**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Dr. Arbab Latif  **Dated:** 20/05/2022

**Semester:** 4th **Section:** BEE 12C

**Group No.: 7**

**EE-222: Microprocessor Systems**

Lab 12: 8086 Assembly Language

Group Members

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Reg. No** | **Report**  **Marks / 10** | **Viva**  **Marks / 5** | **Total**  **Marks / 15** |
| Danial Ahmad | 331388 |  |  |  |
| Muhammad Umer | 345834 |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Experiment # 12

# Introduction to Assembly Language and Turbo Assembler (TASM)

## Objectives

1. Getting introduced to assembly language
2. Learning some basic commands
3. Introduction to the syntax of assembly language programming
4. Learning the use of turbo assembler (TASM)

## Equipment

Software

* *Turbo assembler (TASM) or Microsoft assembler (MASM)*

## Discussion

Before starting coding in assembly we should get familiarized with some basic coding parameters, assembly language syntax and some basics of microprocessors.

## Introduction to Registers

There are four type of registers in microprocessors:

1. AX
2. BX
3. CX
4. DX

AX, BX are mainly used for arithmetic operations and saving address. CX is used for saving the values for counts which is used in executing loop instructions and DX is mainly used for I/O operations.

### Program Structure

## Memory Models

The size of code and data a program can have is determined by specifying memory model using the .MODEL directive. The models used are SMALL, LARGE, and HUGE but the appropriate one is .SMALL. The model directive should come before any segment definition.

## Data Segment

The data segment is used for all the variables definitions. We use .DATA directive followed by variable and constant declaration.

## Stack Segment

The purpose of stack segment is to set aside a block of memory to store the stack. The declaration syntax is:

**.stack size**

If we write:

**.stack 100h**

100 bytes would be reserved for stack. If size omitted 1KB is the default size.

## Code Segment

Code segment contains all the programming instructions. The declaration syntax is:

**.code**

So with minor variations this form may be used

**.model small**

**.stack 100h**

**.data**

**; data definitions go here**

**.code main proc ; instructions go here**

**main endp**

**end main**

Here main proc and main endp delineate the procedure. The last line here should be END directive followed by name of main procedure.

## INT 21h

INT 21h may be used to invoke a large number of DOS functions; a particular function is requested by placing a function number in AH register and invoking INT 21h. Here we are interested in following functions.

|  |  |
| --- | --- |
| **Function Number** | **Functions** |
| 1 | Single Key Input |
| 2 | Single Character Output |
| 9 | Character String Output |

## Getting Started With TASM

* Open the command prompt and switch to the directory where TASM has been installed
* Go to BIN i.e. the address of the directory may look like C:\TASM\BIN
* You can open TASM\BIN\B32TOOLS
* Each step with screenshot is given to make each step easily understandable
* Enter the edit command, that will open the TASM file editor
* Write your code and save your program in the bin directory of TASM, the file extension should be “.asm” as you are programming in assembly language
* Exit the editor
* Now enter “TASM ABC.ASM”
* This will compile your program and will show if there are some errors or warnings against your compiled code
* See the error message and trace it back in your code by again going to the editor
* If there is no error then enter “tlink ABC”, this will generate the “exe” file against your code, by just entering the name of the file i.e. ABC now your program will be executed
* Follow the above steps and execute the required code

# Sample Code

## Program Description

Start with displaying a ”?” and then reading a character from keyboard and displaying it on next line.

## Code

.model small

.stack 100h

.data

.code

main proc

    mov ah,2

    mov dl,'?'

    int 21h

    mov ah,1

    int 21h

    mov bl,al

    mov ah,2

    mov dl,0dh

    int 21h

    mov dl,0ah

    int 21h

    mov dl,bl

    int 21h

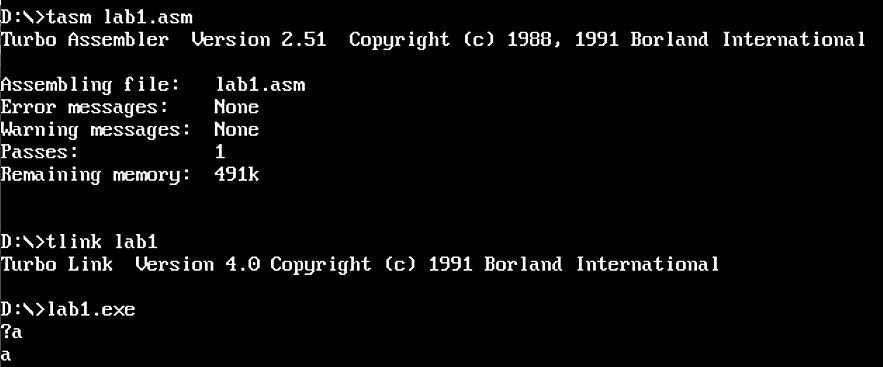
    mov ah,4ch

    int 21h

main endp

end main

## Output



### Exercises

1. Edit the example code above and now move 3 into dl register and then run the program. Observe what is the output and why?

.model small

.stack 100h

.data

.code

main proc

    mov ah,2

    mov dl,'3'

    int 21h

    mov ah,1

    int 21h

    mov bl,al

    mov ah,2

    mov dl,0dh

    int 21h

    mov dl,0ah

    int 21h

    mov dl,bl

    int 21h

    mov ah,4ch

    int 21h

main endp

end main

## Output



1. Manipulate the example code so that the user input that will now be printed on first position of first line. (Hint: the ? will be override by the user input)

.model small

.stack 100h

.data

.code

main proc

    mov dl,'?'

    mov ah,2

    int 21h

    mov ah,1

    int 21h

    mov bl,al

    mov dl,08h

    mov ah,2

    int 21h

    mov dl,20h

    int 21h

    mov dl,0Dh

    int 21h

    mov dl,bl

    mov ah,2

    int 21h

    mov ah,4ch

    int 21h

main endp

end main

## Output

