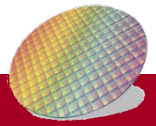
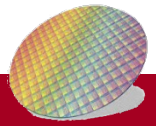
CO Homework 5 CPU



Overview

- Implement CPU module
- Make sure your codes can execute 17 RISC-V instructions in this homework
- Verify your CPU by using Modelsim



Instruction Format

- R-type

31 25	24 20	19 15	14 12	11 7	6 0		
funct7	rs2	rs1	funct3	rd	opcode	Mnemonic	Description
0000000	rs2	rs1	000	rd	0110011	ADD	rd = rs1 + rs2
0100000	rs2	rs1	000	rd	0110011	SUB	rd = rs1 - rs2
0000000	rs2	rs1	100	rd	0110011	XOR	rd = rs1 ^ rs2
0000000	rs2	rs1	110	rd	0110011	OR	rd = rs1 | rs2
0000000	rs2	rs1	111	rd	0110011	AND	rd = rs1 & rs2

Instruction Format

- I-type

31 20	19 15	14 12	11 7	6 0		
imm[11:0]	rs1	funct3	rd	opcode	Mnemonic	Description
imm[11:0]	rs1	010	rd	0000011	LW	rd = M[rs1 + imm]
imm[11:0]	rs1	000	rd	0010011	ADDI	rd = rs1 + imm
imm[11:0]	rs1	100	rd	0010011	XORI	rd = rs1 ^ imm
imm[11:0]	rs1	110	rd	0010011	ORI	rd = rs1 | imm
imm[11:0]	rs1	111	rd	0010011	ANDI	rd = rs1 & imm
imm[11:0]	rs1	000	rd	1100111	JALR	rd = PC + 4 PC = imm + rs1 (Set LSB of PC to 0)

Instruction Format

- S-type

31 25	24 20	19 15	14 12	11 7	6 0		
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	Mnemonic	Description
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW	M[rs1 + imm] = rs2

- B-type

31 25	24 20	19 15	14 12	11 7	6 0		
imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode	Mnemonic	Description
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ	PC = (rs1 == rs2) ? PC + imm : PC + 4
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE	PC = (rs1 != rs2) ? PC + imm : PC + 4

Instruction Format

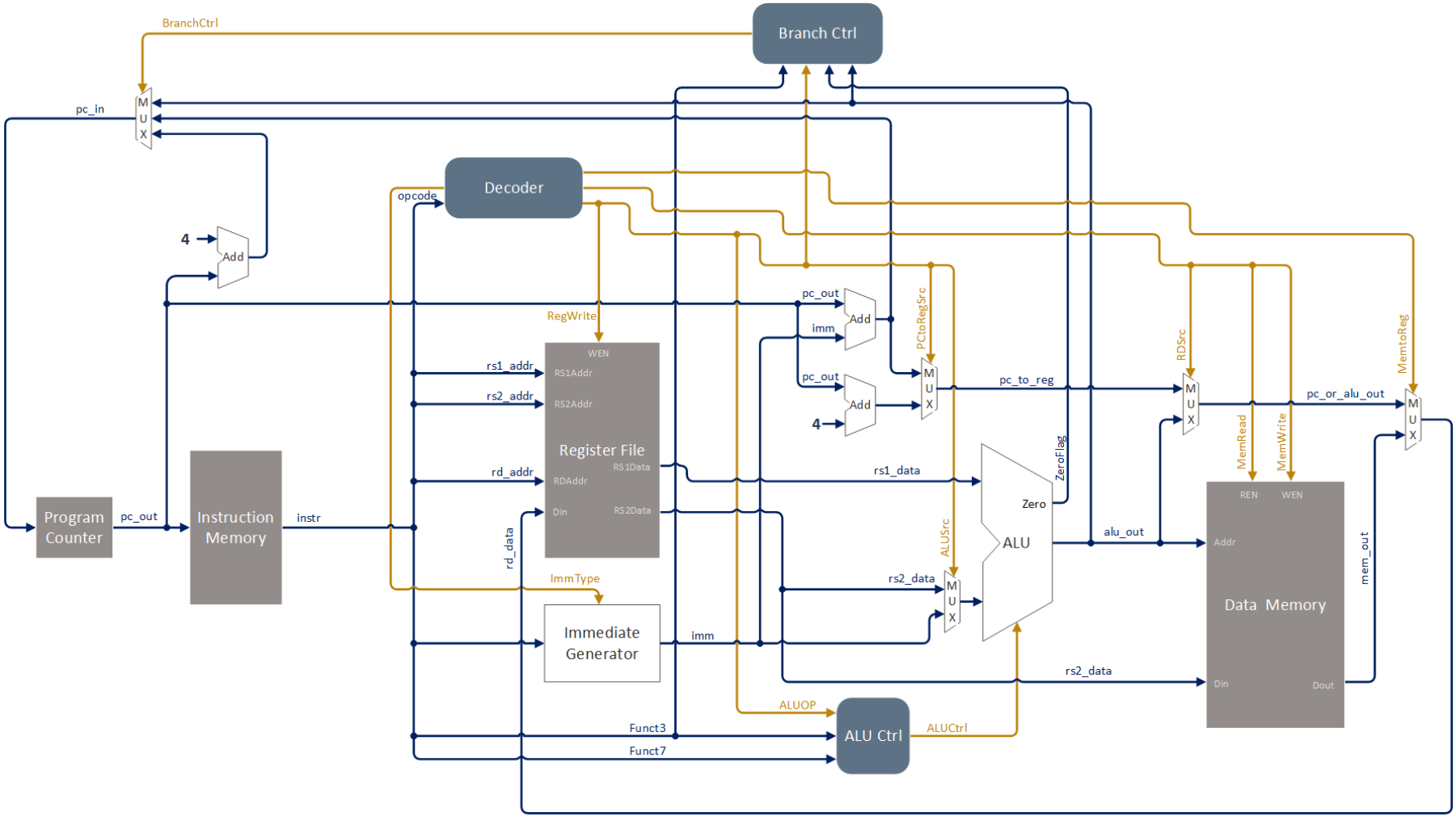
- U-type

31 12	11 7	6 0		
imm[31:12]	rd	opcode	Mnemonic	Description
imm[31:12]	rd	0010111	AUIPC	rd = PC + imm
imm[31:12]	rd	0110111	LUI	rd = imm

- J-type

31 12	11 7	6 0		
imm[20|10:1|11|19:12]	rd	opcode	Mnemonic	Description
imm[20|10:1|11|19:12]	rd	1101111	JAL	rd = PC + 4 PC = PC + imm

Block Diagram



Example

```
////////////////////////////////////////////////
//Immediate
always@(posedge clk or posedge rst)begin
    if(rst)
        Immediate <= 32'h0;
    else if(Instruction_Decode)begin
        case(opcode)
            7'b0000011:begin//LW
                /*add your code*/
            end
            7'b0010011:begin//I-type
                /*add your code*/
            end
            7'b1100111:begin//JALR-type
                /*add your code*/
            end
            7'b0100011:begin//S-type
                /*add your code*/
            end
            7'b1100011:begin//B-type
                /*add your code*/
            end
            7'b0010111:begin//AUIPC
                /*add your code*/
            end
            7'b0110111:begin//LUI
                Immediate[31:12] <= instr_out[31:12];
                Immediate[11:0] <= 12'h0;
            end
            7'b1101111:begin//J-type
                /*add your code*/
            end
        endcase
    end
end
////////////////////////////////////////////////
```

Example

```
//////////////////////////////////////////////
//Register_Files
always@(posedge clk or posedge rst)begin
    if(rst)begin
        for(i = 0; i < 32; i = i + 1)
            Register[i] <= 32'h0;
    end
    else if(Write_Back)begin
        case(opcode)
            7'b0110011:begin//R-type
                case(funct3)
                    3'b000:begin
                        case(funct7)
                            7'b0000000://ADD
                                Register[rd] <= Register[rs1] + Register[rs2];
                            7'b0100000:begin//SUB
                                /*add your code*/
                            end
                        endcase
                    end
                    3'b001:begin
                        case(funct7)
                            7'b0000000://SLL
                                Register[rd] <= Register[rs1] << Register[rs2][4:0];
                        endcase
                    end
                    3'b100:begin
                        case(funct7)
                            7'b0000000:begin//XOR
                                /*add your code*/
                            end
                        endcase
                    end
                    3'b110:begin
                        case(funct7)
                            7'b0000000:begin//OR
                                /*add your code*/
                            end
                        endcase
                    end
                    3'b111:begin
                        case(funct7)
                            7'b0000000:begin//AND
                                /*add your code*/
                            end
                        endcase
                    end
                endcase
            end
```

Example

```
            7'b0000011:begin
                case(funct3)
                    3'b010:begin//LW
                        Register[rd] <= data_out;
                    end
                endcase
            end
            7'b0010011:begin//I-type
                case(funct3)
                    3'b000:begin//ADDI
                        Register[rd] <= Register[rs1] + Immediate;
                    end
                    3'b100:begin//XORI
                        /*add your code*/
                    end
                    3'b110:begin//ORI
                        /*add your code*/
                    end
                    3'b111:begin//ANDI
                        /*add your code*/
                    end
                endcase
            end
            7'b1100111:begin//JALR
                case(funct3)
                    3'b000:begin
                        /*add your code*/
                    end
                endcase
            end
            7'b0010111:begin//AUIPC
                /*add your code*/
            end
            7'b0110111:begin//LUI
                /*add your code*/
            end
            7'b1101111:begin//J-type
                /*add your code*/
            end
        endcase
    end
end
//////////////////////////////////////////////
```

Example

```
//////////////////////////////////////////////
//instr_addr == PC
always@(posedge clk or posedge rst)begin
    if(rst)
        instr_addr <= 0;
    else if(Write_Back)begin
        case(opcode)
            7'b1100111:begin
                case(funct3)
                    3'b000:begin//JALR
                        /*add your code*/
                    end
                endcase
            end
            7'b1100011:begin//B-type
                case(funct3)
                    3'b000:begin//BEQ
                        /*add your code*/
                    end
                    3'b001:begin//BNE
                        /*add your code*/
                    end
                    3'b111:begin
                        if(Register[rs1] >= Register[rs2])
                            instr_addr <= instr_addr + Immediate;
                        else
                            instr_addr <= instr_addr + 4;
                    end
                endcase
            end
            7'b1101111:begin//JAL-type
                /*add your code*/
            end
            default:begin//default
                /*add your code*/
            end
        endcase
    end
end
//////////////////////////////////////////////
```

Example

```
////////////////////////////////////////////////
//data_addr
always@(posedge clk or posedge rst)begin
    if(rst)
        data_addr <= 32'h0;
    else if(Execute)begin
        case(opcode)
            7'b0000011:begin//L-type
                /*add your code*/
            end
            7'b0100011:begin//S-type
                /*add your code*/
            end
        endcase
    end
end
////////////////////////////////////////////////
```

Example

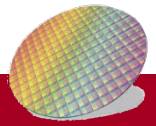
```
////////////////////////////////////////////////
//data_write
always@(posedge clk or posedge rst)begin
    if(rst)
        data_write <= 4'h0;
    else if(Execute)begin
        case(opcode)
            7'b0100011:begin
                case(funct3)
                    3'b010:begin//SW
                        data_write <= 4'hf;
                    end
                endcase
            end
        endcase
    end
    else if(Memory_Access)
        data_write <= 4'h0;
end
////////////////////////////////////////////////
```

Example

```
////////////////////////////////////////////////
//data_in
always@(posedge clk or posedge rst)begin
    if(rst)
        data_in <= 0;
    else if(Execute)begin
        case(opcode)
            7'b0100011:begin
                if(Register[rs1][1:0] + Immediate[1:0] == 2'b0)
                    data_in <= Register[rs2];
            end
        endcase
    end
end
////////////////////////////////////////////////
```

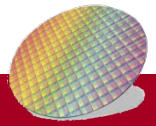
Homework Requirements

- Complete the CPU that can execute 17 instructions from the RISC-V ISA section
- Verify your CPU with the benchmark and take a snapshot
- Finish the Project Report



Score

- Your score is divided into two parts:
  - Functional Simulation (75%):
    - There are 15 test data, if you pass one of them, you will get 5 points
  - Report (25%):
    - take a screenshot of your result, and write your report in “report.docx”



Thanks for listening