**PRACTICAL ONE**

**ST/CS/HND/22/092**

**2.**

**Assembly:** The Assembly code is very low-level and involves explicit manipulation of CPU registers (eax, ebx, ecx). It directly specifies machine instructions for multiplication and addition.

**C++:** The C++ code is high-level and uses common arithmetic operators (\* and +) to express the mathematical operation. It relies on the C++ compiler to generate the corresponding machine code.

**Exercise**

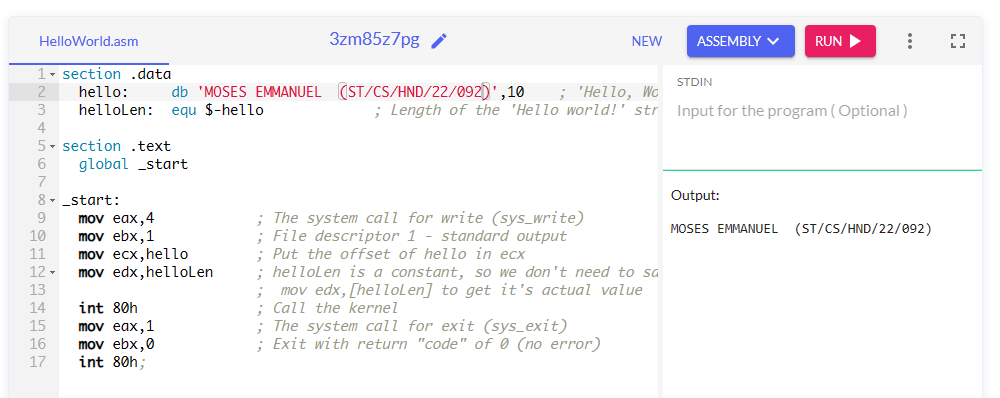
; Perform the calculation A \* B + C

imul eax, ebx ; Multiply EAX by EBX (result in EAX)

add eax, ecx ; Add ECX to EAX (final result in EAX)

**PRACTICAL TWO**

**ST/CS/HND/22/092**



**PRACTICAL THREE**

**ST/CS/HND/22/092**

**CODE**

program HexToDecimal;

#include("stdlib.hhf");

static

hexString: string := "1A"; // Replace with your hexadecimal input

decimalValue: int32;

begin HexToDecimal;

stdout.put("Enter a hexadecimal value: ");

stdin.get(hexString);

mov(hexString, esi); // Load the address of the input string into esi

call HexToDec;

// Print the decimal value

mov(eax, decimalValue);

stdout.put("Decimal equivalent: ");

stdout.puti(decimalValue, nl);

// Exit the program

mov(0, eax);

ret;

end HexToDecimal;

procedure HexToDec;

mov(0, eax); // Clear the eax register (will store the result)

xor(ecx, ecx); // Clear ecx (used for the loop counter)

mov(esi, edi); // Copy the address of the input string to edi

cld; // Set the direction flag to forward

repe scasb; // Find the null terminator

sub(esi, edi); // Calculate the length of the string

mov(0, ecx); // Reset the loop counter

mov(10, ebx); // Set ebx to 10 (base for hexadecimal)

xor(edx, edx); // Clear edx (used for accumulating the decimal result)

convertLoop:

lodsb; // Load the next character into al

cmp(al, 0); // Check for the null terminator

je done; // If null terminator, exit the loop

sub("0", al); // Convert ASCII character to integer

cmp(al, 9); // Check if the digit is less than 9

jbe isDigit; // If yes, it's a valid hexadecimal digit

sub("A"-"9"-1, al); // Adjust for letters A-F

isDigit:

imul(ebx, edx); // Multiply the current result by 10

add(eax, edx); // Add the current digit to the result

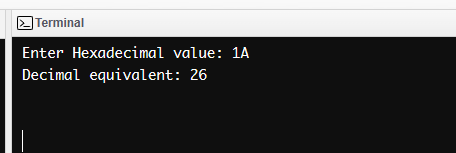
inc(ecx); // Increment the loop counter

jmp convertLoop; // Repeat for the next digit

done:

ret;

end HexToDec;



**PRACTICAL FIVE**

**ST/CS/HND/22/092**

READ n

STORE n

WHILE LOAD n

IF n == 0

GOTO done

ENDIF

LOAD v1

WRITE

READ

STORE n

GOTO while

done:

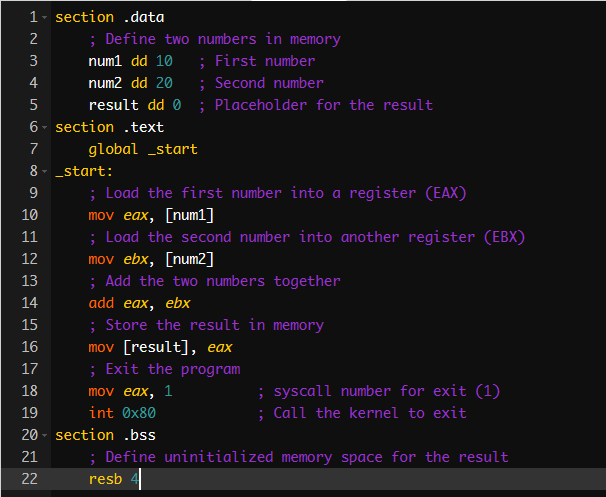
HALT

v1 = 1

n = 0

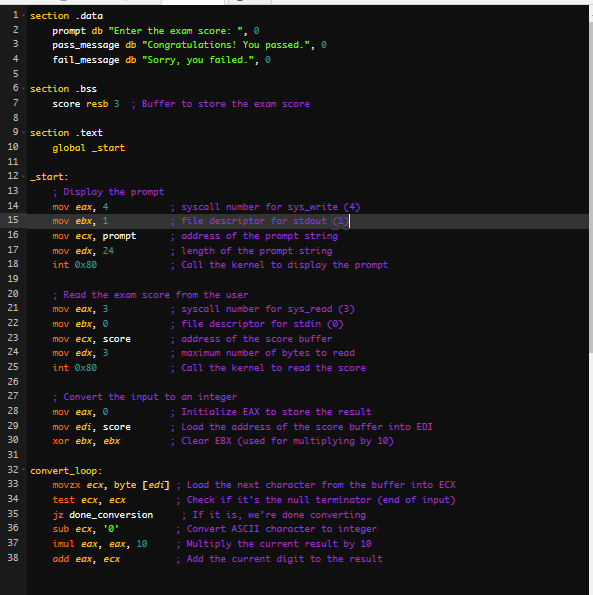
**PRACTICAL SIX**

**ST/CS/HND/22/092**



**PRACTICAL SEVEN**

**ST/CS/HND/22/092**



OUTPUT

**ST/CS/HND/22/092**

