Solutions to Practice Problems

Practice Problem 5.1

If the first byte of a variable is stored at memory location numbered:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0

what is this address in hexadecimal notation?

Solution: 0000 0000 0001 0010 1111 1111 0111 1100

0 0 1 2 F F 7 C

0012FF7C

Practice Problem 5.2

For our x86 architecture, how many hexadecimal digits are in an address?

Solution: 8 hex digits (which equates to 32 bits)

Practice Problem 5.3

Consider the program shown below, along with its corresponding output.

```c
#include<stdio.h>
int main()
{
    int a = 4;
    int *a_ptr;
    a_ptr = &a;

    printf("\nThe value of a is %d and the address is %x \n" , a , &a );

    printf("\nThe value of a_ptr is %x and the address is %x \n\n", a_ptr , &a_ptr );
}
```

midshipman@EE488-VM:~/booksrc $ ./a.out

The value of a is 4 and the address is bffff854
The value of a_ptr is bffff854 and the address is bffff850

In the picture shown below:
(a) Fill in the two red circles.
(b) Draw an arrow showing where a_ptr is stored on the stack.
(c) Annotate the figure to show the value of a_ptr.
Recall that in RAM you have stored the machine language code for your program as well as additional memory allocated for your variables within the program. This latter additional memory is called the stack.

You type into the debugger the command

```
i r ebp
```

and get the result \(0xbffff818\). The register \(ebp\) points to the "bottom" of the stack.

Upon further review of the assembly code you determine that two strings are stored in memory, one at address \(ebp-40\) and the other at \(ebp-24\). (Note that the numbers 40 and 24 are ordinary base 10 numbers, not base-16.)

What are the two hidden words?

Solution: \(ebp-40\): Good and \(ebp-24\): Time
Practice Problem 5.5

A large program contains the following lines of code

```
int a = 11;
int b[2];
b[0] = 10;
b[1] = 6;
```

A section of this program's stack is shown below.

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xBFFFF8F0</td>
<td>0x3E</td>
</tr>
<tr>
<td>0xBFFFF8F1</td>
<td>0x3F</td>
</tr>
<tr>
<td>0xBFFFF8F2</td>
<td>0x4A</td>
</tr>
<tr>
<td>0xBFFFF8F3</td>
<td>0x0A</td>
</tr>
<tr>
<td>0xBFFFF8F4</td>
<td>0x00</td>
</tr>
<tr>
<td>0xBFFFF8F5</td>
<td>0x00</td>
</tr>
<tr>
<td>0xBFFFF8F6</td>
<td>0x00</td>
</tr>
<tr>
<td>0xBFFFF8F7</td>
<td>0x06</td>
</tr>
<tr>
<td>0xBFFFF8F8</td>
<td>0x00</td>
</tr>
<tr>
<td>0xBFFFF8F9</td>
<td>0x00</td>
</tr>
<tr>
<td>0xBFFFF8FA</td>
<td>0x00</td>
</tr>
<tr>
<td>0xBFFFF8FB</td>
<td>0x0B</td>
</tr>
<tr>
<td>0xBFFFF8FC</td>
<td>0x00</td>
</tr>
<tr>
<td>0xBFFFF8FD</td>
<td>0x00</td>
</tr>
<tr>
<td>0xBFFFF8FE</td>
<td>0x00</td>
</tr>
<tr>
<td>0xBFFFF8FF</td>
<td>0x4D</td>
</tr>
<tr>
<td>0xBFFFF900</td>
<td>0x08</td>
</tr>
<tr>
<td>0xBFFFF901</td>
<td>0x2C</td>
</tr>
<tr>
<td>0xBFFFF902</td>
<td>0x33</td>
</tr>
</tbody>
</table>

What would be the result of the statement:

```
printf(“The address of array b is %x \n”, b);
```

Solution:

The address of array b is bffff8f3
Practice Problem 5.6

Recall that in RAM you have stored the object code for your program as well as additional memory allocated for your variables within the program.

You type into the debugger the command

\[ \text{i r ebp} \]

and get the result \(0xbffff810\). Upon further review of the assembly code you determine that two integers are stored in memory, one at address \(\text{ebp} - 8\) and the other at \(\text{ebp} + 4\). What are the hidden decimal numbers?

Solution: At \(\text{ebp} - 8\) we have \(00000000\). This is zero.

At \(\text{ebp} + 4\) we have \(0x08048420\). Converting this to decimal yields 134,513,696.