Welcome back to CSA29H!

Week A

Ed meme recap:


I'VE ALWAYS WANTED TO
PRETEND
TO BE AN ARCHITECT
memegenerator.net


You've
Been
Ghosthed!

Questions on lecture content?
Or about cats?

Quiz everyone say YIPPEE!

Poll

```
assign raddr1 = quiz;
wire [15:0] feedback = rdata1;
```

How was the quiz?

A. easy
B. mostly fine
C. mostly fine, but not enough time
D. too hard, but finished mostly in time
E. too hard and not enough time
F. too hard regardless of time

Stress

- 429H is not an easy class
  - Lots of new materials
  - Unfamiliar programming environments
  - Fast, often relentless pace
- Struggling in this course is normal
  - There will be times you won't know the answer of the solution
  - This is expected—we want we everyone to succeed, but the only way we can help is if you ask for it
- If you find yourself overly overwhelmed or spending more time on this class than you think you should be, please reach out to Dr. Gheith or the TAs
  - We can help out as far as the class goes
  - We can provide other resources where we are not able to help

Mental health resource available at UT

Another Ed Meme???

how i feel about misalignment #542

Anonymous
1 hour ago in General

PIN STAR WATCH 20 VIEWS

4

The Verilog code, a beast of a different nature, a creature of pure logic and cold, hard edges, speaks in a language of modules and wires, of registers and gates, a tongue that is at once precise and unforgiving. To gaze upon it is to stare into the very heart of the machine, to see the switches flipping and the electrons flowing in their preordained paths. But lurking within this orderly realm lies a terror that strikes fear into the hearts of even the most intrepid hardware designers: the specter of misalignment.

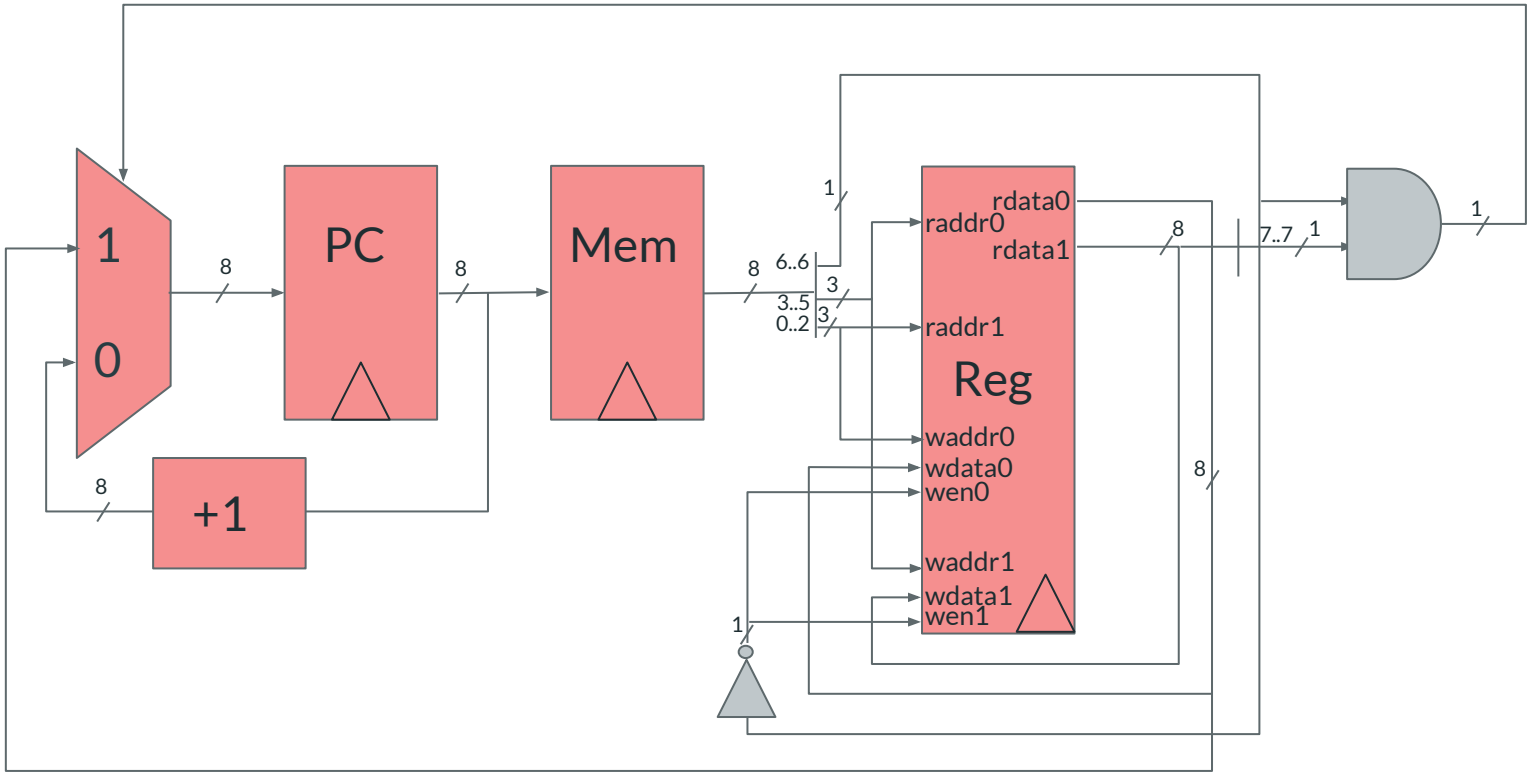
Ah, misalignment! That scourge of the digital world, that bane of the programmer's existence. It is a thing of nightmare, a twisted perversion of the natural order of things. When the program counter, that faithful guide that leads us through the labyrinthine paths of code, falls victim to misalignment, chaos reigns supreme.

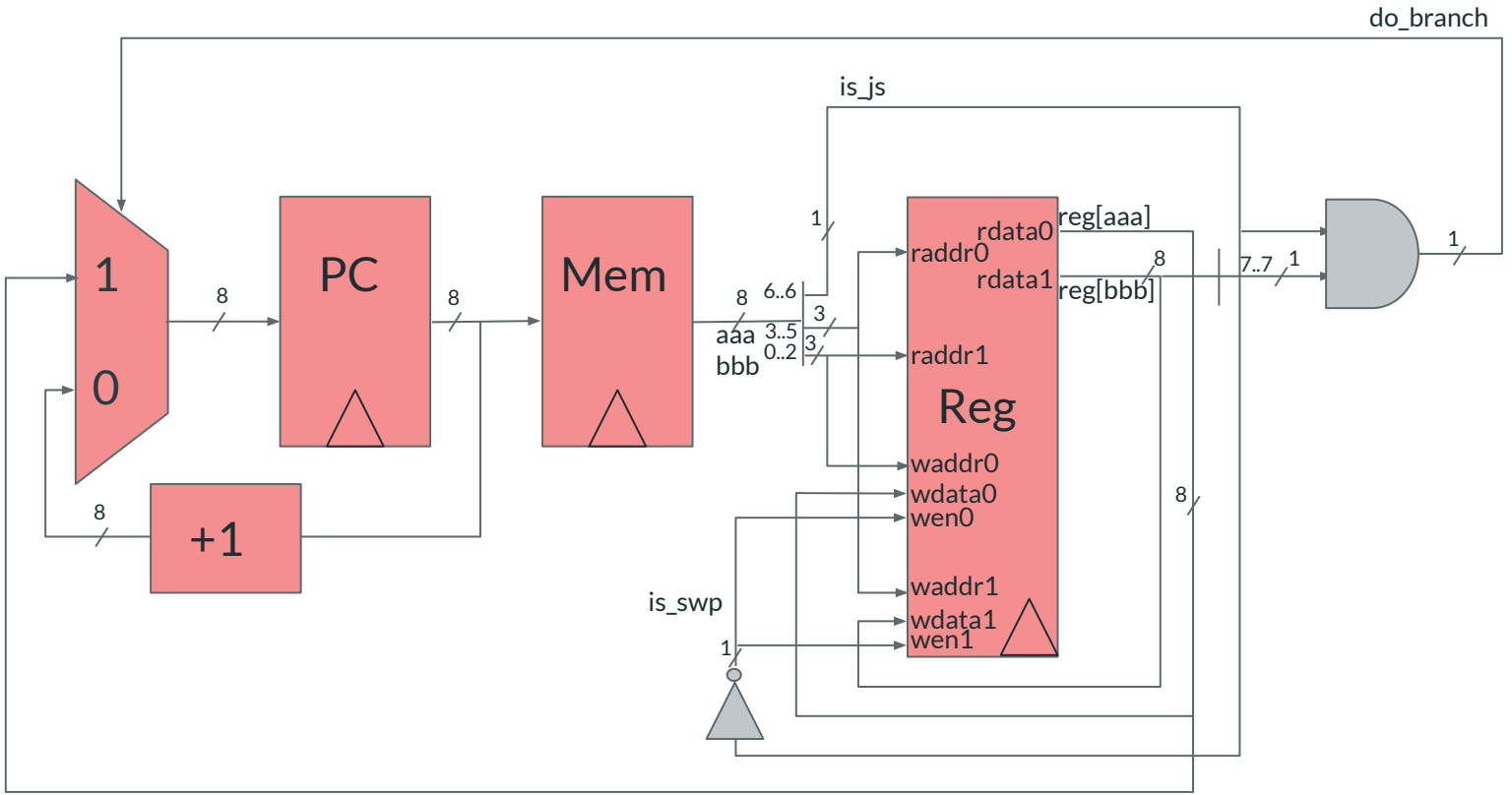
The once-orderly march of instructions becomes a shambling, lurching thing, a grotesque parody of the elegant dance it was meant to be. Bytes and words, the very building blocks of our digital universe, are rent asunder, their boundaries blurred and their meanings distorted. In this mad, misaligned world, the very foundations of logic and reason begin to crumble, and the hapless programmer is left to pick through the rubble, desperately seeking some glimmer of sense amidst the chaos.

The specter of misalignment is always waiting, lurking in the shadows, ready to pounce at the slightest sign of weakness or complacency. It is a constant reminder of the fragility of our pipe-dream world, of the precariousness of the order we have imposed upon the chaos.

Comment Edit Delete Endorse •••

Quiz Review 2: Electric Boogaloo


1
0
8
PC
8
Mem
8
6..6
3..5
0..2
3
3
1
rdata0
raddr0
rdata1
raddr1
Reg
waddr0
wdata0
wen0
waddr1
wdata1
wen1
8
7..7
1
1
8
+1
8
1
1


do_branch
is_js
1
0
8
PC
8
Mem
8
6..6
3..5
0..2
aaa
bbb
1
3
3
rdata0
raddr0
rdata1
raddr1
Reg
waddr0
wdata0
wen0
waddr1
wdata1
wen1
reg[aaa]
reg[bbb]
8
7..7
1
1
8
+1
8
is_swp
1

Final Project

Final Project Info!

- work in groups of **up to four people**
- presentations will be April 25th and April 26th
  - presentation scheduling is up to y'all to organize
  - project final submission will be due April 29
- anything architecture related
  - extend a project we already did
  - something completely new
- a project proposal will be due probably April 15
  - who is in the group
  - what are the main ideas/goals for the project
  - what research have you done
- form groups + ideas **now**

What we are looking for in presentation

- Be prepared!!
  - Have a backup plan if your live demo doesn't work
- Explain your work
  - Provide background that is appropriate for CS429H students
  - Ideally people will learn something about architecture from your presentation!
- Demonstrate what you did
  - Show screenshots of results, live demos, whatever is appropriate for your project

Final Project Ideas !!!

- We have posted a long list of project ideas on Ed
- Note: We have 2 FPGAs (maybe more) so please let us know early if you'll want one!

P8

Poll

How's your status on P8?

A. What's P8?
B. I've heard of it
C. I've cloned the starter code and/or looked through it
D. I've started planning/writing code
E. I'm mostly done but might still have bugs
F. P8 any% speedrun

Out of Order Execution

Order of Out

- what is **Tomasulo's Algorithm**?

Order of Out

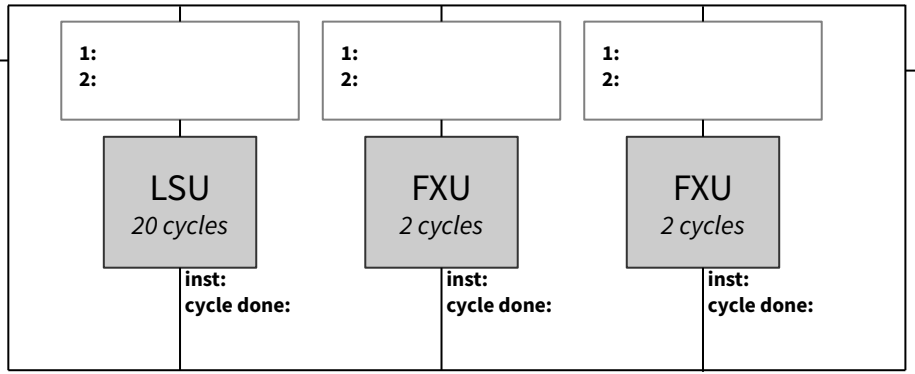
- what is **Tomasulo's Algorithm**?
- what is **register renaming**?
  - why do we do this?
- what is the **common data bus**?
- what is a **reservation station**?
  - what data is stored in a reservation station?
- what is the **reorder buffer**?
  - what data is stored in each entry within the reorder buffer?

Out of Order Processor

register file

	value	busy	owner
r0	0	0	-
r1	0	0	-
r2	0	0	-
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	0	0	-
r7	0	0	-

cycle number: 0



reorder buffer

	inst	valid	result
(H, T) 0	-	0	-
1	-	0	-
2	-	0	-
3	-	0	-
4	-	0	-
5	-	0	-
6	-	0	-
7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

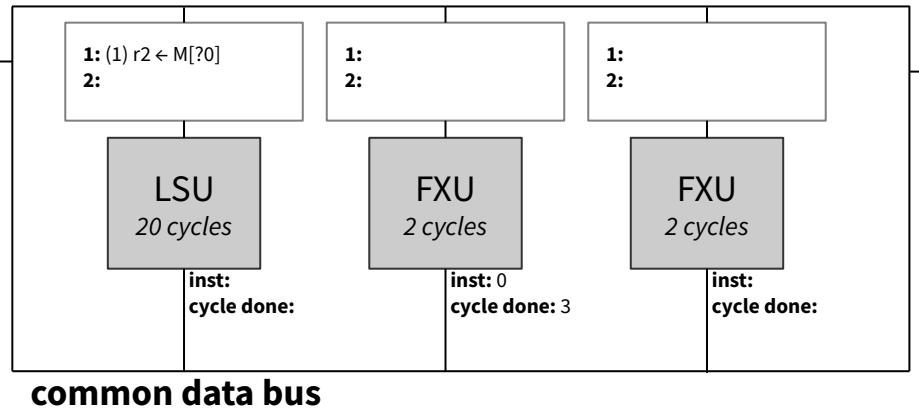
assume you can decode and issue two instructions to either a FU or a RS in one cycle

Out of Order Processor

register file

	value	busy	owner
r0	0	0	-
r1	0	0	-
r2	0	1	1
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	0	1	0
r7	0	0	-

cycle number: 1



reorder buffer

	inst	valid	result
(T) 0	r6 ← 123	0	-
1	r2 ← M[r6]	0	-
(H) 2	-	0	-
3	-	0	-
4	-	0	-
5	-	0	-
6	-	0	-
7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

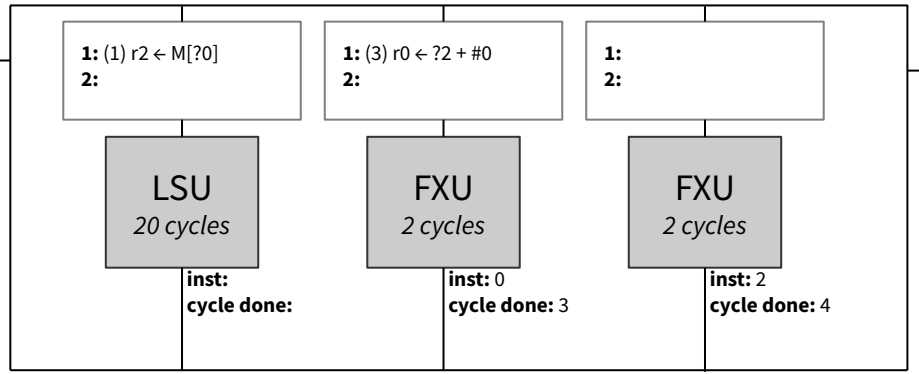
assume you can decode and issue two instructions to either a FU or a RS in one cycle

Out of Order Processor

register file

	value	busy	owner
r0	0	1	3
r1	0	1	2
r2	0	1	1
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	0	1	0
r7	0	0	-

cycle number: 2



reorder buffer

	inst	valid	result
(T) 0	r6 ← 123	0	-
1	r2 ← M[r6]	0	-
2	r1 ← 42	0	-
3	r0 ← r1 + r3	0	-
(H) 4	-	0	-
5	-	0	-
6	-	0	-
7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

assume you can decode and issue two instructions to either a FU or a RS in one cycle

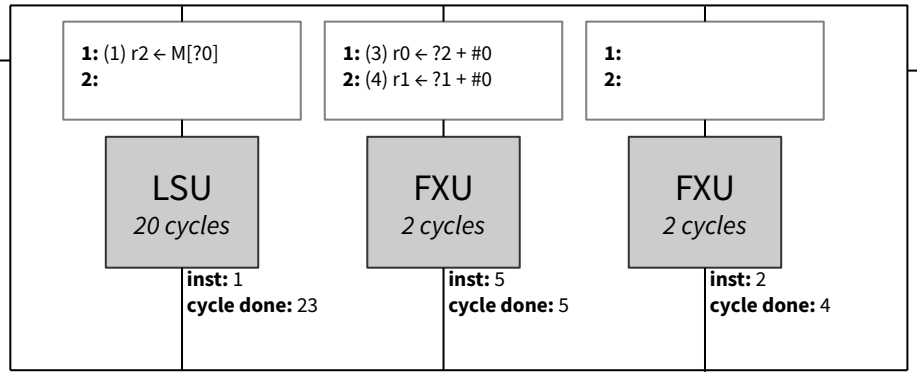
Out of Order Processor

cycle number: 3

register file

	value	busy	owner
r0	0	1	3
r1	0	1	5
r2	0	1	1
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	0	1	0
r7	0	0	-

reservation stations



reorder buffer

	inst	valid	result
(T) 0	r6 ← 123	1	123
1	r2 ← M[r6]	0	-
2	r1 ← 42	0	-
3	r0 ← r1 + r3	0	-
4	r1 ← r2 + r3	0	-
5	r1 ← r3 + r4	0	-
(H) 6	-	0	-
7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

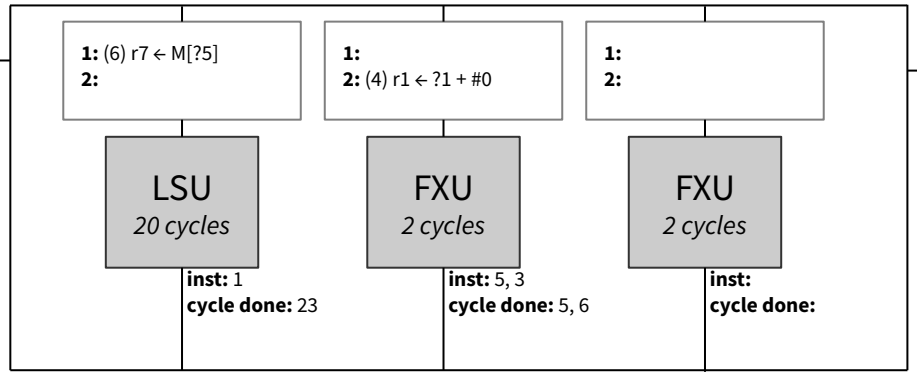
assume you can decode and issue two instructions to either a FU or a RS in one cycle

Out of Order Processor

register file

	value	busy	owner
r0	0	1	3
r1	0	1	5
r2	0	1	1
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	123	0	-
r7	0	1	6

cycle number: 4



reorder buffer

	inst	valid	result
0	-	0	-
(T) 1	r2 ← M[r6]	0	-
2	r1 ← 42	1	42
3	r0 ← r1 + r3	0	-
4	r1 ← r2 + r3	0	-
5	r1 ← r3 + r4	0	-
6	r7 ← M[r1]	0	-
(H) 7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

assume you can decode and issue two instructions to either a FU or a RS in one cycle

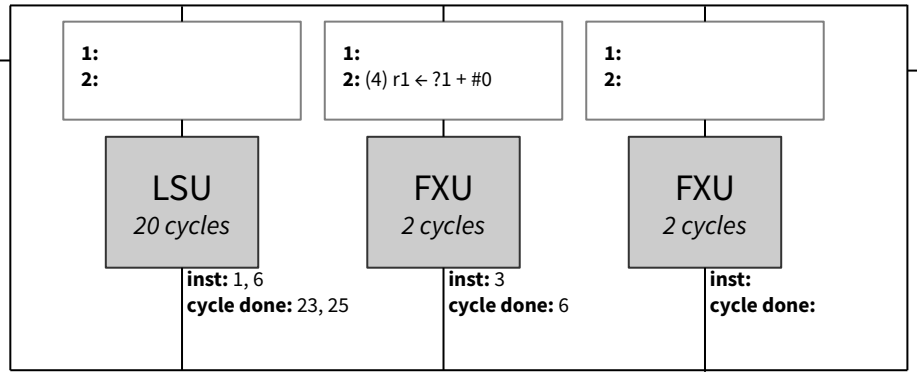
Out of Order Processor

cycle number: 5

register file

	value	busy	owner
r0	0	1	3
r1	0	1	5
r2	0	1	1
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	123	0	-
r7	0	1	6

reservation stations



common data bus

reorder buffer

	inst	valid	result
0	-	0	-
(T) 1	r2 ← M[r6]	0	-
2	r1 ← 42	1	42
3	r0 ← r1 + r3	0	-
4	r1 ← r2 + r3	0	-
5	r1 ← r3 + r4	1	0
6	r7 ← M[r1]	0	-
(H) 7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

assume you can decode and issue two instructions to either a FU or a RS in one cycle

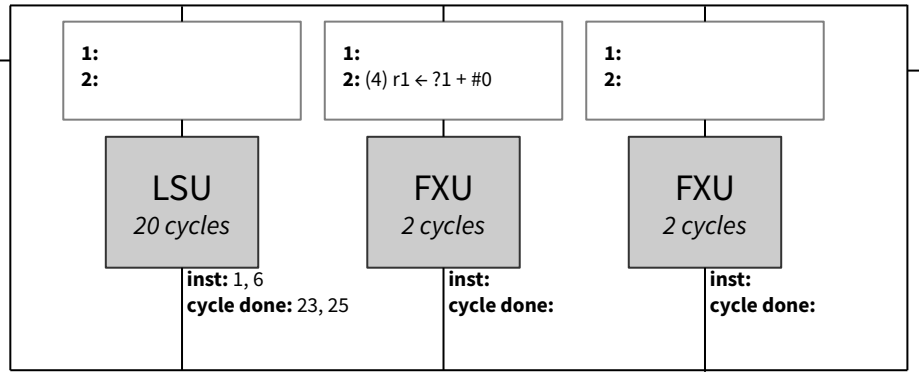
Out of Order Processor

cycle number: 6

register file

	value	busy	owner
r0	0	1	3
r1	0	1	5
r2	0	1	1
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	123	0	-
r7	0	1	6

reservation stations



reorder buffer

	inst	valid	result
0	-	0	-
(T) 1	r2 ← M[r6]	0	-
2	r1 ← 42	1	42
3	r0 ← r1 + r3	1	42
4	r1 ← r2 + r3	0	-
5	r1 ← r3 + r4	1	0
6	r7 ← M[r1]	0	-
(H) 7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

assume you can decode and issue two instructions to either a FU or a RS in one cycle

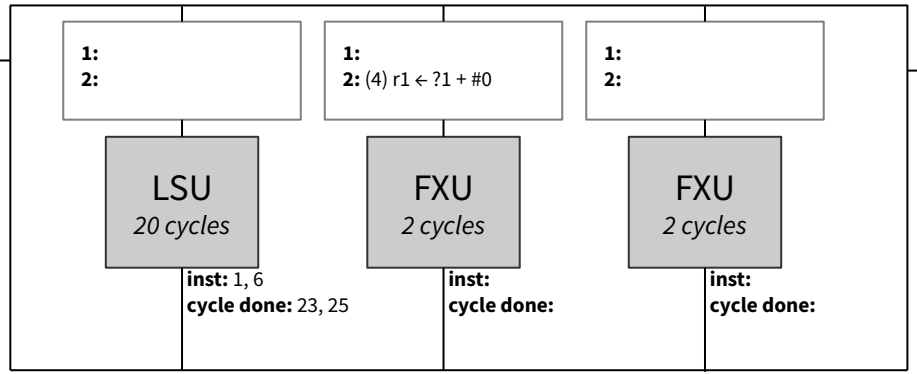
Out of Order Processor

cycle number: 7

register file

	value	busy	owner
r0	0	1	3
r1	0	1	5
r2	0	1	1
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	123	0	-
r7	0	1	6

reservation stations



common data bus

reorder buffer

	inst	valid	result
0	-	0	-
(T) 1	r2 ← M[r6]	0	-
2	r1 ← 42	1	42
3	r0 ← r1 + r3	1	42
4	r1 ← r2 + r3	0	-
5	r1 ← r3 + r4	1	0
6	r7 ← M[r1]	0	-
(H) 7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

assume you can decode and issue two instructions to either a FU or a RS in one cycle

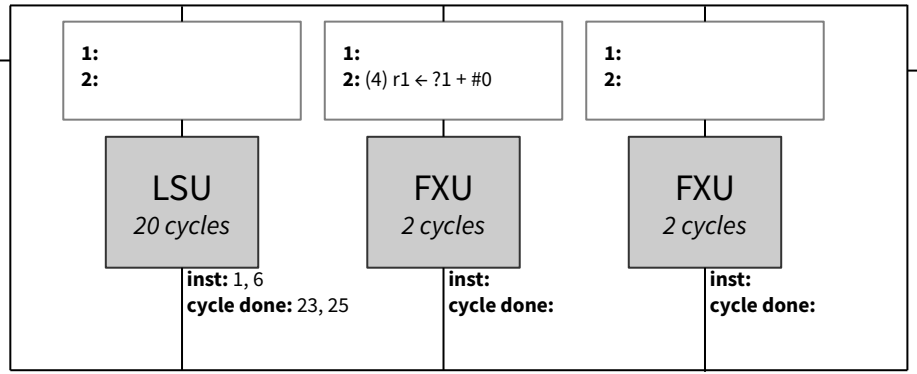
Out of Order Processor

cycle number: 8

register file

	value	busy	owner
r0	0	1	3
r1	0	1	5
r2	0	1	1
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	123	0	-
r7	0	1	6

reservation stations



common data bus

reorder buffer

	inst	valid	result
0	-	0	-
(T) 1	r2 ← M[r6]	0	-
2	r1 ← 42	1	42
3	r0 ← r1 + r3	1	42
4	r1 ← r2 + r3	0	-
5	r1 ← r3 + r4	1	0
6	r7 ← M[r1]	0	-
(H) 7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

assume you can decode and issue two instructions to either a FU or a RS in one cycle

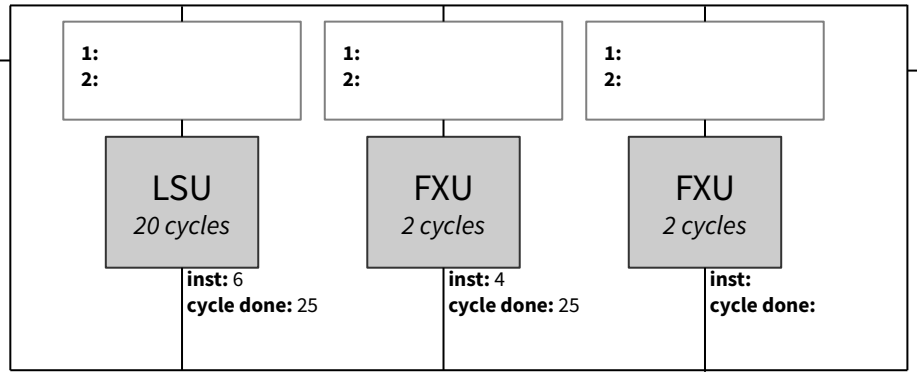
Out of Order Processor

cycle number: 23

register file

	value	busy	owner
r0	0	1	3
r1	0	1	5
r2	0	1	1
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	123	0	-
r7	0	1	6

reservation stations



common data bus

reorder buffer

	inst	valid	result
0	-	0	-
(T) 1	r2 ← M[r6]	1	456
2	r1 ← 42	1	42
3	r0 ← r1 + r3	1	42
4	r1 ← r2 + r3	0	-
5	r1 ← r3 + r4	1	0
6	r7 ← M[r1]	0	-
(H) 7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

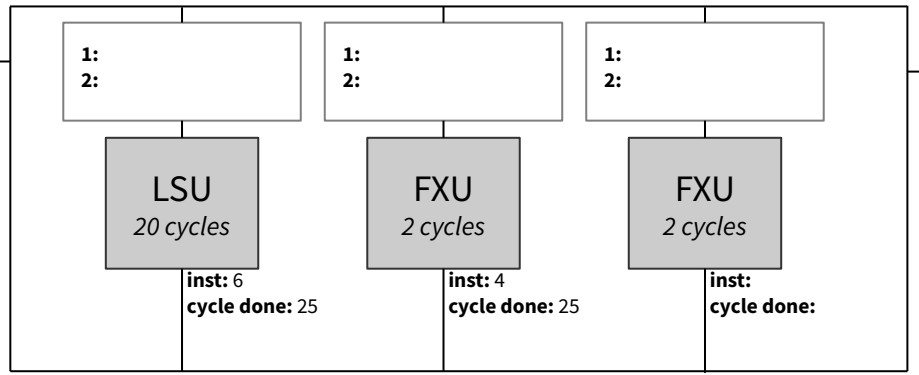
assume you can decode and issue two instructions to either a FU or a RS in one cycle

Out of Order Processor

cycle number: 23

register file

	value	busy	owner
r0	0	1	3
r1	0	1	5
r2	456	0	-
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	123	0	-
r7	0	1	6



reorder buffer

	inst	valid	result
0	-	0	-
1	-	0	-
(T) 2	r1 ← 42	1	42
3	r0 ← r1 + r3	1	42
4	r1 ← r2 + r3	0	-
5	r1 ← r3 + r4	1	0
6	r7 ← M[r1]	0	-
(H) 7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

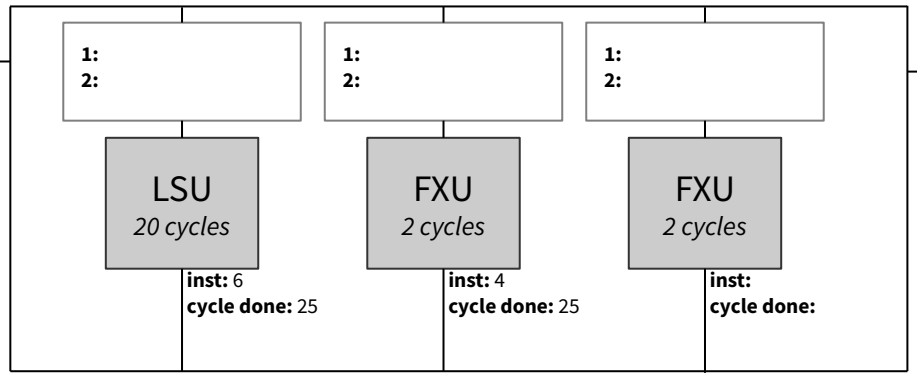
assume you can decode and issue two instructions to either a FU or a RS in one cycle

Out of Order Processor

cycle number: 24

register file

	value	busy	owner
r0	0	1	3
r1	42	1	5
r2	456	0	-
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	123	0	-
r7	0	1	6



reorder buffer

	inst	valid	result
0	-	0	-
1	-	0	-
2	-	0	-
(T) 3	r0 ← r1 + r3	1	42
4	r1 ← r2 + r3	0	-
5	r1 ← r3 + r4	1	0
6	r7 ← M[r1]	0	-
(H) 7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

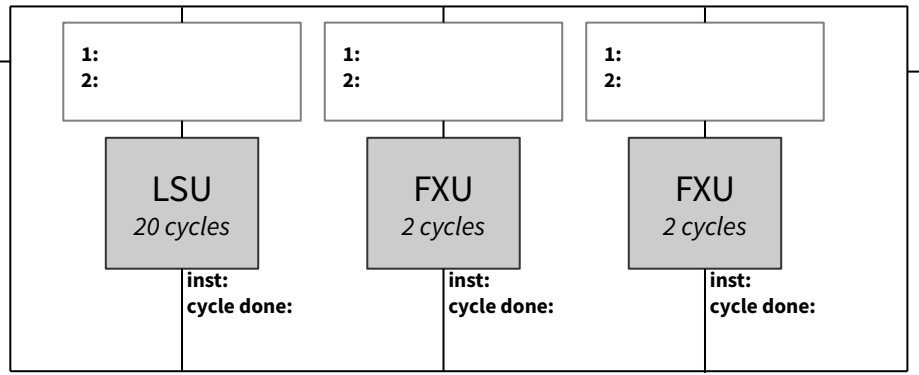
assume you can decode and issue two instructions to either a FU or a RS in one cycle

Out of Order Processor

cycle number: 25

register file

	value	busy	owner
r0	42	0	-
r1	42	1	5
r2	456	0	-
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	123	0	-
r7	0	1	6



reorder buffer

	inst	valid	result
0	-	0	-
1	-	0	-
2	-	0	-
3	-	0	-
(T) 4	r1 ← r2 + r3	1	456
5	r1 ← r3 + r4	1	0
6	r7 ← M[r1]	1	789
(H) 7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

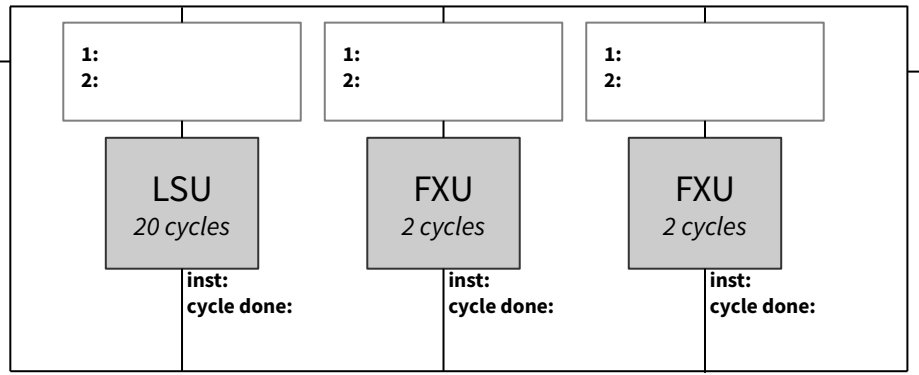
assume you can decode and issue two instructions to either a FU or a RS in one cycle

Out of Order Processor

register file

	value	busy	owner
r0	42	0	-
r1	456	1	5
r2	456	0	-
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	123	0	-
r7	0	1	6

cycle number: 26



reorder buffer

	inst	valid	result
0	-	0	-
1	-	0	-
2	-	0	-
3	-	0	-
4	-	0	-
(T) 5	r1 ← r3 + r4	1	0
6	r7 ← M[r1]	1	789
(H) 7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

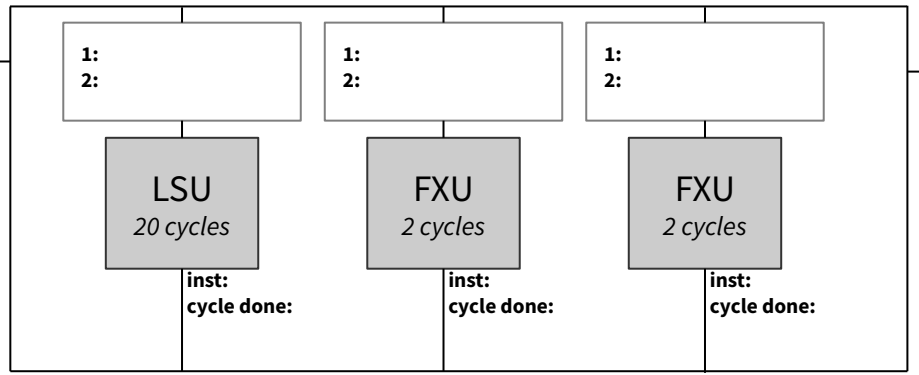
assume you can decode and issue two instructions to either a FU or a RS in one cycle

Out of Order Processor

register file

	value	busy	owner
r0	42	0	-
r1	0	0	-
r2	456	0	-
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	123	0	-
r7	0	1	6

cycle number: 27



reorder buffer

	inst	valid	result
0	-	0	-
1	-	0	-
2	-	0	-
3	-	0	-
4	-	0	-
5	-	0	-
(T) 6	r7 ← M[r1]	1	789
(H) 7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

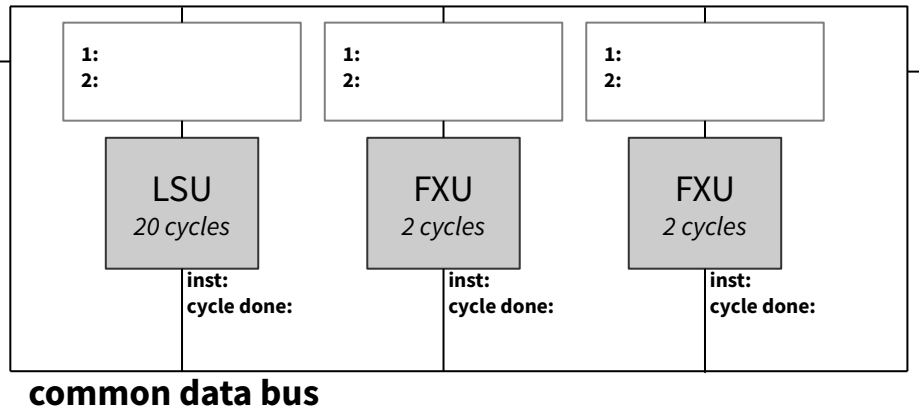
assume you can decode and issue two instructions to either a FU or a RS in one cycle

Out of Order Processor

register file

	value	busy	owner
r0	42	0	-
r1	0	0	-
r2	456	0	-
r3	0	0	-
r4	0	0	-
r5	0	0	-
r6	123	0	-
r7	789	0	-

cycle number: 28



reorder buffer

	inst	valid	result
0	-	0	-
1	-	0	-
2	-	0	-
3	-	0	-
4	-	0	-
5	-	0	-
6	-	0	-
(H, T) 7	-	0	-

instructions:

```
r6 ← 123
r2 ← M[r6]
r1 ← 42
r0 ← r1 + r3
r1 ← r2 + r3
r1 ← r3 + r4
r7 ← M[r1]
```

assume you can decode and issue two instructions to either a FU or a RS in one cycle

Questions?