1. **Move and Load Effective Address Instructions** What value would be stored in register %ebx after each of the following operations?

<table>
<thead>
<tr>
<th>Memory Address</th>
<th>EAX</th>
<th>ECX</th>
<th>EDX</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100</td>
<td>0x8C</td>
<td>0x100</td>
<td>0x1</td>
</tr>
<tr>
<td>0x104</td>
<td>0x86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x108</td>
<td>0x29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10C</td>
<td>0x19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x110</td>
<td>0x10C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x114</td>
<td>0x100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x118</td>
<td>0xFF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) 1 point  
\[
\text{movl } 0x10C, \%ebx
\]

(b) 1 point  
\[
\text{movl } \$0x10C, \%ebx
\]

(c) 1 point  
\[
\text{movl } \%eax, \%ebx
\]

(d) 1 point  
\[
\text{movl } (\%eax), \%ebx
\]

(e) 1 point  
\[
\text{movl } 4(\%eax), \%ebx
\]

(f) 1 point  
\[
\text{leal } 4(\%eax), \%ebx
\]

(g) 1 point  
\[
\text{movl } 8(\%eax, \%ecx, 4), \%ebx
\]

(h) 1 point  
\[
\text{leal } 8(\%eax, \%ecx, 4), \%ebx
\]

(i) 1 point  
\[
\text{leal } 8(, \%ecx, 4), \%ebx
\]
2. Consider the following C code.

```c
int fb(int A[], int thresh, int cap) {
    int i, count=0;
    for (i=0; i<cap; i++) {
        if (A[i]>=thresh) {
            count++;
        }
    }
    return count;
}
```

which, when compiled is translated to the following assembly:

```assembly
.type fb, @function
fb:
pushl %ebp
movl %esp, %ebp
...
/* several other lines here */
...
movl %ebp, %esp
popl %ebp
ret
```

(a) We see the function is returning a count. Where exactly is this stored so that it can later be read by the caller?

(a) ______________

(b) During most of fb, i.e., after the mov instruction in the 2nd line of the function, but before the pop instruction at the end, the value stored in ebp is 0xFFFFD500. If it can be determined based on the information given, what is stored at each of the given addresses. If it cannot be determined, write can’t tell?

i. 0xFFFFD500?

i. ______________

ii. 0xFFFFD504?

ii. ______________

iii. 0xFFFFD508?

iii. ______________

iv. 0xFFFFD50C?

iv. ______________

v. 0xFFFFD510?

v. ______________