Bilkent University

Cs224-01

Osman Buğra Aydın

21704100

Lab05

20 May 2020

PART-1

Question-1

No.	Cache Size KB	N way cache	Word Size in bits	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	2	1	32	4	2^7	21	7	2	2	NO
2	2	2	32	4	2^6	22	6	2	2	YES
3	2	4	32	8	2^5	22	5	3	2	YES
4	2	Full	32	8	2^0	27	0	3	2	YES
9	16	1	16	4	2^11	18	11	2	1	NO
10	16	2	16	4	2^10	19	10	2	1	YES
11	16	4	8	16	2^7	21	7	4	0	YES
12	16	Full	8	16	2^0	28	0	4	0	YES

Question-2

a-)

	1	2	3	4	5
lw $t1, 0xA4($0)	Compulsory	Hit	Hit	Hit	Hit
lw $t2, 0xAC($0)	Compulsory	Hit	Hit	Hit	Hit
lw $t3, 0xA8($0)	Hit	Hit	Hit	Hit	Hit

b-)

Cache capacity is 8 words.
Block size: 2 words
N= 1.

V	Tag	Data	Data
1 bit	27 bit	32 bit	32 bit
1 bit	27 bit	32 bit	32 bit
1 bit	27 bit	32 bit	32 bit

(1 + 27 + 32 + 32) * 4 = 368

c-)

1 AND gate to identify a hit
1 equality comparator to compare tag
1 2:1 mux to select data in the block

Question-3

a-)

	1	2	3	4	5
lw $t1, 0xA4($0)	Compulsory	Capacity	Capacity	Capacity	Capacity
lw $t2, 0xAC($0)	Compulsory	Capacity	Capacity	Capacity	Capacity
lw $t3, 0xA8($0)	Capacity	Capacity	Capacity	Capacity	Capacity

b-)

Cache capacity is 2 words
Block size is 1 word.
There is only 1 set.
The block replacement policy is LRU.

V	Tag	Data	V	Tag	Data
1 bit	30 bit	32 bit	1 bit	30 bit	32 bit

(1 + 30 + 32) * 2 = 126

c-)

2 AND gate and 1 OR gate to identify a hit
2 equality comparator to compare tag
1 2:1 mux to select data in the block

Question-4

```
        .text
Main:
        la    $a0, wel
        li    $v0, 4
        syscall

        jal menu
```

```
        li     $v0, 10
        syscall

menu:
        la     $a0, menuS
        li     $v0, 4
        syscall

        la     $a0, create
        syscall
        la     $a0, sumR
        syscall
        la     $a0, sumC
        syscall
        la     $a0, dispE
        syscall
        la     $a0, dispM
        syscall
        la     $a0, exit
        syscall
        la     $a0, opt
        syscall

        li     $v0, 5
        syscall

        beq $v0, 6, ex

        op1: bne $v0, 1, op2

                beq  $s1, 1, alreadyE
```

```
        li     $s0, 1

        jal    createMatrix

        move      $s2, $v0
        move      $s3, $v1


        j menu

op2: bne $v0, 2, op3

        bne  $s0, 1, matErr

        move      $a0, $s2
        move      $a1, $s3

        jal    calculateSumR

op3: bne $v0, 3, op4

        bne  $s0, 1, matErr

        move      $a0, $s2
        move      $a1, $s3

        jal    calculateSumC

op4: bne $v0, 4, op5
```

```
        bne $s0, 1, matErr


        jal   displayElement

op5: bne $v0, 5, op6

        bne $s0, 1, matErr

        move     $a0, $s2
        move     $a1, $s3


        jal   displayMatrix

op6:
        la    $a0, err
        li    $v0, 4
        syscall
        j     menu

matErr:
        la    $a0, materr
        li    $v0, 4
        syscall
        j     menu
alreadyE:
        la    $a0, alErr
        li    $v0, 4
        syscall
        j     menu
```

```
	ex:

		li	$v0, 10
	syscall

	jr	$ra

createMatrix:
	addi	$sp, $sp, -4
	sw	$s0, 0($sp)

	la	$a0, valN
	li	$v0, 4
	syscall

	li	$v0, 5
	syscall
	j	control

	inputErr:

		la	$a0, inErr
		li	$v0, 4
		syscall
		j	createMatrix

	control: blt	$v0, 1, inputErr

	mul	$s7, $v0, $v0	# NxN -> element size
	mul	$a0, $s7, 4	#For the bytes
	move	$s5, $v0
```

```
li     $v0, 9
syscall

move      $s6, $v0
move      $t0, $v0

li     $s0, 0

storeElements:

        la     $a0, val
        li     $v0, 4
        syscall

        li     $v0, 5
        syscall

        sw   $v0, 0($t0)
        addi        $t0, $t0, 4
        addi        $s0, $s0, 1

        bne  $s7, $s0, storeElements

move      $v0, $s6
move      $v1, $s5

lw   $s0, 0($sp)
addi        $sp, $sp, 4
jr     $ra
```

```
calculateSumR:
        addi $sp, $sp, -16
        sw $s0, 0($sp)
        sw $s1, 4($sp)
        sw $s2, 8($sp)
        sw $s3, 12($sp)

        li     $s0,0
        move      $s1, $a0
        mul $s4, $a1, $a1
        move      $s5, $a1

        li     $s3, 0

        For:

                lw    $a0, 0($s1)

                add  $s3, $s3, $a0

                addi$s1, $s1, 4
                addi$s0, $s0, 1
                bne  $s4,$s0, For

        la     $a0, result
        li     $v0, 4
        syscall

        move      $a0, $s3
        li     $v0, 1
        syscall
```

```
        lw $s3, 12($sp)
        lw $s2, 8($sp)
        lw $s1, 4($sp)
        lw $s0, 0($sp)
        addi $sp, $sp, 16
        jr     $ra

calculateSumC:
        addi $sp, $sp, -16
        sw $s0, 0($sp)
        sw $s1, 4($sp)
        sw $s2, 8($sp)
        sw $s3, 12($sp)

        move      $s0, $a1
        move      $s1, $a0

        li     $s3, 0

        mul $t3, $s0, 4
        li $t0, 0
        li $t1, 0

        rowLoop:

                li     $t2,0
                mul         $t4, $t1, 4
                add  $t5, $t4, $s1
                lw   $t6, 0($t5)
                add  $t0, $t6, $t0
```

```
        colLoop:

                addi        $t2, $t2, 1
                add  $t5, $t3, $t5
                lw    $t6, 0($t5)
                add  $t0, $t6, $t0
                bgt   $s0, $t2, colLoop

        addi        $t1, $t1, 1
        blt    $t1, $s0, rowLoop

    la     $a0, result
    li     $v0, 4
    syscall

    move      $a0, $t0
    li     $v0, 1
    syscall

    lw $s3, 12($sp)
    lw $s2, 8($sp)
    lw $s1, 4($sp)
    lw $s0, 0($sp)
    addi $sp, $sp, 16
    jr     $ra

displayElement:

    jr     $ra
```

```
displayMatrix:
        addi $sp, $sp, -16
        sw $s0, 0($sp)
        sw $s1, 4($sp)
        sw $s2, 8($sp)
        sw $s3, 12($sp)

        li     $s0,0
        move      $s1, $a0
        mul $s4, $a1, $a1


        for:
                lw    $a0, 0($s1)
                li     $v0, 1
                syscall

                la     $a0, space
                li     $v0, 4
                syscall

                beq  $s0, $zero, done

                div  $s3, $s0, $a1
                mfhi       $s3

                addi$s5, $a1, -1

                bne  $s3, $s5, done
                la     $a0, newL
                li     $v0, 4
```

```
        syscall

        done:
        addi$s1, $s1, 4
        addi$s0, $s0, 1
        bne $s4,$s0, for

    lw $s3, 12($sp)
    lw $s2, 8($sp)
    lw $s1, 4($sp)
    lw $s0, 0($sp)
    addi $sp, $sp, 16

    jr    $ra


    .data
wel:        .asciiz "Welcome To My Program! \n \n"
menuS:  .asciiz "<--------------------> OPTIONS
<-------------------->\n \n"
create:    .asciiz "1-)Create NxN matrix.\n"
valN:      .asciiz "\nGive the value of N:"
sumR:    .asciiz "2-)Calculate the summation of matrix
( Row by Row ).\n"
sumC:    .asciiz "3-)Calculate the summation of matrix
( Column by Column ).\n"
dispE:    .asciiz "4-)Display the element by row and
column.\n"
dispM:   .asciiz "5-)Display the matrix.\n"
result:    .asciiz "Result of the summation: "
opt: .asciiz "\n\nWhat is your option: "
```

```
exit:.asciiz "6-)EXIT"
err:  .asciiz "\nPlease give a correct option number!\n\n"
materr:   .asciiz "\nPlease create a matrix before other
options!\n\n"
inErr:     .asciiz "\n\nPlease give a number above 0!\n\n"
val: .asciiz "\nGive a value: "
alErr:     .asciiz "\nMatrix already created!\n"
space:     .asciiz " "
newL:     .asciiz "\n"
row:       .asciiz "Row:"
col: .asciiz "Col:"
```