Modifying Data Sizes (Casting) (Practice)

ICS312

Machine-Level and Systems Programming

Henri Casanova (henric@hawaii.edu)

(q1) Size reduction

- For each fragment of assembly below, say whether the size-reduced value in register al is numerically valid or not

```
mov  bx, 0FF12h
mov  al, bl
```

```
mov  bx, 001AAh
mov  al, bl
```

```
mov  bx, 0007Ah
mov  al, bl
```

```
mov  bx, 0009Ah
mov  al, bl
```

(q1) Solutions

- For each fragment of assembly below, say whether the size-reduced value in register al is numerically valid or not

```
mov   bx, 0FF12h
mov   al, bl
```

Signed: invalid
Unsigned: invalid

```
mov   bx, 001AAh
mov   al, bl
```

Signed: invalid
Unsigned: invalid

```
mov   bx, 0007Ah
mov   al, bl
```

Signed: valid
Unsigned: valid

```
mov   bx, 0009Ah
mov   al, bl
```

Signed: invalid
Unsigned: valid

Note the difference between these two!

(q2) movzx/movsx

- For each fragment of assembly below, give the value of register eax at the end

```
mov  bx, -1
movsx eax, bx
```

```
mov  bl, -1
movzx eax, bl
```

```
mov  bx, 12
movzx eax, bx
```

```
mov  bx, 12
movsx eax, bx
```

(q2) Solutions

- For each fragment of assembly below, give the value of register eax at the end

```
mov  bx, -1
movsx eax, bx
```

FF FF FF FF

```
mov  bl, -1
movzx eax, bl
```

00 00 00 FF

```
mov  bx, 12
movzx eax, bx
```

00 00 00 0C

```
mov  bx, 12
movsx eax, bx
```

00 00 00 0C

(q3) movzx/movsx

- For each fragment of C below, say whether the compiler will use **`movsx`** or **`movzx`**

```
short a = -3;
unsigned int b = a;
```

```
unsigned short a = -3;
unsigned int b = a;
```

```
char a = 13;
int b = a;
```

```
char a = 13;
unsigned int b = a;
```

(q3) Solutions

- For each fragment of C below, say whether the compiler will use **movsx** or **movzx**

```
short a = -3;
unsigned int b = a;
```

movsx

```
unsigned short a = -3;
unsigned int b = a;
```

movzx

```
char a = 13;
int b = a;
```

movsx

```
char a = 13;
unsigned int b = a;
```

movsx

Remember that the choice movsx/movzx depends on the declared type of the value whose size is to be increased, not that of the destination variable!

(q4) movzx/movsx

- For each fragment of C below, give the value of the value of variable b (or just say “large >0 value”, etc.)

```
char a = -1;
short b = a;
```

```
unsigned char a = 12;
int b = a;
```

```
char a = -2;
unsigned short b = a;
```

```
unsigned char a = -1;  // yes, it's valid
int b = a;
```

(q4) movzx/movsx

- For each fragment of C below, give the value of the value of variable b (or just say “large >0 value”, etc.)

```
char a = -1;
short b = a;
```

-1

```
unsigned char a = 12;
int b = a;
```

12

```
char a = -2;
unsigned short b = a;
```

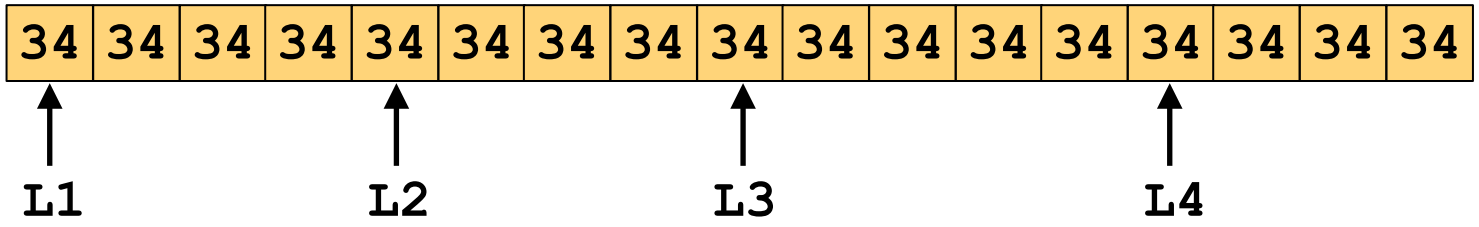
Large >0

```
unsigned char a = -1;  // yes, it's valid
int b = a;
```

255

(q5) Program Tracing

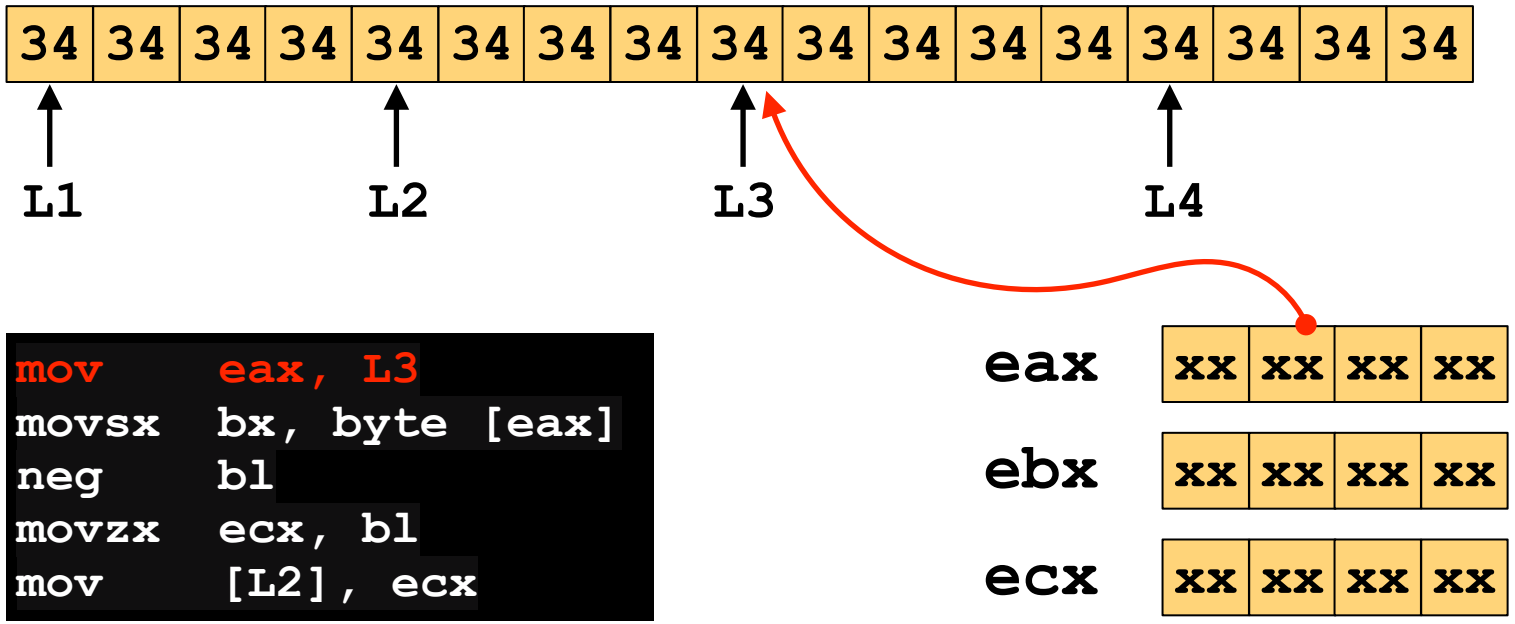
- Give the memory content after the program is done




```
mov     eax, L3
movsx   bx, byte [eax]
neg     bl
movzx   ecx, bl
mov     [L2], ecx
```

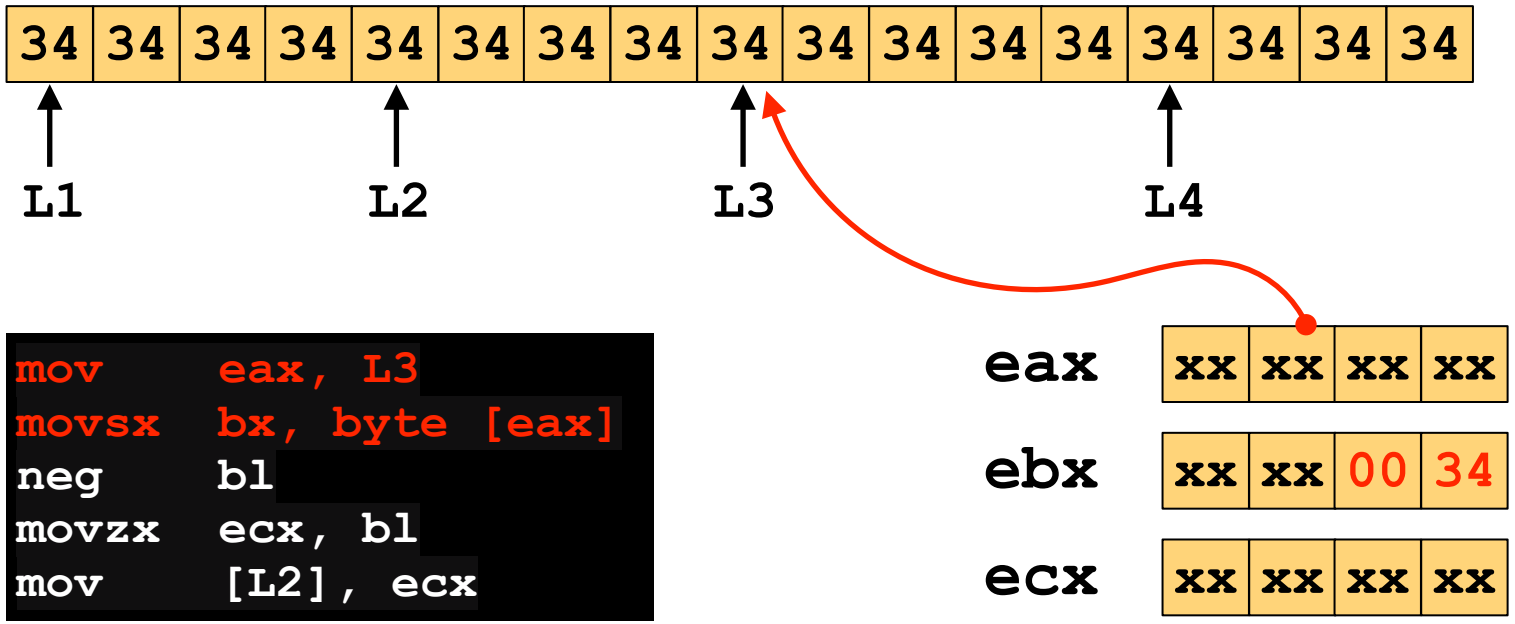
(q5) Solutions

- Give the memory content after the program is done



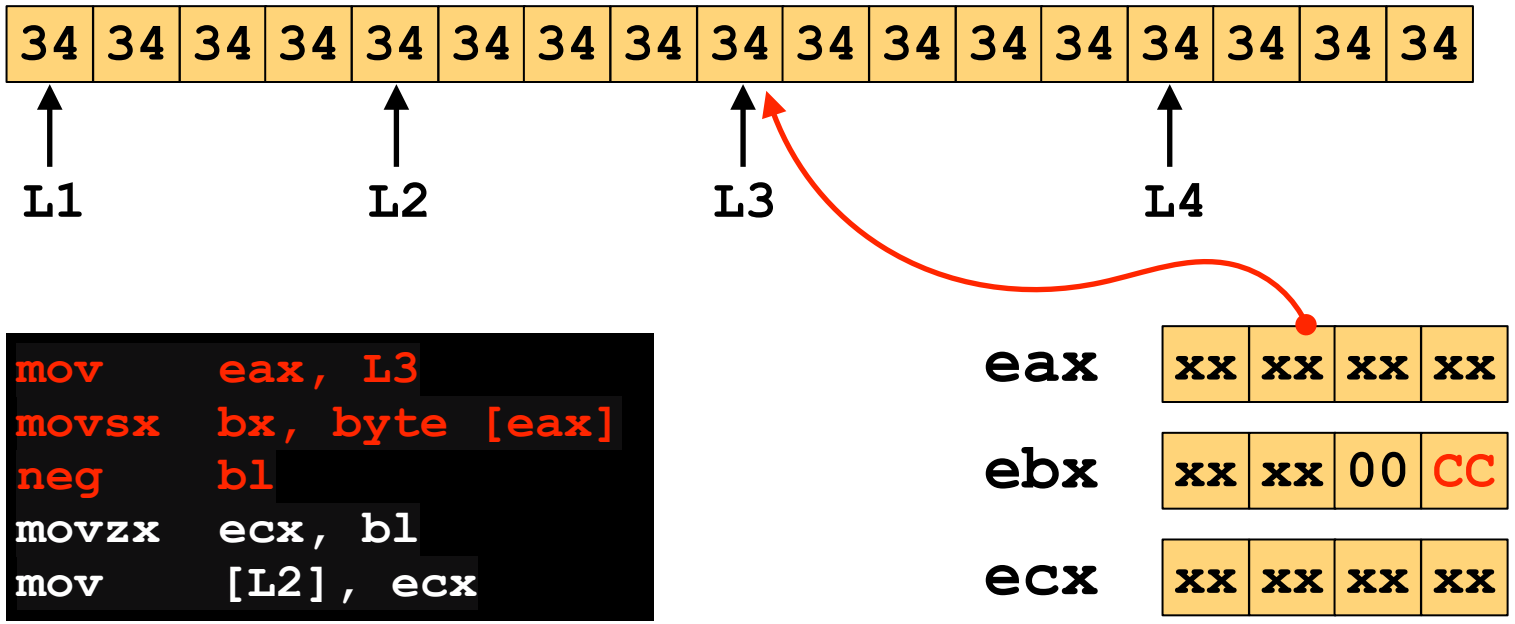
(q5) Solutions

- Give the memory content after the program is done



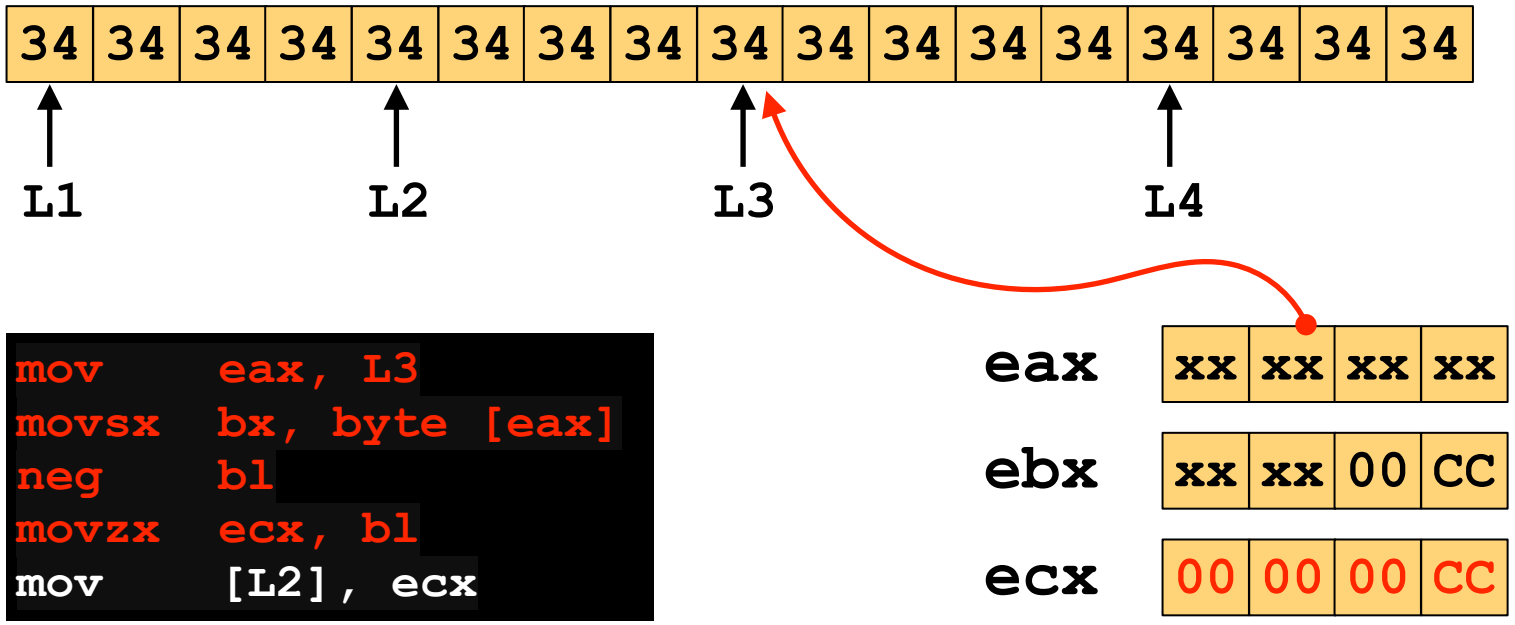
(q5) Solutions

- Give the memory content after the program is done



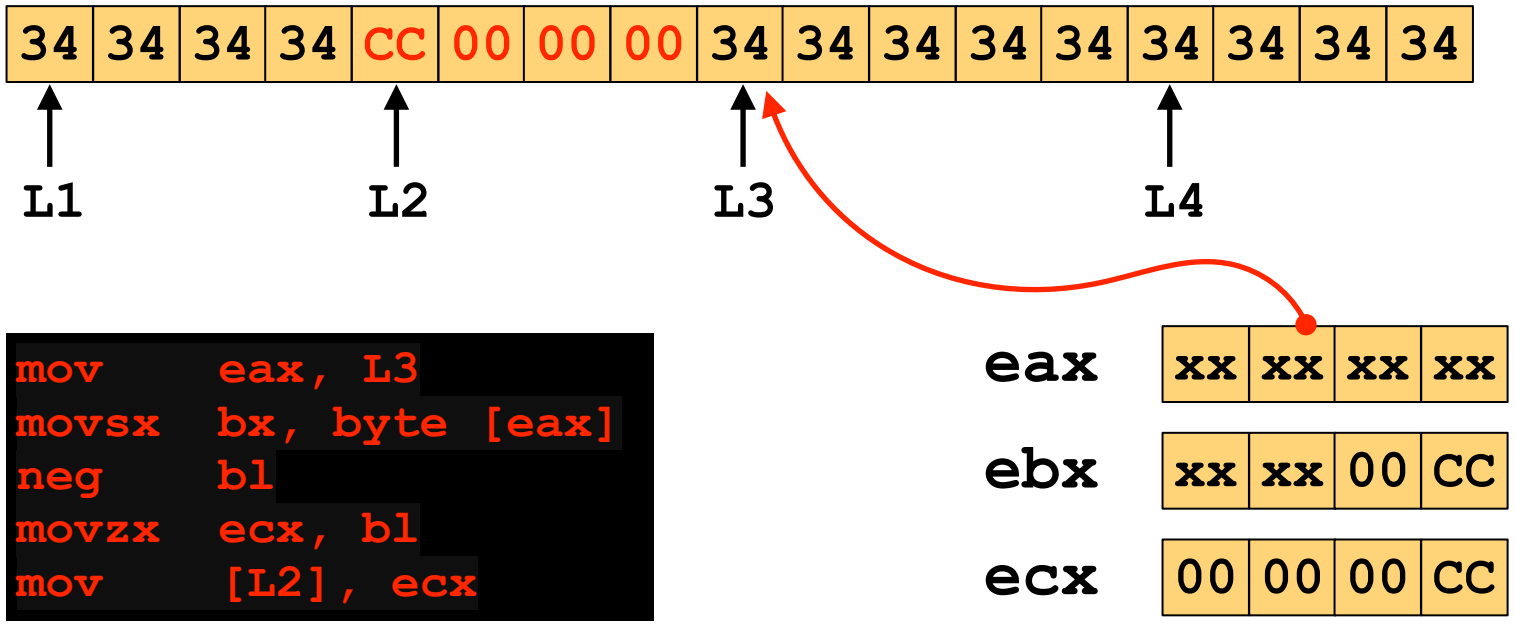
(q5) Solutions

- Give the memory content after the program is done



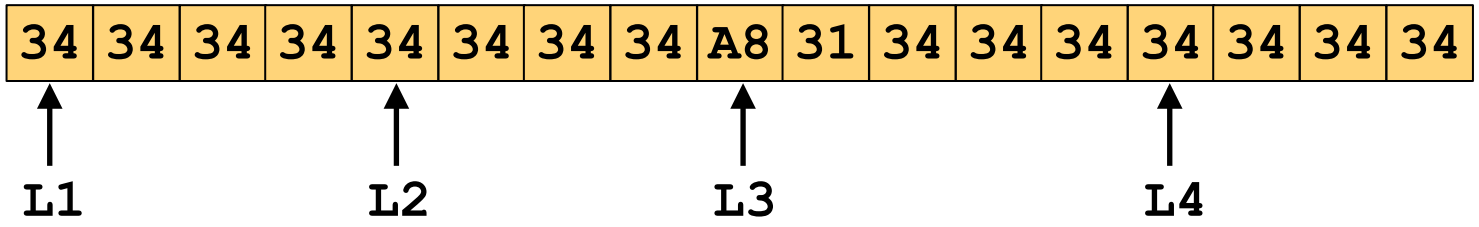
(q5) Solutions

- Give the memory content after the program is done



(q6) Program Tracing

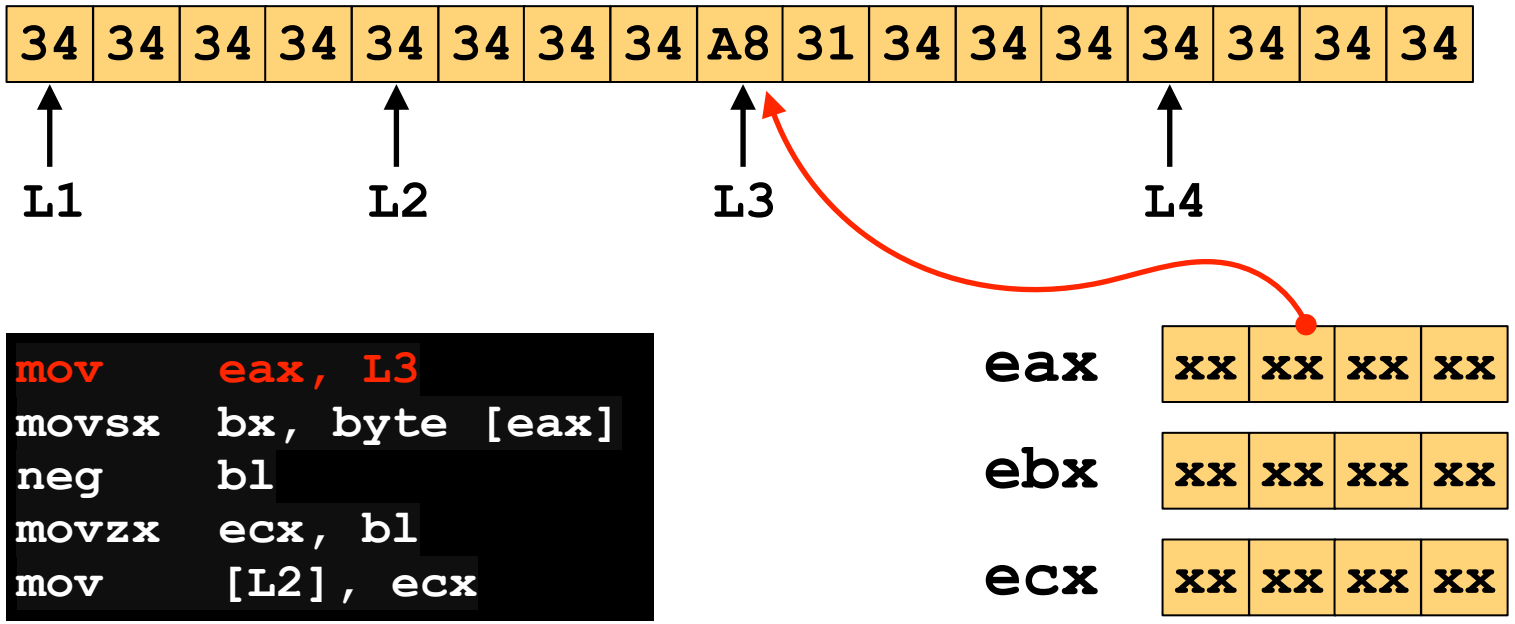
- Give the memory content after the program is done




```
mov     eax, L3
movsx   bx, byte [eax]
neg     bl
movzx   ecx, bl
mov     [L2], ecx
```

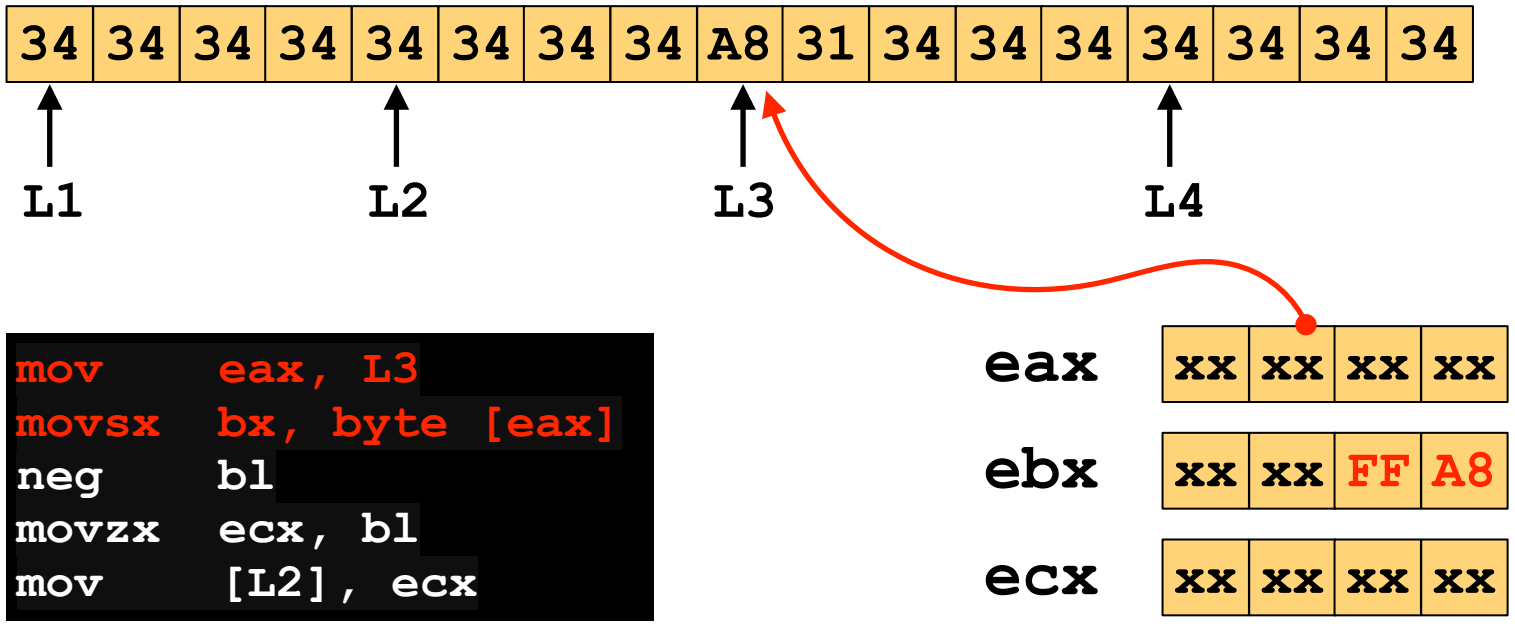
(q6) Solutions

- Give the memory content after the program is done



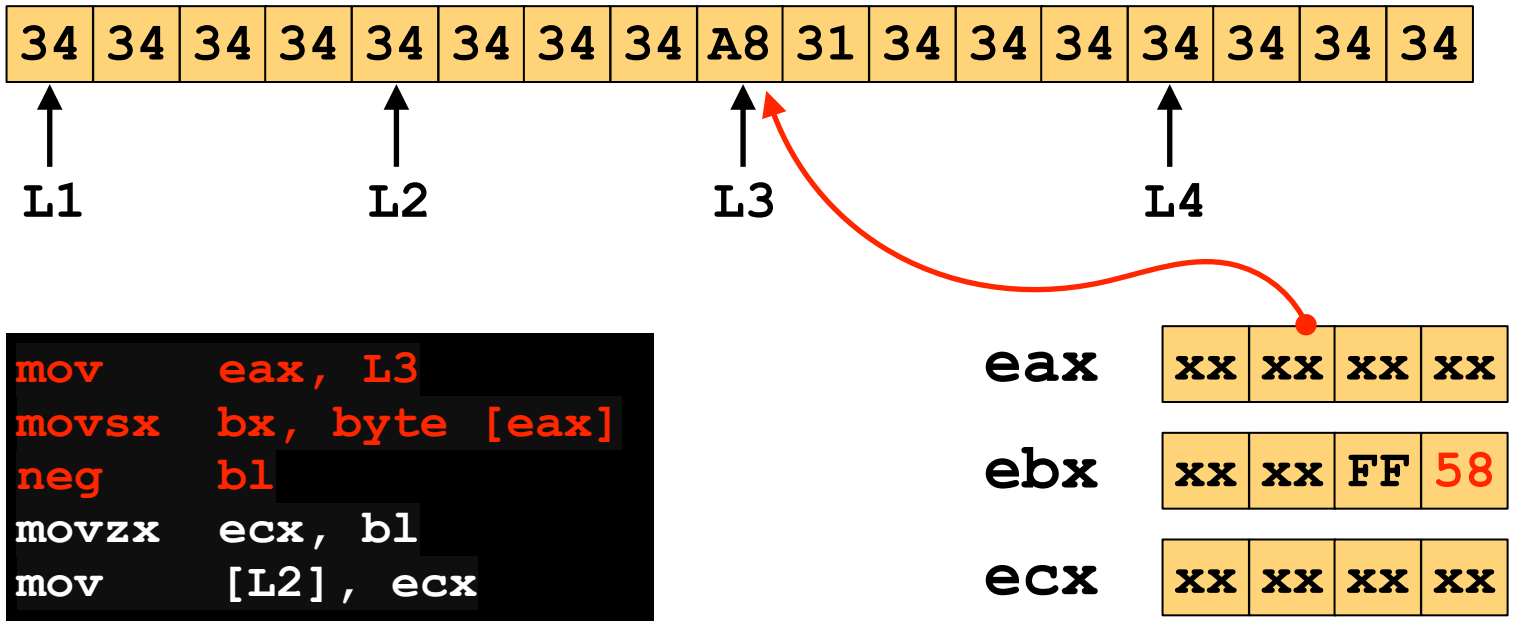
(q6) Solutions

- Give the memory content after the program is done



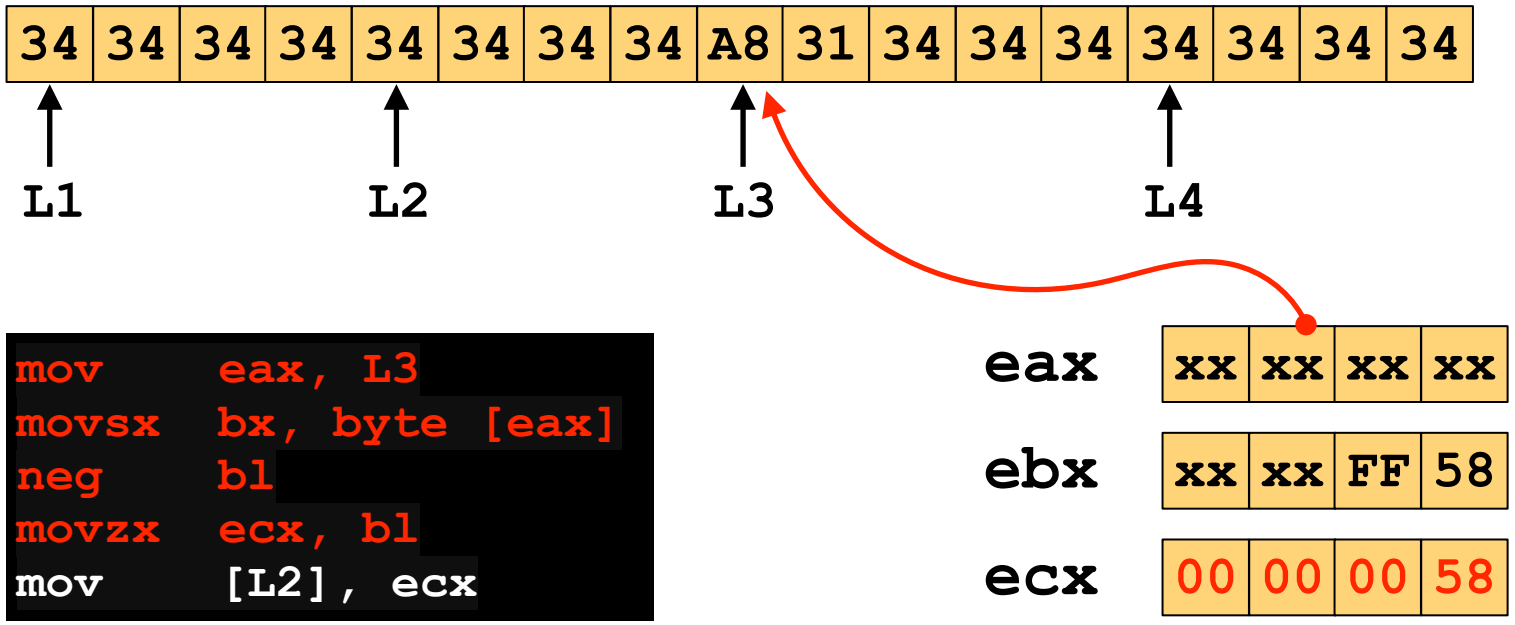
(q6) Solutions

- Give the memory content after the program is done



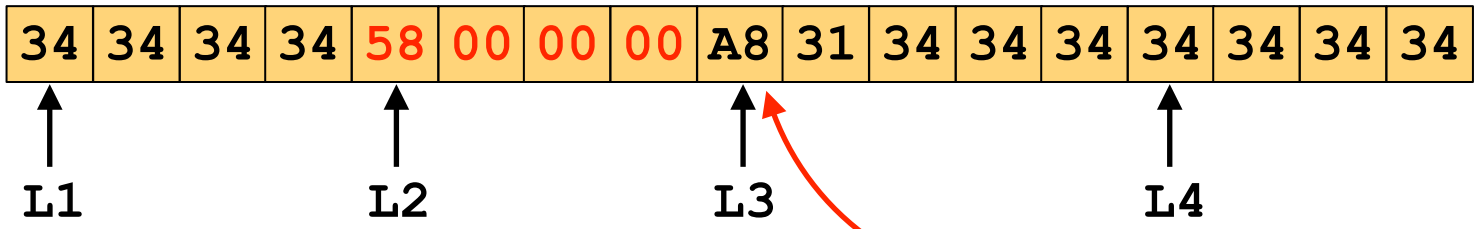
(q6) Solutions

- Give the memory content after the program is done



(q6) Solutions

- Give the memory content after the program is done




```
mov     eax, L3
movsx   bx, byte [eax]
neg     bl
movzx   ecx, bl
mov     [L2], ecx
```

eax	xx	xx	xx	xx
ebx	xx	xx	FF	58
ecx	00	00	00	58

(q6) Printf tracing

- What does the following C code print?

```
unsigned char a = -1;
int b = a;
char c = 0xA1;
unsigned short d = c + 2;

printf("%d\n", a);
printf("%d\n", b);
printf("%u\n", c);
printf("%u\n", d);
```

(q6) Solution

- What does the following C code print?

```
unsigned char a = -1;
int b = a;
char c = 0xA1;
unsigned short d = c + 2;

printf("%d\n", a); // prints 255
printf("%d\n", b); // prints 255
printf("%u\n", c); // prints a huge >0 number
printf("%u\n", d); // prints a large >0 number
```