H1 3.12 Multithreading: Exploiting Thread-Level Parallelism to Improve Uniprocessor Throughput

多线程是一个交叉话题，其与流水线和超标量都有关系，使用线程级并行。

问题：是否有另外的并行形式用于隐藏内存延迟？--多线程

多线程允许多个线程以重叠的方式共享单个处理器的功能单元。

复制处理器核心的每个线程状态意味着为每个线程创建一个单独的寄存器文件、一个单独的PC和一个单独的页表。内存本生通过虚拟内存机制进行共享。

实现多线程的硬件方法

- 细粒度多线程

  在每个时钟的线程之间切换，导致多个线程的指令执行被交叉

  - 优点：可以隐藏由短暂停和长暂停引起的吞吐量损失
  - 缺点：减慢了单个线程的执行速度，用多线程吞吐量的减少来交换单线程性能的损失

- 粗粒度多线程

  作为细粒度多线程的替代而发明。粗粒度多线程只在代价高昂的暂停时切换线程，比如二级或三级缓存未命中。

  - 优点：减少了让线程切换基本自由的需求
  - 缺点：克服吞吐量损失的能力有限

- 同调多线程

  细粒度多线程的一种变体，当细粒度多线程在多期、动态调度的处理器上实现时采用这种方式。

  - 关键点：寄存器重命名和动态调度允许执行来自独立线程的多条指令，而不考虑它们之间的依赖关系；依赖项的解析可以通过动态调度能力来处理。

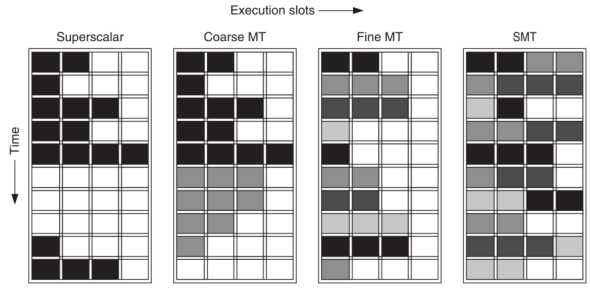

Figure 3.28 How four different approaches use the functional unit execution slots of a superscalar processor. The horizontal dimension represents the instruction execution capability in each clock cycle. The vertical dimension represents a sequence of clock cycles. An empty (white) box indicates that the corresponding execution slot is unused in that clock cycle. The shades of gray and black correspond to four different threads in the multithreading processors. Black is also used to indicate the occupied issue slots in the case of the superscalar without multithreading support. The Sun T1 and T2 (aka Niagara) processors are fine-grained multithreaded processors, while the Intel Core i7 and IBM Power7 processors use SMT. The T2 has eight threads, the Power7 has four, and the Intel i7 has two. In all existing SMTs, instructions issue from only one thread at a time. The difference in SMT is that the subsequent decision to execute an instruction is decoupled and could execute the operations coming from several different instructions in the same clock cycle.

H2 Effectiveness of Fine-Grained Multithreading on the Sun T1

T1完全专注于开发线程级并行，使用多核和多线程来产生吞吐量

Characteristic	Sun T1
Multiprocessor and multithreading support	Eight cores per chip; four threads per core. Fine-grained thread scheduling. One shared floating-point unit for eight cores. Supports only on-chip multiprocessing.
Pipeline structure	Simple, in-order, six-deep pipeline with three-cycle delays for loads and branches.
L1 caches	16 KB instructions; 8 KB data. 64-byte block size. Miss to L2 is 23 cycles, assuming no contention.
L2 caches	Four separate L2 caches, each 750 KB and associated with a memory bank. 64-byte block size. Miss to main memory is 110 clock cycles assuming no contention.
Initial implementation	90 nm process; maximum clock rate of 1.2 GHz; power 79 W; 300 M transistors; 379 mm^2 die.

Figure 3.29 A summary of the T1 processor.

H3 T1 Multithreading Unicore Performance

为了检查T1的性能，使用三个面向服务器的基准测试：TPC-C、SPECJBB、SPECWeb99。

图3.30显示了TPC-C中，每核执行一个线程与每核执行四个线程时，失效率和观察到的失效等待时间的相对增加。

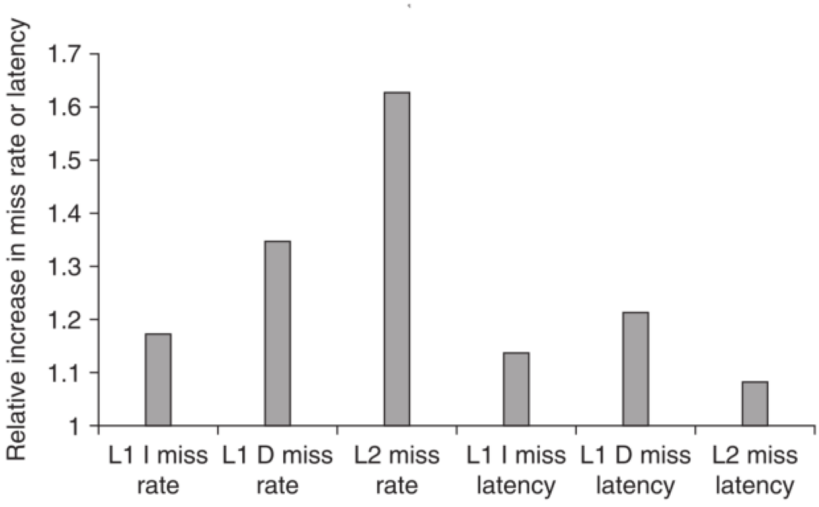

Figure 3.30 The relative change in the miss rates and miss latencies when executing with one thread per core versus four threads per core on the TPC-C benchmark. The latencies are the actual time to return the requested data after a miss. In the four-thread case, the execution of other threads could potentially hide much of this latency.

线程可能因为缓存丢失、流水线延迟以及各种较小的影响而无法就绪。

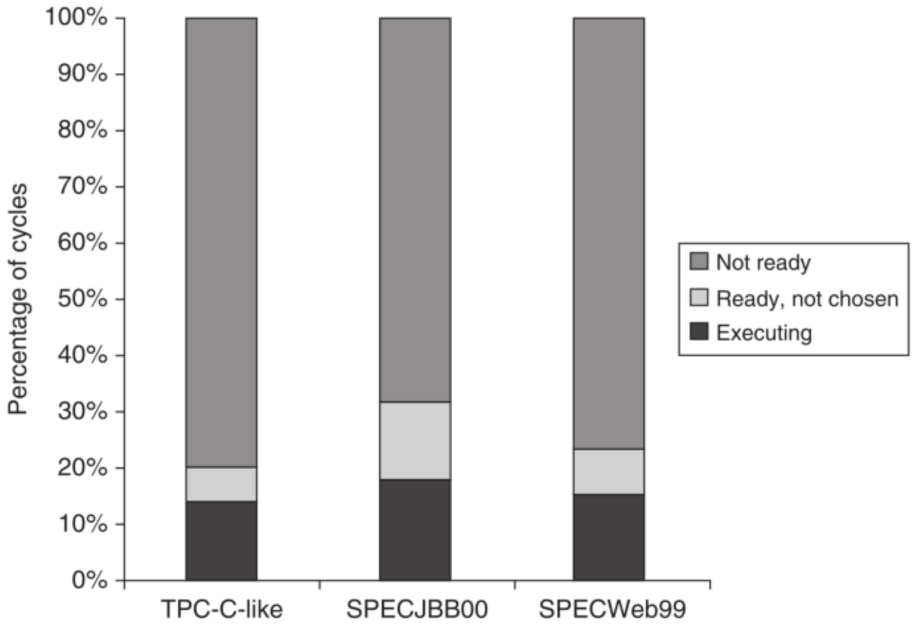

Figure 3.31 Breakdown of the status on an average thread. "Executing" indicates the thread issues an instruction in that cycle. "Ready but not chosen" means it could issue but another thread has been chosen, and "not ready" indicates that the thread is awaiting the completion of an event (a pipeline delay or cache miss, for example).

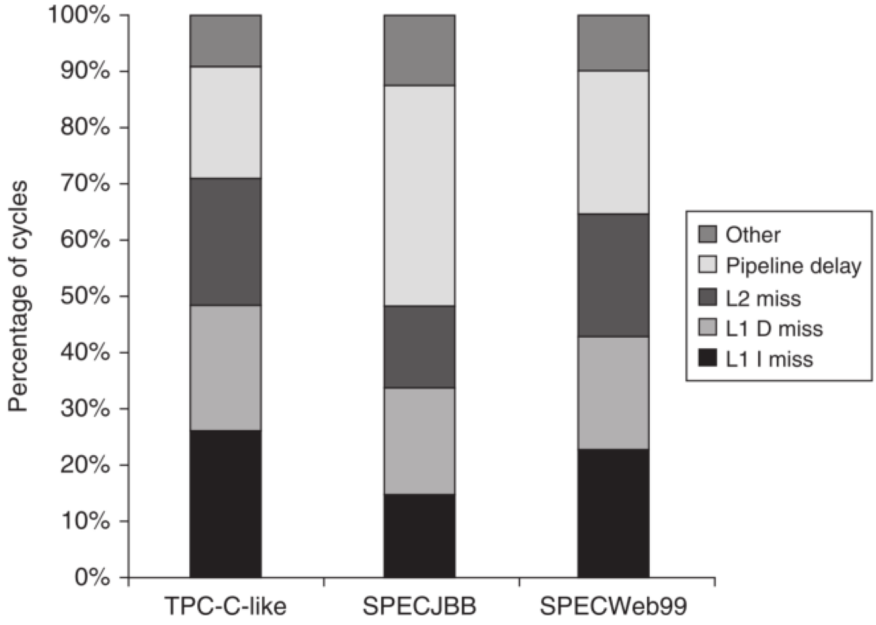

Figure 3.32 The breakdown of causes for a thread being not ready. The contribution to the "other" category varies. In TPC-C, store buffer full is the largest contributor; in SPEC-JBB, atomic instructions are the largest contributor; and in SPECWeb99, both factors contribute.

Benchmark	Per-thread CPI	Per-core CPI
TPC-C	7.2	1.80
SPECJBB	5.6	1.40
SPECWeb99	6.6	1.65

Figure 3.33 The per-thread CPI, the per-core CPI, the effective eight-core CPI, and the effective IPC (inverse of CPI) for the eight-core T1 processor.

H2 Effectiveness of Simultaneous Multithreading on Superscalar Processors

SMT可以以高效节能的方式提高性能

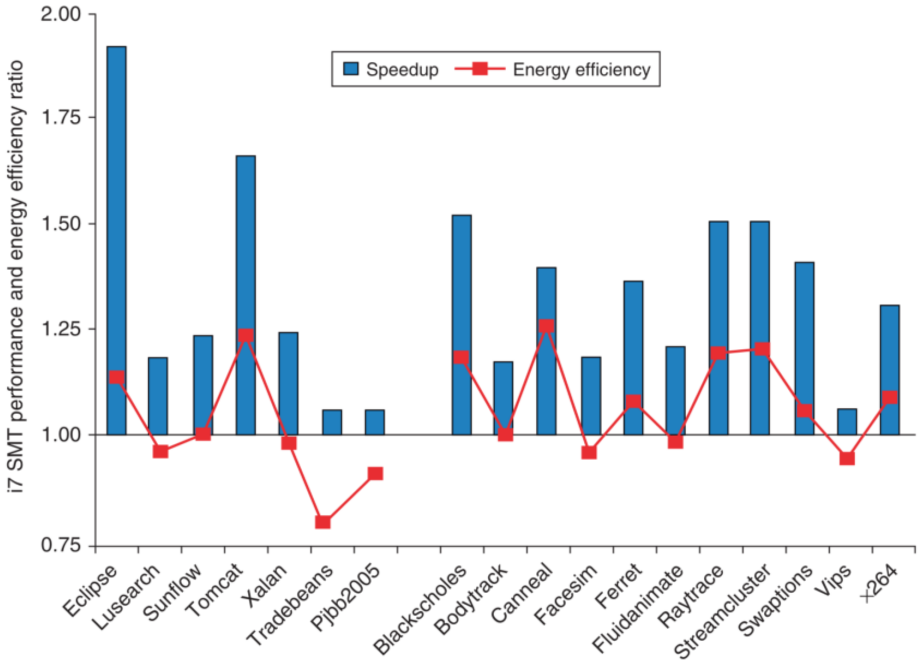

Figure 3.35 The speedup from using multithreading on one core on an i7 processor averages 1.28 for the Java benchmarks and 1.31 for the PARSEC benchmarks (using an unweighted harmonic mean, which implies a workload where the total time spent executing each benchmark in the single-threaded base set was the same). The energy efficiency averages 0.99 and 1.07, respectively (using the harmonic mean). Recall that anything above 1.0 for energy efficiency indicates that the feature reduces execution time by more than it increases average power. Two of the Java benchmarks experience little speedup and have significant negative energy efficiency because of this. Turbo Boost is off in all cases. These data were collected and analyzed by Esmaeilzadeh et al. [2011] using the Oracle (Sun) HotSpot build 16.3-b01 Java 1.6.0 Virtual Machine and the gcc v4.4.1 native compiler.

H1 3.13 Putting it All Together: The Intel Core i7 and ARM Cortex-A8

H2 The ARM Cortex-A8

A8是一个两发射，静态调度超标量与动态检测，它允许处理器每个时钟周期发出一个或两个指令。

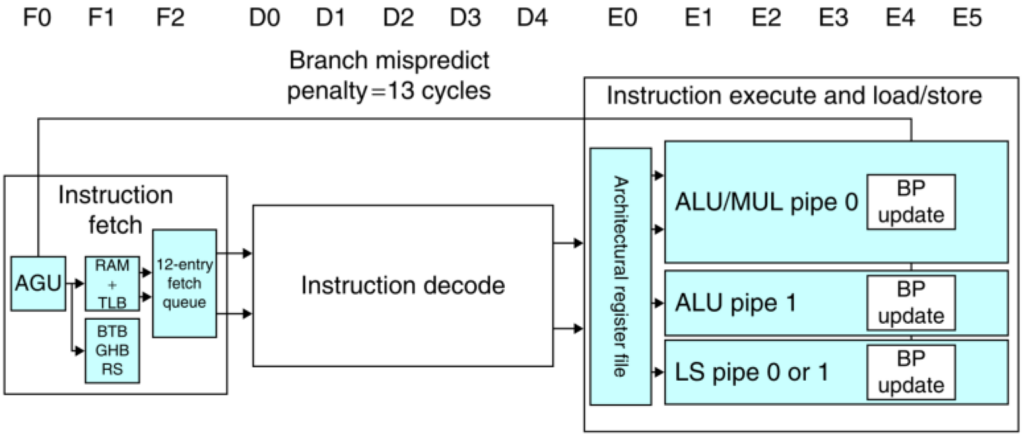

Figure 3.36 The basic structure of the A8 pipeline is 13 stages. Three cycles are used for instruction fetch and four for instruction decode, in addition to a five-cycle integer pipeline. This yields a 13-cycle branch misprediction penalty. The instruction fetch unit tries to keep the 12-entry instruction queue filled.

A8使用一个动态分支预测器，具有一个512输入双向集合关联分支目标缓冲区和一个4K输入全局历史缓冲区，该缓冲区由分支历史和当前PC进行索引。在分支目标缓冲区未命中的情况下，从全局历史缓冲区获得一个预测，然后可以使用该预测来计算分支地址。

图3.37显示了指译码流水线。每个时钟最多顺序发射两条指令，使用一个记分牌用于跟踪何时可以发出指令。

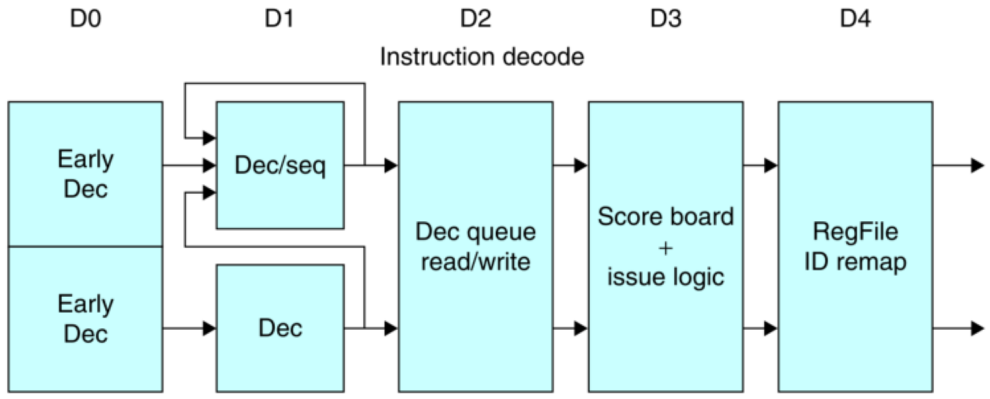

Figure 3.37 The five-stage instruction decode of the A8. In the first stage, a PC produced by the fetch unit (either from the branch target buffer or the PC incrementer) is used to retrieve an 8-byte block from the cache. Up to two instructions are decoded and placed into the decode queue; if neither instruction is a branch, the PC is incremented for the next fetch. Once in the decode queue, the scoreboard logic decides when the instructions can issue. In the issue, the register operands are read; recall that in a simple scoreboard, the operands always come from the registers. The register operands and opcode are sent to the instruction execution portion of the pipeline.

图3.38显示了执行流水线，指令1和指令2都可以进入加载/存储流水线，A8使用一个两发射静态调度超标量

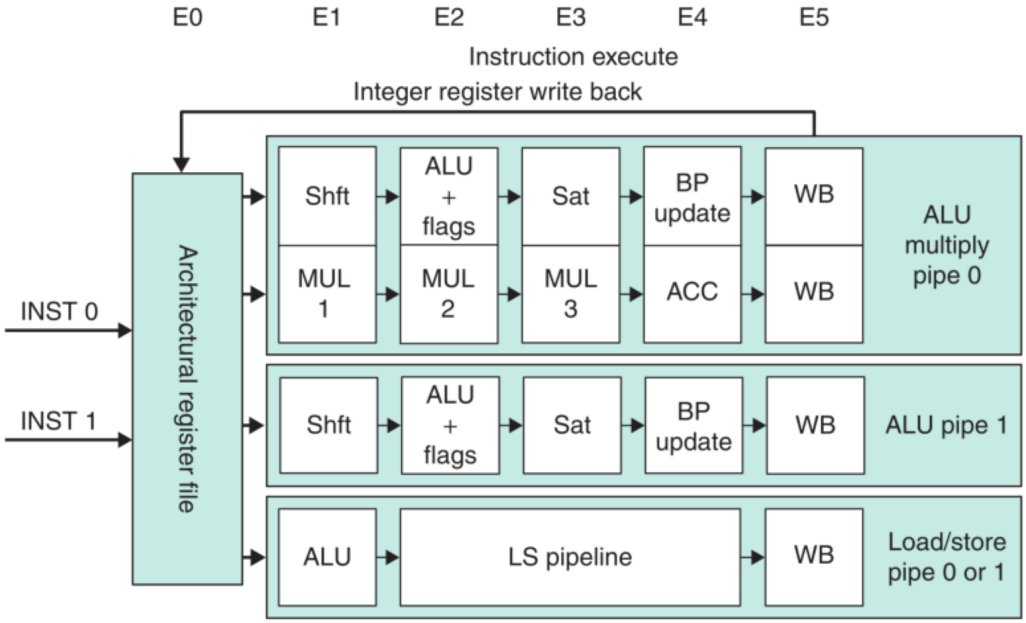

Figure 3.38 The five-stage instruction decode of the A8. Multiply operations are always performed in ALU pipeline 0.

H3 Performance of the A8 Pipeline

由于A8的的双重发射结构，其理想的CPI为0.5。流水线停滞的原因有

- 功能相关，当相邻的两个指令同时使用了一个功能流水线
- 数据相关，在流水线早期检测得到
- 控制相关，当分支预测错误时出现

H2 The Interl Core i7

i7使用了无序推测微体系结构，具有很深的流水线，目的是结合多发射和高时钟率来实现高指令吞吐量。

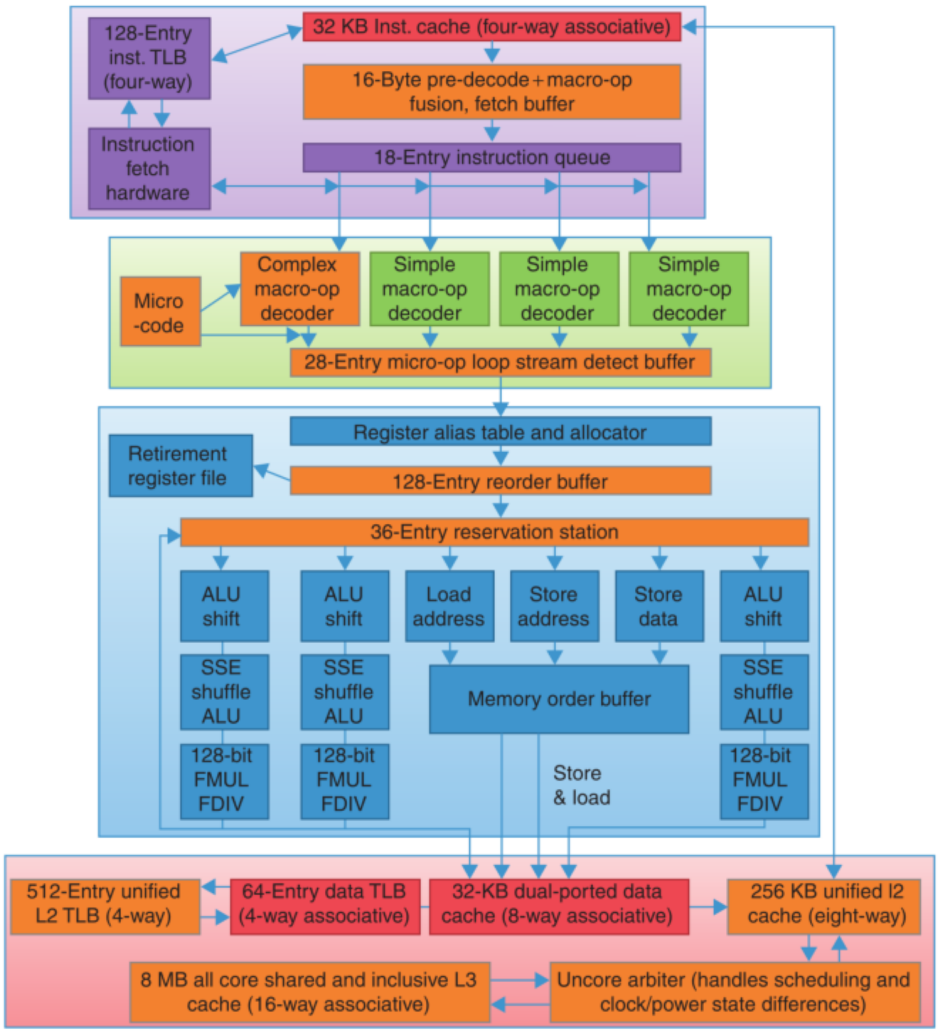

Figure 3.41 The Intel Core i7 pipeline structure shown with the memory system components. The total pipeline depth is 14 stages, with branch mispredictions costing 17 cycles. There are 48 load and 32 store buffers. The six independent functional units can each begin execution of a ready micro-op in the same cycle.

- 指令获取--处理器使用一个多级分支目标缓冲去实现速度和预测精度的平衡。使用一个返回地址堆栈加速函数返回
- 16个字节被放在一个预编码指令缓冲区中，执行一个宏操作融合的过程。
- 微操作解码--单个x86指令被转换成微操作（简单的MIPS类指令，可通过流水线直接执行）
- 微操作缓冲区循环流检测和微融合。微融合结合了load/ALU操作和ALU操作/存储等指令对，并放在一个单独的保留站中，增加了缓冲区的使用量
- 执行基本指令发射--在寄存器表中查找寄存器位置，重命名寄存器，分配重排序缓冲区，并将微操作发送到保留站之前从寄存器或重排序缓冲区获取结果
- i7使用一个36入口的中央预定站，由6个功能单元共享。
- 微操作由各个功能单元执行，然后将结果发送回任何等待保留站以及寄存器单元。
- 当重排序缓冲区头部的一条或多条指令被标记为完成时，在寄存器

H3 Performance of the i7

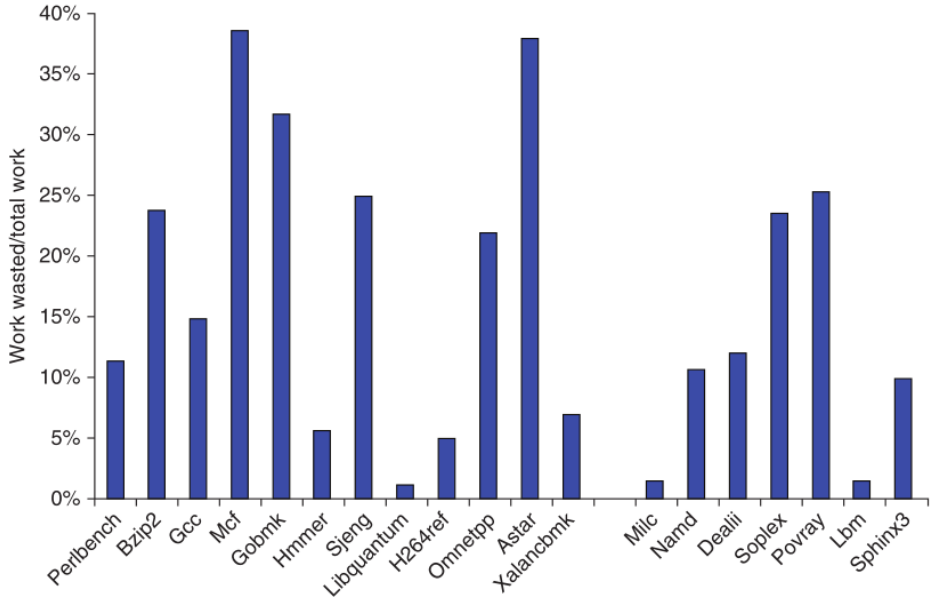

Figure 3.42 The amount of "wasted work" is plotted by taking the ratio of dispatched micro-ops that do not graduate to all dispatched micro-ops. For example, the ratio is 25% for sjeng, meaning that 25% of the dispatched and executed micro-ops are thrown away. The data in this section were collected by Professor Lu Peng and Ph.D. student Ying Zhang, both of Louisiana State University.

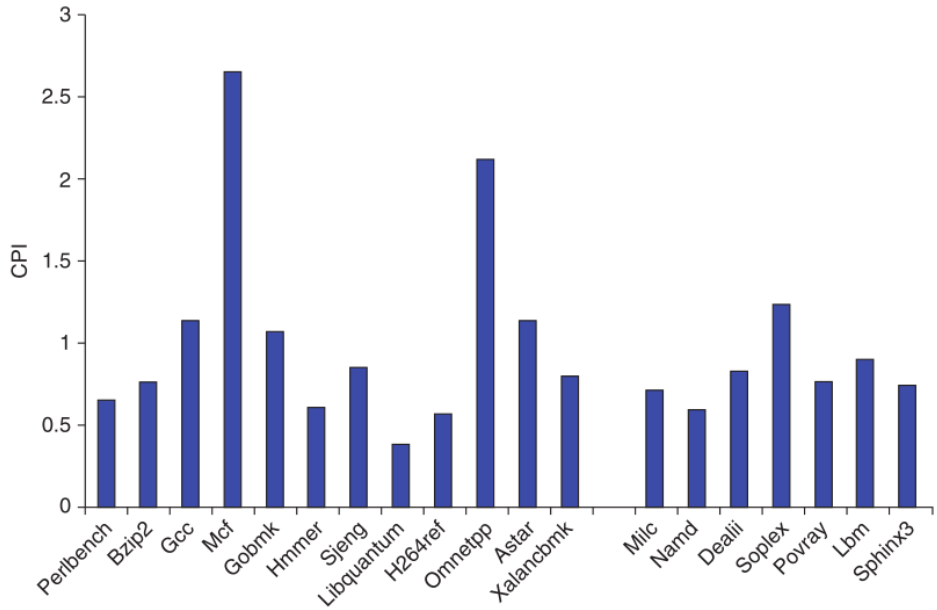

Figure 3.43 The CPI for the 19 SPECCPU2006 benchmarks shows an average CPI for 0.83 for both the FP and integer benchmarks, although the behavior is quite different. In the integer case, the CPI values range from 0.44 to 2.66 with a standard deviation of 0.77, while the variation in the FP case is from 0.62 to 1.38 with a standard deviation of 0.25. The data in this section were collected by Professor Lu Peng and Ph.D. student Ying Zhang, both of Louisiana State University.

虽然i7的动态调度和非阻塞功能可以隐藏一些错过延迟，但缓存内存行为依然是主要原因。