16.317: Microprocessor Systems Design I

Spring 2013

Lab 1: DEBUG intro; assembling and executing instructions

Due **Wednesday, 3/6/13**

Introduction and Objectives

In this lab, you will learn how to:

1. Use the DEBUG software: Run/quit DEBUG software, show/modify register content, show/modify flags, dump memory contents, assemble and debug programs.
2. Assemble instructions into the memory of the PC.
3. Execute an instruction to determine the operation it performs.
4. Verify the operation of data transfer, arithmetic, compare, and jump instructions.

Remember that, although you may work in groups of two, each student is responsible for submitting his or her own lab report. Please read the lab report guidelines listed on the web page, and be sure to include the last page of this document as your lab report cover sheet.

1. Basic DEBUG commands

You will learn how to run DEBUG, a program that will allow you to assemble programs and execute them line by line in later labs. You will gain some basic familiarity with the DEBUG commands in this part of the lab, which will require you to complete some basic examples from Chapter 4 of the textbook.

To practice DEBUG commands, you will reproduce examples from the textbook. **To get full credit for this section, submit a screen shot showing the successful completion of each example or group of examples.** Specifically, you need to practice the following command categories (see the appendix for details of each example, if you do not have the textbook):

Register and flag commands
Example 4.9, 4.10, 4.11 (page 109-111)

Memory
Example 4.14, 4.15, 4.16, 4.17, 4.18, 4.19 (page 112-121)

I/O
Example 4.20 (page 122)

Hexadecimal
Example 4.22 (page 124)

A full list of DOS DEBUG commands is included as part of the appendix, which starts on page 8. For more detailed references, see p. 106 of the textbook or visit:

- http://thestarman.pcministry.com/asm/debug/debug.htm
- http://kb.iu.edu/data/afhs.html

2. Loading and running programs in DEBUG

To practice DEBUG commands used to load programs and initialize registers/memory, complete Example 4.26 from p. 136 of the textbook, which is described below. **To get full credit for this section, submit screen shots showing the completion of Steps 1 and 2, and answer all questions provided.**

Step 1: Complete Example 4.24 (p. 127), which enters a machine code program into memory and saves it on disk.

a. Enter the following all on one line to load the machine code for the program:
`E CS:100 B8 00 20 8E D8 BE 0 01 BF 20 01 B9 10 0 8A 24 88 25 46 47 49 75 F7 90`

b. Verify that the code has been correctly loaded using the command: `D CS:100 117`

c. Disassemble the code to see the actual instructions using: `U CS:100 117`

d. Name this file BLK.1, specifying the full directory path where it is to be stored; for example: `N C:\MGEIGER\BLK.1`

- **IMPORTANT NOTE:** Choose your file location wisely, as writing the file into the wrong location could unintentionally damage other files.

e. To ensure that the file is written correctly, the register pair BX/CX must hold the total number of bytes. Since this program runs from CS:100 to CS:117, it holds 18_{16} bytes. Therefore, initialize CX to 18 and BX to 0 using the following commands:

- `R CX` → Enter 18 after the current value of CX is displayed
- `R BX` → Enter 0 after the current value of BX is displayed

f. Write the file to the specified location: `W CS:100`

g. Exit the DEBUG program; then, navigate to the directory holding the file BLK.1 and rename it BLK.EXE using the DOS REN command.

Step 2: Complete example 4.26, which loads the executable program and executes new commands as follows:

a. Start DEBUG and load the executable into address CS:100:

- N <path>\BLK.EXE
- L CS:100

b. Set DS to the value 2000_{16}, memory from DS:100 through DS:10F as FF_{16}, and memory from DS:120 through DS:12F as 00_{16}. Verify memory was loaded correctly.

c. Reload DS with the value 1342_{16}. Display the state of all registers to verify the result of this operation.

d. Display the assembly language version of the program from CS:100 through CS:117.

e. Use a GO (G) command to execute instructions through address CS:10E. What changes have occurred?

f. Execute through address CS:115. ***What changes are found in the blocks of data?*** Now execute through CS:117. ***What changes are found in the blocks of data?***

3. Testing instructions in DEBUG

In order to successfully be checked off for this assignment, generate at least one screen capture per instruction category. Each category contains several questions to be answered as well; each student must answer these questions in his or her lab report.

Section 3.1: Data transfer I

1. Use DEBUG to assemble the following instructions:
   a. `MOV AX, BX`
   b. `MOV AX, AAAA`
   c. `MOV AX, [BX]`
   d. `MOV AX, [4]`
   e. `MOV AX, [BX+SI]`
   f. `MOV AX, [SI+4]`
   g. `MOV AX, [BX+SI+4]`

2. Initialize the internal registers of the 80386 as follows:
   - `(AX) = 0000H`
   - `(BX) = 0001H`
   - `(CX) = 0002H`
   - `(DX) = 0003H`
   - `(SI) = 0010H`
   - `(DI) = 0020H`
   - `(BP) = 0030H`

   Verify these registers are correctly initialized by displaying the new register contents.

3. Fill all memory locations in the range DS:00 through DS:1F with 00H and initialize the word storage locations listed below:
   - `(DS:0001H) = BBBBH`
   - `(DS:0004H) = CCCCH`
   - `(DS:0011H) = DDDDH`
   - `(DS:0014H) = EEEEH`
   - `(DS:0016H) = FFFFH`

4. Trace the execution of the instructions you assembled in step 1.

***Question 1.1:* Explain the execution of each instruction (a-g), including the addressing mode used, physical address accessed, and the value stored in AX.**

Section 3.2: Data transfer II

1. Assemble the instruction MOV SI, [0ABC] to memory at address CS:100 and verify loading of the instruction.

Question 2.1: ***How many bytes does this instruction take up?***

2. Initialize the word of memory starting at DS:0ABC with the value FFFFH.
3. Clear the SI register and verify that operation by displaying its content.
4. Trace the execution of this instruction.

Question 2.2: ***Describe the operation of this MOV instruction.***

5. Assemble the instruction MOV WORD PTR [SI], ABCD into memory at address CS:100 and then verify loading of the instruction.

Question 2.3: ***How many bytes does this instruction take up?***

6. Initialize the SI register with the value 0ABCH.
7. Clear the word of memory starting at DS:0ABC.
8. Trace the execution of this instruction.

Question 2.4: ***Describe the operation of this MOV instruction.***

Section 3.3: Arithmetic instructions

1. Assemble the instruction ADC AX, [0ABC] to memory at address CS:100 and verify loading of the instruction.

Question 3.1: ***How many bytes does the instruction take up?***

2. Initialize the word of memory starting at DS:0ABC with value FFFFH.
3. Initialize AX with the value 0001H. Verify by displaying register contents.
4. Clear the carry flag.
5. Trace the execution of this instruction.

Question 3.2: ***Describe the operation performed by this ADC instruction.***

Question 3.3: ***Does this instruction produce a carry out from the last bit?***

Section 3.4: Flag control instructions

1. Assemble the following instruction sequence:
    a. `LAHF`
    b. `MOV  BH, A`
    c. `AND  BH, 1F`
    d. `AND  AH, E0`
    e. `MOV  [200], BH`
    f. `SAHF`

Question 4.1: ***How many bytes do these instructions take?***

2. Initialize the byte of memory starting at DS:200 with the value 00H.

3. Clear register AX and BX.

4. Display the current state of the flags and ensure that the flags equal NG, ZR, AC, PE, and CY.

5. Trace the execution of this sequence.

Question 4.2: ***Describe the operation of each instruction. In particular, note what value is initially read out of the flags register, what value is saved in memory, and what value is reloaded into the flags register.***

Section 3.5: Compare instructions

1. Assemble the following instruction sequence into memory at address CS:100 and then verify loading of the instructions:
    a. `MOV  BX, 1111`
    b. `OR   AX, BBBB`
    c. `CMP  BX, AX`

2. Clear register AX and BX.

3. Display the current state of the flags.

4. Trace the execution of this sequence.

Question 5.1: ***What operation does each instruction perform?***

Question 5.2: ***What are the status flags before and after the compare instruction is executed?***

Section 3.6: Jump instructions

1. Download the files L5P3.LST and L5P3.EXE from the course web page into a folder.
2. Open the listing file (L5P3.LST).

***Question 6.1:** What are the offsets from CS for the first instruction (PUSH DS) and the last instruction (RET)?*

3. Use the DEBUG program to load the run module L5P3.EXE.
4. Verify loading of the program by disassembling the contents of the current code segment for the offset range found in step 2. (Use the U ("Unassemble") command.)
5. Execute the program according to the instructions that follow:
    a. GO (G) from address CS:00 to CS:5.
    b. Load the number for which the factorial is to be calculated (N=3) into DX.
    c. Clear the memory location DS:0000 for the value of the factorial (FACT).
    d. GO from address CS:5 to CS:10
    e. Execute the JZ instruction using a TRACE (T) command.
    f. GO from address CS:12 to CS:16.
    g. Execute the JMP instruction with a TRACE command.
    h. GO from address CS:E to CS:10.
    i. Execute the JZ instruction with a TRACE command.
    j. GO from address CS:12 to CS:16.
    k. Execute the JMP instruction with a TRACE command.
    l. GO from address CS:E to CS:10.
    m. Execute the JZ instruction with a TRACE command.
    n. GO from address CS:12 to CS:16.
    o. Execute the JMP instruction with a TRACE command.
    p. GO from address CS:E to CS:10.
    q. Execute the JZ instruction with a TRACE command.
    r. GO to CS:1B.
    s. Display the value stored for FACT in memory.

***Question 6.3:** For each jump instruction in the sequence, describe whether or not the jump was taken and, if taken, what the address of the next instruction to be executed is.*

***Question 6.4:** List all values in AL and the instructions that change that register. At what address is the final value in AL stored in memory as FACT?*

Resources

This assignment requires you to work in the DOS environment; for a list of useful DOS commands, see: http://www.computerhope.com/dostop10.htm

If you are completing this lab on a Windows 7 machine, you cannot use the DEBUG prompt from the Windows command window—you'll need a DOS emulator and a DEBUG clone.

We recommend using the emulator DOSbox, which you can download from http://www.dosbox.com

The DEBUG clone we've successfully used in the past can be found at: http://www.softpedia.com/get/Programming/Debuggers-Decompilers-Dissasemblers/DOS-Debug.shtml

We recommend setting up a single directory for all of your 16.317 lab materials, so you can easily mount that directory as a drive within DOSbox. Once you start DOSbox, use the "mount" command to map your directory to a drive letter. For example, if you want to treat the directory "C:\Users\Michael_Geiger\MicrosI" as the C: drive within DOSbox, type the following at the DOSbox prompt:

```
mount c c:\Users\Michael_Geiger\MicrosI
```

If you then simply type the drive name (C:), it will take you to that directory, and you can run any programs or access any subdirectories contained within.

Appendix

DOS DEBUG commands:

Command	Function
?	Displays a list of debug commands
A	Assembles 8086/8087/8088 mnemonics
C	Compares two portions of memory
D	Displays the contents of a portion of memory
E	Enters data into memory starting at a specified address
F	Fills a range of memory with specified values
G	Runs the executable file that is in memory
H	Performs hexadecimal arithmetic
I (capital i)	Displays one byte value from a specified port
L	Loads the contents of a file or disk sectors into memory
M	Copies the contents of a block of memory
N	Specifies a file for an L or W command, or specifies the parameters for the file you are testing
O	Sends a single byte value to an output port
P	Executes a loop, a repeated string instruction, a software interrupt, or a subroutine
Q	Stops the DEBUG session
R	Displays or alters the contents of one or more registers
S	Searches a portion of memory for a specified pattern of one or more byte values
T	Executes one instruction and then displays the contents of all registers, the status of all flags, and the decoded form of the instruction that Debug will execute next.
U	Disassembles bytes and displays the corresponding source statements
W	Writes the file being tested to a disk
XA	Allocates expanded memory
XD	Deallocates expanded memory
XM	Maps expanded memory pages
XS	Displays the status of expanded memory

Appendix

Examples from Section 1: DEBUG intro

Example 4.9 (p. 109) Verify the initialized state of the 80386 DX by examining the contents of its registers with the register command, "R".

Example 4.10 (p. 109) Issue commands to the debugger on the PC/AT that will cause the value in BX to be modified to $FF00_{16}$, and then verify that this new value is loaded into BX.

Example 4.11 (p. 111) Use the register command to set the parity flag to even parity. Verify that the flag has been changed.

Example 4.14 (p. 114) Issue a dump command that will display the contents of the 32 bytes of memory that are located at offsets 0300_{16} through $031F_{16}$ in the current data segment.

Example 4.15 (p. 115) Use the dump command to examine the 16 bytes of memory just below the top of the stack.

Example 4.16 (p. 117) Start a data entry sequence by examining the contents of address DS:100 and then, without entering new data, depress the – key. What happens?

Example 4.17 (p. 118) Initialize all storage locations in the block of memory from DS:120 through DS:13F with the value 33_{16} and the block of storage locations from DS:140 through DS:15F with the value 44_{16}. Verify that the contents of these ranges of memory are correctly modified.

Example 4.18 (p. 119) Fill each storage location in the block of memory from address DS:100 through DS:11F with the value 11_{16}. Then, copy this block of data to a destination block starting at DS:160. Verify that the block move is correctly done.

Example 4.19 (p. 121) Perform a search of the block of data from addresses DS:100 through DS:17F to determine which memory locations contain 33_{16}.

Example 4.20 (p. 122) Write a command that will display the byte contents of the input port at I/O address $00FE_{16}$.

Example 4.22 (p. 124) If a byte of data is located at physical address $02A34_{16}$ and the data segment register contains 0150_{16}, what value must be loaded into the source index register such that DS:SI points to the byte storage location?

16.317: Microprocessor Systems Design I

Spring 2013

Lab 1: DEBUG intro; assembling and executing instructions

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
Formatting	Your report contains all required sections and is of sufficient length	10	
Section 1: Basic DEBUG commands	Submit screen shots or printouts from each of class of commands. (register/flag, memory, I/O, hex)	20	
Section 2: Loading and running programs in DEBUG	Submit screen shots or printouts from each step and answer the included questions.	10	
Section 3.1: Data transfer I	Submit a screen shot or printout of the section results. Answer question 1.1 correctly.	10	
Section 3.2: Data transfer II	Submit a screen shot or printout of the section results. Answer questions 2.1-2.4 correctly.	10	
Section 3.3: Arithmetic	Submit a screen shot or printout of the section results. Answer questions 3.1-3.3 correctly.	10	
Section 3.4: Flag control	Submit a screen shot or printout of the section results. Answer questions 4.1-4.2 correctly.	10	
Section 3.5: Compare	Submit a screen shot or printout of the section results. Answer questions 5.1-5.2 correctly.	10	
Section 3.6 Jump	Submit a screen shot or printout of the section results. Answer questions 6.1-6.4 correctly.	10	
	TOTAL	100	