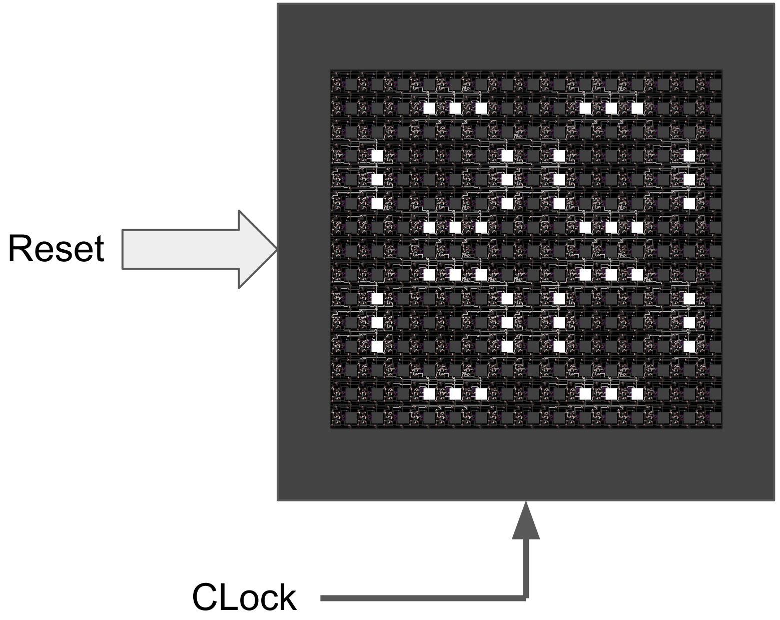
IITB_RISC_32
Six Stage Pipelined Processor

Manish Prajapati *213070074*	**Ajay Verma** *213070059*
Amit Kumar Singh *213070069*	**Subham Singh** *213070076*

Department of Electrical Engineering
IIT Bombay



- IITB-RISC is six stage pipelined processor. It is a 16-bit computer system with 8 registers (R0 to R7).
- It includes hazard mitigation techniques, such as data forwarding and branch prediction.
- Each address corresponds to two bytes of main memory.
- This architecture uses two conditional code registers carry flag (C) and zero flag (Z).
- There are three machine-code instruction formats (R, I, and J type) and a total of 14 instructions.

R-Type	Opcode (4-bit)	RA (3-bit)	RB (3-bit)	RC (3-bit)	Unused (1-bit)	Condition (2-bit)

I-Type	Opcode (4-bit)	RA (3-bit)	RC (3-bit)	Immediate (6-bits)

J-Type	Opcode (4-bit)	RA (3-bit)	Immediate (9-bits)

Instruction Encoding

ADD	00_01	RA	RB	RC	0	00
ADC	00_01	RA	RB	RC	0	10
ADZ	00_01	RA	RB	RC	0	01
ADL	00_01	RA	RB	RC	0	11

ADI	00_00	RA	RB	6 Bit Immediate

NDU	00_10	RA	RB	RC	0	00
NDC	00_10	RA	RB	RC	0	10
NDZ	00_10	RA	RB	RC	0	01

Instruction Encoding

LW	01_00	RA	RB	6 Bit Immediate

SW	01_01	RA	RB	6 Bit Immediate

BEQ	10_00	RA	RB	6 Bit Immediate
JAL	10_01	RA	9 Bit Immediate	
JLR	10_10	RA	RB	000_000
JRI	10_11	RA	9 Bit Immediate	

NOPE	11_11	XXXX

Source
Destination

Instruction Description

Mnemonic	Name & Format	Assembly	Action
ADD	ADD (R)	add rc, ra, rb	Add content of regB to regA and store result in regC. *It modifies C and Z flags.*
ADC	Add if carry set (R)	adc rc, ra, rb	Add content of regB to regA and store result in regC, if carry flag is set. *It modifies C and Z flags.*
ADZ	Add if zero set (R)	adz rc, ra, rb	Add content of regB to regA and store result in regC, if zero flag is set. *It modifies C and Z flags.*
ADL	Add with one bit left shift of RB (R)	adl rc, ra, rb	Add content of regB (after one bit left shift) to regA and store result in regC *It modifies C and Z flags.*
ADI	Add immediate (I)	adi rb, ra, imm6	Add content of regA with Imm (sign extended) and store result in regB. *It modifies C and Z flags.*

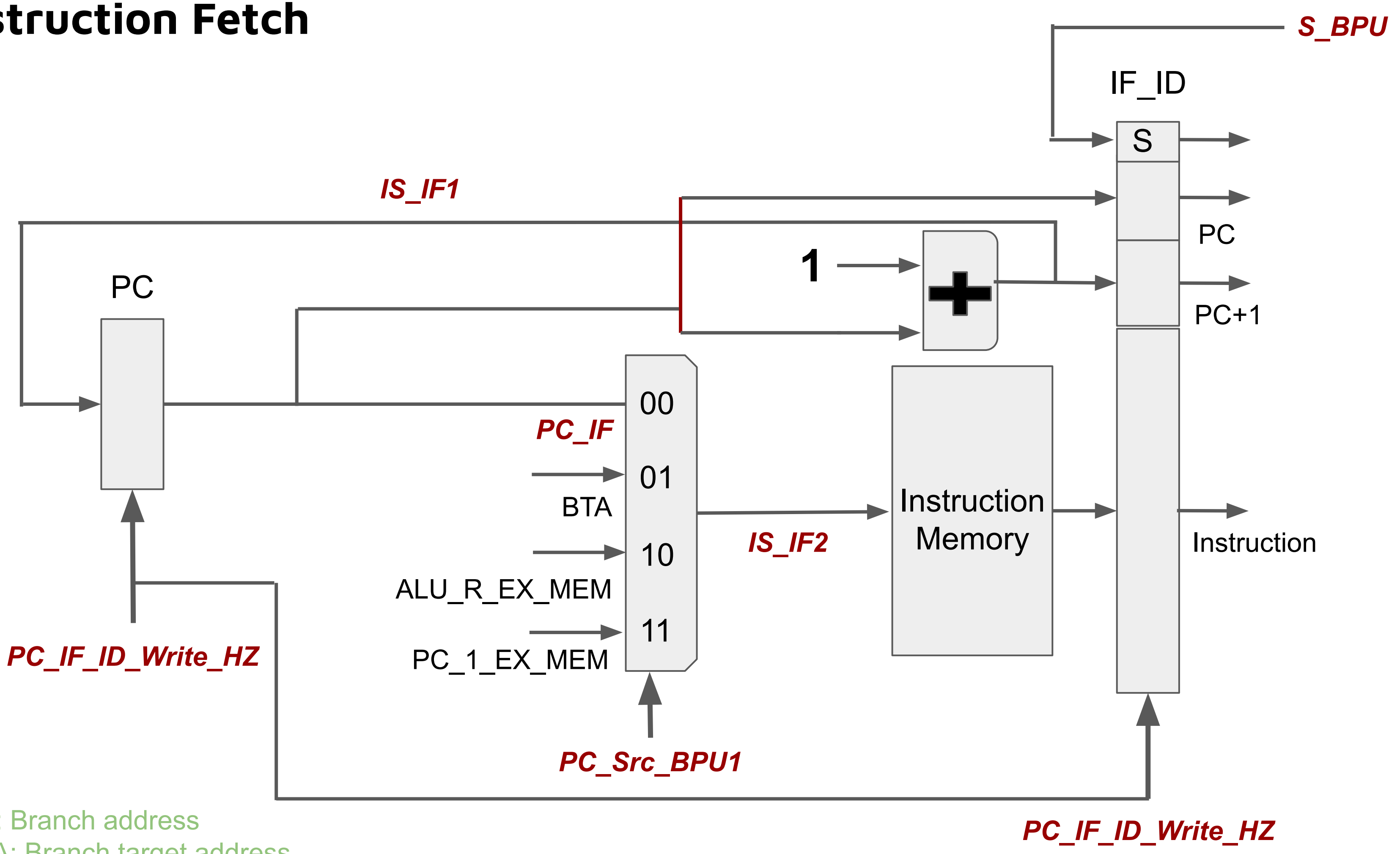
Instruction Description

Mnemonic	Name & Format	Assembly	Action
NDU	Nand (R)	ndu rc, ra, rb	NAND the content of regB to regA and store result in regC. *It modifies Z flags.*
NDC	Nand if carry set (R)	ndc rc, ra, rb	NAND the content of regB to regA and store result in regC if carry flag is set. *It modifies Z flags.*
NDZ	Nand if zero se (R)t	ndz rc, ra, rb	NAND the content of regB to regA and store result in regC if zero flag is set. *It modifies Z flags.*
LW	Load (I)	lw ra, rb, imm6	Load value from memory into reg A. Memory address is formed by adding immediate 6 bits with content of red B. *It modifies Z flags.*
SW	Store (I)	sw ra, rb, imm6	Store value from reg A into memory. Memory address is formed by adding immediate 6 bits with content of red B.

Instruction Description

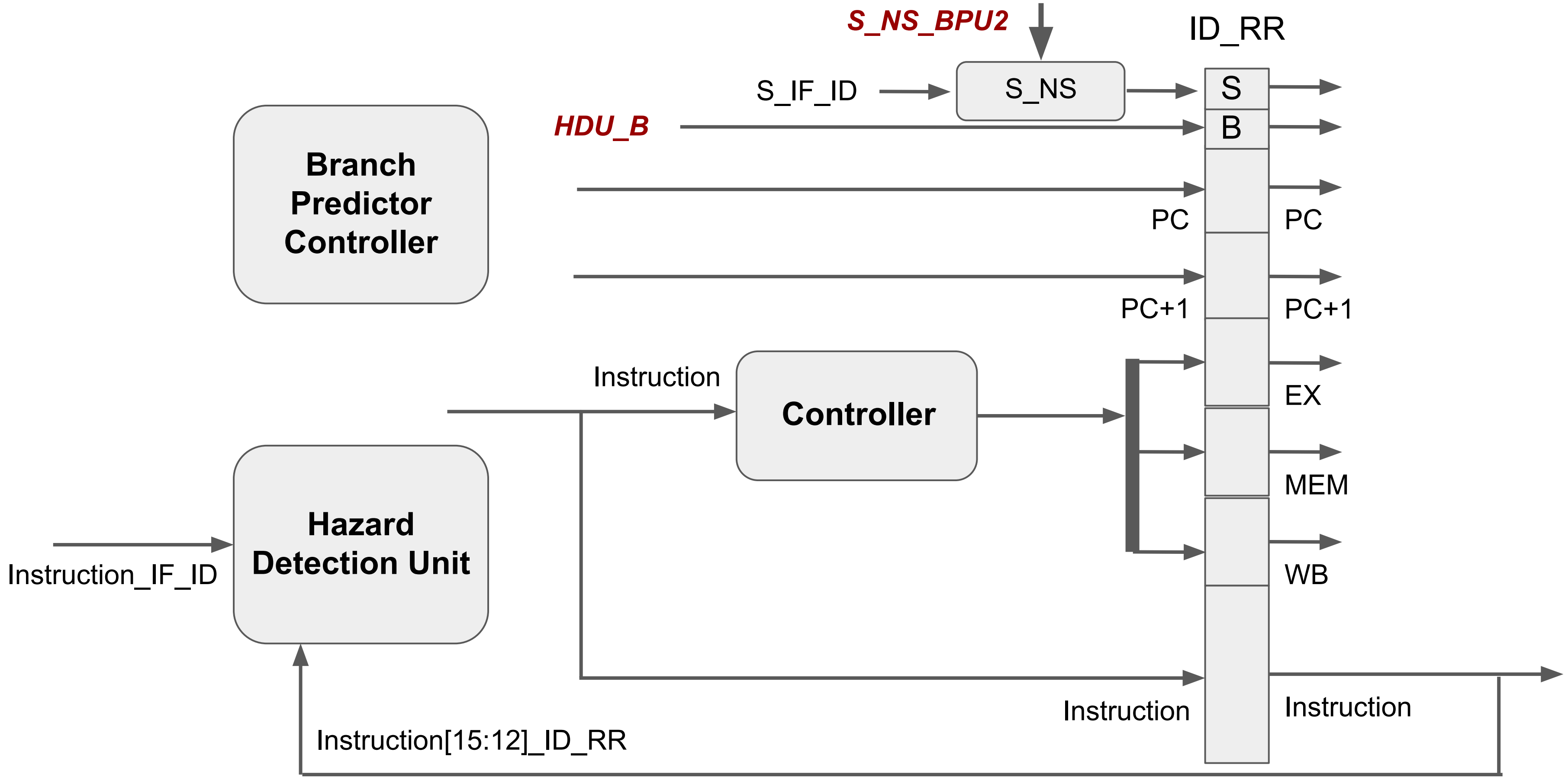
Mnemonic	Name & Format	Assembly	Action
BEQ	Branch on equality (I)	jeq ra, rb, imm6	If content of reg A and reg B are the same, branch to PC+Imm, where PC is the address of beq instruction.
JAL	Jump and link (I)	jalr ra, imm9	Branch to the address PC+ Imm. Store PC+1 into regA, where PC is the address of the jalr instruction.
JLR	Jump and link to register (I)	jalr ra, rb	Branch to the address in regB. Store PC+1 into regA, where PC is the address of the jalr instruction.
JRI	Jump to register (J)	jri ra, imm9	Branch to memory location given by the RA + Imm.

Instruction Fetch

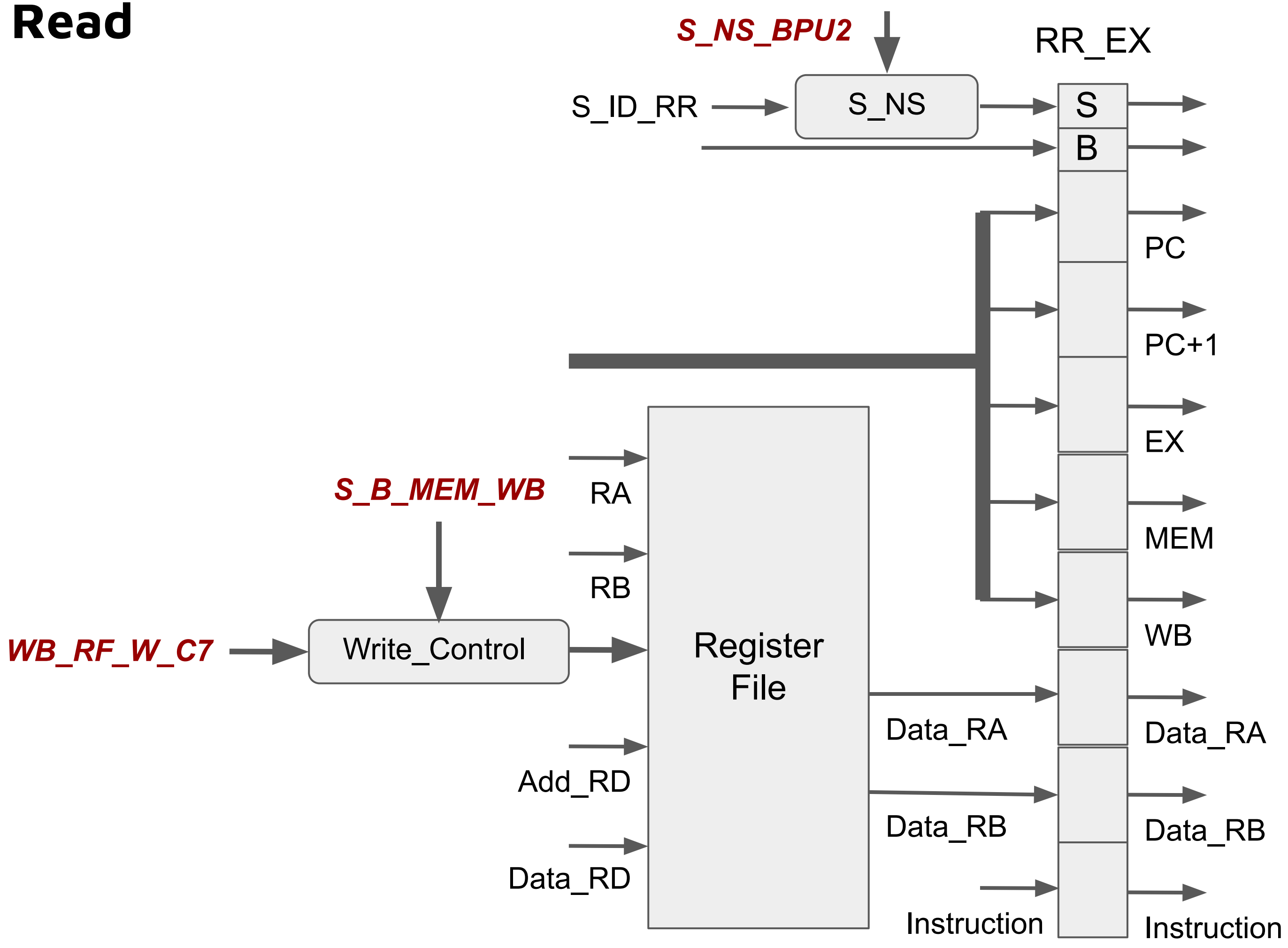


// BA: Branch address
// BTA: Branch target address

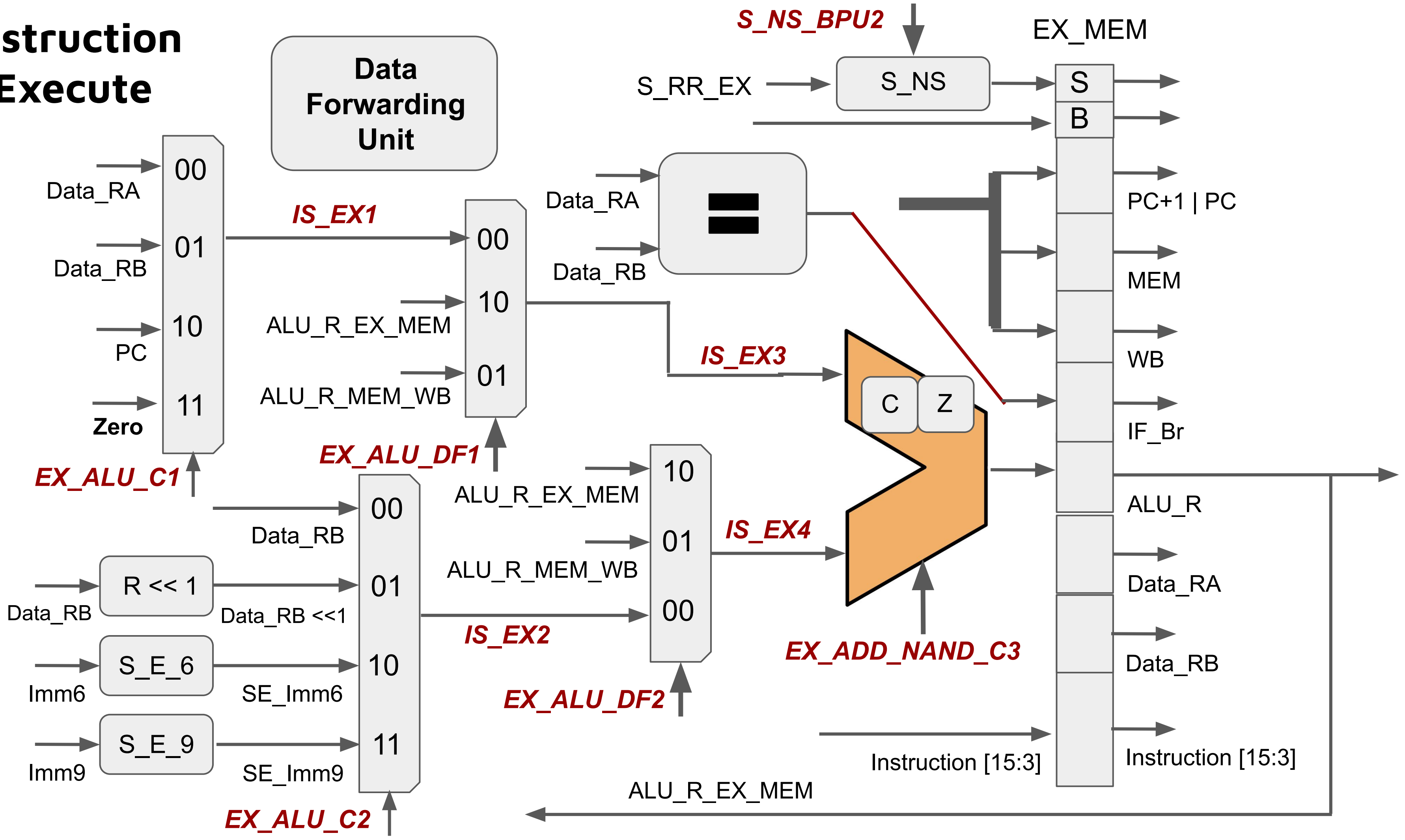
Instruction Decode



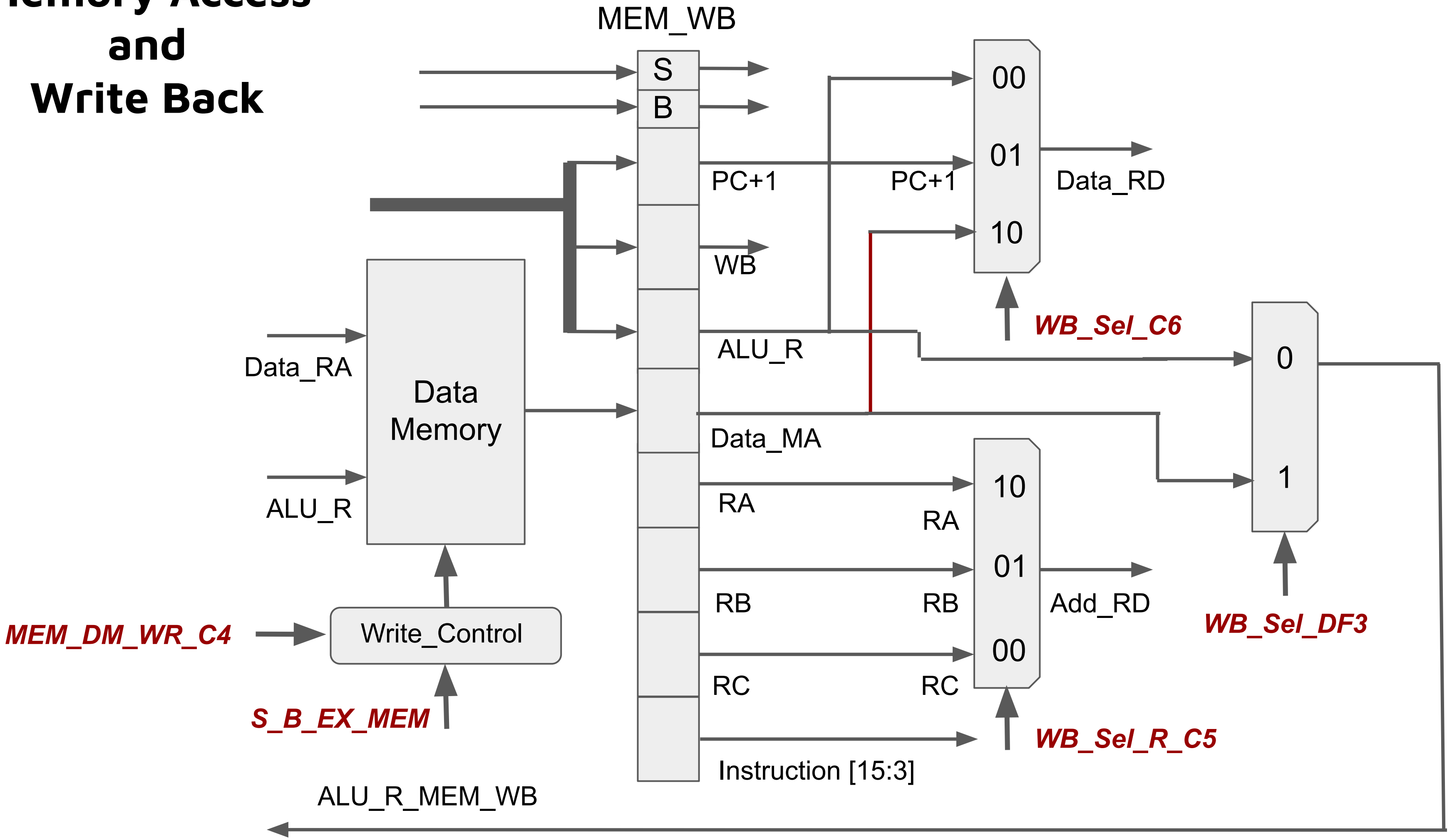
Register Read



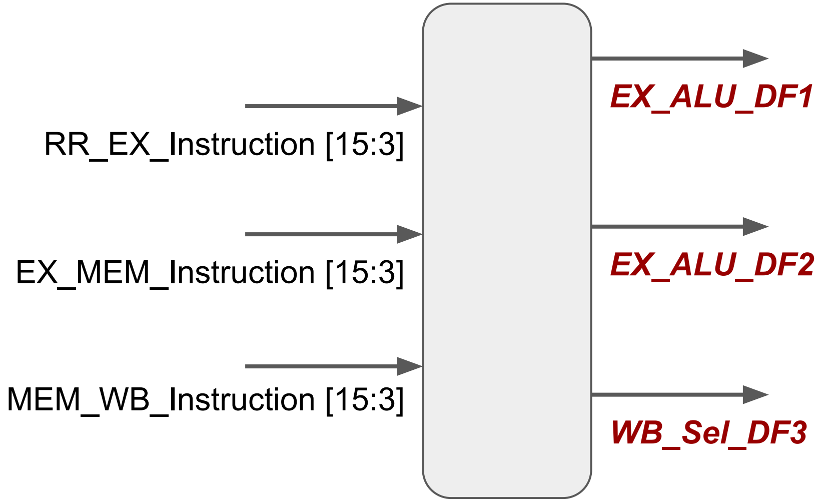
Instruction Execute



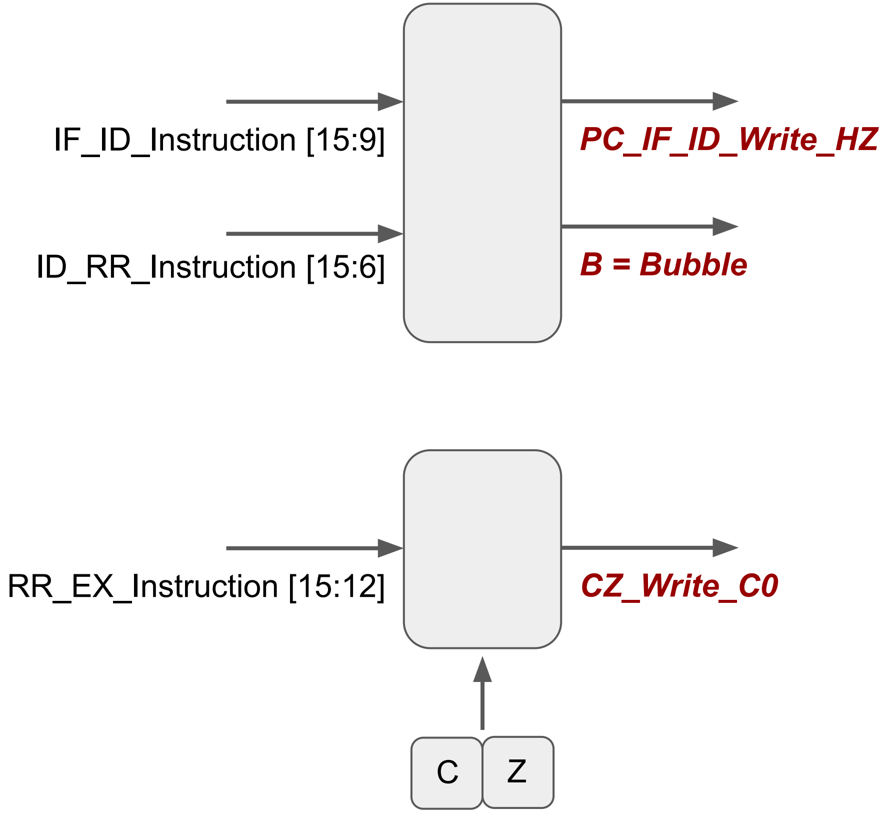
Memory Access and Write Back



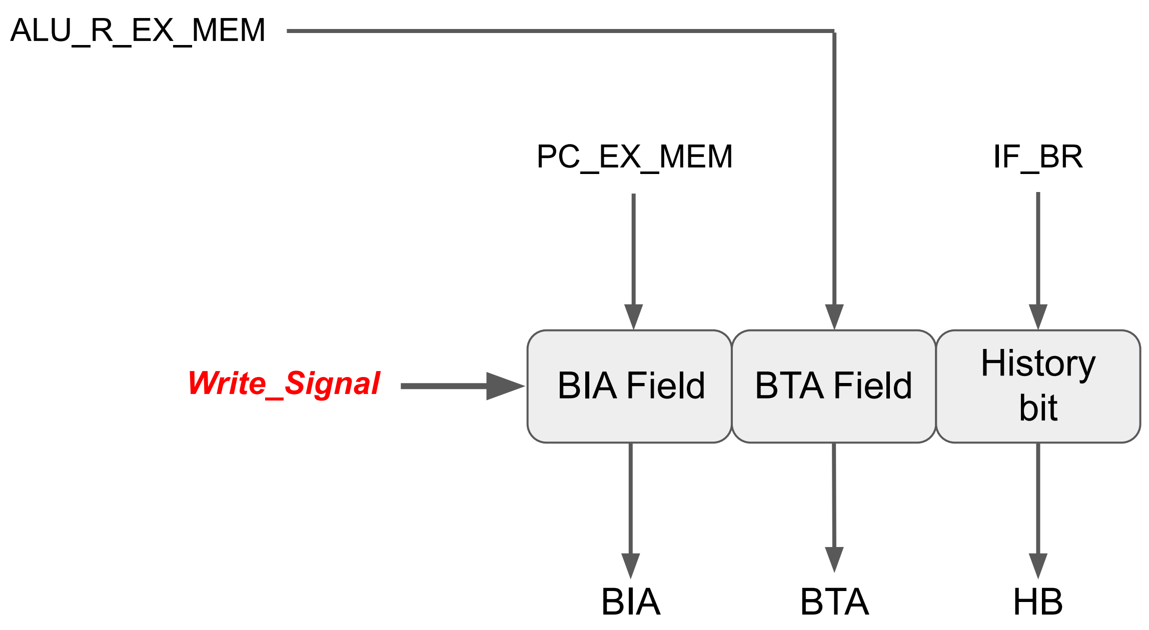
Data Forwarding Unit



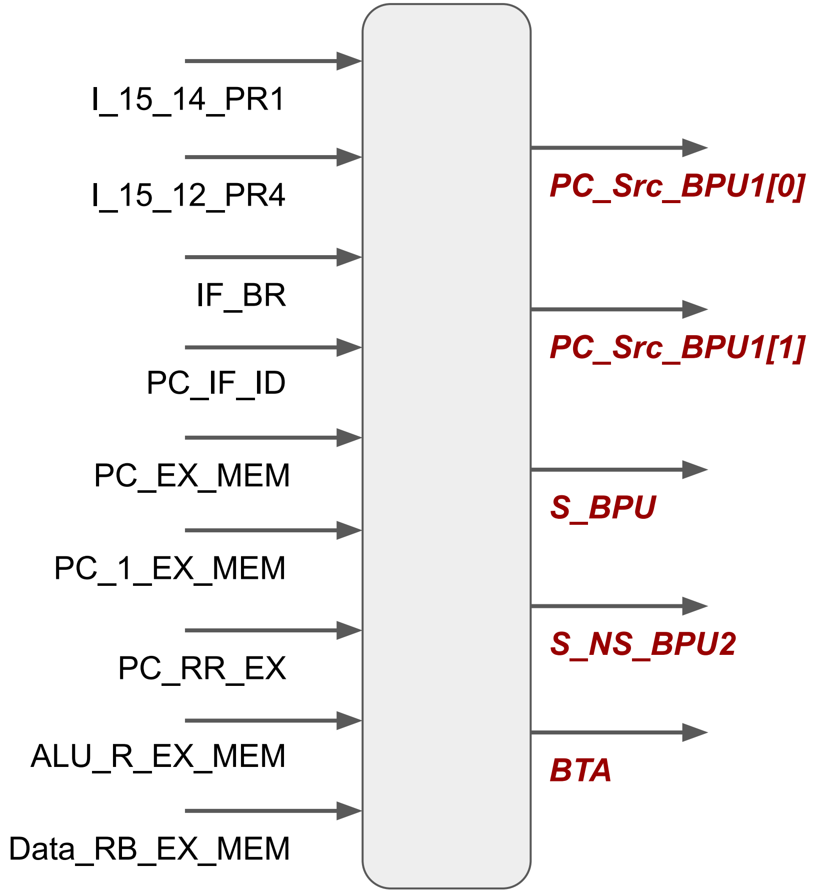
Hazard Detection Unit

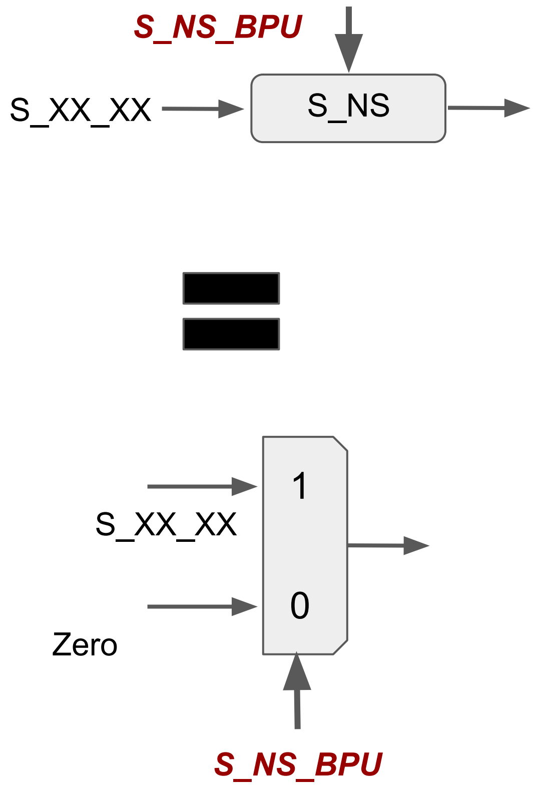


Branch Prediction Unit



Branch Prediction Unit





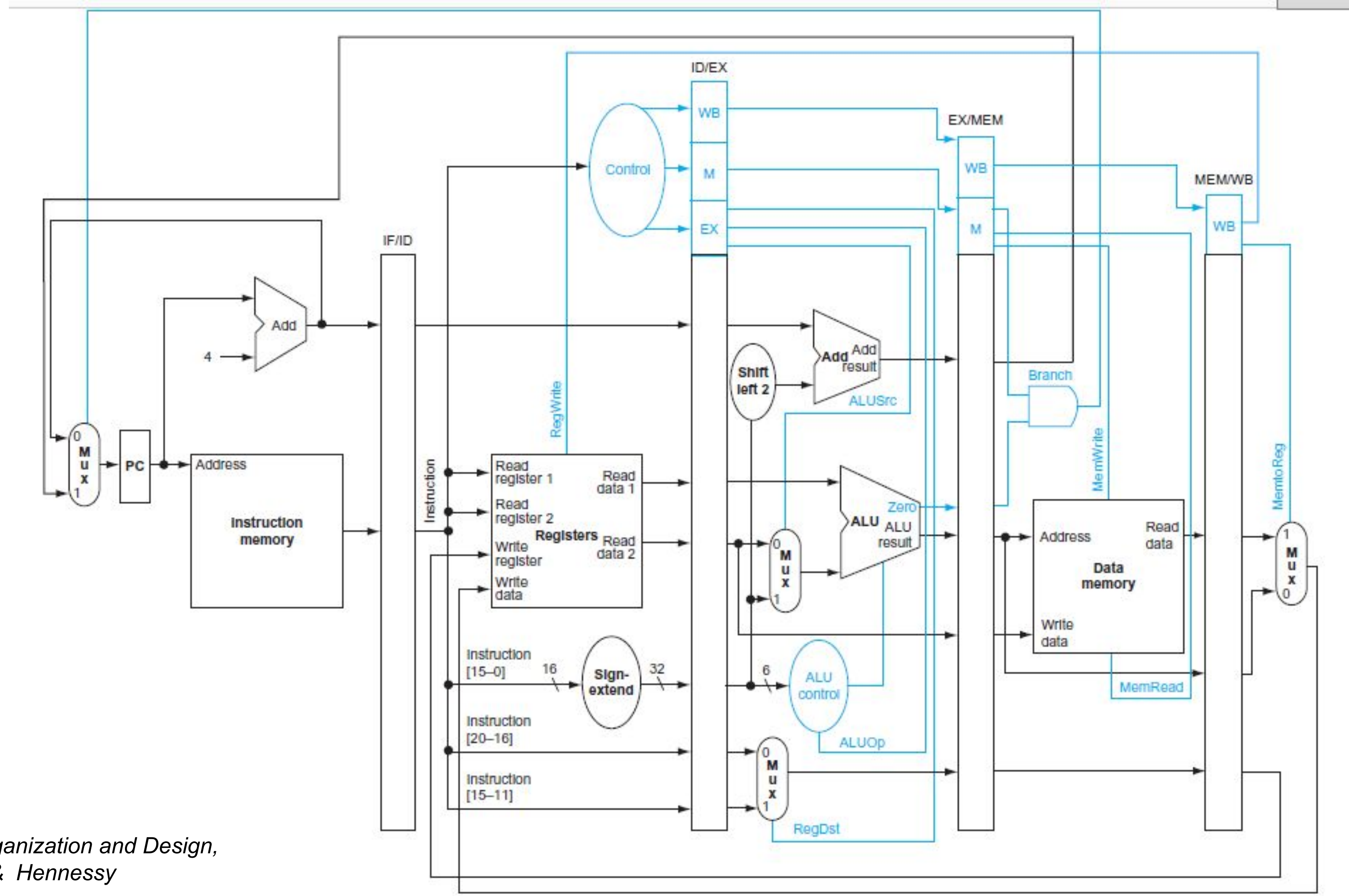
Instruction Decode | Controller

	ADD	ADC	ADZ	ADL	ADI	NDU	NDC	NDZ	LW	SW	BEQ	JAL	JLR	JRI
EX_ALU_C1	00	00	00	00	00	00	00	00	01	01	10	10	11	00
EX_ALU_C2	00	00	00	01	10	00	00	00	10	10	10	11	00	11
EX_ADD_NAND_C3	1	1	1	1	1	0	0	0	1	1	1	1	1	1
MEM_DM_WR_C4	0	0	0	0	0	0	0	0	0	1	0	0	0	0
WB_Sel_R_C5	00	00	00	00	01	00	00	00	10	00	00	10	10	00
WB_Sel_C6	00	00	00	00	00	00	00	00	10	00	00	01	01	00
WB_RF_W_C7	1	1	1	1	1	1	1	1	1	0	0	1	1	0

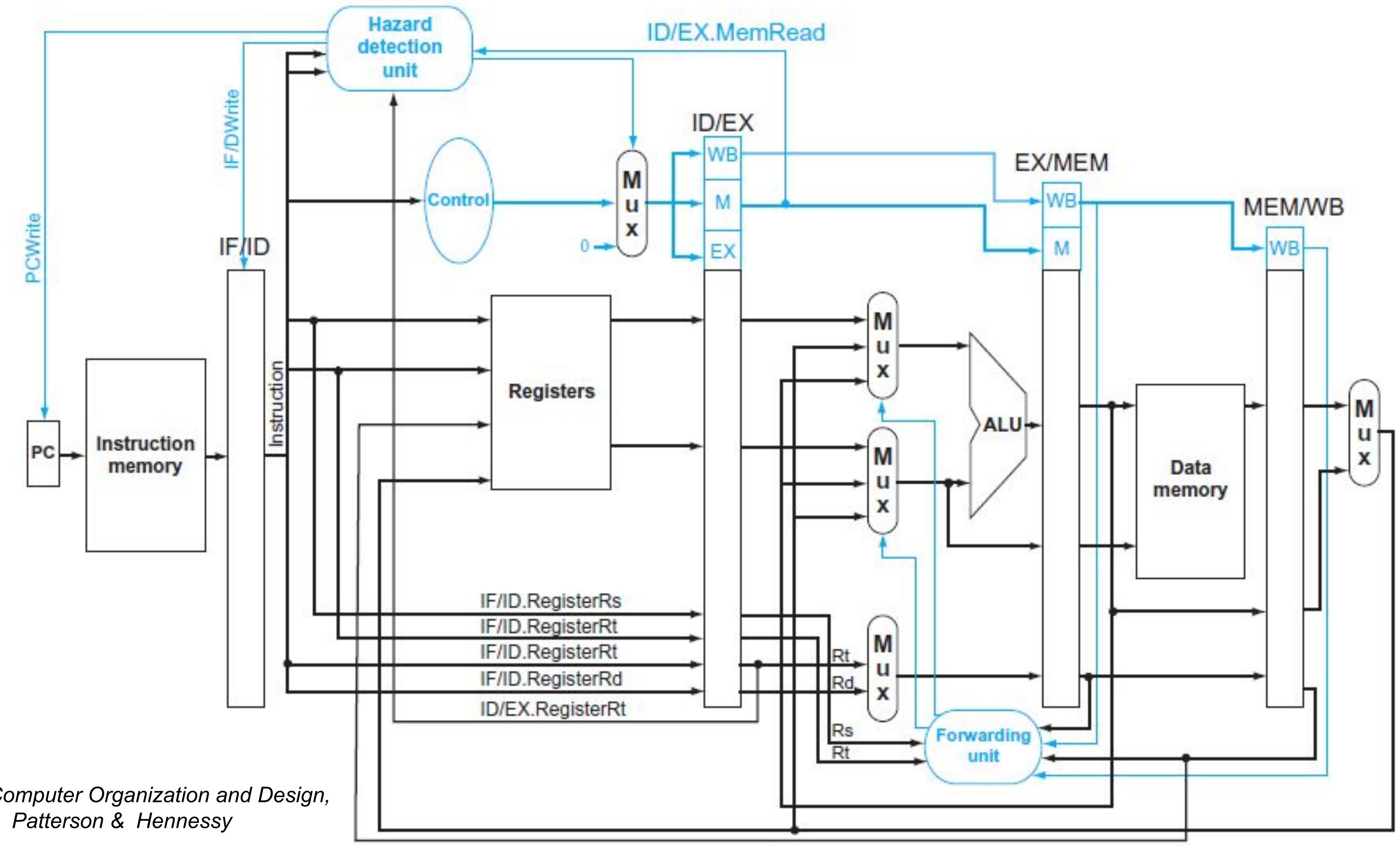
Condition for Bubble: Load is followed by an instruction whose source is the destination for the load instruction.

Condition for Data-Forwarding: Instruction I1 is followed by instruction I2 whose source is destination of I1.

- Use **SW**, **BEQ**, **JAL**, and **JLR** after 3 NOPE. (Data forwarding is not implemented for data stored from register file)



Source: Computer Organization and Design, Patterson & Hennessy



Source: Computer Organization and Design, Patterson & Hennessy