BILKENT UNIVERSITY
COMPUTER ENGINEERING
CS 224
COMPUTER ORGANIZATION

PRELIMINARY DESIGN REPORT
LAB 06

BERK YILDIZ

21502040
SECTION 4

20.12.2018

CS224
Section 4
Fall 2018
Lab 06
Berk Yıldız / 21502040

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	1	27	0	3	2	Yes
9	128	1	16 bits	4	2^13	16	13	2	1	No
10	128	2	16 bits	4	2^12	17	12	2	1	Yes
11	128	4	16 bits	16	2^9	18	9	4	1	Yes
12	128	Full	16 bits	16	1	27	0	4	1	Yes

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				

4.

AMAT = time cache + miss rate cache (time main memory)

AMAT= time L1 + miss rate L1 (time L2) + time L2 + miss rate L2(time main memory)

= 1 + 0.2(4) + 4 + 0.05(40) = 7.8 cycles

2^12 = 4096 operations

4096 * 7.8 = 31948,8 cycles are needed.

4GHz = 4000000000 cycles per second.

31948,8 / 4000000000 = 0,000008 second

5.

```
main:
.data

	promptSize:	.asciiz "(1) Enter matrix size in terms of dimensions.\n"
	promptSizeInput:.asciiz "Enter the dimension of the matrix: "
	promptAllocate:	.asciiz "(2) Allocate matrix.\n"
	promptIndex:	.asciiz "(3) Enter index to display content.\n"
	promptInputI:	.asciiz "Enter i : "
	promptInputJ:	.asciiz "Enter j : "
	promptSumRow:	.asciiz "(4) Summation of matrix elements row-major summation.\n"
	promptSumColumn:.asciiz "(5) Summation of matrix elements column-major summation.\n"
	sumColumnOutput:.asciiz "Column-major summation: "
	sumRowOutput:	.asciiz "Row-major summation: "
```

```
      promptDisplay:   .asciiz "(6) Display desired elements of the matrix by specifying its row and column member.\n"
      promptQuit:      .asciiz "(7) Quit.\n"
      promptOption:    .asciiz "Enter an option: "
      blank:           .asciiz " "
      line:            .asciiz "\n"
      promptIndexOut:  .asciiz "Element is : "
      promptRowColumn: .asciiz "Display elements of row or a column. Enter '0' for a row, '1' for a column: "
      promptDisplayColumn: .asciiz "Which column?: "
      promptDisplayRow: .asciiz "Which row?: "

.text
menu:

      la $a0, line
      li $v0, 4
      syscall

      la $a0, promptSize
      li $v0, 4
      syscall
      li $v0, 4
      la $a0, promptAllocate
      syscall
      li $v0, 4
      la $a0, promptIndex
      syscall
      li $v0, 4
      la $a0, promptSumRow
      syscall
      li $v0, 4
```

```
	la $a0, promptSumColumn
	syscall
	li $v0, 4
	la $a0, promptDisplay
	syscall
	li $v0, 4
	la $a0, promptQuit
	syscall
	li $v0, 4
	la $a0, promptOption
	syscall
	li $v0, 5
	syscall
	# move to the appropriate branch
	addi $a0, $s0, 0
	beq $v0, 1, enterSize
	beq $v0, 2, allocateMatrix
	beq $v0, 3, indexDisplay
	beq $v0, 4, sumRow
	beq $v0, 5, sumColumn
	beq $v0, 6, rowColumnDisplay
	beq $v0, 7, end
enterSize:
	la $a0, promptSizeInput
	li $v0, 4
	syscall
	li $v0, 5
	syscall

	move $v1, $v0         # $v1 dimension of the matrix
	move $s2, $v0         # store dimension in s2 to use later
	j menu
```

```
allocateMatrix:
      mul $v1, $v1, $v1
      sll $a0, $v1, 2  #allocate memory
      li $v0, 9
      syscall

      addi $s0, $v0, 0 # $s0 has beginning address of array
      addi $t0, $zero, 0    # $t0 index
      li $v0, 1
value:
      beq $t0, $v1, done

      sw $v0, ($s0)
      addi $s0, $s0, 4
      addi $t0, $t0, 1
      addi $v0, $v0, 1
      j value
done:
      sll $t0, $t0, 2
      sub $v0, $s0, $t0
      addi $s0, $v0, 0
      addi $a1, $v1, 0
      j menu
indexDisplay:
      la $a0, promptInputI
      li $v0, 4
      syscall
      li $v0, 5
      syscall
```

```
    move $t3, $v0 #move i to t3

    la $a0, promptInputJ
    li $v0, 4
    syscall
    li $v0, 5
    syscall

    move $t4, $v0 #move j to t4

    la $a0, line
    li $v0, 4
    syscall

    #calculate index -> (j-1) * dimension + i
    addi $t4, $t4, -1
    mul $t4, $t4, $s2
    add $t4, $t4, $t3

    addi $s1, $zero, 0 #index
    addi $t0, $s0, 0   #address
    la $a0, promptIndexOut
    li $v0, 4
    syscall
do:
    bge $s1, $t4, print
    addi $s1, $s1, 1
    lw $a0, ($t0)
    addi $t0, $t0, 4
    j do
```

```
print:
      li $v0, 1
      syscall
      j menu
sumRow:
      move $t4, $s2  #move dimension to t4
      li $t5, 0  #sum  t5
      li $t6, 0 #dimension index
      li $t7, 0 #total address
      li $t8, 0
      mul $t1, $t4, 4 #address increment
      mul $t2, $t4, $t4 #size
      mul $t7, $t2, 4
      la $a0, sumRowOutput
      li $v0, 4
      syscall

      addi $t0, $s0, 0   #address
      add $t7, $t7, $t0
      li $s1, 0 #index
doRowSum:
      bge $s1, $t2, printSumRow
      addi $s1, $s1, 1
      bge $t6, $t4, cont
cont:
      addi $t6, $t6, 1
      lw $a0, ($t0)
      add $t5, $t5, $a0
      add $t0, $t0, $t1
      bge $t0, $t7, doRowSum2

      j doRowSum
```

```
doRowSum2:
      move $t0, $s0
      addi $t8, $t8, 4
      add $t0, $t0, $t8
      j doRowSum
printSumRow:
      move $a0, $t5
      li $v0, 1
      syscall
      j menu

sumColumn:
      li $t5, 0  #sum
      la $a0, sumColumnOutput
      li $v0, 4
      syscall

      addi $s1, $zero, 0 #index
      addi $t0, $s0, 0   #address
doColumnSum:
      bge $s1, $a1, printSumColumn
      addi $s1, $s1, 1
      lw $a0, ($t0)
      add $t5, $t5, $a0
      addi $t0, $t0, 4
      j doColumnSum

printSumColumn:
      move $a0, $t5
      li $v0, 1
      syscall
      j menu
```

```
rowColumnDisplay:

      la $a0, promptRowColumn
      li $v0, 4
      syscall
      li $v0, 5
      syscall

      beq $v0, 0, displayRow

      li $t1, 0 # t1 will be used for address calculations
      la $a0, promptDisplayColumn
      li $v0, 4
      syscall
      li $v0, 5
      syscall

      addi $v0, $v0, -1
      addi $t0, $s0, 0   #address
      addi $s1, $zero, 0 #dimension index
      move $t1, $v0
      mul $t1, $t1, $s2
      mul $t1, $t1, 4
      add $t0, $t0, $t1

columnDisplayLoop:
      beq $s1, $s2, menu
      addi $s1, $s1, 1
      lw $a0, ($t0)
      addi $t0, $t0, 4
```

```
	li $v0, 1
	syscall
	la $a0, blank
	li $v0, 4
	syscall

	j columnDisplayLoop



displayRow:
	li $t1, 0 # t1 will be used for address calculations
	la $a0, promptDisplayRow
	li $v0, 4
	syscall
	li $v0, 5
	syscall

	addi $v0, $v0, -1
	addi $t0, $s0, 0   #address
	addi $s1, $zero, 0 #dimension index
	move $t1, $v0
	mul $t1, $t1, 4
	add $t0, $t0, $t1
	li $t2, 0
	mul $t2, $s2, 4 #address increment
```

```
rowDisplayLoop:
      beq $s1, $s2, menu
      addi $s1, $s1, 1
      lw $a0, ($t0)
      add $t0, $t0, $t2

      li $v0, 1
      syscall
      la $a0, blank
      li $v0, 4
      syscall

      j rowDisplayLoop


end:
      li $v0, 10
      syscall
```