DESIGN DOCUMENT : Functional Simulator for Subset of RISC-V instruction set

---

Design of phase-1 code: *F1.py*

INPUT/OUTPUT:

It takes the RISC-V assembly code as input and generates the corresponding Machine Code. Machine code is the code that a computer can understand. For example:-

The format of input would be as follows:

```
addi x6, x10, 0
lw x10, 0(x2)
```

The output generated would be like:

```
0x0   0x00050313
0x4   0x00012503
```

FUNCTIONAL BEHAVIOUR:

The function uses a memory class to store the .data variables. After it detects the .text part of the code, it starts encoding the instructions into machine codes. The program stores all the instructions in a list and it continues till all the instructions are finished.

Design of phase-2 code: *run3.py*

INPUT/OUTPUT:

It takes the Machine Code as input (output of F1.py) and executes the instructions. For example:-

The format of input would be as follows:

```
0x0   0x00050313
0x4   0x00012503
```

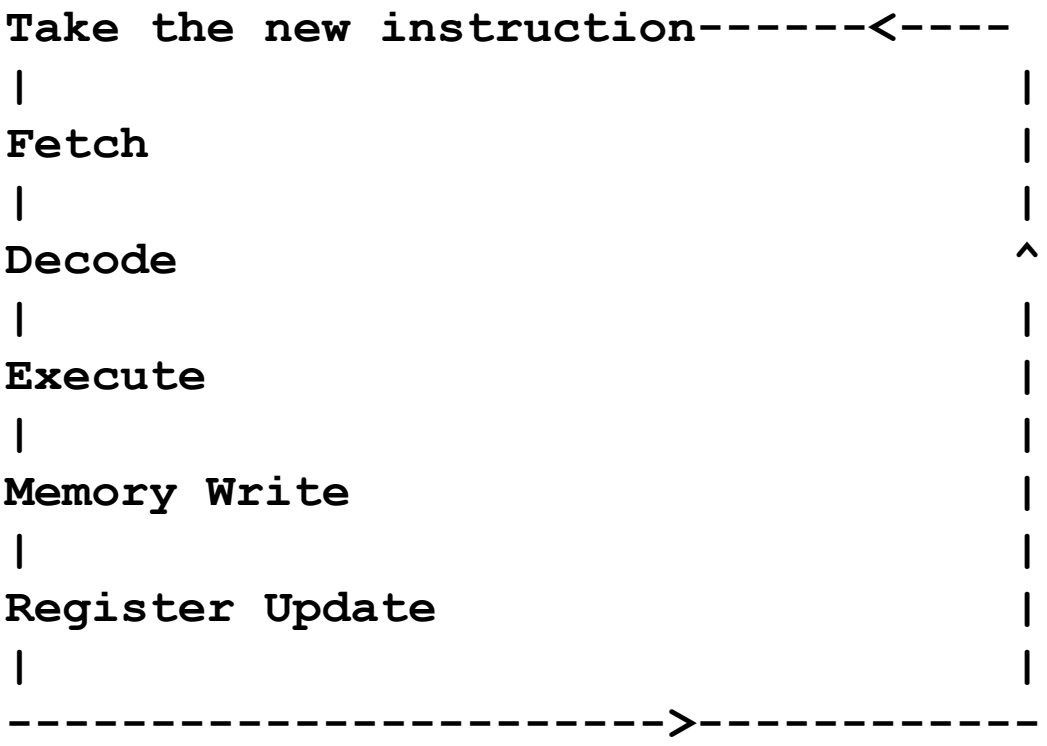
The output generated would be like:

```
FETCH:Fetch instruction 0x10000517 from address 0
DECODE: Operation is AUIPC, first operand is
Immediate field, destination register R 10
DECODE: The immediate value is 65536
Executed
Number of clock cycles: 1
…
…
### Code Exited with NO ERROR ###
------------------Registers-----------------------
x0:0
x1:12
x2:2147483632
x3:268435456
…
…
--------------------Memory-----------------------

0x10000000 :00000011
0x10000001 :00000000
0x10000002 :00000000
…
…
```

FUNCTIONAL BEHAVIOUR:

The function uses a memory class to store the values of sw, sd, sh,etc instructions. It also uses a register class to keep track of all register values. It follows the following flow:



The memory class contains functions for reading the value at a particular address as well as writing the value at any address. The register class too contains the functions or methods to read as well as write values. The only difference is register is an instance of object ‘list’ while memory is an instance of object ‘dictionary’.

All the PCs and the corresponding instructions are stored in a dictionary with PCs as keys and the instructions as values. The PCs of labels are also stored for the conditional and unconditional jumps.

TEST PLAN:

We test the simulator with following assembly programs:

- Fibonacci Program
- Factorial Program
- Bubble Sort Program

THANK YOU