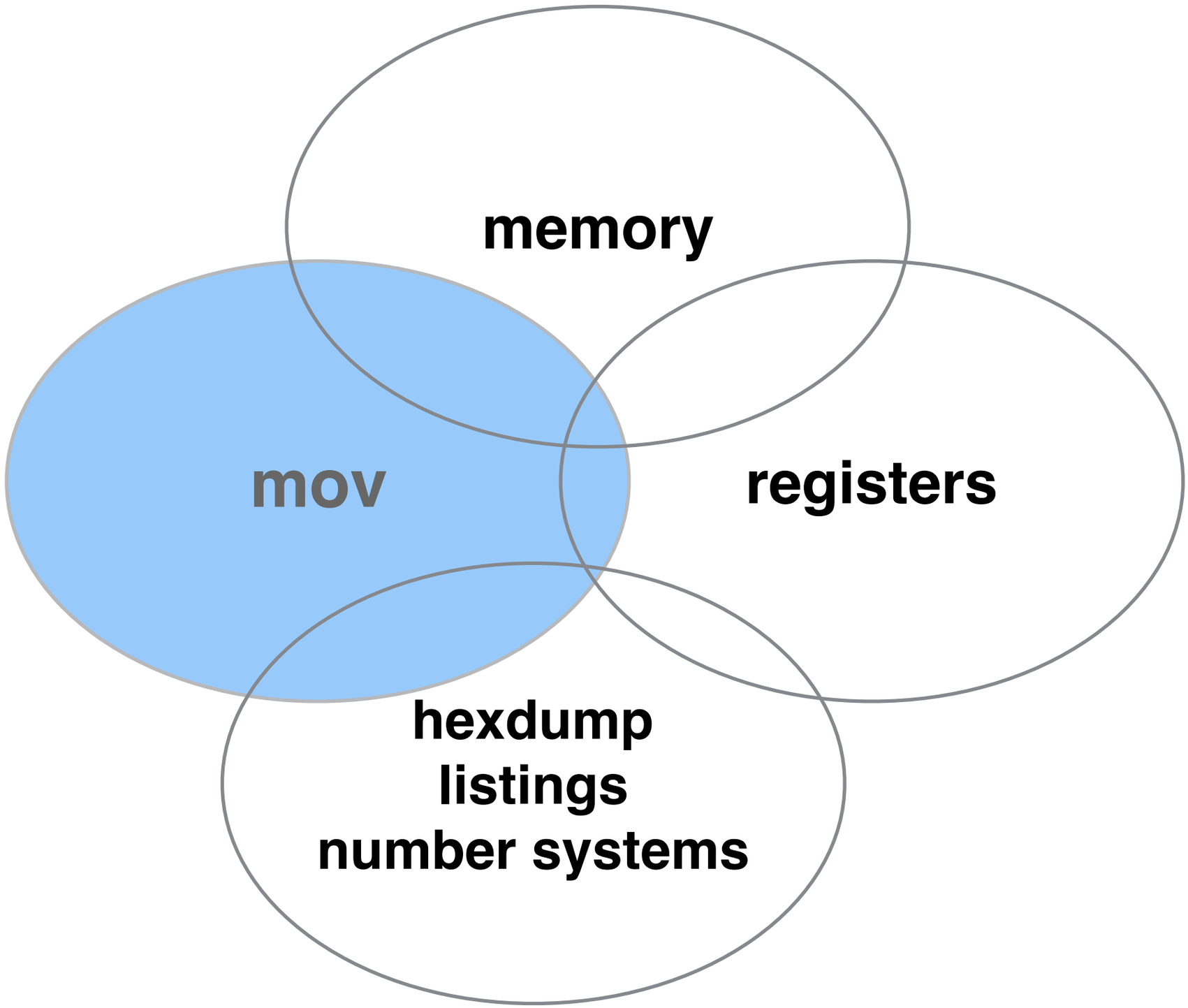
Smith College
Computer Science

CSC231 - Assembly

Week #3

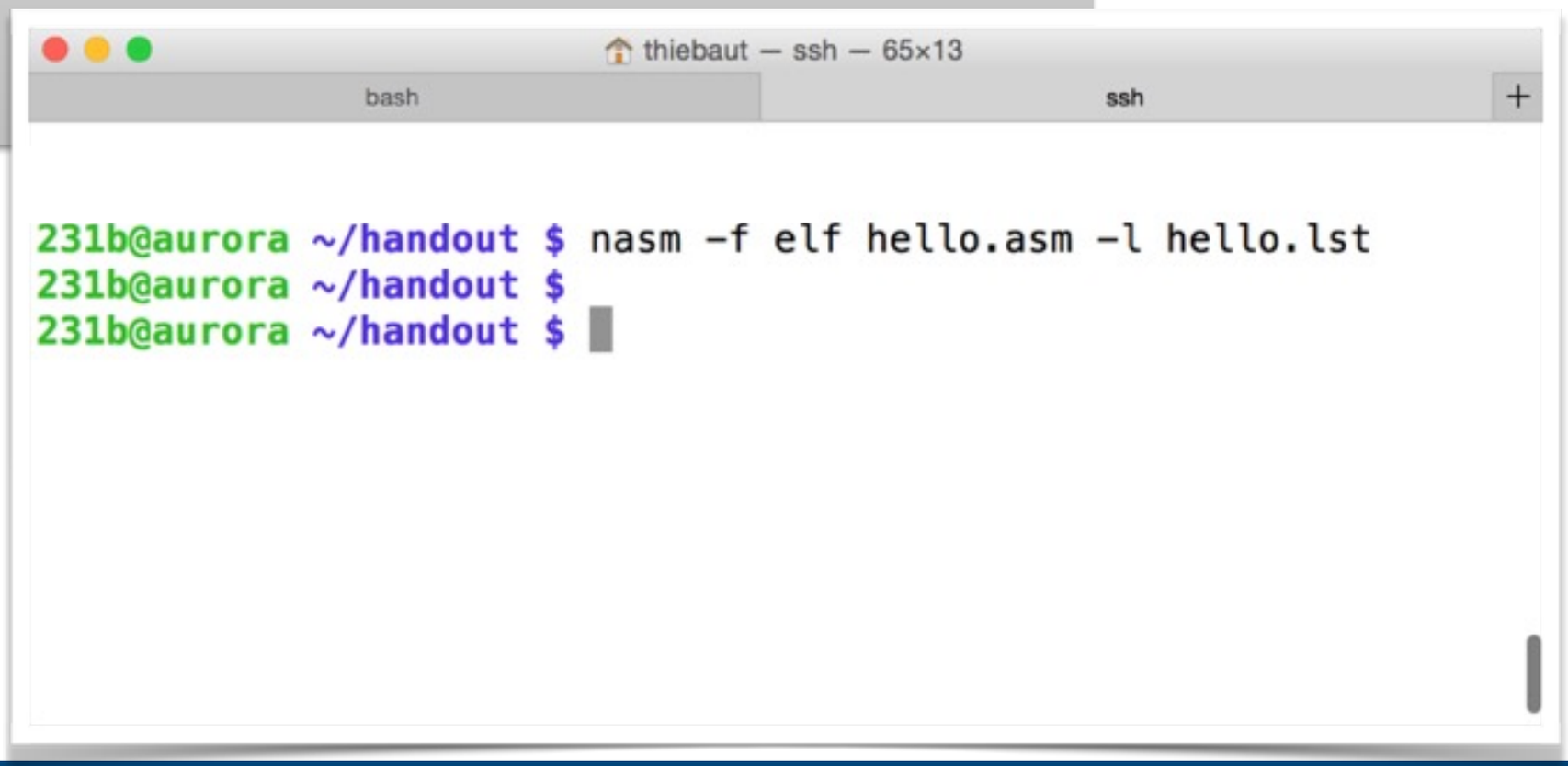
Dominique Thiébaut
dthiebaut@smith.edu


memory
mov
registers
hexdump
listings
number systems

Let's Finish Last Week's Material…

Listing

```
                section .data
Hello           db      "Hello there!", 10, 10
HelloLen        equ     $-Hello

                section .text
                global  _start
_start:

;;; print message
                mov     eax, 4          ; write
                mov     ebx, 1          ; stdout
                mov     ecx, Hello      ; address of message to print
                mov     edx, HelloLen   ; # of chars to print
                int     0x80

;;; exit
                mov     ebx, 0
                mov     eax, 1
                int     0x80
```



Listing

```
11                                      section .data
12 00000000 48656C6C6F20746865-     Hello           db      "Hello there!", 10, 10
13 00000009 7265210A0A
14                                  HelloLen        equ     $-Hello
15
16                                      section .text
17                                      global  _start
18                                  _start:
19
20                                  ;;; print message
21 00000000 B804000000                                  mov     eax, 4          ; write
22 00000005 BB01000000                                  mov     ebx, 1          ; stdout
23 0000000A B9[00000000]                                mov     ecx, Hello      ;
24 0000000F BA0E000000                                  mov     edx, HelloLen   ;
25 00000014 CD80                                        int     0x80
26
27                                  ;;; exit
28 00000016 BB00000000                                  mov     ebx, 0
29 0000001B B801000000                                  mov     eax, 1
30 00000020 CD80                                        int     0x80
```

Listing

```
11                                      section .data
12 00000000 48656C6C6F20746865-     Hello           db      "Hello there!", 10, 10
13 00000009 7265210A0A
14                                  HelloLen        equ     $-Hello
15
16                                      section .text
17                                      global  _start
18                                  _start:
19
20                                  ;;; print message
21 00000000 B804000000                                  mov     eax, 4          ; write
22 00000005 BB01000000                                  mov     ebx, 1          ; stdout
23 0000000A B9[00000000]                                mov     ecx, Hello      ;
24 0000000F BA0E000000                                  mov     edx, HelloLen   ;
25 00000014 CD80                                        int     0x80
26
27                                  ;;; exit
28 00000016 BB00000000                                  mov     ebx, 0
29 0000001B B801000000                                  mov     eax, 1
30 00000020 CD80                                        int     0x80
```

Hexdump

thiebaut — ssh — 89×29

bash | ssh

```
231b@aurora ~/handout $ hexdump -v -C hello
00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  02 00 03 00 01 00 00 00  80 80 04 08 34 00 00 00  |............4...|
00000020  dc 00 00 00 00 00 00 00  34 00 20 00 02 00 28 00  |........4. ...(.|
00000030  06 00 03 00 01 00 00 00  00 00 00 00 00 80 04 08  |................|
00000040  00 80 04 08 a2 00 00 00  a2 00 00 00 05 00 00 00  |................|
00000050  00 10 00 00 01 00 00 00  a4 00 00 00 a4 90 04 08  |................|
00000060  a4 90 04 08 0e 00 00 00  0e 00 00 00 06 00 00 00  |................|
00000070  00 10 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  b8 04 00 00 00 bb 01 00  00 00 b9 a4 90 04 08 ba  |................|
00000090  0e 00 00 00 cd 80 bb 00  00 00 00 b8 01 00 00 00  |................|
000000a0  cd 80 00 00 48 65 6c 6c  6f 20 74 68 65 72 65 21  |....Hello there!|
000000b0  0a 0a 00 2e 73 79 6d 74  61 62 00 2e 73 74 72 74  |....symtab..strt|
000000c0  61 62 00 2e 73 68 73 74  72 74 61 62 00 2e 74 65  |ab..shstrtab..te|
000000d0  78 74 00 2e 64 61 74 61  00 00 00 00 00 00 00 00  |xt..data........|
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000100  00 00 00 00 1b 00 00 00  01 00 00 00 06 00 00 00  |................|
00000110  80 80 04 08 80 00 00 00  22 00 00 00 00 00 00 00  |........".......|
00000120  00 00 00 00 10 00 00 00  00 00 00 00 21 00 00 00  |............!...|
00000130  01 00 00 00 03 00 00 00  a4 90 04 08 a4 00 00 00  |................|
00000140  0e 00 00 00 00 00 00 00  00 00 00 00 04 00 00 00  |................|
00000150  00 00 00 00 11 00 00 00  03 00 00 00 00 00 00 00  |................|
00000160  00 00 00 00 b2 00 00 00  27 00 00 00 00 00 00 00  |........'.......|
00000170  00 00 00 00 01 00 00 00  00 00 00 00 01 00 00 00  |................|
00000180  02 00 00 00 00 00 00 00  00 00 00 00 cc 01 00 00  |................|
00000190  b0 00 00 00 05 00 00 00  07 00 00 00 04 00 00 00  |................|
000001a0  10 00 00 00 09 00 00 00  03 00 00 00 00 00 00 00  |................|
000001b0  00 00 00 00 7c 02 00 00  39 00 00 00 00 00 00 00  |....|...9.......|
```

```
 11                                     section .data
 12 00000000 48656C6C6F20746865-    Hello           db      "Hello there!", 10, 10
 13 00000009 7265210A0A
 14                                 HelloLen        equ     $-Hello
 15
 16                                     section .text
 17                                     global  _start
 18                                 _start:
 19
 20                                 ;;; print message
 21 00000000 B804000000                 mov     eax, 4          ; write
 22 00000005 BB01000000                 mov     ebx, 1          ; stdout
 23 0000000A B9[00000000]               mov     ecx, Hello      ;
 24 0000000F BA0E000000                 mov     edx, HelloLen   ;
 25 00000014 CD80                       int     0x80
 26
 27                                 ;;; exit
 28 00000016 BB00000000                 mov     ebx, 0
 29 0000001B B801000000                 mov     eax, 1
 30 00000020 CD80                       int     0x80
```

ssh

```
                                                                  0 00  |.ELF............|
                                                                  0 00  |............4...|
00000020  dc 00 00 00 00 00 00 00  34 00 20 00 02 00 28 00  |........4. ...(.|
00000030  06 00 03 00 01 00 00 00  00 00 00 00 00 80 04 08  |................|
00000040  00 80 04 08 a2 00 00 00  a2 00 00 00 05 00 00 00  |................|
00000050  00 10 00 00 01 00 00 00  a4 00 00 00 a4 90 04 08  |................|
00000060  a4 90 04 08 0e 00 00 00  0e 00 00 00 06 00 00 00  |................|
00000070  00 10 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  b8 04 00 00 00 bb 01 00  00 00 b9 a4 90 04 08 ba  |................|
00000090  0e 00 00 00 cd 80 bb 00  00 00 00 b8 01 00 00 00  |................|
000000a0  cd 80 00 00 48 65 6c 6c  6f 20 74 68 65 72 65 21  |....Hello there!|
000000b0  0a 0a 00 2e 73 79 6d 74  61 62 00 2e 73 74 72 74  |....symtab..strt|
000000c0  61 62 00 2e 73 68 73 74  72 74 61 62 00 2e 74 65  |ab..shstrtab..te|
000000d0  78 74 00 2e 64 61 74 61  00 00 00 00 00 00 00 00  |xt..data........|
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000100  00 00 00 00 1b 00 00 00  01 00 00 00 06 00 00 00  |................|
00000110  80 80 04 08 80 00 00 00  22 00 00 00 00 00 00 00  |........".......|
00000120  00 00 00 00 10 00 00 00  00 00 00 00 21 00 00 00  |............!...|
00000130  01 00 00 00 03 00 00 00  a4 90 04 08 a4 00 00 00  |................|
00000140  0e 00 00 00 00 00 00 00  00 00 00 00 04 00 00 00  |................|
00000150  00 00 00 00 11 00 00 00  03 00 00 00 00 00 00 00  |................|
00000160  00 00 00 00 b2 00 00 00  27 00 00 00 00 00 00 00  |........'.......|
00000170  00 00 00 00 01 00 00 00  00 00 00 00 01 00 00 00  |................|
00000180  02 00 00 00 00 00 00 00  00 00 00 00 cc 01 00 00  |................|
00000190  b0 00 00 00 05 00 00 00  07 00 00 00 04 00 00 00  |................|
000001a0  10 00 00 00 09 00 00 00  03 00 00 00 00 00 00 00  |................|
000001b0  00 00 00 00 7c 02 00 00  39 00 00 00 00 00 00 00  |....|...9.......|
```

Our Goal for This Week

```
int x, y, sum;

x = 3;
y = 5;
sum = x + y;
```

Plan

- Mov instruction
- Registers
- Memory storage options

You already know some of this material…

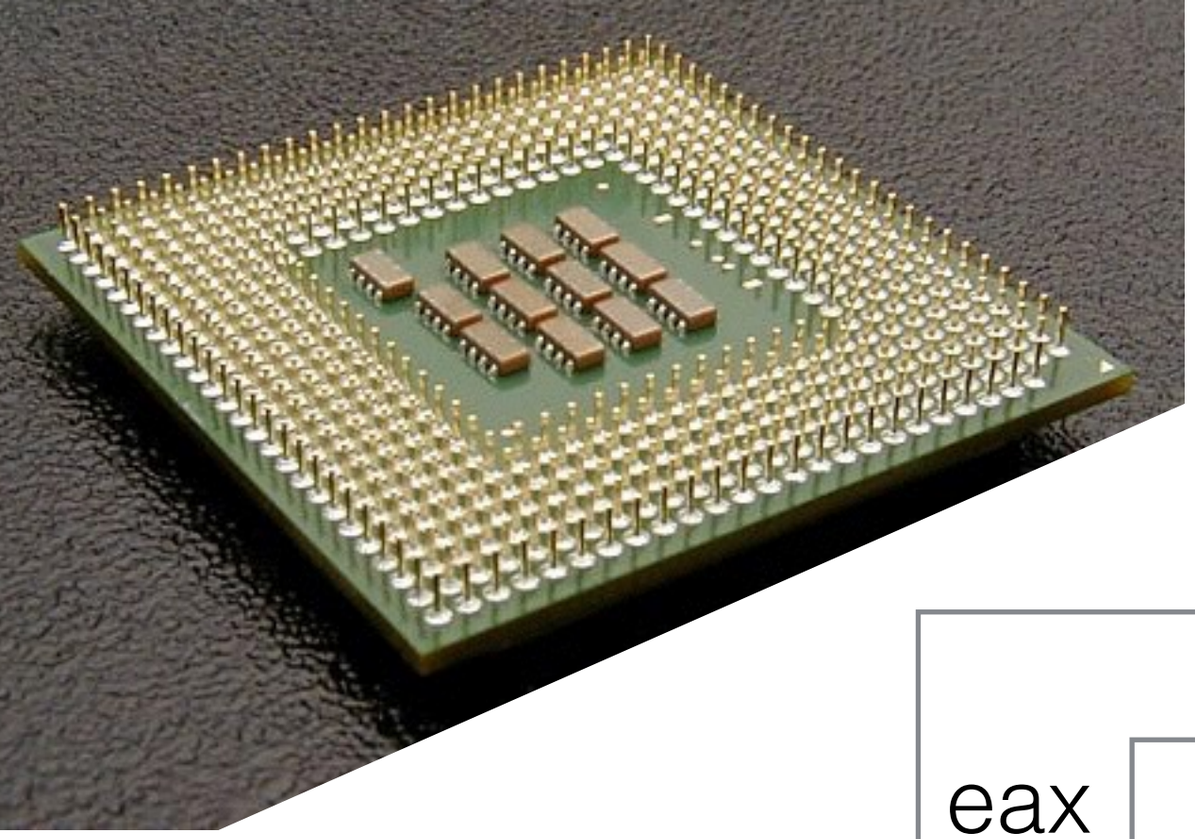
- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

The **mov** instruction

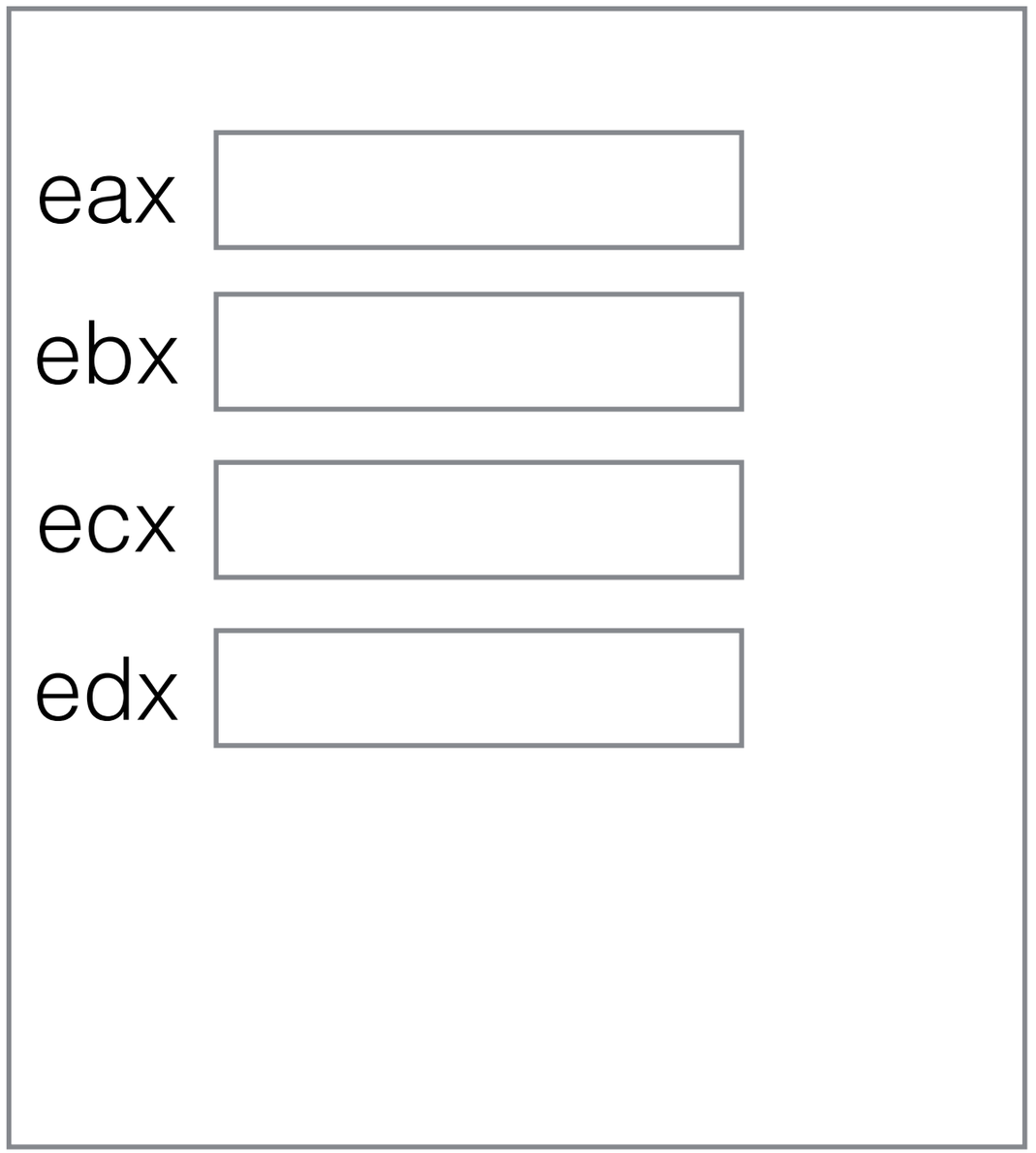
```
mov  dest, source
```

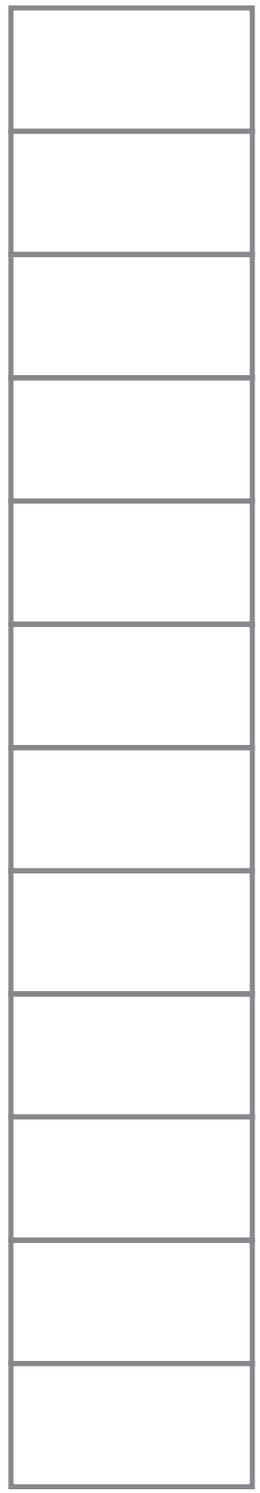
Operands

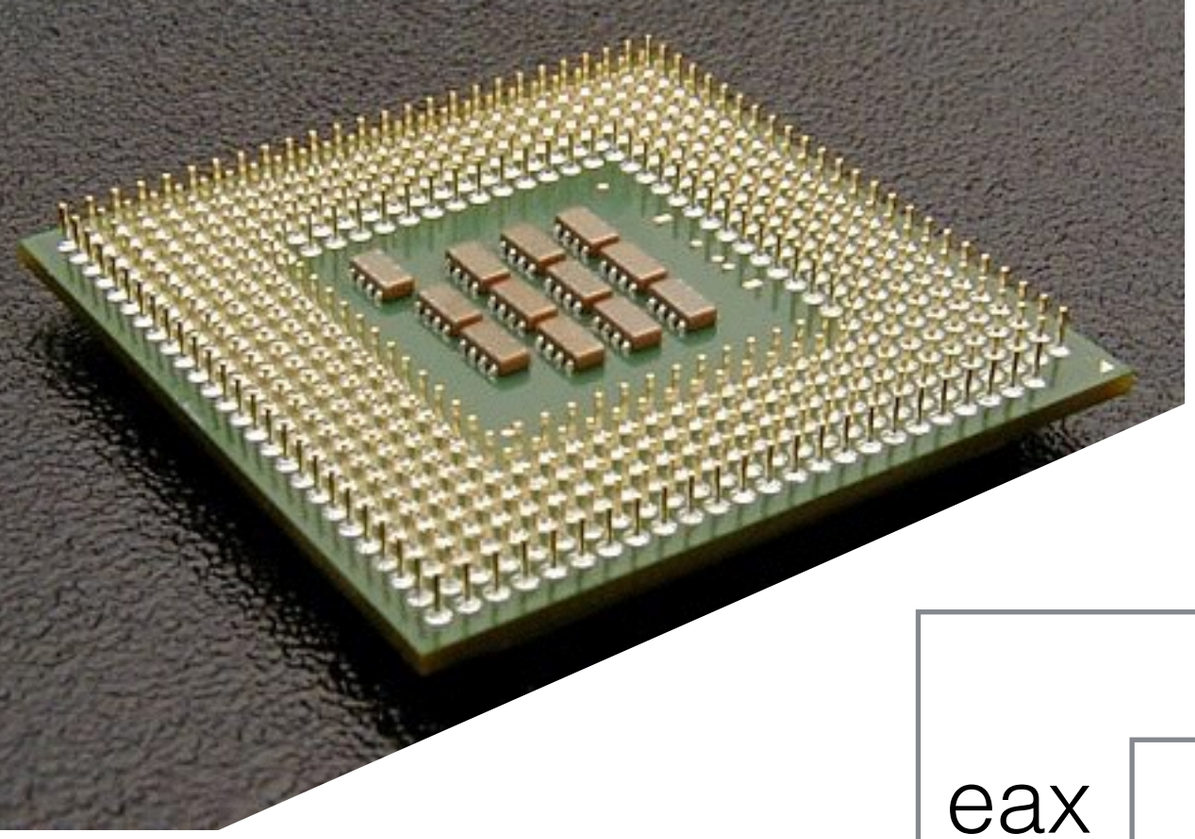
- **mov** reg, reg
- **mov** reg, mem
- **mov** mem, reg
- **mov** reg, imm
- **mov** mem, imm



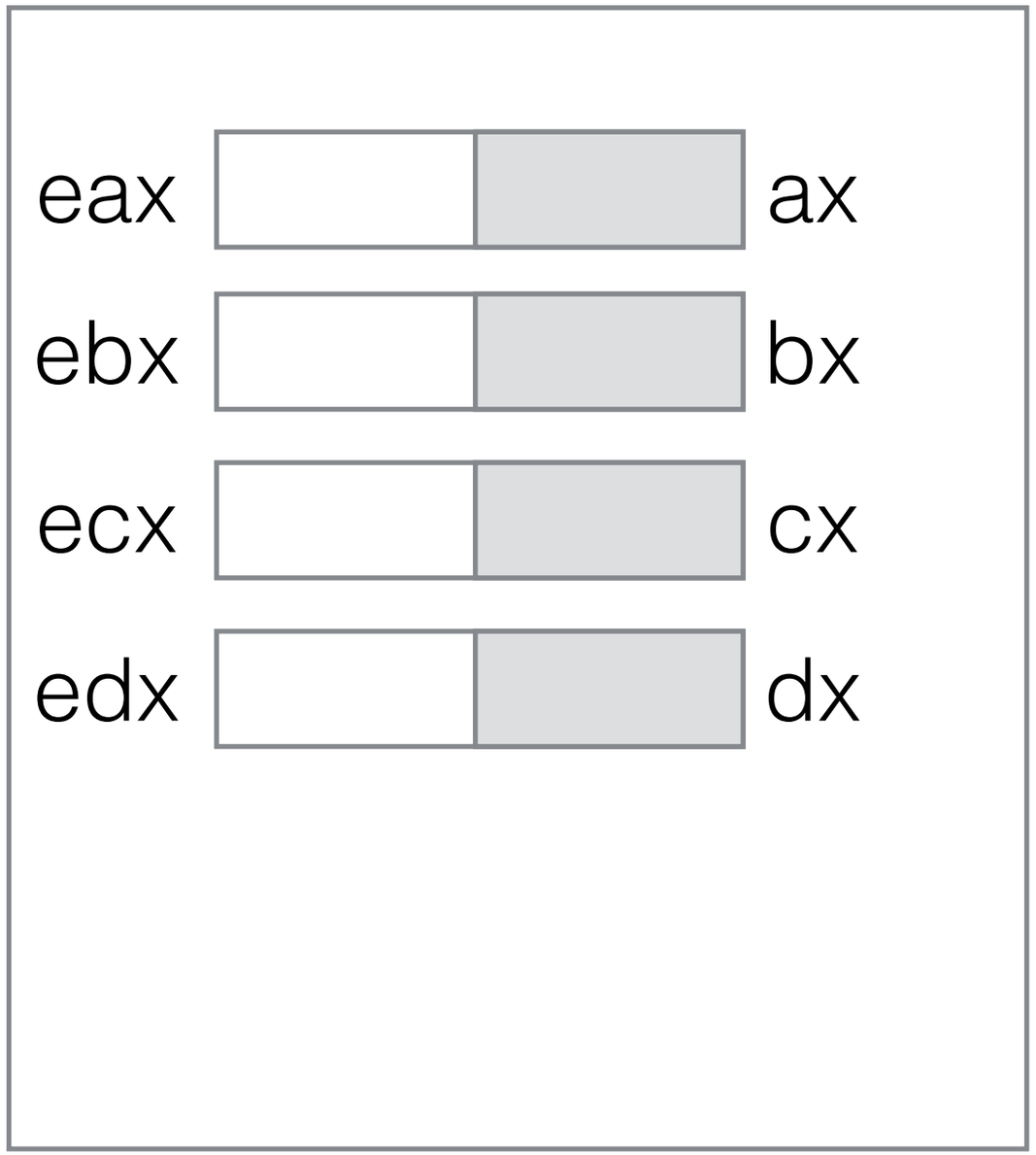
Pentium Registers

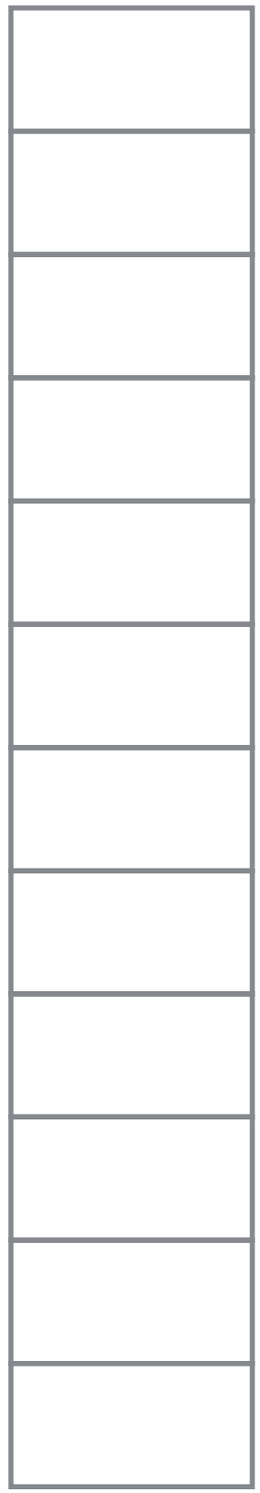


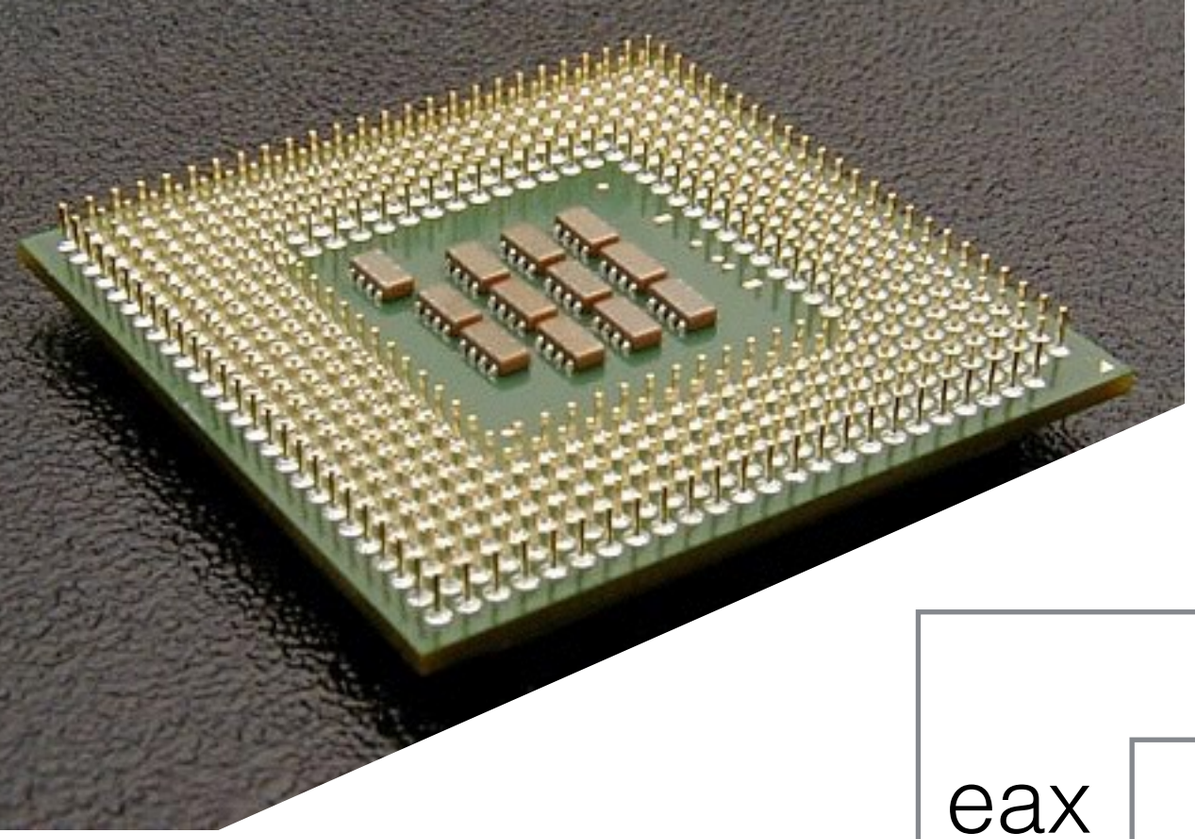




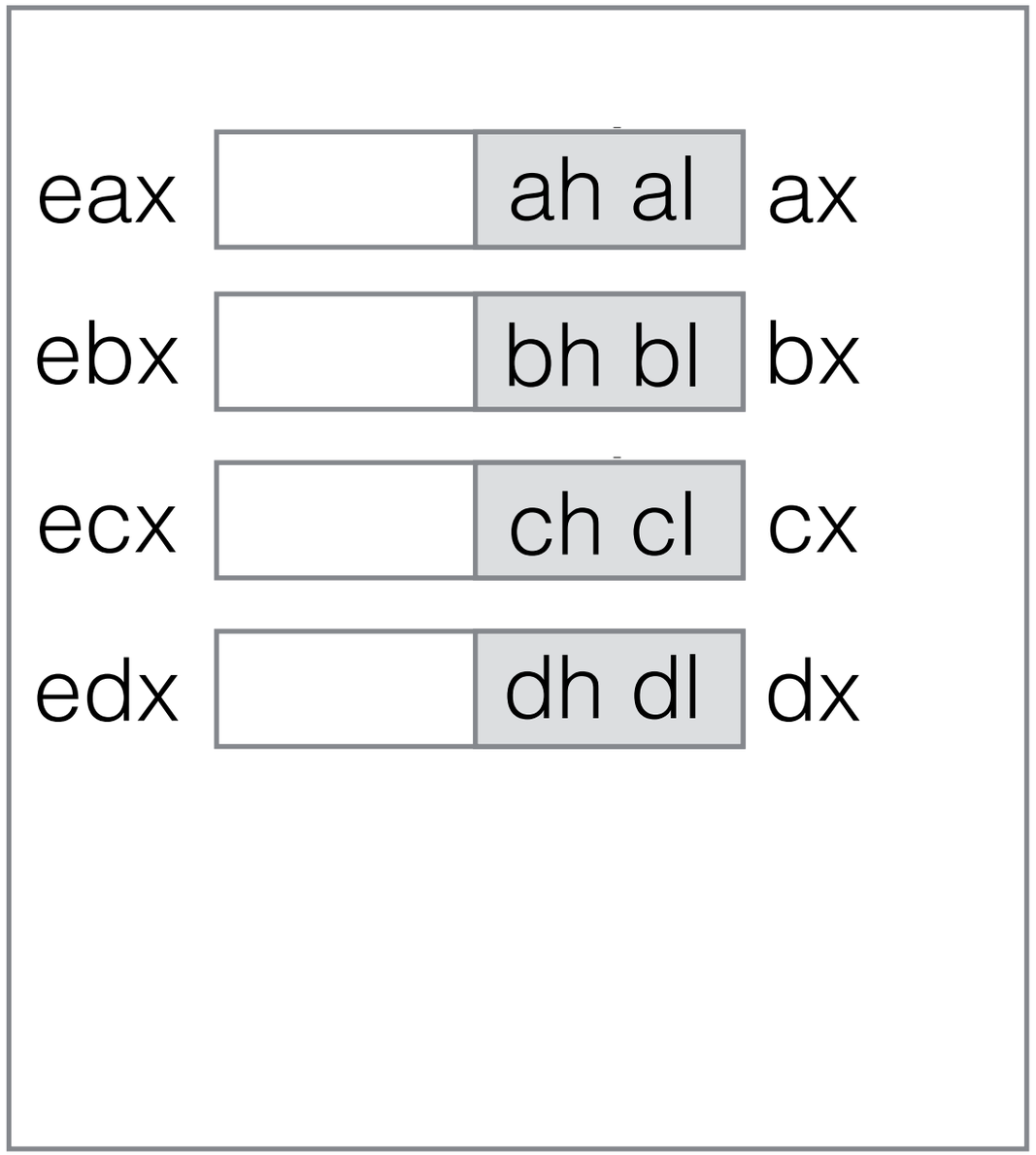
Pentium Registers

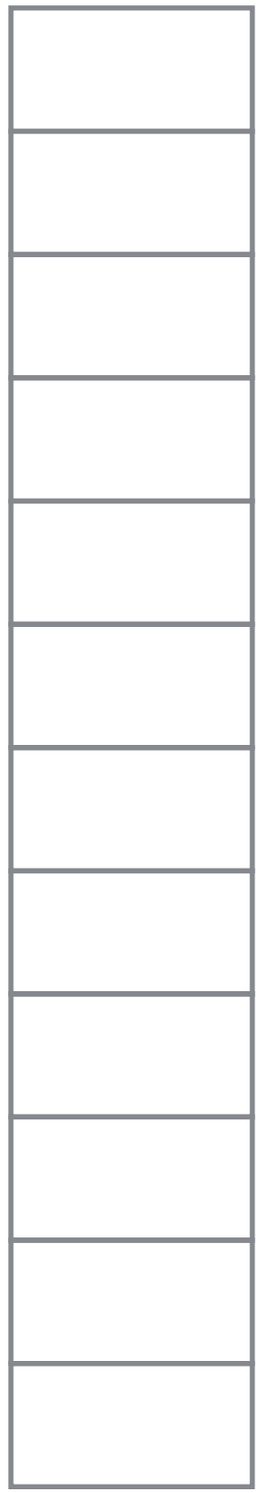




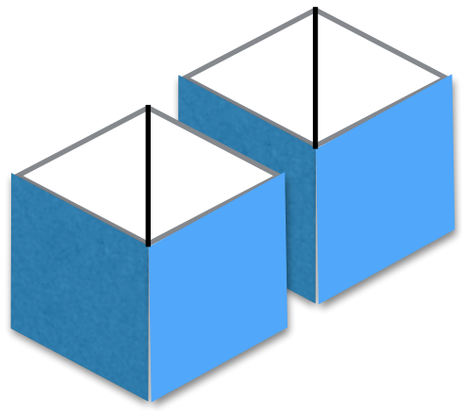


Pentium Registers





*Think of **ah** and **al** as boxes inside a bigger one called **ax**, and **ax** as half of a bigger box still, called **eax**.*



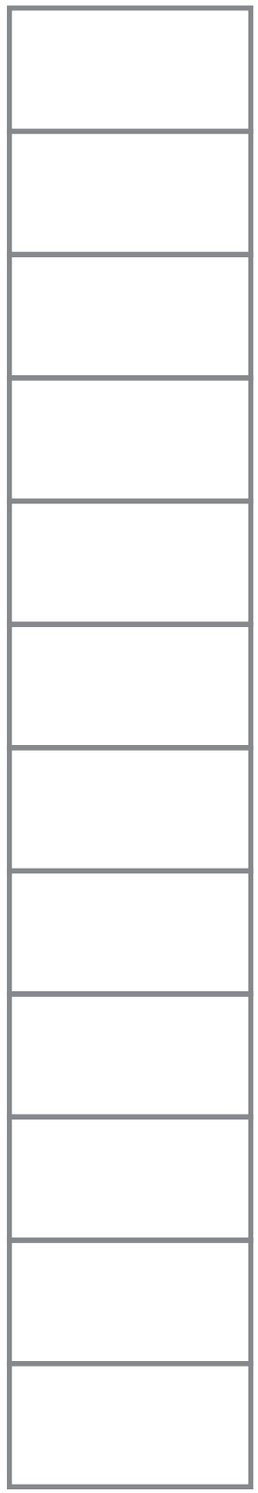
Declaring Variables

db, dw, dd

- **db**: **d**efine **b**yte storage
- **dw**: **d**efine **w**ord storage
- **dd**: **d**efine **d**ouble-word storage

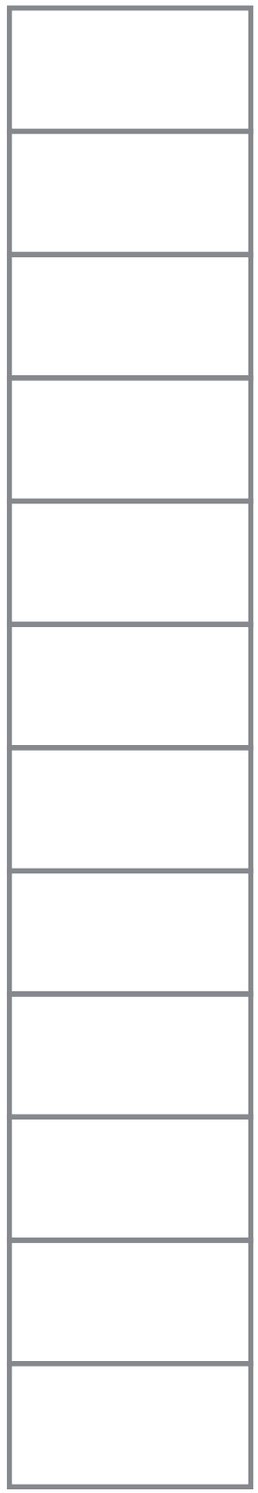
Examples: **db**

```
msg         db          "Hello", 10

a           db          0
b           db          'H'     ; also 72 or 0x48
c           db          255
d           db          0x80
```



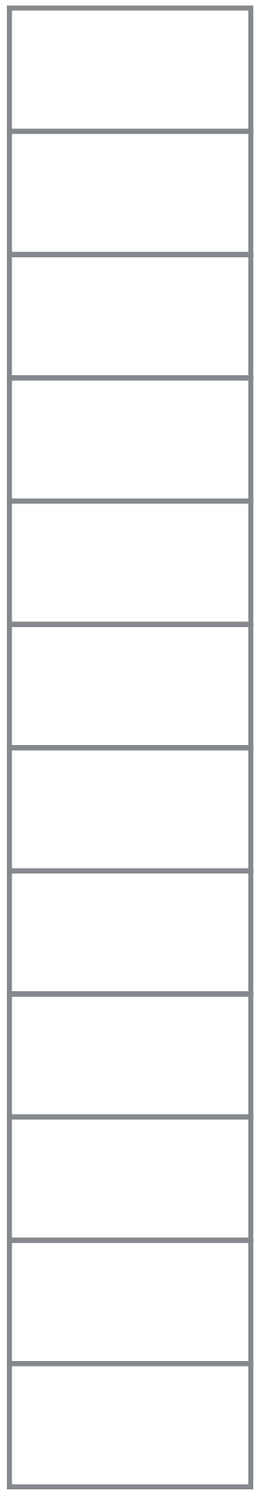
Examples: **dw**

```
x           dw          0
y           dw          1
z           dw          255
t           dw          0x1234
```



Examples: **dd**

```
alpha     dd        0
beta      dd        255
gamma     dd        0x12345678
```



We stopped here last time…

Summary of important concepts just seen

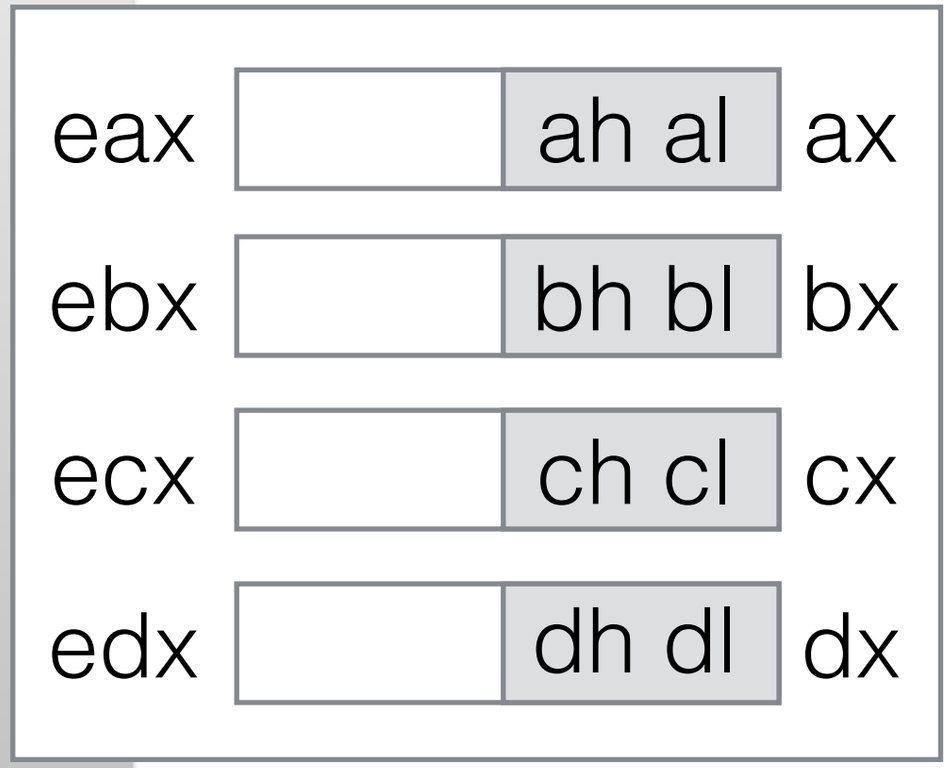
- Numbers
- Op Codes
- Machine Language
- Hexadecimal
- Executable Files

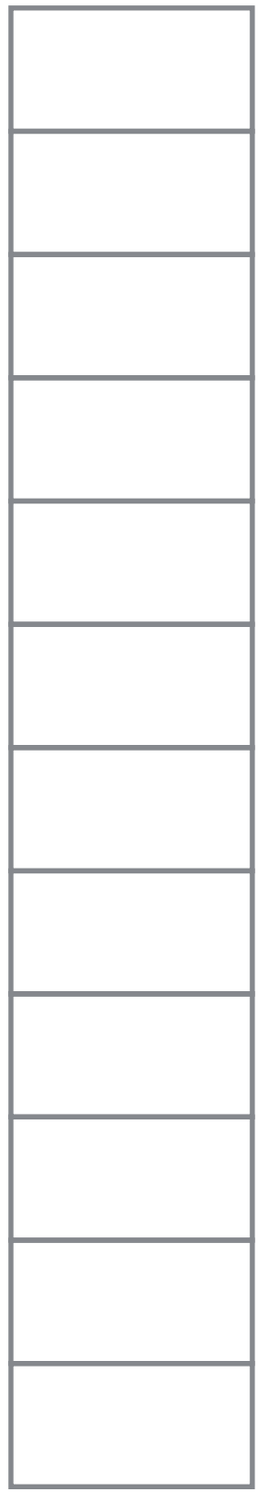
Back to the **mov** instruction

mov dest, source

Test Cases

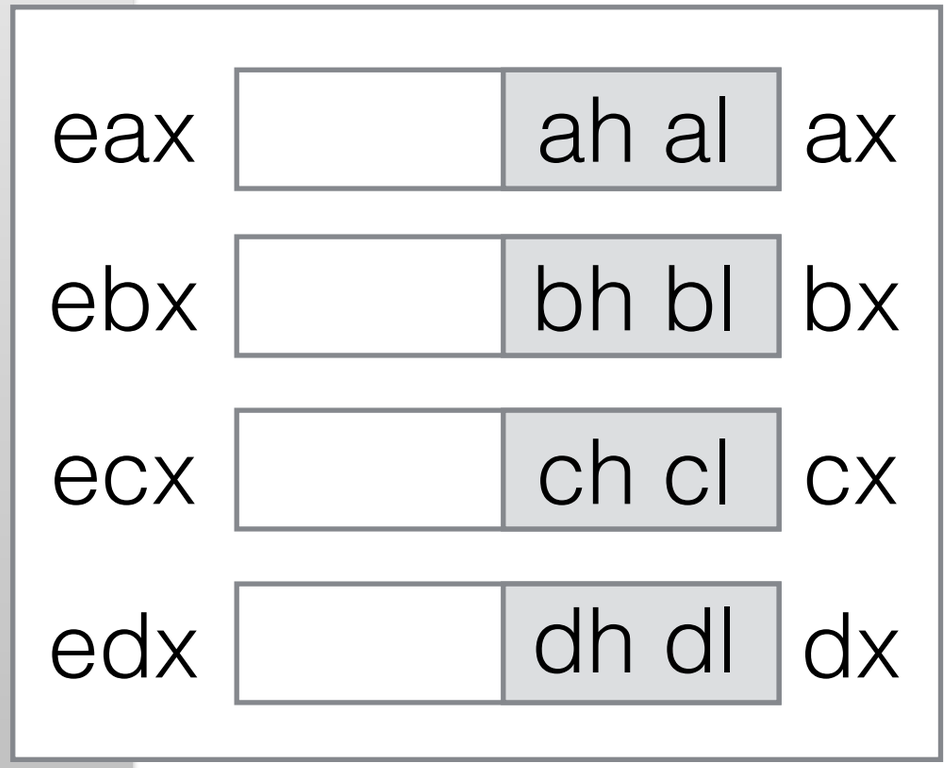
```
            section .data
lf          db      10
ch          db      0
a           dw      0x1234
b           dw      0
x           dd      0
y           dd      0x12345678

            section .text
; put lf in al
```

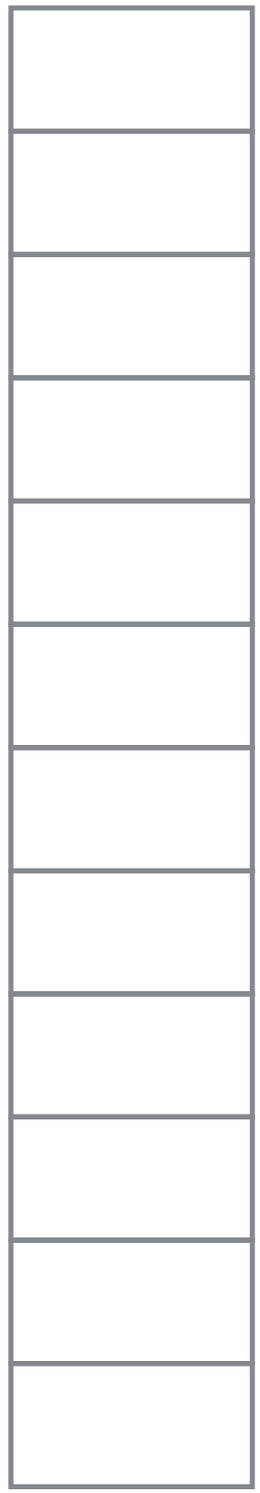




Test Cases

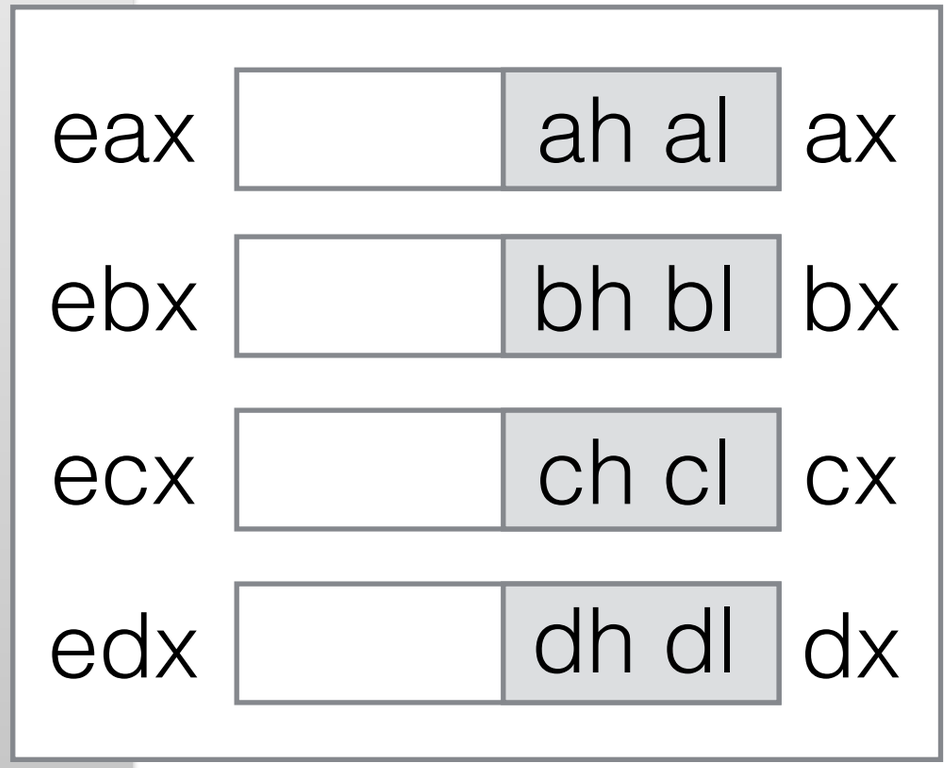
```
            section .data
lf          db      10
ch          db      0
a           dw      0x1234
b           dw      0
x           dd      0
y           dd      0x12345678

            section .text
; put al in ch
```

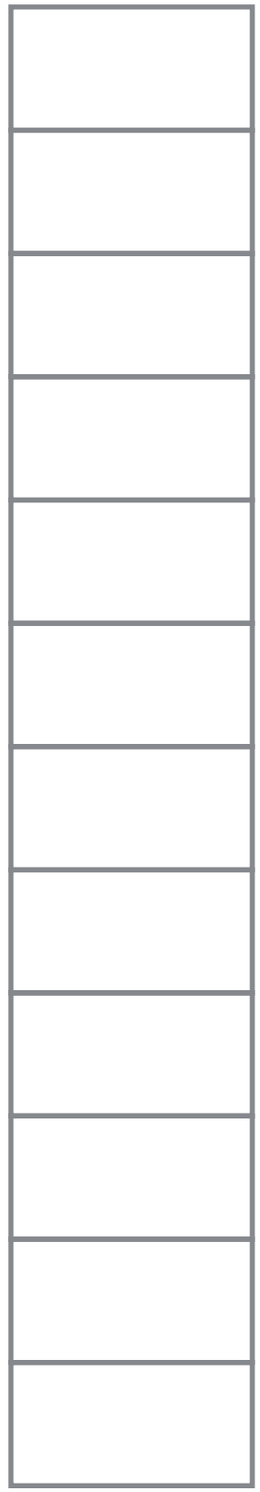




Test Cases

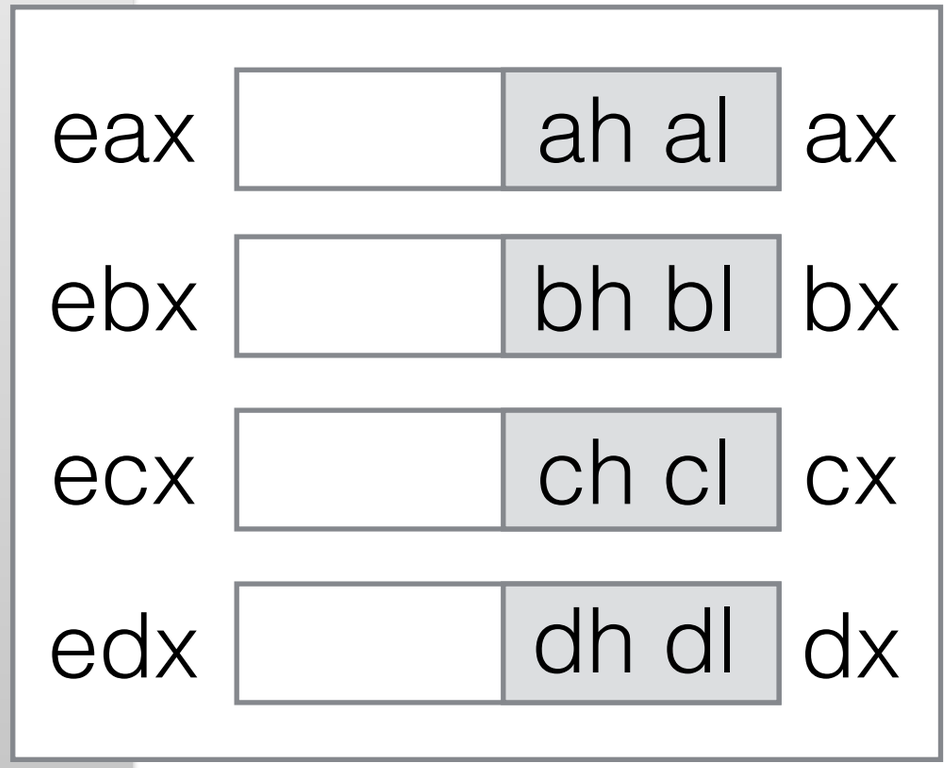
```
            section .data
lf          db      10
ch          db      0
a           dw      0x1234
b           dw      0
x           dd      0
y           dd      0x12345678

            section .text
; put a in bx


; put bx in b


; put bx in ax


; put 0 in cx
```

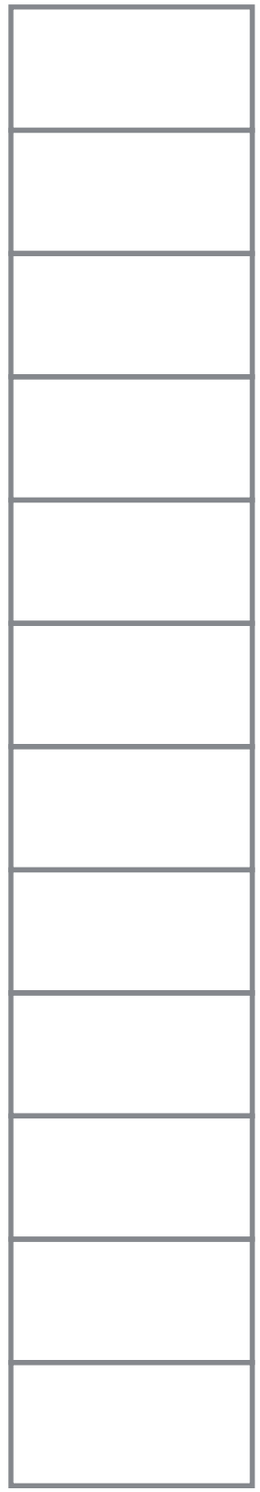




Test Cases

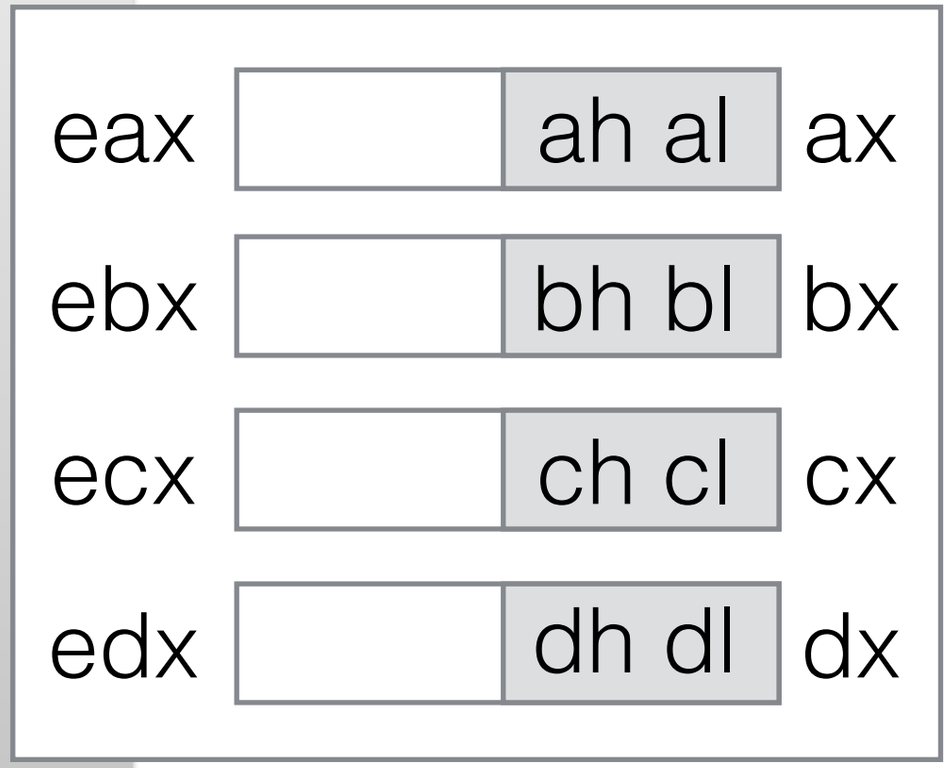
```
            section .data
lf          db      10
ch          db      0
a           dw      0x1234
b           dw      0
x           dd      0
y           dd      0x12345678

            section .text
; put x in eax


; put y in ecx


; put ecx in edx


; put ex into y
```

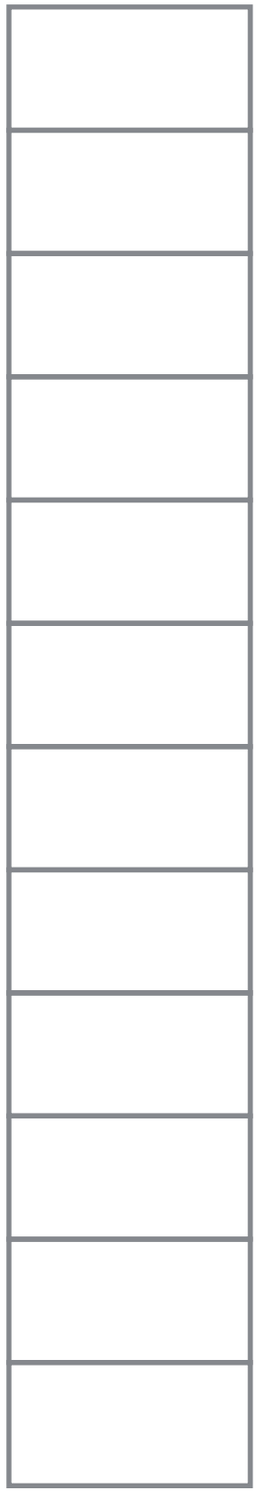




Test Cases

```
            section .data
lf          db      10
ch          db      0
a           dw      0x1234
b           dw      0
x           dd      0
y           dd      0x12345678

            section .text
; put 0 in ah


; put 3 in cx


; put 5 in edx


; put 0x12345678 into eax
```





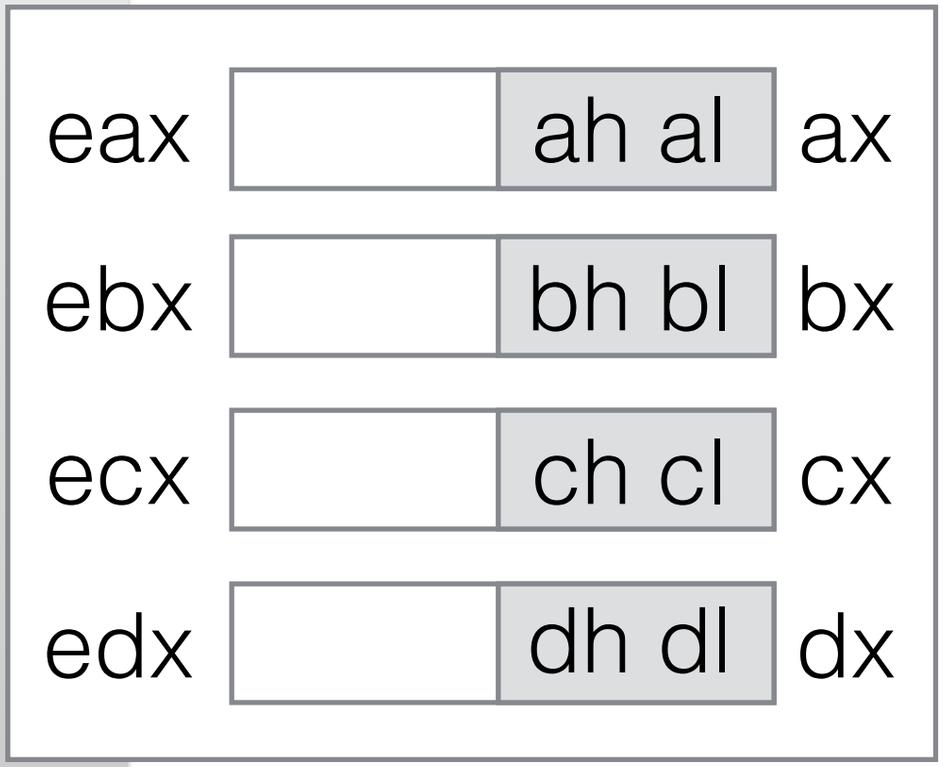
We understand **`mov`**!

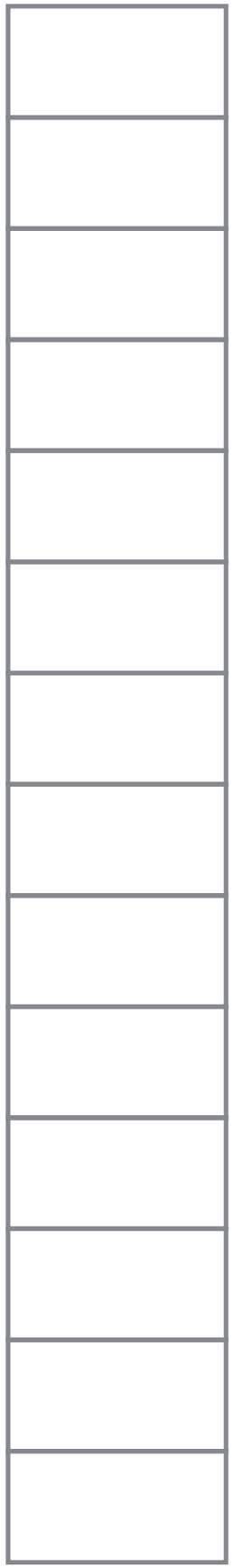
The **add** instruction

add dest, source

Test Cases

```
            section .data
lf          db      10
ch          db      0
a           dw      0x1234
b           dw      0
x           dd      0
y           dd      0x12345678

            section .text
; add 3 to ch


; add 100 to b


; add -1 to edx


; add x to y
```





Reminder: Our Goal is…

```
int x, y, sum;

x = 3;
y = 5;
sum = x + y;
```

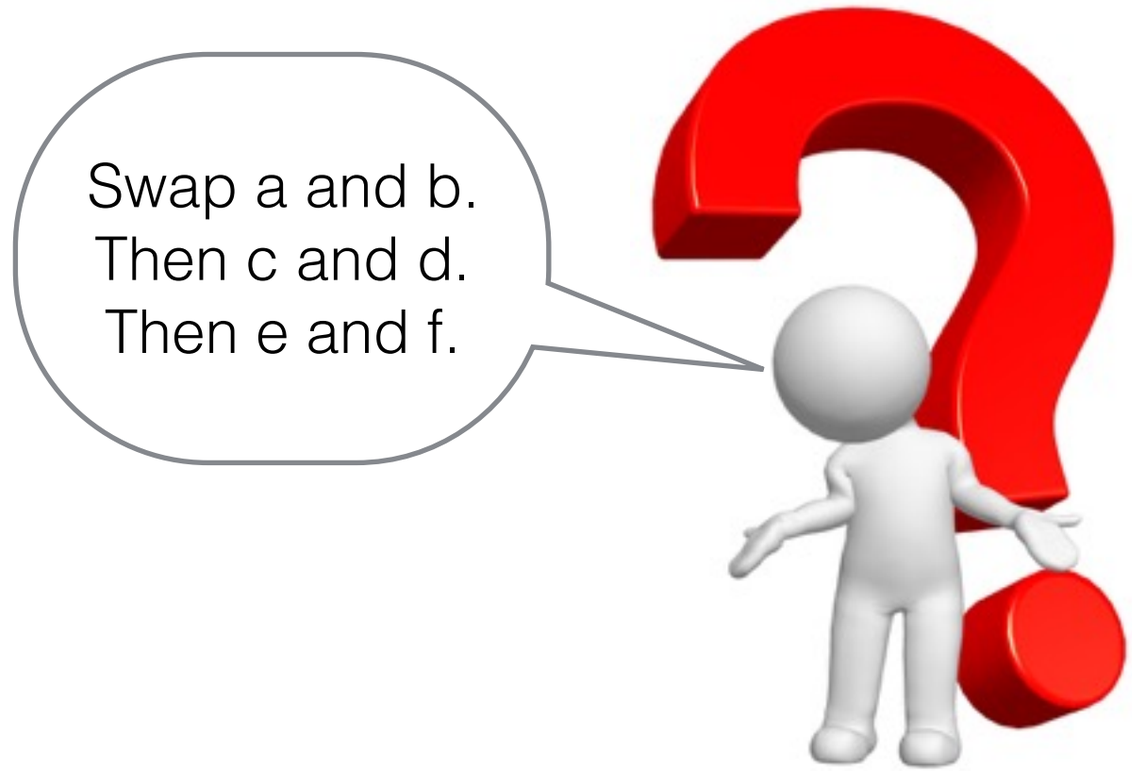
Reminder: Our Goal is…

Translate this into Assembly

```
int x, y, sum;

x = 3;
y = 5;
sum = x + y;
```

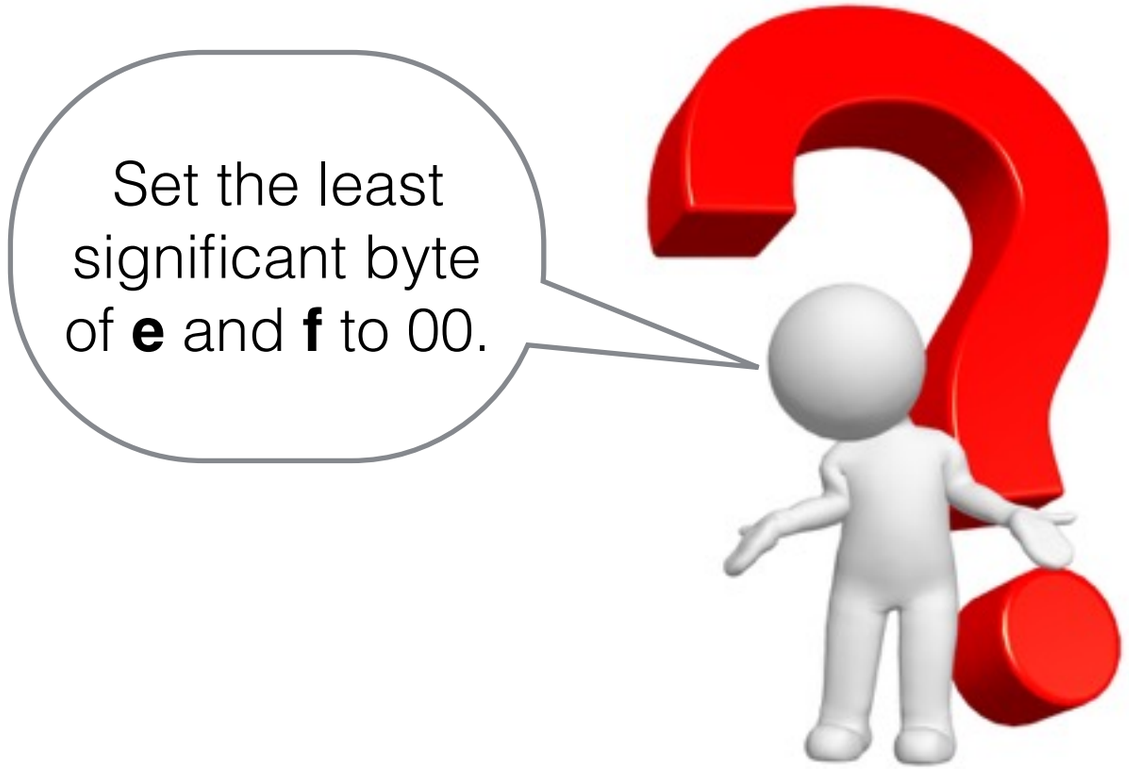
Exercise

```
            section .data
a           db      10
b           db      0
c           dw      0x1234
d           dw      0
e           dd      0
f           dd      0x12345678

            section .text
```



Exercise

```
            section .data
a           db      10
b           db      0
c           dw      0x1234
d           dw      0
e           dd      0xcdef
f           dd      0x12345678

            section .text
```



Exercise

```
            section .data
a           db
b           db
c           dw
d           dw
e           dd
f           db

            section .text
```

99
88
77
66
55
44
33
22
1f
1a
11
00

hex

a

reconstruct the declarations of a, b, c, d, e and f.

typical midterm question!

Exercise

```
            section .data
a           db
b           db
c           dw
d           dw
e           dd
f           db

            section .text
```

99
88
77
66
55
44
33
22
31
26
11
00 ← a

dec

reconstruct the declarations of a, b, c, d, e and f.

typical midterm question!

Follow a step by step execution of the program