CS224
Section No.: 1
Lab No.: 6
İpek Öztaş 22003250

CS224

Section No.: 1

Lab No.: 6

İpek Öztaş

22003250

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[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	2^{0}	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	2^{0}	27	0	4	1	Yes

#1: Number of sets= $(2^6 \times 2^{10})$ / (1 block/set x 4 words/block x 4 bytes/word) = $2^{16}/2^4= 2^{12}$
#2: Number of sets= $(2^6 \times 2^{10})$ / (2 block/set x 4 words/block x 4 bytes/word) = $2^{16}/2^5= 2^{11}$
#3: Number of sets= $(2^6 \times 2^{10})$ / (4 block/set x 8 words/block x 4 bytes/word) = $2^{16}/2^7= 2^{9}$
#9: Number of sets= $(2^7 \times 2^{10})$ / (1 block/set x 4 words/block x 2 bytes/word) = $2^{17}/2^3= 2^{14}$
#10: Number of sets= $(2^7 \times 2^{10})$ / (2 block/set x 4 words/block x 2 bytes/word) = $2^{17}/2^4= 2^{13}$
#11: Number of sets= $(2^7 \times 2^{10})$ / (4 block/set x 16 words/block x 2 bytes/word) = $2^{17}/2^7= 2^{10}$
Tag size= 32 - (index size in bits + block offset size in bits + byte offset size in bits)
In a 1-way cache, it is direct-mapped. There is only one choice to replace a block and therefore no replacement policy is needed.

Memory size is 4GB= 4×2^{30} bytes

[1] Word Block Offset Size in bits: log2(No. of words in a block)
[2] Byte Offset Size in bits: log2(No. of bytes in a word)

CS224
Section No.: 1
Lab No.: 6
İpek Öztaş 22003250

Question 2:

Cache memory size = 8 words

Block size = 2 words

N = 1 (direct mapped)

MIPS memory size is 4GB. Physical memory = 4GB = 2^{32} bytes

Physical memory address size is 32 bits

Number of blocks (B = C/b) = 8/2 = 4

Number of sets (S = B/N) = 4/1 = 4 sets

In MIPS, word size is 32 bits (4 bytes) = 2 bit byte offset

Block size = 2 words = 8 bytes = 2^3. So Offset is 3 bits

Byte Offset 2 bits, since there are 4 bytes per word (32-bit word size)

Block Offset 1 bit, since there are 2 words in each block

There are 4 sets, INDEX/Set is 2 bits

Address has 32 bits, 3 are Offset, 2 are Index/Set, so Tag is 27 bits.

```
      addi $t0, $0, 5
loop: beq $t0, $0, done
      lw $t1, 0x4($0)
      lw $t2, 0xC($0)
      lw $t3, 0x8($0)
      addi $t0, $t0, -1
      j loop
done:
```

a. In the following table indicate the type of miss, if any: Compulsory, Conflict, Capacity.

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

CS224
Section No.: 1
Lab No.: 6
İpek Öztaş 22003250
Larger blocks reduce compulsory misses through spatial locality.

Compulsory Miss:

Occurs during the instruction "lw $t1, 0x4($0)" because it is the first access to that block.

Occurs during the instruction "lw $t2, 0xC($0)" because it is the first access to that block.

No other cache misses occur as the subsequent memory accesses fall within blocks that have already been fetched into the cache.

b. What is the total cache memory size in number of bits? Include the V bit your calculations. Show the details of your calculation.

V bit = 1 bit

Tag = 32-2-1-2 = 27 bits

Data = 32 bits

Index = 2 bits

Block offset = 1 bit

Byte offset = 2 bits

Total cache size = No. of sets x (V bit + Tag + Data) = 4 x (1 + 27 + 2x32) = 4 x 92 = *368* bits

c. State the number of AND and OR gates, EQUALITY COMPARATORs and MULTIPLEXERs needed to implement the cache memory.

1 EQUALITY COMPARATOR, 1 AND gate, and 1 2:1 MULTIPLEXER are needed.

Question 3:

a. In the following table indicate the type of miss, if any: Compulsory, Conflict, Capacity.

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. What is the total cache memory size in number of bits? Include the V bit your calculations. Show the details of your calculation.

V Tag Data V Tag Data

1 30 32 1 30 32 (bits)

Number of bits required for LRU will be equal to 1 since 2^1 = number of blocks in cache Cache size = Number of blocks* (Block size in bits + Valid bit + Tag bit + LRU bit) = 1 + 30 + 32 + 1 + 30 + 32 = 126 bits

c. State the number of AND and OR gates, EQUALITY COMPARATORs and MULTIPLEXERs needed to implement the cache memory.

2 AND gates, 1 OR gate, 2 EQUALITY COMPARATOR and 1 2:1 MULTIPLEXER are needed.

Question 4:

According to the following information:

Access Time for L1 = 1 Clock Cycle

Access Time for L2 = 5 Clock Cycles

Access Time for Main memory = 55 Clock Cycles

Miss rate for L1 = 0.2 Miss Rate for L2 = 0.05

So we have the formula for AMAT (Average Memory Access Time) as:

AMAT = t cache + MRcache[tMM + MRMM(tVM)]

AMAT = Hit Time for L1 + Miss Rate for L1 * (Hit time for L2 + Miss Rate for L2 * (Hit time for Main memory))

= 1 + 0.2*(5 + 0.05*55)

= 1 + 0.2 * 7.75

= 2.55 Clock Cycles

With 4 GHz clock rate how much time is needed for a program with 10^{12} instructions to execute?

4 GHz is equal to 0.25 nanoseconds.

Time taken to execute 10^{12} instructions with 4 GHz processor = 10^{12} * 2.55 * 0.25 = 637.5 seconds

Question 5:

#CS224

#Lab 6.

#Section 1.

#İpek Öztaş

CS224
Section No.: 1
Lab No.: 6
İpek Öztaş 22003250

```
#22003250

#21.5.2023

                .text

        .globl  __start


__start:


        jal main

        li $v0, 10

        syscall




main:


        addi $sp, $sp, -4

        sw $ra, 0($sp)


        menu:

                jal displayMenu

                # Ask user to enter option

                li $v0, 4

                la $a0, prompt

                syscall


                li $v0, 5

                syscall


                #create an array with consecutive numbers
```

```
beq $v0, 1, select2
# display desired element
beq $v0, 2, select3
# sum array row wise
beq $v0, 3, select5
# sum array column wise
beq $v0, 4, select6
#exit
beq $v0, 5, selectExit



select2:
	li $v0, 4
	la $a0, promptDim
	syscall

	li $v0, 5
	syscall

	move $s0, $v0 #N
	mul $s2, $s0, $s0 # N*N
	mul $a0, $s2, 4 # size
	#array allocation
	li $v0, 9
	syscall

	move $a1, $v0 #array address
```

```
        move $s1, $a1

        move $a0, $s2   #N*N

        jal consecutiveValues

        j loopBack

    select3:


        li $v0, 4

        la $a0, enterRow

        syscall


        li $v0, 5

        syscall


        move $t0, $v0 # row - $t0


        li $v0, 4

        la $a0, enterCol

        syscall


        li $v0, 5

        syscall


        move $a1, $v0 # column


        move $a0, $t0


        jal getItem

        j loopBack
```

```
        select5:

            move $a0, $s0  #N

            move $a1, $s1

            # sum matrix row wise

            jal sumMatrixRow

            j loopBack

        select6:

            move $a0, $s2  #N*N

            move $a1, $s1

            # sum matrix column wise

            jal sumMatrixCol

            j loopBack


        loopBack:

            j menu

      selectExit:


    lw $ra, 0($sp)

    addi $sp, $sp, 4

    jr $ra


getItem:


    addi $sp, $sp, -28

    sw $ra, 0($sp)

    sw $s0, 4($sp) #no of elements (N)

    sw $s1, 8($sp)   #address

    sw $s2, 12($sp) #size (N^2)
```

```
sw $s3, 16($sp)
sw $s4, 20($sp)
sw $s5, 24($sp)



move $s4, $a0 # row
move $s3, $a1 # column

subi $s3, $s3, 1 # j-1
mul $s3, $s3, $s0 # (j-1) * N
mul $s3, $s3, 4 # (j-1) * N * 4 byte wise
subi $s4, $s4, 1 # i-1
mul $s4, $s4, 4 # (i-1) * 4
add $s3, $s3, $s4 # (j-1) * N * 4 + (i-1) * 4

add $s5, $s3, $s1 # address

li $v0, 4
la $a0, showItem
syscall

lw $a0, 0($s5)
li $v0, 1
syscall



lw $s5, 24($sp)
```

```
    lw $s4, 20($sp)
    lw $s3, 16($sp)
    lw $s2, 12($sp)
    lw $s1, 8($sp)
    lw $s0, 4($sp)
    lw $ra, 0($sp)
    addi $sp, $sp, 28
    jr $ra
```

```
# summation of elements col-major
sumMatrixCol:

    addi $sp, $sp, -20
    sw $ra, 0($sp)
    sw $s0, 4($sp)  #address
    sw $s1, 8($sp)   #size
    sw $s2, 12($sp) #sum
    sw $s3, 16($sp) #temp

    move $s0, $a1
    move $s1, $a0
    move $s2, $0
    li $s3, 0
```

CS224
Section No.: 1
Lab No.: 6
İpek Öztaş 22003250

```
sumItems:
        lw $s3, 0($s0)    #current item of the matrix
        add $s2, $s2, $s3
        addi $s0, $s0, 4 # next element of the matrix
        addi $s1, $s1, -1 # decrement index
        bne $s1, $zero, sumItems        #stop when there are no elements left to add


    li $v0, 4
    la $a0, displaySum
    syscall

    move $a0, $s2
    li $v0, 1
    syscall

    move $v0, $s2
    lw $s3, 16($sp)
    lw $s2, 12($sp)
    lw $s1, 8($sp)
    lw $s0, 4($sp)
    lw $ra, 0($sp)
    addi $sp, $sp, 20
    jr $ra


sumMatrixRow:
```

CS224
Section No.: 1
Lab No.: 6
İpek Öztaş 22003250

```
#save the address and size of the matrix

addi $sp, $sp, -32

sw $ra, 0($sp)

sw $s0, 4($sp) #no of elements (N)

sw $s1, 8($sp)    #sum

sw $s2, 12($sp) #4N

sw $s3, 16($sp) #address

sw $s5, 24($sp)

sw $s6, 28($sp)

sw $s7, 32($sp)


move $s0, $a0

move $s3, $a1 #address of the matrix

li $s1, 0

mul $s2, $s0, 4 #4N


li $s6, 0 #outer loop counter


outerLoop2:

	bge $s6, $s0, outerLoopEnd2

	li $s7, 0 #inner loop counter

	move $s5, $s3


innerLoop2:


	bge $s7, $s0, innerLoopEnd2


	lw $t1, 0($s3) # element
```

```
        add $s1, $s1, $t1
        add $s3, $s3, $s2  #next row-columnwise
        addi $s7, $s7, 1
        b innerLoop2

    innerLoopEnd2:
        addi $s6, $s6, 1
        move $s3, $s5
        addi $s3, $s3, 4
        b outerLoop2


    outerLoopEnd2:

    li $v0, 4
    la $a0, displaySum
    syscall

    move $a0, $s1
    li $v0, 1
    syscall
    move $v0, $s1


    lw $s7, 32($sp)
    lw $s6, 28($sp)
    lw $s5, 24($sp)
    lw $s3, 16($sp)
```

```
        lw $s2, 12($sp)
        lw $s1, 8($sp)
        lw $s0, 4($sp)
        lw $ra, 0($sp)
        addi $sp, $sp, 32
        jr $ra

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

        li $s2, 1 #array element
        move $s1, $a1   #address
        move $s0, $a0   #N
        fillArray:
                sw $s2, 0($s1)
                addi $s2, $s2, 1
                addi $s1, $s1, 4
                addi $s0, $s0, -1
                bne $s0, 0, fillArray

        lw $s2, 12($sp)
        lw $s1, 8($sp)
        lw $s0, 4($sp)
        lw $ra, 0($sp)
```

```
	addi $sp, $sp, 16
	jr $ra



displayMenu:

	addi $sp, $sp, -4
	sw $ra, 0($sp)
	li $v0, 4

	la $a0, menudisp
	syscall

	la $a0, menu2
	syscall
	la $a0, menu3
	syscall
	la $a0, menu5
	syscall
	la $a0, menu6
	syscall
	la $a0, menuExit
	syscall
	lw $ra, 0($sp)
	addi $sp, $sp, 4
	jr $ra
```

CS224
Section No.: 1
Lab No.: 6
İpek Öztaş 22003250

```
.data

prompt:     .asciiz "Please choose an option:  "
menudisp: .asciiz " \n MENU \n"

menu2: .asciiz " 1) Create matrix (NxN) initialized with consecutive values \n"
menu3: .asciiz " 2) Display desired element of the matrix by row*column \n"
menu5: .asciiz " 3) Obtain summation of matrix elements row-major (row by row) summationse\n"
menu6: .asciiz " 4) Obtain summation of matrix elements column-major (column by column)summation
\n"
menuExit: .asciiz " 5) Exit \n"

enterRow: .asciiz "Please enter Row number i: \n "
enterCol: .asciiz "Please enter Column number j: \n "
showItem: .asciiz "Item in the given row and column is :  "
promptDim: .asciiz "Enter the matrix dimension N of (NxN) matrix: "
displaySum:  .asciiz "\n Sum is : "

line: .asciiz "\n"
space: .asciiz " "
```