matlab编写的矩阵乘法	在 Intel Pentium4 上运行的 Windows Vista 操作系统
	并行	在 Intel Core i7 上运行的使用 Matlab 编写的矩阵乘法	在 Intel Core i7 上运行的 Windows Vista 操作系统

可以看出：

- 串行和并行是描述硬件的，而顺序和并发是描述软件的。
- 顺序软件和并发软件都可以运行在串行或者并行硬件上。

多处理器系统的性能很高，可以以两种方式表现：

- **任务级并行（Task-level Parallelism）**或者**进程级并行（Process-level Parallelism）**：通过在各个处理器上同时运行独立程序的方法来利用多处理器。
- **并行处理程序（Parallel Processing Program）**或**并行软件（Parallel Program）**：同时运行在多个处理器上的单一程序。这一程序可以是顺序的，也可以是并发的。

并行的难点

多处理器看起来很好，但并行处理程序不好开发，且在多处理器上编写程序来提升执行效率很困难。

原因：

- 单处理器本身有指令级并行的技术，如超标量、乱序执行等。除非能在多处理器上用并行处理程序能获得更高的性能或者功耗利用率，否则在单处理器上用更容易编写的顺序程序就够了。
- 并行处理程序面临着调度、分割任务、负载均衡、同步、通信等问题。
- Amdahl 定律告诉我们，多核是有极限的。

并行硬件的分类

一种基于指令流和数据流的数量的分类方法：

		数据流	
		单	多
指令流	单	SISD：Intel Pentium 4	SIMD：x86 的 SSE 指令
	多	MISD：至今没有实例	MIMD：Intel Core i7

- 常规的单处理器为 **SISD（单指令流单数据流）**。
- 常规的多处理器为 **MIMD（多指令流多数据流）**。

通常在 MIMD 计算机上运行单一程序，将其运行在所有处理器上。这种风格为 **SPMD（单程序多数据）**。

SIMD 可用同样的指令在多个数据流上操作。其优点：

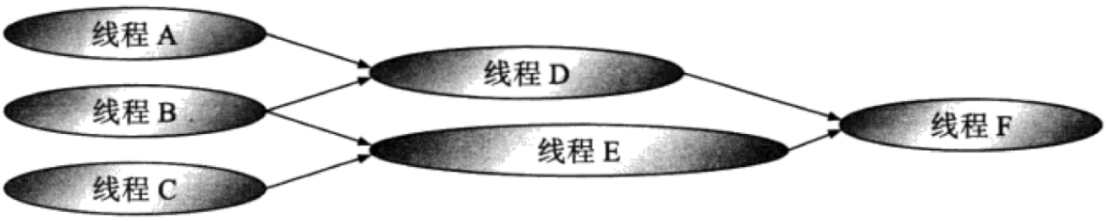
- 所有的并行执行单元都是同步的，它们都源于同一个 PC，但它们都有各自的地址寄存器。对于编程者，这和 SISD 很像。
- 可以降低指令宽度和所占空间。SIMD 每个处理器同时执行同一套指令，而 MIMD 可能每一个处理器都要有一套副本。

硬件多线程

概念：

- **进程（Process）**：一个进程包含一个或多个线程、地址空间和操作系统状态。
  - 进程间一般不共享地址空间。
  - 进程切换通常需要操作系统介入。
  - 进程切换要花成百上千个时钟周期。
- **线程（Thread）**：一个线程包括 PC、寄存器状态及栈。可以看成是一个轻量版的进程。
  - 线程间可以共享地址空间。
  - 线程切换可以不需要操作系统介入。
  - 线程切换可以瞬时完成。

线程可能发生阻塞，即因为某些原因（如依赖）暂停运行一段时间。例如：



此时 D 和 E 理论上可以并行，但 F 需等待 D、E 完成才可以执行。

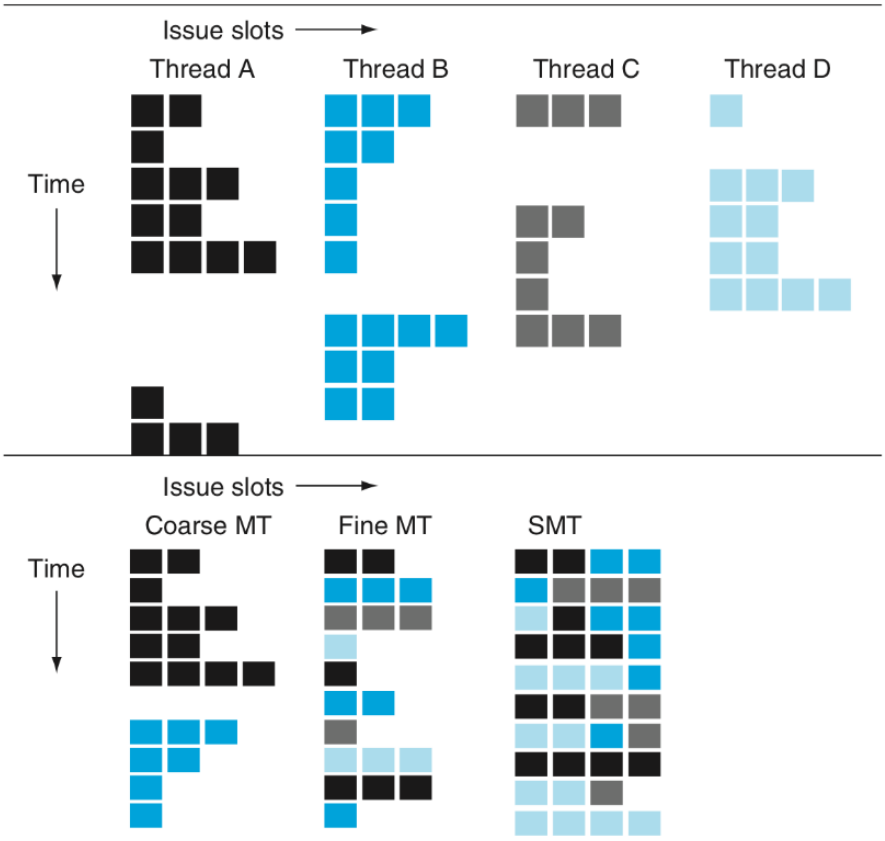
从程序员的角度来看，**硬件多线程（Hardware Multithreading）**就是一个和 MIMD 相关的概念。两种实现方法：

- **细粒度多线程（Fine-grained Multithreading）**：每条指令执行后都进行线程切换。多个线程间交叉循环执行，每个时钟周期跳过处于阻塞状态的线程。
  - 优点：可同时减少由短阻塞和长阻塞引起的吞吐量缺失。因为阻塞时可以运行其他线程的指令。
  - 缺点：降低了单个线程的执行速度。因为 ready 的线程会因为别的线程延迟执行。
- **粗粒度多线程（Coarse-grained Multithreading）**：仅在长阻塞时线程切换。
  - 优点：几乎不会降低单个线程的执行速度，可降低长阻塞引起的吞吐量缺失。
  - 缺点：使用流水线时，难以应对由短阻塞引起的吞吐量缺失。因为切换线程时，需要清空或者冻结流水线。短阻塞频繁发生时，流水线的启动开销是巨大的。而对于长阻塞，启动开销

可忽略不计。

硬件多线程的一个变种是 **SMT（同时多线程）**。它使用多发射、动态调度微架构的资源来挖掘线程级并行，同时保持指令级并行。换言之，其在*保持了线程切换的同时，也尽可能让不同的线程指令并行执行*。

图示：



从左到右表示指令的发射槽，从上到下表示时间。同一时间可以多条指令运行。不同颜色表示不同线程。方便起见，图中未画出粗粒度的流水线启动开销。

多线程 CPU 和多核 CPU

一个多线程 CPU 会有多个虚拟处理器，每一个虚拟处理器上跑一个线程。它和真的处理器的差别在于虚拟处理器共享部分功能单元。

多核 CPU 中的核可以是多线程 CPU。例如有的处理器是“6 核 12 线程”，那么它每一个核就是双线程 CPU。

多处理器通信

有两种方式实现处理器之间的通讯：

- 共享同一个物理地址空间，即 SMP。
- 不共享同一个物理地址空间，通过显式的消息传递机制通信。

SMP

SMP（共享内存处理器，Shared Memory Processor）：

- 物理地址共享。
- 每个处理器可以在自己的虚拟地址空间运行自己独立的程序。
- 处理器间可以用存储器中的共享变量通讯。

SMP 可以分成两种：

- **UMA（统一存储访问，Uniform Memory Access）**：访存时间不依赖于具体哪个处理器或者哪个地址。
- **NUMA（非统一存储访问，Nonuniform Memory Access）**：访存时间和具体哪个处理器或者哪个地址有关。距离较近的访问可能比较短。
  - 包括 CC-NUMA（cache 一致的 NUMA）。
  - 它的编程难度更高，但是可以扩展到更大规模。

同步和锁

共享数据需要数据同步。如果有来自两个不同线程的两个访存请求访问同一个地址，它们连续出现，且至少其中一个是写操作，那么这就发生了**数据竞争（Data-race）**。

可以使用**锁（Lock）**解决。

- 任何时刻，对于一个共享数据，只有一个处理器可以拥有锁，并对数据操作。
- 其他要处理该数据的处理器需要等待，直到解锁。

在硬件层面，可以通过**原子交换（Atomic Exchange 或 Atomic Swap）**实现加锁和解锁。这个操作是不可分割的，作用是将一个寄存器中的值和存储器中的一个值交换。

对于存储器中某锁变量，试图对其加锁，可以这么做：

- 设置某个寄存器为 1（或者非零）。
- 将该寄存器与目标做原子交换。
  - 交换后寄存器为 1（或者非零），那么表示目标已被上锁。续进一步等待。
  - 交换后寄存器为 0，那么目标自由，成功上锁。

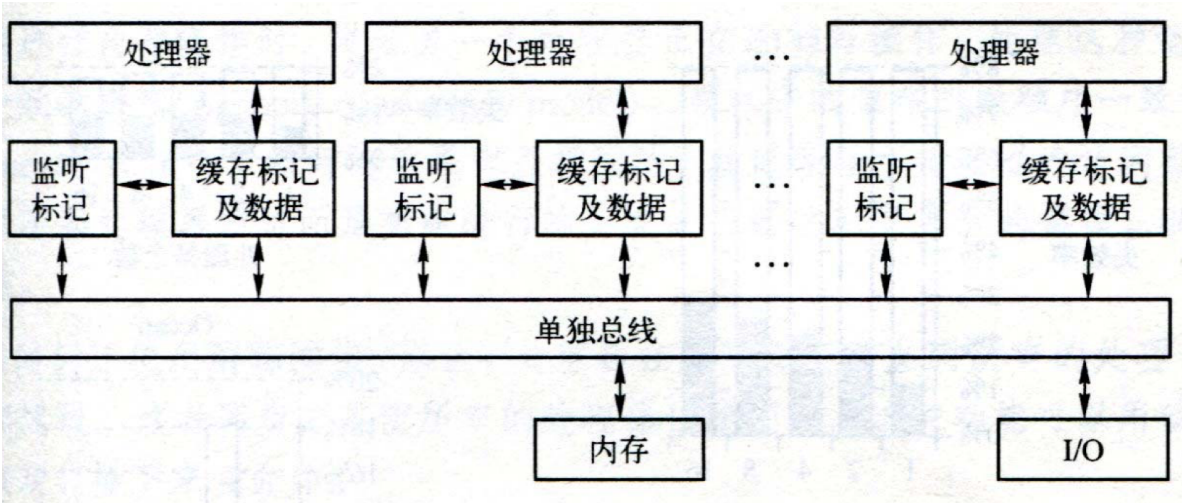
试图解锁，可以直接对目标赋 0。

这种原子操作的设计并不容易，原因在于其一般要分为两步：先读锁变量，再写锁变量。

在 MIPS 中，加锁被设计为一个指令对：`ll` 和 `sc`。

一致性

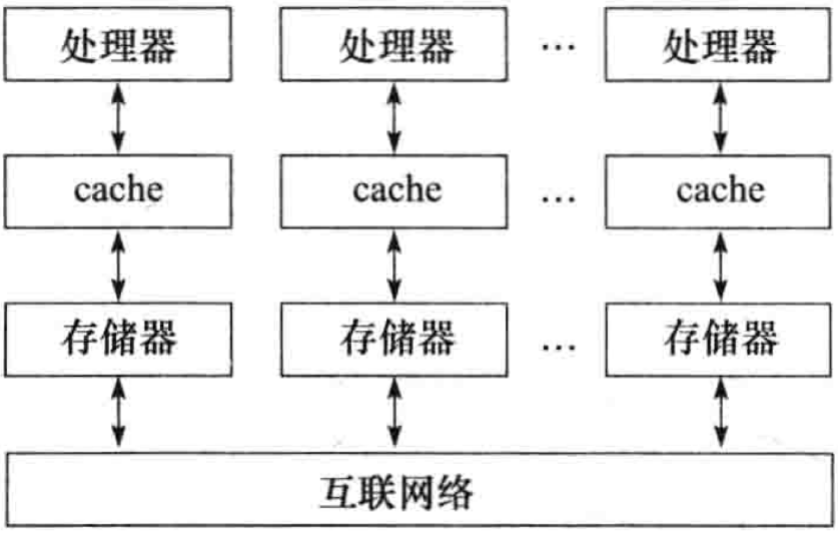
共享数据还会产生 **cache 一致性（Cache Coherence）**的问题。使用监听技术可以解决。



例如写无效（write invalidate）策略：当某个处理器修改了自己的 cache，那么对应的块地址会在总线上被广播，其他处理器的监听器看到了之后会检查自己的 cache，如果有的话就会把对应块无效化。

还可以当别的 cache 要从存储器中读时，在总线上广播，如果某个处理器写了对应的块，那么就先让它写回。

消息传递计算机



计算机集群

集群（Cluster）是通过局域网连接的一组计算机，其作用等同于一个大型的多处理器。

集群中的单个计算机通常称为节点。

致谢

本文的主要内容参考自：

- 上海交通大学《计算机组成》（课程代号：EI332）一课的课程材料
- 《Computer Organization and Architecture (10th edition)》
- 《Computer Organization and Design: Hardware/Software Interface (5th edition)》

本文的图片来自

- 《计算机组成与设计：硬件/软件接口（第 5 版）》
- 《Computer Organization and Design: Hardware/Software Interface (5th edition)》
- 《计算机原理与设计：Verilog HDL 版》