CS204 - Computer Architecture
Course Project Details
Phase 1

This is a group task. Our goal is to build a simulator like Venus. The overall project will have four phases which will eventually bring out such a simulator. The whole task has been split into the below four phases to simplify overall process, and bring-in more clarity to you. User-friendly GUI part will be left to you. Definitely such an effort will be given bonus points (25%) in the overall project evaluation.

Phase	Description	Weightage	Bonus for GUI
1	Assembler	4%	2%
2	Single-cycle execution	4%	0
3	Pipelined execution	4%	1%
4	Memory hierarchy	4%	1%
	Total	16%	4%

1. **Phase 1**: Conversion of Assembly code to Machine code
   Here, you are doing the job of a 16bit RISC-V assembler. You will write a C/C++/Java program to take a `.asm` file as input and generate a `.mc` file.

   i) `.asm` file will have a format like below (one assembly instruction in each line)
   add x1, x2, x3
   andi x5, x6, 10
   ...

   ii) `.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
   ....

   As you would have sensed, your assembler code needs to parse the `.asm` file one or more times to precisely calculate and replace various labels appropriately.
   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 31 instructions:
   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`
Also, you can skip support floating point operations.

In addition, you need to provide support for the assembler directives: `.text, .data, .byte, .half, .word, .dword, .asciz`.

Minimum GUI expected (for bonus marks):
1. Editor pane, Simulator pane.
2. In Simulator pane, provide options for Step, Previous, Breakpoint, Run, Reset, Stop, Dump the dynamic instructions executed thus far. PC, Machine code, Basic code and Original code columns.
3. Display Registers, Memory. Display values in Decimal, Hex, ASCII.
4. Provision to choose text/data/stack/heap segments of the memory. Also, provision to enter an address of interest directly so that display moves to that point for convenience of the user.

You should be able to kick-off Phase 1 right away. Phase 1 and Phase 2 (details will be out during Mid-Feb) need to be submitted by 15th March 2020 11.55PM and will be evaluated during 16th-22nd March 2020.

Revert if you have any questions.