Each line holds physical \(<\text{src1, src2, dest}>\) registers for an instruction, and controls when it executes.

**Execution engine works on the physical registers, not the architecture registers.**
Recall: Throughput and multiple threads

Goal: Use multiple instruction streams to improve (1) throughput of machines that run many programs (2) execution time of multi-threaded programs.

Example: Sun Niagara (32 instruction streams on a chip).

Difficulties: Gaining full advantage requires rewriting applications, OS, libraries.

Ultimate limiter: Amdahl’s law (application dependent). Memory system performance.
This Time: Throughput Computing

Multithreading: Interleave instructions from separate threads on the same hardware. Seen by OS as several CPUs.

Multi-core: Integrating several processors that (partially) share a memory system on the same chip.

Also: A “town meeting” discussion on lessons learned from Lab 4.
Multi-Threading
Power 4 (predates Power 5 shown Tuesday)

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

Image of Power 4+ chip
For most apps, most execution units lie idle

Observation: Most hardware in an out-of-order CPU concerns physical registers. Could several instruction threads share this hardware?

Simultaneous Multi-threading ...

One thread, 8 units

<table>
<thead>
<tr>
<th>Cycle</th>
<th>M</th>
<th>M</th>
<th>FX</th>
<th>FX</th>
<th>FP</th>
<th>FP</th>
<th>BR</th>
<th>CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Two threads, 8 units

<table>
<thead>
<tr>
<th>Cycle</th>
<th>M</th>
<th>M</th>
<th>FX</th>
<th>FX</th>
<th>FP</th>
<th>FP</th>
<th>BR</th>
<th>CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

M = Load/Store, FX = Fixed Point, FP = Floating Point, BR = Branch, CC = Condition Codes
Administrivia: Big Game -- Go Cal!

Thursday 11/18:
Preliminary design document due, by 9 PM.

Friday 11/19: Review design document with TAs in lab section.

Sunday 11/21: Revised design document due in email, by 11:59 PM

Friday 12/3: Demo deep pipeline in lab section.
Administrivia: Mid-term and Field Trip

Mid-Term II Review

Session: Sunday, 11/21, 7-9 PM, 306 Soda.
(no lecture Tuesday)

Mid-Term II: Tuesday, 11/23, 5:30 to 8:30 PM, 101 Morgan. LaVal’s @ 9 PM!

Xilinx field trip: Tuesday 11/30, bus leaves at 8:30 AM, from 4th floor Soda.

Send Doug RSVP by 5PM today!

Thursday 12/2: Advice on Presentations.
Prepare you for your final project talk.
Multi-Threaded (continued)
Power 4

Branch redirects

Instruction fetch

IF → IC → BP

D0 → D1 → D2 → D3 → Xfer → GD

Instruction crack and group formation

Interrupts and flushes

Out-of-order processing

MP → ISS → RF → EX

MP → ISS → RF → EA → DC → Fmt → WB → Xfer

MP → ISS → RF → EX

MP → ISS → RF

MP → ISS → RF

F6 → WB → Xfer

Power 5

2 commits (architected register sets)

2 fetch (PC), 2 initial decodes

Branch redirects

Instruction fetch

IF → IC → BP

D0 → D1 → D2 → D3 → Xfer → GD

Group formation and instruction decode

Interrupts and flushes

Out-of-order processing

MP → ISS → RF → EX

MP → ISS → RF → EA → DC → Fmt → WB → Xfer

MP → ISS → RF → EX

MP → ISS → RF

MP → ISS → RF

F6 → WB → Xfer

Fixed-point pipeline

Floating-point pipeline

Branch pipeline

Load/store pipeline
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.
Multi-Core
Recall: Superscalar utilization by a thread

For an 8-way superscalar.

- memory conflict
- long fp
- short fp
- long integer
- short integer
- load delays
- control hazards
- branch misprediction
- dcache miss
- icache miss
- dtlb miss
- itlb miss
- processor busy

Observation: In many cases, the on-chip cache and DRAM I/O bandwidth is also underutilized by one CPU. So, let 2 cores share them.
Most of Power 5 die is shared hardware

Core #1

Core #2

Shared Components

L2 Cache

L3 Cache Control

DRAM Controller

Power4 and Power4+ systems both have 32-Mbyte L3 caches, whereas Power5 systems can support a 1.875-Mbyte on-chip L2 cache.
Core-to-core interactions stay on chip

(1) Threads on two cores that use shared libraries conserve L2 memory.

(2) Threads on two cores share memory via L2 cache operations. Much faster than 2 CPUs on 2 chips.
The case for Sun’s Niagara...

Observation: Some apps struggle to reach a CPI <= 1. For throughput on these apps, a large number of single-issue cores is better than a few superscalars.
Niagara: 32 threads on one chip

8 cores:
- Single-issue
- 6-stage pipeline
- 4-way multi-threaded
- Fast crypto support

Die size: 340 mm² in 90 nm.
Power: 50–60 W

Shared resources:
- 3MB on-chip cache
- 4 DDR2 interfaces
- 32G DRAM, 20 Gb/s
- 1 shared FP unit
- GB Ethernet ports

Sources: Hot Chips, via EE Times, Infoworld.
J Schwartz weblog (Sun COO)
Niagara status: First motherboard runs

Source: J Schwartz weblog (Sun COO)
Lab 4 “Town Meeting”
Lab 4: Reflections from the TAs

Everyone **worked hard**. Only in retrospect did most students realize they also had to **work smart**.

**Example**: Only one group member knows how to download to board. Once this member falls asleep, the group can’t go on working ...

**Solution**: Actually use the Lab Notebook to document processes. An example of **working smart**.
Lab 4: Reflections from the TAs

**Example:** Comprehensive test rigs seen as a “checkoff item” for Lab report, done last. Actual debugging proceeds in haphazard, painful way.

**A Better Way:** One group spent 10 hours up front writing a cache test module. Brandon “The best cache testing I’ve ever seen”. They finished on time. An example of working smart.
Lab 4: Reflections from the TAs

Example: Group has a long design meeting at start of project. Little is documented about signal names, state machine semantics. Members design incompatible modules, **suffer**.

A Better Way: Carry notebooks (silicon or paper) to meetings, and force documentation of the decisions on details.
Lab 4: Discussion ...
Conclusions: Throughput processing

Simultaneous Multithreading:
Instructions streams can share an out-of-order engine economically.

Multi-core: Once instruction-level parallelism run dry, thread-level parallelism is a good use of die area.

Lab 4: Hard work is admirable, but even reasonable deadlines are hard to meet if you don’t also work smart.