CSCI 2330

Previous TODOs

- Assignment 0 tasks before next class:
  - Read Syllabus
  - Play and beat Bowdoin Dungeon Crawler
  - Play Ant (try to make it to at least level 2)
  - Finish vimtutor Lessons 1.1 through 1.4


jeffrey@polaris: ~/Downloads
File Edit View Terminal Tabs Help
jeffrey@polaris:~/Downloads$ python3 dungeon.py

jeffrey@polaris: ~/Downloads

File Edit View Terminal Tabs Help

Bowdoin Dungeon Crawler: h/j/k/l move, . wait, p potion, q quit.

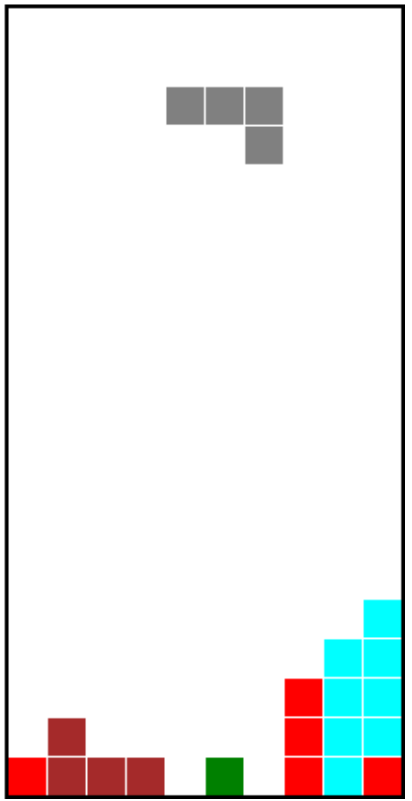
🩸: 24/24 🧪: 2 🕐: 0 🏆: 0

-
-
-
-
-
- You descend into the Bowdoin Dungeon.

Instructions:

- `h`: move left
- `j`: move down
- `k`: rotate
- `l`: move right
- `Shift`+`g`: drop

Level: 3
Lines: 25
Score: 450



How did it go?

- How far did you get in Ant? In vimtutor?
- Was it easy? Frustrating?

A lot of this class will be spent learning new ways of interfacing with computers.

Overview for today

- Data representation
- Different number systems
- Data types
- Binary and logical operations

Binary

Why binary?


centralbureaucracy.tumblr.com
What an awful dream! Ones and zeros everywhere! And I thought I saw a two.
It was just a dream, Bender. There's no such thing as two.

Why binary?

- Computers are composed of digital electronics
- Uses 0s and 1s
- Voltage either on or off
- Robust!
- 0.9 → 1
- 0.2 → 0

Analog media

- VHS tapes
  - Every time you make a copy, the quality gets a little worse
- But DVD, Bluray, *.mp4, etc.
  - Every copy is identical
- “Digitizing” → converting from analog to digital

Is digital always better?

Is digital always better?

Is digital always better?

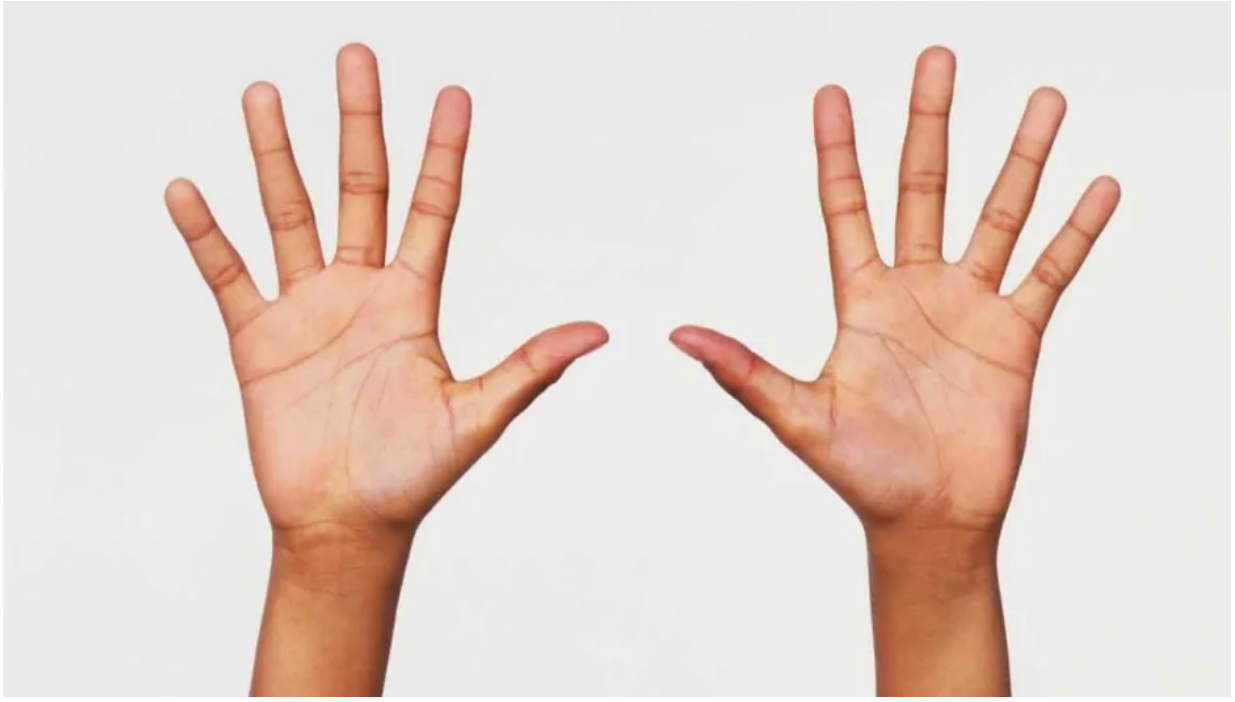
Decimal

- 6789
- $6*10^3 + 7*10^2 + 8*10^1 + 9*10^0$

Why do humans use decimal?

Why do humans use decimal?

- No mathematical reason



Decimal

- 6789
- $6*10^3 + 7*10^2 + 8*10^1 + 9*10^0$

Binary

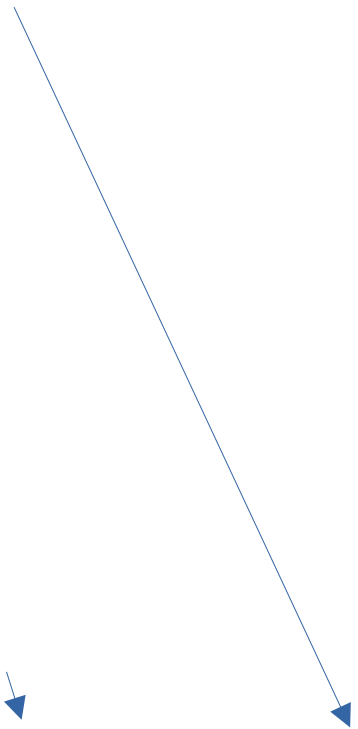
- 1001
- $1*2^3 + 0*2^2 + 0*2^1 + 1*2^0$

Converting from binary → decimal

- 1001
- $1*2^3 + 0*2^2 + 0*2^1 + 1*2^0$
- 1*8 + 0*4 + 0*2 + 1*1
- 9

Converting from decimal → binary

- 154 / 2 = 77 R0
- 77 / 2 = 38 R1
- 38 / 2 = 19 R0
- 19 / 2 = 9 R1
- 9 / 2 = 4 R1
- 4 / 2 = 2 R0
- 2 / 2 = 1 R0
- 1 / 2 = 0 R1
- → 10011010



Converting from decimal → binary

- 154 / 2 = 77 R0
- 77 / 2 = 38 R1
- 38 / 2 = 19 R0
- 19 / 2 = 9 R1
- 9 / 2 = 4 R1
- 4 / 2 = 2 R0
- 2 / 2 = 1 R0
- 1 / 2 = 0 R1
- → 10011010

Dividing by 2
Dividing by 4
Dividing by 8
…

Dividing by high bit

Highest possible value in *n* bits

- $n = 1 \rightarrow 1 \rightarrow 1$

Highest possible value in *n* bits

- $n = 1 \rightarrow 1 \rightarrow 1$
- $n = 2$

Highest possible value in *n* bits

- $n = 1 \rightarrow 1 \rightarrow 1$
- $n = 2 \rightarrow 11 \rightarrow 3$

Highest possible value in *n* bits

- $n = 1 \rightarrow 1 \rightarrow 1$
- $n = 2 \rightarrow 11 \rightarrow 3$
- $n = 3$

Highest possible value in *n* bits

- $n = 1 \rightarrow 1 \rightarrow 1$
- $n = 2 \rightarrow 11 \rightarrow 3$
- $n = 3 \rightarrow 111 \rightarrow 7$

Highest possible value in *n* bits

- $n = 1 \rightarrow 1 \rightarrow 1$
- $n = 2 \rightarrow 11 \rightarrow 3$
- $n = 3 \rightarrow 111 \rightarrow 7$
- $n = 4$

Highest possible value in *n* bits

- $n = 1 \rightarrow 1 \rightarrow 1$
- $n = 2 \rightarrow 11 \rightarrow 3$
- $n = 3 \rightarrow 111 \rightarrow 7$
- $n = 4 \rightarrow 1111 \rightarrow 15$

Highest possible value in *n* bits

- $n = 1 \rightarrow 1 \rightarrow 1$
- $n = 2 \rightarrow 11 \rightarrow 3$
- $n = 3 \rightarrow 111 \rightarrow 7$
- $n = 4 \rightarrow 1111 \rightarrow 15$
- n

Highest possible value in *n* bits

- $n = 1 \rightarrow 1 \rightarrow 1$
- $n = 2 \rightarrow 11 \rightarrow 3$
- $n = 3 \rightarrow 111 \rightarrow 7$
- $n = 4 \rightarrow 1111 \rightarrow 15$
- $n \rightarrow \ldots \rightarrow 2^n\text{-}1$

Highest possible value in *n* bits

- $n = 1 \rightarrow 1 \rightarrow 1$ # we can represent 2 values
- $n = 2 \rightarrow 11 \rightarrow 3$ # we can represent 4 values
- $n = 3 \rightarrow 111 \rightarrow 7$ # we can represent 8 values
- $n = 4 \rightarrow 1111 \rightarrow 15$ # we can represent 16 values
- $n \rightarrow \ldots \rightarrow 2^n\text{-}1$ # we can represent 2^n values
- Zero counts as a value

Bytes

- Computers deal with groups of 8 bits at a time
- 8 bits == *byte*
- You have 32 giga*bytes* of memory
- Your file is 700 *bytes*
- You cannot have a file that is 5 bits
- The smallest (non-empty) file you can have is 1 *byte*

Why *bytes*? Why not bits?

- Bits are usually more precision than we need
- Makes everything slower

Why *8*?

- A power of 2 (faster hardware)
- How many values can a byte store?

Why *8*?

- A power of 2 (faster hardware)
- How many values can a byte store? *256*
- 26 lowercase letters, 26 uppercase, some numbers, some punctuation, etc.

Java integer types

- 8 bits →
- 16 bits →
- 32 bits →
- 64 bits →

Java integer types

- 8 bits → byte
- 16 bits → short (also char)
- 32 bits → int
- 64 bits → long

C integer types

- 8 bits → char
- 16 bits → short
- 32 bits → int
- 64 bits → long / long long
- *on the machines we will be working on

Hexadecimal

- Base 16!?
- Base 2: 01
- Base 10: 0123456789
- Base 16: 0123456789abcdef

What do we call a “digit”?

- Base 2: bit
- Base 10: digit
- Base 16: nibble

What do we call a “digit”?

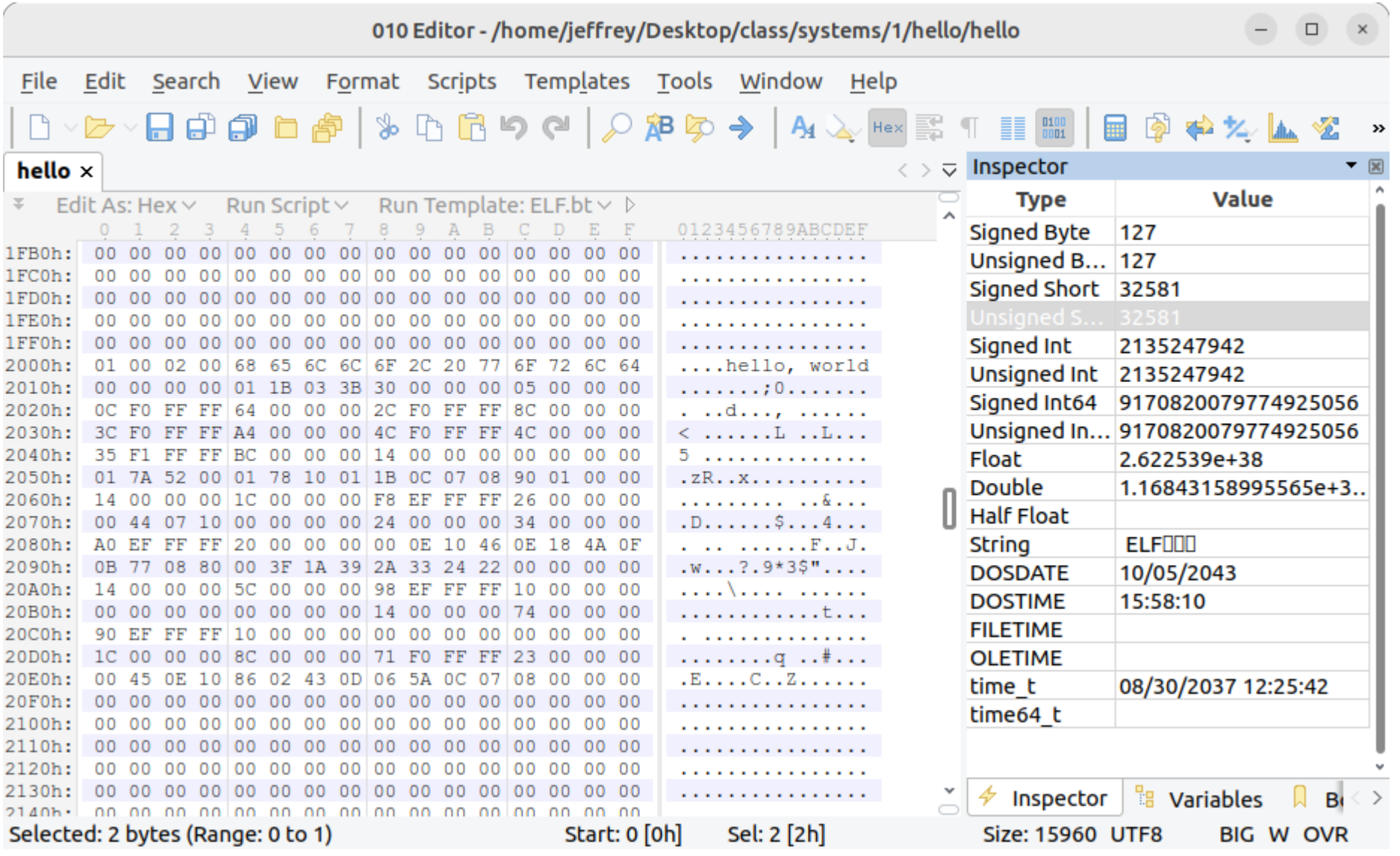
- Base 2: bit
- Base 10: digit
- Base 16: nibble
- Nibble is 4 bits (half a byte)

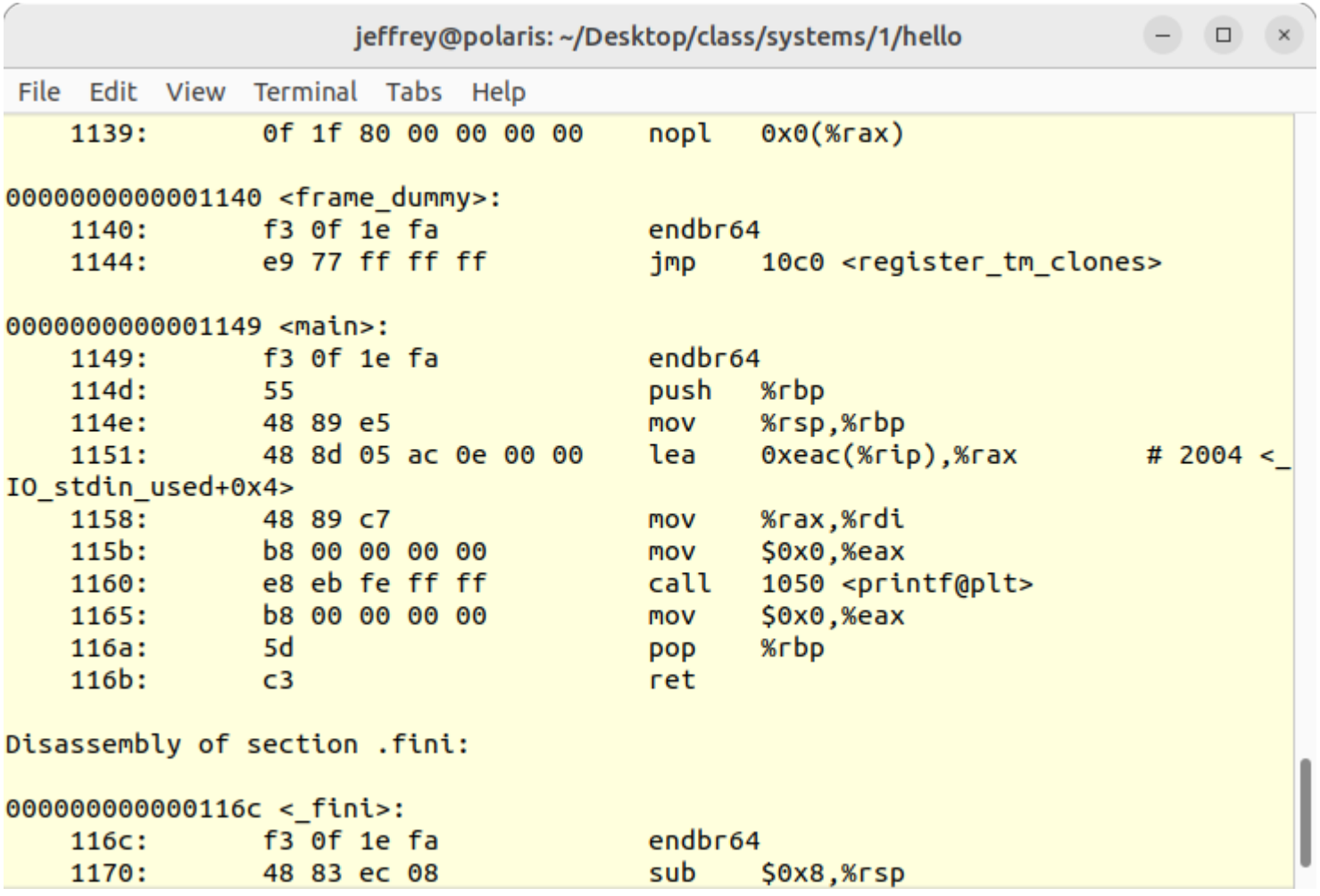
How should we represent a byte?

- Binary: 10011001
- Decimal: 153
- Hex: 99

How should we represent a byte?

- Binary: 10011001
- Decimal: 153
- Hex: 99


010 Editor - /home/jeffrey/Desktop/class/systems/1/hello/hello
File Edit Search View Format Scripts Templates Tools Window Help
hello
Edit As: Hex Run Script Run Template: ELF.bt
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
1FB0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
1FC0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
1FD0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
1FE0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
1FF0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
2000h: 01 00 02 00 68 65 6C 6C 6F 2C 20 77 6F 72 6C 64 ....hello, world
2010h: 00 00 00 00 01 1B 03 3B 30 00 00 00 05 00 00 00 ....;0.......
2020h: 0C F0 FF FF 64 00 00 00 2C F0 FF FF 8C 00 00 00 . ..d..., ...
2030h: 3C F0 FF FF A4 00 00 00 4C F0 FF FF 4C 00 00 00 < ......L ..L...
2040h: 35 F1 FF FF BC 00 00 00 14 00 00 00 00 00 00 00 5 ....
2050h: 01 7A 52 00 01 78 10 01 1B 0C 07 08 90 01 00 00 .zR..x..........
2060h: 14 00 00 00 1C 00 00 00 F8 EF FF FF 26 00 00 00 ....&...
2070h: 00 44 07 10 00 00 00 00 24 00 00 00 34 00 00 00 .D......$...4...
2080h: A0 EF FF FF 20 00 00 00 00 0E 10 46 0E 18 4A 0F . .. .F..J.
2090h: 0B 77 08 80 00 3F 1A 39 2A 33 24 22 00 00 00 00 .w...?.9*3$"....
20A0h: 14 00 00 00 5C 00 00 00 98 EF FF FF 10 00 00 00 ....\.... ......
20B0h: 00 00 00 00 00 00 00 00 14 00 00 00 74 00 00 00 ....t...
20C0h: 90 EF FF FF 10 00 00 00 00 00 00 00 00 00 00 00 . ...
20D0h: 1C 00 00 00 8C 00 00 00 71 F0 FF FF 23 00 00 00 ....q ..#...
20E0h: 00 45 0E 10 86 02 43 0D 06 5A 0C 07 08 00 00 00 .E....C..Z......
20F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
2100h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
2110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
2120h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
2130h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
Inspector
Type Value
Signed Byte 127
Unsigned B... 127
Signed Short 32581
Unsigned S... 32581
Signed Int 2135247942
Unsigned Int 2135247942
Signed Int64 9170820079774925056
Unsigned In... 9170820079774925056
Float 2.622539e+38
Double 1.16843158995565e+3..
Half Float
String ELF□□□
DOSDATE 10/05/2043
DOSTIME 15:58:10
FILETIME
OLETIME
time_t 08/30/2037 12:25:42
time64_t
Inspector Variables
Selected: 2 bytes (Range: 0 to 1) Start: 0 [0h] Sel: 2 [2h] Size: 15960 UTF8 BIG W OVR


jeffrey@polaris: ~/Desktop/class/systems/1/hello
File Edit View Terminal Tabs Help
1139: 0f 1f 80 00 00 00 00 nopl 0x0(%rax)
0000000000001140 <frame_dummy>:
1140: f3 0f 1e fa endbr64
1144: e9 77 ff ff ff jmp 10c0 <register_tm_clones>
0000000000001149 <main>:
1149: f3 0f 1e fa endbr64
114d: 55 push %rbp
114e: 48 89 e5 mov %rsp,%rbp
1151: 48 8d 05 ac 0e 00 00 lea 0xeac(%rip),%rax # 2004 <_
IO_stdin_used+0x4>
1158: 48 89 c7 mov %rax,%rdi
115b: b8 00 00 00 00 mov $0x0,%eax
1160: e8 eb fe ff ff call 1050 <printf@plt>
1165: b8 00 00 00 00 mov $0x0,%eax
116a: 5d pop %rbp
116b: c3 ret
Disassembly of section .fini:
000000000000116c <_fini>:
116c: f3 0f 1e fa endbr64
1170: 48 83 ec 08 sub $0x8,%rsp

Converting from hex → decimal

- 89ab
- $8*16^3 + 9*16^2 + \sout{a}10*16^1 + \sout{b}11*16^0$
- $8*4096 + 9*256 + 10*16 + 11*1$
- 35243

Converting from decimal → hex

- 154 / 16 = 9 R10
- 9 / 16 = 0 R9

Converting from decimal → hex

- 154 / 16 = 9 R~~10~~A
- 9 / 16 = 0 R9

Converting from decimal → hex

- 154 / 16 = 9 R~~10~~A
- 9 / 16 = 0 R9
- → 9A

Is this binary, decimal, or hex?

- 100

Disambiguate

- $100_2 = 4_{10}$
- 100_{10}
- $100_{16} = 256_{10}$

Disambiguate

- 0b100 = 4
- 100
- 0x100 = 256

To decimal

```
>>> 0b100
4
>>> 100
100
>>> 0x100
256
```

From decimal

```
>>> bin(4)
'0b100'
>>> str(100)
'100'
>>> hex(256)
'0x100'
```

Hex → binary, binary → hex

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Hex → binary

- 0xa → 0b???

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Hex → binary

- 0xa → 0b1010

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Hex → binary

- 0xa → 0b1010
- 0xab → 0b???

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Hex → binary

- 0xa → 0b1010
- 0xab → 0b10101011

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Hex → binary

- 0xa → 0b1010
- 0xab → 0b10101011
- 0xabc → 0b???

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Hex → binary

- 0xa → 0b1010
- 0xab → 0b10101011
- 0xabc → 0b101010111100

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Binary → hex

- 0b1011111011101111 → 0x???

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Binary → hex

- 0b1011111011101111 → 0x???
- 0b1011 → 0x?
- 0b1110 → 0x?
- 0b1110 → 0x?
- 0b1111 → 0x?

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Binary → hex

- 0b1011111011101111 → 0x???
- 0b1011 → 0xb
- 0b1110 → 0xe
- 0b1110 → 0xe
- 0b1111 → 0xf

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Binary → hex

- 0b1011111011101111 → 0x???
- 0b1011 → 0xb
- 0b1110 → 0xe
- 0b1110 → 0xe
- 0b1111 → 0xf
- 0b1011111011101111 → 0xbeef

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Byte representation

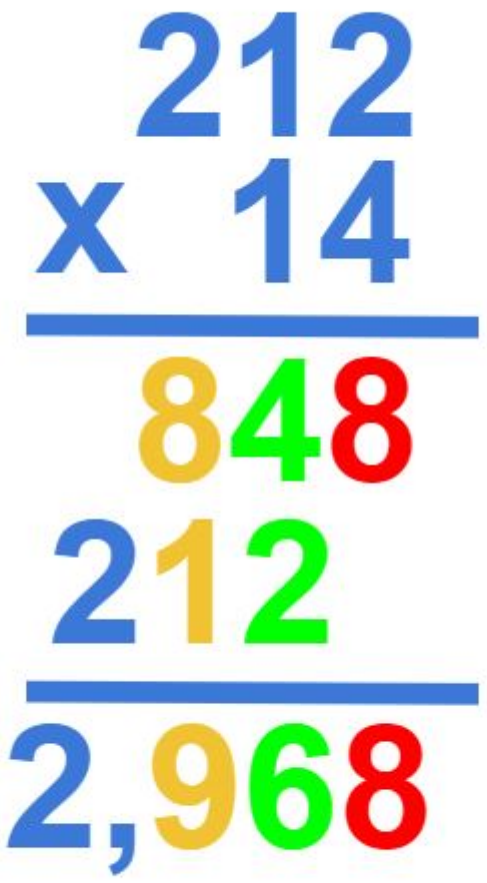
- We use two hexadecimal nibbles (0xe9)
- Why not use a base 256 number system?
- We would only need one “digit” that way…

First in-class exercise!

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Binary arithmetic

- We will be learning some old tricks and some new tricks here!
- Revisiting grade school arithmetic: +, *
- New stuff: ^, &, |, !, ~


212
x 14
848
212
2,968

Binary arithmetic

```
  0110
+ 0100
```

Binary arithmetic

```
  0110
+ 0100
------
     0
```

Binary arithmetic

```
  0110
+ 0100
    10
```

Binary arithmetic

```
  1
  0110
+ 0100
   010
```

Binary arithmetic

```
  0110
+ 0100
  1010
```

Binary arithmetic

$$
\begin{array}{rl}
 & 0110 = 6_{10} \\
+ & \underline{0100} = 4_{10} \\
 & 1010 = 10_{10}
\end{array}
$$

Binary arithmetic

```
  110
*  11
```

Binary arithmetic

```
   110
*   11
  ----
     0
```

Binary arithmetic

```
  110
*  11
----
  10
```

Binary arithmetic

```
  110
*  11
-----
  110
```

Binary arithmetic

```
  110
*  11
-----
  110
    0
```

Binary arithmetic

```
  110
*  11
-----
  110
   00
```

Binary arithmetic

```
  110
*  11
-----
  110
  100
```

Binary arithmetic

```
  110
*  11
  110
 1100
```

Binary arithmetic

```
  110
*  11
----

  110
 1100
----
```

Binary arithmetic

```
  110
*  11
----

  110
 1100
----
    0
```

Binary arithmetic

```
  110
*  11
----

  110
 1100
----
   10
```

Binary arithmetic

```
  110
*  11
 ----
 1
  110
 1100
 ----
  010
```

Binary arithmetic

```
  110
*  11
11
  110
 1100
 0010
```

Binary arithmetic

```
  110
*  11
11
  110
 1100
10010
```

Binary arithmetic

```
   110 =  6₁₀
*   11 =  3₁₀
11
  110
 1100
10010 = 18₁₀
```

Binary and logical operators

`&`	bitwise and
`|`	bitwise or
`^`	bitwise xor
`~`	bitwise negation
`&&`	logical and
`||`	logical or
`!`	logical negation

Truth tables

&	0	1
0	0	0
1	0	1

|	0	1
0	0	1
1	1	1

^	0	1
0	0	1
1	1	0

~: a unary operator that flips the bit

Binary operations

```
  1100
& 1010
  ????
```

Binary operations

```
  1100
& 1010
  1000
```

Binary operations

```
  1100
| 1010
  ????
```

Binary operations

```
  1100
| 1010
  1110
```

Binary operations

```
  1100
^ 1010
  ????
```

Binary operations

```
  1100
^ 1010
  0110
```

Binary operations

```
~1100 = ????
```

Binary operations

```
~1100 = 0011
```

TODO

- Assignment 0 is assigned (due September 11)
- Assignment 1 will be assigned end of next week