Programming Techniques I
SCJ1013
Introduction to Programming

Dr Masitah Ghazali

1.1 Why Program?

- **Computer**–programmable machine designed to follow instructions
- **Program**–instructions in computer memory to make it do something
- **Programmer**–person who writes instructions (programs) to make computer perform a task

SO, without programmers, no programs; without programs, a computer cannot do anything

What is Computer Program?

- A computer program is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.
- Examples of computer program or computer application?

First Programmer

- Ada Lovelace –

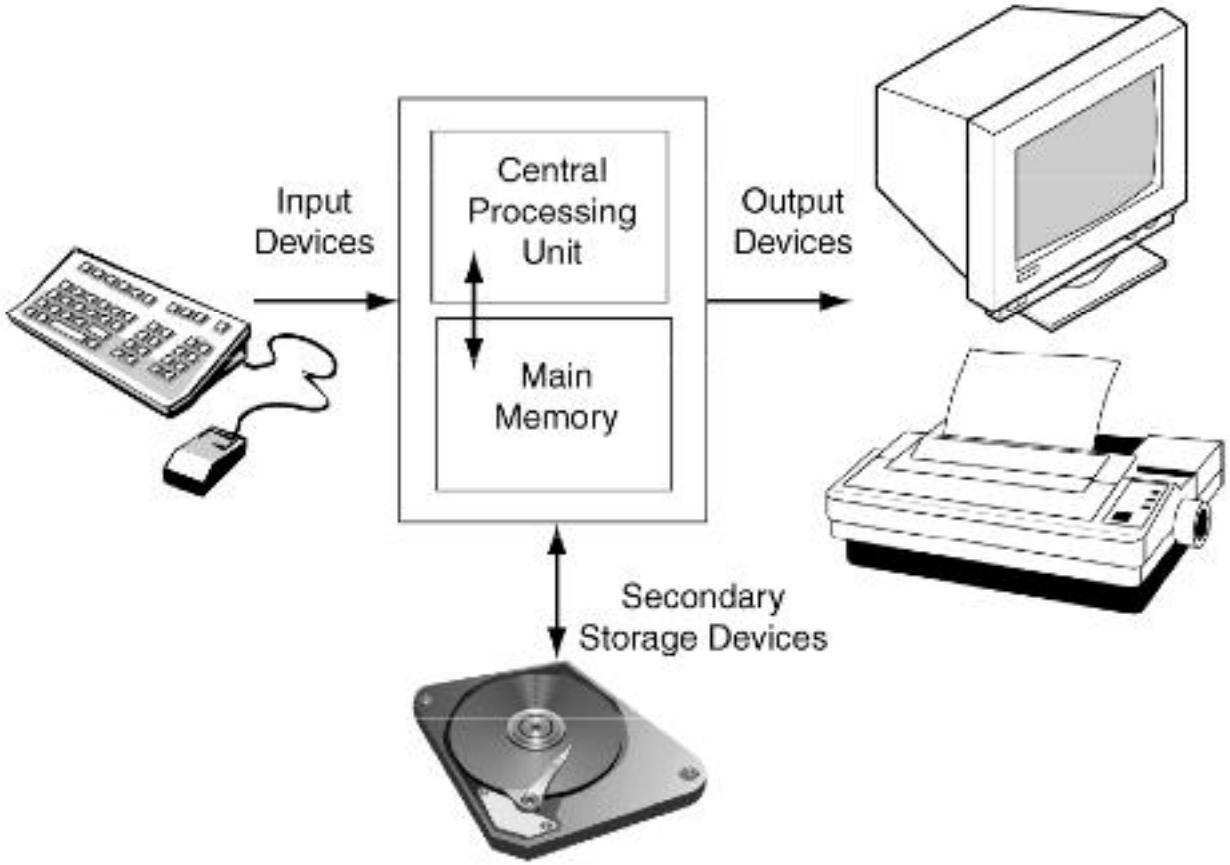
  During a nine-month period in 1842-1843, Ada translated Italian mathematician Luigi Menabrea's memoir on Babbage's newest proposed machine, the Analytical Engine

1.2 Computer Systems: Hardware and Software

Main hardware component categories:

1. Central Processing Unit (CPU)
2. Main Memory
3. Secondary Memory / Storage
4. Input Devices
5. Output Devices

Main hardware component categories



Central Processing Unit (CPU)

Comprised of:

Control Unit

Retrieves and decodes program instructions

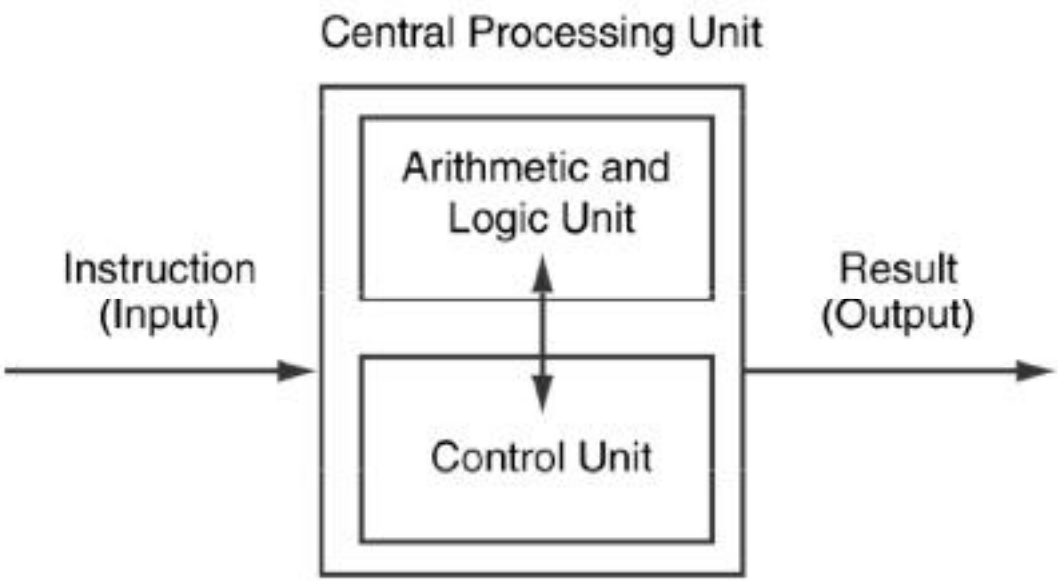
Coordinates activities of all other parts of computer

Arithmetic & Logic Unit

Hardware optimized for high-speed numeric calculation

Hardware designed for true/false, yes/no decisions

CPU Organisation



Typical capabilities of CPU

- Add
- Subtract
- Multiply
- Divide
- Move data from location to location

Main Memory

- Stores instructions and data that are to be processed by the computer.
- It is volatile. Main memory is erased when program terminates or computer is turned off
- Also known as Random Access Memory (RAM)

Main Memory

- Addresses –Each byte in memory is identified by a unique number - *address*
- Organized as follows:

  –Information is stored in *bits or binary digits*.

  –bit: smallest piece of memory. Has values 0 (off, false) or 1 (on, true)

  –byte: 8 consecutive bits. Bytes have addresses.

  –Each cell has its own address that indicate the location of stored data and instruction.

Main Memory

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16 149	17	18	19
20	21	22	23 72	24	25	26	27	28	29

- The number 149 is stored in the byte with the address 16,
- The number 72 is stored at address 23

Main Memory

- The size of a memory often given in terms of *Megabytes(MB) or Gigabytes(GB)*

1 megabyte = 220 = 1,048,576 bytes

Therefore, 300 MB = 300 x 1,048,576 = 314,572,800 bytes of storage

Secondary Storage

- Mass storage device.
- Stores instructions and data between sessions. Non-volatile: data retained when a program is not running or a computer is turned off
- Comes in a variety of media:
  - –magnetic: floppy disk, zip disk, hard drive
  - –optical: CD-ROM
  - –Flash drives, connected to the USB port

Secondary Storage

- Hard disk
  - –Fast
  - –Fixed in the computer and not normally removed
- Floppy disk
  - –Slow
  - –Easily shared with other computers
- Compact disk
  - –Slower than hard disks
  - –Easily shared with other computers
  - –Can be read only or re-writable

Input Devices

- Devices that send information to the computer from outside
- Many devices can provide input:
  - –Keyboard, mouse, scanner, digital camera, microphone
  - –Disk drives and CD-ROM

Output Devices

- Output is information sent from a computer program to the outside world.
- The output is sent to an output device
- Many devices can be used for output:
  - –Computer monitor and printer
  - –Floppy, zip disk drives
  - –Writable CD drives

Software – Programs that run on a computer

- Categories of software:
  - –System Software: programs that manage the computer hardware and the programs that run on them. Example: Operating system (Windows, UNIX, Linux), utilities, programming language systems
  - –Application software: programs that provide services to the user. *Examples: word processing, games, programs to solve specific problems*

1.3 Programs and Programming Languages

- A program is a set of instructions that the computer follows to perform a task
- Programming Language: a language used to write programs
- We start with an *algorithm, which is a set of well-defined steps.*

Example: Algorithm for Calculating Gross Pay

1. Display a message on the screen asking "How many hours did you work?"
2. Wait for the user to enter the number of hours worked. Once the user enters a number, store it in memory.
3. Display a message on the screen asking "How much do you get paid per hour?"
4. Wait for the user to enter an hourly pay rate. Once the user enters a number, store it in memory.
5. Multiply the number of hours by the amount paid per hour, and store the result in memory.
6. Display a message on the screen that tells the amount of money earned. The message must include the result of the calculation performed in Step 5.

Types of Programming Language

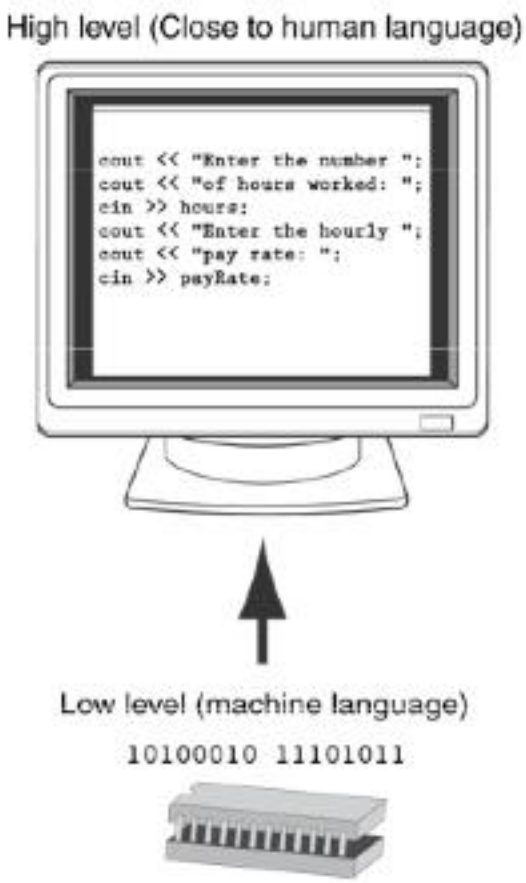
- Machine language: the only language the computer can understand.
  – in binary machine code (0's/1's) directly.
- Assembly language: one level above machine language. A low level language that is processor dependent.
  – Each CPU has its own assembly language.
- High-level language: closer to human language. A language that people can read, write, and understand. – High level programming languages are not processor dependent.
  – A program written in a high-level language must be translated into a language that can be understood by a computer before it can be run

Machine Language

- Although the previous algorithm defines the steps for calculating the gross pay, it is not ready to be executed on the computer.
- The computer only executes *machine language instructions*.
- Machine language instructions are binary numbers, such as as 1011010000000101
- Rather than writing programs in machine language, programmers use programming languages.

Programs and Programming Languages

- Types of languages:
  - –Low-level: used for communication with computer hardware directly. Often written in binary machine code (0's/1's) directly.
  - –High-level: closer to human language language



Some Well-known Programming Languages

Table 1-1

Language	Description
BASIC	Beginners All-purpose Symbolic Instruction Code. A general programming language originally designed to be simple enough for beginners to learn.
FORTRAN	Formula Translator. A language designed for programming complex mathematical algorithms.
COBOL	Common Business-Oriented Language. A language designed for business applications.
Pascal	A structured, general-purpose language designed primarily for teaching programming.
C	A structured, general-purpose language developed at Bell Laboratories. C offers both high-level and low-level features.
C++	Based on the C language, C++ offers object-oriented features not found in C. Also invented at Bell Laboratories.
C#	Pronounced "C sharp." A language invented by Microsoft for developing applications based on the Microsoft .NET platform.
Java	An object-oriented language invented at Sun Microsystems. Java may be used to develop programs that run over the Internet, in a Web browser.
Visual Basic	A Microsoft programming language and software development environment that allows programmers to quickly create Windows-based applications.

From a High-level Program to an Executable File

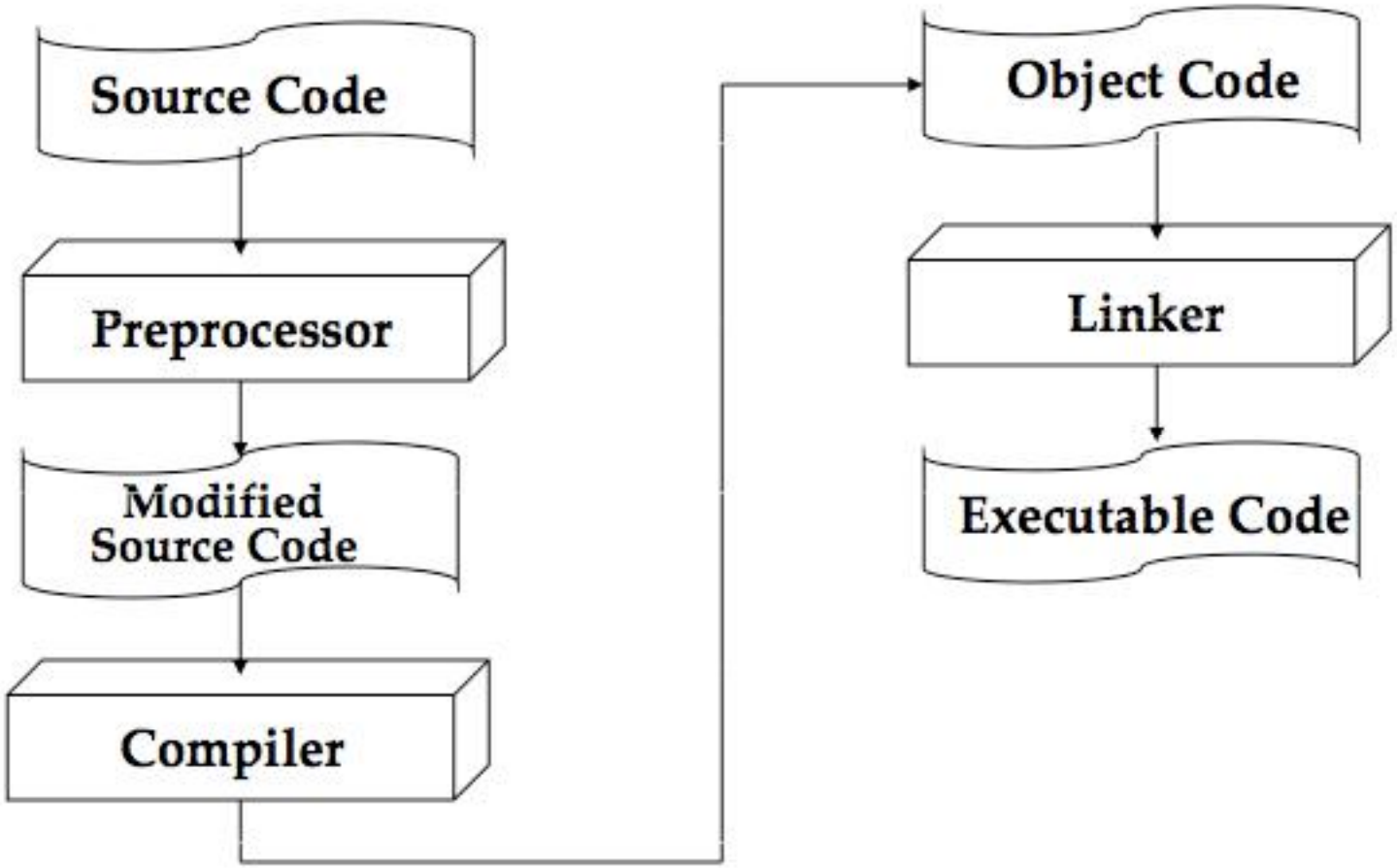
a) Create file containing the program with a text editor.

b) Run preprocessor to convert source file directives to source code program statements.

c) Run compiler to convert source program into machine instructions instructions.

d) Run linker to connect hardware-specific code to machine instructions, producing an executable file.

- Steps b–d are often performed by a single command or button click.
- Errors detected at any step will prevent execution of following steps.


Source Code
Preprocessor
Modified Source Code
Compiler
Object Code
Linker
Executable Code

Integrated Development Environments (IDEs)

- An integrated development environment, or IDE, combine all the tools needed to , write, compile, and debug a program into a singlesoftwareapplication single software application.
- Examples are Microsoft Visual C++, C C Borland C++ Builder, CodeWarrior, etc.

GrossPay - Microsoft Visual C++ [design] - Pr1-1.cpp*

```
// This progam calculates the user's pay.
#include <iostream>
using namespace std;

int main()
{
    double hours, rate, pay;

    // Get the number of hours worked.
    cout << "How many hours did you work? ";
    cin >> hours;

    // Get the hourly pay rate.
    cout << "How much do you get paid per hour? ";
    cin >> rate;

    // Calculate the pay.
    pay = hours * rate;

    // Display the pay.
    cout << "You have earned $" << pay << endl;
    return 0;
}
```

Exercise Week 1

- Do Exercise 1, No. 2-6 in pg. 3-10.
- Do Exercise 2, No. 1-6 in pg. 10-12.
- • Follow the instructions.

- Do Exercise 3, No 1, in pg 12-14.
- Understand the program.