# 16.317: Microprocessor Systems Design I

Summer 2012

Lab 3: Assembly Language Programming
Due **Monday, 7/30/12**

## Introduction

In this lab, you will learn how to develop and debug your own assembly language program. This program will sort three positive hexadecimal integers, each of which can range from 0 to F. The program should follow the steps listed below:

1. Print a prompt ("Enter first number") on the screen; get ready for the first input.
2. Get the first integer from the keyboard. The input will be a one-digit **hexadecimal** number ranging from 0 to F.
3. Print a prompt ("Enter second number") on the screen; get ready for the second input.
4. Obtain the second **hexadecimal** number (0 through F) from the keyboard.
5. Print a prompt ("Enter third number") on the screen, get ready for the third input.
6. Obtain the third **hexadecimal** number (0 through F) from the keyboard.
7. Sort these three numbers.
8. Print the sorted numbers on the screen in **ascending** order. The results should also be in **hexadecimal**.

For up to **20 bonus points,** modify your program to handle two-digit **hexadecimal** numbers (such as "2F") as inputs.

**Remember that each student must generate an individual report that follows the guidelines listed in the "Lab Report Format Guidelines", and that the last page of this assignment contains a cover page each student must use as the first page of that report.**

## Reference

Walter A. Triebel and Avtar Singh, Lab Manual to Accompany the 8088 and 8086 Microprocessors, Prentice Hall, ISBN 0-13-012843-0.

# Part 1: Input/Output Tutorial

Input and output can be handled using a software interrupt (interrupt #21H) that invokes DOS routines. The number of the routine, which is passed in register AH, specifies the function to be performed by the operating system. The examples below show how to use this interrupt to print a string, read a single character, and print a single character.

*Printing a message to the screen*

The following small program shows how to print a line of message on the screen, using the DOS service with function number 9. The string must end with '$'. If you'd like to print a character, please refer to the section below on printing a single character.

```
.model small
.stack
.data
Message db "Hello World!$"      ; message to be displayed
.code
mov dx,OFFSET Message           ; offset of Message in DX
mov ax,SEG Message              ; segment of Message in AX
mov ds,ax                       ; DS:DX points to string
mov ah,9                        ; function 9 –
                                ;    display string
int 21h                         ; call DOS service to
                                ;    print the string
mov ax,4c00h                    ; return to dos DOS
int 21h
END start                       ;end here
```

*Reading a single character*

Invoking the DOS service with function number 1 will call a function that waits for a key to be pressed, then stores the ASCII value of the entered character in AL.

```
mov ah, 1                       ; function 1h - get
                                ;    character
int 21h                         ; call DOS service to
                                ;    get a character
                                ; Character will be in AL
                                ;    at this point
```

*Writing a single character*

Function number 2 will print a character to the screen, assuming the ASCII value of that character is stored in AL when the interrupt is invoked.

```
mov dl, al                      ; move al (ascii code)
                                ;    into dl - required
                                ;    for function 02h
mov ah, 02h                     ; function 2h – print
                                ;    character
int 21h                         ; call DOS service to
                                ;    print a character
```

# Part 2: Creating your Program

We suggest following the steps and hints below to create the program described in the introduction:

1. Make sure you clearly understand the problem and have completed the tutorial.
2. The program **`Lab3sk2.asm`** posted on the website contains skeleton code that can be modified to complete this assignment.
3. Once the inputs have been obtained the keyboard, you must convert them from ASCII code to the actual numeric value. For example, "1" entered from the keyboard is coded as 31H in ASCII, but your program should work with that number as a numeric value. Consider using a subroutine to do this conversion. You can assume only numbers and uppercase letters are possible inputs.
4. Make sure you understand the algorithm for sorting three numbers. Draw a flowchart to guide your implementation of this algorithm in assembly language. **You must include this flowchart in your final report.**
5. After completing your `.asm` file, you can assemble and link it to generate an executable, which you can then run using DEBUG. Chapter 7 of the text contains details on assembly and linking, but we're doing things slightly differently ***(edited 4/4/2012)***:

   - The assembler name is `TASM`, and can be found on any of the lab machines in the directory `C:\TASM`. This program will convert your assembly code to an object format, and will allow you to discover any syntax errors in your code.
   - To generate an object (`.obj`) file with the same name as your .asm file, type `TASM <.asm file name>`. In the next step, you will link your `.obj` file to create an executable.
     - You may need to specify the full path for the assembler (`C:\TASM\TASM`) and/or your `.asm` file
     - Example: `C:\TASM\TASM Lab3sk2.asm`
       - This example assumes you're in the directory containing `Lab3sk2.asm`.
   - To specify a different name for your `.obj` file, type that new file name after the name of the `.asm` file.
     - Example: `C:\TASM\TASM Lab3sk2.asm newfile.obj`
   - If you would like to generate a listing file (`.lst`) as well, type the name of the desired listing file after the `.obj` file name:
     - Note that you must list an object file name before the listing file name—otherwise, the assembler assumes that the second file is your object file.
     - Example: `C:\TASM\TASM Lab3sk2.asm Lab3.obj Lab3.lst`
   - To create an executable (`.exe` file) that can be tested using DEBUG, use the linker `TLINK`, which is found in the same directory as the assembler:
     - Example: `C:\TASM\TLINK Lab3.obj Lab3.exe`

## *Part 2 check off and deliverables*

**To check off, demonstrate your running code to an instructor.**

**You must also submit an electronic copy of your `.asm` file to the TA or professor. The TA will have a USB drive in the lab on which your files can be stored. Failing to submit your code will result in a grade of 0 for this assignment.**

**Also, please note that you must submit the following with your report:**

- **A hard copy of your source code (`.asm` file)**
  - **Your name and your partner's name (if applicable) should be at the top of the file.**
  - **You should comment your code well. Remember that each comment starts with a semicolon.**
- **A flowchart that shows, at a minimum, the process for sorting the integers.**

**I strongly suggest reading the grading rubric on the next page to ensure your program meets all requirements for this lab.**

# 16.317: Microprocessor Systems Design I

Lab 3: Assembly Language Programming

You must include this sheet as the cover page of your lab report. Fill in your name and your partner's name (if applicable). The table below provides the grading rubric for this assignment, as well as space for an instructor to record your grade for each section.

**Student name: _____ Student ID # _____**

**Partner's name: _____**

**Grading rubric**

| **Item** | **Description** | **Max points** | **Actual points** |
|---|---|---|---|
| Prompt | Print out prompts on screen | 10 | |
| Keyboard inputs | Correctly read inputs from keyboard | 20 | |
| Conversion | Correctly convert ASCII characters to hexadecimal values | 20 | |
| Sorting | Sort numbers properly | 20 | |
| Output | Print sorted numbers on screen | 10 | |
| Flowchart | Include a detailed flowchart in your report that shows, at a minimum, the sorting algorithm | 10 | |
| Comments | Your source code must contain detailed comments | 10 | |
| *Bonus* | *Modify your program to handle 2-digit numbers* | *20* | |
| | **TOTAL** | 100 *(120 with bonus)* | |