EE 457 Unit 9c

Thread Level Parallelism

Credits

- Some of the material in this presentation is taken from:
  - Computer Architecture: A Quantitative Approach
    - John Hennessy & David Patterson
- Some of the material in this presentation is derived from course notes and slides from
  - Prof. Michel Dubois (USC)
  - Prof. Murali Annavaram (USC)
  - Prof. David Patterson (UC Berkeley)

BACKGROUND KNOWLEDGE

Power

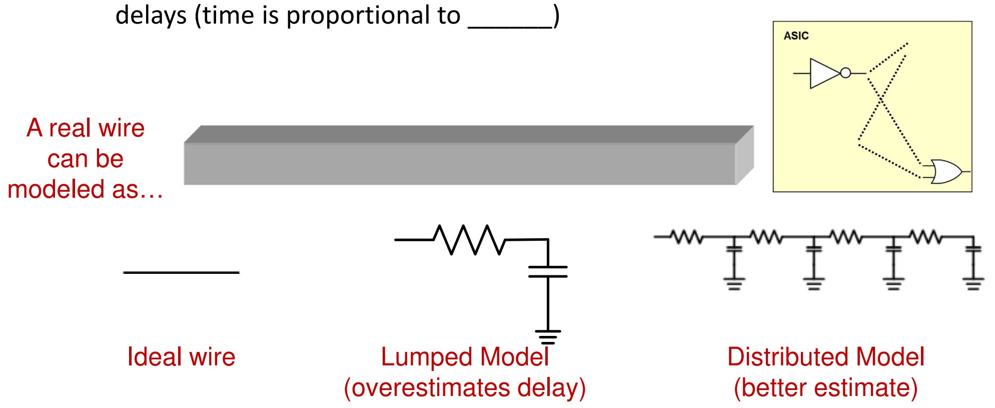
- Power and energy consumption is a MAJOR concern for processors
- Power consumption can be decomposed into:
  - ________ **(P_{STAT})**: Power constantly being dissipated (grows with # of transistors)
  - ____________ **(P_{DYN})**: Power consumed for switching a bit (1 to 0)
- **P_{DYN}** $= I_{DYN} * V_{DD} \approx \frac{1}{2} C_{TOT} V_{DD}^2 f$
  - Recall, $I = C\, dV/dt$
  - V_{DD} is the logic '1' voltage, f = clock frequency
- Dynamic power favors parallel processing vs. higher clock rates
  - V_{DD} value is tied to f, so a reduction/increase in f leads to similar change in Vdd
  - Implies power is proportional to f^3 (a cubic savings in power if we can reduce f)
  - Take a core and replicate it 4x => 4x performance and ____ power
  - Take a core and increase clock rate 4x => 4x performance and ___ power
- Static power
  - Leakage occurs no matter what the frequency is

Temperature

- Temperature is related to power consumption
  - Locations on the chip that burn more power will usually run hotter
    - Locations where bits toggle (register file, etc.) often will become quite hot especially if toggling continues for a long period of time
  - Too much heat can destroy a chip
  - Can use sensors to dynamically sense temperature
- Techniques for controlling temperature
  - External measures: Remove and spread the heat
    - Heat sinks, fans, even liquid cooled machines
  - Architectural measures
    - Throttle performance (run at slower frequencies / lower voltages)
    - Global clock gating (pause..turn off the clock)
    - None…results can be catastrophic

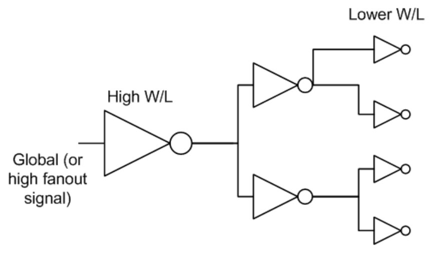
Modeling Interconnect Delay

- In modern circuits wire delay (transmitting the signal) begins to _______________ logic delay (time for gate to switch)
- As wires get longer
  - Resistance goes up and Capacitance goes up causing longer time delays (time is proportional to ______)


A real wire can be modeled as…
ASIC
Ideal wire
Lumped Model (overestimates delay)
Distributed Model (better estimate)

Dealing With Interconnect

- Interconnect delay rivals switching delay
- Important design considerations
  - Long wire traces slow a signal down, thus global signals on a chip require special attention
  - Clock, reset, and other signals must be routed carefully and a whole tree of buffers inserted to decrease the delay

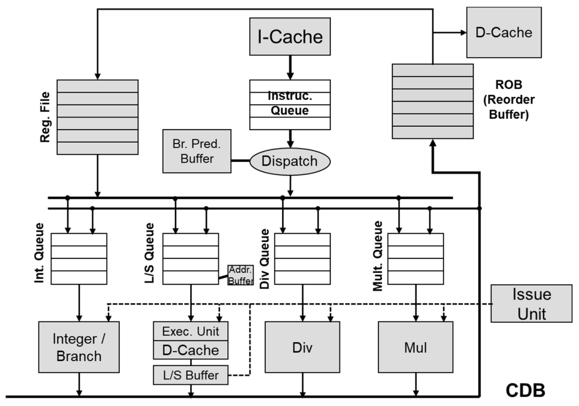


A Case for Thread-Level Parallelism

CHIP MULTITHREADING AND MULTIPROCESSORS

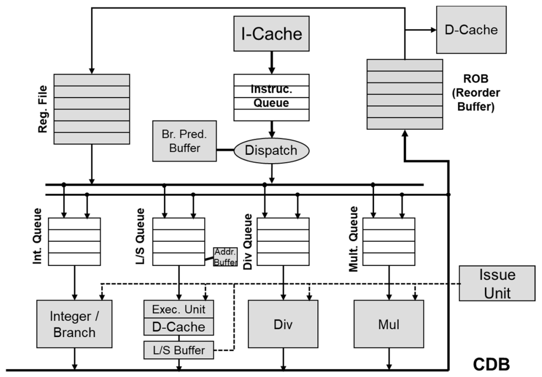
Consideration

- Consider our out-of-order, pipelined processor from Tomasulo part 2.



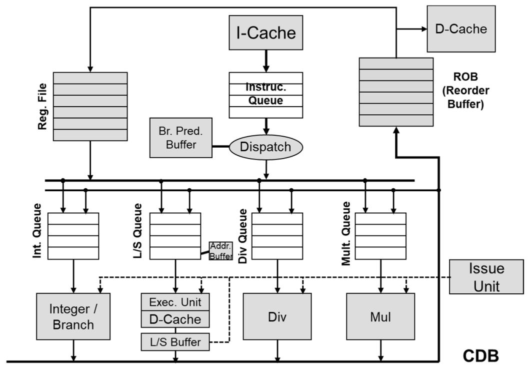
Question 1

- Do we have high frequencies?



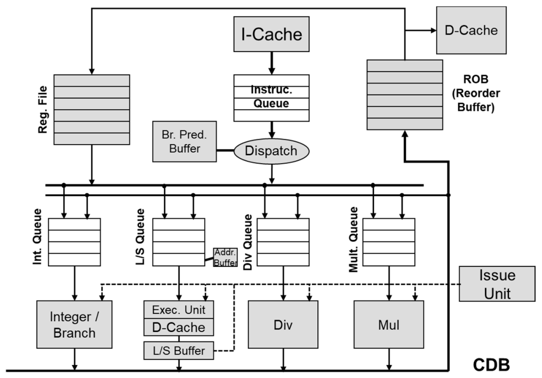
Answer 1

- Do we have high frequencies?
  - Yes, _________ pipelines create shorter clock cycles and higher frequencies
  - Power ↑ ↑ ↑
  - Temp. ↑



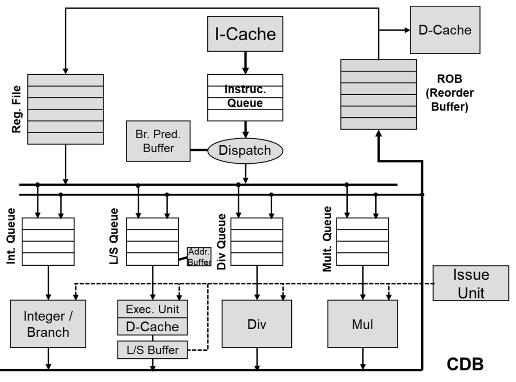
Question 2

- What effect does our OoO processor have on wires length?



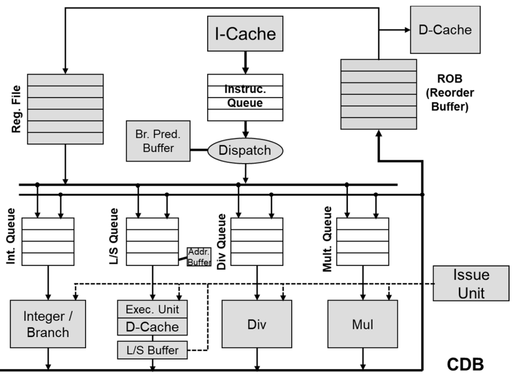
Answer 2

- What effect does our OoO processor have on wires length?
  - Wire length ↑ ↑ with ______, ROB, _______ logic, etc.
  - Time ↑



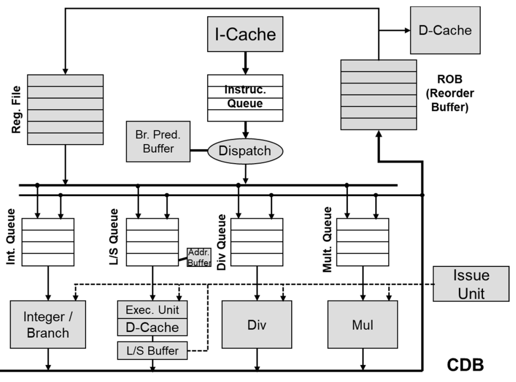
Question 2

- What is the impact of short clock cycles (high freq.) on cache miss penalties?



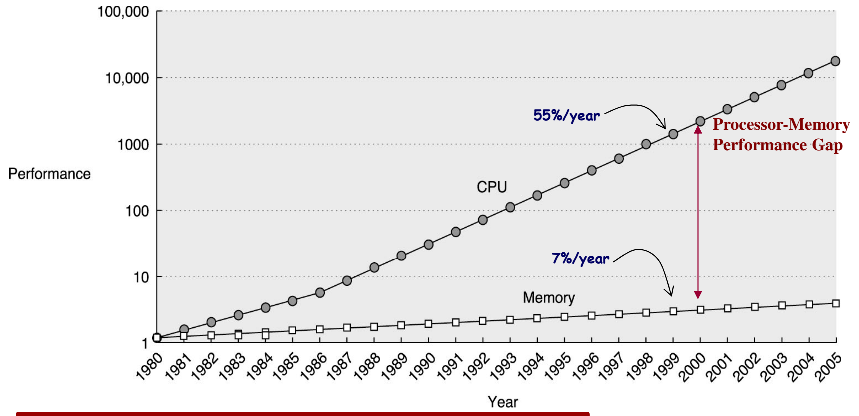
Answer 3

- What is the impact of short clock cycles (high freq.) on cache miss penalties?
  - Cache miss penalties ↑ ↑ relative to processor cycles



Memory Wall Problem

- Processor performance is increasing much faster than memory performance

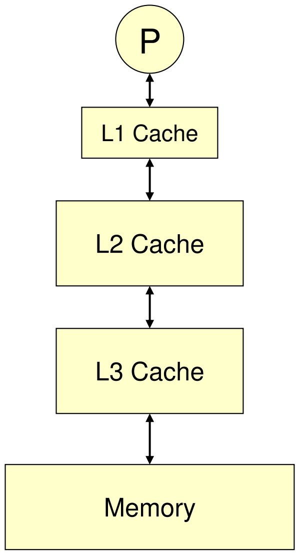


There is a limit to ILP!
If a cache miss requires several hundred clock cyles even OoO pipelines with 10's or 100's of in-flight instructions may stall.

Hennessy and Patterson, *Computer Architecture – A Quantitative Approach* (2003)

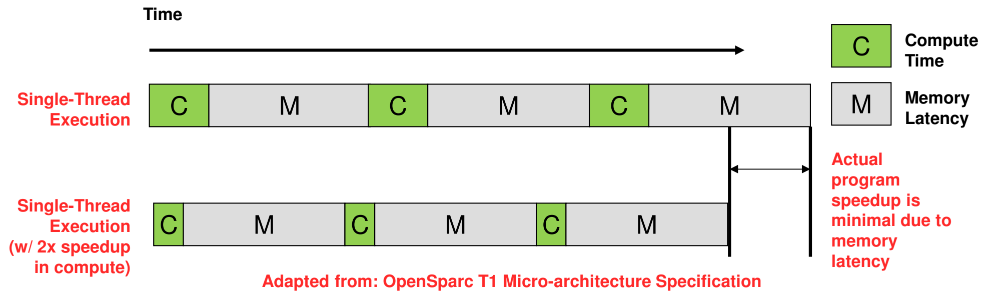
Cache Hierarchy

- A hierarchy of cache can help mitigate the cache miss penalty
- L1 Cache
  - 64 KB
  - 2 cycle access time
  - Common Miss Rate ~ 5%
- L2 Cache
  - 1 MB
  - 10-20 cycle access time
  - Common Miss Rate ~ 1%
- Main Memory
  - ~_____ cycle access time



The Growing Memory Problem

- In an In-Order pipeline, a cache miss causes computation to stall (i.e. a memory induced stall)
  - Suppose we could improve our processor to achieve a 2x speedup in compute time
  - This would only yield a ____________ overall speedup due to memory latency dominating compute
- Out-of-order may do better, but the problem remains


Time
C Compute Time
M Memory Latency
Single-Thread Execution
C M C M C M
Single-Thread Execution (w/ 2x speedup in compute)
C M C M C M
Actual program speedup is minimal due to memory latency

Adapted from: OpenSparc T1 Micro-architecture Specification

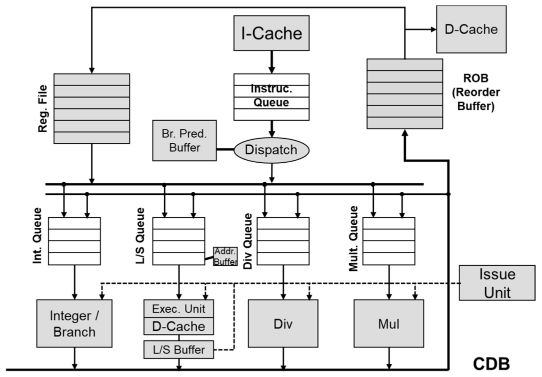
Cache Penalty Example

- Assume 50% of instructions are LW/SW, an L1-D hit rate of 90%, and miss penalty of 20 clock cycles (assuming these misses hit in L2). What is the CPI for our typical 5 stage pipeline?
  - 50% * 10% misses = ___ instructions that cause stalls
  - Other 95 instructions take _____ cycles to execute
  - 5 instructions take _______________ cycles to execute
  - Total 200 cycles for 100 instructions = CPI of 2

Effective CPI = Ideal CPI + Miss Rate*Miss Penalty Cycles

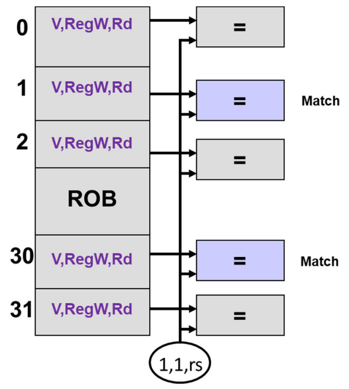
Question 4

- But in OoO processors, can't we just deepen our ROB, Issue queues, Store Address Buffer, etc to hide cache misses?



Answer 4

- But in OoO processors, can't we just deepen our ROB, Issue queues, Store Address Buffer, etc to hide cache misses?
  - Associative ________ structures are expensive and slow down dramatically as they deepen
  - DOES NOT _______ WELL



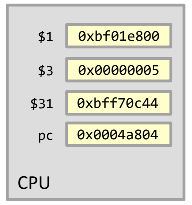
Motivating HW Multithread/Multicore

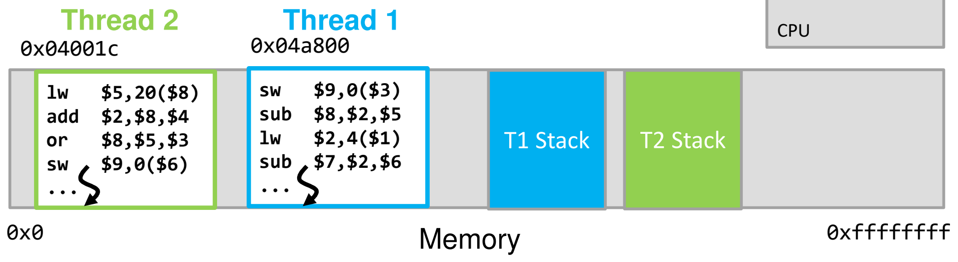
- Issues that prevent us from exploiting ILP in more advanced single-core processors with deeper pipelines and OoO Execution
  - Slow memory hierarchy
  - Increased power with higher clock rates
  - Increased wire delay & size with more advanced structures (ROBs, Issue queues, etc.) for potentially __________ __________
- All of these issues point us to find "easier" sources of parallelism such as: **TLP (Thread-Level Parallelism)**

OVERVIEW OF TLP

What is a Thread?

- **Thread (def.)*:*** Single execution sequence (instruction ________) representing a separately schedulable task
  - Schedulable task: Can be transparently paused and resumed by the OS scheduler
- Consider the processor:
  - For what resources would each thread need their own copy to execute in parallel?





```
lw   $5,20($8)
add  $2,$8,$4
or   $8,$5,$3
sw   $9,0($6)
...
```

```
sw   $9,0($3)
sub  $8,$2,$5
lw   $2,4($1)
sub  $7,$2,$6
...
```

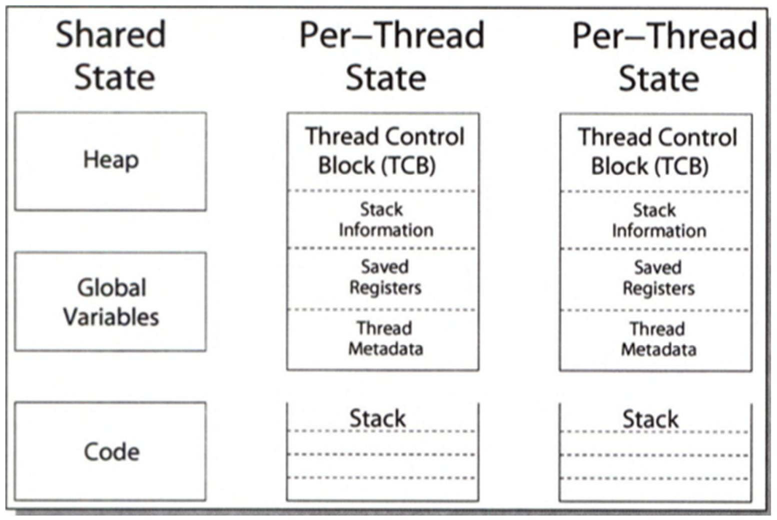
Separate or Shared?

- For what resources would each thread need their own copy to execute in parallel?

1 Per Thread | **Shared among all**

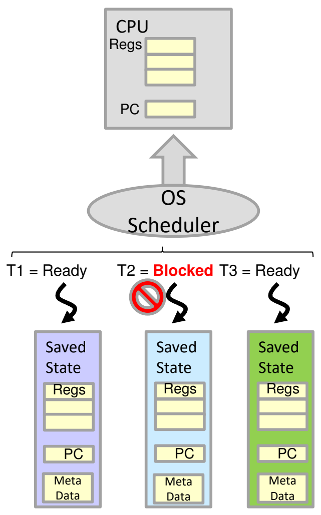
Program Counter
ALUs
Register File
Page Table Base Register
Cache Memory

Shared vs. Private



Software Multithreading

- Used since 1960's on uniprocessors to hide I/O latency
  - Multiple processes with different virtual address spaces and process control blocks
  - On an I/O operation, state is saved and another process is given to the CPU
  - When I/O operation completes the process is rescheduled
- On a context switch…
  - Trap processor and flush pipeline
  - Save state in process control block (____, __________, Interrupt vector, page table base register)
  - Restore state of another process
  - Start execution and fill pipeline
- Context switch is also triggered by _____ for fairness
- **Very high overhead! (___________)**

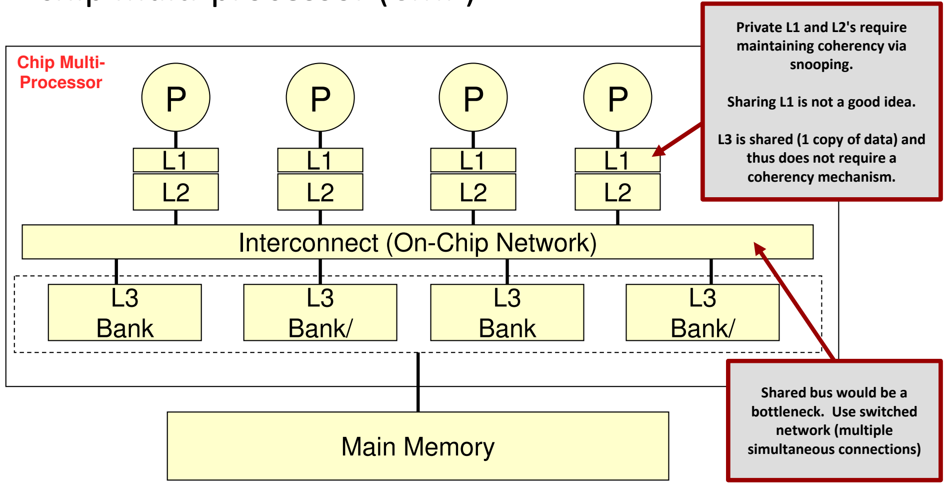


Multicore vs. Multithreaded

- Multicore/Multiprocessor: Single chip containing multiple processor cores that possess all the logic resources necessary to execute one or more threads at a time
  - Require software/OS to context switch from one thread to another and do not share hardware resources between threads.
- Hardware Multithreading: A processor core that has hardware support for executing multiple threads and context switching between them _________ software intervention

Typical Multicore (CMP) Organization

- Can simply replicate entire processor core to create a chip multi-processor (CMP)



Private L1 and L2's require maintaining coherency via snooping.

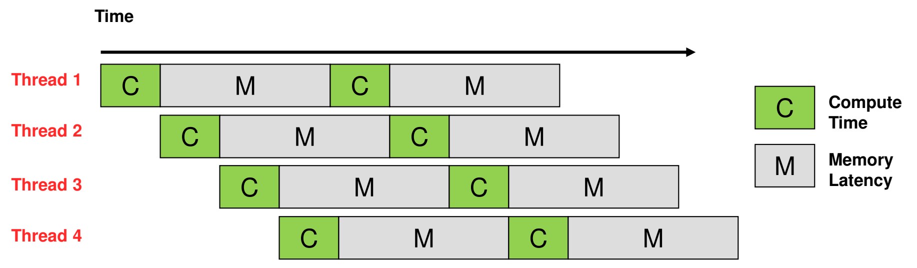
Sharing L1 is not a good idea.

L3 is shared (1 copy of data) and thus does not require a coherency mechanism.

Shared bus would be a bottleneck. Use switched network (multiple simultaneous connections)

Case for Multithreading

- Consider _______________ events:
  - Cache Miss, Exceptions, Lock (Synchronization), Long instructions such as MUL/DIV
  - Such events cause In-order and even OoO pipelines to be underutilized
- **Goal/Idea**: ______ to the next thread immediately (on ____ cycle) when the current thread hits a long-latency event (i.e. cache miss)
  - By executing multiple threads, processors can be kept busy with useful work

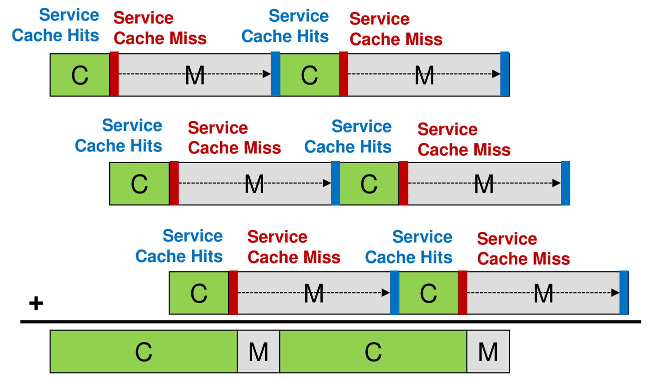


Adapted from: OpenSparc T1 Micro-architecture Specification

IMPLEMENTING HARDWARE MULTITHREADING

MT Needs Non-Blocking Caches

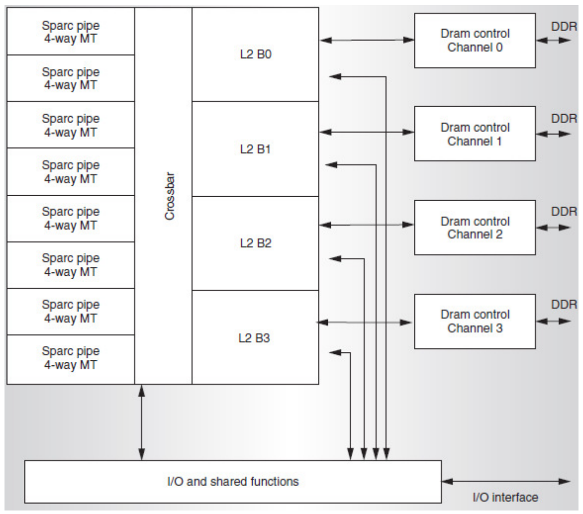
- **Non-blocking cache**: Does not block/pause on a miss but is able to **service** ______ while fetching one or more _______ **requests**
  - Needed to support multithreading
  - **Example**: Pentium Pro has a non-blocking cache capable of handling 4 outstanding misses



Hardware Multithreading

- Run multiple threads on the same core with hardware support for fast context switch
  - Multiple register files
  - Multiple state registers (PCs, page table base registers, interrupt vectors, etc.)
  - Avoids saving context manually (via software)

Sun T1 "Niagara" Case Study



Ex. of Fine-grained Multithreading

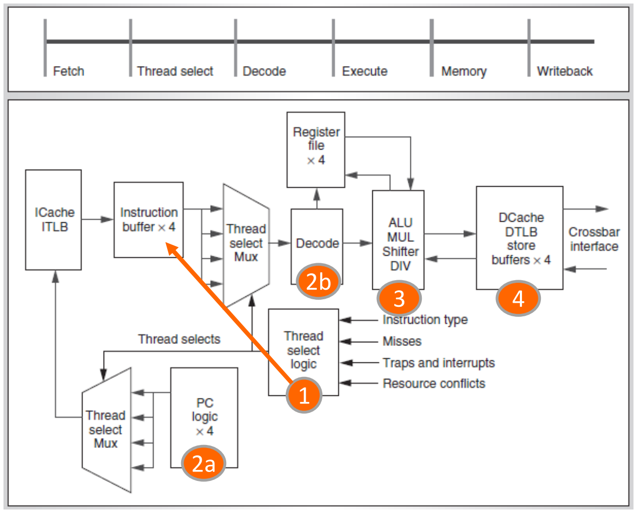
http://ogun.stanford.edu/~kunle/publications/niagra_micro.pdf

Sparc T1 Niagara

- 8 cores each executing 4 threads called a thread group
  - Zero cycle thread switching penalty (round-robin)
  - 6 stage pipeline
- Each core has its own L1 cache
- Each thread has its own
  - Register file, instruction and store buffers
- Threads share…
  - L1 cache, TLB, and execution units
- 3 MB shared L2 Cache, 4-banks, 12-way set-associative
  - Is it a problem that it's not a power of 2? No!

Fetch	(Thread) Select	Decode	Exec.	Mem.	WB

Sun T1 "Niagara" Pipeline



http://ogun.stanford.edu/~kunle/publications/niagra_micro.pdf

T1 Pipeline

- Thread select stage [Stage 2]
  - Choose instructions to issue from ready threads
  - Issues based on
    - Instruction type
    - Misses
    - Resource conflicts
    - Traps and interrupts
- Fetch stage [Stage 1]
  - Thread select mux chooses which thread's instruction to issue and uses that thread's PC to fetch more instructions
  - Access I-TLB and I-Cache
  - 2 instructions fetched per cycle

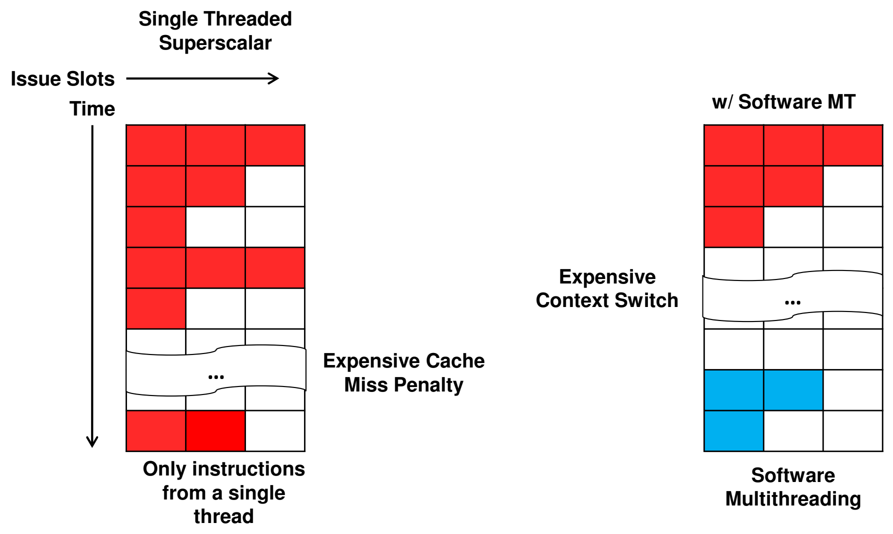
T1 Pipeline

- Decode stage [Stage 3]
  - Accesses register file
- Execute Stage [Stage 4]
  - Includes ALU, shifter, MUL and DIV units
  - Forwarding Unit
- Memory stage [Stage 5]
  - DTLB, Data Cache, and 4 store buffers (1 per thread)
- WB [Stage 6]
  - Write to register file

Pipeline Scheduling

- No pipeline flush on context switch (except potentially of instructions from faulting thread)
- Full forwarding/bypassing to consuming, junior instructions of same thread
- In case of load, wait ___ cycles before an instruction from the same thread is issued
  - Solved ______________________ issue
- Scheduler guarantees fairness between threads by prioritizing the least recently scheduled thread

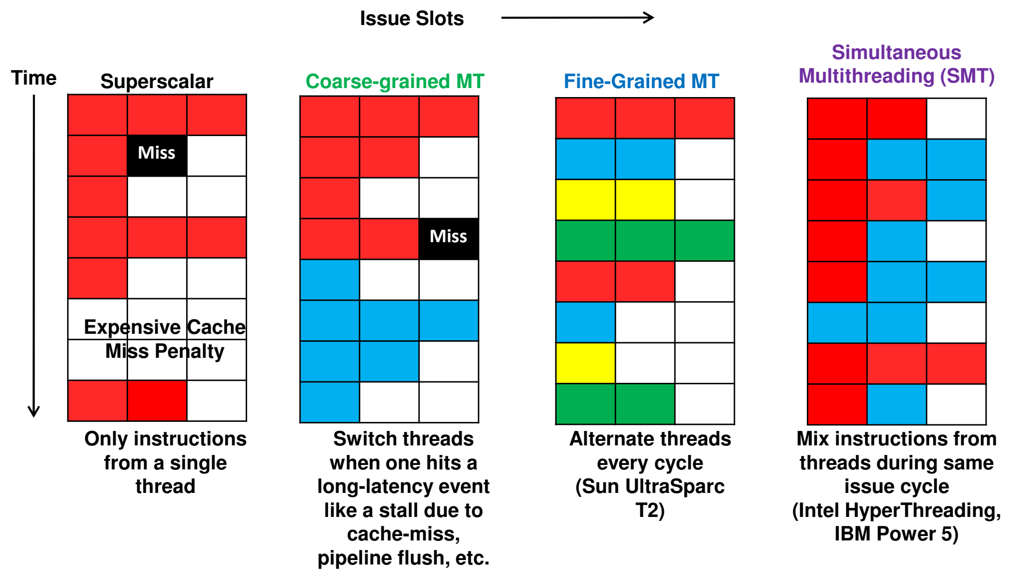
A View Without HW Multithreading


Single Threaded Superscalar
Issue Slots
Time
...
Expensive Cache Miss Penalty
Only instructions from a single thread
w/ Software MT
Expensive Context Switch
...
Software Multithreading

Types/Levels of Multithreading

- How should we overlap and share the HW between instructions from different threads
  - ________-grained Multithreading: Execute one thread with all HW resource until a cache-miss or misprediction will incur a stall or pipeline flush, then switch to another thread
  - ________-grained Multithreading: Alternate fetching instructions from a different thread each clock
  - _______________ Multithreading: Fetch and execute instructions from different threads at the same time

Levels of TLP


Issue Slots
Time
Superscalar
Coarse-grained MT
Fine-Grained MT
Simultaneous Multithreading (SMT)
Miss
Expensive Cache Miss Penalty
Miss
Only instructions from a single thread
Switch threads when one hits a long-latency event like a stall due to cache-miss, pipeline flush, etc.
Alternate threads every cycle (Sun UltraSparc T2)
Mix instructions from threads during same issue cycle (Intel HyperThreading, IBM Power 5)

Fine Grained Multithreading

- Like Sun Niagara
- Alternates issuing instructions from different threads each cycle provided a thread has instructions ready to execute (i.e. not stalled)
- With enough threads, long latency events may be completely hidden
  - Some processors like Cray may have ____ or more threads
- Degrades ________________ performance since it only gets 1 out of every N cycles if all N threads are ready

Coarse Grained Multithreading

- Swaps threads on long-latency event
- Hardware does not have to swap threads in a single cycle (as in fine-grained multithreading) but can take a few cycles since the current thread has hit a long latency event
- Requires flushing pipeline of current thread's instructions and filling pipeline with new thread's
- Better single-thread performance

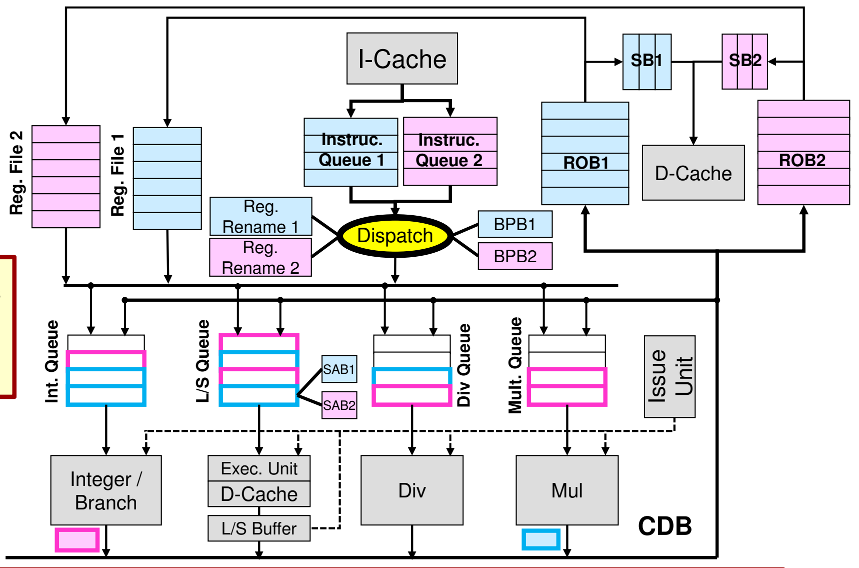
ILP and TLP

- TLP can also help ILP by providing another source of independent instructions
- In a 3- or 4-way issue processor, better utilization can be achieved when instructions from 2 or more threads are executed simultaneously

Simultaneous Multithreading

- Uses multiple-issue, dynamic scheduling mechanisms to execute instructions from multiple threads at the same time by filling issue slots with as many available instructions from either thread
  - Overcome poor utilization due to cache misses or lack of independent instructions
  - Requires HW to ______ instructions based on their thread
- Requires greater level of hardware resources (separate register renamer, branch prediction, store buffers, and multiple register files, etc.)

2-Way SMT Updated Block Diagram



Dispatch can tag instructions with thread ID to separate instructions in the backend

Updated OoO processor block diagram for 2-way hardware, SMT

Example

- Intel HyperThreading Technology (HTT) is essentially SMT
- Recent processors including Core i7 are multi-core, multi-threaded, multi-issue, OoO (dynamically scheduled) superscalar processors

Future of Multicore/Multithreaded

- Multiple cores in shared memory configuration
- Per-core L1 or even L2
- Large on-chip shared cache
- Multiple threads on each core to fight memory wall
- Ever increasing on-chip threads
  - To continue to meet Moore's Law
  - CMP's with 1000's of threads envisioned
  - Only sane option from technology perspective (i.e. out of necessity)
  - The big road block is parallel programming

Parallel Programming

- Implicit parallelism via…
  - Parallelizing compilers
  - Programming frameworks (e.g. MapReduce)
- Explicit parallelism
  - ________________
  - Task Libraries
    - Intel Thread Building Blocks, Java Task Library
  - Native threading (Windows threads, _________ threads)
  - ____________