Computer Organization

架構圖:

設計模組分析:

(almost all modules are from lab 3)

Simple\_Single\_CPU.v: The top module, the key point of this assignment. The only change in this assignment is adding pipe lines to each stage. Connect all the control signals or data buses to the pipe of this stage and then pass to out.

Decoder.v: Decodes the type of instructions and output some control signals for other muxes, like RegWrite and Branch. (fix the lui instruction pattern)

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 (from TA).

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

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. (from TA)But we have no Reg\_File.v from TA.

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.

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.

完成部分: All basic requirements and advances.

遇到問題及解決方法:

心得收穫:

Understand the simple pipeline implementation in MIPS architecture.

分工表:

0016014徐若揚 Basic and advance.

0016045李晏銘 Report.