

Introduction to Programming

Lecture 1:

Introduction



What We Will Learn

- What is this course?
- Computer organization
 - Hardware
 - Software
- Algorithms & Programming
 - Algorithm
 - Programming Language
- Solving problems



What We Will Learn

- What is this course?
- Computer organization
 - Hardware
 - Software
- Algorithms & Programming
 - Algorithm
 - Programming Language



This Course

- Introduction to Computer & Programming

How to use computers to solve
our problems

- The problems are *computational* problems



This Course (cont'd)

➤ What we learn

- Overall overview of computer organization
- Thinking in algorithmic manner
- A programming language: the **C**

➤ What we don't learn

- In depth computer hardware/software details
- Most advanced algorithms
- System programming using C
- Other programming languages: Java, PHP, ...

CA, OS, ...

Alg, DS, ...

OS, ...

AP, IE, ...



This Course (cont'd)

➤ Programming

- is **not** a **pure** theoretical course (mathematics, ...)
 - Read, read, read, read
- is a **practical** course
 - Read, programming, programming, programming,...

➤ Course materials

- Lecture notes (slides) are in (simple) English
- Available in the course homepage:

`ceit.aut.ac.ir/~bakhshis/c`

- Textbook (C: How to Program 7th Edition 2012) + books:

`\\fileserver\common\Bakhshi\Introduction to Programming`



Grading

➤ Four major parts

- Midterm 25%
- Final 25%
- Homework 35%
- Project 15%

➤ TA Classes

- One or Two classes per week
- More details, Practical aspects, Solving HW
- Homework are not accepted after solutions



Any Question?!

- Is CE a good engineering dep. at university?!
Yes 😊
- Is AUT really a top university?! Yes 😊
- Do I need to learn C?! Yes!!!
- Is internet free at university?! Yes 😊
- Is lunch free?! No 😞
- ...



What We Will Learn

- What is this course?
- Computer organization
 - Hardware
 - Software
- Algorithms & Programming
 - Algorithm
 - Programming Language
- Solving problems



Computers: The *Computing* Machines

➤ Computers classification:

➤ Supercomputers

- Weather forecast, Atomic explosion simulation, ...

➤ Mainframe computers

- The servers in large companies: Google, ...

➤ Midsized computers

- The servers in CE department

➤ Micro computers (also called PC)

- Your laptop

➤ Pocket PCs

- Mobile phones



Computers

- Computers are anywhere, anytime. **Why?**
 - They can solve many different problems. **How?**
- Computers are *programmable machines* capable of performing calculations (computation)
 - Changing program leads to different operation
- *Special-purpose* machines
 - Calculators, game-playing machines, ...
- *General-purpose* computers
 - Personal computers, notebooks, ...



Data Units

- Computers are **digital** machines
- Data processed or stored in computer is represented as two-state values
 - either 1 or 0 - **BI**nary digiTs (BIT)
 - 1 Byte = 8 bits
 - 1 kilobyte (KB) = 1024 bytes
 - 1 megabyte (MB) = 1024 kilobyte
 - 1 gigabyte (GB) = 1024 megabyte



Data Representation/Coding

- How to represent our data by 0-1?
- In other word, there are some 0 and 1 in the computer, what is the meaning?

Coding (Representation Standards)

- Major (common) representations (coding)
 - Integer numbers: 1, 1000, -123, 0, ...
 - Floating point numbers: 1.1, 11.232, -12.23, ...
 - Characters: 'A', 'ب', '@', ...



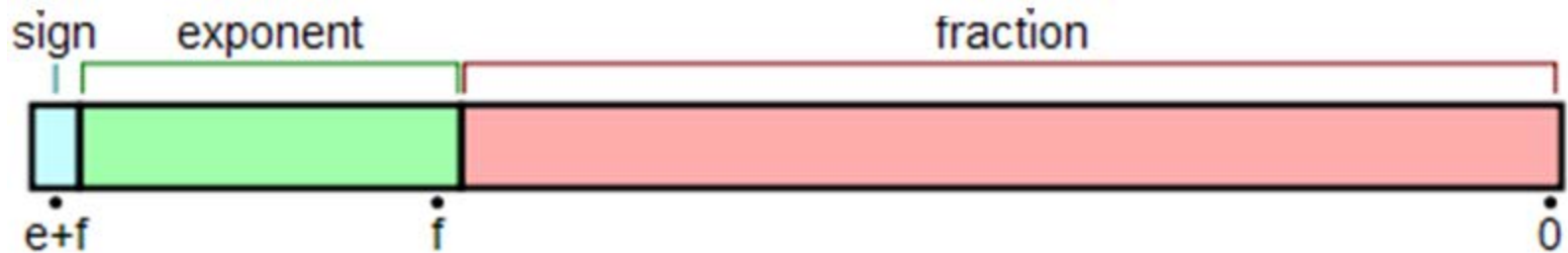
Integer Number Coding

- There are different representations
 - You will learn them (in details) in other courses (e.g. Computer Architecture)
- One the (simple) coding is sing-magnitude coding
 - If we have n bit for coding integers
 - The left bit (the MSB): **sign**
 - $n-1$ bits: **magnitude**
 - E.g., 8 bit for coding
 - 4 → 00000100
 - -4 → 10000100



Floating Point Number Coding

- Usually, this coding pattern



- You will see all details in other courses
- Two precisions
 - Single precision
 - exponent: 8 bit, fraction: 23 bit
 - Double precision:
 - exponent: 11 bit, fraction: 52 bit



Character Coding

➤ Common character encoding: **ASCII**

➤ Character	ASCII Code	Binary (8 bit)
➤ '0'	48	00110000
➤ 'A'	65	01000001

- 8 bits can represent 256 characters; but,
 - There are so many characters (Farsi, Arabic, ...)
 - Solution: UTF (Variable length coding)
 - 0xxxxxxx: 1 byte code
 - 110xxxxx 10xxxxxx: 2 byte code
 - ...



Computer Organization

➤ Major Components

➤ Hardware

- *Physical devices* that are *wired* and performs *basic* operations

➤ Software

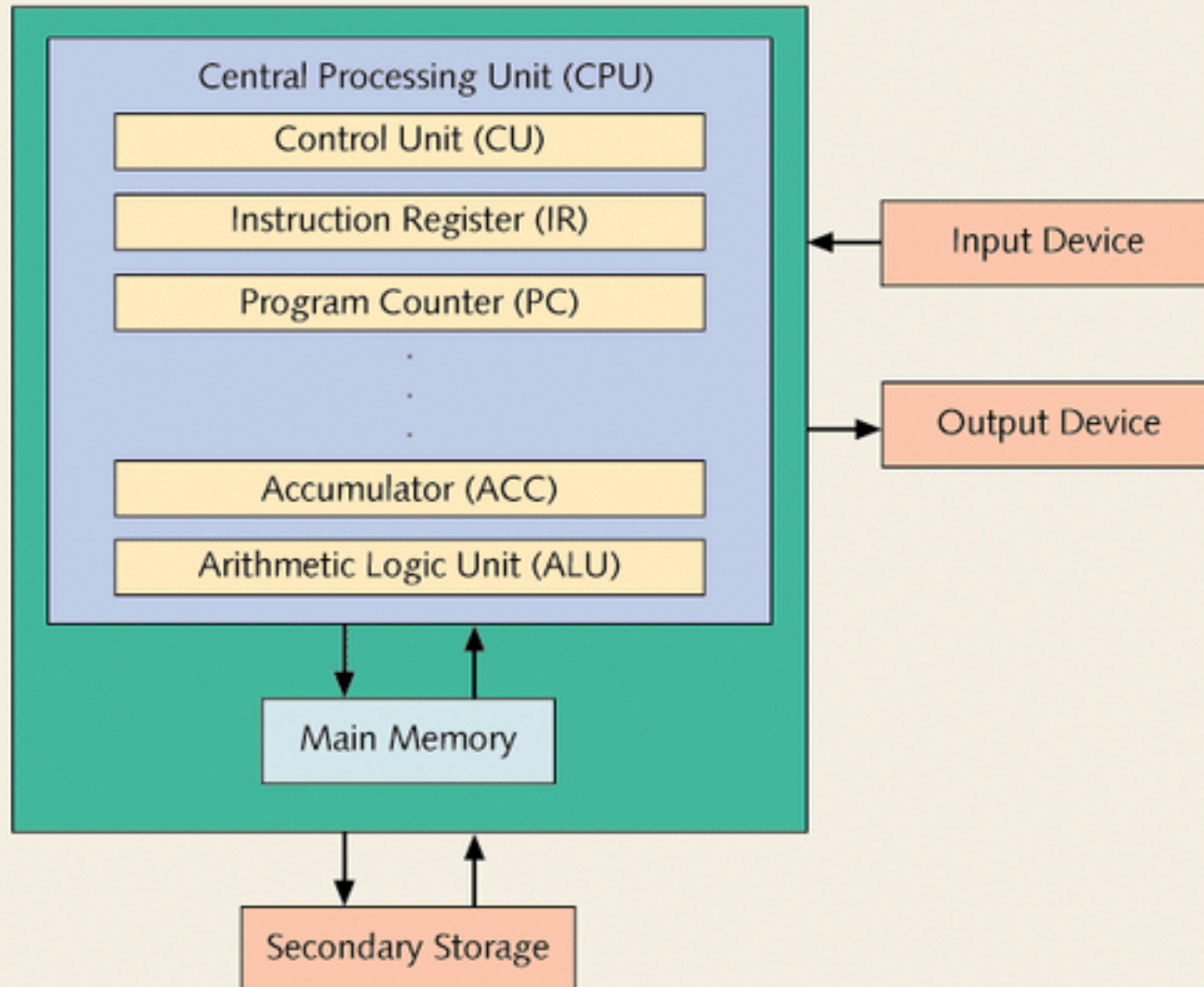
- Set of *programs* that run on the hardware

➤ Hardware

- CPU (Central Processing Unit)
- Main Memory
- Secondary Storage
- Input/output



Computer Organization



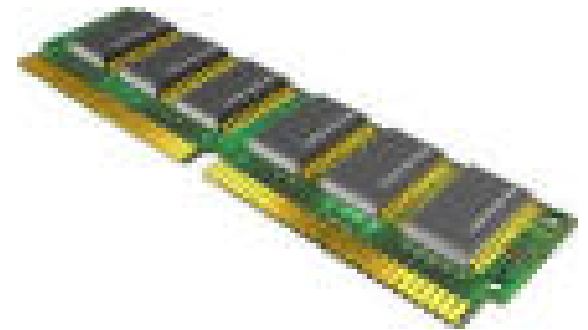
Computer Organization: CPU

- Control Unit
 - Controls system operation
- ALU (Arithmetic Logic Unit)
 - Performs mathematic calculations
 - Makes decision based on conditions
- Special Floating Point processors
- Set of working area: **Registers**
- Operation and operands are required
 - Which are provided by instructions in the **main** memory



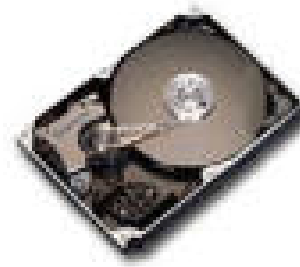
Computer Organization: Main Memory

- Ordered sequence of cells (memory cells)
- Directly connected to CPU
- All programs must be in main memory before execution
- When power is turned off, Main memory is cleared



Computer Organization: Secondary Storage

- Provides permanent storage for information
- Examples of secondary storages:
 - Hard Disks
 - Floppy Disks
 - Flash/Cool/USB Disks
 - CD/DVD
 - Tapes



Computer Organization: Input Devices

- Devices that feed data and programs into computers
- Examples:
 - Keyboard
 - Mouse
 - Network Interface Card
 - Joystick
 - Microphone



Computer Organization: Output Devices

➤ Devices that computer uses to generate results/outputs

➤ Examples:

- Printer
- Monitor
- Speaker
- Network Interface Card



Computer Organization: Software

➤ What can do the Hardware?

- No useful operation, if there isn't any software

➤ Software

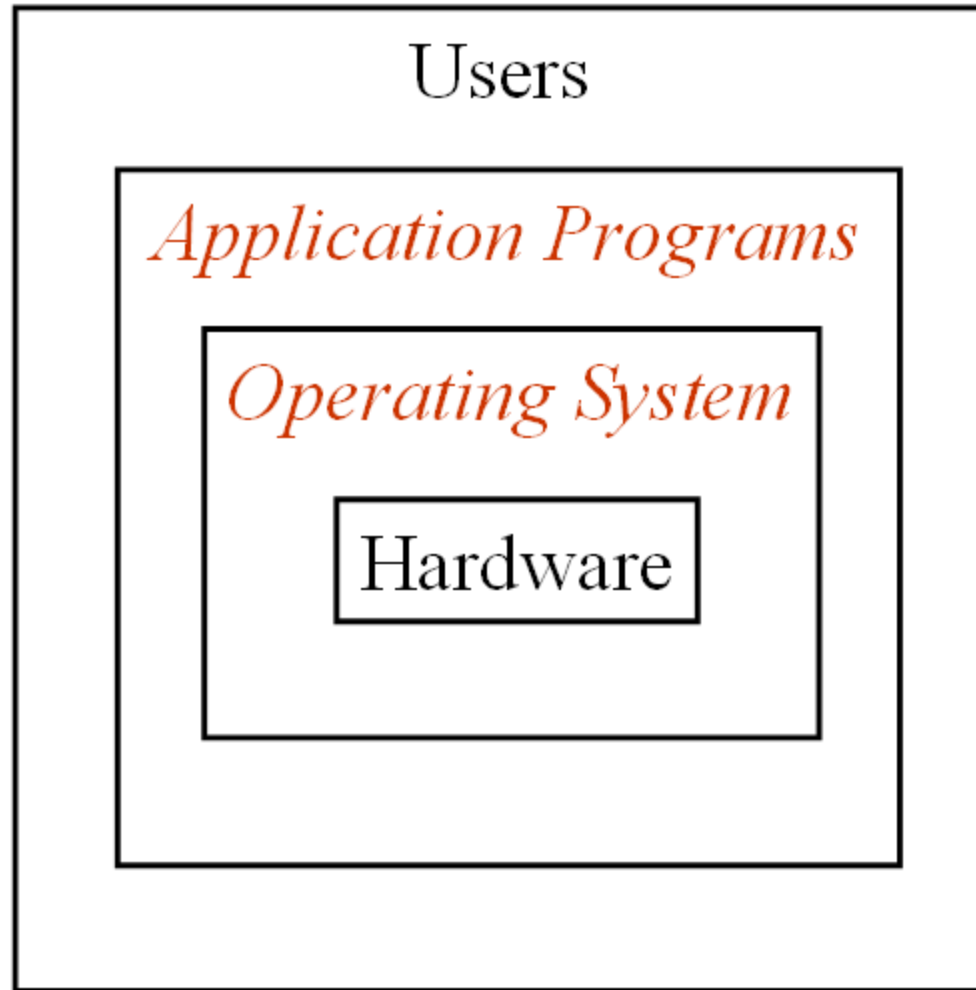
- Computer programs which are designed for a specific task

➤ Major Software

- Operating System
- Libraries
- Applications (**Program**)



Computer Organization



Computer Organization: OS

➤ OS

- *Manages* the hardware
- Application programmers can easily use them
 - *Without knowing the HW details*

➤ Common operating systems

- Windows XP/Vista/8/10, Linux, Unix, ...



Computer Organization: Libraries

- The libraries provide the most common functionalities
- In mathematic programs
 - $\sin(x)$, $\cos(x)$, matrix multiplication/inversion
- In graphical programs
 - Draw a line/cycle, set color, new window
- In multimedia programs
 - Open/close files, jump, ...



Computer Organization: Applications

- An application program
 - Users use them to do some specific things
 - *Without knowing the details of the computer*
- Common application programs
 - *Word, Internet Explorer, FireFox, Messengers*
- Common applications in mathematic:
 - *Matlab, Mathematica, Maple, GAMS, AIMMS*



How to be general purpose machine?

- Hardware is simple & general purpose
 - Only a small set of basic instructions are implemented by hardware
- Complex instructions are developed in software
- Software is translated to the basic instructions
 - It can run by the hardware
- This is the way that we “*program*” computers



Programming Execution Phases

- Program is loaded from secondary storage to main memory by OS
- OS gives the control to the program
- Instructions run
- Required inputs are got from input device & saved in main memory & used by CPU
- Result is saved in main/secondary memory or sent to output devices



Instruction Execution Steps

- Basic steps in running instructions
- Read instruction from main memory: **fetch**
 - `"000110...011"`
- **Decode** the instruction
 - `add 1 to memory location XYZ save result in ABC`
- Get required **operands** from main memory
 - Read value of location XYZ to **temp1**
- **Run** the instruction
 - `temp2 = temp1 + 1`
- Save the **result**
 - Write temp2 in memory location ABC



Reference

- **Reading Assignment:** Chapter 1 and Appendix C of “C How to Program”
- Learn more about computer hardware
 - “How Computer Works”



What We Will Learn

- What is this course?
- Computer organization
 - Hardware
 - Software
- **Algorithms & Programming**
 - Algorithm
 - Programming Language
- Solving problems



Algorithms

- Hardware do basic operations
- We want to solve a real problem by computers
 - Take average, Sort, Painting, Web, Multimedia, ...
- We need a solution that
 - Specifies how the real (complex) problem should be solved *step-by-step* using the basic operations
- The solution is the “Algorithm” of the problem



Algorithms (cont'd)

➤ Common Sense (in computer science):

- 1) An **abstract** strategy to solve a problem
- 2) The way to do some things

➤ Formal Definition:

*“An algorithm is a **finite** list of **well-defined** instructions for **accomplishing some task** that, given an **initial state**, will **proceed** through a well-defined series of successive states, possibly eventually **terminating** in an end-state”*



Algorithms: Examples

- Finding Common Divisor
- Finding 2 largest element in a set
- Finding shortest path in a graph
- Searching in a sorted array
- Sorting a set
- Combining 2 sorted set in a sorted set
- Solving an equation
- Compression algorithms
- Cryptography algorithms
-



Algorithms: Description

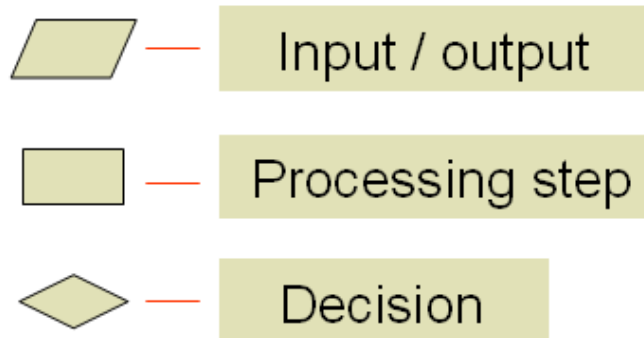
- Algorithms are the problem solving steps/strategy in our mind!!!
- How can we document it (don't forget it)?
- How can we explain/teach it to others peoples?
- How can we explain it to computers?
- We need some methods to describe algorithms!
 - Flow chart
 - Pseudo-codes
 - Codes/Programs



Algorithms: Description (cont'd)

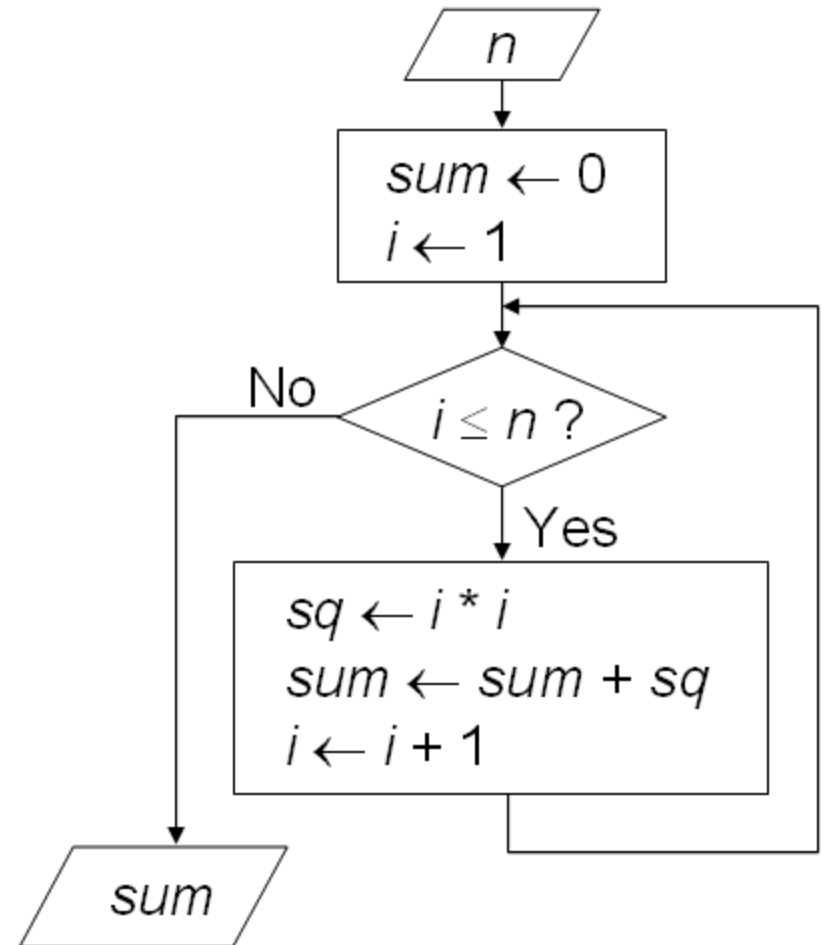
➤ Flowcharts:

➤ Schematic representation



➤ Example:

calculate $1^2 + 2^2 + \dots + n^2$



Algorithms: Description (cont'd)

➤ Pseudo-code

- A sequence of **English** and mathematical statements

Algorithm: calculate $1^2 + 2^2 + \dots + n^2$

Input: n

Output: sum

sum \leftarrow 0

$i \leftarrow 1$

Repeat the following three steps while $i \leq n$:

sq $\leftarrow i * i$

sum \leftarrow sum + sq

$i \leftarrow i + 1$



Algorithms: Description (cont'd)

- Flowcharts and Pseudo-code are for humans not for computer
 - Computer **cannot** run them
- What can computer run?
 - Instructions in main memory
 - The instructions are in “**011100001...**” format
 - To use computers
 - We should describe your algorithm in “01” format
 - **?????** ☹ ☹



What We Will Learn

- What is this course?
- Computer organization
 - Hardware
 - Software
- **Algorithms & Programming**
 - Algorithm
 - **Programming Language**
- Solving problems



Programming Language

- Programming languages are the tools to describe your algorithms for computers
 - Software is developed by programming languages
- When algorithm is described with a programming language
 - It **cannot** be run on computer **directly** 😞
 - There are some other programs that **translate** the programming language to “010...”
 - The output “0101...” **can** run on computers 😊😊
- Why human languages cannot be used?



Programming Language: Machine Level

- Computer's native language
- What is saved in the main memory
- The processor architecture specifies the format of 01s, **machine depended**
- Example
 - Add two numbers: 00100111 1010 0101
- Completely incomprehensible to most people



Programming Language: Assembly

- Programming based on mnemonics
- There are **one-to-one mapping** between machine language and assembly mnemonics

Assembly Language	Machine Language
LOAD	100100
STOR	100010
MULT	100110
ADD	100101
SUB	100011

➤ Example

`load r1, [4000] ; read content of address 4000`

`add r1, 1 ; add 1 to CPU register r1`

`store [5000], r1 ; save the result in location 5000`



Programming Language: High Level

- Easy for programming, English-like keywords
 - More similar to natural languages
- There isn't one-to-one relation between high level statements and machine level statements
- Example: C, C++, Pascal, Java, PHP, Python
- Example:

```
int xyz;  
int abc;  
abc = xyz + 1;
```



Translation of High Level Languages

➤ Two types of translators

- Interpreter (مفسر)
- Compiler (مترجم)

➤ Interpreter

- Checks and runs program lines one-by-one
- Easy, slow, and we need the interpreter

➤ Compiler

- Check all lines, creates executable output file
- Fast and Stand alone program



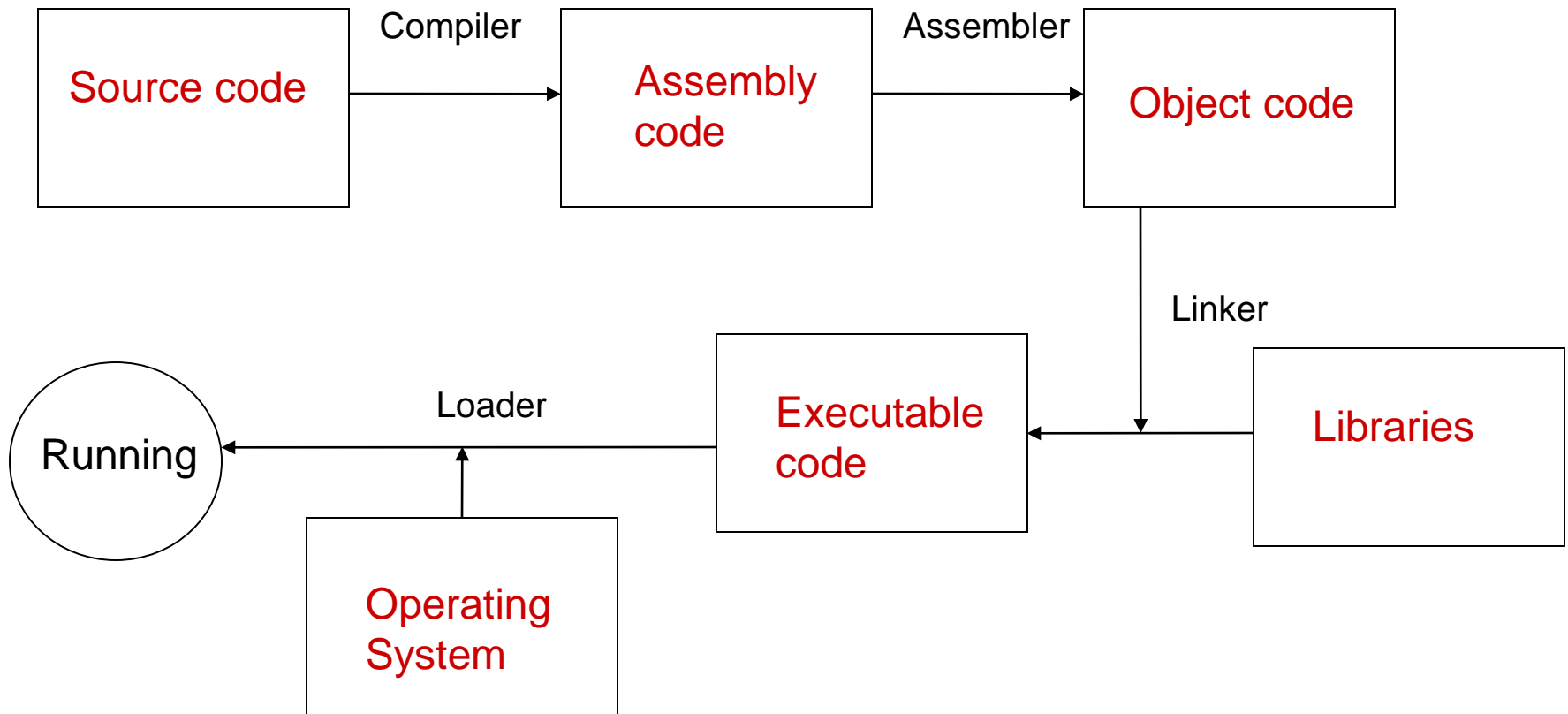
Compiler

➤ Compiler

- A **set** computer programs do the **Compilation**
 - **Preprocessor**: Prepare file for compiler
 - **Compiler**: Create assembly code
 - **Assembler**: Convert assembly code to binary code
 - **Linker**: Collect all required binary files (from libraries) into a single loadable file
 - Each language has its own compiler
- Usually compiler do all above steps, you just compile the file and get a executable file



Building & Running Program



What We Will Learn

- What is this course?
- Computer organization
 - Hardware
 - Software
- Algorithms & Programming
 - Algorithm
 - Programming Language
- Solving problems *using computers*



Solving Problems

- How to solve problems using computers
 - Develop a **program** for it
- Steps
 - Analysis: Input, output
 - Algorithm
 - Coding
 - Compile → program
 - Execution → test
 - Documentation



Solving Problems: Analysis

- Problem solving process consists of

Input → Algorithm → Output

- Determine what information is available as the input to your algorithm
- Determine what information is desired as the output from your algorithm
- What needs to be done on the input to produce the output? **Algorithm**



Solving Problems: Algorithm

- Determine a series of steps that will transform the input data into the output results
- Find all the **special cases** that the must be handled
- If necessary modify or redesign your series of steps so that all special cases are handled
- Verify your algorithm



Solving Problems: Coding

- Describe your algorithm by a programming language
- You must code exactly in the programming language **syntax**
- Compiler itself is a program it isn't a human
 - It is not intelligent
 - It just does the steps of the compiling algorithm
 - It does not understand what do you mean!!!



Solving Program: Execution

- Compiler generated the executable file
- Run the executable code
 - First try to use simple
 - Give the input
 - Get results
 - Then try larger and complex inputs



Errors in Solving Problems

- Compile / Syntax error: Compiler does not recognize your code
- Link error: Linker cannot find the required libraries
- Runtime error: Program does not run correctly
 - Example: Division by zero
- Logical Error: Program does not produce the expected result
 - It is called **bug**
 - No one (compiler, assembler) except debugger can help you ☹
- Why error?
 - You do not understand and analysis the problem correctly
 - You do not develop a right algorithm for the problem
 - You have mistakes in your coding

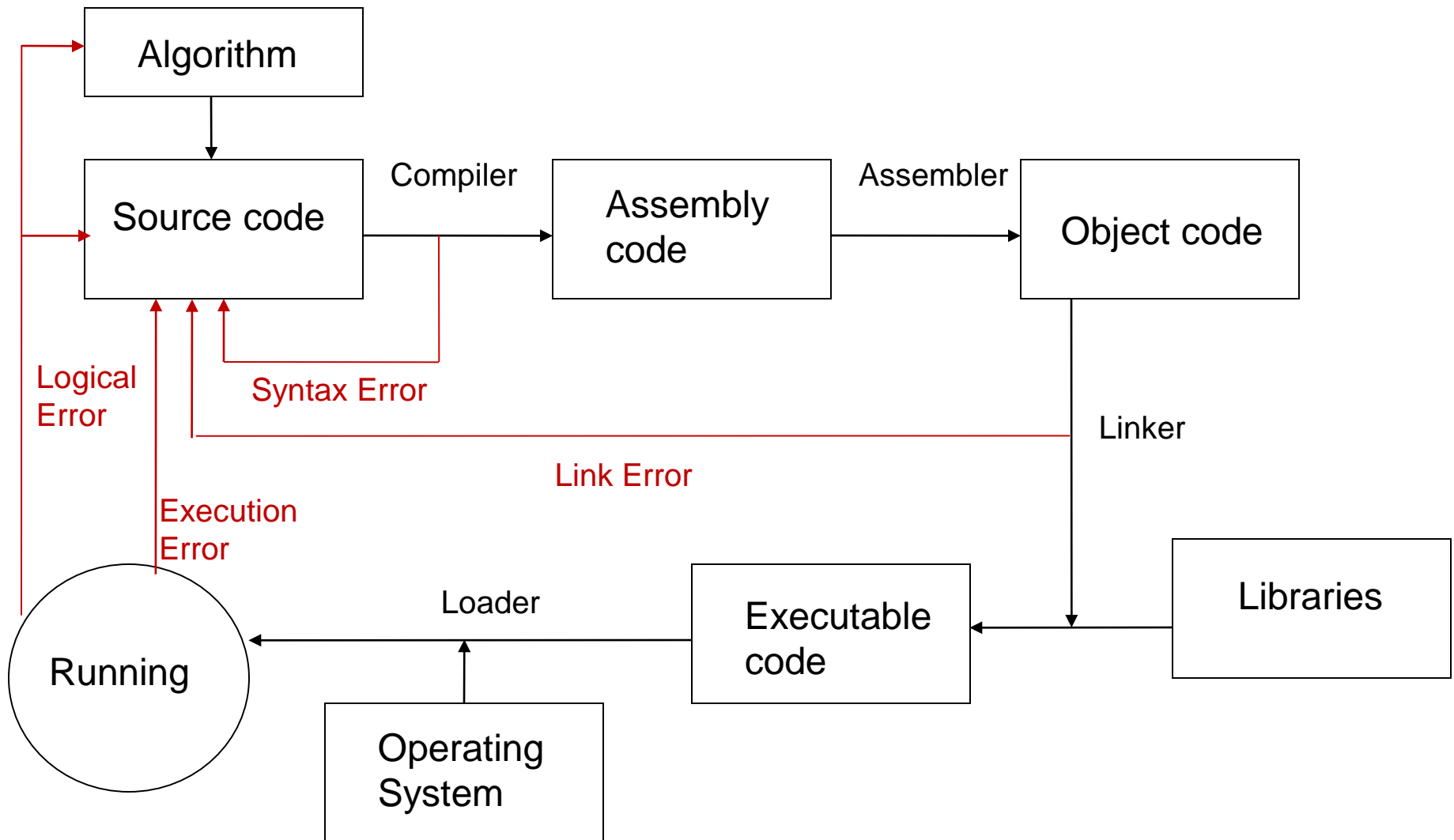


Debugging

- The process of resolving the errors
 - Example: A program to divide two numbers
- Compile/Syntax error
 - Compiler tells where it is → check syntax
- Link error
 - Compiler tells what it is → check syntax & libraries
- Run time error
 - Try to find it → use debugger to run step-by-step, print debug messages
 - Check syntax & semantic of the line
- Logical error
 - Try to find it → use debugger to run step-by-step, print debug messages
 - Check syntax & semantic of program
 - Revise the algorithm



Building & Running Program



Desired Features of Programs

- Integrity (درستی)
 - Correctly solve the problem
- Clarity (وضوح)
 - Easy to read
- Simplicity (سادگی)
 - Easy to understand
- Efficiency (کارایی)
 - Speed and memory
- Modularity (پیمانه‌ای)
 - Break down of a large task
- Generality (عمومیت)
 - Tunable by input as much as possible



Summary

- Computer organization
 - Hardware and Software
- Algorithm & Program
 - What is the difference between them
- How to solve a problem using computer
 - Steps
- Errors in problem solving
- What is the next: Design algorithm → Program



Reference

- **Reading Assignment:** Chapter 1 of “C How to Program”

