Arch Lab 1

Pipelined CPU supporting RISC-V

RVI32 Instructions

Chenlu Miao 09/2022

Tasks

- Understand RISC-V RV32I instructions
- Master the design methods of pipelined CPU executing RV32I instructions
- Master the method of **Pipeline Forwarding Detection** and **bypass unit** design
- Master the methods of 1-cycle stall of **Predict-not-taken branch** design
- Master methods of program verification of Pipelined CPU executing RV32I instructions

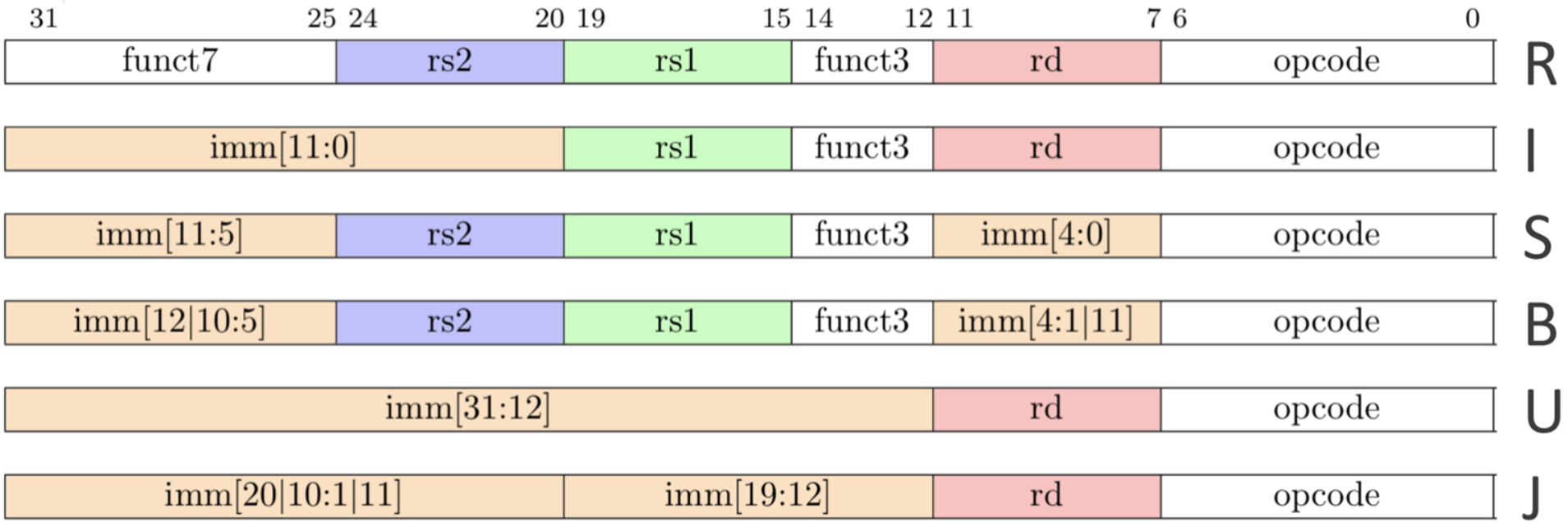
Code Overview

Overview

- RV32I Instructions
- Architecture Overview
- Data Hazard & Forwarding
- Control Hazard & Branch Prediction
- Hazard Detection Unit

RV32I Instructions

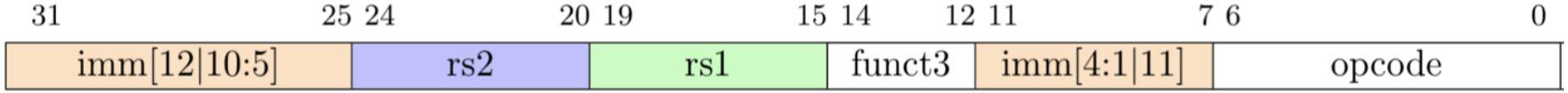
RV32I Instructions



RV32I Instructions – R-type

0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND

RV32I Instructions – B-type

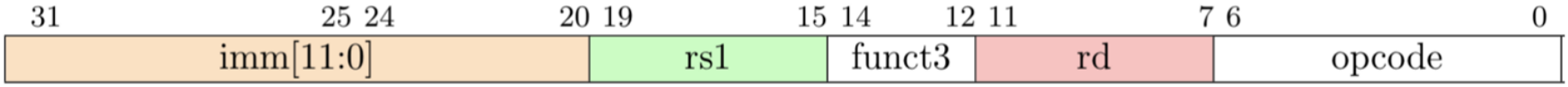


imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU

RV32I Instructions – S-type

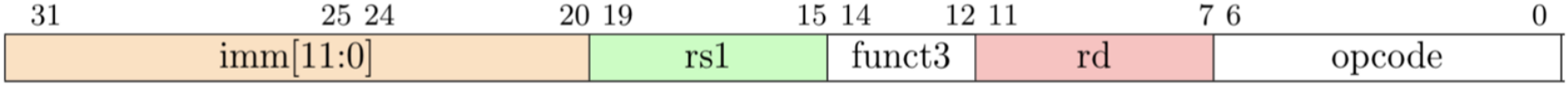
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW

RV32I Instructions – I-type



imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI

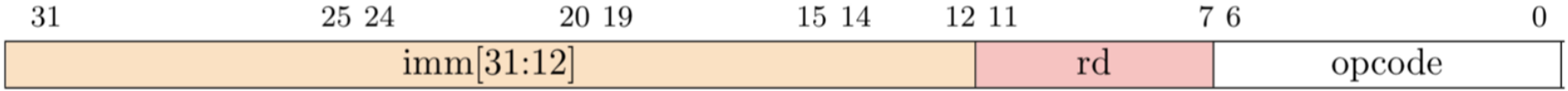
RV32I Instructions – I-type



imm[11:0]	rs1	000	rd	0000011	LB
imm[11:0]	rs1	001	rd	0000011	LH
imm[11:0]	rs1	010	rd	0000011	LW
imm[11:0]	rs1	100	rd	0000011	LBU
imm[11:0]	rs1	101	rd	0000011	LHU

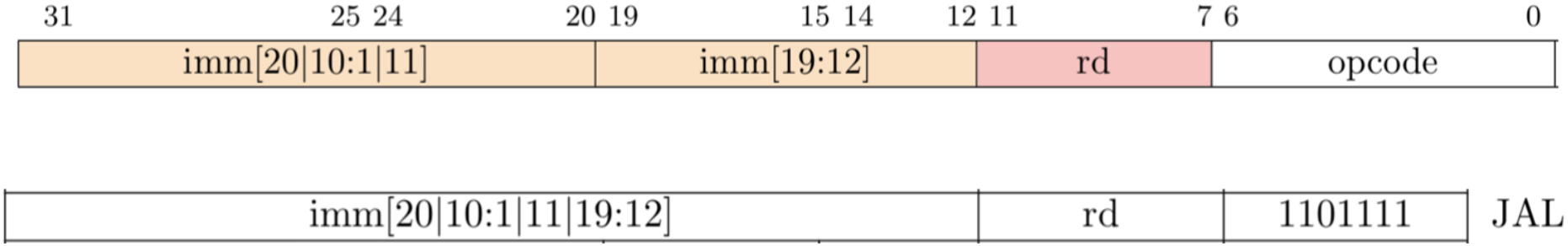
imm[11:0]	rs1	000	rd	1100111	JALR

RV32I Instructions – U-type

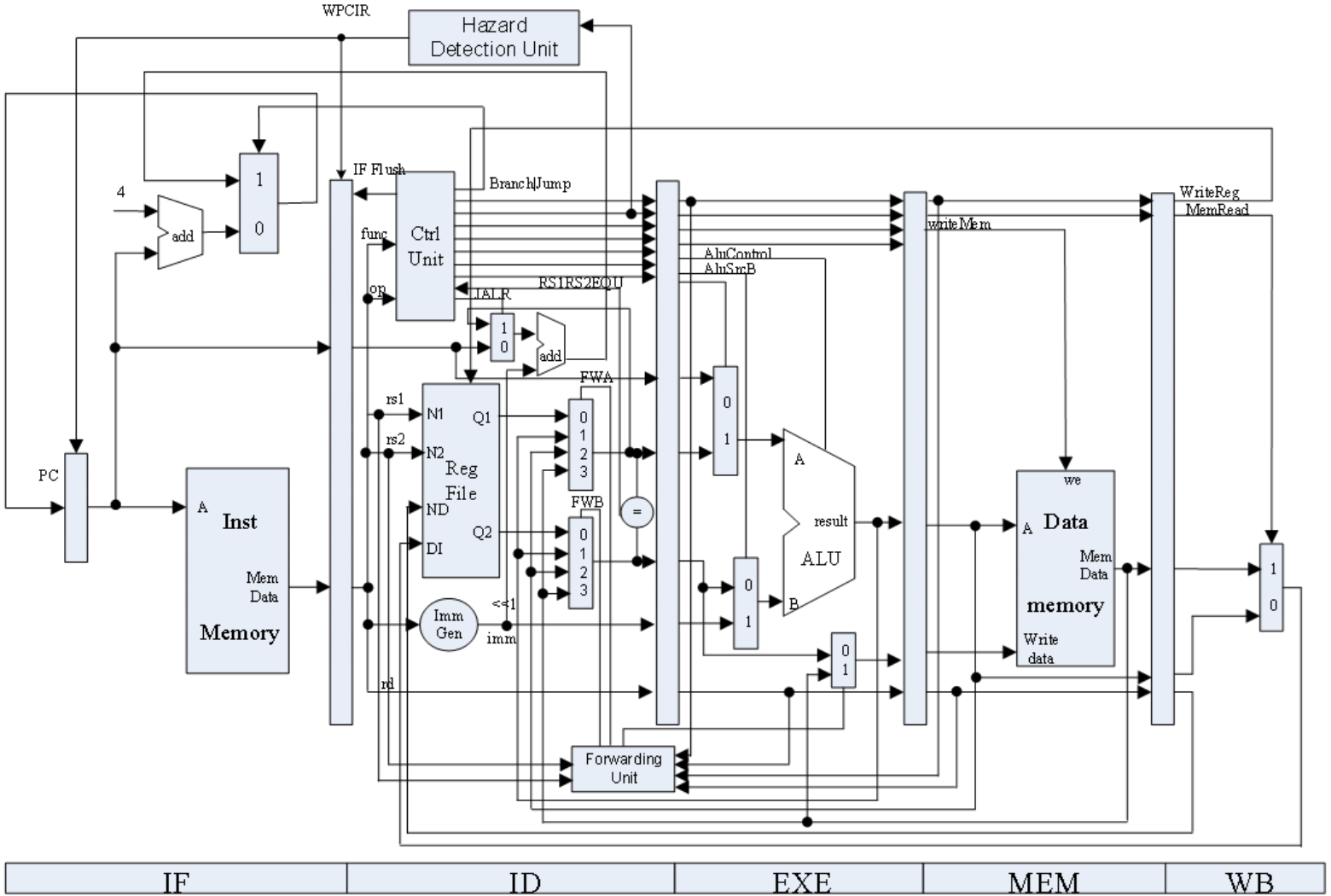


imm[31:12]	rd	0110111	LUI
imm[31:12]	rd	0010111	AUIPC

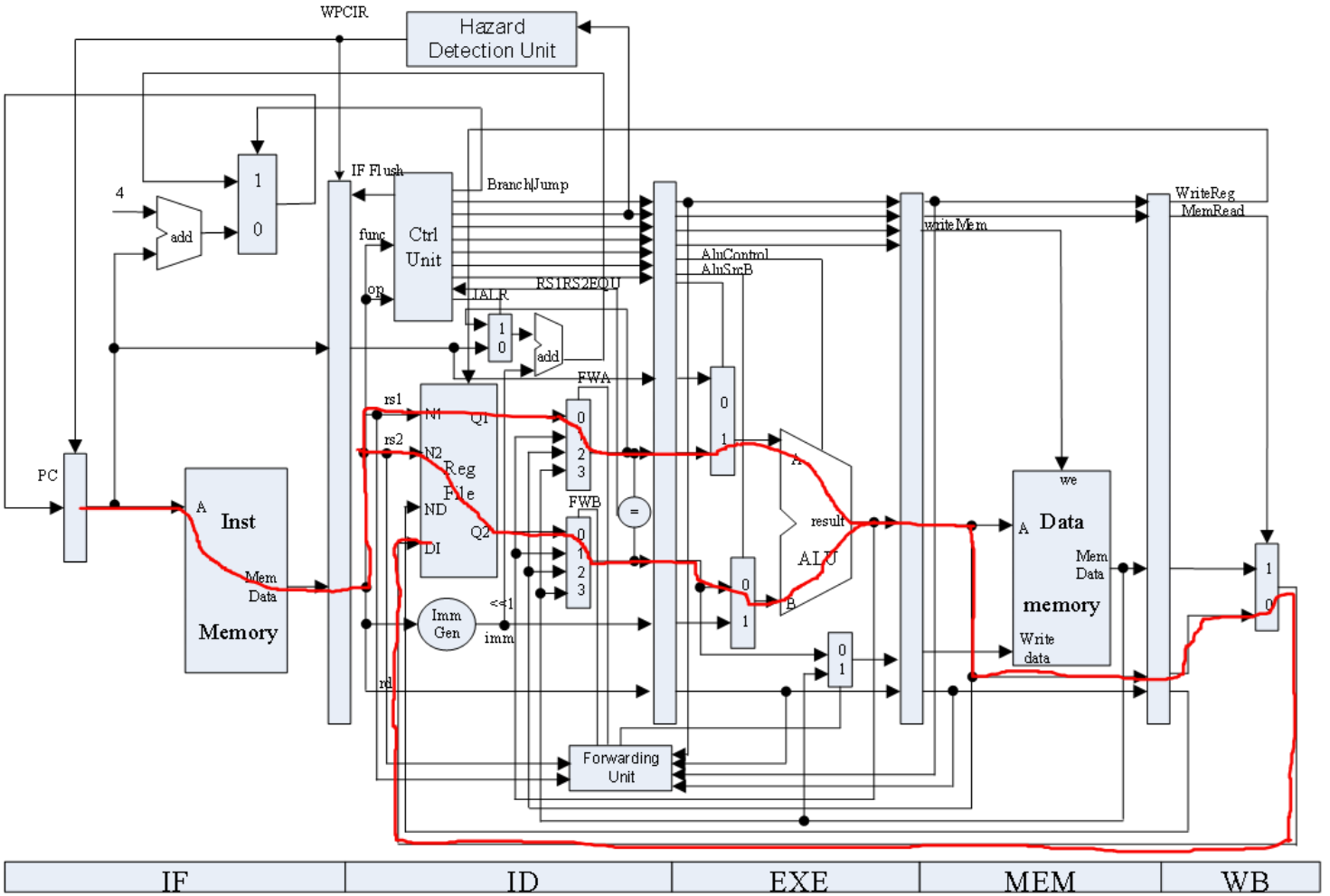
RV32I Instructions – J-type



Architecture Overview


WPCIR
Hazard Detection Unit
4
add
1
0
IF Flush
Branch|Jump
func
Ctrl Unit
op
RS1RS2EQU
|ALR
add
FWA
FWB
rs1
rs2
N1
N2
ND
DI
Q1
Q2
Reg File
Imm Gen
<<1
imm
rd
PC
A
Inst
Mem Data
Memory
AluControl
AluSrcB
writeMem
WriteReg
MemRead
A
result
ALU
B
we
A
Data
Mem Data
memory
Write data
=
Forwarding Unit
IF
ID
EXE
MEM
WB

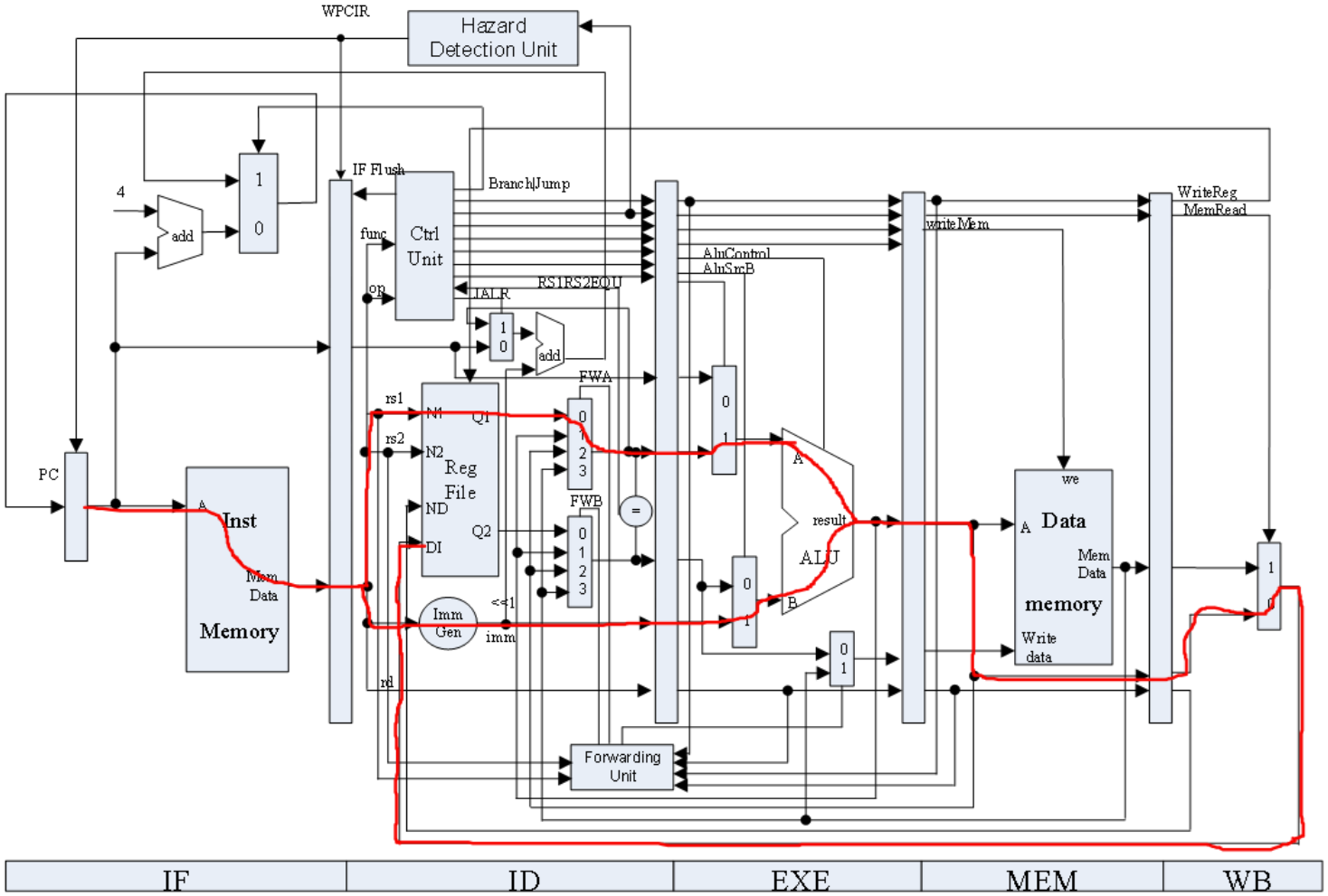
R-type


WPCIR
Hazard Detection Unit
IF Flush
Branch|Jump
4
add
1
0
func
op
Ctrl Unit
RS1RS2EQU
ALUControl
ALUSrcB
writeMem
WriteReg
MemRead
I&LR
FWA
FWB
rs1
rs2
rd
N1
N2
ND
DI
Q1
Q2
Reg File
PC
A
Inst
Mem Data
Memory
Imm Gen
<<1
imm
=
ALU
result
we
Data
memory
Write data
Forwarding Unit
IF
ID
EXE
MEM
WB

I-type

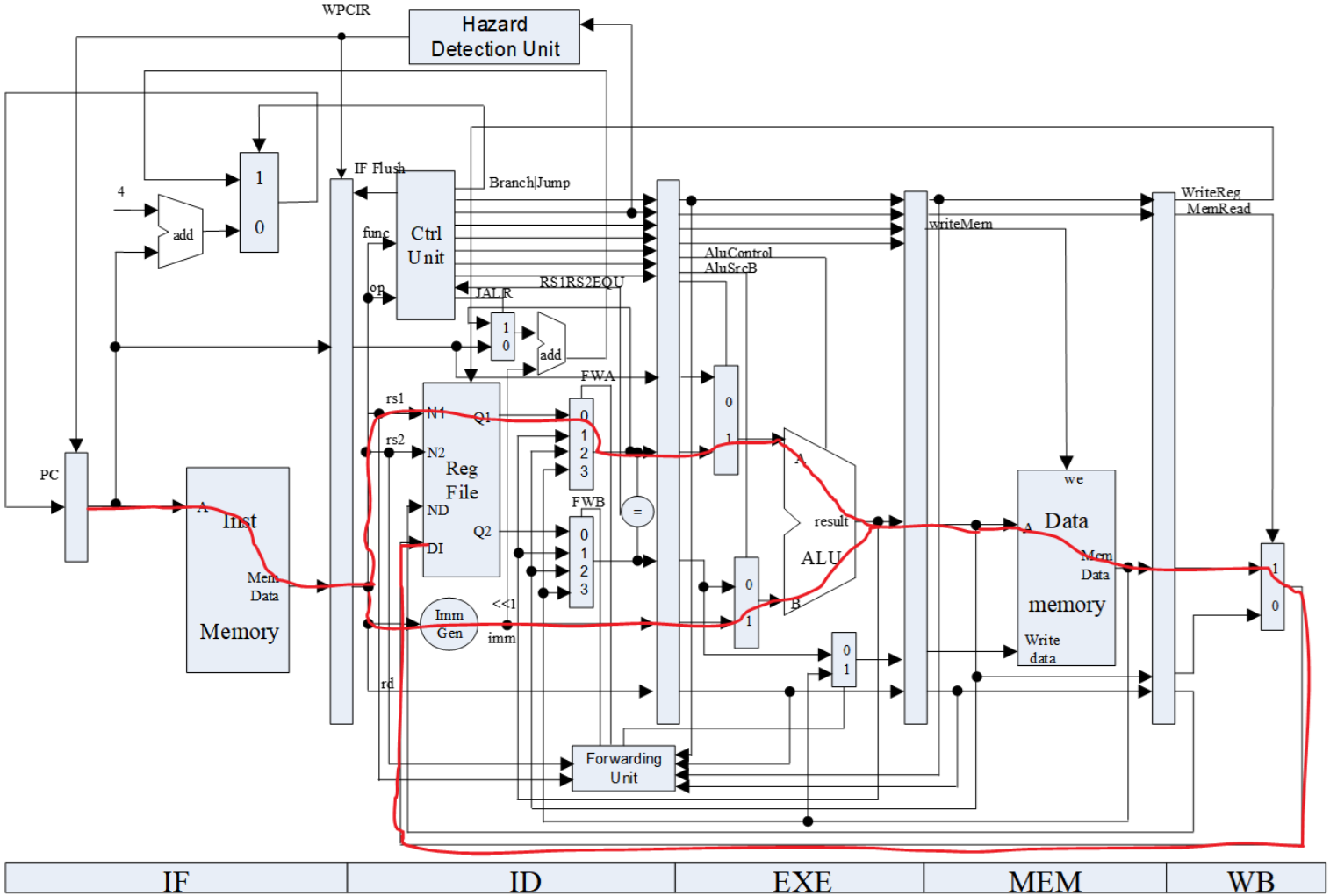
Arithmetic

and Logical



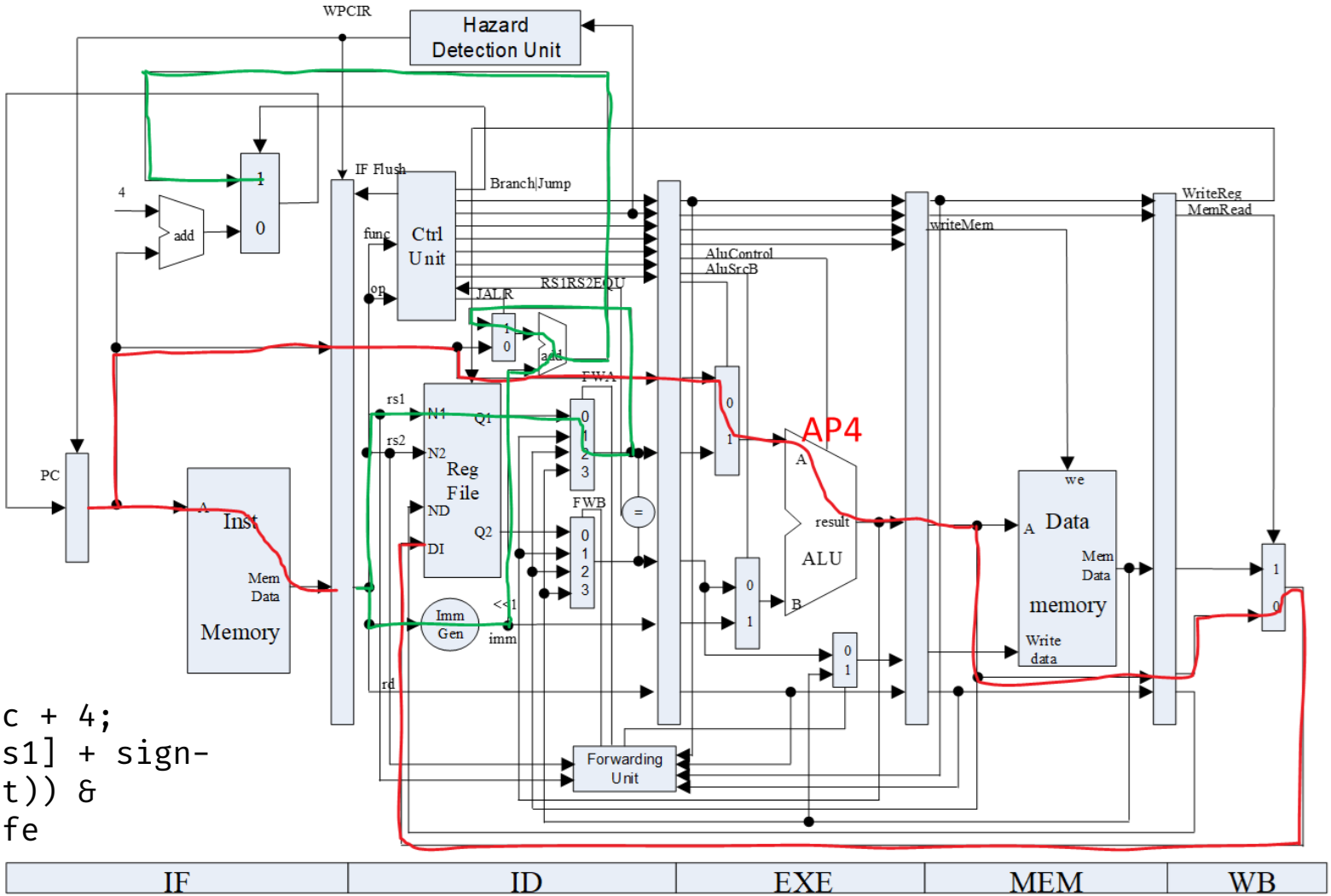
I-type

Load


WPCIR
Hazard Detection Unit
IF Flush
Branch|Jump
4
add
1
0
func
op
Ctrl Unit
JALR
RS1RS2EQU
AluControl
AluSrcB
writeMem
WriteReg
MemRead
FWA
FWB
rs1
rs2
N1
N2
ND
DI
Q1
Q2
Reg File
PC
A
Inst
Mem Data
Memory
Imm Gen
<<1
imm
rd
=
result
ALU
B
we
Data
Mem Data
memory
Write data
Forwarding Unit
IF
ID
EXE
MEM
WB

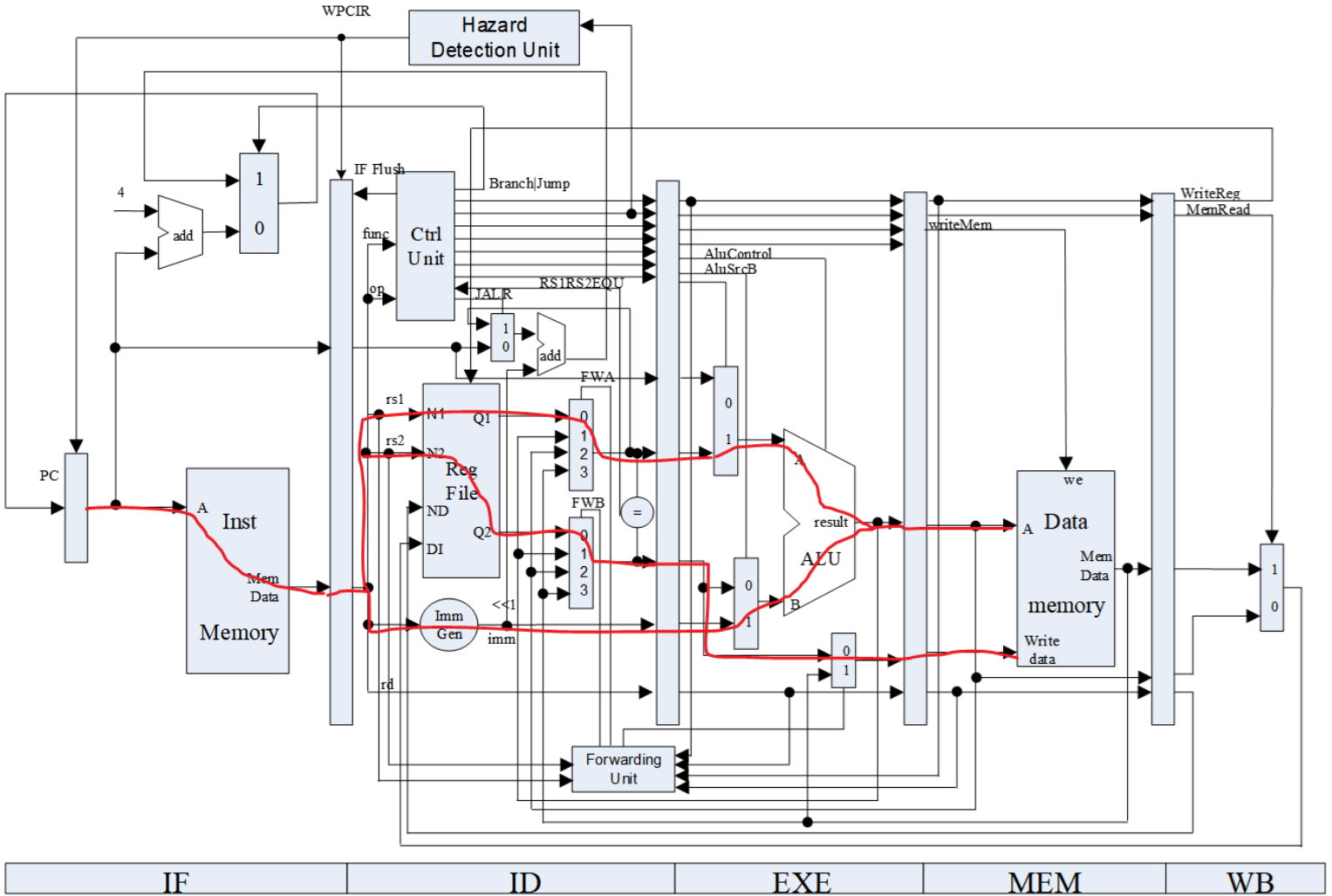
I-type

jalr

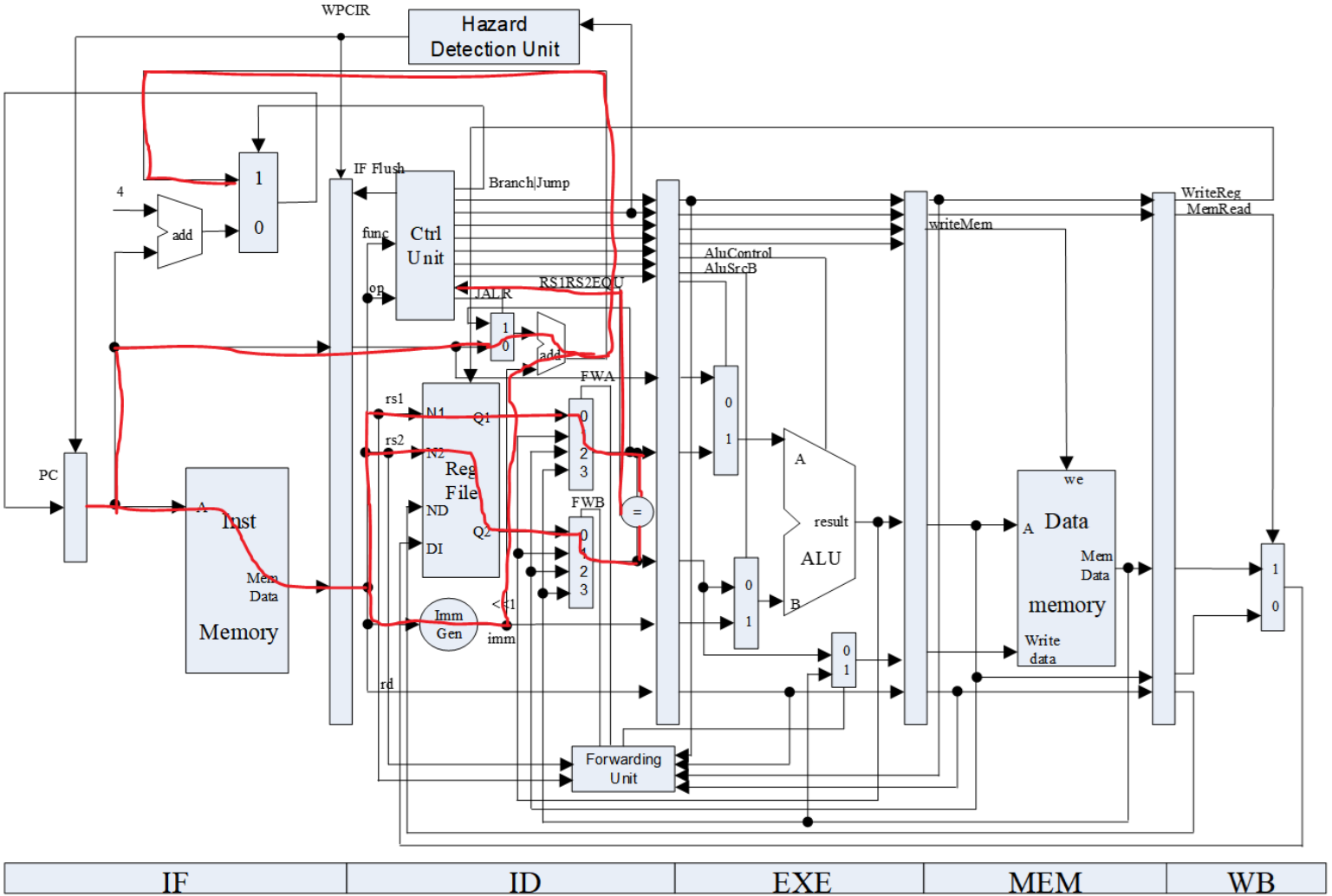


1. `x[rd] = pc + 4;`
2. `pc = (x[rs1] + sign-ext(offset)) & 0xffff_fffe`

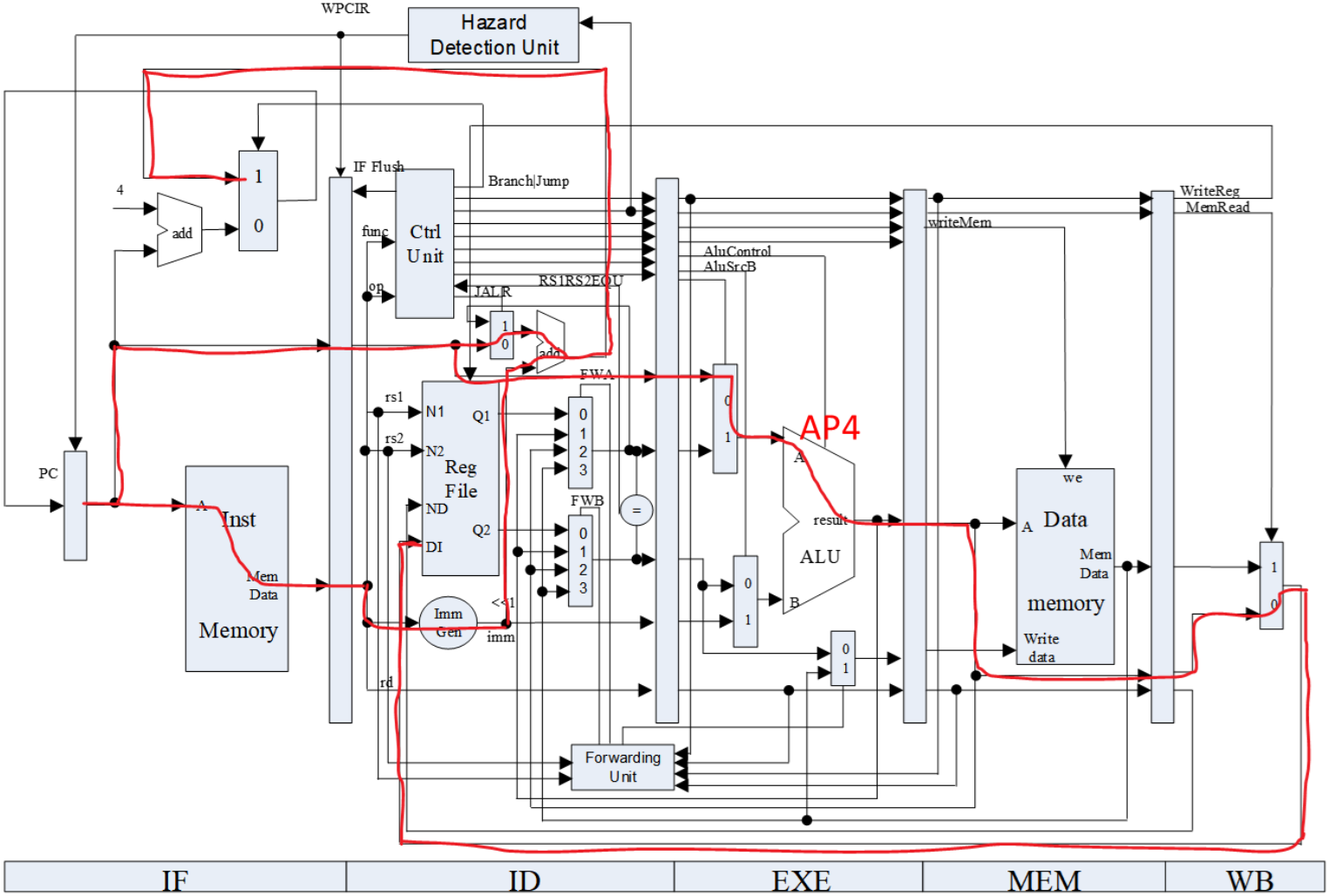
S-type


WPCIR
Hazard Detection Unit
IF Flush
Branch|Jump
4
add
1
0
func
Ctrl Unit
op
JALR
RS1RS2EQU
AluControl
AluSrcB
writeMem
WriteReg
MemRead
FWA
FWB
rs1
rs2
N1
N2
Q1
Q2
Reg File
ND
DI
PC
A
Inst
Mem Data
Memory
Imm Gen
<<1
imm
rd
=
result
ALU
B
we
Data
memory
Write data
Forwarding Unit
IF
ID
EXE
MEM
WB

B-type

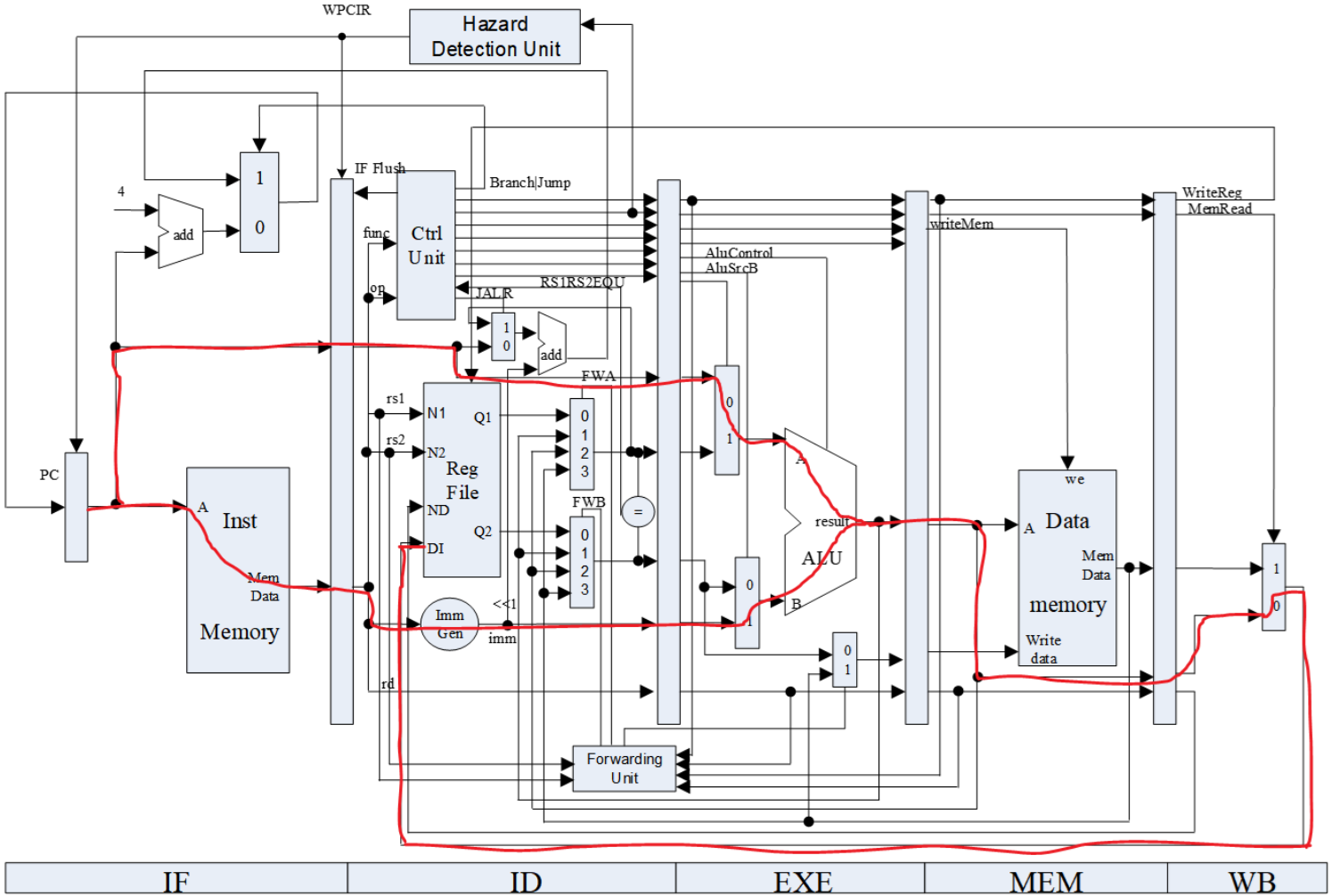

WPCIR
Hazard Detection Unit
IF Flush
Branch|Jump
4
add
1
0
func
Ctrl Unit
op
JALR
RS1RS2EQU
AluControl
AluSrcB
writeMem
WriteReg
MemRead
FWA
FWB
rs1
rs2
N1
N2
Q1
Q2
Reg File
ND
DI
PC
A
Inst
Mem Data
Memory
Imm Gen
imm
rd
ALU
result
we
Data
memory
Write data
Forwarding Unit
IF
ID
EXE
MEM
WB

J-type


WPCIR
Hazard Detection Unit
IF Flush
Branch|Jump
4
add
1
0
func
Ctrl Unit
op
JALR
RS1RS2EQU
AluControl
AluSrcB
writeMem
WriteReg
MemRead
FWA
FWB
rs1
rs2
N1
N2
ND
DI
Q1
Q2
Reg File
PC
Inst
Mem Data
Memory
Imm Gen
<<1
imm
rd
AP4
A
B
ALU
result
we
Data
memory
Write data
Forwarding Unit
IF
ID
EXE
MEM
WB

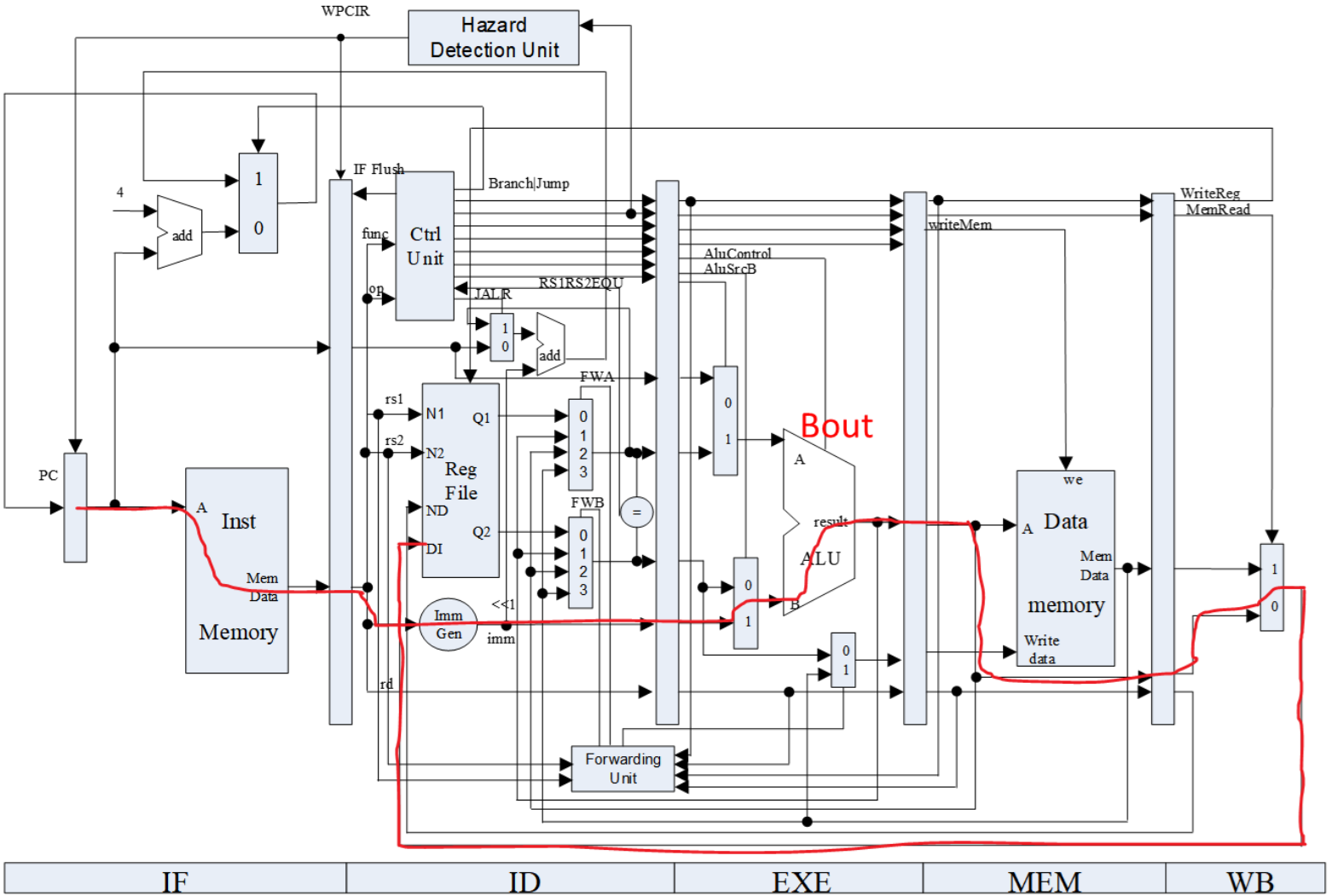
U-type

auipc


WPCIR
Hazard Detection Unit
IF Flush
Branch|Jump
4
add
1
0
func
Ctrl Unit
op
JALR
RS1RS2EQU
AluControl
AluSrcB
writeMem
WriteReg
MemRead
FWA
FWB
rs1
rs2
N1
N2
Q1
Q2
ND
DI
Reg File
PC
A
Inst
Mem Data
Memory
Imm Gen
<<1
imm
rd
ALU
result
we
Data
memory
Write data
Forwarding Unit
IF
ID
EXE
MEM
WB

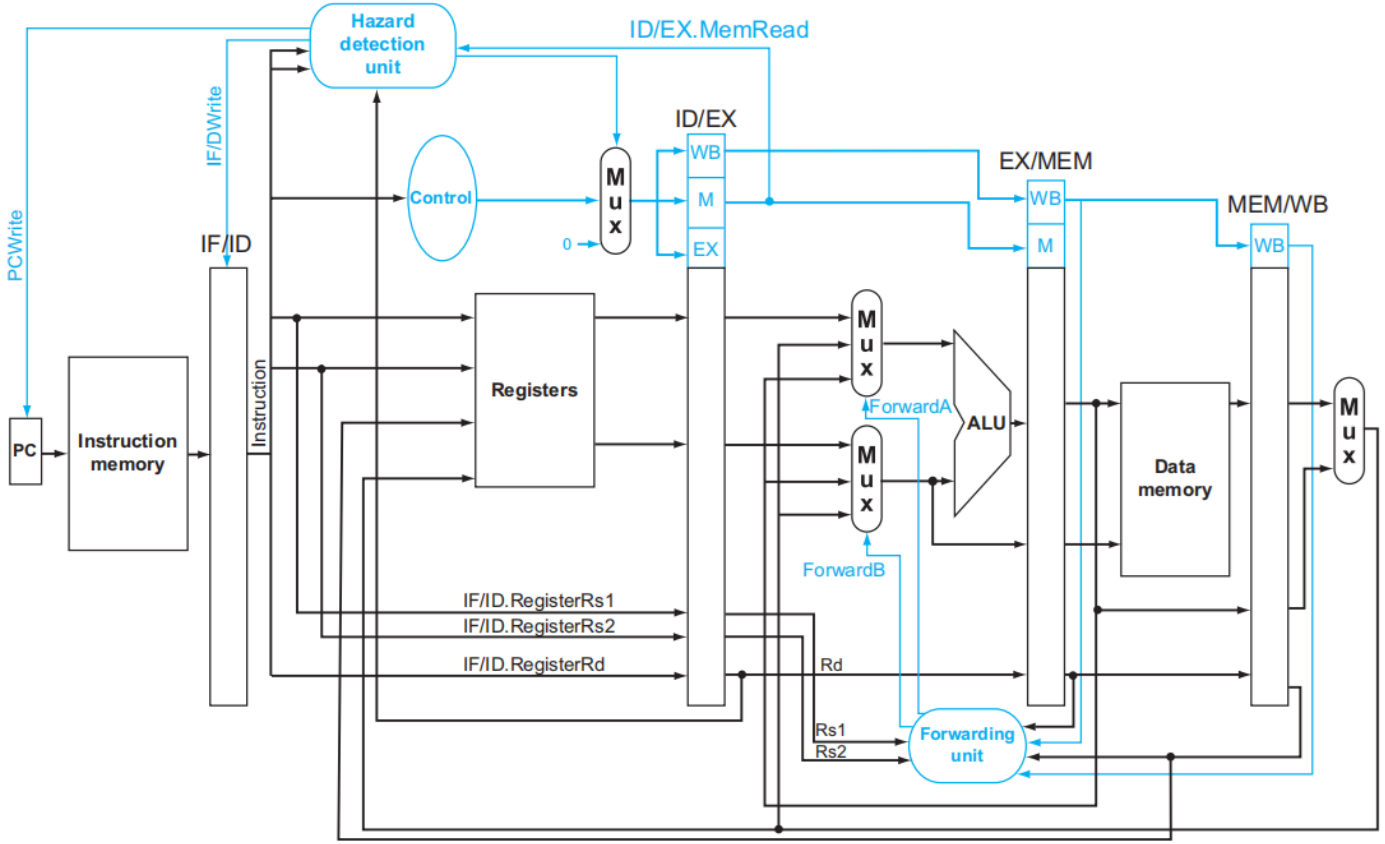
U-type

lui


WPCIR
Hazard Detection Unit
IF Flush
Branch|Jump
4
add
1
0
func
Ctrl Unit
op
JALR
RS1RS2EQU
AluControl
AluSrcB
writeMem
WriteReg
MemRead
FWA
FWB
rs1
rs2
N1
N2
Q1
Q2
Reg File
ND
DI
PC
A
Inst
Mem Data
Memory
Imm Gen
<<1
imm
rd
Bout
ALU
result
B
we
Data
Mem Data
memory
Write data
Forwarding Unit
IF
ID
EXE
MEM
WB

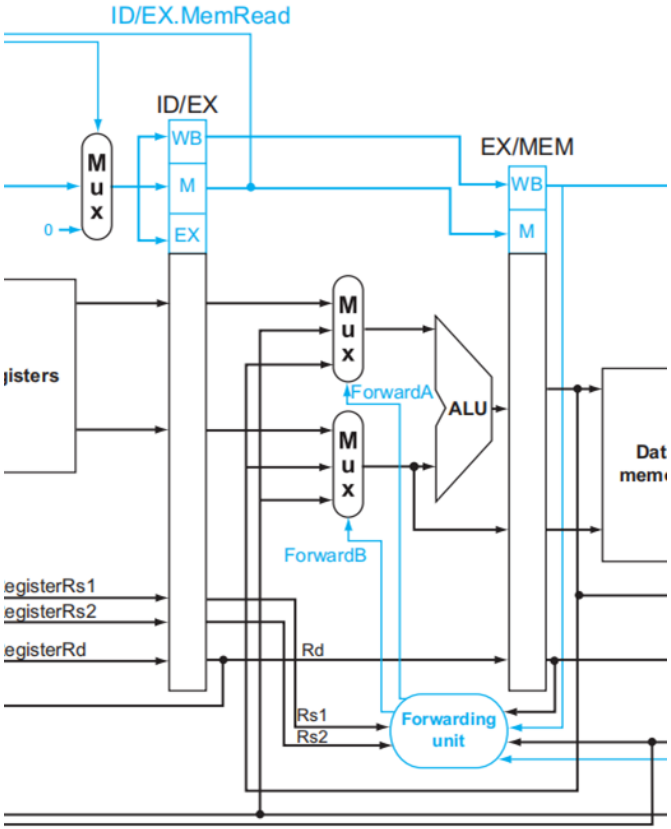
Data Hazard & Forwarding

Data Hazard & Forwarding



Data Hazard & Forwarding

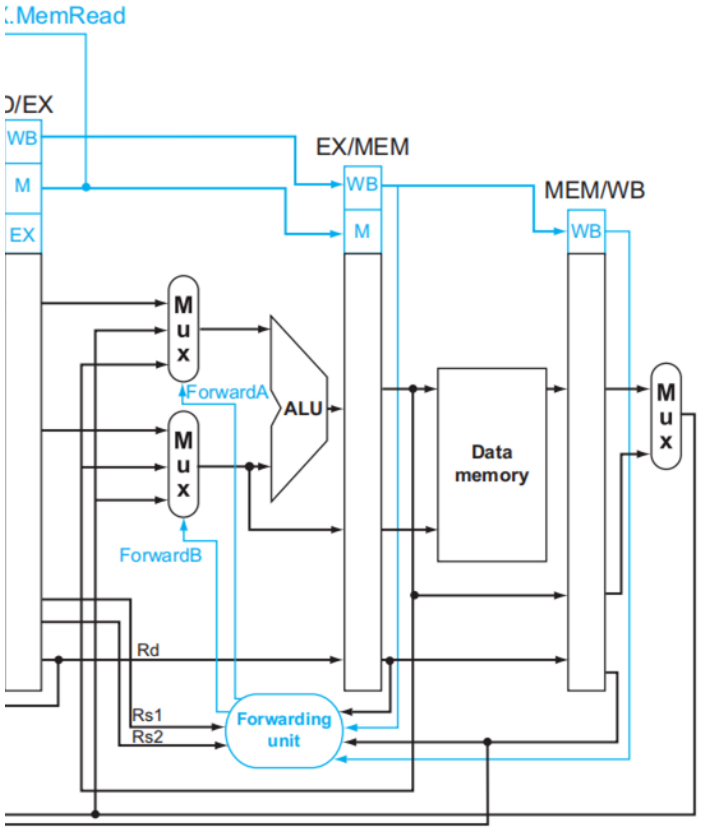
```
if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd ≠ 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRs1))
     ForwardA = 10

if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd ≠ 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRs2))
     ForwardB = 10
```

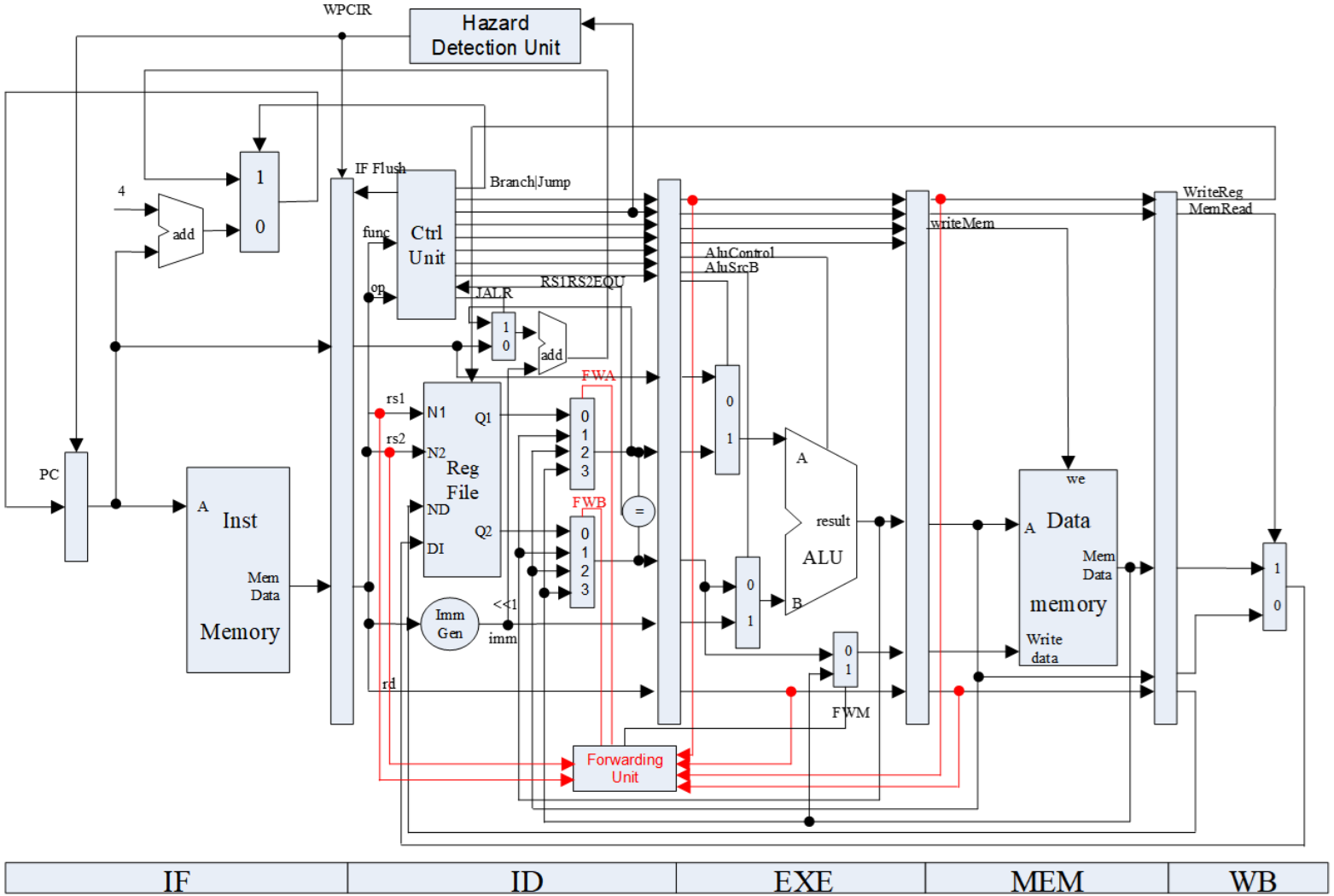


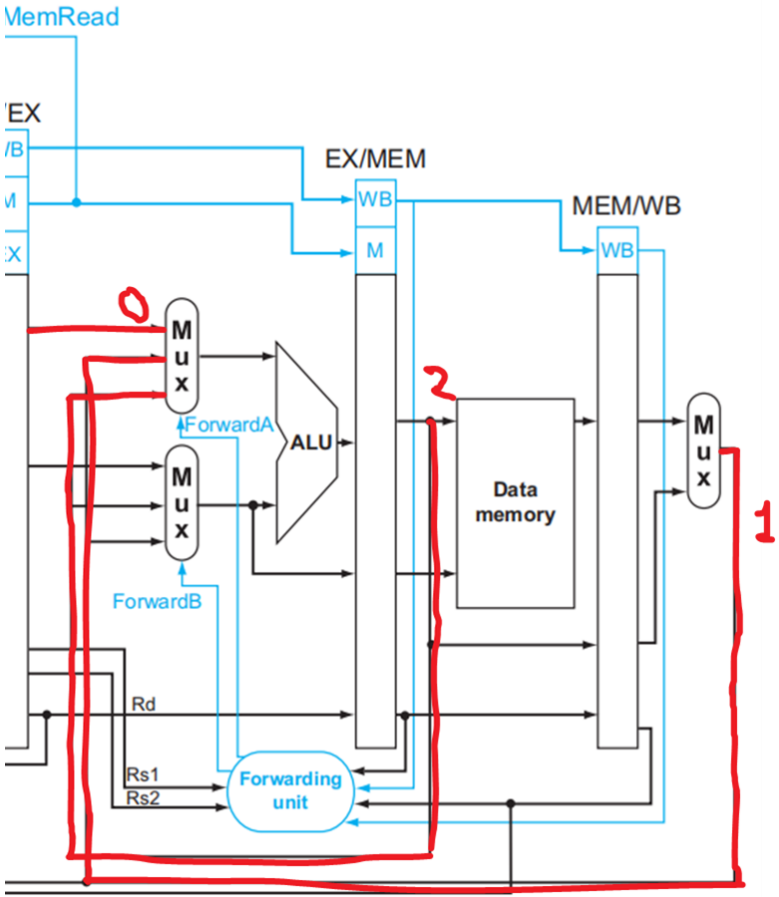
Data Hazard & Forwarding

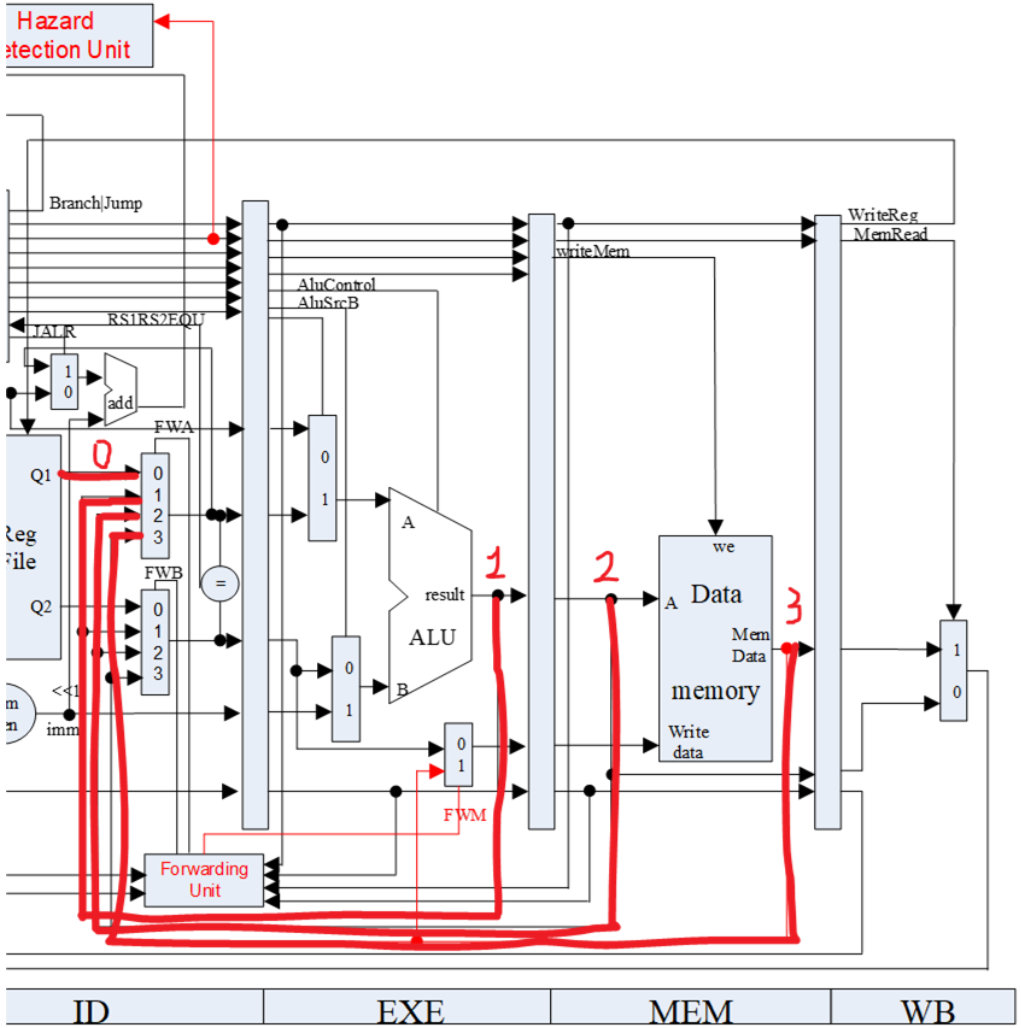
```
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd ≠ 0)
and not(EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
     and (EX/MEM.RegisterRd = ID/EX.RegisterRs1))
and (MEM/WB.RegisterRd = ID/EX.RegisterRs1))
     ForwardA = 01
```

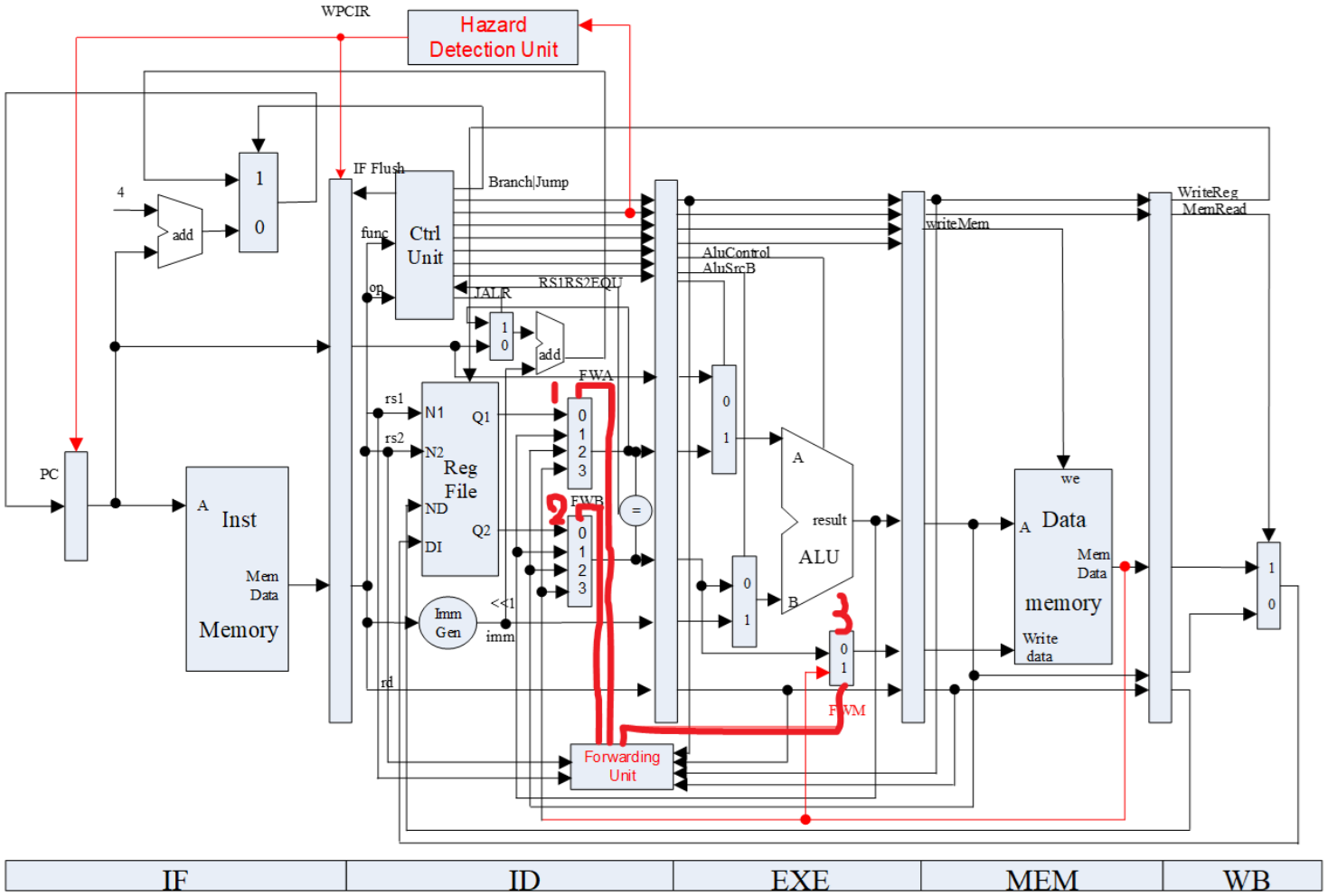
```
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd ≠ 0)
and not(EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
     and (EX/MEM.RegisterRd = ID/EX.RegisterRs2))
and (MEM/WB.RegisterRd = ID/EX.RegisterRs2))
     ForwardB = 01
```




WPCIR
Hazard Detection Unit
IF Flush
Branch|Jump
4
add
1
0
func
Ctrl Unit
op
JALR
RS1RS2EQU
AluControl
AluSrcB
writeMem
WriteReg
MemRead
FWA
FWB
rs1
rs2
N1
N2
ND
DI
Q1
Q2
Reg File
PC
A
Inst
Mem Data
Memory
Imm Gen
<<1
imm
rd
=
ALU
result
B
we
Data
memory
Write data
FWM
Forwarding Unit
IF
ID
EXE
MEM
WB


MemRead
EX
EX/MEM
MEM/WB
WB
M
WB
M u x
ForwardA
ALU
M u x
ForwardB
Data memory
M u x
Rd
Rs1
Rs2
Forwarding unit
0
2
1


Hazard etection Unit
Branch|Jump
JALR
RS1RS2EQU
add
FWA
Q1
Reg File
Q2
FWB
=
<<1
imm
AluControl
AluSrcB
writeMem
WriteReg
MemRead
A
result
ALU
B
we
A
Data
Mem Data
memory
Write data
FWM
Forwarding Unit
0
1
2
3
1
2
3
ID
EXE
MEM
WB


WPCIR
Hazard Detection Unit
IF Flush
Branch|Jump
4
add
1
0
func
op
Ctrl Unit
JALR
RS1RS2EQU
AluControl
AluSrcB
writeMem
WriteReg
MemRead
FWA
FWB
rs1
rs2
N1
N2
ND
DI
Q1
Q2
Reg File
PC
A
Inst
Mem Data
Memory
Imm Gen
<<1
imm
rd
=
0
1
2
3
A
result
ALU
B
we
A
Data
Mem Data
memory
Write data
FWM
Forwarding Unit
IF
ID
EXE
MEM
WB

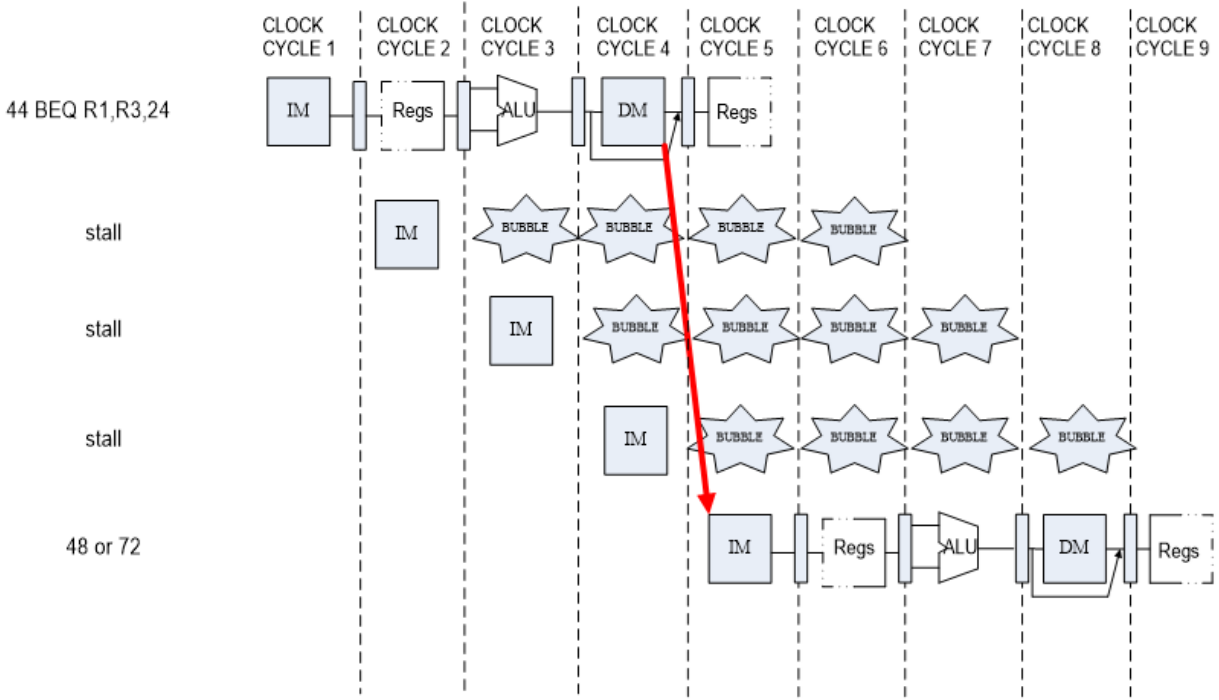
Data Hazard & Forwarding – stall

- PC_EN -> false
- IF/ID stall
- ID/EX flush

Control Hazard
& Branch Prediction

Control Hazard

- Freeze or flush the pipeline → inefficient

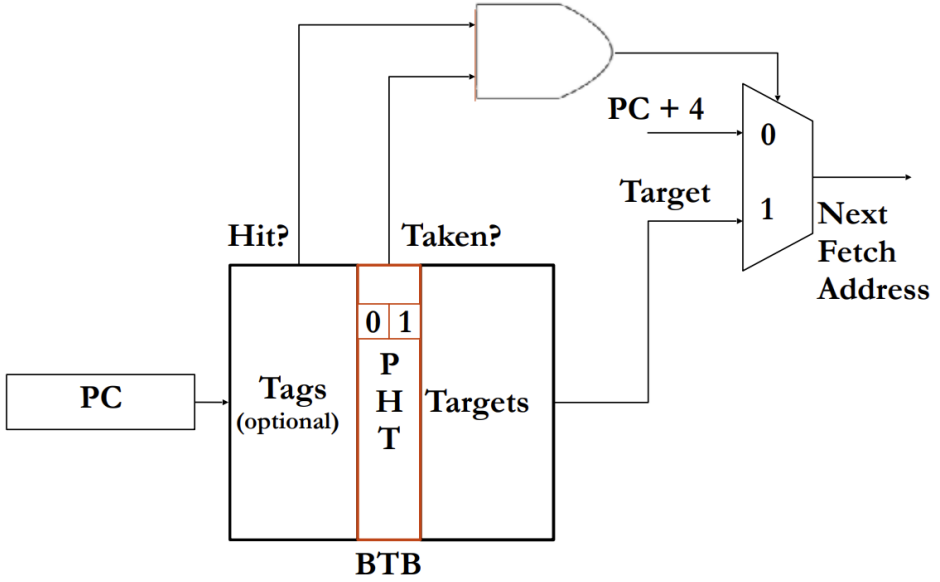


Control Hazard

Branch Prediction!

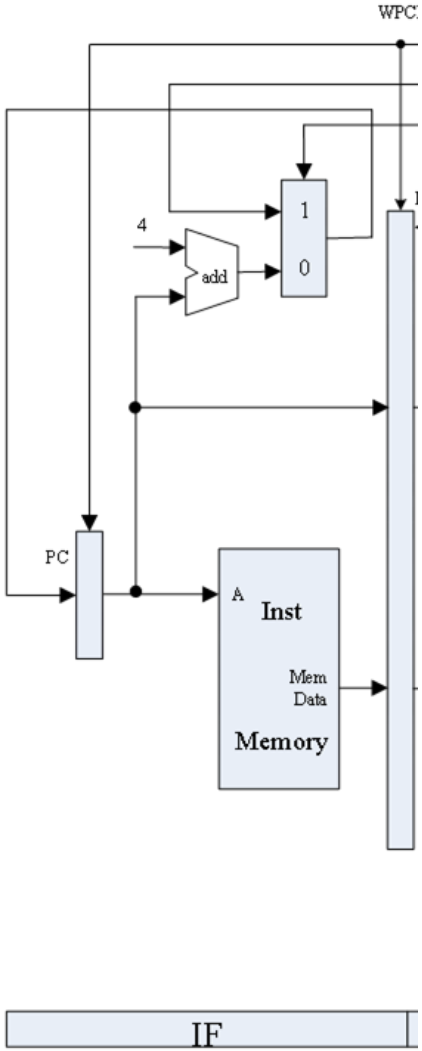
predict direction: taken or not taken

predict target: If taken, target addr ?



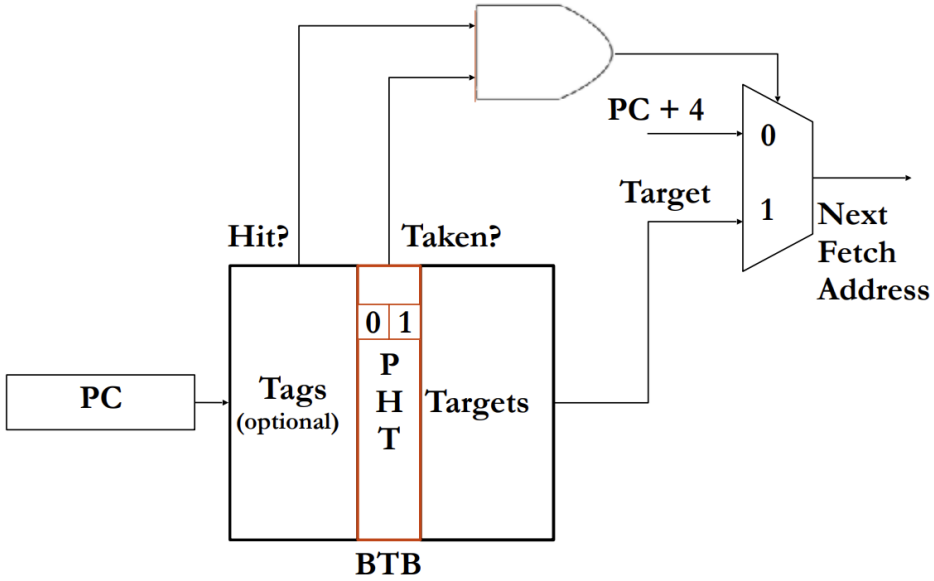
Branch Prediction

- Predict-not-taken
  - **predict direction**: not taken
  - **predict target**



Branch Prediction

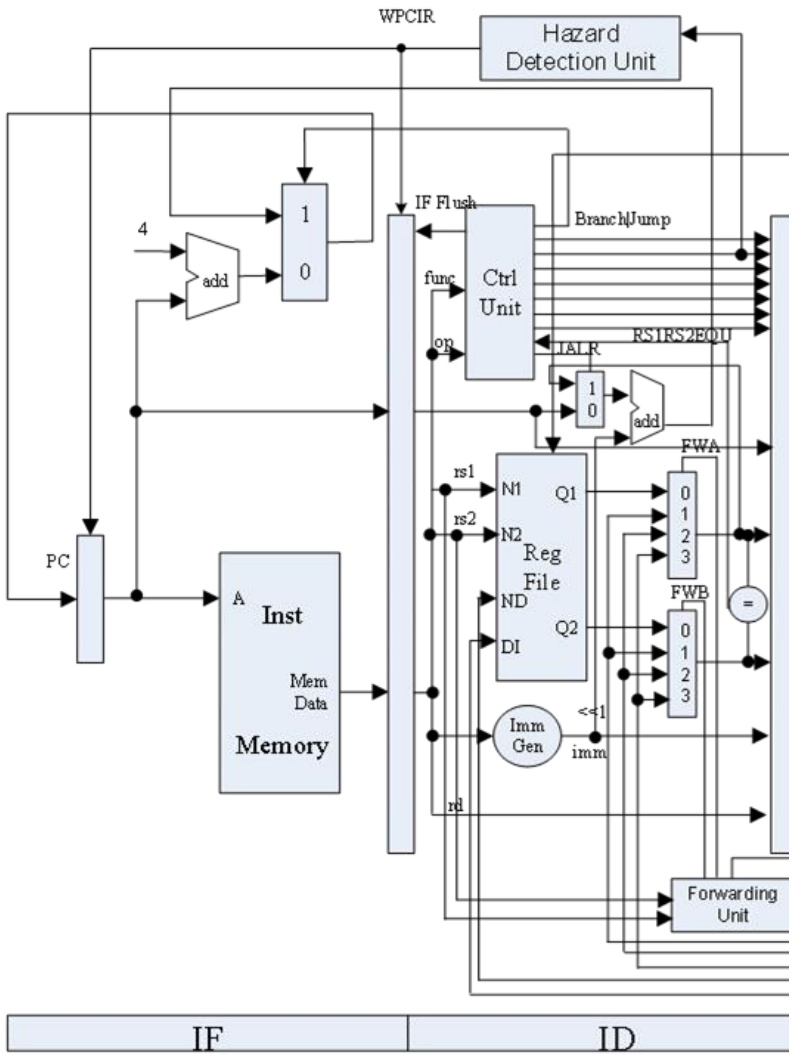
- Predict-taken ?
  - **predict direction**: taken
  - **predict target**: jump addr



Branch Prediction

- Predict-not-taken
  - Rollback → next insn is incorrect

Flush IF/ID pipeline register



Hazard Detection Unit

Hazard Detection Unit

```
module HazardDetectionUnit(
    input clk,
    input Branch_ID, rs1use_ID, rs2use_ID,
    input[1:0] hazard_optype_ID,
    input[4:0] rd_EXE, rd_MEM, rs1_ID, rs2_ID, rs2_EXE,
    output PC_EN_IF, reg_FD_EN, reg_FD_stall, reg_FD_flush,
        reg_DE_EN, reg_DE_flush, reg_EM_EN, reg_EM_flush, reg_MW_EN,
    output forward_ctrl_ls,
    output[1:0] forward_ctrl_A, forward_ctrl_B
);
```

ID stage: ALU/Load/Store ?

```
parameter hazard_optype_ALU = 2'd1;
parameter hazard_optype_LOAD = 2'd2;
parameter hazard_optype_STORE = 2'd3
```

Hazard Detection Unit – forwarding

```
reg[1:0] hazard_optype_EXE, hazard_optype_MEM;
always@(posedge clk) begin
    hazard_optype_MEM <= hazard_optype_EXE & {2{~reg_EM_flush}};
    hazard_optype_EXE <= hazard_optype_ID & {2{~reg_DE_flush}};
end

parameter hazard_optype_ALU = 2'd1;
parameter hazard_optype_LOAD = 2'd2;
parameter hazard_optype_STORE = 2'd3;

wire rs1_forward_1     = ... && hazard_optype_EXE == hazard_optype_ALU;
wire rs1_forward_stall = ...&& hazard_optype_EXE == hazard_optype_LOAD &&
hazard_optype_ID != hazard_optype_STORE;
wire rs1_forward_2     = ... && hazard_optype_MEM == hazard_optype_ALU;
wire rs1_forward_3     = ... && hazard_optype_MEM == hazard_optype_LOAD;

assign forward_ctrl_A = ...
assign forward_ctrl_B = ...
assign forward_ctrl_ls ...
```

Hazard Detection Unit – hazard

```
assign PC_EN_IF = ...;
assign reg_FD_stall = ...;
assign reg_FD_flush = ...;
assign reg_DE_flush = ...;
```

ROM

NO.	Instruction	Addr.	Label	ASM	Comment
0	00000013	0	__start:	addi x0, x0, 0	
1	00402103	4		lw x2, 4(x0)	
2	00802203	8		lw x4, 8(x0)	
3	004100b3	C		add x1, x2, x4	
4	fff08093	10		addi x1, x1, -1	
5	00c02283	14		lw x5, 12(x0)	
6	01002303	18		lw x6, 16(x0)	
7	01402383	1C		lw x7, 20(x0)	
8	402200b3	20		sub x1,x4,x2	
9	002270b3	24		and x1,x4,x2	
10	002260b3	28		or x1,x4,x2	
11	002240b3	2C		xor x1,x4,x2	
12	002210b3	30		sll x1,x4,x2	
13	002220b3	34		slt x1,x4,x2	
14	004120b3	38		slt x1,x2,x4	

ROM

NO.	Instruction	Addr.	Label	ASM	Comment
15	002350b3	3C		srl x1, x6, x2	
16	402350b3	40		sra x1, x6, x2	
17	4023d0b3	44		sra x1, x7, x2	
18	007330b3	48		sltu x1, x6, x7	
19	0063b0b3	4C		sltu x1, x7, x6	
20	00000033	50		add x0,x0,x0	
21	ffd50093	54		addi x1,x10,-3	
22	00f27093	58		andi x1,x4,15	
23	00f26093	5C		ori x1,x4,15	
24	00f24093	60		xori x1,x4,15	
25	00f22093	64		slti x1,x4,15	
26	00121093	68		slli x1,x4,1	
27	00225093	6C		srli x1,x4,2	
28	40c35093	70		srai x1, x6, 12	
29	fff33093	74		sltiu x1, x6, -1	

ROM

NO.	Instruction	Addr.	Label	ASM	Comment
30	fff3b093	78		sltiu x1, x7, -1	
31	00520863	7C		beq x4,x5,label0	
32	00420663	80		beq x4,x4,label0	
33	00000013	84		addi x0,x0,0	
34	00000013	88		addi x0,x0,0	
35	00421863	8C	label0:	bne x4,x4,label1	
36	00521663	90		bne x4,x5,label1	
37	00000013	94		addi x0,x0,0	
38	00000013	98		addi x0,x0,0	
39	0042c863	9C	label1:	blt x5,x4,label2	
40	00524663	A0		blt x4,x5,label2	
41	00000013	A4		addi x0,x0,0	
42	00000013	A8		addi x0,x0,0	
43	00736863	AC	label2:	bltu x6,x7,label3	
44	0063e663	B0		bltu x7,x6,label3	

ROM

NO.	Instruction	Addr.	Label	ASM	Comment
45	00000013	B4		addi x0,x0,0	
46	00000013	B8		addi x0,x0,0	
47	00525863	BC	label3:	bge x4,x5,label4	
48	0042d663	C0		bge x5,x4,label4	
49	00000013	C4		addi x0,x0,0	
50	00000013	C8		addi x0,x0,0	
51	0063f863	CC	label4:	bgeu x7,x6,label5	
52	00737663	D0		bgeu x6,x7,label5	
53	00000013	D4		addi x0,x0,0	
54	00000013	D8		addi x0,x0,0	
55	00425663	DC	label5:	bge x4,x4,label6	
56	00000013	E0		addi x0,x0,0	
57	00000013	E4		addi x0,x0,0	
58	000040b7	E8	label6:	lui x1,4	
59	00c000ef	EC		jal x1,12	

ROM

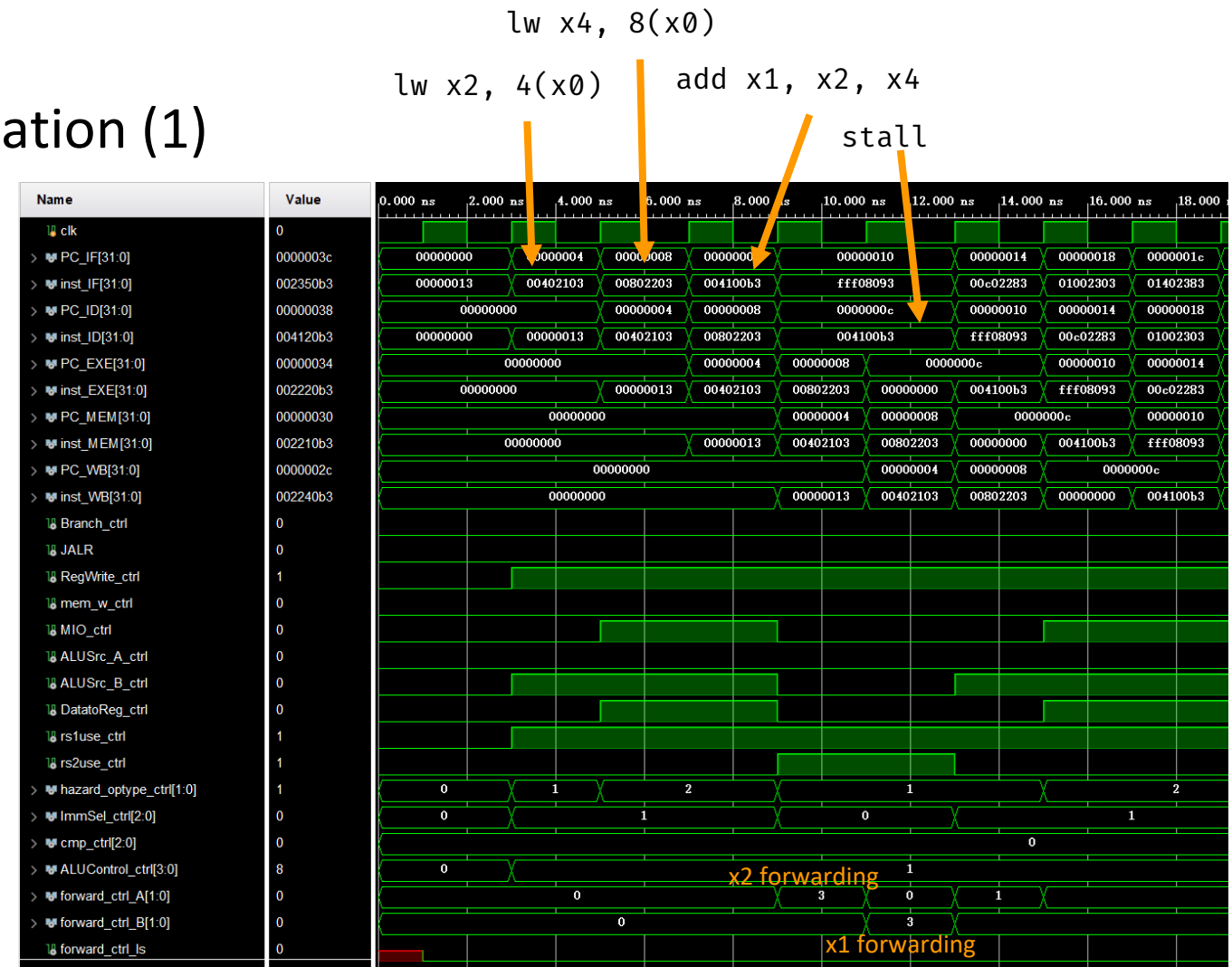
NO.	Instruction	Addr.	Label	ASM	Comment
60	00000013	F0		addi x0,x0,0	
61	00000013	F4		addi x0,x0,0	
62	01802403	F8		lw x8, 24(x0)	
63	00802e23	FC		sw x8, 28(x0)	
64	01c02083	100		lw x1, 28(x0)	
65	02801023	104		sh x8, 32(x0)	
66	02002083	108		lw x1, 32(x0)	
67	02800223	10C		sb x8, 36(x0)	
68	02402083	110		lw x1, 36(x0)	
69	01a01083	114		lh x1, 26(x0)	
70	01a05083	118		lhu x1, 26(x0)	
71	01b00083	11C		lb x1, 27(x0)	
72	01b04083	120		lbu x1, 27(x0)	
73	ffff0097	124		auipc x1, 0xffff0	
74	000000e7	128		jalr x1,0(x0)	

RAM

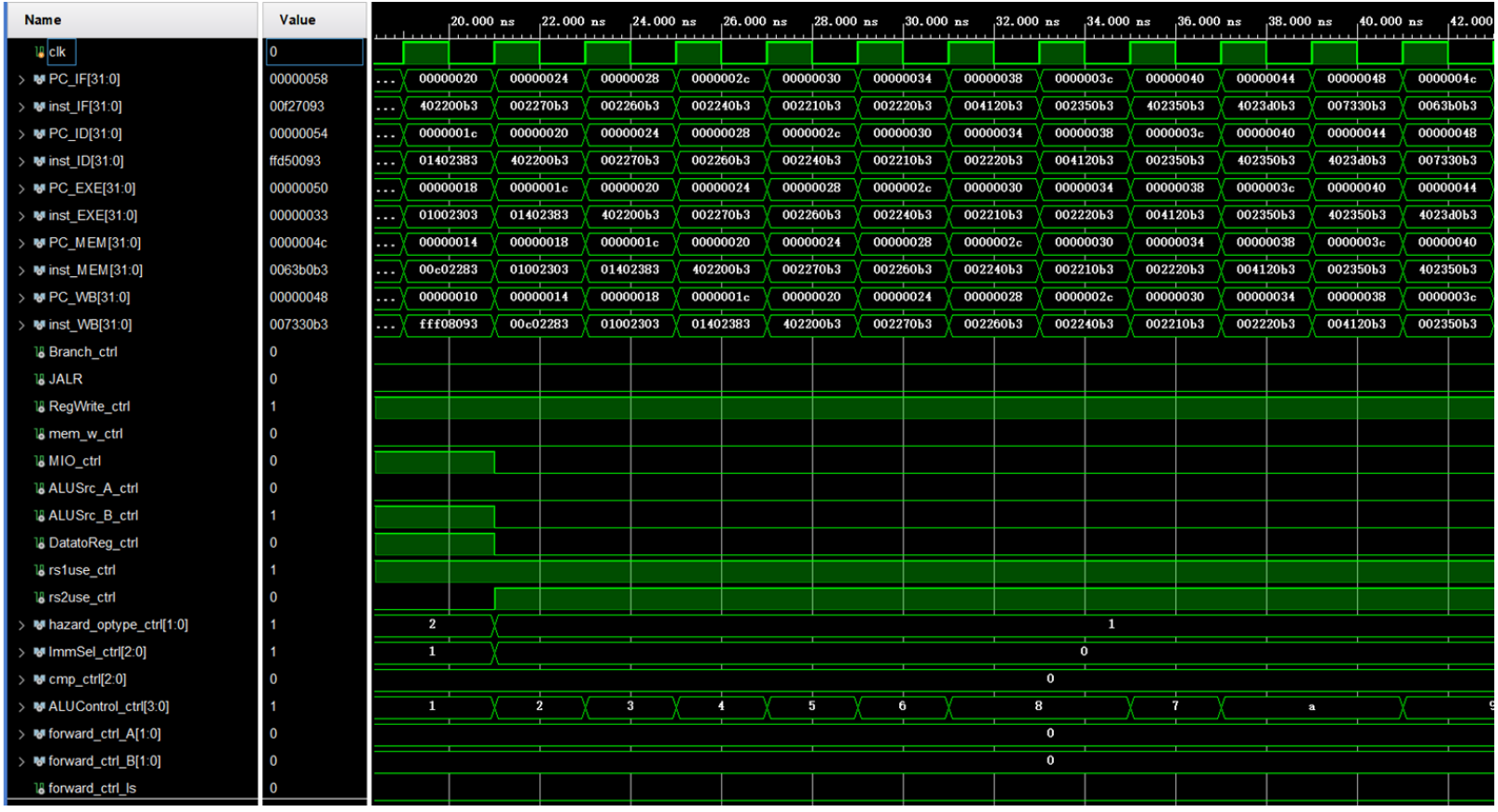
NO.	Data	Addr.
0	000080BF	0
1	00000008	4
2	00000010	8
3	00000014	C
4	FFFF0000	10
5	0FFF0000	14
6	FF000F0F	18
7	F0F0F0F0	1C
8	00000000	20
9	00000000	24
10	00000000	28
11	00000000	2C
12	00000000	30
13	00000000	34
14	00000000	38
15	00000000	3C

NO.	Instruction	Addr.
16	00000000	40
17	00000000	44
18	00000000	48
19	00000000	4C
20	A3000000	50
21	27000000	54
22	79000000	58
23	15100000	5C
24	00000000	60
25	00000000	64
26	00000000	68
27	00000000	6C
28	00000000	70
29	00000000	74
30	00000000	78
31	00000000	7C

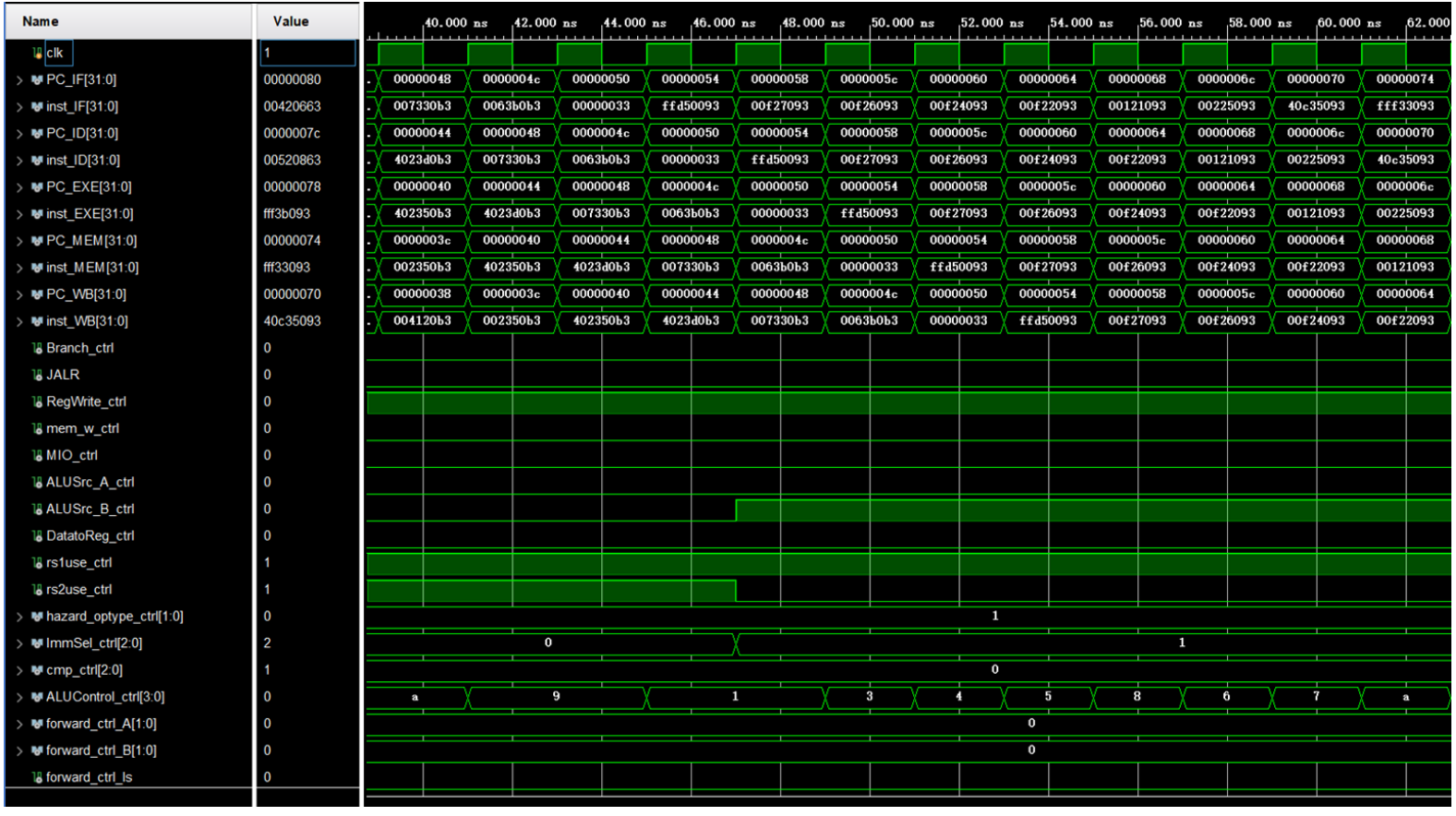
Simulation (1)



Simulation (2)



Simulation (3)

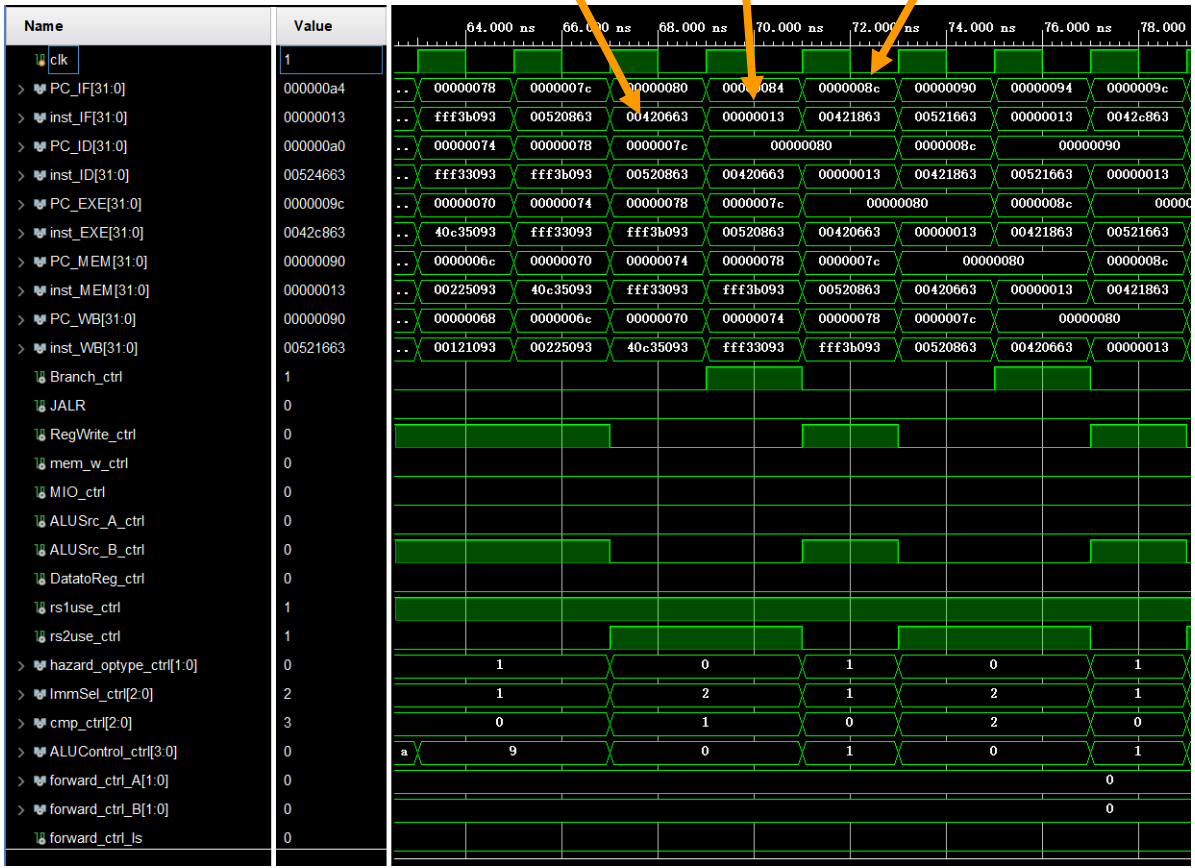


Simulation (4)

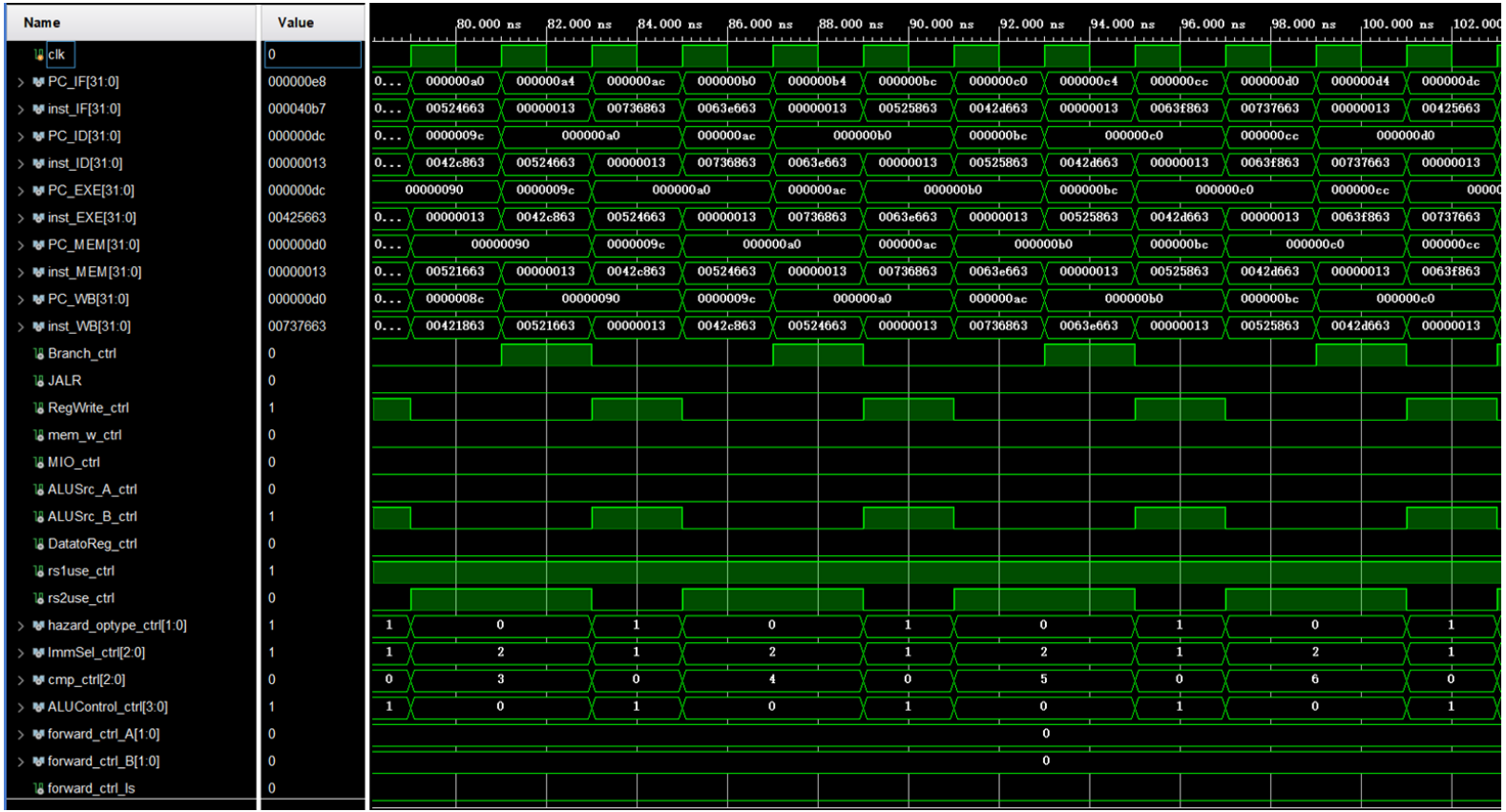
`beq x4, x4, label0`

jump to label0 & flush IF/ID

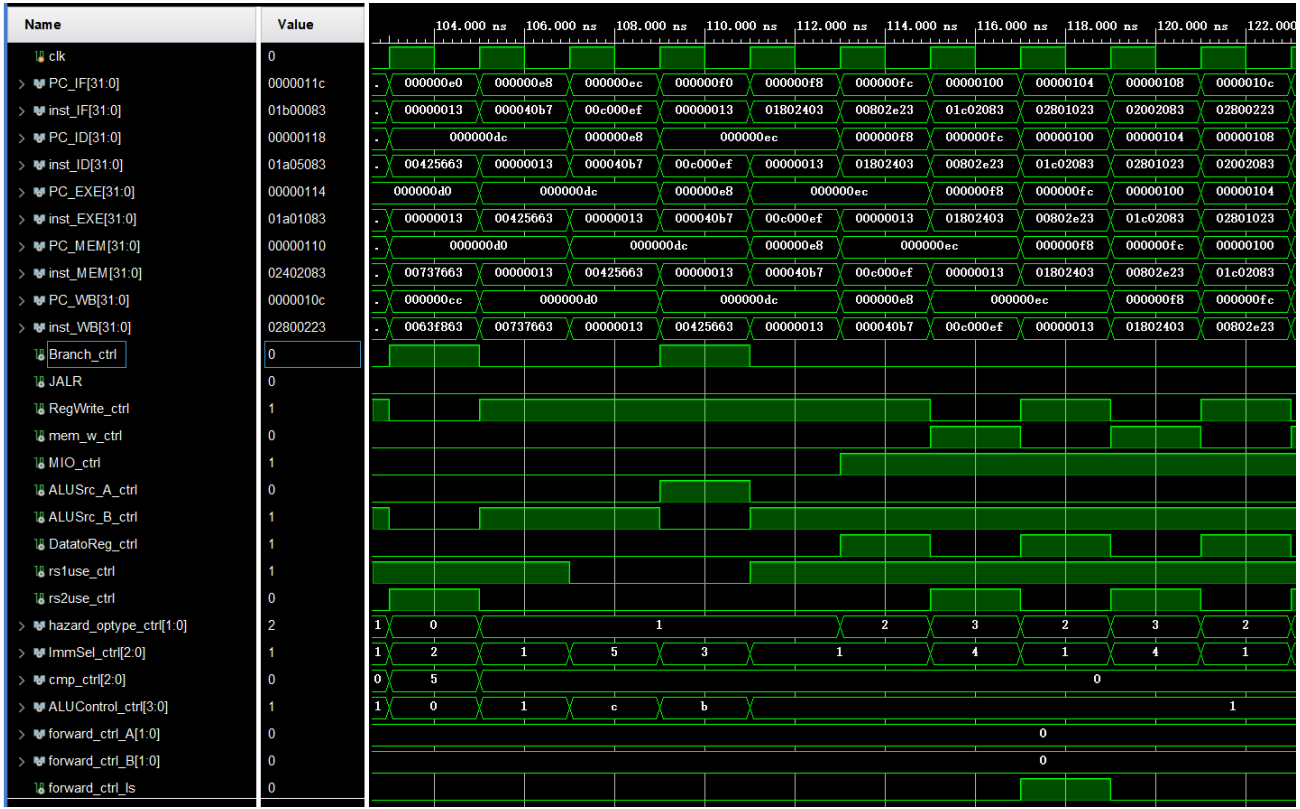
predict-not-taken



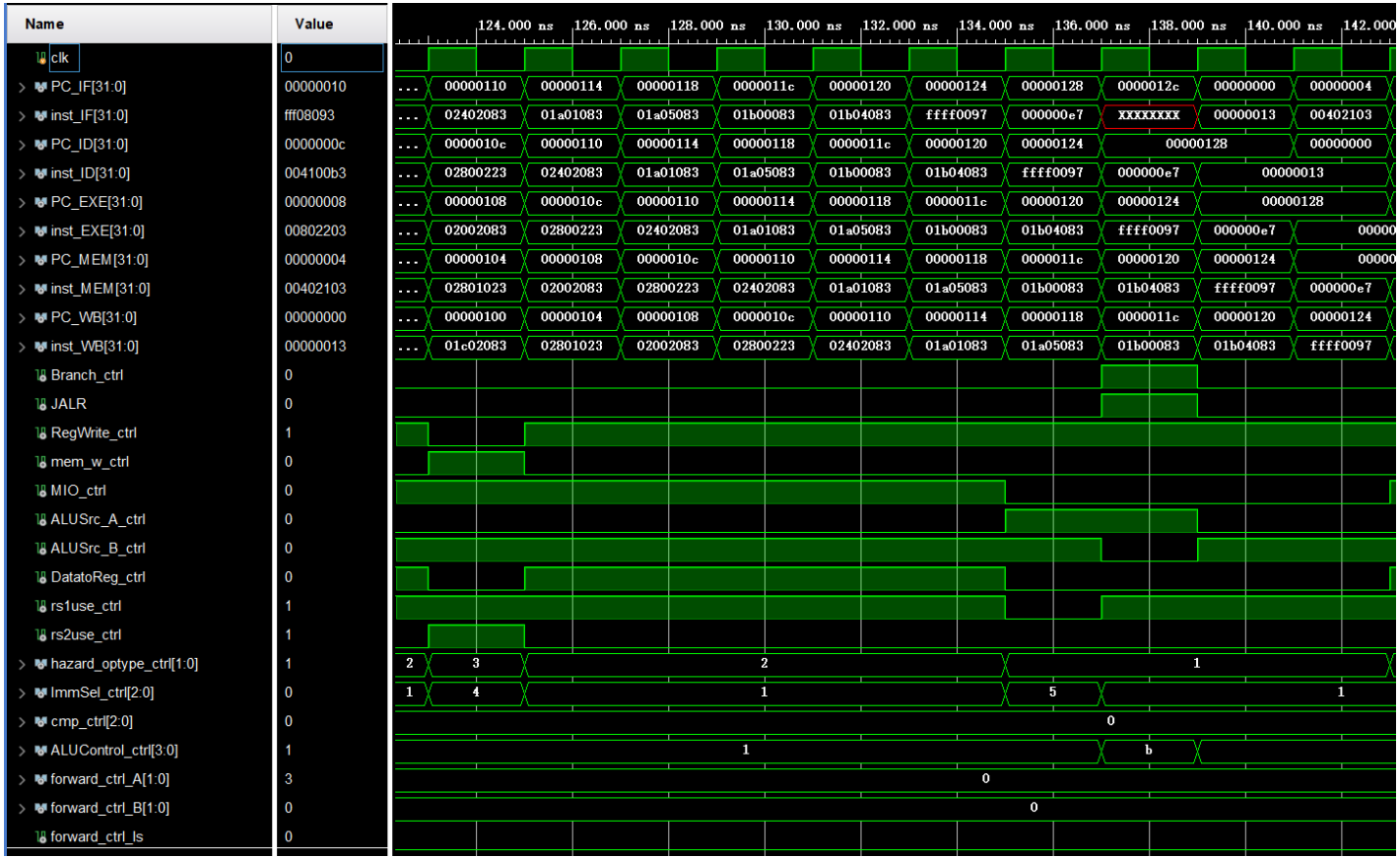
Simulation (5)

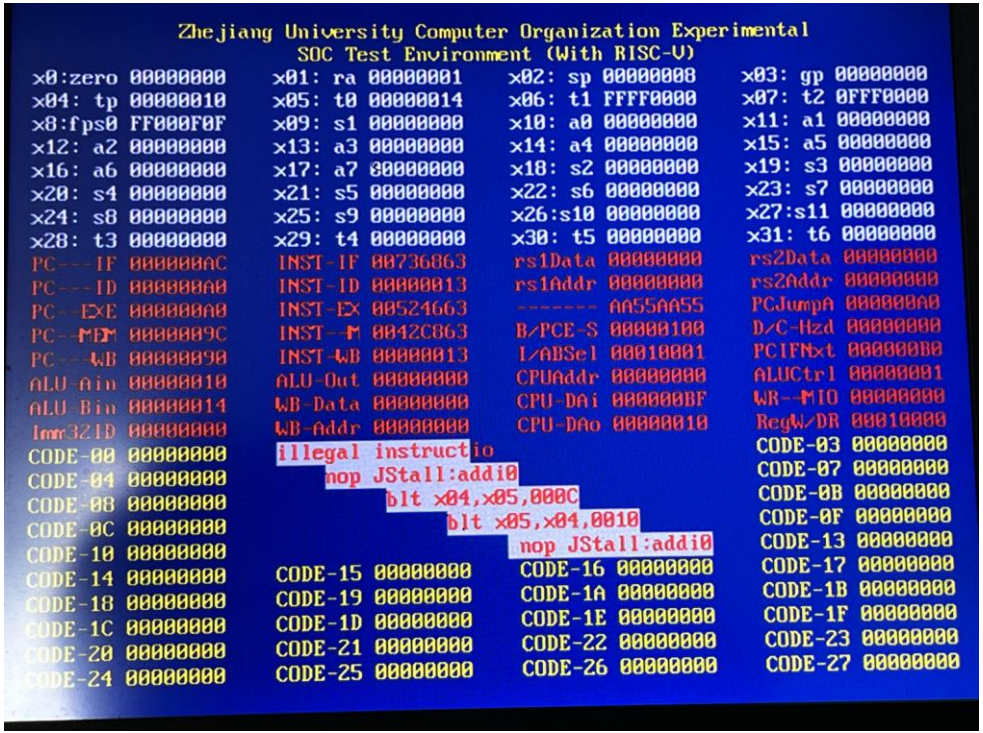


Simulation (6)



Simulation (7)





- Single Step Debug:
  - SW[0] = 1
  - Press BTNX4Y0
- Illegal Instruction
  - Complete Code2Inst.v

References

- https://riscv.org/wp-content/uploads/2018/05/13.15-13-50-Talk-riscv-base-isa-20180507.pdf
- Computer Organization and Design