Assignment 4

Krishnakant Dharekar 2018CS10351
Pushpendra singh Rana 2018CS10371

Approach :

I have a DRAM memory of size 1024*1024 and a row buffer of size 1024.
Now what we do is whenever we encounter multiple lw and sw operations on consequtive lines we try to minimize the count of cycles required to execute all those instructions.
Like we store all the consequtive lw and sw instructions in an queue and try to minimize the number of cycles required to execute them by executing maximum possible number of these instructions.Those memory instructions that can be executed by the memory buffer that is already in use which can be executed without violation of dependencies i.e which are independent of previous instruction that appear before them in the instruction queue.
The possible cases for
1) if the memory location that is used by this instruction is also used by some other instruction in the queue which appears before it in the queue.
2) if the register used by this instruction is also used by some previous instruction in the queue then these instructions are considered as dependent.

Streanth :
In this we have tried to minimize the number of cycles that are required for execution of lw and sw instructions.
We calculated that most of the time that is required is for filling the buffer that is moving the memory elements from the memory to memory buffer so we have tried to minimize these transections .
Now what happens is if there are multiple consequtive lw and sw instructions then we try to inplement maximum possible number of lw or sw instructions that can be executed on the same buffer without changing the buffer. So this saves huge amount of time during execution.

Weakness :
we have only handeled the case when there are multiple sw and lw instructions on the consecutive instruction lines but it can also be possible that lw and sw instructions are not consecutive in that case my approach taken does not bear fruits.

Input format:
first compile the file using

```
g++ assign4.cpp
```

then run

```
./a.out filename ROW_ACCESS_DELAY COLUMN_ACCESS_DELAY
```

output format:
every time there is a memory access operation I print it.
Every time there is some change in registter value I output the whole register file.
Everytime there is change at some memory address . I output the change.
And in the end I output the total cycles required in execution and the number of row buffer updates.