Lecture 1
Introduction and Brief History
Outline

• Review of Course Materials
• Grading Policy
• An Overview of Computer
  – Computer Components
    • Hardware
    • Software
• Introduction to Programming
  – Programming Paradigm
  – Machine Languages
  – Assembly Languages
  – High-Level Languages
• History of C/C++
• Typical C Program Development Environment
  – Compilation Process
Review of Course Materials

- Computer number format
- Data Types, Variables, Operators, Input/Output
- Algorithm and Pseudo code
- Functions
- Strings and Pointers
- Arrays, Structures
- Files
- Object-Oriented Programming
Core reference

1. **C How to Program 8th Edition**, by Paul Deitel, Harvey Deitel, Publisher: Pearson; March 8, 2015.


Grading policy *Tentative*

- Midterm exam: **5 pts** (15 Azar 1397)
- Final exam: **6 pts** (23 Dey 1397)
- Assignments: **4 pts**
- Quizzes: **2 pts**
- Final Programming Project: **3 pts**
- Extra (bonus)
  - Programming Contest: **+1 pts**
  - Extra Assignments: **+1 pts**
  - Regular Presence at Class: **+0.5 pts**
  - Class Activity: **up to +1 pts**
  - Some other bonus chances... ? pts
• **Instructor**
  - Vahid Khodabakhshi
  - vkhodabakhshi@ce.sharif.edu

• **Website**
TAs

1. **Ali Najimi [Head]**, najimi56@gmail.com
2. Mohamadreza Karegar, karegar@gmail.com
3. Mohsen Naghipourfar, mn7697np@gmail.com
4. Poyan Shirzadian, pshirzadian26@gmail.com
5. Amirhossein Khajepour, amirhossein_kh@ymail.com
6. Saba Torabi Ardekani, torabi.saba16@yahoo.com
7. Shayesteh Arasti, shayesteharasti@gmail.com
8. Arvin Azarmina, arvinazarmina@gmail.com
9. Alireza Mahmoodi, mahmoodia1376@gmail.com
10. Sajed Karimi, karimisajed1378@gmail.com
11. Mohamadhosein Shahmoradi, mh.shahmoradi@ie.sharif.edu
12. Kamyar Maroufi, kmaroufi76@gmail.com
13. Mojtaba Farsian, farsian.mojtaba@gmail.com
14. Mohamadreza Ahmadkhaniha, mr.ahmadkhaniha@gmail.com
15. Mohamad Alizadeh, alizadehmohamd77@gmail.com
Homework Submission

Submit your homework on course’s page on Quera.ir

Register to Quera:
Invitation link has been sent to your university e-mail
If not, contact the head TAs
An Overview of Computer

• Computer:
  – **Programmable** general purpose machine
    • Can not do anything without a *program*
  – Receives Input
    • Letters, Numbers, Images
  – Processes and Stores input
  – Provides Output in a useful format
Computer Components

• Hardware
  – The physical parts or components of computer such as monitor, keyboard, hard disk, mouse, etc.

• Software
  – Set of instructions you write to command computers to perform actions on hardware
Overview of Computer Hardware

- Secondary storage
- Ordered sequence of storage location (memory cell)

Input devices → Main memory → Central processing unit → Output devices

Memory

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-27.2</td>
</tr>
<tr>
<td>1</td>
<td>354</td>
</tr>
<tr>
<td>2</td>
<td>0.005</td>
</tr>
<tr>
<td>3</td>
<td>-26</td>
</tr>
<tr>
<td>4</td>
<td>H</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>998</td>
<td>X</td>
</tr>
<tr>
<td>999</td>
<td>75.62</td>
</tr>
</tbody>
</table>

Ordered sequence of storage location (memory cell)
Logical Units

- Regardless of differences in shape (physical appearance), virtually every computer may be envisioned as divided into six logical units or sections:
  1. Input unit
  2. Output unit
  3. Memory unit
  4. Arithmetic and logic unit (ALU)
  5. Central processing unit (CPU)
  6. Secondary storage unit
1. Input unit

• **Receiving** section obtains information (data and computer programs) from **input devices** and places it at the disposal of the other units so that it can be processed

• examples: *keyboards and mouse*
2. Output unit

- This **shipping** section takes information that the computer has processed and places it on various output devices to make it available for use outside the computer.
3. Memory Unit

- Rapid-access, relatively low-capacity (*in comparison with secondary storages*)

- **Warehouse** section retains information that has been entered through the input unit, making it immediately available for processing when needed

- The memory unit also retains processed information until it can be placed on output devices by the output unit

- Information in the memory unit is **volatile**—it’s typically lost when the computer’s power is turned off

- The memory unit is often called either **memory or primary memory (RAM)**
4. Arithmetic and logic unit (ALU)

- **Manufacturing** section performs calculations
  - addition, subtraction, multiplication and division
  - It also contains the decision mechanisms that allow the computer, for example, to compare two items from the memory unit to determine whether they’re equal

- The ALU is usually implemented as part of the next logical unit, the CPU
5. Central processing unit (CPU)

- **Administrative** section coordinates and supervises the operation of the other sections
  - tells the input unit when to read information into the memory unit
  - tells the ALU when information from the memory unit should be used in calculations
  - tells the output unit when to send information from the memory unit to certain output devices

- **Multiprocessors** computers have multiple CPUs and, hence, can perform many operations simultaneously
  - A multi-core processor implements multiprocessing on a single integrated circuit chip
    - Dual-core processor has two CPUs
    - Quad-core processor has four CPUs
6. Secondary Storage Unit

• Long-term

• High-capacity warehousing section

• Programs or data not actively being used by the other units normally are placed on secondary storage devices until they’re again needed

• Information on secondary storage devices is said to be persistent—it is preserved even when the computer’s power is turned off
  – Hard drives, CDs, DVDs and flash drives
History - First Generation Computers

- Mid-1940s
- Used vacuum tubes
- Huge and complex

The ENIAC, weighing 30 tons, using 200 kilowatts of electric power and consisting of 18,000 vacuum tubes
History - Second Generation Computers

- 1955 – 1960
- The invention of Transistor
- The era of miniaturization begins.
History - Third Generation Computers

• 1960s
• The Integrated Circuits, also known as microchips
• Silicon chips containing multiple transistors
History - Fourth Generation Computers

• 1971 – present
• large-scale integration or LSI
  – 1000 devices per chip)
• very large-scale integration or VLSI
  – 10000 devices per chip)
• ...
Overview of Computer Software

• **Operating System (OS)**
  – the collection of computer programs that control the interaction of the user and the computer hardware.
  – E.g. Windows, Unix

• **Application Software**
  – Programs developed to assist a computer user in accomplishing specific tasks.
  – E.g. Microsoft Word

• **In order to create new application software, we need to write lists of instruction (program) to the computer to execute**
Programming Language

• The defining feature of modern computers which distinguishes them from all other machines is that they can be programmed.

• Programming is a process for instructing a computer to do something for you with the help of a Programming Language.

• A programming language contains instructions for the computer to perform a specific action or a specific task:
  – Display “I like programming”
  – Display the current time
Programming Language

• Programming Language is a **Formal Language** used to communicate to a computer
  – Very specific (one word means one thing – **context free**) since to 'talk' to a computer; to instruct a computer; our commands must be 100% clear and correct

• The description of a programming language is usually split into the two components of **syntax** (form) and **semantics** (meaning)

• A **programming paradigm** is a fundamental style of computer programming:
  – Functional: tell what to do but not how (sum [1...10])
  – Imperative: describing step by step
  – Object-Oriented and Logical Programming
Programming Language

• **Special-purpose**: is design for a particular type of application
  – Structured Query Language (SQL)

• **General-purpose**: can be used to obtain solutions for many types of problems.
  – 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
    11010001
    01001100

• 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 (different machine language for different type of CPU) – not portable
Assembly Language

- Enables machine code to be represented in words and numbers.

- Example of a program in assembly language:
  LOAD A, 9999
  LOAD B, 8282
  SUB B
  MOV C, A

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

- Cannot be processed directly by a computer, must be converted to machine language using assemblers.

- Processor and Architecture dependent – not portable.
High-Level Language

- **Machine independent** programming language that combines algebraic expression and English words
- Example:
  \[ c = b - a \]
- Processor **independent** - the same code can be run on different processors
- Examples: Basic, Fortran, Pascal, Cobol, C, C++, Java
- High level language needs to be translated (compiled) to machine code by a program called **compiler** so that it can be **executed** by the processor
Programming Language Abstraction

- C
- C++
- Java
- Fortran
- PL1...

Abstraction

- High-Level Language
- Assembly Language
- Machine Language
- Hardware
C History

- **BCPL**, 1967, Martin Richards
  - writing operating-systems software and compiler
- **B**, 1969, Ken Thomson
  - based on BCPL
- **C**, 1972, Dennis Ritchie
  - based on BCPL and B
  - at Bell Laboratories
  - originally implemented on a DEC PDP-11
C History

• In 1983, the American National Standards Institute (ANSI) established a committee to provide a modern, comprehensive definition of C. The resulting definition, the ANSI standard, or ANSI C, was completed late 1988
  – updated in 1999

• Because C is a **hardware-independent**, widely available language, applications written in C can run with little or no modifications on a wide range of different computer systems
  – Portable programs
C – An Imperative Language

• C is a highly _imperative formal_ language
  – We must tell it _exactly how_ to do what
  – the means and functions to use
  – which _libraries_ to use
  – when to add a new line
  – when an instruction is finished
  – in short: everything and anything...

• _filename.c_
C++ Programming Language

• early 1980s, Bjarne Stroustrup
  – at Bell Laboratoty
  – C++ a superset of C
  – object-oriented programming
    • Objects are essentially reusable software components that model items in the real world

• `filename.c`
• `filename.cpp`
Typical C Program Development Environment

• C systems generally consist of several parts:
  – a program development environment
  – the language
  – the C Standard Library

• C programs typically go through six phases to be executed:
  – edit, preprocess, compile, link, load and execute
Introduction and Brief History – Lecture 1

Phase 1: Programmer creates program in the editor and stores it on disk.
Phase 2: Preprocessor program processes the code.
Phase 3: Compiler creates object code and stores it on disk.
Phase 4: Linker links the object code with the libraries, creates an executable file and stores it on disk.
Phase 5: Loader puts program in memory.
Phase 6: CPU takes each instruction and executes it, possibly storing new data values as the program executes.
Microsoft Visual Studio

- Editing a file with an **editor** program
- Integrated Development Environment (IDE)
Dev-C++

```c
#include <stdio.h>

int main(int argc)
{
    printf("I like programming in C\n");
    return 0;
}
```

I like programming in C
Press any key to continue . . .
Code::Blocks
Preprocessor And compiler

• A preprocessor program executes automatically before the compiler’s translation phase begins
  – The C preprocessor obeys special commands called **preprocessor directives**, which indicate that certain manipulations are to be performed on the program before compilation

• The compiler translates the C program into machine language-code (**object code**)
Linking, Loading And Execution

• C programs typically contain references to functions defined elsewhere, such as in the standard libraries or in the private libraries
  – A linker links the object code with the code for the missing functions to produce an executable image

• Before a program can be executed, the program must first be placed in memory
  – This is done by the loader, which takes the executable image from disk and transfers it to memory
  – Additional components from shared libraries that support the program are also loaded

• Finally, the computer, under the control of its CPU, executes the program one instruction at a time
Compile log
Common Problems of Programming

• **Usability**
  – Your program is too complicated or too simple to be useful to most people

• **Maintainability**
  – Other people, or yourself at a later time can't easily understand the programming behind your program. This means your project won't grow and become all it's capable of being
Summary

• Computer Components
  – Hardware
    • Logical Computer Organization: Input unit, Output unit, Memory unit, ALU, CPU, Secondary storage unit
    • Generations Of Computer Hardware: vacuum tube, transistor, IC, LSI, VLSI
  – Software
    • Operating System
    • Application Software

• Programming Languages
  – programming paradigm: Functional, Imperative, Object-Oriented, Logical
  – Machine Languages: language of processor; represented by 0 and 1
  – Assembly Languages: represented in words and numbers
  – High-Level Languages: machine independent

• History Of C/C++ : based on BCPL, B; imperative language

• Typical C Program Development Environment
  – Compilation Process: edit, preprocess, compile, link, load and execute