Computer Organization Lab05 Report

Author: 109652039 林立倫, 109550003 陳茂祥

1 Implementation Details

Most of the implementations are kept unchanged w.r.t. Lab04.

1.1 Decoder

We set the `ALUOp` of I-type instructions (excluding `load`) to `2'b11`, so that we can assign the control signals correctly in `ALU_Ctrl`.

1.2 Forwarding Unit / Hazard Detection Unit

We implement the logic based on lecture slides.

1.3 Pipeline Registers

We simply connect the input signals to the output registers. Besides, we use non-blocking assignment since they are synchronous sequential circuits.

1.4 Immediate Generator

Compare to Lab4, we implement the `shift left one unit` this time. Therefore, we remove the 1'b0 padding at the end of the immediate value of `branch` and `jump`.

1.5 Pipeline CPU

We simply connect the input and output wires or registers. For the output of decoder, we setup an 8-bit wire by concatenating `{ RegWrite, Jump, MemtoReg, MemRead, MemWrite, ALUOp, ALUSrc }`. For `MUXPCSrc` and `IFID_Flush`, we assign the result of `( Branch & Branch_zero ) | Jump` to them. Last, we determine the value of `Branch_zero` by checking whether `RSdata_o` equals to `RTdata_o` or not.

1.6 Multiplexers

We add a parameter `size` to the 2-to-1 multiplexers to deal with the 8-bit decoder output without lots of efforts. In 3-to-1 multiplexers, we set the output to be `data2_i` when the selection input is `1x`, as a result, we can take `{ Jump, MemtoReg }` as the selection input of the multiplexer in the WB stage.

2 Results

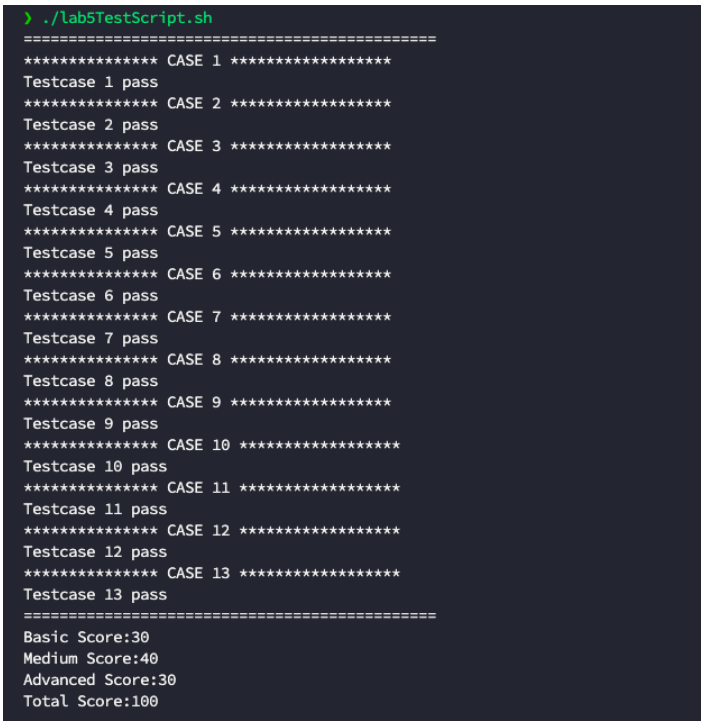

Figure 1: result

3 Difficulties and Solutions

- The pipeline registers delayed two cycles instead of 1 cycle.

  Solution: With misunderstanding, we used internal stored registers, so it took 2 cycles for an input to propagate to the output. We found the mistake when looking at the wave diagram during the debugging of other components, and in order to fix this, we find some online resources and learn how to define pipeline registers.

- Immediate in `load` / `jump` instructions is incorrect when we directly copy the code of Lab04.

  Solution: As we mentioned in 1.4, this is cause by the presence of the shift left one unit, we simply remove the padding zero in LSB to get the correct result.

- Cannot distinguish normal I-type instructions and Load/Store in `ALU_Ctrl`

  Solution: As the description of decoder, we forced the `ALUOp` of I-type instructions to be `2'b11`.