Assignment #3

Objective:

In this assignment, you are asked to simulate a simple **multi-cycle processor** with all the details explained in the class using a limited ISA provided in the ZIP file (**simulate.zip**) with two variations; one with an ideal cache while the other one with a two-level cache organization.

Description:

The ZIP file contains four Python files, i.e. *simulate.py*, *processor.py*, *regfile.py*, and *instload.py*. For you to run the simulation, type at the command prompt as below:

- python3 simulate.py test.s

A sample input program, **test.s**, is a snippet provided to you as is to evaluate your implementation. A sample output, **test.out**, is the result of running the command above.

- Your answer for running **test.s** must match **test.out** exactly.
- You ***are not allowed*** to remove or change the printout format or add extra printout into the output.
- The printout of 'XXXXXXXX' means any value or don't care and has to be strictly followed according to rules in the processor design to optimize the hardware logics.

A few different snippets will be used to evaluate the correctness of your simulation, so you are encouraged to write your own snippets to evaluate your implementation.

- The snippets do not need to declare data section nor beginning of main function.
- The skeleton in the ZIP file assumes the first instruction will start at the address 40000.
- The simulation will stop after executing the last instruction in a snippet.

The file *regfile.py* implements a register file, a collection of registers required to run the simulated processor; to better understand how the *regfile.py* function, you can run it separately like below:

- python3 regfile.py

The file *instload.py* loads a snippet into a list to make the simulation of instruction load into the *processor.py* simple for you; to better understand how the *instload.py* function, you can run it separately like below:

- python3 instload.py test.s

By right, you do not need to change anything in *simulate.py* which initializes the simulations and call the *processor.py*; i.e.

- All your work should be made in *processor.py*.
- Changes to the skeleton and any files which eventually lead to your results not being able to match the requirement in the marking scheme will not be excused.

Before the simulation starts, the initial states and values of the register file are initialized via *register.ini* while the cache and memory settings are defined in *memory.ini*. You are free to change the initial values in order to extensively test your implementation.

Simulation of the 2-level cache organization has to strictly follow the variables defined in *memory.ini*. The assessment will vary those variables to check for your understanding of cache hierarchy. The inclusion of the cache requires you to extend the MEM stage to multiple cycles. The simulation of the caches must include the extraction of the block addresses, the identification of the set entries with associativity, the use of the LRU replacement policy, the decision of the cache hit and miss at different cache levels, and the measurement of the total latencies for all feasible scenarios. The simulated memory is set to all zeros before the first instruction is executed.

Marking scheme: (Total 20 marks)

1. The simulation can run to completion without error messages and the output complies to format described in the sample output. (2 marks)
2. Correct simulation of activities and use of temporary storages in ID, EXE, MEM, and WB stages of the multi-cycle processor implementation. (8 marks)
3. Correct extension of MEM stage to include cache hierarchy in the simulator. (2 marks)
4. Correct cache lookup and hit/miss decision. (2 marks)
5. Correct implementation of LRU replacement policy. (2 marks)
6. Correct calculation of the overall memory access latency. (2 marks)
7. Correct simulation of the cache hierarchy with different cache settings. (2 marks)

Submission

- The deadline is **6pm on 18th September 2020 (Friday)**
- **NO LATE submission** is accepted.
- One submission for one group is sufficient.
- Do NOT submit input files.
- Only submit your solution of **processor.py** by emailing them to laiac@utar.edu.my with email subject equal to
  - **UEEA2283 2020 A3 G[***group number***]**
  - e.g. UEEA2283 2020 A3 G[13] for group #13.
- Failure to conform to the naming format above will result in **10% deduction of your total mark**