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

This document describes the design aspect of myRICSVSim, a functional simulator for a subset of the 32-bit RISC-V instruction set.

# Input/Output

## Input

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

0x0 0xE3A0200A

0x4 0xE3A03002

0x8 0xE0821003

## Functional Behavior and output

The simulator reads the instructions from instruction memory, decodes the instructions, reads the registers, executes the operations, and writes back to the register file. The instruction set supported is the same as the one taught in the lectures.

The execution of instruction continues till it reaches the instruction “swi 0x11”. In other words, as soon as the instruction reads “0xEF000011”, the simulator stops and writes the updated memory contents onto a .mc file.

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

* Fetch
  + “FETCH: Fetch instruction 0xE3A0200A from address 0x0”
* Decode
  + “DECODE: Operation is ADD, first operand is R2, second operand is R3, destination register R1.”
  + “DECODE: Read registers R2 = 10, R3 = 2”
* Execute
  + “EXECUTE: ADD 10 and 2”
* Memory
  + “MEMORY: No memory operation.”
* Writeback
  + “WRITEBACK: Write 12 to R1”

# Design of Simulator

## Data structure

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

## Simulator flow

There are two steps:

1. First, the memory is loaded with an input memory file.
2. Second, the simulator executes instruction one by one.

For the second step, there is an infinite loop, which simulates all the instructions till the instruction sequence reads “swi 0x11”.

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

# Implementation

## Fetch

Description

## Decode

Description

## Execute

Description

## Memory

Description

## Write-Back

Description

# Test plan

We test the simulator with the following assembly programs:

* Fibonacci Program
* Factorial Program
* Bubble Sort Program