CS204 Project Phase-1 Documentation

**Functional simulator for a subset of RISC-V instruction set**

This document describes the design aspect of a GUI-based functional simulator for a subset of RISC-V instruction set.

**Input/Output Mechanism**

Input to the simulator is a “input.mc” file that contains the encoded instructions and their corresponding addresses separated by a space along with data values and their corresponding addresses separated by a space. The instructions and data values are separated by a ‘$’ sign.

*Example:*

0x0 0xE3A0200A

0x4 0xE3A03002

0x8 0xE0821003

0xC $

0x10000000 0x2

0x10000004 0x8

The instructions are stored in the instruction memory and the data values are stored in the data memory.

The simulator then reads instructions from the instruction memory, decodes the instruction, reads from registers, execute the operation, and then writes back to the register file.

The GUI shows all the register values and updated memory values after each executed instruction and the type of each instruction (add, xor, bge, jal, etc.).

**Design Of Simulator**

**Data Structure**

Register, instruction memory, data memory, intermediate value and control signals (for each stage of instruction execution) are declared as global variables. Register values are stored in an integer array of size 32. The instruction and data memory are stored as C++ STL vectors of string and integer type respectively.

**Simulator Flow**

There are mainly two steps :-

1. Memory is loaded using a “input.mc” file.
2. Simulator executes each instruction one by one and updates & displays each register value and updated memory values using its GUI.

**Implementation of Fetch, Decode, Execute, Memory Access, Write Back**

1. **Fetch:**
2. **Decode:** Firstly the instruction format is identified using the opcode, func3 & func7 values, then the operation is identified
3. **Execute:**
4. **Memory Access:**
5. **Write Back:** If RF\_Write signal is on and RD is not equal to 0, then RD is updated.

**Testing Of Simulator**

We test the simulator using the following assembly programs :-

1. Fibonacci Number Program
2. Sum of the array of N elements. Initialize an array in first loop with each element equal to its index. In second loop find the sum of this array, and store the result at Arr[N].
3. Bubble Sort Program