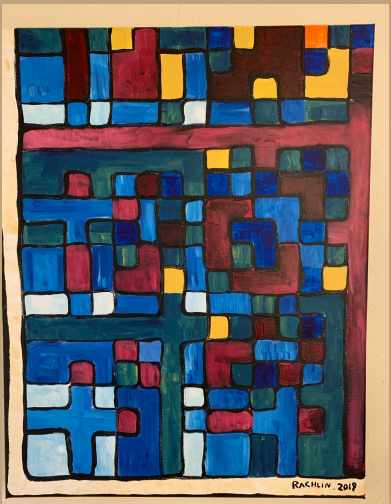
Number Representation

57656c636f6d6520746f20435335303032210a



John Rachlin
Northeastern

Discrete Structures

Binary numbers.

① In computers, everything is represented with 1's and 0's,

- numbers
- characters
- images
- videos
- music
- ⋮
- everything.

These values are physically implemented with transistors which act like tiny switches.

switch	on	off
Logic	True	False
Circuit	+5V	0V
Disk	magnetized	demagnetized
Bits.	1	0

1 BIT = "Binary digIT" = 0 or 1

1 Byte = 8 bits ≈ 1 character of text data (256 possible values)

② Decimal Representation

$$8792 = 8 \times 10^3 + 7 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$$

1000, 100, 10, 1 ← Places = 10^x : $\ldots 10^3, 10^2, 10^1, 10^0$

10 ← Base

Digits : 0 … 9

× 10 : 32 ⇒ 320 (left shift)

$$8792_{10} = 879 \cdot 10 + 2$$
$$= (87 \cdot 10 + 9) \cdot 10 + 2$$
$$= (((8 \cdot 10) + 7) \cdot 10 + 9) \cdot 10 + 2$$

Notes on exponents

$$x^n = \underbrace{x \cdot x \cdot x \cdot \cdots \cdot x}_{n \text{ times}}$$

e.g.

$$10^3 = 10 \cdot 10 \cdot 10 = 1000$$

$$\boxed{x^{-n} = 1/x^n = \left(1/x\right)^n}$$

$$\boxed{x^a x^b = x^{a+b}}$$

$2^2 \cdot 2^3 = 4 \cdot 8 = 32 = 2^5$ ✓

$$\boxed{\left(x^a\right)^b = x^{a \cdot b}}$$

$\left(2^2\right)^3 = 4^3 = 64 = 2^6$ ✓

n	10^n		2^n
-9	$1/10^9$	billionth (nano)	1/512
-6	$1/10^6$	millionth (micro)	1/64
-3	$1/1000$	thousandth (milli)	1/8
-2	$1/100$	hundredth	1/4
-1	$1/10$	tenth	1/2
0	$10^0 = 1$	one	1
1	10	tens	2
2	100	hundreds	4
3	1000	thousands	8
6	1000000	millions	64
9	10^9	billions	512

kb

10 $2^{10} = 1024$ = kilobyte

20 $2^{20} = 1048576 \approx 1$ million — mega byte

30 2^{30} = giga

40 2^{40} = tera

So working the other way, we see "8792" as a sequence of 4 digits. Its value is:

1000	100	10	1
8	7	9	2

$8000 + 700 + 90 + 2 = 8792$

OR:

8

$8 \times 10 + 7 = 80 + 7 = 87$

$87 \times 10 + 9 = 870 + 9 = 879$

$879 \times 10 + 2 = 8790 + 2 = 8792$

③ Binary Representation (Base 2)

Bits: 0, 1

Places: Powers of Two

Binary ⇒ Decimal

$10110_2 =$

2^4	2^3	2^2	2^1	2^0
1	0	1	1	0

"one zero one one zero"

16^s	8^s	4^s	2^s	1^s

$= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 16 + 0 + 4 + 2 + 0$

$= 22_{10}$

We can shift an add as we did with decimal. Now a shift left multiplies by 2 and we add 1 or 0

1 = 1

10 = 1 × 2 + 0 = 2

101 = 2 × 2 + 1 = 5

1011 = 5 × 2 + 1 = 11

$10110 = 11 \times 2 + 0 = 22_{10}$ ✓

④ Decimal → Binary (Two methods)

41_{10}:

64	32	16	8	4	2	1
	1	0	1	0	0	1

$41 - 32 = 9$

$9 - 8 = 1$

$1 - 1 = 0$ (Done)

$41_{10} = 101001_2$

$1 \to 2 \to 5 \to 10 \to 20 \to 41$

27_{10}:

32	16	8	4	2	1
	1	1	0	1	1

$27 - 16 = 11$
$11 - 8 = 3$
$3 - 2 = 1$
$1 - 1 = 0$ ✓

$27_{10} = 11011_2$

24_{10}

32	16	8	4	2	1
	1	1	0	0	0

$24 - 16 = 8$
$8 - 8 = 0$

$32_{10} = 11000_2$

Another Approach: odd: write down 1, compute $(x-1) \div 2$
even: write down 0, compute $x \div 2$

41_{10}:

41	1 ← lowest significant digit
20	0
10	0
5	1
2	0
1	1 ← highest significant digit

$\Rightarrow 101001_2$

27_{10}

13	1
6	0
3	1
1	1

$= 11011_2$

24_{10}

12	0
6	0
3	1
1	1

$= 11000_2$

⑤ Binary to Decimal.

method 1: add powers of 2 together

$$10110_2 \Rightarrow 2^4 + 2^2 + 2^1$$
$$= 16 + 4 + 2 = 22_{10}$$

method 2: multiply by 2 and add 0 or 1 working left to right.

MSB	1	1 = 1
	0	1 x 2 + 0 = 2
	1	2 x 2 + 1 = 5
	1	5 x 2 + 1 = 11
LSB	0	11 x 2 + 0 = 22

A bigger example:

$1101001_2 = ?$

64	32	16	8	4	2	1
1	1	0	1	0	0	1

$\Rightarrow 64 + 32 + 8 + 1 = 105$

1	1 = 1
1	1 x 2 + 1 = 3
0	3 x 2 = 6
1	6 x 2 + 1 = 13
0	13 x 2 = 26
0	26 x 2 = 52
1	52 x 2 + 1 = 105_{10}

$1010011_2 = ?$

1	1
0	2
1	5
0	10
0	20
1	41
1	83

$= 83_{10}$

(6)

Addition and Subtraction

$$\begin{array}{r} {\scriptstyle 1}\ {\scriptstyle 1} \\ 22 \\ +\ 29 \\ \hline 51 \end{array} \Rightarrow \begin{array}{r} {\scriptstyle 1}\ {\scriptstyle 1} \\ 10110 \\ +\ 11101 \\ \hline 110011 \end{array} = \begin{array}{r} {\scriptstyle 1} \\ 16 \\ =\ 1D \\ \hline = 33_{16} \end{array}$$

$$\begin{array}{r} {\scriptstyle 1}\ {\scriptstyle 1} \\ 26 \\ -\ 7 \\ \hline 19 \end{array} \Rightarrow \begin{array}{r} {\scriptstyle 1}\ {\scriptstyle 1}\ {\scriptstyle 0} \\ 1\ 1\ 0\ 1\ 0 \\ -\ 0\ 0\ 1\ 1\ 1 \\ \hline 1\ 0\ 0\ 1\ 1 \end{array}$$

(7) Things every computer scientist should know.

a) Counting:

Range:

n digits: 0 to $10^n - 1$

n bits: 0 to $2^n - 1$

"Byte" = 8 bits

∴ Range = 0 to 255

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

b) Powers of Two

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

2^{10}	Kilo	= 1024	≈	1 thousand
2^{20}	Mega	= 1,048,576	≈	1 million
2^{30}	Giga		≈	1 billion
2^{40}	Tera		≈	1 trillion

2^{38} bytes $= 2^8 \cdot 2^{30} = 256$ Gigabytes

(8) Octal : Base-8 (Digits : 0 to 7)

53_{10} :

8^2	8^1	8^0
64	8	1
	6	5

$53 - 6 \cdot 8 = 5$

$5 - 5$

$53_{10} = 65_8$ "six five"

Notice that each of the octal "digits" can be represented by 3 bits:

000	001	010	.....	110	111
0	1	2	....	6	7

So we could convert 1st to binary then immediatly get our octal result.

53_{10}	1
26	0
13	1
6	0
3	1
1	1

$= \underbrace{110}_{6}\underbrace{101}_{5}{}_2$

$6\ 5_8$

(9) Hexadecimal

Base: 10	2	16
Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Digits 0-9 0-1 0-F

53_{10}: 16^2 16^1 16^0

256 16 1

3 5

$53 = 3 \cdot 16 + 5$

$= 35_{16} = \underbrace{00110101}_{3}\ \underbrace{}_{5}$

47_{10}: $2 \cdot 16 + 15 = 2F_{16}$

$= 00101111_{2}$

Why can we group 4 bits to form hexadecimal digits?

Consider:

$$1011\ 0010\ 0111$$

$$= 1011\ 0000\ 0000 + 0010\ 0000 + 0111$$

1011 shifted 8 times + 0010 shifted 4 times + 0111 unshifted

$$(1011) \times 2^8 + (0010) \times 2^4 + (0111) \times 2^0$$

$$11 \times 16^2 + 2 \times 16 + 7$$

$$= (B\ 2\ 7)_{16}$$

- The same reasoning applies for grouping 3 bits to form an octal #
- Pad MSB if necessary.

  e.g. $(111010)_2 \equiv (0011\ 1010)_2 \equiv 3A_{16}$

Summary.

Base 10: (Decimal)

$$8702_{10} = 8\times10^3 + 7\times10^2 + 0\times10^1 + 2\times10^0$$
$$= 8\times1000 + 7\times100 + 0\times10 + 2\times1$$
$$= 8000 + 700 + 0 + 2$$

Digits: 0...9

Max (n digits) $10^n - 1$

Shift Left $\equiv \times 10$

$8 \rightarrow 80$

Base 2: (Binary)

$$10110_2 = 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$$
$$= 1\times16 + 0\times8 + 1\times4 + 1\times2 + 0\times1$$
$$= 16 + 0 + 4 + 2 + 0$$
$$= 22_{10}$$

Digits: 0, 1 ("Bits")

Max (n bits): $2^n - 1$

Shift Left $\equiv \times 2$

$101_5 \rightarrow 1010_{10}$

Base 8 (Octal)

$$173_8 = 1\times8^2 + 7\times8^1 + 3\times8^0$$
$$= 64 + 56 + 3$$
$$= 123_{10}$$

Digits: 0..7

Max: $8^n - 1$

Shift Left $\equiv \times 8$

$31_8 \Rightarrow 310_8$

$25_{10} \Rightarrow 192+8+0 = 200_{10}$

Base 16 (Hexadecimal)

$$2A6_{16} = 2\times16^2 + 10\times16^1 + 6\times16^0$$
$$= 512 + 160 + 6$$
$$= 678_{10}$$

Digits: 0..9, A..F

Max: $16^n - 1$

Shift Left $\equiv \times 16$

$14_{16} \Rightarrow 140_{16}$

$20_{10} \Rightarrow 256+64 = 320_{10}$

Review: Decimal → Binary Conversion
(and a funny property of binary #'s)

Consider $16 \times 11 = 165_{10}$

Method 1: Find Powers of 2 that sum to 165_{10}

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1
0								

(10) Representing negative numbers.

a) signed magnitude (1940's)

	sign	4	2	1
+5	0	1	0	1
-5	1	1	0	1

↑↑ sign magnitude

Range

+7	0111
-7	1111

15 values

But with 4 bits I can represent $2^4 = 16$ values so one is wasted:

+0	0000
-0	1000

Also I can't add in the usual way:

$$\begin{array}{r} 0101 \\ +\ 1101 \\ \hline 10010 \end{array} \neq +0 \text{ or } \text{"}-0\text{"}$$

b) One's Complement (4 bits)

Pos #: Standard Binary

Neg #: Take Positive value And flip all bits. MSB is sign

$-5 \Rightarrow 0101 \Rightarrow 1010$

$-7 \Rightarrow 0111 \Rightarrow 1000$

b) continued

Two representations of zero

0000 "+Ø"
1111 "−Ø"

```
      7        1
  + −3         0111
  ----         1100
             --------
              1|0011      (not 0100 = 4)
         overflow  3
```

c) Two's - Complement

Pos # : Standard Binary — max $2^{n-1}-1$.

Neg # : Start w/ magnitude in binary
Flip Bits
Add 1

+5 : 0101

−5 : 0101 ⇒ 1010 ⇒ 1011

0 : 0000 ⇒ 1111 ⇒ 1|0000 (ignore the leading 1)

4 : 0100
−3 : 0011 ⇒ 1100 ⇒ 1101

```
   0100
   1101
  ------
 1|0001     (as expected)
ignore  1
```

$$\begin{array}{r} 3 \\ -4 \\ \hline -1 \end{array}$$

0011
0100 ⇒ 1011 ⇒ 1100 add 1 flip

$$\begin{array}{r} 0011 \\ 1100 \\ \hline 1111 \end{array}$$ ⇐ 1110 ⇐ 0001

"−1"

So we can do subtraction by re using addition circuitry which is much simpler!

(11) Converting 2's-Comp to Decimal

Is MSB a ∅ or a 1?

- ∅: Do normal conversion to decimal
- 1: Number is negative
  - a) Convert to Positive binary
  - b) Convert to Decimal as usual
  - c) Add a "−" sign in front

Example: 1110_2

$$\begin{array}{r} 1110 \\ -\ 0001 \\ \hline 1101 \end{array}$$

flip 0010 = 2

add sign ⇒ −2

OR

Flip and add 1 again!

1110	start
0001	flip
0010	+1

⇓

2

add sign

−2

Why can we flip bits and add 1 to reverse the sign (pos → neg or neg → pos)?

Flipping bits (0→1, 1→0) is the same as subtracting from all 1s

For example: $5 = 0101_2$

$$\begin{array}{rr} & 1111 \\ - & 0101 \\ \hline & 1010 \\ + & 0001 \\ \hline & 1011 \end{array}$$

(flip bits) (add +1)

In two's complement 1011 is −5.

But we also recognize 1011 as 11_{10} in standard binary and $16 - 5 = 11$

So, in general the n-bit 2^s-complement of X is

$$2^n - X$$

Flipping and adding 1:

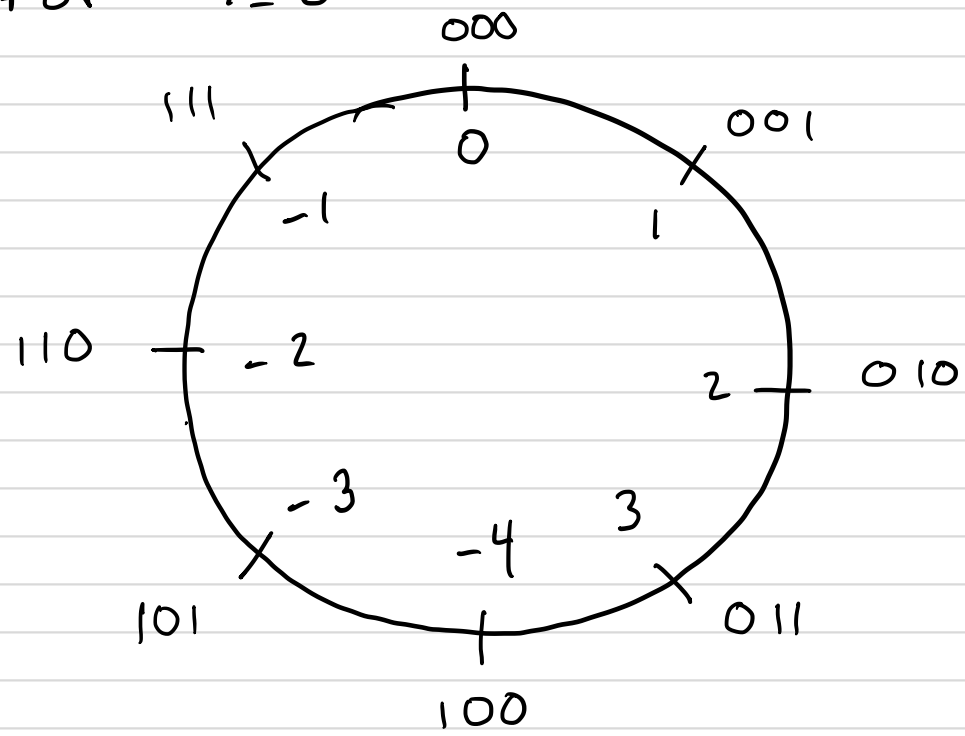
$$\underbrace{\underbrace{2^n - 1}_{n\ 1^s} - X}_{\text{flip}} \underbrace{+1}_{\text{add one}} = 2^n - X \quad \text{(as expected)}$$

Repeating Process: $2^n - 1 - (2^n - X) + 1$

$$= 2^n - 1 - 2^n + X + 1 = X$$

Another way of looking at 2's-complement is to recognize that if we have a fixed # of bits, the numbers wrap around in a circle

For $n = 3$:



min: -2^{n-1}

max: $+2^{n-1} - 1$

$$\begin{array}{rr} 011 & (+3) \\ +\ 010 & (+2) \\ \hline 101 & (-3) \end{array}$$

pos + pos ⇒ neg (overflow)

$$\begin{array}{rr} 101 & (-3) \\ 110 & (-2) \\ \hline 1\,|\,011 & (+3) \end{array}$$

neg + neg ⇒ pos (overflow)

We'll return to numbers on a circle (instead of a line) when we discuss modular arithmetic.