CS2100 COMPUTER ORGANISATION

http://www.comp.nus.edu.sg/~cs2100/

CS2100
Recitation 11

Pipelining

7 April 2025

Aaron Tan

NUS National University of Singapore | School of Computing

Contents

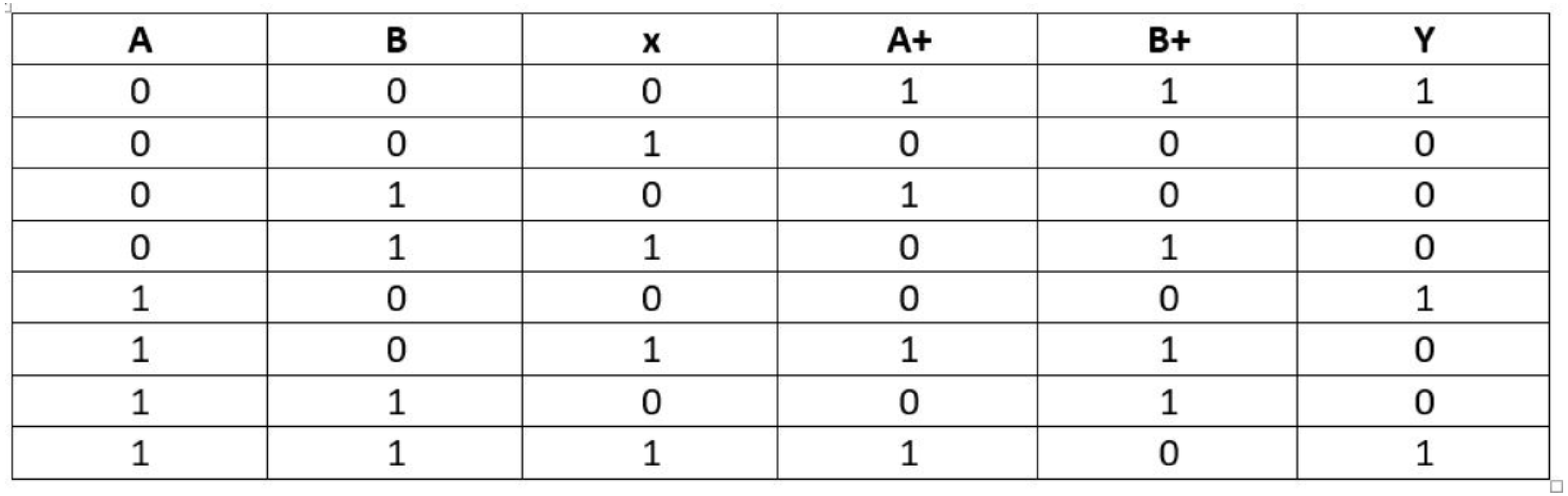
QUIZZES

Sequential Circuits Quiz 1

Sequential Circuits Quiz 2

Sequential Circuits Quiz 1

In this quiz we are given the following table:



A	B	x	A+	B+	Y
0	0	0	1	1	1
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	1	1	0
1	1	0	0	1	0
1	1	1	1	0	1

Sequential Circuits Quiz 1 Question 1

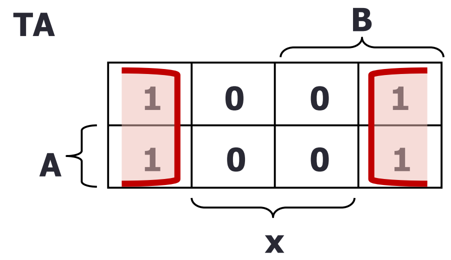
Suppose we use T flip-flops (FFs) to implement the sequential circuit. Choose the most simplified FF input equations and sequential circuit output equations from below:

○ TA = B'.x' + B.x', TB = A'.B'.x' + A.x + A'.B.x', y = B'.x' + A.B.x

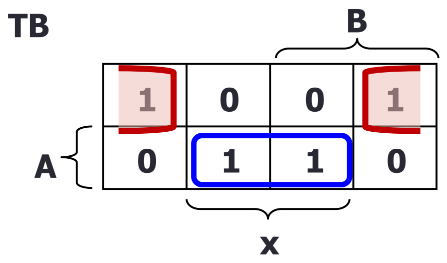
○ TA = A'.x' + A.x, TB = A'.x' + A.x, y = B'.x' + A.B.x

○ [✓ TA = x'. TB = A'.x' + A.x, y = B'.x' + A.B.x]

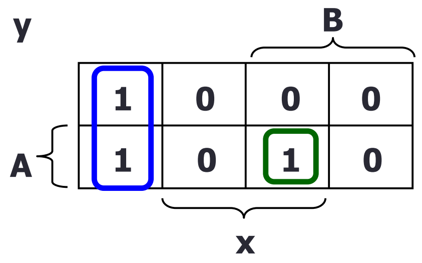
○ TA = x' + B'.x, TB = A'.B'.x' + A.x, y = B'.x' + A.B.x



TA				
	1	0	0	1
A	1	0	0	1



TB				
	1	0	0	1
A	0	1	1	0



y				
	1	0	0	0
A	1	0	1	0

Augmented State Table:

A	B	x	A+	B+	Y	TA	TB
0	0	0	1	1	1	1	1
0	0	1	0	0	0	0	0
0	1	0	1	0	0	1	1
0	1	1	0	1	0	0	0
1	0	0	0	0	1	1	0
1	0	1	1	1	0	0	1
1	1	0	0	1	0	1	0
1	1	1	1	0	1	0	1

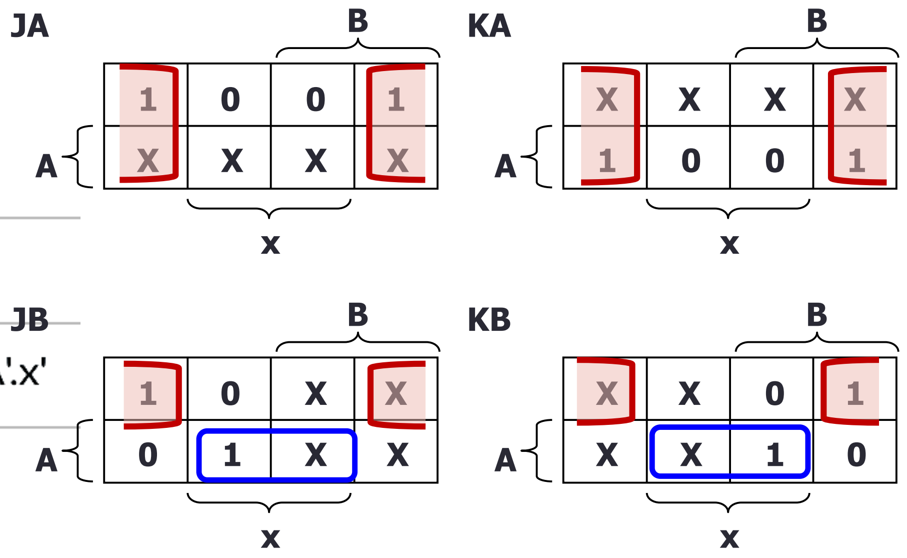
Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

T Flip-flop

Sequential Circuits Quiz 1 Question 2

Now suppose we use JK flip-flops instead. Choose the most simplified FF input equations from below:
(Note: a XNOR b = (a XOR b)').

- ○ JA = x', KA = x', JB = A xor x, KB = A xor x
- [✓ ○ JA = x', KA = x', JB = A xnor x, KB = A xnor x]
- ○ JA = A.x', KA = A.x', JB = A.x, + A'.x' KB = A.x + A'.x'
- ○ JA = x', KA = x', JB = A.x', KB = A.x + A'.x'



Augmented State Table:

A	B	x	A+	B+	Y	JA	KA	JB	KB
0	0	0	1	1	1	1	X	1	X
0	0	1	0	0	0	0	X	0	X
0	1	0	1	0	0	1	X	X	1
0	1	1	0	1	0	0	X	X	0
1	0	0	0	0	1	X	1	0	X
1	0	1	1	1	0	X	0	1	X
1	1	0	0	1	0	X	1	X	0
1	1	1	1	0	1	X	0	X	1

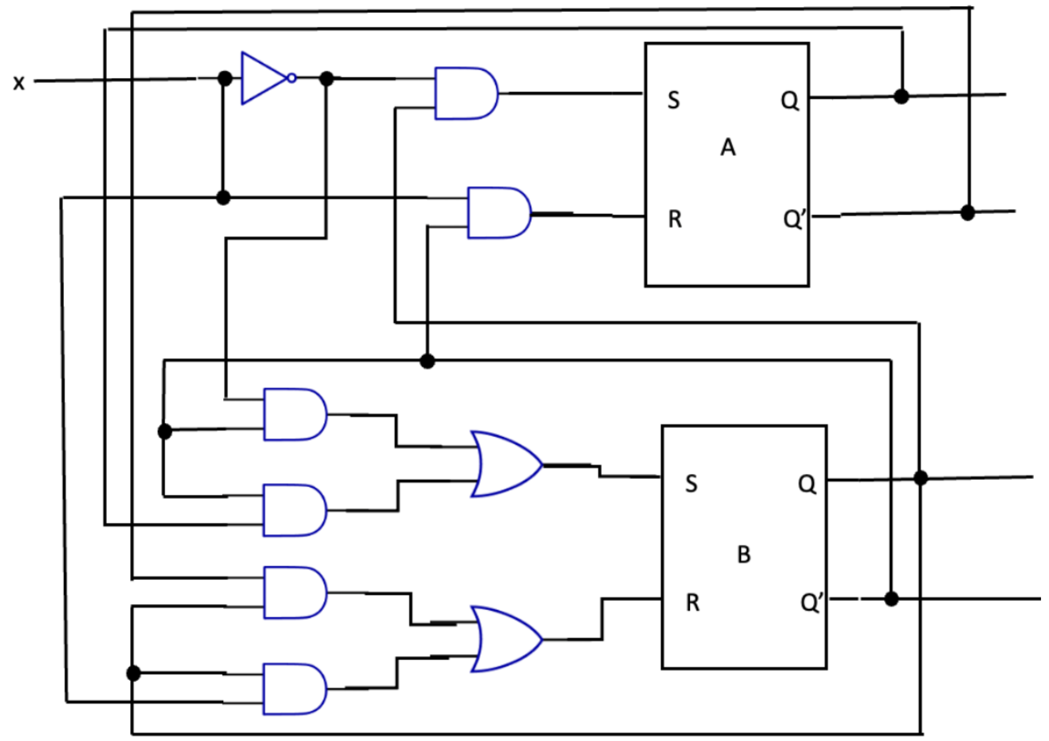
Q	*Q*$^+$	*J*	*K*
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

JK Flip-flop

QnA: Can you please go through Sequential Circuits Quiz 2 more thoroughly? I very get confused with it.

Sequential Circuits Quiz 2

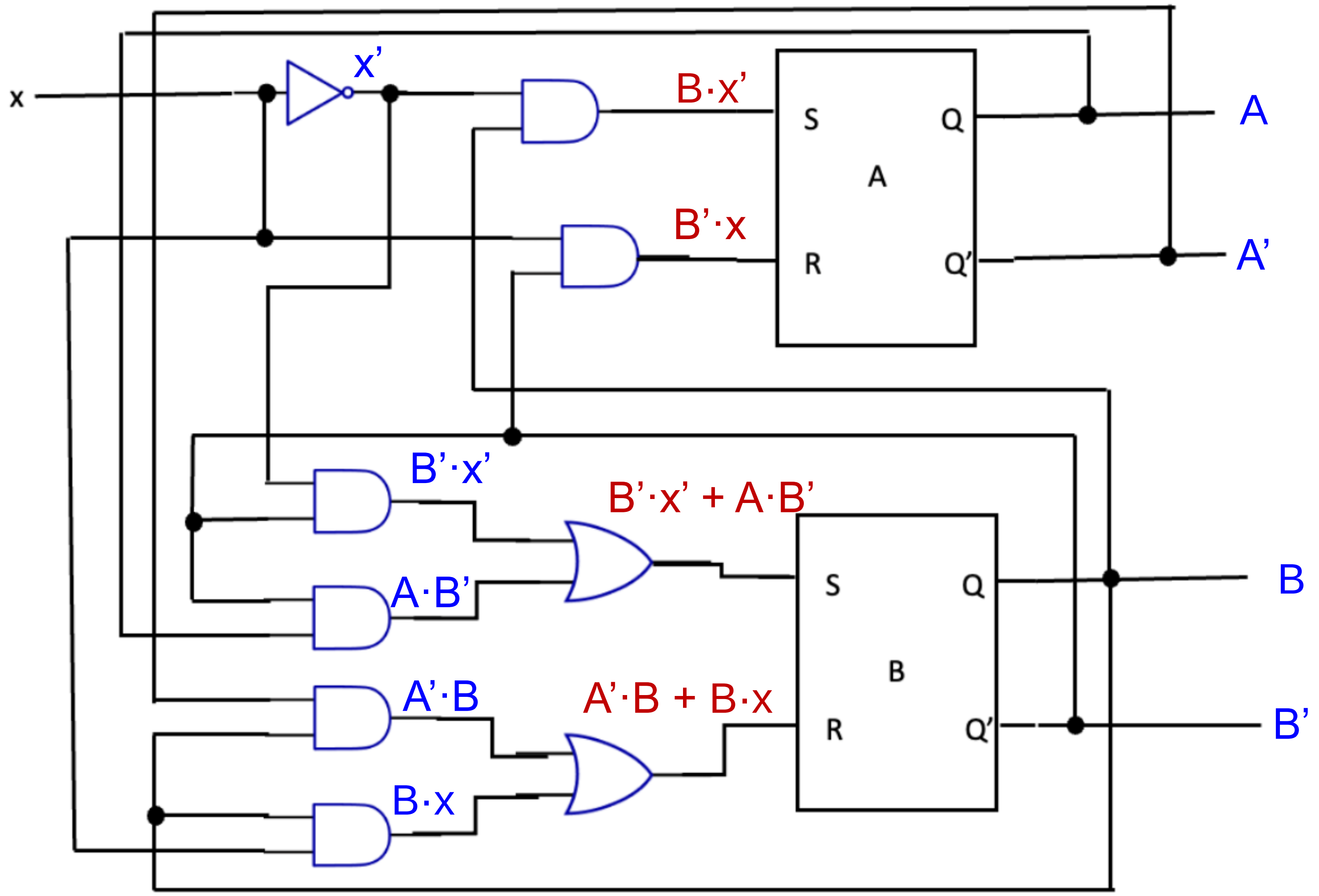
In this quiz we are given the following circuit:



Choose the statement that best describes this circuit:

- ○ This circuit counts from 0 to 3 and staying at 3 when x = 0, and down from 3 to 0 staying at 0 when x = 1
- ○ This circuit counts from 0 to 3 and staying at 3 when x = 1, and down from 3 to 0 staying at 0 when x = 0
- ○ This circuit counts from 0 to 3 rolling over back to 0 when x = 0, and down from 3 to 0 rolling over back to 3 when x = 1
- ○ This circuit counts from 0 to 3 rolling over back to 0 when x = 1, and down from 3 to 0 rolling over back to 3 when x = 0

Sequential Circuits Quiz 2 Question 1

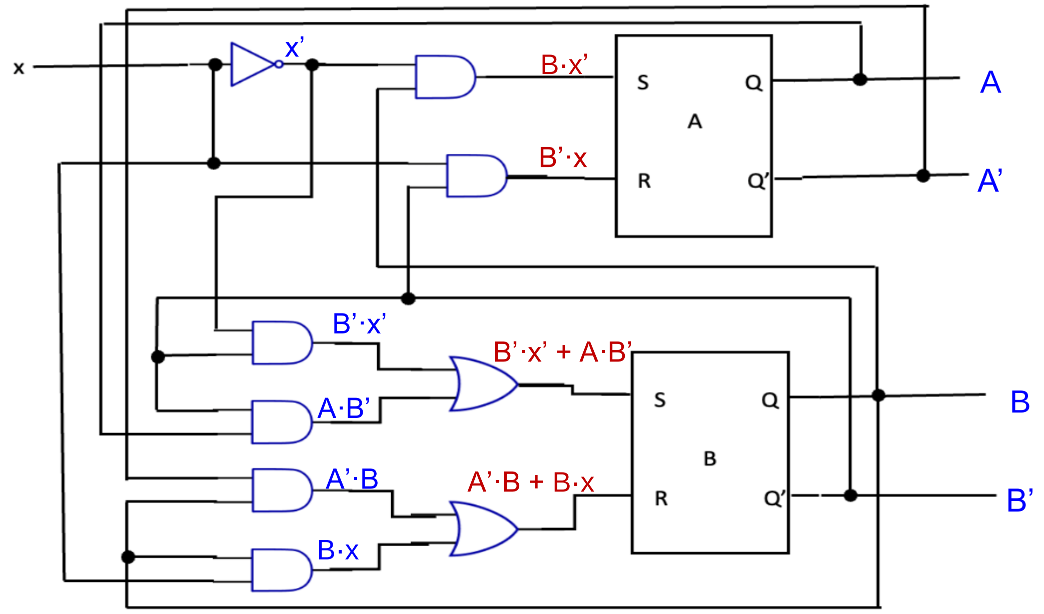


Sequential Circuits Quiz 2 Question 1

State Table

A	B	x	SA = B.x'	RA = B'.x	SB = B'.x' + A.B'	RB = A'.B + B.x	A+	B+
0	0	0						
0	0	1						
0	1	0						
0	1	1						
1	0	0						
1	0	1						
1	1	0						
1	1	1						

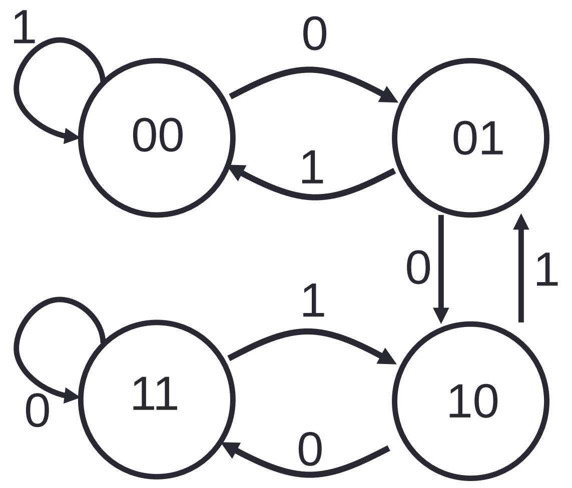
S	*R*	*Q(t+1)*	Comments
0	0	*Q(t)*	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Unpredictable



State Table

A	B	x	SA = B.x'	RA = B'.x	SB = B'.x' + A.B'	RB = A'.B + B.x	A+	B+
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	0	0
0	1	0	1	0	0	1	1	0
0	1	1	0	0	0	1	0	0
1	0	0	0	0	1	0	1	1
1	0	1	0	1	1	0	0	1
1	1	0	1	0	0	0	1	1
1	1	1	0	0	0	1	1	0

State Diagram


$$\begin{tikzcd}
00 \arrow[loop left, "1"] \arrow[r, bend left, "0"] & 01 \arrow[l, bend left, "1"] \arrow[d, shift right, "0"'] \\
11 \arrow[loop left, "0"] \arrow[r, bend left, "1"] & 10 \arrow[l, bend left, "0"] \arrow[u, shift right, "1"']
\end{tikzcd}$$

Sequential Circuits Quiz 2 Question 1

Choose the statement that best describes this circuit:

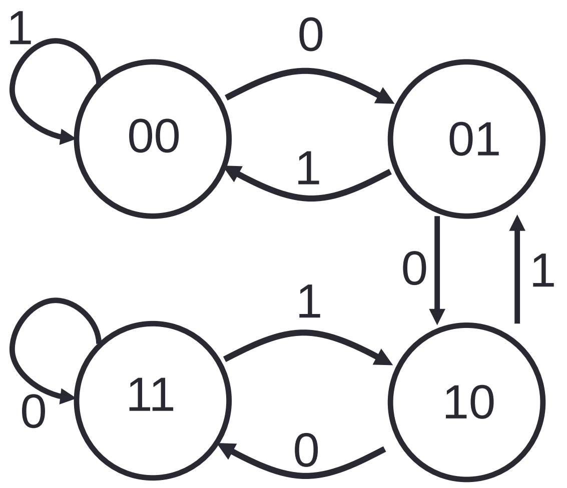
[✓ ○ This circuit counts from 0 to 3 and staying at 3 when x = 0, and down from 3 to 0 staying at 0 when x = 1]

○ This circuit counts from 0 to 3 and staying at 3 when x = 1, and down from 3 to 0 staying at 0 when x = 0

○ This circuit counts from 0 to 3 rolling over back to 0 when x = 0, and down from 3 to 0 rolling over back to 3 when x = 1

○ This circuit counts from 0 to 3 rolling over back to 0 when x = 1, and down from 3 to 0 rolling over back to 3 when x = 0

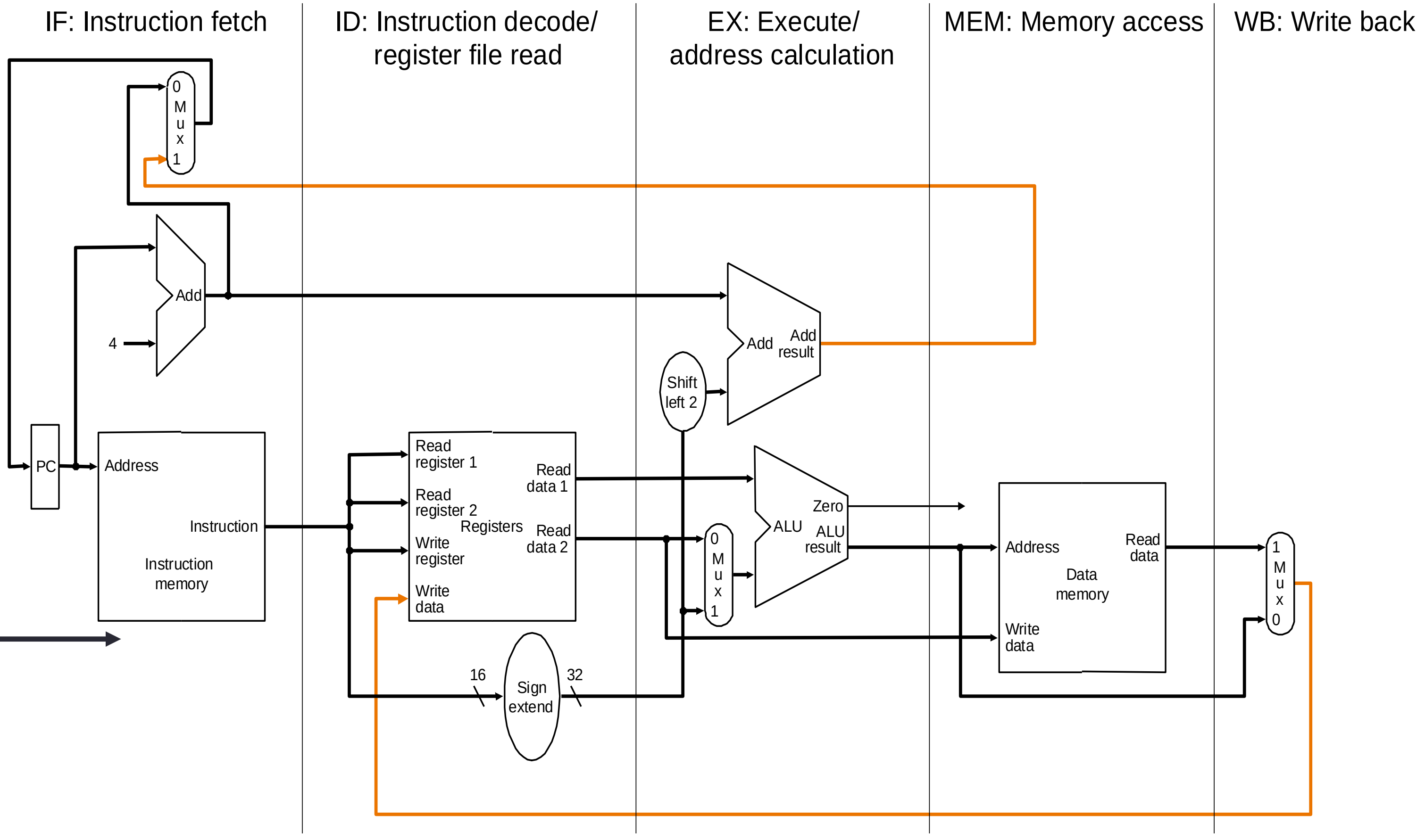
State Diagram

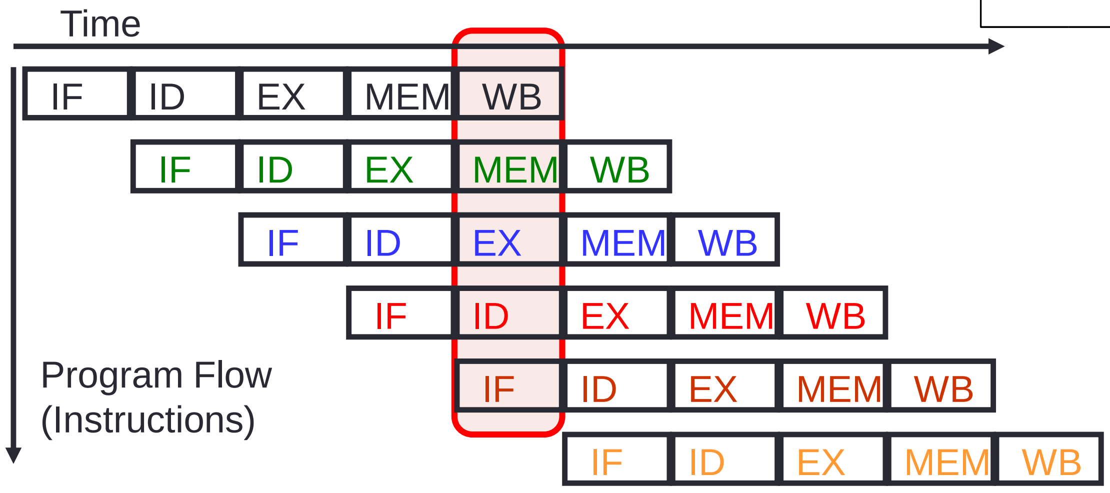


MIPS Pipeline

2. MIPS Pipeline Stages

- **Five** Execution Stages
  - **IF**: Instruction Fetch
  - **ID**: Instruction Decode and Register Read
  - **EX**: Execute an operation or calculate an address
  - **MEM**: Access an operand in data memory
  - **WB**: Write back the result into a register





Several instructions are in the pipeline simultaneously!

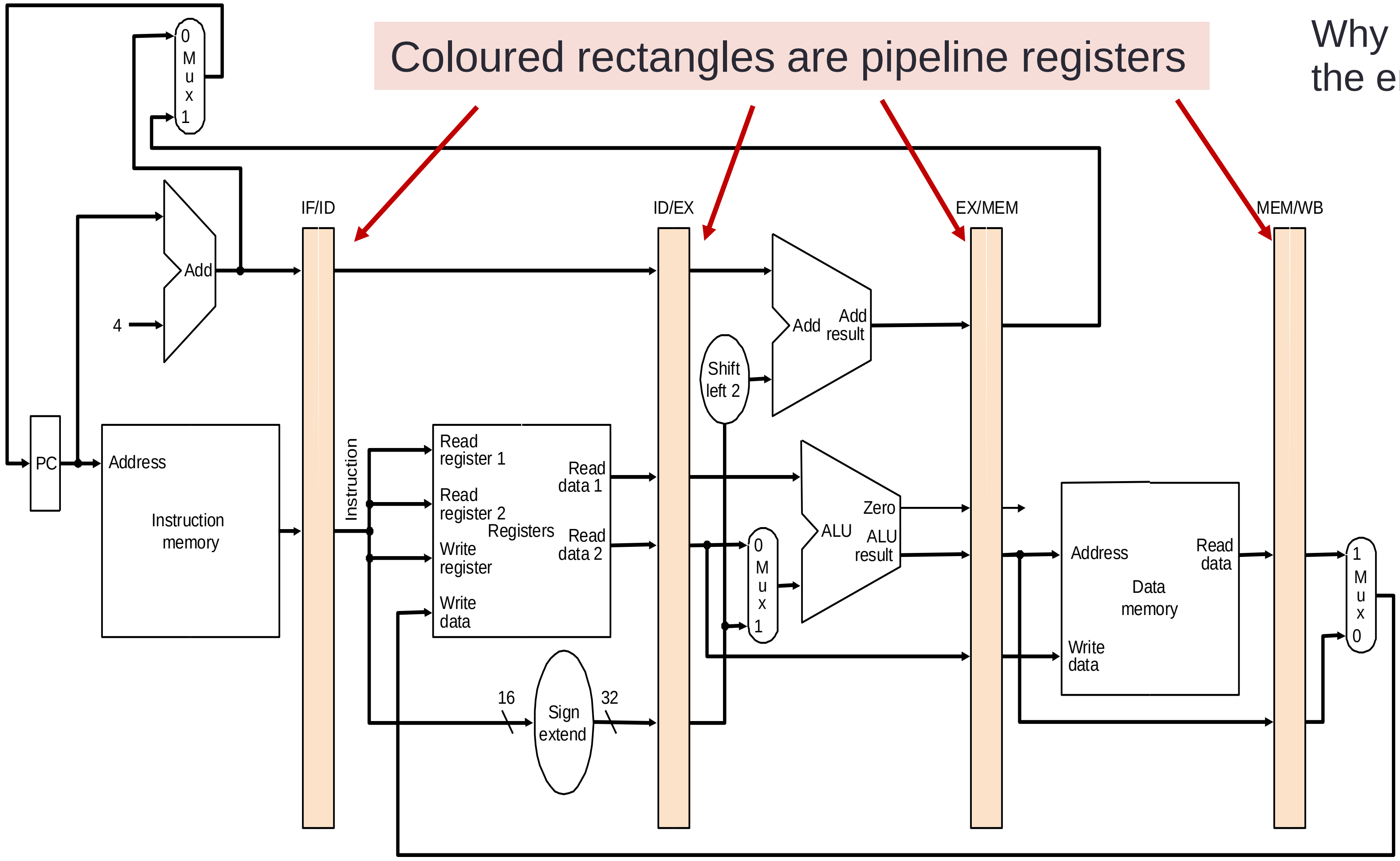
3. MIPS Pipeline: Datapath

- Single-cycle implementation:
  - One cycle per instruction.
  - Update all state elements (**`PC`**, register file, data memory) within a clock cycle.
- Pipelined implementation:
  - One cycle per pipeline stage.
  - Data required for each stage needs to be stored separately (why?)
- Data used by **subsequent instructions:**
  - Store in programmer-visible state elements: **`PC`**, register file and memory
- Data used by **same instruction** in later pipeline stages:
  - Additional registers in datapath called **pipeline registers**
  - **`IF/ID`**: register between **`IF`** and **`ID`**
  - **`ID/EX:`** register between **`ID`** and **`EX`**
  - **`EX/MEM:`** register between **`EX`** and **`MEM`**
  - **`MEM/WB`**: register between **`MEM`** and **`WB`**
- Why no register at the end of **`WB`** stage?

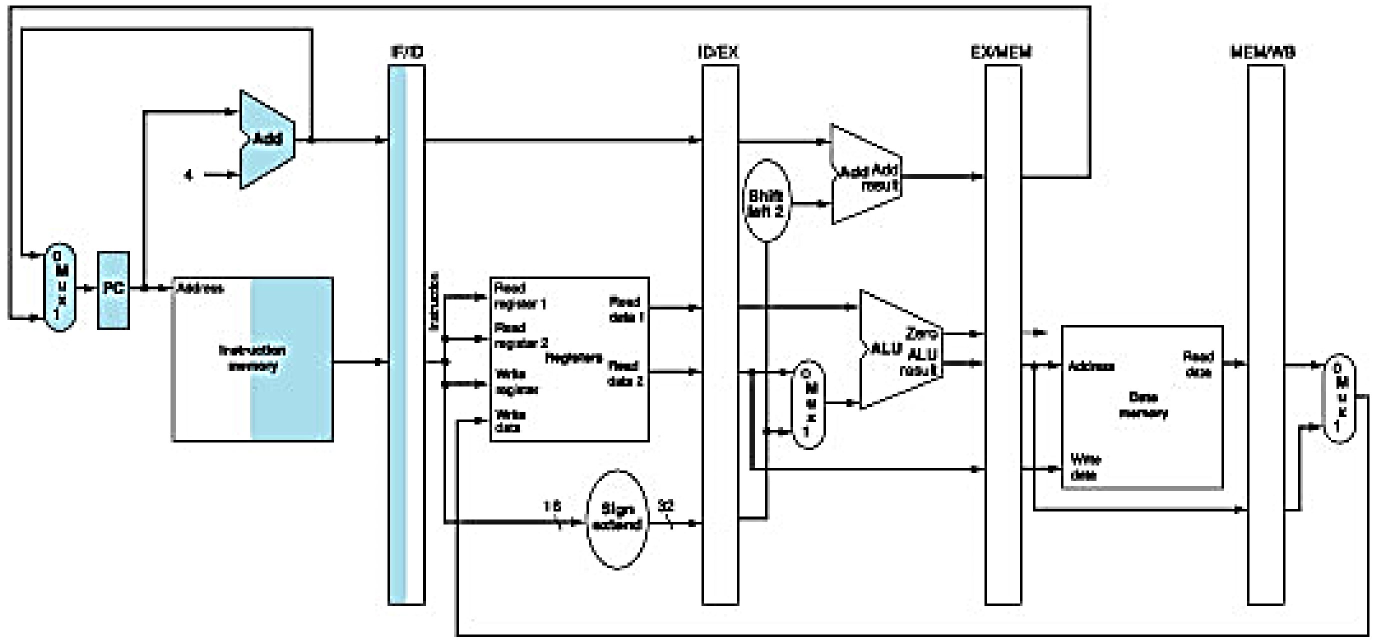
3. MIPS Pipeline: Datapath

Coloured rectangles are pipeline registers

Why no pipeline register at the end of **`WB`** stage?

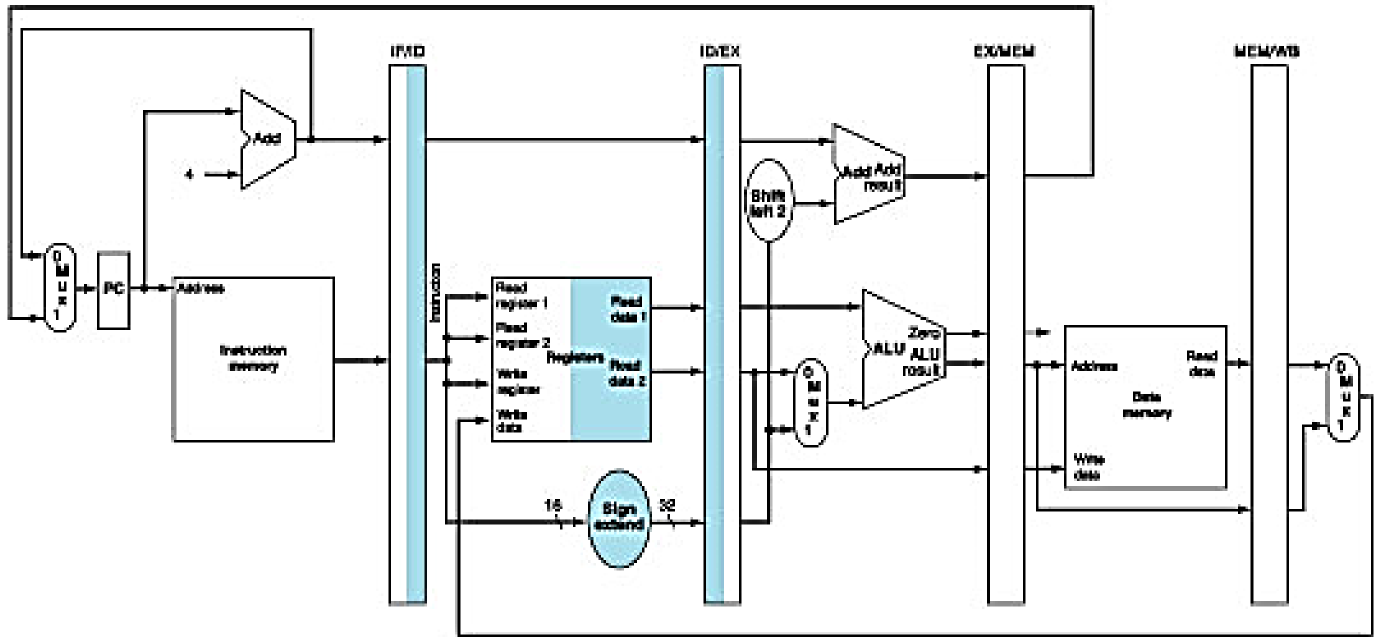


3. Pipeline Datapath: IF Stage



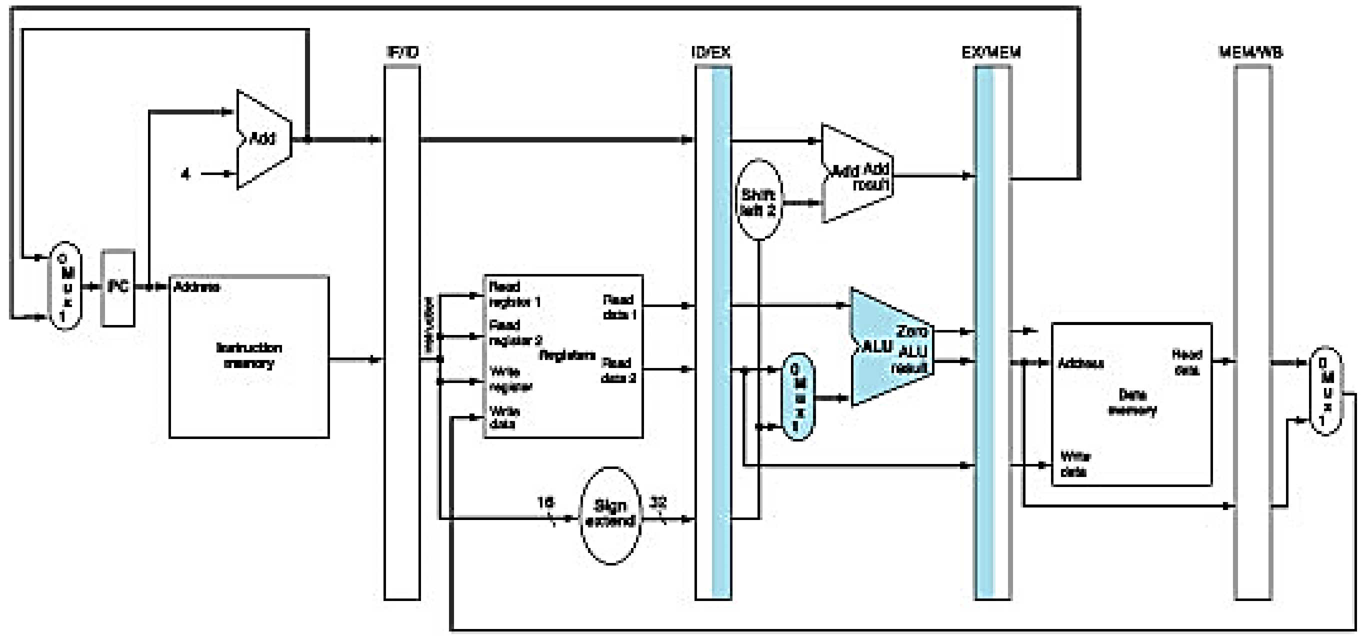
- At the end of a cycle, **IF/ID** receives (stores):
  - Instruction read from InstructionMemory[PC]
  - PC + 4
- PC + 4
  - Also connected to one of the MUX's inputs (another coming later)

3. Pipeline Datapath: ID Stage



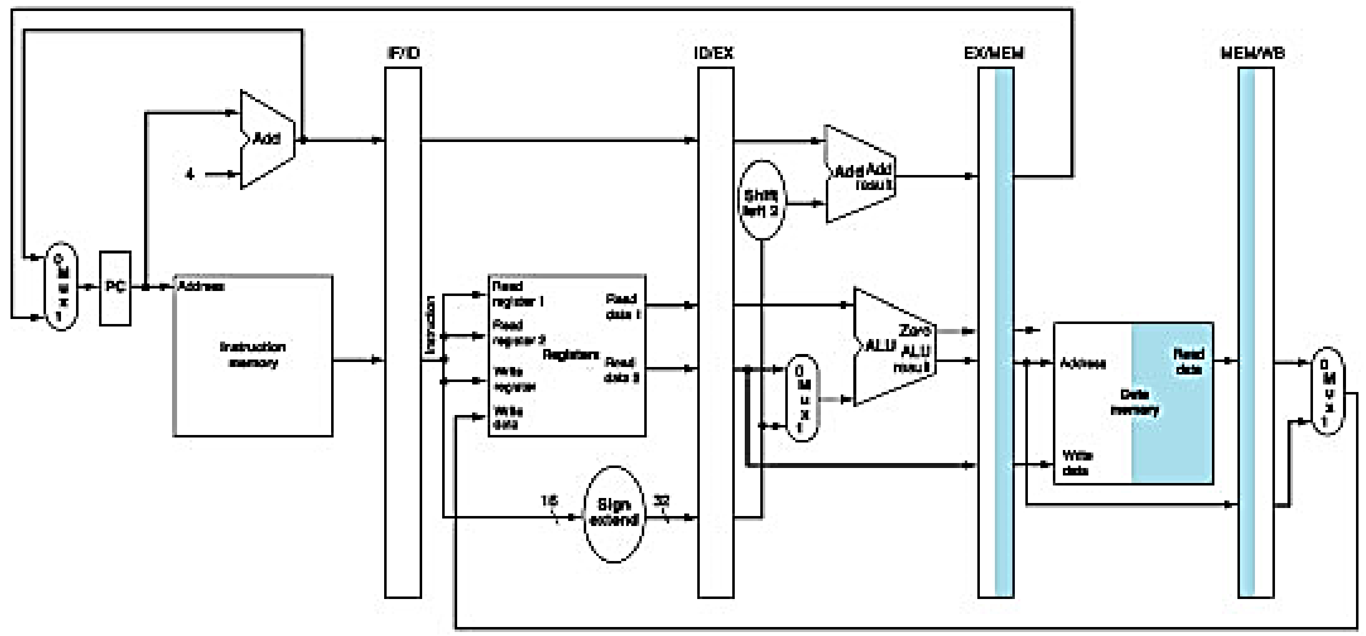
At the beginning of a cycle `IF/ID` register supplies:	At the end of a cycle `ID/EX` receives:
❖ Register numbers for reading two registers ❖ 16-bit offset to be sign-extended to 32-bit ❖ `PC + 4`	❖ Data values read from register file ❖ 32-bit immediate value ❖ `PC + 4`

3. Pipeline Datapath: EX Stage



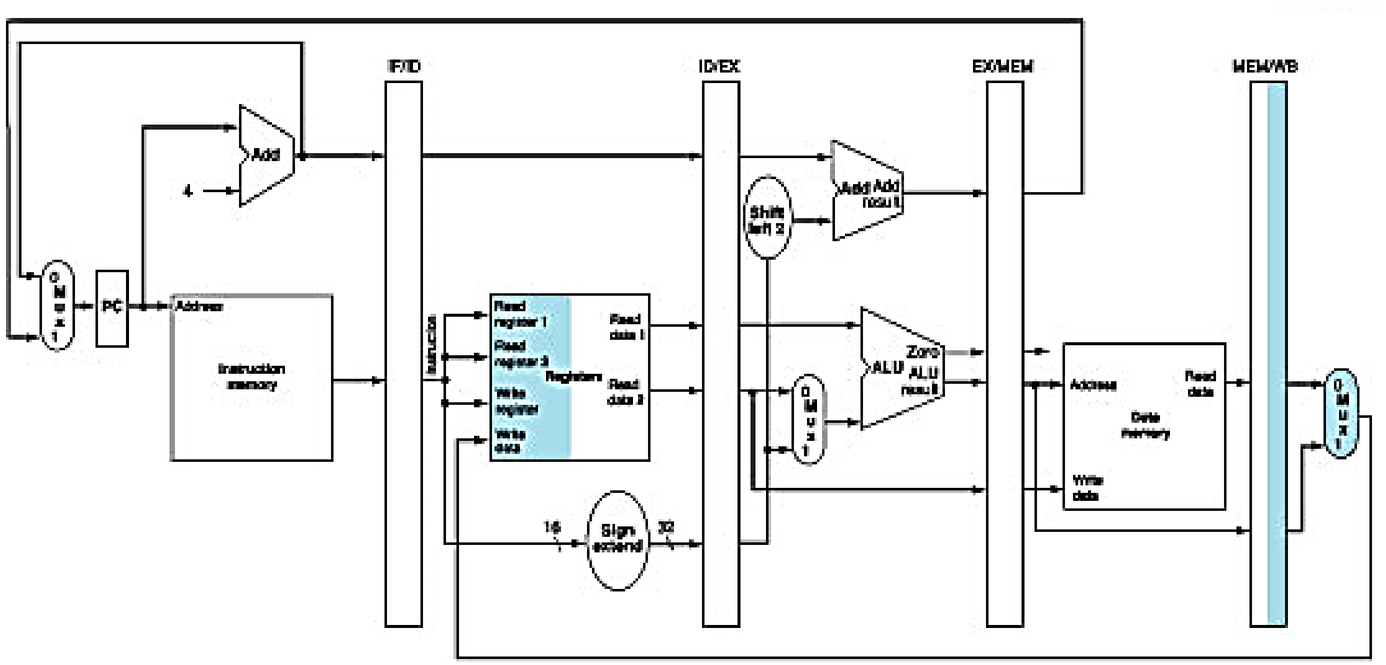
At the beginning of a cycle **ID/EX register supplies:**	At the end of a cycle **EX/MEM receives:**
❖ Data values read from register file ❖ 32-bit immediate value ❖ `PC + 4`	❖ (PC + 4) + (Immediate x 4) ❖ ALU result ❖ `isZero?` signal ❖ Data Read 2 from register file

3. Pipeline Datapath: MEM Stage



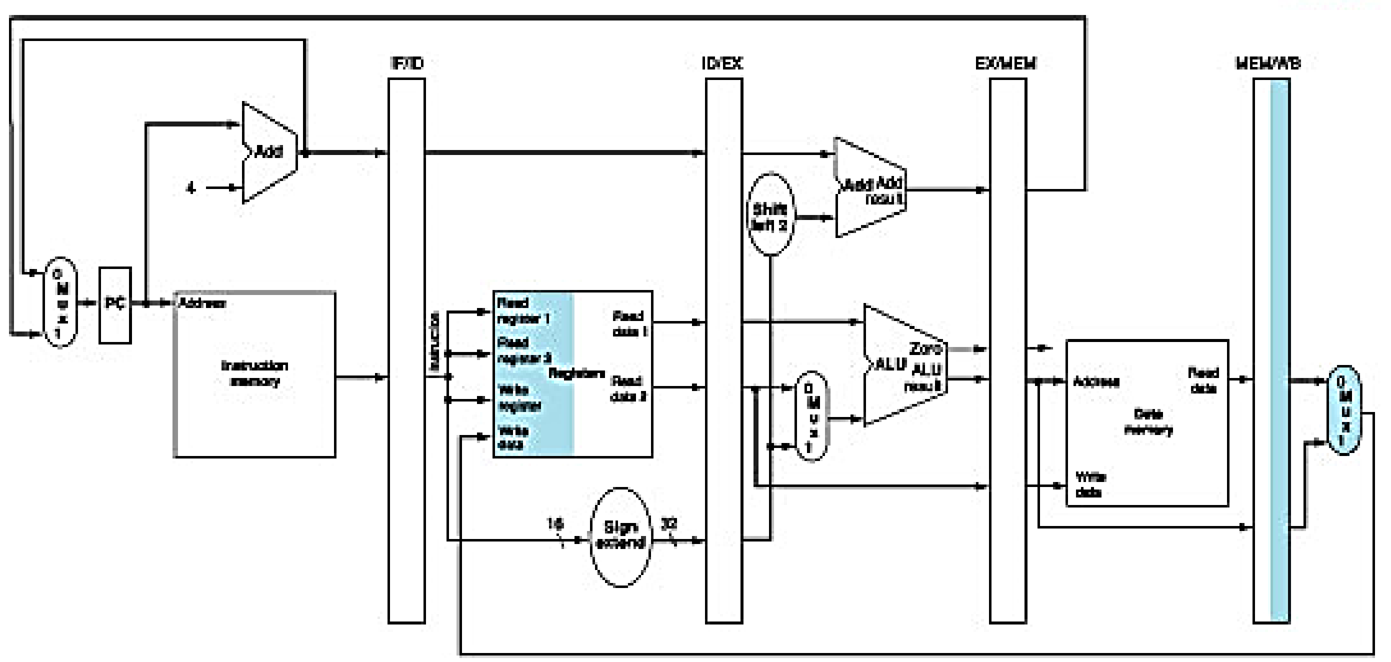
At the beginning of a cycle EX/MEM register supplies:	At the end of a cycle MEM/WB receives:
❖ (PC + 4) + (Immediate x 4) ❖ ALU result ❖ `isZero?` signal ❖ Data Read 2 from register file	❖ ALU result ❖ Memory read data

3. Pipeline Datapath: WB Stage



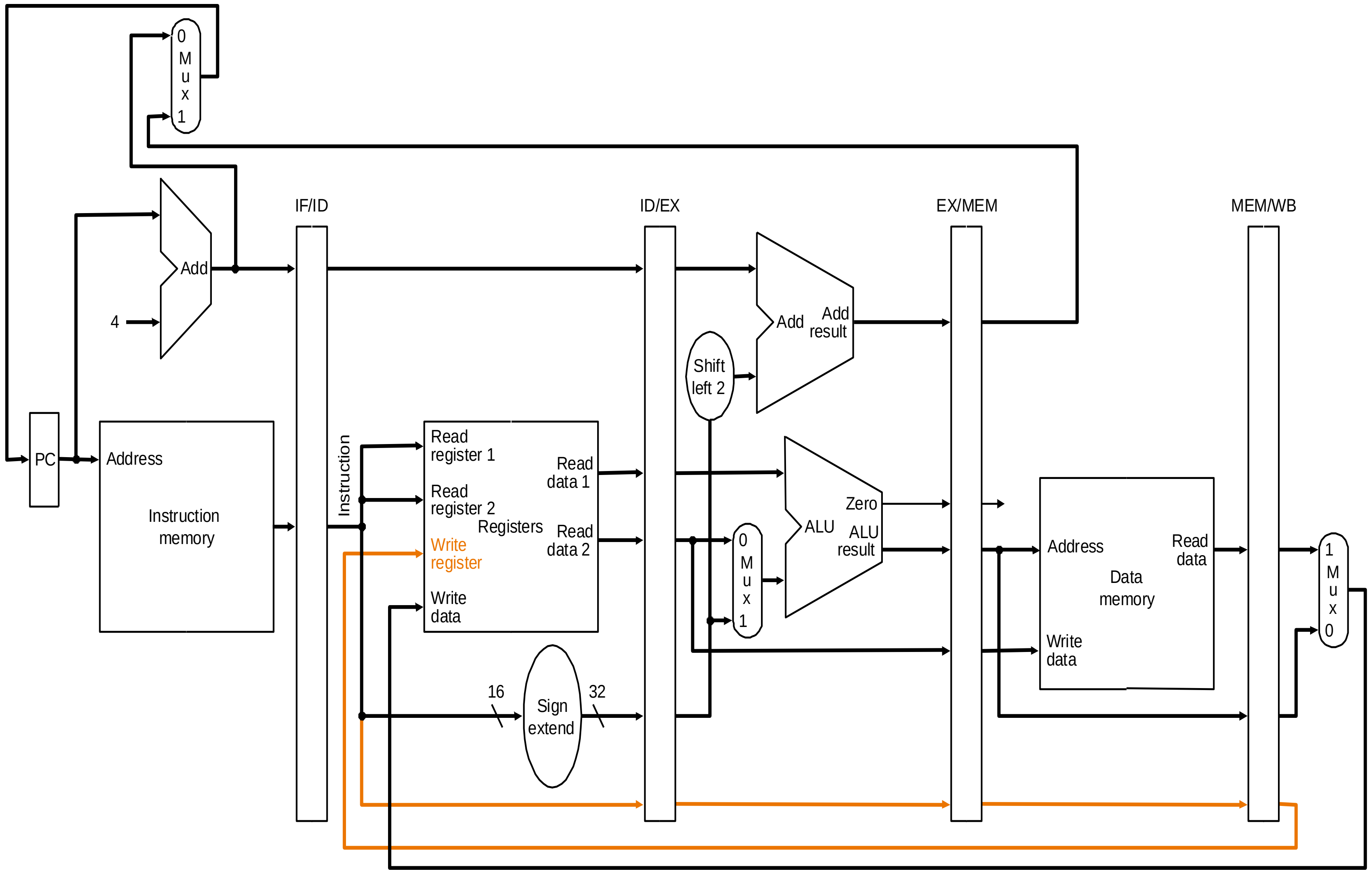
At the beginning of a cycle **MEM/WB register supplies:**	At the end of a cycle
❖ ALU result ❖ Memory read data	❖ Result is written back to register file (if applicable) ❖ **There is a bug here…….**

3. Corrected Datapath (1/2)



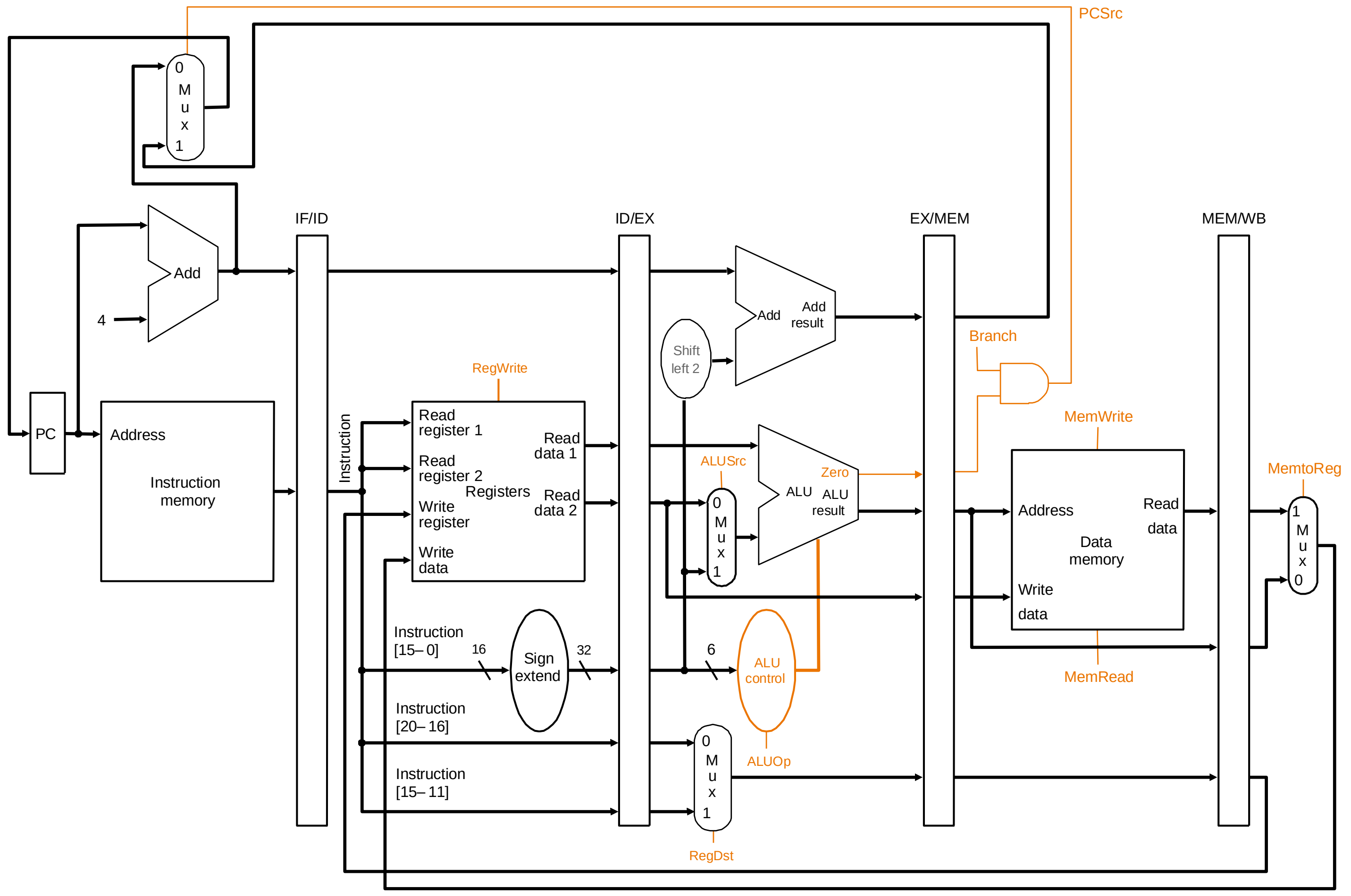
- Observe the “Write register” number
  - Supplied by the **`IF/ID`** pipeline register
  - ➔ It is NOT the correct write register for the instruction now in **`WB`** stage!
- **Solution:**
  - Pass “Write register” number from **`ID/EX`** through **`EX/MEM`** to **`MEM/WB`** pipeline register for use in **`WB`** stage
  - i.e. let the "Write register" number follows the instruction through the pipeline until it is needed in WB stage

3. Corrected Datapath (2/2)



4. Pipeline Control: **Main Idea**

- Same control signals as single-cycle datapath
- **Difference:** Each control signal belongs to a particular pipeline stage



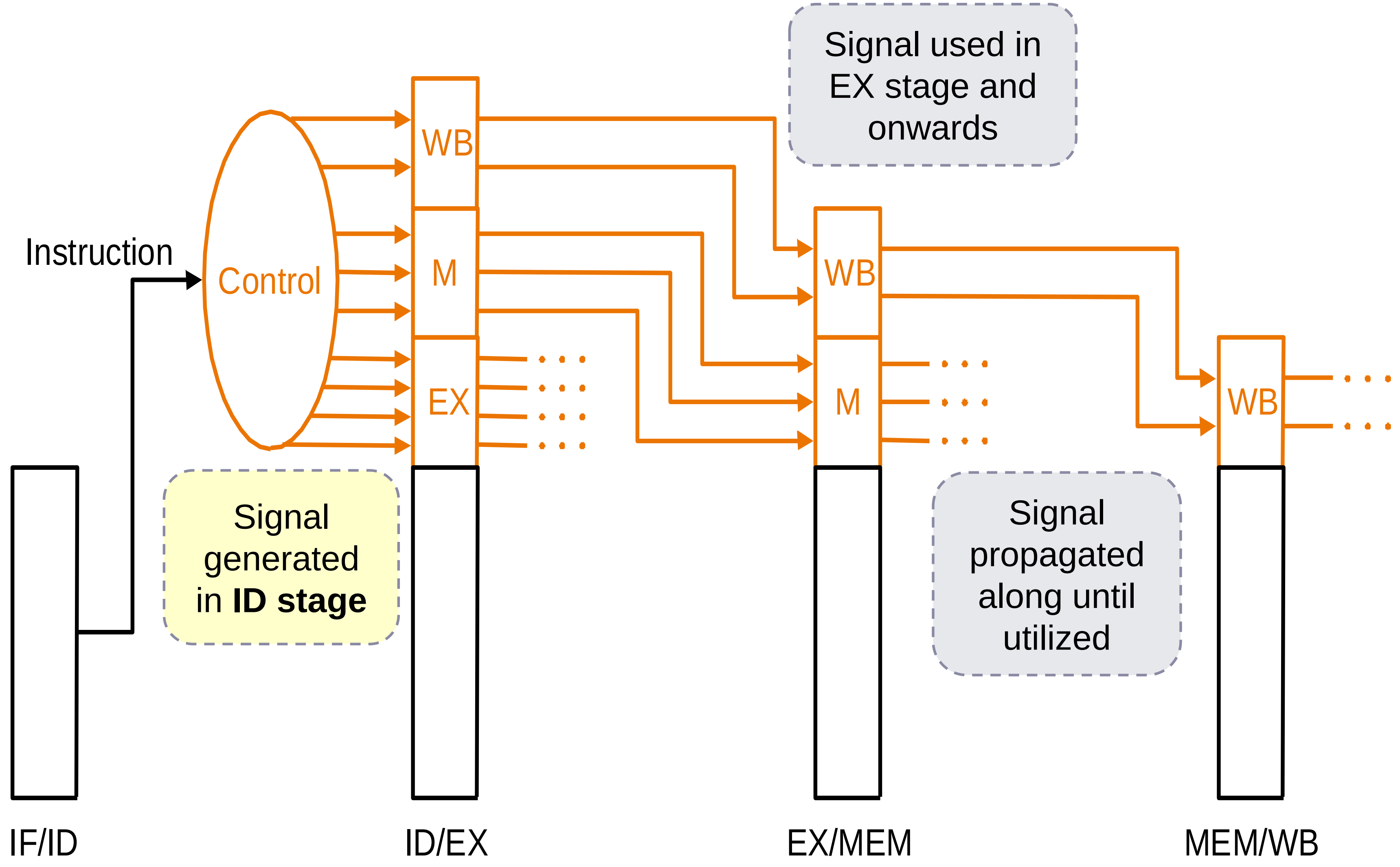
4. Pipeline Control: **Grouping**

- Group control signals according to pipeline stage

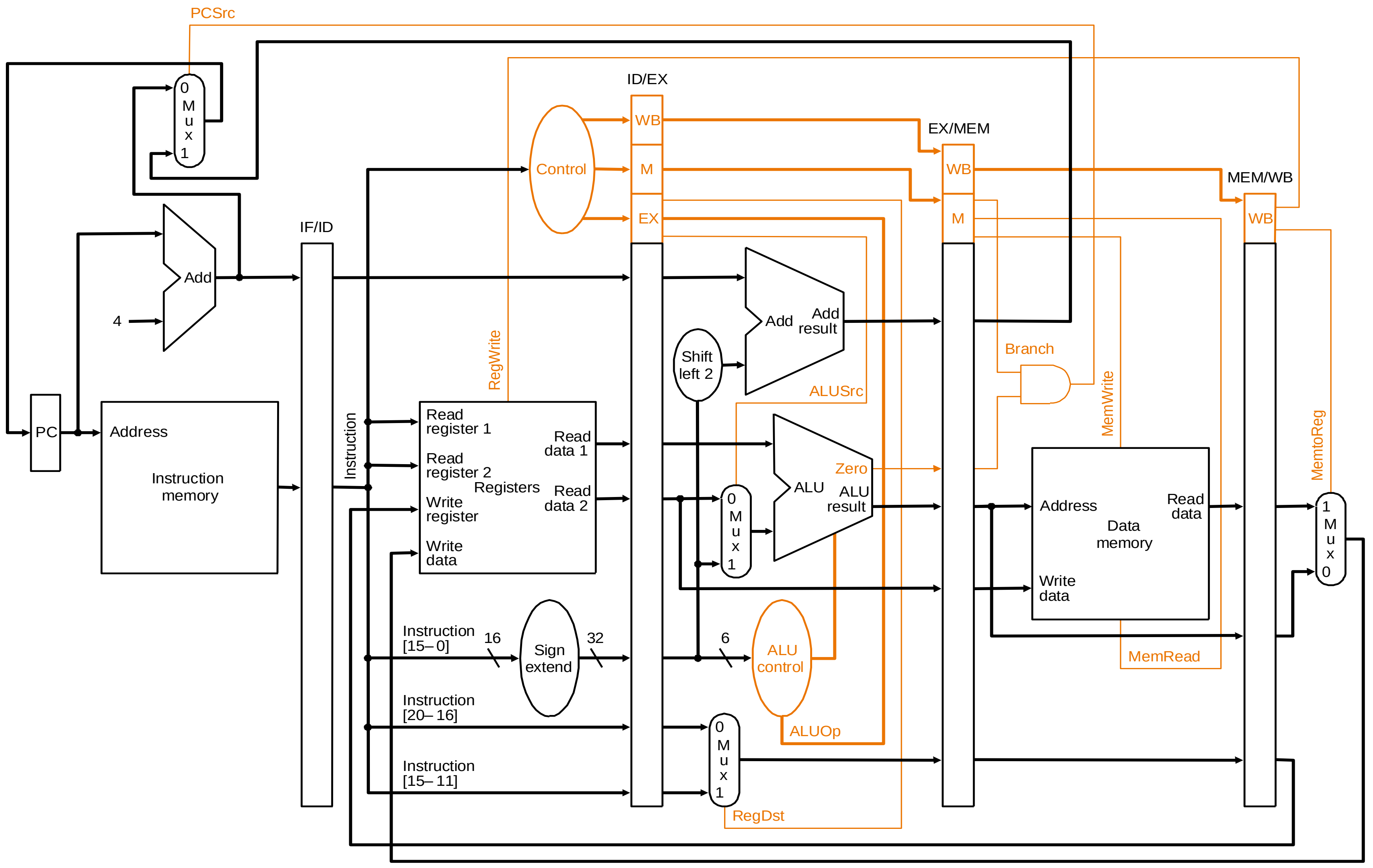
	RegDst	ALUSrc	MemTo Reg	Reg Write	Mem Read	Mem Write	Branch	ALUop	
								op1	op0
R-type	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

	EX Stage				MEM Stage			WB Stage	
	RegDst	ALUSrc	ALUop		Mem Read	Mem Write	Branch	MemTo Reg	Reg Write
			op1	op0					
R-type	1	0	1	0	0	0	0	0	1
lw	0	1	0	0	1	0	0	1	1
sw	X	1	0	0	0	1	0	X	0
beq	X	0	0	1	0	0	1	X	0

4. Pipeline Control: **Implementation**


Signal used in EX stage and onwards
Instruction
Control
WB
M
EX
WB
M
WB
Signal generated in **ID stage**
Signal propagated along until utilized
IF/ID
ID/EX
EX/MEM
MEM/WB

4. Pipeline Control: **Datapath** and **Control**



Ideal Pipeline (i.e., No Hazards)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instr-1	F	D	E	M	W									
Instr-2		F	D	E	M	W								
Instr-3			F	D	E	M	W							
Instr-4				F	D	E	M	W						
Instr-5					F	D	E	M	W					
Instr-6						F	D	E	M	W				
Instr-7							F	D	E	M	W			
Instr-8								F	D	E	M	W		
Instr-9									F	D	E	M	W	
Instr-10										F	D	E	M	W

Number of cycles required = 14 cycles

I = No. of instructions.
N = No. of pipeline stages.
No. of cycles = I + (N – 1).

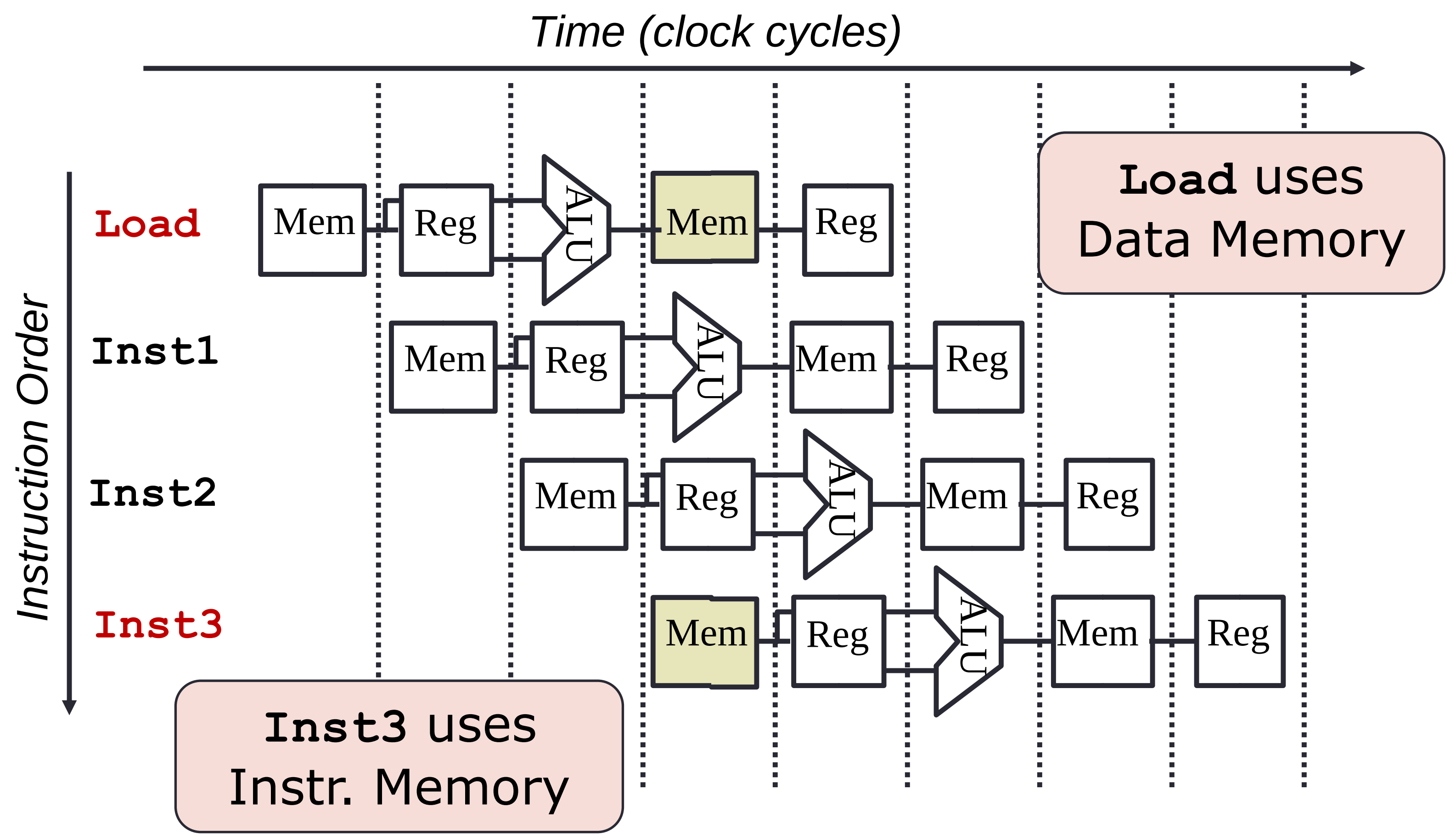
Pipeline Hazards

Pipeline Hazards

- 3 types of pipeline hazards
  - Structural hazards
  - Data hazards
  - Control hazards

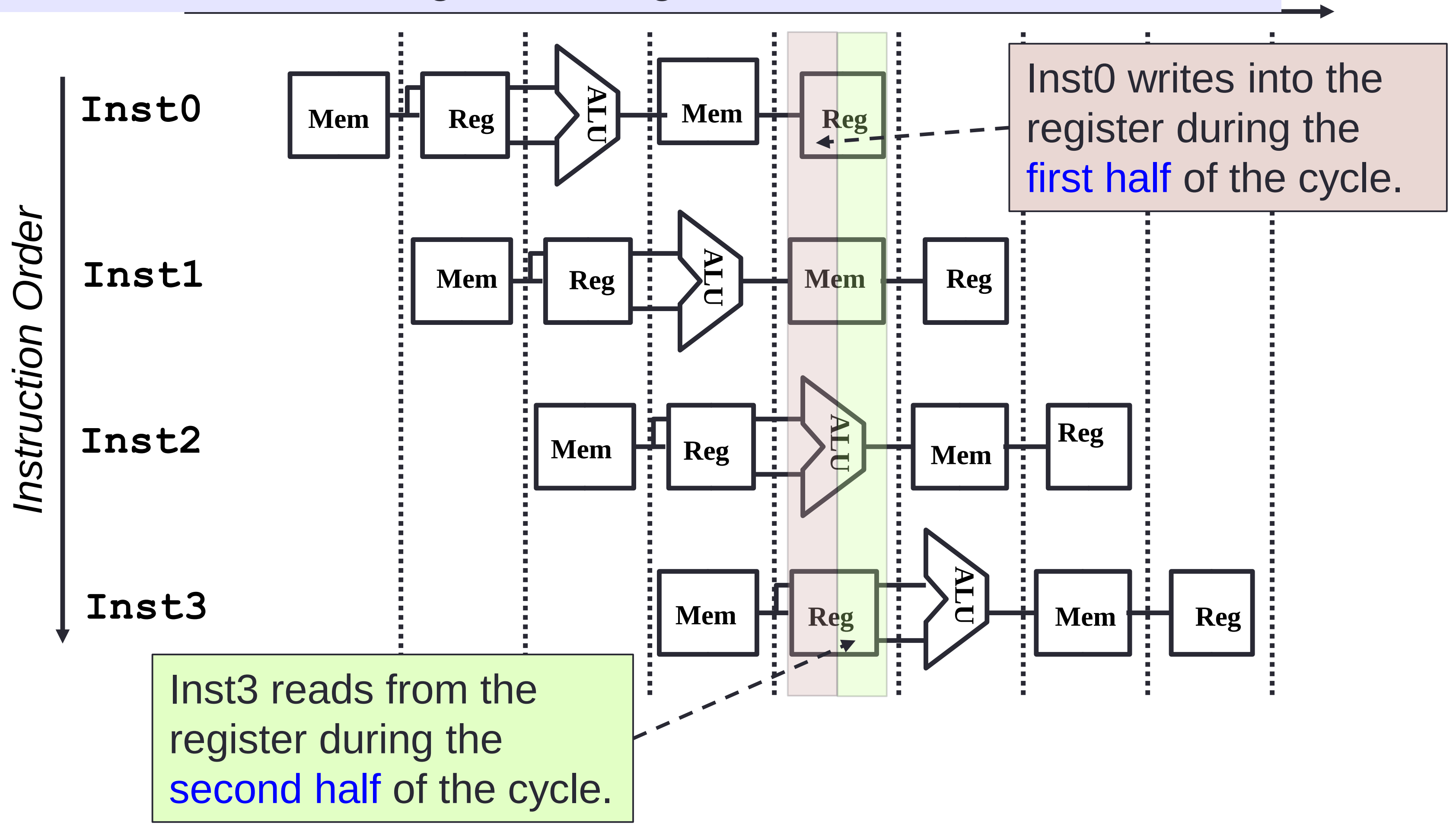
2. Structural Hazard: Memory

- Solution: Split memory into **Data** and **Instruction** memory



2. Structural Hazard: Register File

Recall that registers are very fast memory.
Solution: Split cycle into half; first half for writing into a register; second half for reading from a register.

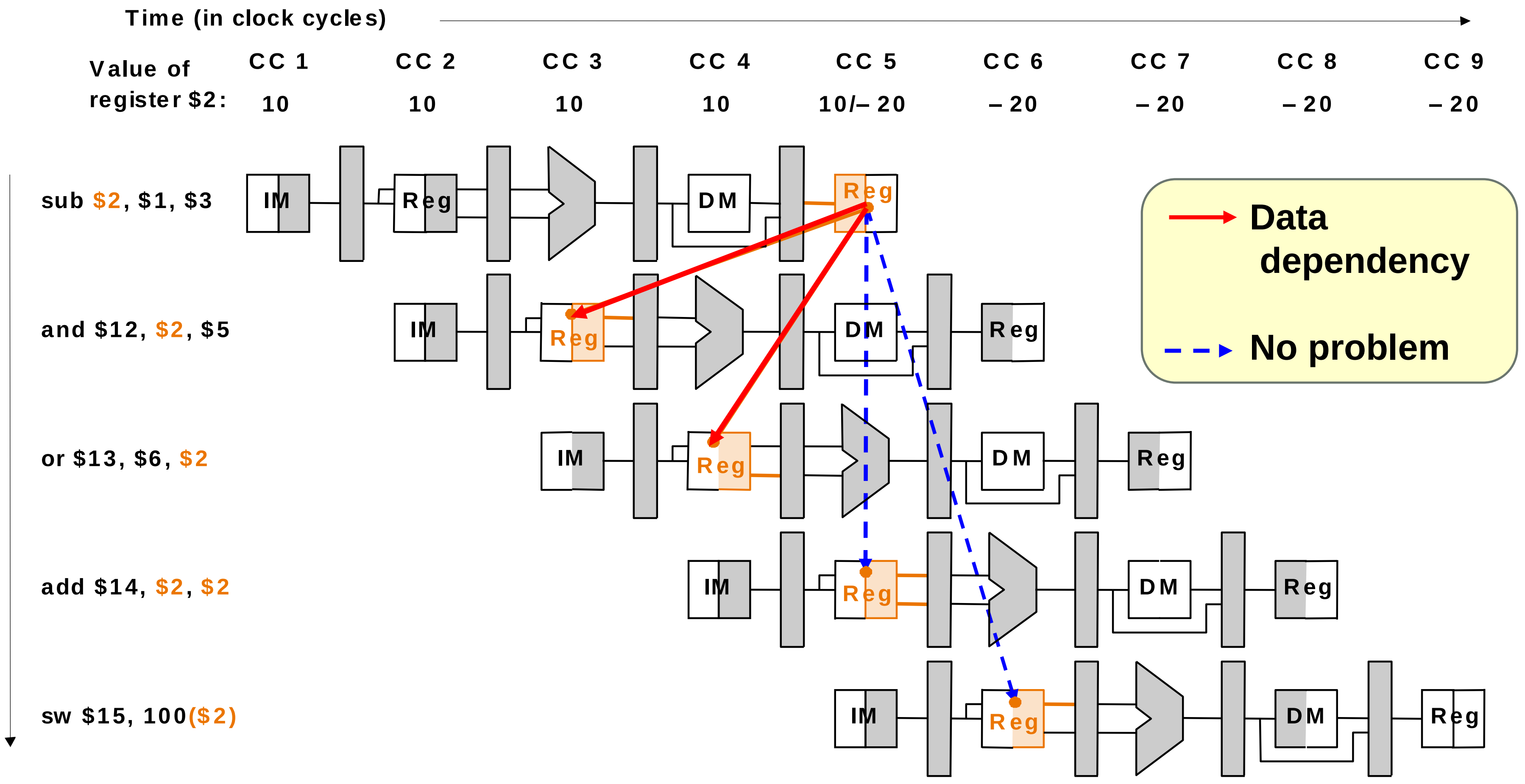


Inst0 writes into the register during the first half of the cycle.

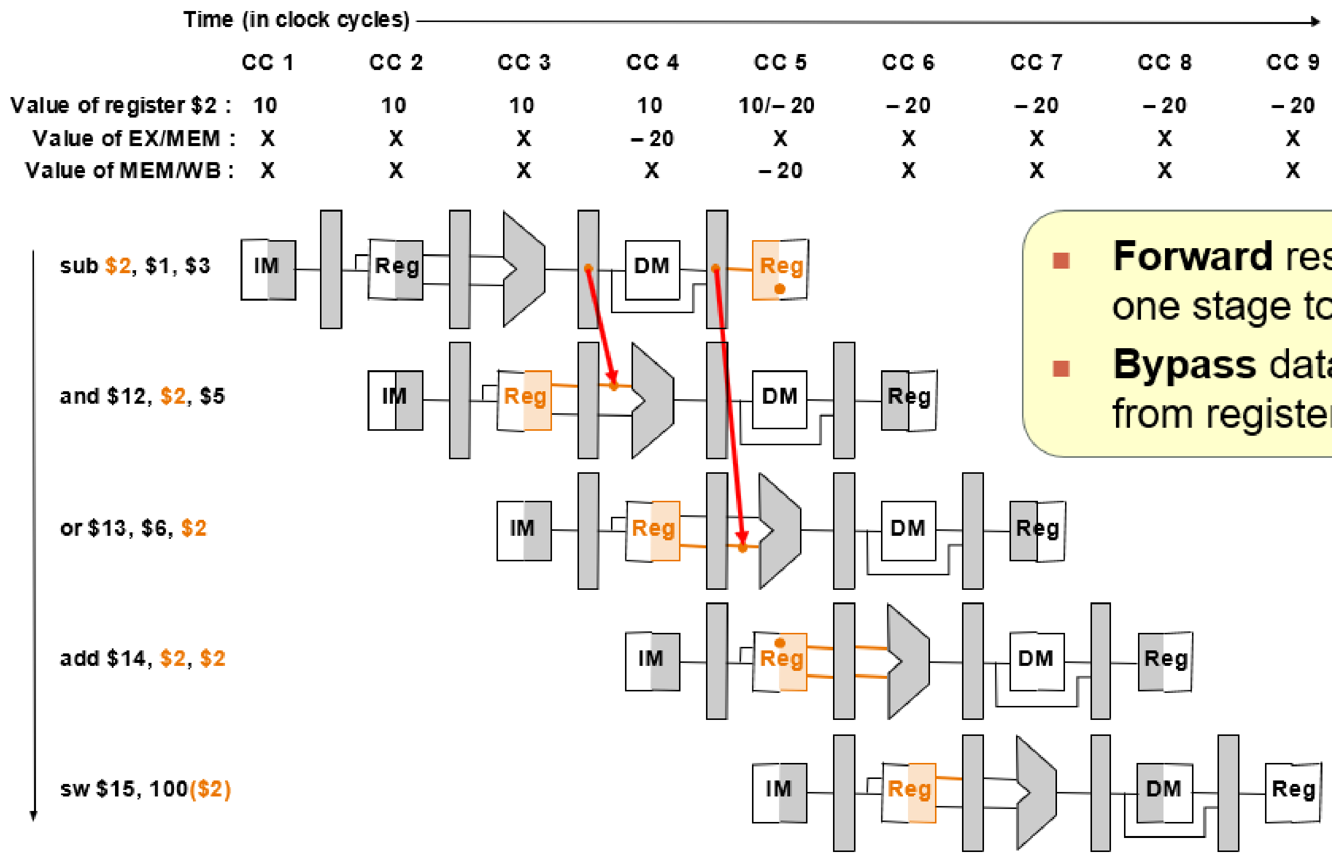
Inst3 reads from the register during the second half of the cycle.

4. RAW Data Hazards

- Value from prior instruction is needed before write back



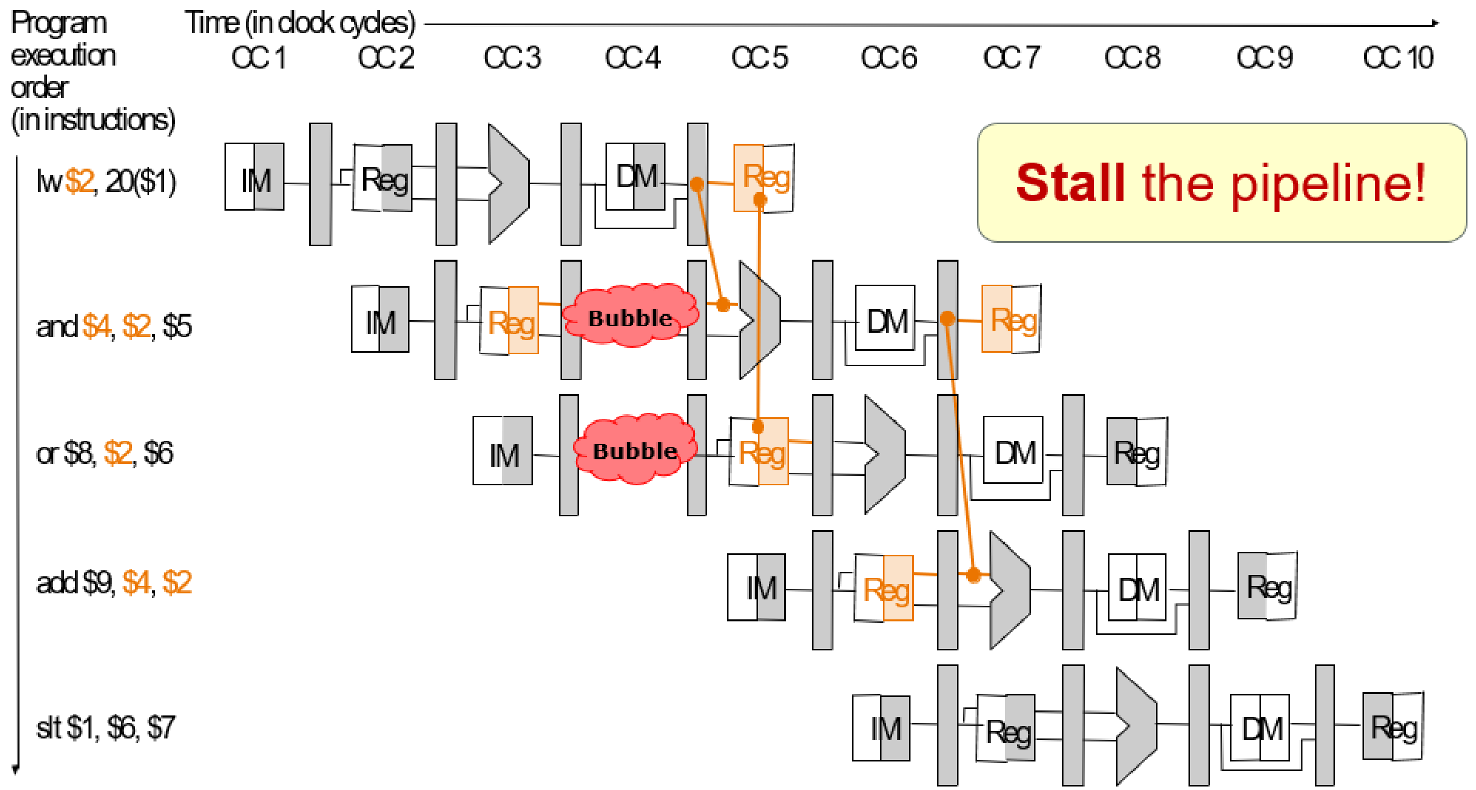
Data Hazards: Forwarding technique (1/2)



Time (in clock cycles)
CC 1 CC 2 CC 3 CC 4 CC 5 CC 6 CC 7 CC 8 CC 9
Value of register $2 : 10 10 10 10 10/–20 –20 –20 –20 –20
Value of EX/MEM : X X X –20 X X X X X
Value of MEM/WB : X X X X –20 X X X X
sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
IM
Reg
DM

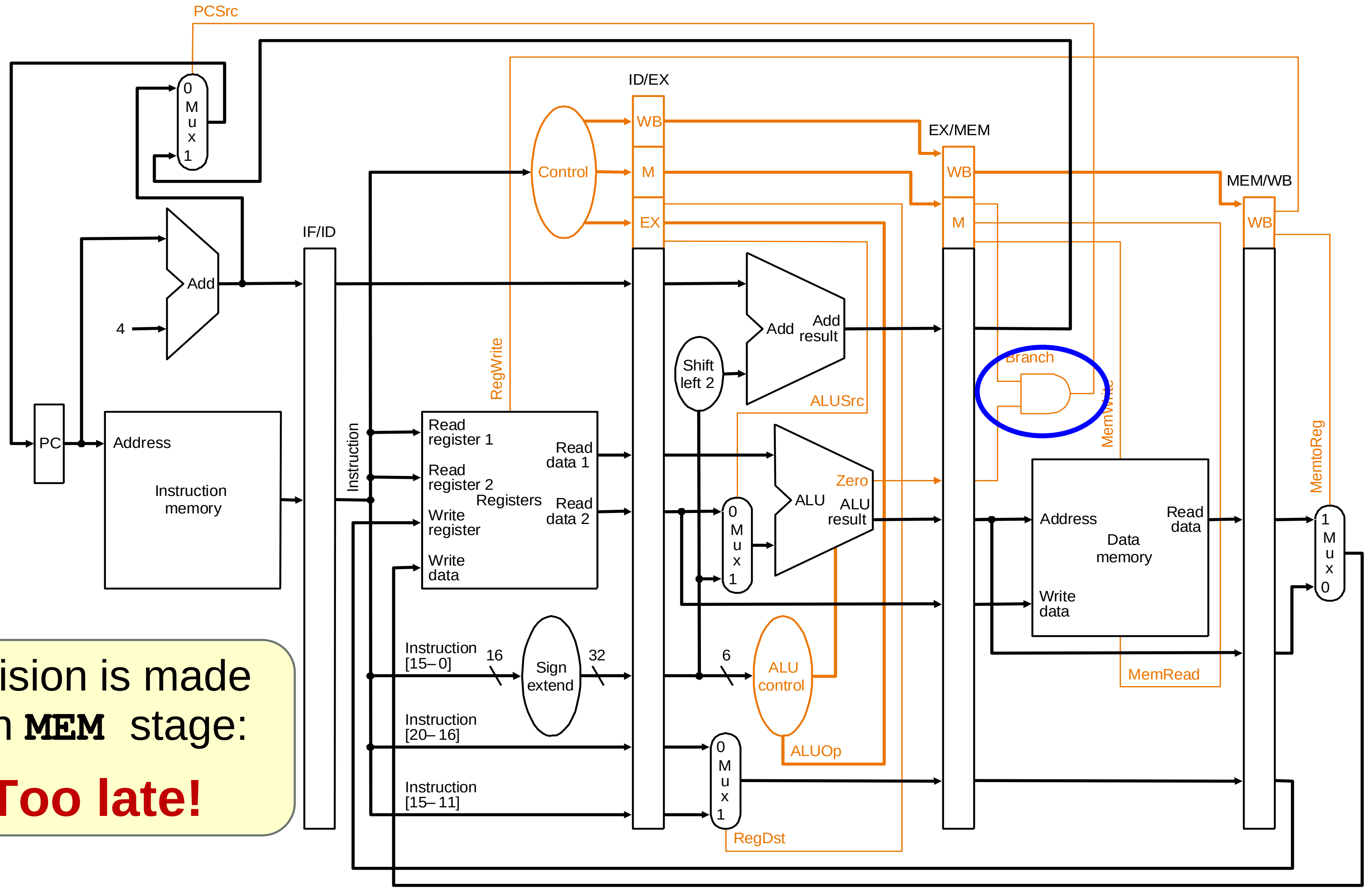
- **Forward** results from one stage to another
- **Bypass** data read from register file

Data Hazards: Forwarding technique (2/2)



- Forwarding removes all delay, except for lw (why?)
- With forwarding, there is still 1 cycle of delay for the instruction after lw.

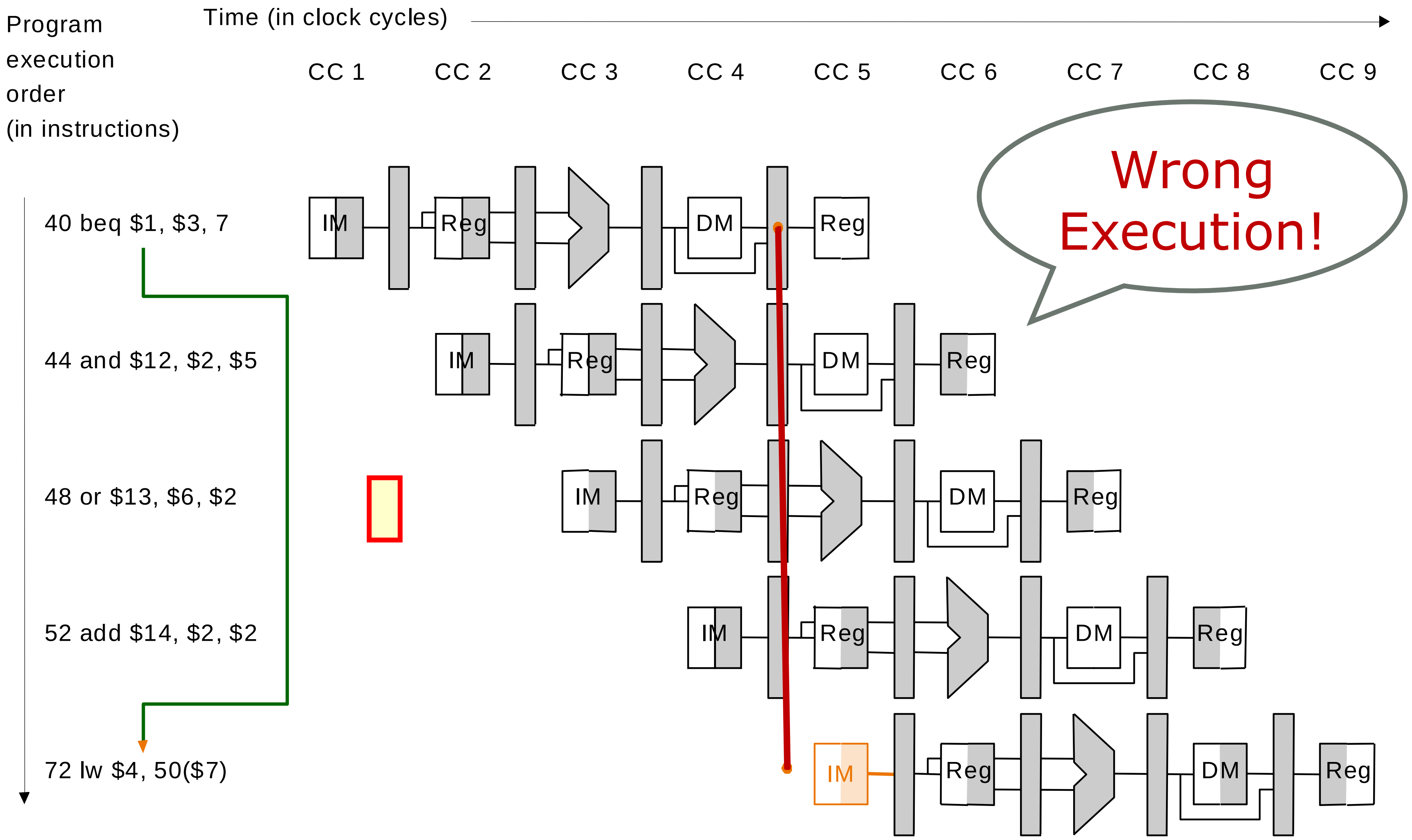
5. Control Dependency: **Why?**



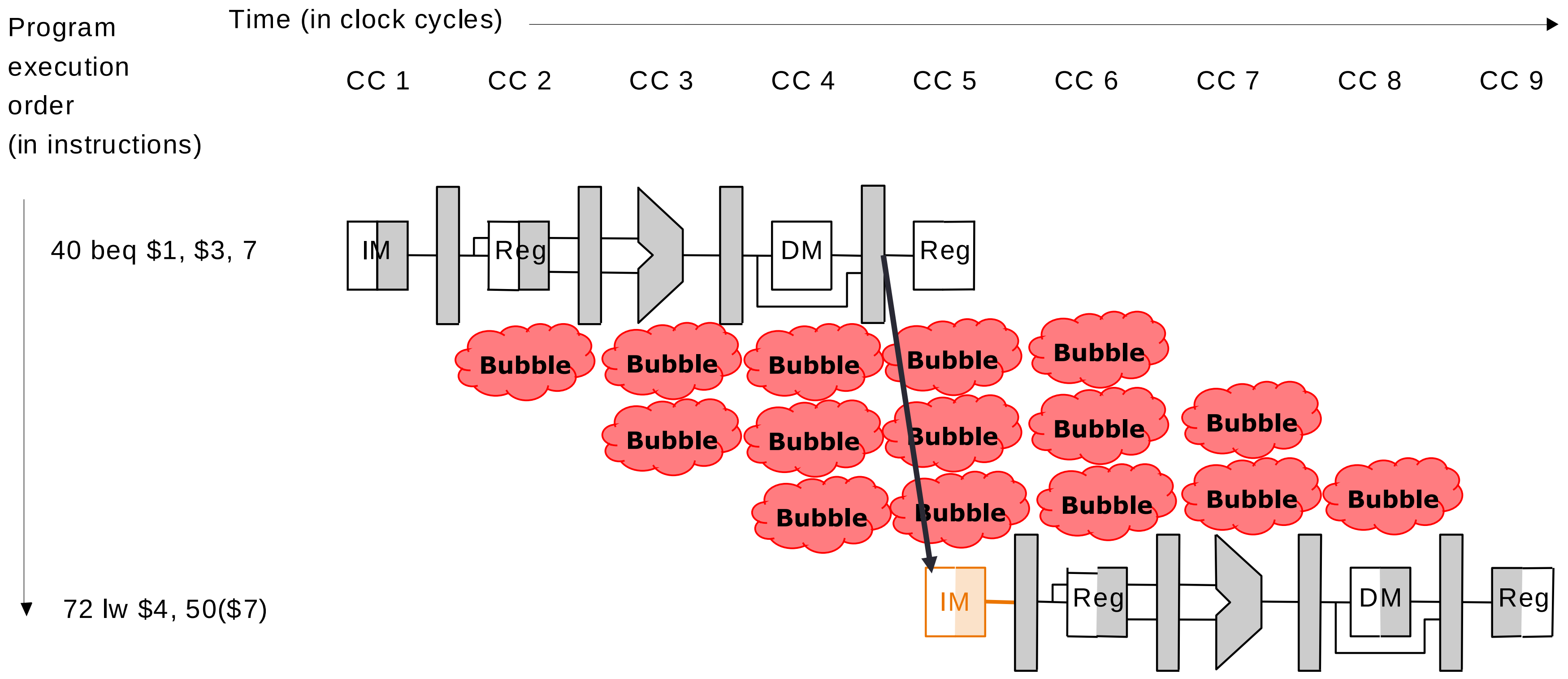
Decision is made in **MEM** stage:

Too late!

5. Control Dependency: **Example**



6. Control Hazards: **Stall Pipeline?**



- Wait until the branch outcome is known and then fetch the correct instructions

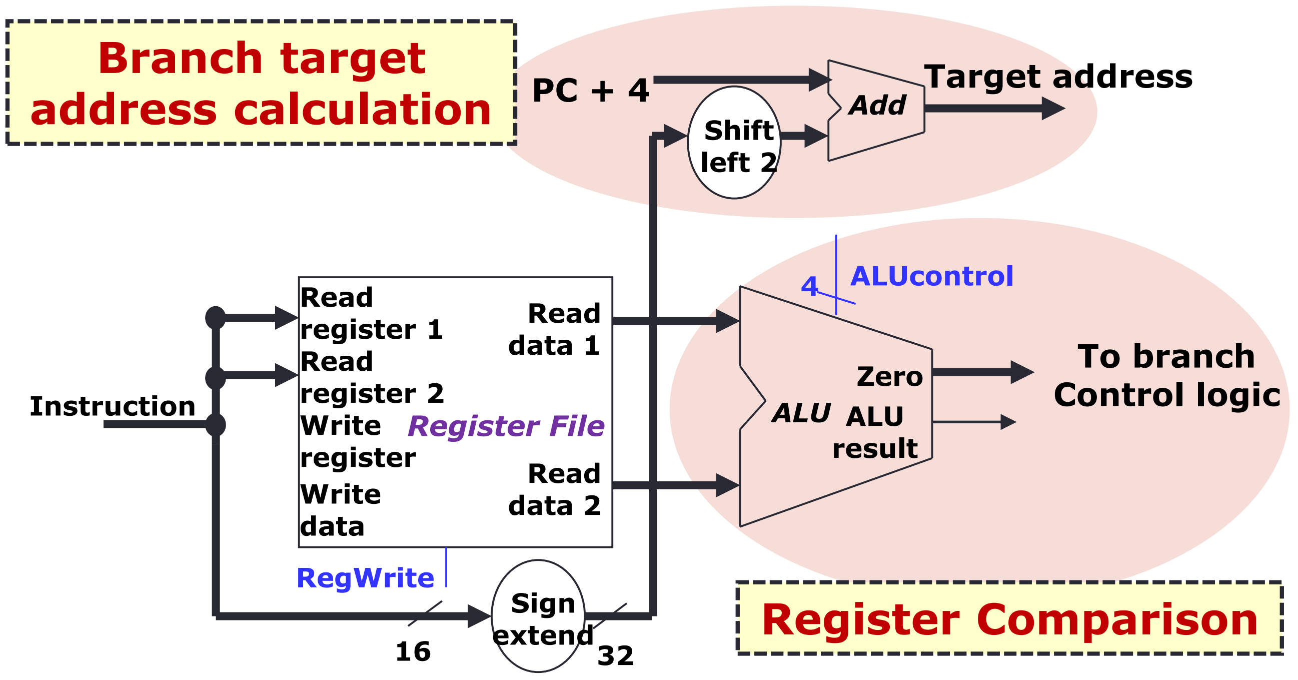
➔ Introduces **3 clock cycles delay**

Control Hazards

- Techniques to reduce stalls:
  1. Early Branching resolution
     - Move branch decision calculation to earlier pipeline stage
  2. Branch Prediction
     - Guess the outcome of the branch
  3. Delayed Branching
     - Do something useful while waiting for the outcome

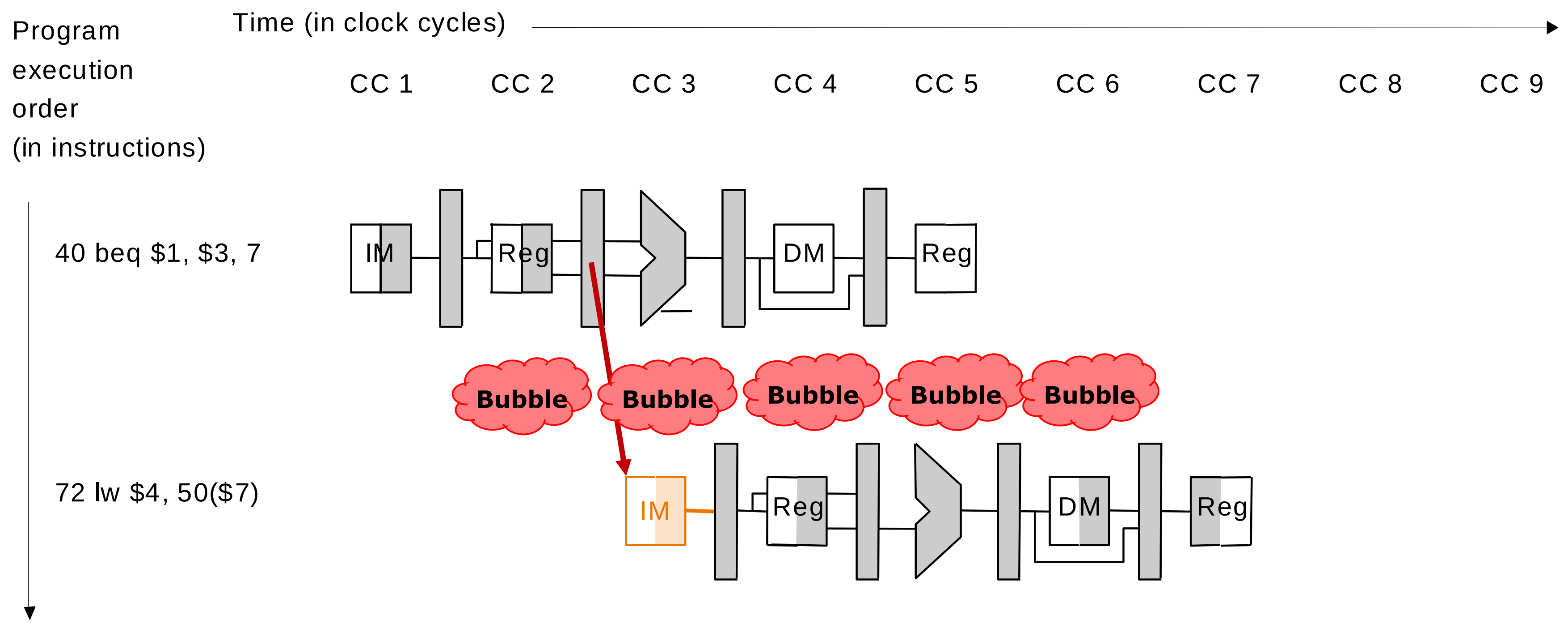
Control Hazards: Early Branching

- Branch resolution made at stage 2 (ID stage)



QnA:
Q2: Why can't the next instruction be fetched right after the branch instruction's ID stage in early branching? Isn't the branch target address calculated in the EX stage?

6.1 Reduce Stalls: **Early Branch** (3/3)

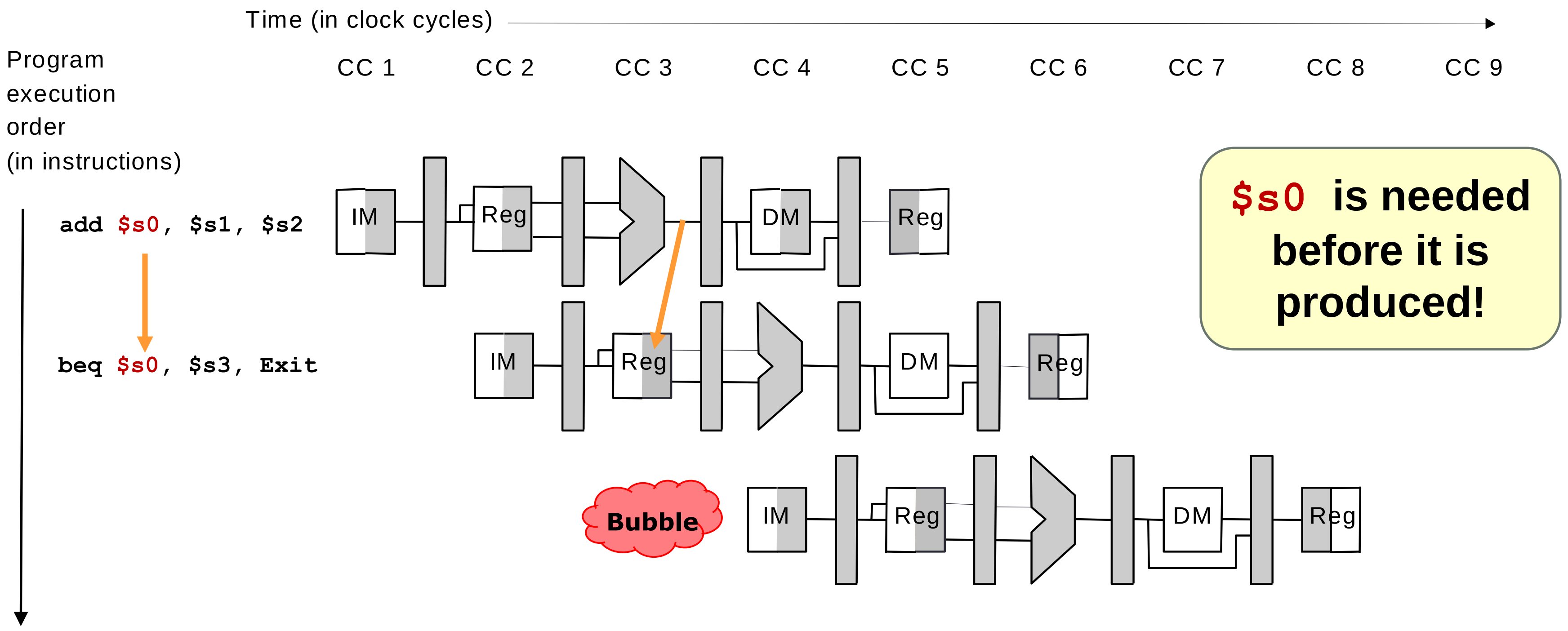


- Wait until the branch decision is known:
  - Then fetch the correct instruction
- Reduced from 3 to **1 clock cycle delay**

6.1 Early Branch: **Problems** (1/3)

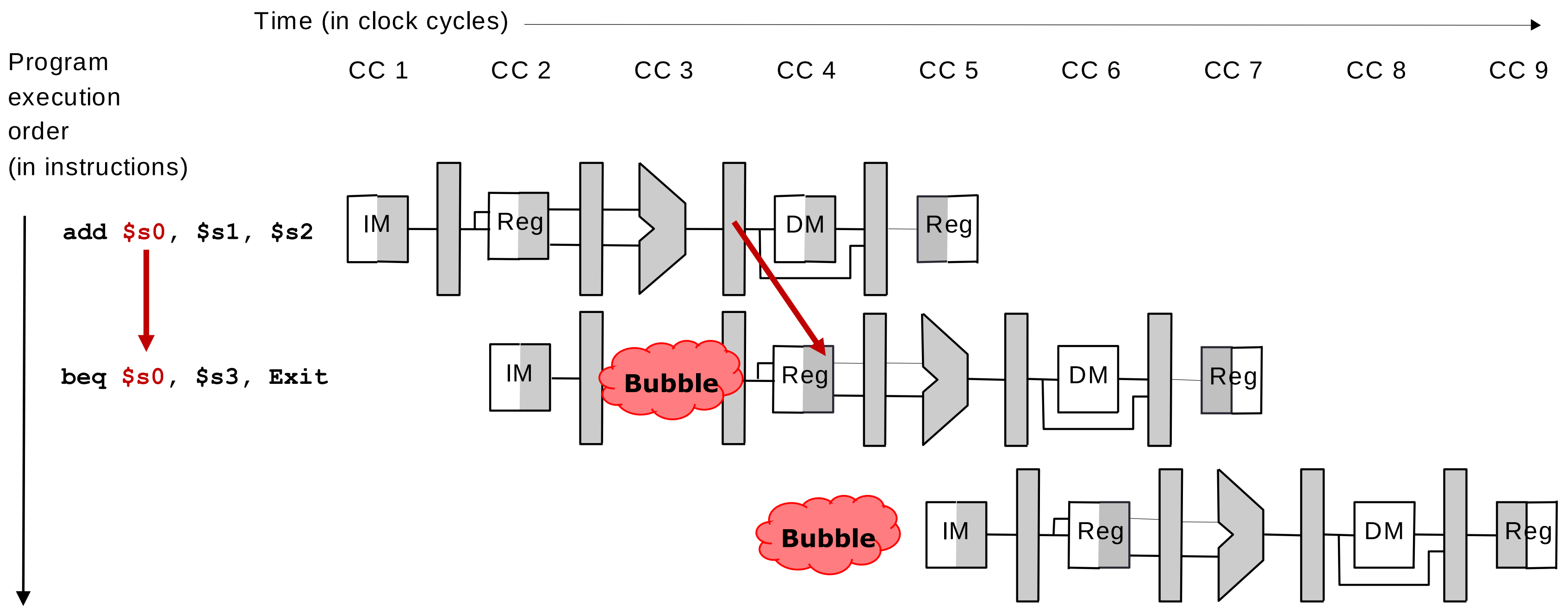
You solved one problem, but introduced another problem!

- However, if the register(s) involved in the comparison is produced by preceding instruction:
  - Further stall is still needed!



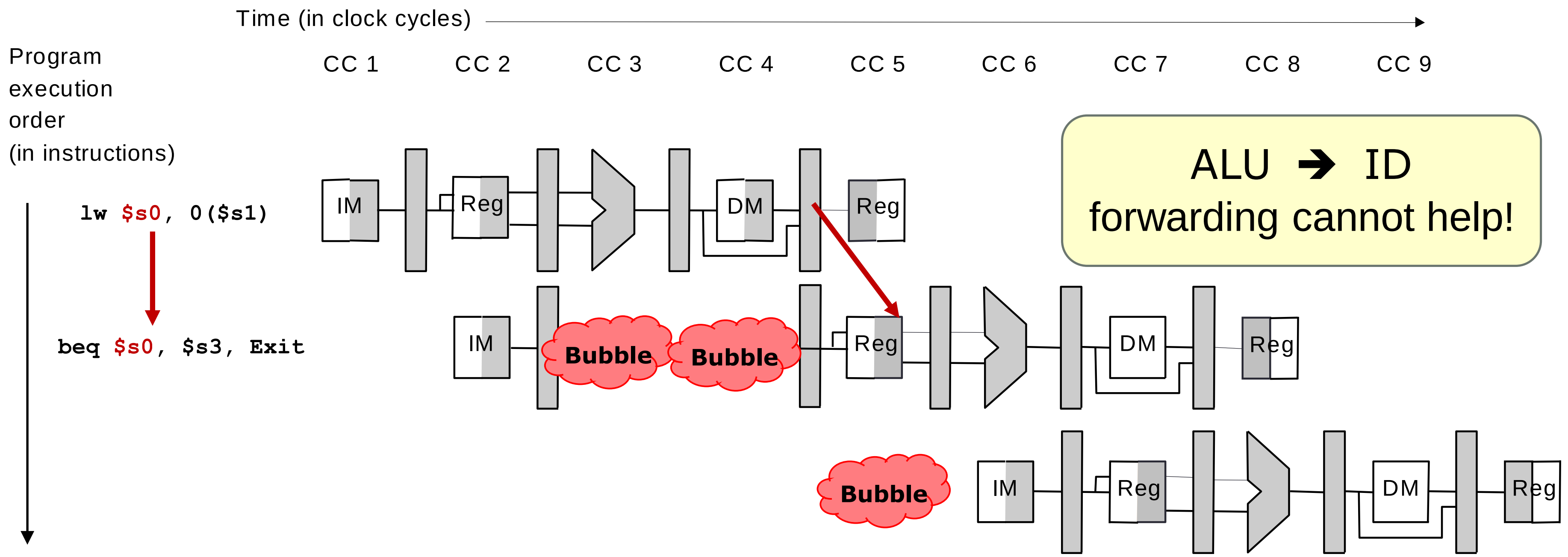
6.1 Early Branch: **Problems** (2/3)

- **Solution:**
  - Add forwarding path from ALU to ID stage
  - **One clock cycle delay** is still needed



6.1 Early Branch: **Problems** (3/3)

- Problem is worse with **load** followed by **branch**
- **Solution:**
  - MEM to ID forwarding and 2 more stall cycles!
  - In this case, we ended up with 3 total stall cycles
    ➔ no improvement!

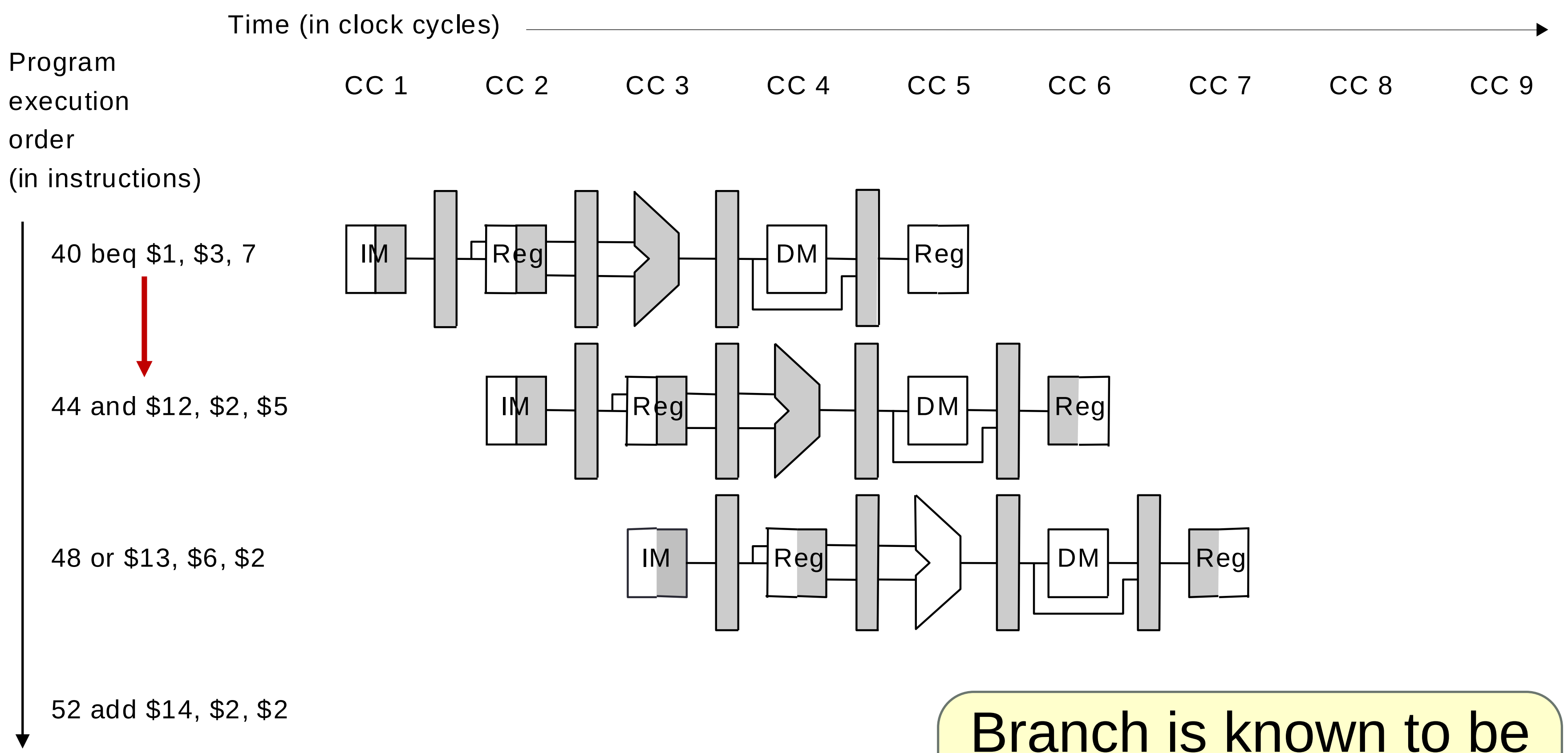


6.2 Reduce Stalls: **Branch Prediction**

- There are many branch prediction schemes
  - We only cover the simplest in this course ☺
- Simple prediction:
  - All branches are assumed to be **not taken**

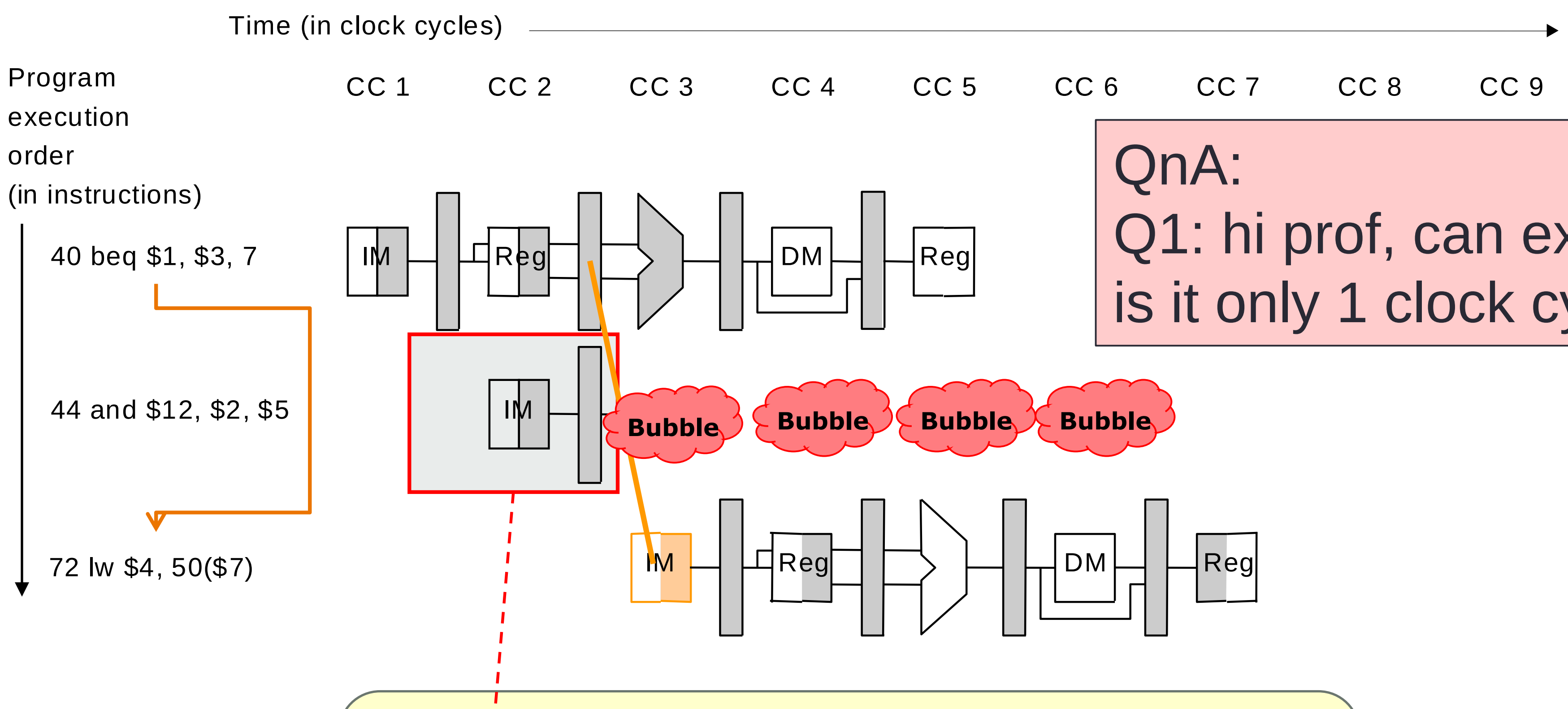
  ➔ Fetch the successor instruction and start pumping it through the pipeline stages
- When the actual branch outcome is known:
  - **Not taken**: Guessed correctly ➔ No pipeline stall
  - **Taken**: Guessed wrongly ➔ Wrong instructions in the pipeline ➔ **Flush** successor instruction from the pipeline

6.2 Branch Prediction: **Correct Prediction**



Branch is known to be **not taken** in cycle 3
➔ no stall needed!

6.2 Branch Prediction: **Wrong Prediction**



QnA:
Q1: hi prof, can explain why is it only 1 clock cycle delay?

Branch is known to be **taken** in cycle 3
➔"**and**" instruction should not be executed
➔ Flushed from pipeline

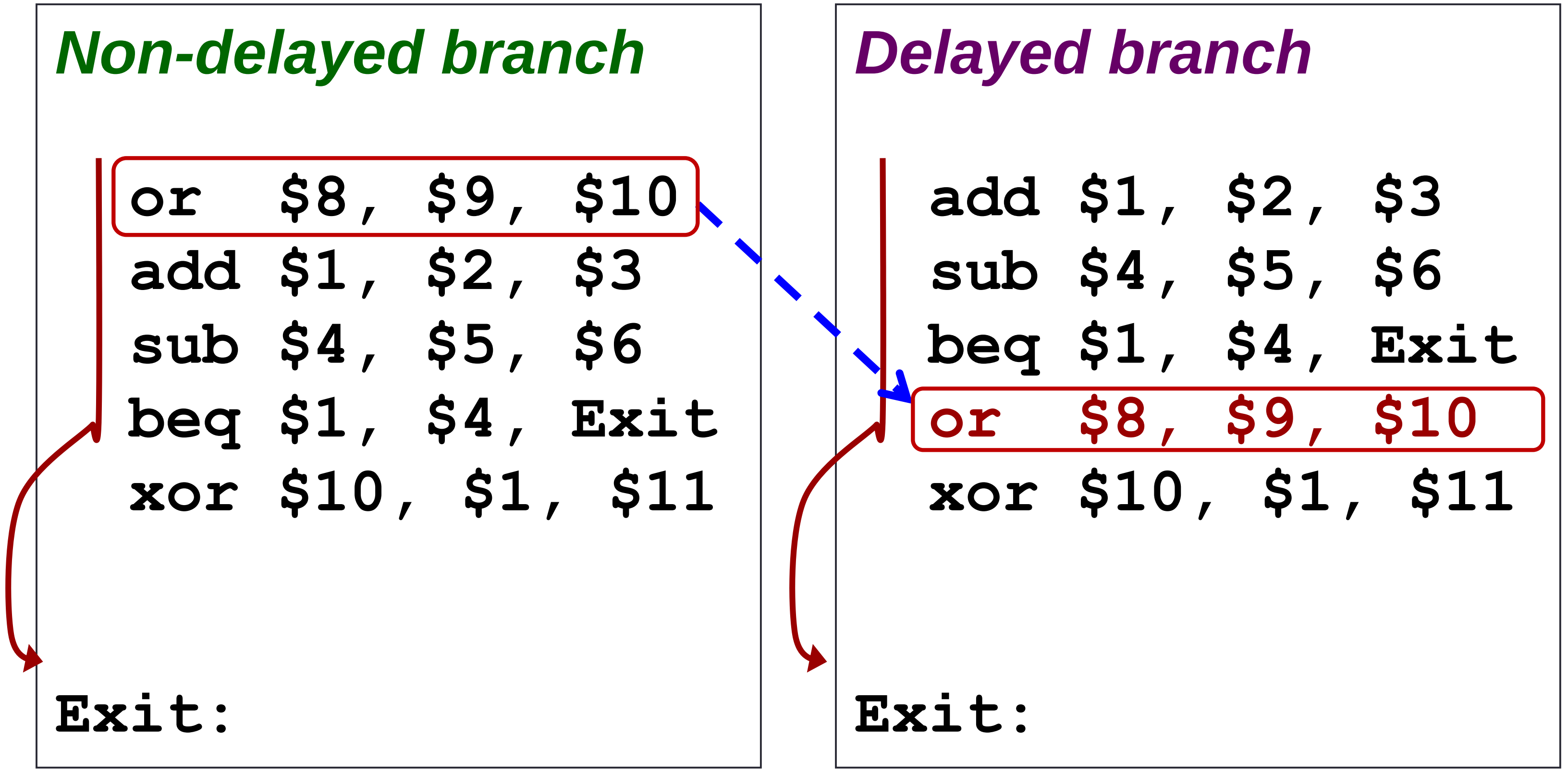
6.3 Reduce Stalls: **Delayed Branch**

- **Observation:**
  - Branch outcome takes **X** number of cycles to be known

  → **X** cycles stall
- **Idea:**
  - Move **non-control dependent instructions** into the X slots following a branch
    - Known as the **branch-delay slot**

  → These instructions are executed **regardless of the branch outcome**
- In our MIPS processor:
  - Branch-Delay slot = **1** (with the early branch)

6.3 Delayed Branch: **Example**



Non-delayed branch

```
or   $8, $9, $10
add  $1, $2, $3
sub  $4, $5, $6
beq  $1, $4, Exit
xor  $10, $1, $11


Exit:
```

Delayed branch

```
add  $1, $2, $3
sub  $4, $5, $6
beq  $1, $4, Exit
or   $8, $9, $10
xor  $10, $1, $11


Exit:
```

- The "`or`" instruction is moved into the delayed slot:
  - Get executed regardless of the branch outcome
- ➔ Same behavior as the original code!

6.3 Delayed Branch: **Observation**

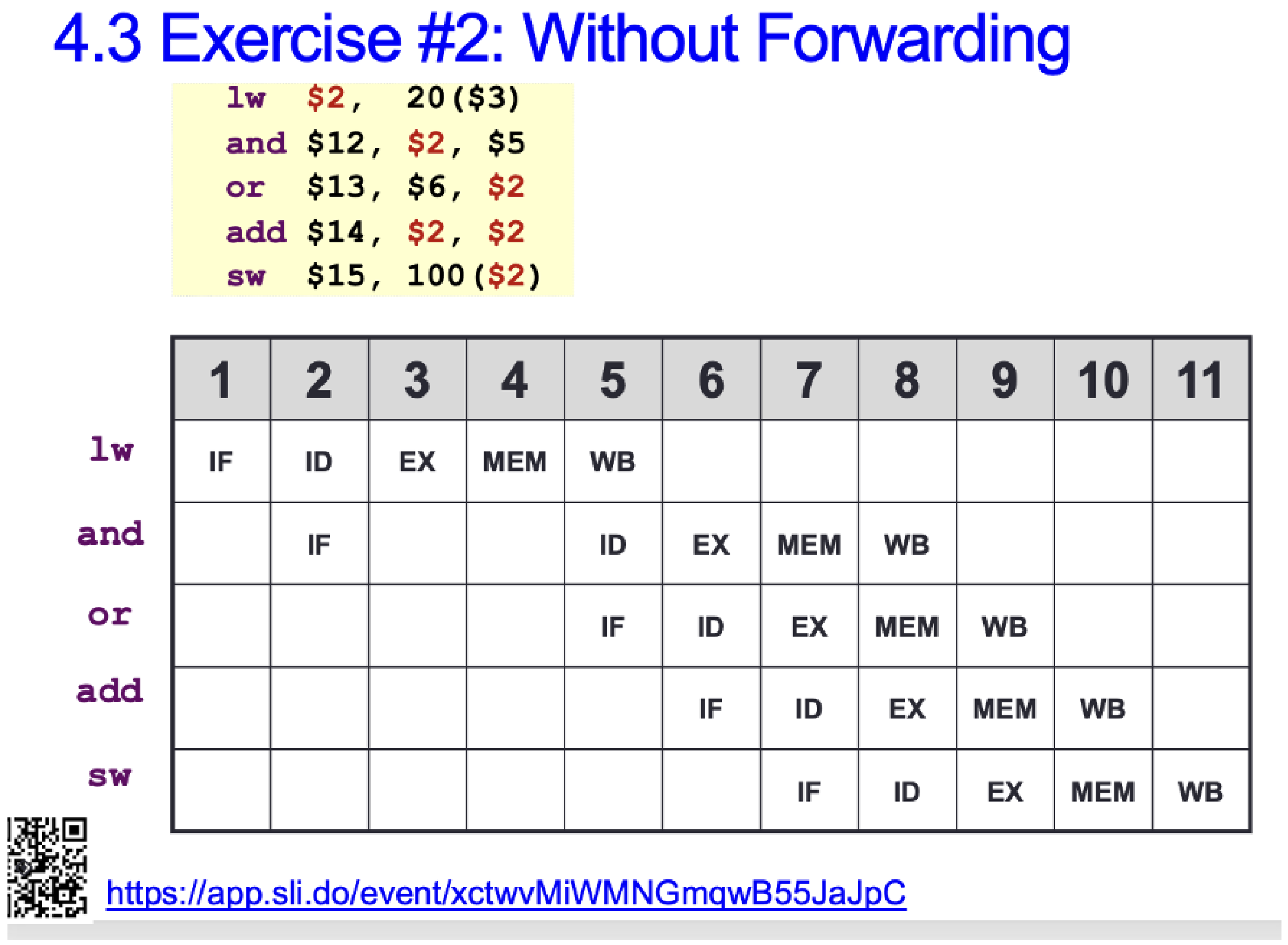
- **Best case scenario**
  - There is an instruction **preceding the branch** which **can be moved** into the delayed slot
    - Program correctness must be preserved!
- **Worst case scenario**
  - Such instruction cannot be found

  ➔ Add a no-op (**nop**) instruction in the branch-delay slot
- Re-ordering instructions is a common method of program optimization
  - Compiler must be smart enough to do this
  - Usually can find such an instruction at least 50% of the time

QnA

Common question

In Lecture 21 Slide 24, why is the IF stage for the OR instruction in clock cycle 5? Can it not be at clock cycle 3?



4.3 Exercise #2: Without Forwarding

```
lw  $2,  20($3)
and $12, $2, $5
or  $13, $6, $2
add $14, $2, $2
sw  $15, 100($2)
```

	1	2	3	4	5	6	7	8	9	10	11
lw	IF	ID	EX	MEM	WB						
and		IF			ID	EX	MEM	WB			
or					IF	ID	EX	MEM	WB		
add						IF	ID	EX	MEM	WB	
sw							IF	ID	EX	MEM	WB

https://app.sli.do/event/xctwvMiWMNGmqwB55JaJpC

QnA Q3

Hi prof, is there any useful observation we can make to calculate the clock cycles efficiently? cus filling in the whole pipeline table does not seem practical under exam conditions.

AY2021/22 Sem1 exam

Refer to the given MIPS code. Assume a 5-stage MIPS pipeline and A[0] > B[0].

(a) How many cycles does this code segment take to complete its execution in the first iteration (**I1** to **I18**) in an ideal pipeline?

Write the total number of additional delay cycles for each of the parts (b) to (d).

(b) Assuming without forwarding and branch decision is made at MEM stage (stage 4). No branch prediction is made.

(c) Assuming with forwarding and branch decision is made at MEM stage (stage 4). No branch prediction is made.

(d) …

```
 add   $t0, $0, $0        # I1
       add  $t9, $0, $0   # I2
       addi $t1, $a1, 0   # I3
       addi $t2, $a2, 0   # I4
       sll  $t8, $a0, 2   # I5
loop:  slt  $t3, $t0, $t8 # I6
       beq  $t3, $0, end  # I7
       lw   $s1, 0($t1)   # I8
       lw   $s2, 0($t2)   # I9
       slt  $t3, $s1, $s2 # I10
       bne  $t3, $0, else # I11
       add  $t9, $t9, $s1 # I12
       j    cont          # I13
~~else: sub  $t9, $t9, $s2 # I14~~
cont:  addi $t0, $t0, 4   # I15
       addi $t1, $t1, 4   # I16
       addi $t2, $t2, 4   # I17
       j    loop          # I18
end:
```

QnA Q3

Hi prof, is there any useful obs
calculate the clock cycles effici
pipeline table does not seem p

AY2021/22 Sem1 exam

Refer to the given MIPS code. Assume a 5-stage MIPS pipeline and A[0] > B[0].

(a) How many cycles does this code segment take to complete its execution in the first iteration (**I1** to **I18**) in an ideal pipeline?

Write the total number of additional delay cycles for each of the parts (b) to (d).

(b) Assuming without forwarding and branch decision is made at MEM stage (stage 4). No branch prediction is made.

(c) Assuming with forwarding and branch decision is made at MEM stage (stage 4). No branch prediction is made.

(d) …

```
                                                (b) (c) (d)
       add  $t0, $0, $0     # I1
       add  $t9, $0, $0     # I2
       addi $t1, $a1, 0     # I3
       addi $t2, $a2, 0     # I4
       sll  $t8, $a0, 2     # I5
loop:  slt  $t3, $t0, $t8   # I6   +2
       beq  $t3, $0, end    # I7   +2

       lw   $s1, 0($t1)     # I8   +3  +3
       lw   $s2, 0($t2)     # I9
       slt  $t3, $s1, $s2   # I10  +2  +1
       bne  $t3, $0, else   # I11  +2
       add  $t9, $t9, $s1   # I12  +3  +3
       j    cont            # I13
~~else:  sub  $t9, $t9, $s2   # I14~~
cont:  addi $t0, $t0, 4     # I15  +1  +1
       addi $t1, $t1, 4     # I16
       addi $t2, $t2, 4     # I17
       j    loop            # I18
end:
                    Total:         +15 +8
```

PIPELINING QUIZ

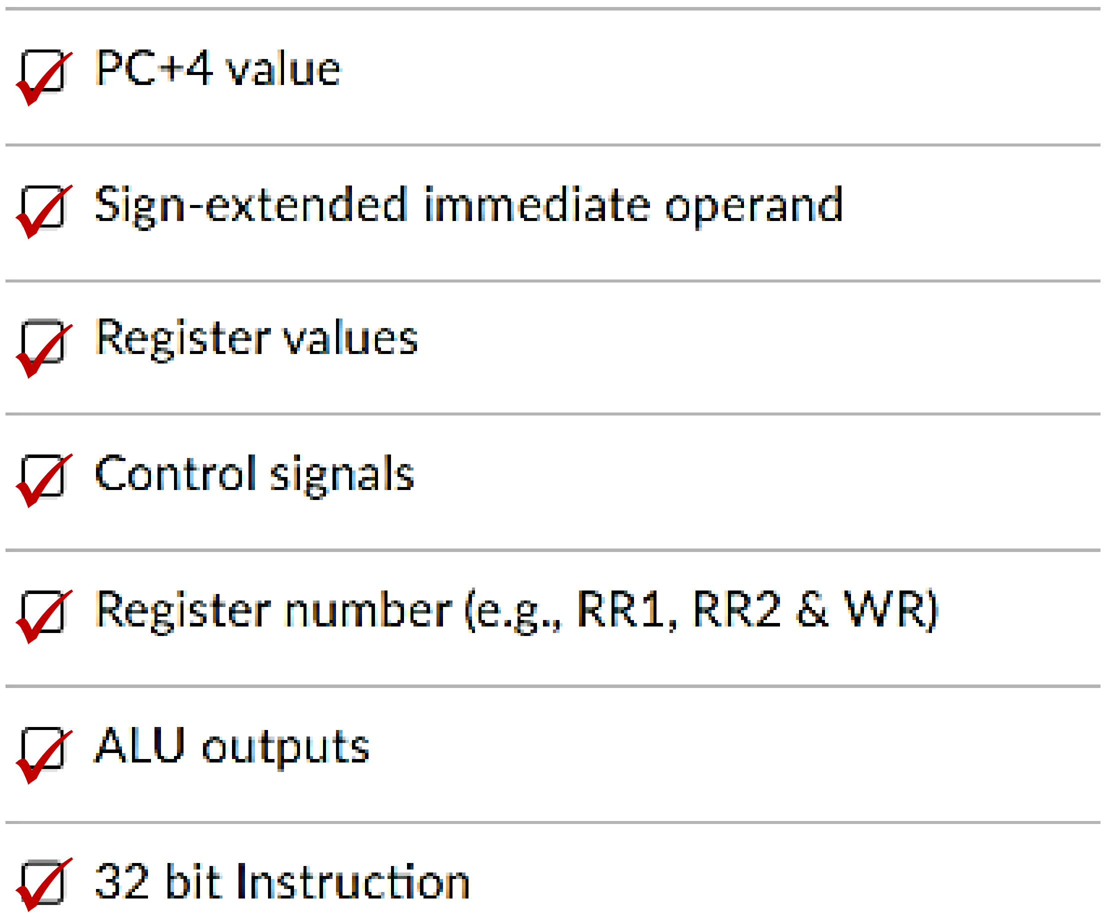
Pipelining Quiz Question 1

Let’s say there are 6 stages in executing an instruction and there are 10 instructions in a program. Each stage takes 1 clock cycle to execute. How many clock cycles are needed to complete the execution of this program in an ideal pipeline?

I = no. of instructions
N = no. of pipeline stages
I + N – 1 = 10 + 6 – 1 = **15 cycles**

Pipelining Quiz Question 2

Pipeline register can store which of the following?



- [✓] PC+4 value
- [✓] Sign-extended immediate operand
- [✓] Register values
- [✓] Control signals
- [✓] Register number (e.g., RR1, RR2 & WR)
- [✓] ALU outputs
- [✓] 32 bit Instruction

Pipelining Quiz Question 3

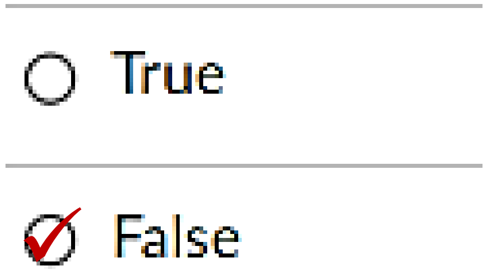
Each of the pipeline registers contains multiple registers of size 32 bits?

[✓ True]

False

Pipelining Quiz Question 4

The MIPS processor that we discussed in this course can execute at most 4 instructions simultaneously in an ideal pipeline.



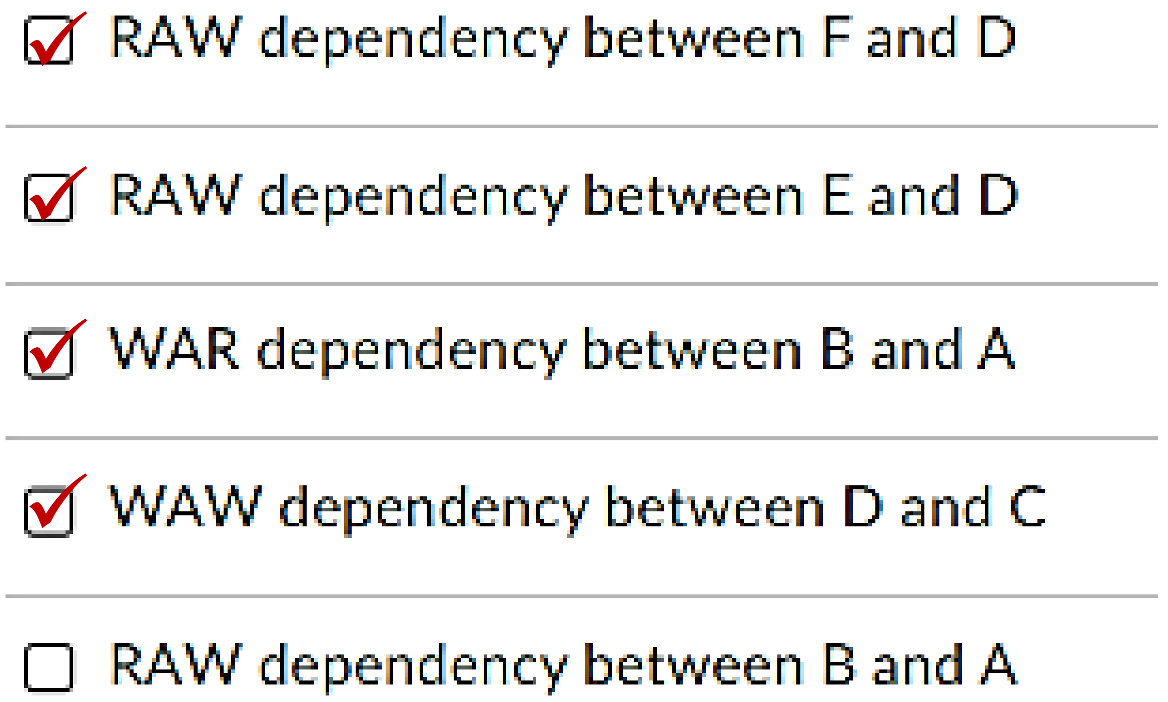
True [✓ False]

Pipelining Quiz Question 5

Below is a sequence of instructions A to F:

```
A.  lw  $t2, 0($t1)
B.  lw  $t1, 100($t6)
C.  sub $t6, $t1, $t2
D.  add $t6, $t2, $t6
E.  and $t3, $t6, $0
F.  sw  $t6, 50($t1)
```

Which of the following are true with respect to RAW, WAR and WAW data dependency?



- ✓ RAW dependency between F and D
- ✓ RAW dependency between E and D
- ✓ WAR dependency between B and A
- ✓ WAW dependency between D and C
- ☐ RAW dependency between B and A

Pipelining Quiz Question 6

How many cycles are needed to execute this program with and without forwarding?

```
A.  lw  $t2, 0($t1)
B.  lw  $t1, 100($t6)
C.  sub $t6, $t1, $t2
D.  add $t6, $t2, $t6
E.  and $t3, $t6, $0
F.  sw  $t6, 50($t1)
```

Without forwarding: 16 cycles

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
lw	F	D	E	M	W											
lw		F	D	E	M	W										
sub			F			D	E	M	W							
add				F					D	E	M	W				
and					F							D	E	M	W	
sw						F							D	E	M	W

With forwarding:

11 cycles

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
lw	F	D	E	M	W											
lw		F	D	E	M	W										
sub			F	D		E	M	W								
add				F	D		E	M	W							
and					F	D		E	M	W						
sw						F	D		E	M	W					

Pipelining Quiz Question 7

How many cycles are needed to execute this program with forwarding, assume branch-taken prediction strategy, branch decision made at MEM, whereas the actual branch in the code is taken.

```
    lw  $s2, 0($s1)
    bne $s2, $s3, L1
    sub $s0, $s4, $s5
L1: add $s0, $s0, $s3
```

	1	2	3	4	5	6	7	8	9	10	11	12
lw	F	D	E	M	W							
bne		F	D		E	M	W					
add			F	D		E	M	W				

8 cycles

Pipelining Quiz Question 8

How many cycles are needed to execute this program with forwarding, assume <u>branch-NOT-taken prediction strategy</u>, branch decision made at MEM, whereas the actual branch in the code is taken.

```
    lw  $s2, 0($s1)
    bne $s2, $s3, L1
    sub $s0, $s4, $s5
L1: add $s0, $s0, $s3
```

	1	2	3	4	5	6	7	8	9	10	11	12
lw	F	D	E	M	W							
bne		F	D		E	M	W					
sub			F	D		E	~~M~~	~~W~~				
add				F	D		~~E~~	~~M~~	~~W~~			
add							F	D	E	M	W	

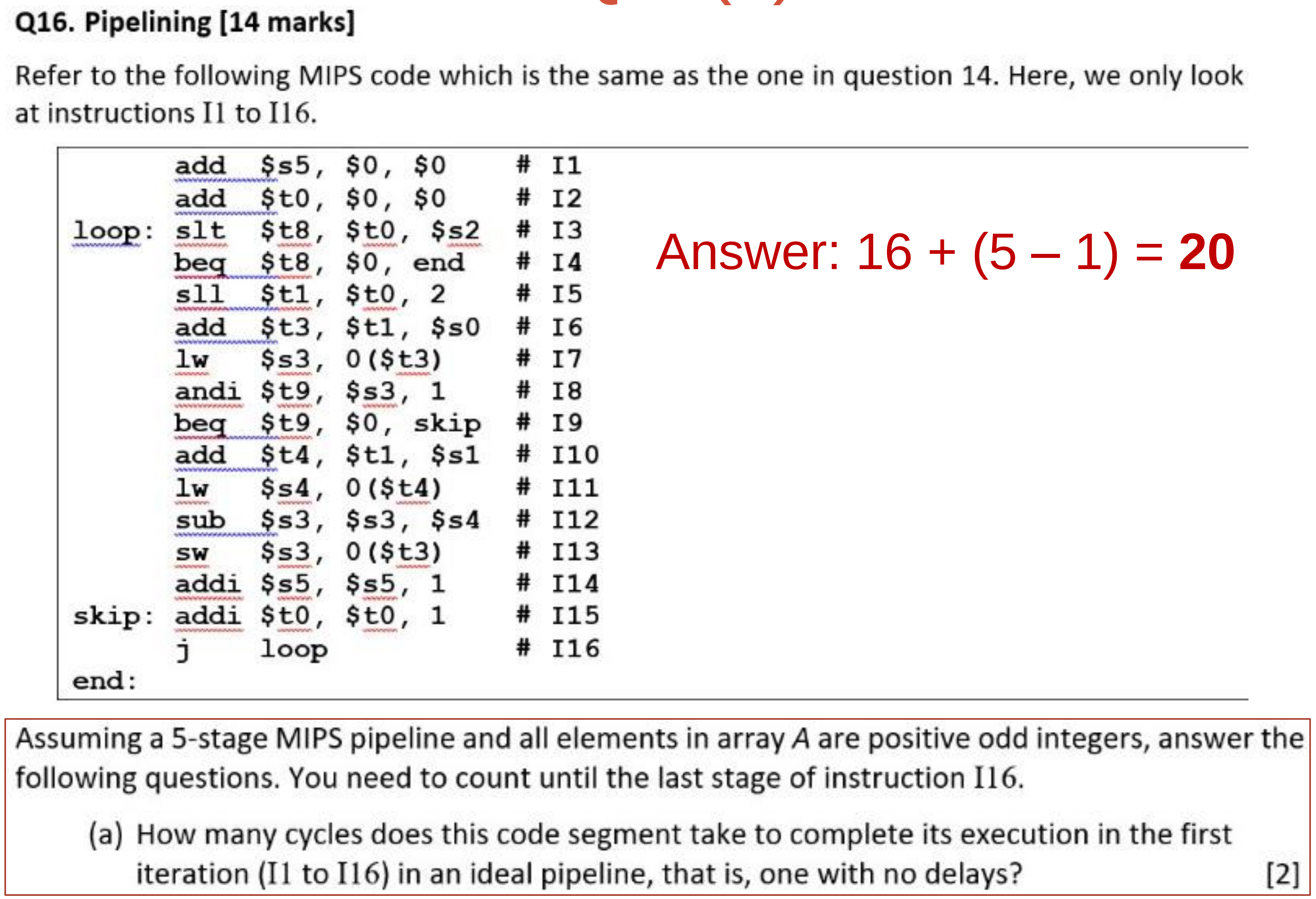
11 cycles

PAST YEARS' QUESTIONS

Pipelining

- AY2020/21 Sem2 Q16
- AY2023/24 Sem1 Q16

AY2020/21 Sem2 Q16(a)



Q16. Pipelining [14 marks]

Refer to the following MIPS code which is the same as the one in question 14. Here, we only look at instructions I1 to I16.

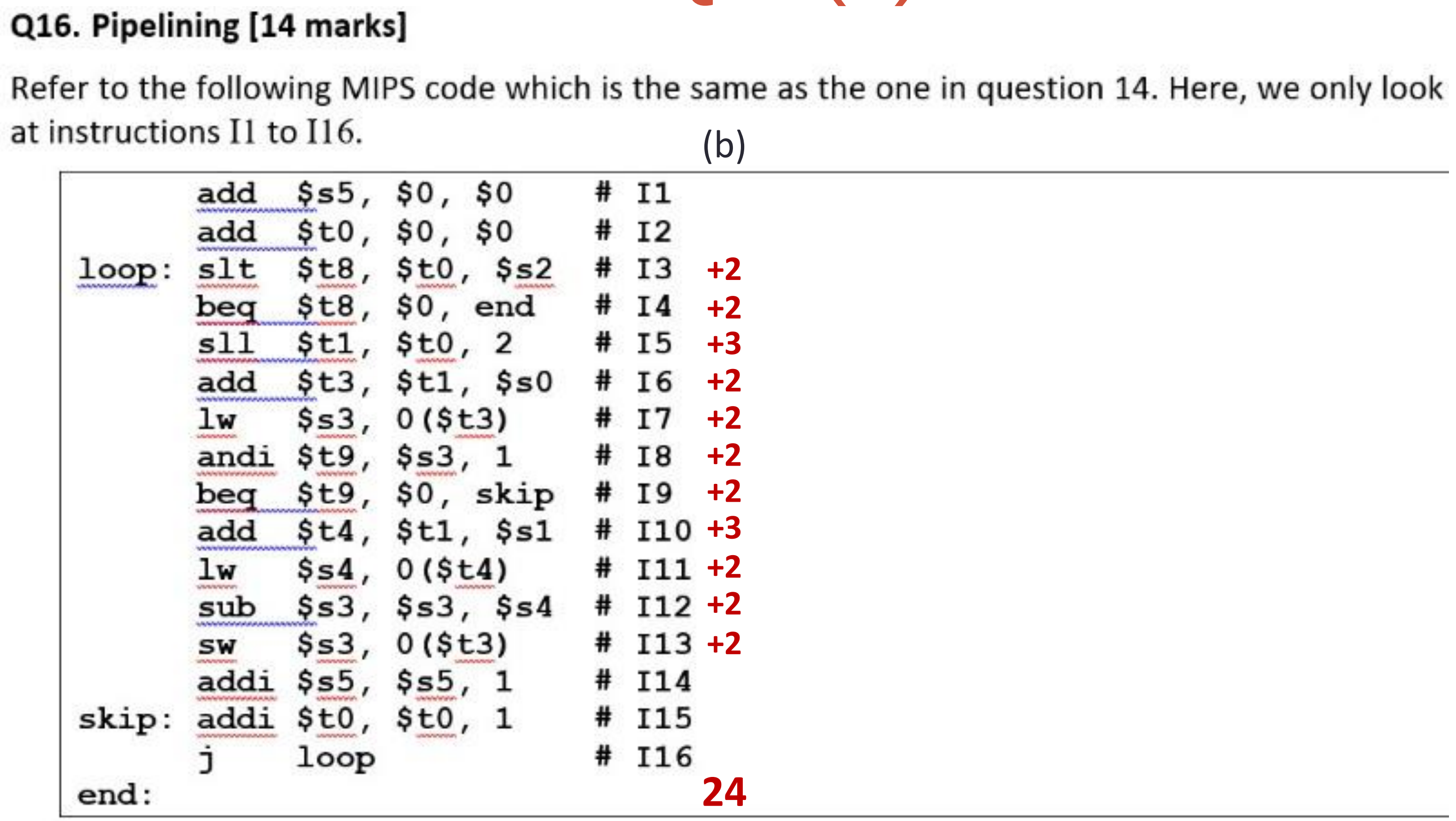
```
       add  $s5, $0, $0     # I1
       add  $t0, $0, $0     # I2
loop:  slt  $t8, $t0, $s2   # I3
       beq  $t8, $0, end    # I4
       sll  $t1, $t0, 2     # I5
       add  $t3, $t1, $s0   # I6
       lw   $s3, 0($t3)     # I7
       andi $t9, $s3, 1     # I8
       beq  $t9, $0, skip   # I9
       add  $t4, $t1, $s1   # I10
       lw   $s4, 0($t4)     # I11
       sub  $s3, $s3, $s4   # I12
       sw   $s3, 0($t3)     # I13
       addi $s5, $s5, 1     # I14
skip:  addi $t0, $t0, 1     # I15
       j    loop            # I16
end:
```

Assuming a 5-stage MIPS pipeline and all elements in array *A* are positive odd integers, answer the following questions. You need to count until the last stage of instruction I16.

(a) How many cycles does this code segment take to complete its execution in the first iteration (I1 to I16) in an ideal pipeline, that is, one with no delays? [2]

Answer: 16 + (5 – 1) = **20**

AY2020/21 Sem2 Q16(b)

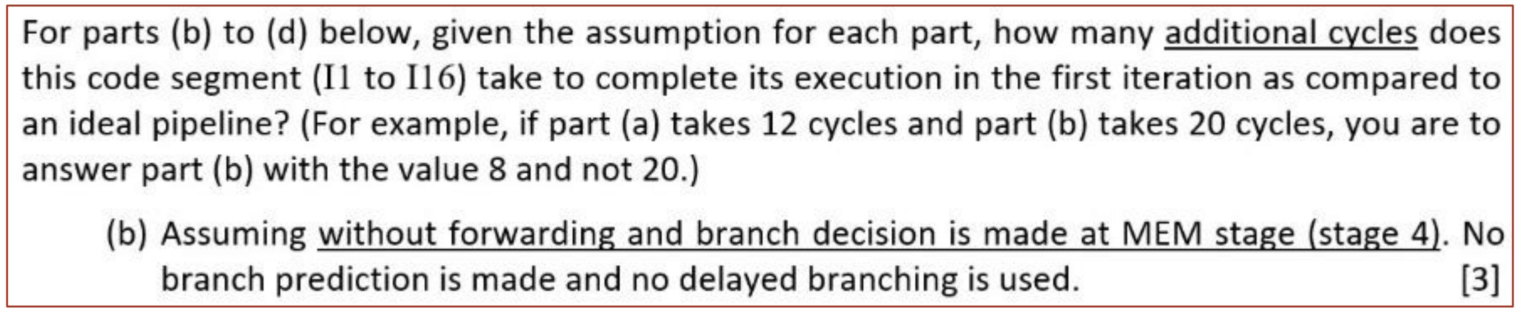


Q16. Pipelining [14 marks]

Refer to the following MIPS code which is the same as the one in question 14. Here, we only look at instructions I1 to I16.

(b)

```
       add  $s5, $0, $0     # I1
       add  $t0, $0, $0     # I2
loop:  slt  $t8, $t0, $s2   # I3   +2
       beq  $t8, $0, end    # I4   +2
       sll  $t1, $t0, 2     # I5   +3
       add  $t3, $t1, $s0   # I6   +2
       lw   $s3, 0($t3)     # I7   +2
       andi $t9, $s3, 1     # I8   +2
       beq  $t9, $0, skip   # I9   +2
       add  $t4, $t1, $s1   # I10  +3
       lw   $s4, 0($t4)     # I11  +2
       sub  $s3, $s3, $s4   # I12  +2
       sw   $s3, 0($t3)     # I13  +2
       addi $s5, $s5, 1     # I14
skip:  addi $t0, $t0, 1     # I15
       j    loop            # I16
end:                              24
```



For parts (b) to (d) below, given the assumption for each part, how many additional cycles does this code segment (I1 to I16) take to complete its execution in the first iteration as compared to an ideal pipeline? (For example, if part (a) takes 12 cycles and part (b) takes 20 cycles, you are to answer part (b) with the value 8 and not 20.)

(b) Assuming without forwarding and branch decision is made at MEM stage (stage 4). No branch prediction is made and no delayed branching is used. [3]

AY2020/21 Sem2 Q16(c)

Q16. Pipelining [14 marks]

Refer to the following MIPS code which is the same as the one in question 14. Here, we only look at instructions I1 to I16.

```
                                             (b)  (c)
      add  $s5, $0, $0      # I1
      add  $t0, $0, $0      # I2
loop: slt  $t8, $t0, $s2    # I3   +2   +2
      beq  $t8, $0, end     # I4   +2   +2
      sll  $t1, $t0, 2      # I5   +3   +1
      add  $t3, $t1, $s0    # I6   +2   +2
      lw   $s3, 0($t3)      # I7   +2   +2
      andi $t9, $s3, 1      # I8   +2   +2
      beq  $t9, $0, skip    # I9   +2   +2
      add  $t4, $t1, $s1    # I10  +3   +1
      lw   $s4, 0($t4)      # I11  +2   +2
      sub  $s3, $s3, $s4    # I12  +2   +2
      sw   $s3, 0($t3)      # I13  +2   +2
      addi $s5, $s5, 1      # I14
skip: addi $t0, $t0, 1      # I15
      j    loop             # I16
end:                              24   22
```

For parts (b) to (d) below, given the assumption for each part, how many additional cycles does this code segment (I1 to I16) take to complete its execution in the first iteration as compared to an ideal pipeline? (For example, if part (a) takes 12 cycles and part (b) takes 20 cycles, you are to answer part (b) with the value 8 and not 20.)

(c) Assuming without forwarding and branch decision is made at ID stage (stage 2). No branch prediction is made and no delayed branching is used. [3]

AY2020/21 Sem2 Q16(d)

Q16. Pipelining [14 marks]

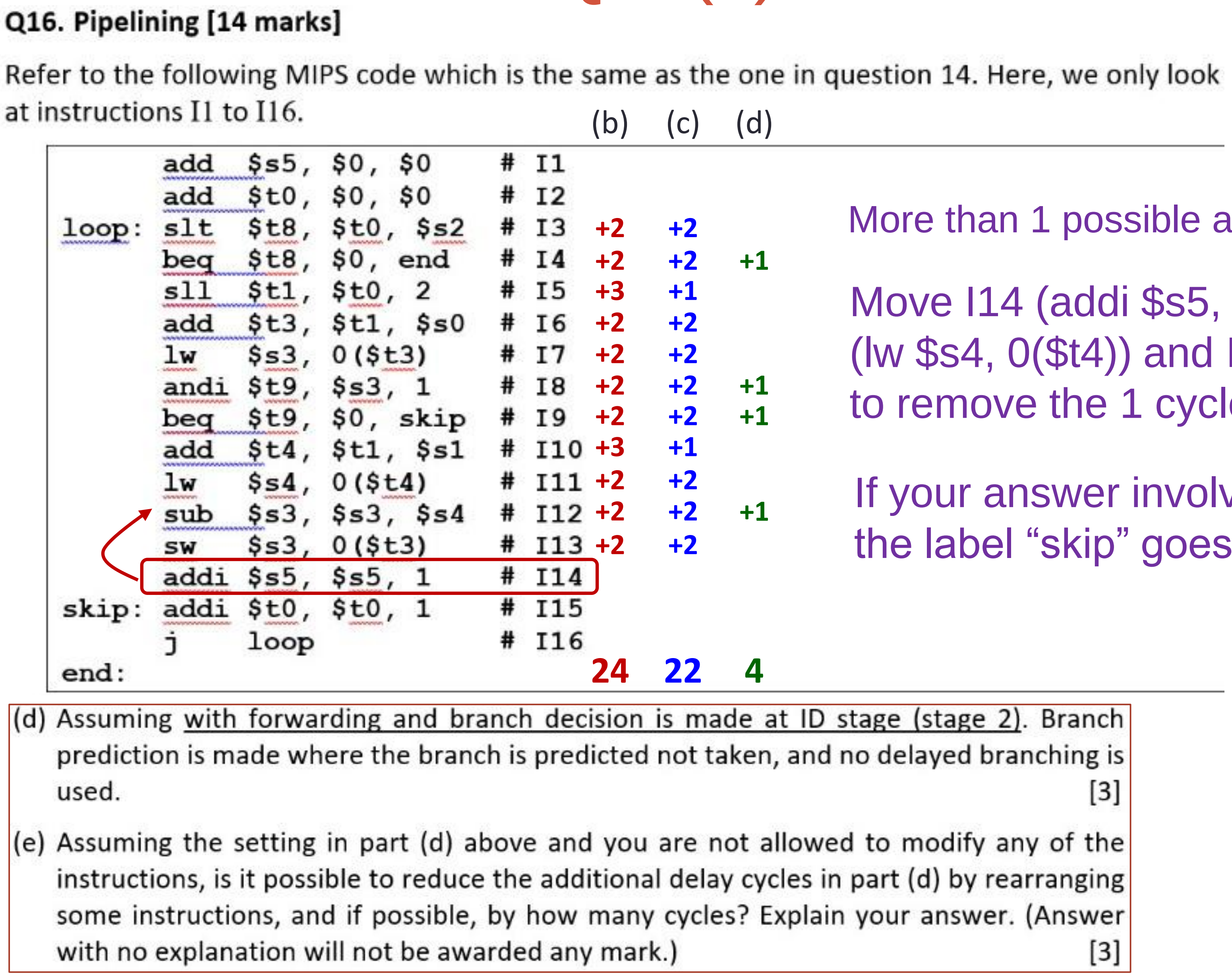
Refer to the following MIPS code which is the same as the one in question 14. Here, we only look at instructions I1 to I16.

```
                                        (b)  (c)  (d)
      add  $s5, $0, $0      # I1
      add  $t0, $0, $0      # I2
loop: slt  $t8, $t0, $s2    # I3       +2   +2
      beq  $t8, $0, end     # I4       +2   +2   +1
      sll  $t1, $t0, 2      # I5       +3   +1
      add  $t3, $t1, $s0    # I6       +2   +2
      lw   $s3, 0($t3)      # I7       +2   +2
      andi $t9, $s3, 1      # I8       +2   +2   +1
      beq  $t9, $0, skip    # I9       +2   +2   +1
      add  $t4, $t1, $s1    # I10      +3   +1
      lw   $s4, 0($t4)      # I11      +2   +2
      sub  $s3, $s3, $s4    # I12      +2   +2   +1
      sw   $s3, 0($t3)      # I13      +2   +2
      addi $s5, $s5, 1      # I14
skip: addi $t0, $t0, 1      # I15
      j    loop             # I16
end:                                   24   22   4
```

For parts (b) to (d) below, given the assumption for each part, how many <u>additional cycles</u> does this code segment (I1 to I16) take to complete its execution in the first iteration as compared to an ideal pipeline? (For example, if part (a) takes 12 cycles and part (b) takes 20 cycles, you are to answer part (b) with the value 8 and not 20.)

(d) Assuming <u>with forwarding and branch decision is made at ID stage (stage 2)</u>. Branch prediction is made where the branch is predicted not taken, and no delayed branching is used. [3]

AY2020/21 Sem2 Q16(e)



Q16. Pipelining [14 marks]

Refer to the following MIPS code which is the same as the one in question 14. Here, we only look at instructions I1 to I16.

```
                                 (b)  (c)  (d)
      add   $s5, $0, $0    # I1
      add   $t0, $0, $0    # I2
loop: slt   $t8, $t0, $s2  # I3   +2   +2
      beq   $t8, $0, end   # I4   +2   +2   +1
      sll   $t1, $t0, 2    # I5   +3   +1
      add   $t3, $t1, $s0  # I6   +2   +2
      lw    $s3, 0($t3)    # I7   +2   +2
      andi  $t9, $s3, 1    # I8   +2   +2   +1
      beq   $t9, $0, skip  # I9   +2   +2   +1
      add   $t4, $t1, $s1  # I10  +3   +1
      lw    $s4, 0($t4)    # I11  +2   +2
      sub   $s3, $s3, $s4  # I12  +2   +2   +1
      sw    $s3, 0($t3)    # I13  +2   +2
      addi  $s5, $s5, 1    # I14
skip: addi  $t0, $t0, 1    # I15
      j     loop           # I16
end:                             24   22    4
```

(d) Assuming with forwarding and branch decision is made at ID stage (stage 2). Branch prediction is made where the branch is predicted not taken, and no delayed branching is used. [3]

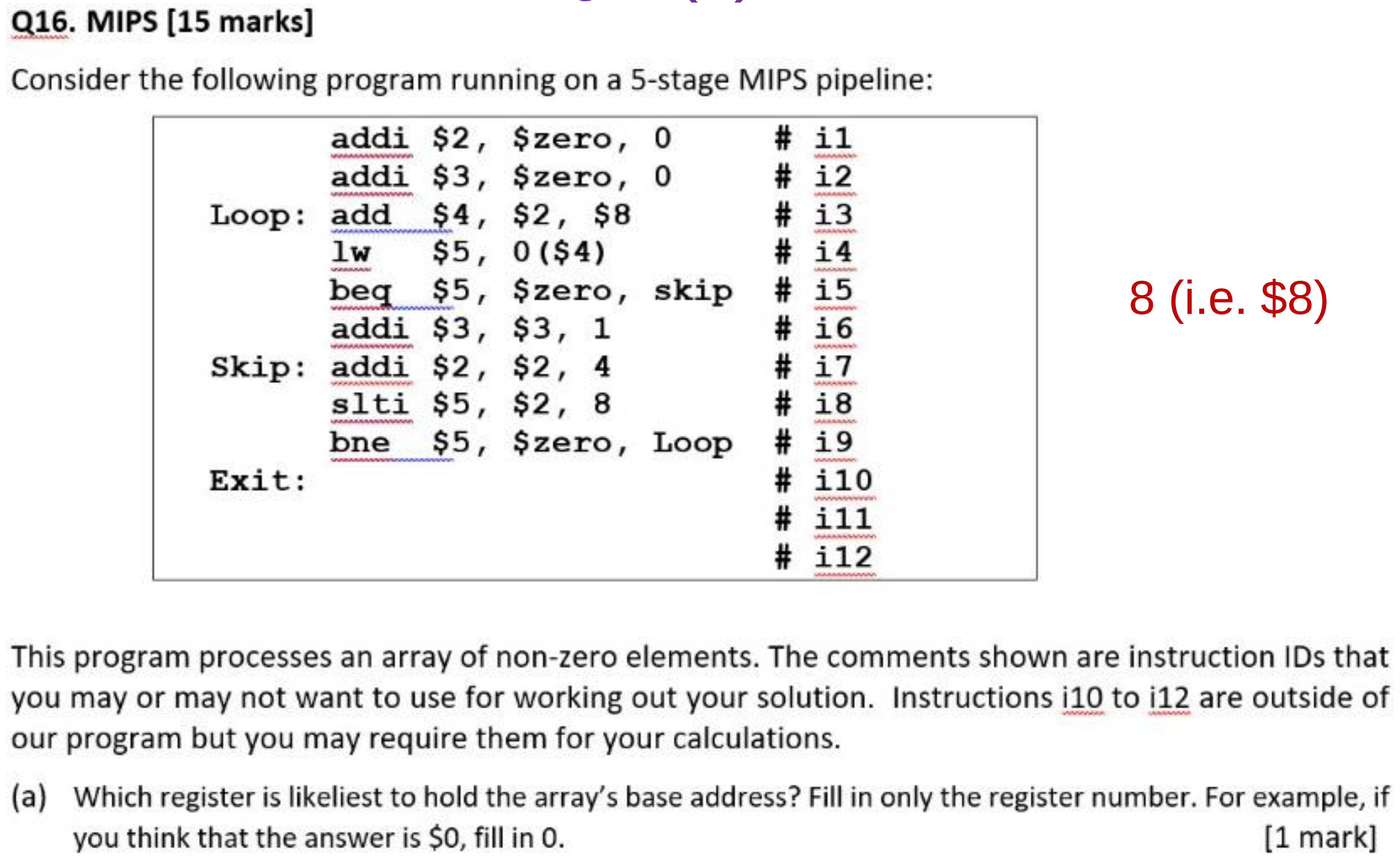
(e) Assuming the setting in part (d) above and you are not allowed to modify any of the instructions, is it possible to reduce the additional delay cycles in part (d) by rearranging some instructions, and if possible, by how many cycles? Explain your answer. (Answer with no explanation will not be awarded any mark.) [3]

More than 1 possible answer.

Move I14 (addi $s5, $s5, 1) to between I11 (lw $s4, 0($t4)) and I12 (sub $s3, $s3, $s4) to remove the 1 cycle of delay at I12.

If your answer involves I15, explain where the label "skip" goes to.

AY2023/24 Sem1 Q16(a)



Q16. MIPS [15 marks]

Consider the following program running on a 5-stage MIPS pipeline:

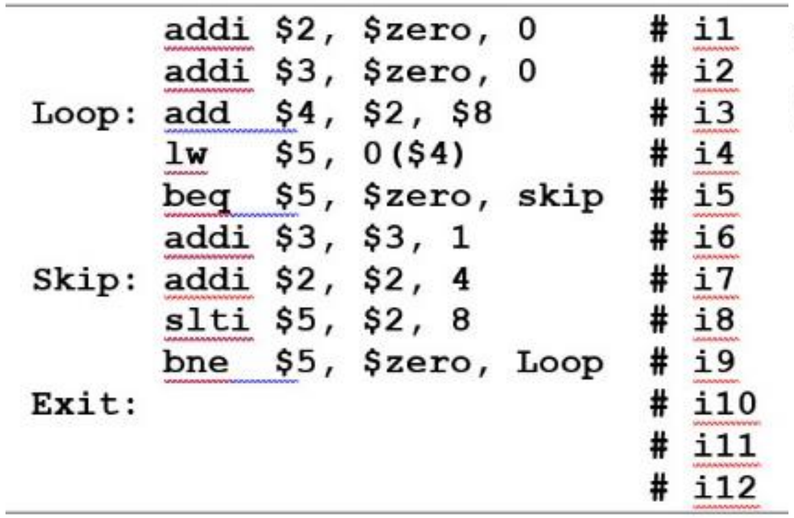
```
       addi $2, $zero, 0      # i1
       addi $3, $zero, 0      # i2
Loop:  add  $4, $2, $8        # i3
       lw   $5, 0($4)         # i4
       beq  $5, $zero, skip   # i5
       addi $3, $3, 1         # i6
Skip:  addi $2, $2, 4         # i7
       slti $5, $2, 8         # i8
       bne  $5, $zero, Loop   # i9
Exit:                         # i10
                              # i11
                              # i12
```

This program processes an array of non-zero elements. The comments shown are instruction IDs that you may or may not want to use for working out your solution. Instructions i10 to i12 are outside of our program but you may require them for your calculations.

(a) Which register is likeliest to hold the array's base address? Fill in only the register number. For example, if you think that the answer is $0, fill in 0. [1 mark]

8 (i.e. $8)

AY2023/24 Sem1 Q16(b)

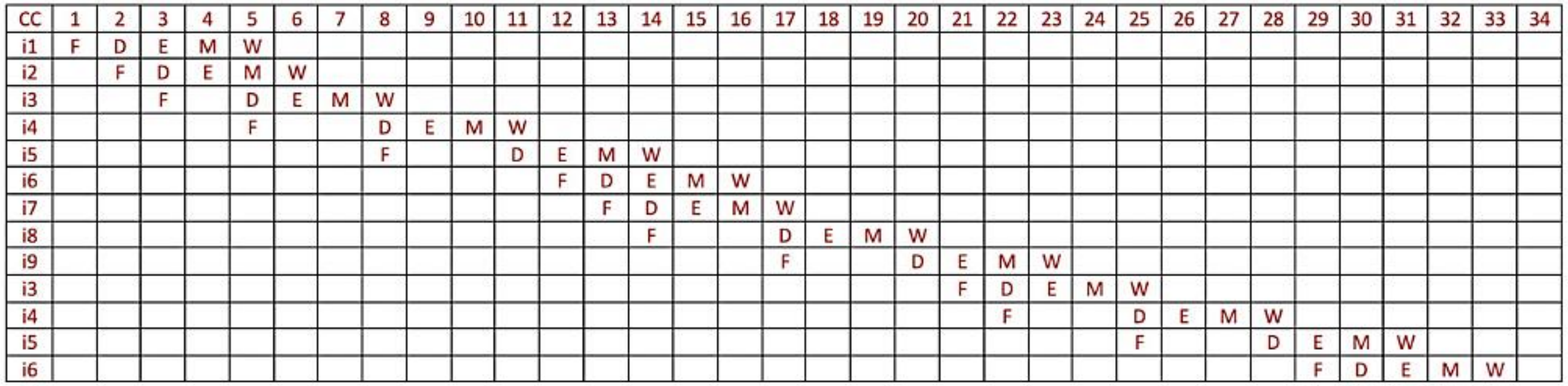


```
       addi $2, $zero, 0       # i1
       addi $3, $zero, 0       # i2
Loop:  add  $4, $2, $8         # i3
       lw   $5, 0($4)          # i4
       beq  $5, $zero, skip    # i5
       addi $3, $3, 1          # i6
Skip:  addi $2, $2, 4          # i7
       slti $5, $2, 8          # i8
       bne  $5, $zero, Loop    # i9
Exit:                          # i10
                               # i11
                               # i12
```

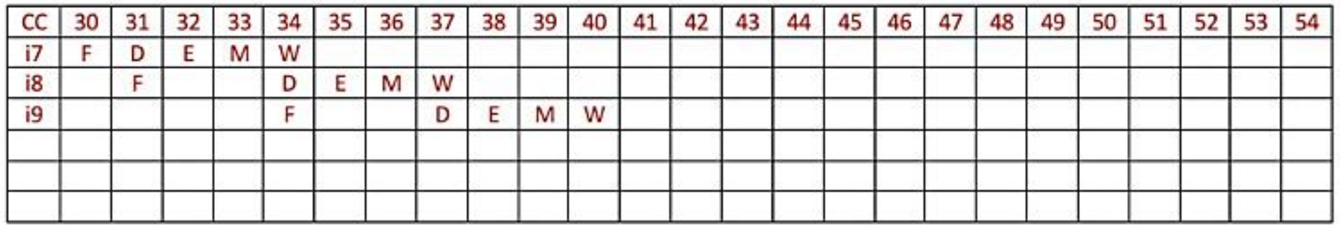
For all your answers below, count only up to the last cycle of the final time that i9 is executed.

(b) Assuming that this program is run on a pipeline without forwarding but with early branching in the ID stage and no branch prediction strategies or delay slots, how many cycles does it take to run the program above to completion? [5 marks]

40 cycles



CC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
i1	F	D	E	M	W																													
i2		F	D	E	M	W																												
i3			F		D	E	M	W																										
i4					F			D	E	M	W																							
i5								F			D	E	M	W																				
i6												F	D	E	M	W																		
i7													F	D	E	M	W																	
i8														F			D	E	M	W														
i9																	F			D	E	M	W											
i3																					F	D	E	M	W									
i4																						F			D	E	M	W						
i5																									F			D	E	M	W			
i6																													F	D	E	M	W	



CC	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
i7	F	D	E	M	W																				
i8		F			D	E	M	W																	
i9					F			D	E	M	W														

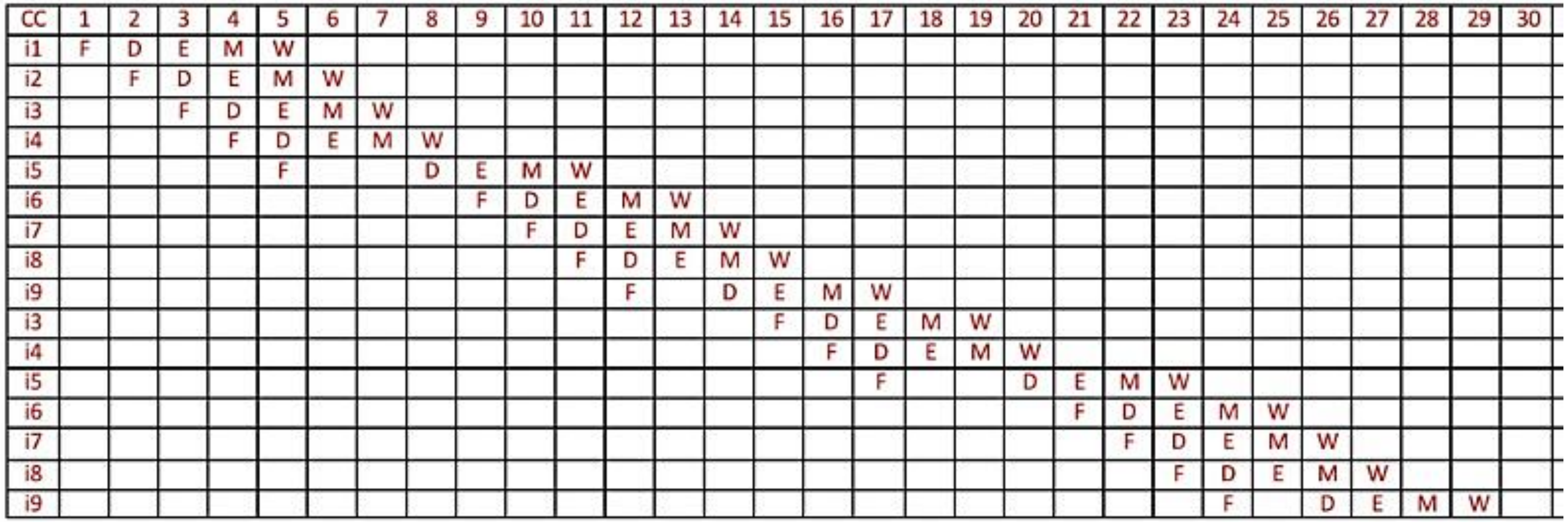
AY2023/24 Sem1 Q16(c)

```
       addi  $2, $zero, 0      # i1
       addi  $3, $zero, 0      # i2
Loop:  add   $4, $2, $8        # i3
       lw    $5, 0($4)         # i4
       beq   $5, $zero, skip   # i5
       addi  $3, $3, 1         # i6
Skip:  addi  $2, $2, 4         # i7
       slti  $5, $2, 8         # i8
       bne   $5, $zero, Loop   # i9
Exit:                          # i10
                               # i11
                               # i12
```

For all your answers below, count only up to the last cycle of the final time that i9 is executed.

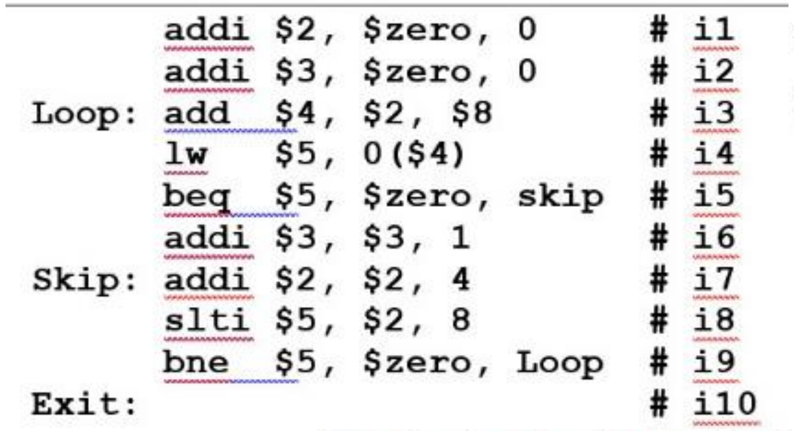
(c) Assuming that this program is run on a pipeline with forwarding and early branching in the decode stage, and with no branch prediction strategies or delay slots, how many cycles does it take to run the program above? [5 marks]

29 cycles



CC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
i1	F	D	E	M	W																									
i2		F	D	E	M	W																								
i3			F	D	E	M	W																							
i4				F	D	E	M	W																						
i5					F			D	E	M	W																			
i6									F	D	E	M	W																	
i7										F	D	E	M	W																
i8											F	D	E	M	W															
i9												F		D	E	M	W													
i3															F	D	E	M	W											
i4																F	D	E	M	W										
i5																	F			D	E	M	W							
i6																					F	D	E	M	W					
i7																						F	D	E	M	W				
i8																							F	D	E	M	W			
i9																								F		D	E	M	W	

AY2023/24 Sem1 Q16(d)

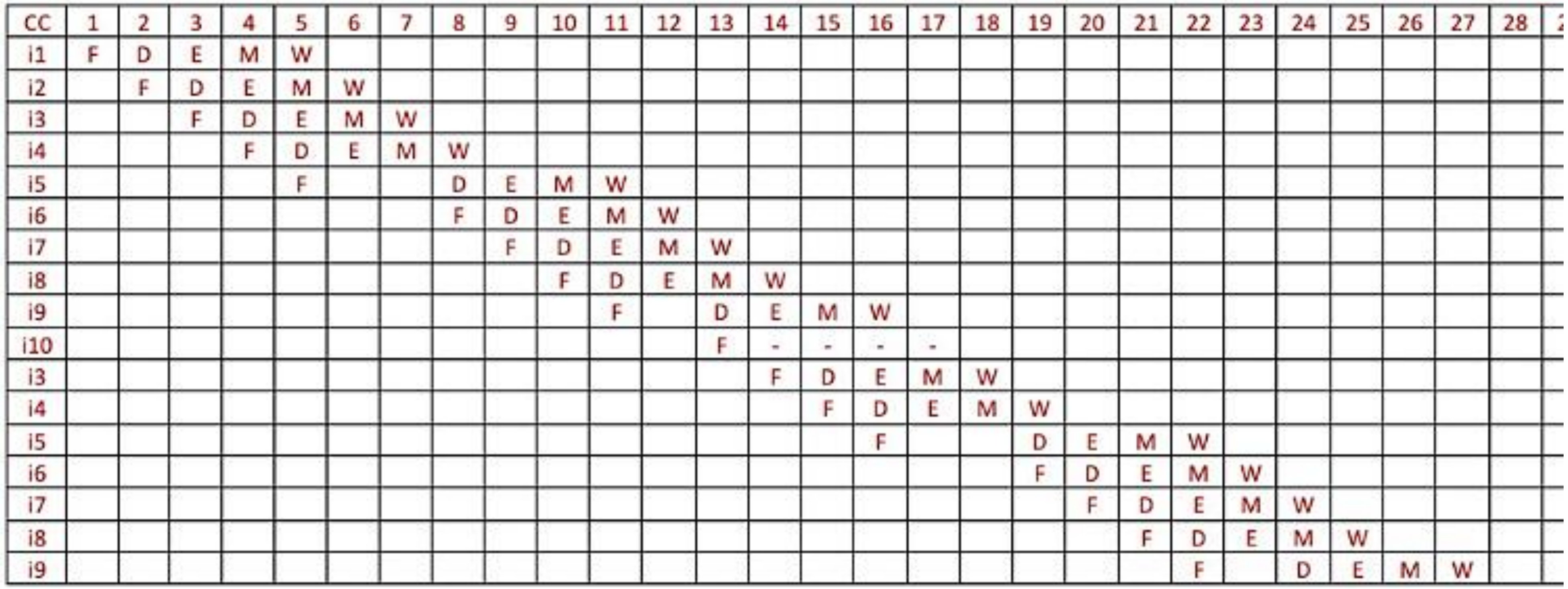


```
       addi $2, $zero, 0      # i1
       addi $3, $zero, 0      # i2
Loop:  add  $4, $2, $8        # i3
       lw   $5, 0($4)         # i4
       beq  $5, $zero, skip   # i5
       addi $3, $3, 1         # i6
Skip:  addi $2, $2, 4         # i7
       slti $5, $2, 8         # i8
       bne  $5, $zero, Loop   # i9
Exit:                         # i10
```

For all your answers below, count only up to the last cycle of the final time that i9 is executed.

(d) Assuming that this program is run on a pipeline with forwarding and early branching and a predict-not-taken prediction strategy, how many cycles does it take to run the program above? [4 marks]

27 cycles



CC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
i1	F	D	E	M	W																							
i2		F	D	E	M	W																						
i3			F	D	E	M	W																					
i4				F	D	E	M	W																				
i5					F			D	E	M	W																	
i6								F	D	E	M	W																
i7									F	D	E	M	W															
i8										F	D	E	M	W														
i9											F		D	E	M	W												
i10													F	-	-	-	-											
i3														F	D	E	M	W										
i4															F	D	E	M	W									
i5																F			D	E	M	W						
i6																			F	D	E	M	W					
i7																				F	D	E	M	W				
i8																					F	D	E	M	W			
i9																						F		D	E	M	W	

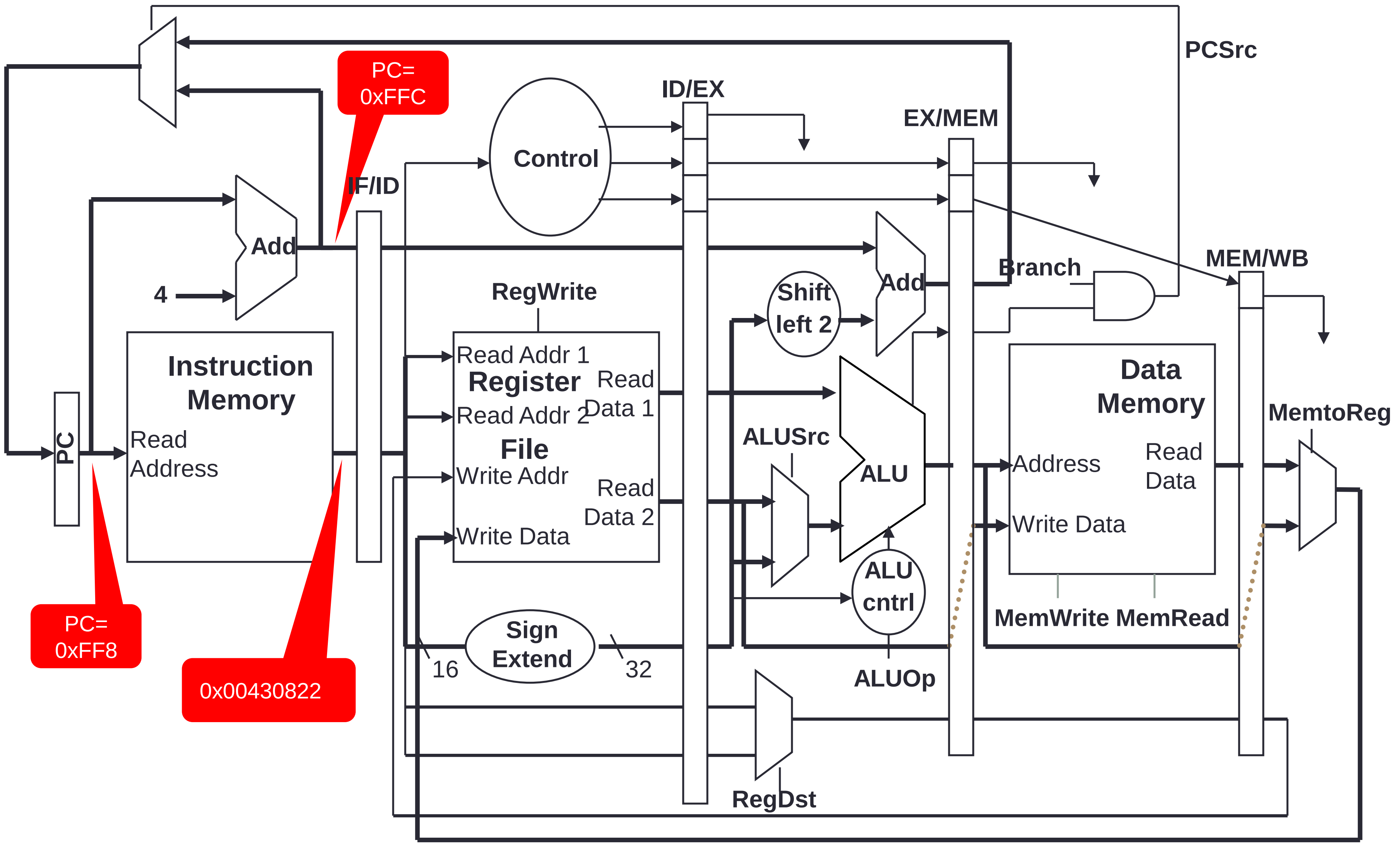
Additional Exercise: Tracing signals and data

Using Powerpoint animation, give the full details of the signals and data bits (highlighting those lines that are active) of the following code that is executed in a pipelined MIPS. Assume that L is at address 0x00000100 and the beq instruction is at address 0x00001000.

```
sub $1, $2, $3
lw  $6, 4($7)
beq $4, $5, L
```

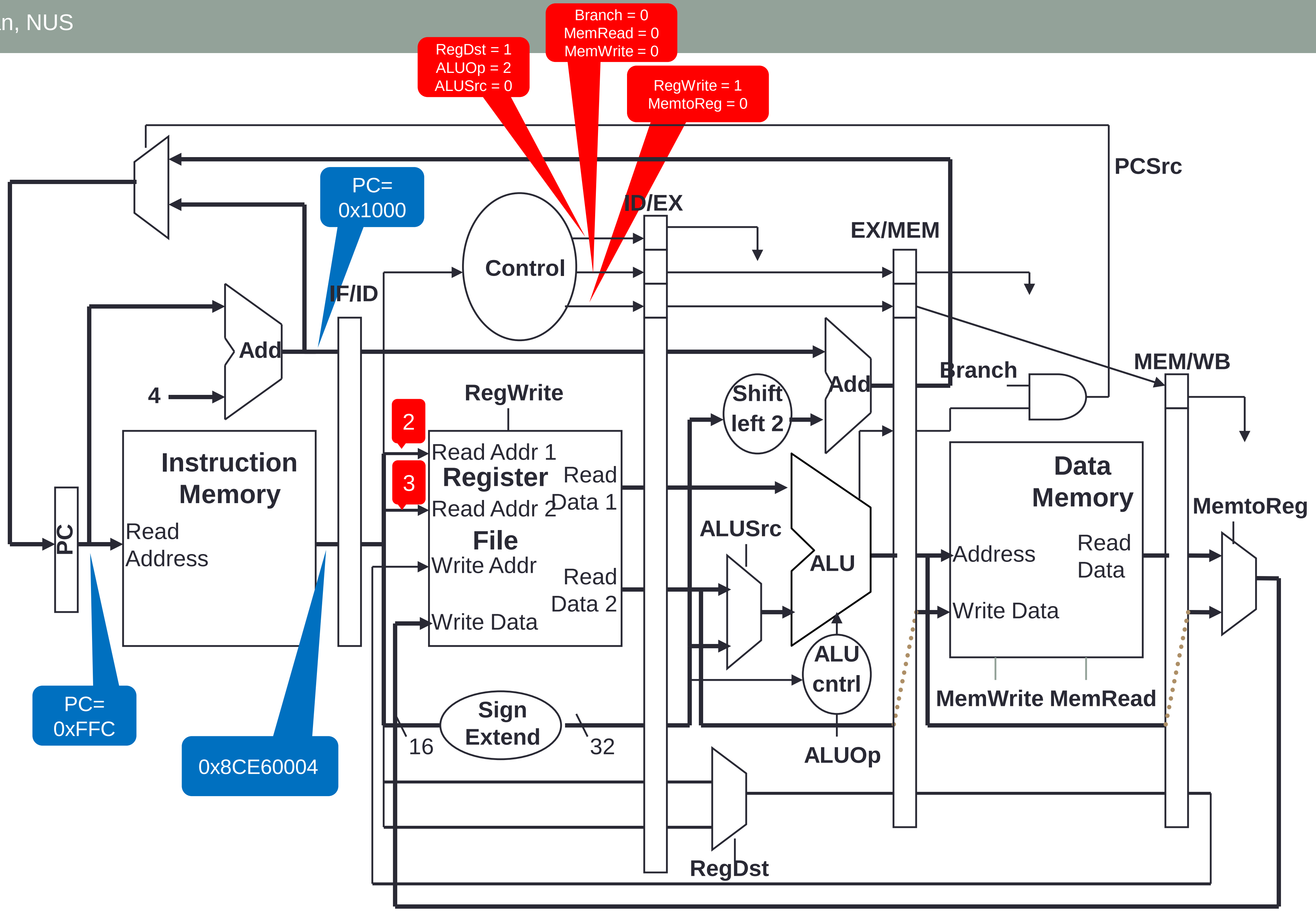
Credit: A/P Anandha Gopalan

Q1


PCSrc
PC= 0xFFC
ID/EX
EX/MEM
Control
IF/ID
Add
4
Instruction Memory
Read Address
PC
RegWrite
Read Addr 1
Register File
Read Addr 2
Write Addr
Write Data
Read Data 1
Read Data 2
Shift left 2
Add
Branch
MEM/WB
ALUSrc
ALU
ALU cntrl
ALUOp
Data Memory
Address
Read Data
Write Data
MemWrite
MemRead
MemtoReg
Sign Extend
16
32
RegDst
PC= 0xFF8
0x00430822

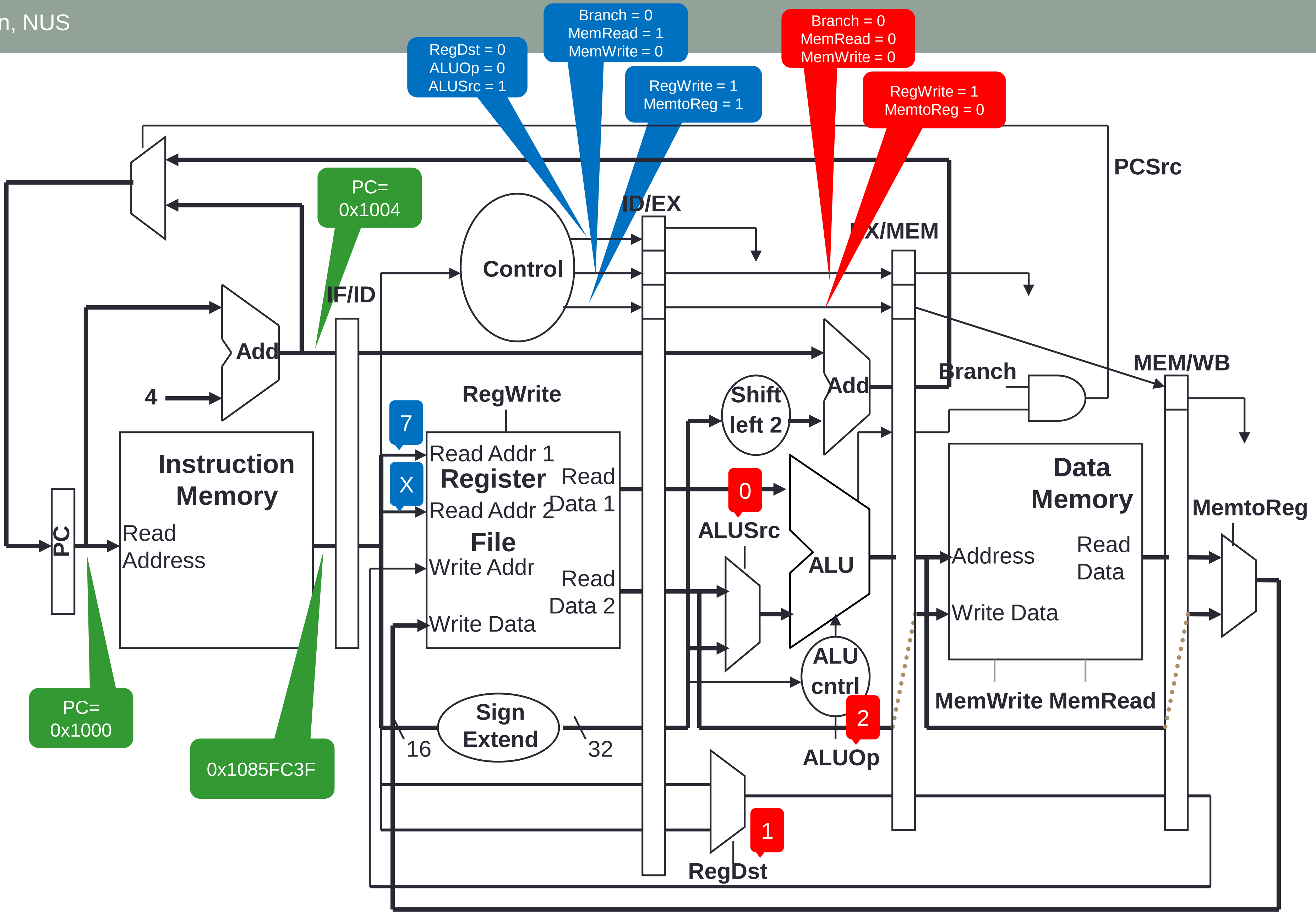
`sub $1, $2, $3`

Q1



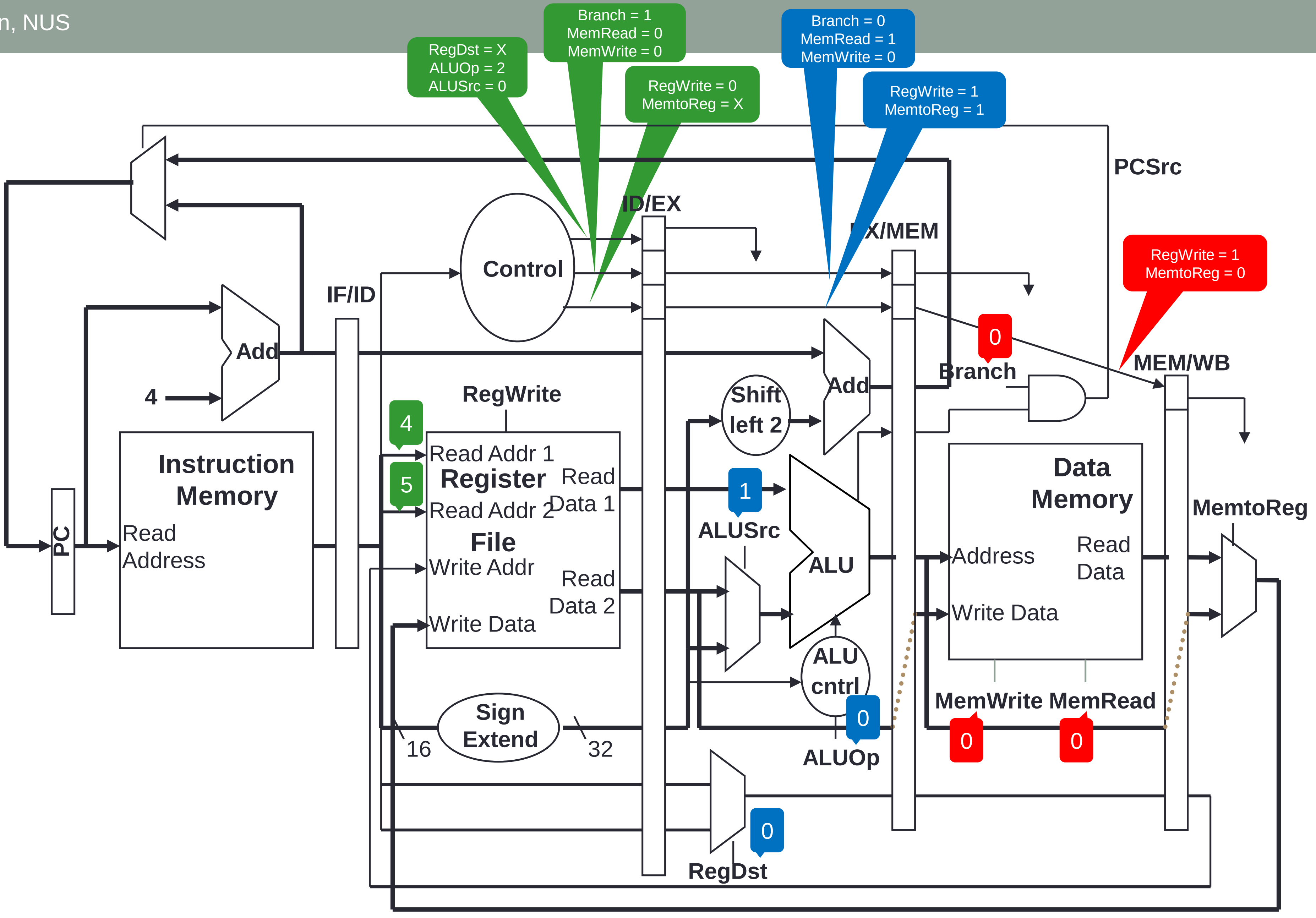
`lw $6, 4($7)`　　`sub $1, $2, $3`

Q1



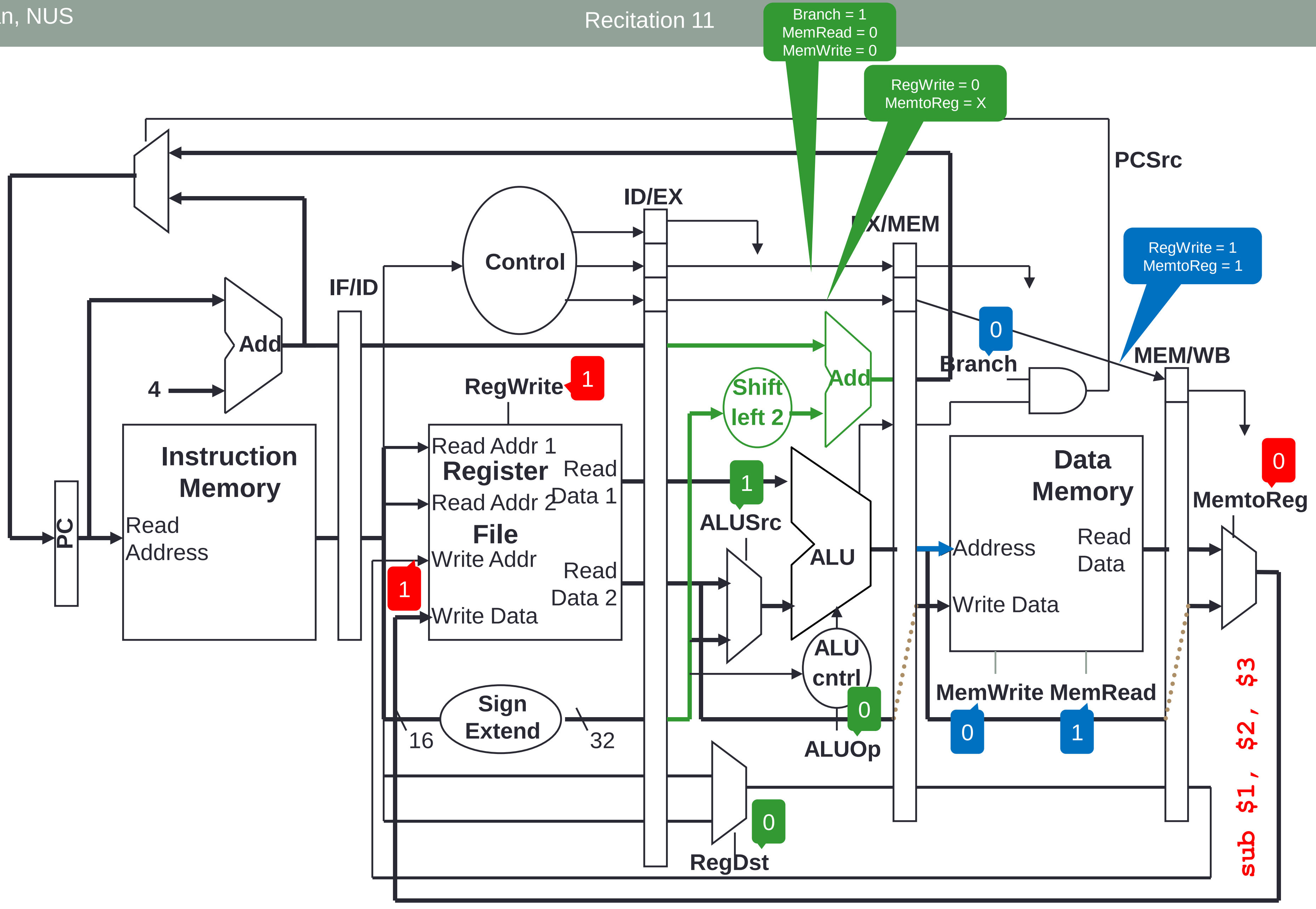
beq $4, $5, L   lw $6, 4($7)   sub $1, $2, $3

Q1



`beq $4, $5, L`   `lw $6, 4($7)`   `sub $1, $2, $3`

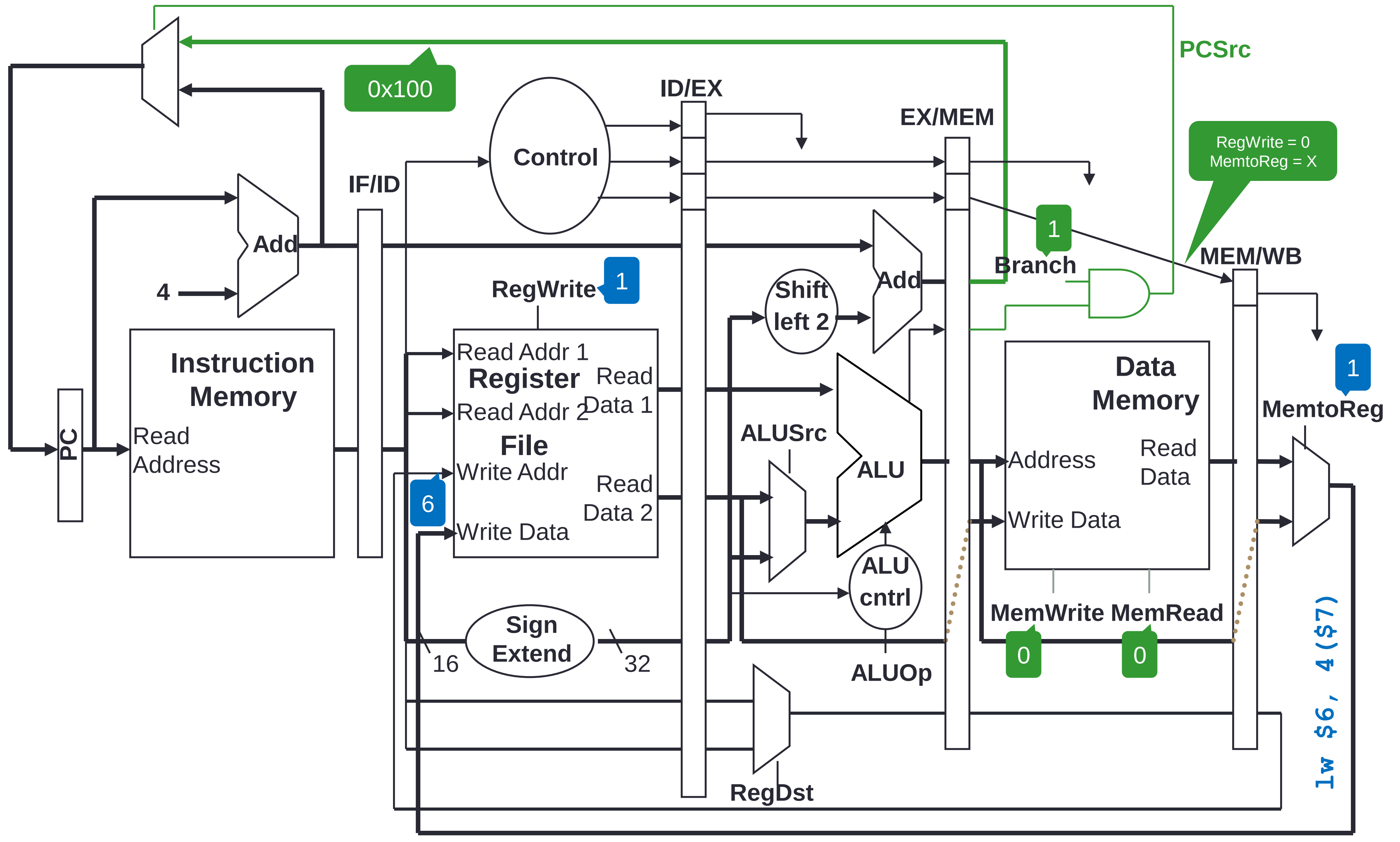
Q1



beq $4, $5, L

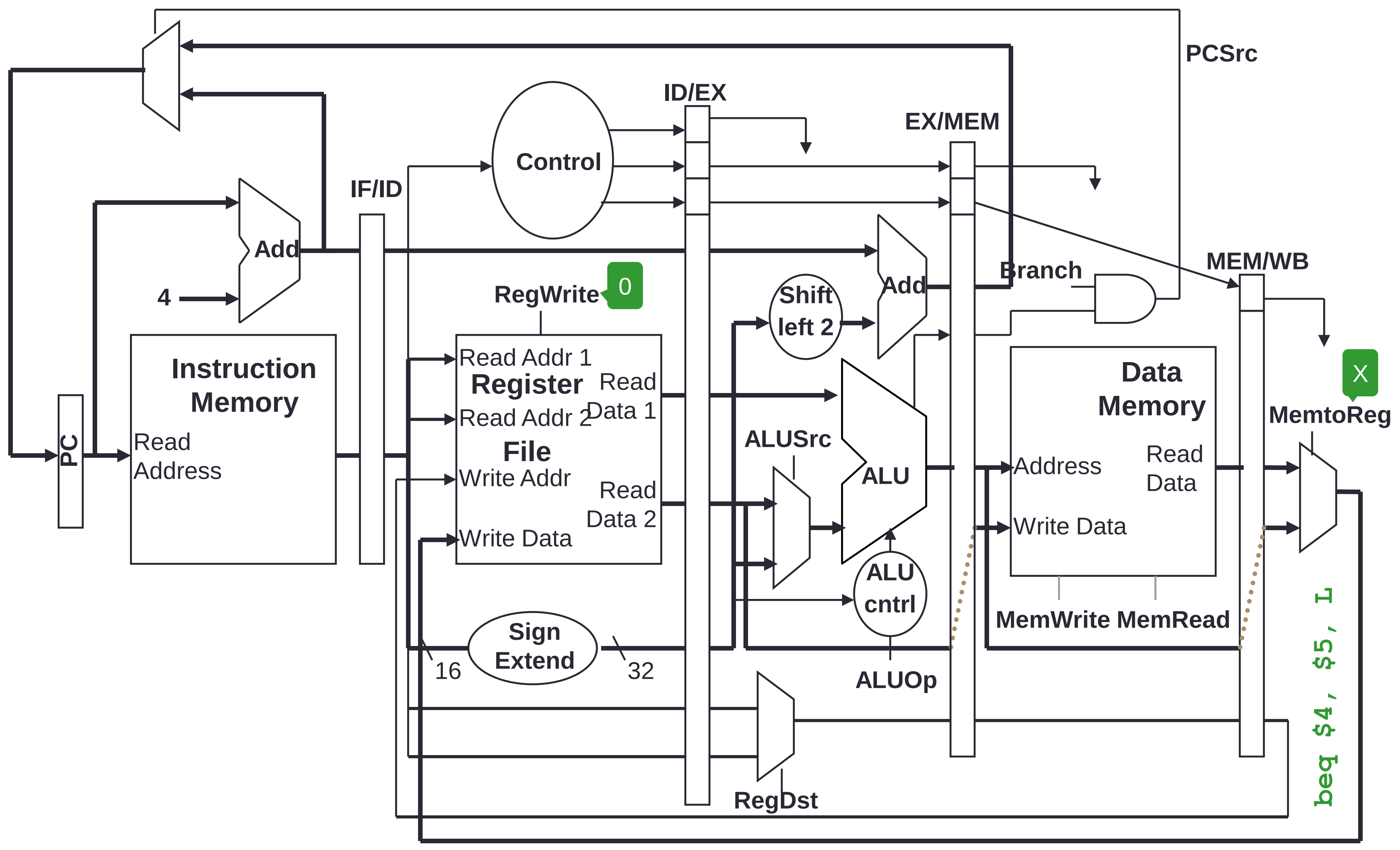
lw $6, 4($7)

Q1


0x100
PCSrc
ID/EX
EX/MEM
MEM/WB
IF/ID
Control
Add
4
PC
Instruction Memory
Read Address
RegWrite
1
Read Addr 1
Register File
Read Addr 2
Write Addr
Write Data
Read Data 1
Read Data 2
6
Sign Extend
16
32
Shift left 2
Add
ALUSrc
ALU
ALU cntrl
ALUOp
RegDst
Branch
1
RegWrite = 0
MemtoReg = X
Data Memory
Address
Read Data
Write Data
MemWrite
MemRead
0
0
MemtoReg
1
lw $6, 4($7)

```
beq $4, $5, L
```

Q1


PCSrc
ID/EX
EX/MEM
IF/ID
Control
Add
4
Instruction Memory
Read Address
PC
RegWrite
0
Read Addr 1
Register
Read Addr 2
File
Write Addr
Write Data
Read Data 1
Read Data 2
Shift left 2
Add
ALUSrc
ALU
ALU cntrl
ALUOp
Sign Extend
16
32
RegDst
Branch
MEM/WB
Data Memory
Address
Read Data
Write Data
MemWrite
MemRead
X
MemtoReg
beq $4, $5, L

Next week – Final recitation!

- Cache

End of File