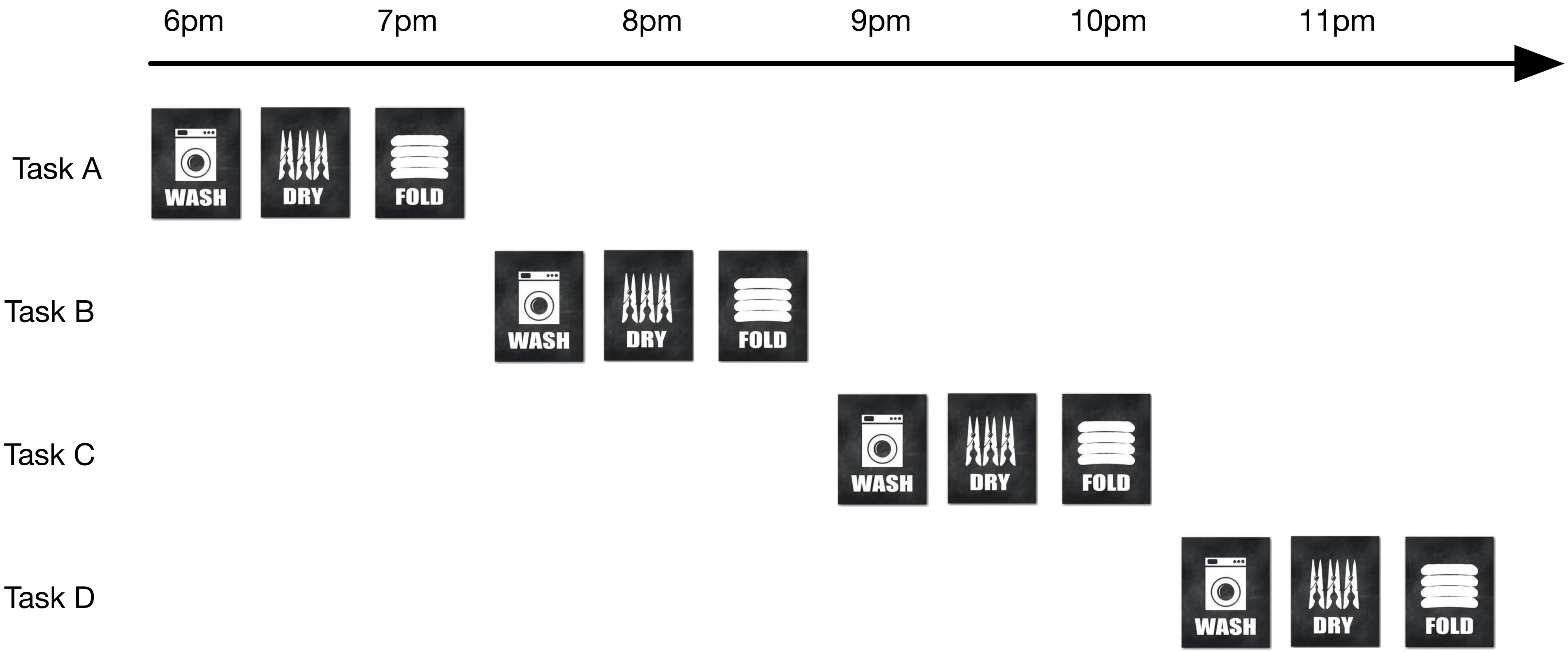
HW4 - Soon!

Pipelining

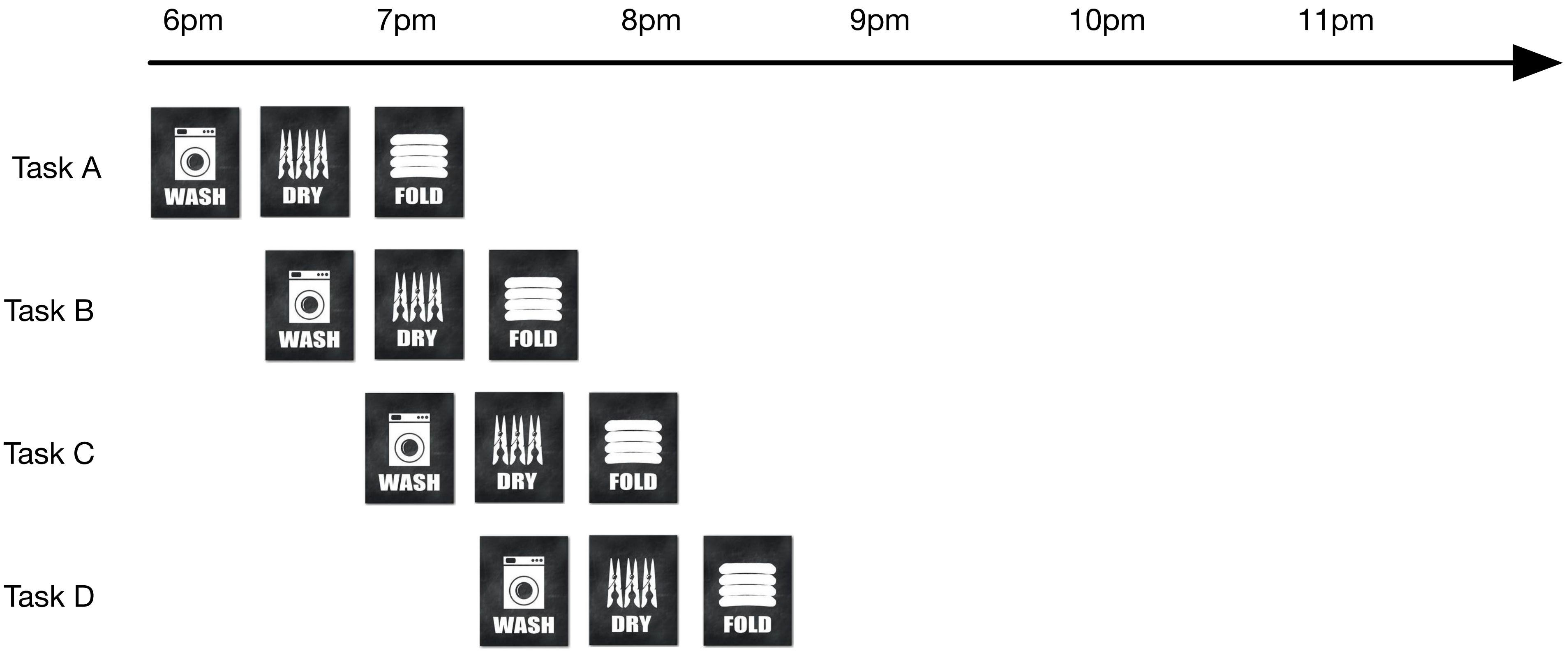
Philipp Koehn

~~26 March 2018~~
7 Oct 2019

Laundry Analogy


6pm
7pm
8pm
9pm
10pm
11pm
Task A
WASH
DRY
FOLD
Task B
WASH
DRY
FOLD
Task C
WASH
DRY
FOLD
Task D
WASH
DRY
FOLD

Laundry Pipelined


6pm
7pm
8pm
9pm
10pm
11pm
Task A
WASH
DRY
FOLD
Task B
WASH
DRY
FOLD
Task C
WASH
DRY
FOLD
Task D
WASH
DRY
FOLD

Speed-up

- Theoretical speed-up: 3 times
- Actual speed-up in example: 2 times
  - sequential: 1:30+1:30+1:30+1:30 = 6 hours
  - pipelined: 1:30+0:30+0:30+0:30 = 3 hours
- Many tasks → speed-up approaches theoretical limit

mips instruction pipeline

MIPS Pipeline

IF
- Fetch instruction from memory

DI
- Read registers and decode instruction
  (note: registers are always encoded in same place in instruction)

EX
- Execute operation OR calculate an address

MEM
- Access an operand in memory

WB
- Write result into a register

Time for Instructions

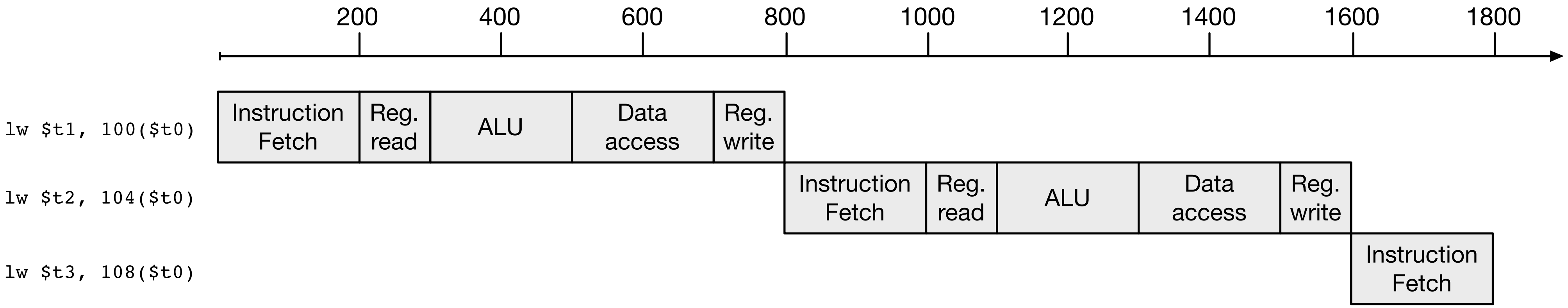
- Breakdown for each type of instruction

optimistic?

RAM is slow

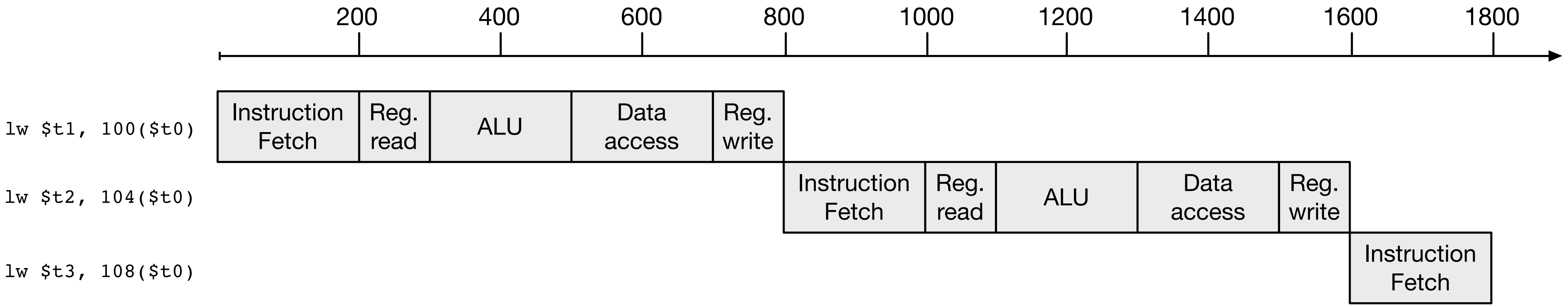
Instruction class	Instr. fetch	Register read	ALU oper.	Data access	Register write	Total time
Load word (lw)	200ps	100ps	200ps	200ps	100ps	800ps
Store word (lw)	200ps	100ps	200ps	200ps		700ps
R-format (add)	200ps	100ps	200ps		100ps	600ps
Brand (beq)	200ps	100ps	200ps			500ps

Pipeline Execution



200
400
600
800
1000
1200
1400
1600
1800
lw $t1, 100($t0)
Instruction Fetch
Reg. read
ALU
Data access
Reg. write
lw $t2, 104($t0)
Instruction Fetch
Reg. read
ALU
Data access
Reg. write
lw $t3, 108($t0)
Instruction Fetch

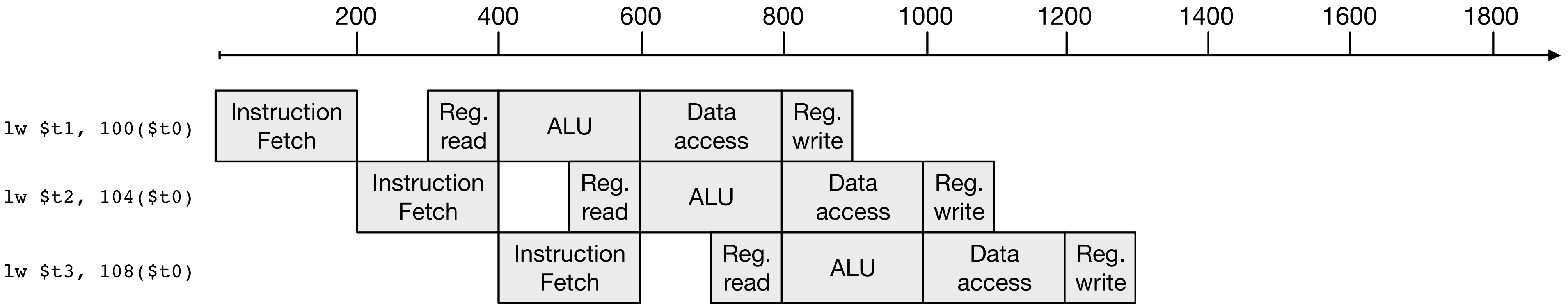
Pipeline Execution


200
400
600
800
1000
1200
1400
1600
1800

`lw $t1, 100($t0)` Instruction Fetch | Reg. read | ALU | Data access | Reg. write

`lw $t2, 104($t0)` Instruction Fetch | Reg. read | ALU | Data access | Reg. write

`lw $t3, 108($t0)` Instruction Fetch


200
400
600
800
1000
1200
1400
1600
1800

`lw $t1, 100($t0)` Instruction Fetch | Reg. read | ALU | Data access | Reg. write

`lw $t2, 104($t0)` Instruction Fetch | Reg. read | ALU | Data access | Reg. write

`lw $t3, 108($t0)` Instruction Fetch | Reg. read | ALU | Data access | Reg. write

Speed-up

- Theoretical speed-up: 4 times
- Actual speed-up in example: 1.71 times
  - sequential: 800ps + 800ps + 800ps = 2400ps
  - pipelined: 1000ps + 200ps + 200ps = 1400ps
- Many tasks → speed-up approaches theoretical limit

Design for Pipelining

- All instructions are 4 bytes

  → easy to fetch next instruction

Design for Pipelining

- All instructions are 4 bytes

  → easy to fetch next instruction

- Few instruction formats

  → parallel op decode and register read

Design for Pipelining

- All instructions are 4 bytes

  → easy to fetch next instruction

- Few instruction formats

  → parallel op decode and register read

- Memory access limited to load and store instructions

  → stage 3 used for memory access, otherwise operation execution

Design for Pipelining

- All instructions are 4 bytes

  → easy to fetch next instruction

- Few instruction formats

  → parallel op decode and register read

- Memory access limited to load and store instructions

  → stage 3 used for memory access, otherwise operation execution

- Words aligned in memory

  → able to read in one instruction

  (aligned = memory address multiple of 4)

hazards

Hazards

- Hazard = next instruction cannot be executed in next clock cycle

- Types
  - structural hazard
  - data hazard
  - control hazard

Structural Hazard

- Definition: instructions overlap in resource use in same stage

- For instance: memory access conflict

	1	2	3	4	5	6	7
i1	FETCH	DECODE	MEMORY	MEMORY	ALU	REGISTER	
i2		FETCH	DECODE	MEMORY	MEMORY	ALU	REGISTER
				conflict			

- MIPS designed to avoid structural hazards

Data Hazard

- Definition: instruction waits on result from prior instruction

- Example

data dependence

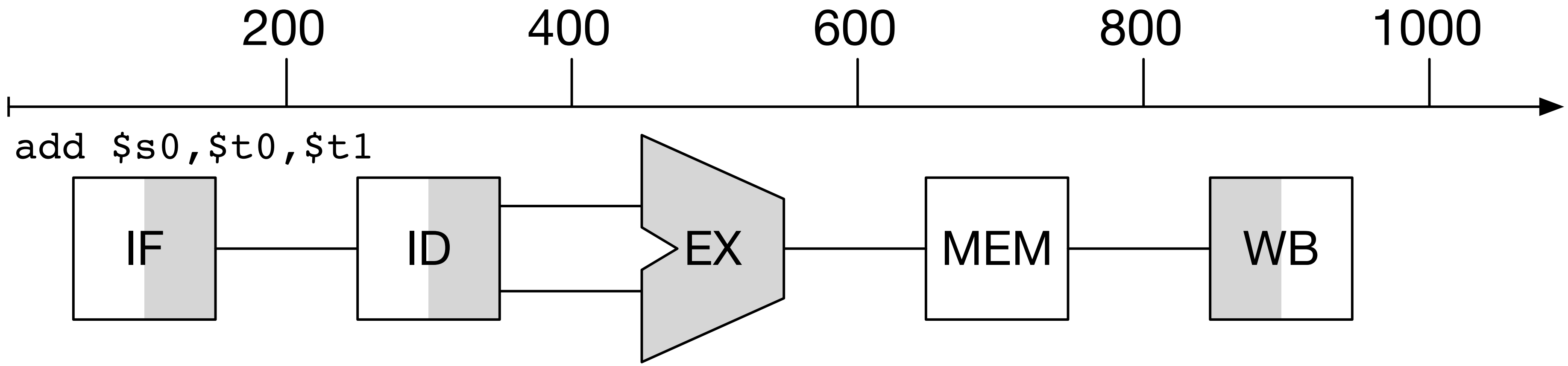
```
add $s0, $t0, $t1
sub $t0, $s0, $t3
```

  - add instruction writes result to register $s0 in stage 5
  - sub instruction reads $s0 in stage 2

⇒ Stage 2 of sub has to be delayed

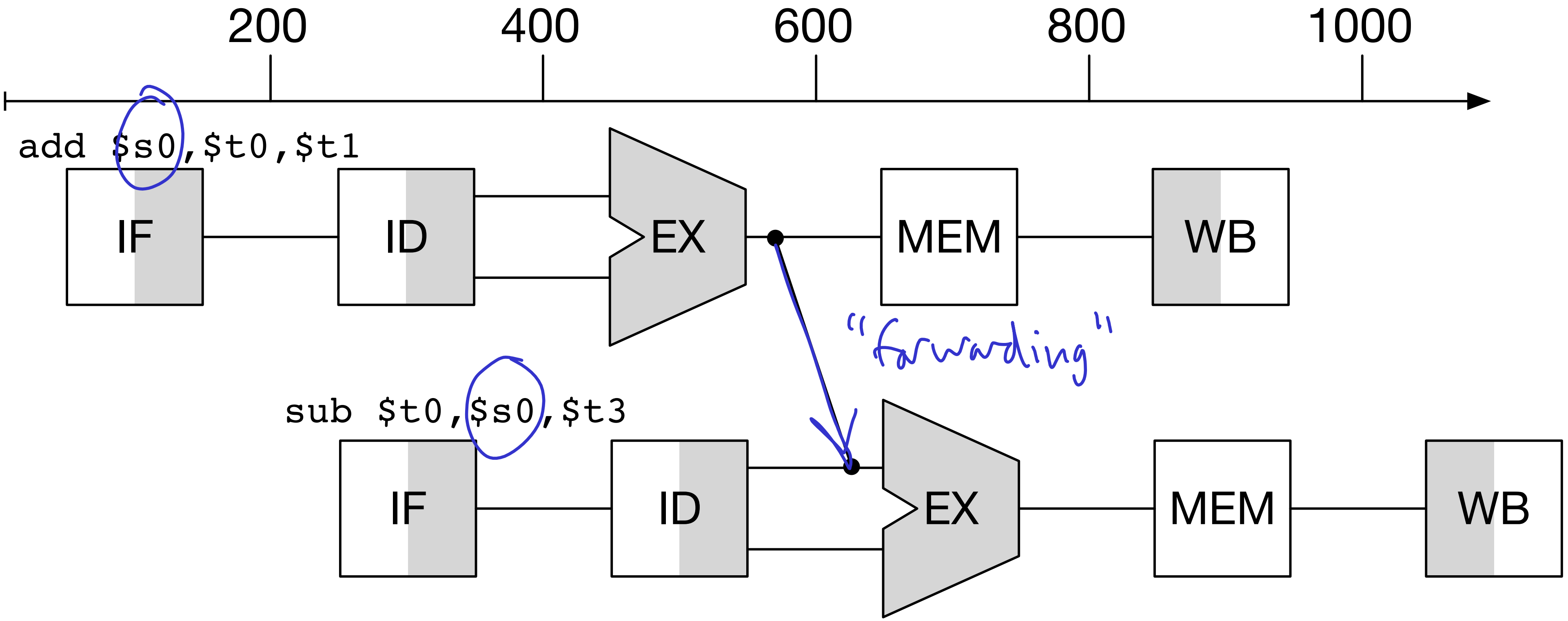
- We overcome this in hardware

Graphical Representation

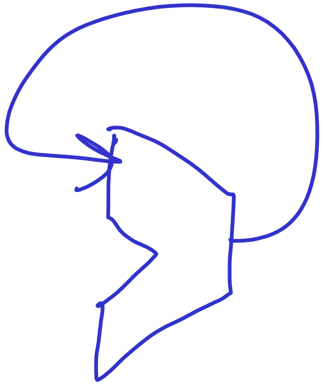

200
400
600
800
1000
add $s0,$t0,$t1
IF
ID
EX
MEM
WB

- IF: instruction fetch
- ID: instruction decode
- EX: execution
- MEM: memory access
- WB: write-back

Add and Subtract

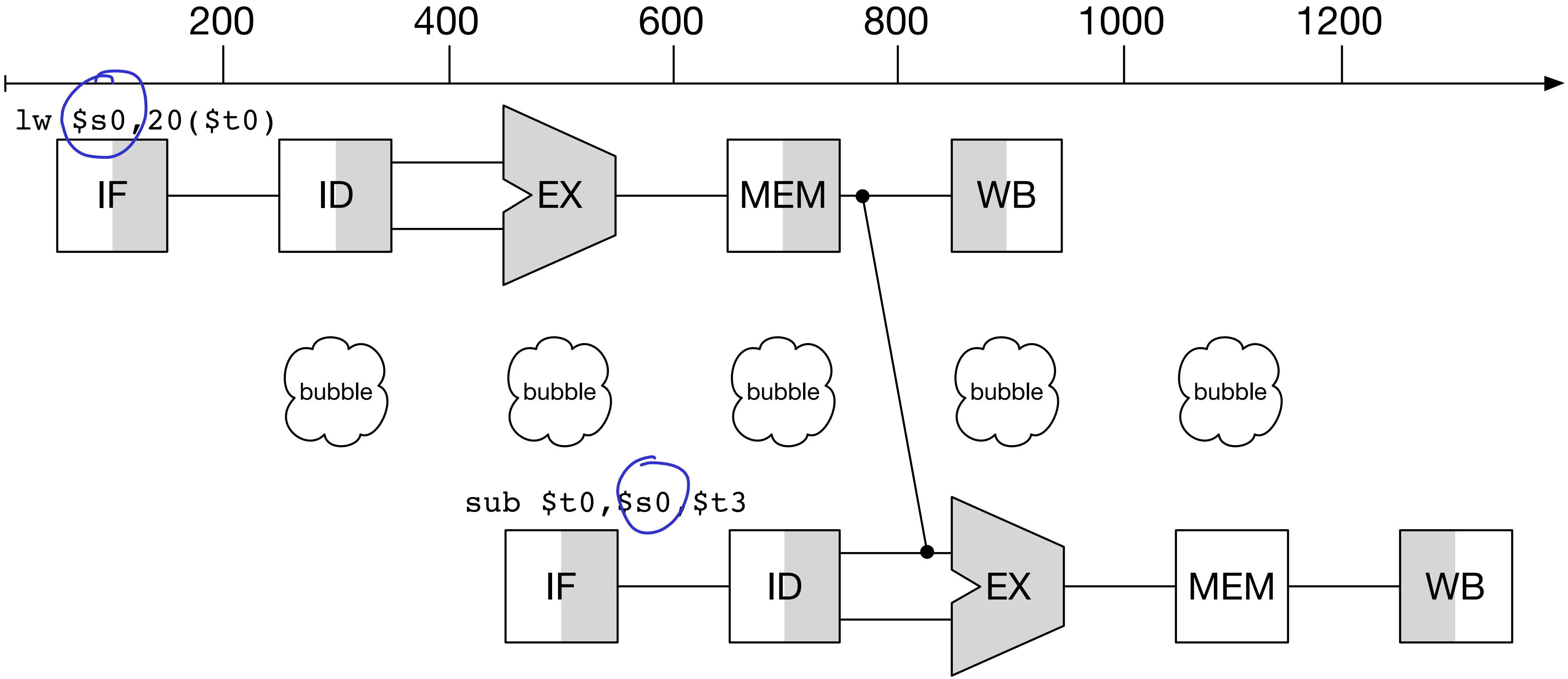

200
400
600
800
1000
add $s0,$t0,$t1
IF
ID
EX
MEM
WB
"forwarding"
sub $t0,$s0,$t3
IF
ID
EX
MEM
WB

- Add wiring to circuit to directly connect output of ALU for next instruction



need register!

Load and Subtract


200
400
600
800
1000
1200
lw $s0,20($t0)
IF
ID
EX
MEM
WB
bubble
bubble
bubble
bubble
bubble
sub $t0,$s0,$t3
IF
ID
EX
MEM
WB

- Add wiring from memory lookup to ALU
- Still 1 cycle unused: "pipeline stall" or "bubble"

Reorder Code

- Code with data hazard

```
lw   $t1, 0($t0)
lw   $t2, 4($t0)
add $t3, $t1, $t2
sw   $t3, 12($t0)
lw   $t4, 8($t0)
add $t5, $t1, $t4
sw   $t5, 16($t0)
```

Reorder Code

- Code with data hazard

```
lw  $t1, 0($t0)
lw  $t2, 4($t0)
add $t3, $t1, $t2
sw  $t3, 12($t0)
lw  $t4, 8($t0)
add $t5, $t1, $t4
sw  $t5, 16($t0)
```

- Reorder code (may be done by compiler)

Reorder Code

- Code with data hazard

```
lw  $t1, 0($t0)
lw  $t2, 4($t0)
add $t3, $t1, $t2
sw  $t3, 12($t0)
lw  $t4, 8($t0)
add $t5, $t1, $t4
sw  $t5, 16($t0)
```

```
lw  $t1, 0($t0)
lw  $t2, 4($t0)
lw  $t4, 8($t0)
add $t3, $t1, $t2
sw  $t3, 12($t0)
add $t5, $t1, $t4
sw  $t5, 16($t0)
```

forwarding

- Reorder code (may be done by compiler)

- Load instruction now completed in time

Control Hazard

- Also called branch hazard

- Selection of next instruction depends on outcome of previous

- Example

```
  add $s0, $t0, $t1
? beq $s0, $s1, ff40
  sub $t0, $s0, $t3
```

  - sub instruction only executed if branch condition fails

→ cannot start until branch condition result known

Branch Prediction

- Assume that branches are never taken
  → full speed if correct

this is not
a full explanation

- More sophisticated
  - keep record of branch taken or not
  - make prediction based on history

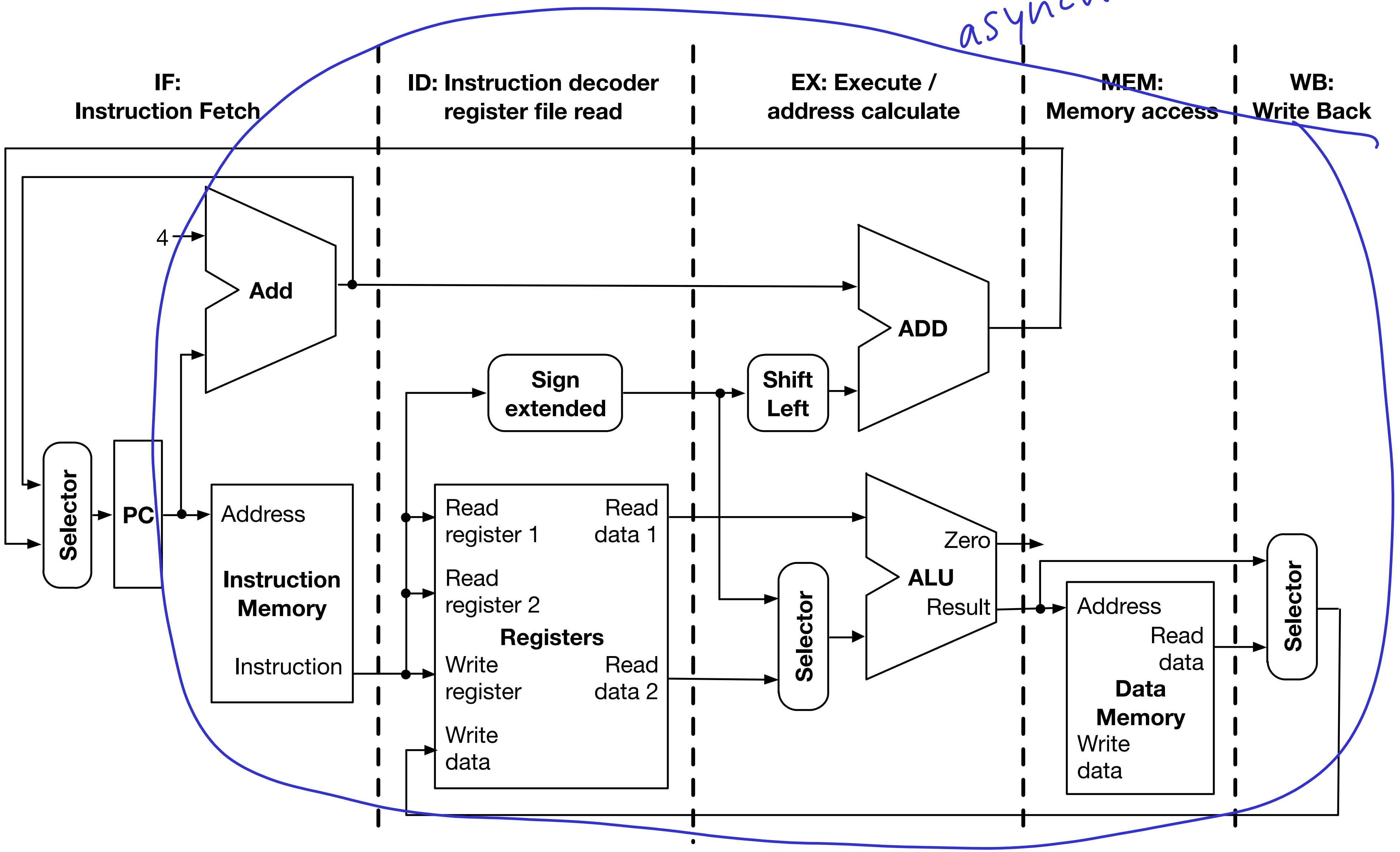
→ "branch delay"

speculative
execution

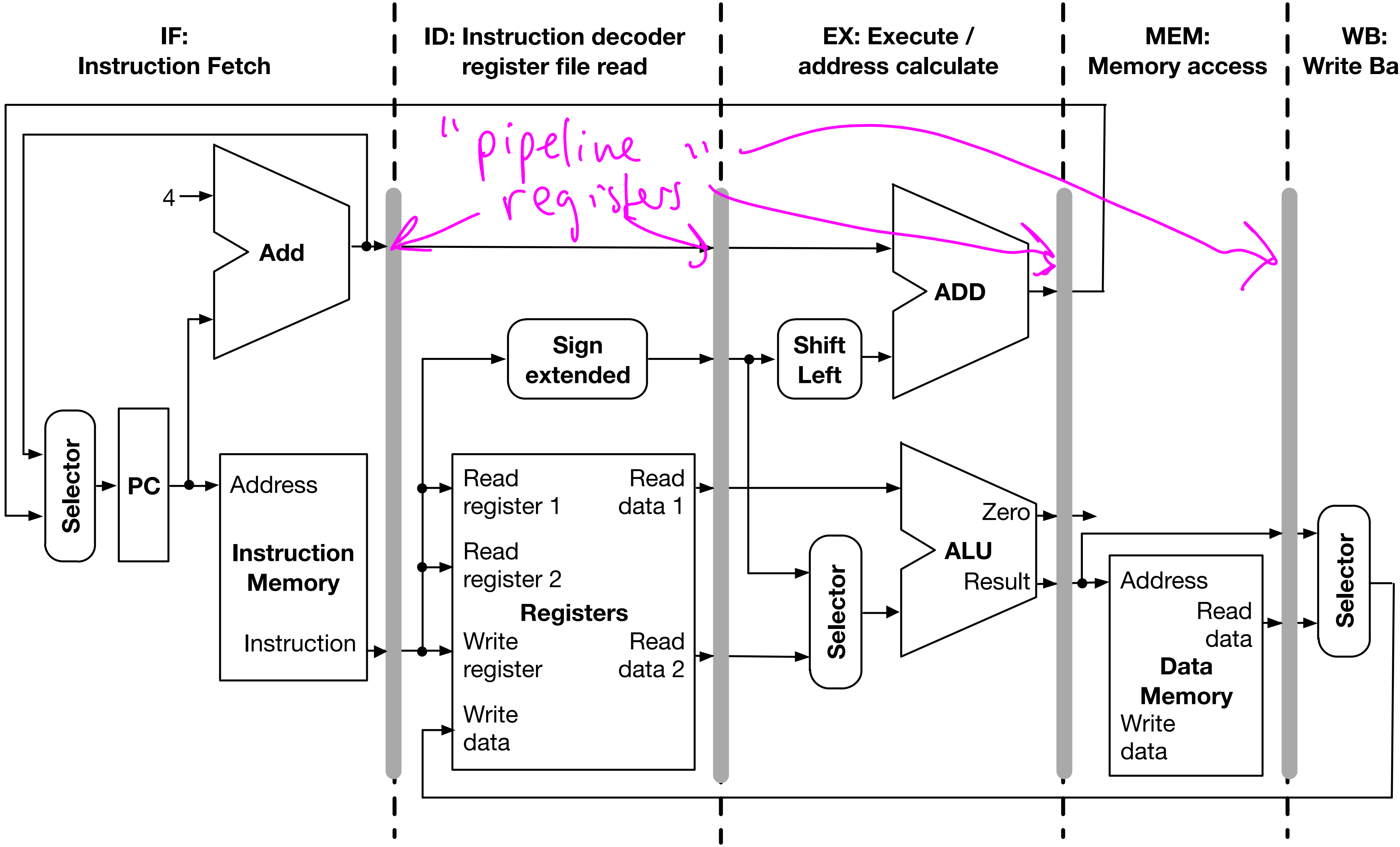
pipelined data path

Datapath

combinational
or
asynchonous

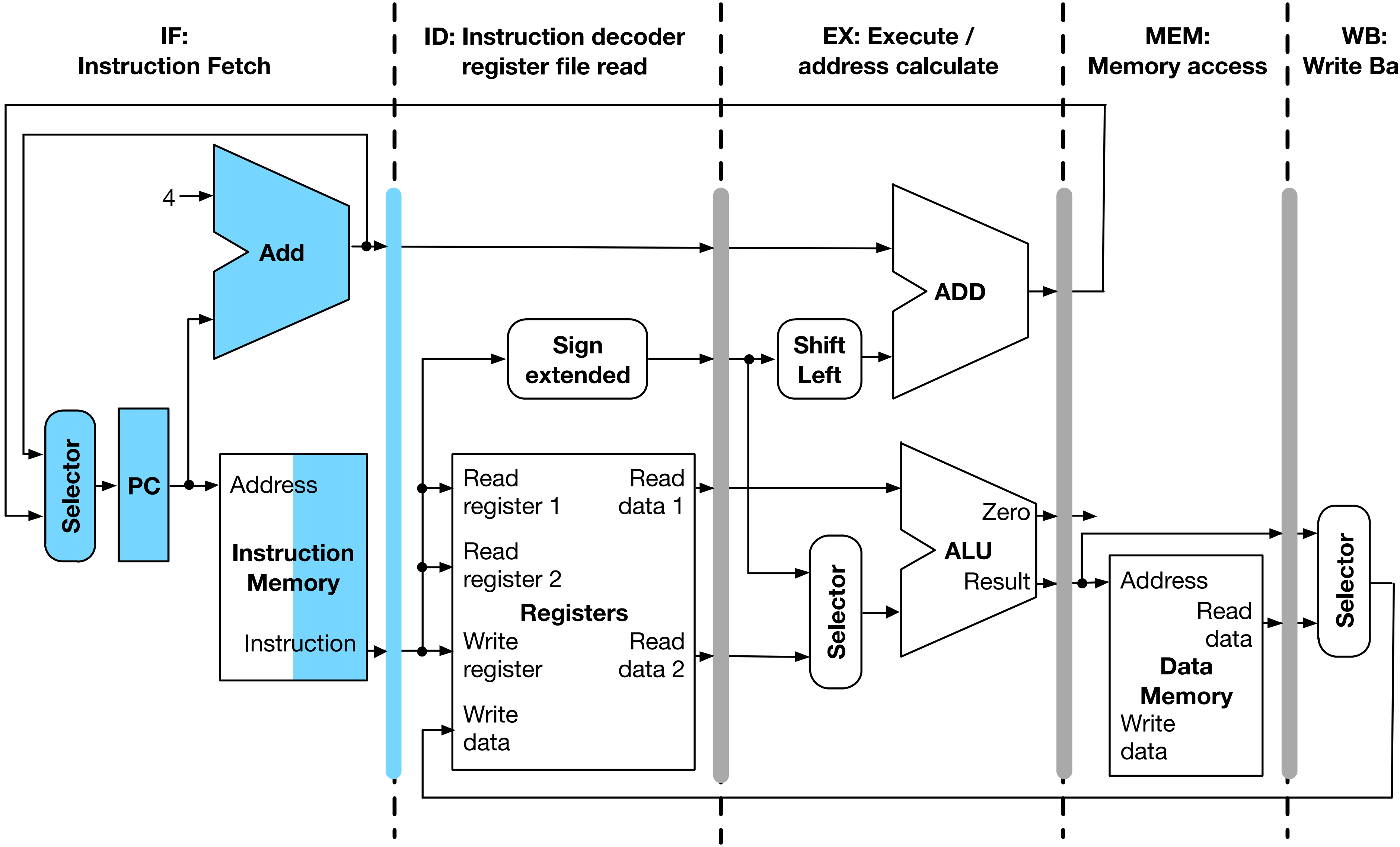

IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
ADD
Sign
extended
Shift
Left
Selector
PC
Address
Instruction
Memory
Instruction
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

Pipelined Datapath

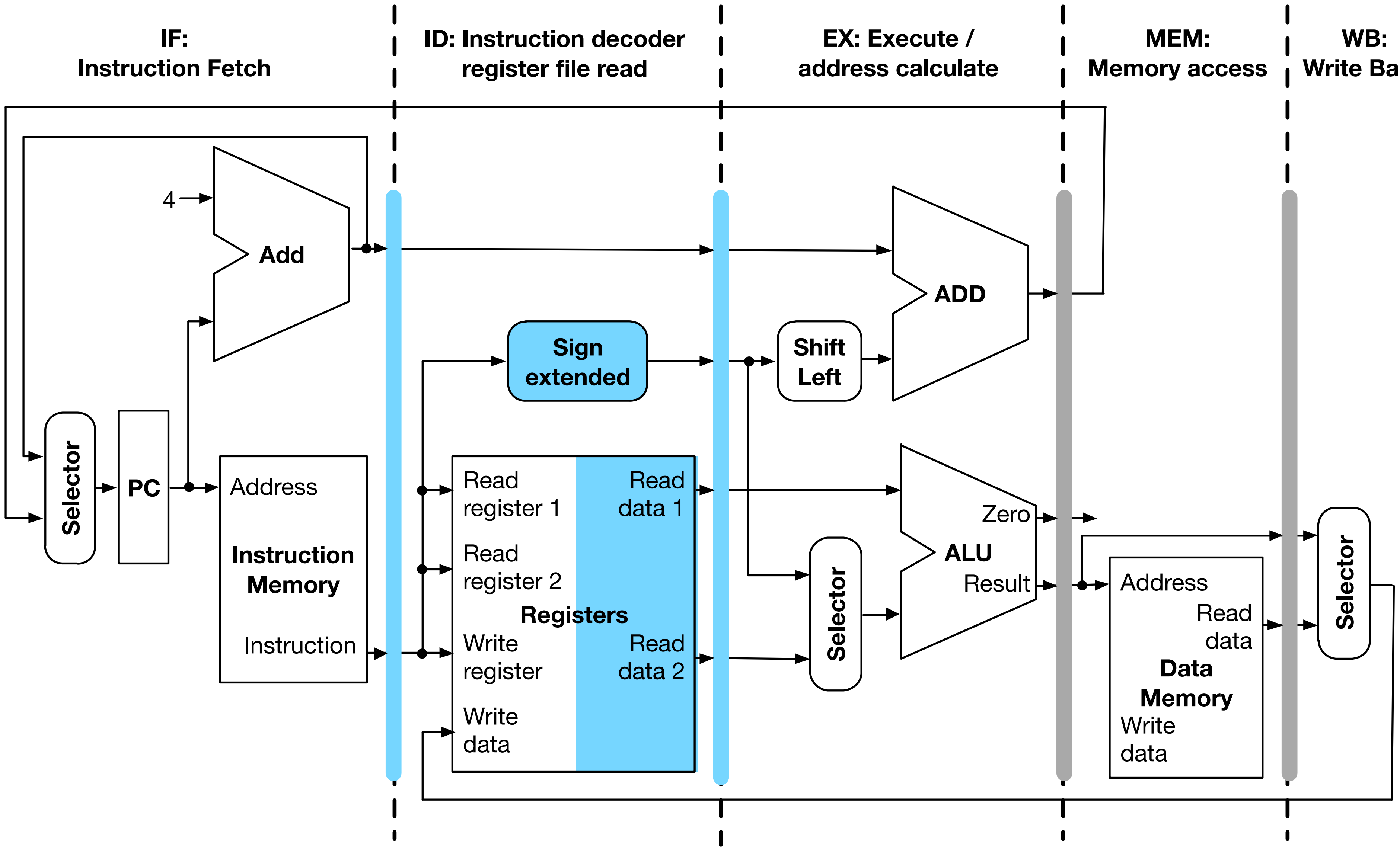

IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
"pipeline registers"
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

`load`

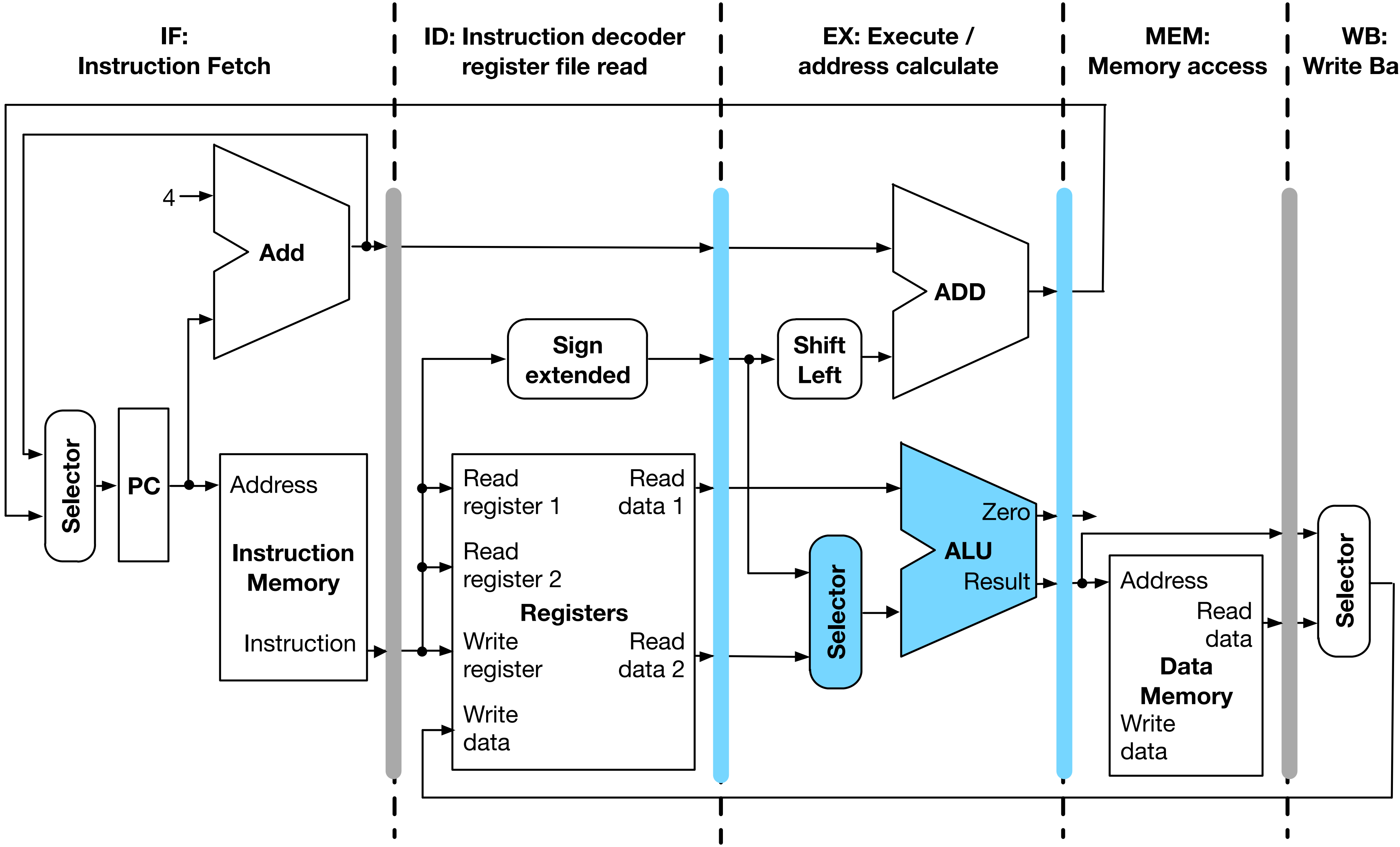
Load: Stage 1


IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

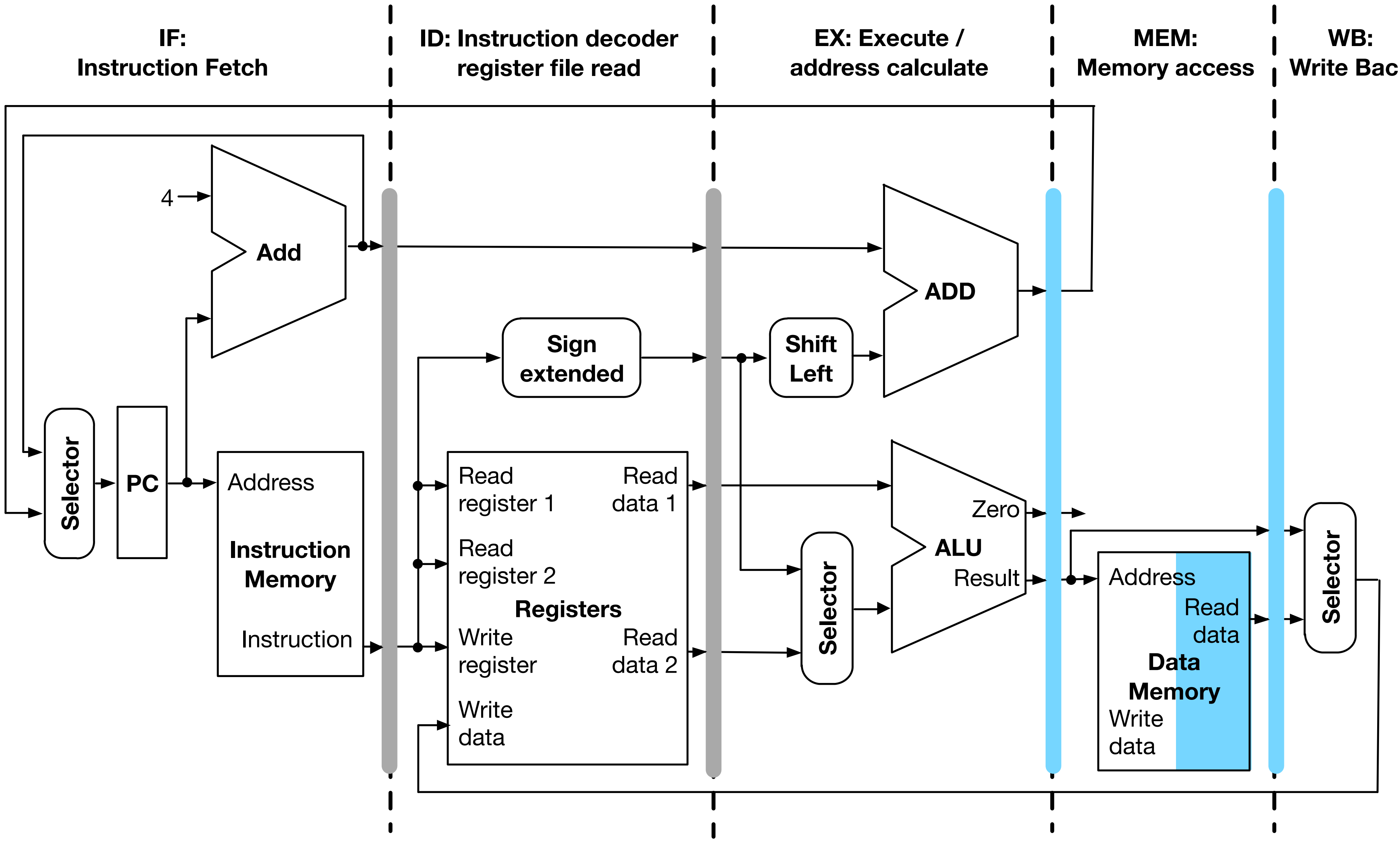
Load: Stage 2


IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Write
register
Write
data
Read
data 1
Read
data 2
Registers
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

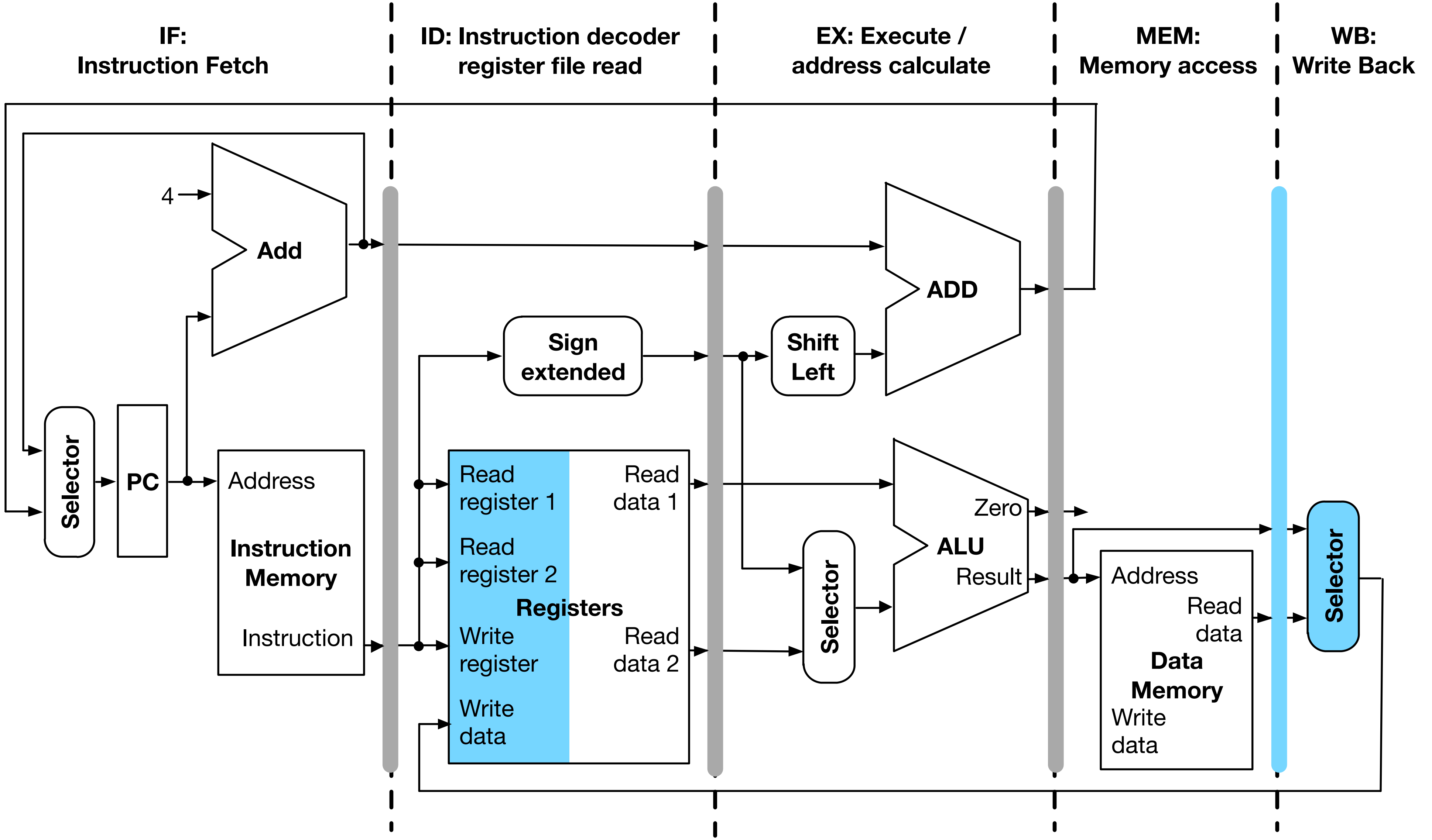
Load: Stage 3


IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

Load: Stage 4

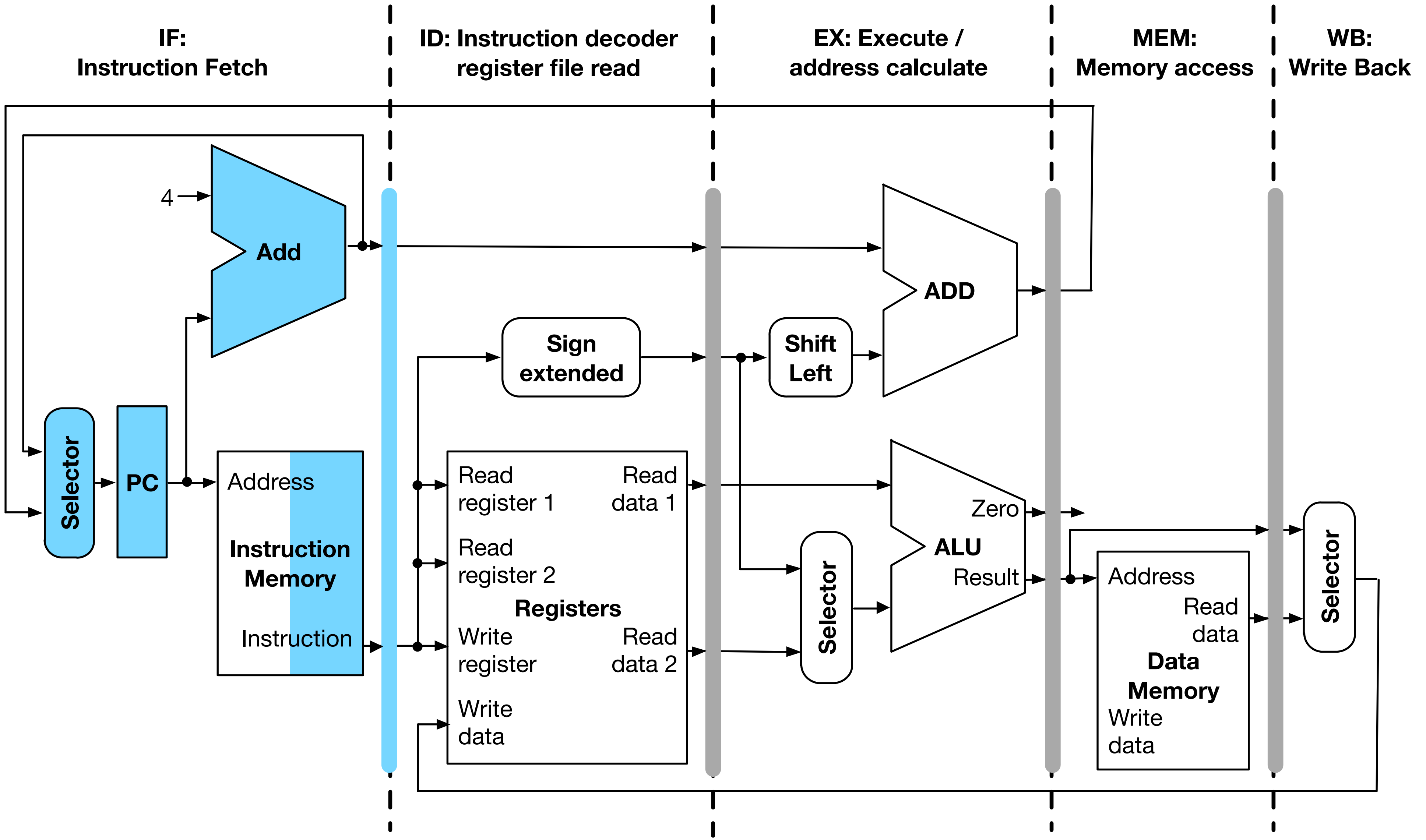

IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

Load: Stage 5

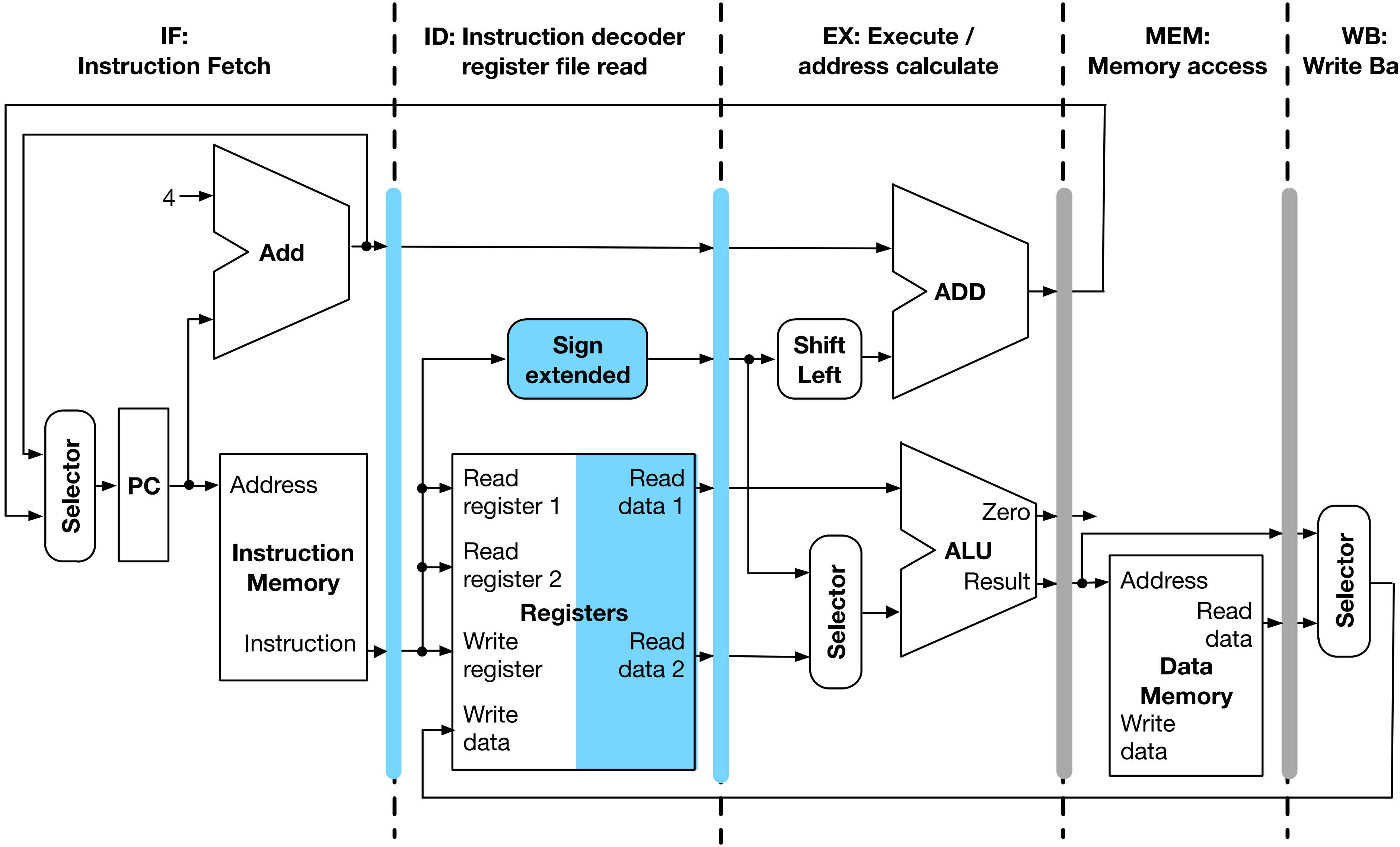

IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

`store`

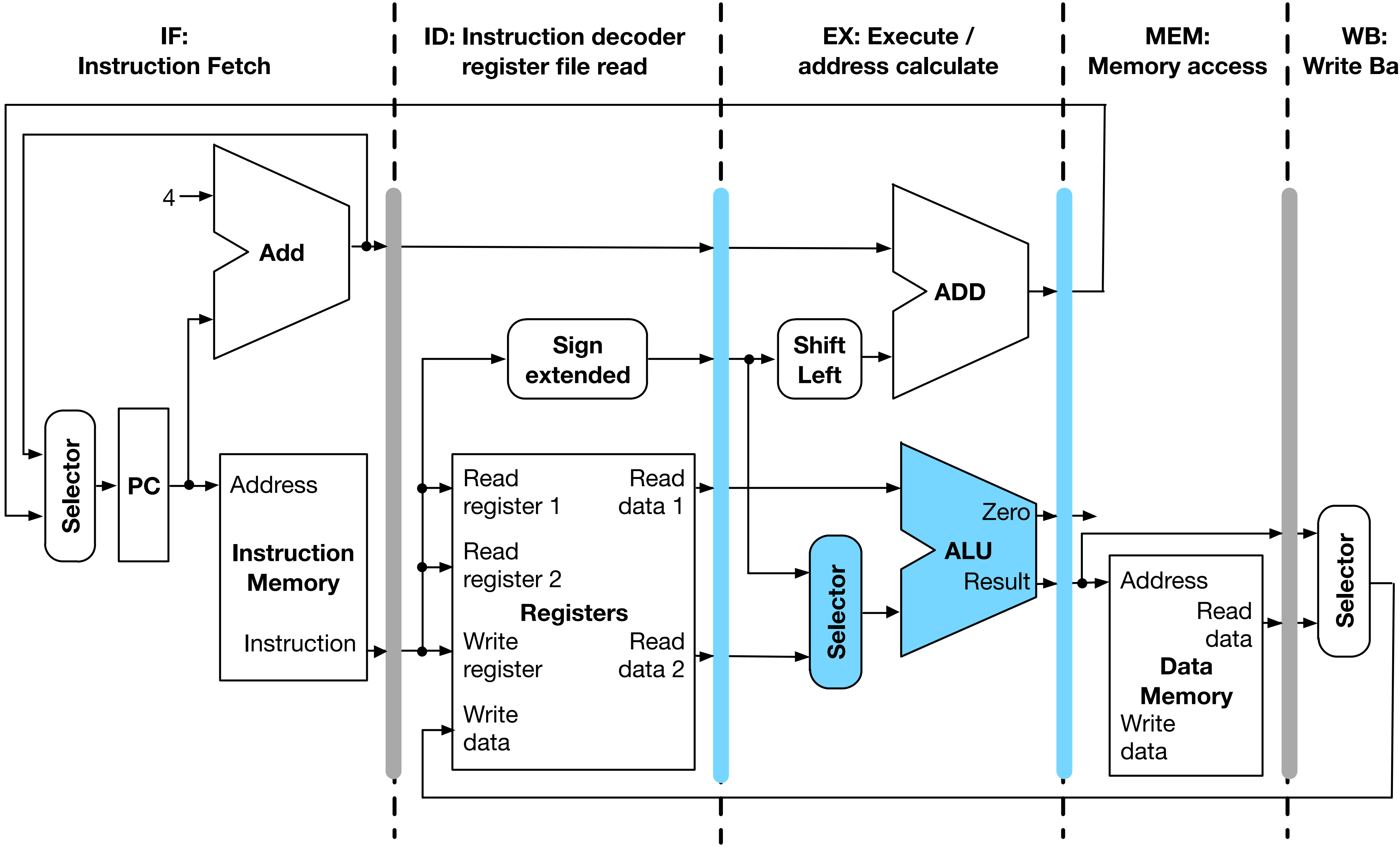
Store: Stage 1


IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

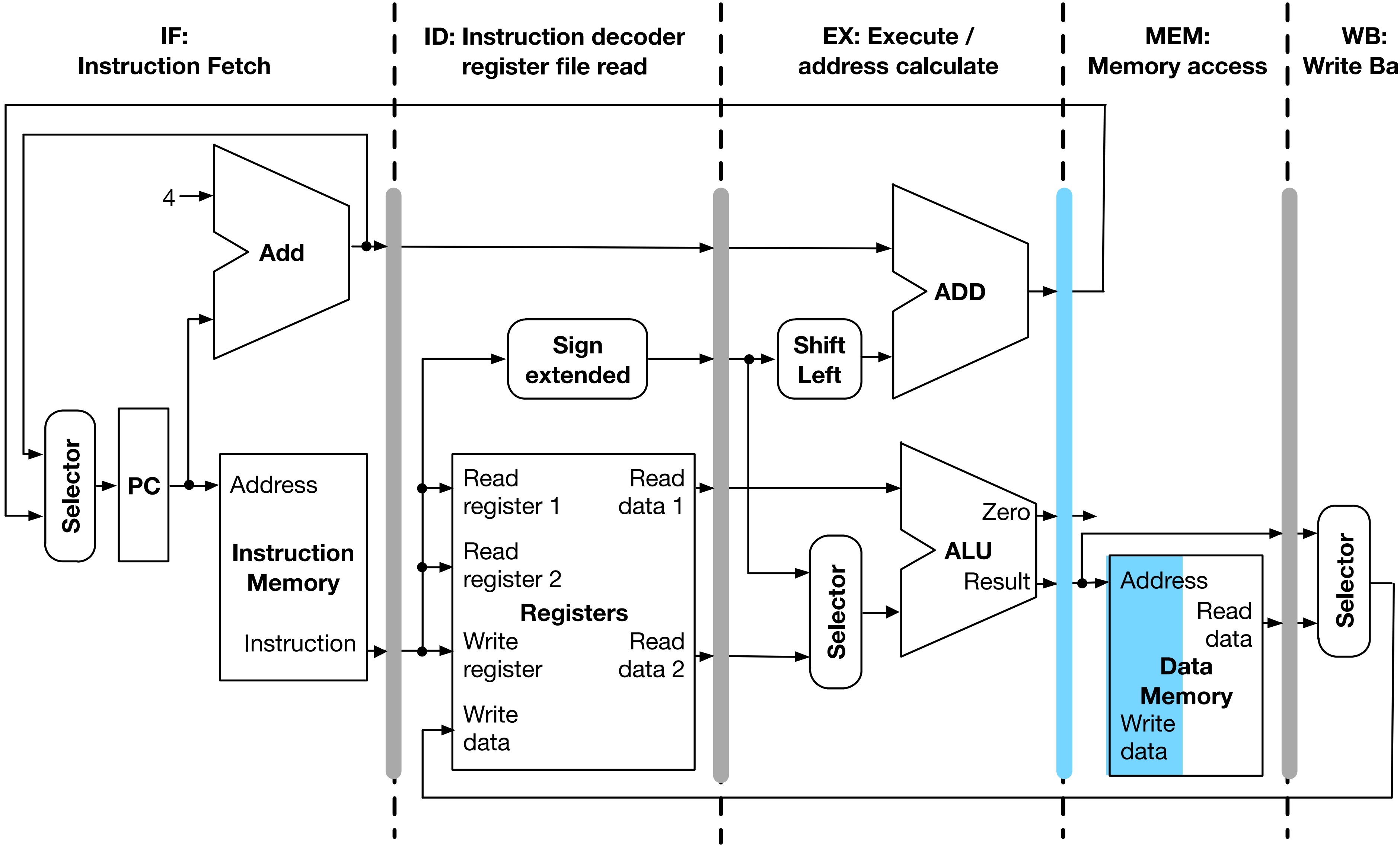
Store: Stage 2


IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

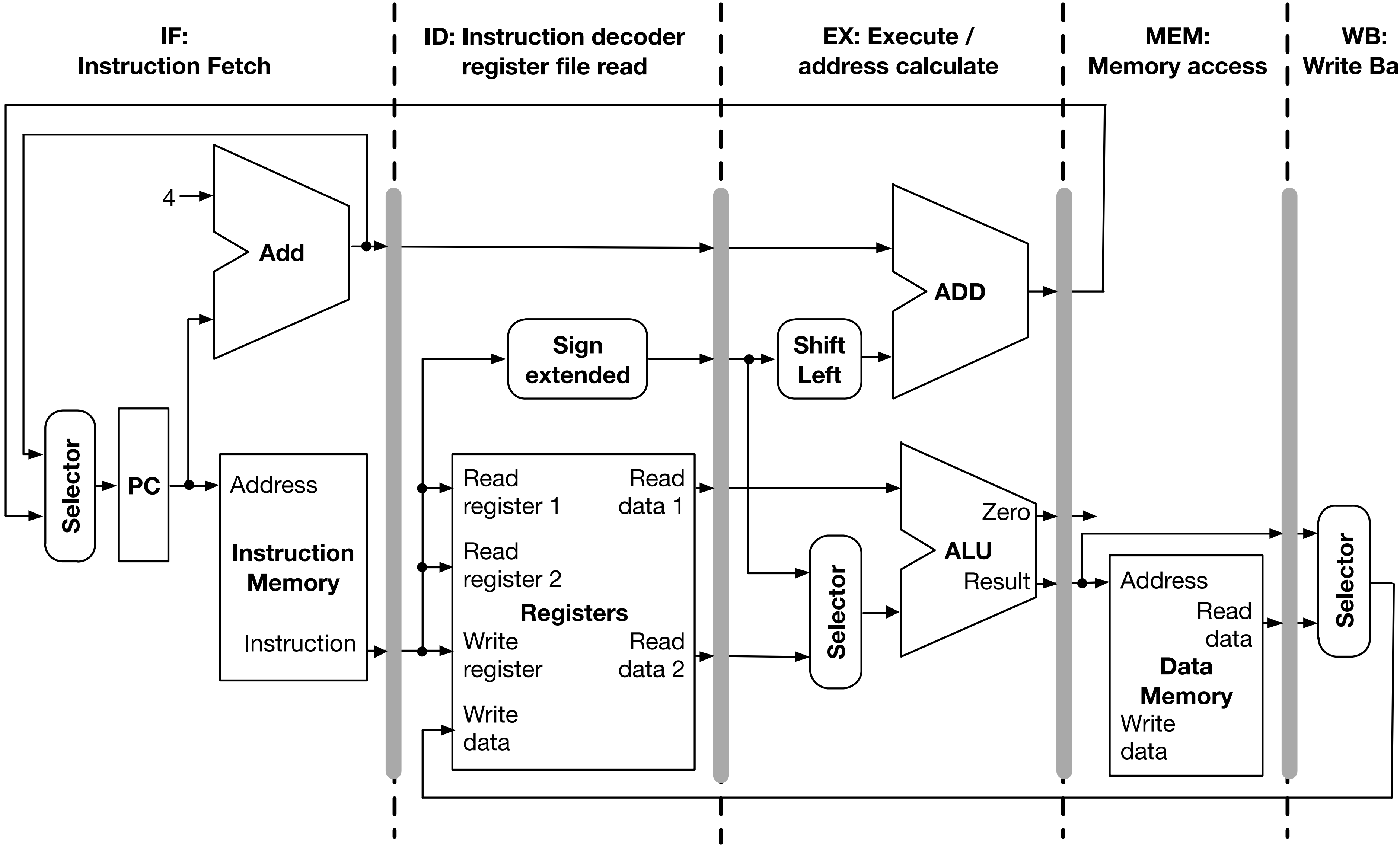
Store: Stage 3


IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

Store: Stage 4

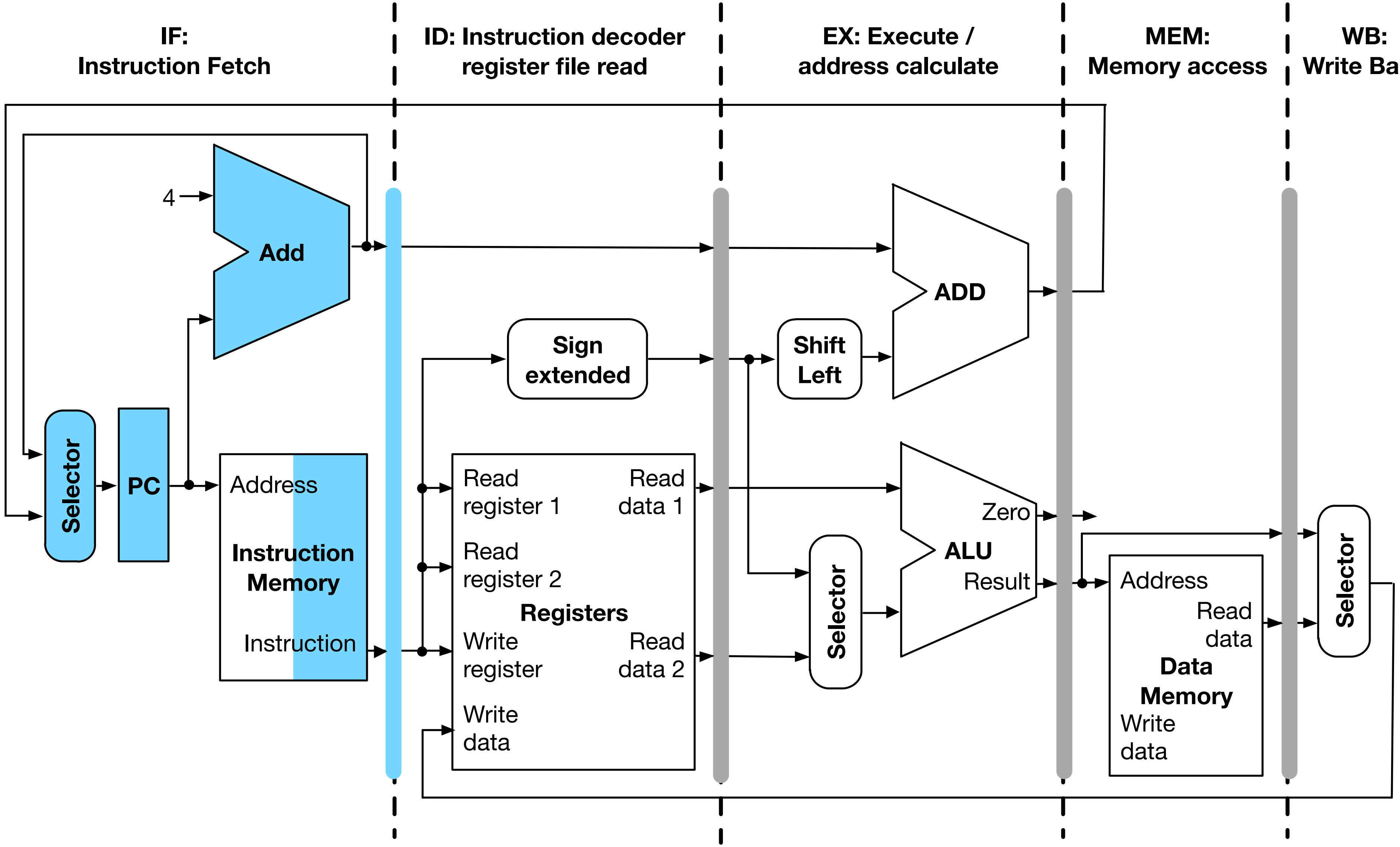

IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

Store: Stage 5

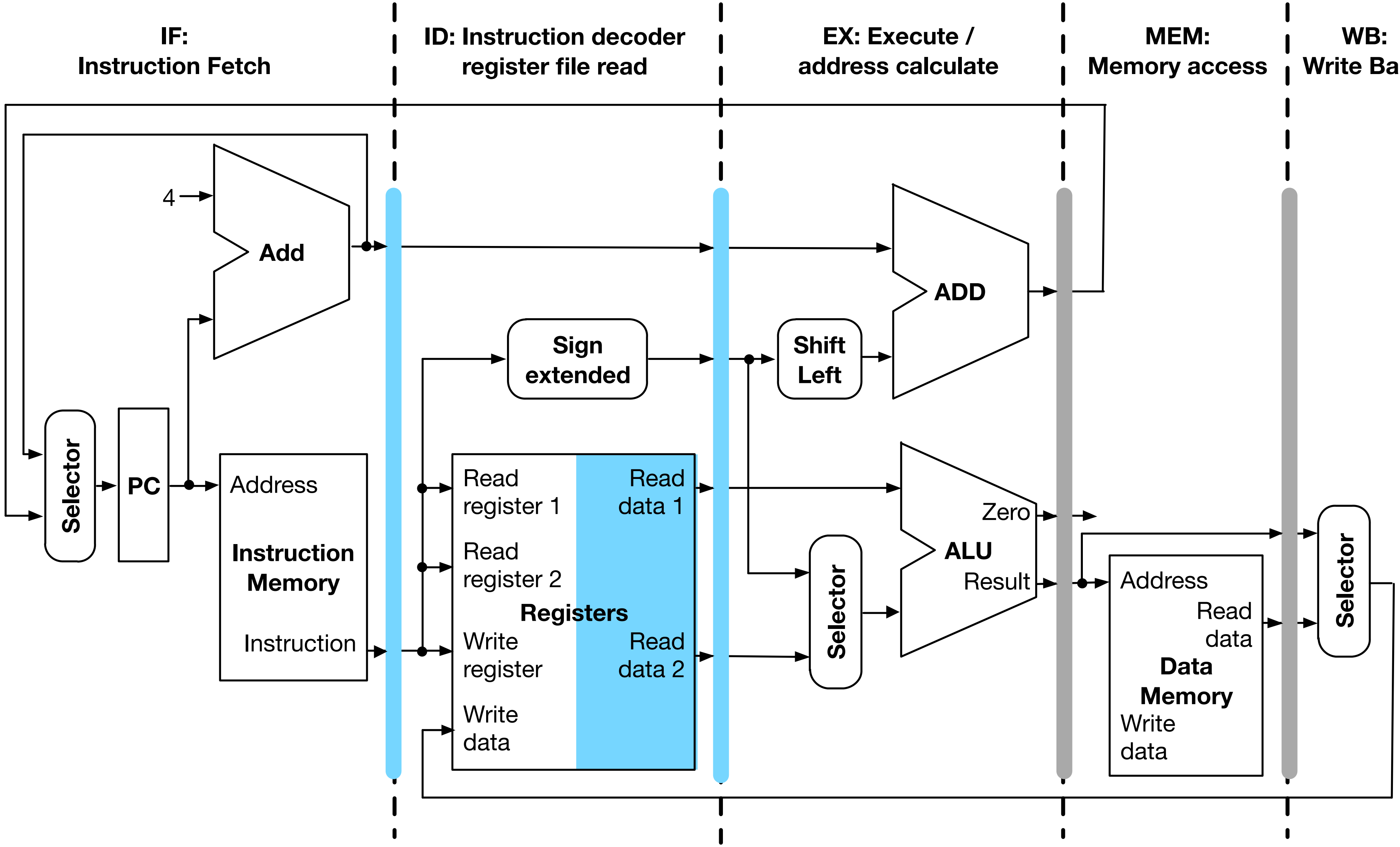

IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

`add`

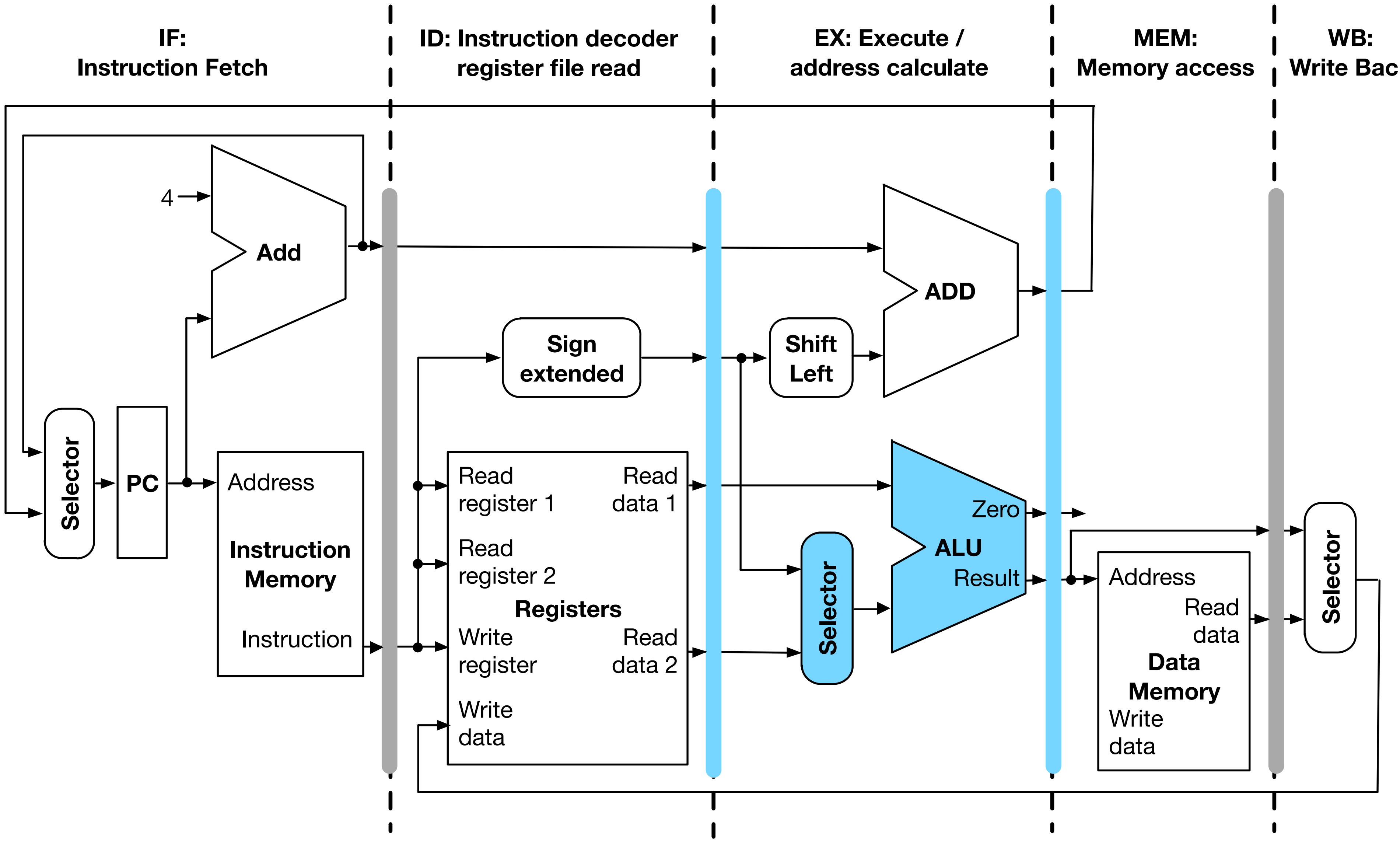
Add: Stage 1


IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

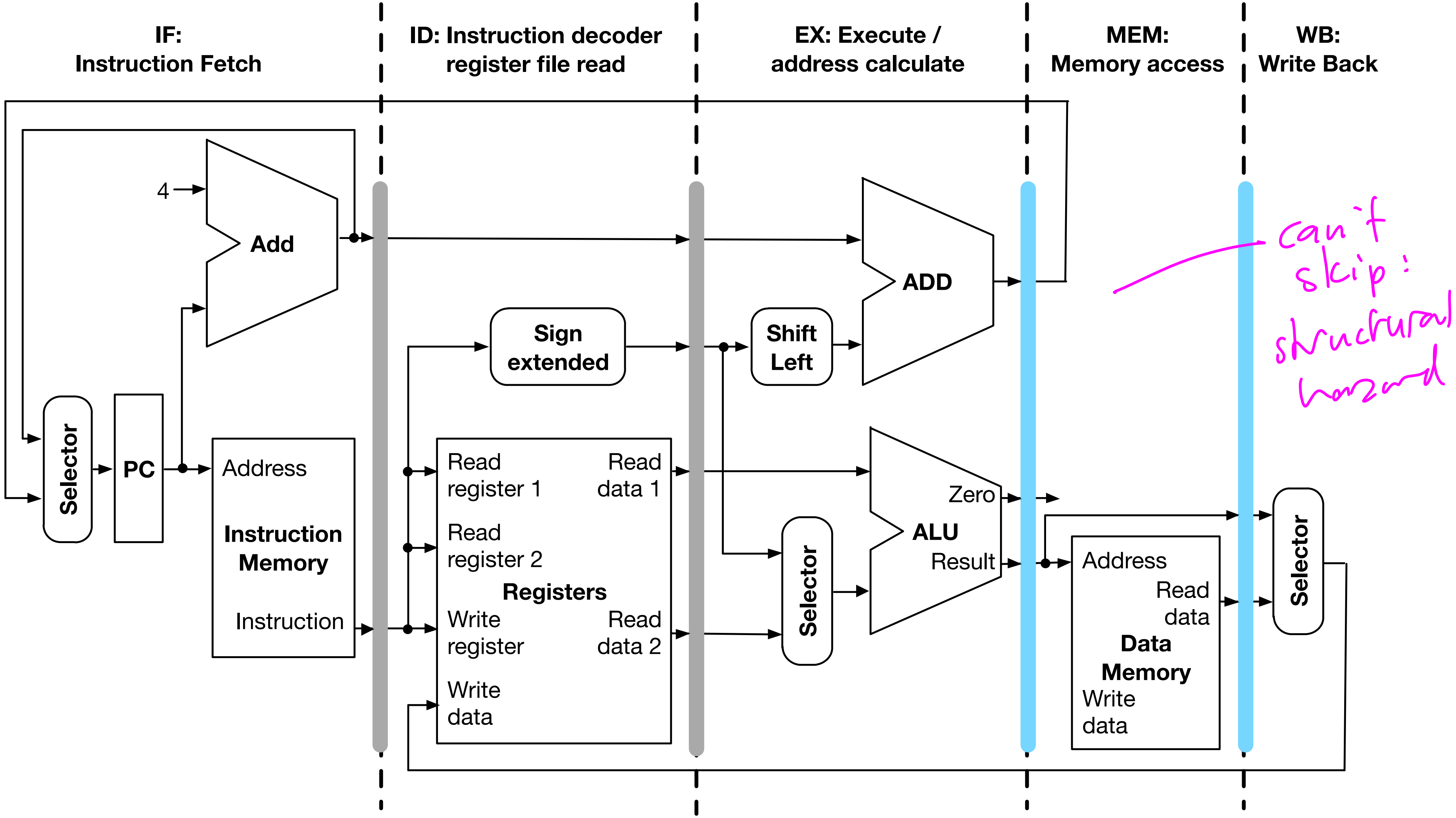
Add: Stage 2


IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
data 1
Read
register 2
Registers
Write
register
Read
data 2
Write
data
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

Add: Stage 3

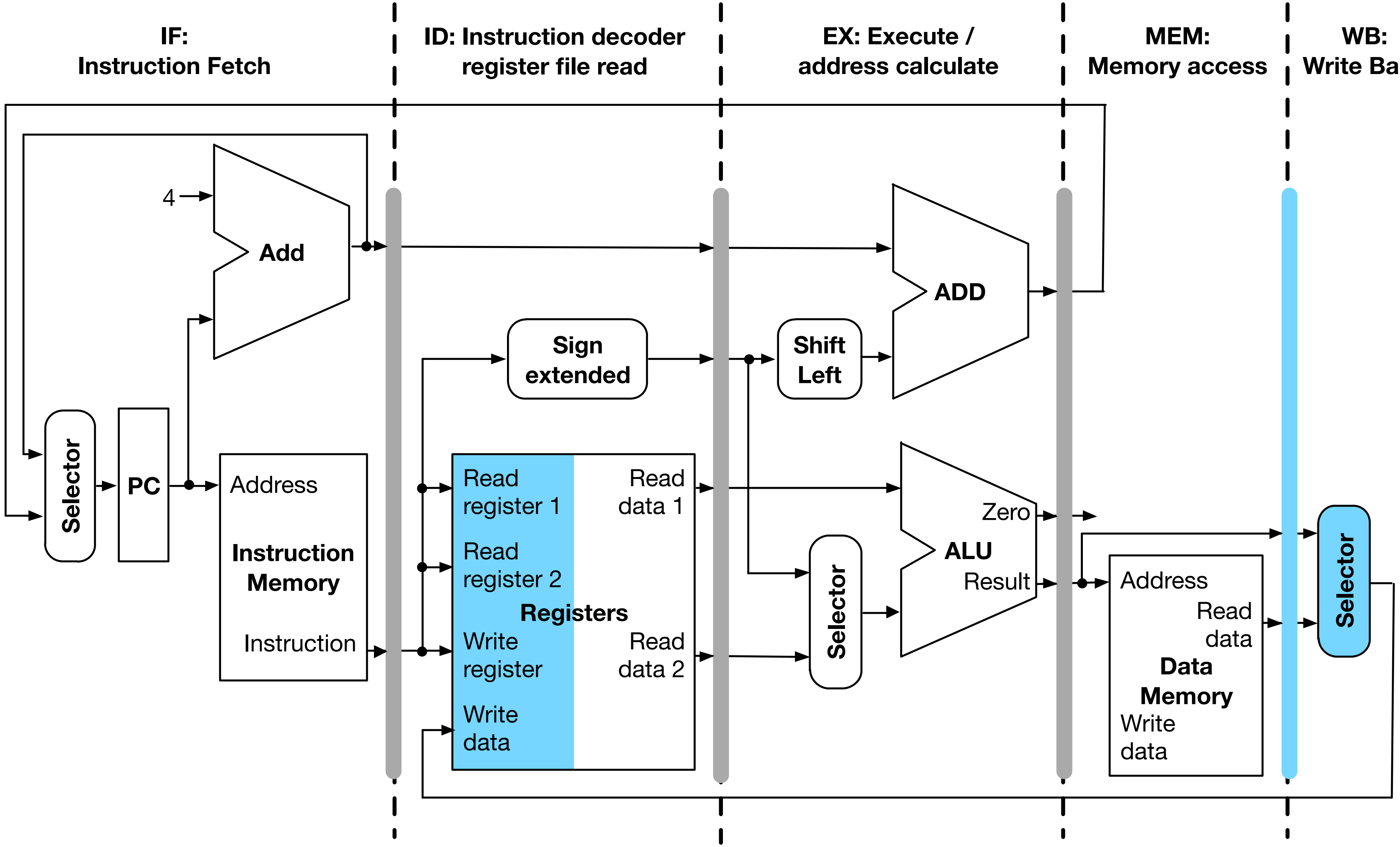

IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

Add: Stage 4


IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

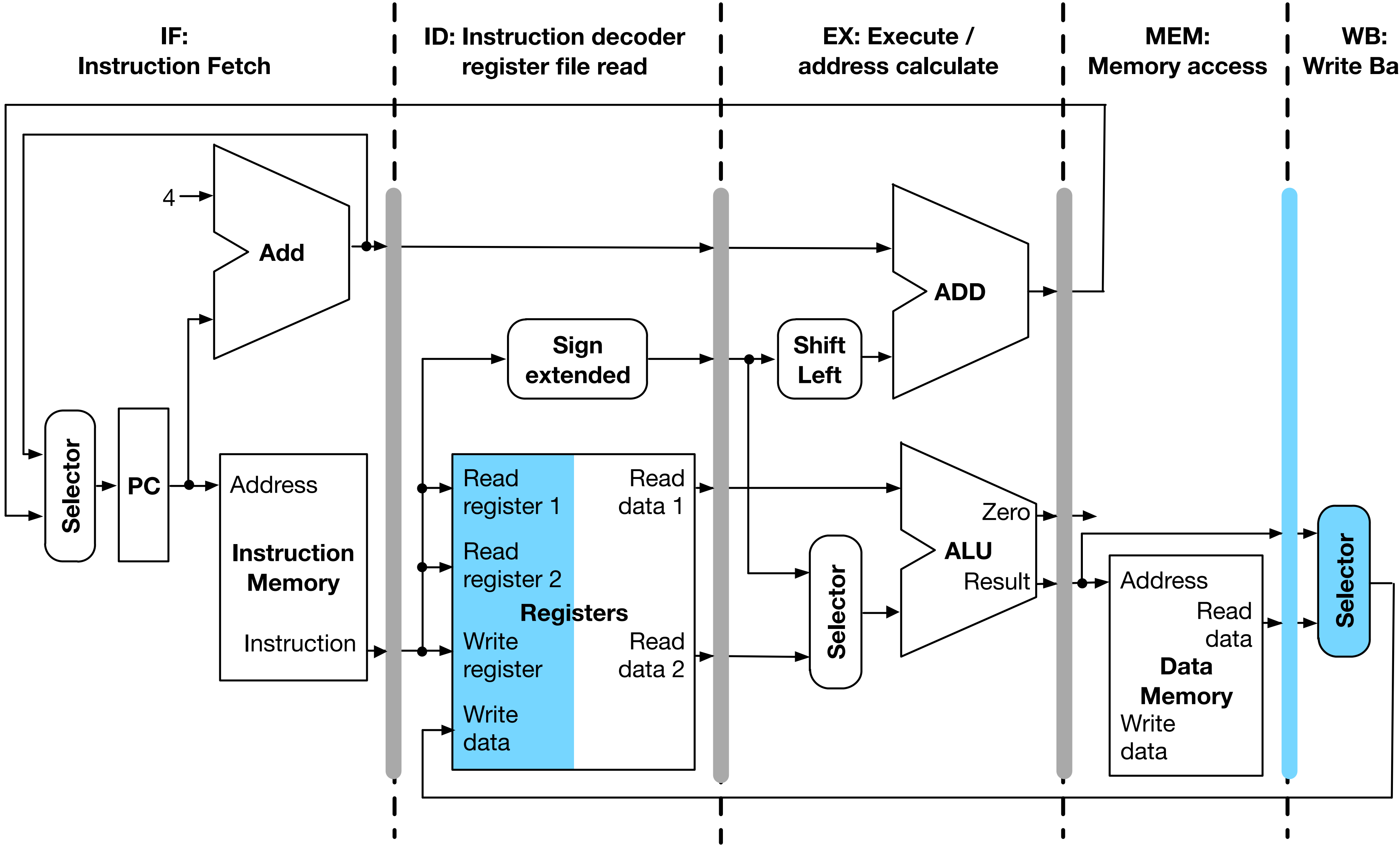
can't skip: structural hazard

Add: Stage 5


IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

`write to register`

Which Register?

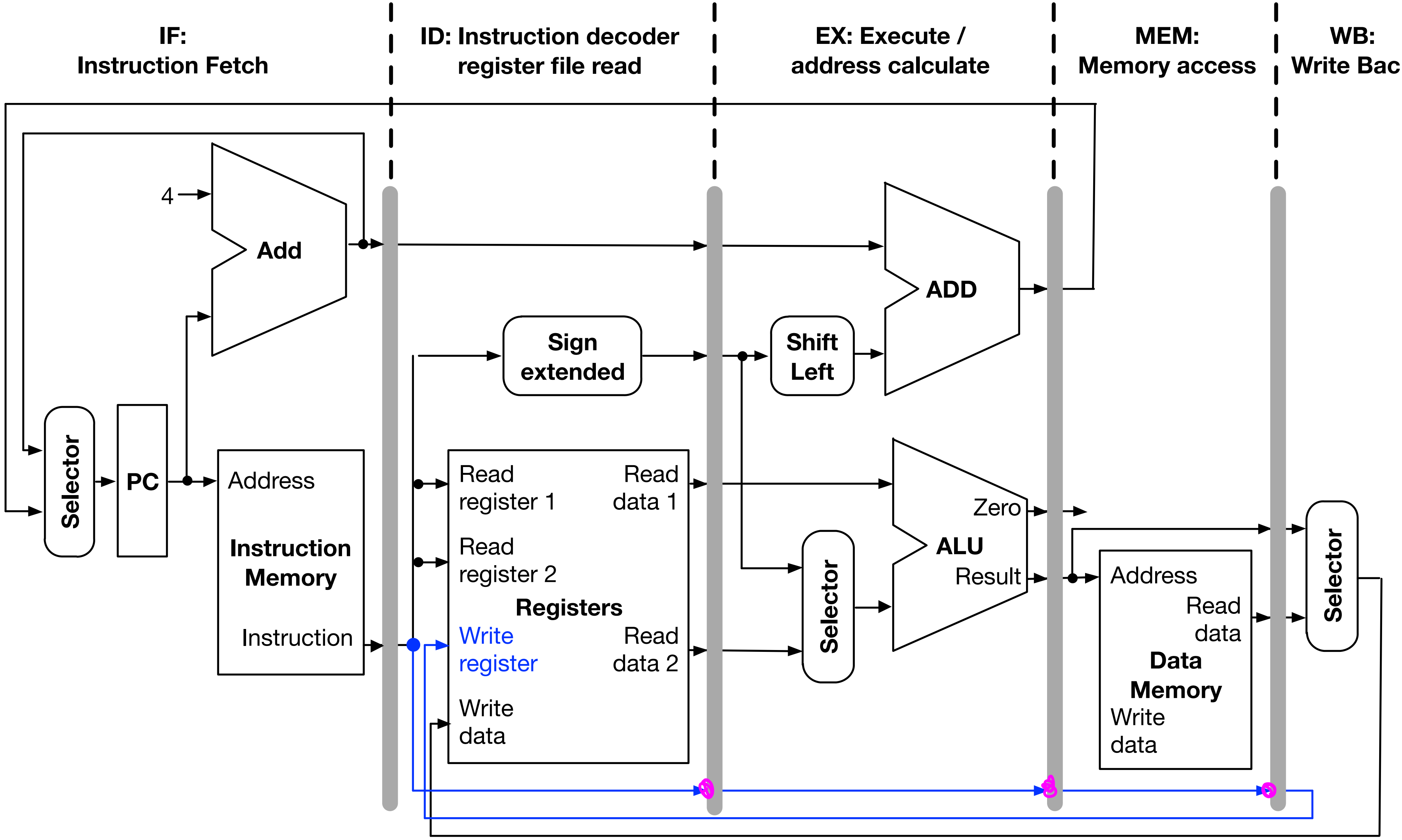

IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

Problem

- Write register
  - decoded in stage 2
  - used in stage 5

- Identity of register has to be passed along

Data Path for Write Register


IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
ADD
Sign
extended
Shift
Left
Selector
PC
Address
Instruction
Memory
Instruction
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
Selector
Zero
ALU
Result
Address
Read
data
Data
Memory
Write
data
Selector

`pipelined control`

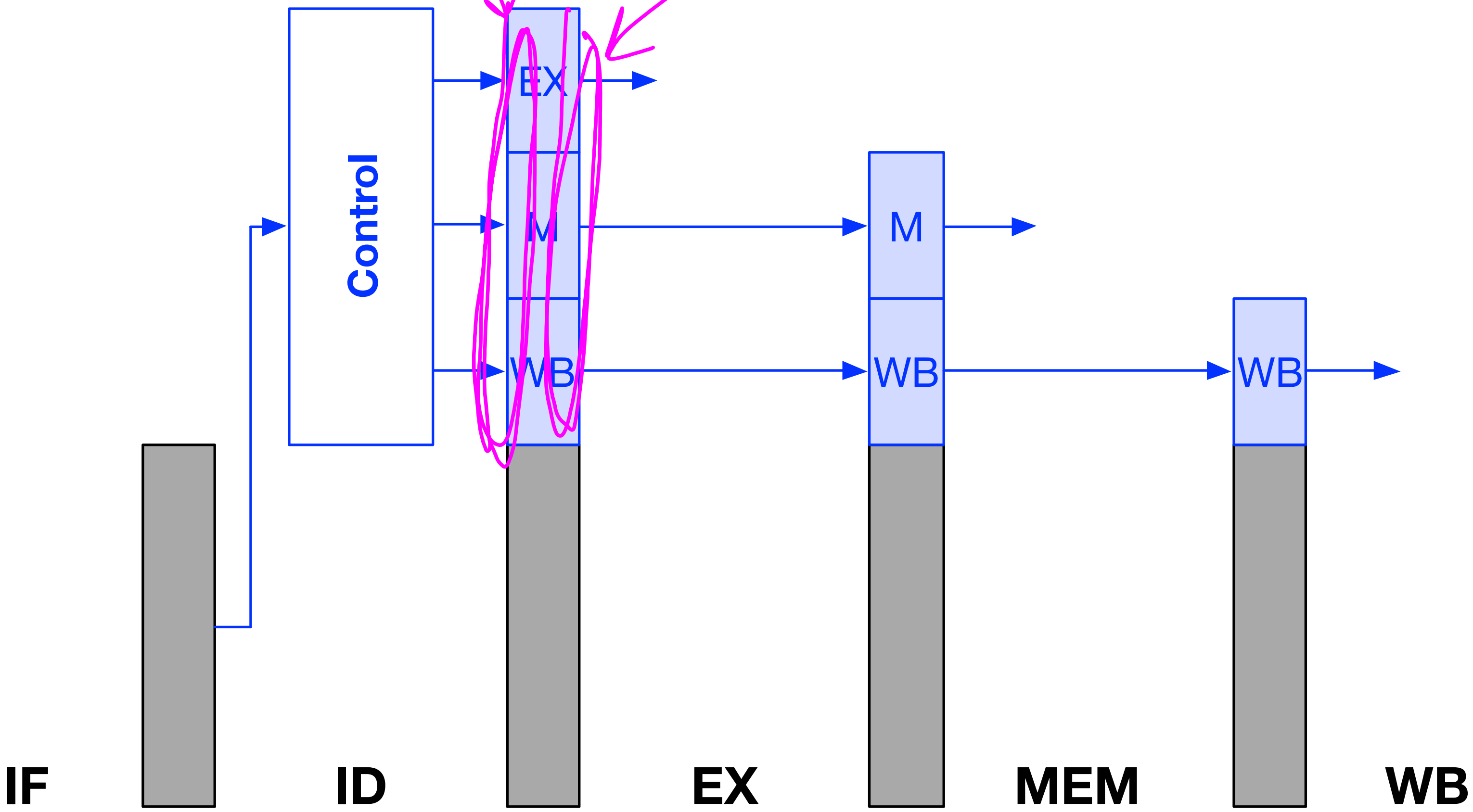
Pipelined Control

- At each stage, information from instruction is needed
  - which ALU operation to execute
  - which memory address to consult
  - which register to write to
- This control information has to be passed through stages

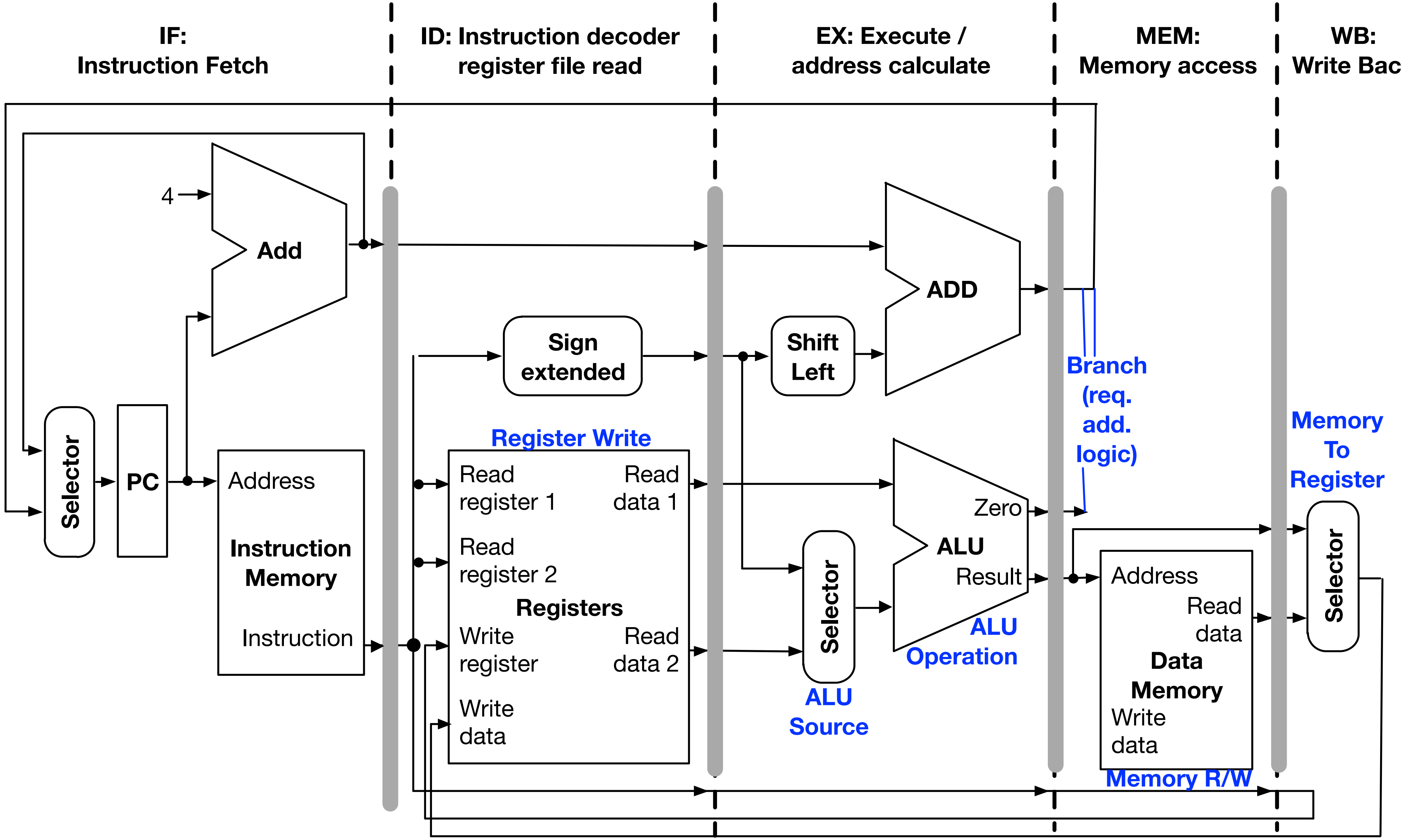
Pipelined Control

time t: written by ID

time t+1: read by EX


Control
EX
M
WB
M
WB
WB
IF
ID
EX
MEM
WB

Control Flags


IF:
Instruction Fetch
ID: Instruction decoder
register file read
EX: Execute /
address calculate
MEM:
Memory access
WB:
Write Back
4
Add
Selector
PC
Address
Instruction
Memory
Instruction
Sign
extended
Register Write
Read
register 1
Read
register 2
Write
register
Write
data
Registers
Read
data 1
Read
data 2
Shift
Left
ADD
Selector
ALU
Source
Zero
ALU
Result
ALU
Operation
Branch
(req.
add.
logic)
Address
Read
data
Data
Memory
Write
data
Memory R/W
Memory
To
Register
Selector