

CST-307

[illegible]

	Miss: 4.556 %	Miss: 4.556 %	Miss: 4.556 %	Miss: 4.556 %	Miss: 4.556 %	Miss: 4.556 %	Miss: 4.556 %	Miss: 4.556 %	Miss: 4.556 %	Miss: 4.556 %	Miss: 4.556 %	Miss: 4.556 %
16 Blocks	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %
32 Blocks	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %
64 Blocks	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %

Wave.prg 2 Processors (Po and P1):

	2 Way	2 Way	2 Way	2 Way	4 Way	4 Way	4 Way	4 Way	8 Way	8 Way	8 Way	8 Way
	LRU	FIFO	RAN D	LFU	LRU	FIFO	RAN D	LFU	LRU	FIFO	RAN D	LFU
8 Blocks	Hit: 67.552 % Miss: 32.448 %	Hit: 67.26 % Miss: 32.74 %	Hit: 66.034 % Miss: 33.966 %	Hit: 67.085 % Miss: 32.915 %	Hit: 67.669 % Miss: 32.331 %	Hit: 67.26 % Miss: 32.74 %	Hit: 65.101 % Miss: 34.899 %	Hit: 67.348 % Miss: 32.652 %	Hit: 68.106 % Miss: 31.894 %	Hit: 68.106 % Miss: 31.894 %	Hit: 65.130 % Miss: 34.870 %	Hit: 67.756 % Miss: 32.244 %
16 Blocks	Hit: 74.263 % Miss: 25.737 %	Hit: 74.380 % Miss: 25.620 %	Hit: 72.833 % Miss: 27.167 %	Hit: 73.913 % Miss: 26.087 %	Hit: 74.876 % Miss: 25.124 %	Hit: 75.080 % Miss: 25.920 %	Hit: 72.221 % Miss: 27.779 %	Hit: 74.584 % Miss: 25.416 %	Hit: 75.197 % Miss: 24.803 %	Hit: 74.73 % Miss: 25.27 %	Hit: 72.542 % Miss: 27.458 %	Hit: 75.051 % Miss: 24.949 %
32 Blocks	Hit: 78.903 % Miss: 21.097 %	Hit: 78.494 % Miss: 21.506 %	Hit: 77.181 % Miss: 22.819 %	Hit: 78.203 % Miss: 21.797 %	Hit: 79.757 % Miss: 21.243 %	Hit: 78.494 % Miss: 21.506 %	Hit: 76.247 % Miss: 23.753 %	Hit: 78.669 % Miss: 21.331 %	Hit: 79.136 % Miss: 20.864 %	Hit: 79.08 % Miss: 20.92 %	Hit: 76.889 % Miss: 23.111 %	Hit: 78.903 % Miss: 21.097 %

64 Blocks	Hit: 80.245 % Miss: 19.755 %	Hit: 79.895 % Miss: 20.201 5%	Hit: 77.998 % Miss: 22.002 %	Hit: 80.070 % Miss: 19.930 %	Hit: 80.683 % Miss: 19.317 %	Hit: 80.537 % Miss: 19.463 %	Hit: 77.882 % Miss: 22.118 %	Hit: 79.924 % Miss: 20.076 %	Hit: 80.712 % Miss: 19.288 %	Hit: 80.391 % Miss: 19.609 %	Hit: 77.210 % Miss: 22.790 %	Hit: 80.537 % Miss: 19.463 %
-----------	---------------------------------	----------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------

Wave_Burst 2 Processors (Po and P1):

	2 Way	2 Way	2 Way	2 Way	4 Way	4 Way	4 Way	4 Way	8 Way	8 Way	8 Way	8 Way
	LRU	FIFO	RAN D	LFU	LRU	FIFO	RAN D	LFU	LRU	FIFO	RAN D	LFU
8 Blocks	Hit: 95.444 % Miss: 4.556 %	Hit: 95.444 % Miss: 4.556 %	Hit: 95.444 % Miss: 4.556 %	Hit: 95.444 % Miss: 4.556 %	Hit: 95.444 % Miss: 4.556 %	Hit: 95.444 % Miss: 4.556 %	Hit: 95.444 % Miss: 4.556 %	Hit: 95.444 % Miss: 4.556 %	Hit: 95.444 % Miss: 4.556 %	Hit: 95.444 % Miss: 4.556 %	Hit: 95.444 % Miss: 4.556 %	Hit: 95.444 % Miss: 4.556 %
16 Blocks	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %	Hit: 97.518 % Miss: 2.482 %
32 Blocks	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %	Hit: 98.548 % Miss: 1.452 %
64 Blocks	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %	Hit: 99.062 % Miss: 0.938 %

a. First use one processor *P0* and then use 2 processors *P0 and P1*.

In the first two tables, one processor was used. In the second two tables, two processors were in use. As seen by the table, the number of processors seemed to have no effect on the wave_burst.prg. However, for the wave.prg, the table with two processors had a lower hit rate and a higher miss rate.

b. Analyze the effectiveness of different block replacement techniques by listing down the miss rate in each case.

The effectiveness of the different block replacement techniques are listed in each of their respective tables. The hit and miss rate was recorded for each. To find these values, the SMPCCache

Simulator was used. After inputting the correct information for each table, the hit and miss rate were calculated by the simulator and recorded into their respective cells on the table.

c. What other block replacement technique can be used and is proved to be the ideal? Explain.

Another block replacement technique that can be used is the Least Frequently Used (LFU) method. This caching algorithm takes the least frequently used block in the cache and removes it whenever the cache becomes overflowed. This method is ideal because it actually gets rid of the least common block in the cache rather than getting rid of a random block in the cache or getting rid of a block in the first in first out method.

Part 2: MIPS Code

Program 1: Print the sum of two integers specified at runtime by the user

```
# Data Declarations
#-----

# Different text lines to print
num1:      .asciiz "Enter integer 1: " # Asking user to enter first int
num2:      .asciiz "Enter integer 2: " # Asking user to enter second int
sum:       .asciiz "Resulting Sum: "   # Printing resultant sum

#-----

        .globl main
        .text

#-----
# Main Code / Text Section

main:

# First Integer

        li $v0, 4          # print_string
        la $a0, num1       # input into a0
        syscall
        li $v0, 5          # read_integer
        syscall
        move $t0, $v0      # move into register t0

# Second Integer

        li $v0, 4          # print_string
        la $a0, num2       # input into a0
        syscall

        li $v0, 5          # read_integer
        syscall

        move $t1, $v0      # move input to register $t1
        li $v0, 4          # print_string
        la $a0, sum        # puts sum in a0
        syscall

        add $t2, $t1, $t0   # add integers
        move $a0, $t2      # move sum to a0 to be printed
        li $v0, 1          # command to print the sum
        syscall

        li $v0, 10         # Ends program
        syscall
```

*The code will also be provided in a separate submitted document.

Execution of Program1:

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [10] Data Text

Int Regs [10]

PC = 4194428
EPC = 0
Cause = 0
BadVAddr = 0
Status = 805371664
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 268500992
R2 [v0] = 10
R3 [v1] = 0
R4 [a0] = 35
R5 [a1] = 2147481616
R6 [a2] = 2147481624
R7 [a3] = 0
R8 [t0] = 10
R9 [t1] = 25
R10 [t2] = 35
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0

Text

```
[00400000] 01a40000 lw $a, 0($29) ; 183: lw $a0  
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $5, $29, 4  
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $6, $5, 4  
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $2, $4, 2  
[00400010] 00c23021 addi $2, $2, 1  
[00400014] 0c100009 jal $ra, 0($ra) ; 187: jal $ra, 0($ra)  
[00400018] 00000000 nop ; 188: nop  
[0040001c] 3402000a ori $a0, $a0, 10 ; 189: ori $a0, $a0, 10  
[00400020] 0000000c sysc ; 190: sysc  
[00400024] 34020004 ori $a0, $a0, 4 ; 191: ori $a0, $a0, 4  
[00400028] 3c041001 lui $a0, 0x41001 ; 192: lui $a0, 0x41001  
[0040002c] 0000000c sysc ; 193: sysc  
[00400030] 34020005 ori $a0, $a0, 5 ; 194: ori $a0, $a0, 5  
[00400034] 0000000c sysc ; 195: sysc  
[00400038] 00024021 addi $a0, $a0, 35 ; 196: addi $a0, $a0, 35  
[0040003c] 34020004 ori $a0, $a0, 4 ; 197: ori $a0, $a0, 4  
[00400040] 3c011001 lui $a0, 0x11001 ; 198: lui $a0, 0x11001  
[00400044] 34240012 ori $a0, $a0, 12 ; 199: ori $a0, $a0, 12  
[00400048] 0000000c sysc ; 200: sysc  
[0040004c] 34020005 ori $a0, $a0, 5 ; 201: ori $a0, $a0, 5  
[00400050] 0000000c sysc ; 202: sysc  
[00400054] 00024821 addi $a0, $a0, 37 ; 203: addi $a0, $a0, 37  
[00400058] 34020004 ori $a0, $a0, 4 ; 204: ori $a0, $a0, 4  
[0040005c] 3c011001 lui $a0, 0x11001 ; 205: lui $a0, 0x11001  
[00400060] 34240024 ori $a0, $a0, 24 ; 206: ori $a0, $a0, 24  
[00400064] 0000000c sysc ; 207: sysc  
[00400068] 01285020 addi $a0, $a0, 1000000 ; 208: addi $a0, $a0, 1000000  
[0040006c] 000a2021 addi $a0, $a0, 10 ; 209: addi $a0, $a0, 10  
[00400070] 34020001 ori $a0, $a0, 1 ; 210: ori $a0, $a0, 1  
[00400074] 0000000c sysc ; 211: sysc  
[00400078] 3402000a ori $a0, $a0, 10 ; 212: ori $a0, $a0, 10  
[0040007c] 0000000c sysc ; 213: sysc
```

Console

```
Enter integer 1: 10  
Enter integer 2: 25  
Resulting Sum: 35
```

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.

Program 2: Print the larger of two numbers specified at runtime by the user.

```

# Colt Uhlenhopp & Jack Utzerath
# Program 2 CLC 1
# 2/3/2023
# This Program prints the larger number inputed by the user in the console

# Data Declarations
# =====
.data

first: .asciiz "Enter first integer: "
second: .asciiz "Enter second integer: "
n: .asciiz "\n"
larger: .asciiz "The larger int is: "

# =====
# Main Code Section

.globl main
.text

main:
    li $v0, 4          # code for printing string
    la $a0, first      # loads address of string to be printed in a0
    syscall

    li $v0, 5          # code for read_int
    syscall
    move $t0, $v0      # moves the user input into t0

    #
    li $v0, 4          # code for printing string
    la $a0, second     # loads address of string to be printed in a0
    syscall

    li $v0, 5          # code for read_int
    syscall
    move $t1, $v0      # moves user input into t1

    bgt $t0, $t1, t0big # if t0 is bigger than t1, go to t0big method
    move $t2, $t1       # if not, move t1 into t2
    b endif             # go to endif method

#=====
# t0big Method

t0big:
    move $t2, $t0      # moves t0 into t2

#=====
# endif Method

endif:
    li $v0, 4          # code for printing string
    la $a0, larger     # load address of string to be printed
    syscall

    move $a0, $t2       # moves t2 into a0 to be printed
    li $v0, 1          # code for printing integer
    syscall

    li $v0, 4          # code for printing string
    la $a0, n          # loads address of string to be printed
    syscall

end:
    li $v0, 10         # code for exit of program
    syscall

```

Execution of Program2:

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [10] Data Text

Int Regs [10]

PC = 4194464
EPC = 0
Cause = 0
BadVAddr = 0
Status = 805371664

HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 268500992
R2 [v0] = 10
R3 [v1] = 0
R4 [a0] = 268501084
R5 [a1] = 2147481616
R6 [a2] = 2147481624
R7 [a3] = 0
R8 [t0] = 17
R9 [t1] = 25
R10 [t2] = 25
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [s8] = 0
R25 [s9] = 0
R26 [s10] = 0
R27 [s11] = 0
R28 [s12] = 0
R29 [s13] = 0
R30 [s14] = 0
R31 [s15] = 0

end:
li \$v0, 10 # code for exit of program
syscall

Text

```
[00400024] 34020004 ori $2, $0, 4  
[00400028] 3c011001 lui $1, 4097 [first]  
[0040002c] 3424002f ori $4, $1, 47 [first]  
[00400030] 0000000c syscall  
[00400034] 34020005 ori  
[00400038] 0000000c syscall  
[0040003c] 00024021 addu  
[00400040] 34020004 ori  
[00400044] 3c011001 lui  
[00400048] 34240045 ori  
[0040004c] 0000000c syscall  
[00400050] 34020005 ori  
[00400054] 0000000c syscall  
[00400058] 00024821 addu  
[0040005c] 0128082a slt  
[00400060] 14200003 bne  
[00400064] 00095021 addu  
[00400068] 04010002 bgez  
[0040006c] 00085021 addu  
[00400070] 34020004 ori  
[00400074] 3c011001 lui  
[00400078] 3424005e ori  
[0040007c] 0000000c syscall  
[00400080] 000a2021 addu  
[00400084] 34020001 ori  
[00400088] 0000000c syscall  
[0040008c] 34020004 ori  
[00400090] 3c011001 lui  
[00400094] 3424005c ori  
[00400098] 0000000c syscall  
[0040009c] 3402000a ori  
[004000a0] 0000000c syscall
```

Console

Enter first integer: 17
Enter second integer: 25
The larger int is: 25