CS224

Section No: 2

Fall 2019

Lab No: 6

İrem Seven

21704269

Date: 18.12.19

Part 1)

No.	Cache Size KB	N way cache	Word Size	Block size (no. of words)	No. of Sets	Tag Size in bits	Index Size (Set No.) in bits	Word Block Offset Size in bits[1]	Byte Offset Size in bits[2]	Block Replacement Policy Needed (Yes/No)
1	64	1	32 bits	4	2^12	16	12	2	2	NO
2	64	2	32 bits	4	2^11	17	11	2	2	YES
3	64	4	32 bits	8	2^9	18	9	3	2	YES
4	64	Full	32 bits	8	2^0	27	0	3	2	YES
9	128	1	16 bits	4	2^14	15	14	2	1	NO
10	128	2	16 bits	4	2^13	16	13	2	1	YES
11	128	4	16 bits	16	2^10	17	10	4	1	YES
12	128	Full	16 bits	16	2^0	27	0	4	1	YES

Part 2)

a.

Instruction	Iteration No.				
	1	2	3	4	5
lw $t1, 0x4($0)	Compulsory	Hit	Hit	Hit	Hit
lw $t2, 0xC($0)	Compulsory	Hit	Hit	Hit	Hit
lw $t3, 0x8($0)	Hit	Hit	Hit	Hit	Hit

b. Capacity is 8 words. Since its mips we have 32 bits for a data. Then a word is 4 bytes and offset bits are 00 (2 bits).
8(word)*4 (byte)*8(bits) = 256 bits
from log2 block offset is 1 bit.
Each set has 2 word we have total of 8 words. Thus, we have 4 sets, 2 bits for set bits.
Then tag is 27 bits.(others: 2 set bits ,1 block offsset,2 byte offset)
Each set has a tag of 27 and 1 bit V = 28.
28*4(set num) + 256 = 368 bits is the total cache memory size.

c.

one 2to1 mux
one AND gate
one equality comparator

are needed.

Part 3)

a.

Instruction	Iteration No.				
	1	2	3	4	5
lw $t1, 0x4($0)	Compulsory	Capacity	Capacity	Capacity	Capacity
lw $t2, 0xC($0)	Compulsory	Capacity	Capacity	Capacity	Capacity
lw $t3, 0x8($0)	Capacity	Capacity	Capacity	Capacity	Capacity

b.

We have only 1 set and only 1 block size ,so we must have 2 ways to satisfy 2 word capacity.
Also for 2 ways we need 1 bit for U to indicate LRU one.
32 bits have 2 byte offset then we have 30 bits for tag.
30(tag)+1(V)=31 for each way. 62 for both from v and tag.
4(byte)*8(bit) = 32 bits of data for each block. 64 for both ways.
64+62+1(U) = 128 bits is the total cache memory size.

c.

one 2to1 mux
two equality comparators
two and gates
one or gate

are needed.

Part 4)

L1 : 1 cycle, L2: 4 cycle, main: 40 cycle.

AMAT = 1 + 0.20 * (4 + 0.05 * 40) = 2.2 cycles

With 4GHz clock rate we need execution time for 10 ^ 12 inst. :

10^12 * 2.2 * (0.25 * 10 ^ 9) = 550 sec.

Part 5)

```
.data

enterj: .asciiz "Enter j(column)\n"
enteri: .asciiz "Enter i(row)\n"
enterVal: .asciiz "Enter the value of "
endl: .asciiz "\n"
space: .asciiz " "
msgMenu: .asciiz "1)Create Matrix\n2)Display Element\n3)Sum row by row\n4)Sum column
by column\n5)display given row and col\n"
msgSize: .asciiz "Size: "
entercol: .asciiz ". col numbers\n"

.text

 main:

        jal Menu

        j main

        li $v0,10
        syscall

#####################################
Menu:
        la $a0, msgMenu
        li $v0, 4
        syscall # Enter the size of Matrix and write values one by one

        li $v0, 5
        syscall

        addi $s7,$v0,0 #S7 İS CHOİCE

        case1: #CREATE
         addi $s6,$zero,1
         bne $s7,$s6,case2

         jal createMatrix
         addi $s0,$v0,0 #s0 is n
         j Menu

        case2: #display element
         addi $s6,$zero,2
         bne $s7,$s6,case3

        jal display
        j Menu

        case3: #row sum
         addi $s6,$zero,3
         bne $s7,$s6,case4
```

```
        jal rowSum
        j Menu

        case4: #col sum
         addi $s6,$zero,4
         bne $s7,$s6,case5

        jal colSum
        j Menu

        case5: #row col display
         addi $s6,$zero,5
         bne $s7,$s6,default

        jal rowcol
        j Menu

        default:

         jr $ra

###################################

createMatrix:

        la $a0, msgSize
        li $v0, 4
        syscall

        li $v0, 5
        syscall

        addi $t0, $v0,0 #N

        mul $t1,$t0,$t0 #t1 : size

        la $a0,($t1)   #allocate space as size
        li $v0, 9      # now, $v0 has the address of allocated memory
        syscall

        addi $t9,$v0,0 #t9 has base address
        addi $s1,$t9,0 #s1 will hold base adress in whole program

        addi $t2,$0,0 #col
        addi $t3,$0,0 #row

        loopcol:
                beq $t2,$t0,coldone

                addi $a0,$t2,0
                li $v0,1
                syscall

                la $a0,entercol
                li $v0, 4
```

```
                syscall

                looprow:
                        beq $t3,$t0,rowdone
                        li $v0,5
                        syscall
                    sw $v0,0($t9)

                    addi $t9,$t9,4
                    addi $t3,$t3,1

                    j looprow
                  rowdone:
                  addi $t3,$0,0
                  addi $t2,$t2,1
                  j loopcol
                coldone:

                addi $v0,$t0,0 #n
                jr $ra
##################################################
display:
        #i
        la $a0, enteri
        li $v0, 4
        syscall

        li $v0, 5
        syscall
        addi $t0, $v0,0

        #j
        la $a0, enterj
        li $v0, 4
        syscall

        li $v0, 5
        syscall
        addi $t1, $v0,0

        #calculate index
        addi $t1, $t1, -1
        mul $t1, $t1, $s0
        mul $t1, $t1, 4
        addi $t0, $t0, -1
        mul $t0, $t0, 4
        add $t1, $t0, $t1


        add $t0, $s1, $t1

        lw $t2, 0($t0)

        ## display the element
        add $a0, $zero, $t2
        li $v0, 1
```

```
        syscall

        la $a0, endl
        li $v0, 4
        syscall

        jr $ra

##########################################
rowSum:
        addi $t9,$0,0

        addi $t0, $zero, 1 #i
        addi $t1, $zero, 1 #j

        addi $t8, $zero, 0
looprow2: bgt $t0, $s0, outrow2
loopcol2: bgt $t1, $s0, outcol2

        #calculate index of matrix
        addi $t3, $t1, -1
        mul $t3, $t3, $s0
        mul $t3, $t3, 4
        addi $t4, $t0, -1
        mul $t4, $t4, 4
        add $t3, $t4, $t3 # $t0 is the index of element

        # add index to beginning of the array
        add $t4, $s1, $t3 # t4 is the address of element

        # take element from mem
        lw $t5, 0($t4)

        add $t8, $t8, $t5

        addi $t1, $t1, 1

        j loopcol2
outcol2:
        ## display the element
        add $a0, $zero, $t8
        li $v0, 1
        syscall

        la $a0, space
        li $v0, 4
        syscall # space

        addi $t1, $zero, 1
        addi $t0, $t0, 1

        add $t9,$t9,$t8 #t9 is total sum

        addi $t8, $zero, 0

        j looprow2
```

```
outrow2:

        la $a0, endl
        li $v0, 4
        syscall

        addi $a0, $t9,0   #gives total sum
        li $v0, 1
        syscall

        la $a0, endl
        li $v0, 4
        syscall

        jr $ra


##########################################

colSum:

        addi $t9,$0,0

        addi $t0, $zero, 1 #i
        addi $t1, $zero, 1 #j
        addi $t8, $zero, 0

loopcol3:
        bgt $t0, $s0, outcol3

looprow3:
        bgt $t1, $s0, outrow3

        #calculate index
        addi $t3, $t0, -1
        mul $t3, $t3, $s0
        mul $t3, $t3, 4
        addi $t4, $t1, -1
        mul $t4, $t4, 4
        add $t3, $t4, $t3

        # add index to beginning of the array
        add $t4, $s1, $t3

        # take element from mem
        lw $t5, 0($t4)

        add $t8, $t8, $t5

        addi $t1, $t1, 1

        j looprow3
outrow3:

        add $t9,$t9,$t8 #t9 is total sum
```

```
        addi $a0, $t8,0
        li $v0, 1
        syscall

        la $a0, space
        li $v0, 4
        syscall

        addi $t1, $zero, 1
        addi $t0, $t0, 1
        addi $t8, $zero, 0


        j loopcol3

outcol3:

        la $a0, endl
        li $v0, 4
        syscall # endl

        addi $a0, $t9,0   #gives total sum
        li $v0, 1
        syscall

        la $a0, endl
        li $v0, 4
        syscall

        jr $ra

##########################################
rowcol:

        la $a0, endl
        li $v0, 4
        syscall

        la $a0, enteri
        li $v0, 4
        syscall

        li $v0, 5
        syscall
        addi $t1, $v0,0 #row i

        addi $t0,$0,1  #col

looprow4:
        bgt $t0, $s0, outrow4

        #calculate index
        addi $t3, $t0, -1
        mul $t3, $t3, $s0
```

```
        mul $t3, $t3, 4
        addi $t4, $t1, -1
        mul $t4, $t4, 4
        add $t3, $t4, $t3

        # add index to beginning of the array
        add $t4, $s1, $t3

        # take element from mem
        lw $t5, 0($t4)

        addi $a0, $t5,0
        li $v0, 1
        syscall

        la $a0, space
        li $v0, 4
        syscall

        addi $t0, $t0, 1

        j looprow4

outrow4:

        la $a0, endl
        li $v0, 4
        syscall

        ##

        la $a0, enterj
        li $v0, 4
        syscall

        li $v0, 5
        syscall
        addi $t0, $v0,0 #col j

        addi $t1,$0,1  #row

loopcol4:
        bgt $t1, $s0, outcol4

        #calculate index
        addi $t3, $t0, -1
        mul $t3, $t3, $s0
        mul $t3, $t3, 4
        addi $t4, $t1, -1
        mul $t4, $t4, 4
        add $t3, $t4, $t3

        # add index to beginning of the array
        add $t4, $s1, $t3

        # take element from mem
```

```
        lw $t5, 0($t4)

        addi $a0, $t5,0
        li $v0, 1
        syscall

        la $a0, space
        li $v0, 4
        syscall

        addi $t1, $t1, 1

        j loopcol4

outcol4:

        la $a0, endl
        li $v0, 4
        syscall

        jr $ra
```