# Lab Manual for Computer Organization and Assembly Language

## Lab-13

Interfacing with HLL (High Level Languages)

# Contents

# 1. Introduction

To interface an assembly routine with a high-level language program means to be able to call functions back and forth. And to be able to do so requires knowledge of certain behavior of the HLL when calling functions. This behavior of calling functions is called the calling conventions of the language. Two prevalent calling conventions are the C calling convention and the Pascal calling convention.

# 2. Objective of Lab

After completing this lab, the student should be able to:

- Understand the purpose/ advantage of interfacing in assembly language.
- Understand how to convert C++ and assembly language codes.
- Understand naming conventions used in interfacing.
- Understand how parameters are passed to a routine.
- Understand the working of registers in interfacing.

# 3. Concept Map

To interface an assembly routine with a high-level language program means to be able to call functions back and forth. And to be able to do so requires knowledge of certain behavior of the HLL when calling functions. This behavior of calling functions is called the calling conventions of the language. Two prevalent calling conventions are the C calling convention and the Pascal calling convention.

## 3.1 What is the naming convention?

C prepends an underscore to every function or variable name while Pascal translates the name to all uppercase. C++ has a weird name mangling scheme that is compiler dependent. To avoid it C++ can be forced to use C style naming with extern “C” directive.

## 3.2 How are parameters passed to the routine?

In C parameters are pushed in reverse order with the rightmost being pushed first. While in Pascal they are pushed in proper order with the leftmost being pushed first.

### 3.3 Which registers must be preserved?

Both standards preserve EBX, ESI, EDI, EBP, ESP, DS, ES, and SS.

### 3.4 Which registers are used as scratch?

Both standards do not preserve or guarantee the value of EAX, ECX, EDX, FS, GS, EFLAGS, and any other registers.

### 3.5 Which register holds the return value?

Both C and Pascal return up to 32bit large values in EAX and up to 64bit large values in EDX:EAX.

### 3.6 Who is responsible for removing the parameters?

In C the caller removes the parameter while in Pascal the callee removes them. The C scheme has reasons pertaining to its provision for variable number of arguments.

## 4. Walkthrough Task

Write a program in which inline assembly is used to calculate greatest common divisor.

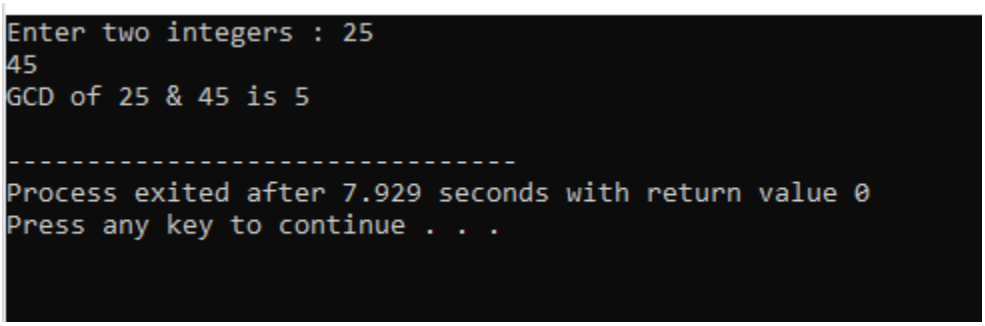
```
#include <stdio.h>

int gcd( int a, int b ) {
    int result ;
    /* Compute Greatest Common Divisor using Euclid's Algorithm */
    __asm__ __volatile__ ( "movl %1, %%eax;"
                           "movl %2, %%ebx;"
                           "CONTD: cmpl $0, %%ebx;"
                           "je DONE;"
                           "xorl %%edx, %%edx;"
                           "idivl %%ebx;"
                           "movl %%ebx, %%eax;"
                           "movl %%edx, %%ebx;"
                           "jmp CONTD;"
                           "DONE: movl %%eax, %0;" : "=g" (result) : "g" (a), "g" (b)
    );

    return result ;
}

int main() {
    int first, second ;
    printf( "Enter two integers : " ) ;
    scanf( "%d%d", &first, &second );
    printf( "GCD of %d & %d is %d\n", first, second, gcd(first, second) ) ;
    return 0 ;
}
```

When the above code is compiled and executed, it produces the following **Output.**




## 5. Practice Tasks

This section will provide more practice exercises which you need to finish during the lab. You need to finish the tasks in the required time. When you finish them, put these tasks in the following folder: \\fs\assignments$\

### 5.1 Practice Task 1 [Expected time = 15mins]

Write a program in which when user enters an integer and passes it to a function, this function should decide either it is even or odd. Use assembly code inside this function.

### 5.2 Practice Task 2 [Expected time = 15mins]

Write a program in which when user enters 2 integers and passes them to a function, this function should calculate the greatest integer among them. Use assembly code inside this function to find out the greatest integer.

### 5.3 Practice Task 3 [Expected time = 15mins]

Write a program to print all prime integers from 0 to 50. Use assembly code inside C++ code to find out the prime integers.

### 5.4 Out comes

After completing this lab, student will be able to setup emu 8086. He/ She will also be able to compile and run basic Assembly programs.

## 6. Evaluation Task (Unseen) [Expected time = 30mins for tasks]

The lab instructor will give you unseen task depending upon the progress of the class.

## 7. Evaluation criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

**Table 3: Evaluation of the Lab**

| Sr. No. | Task No | Description | Marks |
|---|---|---|---|
| 1 | 4 | Problem Modeling | 20 |
| 2 | 6 | Procedures and Tools | 10 |
| 3 | 7 | Practice tasks and Testing | 35 |
| 4 | 8 | Evaluation Tasks (Unseen) | 20 |
| 5 | | Comments | 5 |
| 6 | | Good Programming Practices | 10 |

## 8. Further Reading

This section provides the references to further polish your skills.

### 8.1 Slides

The slides and reading material can be accessed from the folder available at \\fs\lectures$