CS2100

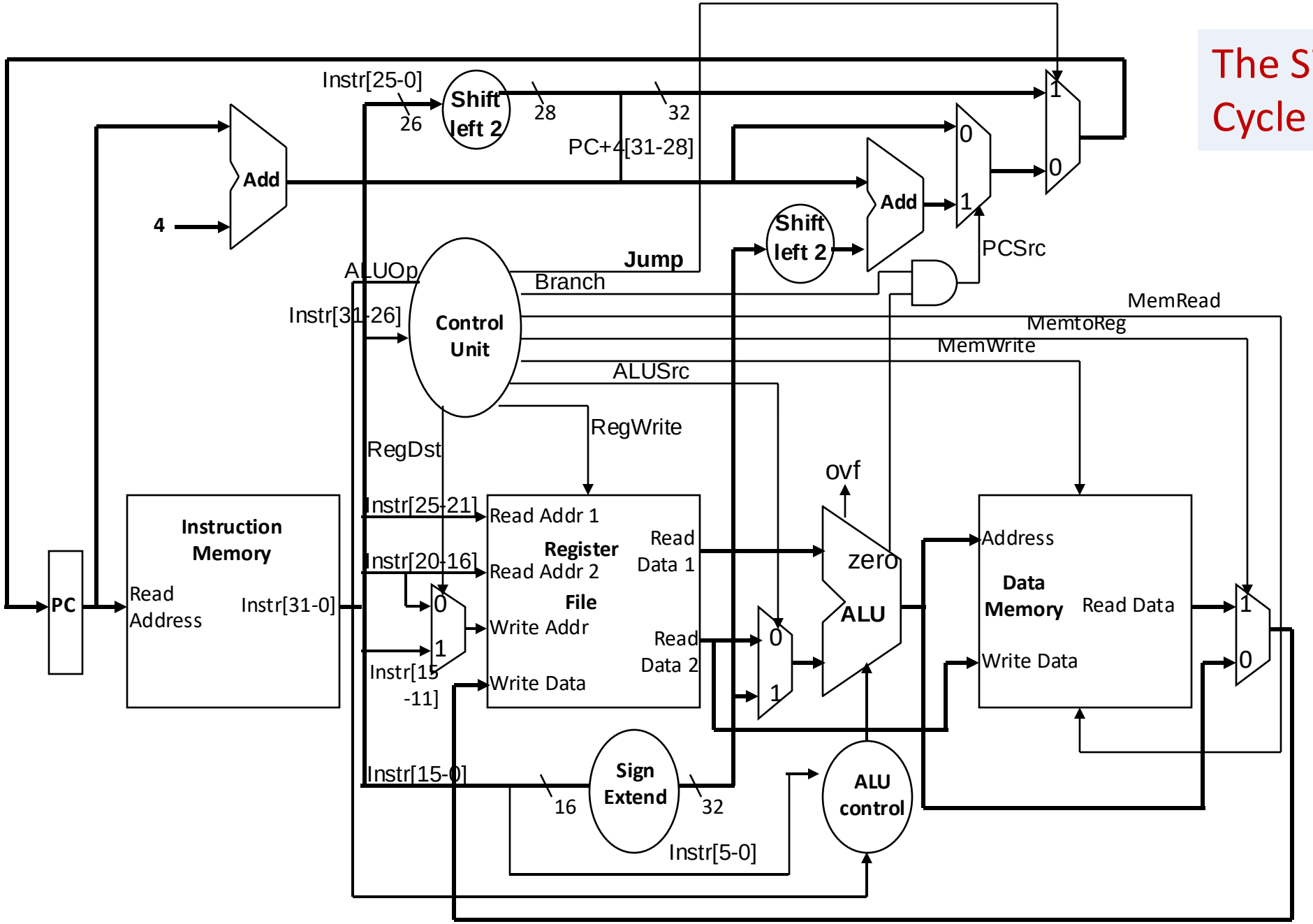
Tutorial #10

PIPELINING

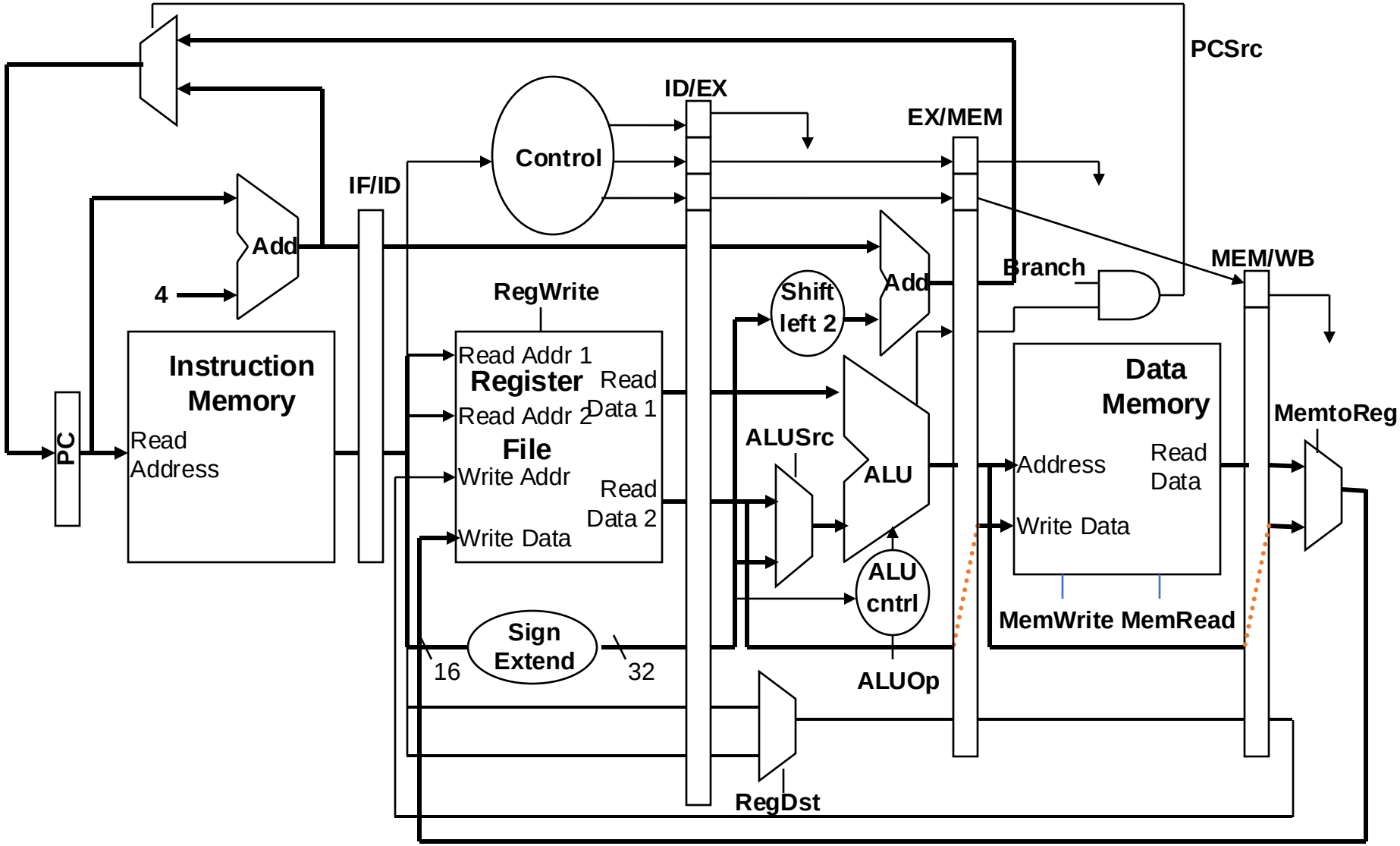
(Prepared by: Aaron Tan)

DATAPATH

The Single Cycle Datapath



Pipelined Datapath



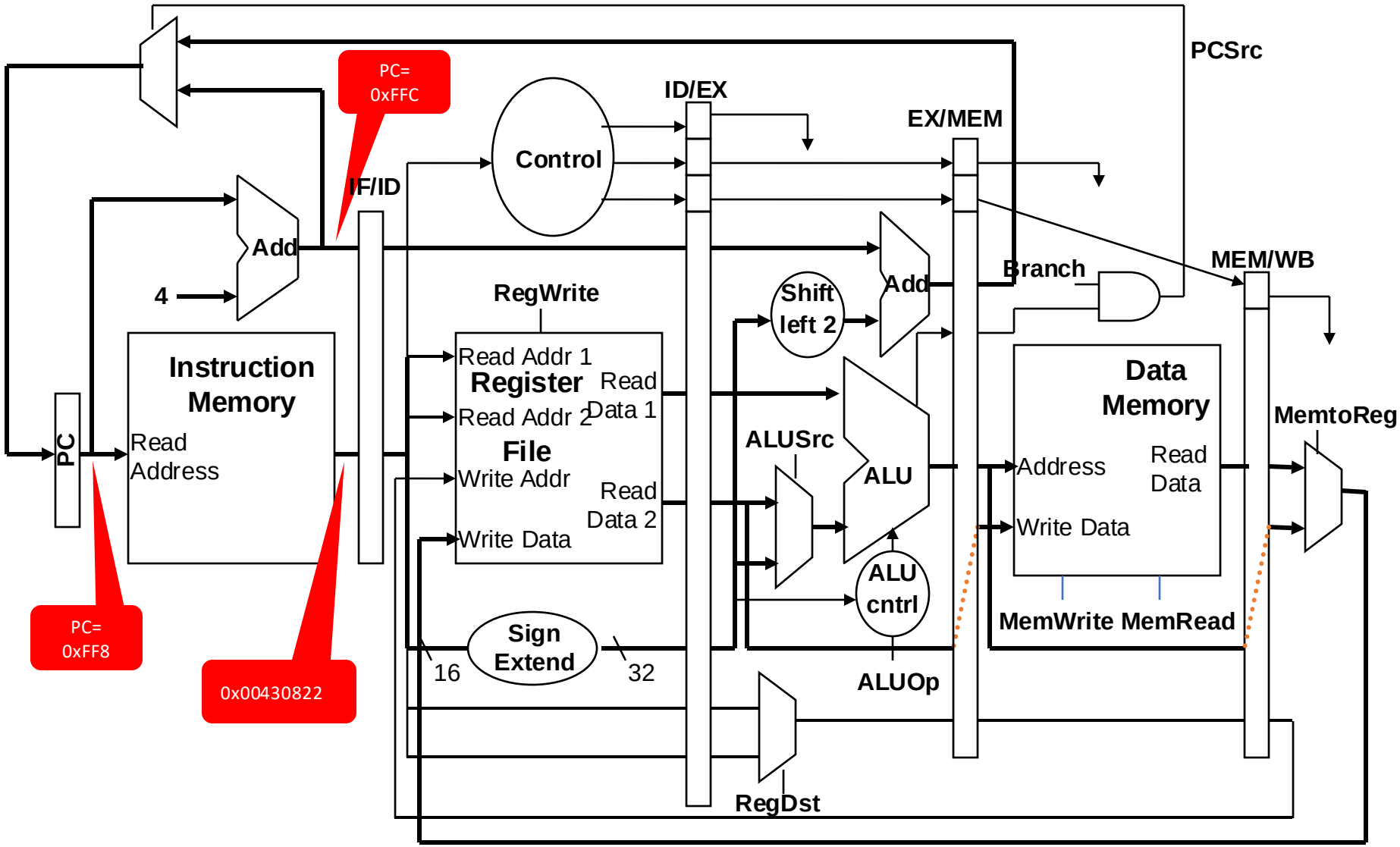
Pipeline Control: Grouping

- Group control signals according to pipeline stage

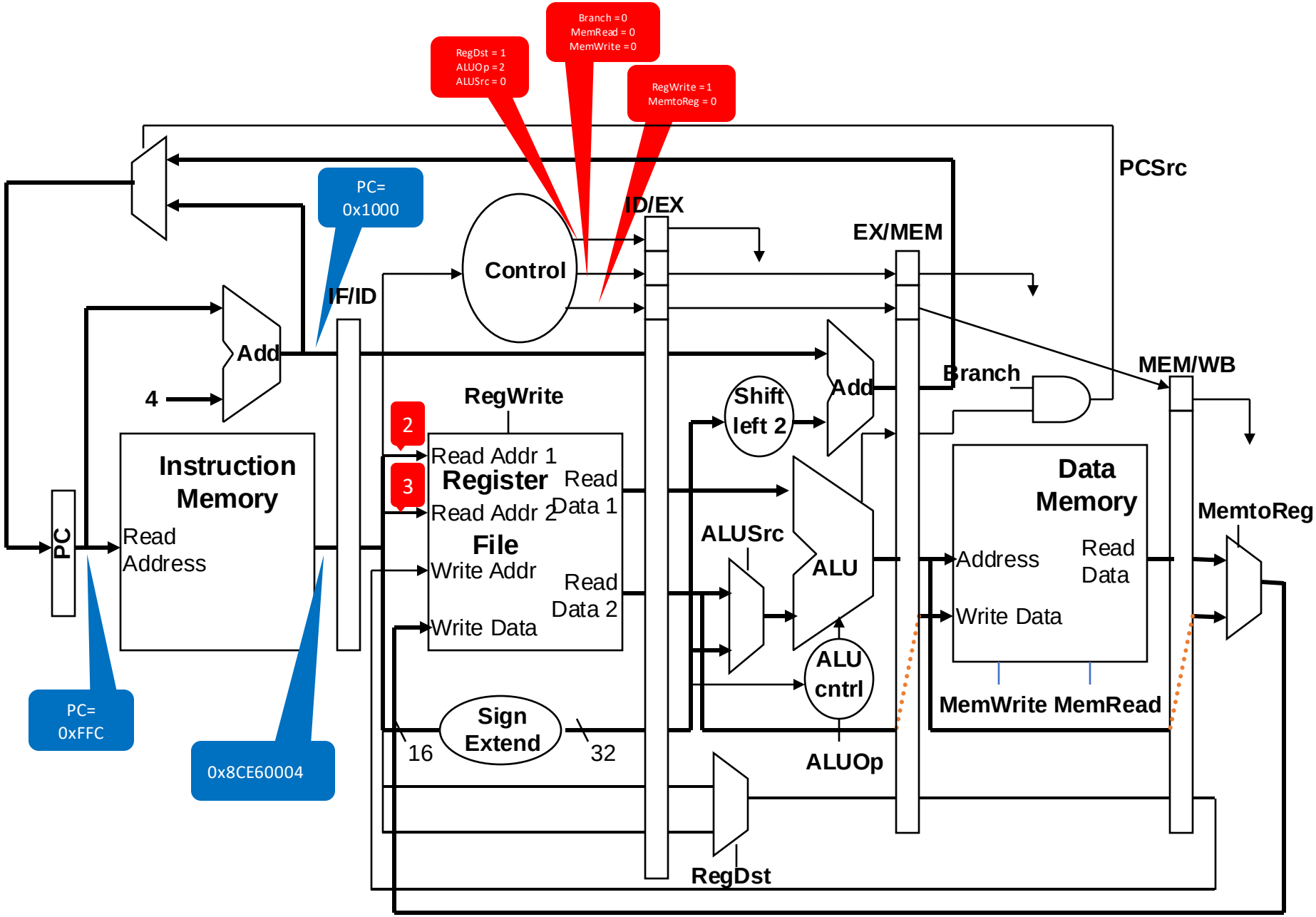
	RegDst	ALUSrc	MemTo Reg	Reg Write	Mem Read	Mem Write	Branch	ALUop	
								op1	op0
R-type	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

	EX Stage				MEM Stage			WB Stage	
	RegDst	ALUSrc	ALUop		Mem Read	Mem Write	Branch	MemTo Reg	Reg Write
			op1	op0					
R-type	1	0	1	0	0	0	0	0	1
lw	0	1	0	0	1	0	0	1	1
sw	X	1	0	0	0	1	0	X	0
beq	X	0	0	1	0	0	1	X	0

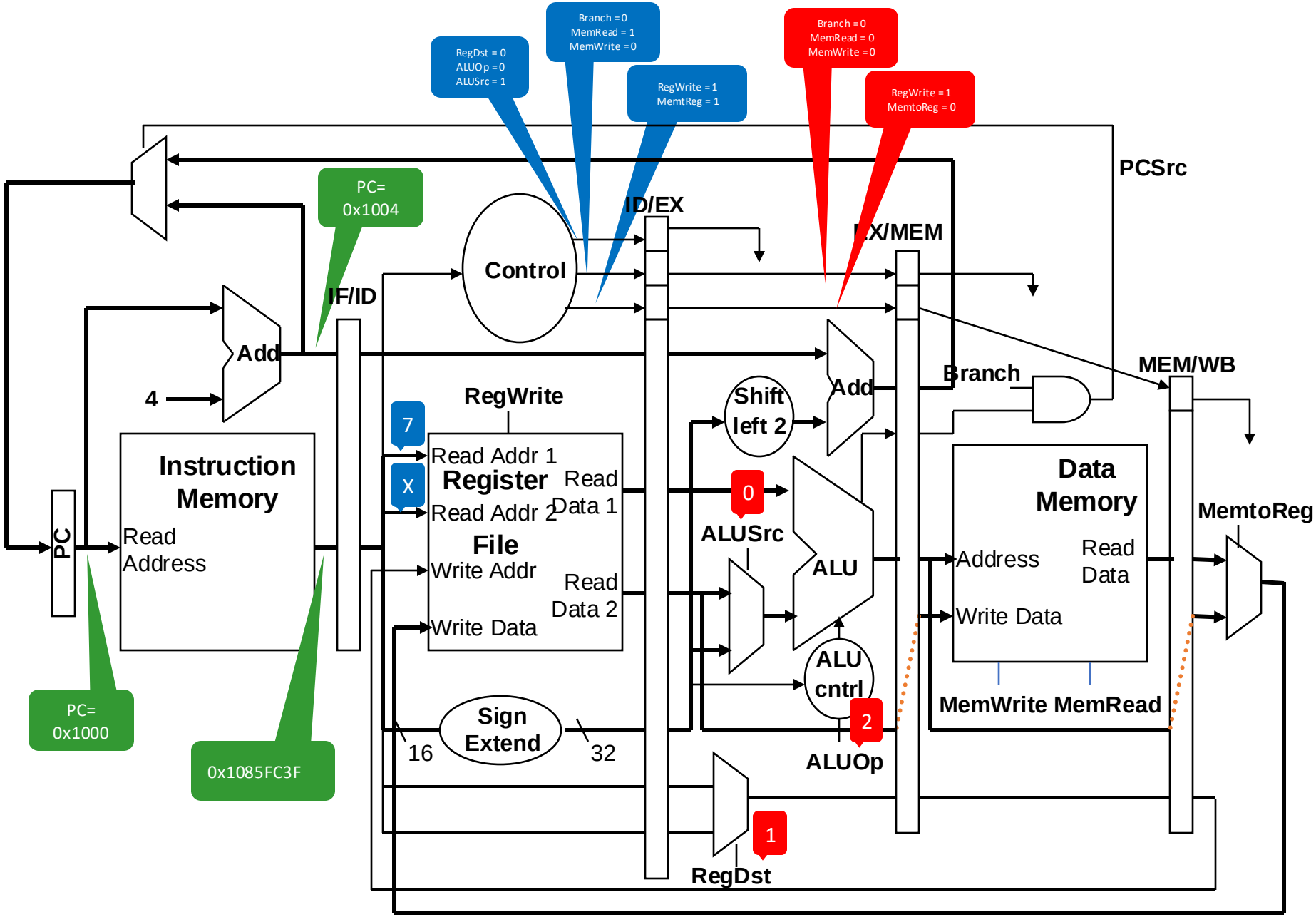
QUESTION 1


PCSrc
PC= 0xFFC
ID/EX
EX/MEM
Control
IF/ID
Add
4
RegWrite
Shift left 2
Add
Branch
MEM/WB
Instruction Memory
Read Address
Read Addr 1
Register File
Read Addr 2
Write Addr
Write Data
Read Data 1
Read Data 2
ALUSrc
ALU
Data Memory
Address
Read Data
Write Data
MemtoReg
PC
PC= 0xFF8
0x00430822
Sign Extend
16
32
ALU cntrl
ALUOp
MemWrite
MemRead
RegDst

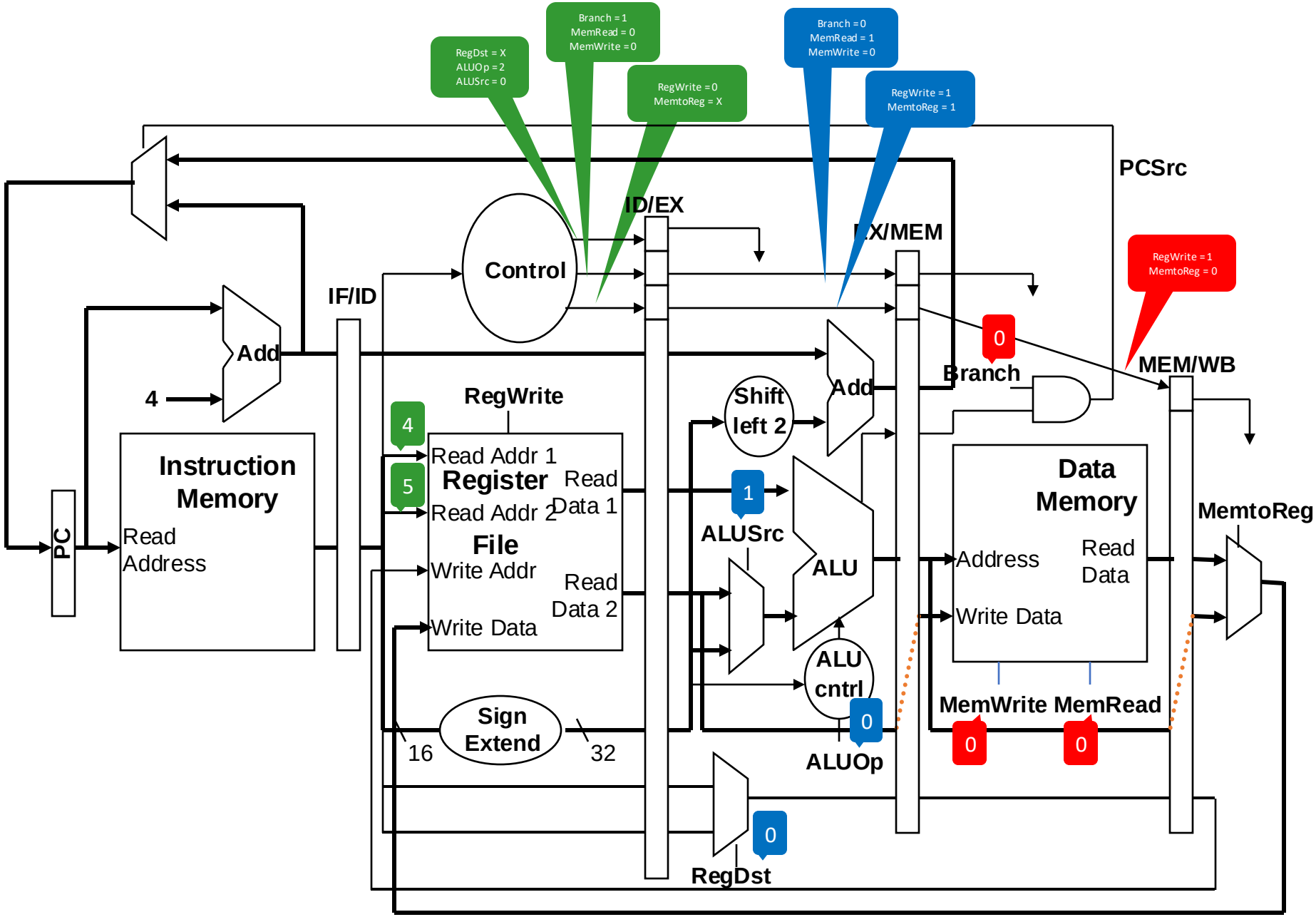
```
sub $1, $2, $3
```



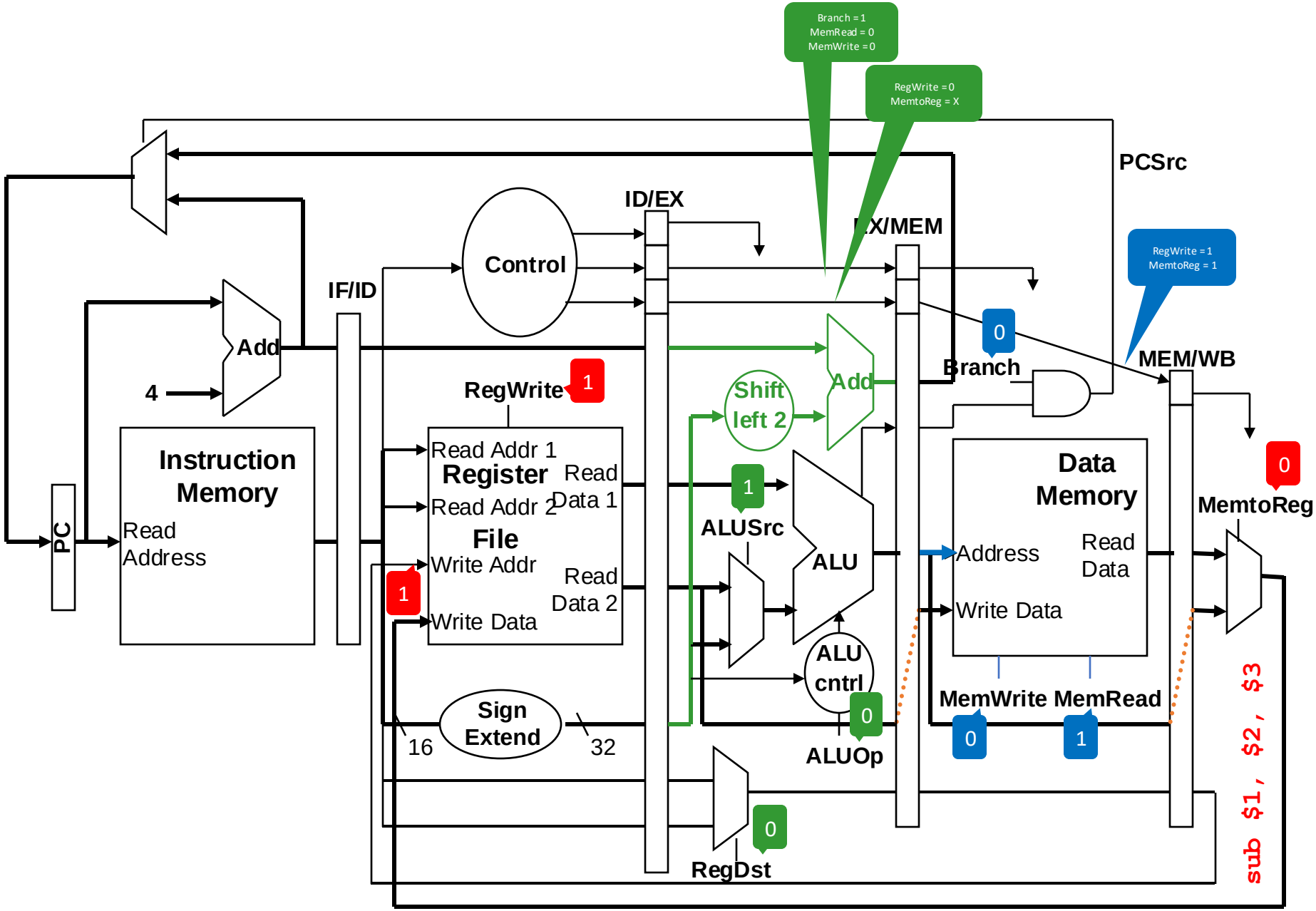

```
lw $6, 4($7)        sub $1, $2, $3
```

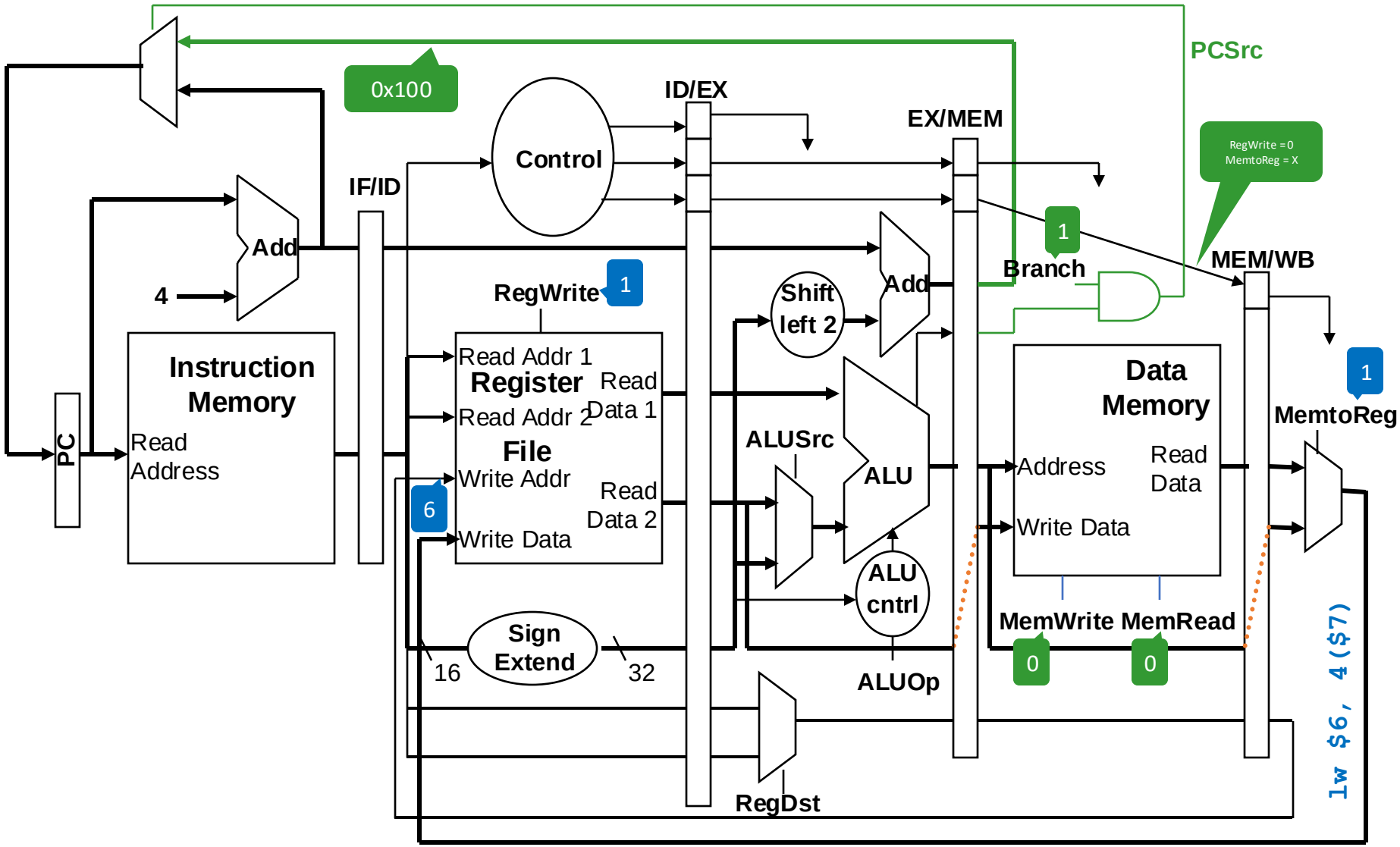


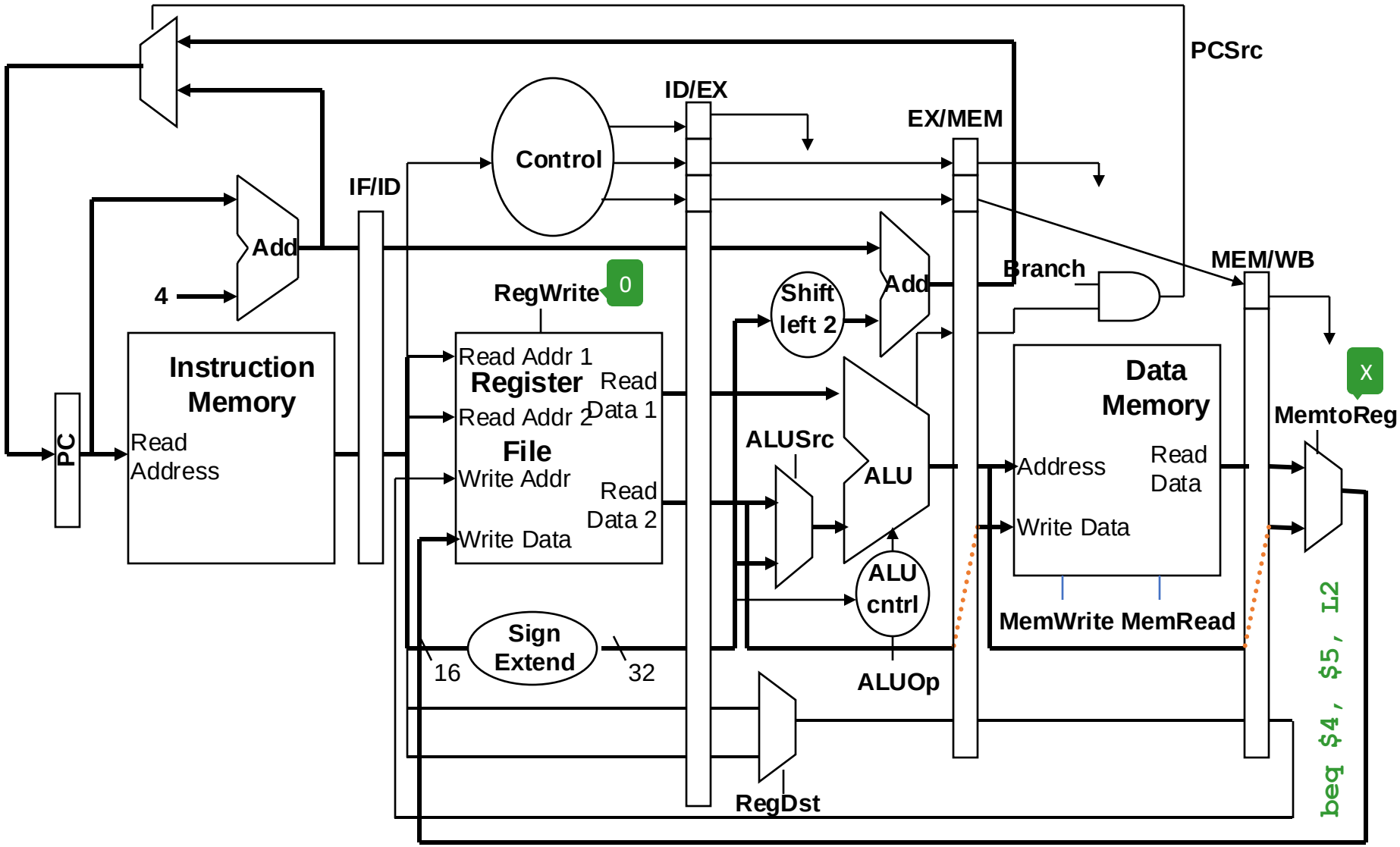

```
beq $4, $5, L2        lw $6, 4($7)    sub $1, $2, $3
```




```
beq $4, $5, L2      lw $6, 4($7)      sub $1, $2, $3
```

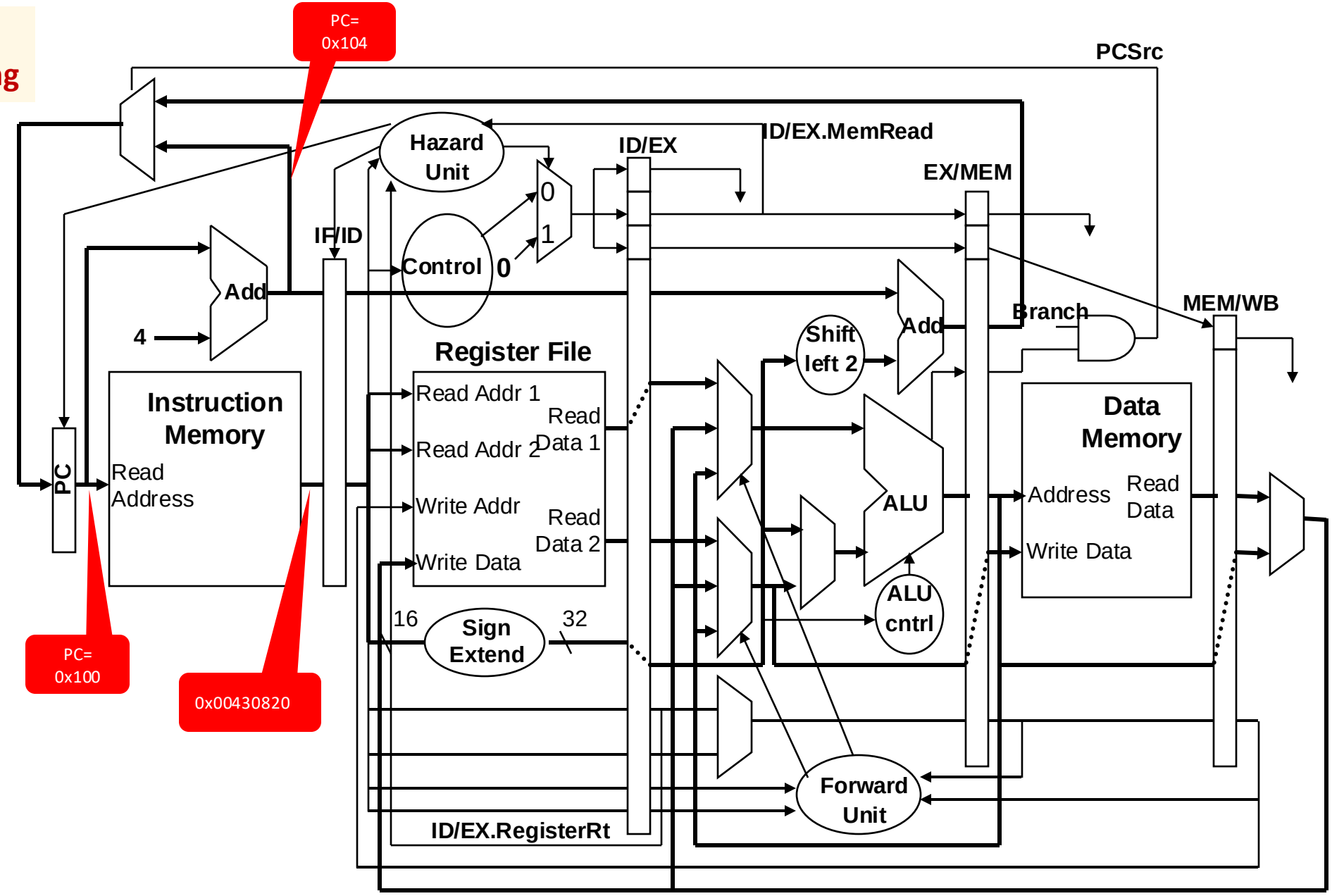

Branch = 1
MemRead = 0
MemWrite = 0
RegWrite = 0
MemtoReg = X
PCSrc
ID/EX
EX/MEM
RegWrite = 1
MemtoReg = 1
Control
IF/ID
Add
4
0
Branch
MEM/WB
RegWrite
1
Shift
left 2
Add
Instruction
Memory
Read
Address
PC
Read Addr 1
Register
Read
Data 1
Read Addr 2
File
Write Addr
Read
Data 2
Write Data
1
1
ALUSrc
ALU
Data
Memory
Address
Read
Data
Write Data
0
MemtoReg
ALU
cntrl
Sign
Extend
16
32
0
ALUOp
MemWrite
MemRead
0
1
0
RegDst
sub $1, $2, $3
beq $4, $5, L2
lw $6, 4($7)


PCSrc
0x100
ID/EX
EX/MEM
Control
IF/ID
RegWrite =0
MemtoReg = X
Add
4
RegWrite
1
Shift
left 2
Add
Branch
1
MEM/WB
Instruction
Memory
Read
Address
PC
Read Addr 1
Register
Read Addr 2
File
Write Addr
Write Data
Read
Data 1
Read
Data 2
6
ALUSrc
ALU
Data
Memory
Address
Read
Data
Write Data
1
MemtoReg
ALU
cntrl
Sign
Extend
16
32
ALUOp
MemWrite
MemRead
0
0
RegDst
lw $6, 4($7)
beq $4, $5, L2


PCSrc
ID/EX
EX/MEM
MEM/WB
Control
IF/ID
Add
4
RegWrite
0
Shift
left 2
Add
Branch
Instruction
Memory
Read
Address
PC
Register
File
Read Addr 1
Read Addr 2
Write Addr
Write Data
Read
Data 1
Read
Data 2
ALUSrc
ALU
ALU
cntrl
Data
Memory
Address
Read
Data
Write Data
MemWrite
MemRead
MemtoReg
X
Sign
Extend
16
32
ALUOp
RegDst
beq $4, $5, L2

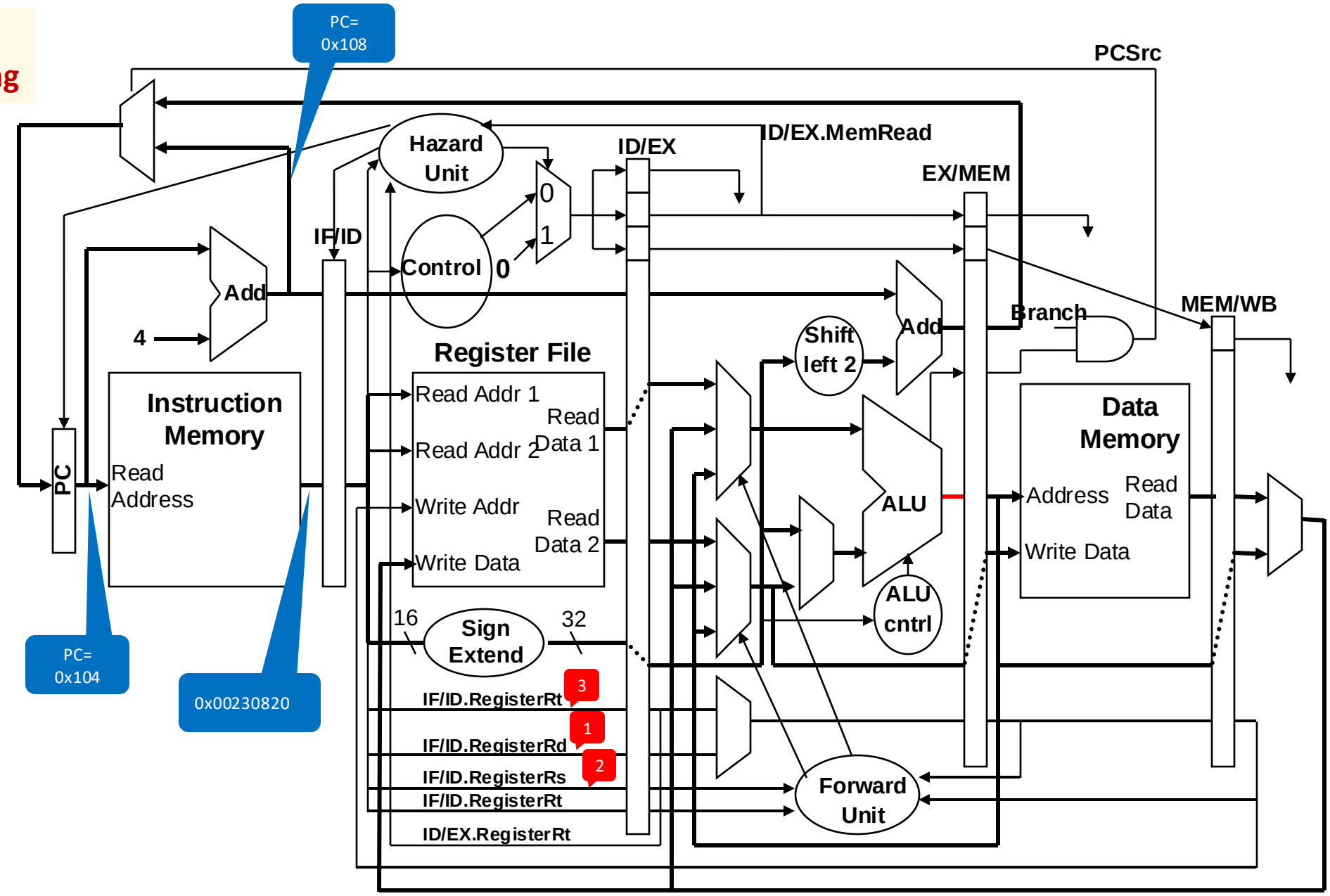
QUESTION 2

Datapath With Forwarding




```
add $1, $2, $3
```

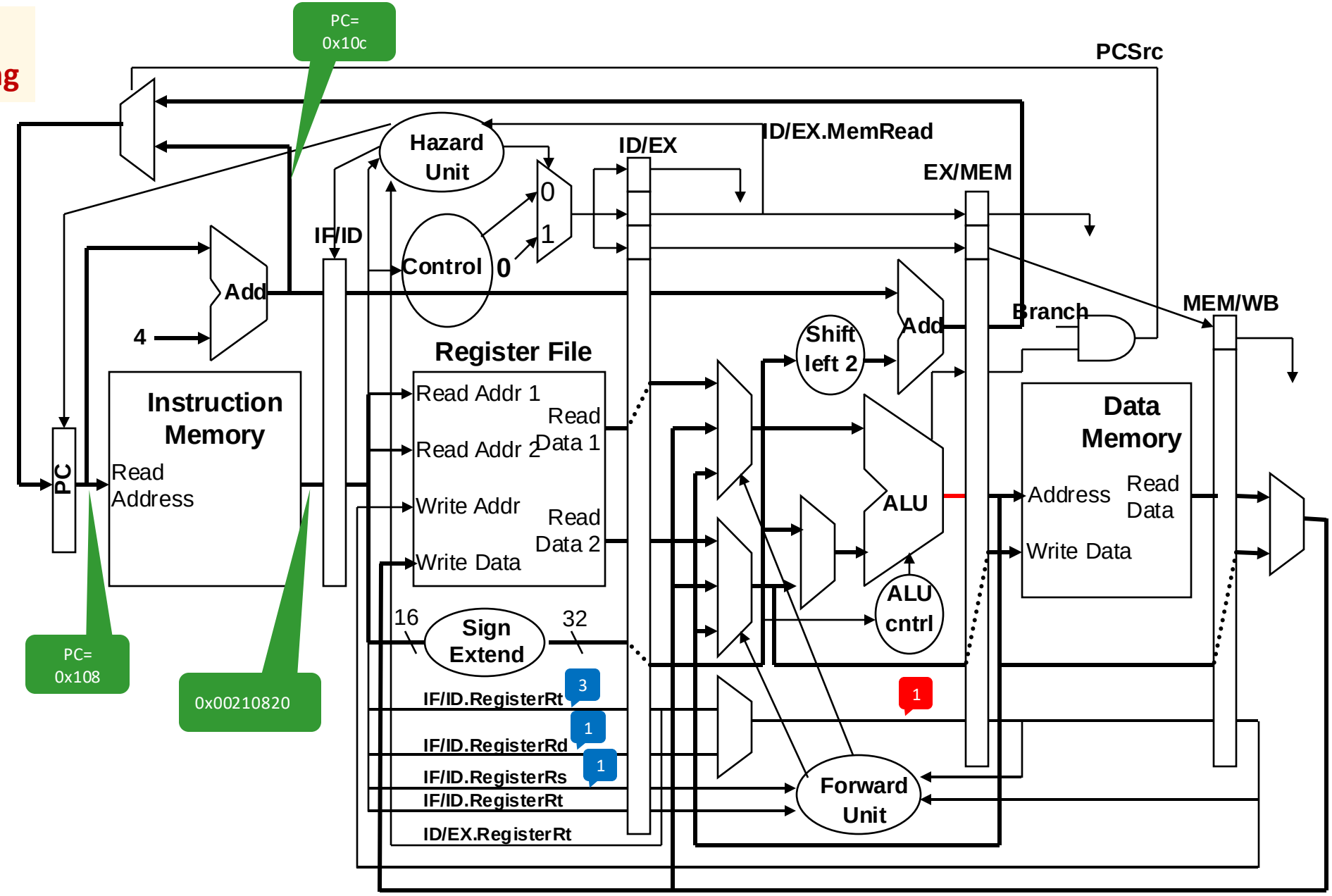
Datapath With Forwarding



`add $1, $1, $3`

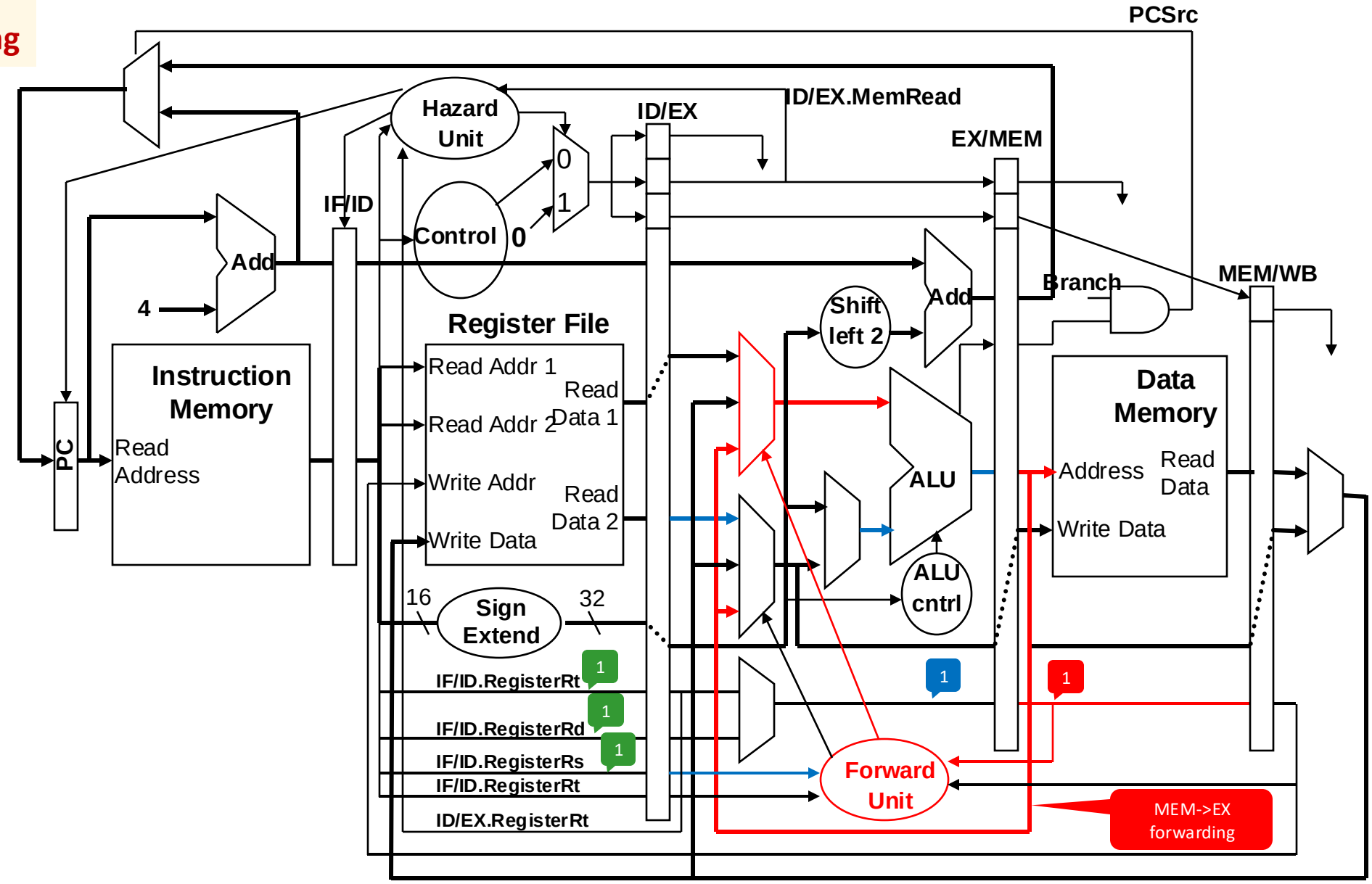
`add $1, $2, $3`

Datapath With Forwarding

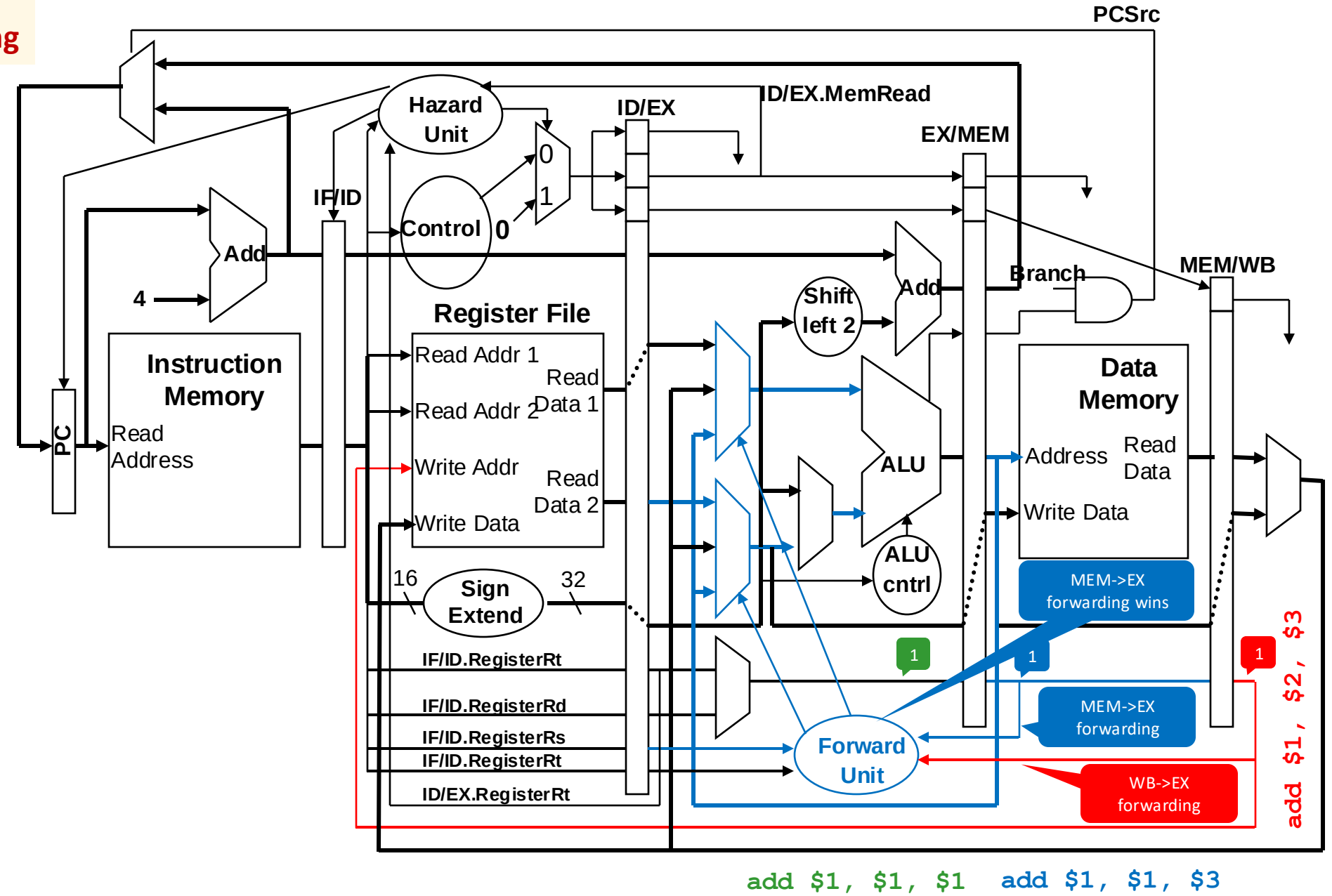



```
add $1, $1, $1      add $1, $1, $3      add $1, $2, $3
```

Datapath With Forwarding

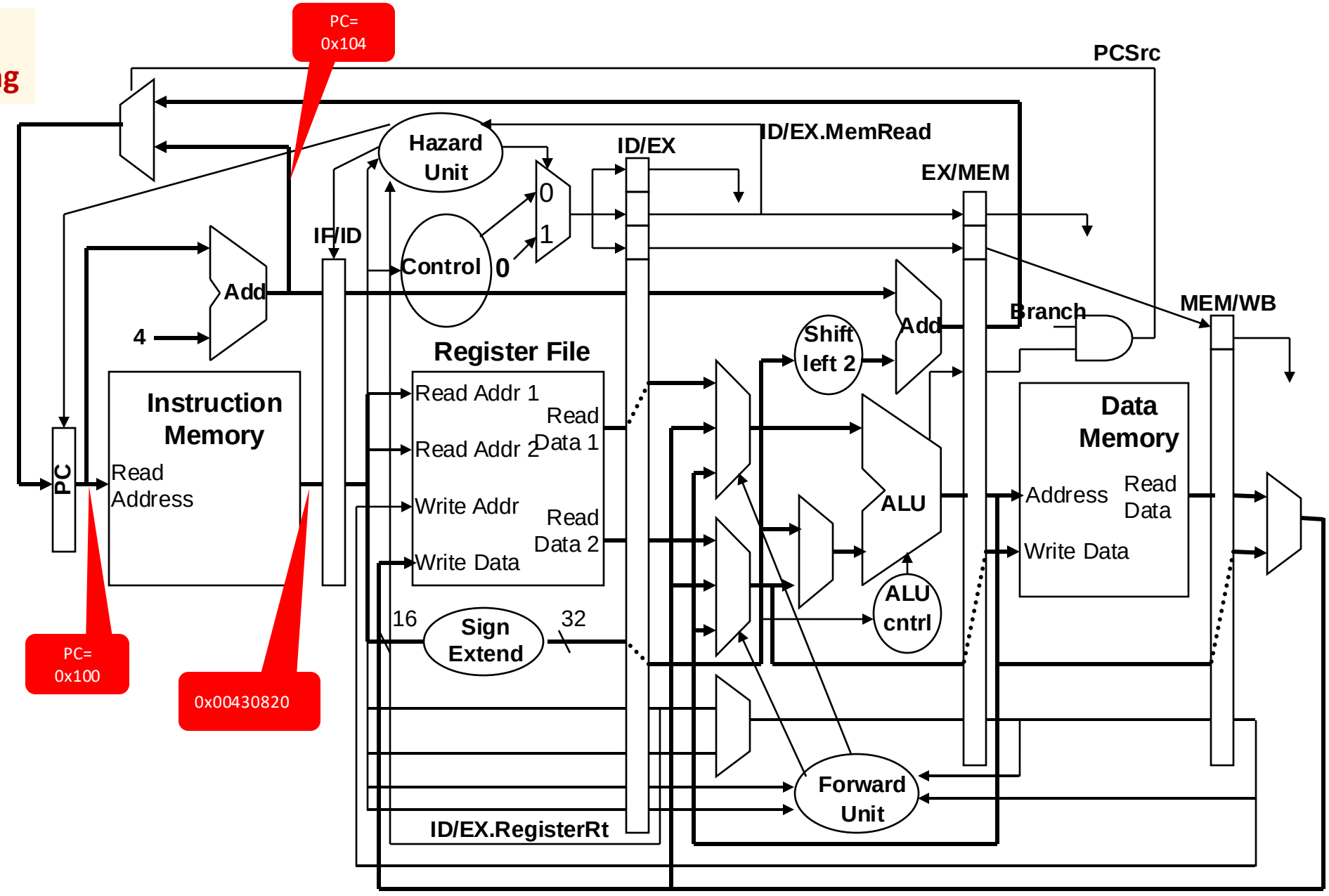


Datapath With Forwarding



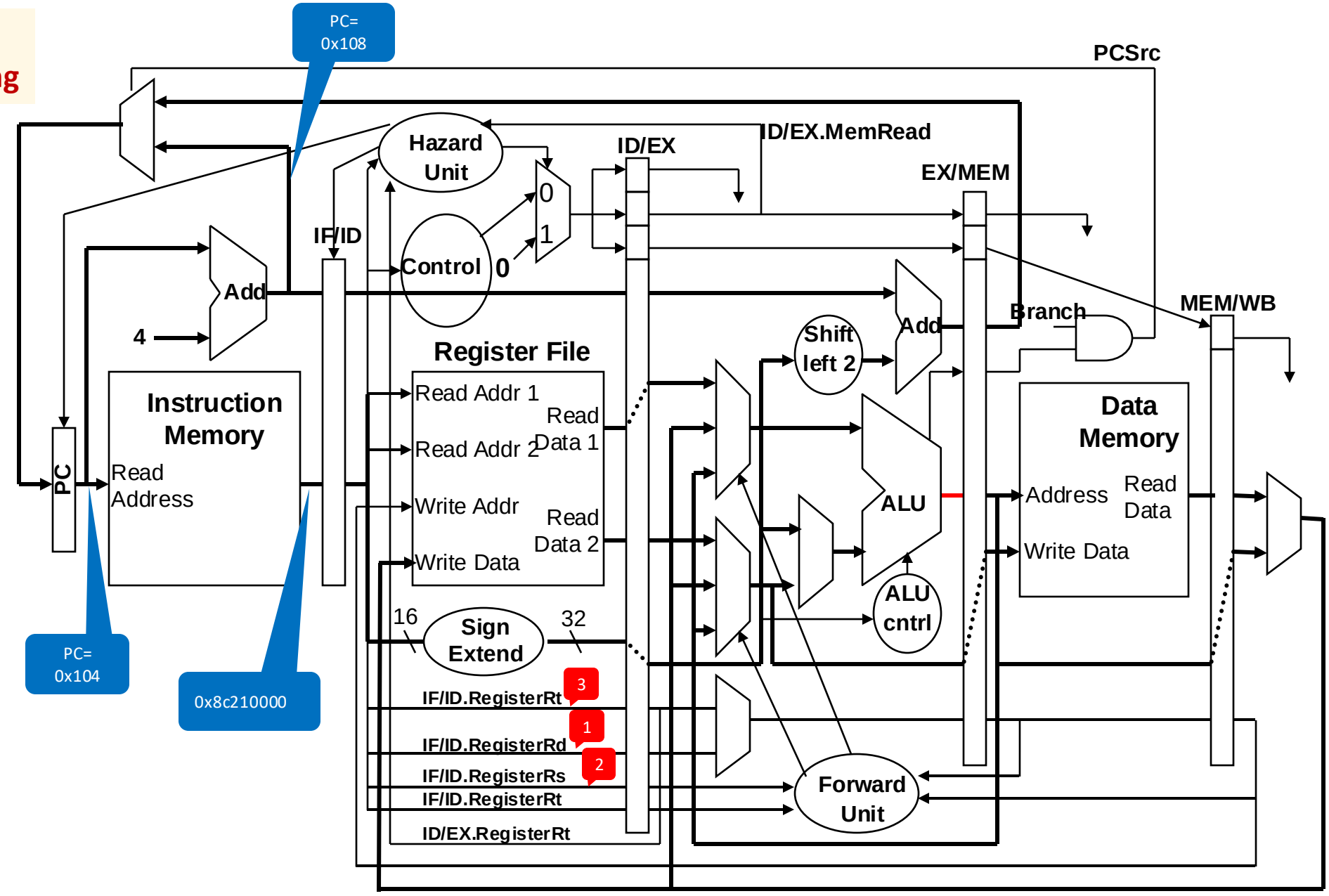
QUESTION 3

Datapath With Forwarding




```
add $1, $2, $3
```

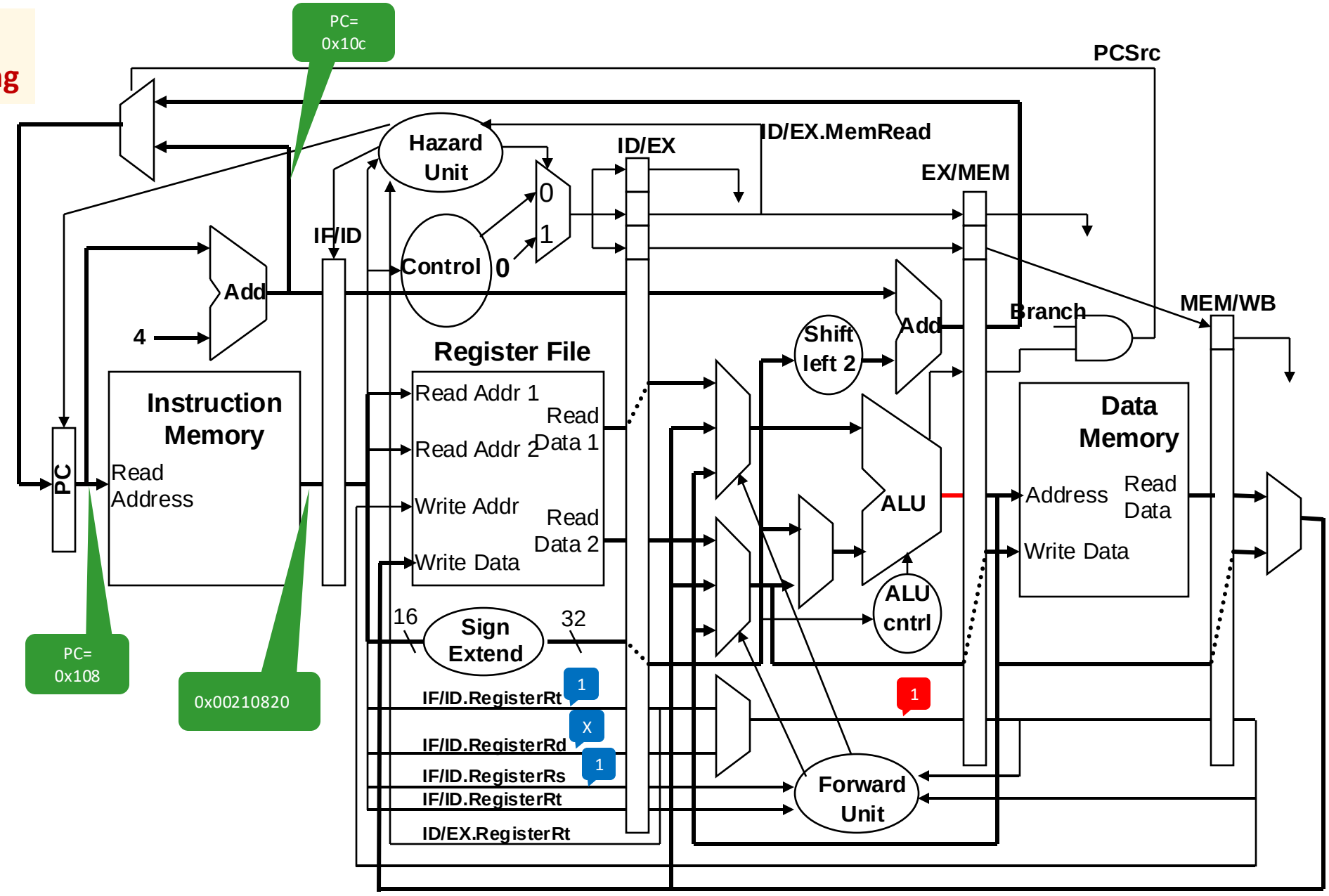
Datapath With Forwarding




```
lw $1, 0($1)
```

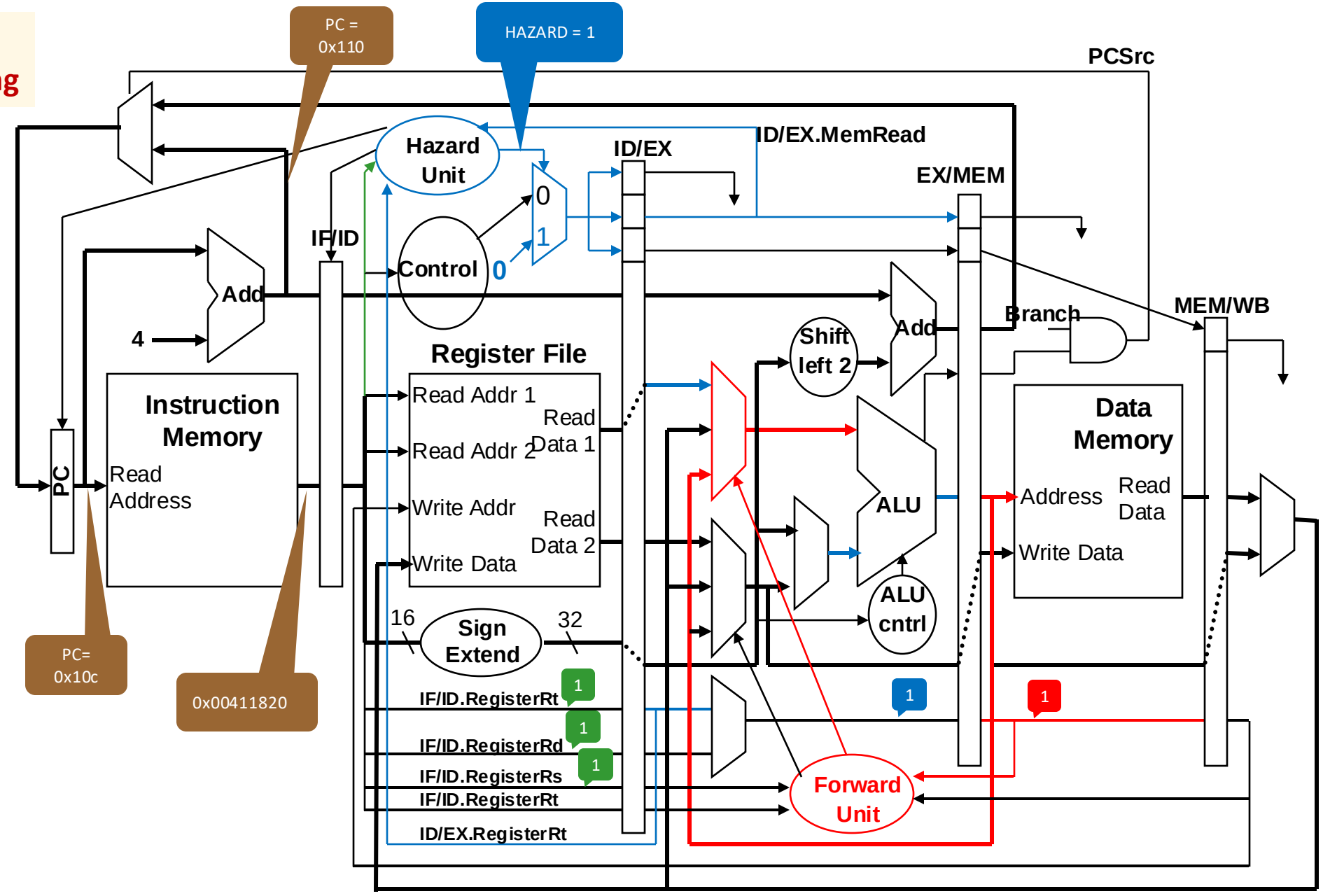
```
add $1, $2, $3
```

Datapath With Forwarding



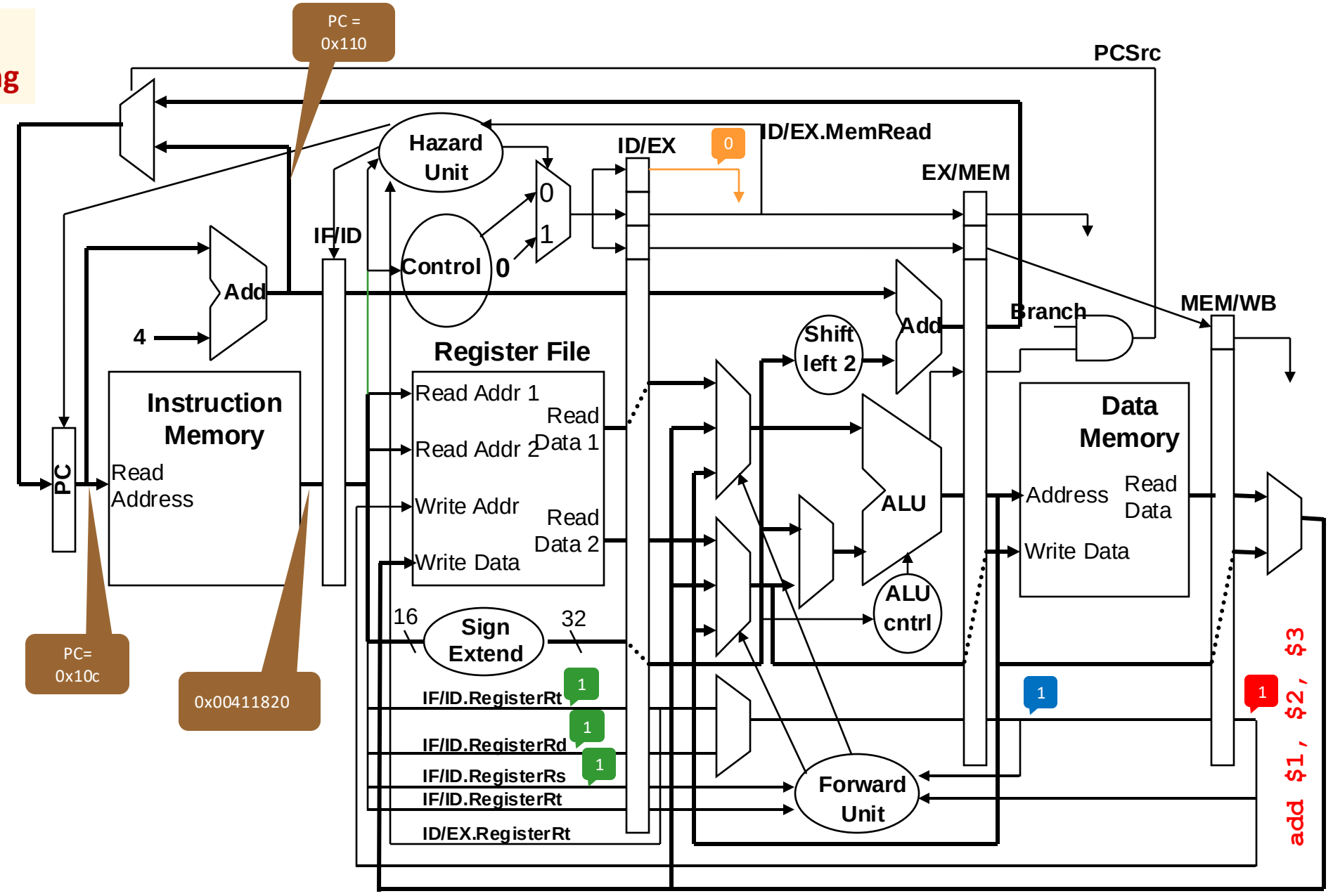

```
add $1, $1, $1        lw $1, 0($1)        add $1, $2, $3
```

Datapath With Forwarding



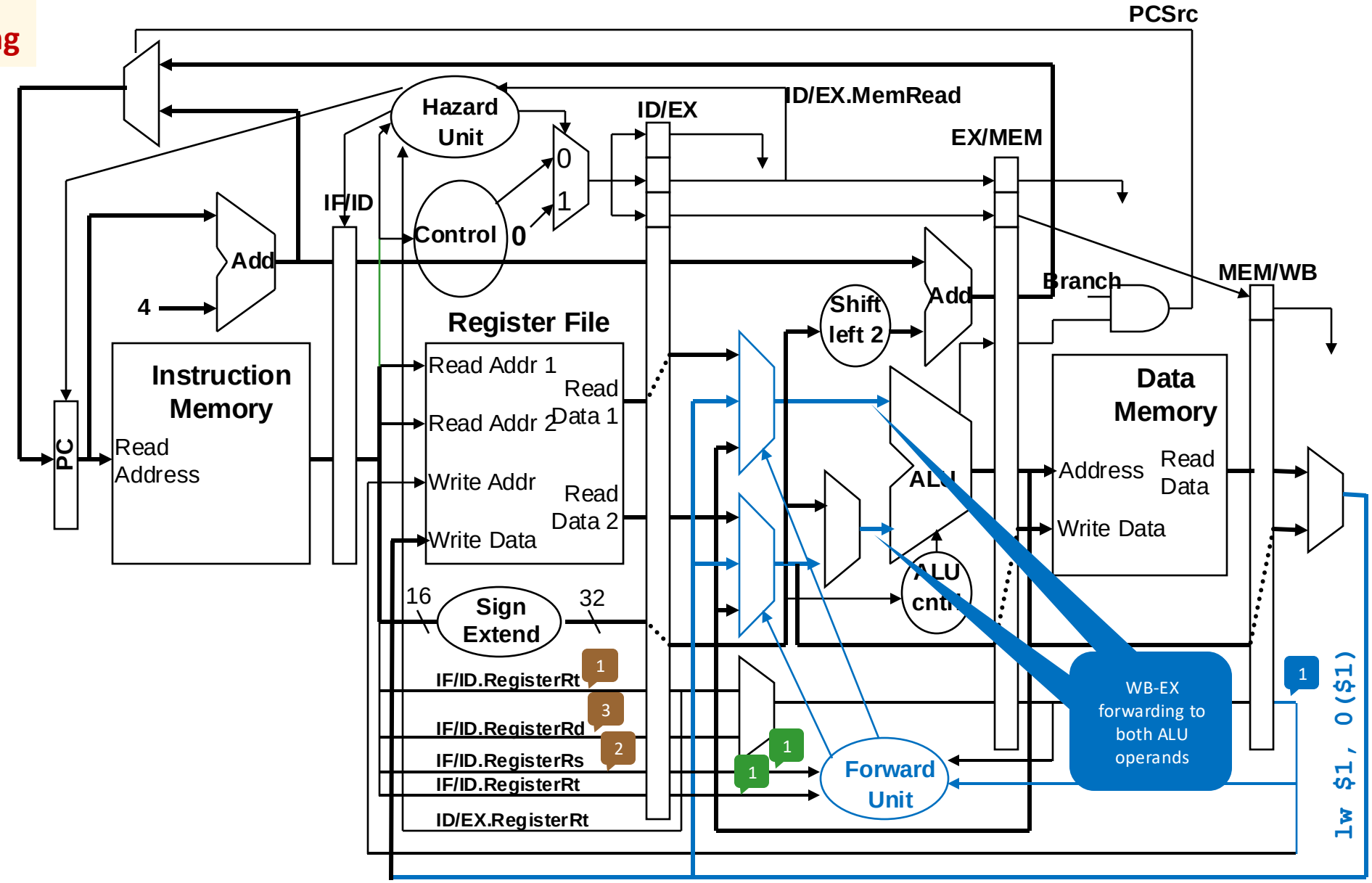
`add $3, $2, $1` `add $1, $1, $1` `lw $1, 0($1)` `add $1, $2, $3`

Datapath With Forwarding

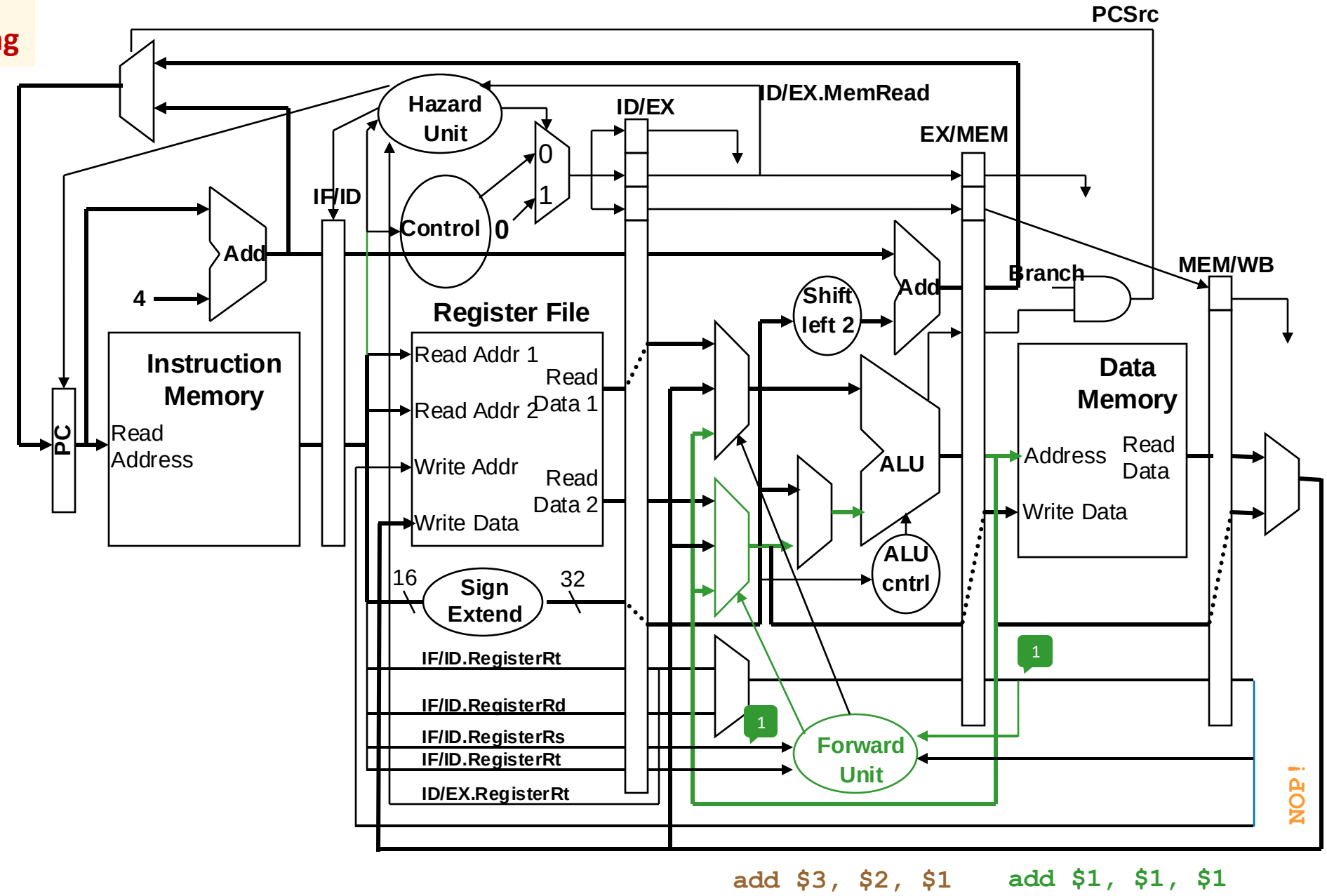


`add $3, $2, $1` `add $1, $1, $1` `NOP!` `lw $1, 0($1)`

Datapath With Forwarding

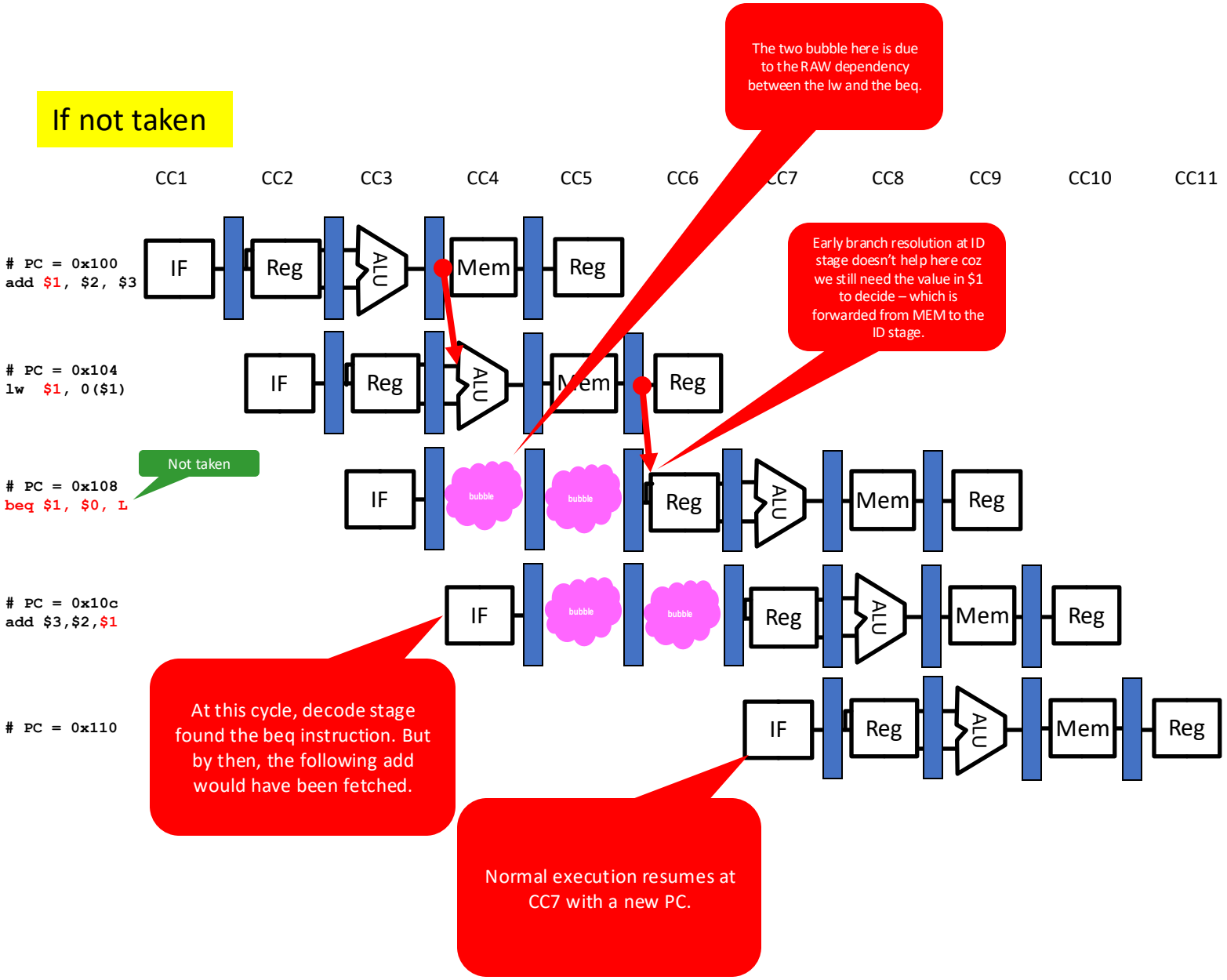


Datapath With Forwarding



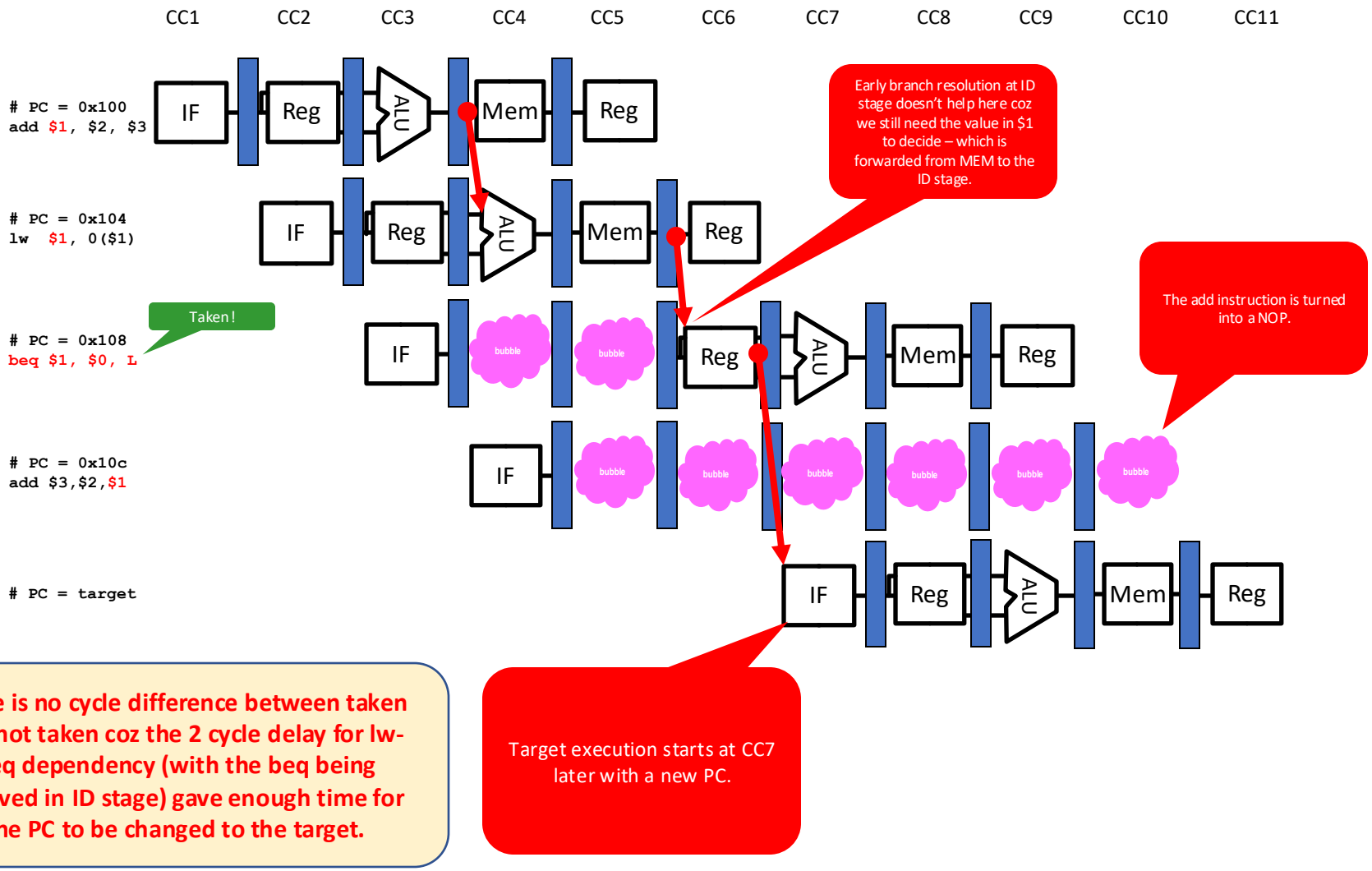
QUESTION 4

Q4a


If not taken
CC1
CC2
CC3
CC4
CC5
CC6
CC7
CC8
CC9
CC10
CC11
# PC = 0x100
add $1, $2, $3
# PC = 0x104
lw $1, 0($1)
# PC = 0x108
beq $1, $0, L
# PC = 0x10c
add $3,$2,$1
# PC = 0x110
IF
Reg
ALU
Mem
Reg
bubble
Not taken
The two bubble here is due to the RAW dependency between the lw and the beq.
Early branch resolution at ID stage doesn't help here coz we still need the value in $1 to decide – which is forwarded from MEM to the ID stage.
At this cycle, decode stage found the beq instruction. But by then, the following add would have been fetched.
Normal execution resumes at CC7 with a new PC.

Q4b

If taken



There is no cycle difference between taken and not taken coz the 2 cycle delay for lw-beq dependency (with the beq being resolved in ID stage) gave enough time for the PC to be changed to the target.

END OF FILE