CPEN 3710 **Lab Exercise 2**

Introduction to Assembly Language using MASM

Required Materials:

- Your textbook, *Assembly Language for Intel-Based Computers* (6th edition)
- Removable or network device (Flash drive, memory card, MyMocsNet account mapped to a drive letter, etc.) for storage of your programs
- These instructions
- Intel-compatible, Windows-based personal computer with MASM 6.11 and 6.15, DOSBox 0.74, and text editor (like the ones in EMCS 306)

Preparation for Laboratory:

Read sections 1.1 (pages 1-7), 1.3 (pages 9-19), 3.1 through 3.3 (pages 58-77), and 3.6 (pages 90-91) in your textbook. Also read over the instructions below.

Discussion:

The purpose of this lab is to become familiar with the basic steps of editing, assembling, linking, and running assembly language programs written for protected mode and real mode execution in the Windows environment. Any text editor that will produce an ASCII file (such as Notepad or WordPad that are included with Windows, or any of several freeware/shareware/commercial alternatives such as TextPad) can be used to edit an assembly language source program. The assembler (and linker) we will be using in the lab are part of the freely available Microsoft Assembler (MASM) distribution, version 6.15 (6.11 for real-mode programs). These programs are all pre-installed on the machines in the lab. MASM supports all the standard Intel assembly language operations and directives that are covered in our book; it accepts an assembly language source file(s) (with an .ASM extension) and creates machine language object file(s) (with an .OBJ extension) and then links these to form an executable program (.EXE extension). The executable program can be run from a command prompt or – as we shall see in a later exercise – tested by running it within CodeView (inside DOSBox, for real-mode programs) or the Visual Studio debugger (protected mode programs).

Instructions:

Find an available machine in the EMCS 306 lab and login (see me if you do not know the password). Double-click the CPEN3710 desktop icon and then open a command window by double-clicking the **cmd.exe** shortcut. Place your removable media in the appropriate drive or USB port, or establish a network drive connection as needed. Determine the drive letter associated with your removable/network storage device (hint – look under **My Computer**) and change to its command prompt by typing in its drive letter followed by a colon. For example, if it is the E drive, type in **E:**

Find the sample protected mode program on pages 66-67 of your textbook that adds two numbers and subtracts a third from the sum. Use a text editor to type in the program as shown in the text ***(add a comment line(s) with your name, the course number and section, and the date)***, then save this program onto your disk with an extension of .ASM (for example, ADDSUB.ASM). Make sure it is saved as a plain ASCII text file so the assembler will be able to process it.

The assembler and linker (MASM.EXE and LINK32.EXE) are installed on each machine in the lab in the C:\CPEN3710\MASM615 folder. The machines in the EMCS 306 lab should already have this

directory set in the search path when you log on, such that you can run the assembler and linker from any directory in which your source file resides. If you wanted to work at home, or if for any reason your lab machine did not already have this configuration, you could set the path manually using a command such as the following:

```
SET PATH=C:\CPEN3710\MASM615;%PATH%
```

Now, from the command prompt, execute the commands

```
SET LIB=C:\CPEN3710\MASM615\LIB
```

and

```
SET INCLUDE=C:\CPEN3710\MASM615\INCLUDE
```

in order to get your programs to assemble and link properly.

Once the LIB and INCLUDE environment variables are set correctly, you should be able to assemble your program using the following command (assuming your source file is named ADDSUB.ASM):

```
ml /c /coff /Zi /Fl /Sn addsub.asm
```

Note that the slashes in the command line are "forward" slashes (same key as the question mark but without shifting) and the capitalization after the third and fourth slashes is significant. The fourth option is an upper-case F followed by a lower-case L, not the numeral 1. The `/c` option assembles the program without linking it (yet). `/coff` selects the particular object file format that is needed for protected mode programs as opposed to real mode programs. `/Zi` tells the assembler to generate the symbolic debugging information needed by the Visual Studio debugger. `/Fl` causes the assembler to generate a listing file (which you will turn in for grading for each program you write this semester). `/Sn` causes the assembler to suppress the symbol table that is normally included in the listing file, saving paper when you print the program listing.

If you get any error messages, correct the errors in your source file and re-assemble the program. If you entered the source program with no errors, the assembly should succeed and you should obtain an .OBJ file and a .LST file with the same name as your source file. Next, it is time to link your program using the following command:

```
link32 addsub.obj irvine32.lib kernel32.lib /SUBSYSTEM:CONSOLE /DEBUG
```

If you had a "good" object file from the previous step and if you typed in the above command properly, you should now have an executable file (ADDSUB.EXE). The executable file can be run by typing in its name, **ADDSUB** (it is ok, but not necessary, to type the .EXE extension) at the command prompt. Do this and note the results, if any, that are produced. Use **alt-PrintScreen** to capture a screen shot and paste it into a word processor document so you can print it out to document your results.

Note for future reference that the author of our textbook has created a batch file MAKE32.BAT that will set the environment variables and execute the above two commands on a given (protected mode format) .ASM file to produce the .OBJ, .LST, and .EXE files. This file is present in the

C:\CPEN3710\MASM615 directory on your machine. From now on, you can conveniently "make" (assemble and link) your program by simply entering the command **`make32 filename`** to carry out both of the above steps in sequence.

Next, let's go through the process of creating a real mode (MS-DOS compatible) program. Use your favorite text editor to create another ASCII text source file, ADDSUB2.ASM, containing the program statements on page 91 of your text. **Again, for identification purposes add a comment line(s) including your name, the course number and section, and the date.** Copy your .ASM file into the C:\CPEN3710 directory or mount the directory where it resides as a drive in DOSBox. **Run DOSBox; from the DOSBox command screen**, assemble and link the program using the following command:

```
ml /Fl /Zi /Sn addsub2.asm irvine16.lib
```

(Note that, as above, there is a MAKE16.BAT file that is available to simplify the process of assembling and linking real-mode programs. It is located in C:\CPEN3710\MASM611\BIN, which is mapped as A:\MASM611\BIN in DOSBox.)

Again verify that you received no error messages and that the ADDSUB2.LST, ADDSUB2.OBJ, and ADDSUB2.EXE files were produced. Run the executable file by typing **ADDSUB2** at the command prompt. Note the results, if any, that are produced and use **alt-PrintScreen** and a word processor as before to capture and print a screen shot to document your results.

Finally, modify the ADDSUB2 program such that all three numbers are added together. Call the new version of the program ADDTHREE. **This time, the numbers should be stored as words (rather than as doublewords) and their sum should be calculated in a 16-bit register. Initialize the three values to 12EF, ABCD, and 09A6 hexadecimal and store their 16-bit sum in finalVal, then dump the registers as before.** Assemble, link, and run the program and capture and print a screen shot of the output produced.

To Hand In: *(due no later than 3:00 p.m. Tuesday, September 9)*

Turn in a **printed copy of all three .LST files** (not the .ASM files) and the **three screen shots showing the program results**. (To help me in grading, put them in the order of listing 1, output 1, listing 2, output 2, listing 3, output 3.) Also, **use a word processor to type up and print out answers to the following questions, and include them at the end of your report:**

1. **Explain in your own words** the differences between the first and second programs (ADDSUB and ADDSUB2). Consider **both** the differences in structure between protected and real mode code and the differences in how the task itself was accomplished.

2. **Repeat question 1** for the third program (ADDTHREE) with respect to the second (ADDSUB2). This time, since both programs are written for real mode execution, the differences you are explaining should only be in terms of the data definition directives and the code itself.