LOCKUP-FREE INSTRUCTION FETCH/PREFETCH CACHE ORGANIZATION

作者在这篇论文中提出了 non-blocking cache。这对于提高计算机处理器的效率具有重要意义，主要通过减少 cache miss penalty 的方法，以增强 cache 的性能

MSHRs（Miss Status Holding Registers） 是实现 non-blocking cache 的核心组件。它们用于追踪缓存失效（cache miss）请求的状态，并管理这些请求的返回数据。在一种典型的实现中，每个 MSHR 对应一个 cache block，然后每个 MSHR 有若干 subblock，每个 subblock 记录了 cpu 请求的地址在 cache block 中的偏移。这样若干个访问同一个 cache block 的内存请求会被合并到一个 MSHR 中。当一个内存请求得到下层 cache 的答复时，将数据回传给 cpu，同时根据地址清除掉对应 MSHR 表项中的 subblock，当一个 MSHR 中所有 subblock 都被清除后，这个 MSHR 被释放，可被分配给新的 missed cache request

A VLIW architecture for a trace scheduling compiler

Fisher 于 1983 年提出了超长指令字(Very Long Instruction Word, VLIW)的计算机体系结构和编译技术。此论文中，Fisher 通过详细研究与实例，展现了 VLIW 体系结构与轨迹调度(Treac Scheduling)编译器如何协同工作，进而提升程序的执行效率。 VLIW 体系结构的关键概念在于：通过并行执行多个指令来提高处理器性能。具体来说，传统的 RISC（精简指令集计算）与 CISC（复杂指令集计算）体系结构主要追求单一指令流的优化。然而，VLIW 体系结构通过将多条指令组合为一个超长指令字，一次性发出，允许多个功能单元同时并行操作。这种设计极大地减少了传统流水线处理中的控制开销和数据依赖性带来的停顿。此外，硬件设计相对简单，避免了分支预测和动态调度复杂性

本文的作者面向 VLIW 体系结构，设计了 TRACE 机器，这台机器由 4 个 I board 和 4 个 F board 组成，I board 和 F board 分别用来执行整点和浮点指令。每个 I board 和 F board 都有两个 ALU，一个 I-F 对指令长 256bits，在这四个 ALU 上执行运算，因此 TRACE 机器的一条指令长为 1024bits，最多能同时执行 16 个运算指令

在编译层面，TRACE 机器的性能依赖于编译器的精准调度，因为它把每个运算指令的执行周期，延迟都暴露给了编译器。为了将更多指令打包到 VLIW 中，TRACE 机器实现了有多个分支跳转条件和跳转目标的指令

在内存系统上，TRACE 机器认为指令缓存能够显著提升取指效率，但数据缓存并不总是这样，许多科学计算程序并没有良好的数据局部性，使用数据缓存反而可能增加数据的访存延迟。因此在 TRACE 机器上并没有使用 Cache，而是多 bank 的 interleaved memory system。为了提升内存效率，要求在一条 VLIW 指令中不能有两个访存指令访存到同一个 bank 上，这也给编译器提出了挑战，这要求编译器通过指针分析等方法保守地静态预测每个访存指令所在的 bank，只有当编译器能够证明这些访存指令不在一个 bank 上时，才能将它们打包到一条 VLIW 指令中

另外，TRACE 机器中实现了虚拟内存，每个 I board 上都配置有 DTLB。但为了并行执行的效率，TRACE 机器并没有实现精确中断。为了在发生 tlb miss 时找出尚未完成的内存指令，TRACE 机器中引入了 history queue，这个 queue 中记录了尚未完成了内存指令和它们的虚拟地址，由 trap handler 负责执行这些指令

总的来说，VLIW 为计算机体系结构设计提供了一种全新的思路，通过硬件和软件的协同优化，实现了处理器性能的显著提升

Simultaneous Multithreading: Maximizing On-Chip Parallelism

这篇论文对同时多线程（Simultaneous Multithreading, SMT）提出了详尽的分析和设计，使其成为现代处理器架构研究和开发的重要参考。在论文讨论了四种线程级并行的设计方式

- superscalar：超标量设计，没有线程级并行，挖掘单一线程内的指令级并行
- traditional multithreaded processors：传统的多线程并行方式，每个周期选择一个线程发射指令，但同一个周期只能有一个线程发射指令
- simultaneous multithreaded processors：同时多线程模式，每个周期所有线程都可以发射指令。一个周期内发射槽可以接收来自多个线程的指令
- small-scale multiple-issue multiprocessors：多处理器设计，在一个芯片上集成多个处理器，各个处理器单独运行各自的线程

论文中首先分析了 superscalar 的性能瓶颈，进行模拟发现，一个 8 发射槽的 superscalar 处理器在执行 spec 测试程序中仅能达到大约 1.5 的 IPC。作者将 superscalar 处理器的功能单元的低利用率的原因分为两类

- Horizontal waste，指在同一个周期发射的指令没能够占满所有的发射槽
- Vertical waste，指因为 cache miss 或者数据，控制依赖等原因，这一个周期发射槽没有发射指令

比较传统的多线程模式和同时多线程模式，可以发现传统的多线程模式没办法消除 horizontal waste，它只能将 vertical waste 转化为 horizontal waste。而同时多线程模式则具有消除 horizontal waste 和 vertical waste 的潜力

但是完整的同时多线程处理器具有巨大的硬件复杂度，因此作者在接下来的模拟实验中采取了多种可能的对同时多线程处理器的约束，并测试了各种约束对性能的影响。这些约束包括（假设是 8 个发射槽的处理器）

- 每个线程每个周期至多发射 1 / 2 / 4 条指令
- 对处理单元进行分区，一部分处理单元仅供这些线程使用，另一部分处理单元仅供其余线程使用等等

作者通过实验发现，在各种约束条件下，同时多线程处理器仍然取得了比传统的多线程处理器更高的 IPC

但是，同时多线程处理器对 cache 很不友好，因为它同时运行多个线程，指令和数据的局部性更差，作者在实验中发现，随着线程数目的增多，cache 和 tlb 的命中率逐渐下降。因此，作者提出了四种可能的同时多线程处理器的 cache 设计

- 私有指令 cache vs 共享指令 cache
- 私有数据 cache vs 共享数据 cache

它们两两组合，共有四种可能，作者通过实验发现，线程数目较小时，共享指令 cache 与共享数据 cache 搭配取得了最好的效果。但随着线程数目的增加，私有指令 cache 与共享数据 cache 搭配取得了最好的效果

因此本文通过详细的模拟实验，表明同时多线程是最具有潜力的线程并行的处理器模型

The Tera Computer System

Tera 计算机系统是由美国华盛顿州的Tera计算公司（现称为Cray Inc.）开发的一套高性能并行超级计算机系统。伴随日益增长的大规模科学计算和数据处理需求，传统计算机体系结构在并行性和扩展性方面面临很大瓶颈。Tera系统的设计就是为了解决这些瓶颈，使得计算性能能在更多并行任务负荷下得到有效提升

Tera 系统强调处理器和内存之间的高带宽以及线程级并行性的扩展。Tera计算机系统采用了称为"高级同步并行（ADVANCED SYNCHRONOUS PARALLEL，ASP）"的设计，即处理器以相同的指令集架构并行运行许多线程，而且这些线程之间进行同步操作。这种具有锁存功能的多线程结构有助于在大量并行运算时减少数据冲突及等待时间，从而显著提高系统整体性能。具体来说，Tera 使用 mesh 网格结构互联了 256 个处理器，512 个内存单元，256 个 IO 处理单元以及 256 个 IO cache 单元。每个处理器又可以支持最多 128 个线程 （文中使用了 stream 这个术语）同时运行

Tera 的处理器的指令借鉴了 VLIW 的设计，每条指令包含三条实际执行的指令：第一条是访存指令，第二条是运算指令，第三条是条件跳转指令或者运算指令。为了支持多个线程同时运行，每个处理器中包含大量的通用寄存器，以此避免上下文切换的开销。每个线程会占用 32 个 64bits 的通用寄存器，因此处理器总共包含 4096 个 64bits 的通用寄存器。有如此庞大的寄存器堆，如果需要硬件来判断指令间的依赖冲突的话，需要使用一个庞大的计分板。Tera 处理器没有使用硬件来判断指令间依赖。处理器每个周期发射一条指令，但每个周期选择不同的线程进行执行，直到某一线程的一条指令执行完毕后，这个线程才被允许发射下一条指令。论文中谈到一条指令平均的延迟是 70 个周期，因此为了保证处理器能够被充分利用，需要 70 个线程同时运行。为了进一步降低充分利用处理器需要的线程数量，Tera 处理器中使用了 lookahead 的技术。每条指令中包含 3bits 的提示位，表示接下来多少条指令与当前指令没有数据依赖关系。3bits 最大可以表示 7 条指令，因此一个 stream 最多可以有 8 条指令在处理器中同时执行。因此在指令间依赖关系很少时，lookahead 技术使得 9 个线程就可以充分利用处理器

由于处理器上运行着大量的线程，它们需要进行同步。Tera 使用了 tagged memory，为每个 data word 提供额外的状态位来实现线程间的同步，其中 1bit 指示当前 word 处于 full 还是 empty 状态。访问内存的地址中，两个高位 bits 不用来定位地址，而是指示 access control，当其值为 3 时，如果是 load 指令，该 load 仅在对应的地址的状态位为 full 时才成功，并且成功时会将状态位改写为 empty。而如果是 store 指令，那么该 store 仅在对应地址的状态位为 empty 时才成功，并且成功时会将状态位改写为 full。可以看到，这样的机制就提供了生产者，消费者的同步源语

Tera计算机系统的革新性设计为后续很多高性能计算（HPC）机型提供了重要参考和启发