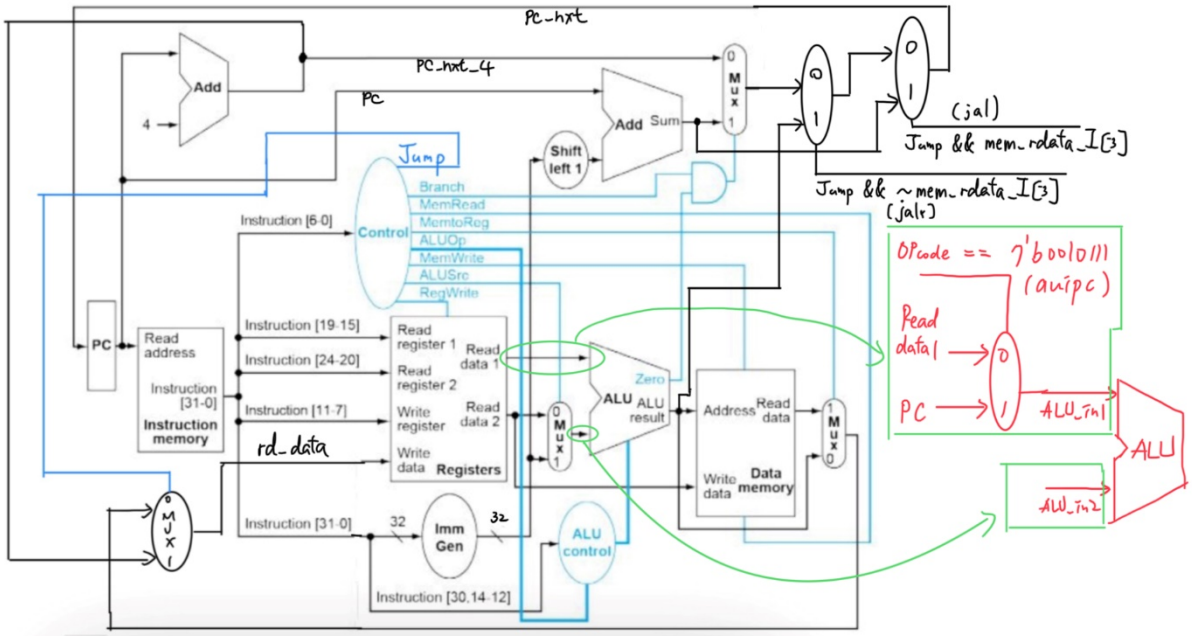
Computer Architecture Final Report

組員 ： 林楷崴 **b08505039** 工海四 / 劉旻鑫 **b08505049** 工海四

CPU Architecture



CHIP.v

```
63    assign  rs1 = mem_rdata_I[19:15];
64    assign  rs2 = mem_rdata_I[24:20];
65    assign  rd  = mem_rdata_I[11:7];
66    //==== Submodule Connection ==================
67    //------------------------------------//
68    // Do not modify this part!!!         //
69    reg_file reg0(                        //
70        .clk(clk),                        //
71        .rst_n(rst_n),                    //
72        .wen(regWrite),                   //
73        .a1(rs1),                         //
74        .a2(rs2),                         //
75        .aw(rd),                          //
76        .d(rd_data),                      //
77        .q1(rs1_data),                    //
78        .q2(rs2_data));                   //
79    //------------------------------------//
80    // Todo: other submodules
81    ImmdiateGeneration i1(
82        .in_imm(mem_rdata_I),
83        .out_imm(Immediate));
84
85    Control c1(
86        .OpCode(mem_rdata_I[6:0]),
87        .Branch_w(Branch),
88        .MemRead_w(MemRead),
89        .MemtoReg_w(MemtoReg),
90        .ALUOp_w(ALUOp),
91        .MemWrite_w(MemWrite),
92        .RegWrite_w(regWrite),
93        .ALUSrc_w(ALUSrc),
94        .Jump_w(Jump));
95
96    ALUControl a1(
97        .ALUOpt(ALUOp),
98        .Funct({mem_rdata_I[30],mem_rdata_I[14:12]}),
99        .ALUCtrl(ALUCntrol),
100       .MD(mem_rdata_I[25]));
101
102   ALU alu1(
103       .clk(clk),
104       .rst_n(rst_n),
105       .in1(ALU_in1),
106       .in2(ALU_in2),
107       .ALUCtrol(ALUCntrol),
108       .out(ALU_out),
109       .Fuct({mem_rdata_I[30],mem_rdata_I[14:12]}),
110       .zero_o(ALU_zero),
111       .ready_o(ALU_ready));
112
113   assign rd_data = ((Jump) ? PC_nxt_4 : ((MemtoReg) ? mem_rdata_D : ALU_out));
114   assign PC_nxt_4 = (PC + 32'd4);
115   assign PC_nxt_imm = (PC + (Immediate << 1));
116   assign ALU_in1 = ((mem_rdata_I[6:0] == 7'b0010111) ? PC : rs1_data);
117   assign ALU_in2 = (ALUSrc ? Immediate : rs2_data);
```

We use several submodules in CHIP.v to build our CPU architecture.

reg_file handles the registers.

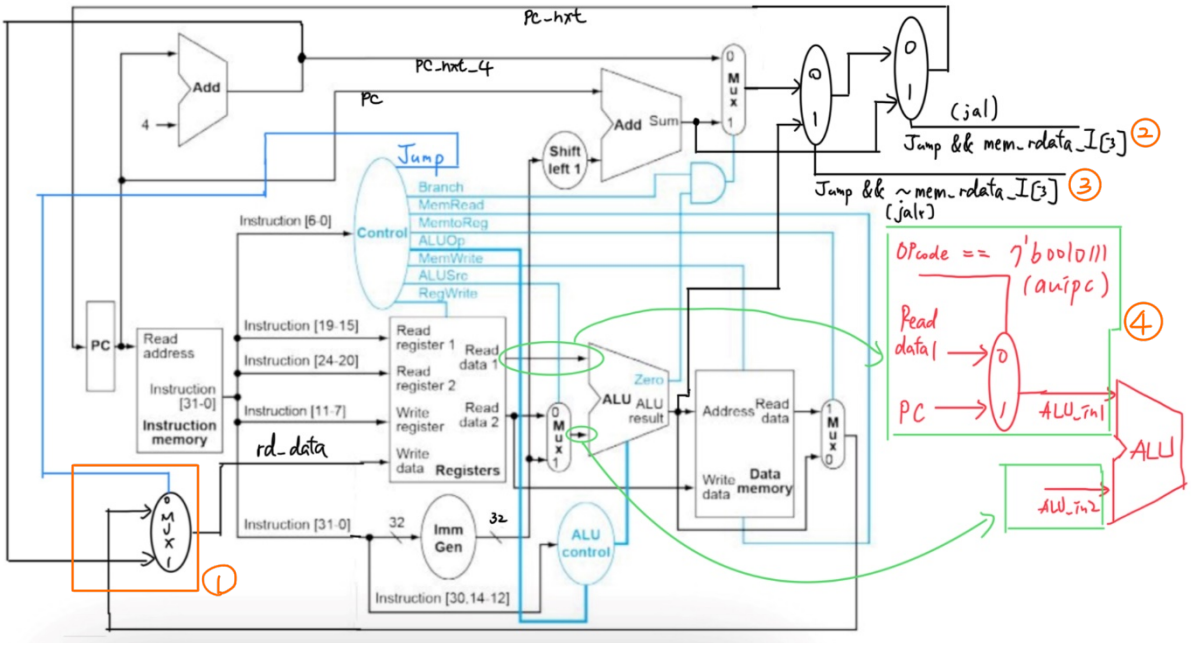
ImmdiateGeneration handles the immediate expansion from instruction to 32-bit immediate.

Control handles key control signals by determining different types of instructions.

ALUControl handles how ALU operates based on Function3 and ALUOp.

ALU handles all operations between ALU_in1 and ALU_in2. Besides, dealing with Branch operations and generating Zero if B-type instructions are satisfied.

Data Path of Instructions (jal, jalr, auipc)



jal : store address of (PC+4) to rd { (if Jump == 1) rd_data = PC_nxt_4 } ①
and PC jump to (PC+imm*2) { (if Jump && mem_rdata_I[3] == 1) PC_nxt = PC_nxt_imm } ②

jalr : store address of (PC+4) to rd { (if Jump == 1) rd_data = PC_nxt_4 } ①
and PC jump to (rs1+imm) {(if Jump&&~mem_rdata_I[3] == 1) PC_nxt = ALU_out } ③

auipc : store address of (PC+imm) to rd { (if auipc occur) ALU_in1 = PC } ④

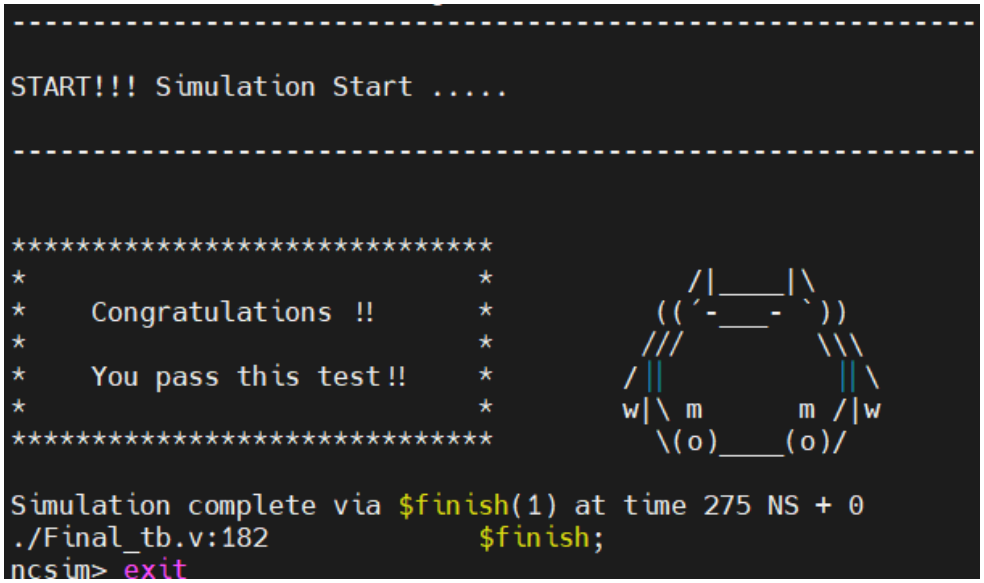
Handle multi-cycle instruction (mul)

```
118   //==== Combinational Part ====================
119   // Todo: any combinational/sequential circuit
120   always @(*)
121   begin
122       case(state)
123           Standard:
124           begin
125               if((ALUCntrol == 4'b0011) || (ALUCntrol == 4'b0100))
126               begin
127                   state_nxt = MulDiv;
128                   PC_nxt <= PC;
129               end
130               else
131               begin
132                   state_nxt = state;
133                   PC_nxt <= (Jump && mem_rdata_I[3])? PC_nxt_imm : ((Jump && ~mem_rdata_I[3]) ? ALU_out : ((Branch && ALU_zero) ? PC_nxt_imm : PC_nxt_4));
134               end
135           end
136           MulDiv :
137           begin
138               if(ALU_ready)
139               begin
140                   state_nxt = Standard;
141                   PC_nxt <= (Jump && mem_rdata_I[3])? PC_nxt_imm : ((Jump && ~mem_rdata_I[3]) ? ALU_out : ((Branch && ALU_zero) ? PC_nxt_imm : PC_nxt_4));
142               end
143               else
144               begin
145                   state_nxt = state;
146                   PC_nxt <= PC;
147               end
148           end
149           default :
150           begin
151               state_nxt = state;
152               PC_nxt <= PC;
153           end
154       endcase
155   end
156
157   //==== Sequential Part =======================
158   always @(posedge clk or negedge rst_n) begin
159       if (!rst_n) begin
160           PC <= 32'h00400000; // Do not modify this value!!!
161           state <= Standard;
162       end
163       else begin
164           PC <= PC_nxt;
165           state <= state_nxt;
166       end
167   end
```

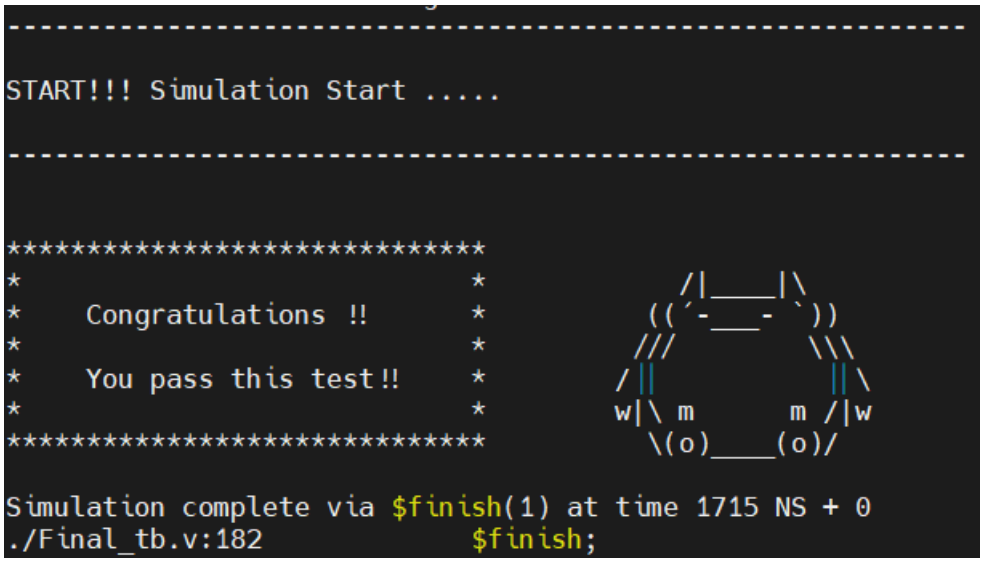
We use two states to tackle multi-cycle tasks. There are Standard and MulDiv. Standard state is the general state. CPU remains at this state until it gets the instructions of mul or div. At this state, PC_nxt will vary from the received instructions, which means CPU will keep running the following instructions. However, when CPU gets mul or div instructions, it will change to MulDiv state. It will stick to this state until ALU_ready is true, which means the multi-cycle task is done. At this state, PC_nxt will remain at this current PC until the mul or div instruction is finished. PC won't change until the multi-cycle task is done and then the state will go back to Standard and keep on running the following instructions.

Total simulation time

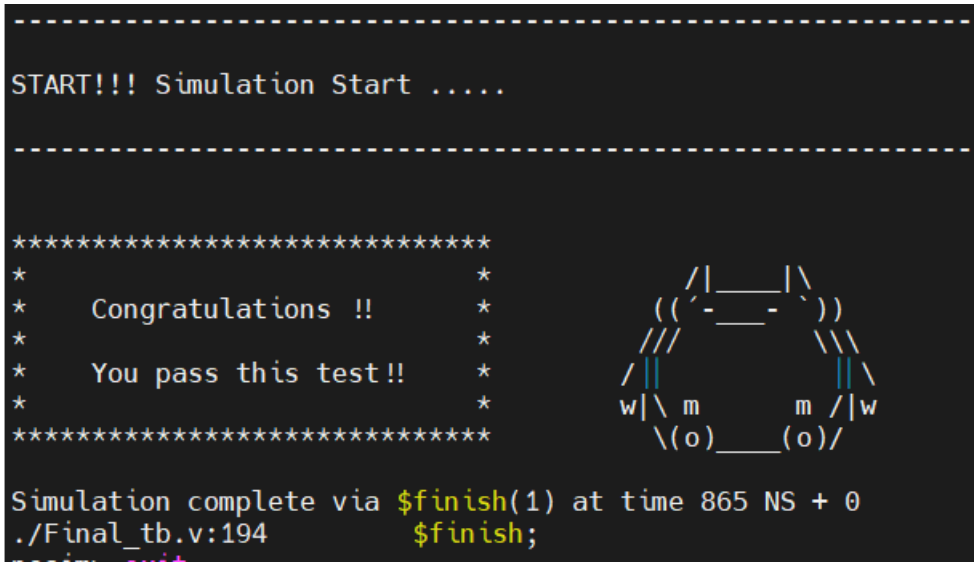
Leaf : a = 3, b = 9, c = 5, d = 17



Perm : n = 8, r = 5



Bonus : n=11



Observation :

We observe that leaf takes the least time to finish the task because it has the fewest instructions and there is no MUL instruction(multi-cycle task), which would take 32 cycles. As for perm and bonus, although perm has fewer instructions than bonus, it still takes more time than bonus because the instructions of perm contain MUL, which takes much more time to be completed than the other instructions. (We don’t use MUL in bonus)

From the above observation, we realize that it takes much more time with instructions like MUL (takes more cycles), even though it takes only one instruction and allows us to reduce the usage of instruction memory. Overall, It demonstrates the trade-off between memory and time.

Register table:

```
design_vision> read_verilog CHIP.v
Loading db file '/usr/cad/synopsys/synthesis/cur/libraries/syn/gtech.db'
Loading db file '/usr/cad/synopsys/synthesis/cur/libraries/syn/standard.sldb'
  Loading link library 'gtech'
Loading verilog file '/home/raid7_2/userb08/b8505049/CA_final/CHIP.v'
Detecting input file type automatically (-rtl or -netlist).
Reading with Presto HDL Compiler (equivalent to -rtl option).
Running PRESTO HDLC
Warning: Can't read link_library file 'your_library.db'. (UID-3)
Compiling source file /home/raid7_2/userb08/b8505049/CA_final/CHIP.v

Statistics for case statements in always block at line 120 in file
        '/home/raid7_2/userb08/b8505049/CA_final/CHIP.v'
================================================
|           Line           |  full/ parallel  |
================================================
|           122            |    auto/auto     |
================================================

Inferred memory devices in process
        in routine CHIP line 158 in file
                '/home/raid7_2/userb08/b8505049/CA_final/CHIP.v'.
===============================================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================================
|       PC_reg        | Flip-flop |  31   |  Y  | N  | Y  | N  | N  | N  | N  |
|       PC_reg        | Flip-flop |   1   |  N  | N  | N  | Y  | N  | N  | N  |
|      state_reg      | Flip-flop |   3   |  Y  | N  | Y  | N  | N  | N  | N  |
===============================================================================================

Statistics for case statements in always block at line 183 in file
        '/home/raid7_2/userb08/b8505049/CA_final/CHIP.v'
================================================
|           Line           |  full/ parallel  |
================================================
|           184            |    auto/auto     |
================================================
```

```
Statistics for case statements in always block at line 222 in file
       '/home/raid7_2/userb08/b8505049/CA_final/CHIP.v'
============================================
|          Line          |  full/ parallel  |
============================================
|          223           |    auto/auto     |
============================================

Statistics for case statements in always block at line 258 in file
       '/home/raid7_2/userb08/b8505049/CA_final/CHIP.v'
============================================
|          Line          |  full/ parallel  |
============================================
|          259           |    auto/auto     |
|          261           |    auto/auto     |
|          271           |    auto/auto     |
============================================

Statistics for case statements in always block at line 326 in file
       '/home/raid7_2/userb08/b8505049/CA_final/CHIP.v'
============================================
|          Line          |  full/ parallel  |
============================================
|          327           |    auto/auto     |
|          342           |    auto/auto     |
============================================
Warning:  /home/raid7_2/userb08/b8505049/CA_final/CHIP.v:375: signed to unsigned conversion occurs. (VER-318)
Warning:  /home/raid7_2/userb08/b8505049/CA_final/CHIP.v:382: signed to unsigned conversion occurs. (VER-318)

Inferred memory devices in process
       in routine reg_file line 378 in file
               '/home/raid7_2/userb08/b8505049/CA_final/CHIP.v'.
=================================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
=================================================================================
|       mem_reg       | Flip-flop |  994  |  Y  | N  | Y  | N  | N  | N  | N  |
|       mem_reg       | Flip-flop |  30   |  Y  | N  | N  | Y  | N  | N  | N  |
=================================================================================
```

```
Statistics for MUX_OPs
==================================================
| block name/line  | Inputs | Outputs | # sel inputs |
==================================================
|   reg_file/370   |   32   |   32    |      5       |
|   reg_file/371   |   32   |   32    |      5       |
==================================================
Warning:  /home/raid7_2/userb08/b8505049/CA_final/CHIP.v:425: signed to unsigned assignment occurs. (VER-318)

Statistics for case statements in always block at line 430 in file
        '/home/raid7_2/userb08/b8505049/CA_final/CHIP.v'
===============================================
|          Line           |  full/ parallel  |
===============================================
|          432            |    auto/auto     |
|          437            |    auto/auto     |
===============================================

Statistics for case statements in always block at line 490 in file
        '/home/raid7_2/userb08/b8505049/CA_final/CHIP.v'
===============================================
|          Line           |  full/ parallel  |
===============================================
|          492            |    auto/auto     |
===============================================

Statistics for case statements in always block at line 508 in file
        '/home/raid7_2/userb08/b8505049/CA_final/CHIP.v'
===============================================
|          Line           |  full/ parallel  |
===============================================
|          510            |    auto/auto     |
===============================================

Statistics for case statements in always block at line 536 in file
        '/home/raid7_2/userb08/b8505049/CA_final/CHIP.v'
===============================================
|          Line           |  full/ parallel  |
===============================================
|          538            |    auto/auto     |
===============================================
```

```
Inferred memory devices in process
        in routine mulDiv line 567 in file
                '/home/raid7_2/userb08/b8505049/CA_final/CHIP.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|      shreg_reg      | Flip-flop |  64   |  Y  | N  | Y  | N  | N  | N  | N  |
|     alu_in_reg      | Flip-flop |  32   |  Y  | N  | Y  | N  | N  | N  | N  |
|      state_reg      | Flip-flop |   3   |  Y  | N  | Y  | N  | N  | N  | N  |
|     counter_reg     | Flip-flop |   5   |  Y  | N  | Y  | N  | N  | N  | N  |
===============================================================================
Presto compilation completed successfully.
Current design is now '/home/raid7_2/userb08/b8505049/CA_final/CHIP.db:CHIP'
Loaded 7 designs.
Current design is 'CHIP'.
CHIP ImmdiateGeneration Control ALUControl ALU reg_file mulDiv
```

Distribution table:

林楷崴 b08505039	劉旻鑫 b08505049
CPU architecture	Code debugging
Code debugging	RISC-V code for bonus
Report	Report