Data Storage

National Chiao Tung University
Chun-Jen Tsai
2/24/2012

Bits and Their Meaning

- First, we must consider how information can be stored inside computers
  - By “Information,” we mean numbers, text, images, sound, video, …, etc.
- For today’s computers, information is encoded as patterns of 0’s and 1’s called bits (BInary digiTs)
  - The reason we use only two symbols (0 and 1) for information encoding is because it’s *simple*, not because it’s powerful.
  - *Non-binary computers* made of multi-level electrical components are possible but not popular yet.

Digital Data Representation

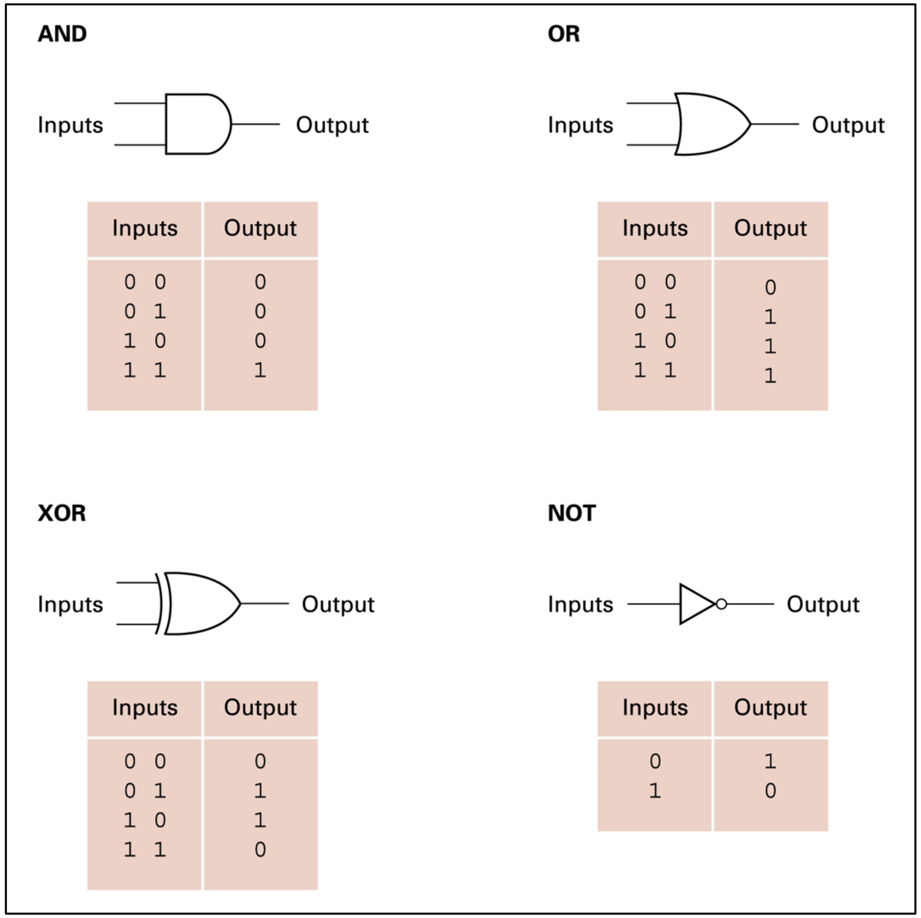
- Some possible meanings for a single bit
  - Numeric value (1 or 0)
  - Boolean value (true or false)
  - Voltage (high or low)
- A bit can only represent one of two values, for more values, we need a long string of bits to represent them

Boolean Operations

- Human beings are made of cells;
  Computers are made of small devices that can compute Boolean functions extremely fast
  - Boolean operations are operations that manipulate one or more 1/0 (or true/false) values
- Some examples of Boolean operations:
  - AND: $\begin{array}{r} 0 \\ \text{AND}\ \ 0 \\ \hline 0 \end{array}$ $\quad\begin{array}{r} 0 \\ \text{AND}\ \ 1 \\ \hline 0 \end{array}$ $\quad\begin{array}{r} 1 \\ \text{AND}\ \ 0 \\ \hline 0 \end{array}$ $\quad\begin{array}{r} 1 \\ \text{AND}\ \ 1 \\ \hline 1 \end{array}$
  - OR: $\begin{array}{r} 0 \\ \text{OR}\ \ 0 \\ \hline 0 \end{array}$ $\quad\begin{array}{r} 0 \\ \text{OR}\ \ 1 \\ \hline 1 \end{array}$ $\quad\begin{array}{r} 1 \\ \text{OR}\ \ 0 \\ \hline 1 \end{array}$ $\quad\begin{array}{r} 1 \\ \text{OR}\ \ 1 \\ \hline 1 \end{array}$
  - XOR: $\begin{array}{r} 0 \\ \text{XOR}\ \ 0 \\ \hline 0 \end{array}$ $\quad\begin{array}{r} 0 \\ \text{XOR}\ \ 1 \\ \hline 1 \end{array}$ $\quad\begin{array}{r} 1 \\ \text{XOR}\ \ 0 \\ \hline 1 \end{array}$ $\quad\begin{array}{r} 1 \\ \text{XOR}\ \ 1 \\ \hline 0 \end{array}$
  - NOT: $\begin{array}{r} \text{NOT}\ \ 0 \\ \hline 1 \end{array}$ $\quad\begin{array}{r} \text{NOT}\ \ 1 \\ \hline 0 \end{array}$

Gates

- The small devices that compute Boolean operations are called “gates”
  - Often implemented as electronic circuits
  - The building blocks from which computers are constructed



AND

Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

OR

Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1

XOR

Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

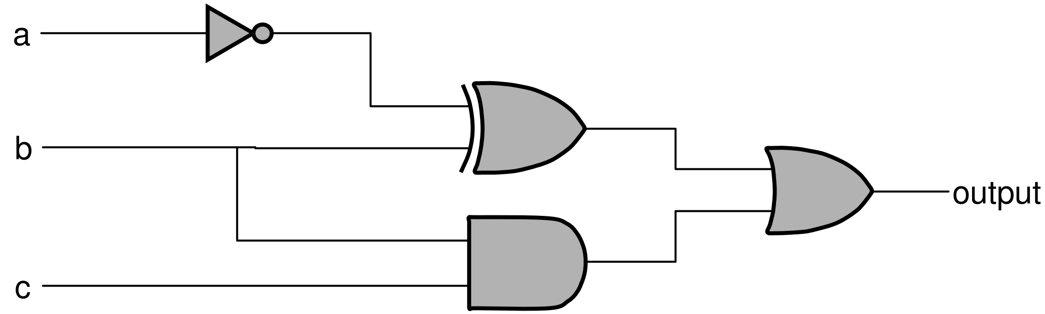
NOT

Inputs	Output
0	1
1	0

Example of Simple Circuit

- A digital circuit's operation can be summarized by a *truth table*

Circuit

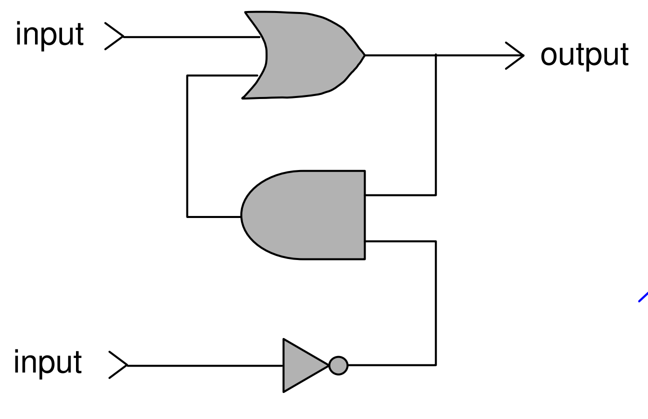


Truth Table

Input a, b, c	Output
000	1
001	1
010	0
011	1
100	0
101	0
110	1
111	1

Flip-Flops

- In addition to performing Boolean operations, gates can also be used to store information
- A flip-flop is built from gates that stores one bit of data
  - Has an input line which sets its stored value to 1
  - Has an input line which sets its stored value to 0
  - When both inputs = 0, most recently stored value is preserved

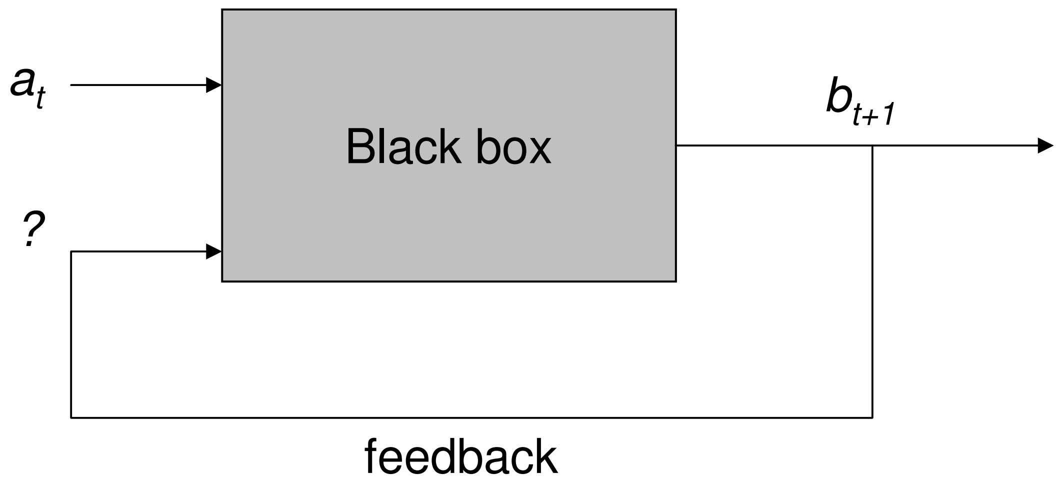


An example circuit of flip-flop

Timing is important when analyzing the operation of a flip-flop!

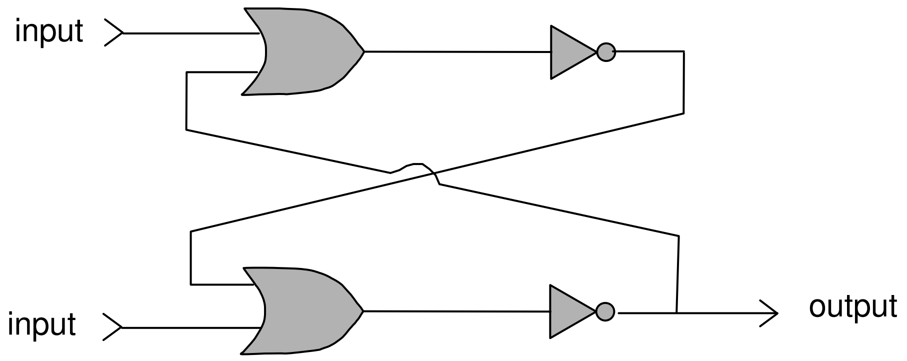
Note: Feedback Systems

- To understand the operation of flip-flops, you must understand the concept of a feedback system:
  - At time t, we have two inputs to the "black box", one is a_t, what is the other one?



Alternative Flip-Flops

- There is more than one way to implement a flip-flop:



- Compare this to previous design, which one is better?

Data Storage Hardware

- In addition to flip-flops, there are other fundamental bit-level data-storage devices based on, for example, magnetic or optical technologies
- A data storage device can be volatile or non-volatile:
  - Volatile memory – holds its value until the power is turned off – Example: flip-flops
  - Non-volatile memory – holds its value after the power is off – Example: magnetic storage

Radix-N Number Coding

- Bit numbers are coded in radix-2 (a.k.a. base-2) while human-friendly numbers are coded in radix-10
- The conversion between radix-2 and radix-10 is not trivial, sometimes, it is easier to use a number coding system that is:
  - Easily readable by human
  - Can be converted to/from radix-2 easily (why?)
- Popular radix-N notation that fulfill these two points are radix-8 (octal notation) and radix-16 (hexadecimal notation)

Different Coding Systems

Binary	Octal	Hexadecimal	Decimal
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	A	10
1011	13	B	11
1100	14	C	12
1101	15	D	13
1110	16	E	14
1111	17	F	15

Main Memory Logical Units

- Data are stored in memory cells in a computer system; each cell is a memory access unit
- The smallest memory access unit of the CPU is called a *byte*, usually equals 8 bits; If you want to emphasize that it is composed of 8 bits, you can call it an "octet"
  - In the good old days, a byte may be composed of 10 bits
- Bits in a byte has orders:

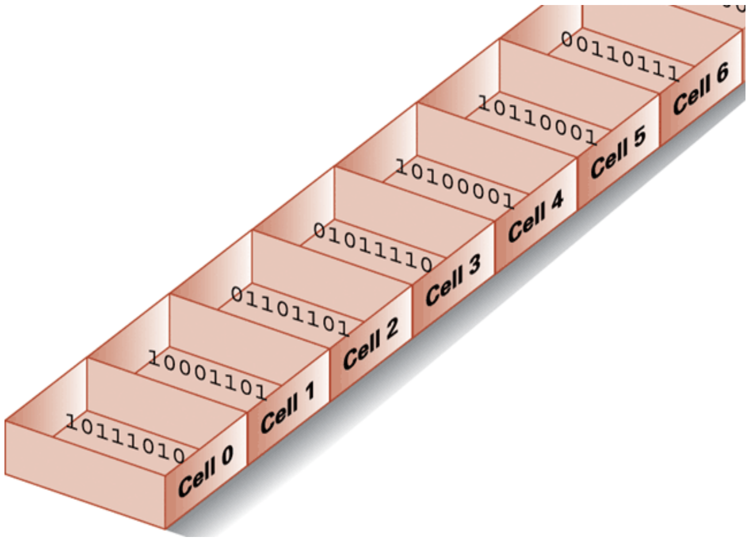
High-order end 0 1 0 1 1 0 1 0 Low-order end

most significant bit

least significant bit

Main Memory Addresses

- An *address* is a “name” to uniquely identify one ***cell*** in the computer’s main memory
  - The addresses for cells in a computer are consecutive numbers, usually starting at zero
- A Random Access Memory (RAM) is a memory device where any cell can be accessed independently

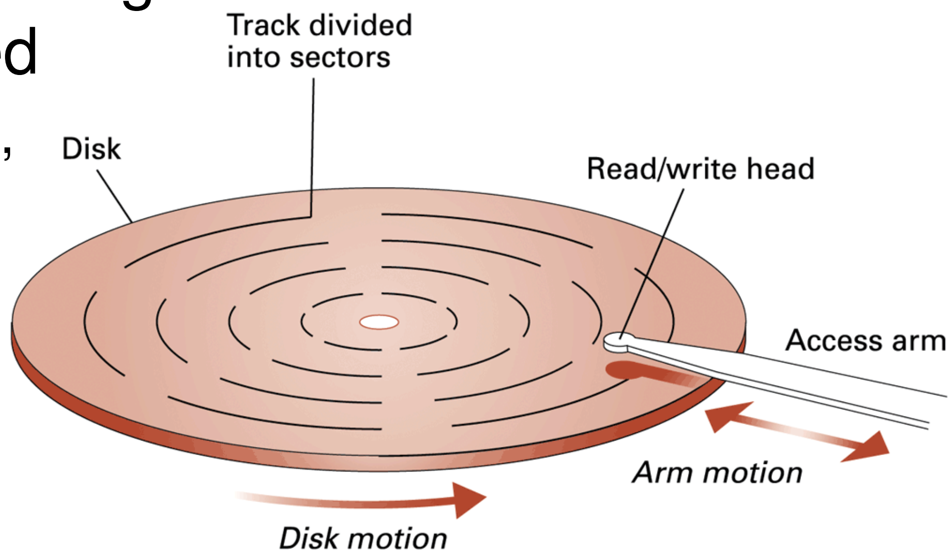


Memory Capacity Measure

- For computer memory/storage, the unit prefixes are slightly different from those of the metric system:
  - "kB" means 1,000 byte; but KB (or more precisely, KiB) means 2^{10} = 1024 bytes
  - "MB" normally means 1,000,000 byte; but MiB means 2^{20} = 1,048,576 bytes
  - Note: the unit KiB (kibi), MiB (mebi), GiB (gibi), etc., were adopted by IEC in 1999.
- For any other computer related units, we stick only to the metric prefix system:
  - The bandwidth unit Mbps always means 10^6 bits-per-second

Mass Storage Systems (1/3)

- For mass storage, rotating disks are usually used
  - Hard disk, floppy disk, CD-ROM



  - Characteristics:
    - Seek time – the time to move the head to the right track
    - Latency time – the time to rotate the disk for half a cycle
    - Access time – seek time + latency time
    - Transfer rate – the speed data transferred to or from the disk

Mass Storage Systems (2/3)

- Back in 1956, disk storage are huge†

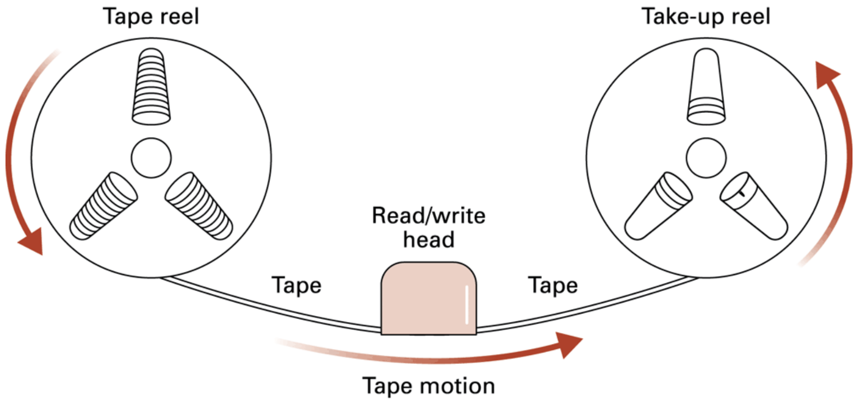
IBM 305 Computer System

IBM 350 Disk Storage Unit
(1200 rpm, 5MB storage)

† http://www.ibm.com/ibm/history

Mass Storage Systems (3/3)

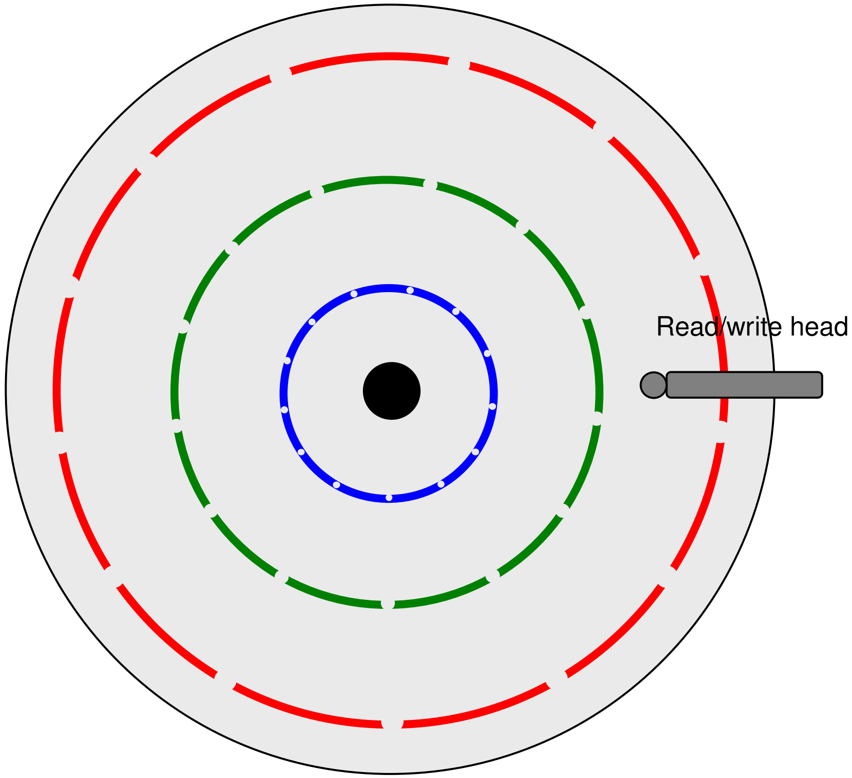
- Some “old” mass storage devices use magnetic tapes:



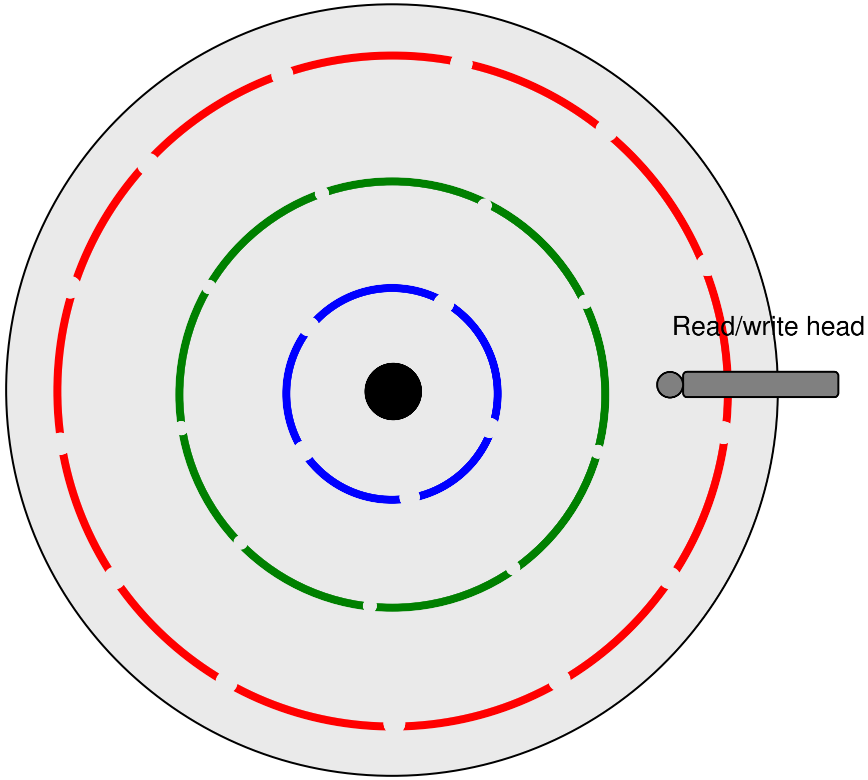
- Some “newer” mass storages use flash memory:


Disk Rotating Speed & Sector Size

- Constant angular speed

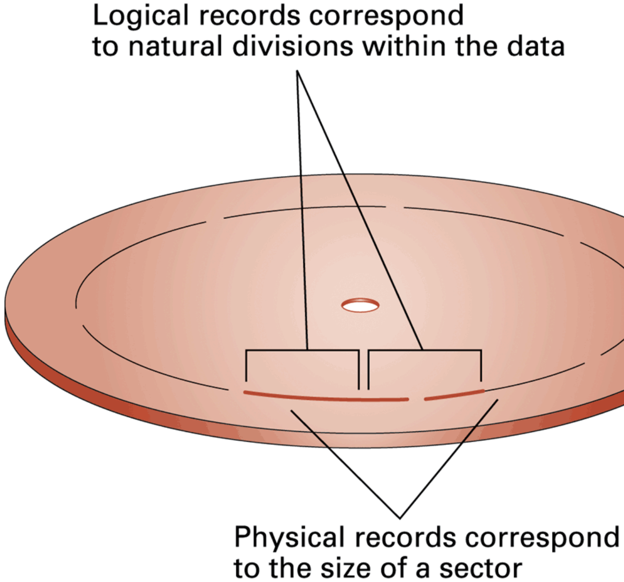


- Variable angular speed



Logical Record vs. Physical Record

- Information stored in a mass storage is organized into files; each file is composed of many smaller units, called logical records
- The mapping between logical record and physical record is not one-to-one:



Representing Text

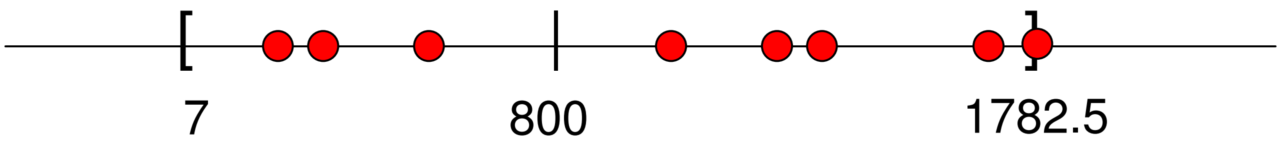
- For text-representation, each character (letter, punctuation, etc.) is assigned a unique bit pattern. There are several coding standards:
  - ASCII: 7-bit coding for most symbols used in written English

01001000	01100101	01101100	01101100	01101111	00101110
H	**e**	**l**	**l**	**o**	**.**

  - Extended ASCII (ISO 8859-1): 8-bit coding, including more western language symbols
  - Unicode: 16-bit coding for most symbols used in most world languages today
  - ISO 10646 Universal Character Set (UCS): a text coding system which uses 32-bit values for each characters

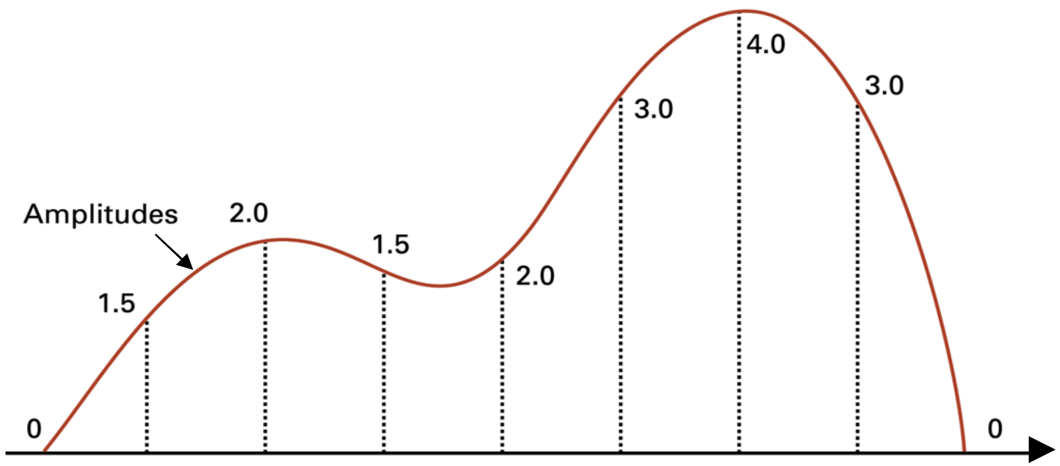
Representing Numeric Values

- There are some limitations of computer representations of numeric values
  - *Overflow* happens if a number is too big to be represented
  - *Underflow* happens when the number is too small
  - *Truncation* happens when a number is fallen between two representable numbers
- For example, 3-bit encoding of numbers can only represent 8 different values; each red dot in the following interval is a numerical values we can represent *accurately*



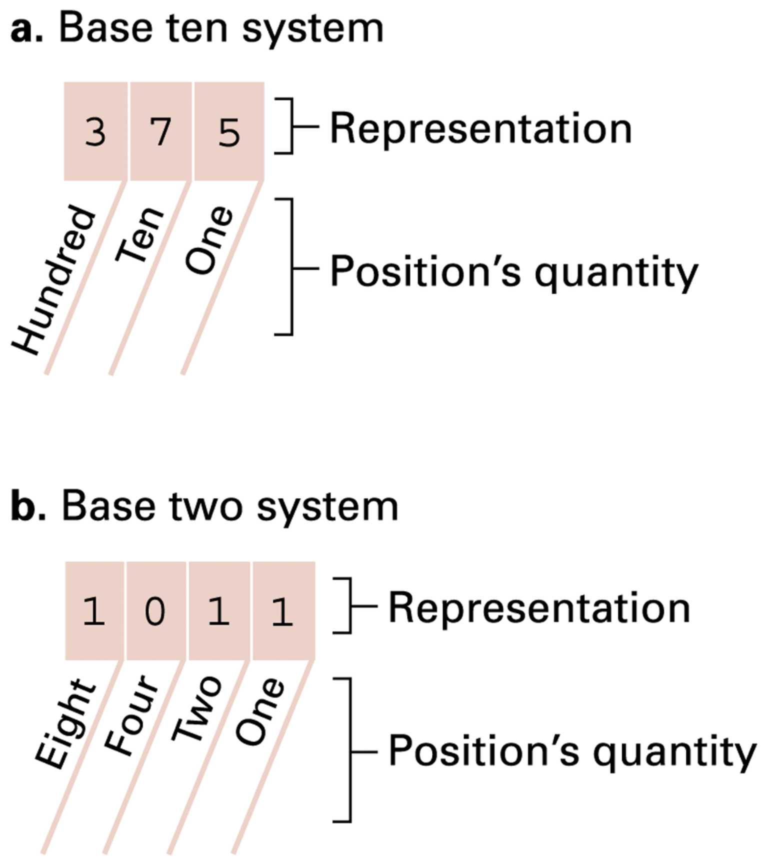
Representing Natural Phenomena

- Natural phenomena is usually detected in *analog* form; must be converted to *digital* form for computer to store and/or process
- Example: a sound wave can be represented by a sequence of numbers: 0, 1.5, 2.0, 1.5, 2.0, 3.0, 4.0, 3.0, 0



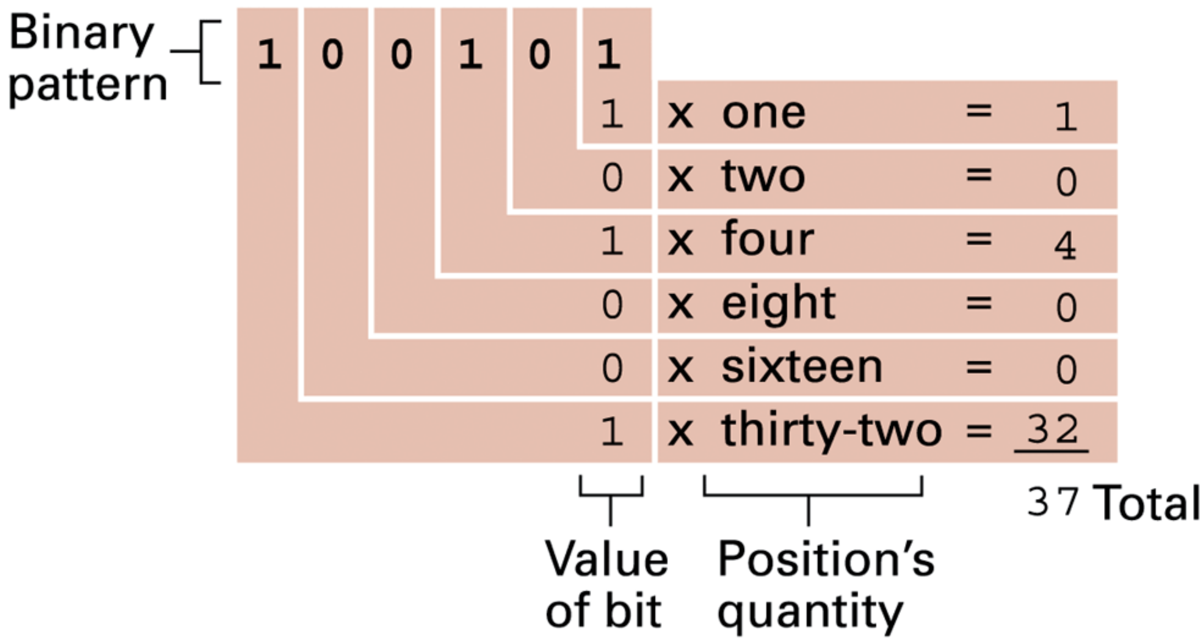
Radix Conversion

- The conversion of number from one radix (base) to another is important
- Each digit in a number has a base quantity called position's quantity
- Position's quantity is the power of the base



Conversion from Radix-2 to Radix-10

- For example, 100101 can be decoded as follows:



Conversion from Radix-10 to Radix-2

- Method to convert from Radix-10 to radix-2:
  - Divide the value by 2 and record the remainder
  - Continue to divide the quotient by two and record the remainder until the quotient is zero
- For example, convert $(13)_{10}$ to radix-2

$$13 = 2 \times 6 + 1$$
$$6 = 2 \times 3 + 0$$
$$3 = 2 \times 1 + 1$$
$$1 = 2 \times 0 + 1$$

$$\longrightarrow (13)_{10} = (1101)_2$$

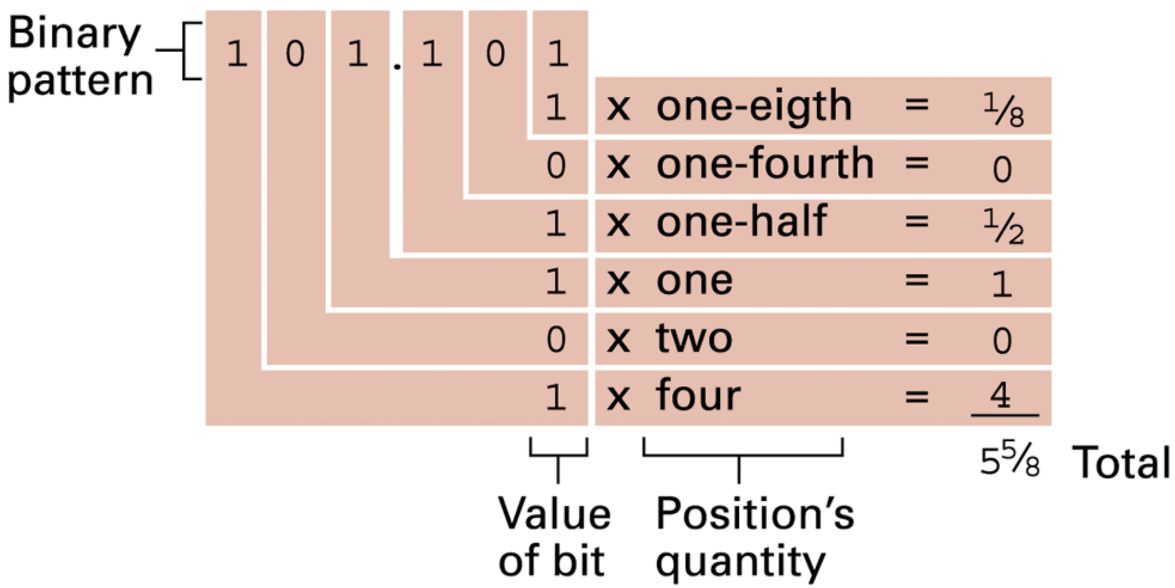
Binary Arithmetic

- Modern computers usually use binary representation of numerical data, therefore, numerical calculations must be done in radix-2 as well
- Binary calculations are simple, but remembering a long sequence of binary numbers is the difficult part (good thing that computers have photographic memory):

$$\begin{array}{r} 0 \\ +\,0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ +\,0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0 \\ +\,1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\,1 \\ \hline 10 \end{array}$$

Binary Representation of Fractions

- A simple way to represent numbers with fractions is to use a fixed point representation:



Converting Fractions to Binary Rep.

- To convert $^5/_8$ to binary digits 0.101, you can inverse the above-mentioned process:

$$\frac{5}{8}\times 2=\frac{5}{4}=1+\frac{1}{4} \quad\rightarrow\quad \frac{5}{8}=1\times 2^{-1}+2^{-1}\times\frac{1}{4}$$

$$\frac{1}{4}\times 2=\frac{1}{2}=0+\frac{1}{2} \quad\rightarrow\quad \frac{1}{4}=0\times 2^{-1}+2^{-1}\times\frac{1}{2}$$

$$\rightarrow\quad \frac{5}{8}=1\times 2^{-1}+2^{-1}\times(0\times 2^{-1}+2^{-1}\times\frac{1}{2})$$

$$\rightarrow\quad \frac{5}{8}=1\times 2^{-1}+0\times 2^{-2}+1\times 2^{-3}$$

Representing Integers

- Unsigned integers can be represented in base two
- Signed integers are numbers that can be positive or negative. For negative numbers, there are several ways to represent it:
  - One's complement –
    bitwise "NOT" of a positive number is the negative representation of the number
  - Two's complement notation –
    the most popular representation of negative numbers
  - Excess notation –
    often used in representing floating point exponents

Two's Compliment Representation

a. Using patterns of length three

Bit pattern	Value represented
011	3
010	2
001	1
000	0
111	−1
110	−2
101	−3
100	−4

b. Using patterns of length four

Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

Coding Rules of 2's Complement

- Algorithm:
  - First, complement all the digits of the number
  - The, add one to the complemented number
- For example, –6 can be represented as follows:
  - $(6)_{10} = (0110)_2$
  - $0110 \rightarrow 1001$
  - $1001 + 1 = 1010 = (-6)_{10}$
- The nice part about 2's Complement representation of negative numbers is that addition matches natural representation

2's Complement Arithmetic

Problem in base ten	Problem in two's complement	Answer in base ten
3 + 2	0011 + 0010 = 0101	5
−3 + −2	1101 + 1110 = 1011	−5
7 + −5	0111 + 1011 = 0010	2

Excess Notation Systems

- Excess notation preserves the natural ordering of (negative) numbers
- Arithmetic operations become less trivial
- Examples of negative numbers represented using excess system:

Bit pattern	Value represented
111	3
110	2
101	1
100	0
011	-1
010	-2
001	-3
000	-4

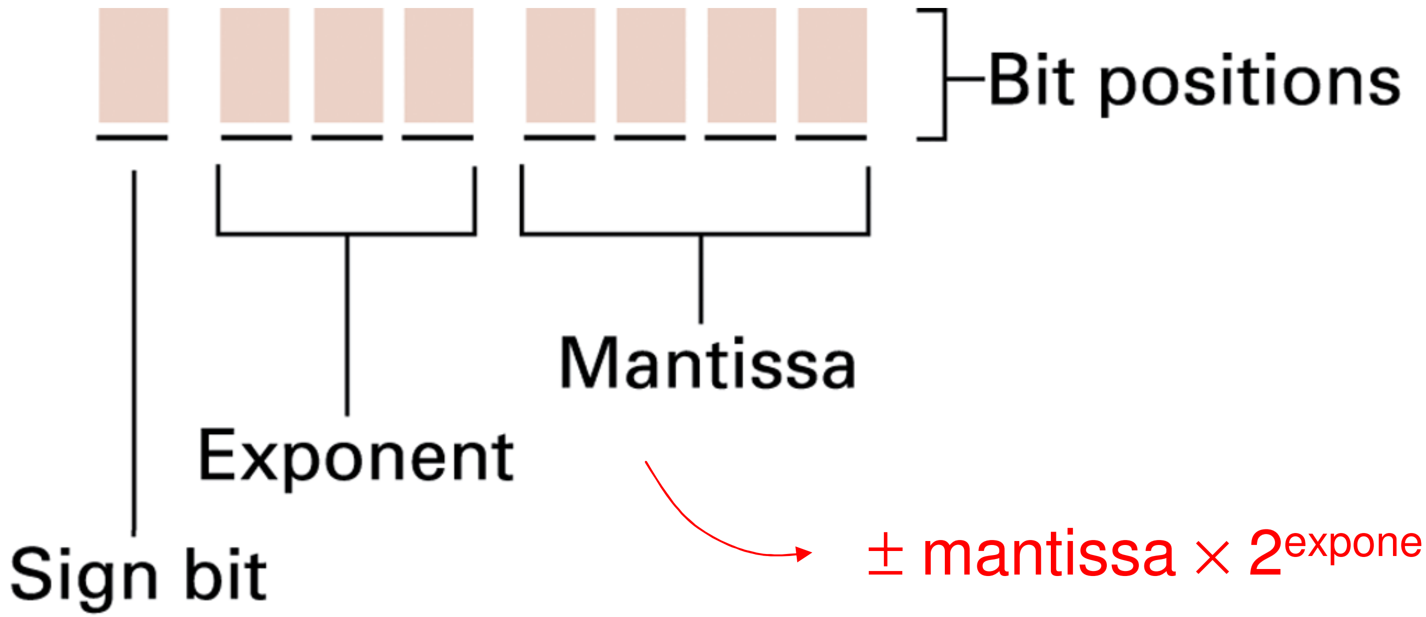
excess 3

Bit pattern	Value represented
1111	7
1110	6
1101	5
1100	4
1011	3
1010	2
1001	1
1000	0
0111	-1
0110	-2
0101	-3
0100	-4
0011	-5
0010	-6
0001	-7
0000	-8

excess 8

Floating Point Representation (1/2)

- A number with fractions can be represented in fixed point, as well as in floating point
- A floating point representation is composed of four parts:

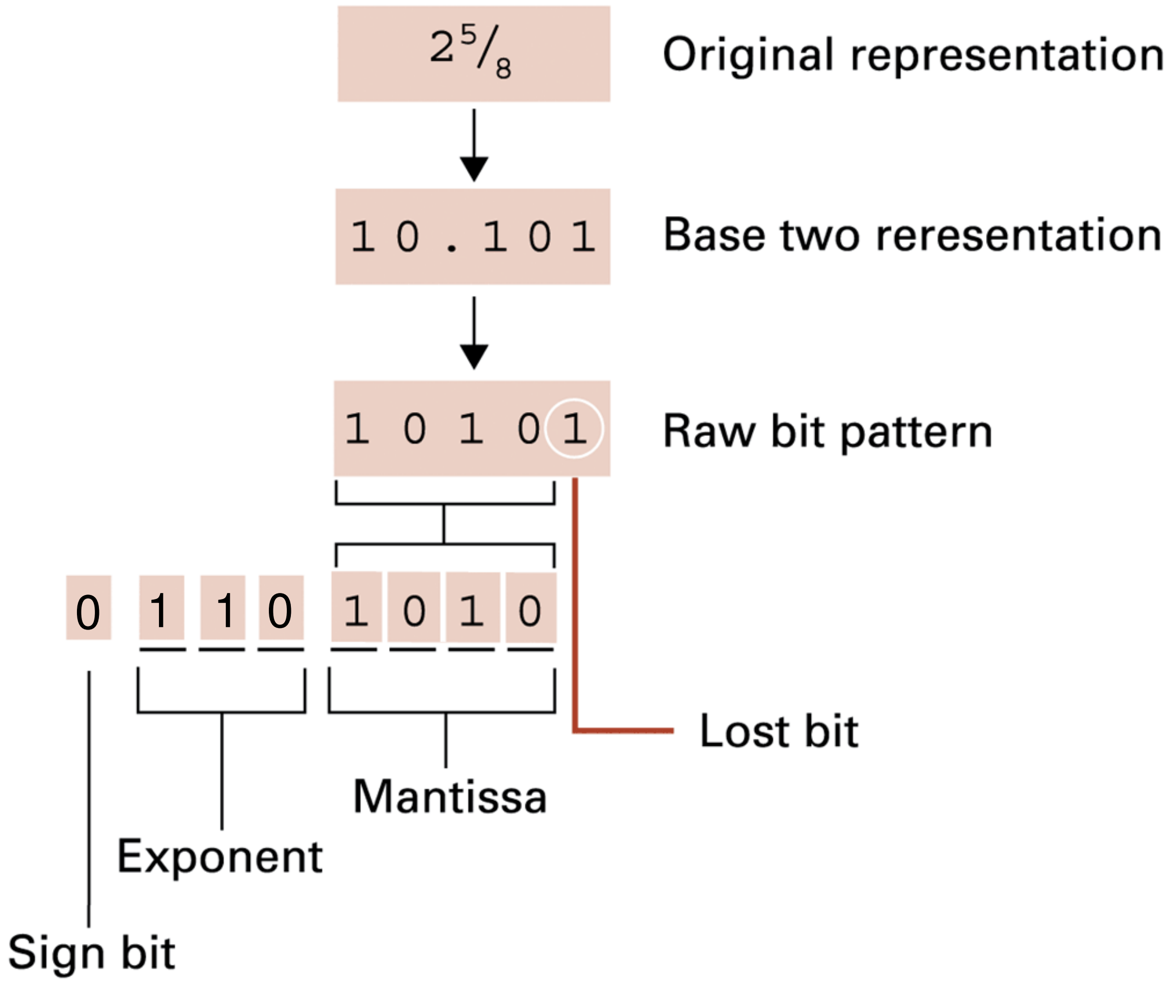


$\pm$ mantissa $\times$ $2^{exponent}$

Floating Point Representation (2/2)

- Coding of each field of a floating point number:
  - For mantissa, fixed point representation is obviously the logical choice (remember scientific notations?)
  - For exponent, we need to represent positive and negative numbers $\rightarrow$ excess notation is often used (why?)
  - For sign bit, a single bit can be used to record the sign
- There is an international standard, IEEE-754, that defines the representation as well as the arithmetic of floating point computations
  - It is different (more complicated) from the one described in our textbook, but the concept is the same

Example: Coding of $2\frac{5}{8}$

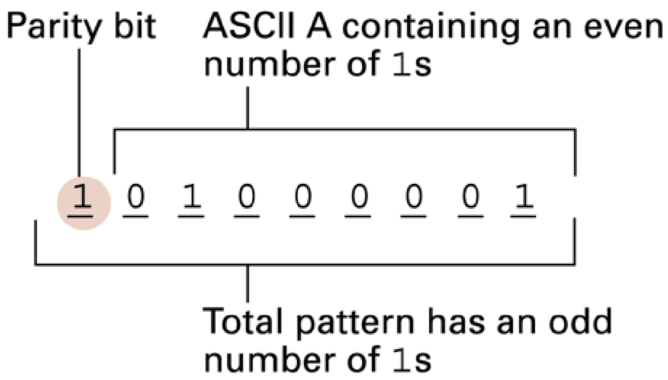


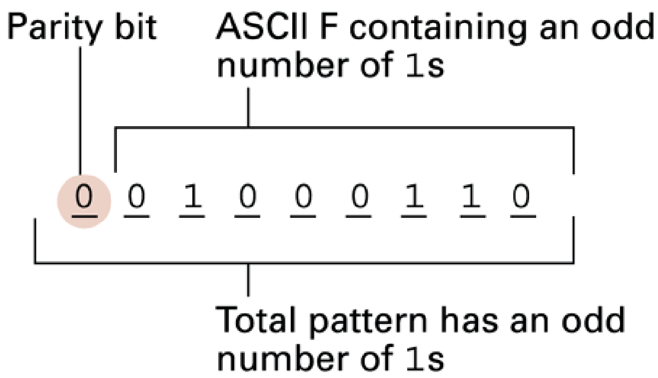
Coding for Error Protection

- Sometimes, computing systems are not reliable. Numbers stored in a computing system can have random bit errors over time.
- To detect and/or correct errors, the number representation has to be robust to random bit errors
- Two types of error-robust coding are possible:
  - Error detection coding (e.g. parity bits)
  - Error correcting coding (add redundant bits to increase robustness)

Parity Bit Coding

- Each number can be appended with one extra bit (called parity bit) for error detection
  - The extra bit force the total bit pattern has an odd or even number of 1's.
  - Odd parity – the number of 1's is odd
  - Even parity – the number of 1's is even





Error Correction Coding (ECC)

- For ECC, we don't use all possible representations, instead, we increase the "distance" between meaningful representations:

Symbol	Code
A	000000
B	001111
C	010011
D	011100
E	100110
F	101001
G	110101
H	111010

distance = 4

distance = 4

distance = 3

Example: Decoding of 010100

Character	Distance between the received pattern and the character being considered
A	2
B	4
C	3
D	1 *Smallest distance*
E	3
F	5
G	2
H	4