Faculty of Engineering & Technology

Electrical & Computer Engineering Department

Computer Architecture

ENCS4370

Project #2 report

Multi cycle Processor

Prepared By:

1. Omar Hamayel 1220356
2. Qusay Bdire 1220649
3. Lana Hamayel 1200209

Instructor : Ayman Hroub

Date : 14/1

Section : 2

Abstract :

To create and install a multicycle 16-bit RISC simulator processor with Verilog, this is the project. There will be an instruction word size of 16 as specifications with the capability of having eight general-purpose registers along with a program counter with a return register for function calls. There will be and have an instruction set architecture consisting of three instruction types, namely R-type, I-type, and J-type, and it will also separate instruction and data memories. It has many functions such as arithmetic logic, memory access, and control instructions.

The multicycle design optimizes resource usage to issue multiple clock cycles for executing instructions such as ALU and memory. Deliverables will be complete design and implementation of the entire Datapath and control path, including the generation of control signals, and processor verification via simulation. A testbench was created to execute and validate code sequences in the ISA along with in-depth outputs and simulation reports.

The project is modular, correct, and efficient enough to make the whole multicyle processor work and gain hands-on experience in computer architecture and hardware design.

Table of Contents

Table of Figure

1. Design and Implementation:

Data Path:

The design of the data path is for a multi-cycle processor through its five stages: Instruction Fetch, Instruction Decode, Instruction Execution, Memory Access, and Write Back. This has been achieved with this processor in that all components operate under a common synchronized clock triggered by the positive edge. All assembled from the following components.

➢ Program Counter (PC):

Next instruction addresses refer PC Counter. Figure 1 illustrates 16-bit PC register, a mux 6x1 used to choose the PC input according to the 3-bits PC source signal that generated by PC control, plus 1 component for normal update on the PC, and wires between components. The inputs of the PC feature five options:

1. The normal PC (PC+1).
2. The jump target address (JTA) which is (PC [15:9] || 9-bit offset).
3. The branch target address (BTA) which is (Current PC + sign extended immediate).
4. A PC for the RET instruction which is (PC = RR).
5. A PC for the FOR-Loop instruction which is (PC = Value Rs).

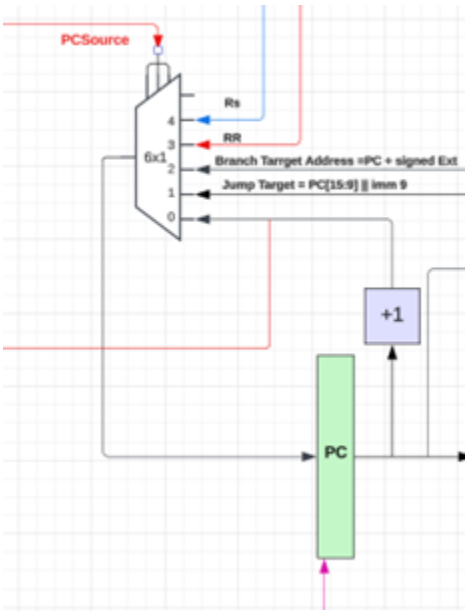

Figure 1:Program Counter Component

Also, the output of PC component enters the Instructions Memory component.

➢ Instructions Memory:

This specific component of the memory is the instruction memory, which is defined as physical memory that is used to carry the load of instructions. It carries 16-bit instructions for four types of instructions: R and I types, as well as J type. Figure 2 presents the model of the instruction memory component, which accepts an input of 16 bits-an address-and produces an output of 16 bits, which it terms an instruction. Apart from that, it also has an IR buffer to hold temporarily the result of the current stage, IF, for usage in the next stage, ID, under the control of a shared synchronous clock. Thus, once the operation is over, the instruction is successfully fetched.

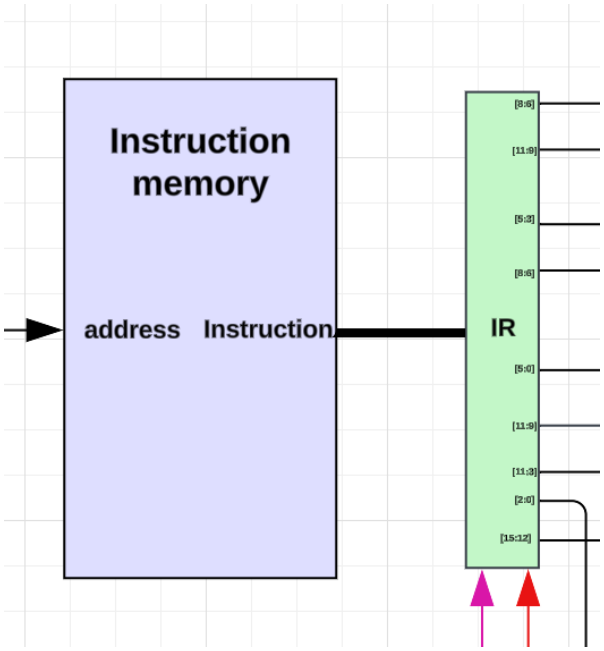

Figure 2:Instruction Memory

➢ Registers File:

A register file is a collection of registers which are accessed in a temporary way for conveying the data between the processor's memory and its functional units. The processor employs two specific registers: Rs1 and Rs2, for the purpose of reading data. Bus1 is assigned to carry Rs1, and Bus2 holds data from Rs2. The register, called Rd, accepts data to be written. The control signal RegWr being set to 1 means that the data coming in on Bus_W will be written into Rd.

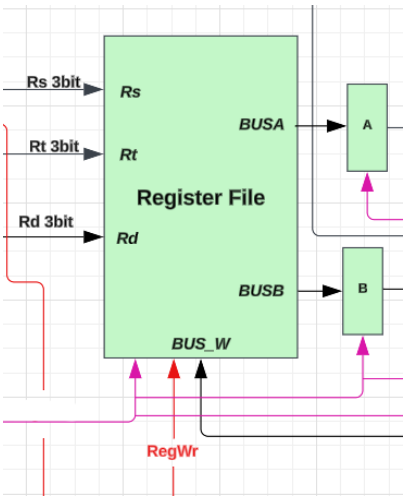

Figure 3:Registers File

➢ Sign Extender:

The Extender role is to extend a 6-bit immediate number, into a 16-bit number. It has two inputs, one 6-bit input, which is the number that needs to be extended to 16-bit, and a EXT OP input, that specifies if this is signed or unsigned extending. It has only one 16-bit output, which is the extended number.

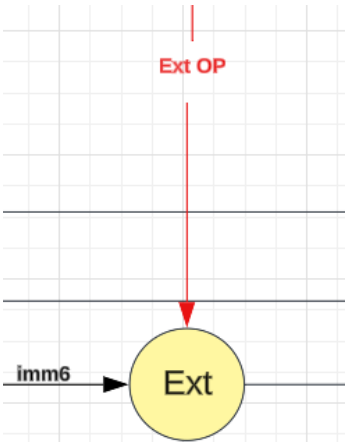

Figure 4:Extender

➢ Arithmetic & Logical Unit (ALU):

The ALU is a component of the CPU that handles arithmetic and logic operations, such as adding, subtracting, and logical comparisons.

In our Datapath, the ALU has two main inputs, where it gets two sets of 16 pieces of information, called bits. The ALU can do several things with these bits: it can combine them using a logic rule (AND), add them together, or subtract one from the other , or shifting them. Then, based on a small 3-bit control signal (ALUCtrl), To picks which math or logic operation to use. The chosen result is then sent out of the ALU as the definitive answer.

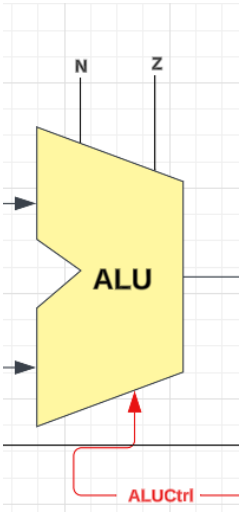

Figure 5:ALU

➢ Data Memory:

Figure 3 illustrates the data memory, which acts as the storage space for the data the processor uses. This data memory relates to the rest of the Datapath through an address input and two data lines: one to put data into storage and another to get data out. Here's how it works: When the MemRd signal is 1, the data memory finds the data at the specified address and sends it out. This is for reading data. When the MemWr signal is 1, the data memory takes in data from the line and keeps it at the address given. This is for writing data.

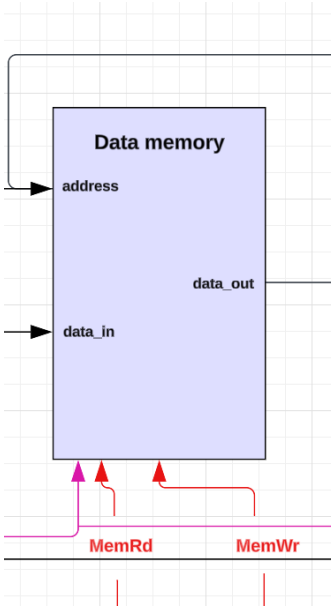

Figure 6:Data Memory

➢ Adder for BTA

Since the implementation of this processor does not support the instruction to use the same component in the data path more than once, an additional adder was added to the data path to calculate the branch target adders, while the ALU is handling the condition checking of the branch instruction. Inputs and outputs for adder are:

1. 16-bit input with the value of the PC.
2. 16-bit input immediate value after being signed extended.
3. 16-bit output of the branch target address that feeds in the 6x1 MUX of PC source.

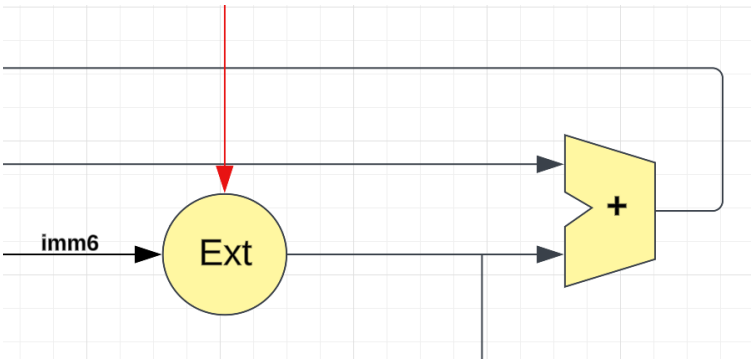

Figure 7:Adder for BTA

➢ Write Back Stage:

In this stage, the output of the ALU or the output of the memory is stored back, and written on the register file, the choice between ALU output and memory output is done by using a 2x1 Mux that selects based on a signal called WBData, if the data written back is from memory or ALU output. For the data to be written on the register file, the control signal RegWr must be set ('1').

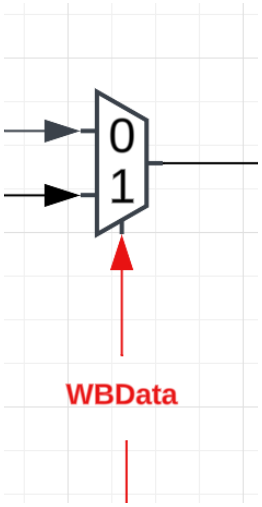

Figure 8:Write Back Stage

2. Control Signals:

Main central control unit, PC control unit and ALU control unit are three distinctive units for generating a lot of control signals. We thus split the control signals coming from these 3 units: to understand each control signal better and to build and test it more efficiently. Control signals are used whenever instruction type is to distinguish which functional unit needs to be activated; for example, for R-type instructions, write back data goes to ALU, but for load instructions, it is the data loaded from memory. So basically, control signals are really the lifelines of flow in our processor; also, the processor uses them to help him distinguish instructions.

Main control														
input		output												
OP	Function	state	pcwriteUNcond	IRWrite	RegDstB	RegDstC	ExtOP	RegWr	ALUSrcA	ALUSrcB	ALUOp	MemRd	MemWr	WBData
R-TYPE														
0	0	4	1	1	0	1	x	1	1	0	and	0	0	0
0	1	4	1	1	0	1	x	1	1	0	add	0	0	0
0	2	4	1	1	0	1	x	1	1	0	sub	0	0	0
0	3	4	1	1	0	1	x	1	1	0	sll	0	0	0
0	4	4	1	1	0	1	x	1	1	0	srl	0	0	0
I-Type Instructions														
2	xxxx	4	1	1	1	0	1	1	1	1	and	0	0	0
3	xxxx	4	1	1	1	0	0	1	1	1	add	0	0	0
4	xxxx	5	1	1	1	0	0	1	1	1	add	1	0	1
5	xxxx	4	1	1	1	x	0	0	1	1	add	0	1	x
6	xxxx	3	0	1	0	x	1	0	1	1	sub	0	0	x
7	xxxx	3	0	1	0	x	1	0	1	1	sub	0	0	x
8	xxxx	4	0	1	1	0	x	1	0	0	add	0	0	0
J-Type Instructions														
1	0	2	0	1	xxx	xxx	x	0	xxx	xxx	xxx	0	0	x
1	1	2	0	1	xxx	xxx	x	0	xxx	xxx	xxx	0	0	x
1	2	2	0	1	xxx	xxx	x	0	xxx	xxx	xxx	0	0	x

Table 1:Main Control

ALU Ctrl	
input	output
ALUOp	ALUCtrl
R-TYPE	
0	AND
0	ADD
0	SUB
0	SLL
0	SRL
I-TYPE	
2	ANDI
3	ADD
4	ADD
5	ADD
6	SUB
7	SUB
8	SUB
J-TYPE	
1	XX
1	XX
1	XX

Table 2:ALU Control

PC Ctrl					
input				output	
op	pcwriteUNcond	Z flag	N flag	PCWrite	PCSource
R-TYPE					
0	1	x	x	1	0
0	1	x	x	1	0
0	1	x	x	1	0
0	1	x	x	1	0
0	1	x	x	1	0
I-TYPE					
2	1	x	x	1	0
3	1	x	x	1	0
4	1	x	x	1	0
5	1	x	x	1	0
6	0	1	x	3	2
7	0	0	x	3	2
8	0	x	1	5	4
J-Type Instructions					
1	0	x	x	2	1
1	0	x	x	2	1
1	0	x	x	2	3

Table 3:PC Control

Now, we're ready to find the Boolean equations for each control signal.

➢ Register Write Signal

The RegWr signal determines whether data should be written to the register file. It is set to 1 for instructions that require writing to the register file and 0 for those that do not.

RegWr = AND + ADD + SUB + SLL + SRL + ADDI + ANDI + LW + FOR.

0 → don't write on the register file.
1 → write on the register file.

➢ Extension signals

The EXT OP signal determines if the instruction types are I-type. If the opcode is Branch, the least significant 6 bits are sign-extended. else the extension depends on whether the opcode represents a logical operation or not

0 → logical instructions. (Unsigned)

1 → Branch target and all instructions except logical instructions. (Signed)

➢ ALU source signal

The ALUSrcA signal determines which input is provided to the ALU based on the opcode. It selects between -1 or the output from the register file (BusA). And the ALUSrcB signal selects between the extended immediate value or the output from the register file (BusB).

ALUSrcA = AND + ADD + SUB + SLL + SRL + ADDI + ANDI + LW + SW.

0 → -1.

1 → BusA.

ALUSrcB = ADDI + ANDI + LW + SW.

0 → BusB.

1 → extended immediate.

➢ Read Write Memory Signals

These signals control whether data memory is read from, written to, or disabled:

MemRd = LW.

0 → don't read from memory.

1 → read from memory.

MemWr = SW.

0 → don't write from memory.

1 → write to memory.

➢ ALU operation

This signal dictates the operation that the ALU will execute, which includes AND ADD, SUB, SLL and SRL. It determines whether the ALU performs logically AND, addition, subtraction or shift left or shift right.

ALUCtrl [0] = SLL + ADD + ADDI + FOR + LW + SW.
ALUCtrl [1] = SUB + SLL+ BEQ + BNE.
ALUCtrl [2] = SRL.

000 → AND → AND, ANDI.
001 → ADD → ADD, ADDI, FOR, LW, SW.

010 → SUB → SUB, BEQ, BNE.
011 → SLL → SLL.

100 → SRL → SRL.

➢ RegDstB source

This signal determines the address of the second output of the register file.

RegDstB = ANDI + ADDI + LW + SW + FOR.

0 → ins [5:3]. (R-Type)
1 → ins [11:9]. (I-Type)

➢ RegDstC source

This signal selects the address of the register that will be written to.

RegDstC = AND + ADD + SUB + SLL + SRL.

0 → ins [5:3]. (R-Type)
1 → ins [11:9]. (I-Type)

➢ Write back signal

This signal specifies the data that will be written into the register file.

WBData = ~ (AND + ADD + SUB + ANDI + ADDI + LW + FOR).

0 → ALU Output
1 → Data Out from Memory.

➢ PC source

This signal selects the correct value for the Program Counter (PC).

PCSource [0] = JMP + CALL + RET.
PCSource [1] = BEQ + BNE + RET.
PCSource [2] = FOR.

000 → R-Type Instruction.
001 → Jump Target.

010 → Branch Target.
011 → RR.

100 → Rs

.

3. Datapath Block Diagram:

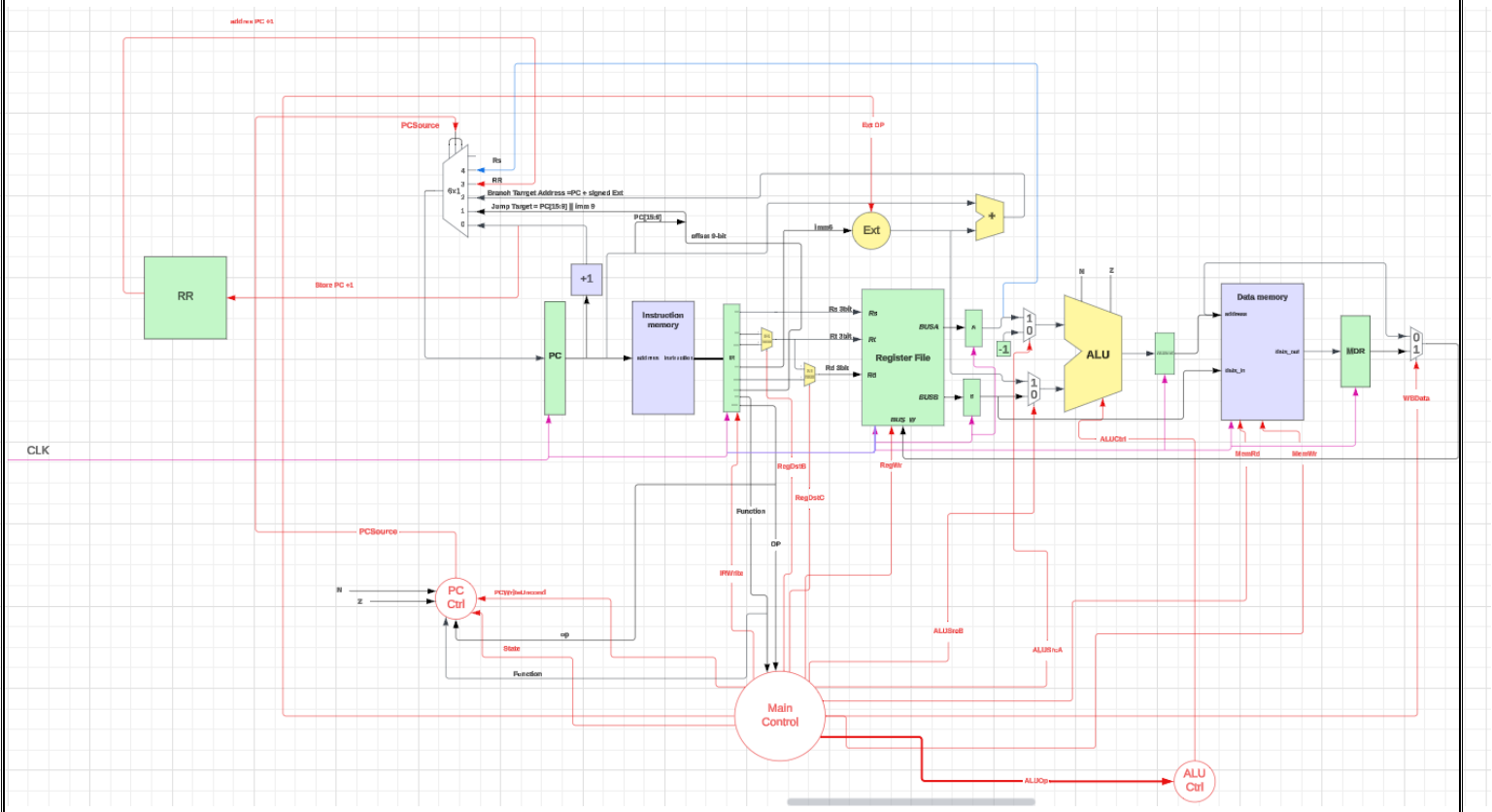

Figure 9:Datapath

4. Main control state diagram:

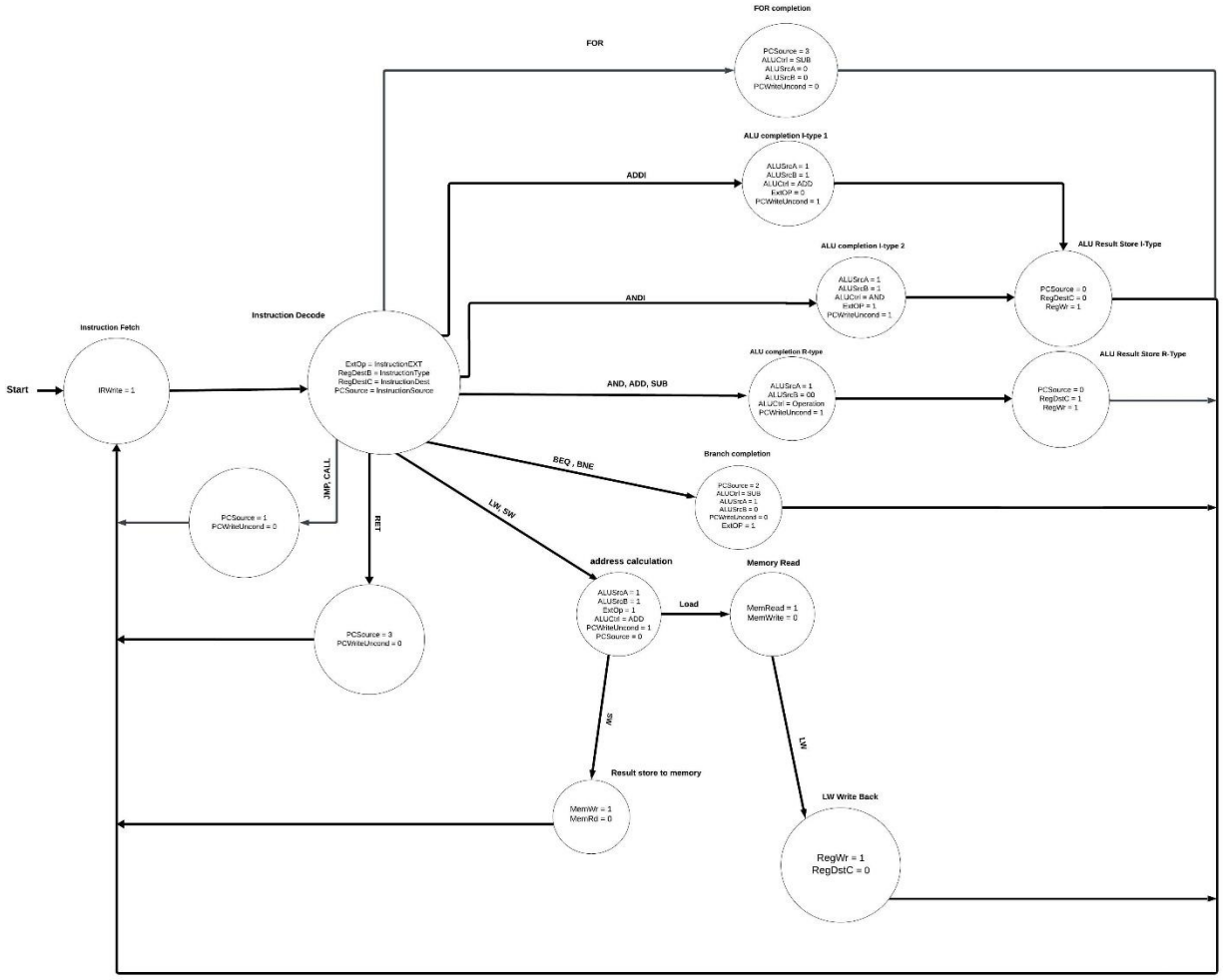

Figure 10:Main control state diagram

5. Testing:

➢ ADDI Instructor

```
-----------------Reset Program-----------------

Time=0 | ALUOut=xxxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

-----------------Reset Program-----------------


-----------------Fetch Cycle-----------------

PC=0000000000000000
Instruction = 0011011100000100


-----------------Decode Cycle-----------------

Time: 30 | Opcode: 0011 | Function: 100 | State: 100 | IRWrite: 1 | RegDstB: 1 | pcwriteUNcond: 1 | RegDstC: 0 | ExtOP:0 | RegWr: 1 | ALUSrcA: 1 | ALUSrcB: 1 | ALUOp: 001 | MemRd: 0 | MemWr: 0 | WBData: 0
RegSucre4=0000000000000100


ReggDst: 011, outA: 0000000000000100, outB: 0000000000000011

-----------------Execute Cycle-----------------

Extended immediate= 4, BusA= 4, BusB= 3

Alusurce = 001

a= 0000000000000100, b=0000000000000100

ALU result= 0000000000001000, z=0, n=0
Time=40 | ALUOut=0000000000001000 | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=0 | OverflowFlag=x | NegativeFlag=0

-----------------Write Back Cycle-----------------

Registers3=0000000000001000
```

Figure 11:ADDI Instructor

➢ ANDI Instructor:

```
*Welcome to our processor*


-----------------Reset Program-----------------

Time=0 | ALUOut=xxxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

-----------------Reset Program-----------------


-----------------Fetch Cycle-----------------

PC=0000000000000000
Instruction = 0010101010000010


-----------------Decode Cycle-----------------

Time: 30 | Opcode: 0010 | Function: 010 | State: 100 | IRWrite: 1 | RegDstB: 1 | pcwriteUNcond: 1 | RegDstC: 0 | ExtOP:1 | RegWr: 1 | ALUSrcA: 1 | ALUSrcB: 1 | ALUOp: 000 | MemRd: 0 | MemWr: 0 | WBData: 0
RegSucre2=0000000000000010


ReggDst: 101, outA: 0000000000000010, outB: 0000000000000101

-----------------Execute Cycle-----------------

Extended immediate= 2, BusA= 2, BusB= 5

Alusurce = 000

a= 0000000000000010, b=0000000000000010

ALU result= 0000000000000010, z=0, n=0
Time=40 | ALUOut=0000000000000010 | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=0 | OverflowFlag=x | NegativeFlag=0

-----------------Write Back Cycle-----------------

Registers5=0000000000000010
```

Figure 12:ANDI Instructor

➢ LW Instructor:

```
*Welcome to our processor*


------------------Reset Program------------------

Time=0 | ALUOut=xxxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

------------------Reset Program------------------


------------------Fetch Cycle------------------

PC=0000000000000000
Instruction = 0100100010000001


------------------Decode Cycle------------------

Time: 30 | Opcode: 0100 | Function: 001 | State: 101 | IRWrite: 1 | RegDstB: 1 | pcwriteUNcond: 1 | RegDstC: 0 | ExtOP:0 | RegWr: 1 | ALUSrcA: 1 | ALUSrcB: 1 | ALUOp: 001 | MemRd: 1 | MemWr: 0 | WBData: 1
RegSucre2=0000000000000010


ReggDst: 100, outA: 0000000000000010, outB: 0000000000000100

------------------Execute Cycle------------------

Extended immediate= 1, BusA= 2, BusB= 4

Alusurce = 001

a= 0000000000000010, b=0000000000000001

ALU result= 0000000000000011, z=0, n=0
Time=40 | ALUOut=0000000000000011 | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=0 | OverflowFlag=x | NegativeFlag=0

------------------Memory Cycle------------------

dataIn:     4, dataOut:  4369, address:     3
MemR: 1, MemW: 0, Memory[3]:  4369

Time=50 | ALUOut=0000000000000011 | DataOut=0001000100010001 | ZeroFlag=0 | OverflowFlag=x | NegativeFlag=0

------------------Write Back Cycle------------------

Registers4=0001000100010001
```

Figure 13:LW Instructor

➢ SW Instructor:

```
*Welcome to our processor*


------------------Reset Program------------------

Time=0 | ALUOut=xxxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

------------------Reset Program------------------


------------------Fetch Cycle------------------

PC=0000000000000000
Instruction = 0101100010000001


------------------Decode Cycle------------------

Time: 30 | Opcode: 0101 | Function: 001 | State: 100 | IRWrite: 1 | RegDstB: 1 | pcwriteUNcond: 1 | RegDstC: x | ExtOP:0 | RegWr: 0 | ALUSrcA: 1 | ALUSrcB: 1 | ALUOp: 001 | MemRd: 0 | MemWr: 1 | WBData: x
RegSucre2=0000000000000010


ReggDst: 100, outA: 0000000000000010, outB: 0000000000000100

------------------Execute Cycle------------------

Extended immediate= 1, BusA= 2, BusB= 4

Alusurce = 001

a= 0000000000000010, b=0000000000000001

ALU result= 0000000000000011, z=0, n=0
Time=40 | ALUOut=0000000000000011 | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=0 | OverflowFlag=x | NegativeFlag=0

------------------Memory Cycle------------------

Memeory=0000000000000100

dataIn:     4, dataOut:     x, address:     3
MemR: 0, MemW: 1, Memory[3]:     4
```

Figure 14:SW Instructor

➢ BEQ Instructor:

```
*Welcome to our processor*

------------------Reset Program------------------

Time=0 | ALUOut=xxxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

------------------Reset Program------------------

------------------Fetch Cycle------------------

PC=0000000000000000
Instruction = 0110010010000101

------------------Decode Cycle------------------

Time: 30 | Opcode: 0110 | Function: 101 | State: 011 | IRWrite: 1 | RegDstB: 1 | pcwriteUNcond: 0 | RegDstC: x | ExtOP:1 | RegWr: 0 | ALUSrcA: 1 | ALUSrcB: 0 | ALUOp: 010 | MemRd: 0 | MemWr: 0 | WBData: x
RegSucre2=0000000000000010

ReggDst: 010, outA: 0000000000000010, outB: 0000000000000010

------------------Execute Cycle------------------

Extended immediate= 5, BusA= 2, BusB= 2

Alusurce = 010

a= 0000000000000010, b=0000000000000010

ALU result= 0000000000000000, z=1, n=0
Time=40 | ALUOut=0000000000000000 | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=1 | OverflowFlag=x | NegativeFlag=0

------------------Fetch Cycle------------------

PC=0000000000000101
Instruction = xxxxxxxxxxxxxxx
```

Figure 15:BEQ Instructor

➢ BNE Instructor:

```
*Welcome to our processor*

------------------Reset Program------------------

Time=0 | ALUOut=xxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

------------------Reset Program------------------

------------------Fetch Cycle------------------

PC=0000000000000000
Instruction = 0111010101000010

------------------Decode Cycle------------------

Time: 30 | Opcode: 0111 | Function: 010 | State: 011 | IRWrite: 1 | RegDstB: 1 | pcwriteUNcond: 0 | RegDstC: x | ExtOP:1 | RegWr: 0 | ALUSrcA: 1 | ALUSrcB: 1 | ALUOp: 010 | MemRd: 0 | MemWr: 0 | WBData: x
RegSucre5=0000000000000101

ReggDst: 010, outA: 0000000000000101, outB: 0000000000000010

------------------Execute Cycle------------------

Extended immediate= 2, BusA= 5, BusB= 2

Alusurce = 010

a= 0000000000000101, b=0000000000000010

ALU result= 0000000000000011, z=0, n=0
Time=40 | ALUOut=0000000000000011 | DataOut=xxxxxxxxxxxxxxx | ZeroFlag=0 | OverflowFlag=x | NegativeFlag=0

------------------Fetch Cycle------------------

PC=0000000000000010
Instruction = xxxxxxxxxxxxxxx
```

Figure 16:BNE Instructor

➢ JMP Instructor:

```
*Welcome to our processor*

------------------Reset Program------------------

Time=0 | ALUOut=xxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

------------------Reset Program------------------

------------------Fetch Cycle------------------

PC=0000000000000000
Instruction = 0001111111111000

------------------Decode Cycle------------------

Time: 30 | Opcode: 0001 | Function: 000 | State: 010 | IRWrite: 1 | RegDstB: x | pcwriteUNcond: 0 | RegDstC: x | ExtOP:x | RegWr: 0 | ALUSrcA: x | ALUSrcB: x | ALUOp: xxx | MemRd: 0 | MemWr: 0 | WBData: x
RegSucre7=0000000000000000

ReggDst: 111, outA: 0000000000000000, outB: 0000000000000000

------------------Fetch Cycle------------------

PC=0000000111111111
Instruction = xxxxxxxxxxxxxxx
```

Figure 17:JMP Instructor

➢ CALL Instructor:

```
*Welcome to our processor*

------------------Reset Program------------------

Time=0 | ALUOut=xxxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

------------------Reset Program------------------

------------------Fetch Cycle------------------

PC=0000000000000000
Instruction = 0001111111111001

------------------Decode Cycle------------------

Time: 30 | Opcode: 0001 | Function: 001 | State: 010 | IRWrite: 1 | RegDstB: x | pcwriteUNcond: 0 | RegDstC: x | ExtOP:x | RegWr: 0 | ALUSrcA: x | ALUSrcB: x | ALUOp: xxx | MemRd: 0 | MemWr: 0 | WBData: x
RegSucre7=0000000000000000

ReggDst: 111, outA: 0000000000000000, outB: 0000000000000000

------------------Fetch Cycle------------------

PC=0000000111111111
Instruction = xxxxxxxxxxxxxxx
```

Figure 18:CALL Instructor

➢ AND Instructor:

```
*Welcome to our processor*

------------------Reset Program------------------

Time=0 | ALUOut=xxxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

------------------Reset Program------------------

------------------Fetch Cycle------------------

PC=0000000000000000
Instruction = 0000001010101000

------------------Decode Cycle------------------

Time: 30 | Opcode: 0000 | Function: 000 | State: 100 | IRWrite: 1 | RegDstB: 0 | pcwriteUNcond: 1 | RegDstC: 1 | ExtOP:x | RegWr: 1 | ALUSrcA: 1 | ALUSrcB: 0 | ALUOp: 000 | MemRd: 0 | MemWr: 0 | WBData: 0
RegSucre2=0000000000000010

ReggDst: 001, outA: 0000000000000010, outB: 0000000000000101

------------------Execute Cycle------------------

Extended immediate=65512, BusA= 2, BusB= 5

Alusurce = 000

a= 0000000000000010, b=0000000000000101

ALU result= 0000000000000000, z=1, n=0
Time=40 | ALUOut=0000000000000000 | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=1 | OverflowFlag=x | NegativeFlag=0

------------------Write Back Cycle------------------

Registers1=0000000000000000

------------------Fetch Cycle------------------

PC=0000000000000001
Instruction = 0001111111111010
```

Figure 19:AND Instructor

➢ ADD Instructor:

```
*Welcome to our processor*

------------------Reset Program------------------

Time=0 | ALUOut=xxxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

------------------Reset Program------------------

------------------Fetch Cycle------------------

PC=0000000000000000
Instruction = 0000101011101001

------------------Decode Cycle------------------

Time: 30 | Opcode: 0000 | Function: 001 | State: 100 | IRWrite: 1 | RegDstB: 0 | pcwriteUNcond: 1 | RegDstC: 1 | ExtOP:x | RegWr: 1 | ALUSrcA: 1 | ALUSrcB: 0 | ALUOp: 001 | MemRd: 0 | MemWr: 0 | WBData: 0
RegSucre3=0000000000000011

ReggDst: 101, outA: 0000000000000011, outB: 0000000000000101

------------------Execute Cycle------------------

Extended immediate=65513, BusA= 3, BusB= 5

Alusurce = 001

a= 0000000000000011, b=0000000000000101

ALU result= 0000000000001000, z=0, n=0
Time=40 | ALUOut=0000000000001000 | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=0 | OverflowFlag=x | NegativeFlag=0

------------------Write Back Cycle------------------

Registers5=0000000000001000

------------------Fetch Cycle------------------

PC=0000000000000001
Instruction = 0001111111111010
```

Figure 20:ADD Instructor

➢ SUB Instructor:

```
*Welcome to our processor*

------------------Reset Program------------------

Time=0 | ALUOut=xxxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

------------------Reset Program------------------

------------------Fetch Cycle------------------

PC=0000000000000000
Instruction = 0000101101011010

------------------Decode Cycle------------------

Time: 30 | Opcode: 0000 | Function: 010 | State: 100 | IRWrite: 1 | RegDstB: 0 | pcwriteUNcond: 1 | RegDstC: 1 | ExtOP:x | RegWr: 1 | ALUSrcA: 1 | ALUSrcB: 0 | ALUOp: 010 | MemRd: 0 | MemWr: 0 | WBData: 0
RegSucre5=0000000000000101

ReggDst: 101, outA: 0000000000000101, outB: 0000000000000011

------------------Execute Cycle------------------

Extended immediate=26, BusA= 5, BusB= 3

Alusurce = 010

a= 0000000000000101, b=0000000000000011

ALU result= 0000000000000010, z=0, n=0
Time=40 | ALUOut=0000000000000010 | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=0 | OverflowFlag=x | NegativeFlag=0

------------------Write Back Cycle------------------

Registers5=0000000000000010

------------------Fetch Cycle------------------

PC=0000000000000001
Instruction = 0001111111111010
```

Figure 21:SUB Instructor

➢ SLL Instructor:

```
*Welcome to our processor*

------------------Reset Program------------------

Time=0 | ALUOut=xxxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

------------------Reset Program------------------

------------------Fetch Cycle------------------

PC=0000000000000000
Instruction = 0000101010011011

------------------Decode Cycle------------------

Time: 30 | Opcode: 0000 | Function: 011 | State: 100 | IRWrite: 1 | RegDstB: 0 | pcwriteUNcond: 1 | RegDstC: 1 | ExtOP:x | RegWr: 1 | ALUSrcA: 1 | ALUSrcB: 0 | ALUOp: 011 | MemRd: 0 | MemWr: 0 | WBData: 0
RegSucre2=0000000000000010

ReggDst: 101, outA: 0000000000000010, outB: 0000000000000011

------------------Execute Cycle------------------

Extended immediate=27, BusA= 2, BusB= 3

Alusurce = 011

a= 0000000000000010, b=0000000000000011

ALU result= 0000000000010000, z=0, n=0
Time=40 | ALUOut=0000000000010000 | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=0 | OverflowFlag=x | NegativeFlag=0

------------------Write Back Cycle------------------

Registers5=0000000000010000

------------------Fetch Cycle------------------

PC=0000000000000001
Instruction = 0001111111111010
```

Figure 22:SLL Instructor

➢ SLL Instructor:

```
*Welcome to our processor*

------------------Reset Program------------------

Time=0 | ALUOut=xxxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

------------------Reset Program------------------

------------------Fetch Cycle------------------

PC=0000000000000000
Instruction = 0000101010011011

------------------Decode Cycle------------------

Time: 30 | Opcode: 0000 | Function: 011 | State: 100 | IRWrite: 1 | RegDstB: 0 | pcwriteUNcond: 1 | RegDstC: 1 | ExtOP:x | RegWr: 1 | ALUSrcA: 1 | ALUSrcB: 0 | ALUOp: 011 | MemRd: 0 | MemWr: 0 | WBData: 0
RegSucre2=0000000000000010

ReggDst: 101, outA: 0000000000000010, outB: 0000000000000011

------------------Execute Cycle------------------

Extended immediate=27, BusA= 2, BusB= 3

Alusurce = 011

a= 0000000000000010, b=0000000000000011

ALU result= 0000000000010000, z=0, n=0
Time=40 | ALUOut=0000000000010000 | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=0 | OverflowFlag=x | NegativeFlag=0

------------------Write Back Cycle------------------

Registers5=0000000000010000

------------------Fetch Cycle------------------

PC=0000000000000001
Instruction = 0001111111111010
```

Figure 23:SLL Instructor

➢ SRL Instructor:

```
*Welcome to our processor*

------------------Reset Program------------------

Time=0 | ALUOut=xxxxxxxxxxxxxxxx | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=x | OverflowFlag=x | NegativeFlag=x

------------------Reset Program------------------

------------------Fetch Cycle------------------

PC=0000000000000000
Instruction = 0000101110010100

------------------Decode Cycle------------------

Time: 30 | Opcode: 0000 | Function: 100 | State: 100 | IRWrite: 1 | RegDstB: 0 | pcwriteUNcond: 1 | RegDstC: 1 | ExtOP:x | RegWr: 1 | ALUSrcA: 1 | ALUSrcB: 0 | ALUOp: 100 | MemRd: 0 | MemWr: 0 | WBData: 0
RegSucre6=0000000000000110

ReggDst: 101, outA: 0000000000000110, outB: 0000000000000010

------------------Execute Cycle------------------

Extended immediate=20, BusA= 6, BusB= 2

Alusurce = 100

a= 0000000000000110, b=0000000000000010

ALU result= 0000000000000001, z=0, n=0
Time=40 | ALUOut=0000000000000001 | DataOut=xxxxxxxxxxxxxxxx | ZeroFlag=0 | OverflowFlag=x | NegativeFlag=0

------------------Write Back Cycle------------------

Registers5=0000000000000001

------------------Fetch Cycle------------------

PC=0000000000000001
Instruction = 0001111111111010
```

Figure 24:SRL Instructor