Course No.: CS224
Lab No.: 6
Section No.: 6
Full Name: Yasemin Akın
Bilkent ID.: 22101782

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	Word Block Offset Size in bits	Byte Offset Size in bits	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	1	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	1	27	0	4	1	Yes

Question 2.

a.

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

b.

Total Cache Size = Number of Blocks x (Block Size in bits + Tag bit + V bit) = 4 x (64 + 1 + 27)

= 4 x (92 bits) = **368 bits**

c.

2:1 Multiplexer = 1

Equality Comparator = 1

AND Gate = 1

Question 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)	Compulsory	Capacity	Capacity	Capacity	Capacity

b.

# of bits required for LRU will equal 1 as 21 equals to the number of blocks in the cache.

Total Cache Size = Number of Blocks x (Block Size in bits + Tag bit + V bit + LRU bit)

= 2 x (32 + 1 + 29 + 1) = 2 x (63 bits) = **126 bits**

c.

2: 1 Multiplexer = 1

Equality Comparator = 2

AND Gate = 2

OR Gate = 1

Question 4.

AMAT = Access time for L1 + Miss rate for L1 x (Access time for L2 + Miss rate for L2 x Access time for main memory)

= 1 + 0.2 x (4 + 0.05 x 40

= 2.2 clock cycles

Clock rate with 4GHz = 0.25 ns

10^{12} instructions requires time = 10^{12} x 0.25 x 2.2 = 0.55 x 10^{12} ns = 550 s

Question 5.

#Course No.: CS224

#Lab No.: 6

#Section No.: 6

#Full Name: Yasemin Akın

#Bilkent ID.: 22101782

```
                .data
menuTitle:      .asciiz   "\n\nMENU"
functPrompt:    .asciiz   "\nEnter an integer according to the choosen functionality: "
f1:             .asciiz   "\n1. Enter the matrix size in terms of its dimensions (N)"
f2:             .asciiz   "\n2. Allocate an array with proper size"
f3:             .asciiz "\n3. Obtain summation of matrix elements row-major (row by row) summation"
f4:             .asciiz "\n4. Obtain summation of matrix elements column-major (column by column) summation"
f5:             .asciiz "\n5. Display desired elements of the matrix by specifying its row and column member"
exitPrompt:     .asciiz "\n6. Exit"
askN:           .asciiz "\nPlease enter the matrix size in terms of its dimensions (N): "
rPrompt:        .asciiz "\nSummation of matrix elements in row by row major: "
cPrompt:        .asciiz "\nSummation of matrix elements in column by column major: "
rowNoPrompt:    .asciiz "\nEnter rowNo: "
columnNoPrompt: .asciiz "\nEnter columnNo: "
elementPrompt:          .asciiz "\nItem in the requested index = "
exitPromptM:    .asciiz "\nExiting the program..."

                .text
```

```
Main:
        jal     Menu
        li      $v0, 10
        syscall

Menu:
# Displaying the menu
        addi    $sp, $sp, -4
        sw      $ra, 0($sp)
        li      $v0, 4
        la      $a0, menuTitle
        syscall
        li      $v0, 4
        la      $a0, f1
        syscall
        la      $a0, f2
        syscall
        la      $a0, f3
        syscall
        la      $a0, f4
        syscall
        la      $a0, f5
        syscall
        la      $a0, exitPrompt
        syscall

        li      $v0, 4
        la      $a0, functPrompt
        syscall
```

```
        li      $v0, 5
        syscall

        beq     $v0, 1, funct1
        beq     $v0, 2, funct2
        beq     $v0, 3, funct3
        beq     $v0, 4, funct4
        beq     $v0, 5, funct5
        beq     $v0, 6, funct6

funct1:
        li      $v0, 4
        la      $a0, askN
        syscall
        li      $v0, 5
        syscall
        move    $s0, $v0                # N in $s0
        j       Menu

funct2:
        mul     $s2, $s0, $s0           # number of elements in array in $s2
        mul     $a0, $s2, 4
        li      $v0, 9
        syscall
        move    $s1, $v0                # array base address in $s1
        move    $s7, $s1
        # insertion starting from 1
        li      $t0, 1

insert:
```

```
        sw      $t0, 0($s7)
        addi    $s7, $s7, 4
        addi    $t0, $t0, 1
        ble     $t0, $s2, insert

        j       Menu

# row by row summation
funct3:
        addi    $sp, $sp, -8
        sw      $s1, 0($sp)
        sw      $s2, 4($sp)

        mul     $s4, $s0, 4             # column offset in $s4
        li      $s3, 0                  # total sum in $s3
        li      $t1, 0                  # row - 1 in $t1

loopC2:
        mul     $s5, $t1, 4             # row offset in $s5
        move    $s6, $0         # column - 1 in $s6
        add     $t5, $s1, $s5           # address in $t5
        lw      $t7, 0($t5)             # current element in $t7
        add     $s3, $t7, $s3

loopR:
        add     $t5, $s4, $t5           # address in $t5
        lw      $t7, 0($t5)             # current element in $t7
        add     $s3, $t7, $s3
        addi    $s6, $s6, 1             # column + 1
        blt     $s6, $s0, loopR         # if #column < N(in $s0) then cont
```

```
        addi    $t1, $t1, 1             # row + 1
        blt     $t1, $s0, loopC2

        li      $v0, 4
        la      $a0, rPrompt
        syscall
        move    $a0, $s3
        li      $v0, 1
        syscall

        lw      $s2, 4($sp)
        lw      $s1, 0($sp)
        addi    $sp, $sp, 8
        j       Menu

# column by column summation
funct4:
        addi    $sp, $sp, -8
        sw      $s1, 0($sp)
        sw      $s2, 4($sp)

        li      $s3, 0                  # total sum in $s3

loopC:
        lw      $s4, 0($s1)             # current item to add in $s4
        addi    $s1, $s1, 4
        add     $s3, $s3, $s4
        subi    $s2, $s2, 1
        bgt     $s2, 0, loopC
```

```
        li      $v0, 4
        la      $a0, cPrompt
        syscall
        move    $a0, $s3
        li      $v0, 1
        syscall

        lw      $s2, 4($sp)
        lw      $s1, 0($sp)
        addi    $sp, $sp, 8
        j       Menu

funct5:
        li      $v0, 4                  # ask for rowNo
        la      $a0, rowNoPrompt
        syscall
        li      $v0, 5
        syscall
        move    $s4, $v0                # rowNo in $s4

        li      $v0, 4                  # ask for columnNo
        la      $a0, columnNoPrompt
        syscall
        li      $v0, 5
        syscall
        move    $s3, $v0                # columnNo in $s3

        subi    $s3, $s3, 1
        mul     $s3, $s3, $s0
        mul     $s3, $s3, 4
```

```
        subi    $s4, $s4, 1
        mul     $s4, $s4, 4
        add     $s3, $s3, $s4
        add     $s5, $s3, $s1           # address of the requested index in $s5

        li      $v0, 4
        la      $a0, elementPrompt
        syscall
        lw      $a0, 0($s5)
        li      $v0, 1
        syscall
        j       Menu

# Exit
funct6:
        li      $v0, 4
        la      $a0, exitPromptM
        syscall
        lw      $ra, 0($sp)
        addi    $sp, $sp, 4
        jr      $ra
```