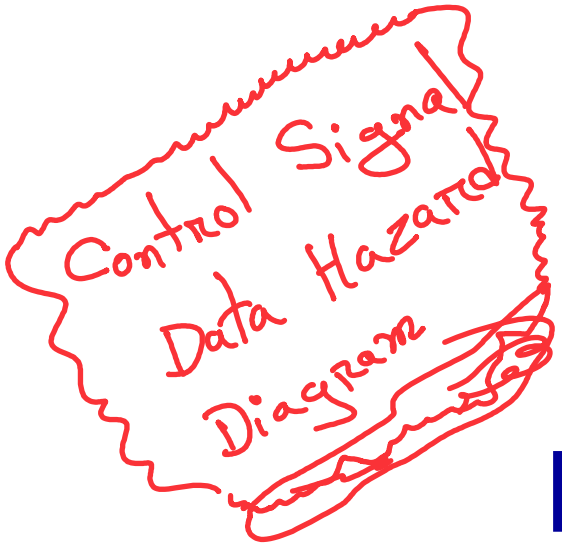
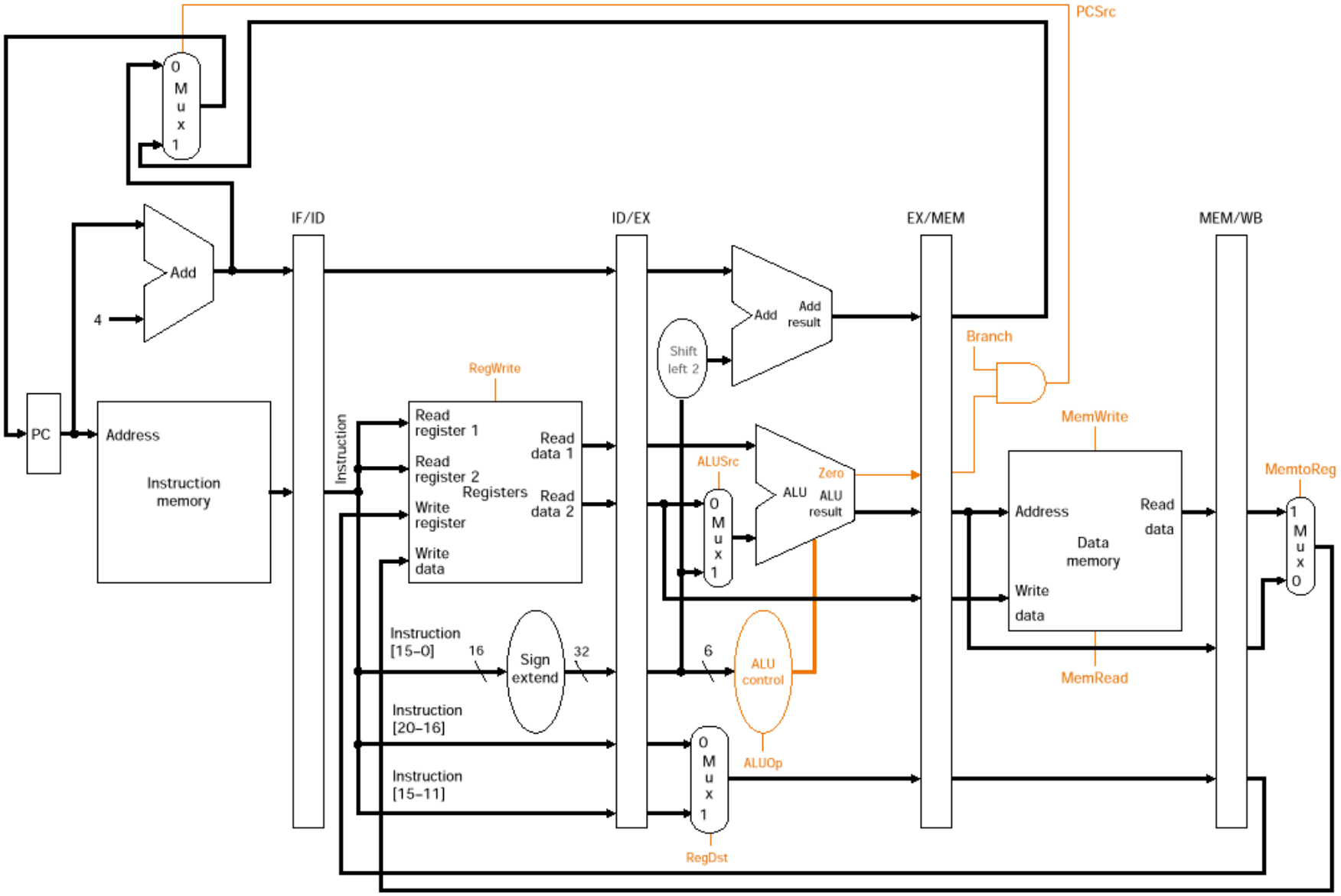


Lecture – 18

Pipelined Datapath with Control Signals



Control Signal for the Pipeline Datapath

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

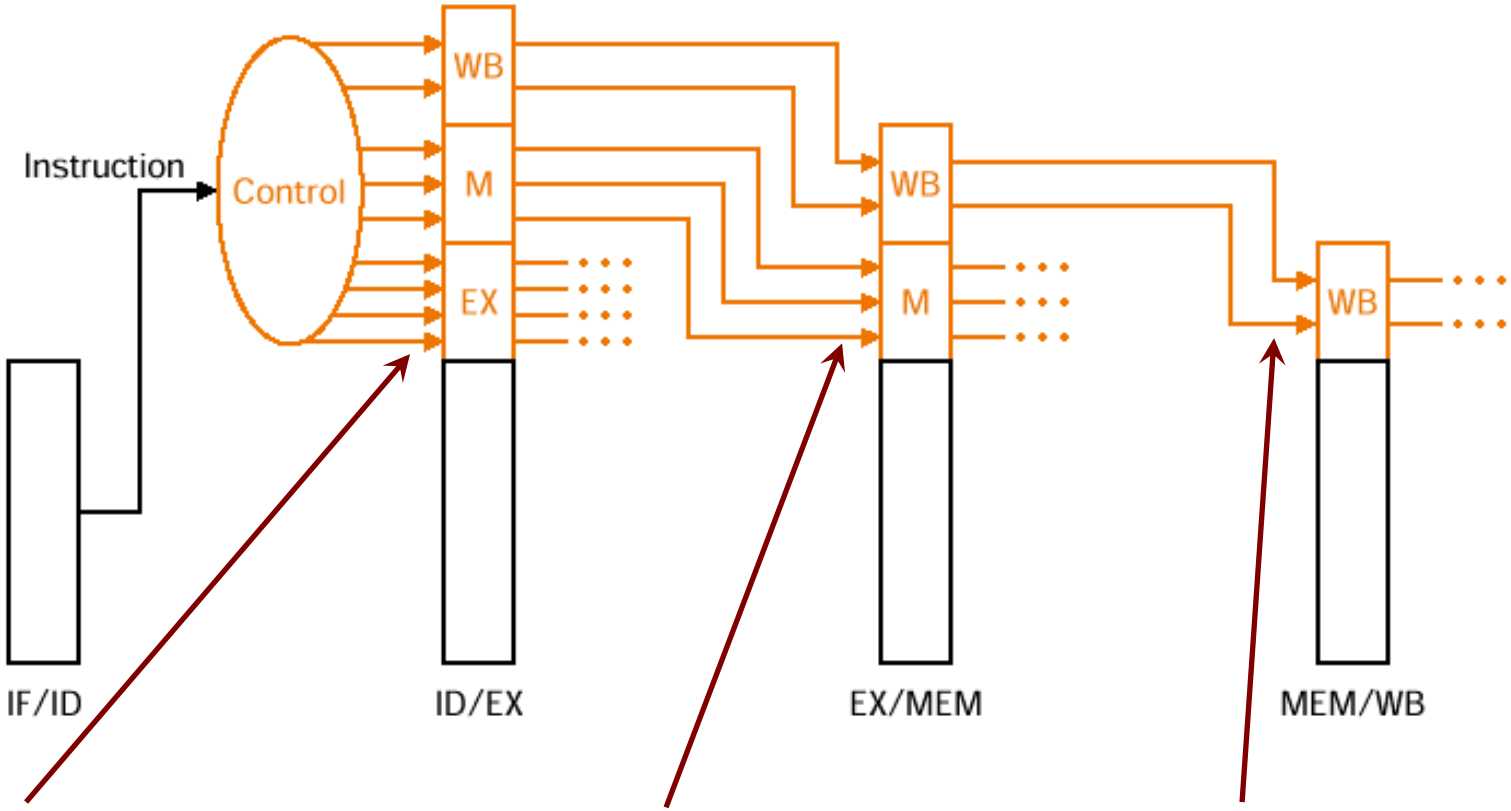
Control Signal for the Pipeline Datapath

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 Signal for the Pipeline Datapath

Instruction	Execution/address calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	Reg Dst	ALU Op1	ALU Op0	ALU Src	Branch	Mem Read	Mem Write	Reg Write	Mem to 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

Control Signals Generation

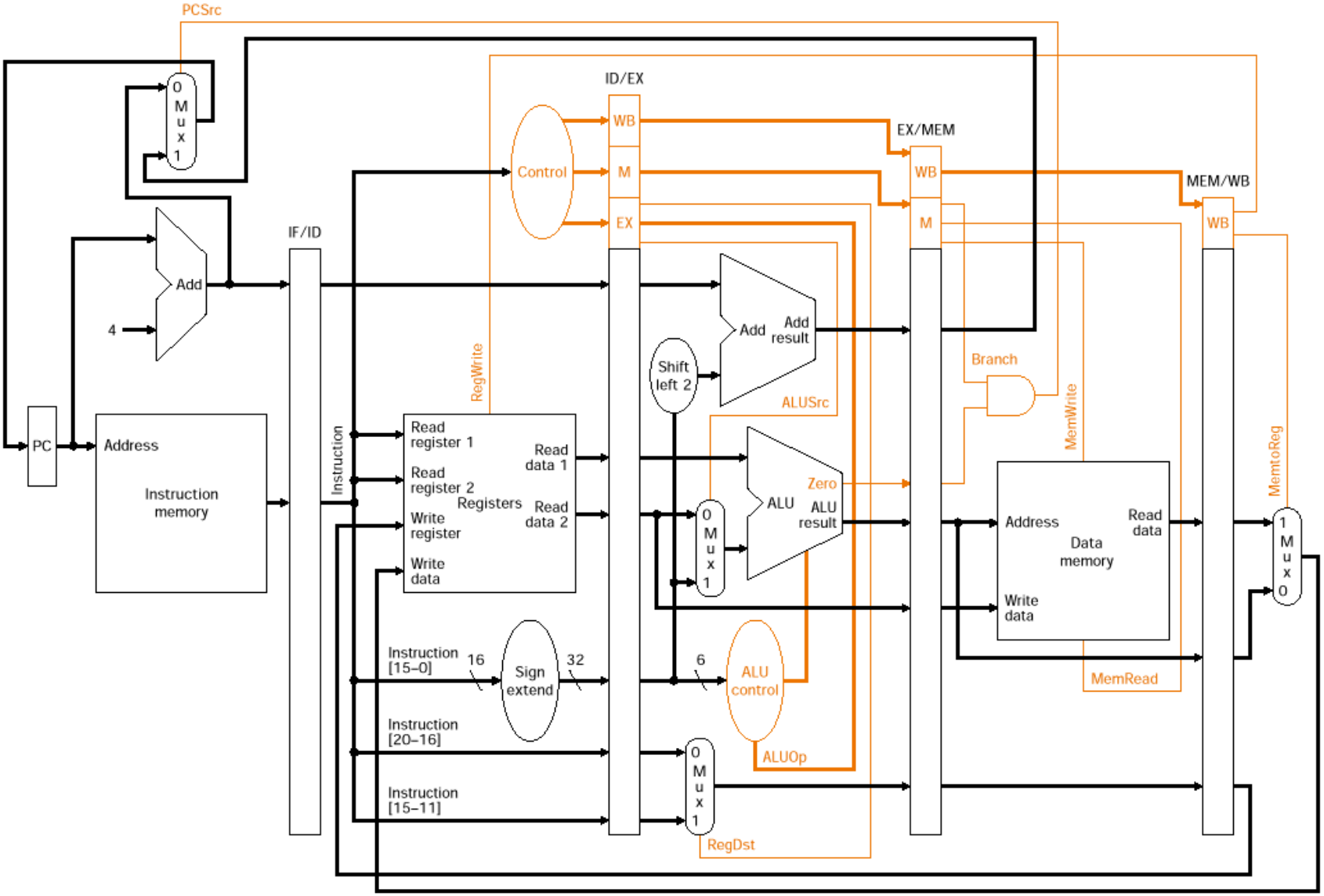


RegDst, ALUOp, ALUSrc

Branch, MemRead, MemWrite

MemtoReg, RegWrite

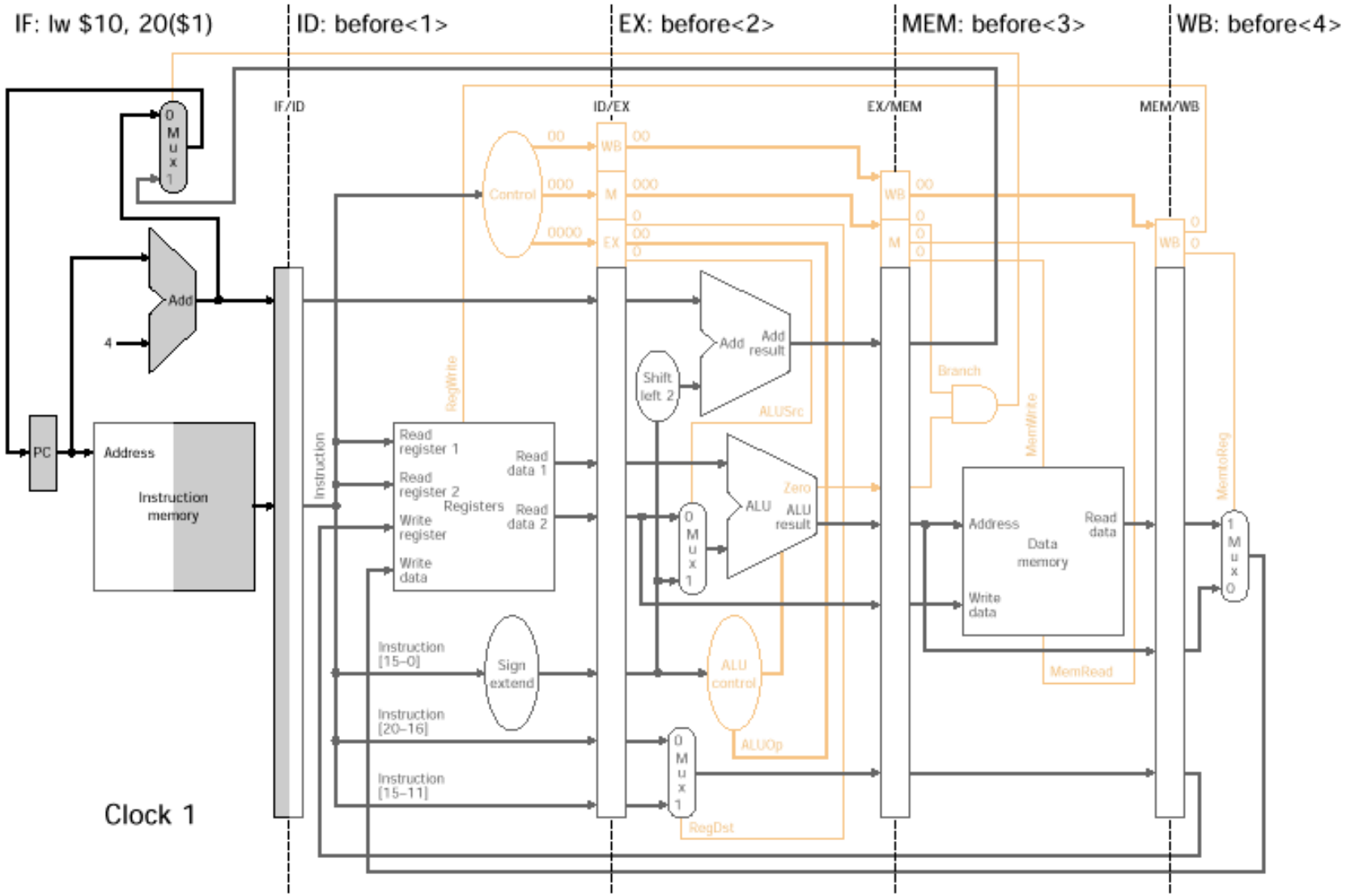
Pipelined Datapath with control Signals Connected



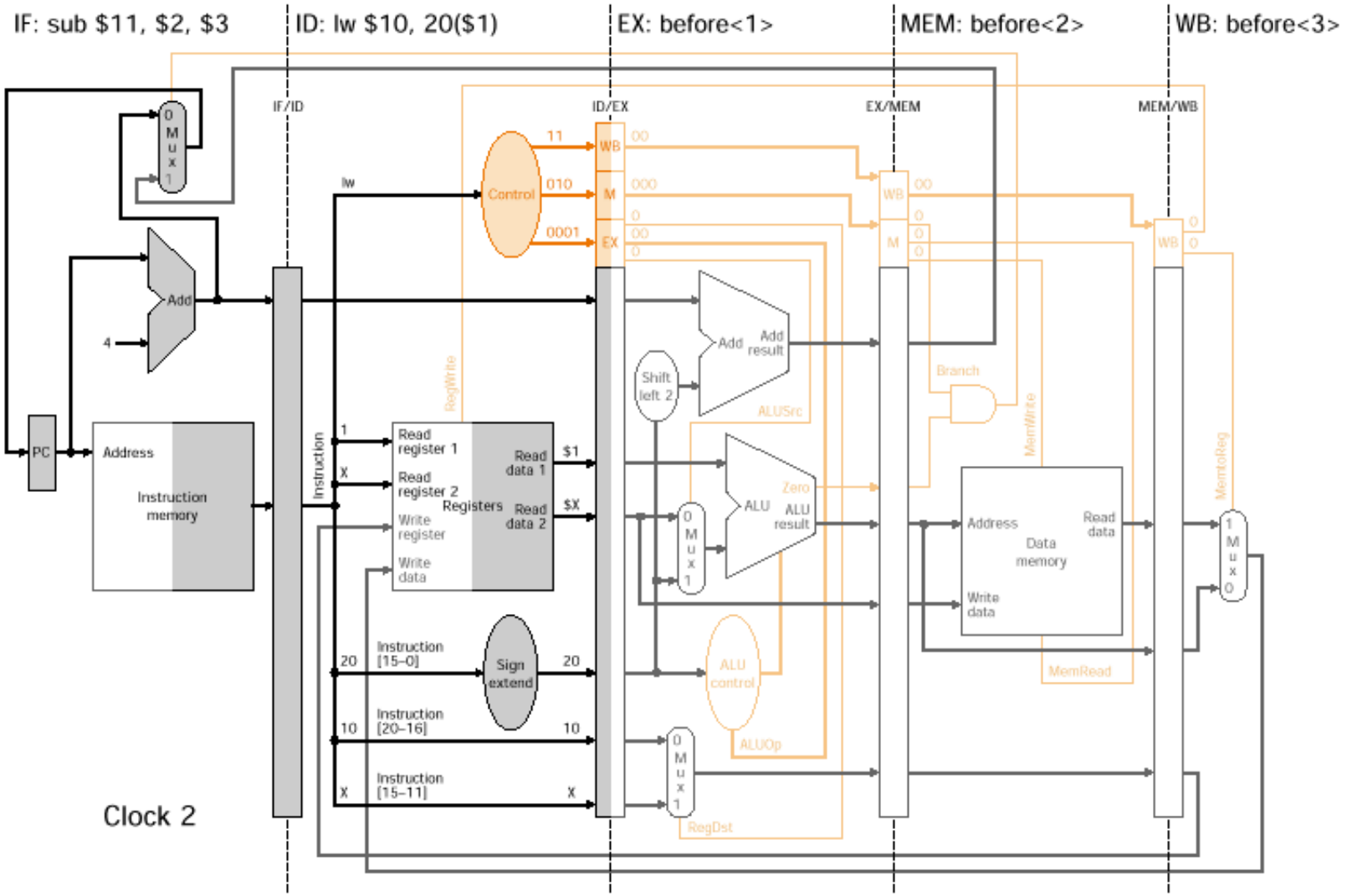
Example

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

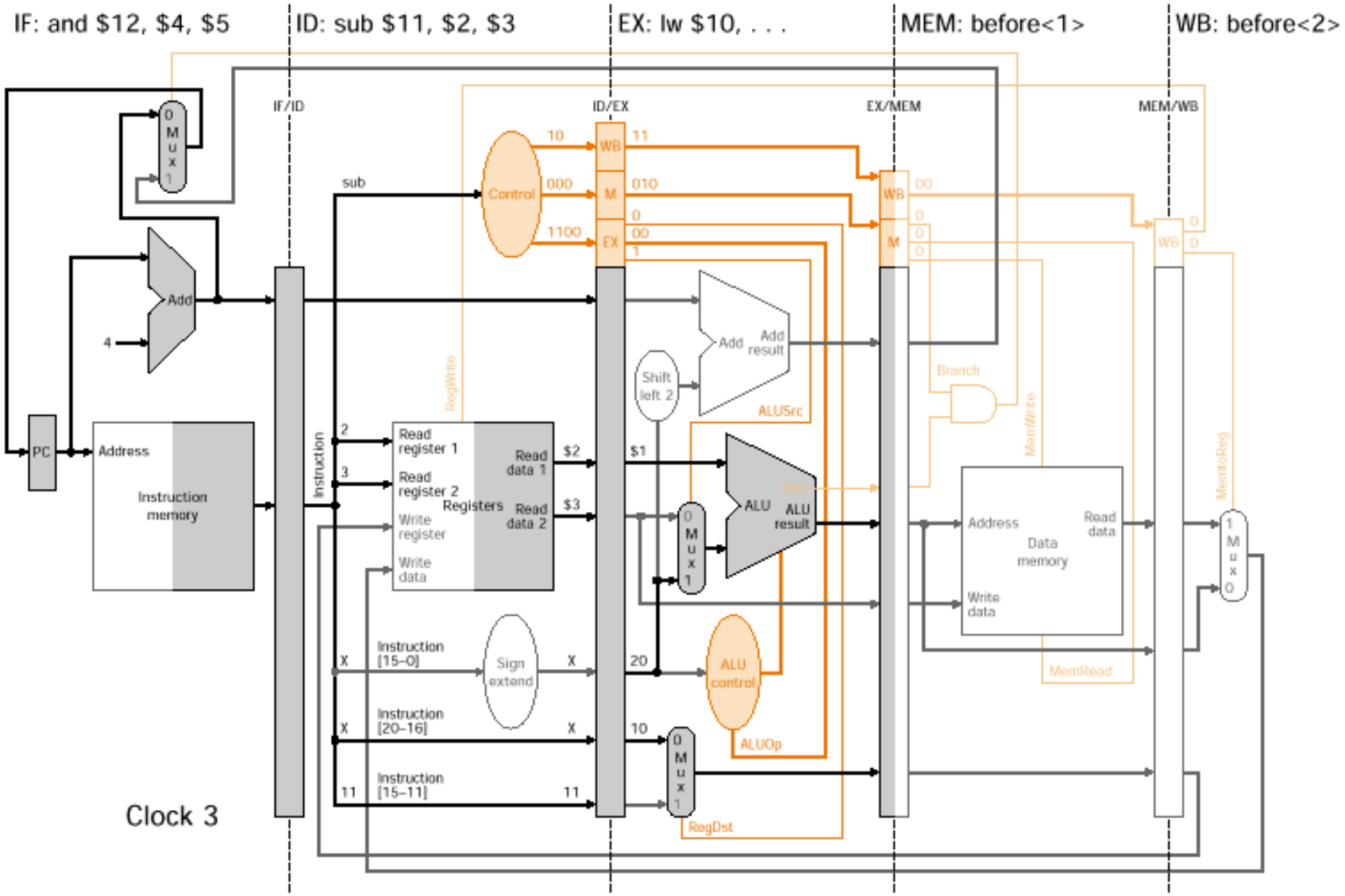
Example


IF: lw $10, 20($1)
ID: before<1>
EX: before<2>
MEM: before<3>
WB: before<4>
IF/ID
ID/EX
EX/MEM
MEM/WB
M u x
Add
4
PC
Address
Instruction memory
Instruction
Control
WB
M
EX
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
RegWrite
Shift left 2
Add
Add result
ALUSrc
Zero
ALU
ALU result
Branch
MemWrite
MemtoReg
Address
Data memory
Read data
Write data
MemRead
Instruction [15–0]
Sign extend
Instruction [20–16]
Instruction [15–11]
ALU control
ALUOp
RegDst
Clock 1

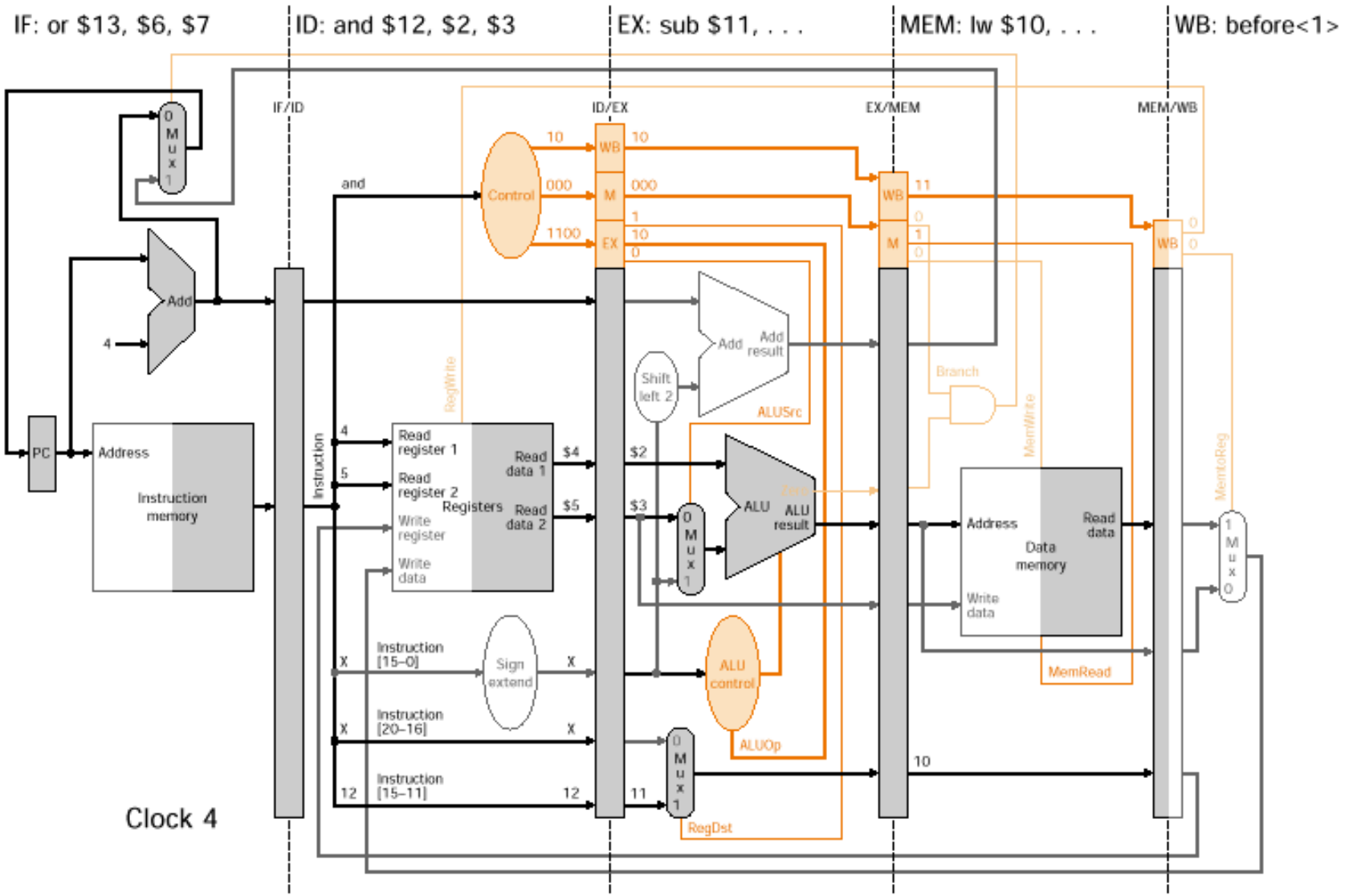
Example


IF: sub $11, $2, $3
ID: lw $10, 20($1)
EX: before<1>
MEM: before<2>
WB: before<3>
IF/ID
ID/EX
EX/MEM
MEM/WB
Clock 2
PC
Address
Instruction memory
Instruction
Add
4
lw
Control
11
010
0001
WB
M
EX
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
$1
$X
RegWrite
Instruction [15–0]
Instruction [20–16]
Instruction [15–11]
Sign extend
20
10
X
Shift left 2
Add result
ALUSrc
ALU
Zero
ALU result
ALU control
ALUOp
RegDst
Branch
MemWrite
MemRead
Data memory
Read data
MemtoReg
Mux

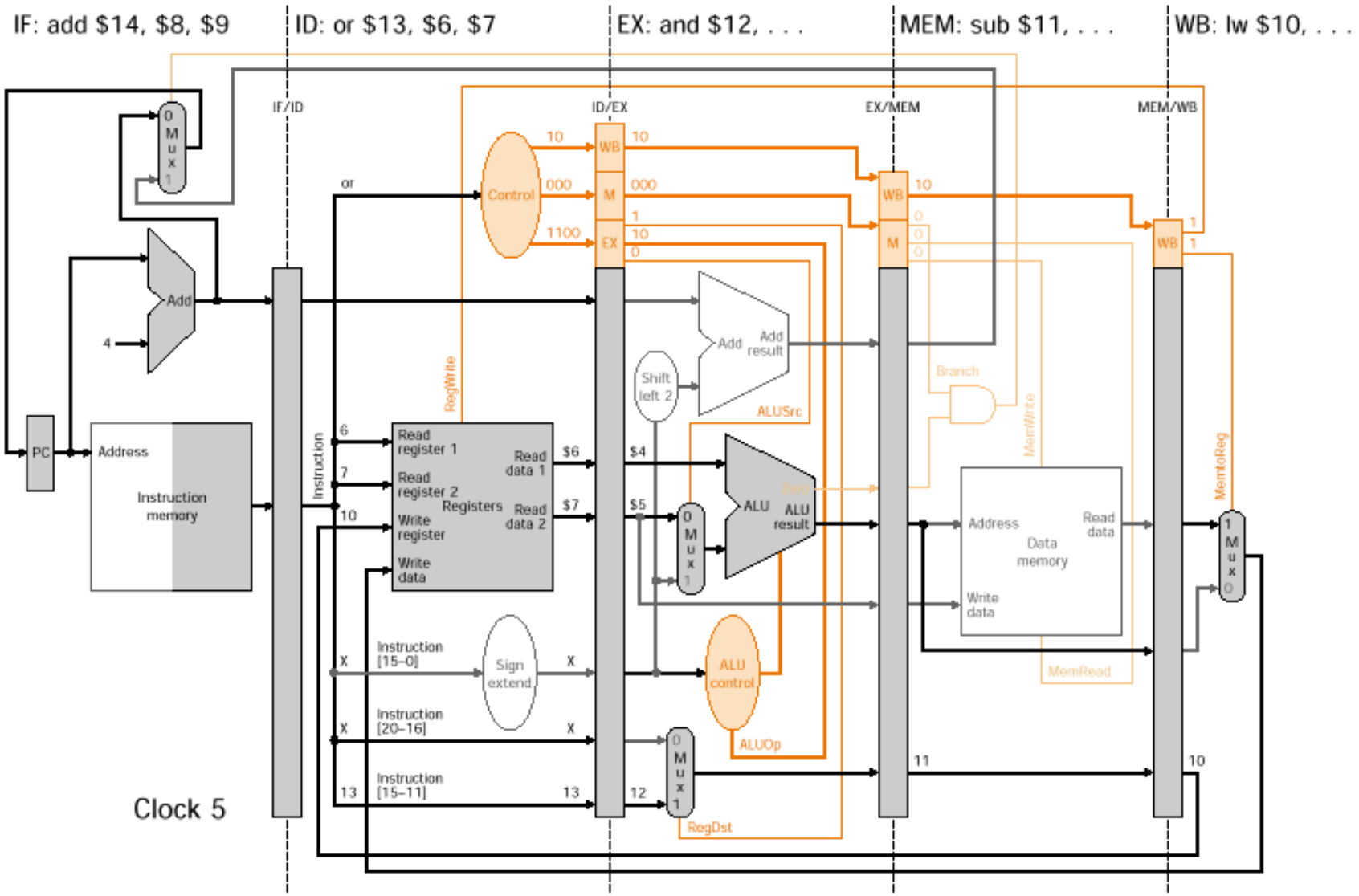
Example


IF: and $12, $4, $5
ID: sub $11, $2, $3
EX: lw $10, . . .
MEM: before<1>
WB: before<2>
IF/ID
ID/EX
EX/MEM
MEM/WB
Clock 3
PC
Address
Instruction memory
Add
4
M u x
sub
Control
10
000
1100
WB
M
EX
11
010
00
1
Instruction
2
3
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
$2
$3
RegWrite
Shift left 2
Add
Add result
ALUSrc
$1
ALU
ALU result
Zero
Branch
MemWrite
MemtoReg
Address
Data memory
Read data
Write data
MemRead
Instruction [15–0]
Instruction [20–16]
Instruction [15–11]
Sign extend
X
20
10
11
ALU control
ALUOp
RegDst

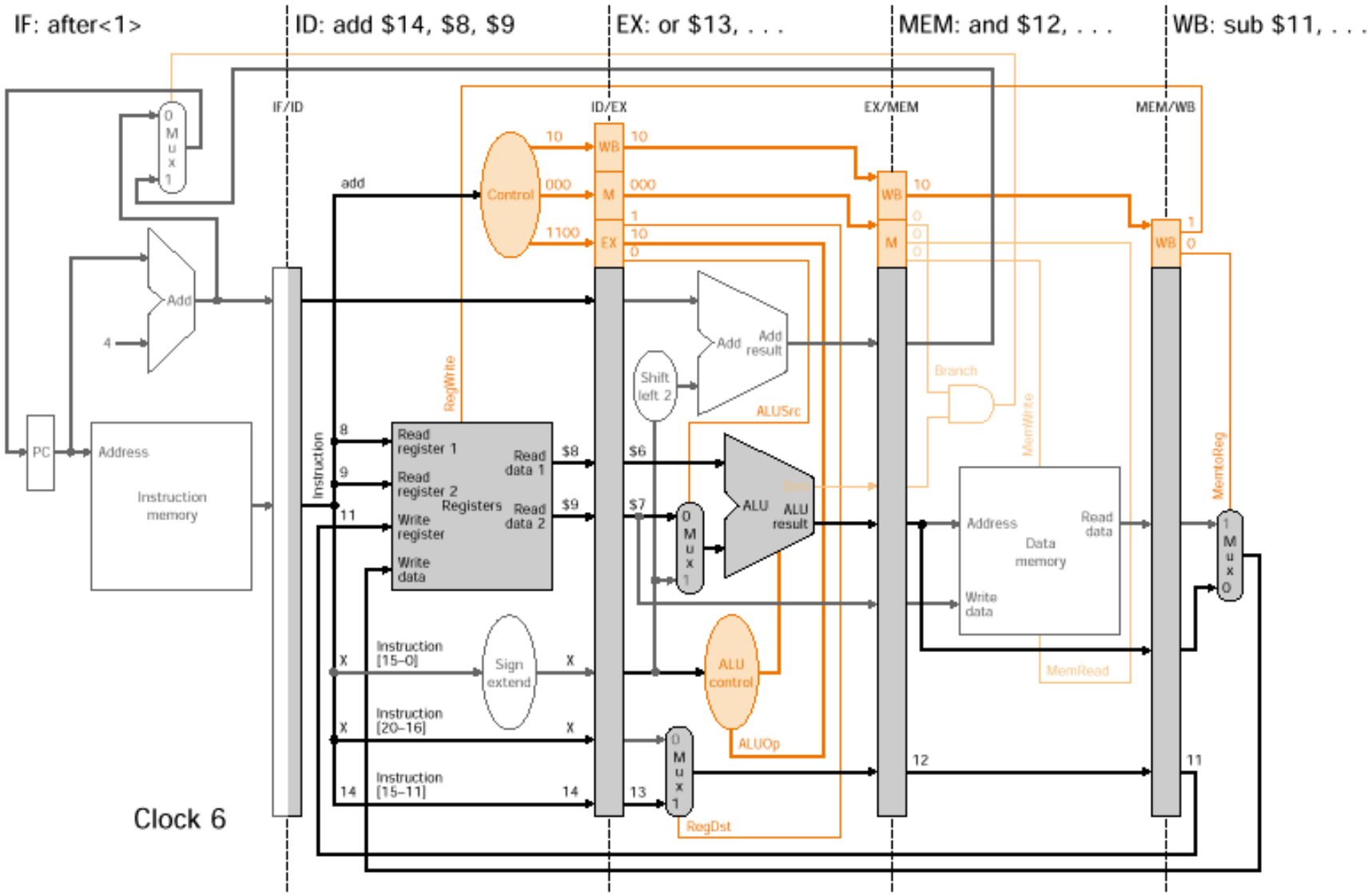
Example


IF: or $13, $6, $7
ID: and $12, $2, $3
EX: sub $11, . . .
MEM: lw $10, . . .
WB: before<1>
IF/ID
ID/EX
EX/MEM
MEM/WB
and
Control
10
000
1100
WB
M
EX
Add
4
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
RegWrite
Shift left 2
Add result
ALUSrc
ALU
ALU result
Zero
Branch
MemWrite
MemtoReg
Data memory
Read data
MemRead
ALU control
ALUOp
Sign extend
Instruction [15–0]
Instruction [20–16]
Instruction [15–11]
RegDst
Clock 4

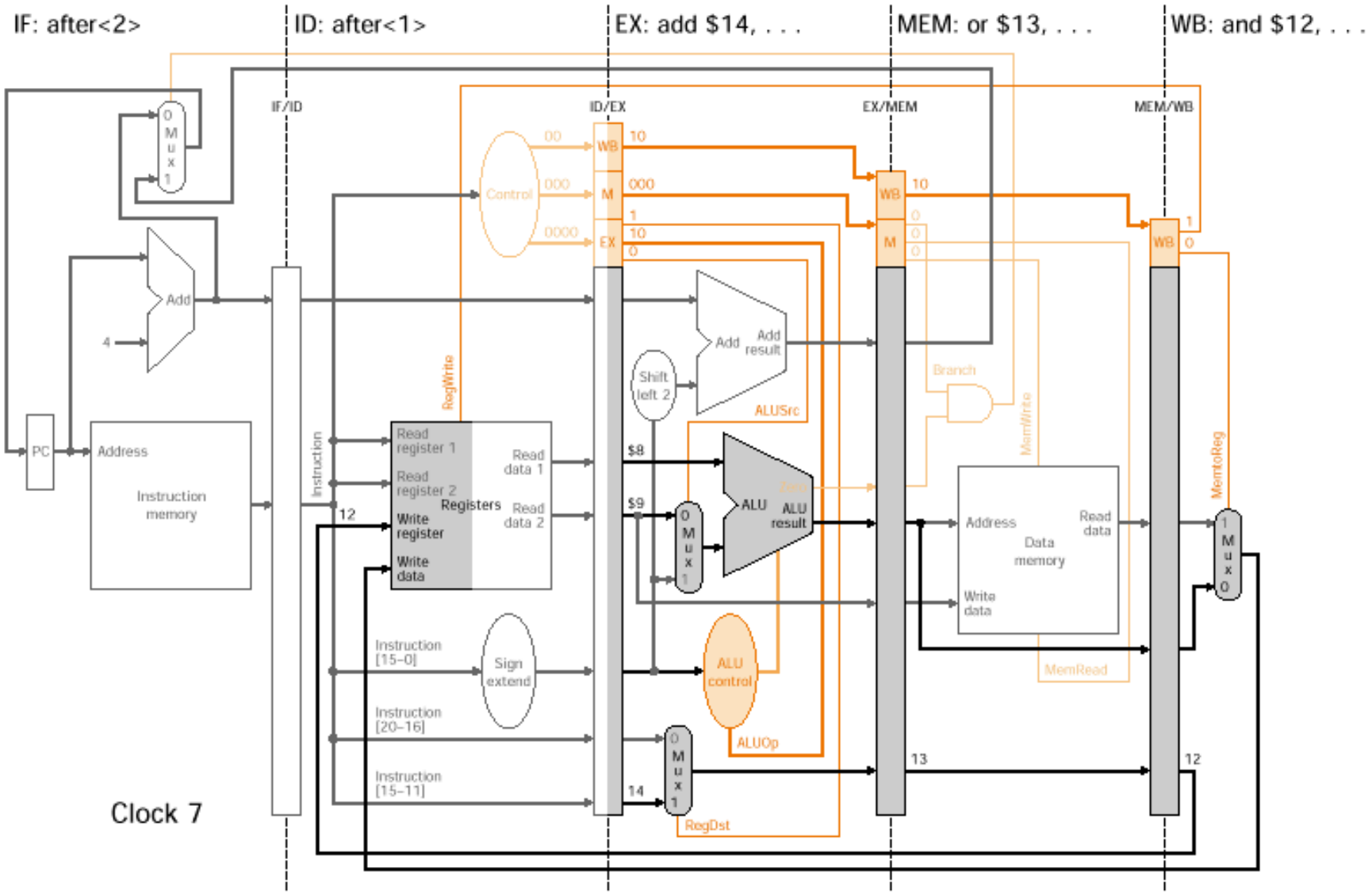
Example


IF: add $14, $8, $9
ID: or $13, $6, $7
EX: and $12, . . .
MEM: sub $11, . . .
WB: lw $10, . . .
IF/ID
ID/EX
EX/MEM
MEM/WB
Clock 5
PC
Address
Instruction memory
Instruction
Add
4
M u x
Control
WB
M
EX
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
RegWrite
Shift left 2
Add result
ALUSrc
ALU
ALU result
Zero
Branch
MemWrite
MemtoReg
Address
Data memory
Read data
Write data
MemRead
Instruction [15–0]
Instruction [20–16]
Instruction [15–11]
Sign extend
ALU control
ALUOp
RegDst

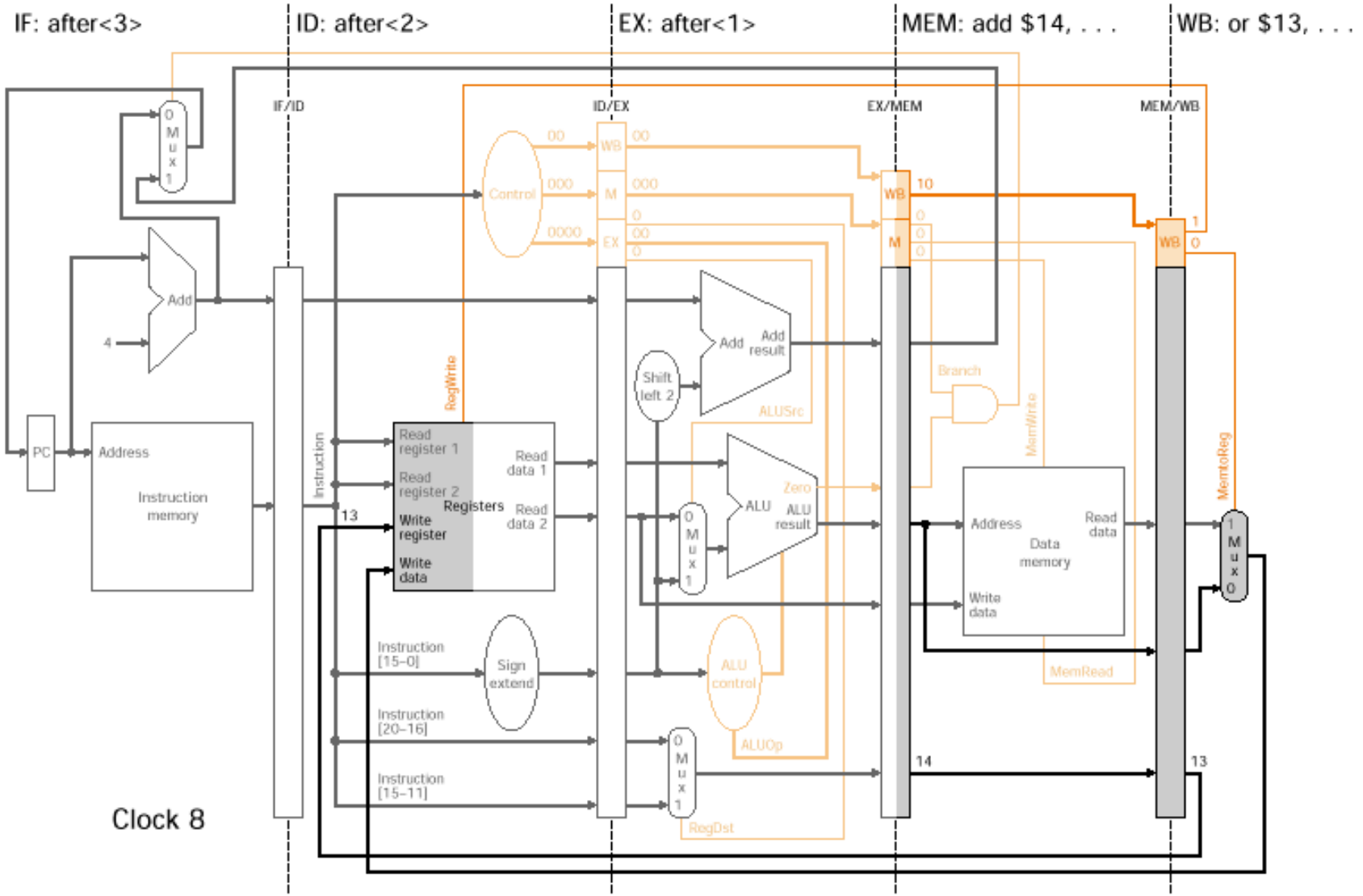
Example


IF: after<1>
ID: add $14, $8, $9
EX: or $13, . . .
MEM: and $12, . . .
WB: sub $11, . . .
IF/ID
ID/EX
EX/MEM
MEM/WB
Clock 6
PC
Address
Instruction memory
Instruction
Add
4
add
Control
10
000
1100
WB
M
EX
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
RegWrite
$8
$9
$6
$7
Shift left 2
Add result
ALUSrc
ALU
ALU result
Zero
Branch
MemWrite
MemRead
MemtoReg
Address
Data memory
Read data
Write data
ALU control
ALUOp
RegDst
Instruction [15–0]
Instruction [20–16]
Instruction [15–11]
Sign extend
X
8
9
11
14
13
12
M u x

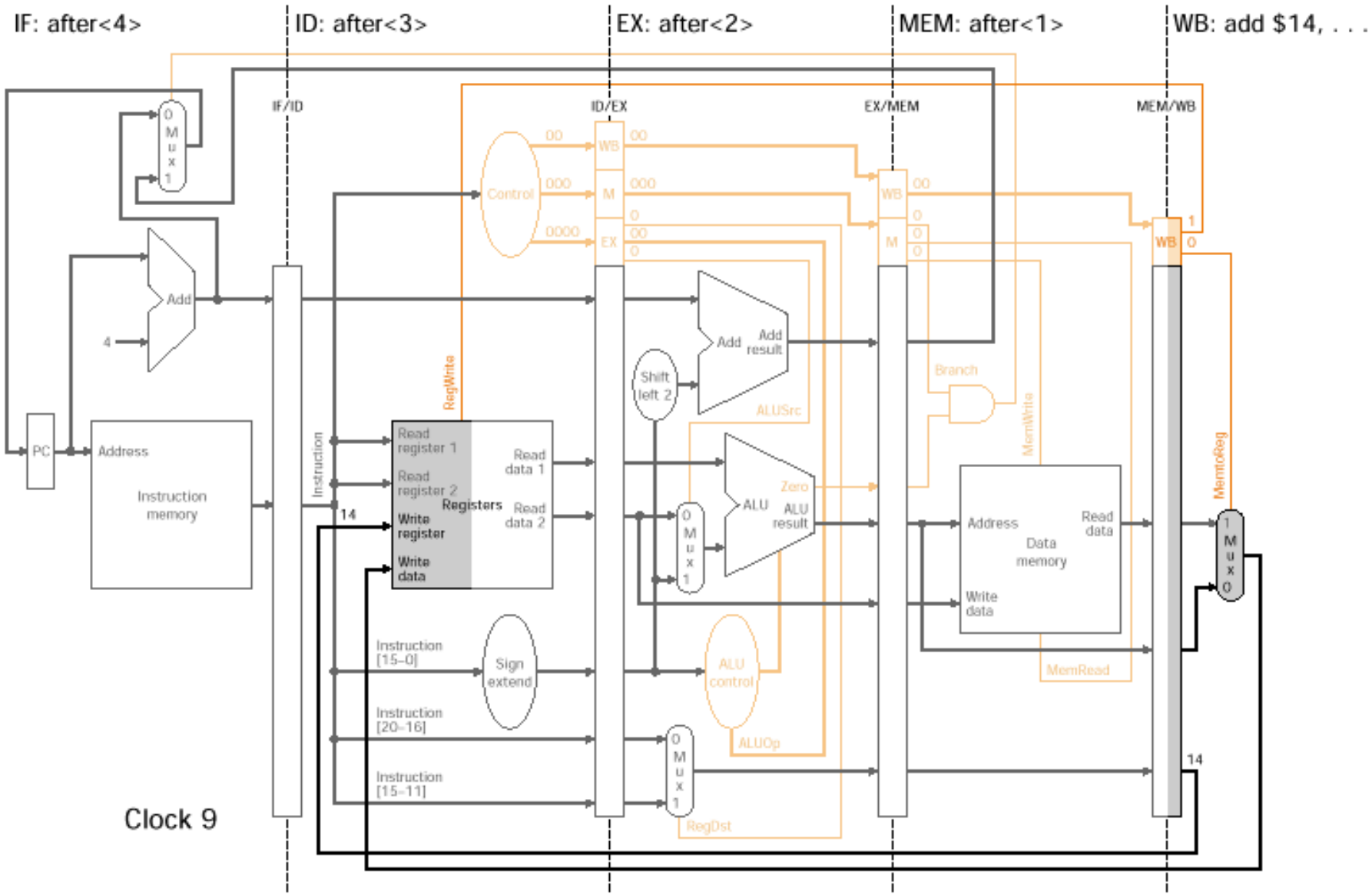
Example


IF: after<2>
ID: after<1>
EX: add $14, . . .
MEM: or $13, . . .
WB: and $12, . . .
IF/ID
ID/EX
EX/MEM
MEM/WB
M u x
Add
4
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
12
Control
00
000
0000
WB
M
EX
10
000
1
10
0
Add
Add result
Shift left 2
ALUSrc
$8
$9
ALU
ALU result
Zero
Branch
MemWrite
Address
Data memory
Read data
Write data
MemRead
MemtoReg
Instruction [15–0]
Sign extend
Instruction [20–16]
Instruction [15–11]
ALU control
ALUOp
14
RegDst
13
12
Clock 7

Example


IF: after<3>
ID: after<2>
EX: after<1>
MEM: add $14, . . .
WB: or $13, . . .
IF/ID
ID/EX
EX/MEM
MEM/WB
Clock 8
PC
Address
Instruction memory
Add
4
M u x
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
RegWrite
13
Control
WB
M
EX
00
000
0000
Shift left 2
Add result
ALUSrc
ALU
Zero
ALU result
ALU control
ALUOp
RegDst
Instruction [15–0]
Instruction [20–16]
Instruction [15–11]
Sign extend
Branch
MemWrite
MemRead
Data memory
Address
Read data
Write data
MemtoReg
10
14

Example


IF: after<4>
ID: after<3>
EX: after<2>
MEM: after<1>
WB: add $14, . . .
IF/ID
ID/EX
EX/MEM
MEM/WB
M u x
0
1
Add
4
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
14
Control
00
000
0000
WB
M
EX
Shift left 2
Add
Add result
ALUSrc
Zero
ALU
ALU result
Branch
MemWrite
MemtoReg
Address
Data memory
Read data
Write data
MemRead
Instruction [15–0]
Sign extend
Instruction [20–16]
Instruction [15–11]
ALU control
ALUOp
RegDst
Clock 9

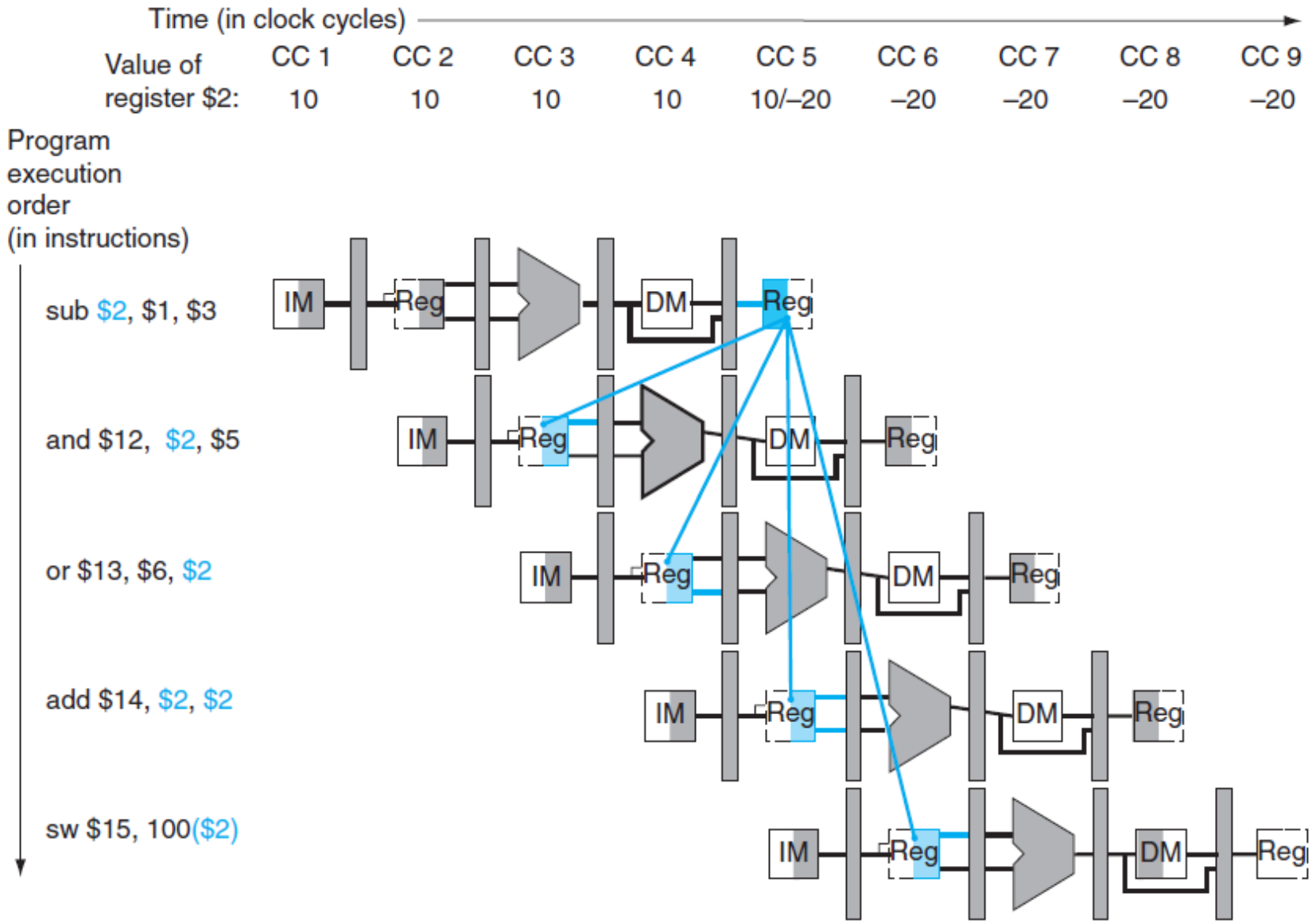
Data Hazards

- Consider the following code segments:

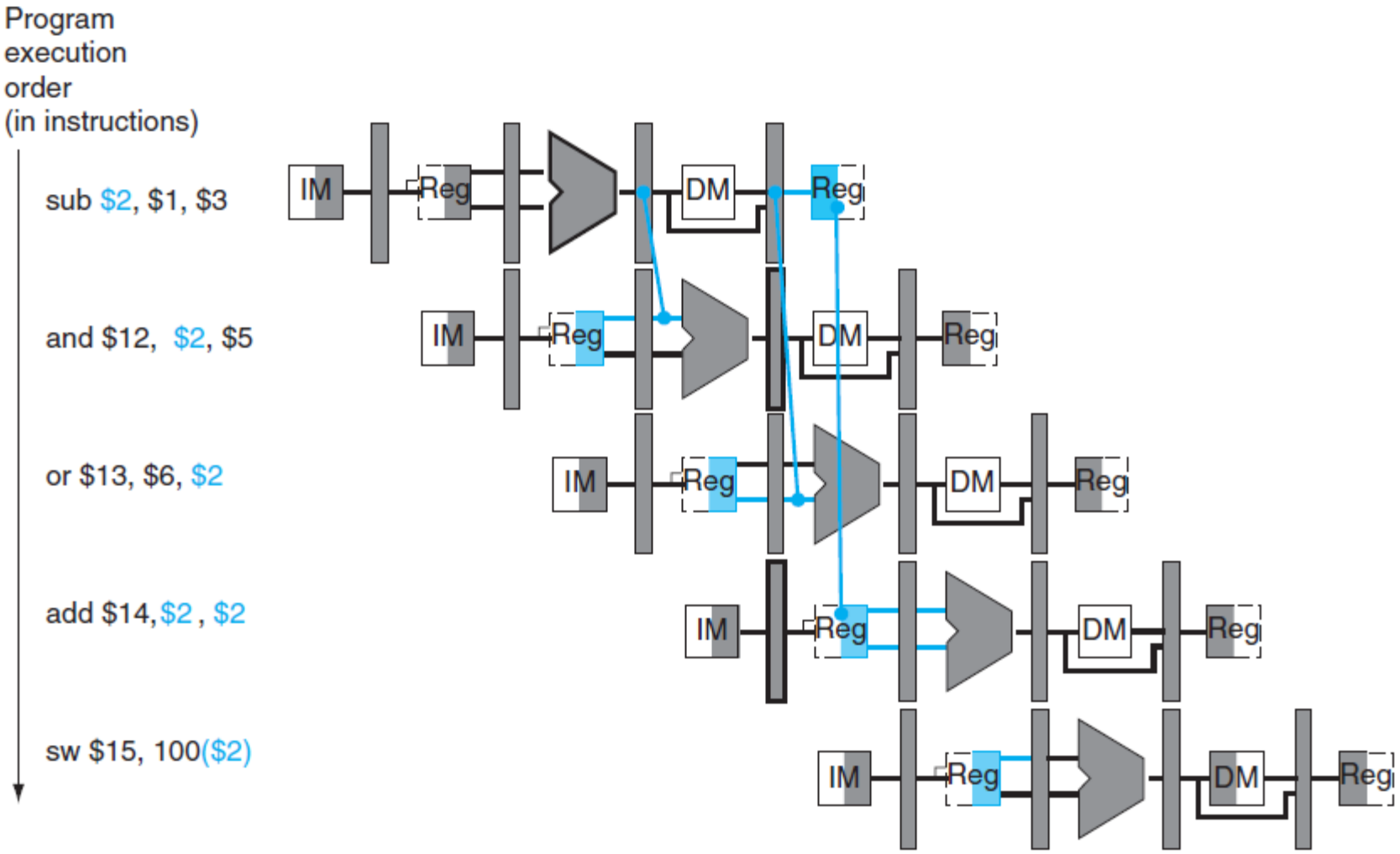
```
sub   $2, $1, $3       //$2 written by sub
and $12, $2, $5        //1st operand ($2) depends on sub
or    $13, $6, $2      //2nd operand ($2) depends on sub
add  $14, $2, $2       //1st ($2) and 2nd ($2) operand depend on sub
sw    $15, 100($2)     //base ($2) depends on sub
```

- One hazard resolved by the design of register file. Write in the first half of the clock cycle and the read in the second half. So read delivers what is written.

Pipelined Dependences



Fixing Data Hazard by Forwarding



Data Forwarding

- Data can be forwarded from EX/MEM register and MEM/WB registers.
- Forward by taking the inputs to the ALU from any pipeline register rather than just ID/EX by
  - adding multiplexors to the inputs of the ALU so can pass Rd back to either (or both) of the EX's stage Rs and Rt ALU inputs
    00: normal input (ID/EX pipeline registers)
    10: forward from previous instr (EX/MEM pipeline registers)
    01: forward from instr 2 back (MEM/WB pipeline registers)
  - adding the proper control hardware
- With forwarding can run at full speed even in the presence of data dependencies

Two Pairs of Hazard Conditions

EX/MEM Hazard:

1a.if (EX/MEM.RegisterRd ==ID/EX.RegisterRs)

forwardA=10 [Between sub and and]

1b. If (EX/MEM.RegisterRd == ID/EX.RegisterRt)

forwardB=10

MEM/WB Hazard:

2a. If (MEM/WB.RegisterRd == ID/EX.RegisterRs)

forwardA=01

2b. If (MEM/WB.RegisterRd == ID/EX.RegisterRt)

forwardB=01 [Between sub and or]

Hazard Conditions

1. **EX/MEM hazard:**

if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd == ID/EX.RegisterRs))
ForwardA = 10
if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd == ID/EX.RegisterRt))
ForwardB = 10

Forwards the result from the previous instr. to either input of the ALU **provided it writes**.

2. **MEM/WB hazard:**

if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd == ID/EX.RegisterRs))
ForwardA = 01
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd == ID/EX.RegisterRt))
ForwardB = 01

Forwards the result from the second previous instr. to either input of the ALU **provided it writes**.

Hazard Conditions

1. **EX/MEM hazard:**

if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd != 0)
and (EX/MEM.RegisterRd == ID/EX.RegisterRs))
ForwardA = 10
if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd != 0)
and (EX/MEM.RegisterRd == ID/EX.RegisterRt))
ForwardB = 10

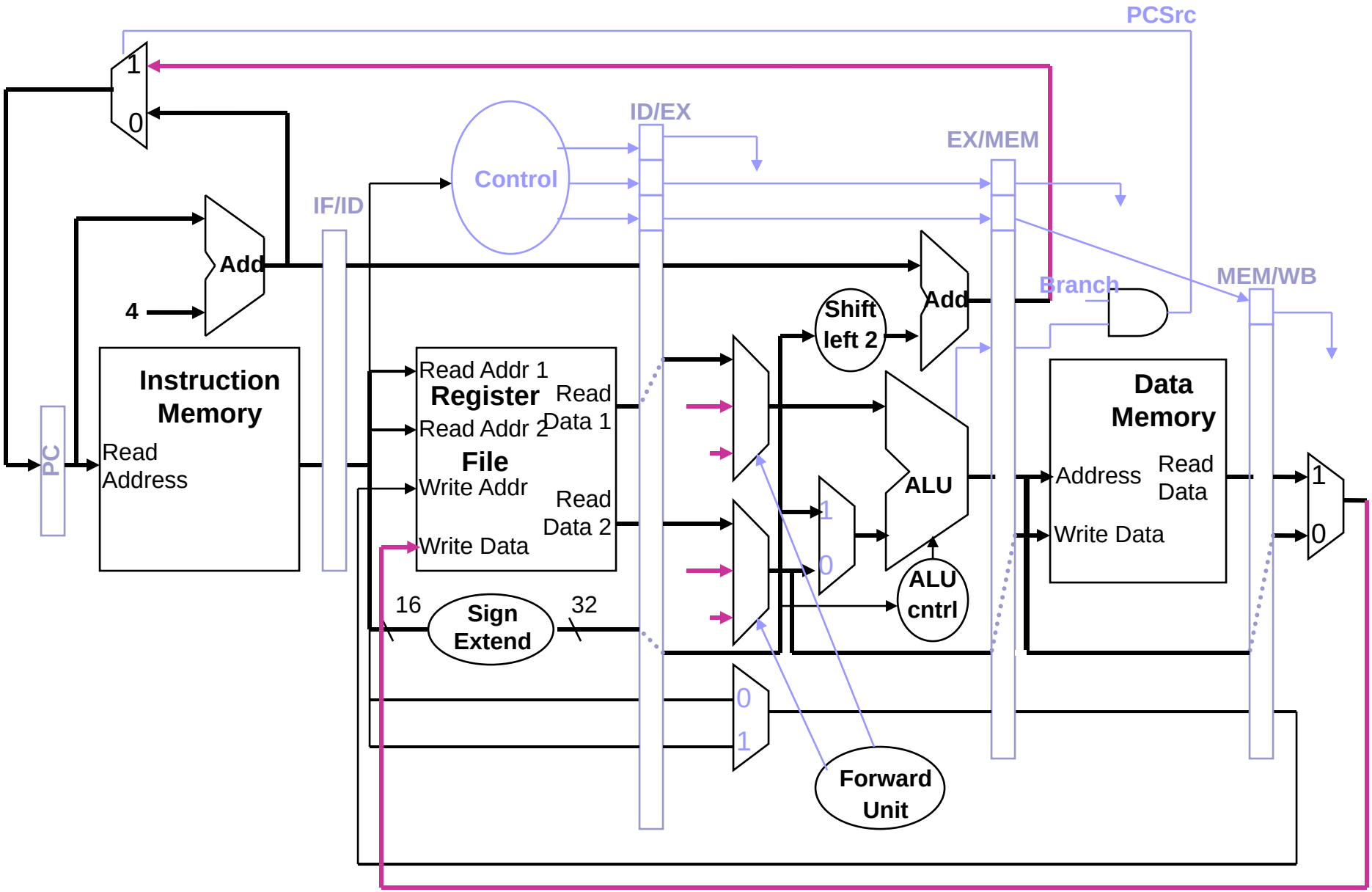
Forwards the result from the previous instr. to either input of the ALU provided it writes **and != 0**.

2. **MEM/WB hazard:**

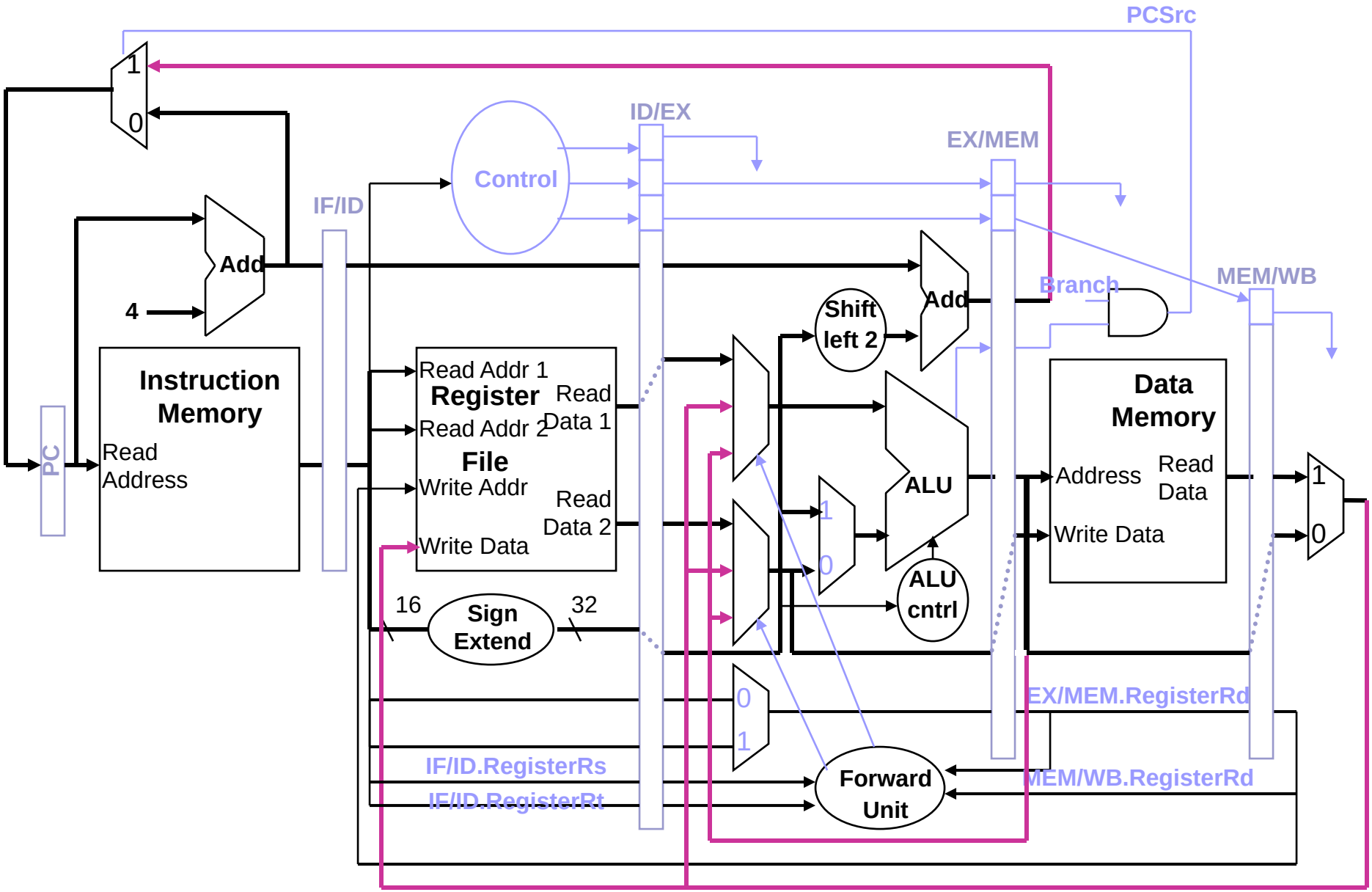
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (MEM/WB.RegisterRd == ID/EX.RegisterRs))
ForwardA = 01
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (MEM/WB.RegisterRd == ID/EX.RegisterRt))
ForwardB = 01

Forwards the result from the second previous instr. to either input of the ALU provided it writes **and != 0**

Datapath with Forwarding Hardware

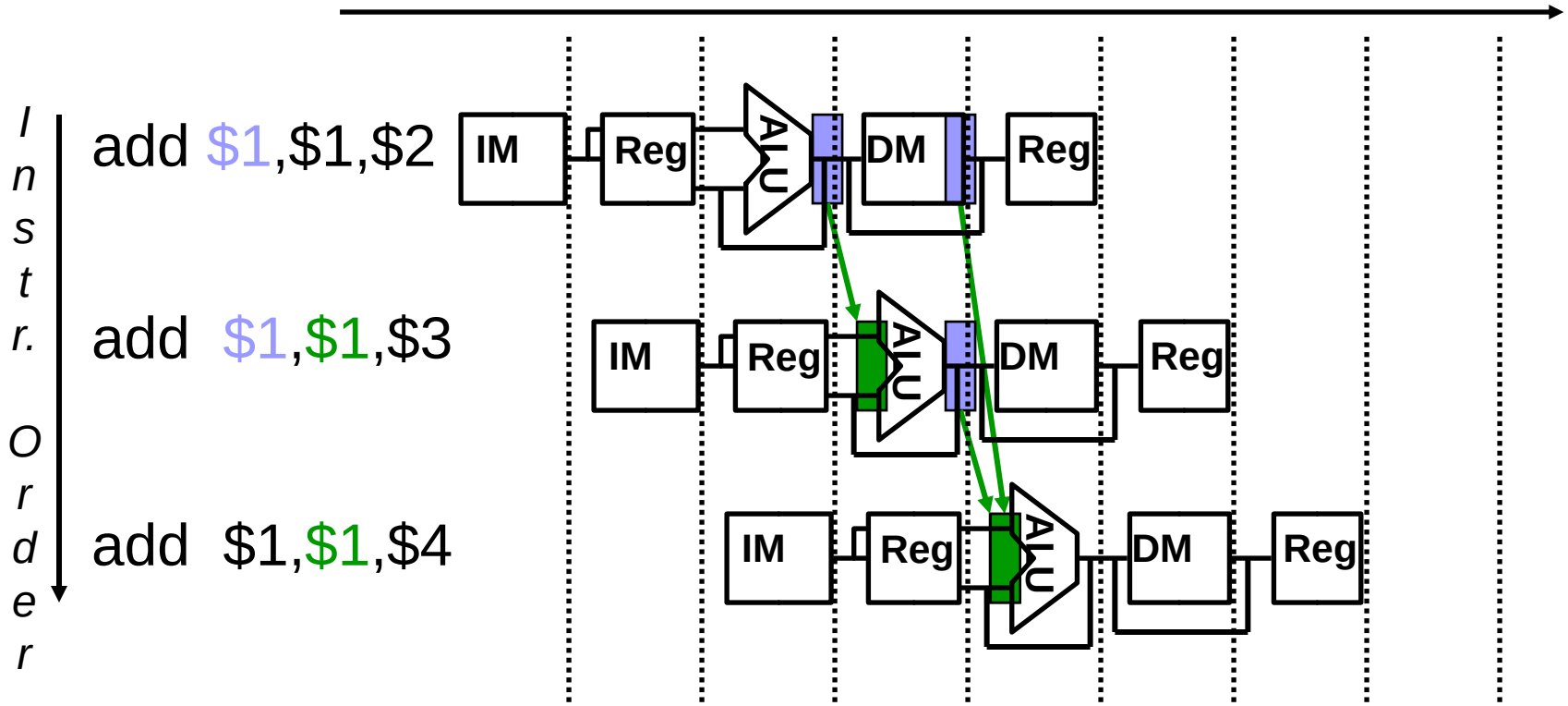


Datapath with Forwarding Hardware



Yet Another Complication!

- Another potential data hazard can occur when there is a conflict between the result of the WB stage instruction and the MEM stage instruction – which should be forwarded? More recent result!



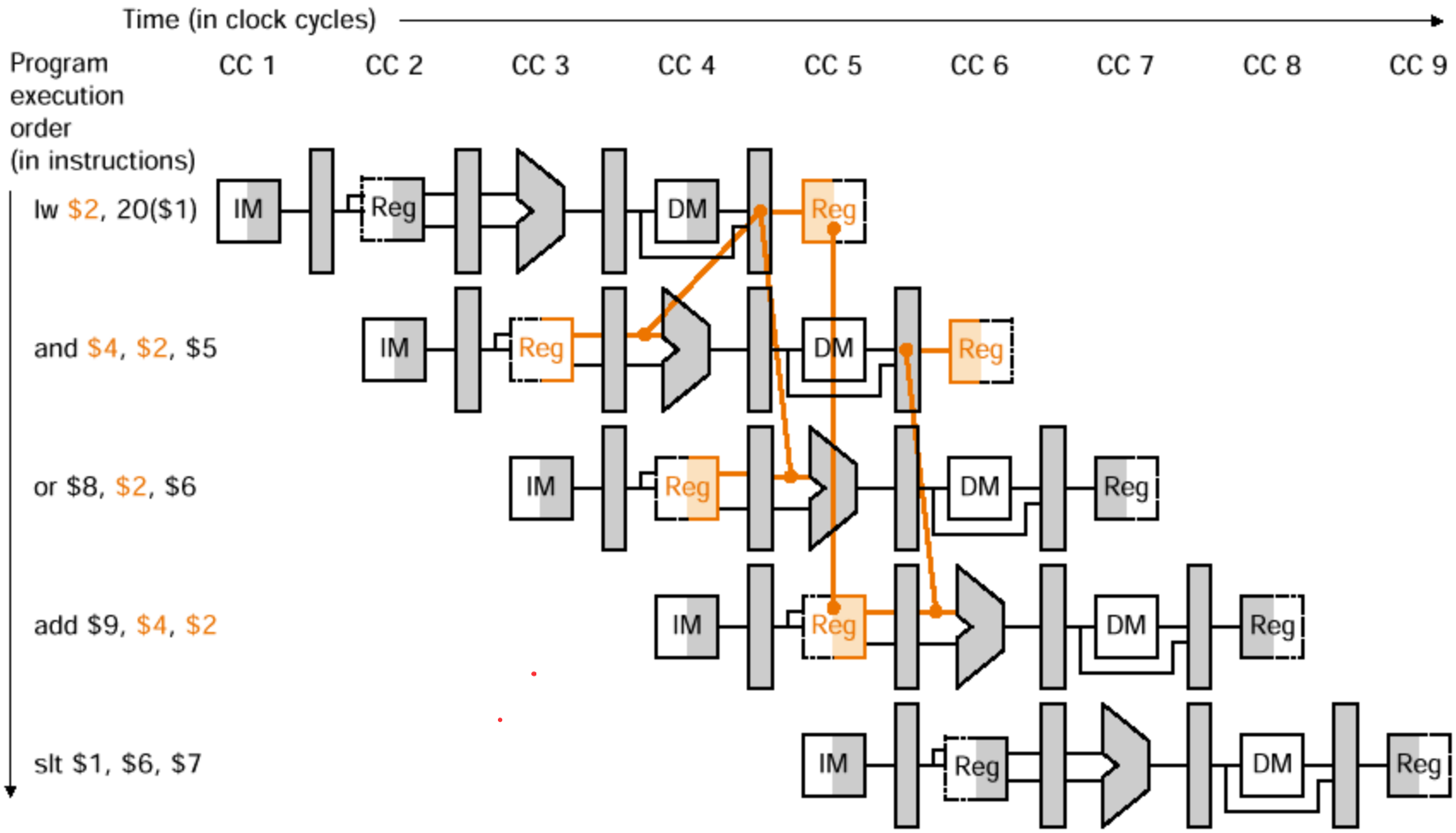
Hazard Conditions

2. **MEM/WB hazard:**

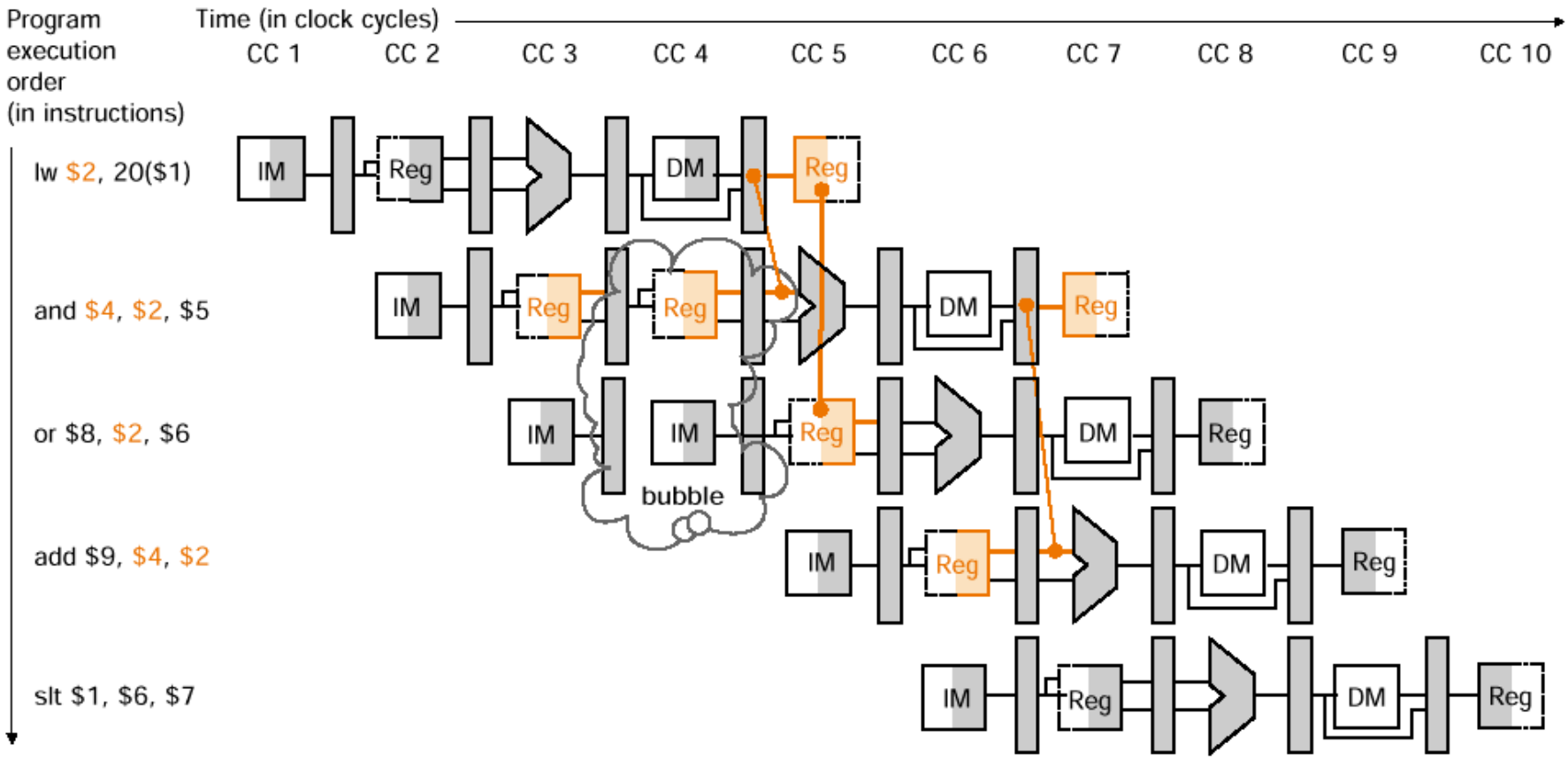
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (EX/MEM.RegisterRd != ID/EX.RegisterRs)
and(MEM/WB.RegisterRd == ID/EX.RegisterRs))
ForwardA = 01

if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (EX/MEM.RegisterRd != ID/EX.RegisterRt)
and (MEM/WB.RegisterRd == ID/EX.RegisterRt))
ForwardB = 01

Data Hazard Requiring a Stall



Data Hazard Requiring a Stall



Hazard Detection Unit

- We need hazard detection unit beside the forwarding unit.
- It operates during the ID stage so that it can insert the stall between the load and it's use.
- Condition

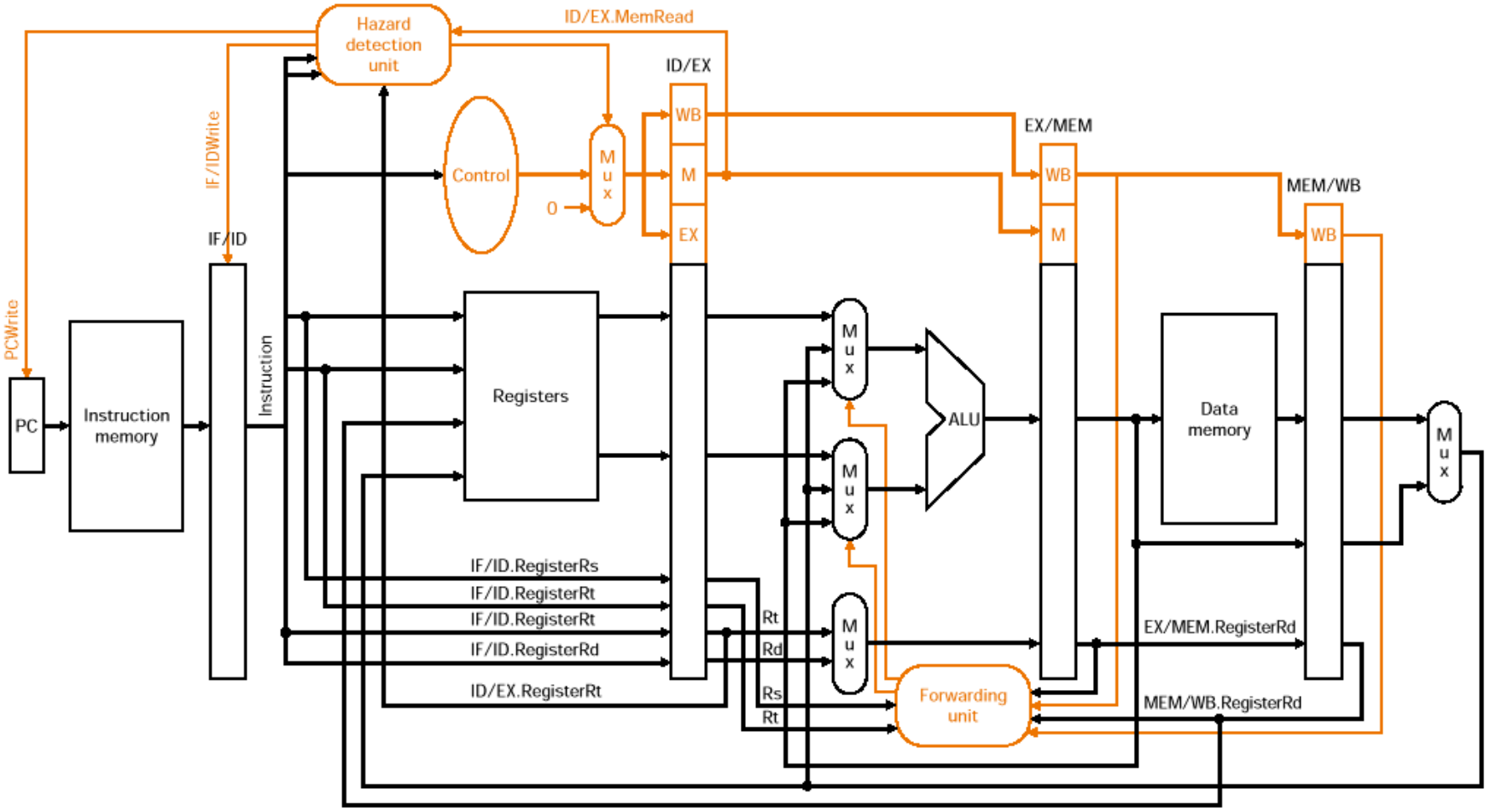
  if (ID/EX.MemRead and

  ((ID/EX.RegisterRt = IF/ID.RegisterRs) or

  (ID/EX.RegisterRt = IF/ID.RegisterRt)))

  stall the pipeline

Stall Logic


Hazard detection unit
ID/EX.MemRead
ID/EX
WB
M
EX
EX/MEM
WB
M
MEM/WB
WB
Control
M u x
0
IF/IDWrite
IF/ID
PCWrite
PC
Instruction memory
Instruction
Registers
M u x
M u x
M u x
ALU
Data memory
M u x
IF/ID.RegisterRs
IF/ID.RegisterRt
IF/ID.RegisterRt
IF/ID.RegisterRd
ID/EX.RegisterRt
Rt
Rd
Rs
Rt
Forwarding unit
EX/MEM.RegisterRd
MEM/WB.RegisterRd

Nops

- Both instructions in ID and IF stage must be stalled.
- For this reason, the PC register and the IF/ID register are preserved.
- The EX stage must do **nops**.
- To insert **nops** we must deassert all nine control signal in EX, MEM and WB stage.