NASM: data and bss Examples and Sample Problems

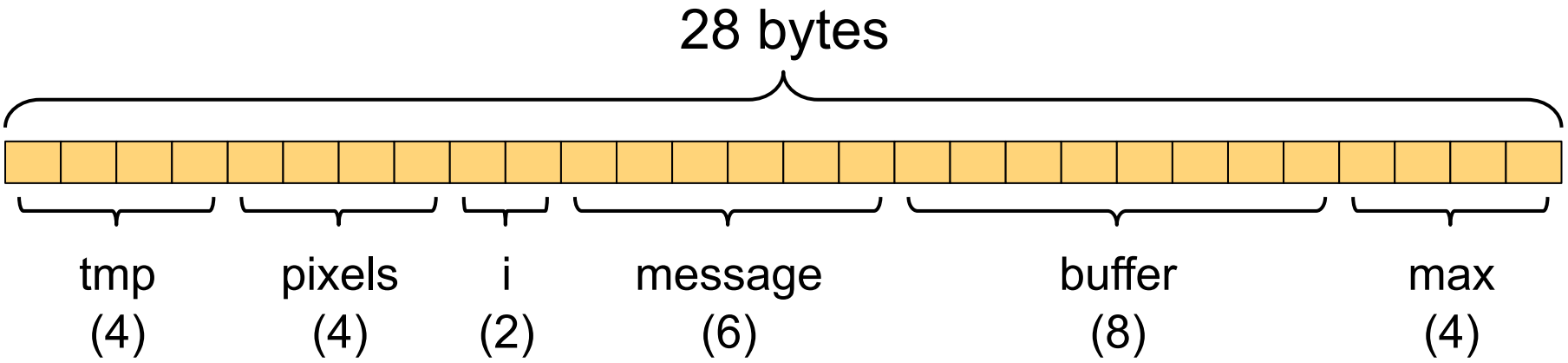
ICS312

Machine-Level and Systems Programming

Henri Casanova (henric@hawaii.edu)

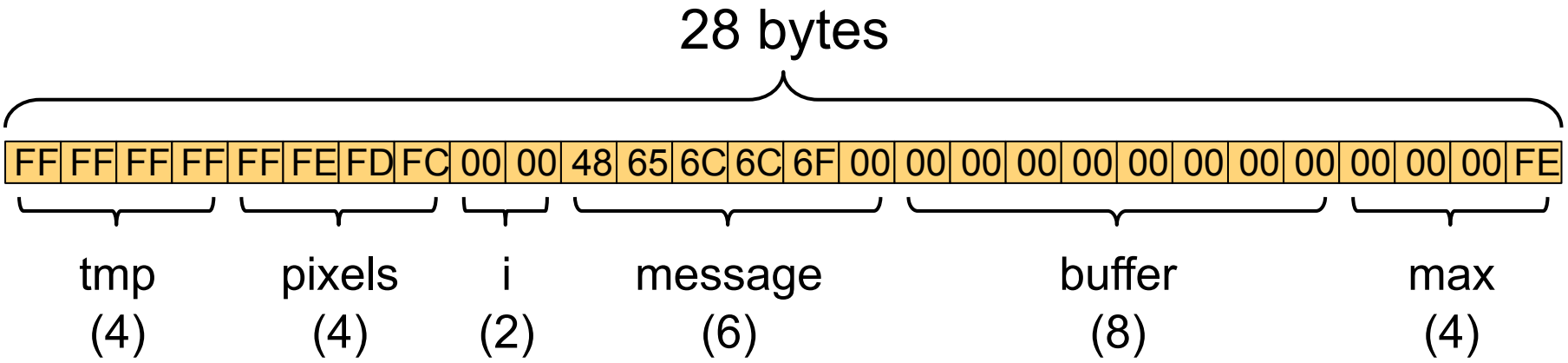
Data segment example

```
tmp         dd      -1
pixels      db      0FFh, 0FEh, 0FDh, 0FCh
i           dw      0
message     db      "H", "e", "llo", 0
buffer      times   8     db   0
max         dd      254
```



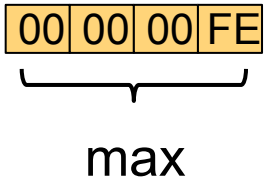
Data segment example

```
tmp         dd     -1
pixels      db     0FFh, 0FEh, 0FDh, 0FCh
i           dw     0
message     db     "H", "e", "llo", 0
buffer      times   8     db   0
max         dd     254
```

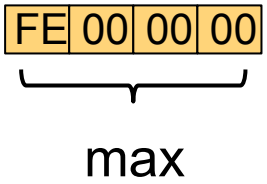


Endianness?

```
max             dd      254
```

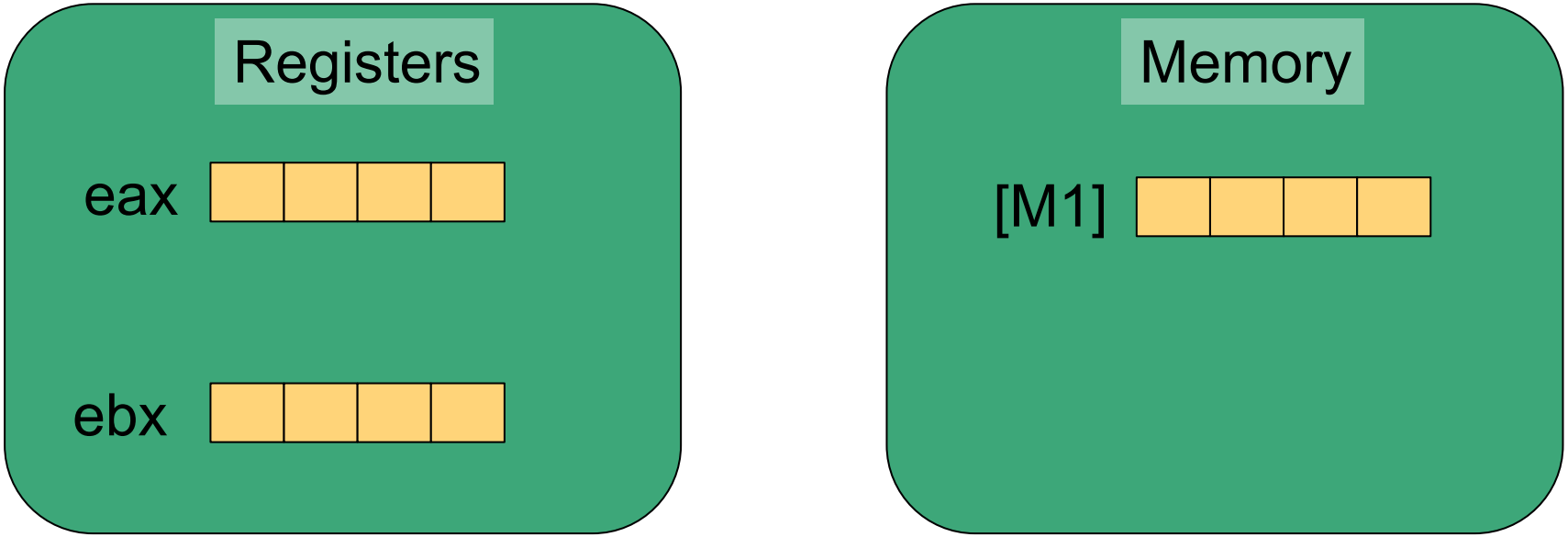


- In the previous slide we showed the above 4-byte memory content for a double-word that contains 254 = 000000FEh
- While this seems to make sense, it turns out that Intel processors do not do this!
  - Yes, the last 4 bytes shown in the previous slide are wrong
- The scheme shown above (i.e., bytes in memory follow the "natural" order): Big Endian
- Instead, Intel processors use Little Endian:



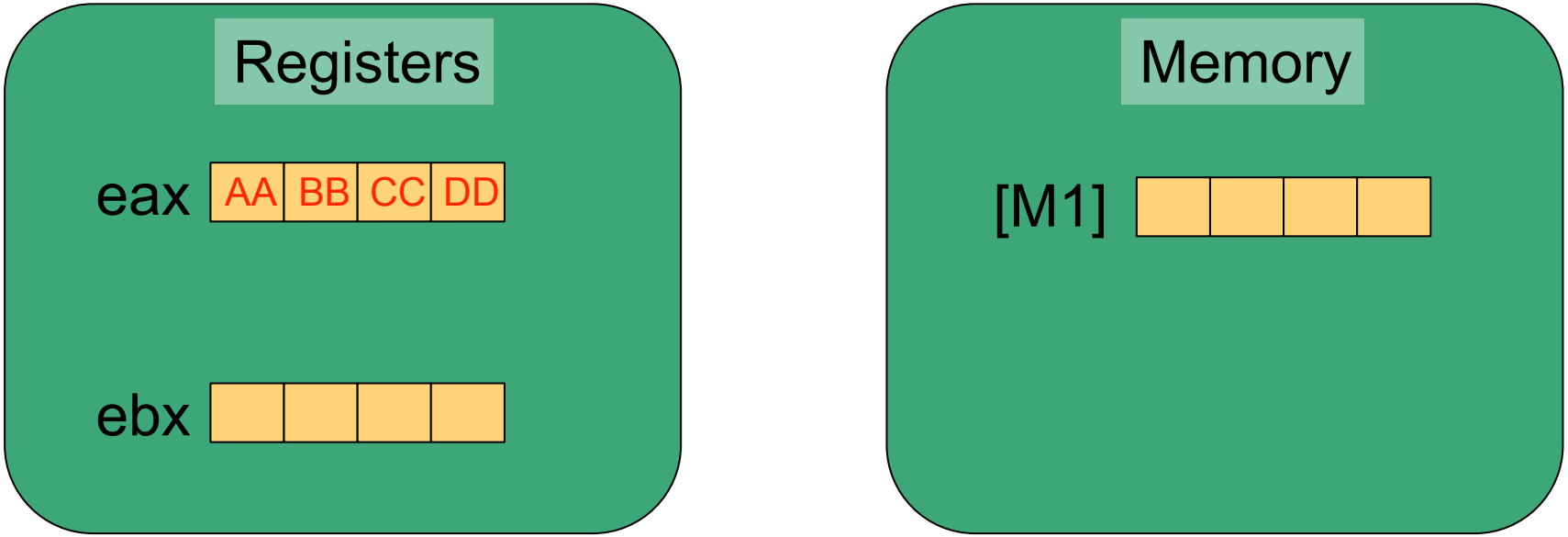
Little Endian

```
mov eax, 0AABBCCDDh
mov [M1], eax
mov ebx, [M1]
```



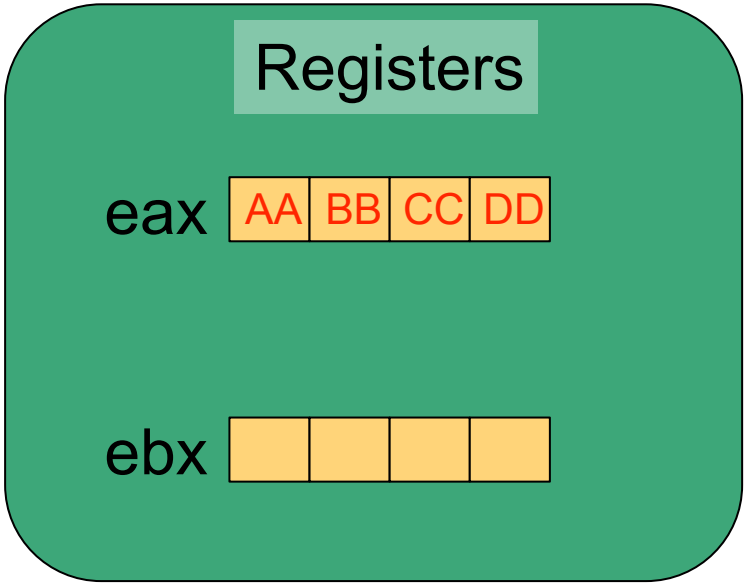
Little Endian

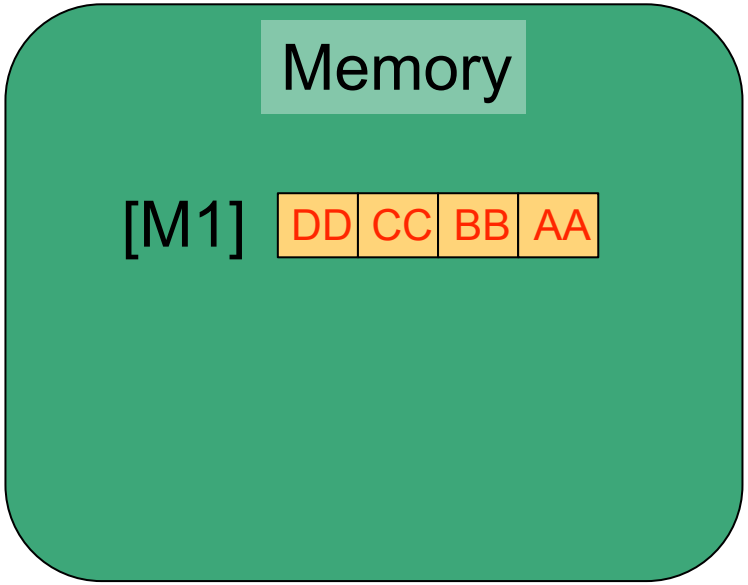
mov eax, 0AABBCCDDh
mov [M1], eax
mov ebx, [M1]



Little Endian

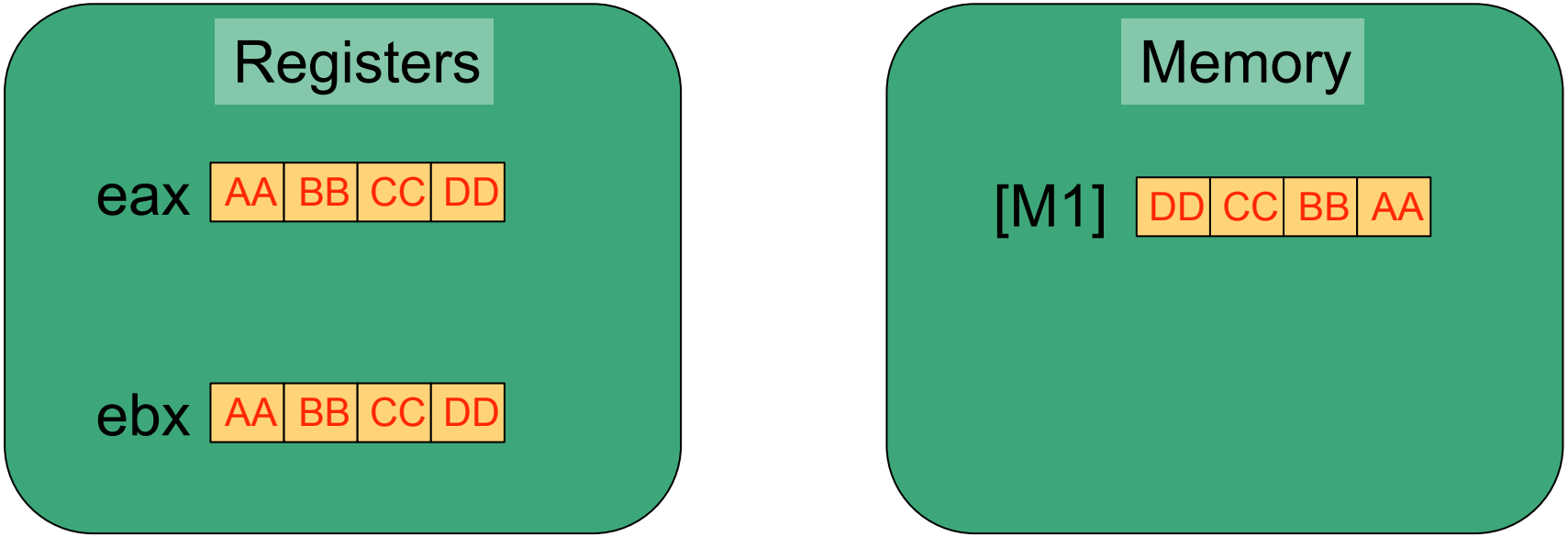
mov eax, 0AABBCCDDh
mov [M1], eax
mov ebx, [M1]





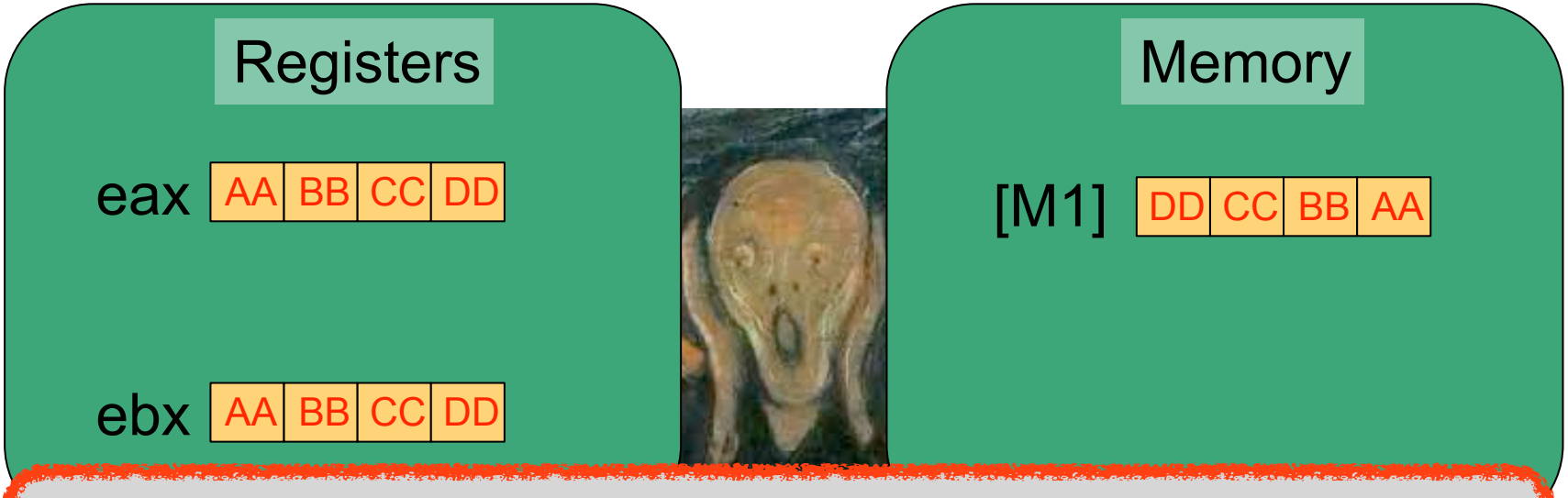
Little Endian

```
mov eax, 0AABBCCDDh
mov [M1], eax
mov ebx, [M1]
```



Little Endian

```
mov eax, 0AABBCCDDh
mov [M1], eax
mov ebx, [M1]
```



In-register byte order and in-memory byte order, within a single multi-byte value, are different!

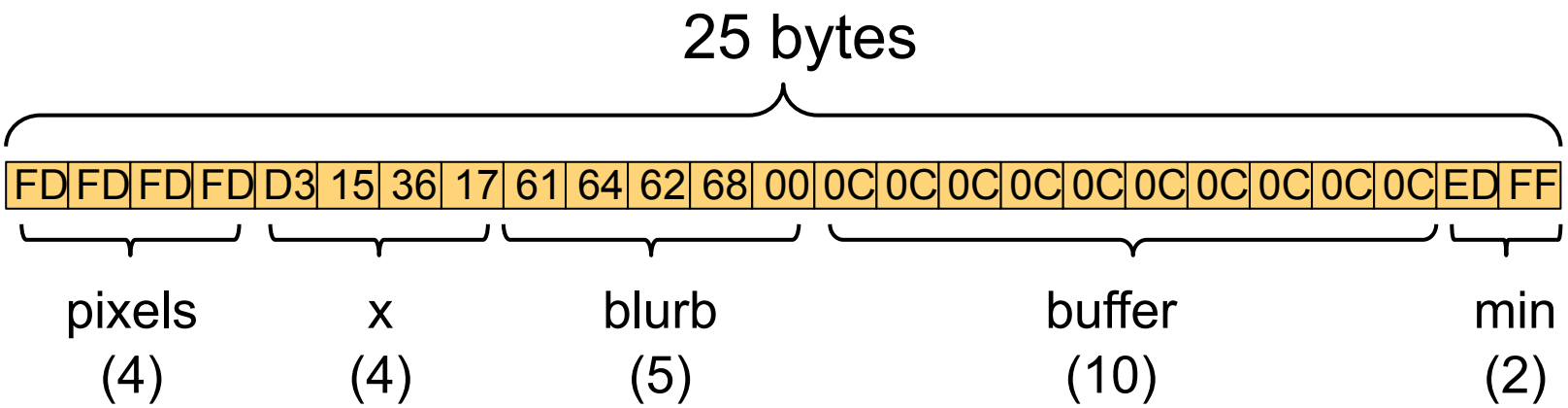
Example

```
pixels      times 4     db    0FDh
x           dd    0001011100110110000101011101 0011b
blurb       db    "ad", "b", "h", 0
buffer      times  10   db  14o
min         dw    -19
```

- What is the layout and the content of the data memory segment on a Little Endian machine?
  - Byte per byte, in hex

Example

```
pixels      times 4     db    0FDh
x           dd    00010111001101100001010111010011b
blurb       db    "ad", "b", "h", 0
buffer      times  10   db  14o
min         dw    -19
```



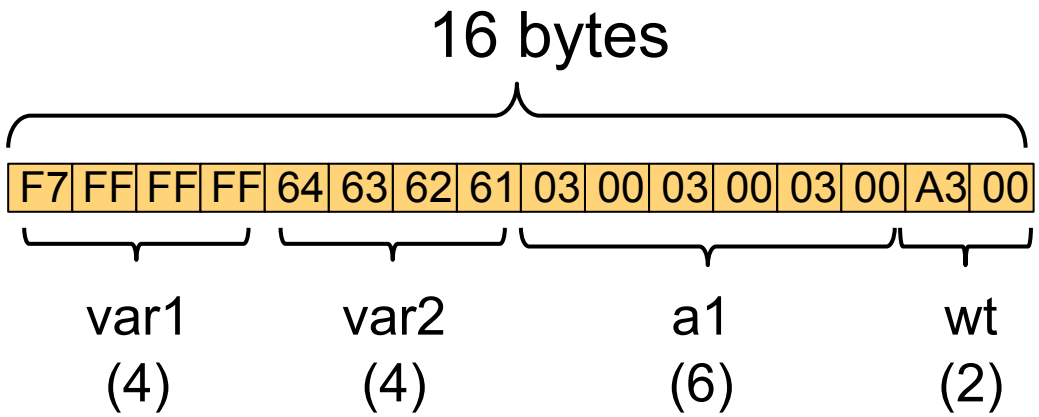
Practice #1

```
var1        dd     -9
var2        db     "dcba"
a1          times  3      dw     011b
wt          db     0A3h, 0
```

- What is the layout and the content of the data memory segment on a Little Endian machine?
  - Byte per byte, in hex

Practice #1

```
var1          dd     -9
var2          db     “dcba”
a1            times  3      dw     011b
wt            db     0A3h, 0
```



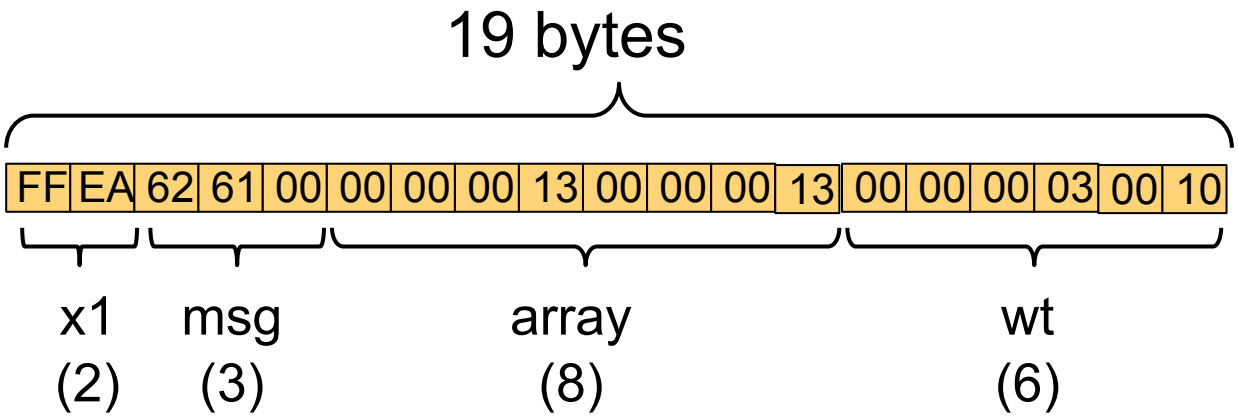
Practice #2

```
x1              dw      -22
msg             db      “ba”, 0
array           times 2       dd      023o
wt              dw      0,011b,020o
```

- What is the layout and the content of the data memory segment on a BIG ENDIAN machine?
  - Byte per byte, in hex

Practice #2

```
x1              dw      -22
msg             db      “ba”, 0
array           times 2       dd      023o
wt              dw      0,011b,020o
```



Example

- Data segment:

```
L1      db      0AAh, 0BBh
L2      dw      0CCDDh
L3      db      0EEh, 0FFh
```

- Program:

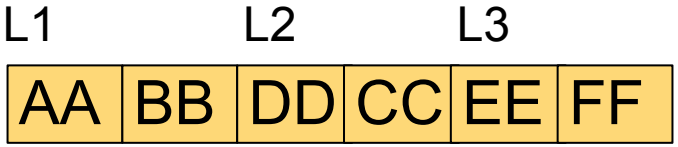
```
mov eax, [L2]
mov ax, [L3]
mov [L1], eax
```

- What's the memory content?

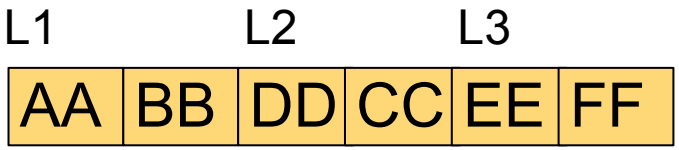
Solution

- Data segment:

```
L1      db      0AAh, 0BBh
L2      dw      0CCDDh
L3      db      0EEh, 0FFh
```



Solution




```
mov eax, [L2]     ; eax = FF EE CC DD

mov ax, [L3]      ; eax = FF EE FF EE

mov [L1], eax     ; L1 points to EE FF EE FF
```

Final memory content

Example

```
first       db    00h, 04Fh, 012h, 0A4h
second      dw    165
third       db    "adf"
```

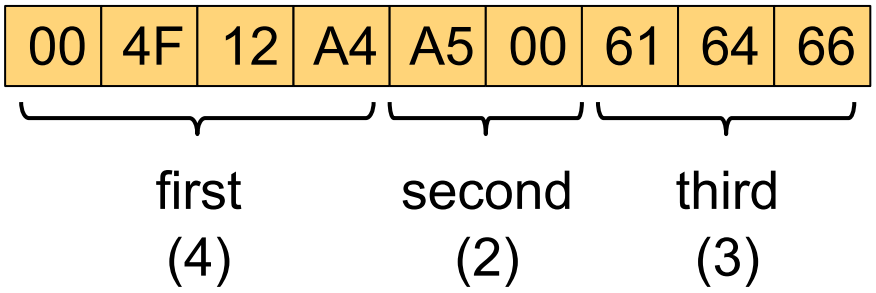
```
mov   eax, first
inc   eax
mov   ebx, [eax]
mov   [second], ebx
mov   byte [third], 11o
```

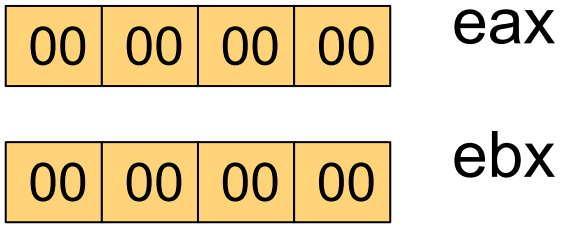
What is the content of "data" memory after the code executes on a Little Endian Machine?

Example

```
first       db    00h, 04Fh, 012h, 0A4h
second      dw    165
third       db    "adf"
```

```
mov   eax, first
inc   eax
mov   ebx, [eax]
mov   [second], ebx
mov   byte [third], 11o
```



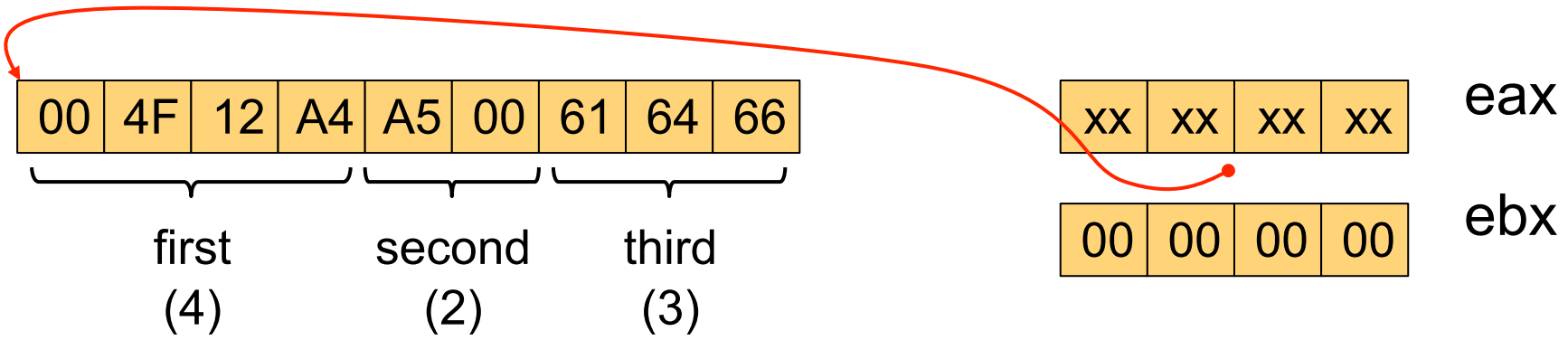


Example

```
first       db     00h, 04Fh, 012h, 0A4h
second      dw     165
third       db     "adf"
```

```
mov    eax, first
inc    eax
mov    ebx, [eax]
mov    [second], ebx
mov    byte [third], 11o
```

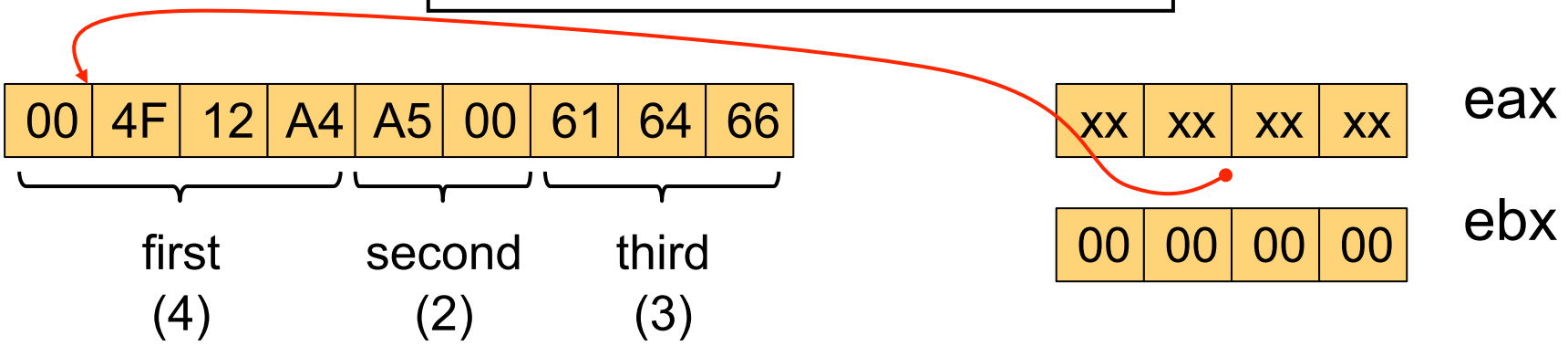
Put an **address** into eax
(addresses are 32-bit)



Example

```
first       db      00h, 04Fh, 012h, 0A4h
second      dw      165
third       db      "adf"
```

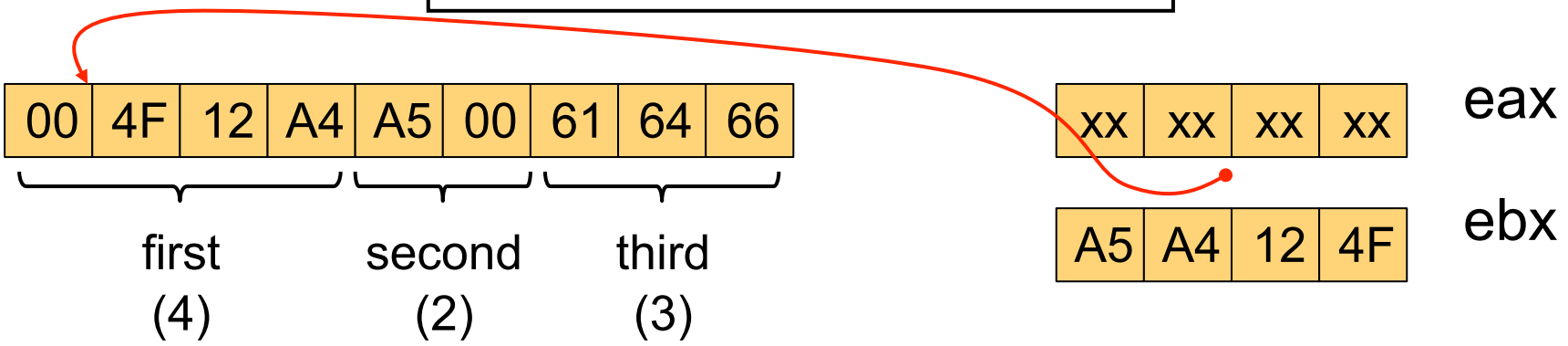
```
mov   eax, first
inc   eax
mov   ebx, [eax]
mov   [second], ebx
mov   byte [third], 11o
```



Example

```
first       db     00h, 04Fh, 012h, 0A4h
second      dw     165
third       db     "adf"
```

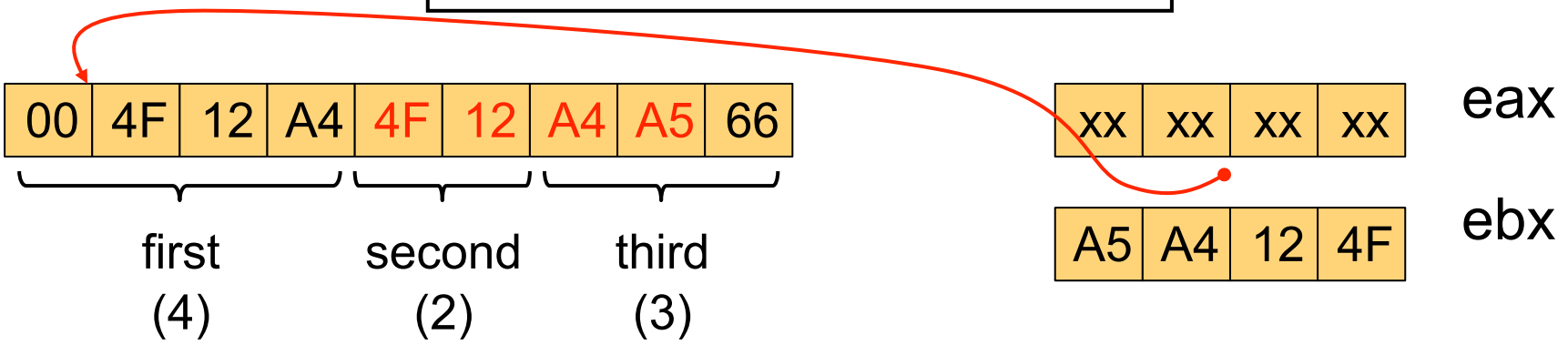
```
mov    eax, first
inc    eax
mov    ebx, [eax]
mov    [second], ebx
mov    byte [third], 11o
```



Example

```
first       db     00h, 04Fh, 012h, 0A4h
second      dw     165
third       db     "adf"
```

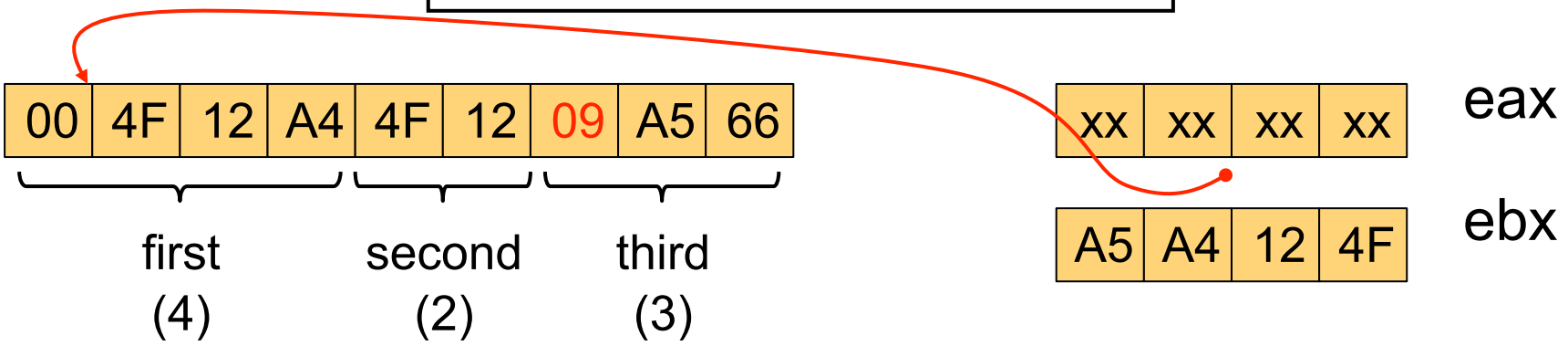
```
mov    eax, first
inc    eax
mov    ebx, [eax]
mov    [second], ebx
mov    byte [third], 11o
```



Example

```
first       db     00h, 04Fh, 012h, 0A4h
second      dw     165
third       db     "adf"
```

```
mov    eax, first
inc    eax
mov    ebx, [eax]
mov    [second], ebx
mov    byte [third], 11o
```



Practice #3

- Consider the following program

```
var1            dd      179
var2            db      0A3h, 017h, 012h
var3            db      "bca"
```

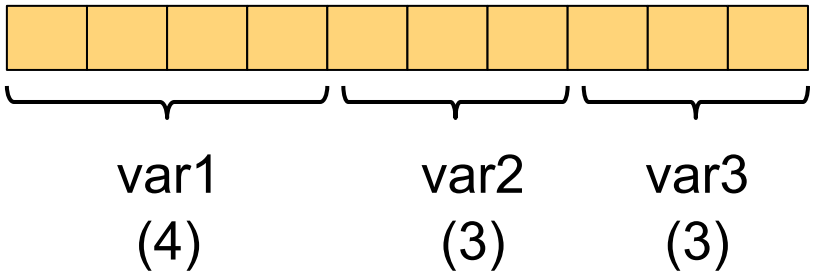
```
mov     eax, var1
add     eax, 3
mov     ebx, [eax]
add     ebx, 5
mov     [var1], ebx
```

- What is the layout of memory starting at address var1 on a Little Endian Machine?

Practice #3

```
var1        dd      179
var2        db      0A3h, 017h, 012h
var3        db      "bca"
```

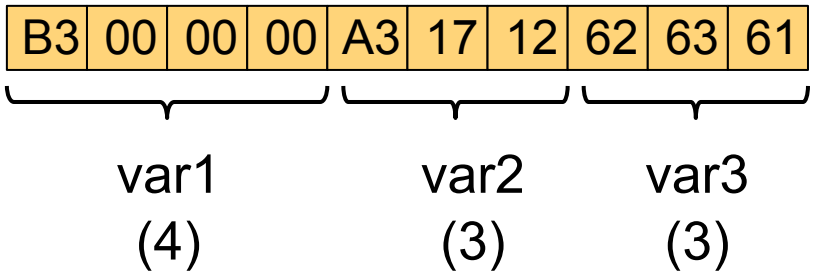
```
mov     eax, var1
add     eax, 3
mov     ebx, [eax]
add     ebx, 5
mov     [var1], ebx
```



Practice #3

```
var1            dd      179
var2            db      0A3h, 017h, 012h
var3            db      "bca"
```

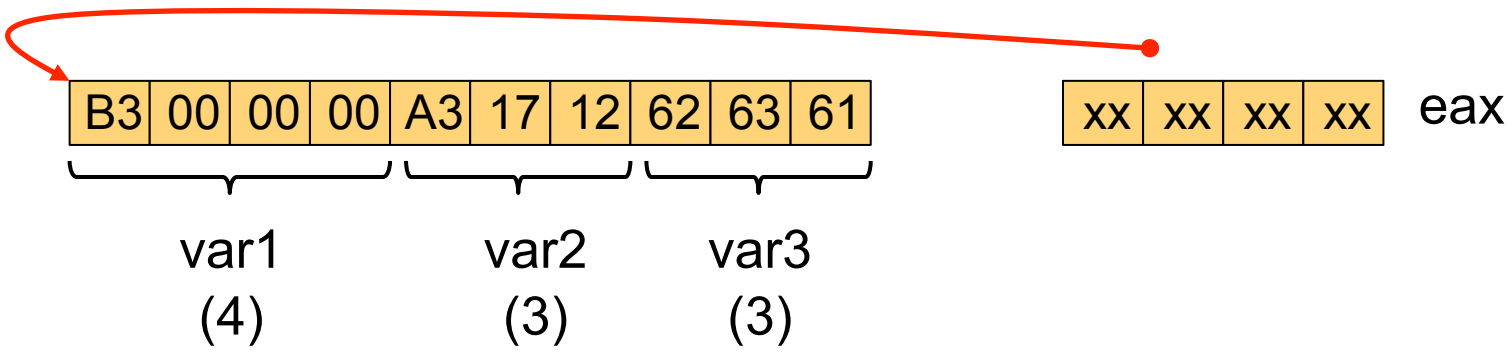
```
mov     eax, var1
add     eax, 3
mov     ebx, [eax]
add     ebx, 5
mov     [var1], ebx
```



Practice #3

```
var1        dd      179
var2        db      0A3h, 017h, 012h
var3        db      "bca"
```

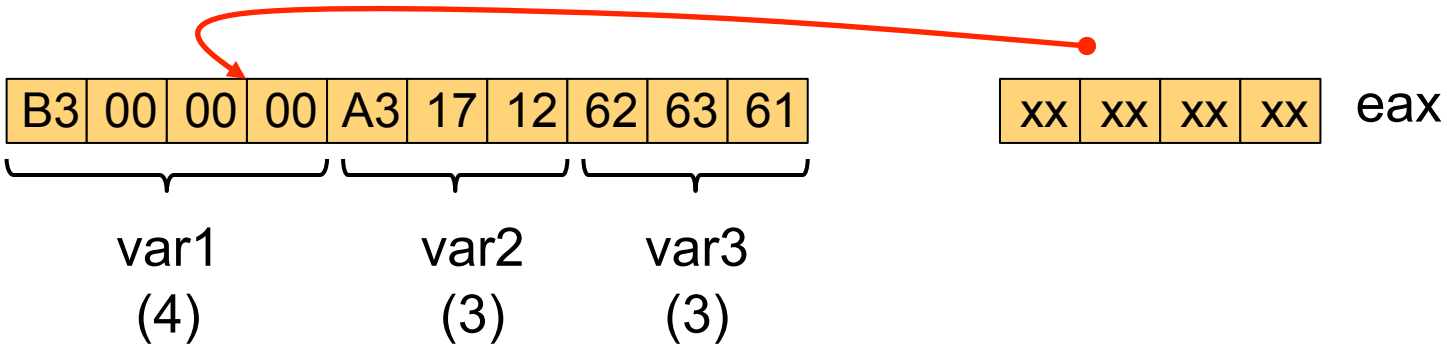
```
mov     eax, var1
add     eax, 3
mov     ebx, [eax]
add     ebx, 5
mov     [var1], ebx
```



Practice #3

```
var1            dd      179
var2            db      0A3h, 017h, 012h
var3            db      "bca"
```

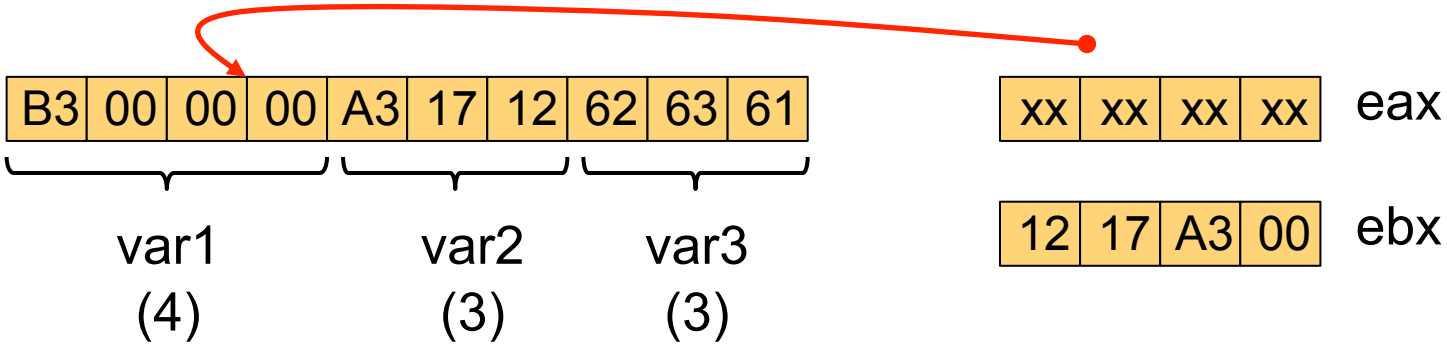
```
mov     eax, var1
add     eax, 3
mov     ebx, [eax]
add     ebx, 5
mov     [var1], ebx
```



Practice #3

```
var1            dd      179
var2            db      0A3h, 017h, 012h
var3            db      "bca"
```

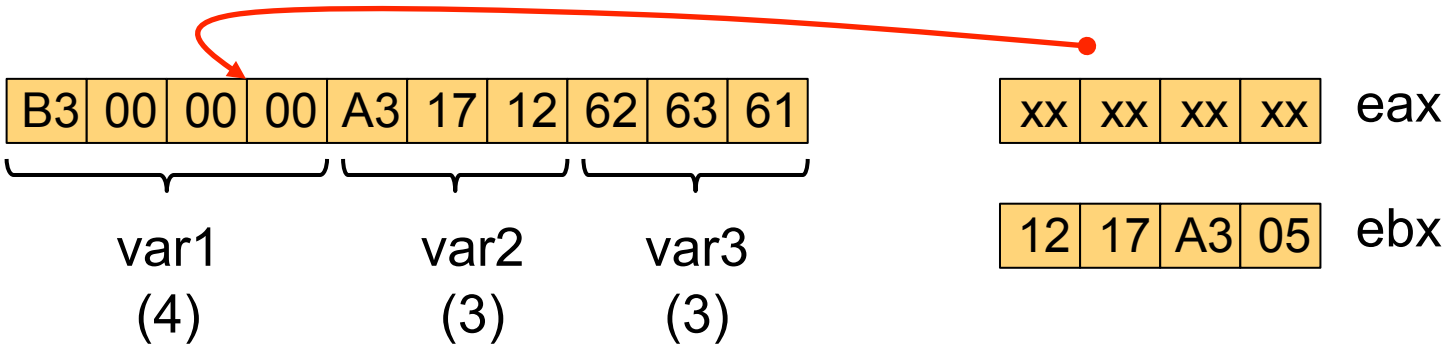
```
mov     eax, var1
add     eax, 3
mov     ebx, [eax]
add     ebx, 5
mov     [var1], ebx
```



Practice #3

```
var1        dd      179
var2        db      0A3h, 017h, 012h
var3        db      "bca"
```

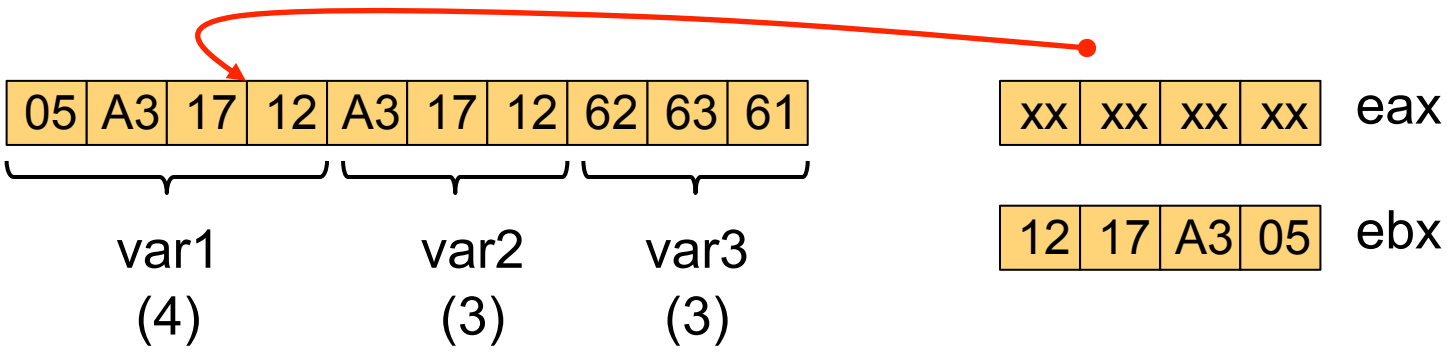
```
mov     eax, var1
add     eax, 3
mov     ebx, [eax]
add     ebx, 5
mov     [var1], ebx
```



Practice #3

```
var1        dd      179
var2        db      0A3h, 017h, 012h
var3        db      "bca"
```

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



Practice #4

- Consider the following program

```
var1            db      "b","ca",0
var2            db 1,2,3,4
var3            times 2 dw    012h
```

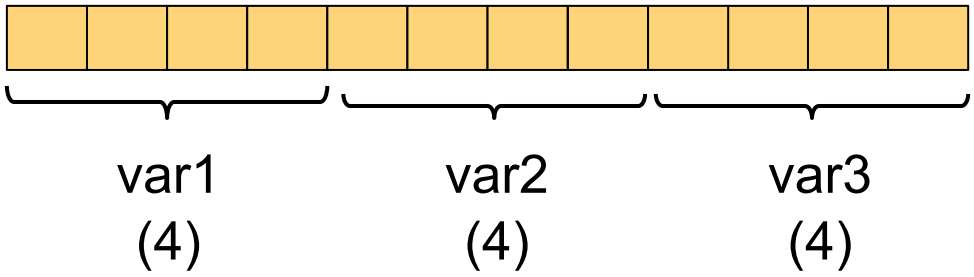
```
mov     eax, var3
mov     ebx, var1
sub     eax, 4
add     ebx, [eax]
mov     dword [ebx], 42
```

- What is the layout of memory starting at address var1 on a Little Endian Machine?

Practice #4

```
var1            db      "b","ca",0
var2            times   db 3,0,0,0
var3            times 2 dw    012h
```

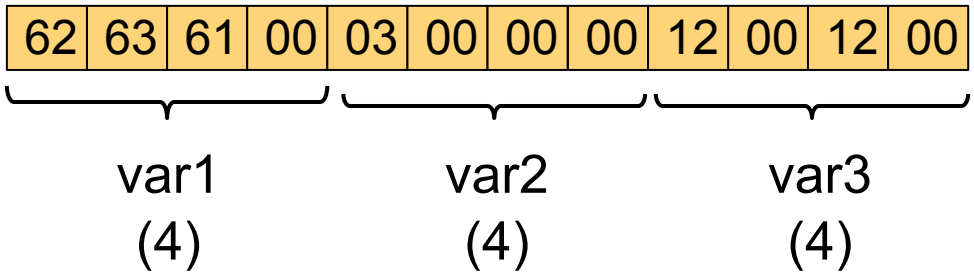
```
mov     eax, var3
mov     ebx, var1
sub     eax, 4
add     ebx, [eax]
mov     dword [ebx], 42
```



Practice #4

```
var1            db      "b","ca",0
var2            times   db 3,0,0,0
var3            times 2 dw    012h
```

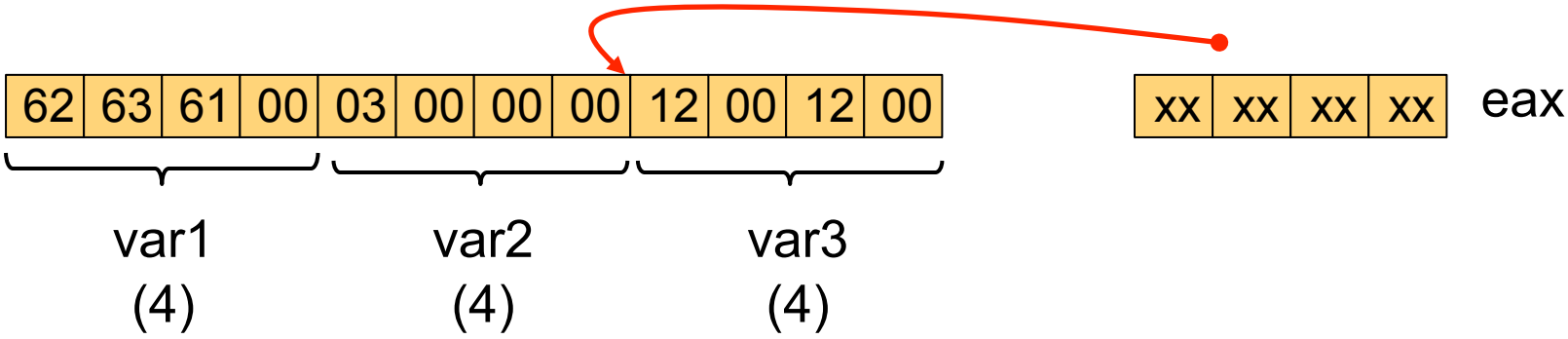
```
mov     eax, var3
mov     ebx, var1
sub     eax, 4
add     ebx, [eax]
mov     dword [ebx], 42
```



Practice #4

```
var1            db      "b","ca",0
var2            times   db 3,0,0,0
var3            times 2 dw    012h
```

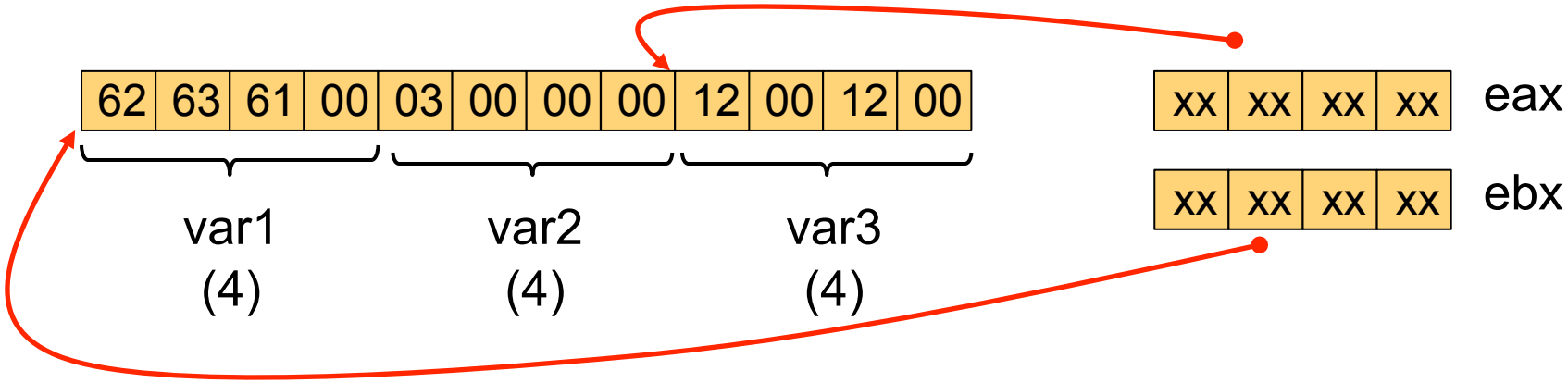
```
mov     eax, var3
mov     ebx, var1
sub     eax, 4
add     ebx, [eax]
mov     dword [ebx], 42
```



Practice #4

```
var1            db      "b","ca",0
var2            times   db 3,0,0,0
var3            times 2 dw    012h
```

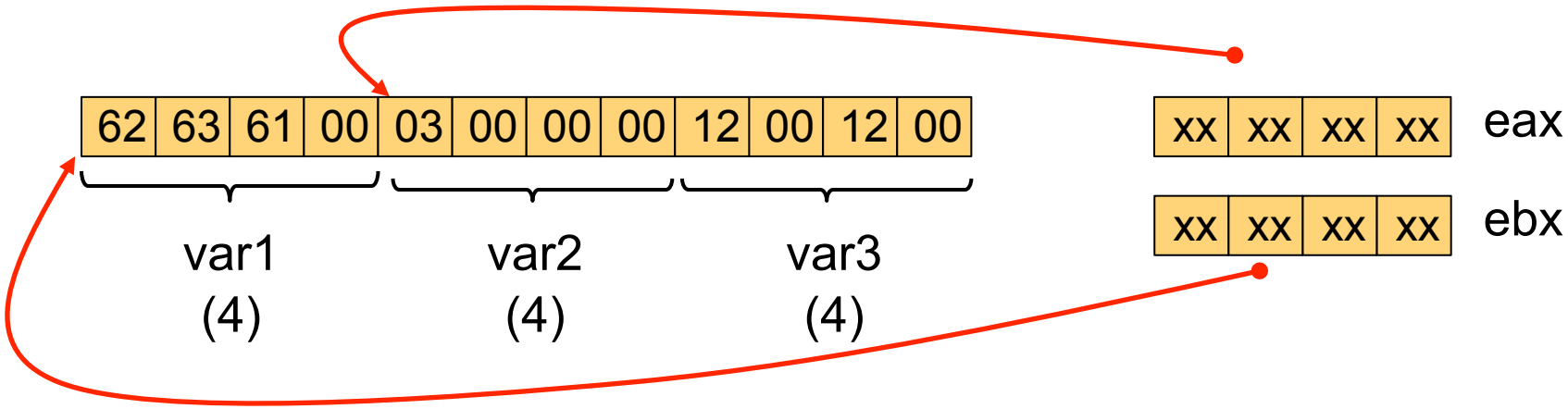
```
mov     eax, var3
mov     ebx, var1
sub     eax, 4
add     ebx, [eax]
mov     dword [ebx], 42
```



Practice #4

```
var1            db      "b","ca",0
var2            times   db 3,0,0,0
var3            times 2 dw    012h
```

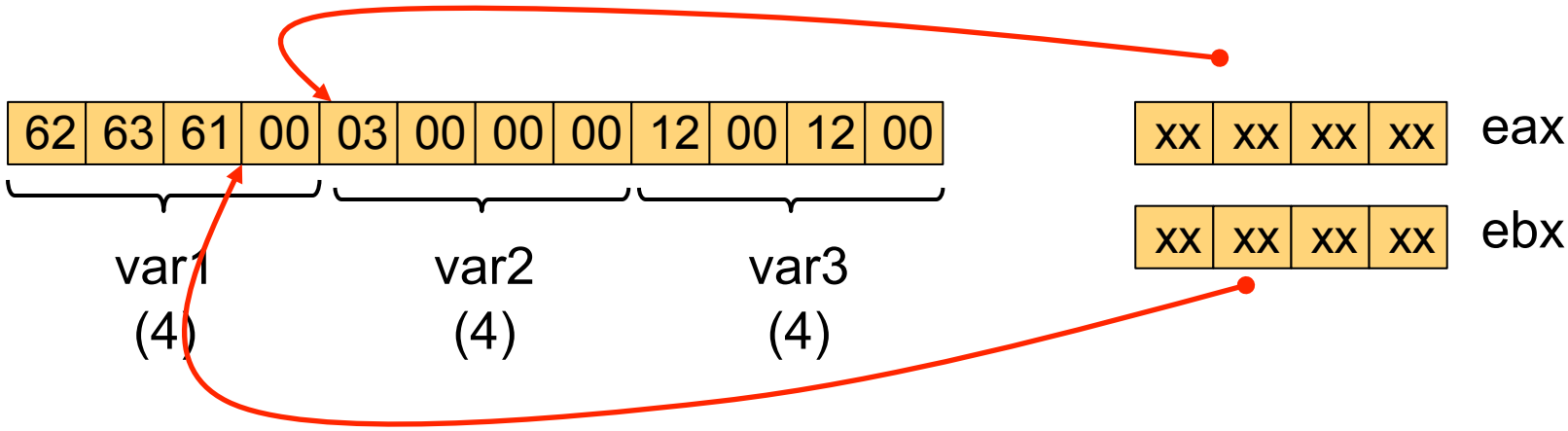
```
mov     eax, var3
mov     ebx, var1
sub     eax, 4
add     ebx, [eax]
mov     dword [ebx], 42
```



Practice #4

```
var1            db      "b","ca",0
var2            times   db 3,0,0,0
var3            times 2 dw    012h
```

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



Practice #4

```
var1            db      "b","ca",0
var2            times   db 3,0,0,0
var3            times 2 dw    012h
```

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