



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Introduction to Computer Engineering: Programming and Applications

Lecture 1 Introduction

Prof. Junhua Zhao

School of Science and Engineering

Learning Objectives

- This course introduces the basics of computer programming using **Python**
- Students will learn the basic elements of **modern computer systems, key programming concepts, problem solving and basic algorithm design**
- Students will also learn the tricks and skills of applying computer programming to solve practical problems (math, physics, chemistry and engineering)

Key Topics

- Introduction to modern computers
- Preliminary knowledge for computer programming
- Basic introduction to Python language
- Data types and operators in Python language
- Input/output
- Flow control and loop
- Function
- List
- Basic data structure
- Introduction to object oriented programming
- Applications of computer programming in science and engineering.
- Programming assisted by large language models and the use of prompt engineering.

Assessment

Assignments × 4	10% × 4
Mid-term quiz	20%
Final exam	40%

Course Materials

- All lecture notes and sample code used in classes will be provided to students via **Blackboard**
- Recommended readings
 - Online resources: <https://www.python.org/doc/>
 - **Automate the boring stuff with Python**, Al Sweigart
 - **Learning Python the hard way**, Zed A. Shaw
 - **Beginning Python, From Novice to Professional**, Magnus Lie Hetland

Course Components

Activity	Hours/week
Lecture	50 minutes × 3
Tutorial	50 minutes × 1

Indicative Teaching Plans

Week	Content/ topic/ activity
1	Introduction to modern computers; Preliminary knowledge for computer programming;
2	Basic introduction to Python language; Data types and operators in Python language; Input/output;
3	Flow control and loop;
4	Function;
5	List;
6	Introduction to object oriented programming, part I
7	Review for mid-term quiz;
8	Introduction to object oriented programming, part II
9	Data Structure, part I;
10	Data Structure, part II;
11	LLM assisted programming and prompt engineering;
12	Programming applications in science and engineering, part I;
13	Programming applications in science and engineering, part II;
14	Review for final exam;

Why learn programming?

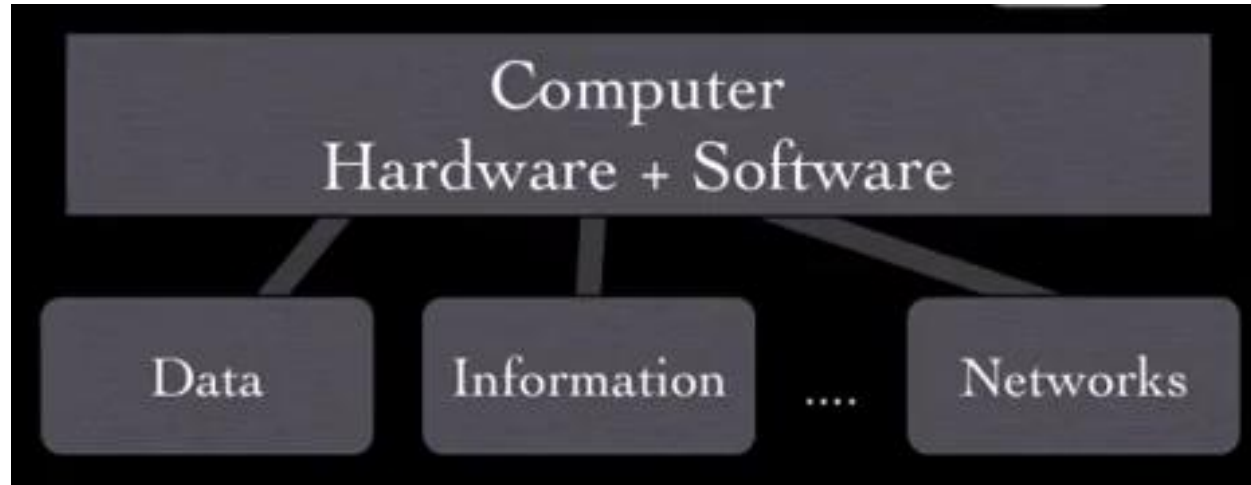
- Computer is built to help people **solve problems**
- Computer **does not** understand what we say
- We need to communicate with computers using their languages (**computer programming language**)
- Assembly, C, C++, Java and **Python**



User



Interface

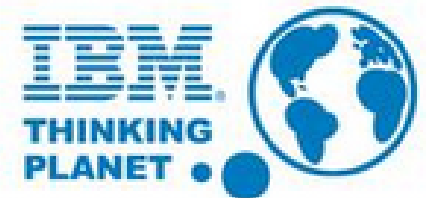
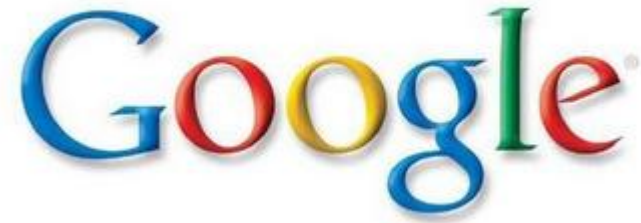


Programmer

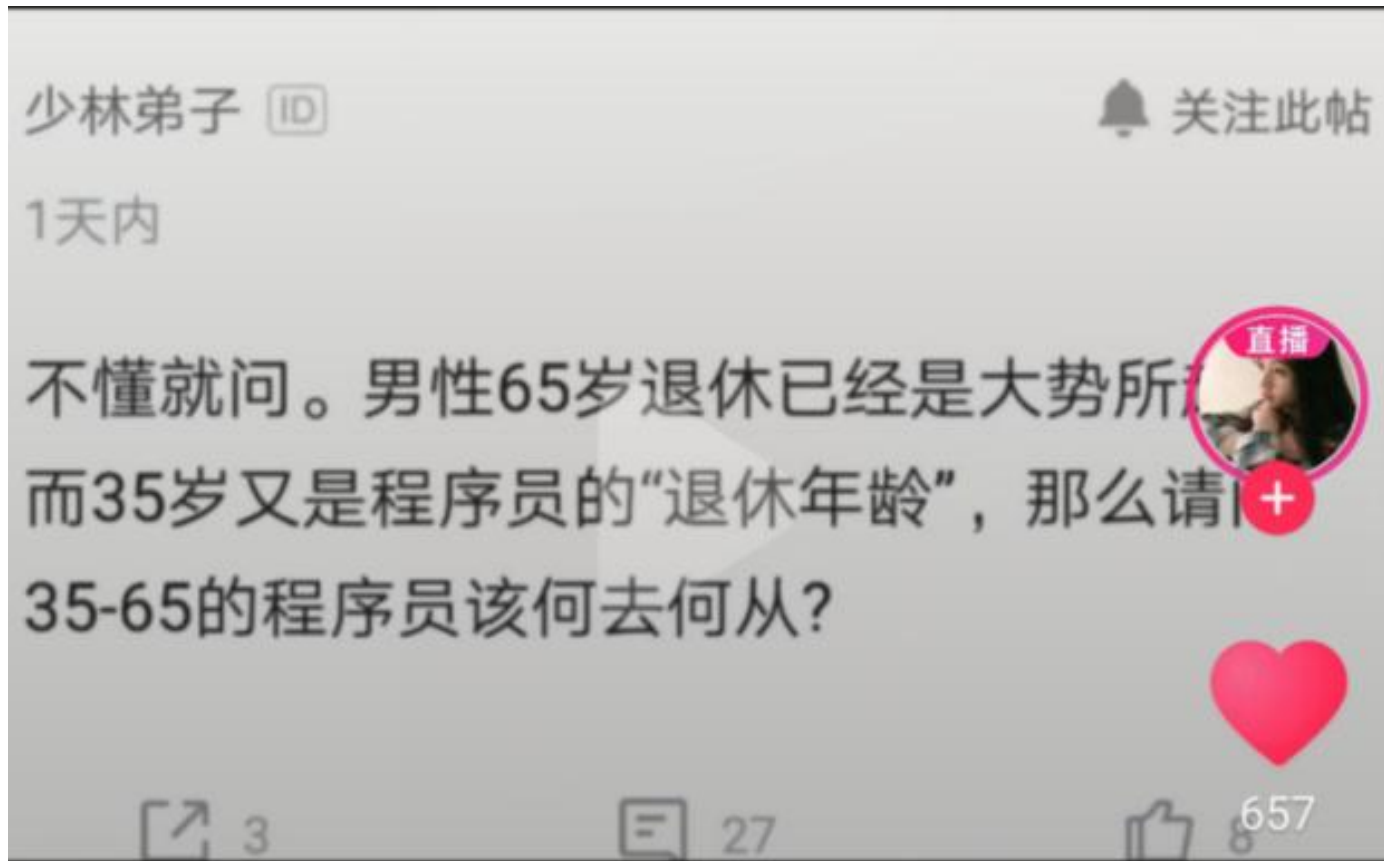
- Programmers solve problems like data, information, networks on behalf of users

Programmer

- **Professional programmer** writes computer programs and develops software
- A junior programmer gets a salary of 10-20k RMB in an INTERNET company like Tencent
- A programmer can earn up to **500k – 1m USD** in Google!!
- Software and INTERNET are huge **industries**.



The Retirement of Programmer



- It is well-known that programmers in China retires at around **35**
- The official retirement age in China is **65!!**

The Retirement of Programmer



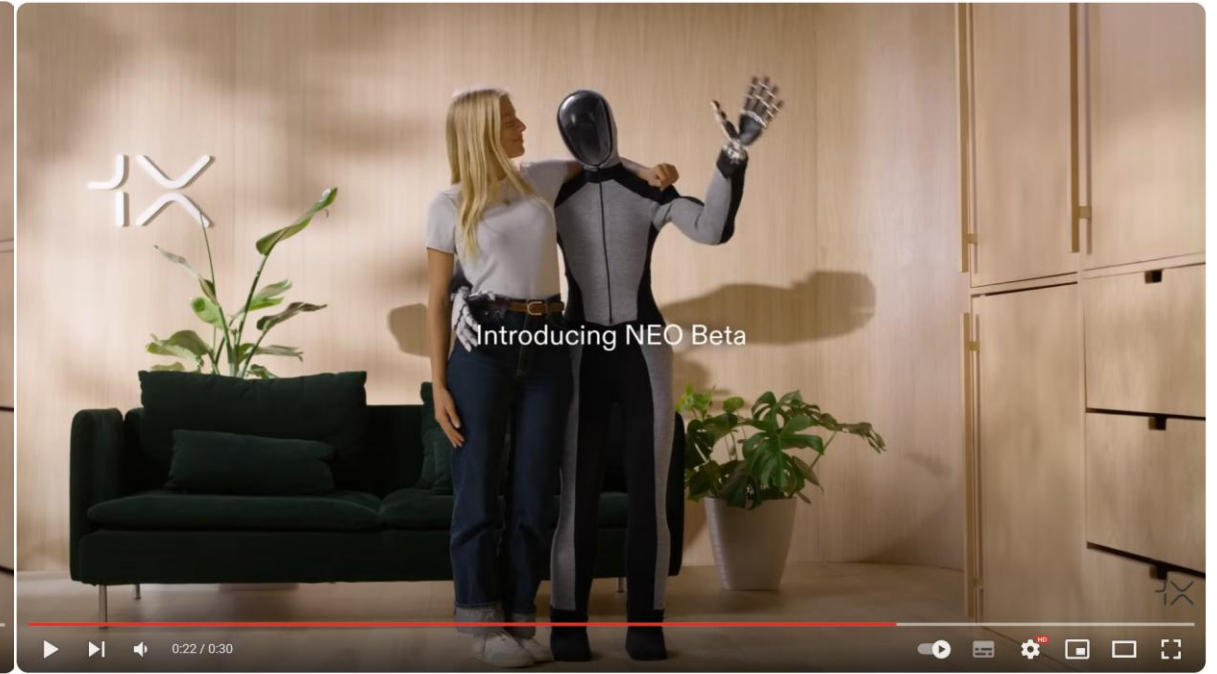


Larry Ellison, CEO of Oracle Corporation

Odds are the person on your left is going to be a loser. The person on your right, meanwhile, will also be a loser. And you, in the middle? What can you expect? Loser.

ORACLE®

The Future: Robot and Embodied Intelligence



- The possibility of being replaced by AI and robots exists for both physical and mental labor jobs in the next 10-20 years.

Why be a programmer?

- Programming is pervasive in your life, even if you are **NOT** in the IT industry
 - Scientists – Math, Physics, Chemistry, etc
 - Electrical/electronic engineer – control program
 - Economist – mathematical modeling
 - Salesman – analyzing sales data
 - ...

What is Code? Software? Program?

- A sequence of instructions
- Computers take the instructions and execute them
- It is a little piece of our intelligence in the computer
- Intelligence which is **re-usable**

Programs for human

- Right hand around your head
- Left hand around your belly
- Straighten out right hand
- Left hand rests on your hip
- One step to the right while straighten out left hand
- Keep down and swing your ham
-

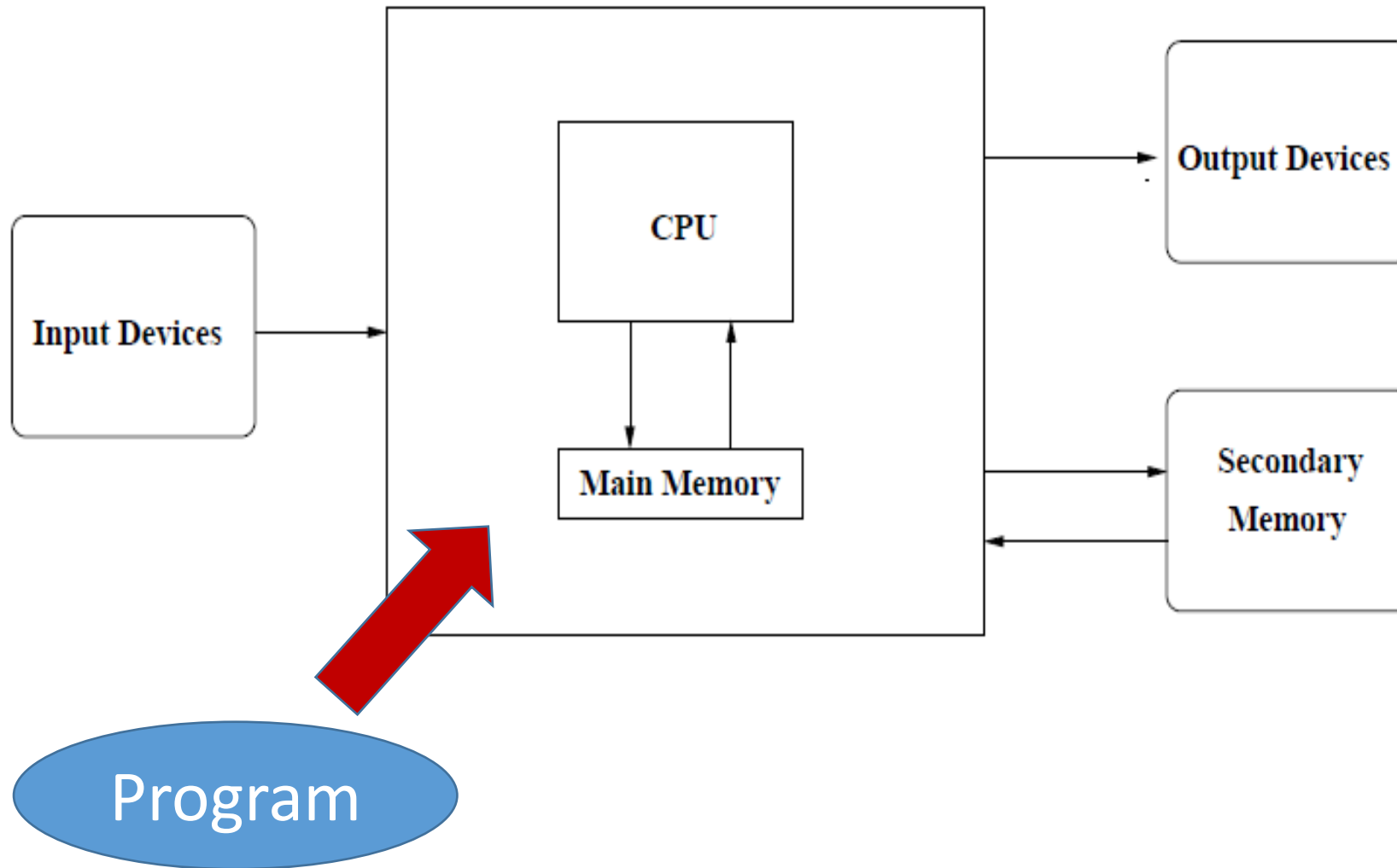


http://v.youku.com/v_show/id_XNzM1MTgyMTA4.html?from=y1.2-1-98.3.2-2.1-1-1-0

Computers are good at following instructions

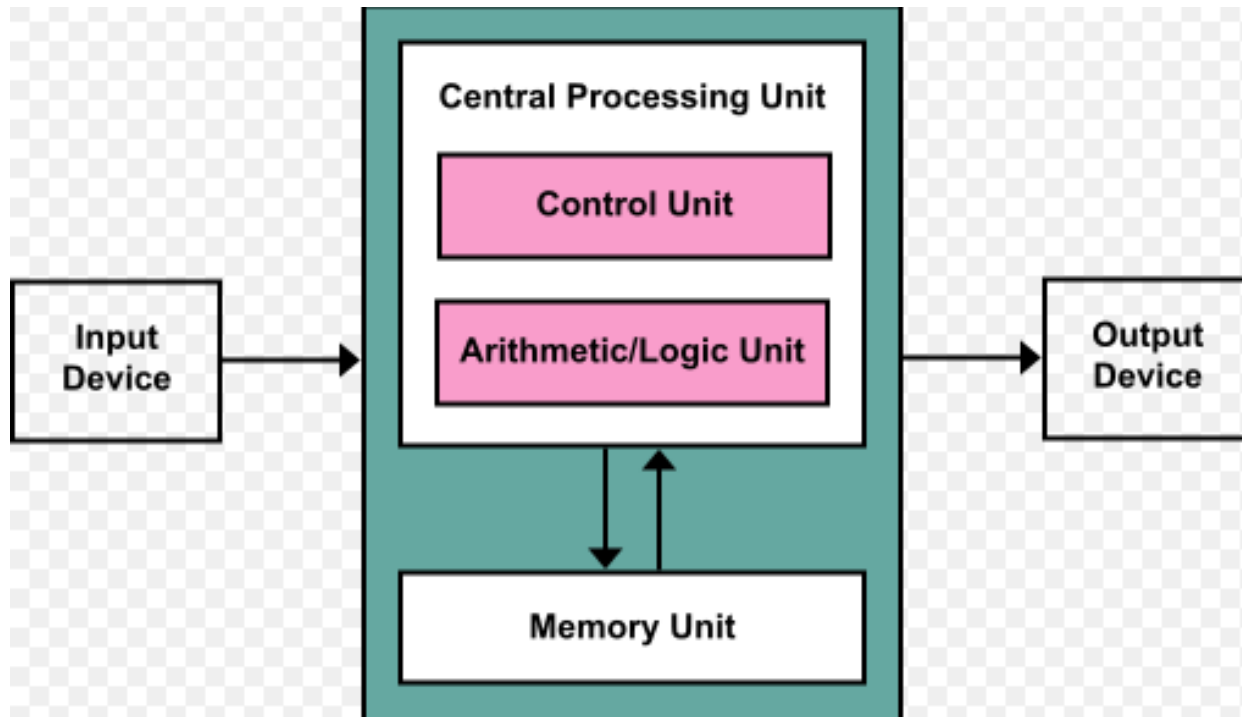
- Humans can easily make mistakes when following a set of instructions
- On the contrary, computers (usually) **won't make mistakes**, regardless of they are given 10 or 10 billion instructions !!

Computer Hardware



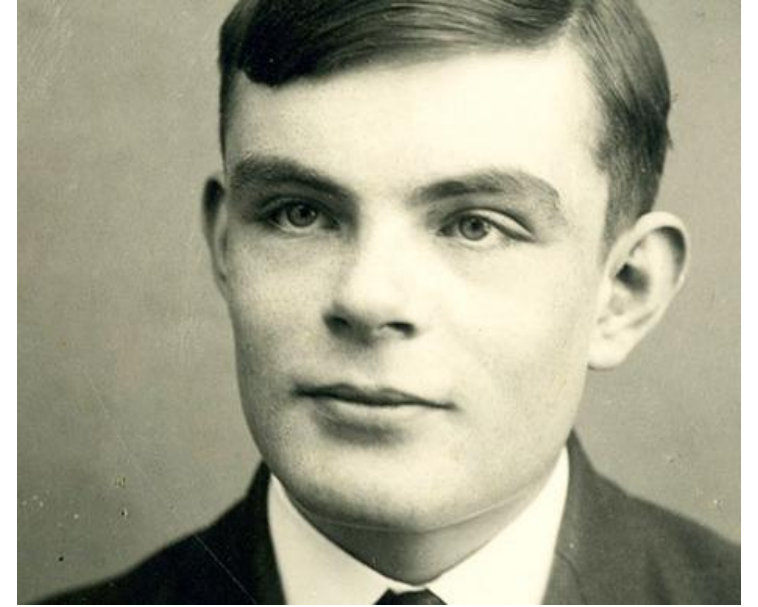
Von Neumann Architecture

- The modern computer architecture is proposed by **John Von Neumann**



The theoretical foundation of computer science

- The theoretical foundation of computer science are built by Alan Turing
- Father of theoretical computer science and artificial intelligence
- Computability theory and Turing test



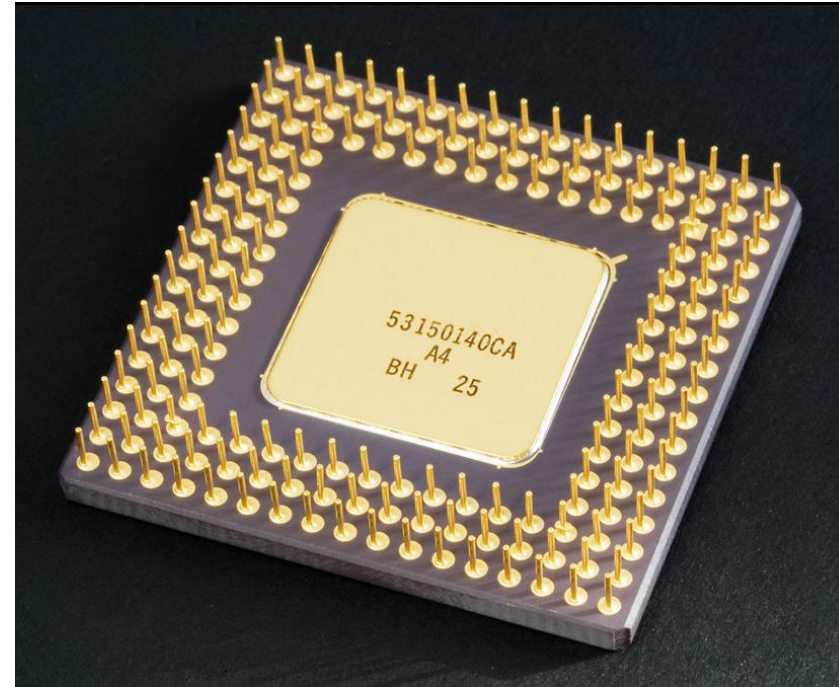
Key components in a computer

- **Central processing unit (CPU)**: execute your program. Similar to human brain, very fast but not that smart
- **Input device**: take inputs from users or other devices
- **Output device**: output information to users or other devices
- **Main memory**: store data, fast and temporary storage
- **Secondary memory**: slower but large size, permanent storage

Central Processing Unit

- A processor contains two units, a control unit (CU) and an arithmetic/logic unit (ALU)
- **CU** is used to fetch commands from the memory
- **ALU** contains the electric circuits which can execute commands

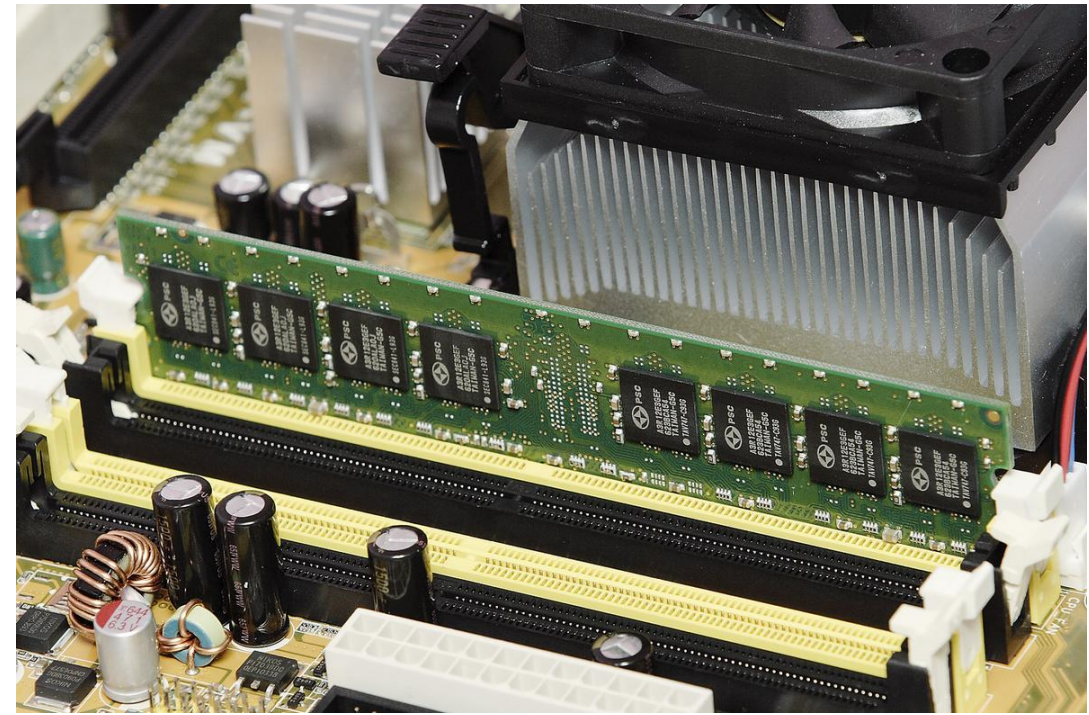
Central Processing Unit



- Processor manufacturer: **Intel, AMD, ARM**, etc

Memory/Storage

- High speed cache
- Internal RAM
- Internal ROM
- External RAM
- Flash
- Hard disk



Input/output devices

- **Input devices:** mouse, keyboard, panel, touch screen, audio input, mind reading, etc
- **Output devices:** screen, audio output, etc



How the hard disk works

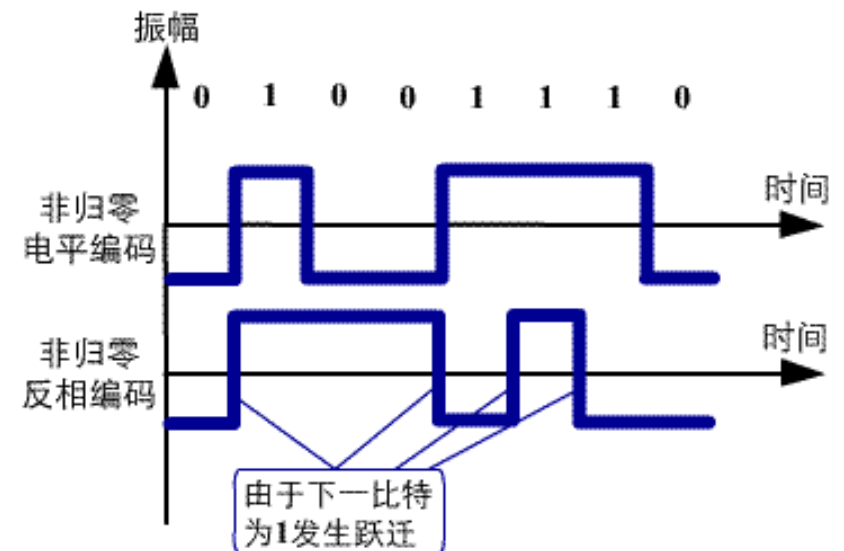


http://v.youku.com/v_show/id_XNjA4NzMxNDk2.html?from=s1.8-1-1.2

What can a computer actually understand?

- The computers used nowadays can understand only binary number (i.e. 0 and 1)
- Computers use voltage levels to represent 0 and 1
- NRZL and NRZI coding
- The instructions expressed in binary code is called **machine language**

0 0 0 1	numerical value 2^0
0 0 1 0	numerical value 2^1
0 1 0 0	numerical value 2^2
1 0 0 0	numerical value 2^3

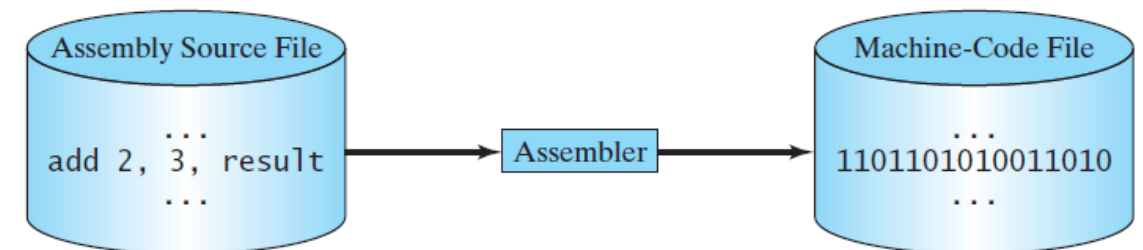


Low level language – Assembly Language

- An **assembly language** is a low-level programming language, in which there is a very strong (generally one-to-one) correspondence between the language and machine code instructions.
- Each assembly language is specific to a particular computer architecture
- Assembly language is converted into executable machine code by a utility program referred to as an **assembler**

```
*****
* FUNCTION: INHEX - INPUT HEX DIGIT
* INPUT: none
* OUTPUT: Digit in acc A
* CALLS: INCH
* DESTROYS: acc A
* Returns to monitor if not HEX input
```

C01E 8D F0	INHEX	BSR	INCH	GET A CHAR
C020 81 30		CMP A	#'0	ZERO
C022 2B 11		BMI	HEXERR	NOT HEX
C024 81 39		CMP A	#'9	NINE
C026 2F 0A		BLE	HEXRTS	GOOD HEX
C028 81 41		CMP A	#'A	
C02A 2B 09		BMI	HEXERR	NOT HEX
C02C 81 46		CMP A	#'F	
C02E 2E 05		BGT	HEXERR	
C030 80 07		SUB A	#7	FIX A-F
C032 84 0F	HEXRTS	AND A	#\$0F	CONVERT ASCII TO DIGIT
C034 39		RTS		
C035 7E C0 AF	HEXERR	JMP	CTRL	RETURN TO CONTROL LOOP



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** (which has since been acquired by Oracle Corporation) and released in 1995
- 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)

- 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

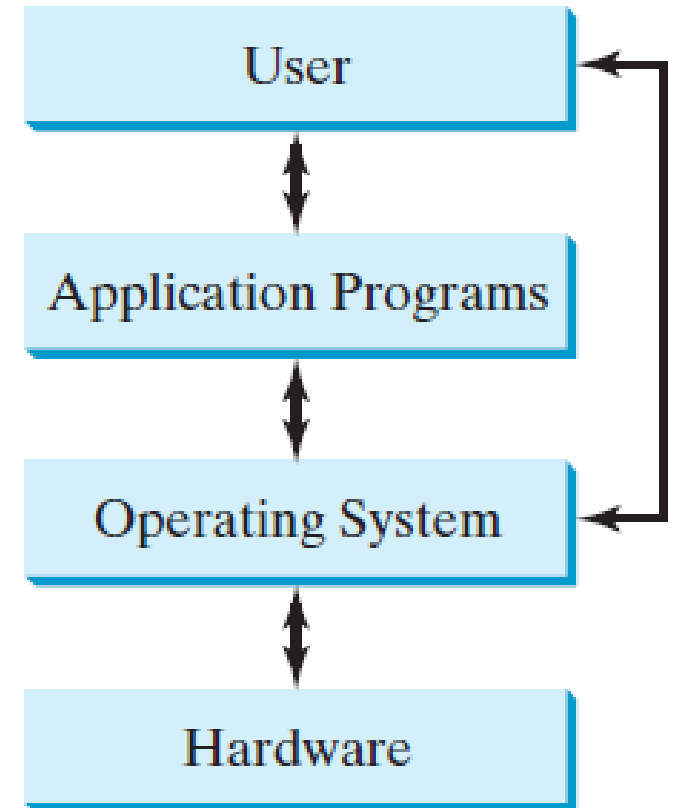
Aug 2023	Aug 2022	Change	Programming Language		Ratings	Change
1	1			Python	13.33%	-2.30%
2	2			C	11.41%	-3.35%
3	4	▲		C++	10.63%	+0.49%
4	3	▼		Java	10.33%	-2.14%
5	5			C#	7.04%	+1.64%
6	8	▲		JavaScript	3.29%	+0.89%
7	6	▼		Visual Basic	2.63%	-2.26%
8	9	▲		SQL	1.53%	-0.14%
9	7	▼		Assembly language	1.34%	-1.41%
10	10			PHP	1.27%	-0.09%
11	21	▲▲		Scratch	1.22%	+0.63%
12	15	▲		Go	1.16%	+0.20%
13	17	▲▲		MATLAB	1.05%	+0.17%
14	18	▲▲		Fortran	1.03%	+0.24%
15	31	▲▲		COBOL	0.96%	+0.59%
16	16			R	0.92%	+0.01%
17	19	▲		Ruby	0.91%	+0.18%
18	11	▼▼		Swift	0.90%	-0.35%
19	22	▲		Rust	0.89%	+0.32%

Language efficiency v.s. development efficiency

- High level languages **cannot be executed directly**
- High level languages **must be converted** into low level languages first
- Lower level languages have **higher language efficiency** (they are faster to run on a computer)
- Higher level languages have **higher development efficiency** (it is easier to write programs in these languages)

Operating Systems

- The operating system (OS) is a **low level program**, which provides all **basic services** for managing and controlling a computer's activities
- Applications are programs which are built based upon an OS
- **Main functions** of an OS:
 - ✓ Controlling and monitoring system activities
 - ✓ Allocating and assigning system resources
 - ✓ Scheduling operations
- Popular OS: Windows, Mac OS, Linux, iOS, Android...



Data Representation and Conversion

- We use **positional notation** (进位记数法) to represent or encode numbers in a computer
- Data are stored essentially as **binary numbers** in a computer
- In practice, we usually represent data using either **binary** (二进制), **decimal** (十进制), **octal** (八进制) or **hexadecimal** (十六进制) number systems
- We may need to **convert data** between different number systems

The basic idea of positional notation

- Each positional number system contains two elements, a **base (基数)** and **a set of symbols**
- Using the decimal system (十进制系统) as an example, its **base is 10**, and the **symbols are {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}**
- When a number “hits” 9, the next number will not be a different symbol, but a “1” followed by a “0” (**逢十进一**)

Decimal number system

- In the decimal number system, the **base** is 10, the **symbols** include 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Every number can be decomposed into the **sum** of a series of numbers, each is represented by a **positional value** times a **weight**
- $$N = a_n \times 10^n + a_{n-1} \times 10^{n-1} + a_{n-2} \times 10^{n-2} \dots \dots + a_0 \times 10^0 + a_{-1} \times 10^{-1} + a_{-2} \times 10^{-2} \dots$$
- a_n is the positional value (ranging from 0 to 9), while 10^n represents the weight

Binary number system

- In the binary system, the **base** is 2, we use **only two symbols** 0 and 1
- “10” is used when we hit **2 (逢二进一)**
- $$N = a_n \times 2^n + a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} \dots \dots + a_0 \times 2^0 + a_{-1} \times 2^{-1} + a_{-2} \times 2^{-2} \dots$$
- a_n is the positional value (ranging from 0 to 1), while 2^n represents the weight

Why use binary number?

- Easy to implement physically
- Simple calculation rules
- Easy to combine arithmetic and logic operations

Hexadecimal number system

- In the hexadecimal system, the **base** is 16, we use **16 symbols** {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f}
- “10” is used when we hit **16 (逢十六进一)**
- $$N = a_n \times 16^n + a_{n-1} \times 16^{n-1} + a_{n-2} \times 16^{n-2} \dots \dots + a_0 \times 16^0 + a_{-1} \times 16^{-1} + a_{-2} \times 16^{-2} \dots$$
- a_n is the positional value (ranging from 0 to 15), while 16^n represents the weight

Octal number system



Converting binary number into decimal number

Example $(1101.01)_2$
 $= (1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2})_{10}$
 $= (13.25)_{10}$

Practice $(10110.11)_2 = (?)_{10}$

Converting binary number into decimal number

Answer

(10110.11)

$$=(1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2})_{10} = (22.75)_{10}.$$

Converting octal number into decimal number

Example $(24.67)_8 = (2 \times 8^1 + 4 \times 8^0 + 6 \times 8^{-1} + 7 \times 8^{-2})_{10}$
 $= (20.859375)_{10}$

Practice $(35.7)_8 = (?)_{10}$

Converting octal number into decimal number

Answer $(35.7)_8 = (3 \times 8^1 + 5 \times 8^0 + 7 \times 8^{-1})_{10}$
 $= (29.875)_{10}$

Converting hexadecimal number into decimal number

Example $(2AB.C)_{16}$

$$=(2 \times 16^2 + 10 \times 16^1 + 11 \times 16^0 + 12 \times 16^{-1})_{10}$$
$$=(683.75)_{10}$$

Practice $(A7D.E)_{16} = (?)_{10}$

Converting hexadecimal number into decimal number

Answer

$$\begin{aligned}(A7D.E)_{16} &= (10 \times 16^2 + 7 \times 16^1 + 13 \times 16^0 + 14 \times 16^{-1})_{10} \\ &= (2685.875)_{10}\end{aligned}$$

Converting other number system into decimal system

- Other number system can also be converted into decimal system in a similar way
- We just need to change the corresponding base

Converting decimal integer into binary integer

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

2	57	1	<div>Lower position ↑ $(57)_{10} = (111001)_2$ Higher position</div>
2	28	0	
2	14	0	
2	7	1	
2	3	1	
2	1	1	

Converting decimal fraction into binary fraction

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

$0.875 \times 2 = 1.75$	Integer part: 1	<div>Higher position</div> <div>↓</div> <div>Lower position</div>
$0.75 \times 2 = 1.5$	Integer part: 1	
$0.5 \times 2 = 1$	Integer part: 1	

Answer: $(0.875)_{10} = (0.111)_2$

Practice: $(0.6875)_{10} = (?)_2$

Converting decimal fraction into binary fraction

Answer:

$$0.6875 \times 2 = 1.375 \quad \text{Integer part: 1}$$

$$0.375 \times 2 = 0.75 \quad \text{Integer part: 0}$$

$$0.75 \times 2 = 1.5 \quad \text{Integer part: 1}$$

$$0.5 \times 2 = 1 \quad \text{Integer part: 1}$$

Higher position



Lower position

$$\text{So, } (0.6875)_{10} = (0.1011)_2$$

Converting decimal number into binary number

- For a decimal number that has both integer and fractional parts
- Convert the integer and fractional parts **separately**
- **Example:** $(215.675)_{10} = (?)_2$

Converting decimal number into binary number

Answer:

$$(215)_{10} = (11010111)_2$$

$$(0.675)_{10} = (0.1011)_2$$

$$(215.675)_{10} = (11010111.1011)_2$$

The one-to-one relationship between binary and octal numbers

There is a “one-to-one” (一一对应) relationship between three digits binary number and one digit octal number

$$(0)_8 = (000)_2$$

$$(1)_8 = (001)_2$$

$$(2)_8 = (010)_2$$

$$(3)_8 = (011)_2$$

$$(4)_8 = (100)_2$$

$$(5)_8 = (101)_2$$

$$(6)_8 = (110)_2$$

$$(7)_8 = (111)_2$$

Converting octal number into binary number

- Convert **each octal digit** into binary number of **three digits**
- Keep the digit order **unchanged**
- **Example:** $(0.754)_8 = (?)_2$

$$\begin{array}{rcl} (0.754)_8 & = & (\underline{000}.\underline{111} \ \underline{101} \ \underline{100})_2 \\ & = & \underline{(0.1111011)}_2 \end{array}$$

- **Practice:** $(16.327)_8 = (?)_2$

Converting octal number into binary number

Answer:

$$\begin{aligned} & (16.327)_8 \\ &= (\underline{001\ 110}.\underline{011}\ \underline{010}\ \underline{111})_2 \\ &= (1110.011010111)_2 \end{aligned}$$

Converting hexadecimal number into binary number

- Convert **each hexadecimal digit** into binary number of **four digits**
- Keep the digit order **unchanged**

- **Example:** $(4C.2E)_{16} = (?)_2$

$$\begin{aligned} & (4C.2E)_{16} \\ &= (\underline{0100} \ \underline{1100}.\underline{0010} \ \underline{1110})_2 \\ &= (1001100.0010111)_2 \end{aligned}$$

- **Practice:** $(AD.7F)_{16} = (?)_2$

Converting hexadecimal number into binary number

Answer:

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

$$= (\underline{1010} \ \underline{1101} . \underline{0111} \ \underline{1111})_2$$

$$= (10101101.01111111)_2$$

Converting binary number into octal number

- Starting from lower positions, convert every **three digits of the integer part** into a octal digit
- When there is not enough **higher positions in the integer part**, fill with 0
- Starting from higher positions, convert every **three digits of the fractional part** into a octal digit
- When there is not enough **lower positions in the fractional part**, fill with 0
- Keep the digit order **unchanged**

Converting binary number into octal number

Example:

$$\begin{aligned}(0.10111)_2 &= (\underline{000}.\underline{101}\underline{110})_2 = (0.56)_8 \\ (11101.01)_2 &= (\underline{011}\underline{101}.\underline{010})_2 = (35.2)_8\end{aligned}$$

Practice:

$$(1101101.011)_2$$

Converting binary number into octal number

Answer:

$$\begin{aligned} (1101101.011)_2 &= (\underline{001} \ \underline{101} \ \underline{101} . \underline{011})_2 \\ &= (155.3)_8 \end{aligned}$$

Converting binary number into hexadecimal number

- Starting from lower positions, convert every **four digits of the integer part** into a octal digit
- When there is not enough **higher positions in the integer part**, fill with 0
- Starting from higher positions, convert every **four digits of the fractional part** into a octal digit
- When there is not enough **lower positions in the fractional part**, fill with 0
- Keep the digit order **unchanged**

Converting binary number into hexadecimal number

Example:

$$\begin{aligned} (11101.01)_2 &= (\underline{0001} \ \underline{1101}. \ \underline{0100})_2 \\ &= (1D.4)_{16} \end{aligned}$$

The units of information (data)

- Bit (比特/位): a binary digit which takes either 0 or 1
- Bit is the smallest information unit in computer programming
- Byte (字节): 1 byte = 8 bits, every English character is represented by 1 byte
- KB (千字节): $1 \text{ KB} = 2^{10} \text{ B} = 1024 \text{ B}$
- MB (兆字节): $1 \text{ MB} = 2^{20} \text{ B} = 1024 \text{ KB}$
- GB (千兆字节): $1 \text{ GB} = 2^{30} \text{ B} = 1024 \text{ MB}$
- TB (兆兆字节): $1 \text{ TB} = 2^{40} \text{ B} = 1024 \text{ GB}$

Memory and addressing

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

Memory address		Memory content	
.	↓	.	
.		.	
.		.	
2000		01000011	Encoding for character 'C'
2001		01110010	Encoding for character 'r'
2002		01100101	Encoding for character 'e'
2003		01110111	Encoding for character 'w'
2004		00000011	Encoding for number 3
.		.	