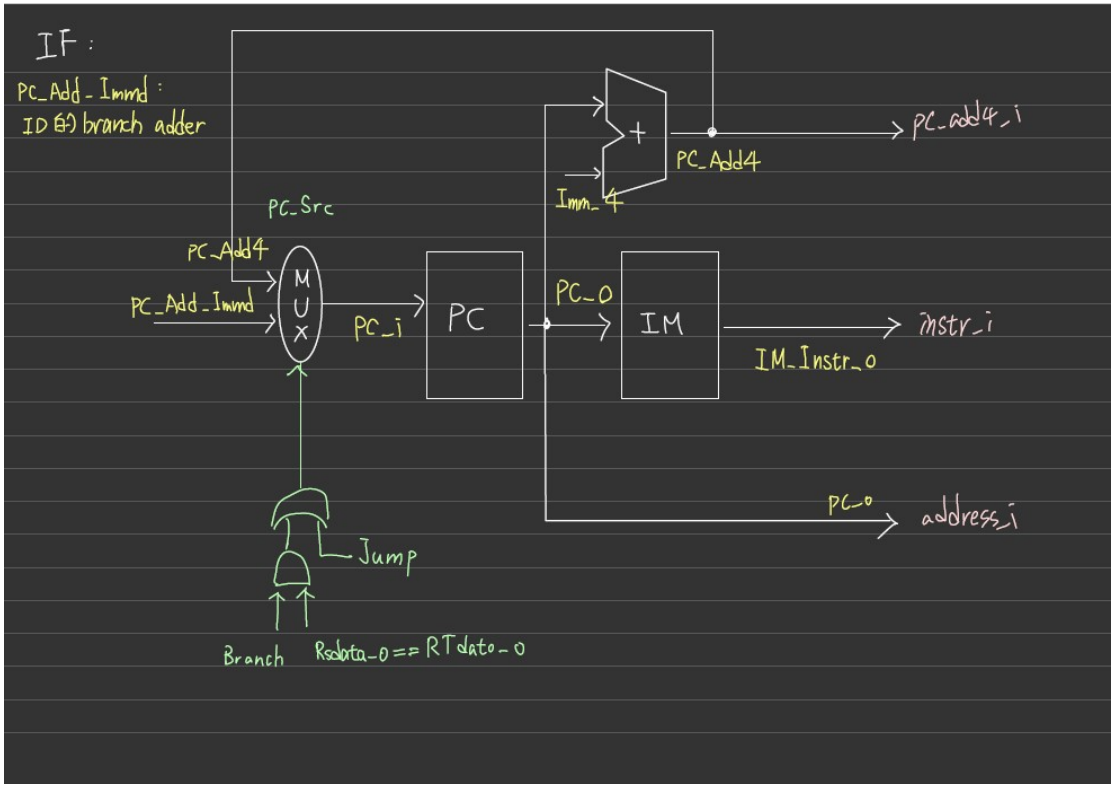
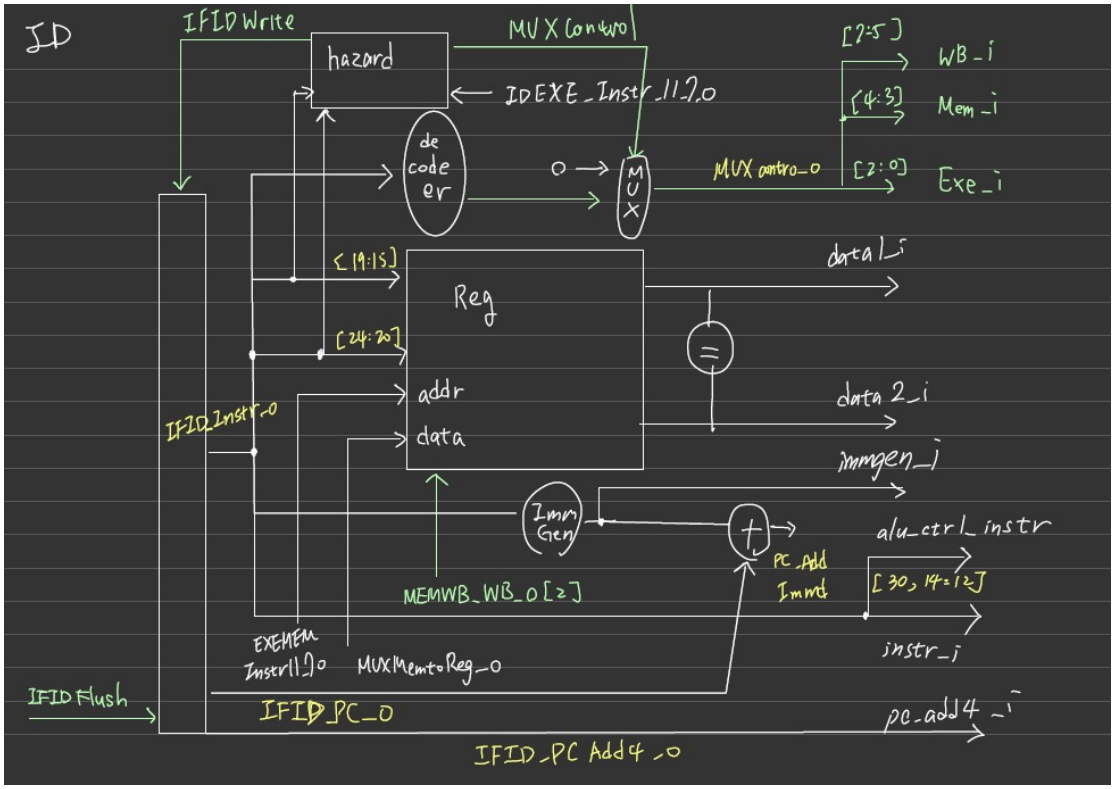
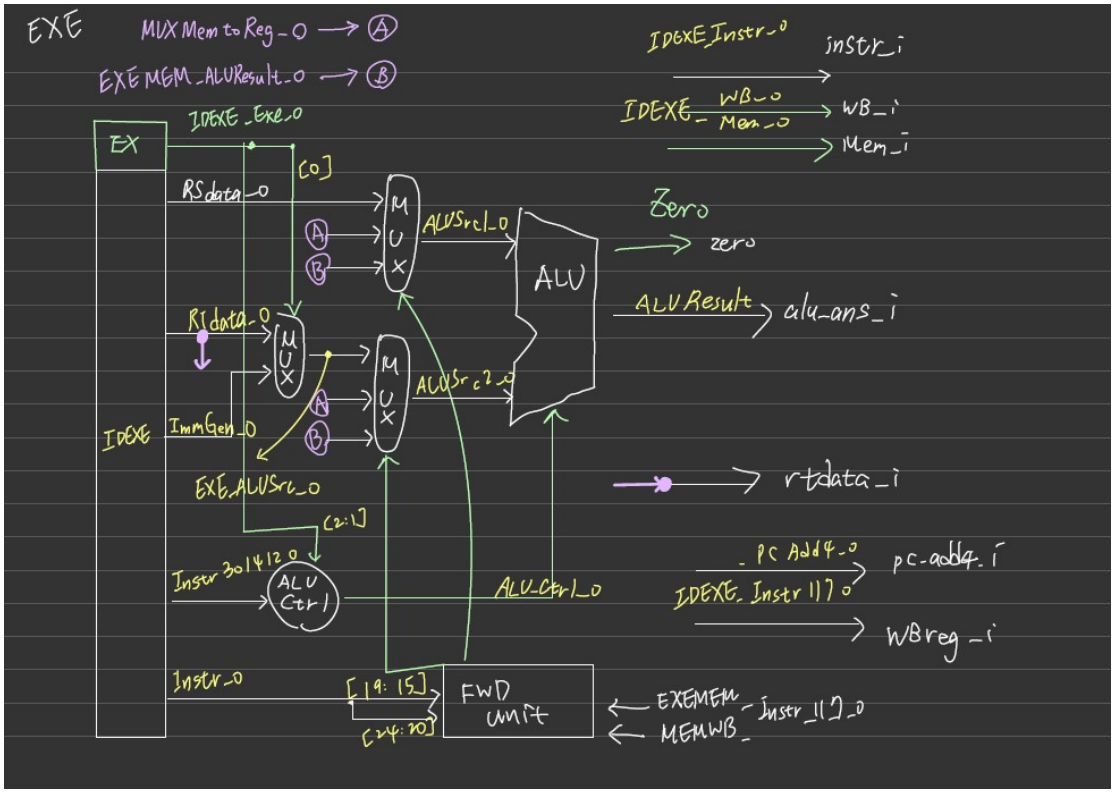
Block diagram:

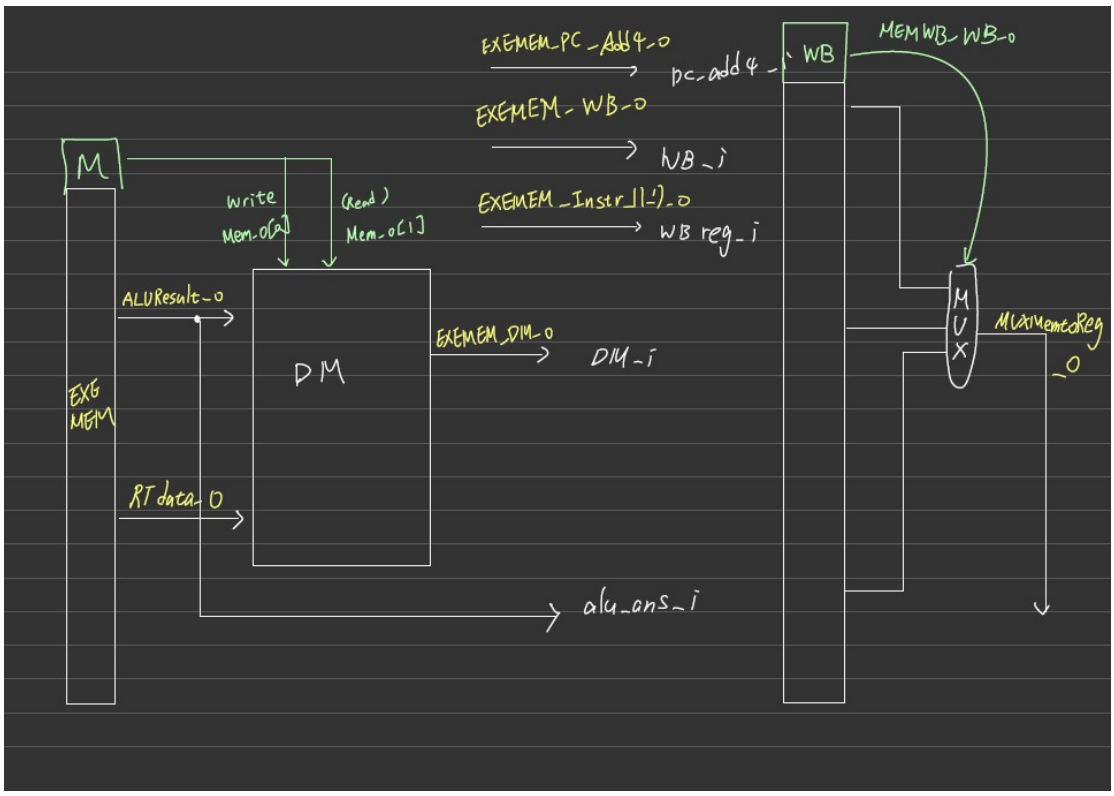
Since this Lab is quite complicated, I separated the block diagram of each parts.



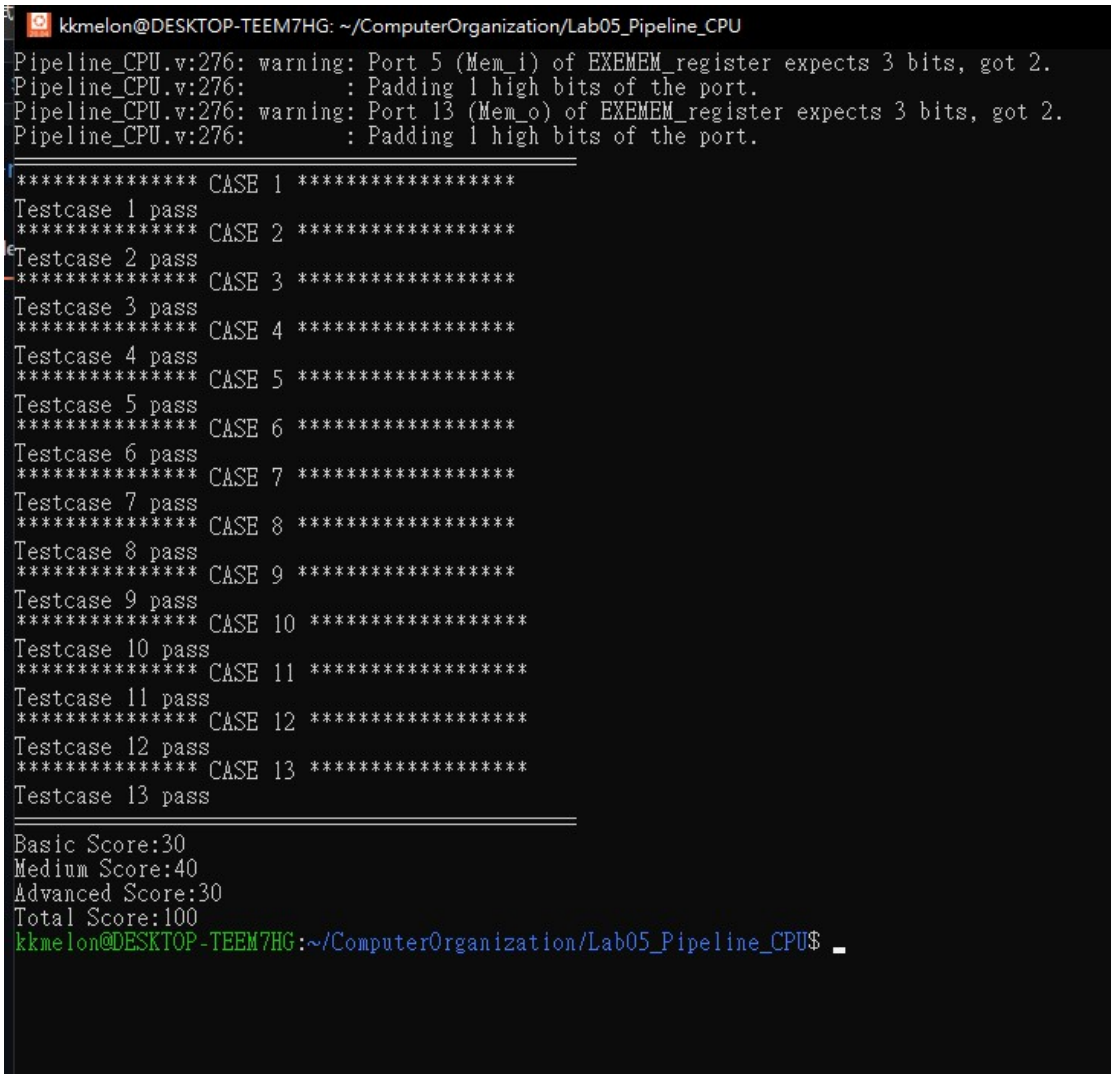



EXE
MUX Mem to Reg_0
EXE MEM_ALUResult_0
IDEXE_Exe_0
EX
RS data_0
RT data_0
IDEXE
Imm Gen_0
EXE_ALUSrc_0
[2:1]
Instr 30:14:12_0
ALU Ctrl
Instr_0
[19:15]
[24:20]
M U X
A
B
ALUSrc1_0
ALUSrc2_0
ALU
ALU_Ctrl_0
FWD unit
IDEXE_Instr_0
instr_i
IDEXE_WB_0
WB_i
Mem_0
Mem_i
Zero
zero
ALU Result
alu_ans_i
rtdata_i
PC Add4_0
pc_add4_i
IDEXE_Instr[11:7]_0
WB reg_i
EXEMEM_Instr[11:7]_0
MEMWB_

MEM and WB

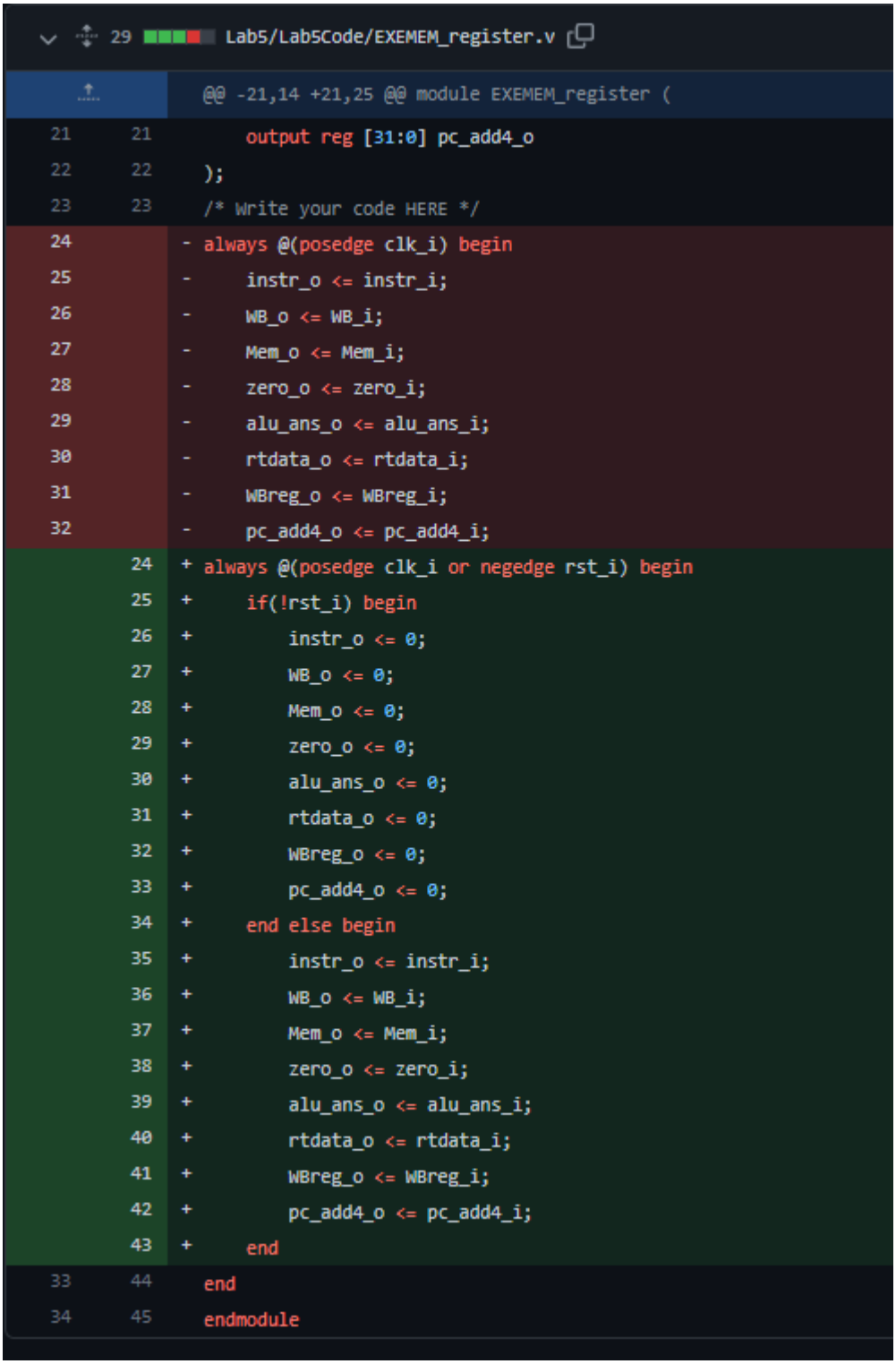

EXEMEM_PC_Add4_0
pc_add4_i
WB
MEMWB_WB_0
EXEMEM_WB_0
WB_i
EXEMEM_Instr[11:7]_0
WB reg_i
M
write Mem_0[2]
(Read) Mem_0[1]
ALUResult_0
DM
EXEMEM_DM_0
DM_i
EXE MEM
RT data_0
alu_ans_i
M U X
MemtoReg_0

Implementation results:



Problems encountered:

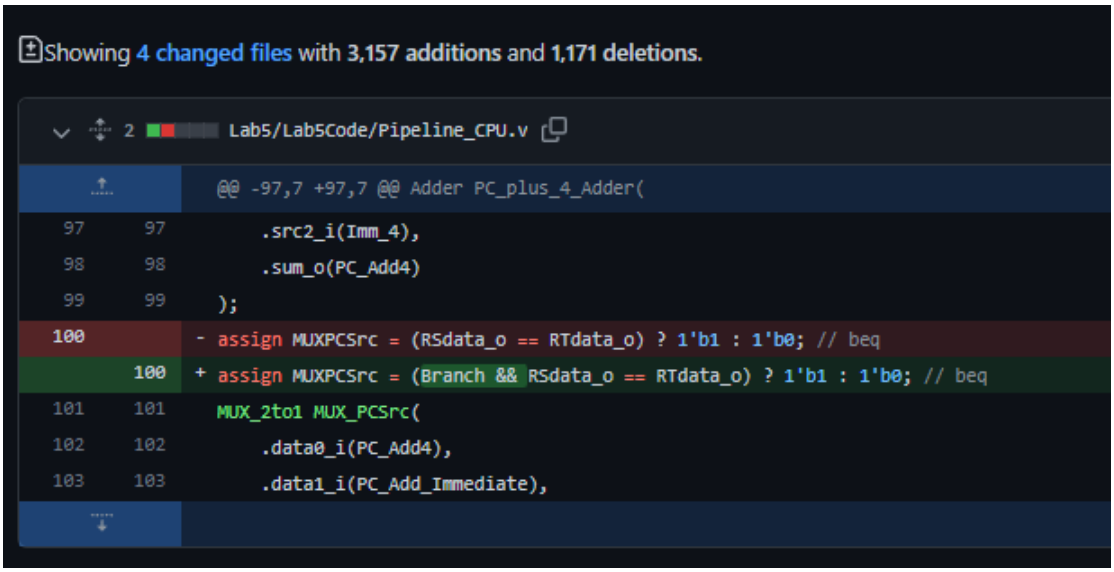
1. Register output should be reset. We didn't set the value to zero during initial reset, so our PC value is wrong initially.



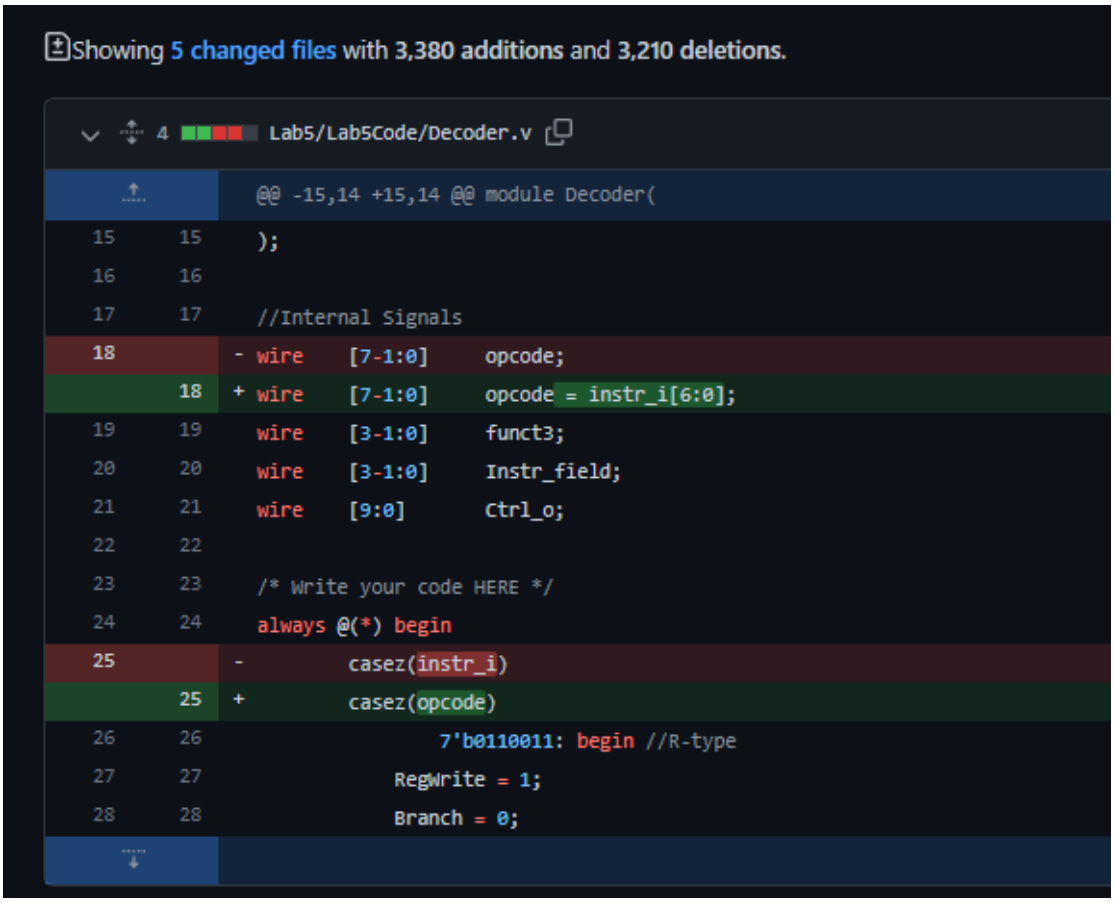
29 Lab5/Lab5Code/EXEMEM_register.v

```
@@ -21,14 +21,25 @@ module EXEMEM_register (
    output reg [31:0] pc_add4_o
);
/* Write your code HERE */
- always @(posedge clk_i) begin
-     instr_o <= instr_i;
-     WB_o <= WB_i;
-     Mem_o <= Mem_i;
-     zero_o <= zero_i;
-     alu_ans_o <= alu_ans_i;
-     rtdata_o <= rtdata_i;
-     WBreg_o <= WBreg_i;
-     pc_add4_o <= pc_add4_i;
+ always @(posedge clk_i or negedge rst_i) begin
+     if(!rst_i) begin
+         instr_o <= 0;
+         WB_o <= 0;
+         Mem_o <= 0;
+         zero_o <= 0;
+         alu_ans_o <= 0;
+         rtdata_o <= 0;
+         WBreg_o <= 0;
+         pc_add4_o <= 0;
+     end else begin
+         instr_o <= instr_i;
+         WB_o <= WB_i;
+         Mem_o <= Mem_i;
+         zero_o <= zero_i;
+         alu_ans_o <= alu_ans_i;
+         rtdata_o <= rtdata_i;
+         WBreg_o <= WBreg_i;
+         pc_add4_o <= pc_add4_i;
+     end
 end
endmodule
```

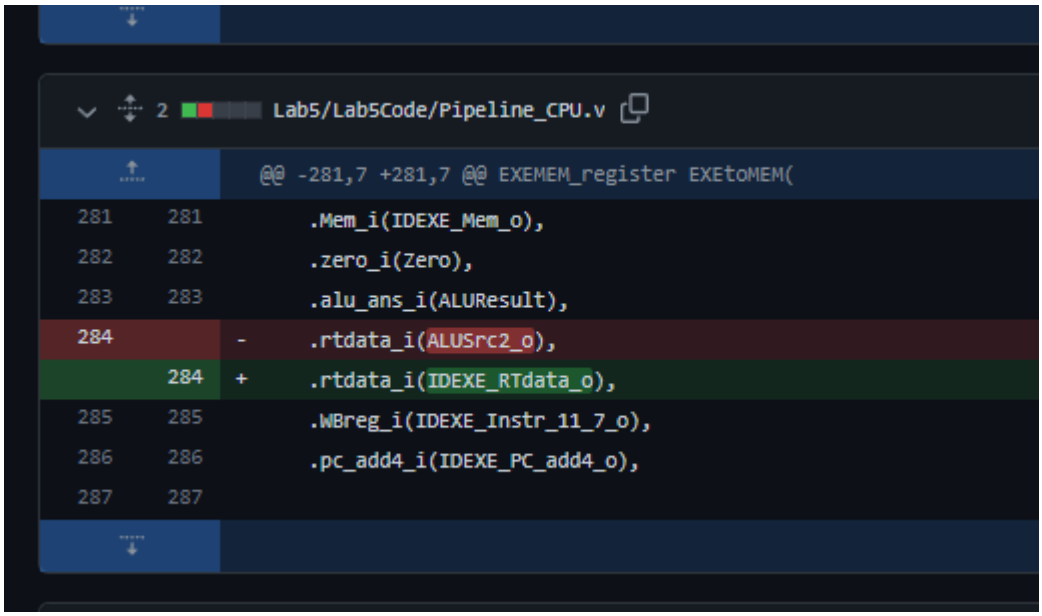
2. I didn’t put branch signal in the MUXPCSrc, so the PC value was wrong



3. Our opcode is not assigned to instr_i[6:0], so the decoder output was always xxx;



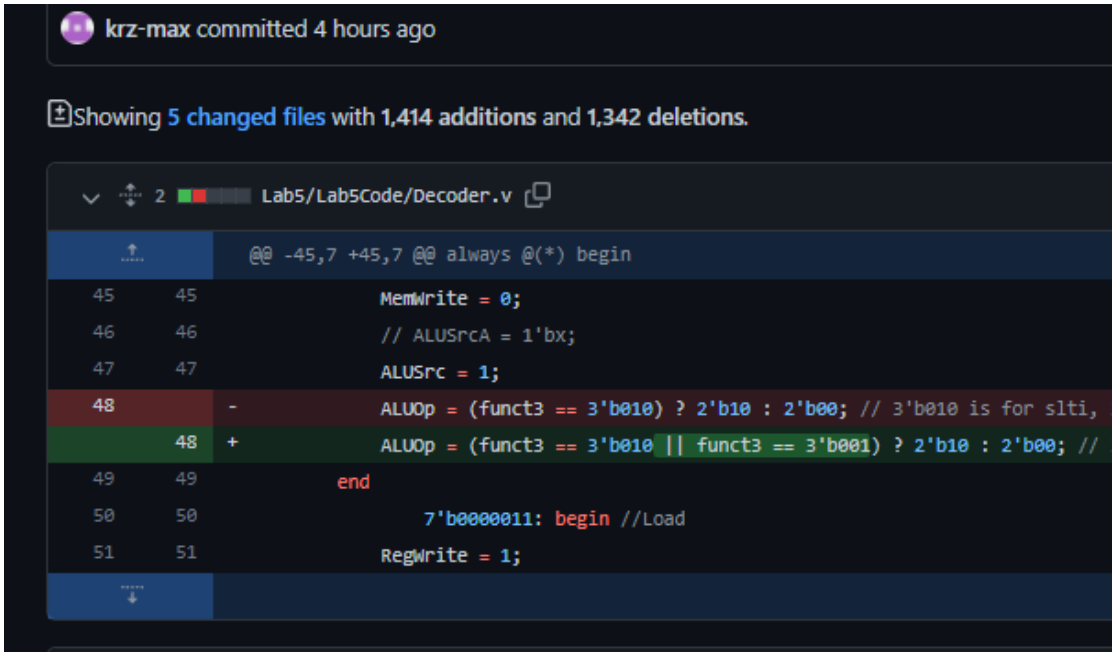
4. I fed the wrong value to EXEMEM_register initially, it should be rt_data



2 Lab5/Lab5Code/Pipeline_CPU.v

```
@@ -281,7 +281,7 @@ EXEMEM_register EXEtoMEM(
281 281      .Mem_i(IDEXE_Mem_o),
282 282      .zero_i(Zero),
283 283      .alu_ans_i(ALUResult),
284     -    .rtdata_i(ALUSrc2_o),
    284 +    .rtdata_i(IDEXE_RTdata_o),
285 285      .WBreg_i(IDEXE_Instr_11_7_o),
286 286      .pc_add4_i(IDEXE_PC_add4_o),
287 287
```

5. In some testcases, there are slti and slli instructions, but we didn't cover them in Lab4, so the decoder value was wrong.



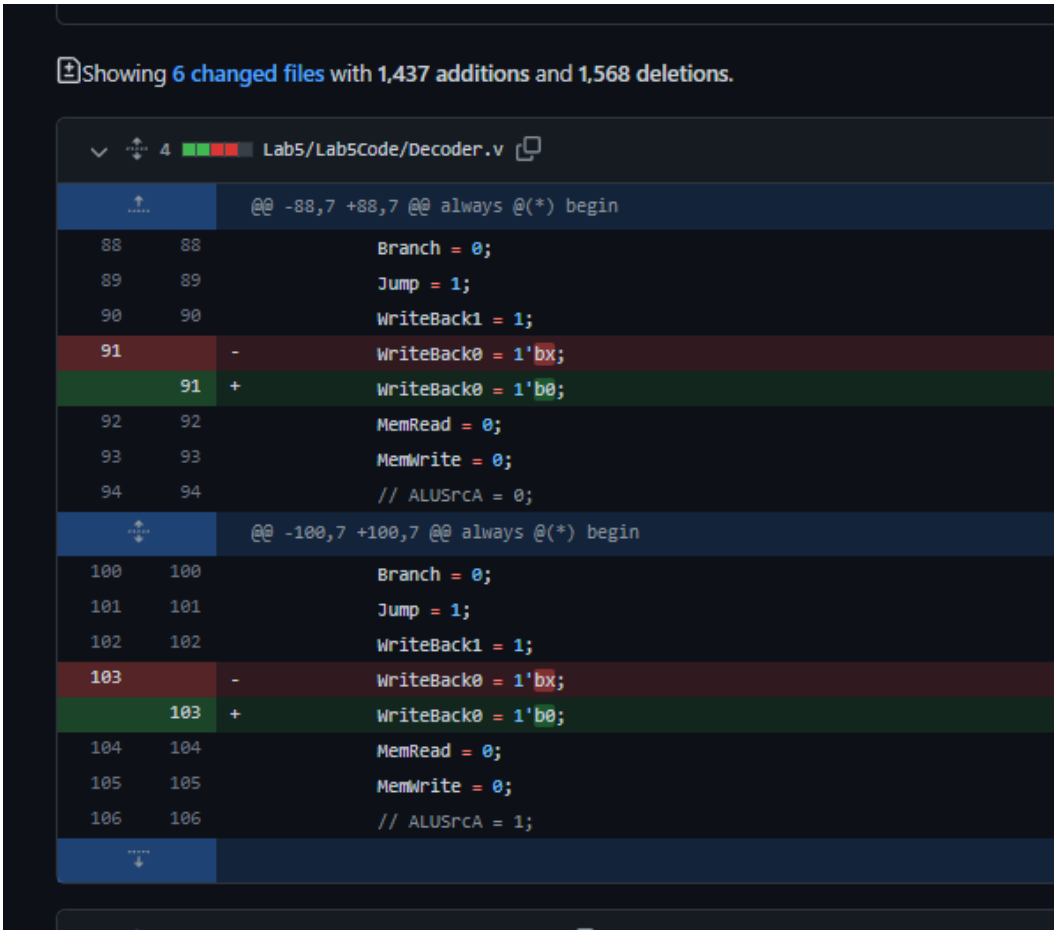
krz-max committed 4 hours ago

Showing 5 changed files with 1,414 additions and 1,342 deletions.

2 Lab5/Lab5Code/Decoder.v

```
@@ -45,7 +45,7 @@ always @(*) begin
45 45              MemWrite = 0;
46 46              // ALUSrcA = 1'bx;
47 47              ALUSrc = 1;
48     -           ALUOp = (funct3 == 3'b010) ? 2'b10 : 2'b00; // 3'b010 is for slti,
    48 +           ALUOp = (funct3 == 3'b010 || funct3 == 3'b001) ? 2'b10 : 2'b00; //
49 49          end
50 50              7'b0000011: begin //Load
51 51              RegWrite = 1;
```

6. Because in Lab4, the writeback MUX was divided into two 2to1 MUX, and we modified using a 3to1 MUX, so the control bit should not be don't care, or the output of the MUX will be xxx.



7. I am not sure why in there wasn't a shiftleft1 in Lab4, so I tried removing the unit in Lab5, after that, we passed all the testcases.