NASM Basics III Using Registers and RAM (Practice)

ICS312

Machine-Level and Systems Programming

Henri Casanova (henric@hawaii.edu)

(q1) Bytes read/written

- For each instruction below, say how many bytes are written or read to/from RAM:
  - `mov [L1], eax`
  - `mov al, [L2]`
  - `mov dword [L1], 0FFFFFFFFh`
  - `mov dword [L1], 0FFh`
  - `mov byte [L2], 12`
  - `mov ax, [L3]`

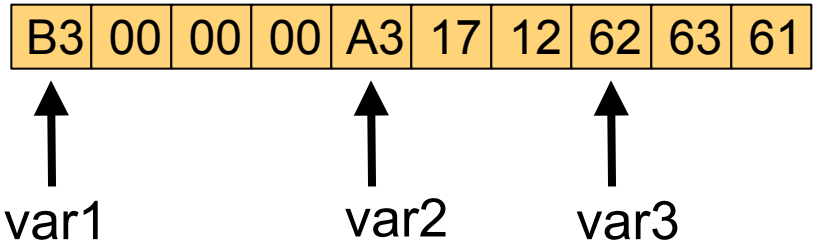
(q1) Solutions

- For each instruction below, say how many bytes are written or read to/from RAM:

```
mov    [L1], eax                  ; 4 written
mov    al, [L2]                   ; 1 read
mov    dword [L1], 0FFFFFFFFh     ; 4 written
mov    dword [L1], 0FFh           ; 4 written
mov    byte [L2], 12              ; 1 written
mov    ax, [L3]                   ; 2 read
```

(q2) Program Tracing

- Consider the following data segment

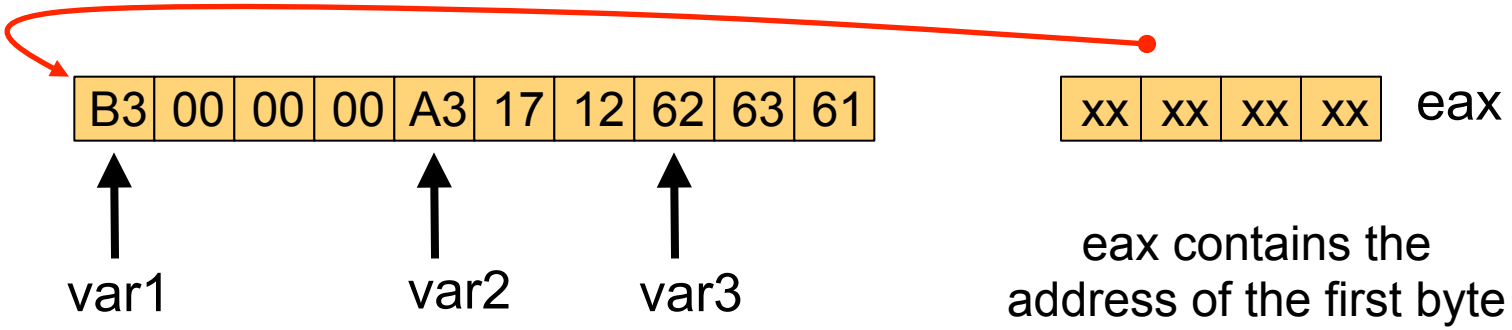


- What is the content of the data segment (on a Little Endian machine) after this program runs?

```
mov     eax, var1
add     eax, 3
mov     ebx, [eax]
add     ebx, 5
mov     [var1], ebx
```

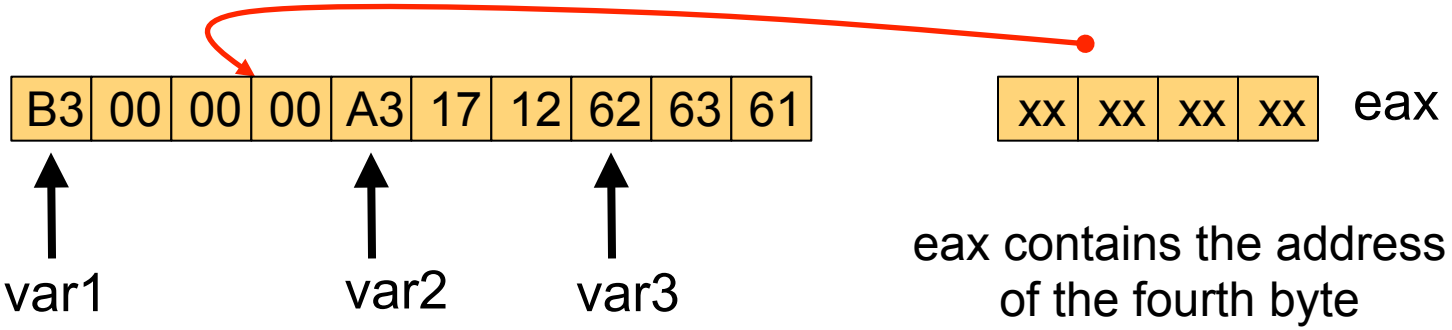
(q2) Solutions

```
mov    eax, var1
add    eax, 3
mov    ebx, [eax]
add    ebx, 5
mov    [var1], ebx
```



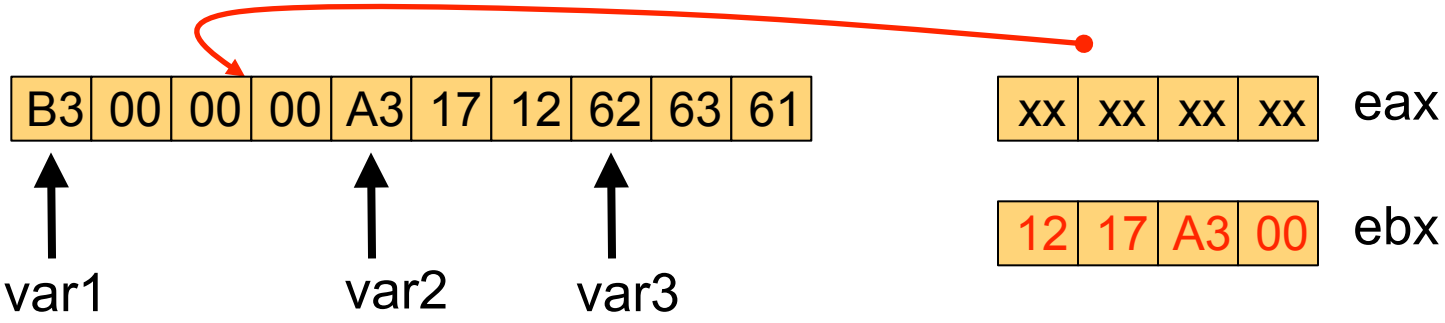
(q2) Solutions

```
mov    eax, var1
add    eax, 3
mov    ebx, [eax]
add    ebx, 5
mov    [var1], ebx
```



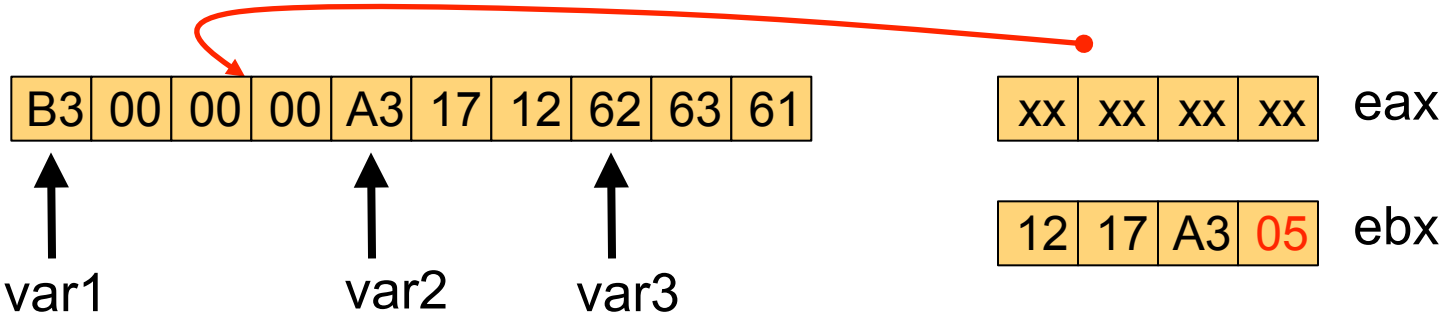
(q2) Solutions

```
mov    eax, var1
add    eax, 3
mov    ebx, [eax]
add    ebx, 5
mov    [var1], ebx
```



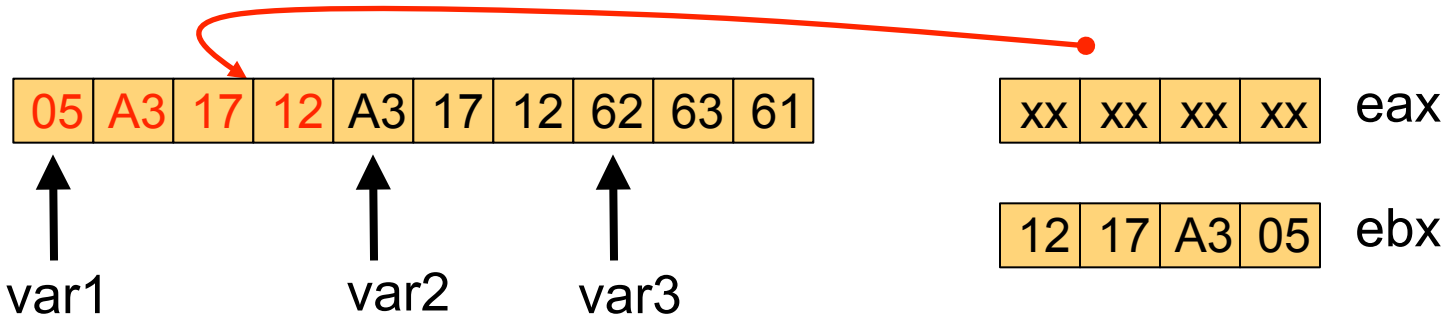
(q2) Solutions

```
mov    eax, var1
add    eax, 3
mov    ebx, [eax]
add    ebx, 5
mov    [var1], ebx
```



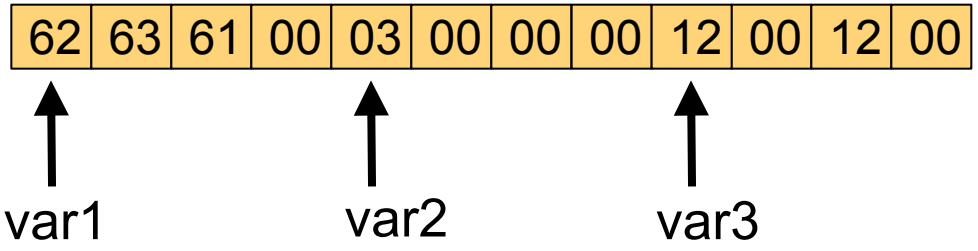
(q2) Solutions

```
mov    eax, var1
add    eax, 3
mov    ebx, [eax]
add    ebx, 5
mov    [var1], ebx
```



(q3) Program Tracing

- Consider the following data segment

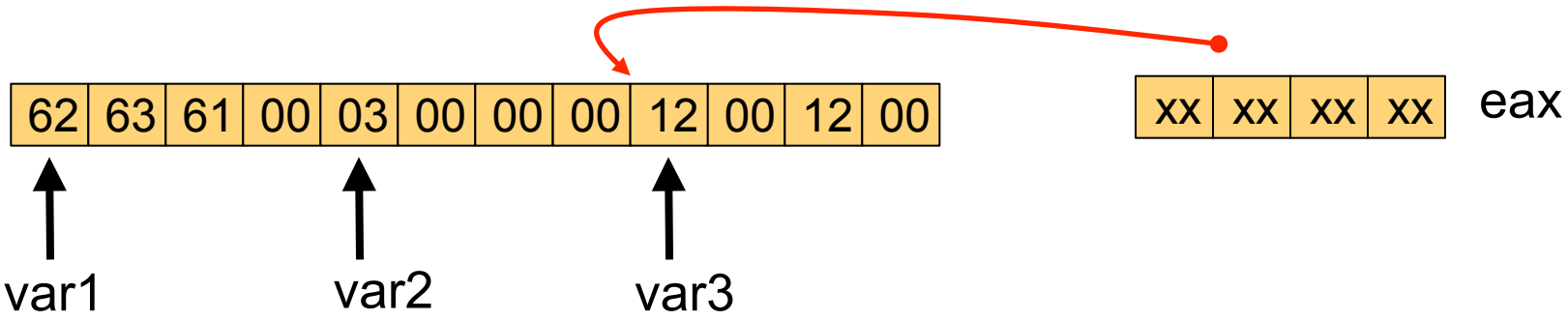


- What is the content of the data segment (on a Little Endian machine) after this program runs?

```
mov     eax, var3
mov     ebx, var1
sub     eax, 4
add     ebx, [eax]
mov     dword [ebx], 42
```

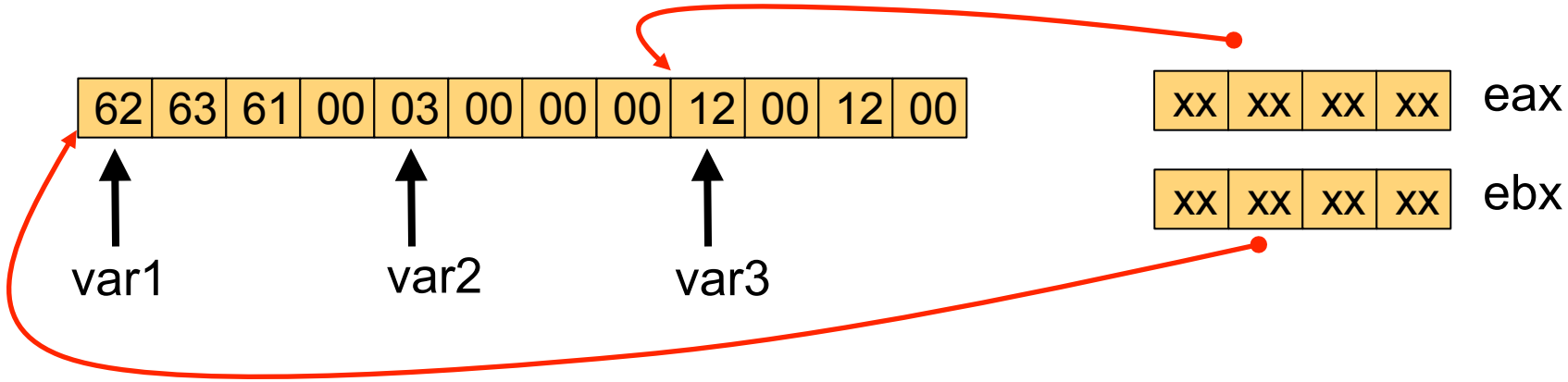
(q3) Solutions

```
mov     eax, var3
mov     ebx, var1
sub     eax, 4
add     ebx, [eax]
mov     dword [ebx], 42
```



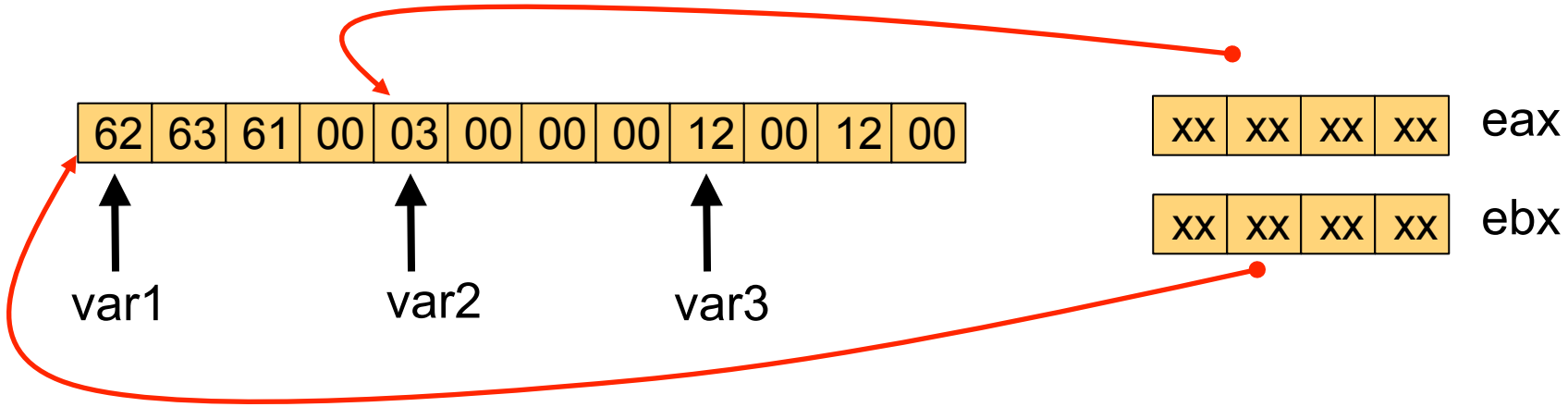
(q3) Solutions

```
mov     eax, var3
mov     ebx, var1
sub     eax, 4
add     ebx, [eax]
mov     dword [ebx], 42
```



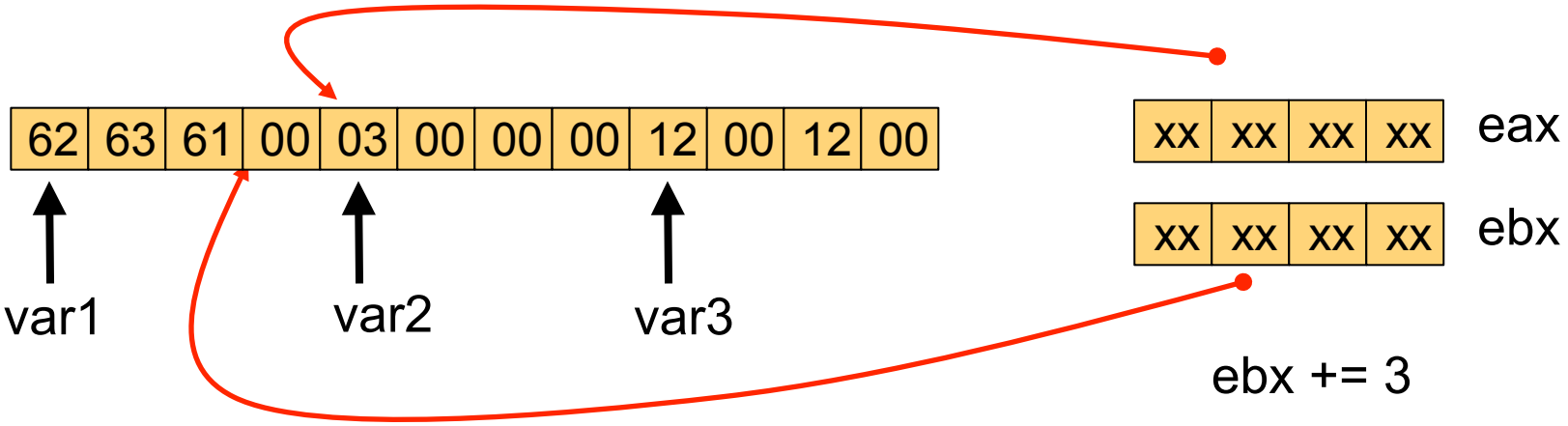
(q3) Solutions

```
mov     eax, var3
mov     ebx, var1
sub     eax, 4
add     ebx, [eax]
mov     dword [ebx], 42
```



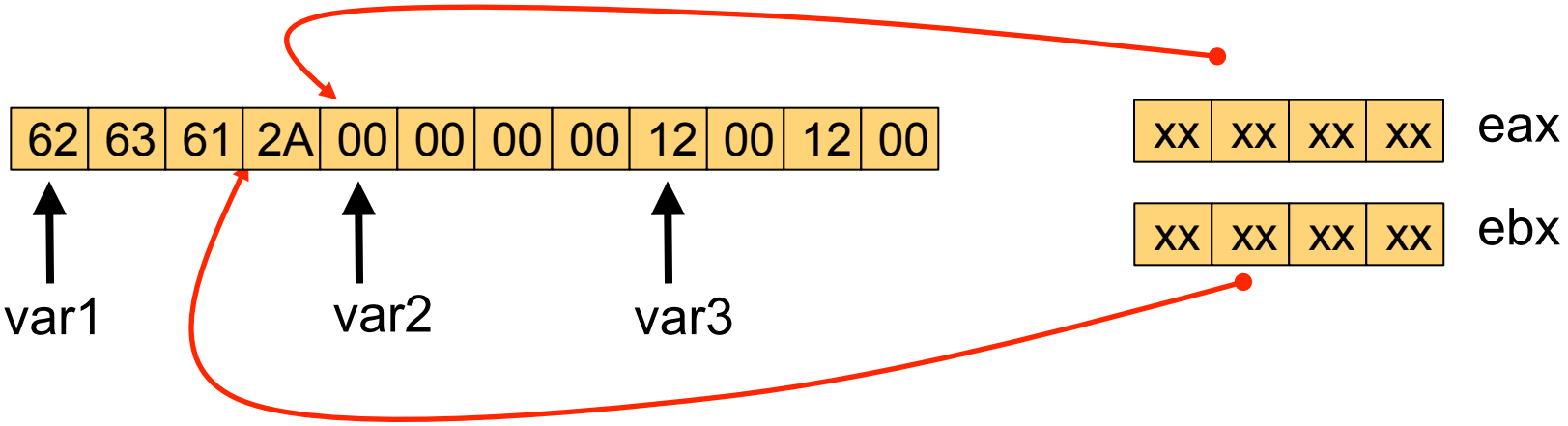
(q3) Solutions

```
mov     eax, var3
mov     ebx, var1
sub     eax, 4
add     ebx, [eax]
mov     dword [ebx], 42
```



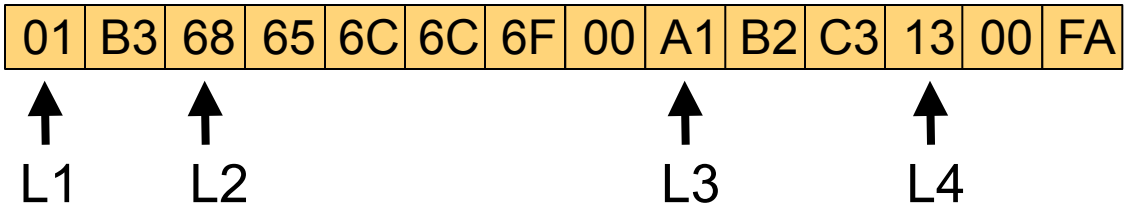
(q3) Solutions

```
mov     eax, var3
mov     ebx, var1
sub     eax, 4
add     ebx, [eax]
mov     dword [ebx], 42
```



(q4) Program Tracing

- Consider the following data segment

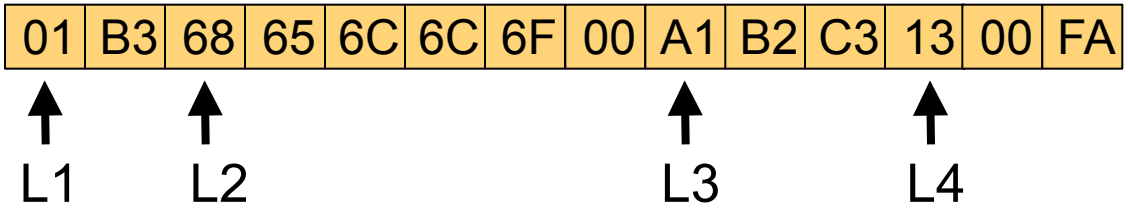


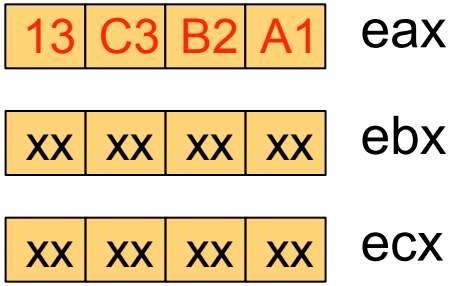
- What is the content of the data segment (on a Little Endian machine) after this program runs?

```
mov     eax, [L3]
inc     eax
mov     [L2], eax
mov     bx, [L4]
mov     eax, L3
mov     ecx, [L1]
add     bl, cl
mov     [eax], bx
```

(q4) Solutions

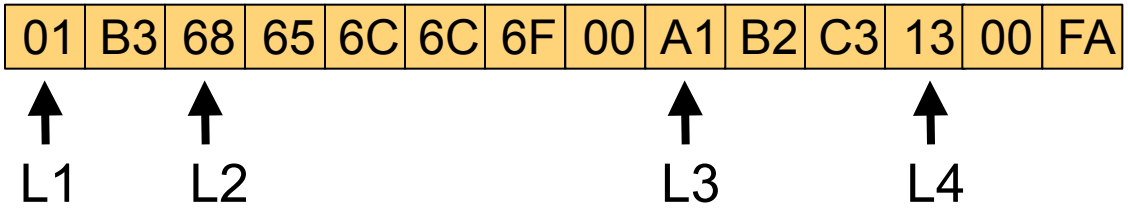
```
mov     eax, [L3]
inc     eax
mov     [L2], eax
mov     bx, [L4]
mov     eax, L3
mov     ecx, [L1]
add     bl, cl
mov     [eax], bx
```

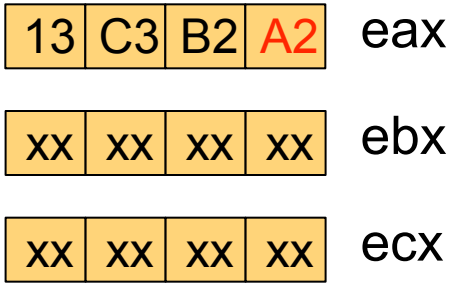




(q4) Solutions

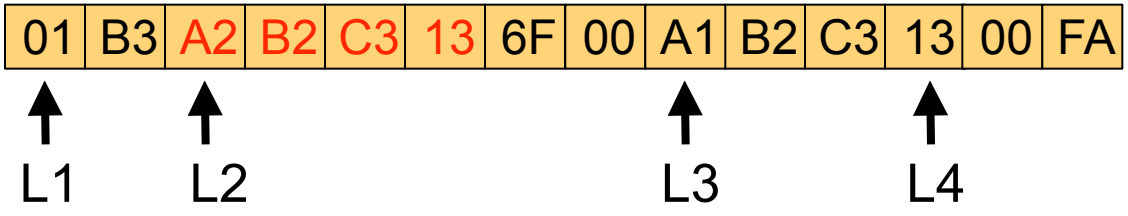
```
mov     eax, [L3]
inc     eax
mov     [L2], eax
mov     bx, [L4]
mov     eax, L3
mov     ecx, [L1]
add     bl, cl
mov     [eax], bx
```

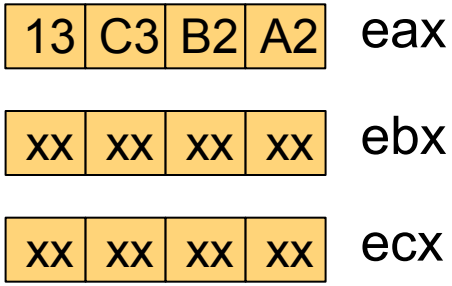




(q4) Solutions

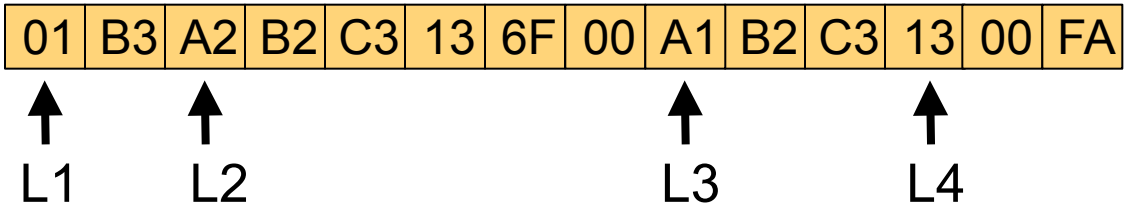
```
mov     eax, [L3]
inc     eax
mov     [L2], eax
mov     bx, [L4]
mov     eax, L3
mov     ecx, [L1]
add     bl, cl
mov     [eax], bx
```

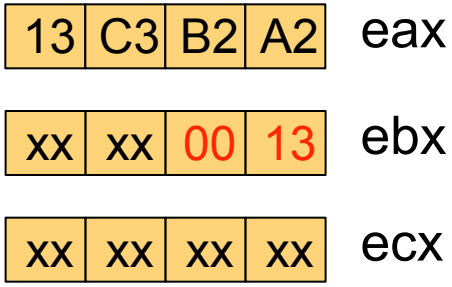




(q4) Solutions

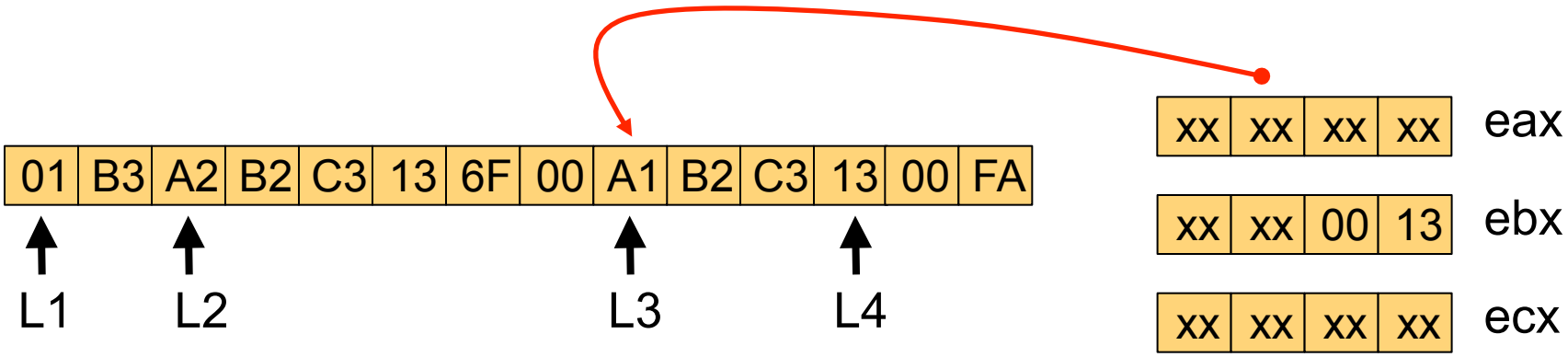
```
mov     eax, [L3]
inc     eax
mov     [L2], eax
mov     bx, [L4]
mov     eax, L3
mov     ecx, [L1]
add     bl, cl
mov     [eax], bx
```





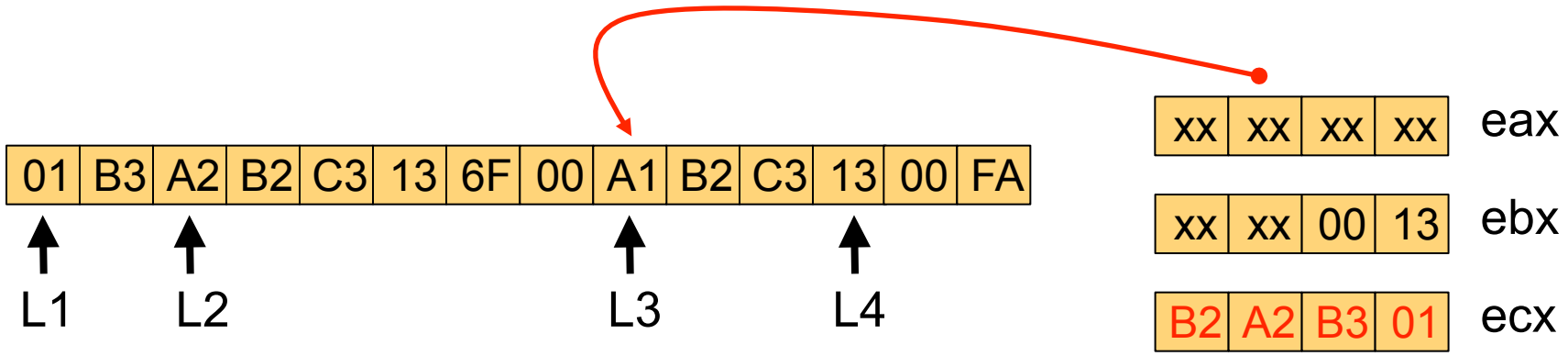
(q4) Solutions

```
mov     eax, [L3]
inc     eax
mov     [L2], eax
mov     bx, [L4]
mov     eax, L3
mov     ecx, [L1]
add     bl, cl
mov     [eax], bx
```



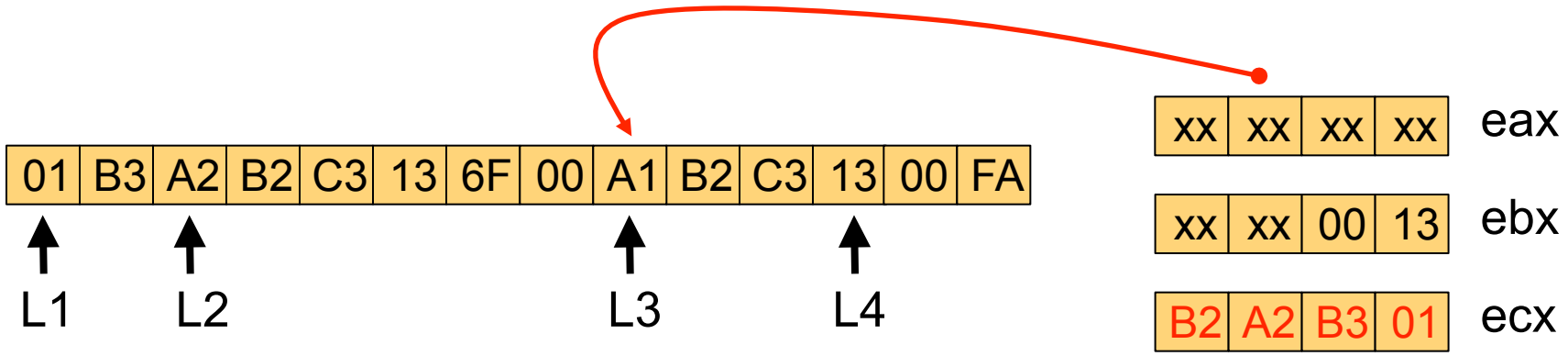
(q4) Solutions

```
mov     eax, [L3]
inc     eax
mov     [L2], eax
mov     bx, [L4]
mov     eax, L3
mov     ecx, [L1]
add     bl, cl
mov     [eax], bx
```



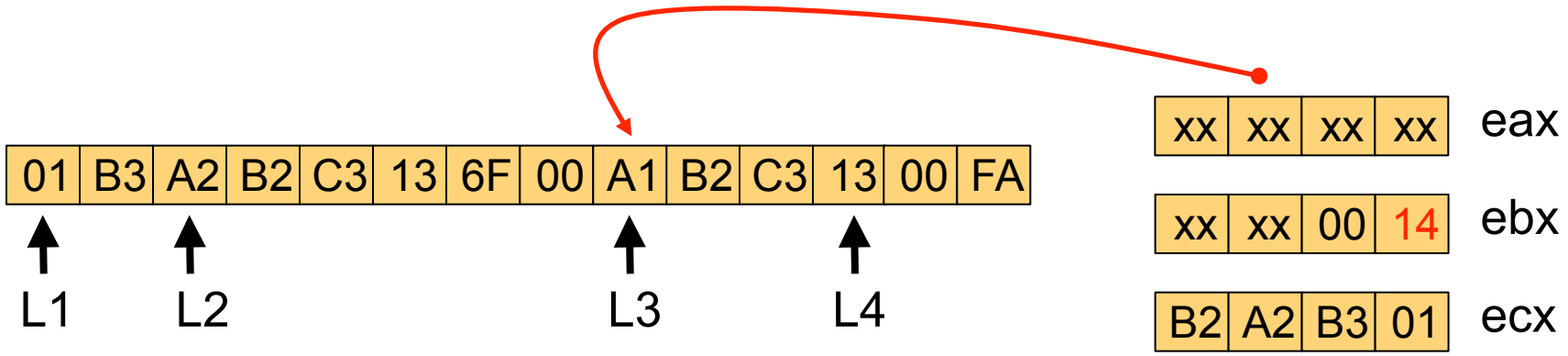
(q4) Solutions

```
mov     eax, [L3]
inc     eax
mov     [L2], eax
mov     bx, [L4]
mov     eax, L3
mov     ecx, [L1]
add     bl, cl
mov     [eax], bx
```



(q4) Solutions

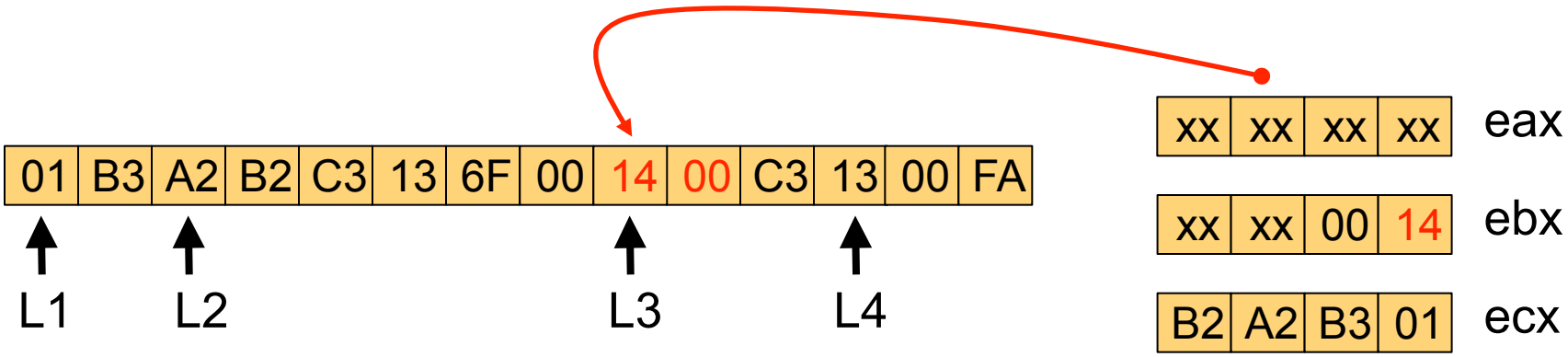
```
mov     eax, [L3]
inc     eax
mov     [L2], eax
mov     bx, [L4]
mov     eax, L3
mov     ecx, [L1]
add     bl, cl
mov     [eax], bx
```



13 + 01 = 14

(q4) Solutions

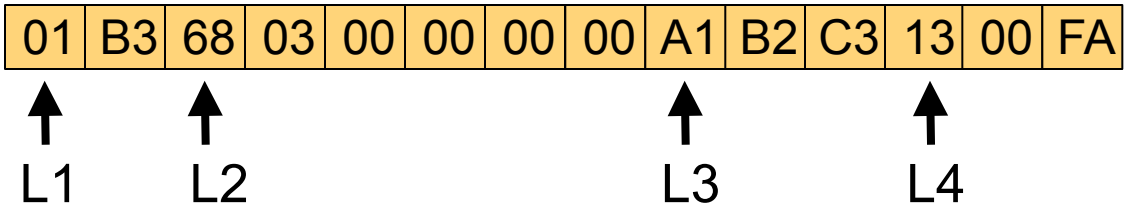
```
mov     eax, [L3]
inc     eax
mov     [L2], eax
mov     bx, [L4]
mov     eax, L3
mov     ecx, [L1]
add     bl, cl
mov     [eax], bx
```



13 + 01 = 14

(q5) Program Tracing

- Consider the following data segment

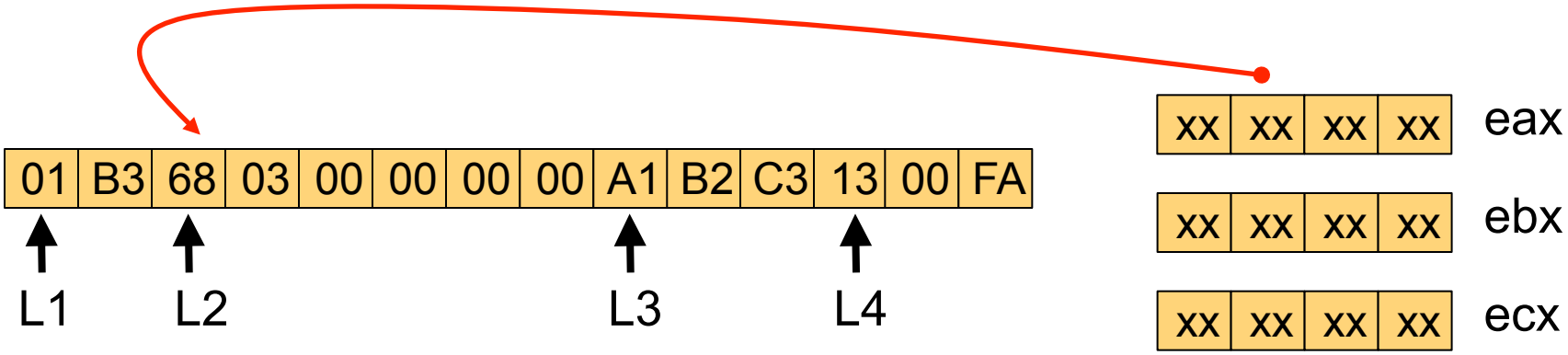


- What is the content of the data segment (on a Little Endian machine) after this program runs?

```
mov     eax, L2
inc     eax
mov     ebx, [eax]
neg     ebx
mov     eax, L4
add     eax, ebx
mov     [eax], ebx
```

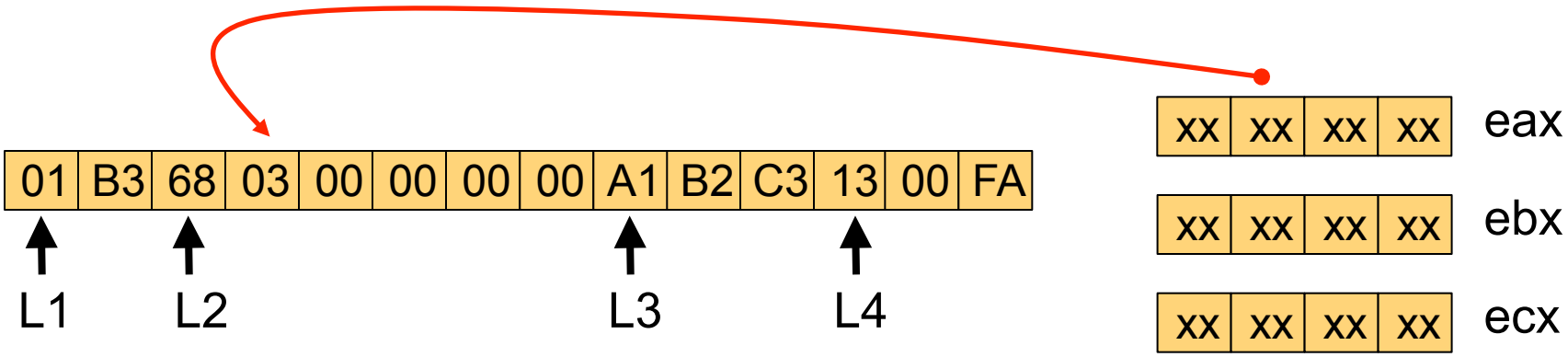
(q5) Solutions

```
mov     eax, L2
inc     eax
mov     ebx, [eax]
neg     ebx
mov     eax, L4
add     eax, ebx
mov     [eax], ebx
```



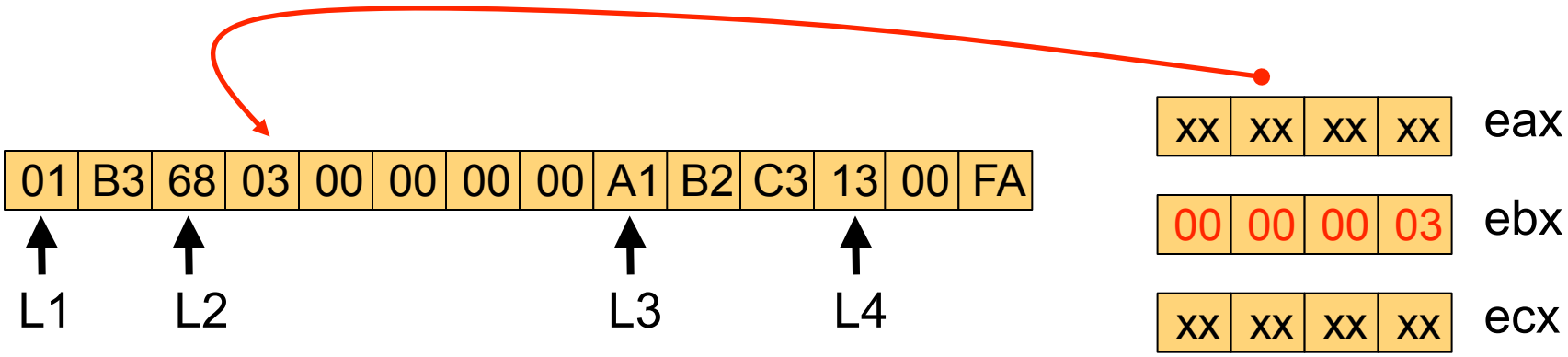
(q5) Solutions

```
mov     eax, L2
inc     eax
mov     ebx, [eax]
neg     ebx
mov     eax, L4
add     eax, ebx
mov     [eax], ebx
```



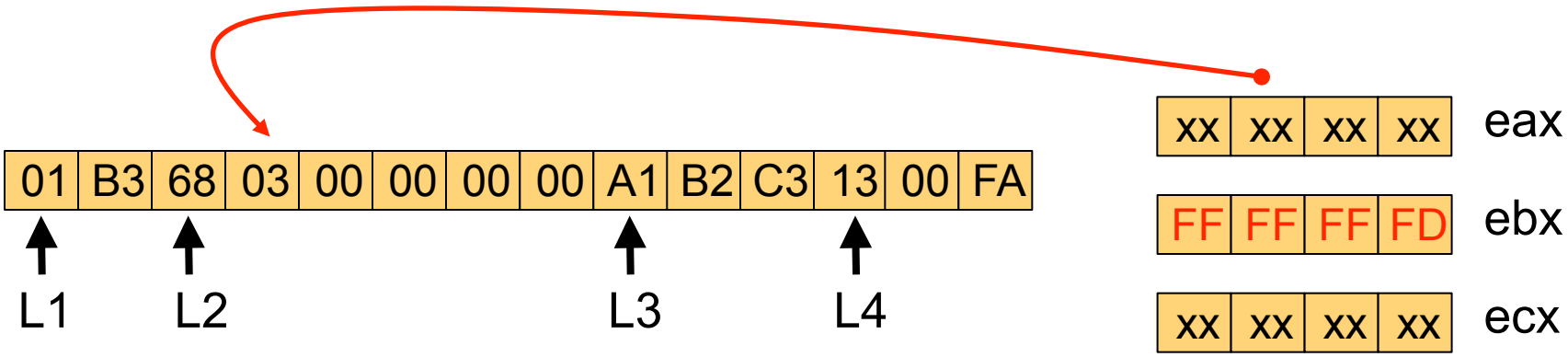
(q5) Solutions

```
mov     eax, L2
inc     eax
mov     ebx, [eax]
neg     ebx
mov     eax, L4
add     eax, ebx
mov     [eax], ebx
```



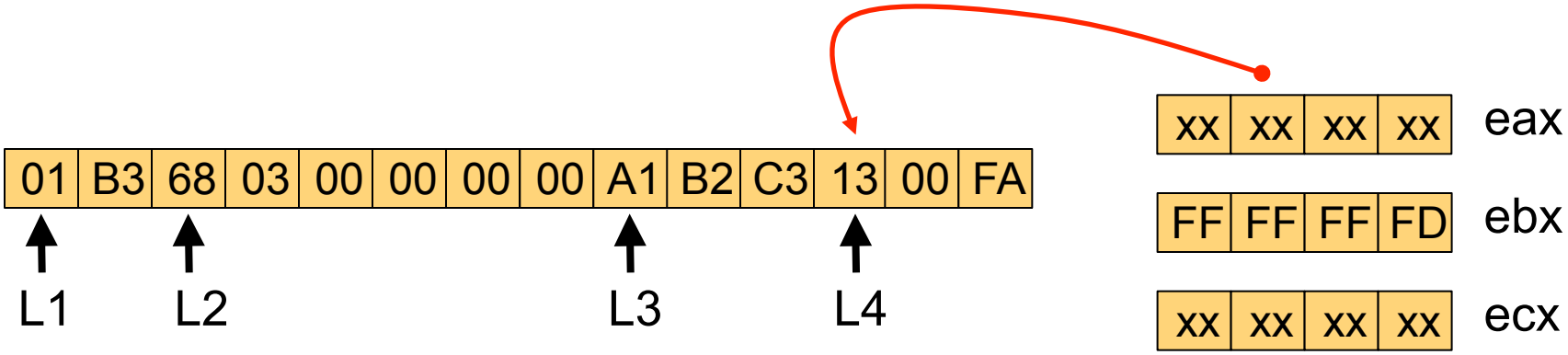
(q5) Solutions

```
mov     eax, L2
inc     eax
mov     ebx, [eax]
neg     ebx
mov     eax, L4
add     eax, ebx
mov     [eax], ebx
```



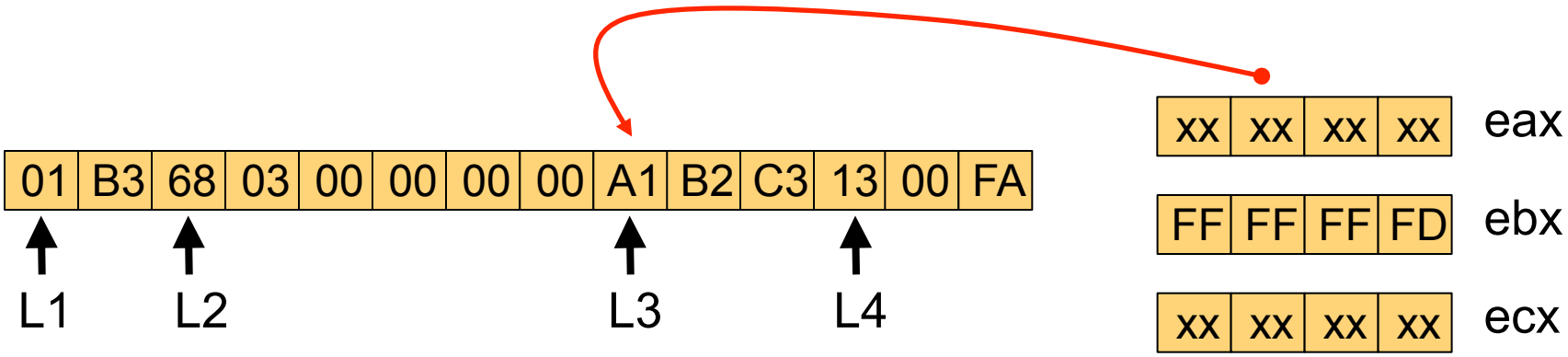
(q5) Solutions

```
mov     eax, L2
inc     eax
mov     ebx, [eax]
neg     ebx
mov     eax, L4
add     eax, ebx
mov     [eax], ebx
```



(q5) Solutions

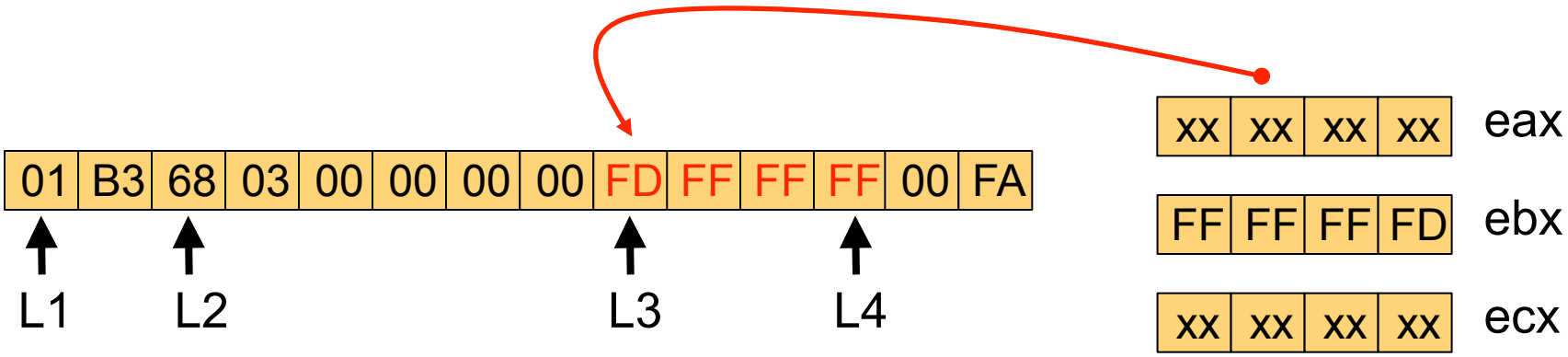
```
mov     eax, L2
inc     eax
mov     ebx, [eax]
neg     ebx
mov     eax, L4
add     eax, ebx
mov     [eax], ebx
```



Adding FF FF FF FD is the same as subtracting 00 00 00 03!!

(q5) Solutions

```
mov     eax, L2
inc     eax
mov     ebx, [eax]
neg     ebx
mov     eax, L4
add     eax, ebx
mov     [eax], ebx
```



(q6) Low-/High-Level Indirection

- Translate this fragment of C code to assembly

```
char *L1;
*((short *)((int *)L1 + 4)) += 1;
```

(q6) Solutions

- Translate this fragment of C code to assembly

```
char *L1;
*((short *)((int *)L1 + 4)) += 1;
```

```
inc word [L1 + 16]
```

(q7) Low-/High-Level Indirection

- Does this program perform an out-of-bounds memory write?

```
short array[4] = {1, 2, 3, 4};
char *ptr = (char *)array;
*(ptr + 5) = 'a';
```

(q7) Solutions

- Does this program perform an out-of-bounds memory write?

```
short array[4] = {1, 2, 3, 4};
char *ptr = (char *)array;
*(ptr + 5) = 'a';
```

- NO: it updates the 5-th byte of the array, and the array is an 8-byte array!
  - The new array content is: {1, 2, 24835, 4}