CMPE 200 – Assignment 6

Haonan Wang

Reference: Donald Hung

Computer Engineering Department, San Jose State University

Enhanced Single-cycle MIPS Processor

Description:

In this lab you will extend the initial design of the single-cycle MIPS processor (from Lab #5) to support more MIPS instructions. You are required to enhance the single-cycle MIPS processor's functionality by extending its instruction set (`add, sub, and, or, slt, lw, sw, beq, j, addi`) to cover the following additional instructions: `MULTU, MFHI, MFLO, JR, JAL, SLL, SLR.`

Your design must be tested via both functional verification. Schematics of the initial MIPS design (used in Lab #5 with minor differences) and the test program (fatorial.asm) for your extended design are provided in Attachment 1. As a reminder, machine code of a test program should be stored in the memory file named *memfile.dat* (or *memfile.mem*), which should be generated by the MIPS assembler and placed in your project directory. This file should contain one 32-bit piece of machine code (in hex) per line (unused memory space should be filled with zero's).

You should document your work professionally. In addition to the essential sections, your report should include schematics (generated by professional drawing tools) for the enhanced processor micro architecture, tables for the control unit decoders, the simulation log, testbench waveforms, and commented source code. Signal naming must adhere with the list given in Assignment 5. **Any microarchitecture changes (modification and/or extension) to the initial design (figure and source code) must be in a different color and clearly noted.**

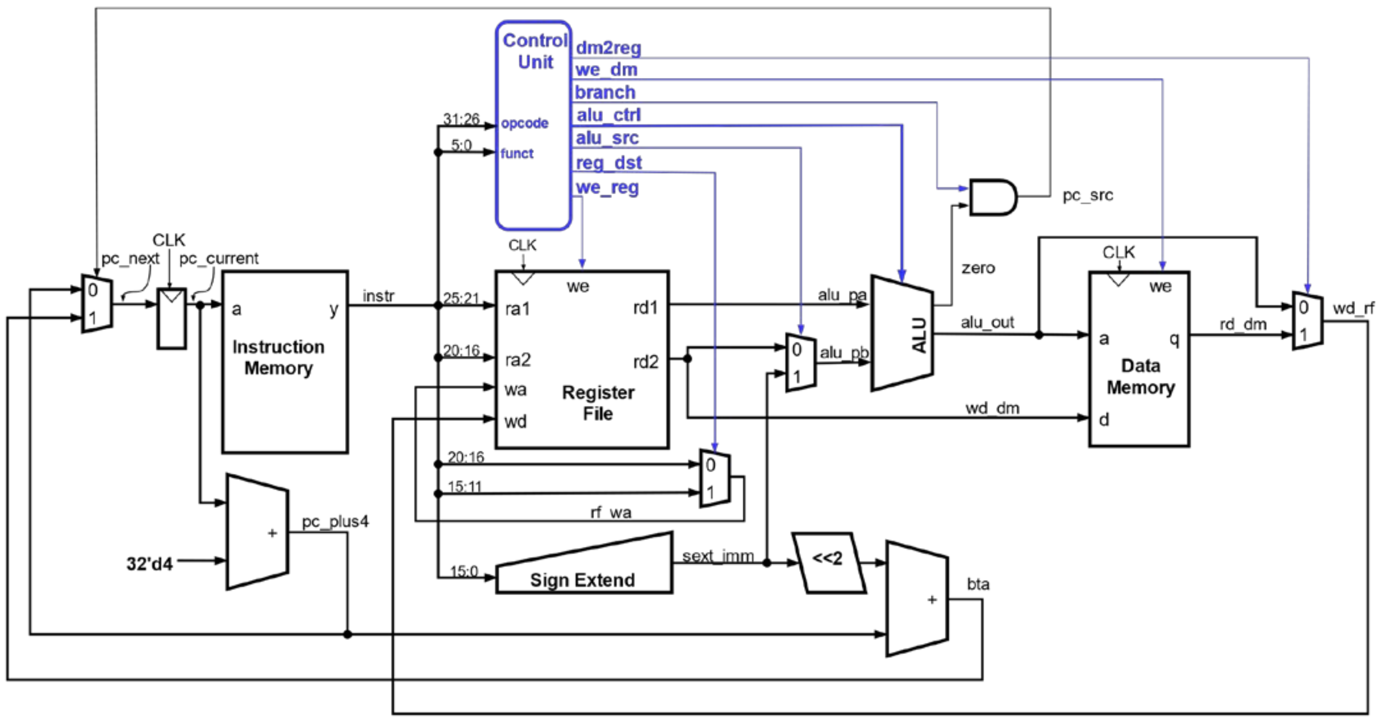
This is a two week long lab. This extended time is to allow for proper development of the extended MIPS processor. The task is as follows:

1. Official draft (generated by a professional drawing tool) of extended MIPS microarchitecture (both datapath and control)

2.Control unit truth tables (in MS Excel files)

3. Test plan, testbench, and simulation results

Attachment 1: Initial Design of the Single-cycle MIPS (w/o jump)



Attachment 2: Test Program I

Test code for extended design. Use the MIPS assembler/simulator to test the code first!

```
main:
       addi $sp, $0, 48
       addi $a0, $0, 4      # set arg
       jal factorial        # compute the factorial
       add $s0, $v0, $0     # move result into $s0
       j end
factorial:
       addi $sp, $sp, -8    # make room on stack
       sw $a0, 4($sp)       # store $a0
       sw $ra, 0($sp)       # store $ra
       addi $t0, $0, 2      # $t0 = 2
       slt $t0, $a0, $t0    # a <= 1 ?
       beq $t0, $0, else    # no - goto else
       addi $v0, $0, 1      # yes - return 1
       addi $sp, $sp, 8     # restore $sp
       jr $ra               # return
else:
       addi $a0, $a0, -1    # n = n - 1
       jal factorial        # recursive call
       lw $ra, 0($sp)       # restore $ra
       lw $a0, 4($sp)       # restore $a0
       addi $sp, $sp, 8     # restore $sp
       multu $a0, $v0       # n * factorial(n-1)
       mflo $v0             # mv result into $v0
       jr $ra
end:
```