Solutions to End of Chapter 5 Problems

1. Consider the picture below, where all memory contents are in hexadecimal:

   ![Memory Contents Diagram]

   a) In words: what is held in the eip register, i.e., what is the purpose of this register? (Your answer should not be: "804838d").
      
      The address of the next instruction to be executed

   b) What would be displayed on the monitor by the command: `i r eip`?
      
      `eip 0x804838d 0x804838d <func+offset>`

   c) What would be displayed on the monitor by the command: `x/xb $eip`?
      
      `0x9b`

   d) What would be displayed on the monitor by the command: `i r esp`?
      
      `esp 0xbffff810 0xbffff810`

   e) What would be displayed on the monitor by the command: `x/xw $esp`?
      
      `0xbffff810: 0x8048475`

   f) What would be displayed on the monitor by the command: `x/xw 0x08048475`?
      
      `0x08048475: 0x59`

   g) What would be displayed on the monitor by the command: `x/xs 0x08048475`?
      
      `0x08048475: Yes`

2. The program `firstprog.c` is shown below:

   ```c
   #include <stdio.h>
   int main()
   {
     int i;
     for( i = 0; i < 10; i++ )
     {
       printf("Hello World!\n");
     }
   }
   ```

   Compile your program using `gcc`:

   ```bash
   gcc firstprog.c -o firstprog
   ```
gcc -g -o firstprog.exe firstprog.c
and then run your program
./firstprog.exe
to confirm it executes as expected.

Then start the debugger by entering the following commands (hitting ENTER after each command)
gdb -q ./firstprog.exe
list
disassemble main
break main
run

(a) The program has now stopped at the first line of code after the line int main( ). Recall that the eip register holds the address of the next instruction that will be executed. What is the address stored in the eip register? 0x8048384

(b) What is the next assembly language instruction that will be executed?
    mov DWORD PTR [ebp-4], 0x0

(c) Consider the assembly language instruction
    mov DWORD PTR [ebp-4], 0x0

    This instruction places the value 0 into the memory location whose address is stored at ebp-4 and stores it in 4 bytes.

    Enter nexti to execute this instruction

    What is the value of ebp? 0xbffff818

(d) What is the value of ebp-4? 0xbffff814

(e) What is stored in the address specified by the value ebp-4? Hint: Use the x/xw with your answer to question (d).
    0xbffff814: 0x00000000

(f) Look at the value of the instruction pointer (eip). Has it changed from your answer to part (a)? Why? / Why not?
    eip = 0x804838b. It has changed because of the nexti in step c.

(g) The next assembly language instruction that will be executed is:
    cmp DWORD PTR [ebp-4], 0x9

    This instruction will compare the value of 9 to the value you examined in question (e). Referring back to the C source code, what do you think this assembly language instruction is doing?

    It is the for loop exit condition

    Enter nexti to execute this instruction

(h) The assembly language instruction that will be executed next is
This instruction means:

If the result of the preceding comparison showed that the value stored at the memory location whose address is stored at ebp-4 is less than or equal to 9, jump to address 0x8048393.

Enter nexti once. What is the value of the eip register? 0x8048393

(i) Explain, in words, why the instruction pointer has the value that it has. The CMP instruction compared 0 to 9. The following JLE instruction jumped to 0x8048393 because 0 is less than 9.

(j) The assembly language instructions

```
mov DWORD PTR [esp],0x8048484
```

moves the value 8048484 into the location pointed to by the stack pointer.

Enter nexti once. What is the address stored in the stack pointer (esp)? 0xbffff810

(k) What is stored at the memory location whose address is in the stack pointer? (Hint: use x/xw to examine the value stored at the address specified by the stack pointer. 0x08048484

(l) We would like to know the significance of the address 0x8048484. What is stored at this location? (Hint: Examine the first four bytes stored starting at this memory location...think ASCII...could this be a string?)

```
(gdb) x/4x 0x8048484
0x8048484: 0x48 0x65 0x6c 0x6c
(gdb) x/4c 0x8048484
0x8048484: 72 'H' 101 'e' 108 'l' 108 'l'
(gdb) x/s 0x8048484
0x8048484: "Hello World!\n"
```