CASPER

CS305: Computer Architecture

Pipeline Hazards: Mitigations

https://www.cse.iitb.ac.in/~biswa/courses/CS305/main.html

https://www.cse.iitb.ac.in/~biswa/

Data Hazard Detector and stalls

Execute to decode:

EX/MEM.RegisterRd = ID/EX.RegisterRs

EX/MEM.RegisterRd = ID/EX.RegisterRt

Memory to decode:

MEM/WB.RegisterRd = ID/EX.RegisterRs

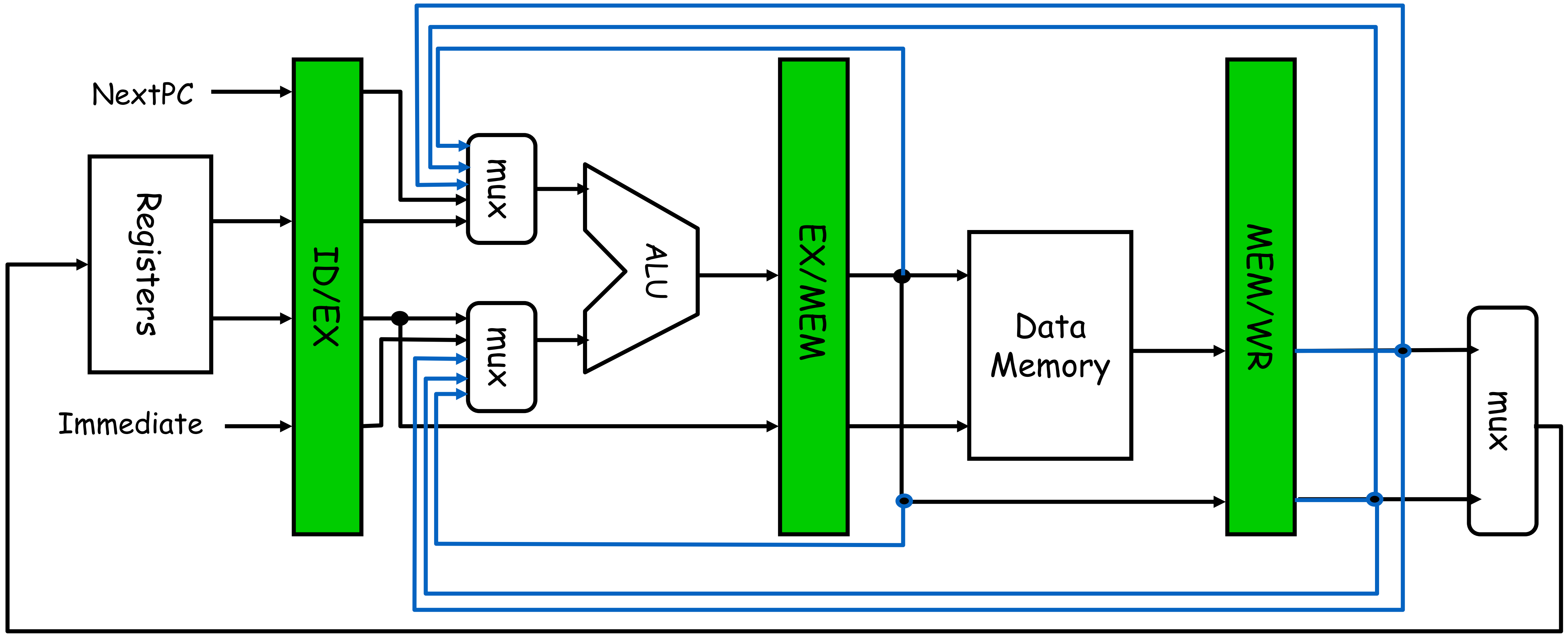
MEM/WB.RegisterRd = ID/EX.RegisterRt

what about instructions do not write into the registers?

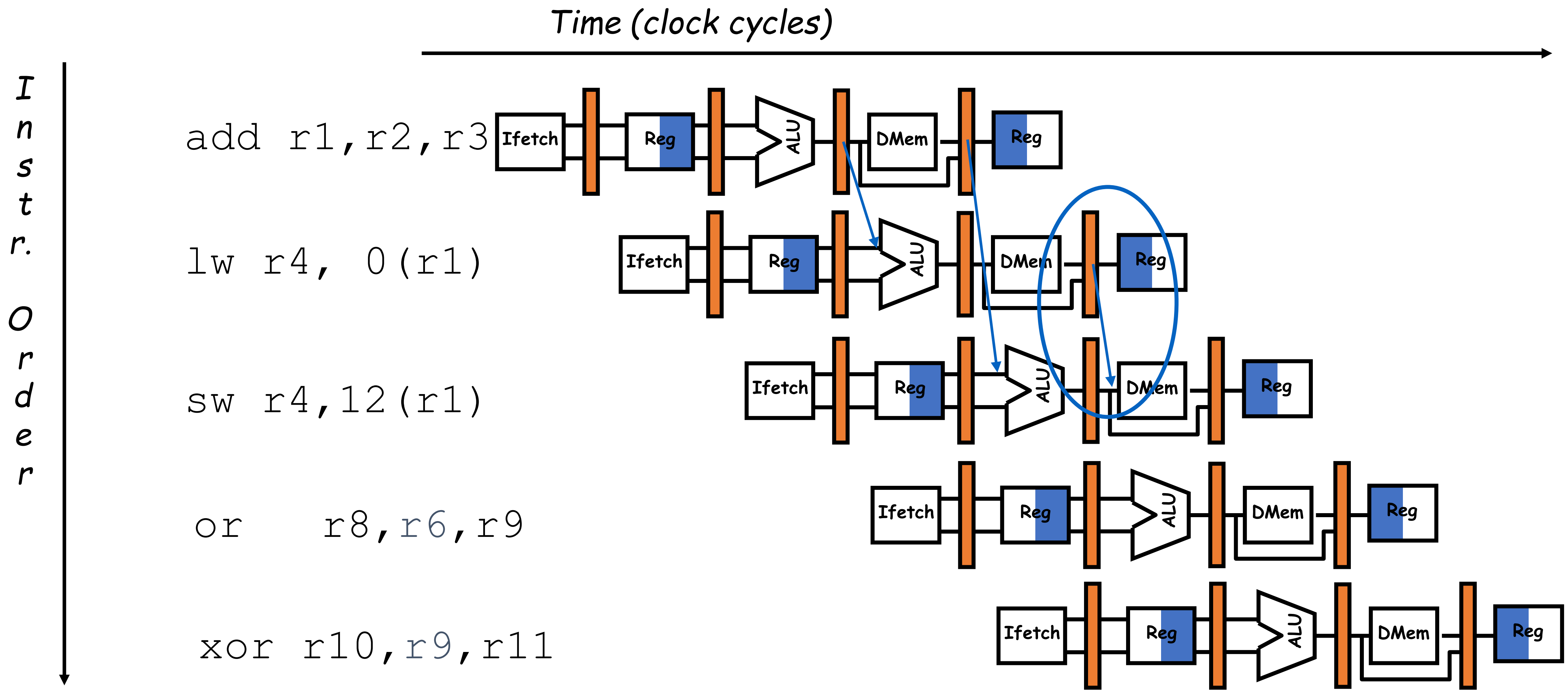
Bypassing

Route data as soon as possible after it is calculated to the earlier pipeline stage

Bypassing/forwarding: Updated Datapath



How does it help?

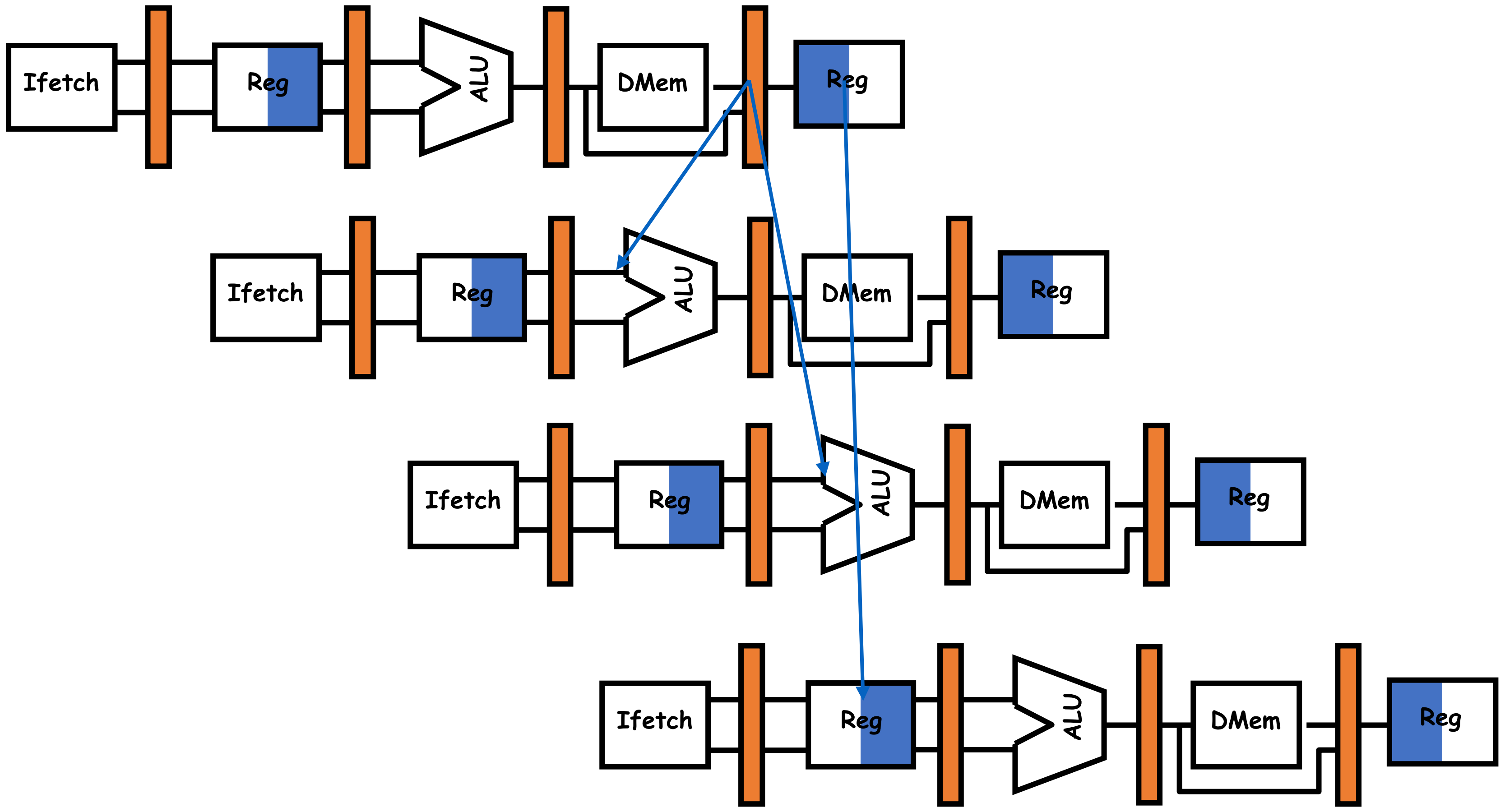


Does it help always?

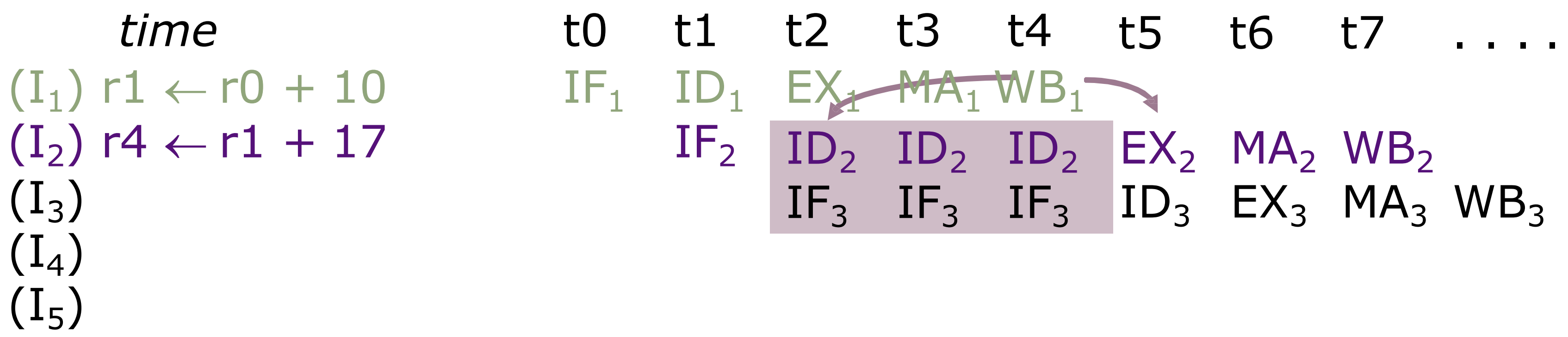
Time (clock cycles)

Instr. Order

```
lw r1, 0(r2)

sub r4,r1,r6

and r6,r1,r7

or   r8,r1,r9
```



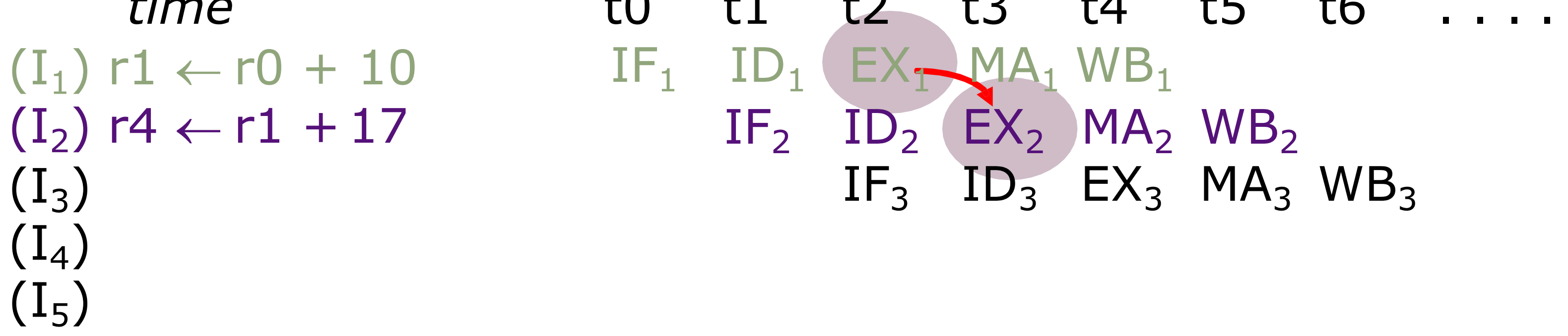
Bypassing: Visualizing Pipeline



time	t0	t1	t2	t3	t4	t5	t6	t7	. . . .
(I_1) $r1 \leftarrow r0 + 10$	IF_1	ID_1	EX_1	MA_1	WB_1				
(I_2) $r4 \leftarrow r1 + 17$		IF_2	ID_2	ID_2	ID_2	EX_2	MA_2	WB_2	
(I_3)			IF_3	IF_3	IF_3	ID_3	EX_3	MA_3	WB_3
(I_4)									
(I_5)									

Each *stall or kill* introduces a bubble $\Rightarrow$ *CPI > 1*

When is data actually available? At Execute



time	t0	t1	t2	t3	t4	t5	t6	. . . .
(I_1) $r1 \leftarrow r0 + 10$	IF_1	ID_1	EX_1	MA_1	WB_1			
(I_2) $r4 \leftarrow r1 + 17$		IF_2	ID_2	EX_2	MA_2	WB_2		
(I_3)			IF_3	ID_3	EX_3	MA_3	WB_3	
(I_4)								
(I_5)								

A new datapath, i.e., *a bypass*, can get the data from the output of the ALU to its input. Note that bypassing does not mitigate control hazards

What and Where? Control Hazard

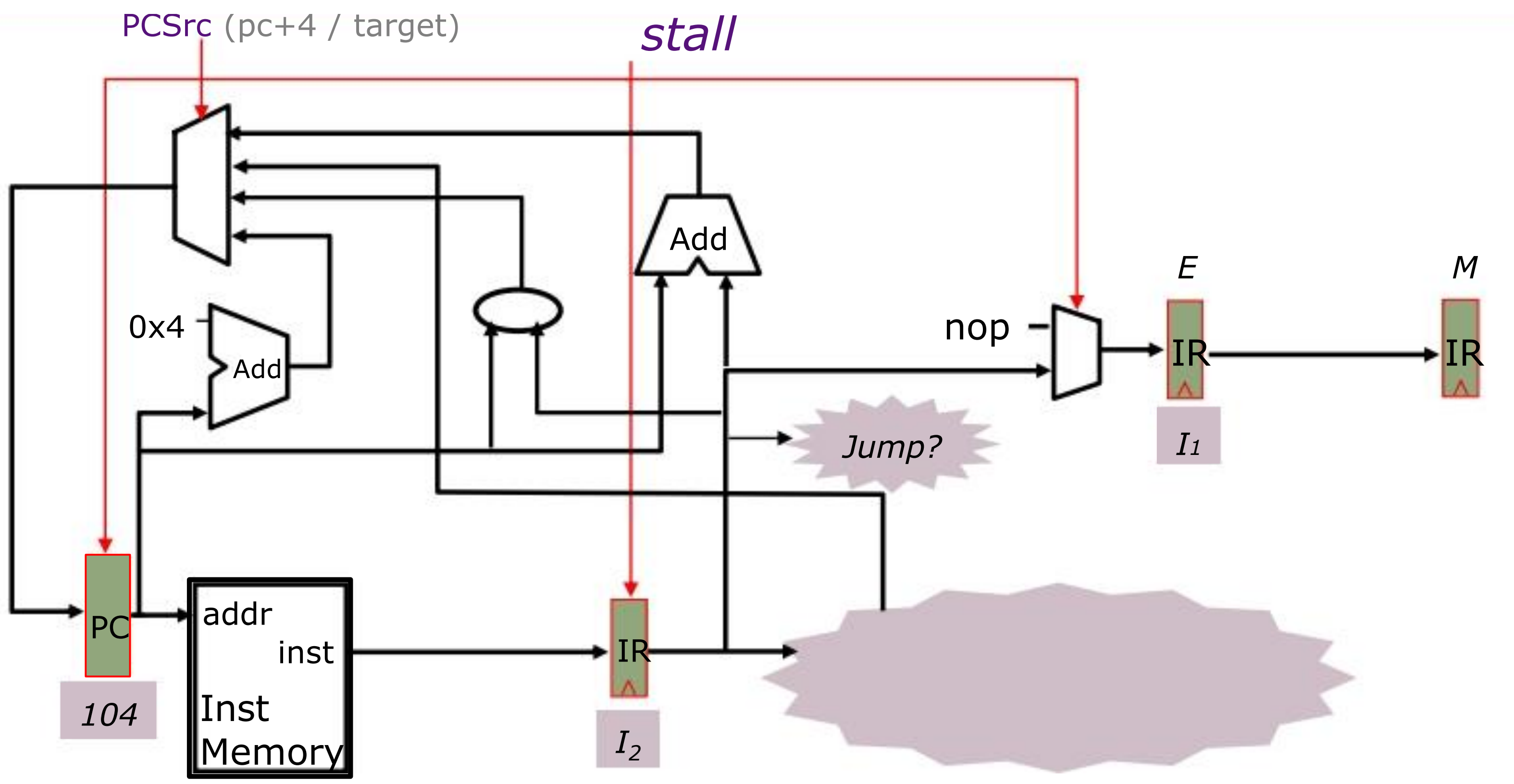
What do we need to calculate next PC?

- For Jumps
  - Opcode, offset, and PC
- For Jump Register
  - Opcode and register value
- For Conditional Branches
  - Opcode, offset, PC, and register (for condition)
- For all others
  - Opcode and PC

In what stage do we know these?

- PC - Fetch
- Opcode, offset - Decode (or Fetch?)
- Register value - Decode
- Branch condition ((rs)==0) - Execute (or Decode?)

Speculate, PC=PC+4



I1	096	ADD
I2	100	J304
I3	~~104~~	~~ADD~~
I4	304	ADD

What happens on mis-speculation, i.e., when next instruction is not PC+4?

kill *How? Insert NOPs*

Conditional branches

I_1	096	ADD
I_2	100	BEQZ r1 200
I_3	104	ADD
I_4	304	ADD

Branch condition is not known until the execute stage

Instructions between a branch instruction and the target are in the wrong-path if the branch is not taken

Again (stalls/NOPs)

	time t0	t1	t2	t3	t4	t5	t6	t7	. . . .
(I_1) 096: ADD	IF_1	ID_1	EX_1	MA_1	WB_1				
(I_2) 100: BEQZ 200		IF_2	ID_2	EX_2	MA_2	WB_2			
(I_3) 104: ADD			IF_3	ID_3	nop	nop	nop		
(I_4) 108:				IF_4	nop	nop	nop	nop	
(I_5) 304: ADD					IF_5	ID_5	EX_5	MA_5	WB_5

Resource Usage

	time t0	t1	t2	t3	t4	t5	t6	t7	. . . .
IF	I_1	I_2	I_3	I_4	I_5				
ID		I_1	I_2	I_3	nop	I_5			
EX			I_1	I_2	nop	nop	I_5		
MA				I_1	I_2	nop	nop	I_5	
WB					I_1	I_2	nop	nop	I_5

Branches: Taken/Not Taken and Target

Instruction	*Taken known?*	*Target known?*
J	After Inst. Decode	After Inst. Decode
BEQZ/BNEZ	After Inst. Execute	After Inst. Execute

what action should be taken in the decode stage?

Can we add an ALU in the decode stage?

Branches: Taken/Not Taken and Target

Instruction	*Taken known?*	*Target known?*
J	After Inst. Decode	After Inst. Decode
BEQZ/BNEZ	After Inst. Decode	After Inst. Execute

Assumption that the decode stage has an ALU (comparator)

Takk