Introduction to Object-Oriented Programming

COMP2011: Introduction

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

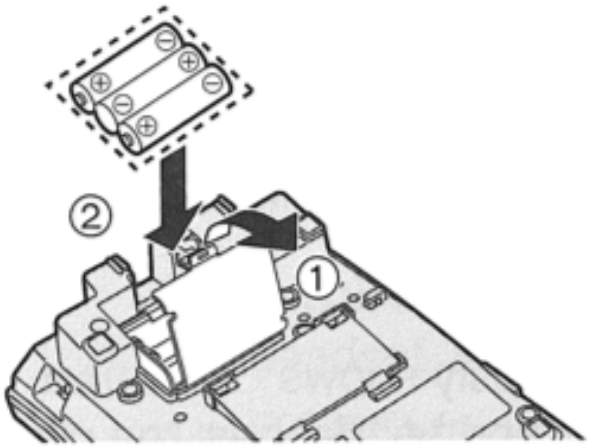
Department of Computer Science & Engineering
The Hong Kong University of Science and Technology
Hong Kong SAR, China

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).

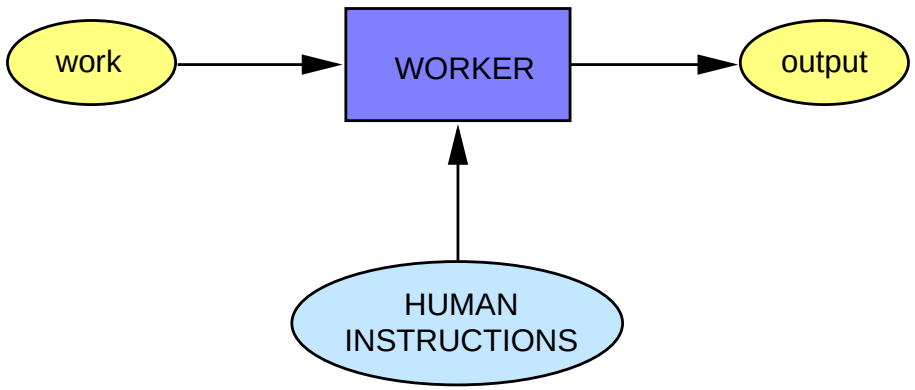
2 Install the batteries in the proper order as shown (②), matching the correct polarity.

3 Close the battery cover.

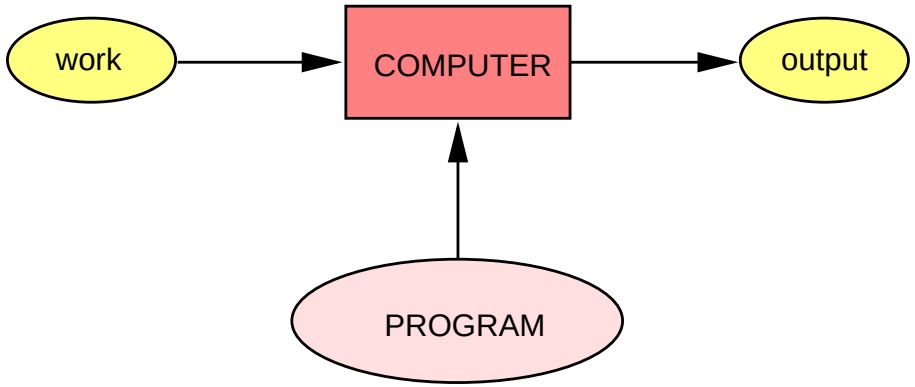
- Batteries are not included in the unit.
- Install three high quality "AA" size Alkaline (LR6) or Manganese (R6, UM-3) batteries. We recommend to use Alkaline batteries.
  Battery life is: —about six months in use of Alkaline batteries.
  —about three months in use of Manganese batteries.
  Battery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

What's a Computer Program? ..

Human work model

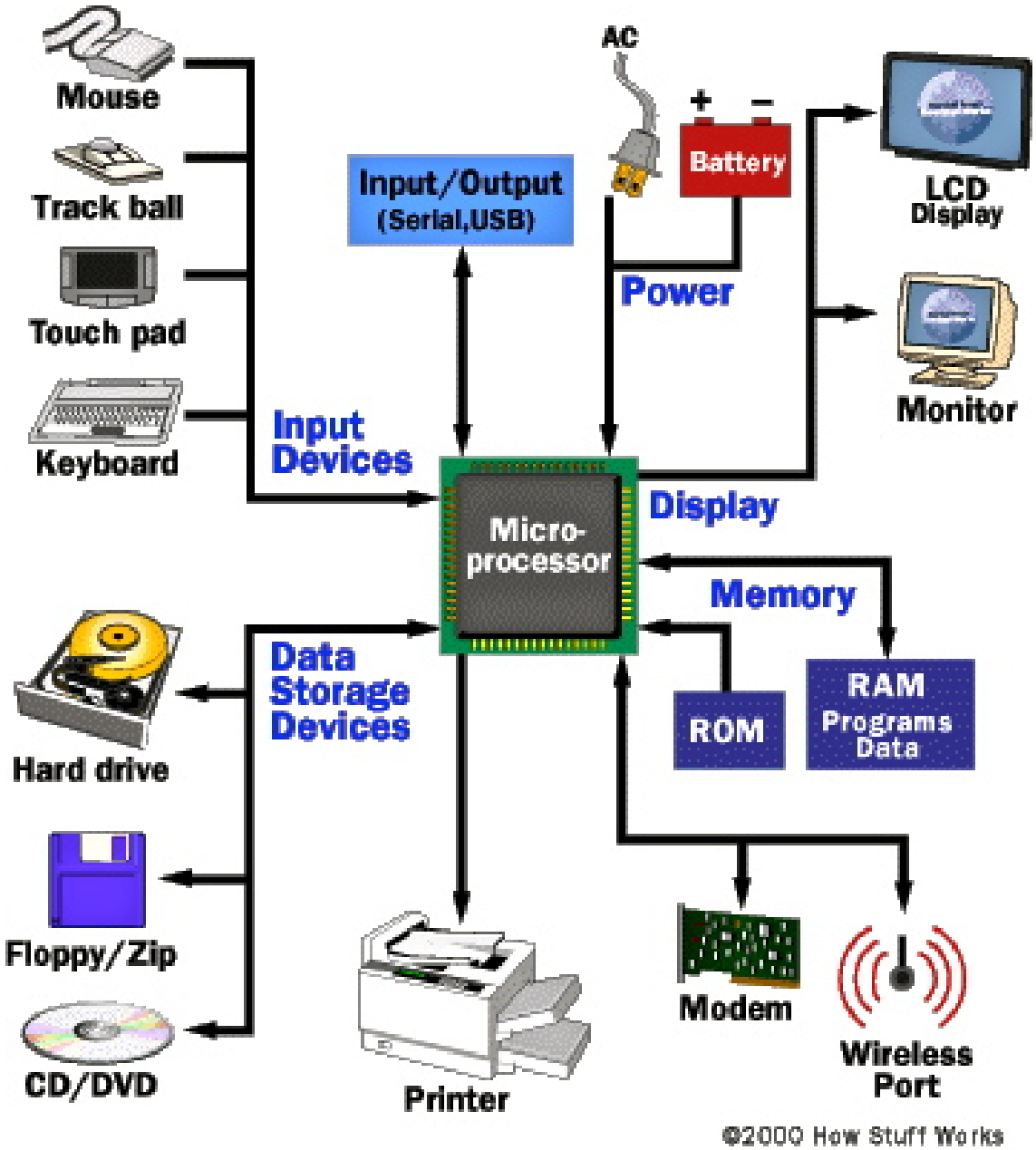


Computer work model

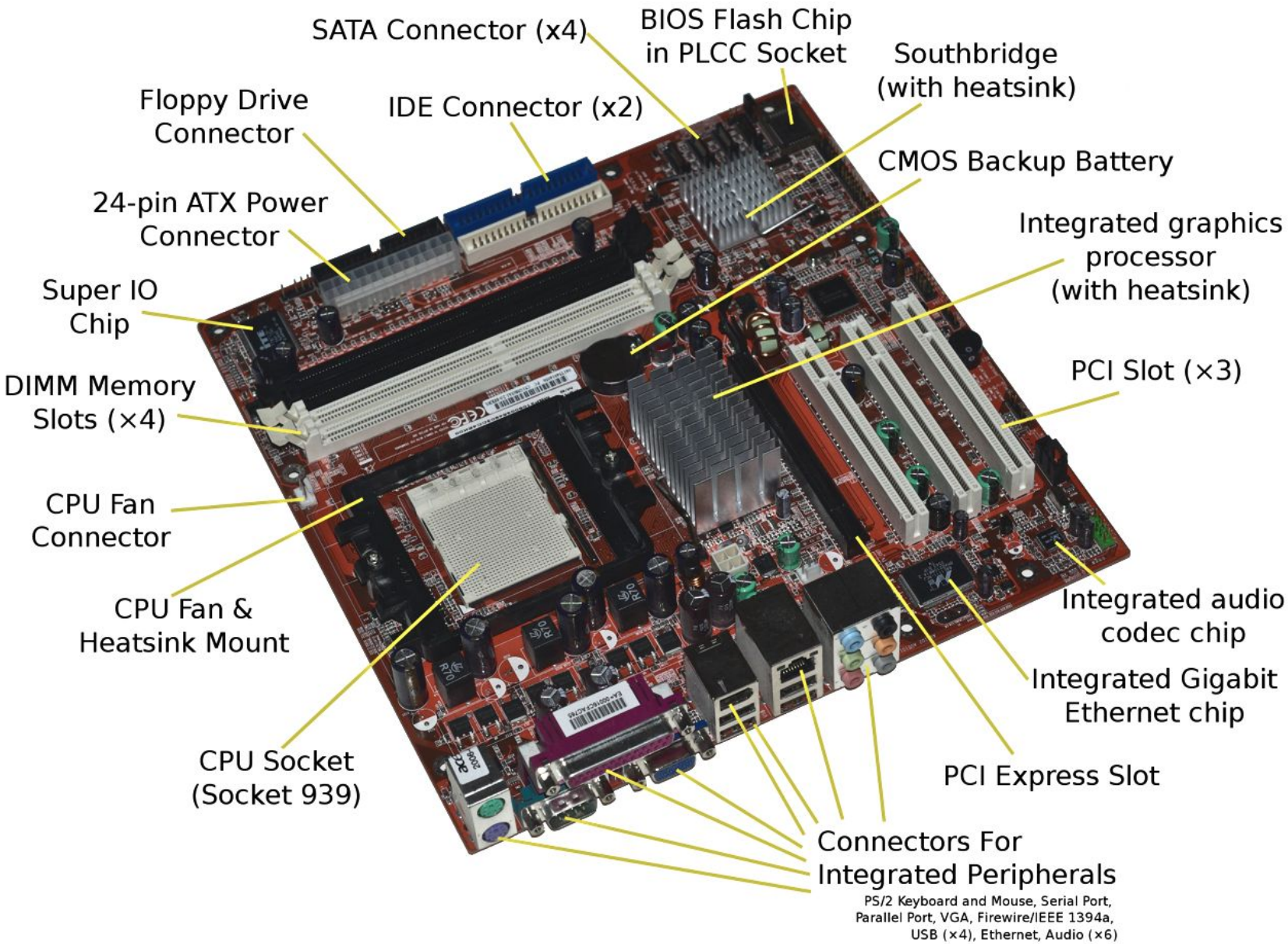


- A computer program is a set of machine-readable instructions that tells a computer how to perform a specific task. (During the execution of the program, it may interact with the users and its environment.)

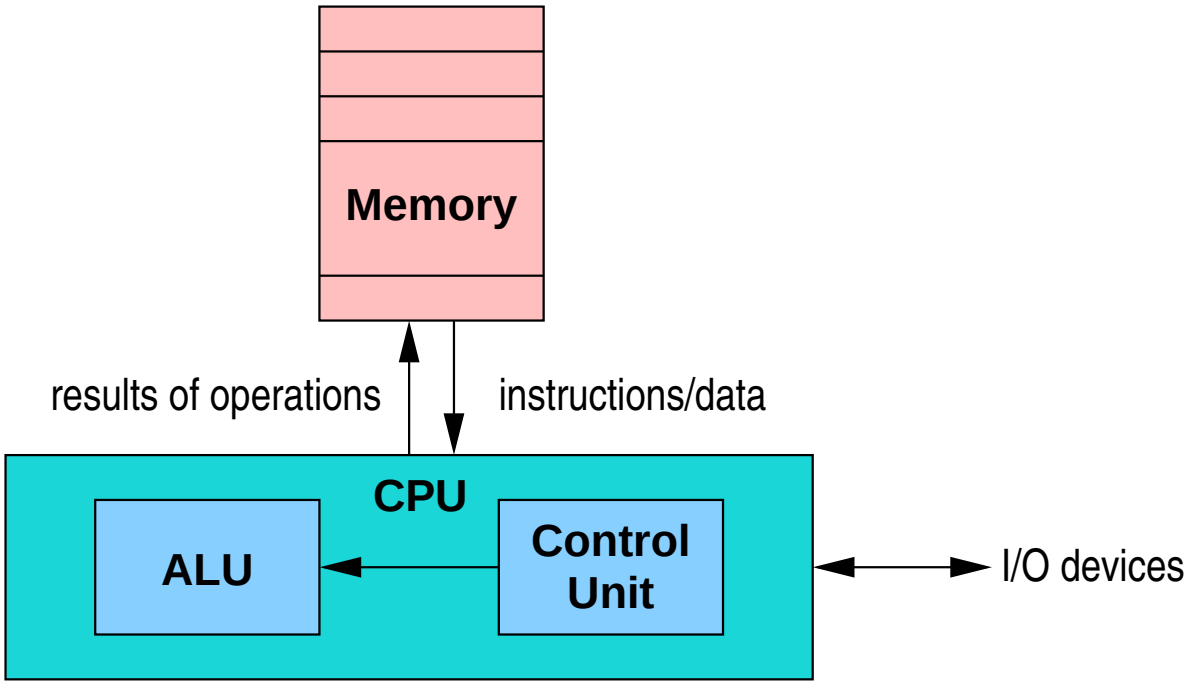
Schematic Diagram of a Personal Computer



A Typical Motherboard

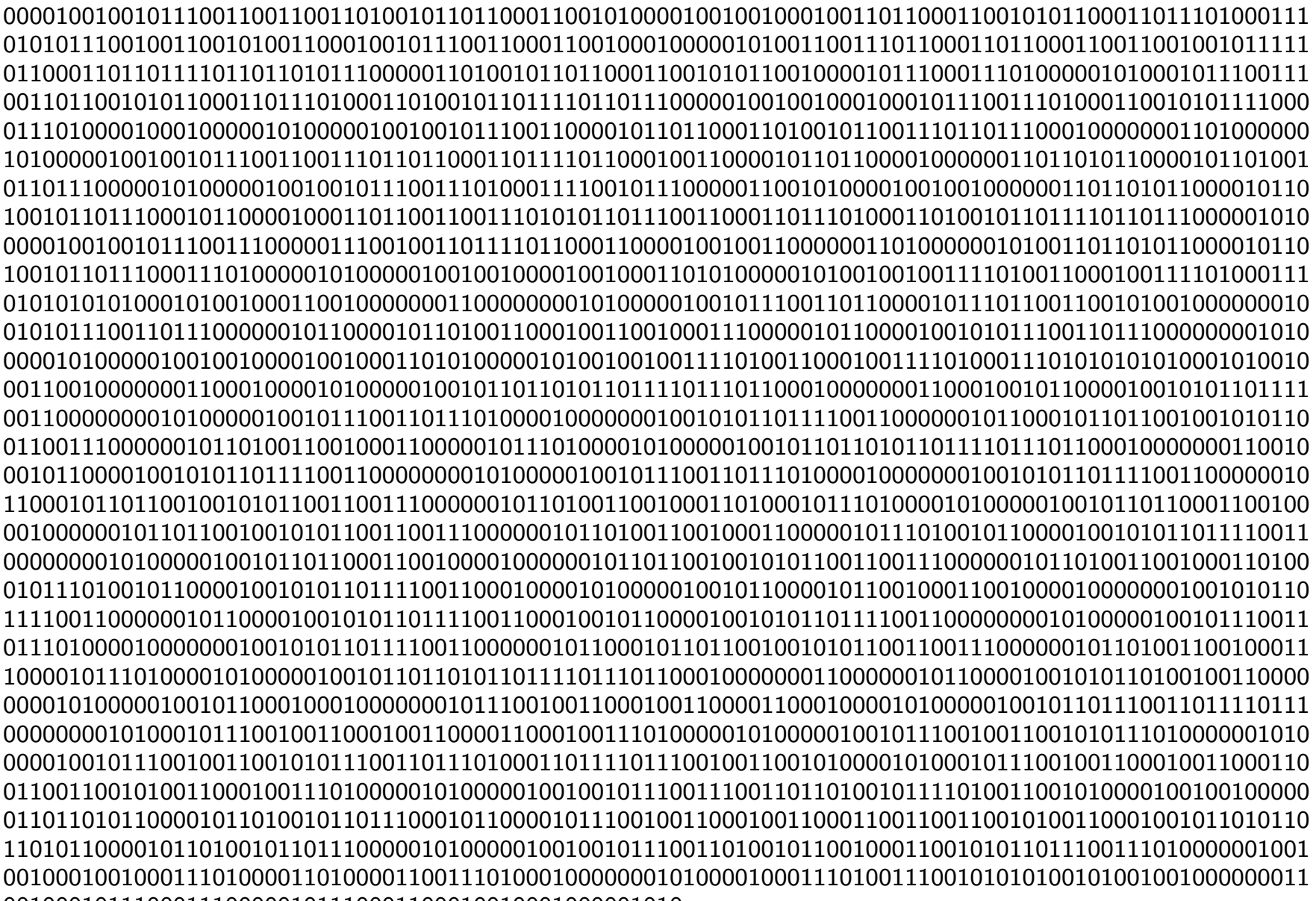


von Neumann Computer Architecture



- 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?



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

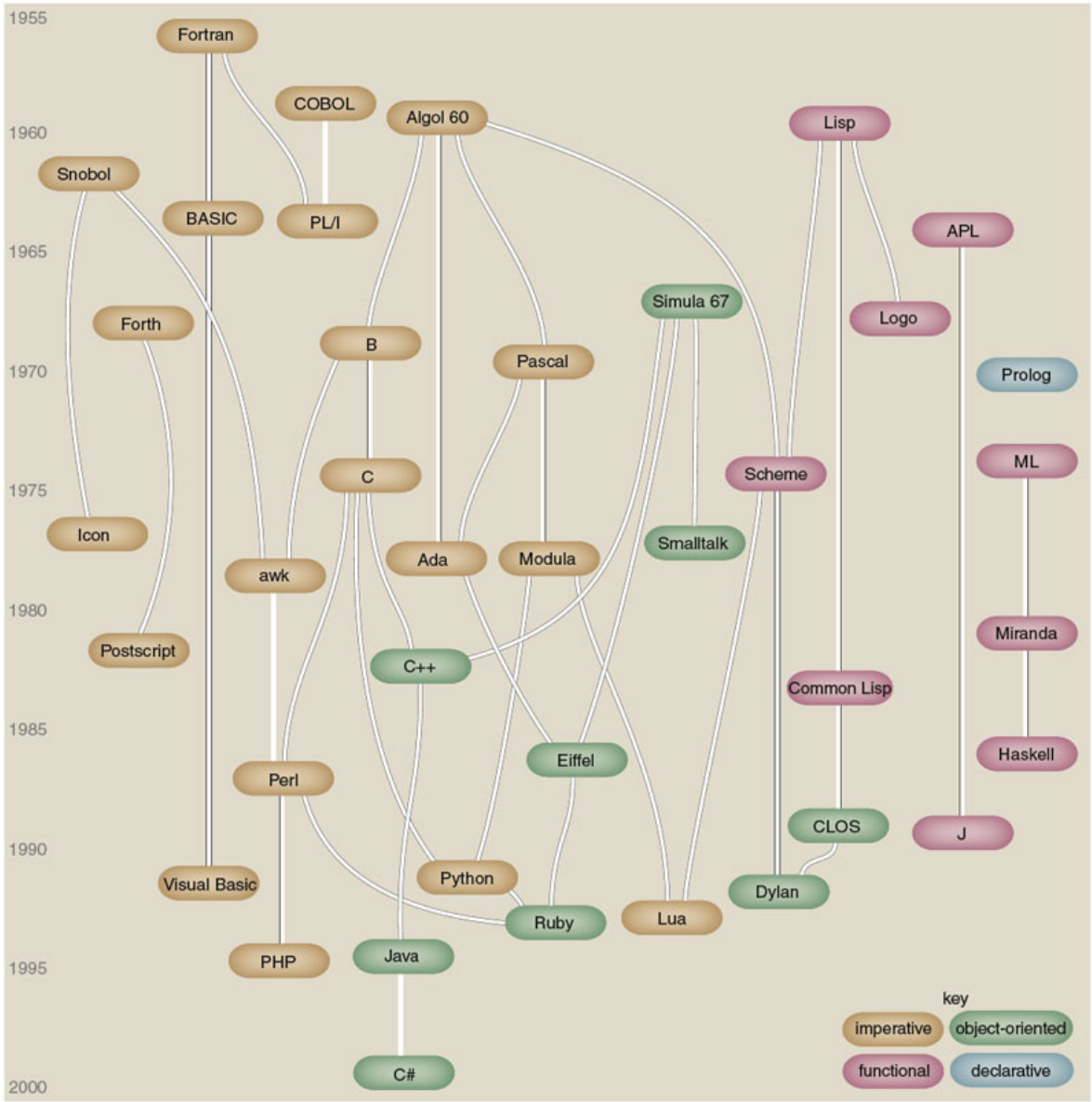
- There are 3 integer-value-holding objects: x, y, and z.
- x and y have the value of 1 and 2 respectively.
- z's value is the sum of x's and y's.

```
int main( )
{
    int x, y, z;

    x = 1;
    y = 2;
    z = x+y;

    return 0;
}
```

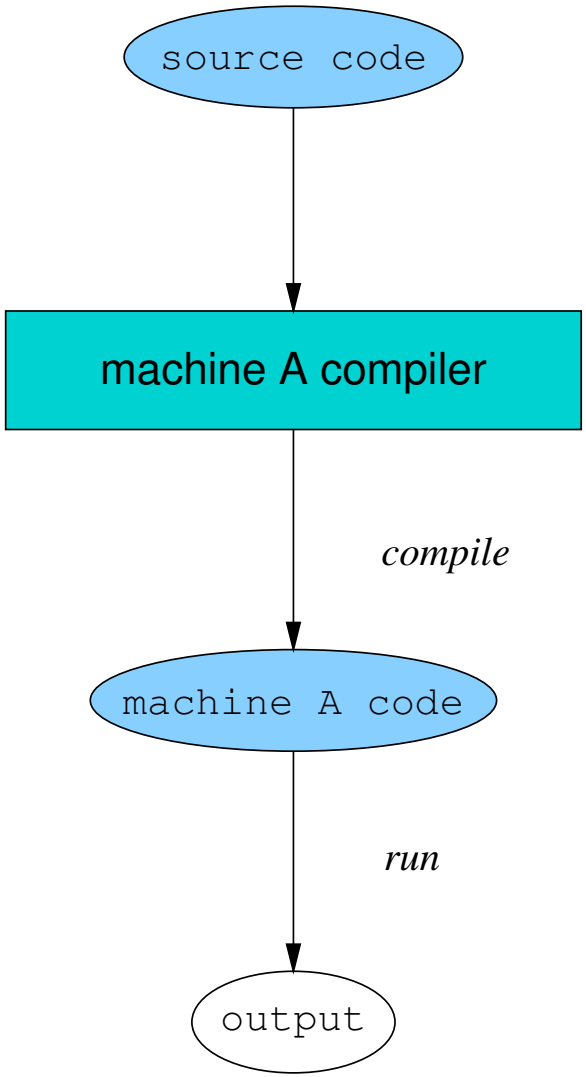
Levels of Programming Languages

- 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.

Summary

- 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 $\Rightarrow \cdots$
  - 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.