Design Document: Functional Simulator for Subset of RISC-V instruction set

The document describes the design aspect of myRISCVsim, a functional simulator for a subset of the RISC- V instruction set.

# Input/Output

## Input

Input to the simulator is a MC file that contains the encoded instruction and the corresponding address at which instruction is supposed to be stored, separated by space. For example:

0x0 0xFFC10113

0x4 0x00D00193

0x8 0x00312023

0xC 0x00012203

0x10 0x00410113

0x14 0xEF00007F

## Functional Behavior and output

The simulator reads the instruction from instruction memory, decodes the instruction, reads the register, executes the operation, and writes back to the register file. The instruction set supported is the same as given in the lecture notes.

The execution of instruction continues till it reaches instruction “0x0000007F”, simulator stops and writes the updated memory contents onto a memory text file.

The simulator also prints messages for each stage, for example for the third instruction above the following messages are printed.

* Fetching instruction 312023 at pc=8
* Decoded instruction as:
* opcode: 23
* rd: 0
* func3: 2
* rs1: 2
* rs2: 3
* func7: 0
* imm12: 3
* executing instruction: 2 + 0
* Reading from memory
* Writing to memory

# Design of Simulator

## Data structure

Registers, memories, intermediate output for each stage of instruction execution are declared as global variables. Being static, the variables are not visible outside the file, thus making the data encapsulated in the myRISCVSim.c.

## Simulator flow:

There are two steps:

1. First memory is loaded with an input memory file.
2. Simulator executes instructions one by one.

For the second step, there is an infinite loop, which simulates all the instructions till the instruction sequence reads “0x7f”.

Next we describe the implementation of fetch, decode, execute, memory, and write-back function.

TODO

# Test plan

We test the simulator with following assembly programs:

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