LABORATORY 6:

EXPLORING THE MEMORY AND DISPLAY SUBSYSTEM OF THE PC

NAME: STUDENT ID#:  

Objectives

Learn how to:
- Explore the implemented and unimplemented parts of the memory space in the PC.
- Explore the R/W memory and ROM parts of the implemented memory.
- Determine the ROM BIOS release data.
- Execute a routine to test a block of storage locations in memory.
- Write a memory test program.
- Explore the relationship between the display memory buffer contents and corresponding characters on the display.
- Execute a program to display the ASCII character set of the PC one character at a time on the screen.
- Write a program that displays the complete ASCII character set on the screen.

Part 1: IBM PC Memory

The microprocessor in the PC is capable of accessing 1M-byte of memory. The address range 016 through BFFFF16 is meant for R/W memory and CC00016 to FFFFF16 is for the ROM. Not all the ROM or R/W memory address space is implemented in a particular PC. Here we will explore various address ranges to determine if they are implemented and, if a range is implemented, whether it is R/W memory or ROM. Check off each step as it is completed.

<table>
<thead>
<tr>
<th>Check</th>
<th>Step</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.</td>
<td>Load the DEBUG program by entering C:&gt;DEBUG   (~)</td>
</tr>
<tr>
<td></td>
<td>2.</td>
<td>Perform the debug operations that follow:</td>
</tr>
</tbody>
</table>

【LAB06-1】
a. Dump the 128 memory locations starting at address 100:0.
b. Dump the 128 memory locations starting at address 4000:0.
c. Dump the 128 memory locations starting at address B800:0.
d. Dump the 128 memory locations starting at address F600:0.

e. Comparing the displays produced for steps a, b, c, and d, can you identify an area of ROM?

3. Try writing 55H to the 128 locations starting at 100:0, 4000:0, B800:0, and F600:0. To verify writing, dump the contents of these locations. Are there locations that you were not able to write to? ________________________________
   If yes, what are these locations? ____________________________________________

4. Dump the 8-byte memory contents starting at address F000:FFF5. What do you see?

5. Quit the DEBUG program.

Part 2: Executing a Memory Test Program

L06P2.ASM is the source program that writes the pattern 0755H to the 128-byte memory block starting at location B800:0. You should verify that usable RAM is available in this memory range of your PC. After writing the pattern, the storage locations are read and their contents compared with the pattern. This program does not verify that each bit in a word can be set to 0 or 1. It simply determines whether or not a specific bit pattern can be written into or read from each memory location in the block. The run module is available as file L06P2.EXE. Now we will execute this program and verify if the memory passes the read/write test.

<table>
<thead>
<tr>
<th>Check</th>
<th>Step</th>
<th>Procedure</th>
</tr>
</thead>
</table>
| _____ | 1.   | Clear the screen by issuing the command  
|       |      | C:\>CLS (↓)  
|       |      | **NOTE:** This operation must be performed; otherwise the steps that follow will not provide the desired results.  
| _____ | 2.   | Load the run module L06P2.EXE with the debugger.  

【LAB06-2】
3. Run the program by executing it to the end. What happens to the top line of the display?

Why?

4. DUMP the register contents of DX. What is the significance of the value in DX with respect to the state of the memory block?

5. DUMP the memory contents starting at address B800:0. What do they signify?

6. Quit the DEBUG program.

Part 3: Modifying the Memory Test Program

Earlier we pointed out that the program run in Part 2 verifies that a particular pattern can be written into and read from each word of the block of memory. It does not verify that every bit position in the segment of memory can be set to 1, reset to 0, and read back correctly. Here we will modify this program to test the block of memory using bit patterns that allow better bit testing.

Check Step Procedure

1. Modify the memory test program of L06P2.ASM as follows:
   a. Address range of the block of memory locations to be from 10000H through 100BFH.
   b. The storage locations in memory are to be tested as bytes instead of words.
   c. The bytes of memory are to be tested with each of these patterns 00H, FFH, AAH, and 55H.
   d. The patterns are to be read from a table in memory starting at address PATTERN.
   e. If the test passes, the contents of registers AX, BX, CX, and DX should all be zero. However, if the test fails for a byte, the program must stop immediately and update the values in the registers as follows:
(AX) = Current contents of DS
(BX) = Offset of the memory location that failed
(CX) = The data that failed
(DX) = 0BADH

2. Create the source program with an editor, assemble into an object program, and create a run module in file MTEST.EXE.

3. Verify the operation of the program for good memory by running it on the PC.

4. Use a debug sequence to verify the operation of the program for a bad memory by executing the program up to the compare instruction; before performing the compare operation, modify the data read from memory with an E command so that it simulates an error; then run the program to completion; finally, examine the registers to determine if the error condition is correctly identified. Repeat this process for each of the patterns.

Part 4: Display Memory Buffer

A part of the PC’s memory known as the color display buffer resides from address B800:0 through B800:0FFF in memory. Figures L6.1(a) and (b) show the relationship between the lines and columns of the screen and the contents of the display buffer memory. Here we will study the relationship between the contents of this buffer and information displayed on the screen. For every character displayed on the screen, there are 2 consecutive bytes of information in display memory. The byte marked ASCII, which is located at the even address, is the code for the character. The table in Figure L6.2 shows the ASCII codes for all numbers, characters, and symbols that can be displayed on the screen of the original IBM PC. The contents of the ATTR byte, which is at the odd address, selects display attributes for the character. The attributes available for characters displayed on the monochrome display are normal, blinking, underlined, intensified, and reverse-video. The chart in figure L6.3 shows the relationship between the bits of the attributes byte and display features. By performing the steps that follow, we will examine the relationship between the contents of the character and attribute bytes and displayed information. Check off each step as it is completed.

NOTE: If the PC you are using has a monochrome display, instead of a color display, the display buffer starts at address B000:0. Therefore, to run the laboratory exercises that follow with a monochrome monitor, simply replace addresses B800:0 through
B800:0FFF with the corresponding address in the range B000:0 through B000:0FFF. The results obtained will be similar.

Figure L6.1 (a) Organization of the screen as rows and columns. (b) Storage of character information in the display buffer.

Figure L6.2 ASCII codes and corresponding display character.
Check Step Procedure

1. Clear the screen by issuing the command
   
   `C:\>CLS` 
   
   **Note:** This operation must be performed; otherwise, the steps that follow will not provide the desired results.

2. Load DEBUG by entering
   
   `C:\>DEBUG` 
   
   Write down what is displayed on the screen.

3. Dump the contents of the display buffer from address B800:0 through B800:13F.
   
   **Note:** If you make any error in the key sequence, you must start over by returning to DOS and clearing the screen.

At what addresses in the buffer are the characters of the word DEBUG held? Also list the ASCII code and the attribute byte for each character.

<table>
<thead>
<tr>
<th>Character</th>
<th>Address</th>
<th>ASCII Code</th>
<th>Attribute Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>_______</td>
<td>__________</td>
<td>______________</td>
</tr>
<tr>
<td>E</td>
<td>_______</td>
<td>__________</td>
<td>______________</td>
</tr>
<tr>
<td>B</td>
<td>_______</td>
<td>__________</td>
<td>______________</td>
</tr>
<tr>
<td>U</td>
<td>_______</td>
<td>__________</td>
<td>______________</td>
</tr>
<tr>
<td>G</td>
<td>_______</td>
<td>__________</td>
<td>______________</td>
</tr>
</tbody>
</table>

Why are the contents of the attributes bytes all displayed as a period at the...
right of the screen?

____________________________________________________________

When no character is displayed in a position on the screen, what ASCII code is put into the display buffer address corresponding to this location?

____________________________________________________________

What does this ASCII code stand for? ____________

4. Return to DOS, clear the screen, and reenter DEBUG.

5. Use a DUMP command to display the contents of the display buffer from address B800:A0 through B800:BF. How does this compare to the results observed in step 3? ____________________________________________

6. Display the contents of the display buffer for the address range B800:140 through B800:15F. What line of information on the screen does this represent?

____________________________________________________________

7. Dump the contents of the display buffer for the address range B800:1E0 through B800:27F. What line of information does it represent on the screen?

____________________________________________________________

8. Enter DEBUG and then display the contents of the display buffer from address B800:00 through B800:27F. Compare the displayed information to the contents of the buffer. What is the address range in the buffer of the line of the screen where the DEBUG command is displayed?

____________________________________________________________

Find the address range in the buffer of the line of the screen where the DUMP command is displayed.

____________________________________________________________

At what address range in the display buffer is the first line of information displayed with the DUMP command held?

____________________________________________________________

9. Perform the DEBUG operations that follow.

a. Enter the ASCII characters of the word NORMAL separated and
terminated by the hexadecimal code 07 starting at address B800:0. That is, enter ‘N’07’O’07’R’07’M’07’A’07’L’07 into address B800:0 through B800:B. What is displayed at the top left corner of the screen?

b. Enter the ASCII characters of the word BLINKING separated and terminated by hexadecimal code 87 starting at address B800:0. What is displayed at the top left corner of the screen?

c. Enter the ASCII characters of the word BLUE separated and terminated by hexadecimal code 01 starting at address B800:0. What is displayed at the top left corner of the screen?

d. Enter the ASCII characters of the word INTENSIFIED separated and terminated by hexadecimal code 0F starting at address B800:0. What is displayed at the top left corner of the screen?

e. Enter the ASCII characters of the word REVERSEVIDEO separated and terminated by hexadecimal code 70 starting at address B800:0. What is displayed at the top left corner of the screen?

f. What are the addresses of the memory locations that contain the ASCII code and attributes of a character displayed on line 20 at column 50?

12. Quit the DEBUG program.
Part 5: Executing a Program that Displays the Characters of the ASCII Character Set on the Screen

In this part of the laboratory exercise, we will examine the ASCII display character set of the PC. Here we will assemble, save, execute, and analyze a program that when run displays the characters of the ASCII character set on the screen.

<table>
<thead>
<tr>
<th>Check</th>
<th>Step</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>_____</td>
<td>1.</td>
<td>Bring up the DEBUG program.</td>
</tr>
<tr>
<td>_____</td>
<td>2.</td>
<td>Assemble the program that follows into memory starting at address CS:100.</td>
</tr>
</tbody>
</table>

```assembly
CS:100H    MOV  AX, B800 ;  _______________________
CS:103H    MOV   DS, AX   ;  _______________________
CS:105H    MOV   DI, 0C80  ;  _______________________
CS:108H    MOV   AX, 0721  ;  _______________________
CS:10BH    MOV   DL, FF    ;  _______________________
CS:10DH    MOV   [DI], AX  ;  _______________________
CS:10FH    MOV   CX, FFFF ;  _______________________
CS:112H    DEC    CX    ;  _______________________
CS:113H    JNZ    0112    ;  _______________________
CS:115H    INC    DI    ;  _______________________
CS:116H    DEC    DL    ;  _______________________
CS:118H    JZ    011C    ;  _______________________
CS:11AH    JMP    010D    ;  _______________________
CS:11CH    NOP         ;  _______________________
```

Disassemble the program to verify that it has loaded correctly. Add comments to the program to explain what each instruction does. What is the ending address of the program? ______________

How many bytes of memory does the program take up? ______________

How many times does the loop implemented with the JNZ instruction get repeated? ______________

What is the purpose of this loop?

How many times does the loop performed by the JZ instruction get repeated? ______________

Describe the operation performed by this loop.
3. Save the program in file CHAR.1.
4. Quit the debugger, clear the screen with a CLS command, and then bring DEBUG back up.
5. Load the program from CHAR.1 at CS:100 and verify correct loading.
6. Run the program with a single GO command. Write the command so that execution stops at the NOP instruction.
7. Describe the events observed on the screen.

8. Describe how the program performs this function in detail.

---

**Part 6: Displaying the Complete ASCII Character Set on the Screen**

In the last part of the laboratory, we executed an existing program. Now we will modify that program to perform a different display operation. The modified program will be debugged, executed, and then saved into a file.

<table>
<thead>
<tr>
<th>Check</th>
<th>Step</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.</td>
<td>Bring up the DEBUG program, load the file CHAR.1; view the program with an unassembled command.</td>
</tr>
<tr>
<td></td>
<td>2.</td>
<td>Modify the program such that each new ASCII character is always loaded into the display buffer at the address corresponding to the next character position on the display. In the way, the complete character set will end up displayed on the screen.</td>
</tr>
<tr>
<td></td>
<td>3.</td>
<td>Assemble the modified program into memory at CS:100 and then save it on a diskette in file CHAR.2.</td>
</tr>
<tr>
<td></td>
<td>4.</td>
<td>Quit the debugger, clear the screen, and then bring DEBUG back up.</td>
</tr>
<tr>
<td></td>
<td>5.</td>
<td>Load the program from CHAR.2 at CS:100.</td>
</tr>
</tbody>
</table>
|      | 6.   | Run the program with single GO command. Does the program perform the desired operation? ________________

If not, debug the program and then repeat steps 2 through 6.