MTH410E – RISC-V Architecture and Processor Design

Homework 3 – Pipelined RISC-V Processor

Write a SystemVerilog code of a pipelined RISC-V processor. The designed micro-architecture should include all the instructions in RV32I. The processor should work as in-order and single-issue, so there is no need for FENCE instruction implementation. The processor should resolve the all data control hazards during runtime.

The top file of the processor should be as follows;

```
module riscv_singlecycle (
        parameter DMemInitFile = “dmem.mem”; // data memory initialization file
        parameter IMemInitFile = “imem.mem”; // instruction memory initialization file
        parameter LogFile = “rv_track.txt”; // log file to track the processor state and used for verification/grading
) (
        input clk_i; // input clock
        input rst_ni; // logic-0 asserted asynch reset
        output  [31:0] rf_data_hex // just for synthesis, otherwise the tools can remove everything for optimization
);

// module body
// you can also use multiple modules in different files

endmodule
```

The data and instruction memories of the processor should be initialized by *DMemInitFile* and *IMemInitFile* respectively (i.e., use $readmemh in the initial block of the memories).

To trace the processor state, RF write and DMEM write operations should be logged to *LogFile*. Therefore, following two functions should be used in RF and DMEM modules to track the written values. You can change the signal names (e.g., rf_idx_dec) with respect to your signal names.

```
$fwrite(LogFile, "x%0d 0x%16h", rf_idx_dec, rf_data_hex); // log the register file writes
$fwrite(LogFile, "mem 0x%h 0x%h", dmem_idx_dec, dmem_data_hex); // log the data memory writes
```

Deadline: 18/12/2023 23:59

Some resources;

- RISC-V: An Overview of the Instruction Set Architecture