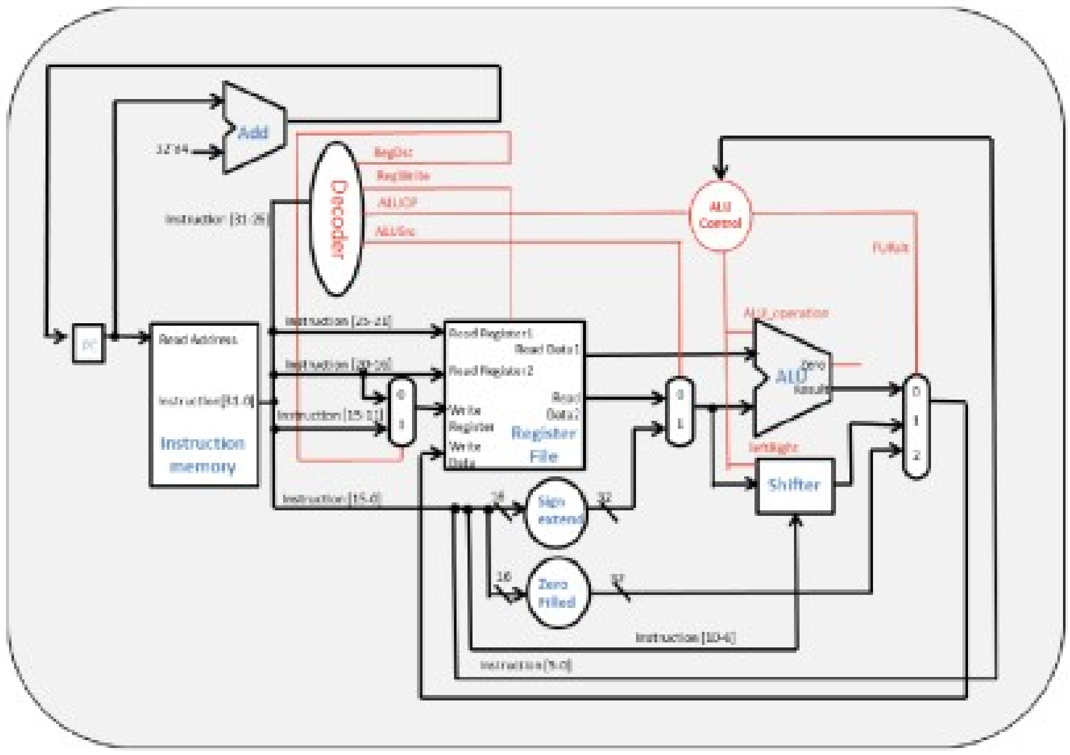
Computer Organization

Architecture diagrams:



Hardware module analysis:

ALU_top.v: From Lab1's document. It's a basic one-bit ALU, including and, or, less, add four operations.

ALU.v: From Lab1's document. Comprised by 32 ALU_top modules, perform a 32-bit ALU. It decodes the ALU control signal and performs the operations which we want.

Instr_Memory.v: From TA's document. It's the instuction memory of CPU.

Program_Counter.v: From TA's document. Pass the 32-b its bus from pc_in_i to pc_out_i during clock edges.

Reg_file.v: From TA's document. All the register datas are inside this module and the 'Reg_Write_i' is used to determine whether the data should be written into the register or not.

TestBench.v: From TA's document. Test the correctness of our design.

Adder.v: A 32-bit adder provides for our PC counter(+4).

Mux2to1.v: A simple 2x1 mux.

Mux3to1.v: A simple 3x1 mux.

ALU_Ctrl.v: Get the ALU_op signals from the decoder and the fuction field (instr   [5:0]), then output the ALU_operation for ALU.v and FURslt for 3x1 mux.

Decoder.v: Decode the instruction [31:26] and get the control signals for RegDst, Reg_Write, ALU_op, ALU_Src.

Shifter.v: Shift sftSrc left or right logically.

Sign_Extended.v: Repeat the MSB 16 times in the front of 'data_i '.

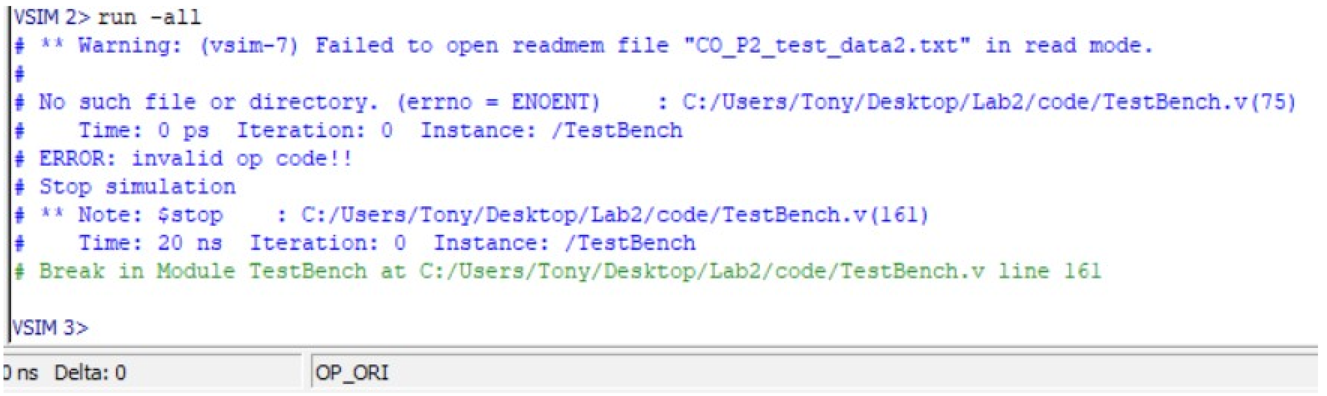
Simple_Single_CPU.v: The top module connects all the modules inside the CPU.

Zero_Filled.v: Fill 16 zeros at the back of 'data_i'.

Finished part:

All the basic instructions in the TA's pdf.

Problems you met and solutions:



```
VSIM 2> run -all
# ** Warning: (vsim-7) Failed to open readmem file "CO_P2_test_data2.txt" in read mode.
#
# No such file or directory. (errno = ENOENT)     : C:/Users/Tony/Desktop/Lab2/code/TestBench.v(75)
#    Time: 0 ps  Iteration: 0  Instance: /TestBench
# ERROR: invalid op code!!
# Stop simulation
# ** Note: $stop    : C:/Users/Tony/Desktop/Lab2/code/TestBench.v(161)
#    Time: 20 ns  Iteration: 0  Instance: /TestBench
# Break in Module TestBench at C:/Users/Tony/Desktop/Lab2/code/TestBench.v line 161

VSIM 3>
```

Solution:

Put all *.mpi, *.mpf, and *.txt files together.

Problem:
Bonus instrutions

Solution:
We create a mux for seletcing the shamt's data source.

Summary:

Understand MIPS instruction how to control Simple_Single_CPU.
Understand each modules in the Simple_Single_CPU.
SRLV and SLLV is hard to implement QQ.

Reference:

TA’s pdf
http://fourier.eng.hmc.edu/e85_old/lectures/processor/node5.html
https://www.d.umn.edu/~gshute/mips/control-signal-summary.xhtml
http://stenlyho.blogspot.com/2008/08/16-bits-sign-extender-to-32-bits.html
https://hackmd.io/s/ryv1NT3S#%E7%AC%AC15%EF%BD%9E17%E8%AC%9B-Single-Cycle-Processor-73