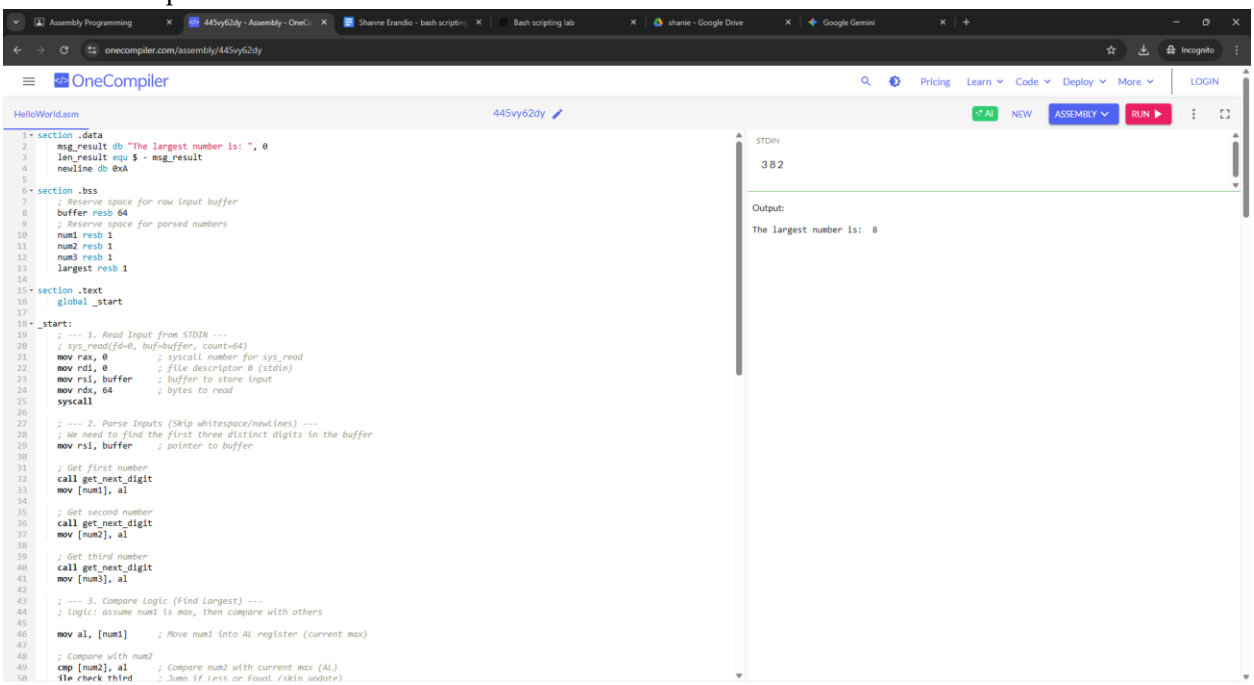
Name: Shanne Edralyn N. Erandio **Date: 11/27/2025**
Course, Year and Section: BSIT 2A

Assembly Programming
Case Study 5: Largest of Three Numbers

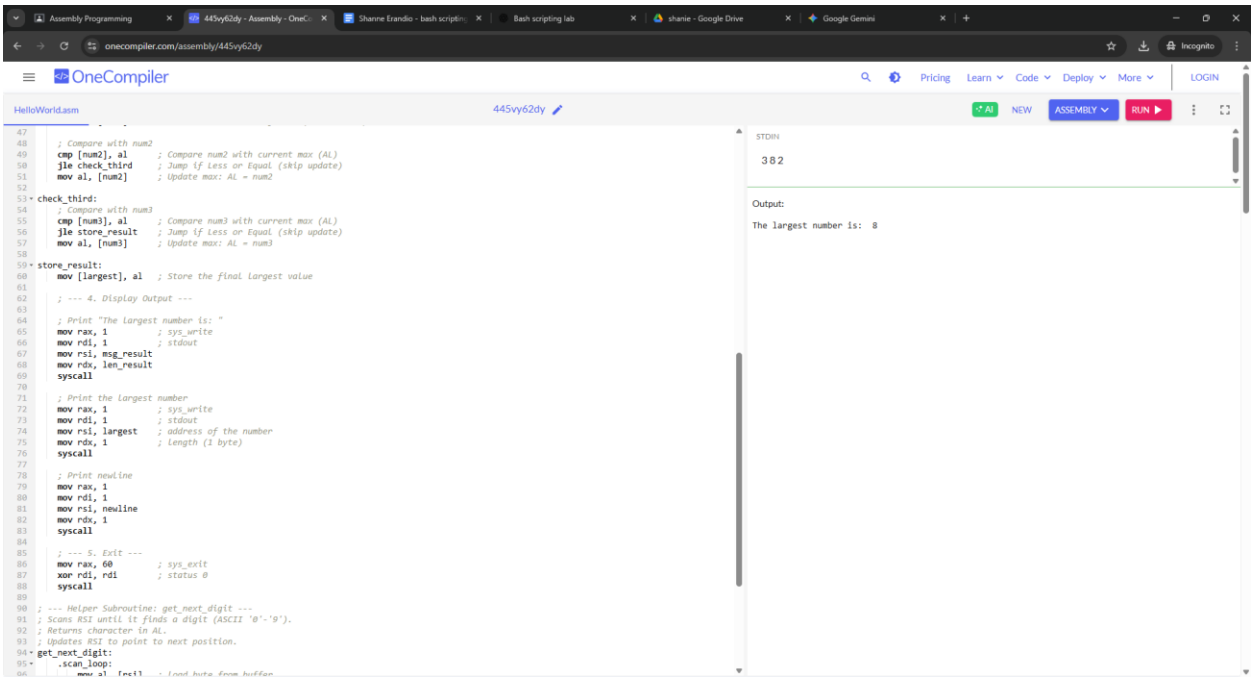
Write an Assembly program that:

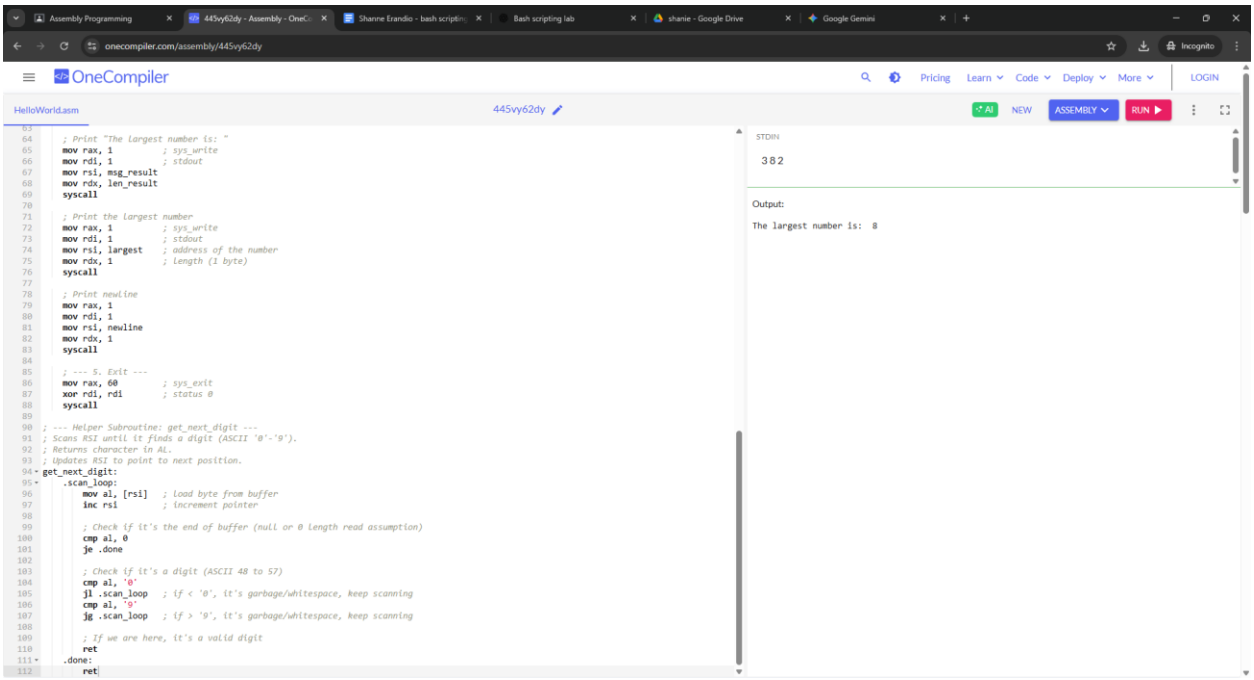
- Takes three single-digit inputs
- Compares them
- Displays the largest number
- Use conditional jumps (JG, JL, JE) or CMP instructions.

Code and Output




```
section .data
    msg_result db "The largest number is: ", 0
    len_result equ $ - msg_result
    newline db 0xA

section .bss
    ; Reserve space for raw input buffer
    buffer resb 64
    ; Reserve space for parsed numbers
    num1 resb 1
    num2 resb 1
    num3 resb 1
    largest resb 1

section .text
    global _start

_start:
    ; --- 1. Read Input from STDIN ---
    ; sys_read(fd=0, buf=buffer, count=64)
    mov rax, 0          ; syscall number for sys_read
    mov rdi, 0          ; file descriptor 0 (stdin)
    mov rsi, buffer     ; buffer to store input
    mov rdx, 64         ; bytes to read
    syscall

    ; --- 2. Parse Inputs (Skip whitespace/newlines) ---
    ; We need to find the first three distinct digits in the buffer
    mov rsi, buffer     ; pointer to buffer

    ; Get first number
    call get_next_digit
    mov [num1], al

    ; Get second number
    call get_next_digit
    mov [num2], al

    ; Get third number
    call get_next_digit
    mov [num3], al

    ; --- 3. Compare Logic (Find Largest) ---
    ; Logic: assume num1 is max, then compare with others

    mov al, [num1]      ; Move num1 into AL register (current max)

    ; Compare with num2
    cmp [num2], al      ; Compare num2 with current max (AL)
    jle check_third     ; Jump if Less or Equal (skip update)
```





In this activity, I learned how Assembly handles input, comparisons, and output at a very low level using system interrupts. Writing a program to determine the largest number helped me understand the use of CMP and conditional jump instructions like JG and JL. I also learned how ASCII values must be converted before comparing numbers. Overall, this exercise improved my understanding of how high-level logic is translated into step-by-step CPU operations.

The primary goal of this case study was to implement conditional logic in a low-level language. Unlike high-level languages where `if/else` structures are built-in, Assembly requires the programmer to manually manage control flow using the processor's status flags and jump instructions. The specific task was to determine the largest of three single-digit integers provided via standard input.

This exercise highlighted the trade-offs of Assembly language. While it offers precise control over registers and execution flow, it demands explicit management of trivial tasks like parsing ASCII characters and skipping whitespace. Mastering `CMP` and conditional jumps (`JG`, `JL`, `JE`) provides a fundamental understanding of how computer processors actually make decisions at the hardware level.