Laboratory 4 Using MUN-88 Function Calls

Last Revised: June 2000

1 Objectives

With the completion of this lab, you should be able to:

- Use MUN-88 function calls to access the built-in system functions.
- Write a simple calculator program.

2 Introduction

In this lab, you will have to write a series of short programs that use the MUN-88 function calls to light LEDs, accept input from the PC keyboard, and print messages to the PC screen. You will finally tie the programs together to produce a very simple calculator program.

3 The Programs

For each of the following programs, use the appropriate MUN-88 function calls – for example, do **not** light the LEDs with the assembly instruction `OUT 30h, AL`. Comment your code, and include a header block at the beginning of your code with details such as your name, a title, the date, and a short description.

Program #1 – Keyboard input

Write a program that reads a line of text (up to 80 characters) from the PC keyboard, and store the characters in an array of bytes you define in your program.

After the MUN-88 function call ends, check to see whether the user pressed the RETURN key or the ESC key. If the RETURN key was pressed, light LED L1; if the ESC key was pressed, light LED L2.

Program #2 – String output

Write a program that takes a NULL-terminated string of characters and prints them to the PC screen. Your test string should be at least 10 characters long, and stored as an array in your program. **NOTE: Function call #4 uses BX, not SI.**

Program #3 – Number input

Write a program that prints the message **`Number:`**, and waits for the user to type in 2 characters. Then check whether or not the characters represent a valid hexadecimal number by using a function call. If valid, store the number in register DX. If not, repeatedly ask for a number until a valid value is given. However, if the user presses the ESC key, you should end your program immediately.

Program #4 – Number output

Write a program that takes the 16-bit value in BX, and converts it to four ASCII hexadecimal characters (stored in memory). Then print the message **`Answer:`**, followed by the four numbers.

Program #5 – Calculator

This program combines the previous two programs to create a simple calculator application.

Start up by printing a string **`Number 1:`**, and wait for the user to enter a 2-digit hexadecimal value. When the user presses return, check that the value is a valid number; if not, ask for **`Number 1:`** again. Do this until the user enters a valid number (and store it), or until the user presses the ESC key. In that case, your program should exit immediately.

Next, ask for **`Number 2:`**, with a similar procedure to check for a valid number, and for the ESC key.

Finally, add the two 8-bit numbers together. Print the message **`Sum:`** to the screen, followed by the **16-bit** representation of the answer. Repeat the entire program, until the user presses the ESC key.

BONUS MARK: Add a message that appears only at the beginning, giving the name of the program and brief description. Only display the message again if the user enters "h" or "H" at one of the "Number" prompts.

4 Submission

Print your list files for programs #1, #2, and #5. Please demonstrate each of these programs to a TA, who will sign your sheet. Only if you are unable to get program #5 to work should you print off and submit programs 3 and 4. Also submit your prelab.