CSE M226: Programming Languages

Lecture 0: Introduction and Course Outline

Rokan Uddin Faruqui

Dept. of Computer Science and Engineering
University of Chittagong, Bangladesh
Email: rufaruqui@cu.ac.bd

What is programming?

- Computers are deterministic machines, controlled by low-level (usually binary) machine code instructions.
- A computer can **[only] do whatever we know how to order it to perform** (Ada Lovelace, 1842)
- Programming is communication:
  - between a person and a machine, to tell the machine what to do
  - between people, to communicate ideas about algorithms and computation

From machine code to programming languages

- The first programmers wrote all of their code directly in machine instructions
  - ultimately, these are just raw sequences of bits.
- Such programs are extremely difficult to write, debug or understand.
- Simple "assembly languages" were introduced very early (1950's) as a human-readable notation for machine code
- FORTRAN (1957) - one of the first high-level - languages (procedures, loops, etc.)

What is a programming language?

- For the purpose of this course, a programming language is a **formal**, **executable** language for **computations**
- Non-examples:
  - English (not formal)
  - First-order Logic (formal, but not executable in general)
  - HTML4 (formal, executable but not computational)
- (HTML is in a gray area ? with JavaScript or HTML5 extensions it is a lot more ?computational?)

Why are there so many?

- Assembly languages were invented to allow operations to be expressed with **mnemonic** abbreviations
- Assemblers were eventually augmented with elaborate "**macro expansion**" facilities
  - to permit programmers to define parameterized abbreviations for common sequences of instructions
- **Problem**: each different kind of computer had to be programmed in its own assembly language
- People began to wish for a machine-independent languages
- These wishes led in the mid-1950s to the development of standard higher-level languages compiled for different architectures by

Why are there so many?

- Evolution

Why are there so many?

- Evolution
  - Revolution in structured programming. **Go to is Harmful**
  - Object-Oriented Programming
  - Compilation vs. Interpretation
- Special Purpose

Why are there so many?

- Evolution
  - Revolution in structured programming. **Go to is Harmful**
  - Object-Oriented Programming
  - Compilation vs. Interpretation
- Special Purpose
  - List processing
  - Character manipulation
  - System Programming
- Personal Preference

Why are there so many?

- Evolution
  - Revolution in structured programming. **Go to is Harmful**
  - Object-Oriented Programming
  - Compilation vs. Interpretation
- Special Purpose
  - List processing
  - Character manipulation
  - System Programming
- Personal Preference

What makes a language successful?

- Expressive Power

What makes a language successful?

- Expressive Power
  - **Turing Complete:** A computer or a programming language is said to be Turing complete if it can implement a Turing machine.

What makes a language successful?

- Expressive Power
  - **Turing Complete:** A computer or a programming language is said to be Turing complete if it can implement a Turing machine.
  - A Turing machine is a mathematical model of computation that can, in principle, perform any calculation that any other programmable computer can.

What makes a language successful?

- Expressive Power
  - **Turing Complete:** A computer or a programming language is said to be Turing complete if it can implement a Turing machine.
  - A Turing machine is a mathematical model of computation that can, in principle, perform any calculation that any other programmable computer can.
  - In a formal mathematical sense, all programming languages are **Turing Complete**
  - Language features to write clear, concise, and maintainable code are the most dominating factores.
- Ease of Use for the Novice

What makes a language successful?

- Expressive Power
  - **Turing Complete:** A computer or a programming language is said to be Turing complete if it can implement a Turing machine.
  - A Turing machine is a mathematical model of computation that can, in principle, perform any calculation that any other programmable computer can.
  - In a formal mathematical sense, all programming languages are **Turing Complete**
  - Language features to write clear, concise, and maintainable code are the most dominating factores.
- Ease of Use for the Novice
  - Learning curve. **Pascal** to **Java**

What makes a language successful?

- Expressive Power
  - **Turing Complete:** A computer or a programming language is said to be Turing complete if it can implement a Turing machine.
  - A Turing machine is a mathematical model of computation that can, in principle, perform any calculation that any other programmable computer can.
  - In a formal mathematical sense, all programming languages are **Turing Complete**
  - Language features to write clear, concise, and maintainable code are the most dominating factores.
- Ease of Use for the Novice
  - Learning curve. **Pascal** to **Java**

What makes a language successful?

- Ease of Implementation

What makes a language successful?

- Ease of Implementation
- Standardization

What makes a language successful?

- Ease of Implementation
- Standardization
- Open Source

What makes a language successful?

- Ease of Implementation
- Standardization
- Open Source
- Excellent Compilers

What makes a language successful?

- Ease of Implementation
- Standardization
- Open Source
- Excellent Compilers
- Economics, Patronage, and Inertia

What makes a language successful?

- Ease of Implementation
- Standardization
- Open Source
- Excellent Compilers
- Economics, Patronage, and Inertia

The Programming Language Spectrum

- **Declarative** languages
  - the focus is on what the computer is to do

The Programming Language Spectrum

- **Declarative** languages
  - the focus is on what the computer is to do
- **Imperative** languages
  - the focus is on how the computer should do it

The Programming Language Spectrum

- **Declarative** languages
  - the focus is on what the computer is to do
- **Imperative** languages
  - the focus is on how the computer should do it
- **Functional** languages
  - a computational model based on the recursive definition of functions.
  - inspiration from the lambda calculus, a formal computational model developed by Alonzo Church in the 1930s.
  - a program is considered a function from inputs to outputs, defined in terms of simpler functions through a process of refinement.
  - Lisp, ML, and Haskell.

The Programming Language Spectrum

- **Declarative** languages
  - the focus is on what the computer is to do
- **Imperative** languages
  - the focus is on how the computer should do it
- **Functional** languages
  - a computational model based on the recursive definition of functions.
  - inspiration from the lambda calculus, a formal computational model developed by Alonzo Church in the 1930s.
  - a program is considered a function from inputs to outputs, defined in terms of simpler functions through a process of refinement.
  - Lisp, ML, and Haskell.

The Programming Language Spectrum

- **Logic or constraint-based** languages
  - inspiration from predicate logic
  - model computation as an attempt to find values that satisfy certain specified relationships
  - goal-directed search through a list of logical rules
  - Prolog

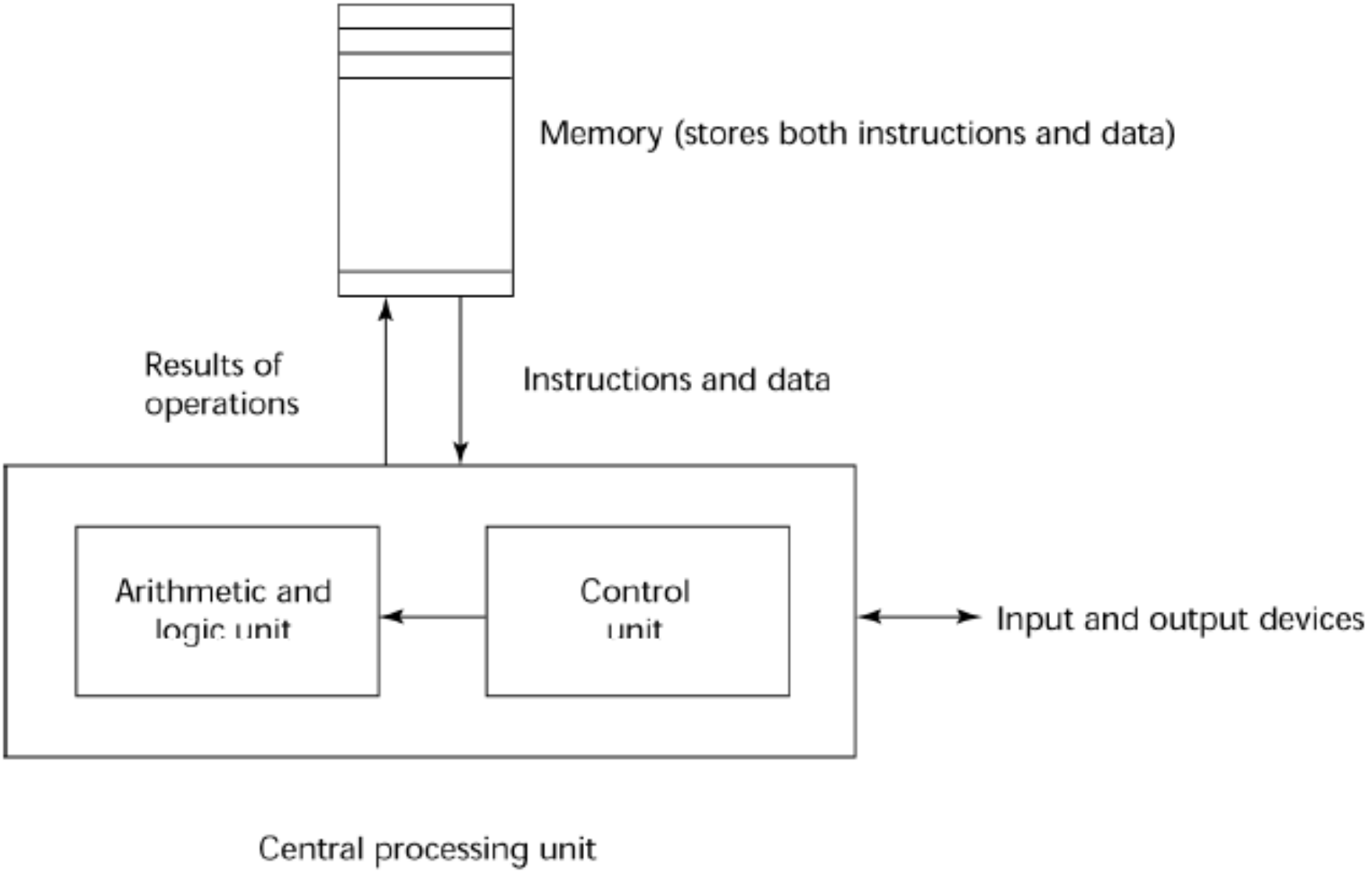
The Programming Language Spectrum

- **Logic or constraint-based** languages
  - inspiration from predicate logic
  - model computation as an attempt to find values that satisfy certain specified relationships
  - goal-directed search through a list of logical rules
  - Prolog
- **von Neumann** languages
  - the most familiar and widely used
  - Fortran, Ada, C
  - computation is the modification of variables
  - functional languages are based on expressions that have values, **von Neumann** languages are based on statements

The Programming Language Spectrum

- **Logic or constraint-based** languages
  - inspiration from predicate logic
  - model computation as an attempt to find values that satisfy certain specified relationships
  - goal-directed search through a list of logical rules
  - Prolog
- **von Neumann** languages
  - the most familiar and widely used
  - Fortran, Ada, C
  - computation is the modification of variables
  - functional languages are based on expressions that have values, **von Neumann** languages are based on statements

von Neumann Architecture



The Programming Language Spectrum

- **Object-oriented** languages
  - trace their roots to **Simula 67**

The Programming Language Spectrum

- **Object-oriented** languages
  - trace their roots to **Simula 67**
- **Scripting** languages
  - **csh** and **bash** are the input languages of job control (shell) programs
  - **PHP** and JavaScript are primarily intended for the generation of dynamic web content;
  - **Lua** is widely used to control computer games. Other languages, including
  - **Perl**, **Python**, and **Ruby**, are more deliberately general purpose.

The Programming Language Spectrum

- **Object-oriented** languages
  - trace their roots to **Simula 67**
- **Scripting** languages
  - **csh** and **bash** are the input languages of job control (shell) programs
  - **PHP** and JavaScript are primarily intended for the generation of dynamic web content;
  - **Lua** is widely used to control computer games. Other languages, including
  - **Perl**, **Python**, and **Ruby**, are more deliberately general purpose.

Course Format

Class Test: 2, Presentation: 1, Homework:1

- Class Test:
  - based on class lectures.
  - Do not depend exclusively on slides. Read text books, Please :)
- Presentation:
  - Group: maximum 2 person
  - Will provide a list of items. You can also suggest. It must be a programming language.
- Homework:
  - A small programming project implemented by the language chosen for the class presentation.

Text Book

- Scott, Michael L., **Programming Language Pragmatics (PLP)**, (4^{th} ed.) Morgan Kaufmann, 2015
- Robert Harper, **Practical Foundations for Programming Languages (PFPL)** (2^{nd} Edition). Cambridge University Press, 2016.
- John Mitchell , **Concepts in Programming Languages (CPL)** (11^{th} Edition), Cambridge University Press, 2015.