Computer Organization

架構圖:

設計模組分析:

(almost all modules are from lab 4)

Pipe\_CPU.v: The top module,we add the hazard detection unit and forward unit to this CPU. If we find the's a hazard, we need to fetch the temporary results in pipe register and send them to previous stages and maybe raise some stall/flush signals.

Decoder.v: Decodes the type of instructions and output some control signals for other muxes, like RegWrite and Branch.

alu\_top.v: The very basic a-bit ALU, including and, or, add, less four operations.

ALU.v: The 32 bit ALU, connected by 32 alu\_top modules. It decodes the ALU\_control signal and does the corresponding operations.

ALU\_Ctrl.v: According to the ALUOp signal from the instruction decoder to generate control signals for ALU.v Input: which type the instruction is. Output: what operation should the ALU do.

Adder.v: Just a very simple 32-bit adder, used for program counter (+4 and jump).

Data\_Memory.v: Used for memory storage.

Instr\_Memory.v: The instruction Memory of the CPU, where we store our program.

Reg\_File.v: Our register file, r0 to r14 live here. RegWrite\_i used to decide if we want to overwrite the registers or not.

MUX\_2to1.v & MUX\_4to1: Implement 2x1 and 4x1 muxes.

ProgramCounter.v: The PC of our CPU, it just passes the 32-bits bus from pc\_in\_i to pc\_out\_i during clock edges. Add another signal called pcwrite to decide if we can modify PC or not (load-use cases).

ShiftAdmount\_Extend.v: Does zero extension actually.

Shift\_Left\_Two\_32.v: Append two zeros to the input signal.

Sign\_Extend.v: Just repeat the MSB for 16 times and insert that in front of data\_i. In 'ori' case, we insert 16 zeros.

Hazard.v : Implement the hazard detection on the textbook, consider branch and jump cases.

Pipe\_Reg.v: The pipeline unit, keep the registers and control signals and pass them to next stages. We fix/rename the misunderstanding of stall signal to flush.

Forwading.v: The forwarding unit on the textbook. It the previous stage of pipelining need the result of the following stages (src == dst), than send them back. We have to notice that R0 is immutable. We add two muxes to read the forwarda/forwardb signal and choose the correct input of ALU.

完成部分: All basic requirements and advances.

遇到問題及解決方法:

In lw stall cased we need insert another nop and keep the old value of regwrite.  
There's no testcase from TAs so we need to write our own.

心得收穫:

Understand the issues and solutions/implementation of hazard detection mechanism in MIPS architecture.

分工表:

0016014徐若揚 Basic and Advance.

0016045李晏銘 Code review and Report.