Design Document: Functional Simulator for RISC-V instruction set

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

Instructions that Simulator supports are-

R format - add, and, or, sll, slt, sra, srl, sub, xor, mul, div, rem

I format - addi, andi, ori, lb, ld, lh, lw, jalr

S format - sb, sw, sd, sh

SB format - beq, bne, bge, blt

U format - auipc, lui UJ format - jal

# Inout/Output

## Input

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

0x0 0xE3A0200A

0x4 0xE3A03002

0x8 0x003202B3

## Functional Behavior and output

The simulator reads the instruction from instruction memory, decodes the instruction, read the register, execute the operation, and write back to the register file. The instruction set supported is same as given in the Project\_Phase1.pdf .

~~/The execution of instruction continues till it reaches at the end of instruction.mc file. simulator stops and writes the updated memory contents on to a memory text file. /~~

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

* Fetch prints:
  + INSTRUCTION : 0x003202B3
* Decode prints:
  + decode:
  + instruction is of r format(load)
  + add rs1: 4 rs2: 3 rd: 5
  + values [rs1]: 0 [rs2]:0 “(values that you stores initially in registers 4 and 5 here I am assuming as 0 and 0) “
* Execute prints:
  + ALU
  + OPERATION Preforming : add
  + RZ = sum : 0
* Memory:
* Writeback:
  + writes 0 in register 5

# Design of Simulator

## Data structure

Registers, memories, intermediate output for each stage of instruction execution are declared as global variable .

## Simulator flow:

There are two steps:

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

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

decode()

This function takes instruction as an input from IR(updated by fetch stage) and extracts its opcode,fun3,fun7 based on this it’s decode the type and format of instruction and return instruction’s mnemonic value

And update the global variable RA,RB,rd,imme as input(value) of source register 1,value of source register 2 ,destination register number and immediate value(sign extended) based on type of instruction.

Also prints the values in specific format shown above in decode prints stage

alu()

This function takes the variable ‘operation’ (output of decode) and global variables RA, RB, imme as its input.

The function computes either the arithmetic operation value or the target address according to the input operation and updates the grobal variable RZ as output.

The function also prints the name of operation it is performing, type of value in RZ and the value of the RZ

# Test plan

We test the simulator with following assembly programs:

* Fibonacci Program