RV32IM Pipeline Implementation

Hardware Units

GitHub Repository
Project Page

Damsy De Silva (E/16/069)
Shirly Ekanayaka (E/16/094)
Buddhi Perera (E/16/276)

Table of Contents

1. Pipeline Diagram

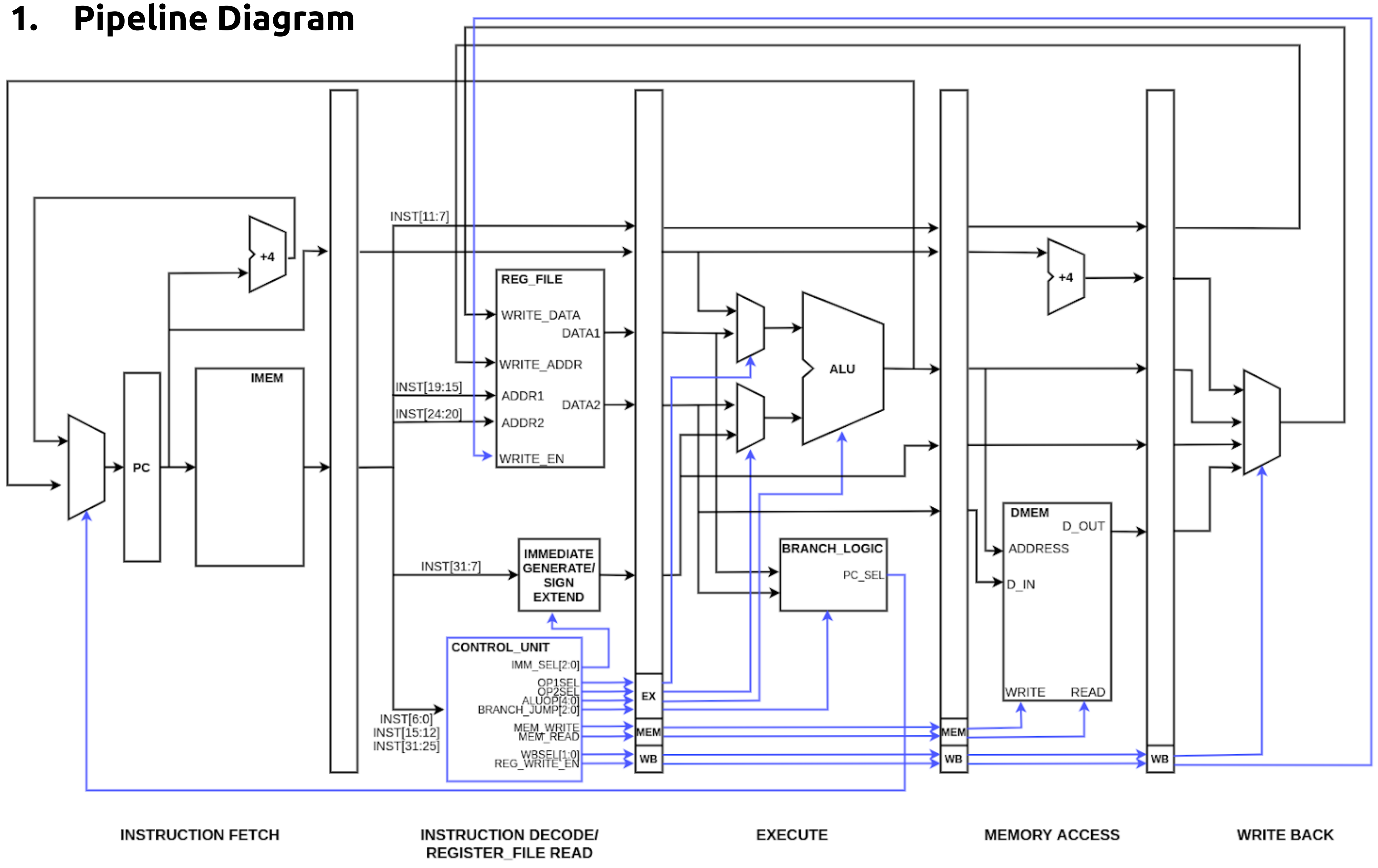

FIGURE 1 : Pipeline Diagram

2. Control Unit

The control unit generates the necessary control signals to select the proper data path for the instruction.

Inputs to the control unit are,

- OPCODE[6:0]
- FUNCT3[2:0]
- FUNCT7[6:0]

Outputs generated from the control unit are,

- IMM_SEL[2:0]
- OP1_SEL
- OP2_SEL
- ALU_OP[4:0]
- BRANCH_JUMP[2:0]
- MEM_WRITE
- MEM_READ
- WB_SEL[1:0]
- REG_WRITE_EN

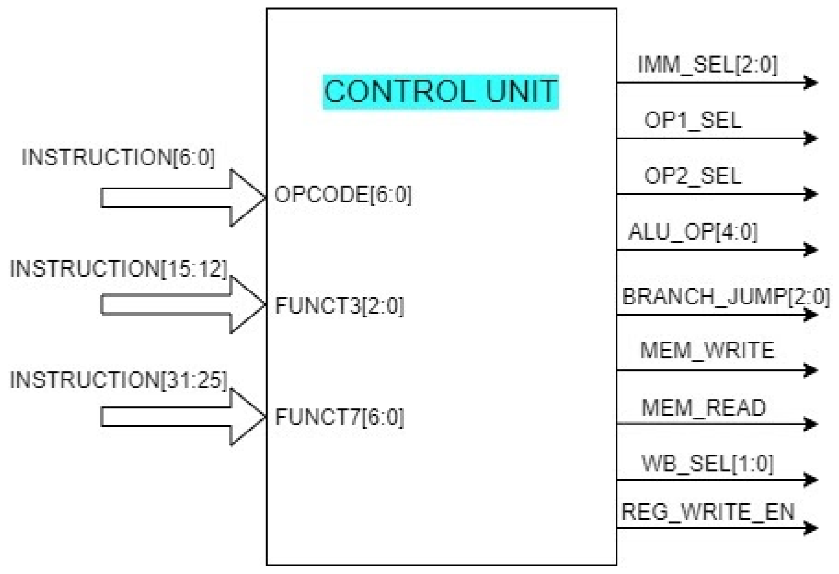

FIGURE 2 : Control Unit

2.1. Control Signals Generated by the Control Unit

2.1.1. IMM_SEL

This control signal is for the immediate value generation unit. In RISC-V ISA, according to the ordering of the immediate value bits there are 7 variants (See Immediate value generation unit for more details).

- U - Type
- J - Type
- S - Type
- B - Type
- I - Type signed
- I - Type containing shift amount
- I - Type unsigned

The immediate value generation unit will generate these 7 types of immediate values and the IMM_SEL control signal will select the relevant immediate value. The IMM_SEL control signal is a 3-bit signal and the encoding of the bits are as shown in Table 1.

Immediate Type	IMM_SEL[2]	IMM_SEL[1]	IMM_SEL[0]
U	0	0	0
J	0	0	1
S	0	1	0
B	0	1	1
I_signed	1	0	0
I_shift	1	0	1
I_unsigned	1	1	1

TABLE 1 : IMM_SEL Encodings

2.1.2. OP1_SEL

The input operand 1 of the ALU unit is of 2 values.

- PC value - For AUIPC, JAL, B - Type instructions
- DATA1 (value from the register file) - For all the other remaining instructions

This control signal will select between these two values. This is a 1 - bit control signal and the encoding is shown in Table 2.

Operand 1 Value	OP1_SEL
DATA1	0
PC	1

TABLE 2 : OP1_SEL Encoding

2.1.3. OP2_SEL

The input operand 2 of the ALU unit is of 2 values.

- DATA2 (value from the register file) - For R - Type instructions
- Immediate value - For all the other remaining instructions

This control signal will select between these two values. This is a 1 - bit control signal and the encoding is shown in Table 3.

Operand 2 Value	OP2_SEL
DATA2	0
Immediate Value	1

TABLE 3 : OP2_SEL Encoding

2.1.4. ALU_OP

This signal will select the relevant ALU operation out of the 18 ALU operations . This is a 5 - bit control signal and the encoding is shown in Table 4.

ALU operation	ALU_OP[4]	ALU_OP[3]	ALU_OP[2]	ALU_OP[1]	ALU_OP[0]
ADD	0	0	0	0	0
SUB	0	0	0	1	0
SLL	0	0	1	0	0
SLT	0	1	0	0	0
SLTU	0	1	1	0	0
XOR	1	0	0	0	0
SRL	1	0	1	0	0
SRA	1	0	1	1	0
OR	1	1	0	0	0
AND	1	1	1	0	0
MUL	0	0	0	0	1
MULH	0	0	1	0	1
MULHU	0	1	0	0	1
MULHSU	0	1	1	0	1
DIV	1	0	0	0	1
DIVU	1	0	1	0	1
REM	1	1	0	0	1
REMU	1	1	1	0	1

TABLE 4 : ALU_OP Encoding

2.1.5. BRANCH_JUMP

This control signal will select the type of branching to be considered by the Branching and Jump detection unit. Instructions in RV32IM can be categorized into 8 categories depending on their branching (See Branching and Jump Detection Unit for more details).

- BEQ - For BEQ instruction
- BNE - For BNE instruction
- J - For J - Type instruction
- BLT - For BLT instruction
- BGE - For BGE instruction
- BLTU - For BLTU instruction
- BGEU - For BGEU instruction
- NO - For all other remaining instructions

BRANCH_JUMP control signal is a 3 - bit control signal and encoding is shown in Table 5.

Branch Type	**BRANCH_JUMP [2]**	**BRANCH_JUMP [1]**	**BRANCH_JUMP [0]**
BEQ	0	0	0
BNE	0	0	1
NO	0	1	0
J	0	1	1
BLT	1	0	0
BGE	1	0	1
BLTU	1	1	0
BGEU	1	1	1

TABLE 5 : BRANCH_JUMP Encoding

2.1.6. MEM_WRITE

This control signal will enable writing to the data memory. When MEM_WRITE is set, data is written to the data memory and when MEM_WRITE is cleared, data is not written to the data memory.

2.1.7. MEM_READ

This control signal will enable reading from the data memory. When MEM_READ is set, data is read from the data memory and when MEM_READ is cleared, data is not read from the memory.

2.1.8. WB_SEL

There are 4 sources for the write back value to be written to the register file.

- ALU result - For AUIPC, I - Type and R - Type
- Data from the data memory - For Load instructions
- Immediate value - For LUI instruction
- PC + 4 value - For J - Type instruction

This control signal will select between these 4 sources. The WB_SEL signal is a 2 - bit control signal and the encoding is shown in Table 6.

Writeback Source	**WB_SEL[1]**	**WB_SEL[0]**
ALU result	0	0
Data from data memory	0	1
Immediate value	1	0
PC + 4	1	1

TABLE 6 : WB_SEL Encoding

2.1.9. REG_WRITE_EN

This control signal will enable writing to the register file. When REG_WRITE_EN is set, the write back value is written to the register file and when REG_WRITE_EN is cleared, the write back value is not written to the register file.

2.2. Control Unit Design and Design Decisions

2.2.1. Control Signals Generated from the OPCODE

Control unit was implemented using combinational logic. Some control signals were generated by using the OPCODE bits in the instruction and for some control signals an intermediate signal was generated using the OPCODE bits and then the intermediate signal, FUNCT3 bits and FUNCT7 bits of the instruction were used to generate the control signals.

Control signals generated using the OPCODE,

- OP1_SEL
- OP2_SEL
- MEM_WRITE
- MEM_READ
- REG_WRITE_EN
- WB_SEL[1:0]

Control signals generated using the OPCODE, FUNCT3 and FUNCT7,

- IMM_SEL[2:0]
- ALU_OP[4:0]
- BRANCH_JUMP[2:0]

Figure 3 shows the combinational logic circuit designed to generate the control signals and the intermediate signals using the OPCODE bits. ALUOP_TYPE, BL and IMM_TYPE are the intermediate signals generated using the OPCODE bits which will be later used by separate combinational logic circuits to generate the ALU_OP, BRANCH_JUMP and IMM_SEL control signals respectively.

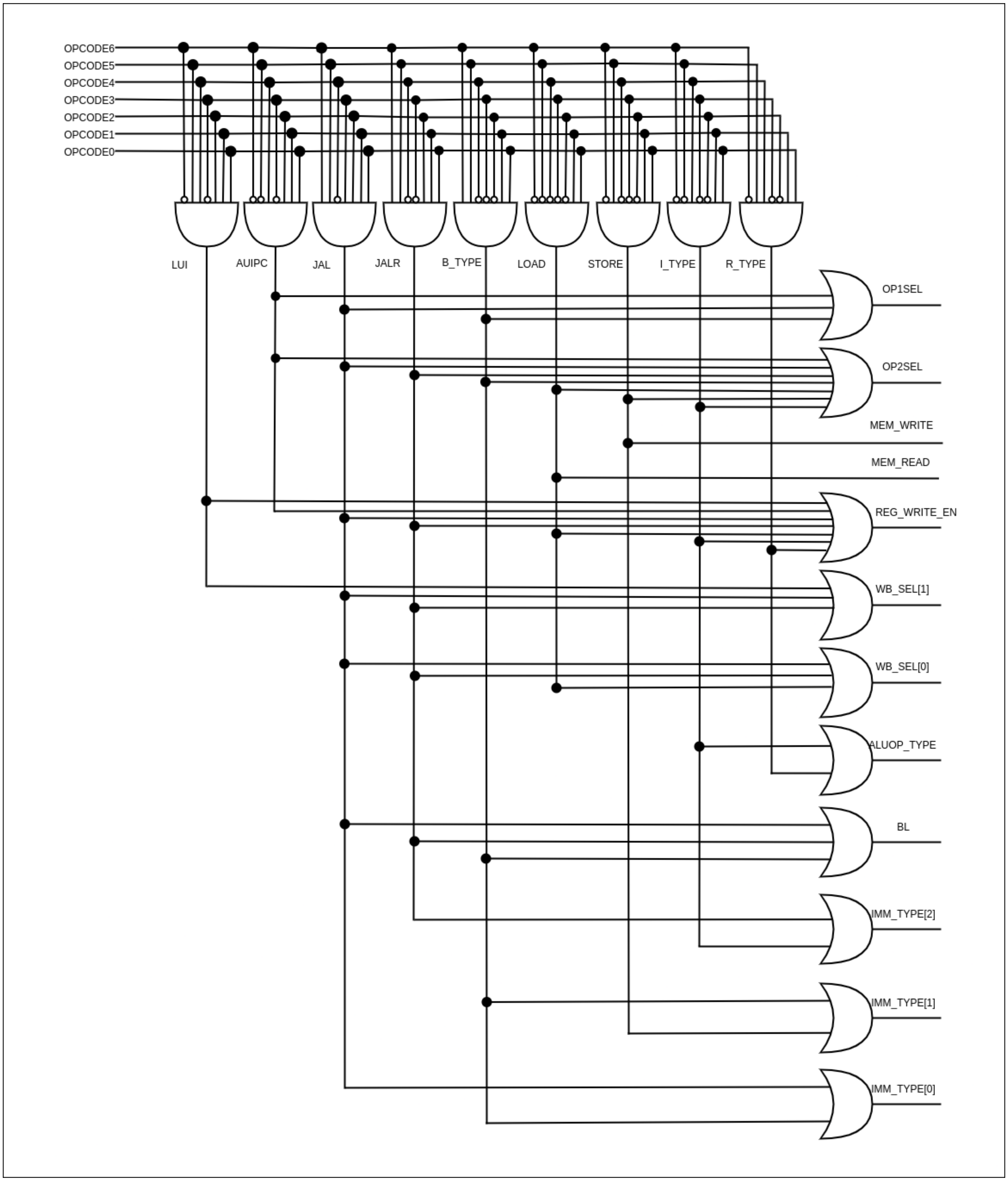

FIGURE 3 : Combinational circuit generating the control signals from OPCODE

2.2.2. Generating the IMM_SEL control signal

To generate the IMM_SEL control signal, the FUCNT3 bits of the instruction were used to obtain the proper immediate variant for the instructions given below.

- SLTIU - This is the only instruction that requires an unsigned immediate variant.
- SLLI, SRLI, SRAI - These instructions require the immediate variant that contains the shift amount.

FUNCT3 bits in SLTIU, SLLI, SRLI and SRAI instructions were used along with the IMM_TYPE intermediate signal to generate the IMM_SEL control signal.

Immediate Type	**INPUT**						**OUTPUT**		
	IMM_TYPE[2]	**IMM_TYPE[1]**	**IMM_TYPE[0]**	**FUNCT[2]**	**FUNCT[1]**	**FUNCT[0]**	**IMM_SEL[2]**	**IMM_SEL[1]**	**IMM_SEL[0]**
U	0	0	0	x	x	x	0	0	0
J	0	0	1	x	x	x	0	0	1
S	0	1	0	x	x	x	0	1	0
B	0	1	1	x	x	x	0	1	1
I_signed	1	0	0	x	x	x	1	0	0
I_shift	1	0	0	x	0	1	1	0	1
I_unigned	1	0	0	0	1	1	1	1	1

Table 7 : Truth table for generating the IMM_SEL control signal

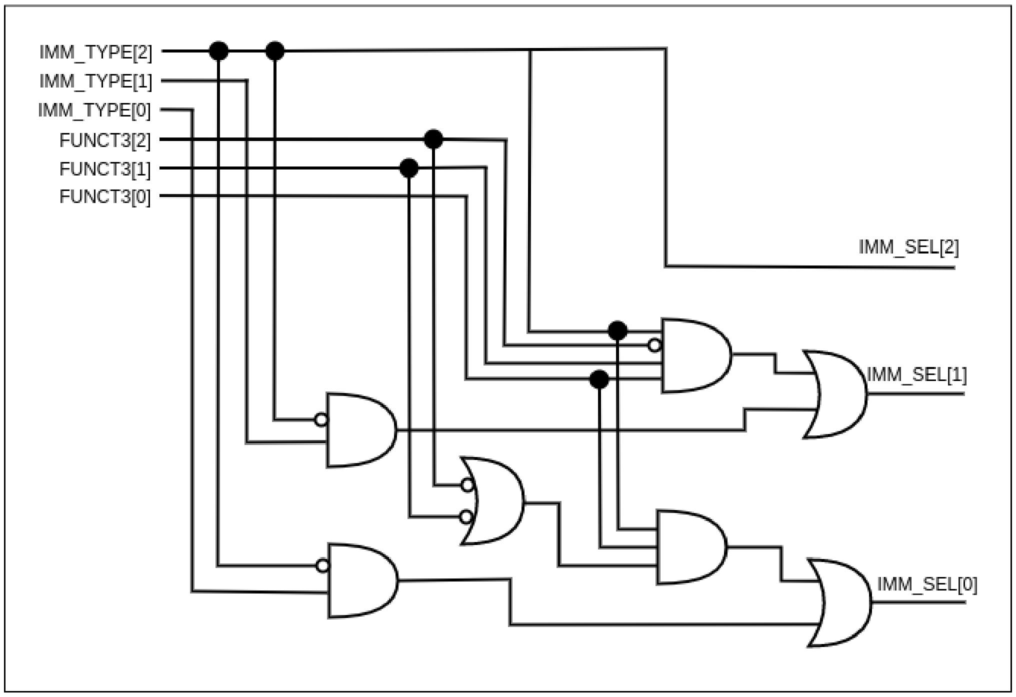

FIGURE 4 : Combinational circuit generating the IMM_SEL control signal

2.2.3. Generating the ALU_OP control signal

ALUOP_TYPE intermediate signal, FUNCT3, FUNCT7, IMM_SEL control signal, and R_TYPE(generated from the OPCODE) were used to generate the ALU_OP control signal. 4 variants can be considered when generating the control signal.

- R - Type, SLLI, SRLI and SRAI instructions

ALU_OP for these instructions were generated by concatenating the 3 bits in the FUNCT3, 5th bit in FUNCT7 and 0th bit in the FUNCT7.

- I - Type instructions without SLLI, SRLI and SRAI instructions

ALU_OP for these instructions were generated by concatenating the 3 bits in the FUNCT3, 2 zero signals.

- All other instructions that uses ALU

ALU_OP for these instructions was an ADD operation.

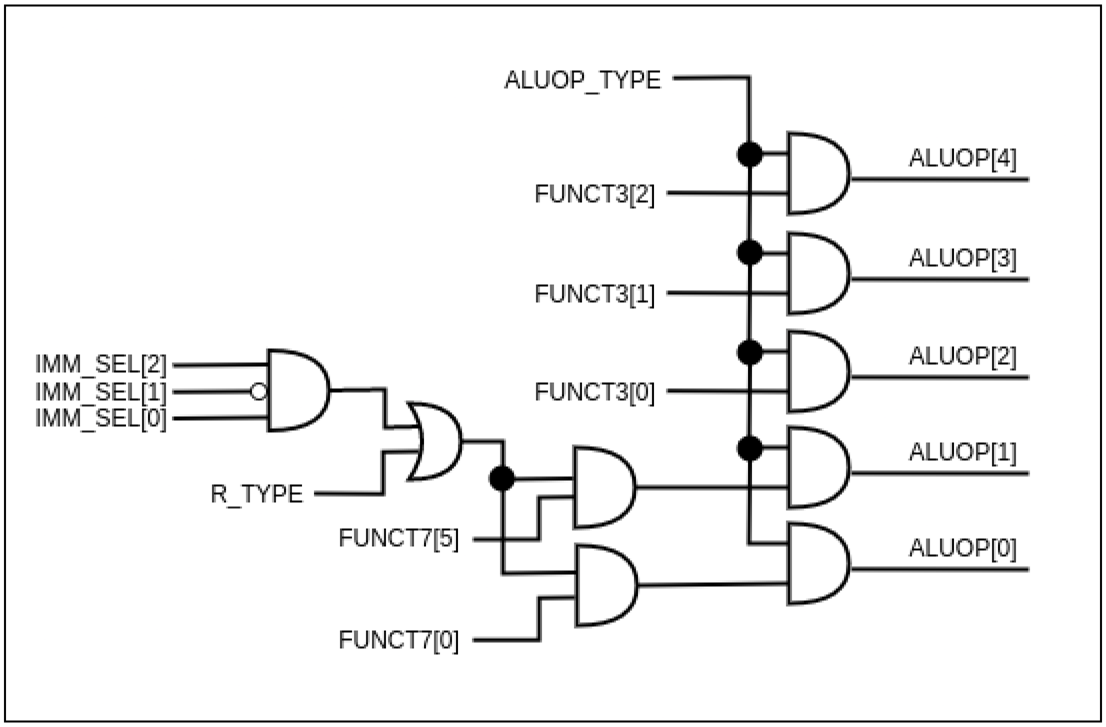

FIGURE 5 : Combinational circuit generating the ALU_OP control signal

2.2.4. Generating the BRANCH_JUMP control signal

BL intermediate signal, OPCODE[2] and FUNCT3 bits were used to generate the BRANCH_JUMP control signal. In B - Type instructions, the FUNCT3 bits define the type of branching. Therefore FUNCT3 bits were used to identify the branch type. OPCODE[2] bit was used to distinguish between the B - Type and the J - Type instructions.

Branch Type	**INPUT**					**OUTPUT**		
	OPCODE[2]	**BL**	**FUCNT3[2]**	**FUCNT3[1]**	**FUCNT3[0]**	**BRANCH_JUMP[2]**	**BRANCH_JUMP[1]**	**BRANCH_JUMP[0]**
BEQ	0	1	0	0	0	0	0	0
BNE	0	1	0	0	1	0	0	1
NO	x	0	x	x	x	0	1	0
J	1	1	x	x	x	0	1	1
BLT	0	1	1	0	0	1	0	0
BGE	0	1	1	0	1	1	0	1
BLTU	0	1	1	1	0	1	1	0
BGEU	0	1	1	1	1	1	1	1

Table 8 : Truth table for generating the BRANCH_JUMP control signal

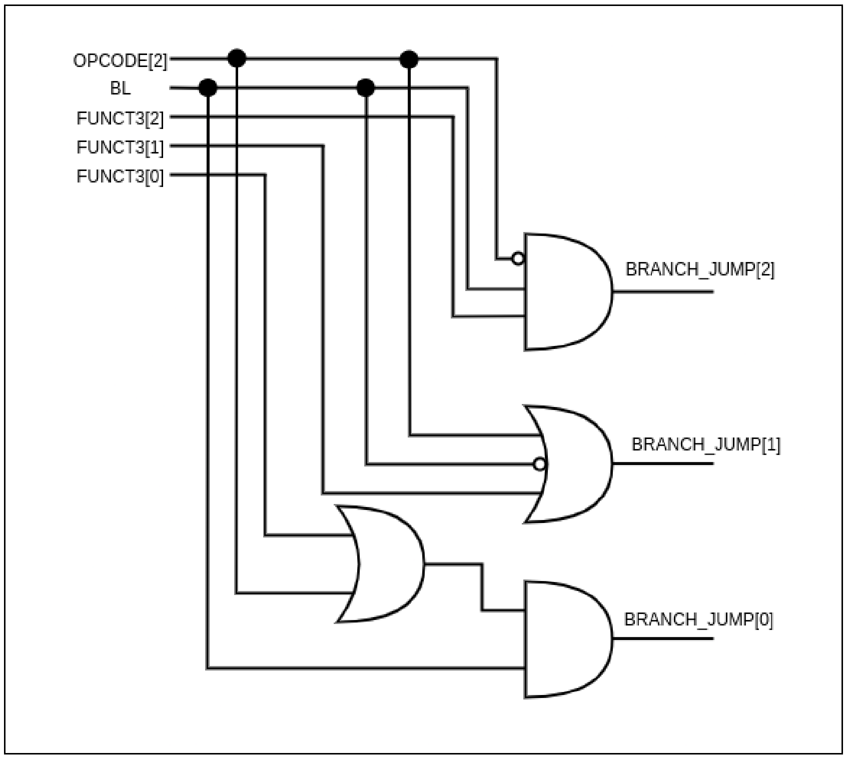

FIGURE 6 : Combinational circuit generating the BRANCH_JUMP control signal

2.3. Instructions and the Control Signals

INSTRUCTION	PC_SEL	IMM_SEL	OP1SEL	OP2SEL	ALUOP	MEM_WRITE	MEM_READ	REG_WRITE_EN	WB_SEL	BRANCH_JUMP
LUI	PC + 4	U	*	*	*	0	0	1	IMM	000
AUIPC	PC + 4	U	PC	IMM	ADD	0	0	1	ALU	000
JAL	ALU	J	PC	IMM	ADD	0	0	1	PC + 4	111
JALR	ALU	I	DATA1	IMM	ADD	0	0	1	PC + 4	111
BEQ	PC + 4 / ALU	B	PC	IMM	ADD	0	0	0	*	001
BNE	PC + 4 / ALU	B	PC	IMM	ADD	0	0	0	*	010
BLT	PC + 4 / ALU	B	PC	IMM	ADD	0	0	0	*	011
BGE	PC + 4 / ALU	B	PC	IMM	ADD	0	0	0	*	100
BLTU	PC + 4 / ALU	B	PC	IMM	ADD	0	0	0	*	101
BGEU	PC + 4 / ALU	B	PC	IMM	ADD	0	0	0	*	110
LB	PC + 4	I_signed	DATA1	IMM	ADD	0	1	1	MEM	000
LH	PC + 4	I_signed	DATA1	IMM	ADD	0	1	1	MEM	000
LW	PC + 4	I_signed	DATA1	IMM	ADD	0	1	1	MEM	000
LBU	PC + 4	I_signed	DATA1	IMM	ADD	0	1	1	MEM	000
LHU	PC + 4	I_signed	DATA1	IMM	ADD	0	1	1	MEM	000
SB	PC + 4	S	DATA1	IMM	ADD	1	0	0	*	000
SH	PC + 4	S	DATA1	IMM	ADD	1	0	0	*	000
SW	PC + 4	S	DATA1	IMM	ADD	1	0	0	*	000
ADDI	PC + 4	I_signed	DATA1	IMM	ADD	0	0	1	ALU	000
SLTI	PC + 4	I_signed	DATA1	IMM	SLT	0	0	1	ALU	000
SLTIU	PC + 4	I_unsigned	DATA1	IMM	SLTU	0	0	1	ALU	000
XORI	PC + 4	I_signed	DATA1	IMM	XOR	0	0	1	ALU	000
ORI	PC + 4	I_signed	DATA1	IMM	OR	0	0	1	ALU	000

ANDI	PC + 4	I_signed	DATA1	IMM	AND	0	0	1	ALU	000
SLLI	PC + 4	I_shift	DATA1	IMM	SLL	0	0	1	ALU	000
SRLI	PC + 4	I_shift	DATA1	IMM	SRL	0	0	1	ALU	000
SRAI	PC + 4	I_shift	DATA1	IMM	SRA	0	0	1	ALU	000
ADD	PC + 4	*	DATA1	DATA2	ADD	0	0	1	ALU	000
SUB	PC + 4	*	DATA1	DATA2	SUB	0	0	1	ALU	000
SLL	PC + 4	*	DATA1	DATA2	SLL	0	0	1	ALU	000
SLT	PC + 4	*	DATA1	DATA2	SLT	0	0	1	ALU	000
SLTU	PC + 4	*	DATA1	DATA2	SLTU	0	0	1	ALU	000
XOR	PC + 4	*	DATA1	DATA2	XOR	0	0	1	ALU	000
SRL	PC + 4	*	DATA1	DATA2	SRL	0	0	1	ALU	000
SRA	PC + 4	*	DATA1	DATA2	SRA	0	0	1	ALU	000
OR	PC + 4	*	DATA1	DATA2	OR	0	0	1	ALU	000
AND	PC + 4	*	DATA1	DATA2	AND	0	0	1	ALU	000
MUL	PC + 4	*	DATA1	DATA2	MUL	0	0	1	ALU	000
MULH	PC + 4	*	DATA1	DATA2	MULH	0	0	1	ALU	000
MULHSU	PC + 4	*	DATA1	DATA2	MULHSU	0	0	1	ALU	000
MULHU	PC + 4	*	DATA1	DATA2	MULHU	0	0	1	ALU	000
DIV	PC + 4	*	DATA1	DATA2	DIV	0	0	1	ALU	000
DIVU	PC + 4	*	DATA1	DATA2	DIVU	0	0	1	ALU	000
REM	PC + 4	*	DATA1	DATA2	REM	0	0	1	ALU	000
REMU	PC + 4	*	DATA1	DATA2	REMU	0	0	1	ALU	000

Table 8 : Instructions and Control Signals

3. ALU

ALU is the main hardware unit that performs arithmetic and logic operations. This ALU supports 18 operations.

- ADD
- SUB
- SLL
- SLT
- SLTU
- XOR
- SRL
- SRA
- OR
- AND
- MUL
- MULH
- MULHU
- MULHSU
- DIV
- DIVU
- REM
- REMU

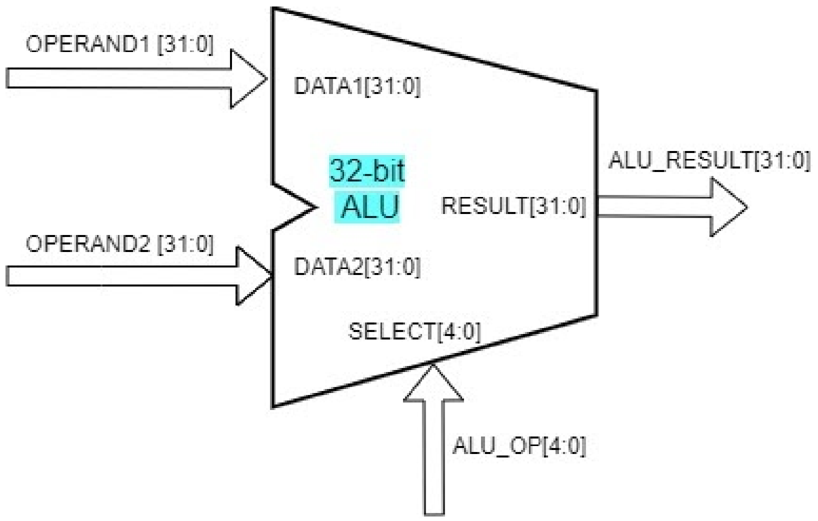

FIGURE 7 : ALU

Inputs to the ALU are,

- DATA1[31:0]
- DATA2[31:0]
- ALU_OP[4:0] - control signal
  (See ALU_OP for encoding and Generating the ALU_OP control signal for design)

Output of the ALU,

- ALU_RESULT[31:0]

4. Register File

Register file contains 32 registers and the size of a register is 32 bits. Register x0 is set to zero by making all the bits in register x0 to 0. Registers x1 to x31 can be used by the instructions.

In RISC-V ISA NOP instruction is encoded as an ADDI instruction. In this case the register x0 containing the value zero will be used.

NOP ⇒ ADDI x0, x0, 0

Inputs to the register file,

- WRITE_DATA[31:0]
- WRITE_ADDRESS[4:0]
- DATA1_ADDRESS[4:0]
- DATA2_ADDRESS[4:0]
- REG_WRITE_EN control signal
- RESET
- CLK

Outputs of the register file,

- DATA1[31:0]
- DATA2[31:0]

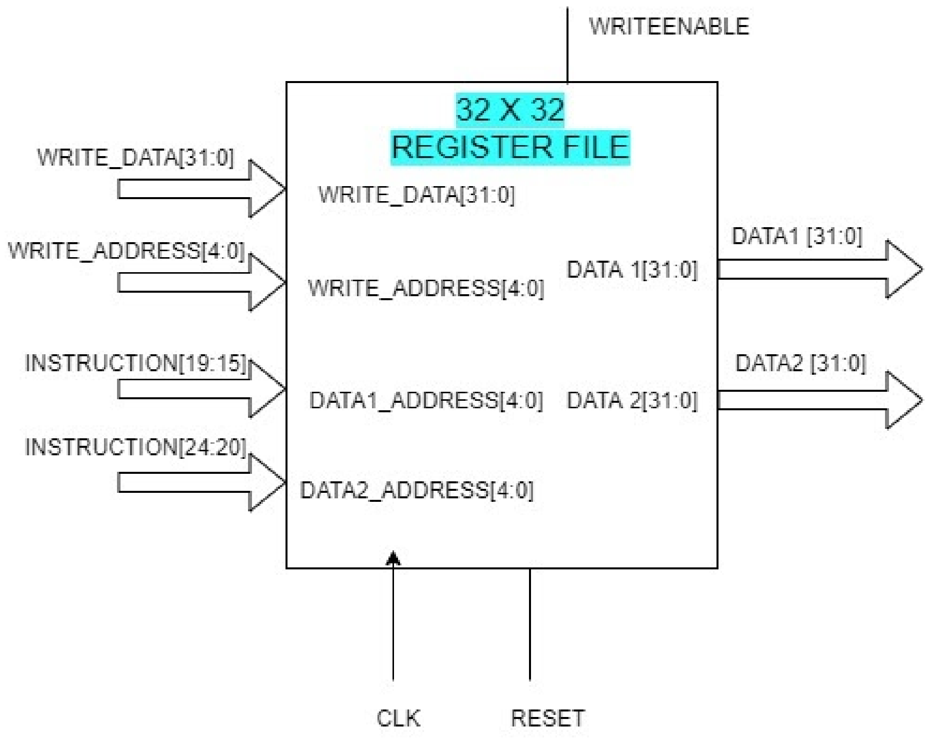

FIGURE 8 : Register File

5. Branching and Jump Detection Unit

This unit is for detecting whether the branch or the jump has to be taken or not. Inputs to this unit are,

- DATA1[31:0]
- DATA2[31:0]
- BRANCH_JUMP control signal (See BRANCH_JUMP for encoding)

Output of this unit is,

- PC_SELECT

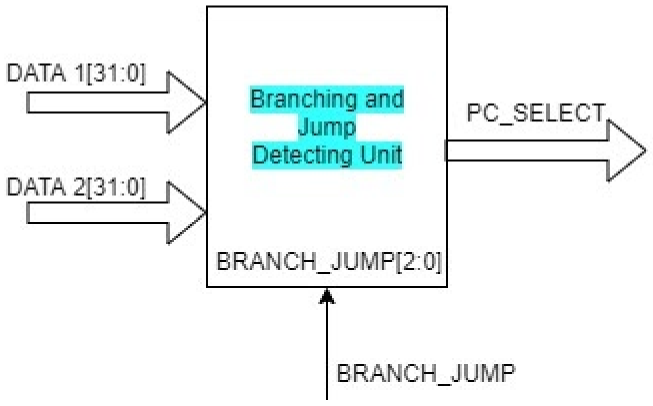

FIGURE 9 : Branch Jump Detection Unit

This unit will contain a comparator implemented using behavioural modeling and a combinational logic circuit to generate the PC_SELECT control signal. Branch and jump detection is done in 2 steps.

1. DATA1 and DATA2 values will be the inputs for the comparator. The comparator will output two 1 bit signals by comparing the input values,
   - EQUAL - If DATA1 and DATA2 are equal this signal will be set. Else it will be cleared.
   - LESS_THAN - If DATA1 is less than DATA2 this signal will be set. Else it will be cleared.

2. BRANCH_JUMP control signal, EQUAL and LESS_THAN intermediate signals will be the inputs to the combinational logic circuit. The combinational logic circuit will generate the PC_SELECT control signal depending on its inputs.

Branch Type	INPUT					OUTPUT
	BRANCH_JUMP[2]	BRANCH_JUMP[1]	BRANCH_JUMP[0]	Equal	Less Than	PC_SELECT
BEQ	0	0	0	0	x	0
				1	x	1
BNE	0	0	1	0	x	1
				1	x	0
NO	0	1	0	x	x	0
J	0	1	1	x	x	1
BLT	1	0	0	x	0	0
				x	1	1
BGE	1	0	1	0	0	1
				0	1	0
				1	x	1
BLTU	1	1	0	x	0	0
				x	1	1
BGEU	1	1	1	0	0	1
				0	1	0
				1	x	1

Table 9 : Truth Table of the combinational logic circuit

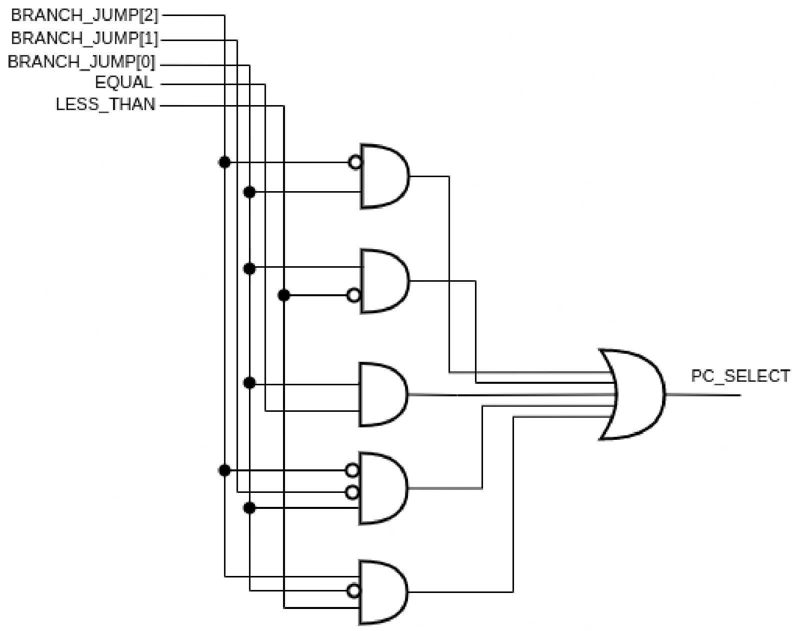

FIGURE 10 : Combinational Circuit

5.1. PC_SELECT Control Signal

PC_SELECT is a 1 bit control signal generated from the branch and jump detection unit. This signal will select the address source for the PC register. The PC register has 2 address sources.

- Address computed by the ALU
- Address computed by adding 4 to the current PC

Encoding of the PC_SELECT signal is shown in Table 10.

Address Source	**PC_SELECT**
PC + 4	0
Computer address from ALU	1

Table 10 : PC_SELECT Encoding

6. Immediate Value Generation Unit

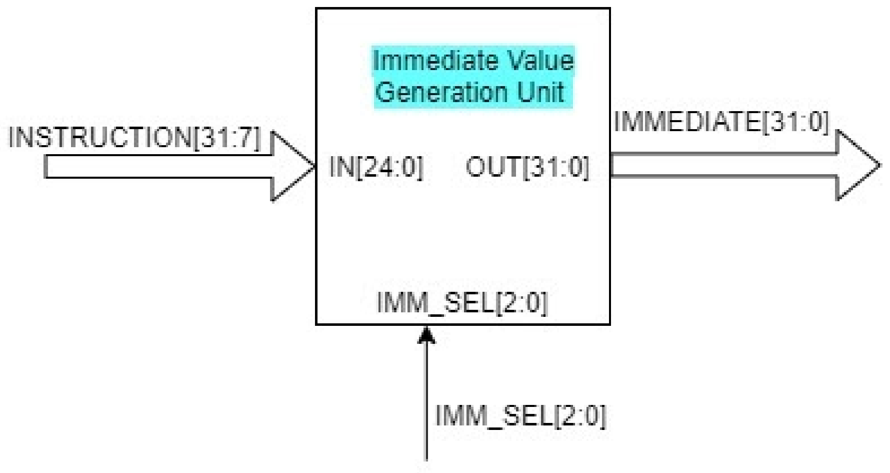

FIGURE 11 : Immediate Value Generation Unit

Instructions with immediate values can be categorized into 4 groups and I-Type instructions can be further categorized into 3 sub groups.

- U type Immediate

imm[31:12]	rd	opcode

- J type Immediate

imm[20]	imm[10:1]	imm[11]	imm[19:12]	rd	opcode

- S type Immediate

imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode

- B type Immediate

imm[12]	imm[10:5]	rs2	rs1	funct3	imm[4:1]	imm[11]	opcode

- I type Immediate with signed Extension (Extend with sign bit)

imm[11:0]	rs1	funct3	rd	opcode

- I type immediate containing shift amount

imm[11:5]	imm[4:0]	rs1	funct3	rd	opcode

- I type immediate with unsigned Extension (Extend with Zero)

imm[11:0]	rs1	funct3	rd	opcode

Inputs to the immediate value generation unit are,

- INSTRUCTION[31:7]
- IMM_SEL control signal[2:0]

Output of the immediate value generation unit is,

- IMMEDIATE_VALUE[31:0]

Immediate value generation unit will generate the immediate values and sign extends by rerouting the wires. Depending on the IMM_SEL control signal, this unit will output the proper immediate value. Immediate value generation and sign extending of the 7 categories are shown in the figure below.

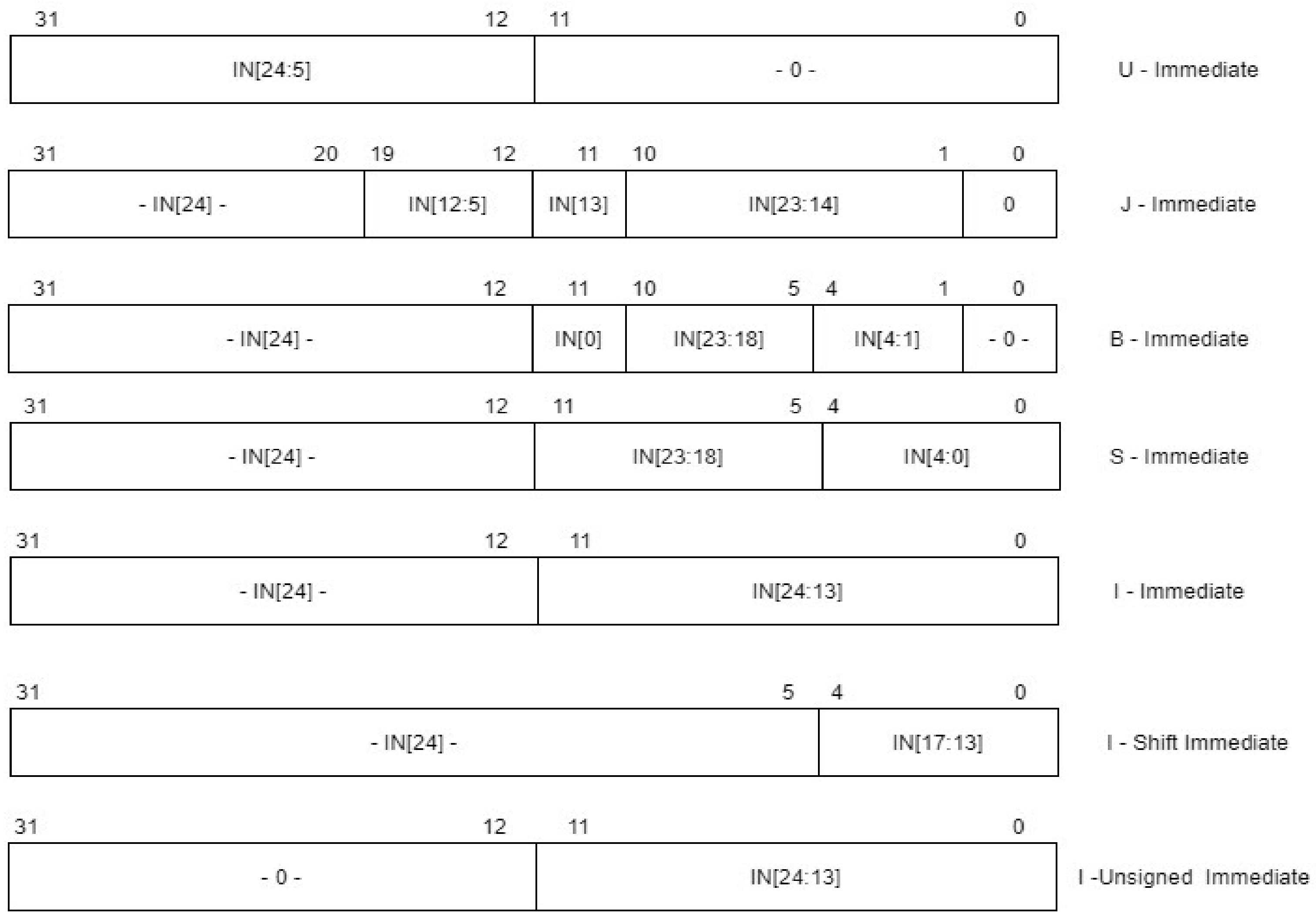

FIGURE 12 : Rerouting and Sign Extension of Immediate Values

7. Program counter register

Program counter register stores the address of the instruction. The value stored in this register is used when fetching the instruction from the instruction memory. Writing to the PC register is synchronous to the positive clock edge and the reading from the PC register is asynchronous. When the reset signal is set, the value in the PC register will be set to -4 and the program will restart from the next clock cycle.

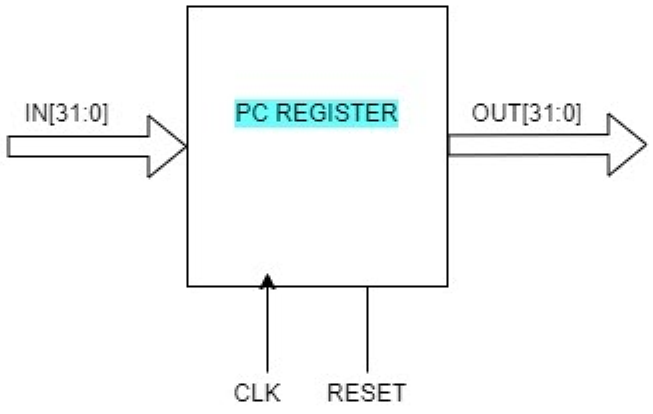

FIGURE 13 : Program Counter Register

8. Multiplexers

Three 2x1 and one 4x1 multiplexers are used in the design. Depending on the select signal, the multiplexer will output the corresponding value.

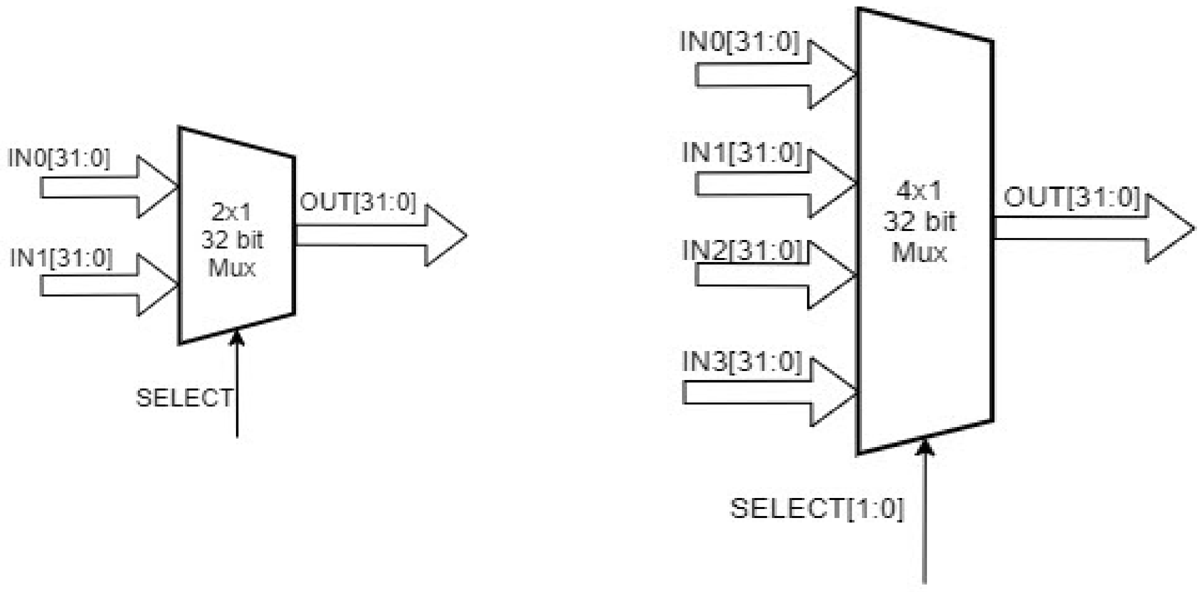

FIGURE 14 : Multiplexers

Table 11 shows the details about the multiplexers.

	Type of MUX	Inputs from	Select signal	Output to
Selecting the address source for PC register	2x1 MUX	PC+4, Address computer by the ALU	PC_SELECT control signal	PC register
Selecting the operand 1 for the ALU	2x1 MUX	DATA1 from register file, PC value	OP1_SEL	DATA1 input of the ALU
Selecting the operand 2 for the ALU	2x1 MUX	DATA2 from register file, Immediate value	OP2_SEL	DATA2 input of the ALU
Selecting the writeback source	4x1 MUX	ALU result, PC + 4 value, Immediate value, Read data from data memory	WB_SEL	Write data input of the register file

Table 11 : Multiplexer Details

9. Program counter incrementing adder

This hardware unit will add 4 to the input value. This unit is used for incrementing the PC and the PC + 4 value will be the output of this device.

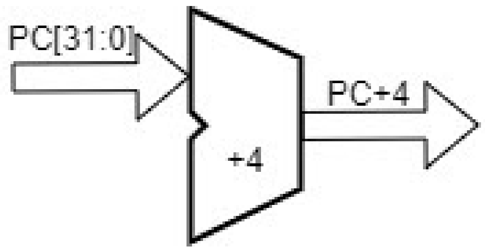

FIGURE 15 : PC incrementing adder