Activity No. 4	
C TRANSLATION TO ASSEMBLY LANGUAGE	
Course Code: CPE021A	**Program:**
Course Title: Computer Systems Organization with Assembly Language	**Date Performed:**
Section:	**Date Submitted:**
Name:	**Instructor:**

1. Objective:

This activity aims to show the relationship of C programming language to assembly language

2. Intended Learning Outcomes (ILOs):

After completion of this activity the students should be able to:

2.1 Compare C programming and Assembly programming

2.2 Convert a C program to Assembly program

3. Discussion :

ASSEMBLY LANGUAGE AND THE C LANGUAGE

Assembly language is the basis of the C programming language it is the reason why a program in C can be easily translated in assembly language.

For example, conditional statement of C language of the form:

```
if (expression)
{
statement1;
statement2;
:
statementn;
}
else
{
statement1;
statement2;
:
statementn;
}
```

Can be implemented in Assembly language as:

```
CMP_ instruction
Conditional_loop instruction <label>
Instruction1
Instruction2
:
Instruction
JMP_instruction<label>
```

The WHILE statement in C language of the form:

```
do
{
Statement1;
Statement2;
:
statementn
} while (expression);
```

Can be implemented in Assembly language as:

```
label:
      Instruction1
Instruction2:
Instruction
CMP instruction
Conditional_jump instruction
Label
```

The DO WHILE statement in C language of the form:

```
While
{
Statement1;
Statement2;
:
statementn
}
```

Can be implemented in Assembly language as:

```
Label1:
CMP instruction
Conditional jmp instruction label
Instruction1:
Instruction2
:
instructionn
JMP label
Conditional_jump instruction
Label2:
Instruction1:
Instruction2
:
Instruction
```

Label

Acts as an identifier that acts as a place marker for instructions and data. When placed just before an instruction implies the instruction's address. If placed just before a variable implies the variable's address.

Loops

Loops or repetition allow a set of instructions to be repeated until certain condition is reached is also used in Assembly using the LOOP command.

4. Resources:

Computer with 32-bit Operating System
TASM

5. Procedure:

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.
3. Analyze the outputs.
   What did you observe about the outputs?
   _____
   _____
   _____
4. Record the outputs in Table 4.1 and Table 4.2 respectively.

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
```

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

```
TITLE Triangle
.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.

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

   _____
   _____
   _____

4. Record the output in Table 4.3 and Table 4.4

6. DATA ANALYSIS:

Table 4.1- Output of prog4_1.asm	*Table 4.2- Output of prog4_2.asm*

Table 4.3 Output of Program Equal	*Table 4.4 Output of Program Triangle*

7. PROBLEMS:

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;
}
```

```
//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++;}
}
```

```
//Prog4_3.c

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

2. Convert the each of the following C codes into its equivalent assembly code:
    a. if (ebx<=ecx) { eax=5;edx=6;}

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

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

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

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

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

8. CONCLUSIONS:

9. Assessment (Rubric for Laboratory Performance):

RUBRIC FOR CONDUCT OF PROGRAMMING EXERCISES

Student Outcome: Synthesize a programming solution using the necessary algorithm, and develop the skills based on the topics discussed on the activities.

Name: _____ Program:_____ Course: _____ Section: _____ ___Semester, School Year ____

Performance Indicators	Beginner 1	Acceptable 2	Proficient 3	Score
Algorithm	The student was able to provide few of the correct algorithms needed for the given application programs.	The student was able to provide most of the correct algorithms needed for the given application programs.	The student was able to provide ALL of the correct algorithms needed for the given application programs.	
Program Code	The student was able to provide few of the correct program codes needed for the given application programs.	The student was able to provide most of the correct program codes needed for the given application programs.	The student was able to provide ALL the correct program codes needed for the given application programs.	
Questions to be answered	The student answered few of the given questions on the activity correctly.	The student answered most of the given questions on the activity correctly.	The student answered ALL the questions on the activity correctly.	
Timeliness	The student accomplished few of the requirements needed for the activity within the given period.	The student accomplished most of the requirements needed for the activity within the given period.	The student accomplished ALL the requirements needed for the activity within the given period.	
			Total Score	
			Mean Score = (Total Score / 3)	
			Percentage Score = (Total Score / 12) x 100%	

Evaluated by:

_____ _____

Printed Name and Signature of Faculty Member Date