This Quiz concerns the following C code:

```c
/* copy string x to buf */
void foo(char *x) {
    int buf[1];
    strcpy((char *)buf, x);
}

void callfoo() {
    foo("abcdefg hi");
}
```

Here is the corresponding machine code on a Linux/x86 machine:

```
080484f4 <foo>:   pushl  %ebp
080484f4:  55       pushl  %ebp
080484f4:  89 e5     movl   %esp,%ebp
080484f5:  83 ec 18   subl   $0x18,%esp
080484fa:  8b 45 08   movl   0x8(%ebp),%eax
080484fc:  83 c4 f8   addl   $0xffffffff8,%esp
08048500:  50       pushl  %eax
08048501:  8d 45 fc   leal   0xffffffff(%ebp),%eax
08048504:  50       pushl  %eax
08048505:  e8 ba fe ff ff  call   80483c4 <strcpy>
0804850a:  89 ec     movl   %ebp,%esp
0804850c:  5d       popl   %ebp
0804850d:  c3       ret

08048510 <callfoo>:    pushl  %ebp
08048510:  55       pushl  %ebp
08048511:  89 e5     movl   %esp,%ebp
08048512:  83 ec 08   subl   $0x8,%esp
08048516:  83 c4 f4   addl   $0xffffffff4,%esp
08048519:  68 9c 85 04 08  pushl  $0x804859c  { push string address}
0804851e:  e8 d1 ff ff ff  call   80484f4 <foo>
08048523:  89 ec     movl   %ebp,%esp
08048525:  5d       popl   %ebp
08048526:  c3       ret
```
This problem tests your understanding of the stack discipline and byte ordering. Here are some notes to help you work the problem:

- if you hope to solve this problem, then you need to draw yourself a picture of the stack at the point that strcpy is called showing foo’s frame.

- `strcpy(char *dst, char *src)` copies the string at address src (including the terminating ‘\0’ character) to address dst. It does not check the size of the destination buffer.

- Recall that Linux/x86 machines are Little Endian.

- You will need to know the hex values of the following characters:

<table>
<thead>
<tr>
<th>Character</th>
<th>Hex value</th>
<th>Character</th>
<th>Hex value</th>
</tr>
</thead>
<tbody>
<tr>
<td>'a'</td>
<td>0x61</td>
<td>'f'</td>
<td>0x66</td>
</tr>
<tr>
<td>'b'</td>
<td>0x62</td>
<td>'g'</td>
<td>0x67</td>
</tr>
<tr>
<td>'c'</td>
<td>0x63</td>
<td>'h'</td>
<td>0x68</td>
</tr>
<tr>
<td>'d'</td>
<td>0x64</td>
<td>'i'</td>
<td>0x69</td>
</tr>
<tr>
<td>'e'</td>
<td>0x65</td>
<td>'r'</td>
<td>0x72</td>
</tr>
<tr>
<td>'\0'</td>
<td>0x00</td>
<td>'s'</td>
<td>0x73</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'t'</td>
<td>0x74</td>
</tr>
</tbody>
</table>

Now consider what happens on a Linux/x86 machine when callfoo calls foo with the input string “bestberight”.

- List the contents of the following memory locations immediately after strcpy returns to foo. Each answer should be an unsigned 4-byte integer expressed as 8 hex digits.

  `buf[0] = 0x74 73 65 62`  
  `buf[1] = 0x69 72 65 62`  
  `buf[2] = 0x64 74 68 67`

- Immediately before the ret instruction at address 0x0804850d executes, what is the value of the frame pointer register `%ebp`?

  `%ebp = 0x69 72 65 62`

- Immediately after the ret instruction at address 0x0804850d executes, what is the value of the program counter register `%eip`?

  `%eip = 0xd4 74 68 67`