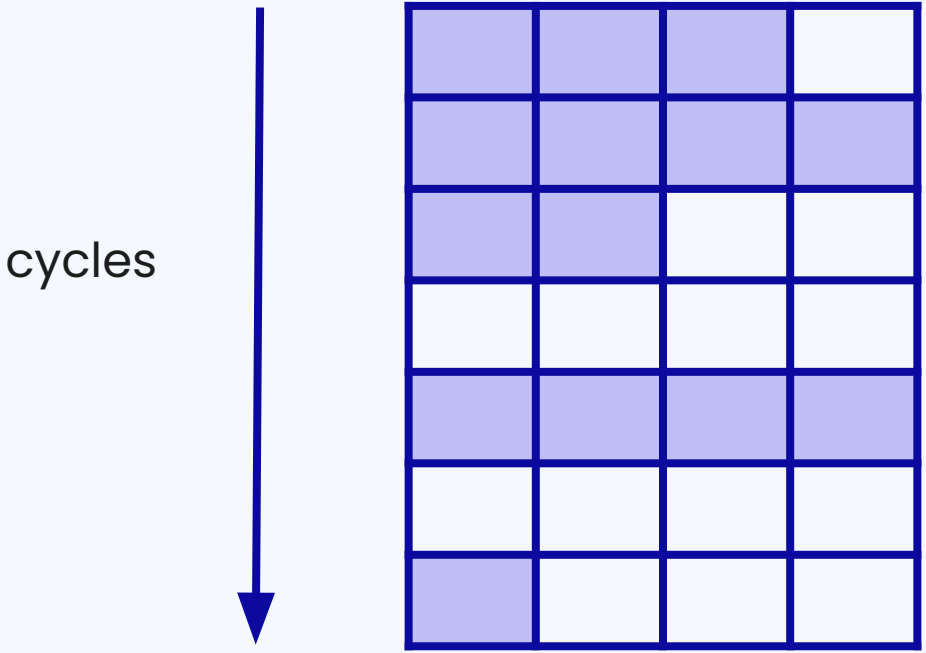
(Uniprocessor) Thread-level parallelism

ILP can hide a lot of latency:

- Multi-cycle instructions
- Data hazards
- Control hazards

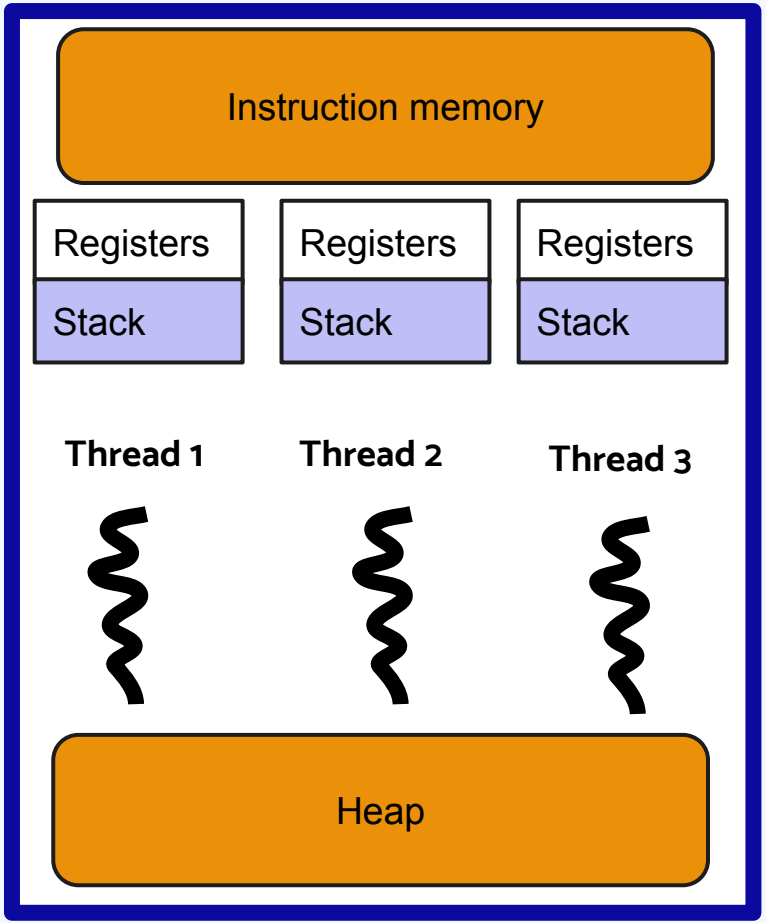
What latency *can't* ILP hide?

Superscalar waste



Threads

Individual imperative programs that run concurrently and share an address space



Uniprocessor multithreading

Allow multiple threads to share CPU resources (FUs)

Contrast with *general TLP (thread-level parallelism)* in multi-processor systems (each processor provides separate resources)

Usually multithreaded per processor as well

NOT instruction-level parallelism!!! (but can play well with ILP techniques)

What do we need to add (or replicate) in the hardware to support multithreading?

RISC-V harts🧡

The base RISC-V ISA supports multiple concurrent threads of execution within a single user address space. Each RISC-V hardware thread, or *hart*, has its own user register state and program counter, and executes an independent sequential instruction stream. The execution environment will define how RISC-V harts are created and managed. RISC-V harts can communicate and synchronize with other harts either via calls to the execution environment, which are documented separately in the specification for each execution environment, or directly via the shared memory system. RISC-V harts can also interact with I/O devices, and indirectly with each other, via loads and stores to portions of the address space assigned to I/O.

Granularity of multithreading

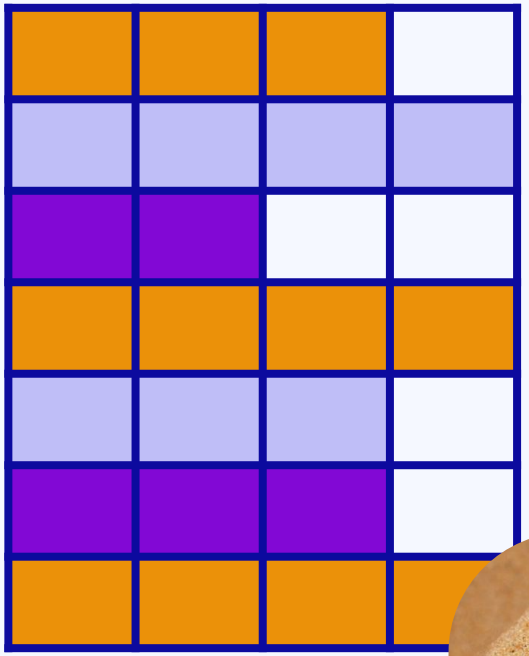
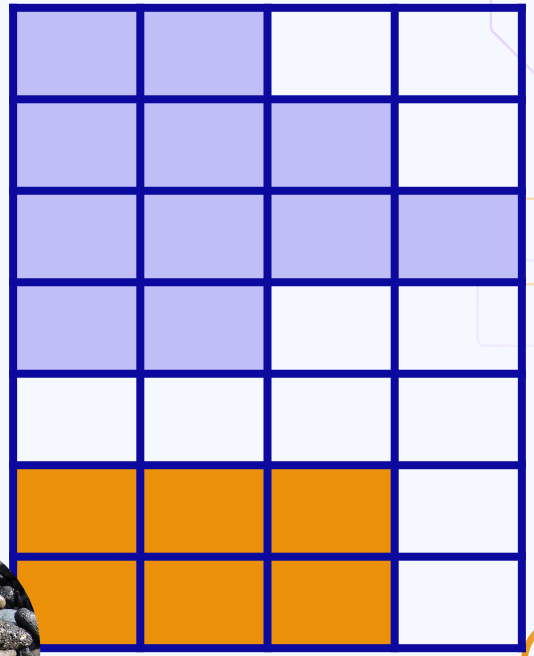


image source

image source



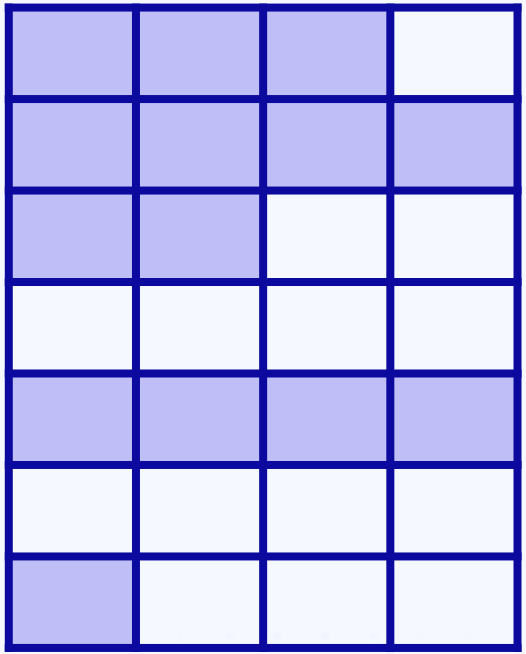
Fine vs. coarse grain

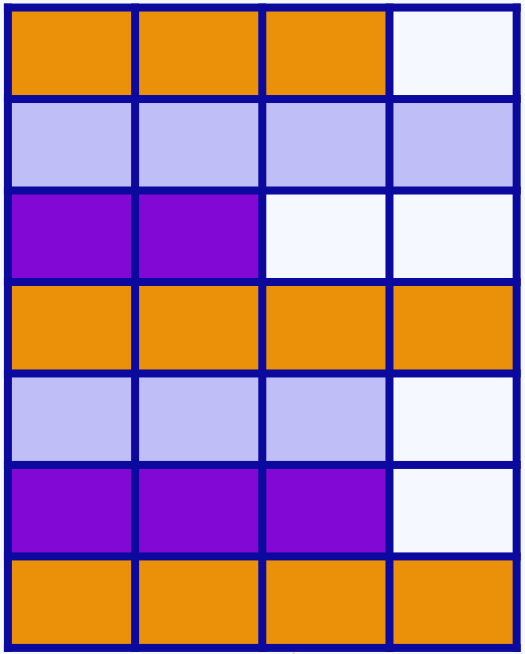
Coarse: waste a cycle on “context switching”

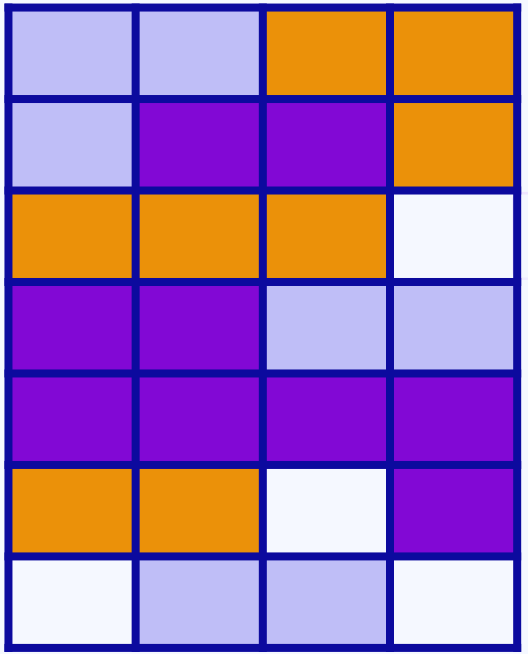
Coarse: feels like there would be less complexity (doesn’t require ability to schedule a new thread every cycle)

Coarse: might lead to starvation :(

Simultaneous MultiThreading







More info on SMT

Each thread has own PC, own ROB

Execute is agnostic to which instruction belongs to which thread

One or more threads issue instructions per cycle on OOO processor

Execution and committing might happen from multiple threads

Intel's name for SMT: "Hyperthreading"

Intel, AMD, ARM implementations allow for two threads

- Not all processors support SMT

How does SMT differ from OS/SW-based scheduling?

- SMT is "simpler" / OS has to save a bunch of info on context switch

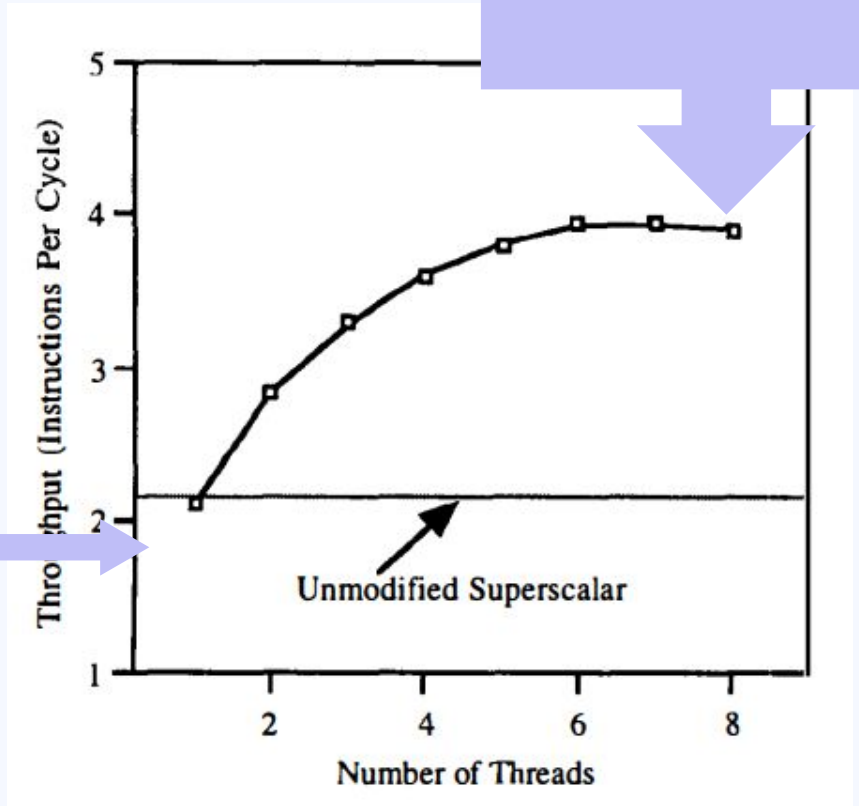
What sorts of workloads wouldn’t work well for SMT?

Diminishing gains

H. M. Levy, Jack L. Lo, J. S. Emer, R. L. Stamm, S. J. Eggers and D. M. Tullsen, "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor," 23rd Annual International Symposium on Computer Architecture (ISCA'96), Philadelphia, PA, USA, 1996, pp. 191-191, link

What leads to this drop?

Why is single thread SMT slightly worse than superscalar without SMT?



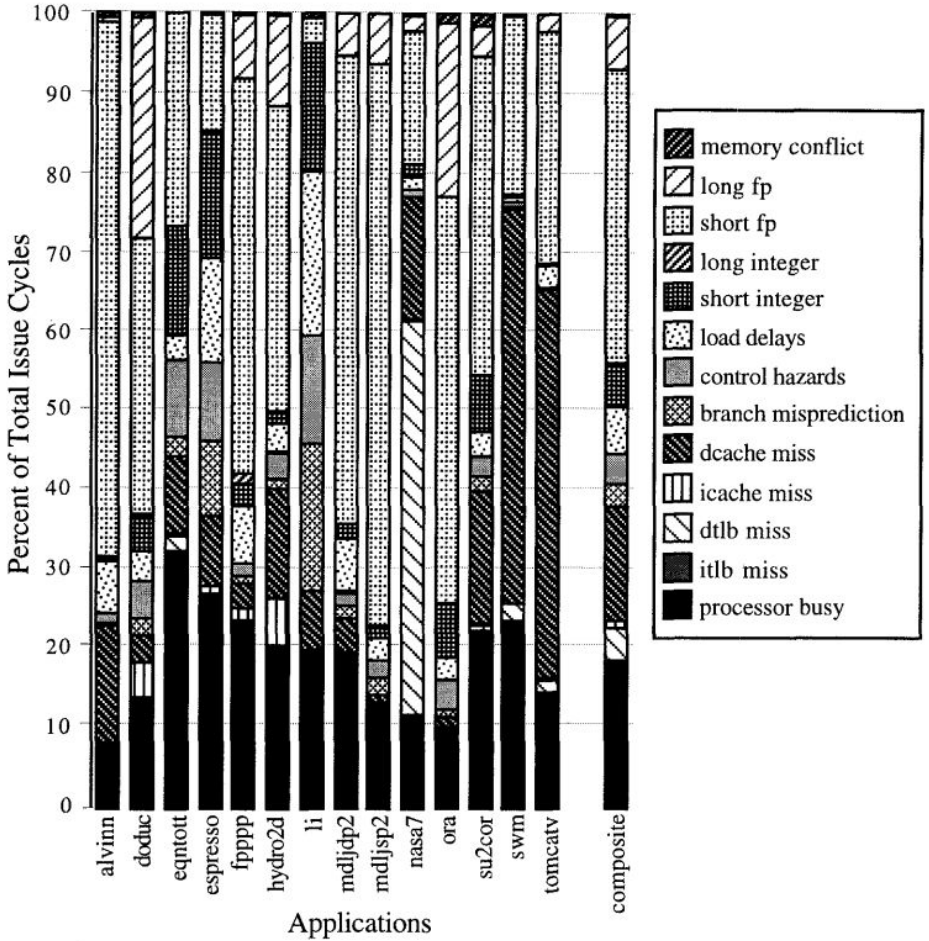


Figure 2: **Sources of all unused issue cycles in an 8-issue superscalar processor.** ***Processor busy*** **represents the utilized issue slots; all others represent wasted issue slots.**

Dean M. Tullsen, Susan J. Eggers, and Henry M. Levy. 1995. Simultaneous multithreading: maximizing on-chip parallelism. In Proceedings of the 22nd annual international symposium on Computer architecture (ISCA '95). Association for Computing Machinery, New York, NY, USA, 392–403. link

How do we choose which thread to fetch instructions for in each cycle?

Fetch policies

ICOUNT: fetches from thread with the least number of instructions in decode/rename /instruction queue

Why is this good for throughput?

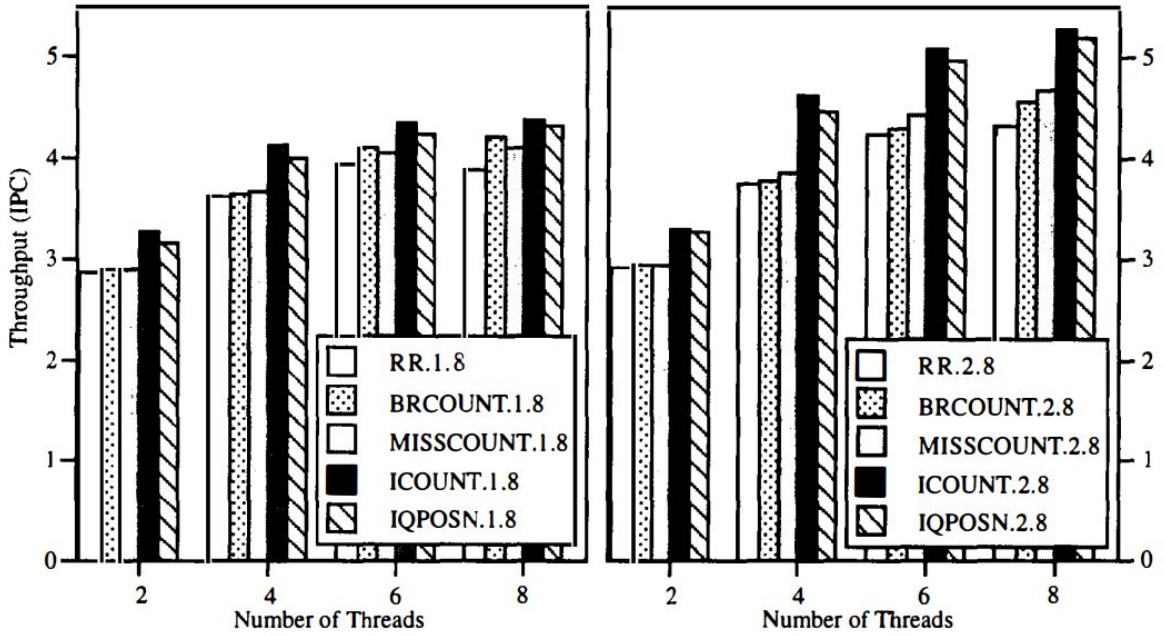
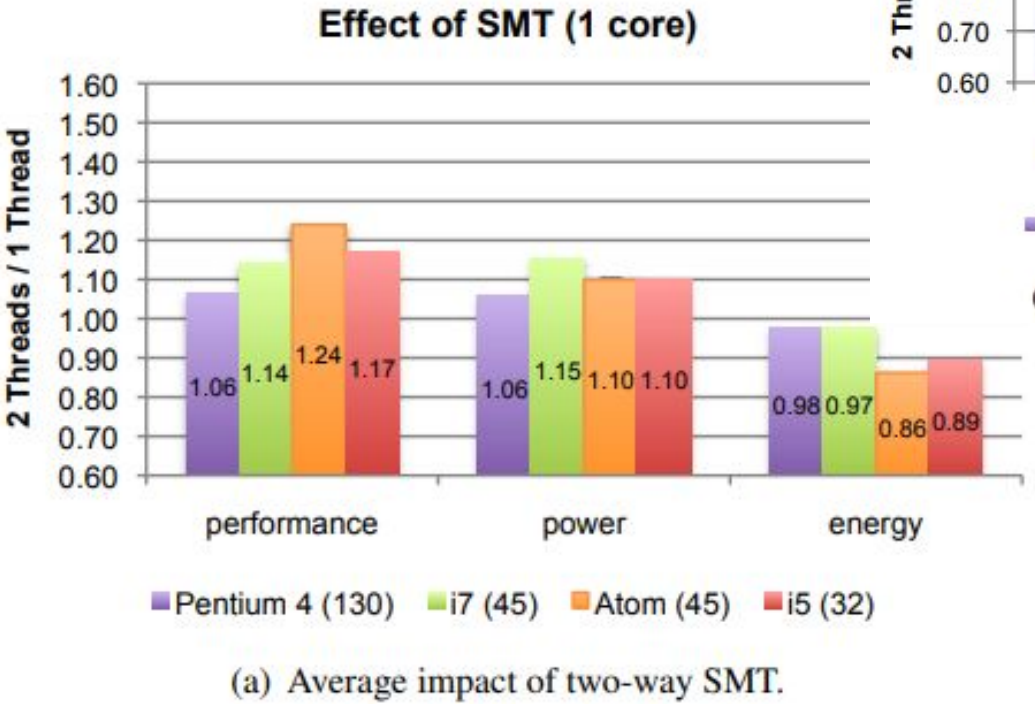
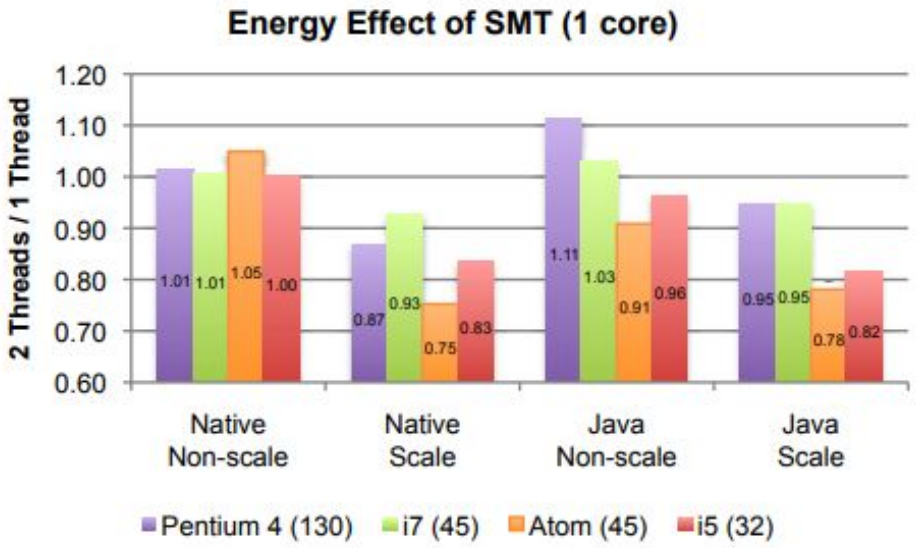


Figure 5: **Instruction throughput for fetching based on several priority heuristics, all compared to the baseline round-robin scheme. The results for 1 thread are the same for all schemes, and thus not shown.**

Power, Energy



(a) Average impact of two-way SMT.



(b) Workload energy impact of two-way SMT.

Hadi Esmaeilzadeh, Ting Cao, Yang Xi, Stephen M. Blackburn, and Kathryn S. McKinley. 2011. Looking back on the language and hardware revolutions: measured power, performance, and scaling. SIGARCH Comput. Archit. News 39, 1 (March 2011), 319–332. link

Whether an application will benefit from SMT is not obvious – what effect does this have on the programmer? What could be done about this?