CS204 - Computer Architecture
Course Project Details
Phase 1 Release Date: 15th Apr 2023
Deadline: 13th March 2023.

Our goal is to build a simulator that will execute RISCV-32I ISA. The overall project will have three phases, which will eventually bring out such a simulator. The whole task has been split into the below three phases to simplify overall process, and bring-in more clarity to you.

Phase	Description	Weightage	Bonus for GUI
1	Single-cycle execution	5%	1%
2	Pipelined execution	5%	1%
3	Memory hierarchy	5%	1%
	Total	15%	3%

Phase 1: Code for a 5 step instruction execution.
Design and implement the function simulator 32bit RISCV-32I ISA instructions.
Here, we would take the input from the .mc file (memory dump from venus simulator) and pick one instruction at a time, using the value in PC (0x0). You will write a C/C++ program. You would need to include structures for all the registers here - PC, IR, Register File, temporary registers (like RM, RY, etc), etc., five steps of instruction execution as functions.

All the instructions in the given in the input `.mc` file is executed as per the functional behavior of the instructions. Each instruction must go through the following steps:
Step 1: Fetch
Step 2: Decode
Step 3: Execute
Step 4: Memory Access
Step 5: Register Update or Writeback

Along with execution of instruction in steps, simulator should also provide messages what it is doing in each stage as an update in all the structures that would change. As part of the simulator, you should implement an additional instruction which exits the simulator, writes the data memory in `.mc` file before exiting. Further, introduce a variable "clock" that increments once for every instruction. Print the number of clock cycles at the end of each cycle.

Input to Phase 1: A file with RISC-V machine code. For convenience, write your assembly code and get the machine code from Venus simulator. Save it in a file, with extension ".mc". Make sure that text and data segments are clearly seperated (i.e., their address ranges are proper) in the file. `.mc` file is expected to have a format like below (<address of instruction><delimiter - space><machine code of the instruction>, one in each line):
Similar to Venus let us assume that code segment starts at 0x00000000 and data segment starts at 0x10000000, stack segment at 0x7FFFFFFC and heap segment at 0x10007FE8.

So, your code segment of `.mc` file would look like:
0x0 0x003100B3
0x4 0x00A37293
0x8 <your termination code to signify end of text segment and in turn, end of the assembly program>
Your data segment in the same `.mc` would look like:
0x10000000 0x10
....

Regarding the instructions that you need to consider:
1. Support for Pseudo instructions is not compulsory.
2. Limit to RISC-V 32bit ISA, specifically, below 26 instructions:
R format - `add, and, or, sll, slt, sra, srl, sub, xor`
I format - `addi, andi, ori, lb, lh, lw, jalr`
S format - `sb, sw, sh`
SB format - `beq, bne, bge, blt`
U format - `auipc, lui`
J format - `jal`
Also, you need not support other RISC-V instructions.

User-friendly GUI part will be left to you. Definitely such an effort will be given bonus points in the overall project evaluation. The bonus points would be considered within the weightage of the lab part of the course.

You must test the software using the machine code of the below programs:
- Fibonacci Program
- 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].
- Bubble Sort Program

To give an overall idea, I am sharing a `.rar` file with you. It contains several files for a C like implementation. Reading the `.mc` file might be slightly different but the essence remains same. The template implementation attached contains a README file, a design document, Makefile, and C code to start with. As part of your submission you are expected to submit the source code along with updated design document, README, input test files.
Overall project evaluation will be on:
- Functional completeness
- Documentation completeness
- Testcase

Revert if you have any questions.