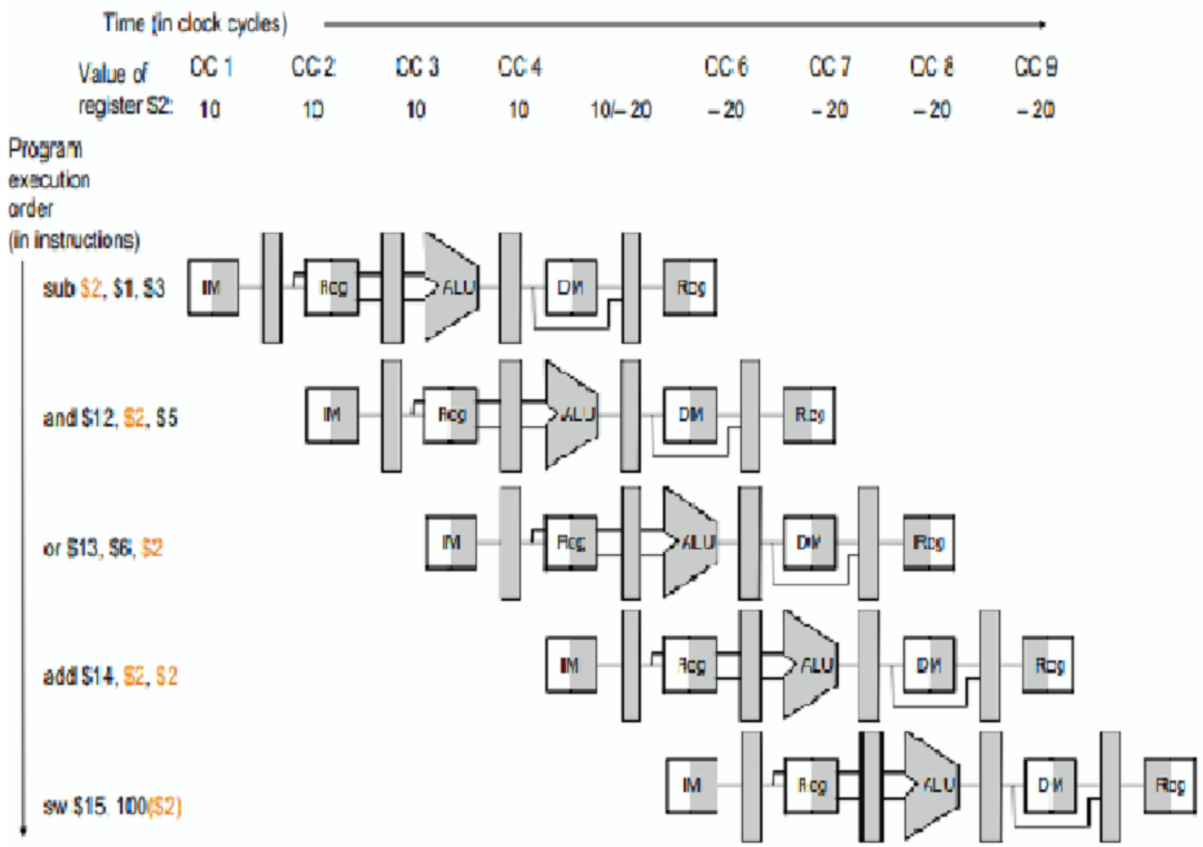
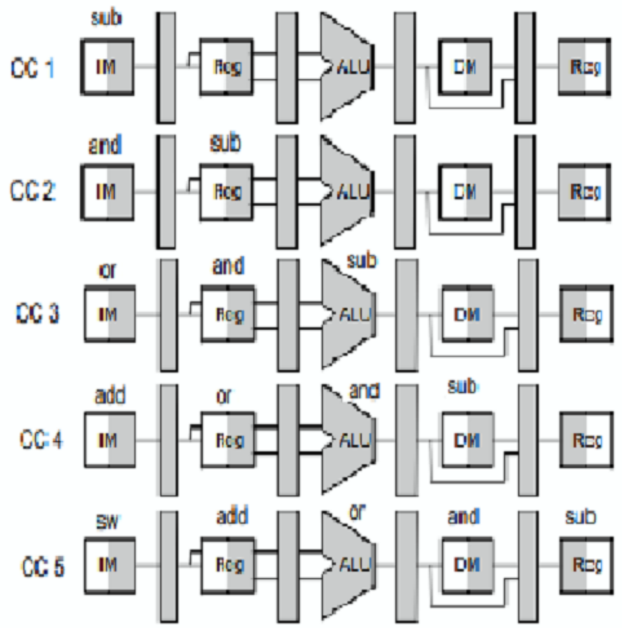
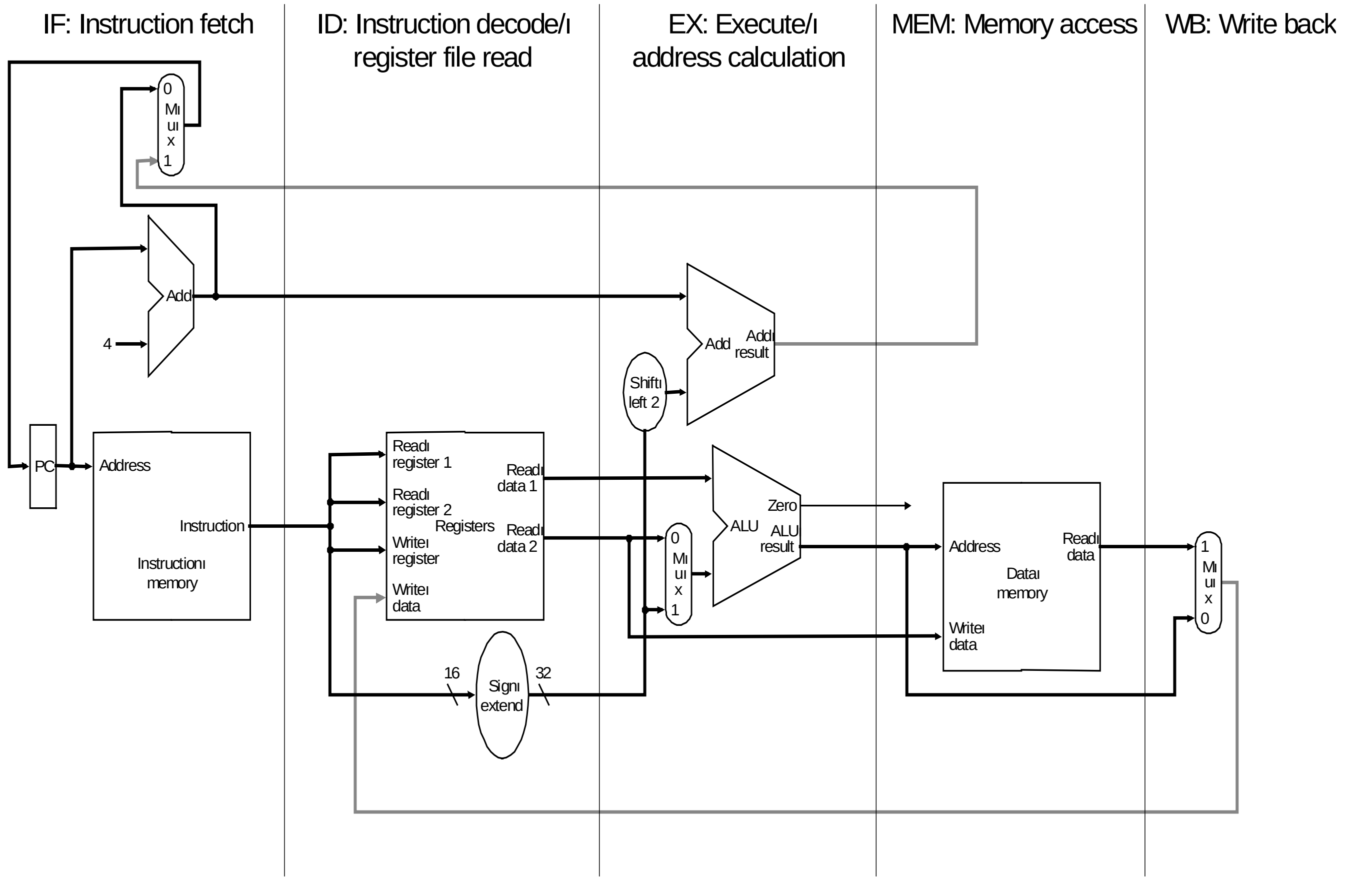


Time (in clock cycles)
Value of register $2:
CC 1 10
CC 2 10
CC 3 10
CC 4 10
10/–20
CC 6 –20
CC 7 –20
CC 8 –20
CC 9 –20
Program execution order (in instructions)
sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
IM
Reg
ALU
DM
Reg



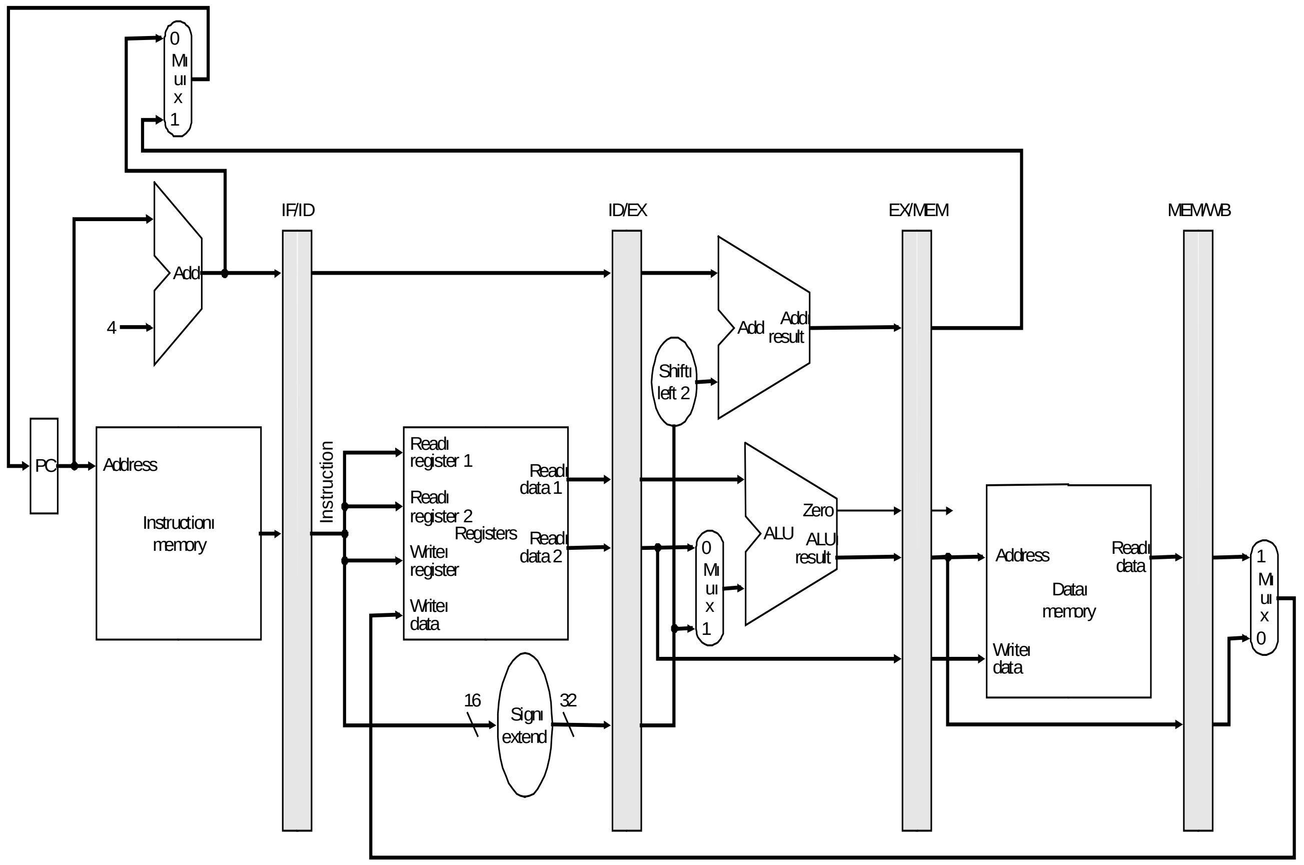
CC 1
sub
CC 2
and
sub
CC 3
or
and
sub
CC 4
add
or
and
sub
CC 5
sw
add
or
and
sub
IM
Reg
ALU
DM
Reg

Implementation of the pipelined datapath



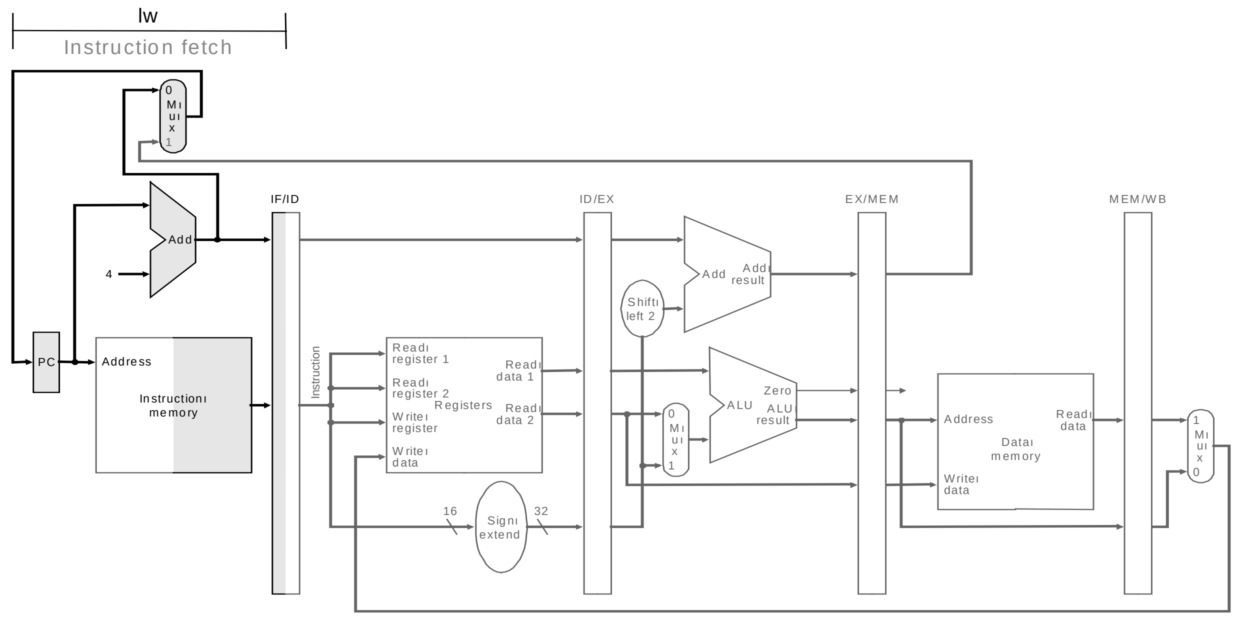
- *What do we need to add to actually split the datapath into stages?*

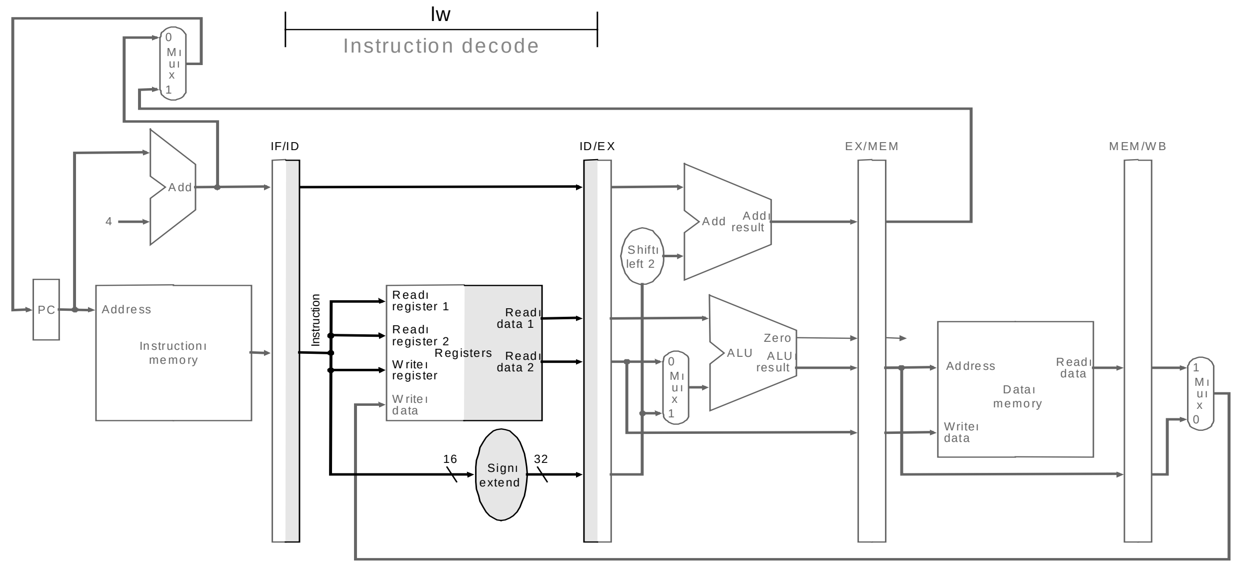
Pipelined Datapath



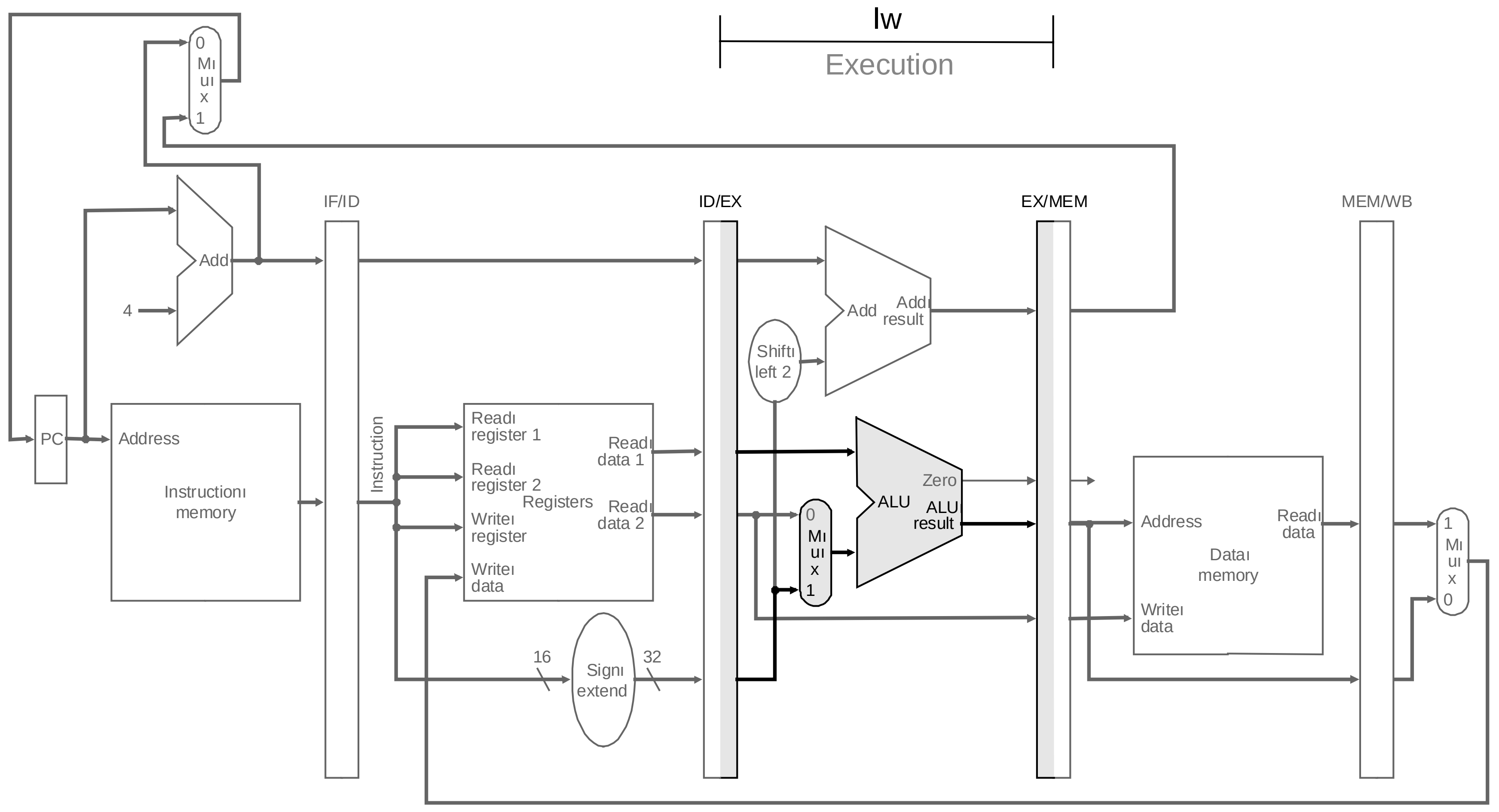
Can you find a problem even if there are no dependencies?
What instructions can we execute to manifest the problem?

Lw: Instruction fetch and Instruction decode

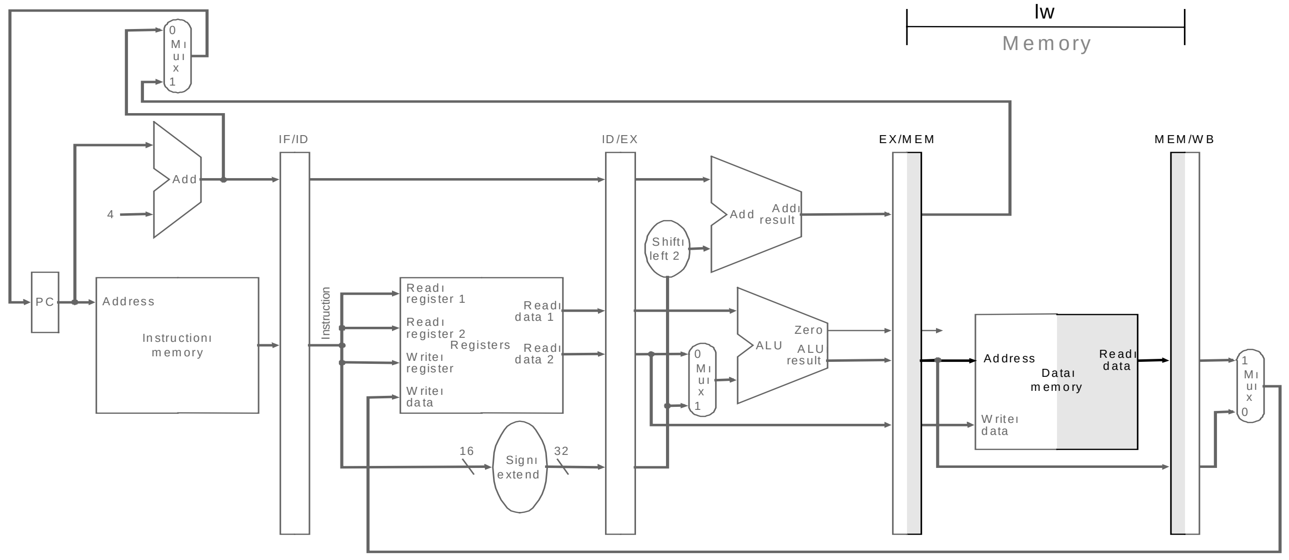


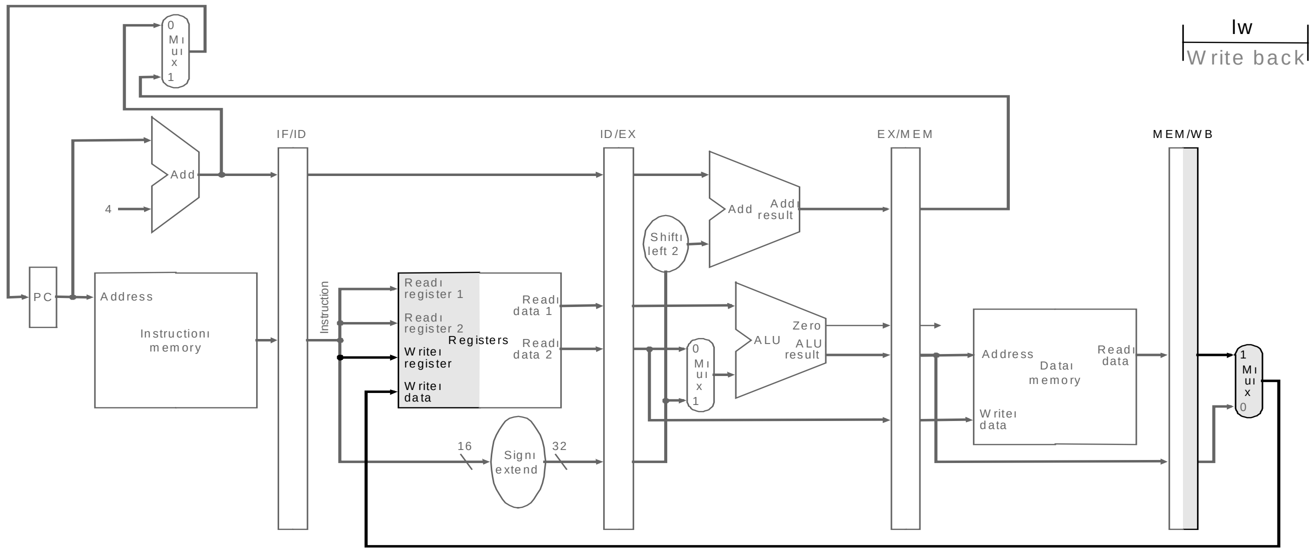


Lw: Execution

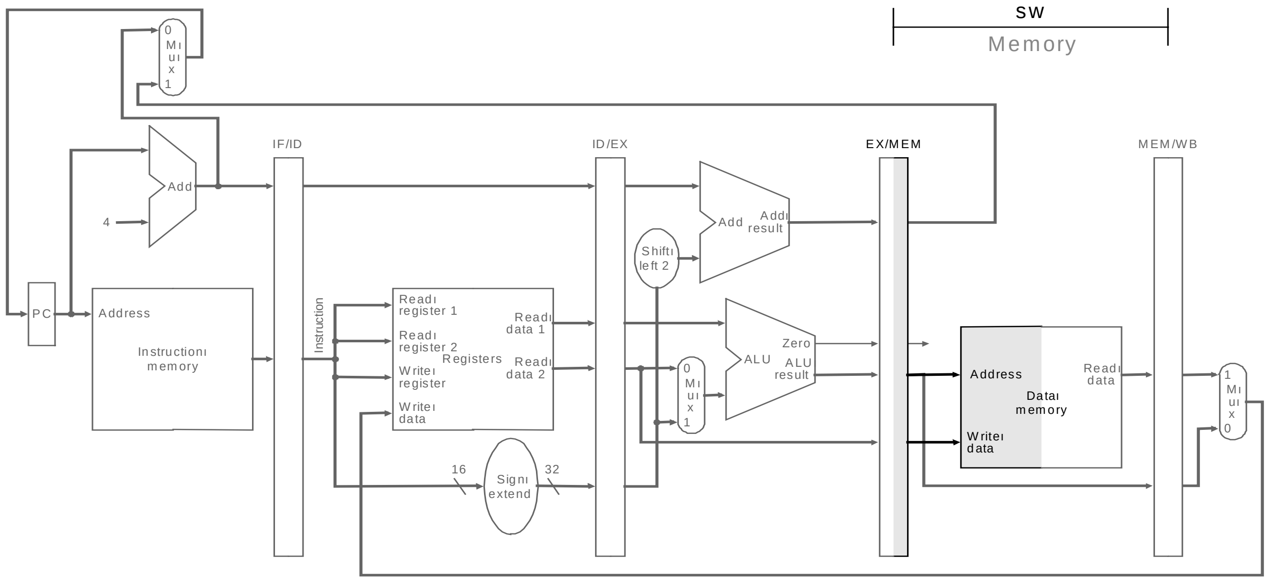


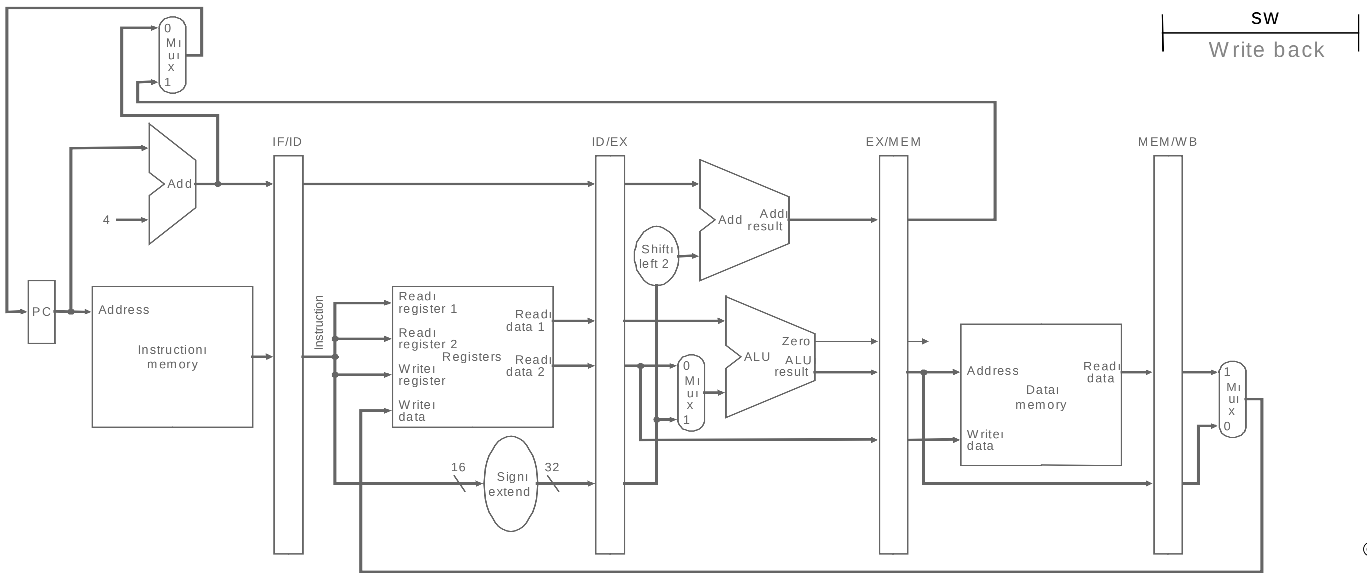
Lw: Memory and Write back



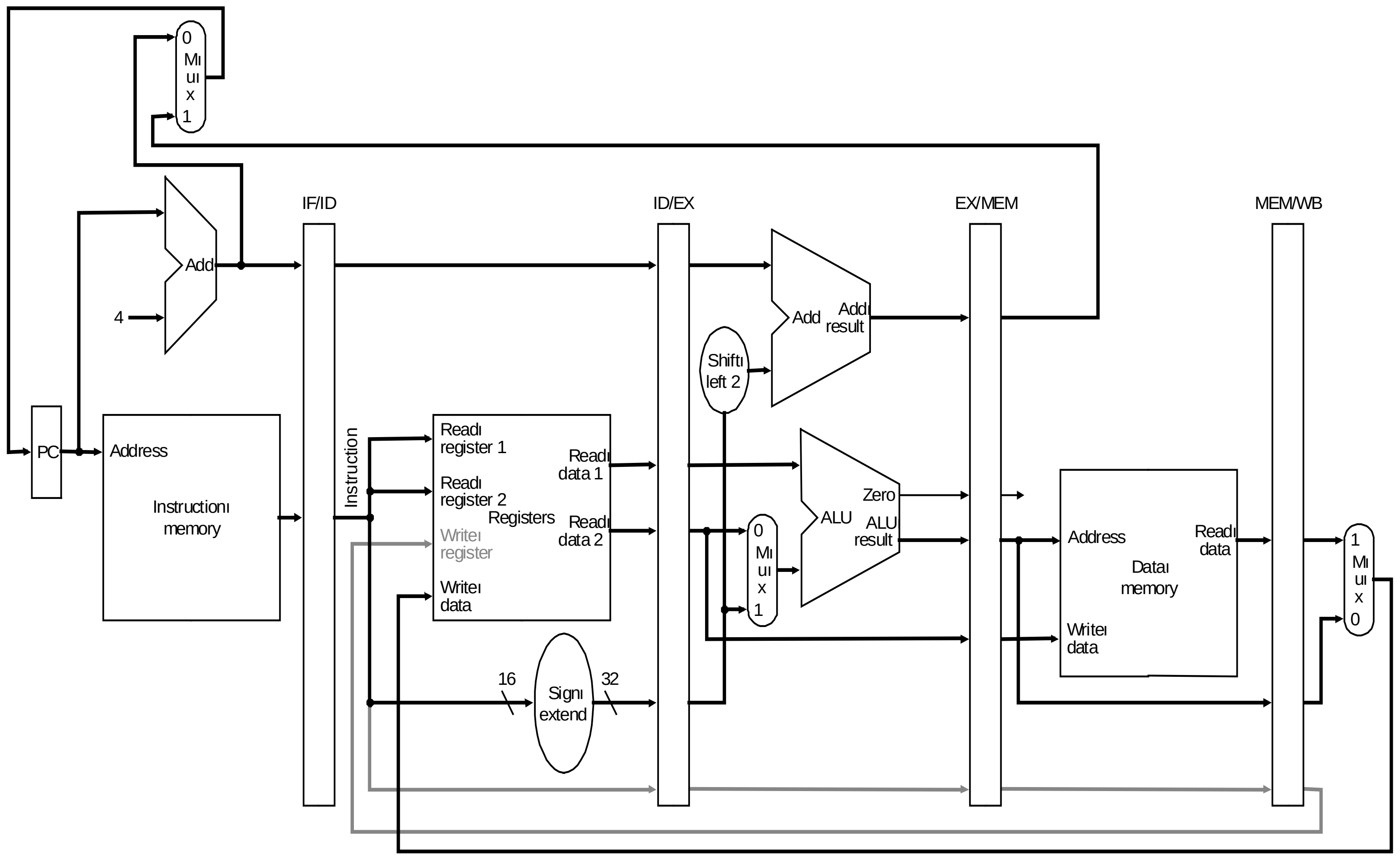


Sw: Memory and Write back

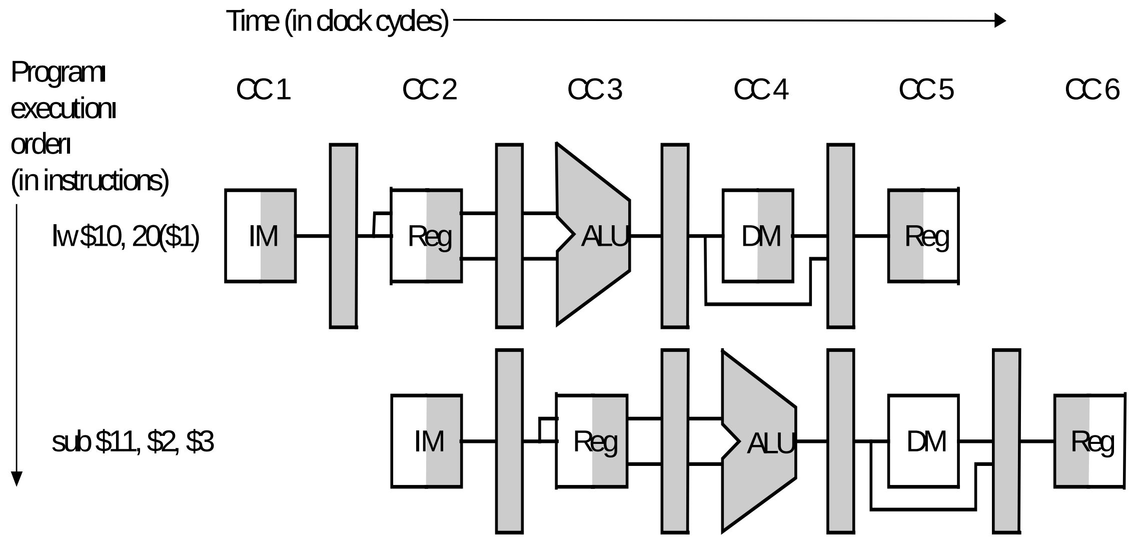




Corrected Datapath

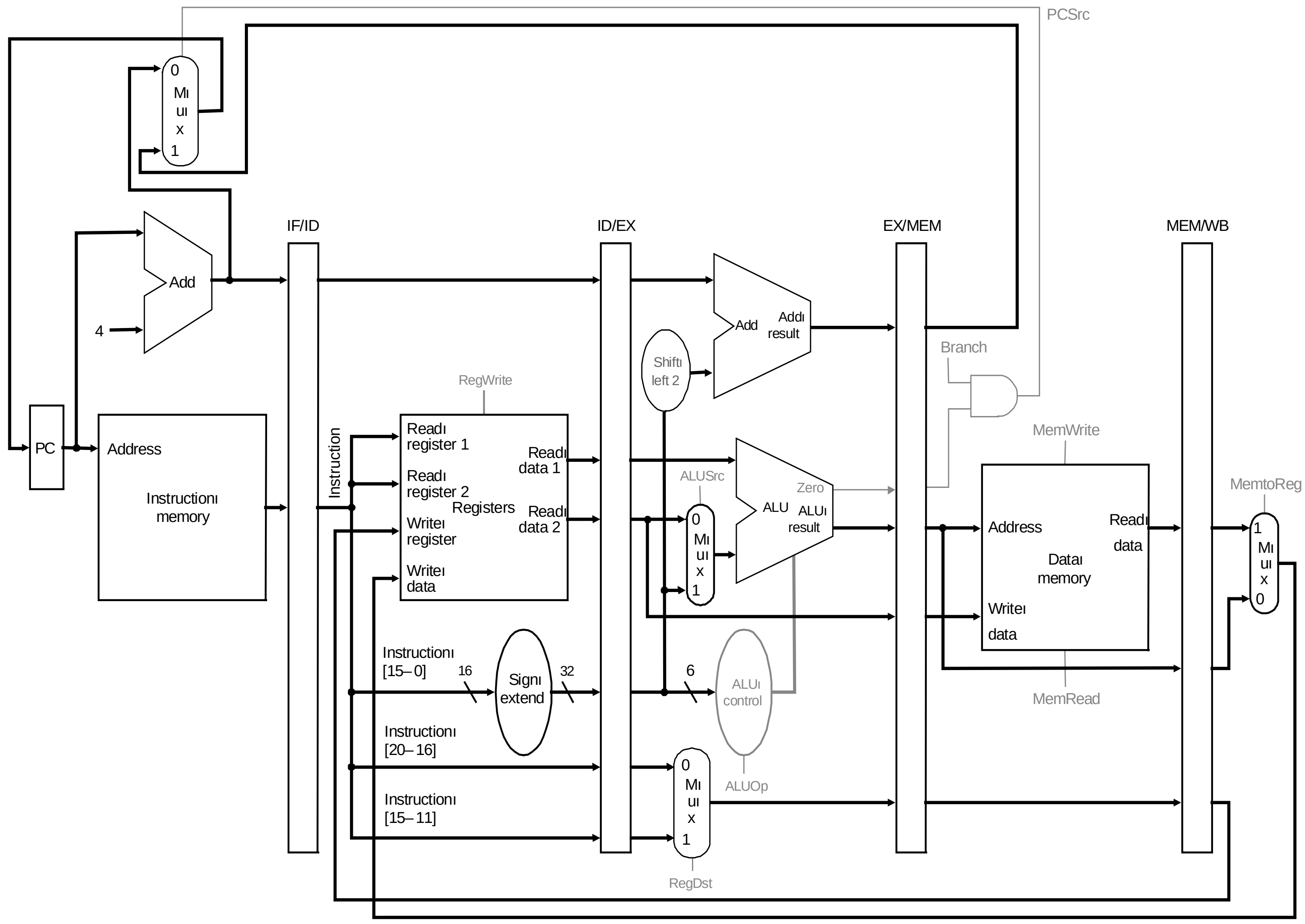


Graphically Representing Pipelines



- **Multiple-clock-cycle diagram vs. Single-clock-cycle diagram**

Pipeline Control



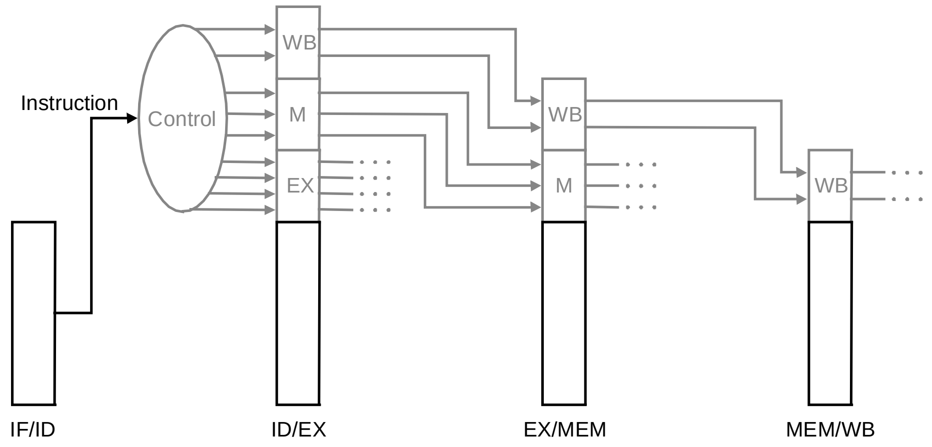
Pipeline control

- We have 5 stages. What needs to be controlled in each stage?
  - Instruction Fetch and PC Increment: the control signals are always asserted (this stage is executed at every clock cycle)
  - Instruction Decode / Register Fetch: the control signals are always asserted (this stage is executed at every clock cycle)
  - Execution: set RegDst, ALUOp, ALUSrc (select the result register, the ALU operation, and either Read data 2 or a sign-extended immediate for the ALU).
  - Memory Stage: Branch, MemRead, MemWrite (set by beq, lw, sw).
  - Write Back: MemtoReg, RegWrite (send the ALU result to the memory or to the register file, write the value).

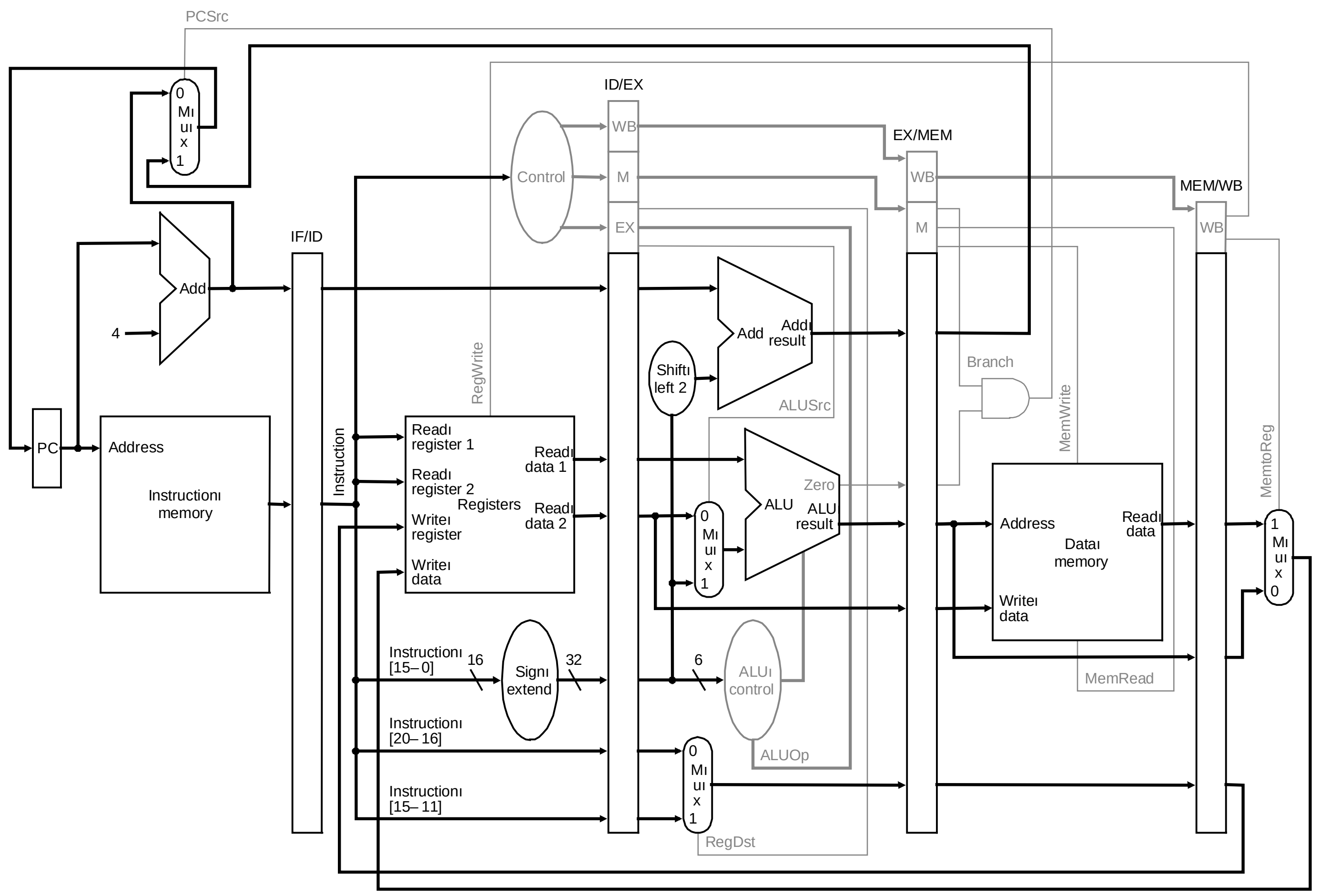
Pipeline Control

- **Pass control signals along just like the data**

	Execution/Address Calculation stage control lines				**Memory access stage control lines**			**stage control lines**	
Instruction	**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



Datapath with Control



Example

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