CC-112 Programming Fundamentals

Introduction to Computers

Nazar Khan
Department of Computer Science
University of the Punjab

Introduction

- Computers are hardware – physical objects that run on electricity.
  - keyboard
  - screen
  - mouse
  - hard disks
  - memory
  - CD/DVD drives
  - processing units
- They are controlled by software.
- We will learn how to command computers to perform actions and make decisions.

Hardware

- Computers perform calculations and make logical decisions phenomenally faster than human beings.
- Billions of calculations/second – more than a human can perform in a lifetime.
- Supercomputers perform thousands of trillions instructions/second.
- Computing hardware keeps getting faster and cheaper.
- Moore's law: In about every two years, computers become twice as better – inexpensively.

Computer Organization

- Any computer can be divided into 6 logical units.
  1. Input unit
  2. Output unit
  3. Memory unit
  4. Arithmetic and logic unit (ALU)
  5. Central processing unit (CPU)
  6. Secondary storage unit

Input unit

- Keyboards, touch screens and mouse devices.
- Voice commands, scanning images, barcodes and QRcodes.
- Reading from storage devices such as hard drives, DVD drives, and USB flash drives.
- Video from a webcam or YouTube.
- An e-book or PDF from the Internet.
- Position data from GPS device, and motion and orientation information from an accelerometer.

Output unit

- Screens (including touch screens).
- Printers.
- Audio/video on PC or giant screens.
- Internet messages.
- Control of other devices, such as robots and “intelligent” appliances.
- Secondary storage devices – hard drives, DVD drives and USB flash drives.
- Smart-phone and game controller vibration.
- Virtual reality devices like Oculus Rift.

Memory unit

- Stores information that is immediately available for processing.
- Also stores processed information until it can be placed on output device.
- Volatile information – typically lost when the computer's power is turned off.
- Often called memory, primary memory or RAM (Random Access Memory).
- Most commonly 2 to 16 GB.
- GB stands for gigabytes – a gigabyte is approximately one billion bytes.
- A byte is eight bits.
- A bit is either a 0 or a 1.

Arithmetic and logic unit (ALU)

- Performs calculations, such as addition, subtraction, multiplication and division.
- Also makes decisions such as less than, greater than, etc.

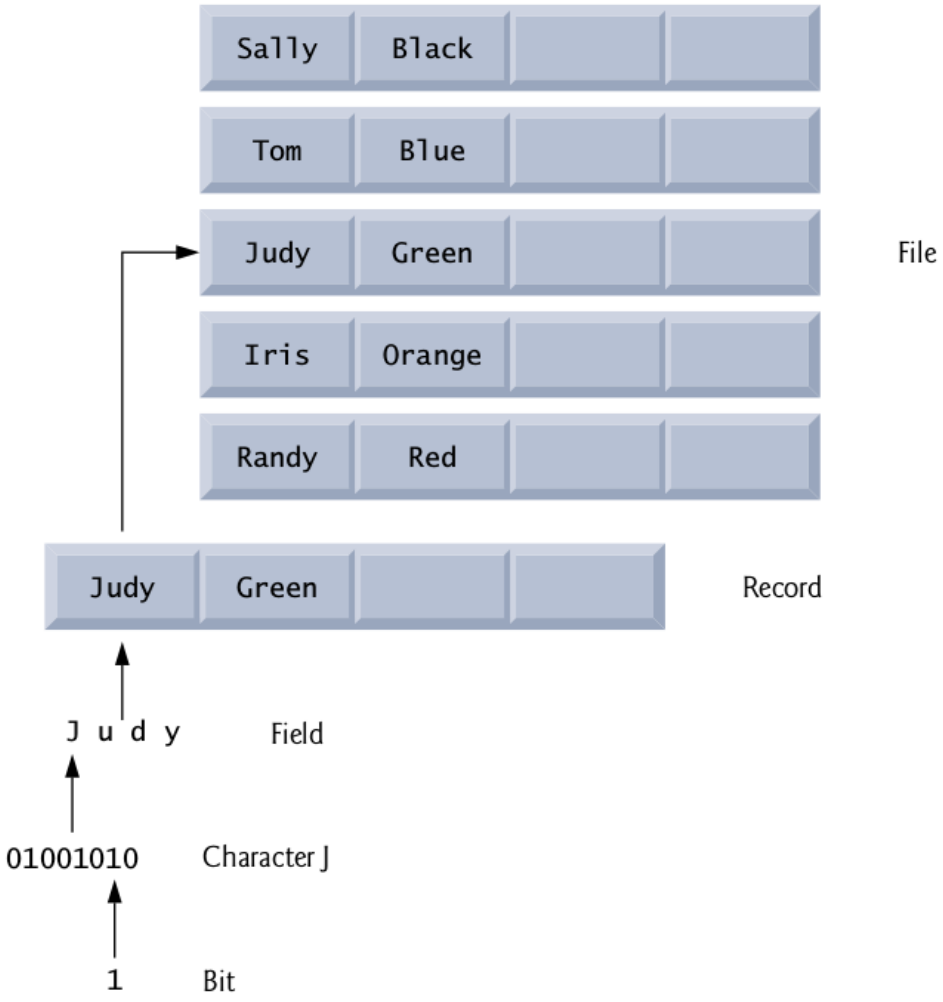
Central processing unit (CPU)

- Coordinates and supervises the operation of the other units.
- Tells the input unit when information should be read 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.
- Many computers have multiple CPUs and, hence, can perform many operations simultaneously.
- A multi-core processor implements multiple processors on a single integrated-circuit chip
  - a dual-core processor has two CPUs and a quad-core processor has four CPUs.
- Today's desktop computers have processors that can execute billions of instructions per second.

Secondary storage unit

- To store programs or data not actively being used by other units until they're again needed, possibly hours, days, months or even years later.
- Persistent information – preserved even when the computer's power is turned off.
- Takes much longer to access than information in primary memory, but its cost per unit is much less.
- Hard drives, DVD drives and USB flash drives, some of which can hold over 2 TB.
- TB stands for terabytes – a terabyte is approximately one trillion bytes.
- Typical hard drives on desktop and notebook computers hold up to 2 TB.

Data Hierarchy



Bits

- The smallest data item in a computer can assume the value 0 or the value 1.
- Called a bit (short for "binary digit" – a digit that can assume one of two values).
- Impressive functions performed by computers involve only the simplest manipulations of 0s and 1s.
  - examining a bit's value,
  - setting a bit's value, and
  - reversing a bit's value (from 1 to 0 or from 0 to 1).

Characters

- Tedious for people to work with data in the low-level form of bits like 1001100101110.
- Easier to work with
  - decimal digits (0–9),
  - letters (A–Z and a–z), and
  - special symbols (e.g., $, @, %, &, *,(,), –, +, ", :, ? and /).
- Digits, letters and special symbols are known as *characters*.
- A computer's *character set* is the set of all the characters used to write programs and represent data items.
- Computers process only 1s and 0s, so a computer's character set represents every character as a pattern of 1s and 0s.

Unicode and ASCII Character Sets

- *Unicode* character set is composed of characters represented using one, two or four bytes (8, 16 or 32 bits).
- Unicode contains characters for many of the world's languages, like English, Urdu, Arabic, Spanish etc.
- *ASCII* (American Standard Code for Information Interchange) character set is a popular subset of Unicode that represents uppercase and lowercase letters, digits and some common special characters.
- `https://www.youtube.com/watch?v=ngr0SIrfz6M`

Fields

- Just as characters are composed of bits, fields are composed of characters or bytes.
- A field is a group of characters or bytes that conveys meaning. For example,
  - a field consisting of uppercase and lowercase letters can be used to represent a person's name, and
  - a field consisting of decimal digits could represent a person's age.

Records

- Several related fields can be used to compose a record.
- Record for an employee might consist of the following fields
  - Employee identification number (a whole number)
  - Name (a string of characters)
  - Address (a string of characters)
  - Hourly pay rate (a number with a decimal point)
  - Year-to-date earnings (a number with a decimal point)
  - Amount of taxes withheld (a number with a decimal point)
- Thus, a record is a group of related fields.

Files

- A file is a group of related records.
- More generally, a file contains arbitrary data in arbitrary formats.
- In some operating systems, a file is viewed simply as a sequence of bytes – any organization of the bytes in a file, such as organizing the data into records, is a *view* created by the application programmer.
- It's not unusual for an organization to have many files, some containing billions, or even trillions, of characters of information.

Database

- A database is a collection of data organized for easy access and manipulation.
- Most popular is the *relational database*, in which data is stored in simple tables.
- A table includes records and fields.
- For example, a table of students might include
  - first name,
  - last name,
  - major,
  - year,
  - student ID number, and
  - grade point average fields.
- The data for each student is a *record*, and the individual pieces of information in each record are the *fields*.

Database

- You can search, sort and otherwise manipulate the data based on its relationship to multiple tables or databases.
- For example, a university might use data from the student database in combination with data from databases of courses, hostels, etc.

Machine, Assembly and High-Level Languages

- Programmers write instructions in various programming languages.
- Some are directly understandable by computers and others require intermediate translation steps.
- Hundreds of such languages are in use today. These may be divided into three general types:
  1. Machine languages
  2. Assembly languages
  3. High-level languages

Machine Languages

- Any computer can directly understand only its own machine language, defined by its hardware design.
- Machine languages generally consist of strings of numbers (ultimately reduced to 1s and 0s) that instruct computers to perform their most elementary operations one at a time.
- Machine languages are machine dependent (a particular machine language can be used on only one type of computer).
- Such languages are hard to understand for humans.
- For example, an early machine-language payroll program that adds overtime pay to base pay and stores the result in total pay:

```
+1300042774
+1400593419
+1200274027
```

Assembly Languages and Assemblers

- Programming in machine language is too slow and tedious.
- Instead of using strings of numbers that computers could directly understand, programmers began using English-like abbreviations to represent elementary operations.
  - `load`, `store`, `add`, `multiply`, etc.
- These abbreviations formed the basis of assembly languages.
- Translator programs called *assemblers* convert assembly-language programs to machine language.

Assembly Languages and Assemblers

- Assembly-language program that adds overtime pay to base pay and stores the result in total pay:

```
load basepay
add overpay
store totalpay
```

- Although clearer to humans, it's incomprehensible to computers until translated to machine language.

High-Level Languages and Compilers

- With the advent of assembly languages, computer usage increased rapidly.
- But programmers still had to use numerous instructions to accomplish even the simplest tasks.
- To speed the programming process, *high-level languages* were developed in which single statements could be written to accomplish substantial tasks.
- Translator programs called *compilers* convert high-level language programs into machine language.

High-Level Languages and Compilers

- High-level languages allow you to write instructions that look almost like everyday English and contain commonly used mathematical notations.
- High-level language might contain a single statement such as

  ```
  totalPay = basePay + overTimePay
  ```

- From programmer's standpoint, high-level languages are preferable to machine and assembly languages.
- C is one of the most widely used high-level programming languages.

Interpreters

- Compiling a large high-level language program into machine language can take considerable computer time.
- Interpreter programs, developed to execute high-level language programs directly, avoid the delay of compilation.
- However, interpreted programs run slower than compiled programs.

Interpreted languages like Python, JavaScript, Ruby, and PHP run by converting your source code on the fly into machine code as it is running. Because this conversion process happens while the code is running and adds overhead, interpreted languages are slower than compiled languages.