

Digital Logic Circuit (SE273 – Fall 2020)

Lecture 2: Number Systems

Jaesok Yu, Ph.D. (jaesok.yu@dgist.ac.kr)

Assistant Professor
Department of Robotics Engineering, DGIST

▶ Property of Digital Computers

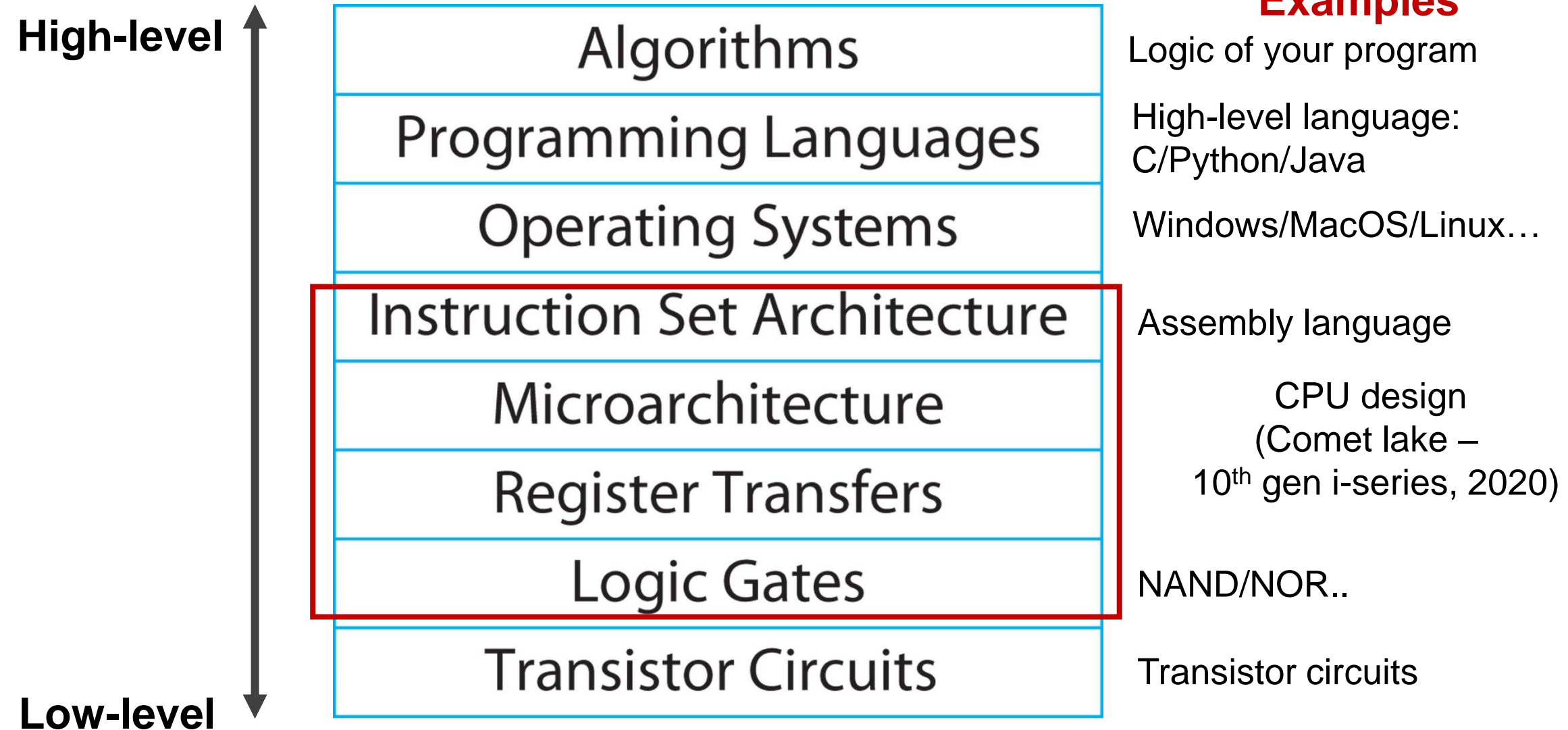
- Generality
 - Follow a sequence of instructions, called a program, that operates on given data
 - User specifies the program and data depending on the application
 - This provides “**flexibility**”
- What is the characteristics of digital systems/computers?

► What is a Digital System?

- A system that manipulates discrete elements of information
 - 10 decimal digits
 - 26 letters of the alphabet
 - 52 playing cards
- These discrete information are represented by physical “signals”
 - Electrical signals: voltages or currents
 - When dealing with discrete signals → digital system

► Typical layers of abstraction in Modern Computer

Examples

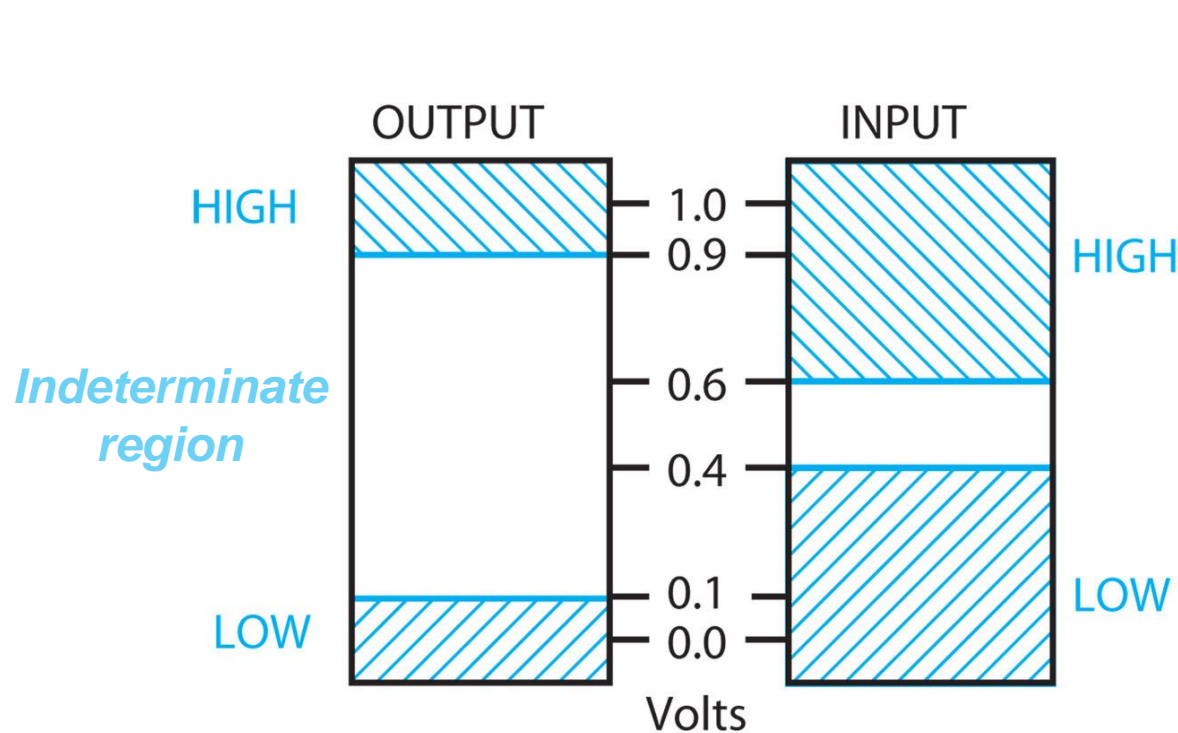


► Binary System

- A digital system that runs with binary instructions and data
- Ex) voltage ranges for binary signals
 - Wider input range to handle undesirable noise added to the output signal
 - Signal names: HIGH-LOW, TRUE-FALSE, 1-0
- Why binary?
 - To design a reliable system
 - Suppose 10 discrete values to represent..

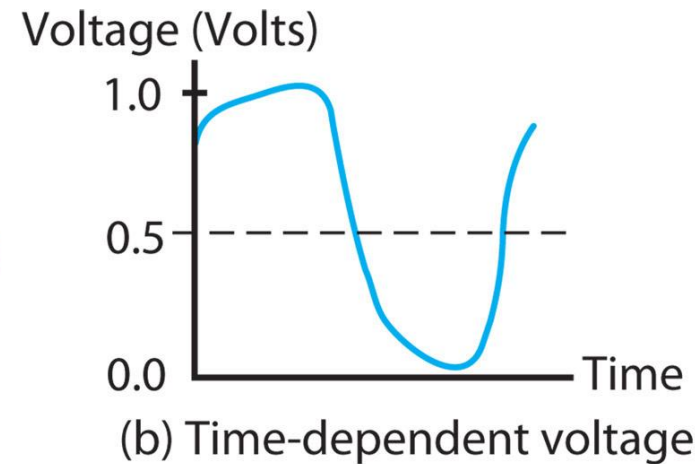
► Binary System

- Voltage ranges and waveforms for Binary signals

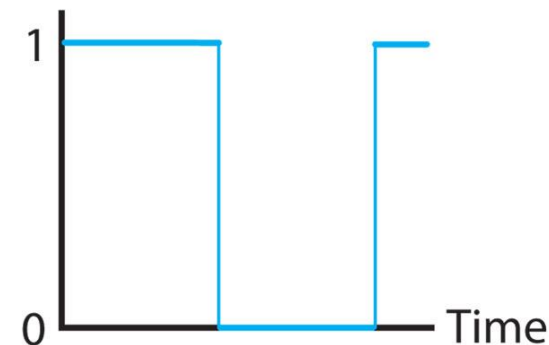


Why difference?

(a) Example voltage ranges



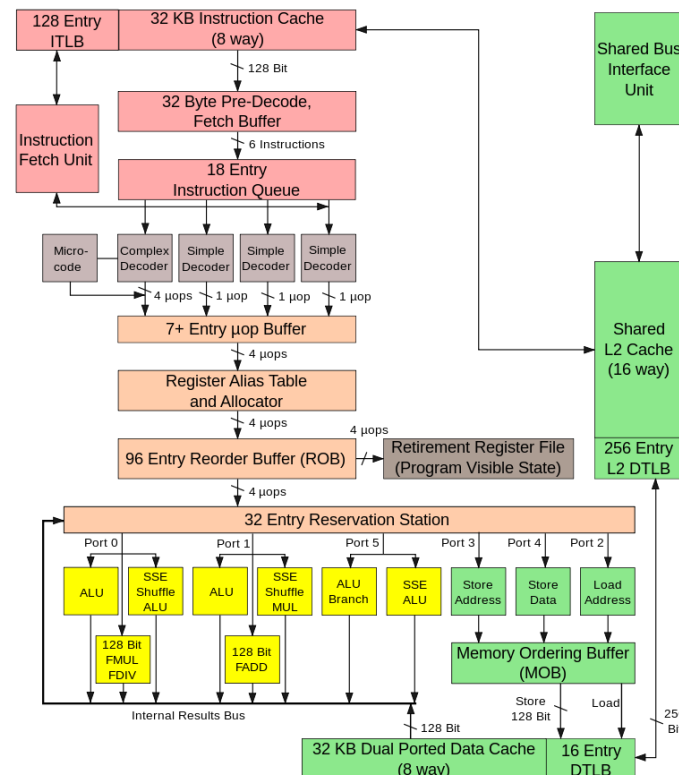
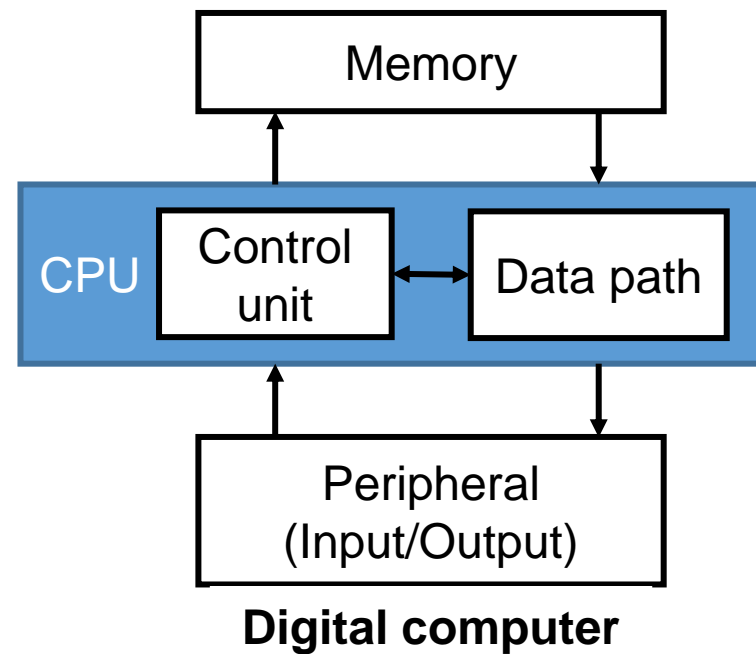
(b) Time-dependent voltage



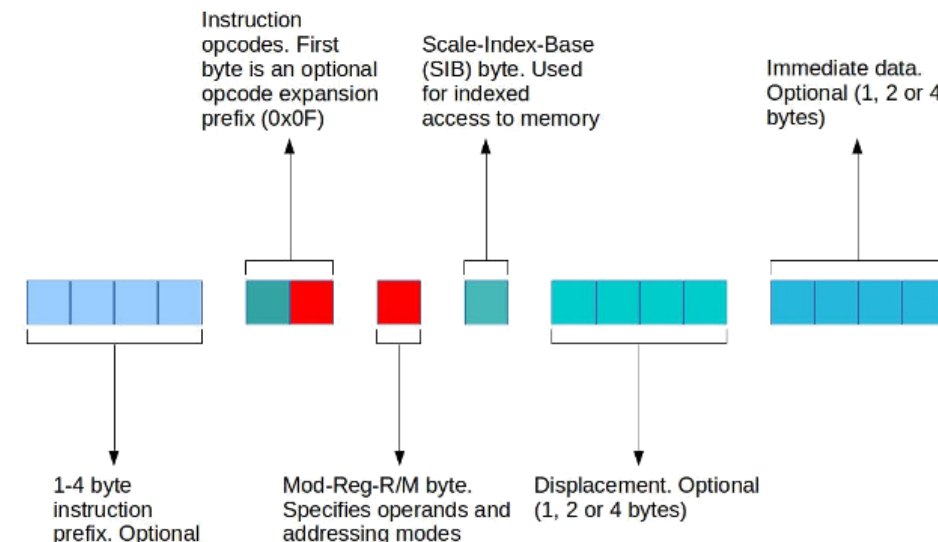
(c) Binary model of time-dependent voltage

► Information Representation

- A binary digit is called a “**bit**”
- Information is represented by a group of bits (ex: 32 bit or 64 bit)
 - Even a set of instructions in computer system is specified

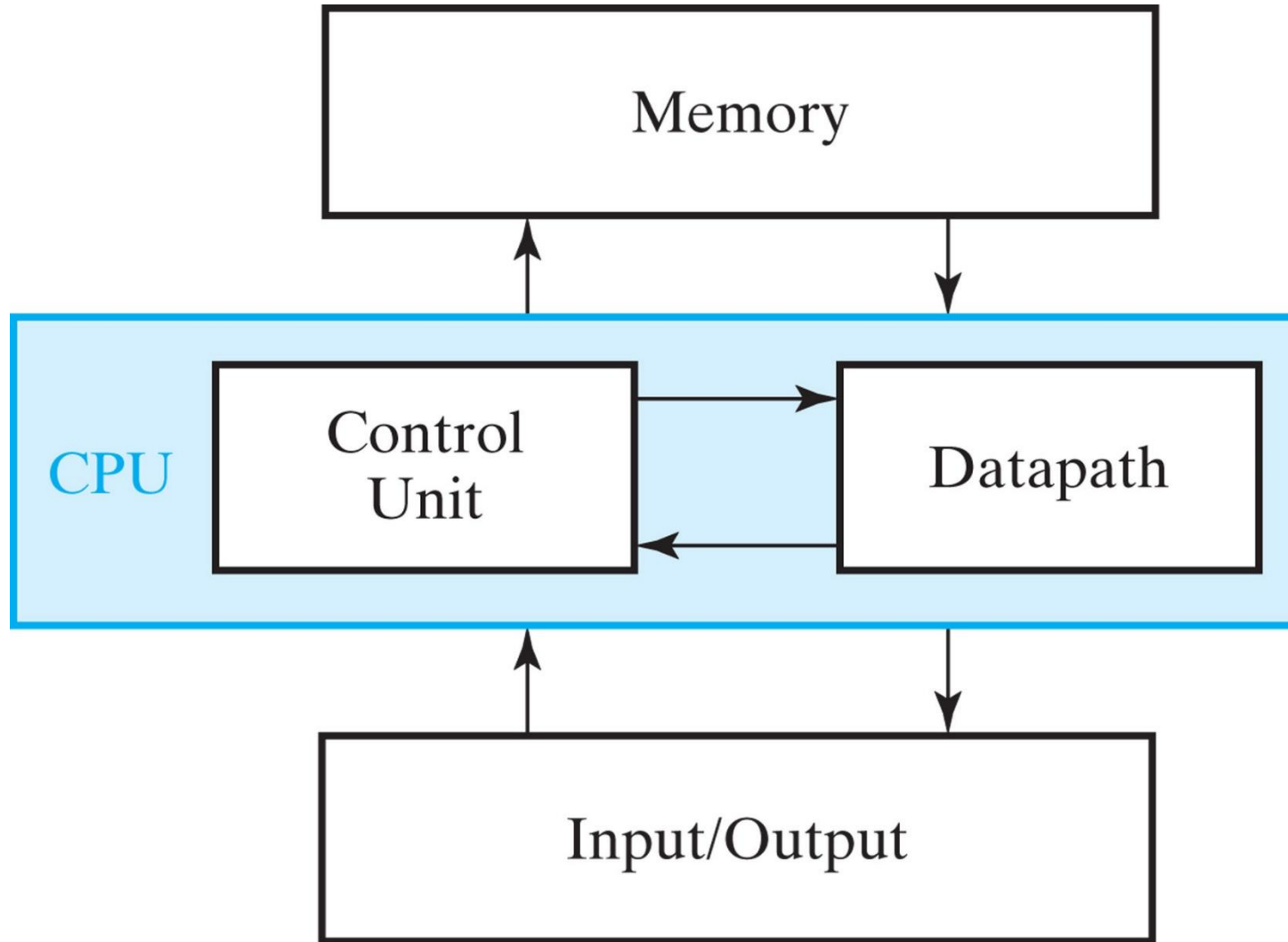


Intel Core 2 Architecture

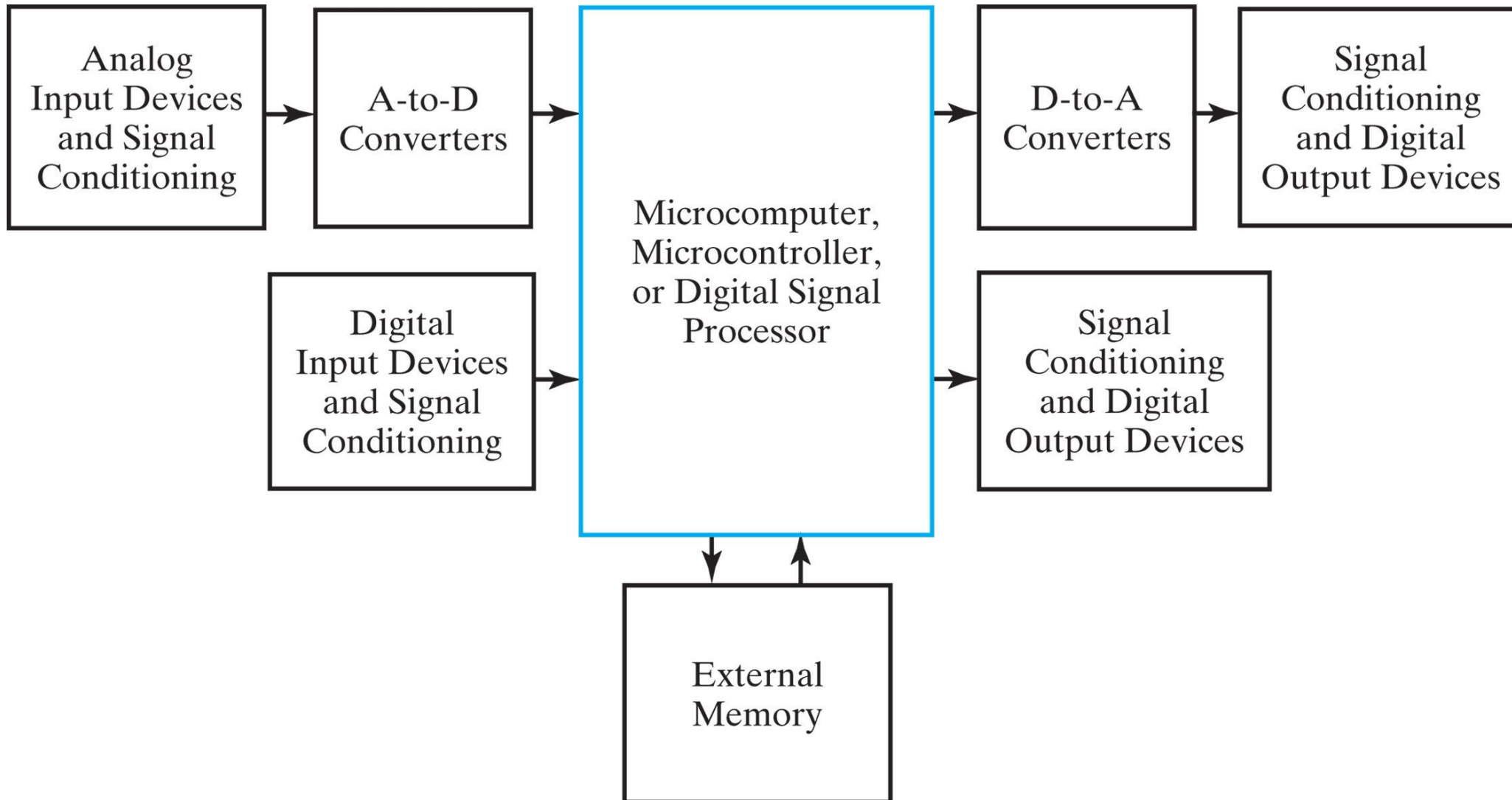


x86 instruction encoding

▶ Digital Computer



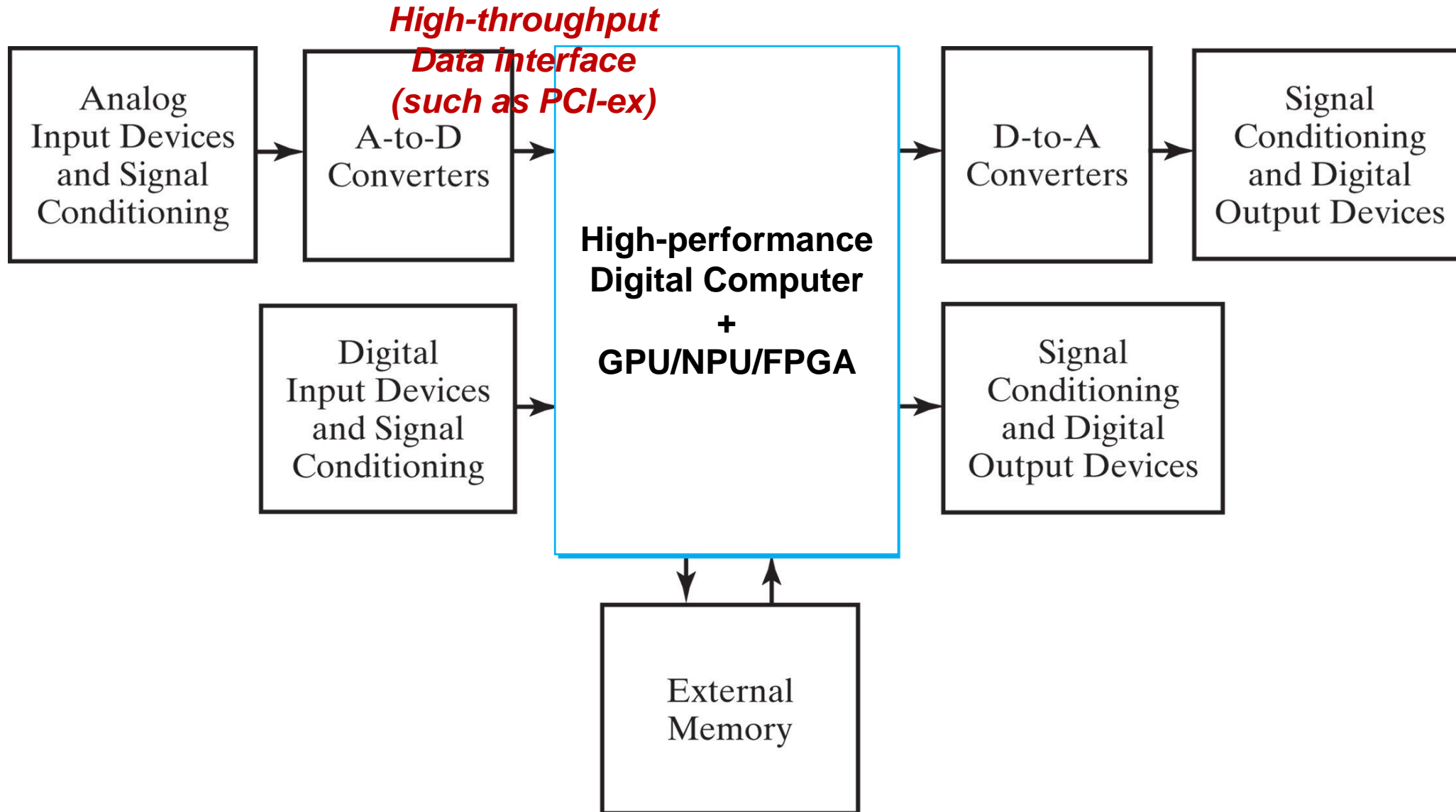
► Embedded System



► Embedded System

Application Area	Product
Banking, commerce and manufacturing	Copiers, FAX machines, UPC scanners, vending machines, automatic teller machines, automated warehouses, industrial robots, 3D printers
Communication	Wireless access points, network routers, satellites
Games and toys	Video games, handheld games, talking stuffed toys
Home appliances	Digital alarm clocks, conventional and microwave ovens, dishwashers
Media	CD players, DVD players, flat panel TVs, digital cameras, digital video cameras
Medical equipment	Pacemakers, incubators, magnetic resonance imaging
Personal	Digital watches, MP3 players, smart phones, wearable fitness trackers
Transportation and navigation	Electronic engine controls, traffic light controllers, aircraft flight controls, global positioning systems

► Trend: H/W-based to S/W-based Architecture



► Bottleneck in S/W-based architecture : Data transfer

Example,

Premium Ultrasound System Data Input

- A/D-converter (ADC) Resolution: 16 bits (= 2 Bytes)
- A/D-converter (ADC) Speed: 80 MHz
- No of A/D-converter (ADC): 256 channels

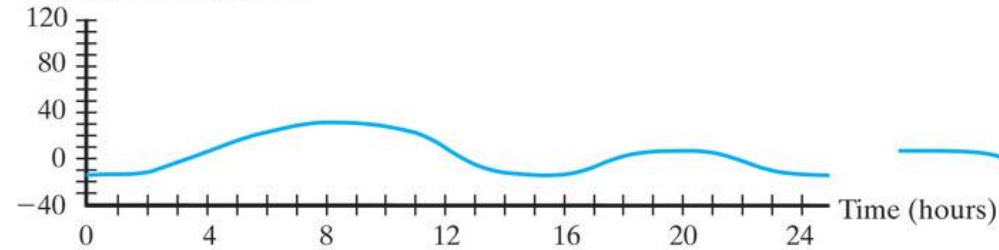
Estimated data rate: $256 \times 80\text{e}6 \times 2 = 40 \text{ GB/s}$

PCI Express link performance^{[39][40]}

Version	Introduced	Line code	Transfer rate ^{[i][ii]}	Throughput ^{[i][iii]}				
				x1	x2	x4	x8	x16
1.0	2003	8b/10b	2.5 GT/s	250 MB/s	500 MB/s	1.0 GB/s	2.0 GB/s	4.0 GB/s
2.0	2007	8b/10b	5.0 GT/s	500 MB/s	1.0 GB/s	2.0 GB/s	4.0 GB/s	8.0 GB/s
3.0	2010	128b/130b	8.0 GT/s	984.6 MB/s	1969.2 MB/s	3938.5 MB/s	7.877 GB/s	15.754 GB/s
4.0	2017	128b/130b	16.0 GT/s	1969.2 MB/s	3938.5 MB/s	7.877 GB/s	15.754 GB/s	31.508 GB/s
5.0	2019	128b/130b	32.0 GT/s ^[iv]	3938.5 MB/s	7.877 GB/s	15.754 GB/s	31.508 GB/s	63.015 GB/s
6.0 (planned)	2021	128b/130b & PAM-4	64.0 GT/s	7.877 GB/s	15.754 GB/s	31.508 GB/s	63.015 GB/s	126.03 GB/s

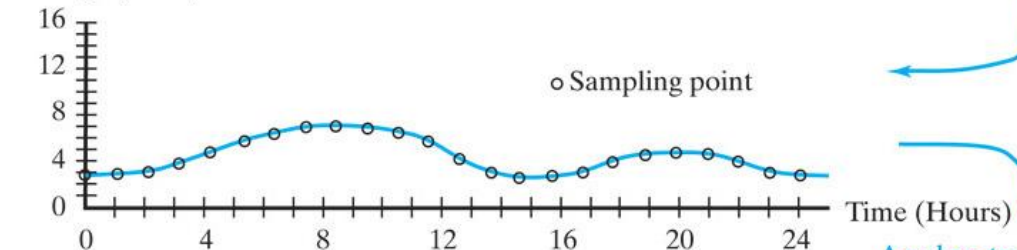
▶ from Real World to Digital World, vice versa

Temperature (degrees F)



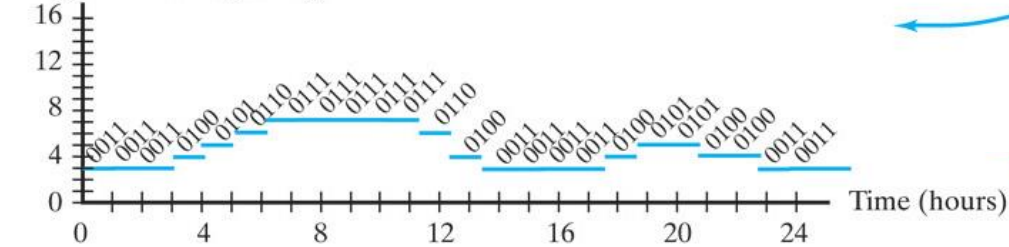
(a) Analog temperature

Voltage (Volts)



(b) Continuous (analog) voltage

Digital numbers (binary)



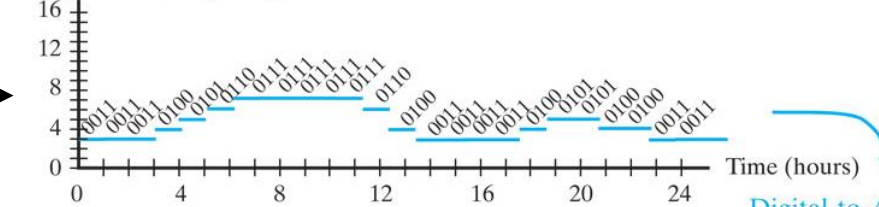
(c) Digital voltage

Sensor and
Signal Conditioning

Analog-to-Digital
(A/D) Conversion

**Signal
processing**

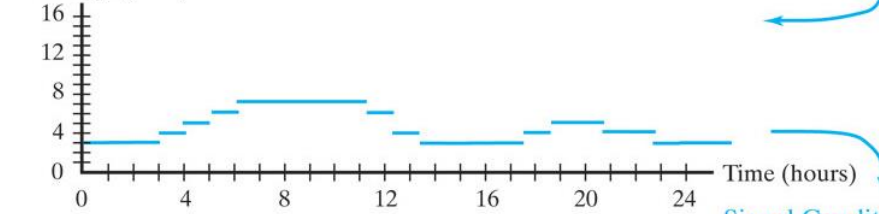
Digital numbers (binary)



(c) Digital voltage

Digital-to-Analog
(D/A) Conversion

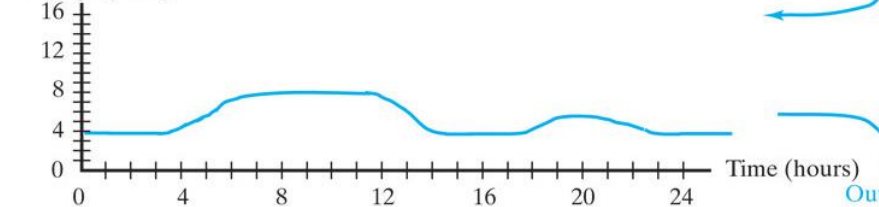
Voltage (volts)



(d) Discrete (digital) voltage

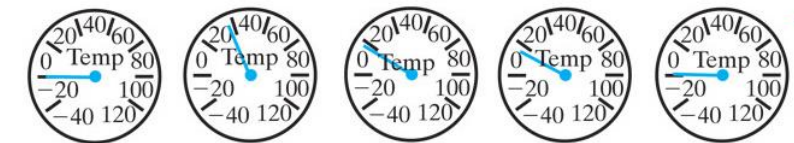
Signal Conditioning

Voltage (volts)



(e) Continuous (analog) voltage

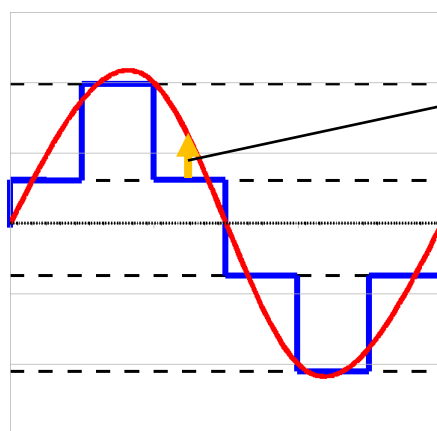
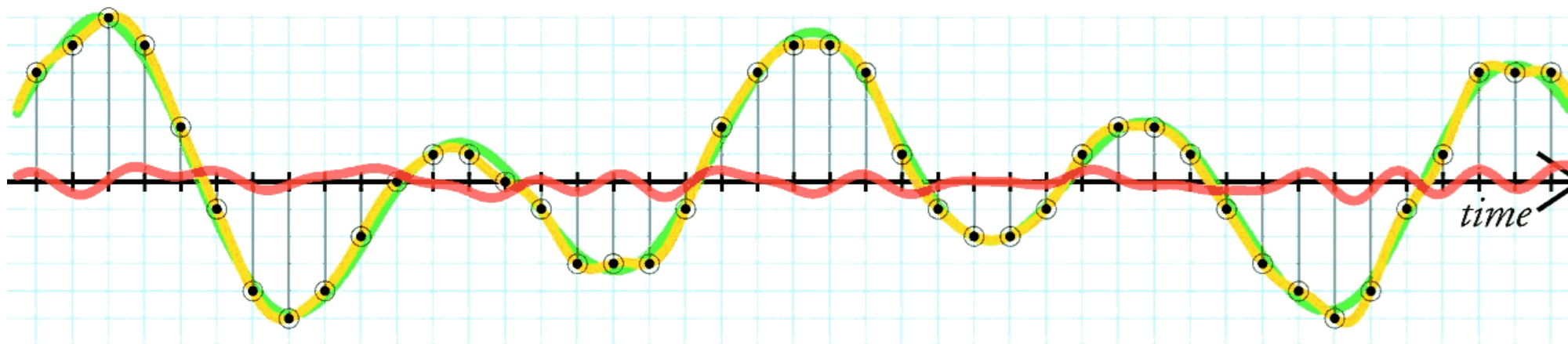
Output



(f) Continuous (analog) readout

Quantization Error

original signal
quantized signal
quantization noise

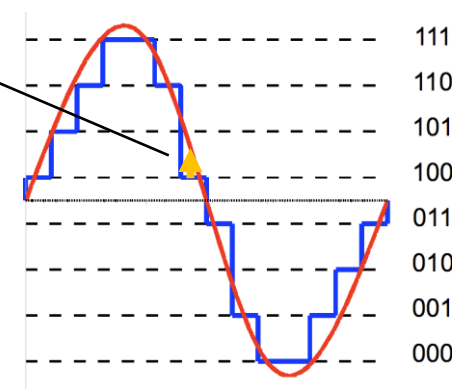


2-bits

11
10
01
00

Quantization error

vs



3-bits

▶ Example: Compact Disc Digital Audio / MP3

• The specification of CD Digital Audio (Red book)

- Sampling rate: 44,100Hz (doubling of maximum audible frequency) Nyquist theory!
- Sampling resolution: 16 bits (65,536 digitized levels)
 - Signal-to-quantization-noise ratio: $6.02 \times 16 = 96.3 \text{ dB}$ (can be negligible)
- No of Channels: 2 (Stereo)
- Required data rate: $44,100 \text{ Hz} \times 2 \text{ Bytes} \times 2 \text{ Channels} = 172.3 \text{ KB/s}$
- Full Audio-CD read Time: 74min (@ 734 MB)

The capacity (=Disc size) was decided for recording of Ludwig van Beethoven's Ninth Symphony (74mins).

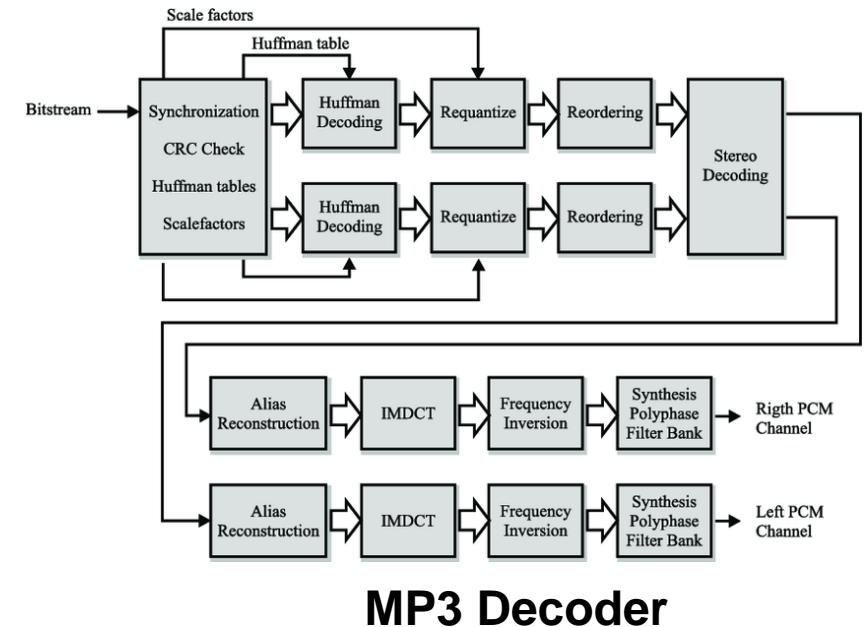
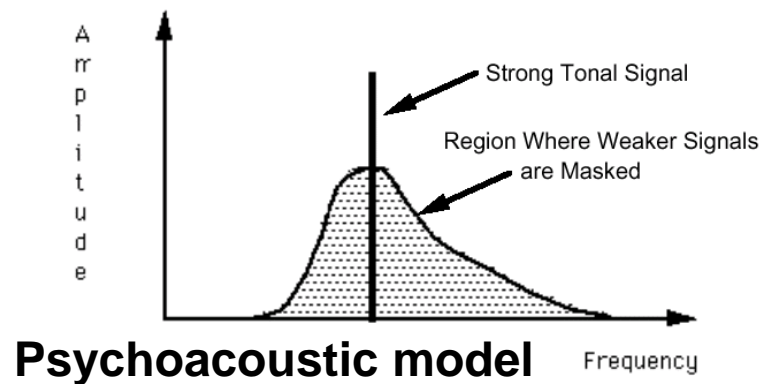


COMPACT
disc
DIGITAL AUDIO

▶ Example: Compact Disc Digital Audio / MP3

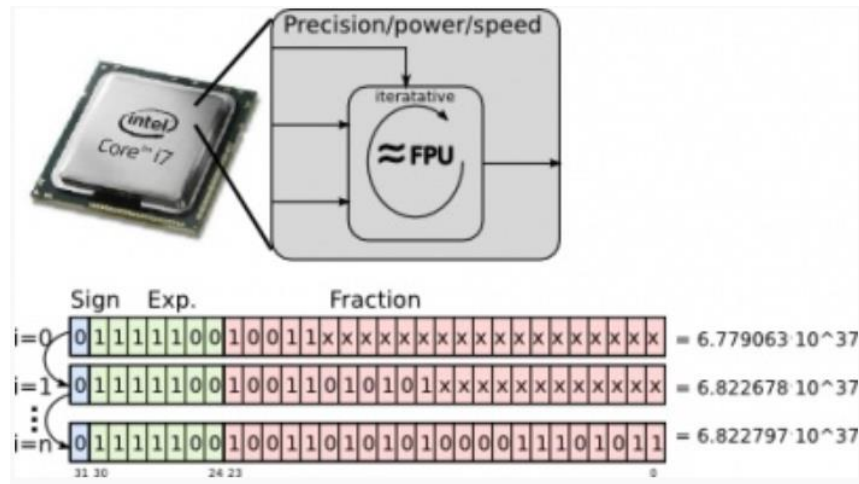
• The specification of MP3 (MPEG 1.0 Audio Layer-3)

- Audio part of the Video-CD (MPEG 1.0) format
- The specification was determined in part to ensure compatibility with existing CD drive technology, specifically the earliest "1x" speed CD drives.
 - Video: 1,150 Kbit/s + **Audio: 224Kbits/s** (vs CD-DA: 1,376 Kbits/s)
- Compression technology is required (around 1:10)
 - Based on Psychoacoustic model
 - MP3 should be compressed around 128~256 Kbits/s

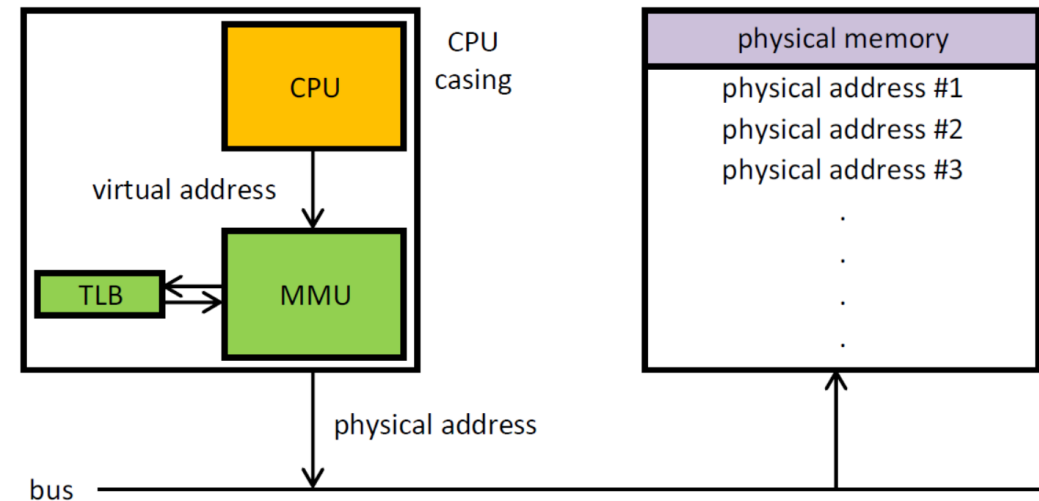


▶ Structure of a processor

- Processor consists of CPU, FPU, MMU, and on-chip memory
 - CPU: control unit + data path (arithmetic and other data-processing operations)
 - FPU: Floating-Point Unit
 - MMU: Memory Management Unit



FPU



CPU: Central Processing Unit
MMU: Memory Management Unit
TLB: Translation lookaside buffer

MMU

▶ Structure of a processor – Historical processor



- Introduced in 1985
- First 32bit processor: Instruction set and Architecture (IA-32)
- Compatible processors of 80386: x86 (i386) Architecture
- Manufactured until 2007 for Aerospace or Embedded System (80386EX)
 - Used aboard several orbiting satellites and microsatellites
 - Used in NASA's Flight Linux project
 - Used in US Robotics Courier I modem V.everything
 - Used in Ericsson R290 satellite phone
 - Used in many older Garmin GPS units, such as the GPS 48, II, III, and 12.
 - Used in Akai S5000 & S6000 digital samplers.
- External **FPU** (i387) – integrated into 80486 CPU



► Number systems: Base-10 (Decimal)

- Decimal numbers (base-10)

- 7,392 can be represented by $7 \times 10^3 + 3 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$
- The convention is to write only the numeric coefficients and deduce the necessary powers of 10 with powers increasing from right to left

$$a_5a_4a_3a_2a_1a_0.a_{-1}a_{-2}a_{-3}$$



$$10^5a_5 + 10^4a_4 + 10^3a_3 + 10^2a_2 + 10^1a_1 + 10^0a_0 + 10^{-1}a_{-1} + 10^{-2}a_{-2} + 10^{-3}a_{-3}$$

► Number systems: Base-2 (Binary)

- Binary numbers (base-2)

- The decimal equivalent of the binary number 11010.11 is 26.75

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 26.75$$

- 2^{10} is referred to as K (kilo), 2^{20} as M (mega), and 2^{30} as G (giga)

Table 1.1
Powers of Two

<i>n</i>	2^n	<i>n</i>	2^n	<i>n</i>	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024 (1K)	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096 (4K)	20	1,048,576 (1M)
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

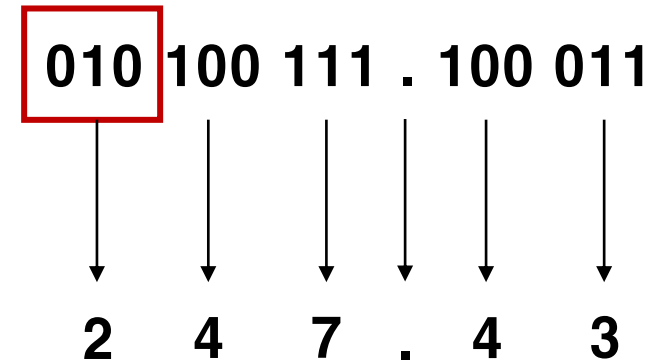
► Number systems: Base-8 (Octal)

- Octal numbers (base-8)

- “Direct” Conversion from binary to octal for the reduced representation
- ex) Used in Linux/Unix for file permissions (chmod)

Octal number	Binary number
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Binary Number: **10100111.100011₂**



Octal Number: **247.43₈**

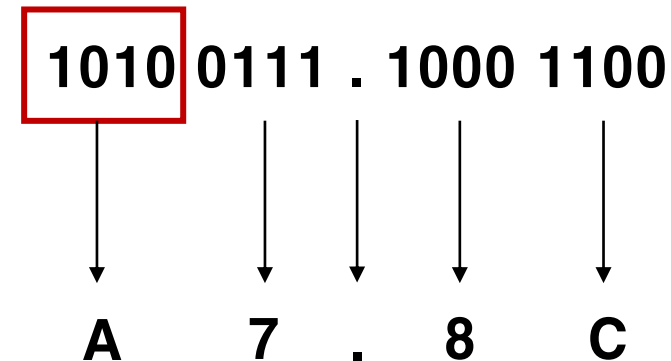
► Number systems: Base-16 (Hexadecimal)

• Octal numbers (base-16)

- “Direct” Conversion from binary to hexadecimal for the reduced representation
- Typical number system in Digital Computer

Hex.	Binary	Hex.	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Binary Number: **10100111.100011₂**



Hexadecimal Number: **A7.8C₁₆**

► Number systems: Base- r

- A number with an arbitrary base ' r '

- It contains r different digits
- It can be expressed as

$$A_{n-1}r^{n-1} + A_{n-2}r^{n-2} + \dots + A_1r^1 + A_0r^0 \\ + A_{-1}r^{-1} + A_{-2}r^{-2} + \dots + A_{-m+1}r^{-m+1} + A_{-m}r^{-m}$$

- Example: a base-5 number

$$(312.4)_5 = 3 \times 5^2 + 1 \times 5^1 + 2 \times 5^0 + 4 \times 5^{-1} \\ = 75 + 5 + 2 + 0.8 = (82.8)_{10}$$

Numbers with Different Bases

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

▶ A Metric Prefix - Kilo / Mega / Giga / Tera

- Representation difference SI vs JEDEC

- Manufacture (following SI standard)

- Kilo = 10^3 / Mega = 10^6 / Giga = 10^9 / Tera = 10^{12}

- OS (following JEDEC standard)

- Kilo = 2^{10} / Mega = 2^{20} / Giga = 2^{30} / Tera = 2^{40}

- ex) for 512 GB Hard disk,

- Manufacture representation: 512 GB = 512×10^9 Bytes (Real capacity)

- Windows representation : $512 \times 10^9 \text{ Bytes} / 2^{30} = 476 \text{ GB}$

Local Disk (C:) - 476 GB



This is how your storage is used and how you can free up space.

512GB SSD shown in Windows as 476 GB

► Arithmetic Operations of Binary Numbers

- In a binary system, we only allow two digits at each bit-position
 - Only 0 and 1 are allowed at each bit
 - These operations are basics of digital computers

augend:	101101	minuend:	101101	multiplicand:	1011
addend:	<u>+100111</u>	subtrahend:	<u>-100111</u>	multiplier:	<u>× 101</u>
sum:	1010100	difference:	000110		1011
				partial product:	0000
					1011
				product:	<u>110111</u>

► Addition

- Carries are important as we will see in later lectures
 - Only 0 and 1 are allowed at each bit
 - A carry in binary occurs if the sum in any bit position is greater than 1

Carries:	00000	101100
Augend:	01100	10110
Addend:	+10001	+10111
Sum:	<hr/> 11101	<hr/> 101101

► Subtraction

- The rules for subtraction are the same as in decimal
 - Except it borrows 2 from higher bit-position
 - For the case that subtrahend is larger than the minuend, we **subtract the minuend from the subtrahend** and give the result a minus sign

Borrows:	00000	00110		00110
Minuend:	10110	10110	10011	11110
Subtrahend:	-10010	-10011	-11110	-10011
Difference:	<hr/> 00100	<hr/> 00011	<hr/>	<hr/> -01011

► Multiplication

- The multiplier digits are always 0 or 1 in a binary system
 - The partial products are equal either to the multiplicand or to 0

Multiplicand:	1011
Multiplier:	× 101
	<hr/>
	1011
	0000
	1011
	<hr/>
Product:	110111

► Hexadecimal/Octal Addition

- Perform the addition $(59F)_{16} + (E46)_{16}$

Hexadecimal	Equivalent Decimal Calculation
$\begin{array}{r} 59F \\ E46 \\ \hline 13E5 \end{array}$	<div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: right;"> $\begin{array}{r} 1 \leftarrow \\ 5 \\ 14 \\ \hline 19 = 16 + 3 \end{array}$ </div> <div style="text-align: right;"> $\begin{array}{r} 1 \leftarrow \\ 9 \quad 15 \\ 4 \quad 6 \\ \hline 14 = E \quad 21 = 16 + 5 \end{array}$ </div> </div>

- Perform the addition $(762)_8 + (45)_8$

Octal	Octal	Decimal	Octal
7 6 2	5×2	$= 10 = 8 + 2$	$= 12$
4 5	$5 \times 6 + 1$	$= 31 = 24 + 7$	$= 37$
4 6 7 2	$5 \times 7 + 3$	$= 38 = 32 + 6$	$= 46$
3 7 1 0	4×2	$= 8 = 8 + 0$	$= 10$
4 3 7 7 2	$4 \times 6 + 1$	$= 25 = 24 + 1$	$= 31$
	$4 \times 7 + 3$	$= 31 = 24 + 7$	$= 37$

► Signed Integer – 2's complement

- Representation of signed integer
- ex) Representation of “-7”

0111



1000



1001

Step 1: Positive number
(= 7)

Step 2: Take a 1's complement
(Invert each bit)

Step 3: +1 (if negative)
2's Complement

Bit pattern: 1000 1001 1010 1011 1100 1101 1110 1111 0000 0001 0010 0011 0100 0101 0110 0111

1's complement: -7 -6 -5 -4 -3 -2 -1 0 0 1 2 3 4 5 6 7

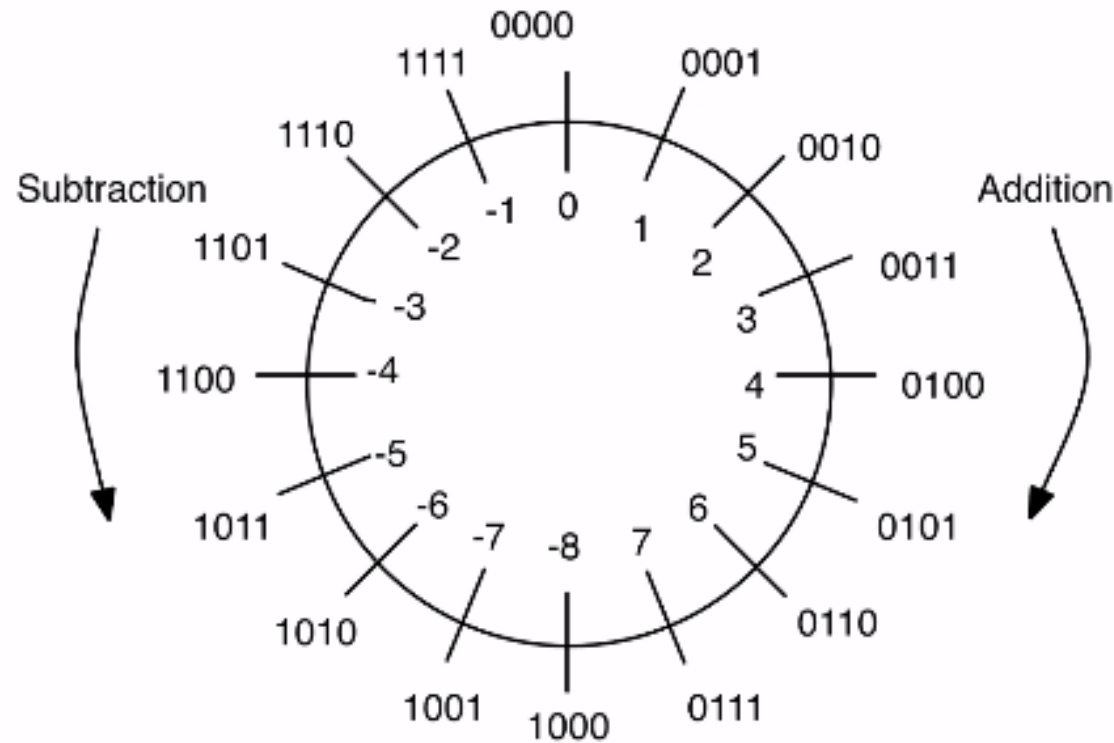
2's complement: -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7

- Why? > Faster (in perspective of the hardware)

► Signed Integer – 2's complement

- Representation of signed integer

N=4



Bit pattern: 1000 1001 1010 1011 1100 1101 1110 1111 0000 0001 0010 0011 0100 0101 0110 0111

1's complement: -7 -6 -5 -4 -3 -2 -1 0 0 1 2 3 4 5 6 7

2's complement: -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7

► Signed Integer – 2's complement

- ex) Subtraction 6-3

Case 1: 6 - 3

	0	1	1	0
-	0	0	1	1
<hr/>				
	0	0	1	1

Case 2: 6+(-3) by 2's complement

	0	1	1	0
+	1	1	0	1
<hr/>				
	0	0	1	1

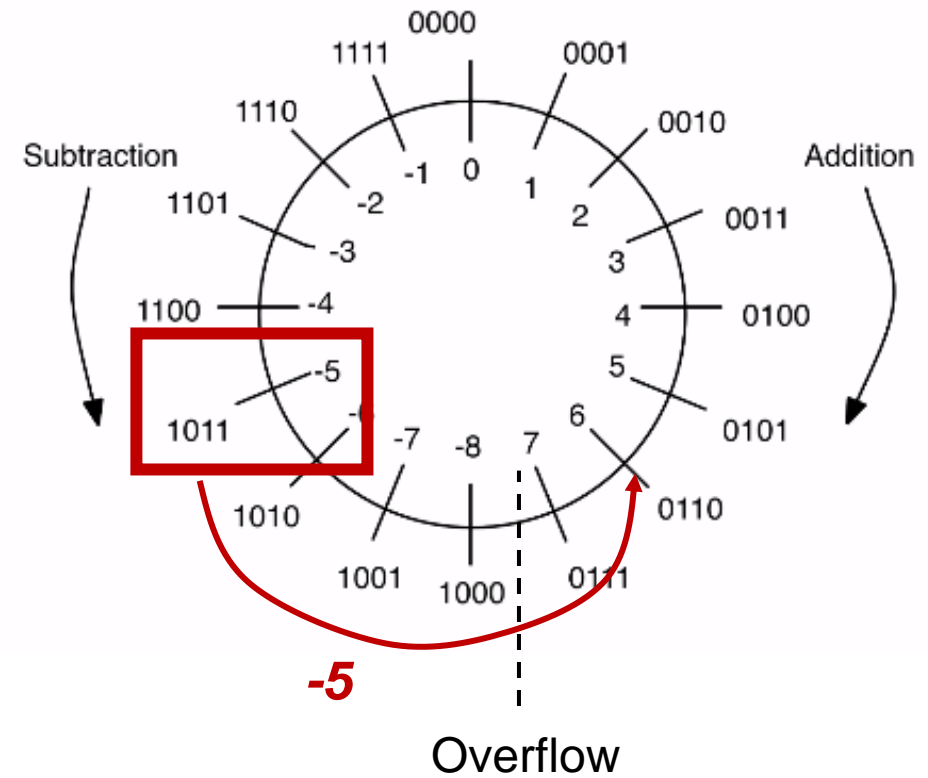
► Signed Integer – 2's complement

- ex) Overflow in subtraction

Case 1: $-5 - 5$

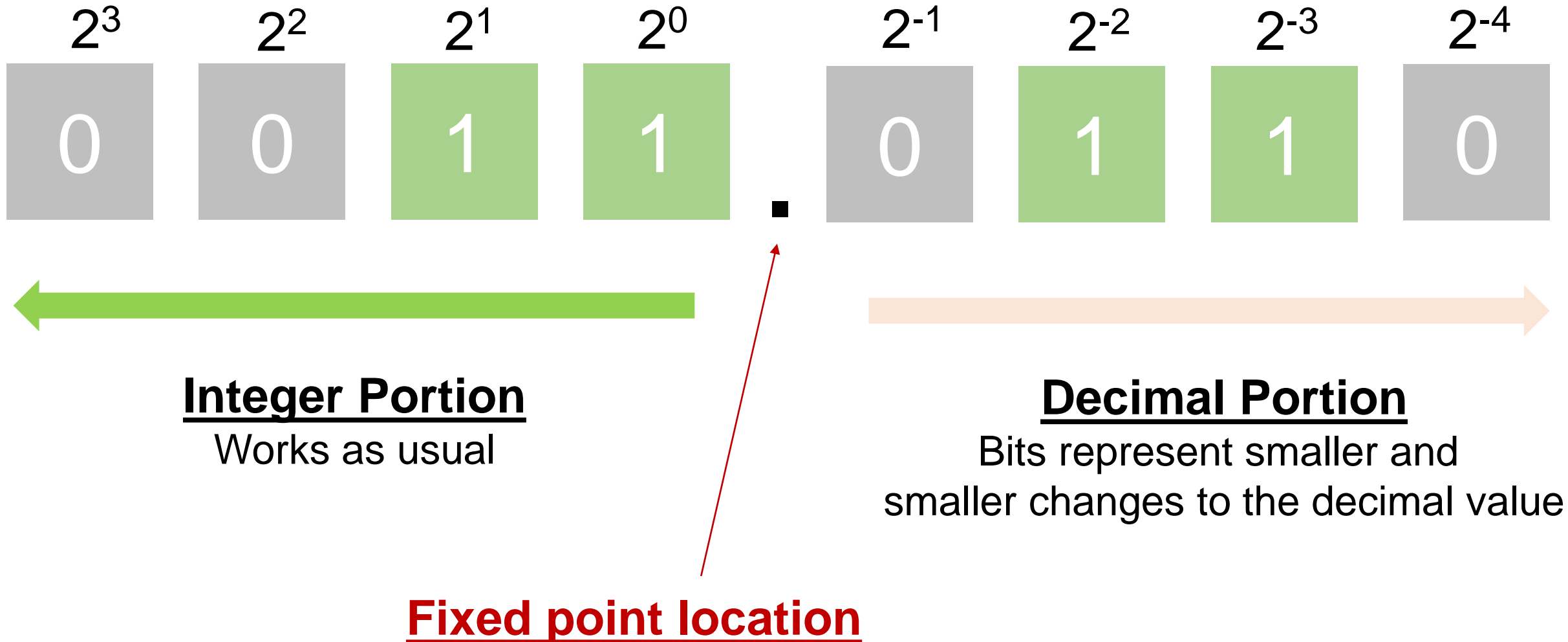
	1	0	1	1	-5
+	1	0	1	1	-5
<hr/>					
	0	1	1	0	6 (Overflow)

N=4



We will discuss in more detail in the future lecture

► Fixed Point Numbers



► Fixed Point Numbers

For 8-bits unsigned Integer

Smallest value = 0

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

Largest value = 255 (2^8-1)

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

For 8-bits unsigned Fixed Point

Smallest value = 0

0	0	0	0	.	0	0	0
---	---	---	---	---	---	---	---

Largest value = 15.9375

1	1	1	1	.	1	1	1
---	---	---	---	---	---	---	---

Limitation: You can only represent 16 possible decimal numbers (8-bit fixed point)

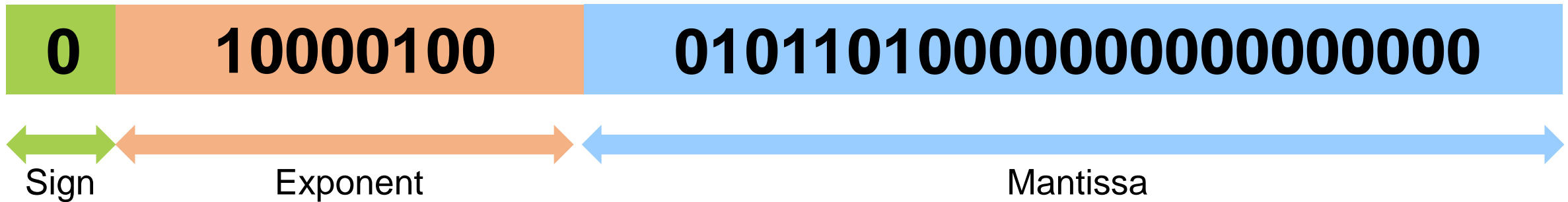
▶ Floating Point Numbers (IEEE 754)

43.25

$> 101011.01_2$

$= +1.0101101_2 \times 2^5$

0 = Positive
1 = Negative



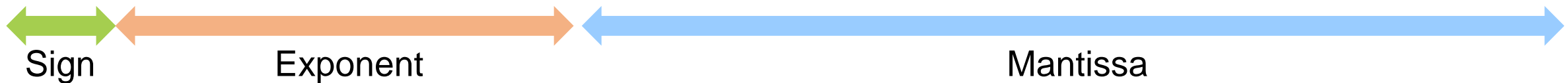
▶ Floating Point Numbers (IEEE 754)

43.25

$$= + 1.0101101_2 \times 2^5$$

$$5 + 127 (\text{bias offset for negative expression}) = 132 = 1000100_2$$

1 0 0 0 0 1 0 0

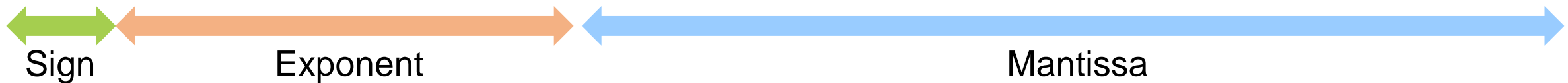


Reserved: 0, 255 / Available range 1 ~ 254

▶ Floating Point Numbers (IEEE 754)

$$43.25 = + 1.0101101_2 \times 2^5$$

1 0 1 0 1 1 0 1

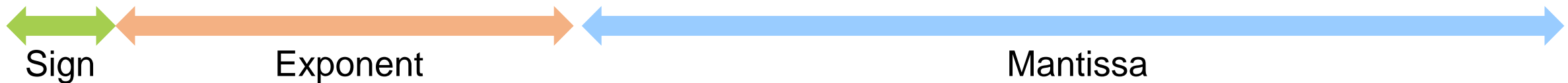


▶ Floating Point Numbers (IEEE 754)

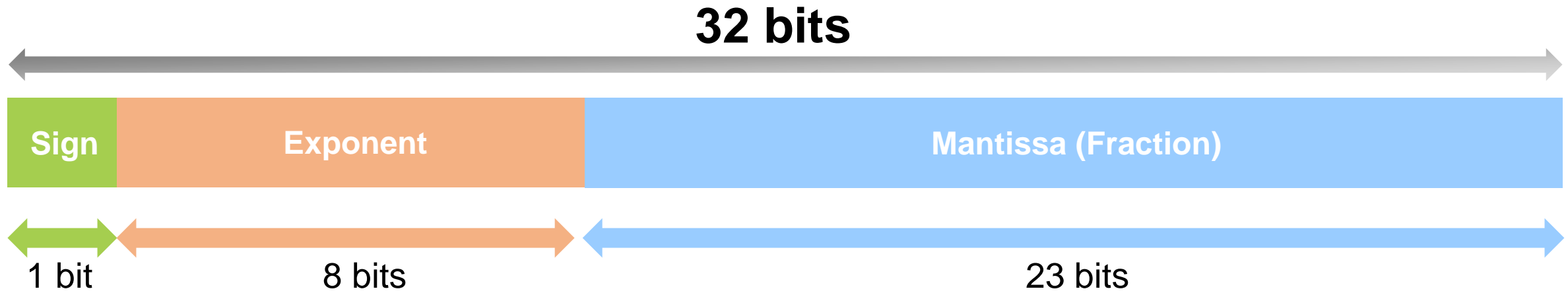
43.25

$$= + \boxed{1.0101101}_2 \times 2^5$$

Most Significant Bit (MSB) is always 1



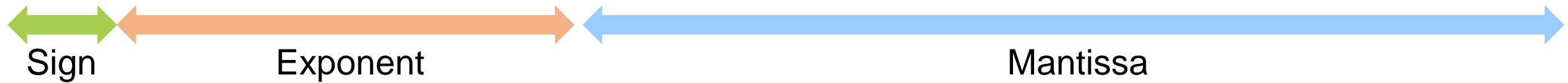
▶ Floating Point Numbers (IEEE 754 - Single precision)



$$\text{Sign} \left(1 + \text{Mantissa} \right) \times 2^{(\text{Exponent} - 127)}$$

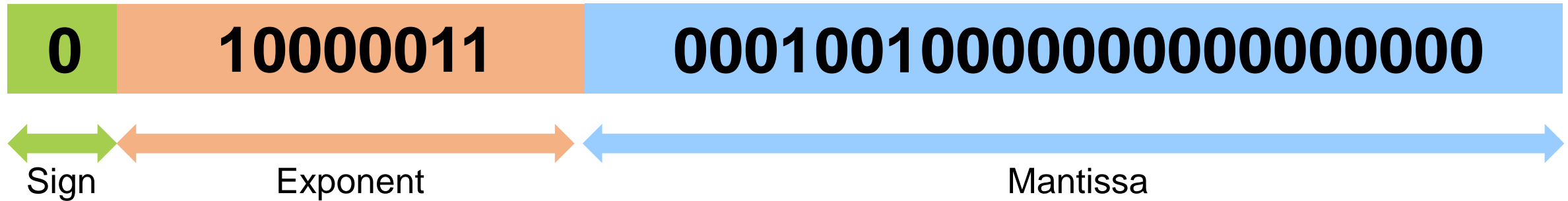
Offset number
(like 2's complement)

▶ Floating Point Numbers (IEEE 754)

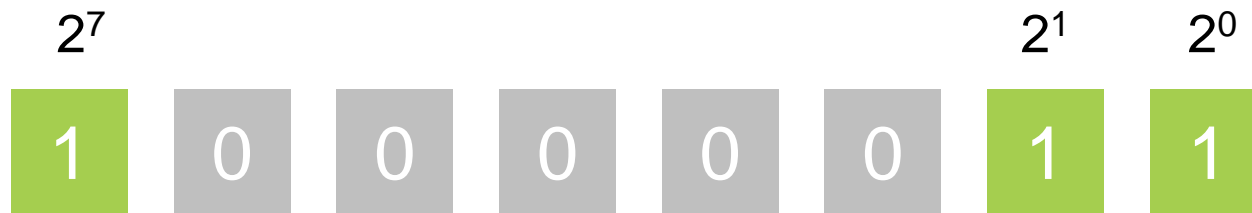


0 = Positive
1 = Negative

▶ Floating Point Numbers (IEEE 754)



Reserved: 0 & 255(0xFF)



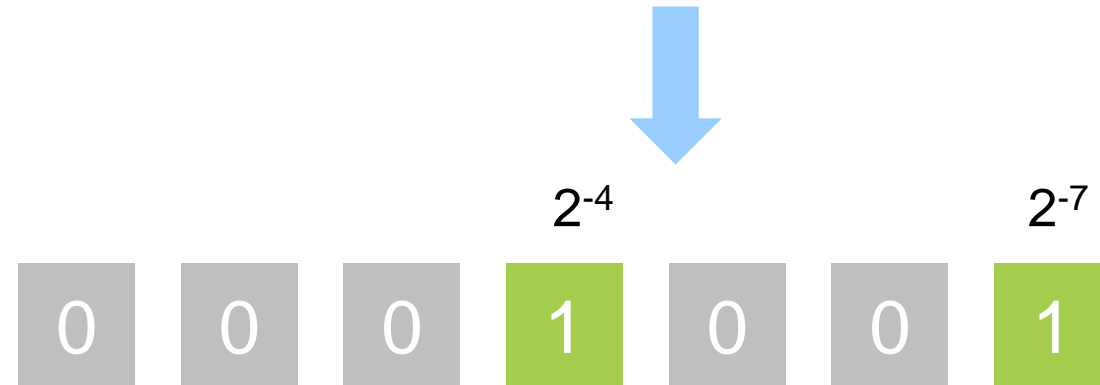
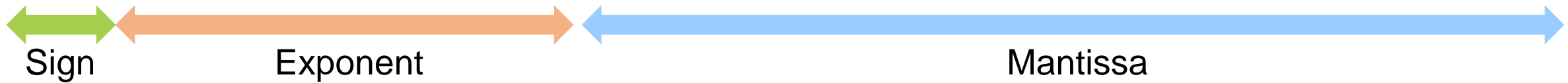
$$= 2^7 + 2^1 + 2^0$$

$$= 131$$



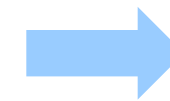
$$131 - 127 \text{ (offset number)} = 4$$

▶ Floating Point Numbers (IEEE 754)



$$= 2^{-4} + 2^{-7}$$

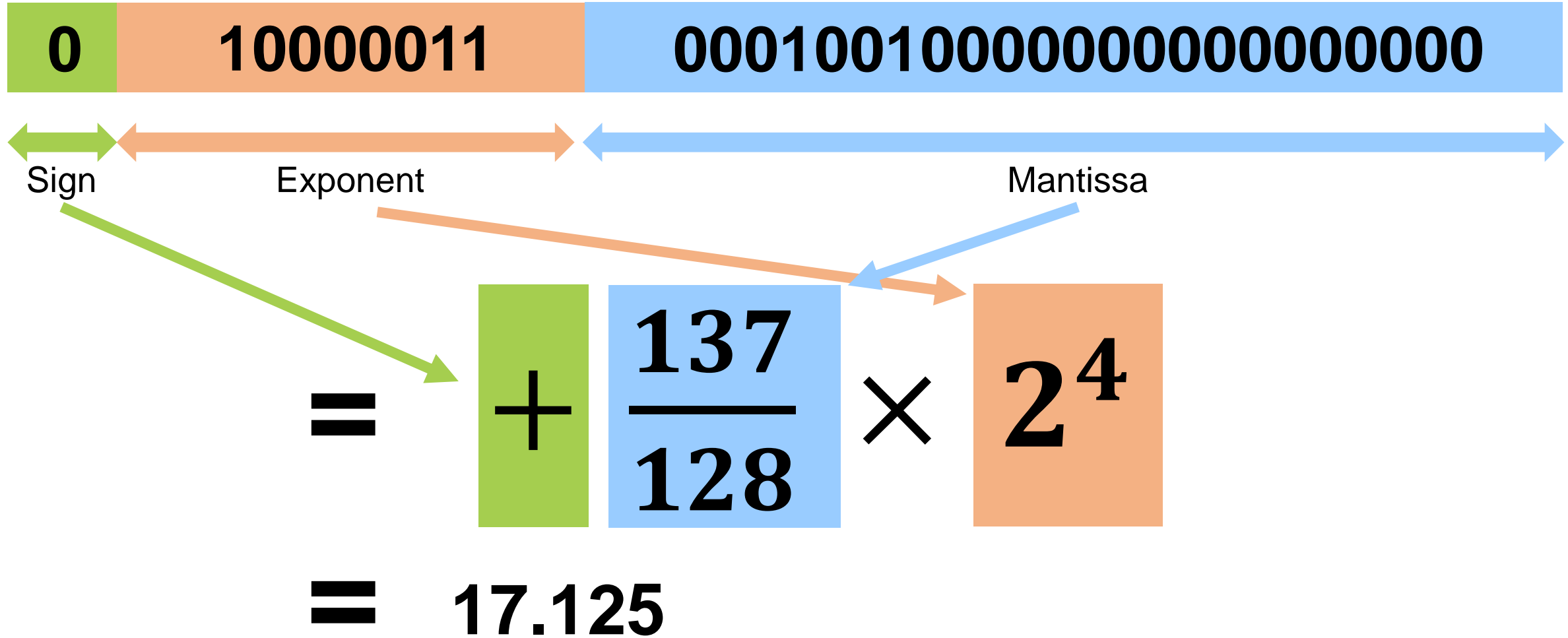
$$= 9/128$$



$$9/128 + 1$$

$$= 137/128$$

▶ Floating Point Numbers (IEEE 754)



► Limitations of Fixed Point Numbers

For 8-bits unsigned integer,

Smallest value = 0

Largest value = 255 (2^8-1)

For 8-bits unsigned Fixed Point

Smallest value = 0

Largest value = 15.9375

▶ Floating Point vs Fixed Point

- Need to understand the different binary number representations

32-Bit Fixed Point Notation



Sign

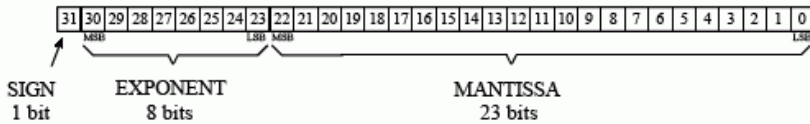
Bits

32-Bit Floating Point Notation (IEEE 754 Binary32 Standard)



Sign Exponent

Mantissa



Example 1

0	00000111	110000000000000000000000	
↓	↓	↓	
+	7	0.75	$+ 1.75 \times 2^{(7-127)} = + 1.316554 \times 10^{-36}$

Example 2

1	10000001	011000000000000000000000	
↓	↓	↓	
-	129	0.375	$- 1.375 \times 2^{(129-127)} = - 5.500000$

For Fixed Point Notation:

Integer Value = $-1\text{Sign} \times \text{Bits}$

Sign = 0 (Positive value) or 1 (Negative value)

Bits = 231 possible values

Positive Min: 1

Positive Max: +2147483647

With simple scaling, fractional numbers can be represented as well. The represented values are equally spaced across the whole range. The gaps between adjacent values are always the same.

For 32-bit Floating Point (IEEE 754):

Value = $-1\text{Sign} \times [(1 + \text{Mantissa}) \times 2^{(\text{Exponent}-127)}]$

Sign = 0 (Positive value) or 1 (Negative value)

Exponent = 1 to 254 (0 and 255 are reserved for special cases). Subtracting 127 gives -126 to 127

Mantissa = 0 to 0.999999881 for all '0s' to all '1s'. Adding 1 gives 1.000000000 to 1.999999881

Positive Min: $1 \times 2^{-126} \approx 1.2 \times 10^{-38}$

Positive Max: $1.99999881 \times 2^{127} \approx 3.4 \times 10^{38}$

▶ Floating Point vs Fixed Point

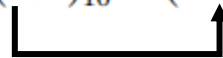
- Need to understand the different binary number representations

FIXED POINT	FLOATING POINT
A representation of real data type for a number that has a fixed number of digits after the radix point	A formulaic representation of real numbers as an approximation so as to support a tradeoff between range and precision
Used to represent a limited range of values	Used to represent a wide range of values
Higher performance	Lower performance
Less flexible	More flexible

► Decimal Code (BCD code)

- The binary number system is the most natural one for a computer
 - But, people are accustomed to the decimal system
 - One way around is that we store a conversion table (from decimal to binary) in computers
 - An n-bit binary code may represent up to 2^n distinct symbols

Example of using BCD

$$(185)_{10} = (0001 \ 1000 \ 0101)_{\text{BCD}} = (10111001)_2$$


Binary-Coded Decimal (BCD)	
Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

► BCD addition

• Consider the addition of two decimal digits in BCD

- Each digit does not exceed 9, the sum cannot be greater than 19 (1 becomes a carry)
- Addition at each digit is an addition of two binary numbers (the result ranges from **0**:(0000)₂ to **19**:(1 0011)₂)

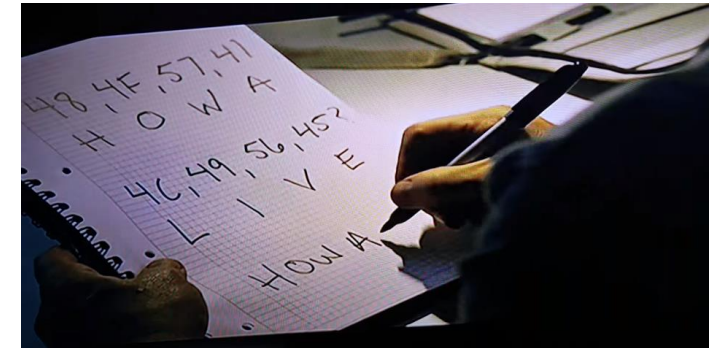
110 448 +489 ----- 937	<p>BCD carry</p> <p>Binary sum</p> <p>Add 6</p> <p>BCD sum</p> <p>BCD result</p>	<p>1 ←</p> <p>0100 +0100 ----- 1001</p> <p>1 ←</p> <p>0100 +1000 ----- 1101 +0110 ----- 1 0011</p> <p>1 ←</p> <p>1000 +1001 ----- 1 0001 +0110 ----- 1 0111</p>
		<p>1001</p> <p>0011</p> <p>0111</p>

▶ Alphanumeric Codes

- Many applications of digital computers require data with both numbers and letters
 - Insurance companies: **names** and other info in letters
 - Need to formulate a binary code for the letters of the alphabet
 - The codes must be in binary!
- ASCII, EBCDIC...

► ASCII Character Code

- The standard binary code for the alphanumeric characters
 - It uses 7 bits to code 128 characters $[B_7:B_6: \dots : B_1]$



Film "The Martian" (2015)
Space Communication using ASCII

American Standard Code for Information Interchange (ASCII)

$B_4B_3B_2B_1$	$B_7B_6B_5$							
	000	001	010	011	100	101	110	111
0000	NULL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Control Characters

NULL	NULL	DLE	Data link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End of transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

▶ How to use Korean in Computer?

- Beginning: External Korean font card to use Korean in ASCII
 - Non-standard – ex) lower-case letter + capital letter + Roman to represent



Korean Font Card



► Unicode (UTF-8)

- The standard binary code for the non-alphabet characters (Universal Coded Character Set + Transformation Format – 8-bit)
 - Variable-width character encoding
 - Cover all-existed characters in the world

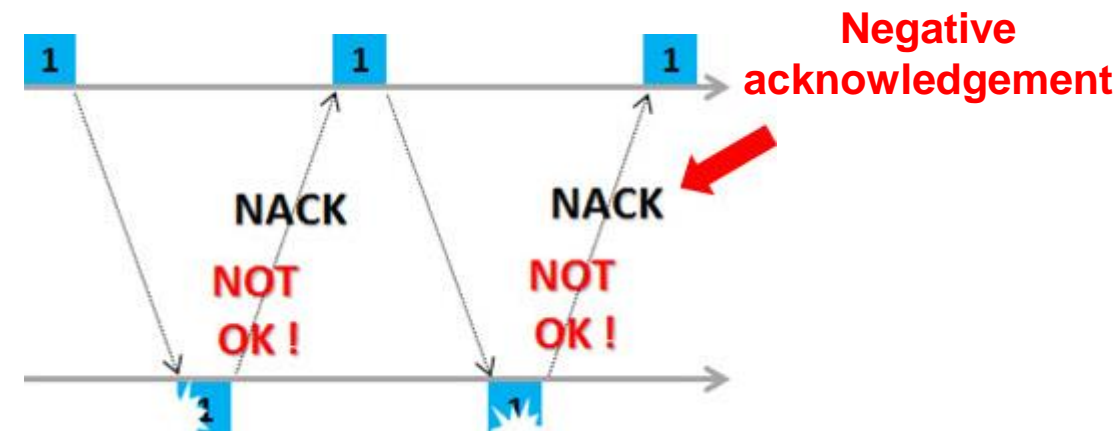
□ **TABLE 1-6**
UTF-8 Encoding for Unicode Code Points

Code point range (hexadecimal)	UTF-8 encoding (binary, where bit positions with x are the bits of the code point value)
U+0000 0000 to U+0000 007F	0xxxxxxx <i>Same as ASCII (for backward compatibility)</i>
U+0000 0080 to U+0000 07FF	110xxxxx 10xxxxxx
U+0000 0800 to U+0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx
U+0001 0000 to U+0010 FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

► Parity bit

- To detect errors in data communication and processing, an additional bit is sometimes added to a binary code
 - A **parity bit** is an extra bit included to make the total # of 1s in the resulting code either even or odd

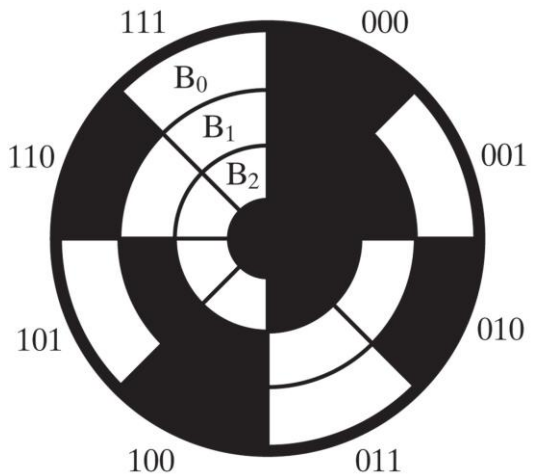
	With Even Parity	With Odd Parity
1000001	01000001	11000001
1010100	11010100	01010100



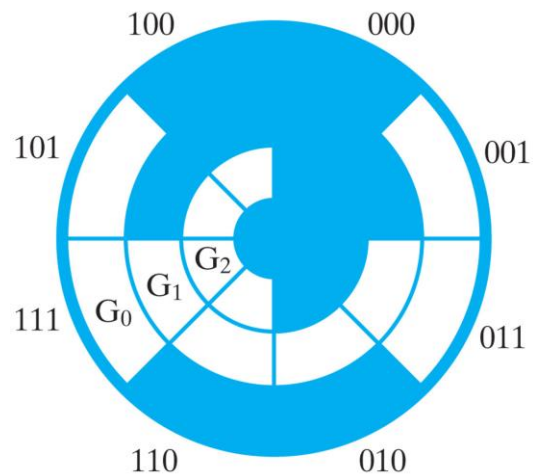
- Detection only (Not correction), CRC for correction
- If two bits are changed? > cannot detect

► Gray Code

- As we count up/down using binary codes, # of bits that change from one value to the next **varies**
- In some cases, it is good to encode symbols in a way that **a single bit changes** between two codes

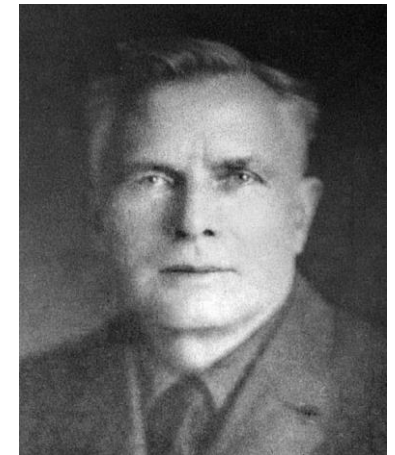


(a) Binary code for positions 0 through 7



(b) Gray code for positions 0 through 7

Gray Code			
Binary Code	Bit Changes	Gray Code	Bit Changes
000		000	
001	1	001	1
010	2	011	1
011	1	010	1
100	3	110	1
101	1	111	1
110	2	101	1
111	1	100	1
000	3	000	1



Patented by
Frank Gray
(1953)

▶ Summary

- Introduced digital systems and digital computers
 - Binary systems
 - Structure of stored-program digital computer
- Arithmetic operations in arbitrary number systems
- Binary codes for digital computers
 - BCD code
 - ASCII character code (+ parity bit)
 - Gray code