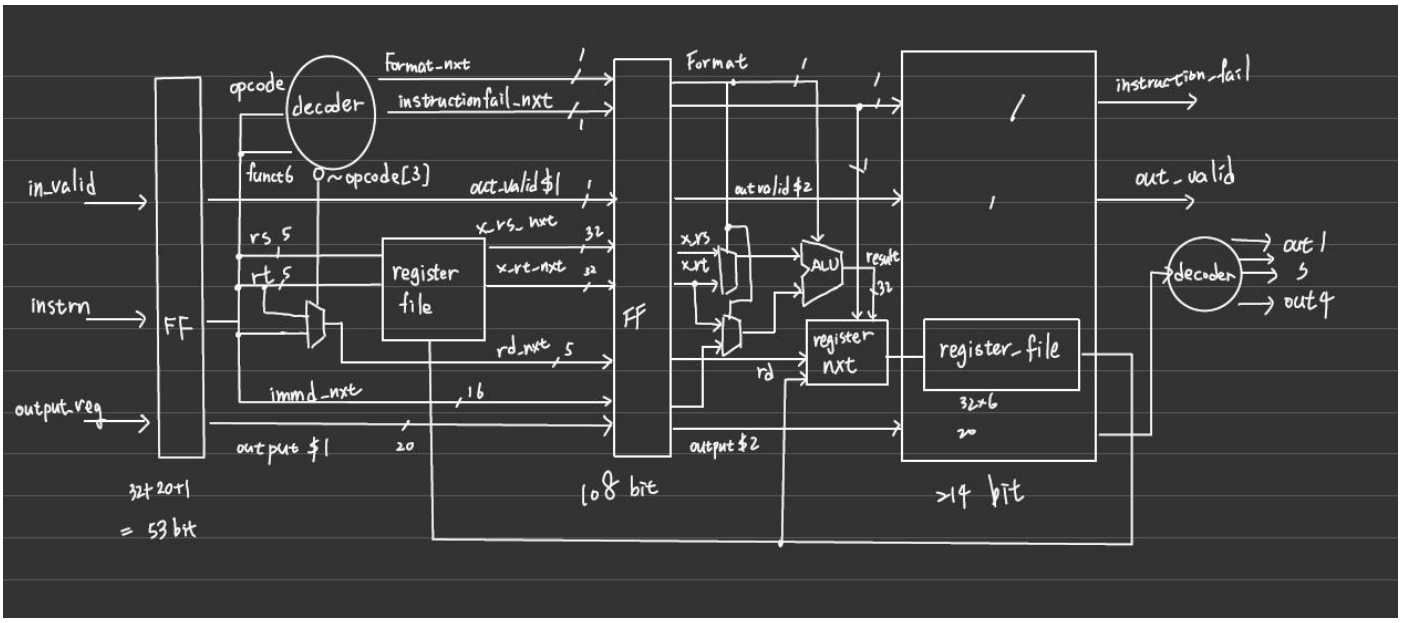
1. 這是這次的架構圖，參考助教給的 pdf 以及計組的講義，我把它畫了出來。



2. 這次的作業其實蠻簡單的，雖然因為是第一次做 pipeline 的設計，看到題目的時候花了一點時間思考該怎麼打扣才會跑出自己想要的 design。但想到了之後很順的就打了出來，這次大概計算了一下好像 3 小時就寫完了。
3. 自己有發現一個可能蠻直觀能夠避免自己在打的時候亂掉的方法，就是從第一級開始慢慢往後面打過去。所以我的扣才會分成下面圖中那樣

```
assign instruction_nxt = (in_valid == 1) ? instruction : 0;
assign output_reg$1_nxt = (in_valid == 1) ? output_reg : 0;
// level 1 : INPUT
// stage 1 : FF1
always_ff @(posedge clk or negedge rst_n) begin
    if(!rst_n) begin
        instrn_reg <= 0;
        output_reg$1 <= 0;
        out_valid$1 <= 0;
    end
    else begin
        instrn_reg <= instruction_nxt;
        output_reg$1 <= output_reg$1_nxt;
        out_valid$1 <= in_valid;
    end
end
```

```
// level 2 : Decode & Read register
// {Opcode, rs, rt, rd_nxt, Shamt_nxt, funct6_nxt} = instrn_reg
// {Opcode, rs, rt, immd_nxt} = instrn_reg
assign Opcode = instrn_reg[31:26];
assign rs = instrn_reg[25:21];
assign rt = instrn_reg[20:16];
assign rd_nxt = (Format_nxt == 0) ? rt : instrn_reg[15:11];
assign immd_nxt = instrn_reg[15:0];
// Format :
// 0 : illegal
// 1 : R format
// 2 : I format
assign Format_nxt = ~Opcode[3];
always_comb begin
    if(Opcode == 6'b000000) begin
        case(instrn_reg[5:0])
            6'b100000, 6'b100100, 6'b100101, 6'b100111, 6'b000000, 6'b000010: legal = 1;
            default: legal = 0;
        endcase
    end
    else if (Opcode == 6'b001000) begin
        legal = 1;
    end
    else begin
        legal = 0;
    end
end
```

```
// stage 2 : FF2
always_ff @(posedge clk or negedge rst_n) begin
    if(!rst_n) begin
        output_reg$2 <= 0;
        out_valid$2 <= 0;
        instruction_fail$1 <= 0;
        x_rs <= 0;
        x_rt <= 0;
        rd <= 0;
        immd <= 0;
        Format <= 0;
    end
    else begin
        output_reg$2 <= output_reg$1;
        out_valid$2 <= out_valid$1;
        instruction_fail$1 <= instruction_fail_nxt;
        x_rs <= x_rs_nxt;
        x_rt <= x_rt_nxt;
        rd <= rd_nxt;
        immd <= immd_nxt;
        Format <= Format_nxt;
    end
end
```

```
// level 3 : ALU Calculate
always_comb begin
    if(Format == 0) begin
        alu_result = x_rs + immd;
    end
        case(immd[5:0])
            6'b100000: alu_result = x_rs + x_rt;
            6'b100100: alu_result = x_rs & x_rt;
            6'b100101: alu_result = x_rs | x_rt;
            6'b100111: alu_result = ~(x_rs | x_rt);
            6'b000000: alu_result = x_rt << immd[10:6];
            6'b000010: alu_result = x_rt >> immd[10:6];
            default: alu_result = 0;
        endcase
    end
end
// if illegal, register is not changed.
// rd = rt if is I type
// rd = rd if is R type(is determined in previous stage)
always_comb begin
    if(instruction_fail$1 == 1) begin
        register_file_nxt = register_file;
    end
    else begin
        case(rd)
```

```
// stage 3 : FF and write back
always_ff @(posedge clk or negedge rst_n) begin
    if(!rst_n) begin
        register_file <= {0, 0, 0, 0, 0, 0};
        out_valid$3 <= 0;
        instruction_fail$2 <= 0;
    end
    else begin
        output_reg$3 <= output_reg$2;
        register_file <= register_file_nxt;
        out_valid$3 <= out_valid$2;
        instruction_fail$2 <= instruction_fail$1;
    end
end
```

```
// level 4 : OUTPUT select
assign instruction_fail = instruction_fail$2;
assign out_valid = out_valid$3;
    if(instruction_fail$2 == 1) begin
        out_1 = 0;
        out_2 = 0;
        out_3 = 0;
        out_4 = 0;
    end
    else begin
```