Instruction_type	Reg_write	Imm_src	Alu_src	Mem_write	Result_src	brunch	Alu_op	jump	jalr
lw	1	000	1	0	01	0	00	0	0
sw	0	001	1	1	00	0	00	0	0
R_type	1	xxx	0	0	00	0	10	0	0
B_type	0	010	0	0	xx	1	10	0	0
I_type	1	000	1	0	00	0	10	0	0
Jalr	1	000	1	0	10	0	00	1	1
jal	1	011	x	0	10	0	xx	1	0
lui	1	100	x	0	11	0	xx	0	0

```
RtypeSub = funct7b5 & opb5; // TRUE for R-type subtract
```

```
case (ALUOp)
  2'b00: {ALUControl, JumpType} = 5'b000_00; // addition
  2'b01: {ALUControl, JumpType} = 5'b001_00; // subtraction
  default: case (funct3) // R-type or I-type ALU
    3'b000: if(Branch)
              {ALUControl, JumpType} = 5'b001_00; // beq
            else if (RtypeSub)
              {ALUControl, JumpType} = 5'b001_00; // sub
            else
              {ALUControl, JumpType} = 5'b000_00; // add, addi
    3'b001: {ALUControl, JumpType} = 5'b001_01; // bne
    3'b010: {ALUControl, JumpType} = 5'b101_00; // slt, slti
    3'b100: {ALUControl, JumpType} = 5'b101_10; // blt
    3'b101: {ALUControl, JumpType} = 5'b101_11; // bge
    3'b110: {ALUControl, JumpType} = 5'b011_00; // or, ori
    3'b111: {ALUControl, JumpType} = 5'b010_00; // and, andi
```

```
// Checking jump condition is true or not
JumpCondition = (JumpType==`BEQ)?
  Zero : (JumpType==`BNE)?
  ~Zero :(JumpType==`BLT)?
  SltOut : ~SltOut; // BGE
```

```
// Calculate PCSrc and assign value
PCSrc = Jalr? (2'b10) :
  ((Branch & JumpCondition) | Jump)? (2'b01): (2'b00);
```