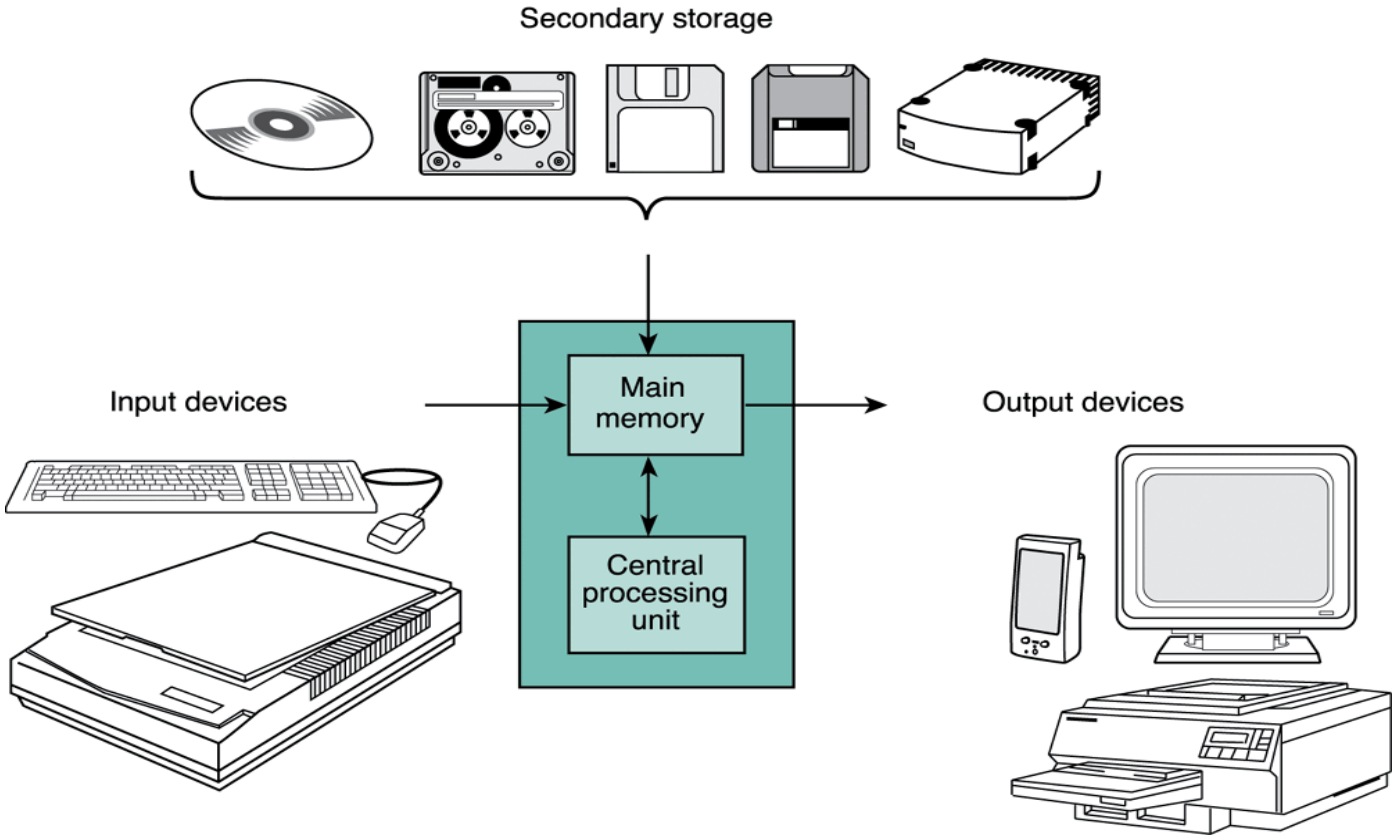
Lecture 02

Overview of Programming

CSE115: Computing Concepts

Computer Hardware Components



Components of a PC

Input / Output Devices

- Input Devices
  - Accepts information from the user and transforms it to ***digital codes*** that the computer can process
  - Example: keyboard, mouse, scanner
- Output Devices
  - An ***interface*** by which the computer conveys the output to the user
  - Example: monitor, printer

Main Memory

- A semiconductor device which stores the information necessary for a program to run.
- 2 types
  - ROM (Read Only Memory)
  - Contains information that is necessary for the computer to boot up
  - The information stays there permanently even when the computer is turned off.
  - RAM (Random Access Memory)
  - Contains instruction or data needed for a program to run
  - Gets erased when the computer is turned off.

Central Processing Unit (CPU)

- Does most of the work in executing a program
- The CPU inside a PC is usually the microprocessor
- 3 main parts:
  - Control Unit
  - Fetch instructions from main memory and put them in the instruction register
  - ALU (Arithmetic Logic Unit)
  - Executes arithmetic operations
  - Registers
  - Temporarily store instructions or data fetched from memory

Storage Devices

- A magnetic device used to store a large amount of information.
- Store the software components or data needed for the computer to execute its tasks.
- Can be “read only” or “writable”.
- Example: Hard drive, CD ROM, floppy disks

But is hardware enough?

- A body does not work without soul
- Similarly, a computer does not work without any *software/program*
- More specifically, computer is just a tool that can be used to achieve many goals
- Certain software/program are needed to achieve certain goals; e.g.
  - Windows: to operate the computer
  - MS Word: to edit documents
  - MS Paint: to draw pictures

What is a program?

A program is a list of ***instructions*** for the computer to perform a specific action or a specific task such as:

'Calculate the sum of the numbers from 1 to 10‘
'Print “I like programming”‘
'Output the current time'

What is Programming?

- ***Programming*** is instructing a computer to do something for you
- But computer can only understand 0 and 1!
  - It doesn't understand our language
  - How can we tell computer what to do and how?
- We can do this with the help of a **programming language**

Programming Language

- Can be classified into:
  - Machine Languages
  - Assembly Languages
  - High-Level Languages

Machine Language

- The only language that the processor actually 'understands‘
- Consists of binary codes: 0 and 1
  - Example: 00010101 (add 010 and 101)
    11010001 (subtract 001 from 010)
    01001100 (multiply 001 with 100)
- Each of the lines above corresponds to a specific task to be done by the ***processor***.
- Programming in machine code is difficult and slow since it is difficult to memorize all the instructions.
- Mistakes can happen very easily.
- Processor and Architecture dependent

Assembly Language

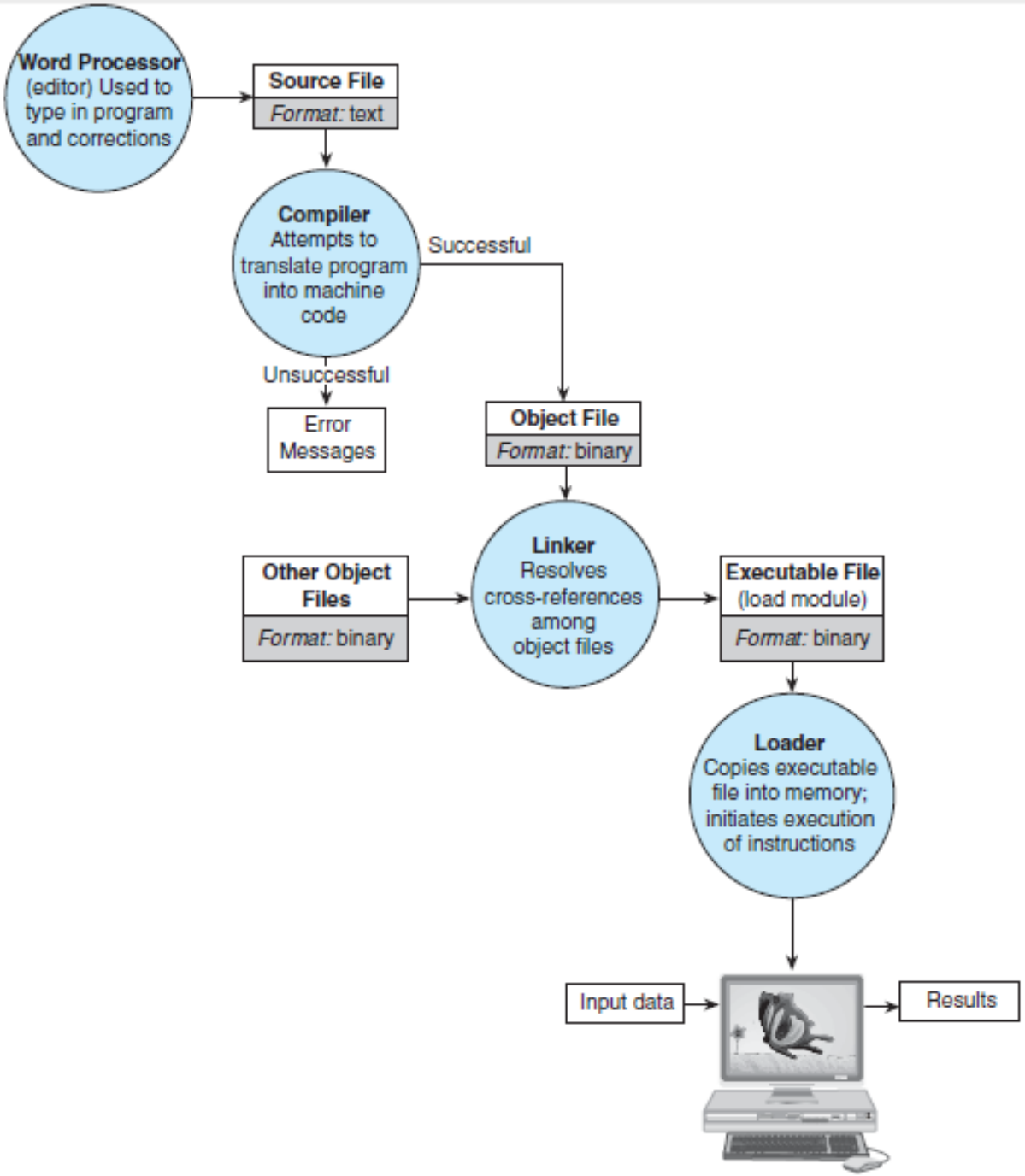
- Enables machine code to be ***represented*** in words and numbers.
- Example of a program in ***assembler language***:

```
MOV A, 010
ADD A, 101
```

- Easier to understand and memorize (called ***Mnemonics***), compared to **machine code** but still quite difficult to use.
- Processor and Architecture dependent

High-Level Language

- Use more English words. They try to resemble English sentences. Therefore, it is easier to program in these languages.
- The programming structure is **problem oriented** - does not need to know how the computer actually ***executes*** the instructions.
- Processor **independent** - the same code can be run on different processors.
- Examples: ***Basic, Fortran, Pascal, Cobol, C, C++, Java***
- A high level language needs to be translated (or, *compiled*) into **machine code** so that it can be **executed** by the processor.
  - **compiler** is a software that can do this translation


Word Processor
(editor) Used to type in program and corrections
Source File
Format: text
Compiler
Attempts to translate program into machine code
Successful
Unsuccessful
Error Messages
Object File
Format: binary
Other Object Files
Format: binary
Linker
Resolves cross-references among object files
Executable File
(load module)
Format: binary
Loader
Copies executable file into memory; initiates execution of instructions
Input data
Results

C Programming Language

Why 'C' ?

- Because based on '**B**'; developed at **B**ell Laboratories
- Developed by **Dennis Ritchie** at Bell Laboratories in the 1960s
- In cooperation with **Ken Thomson** it was used for Unix systems

A Simple Program in C

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

A Simple Program in C

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

stan**d**ard Library, **i**nput-**o**utput, **h**eader-file

A Simple Program in C

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

Beginning of program

A Simple Program in C

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

Start of Segment

End of Segment

A Simple Program in C

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

Function for printing text

End of statement

Output

Hello world!