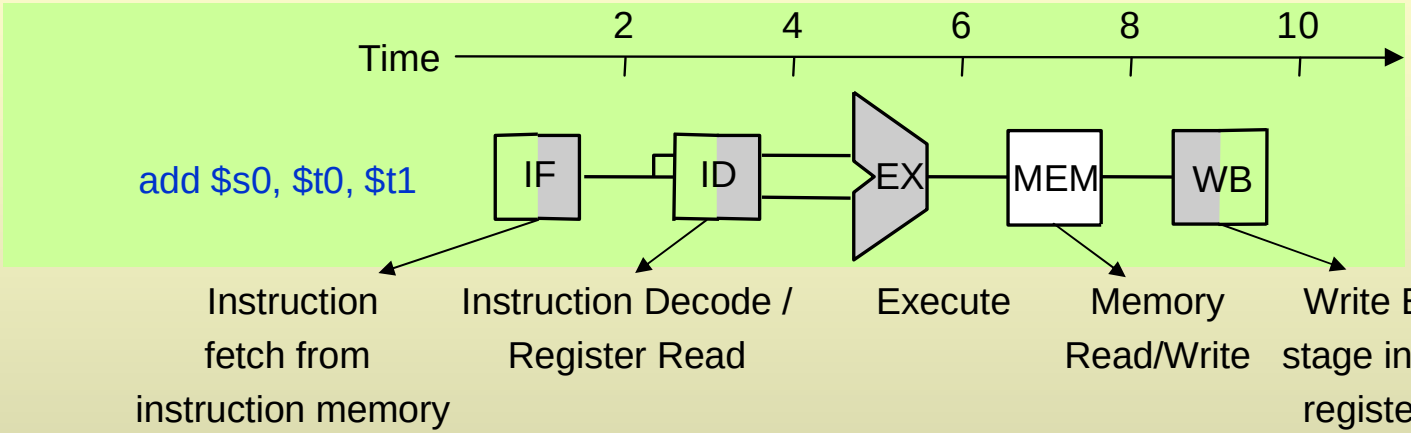
CSE 2021
COMPUTER ORGANIZATION

HUGH CHESSER,
CSEB 1012U

W12-M

Graphical Representation

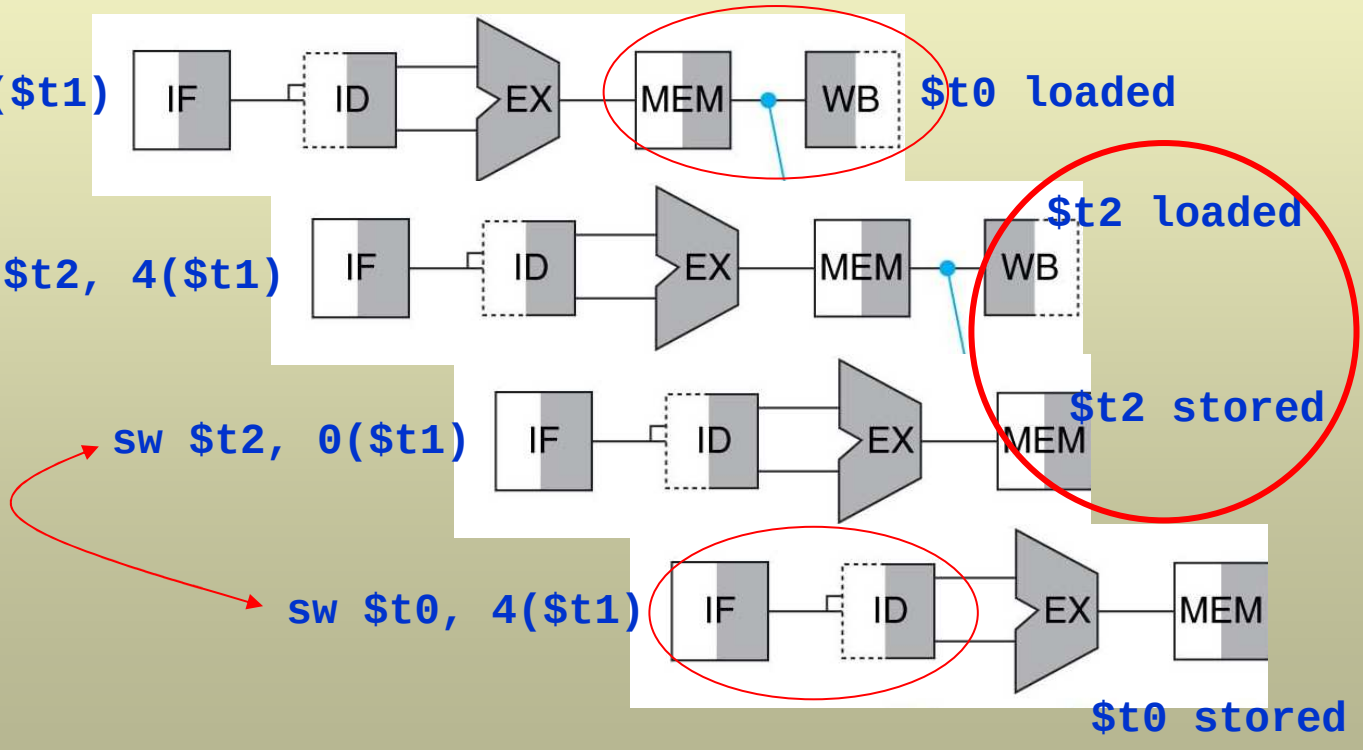


1. Shading in each block indicates the element is used for in the instruction. Since memory is not accessed in an add instruction, it is not shaded.
2. Shading on the left half of the block indicates that the element is being written. During instruction fetch, the instruction memory is read so the right half of IF block is shaded.
3. Shading on the right half of the block indicates that the element is being read. During write back stage, the register file is written so the left half of the WB block is shaded.

Activity 2

Using the graphical representation, show that the following swap procedure has a pipeline hazard. Reorder the instructions to avoid pipeline stalls.

```
lw $t0, 0($t1)
lw $t2, 4($t1)
sw $t2, 0($t1)
sw $t0, 4($t1)
```



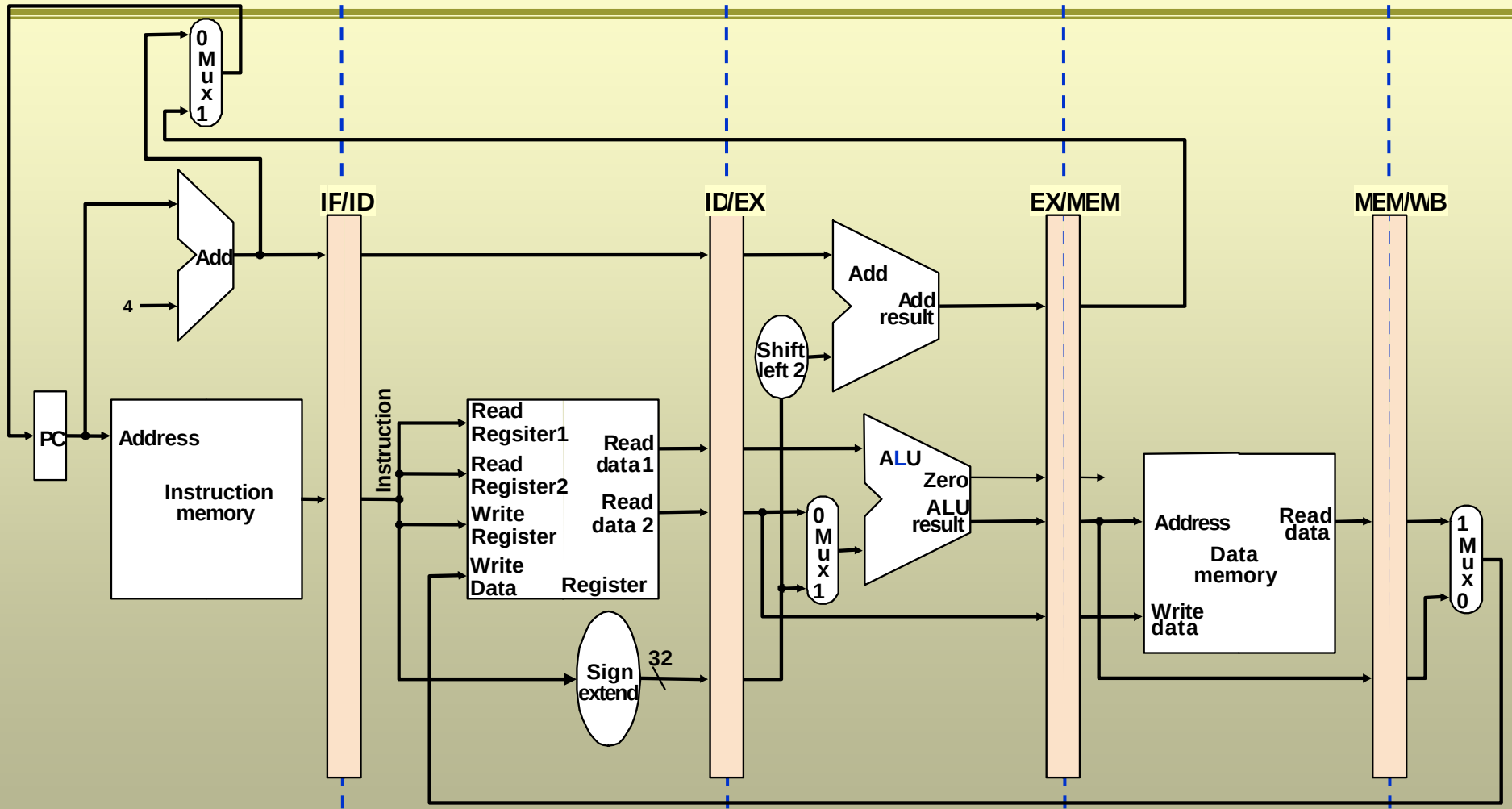
Agenda

Topics:

1. Pipeline Datapath and Control

Patterson: 4.5

Pipelined Datapath (1)



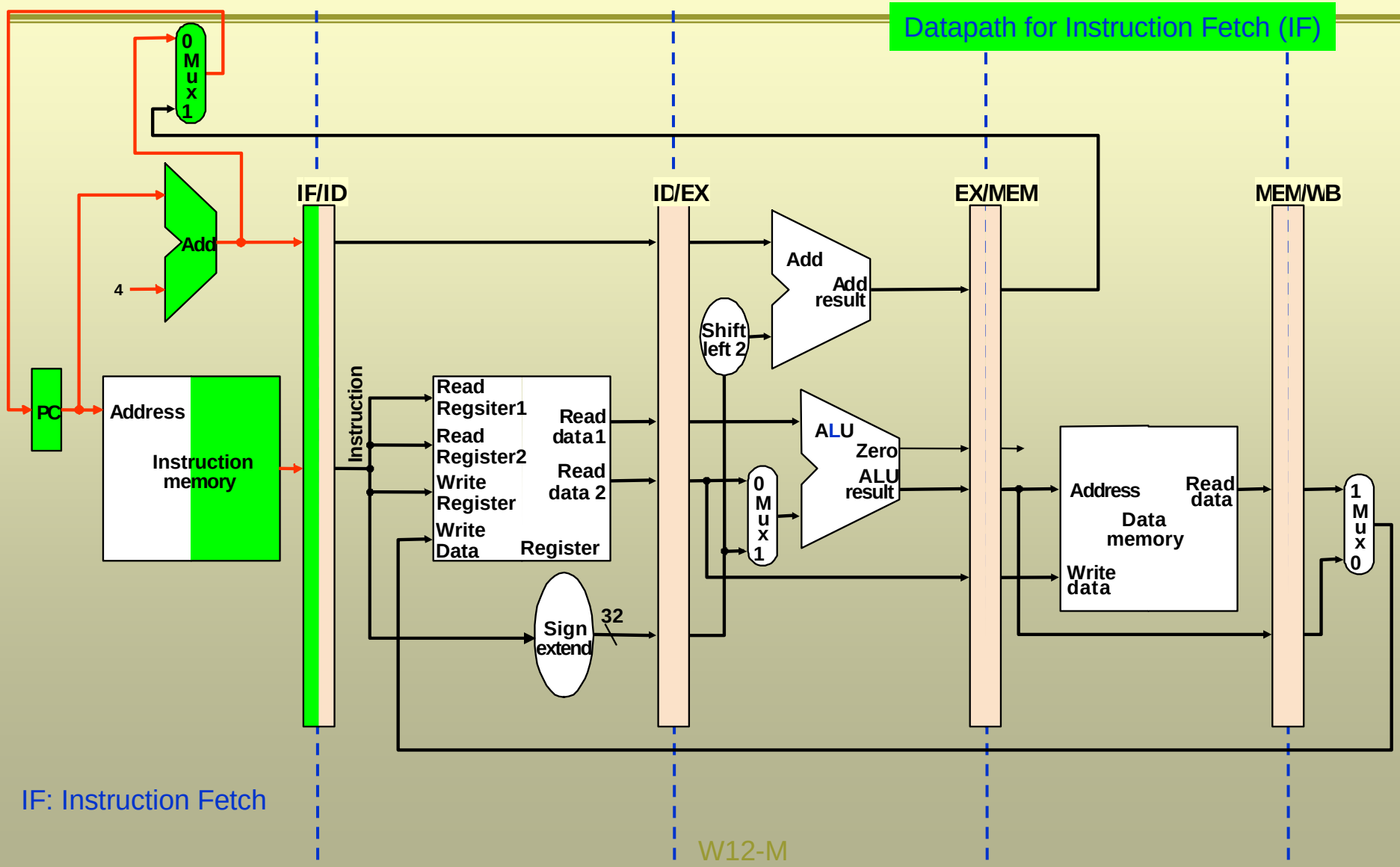
IF: Instruction Fetch | ID: Instruction Decode / Register file read | EX: Execute / Address Calculation | MEM: Memory Access | WB: Write back

Pipelined Datapath (2)

- In pipelined datapath, each instruction is broken in five steps:

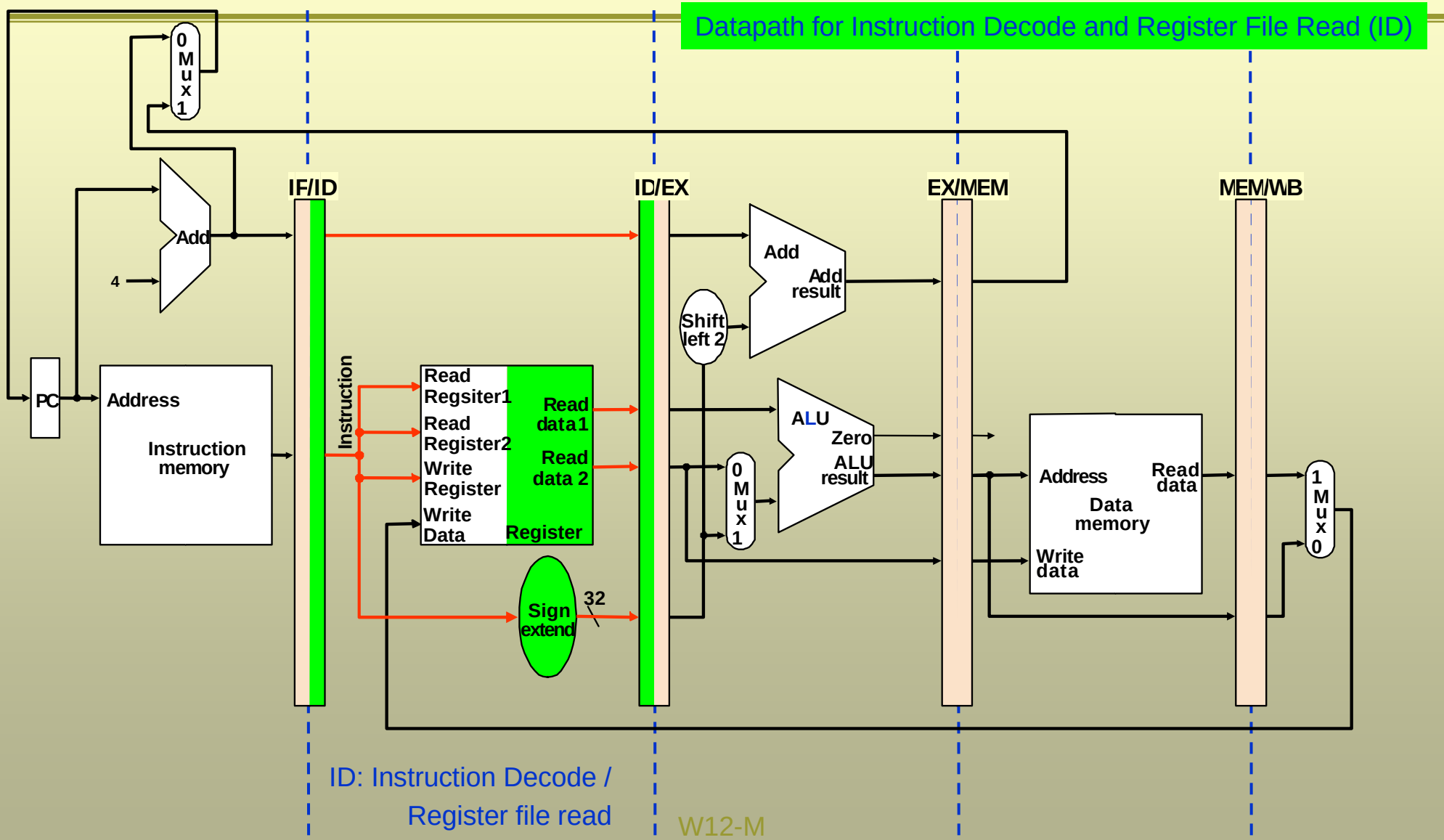
  IF (Instruction Fetch), ID (Instruction Decode and register file read), EX (Execution or address calculation), MEM (Data Memory Access), and WB (Write Back).
- Each of the above step takes one clock cycle.
- Instructions and data advance forward by from left to right.
- Data flows from right to left only in two cases
  1. Write back stage placing the data in the register file
  2. Selection of the value for PC between (PC + 4) and branch target address
- Registers in between different stages store the store values to be used by next stage
- Name of registers are based on the two pipelined stages that the registers separate
- Each pipelining register has a different size: IF/ID register is 64 bits wide; ID/EX register is 128 bits wide; EX/MEM register is 97 bits wide; and MEM/WB is 64 bits wide
- There are no pipeline registers at the end of the write-back stage as data is written directly into memory or register file or the PC.

How pipelining works (1): Example `lw $s1, 0($s2)`



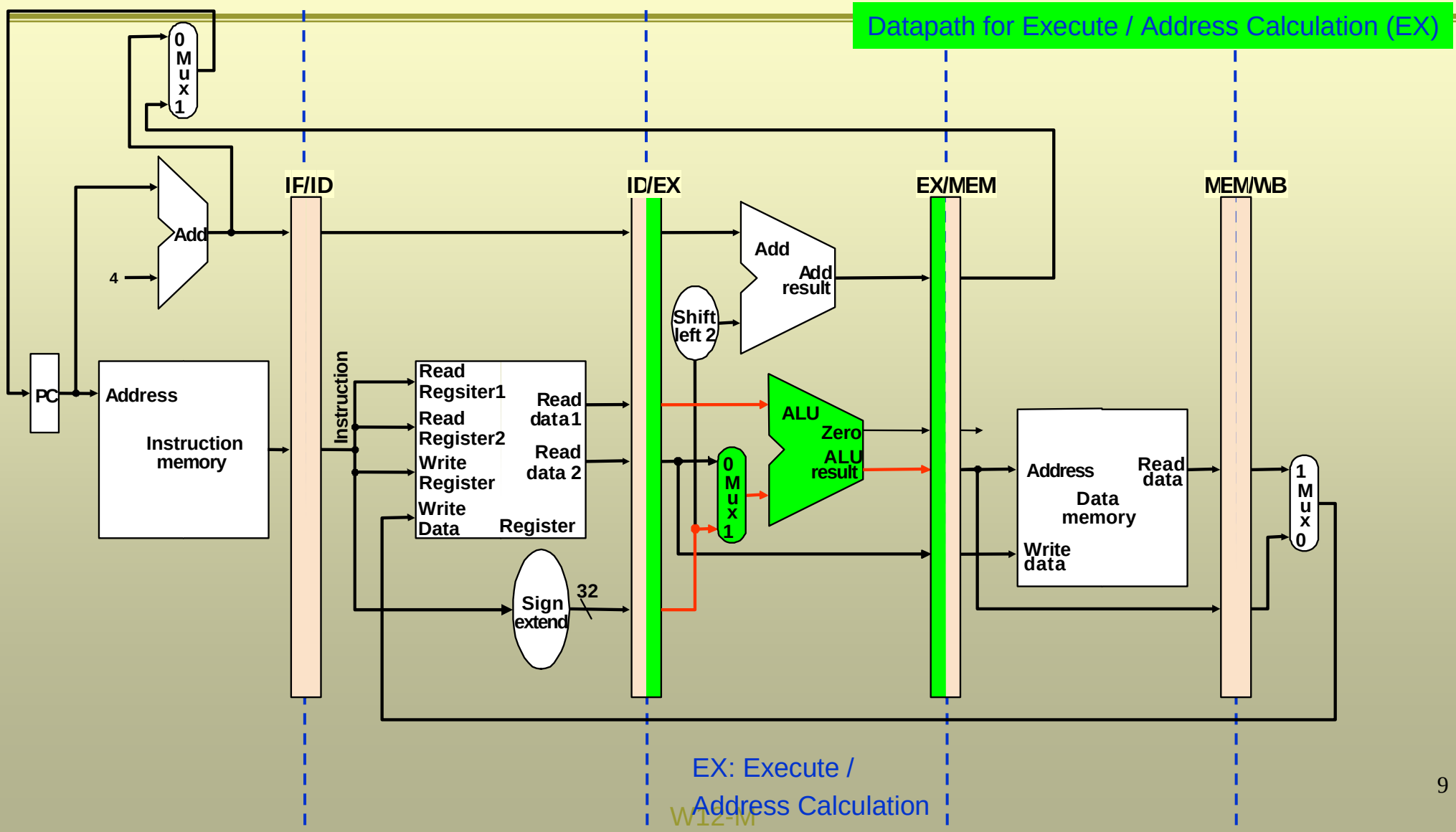
IF: Instruction Fetch

How pipelining works (2): Example `lw $s1, 0($s2)`



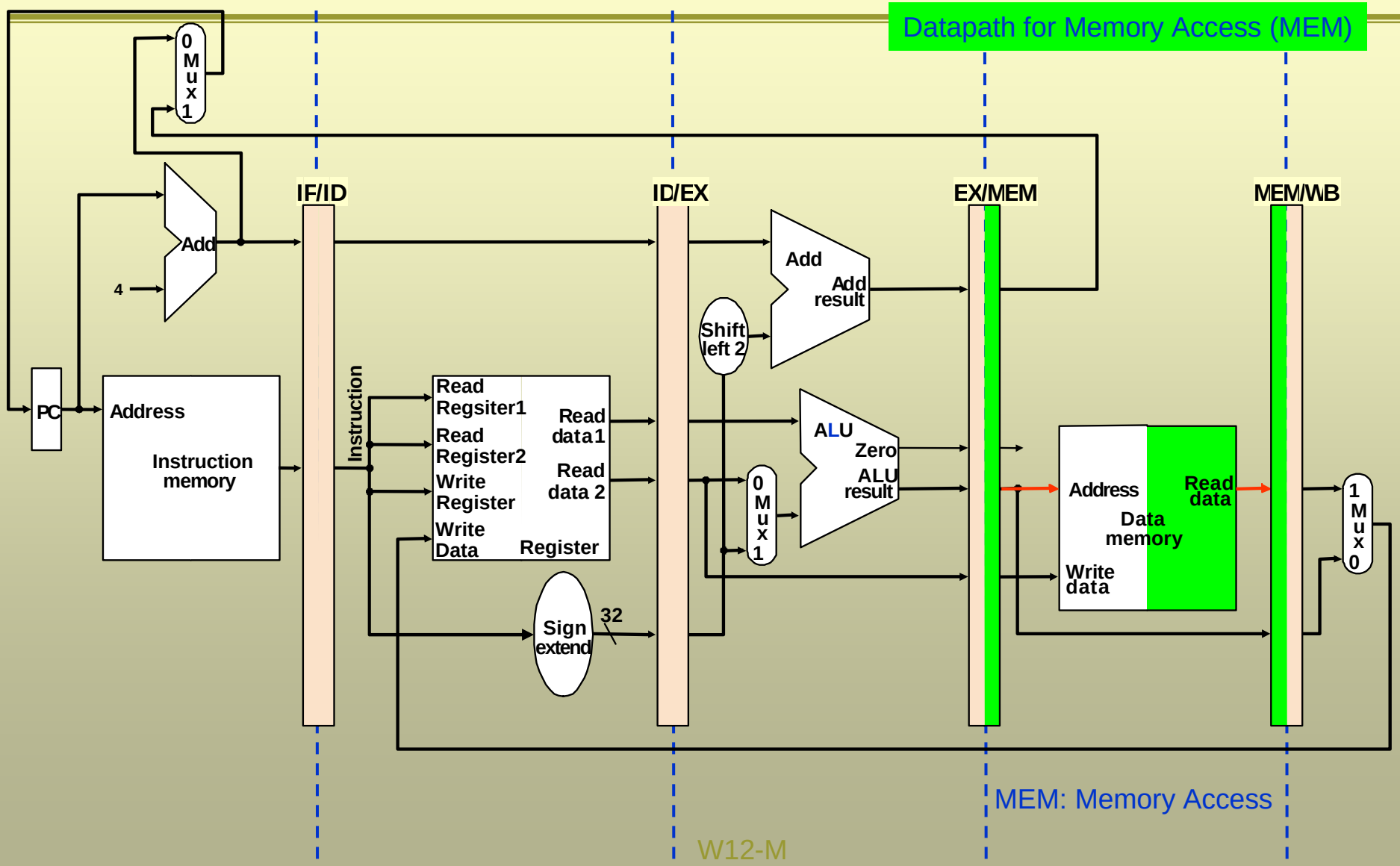
ID: Instruction Decode / Register file read

How pipelining works (3): Example `lw $s1, 0($s2)`

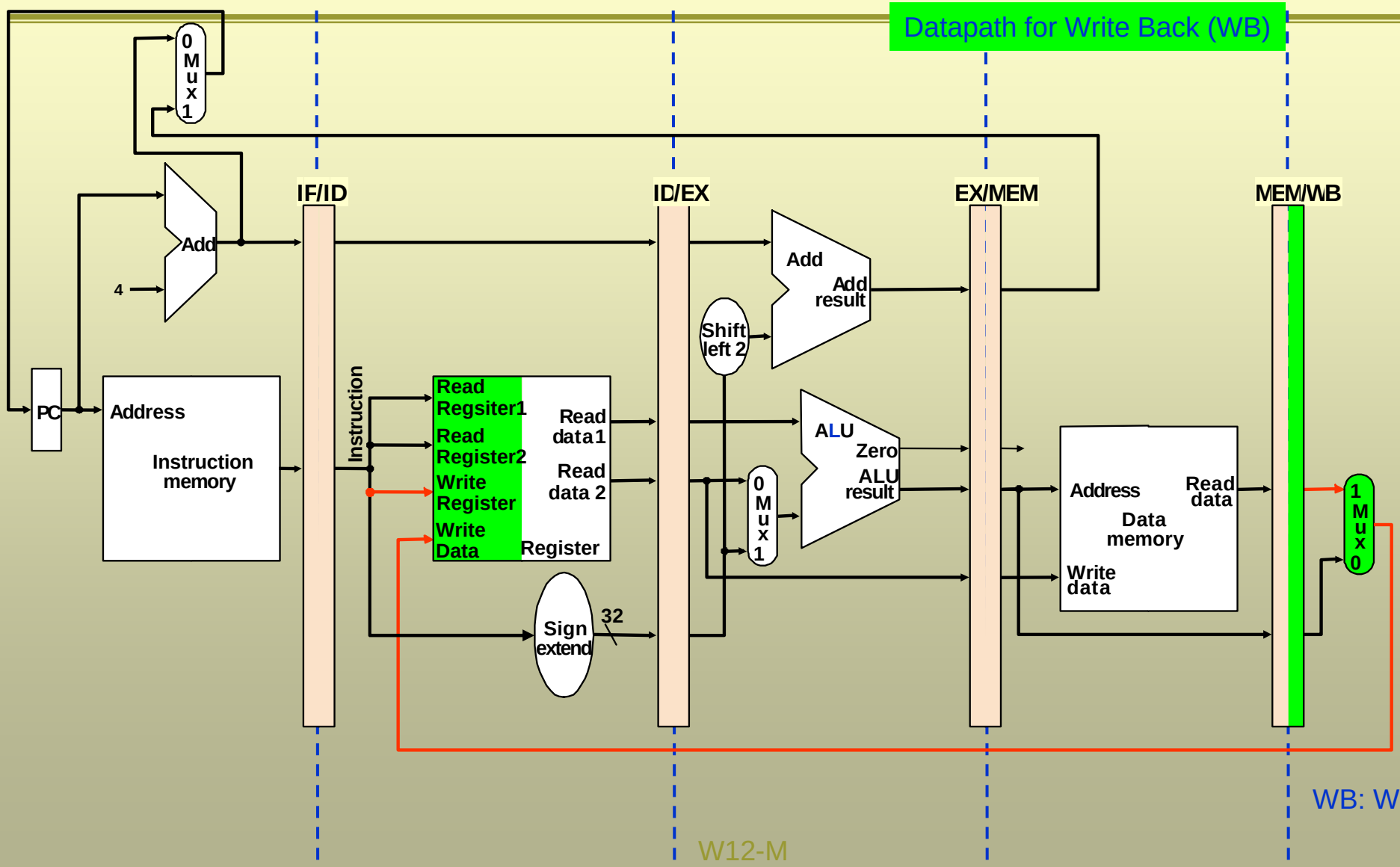


EX: Execute / Address Calculation

How pipelining works (4): Example `lw $s1, 0($s2)`

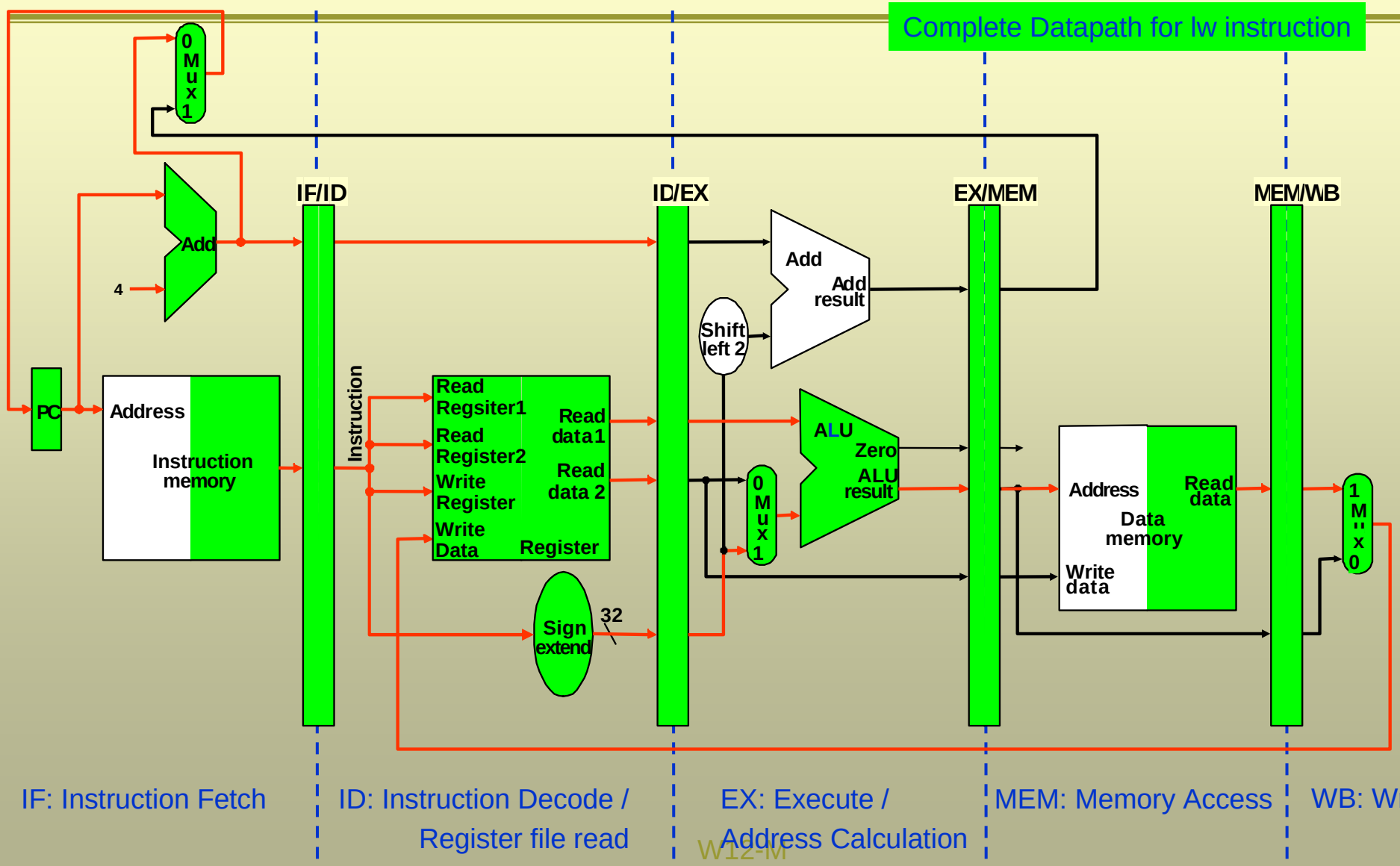


How pipelining works (5): Example `lw $s1, 0($s2)`



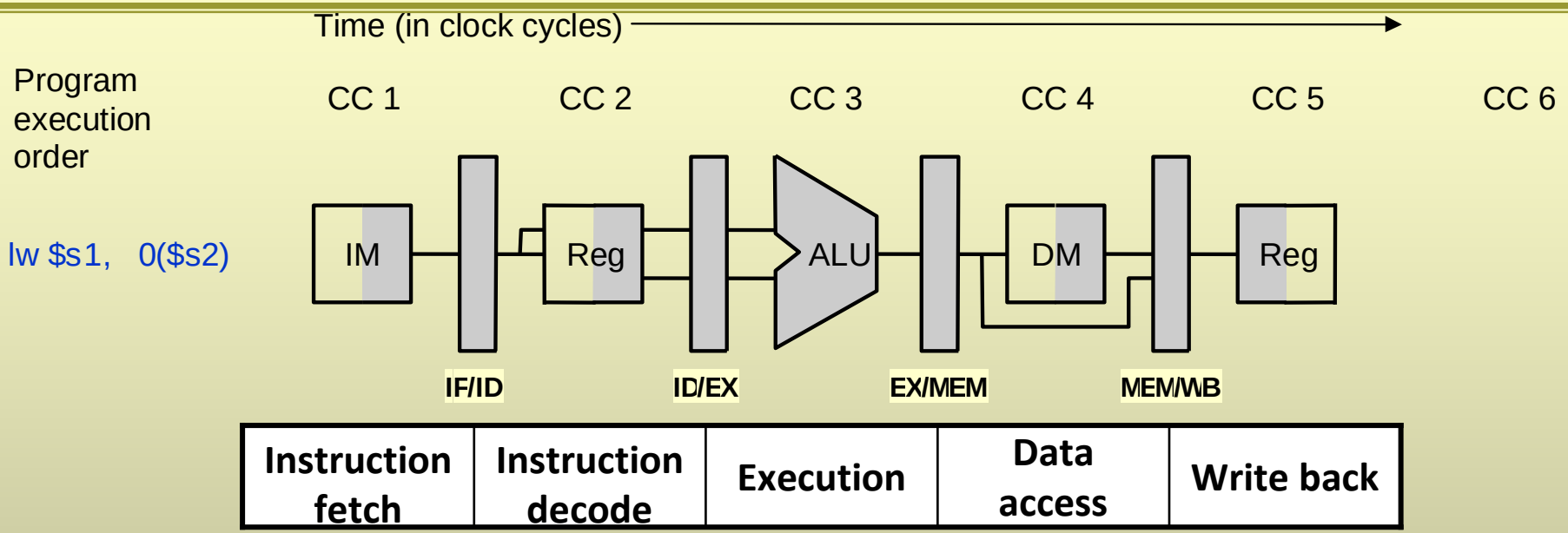
WB: Write back

How pipelining works (6): Example `lw $s1, 0($s2)`



IF: Instruction Fetch | ID: Instruction Decode / Register file read | EX: Execute / Address Calculation | MEM: Memory Access | WB: Write back

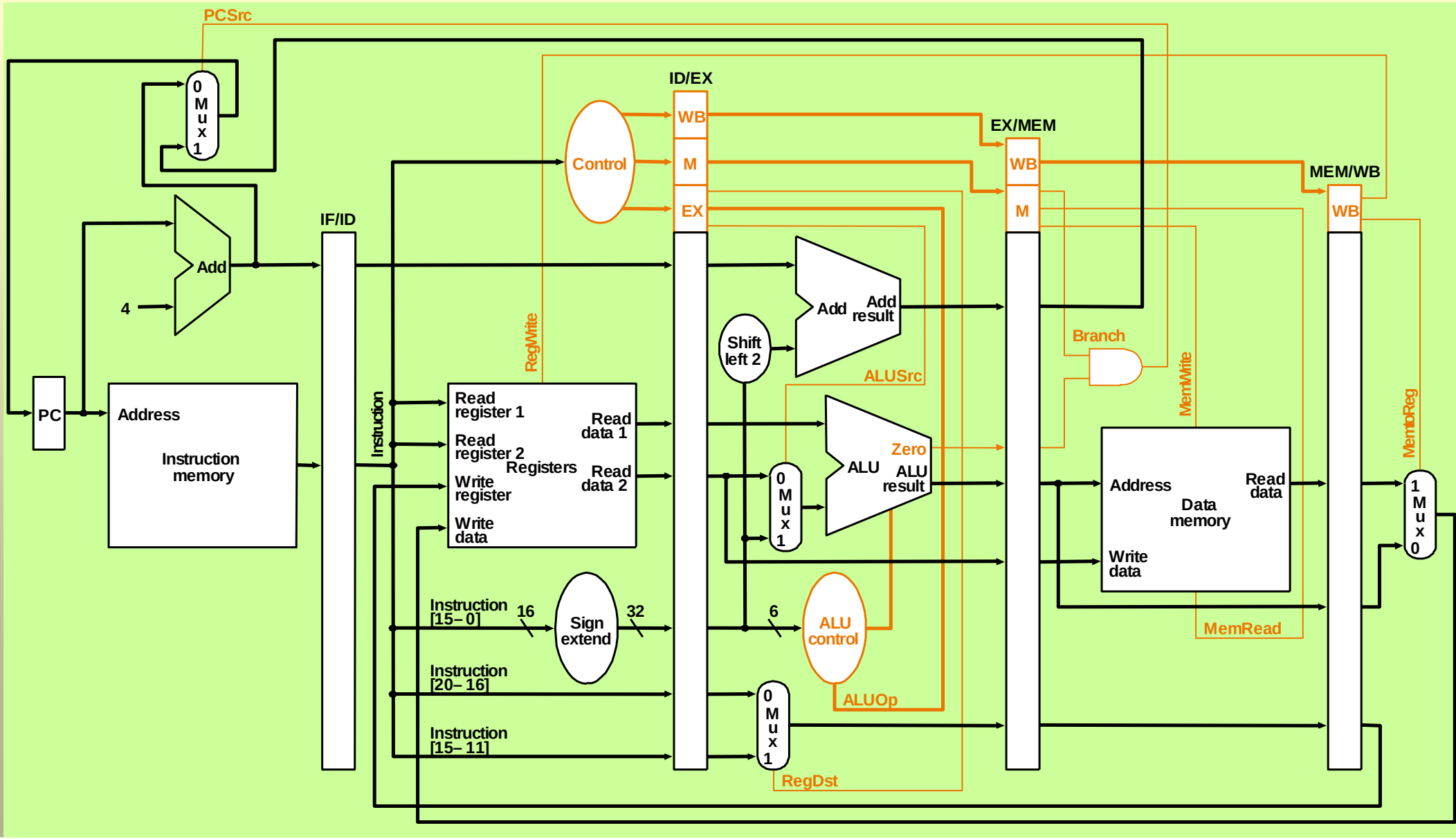
Multiple Clock Cycle Pipeline Diagram



Activity 3: Using the graphical representation, show that the multiple clock cycle pipeline diagram of the following two instructions

```
lw $t0,0($t1)
sub $s0,$s1,$s2
```

Pipelined Control (1)



Pipelined Control (2)

— Control lines in pipelined implementation is 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.

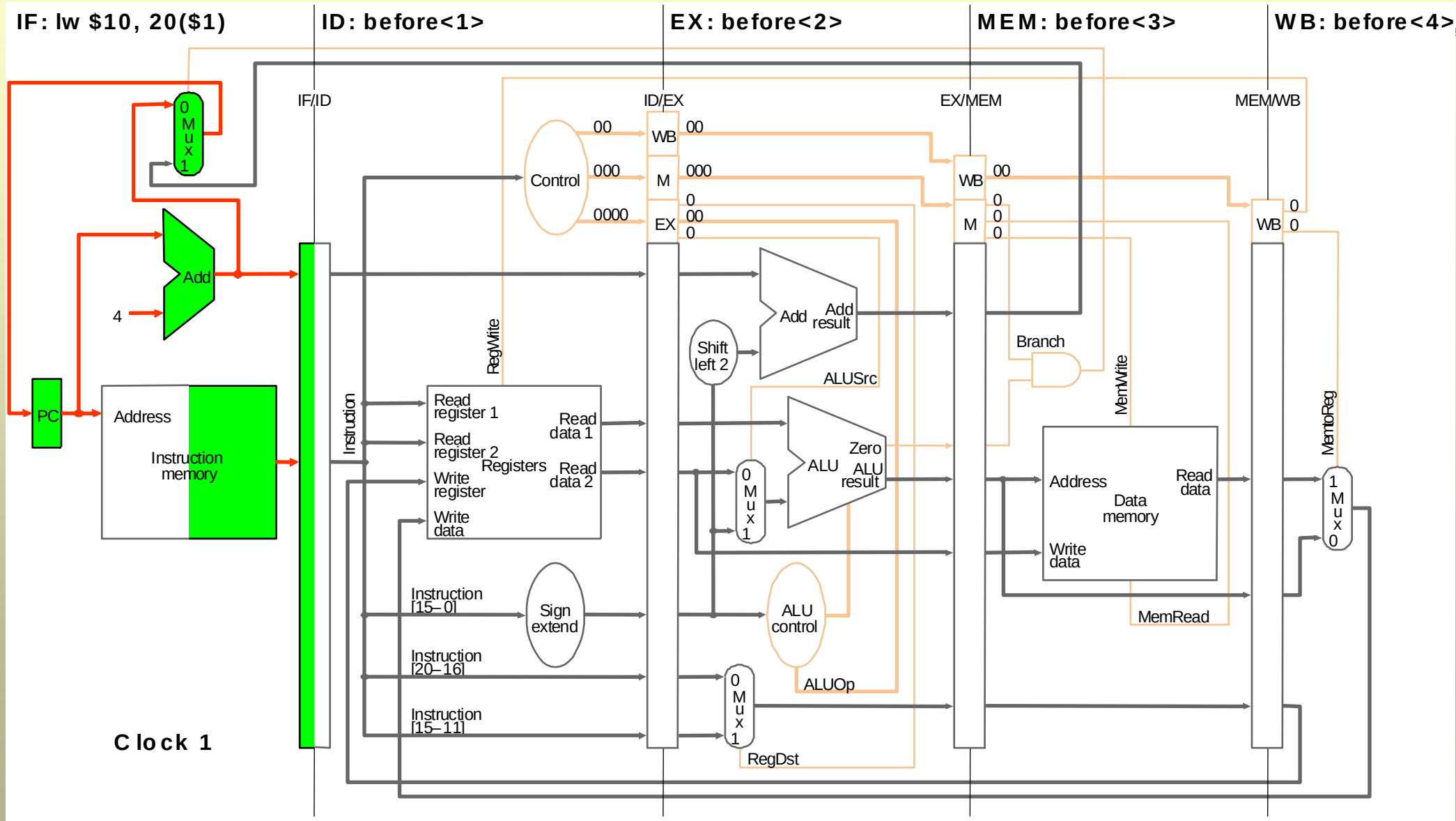
— Pipeline registers are extended to include the control signals for each stage of an instruction.

Activity 4

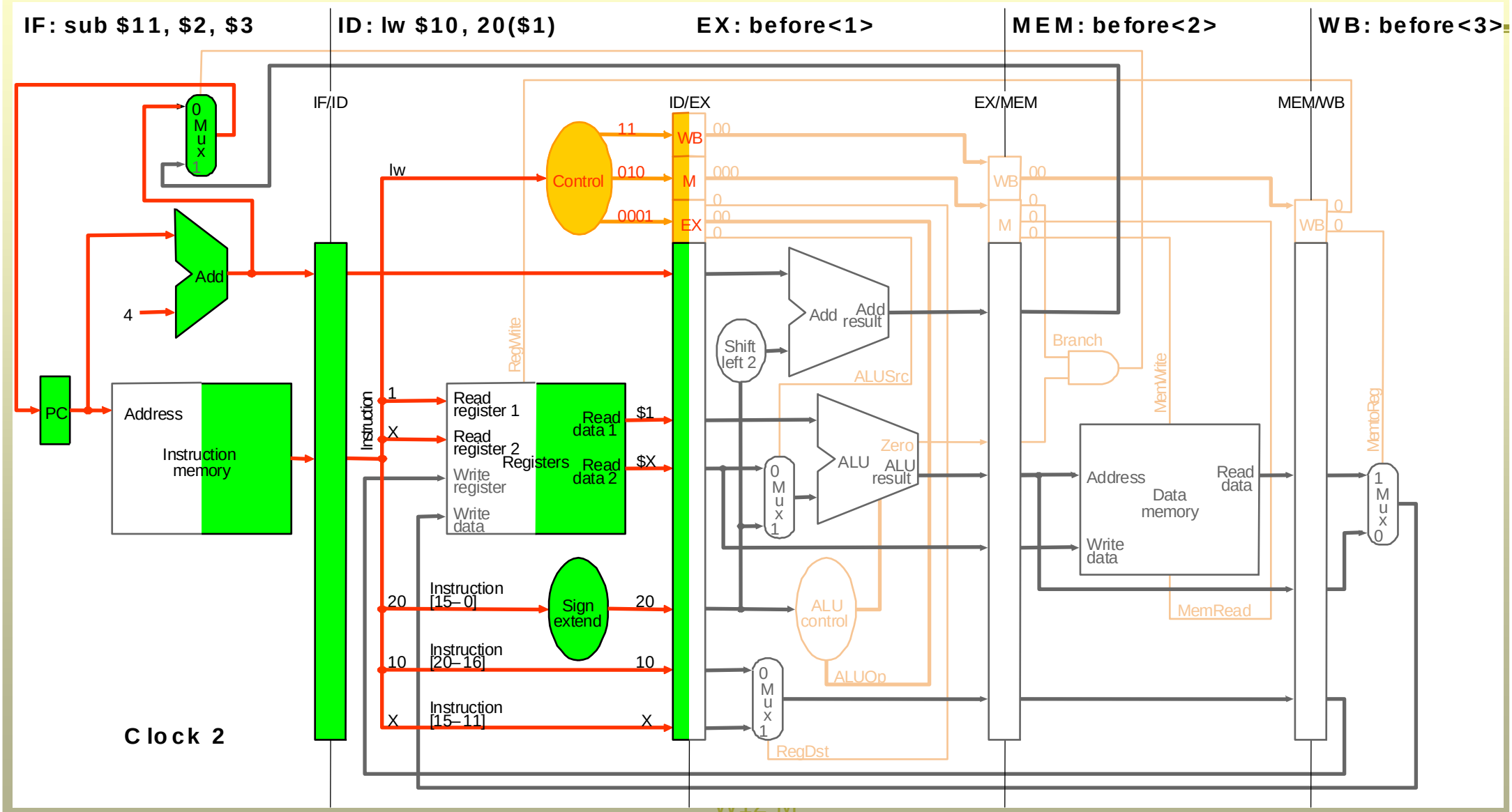
Show the following instructions going through the pipeline:

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

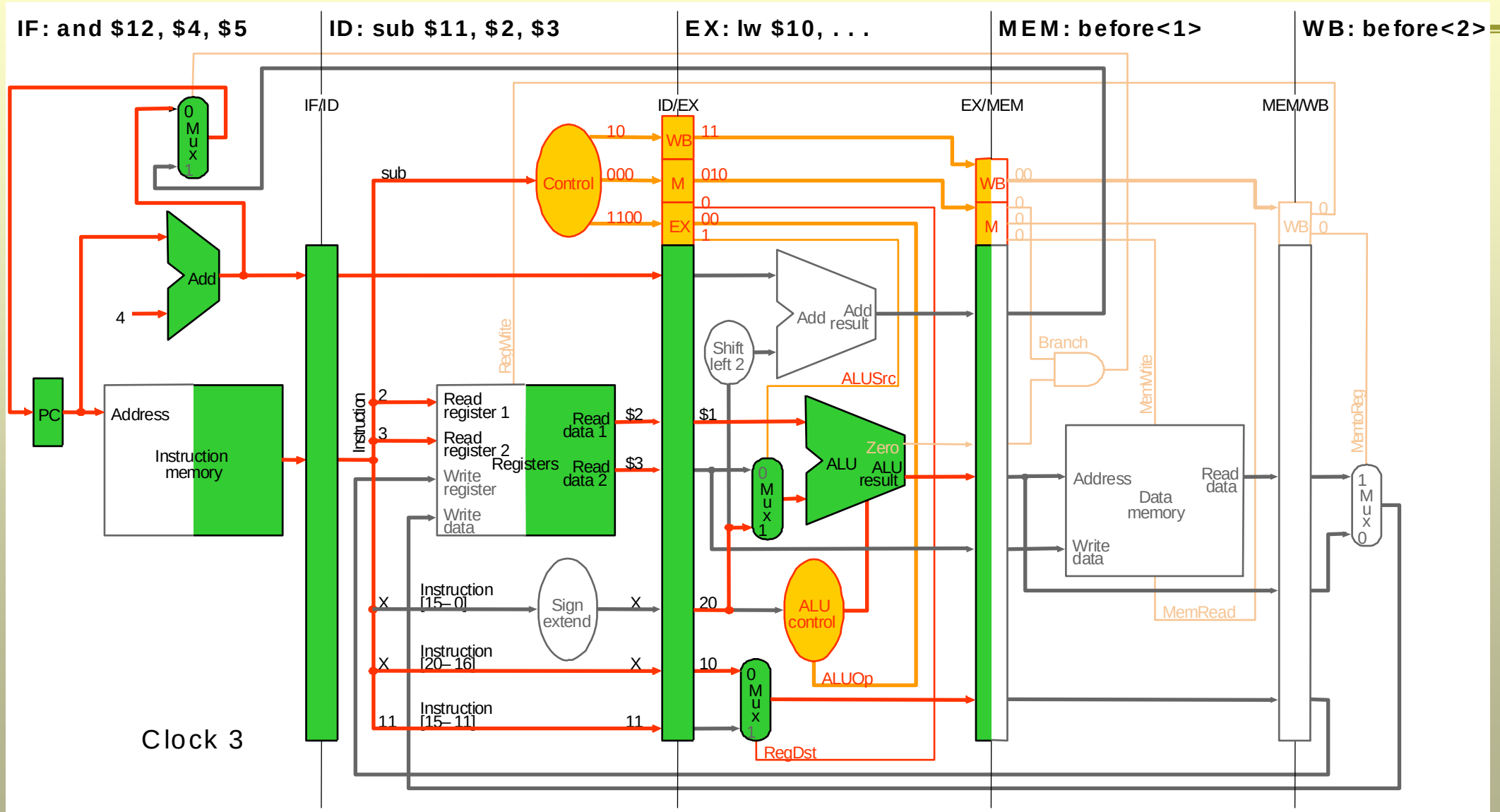
Activity 4: Clock Cycle # 1



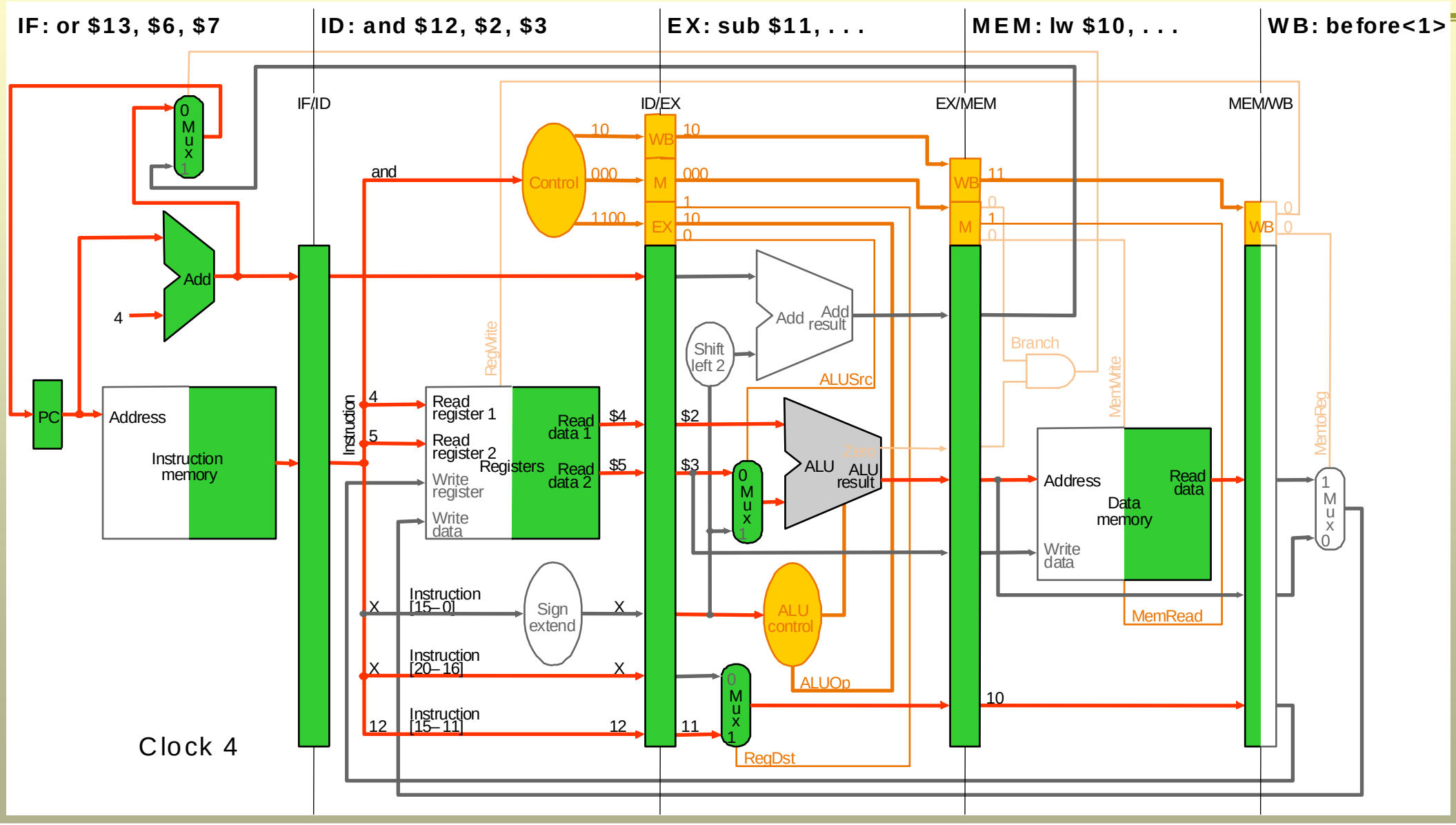
Activity 4: Clock Cycle # 2



Activity 4: Clock Cycle # 3



Activity 4: Clock Cycle # 4



Activity 4: Clock Cycle # 5