COL216 Minor - MIPS Simulator

Karan Aggarwal

2019CS10699

Approach

1. A single C++ file takes the file name, mode, ROW_ACCESS_DELAY, COL_ACCESS_DELAY as command-line inputs and executes Part 1 or Part 2 according to the mode. (Execution instructions are explained in the README file)
2. If an lw instruction is encountered, assume $r is the register associated with it. For the next few clock cycles (depending on the delay period and row buffer changes), if $r is mentioned in any of the instruction, its execution is paused. It is executed after DRAM finishes its process.
3. If $r is not encountered, the instructions (other than lw or sw) get executed parallelly with DRAM.
4. If an sw instruction is encountered, all instructions (other than lw or sw) are executed parallelly with DRAM.
5. During DRAM execution of lw or sw, if another instruction of type lw or sw is encountered, it is paused and waits for the previous instruction to complete.
6. At the end of execution, the statistics of execution are printed.

Design Decisions:

1. Data structures used:
   - Registers are simulated as C++ maps from strings to integers (Register name => value stored)
   - DRAM is simulated as a 2-D vector of size 1024 × 1024
   - Buffer is simulated as a 1-D vector of size 1024
   - Allowed commands are stored as a C++ set of strings
   - Labels are stored as a map from strings to integers (Label name => instruction number)
   - Instruction set is stored as a dynamic vector whose size is determined while scanning the input
2. While storing integers to RAM, the 32 bits are split into 4 parts of 8 bits (1 byte) each and stored separately in consecutive indices of the corresponding DRAM row.
3. After the execution is complete, the row from the buffer is added back to the memory.

Special care was taken for the following cases:

- All syntax errors are treated as compile-time errors and the program execution is aborted for these cases.
- All errors in the command line input (described in the Testing) are compile-time errors and the program execution is aborted for these cases.
- When an invalid memory address is encountered, the instruction is skipped and the next instruction is executed. This shows a warning to the user and execution is not aborted.
- The program works with all original commands of Assignment 3.
- Inputs with zero delay are also handled.

Strengths:

- This implements Non Blocking Memory and reduces the number of clock cycles by parallelly carrying out tasks of DRAM with processor.

- I have simulated the execution exactly how the Chip processes take place - Buffer arrays are updated at the required times, and execution of dependent instructions is paused accordingly.
- This is a concise implementation and is extremely less prone to errors.
- All possible syntax, run-time and user input errors are handled and reported.
- The program prevents an infinite loop by automatically stopping after 1,000,000 instructions are executed.
- The C++ execution is extremely fast and is $O(n)$ in the number of instructions.
- Extreme care is taken to ensure all valid user inputs are executed.

Weaknesses:

- I have not dome a queue implementation of the DRAM processes. This makes this implementation simulate slightly more clock cycles than the queue implementation.
- The code executes a small subset of the entire MIPS command set.

Testing

Extensive testing has been done over various files and the test files(.asm) have been attached in the test/ directory. These cases also include all edge cases for Part 2.

- test_sample.asm: Sample test given by instructors.
- test1.asm: This is a complete MIPS program with and tests the working of all commands, and DRAM cases. Refer file for more info.
- test2.asm: Execution is not interrupted when the register of lw is not encountered.
- test3.asm: Execution is not interrupted when the same register as sw is encountered.
- test4.asm: Execution is interrupted when the register of lw is encountered.
- test5.asm: lw Execution interrupted by lw.
- test6.asm: lw Execution interrupted by sw.
- test7.asm: sw Execution interrupted by lw.
- test8.asm: sw Execution interrupted by sw.
- test9.asm: Multiple lw and sw commands.
- test10.asm: Invalid memory address accessed.

Correctness of parameters passed in the command line is also checked:

- If the input file name is incorrect, execution is aborted.
- If the mode is incorrect, execution is aborted.
- If the ROW_ACCESS_DELAY or COL_ACCESS_DELAY are not integers / negative integers / do not fit in 16 bits.