CS224

Section No: 05

Spring 2021

Lab No: 06

Turgut Alp Edis / 21702587

1.

No	Cache Size KB	N Way cache	Word size in bits	Block Size (no. of words)	No. of Sets	Tag Size in bits	Index Size (Set No.) in bits	Word Block Offset Size in bits	Byte Offset Size in bits	Block Replacement Policy Needed (Yes/No)
1	8	1	8	8	1					
2	8	2	16	8	1					
3	8	4	16	4	1					
4	8	Full	16	4	1					
9	32	1	16	2	8					
10	32	2	16	2	4					
11	32	4	8	8						
12	32	Full	8	8	1					

2.

a.

Instruction	Iteration No.				
	1	**2**	**3**	**4**	**5**
lw $t1, 0xA4($0)	Compulsory	Hit	Hit	Hit	Hit
lw $t2, 0xA8($0)	Compulsory	Hit	Hit	Hit	Hit
lw $t3, 0xAC($0)	Compulsory	Hit	Hit	Hit	Hit

b.

memory size => 2^{32} bits

length of instruction => $\log_2(2^{32}) = 32$ bits

N = 1, so cache is directly mapped.

Block size => 8 words

Byte offset => 2 bits

Set => 2 bits

Block offset => 1 bit, tag => 32-(2+2+1) = 27 bits

Total cache => (1+27+32+32) * 4 = 168 bits

c.

No multiplexer and OR gate is needed. 1 equality comparator and 1 AND gate is needed.

3.

a.

Instruction	**Iteration No.**				
	1	**2**	**3**	**4**	**5**
lw $t1, 0xA4($0)	Compulsory	Capacity	Capacity	Capacity	Capacity
lw $t2, 0xA8($0)	Compulsory	Capacity	Capacity	Capacity	Capacity
lw $t3, 0xAC($0)	Capacity	Capacity	Capacity	Capacity	Capacity

b.

Memory size: 2^{32} bits, instruction length: 32 bits, cache is 1 since n = 1

Block size: 1 word, byte offset: 2 bits, set: 0 bit, block offset: 0 bits, tag: 32-2=30 bits

Total cache = (1+30+32+1+30+32) * 1 = 126 bits

4.

```
#Prelim Lab 6 - Turgut Alp Edis 21702587
#Matrix implementation in MIPS

.data

firstS: .asciiz "\nEnter the number that represents the functionalities\n"
prom1: .asciiz "1 to Create the matrix with identified size \n"
prom2: .asciiz "2 to access the matrix element \n"
prom3: .asciiz "3 to obtain summation of matrix elements row-major summation \n"
prom4: .asciiz "4 to obtain summation of matrix elements column-major summation \n"
prom5: .asciiz "5 to display desired elements of the matrix by specifying its row and column
member \n"
promC: .asciiz "Enter the column index: "
promR: .asciiz "Enter the row index: "
promRorC: .asciiz "Choose to display row or column (Enter 1 for row, 2 for column): "
prom_size: .asciiz "Enter the one side's size: "
sizze: .asciiz "Enter the matrix size in terms of its dimensions (enter the one side's size): "
sum_msg: .asciiz "The sum is: "
tab: .asciiz "\t"
newLine: .asciiz "\n"


.text
main:
        #prompts for interface menu
        li $v0, 4
        la $a0, firstS
        syscall
        li $v0, 4
        la $a0, prom1
        syscall
        li $v0, 4
        la $a0, prom2
        syscall
        li $v0, 4
        la $a0, prom3
        syscall
        li $v0, 4
        la $a0, prom4
        syscall
        li $v0, 4
        la $a0, prom5
        syscall
        #get the num
        li $v0, 5
```

```
        syscall

        beq $v0, 1, o1
        beq $v0, 2, o2
        beq $v0, 3, o3
        beq $v0, 4, o4
        beq $v0, 5, o5

li $v0, 10
syscall

o1:
        #s0 = beginning address of array, s1 = N, a1 = matrix size
        #For taking size number
        li $v0, 4
        la $a0, sizze
        syscall
        #get the size
        li $v0, 5
        syscall
        move $s1, $v0

        mul $a1, $s1, $s1

        #create array for matrix
        mul $a0, $a1, 4
        li $v0, 9
        syscall

        move $s0, $v0
        #s4 = (constant) address of beginning
        move $s4, $v0
        #t0 = element, t1 = index number
        addi $t0, $0, 1
        addi $t1, $0, 0

        for:
                beq $t1, $a1, exit
                sw $t0, ($s0)
                add $s0, $s0, 4
                add $t0, $t0, 1
                add $t1, $t1, 1
                j for
        exit:
                add $s0, $s4, 0

        j main
```

o2:

```
#col number
li $v0, 4
la $a0, promC
syscall
#get the col num
li $v0, 5
syscall
move $s2, $v0

#row number
li $v0, 4
la $a0, promR
syscall
#get the row num
li $v0, 5
syscall
move $s3, $v0
#Index formula = (j - 1) x N x 4 + (i - 1) x 4

sub $t2, $s2, 1
mul $t2, $s1, $t2
mul $t2, $t2, 4

sub $t3, $s3, 1
mul $t3, $t3, 4

add $t2, $t2, $t3
#Access element
add $s0, $t2, $s0
#Load to a0
lw $a0, ($s0)
#reset the array
add $s0, $s4, 0
#print element
li $v0, 1
syscall

j main
```

o3:

```
#t2 x N x 4 + t0 x 4
#t1 = t0 x 4, t3 = t2 x N x 4, t6 = t3 + t1
#s1 = N, s0 = array address, t2 = i-1, t0 = j-1, t5 = sum
#Reset elements
li $t0, 0
```

```
        #col for loop
        ffor:
                beq $t0, $s1, exit_1
                mul $t1, $t0, 4
                li $t2, 0
                li $t5, 0
        #row for loop
        ffor2:
                beq $t2, $s1, exit_2
                mul $t3, $t2, 4
                mul $t3, $t3, $s1
                add $t6, $t1, $t3
                #access index
                add $s0, $t6, $s0
                lw $t4, ($s0)

                add $t5, $t5, $t4

                add $s0, $s4, 0

                add $t2, $t2, 1
                j ffor2

        exit_2:

                li $v0, 4
                la $a0, sum_msg
                syscall

                move $a0, $t5
                li $v0, 1
        syscall

        li $v0, 4
        la $a0, newLine
        syscall

        add $t0, $t0, 1
        j ffor

        exit_1:
                add $s0, $s4, 0
                j main

o4:
        #Formula and variable is same as o3
        #but t3 is t5, t1 is col num, t0 is row num
```

```
        li $t0, 0
        ffor3:
                beq $t0, $s1, exitffor3
                li $t3, 0
                li $t1, 0

        ffor4:
                beq $t1, $s1, exitffor4
                lw $t2, ($s0)
                add $t3, $t3, $t2
                add $s0, $s0, 4
                add $t1, $t1, 1

                j ffor4

        exitffor4:
                li $v0, 4
                la $a0, sum_msg
                syscall

                move $a0, $t3
                li $v0, 1
                syscall

                li $v0, 4
                la $a0, newLine
                syscall

                add $t0, $t0, 1
                j ffor3

        exitffor3:
                add $s0, $s4, 0
                j main

o5:
        #Display elements with using same method
        #$t1 x N x 4 + $t0 x 4
        li $v0, 4
        la $a0, promRorC
        syscall
        li $v0, 5
        syscall

        beq $v0, 1, row
        beq $v0, 2, column
```

```
j main

row:
        li $v0, 4
        la $a0, promR
        syscall
        li $v0, 5
        syscall

        move $t0, $v0
        sub $t0, $t0, 1

        mul $t0, $t0, 4

        li $t1, 0

for_row:
        beq $t1, $s1, exit_5
        mul $t2, $t1, 4
        mul $t2, $t2, $s1

        add $t3, $t2, $t0#index in array

        add $s0, $t3, $s0
        lw $t4, ($s0)

        move $a0, $t4
        li $v0, 1   #print integer
        syscall

        li $v0, 4
        la $a0, tab
        syscall

        add $s0, $s4, 0
        add $t1, $t1, 1
        j for_row

column:
        #$t0 x N x 4 + $t1 x 4
        li $v0, 4
        la $a0, promC
        syscall
        li $v0, 5
        syscall

        move $t0, $v0
```

```
        sub $t0, $t0, 1
        mul $t0, $t0, 4
        mul $t0, $s1, $t0

        li $t1, 0

for_col:

        beq $t1, $s1, exit_5

        mul $t2, $t1, 4

        add $t3, $t2, $t0

        add $s0, $t3, $s0
        lw $t4, ($s0)

        move $a0, $t4
        li $v0, 1
        syscall

        li $v0, 4
        la $a0, tab
        syscall

        add $s0, $s4, 0
        add $t1, $t1, 1
        j for_col

exit_5:
        j main
```