Indian Institute of Technology, Delhi

FALL, 2011

COMPUTER ARCHITECTURE

Homework 3

(DEADLINE:Friday October 7th, 11:55 PM)

Total Marks: 20

Policies:

1. If you are late by n days and m minutes, then we will assume that you are n + 1 days late. There is a 4 mark penalty per day.
2. You are required to **submit 1 zip** file containing all the .circ files for your design **on moodle.** The name of the .zip file should have the format <entry no.>.zip. Use lowercase letters.

Submission: All submissions will be done through Moodle. https://jaijaivanti.cse.iitd.ernet.in

Files in this homework:
1. hw3.pdf

Consider a toy architecture with the following details:

Instruction	Opcode	Instruction format
Add	0001	Opcode rs rt rd
Sub	1110	Opcode rs rt rd
And	0011	Opcode rs rt rd
Or	1100	Opcode rs rt rd
Load	0111	Opcode rd memadd
Store	1000	Opcode rd memadd
Logical Shift Left	0101	Opcode rs rd num
Arithmetic Shift Right	1101	Opcode rs rd num

Where:

rd -> destination register
rs -> source register 1
rt -> source register 2
memadd -> memory address. It is the actual physical address of your memory, implying that memadd = 4 is the 5^{th} entry in your memory.
num -> num of bits to be shifted

Assume that Instructions are of 16 bits with following split:
opcode – 4 bits
register address (i.e. rs/rt/rd) – 4 bits
memadd -> 8 bits
num-> 4 bits

All the registers are 16 bit wide and the register file contains 16 registers.
Instruction and data memory have 256 entries each.

In this assignment you will be using an open-source hardware simulation tool – **Logisim :** http://ozark.hendrix.edu/~burch/logisim/index.html .
Please download version 2.7.1 .

You are required to implement the following modules:

1. ALU – This unit has to be completely designed by you.
2. Register File – A group of registers with logic to choose the correct register to be read/written.
3. Instruction Memory
4. Data Memory
5. Control Unit – You can use “Window->Combinational Analysis” to implement this unit.

You have to do **non-pipelined** implementation, i.e. An instruction is executed when the previous one has finished.

Initialize PC to 0, i.e. instructions to be executed will be available from address 0 in the instruction memory. Increment PC only after the instruction has completed its execution. You can use a **counter** to achieve the same. Also, you are allowed to **use the inbuilt adder only to increment the PC**.
You can only use the following entities in logisim to implement your logic:

1. Everything from Gates directory.
2. Everything from Plexers directory.
3. Everything from Wiring directory.
4. Everything from Memory directory.
5. Adder from Arithmetic Directory only to increment the PC. You will have to implement your own adder for ALU.

   You are encouraged to keep the design as modular as possible. Please create library units for each unit and use that in your main circuit.
   Name your main circuit file as **hw3.circ** .