lecture 1

representing integers in binary

Base 10

$$238 \equiv 2 \cdot 10^2 + 3 \cdot 10^1 + 8 \cdot 10^0$$

Base 2

$$11010 \equiv 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$$

$$= 16 + 8 + 2 = 26$$

Counting in Binary

decimal	binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
⋮	⋮

To convert from binary to decimal, you need to know the powers of 2.

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
⋮	⋮

memorize

How can we convert m from decimal to binary?

- idea 1: find the biggest power of 2 less than m and subtract it. Iterate.
- idea 2: ?

Use familiar idea from base 10:

$$m = (m/10) \times 10 + m \,\%\, 10$$

$$238 = 230 + 8$$

Same idea works in base 2

$$m = (m/2) \times 2 + m \% 2$$

Example:

$m = (10011)_{two}$

$m/2 = 1001$

$m \% 2 = 1$

$$m = \sum_{i=0}^{n-1} b_i \, 2^i$$

$$= (b_{n-1} b_{n-2} \cdots b_2 b_1 b_0)_{two}$$

$$m/2 = \sum_{i=1}^{n-1} b_i \, 2^{i-1}$$

$$= (b_{n-1} b_{n-2} \cdots b_2 b_1)_{two}$$

$$m \% 2 = b_0$$

Algorithm: given m in decimal, convert it to binary

```
i ← 0
while   m > 0 {
    b_i  ←  m % 2
    m    ←  m / 2
    i    ←  i + 1
}
```

Example $m = 241 = (11110001)_{two}$

m	b_i
241	
$= 120 \times 2 +$	1
60	0
30	0
15	0
7	1
3	1
1	1
0	1

Addition (base 10)

```
   1 0 1      ← carry
   2 3 4 3
+  5 8 1 9
   -------
   8 1 6 2
```

You needed to memorize single digit sums to do it.

Addition (base 2)

```
 0 0 1 1 0 1 0
0 0 0 1 1 0 1 0   ← 26
0 0 0 1 1 0 1 1   ← 27
---------------
0 0 1 1 0 1 0 1   ← 53
```

Subtraction (base 10)

$$\begin{array}{r} 2343 \\ -\ 5819 \\ \hline \end{array}$$

$a - b$

$= a + (-b)$

Consider another way of representing negative numbers (decimal, then binary)

Recall: car odometer

$$\begin{array}{r} 0000999 \\ +\ 0000001 \\ \hline 0001000 \end{array}$$

$$\begin{array}{r} 999999 \\ +\ 000001 \\ \hline 000000 \end{array}$$

$\Rightarrow 999999 \equiv -1$

$$\begin{array}{r} 328769 \\ +\ 671231 \\ \hline 000000 \end{array}$$

$\equiv -328769$

Negative numbers in binary

$$\begin{array}{r} 00011010 \\ \hline 00000000 \end{array}$$

"two's complement"

```
  00011010
  11100101   ← invert bits
+        1   ← add 1
----------
  00000000
```

$26 + (-26) = 0$

Tricky cases

```
  00000000
  11111111
+        1
----------
  00000000
```

Tricky cases

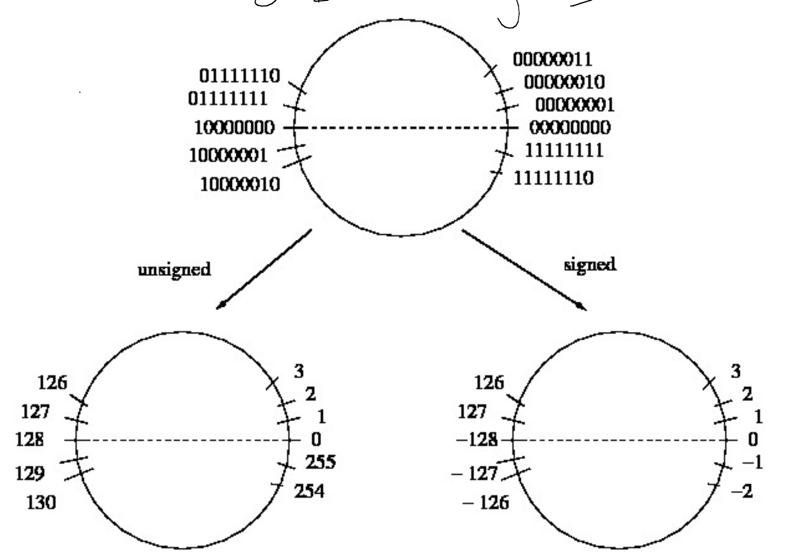
```
  10000000
  01111111
+        1
----------
  00000000
```

binary	signed	unsigned
00000000	0	0
00000001	1	1
00000010	2	2
00000011	3	3
00000100	4	4
⋮	⋮	⋮
01111111	127	127
10000000	−128	128
⋮	⋮	
11111111	−1	255

Signed integers

```
                          00000000
                          0
                          0
most                  0   ⋮            positive
significant               0
bit is                    0
                          --------
                      1   1
                          1            negative
                          1
                          ⋮
                          11111111
```

8 bit integers



n bits $\Rightarrow$ 2^n integers

unsigned

$0, 1, \ldots \ldots, 2^n - 1$

signed

$-2^{n-1}, \ldots, 0 \ldots, 2^{n-1} - 1$

Hexadecimal (base 16)

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
a	1010
b	1011
c	1100
d	1101
e	1110
f	1111

0010 1111 1010 0011

4 f a 3

We write 0x4fa3

or 0X4FA3

101100

0x2c

(not 0xf0)