CSE 112: Computer Organization

Instructor: Sujay Deb

Lecture 12

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

AMS Lab @IIITD

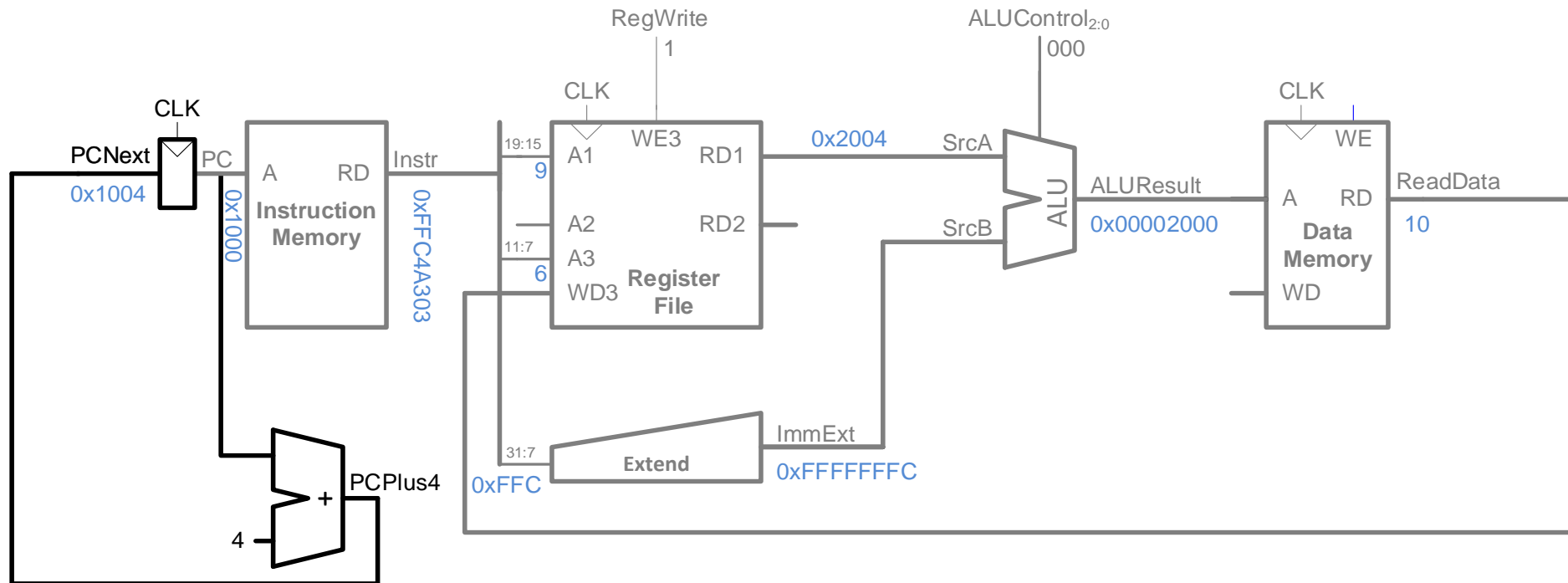# Single-Cycle
RISC-V Processor

Example Program

- Design datapath
- View example program executing

Example Program:

Address	Instruction	Type	Fields						Machine Language

			$imm_{11:0}$		**rs1**	**f3**	**rd**	**op**	
0x1000 L7:	lw x6, -4(x9)	I	111111111100		01001	010	00110	0000011	FFC4A303

			$imm_{11:5}$	**rs2**	**rs1**	**f3**	**$imm_{4:0}$**	**op**	
0x1004	sw x6, 8(x9)	S	0000000	00110	01001	010	01000	0100011	0064A423

			funct7	**rs2**	**rs1**	**f3**	**rd**	**op**	
0x1008	or x4, x5, x6	R	0000000	00110	00101	110	00100	0110011	0062E233

			$imm_{12,10:5}$	**rs2**	**rs1**	**f3**	**$imm_{4:1,11}$**	**op**	
0x100C	beq x4, x4, L7	B	1111111	00100	00100	000	10101	1100011	FE420AE3

Single-Cycle Datapath: PC Increment

STEP 6: Determine address of next instruction

Address	Instruction	Type	Fields					Machine Language
			$imm_{11:0}$	rs1	f3	rd	op	
0x1000	L7: lw x6, -4(x9)	I	111111111100	01001	010	00110	0000011	FFC4A303

Single-Cycle Datapath: Other Instructions

Single-Cycle Datapath: `sw`

- **Immediate:** now in {instr[31:25], instr[11:7]}
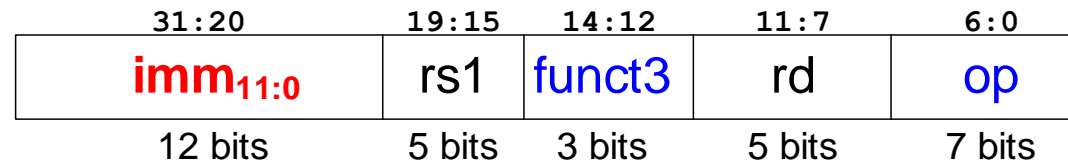- **Add control signals:** ImmSrc, MemWrite
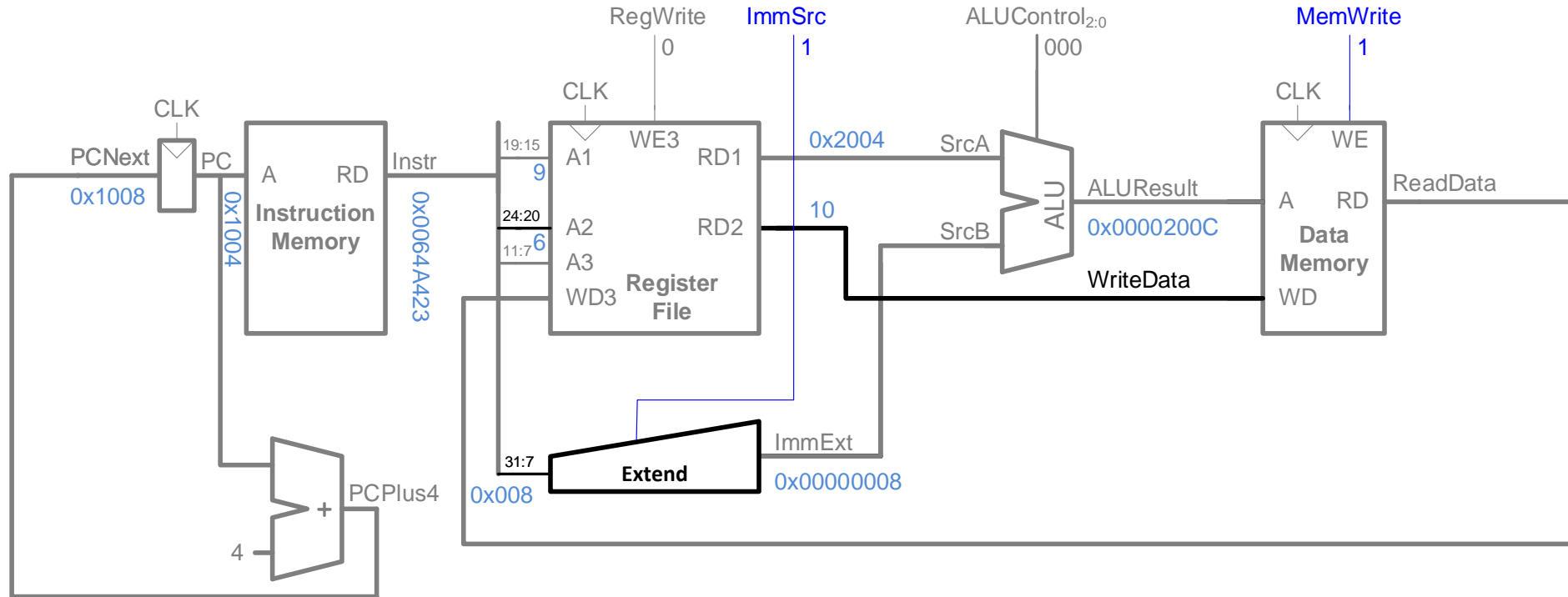
Address	Instruction	Type	Fields						Machine Language
			$imm_{11:5}$	rs2	rs1	f3	$imm_{4:0}$	op	
0x1004	sw x6, 8(x9)	S	0000000	00110	01001	010	01000	0100011	0064A423

Single-Cycle Datapath: Immediate

ImmSrc	ImmExt	Instruction Type
0	{{20{instr[31]}}, **instr[31:20]**}	I-Type
1	{{20{instr[31]}}, **instr[31:25], instr[11:7]**}	S-Type

I-Type

31:20	19:15	14:12	11:7	6:0
$imm_{11:0}$	rs1	funct3	rd	op
12 bits	5 bits	3 bits	5 bits	7 bits

S-Type

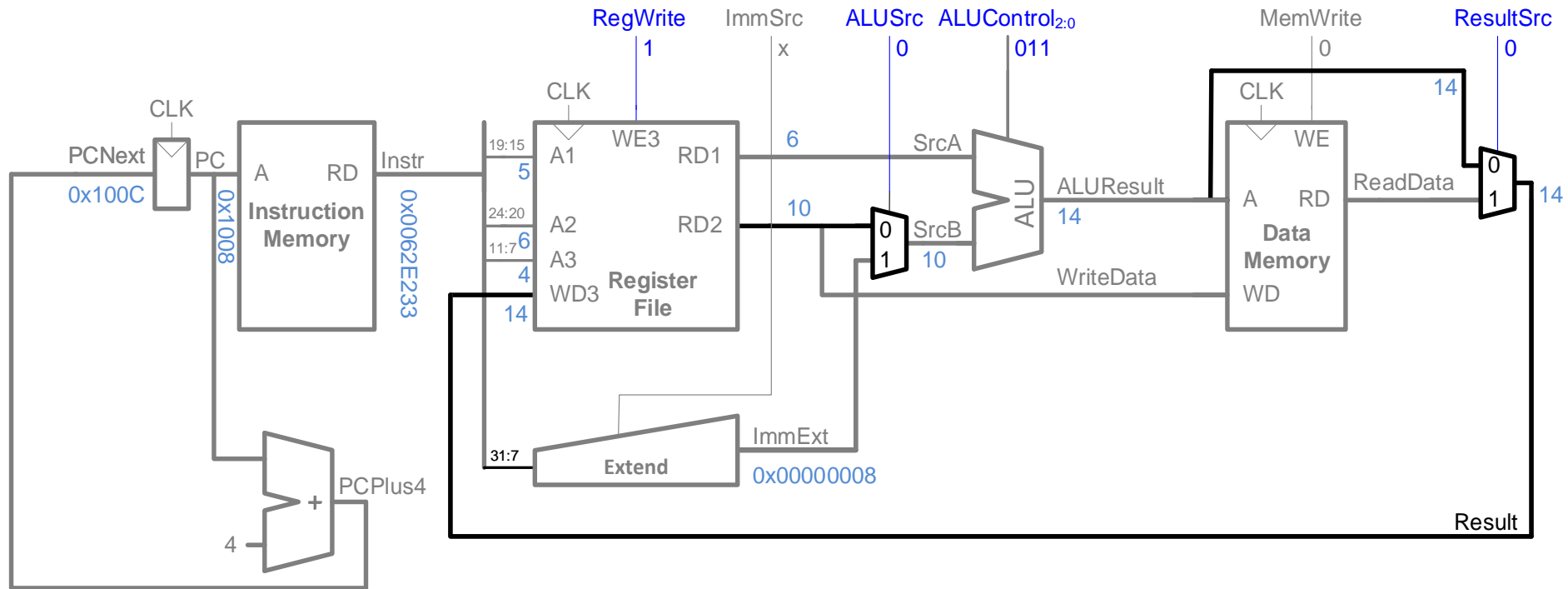| 31:25	24:20	19:15	14:12	11:7	6:0
$imm_{11:5}$	rs2	rs1	funct3	$imm_{4:0}$	op
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits
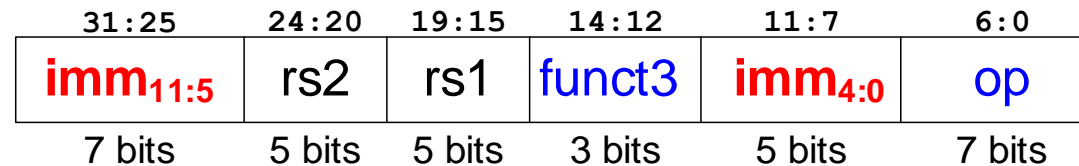
Single-Cycle Datapath: R-type

- Read from **rs1** and **rs2** (instead of **imm**)
- Write *ALUResult* to **rd**

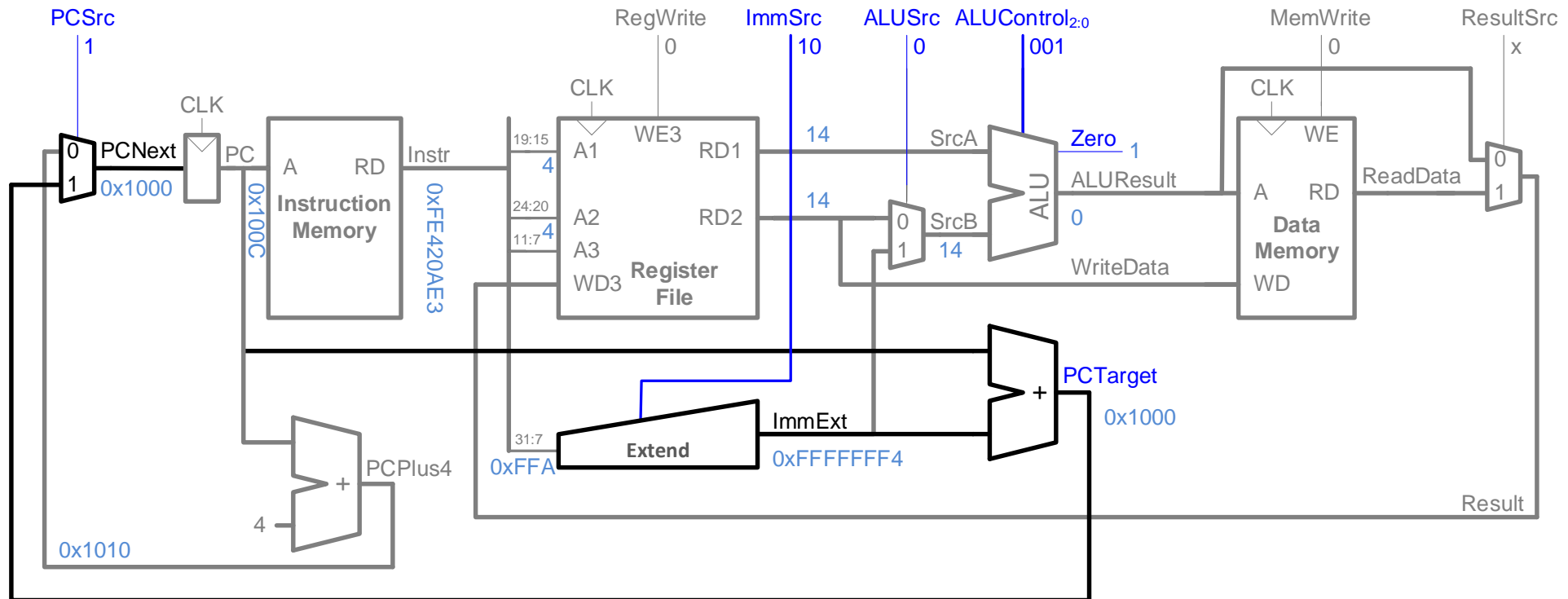| Address	Instruction	Type	funct7	rs2	rs1	f3	rd	op	Machine Language
						Fields			
0x1008	or x4, x5, x6	R	0000000	00110	00101	110	00100	0110011	0062E233

Single-Cycle Datapath: beq

Calculate target address: PCTarget = PC + imm

Address	Instruction	Type	Fields						Machine Language
			$imm_{12,10:5}$	rs2	rs1	f3	$imm_{4:1,11}$	op	
0x100C	beq x4, x4, L7	B	1111111	00100	00100	000	10101	1100011	FE420AE3

9

Single-Cycle Datapath: ImmExt

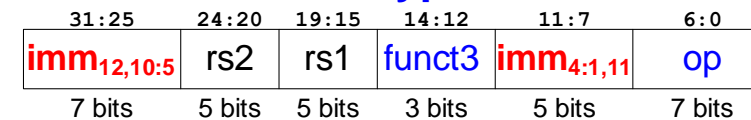| ImmSrc$_{1:0}$	ImmExt	Instruction Type
00	{{20{instr[31]}}, **instr[31:20]**}	I-Type
01	{{20{instr[31]}}, **instr[31:25], instr[11:7]**}	S-Type
10	{{19{instr[31]}}, **instr[31], instr[7], instr[30:25], instr[11:8], 1'b0**}	B-Type

I-Type

31:20	19:15	14:12	11:7	6:0
imm$_{11:0}$	rs1	funct3	rd	op
12 bits	5 bits	3 bits	5 bits	7 bits

S-Type

31:25	24:20	19:15	14:12	11:7	6:0
imm$_{11:5}$	rs2	rs1	funct3	**imm$_{4:0}$**	op
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

B-Type

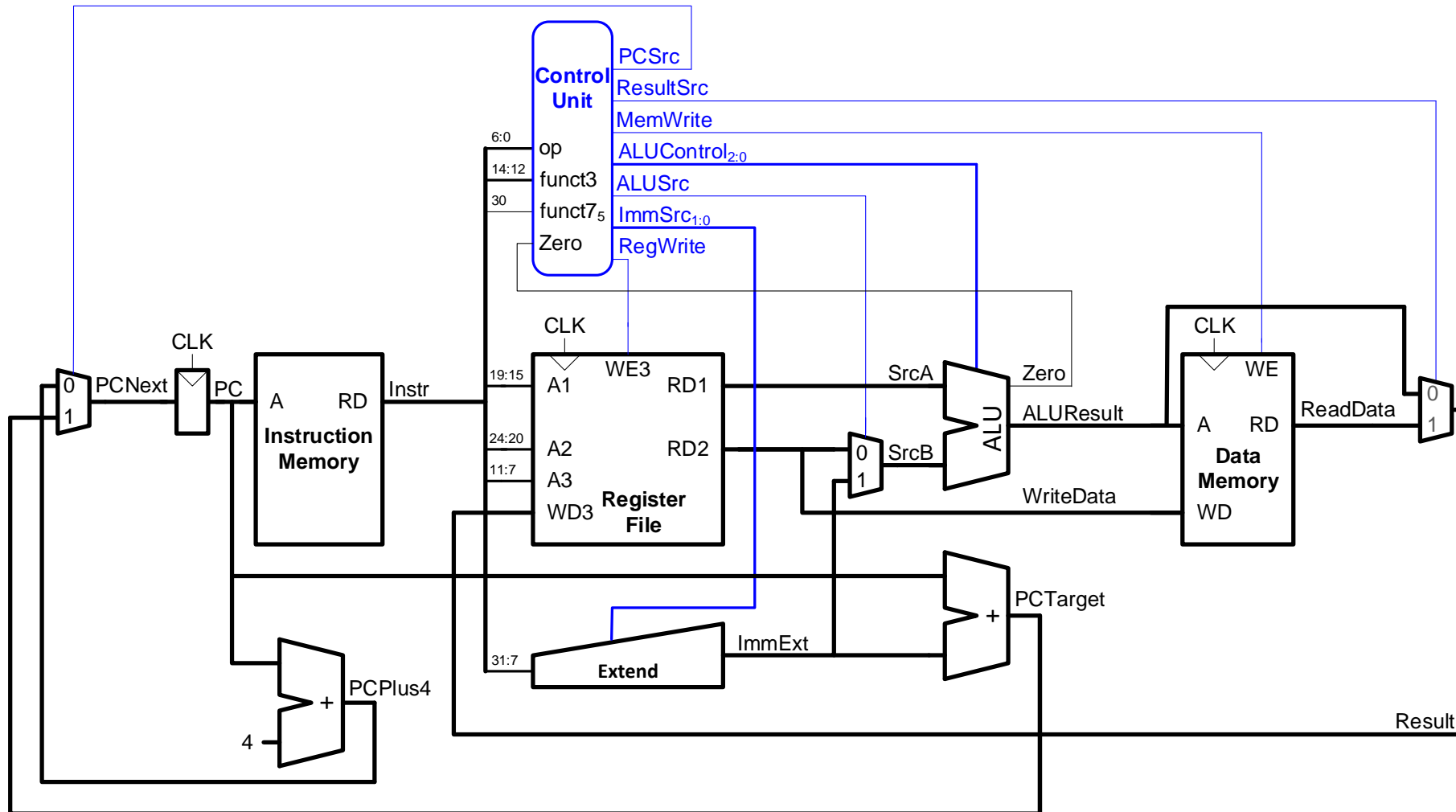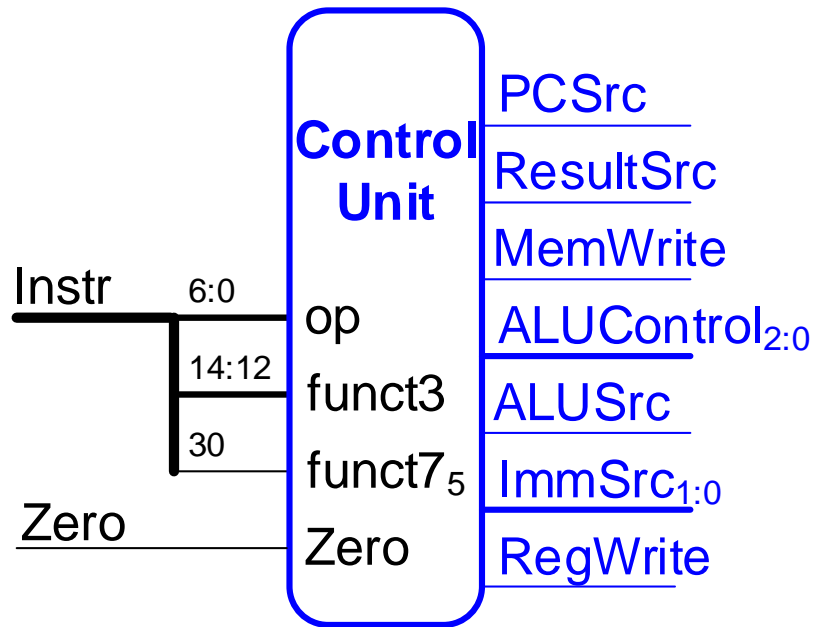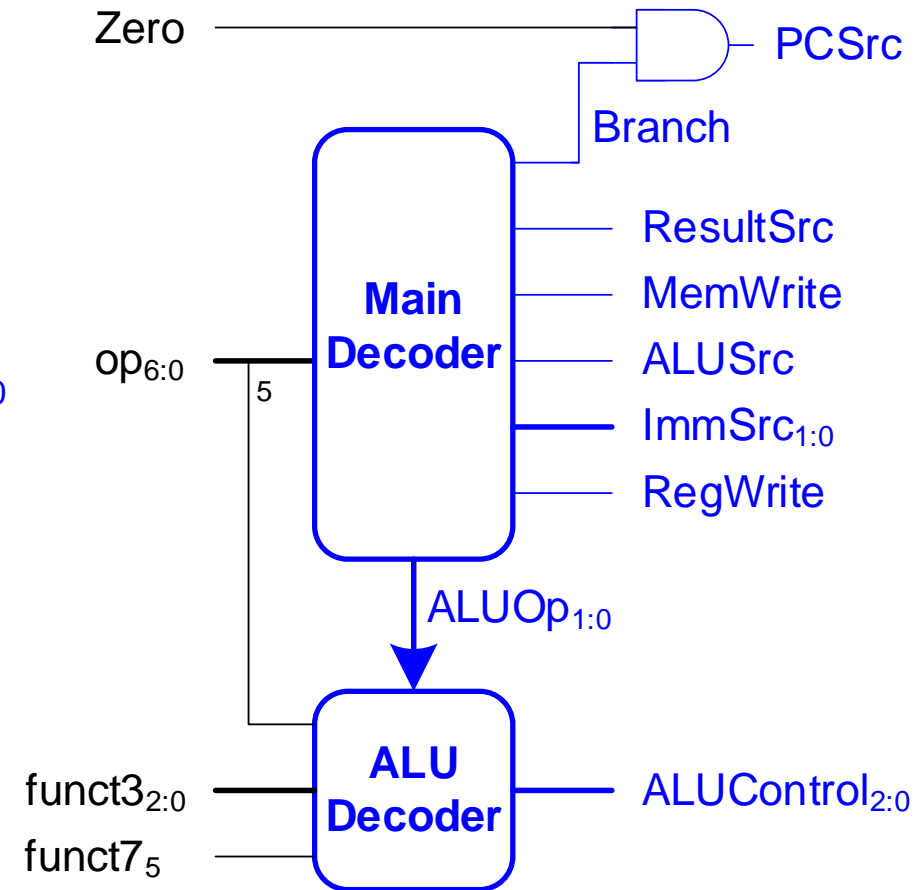| 31:25	24:20	19:15	14:12	11:7	6:0
imm$_{12,10:5}$	rs2	rs1	funct3	**imm$_{4:1,11}$**	op
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits
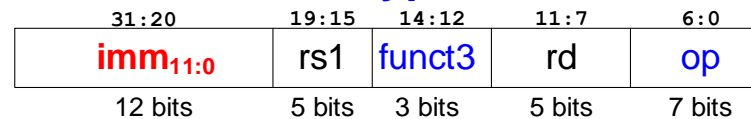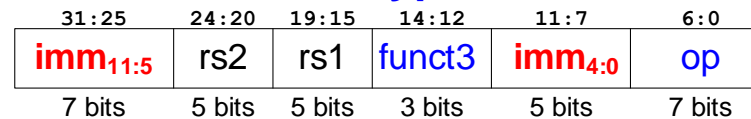
Single-Cycle RISC-V Processor

Single-Cycle Control

Single-Cycle Control

High-Level View

Low-Level View

Single-Cycle Control: Main Decoder
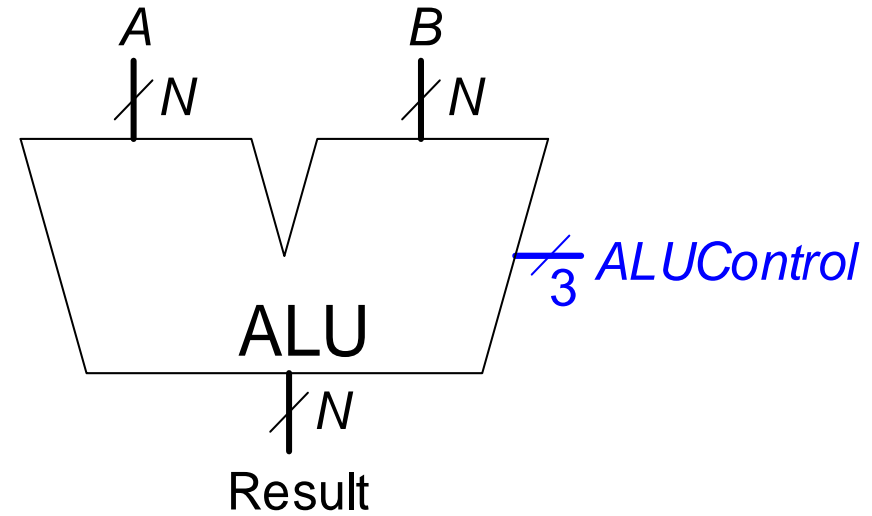
op	Instr.	RegWrite	ImmSrc	ALUSrc	MemWrite	ResultSrc	Branch	ALUOp
3	**lw**							
35	**sw**							
51	R-type							
99	**beq**							
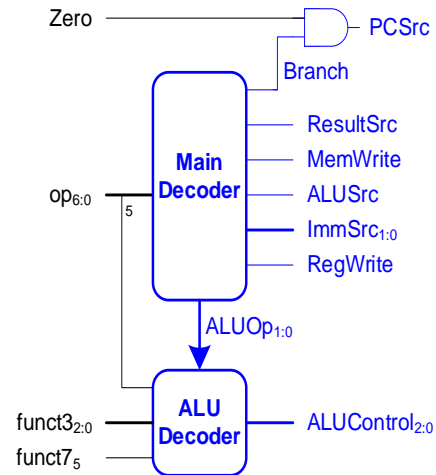
Review: ALU

ALUControl$_{2:0}$	Function
000	add
001	subtract
010	and
011	or
101	SLT

A N B N

N 3 *ALUControl*

ALU

N

Result

Review: ALU

ALUControl$_{2:0}$	Function
000	add
001	subtract
010	and
011	or
101	SLT

Single-Cycle Control: ALU Decoder
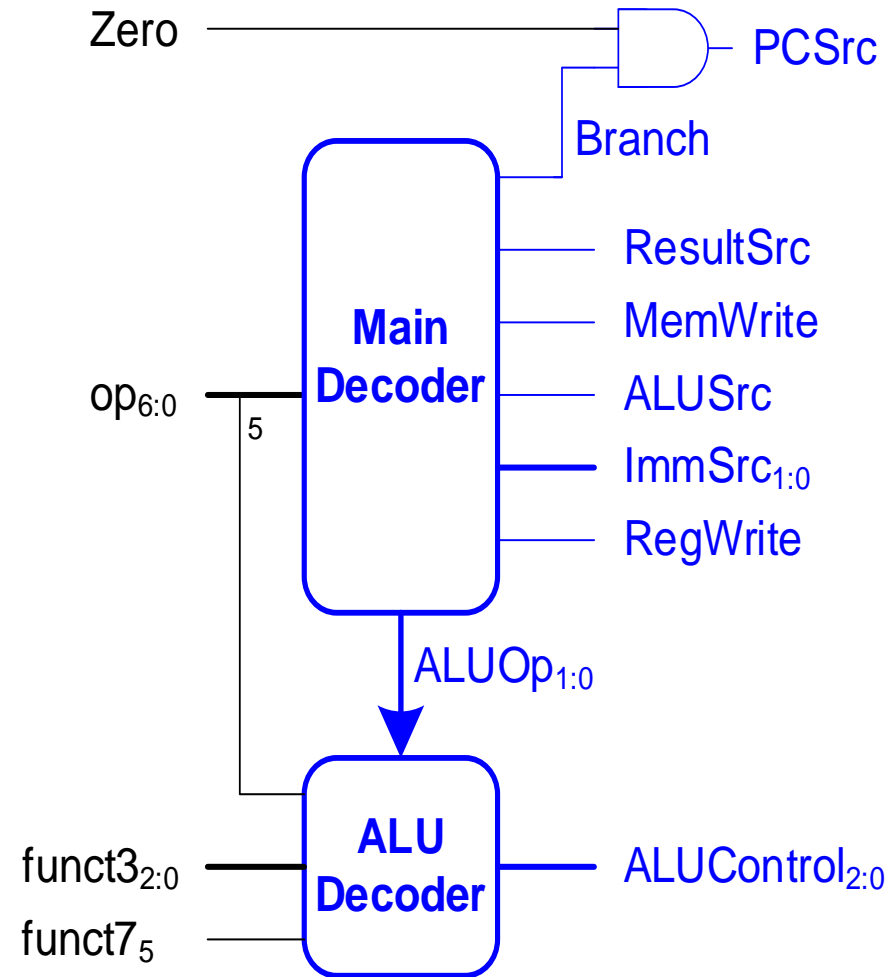
Single-Cycle Control: ALU Decoder

ALUOp	funct3	op_5, $funct7_5$	Instruction	$ALUControl_{2:0}$
00	x	x	**lw, sw**	000 (add)
01	x	x	**beq**	001 (subtract)
10	000	00, 01, 10	**add**	000 (add)
	000	11	**sub**	001 (subtract)
	010	x	**slt**	101 (set less than)
	110	x	**or**	011 (or)
	111	x	**and**	010 (and)

$ALUOp_{1:0}$

op_5
$funct3_{2:0}$
$funct7_5$

ALU Decoder

$ALUControl_{2:0}$
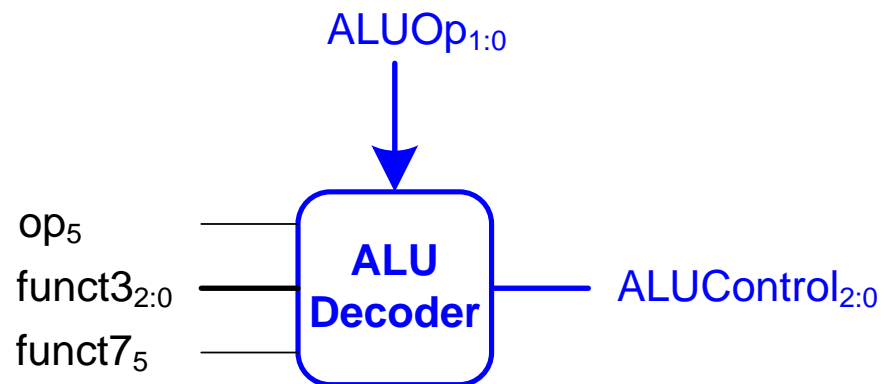
Example: `and`

op	Instruct	RegWrite	ImmSrc	ALUSrc	MemWrite	ResultSrc	Branch	ALUOp
51	R-type	1	XX	0	0	0	0	10
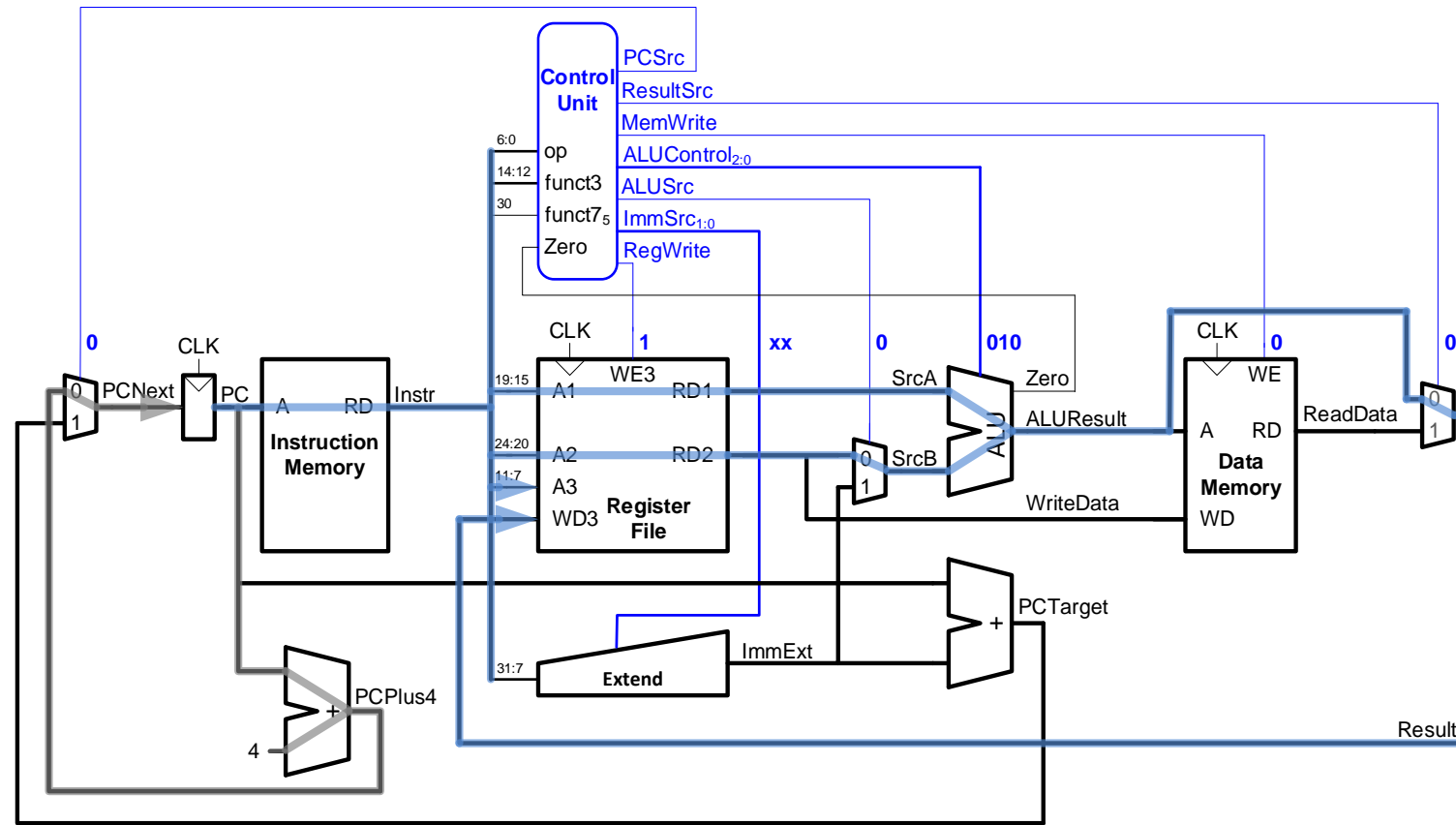
`and x5, x6, x7`

Extending the Single-Cycle Processor

Extended Functionality: I-Type ALU

Enhance the single-cycle processor to handle **I-Type ALU instructions**: `addi, andi, ori,` and `slti`

- **Similar to R-type** instructions

- But **second source** comes from **immediate**

- Change *ALUSrc* to select the immediate

- And *ImmSrc* to pick the correct immediate

Extended Functionality: I-Type ALU

op	Instruct.	RegWrite	ImmSrc	ALUSrc	MemWrite	ResultSrc	Branch	ALUOp
3	**lw**	1	00	1	0	1	0	00
35	**sw**	0	01	1	1	X	0	00
51	R-type	1	XX	0	0	0	0	10
99	**beq**	0	10	0	0	X	1	01
19	**I-type**	1	00	1	0	0	0	10

Extended Functionality: `addi`

op	Instruct.	RegWrite	ImmSrc	ALUSrc	MemWrite	ResultSrc	Branch	ALUOp
19	**I-type**	1	00	1	0	0	0	10
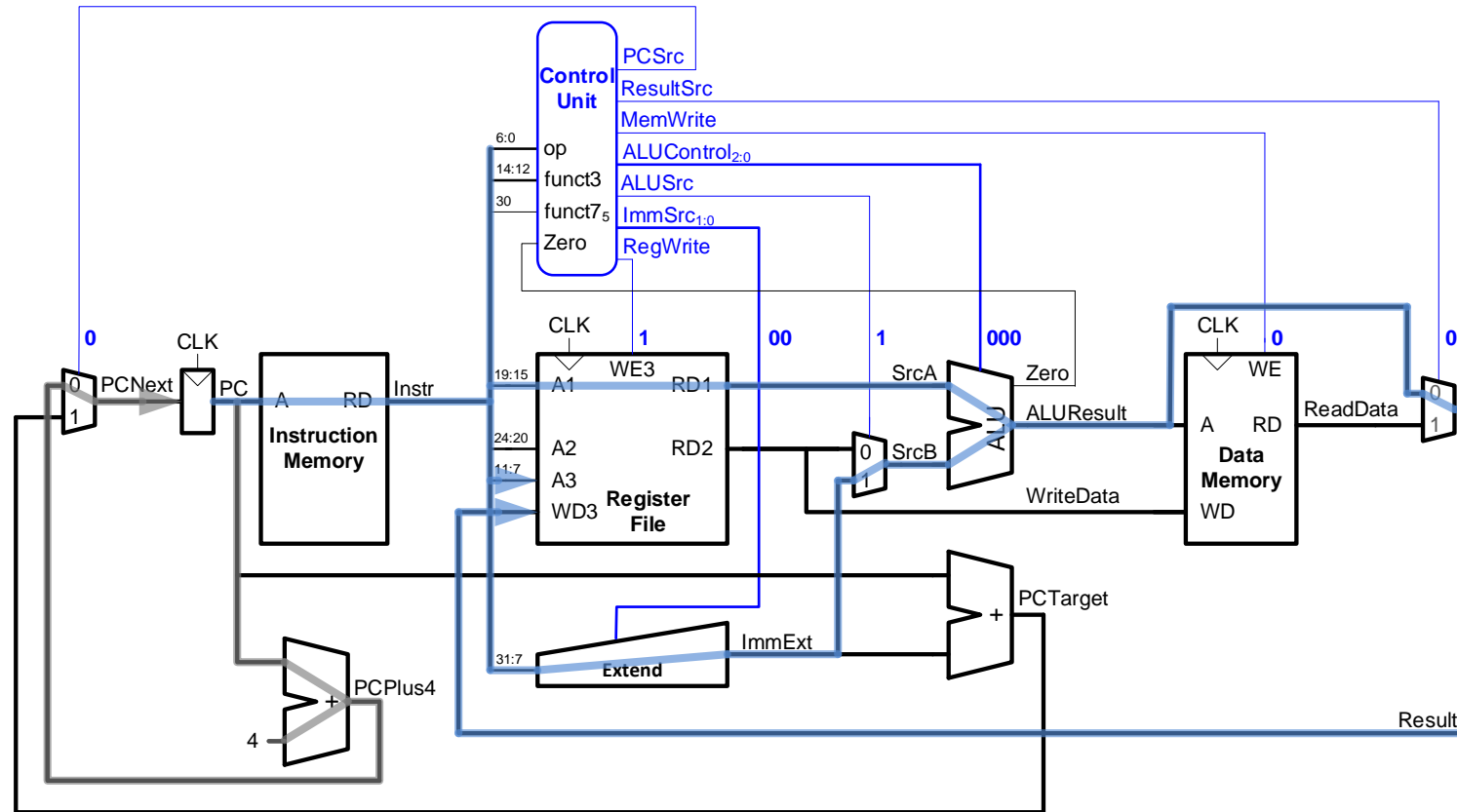
`addi x5, x6, -33`

Class Interaction # 14