Computer Architecture

Project 2

Cycle Accurate Simulator

Dr. Chris Martinez

Spring 2017

The final project is to create a non-pipeline DLX hardware simulator. For this project you will be using the assembler from project 1 and the data path for the DLX CPU. You will need to 1) create the DLX simulator and 2) write a test programs for the simulator.

How to run the simulator:

The simulator will run in a single step cycle function.  The first part of the assignment is

to create the state machine for the following instructions

ADD      ADDI      AND

BEQZ      BNEZ      J      LB

LH      LW

OR      ORI      SLL      SRL

SB      SH      SW      SUB

The simulator will run one state at a time and display all the registers found in the CPU.  An example of what should be display at the end of each cycle is:

 Register File:

R0: 0   R1: 13   R2:  0  R3:  12  R4: 40  R5: 60  and so on for all 32 registers

 Memory Registers:

MDR:  0   MAR: 0

 Hardware Registers:

IR:             PC:

 Data Bus:

S1:             S2:             Dest:          Addr:               Data:

 Register Buffers

A:              B:                C:

After the cycle is complete the simulator will wait until the user says it should continue to the next cycle.

The program should be created so that the simulation runs using the datapath discuss in class. The setting of the bus and registers should be done through the control signals in the state machine. For example, PC is updated only if the PCload signal is set to 1. All instructions will run the same segment of code in your program. The control signals will set the registers and how to use the ALU.

Help in creating the simulator:

 Memory:

The simulator will have main memory for reading and storing data and instructions.  You should have the following items for memory:

char MAIN\_MEMORY[5000];

Main memory will be created as an array of bytes

You will also need to have the following functions

 int Load\_Program\_Code(char\* filename);

This function will read in the hex file from your assembler.  You need to set which type of Endian you are going to use. You read in a word and store the bytes correctly. The function will return 1 if file is read in successfully or 0 otherwise.

int Read\_Memory(int address, int word\_byte\_half);

This function will be used to read from main memory.  You will supply the address

and the data size word, byte or half word. The value return is the data from memory

 int Write\_Memory(int address, int data, int word\_byte\_half);

This function will be used to write to main memory.  You will supply the address, data

and data size .  The value return is 1 if the write was successful and 0 otherwise

State machine:

In order to make your program work you will need to develop the state machine. Each state has the control signals needed.

struct state{

char S2op, ALUop, Cload, REGload, Aload, Aoe, Bload, Boe, REGselect, IRload, IRoeS1,IRoeS2, Reset, PCload, PCoeS1, PCMARselect, MARload, MemRead, MemWrite,

MemOp;

//other information maybe needed depending on how you decide to create the state

};

Test Files:

You will be testing your simulator by writing two test files.

First Test File

Create fibonacci sequence for the first 10 terms. The Fibonacci sequence should be created using a loop that stores the terms in memory. When the loop is done read from memory and store the results in to R20 to R292

 The second test file will do the following:

1) Load the value 45690 into R3.

2) OR R3 with the imm value of 75 and place result in R4

3) Store the byte of R3 in one memory location

4) Store the halfword of R3 in one memory location

5) Store the word of R3 in one memory location

6) Take the value in R4 and shift left 18 times and store in R5

7) Take the value in R5 and shift right 5 times and store in R6.

8) Take the value in R6 and subtract 5 from the answer and store in R7

9) And R7 and R3 together and place in R8

10) OR R8 and R3 together and place in R9