Getting More Out Of Processors

Everyone wants to compute faster, but how?

- Higher chip densities (Moore's Law) don't help because more transistors are not problem
- Faster clock rates slice EX into tinier pieces
- Power is an issue, a BIG ISSUE
- Lots of delays in the system … like accessing mem

1st idea: Use what we've got better: multi-threading

- Keep multiple contexts, switch when there's a stall such L1 cache miss
- Need 2x registers, 2 PCs and some care in design
- Threads share caches, TLB, and all HW units

1

More Ideas for Speed

2nd Idea: Forget 1-at-a-time EX -- superscalar

- Issue (Fetch/Decode) multiple instructions at a time
- Keep careful track of which operand registers the computation depends on
  - like hazard detection on steroids
  - forget using actual registers until the dust settles
- Actually execute instructions when data available
- Commit instructions (write to memory, etc.) in order
- Effective multi-issue machines have CPI < 1

2

Parallelism at the Instruction Level

```
add $2 <- $3, $4
or $2 <- $2, $4
lw $4 <- 0($6)
addi $7 <- $6, 0x5
sub $8 <- $8, $4
```

Dependences?
RAW
WAW
WAR

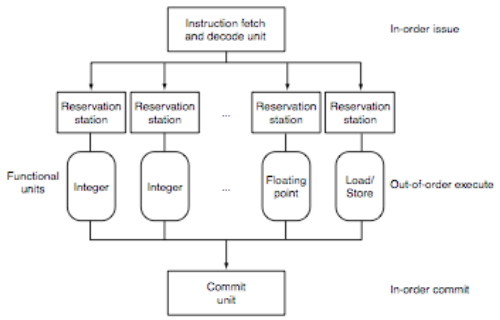
When can we reorder instructions?

When can we reorder instructions?

```
add $2 <- $3, $4
or $5 <- $2, $4
lw $6 <- 0($4)
sub $8 <- $8, $4
addi $7 <- $6, 0x5
```

Superscalar Processors:
Multiple instructions executing in parallel at *same* stage

OoO Execution Hardware



Bright Ideas (continued)

3rd Idea: Add multiple processors -- Parallelism

Parallelism is implemented using high transistor counts as multi-core processor chips

- Using multi-core (and all parallelism) requires multiple instruction streams -- threads
- One option is to use separate tasks: OS, virus checker, multimedia apps, etc.
- Only way to speed 1 computation is to express it as multiple threads of computation == parallel program == tough
- After decades of research automatic conversion of sequential programs into parallel programs has not worked == bad news == fun stuff to learn

5

Exploiting Parallelism

Of the computing problems for which performance is important, many have inherent parallelism

Best example: computer games

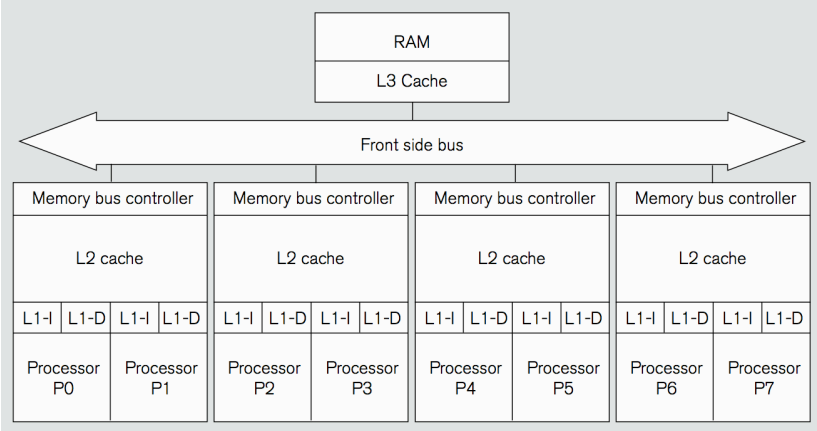
- Graphics, physics, sound, AI etc. can be done separately
- Furthermore, there is often parallelism within each of these:
  - Each pixel on the screen's color can be computed independently
  - Non-contacting objects can be updated/simulated independently
  - Artificial intelligence of non-human entities done independently

Another example: Google queries

- Every query is independent
- Index is read-only!!

6

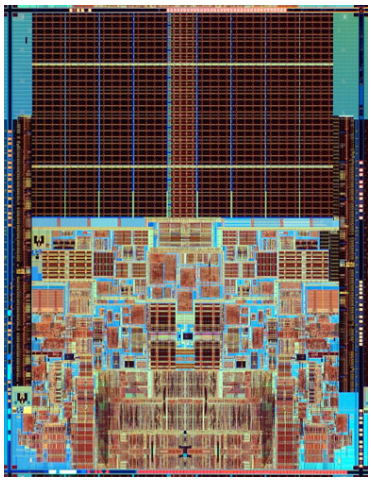
Quad Core Logical Structure



7

Multicore Machines

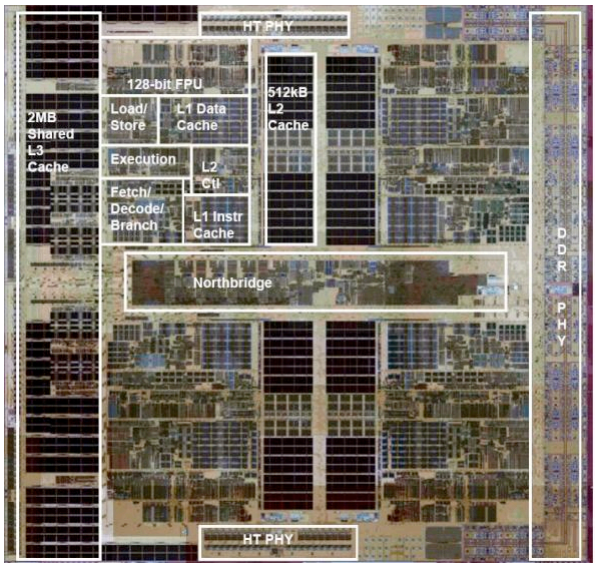
Intel CoreDuo



8

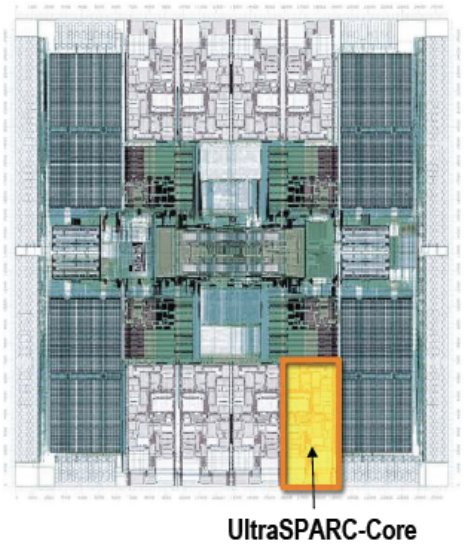
Multicore

AMD
Barcelona



Multicore

Sun Ultrasparc



10

Features of Multicore Processors

Multicore processors are

- ❖ Standard pipelined architectures … built with the "best" features
- ❖ Often exploit multithreading
- ❖ Private L1 Instruction and Data caches
- ❖ Shared L2 and L3 combined caches
- ❖ Shared off-chip ports

Think about these properties and what they imply for each processor's view

As parallel machines multicore "have a problem"

11

Parallel Threads Are Independent

When multiple threads collaborate to solve a single problem (parallel execution) the interactions in memory are key

Thread 0

```
x++
...
```

translates to

```
lw $8, 0($sp)
lw $9, 0xf0($fp)
addi $8, 1
sw $8, 0($sp)
```

Typical Execution

Load into $8 miss in L1,
and hits in L2, fill line ← delay
Load into $9 miss in L1,
and hits in L2, fill line ← delay
When stall over, add 1
Store into L1

What happens if thread T1 executes same code at same time, but hits in L1?

12

Cache Coherence

To ensure each processor references fresh data, the memory activity of each processor is snooped

The idea is that as each processor references memory, the controller for the other processor's cache checks (snoops) to see if it also has the location … if so, it may intervene in the memory reference behavior

This is a process is called cache coherency

It requires extra bits on each cache line beside the valid bit: MESI, MOESI, etc.

13

MESI Protocol

MESI has four states:

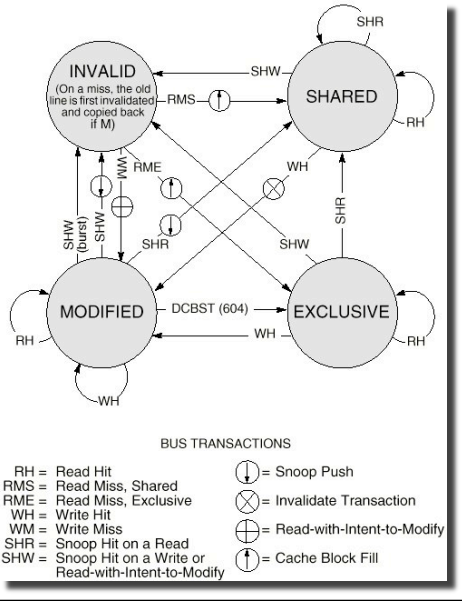
modified

exclusive

shared

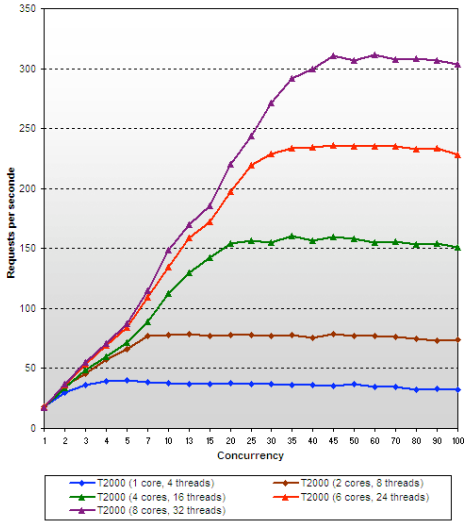
invalid

Very rich



Thanks: Slater & Tibrewala of CMU

How Good Is Performance: SQL



15

Summary

Multi-core is having more than one processor on the same chip.

- Soon all PCs/servers and game consoles will be multi-core
- Results from Moore's law and power constraint

Exploiting multi-core requires parallel programming

- Automatically extracting parallelism too hard for compiler, in general.
- But, can have compiler do much of the bookkeeping for us

Next time – parallel programming

16