



life.augmented

Multicycle MIPS Microarchitecture

Names:

1. Aya Ashraf Mohamed
2. Mohamed Hossam Taha
3. Mohamed Khaled Alahmady
4. Yara Gooda Ahmed

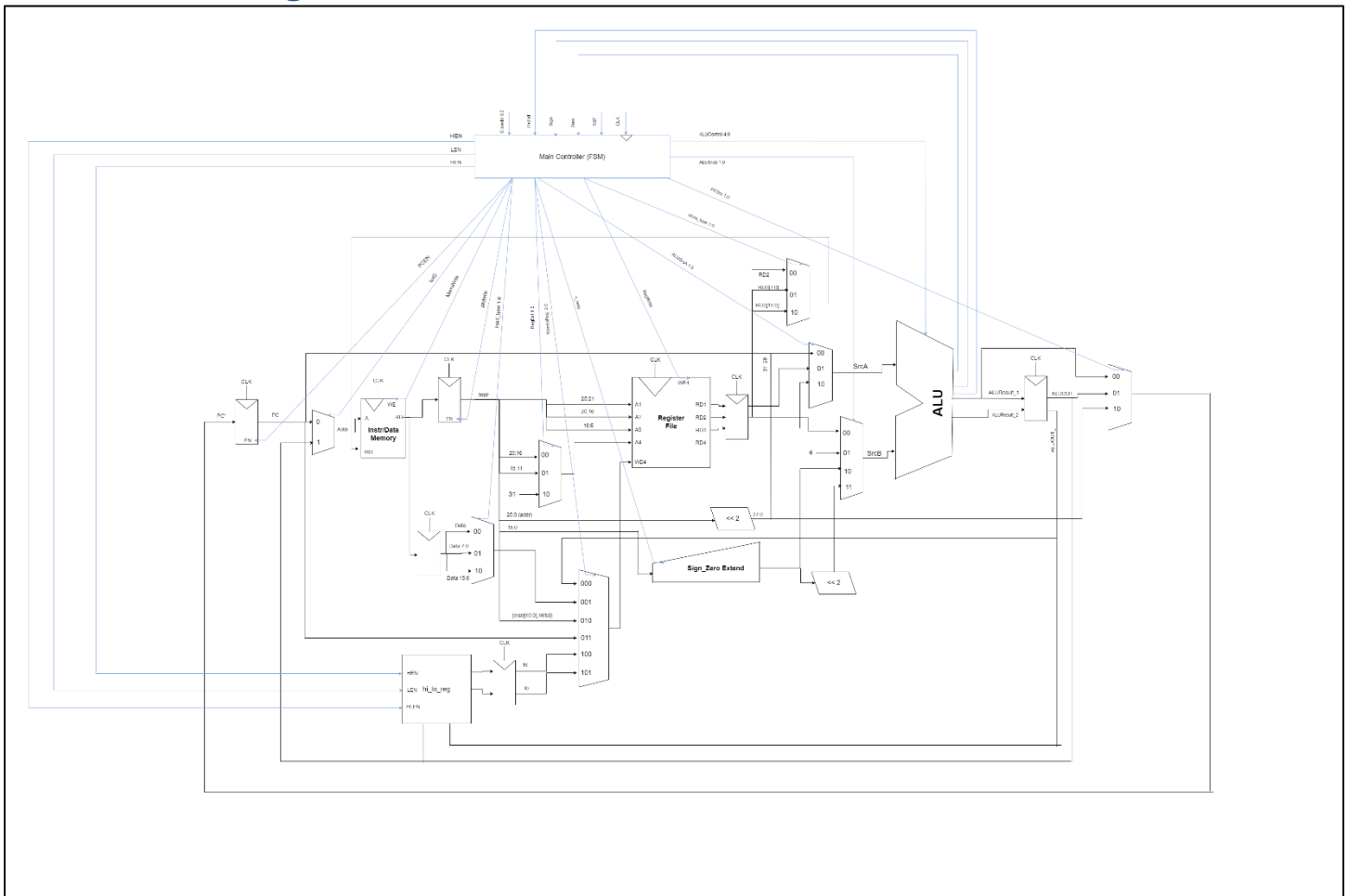
Green Team

MIPS Complete Architecture

The system contains 16 blocks:

- ALU
- REG_File
- Memory
- Shifter
- Shifter_Jump
- Sign_Zero_Extend
- hi_lo_reg
- Control_Unit
- Muxes:
 - mux_2x1_lorD
 - mux_4x1_RegDst
 - mux_8x1_MemtoReg
 - mux_2x1_ALUSrcA
 - mux_4x1_ALUSrcB
 - mux_4x1_PCSrc
 - mux_4x1_load MUX
 - mux_4x1_store MUX

Whole diagram



ALUOp encoding

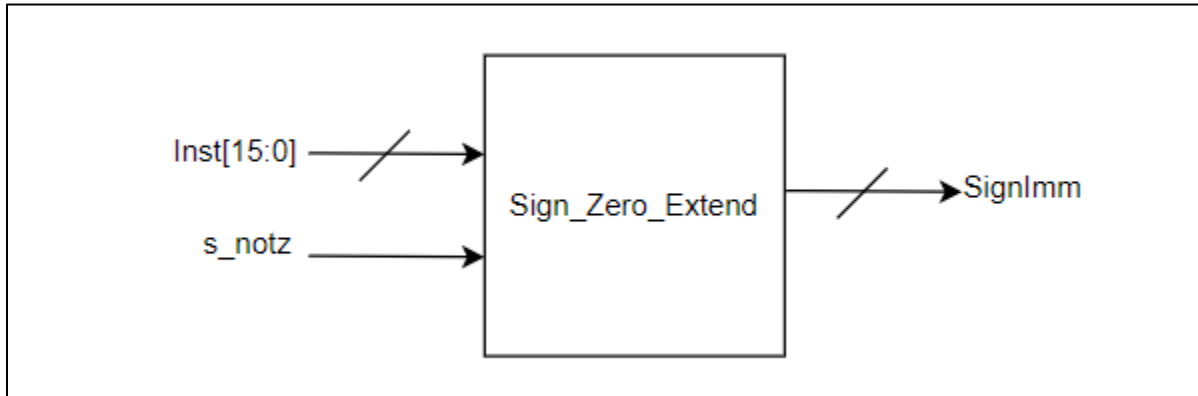
ALUOp	Meaning
0000	ADD
0001	SUB
0010	AND
0011	OR
0100	XOR
0101	NOR
0110	SL: Shift Left
0111	SR: Shift Right
1000	SRA: Shift Right Arithmetic
1001	Look at funct field

ALU decoder truth table

ALUOp	Funct	ALUControl
0000	X	00000 (add)
0001	X	00001 (sub)
0010	X	00010 (and)
0011	X	00011 (or)
0100	X	00100 (xor)
0101	X	00101 (nor)
0110	X	00110 (sll: shift left logical)
0111	X	00111 (srl: shift right logical)
1000	X	01000 (sra: shift right arithmetic)
1001	000000	00110 (sll: shift left logical)
1001	000010	00111 (srl: shift right logical)
1001	000011	01000 (sra: shift right arithmetic)
1001	000100	00110 (sllv: shift left logical variable)
1001	000110	00111 (srlv: shift right logical variable)
1001	000111	01000 (sra: shift right arithmetic variable)
1001	001000	00000 (add) "jr: jump register //PC = [rs]"

1001	001001	00000 (add) "jalr jump and link register //\$ra = PC + 4, PC =[rs]"
1001	010000	00000 (add) "mfhi move from hi //[rd] = [hi]"
1001	010001	00000 (add) "mthi move to hi //[hi] = [rs]"
1001	010010	00000 (add) "mflo move from lo //[rd] = [lo]"
1001	010011	00000 (add) "mtlo move to lo //[lo] = [rs]"
1001	011000	01001 (mult: multiply)
1001	011001	01100 (multu: multiply unsigned)
1001	011010	01010 (div)
1001	011011	01101 (divu: divide unsigned)
1001	100000	00000 (add)
1001	100001	01110 (addu: add unsigned)
1001	100010	00001 (sub)
1001	100011	01111 (subu: subtract unsigned)
1001	100100	00010 (and)
1001	100101	00011 (or)
1001	100110	00100 (xor)
1001	100111	00101 (nor)
1001	101010	01011 (slt: Set Less Than)
1001	101011	10000 (sltu: Set Less Than unsigned)

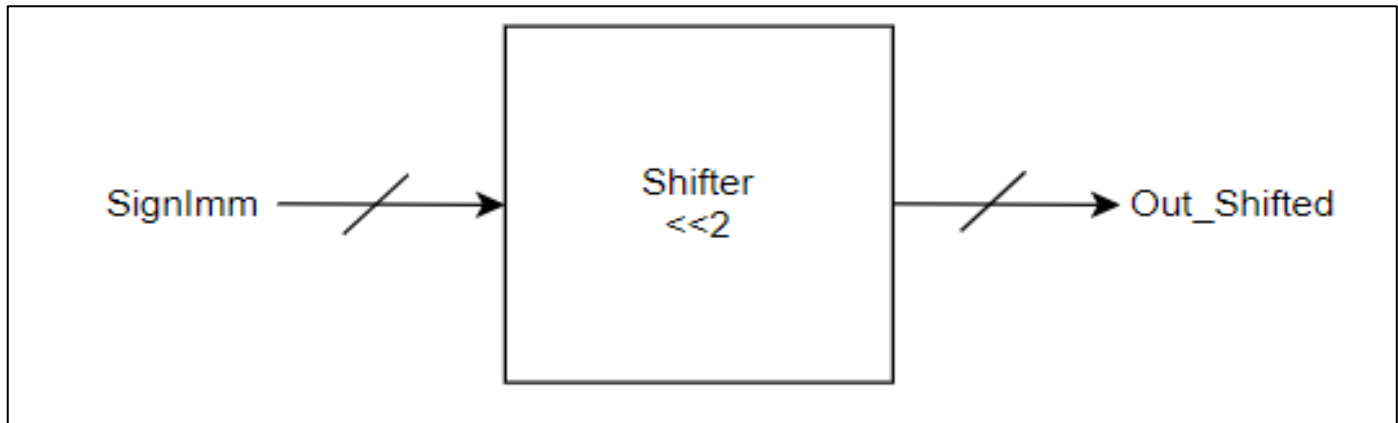
1. Sign_Zero_Extend Block



Port	Direction	Width	Description
Instr[15:0]	IN	16 bits	Immediate value
s_notz	IN	1 bit	Flag to select sign or zero extension
SignImm	OUT	32 bits	Extended value

- Because the 16-bit immediate might be either positive or negative, it must be sign-extended to 32 bits. The 32-bit sign-extended value is called SignImm.
- The sign extension simply copies the sign bit (most significant bit) of a short input into all of the upper bits of the longer output.
- Specifically, $\text{SignImm}[15:0] = \text{Instr}[15:0]$ and $\text{SignImm}[31:16] = \text{Instr}[15]$.
- Zero extension is used for some instructions as `addi,ori,xori`.

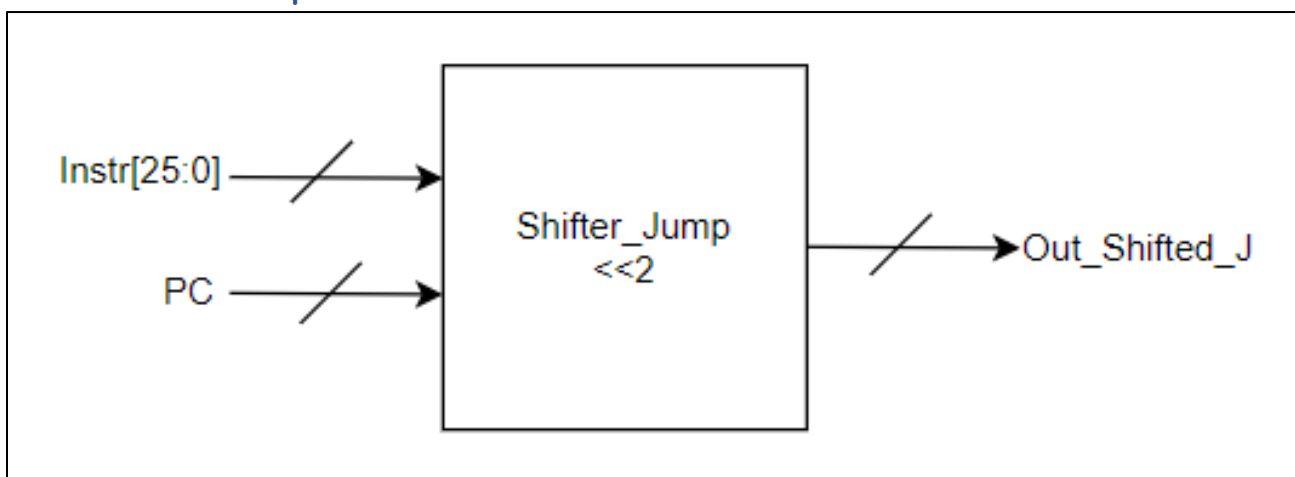
2. Shifter Block



Port	Direction	Width	Description
SignImm	IN	32 bits	Extended value
Out_Shifted	OUT	32 bits	Shifted value

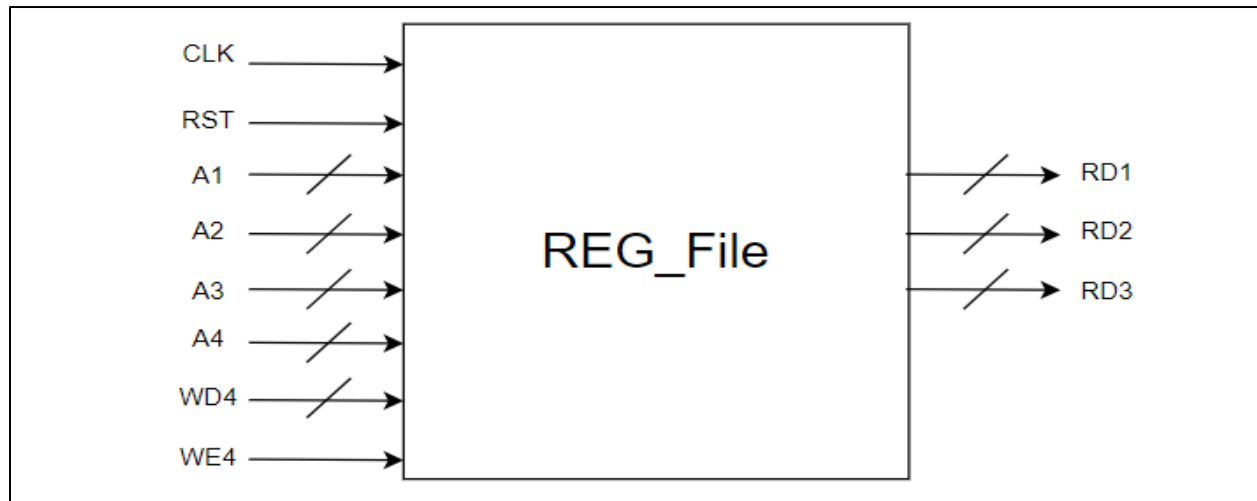
- In branch instructions such as beq, if the branch is taken the PC' value is incremented by the SignImm, however the value of the SignImm is the actual number of words and the MIPS is a byteaddressable system so a multiplication by 4 (<<2) is mandatory.
- $PC' = PC + 4 + \text{SignImm} \times 4$

3. Shifter_Jump Block



Port	Direction	Width	Description
Instr[25:0]	IN	32 bits	Extended value
PC	IN	32 bits	Program Counter
Out_Shifted_J	OUT	32 bits	Shifted value

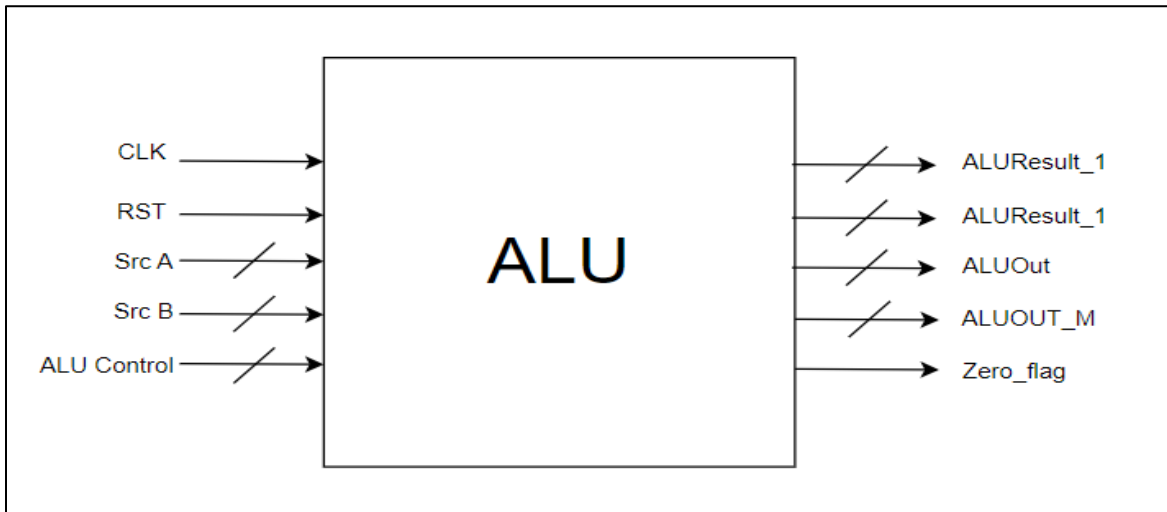
- In Jump instruction, the destination address is formed by left-shifting the 26-bit addr field of the instruction by two bits, then prepending the four most significant bits of the already incremented PC.
- $\text{Out_Shifted_J} = \{\text{PC}[31:28], \text{Instr} \ll 2\}$



4. REG_FILE Block

Port	Direction	Width	Description
A1	IN	5 bits	Address to first source register
A2	IN	5 bits	Address to second source register
A3	IN	5 bits	Address for shamt
A4	IN	5 bits	Destination address for register
WE4	IN	1 bit	Writing enable signal
WD4	IN	32 bits	Register to store the needed value to be written inside the register file
CLK	IN	1 bit	Used as a clock signal
RST	IN	1 bit	Used to reset the register file
RD1	OUT	32 bits	Output_1 from register file
RD2	OUT	32 bits	Output_2 from register file
RD3	OUT	32 bits	Output_3 from register file

5. ALU Block

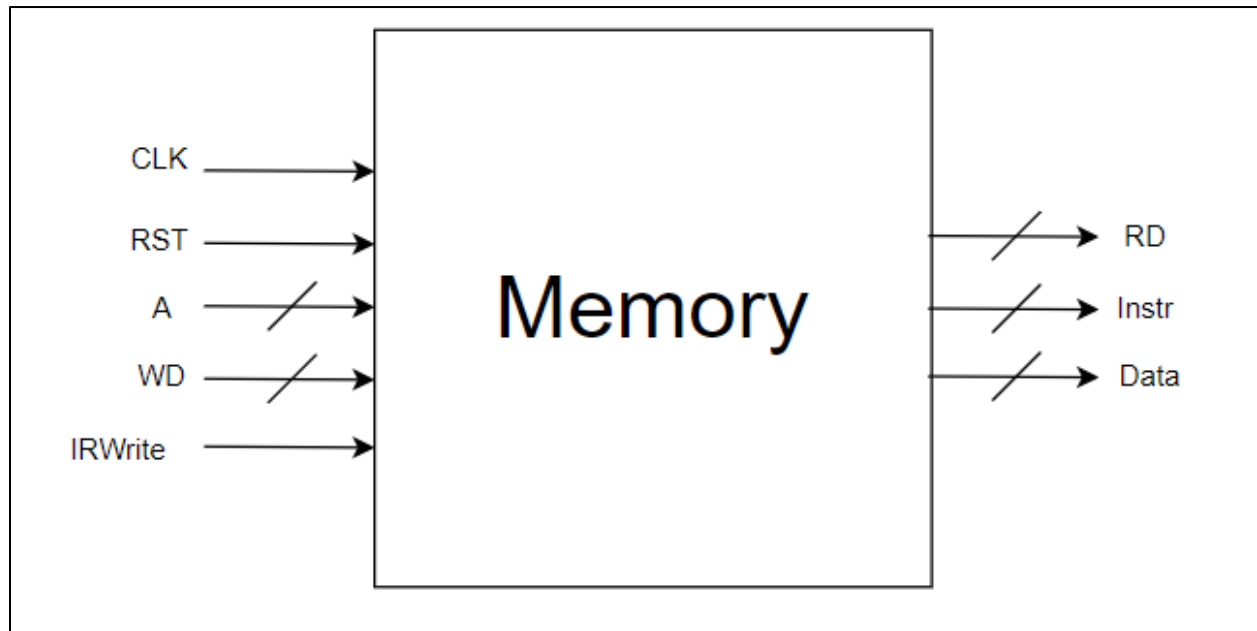


Port	Direction	Width	Description
SrcA	IN	32 bits	Operand_1 for ALU
SrcB	IN	32 bits	Operand_2 for ALU
ALUControl	IN	6 bits	Control signal to detect the operation which the ALU will perform.
CLK	IN	1bit	Used as a clock signal
RST	IN	1bit	Used to reset the output of the ALU
ALUResult_1	OUT	32 bits	First combinational output of ALU
ALUResult_2	OUT	32 bits	Second combinational output of ALU
ALUOut	OUT	32 bits	First sequential output of ALU
ALUOut_	OUT	32 bits	Second sequential output of ALU
Zero_flag	OUT	1 bit	The flag is high if the ALU output equals 0 otherwise it is low

- The ALU performs certain operations according to ALUControl signal.
- It has two input signals for data and an two output signal to accommodate the largest size operation which is the multiplication. (32bit * 32bit = 64 bits)

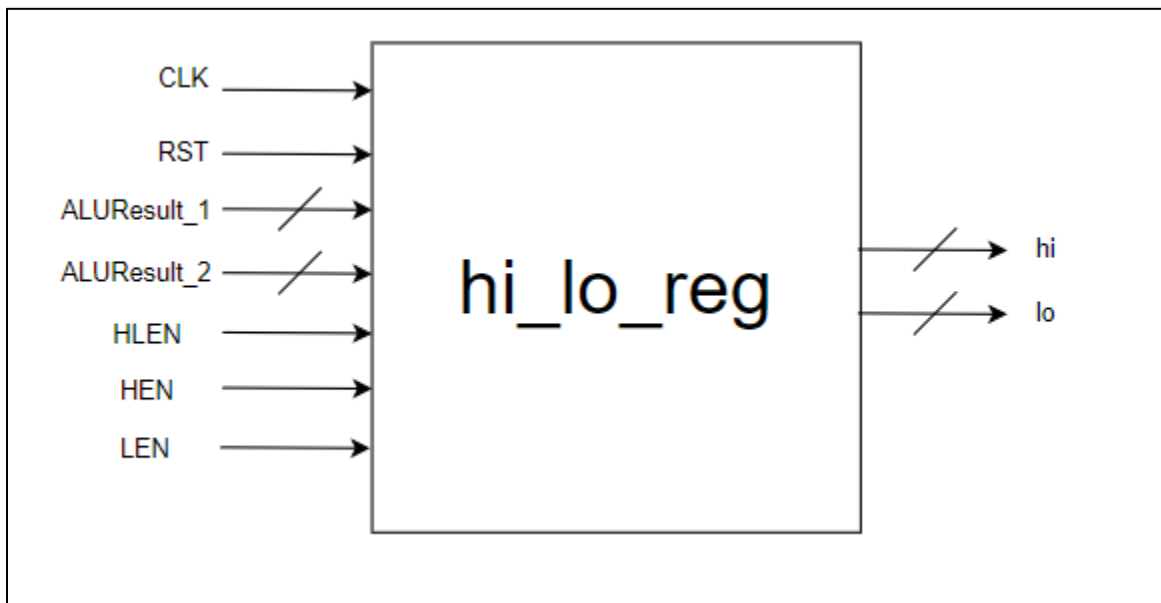
- It has zero flag which is an indication for zero output.

6. Memory



Port	Direction	Width	Description
A	IN	32 bits	An address to a dedicated place inside the memory
WD	IN	32 bits	Register to store the needed value to be written inside the memory
WE	IN	1 bit	Writing enable signal
IRWrite	IN	1 bit	Flag for saving the output into a register
CLK	IN	1 bit	Used as a clock signal
RST	IN	1 bit	Used to reset the memory
RD	OUT	32 bits	The output of the memory
Instr	OUT	32 bits	Output as Instruction
Data	OUT	32 bits	Output as Data

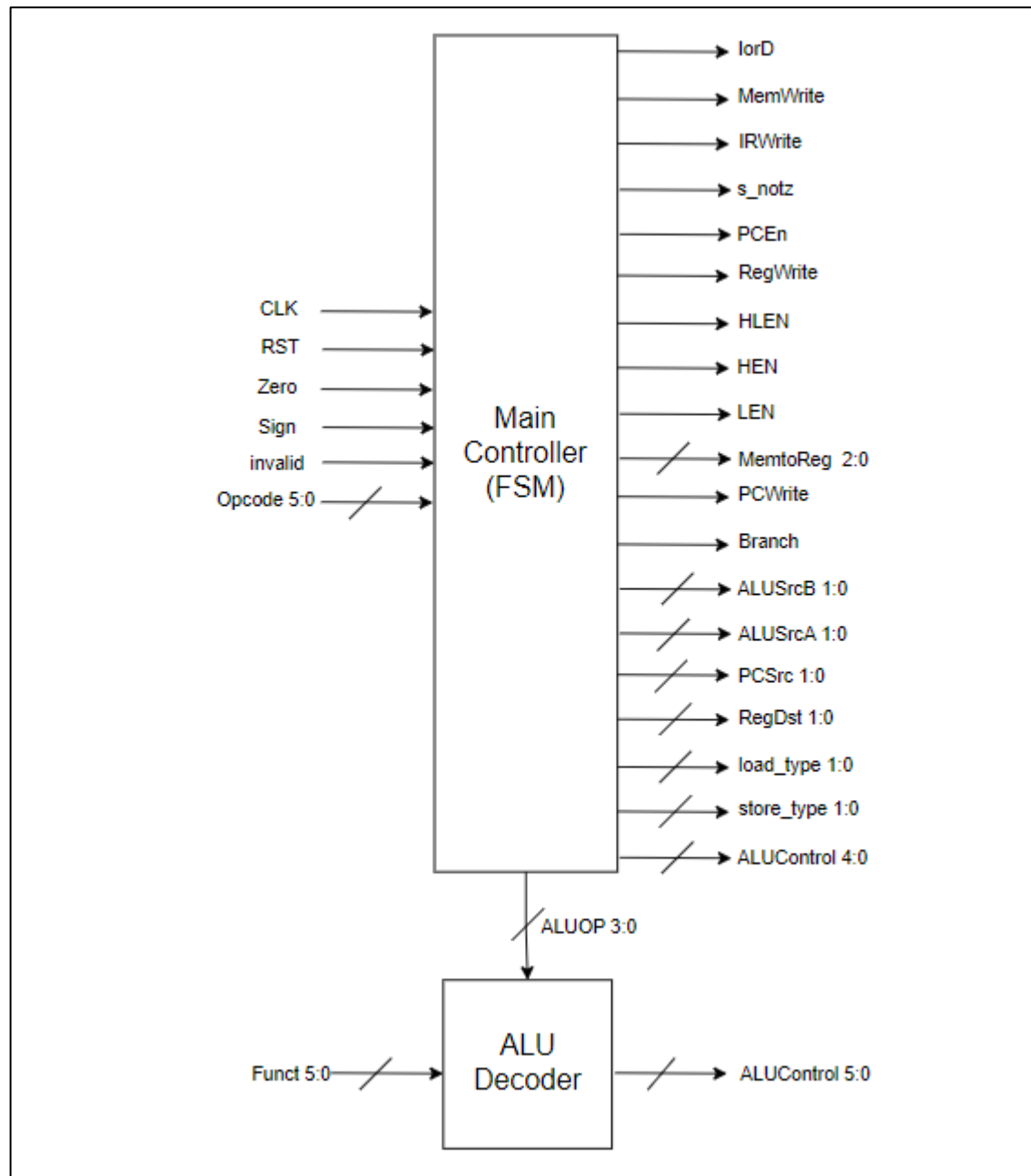
7. hi_lo_reg Block



Port	Direction	Width	Description
ALUResult_1	IN	32 bits	First combinational output of ALU
ALUResult_2	IN	32 bits	Second combinational output of ALU
CLK	IN	1 bit	Used as a clock signal
RST	IN	1bit	Used to reset the output registers
HLEN	IN	1bit	Enable loading in both hi register
HEN	IN	1bit	Enable loading in hi register
LEN	IN	1bit	Enable loading in lo register
hi	OUT	32 bits	High register
lo	OUT	32 bits	Low register

- The output of the multiplication of two 32bit operands is a 64 bit long. Thus the 32 most significant bits of the product are stored in the hi register while the 32 least significant bits of the product are stored in the lo register.
- For other instructions like mthi and mtlo, the output of the ALU is stored in the hi register only or the lo register only thus enable signals are mandatory.

8. Control unit Block



Port	Direction	Width	Description
CLK	Input	1 bit	Clock signal of the control unit
RST	Input	1 bit	Reset signal of the control unit
Invalid	Input	1 bit	Input signal which asserted when we execute invalid operation
Sign	Input	1 bit	Input signal asserted when the result of ALU operation is negative
Zero	Input	1 bit	Input signal asserted to one when the result of ALU is zero
Opcode	Input	6 bits	Input signal determine the operation that will be executed.
IorD	Output	1 bit	Selection signal of the multiplexer that determine the type of the fetched word from the memory "instruction or data"
MemWrite	Output	1 bit	Write enable signal of the memory
IRWrite	Output	1 bit	Enable to the register which store the instruction
S_notz	Output	1 bit	Signal to the sign extend block to determine the type of the extend that will be performed
PCEn	Output	1 bit	Enable signal to PC register which store PC
RegWrite	Output	1 bit	Enable signal to write in the register file
HLEN	Output	1 bit	Enable signal to write at HI & LO FFs
HEN	Output	1 bit	Enable signal to write at HI FF
LEN	Output	1 bit	Enable signal to write at LO FF
MemtoReg	Output	3 bits	Selection signal of the multiplexer that select the data that will be written at the register file
PCWrite	Output	1 bit	Enable signal to write at PC FF
Branch	Output	1 bit	Signal that will be asserted in case of branch instruction
ALUSrcB	Output	2 bits	Selection signal of the multiplexer that select the second operand of the ALU.
ALUSrcA	Output	2bits	Selection signal that select the first operand of the ALU

PCSrc	Output	2 bits	Selection signal of the multiplexer that select the input of PC FF
RegDst	Output	2 bits	Selection signal of the multiplexer that select the address of the register that will store the data in the register file.
Load_tyoe	Output	2 bits	Selection signal of the multiplexer that detect the type of data that will be loaded from the memory "byte, word, etc.."
Store_type	Output	2 bits	Selection signal of the multiplexer that detect the type of data that will be stored in the memory "byte, word, etc.."
ALUControl	Output	5 bits	Input to ALU to detect the operation that will be performed
ALUOP	Output	4 bits	Input to ALUDecoder to detect ALUControl

Test bench

→ Test Cases:

Several test cases were executed to verify the MIPS processor's functionality:

1. *Test Case 1-46:*

→ Instructions Loaded: all R-type instruction (add,sub,sll,srl,...), j(jump), I-type(bltz,beq,bne,blez,bgtz,addi,and,or,xori,lui,lw,sw,...)

Simulation Outcome: Passed

2. *Test Case 47:* Reference test bench

→ Expected Result: memory [84] =7

Simulation Outcome: Passed

3. *Test Case 48:* Factorial of 5

→ Expected Result: register file [12] = 120

Simulation Outcome: Passed

4. *Test Case 49:* GCD (68,119)

→ Expected Result: register file [17] = 17

Simulation Outcome: Passed

5. Test Case 50-60:

➔ Instructions Loaded: (lb, lh, lw, sb, sh, sw, slti, sltiu, dividing by zero case, storing at r[0])


Simulation Outcome: Passed

Simulation Results:


- Reference TB

		Msgs								
 /MIPS_TOP_TB/DUT/mem/MEM[84]	7		0	12	10					

- Factorial of 5

		Msgs								
 /MIPS_TOP_TB/DUT/re/REG[12]	120		120					0		

- GCD(119,68)

		Msgs								
 /MIPS_TOP_TB/DUT/re/REG[17]	17		17							