COL216: Assignment 4

Divyanka Chaudhari (2019CS50429)
Pranay Gupta (2019CS10383)

Idea

We implement re-ordering of DRAM instructions so that there are the least amount of row/buffer updates. This is done after taking care of the dependencies that arise during looking ahead in the code.The dependencies are seen if there are instructions that make use of registers in which value is being loaded or if we try to access a register that has a dependency from an instruction, We have also modified the code to accommodate the test-cases in the form they're given.

Implementation

Over the previous implementation, we have implemented 2 new functions. They are `findNextRequests` and `efficientProcess`.

`findNextRequests` looks into the next instructions and keeps a track of the memories and registers. If it encounters any instructions using busy memory or registers, it breaks. In this manner, it stores the DRAM instructions which can be safely re-ordered.

We pass the stored instructions in `efficientProcess` to process them recursively in an efficient manner. We run through a for loop several times so that only the DRAM instructions which do not alter row buffer are executed first.

When we encounter `sw` or `lw` in our main while loop, it jumps into the functions above. If we encounter any non-DRAM instruction, we keep a track of the memory and register used by it in different vectors. We then implement a check in the main while loop to execute them safely after our safe DRAM instructions are encountered.

At any point while moving through instructions if a non-safe/busy register/memory is encountered, we break and wait.

Test cases

All the input test cases are present in the folder `testcases` and their corresponding outputs are present in the folder `testcaseoutputs`.

1. Testcase 1: Basic implementation of the code that contains a few `sw` and `lw` commands along with a few commands in between.
2. Testcase 2: Checking that the branch change works fine.
3. Testcase 3: Executing the code containing multiple store word and load word commands.
4. Testcase 4: Basic implementation with a few store word and load word commands.
5. Testcase 5: Checking the branch change along with store word and load word commands.