CSE 2021
COMPUTER ORGANIZATION

HUGH CHESSER
CSE B 1012U

W12-M

Pipelined Control (2)

— Control lines in pipelined implementation are divided into five groups according to the pipeline stage

1. Instruction Fetch: No control needed as the "write control" of PC and "read control" of instruction memory is always asserted.
2. Instruction Decode/Register File Read: No controls needed as the register file in being read during each instruction.
3. Execution/Address Calculation: Control signals are ALUSrc, RegDst, and ALUOp.

   For lw/sw instructions, ALUSrc = 1, RegDst = 0 and ALUOp = 00. For R-type instructions, ALUSrc = 0, RegDst = 1, and ALUOp = 10.
4. Memory Access: Control signals are Branch, MemWrite, and MemRead. For lw instruction, MemRead = 1 and Branch = MemWrite = 0. For sw instruction, MemWrite = 1 and Branch = MemRead = 0. For branch instructions, Branch = 1 and Memwrite = MemRead = 0. For R-type instructions, Branch = MemWrite = MemRead = 0.
5. Write Back: Control signals are MemtoReg. For lw instructions, MemtoReg = 1. For R-type instructions, MemtoReg = 0.

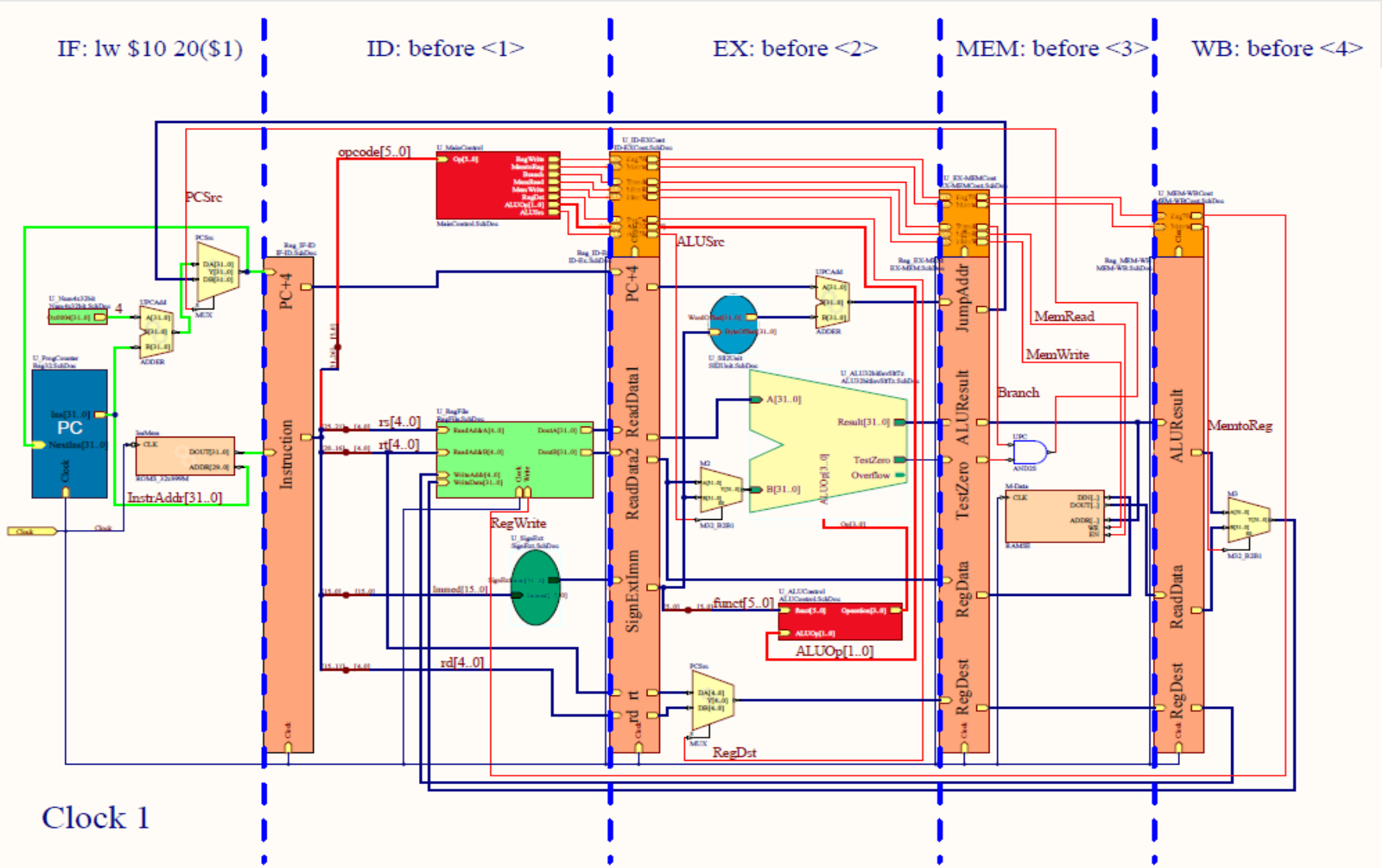
— Pipeline registers are extended to include the control signals for each stage of an instruction.

Activity 4

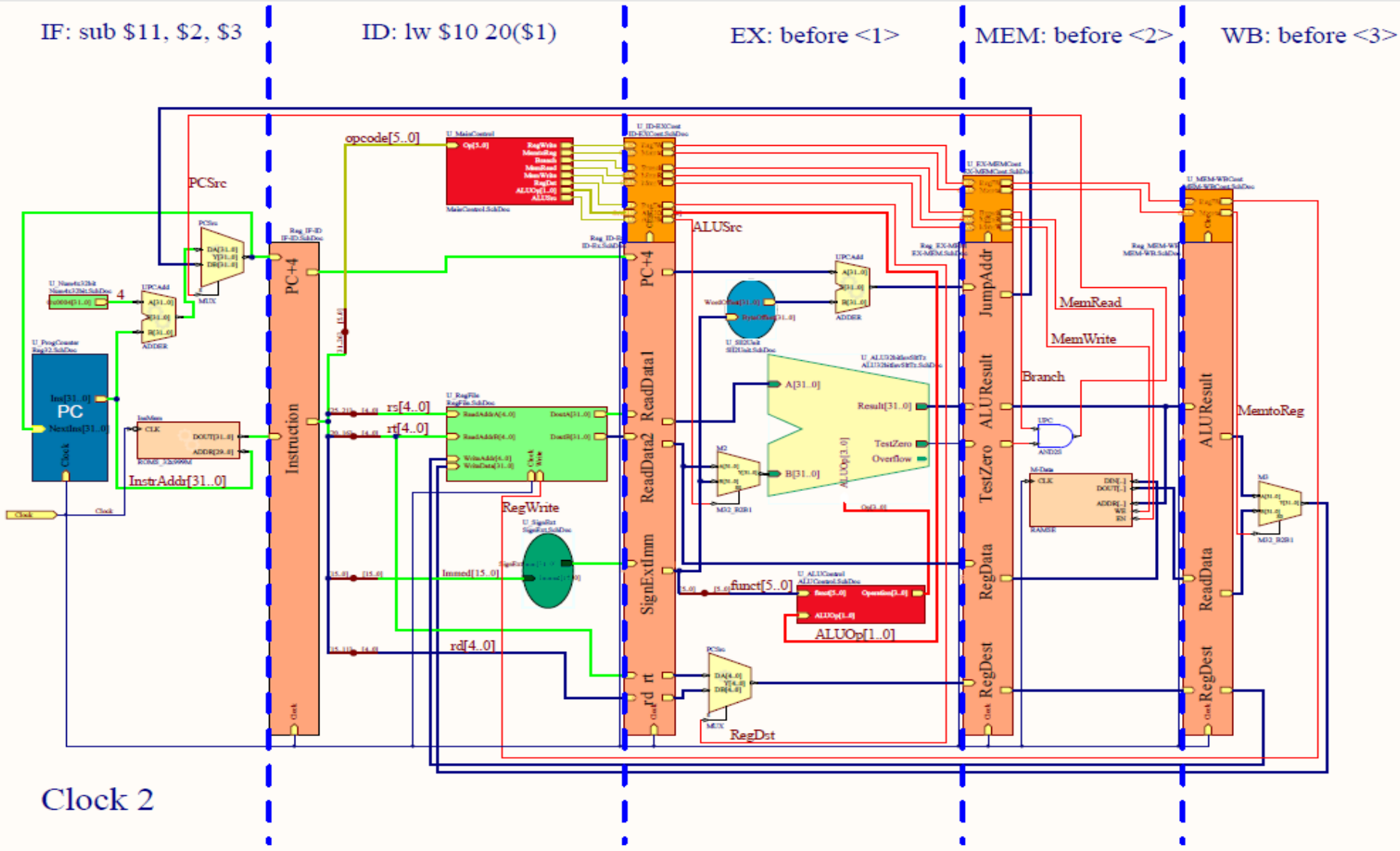
Show the following instructions going through the pipeline:

```
lw $10, 20($1)
sub $11,$2,$3
and $12,$4,$5
or $13,$6,$7
and $14,$8,$9
```

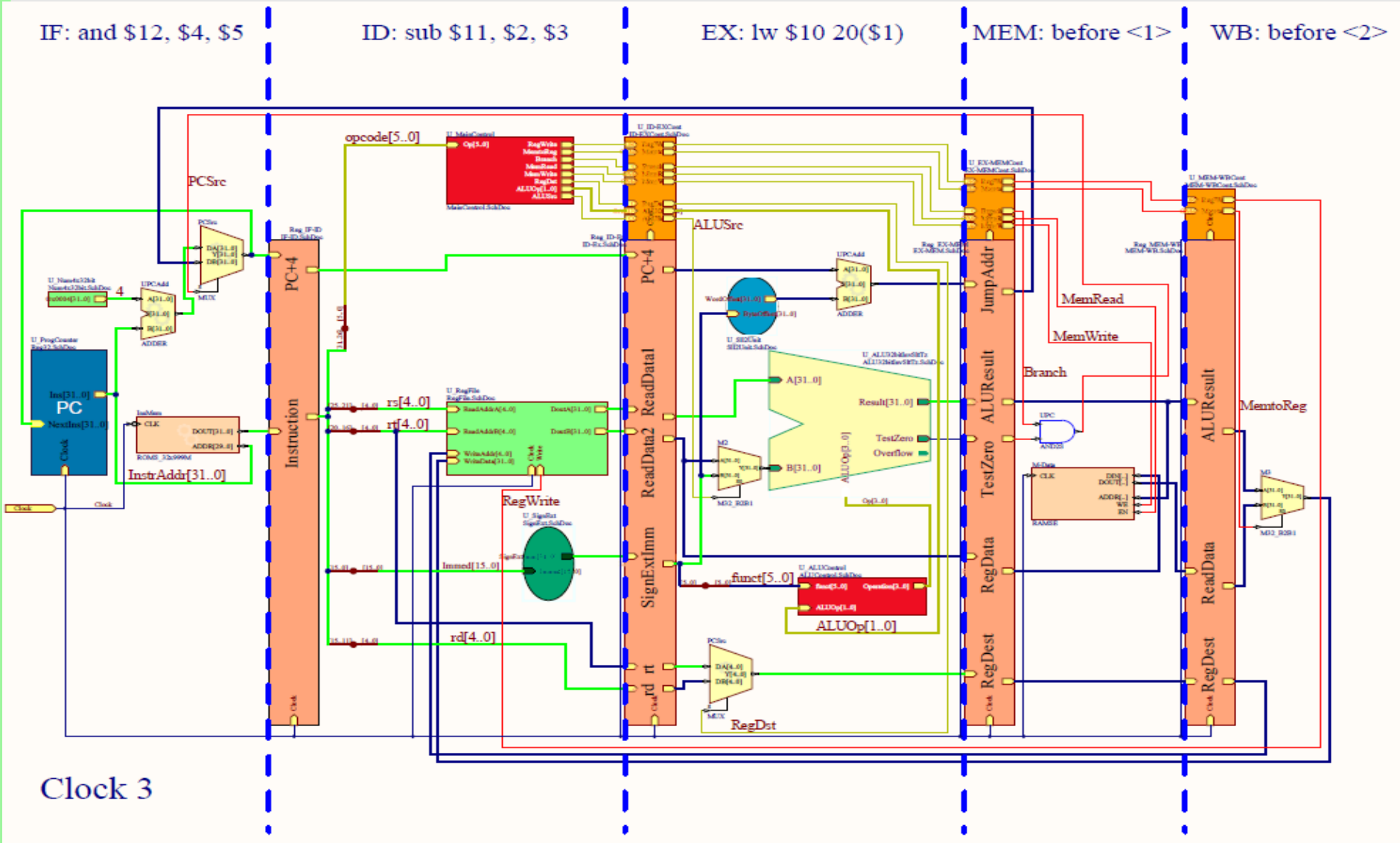
Activity 4: Clock Cycle # 1



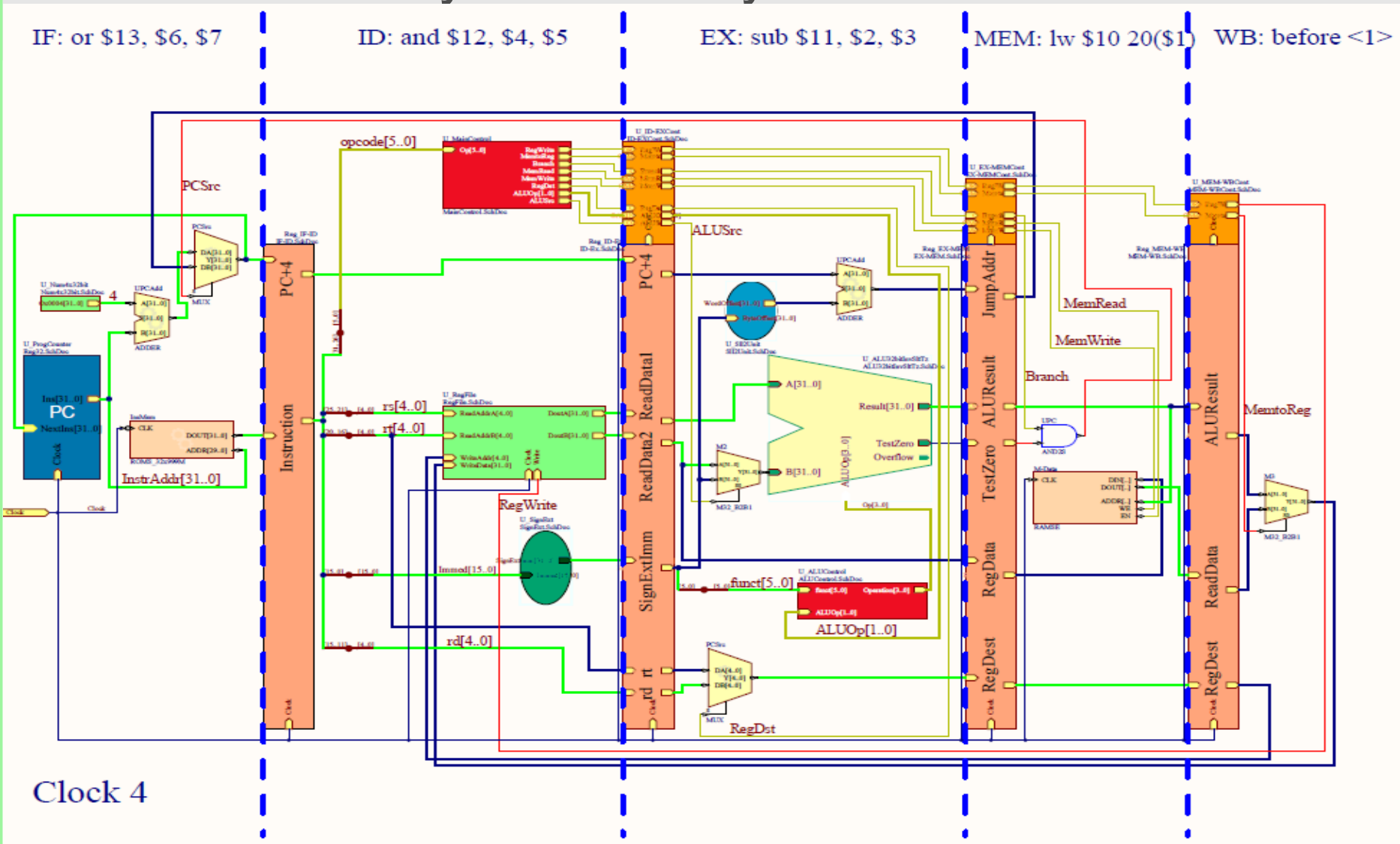
Activity 4: Clock Cycle # 2



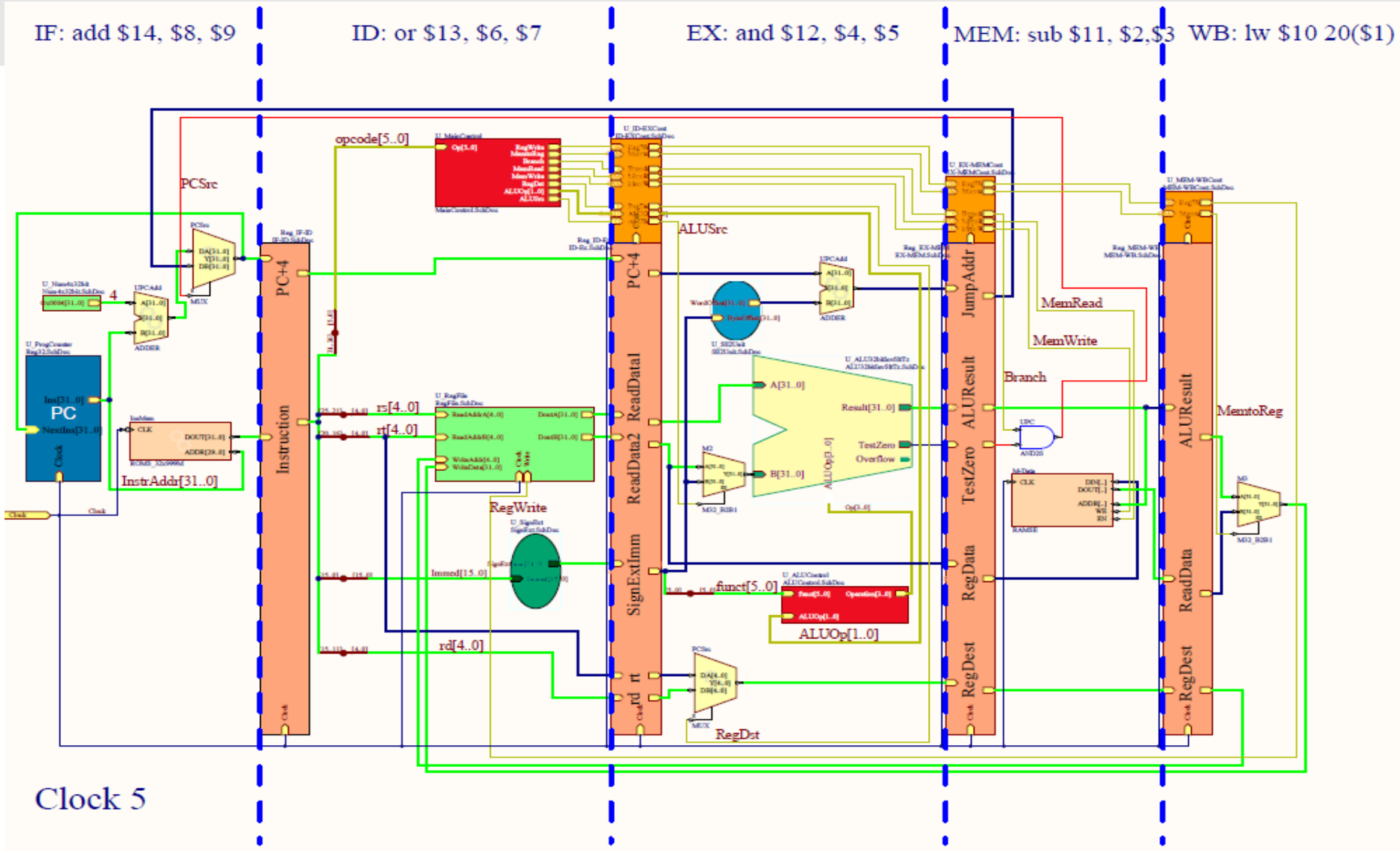
Activity 4: Clock Cycle # 3



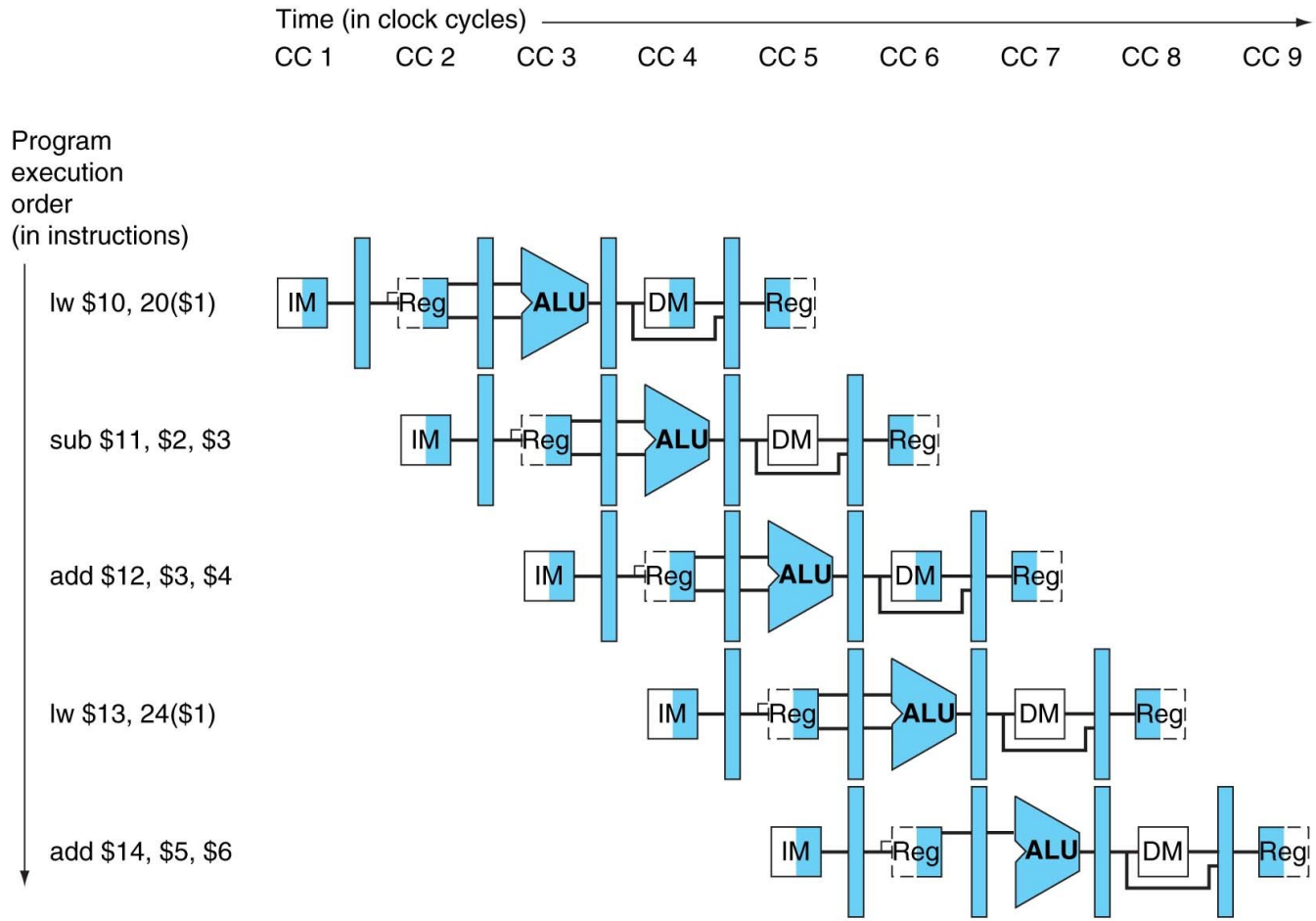
Activity 4: Clock Cycle # 4



Activity 4: Clock Cycle # 5



Pipeline Diagram (Simplified Notation)



Agenda

Topics:

1. Pipelined Control (complete)
2. Data Hazards – Forwarding

Patterson: 4.6, 4.7

Remaining Schedule

11	Nov 14	□	□	M	Pipelining
12	Nov 21	□	□	N	Caches
13	Nov 28	□	Quiz #3	Make-up Labs	
14	Dec 05	□	-	-	No lecture on Wednesday

You're Cordially Invited:

Final Exam

Monday, December 19th

SLH A

14:00 to 17:00

- Remaining Lecture Topics (Exam):
- 4.7 Data Hazards – Forwarding
- 4.8 Control Hazards
- 5.2 Cache Basics

ALU Control Actions

Instruction opcode	ALUOp	Instruction operation	Function code	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	AND	0000
R-type	10	OR	100101	OR	0001
R-type	10	set on less than	101010	set on less than	0111

Action of Pipeline Control Signals

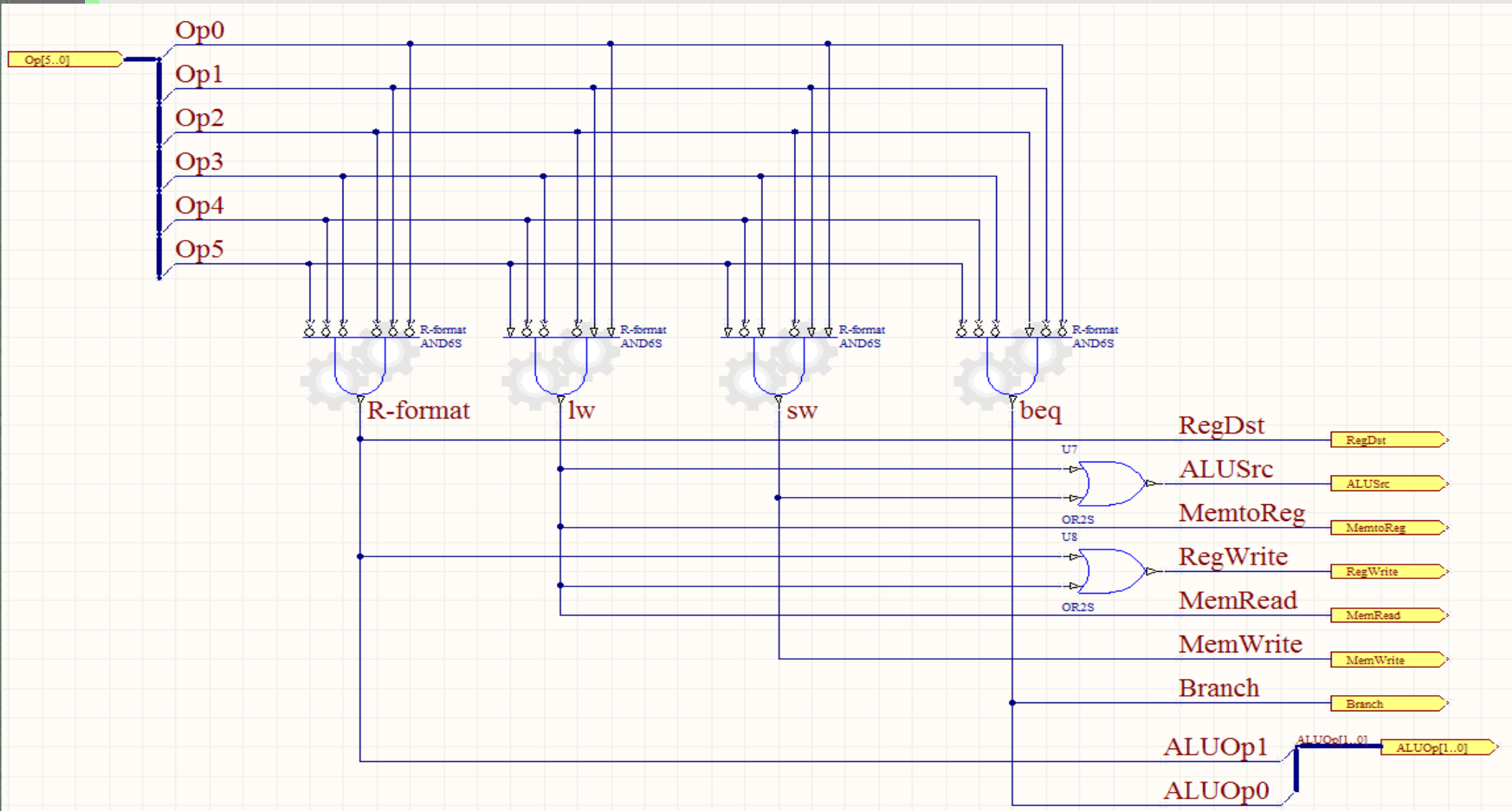
Signal name	Effect when deasserted (0)	Effect when asserted (1)
RegDst	The register destination number for the Write register comes from the rt field (bits 20:16).	The register destination number for the Write register comes from the rd field (bits 15:11).
RegWrite	None.	The register on the Write register input is written with the value on the Write data input.
ALUSrc	The second ALU operand comes from the second register file output (Read data 2).	The second ALU operand is the sign-extended, lower 16 bits of the instruction.
PCSrc	The PC is replaced by the output of the adder that computes the value of PC + 4.	The PC is replaced by the output of the adder that computes the branch target.
MemRead	None.	Data memory contents designated by the address input are put on the Read data output.
MemWrite	None.	Data memory contents designated by the address input are replaced by the value on the Write data input.
MemtoReg	The value fed to the register Write data input comes from the ALU.	The value fed to the register Write data input comes from the data memory.

Control for Pipeline – Arranged by Pipeline Stage

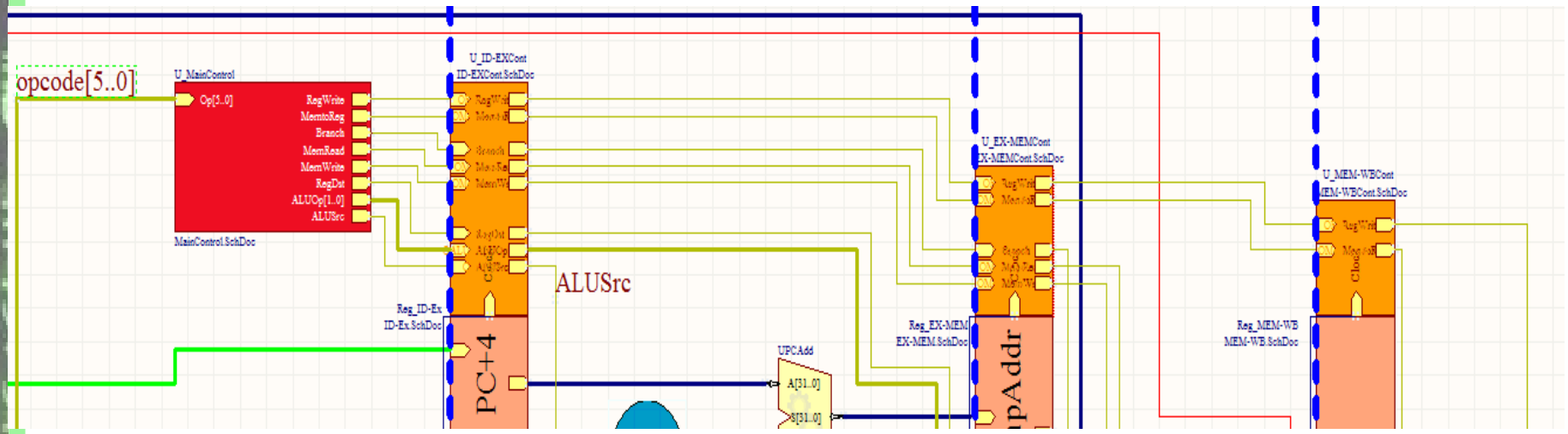
Instruction	Execution/address calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	RegDst	ALUOp1	ALUOp0	ALUSrc	Branch	Mem-Read	Mem-Write	Reg-Write	Memto-Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

Controls same as before for the single or multi-cycle implementations, rearranged according to pipeline stage

Main Control (6)



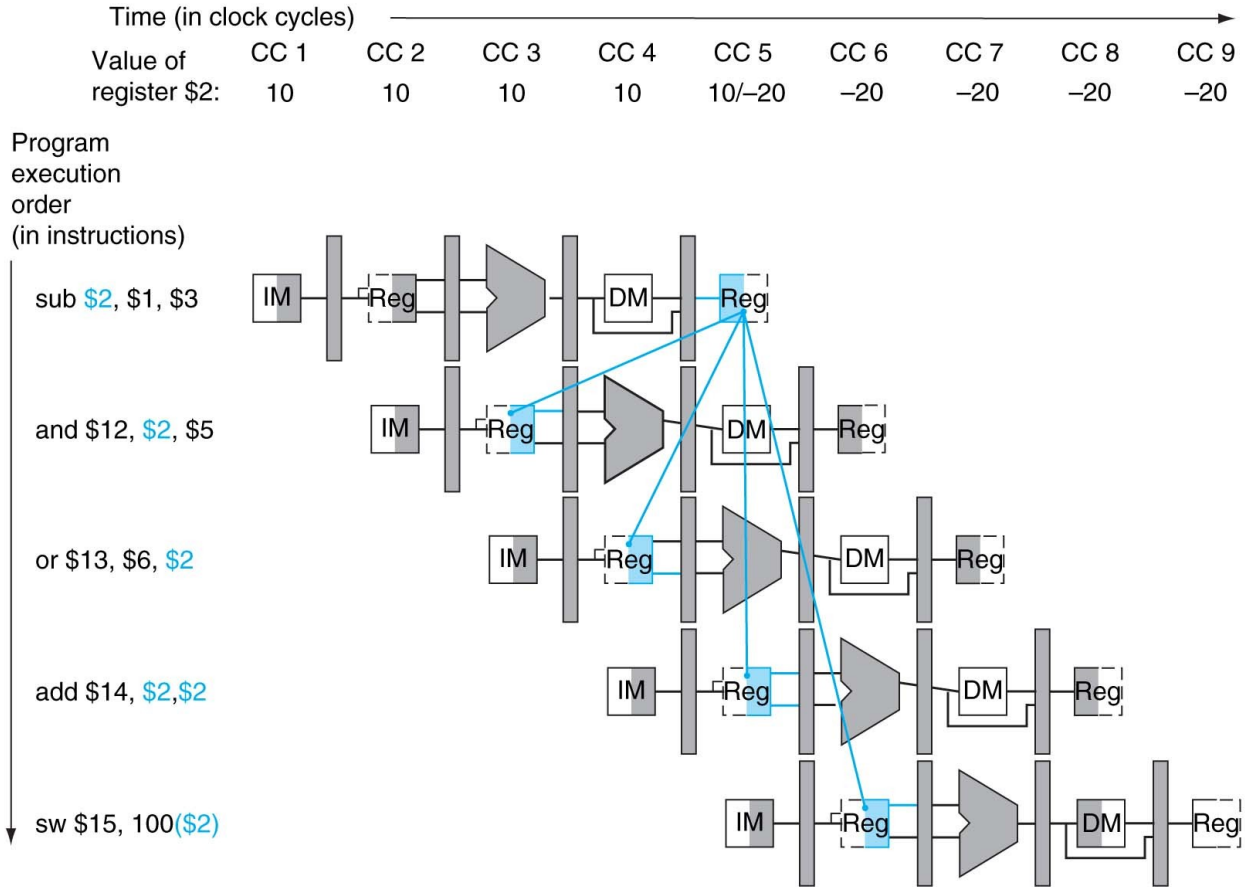
Wiring of Control Outputs for the Pipeline Implementation



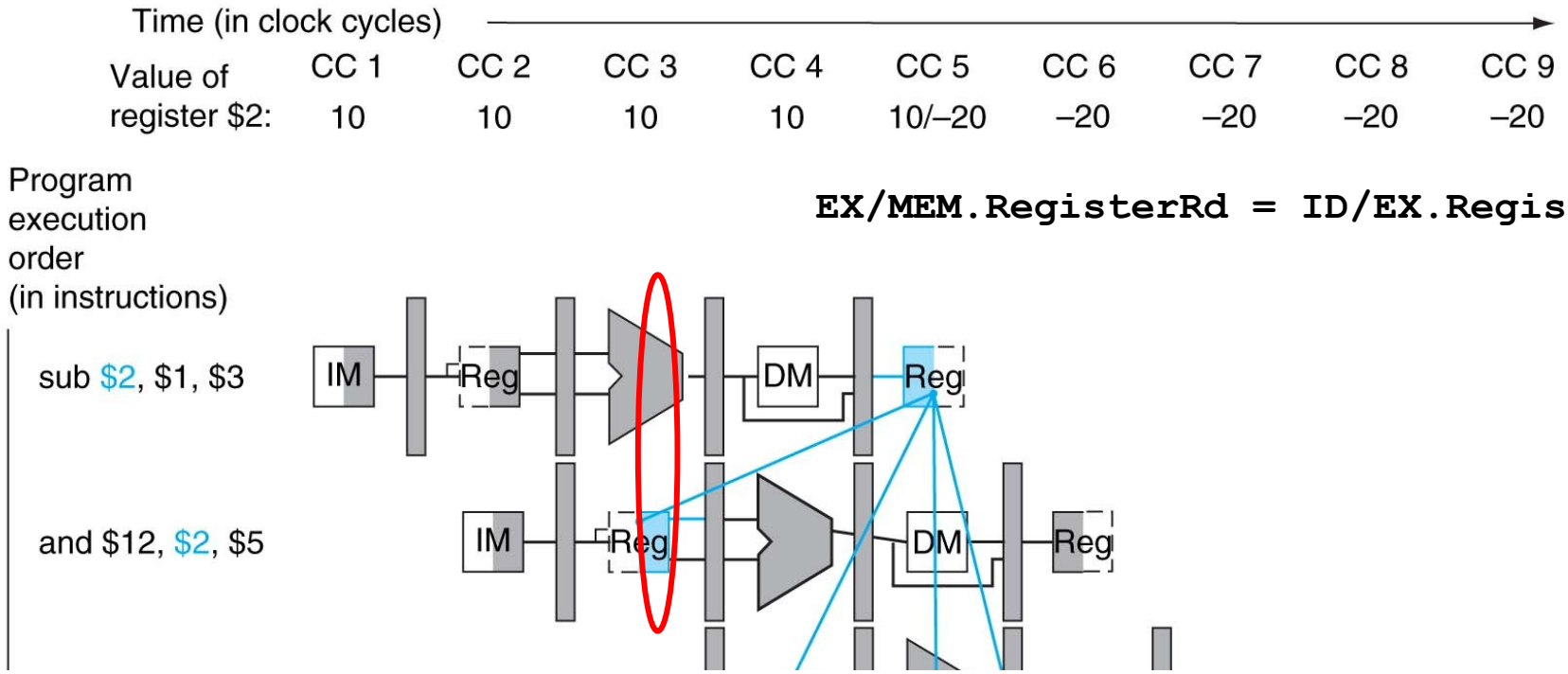
Outputs travel between registers and are wired to correct datapath unit at the appropriate instruction stage

Data Hazards

Consider the following instruction sequence and the resulting pipeline diagram…

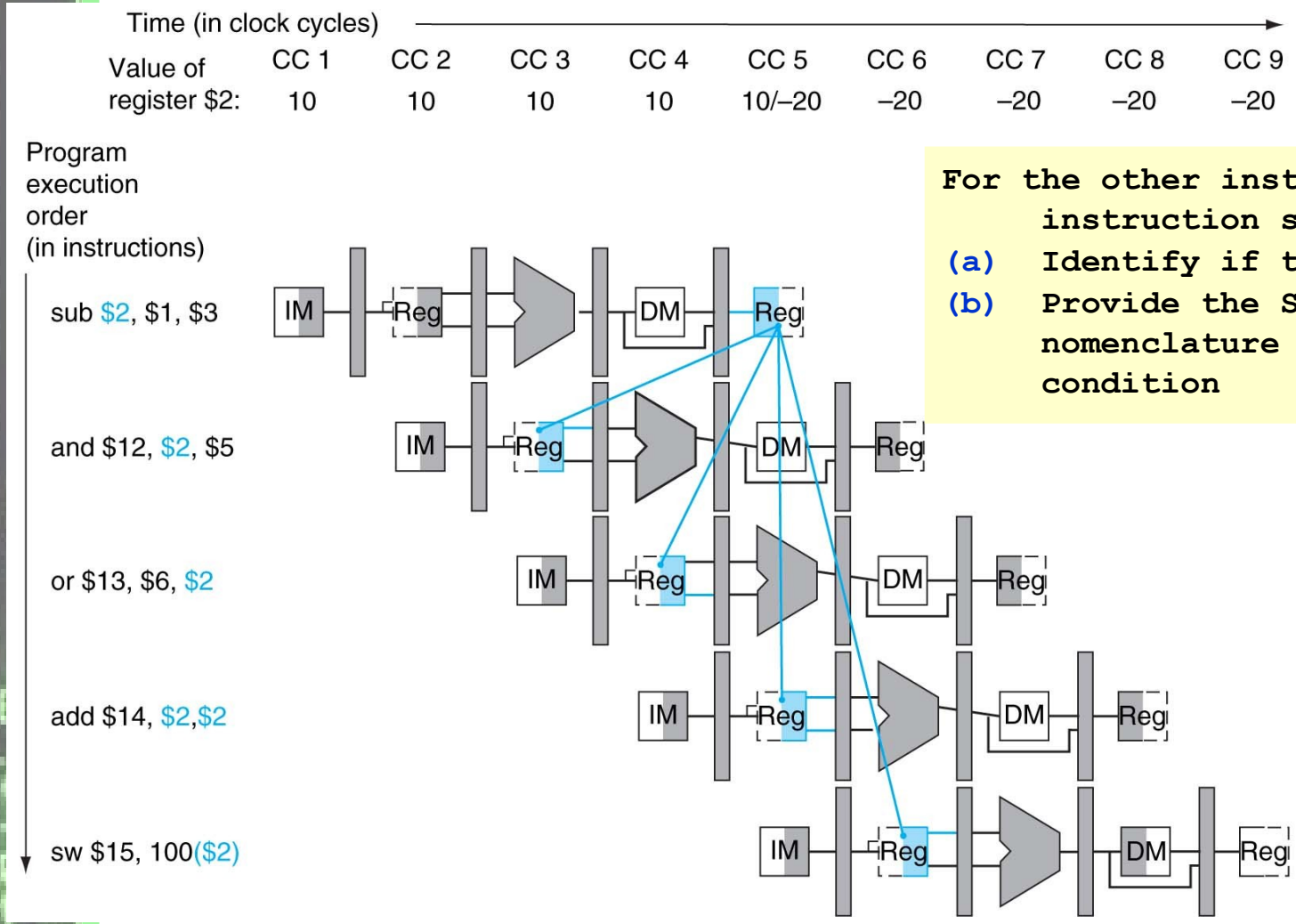


Data Hazard Nomenclature



`EX/MEM.RegisterRd = ID/EX.RegisterRs`

Data Hazard Nomenclature - Activity



For the other instances of $2 in the instruction set:

(a) Identify if there is a data hazard

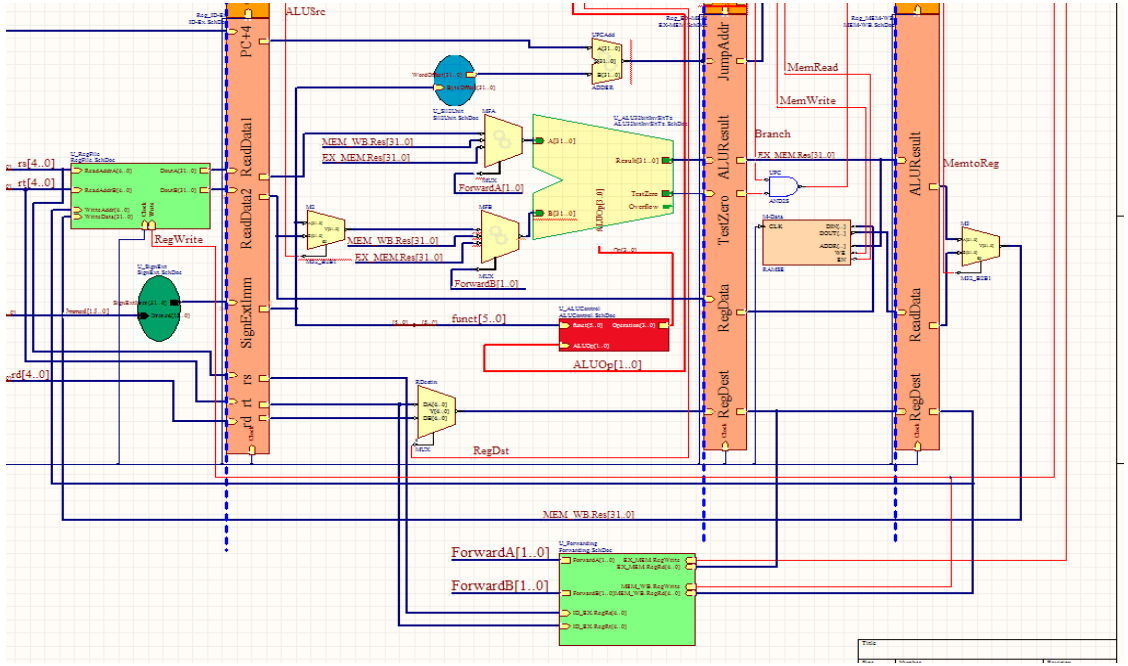
(b) Provide the S1/S2.Sect = S3/S4.Sect nomenclature to define the hazard condition

Forwarding from EX/MEM Pipeline Register

Conditions:

```
(EX/MEM.RegWrite &
EX/MEM.RegisterRd ≠ 0 &
EX/MEM.RegisterRd=ID/EX.RegisterRs) ->
ForwardA = 10

(EX/MEM.RegWrite &
EX/MEM.RegisterRd ≠ 0 &
EX/MEM.RegisterRd=ID/EX.RegisterRt) ->
ForwardB = 10
```




```
sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
Note: Only R-type instructions covered
for forwarding,
no I-type, eg - sw $2, 0($13) (Rd = 0)
```

Field	0	rs	rt	rd	shamt	funct
Bit positions	31:26	25:21	20:16	15:11	10:6	5:0

a. R-type instruction

Field	35 or 43	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

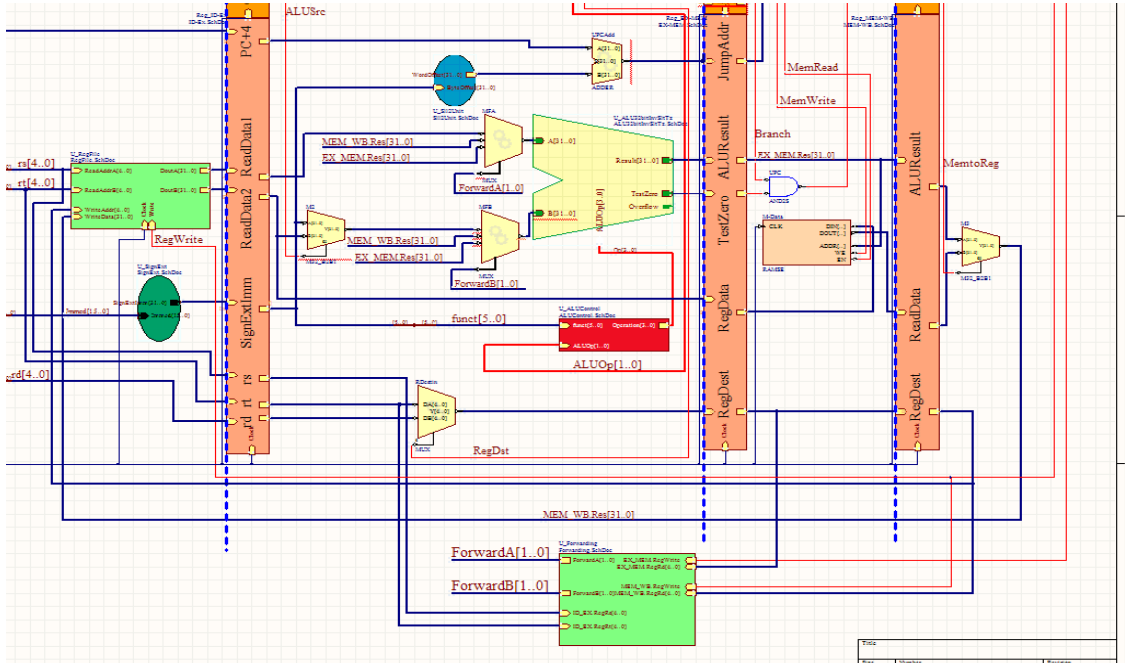
b. Load or store instruction

Forwarding from MEM/WB Pipeline Register

Conditions:

```
(MEM/WB.RegWrite &
MEM/WB.RegisterRd ≠ 0 &
MEM/WB.RegisterRd=ID/EX.RegisterRs) ->
ForwardA = 01

(MEM/WB.RegWrite &
MEM/WB.RegisterRd ≠ 0 &
MEM/WB.RegisterRd=ID/EX.RegisterRt) ->
ForwardB = 01
```




```
sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
[Note: Only R-type instructions
covered for forwarding,
no I-type, eg - sw $2, 0($13) (Rd = 0)
```

Field	0	rs	rt	rd	shamt	funct
Bit positions	31:26	25:21	20:16	15:11	10:6	5:0

a. R-type instruction

Field	35 or 43	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

b. Load or store instruction

Mux Truth Table For Forwarding

Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.