**CS 224**

**Lab 5**

**Preliminary Report**

Group

Cüneyt EREM

21202398

8.4.2016

b, c, d, e, f, g, h)

f) because of different o notations (n vs n^2)

g) first is quick, but second is slow, they can be showns in the table,(by using watch,; for the first size, it is 1,5 sn and 2,3 sn, so it is a bit different from computer result)

All other b,c,d,e,h parts are in the code;

I)

Analysis Table

|  |  |  |
| --- | --- | --- |
| **Size** | **Fill time in seconds** | **Sum Time in seconds** |
| **50** | **1.773** | **1.937** |
| **100** | **7.128** | **7.767** |
| **200** | **28.530** | **31.137** |
| **400** | **114.454** | **124.459** |

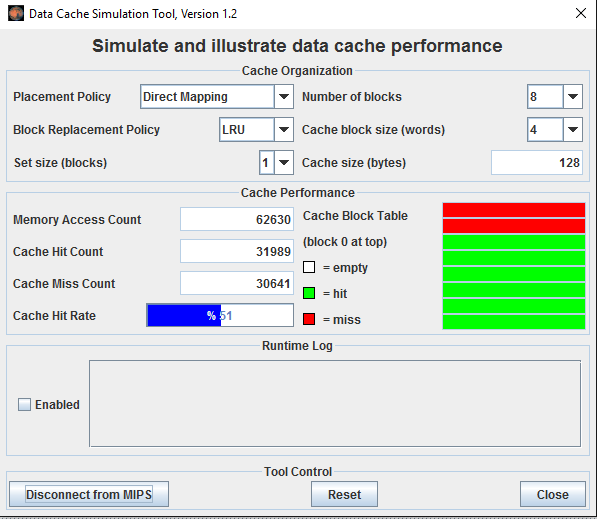
Normally by using cache, execution time should be faster but because of this program written in java, it shows slower.

J)

|  |  |  |
| --- | --- | --- |
| Wıdth/Total Time | Without Data cache simulator | With Data cache simulator |
| N = 50 | 3.65 | 4.8 |
| N = 100 | 14.8 | 19.4 |

AS you can see in the second table the data cache simulator increase the time 32%.

By using direct map, when num of block are 8, block size 4 and cache size is 128 bytes, cache hit rate, and hit/miss counts are as in this;



b,c,d,e,h)

**Programming Code for Matrix sum:**

.text

.globl begin

begin:

la $a0,start

li $v0,4

syscall

li $v0, 30 # time

syscall

move $t9, $a0

la $t0,size # s0 = N

lw $s0,0($t0)

alocateHeap:

mul $s4, $s0,$s0 # N \* N

sll $a0, $s4, 2 # 4 \* N \* N

li $v0, 9

syscall

move $s1, $v0 # s1 = A

li $v0, 9

syscall

move $s2, $v0 # s2 = B

li $v0, 9

syscall

move $s3, $v0 # s3 = C

move $t0,$0 # int i = 0

move $t1,$s2 # base adress of B

FillMatrixB:

sw $t0, 0($t1)

addi $t0, $t0, 1 # i++

addi $t1, $t1, 4

bne $t0, $s4, FillMatrixB

move $t0,$s4 # int i = N\*N

move $t1,$s3 # base adress

FillMatrixC:

sw $t0, 0($t1)

subi $t0, $t0, 1 #i--

addi $t1, $t1, 4

bne $t0, $0, FillMatrixC

li $v0, 30 # time

syscall

move $t8, $a0

sub $t0,$t8,$t9

la $a0,fill

li $v0,4

syscall

move $a0, $t0

li $v0, 1 # prints time

syscall

la $a0,passLine

li $v0,4

syscall

move $t0,$0

move $t1,$s1

move $t2,$s2

move $t3,$s3

sumTop:

lw $t4, 0($t2)

lw $t5, 0($t3)

add $t4,$t4,$t5

sw $t4, 0($t1)

addi $t0, $t0, 1

addi $t1, $t1, 4

addi $t2, $t2, 4

addi $t3, $t3, 4

bne $t0, $s4, sumTop

li $v0, 30 # time

syscall

move $t7, $a0

sub $t0,$t7,$t8

la $a0,endMessage

li $v0,4

syscall

move $a0, $t0

li $v0, 1 # prints time

syscall

la $a0,passLine

li $v0,4

syscall

#skip the display result

j end

move $t0,$0

move $t2,$s1

DisplayResult:

move $t1,$0

RowDisplay:

lw $a0, 0($t2)

li $v0, 1

syscall

la $a0,tabCH

li $v0,4

syscall

addi $t0, $t0, 1

addi $t1, $t1, 1

addi $t2, $t2, 4

bne $t1, $s0, RowDisplay #end of the row

la $a0,passLine

li $v0,4

syscall

bne $t0, $s4, DisplayResult

end:

li $v0,10 # system call to exit

syscall # bye bye

ReadFile:

li $v0, 13 # system call for open file

la $a0, fileName1# board file name

li $a1, 0 # Open for reading

li $a2, 0

syscall # open a file (file descriptor returned in $v0)

move $s6, $v0 # save the file descriptor

#read from file

li $v0, 14 # system call for read from file

move $a0, $s6 # file descriptor

la $a1, buffer # address of buffer to which to read

li $a2, 1024 # hardcoded buffer length

syscall # read from file

# Close the file

li $v0, 16 # system call for close file

move $a0, $s6 # file descriptor to close

syscall # close file

.data

size: .word 400

start: .asciiz "Program is started\n"

fill: .asciiz "Matrix is filled : "

endMessage: .asciiz "Program is ended : "

tabCH: .asciiz "\t"

passLine: .asciiz "\n"

buffer:.space 1024

fileName1:.asciiz "matrix1.dat"

fileName2:.asciiz "matrix2.dat"