

Introduction to Object-Oriented Programming

COMP2011: Introduction

Dr. Cindy Li
Dr. Brian Mak
Dr. Dit-Yan Yeung

Department of Computer Science & Engineering
The Hong Kong University of Science and Technology
Hong Kong SAR, China

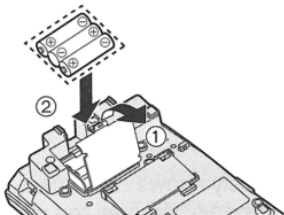


Course Objectives

- To learn how to **solve problems** by writing **computer programs**.
- To learn how to **design** a computer program.
- To learn how to program in **C++**.
- To learn how to **debug** a computer program.
- To learn **object-oriented programming**.
- To prepare you for COMP2012 (OOP & Data Structures), etc.

Question: *computer science = programming?*

Installing the Batteries

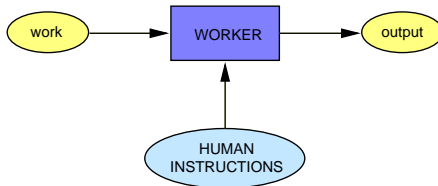


- 1 Press down in the direction of the arrow and open the cover (①).
- 2 Install the batteries in the proper order as shown (②), matching the correct polarity.
- 3 Close the battery cover.

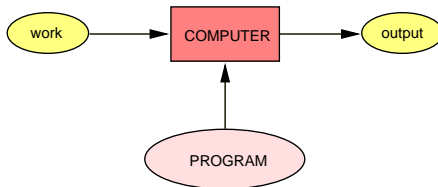
- Batteries are not included in the unit.
- Install three high quality “AA” size Alkaline (LR6) or Manganese (R6, UM-3) batteries. We recommend to use Alkaline batteries.
Battery life is: —about six months in use of Alkaline batteries.
 —about three months in use of Manganese batteries.
Battery life may depend on usage conditions and ambient temperature.

What's a Computer Program? ..

Human work model

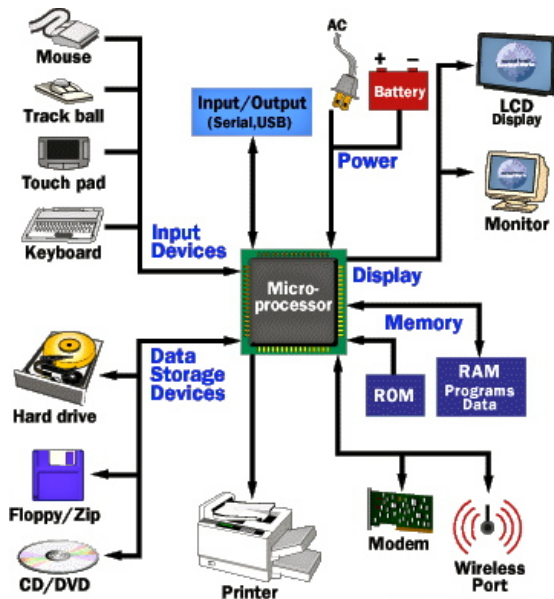


Computer work model



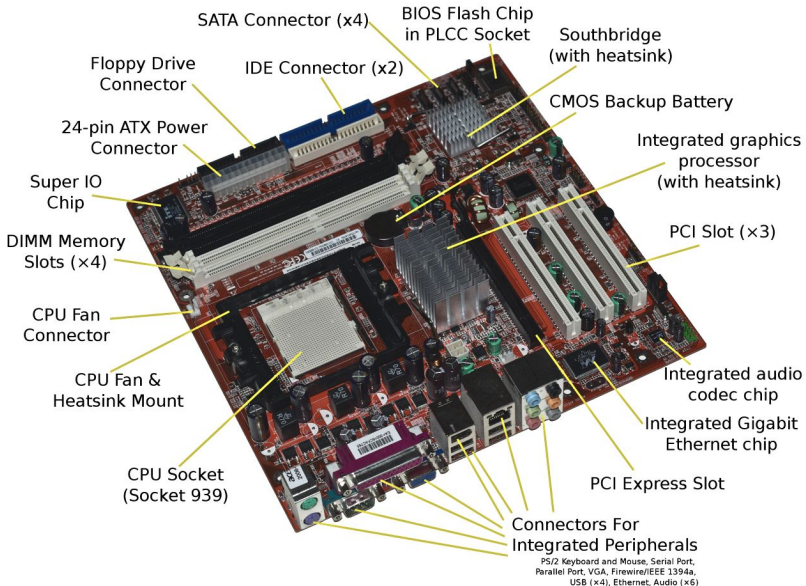
- A **computer program** is a set of **machine-readable instructions** that tells a computer how to perform a specific task. (During the execution of the program, it may interact with the users and its environment.)

Schematic Diagram of a Personal Computer

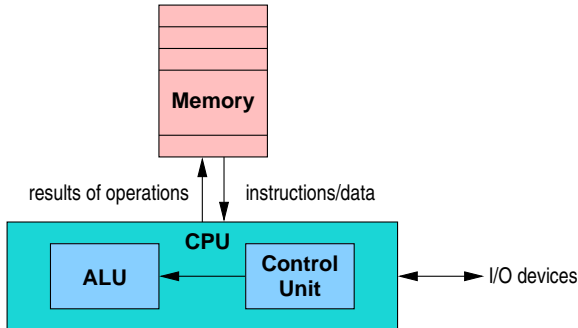


©2000 How Stuff Works

A Typical Motherboard



von Neumann Computer Architecture



- Designed by **John von Neumann**, a mathematician, in 1945.
- It is still today's dominant computer architecture.
- **CPU** = Central Processing Unit
- **ALU** = Arithmetic Logic Unit.
- For **efficiency**, many programming languages, including C++, are designed to take advantage of the architecture.
- More on this in COMP2611 (Computer Organization).

Can You Understand This?

0000100100101110011001100110100101101100011001010000100100100010011011000110010101100011011101000111
010101110010011001010011000100101110011000110010001000001010011001110110001101100011001100100101111
011000110110111101101101011100000110100101101100011001010110010000101110001110100000101000101110011
001101100101011000110111010001101001011011110110111000001001001000100010111001110100011001010111000
011101000010001000001010000010010010111001100001011011000110100101100111011011100010000000110100000
1010000010010010111001100110110110001011110110001001100000101101100001000000110110110000101101001
011011100000101000001001001011100111010001111001011100000110010100001001000000110101011000010110
10010101110000101100001000110100110011101010111001100011011101000110100101101111010111000001010
0000100100101110011100000111001001101111011000110000100100110000001101000000101001101101011000010110
1001011011100011101000001010000010010010000100100011010100000101001001001111010011000100111101000111
0101010101000101001000110010000000110000000010100000100101110011011000010111011001100101001000000010
010101110011011100000010110000101101001100010011001000111000001011000010010101110011011000000001010
000010100000100100100001001000110101000001001001001111010011000100111101000111010101010001010010
001100100000001100010000101000001001011011010110111101110110001000000011000100101100001001010110111
00110000000010100000100101110011011101000010000000100101101111001100000010110001011011001001010110
011001110000001011010011001000110000010111010000101000001001011010101101111011101100010000000110010
00101100001001010110111001100000000101000001001011100110110100001000000010010101110011000000010
11000101101100100101100111000000101100110010001010000101110100001010000010010110001100100
0010000001011011001001011001100111000000010110011001000100100010000010111010010110000100101011110011
0000000010100000100101101100011001000010000001011011001001010110011001110000001011010011001000110100
010111010010110000100101011011100110001000010100000100101100001011001000110010000100000001001010110
11110011000000101100001001010110111001100010010110000100101011011100110000000010100000100101110011
0111010000100000001001010111100110000001011000101100100100101011001100111000000101101001100100011
10000101101000001010000010010101101011101110110011000000001000000010100001001010100100110000
000010100000100101100010001000000010111001001100010011000011000100001010000010010110111001101110111
0000000010100010111001001100010011000011000100111010000010100000100101110010011001010111010000001010
00001001011100100110010101110011011101000110111101110010011001010000010100010111001001100010011000110
0110011001010011000100111010000010100000100100101110011100110110100101111010011001010000100100100000
011011010110000101101001011011100001011000001011001001100001001100011001100101001100001001011010110
110101100000101101001011011100000101000001001001110011010010110010001100011001011011100111010000001001
00100010010001110100001101000011001110100010000000101000010001110100111001010100101001001000000011
0010001011100011100000101110001100010010001000001010

How About This?

main:

```
    !#PROLOGUE# 0  
    save %sp,-128,%sp
```

```
    !#PROLOGUE# 1  
    mov 1,%o0  
    st %o0,[%fp-20]  
    mov 2,%o0  
    st %o0,[%fp-24]  
    ld [%fp-20],%o0  
    ld [%fp-24],%o1  
    add %o0,%o1,%o0  
    st %o0,[%fp-28]  
    mov 0,%i0  
    nop
```

Is This Better Now?

```
int main( )  
{  
    int x, y, z;  
  
    x = 1;  
    y = 2;  
    z = x+y;  
  
    return 0;  
}
```

Example: Write a Program to Sum 2 Numbers

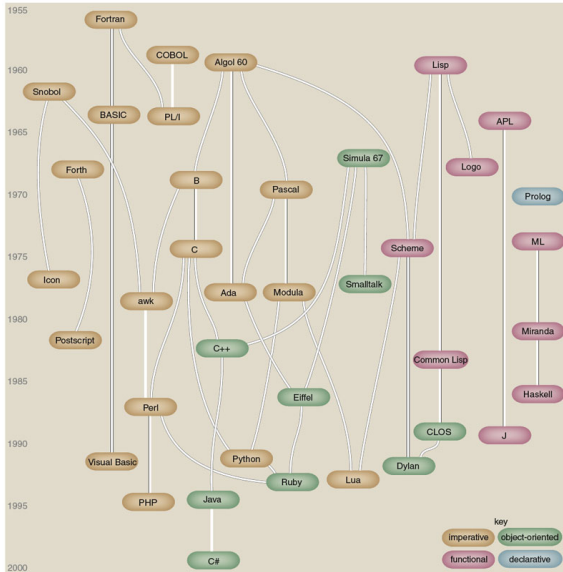
- There are 3 integer-value-holding objects: x, y, and z.
- x and y have the value of 1 and 2 respectively.
- z's value is the sum of x's and y's.

```
int main( )  
{  
    int x, y, z;  
  
    x = 1;  
    y = 2;  
    z = x+y;  
  
    return 0;  
}
```

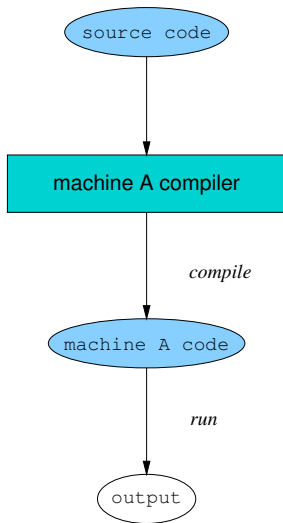
Levels of Programming Languages

- machine (binary) language is unintelligible
- assembly language is low level
 - mnemonic names for machine operations
 - explicit manipulation of memory addresses/contents
 - machine-dependent
- high level language
 - readable
 - instructions are easy to remember
 - faster coding
 - less error-prone (fewer bugs?)
 - easier to maintain
 - no mention of memory locations
 - machine-independent = portable

Chronology of Some Programming Languages



Compilation: From Source to Runnable Program



A **compiler** translates **source programs** into **machine codes** that run directly on the target computer.

For example,
`a.cpp` \longrightarrow `a.out` (or `a.exe`).

Some C++ compilers:
`gcc/g++`, `VC++`.

- static codes
- compile once, run many
- optimized codes
 \Rightarrow more efficient
- examples: FORTRAN, Pascal, C++

Programming as Problem Solving

- **Understand** and **define** the problem clearly.
 - What are the input(s) and output(s)?
 - Any constraints?
 - Which information is essential?
- **Develop** a solution.
 - Construct an algorithm.
- **Translate** the algorithm into a C++ program.
- **Compile** the program.
- **Test** the program.
- **Debug** the program.
- **Document** the program as you write the program.
- **Maintain** the program
 - modify the codes when conditions change.
 - enhance the codes to improve the solution.

- Why C++?

Read the FAQ from the designer of C++, Bjarne Stroustrup.

- Which C++?

- The language has been evolving:

C++ 1983 \Rightarrow C++ 1998 \Rightarrow C++ 2003 \Rightarrow C++ 2011 \Rightarrow ...

- In this course, we use C++ 2003.

(C++ 2011 is too new and its compilers are experimental.)

- Which compiler?

GNU gcc/g++. It is free.

- Which IDE (integrated development environment) for writing programs?

Eclipse. It is free and supported by many operating systems such as Windows, Mac OS, and Linux.