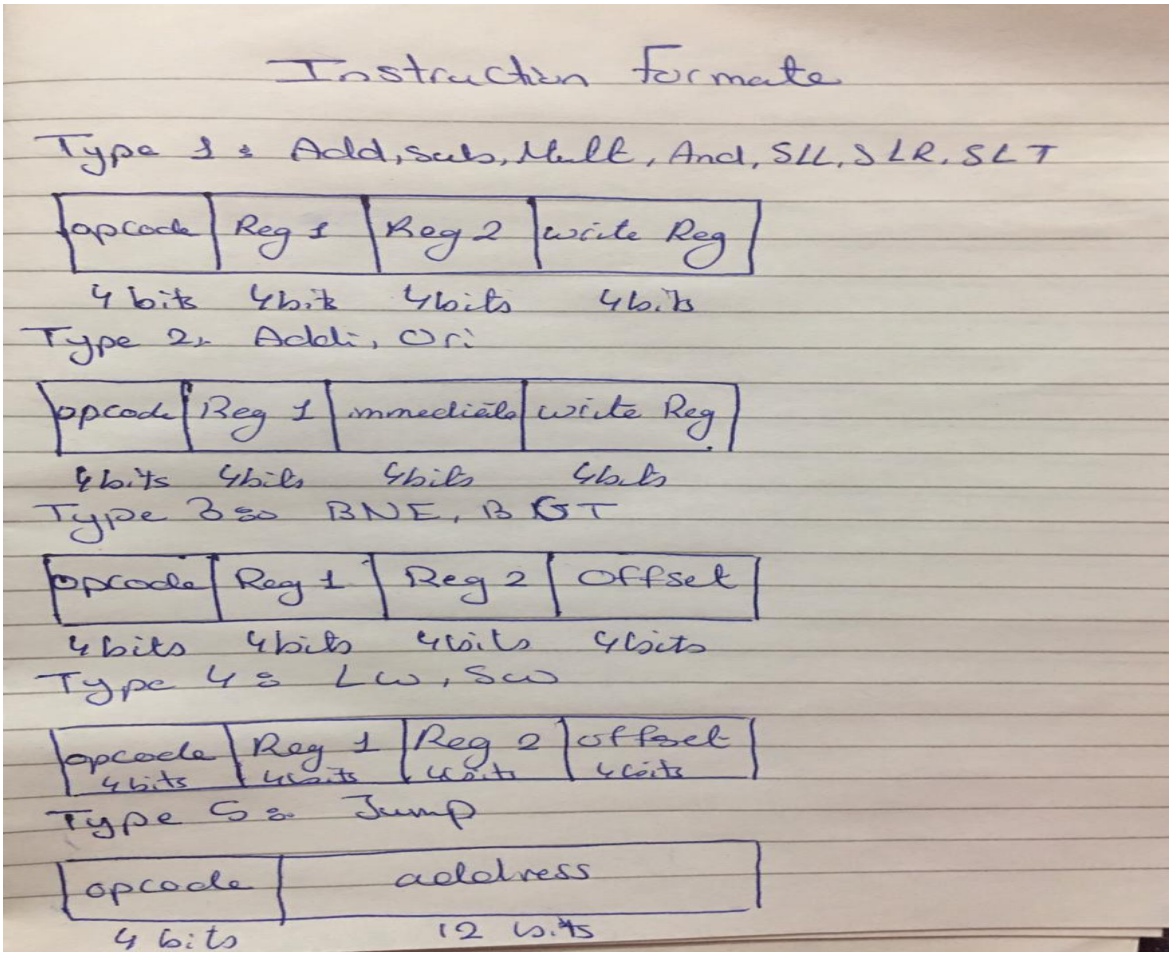
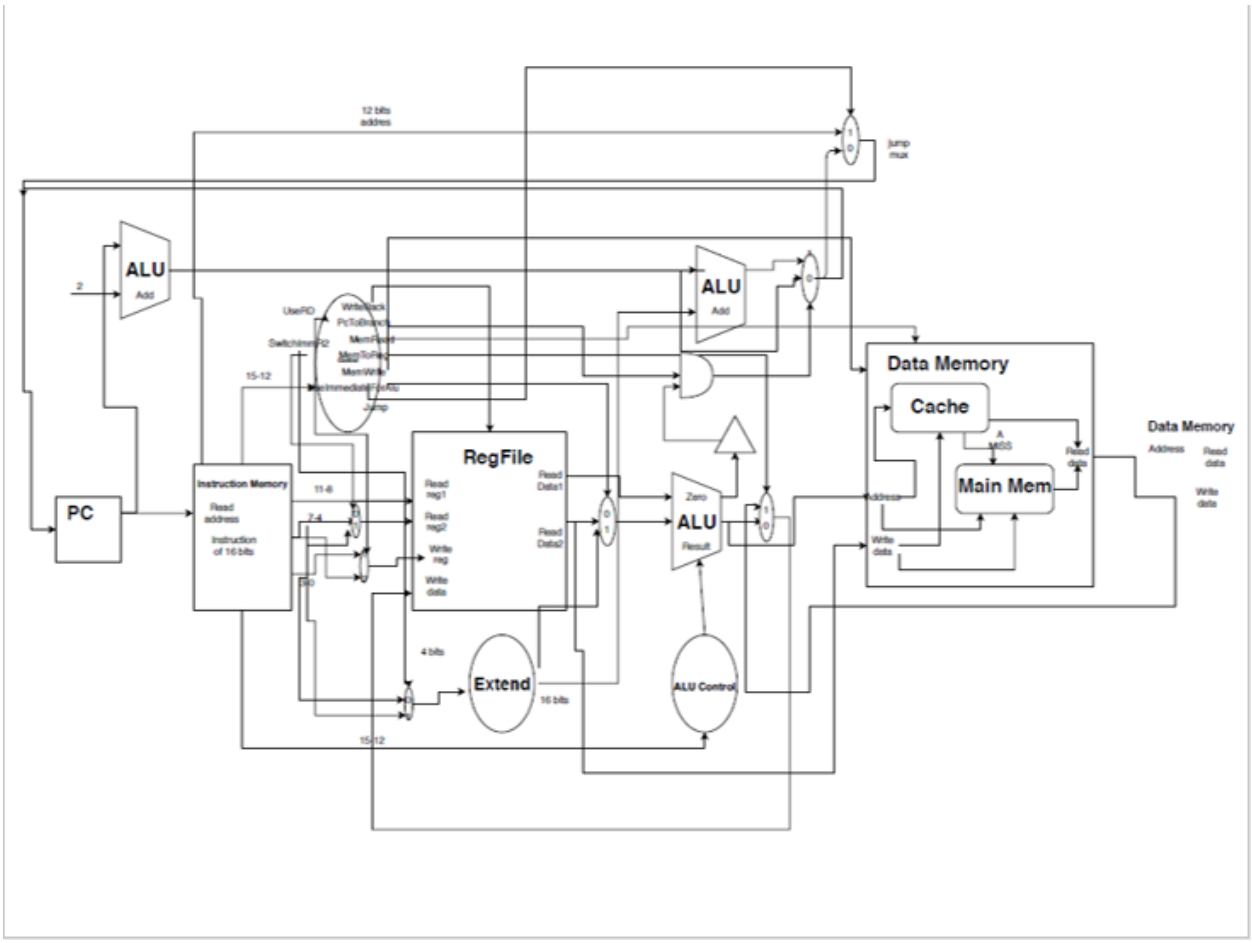
Instruction Format

- Since instruction and data memory are 16 bits, the memory size is 16 bits which is reflected on the size of the instruction.
- For all operations opcode is 4 bits, since we have 14 instructions to be covered which is well covered in 4 bits, each instruction could have a unique opcode.
- A unique format is set for each instruction type.
- For memory accessing in case of Load/Store, add the 2nd Register value to the immediate.



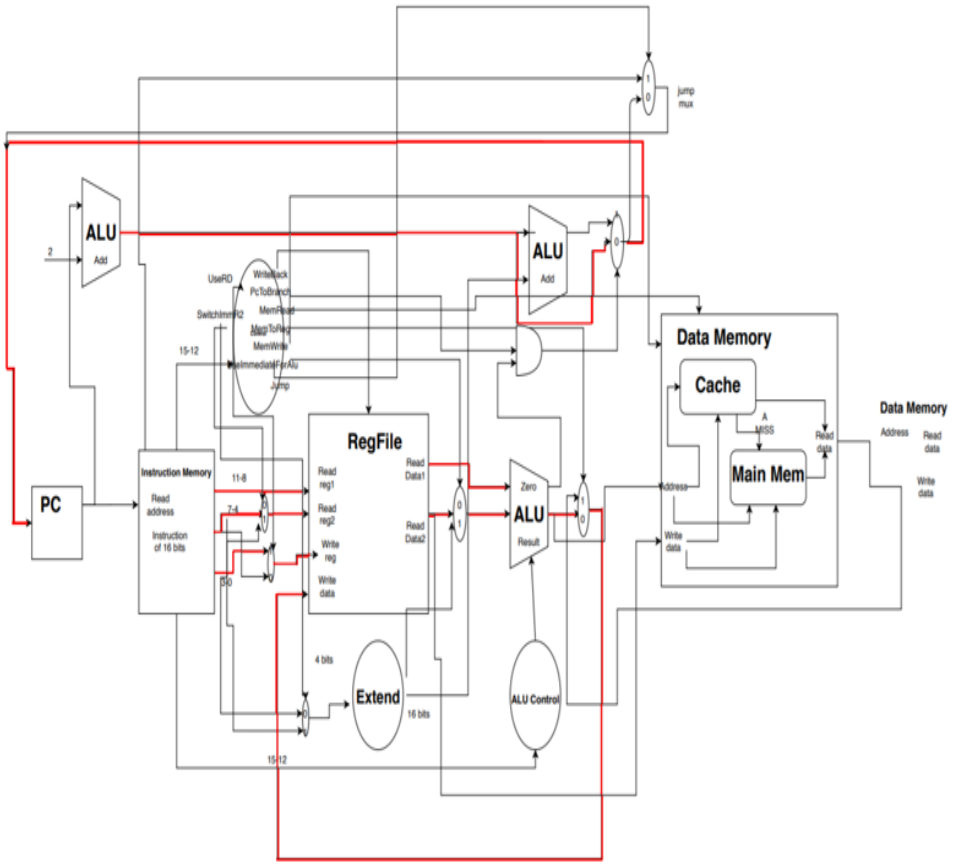
Branch Instructions

Opcode	Register 1	Register 2	offset


12 bits address
jump mux
ALU
Add
2
ALU
Add
Data Memory
Cache
Main Mem
Data Memory
Address
Read data
Write data
PC
Instruction Memory
Read address
Instruction of 16 bits
RegFile
Read reg1
Read reg2
Write reg
Write data
Read Data1
Read Data2
Zero
ALU
Result
Extend
4 bits
16 bits
ALU Control
15-12
11-8
7-4
3-0
UseRD
SwitchImmReg
MemRead
MemWrite
Jump

Shifting

15-12	11-8	7-4	3-0
OP Code 0100: SLL 0101: SRL	Register 1	Register 2	Destination Register

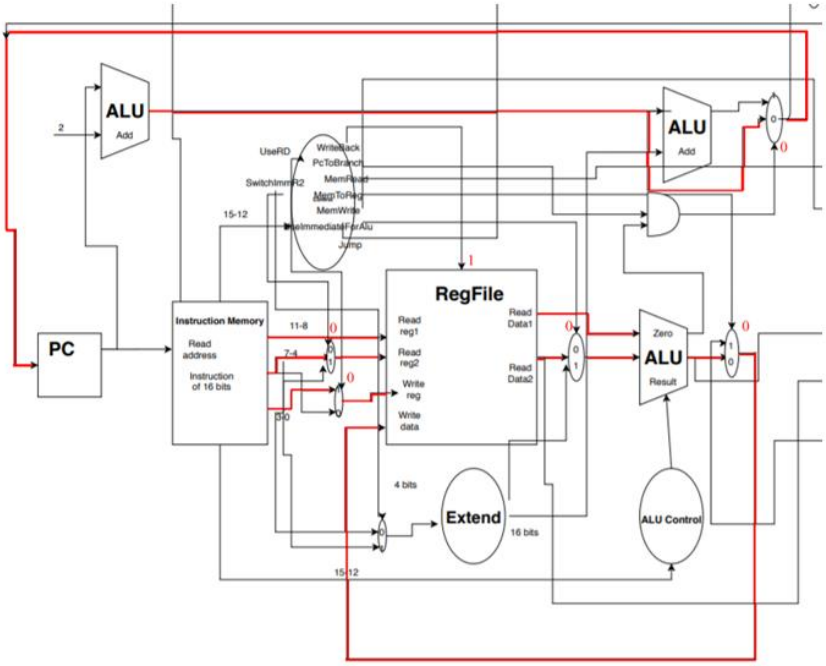


Control Signals:

WriteBack	UseImmediateForALU	PCSrc	MemRead	MemWrite	SwitchImmR2	Jump
1	0	0	0	0	0	0

ALUControl:

```
switch(opcode) {
case("0000") : bit4 = "0010";break;
case("0001") : bit4 = "0110";break;
case("0010") : bit4 = "0011";break;
case("0011") : bit4 = "0000";break;
case("0100") : bit4 = "0100";break;
case("0101") : bit4 = "0101";break;
case("0110") : bit4 = "0111";break;
case("0111") : bit4 = "0010";break;
case("1000") : bit4 = "0001";break;
case("1001") : bit4 = "0010";break;
case("1010") : bit4 = "0010";break;
case("1011") : bit4 = "0110";break;
case("1100") : bit4 = "0110";break;
case("1101") : bit4 = "0010";break;
}
```



```java
public String SHL(String operand1 , String operand2) {

    int op1 = Integer.parseInt(operand1,2);
    int op2 = Integer.parseInt(operand2,2);

    System.out.println("Operation Name : ShiftLeft");
    System.out.println("1st operand : " + signExtend((operand1)) + "/" +Integer.parseInt(operand1,2));
    System.out.println("1st operand : " + signExtend((operand2)) + "/" +Integer.parseInt(operand2,2));
    int result = op1 * 2^op2;
    if(result ==0) {
        this.Z = true;
    }
    System.out.println( "Output: " + signExtend(Integer.toBinaryString(result))   + "/" + result);
    System.out.println("Z Flag : " + this.Z);
    this.output = Integer.toBinaryString(result);
    return(Integer.toBinaryString(result));



}
```