COL-216
ASSIGNMENT-4
MEMORY REQUEST ORDERING

Made By

Dishant Dhiman *2019CS10347*

R Hari Shankar *2019CS10386*

DESIGN DESCRIPTION

The DRAM implementation made in Minor (by Dishant Dhiman) has been used for implementing the Ordering of Memory request . The memory requests are called by the “lw” and “sw” command .

For increasing the performance of the simulator which initially executed all DRAM commands sequentially which now is faster (takes less clock cycles if possible) as all the memory request commands are stored in a vector named dramRequests which contains pair of memory address and PC value as data which allows us to organise the Memory request in an order such that the need to change the row buffer is reduced significantly which directly means less clock cycles due to less row access delays.

A variable is kept to keep track if DRAM memory is active or not.

The exhausted label is used for implementing the DRAM requests in an organised manner which includes changing the memory or register values and updating clock cycles, row buffer etc.

Once these requests are executed then its data is removed from the vector.

We have ensured that this ordering doesn’t hamper with the actual result which we would have initially got using the simulator for DRAM memory developed in minor.

The execution of the other functions are similar/same as those implemented before.

STRENGTHS OF THIS IMPLEMENTATION

The memory request ordering allows us to execute sw or lw commands in less clock cycle than the initial DRAM implementation as these commands are executed such that the memory changes are made by performing changes in memory having same row buffer in order to reduce clock cycles taken for the row access delay thus enhancing the performance of the simulator.

The ordering also helps to reduce the number of row buffer updates.

These reduction in running time can be seen from the figures given in the testcases.

“dramActive” variable also keeps tabs of the DRAM so that if another DRAM command comes it will wait for the first one to be executed . This prevents cross channelling of the DRAM memory .

The memory changes are also tracked using vector modifiedMemory which are printed at the end of execution.

The check on the row buffer helps to reduce the runtime.

The dramRequests vector keeps an efficient track of the various memory element changes in process.

All the changes in memory, register and clock are given if intermediate values need to be checked.

The output follows the same pattern as given as expected output in the testcase pdf.

Our program works on all the required test formats (Given in preparatory material).

WEAKNESS (SHORTCOMINGS) OF THIS IMPLEMENTATION

If all the commands require access to different rows or same rows of memory then this method will have same clock cycle as the initial implementation.

The program will wait for the previous DRAM access to be executed before the next command can be executed.

Multiple row buffers can be used to increase efficiency.

Non-blocking memory is not implemented which could have increased the efficiency.

HOW TO RUN THE PROGRAM

1) Set your directory to the directory in which the program is present using your terminal/console.
2) Now type g++ nonBlocking.cpp -o interpreter.out
3) Now type ./interpreter.out <input file> <row access delay> <column access delay>
   Eg) ./interpreter.out input.txt 10 2
4) The input file is the file that contains the MIPS instruction.
5) The output will be shown in your console.

OUTPUT FORMAT

If the input is of correct format then the output will be displayed else an error message will be displayed stating that the input format is incorrect.

FORMAT:

All the executed commands with PC values.

Action performed in each cycle

After all the instruction are evaluated the total number of clock cycles, Number of row buffer updates and number of times each instruction is executed is printed in the console.

The memory contents and the register contents are printed after this.

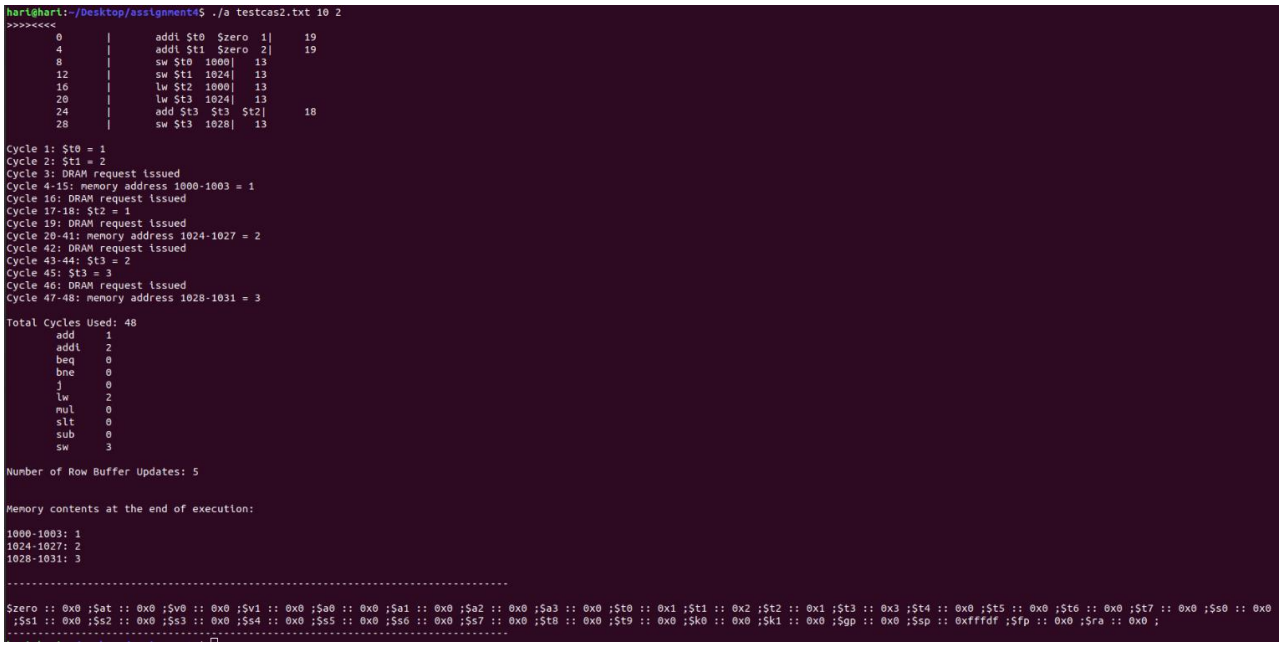

Fig: Output for a testcase

TESTING STRATEGY

To check the correctness of our code as an interpreter of the mips assembly language we have used exhaustive test cases so that our program runs correctly in all cases (including corner cases).

Test Cases:

- When correct format of file is given using only the add ,sub ,mult ,beq ,bne ,slt ,lw ,sw ,addi ,j instructions .
- With multiple lw,sw commands with differing row buffers.
- With multiple lw,sw commands with same row buffers but other instructions in between.

Invalid Input Cases:

- When input file is not found.
- When input file is empty.
- Incorrect register name is given.
- Syntax error for various instruction.
- Memory used more than 2^{20} bytes.
- When instruction other than add ,sub ,mult ,beq ,bne ,slt ,lw ,sw ,addi ,j is used.

All the testcases used to check correctness of program are given in the file testcases.

The expected output are verified both manually and using QtSPIM MIPS.

```
hari@hari:~/Desktop/assignment4$ ./a testing3.txt 10 2
>>>><<<<
        0       |       addi $t0  $zero  1|     19
        4       |       sw $t0  1000|   13
        8       |       sw $t0  1024|   13
        12      |       lw $t0  1000|   13
        16      |       add $t0  $t0  $t1|      18
        20      |       lw $t1  1024|   13

Cycle 1: $t0 = 1
Cycle 2: DRAM request issued
Cycle 3-14: memory address 1000-1003 = 1
Cycle 15: DRAM request issued
Cycle 16-17: $t0 = 1
Cycle 18: DRAM request issued
Cycle 19-40: memory address 1024-1027 = 1
Cycle 41: $t0 = 1
Cycle 42: DRAM request issued
Cycle 43-44: $t1 = 1

Total Cycles Used: 44
        add     1
        addi    1
        beq     0
        bne     0
        j       0
        lw      2
        mul     0
        slt     0
        sub     0
        sw      2

Number of Row Buffer Updates: 4


Memory contents at the end of execution:

1000-1003: 1
1024-1027: 1

--------------------------------------------------------------------------------

$zero :: 0x0 ;$at :: 0x0 ;$v0 :: 0x0 ;$v1 :: 0x0 ;$a0 :: 0x0 ;$a1 :: 0x0 ;$a2 :: 0x0 ;$a3 :: 0x0 ;$t0 :: 0x1 ;$t1 :: 0x1 ;$t2 :: 0x0 ;$t3 :: 0x0 ;$t4 :: 0x0 ;$t5 :: 0x0 ;$t6 :: 0x0 ;$t7 :: 0x0 ;$s0 :: 0x0
 ;$s1 :: 0x0 ;$s2 :: 0x0 ;$s3 :: 0x0 ;$s4 :: 0x0 ;$s5 :: 0x0 ;$s6 :: 0x0 ;$s7 :: 0x0 ;$t8 :: 0x0 ;$t9 :: 0x0 ;$k0 :: 0x0 ;$k1 :: 0x0 ;$gp :: 0x0 ;$sp :: 0xfffe7 ;$fp :: 0x0 ;$ra :: 0x0 ;
--------------------------------------------------------------------------------
```

```
hari@hari:~/Desktop/assignment4$ ./a testcase.txt
        0       |       addi $s0  $zero  1000|  22
        4       |       addi $s1  $zero  2500|  22
        8       |       addi $t0  $zero  1|     19
        12      |       addi $t1  $zero  2|     19
        16      |       addi $t2  $zero  3|     19
        20      |       addi $t3  $zero  4|     19
        24      |       sw $t0  0($s0)  #store 1 at location 1000|      41
        28      |       sw $t1  0($s1)  #store 2 at location 2500|      41
        32      |       sw $t2  4($s0)  #store 3 at location 1004|      41
        36      |       sw $t4  4($s1)  #store 4 at location 2504|      41
        40      |       lw $t5  0($s0)| 15
        44      |       lw $t6  0($s1)| 15
        48      |       lw $t7  4($s0)| 15
        52      |       lw $t8  4($s1)| 15

Cycle 1: $s0 = 1000
Cycle 2: $s1 = 2500
Cycle 3: $t0 = 1
Cycle 4: $t1 = 2
Cycle 5: $t2 = 3
Cycle 6: $t3 = 4
Cycle 7: DRAM request issued
Cycle 8-19: memory address 1000-1003 = 1
Cycle 20: DRAM request issued
Cycle 21-22: memory address 1004-1007 = 3
Cycle 23: DRAM request issued
Cycle 24-25: $t5 = 1
Cycle 26: DRAM request issued
Cycle 27-28: $t7 = 3
Cycle 29: DRAM request issued
Cycle 30-51: memory address 2500-2503 = 2
Cycle 52: DRAM request issued
Cycle 53-54: memory address 2504-2507 = 0
Cycle 55: DRAM request issued
Cycle 56-57: $t6 = 2
Cycle 58: DRAM request issued
Cycle 59-60: $t8 = 0

Total Cycles Used: 60
        add     0
        addi    6
        beq     0
        bne     0
        j       0
        lw      4
        mul     0
        slt     0
        sub     0
        sw      4

Number of Row Buffer Updates: 6
```

```
hari@hari:~/Desktop/assignment4$ ./a testcase1.txt
        0       |       addi $s0  $zero  1000|  22
        4       |       addi $s1  $zero  0|     19
        8       |       addi $s2  $zero  10|    20
        12      |       addi $t1  $zero  0|     19
        16      |       addi $t1  $t1  1|       17
        20      |       sw $t1  0($s0)| 15
        24      |       addi $s0  $s0  4|       17
        28      |       addi $s1  $s1  1|       17
        32      |       slt $s3  $s1  $s2|      18
        36      |       bne $s3  $zero  initloop|       25
        40      |       addi $s0  $zero  1000|  22
        44      |       addi $s1  $zero  0|     19
        48      |       addi $s3  $zero  0|     19
        52      |       addi $s2  $zero  9|     19
        56      |       lw $t0  0($s0)| 15
        60      |       addi $s0  $s0  4|       17
        64      |       lw $t1  0($s0)| 15
        68      |       add $t2  $t0  $t1|      18
        72      |       sw $t2  0($s0)| 15
        76      |       addi $s1  $s1  1|       17
        80      |       slt $s3  $s1  $s2|      18
        84      |       bne $s3  $zero  sumloop|        24

Cycle 1: $s0 = 1000
Cycle 2: $s1 = 0
Cycle 3: $s2 = 10
Cycle 4: $t1 = 0
Cycle 5: $t1 = 1
Cycle 6: DRAM request issued
Cycle 7-18: memory address 1000-1003 = 1
Cycle 19: $s0 = 1004
Cycle 20: $s1 = 1
Cycle 21: $s3 = 1
Cycle 22: Jumping to address = 16
Cycle 23: $t1 = 2
Cycle 24: DRAM request issued
Cycle 25-26: memory address 1004-1007 = 2
Cycle 27: $s0 = 1008
Cycle 28: $s1 = 2
Cycle 29: $s3 = 1
Cycle 30: Jumping to address = 16
Cycle 31: $t1 = 3
Cycle 32: DRAM request issued
Cycle 33-34: memory address 1008-1011 = 3
Cycle 35: $s0 = 1012
Cycle 36: $s1 = 3
Cycle 37: $s3 = 1
Cycle 38: Jumping to address = 16
Cycle 39: $t1 = 4
Cycle 40: DRAM request issued
Cycle 41-42: memory address 1012-1015 = 4
Cycle 43: $s0 = 1016
Cycle 44: $s1 = 4
Cycle 45: $s3 = 1
```

```
Cycle 271: DRAM request issued
Cycle 272-273: $t0 = 45
Cycle 274: $s0 = 1036
Cycle 275: DRAM request issued
Cycle 276-277: $t1 = 10
Cycle 278: $t2 = 55
Cycle 279: DRAM request issued
Cycle 280-281: memory address 1036-1039 = 55
Cycle 282: $s1 = 9
Cycle 283: $s3 = 0
Cycle 284: No jump

Total Cycles Used: 284
        add     9
        addi    56
        beq     0
        bne     19
        j       0
        lw      18
        mul     0
        slt     19
        sub     0
        sw      19

Number of Row Buffer Updates: 23


Memory contents at the end of execution:

1000-1003: 1
1004-1007: 3
1008-1011: 6
1012-1015: 10
1016-1019: 15
1020-1023: 21
1024-1027: 28
1028-1031: 36
1032-1035: 45
1036-1039: 55
1004-1007: 3
1008-1011: 6
1012-1015: 10
1016-1019: 15
1020-1023: 21
1024-1027: 28
1028-1031: 36
1032-1035: 45
1036-1039: 55

--------------------------------------------------------------------------------

$zero :: 0x0 ;$at :: 0x0 ;$v0 :: 0x0 ;$v1 :: 0x0 ;$a0 :: 0x0 ;$a1 :: 0x0 ;$a2 :: 0x0 ;$a3 :: 0x0 ;$t0 :: 0x2d ;$t1 :: 0xa ;$t2 :: 0x37 ;$t3 :: 0x0 ;$t4 :: 0x0 ;$t5 :: 0x0 ;$t6 :: 0x0 ;$t7 :: 0x0 ;$s0 :: 0
x40c ;$s1 :: 0x9 ;$s2 :: 0x9 ;$s3 :: 0x0 ;$s4 :: 0x0 ;$s5 :: 0x0 ;$s6 :: 0x0 ;$s7 :: 0x0 ;$t8 :: 0x0 ;$t9 :: 0x0 ;$k0 :: 0x0 ;$k1 :: 0x0 ;$gp :: 0x0 ;$sp :: 0xfffa7 ;$fp :: 0x0 ;$ra :: 0x0 ;
--------------------------------------------------------------------------------
```