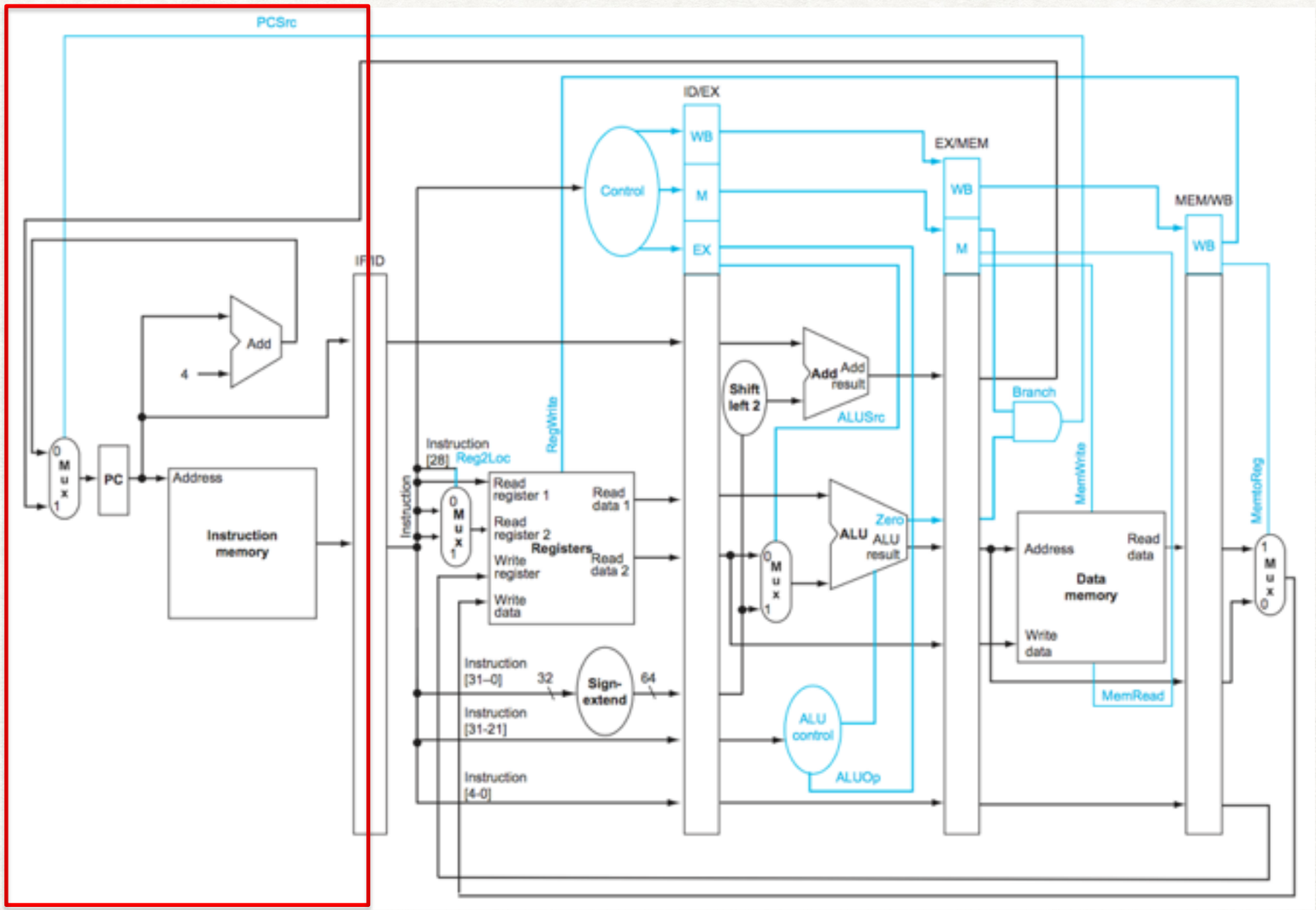
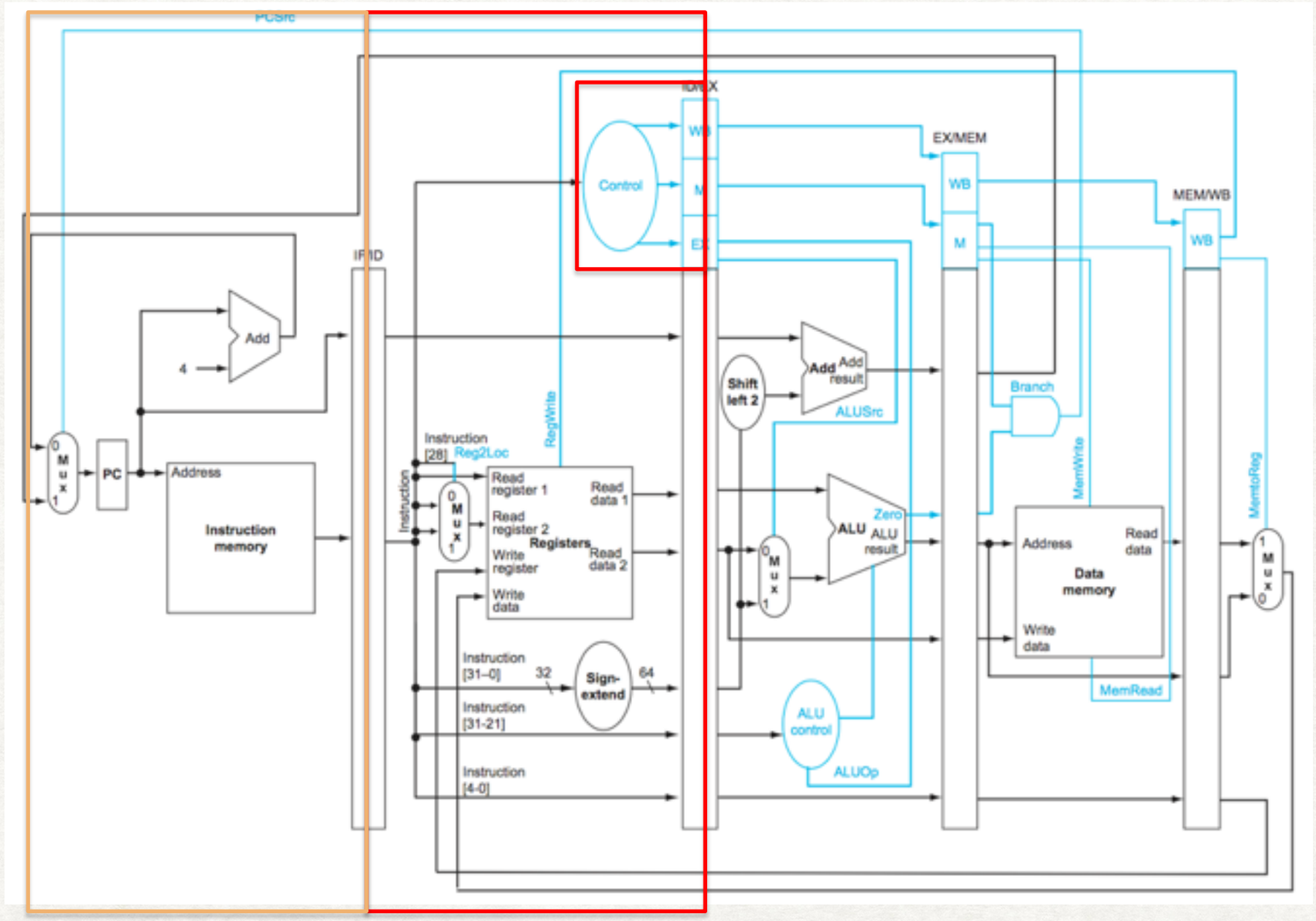
CDA 3101

PIPELINING

LDUR


PCSrc
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Control
IF/ID
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction [28]
Reg2Loc
RegWrite
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Instruction [31–0]
32
Sign-extend
64
Instruction [31-21]
Instruction [4-0]
Shift left 2
Add
Add result
ALUSrc
Zero
ALU
ALU result
ALU control
ALUOp
Branch
MemWrite
MemRead
MemtoReg
Address
Read data
Data memory
Write data

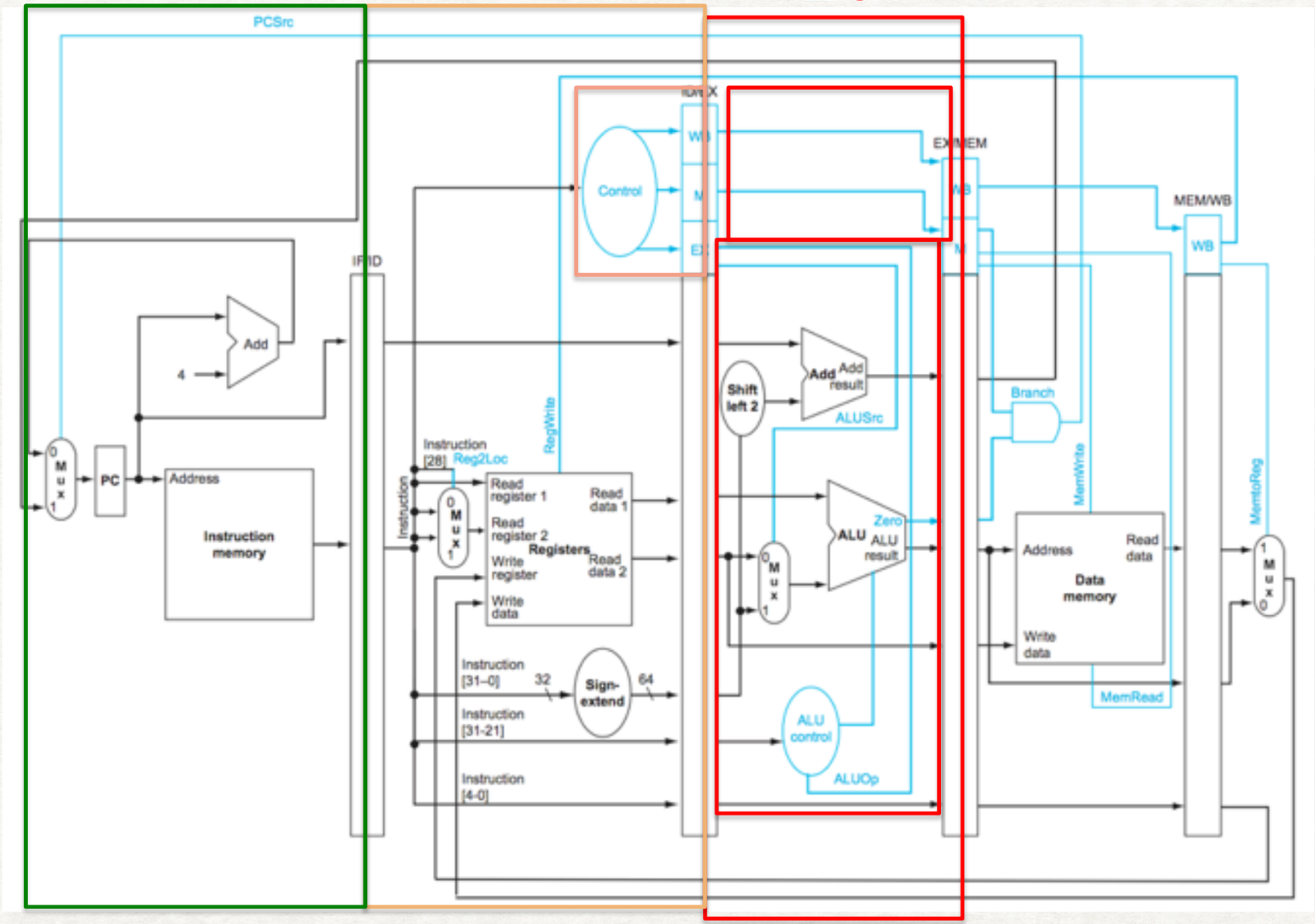
SUB LDUR

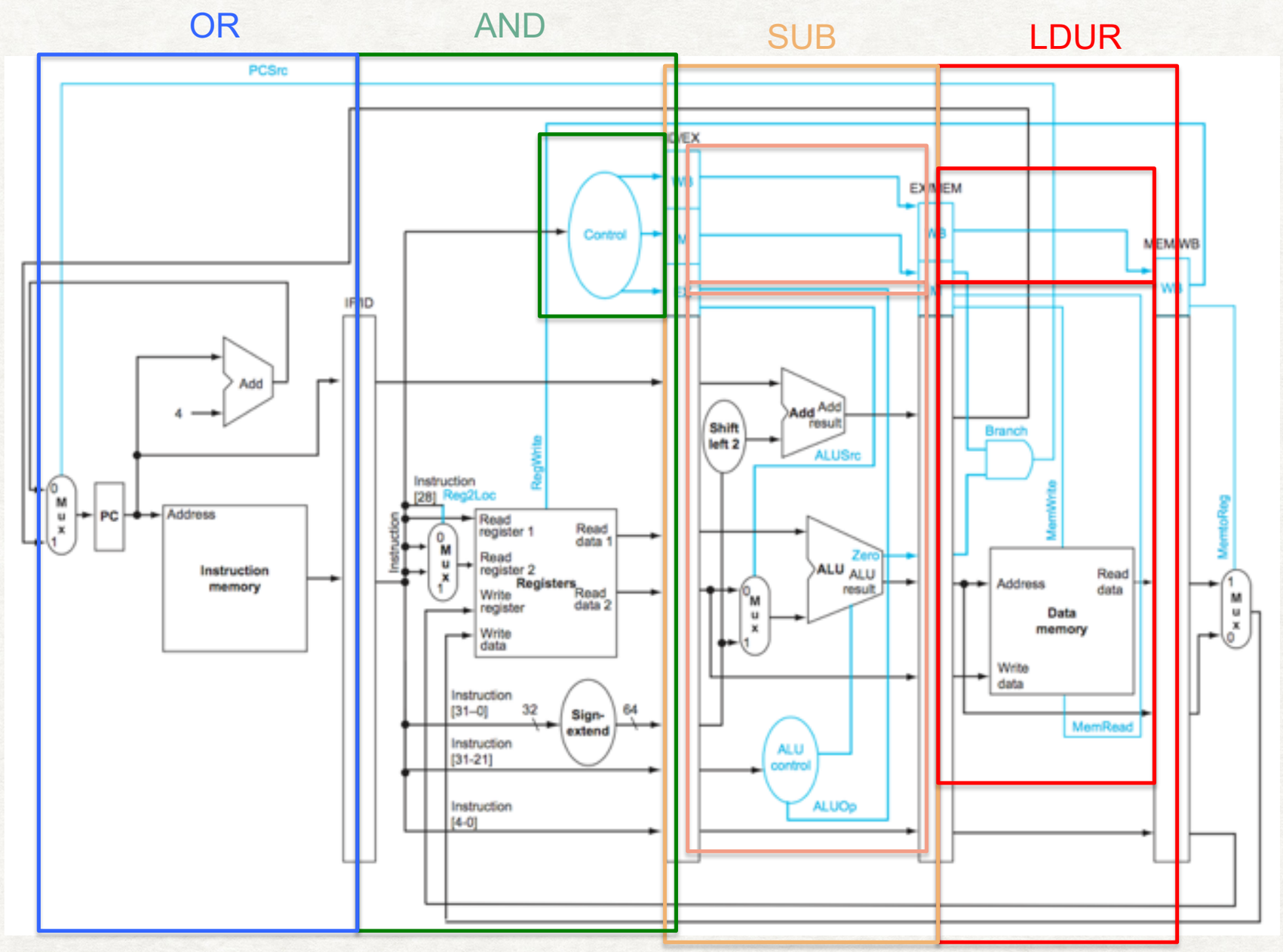

PCSrc
IF/ID
ID/EX
EX/MEM
MEM/WB
Control
WB
M
EX
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
[28]
Reg2Loc
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Instruction [31–0]
32
Sign-extend
64
Instruction [31-21]
Instruction [4-0]
Shift left 2
Add
Add result
ALUSrc
ALU
Zero
ALU result
ALU control
ALUOp
Branch
MemWrite
MemRead
MemtoReg
Address
Read data
Data memory
Write data

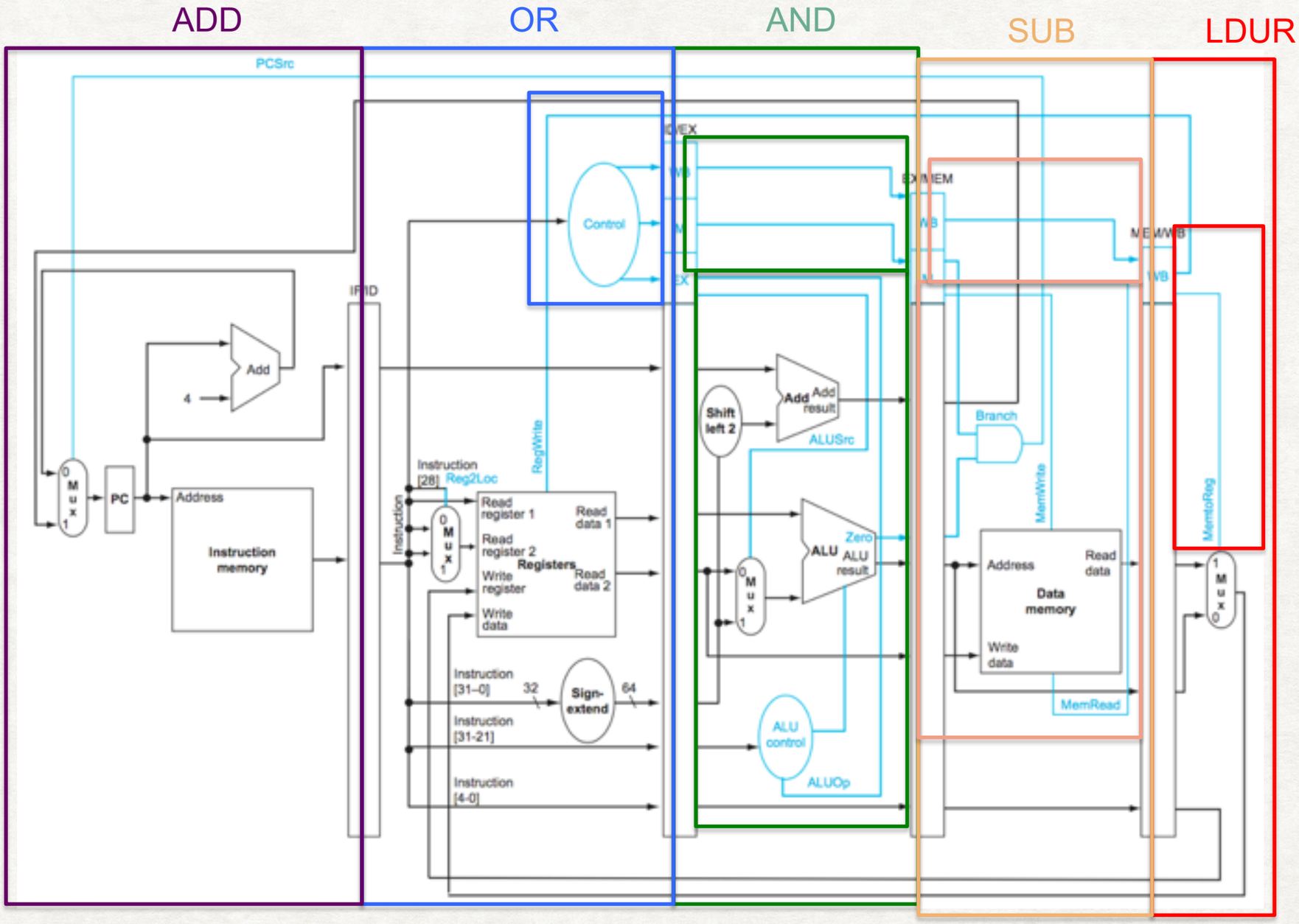
AND

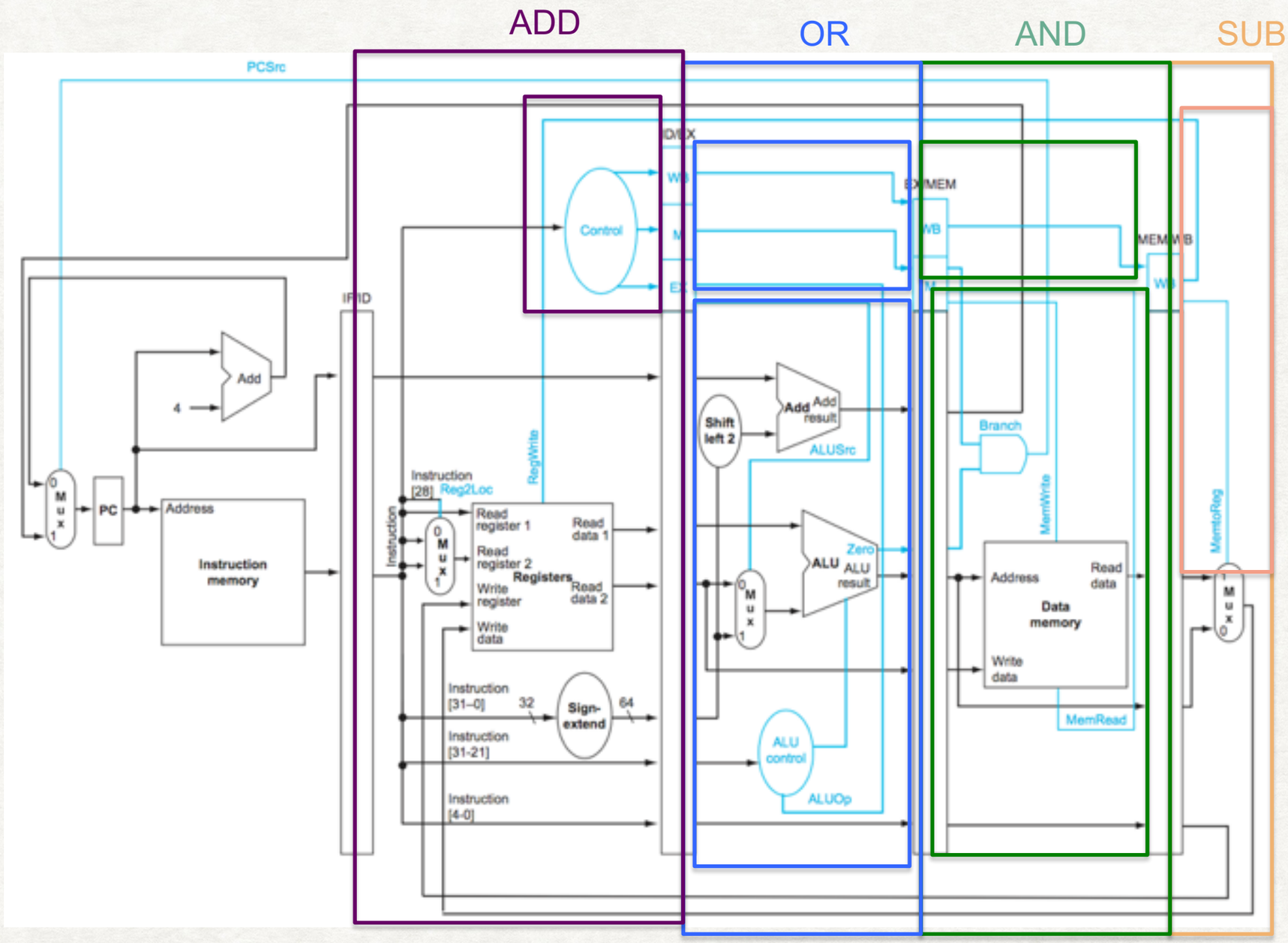
SUB

LDUR


PCSrc
Control
IF/ID
EX/MEM
MEM/WB
WB
M
EX
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
[28]
Reg2Loc
RegWrite
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Instruction [31–0]
32
Sign-extend
64
Instruction [31-21]
Instruction [4-0]
Shift left 2
Add
Add result
ALUSrc
Branch
ALU
Zero
ALU result
ALU control
ALUOp
MemWrite
MemRead
Address
Read data
Data memory
Write data
MemtoReg


OR
AND
SUB
LDUR
PCSrc
Control
IF/ID
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
[28]
Reg2Loc
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Instruction [31–0]
32
Sign-extend
64
Instruction [31-21]
Instruction [4-0]
Shift left 2
Add
Add result
ALUSrc
Zero
ALU
ALU result
ALU control
ALUOp
Branch
MemWrite
MemRead
MemtoReg
Address
Read data
Data memory
Write data

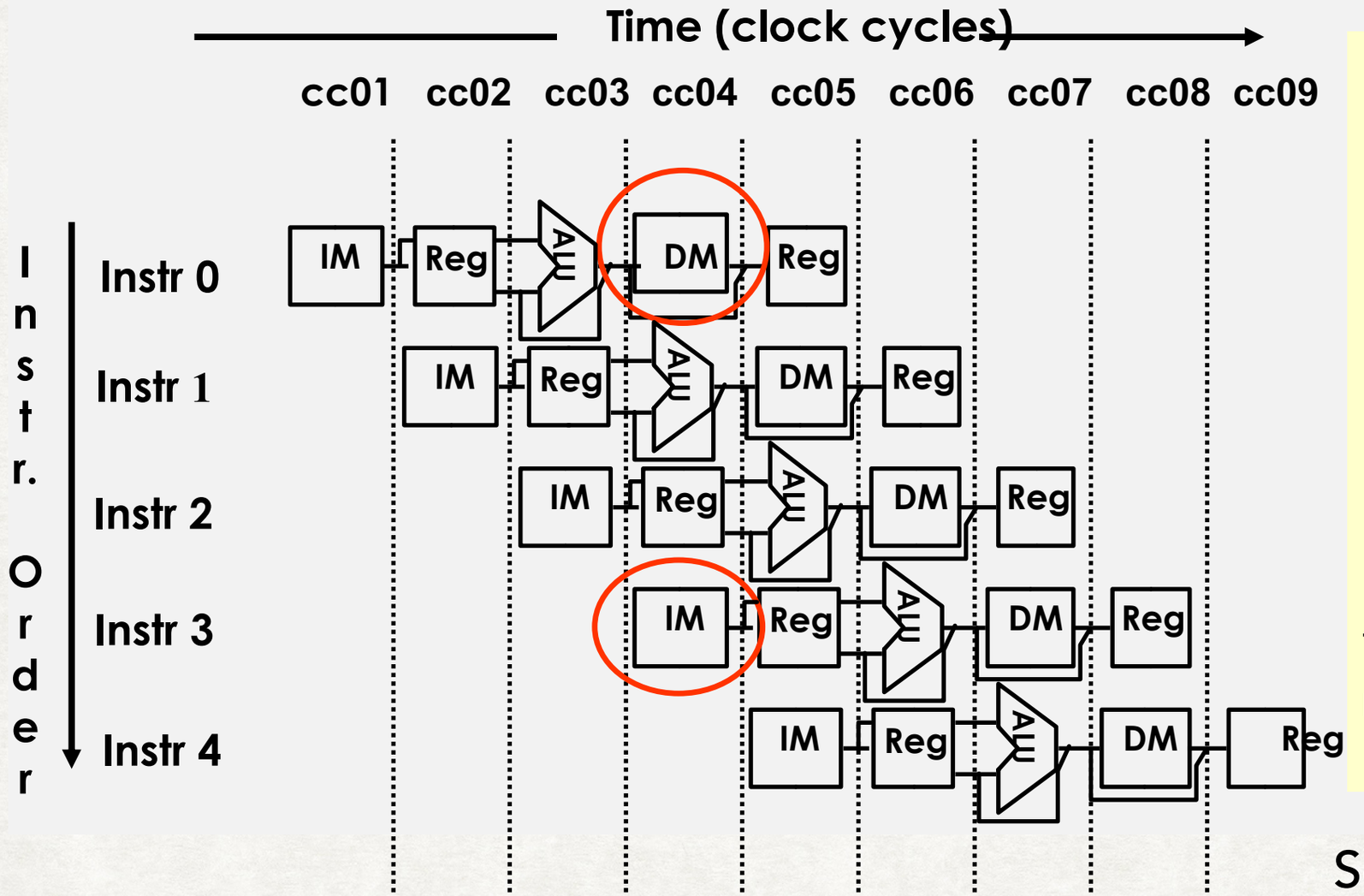

ADD
OR
AND
SUB
LDUR
PCSrc
Add
4
M u x
0
1
PC
Address
Instruction memory
IF/ID
Control
WB
M
EX
ID/EX
EX/MEM
MEM/WB
Instruction [28]
Reg2Loc
RegWrite
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Instruction [31–0]
32
Sign-extend
64
Instruction [31-21]
Instruction [4-0]
Shift left 2
Add
Add result
ALUSrc
ALU
Zero
ALU result
ALU control
ALUOp
Branch
MemWrite
Address
Read data
Data memory
Write data
MemRead
MemtoReg


ADD
OR
AND
SUB
PCSrc
Control
WB
M
EX
ID/EX
EX/MEM
MEM/WB
IF/ID
Add
4
M u x
0
1
PC
Address
Instruction memory
Instruction [28]
Reg2Loc
RegWrite
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Instruction [31–0]
32
Sign-extend
64
Instruction [31-21]
Instruction [4-0]
Shift left 2
Add
Add result
ALUSrc
Zero
ALU
ALU result
ALU control
ALUOp
Branch
MemWrite
Address
Read data
Data memory
Write data
MemRead
MemtoReg

HAZARDS

- Cause the instruction to not be able to execute
- Structural hazard
  - Two instructions need the same hardware in the same cycle
- Data hazard
  - An instruction needs data from a register before it has been calculated
- Control hazard
  - Which instruction to fetch is unknown because the result of a branch decision is not known yet

STRUCTURAL HAZARD – Example 1: Single Memory



Notation (per MIPS):

- **IM ≡ IF (Instr.Fetch)**
- **Reg(1) ≡ ID (Instr. Decode and DataFetch)**
- **ALU ≡ EXE**
- **DM ≡ MEM**
- **Reg(2) ≡ WB**

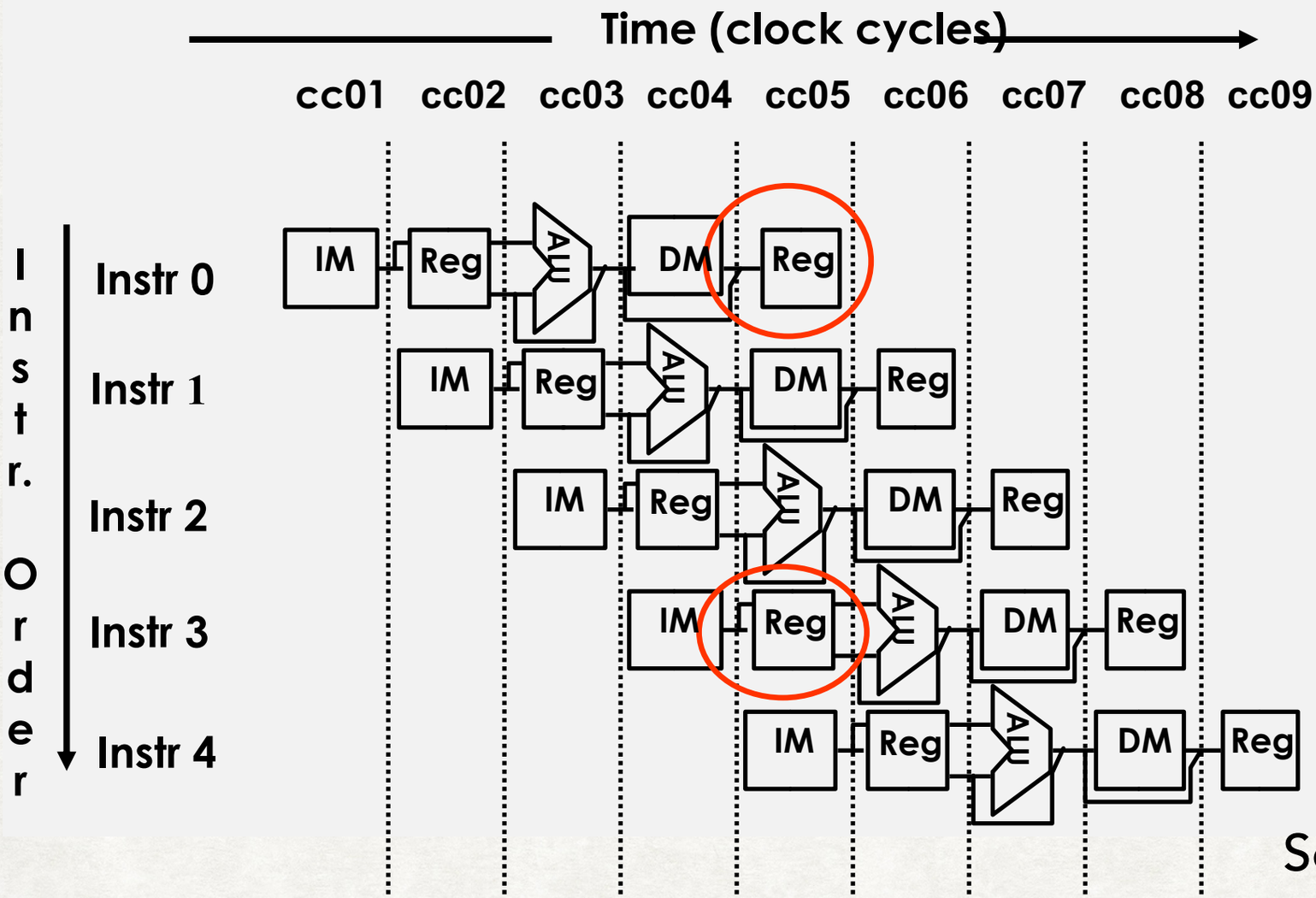
Problem:

In a single memory pipe-line, IM = DM

Thus, two instructions try to read the same memory (IM=DM) in one cycle

Solution: separate data and instruction memory "Harvard architecture"

STRUCTURAL HAZARD – Example 2: Register File



Notation (per MIPS):

- **IM ≡ IF (Instr.Fetch)**
- **Reg(1) ≡ ID (Instr. Decode and DataFetch)**
- **ALU ≡ EXE**
- **DM ≡ MEM**
- **Reg(2) ≡ WB**

Problem:

Assume there is only one Register File (usual case)

Two instructions try to read the Register File in one clock cycle … oops!

Solution: Split-cycle I/O

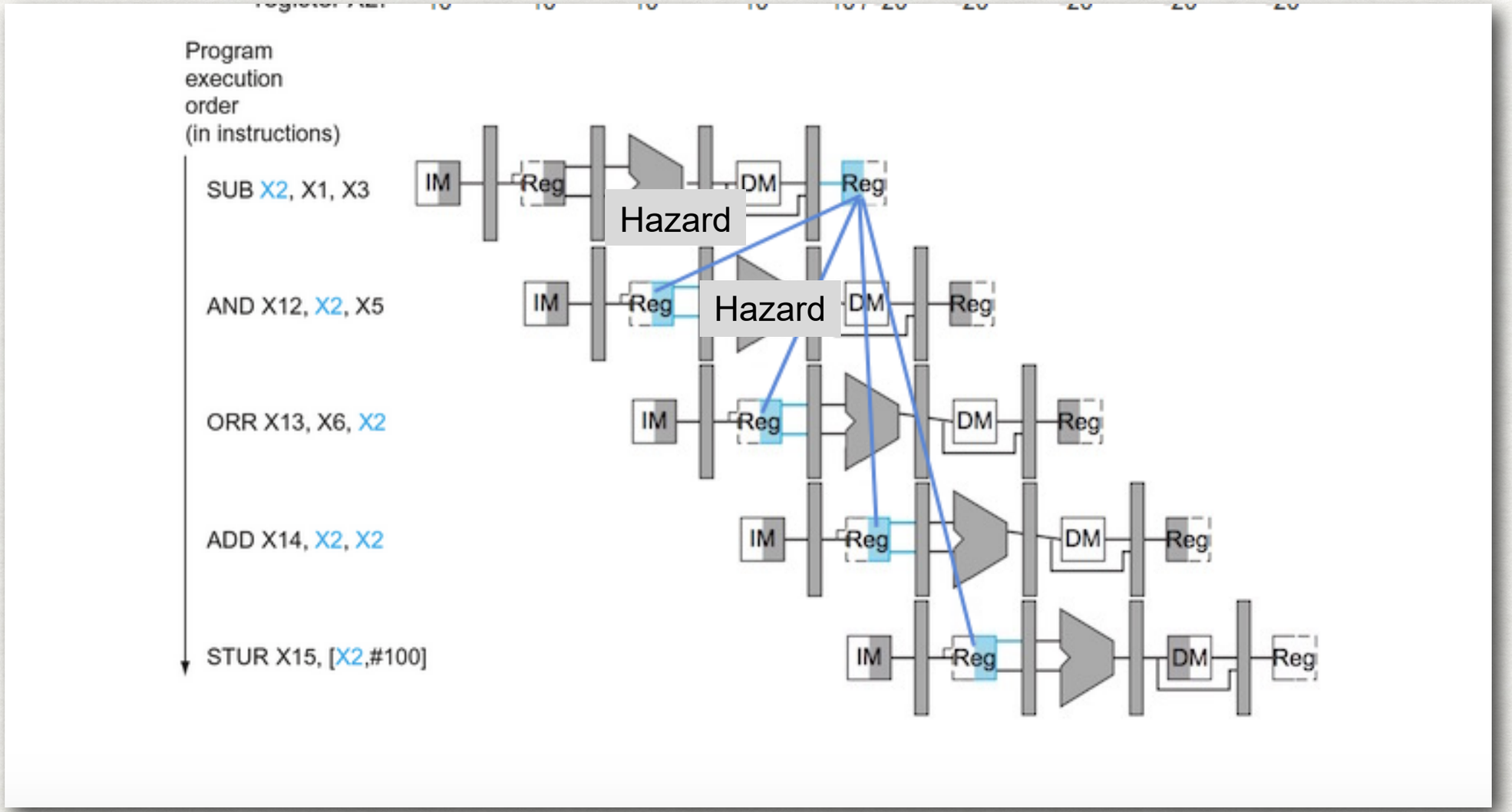
- Write occurs in the first half of the cycle.
- Read occurs in the second half of the cycle

DATA HAZARDS

```
SUB   X2, X1,X3      // Register X2 written by SUB
AND   X12,X2,X5      // 1st operand(X2) depends on SUB
OR    X13,X6,X2      // 2nd operand(X2) depends on SUB
ADD   X14,X2,X2      // 1st(X2) & 2nd(X2) depend on SUB
STUR  X15,[X2,#100] // Base (X2) depends on SUB
```

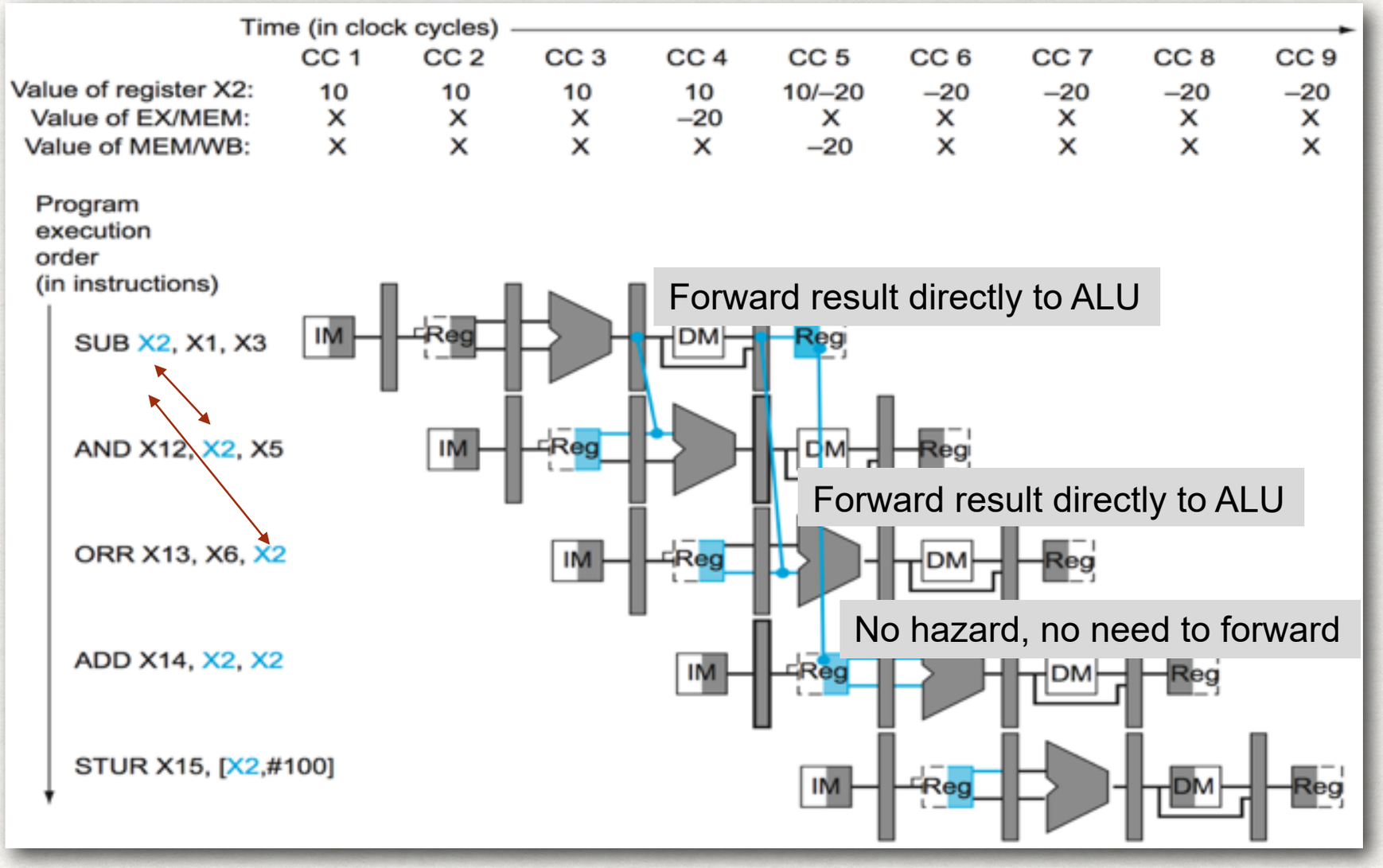
DATA HAZARDS

```
SUB   X2, X1,X3      // Register X2 written by SUB
AND   X12,X2,X5      // 1st operand(X2) depends on SUB
OR    X13,X6,X2      // 2nd operand(X2) depends on SUB
ADD   X14,X2,X2      // 1st(X2) & 2nd(X2) depend on SUB
STUR  X15,[X2,#100] // Base (X2) depends on SUB
```

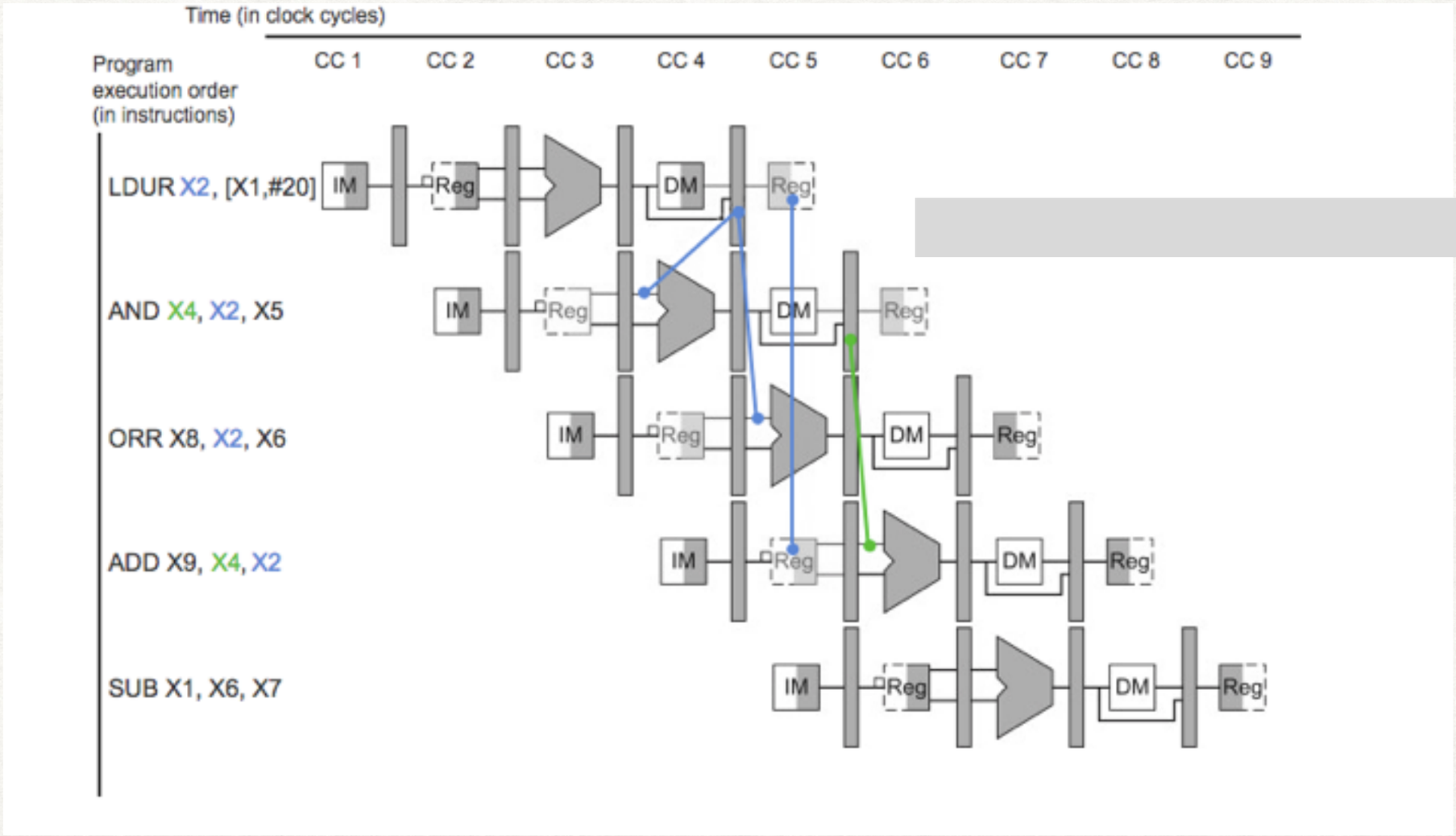


FORWARDING: R-TYPE

Can eliminate all R-type data hazards with forwarding!



LOAD-USE HAZARD



ding)

PIPELINE TRACING - EXAMPLE PROBLEM

Trace the execution of the following program

```
LDUR X2, [X1, #20]
AND  X4,  X2,  X5
ORR  X8,  X4,  X6
ADD  X9,  X8,  X2
SUB X1, X6, X7
```

PIPELINE TRACING - EXAMPLE PROBLEM

Trace the execution of the following program

```
LDUR X2, [X1, #20]
AND X4, X2, X5
ORR X8, X4, X6
ADD X9, X8, X2
SUB X1, X6, X7
```

Load/Use Hazard, Requires a Stall

PIPELINE TRACING - EXAMPLE PROBLEM

Trace the execution of the following program

```
LDUR X2, [X1, #20]
AND X4, X2, X5
ORR X8, X4, X6
ADD X9, X8, X2
SUB X1, X6, X7
```

Load/Use Hazard, Requires a Stall

Forwarding takes care of this, no stall

PIPELINE TRACING - EXAMPLE PROBLEM

Trace the execution of the following program

```
LDUR X2, [X1, #20]
AND X4, X2, X5
ORR X8, X4, X6
ADD X9, X8, X2
```

Load/Use Hazard, Requires a Stall

Forwarding takes care of this, no stall

Forwarding takes care of this, no stall

PIPELINE TRACING - EXAMPLE PROBLEM

Trace the execution of the following program

LDUR X2, [X1, #20]
AND X4, X2, X5
ORR X8, X4, X6
ADD X9, X8, X2

Cycle ->	1	2	3	4
LDUR X2, [X1, #20]	IF			

PIPELINE TRACING - EXAMPLE PROBLEM

Trace the execution of the following program

LDUR X2, [X1, #20]
AND X4, X2, X5
ORR X8, X4, X6
ADD X9, X8, X2

Cycle ->	1	2	3	4	5
LDUR X2, [X1, #20]	IF	ID	EX		
AND X4, X2, X5		IF	ID		
ORR X8, X4, X6			OR		

PIPELINE TRACING - EXAMPLE PROBLEM

Trace the execution of the following program

LDUR X2, [X1, #20]
AND X4, X2, X5
ORR X8, X4, X6
ADD X9, X8, X2

AND can't execute yet because value to be in X2 hasn't been read from memory yet

Cycle ->	1	2	3	4	5
LDUR X2, [X1, #20]	IF	ID	EX	MEM	
AND X4, X2, X5		IF	ID	ID	
ORR X8, X4, X6			IF	IF	
ADD X9, X8, X2					

PIPELINE TRACING - EXAMPLE PROBLEM

Trace the execution of the following program

LDUR X2, [X1, #20]
AND X4, X2, X5
ORR X8, X4, X6
ADD X9, X8, X2

Cycle ->	1	2	3	4	5
LDUR X2, [X1, #20]	IF	ID	EX	MEM	WB
AND X4, X2, X5		IF	ID	ID	EX
ORR X8, X4, X6			IF	IF	ID
ADD X9, X8, X2					IF

PIPELINE TRACING - EXAMPLE PROBLEM

Trace the execution of the following program

LDUR X2, [X1, #20]
AND X4, X2, X5
ORR X8, X4, X6
ADD X9, X8, X2

Cycle ->	1	2	3	4	5	6
LDUR X2, [X1, #20]	IF	ID	EX	MEM	WB	
AND X4, X2, X5		IF	ID	ID	EX	MEM
ORR X8, X4, X6			IF	IF	ID	EX
ADD X9, X8, X2					IF	ID

PIPELINE TRACING - EXAMPLE PROBLEM

Trace the execution of the following program

LDUR X2, [X1, #20]
AND X4, X2, X5
ORR X8, X4, X6
ADD X9, X8, X2

Cycle ->	1	2	3	4	5	6	7	8	9
LDUR X2, [X1, #20]	IF	ID	EX	MEM	WB				
AND X4, X2, X5		IF	ID	ID	EX	MEM	WB		
ORR X8, X4, X6			IF	IF	ID	EX	MEM	WB	
ADD X9, X8, X2					IF	ID	EX	MEM	WB

PIPELINE TRACING - ANOTHER VIEW OF THE SAME PROBLEM

Trace the execution of the following program

LDUR X2, [X1, #20]
AND X4, X2, X5
ORR X8, X4, X6
ADD X9, X8, X2

Use whichever option helps you understand better

Cycle	IF	ID	EX	MEM	WB
1	LDUR X2, [X1, #20]				
2	AND X4, X2, X5	LDUR X2, [X1, #20]			
3	ORR X8, X4, X6	AND X4, X2, X5	LDUR X2, [X1, #20]		

PIPELINE TRACING - EXAMPLE PROBLEM

Trace the execution of the following program

LDUR X2, [X1, #20]
AND X4, X2, X5
ORR X8, X4, X6
ADD X9, X8, X2

Use whichever option helps you understand better

AND can't move to EX stage yet, insert stall

Cycle	IF	ID	EX	MEM	WB
1	LDUR X2, [X1, #20]				
2	AND X4, X2, X5	LDUR X2, [X1, #20]			
3	ORR X8, X4, X6	AND X4, X2, X5	LDUR X2, [X1, #20]		
4	ORR X8, X4, X6	AND X4, X2, X5	stall	LDUR X2, [X1, #20]	
5					

PIPELINE TRACING - EXAMPLE PROBLEM

Trace the execution of the following program

LDUR X2, [X1, #20]
AND X4, X2, X5
ORR X8, X4, X6
ADD X9, X8, X2

Use whichever option helps you understand better

Cycle	IF	ID	EX	MEM	WB
1	LDUR X2, [X1, #20]				
2	AND X4, X2, X5	LDUR X2, [X1, #20]			
3	ORR X8, X4, X6	AND X4, X2, X5	LDUR X2, [X1, #20]		
4	ORR X8, X4, X6	AND X4, X2, X5	stall	LDUR X2, [X1, #20]	
5	ADD X9, X8, X2	ORR X8, X4, X6	AND X4, X2, X5	stall	LDUR X2, [X1, #20]

PIPELINE TRACING - EXAMPLE PROBLEM

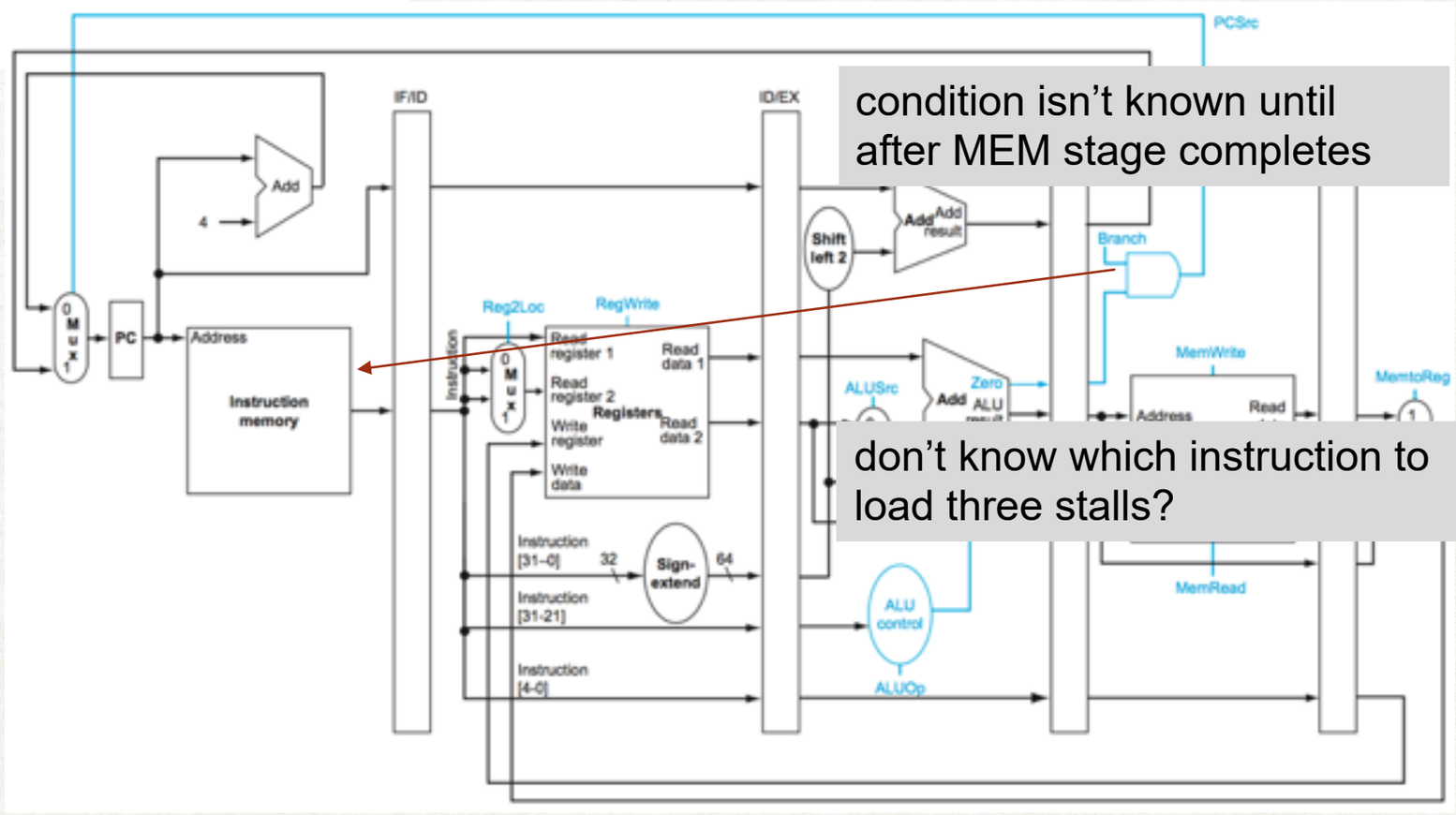
Trace the execution of the following program

LDUR X2, [X1, #20]
AND X4, X2, X5
ORR X8, X4, X6
ADD X9, X8, X2

Cycle	IF	ID	EX	MEM	WB
1	LDUR X2, [X1, #20]				
2	AND X4, X2, X5	LDUR X2, [X1, #20]			
3	ORR X8, X4, X6	AND X4, X2, X5	LDUR X2, [X1, #20]		
4	ORR X8, X4, X6	AND X4, X2, X5	stall	LDUR X2, [X1, #20]	
5	ADD X9, X8, X2	ORR X8, X4, X6	AND X4, X2, X5	stall	LDUR X2, [X1, #20]
6		ADD X9, X8, X2	ORR X8, X4, X6	AND X4, X2, X5	stall
7			ADD X9, X8, X2	ORR X8, X4, X6	AND X4, X2, X5
8				ADD X9, X8, X2	ORR X8, X4, X6
9					ADD X9, X8, X2
10					

CONTROL HAZARDS

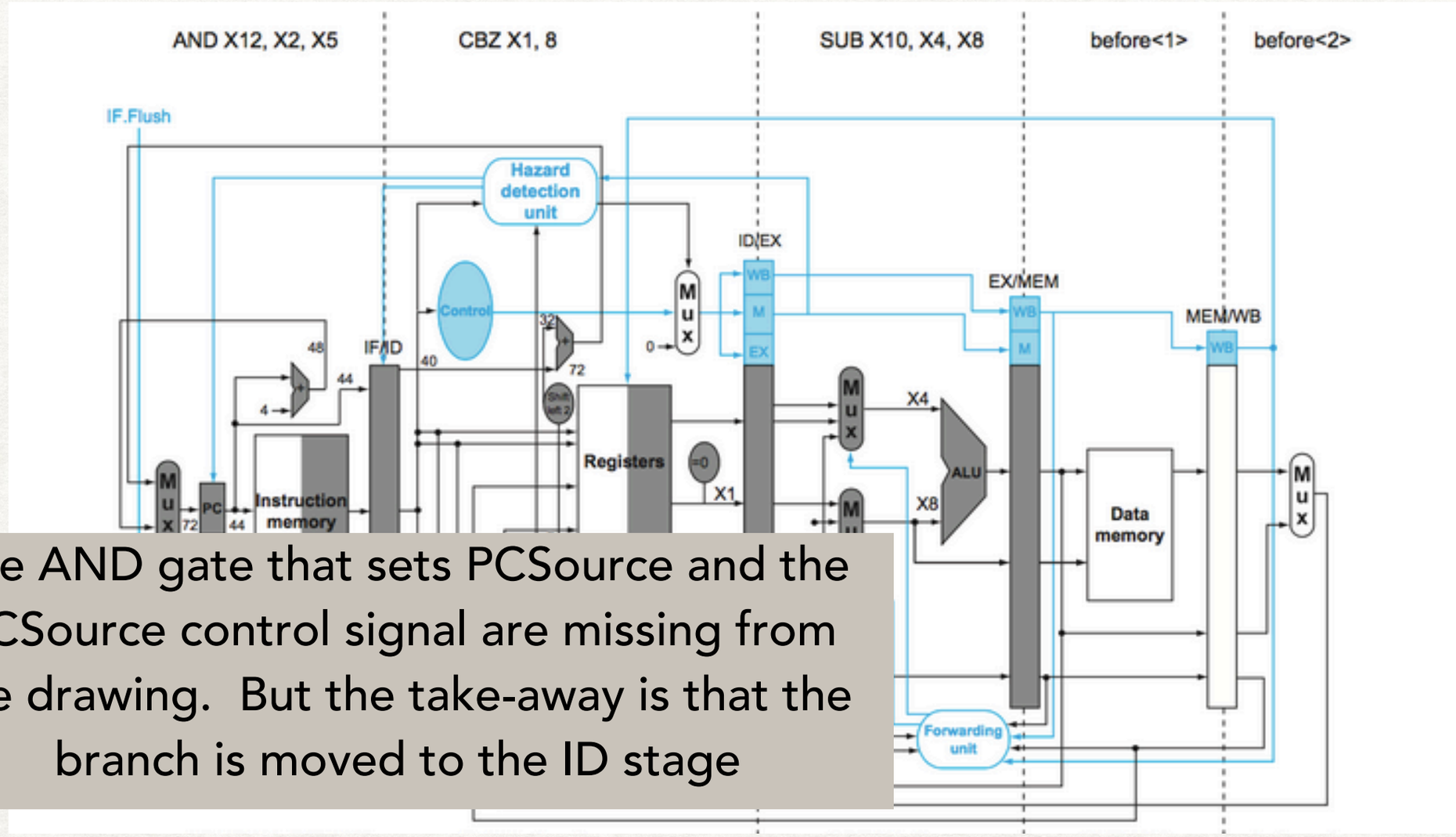
```
36 SUB X10, X4, X8
40 CBZ X1,  X3, 8   // PC-relative branch to 40+8*4=72
44 AND X12, X2, X5
48 ORR X13, X2, X6
52 ADD X14, X4, X2
56 SUB X15, X6, X7
. . .
72 LDUR  X4,  [X7,#50]
```



condition isn't known until after MEM stage completes

don't know which instruction to load three stalls?

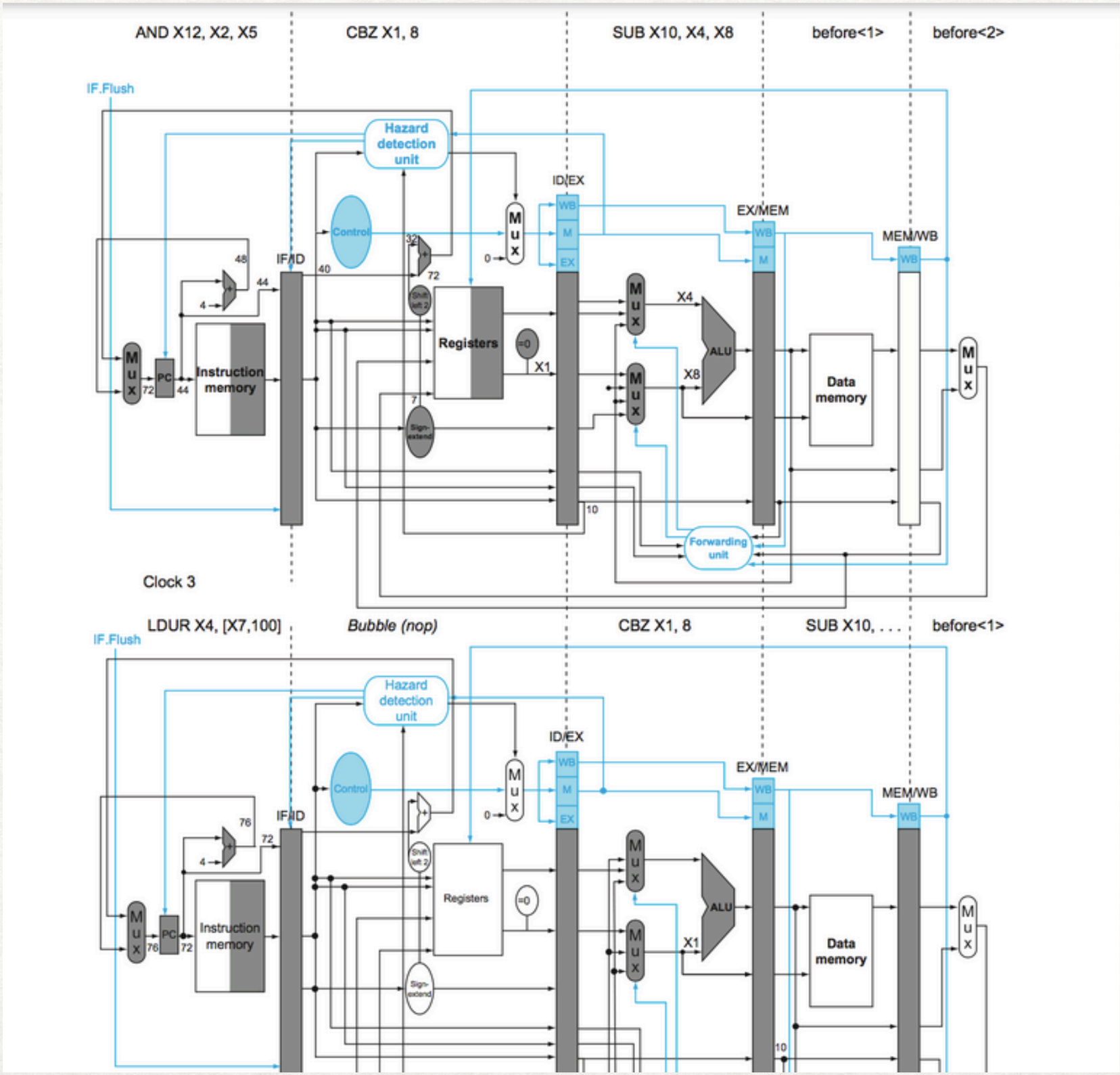
CONTROL HAZARDS



The AND gate that sets PCSource and the PCSource control signal are missing from the drawing. But the take-away is that the branch is moved to the ID stage

Move the branch calculation and condition checking to the ID stage

CONTROL HAZARDS



PIPELINE TRACING

CYCLE	IF	ID	EX	MEM	WB
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					

Branch prediction – assume branch not taken

```
                ADD X3, XZR, XZR
                CBZ X3, myBranch
                AND X9, X9, X2
                LDUR X3, [X1, #20]
                ADD  X4, X3, X5
                LDUR X5, [X1, #12]
myBranch:       ADD X4, X4, X3
```

PIPELINE TRACING

CYCLE	IF	ID	EX	MEM	WB
1	ADD X3, X2R, X2R				
2	CBZ	ADD X3, XZR, X2R			
3	AND	CBZ	ADD X3, XZR, XZR		
4	AND	CBZ	stall	ADD X3, XZR, XZR	
5	ADD X4, X3, X3	flush	CBZ	stall	ADD X3, XZR, XZR
6		ADD X4, X3, X3	flush	CBZ	stall
7			ADD X4, X3, X3	flush	CBZ
8				ADD X4, X3, X3	flush
9					ADD X4, X3, X3
10					
11					
12					
13					
14					

toy example where CBZ evaluates to true

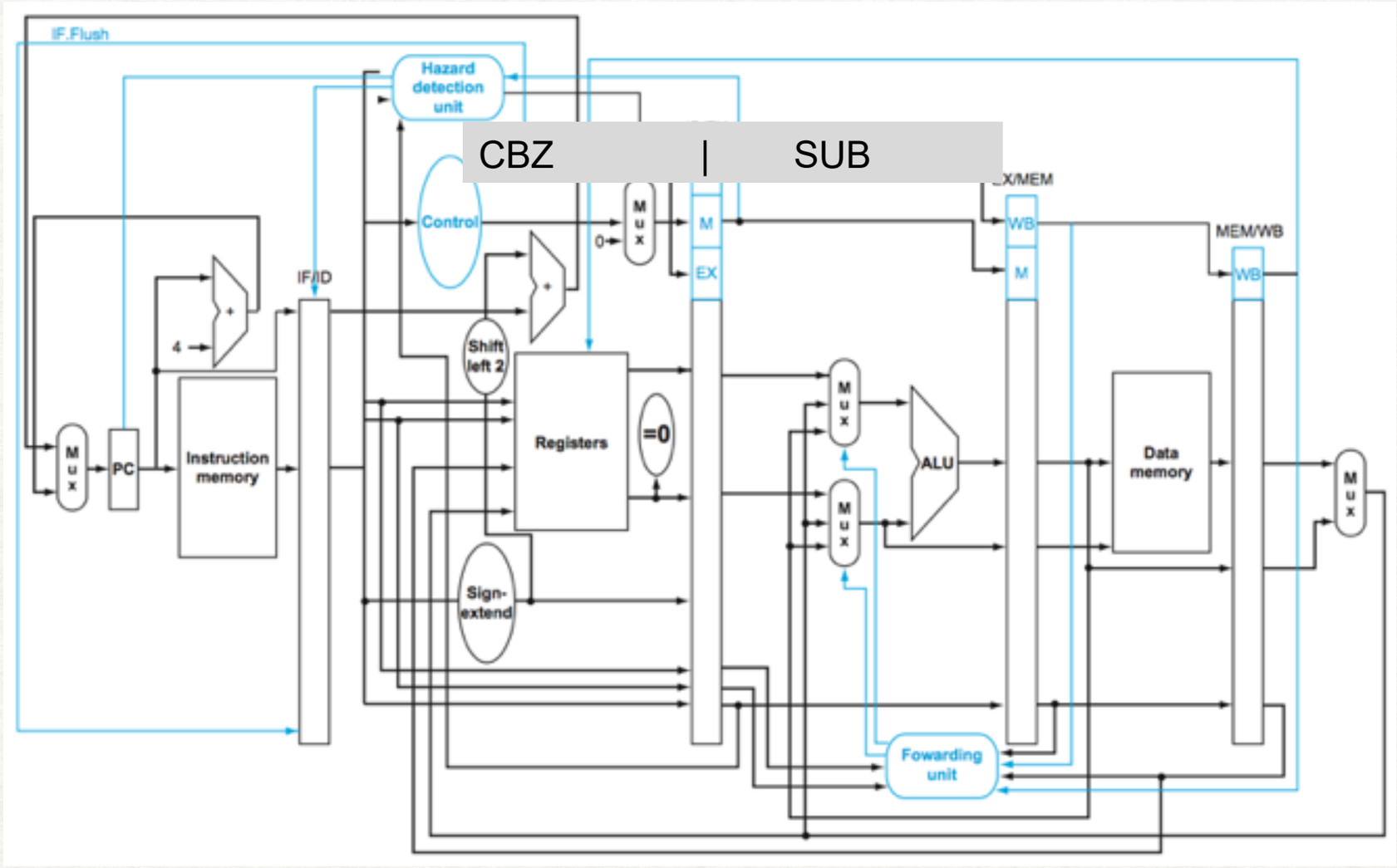
Branch prediction – assume branch not taken

```
                ADD X3, XZR, XZR
                CBZ X3, myBranch
                AND X9, X9, X2
                LDUR X3, [X1, #20]
                ADD  X4, X3, X5
                LDUR X5, [X1, #12]
myBranch:       ADD X4, X4, X3
```

DATA HAZARDS INVOLVING BRANCH

SUB X9, X10, X10
CBZ X9, #8

This will require a stall, because CBZ ID stage needs to wait for SUB EX stage



DATA HAZARDS INVOLVING BRANCH

```
36 AND X19, X20, X21
40 SUB X9, X10, X10
44 CBZ X9, #8
48 ADD X10, X10, 1
52
…
76 LDUR X4, [X7, #50]
```

Branch prediction – assume branch not taken
CBZ evaluates to true

CYCLE	IF	ID	EX	MEM	WB
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					

DATA HAZARDS INVOLVING BRANCH

```
36 AND X19, X20, X21
40 SUB X9, X10, X10
44 CBZ X9, #8
48 ADD X10, X10, 1
52
…
76 LDUR X4, [X7, #50]
```

CYCLE	IF	ID	EX	MEM	WB
1	AND X19, X20, X21				
2	SUB X9, X10, X10	AND X19, X20, X21			
3	CBZ X9, #8	SUB X9, X10, X10	AND X19, X20, X21		
4	ADD X10, X10, 1	CBZ X9, #8	SUB X9, X10, X10	AND X19, X20, X21	
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					

DATA HAZARDS INVOLVING BRANCH

```
36 AND X19, X20, X21
40 SUB X9, X10, X10
44 CBZ X9, #8
48 ADD X10, X10, 1
52
…
76 LDUR X4, [X7, #50]
```

CYCLE	IF	ID	EX	MEM	WB
1	AND X19, X20, X21				
2	SUB X9, X10, X10	AND X19, X20, X21			
3	CBZ X9, #8	SUB X9, X10, X10	AND X19, X20, X21		
4	ADD X10, X10, 1	CBZ X9, #8	SUB X9, X10, X10	AND X19, X20, X21	
5	ADD X10, X10, 1	CBZ X9, #8	**stall**	SUB X9, X10, X10	AND X19, X20, X21
6					
7					
8					
9					
10					
11					
12					
13					
14					

DATA HAZARDS INVOLVING BRANCH

```
36 AND X19, X20, X21
40 SUB X9, X10, X10
44 CBZ X9, #8
48 ADD X10, X10, 1
52
…
76 LDUR X4, [X7, #50]
```

CYCLE	IF	ID	EX	MEM	WB
1	AND X19, X20, X21				
2	SUB X9, X10, X10	AND X19, X20, X21			
3	CBZ X9, #8	SUB X9, X10, X10	AND X19, X20, X21		
4	ADD X10, X10, 1	CBZ X9, #8	SUB X9, X10, X10	AND X19, X20, X21	
5	ADD X10, X10, 1	CBZ X9, #8	**stall**	SUB X9, X10, X10	AND X19, X20, X21
6	LDUR X4, [X7, #50]	flush	CBZ X9, #8	stall	SUB X9, X10, X10
7					
8					
9					
10					
11					
12					
13					
14					

DATA HAZARDS INVOLVING BRANCH

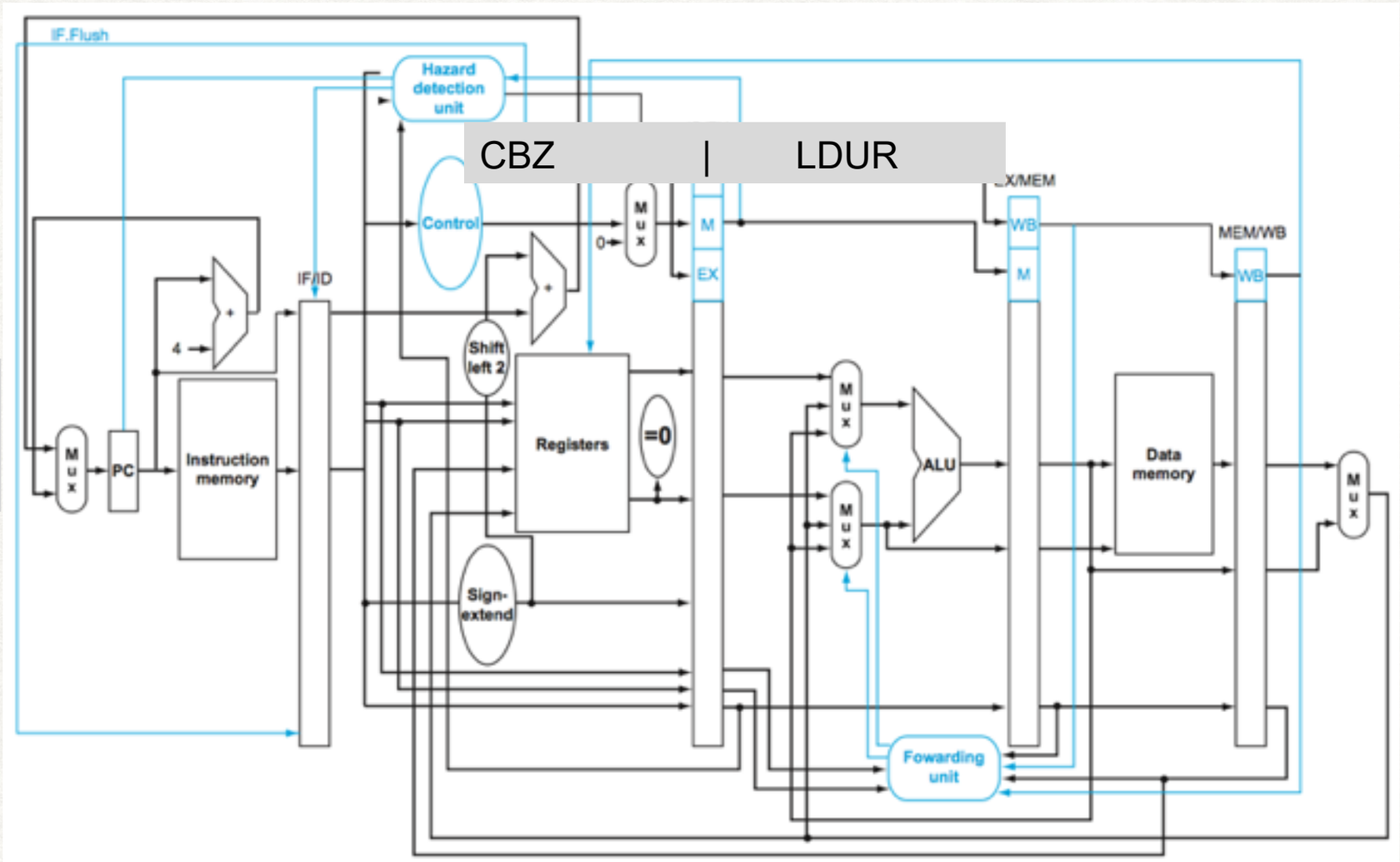
```
36 AND X19, X20, X21
40 SUB X9, X10, X10
44 CBZ X9, #8
48 ADD X10, X10, 1
52
…
76 LDUR X4, [X7, #50]
```

CYCLE	IF	ID	EX	MEM	WB
1	AND X19, X20, X21				
2	SUB X9, X10, X10	AND X19, X20, X21			
3	CBZ X9, #8	SUB X9, X10, X10	AND X19, X20, X21		
4	ADD X10, X10, 1	CBZ X9, #8	SUB X9, X10, X10	AND X19, X20, X21	
5	ADD X10, X10, 1	CBZ X9, #8	**stall**	SUB X9, X10, X10	AND X19, X20, X21
6	LDUR X4, [X7, #50]	flush	CBZ X9, #8	stall	SUB X9, X10, X10
7		LDUR X4, [X7, #50]	flush	CBZ X9, #8	stall
8			LDUR X4, [X7, #50]	flush	CBZ X9, #8
9				LDUR X4, [X7, #50]	flush
10					LDUR X4, [X7, #50]
11					
12					
13					
14					

DATA HAZARDS INVOLVING BRANCH

LDUR X9, [X10, #0]
CBZ X9, #8

This will require a two stalls, because CBZ ID stage needs to wait for LDUR MEM stage



DATA HAZARDS INVOLVING BRANCH

```
36 AND X19, X20, X21
40 LDUR X9, [X10, #0]
44 CBZ X9, #4
48 SUB X11, X11, #1
52 LSL  X19, #2
56 B Exit
60 LDUR X4, [X7, #50]
Exit: 64
```

Branch prediction – assume branch not taken
Actual - branch evaluates to false (should NOT be taken)

CYCLE	IF	ID	EX	MEM	WB
1					
2					
3					
4					
5					
7					
8					
9					
10					
11					
12					
13					
14					
15					

DATA HAZARDS INVOLVING BRANCH

```
36 AND X19, X20, X21
40 LDUR X9, [X10, #0]
44 CBZ X9, #4
48 SUB X11, X11, #1
52 LSL  X19, #2
56 B Exit
60 LDUR X4, [X7, #50]
Exit: 64
```

CYCLE	IF	ID	EX	MEM	WB
1	AND X19, X20, X21				
2	LDUR X9, [X10, #0]	AND X19, X20, X21			
3	CBZ X9, #8	LDUR X9, [X10, #0]	AND X19, X20, X21		
4	SUB X11, X11, #1	CBZ X9, #8	LDUR X9, [X10, #0]	AND X19, X20, X21	
5	SUB X11, X11, #1	CBZ X9, #8	bubble	LDUR X9, [X10, #0]	AND X19, X20, X21
	SUB X11, X11, #1	CBZ X9, #8	bubble	bubble	LDUR X9, [X10, #0]
7	LSL X19, #2	SUB X11, X11, #1	CBZ X9, #8	bubble	bubble
8	B Exit	LSL X19, #2	SUB X11, X11, #1	CBZ X9, #8	bubble
9		B Exit	**LSL X19, #2**	SUB X11, X11, #1	CBZ X9, #8
10			B Exit	LSL X19, #2	SUB X11, X11, #1
11				B Exit	**LSL X19, #2**
12					B Exit
13					
14					
15					

36 LDUR X18, [X19, #4] 40
ADD X20, X18, X17 44
ADD X20, X20, X21

Cycle	IF	ID	EX	MEM	WB

```
        40 ADD X20, X18, X17
        44 ADD X20, X20, X21
        48 CBZ X20, #3
        52 SUB X23, X21, X20
        56 B Exit
        60 ADD X23, X21, X0
Exit:   68 BR X30
```

Branch prediction: assume not taken
Result: the result of the branch is the branch should be taken

Cycle	IF	ID	EX	MEM	WB

```
        40 ADD X20, X18, X17
        44 LDUR X20, [X19,
        #4] 48 CBZ X20, #3
        52 SUB X23, X21, X20
        56 B Exit
        60 ADD X23, X21, X0
Exit:   68 BR X30
```

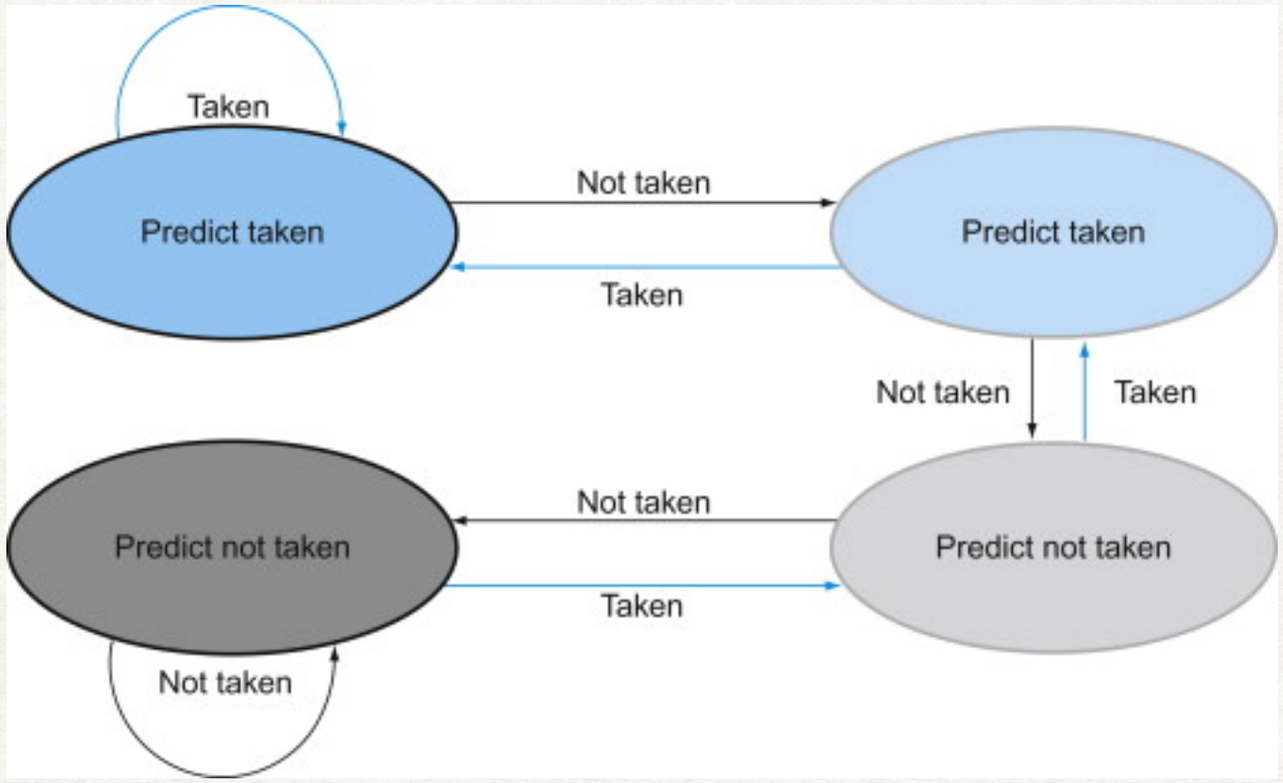
Branch prediction: assume not taken
Result: the result of the branch is the branch should not be taken

Cycle	IF	ID	EX	MEM	WB

STATIC BRANCH PREDICTION

- At compile time
  - Always not taken
  - Always taken
  - Backward branches always taken, forward branches always not taken
  - Profile information based on previous runs

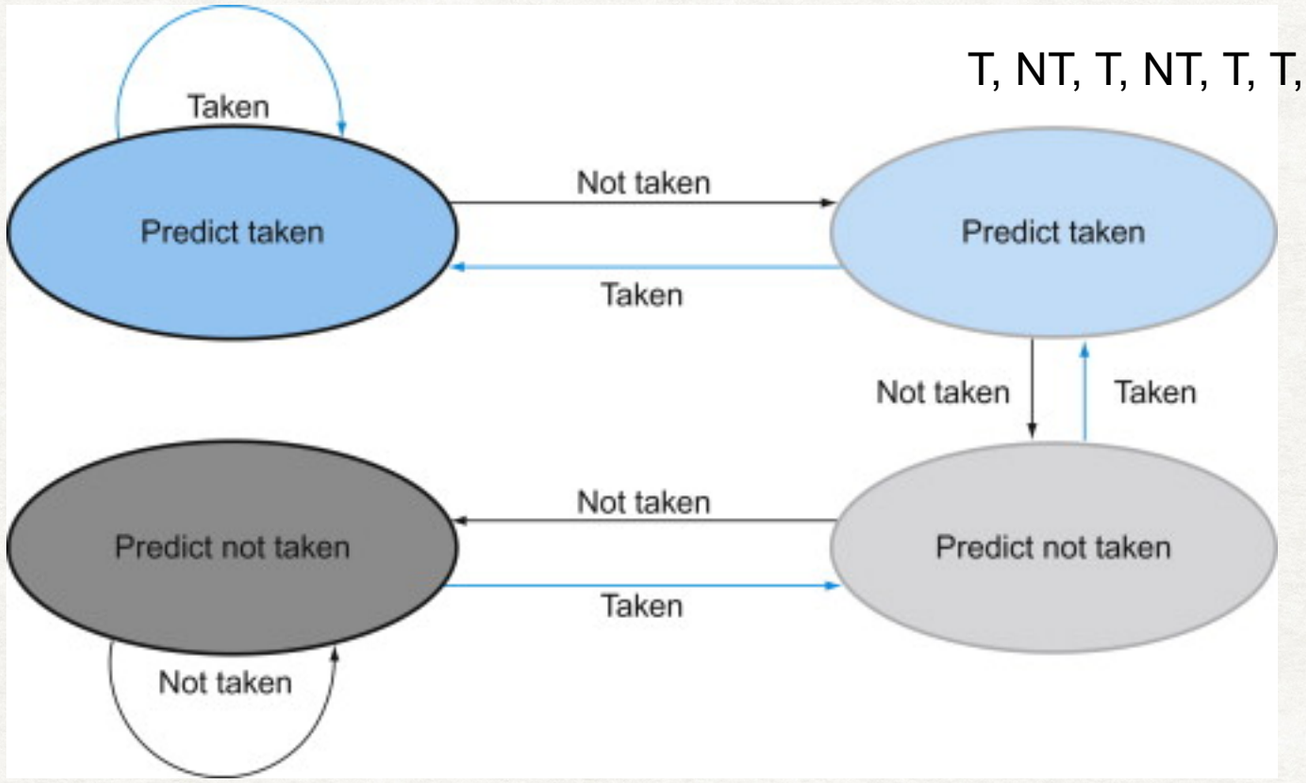
DYNAMIC BRANCH PREDICTION - TWO BIT PREDICTOR



DYNAMIC BRANCH PREDICTION - TWO BIT PREDICTOR

Example problem:
Predictor starts in lower left
What is the accuracy of the following set of predictions?

T, NT, T, NT, T, T, NT



DYNAMIC BRANCH PREDICTION - TWO BIT PREDICTOR

Example problem:
Predictor starts in lower left
What is the accuracy of the following set of predictions?

NT, NT, NT, NT, T