Hands-On Activity 4.1

C TRANSLATION TO ASSEMBLY LANGUAGE

Course Code: CPE021	**Program:** Computer Engineering
Course Title: Computer Architecture and Organization	**Date Performed:** March 10, 2025
Section: CPE22S2	**Date Submitted:** March 10, 2025
Name: Adia, James Russel E.	**Instructor:** Engr. Maria Rizette H. Sayo

A. Procedure: Output(s) and Observation(s)

Sample Problem 1:

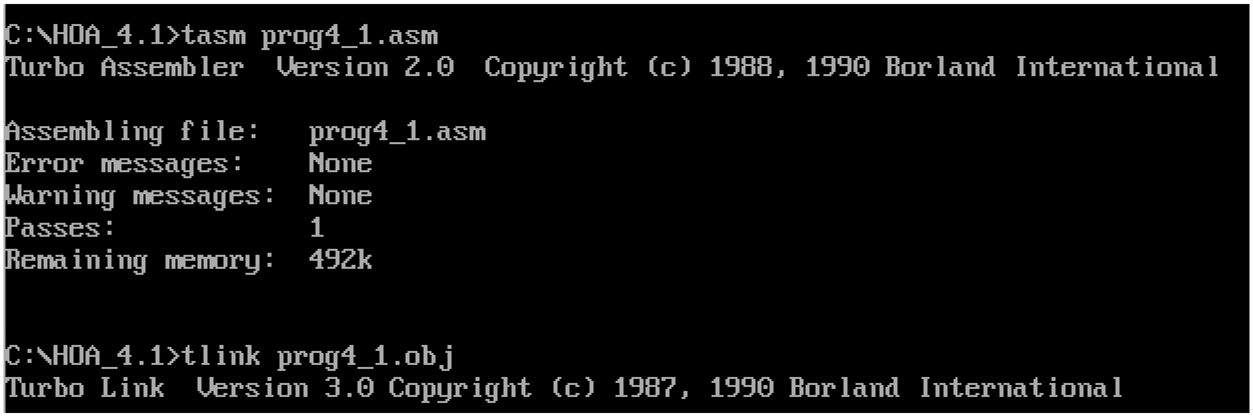
1. Type the following programs in Notepad.

```
TITLE prog4_1.asm
Dosseg
.model small
.stack 0100h
.data
.code
movax,@data
mov ds, ax
mov cx,001Eh
mov ah,02h      ;request display character
mov dl,'*'      ;character to display
A:      int 21h ;call interrupt service
        loop A
mov ax, 4c00h   ;end
int 21h
end
```

```
TITLE prog4_2.asm
.model small
.stack
.data
.code
movax,@data
mov ds, ax
mov cx,001Eh
mov ah,02h      ;request display character
movdl,'A'       ;character to display
B:      int 21h ;call interrupt service
inc dl
loop B
mov ax, 4c00h   ;end
int 21h
end
```

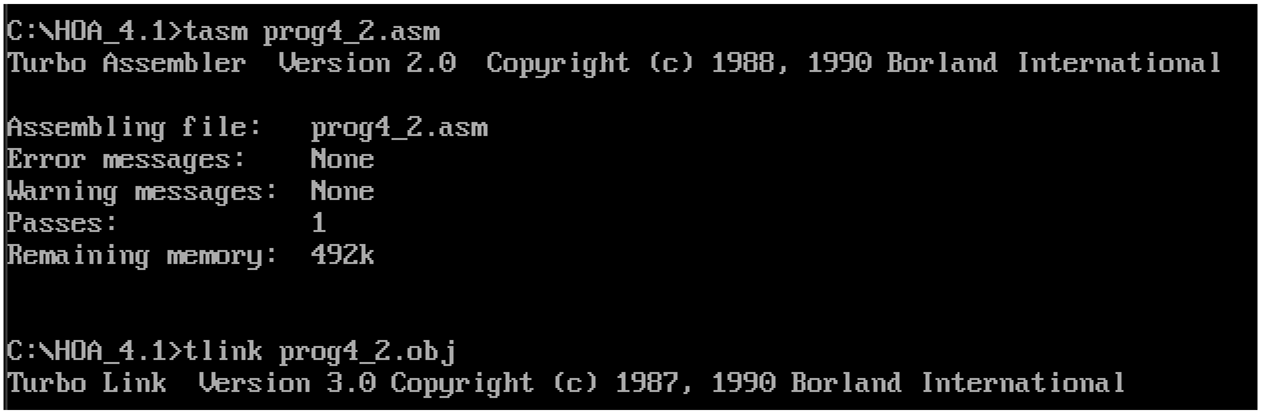
2. Assemble and execute these programs.

Assembling prog4_1.asm



```
C:\HOA_4.1>tasm prog4_1.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   prog4_1.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  492k


C:\HOA_4.1>tlink prog4_1.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```

Assembling prog4_2.asm

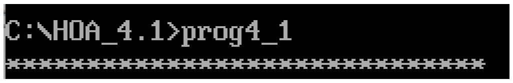
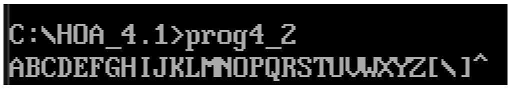


```
C:\HOA_4.1>tasm prog4_2.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   prog4_2.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  492k


C:\HOA_4.1>tlink prog4_2.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```

3. Analyze the outputs.
   What did you observe about the outputs?
   - For prog4_1.asm the program displayed the ‘*’ character 30 times before terminating since the cx register was loaded with the value 30 (1Eh in hexadecimal) which was used as the counter for the loop

   - For prog4_2.asm the program displayed the characters starting with the ASCII value of ‘A’ to the dl register while incrementing the value in dl 30 times since the cx register was also loaded the value 30 (1Eh in hexadecimal) which was also used as the counter for the loop.

4. Record the outputs in Table 4.1 and Table 4.2 respectively.

Table 4.1 - Output for prog4_1.asm	**Table 4.2 Output for prog4_2.asm**
 C:\HOA_4.1>prog4_1 ******************************	 C:\HOA_4.1>prog4_2 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^

Sample Problem 2:

1. Type the following programs in Notepad.

```
      TITLE Equal.asm
       MAIN SEGMENT
       ASSUME
CS:MAIN,DS:MAIN,ES:MAIN,SS:MAIN
      ORG 100h
      START:
              MOV DL,41h
              MOV DH,41h
              CMP DH,DL
              JE TheyAreEqual
              JMP TheyAreNotEqual
TheyAreNotEqual:
          MOV AH,02h
          MOV DL,4Eh
          INT 21h
```

```
// Equal.c
#include<stdio.h>
#include<conio.h>
main()
{
int DH,DL;
DL = 41;
DH = 41;
if (DH == DL)
printf("Y");
else
printf("N");
getch();
return 0;
```

```
          INT 20h
TheyAreEqual:
          MOV AH,02h
          MOV DL,59h
          INT 21h
          INT 20h
MAIN ENDS
END START
```

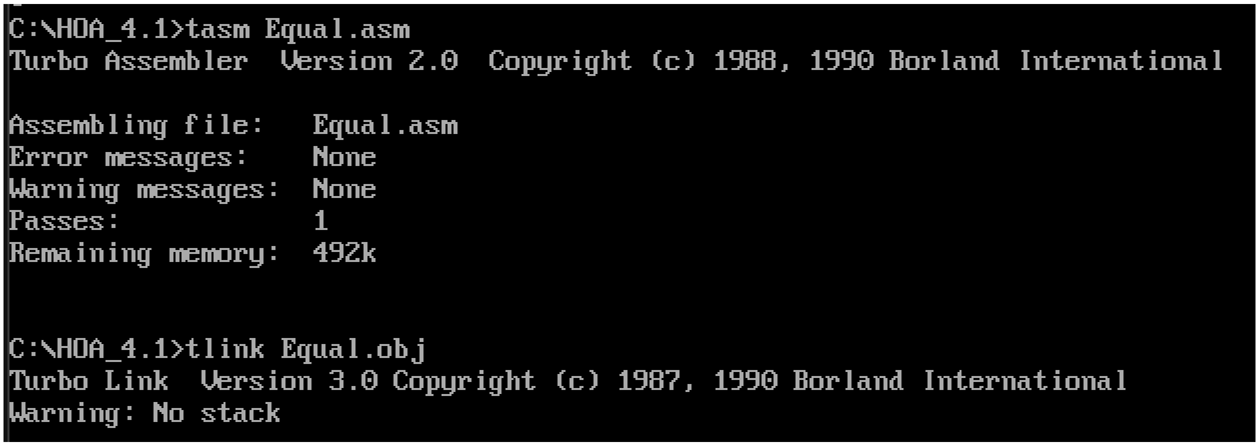
```
          }
```

```
TITLE Triangle.asm
.model small
.code
org 100h
start:

          mov cl,1
          mov bl,0
          mov ch,4

          looprow:cmp ch,0
          jgloopcol
          jmp quit

          loopcol:
          cmpbl,cl
          jldsplay
          jmp next

          dsplay:mov ah,2h
          mov dl,'*'    ;display asterisk
          int 21h
          incbl
          jmploopcol

          next:mov dl,0Ah
          int 21h       ;next line
          mov dl,0Dh
          int 21h

          mov bl,0
          decch
          inc cl
          jmplooprow
          quit:int 20h
          end start
```

```
//Triangle.c
#include<stdio.h>
#include<conio.h>
main()
{
          int z=1;int x=0;int y=4;



          while (y>0)
          {


           while(x<z)
              {




          printf("*");
              x++;
              }




          printf("\n");;
          x=0;y--;z++;
          }








          getch();
          return 0;}
```

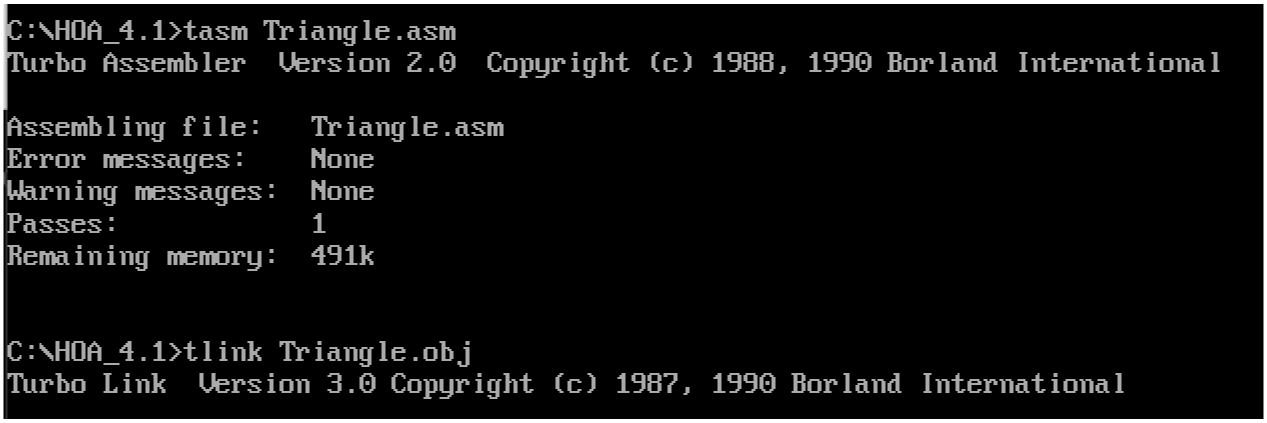
2. Assemble and execute each program.

Assembling Equal.asm



```
C:\HOA_4.1>tasm Equal.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   Equal.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  492k


C:\HOA_4.1>tlink Equal.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
Warning: No stack
```

Assembling Triangle.asm

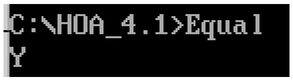
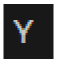
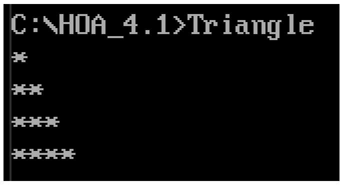


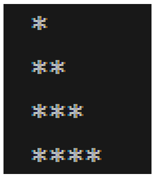
```
C:\HOA_4.1>tasm Triangle.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   Triangle.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  491k


C:\HOA_4.1>tlink Triangle.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```

3. Observe the output.
   What did you observe about the output?

   - What I observed about the outputs is that, the asm and c program outputs the same output meaning that the c program was translated to an asm program. I also observed that we can perform loops in assembly to perform repetitive tasks in the program just like other programming languages

4. Record the output in Table 4.3 and Table 4.4

Table 4.3 Output of Program Equal	**Table 4.4 Output of Program Triangle**
Assembly Program Output:  C:\HOA_4.1>Equal Y C Program Output:  Y	Assembly Program Output:  C:\HOA_4.1>Triangle * ** *** **** C Program Output:



```
*
**
***
****
```

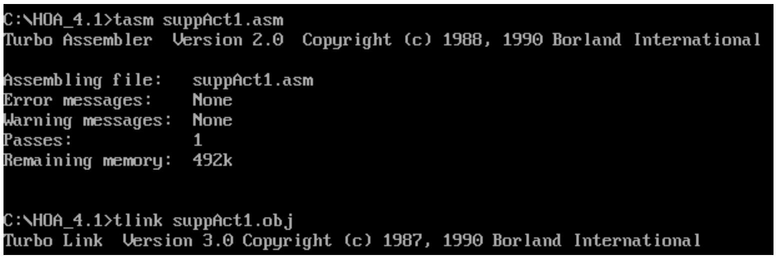
B. Supplementary Activity: Output(s) and Observation(s)

1. Translate the following C program to their equivalent assembly codes. Use the space provided.

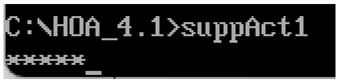
```
//Prog4_1.c

#include<stdio.h>
#include<conio.h>
main()
{
int cx;
for (cx=0;cx<5; cx++)
printf("*");
getch();
          return 0;

}
```

```
.model small
.stack 100h
.data
   asterisk db '*$'    ; Character to be printed with $ terminator

.code
main proc
   mov ax, @data       ; Initialize data segment
   mov ds, ax

   mov cx, 0           ; Initialize loop counter cx = 0

for_loop:
   cmp cx, 5           ; Compare cx with 5
   jge end_for         ; Jump to end if cx >= 5

   ; Print asterisk
   mov ah, 09h         ; DOS function to display a string
   mov dx, offset asterisk
   int 21h             ; Call DOS function

   inc cx              ; Increment counter (cx++)
   jmp for_loop        ; Continue loop

end_for:
   ; Wait for a key press (getch())
   mov ah, 01h         ; DOS function to read a character
   int 21h

   ; Exit program
   mov ah, 4ch         ; DOS function to exit program
   mov al, 0           ; Return code 0
   int 21h
main endp
end main
```

Assembling and Output:



```
C:\HOA_4.1>tasm suppAct1.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   suppAct1.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  492k


C:\HOA_4.1>tlink suppAct1.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```



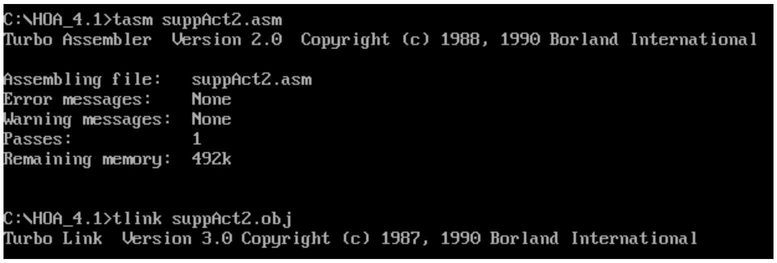
```
C:\HOA_4.1>suppAct1
*****_
```

```
//Prog4_2.c

#include<stdio.h>
#include<conio.h>
main()
{
void print();
print();
getch();
return 0;
}

void print()
{
int cx=1;
while (cx<=5){
printf("*");
 cx++;}
}
```

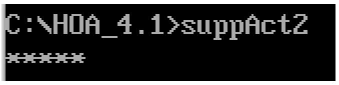
```
.model small
.stack 100h
.data
    asterisk db '*$'    ; Define asterisk character with $ terminator
for DOS output

.code
main proc
    mov ax, @data       ; Initialize data segment
    mov ds, ax

    call print_proc     ; Call the print function

    mov ah, 01h         ; Wait for a key press (equivalent to getch())
    int 21h

    mov ah, 4Ch         ; Return to DOS (equivalent to return 0)
    int 21h
main endp

print_proc proc
    mov cx, 1           ; Initialize cx = 1 (counter variable)

print_loop:
    cmp cx, 5           ; Compare cx with 5
    jg exit_print       ; If cx > 5, exit the loop

    ; Print asterisk
    mov ah, 09h         ; DOS function to print string
    mov dx, offset asterisk    ; Load address of asterisk (replaced
lea with mov offset)
    int 21h

    inc cx              ; Increment cx (cx++)
    jmp print_loop      ; Repeat the loop

exit_print:
    ret                 ; Return from procedure
print_proc endp
```

```
end main
```

Assembling and Output:



```
C:\HOA_4.1>tasm suppAct2.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   suppAct2.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  492k


C:\HOA_4.1>tlink suppAct2.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```

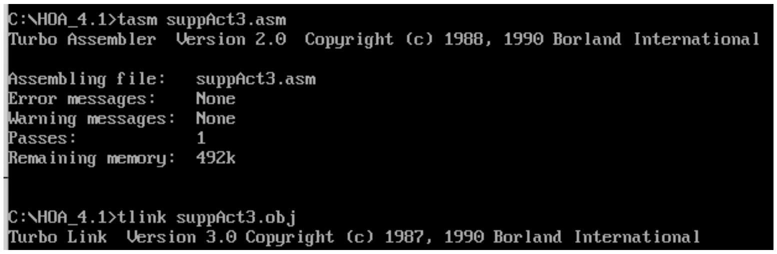


```
C:\HOA_4.1>suppAct2
*****
```

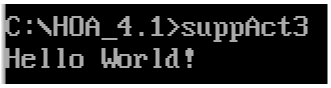
```
//Prog4_3.c

#include<stdio.h>
#include<conio.h>
main()
{
char message[]="Hello World!";
printf("%s",message);
getch();
return 0;
}
```

```
.model small
.stack 100h

.data
    message db 'Hello World!', '$'    ; Define string with $ terminator
for DOS output

.code
main proc
    ; Set up data segment
    mov ax, @data
    mov ds, ax

    ; Display message (printf equivalent)
    mov ah, 09h          ; DOS function for printing a string
    mov dx, offset message
    int 21h              ; Call DOS interrupt

    ; Wait for keypress (getch equivalent)
    mov ah, 01h          ; DOS function for reading a character
    int 21h              ; Call DOS interrupt

    ; Return to DOS (return 0 equivalent)
    mov ah, 4ch          ; DOS function to terminate program
    mov al, 00h          ; Return code 0
    int 21h              ; Call DOS interrupt
main endp
end main
```

Assembling and Output:



```
C:\HOA_4.1>tasm suppAct3.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   suppAct3.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  492k

C:\HOA_4.1>tlink suppAct3.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```



```
C:\HOA_4.1>suppAct3
Hello World!
```

2. Convert the each of the following C codes into its equivalent assembly code:

   a. if (ebx<=ecx) { eax=5;edx=6;}

```
cmp ebx, ecx        ; Compare ebx and ecx
    jg skip_block1      ; Jump if ebx > ecx (condition not met)
    mov eax, 5          ; Set eax = 5
    mov edx, 6          ; Set edx = 6
skip_block1:
```

   b. if (var1<=var2) var3=15; else var3=10;var4=20;

```
mov eax, [var1]     ; Load var1 into eax
    cmp eax, [var2]     ; Compare var1 and var2
    jg else_block       ; Jump if var1 > var2
    mov dword [var3], 15 ; var3 = 15
    jmp end_if          ; Skip else block
else_block:
    mov dword [var3], 10 ; var3 = 10
end_if:
    mov dword [var4], 20 ; var4 = 20 (always executed)
```

c. if (al>bl) && (bl=cl) x=1;

```
cmp al, bl          ; Compare al and bl
    jle skip_block3     ; Jump if al <= bl (first condition fails)
    cmp bl, cl          ; Compare bl and cl
    jne skip_block3     ; Jump if bl != cl (second condition fails)
    mov dword [x], 1    ; Set x = 1 if both conditions are true
skip_block3:
```

d. if (al >bl) || (bl> cl) x=1;

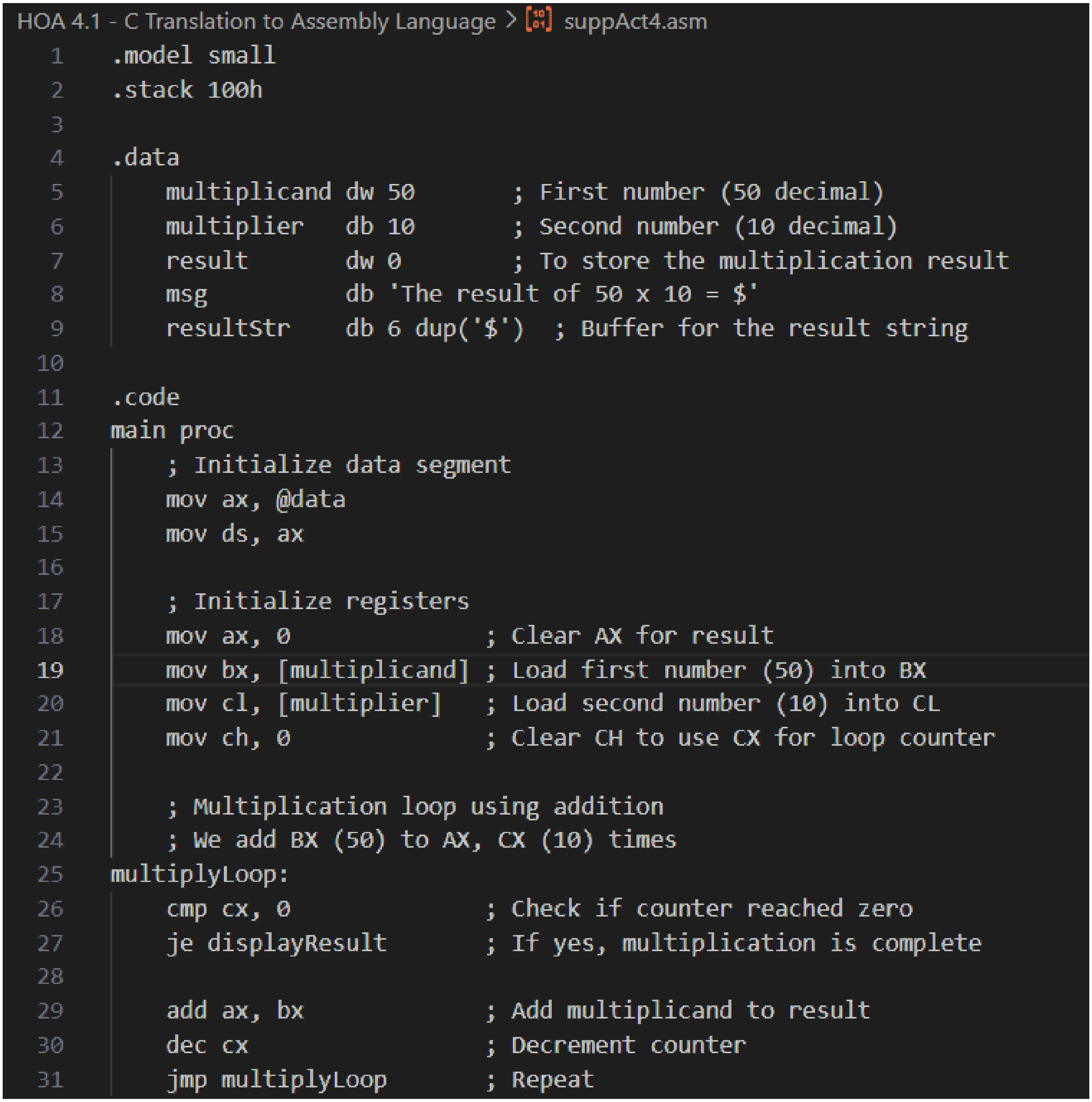
```
cmp al, bl          ; Compare al and bl
    jg set_x            ; Jump if al > bl (first condition true)
    cmp bl, cl          ; Compare bl and cl
    jle skip_block4     ; Jump if bl <= cl (both conditions fail)
set_x:
    mov dword [x], 1    ; Set x = 1 if either condition is true
skip_block4:
```

e. while (eax<ebx) eax =eax +1;

```
while_loop:
    cmp eax, ebx        ; Compare eax and ebx
    jge end_while       ; Jump if eax >= ebx (loop condition false)
    inc eax             ; Increment eax by 1
    jmp while_loop      ; Return to loop condition check
end_while:
```

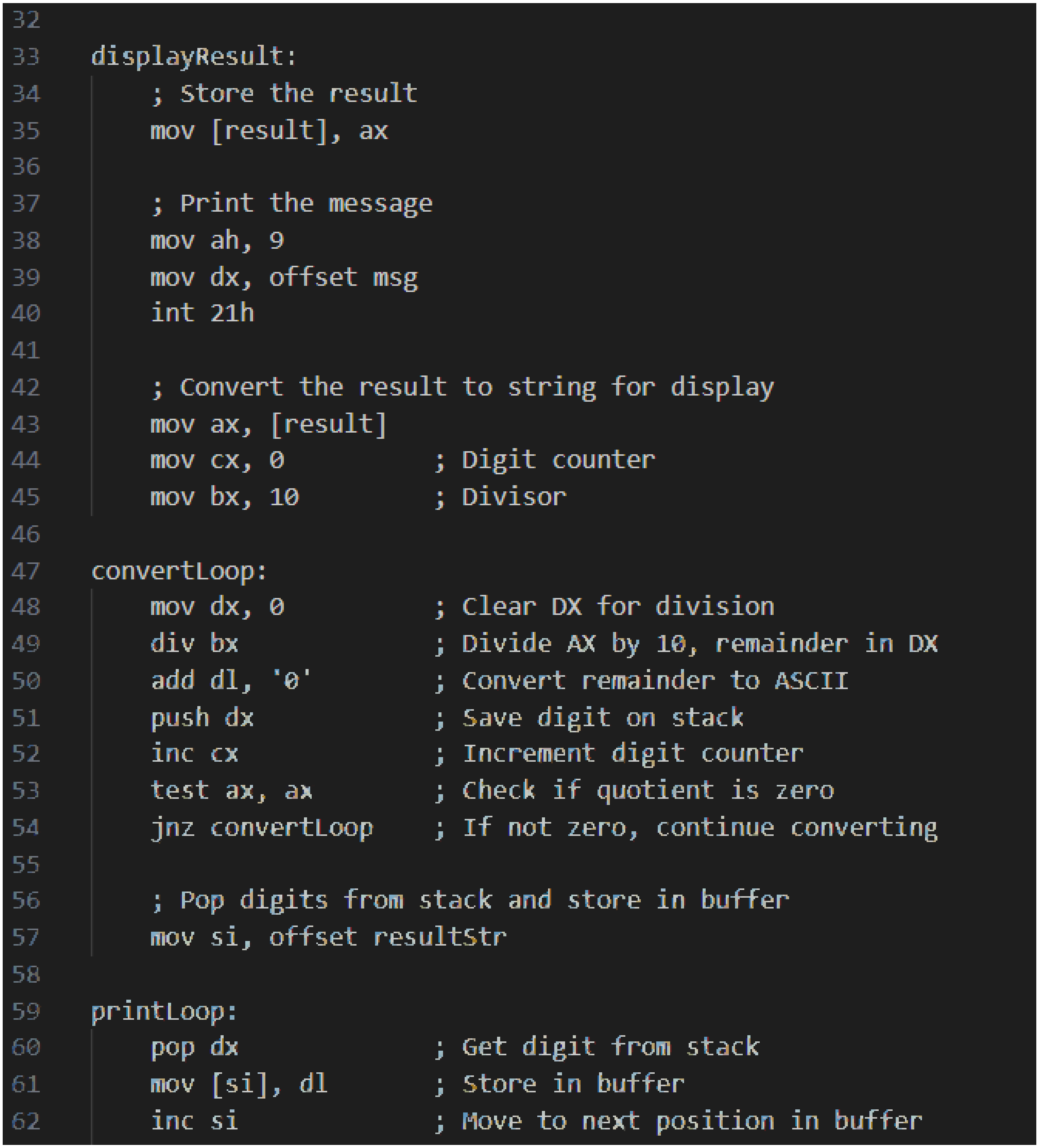
3. Show a program that multiples 50 (decimal) and 10 (decimal) without using the MUL and IMUL instructions.

Program Screenshot (.asm was viewed in VSCode for better readability)

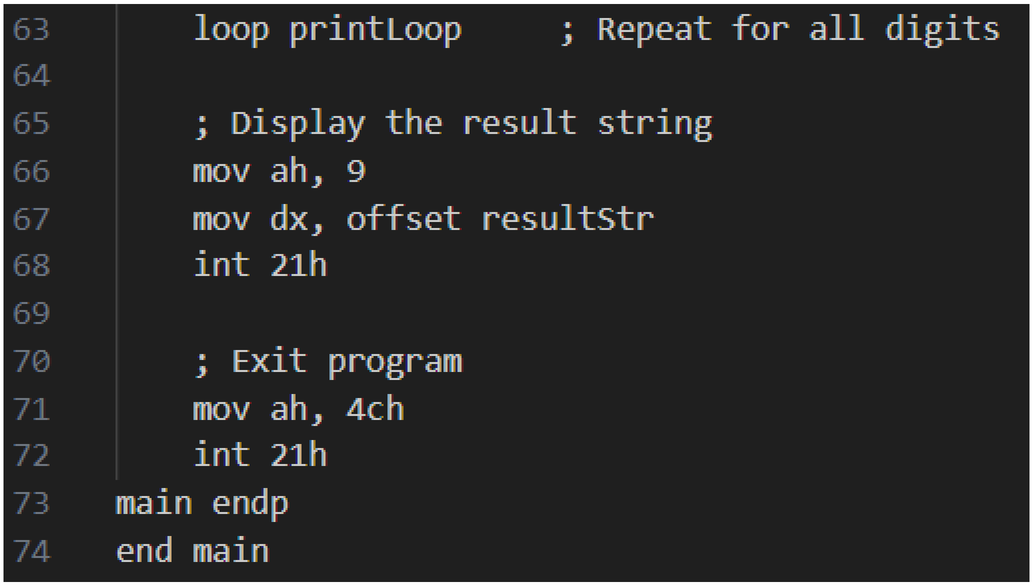


HOA 4.1 - C Translation to Assembly Language > suppAct4.asm

```
.model small
.stack 100h

.data
    multiplicand dw 50       ; First number (50 decimal)
    multiplier   db 10       ; Second number (10 decimal)
    result       dw 0        ; To store the multiplication result
    msg          db 'The result of 50 x 10 = $'
    resultStr    db 6 dup('$')  ; Buffer for the result string

.code
main proc
    ; Initialize data segment
    mov ax, @data
    mov ds, ax

    ; Initialize registers
    mov ax, 0              ; Clear AX for result
    mov bx, [multiplicand] ; Load first number (50) into BX
    mov cl, [multiplier]   ; Load second number (10) into CL
    mov ch, 0              ; Clear CH to use CX for loop counter

    ; Multiplication loop using addition
    ; We add BX (50) to AX, CX (10) times
multiplyLoop:
    cmp cx, 0              ; Check if counter reached zero
    je displayResult       ; If yes, multiplication is complete

    add ax, bx             ; Add multiplicand to result
    dec cx                 ; Decrement counter
    jmp multiplyLoop       ; Repeat
```

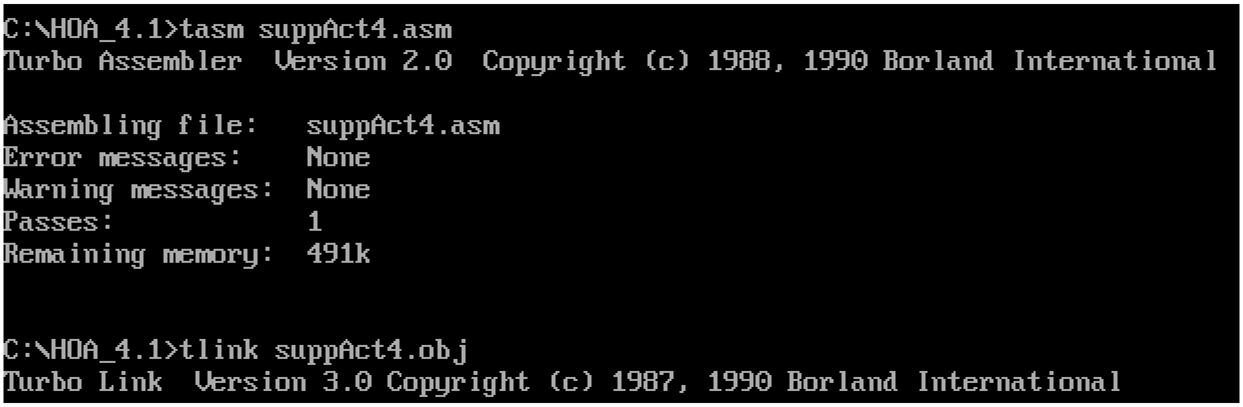


```
    displayResult:
        ; Store the result
        mov [result], ax

        ; Print the message
        mov ah, 9
        mov dx, offset msg
        int 21h

        ; Convert the result to string for display
        mov ax, [result]
        mov cx, 0          ; Digit counter
        mov bx, 10         ; Divisor

    convertLoop:
        mov dx, 0          ; Clear DX for division
        div bx             ; Divide AX by 10, remainder in DX
        add dl, '0'        ; Convert remainder to ASCII
        push dx            ; Save digit on stack
        inc cx             ; Increment digit counter
        test ax, ax        ; Check if quotient is zero
        jnz convertLoop    ; If not zero, continue converting

        ; Pop digits from stack and store in buffer
        mov si, offset resultStr

    printLoop:
        pop dx             ; Get digit from stack
        mov [si], dl       ; Store in buffer
        inc si             ; Move to next position in buffer
```



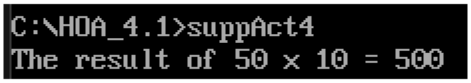
```
    loop printLoop     ; Repeat for all digits

    ; Display the result string
    mov ah, 9
    mov dx, offset resultStr
    int 21h

    ; Exit program
    mov ah, 4ch
    int 21h
main endp
end main
```

Assembling Program:



```
C:\HOA_4.1>tasm suppAct4.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   suppAct4.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  491k


C:\HOA_4.1>tlink suppAct4.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```

Output:



```
C:\HOA_4.1>suppAct4
The result of 50 x 10 = 500
```

C. Conclusion & Lessons Learned

In conclusion, the hands-on activity provided me with a practical understanding of the differences between C programming and Assembly programming. By doing the procedures and supplementary activities, I was able to directly compare a high-level programming language such as C with the low-level operations of Assembly. This made me realize that high-level languages are relatively easy to understand than low-level languages since they are closer to human language. The activity also required me to convert a C program into its Assembly equivalent, which reinforced my comprehension of both languages. This conversion process highlighted how detailed you can get in assembly programming. Overall, I was able to successfully do the tasks required and achieve the intended learning outcomes of this hands-on activity.