Dept. of Computer Science and Engineering
IIT Delhi
COL216 : Minor Exam
II Semester 2020-2021

Release date: 20 March 2021 (Saturday), 2:00 PM
Submission deadline: 21 March 2021 (Sunday), 11:59 PM

General Instructions

1. The assignment substitutes for the Minor exam, and will be done **individually**.
2. Demos (online/phone) would be held for all the lab assignments.
3. Adopting any unfair means may lead to -MAX marks and a report to the Institute Disciplinary Committee.
4. This assignment has a total of 20% overall weightage.
5. **Submissions are due at 11:59 PM on 21/03/2021.** The system will accept submissions until 12:30 AM on 22/03/2021 to allow for any connectivity issues, and will close after 12:30 AM**.**

Submission instructions

- Prepare a small write-up (1-2 pages) on the approach taken to solve the problem along with test cases you have considered. The write-up can consist of handwritten notes.
- Explain the approach along with its strengths and weaknesses.
- Explain the testing strategy.
- Zip the document along with the C++ file and test cases and submit at the Moodle submission link.

Problem Statement: MIPS simulator with DRAM timing model

A basic MIPS interpreter handling a subset of the ISA is developed by you in Assignment 3. The objective is to enhance it with two new features, making it a MIPS simulator.

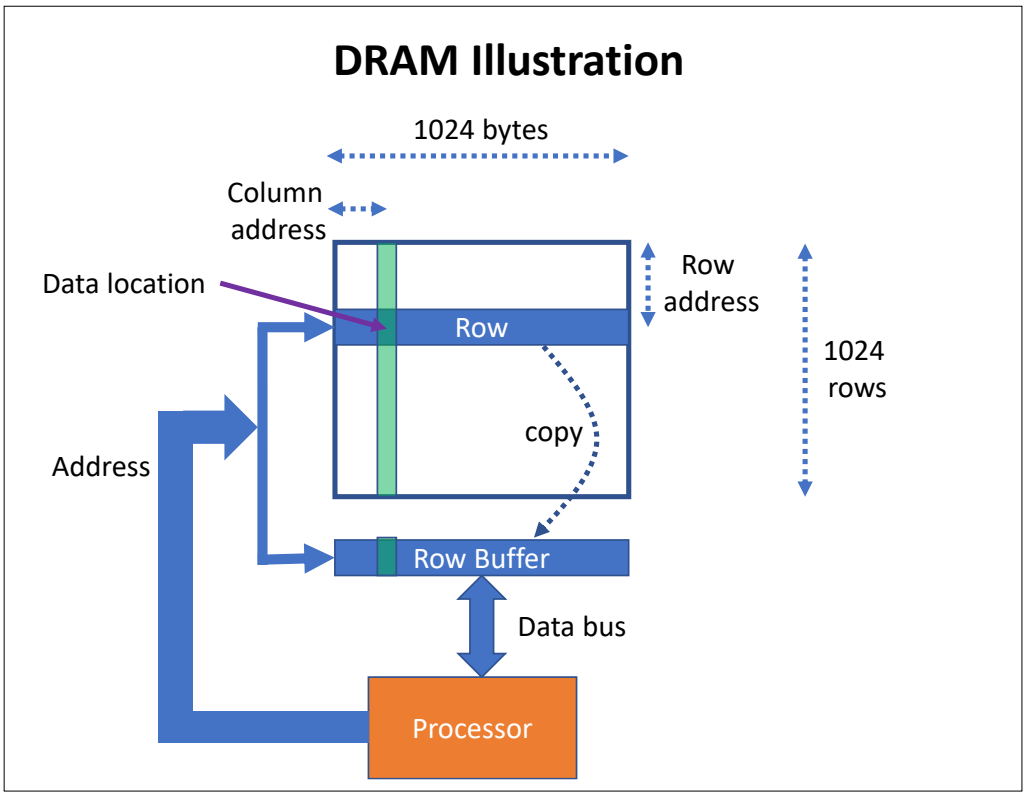
1. Develop a model for the Main Memory and integrate into the basic interpreter.
2. The memory access should be *non-blocking* (subsequent instructions don't always wait for the previous instructions to complete).

Input:

1. MIPS assembly language program (as text file, NOT machine instructions).
2. DRAM timing values ROW_ACCESS_DELAY and COL_ACCESS_DELAY in cycles (as command line arguments). Typical values could be 10 cycles and 2 cycles respectively.

Main Memory Timing Characteristics

You will model the access times corresponding to a simplified version of the Dynamic Random Access Memory (DRAM). Such a memory can be thought of as a 2-dimensional array of ROWS and COLUMNS (see picture below). Our memory of size 2^{20} bytes consists of 1024 rows, with 1024 columns (bytes) in each row. A single Read/Write in a lw/sw instruction corresponds to accessing a 32-bit (4 byte) value from the DRAM. Within a row, a given piece of data is located at a column offset, so a memory address could be thought of as consisting of a ROW ADDRESS and COLUMN ADDRESS.



A READ operation (corresponding to an **lw** instruction) works as follows.

1. Activate the ROW corresponding to this address by COPYING the row to a ROW BUFFER at the bottom of the structure. Time required for this operation: ROW_ACCESS_DELAY.
2. Copy the data at the column offset from the row buffer to the REGISTER. Time for this operation: COL_ACCESS_DELAY.

For subsequent READ operations, we first check whether the corresponding row is already located in the ROW BUFFER. If yes, then just copy the data at the column offset to the data bus after time COL_ACCESS_DELAY (no need for copying the row into the row buffer, since it is already present). However, if the row is different from the one currently located in the row buffer, then:

1. First copy the row buffer back to its row (Time for this operation: ROW_ACCESS_DELAY).
2. Access the data from a new row going through the READ operation protocol described earlier.

A WRITE operation (corresponding to an **sw** instruction) works as follows. If the row is present in the row buffer, update the row buffer data (at the column offset) in time COL_ACCESS_DELAY. If the row is not present, then:

1. Copy the row buffer back (Time for this operation: ROW_ACCESS_DELAY).
2. Copy the new row to the row buffer (Time for this operation: ROW_ACCESS_DELAY).
3. Update the data in the row buffer in time COL_ACCESS_DELAY.

Non-blocking Memory Access

In implementing the DRAM timing above, we notice that the processor slows down significantly because of DRAM accesses. Try reducing the execution time by observing that it may be possible to

execute the next instruction (or next few instructions) even if a lw/sw instruction has not completed. When is it safe to do so? Explain, and implement this feature in your MIPS simulator.

Note: Try to first implement a relatively simple solution before trying out more complex ideas.

Output:

1. At every clock cycle, print the clock cycle number and all activity in that cycle, such as:
    a. Print executed instruction
    b. Modified registers, if any (register number and new value)
    c. Modified memory locations, if any (memory location and new value (4 bytes))
    d. Activity on the DRAM, if any (memory location, row buffer updates)
2. After execution completes, print the relevant statistics such as:
    a. Total execution time in clock cycles
    b. Number of row buffer updates

Assume the following:

1. Instructions may be in memory, but are not fetched from DRAM (consider instruction access delay to be zero). Only lw/sw instructions result in DRAM accesses.
2. Use the same architectural and ISA assumptions as in Assignment 3.

Marks Distribution:

1. Part 1 (DRAM implementation): 10 marks
2. Part 2 (Non-blocking memory): 10 marks

Test cases: Please refer to the following:

1. Presentation having the detailed description of sample test cases.
2. Test cases uploaded in a zip folder.