**Title:** A 24-bit single cycle CPU that can perform R-type, I-type, J-type instructions.

#### **Project Objectives:**

Our objective is to implement a 24 bit single cycle CPU which can perform the following instructions:

- R-type
- I-type
- J-type

#### Types of instructions and their format:

#### R-type:

MSB LSB

	Opcode(4)	rs(4)	rt(4)	rd(4)	shamt(4)	Function(4)
2:	₹ 70	19 16	15 12	11 8	7 4:	3 0

#### I-type:

	Opcode(4)	rs(4)	rt(4)	Immediate(12)
23	20	19 16	15 12	11 0

#### J-type:

	Opcode(4)	Target(20)
23	20	19 0

Here,

**Opcode:** Operation Code

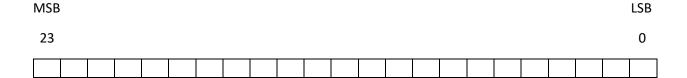
rs: Source Register

rt: Temporary Register

rd: Destination Register

shamt: Shift Amount

Data Bits: 24 bits throughout the project



Instructions	Opcode
R type	0000
LW	0111
SW	1000
BEQ	1001
BNE	1010
Jump	1111

## **TABLES**

#### **Instruction List:**

R type						
Instruction	Name	Action	Function			
AND rd, rs, rt	AND	rd= rs & rt	0000			
OR rd, rs, rt	OR	rd= rs   rt	0001			
ADD rd, rs, rt	Addition	rd= rs + rt	0010			
SLT rd, rs, rt	Set Less Than	if(rs <rt) else="" rd="0&lt;/td"><td>0011</td></rt)>	0011			
NOR rd, rs, rt	NOR	rd= (rs   rt)'	0100			
NAND rd, rs, rt	NAND	rd= (rs & rt)'	0101			
SUB rd, rs, rt	Subtraction	rd= rs - rt	0110			
SRL rd, rt ,shamt	Shift Right Logical	rd= rs >> shamt	0111			
SLL rd, rt ,shamt	Shift Left Logical	rd= rs << shamt	1000			

I type							
Instruction	Name	Action	Function				
SW rt, offset(rs)	Store Word	M[offset + rs] = rt	XXXX				
LW rt, offset(rs)	Load Word	rt = M[offset + rs]	XXXX				
BEQ rs, rt, offset	Branch On Equal	if(rs==rt) than pc = pc + offset	XXXX				
BNE rs, rt, offset	Branch On Not Equal	if(rs!=rt) than pc = pc + offset	XXXX				
ADDI rt, rs, imm	Add immediate	rt = rs + imm	XXXX				
SUBI rt, rs, imm	Subtract immediate	rt= rs-imm	XXXX				

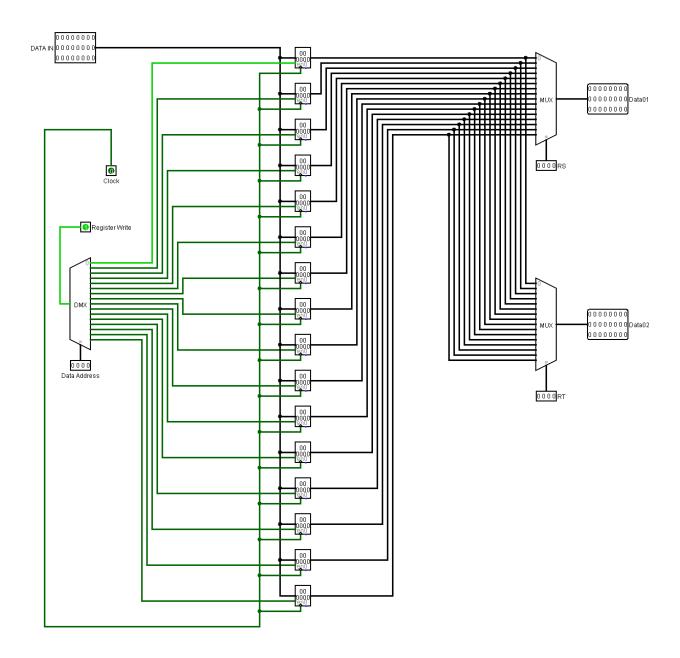
J type						
Instruction	Name	Action	Opcode			
J Target	JUMP	pc[0-19] = target; pc[20- 23] = (pc+1)[20-23];	1111			

Control Unit Signals											
Instruction	Reg	ALUSrc	MemTo	Reg	Mem	Mem	BEQ	BNE	Jump	ALU	ALU
	Dest		Reg	Write	Read	Write				OP1	OP0
R-Type	1	0	0	1	0	0	0	0	0	1	0
LW	0	1	1	1	1	0	0	0	0	0	0
SW	0	1	0	0	0	1	0	0	0	0	0
BEQ	0	0	0	0	0	0	1	0	0	0	1
BNE	0	0	0	0	0	0	0	1	0	0	1
JUMP	0	0	0	0	0	0	0	0	1	Χ	Х
ADDI	0	1	0	1	0	0	0	0	0	0	0
SUBI	0	1	0	1	0	0	0	0	0	0	1

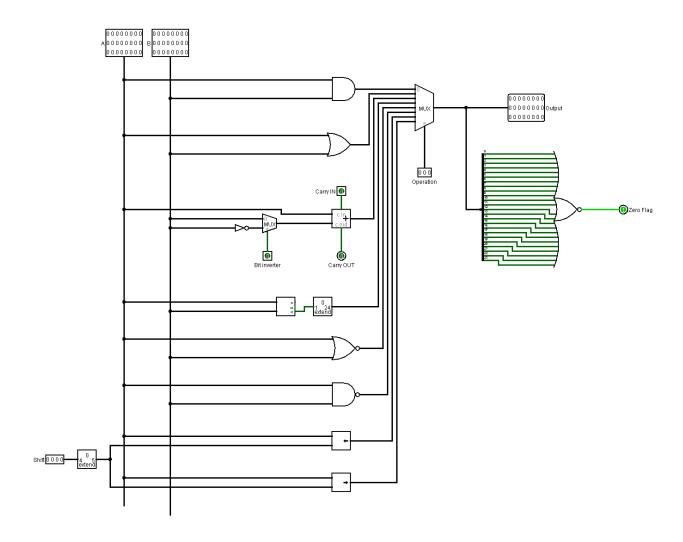
ALU CONTROL								
Instruction	<b>Control Unit</b>	ALUOp	Instruction	Function	ALU Action	ALU Control		
	Opcode		Operation	Field		Input		
R-Type	0000	10	AND	0000	AND	0000		
R-Type	0000	10	OR	0001	OR	0001		
R-Type	0000	10	ADD	0010	ADD	0010		
R-Type	0000	10	SLT	0011	SLT	0011		
R-Type	0000	10	NOR	0100	NOR	0100		
R-Type	0000	10	NAND	0101	NAND	0101		
R-Type	0000	10	SUB	0110	SUB	0110		
R-Type	0000	10	SRL	0111	SHIFT LEFT	0111		
R-Type	0000	10	SLL	1000	SHIFT RIGHT	1000		
LW	0111	00	Load	XXXX	ADD	0010		
SW	1000	00	Store	XXXX	ADD	0010		
BEQ	1001	01	Branch Equal	XXXX	SUB	0110		
BNE	1010	01	Branch not	XXXX	SUB	0110		
			Equal					
JUMP	1111	XX	Jump	XXXX	XX	XXXX		
ADDI	1011	00	Add imm	XXXX	ADD	0010		
SUBI	1100	01	Sub imm	XXXX	SUB	0110		

ALU Control Table for ALU Operation							
ALU	JOp		Functi	on Bits		Operation	
ALUOp1	ALUOp0	F3	F2	F1	F0		
0	0	X	X	X	X	0010	
0	1	Х	Х	Х	Х	0110	
1	0	0	0	0	0	0000	
1	0	0	0	0	1	0001	
1	0	0	0	1	0	0010	
1	0	0	0	1	1	0011	
1	0	0	1	0	0	0100	
1	0	0	1	0	1	0101	
1	0	0	1	1	0	0010	
1	0	0	1	1	1	0111	
1	0	1	0	0	0	0110	
1	1	Х	X	Х	X	XXXX	

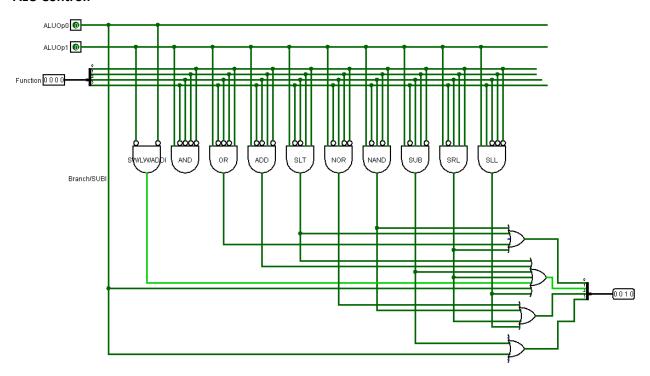
# **Register Circuit:**



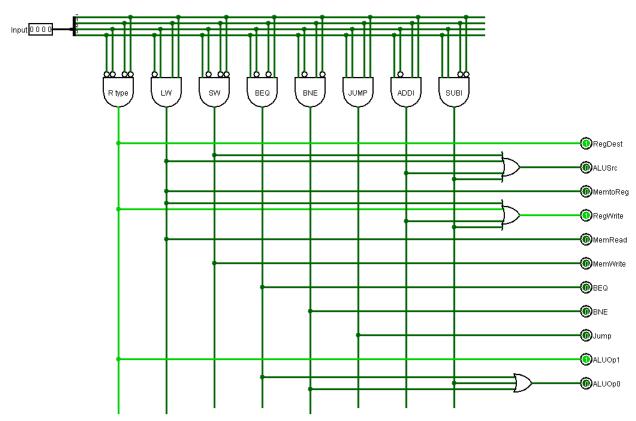
#### ALU:



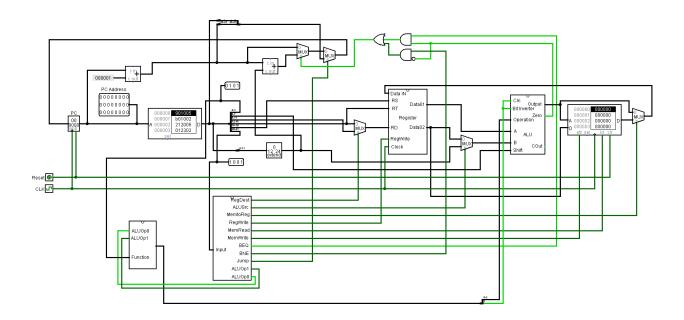
#### **ALU Control:**



#### **Control Unit:**



# Datapath:



# Sample program:

## **Binary equivalent**

## **Hexadecimal equivalent**

1011 0000 0000 0000 0000 0000	b00000
1011 0001 0001 0000 0000 0000	b11000
1011 0010 0010 0000 0000 0101	b22005
1001 0010 0000 0000 0000 0011	920003
1011 0001 0001 0000 0000 0001	b11001
1011 0000 0000 0000 0000 0001	b00001
1111 0000 0000 0000 0000 0101	f00003