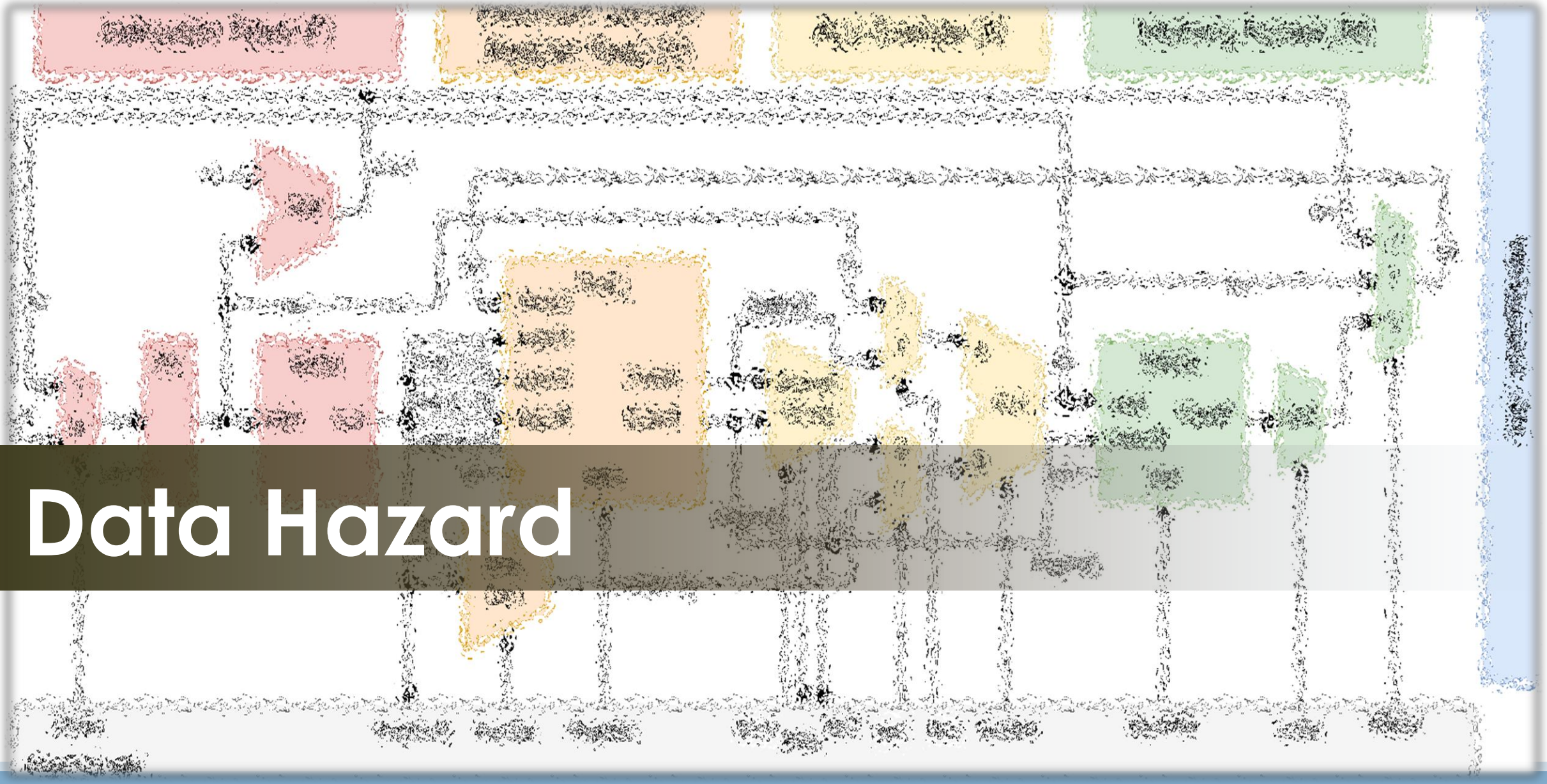
EECS 151/251A SP2022 Discussion 5

GSI: Yikuan Chen, Dima Nikiforov

Slides modified from Alisha Menon's, Zhenghan Lin's and Daniel Grubb's slides

Agenda

- Data Hazard – Forwarding
- Control Hazard

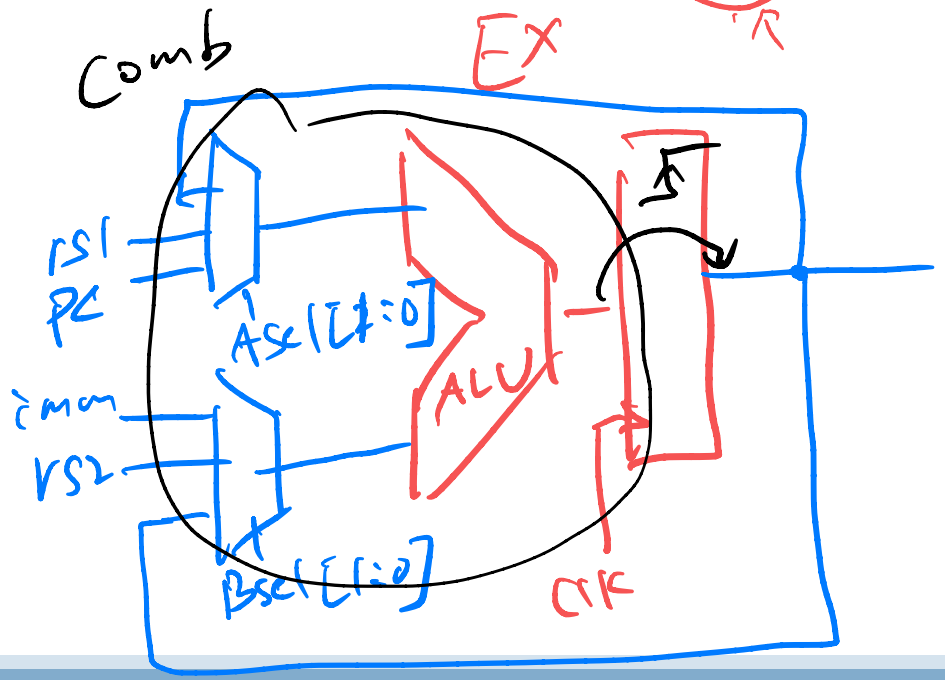


Data Hazard

Data Hazard - forwarding

Consider a 5-stage pipeline:

```
add x3, x1, x2
sub x5, x4, x3
```



No Forwarding

cycle	1	2	3	4	5	6	7	8
add	IF	ID	EX	M	WB			
sub		IF	ID	-	-	EX	M	WB

With Forwarding

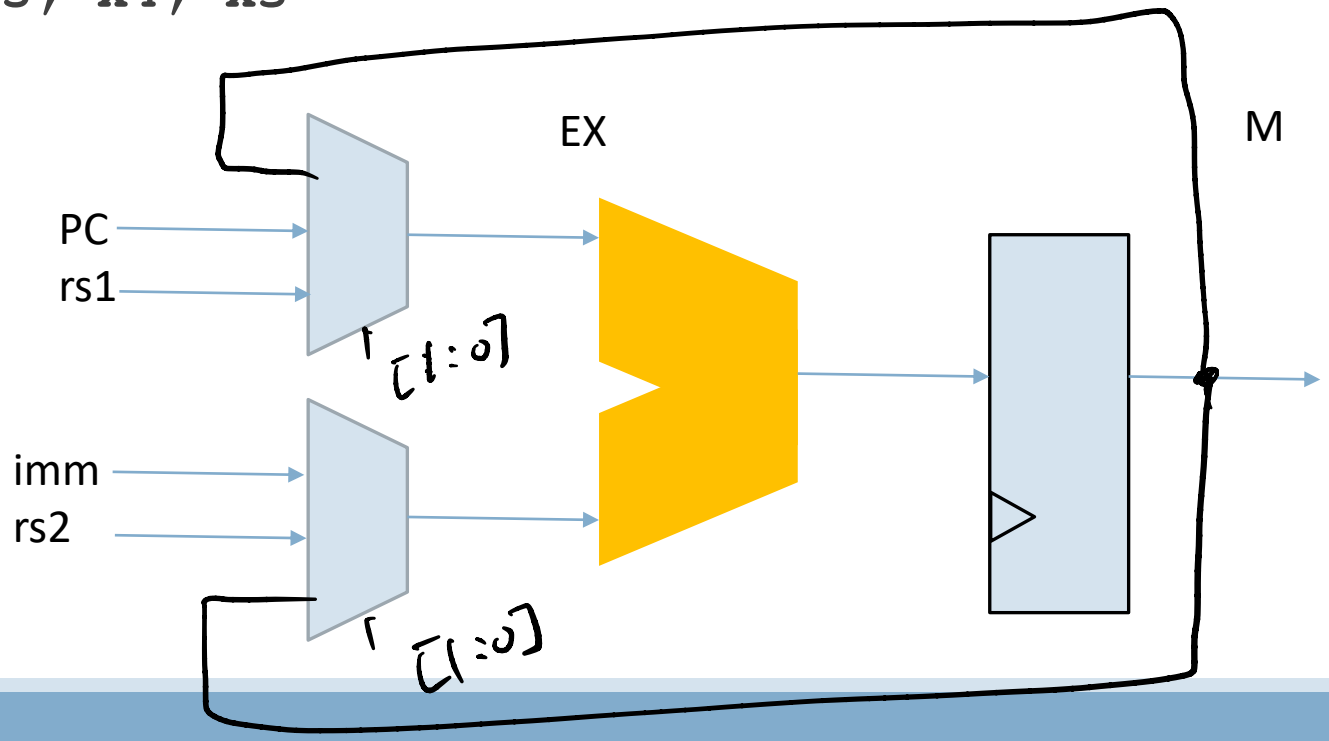
cycle	1	2	3	4	5	6	7	8
add	IF	ID	EX	M	WB			
sub		IF	ID	EX	M	WB		

Data Hazard – extra hardware for forwarding

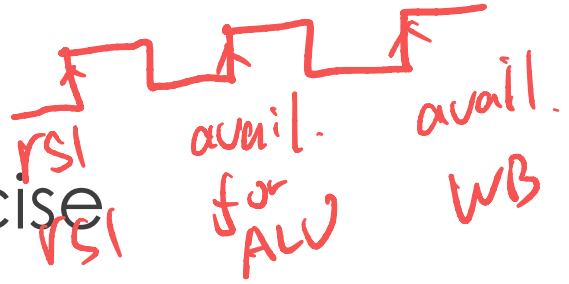
Consider a 5-stage pipeline:

```
add x3, x1, x2
sub x5, x4, x3
```

cycle	1	2	3	4	5	6	7	8
add	IF	ID	EX	M	WB			
sub		IF	ID	EX	M	WB		

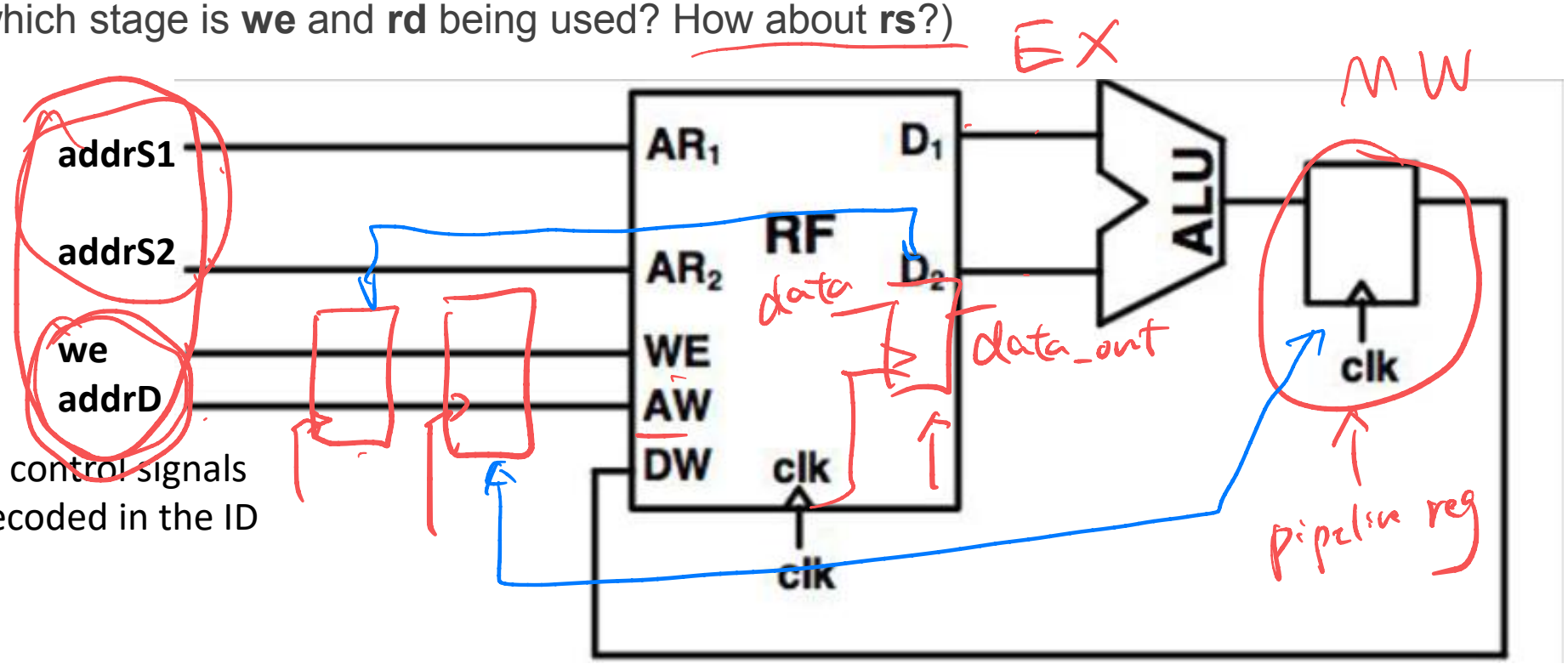


Pipeline Exercise



Async Read ⟺ Comb. wire
Sync Read ⟺ extra flipflop at read port

Below is part of a pipelined datapath with **synchronous reads and writes RegFile**. Assuming no potential data hazards, this design still does not function corretly for **r-type instructions** (ignore immediate instructions for now). What caused the error? Add any extra components necessary to correct the design (Hint: which stage is **we** and **rd** being used? How about **rs**?)

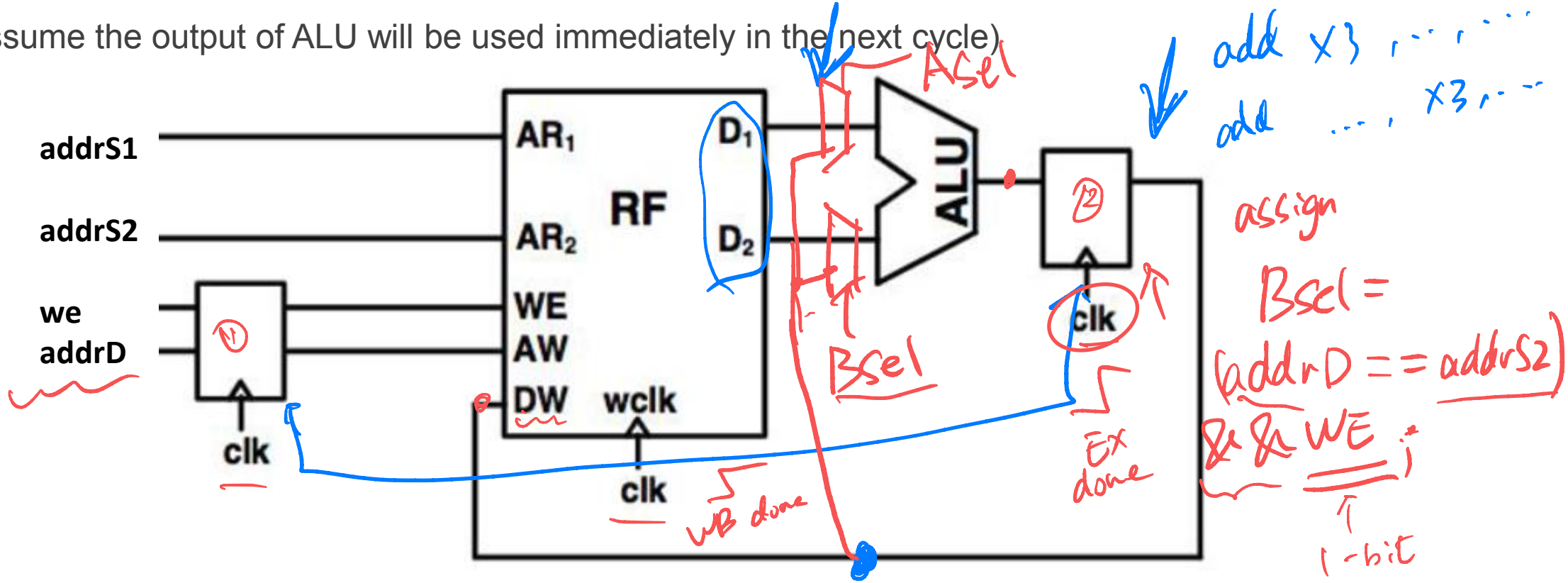


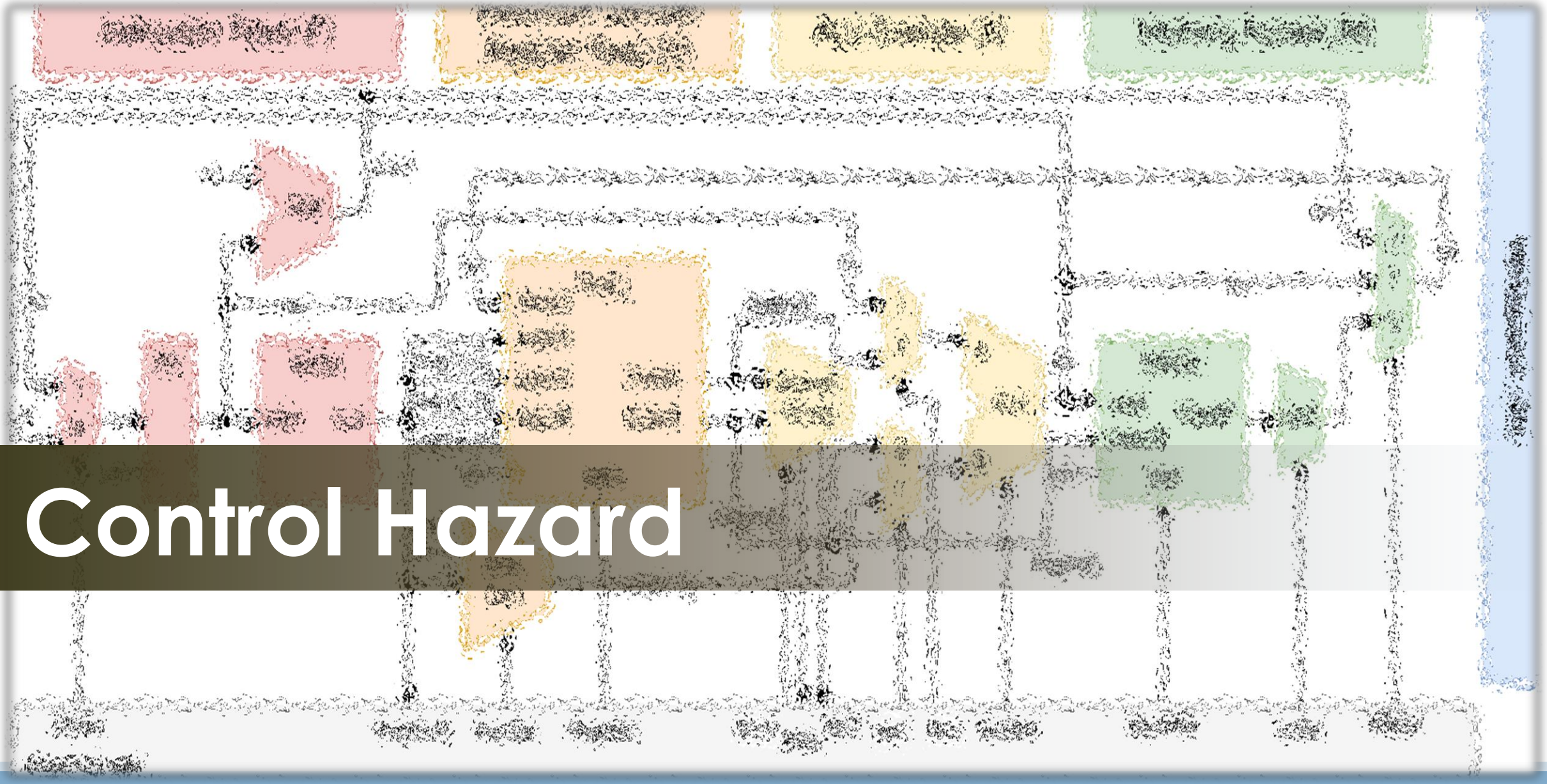
These control signals are decoded in the ID stage

Pipeline Exercise - Forwarding

Now this **RegFile** is modified and it has **synchronous writes** with **asynchronous reads**. Add appropriate forwarding to eliminate all data hazards (ignore hardware for immediate instructions for now).

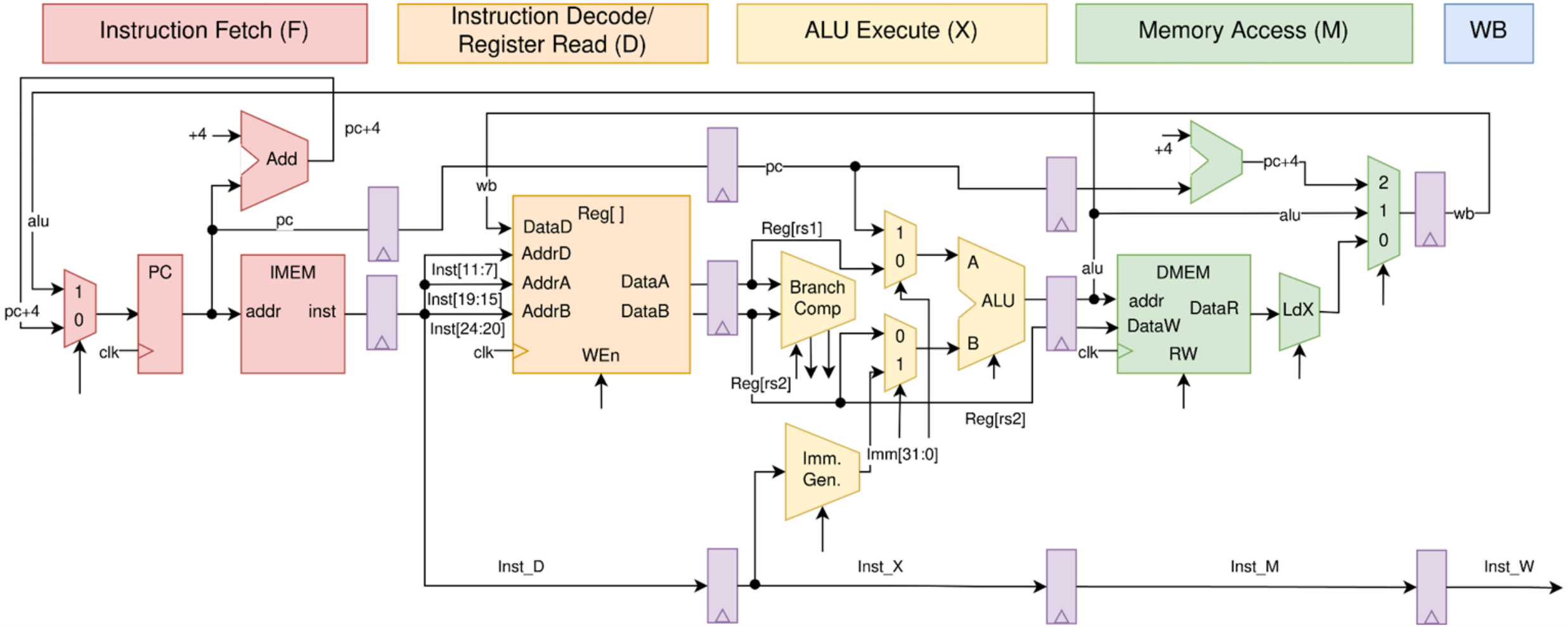
(Hint: Assume the output of ALU will be used immediately in the next cycle)





Control Hazard

Control Hazard



Control Hazard – case 1

- Branches are **not** taken by default
  - x1 = x2

```
  beq x1, x2, imm
      add x3, x1, x2
      sub x4, x1, x2
      xor x5, x1, x2
      or  x6, x1, x2
        ...
imm:  and x3, x1, x2
      nop
```

#	IF	D	EX	M	WB
1	beq				
2	add	beq			
3	sub	add	beq		
4	xor	sub	add	beq	
5	and	–	–	–	beq
6	nop	and	–	–	–
7					
8					
9					
10					

predict not taken

branch result avail.

Control Hazard – case 2

Branches are **taken** by default

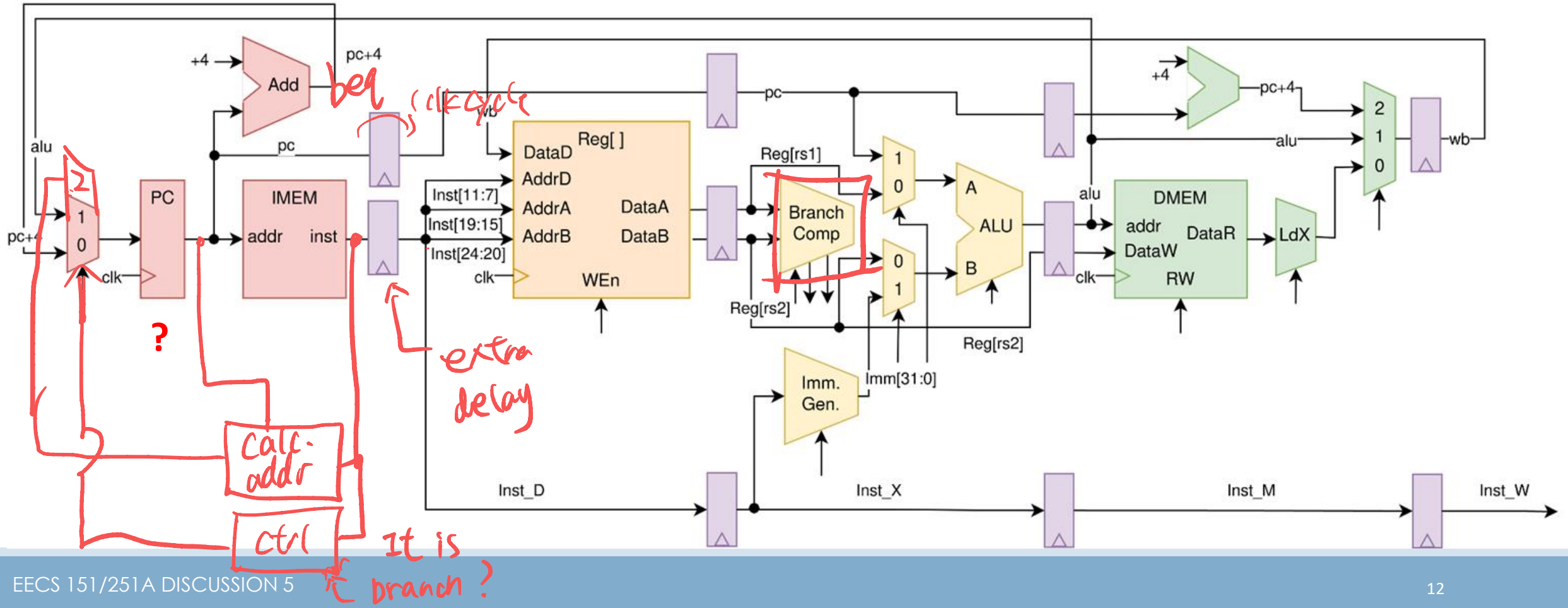
- with forwarding – assume no data hazard
- Assume x1 = x2

```
    beq x1, x2, imm
    add x3, x1, x2
    sub x4, x1, x2
    xor x5, x1, x2
    or  x6, x1, x2
      ...
imm: and x3, x1, x2
    nop
```

#	IF	D	EX	M	WB
1	beq				
2	and	beq			
3	nop	and	beq		
4		nop	and	beq	
5			nop	and	beq
6				nop	and
7					nop
8					
9					
10					

Control Hazard

- How do we modify the datapath to make sure branch address is available at next cycle? (Hint: what should we forward in this case?)



Control Hazard – Branch Prediction

- Base on last choice is a naïve but useful strategy in many cases
- Consider the following (very common case) code. If we predict based on previous branch result, what's the success vs failure rate?

```
addi x1 , x0 , 0
addi x2 , x0 , 1
addi x10, x0 , 100
add  x1 , x1 , x2
addi x2 , x2 , 1
blt  x2 , x10, -8
nop
```

```
// written in c
int s = 0;
for (int i=0; i<100; i++){
  s += i;
}
```

99 success

1 fail

1st : i = 0 predict taken (✓)

i = 1 again (✓)

⋮

i = 99 again (✓)

i = 100 again (✗)