CS224
Section No.: 2
Spring 2018
Lab No.: 6
EFE ACER / 21602217

## PRELIMINARY DESIGN REPORT:

## Question 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 | Block offset size in bits | Byte offset size in bits | Block replacement policy needed (Yes/No) |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | **256** | **1** | **32 bits** | **4** | $2^{14}$ | 14 | 14 | 2 | 2 | No |
| **2** | **256** | **2** | **32 bits** | **4** | $2^{13}$ | 15 | 13 | 2 | 2 | Yes |
| **3** | **256** | **4** | **32 bits** | **8** | $2^{11}$ | 16 | 11 | 3 | 2 | Yes |
| **4** | **256** | **Full** | **32 bits** | **8** | $2^{0}$ | 27 | 0 | 3 | 2 | Yes |
| **9** | **512** | **1** | **16 bits** | **4** | $2^{16}$ | 13 | 16 | 2 | 1 | No |
| **10** | **512** | **2** | **16 bits** | **4** | $2^{15}$ | 14 | 15 | 2 | 1 | Yes |
| **11** | **512** | **4** | **16 bits** | **16** | $2^{12}$ | 15 | 12 | 4 | 1 | Yes |
| **12** | **512** | **Full** | **16 bits** | **16** | $2^{0}$ | 27 | 0 | 4 | 1 | Yes |

Question 2:

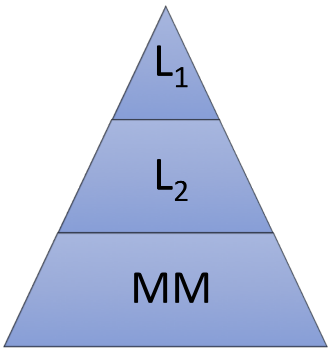
| Memory Address Accessed (hex) | Set no. | Hit (Yes/No) |
|---|---|---|
| **00 00 00 28** | 01 | No |
| **00 00 00 49** | 01 | No |
| **00 00 00 6C** | 01 | No |
| **00 00 00 0C** | 01 | No |
| **00 00 00 0B** | 01 | Yes |
| **00 00 00 0D** | 01 | Yes |

## Question 3:

| **Memory Address Accessed (hex)** | **Set no.** | **Hit (Yes/No)** |
|---|---|---|
| **00 00 00 28** | 01 | No |
| **00 00 00 49** | 01 | No |
| **00 00 00 4C** | 01 | Yes |
| **00 00 00 0C** | 01 | No |
| **00 00 00 0B** | 01 | Yes |
| **00 00 00 0D** | 01 | Yes |

## Question 4:

Memory hierarchy is as follows:




It is given that:

$t_{L1}$ = 1 clock cycle, $t_{L2}$ = 3 clock cycles (2 times more than $t_{L1}$), $t_{MM}$ = 33 clock cycles (10 times more than $t_{L2}$)

$mr_{L1}$ = 20%, $mr_{L2}$ = 5%

where t denotes the access time, mr denotes miss rate and MM is the main memory.

Thus,

AMAT = $t_{L1} + mr_{L1} \cdot (t_{L2} + mr_{L2} \cdot t_{MM})$ = 1 + 20% . (3 + 5% . 33) = <u>1.93 clock cycles</u>

And if clock rate is 2GHz, clock period is 1 / 2GHz = 0.5ns, then:

Time needed for $10^{12}$ instructions = $10^{12}$ . 0.5ns . 1.93 = <u>965s</u>

## Question 5:

```
#Prelab 6
#This program initalizes a matrix and adds it and itself up in different ways.
#Author: EFE ACER

                .text
        .globl  __start

__start:
        jal monitor
        li $v0, 10 #stop execution
        syscall

#subprogram to display the matrix element in the specified position
#$a0 contains the row number and $a1 contains the column number as parameters
displayElement:
        subi $sp, $sp, 4 #save the return address
        sw $ra, 0($sp)
        subi $t0, $a0, 1 #$t0 becomes (rowNum - 1) x N x 4 + (colNum - 1) x 4
        mul $t0, $t0, $s0
        mul $t0, $t0, 4
        subi $t1, $a1, 1
        mul $t1, $t1, 4
        add $t0, $t0, $t1
        add $t0, $t0, $s2 #$t0 becomes the memory address of the specified element
        li $v0, 4
        la $a0, result2
        syscall
        li $v0, 1
        lw $a0, 0($t0) #$a0 becomes the specified element
        syscall
        lw $ra, 0($sp)
        addi $sp, $sp, 4
        jr $ra

#subprogram to display the row and column of the specified matrix element
#$a0 contains the specified matrix element as a parameter
displayPosition:
        subi $sp, $sp, 4 #save the return address
        sw $ra, 0($sp)
        div $a0, $s0 #modular arithmetic
        mflo $t0 #row
        mfhi $t1 #column
        beq $t1, $zero, specialCase
        addi $t0, $t0, 1
        j display
        specialCase:
                move $t1, $s0
        display:
                li $v0, 4 #display row
                la $a0, result3
                syscall
                li $v0, 1
                move $a0, $t0
                syscall
```

```
                li $v0, 4 #display column
                la $a0, result4
                syscall
                li $v0, 1
                move $a0, $t1
                syscall
        lw $ra, 0($sp)
        addi $sp, $sp, 4
        jr $ra #return

#subprogram to fill the matrix
fillMatrix:
        subi $sp, $sp, 8 #save the return address and $s2
        sw $ra, 0($sp)
        sw $s2, 4($sp)
        li $t0, 1
        fillLoop:
                sw $t0, 0($s2)
                addi $s2, $s2, 4 #iterate over the matrix
                addi $t0, $t0, 1
                ble $t0, $s1, fillLoop
        lw $s2, 4($sp)
        lw $ra, 0($sp)
        addi $sp, $sp, 8
        jr $ra #return

#subprogram to find and display the summation of the elements of a matrix by row major summation
#displays the resulting summation
addRowByRow:
        subi $sp, $sp, 12 #save the return address, $s1 and $s2
        sw $ra, 0($sp)
        sw $s1, 4($sp)
        sw $s2, 8($sp)
        move $t1, $zero #$t1 holds the summation
        rowMajorLoop:
                lw $t0, 0($s2) #add the values in the matrix and update the result
                add $t1, $t1, $t0
                addi $s2, $s2, 4 #iterate over the matrix
                subi $s1, $s1, 1
                bgt $s1, $zero, rowMajorLoop
        li $v0, 4 #display the result
        la $a0, result1
        syscall
        li $v0, 1
        move $a0, $t1
        syscall
        lw $s2, 8($sp) #load the return address, $s1 and $s2
        lw $s1, 4($sp)
        lw $ra, 0($sp)
        addi $sp, $sp, 12
        jr $ra #return

#subprogram to find and display the summation of the elements of a matrix by column major summation
#displays the resulting summation
addColumnByColumn:
        subi $sp, $sp, 12 #save the return address, $s1 and $s2
        sw $ra, 0($sp)
        sw $s1, 4($sp)
        sw $s2, 8($sp)
```

```
		move $t1, $zero #$t1 holds the summation
		move $t2, $zero #$t2 hols the current colNum - 1
		mul $t5, $s0, 4 #displacement between rows
		columnMajorLoop1:
			mul $t4, $t2, 4 #displacement in column
			add $t0, $s2, $t4 #$t0 holds the memory addresses of the accessed elements
			move $t3, $zero #$t3 hols the current rowNum - 1
			lw $t6, 0($t0)
			add $t1, $t1, $t6
			columnMajorLoop2:
				add $t0, $t0, $t5
				lw $t6, 0($t0) #add the values in the matrix and update the result
				add $t1, $t1, $t6
				addi $t3, $t3, 1 #update rowNum
				blt $t3, $s0, columnMajorLoop2
			addi $t2, $t2, 1 #update colNum
			blt $t2, $s0, columnMajorLoop1
		li $v0, 4 #display the result
		la $a0, result1
		syscall
		li $v0, 1
		move $a0, $t1
		syscall
		lw $s2, 8($sp) #load the return address, $s1 and $
		lw $s1, 4($sp)
		lw $ra, 0($sp)
		addi $sp, $sp, 12
		jr $ra #return

#a subprogram that calls the subprograms and controls the user experience
monitor:
		subi $sp, $sp, 4 #save the return addres
		sw $ra, 0($sp)
		li $v0, 4
		la $a0, intro #display the intro
		syscall
		mainLoop:
			jal printOptions #Print the options
			li $t1, '1'
			li $t2, '2'
			li $t3, '3'
			li $t4, '4'
			li $t5, '5'
			li $t6, '6'
			li $t0, 'q'
			li $v0, 12 #reading a character
			syscall #different cases regarding different menu options
			move $s3, $v0
			case1: #read the size of the matrix
				bne $s3, $t1, case2
				li $v0, 4
				la $a0, prompt1 #print a prompt to read the size of the matrix
				syscall
				li $v0, 5 #read the size of the matrix
				syscall
				move $s0, $v0 #$s0 has the size of matrix from now on
				mul $s1, $s0, $s0 #number of elements in the matrix is in $s1 from now on
				j default
			case2: #allocate and initialize matrix
```

```
                        bne $s3, $t2, case3
                        mul $a0, $s1, 4 #compute number of bytes to allocate for the matrix
                        li $v0, 9 #allocate heap memory
                        syscall #$v0 has the base address of the matrix
                        move $s2, $v0 #$s2 has the base address of the matrix from now on
                        jal fillMatrix
                        j default
                case3: #access and display a certain element
                        bne $s3, $t3, case4
                        li $v0, 4 #prompt for and read the row and column of the specified element
                        la $a0, prompt2
                        syscall
                        li $v0, 5
                        syscall
                        move $t0, $v0 #t0 holds the row number
                        li $v0, 4
                        la $a0, prompt3
                        syscall
                        li $v0, 5
                        syscall
                        move $a1, $v0 #a1 holds the column number
                        move $a0, $t0 #a0 holds the row number
                        jal displayElement #call the subprogram to display the specified element
                        j default
                case4: #summation of matrix elements by row-major summation
                        bne $s3, $t4, case5
                        jal addRowByRow
                        j default
                case5: #summation of matrix elements by column-major summation
                        bne $s3, $t5, case6
                        jal addColumnByColumn
                        j default
                case6: #get row and column of an element
                        bne $s3, $t6, default
                        li $v0, 4  #print a prompt to read the element
                        la $a0, prompt4
                        syscall
                        li $v0, 5 #read the element
                        syscall
                        move $a0, $v0 #$a0 has the element
                        jal displayPosition #call subprogram with the element as a parameter
                        j default
                default:
                        bne $s3, $t0, mainLoop
        lw $ra, 0($sp) #load the return address
        addi $sp, $sp, 4
        jr $ra #return

#subprogram to print user's options
printOptions:
        subi $sp, $sp, 4 #save the return address
        sw $ra, 0($sp)
        li $v0, 4
        la $a0, option1 #display the options
        syscall
        la $a0, option2
        syscall
        la $a0, option3
        syscall
```

```
	la $a0, option4
	syscall
	la $a0, option5
	syscall
	la $a0, option6
	syscall
	la $a0, optionQ
	syscall
	lw $ra, 0($sp) #load the return address
	addi $sp, $sp, 4
	jr $ra #return

		.data
intro:		.asciiz		"This program initalizes a matrix and adds it and itself up in different ways."
option1:	.asciiz		"\n\n1 - Enter a size in terms of the dimension of the matrix (N)."
option2:	.asciiz	"\n2 - Allocate and initialize an array for the matrix with proper size."
option3:	.asciiz	"\n3 - Access and display a certain element of the matrix."
option4:	.asciiz	"\n4 - Obtain summation of matrix elements by row-major (row by row) summation"
option5:	.asciiz	"\n5 - Obtain summation of matrix elements by column-major (column by column) summation"
option6:	.asciiz	"\n6 - Display desired element of the matrix by specifying its row and column member."
optionQ:	.asciiz	"\nq - Quit.\n\n"
prompt1:	.asciiz		"\nEnter the size: "
prompt2:	.asciiz		"\nEnter the row of the element: "
prompt3:	.asciiz		"\nEnter the column of the element: "
prompt4:	.asciiz		"\nEnter the element: "
result1: .asciiz	"\nThe summation of the matrix elements is: "
result2: .asciiz	"\nThe element in the specified position is: "
result3: .asciiz	"\nThe row number of the specified element: "
result4: .asciiz	"\nThe column number of the specified element: "
```