Lecture 1 Introduction (2021.9.7&9)

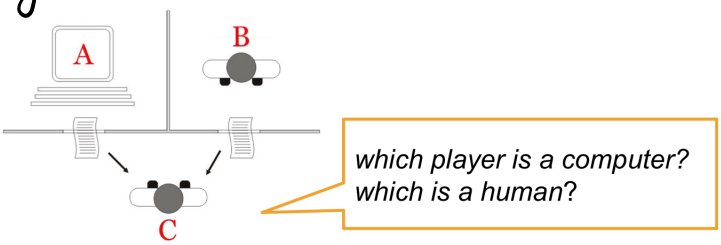
§1 The Story of Computer

1. The theoretical foundation of computer science

1° Built by Alan Turing (1912-1954)

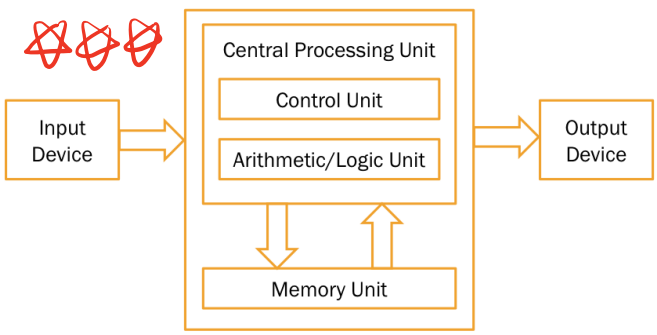
2° Turing is the father of theoretical computer science and artificial intelligence.

3° Turing Test

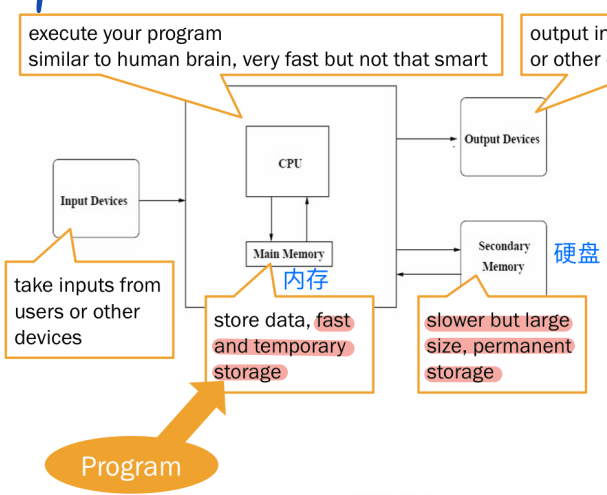


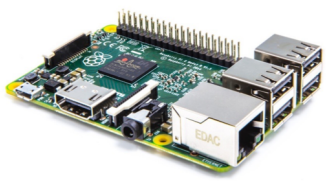
2. Von Neumann architecture

The modern computer architecture is proposed by John von Neumann



§2 Computer Hardware





Raspberry PI
single-board computer

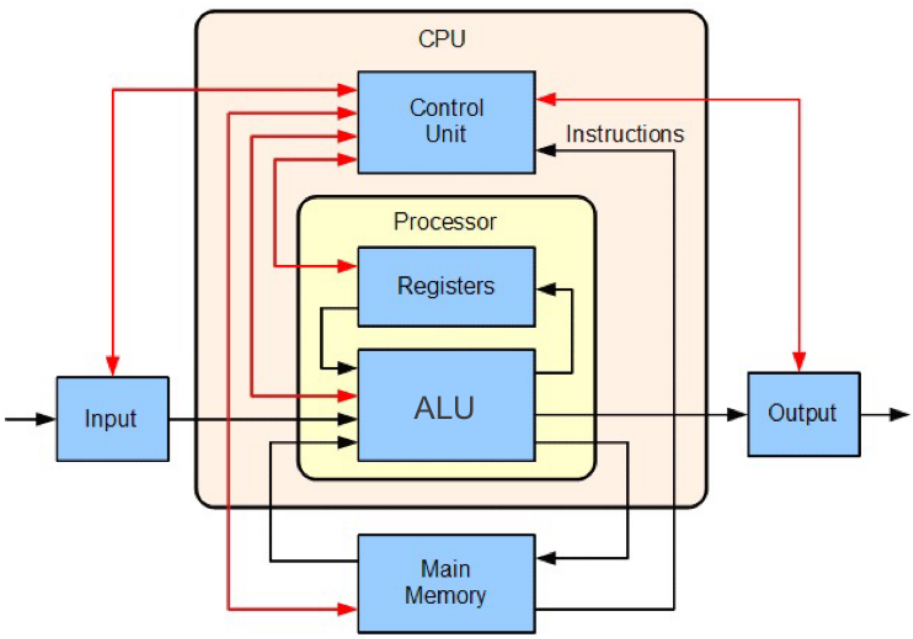
1. Central processing unit

1° A processor { Control Unit (CU)
Arithmetic / Logic Unit (ALU)

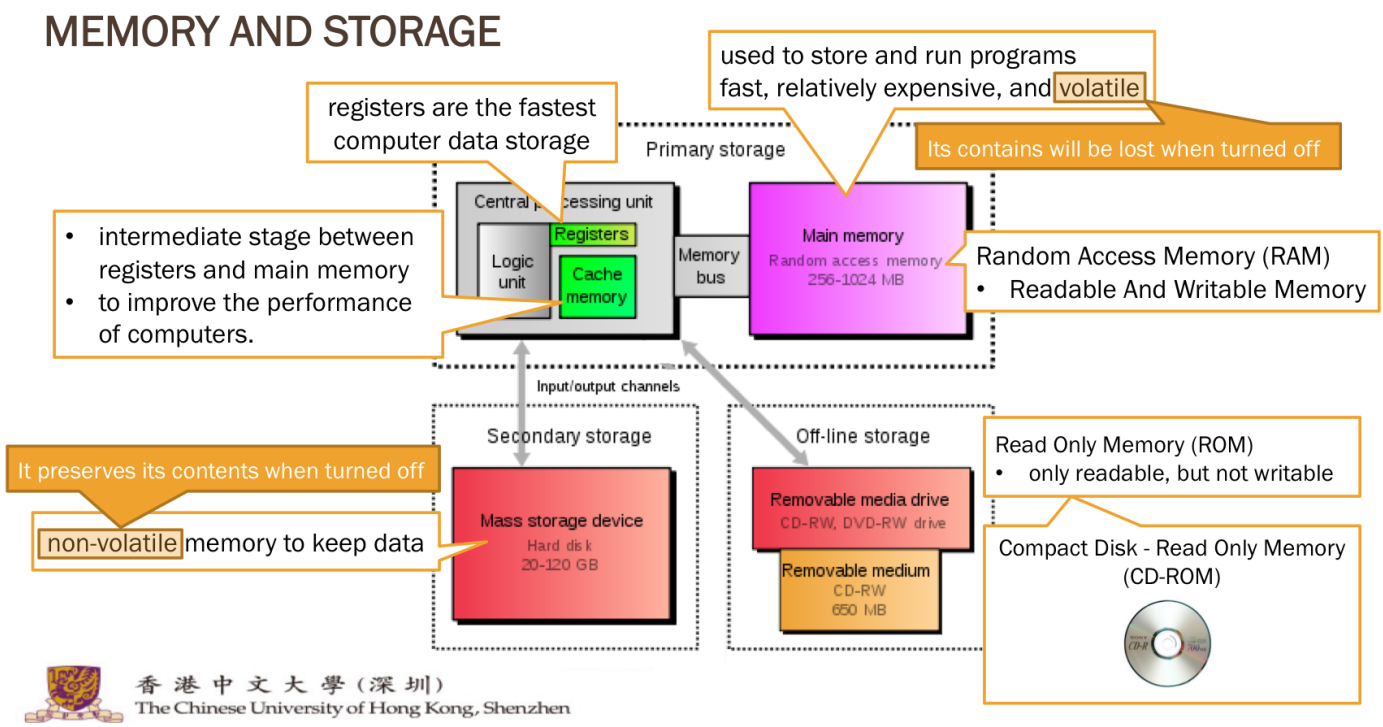
2° CU: used to fetch commands from the memory

ALU: contains the electric circuits which can execute commands

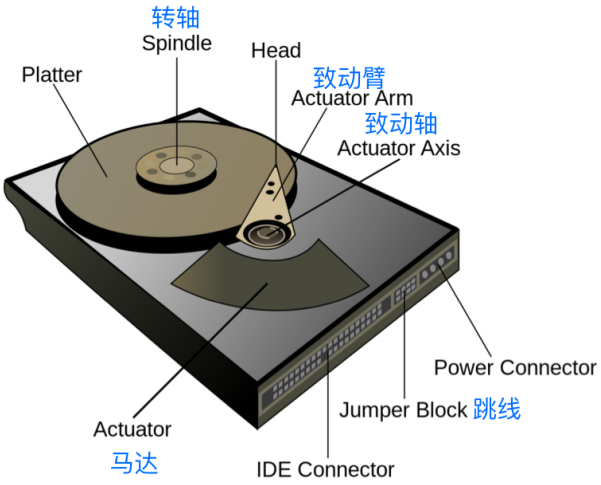
3° Processor manufacturer: Intel, AMD, ARM, etc.



2. Memory and storage



3. Hard disk



4. Input/Output devices

1° Input devices: mouse, keyboard, panel, touch screen, audio input, mind reading, etc.

2° Output devices: screen, audio output, etc.

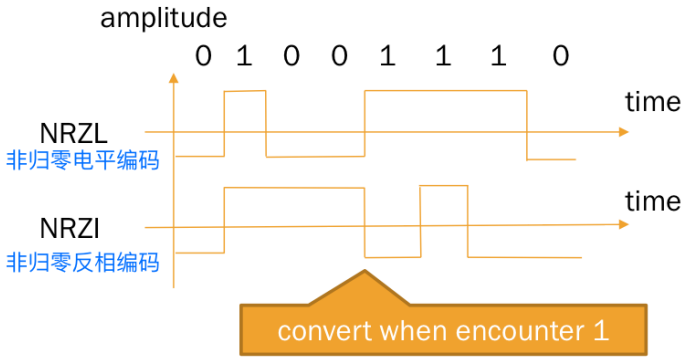
§3 Machine Language

1. Coding methods

1° The computer used nowadays can understand only binary number (i.e. 0 and 1)

2° Computer use voltage levels to represent 0 and 1

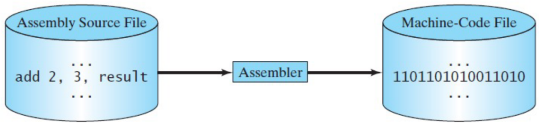
- Non Return to Zero Level Coding (NRZL)
- Non Return to Zero Inverted (NRZI)



2. Low level of language — Assembly (Hardware) Language

1° Hardware dependency

2° Assembly language is converted into executable machine code by a utility program referred to as an assembler.



3. High level languages

1° Designed to be easy for human beings to write and read. Resemble human languages in many ways.

2°

C LANGUAGE (1969 - 1973)

- C was developed by Dennis Ritchie between 1969 and 1973 at AT&T Bell Labs
- One of the early high-level programming language
  - Somewhere between assembly and other high level languages
- Provide powerful functionalities for *low level memory* manipulations
- Have the highest efficiency within high level languages
- Very widely used in low level applications, such as operating systems, embedded programming, super computers, etc

C++ LANGUAGE (1979)

- C++ was developed by Bjarne Stroustrup at Bell Labs since 1979
- Inherent major features of C
- An *object-oriented* programming language, supporting code reuse
- High efficiency and powerful in low level memory manipulation
- Still platform dependent

JAVA LANGUAGE (1995)

- Java was developed by James Gosling at Sun Microsystems and released in 1995
  - acquired by Oracle Corporation
- A new generation of general-purpose object-oriented programming language
- Platform independent,
  - Write Once, Run Anywhere (WORA)
- Java is one of the most popular programming languages currently in use

PYTHON (1991) friendly to programmers

- Developed by Guido van Rossum in 1989, and formally released in 1991
- An open source, object-oriented programming language
- Powerful Libraries
- Powerful interfaces to integrate other programming languages (C/C++, Java, and many other languages)
- Programming language of the year 2010

§4. Operating System

1. Operating system (OS) is a low level program which provides all basic services for managing and controlling a computer's activities

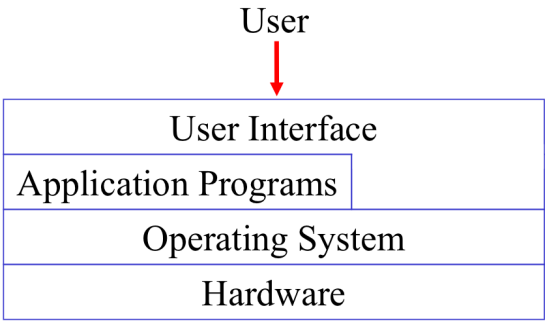
2. Main Functions

1° Controlling and monitoring system activities

2° Allocating and assigning system resources

3° Scheduling operations

* Applications are built based upon an OS



§5. Data Representation

1. Data Representation

We use positional notation to represent or encode numbers in a computer. We usually represent data using number systems

binary / decimal / octal / hexadecimal

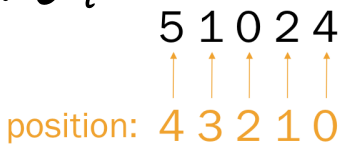
2. The Basic Idea of Positional Notation

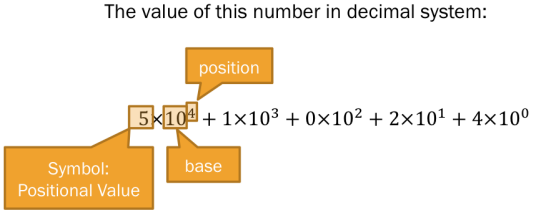
1° Each positional number system contains two elements, a base and a set of symbols

e.g. Decimal system

① base is 10

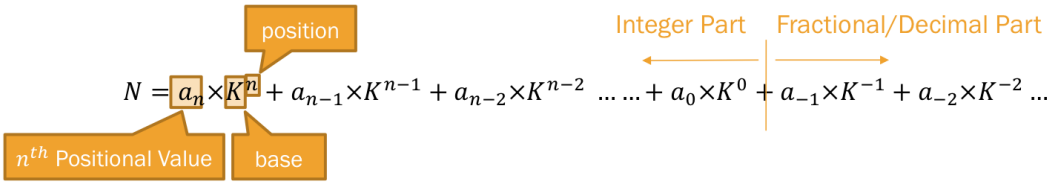
② symbols are {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}





3. Number System

Every number can be decomposed into the sum of a series of numbers.



- Decimal
  - $N = a_n \times 10^n + a_{n-1} \times 10^{n-1} + a_{n-2} \times 10^{n-2} \ldots\ldots + a_0 \times 10^0 + a_{-1} \times 10^{-1} + a_{-2} \times 10^{-2} \ldots$
- Binary
  - $N = a_n \times 2^n + a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} \ldots\ldots + a_0 \times 2^0 + a_{-1} \times 2^{-1} + a_{-2} \times 2^{-2} \ldots$
- Octal
  - $N = a_n \times 8^n + a_{n-1} \times 8^{n-1} + a_{n-2} \times 8^{n-2} \ldots\ldots + a_0 \times 8^0 + a_{-1} \times 8^{-1} + a_{-2} \times 8^{-2} \ldots$
- Hexadecimal
  - $N = a_n \times 16^n + a_{n-1} \times 16^{n-1} + a_{n-2} \times 16^{n-2} \ldots\ldots + a_0 \times 16^0 + a_{-1} \times 16^{-1} + a_{-2} \times 16^{-2} \ldots$

4. Binary System

$N = a_n \times 2^n + a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} \ldots\ldots + a_0 \times 2^0 + a_{-1} \times 2^{-1} + a_{-2} \times 2^{-2} \ldots$

1° The base is 2, the symbols = {0, 1}

2° The advantage of binary:

① Easy to implement physically

② Simple calculation rules

③ Easy to combine arithmetic and logic operations

$(0)_2=(0)_{10}$ $(1)_2=(1)_{10}$ $(10)_2=(2)_{10}$ $(11)_2=(3)_{10}$ $(100)_2=(4)_{10}$ $(101)_2=(5)_{10}$ $(110)_2=(6)_{10}$ $(111)_2=(7)_{10}$ $(1000)_2=(8)_{10}$

4. Hexadecimal System

$N = a_n \times 16^n + a_{n-1} \times 16^{n-1} + a_{n-2} \times 16^{n-2} \ldots\ldots + a_0 \times 16^0 + a_{-1} \times 16^{-1} + a_{-2} \times 16^{-2} \ldots$

1° The base is 16, the symbols = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

$(0)_{16}=(0)_{10}$ $(1)_{16}=(1)_{10}$ $(2)_{16}=(2)_{10}$ ⋮ $(9)_{16}=(9)_{10}$

$(A)_{16}=(10)_{10}$ $(B)_{16}=(11)_{10}$ $(C)_{16}=(12)_{10}$ $(D)_{16}=(13)_{10}$ $(E)_{16}=(14)_{10}$ $(F)_{16}=(15)_{10}$

$(10)_{16}=(16)_{10}$ $(11)_{16}=(17)_{10}$ $(12)_{16}=(18)_{10}$ ⋮ $(19)_{16}=(25)_{10}$

$(1A)_{16}=(26)_{10}$ $(1B)_{16}=(27)_{10}$ $(1C)_{16}=(28)_{10}$ $(1D)_{16}=(29)_{10}$ $(1E)_{16}=(30)_{10}$ $(1F)_{16}=(31)_{10}$

$(20)_{16}=(32)_{10}$ $(21)_{16}=(33)_{10}$ ⋮

2° The advantage of hexadecimal:

There is a "one-to-one" relationship between 4 digits binary number and 1 digit hexadecimal number.

$(0000)_2=(0)_{16}$ $(0001)_2=(1)_{16}$ $(0010)_2=(2)_{16}$ $(0011)_2=(3)_{16}$ $(0100)_2=(4)_{16}$ $(0101)_2=(5)_{16}$ $(0110)_2=(6)_{16}$ $(0111)_2=(7)_{16}$

$(1000)_2=(8)_{16}$ $(1001)_2=(9)_{16}$ $(1010)_2=(A)_{16}$ $(1011)_2=(B)_{16}$ $(1100)_2=(C)_{16}$ $(1101)_2=(D)_{16}$ $(1110)_2=(E)_{16}$ $(1111)_2=(F)_{16}$

$(00010000)_2=(10)_{16}$ $(00010001)_2=(11)_{16}$ $(00010010)_2=(12)_{16}$ $(00010011)_2=(13)_{16}$ $(00010100)_2=(14)_{16}$ $(00010101)_2=(15)_{16}$ $(00010110)_2=(16)_{16}$ $(00010111)_2=(17)_{16}$

$(00011000)_2=(18)_{16}$ $(00011001)_2=(19)_{16}$ $(00011010)_2=(1A)_{16}$ $(00011011)_2=(1B)_{16}$ $(00011100)_2=(1C)_{16}$ $(00011101)_2=(1D)_{16}$ $(00011110)_2=(1E)_{16}$ $(00011111)_2=(1F)_{16}$

4. Octal System

$$N = a_n \times 8^n + a_{n-1} \times 8^{n-1} + a_{n-2} \times 8^{n-2} \ \ldots\ldots + a_0 \times 8^0 + a_{-1} \times 8^{-1} + a_{-2} \times 8^{-2} \ldots$$

1° The base is 8, the symbols = {0, 1, 2, 3, 4, 5, 6, 7}

$(0)_8=(0)_{10}$	$(10)_8=(8)_{10}$	$(20)_8=(16)_{10}$
$(1)_8=(1)_{10}$	$(11)_8=(9)_{10}$	$(21)_8=(17)_{10}$
$(2)_8=(2)_{10}$	$(12)_8=(10)_{10}$	$(22)_8=(18)_{10}$
$(3)_8=(3)_{10}$	$(13)_8=(11)_{10}$	$(23)_8=(19)_{10}$
$(4)_8=(4)_{10}$	$(14)_8=(12)_{10}$	$(24)_8=(20)_{10}$
$(5)_8=(5)_{10}$	$(15)_8=(13)_{10}$	$(25)_8=(21)_{10}$
$(6)_8=(6)_{10}$	$(16)_8=(14)_{10}$	$(26)_8=(22)_{10}$
$(7)_8=(7)_{10}$	$(17)_8=(15)_{10}$	$(27)_8=(23)_{10}$

2° The advantage of octal:

There is a "one-to-one" relationship between 3 digits binary number and 1 digit octal number.

$(000)_2=(0)_8$	$(001000)_2=(10)_8$	$(010000)_2=(20)_8$
$(001)_2=(1)_8$	$(001001)_2=(11)_8$	$(010001)_2=(21)_8$
$(010)_2=(2)_8$	$(001010)_2=(12)_8$	$(010010)_2=(22)_8$
$(011)_2=(3)_8$	$(001011)_2=(13)_8$	$(010011)_2=(23)_8$
$(100)_2=(4)_8$	$(001100)_2=(14)_8$	$(010100)_2=(24)_8$
$(101)_2=(5)_8$	$(001101)_2=(15)_8$	$(010101)_2=(25)_8$
$(110)_2=(6)_8$	$(001110)_2=(16)_8$	$(010110)_2=(26)_8$
$(111)_2=(7)_8$	$(001111)_2=(17)_8$	$(010111)_2=(27)_8$

§6 Data Conversion

1. Converting xxx Number into Decimal Number

1° Binary number

Example: $(10110.111)_2$

$$= (1\times2^4+0\times2^3+1\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2})_{10}$$

$$= (22.75)_{10}$$

2° Octal number

Example: $(35.7)_8$

$$= (3\times8^1+5\times8^0+7\times8^{-1})_{10}$$

$$= (29.875)_{10}$$

3° Hexadecimal number

Example: $(A7D.E)_{16}$

$= (10\times16^2+7\times16^1+13\times16^0+14\times16^{-1})_{10}$

$= (2685.875)_{10}$

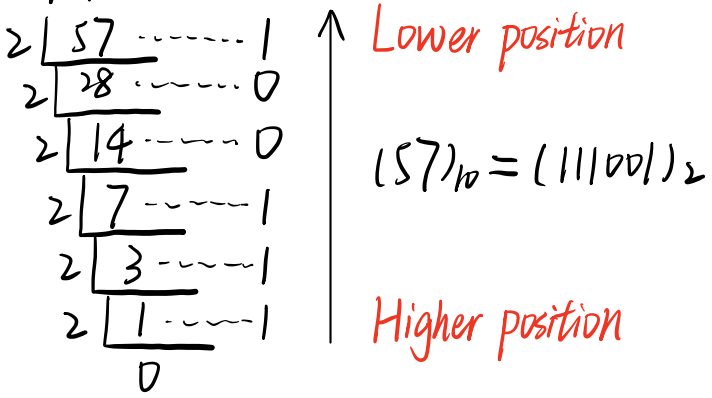
4° Other number

We just need to change the corresponding base

2. Converting Decimal Number into Binary Number

1° Integer part

Example: $(57)_{10}=(?)_2$



2° Fraction part

Example: $(0.875)_{10}=(?)_2$

$0.875\times2=1.75$ Integer part: 1 — Higher position

$0.75\times2=1.5$ Integer part: 1

$0.5\times2=1$ Integer part: 1 — Lower position

So, $(0.875)_{10}=(0.111)_2$

3. Converting xxx Number into Binary Number

1° Octal number

① Convert each octal digit into binary number of three digits

② Keep the digit order unchanged

Example: $(16.327)_8$

$=(001\,110.011\,010\,111)_2$

$=(1110.011010111)_2$

2° Hexadecimal number

① Convert each hexadecimal digit into binary number of four digits

② Keep the digit order unchanged

Example: $(AD.7F)_{16}$

$=(1010\ 1101.0111\ 1111)_2$

$=(10101101.01111111)_2$

3. Converting Binary Number into xxx Number

1° Octal number

① Starting from lower positions, convert every three digits of the integer part into a octal digit

② Starting from higher positions, convert every three digits of the fractional part into a octal digit

③ When there is not enough higher (lower) positions in the integer (fractional) part, fill with 0

④ Keep the digit order unchanged

Example: $(1101101.011)_2$

$=(001\ 101\ 101.011)_2$

$=(155.3)_8$

2° Hexadecimal number:

① Starting from lower positions, convert every four digits of the integer part into a hexadecimal digit

② Starting from higher positions, convert every four digits of the fractional part into a hexadecimal digit

③ When there is not enough higher (lower) positions in the

integer (fractional) part, fill with 0

④ Keep the digit order unchanged

Example: $(11101.01)_2$

$= (0001\ 1101.0100)_2$

$= (1D.4)_{16}$

§7 The Units of Information (Data)

		Actual Bytes	
Bit	either 0 or 1	the smallest information unit in computer	
Byte	8 bits	1 byte	1 byte
KB	1024 B	2^{10} bytes	1 024 bytes
MB	1024 KB	2^{20} bytes	1 048 576 bytes
GB	1024 MB	2^{30} bytes	1 073 741 824 bytes
TB	1024 GB	2^{40} bytes	1 099 511 627 776 bytes

§8 Memory And Addressing

1. A computer's memory consists of an ordered sequence of bytes for storing data.
2. Every location in the memory has a unique address
3. The key difference between high and low level programming languages is whether programmer has to deal with memory addressing directly.

§9. Character Set

1. ASCII Character set

(American Standard Code for Information Interchange)

1° Originally used seven bits to represent each character.

→ allow for 128 unique characters.

2° Later, all eight bits were used.

→ allow for 256 characters.

Left Digit(s) \ Right Digit	ASCII 0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1	LF	VT	FF	CR	SO	SI	DLE	DC1	DC2	DC3
2	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3	RS	US	□	!	"	#	$	%	&	'
4	(	)	*	+	,	−	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	DEL		

ASCII Table

2. Unicode Character Set

uses 16 bits per character

→ represent 2^{16} (65536) characters