**Preliminary Work / Preliminary Design Report (50 points)**

CS224

Section No.: 05

Spring 2018

Lab No.: 06

Asuman Aydın/21502604

1. **(** **5 points:** **With 2 or more errors you get 0 points. Otherwise full point.)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 bits1** | **Byte**  **Offset**  **Size in bits2** | **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 | 1 | 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 | 1 | 27 | 0 | 4 | 1 | YES |

**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)

**2**. **(5 points:** **With 2 or more errors you get 0 points. Otherwise full point.)**

|  |  |  |
| --- | --- | --- |
| **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 |

**3**. **(5 points:** **With 2 or more errors you get 0 points. Otherwise full point.)**

|  |  |  |
| --- | --- | --- |
| **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 |

**4**. **(5 points, With 1 or more errors you get 0 points. Otherwise full point.)**

L1: 1 clock cycle

L2: 2 clock cycles

Main Memory: as L2 is 2 clock cycles, 20 clock cycles

L1 miss = %20

L2 miss = %5

AMAT = L1\*HITRATE + L2\*HITRATE + MAIN\*HITRATE

= 0.8 + 0.2\*0.95\*3 + 0.2\*0.05\*1\*23

= 1.6 clock cycles

Every time we check L1, L2 so in the L2 time we add L1 clock cycle and miss rate too. Also for the main, it applies.

For 2 GHz clock = (Expected clock cycle per ins \* # of inst)/2 GHz

= 1.6\*10^12\*2\*10^9

= 800

**5**. **(30 points)**

.data

space : .asciiz " "

boslul: .asciiz " \n"

Menu: .asciiz "Menu\n"

Menu1: .asciiz "1. enter the matrix size in terms of its dimensions (N) \n"

Menu2: .asciiz "Allocate an array with proper size using syscall code 9\n"

Menu3: .asciiz "3.Ask user the matrix element to be accessed and display the content\n"

Menu4: .asciiz "4.Obtain summation of matrix elements row-major (row by row) summation\n"

Menu5: .asciiz "5.Obtain summation of matrix elements column-major (column by column) summation\n"

Menu6: .asciiz "6.Display desired elements of the matrix by specifying its row and column member\n"

Menu7: .asciiz "QUİT"

Choose: .asciiz "7.Enter the option you want to execute \n"

.text

menu:

li $v0,4

la $a0, Menu

syscall

li $v0,4

la $a0, Menu1

syscall

li $v0,4

la $a0, Menu2

syscall

li $v0,4

la $a0, Menu3

syscall

li $v0,4

la $a0, Menu4

syscall

li $v0,4

la $a0, Menu5

syscall

li $v0,4

la $a0, Menu6

syscall

li $v0,4

la $a0, Choose

syscall

li $v0, 5

syscall

move $t5, $v0

beq $t5,1, menu1

beq $t5,3, display

beq $t5, 4, summation

beq $t5, 5, colSummation

beq $t5, 6, desiredDisplay

beq $t5, 7, quit

#number of row and column

menu1:

.text

li $v0, 4 # prompt for number

la $a0,Menu1

syscall

li $v0, 5 # read a integer number

syscall

#row

move $t1, $v0

mul $s1,$t1,$t1

move $a0, $s1

#memory allocation

li $v0, 9

#base address

move $a0, $s1

syscall

#yedekleme

move $s2, $v0

move $s7,$s2 #display

move $t6, $s2 #sum

move $t0, $s2 #colsum

#initiliaziton of array has two loops

li $t3, 0

li $s6, 1

arrayBeginning:

bge $t3, $t1, endX

li $t4, 0

arrayX:

bge $t4, $t1, end

sw $s6, 0($s2)

addi $s2, $s2, 4

addi $s6, $s6, 1

addi $t4, $t4, 1

b arrayX

end:

addi $t3,$t3, 1

b arrayBeginning

endX:

j menu

display:

li $t8, 0

display2:

#t1=size, t8=counter

bge $t8, $t1, endEnd

li $t9, 0#second loop counter

arrayX2:

bge $t9, $t1, doneDisplay

li $s5,0

lw $s5, 0($s7)

#li $v0,4

move $a0, $s5

li $v0,1

syscall

addi $t9, $t9, 1

addi $s7, $s7, 4

b arrayX2

doneDisplay:

li $v0, 4

la $a0,boslul

syscall

addi $t8, $t8, 1

b display2

endEnd:

j menu

summation:

li $t8, 0

li $t7, 0

summation2:

#t1=size, t8=counter

bge $t8, $t1, endSummation

li $t9, 0#second loop counter

firstloop:

bge $t9, $t1, sum

li $s5,0

lw $s5, 0($t6)

#li $v0,4

add $t7, $t7, $s5

addi $t9, $t9, 1

addi $t6, $t6, 4

b firstloop

sum:

move $a0, $t7

li $v0,1

syscall

li $v0, 4

la $a0,boslul

syscall

li $t7,0

addi $t8, $t8, 1

b summation2

endSummation:

j menu

colSummation:

li $t8, 0

li $t7, 0

move $s3, $t0

#t1=size, t8=counter

summationCol:

bge $t8, $t1, endSummation2

move $t5, $t1

li $t9, 0#second loop counter

firstloop2:

bge $t9, $t1, sum2

li $s5,0

lw $s5, 0($s3)

#li $v0,4

add $t7, $t7, $s5

mul $t5, $t5, 4 #size kadar jump

addi $t9, $t9, 1

add $s3, $s3, $t5

b firstloop2

sum2:

move $a0, $t7

li $v0,1

syscall

li $v0, 4

la $a0,boslul

syscall

li $t7,0

move $s3, $t0

add $s3, $s3, 4

addi $t8, $t8, 1

b summationCol

endSummation2:

j menu

desiredDisplay:

li $v0, 4 # prompt for number

la $a0,Menu1

syscall

li $v0, 5 # read a integer number

syscall

move $t4, $v0

li $t8,0

display3:

#t1=size, t8=counter

bge $t8, $t1, endloop

li $t9, 0#second loop counter

loop3:

bge $t9, $t1, doneDisplayDesired

li $s5,0

lw $s5, 0($s7)

#li $v0,4

beq $s5, $t4, displaycont

addi $t9, $t9, 1

addi $s7, $s7, 4

b arrayX2

doneDisplayDesired:

addi $t8, $t8, 1

b display2

displaycont:

li $v0,1

la $a0, $s5

syscall

li $v0, 4

la $a0,boslul

syscall

endloop:

j menu

quit:

li $v0,10

syscall