

Principles of Programming

Dr. Mohammad Ahmad

Outline

- **Syllabus**
- **First Program**

Focus of the Course

- **This is an intro to problem solving and programming class (that uses the C programming language)**
- **The focus is on**
 - **problem solving**
 - **the logic of programming**
 - **program design, implementation, and testing**
 - **the fundamentals of programming**

Warnings

- **Learning how to program takes a lot of time!**
- **It also requires a lot of patience.**
- **You cannot learn how to program by just reading the textbook. You have to spend long hours in front of the computer.**
- **If you want to learn how to program well you will have to take at least 2-3 other programming classes as well. This class alone is not enough!**
- **This class is not exclusively about writing code. It also emphasizes Problem Solving.**



First Program

The C Programming Language

- A *programming language* specifies the words and symbols that we can use to write a program.
- A programming language employs a set of rules that dictate how the words and symbols can be put together to form valid *program statements*.

Note: The computer will always do what you tell it to do: not what you want it to do.

Reserved Words in C

**auto break case char const continue
default do double else enum extern
float for goto if int long register
return short signed sizeof static
struct switch typedef union
unsigned void volatile while**

C Program Structure

```
#include <stdio.h>
main()
{
    // comments about the program
    printf("Hello World! \n");
}
```


C Program Structure

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

```
main()
```

```
{
```

All programs must have a main function



```
}
```

C Program Structure

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

```
main( )
```

```
{
```

main function body

Comments can be placed almost anywhere

```
}
```

Comments

- Comments in a program are called *inline documentation*
- They should be included to explain the purpose of the program and describe processing steps
- They do not affect how a program works
- C comments can take two forms:

```
// this comment runs to the end of the line
```

```
/* this comment runs to the terminating  
symbol, even across line breaks */
```

White Space

- Spaces, blank lines, and tabs are called *white space*
- White space is used to separate words and symbols in a program
- Extra white space is ignored
- A valid C program can be formatted many ways
- Programs should be formatted to enhance readability, using consistent indentation

Identifiers

- *Identifiers* are the words a programmer uses in a program
- An identifier can be made up of letters, digits, the underscore character (`_`), and the dollar sign
- Identifiers cannot begin with a digit
- **C is *case sensitive*** - `Total`, `total`, and `TOTAL` are different identifiers

Hardware and Software

- **Hardware**
 - the physical, tangible parts of a computer
 - keyboard, monitor, disks, wires, chips, etc.
- **Software**
 - programs and data
 - a *program* is a series of instructions
- **A computer requires both hardware and software**
- **Each is essentially useless without the other**

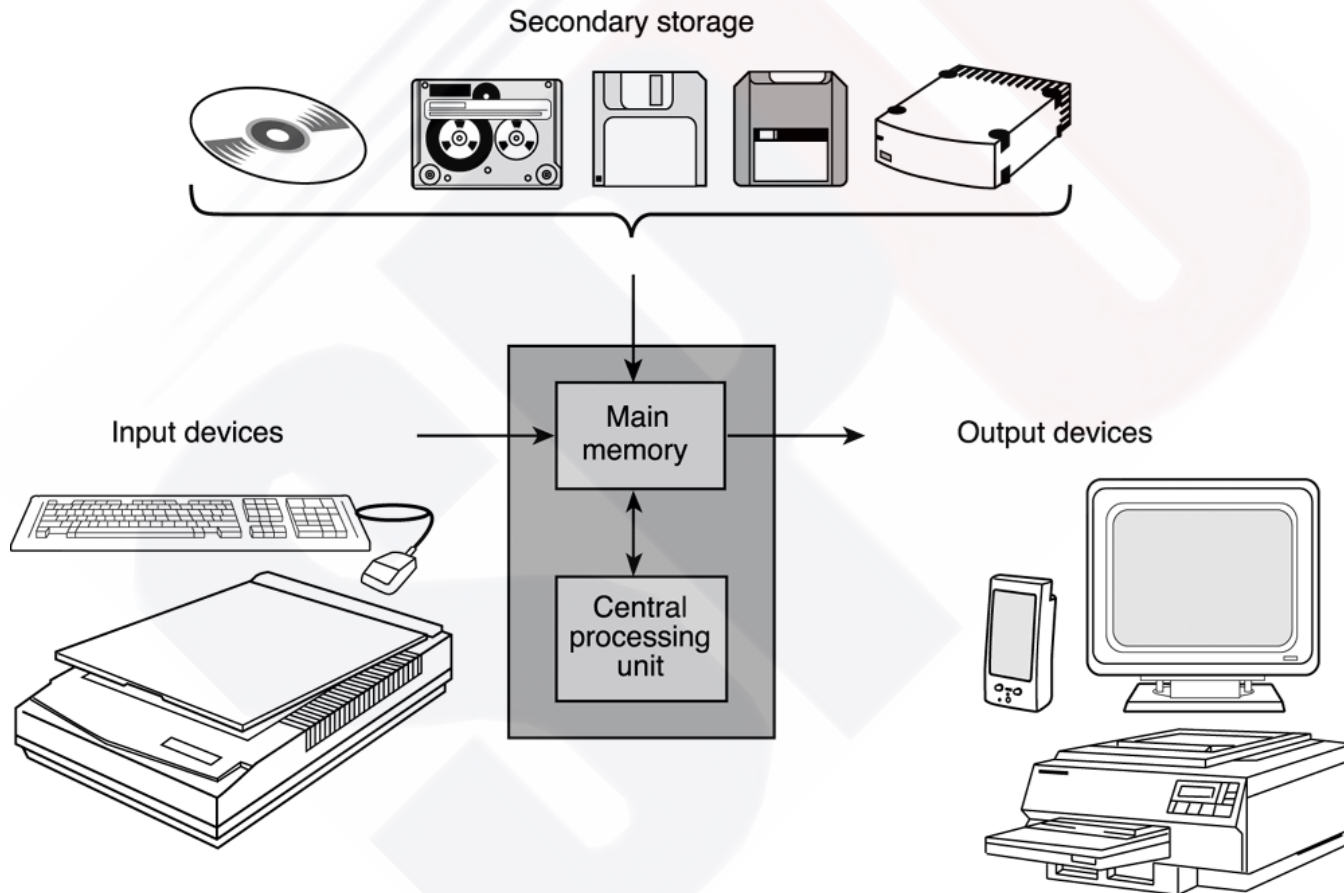
Software Categories

- **Operating System**
 - controls all machine activities
 - provides the user interface to the computer
 - manages resources such as the CPU and memory
 - Windows XP, Unix, Linux, Mac OS
- **Application program**
 - generic term for any other kind of software
 - word processors, missile control systems, games
- **Most operating systems and application programs have a *graphical user interface* (GUI)**

A Computer Specification

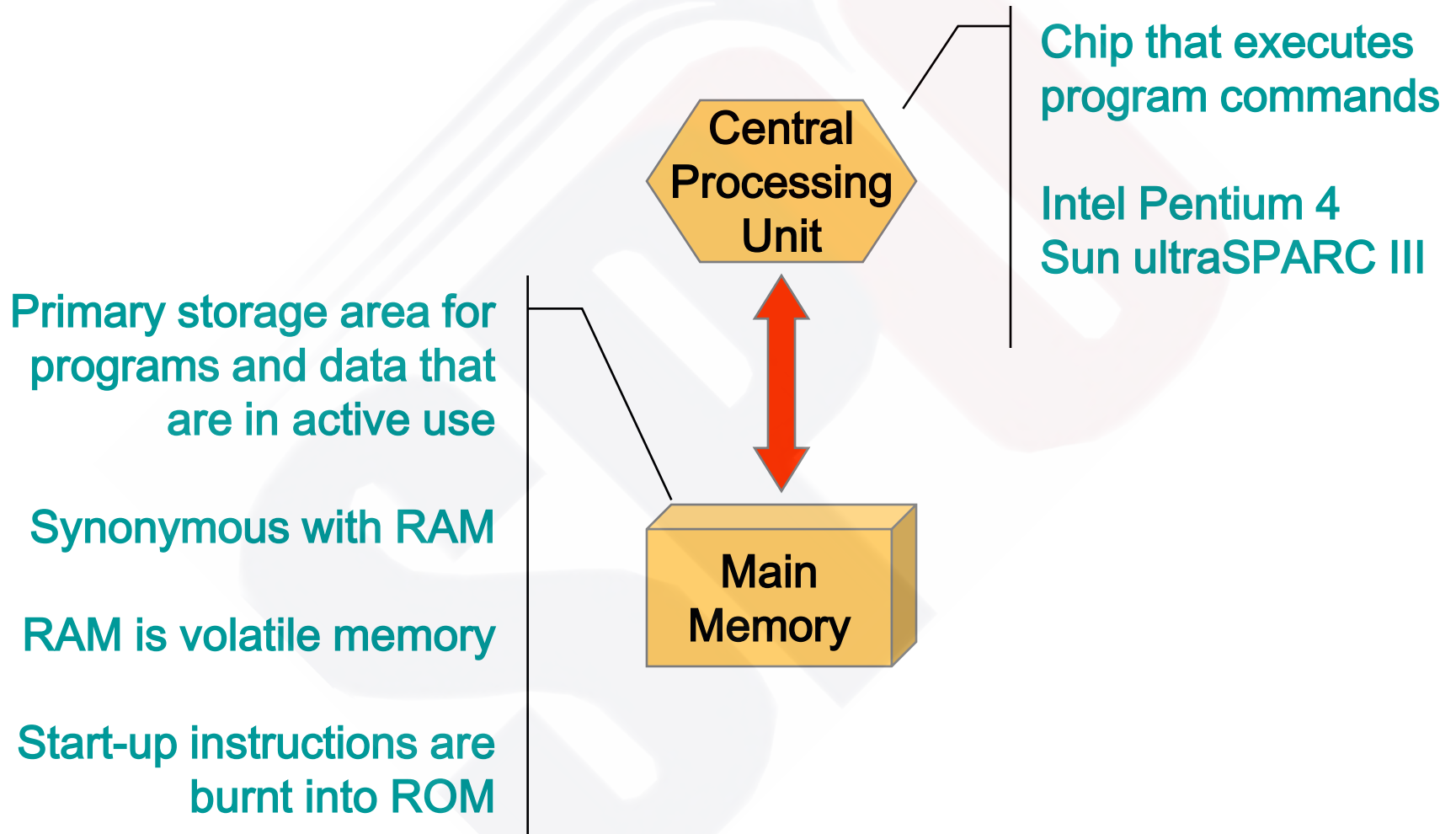
- **Consider the following specification for a personal computer:**
 - **2.8 GHz Pentium 4 Processor**
 - **512 MB RAM**
 - **80 GB Hard Disk**
 - **48x CD-RW / DVD-ROM Combo Drive**
 - **17" Video Display with 1280 x 1024 resolution**
 - **56 Kb/s Modem**
- **What does it all mean?**

Components of a Computer



[Figure 1.3 in the textbook]

CPU and Main Memory



Memory

Address	Contents
0	-27.2
1	354
2	0.005
3	-26
4	H
.	.
.	.
.	.
998	X
999	75.62

Most modern computers are byte-addressable

Memory Analogy



Memory Analogy



Memory Analogy



Relationship Between a Byte and a Bit



Digital Information

- **Computers store all information digitally:**
 - numbers
 - text
 - graphics and images
 - video
 - audio
 - program instructions
- **In some way, all information is *digitized* - broken down into pieces and represented as numbers**

Representing Text Digitally

- For example, every character is stored as a number, including spaces, digits, and punctuation
- Corresponding upper and lower case letters are separate characters

Hi, Heather.

72 105 44 32 72 101 97 116 104 101 114 46

H	i	,		H	e	a	t	h	e	r	.
72	105	44	32	72	101	97	116	104	101	114	46

Binary Numbers

- **Once information is digitized, it is represented and stored in memory using the *binary number system***
- **A single binary digit (0 or 1) is called a *bit***
- **Devices that store and move information are cheaper and more reliable if they have to represent only two states**
- **A single bit can represent two possible states, like a light bulb that is either on (1) or off (0)**
- **Permutations of bits are used to store values**

Bit Permutations

<u>1 bit</u>	<u>2 bits</u>	<u>3 bits</u>	<u>4 bits</u>	<u>4 bits</u>
0	00	000	0000	1000
1	01	001	0001	1001
	10	010	0010	1010
	11	011	0011	1011
		100	0100	1100
		101	0101	1101
		110	0110	1110
		111	0111	1111

Each additional bit doubles the number of possible permutations

Bit Permutations

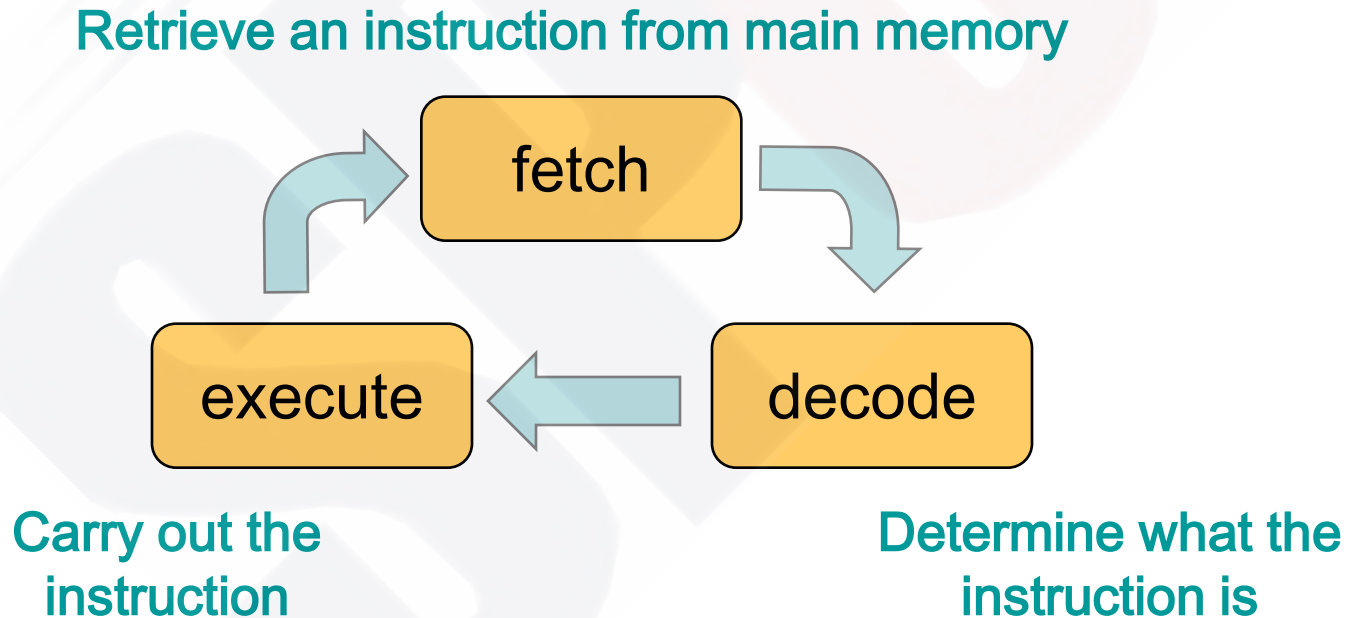
- Each permutation can represent a particular item
- There are 2^N permutations of N bits
- Therefore, N bits are needed to represent 2^N unique items

How many
items can be
represented by

1 bit ?	$2^1 = 2$ items
2 bits ?	$2^2 = 4$ items
3 bits ?	$2^3 = 8$ items
4 bits ?	$2^4 = 16$ items
5 bits ?	$2^5 = 32$ items

The Central Processing Unit

- A CPU is on a chip called a *microprocessor*
- It continuously follows the *fetch-decode-execute cycle*:

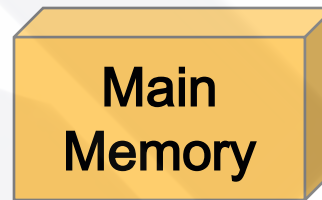
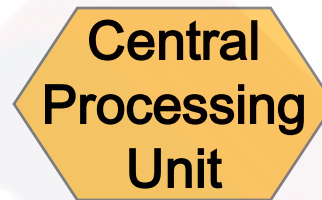


Secondary Memory Devices

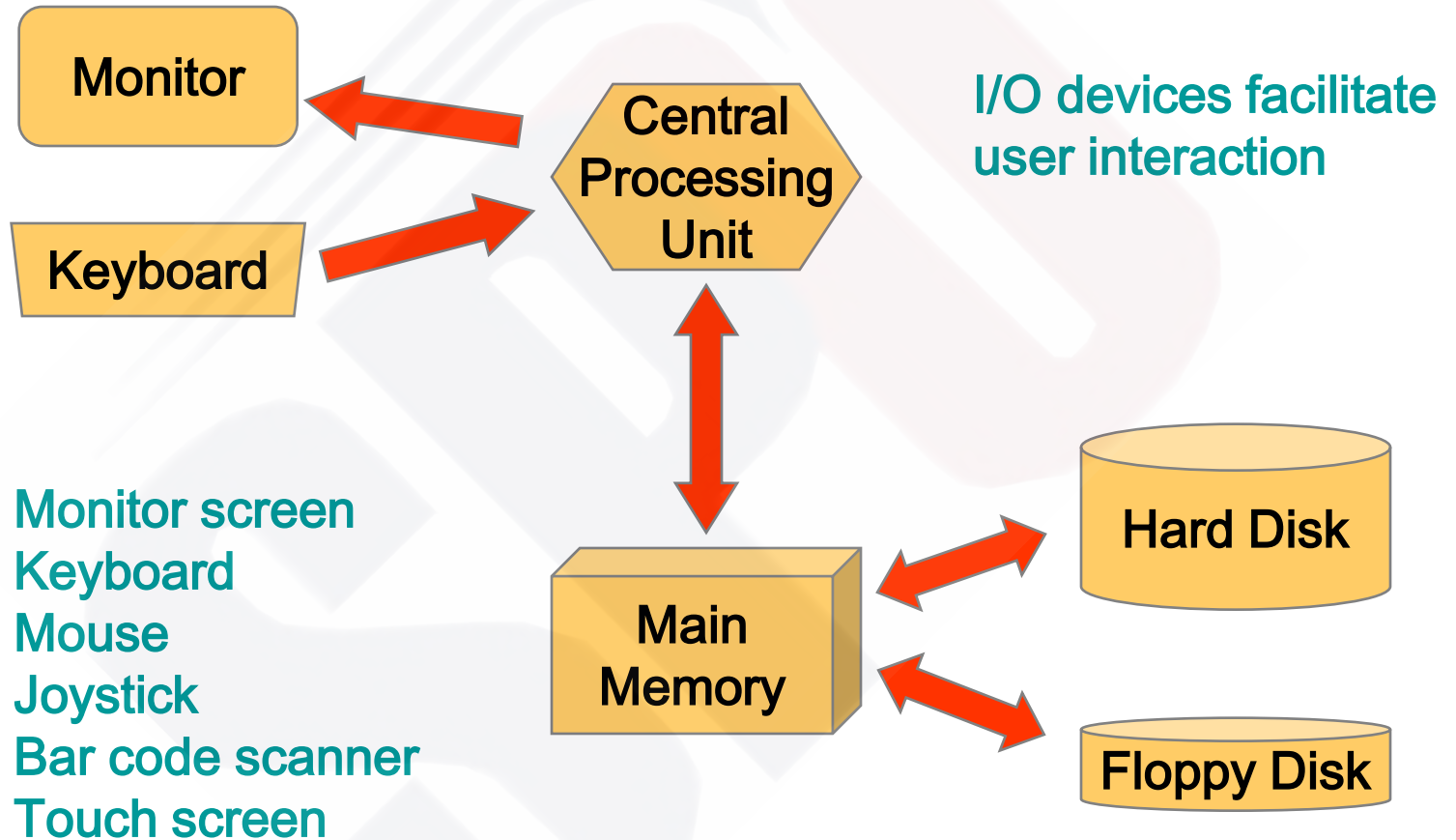
Secondary memory devices provide long-term storage

Information is moved between main memory and secondary memory as needed

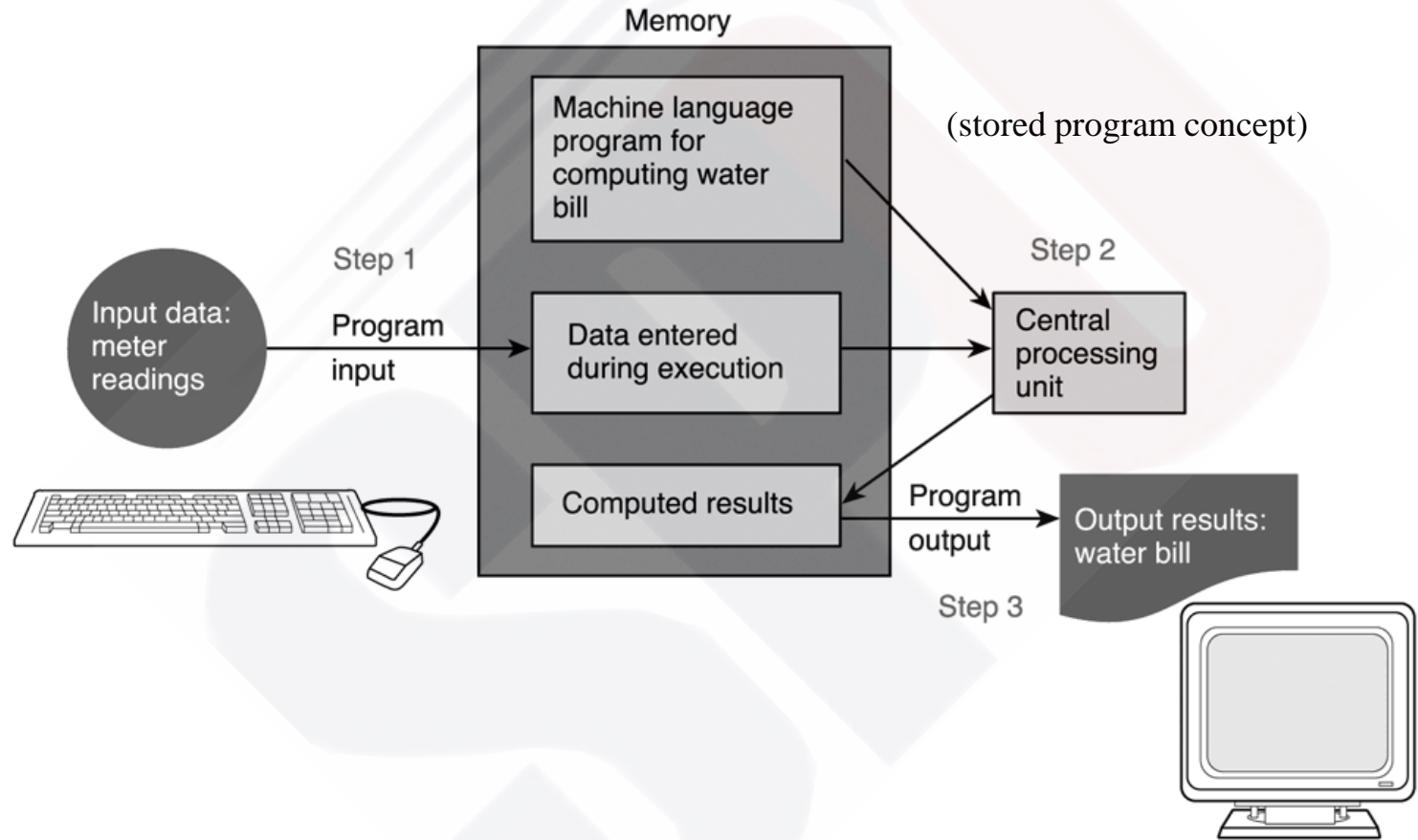
Hard disks
Floppy disks
ZIP disks
Writable CDs
Writable DVDs
Tapes



Input / Output Devices



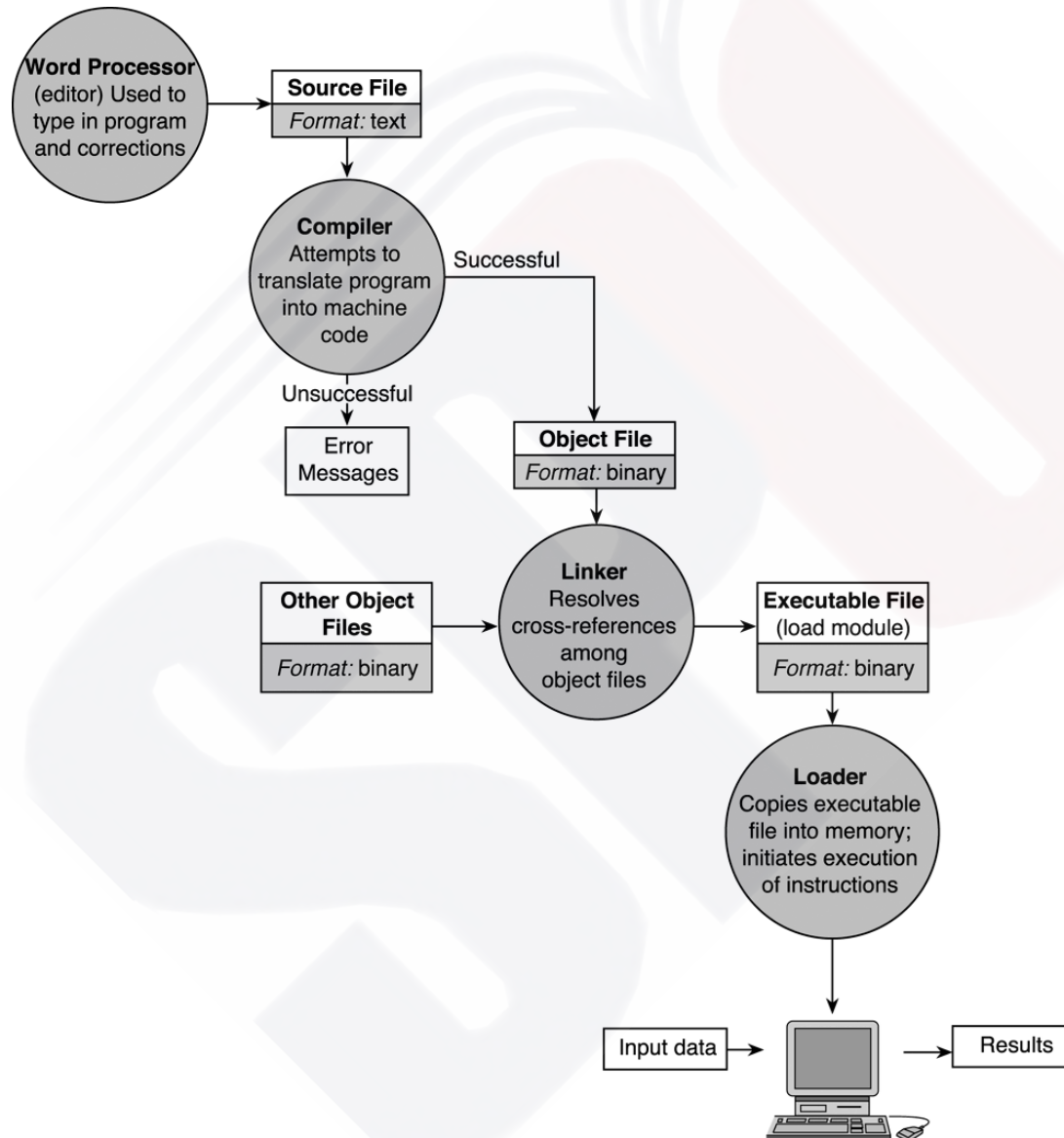
Flow of Information During Program Execution



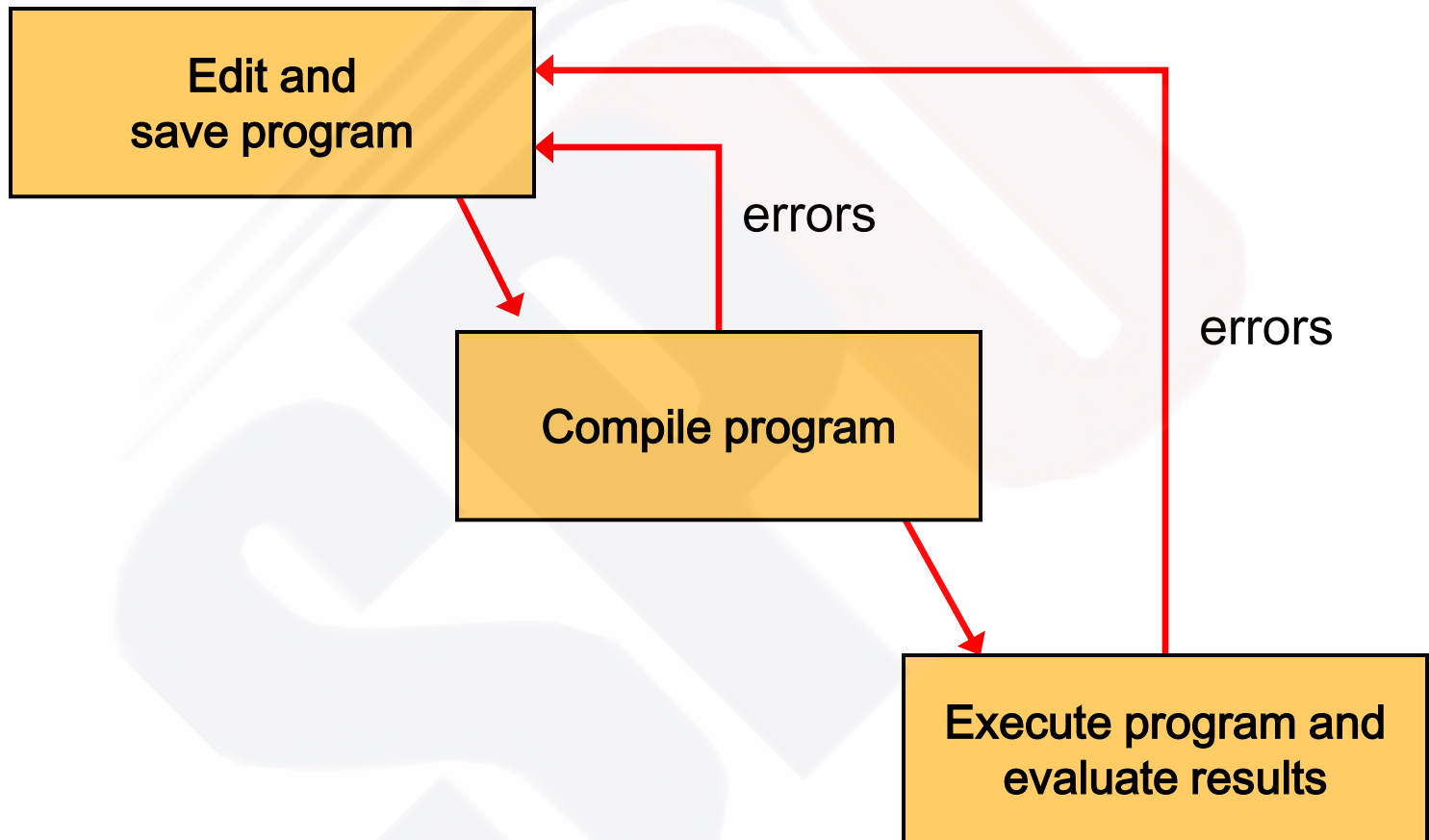
Program Development

- **The mechanics of developing a program include several activities**
 - **writing the program in a specific programming language**
 - **translating the program into a form that the computer can execute**
 - **investigating and fixing various types of errors that can occur**
- **Software tools can be used to help with all parts of this process**

Entering, Translating, and Running a High-Level Language Program



Basic Program Development



Errors (a.k.a., bugs)

- A program can have three types of errors
- The compiler will find syntax errors and other basic problems (*compile-time errors*)
 - If compile-time errors exist, an executable version of the program is not created
- A problem can occur during program execution, such as trying to divide by zero, which causes a program to terminate abnormally (*run-time errors*)
- A program may run, but produce incorrect results, perhaps using an incorrect formula (*logical errors*)

Programming Languages

- Each type of CPU executes only a particular *machine language*
- A program must be translated into machine language before it can be executed
- A *compiler* is a software tool which translates *source code* into a specific target language
- Often, that target language is the machine language for a particular CPU type

Syntax and Semantics

- The *syntax rules* of a language define how we can put together symbols, reserved words, and identifiers to make a valid program
- The *semantics* of a program statement define what that statement means (its purpose or role in a program)
- A program that is syntactically correct is not necessarily logically (semantically) correct
- A program will always do what we tell it to do, not what we meant to tell it to do