## **RISC-V SIMULATOR**

Team Details:

Muskan Gupta 2019CSB1100

Manan Singhal 2019CSB1099

Praveen kumar 2019CSB1108

Pratima Singh 2018CEB1021

Rajasekhar 2019CSB1105

**Design document:**

This document describes the design of the Phase3.py , a RISC-V Simulator.

# **How to execute a program:**

**Use Command:**

$python3 Phase3.py “machine\_file\_name.mc”

Example: # $python3 Phase3.py testcase1.mc

**Input:**

Input to the terminal is a .mc file **before running as shown in the example above** the program that contains the encoded instruction and the corresponding address at which instruction is supposed to be stored, separated by space. For example:

0x0 0x00500513

0x4 0x008000EF

0x8 0x04000463

And, data segment is also present in .mc file after an empty line in between text segment and data segment.

**While running the file:**

When you run the file,

1. First it will ask the user whether to use Pipelining or non pipelining (Knob1)
   1. Enter 0 for Non pipelining or 1 for Pipelining
2. Next , it will ask for the following inputs in a sequence. Give the inputs accordingly.

“Enter CacheSize :”

Enter CacheBlock Size :

Enter no. of ways per set of associativity :

We have assumed that CacheBlock Size is of power 2 and greater than equal to 4. So enter the input accordingly

Then the code will run and give output accordingly.

**How the Simulator works and Output format:**

First we initialise the PC to zero and then we will increment the PC by 4 after every instruction. For every PC address we will first try to find that address in the Instruction cache , if it is found then we will decode the instruction, read the register, execute the operation,access the memory (if required) and write back if required to the register file. If it is not found in the Instruction cache, then we will fetch that instruction from the main memory and we perform the rest of the stages : decode, execute, memory access,writeback etc.

Similarly We implemented Data cache too.We assumed that we are accessing data cache for every byte we need.

The instruction set supported is the same as given in the lecture notes.

OUTPUT FORMAT(Example):

The output will be printed in the following way:

“Stats for instruction cache:

Number of instruction cache access : 109

Number of instruction main memory access : 20

Number of hits in instruction cache : 69

Number of miss in instruction cache : 20

Stats for data cache:

Number of data cache access : 90

Number of data main memory access : 50

Number of hits in data cache : 58

Number of miss in data cache : 40 “

**Test cases**

* We have included some test cases files we have tested upon in the zip file.