Enhancing Performance with Pipelining

Chapter Four

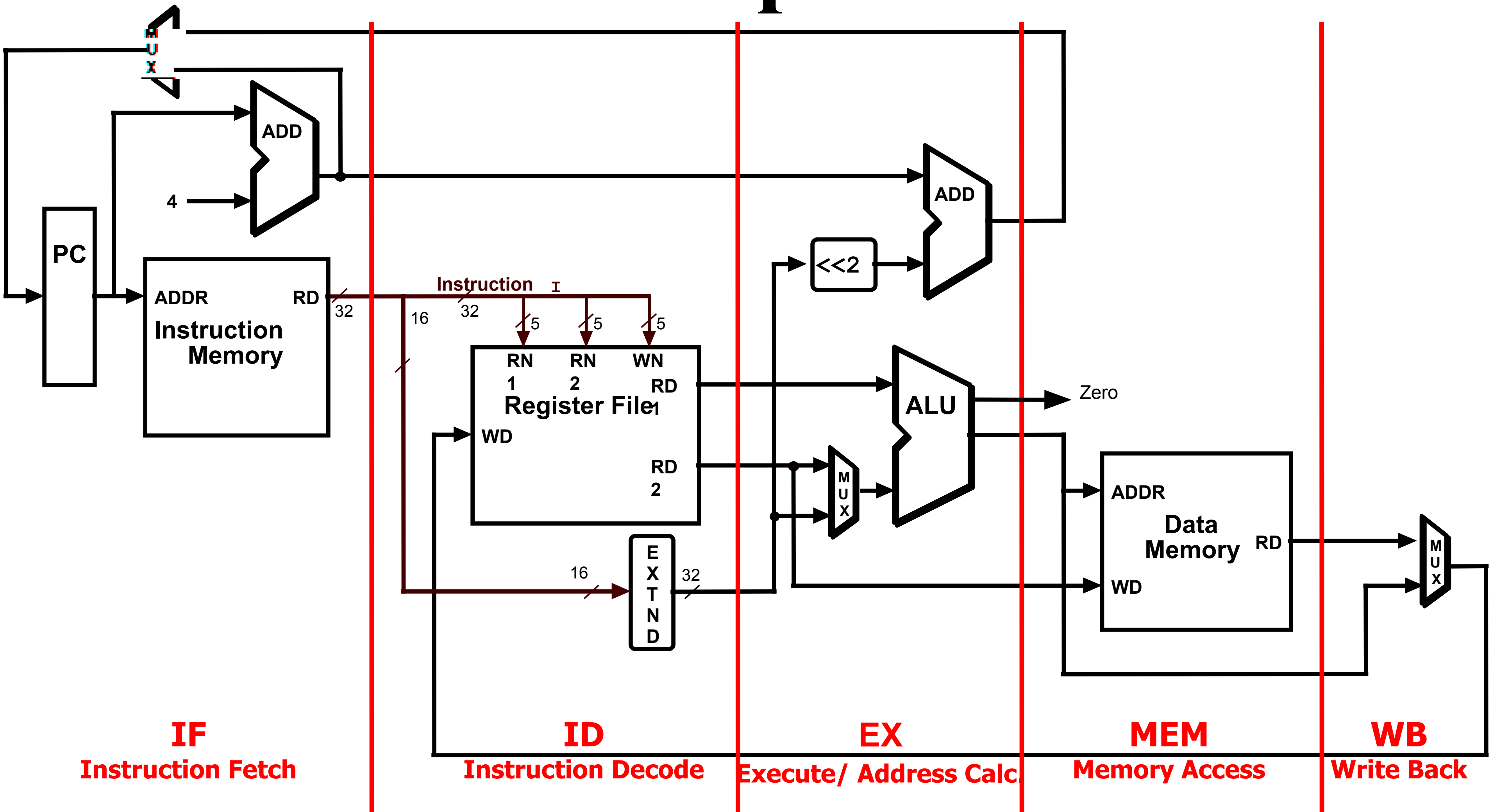
of

The Book of David A. Patterson

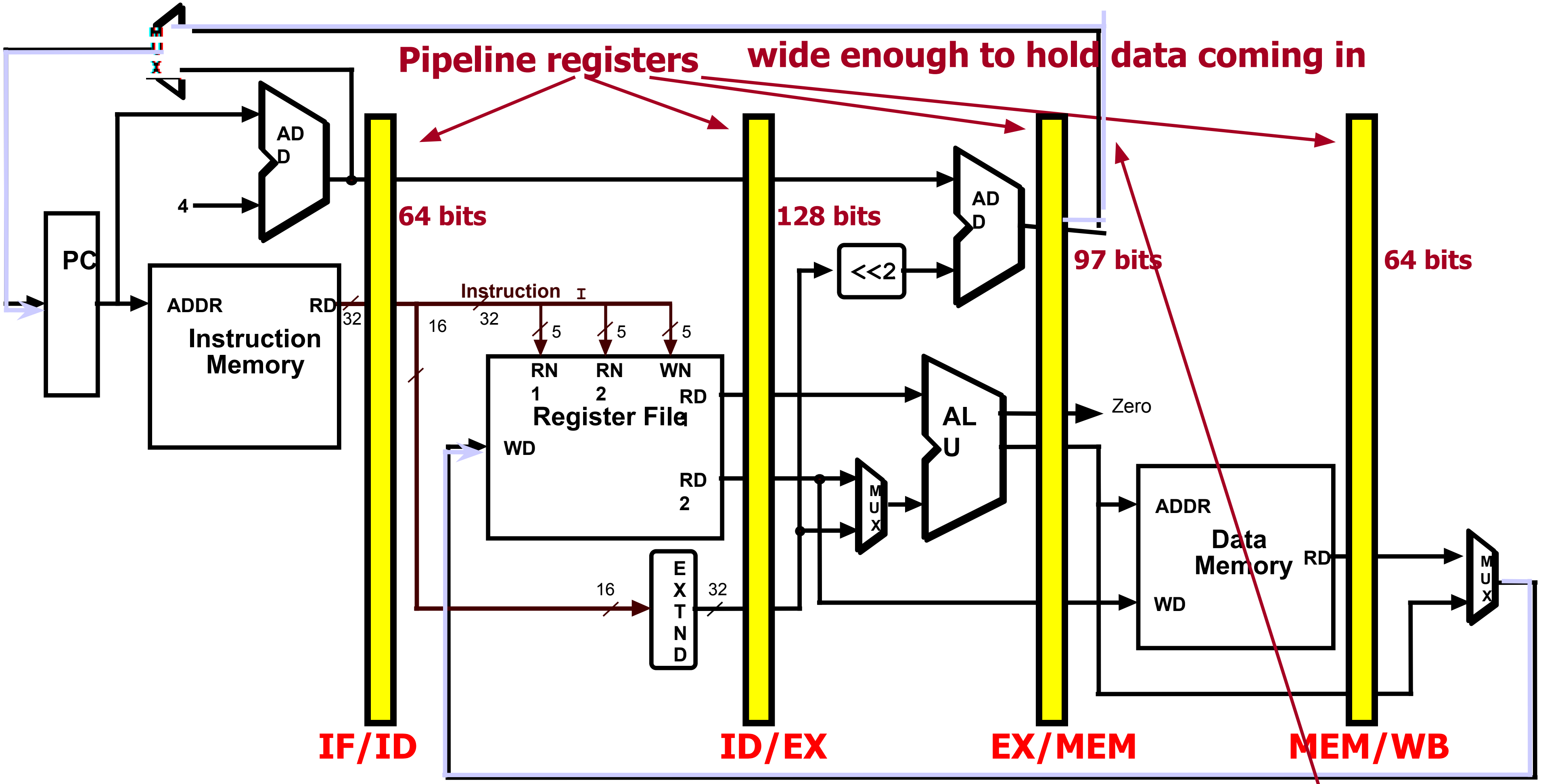
Pipelined Datapath

- ✔ The 5 steps in instruction execution are:
  1. Instruction Fetch & PC Increment (**IF**)
  2. Instruction Decode and Register Read (**ID**)
  3. Execution or calculate address (**EX**)
  4. Memory access (**MEM**)
  5. Write result into register (**WB**)
- ✔ Review: single-cycle processor
  1. all 5 steps done in a single clock cycle
  2. dedicated hardware required for each step

Review - Single-Cycle Datapath “Steps”



Pipelined Datapath



Pipeline registers

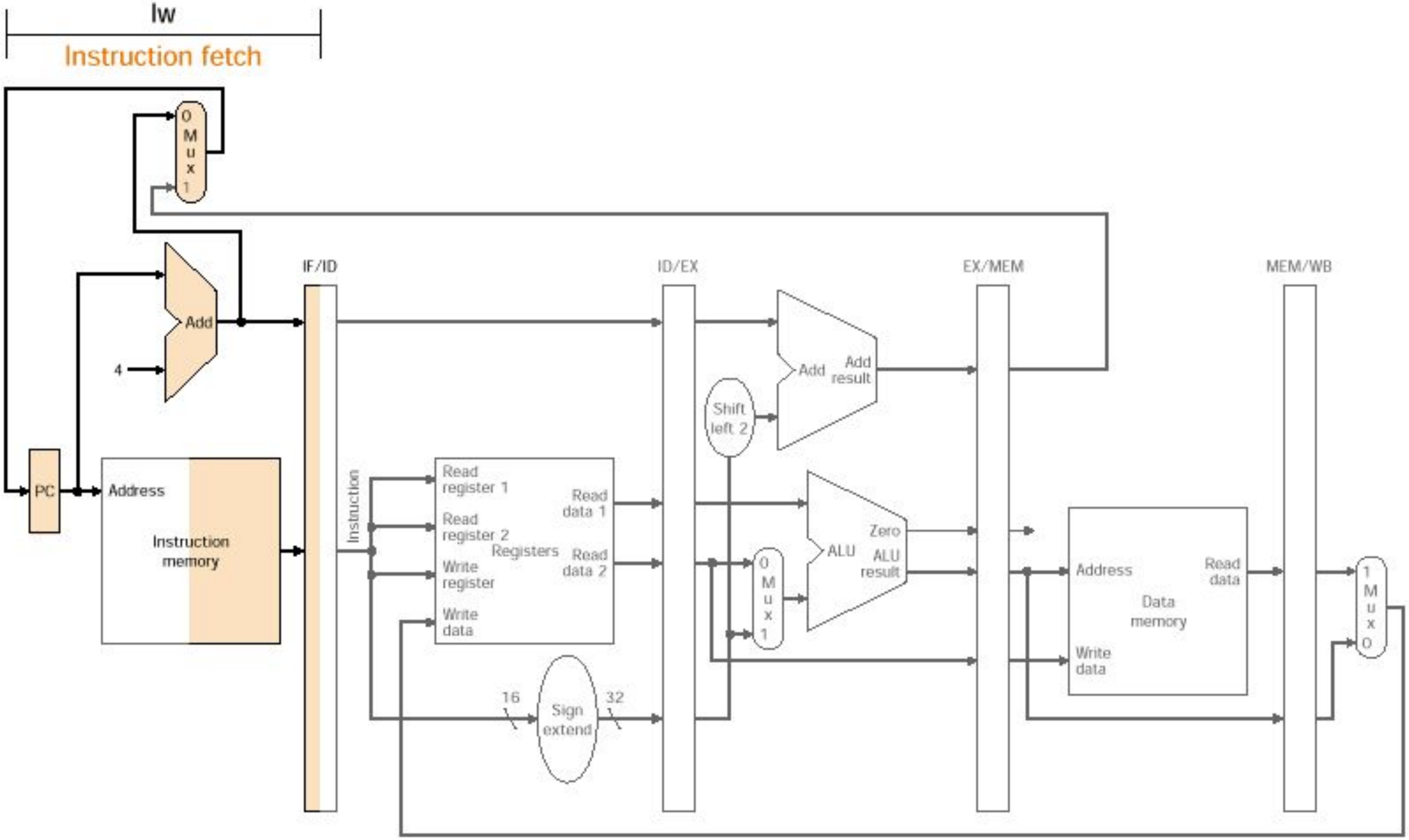
wide enough to hold data coming in

Only data flowing right to left may cause hazard..., why?

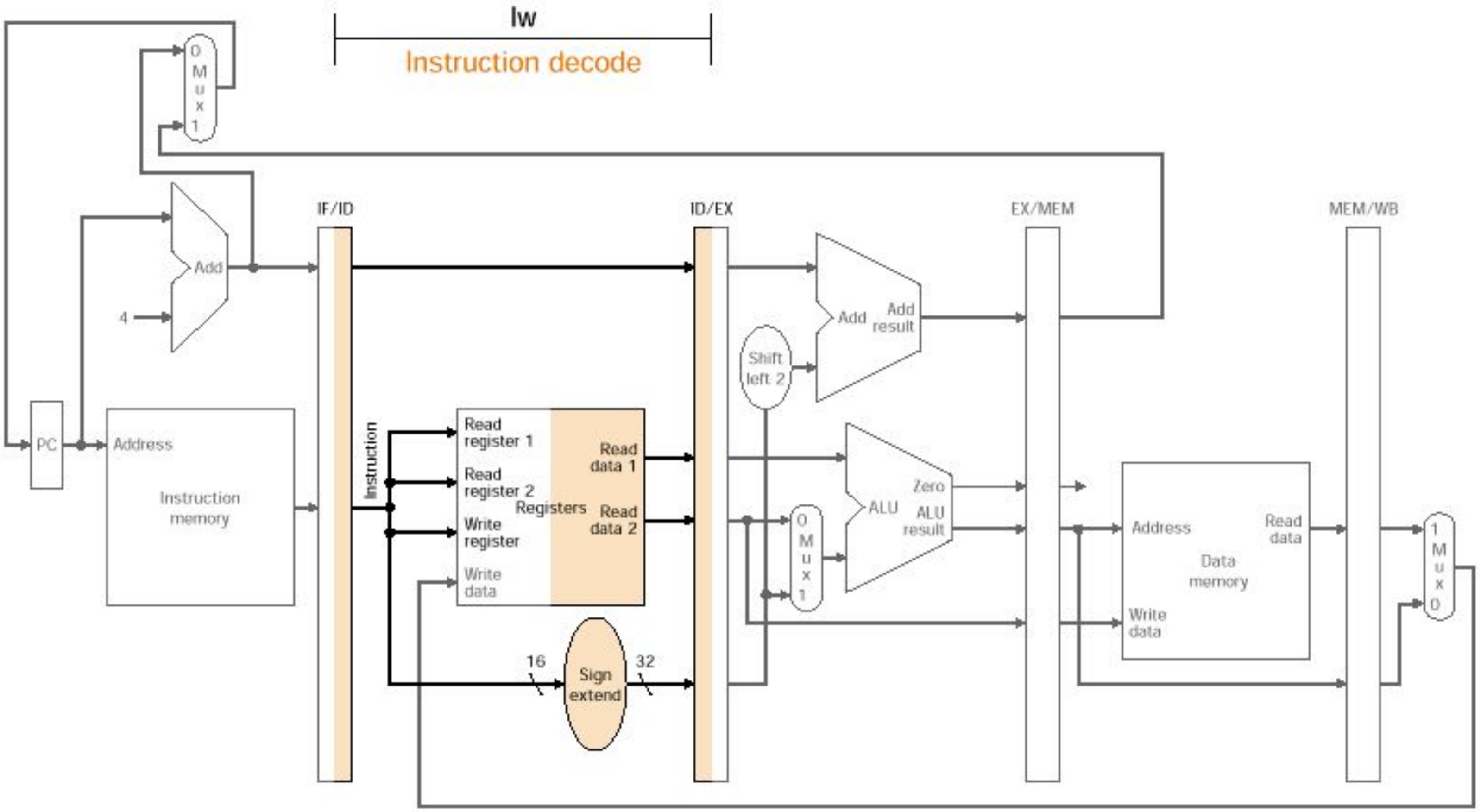
Pipelined Datapath

- ✔ No pipeline register at the end of write back stage. In this stage we write a register in the register file and any later instruction can get this data by reading this register. So no need for redundant register.
- ✔ Each component of the data path is associated with a single stage.
- ✔ Each register contains a portion of the instruction needed for that stage and latter stages.

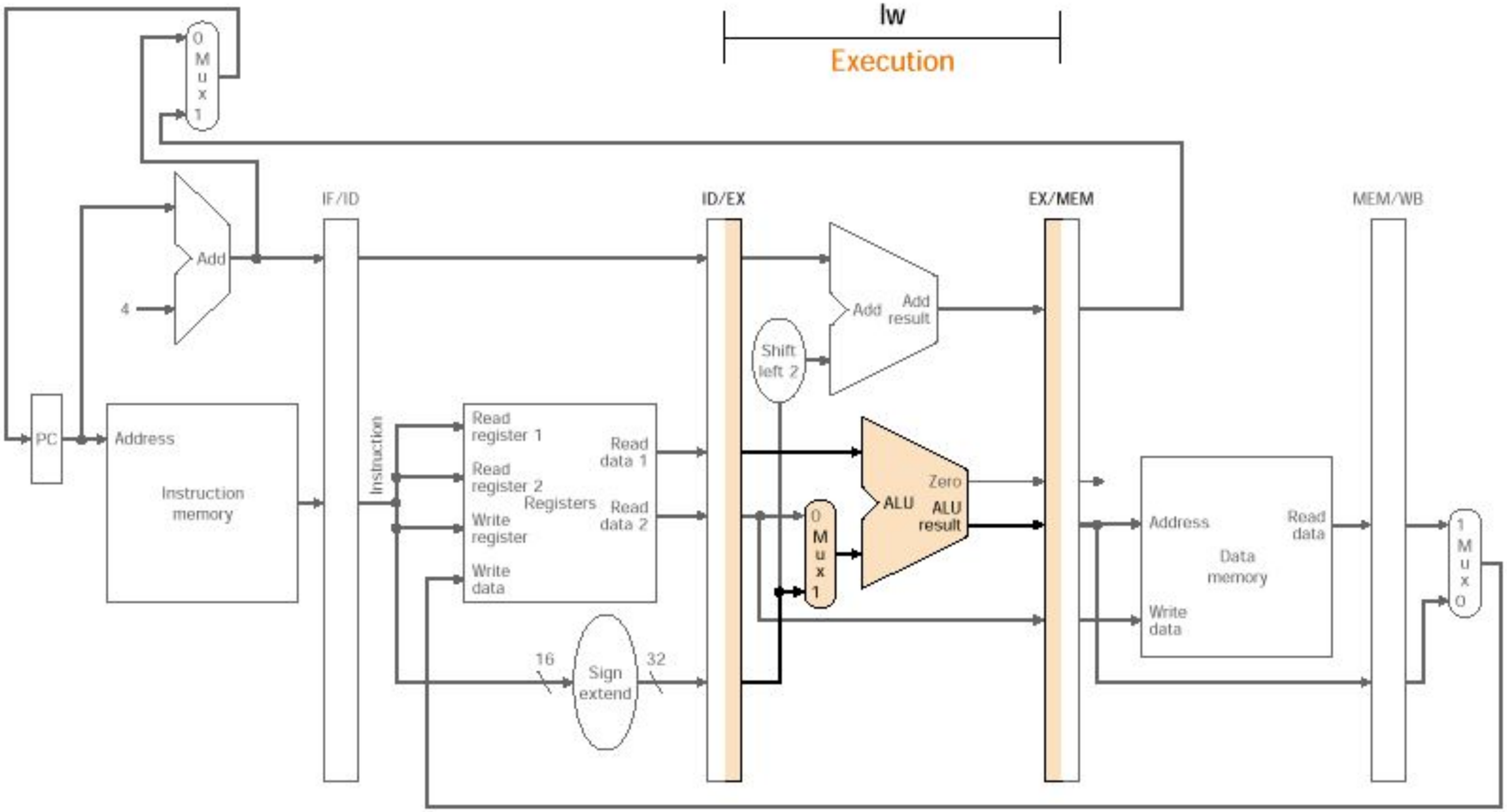
Pipeline Flow (lw example)



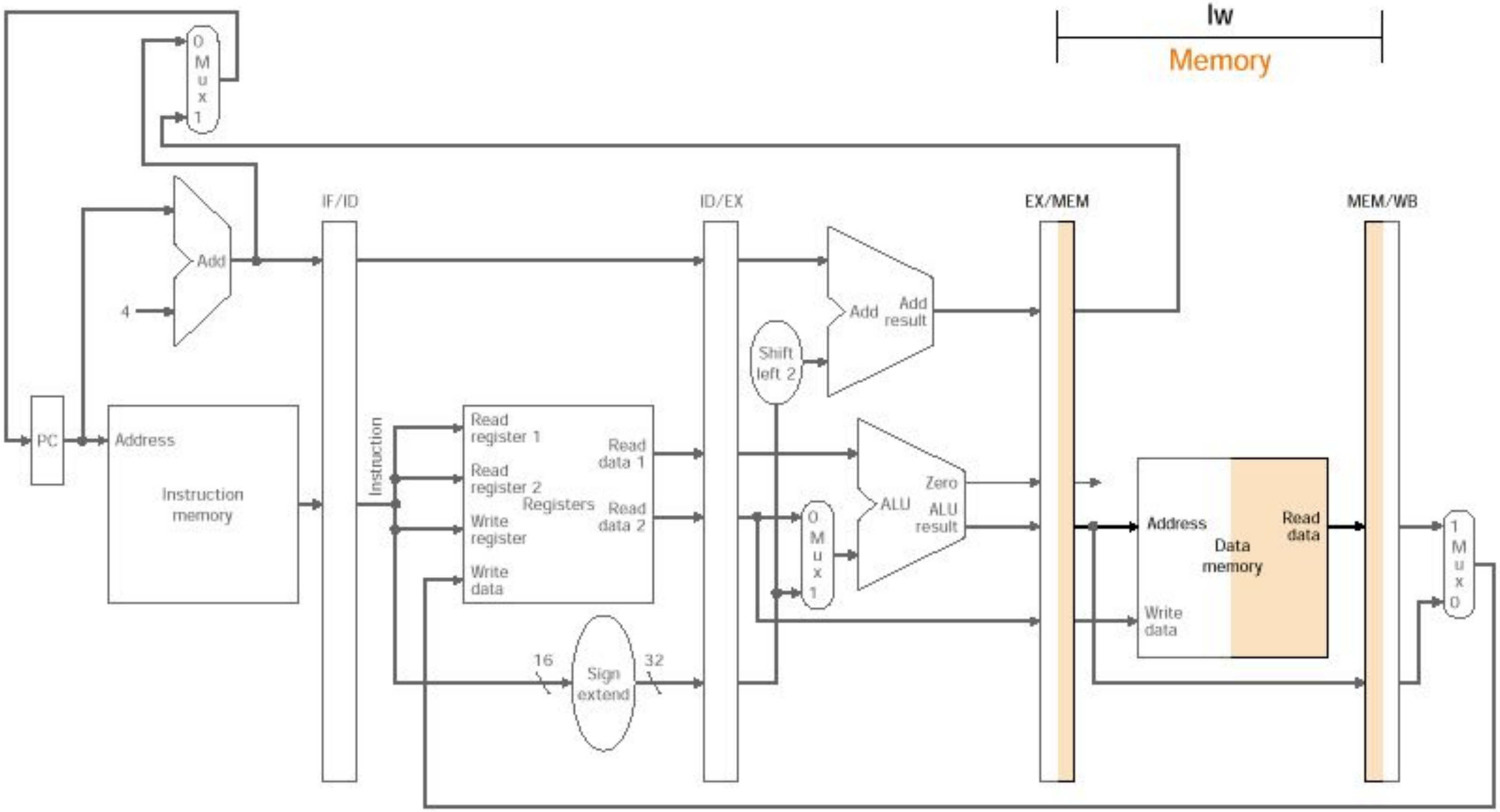
Pipeline Flow (lw example)



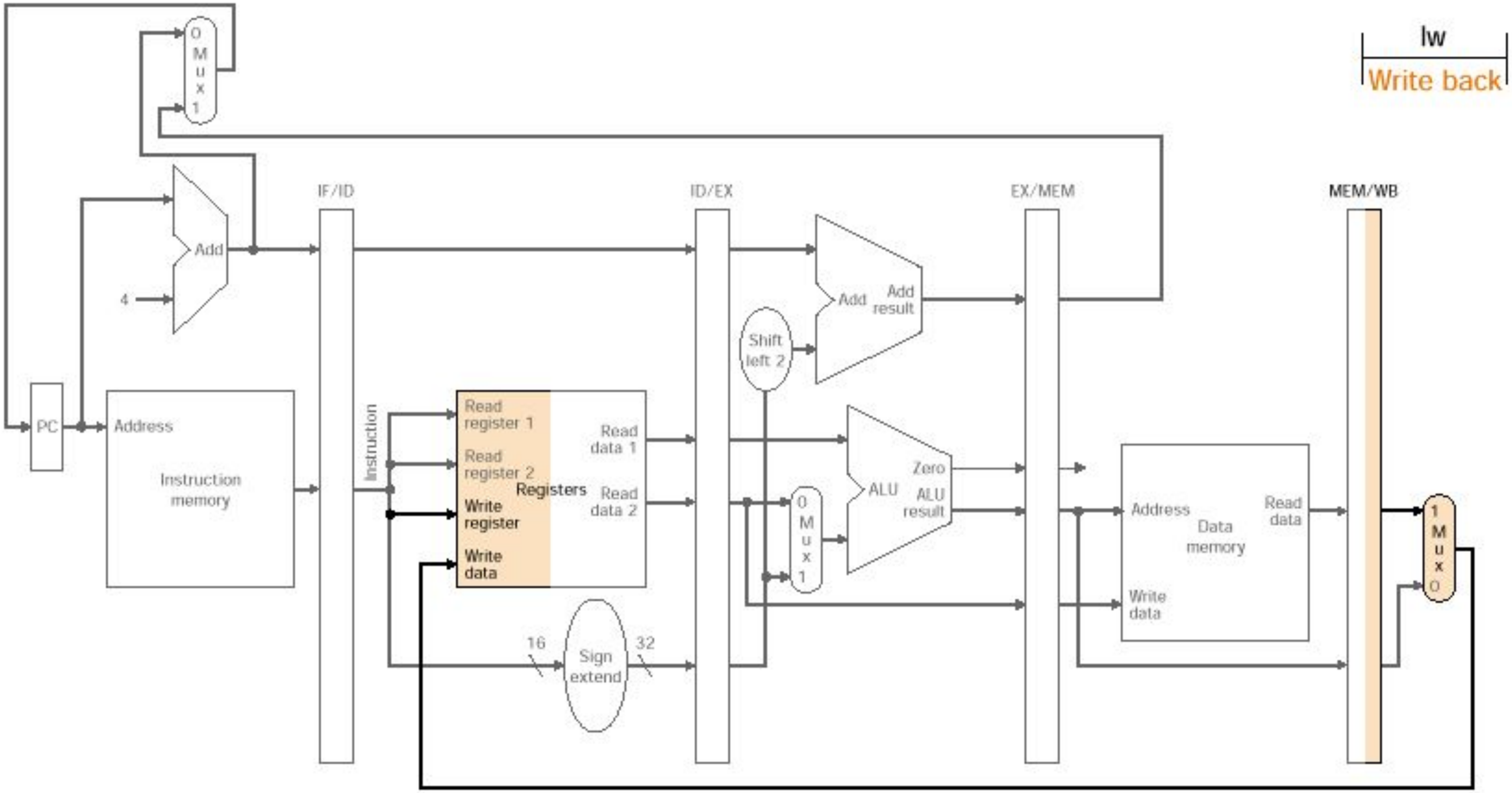
Pipeline Flow (lw example)



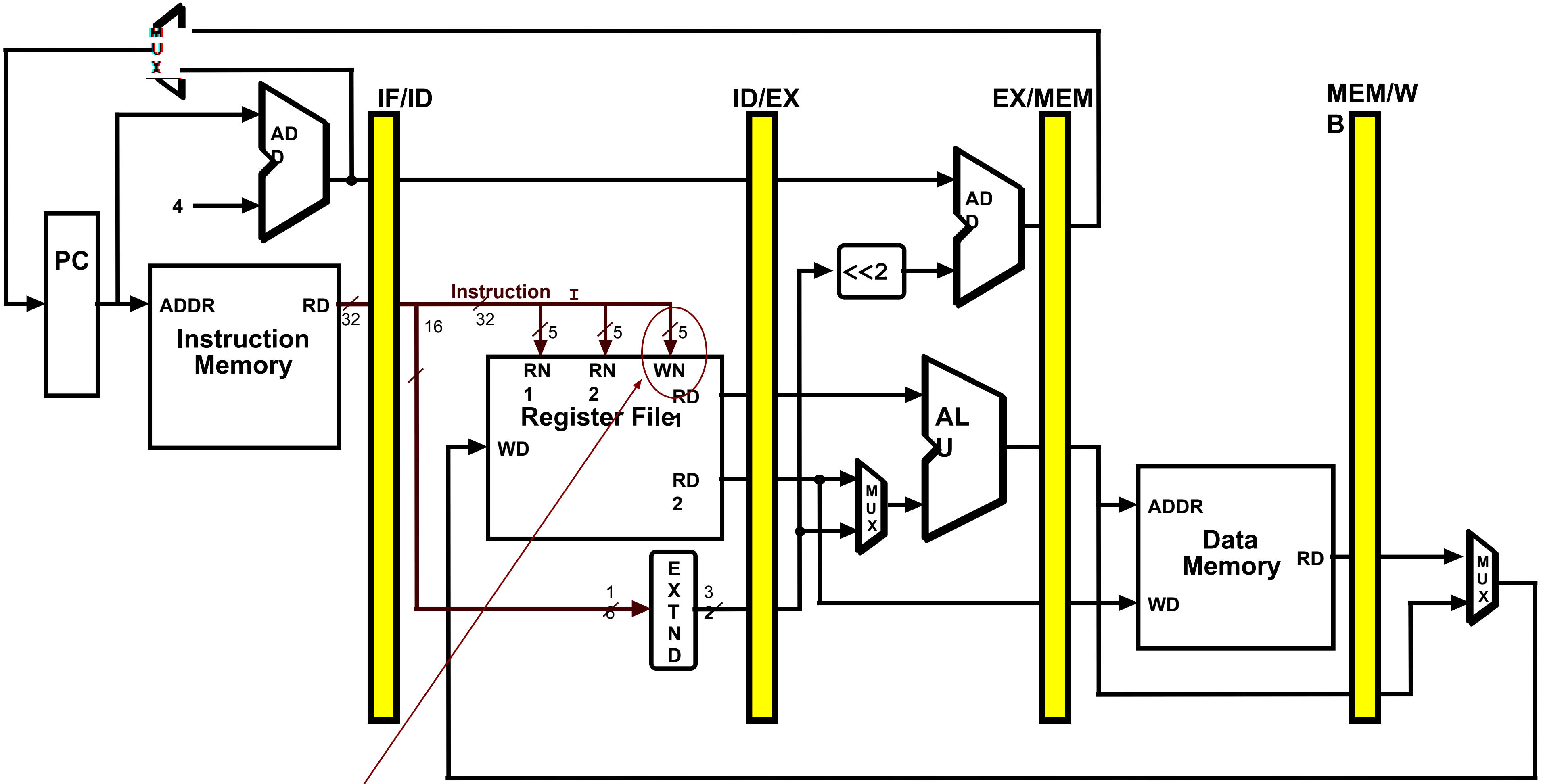
Pipeline Flow (lw example)



Pipeline Flow (lw example)

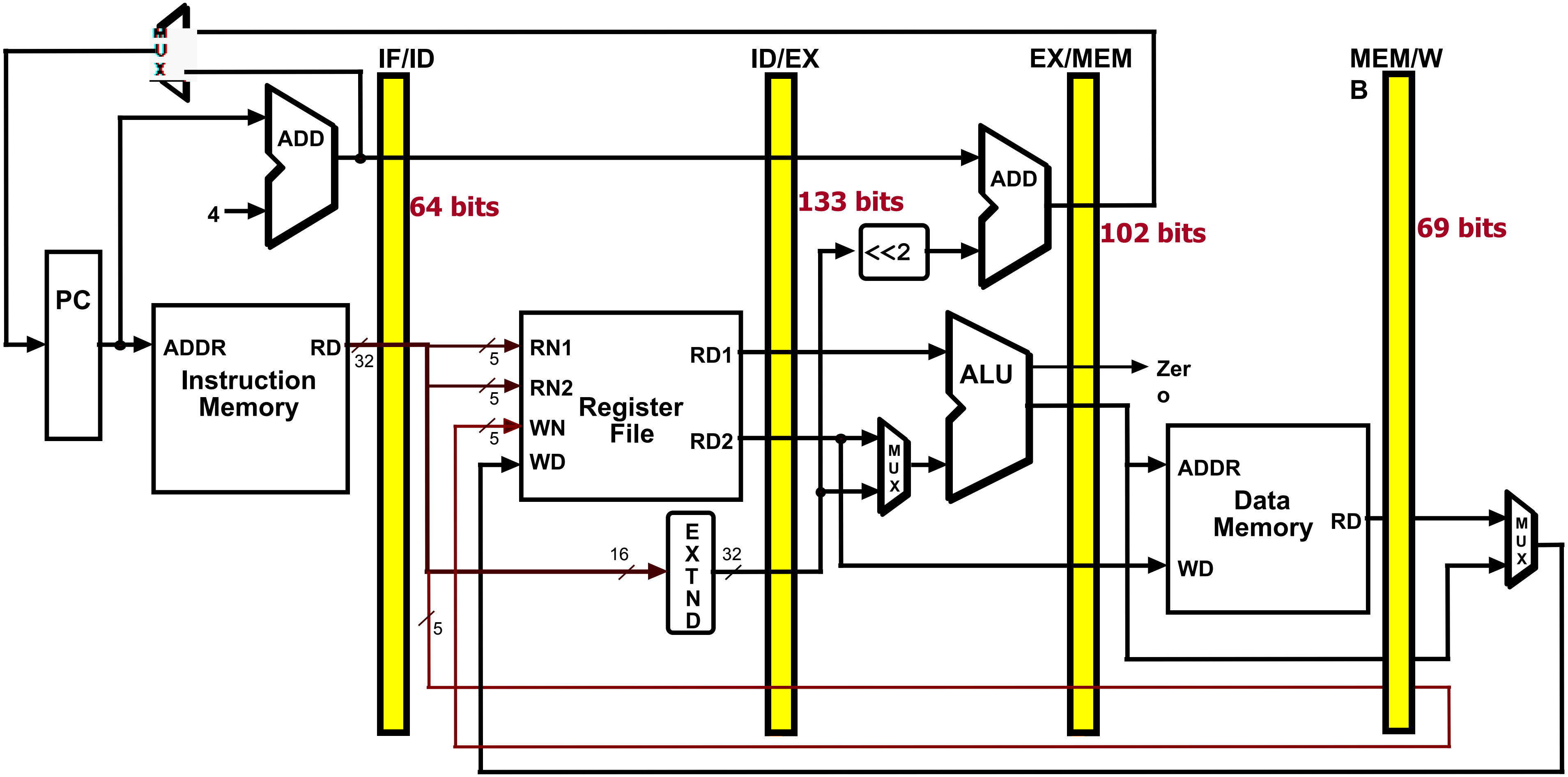


Bug in the Datapath



Write register number comes from another *later* instruction!

Corrected Datapath



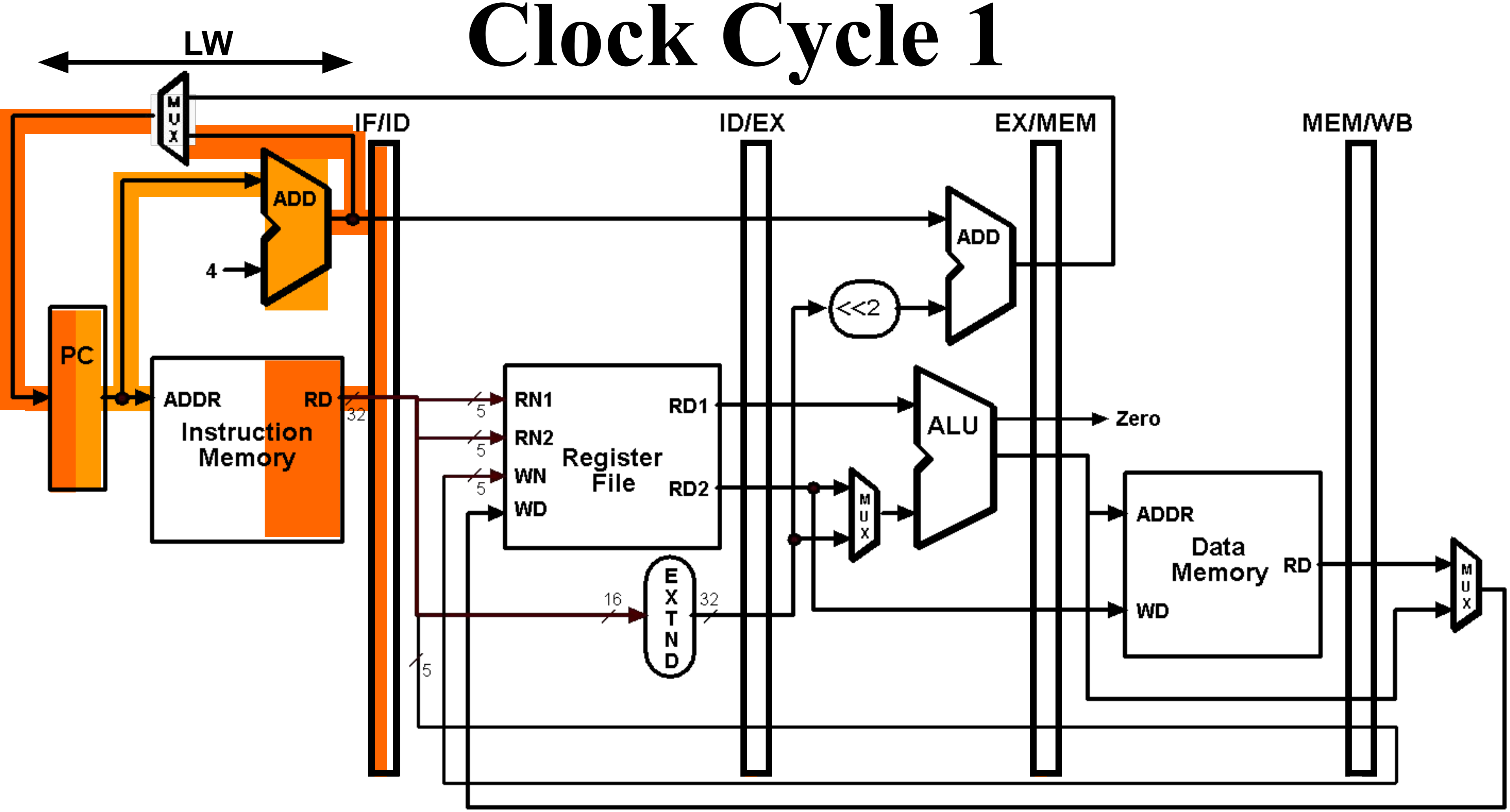
Destination register number is also passed through ID/EX, EX/MEM and MEM/WB registers, which are now wider by 5 bits

Pipelined Example

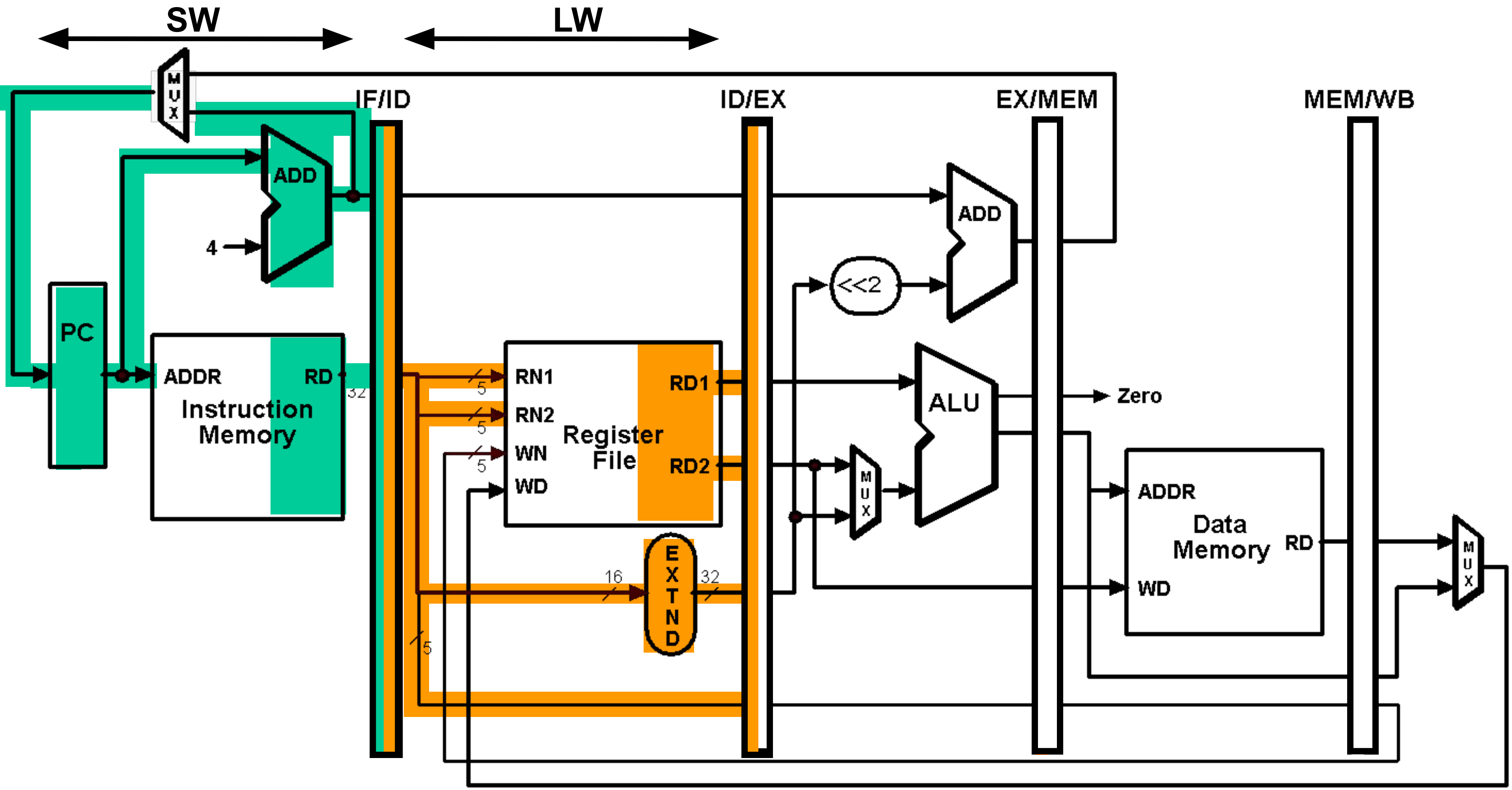
- Consider the following instruction sequence:

```
lw  $t0,  10($t1)
sw  $t3, 20($t4)
add $t5,  $t6,  $t7
sub $t8,  $t9,  $t10
```

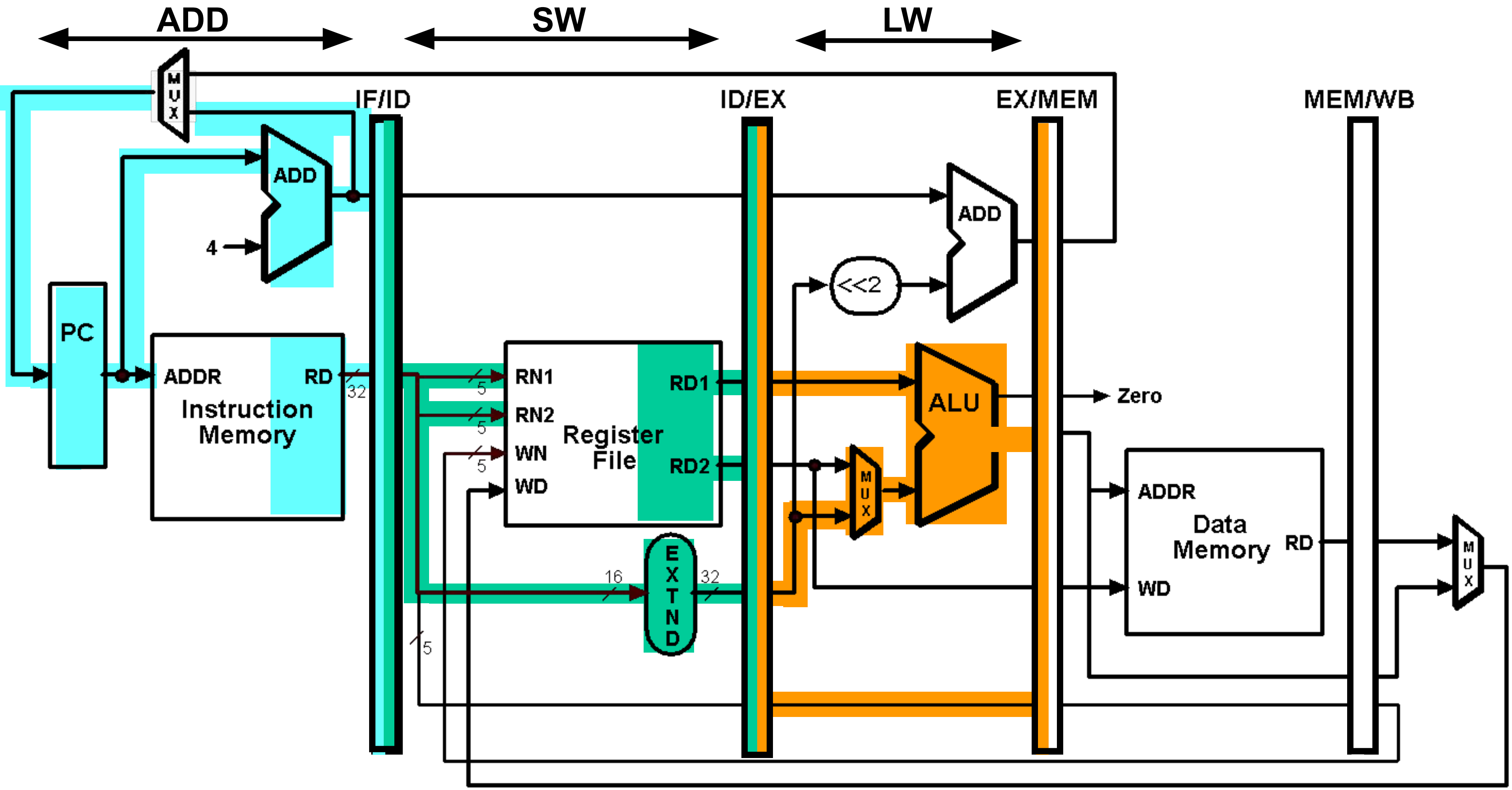
Single-Clock-Cycle Diagram: Clock Cycle 1



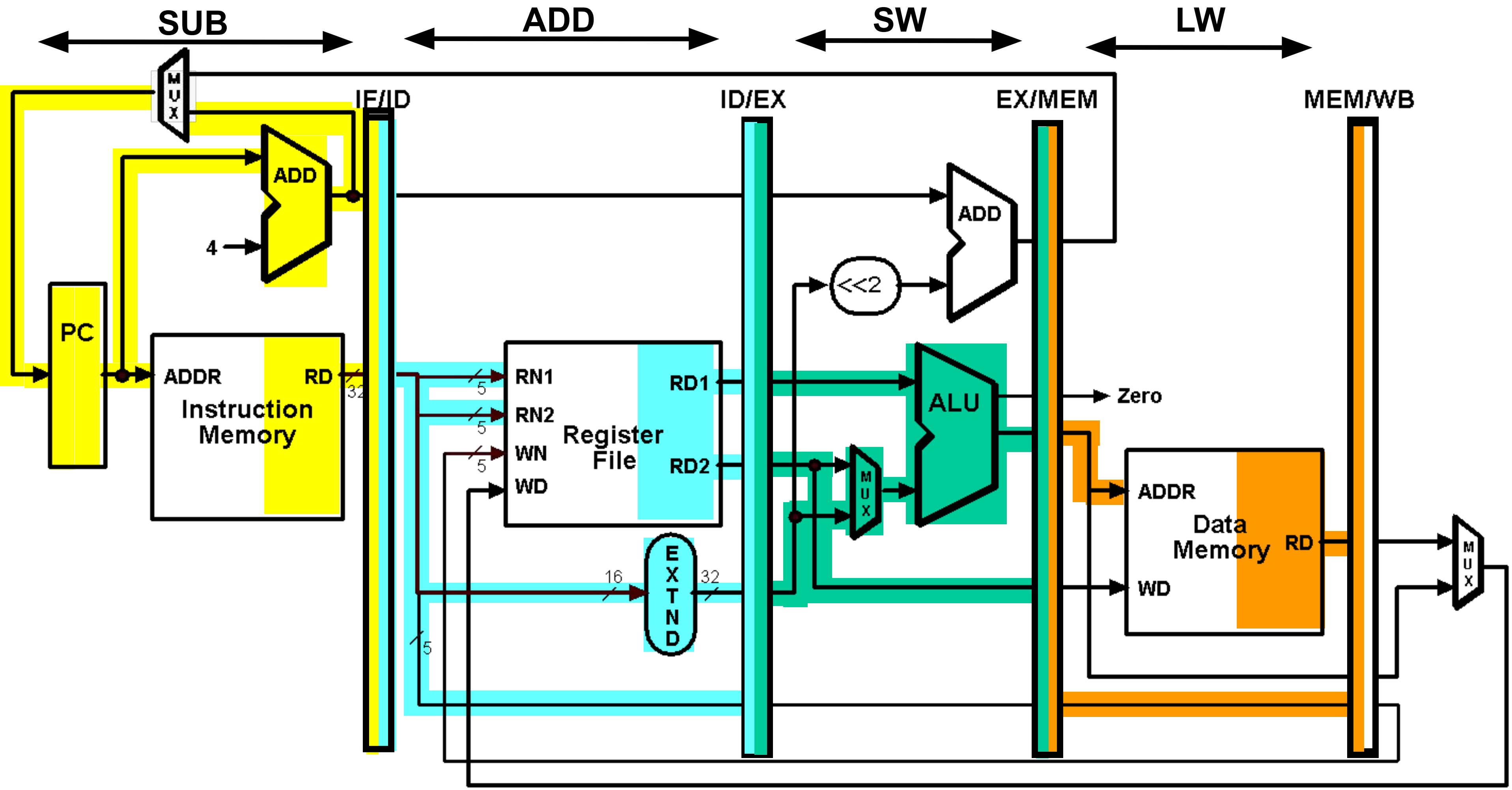
Single-Clock-Cycle Diagram: Clock Cycle 2



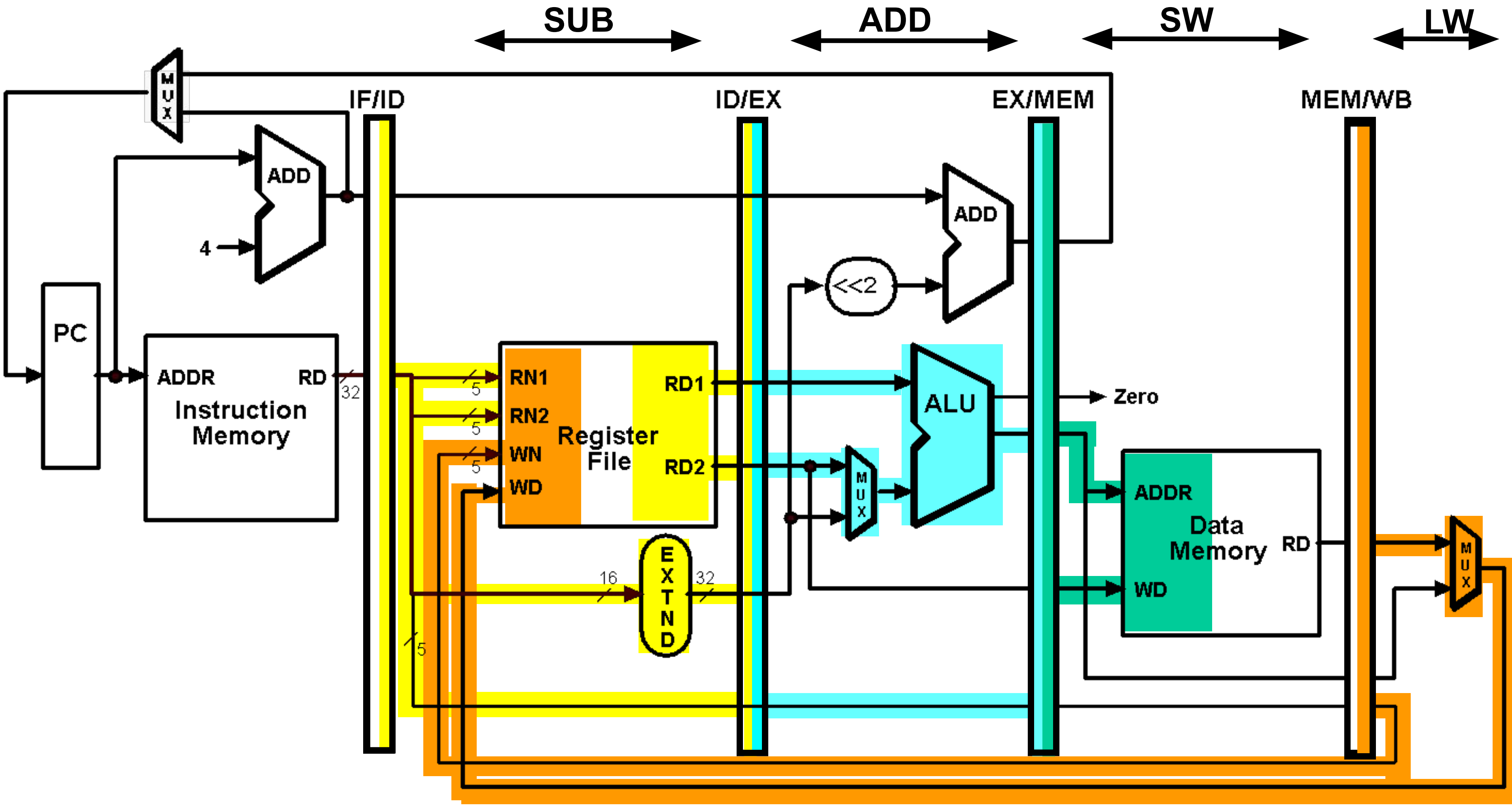
Single-Clock-Cycle Diagram: Clock Cycle 3



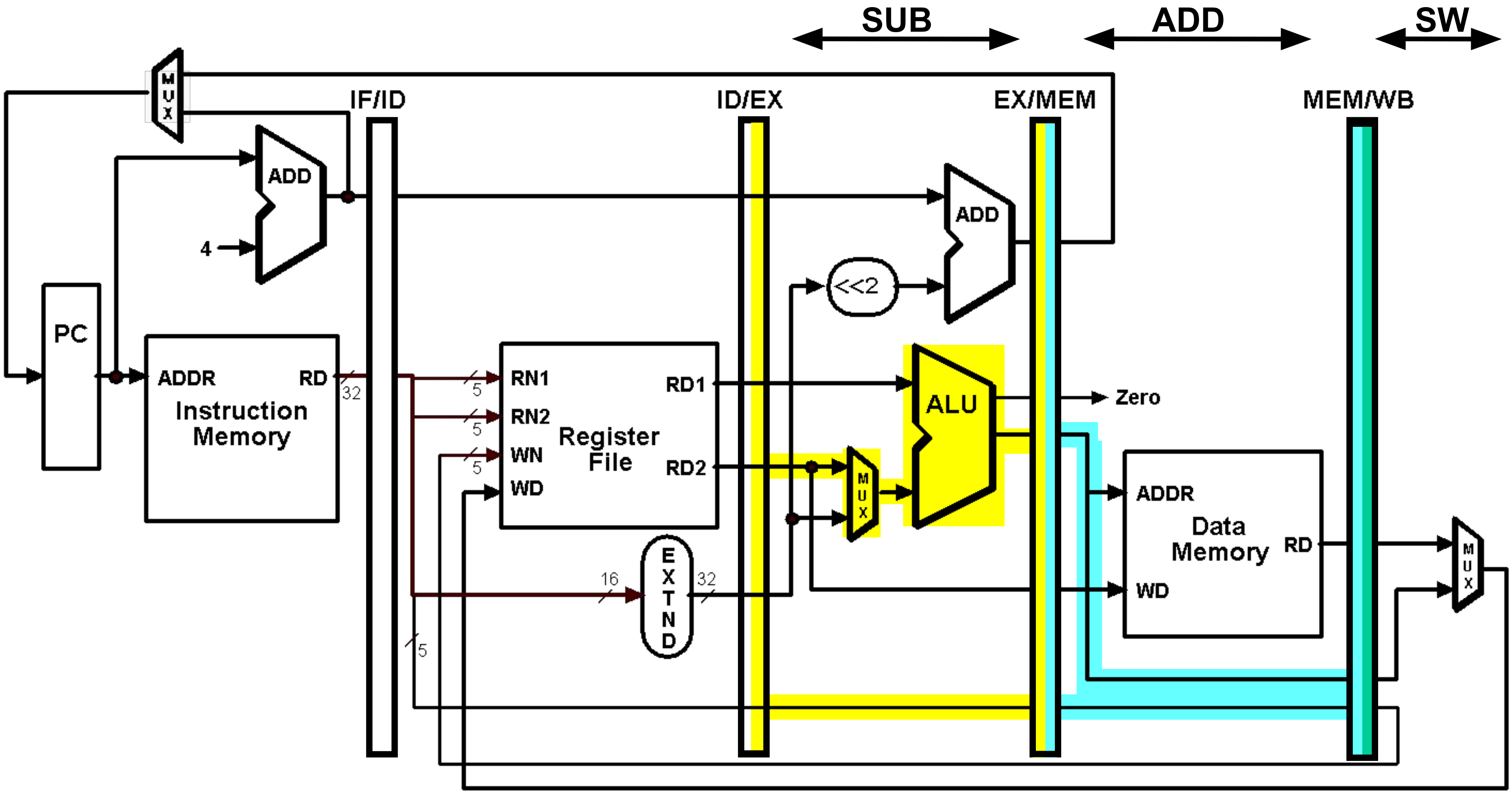
Single-Clock-Cycle Diagram: Clock Cycle 4



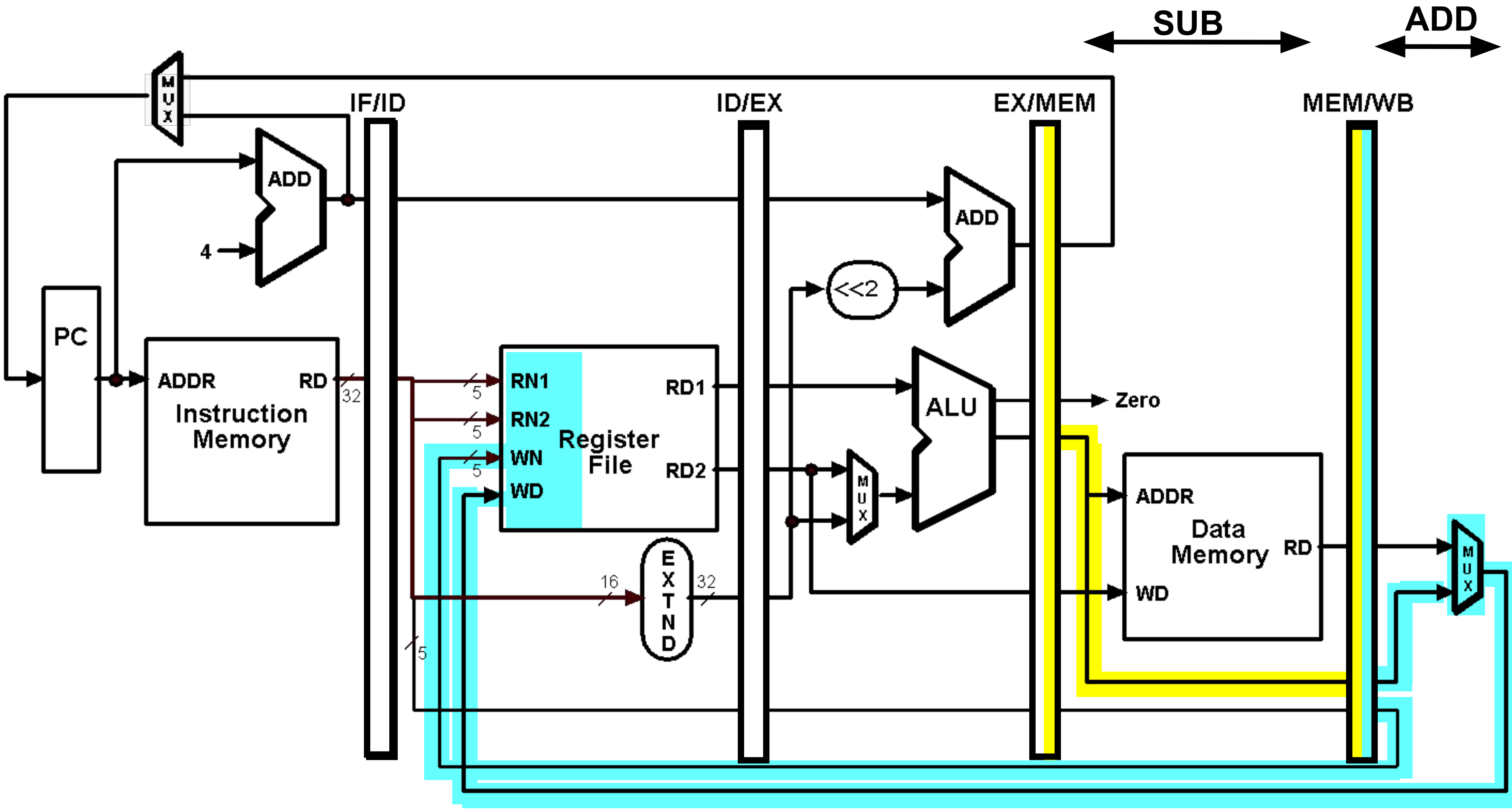
Single-Clock-Cycle Diagram: Clock Cycle 5



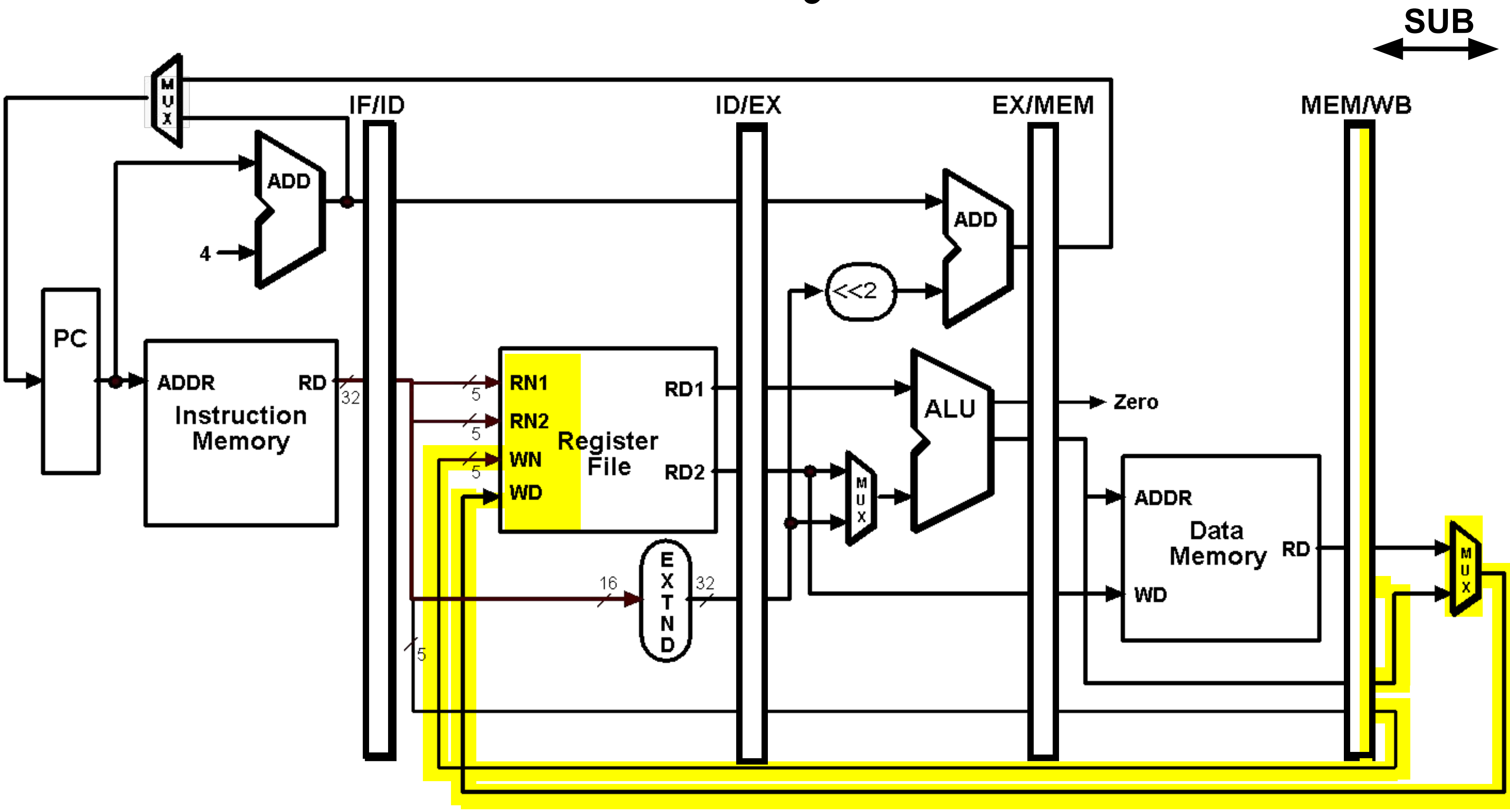
Single-Clock-Cycle Diagram: Clock Cycle 6



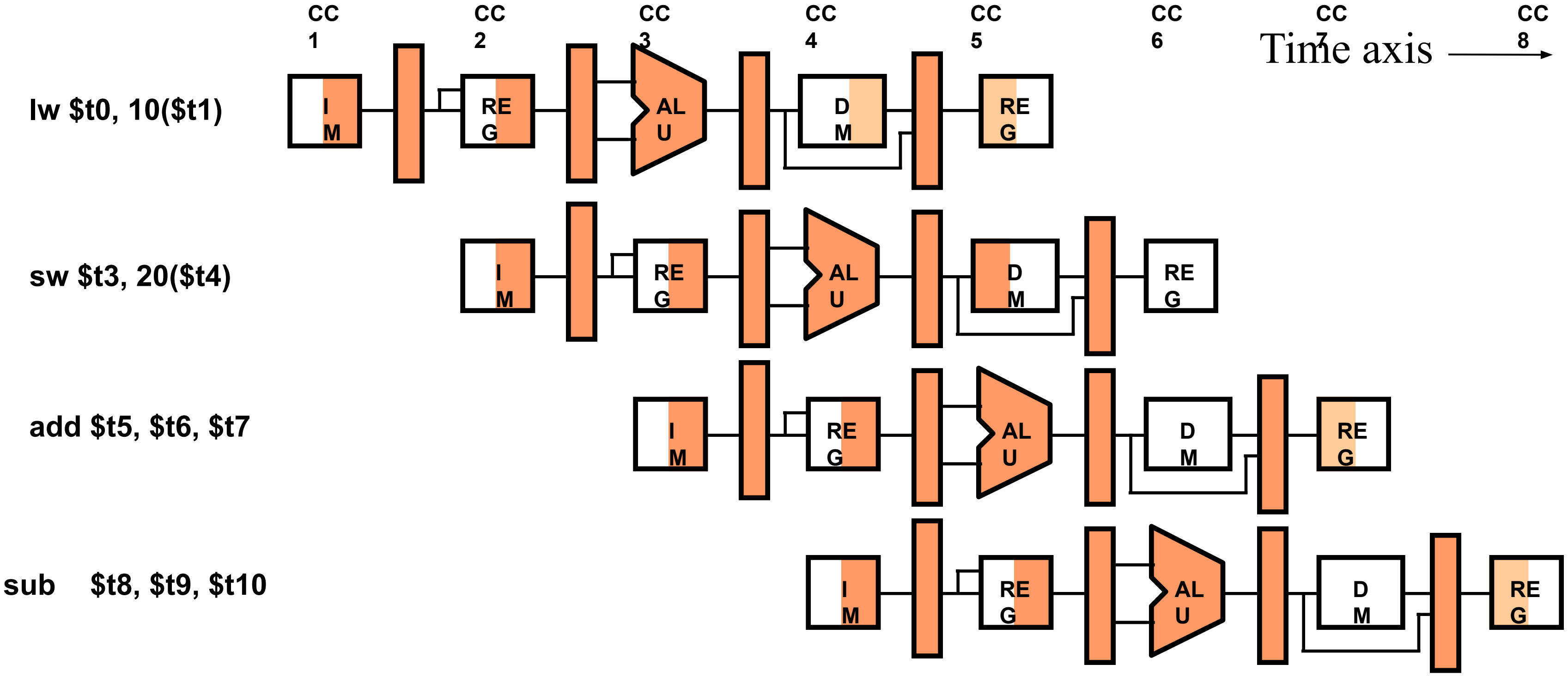
Single-Clock-Cycle Diagram: Clock Cycle 7



Single-Clock-Cycle Diagram: Clock Cycle 8



Alternative View – Multiple-Clock-Cycle Diagram



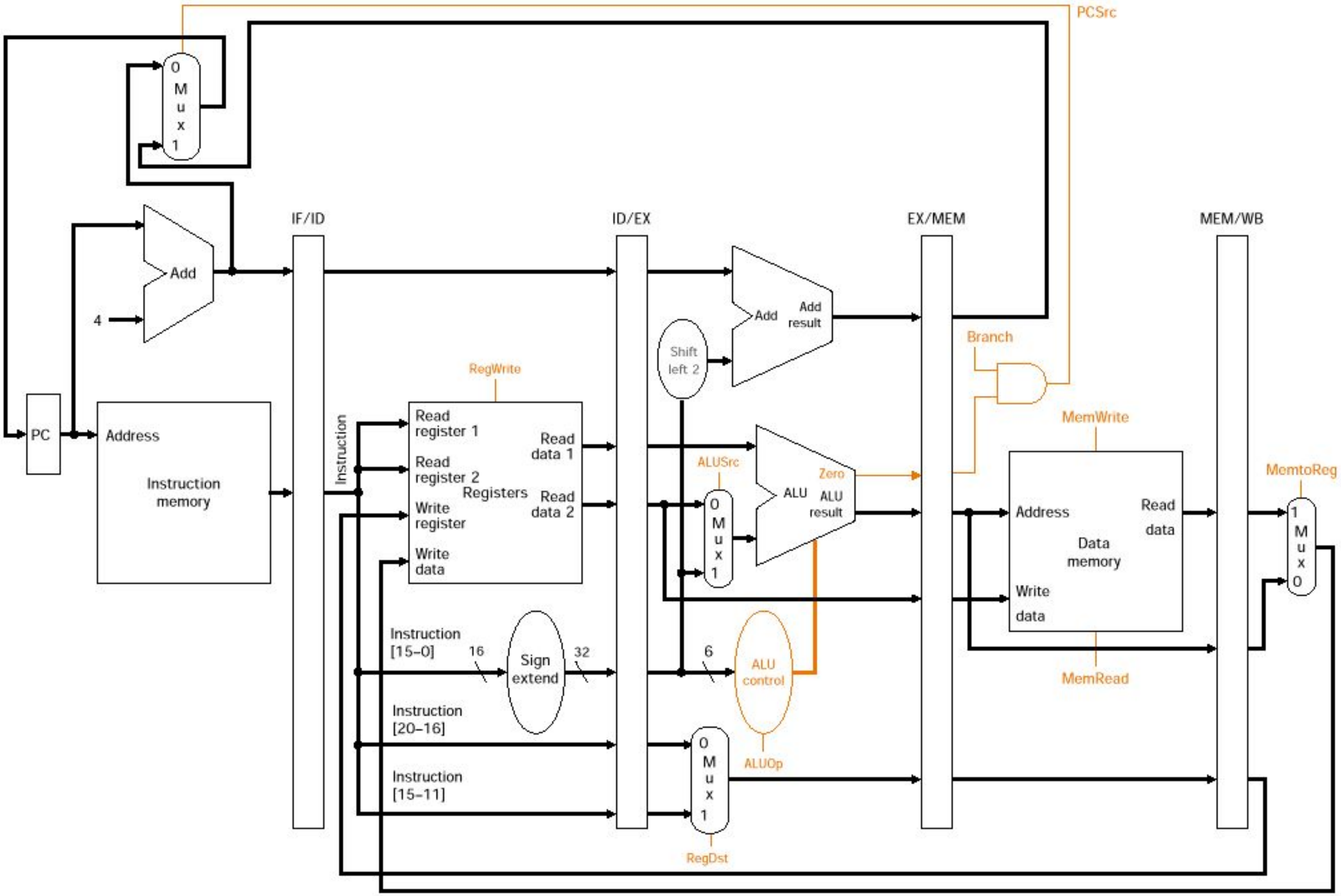
lw $t0, 10($t1)

sw $t3, 20($t4)

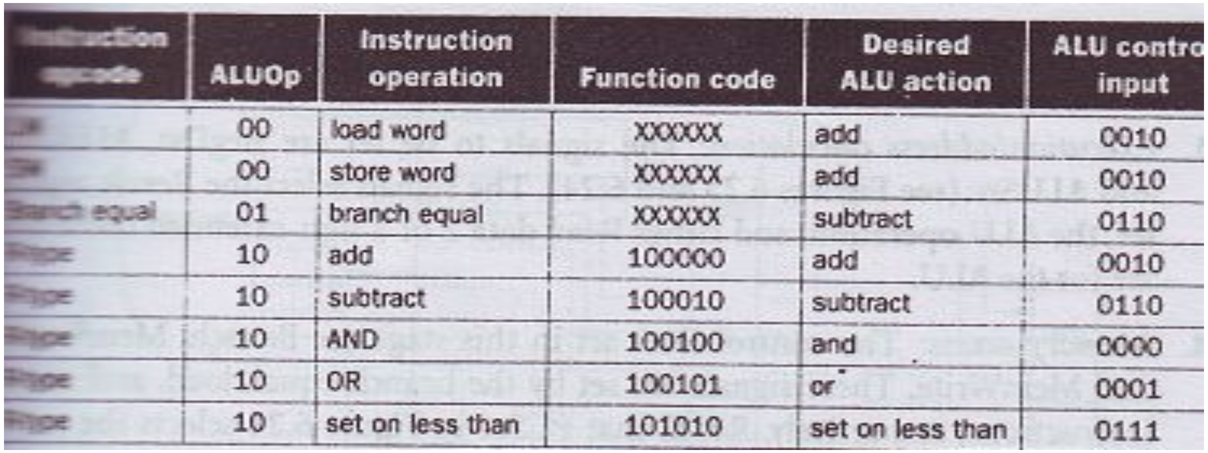
add $t5, $t6, $t7

sub $t8, $t9, $t10

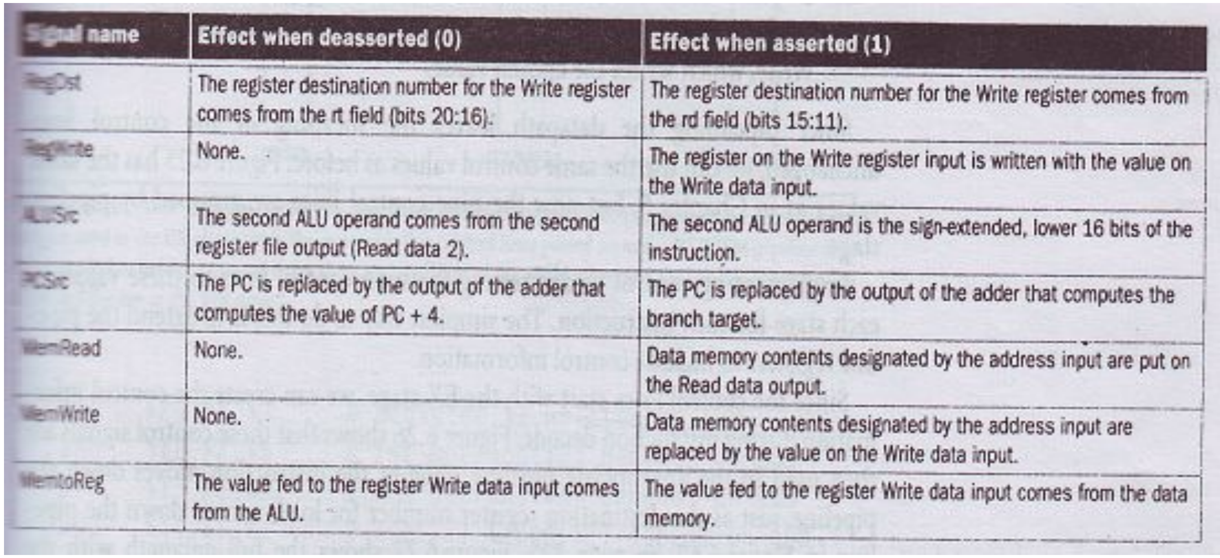
Pipelined Datapath with Control Signals



Control Signal for the Pipeline Datapath

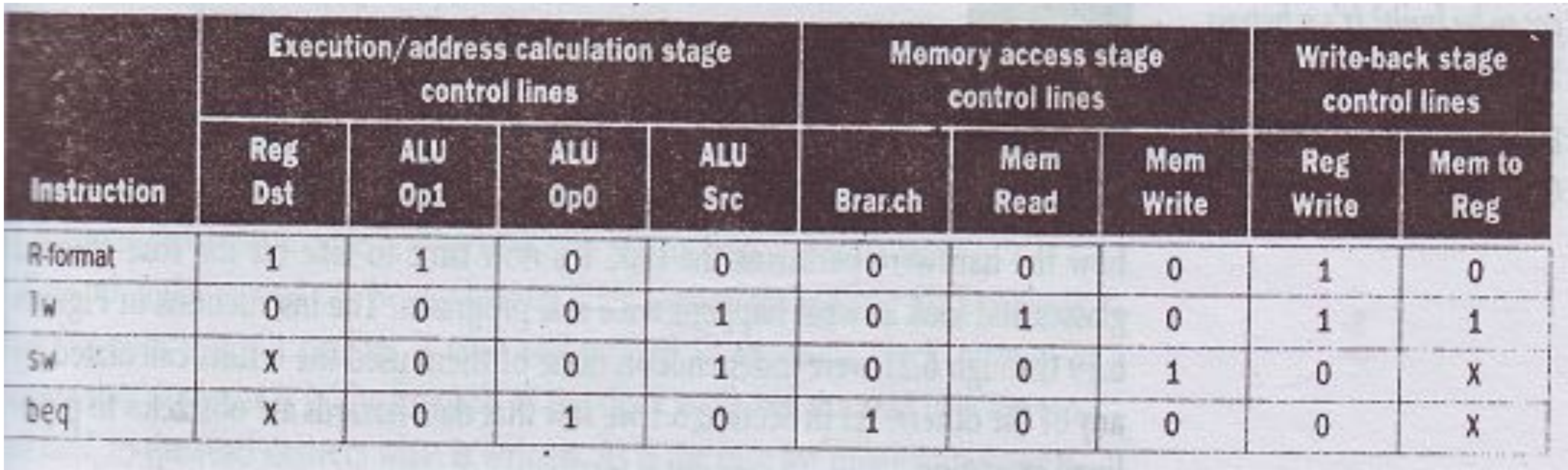


Instruction opcode	ALUOp	Instruction operation	Function code	Desired ALU action	ALU contro input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100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



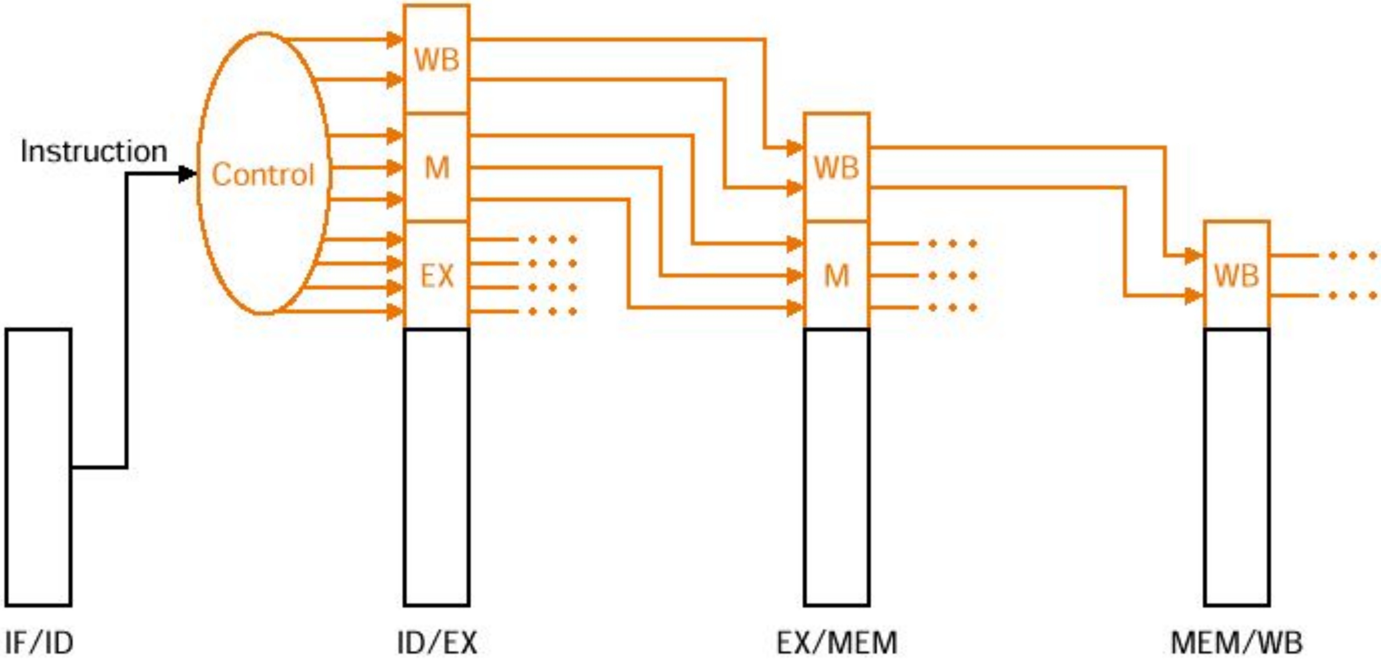
Signal name	Effect when deasserted (0)	Effect when asserted (1)
RegDst	The register destination number for the Write register comes from the rt field (bits 20:16).	The register destination number for the Write register comes from the rd field (bits 15:11).
RegWrite	None.	The register on the Write register 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



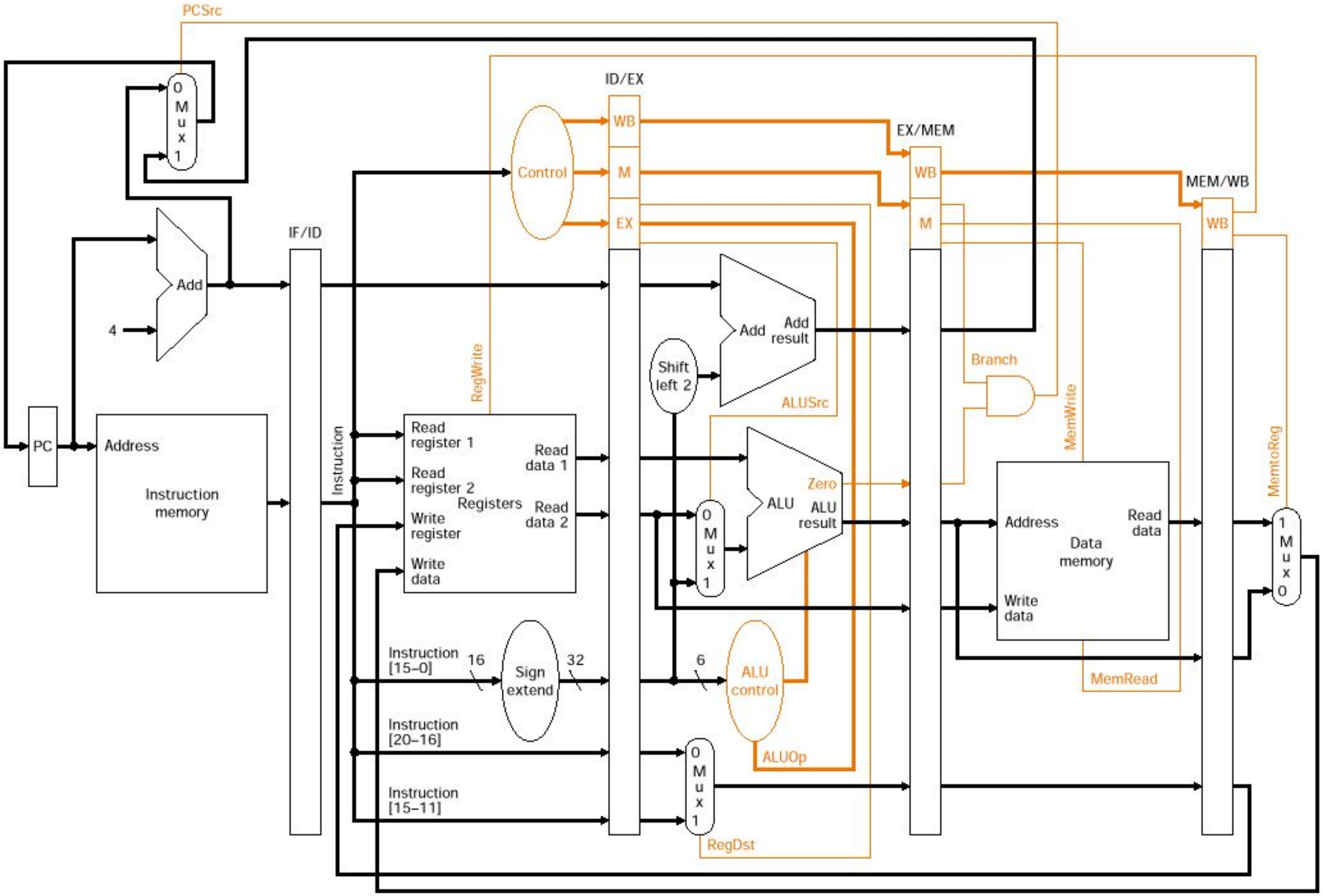
Instruction	Execution/address calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	Reg Dst	ALU Op1	ALU Op0	ALU Src	Brar.ch	Mem Read	Mem Write	Reg Write	Mem to Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

Control Signals Generation



Pipelined Datapath with control Signals Connected

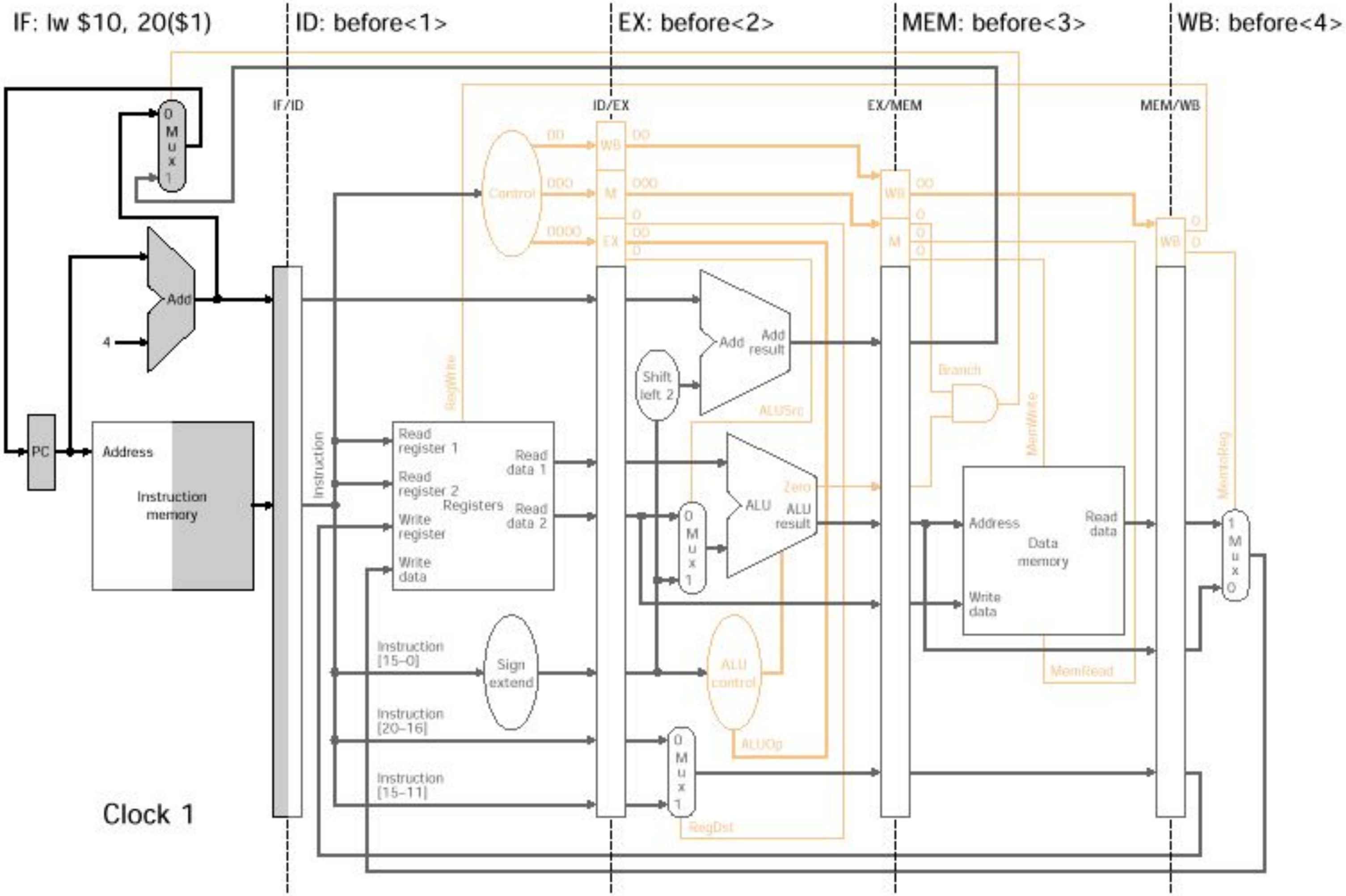
https://www.youtube.com/watch?v=u



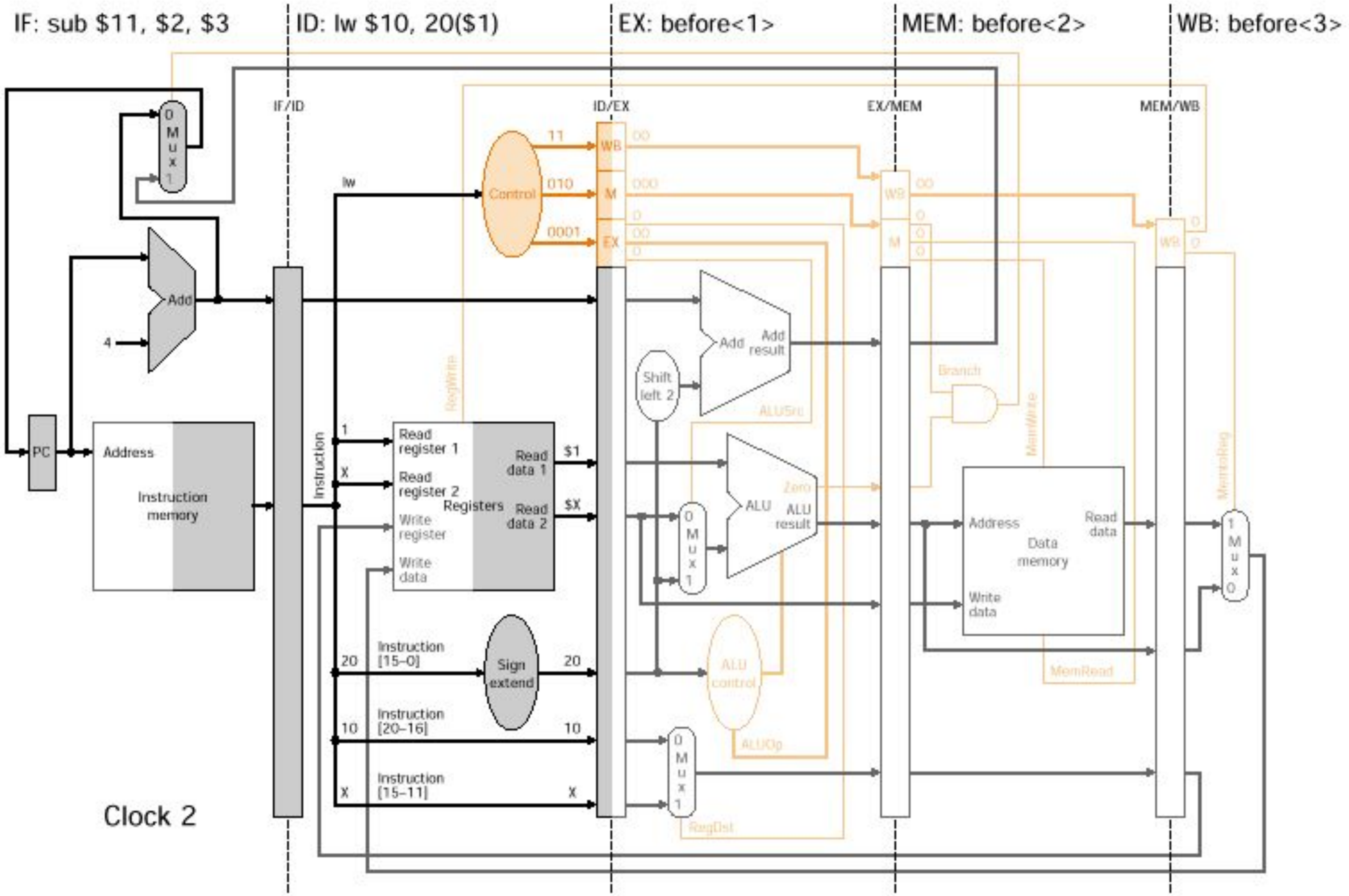
The Grand Example

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

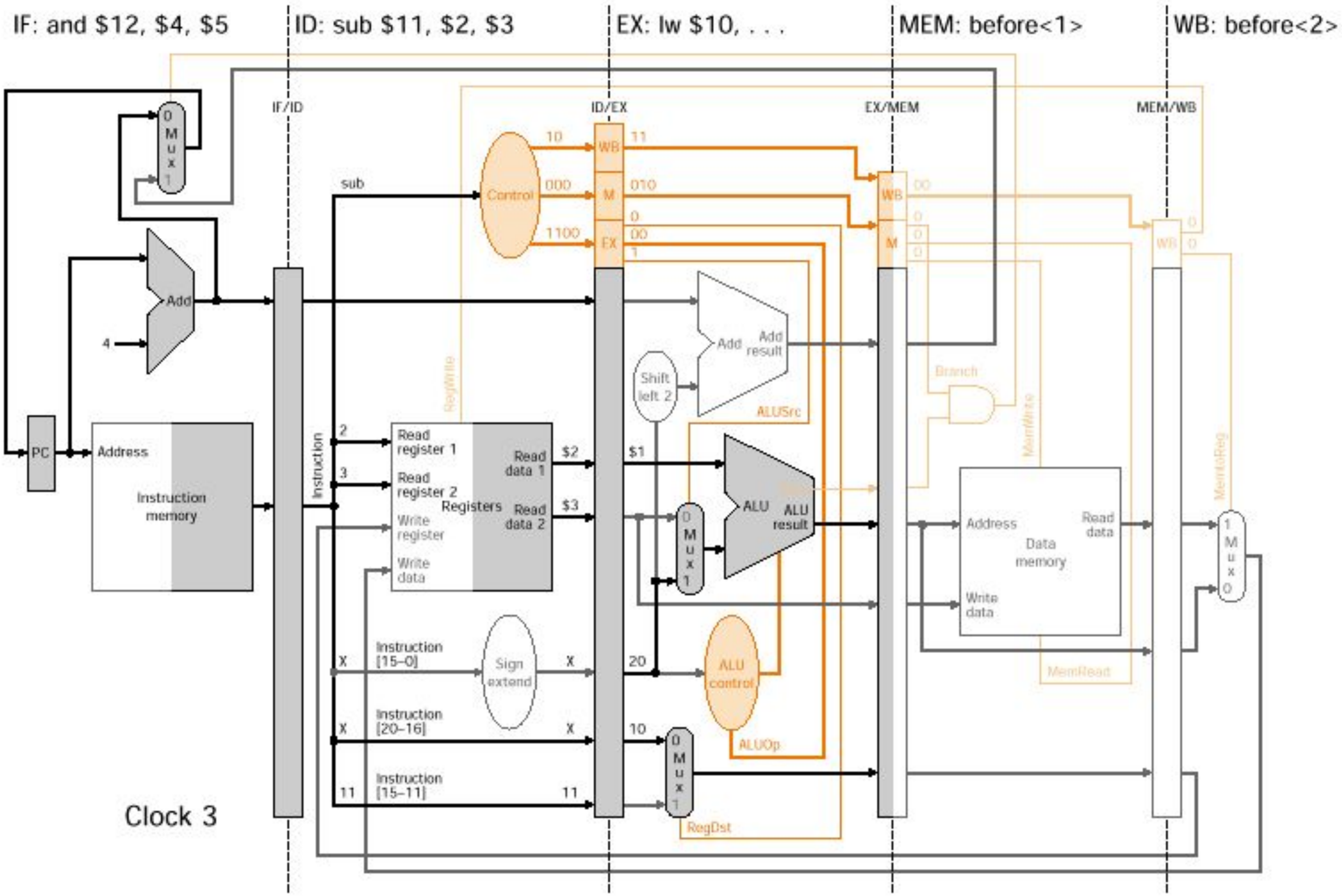
The Grand Example



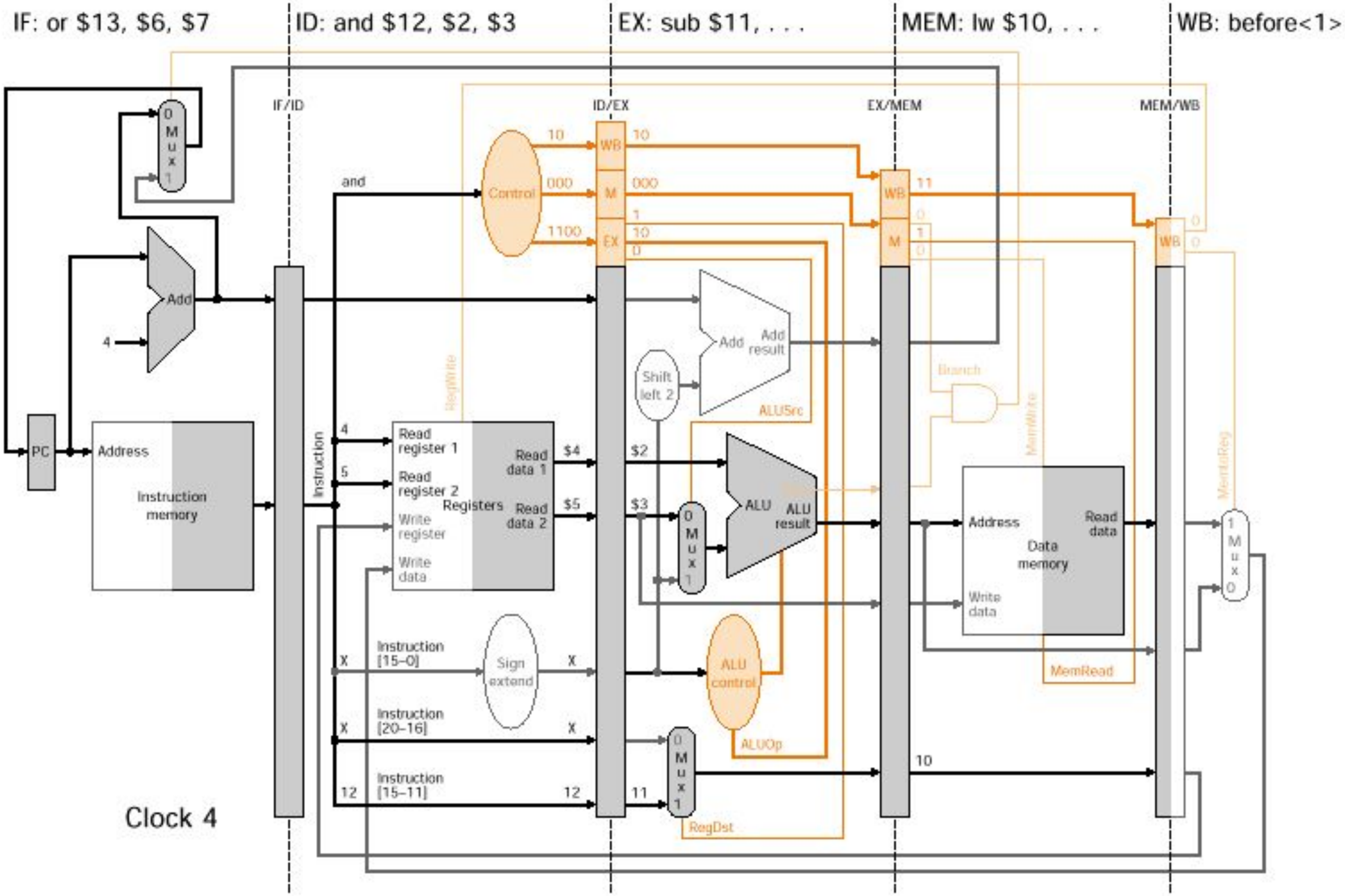
The Grand Example



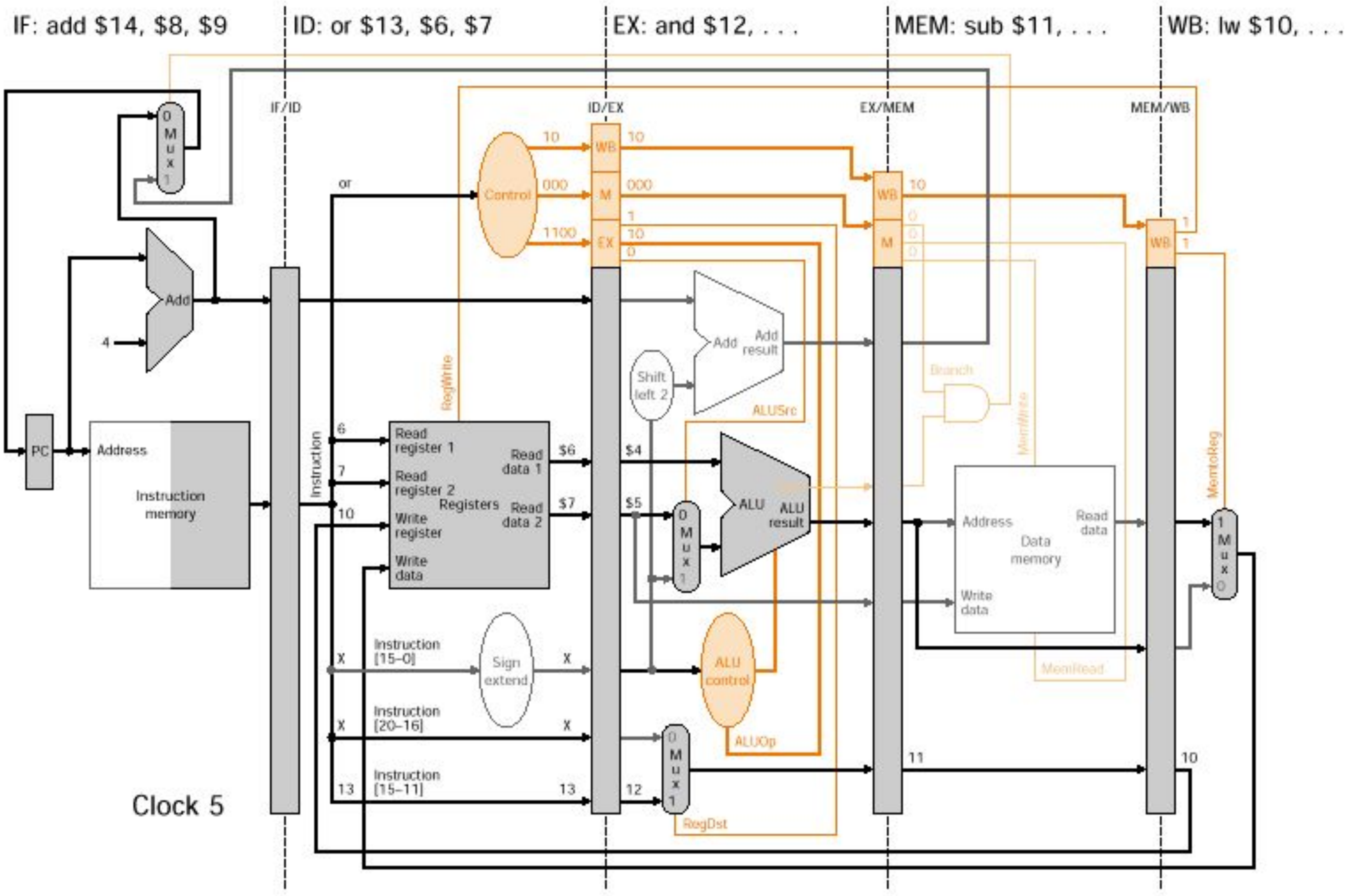
The Grand Example



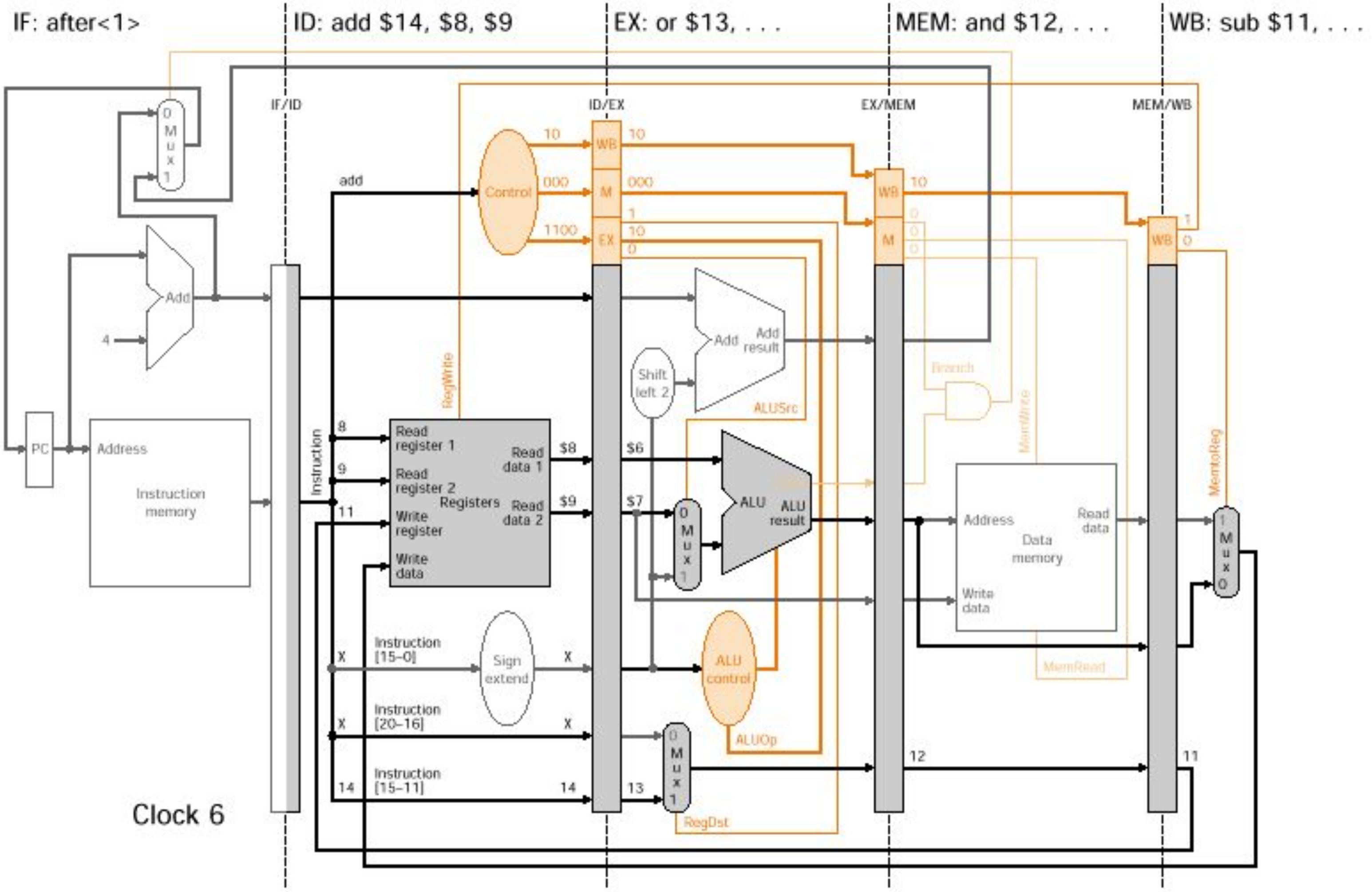
The Grand Example



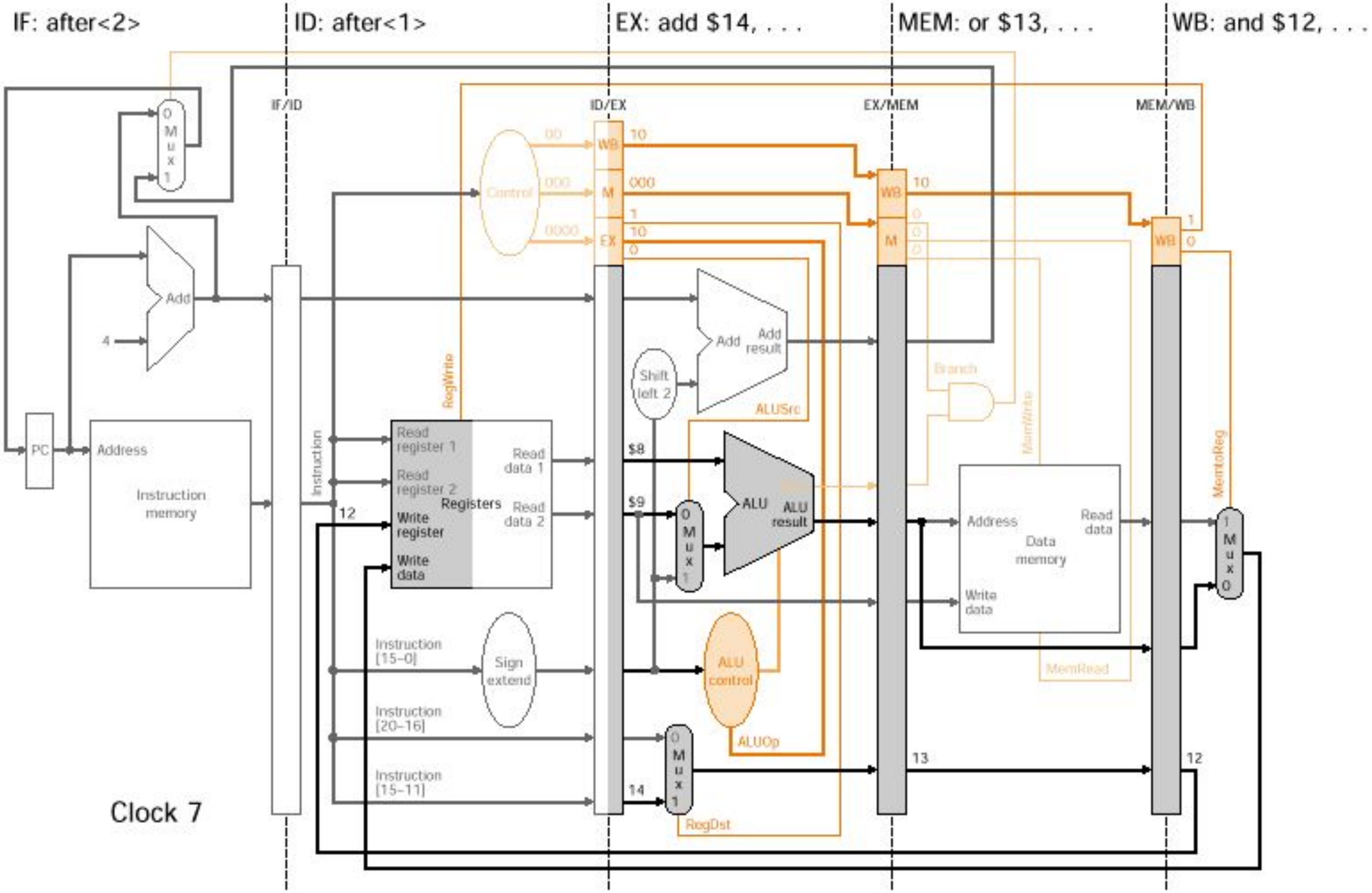
The Grand Example



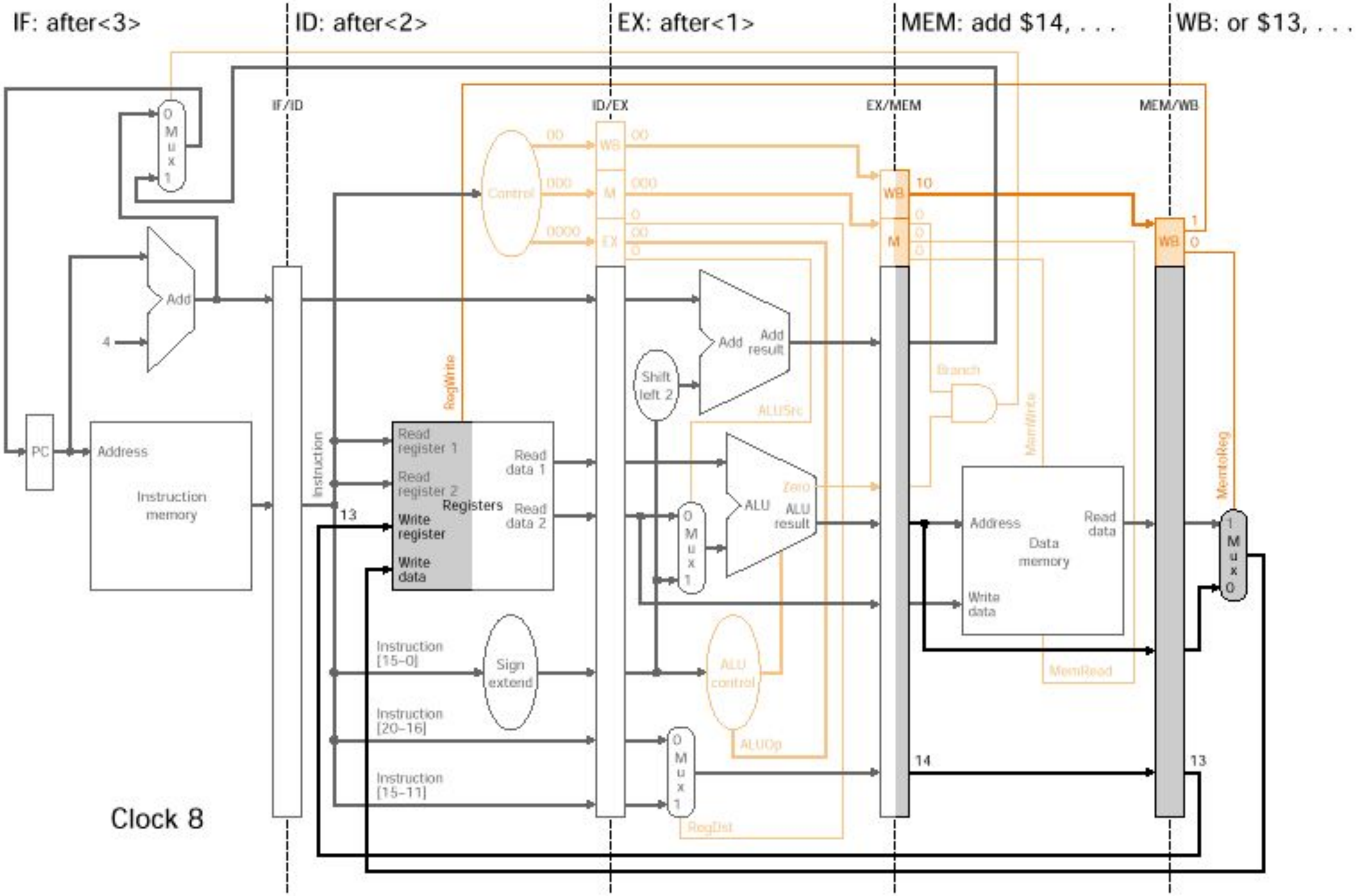
The Grand Example



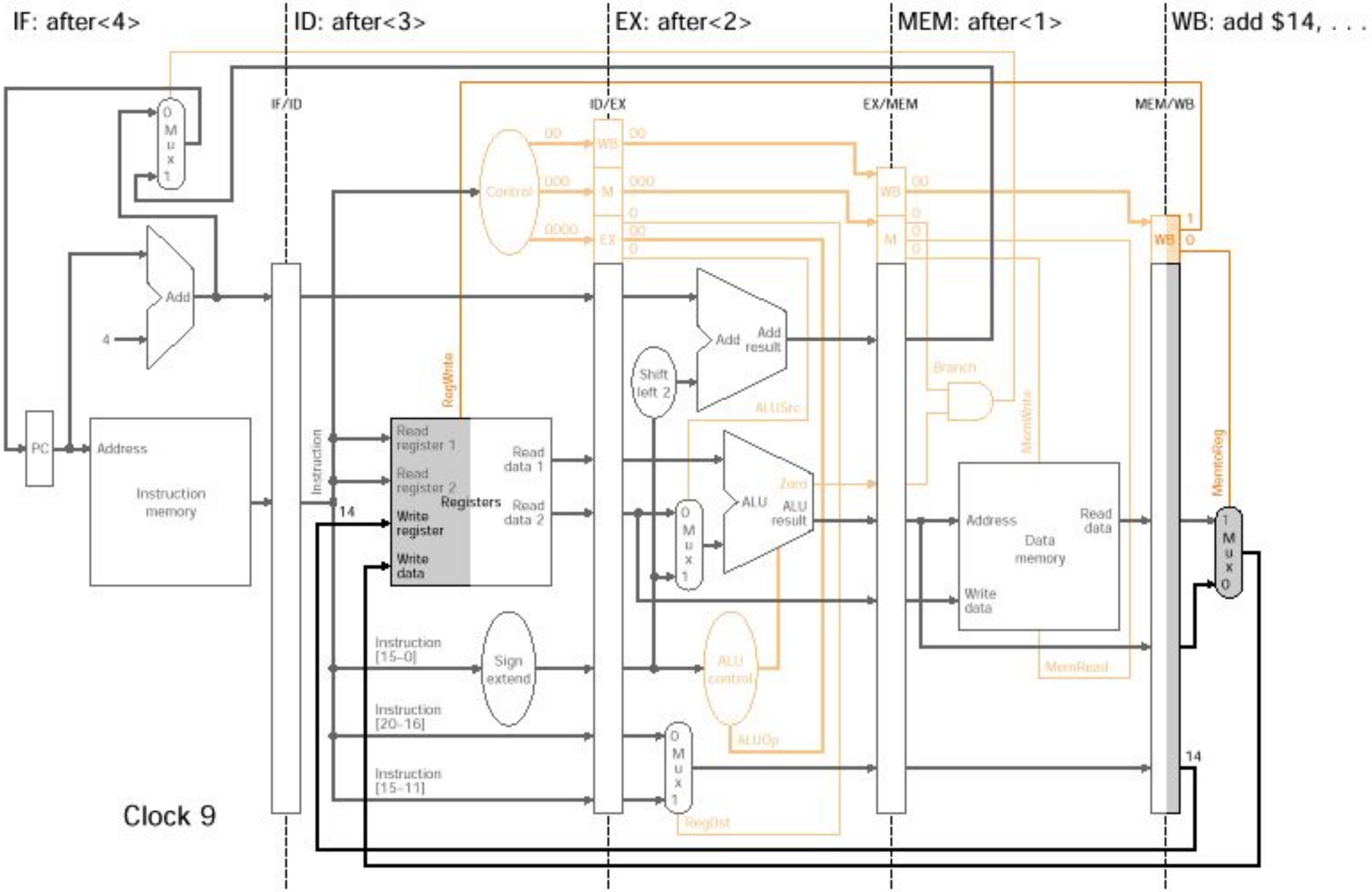
The Grand Example



The Grand Example



The Grand Example


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