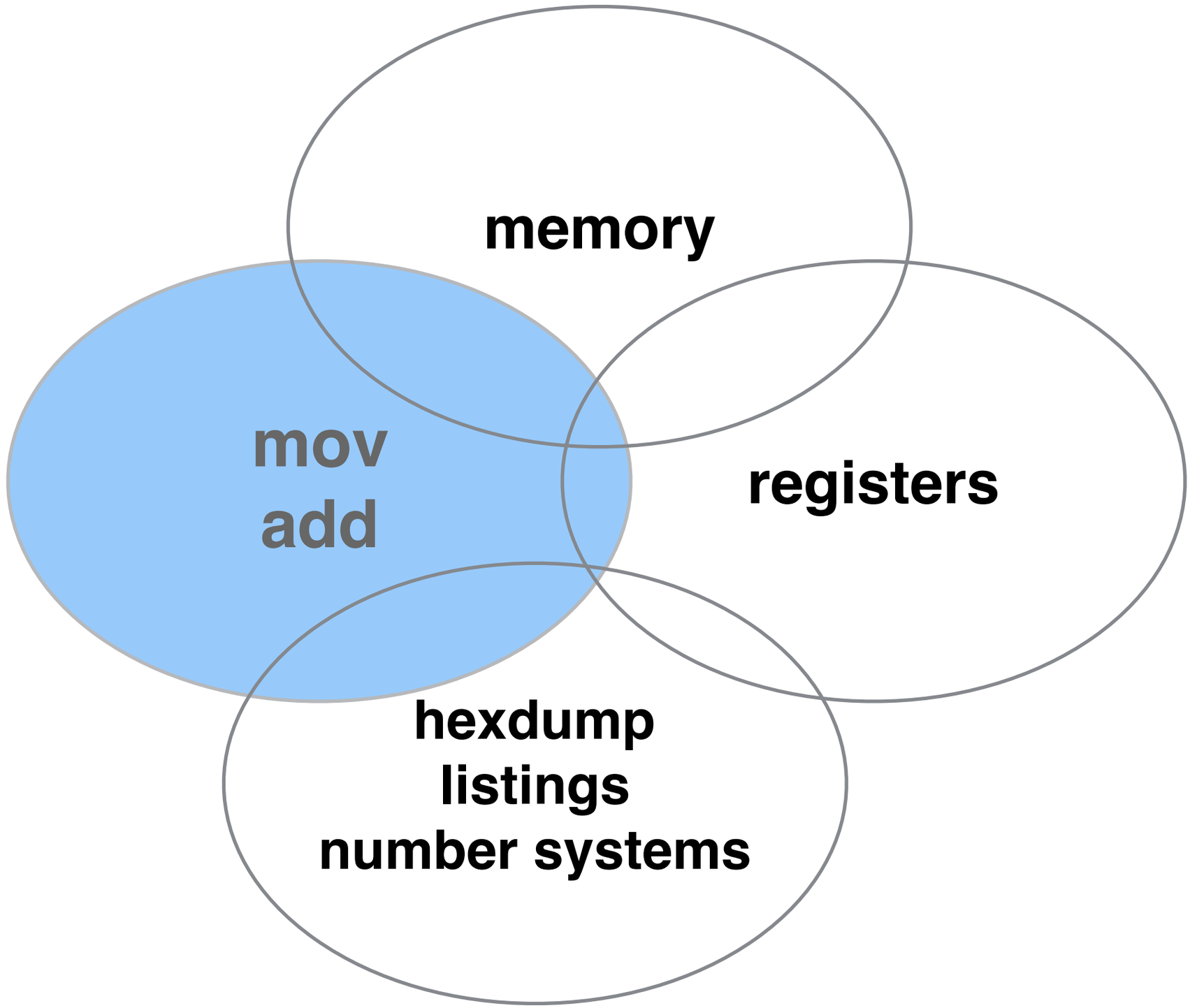
Smith College

Computer Science

CSC231 - Assembly

Week #3

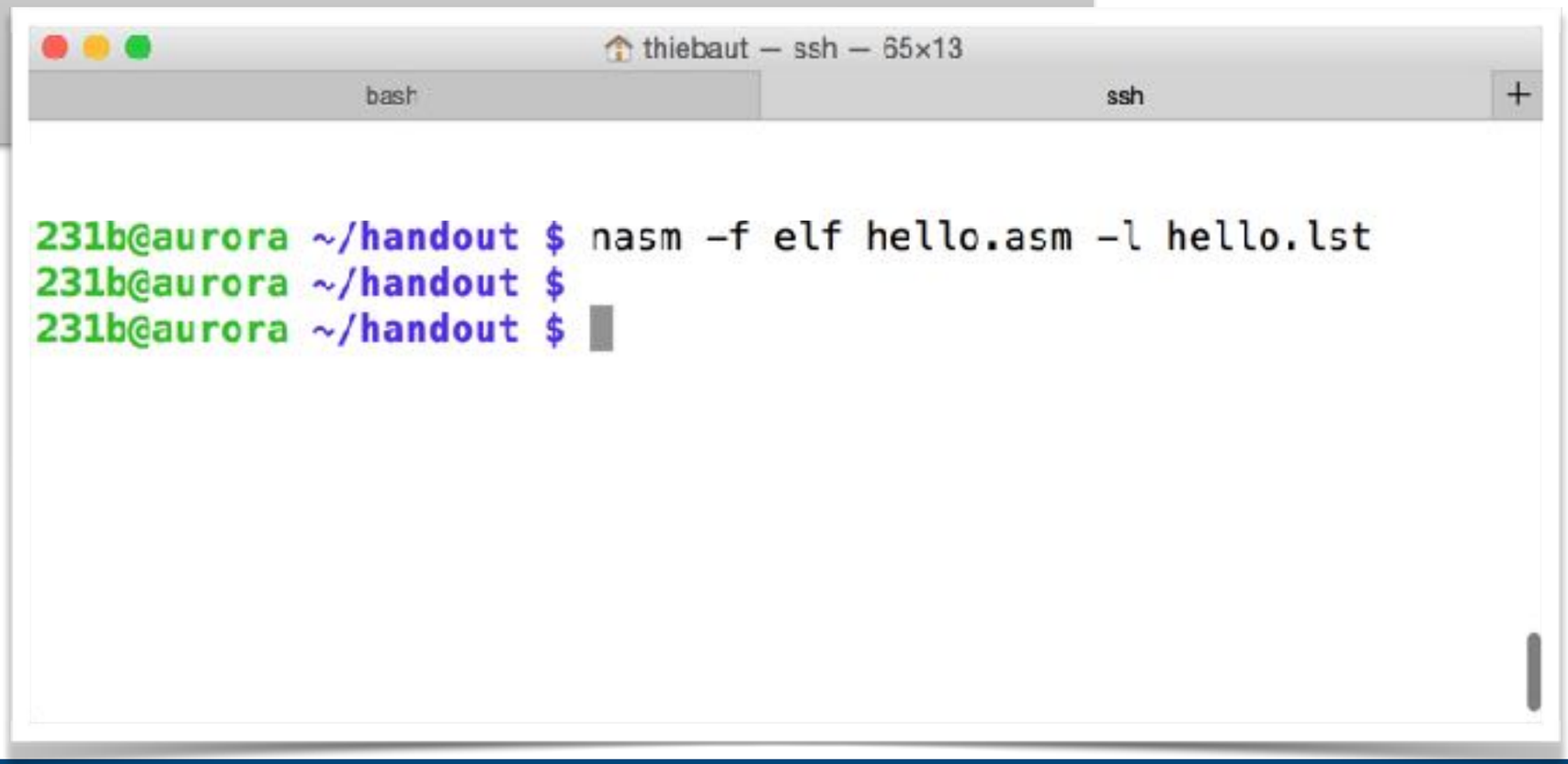
Dominique Thiébaut
dthiebaut@smith.edu


memory
mov
add
registers
hexdump
listings
number systems

Let's Review 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
```



```
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
```

```
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

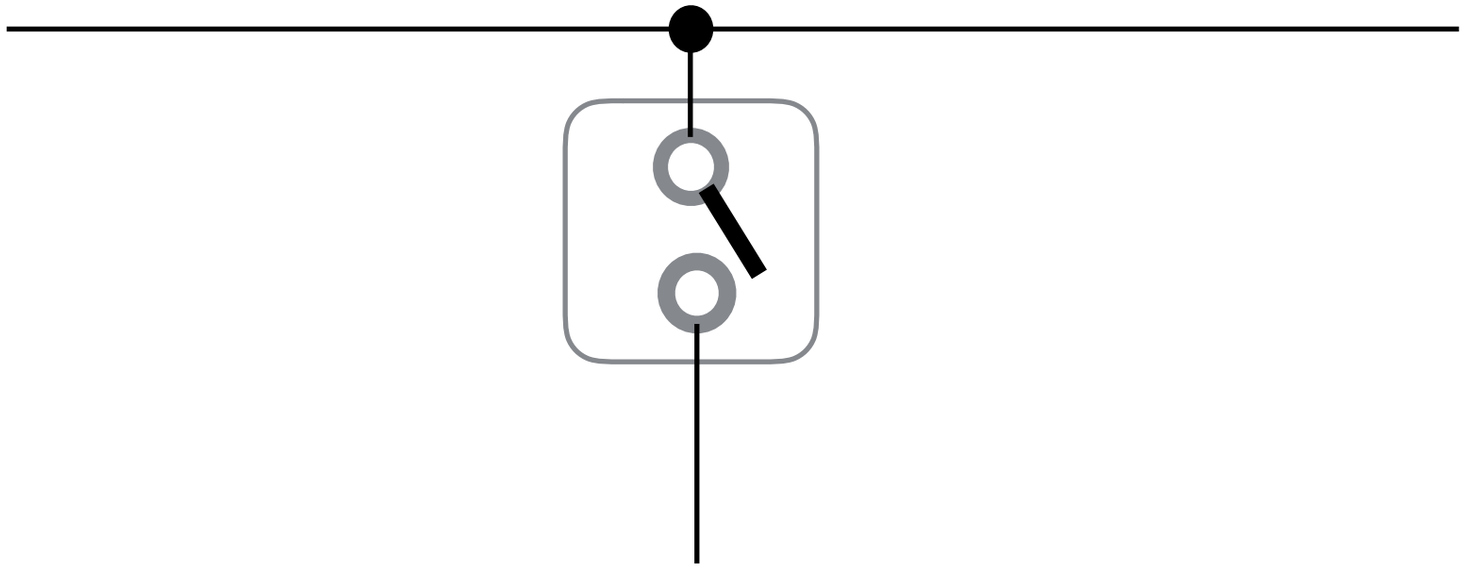
```
                                                                  0 00  |.ELF............|
                                                                  0 00  |............4...|
00000020  dc 00 00 00 00 00 00 00  34 00 20 00 02 00 28 00  |........4. ...(.|
00000030  06 00 03 00 01 00 00 00  00 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.......|
```

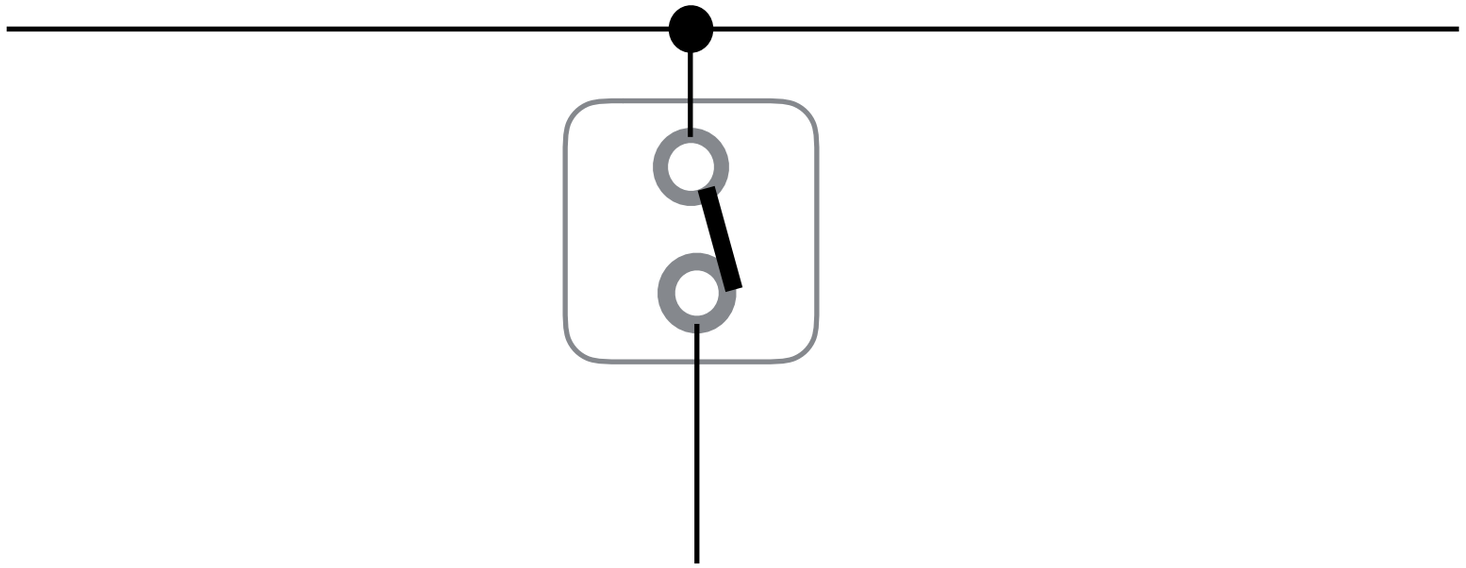
```
 11                                             section .data
 12 00000000 48656C6C6F20746865-     Hello      db      "Hello there!", 10, 10
 13 00000009 7265210A0A
 14                                  HelloLen   equ     $-Hello
 15
 16                                             section .text
 17                                             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
```

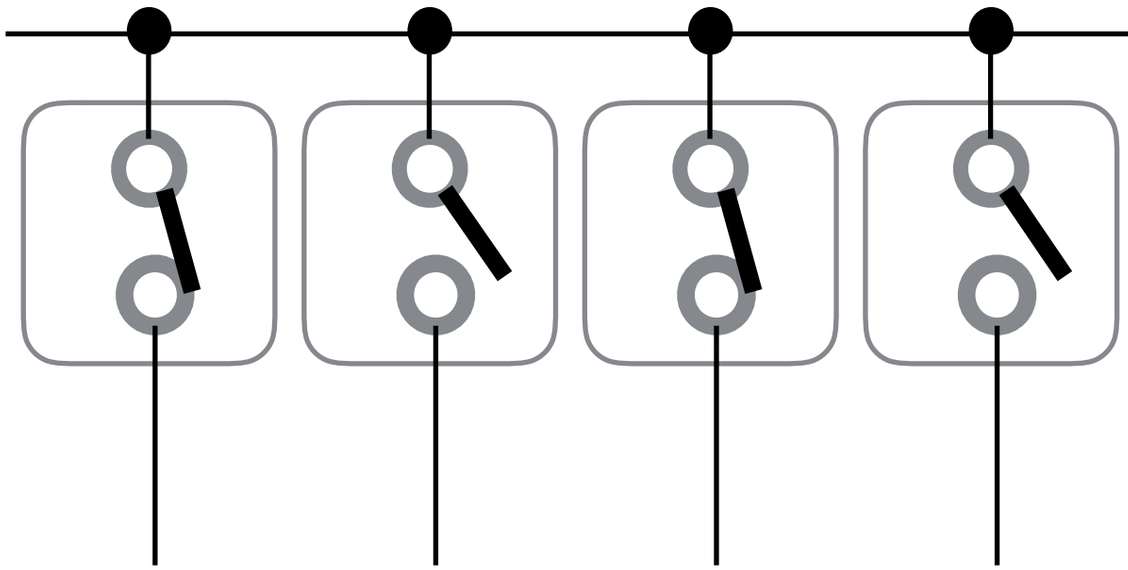
Why Hexadecimal?

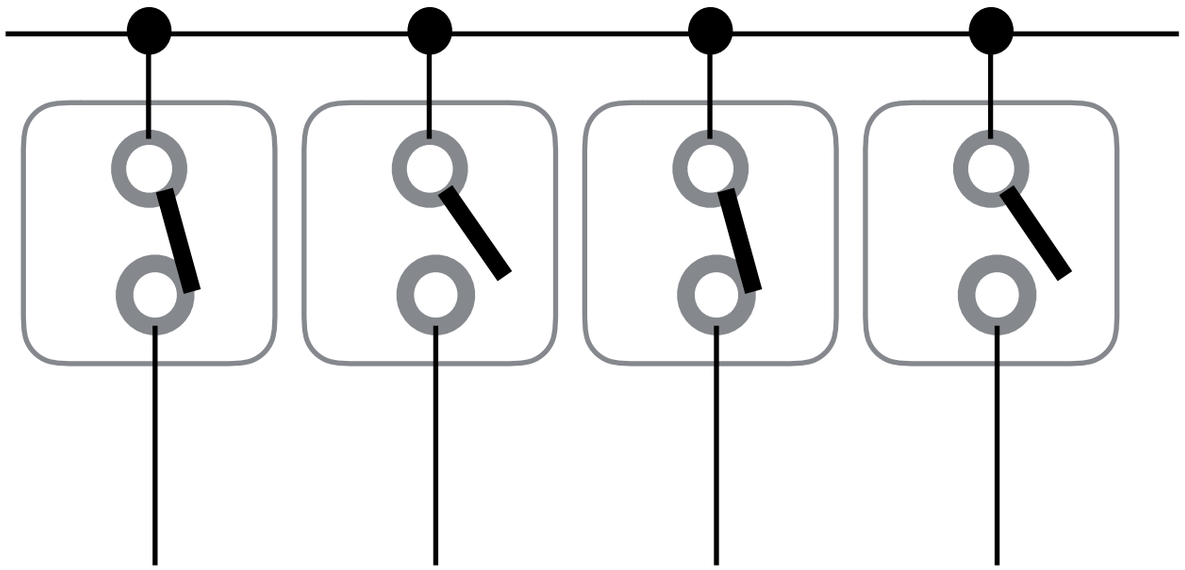
ssh

```
                                                                  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.......|
```

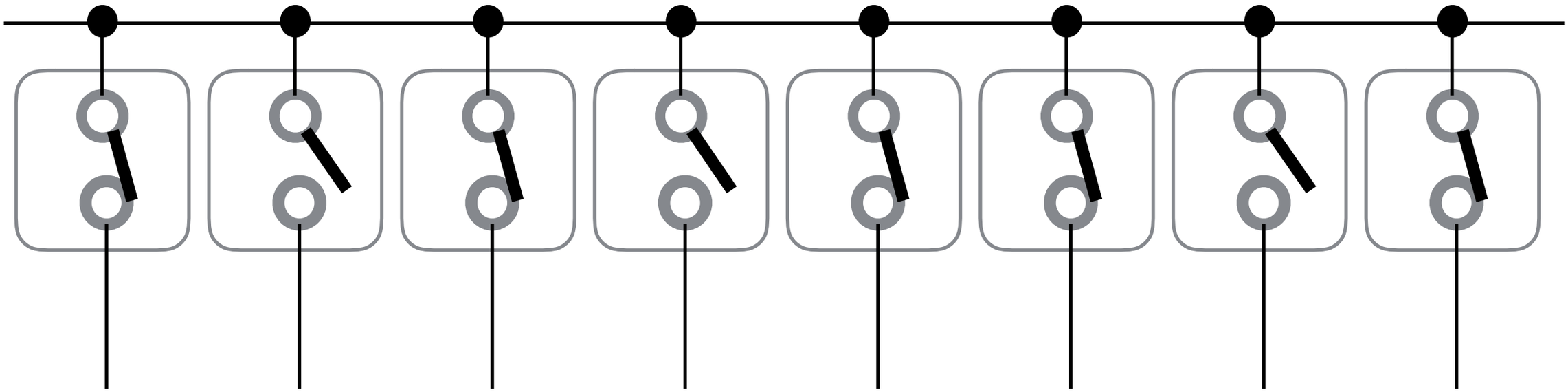




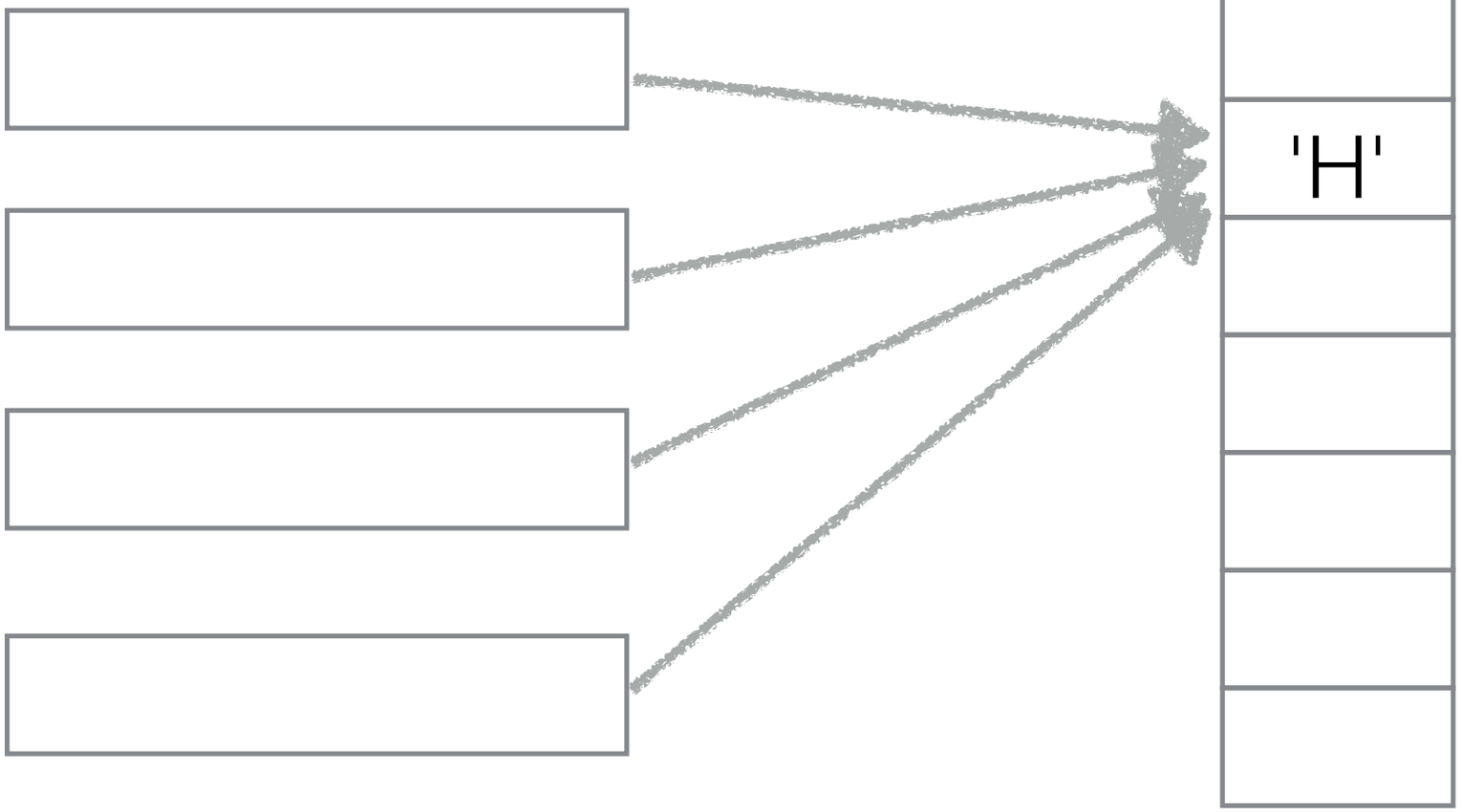




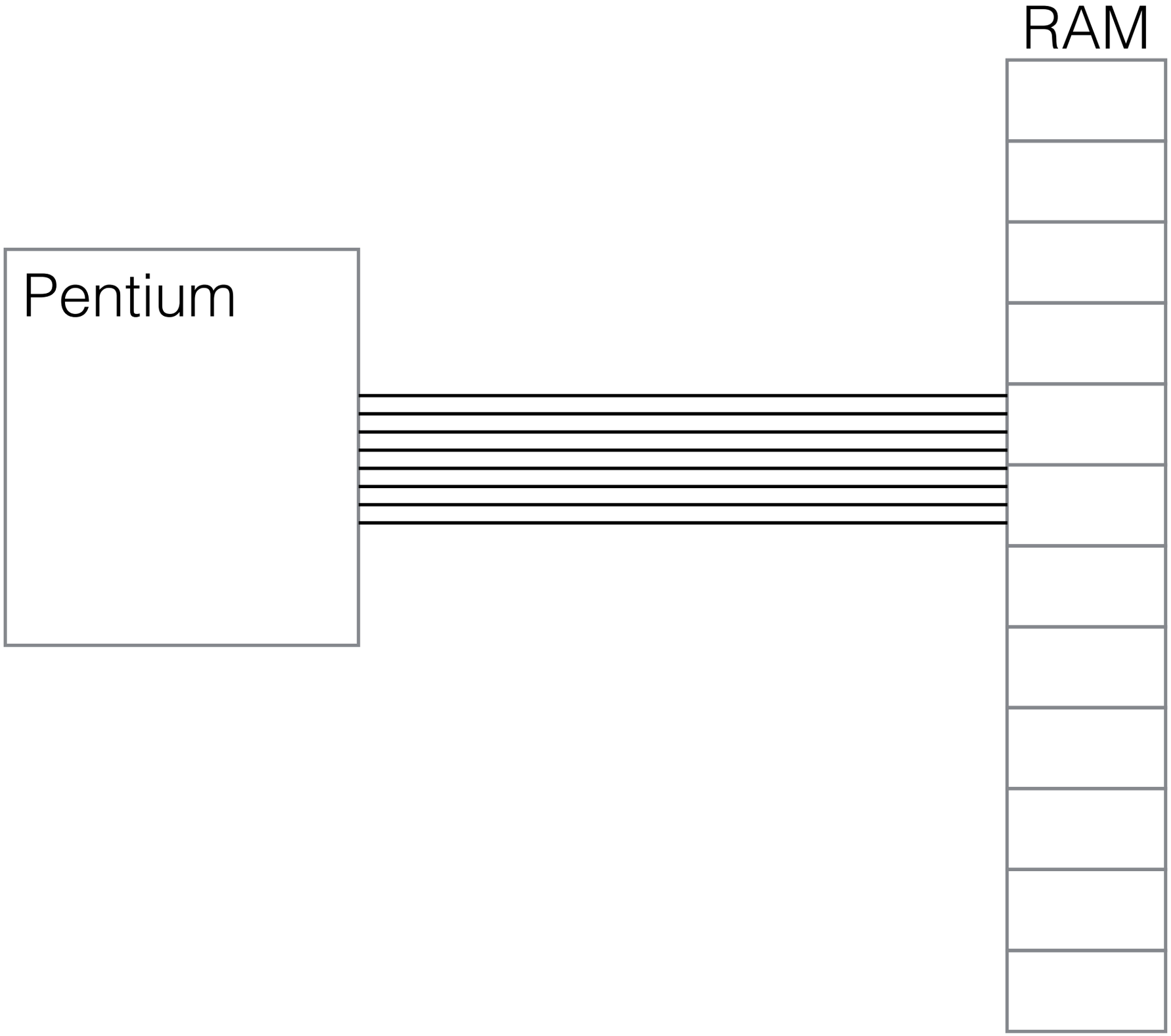
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111



Dec	Hx	Oct	Html	Chr
64	40	100	@	@
65	41	101	A	A
66	42	102	B	B
67	43	103	C	C
68	44	104	D	D
69	45	105	E	E
70	46	106	F	F
71	47	107	G	G
72	48	110	H	H
73	49	111	I	I



Processor-RAM Connection



Our Goal for This Week

```
int x, y, sum;

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

Plan

- Mov & add instructions
- Registers
- Memory storage options

The mov instruction

```
mov  dest, source
```

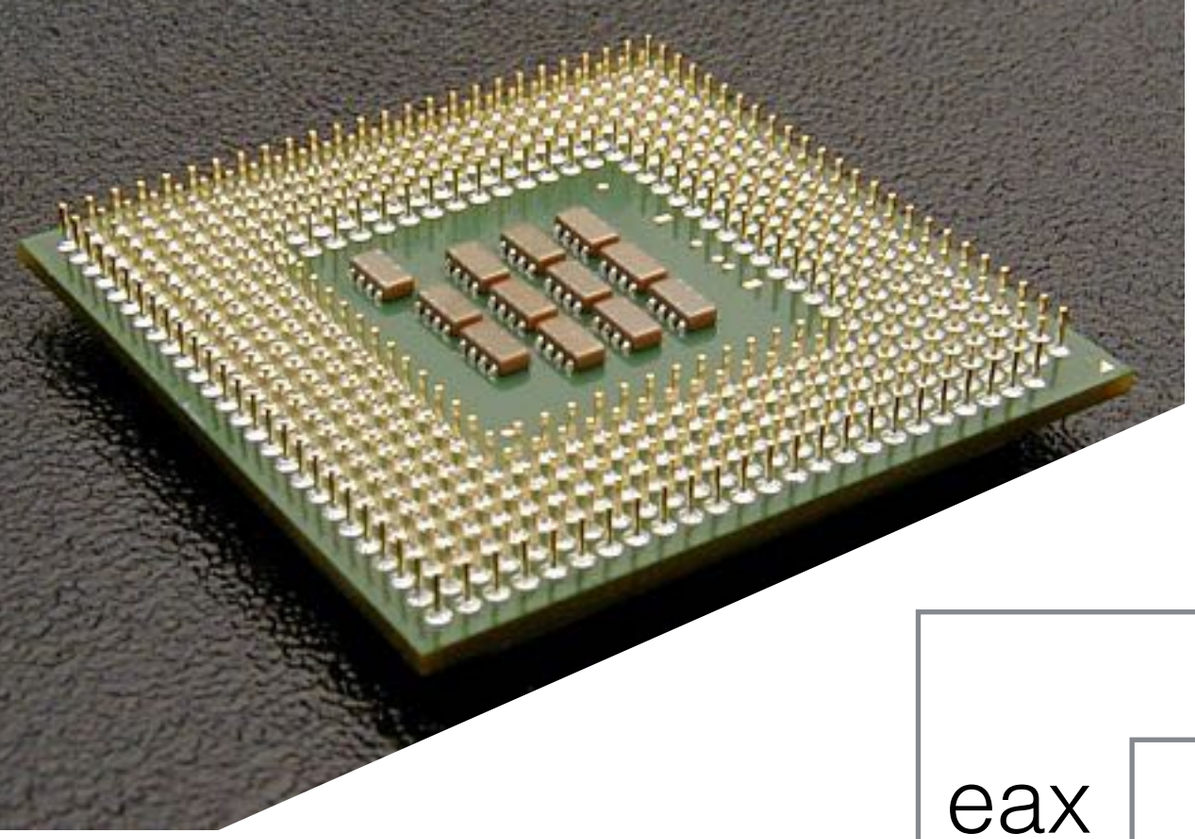
The mov instruction

```
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
```

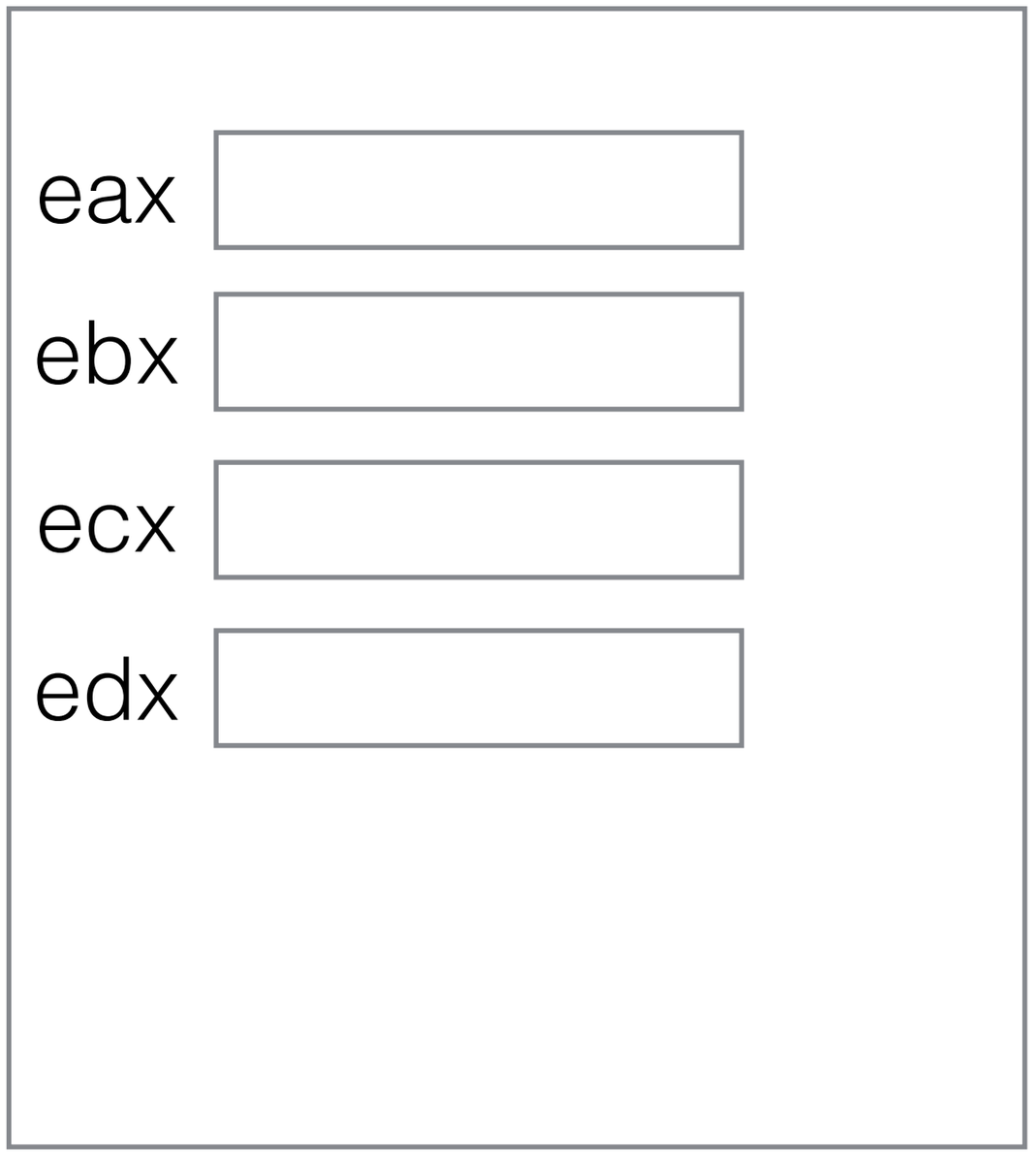
mov dest, source

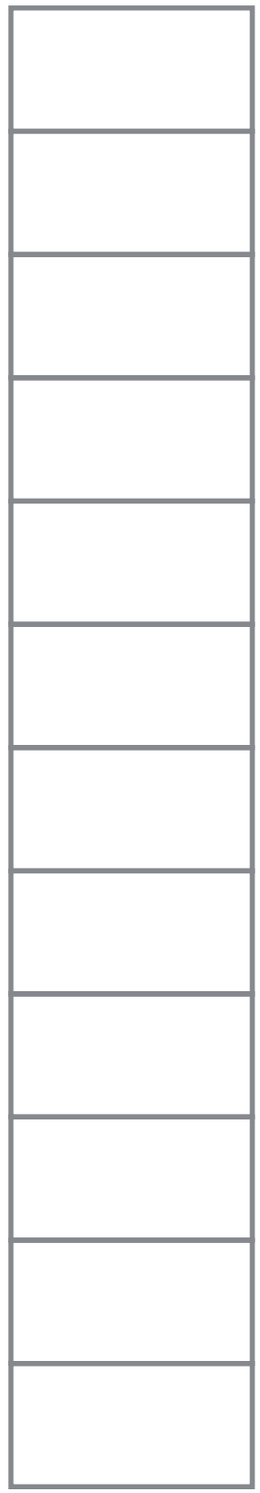
Operands

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

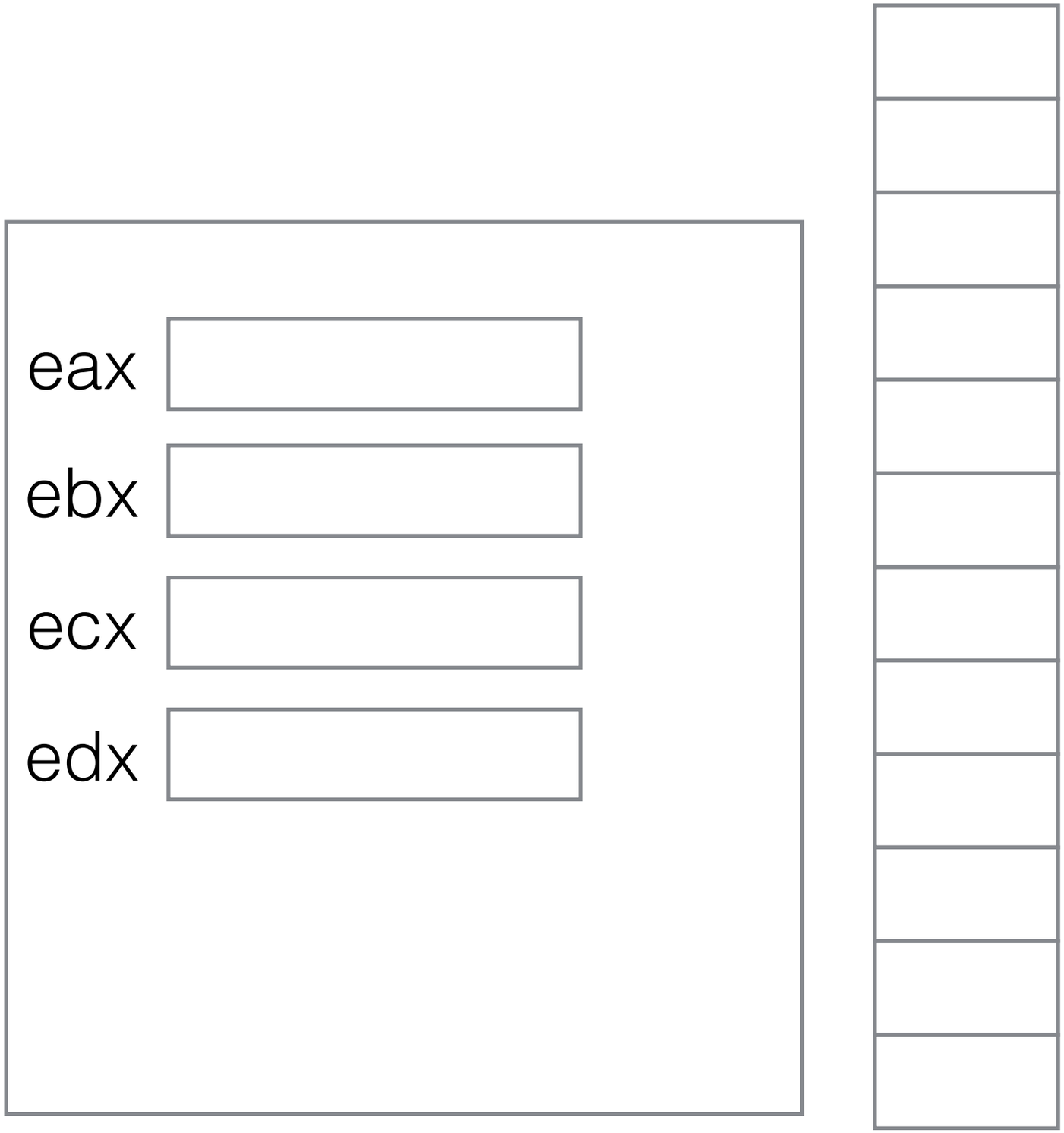


Pentium Registers

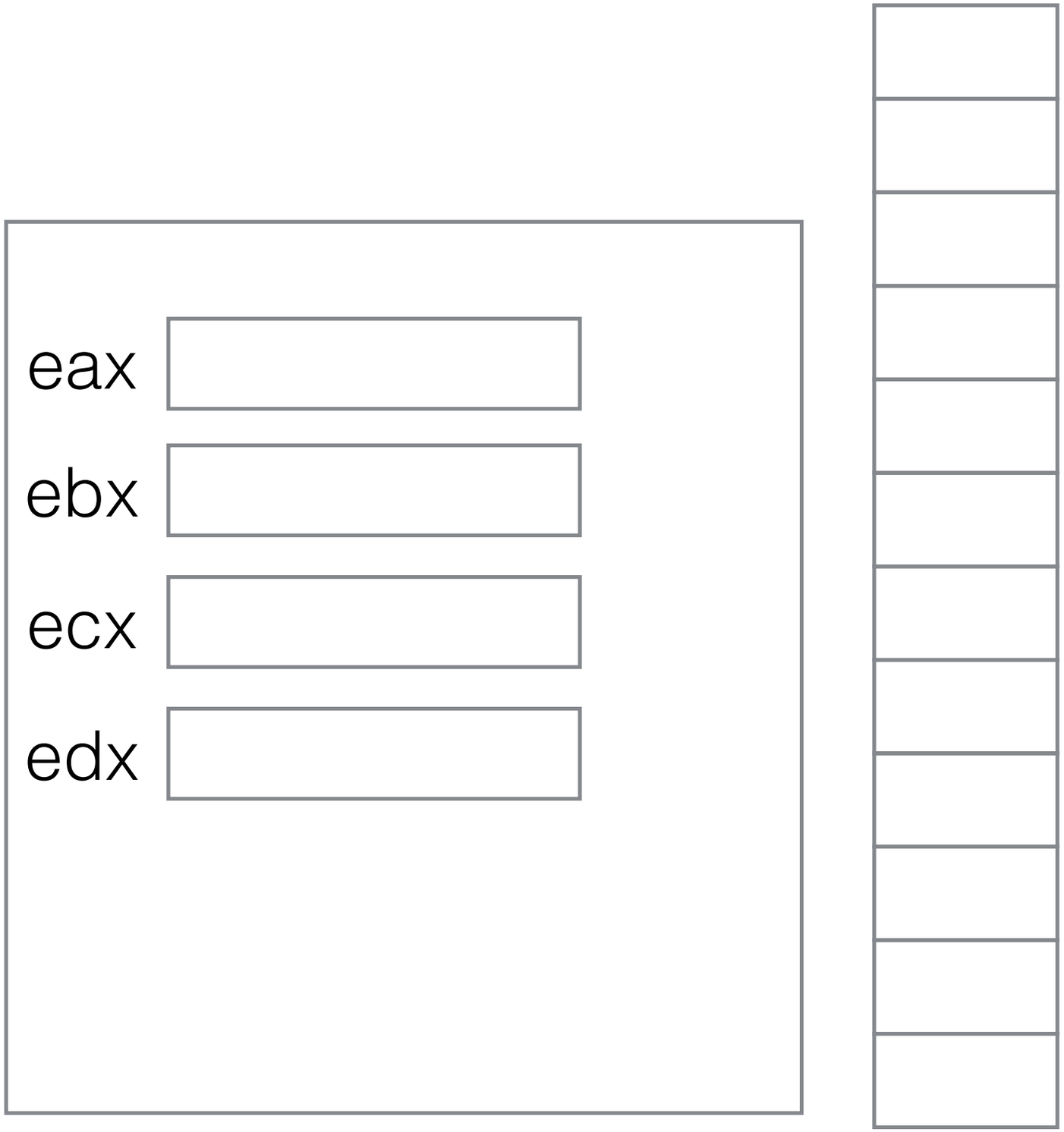




```
        section .data
a       dd      1234


        section .text
        mov     eax, 34
        mov     ebx, 12345

        mov     edx, eax
        mov     ecx, ebx
```



```
        section .data
a       dd      1234


        section .text
        mov     eax, dword[a]
        mov     ebx, eax

        mov     eax, 1234
        mov     dword[a], eax
```



We stopped here last time…

Review

- bit
- nybble
- byte
- double-word

Review

- Hexadecimal:

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Review

0
1 2^1

00
01
10 2^2
11

0000
0001
…
1110
1111

$2^4 = 16$

0000 0000
0000 0001
…
1111 1110
1111 1111

$2^8 = 256$

Review

00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000001

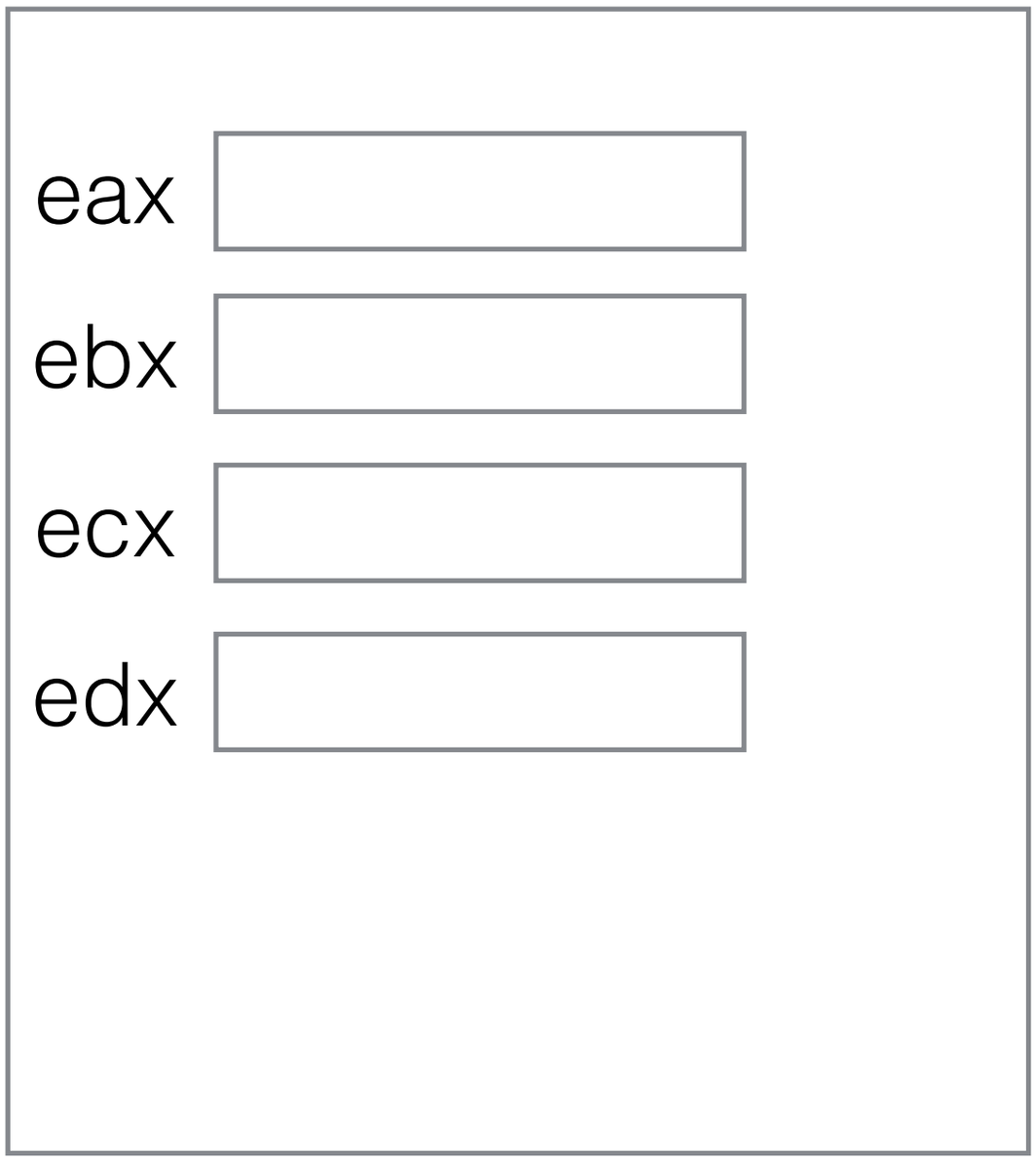
…

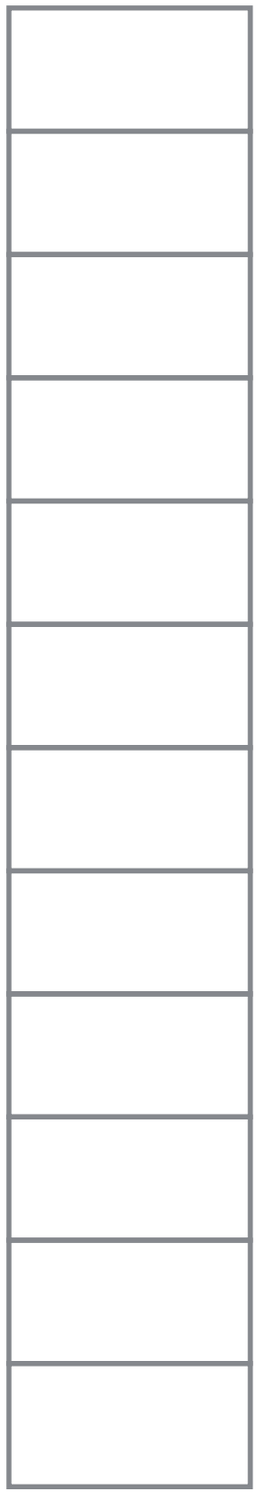
11111111 11111111 11111111 11111110
11111111 11111111 11111111 11111111

$2^{32} = 4{,}294{,}967{,}296$

Exercise

```
        section .data
hello   db      "Hi!"
helloL  equ     $-hello
a       dd      1234


        section .text
        mov     eax, 123456789
        mov     dword[a], eax

        mov     dword[a], 0
        mov     ecx, dword[a]
```



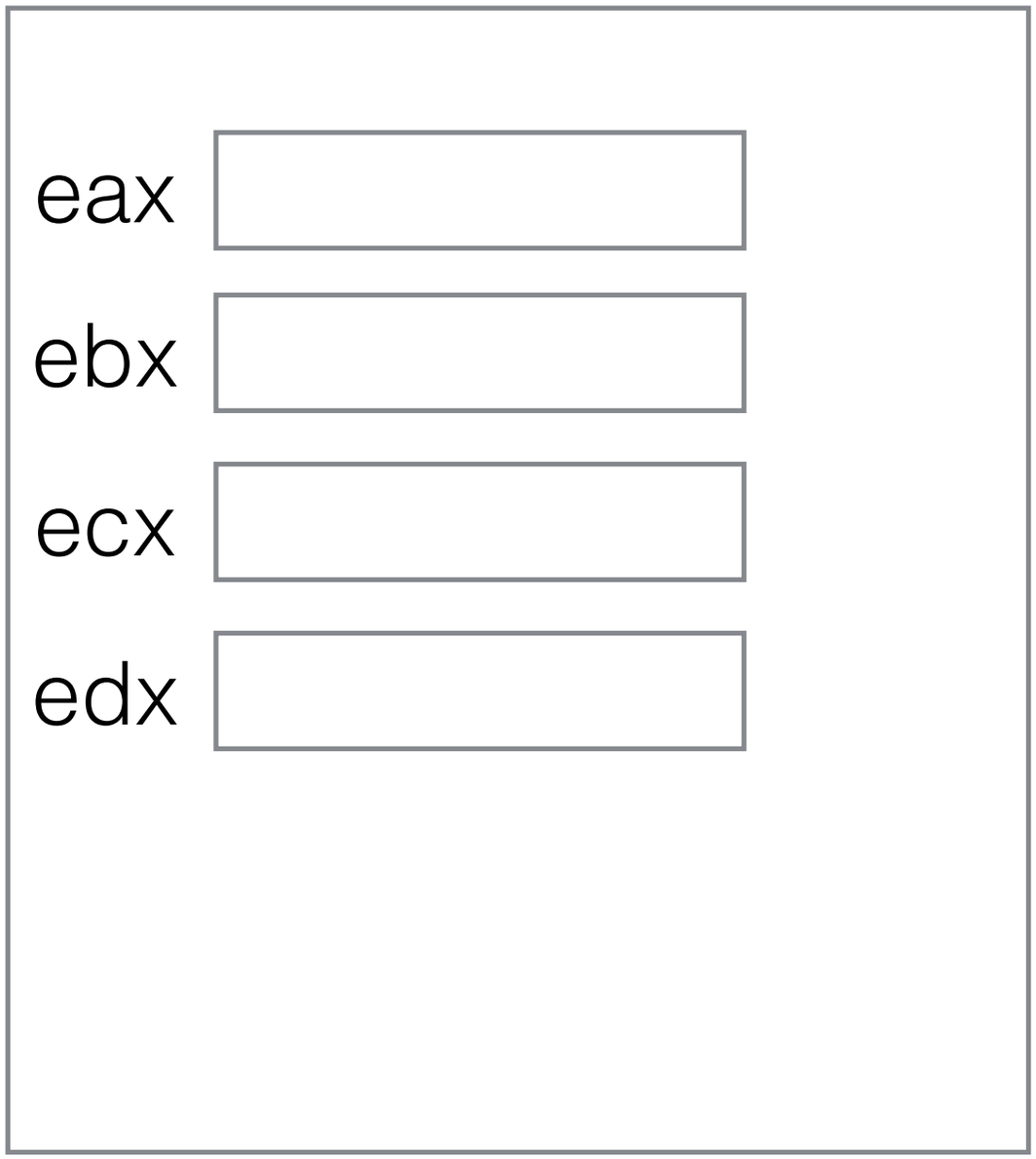


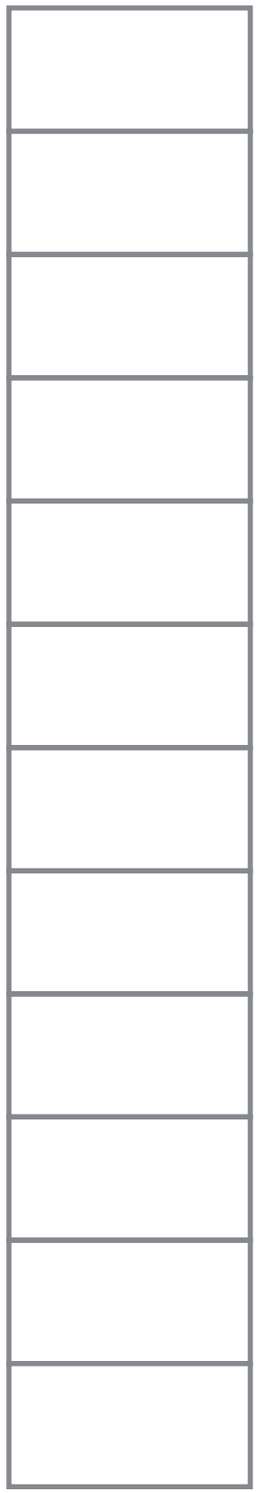
The add instruction

```
add  dest, source
```

Exercise

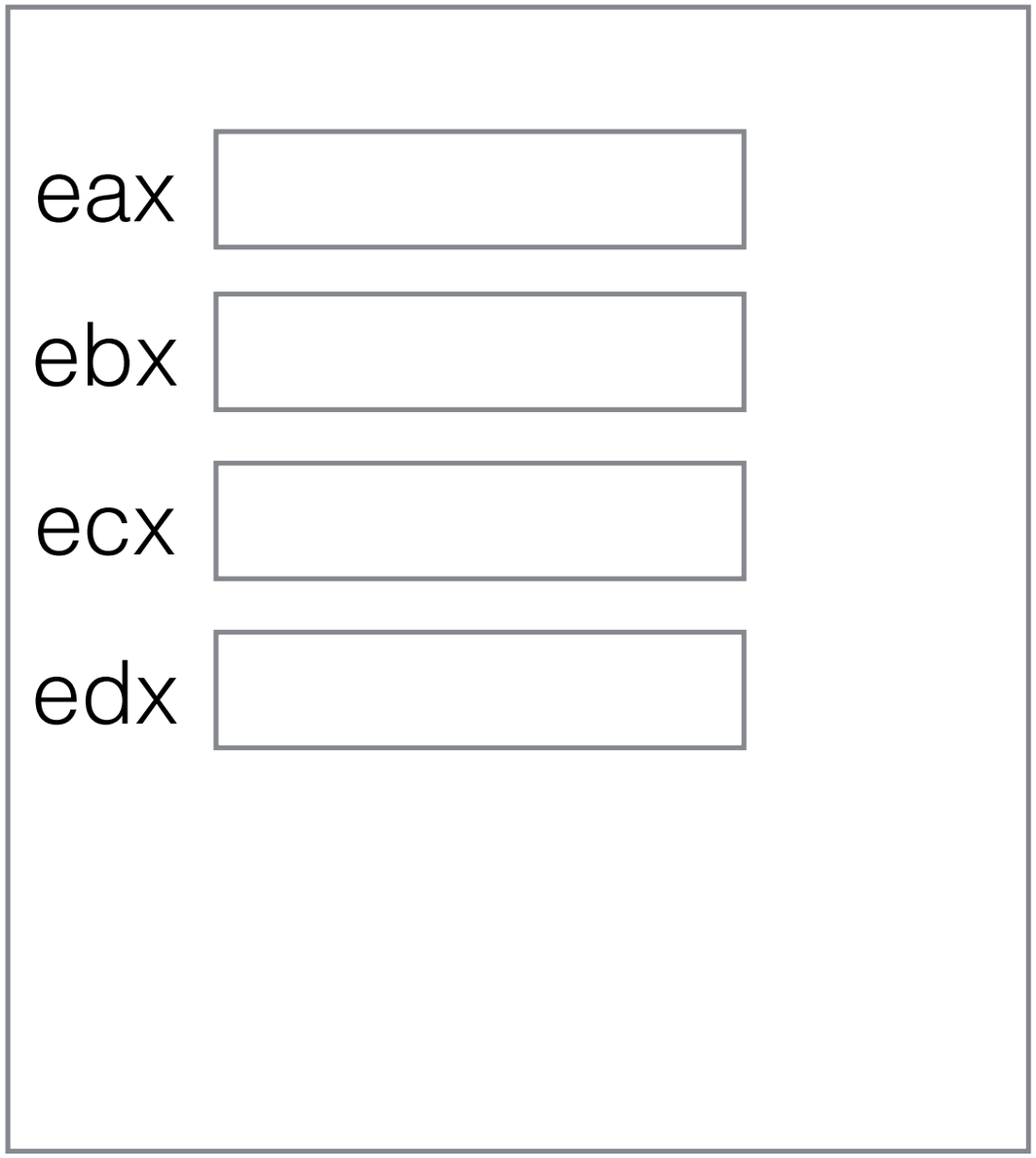
```
        section .data
a       dd      1234


        section .text
        mov     eax, 3
        mov     ebx, 5

        add     eax, ebx
```

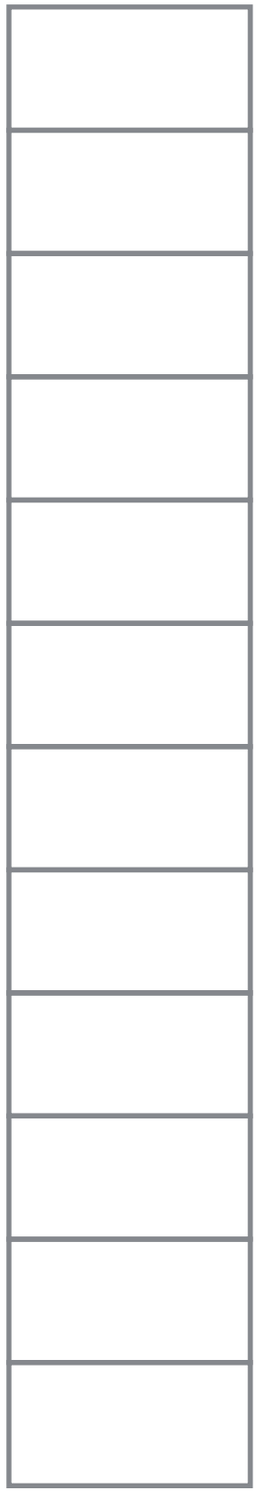




Exercise

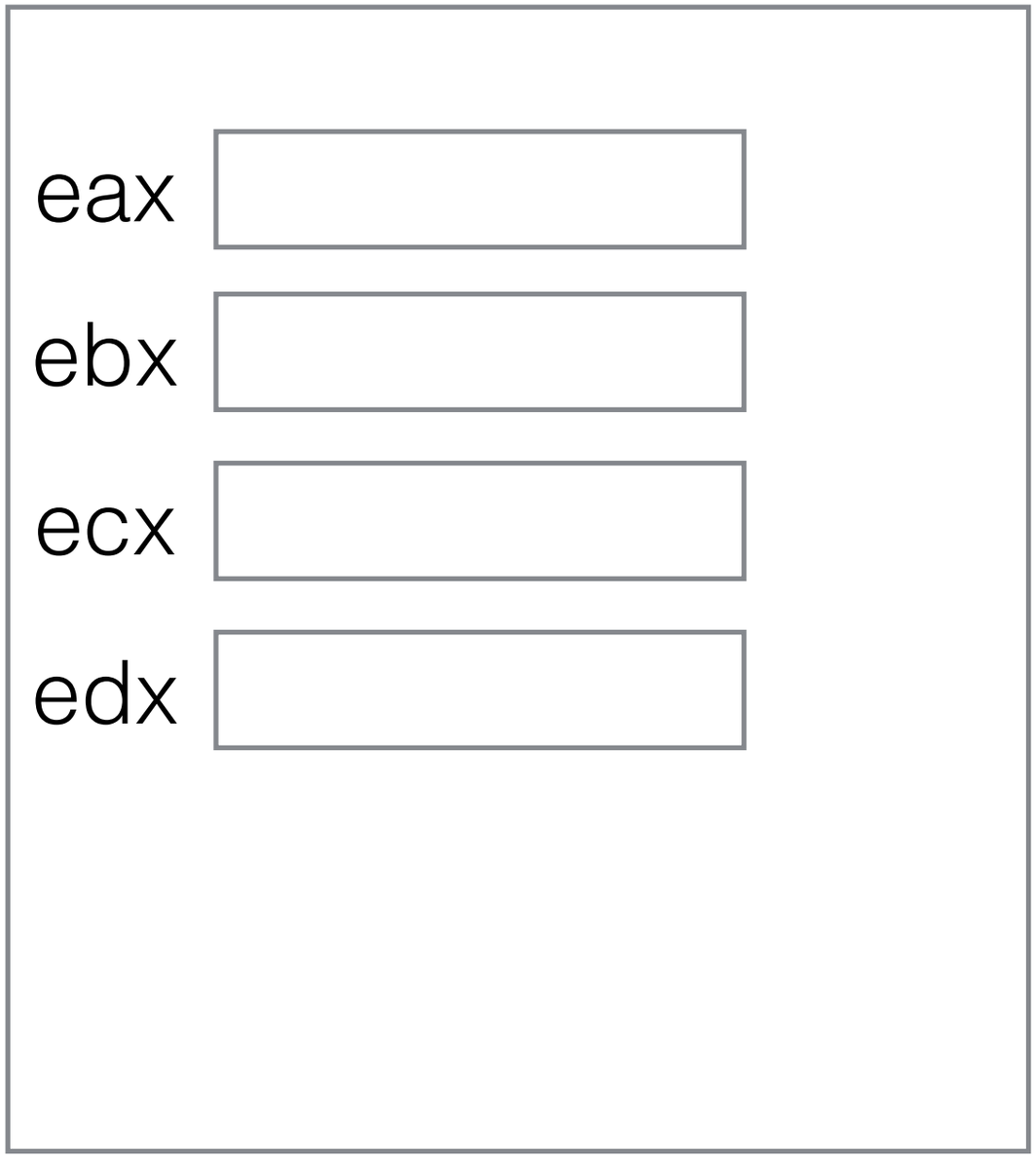
```
        section .data
a       dd      1234


        section .text
        mov     eax, dword[a]
        add     eax, 1

        mov     dword[a], eax
```

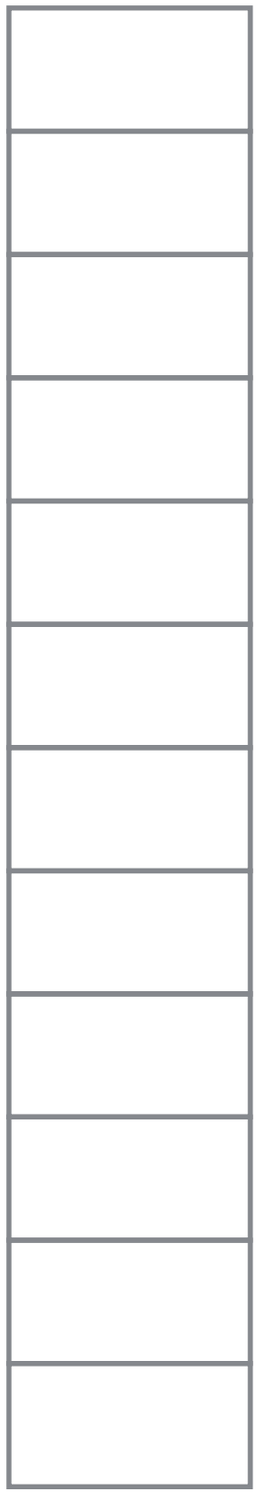




Exercise

```
        section .data
a       dd      1234


        section .text
        add     dword[a], 1
        mov     eax, dword[a]
```





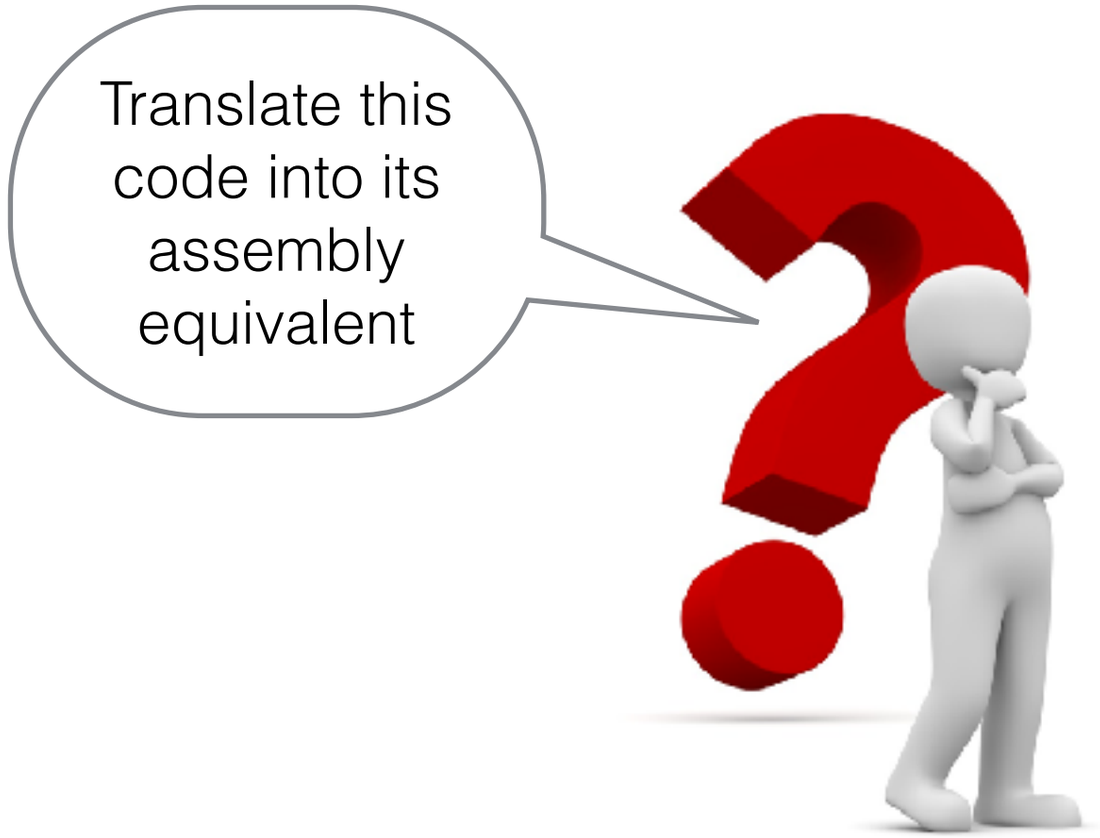
```
int x = 3;
int y = 5;
int sum;

sum = x + y;
```

Exercise

Translate this code into its assembly equivalent

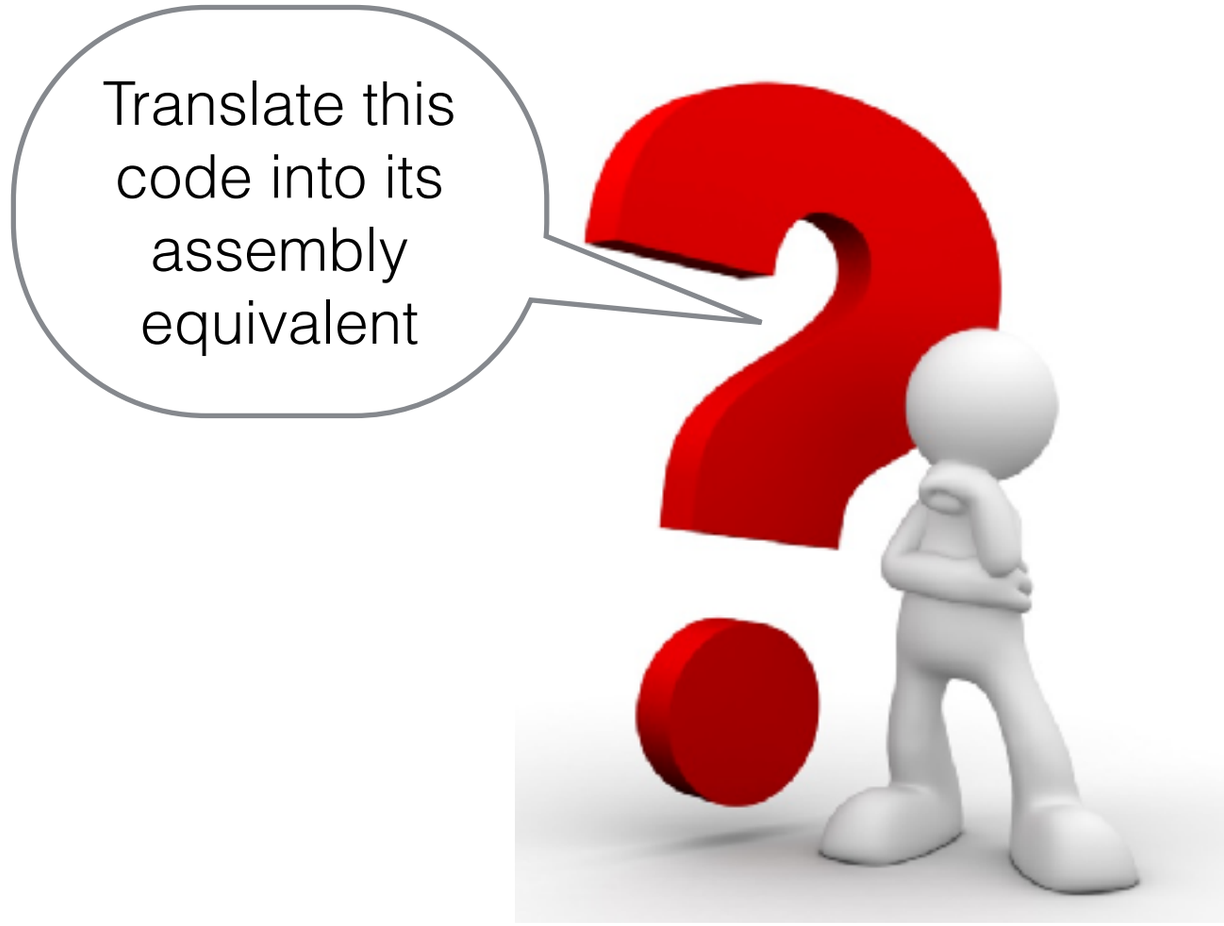
Exercise

```
int x = 3;
int y = 5;
int z = 10;
int sum;

sum = x + y + 2*z;
```



Exercise

```
int x = 3;
int y = 5;
int z = 10;
int sum;

sum = x + 2*y - 3*(z-1);
```



Getting a sense of Speed of Execution

2.8GHz of quad-core processing power and Turbo Boost speeds, which automatically increases the speed of active cores – up to 3.8GHz. Work, play

Getting a sense of Speed of Execution

```
; sum = x + 2*y + 3*(z-1)
mov eax,dword[x]
add eax, dword[y]
add eax, dword[y]
mov ebx, dword[z]
add ebx, -1
mov ecx, ebx
add ebx, ebx
add ebx, ecx
add eax, ebx
mov dword[sum], eax
```

2.8 GHz CPU

2.8 billion cycles/second

1instruction / cycle

cycle =1/(2.8E+9) = 0.35 ns

10 instructions ==> 3.5 ns

can compute **280 million similar equations in 1 second**

We stopped here last time…