Instruction-Level Parallelism

CSC 211 – December 10, 2020

Datapath Lab Q&A

Do we need to submit a branch instruction test?

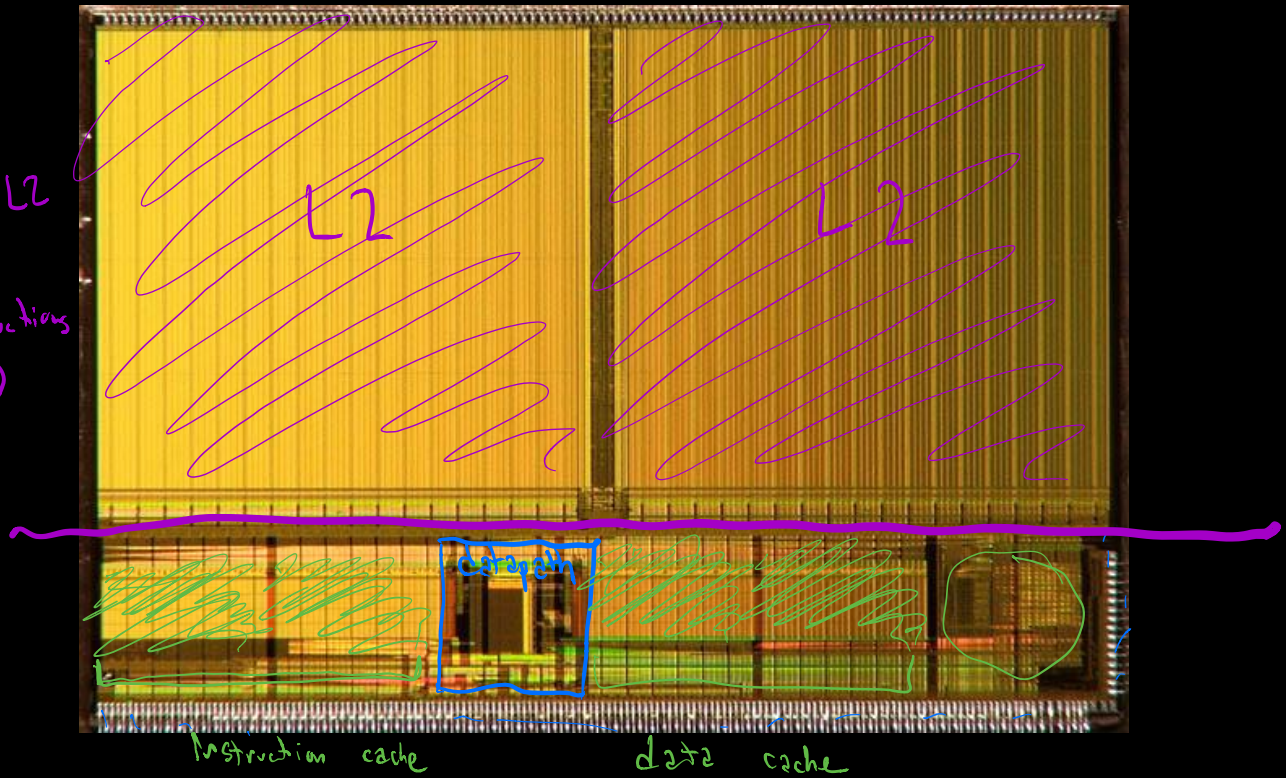
No, but I still recommend testing them.

Pay close attention to the instructions push and pop.

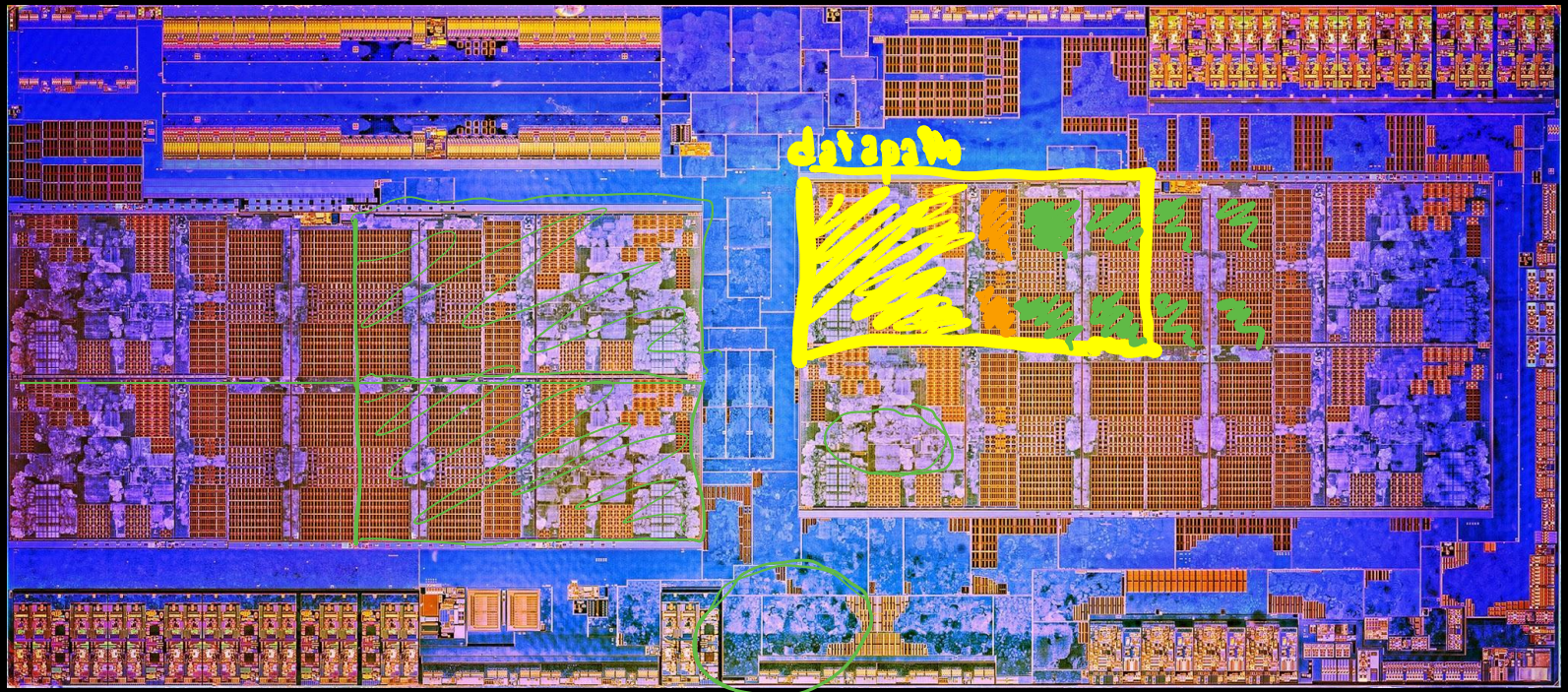
The parameters "-2" and "0" must be in quotes.

Memory Hierarchy, Continued

Multi-Level Caching

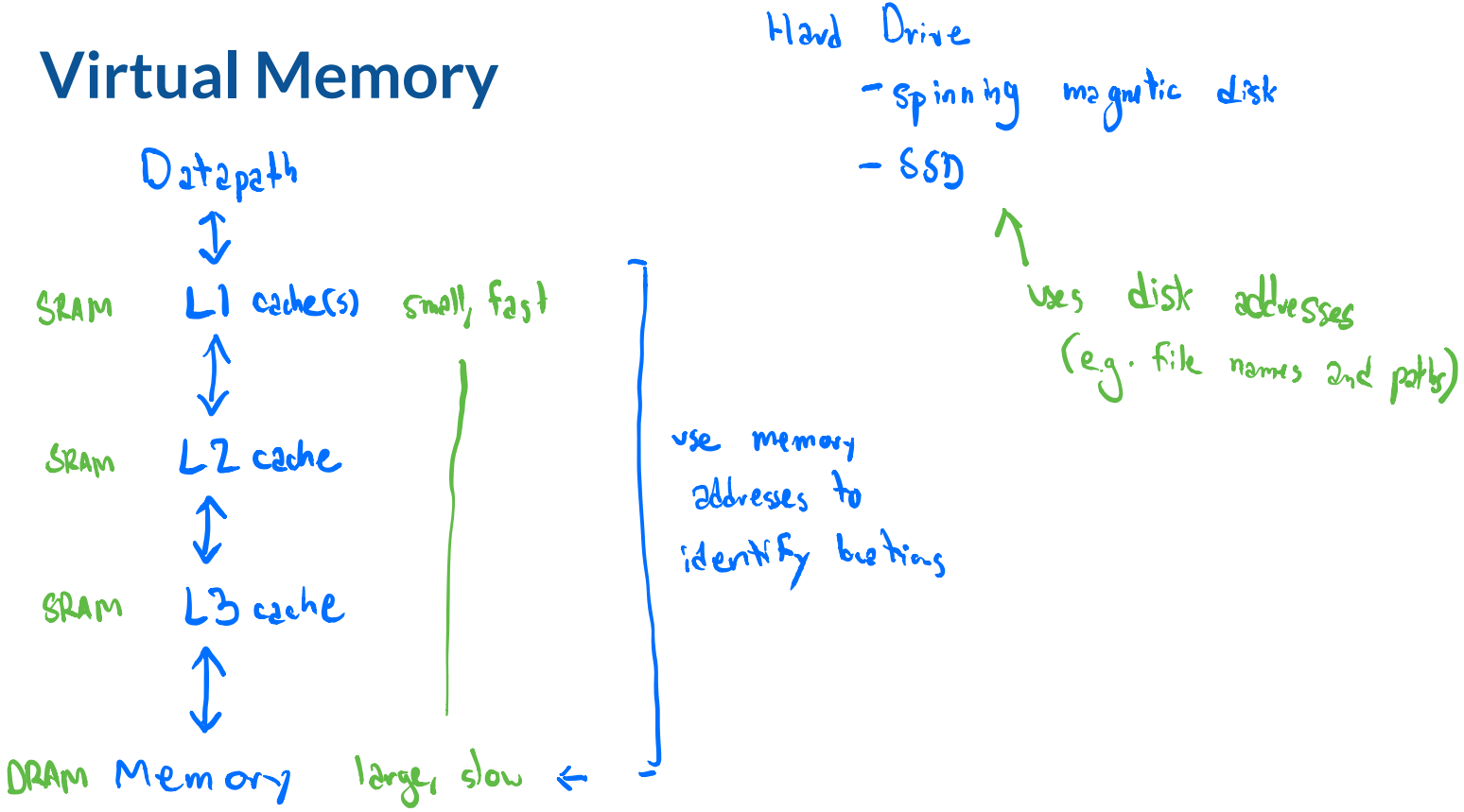


Intel Pentium Pro 256KB Cache (source: CPU-World)

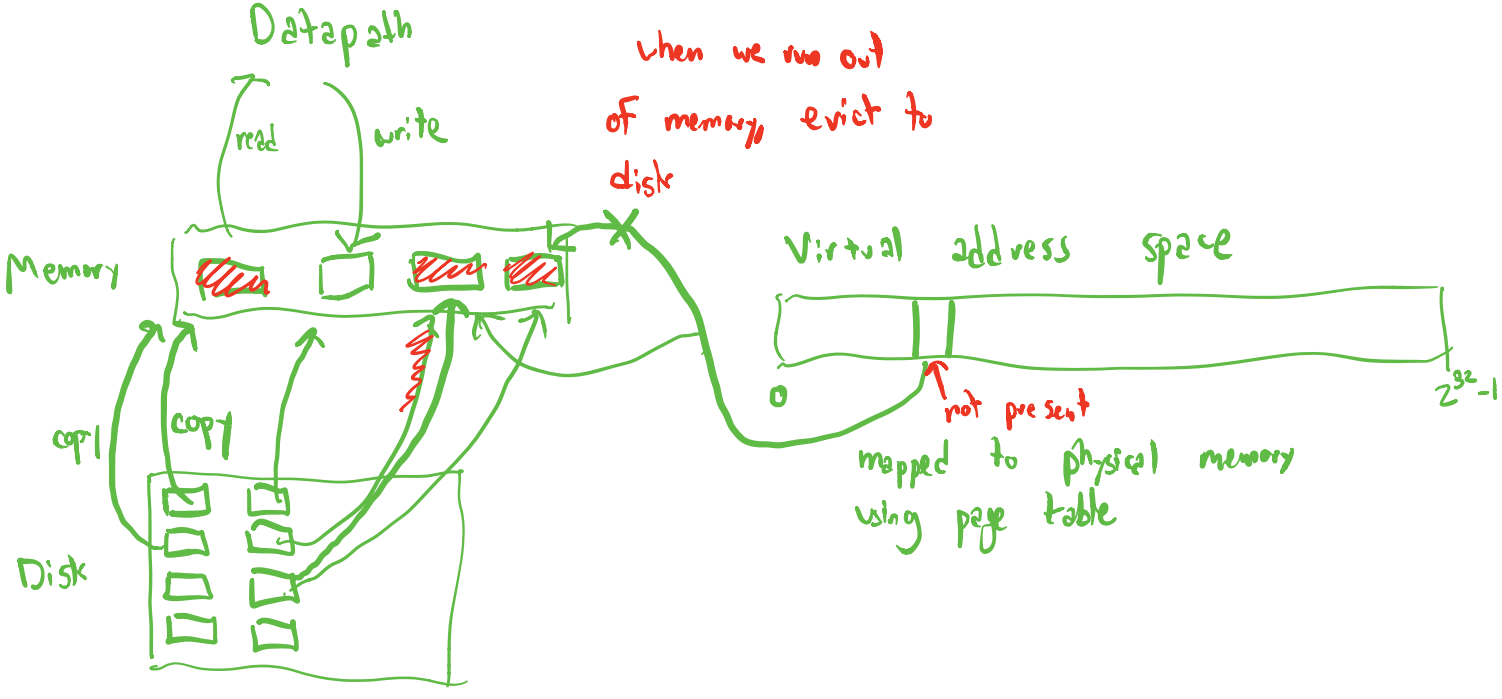


AMD Ryzen 4-Core Unit (source: PCWorld)

Virtual Memory



Virtual Memory



Instruction-Level Parallelism

Looking Back at Pipelining

Why did we complicate our nice datapath with pipelining?

We wanted to run more instructions per second.

Performance!

What effect did pipelining have?

Most or all parts of the datapath are busy at the same time.

Alternately: our programs now run five instructions at a time.

Are there situations where pipelining doesn't help as much as we'd like?

A single instruction will take longer — even more reasonable short sequences of instructions may not benefit from pipelining.

Instruction sequences w/ lots of dependencies sometimes need to stall.

Consider this code snippet:

```
addi $t0, $t0, 1
addi $t1, $t1, 1
addi $t0, $t0, 1
addi $t1, $t1, 1
addi $t0, $t0, 1
addi $t1, $t1, 1
```

depends on

Your Tasks:

1. Sketch the dependencies in this instruction sequence.
2. How many cycles it would take to execute these instructions on our regular pipeline with forwarding?

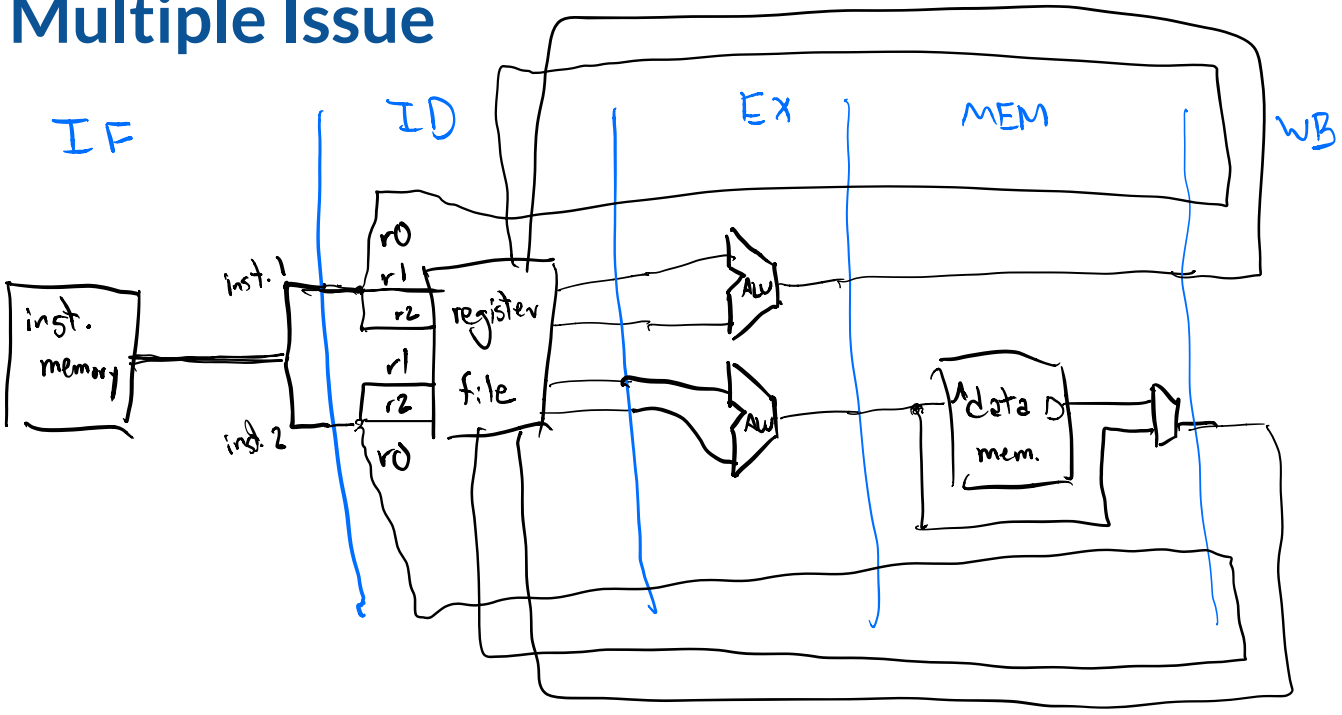
   6 cycles to start all, +4 to finish last inst = 10 cycles

3. Is there anything we can do to run this instruction sequence faster?

Why not run both of these at the same time?

it only take 7 cycles if we could run instructions in pairs.

Multiple Issue



Consider this code snippet:

```
addi $t0, $t0, 1
addi $t1, $t1, 1

addi $t0, $t0, 1
addi $t1, $t1, 1

addi $t0, $t0, 1
addi $t1, $t1, 1
```

Imagine we have a processor that can issue two adjacent instructions in each cycle.

Your Task:

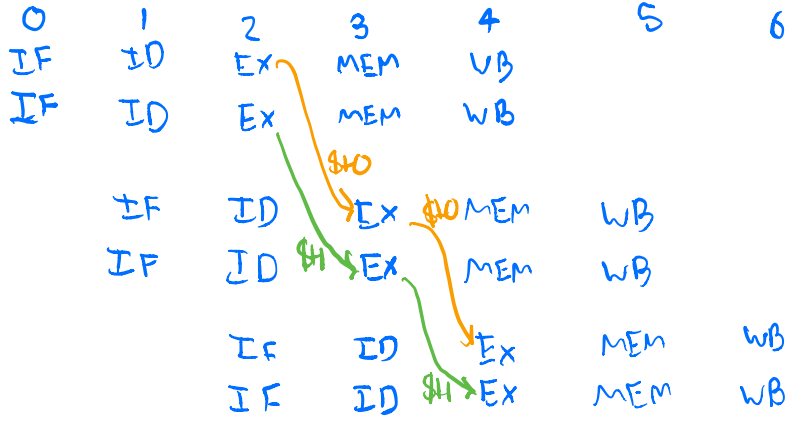
1. Sketch the pipeline schedule for this instruction sequence on such a processor.
2. How many cycles will this instruction sequence take to execute on our multiple-issue processor?

Consider this code snippet:

```
addi $t0, $t0, 1
addi $t1, $t1, 1

addi $t0, $t0, 1
addi $t1, $t1, 1

addi $t0, $t0, 1
addi $t1, $t1, 1
```



Consider this code snippet:

```
addi $t0, $t0, 1
addi $t0, $t0, 1

addi $t0, $t0, 1
addi $t1, $t1, 1

addi $t1, $t1, 1
addi $t1, $t1, 1
```

Consider this *slightly* different instruction sequence executing on the same multiple-issue processor.

1. What do we need to do to make this instruction sequence run correctly on our processor?

2. How could we improve the processor to run this sequence faster?

Consider this code snippet:

```
addi $t0, $t0, 1
addi $t0, $t0, 1

addi $t0, $t0, 1
addi $t1, $t1, 1

addi $t1, $t1, 1
addi $t1, $t1, 1
```

Consider this code snippet:

```
addi $t0, $t0, 1
addi $t0, $t0, 1

addi $t0, $t0, 1
addi $t1, $t1, 1

addi $t1, $t1, 1
addi $t1, $t1, 1
```

Extra Notes

Extra Notes