Session 2: Numeral Systems and Data Storage

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Content

- Bits and their Storage
  - Bits, Gates, Flip-Flop
- Main Memory
- Representing Information as Bit Patterns
  - Text, number, images, sound
- Binary System
- Storing Integers
- Storing Fractions
- Mass Storage

Question

- How computers store data?
  - Number, text, image, sound, video
- How computers can mapping data to the real world?

Bits and their storage

- Bits
- Gates
- Flip-flop

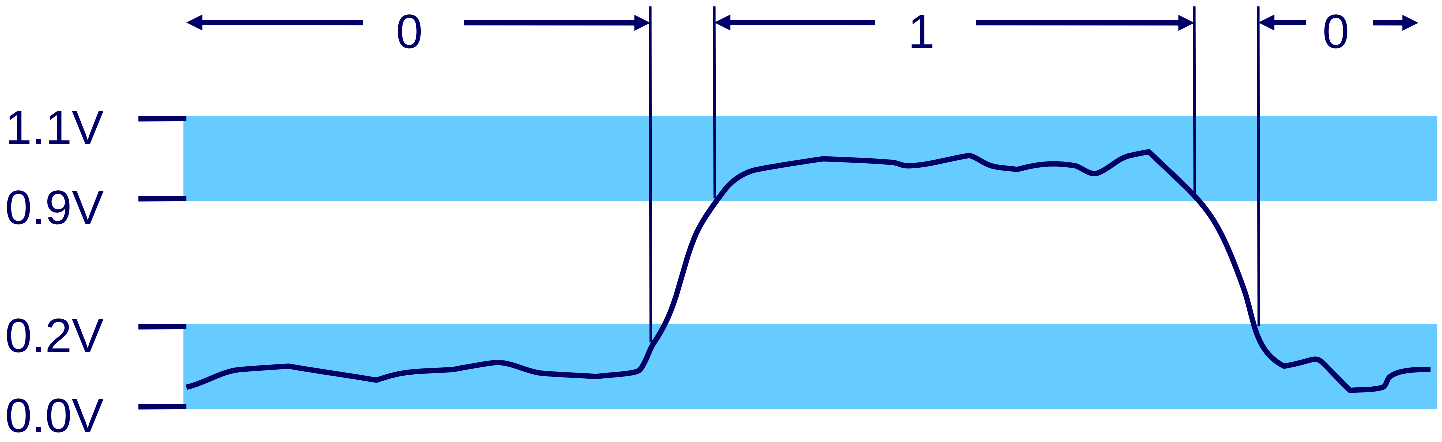
Bits

- Binary uses two digits: **0** and **1**.
- bit (Binary Digit): smallest unit storing information.
- Can be stored in memory (cell) or register.
- Register 1 byte (8 bit) or 1 word (16 bit), etc.

Bits

- Why using 2 digits 0 and 1 to encode data?

- Electronic implementation



Nguồn: Computer System – A Programmer's Perspective, 3e

Bits

- Easily to encode:
  - Numeric value : 1 & 0
  - Boolean value : true & false
  - Voltage : high & low
  - Punched card : punched & not punched

- Data → encode using binary system to store in computers

Bits – Boolean Operations

- An operation that manipulates one or more *true/false* values
  - Bit 0 ~ False
  - Bit 1 ~ True
- Specific operations : AND, OR, XOR, NOT
- Why Boolean operations?
  - Computers are built by small components
  - These components can process Boolean operations quite fast

Bits – Boolean Operations

The AND operation

0	0	1	1
AND 0	AND 1	AND 0	AND 1
0	0	0	1

The OR operation

0	0	1	1
OR 0	OR 1	OR 0	OR 1
0	1	1	1

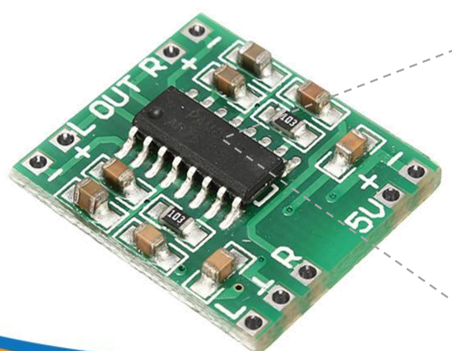
The XOR operation

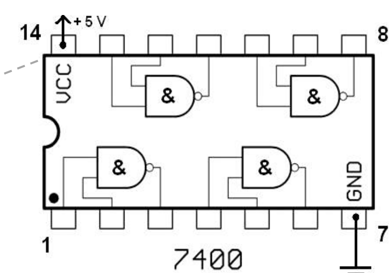
0	0	1	1
XOR 0	XOR 1	XOR 0	XOR 1
0	1	1	0

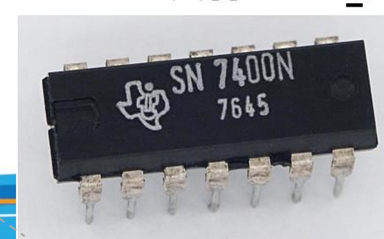
Source: Computer Science - An Overview, 12e

Gates

- A device that computes a Boolean operation
- Often implemented as (small) electronic circuits:
  - Including: resistor (điện trở), transistor (bòng bán dẫn), Capacitor (tụ điện), diot (điốt), …
  - 0 & 1 ~ voltage



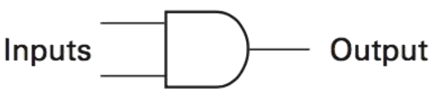




Nguồn: Wikipedia

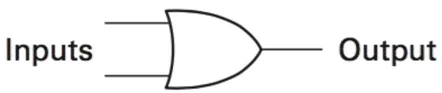
A pictorial representation of gates

AND



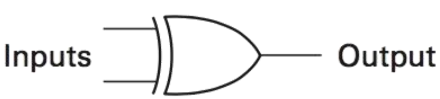
Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

OR



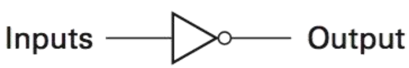
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1

XOR



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

NOT

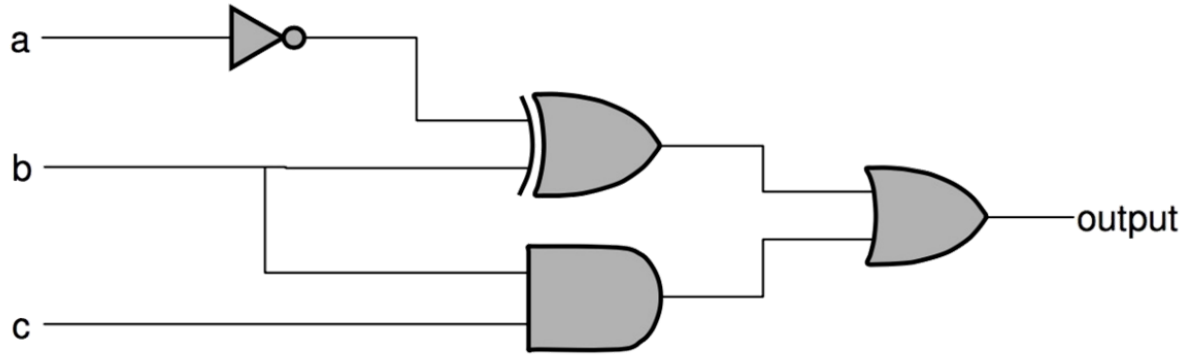


Inputs	Output
0	1
1	0

Nguồn: Computer Science - An Overview, 12e

Example – Simple circuit

Circuit



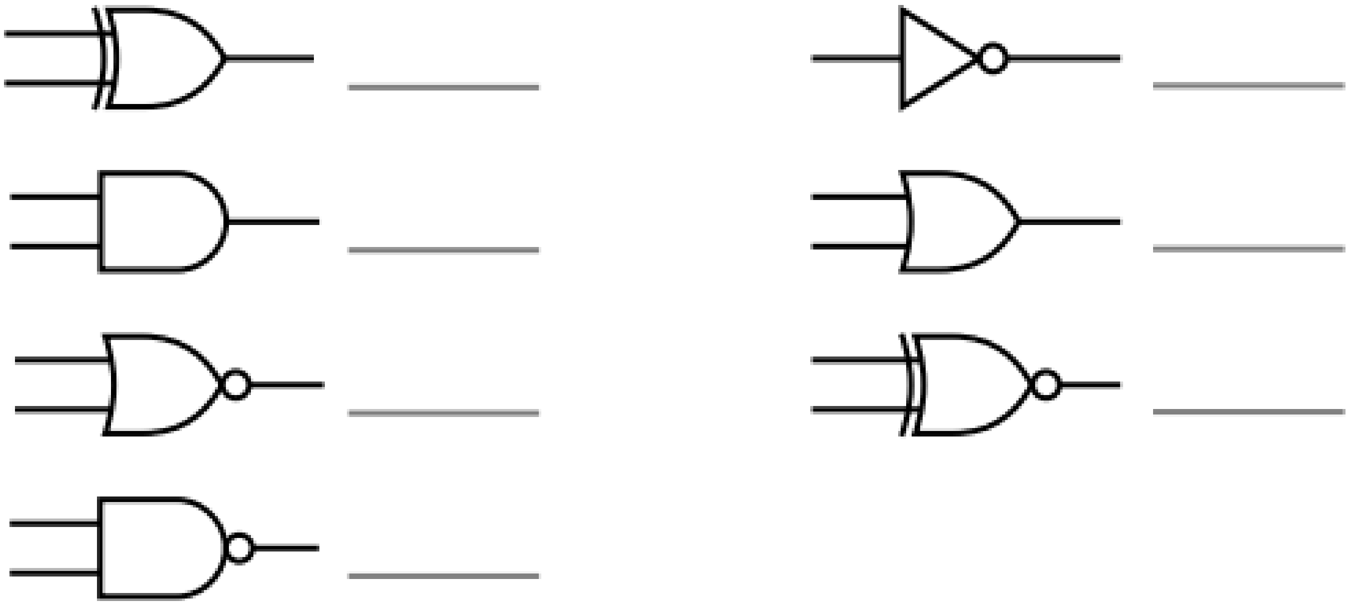
Truth Table

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

Source Chun-Jen Tsai, ics12, National Chiao Tung University

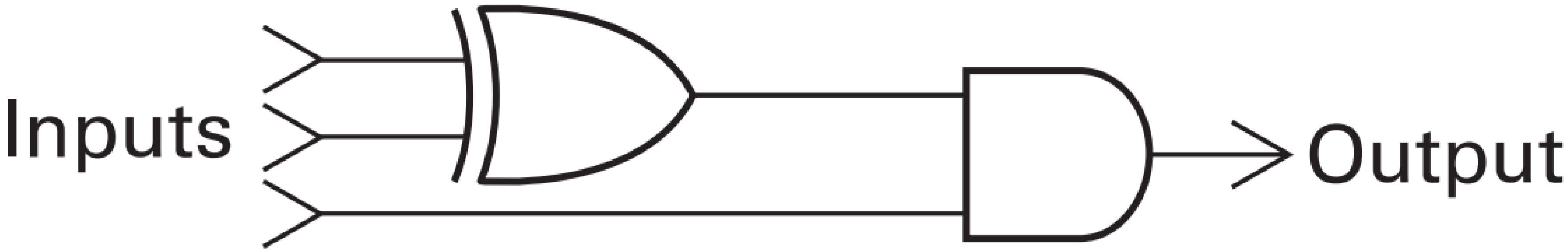
Quiz

- What are the names of these gates?



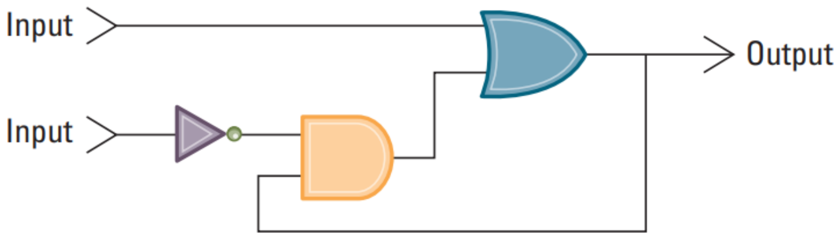
Quiz

- What input bit patterns will cause the following circuit to produce output of 1?



Flip-Flop

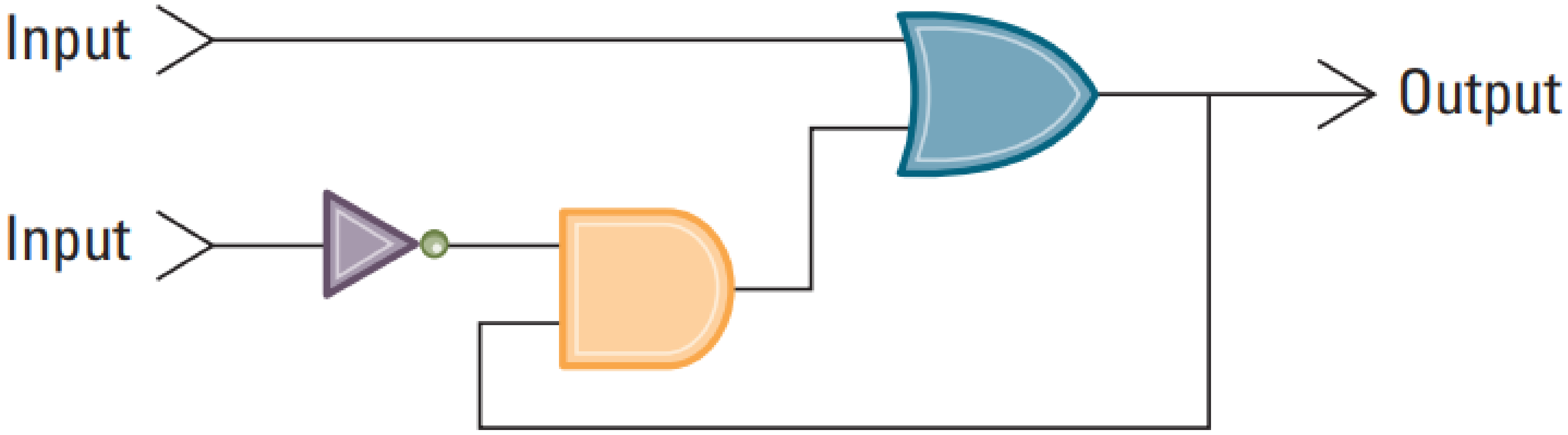
- A circuit built from gates that can store one bit
- A circuit producing output (or "preserved") 0 or 1, but remain constant until the pulse from another circuit makes it change to another value



  - One input line is used to set its stored value to 1 (output is 1)
  - One input line is used to set its stored value to 0 (output is 0)
  - While both input lines are 0, the most recently stored value is preserved

source: Computer Science - An Overview, 12e

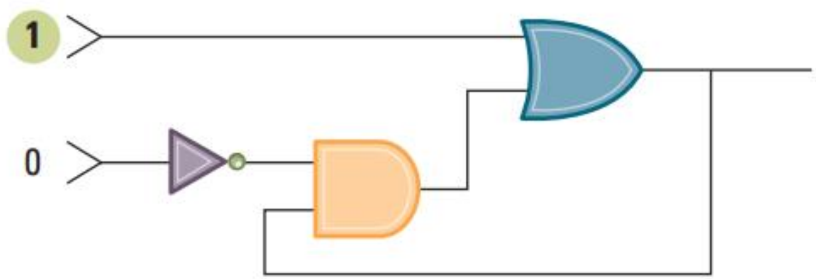
A simple flip-flop circuit



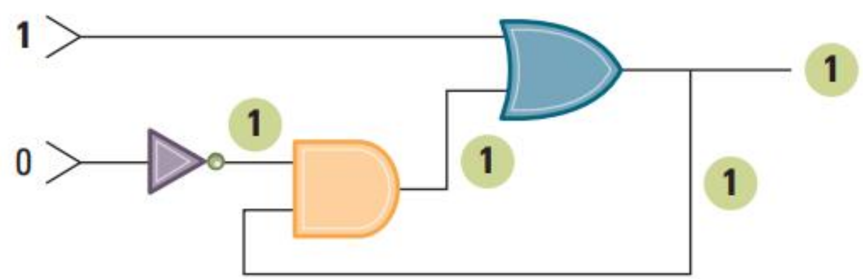
Flip-Flop

- Setting the output of a flip-flop to 1

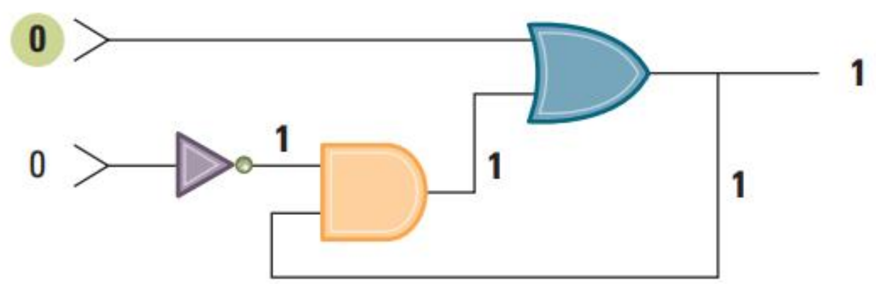
a. First, a 1 is placed on the upper input.



b. This causes the output of the OR gate to be 1 and, in turn, the output of the AND gate to be 1.



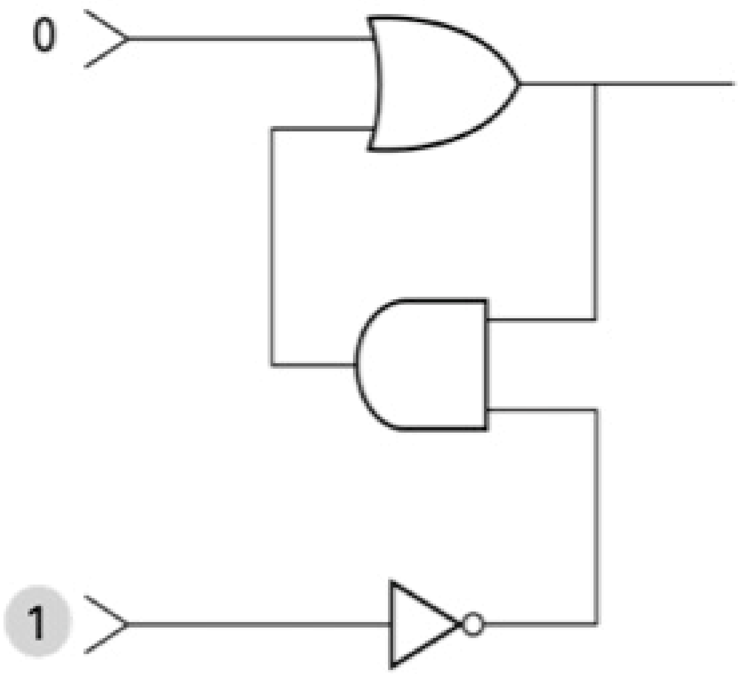
c. Finally, the 1 from the AND gate keeps the OR gate from changing after the upper input returns to 0.



Source: Computer Science - An Overview, 12e

Flip-Flop

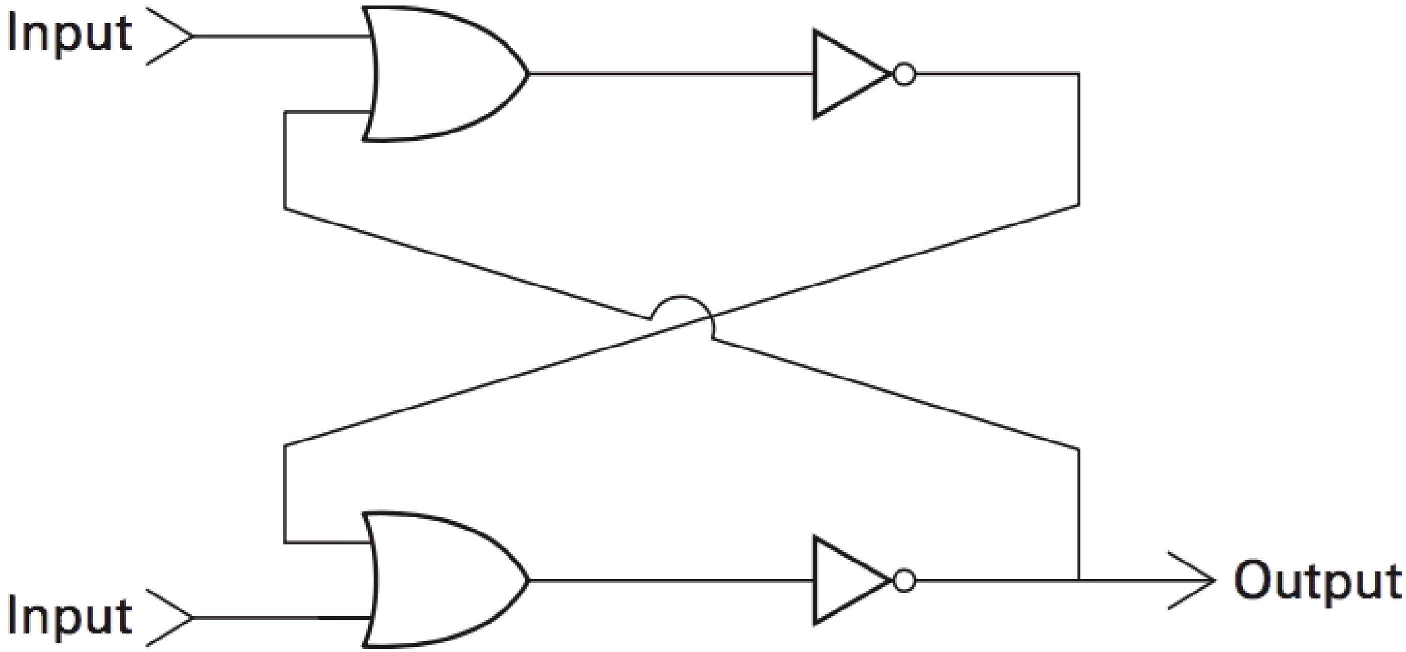
- Setting the output of a flip-flop to 0



Nguồn: *Computer Science - An Overview, 12e*

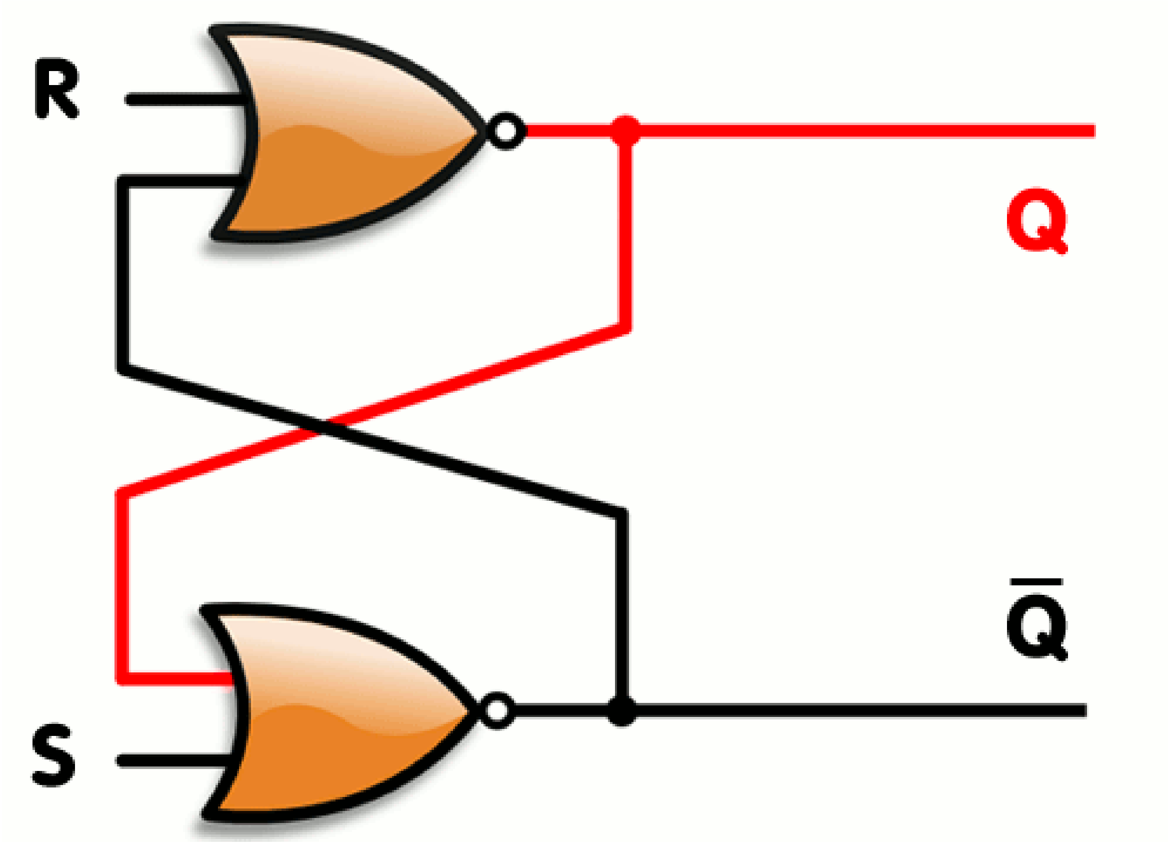
Quiz - Flip-Flop?

- If upper input is 1 and lower input is 0, what is the output?



Source: Computer Science - An Overview, 12e

Quiz- Flip-Flop?



Source :wikipedia

Flip-flop - Activity

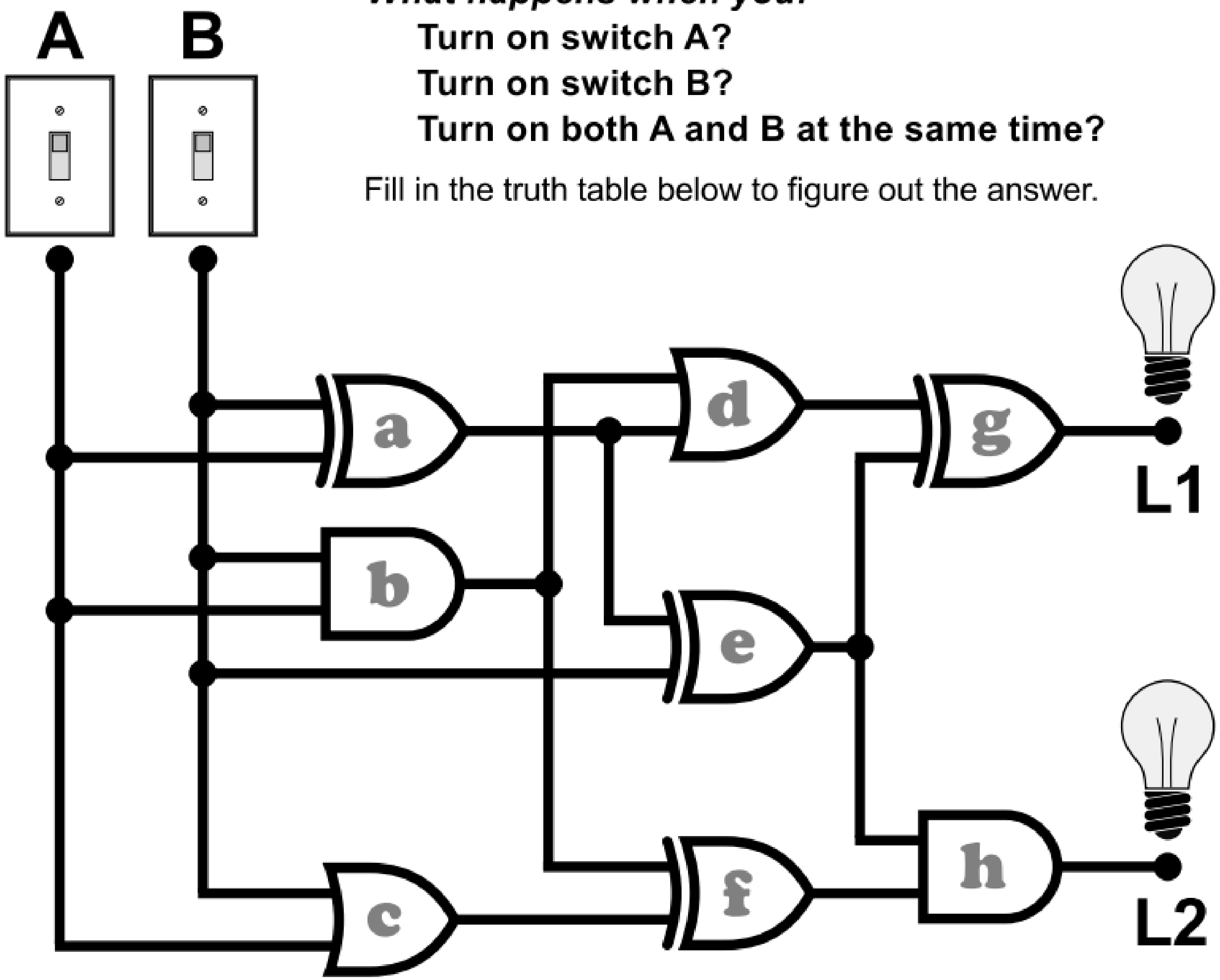
What happens when you:

Turn on switch A?

Turn on switch B?

Turn on both A and B at the same time?

Fill in the truth table below to figure out the answer.



Hexadecimal Notation

- Hexadecimal notation: A shorthand notation for long bit patterns
  - Divides a pattern into groups of four bits each
  - Represents each group by a single symbol

- Example: 10100011 becomes A3

The hexadecimal coding system

Bit pattern	Hexadecimal representation
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

Quiz

- What bit patterns are represented by the following hexadecimal patterns?
  - 5FD97
  - 610A
  - ABCD
  - 0100

MAIN MEMORY

Introduction

- We know
  - How machines encode information into chain of bits
  - Basic storage devices

- So
  - To store data, machines need to have a million of circuits (a circuit stores 1 bit)

→ Place containing these bits is called ***Main Memory***

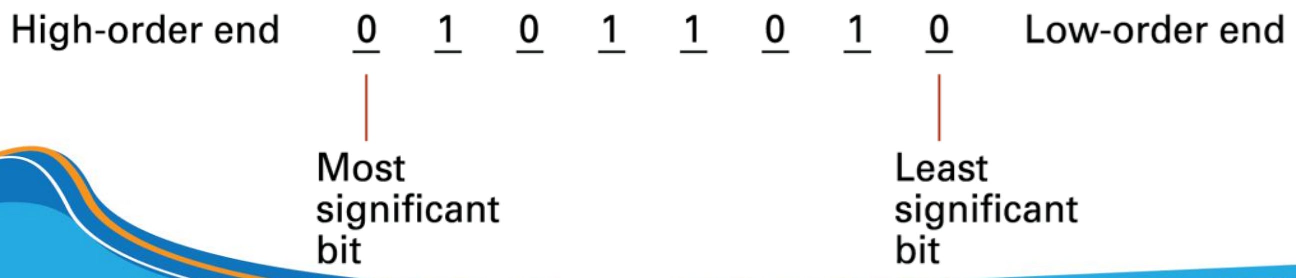
Introduction

- In additional to Flip-flops, machines have other storage devices (called external memory)
  - Magnetic, optical, flash deices

- Storage devices
  - Volatile memory (bộ nhớ khả biến)
    - Requires power to maintain the stored information
  - Non-volatile memory (bộ nhớ bất khả biến)
    - Can retrieve stored information even after having been power cycled

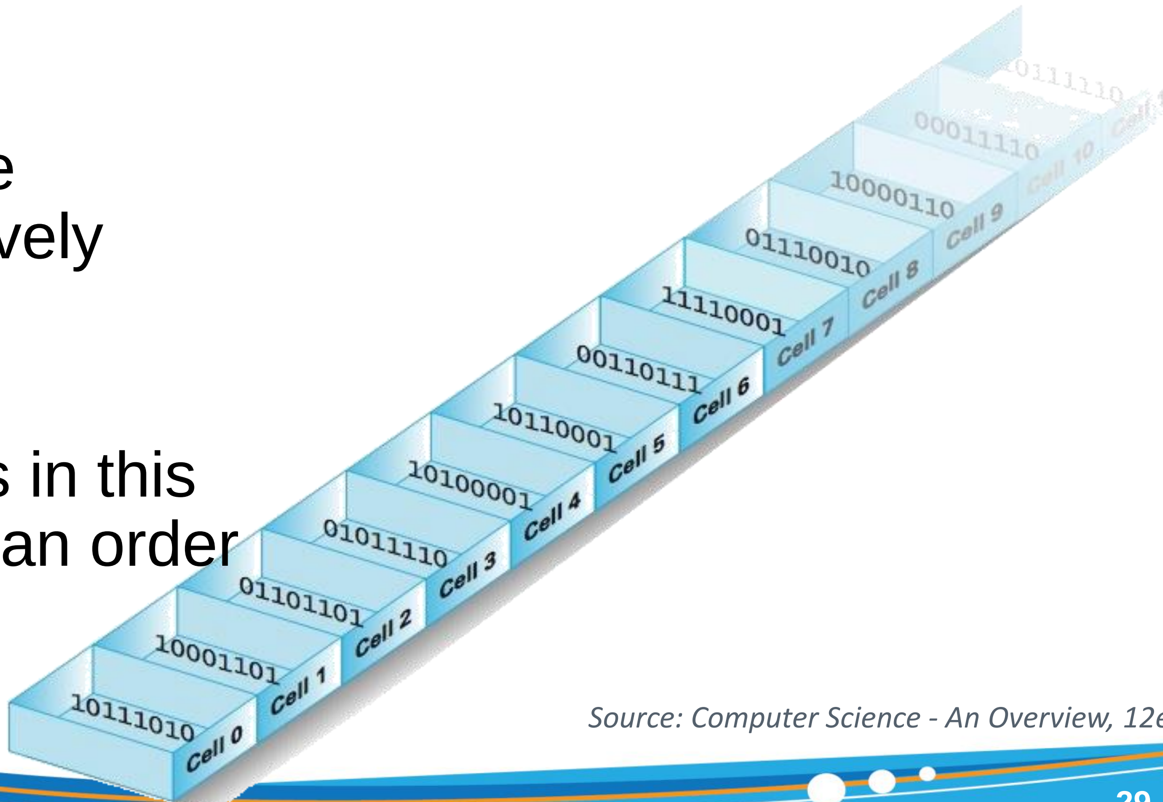
Main memory cells

- Cell: A unit of main memory (typically 8 bits which is one **byte**)
  - Most significant bit: the bit at the left (high-order) end of the conceptual row of bits in a memory cell
  - Least significant bit: the bit at the right (low-order) end of the conceptual row of bits in a memory cell
- Organization of a byte-size memory cell
  - Size ***8 bits*** (***1 byte***)
  - Sequence of bits



Main memory address

- ***Address:*** A “name” that uniquely identifies one cell in the computer’s main memory
  - “Names” are actually numbers
  - These numbers are assigned consecutively starting at zero
  - Numbering the cells in this manner associates an order to the memory cells



Source: Computer Science - An Overview, 12e

Terminology

- Random Access Memory (RAM)
  - Memory in which individual cells can be easily accessed in any order

- Dynamic Memory (DRAM)
  - RAM composed of volatile memory

Measuring memory capacity

Bits	Bit positions	Combinations
1 bit	0	2
2 bit	1 0	2^2
3 bit	2 1 0	2^3
n bit	n-1 … 5 4 3 2 1 0	2^n

$0...000 \rightarrow 1...111 = 2^n - 1$

Measuring memory capacity

Capacity		Value
Byte	B	8 bit
KiloByte	KB	2^{10} B = 1024 Byte
MegaByte	MB	2^{10} KB = 2^{20} Byte
GigaByte	GB	2^{10} MB = 2^{30} Byte
TeraByte	TB	2^{10} GB = 2^{40} Byte
Peta	PB	2^{10} TB = 2^{50} Byte
Exabyte	EB	2^{10} PB= 2^{60} Byte

REPRESENTING INFORMATION AS BIT PATTERNS

Representing Text

- Each character (letter, punctuation, etc.) is assigned a unique bit pattern
  - ASCII: Uses patterns of 7-bits to represent most symbols used in written English text
  - ISO developed a number of 8 bit extensions to ASCII, each designed to accommodate a major language group
  - Unicode: Uses patterns of 16-bits to represent the major symbols used in languages world wide (UTF-8, UTF-16,…)

Representing Text

dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0	0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
10	a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
11	b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
12	c	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	l
13	d	015	CR	45	2d	055	-	77	4d	115	M	109	6d	155	m
14	e	016	SO	46	2e	056	.	78	4e	116	N	110	6e	156	n
15	f	017	SI	47	2f	057	/	79	4f	117	O	111	6f	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
27	1b	033	ESC	59	3b	073	;	91	5b	133	[	123	7b	173	{
28	1c	034	FS	60	3c	074	<	92	5c	134	\	124	7c	174	|
29	1d	035	GS	61	3d	075	=	93	5d	135	]	125	7d	175	}
30	1e	036	RS	62	3e	076	>	94	5e	136	^	126	7e	176	~
31	1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL

The message “Hello.” in ASCII

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

Storage - Represent

- Storage and processing: bit

- Display/represent: character

  $\Rightarrow$ Need to have a map table between the two, do the mapping between numerical values and character values.

- ASCII and Unicode.

ASCII

- **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.

- First edition was published in 1963.

- Based on the English alphabet (‘a’– ‘z’, ‘A’ – ‘Z’).

- ASCII encodes 128 specified characters into seven-bit integers (digits 0 to 9, lowercase letters a to z, uppercase letters A to Z, and punctuation symbols).

ASCII – Characters

- Printing characters
  - ‘ ’ (blank): 32 (0x20)
  - ‘0’ -> ‘9’: 48 (0x30) -> 57 (0x39)
  - ‘A’ -> ‘Z’: 65 (0x41) -> 90 (0x5A)
  - ‘a’ -> ‘z’: 97 (0x61) -> 122 (0x7A)

- Non-printing/control characters
  - null: 0
  - ‘	’ (tab): 9
  - enter/ line feed: 10
  - carriage return: 13

ASCII

- ASCII extent: 256 characters.
  - 128 as the first edition.
  - 128 extent to characters including: Greece ('α', 'β', 'π', …), currency ('£', '¥', …), …

- ASCII cannot represent characters in other languages such as Vietnamese, Russian, Japanese, Arabic, etc.

Unicode

- **Unicode** is a computing industry standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems.
- Containing 1.114.112 code points, divived into 17 regions, each has 65535 (2^{16}) code points.

Unicode

- There are different in using the Unicode, depending on the storage size of code point

  - UTF – 8: storage size from 1 -> 4 Bytes.
  - UTF – 16: storage size 2 Bytes.
  - UTF – 32: storage size 4 Bytes.

Unicode and Vietnamese

- Unicode contains Vietnamese code points is at:

- http://vietunicode.sourceforge.net/charset/v3.htm

Unicode - Font

- Each Unicode set has different ways to be represented.

- Unicode is implemented as a digital data file containing a set of graphically related glyphs, characters, or symbols.

- Fonts support Unicode (with Vietnamese) :
  - Times New Roman,
  - Arial,
  - Tahoma,
  - …

Representing Numeric Values

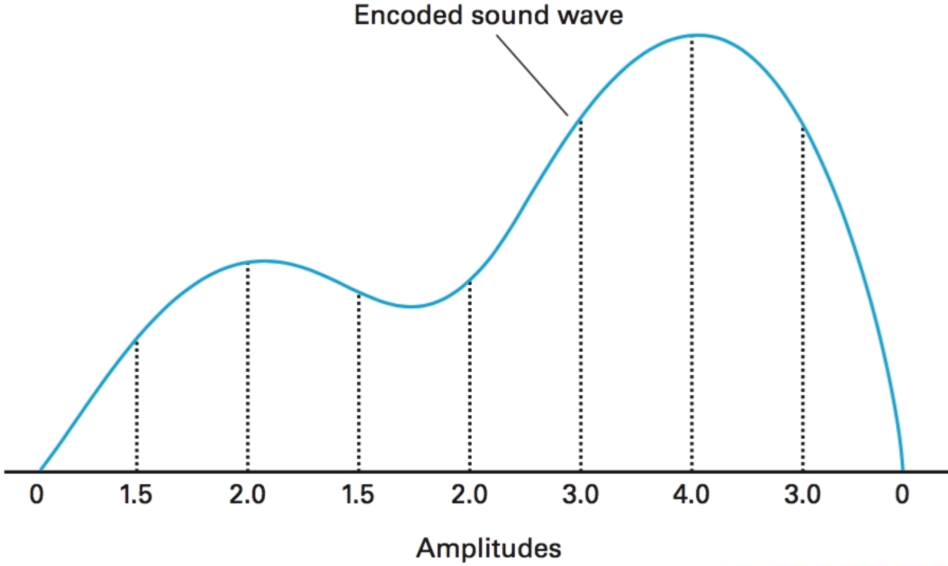
- Binary notation: uses bits to represent a number in base two
- Limitations of computer representations of numeric values
  - Overflow: occurs when a value is too big to be represented
  - Underflow: occurs when a value is too small to be represented
  - Truncation: occurs when a value cannot be represented accurately

Representing Images

- Bit map techniques
  - Pixel: short for “picture element”
  - Encoding
    - RGB
    - Luminance (cường độ sáng) and chrominance (độ đậm/nhạt của màu)
- Vector: can zoom in/out → not affected to the quality
  - Scalable
  - TrueType and PostScript

Representing Sound

- Sampling techniques: get the amplitude of the sound wave at regular intervals and store the series of values
  - Used for high quality recordings
  - Records actual audio
- Sampling rate (Hertz – Hz)
  - Telephone: 8,000 samples/second – 8 KHz
  - Music: 44,100 samples/second – 44,1 KHz
    - High definition: 16 bits/sample
    - Stereo: 32 bits/sample



Quiz

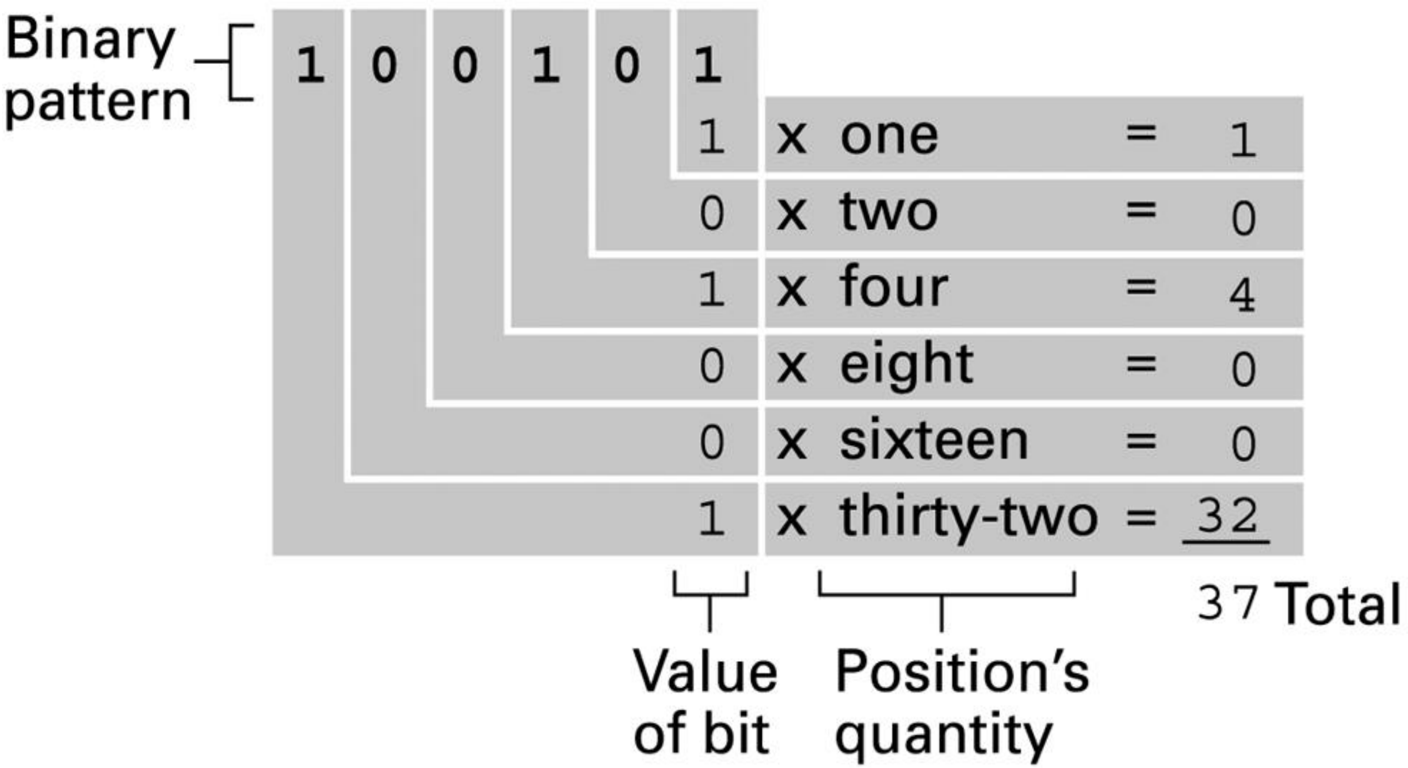
- Suppose a stereo recording of one hour of music is encoded using a sample rate of 44,100 samples per second as discussed.
- What is the size of the encoded version?

BINARY SYSTEM

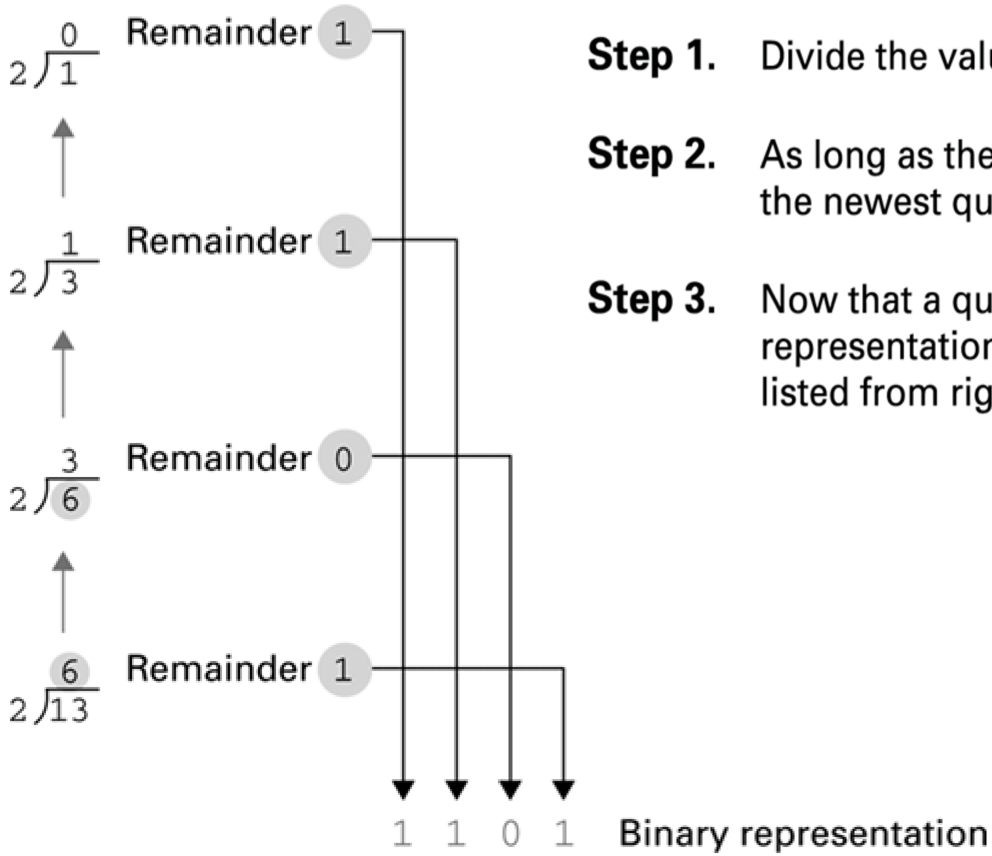
Binary system

- The traditional decimal system is based on powers of ten.
- The Binary system is based on powers of two
- The base 10 and the binary system

Decoding the binary representing 100101



Base 10 to binary



Step 1. Divide the value by two and record the remainder.

Step 2. As long as the quotient obtained is not zero, continue to divide the newest quotient by two and record the remainder.

Step 3. Now that a quotient of zero has been obtained, the binary representation of the original value consists of the remainders listed from right to left in the order they were recorded.

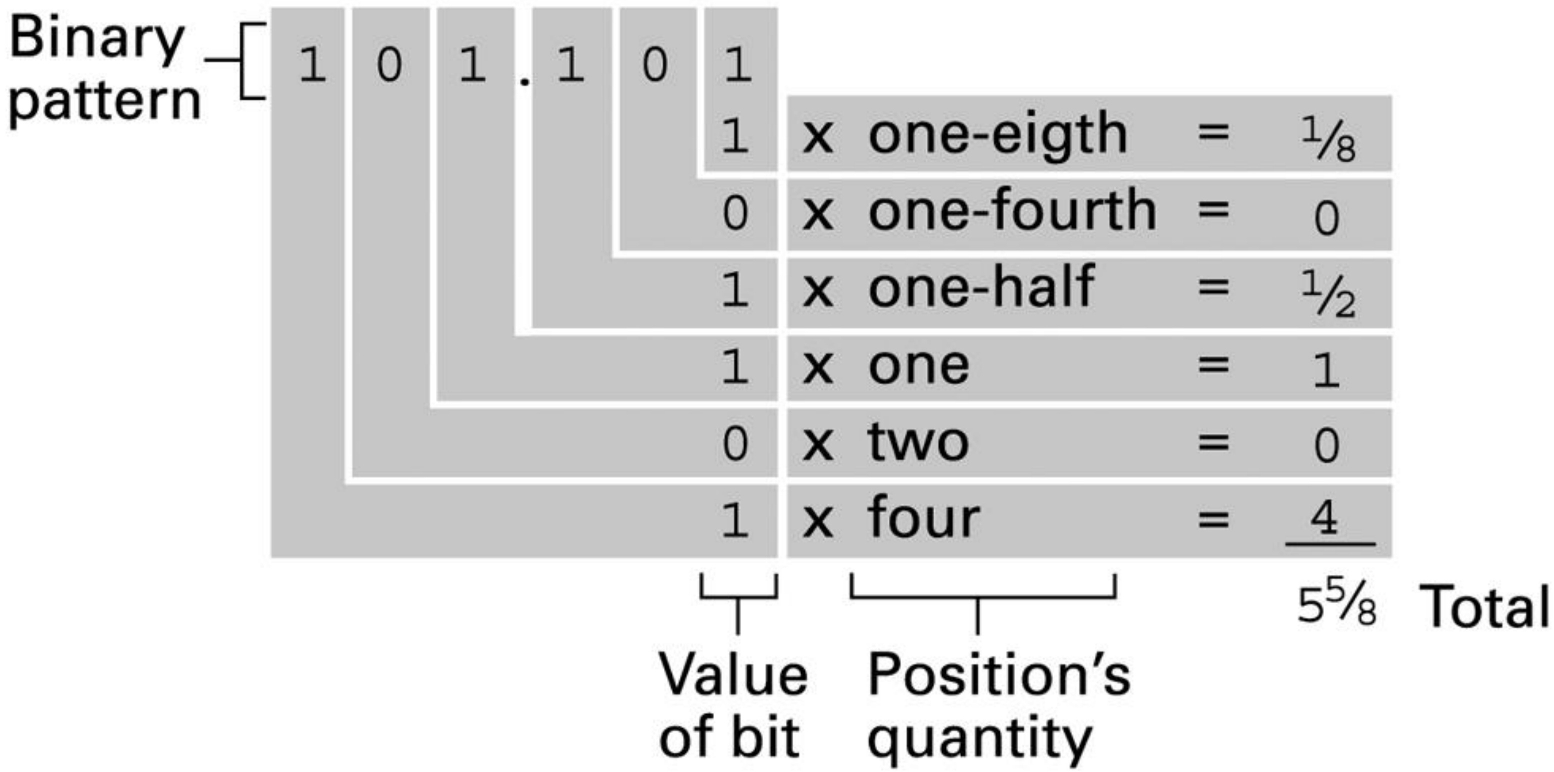
Binary additional facts

- Additional table

+	0	1
0	0	1
1	1	10

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

Decoding the binary representation (Binary to base 10)



$1010.11_2 = 1*2^3+0*2^2+1*2^1+0*2^0+1*2^{-1}+1*2^{-2}$

$1010.11_2 = 8+0+2+0+0.5+0.25 = 10.75_{10}$

Quiz

- Convert each of the following base 10 representations to its equivalent binary form:
  - 32
  - 15
  - 27
  - 53

Quiz

- Convert each of the following binary representations to its equivalent base 10 form:
  - 11.01
  - 101.111
  - 10.1
  - 110.011
  - 0.101

Quiz

- Express the following values in binary notation:

a. $4\frac{1}{2}$ **b.** $2\frac{3}{4}$ **c.** $1\frac{1}{8}$ **d.** $\frac{5}{16}$ **e.** $5\frac{5}{8}$

Quiz

- Convert to Binary
  - 21.125_{10}

STORING INTEGERS

Unsigned Integers

- Representations of quantities are always positive
- Ex: height, weight, ASCII code, etc.
- All bits are used for value
- Max value of 1 unsigned byte:
  - $1111\ 1111_2 = 2^8 - 1 = 255_{10}$
- Max value of 1 unsigned word (2 bytes):
  - $1111\ 1111\ 1111\ 1111_2 = 2^{16} - 1 = 65535_{10}$

Signed Integers

- Storing/representing of negative and positive integers
- The leftmost bit is used to represent the sign
- Ex:
  - 0 positive: 0101 0011
  - 1 negative: 1101 0011
- Negative integers stored in machine as **two's complement** or **excess notation**

Two's complement notation systems

- Leftmost bit: sign bit

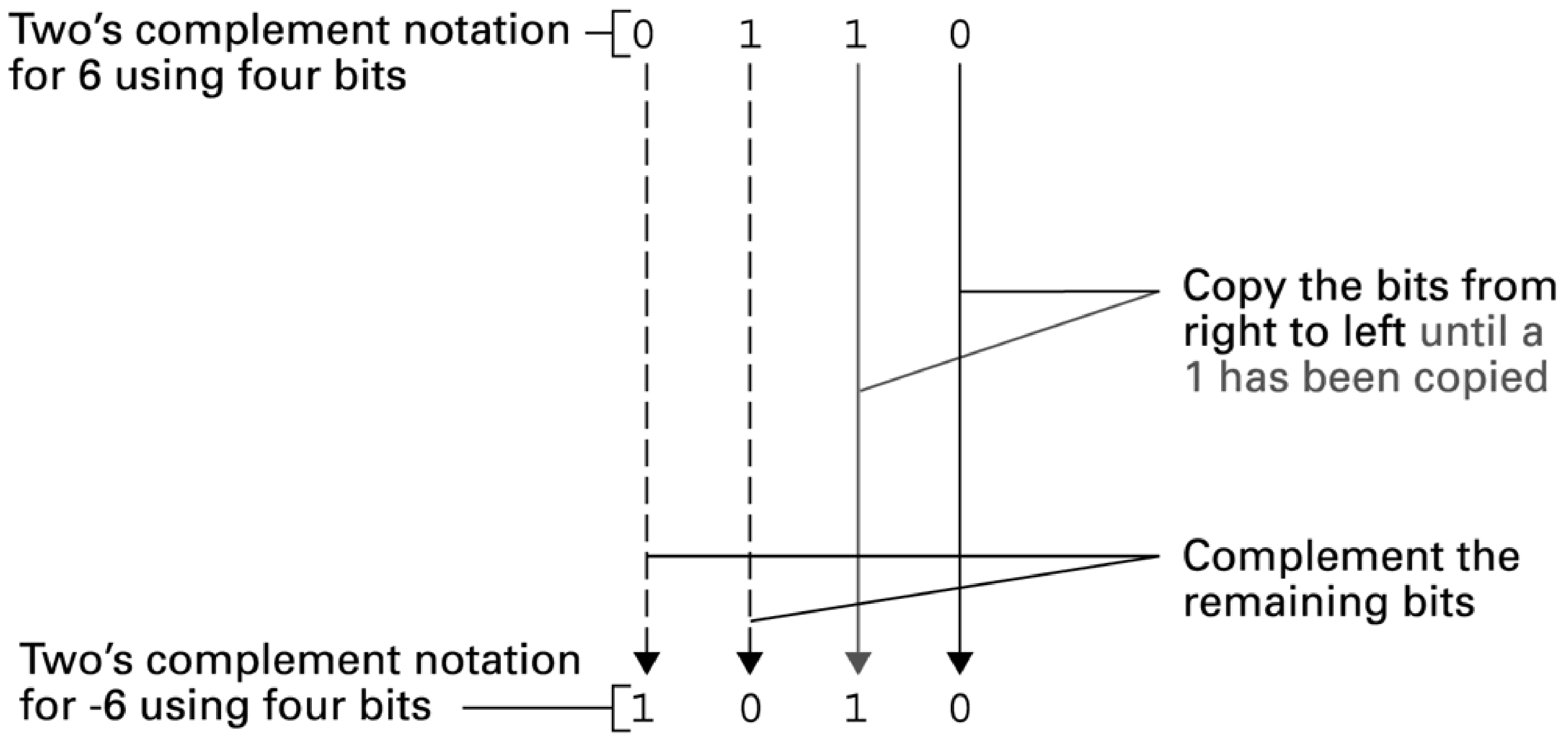
- One's complement :
  - Changing all the 0s to 1s, all the 1s to 0s.
  - Ex: 0110 and 1001 are complements

- Two's complement: add 1 to the one's complement
  - Ex: 1001 + 0001 = 1010

One's and two's complements

5 (1 byte)		0	0	0	0	0	1	0	1
One's complement of 5		1	1	1	1	1	0	1	0
+									1
Two's complement of 5		1	1	1	1	1	0	1	1
+ 5		0	0	0	0	0	1	0	1
Result	1	0	0	0	0	0	0	0	0

Coding the value -6 in two's complement notation using 4 bits



Two's complement notation systems

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

Addition problems converted to two's complement notation

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

Quiz

- Convert each of the following two's complement representations to its equivalence base 10 form:
  - 00011
  - 01111
  - 11100
  - 11010
  - 00000
  - 10000

Quiz

- Convert to two's complement form using patterns of 8 bits:
  - 6
  - -6
  - -17
  - 13
  - -1
  - 0

Quiz

- What are the largest and smallest numbers that can be stored if the machine uses bit patterns of the following lengths?
  - A. four
  - B. six
  - C. eight

Min and max of signed integers

N bits	minimum	maximum
8	-2^7 = **-128**	$2^7 - 1$ = **+127**
16	-2^{15} = **-32,768**	$2^{15} - 1$ = **+32,767**
32	-2^{31} = **-2,147,483,648**	$2^{31} - 1$ = **+2,147,483,647**
64	-2^{63} = **-9,223,372,036,854,775,808**	$2^{63}-1$ = **+9,223,372,036,854,775,807**

Min and max of unsigned integers

n	Minimum	Maximum
8	0	$2^8 - 1 =$ **255**
16	0	$2^{16} - 1 =$ **65,535**
32	0	$2^{32} - 1 =$ **4,294,967,295**
64	0	$2^{64} - 1 =$ **18,446,744,073,709,551,615**

Excess

- Another representation of signed integers
- Sign bit – opposite of two's compement
  - 1 – positive
  - 0 – negative
- Leftmost bit is 1 → number 0
  - 1000 → 0 (in 4 bits)
  - 100 → 0 (in 3 bits)

Excess

Two's complement

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

Excess

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

Excess

Two's complement

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

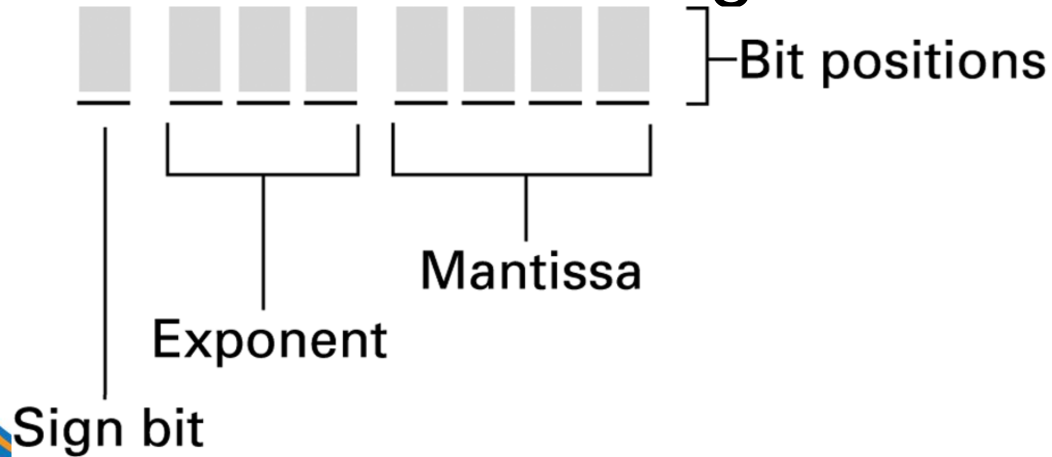
Excess

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

STORING FRACTIONS

Fractions

- Floating-points:
  - Consisting of a sign bit, a mantissa field, and an exponent field.
  - Radix point: used to separate integer part and the fraction part of a number
- Mantissa: also fraction/significant



Example 1

- **01101011**
  - Signed bit: 0
  - Exponent: 110
  - Mantissa: 1011

  - .1011
  - 110 = 2 (excess with 3-bit)
  - 10.11 = **2 ¾**

Example 2

- 00111100
  - Signed bit: 0
  - Exponent: 011
  - Mantissa: 1100

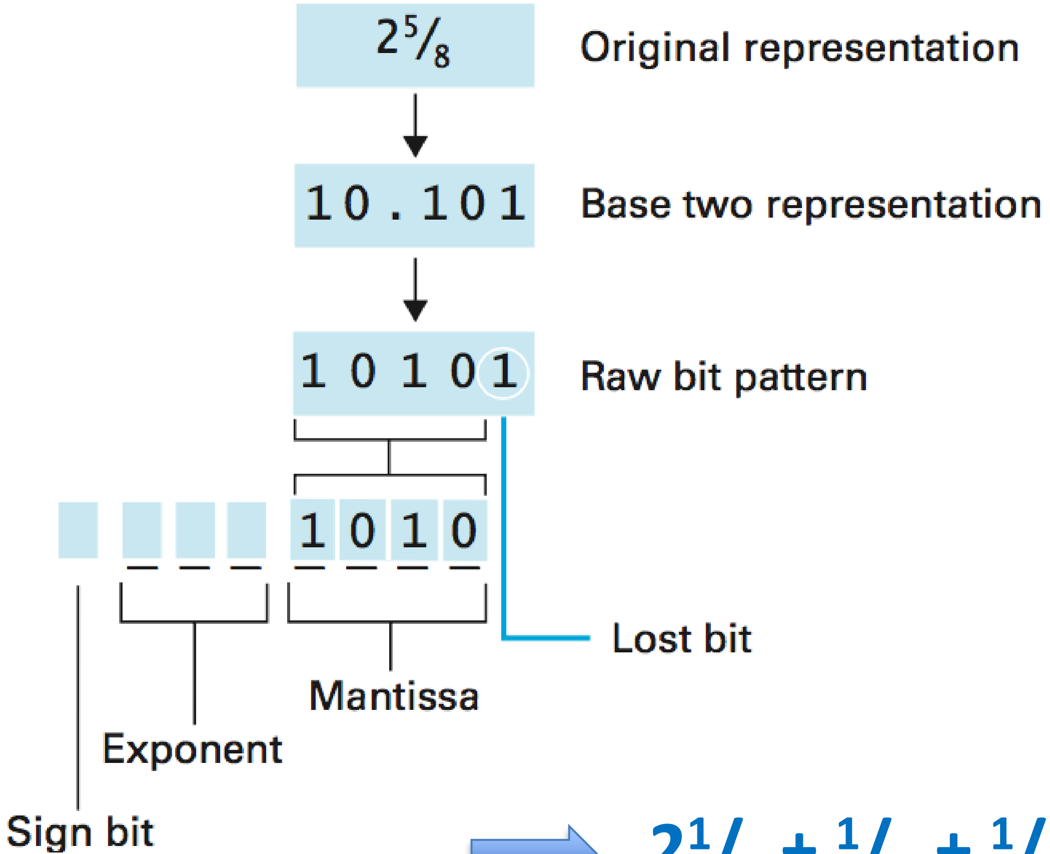
  - .1100
  - 011 = -1 (excess with 3-bit)
  - .01100 = 3/8

Example 3

- Encode 1⅛
  - Convert to binary 1.001
  - Mantissa: _ _ _ _ 1 0 0 1
    - From left to right, start with leftmost bit 1
  - Radix point: from .1001 to 1.001
  - → Exponent: +1 = 101 (excess 3-bit)
  - Signed bit: 0

  - 1 ⅛ = **01011001**

Encoding the value 2 5⁄8 → Truncation error



0 110 1010

↓

$2^1/_2$

$\Rightarrow$ $2^1/_2 + {}^1/_8 + {}^1/_8 = ?$

Floating points

- Single precision floating point (32 bits)
  - 1 bit (sign) + 8 bits (exponent) + 23 bits (mantissa)
  - Value: from 10^{-37} to 10^{38}
  - Precision: 7 decimal fraction

- Double precision floating point (64 bits)
  - 15 decimal fraction

Fractions

- Single-precision : 32 bits

s	exp	frac
1	8-bits	23-bits

- Double-precision : 64 bits

s	exp	frac
1	11-bits	52-bits

- Extend-precision : 80 bits (Intel only)

s	exp	frac
1	15-bits	63 or 64-bits

IEEE-754 32-bit Single-Precision Floating-Point Numbers

- IEEE-754:

  N = (-1)^S x 1.F x 2^(E-127)

  - S: Sign bit
  - F: Fraction part
  - E: Exponent part
  - $127 = 2^{8-1}$ (excess 8-bit)

Example 1

- 0 10000000 110 0000 0000 0000 0000 0000

```
Sign bit S = 0 ⇒ positive number
E = 1000 0000B = 128D
Fraction is 1.11B (with an implicit
leading 1) = 1 + 1×2^-1 + 1×2^-2 = 1.75D

The number is +1.75 × 2^(128-127) = +3.5D
```

Example 2

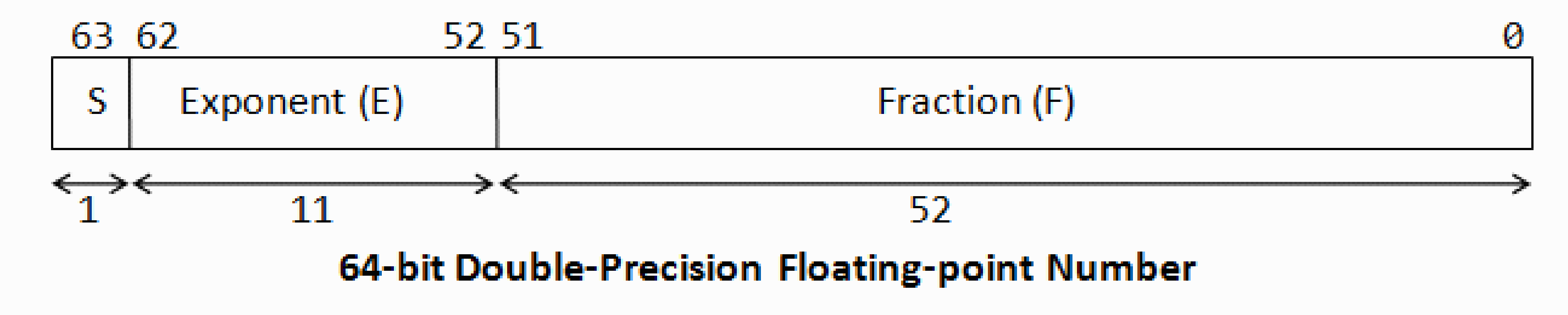
- 1 01111110 100 0000 0000 0000 0000 0000

```
Sign bit S = 1 ⇒ negative number
E = 0111 1110B = 126D
Fraction is 1.1B (with an implicit leading 1) =
1 + 2^-1 = 1.5D

The number is -1.5 × 2^(126-127)=-0.75D
```

IEEE-754 64-bit Double-Precision Floating-Point Numbers

- Exponent: excess – 1023
- Fraction: implicit leading bit (before radix point).
  - N = (-1)^S x 1.F x 2^(E-1023)



Min and Max of Floating-Point Numbers

Precision	Min	Max
Single	**1.1754×10^{-38}**	**3.40282×10^{38}**
Double	**2.2250×10^{-308}**	**1.7976×10^{308}**

COMMUNICATION ERRORS

Communication error

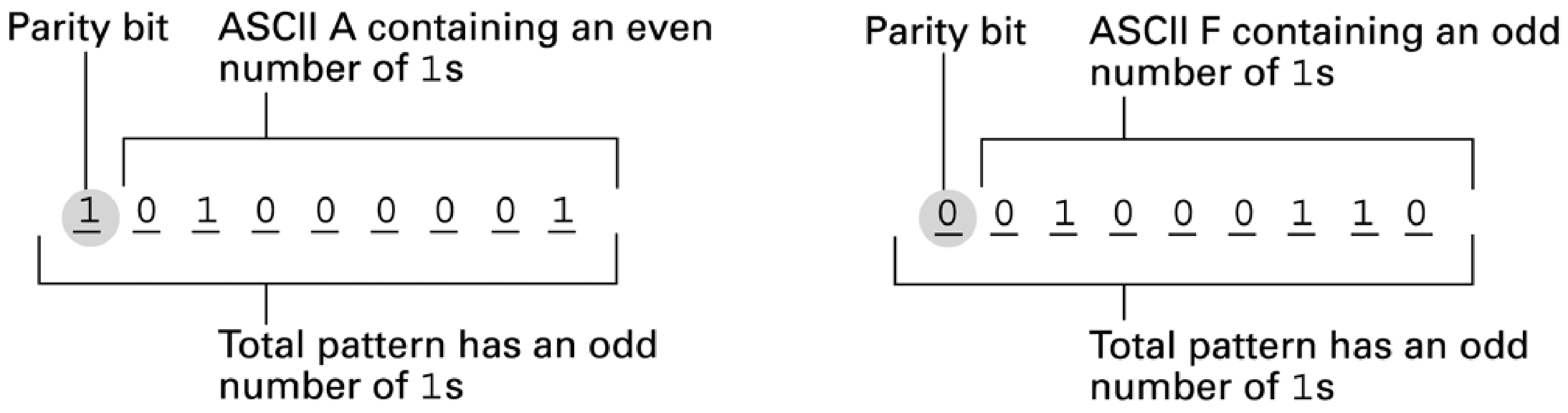
- Happening when:
  - Data transferred between components in computer
  - Storing data
- Bits received are not the same with originals
- Reasons:
  - Dirt on disc surface
  - Broken circuit makes reading / writing inaccurate
  - Data transmission line broken
  - Radiation changes the sequence of bits on main memory

Techniques

- Parity bits (even vs odd)
- Checkbytes
- Error correcting codes

Parity bits

- Error detection is based on: odd-number of bits 1 and even-number of bits 1 is found → there must be an error.



Checkbyte

- Set of parity bits
- Each parity bit is scattered in the bit patterns
  - For example, a parity bit is associated with every eighth bit in the bit patterns

- Checksum

Error-correcting code

- Hamming distance (of two bit patterns):
  - Number of bits in which the patterns differ.

- Example:
  - Hamming(00**0000**, 00**1111**) = 4
  - Hamming(**10**10**1**100, **01**10**0**100) = 3

Error-correcting code

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

Error-correcting code

Character	Code	Pattern received	Distance between received pattern and code
A	0 0 0 0 0 0	0 **1** 0 **1** 0 0	2
B	0 0 1 1 1 1	0 **1** **0** 1 **0** **0**	4
C	0 1 0 0 1 1	0 1 0 **1** **0** **0**	3
D	0 1 1 1 0 0	0 1 **0** 1 0 0	**1** — Smallest distance
E	1 0 0 1 1 0	**0** **1** 0 1 **0** 0	3
F	1 0 1 0 0 1	**0** **1** **0** **1** 0 **0**	5
G	1 1 0 1 0 1	**0** 1 0 1 0 **0**	2
H	1 1 1 0 1 0	**0** 1 **0** **1** **0** 0	4

D is the answer because it has smallest distance

Quiz

- The following bytes were originally encoded using odd parity. In which of them do you know that an error has occurred?

a. 100101101

b. 100000001

c. 000000000

d. 111000000

e. 011111111

Quiz

- Using the error-correcting code presented before, decode the following messages:

a. 001111 100100 001100

b. 010001 000000 001011

c. 011010 110110 100000 011100

Quiz

- Construct a code for the characters A, B, C, D using bit patterns of length five so that the Hamming distance between any two patterns is at least three.

MASS STORAGE

Mass Storage

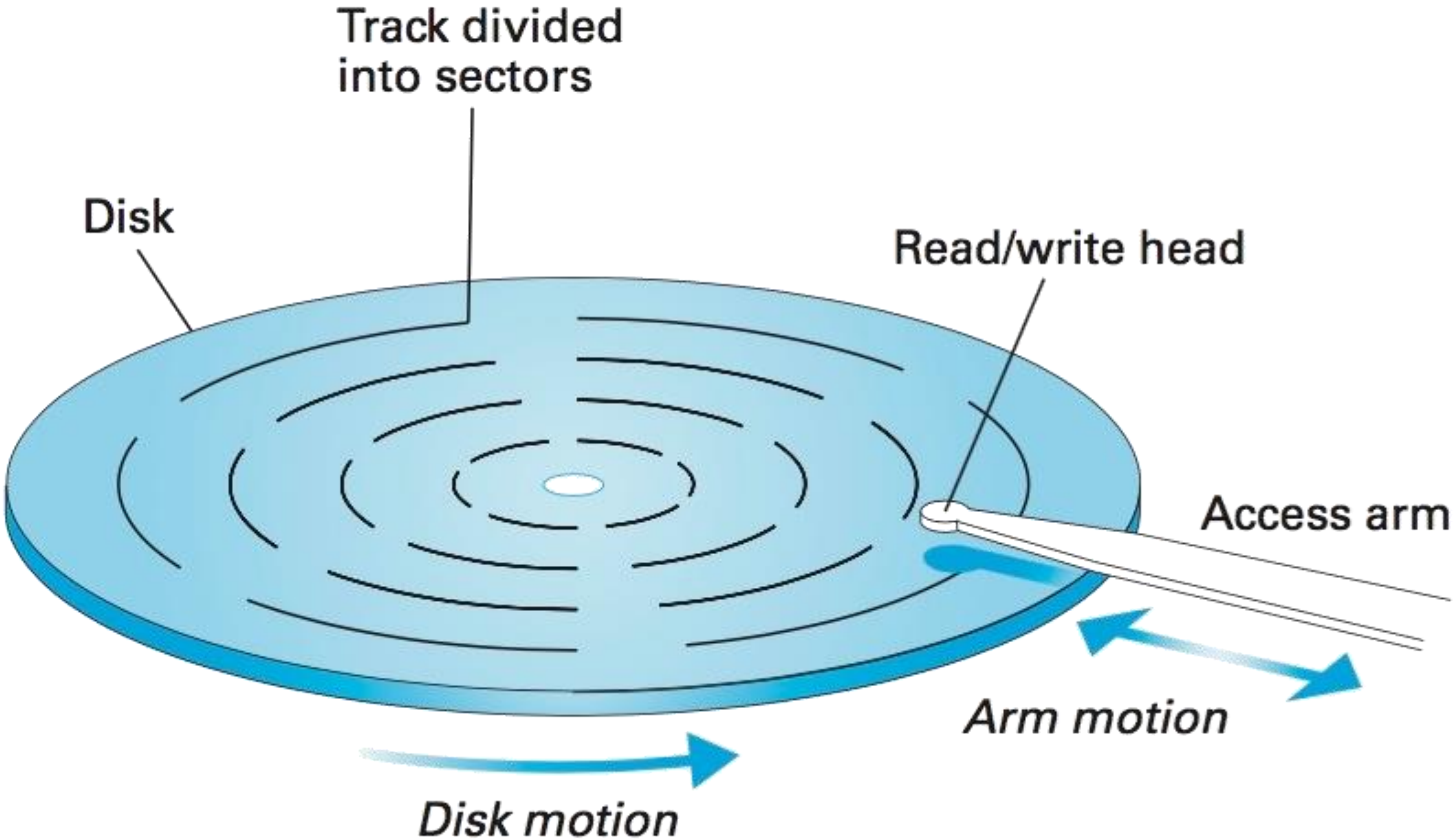
- On-line versus off-line
- Typically larger than main memory
- Typically less volatile than main memory
- Typically slower than main memory
- Typically lower cost than main memory

Mass Storage Systems

- Magnetic Systems
  - Disk
  - Tape
- Optical Systems
  - CD
  - DVD
- Flash Technology
  - Flash Drives
  - Secure Digital (SD) Memory Card


8 THS
MEQUON, WISCONSIN U.S.A.

Magnetic disk storage system



Magnetic disk storage system

- Seek time
  - Time needed to position the read/write head over the correct track
- Latency
  - Time for the beginning of the desired sector to rotate under the read/write head
- Transfer time
  - Time for the entire sector to pass under the read/write head and have its contents read into or written from memory

Magnetic disk storage system

- Given:
  - Rotation speed = 7,200 rev/min=120 rev/sec = 8.33 msec/rev
  - Arm movement time = 0.02 msec to move to an adjacent track
  - On average, the read/write head must move about 300 tracks
  - Number of tracks/surface = 1,000
  - Number of sectors/track = 64
  - Number of bytes/sector = 1,024
- Seek time
  - Best case = 0 msec;
  - Worst case = 999*0.02=19.98 msec
  - Average case = 300*0.02 = 6 msec

Magnetic disk storage system

- Given:
  - Rotation speed = 7,200 rev/min=120 rev/sec = 8.33 msec/rev
  - Arm movement time = 0.02 msec to move to an adjacent track
  - On average, the read/write head must move about 300 tracks
  - Number of tracks/surface = 1,000
  - Number of sectors/track = 64
  - Number of bytes/sector = 1,024
- Latency
  - Best case = 0 msec;
  - Worst case = 8.33 msec
  - Average case = 4.17 msec

Magnetic disk storage system

- Given:
  - Rotation speed = 7,200 rev/min=120 rev/sec = 8.33 msec/rev
  - Arm movement time = 0.02 msec to move to an adjacent track
  - On average, the read/write head must move about 300 tracks
  - Number of tracks/surface = 1,000
  - Number of sectors/track = 64
  - Number of bytes/sector = 1,024
- Transfer time
  - 1/64 * 8.33 msec = 4.17 msec

Magnetic tape storage