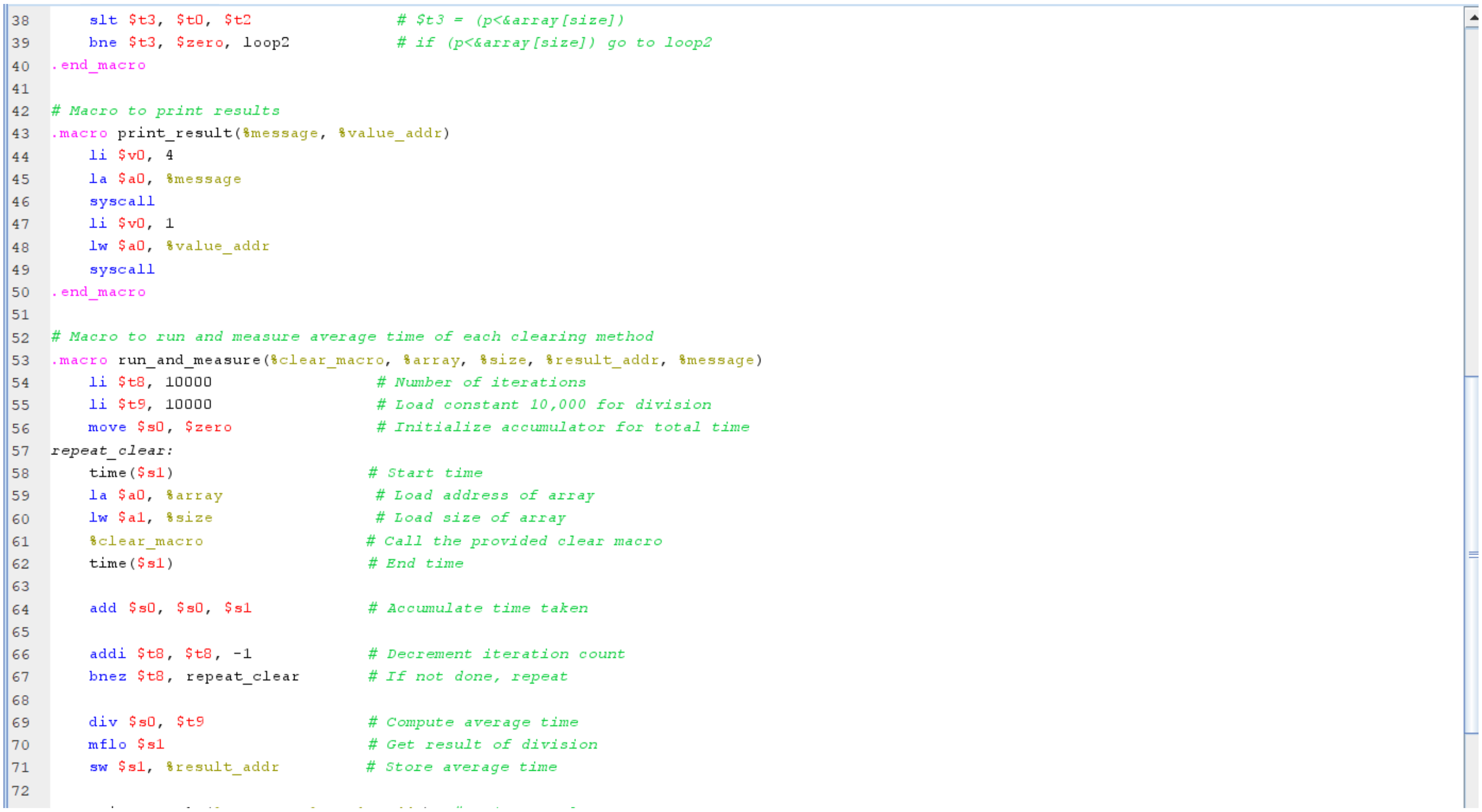
Time measurement of arrays

Bello Melido

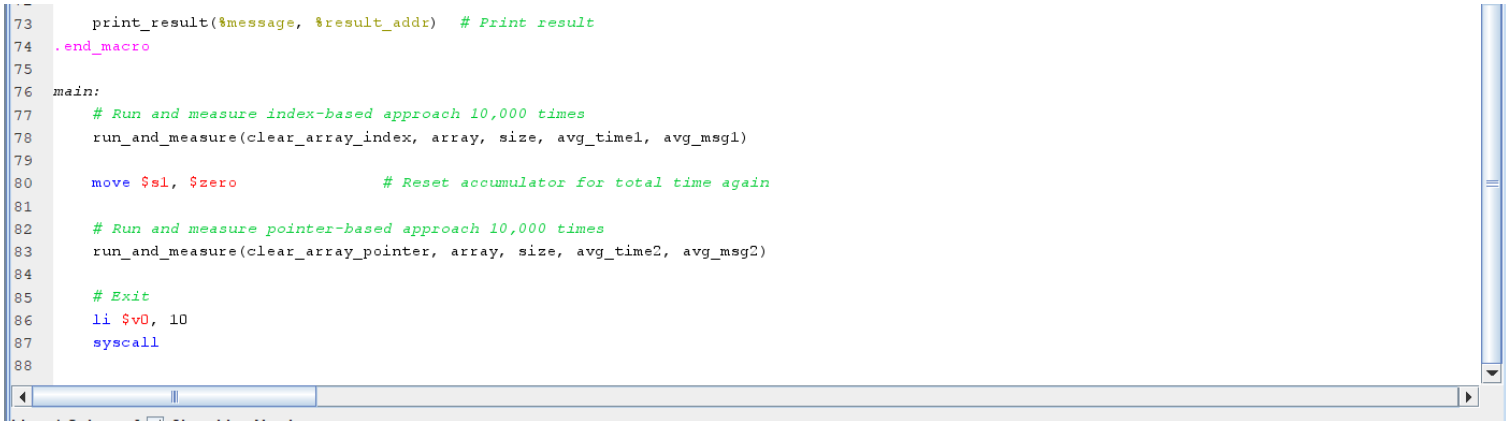
10/24/2023

CSC 210

Code

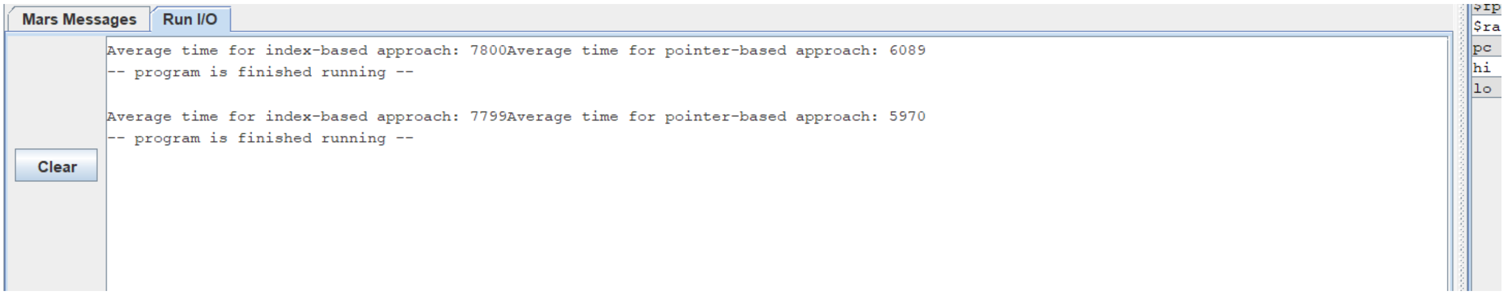


```
    slt $t3, $t0, $t2                # $t3 = (p<&array[size])
    bne $t3, $zero, loop2            # if (p<&array[size]) go to loop2
.end_macro

# Macro to print results
.macro print_result(%message, %value_addr)
    li $v0, 4
    la $a0, %message
    syscall
    li $v0, 1
    lw $a0, %value_addr
    syscall
.end_macro

# Macro to run and measure average time of each clearing method
.macro run_and_measure(%clear_macro, %array, %size, %result_addr, %message)
    li $t8, 10000                   # Number of iterations
    li $t9, 10000                   # Load constant 10,000 for division
    move $s0, $zero                 # Initialize accumulator for total time
repeat_clear:
    time($s1)                      # Start time
    la $a0, %array                  # Load address of array
    lw $a1, %size                   # Load size of array
    %clear_macro                   # Call the provided clear macro
    time($s1)                      # End time

    add $s0, $s0, $s1              # Accumulate time taken

    addi $t8, $t8, -1              # Decrement iteration count
    bnez $t8, repeat_clear         # If not done, repeat

    div $s0, $t9                   # Compute average time
    mflo $s1                       # Get result of division
    sw $s1, %result_addr           # Store average time

```



```
    print_result(%message, %result_addr)   # Print result
.end_macro

main:
    # Run and measure index-based approach 10,000 times
    run_and_measure(clear_array_index, array, size, avg_time1, avg_msg1)

    move $s1, $zero                 # Reset accumulator for total time again

    # Run and measure pointer-based approach 10,000 times
    run_and_measure(clear_array_pointer, array, size, avg_time2, avg_msg2)

    # Exit
    li $v0, 10
    syscall

```

Result



```
Average time for index-based approach: 7800Average time for pointer-based approach: 6089
-- program is finished running --

Average time for index-based approach: 7799Average time for pointer-based approach: 5970
-- program is finished running --
```

Explanation

The provided output shows the average time taken for two different methods to clear an array: an index-based approach and a pointer-based approach. The results indicate that the pointer-based approach is consistently faster than the index-based approach by a margin of around 1,500 to 2,000 units.

How:

Both methods aim to set each element of an array to zero.

1. Index-Based Approach: This method uses an index variable to iterate through the array. For each index, the program calculates the memory address of the corresponding array element and sets its value to zero. This involves additional arithmetic operations to determine the address of each element.
2. Pointer-Based Approach: This method uses a pointer to directly reference memory addresses. It starts at the address of the first element and moves through the array by incrementing the pointer. This approach directly accesses and modifies memory without the need for intermediate calculations.

Why:

Fewer Operations: The pointer method eliminates the need to calculate the address of each array element, which the index-based method requires in every loop iteration. This reduction in operations per iteration contributes to the overall faster execution time.