Python- and R-programming

Lecture 1 - Introduction to programming

Introduction

Computers can be programmed

- Designed to do any job that a program tells them to

Program: set of instructions that a computer follows to perform a task

- Commonly referred to as Software

Programmer: person who can design, create, and test computer programs

- Also known as software developer

Hardware (HW) and Software (SW)

HW - physical devices/parts that make up the computer

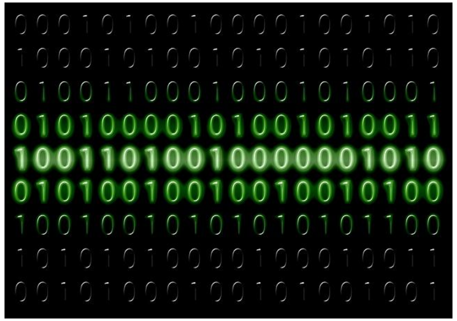
- Several components makes up the computer and work together
- ex. CPU, main memory, secondary storage devices, input and output devices

SW - all tasks a computer does is controlled by software

ex. application software, system software

Application software : programs that make computer useful

- ex. word processing, email, web browsers etc.



Software cont’d

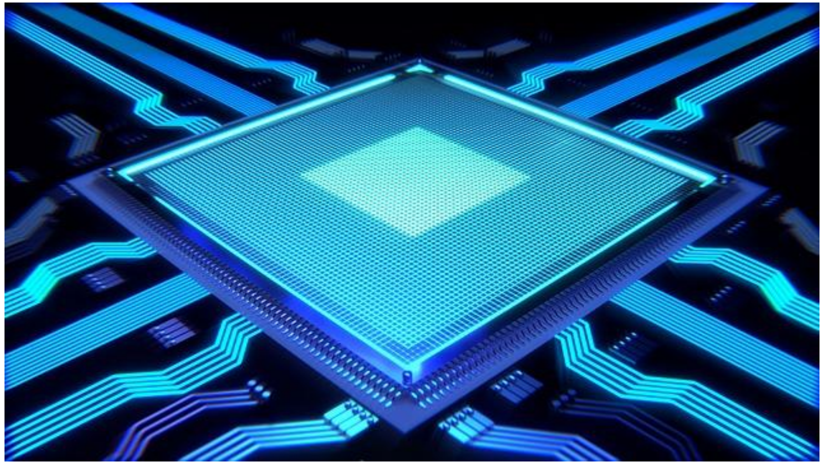
System software: programs that control and manage basic operations of a computer

- Operating system: controls operations of hardware components
- Utility Program: performs specific task to enhance computer operation or safeguard data
- Software development tools: used to create, modify, and test software programs

The CPU - Central Processing Unit

The part that actually runs programs

- Most important component
- If we don’t have this - no software can run



Main memory

This is where the computer stores programs while they are running as well as data that a program needs

Known as **RAM** - Random Access memory

- quick access by the CPU
- volatile memory, used for temporary storage while program(s) is(are) running
- erased when computer is turned off

Secondary storage devices

Can hold data for long periods of time

- Programs/data normally stored here and loaded to main memory when needed
- Types of secondary memory
    - Disk drives: magnetically encodes data onto a spinning circular disk
    - Solid state drive: faster than disk drive, no moving parts, stores data in solid state memory
        - SSD’s
    - Flash memory: portable, no physical disk
        - can be found in car radios, cell phones, digital cameras, solid-state drives, and printers
    - Optical devices: data encoded optically
        - allows users to use DVDs, CDs and Blu-ray optical drives, DVD’s

Input Devices

- Input: data the computer collects from people and other devices
- Input device: component that collects the data

Examples: keyboard, mouse, touchscreen, scanner, camera

Disk drives can be considered input devices because they load programs into the main memory

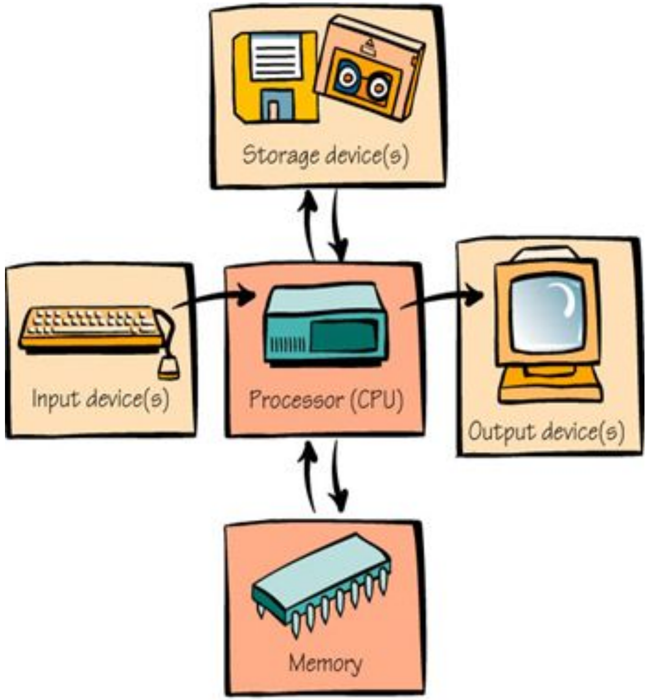
Output Devices

Output: data produced by the computer for other people or devices

- Can be text, image, audio, or bit stream

Output device: formats and presents output

- Examples: video display, printer
- Disk drives and USB drives can be considered output devices because data can be sent to them and saved



How computers store data

- All data in a computer is stored in sequences of 0s and 1s

- Byte: just enough memory to store letter or small number
  - Divided into eight bits
  - Bit: electrical component that can hold positive or negative charge, like on/off switch
  - The on/off pattern of bits in a byte represents data stored in the byte

Storing numbers

- Bit represents two values, 0 and 1
- Computers use binary numbering system
  - Position of digit j is assigned the value 2^{j-1}
  - To determine value of binary number sum position values of the 1s
- Byte size limits are 0 and 255
  - 0 = all bits off; 255 = all bits on
  - To store larger number, use several bytes

Storing characters

- Data stored in computer must be stored as binary number
- Characters are converted to numeric code, numeric code stored in memory
    - Most important coding scheme is ASCII
        - ASCII is limited: defines codes for only 128 characters
    - Unicode coding scheme becoming standard
        - Compatible with ASCII
        - Can represent characters for other languages

How does a program work

- The CPU is designed to perform simple operations on pieces of data
  - Ex: reading data, adding, subtracting, multiplying, and dividing numbers
  - Understands instructions written in machine language and included in its instruction set
  - Each brand of CPU has its own instruction set

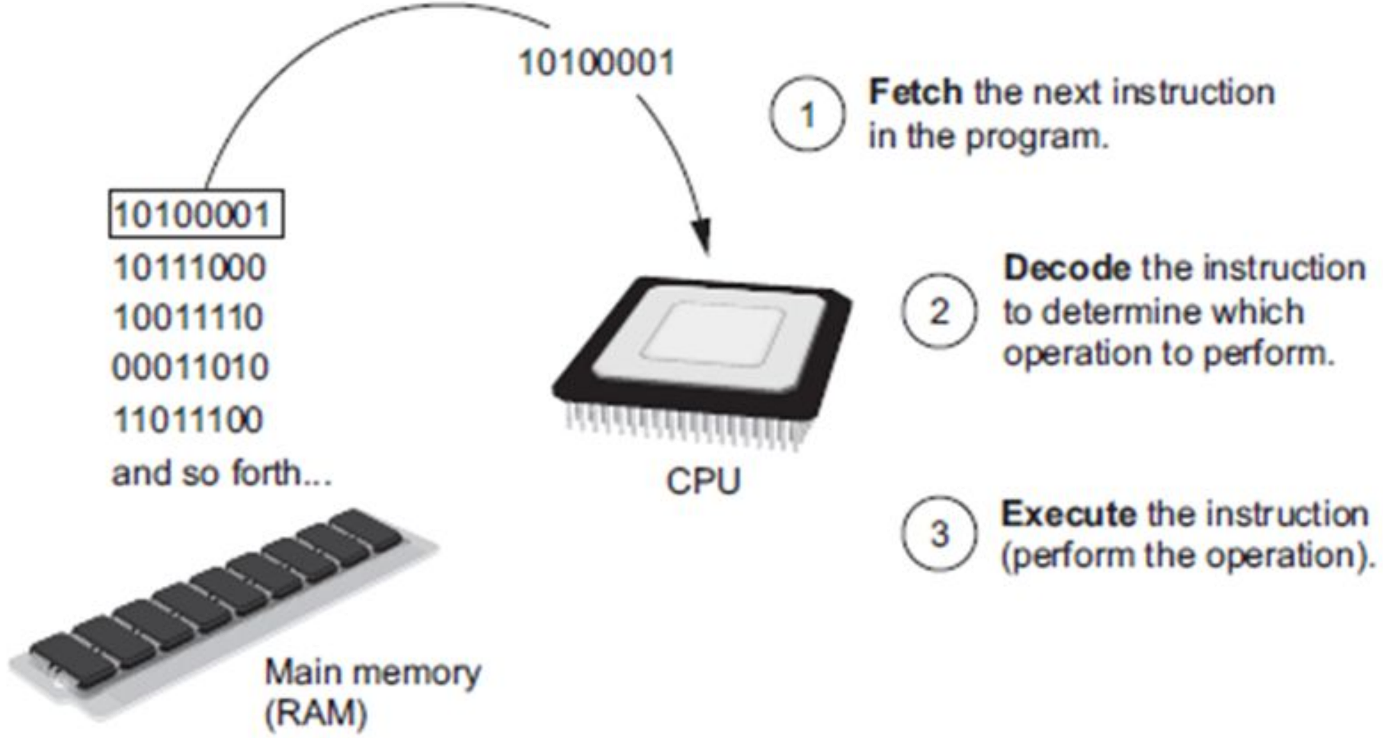
- To carry out meaningful calculation, CPU must perform many operations

How does a program work, cont’d

- Program must be copied from secondary memory to RAM each time CPU executes it

- The CPU executes programs in cycles:
  - Fetch: read the next instruction from memory into CPU
  - Decode: CPU decodes fetched instruction to determine which operation to perform
  - Execute: perform the operation

How does a program work, cont'd


10100001
10100001
10111000
10011110
00011010
11011100
and so forth...
Main memory
(RAM)
CPU
1 **Fetch** the next instruction in the program.
2 **Decode** the instruction to determine which operation to perform.
3 **Execute** the instruction (perform the operation).

From Machine Language to Assembly Language

- Impractical for people to write in machine language

- Assembly language: uses short words (mnemonics) for instructions instead of binary numbers
    - Easier for programmers to work with

- Assembler: translates assembly language to machine language for execution by CPU. So assembler is a program that converts assembly language into machine code. It takes the basic commands and operations from assembly code and converts them into binary code that can be recognized by a specific type of processor.

mnemonics

In computer assembler (or assembly) language, a mnemonic is an abbreviation for a certain operation. It's entered in the operation code field of each assembler program instruction.

Example, on an Intel microprocessor, **inc** ("increase by one") is a mnemonic.

High-Level languages

- Low-level language: close in nature to machine language
    - Example: assembly language

- High-Level language: allows simple creation of powerful and complex programs
    - No need to know how CPU works or write large number of instructions
    - More intuitive to understand

Key Words, Operators, and Syntax

- Key words: predefined words used to write program in high-level language
  - Each key word has specific meaning
- Operators: perform operations on data
  - Example: math operators to perform arithmetic
- Syntax: set of rules to be followed when writing program
- Statement: individual instruction used in high-level language

Interpreters and Compilers

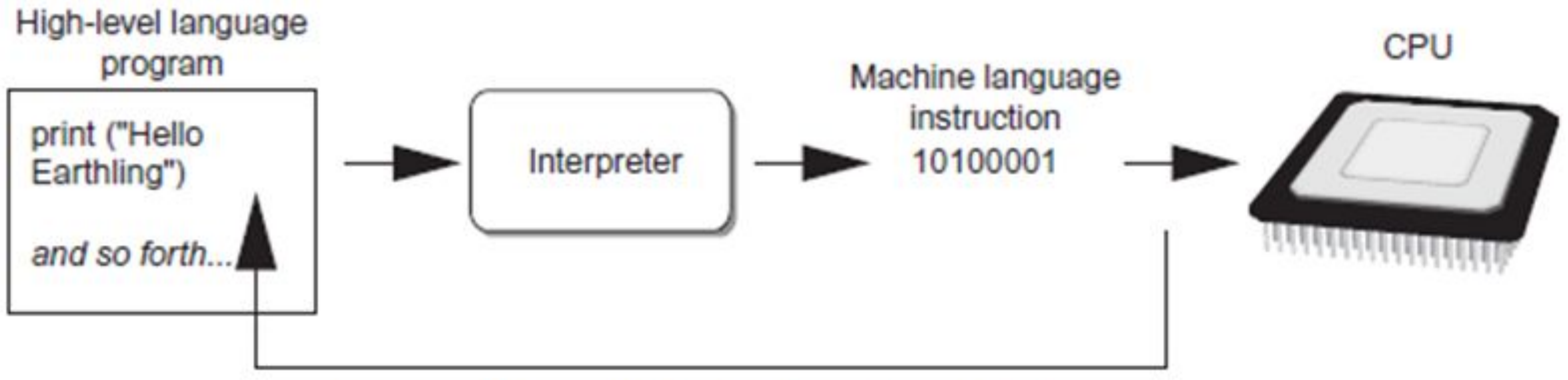
- Programs written in high-level languages must be translated into machine language to be executed

- Compiler: translates high-level language program into separate machine language program
    - Machine language program can be executed at any time

Compilers and Interpreters cont’d

- Interpreter: translates and executes instructions in high-level language program
    - Used by Python language
    - Interprets one instruction at a time
    - No separate machine language program
- Source code: statements written by programmer
    - Syntax error: prevents code from being translated

Interpreter translates high-level program instruction


High-level language program
print ("Hello Earthling")
and so forth...
Interpreter
Machine language instruction
10100001
CPU

The interpreter translates each high-level instruction to its equivalent machine language instructions and immediately executes them.

This process is repeated for each high-level instruction.

Using Python

- Python must be installed and configured prior to use
  - One of the items installed is the Python interpreter

- Python interpreter can be used in two modes:
  - Interactive mode: enter statements on keyboard
  - Script mode: save statements in Python script

Interactive Mode

- When you start Python in interactive mode, you will see a prompt
  - Indicates the interpreter is waiting for a Python statement to be typed
  - Prompt reappears after previous statement is executed
  - Error message displayed If you incorrectly type a statement

Good way to learn new parts of Python

Writing Python programs and run in Script Mode

- Statements entered in interactive mode are not saved as a program
- To have a program use script mode
  - Save a set of Python statements in a file
  - **The filename should have the .py extension**

To run the file, or script, type

```
python filename
```

at the operating system command line

The IDLE Programming Environment

- IDLE (Integrated Development Program): single program that provides tools to write, execute and test a program
  - Automatically installed when Python language is installed
  - Runs in interactive mode
  - Has built-in text editor with features designed to help write Python programs