COL216: ASSIGNMENT – 1

Student 1: Bommakanti Venkata Naga Sai Aditya **Entry No. 1:** 2019CS50471

WRITE-UP

Input:

Input to our program is a text file with MIPS instructions. Input file is given through command line. Execute using the following instruction:

Part-1:

```
./2019CS50471_Minor_1 <path to inputfile> <ROW_ACCESS_DELAY> <COL_ACCESS_DELAY>
```

Part-2:

```
./2019CS50471_Minor_2 <path to inputfile> <ROW_ACCESS_DELAY> <COL_ACCESS_DELAY>
```

Approach:

Note: All the possible errors are handled in Assignment-3. Same code is used in this assignment. So, all possible errors are handled in this assignment.

PART-1-DRAM IMPLEMENTATION

Instructions are also stored in DRAM and data is stored after the instructions memory. The program executes all the instructions (all the 10 instructions) sequentially and if the instructions are not lw/ sw, it executes in once clock cycle. If the instructions are sw/ lw, then the program checks the row buffer and if the it has the row that we required, then it updates the row buffer, which takes COL_ACCESS_DELAY clock cycles. If not, it copys the row buffer into memory and activates the row that we required, which takes 2*ROW_ACCESS_DELAY + COL_ACCESS_DELAY clock cycles. If row buffer is empty, we activate the row we required, which takes ROW_ACCESS_DELAY + COL_ACCESS_DELAY clock cycles.

Only positive values(>0) are allowed for ROW_ACCESS_DELAY, COL_ACCESS_DELAY. Program raises an error if this condition is violated.

At the end of all the instructions, the row buffer is copied into memory which takes ROW_ACCESS_DELAY clock cycles.

PART-2-NON BLOCKING MEMORY

In the above part, program slows down during DRAM access. So, to minimize execution time we can perform instructions next to lw/sw which are independent of lw/sw instruction.

All the 10 instructions can be used for this part too.

Strengths:

Let reg1, reg2 be the registers used in lw/sw instruction

1) When the instructions following lw/sw donot use any of these registers, then the program executes that instruction.

2) Even if the instructions following lw/ sw uses reg1, reg2 such that the values in reg1, reg2 are not updated, the program executes that instruction. For example:

I1: sw $t0, 0($s0)

I2: slt $t1, $t0, $t2

Here, I2 uses the register $t0, but the value in $t0 won't change because of this instruction. So, this program executes I2.

3) If there are many instructions that are independent of lw/sw but instructions are executed parallelly until the DRAM cycle is finished.

4) It can execute the instructions following lw/ sw as many as possible i.e., the program executes all instructions next to lw/sw if they satisfies any of above two conditions and are not lw/sw, until the DRAM cycle is finished.

Weakness:

1) If a second lw/sw instruction appears after first lw/ sw, program waits for the completion of first lw/ sw instruction and then proceeds to the second lw/ sw instruction. Instructions after second lw/ sw can be independent of both lw/ sw instructions. But these can't be executed.

Output:

At every clock cycle period, the program prints the instruction executed, modified registers, modified memory addresses and DRAM activity in the clock cycle period. At the end, the program prints total no of clock cycles, data stored in the memory and total row buffer updates.

TESTING

Syntax Errors: Syntax errors are tested on this code in Assignment-3

General Case: All the instructions follow the given syntax as mentioned above.

I have found the following possible corner cases for part-2:

If entered value of ROW_ACCESS_DELAY & COL_ACCESS_DELAY are less tham or equal to zero, then the program exits.

Test Cases:

Testing Strategy: I computed the output manually and verified it with the output

Testcase-1:

Instructions used: addi, add, sw, lw

All lw/sw instructions are consecutive. So, there won't be any decrease in clock cycles when sent to part-1 and 2.

Testcase-2:

Instructions used: addi, sub, mul, sw, slt

The instructions following sw in line-4 are not using registers used by sw. So, instructions in line 5, 6, 7, 8 are executed along with DRAM cycles.

Testcase-3:

Instructions used: addi, add, sw, lw

The instructions following sw in line-5 are not using registers used by sw. So, instructions in line 6, 7 are executed along with DRAM cycles. When program finds lw in line 8, it completes the DRAM cycle and then start lw instruction. The instructions following lw in line-8 are not using registers used by lw. So, instructions in line 9, 10 are executed along with DRAM cycles.

Testcase-4:

Instructions used: addi, add, sw, lw

The instructions following sw in line-5 are using registers used by sw but they are not affecting the value of those registers. So, instructions in line 6, 7 are executed along with DRAM cycles. When program finds lw in line 8, it completes the DRAM cycle and then start lw instruction. The instructions following lw in line-8 are using registers used by lw but they are not affecting the value of those registers. So, instruction in line 9 is executed along with DRAM cycles.

Testcase-5:

Instructions used: addi, add, sw, lw, j, beq, sub, mul

The instructions following sw in line-5 are using registers used by sw but they are not affecting the value of those registers. So, instructions in line 6, 7, 8 are executed along with DRAM cycles. When program finds lw in line 9, it completes the DRAM cycle and then start lw instruction. The instructions

following lw in line-9 are using registers used by lw but they are not affecting the value of those registers. So, instructions in line 10,11,12 are executed along with DRAM cycles.

Testcase-6:

Instructions used: addi, add, sw, lw

The instructions following sw in line-5 are using registers used by sw and they are affecting the value of those registers. So, DRAM cycle is completed and then the program goes to execute line-6. The instructions following lw in line-7 are using registers used by lw and they are affected by the value of those registers. So, instructions in line 8 are executed after DRAM cycle is executed.

Testcase-7:

Instructions used: addi, add, sw, lw, beq

The instruction following sw in line-5 is lw. So, DRAM cycle is completed and then the program goes to execute line-6. The instruction in line-7 following lw in line-6 is not using registers used by lw and they are not affected by the value of those registers. So, it is exectued along with DRAM cycle. But instructions in line -8, 9, 10 are affected by the values from lw instruction. So, instructions in line 7, 8, 9, 10 are executed after DRAM cycle is executed.

Testcase-8:

Instructions used: addi, mul, sw, lw, bne

The instructions following sw in line-5 are using registers used by sw and they are affecting the value of those registers. So, DRAM cycle is completed and then the program goes to execute line-6. The instructions following lw in line-7 are using registers used by lw and they are affected by the value of those registers. So, instructions in line 8,9 are executed after DRAM cycle is executed.

Testcase-9:

Instructions used: addi, add, sw, lw

The instructions following sw in line-5 are using registers used by sw and they are not affecting the value of those registers. So, they can be executed along with DRAM cycle. But once the DRAM cycle is completed the next instructions are executed individually.

Used ROW_ACCESS_DELAY = 10; COL_ACCESS_DELAY = 2 for sample outputs.