16.070

Introduction to Computers & Programming

Machine architecture:
data storage, memory organisation, logic gates

Prof. Kristina Lundqvist
Dept. of Aero/Astro, MIT

Chapter Summary: B1

- Chapter 1 presents the rudiments of data storage within digital computers.
- It introduces the basics of digital circuitry and how a simple flip-flop can be used to store a single bit.
- It then discusses addressable memory cells and mass storage systems (magnetic disk, compact disks, and magnetic tape).
- The chapter then discusses how information (text, numeric values, images, and sound) are encoded as bit patterns. The optional sections delve more deeply into these topics by presenting the problems of overflow errors, truncation errors, error detection and correction techniques, and data compression.

Bits and Their Storage

- Bit (binary digit)
  - Two symbols: 0 and 1
- Boolean operations
  - AND, OR, XOR, and NOT

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

Pictorial Representation of AND and OR

AND

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

OR

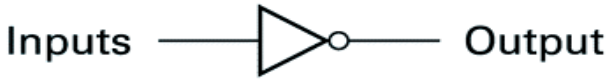
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1

Pictorial Representation of XOR and NOT

XOR

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

NOT



Inputs	Output
0	1
1	0

NAND / NOR

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

DeMorgans's Theorem

$\overline{A + B} = \overline{A}\,\overline{B}$ and $\overline{AB} = \overline{A} + \overline{B}$

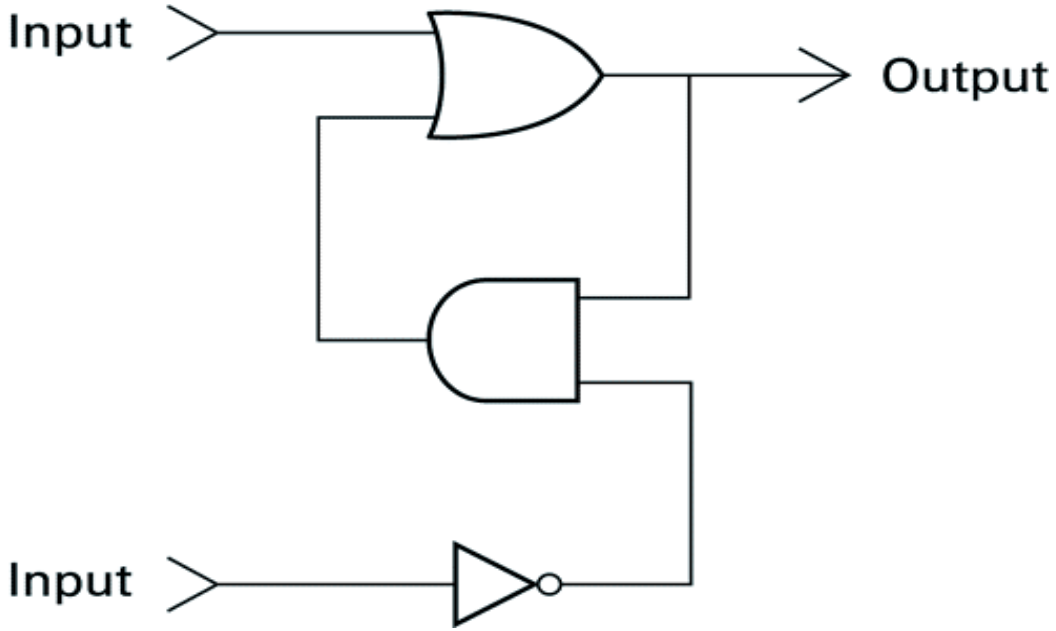
http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/nor.html#c1

Boolean Algebra Theorems

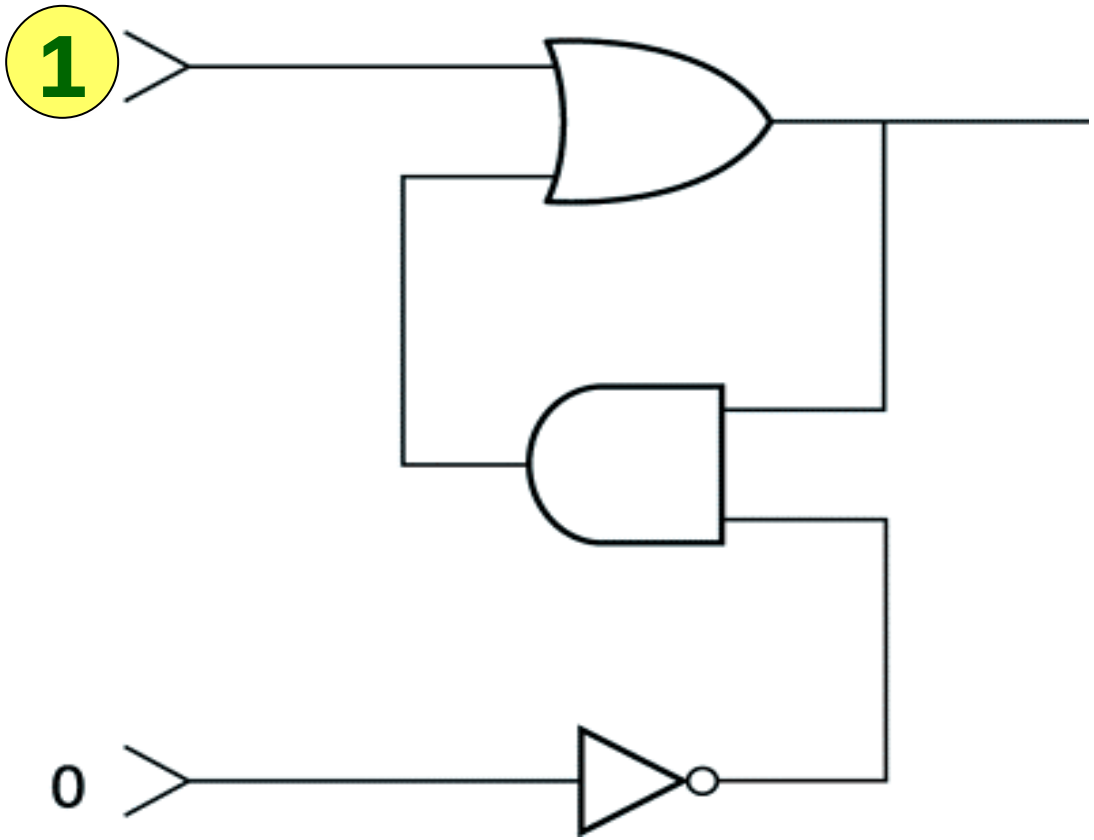
$ABC=(AB)C=A(BC),\ A+B+C=(A+B)+C=A+(B+C)$	AND, OR are associative
$AB=BA,\ A+B=B+A$	AND, OR operations are commutative
$A+BC=(A+B)(A+C),\ A(B+C)=AB+AC$	Forms of the distributive property
$\overline{A+B}=\bar{A}\bar{B}$	a form of DeMorgan's Theorem
$\overline{AB}=\bar{A}+\bar{B}$	a form of DeMorgan's Theorem
$AA=A,\ A+A=A,\ \bar{A}+A=1,\ A\bar{A}=0,\ A=\bar{\bar{A}}$	Single Variable Theorems
$A+AB=A,\ A+\bar{A}B=A+B$	Two-variable theorems
$A1=A,\ A+1=1,\ A+0=A,\ A0=0,\ \bar{1}=0,\ \bar{0}=1$	Identity and Null operations

A Simple Flip-flop Circuit

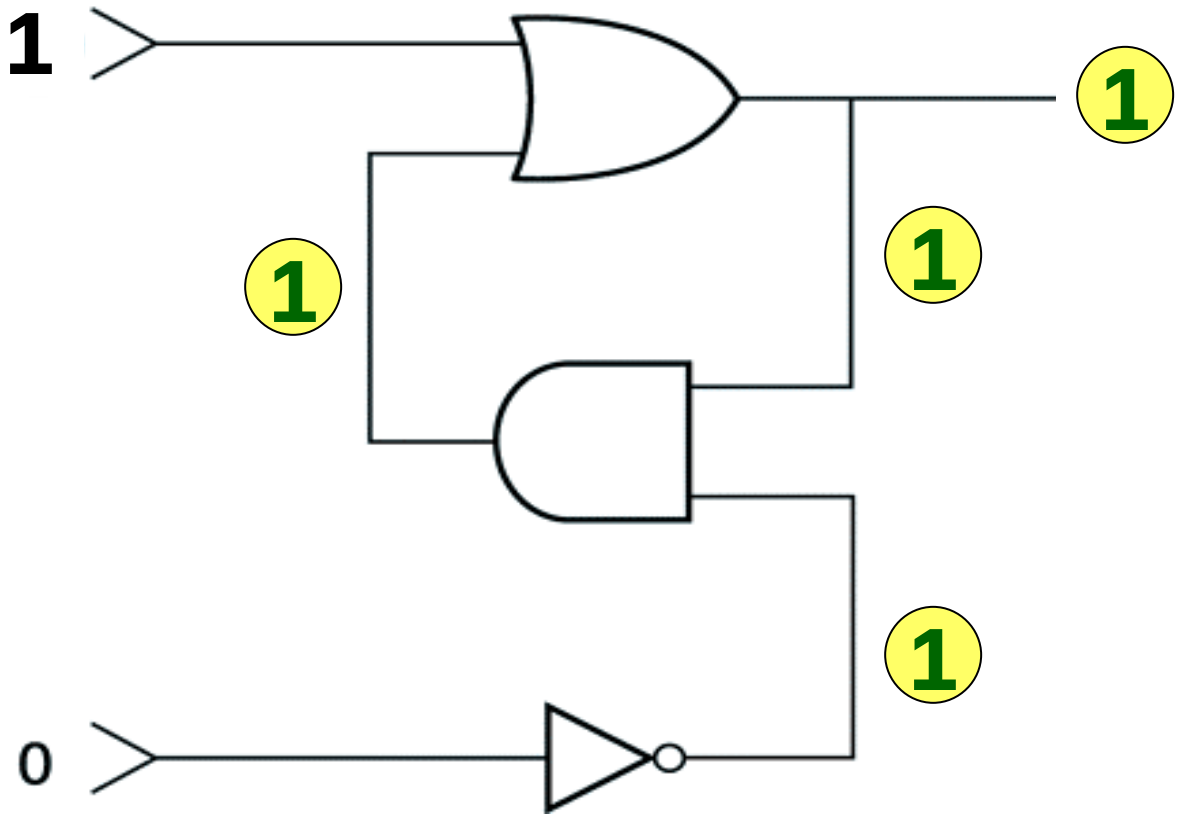
- A flip-flop
  - The output will flip or flop between two values under control of external stimuli



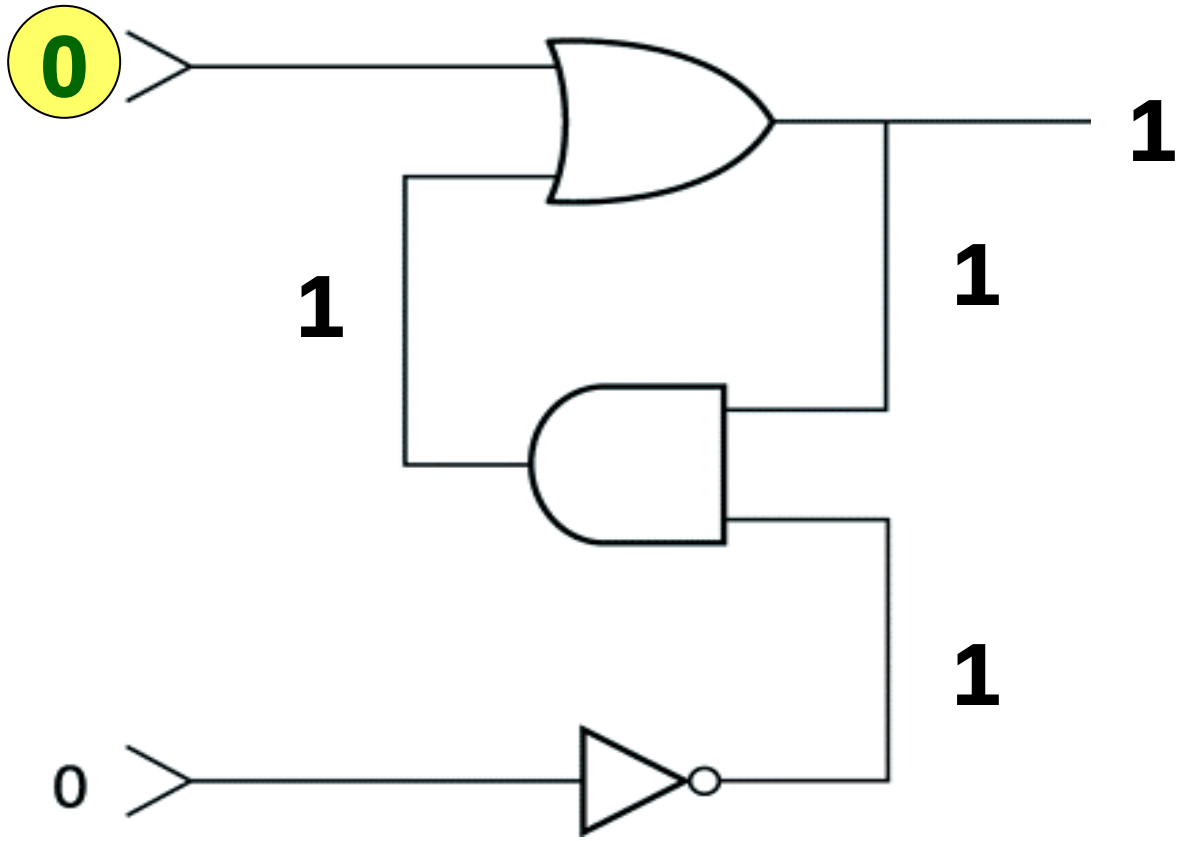
Setting the Output of a Flip-Flop to 1



Setting the Output of a Flip-Flop to 1

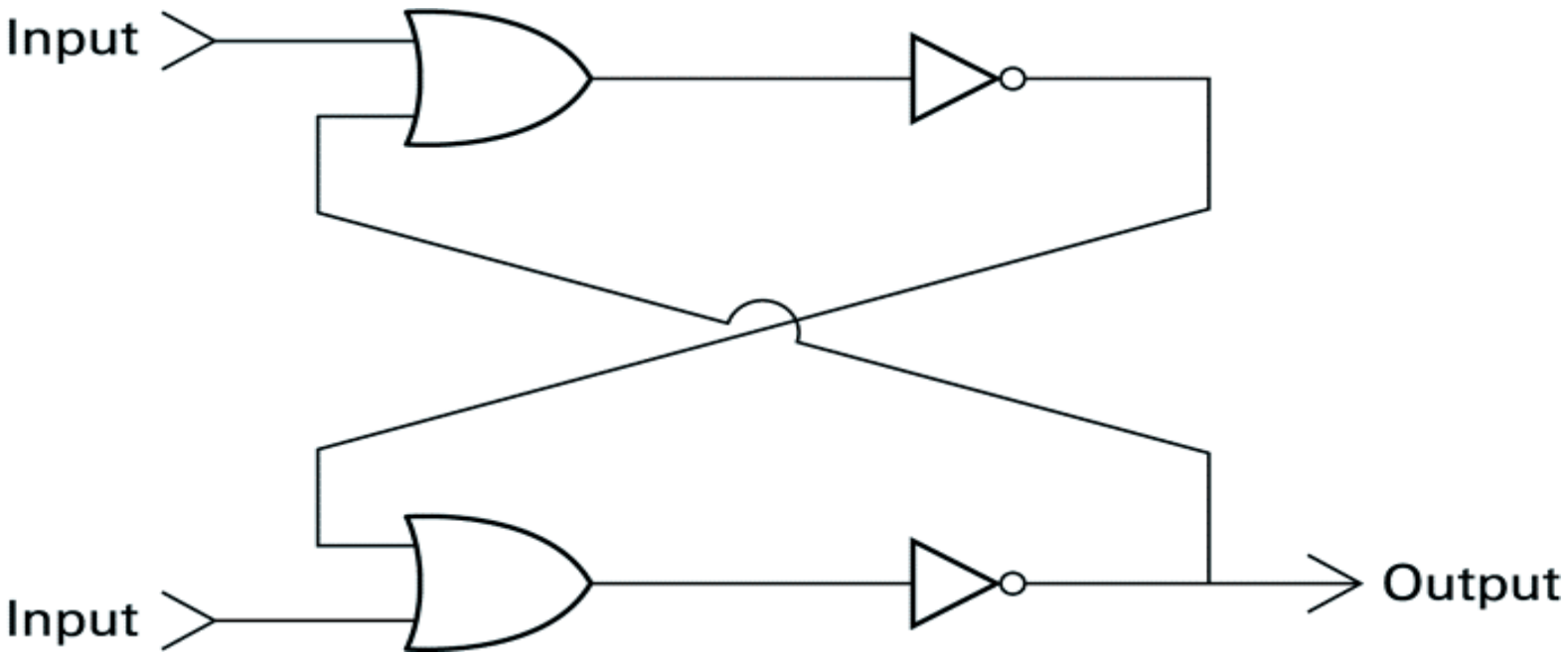


Setting the Output of a Flip-Flop to 1



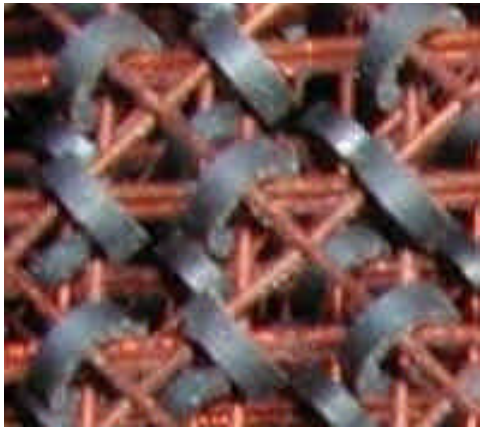
Another way of constructing a flip-flop

- Alternative way to build flip-flop



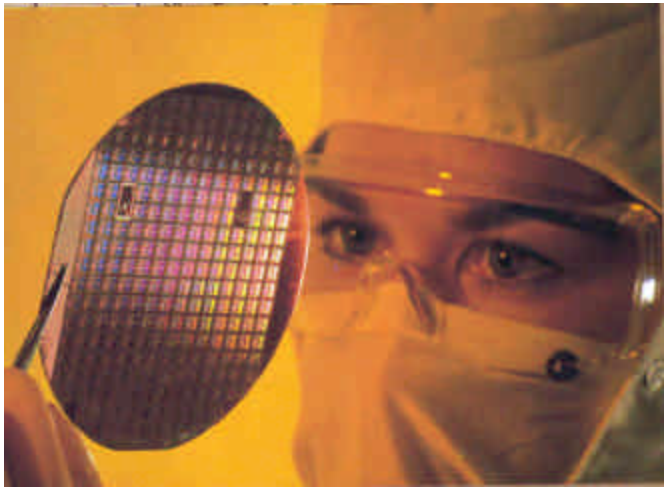
Other Storage Techniques

- Cores



http://www.fortunecity.com/marina/reach/435/coremem.htm

- DRAM (Capacitors, transistors)



http://www.cordis.lu/nanotechnology/icons/count_projectsilicon-wafer.jpg

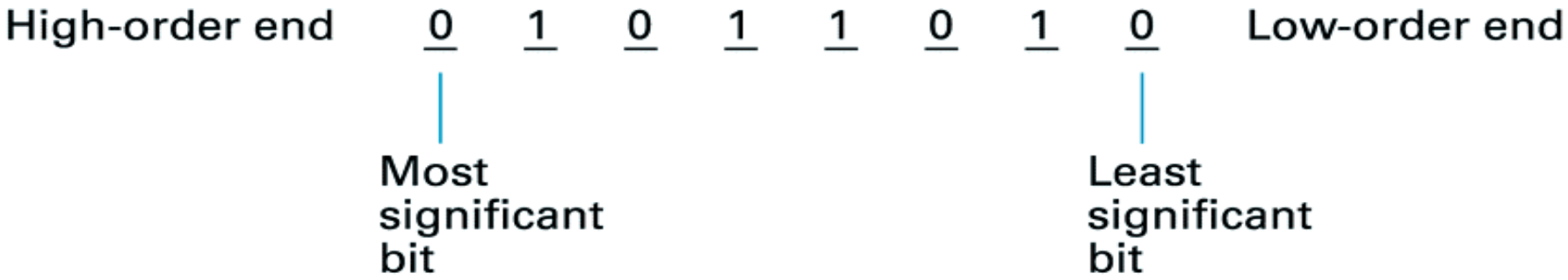
The Hexadecimal Coding System

- Hex is often used in processing telemetry from a spacecraft
- Data transmitted from the spacecraft to the ground is a stream of bits – 1's and 0's: 000101100000011100001011 10101101110010101011 …

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

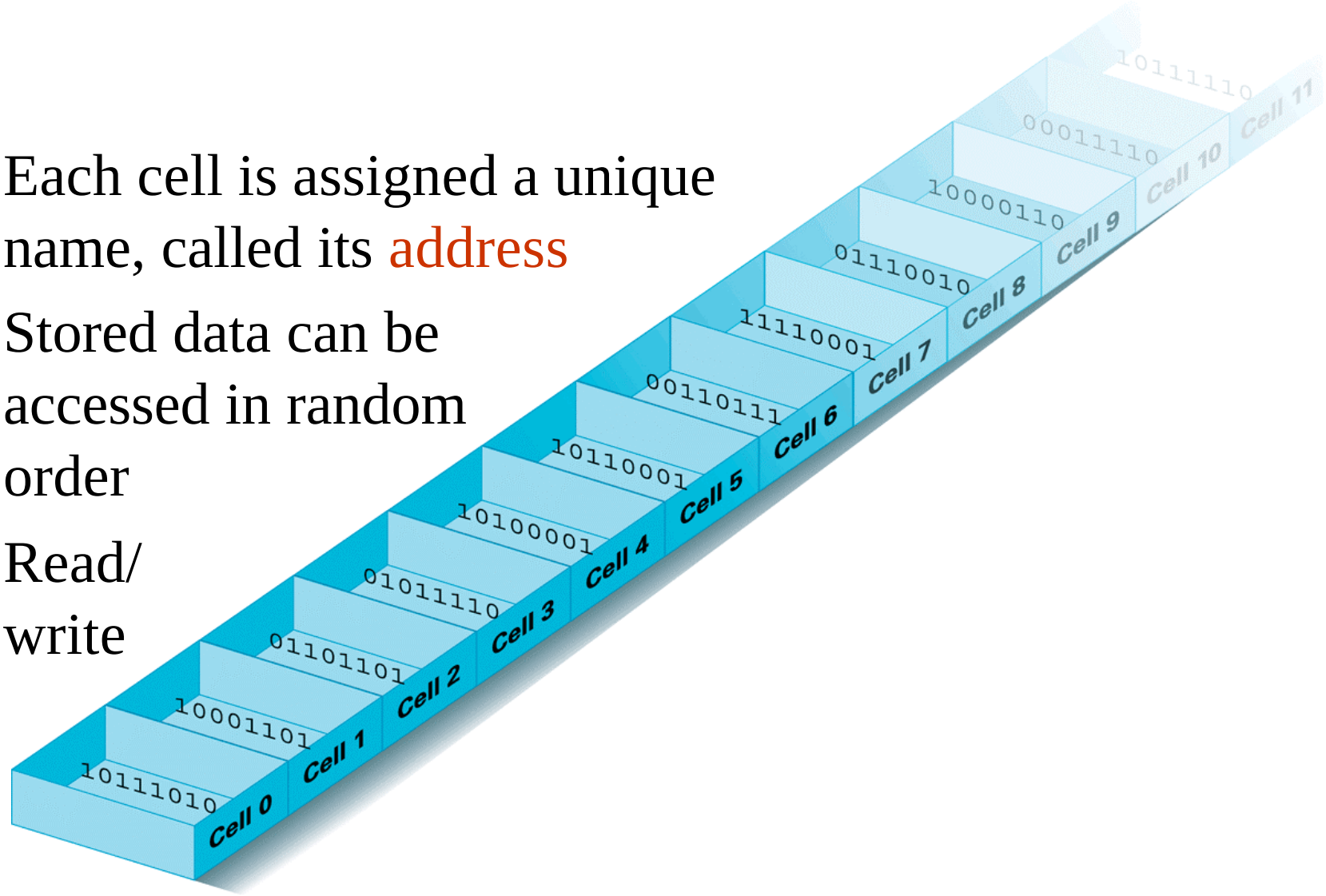
Main Memory

- Main memory arranged in manageable units called: Cells
  - Typically 8 bits: Byte



Memory Cells

- Each cell is assigned a unique name, called its address
- Stored data can be accessed in random order
- Read/ write



Little / Big-Endian

- 00000000 00000000 00000100 00000001

Address	Big-Endian repr. of 1025	Little-Endian repr. of 1025
00	0000 0000	0000 0001
01	0000 0000	0000 0100
02	0000 0100	0000 0000
03	0000 0001	0000 0000

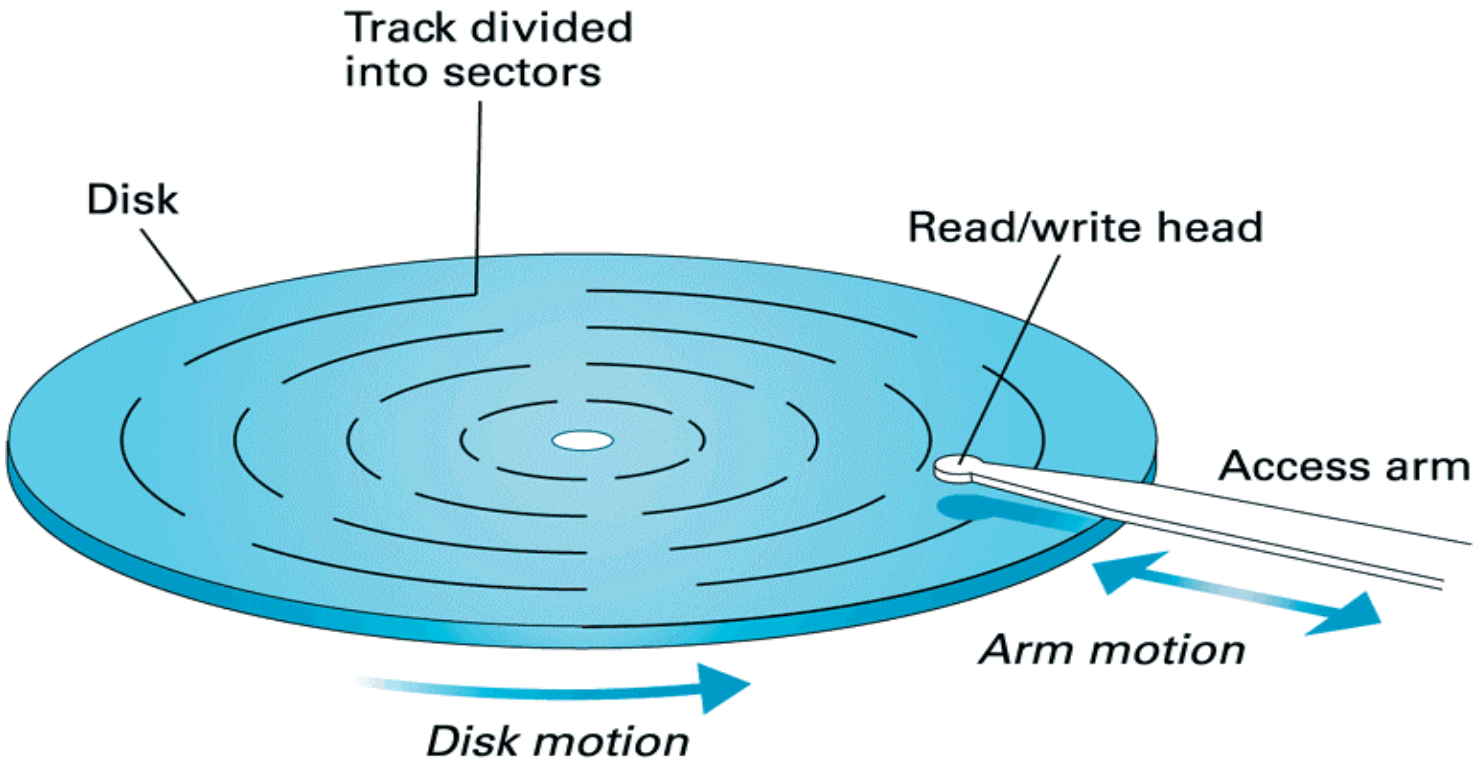
Memory Capacity

- Main memory systems usually has total number of cells as a power of two
  - 2^{10} = 1024 = KB
  - 2^{20} = 1,048,576 = MB
  - 2^{30} = 1,073,741,824 = GB

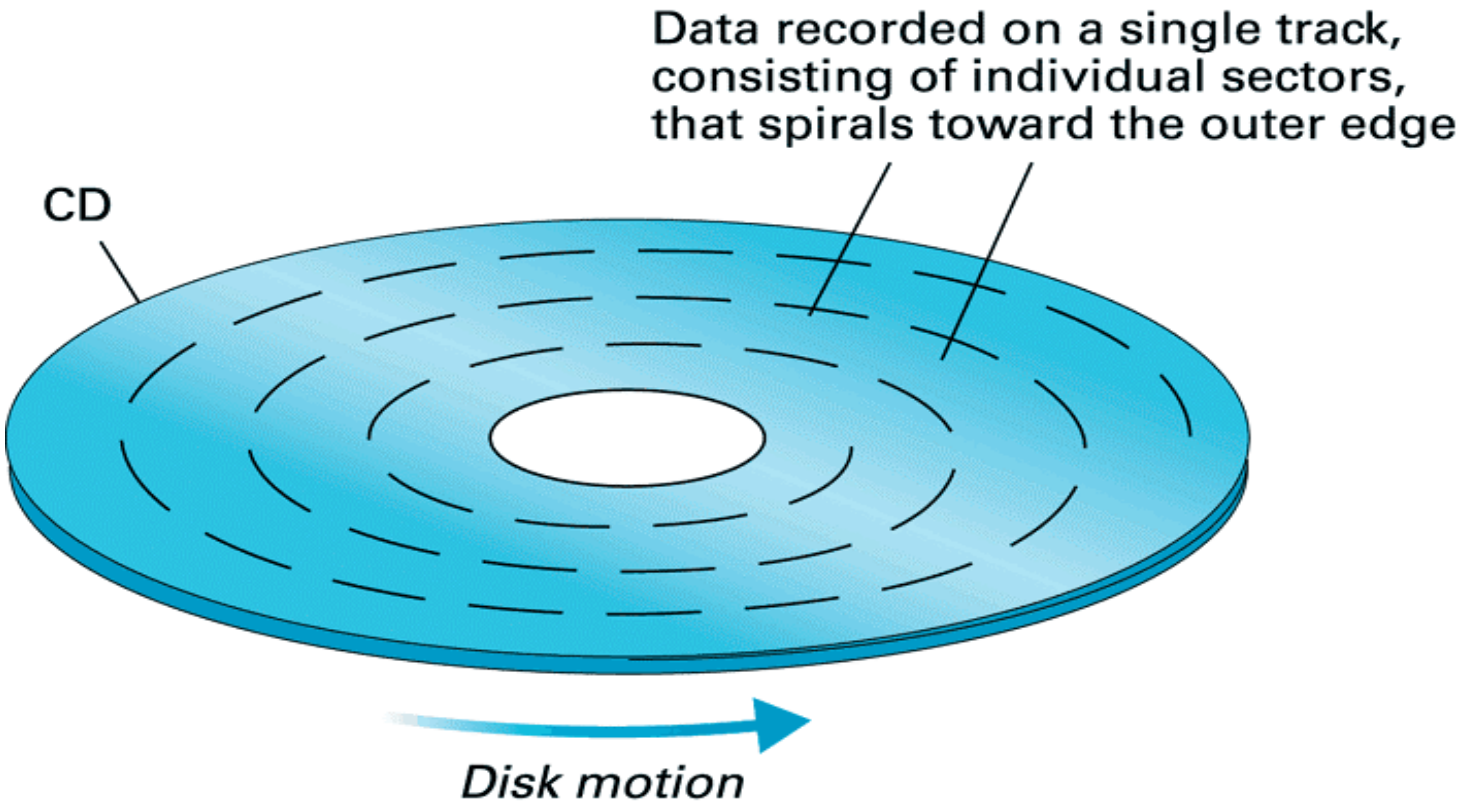
Mass Storage

- Mass storage systems
  - Magnetic disk, CD, Magnetic tape
  - Less volatility
  - Large storage capacity
  - Remove form machine for archival purpose

Memory Cells Arranged by Address



CD Storage Format



Information as Bit Patterns

- Representing text, numeric values, images, sound
- Text
  - (Extended) ASCII (American Standard Code for Information Interchange)

Character on the screen	Binary value used to process it	Character on the screen	Binary value used to process it
1	0110001	A	1000001
2	0110010	B	1000010
3	0110011	C	1000011
4	0110100	D	1000100
5	0110101	E	1000101

http://tronweb.super-nova.co.jp/characcodehist.html

  - EBCDIC (Extended Binary Coded Decimal interchange Code)
  - Unicode
  - ISO standards

ASCII

ASCII	Hex	Symbol
0	0	NUL
1	1	SOH
2	2	STX
3	3	ETX
4	4	EOT
5	5	ENQ
6	6	ACK
7	7	BEL
8	8	BS
9	9	TAB
10	A	LF
11	B	VT
12	C	FF
13	D	CR
14	E	SO
15	F	SI

...

ASCII	Hex	Symbol
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

...

ASCII	Hex	Symbol
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o

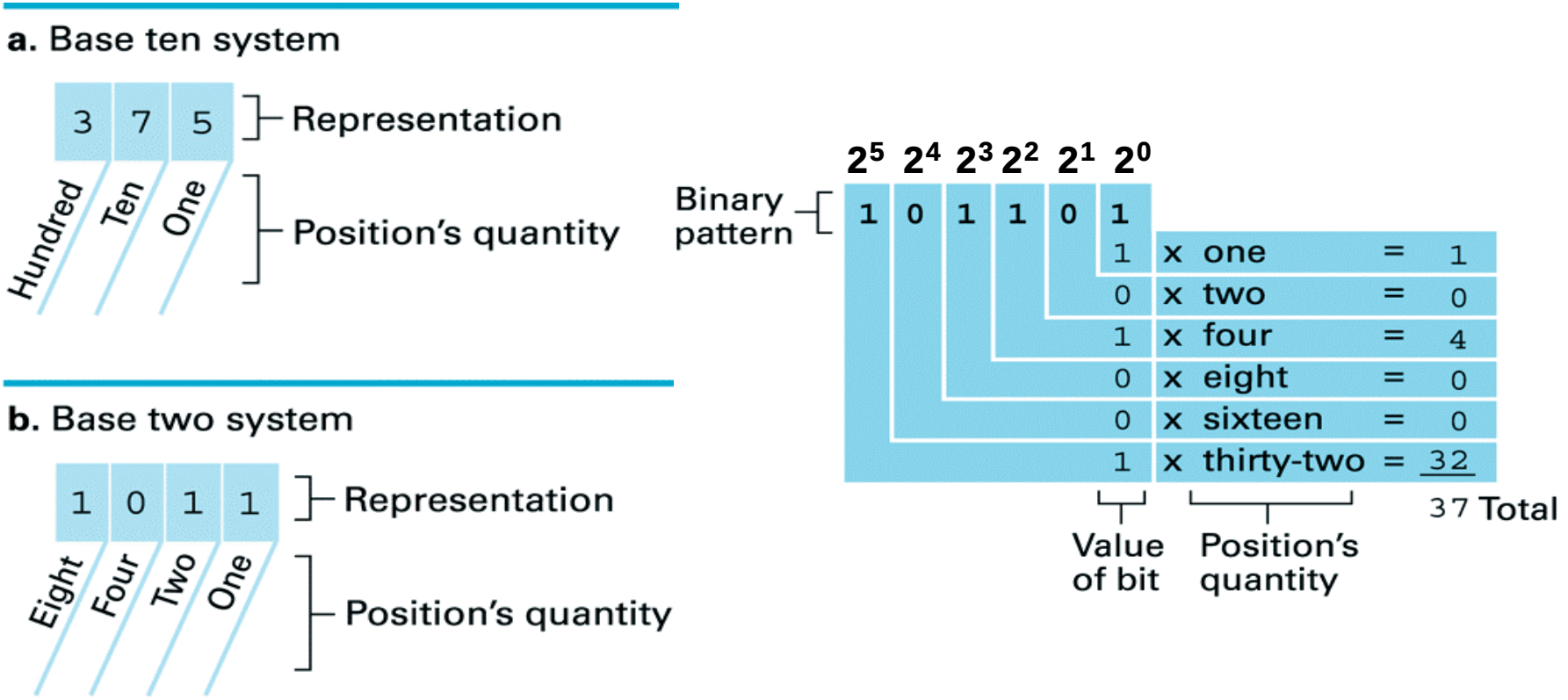
01101000 01100101 01101100 01101100 01101111

Numeric Values

- Storing the value of 25_{10} using ASCII:

  00110010 00110101

- Binary notation: 00000000 000011001_2



Finding Binary Representation of Large Values

1. Divide the value by 2 and record the remainder
2. As long as the quotient obtained is not 0, continue to divide the newest quotient by 2 and record the remainder
3. Now that a quotient of 0 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