Computer Science
North University of China

CS101 Fall 2017

Introduction to
Computer Science

Instructor: Qingsong Guo
School of Computer Science & Technology

`http://abelgo.cn/cs101.html`

Lecture 2, Lecture 3:
Welcome to the Digital World !

Think in Data: Data Storage

Bits, Data Representation, and Binary System

What types of data can you recognize in your daily life?

Think in Data: Data Storage

01 Bits and Their Storage

02 Representing Information as Bit Patterns

03 The Binary System

04 Storing Integers & Fractions

05 Main Memory & Mass Storage

1.
Bits and Their Storage

Hardware Advances

Data storage relies on the following techniques

- Electronics (电), magnetics (磁), optics (光)

Representing information as electrical signals led to telegraph in mid-1800s (电报, 电信号, 电存储)

- needed device to control current flow
- telegraph clicker, relays, vacuum tubes

Magnetic storage - 1878 (磁介质, 磁存储)

- magnetic tape, hard disk

Photography – end of 18th century (摄影术, 光存储)

- based on principle that some chemicals change their properties when exposed to light
- silver nitrate changes to metallic silver
- optical storage such as CD & DVD

Bits and Bit Patterns

Bit

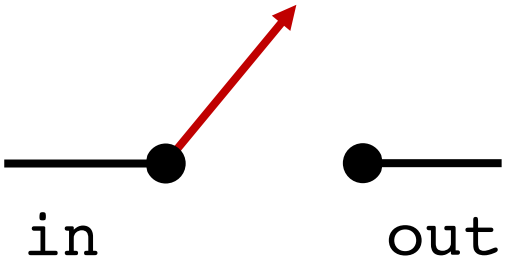
- Binary Digit (0 or 1)

Bit Patterns are used to represent information

- Numbers: positive integer, negative integer, fraction, etc.
- Text characters
- Images
- Sound
- And others

Switches(开关)

An open switch



in	out
0	0
1	0

A closed switch

in	out
0	0
1	1

Boolean Operations

Boolean operation

- An operation that manipulates one or more true/false values and generate a true/false output

Specific operations

- AND
- OR
- XOR (exclusive or)
- NOT

The 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

Logical Table or Truth Table (真值表)

NOT/Negation (¬)

P	$\neg P$
0	1
1	0

AND (*, ∧)

P*Q	Q = 0	Q = 1
P = 0	0	0
P = 1	0	1

OR(+, ∨)

P+Q	Q = 0	Q = 1
P = 0	0	1
P = 1	1	1

XOR (⊕)

P ⊕ Q	Q = 0	Q = 1
P = 0	0	1
P = 1	1	0

Gates(逻辑门)

Gate: A device that computes a Boolean operation

- Often implemented as (small) electronic circuits
- Provide the building blocks from which computers are constructed
- AND, OR, XOR (Exclusive OR) , NOT

VLSI (Very Large Scale Integration**)**

- Density of integration
- Integrate a huge number of gates into a Chip(芯片)

Gates and Their Computations

AND

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

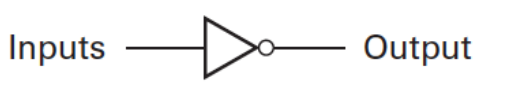
OR

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

Logical Expression and XOR

Logical expression of XOR

$$P \oplus Q = (\neg P * Q) + (P * \neg Q)$$

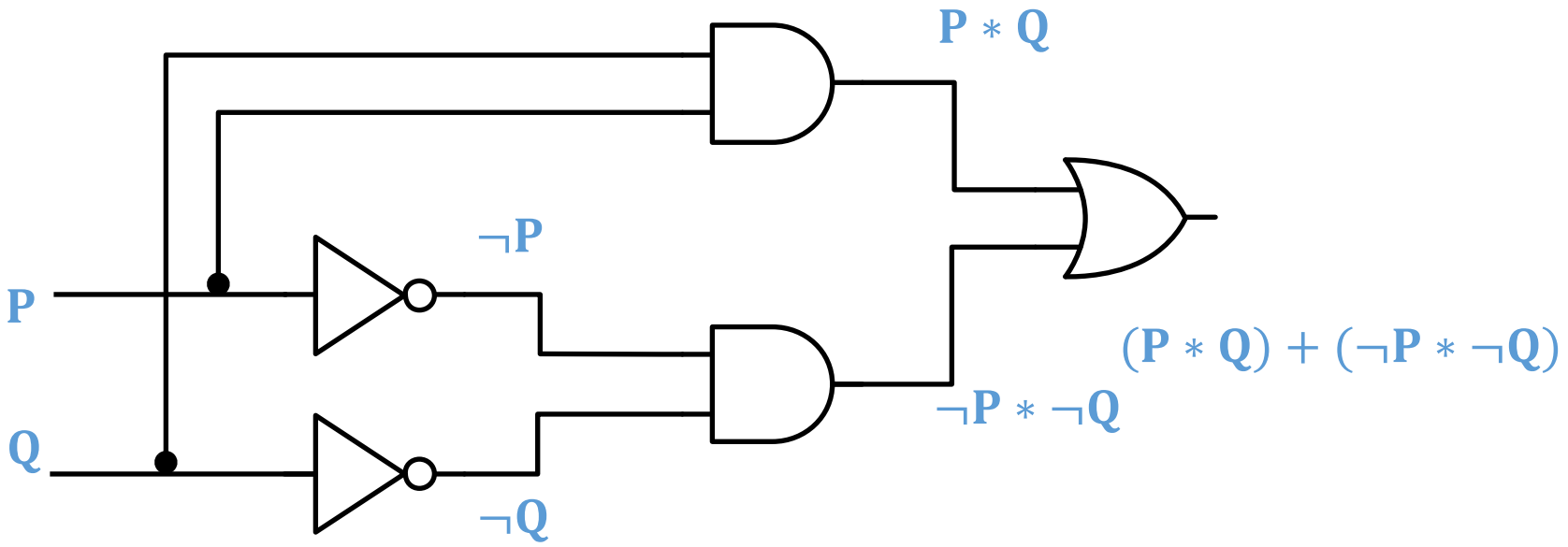
$P \oplus Q$	Q = 0	Q = 1
P = 0	0	1
P = 1	1	0

P	Q	¬P	¬Q	¬P * Q	P * ¬Q	(¬P * Q) + (P * ¬Q)
1	1	0	0	0	0	0
1	0	0	1	0	1	1
0	1	1	0	1	0	1
0	0	1	1	0	0	0

Truth Table For 1-bit Comparator

P	Q	$\neg P$	$\neg Q$	$P * Q$	$\neg P * \neg Q$	$(P * Q) + (\neg P * \neg Q)$
1	1	0	0	1	0	1
1	0	0	1	0	0	0
0	1	1	0	0	0	0
0	0	1	1	0	1	1

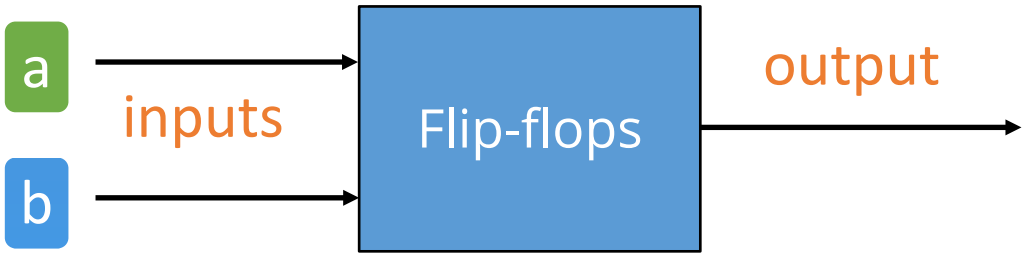
Circuit For 1-bit Comparator



Flip-flops (触发器)

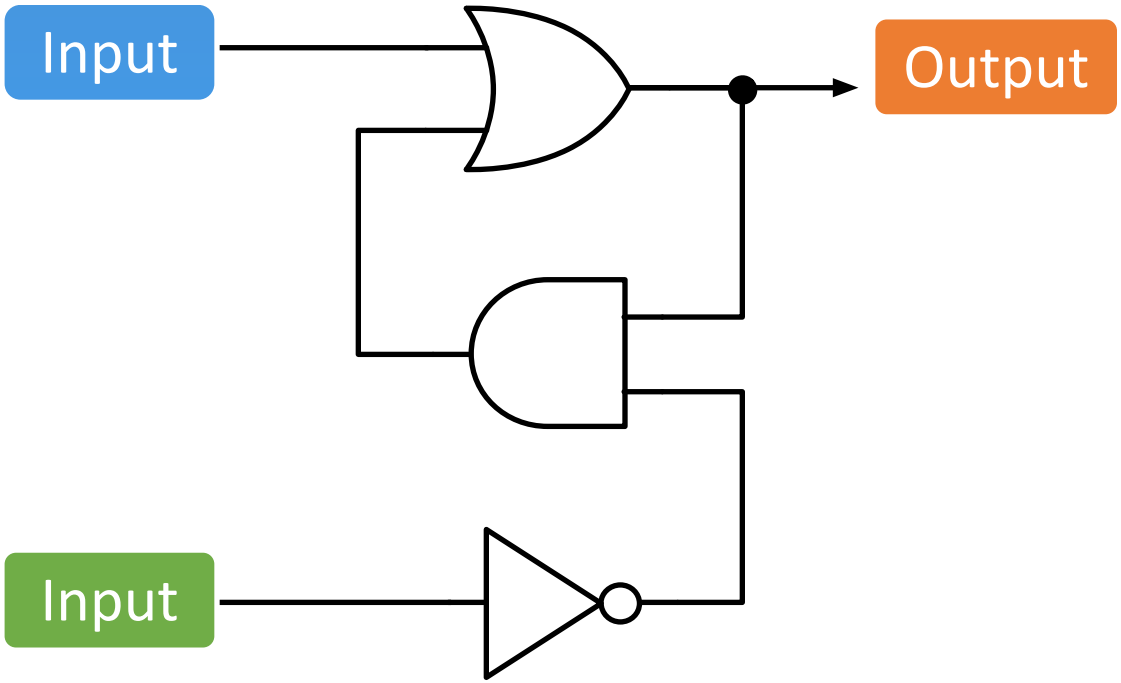
Flip-flop: A circuit built from gates that can store one bit (binary digit) of data.

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



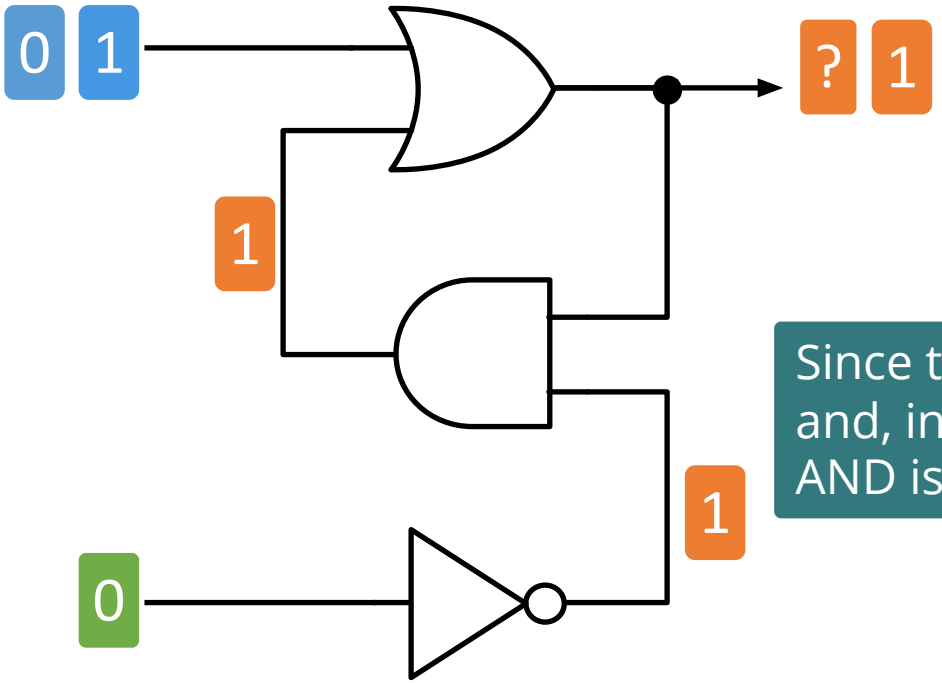
A black box model of Flip-flops

A Simple Flip-flop Circuit



Setting the Output of a Flip-flop to 1

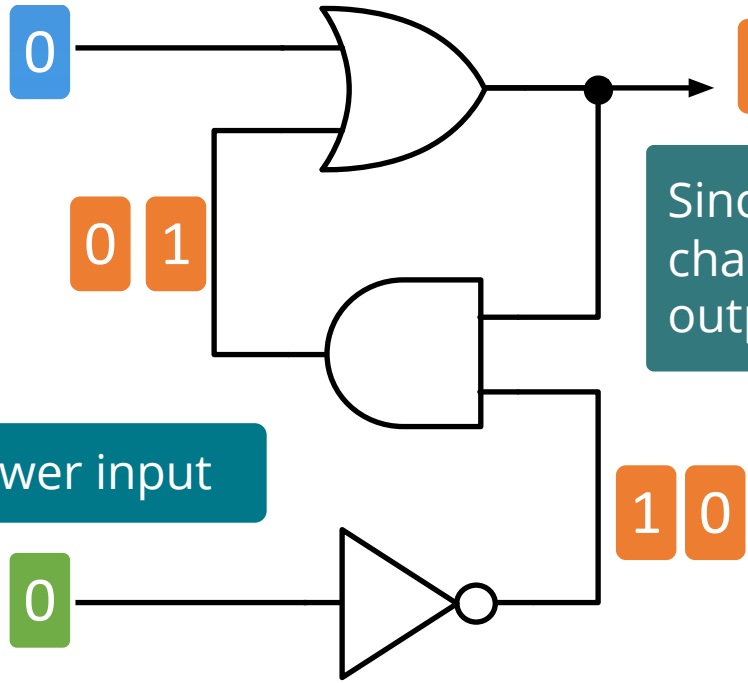
Place a 1 on the upper input



Since the output of OR is 1 and, in turn, the output of AND is 1.

The 1 from AND keeps the OR from changing after the upper input returns to 0.

Setting the Output of a Flip-flop to 0

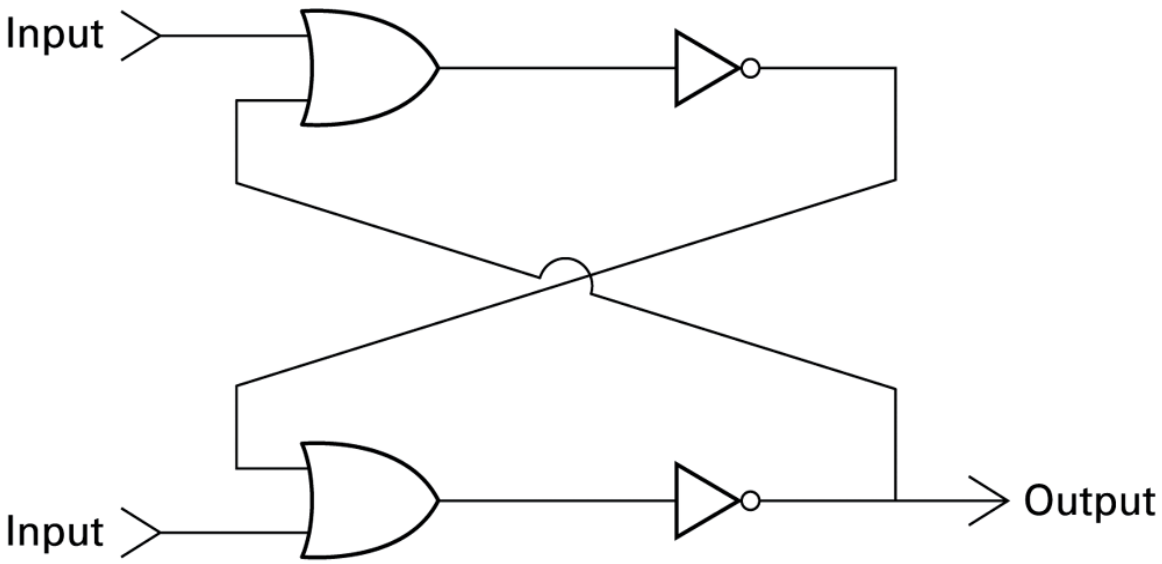


Since the output of AND changes to 0 and, in turn, the output of OR changes to 0.

Place a 1 on the lower input

The 0 from output keeps the AND from changing after the lower input returns to 0.

Another Way to Construct a Flip-flop



Here is another way to construct a flip-flop. Please explain why it can hold one bit of information by you self. I will ask someone to explain it to me in the next lecture.

2.
Representing Information as Bit Patterns

Representing Text

Each character (letter, punctuation, etc.) is assigned a unique bit pattern.

ASCII (American Standard Code, ASS-kee)

- Uses patterns of 7-bits to represent most symbols used in written English text
- Modern – uses 8-bit code, where last 128 characters are dependent on manufacturer

ISO standard

- Uses patterns of 32-bits to represent most symbols used in languages world wide – billions of characters

Unicode

- Both ASCII and ISO are originally designed for English, and thus are deficiency for international use.
- Uses patterns of 8/16-bits to represent the major symbols used in languages world wide
- UTF-8, UTF-16 - Unicode Transformation Format 8/16-bit

ASCII Table

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

Represent “Hello” in ASCII

“Hello”

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

Question:
What is the ASCII of “World”?

Question:
57 6F 72 6C 64
W o r l d

Representing Numeric Values

Binary notation

- Uses bits to represent a number in base two

Limitations of digital representations of numeric values

- Overflow – occurs when a value is too big to be represented
- Truncation – occurs when a value cannot be represented accurately

8 bits can represent 2^8=256 different integers

How to represent fraction 1/3=0.3… with 8 bits

Representing Other Types of Data

Representing images

- Bit map – representing an image as a collection of dots, each of which is called a pixel
- RGB, etc.

Representing sound

- Sample the amplitude of the sound wave at regular intervals and record it as time-series value
- MP3, MP4, etc.

3.

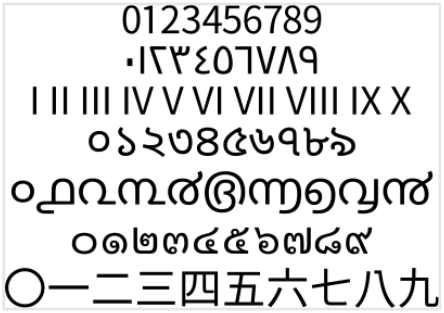
The Binary System

Numeral Systems (数字系统)

What types of numeral systems do you know?

Decimal system (十进制)

- Arabic numerals: 0,1,2,3,4,5,6,7,8,9
- The traditional decimal system is based on powers of ten



Binary system (二进制)

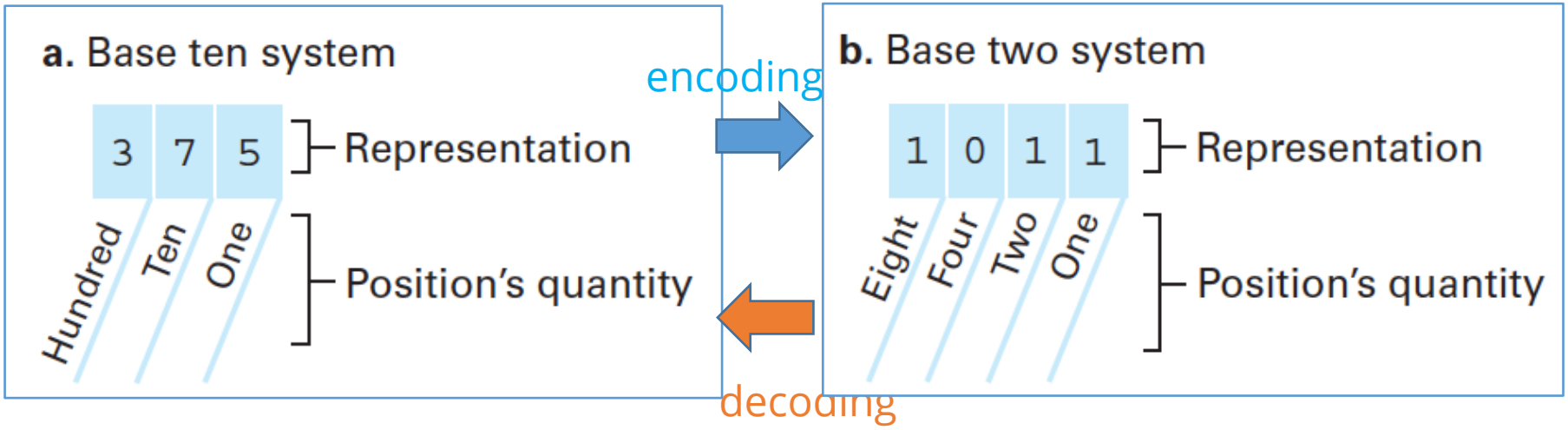
- Binary number: 0, 1
- Octonary: 0-7, 3 bits of binary
- Hexadecimal: 0-9 A-F, 4 bits of binary

Others

- 12 (dozens, month/year), 24(hours/day), 60 (minutes/hour, 甲子/circle), 100(century), etc.

$$(a_n a_{n-1} \cdots a_1 a_0 . c_1 c_2 c_3 \cdots)_b = \sum_{k=0}^{n} a_k b^k + \sum_{k=1}^{\infty} c_k b^{-k}$$

The Base Ten and Binary System



radix point

$$(a_n a_{n-1} \cdots a_1 a_0 . c_1 c_2 c_3 \cdots)_2 = \sum_{k=0}^{n} a_k 2^k + \sum_{k=1}^{\infty} c_k 2^{-k}$$

integer part fractional part

base 2 base 10

a_n	...	a_2	a_1	a_0	.	c_1	c_2	c_3	...	c_m
2^{n-1}		2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}		2^{-m}

The Base 10/2 Notations of Fractions

$$13 = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0$$

$$.8125 = 1 * 2^{-1} + 1 * 2^{-2} + 0 * 2^{-3} + 1 * 2^{-4}$$

radix point

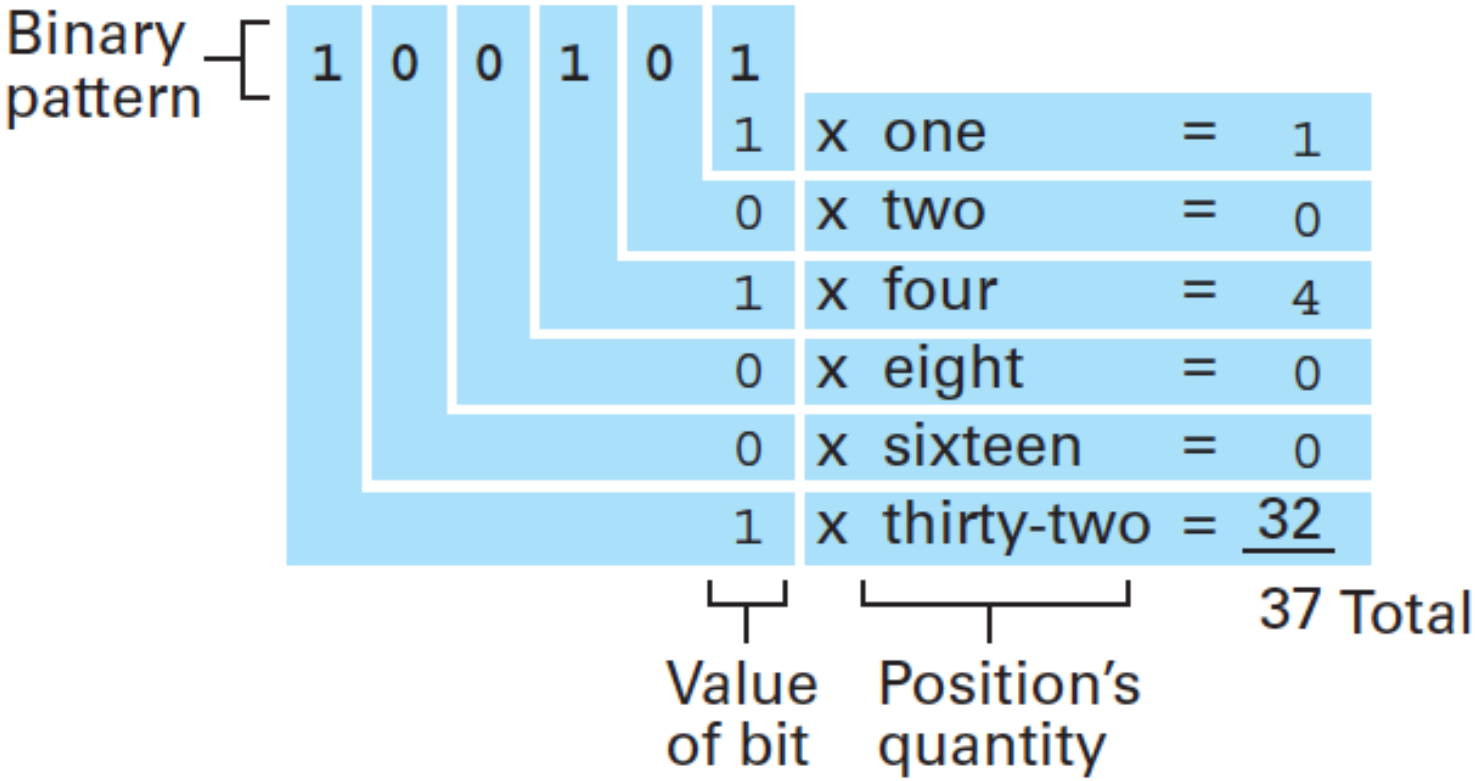
$$(a_n a_{n-1} \cdots a_1 a_0 . c_1 c_2 c_3 \cdots)_2 = \sum_{k=0}^{n} a_k 2^k + \sum_{k=1}^{\infty} c_k 2^{-k}$$

integer part fractional part

base 2 base 10

Converting Binary Representations to Decimal Number

Decoding the binary representation 100101

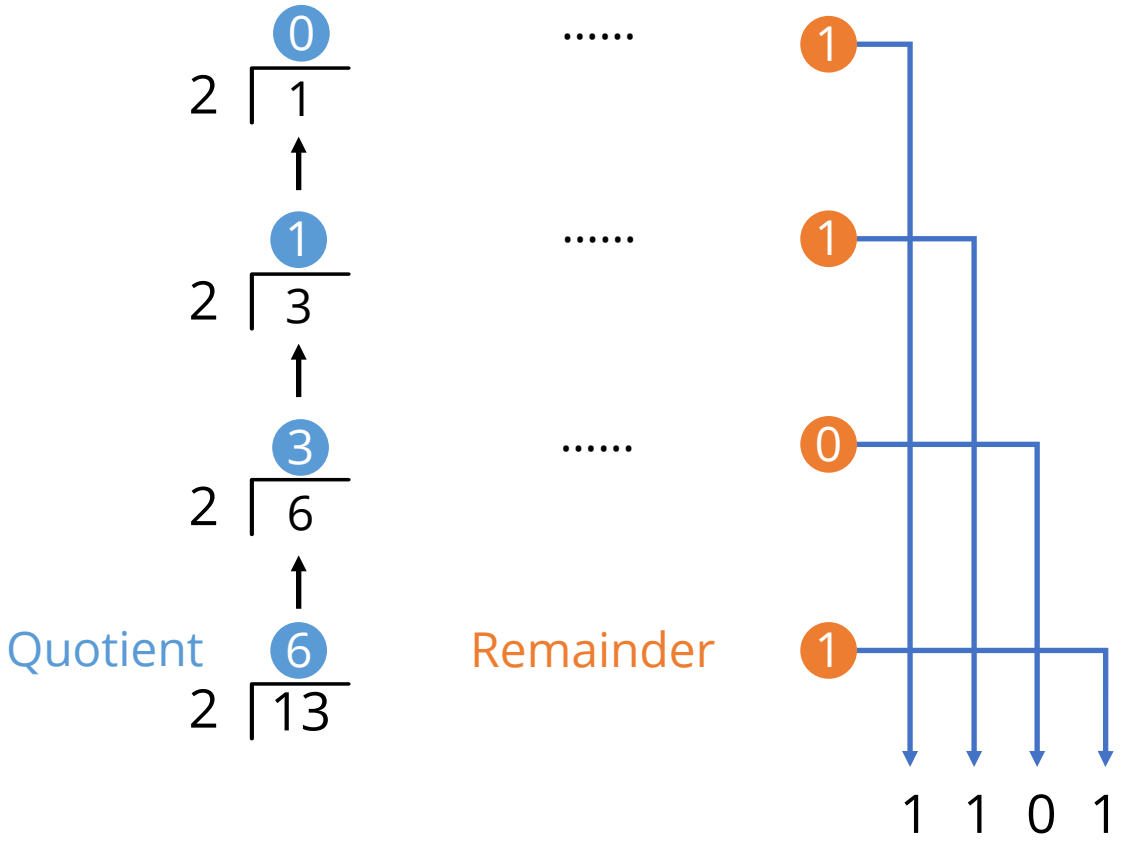


Algorithm 1: Encoding Positive Integers to Binary Representations

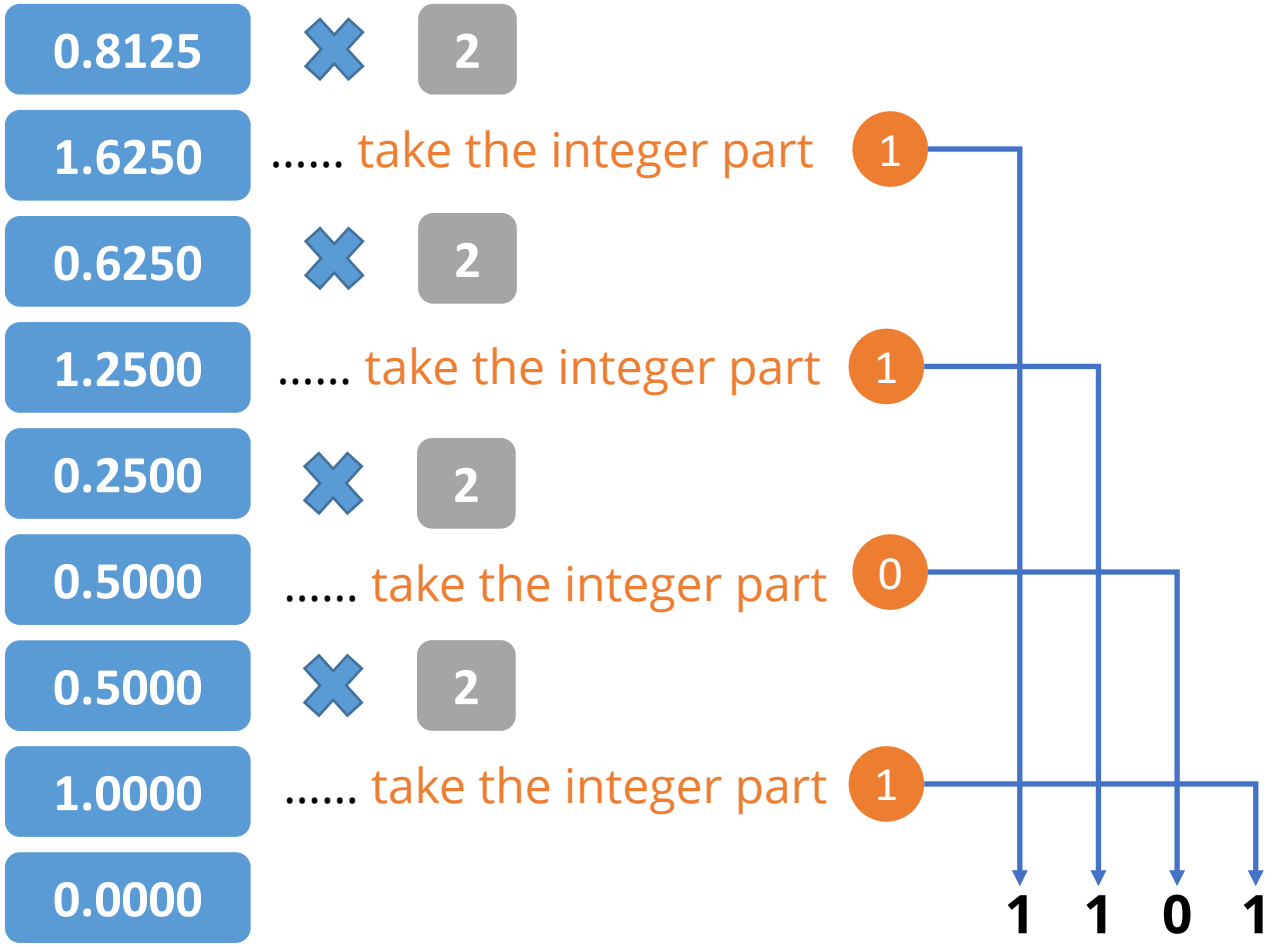
Algorithm procedures

- Step 1. Divide the value by 2 and record the remainder
- Step 2. If the quotient obtained is not zero, then take the quotient as the value and repeat Step 1, i.e. divide the newest quotient by 2 and record the remainder.
- Step3. 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

Encoding 13 in Binary



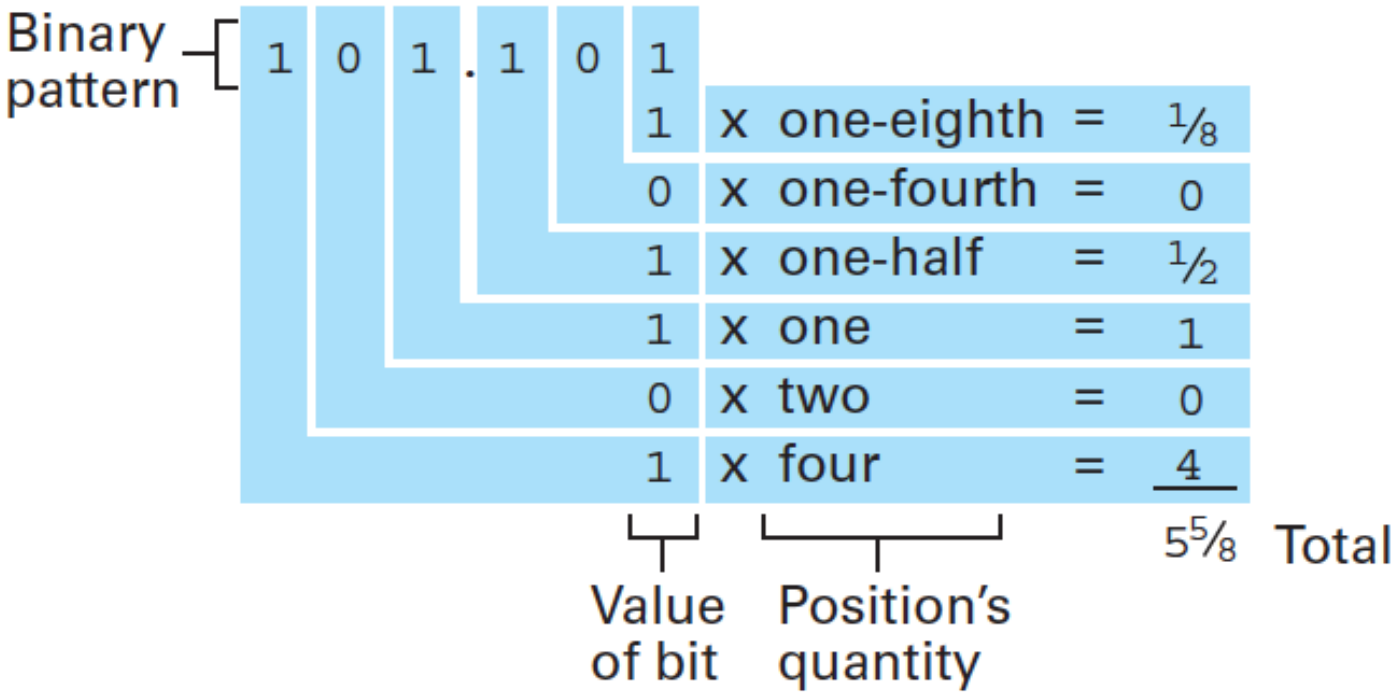
Encoding 0.8125 in Binary



The remainder is 0 and stop

Decoding the Binary Representation

Example: 101.101



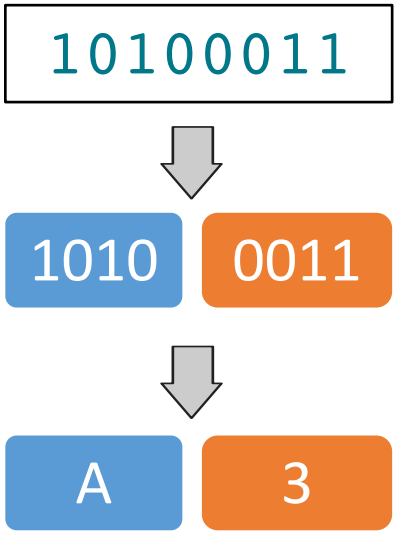
Hexadecimal Notation

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:

- encoding 10100011
- 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

To convert from binary to hex

- starting on the right of the binary number, arrange the binary digits in groups of four
- convert each quartet to the corresponding hex digit
- 110 1110 1100
- 6 E C

To convert from hex to binary

- replace each hex digit with its 4-bit binary equivalent
- 8 A 5
- 1000 1010 0101

In-class Exercises

Convert the following decimal numbers to binary representation

- 9, 11, 17
- 2^{16}, 2^{32} -1
- 1/16, 1/128, 5/8, $1/2^{16}$

Convert the following binary representations to decimal values

- 101010, 111111, 0.011, 101.111
- 100…0 (with 10 zeros), 100…0 (with 20 zeros), 100…0 (with 30 zeros),
- 11…1(with 30 ones)

Summary of Binary System

By far, we learned how to represent unsigned/positive integers/fractions as binary numbers

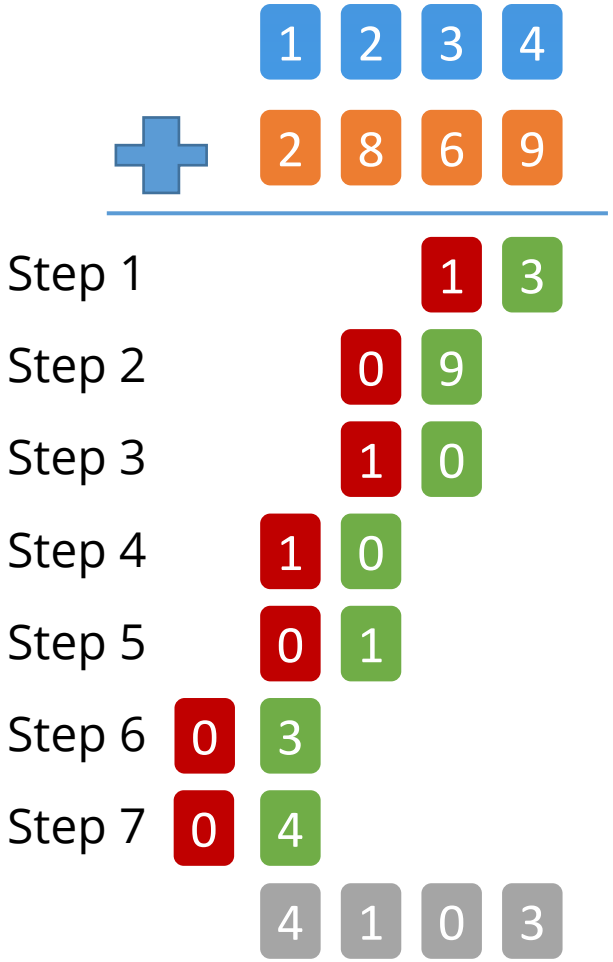
- We can store two different values in 1 bit: 0 or 1.
- In general, in n bits, you can store 2^n different values.
- So with 4 bits, we can store 16 values - 0 to 15, but there are no negative values.

How to represent negative values?

- In general, we can split the number of values in half, making half positive, half negative, and zero.
- Two's complement notation (补码)

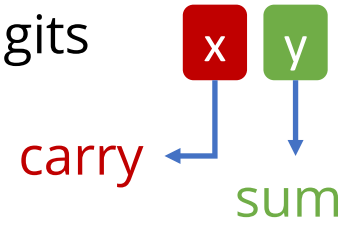
Arithmetic – Binary Addition

Decimal Addition



$a, b = 0 \cdots 9 \quad a + b \in [0,18]$

Two digits



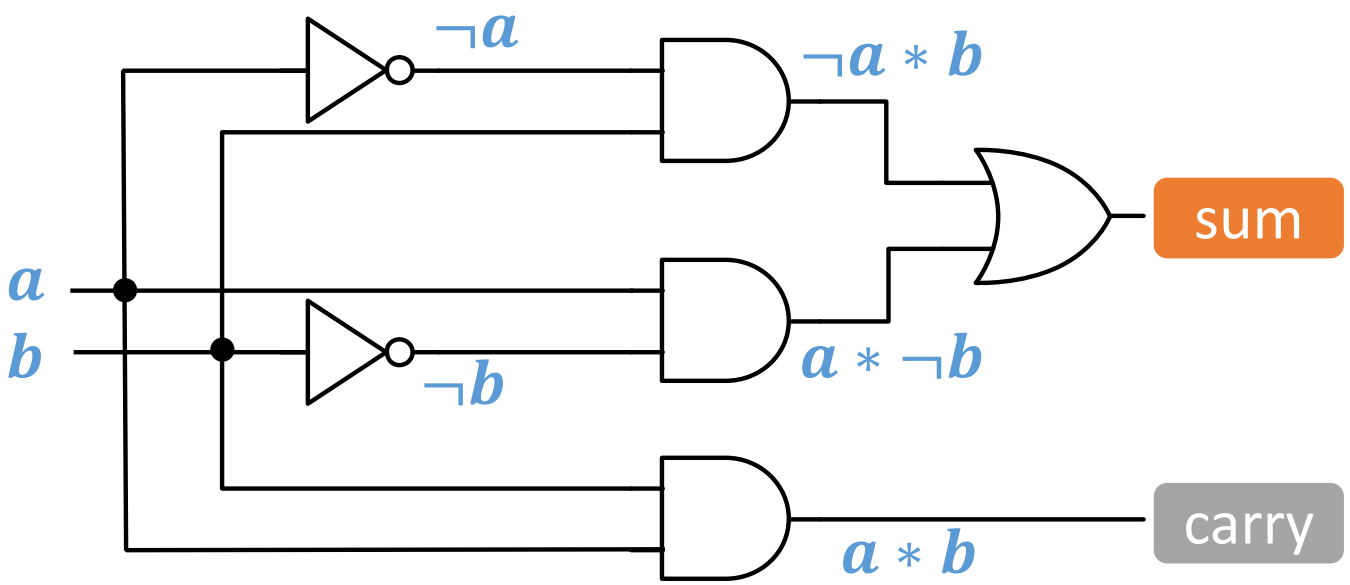
Binary Addition

- The sum digit is 1 only when both inputs are different
- The carry digit is 1 only when both inputs are 1
- so carry bit is $carry = a * b$
- build a logic table for the sum

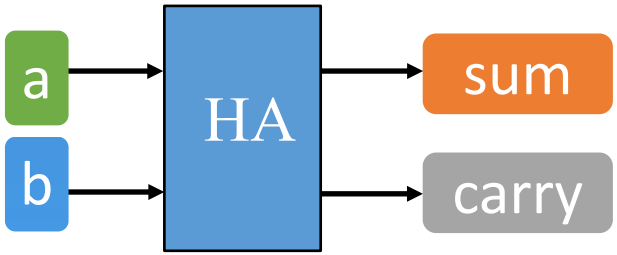
Binary Addition

a	b	$\neg b$	$a * \neg b$	$\neg a$	$\neg a * b$	$(a\neg b) + (a\neg b)$
1	1	0	0	0	0	0
1	0	1	1	0	0	1
0	1	0	0	1	1	1
0	0	1	0	1	0	0

Addition – Half Adder



Logical circuit of half adder (HA)

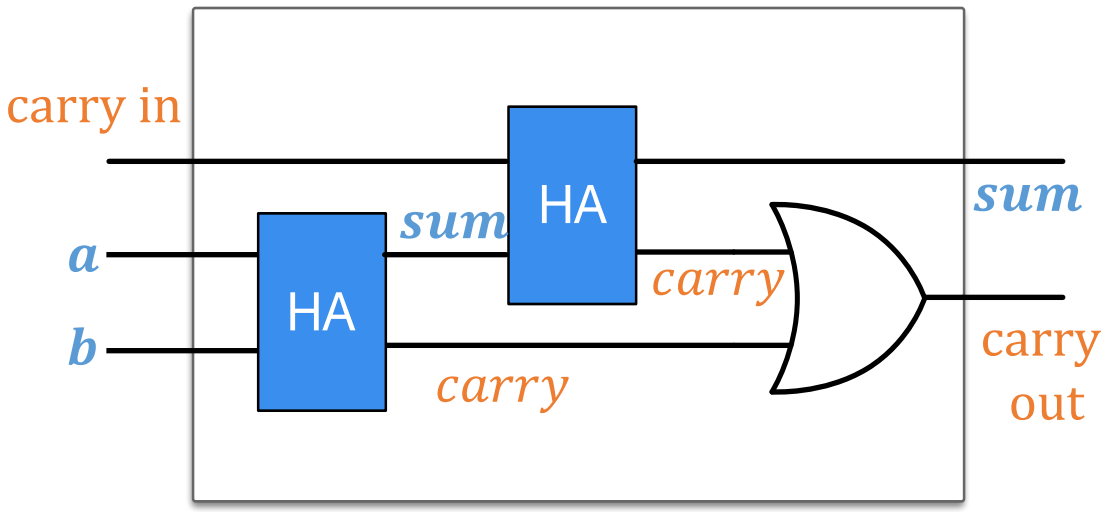


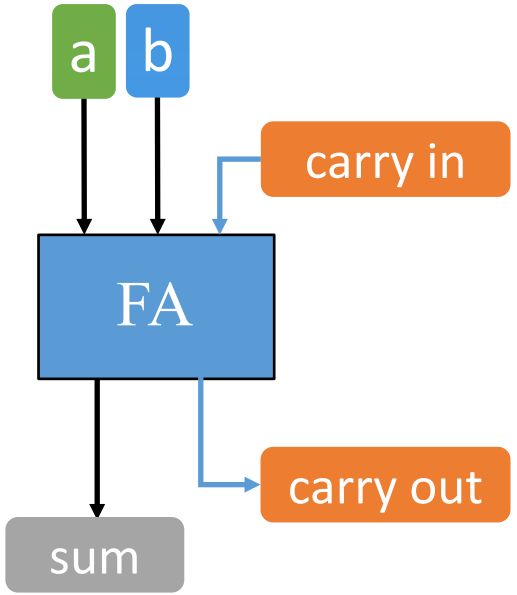
A black box model of HA

Addition – Full Adder

Full adder (HA)

- Half adder adds two numbers giving sum and carry result, but no provision for a carry in.
- So we hook two half adders together to create a 1-bit full-adder

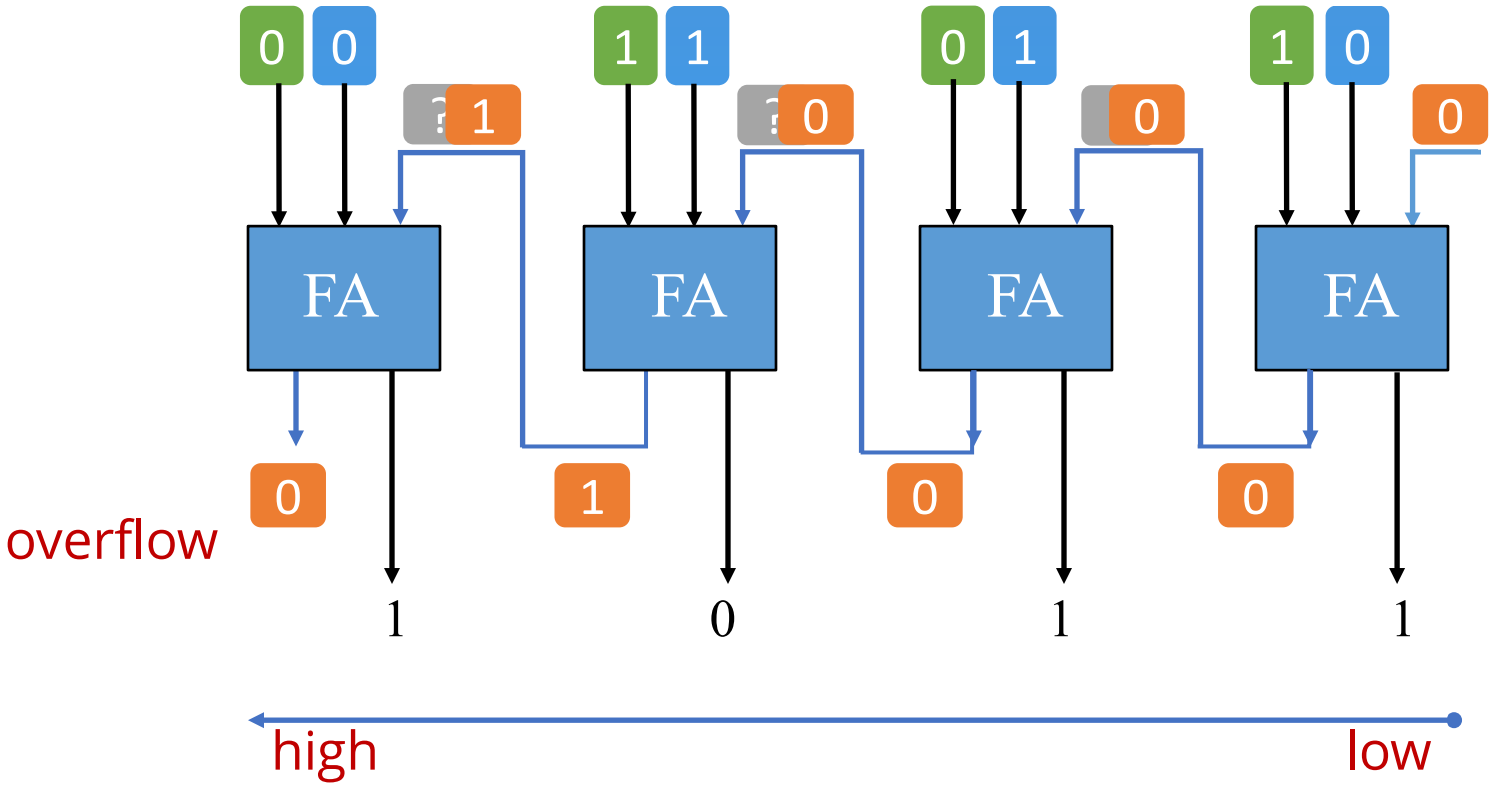




A black box model of HA

Adding Two 4-bit Numbers

Let's add 0101 and 0110



4.
Storing Integers & Fractions

Storing Integers

Two's complement notation (补码)

- The most popular means of representing integer values
- The standard using pattern is with 32 bits, 8 bits in the earlier days

Excess notation

- Another means of representing integer values – used in floating point notation
- Both can suffer from overflow errors.

Two's Complement Notation Systems

a. Using patterns of length three

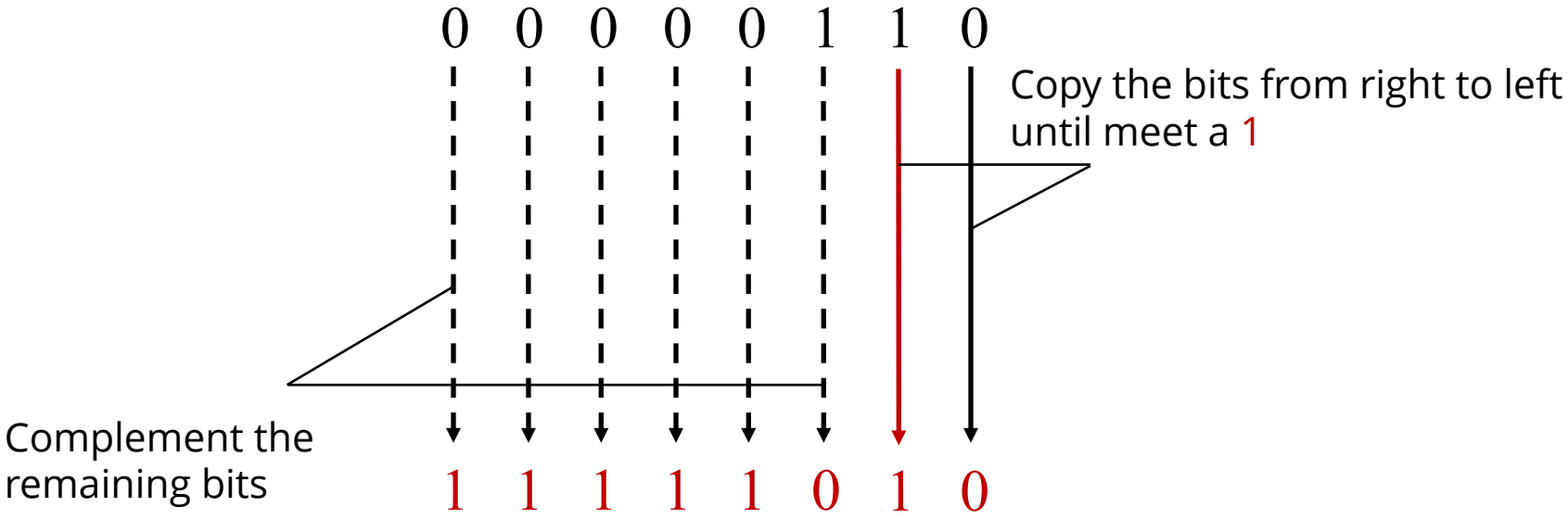
Bit pattern	Value represented
011	3
010	2
001	1
000	0
111	−1
110	−2
101	−3
100	−4

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

Two's Complement Notation Using 8 Bits

Two's complement notation for 6



Two's complement notation for -6

Another method to compute two's complement of 6

- One's complement (反码): flip all the bits, 11111001
- add 1 to the result: 11111001+00000001 = 11111010 = -6

How Many Values with the Following Bit Patterns ?

With 4 bits

- min = $-8 = -2^{4-1}$ and max = $7 = 2^{4-1} - 1$

With 16 bits

- min = $-32{,}768 = -2^{16-1}$ and max = $32{,}767 = 2^{16-1} - 1$

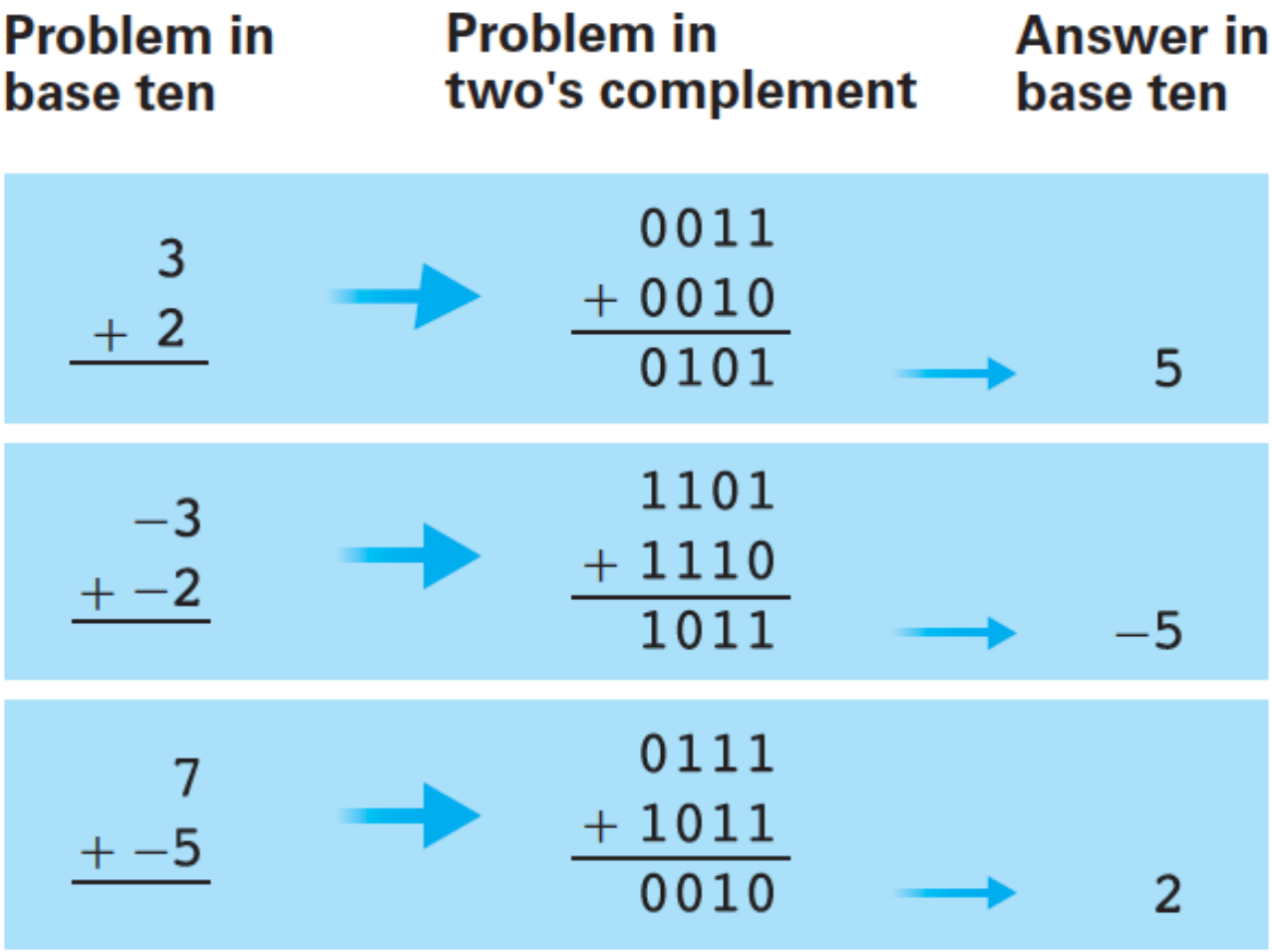
With 16 bits

- min = $-32{,}768 = -2^{16-1}$ and max = $32{,}767 = 2^{16-1} - 1$

With 32 bits

- min = $-2{,}147{,}483{,}648 = -2^{32-1}$ and max = $2{,}147{,}483{,}647 = 2^{32-1} - 1$

Addition Problems Converted to Two's Complement Notation



Overflow – Trouble in Paradise

Using 6 bits we can represent values from -32 to 31, so what happens when we try to add 19 plus 14 or -19 and -14.

19 +14 33	010011 +001110 100001	we have added two positive numbers and gotten a negative result – this is overflow
-19 -14 -33	101101 +110010 011111	we have added two negative numbers and gotten a positive result – this is overflow

Excess Notation System

Four-bits pattern

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

Three-bits Pattern

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

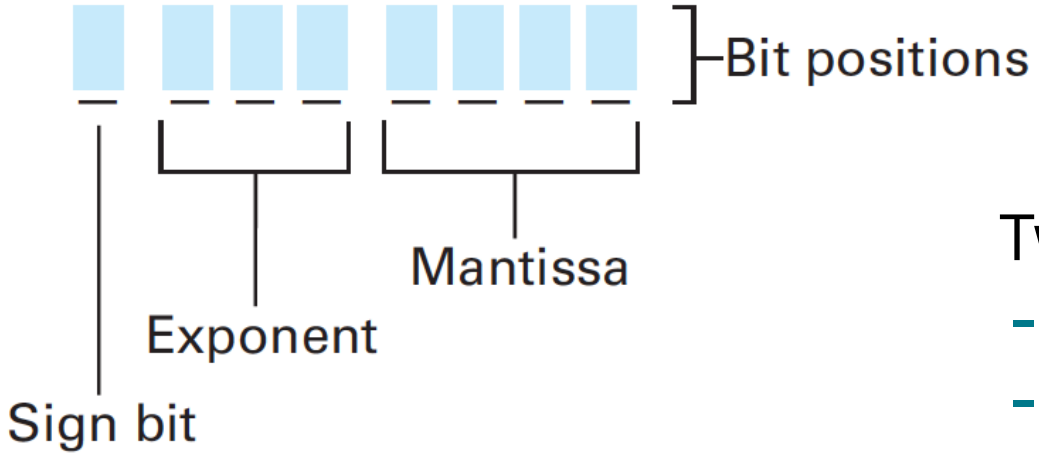
N-bits pattern:

excess notation = binary code – 2^{N-1}

Storing Fractions – Floating-Point

Floating-point notation (FP)

- Consists of a sign bit (符号位), a mantissa field (小数位), and an exponent field (指数位)
- Suppose we use 8 bits to store FP



Two related topics

- Normalized form
- Truncation errors

Algorithm 2: Converting Fractions to FP

Algorithm procedures

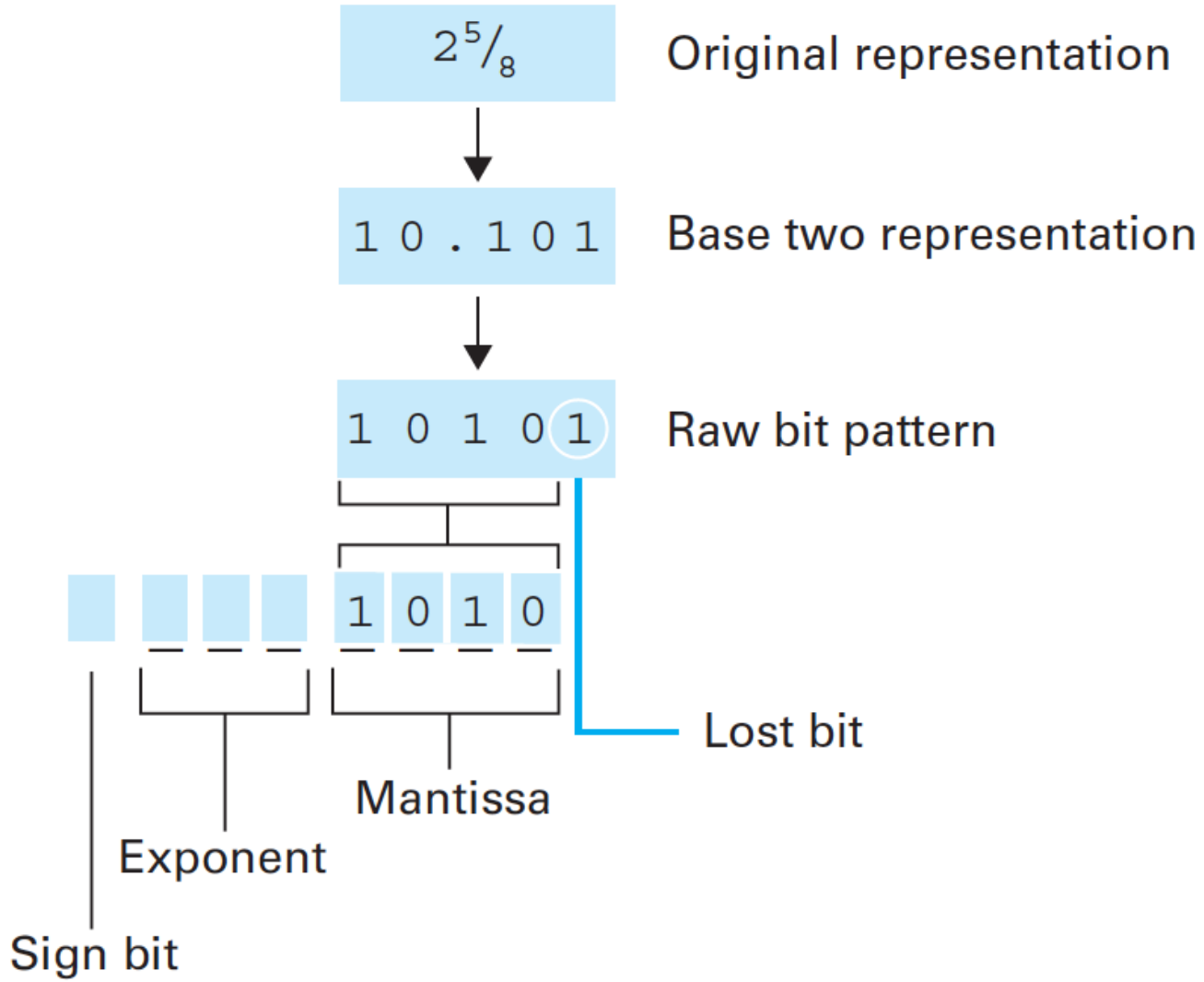
- Represent the fraction x in binary
  - such as 2¼ → 10.01
- Normalize the number (move the binary point to the left of the most significant 1 – leftmost one) and adjust the exponent, similar to scientific notation
  - $10.01 * 2^0 = .1001 * 2^2$, so 2 is the exponent value
- Calculate the exponent by adding the excess value to the exponent value:
  - 2 + 4 = 6 = 110 in binary
- Figure out the sign – positive is 0
- Put all together
  - 0 110 1001

Example 1: Represent Fractions in FP

Representing -3/8 in FP

- Remember if the number has a whole portion, the exponent will be positive. If the number is 0 or a fraction, the exponent will be 0 or negative
- -3/8 = .011 in binary
- normalize $.11 * 2^{-1}$ (pad fraction = .1100)
- calculate exponent: -1 + 4 = 3 = 011
- calculate sign: 1 for negative
- put it together: 1 011 1100

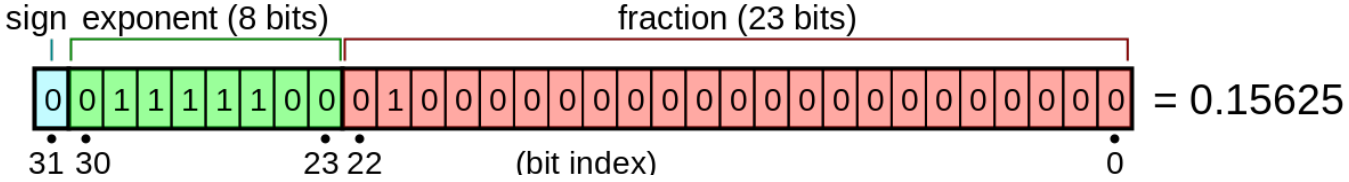
Example 2: Truncation Error



Representing floating point

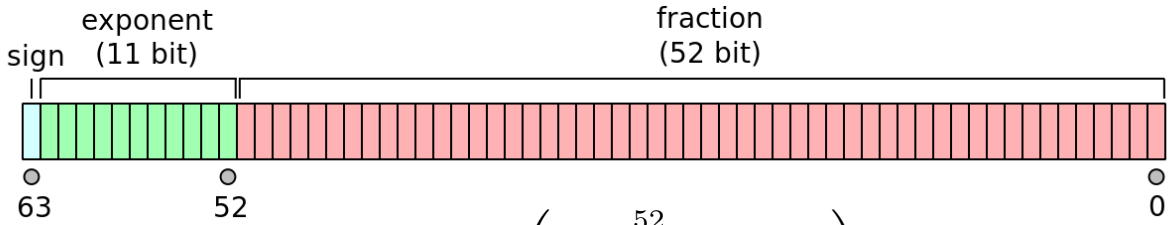
Most common forms are binary32 (single precision) and binary64 (double precision)

Binary32 (single-precision FP, en.wikipedia.org/wiki/Binary32)



$$\text{value} = (-1)^{\text{sign}} \times \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i}\right) \times 2^{(e-127)}$$

Binary64(double-precision FP, en.wikipedia.org/wiki/Binary64)



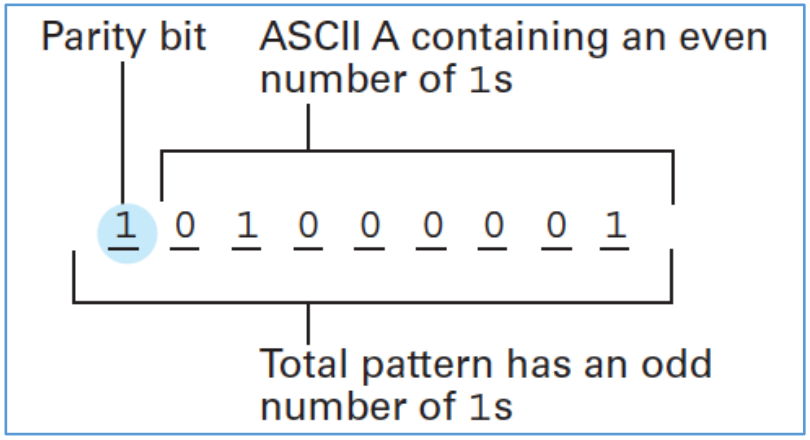
$$\text{value} = (-1)^{\text{sign}} \left(1 + \sum_{i=1}^{52} b_{52-i} 2^{-i}\right) \times 2^{e-1023}$$

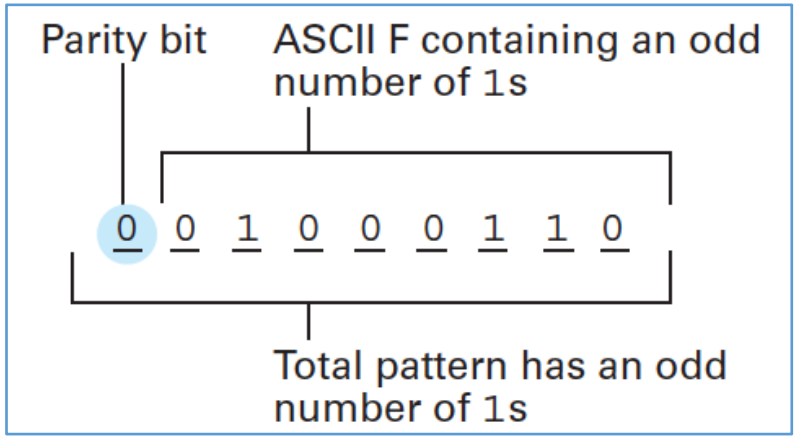
Error Detection

Parity bits (校验位)

- add an extra bit to each memory cell
- for odd parity
  - count the number of 1s in memory cell
  - set extra parity bit to 0 if value already contains an odd number of 1s
  - set parity bit to 1 if number of 1 bits is even
  - so memory cell + parity bit will always have an odd number of 1s

The ASCII codes for the letters A and F adjusted for odd parity





5.
Main Memory & Mass Storage

Storage Devices

The commonly used storage devices

- Main memory: the bit at the left (high-order) end of the conceptual row of bits in a memory cell

Massive storages

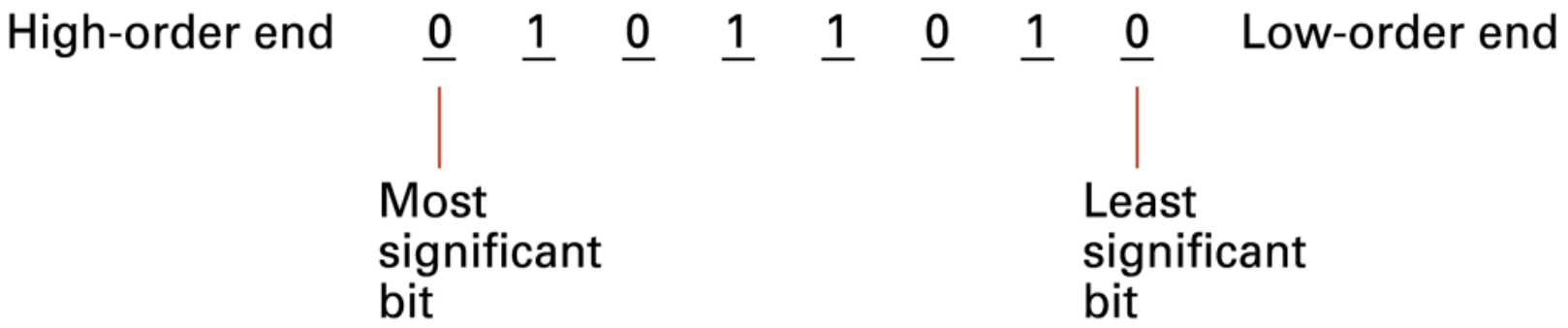
- Magnetic Systems
  - Disk
  - Tape
- Optical Systems
  - CD
  - DVD
- Flash Drives

Main Memory Cells

Cell: A unit of main memory

- typically 8 bits form 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

The organization of a byte-size memory cell

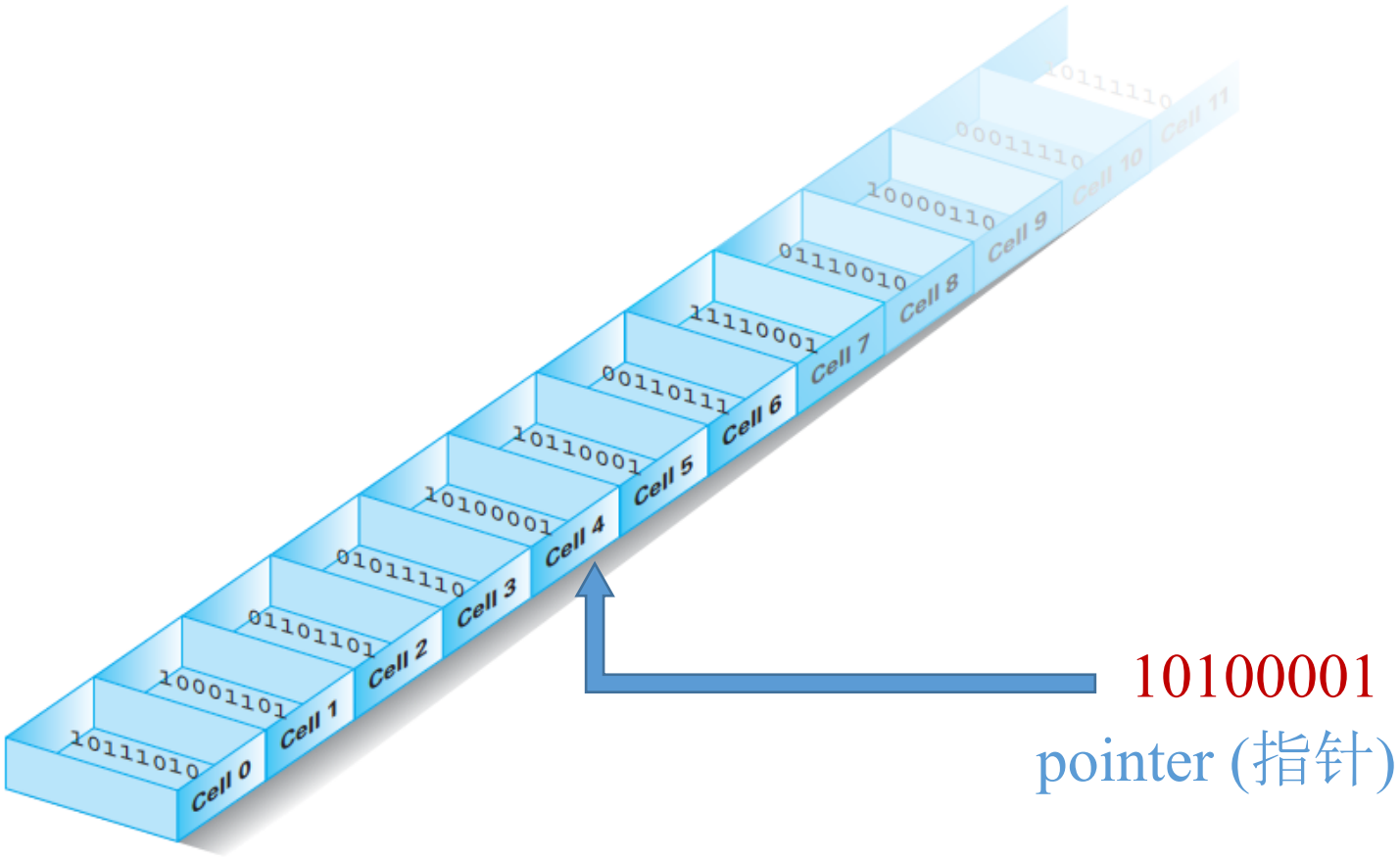


Main Memory Addresses

Address: A "name" that uniquely identifies one cell in the computer's main memory

- The names are actually numbers.
- These numbers are assigned consecutively starting at zero.
- Numbering the cells in this manner associates an order with the memory cells.

Memory Cells Arranged by Address



Memory 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

Kilobyte: 2^{10} bytes = 1024 bytes

- Example: 3 KB = 3 times 1024 bytes
- “kibi” in short

Megabyte: 2^{20} bytes = 1,048,576 bytes

- Example: 3 MB = 3 times 1,048,576 bytes
- “megi” in short

Gigabyte: 2^{30} bytes = 1,073,741,824 bytes

- Example: 3 GB = 3 times 1,073,741,824 bytes
- “gigi” in short

Terabyte: 2^{40} bytes

Petabyte: 2^{50} bytes

Exabyte: 2^{60} bytes

Mass Storage

Why mass storage?

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

Magnetic Systems

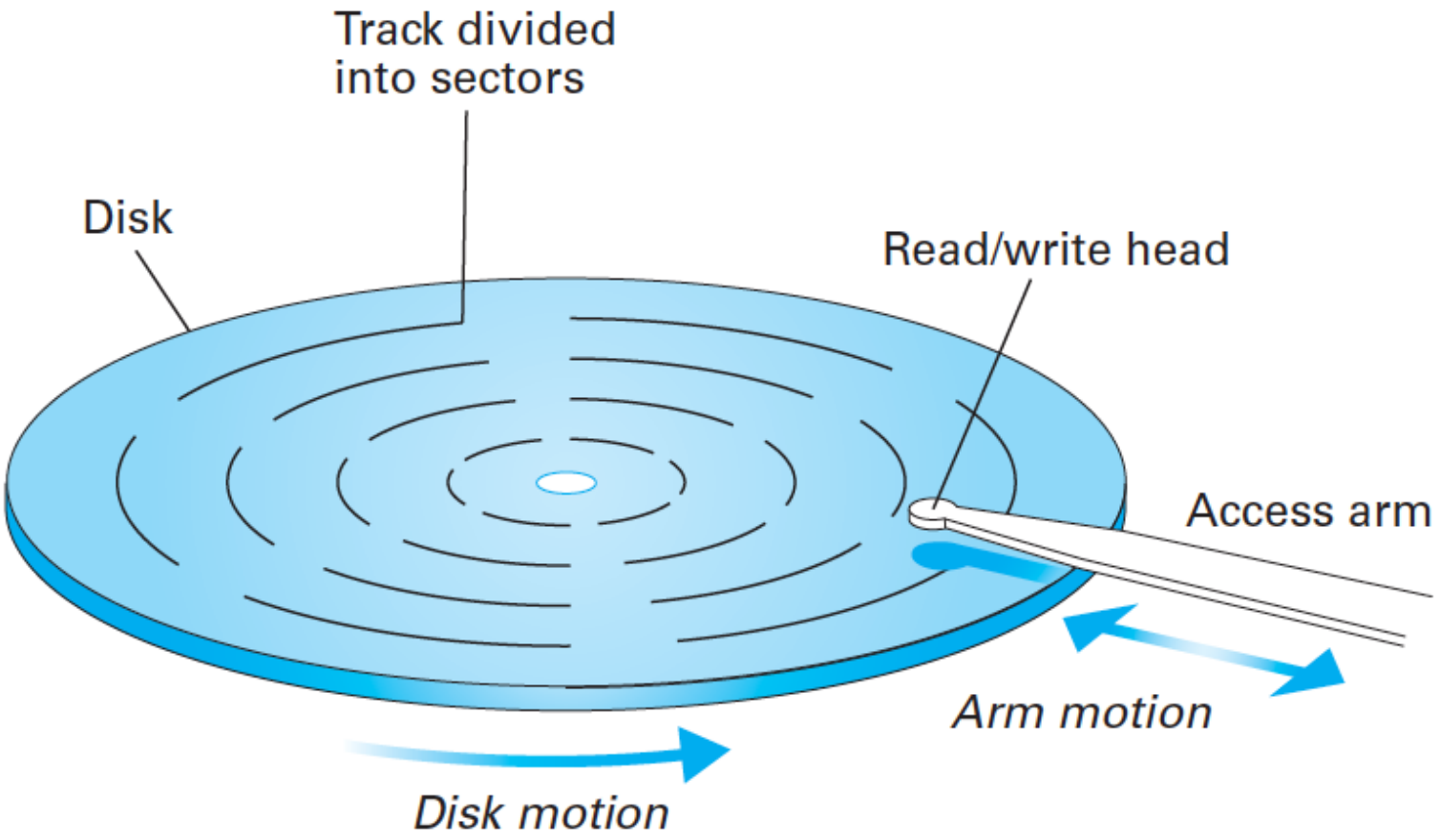
- Disk
- Tape

Optical Systems

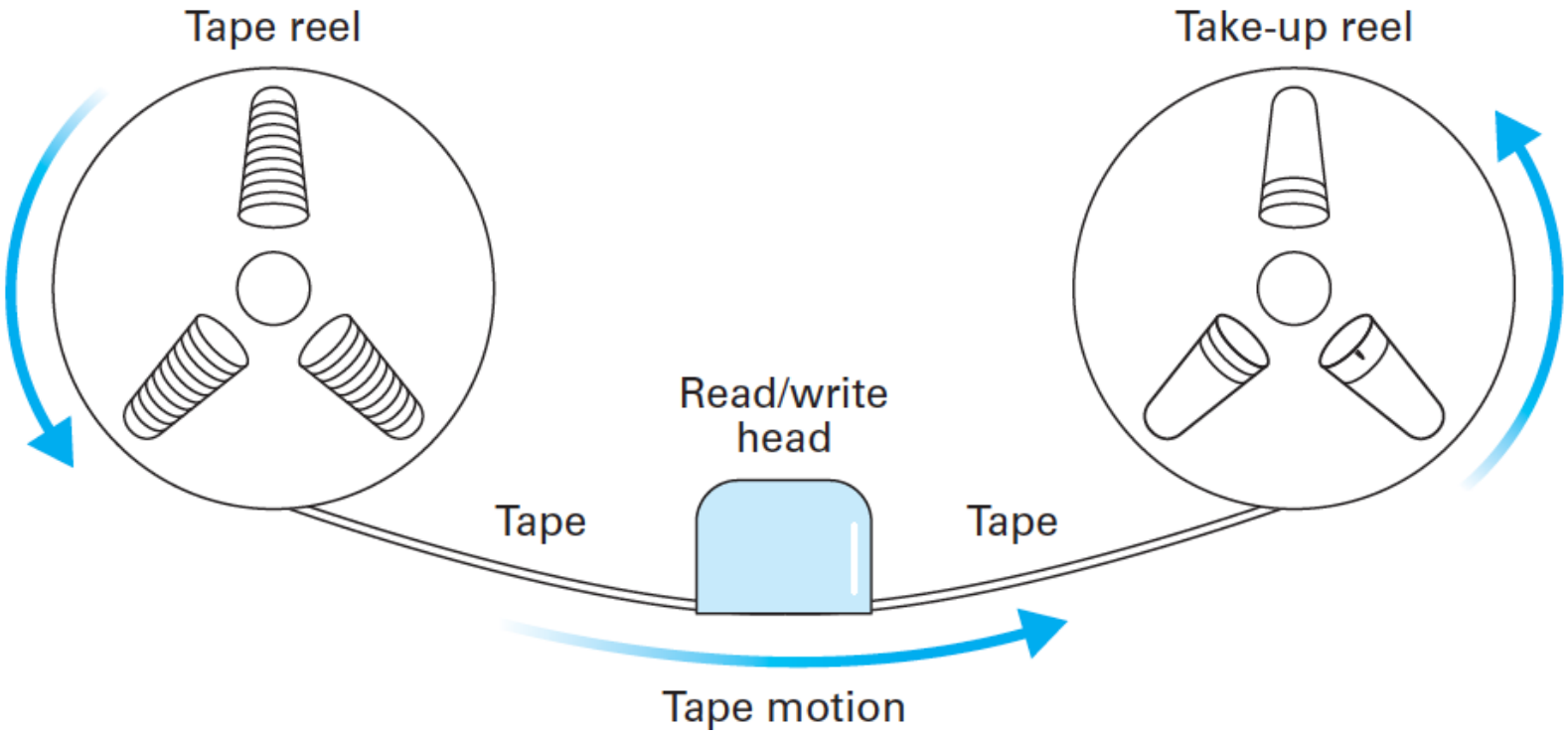
- CD
- DVD

Flash Drives

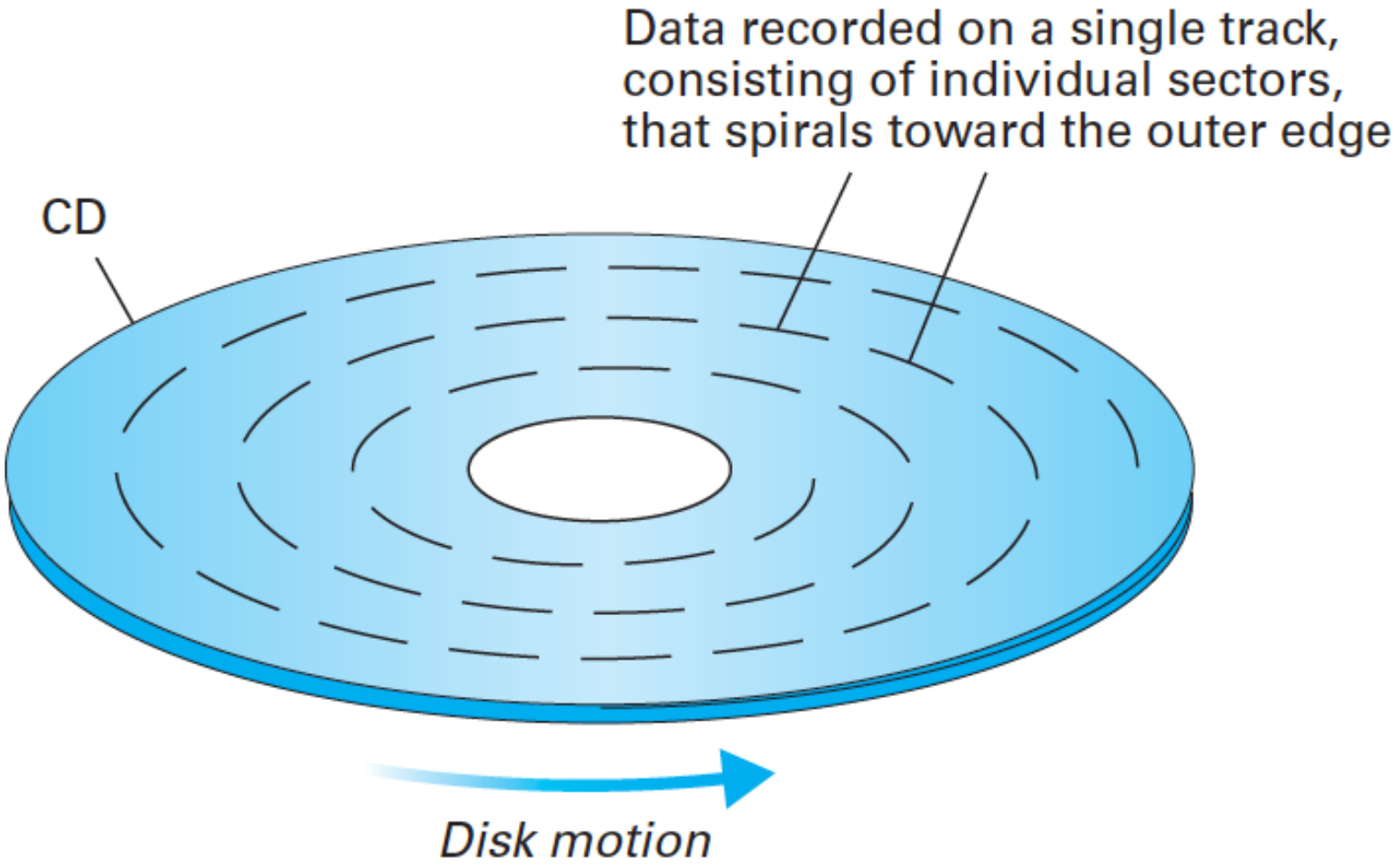
A Magnetic Disk Storage System



Magnetic Tape Storage



Compact Disc CD storage



CDs and DVDs

- Reflective material covered with clear protective coating.
- Information is recorded by creating variations in this reflective surface
- High powered laser beams to created pits
- Low powered laser beam to retrieve data
- Smooth unpitted area is a 1, pitted area is interpreted as a 0

Flash Drives

- Magneticc & optical devices require physical motion to store and retrieve data
- slow
- In flash memory, bits are stored by sending electronic signals directly to the storage medium where they cause electrons to be trapped in tiny chambers of silicon dioxide, thus altering the characteristics of small electronic circuits
- Good for off-line storage, digital cameras, phones, PDAs

Files

File: A unit of data stored in mass storage system

- Fields and keyfields
- Operating System (OS)

Physical record versus Logical record

Buffer: A memory area used for the temporary storage of data (usually as a step in transferring the data)

In-class Exercise

Convert the following decimal numbers to binary representation

- 9, 11, 17
- 2^{16}, 2^{32} -1
- 1/16, 1/128, 5/8, $1/2^{16}$

Convert the following binary representations to decimal values

- 101010, 111111, 0.011, 101.111
- 100…0 (with 10 zeros), 100…0 (with 20 zeros), 100…0 (with 30 zeros),
- 11…1(with 30 ones)

THANKS