CS252
Graduate Computer Architecture
Lecture 12

Multithreading / Vector Processing
March 2nd, 2011

John Kubiatowicz
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www.eecs.berkeley.edu/~kubitron/cs252

Performance beyond single thread ILP

- There can be much higher natural parallelism in some applications
  - e.g., Database or Scientific codes
  - Explicit Thread Level Parallelism or Data Level Parallelism
- Thread: instruction stream with own PC and data
  - thread may be a process part of a parallel program of multiple processes, or it may be an independent program
  - Each thread has all the state (instructions, data, PC, register state, and so on) necessary to allow it to execute
- Thread Level Parallelism (TLP):
  - Exploit the parallelism inherent between threads to improve performance
- Data Level Parallelism (DLP):
  - Perform identical operations on data, and lots of data

One approach to exploiting threads: Multithreading (TLP within processor)

- Multithreading: multiple threads to share the functional units of 1 processor via overlapping
  - processor must duplicate independent state of each thread e.g., a separate copy of register file, a separate PC, and for running independent programs, a separate page table
  - memory shared through the virtual memory mechanisms, which already support multiple processes
  - HW for fast thread switch; much faster than full process switch ≈ 100s to 1000s of clocks
- When switch?
  - Alternate instruction per thread (fine grain)
  - When a thread is stalled, perhaps for a cache miss, another thread can be executed (coarse grain)

Fine-Grained Multithreading

- Switches between threads on each instruction, causing the execution of multiples threads to be interleaved
  - Usually done in a round-robin fashion, skipping any stalled threads
  - CPU must be able to switch threads every clock
- Advantage:
  - can hide both short and long stalls, since instructions from other threads executed when one thread stalls
- Disadvantage:
  - slows down execution of individual threads, since a thread ready to execute without stalls will be delayed by instructions from other threads
- Used on Sun's Niagra (recent), several research multiprocessors, Tera

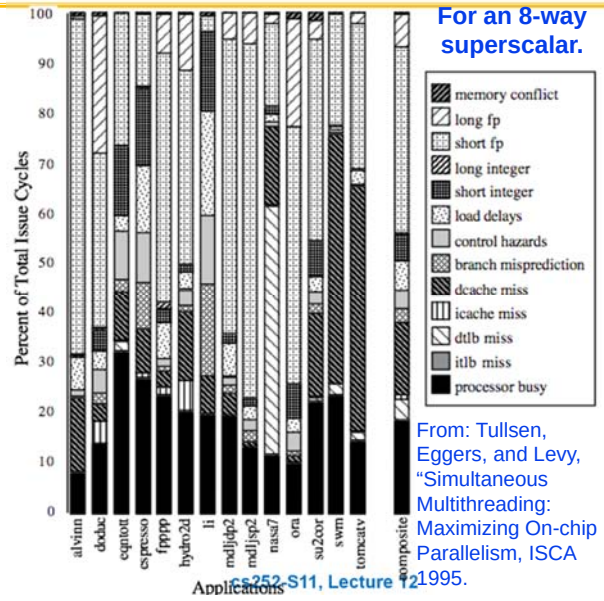
Course-Grained Multithreading

- Switches threads only on costly stalls, such as L2 cache misses
- Advantages
  - Relieves need to have very fast thread-switching
  - Doesn't slow down thread, since instructions from other threads issued only when the thread encounters a costly stall
- Disadvantage is hard to overcome throughput losses from shorter stalls, due to pipeline start-up costs
  - Since CPU issues instructions from 1 thread, when a stall occurs, the pipeline must be emptied or frozen
  - New thread must fill pipeline before instructions can complete
- Because of this start-up overhead, coarse-grained multithreading is better for reducing penalty of high cost stalls, where pipeline refill << stall time
- Used in IBM AS/400, Sparcle (for Alewife)

Simultaneous Multithreading (SMT): Do both ILP and TLP

- TLP and ILP exploit two different kinds of parallel structure in a program
- Could a processor oriented at ILP to exploit TLP?
  - functional units are often idle in data path designed for ILP because of either stalls or dependences in the code
- Could the TLP be used as a source of independent instructions that might keep the processor busy during stalls?
- Could TLP be used to employ the functional units that would otherwise lie idle when insufficient ILP exists?

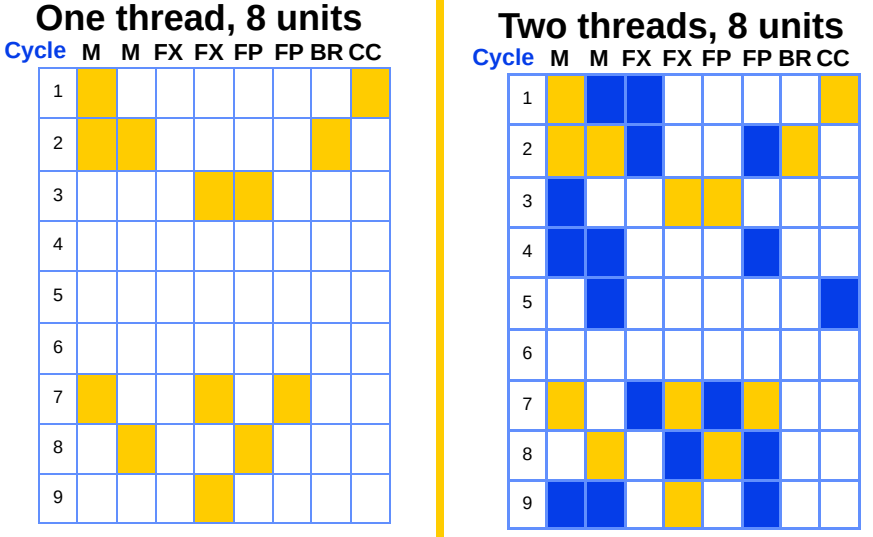
Justification: For most apps, most execution units lie idle



For an 8-way superscalar.

From: Tullsen, Eggers, and Levy, "Simultaneous Multithreading: Maximizing On-chip Parallelism, ISCA 1995.

Simultaneous Multi-threading ...



M = Load/Store, FX = Fixed Point, FP = Floating Point, BR = Branch, CC = Condition Codes

Simultaneous Multithreading Details

- Simultaneous multithreading (SMT): insight that dynamically scheduled processor already has many HW mechanisms to support multithreading
  - Large set of virtual registers that can be used to hold the register sets of independent threads
  - Register renaming provides unique register identifiers, so instructions from multiple threads can be mixed in datapath without confusing sources and destinations across threads
  - Out-of-order completion allows the threads to execute out of order, and get better utilization of the HW
- Just adding a per thread renaming table and keeping separate PCs
  - Independent commitment can be supported by logically keeping a separate reorder buffer for each thread

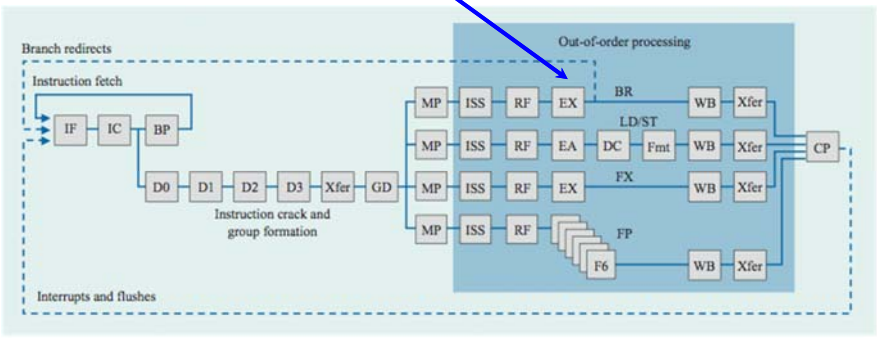
Source: Micrprocessor Report, December 6, 1999
"Compaq Chooses SMT for Alpha"

Design Challenges in SMT

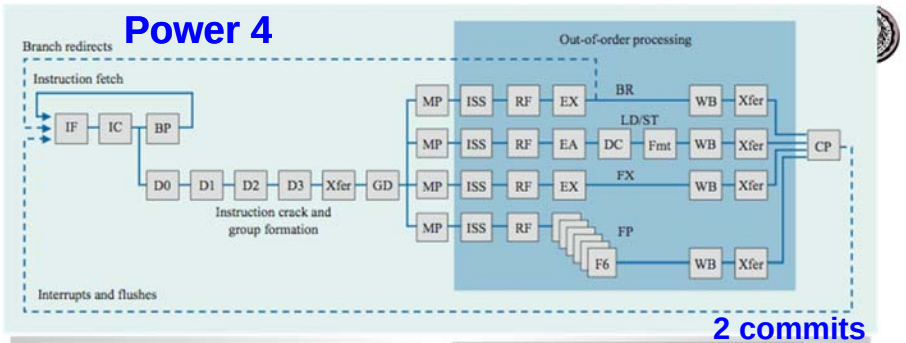
- Since SMT makes sense only with fine-grained implementation, impact of fine-grained scheduling on single thread performance?
  - A preferred thread approach sacrifices neither throughput nor single-thread performance?
  - Unfortunately, with a preferred thread, the processor is likely to sacrifice some throughput, when preferred thread stalls
- Larger register file needed to hold multiple contexts
- Clock cycle time, especially in:
  - Instruction issue - more candidate instructions need to be considered
  - Instruction completion - choosing which instructions to commit may be challenging
- Ensuring that cache and TLB conflicts generated by SMT do not degrade performance

Power 4

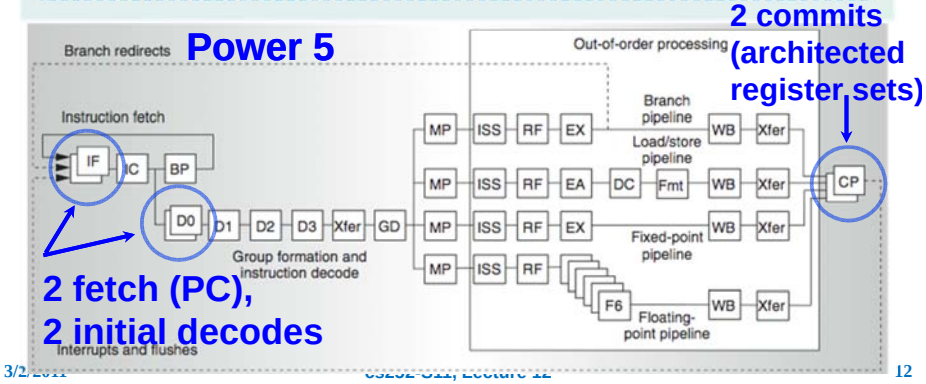
Single-threaded predecessor to Power 5. 8 execution units in out-of-order engine, each may issue an instruction each cycle.



Power 4



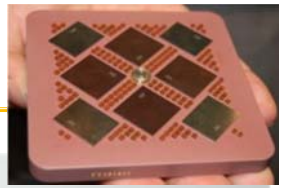
Power 5

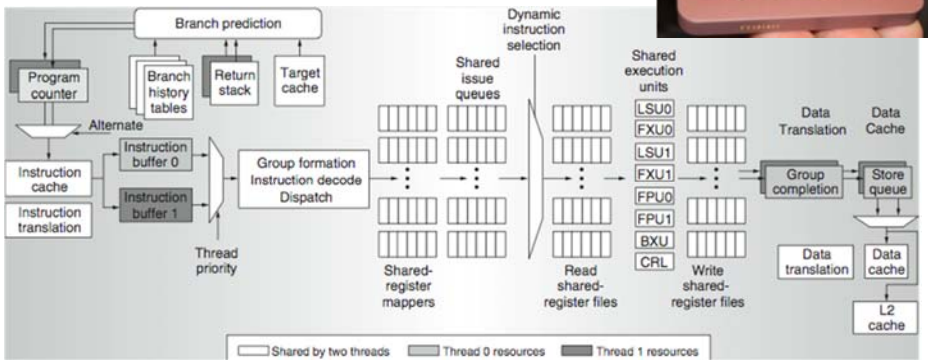


2 commits (architected register sets)

2 fetch (PC), 2 initial decodes

Power 5 data flow ...



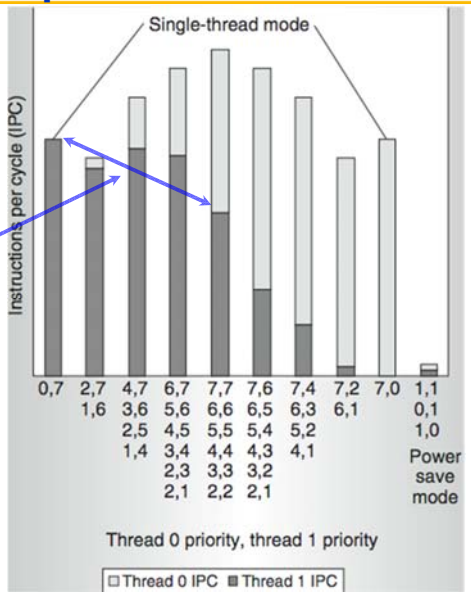


Why only 2 threads? With 4, one of the shared resources (physical registers, cache, memory bandwidth) would be prone to bottleneck

Power 5 thread performance ...

Relative priority of each thread controllable in hardware.

For balanced operation, both threads run slower than if they "owned" the machine.



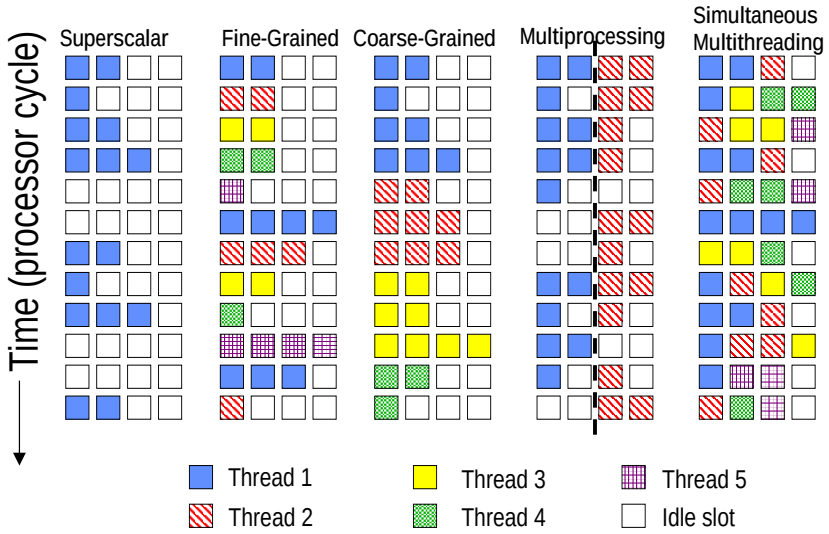
Changes in Power 5 to support SMT

- Increased associativity of L1 instruction cache and the instruction address translation buffers
- Added per thread load and store queues
- Increased size of the L2 (1.92 vs. 1.44 MB) and L3 caches
- Added separate instruction prefetch and buffering per thread
- Increased the number of virtual registers from 152 to 240
- Increased the size of several issue queues
- The Power5 core is about 24% larger than the Power4 core because of the addition of SMT support

Initial Performance of SMT

- Pentium 4 Extreme SMT yields 1.01 speedup for SPECint_rate benchmark and 1.07 for SPECfp_rate
  - Pentium 4 is dual threaded SMT
  - SPECRate requires that each SPEC benchmark be run against a vendor-selected number of copies of the same benchmark
- Running on Pentium 4 each of 26 SPEC benchmarks paired with every other (26^2 runs) speed-ups from 0.90 to 1.58; average was 1.20
- Power 5, 8 processor server 1.23 faster for SPECint_rate with SMT, 1.16 faster for SPECfp_rate
- Power 5 running 2 copies of each app speedup between 0.89 and 1.41
  - Most gained some
  - Fl.Pt. apps had most cache conflicts and least gains

Multithreaded Categories



Administrivia

- **Exam: Wednesday 3/30**
  Location: 320 Soda
  TIME: 2:30-5:30
  - This info is on the Lecture page (has been)
  - Get on 8 ½ by 11 sheet of notes (both sides)
  - Meet at LaVal's afterwards for Pizza and Beverages
- **CS252 First Project proposal due by Friday 3/4**
  - Need two people/project (although can justify three for right project)
  - Complete Research project in 9 weeks
    - » Typically investigate hypothesis by building an artifact and measuring it against a "base case"
    - » Generate conference-length paper/give oral presentation
    - » Often, can lead to an actual publication.

Discussion of SPARCLE paper

- Example of close coupling between processor and memory controller (CMMU)
  - All of features mentioned in this paper implemented by combination of processor and memory controller
  - Some functions implemented as special "coprocessor" instructions
  - Others implemented as "Tagged" loads/stores/swaps
- Course Grained Multithreading
  - Using SPARC register windows
  - Automatic synchronous trap on cache miss
  - Fast handling of all other traps/interrupts (great for message interface!)
  - Multithreading half in hardware/half software (hence 14 cycles)
- Fine Grained Synchronization
  - Full-Empty bit/32 bit word (effectively 33 bits)
    - » Groups of 4 words/cache line ⇒ F/E bits put into memory TAG
  - Fast TRAP on bad condition
  - Multiple instructions. Examples:
    - » LDT (load/trap if empty)
    - » LDET (load/set empty/trap if empty)
    - » STF (Store/set full)
    - » STFT (store/set full/trap if full)

Discussion of Papers: Sparcle (Con't)

- Message Interface
  - Closely couple with processor
    - » Interface at speed of first-level cache
  - Atomic message launch:
    - » Describe message (including DMA ops) with simple stio insts
    - » Atomic launch instruction (ipilaunch)
  - Message Reception
    - » Possible interrupt on message receive: use fast context switch
    - » Examine message with simple ldio instructions
    - » Discard in pieces, possibly with DMA
    - » Free message (ipicst, i.e "coherent storeback")
- We will talk about message interface in greater detail

Supercomputers

Definition of a supercomputer:

- Fastest machine in world at given task
- A device to turn a compute-bound problem into an I/O bound problem
- Any machine costing $30M+
- Any machine designed by Seymour Cray

CDC6600 (Cray, 1964) regarded as first supercomputer

Vector Supercomputers

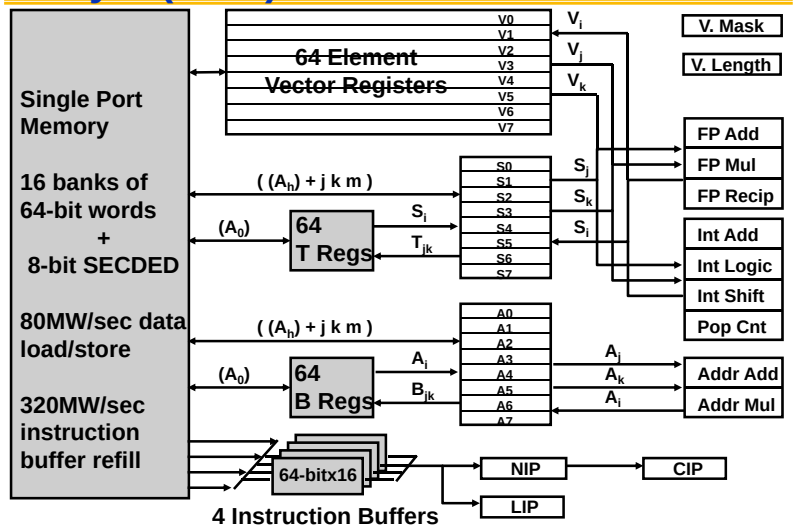
Epitomized by Cray-1, 1976:

Scalar Unit + Vector Extensions

- Load/Store Architecture
- Vector Registers
- Vector Instructions
- Hardwired Control
- Highly Pipelined Functional Units
- Interleaved Memory System
- No Data Caches
- No Virtual Memory

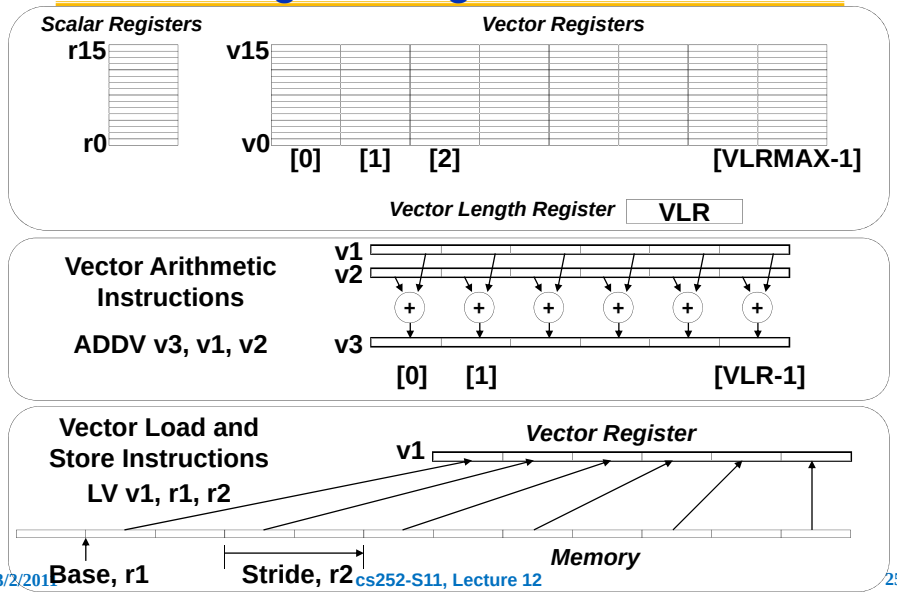
Cray-1 (1976)

Cray-1 (1976)



memory bank cycle 50 ns *processor cycle* 12.5 ns (80MHz)

Vector Programming Model



Multithreading and Vector Summary

- Explicitly parallel (Data level parallelism or Thread level parallelism) is next step to performance
- Coarse grain vs. Fine grained multihreading
  - Only on big stall vs. every clock cycle
- Simultaneous Multithreading if fine grained multithreading based on OOO superscalar microarchitecture
  - Instead of replicating registers, reuse rename registers
- Vector is alternative model for exploiting ILP
  - If code is vectorizable, then simpler hardware, more energy efficient, and better real-time model than Out-of-order machines
  - Design issues include number of lanes, number of functional units, number of vector registers, length of vector registers, exception handling, conditional operations
- Fundamental design issue is memory bandwidth
  - With virtual address translation and caching