DRAM IMPLEMENTATION

-Vivek Choudhary (2019CS10413)

PART-A

Using Assignment3 simulator with slight modification for printing given format on console and removing extra stuffs.

"sw":

- row finding done by dividing memory position by 1024
- col finding done by mod(%) memory position by 1024
- if buffer row is used for the first time then increment cycle count by O(row_delay+col_delay)
- if buffer row is not changing O(col_delay)
- if buffer row is new than last one then increment cycle by O(2*row_delay+col_delay)

"lw":

- same as "sw" , difference only in printing style

*Input can be in this form 1004($t1) as well as 1004.

PART-B

To increase efficiency or to reduce the number of cycles in executing the code we will use Non-Blocking technique. And will follow the given question statement.

So, like when there is lw/sw it will store the address which will be updated by this lw/sw and this process will take O(ROW_ACCESS_DELAY+COL_ACCESS_DELAY) cycles. Hence if line next to this or above have no dependence on it then waiting for that line execution is time waste because we can execute both parallelly.

Using this concept I made a 2-d vector("selector") which will store, which "cycle_count" will be printed on console. Selector{line number, checker_flag, cycle_count, span} : line number will tell which operation to be looked, checker_flag will tell whether parallel work is going on or individual, cycle_count will tell present cycle number, span will tell the span of present lw/sw. So, once this whole process is done program will start printing on console in their respective format. Skipping all other things because that all has been explained in Assignment3 and in Part A.

*Input format only of type 1004 in sw/lw.

*Printing format is as given in sample output along with total number of cycles, count of buffer update, instruction size in memory.

Strengths:

- Made simulator more efficient.
- Many strengths can be seen by running program on uploaded test cases as those test cases have covered all boundary cases as well as good input.

Weakness:

- Using some extra space to store information in “selector”.
- If sw’s/lw’s work is done and it’s parallel execution is not dependent on it then console will not show when did sw/lw stopped.
- In Part B we can only take input in “int” format not in “104($s0) format.
- Only looking for next lw/sw or address match or end of delay to stop that lw/sw’s line work, there may be the case in which this is not sufficient. And these can be in inserted image.