CS224
Section No.: 2
Spring 2019
Lab No. 6
Munib Emre Sevilgen / 21602416

#Part 1

#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	Byte Offset Size in bits	Block Replacement Policy Needed (Yes/No)
1	32	1	32 bits	4	2^{11}	14	11	2	2	No
2	32	2	32 bits	4	2^{10}	15	10	2	2	Yes
3	32	4	32 bits	8	2^{8}	16	8	3	2	Yes
4	32	Full	32 bits	8	1	24	0	3	2	Yes
9	256	1	16 bits	4	2^{15}	11	15	2	1	No
10	256	2	16 bits	4	2^{14}	12	14	2	1	Yes
11	256	4	8 bits	16	2^{13}	12	13	4	0	Yes
12	256	Full	8 bits	16	1	25	0	4	0	Yes

#2.
#a.

Instruction	Iteration No.				
	1	**2**	**3**	**4**	**5**
lw $t1, 0x24($0)	Compulsory	Hit	Hit	Hit	Hit
lw $t2, 0x2C($0)	Compulsory	Hit	Hit	Hit	Hit
lw $t3, 0x28($0)	Hit	Hit	Hit	Hit	Hit

#b.
One set is 92 bits: 1 bit for valid, 27 bits for tag, and 64 bits for the data. There are 4 sets. Therefore, cache memory size is 368 bits.

#c.
One equality comparator is needed for tag comparison, One and gate is needed to check to see if valid is 1 while the tag matches and one (2x1) mux is needed to get the data we are looking for.

#3.
#a.

Instruction	**Iteration No.**	**Iteration No.**	**Iteration No.**	**Iteration No.**	**Iteration No.**
	1	**2**	**3**	**4**	**5**
lw $t1, 0x24($0)	Compulsory	Conflict	Conflict	Conflict	Conflict
lw $t2, 0x2C($0)	Compulsory	Conflict	Conflict	Conflict	Conflict
lw $t3, 0x28($0)	Conflict	Conflict	Conflict	Conflict	Conflict

#b.
One set is 127 bits: 2 for valid, 1 for LRU, 30*2 for tags, and 32*2 for data. There is only one set. Therefore, cache memory size is 127 bits.

#c.
Two equality comparators are needed for tag comparisons, two and gates are needed to check to see if valids are 1, while the tag matches and one mux is needed to get the data we are looking for. Also one or gate is needed to check whether the cache hits or not.

#4.
L1 takes 1 clock cycle, L2 takes 4 clock cycle, main memory access takes 80 clock cycle. L1 miss rate = %5, L2 miss rate = %25.
AMAT = 1 + 0.05 * (4 + 0.25 * 80) = 2.2 cycles.

Time for one instruction = Cycles * (1/clock_speed) = 2.2 * (1/2GHz) = 2.2 * (1/(2*10^9)) = 1.1 * 10^-9 sec.

Time for one instruction * instruction count = 1.1 * 10^-9 * 10^12 = 1100 seconds.

#5.

```
        .data
selection1: .asciiz "1 - To enter the matrix size in terms of its dimensions (N) and the elements row by row.\n"
selection2: .asciiz "2 - To enter the matrix size in terms of its dimensions (N) and initialize the matrix entries with consecutive values (1, 2, 3 ...).\n"
enterSize: .asciiz "Enter the matrix size: "
enterElement: .asciiz "Enter an element: "
selection3: .asciiz "3 - To display a desired element of the matrix.\n"
enterRow: .asciiz "Enter the row number: "
enterCol: .asciiz "Enter the column number: "
selection4: .asciiz "4 - To display entire matrix row by row.\n"
selection5: .asciiz "5 - To obtain trace of the matrix and display.\n"
selection6: .asciiz "6 - To obtain trace like summation using the other diagonal of the matrix and display.\n"
selection7: .asciiz "7 - To obtain sum of matrix elements by row-major (row by row) summation.\n"
selection8: .asciiz "8 - To obtain sum of matrix elements by column-major (column by column) summation.\n"
selection9: .asciiz "9 - To exit.\n"
enterSelection: .asciiz "Enter your selection: "
wrongSelection: .asciiz "Wrong selection.\n"
nextL: .asciiz "\n"
space: .asciiz " "
result: .asciiz "The result: "

        .text
main:
        li $v0, 4
        la $a0, selection1
        syscall

        la $a0, selection2
        syscall

        la $a0, selection3
        syscall

        la $a0, selection4
        syscall

        la $a0, selection5
        syscall
```

```
la $a0, selection6
syscall

la $a0, selection7
syscall

la $a0, selection8
syscall

la $a0, selection9
syscall

la $a0, enterSelection
syscall

li $v0, 5
syscall

move $s0, $v0 # s0 keeps selection

beq $s0, 1, first
beq $s0, 2, second
beq $s0, 3, third
beq $s0, 4, fourth
beq $s0, 5, fifth
beq $s0, 6, sixth
beq $s0, 7, seventh
beq $s0, 8, eighth
beq $s0, 9, end

li $v0, 4
la $a0, wrongSelection
syscall
j main

first:
        la $a0, enterSize
        li $v0, 4
        syscall

        li $v0, 5
        syscall
        move $s2, $v0 # s2 keeps the N

        li $t0, 4
```

```
        mult $s2, $s2
        mflo $s3 # s3 keeps N^2
        mult $t0, $s3
        mflo $a0
        li $v0, 9
        syscall

        move $s1, $v0 # pointer to the array
        move $t0, $zero
        move $t1, $s1

        loopFirst:
                beq $t0, $s3 loopFirstExit
                la $a0, enterElement
                li $v0, 4
                syscall
                li $v0, 5
                syscall
                move $t2, $v0 # the current element
                sw $t2, 0($t1)
                addi $t0, $t0, 1
                addi $t1, $t1, 4
                j loopFirst

        loopFirstExit:
        j main

second:
        la $a0, enterSize
        li $v0, 4
        syscall

        li $v0, 5
        syscall
        move $s2, $v0 # s2 keeps the N

        li $t0, 4
        mult $s2, $s2
        mflo $s3 # s3 keeps N^2
        mult $t0, $s3
        mflo $a0
        li $v0, 9
        syscall

        move $s1, $v0 # pointer to the array
```

```
        move $t0, $zero
        move $t1, $s1

        loopSecond:
                beq $t0, $s3 loopSecondExit
                addi $t2, $t0, 1 # the current element
                sw $t2, 0($t1)
                addi $t0, $t0, 1
                addi $t1, $t1, 4
                j loopSecond

        loopSecondExit:
        j main

third:
        la $a0, enterRow
        li $v0, 4
        syscall

        li $v0, 5
        syscall
        move $t0, $v0 # row number

        la $a0, enterCol
        li $v0, 4
        syscall

        li $v0, 5
        syscall
        move $t1, $v0 # col number

        subi $t0, $t0, 1
        mult $t0, $s2
        mflo $t0
        subi $t1, $t1, 1
        add $t0, $t0, $t1
        sll $t0, $t0, 2

        add $t0, $t0, $s1

        la $a0, result
        li $v0, 4
        syscall

        lw $a0, 0($t0)
```

```
        li $v0, 1
        syscall

        la $a0, nextL
        li $v0, 4
        syscall


        j main

fourth:
        move $t0, $zero
        move $t3, $s1
        loopFourth1: #row loop
                move $t1, $zero
                loopFourth2: #col loop
                        lw $a0, 0($t3)
                        li $v0, 1
                        syscall

                        la $a0, space
                        la $v0, 4
                        syscall

                        addi $t1, $t1, 1
                        addi $t3, $t3, 4
                        bne $t1, $s2 loopFourth2

                la $a0, nextL
                la $v0, 4
                syscall
                addi $t0, $t0, 1
                bne $t0, $s2 loopFourth1
        j main

fifth:
        move $t0, $zero # count
        addi $t1, $s2, 1 # N + 1
        move $t3, $zero # sum
        loopFifth:
                move $t2, $t0
                mult $t2, $t1
                mflo $t2
                sll $t2, $t2, 2
                add $t2, $t2, $s1
```

```
            lw $t4, 0($t2)
            add $t3, $t3, $t4
            addi $t0, $t0, 1
            bne $t0, $s2, loopFifth

        la $a0, result
        li $v0, 4
        syscall

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

        la $a0, nextL
        li $v0, 4
        syscall

        j main

sixth:
        move $t0, $zero # count
        move $t5, $s2
        move $t3, $zero # sum
        loopSixth:
            move $t2, $t0
            subi $t5, $t5, 1
            mult $t2, $s2
            mflo $t2
            add $t2, $t2, $t5
            sll $t2, $t2, 2
            add $t2, $t2, $s1
            lw $t4, 0($t2)
            add $t3, $t3, $t4
            addi $t0, $t0, 1
            bne $t0, $s2, loopSixth

        la $a0, result
        li $v0, 4
        syscall

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

        la $a0, nextL
```

```
        li $v0, 4
        syscall

        j main

seventh:
        move $t0, $zero
        move $t2, $zero
        move $t4, $zero
        loopSeventh1: #row loop
                move $t1, $zero
                loopSeventh2: #col loop
                        mult $t0, $s2
                        mflo $t2
                        add $t2, $t2, $t1
                        sll $t2, $t2, 2
                        add $t2, $t2, $s1
                        lw $t3, 0($t2)
                        add $t4, $t4, $t3
                        addi $t1, $t1, 1
                        bne $t1, $s2 loopSeventh2
                addi $t0, $t0, 1
                bne $t0, $s2 loopSeventh1
        la $a0, result
        li $v0, 4
        syscall

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

        la $a0, nextL
        li $v0, 4
        syscall

        j main

eighth:
        move $t0, $zero
        move $t2, $zero
        move $t4, $zero
        loopEight1: #col loop
                move $t1, $zero
                loopEight2: #row loop
                        mult $t1, $s2
```

```
                        mflo $t2
                        add $t2, $t2, $t0
                        sll $t2, $t2, 2
                        add $t2, $t2, $s1
                        lw $t3, 0($t2)
                        add $t4, $t4, $t3
                        addi $t1, $t1, 1
                        bne $t1, $s2 loopEight2
                addi $t0, $t0, 1
                bne $t0, $s2 loopEight1
        la $a0, result
        li $v0, 4
        syscall

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

        la $a0, nextL
        li $v0, 4
        syscall
        j main


end:
        li $v0, 10
        syscall
```