Concepts of Programming Languages

(R.W.Sebesta, Tenth Ed.)

Spring, 2017

유재우 교수

CHAPTER 1
PRELIMINARIES

Chapter 1. Preliminaries

- Reasons for studying concepts of PLs
- Programming domains
- Language evaluation criteria
- Influences on language design
- Language categories
- Language design trade-offs
- Implementation methods
- Programming environments

Reasons for Studying PLs

- Increased capacity to express ideas
- Improved background for choosing appropriate languages
- Increased ability to learn and design new languages
- Better understanding of the significance of implementation
- Better use of languages that are already known
- Overall advancement of computing

Programming Domains

- Scientific applications: FORTRAN,ALGOL60
  - floating point arithmetic computation
- Business application: COBOL
  - I/O, decimal data
- Artificial intelligence: LISP, PROLOG
  - symbolic manipulation
- Systems programming: PL/S, Bliss, C
  - high level languages writing OS
- Web software
  - HTML, XHTML, JavaScript

Programming Domains (cont.)

- Very high level language: SHELL, 4GL
  - scripting languages
- Special purpose languages: GPSS, OpenGL
  - simulation, graphics
- Objected oriented: C++, Smalltalk, Ada
  - Abstraction, Encapsulation
- Java, XML

Programming Domains (cont.)

- What next?
  - network language
  - multimedia language
  - agent programming language
  - networked virtual computing language
  - markup language
  - game language
  - parallel.....

Language Evaluation Criteria

- Readability
- Writability
- Reliability
- Cost

(See the table1.1)

Readability

- One of the most important criteria
  - maintenence
- Overall simplicity
  - Basic components:
    - a large number(PL/I) *vs* a small number(Pascal)
  - Too many features is bad
  - Multiplicity of features is bad
    - eg. In C, increment operator: ++, +=, +
  - Operator overloading

Readability(cont.)

- Orthogonality
  - A way to build the control and data structure of the language: small set
  - The more orthogonal, the fewer exception
    - eg, LISP *vs* C
  - Too much orthogonality can cause problems
    - ALGOL60
  - Need good combination of simplicity and orthogonality

Readability (cont.)

- Control statements
  - *goto* statements
- Data types and syntactic structures
  - integer, boolean, record, abstraction …
  - built-in data types
  - syntax considerations
    - identifier, special words, form and meaning

Writability

- *How easily a language can be used to create programs for a chosen problem domain?*
- Simplicity and orthogonality
  - A small number of primitive constructs and consistent set of rules for combining them
- Support for abstraction
  - process abstraction, data abstraction
- Expressivity
  - increment operator (in C): ++, +=

Reliability

- Type checking
  - testing for type errors
  - compile time checking *vs.* run time checking
- Exception handling
  - ability of a program to intercept run-time errors
- Aliasing
- Readability and writability

Cost Categories

- programmer training
- writing programs
- compiling program
- executing program
- language implementation system
- poor reliability
- maintaining programs

Influences on Language Design

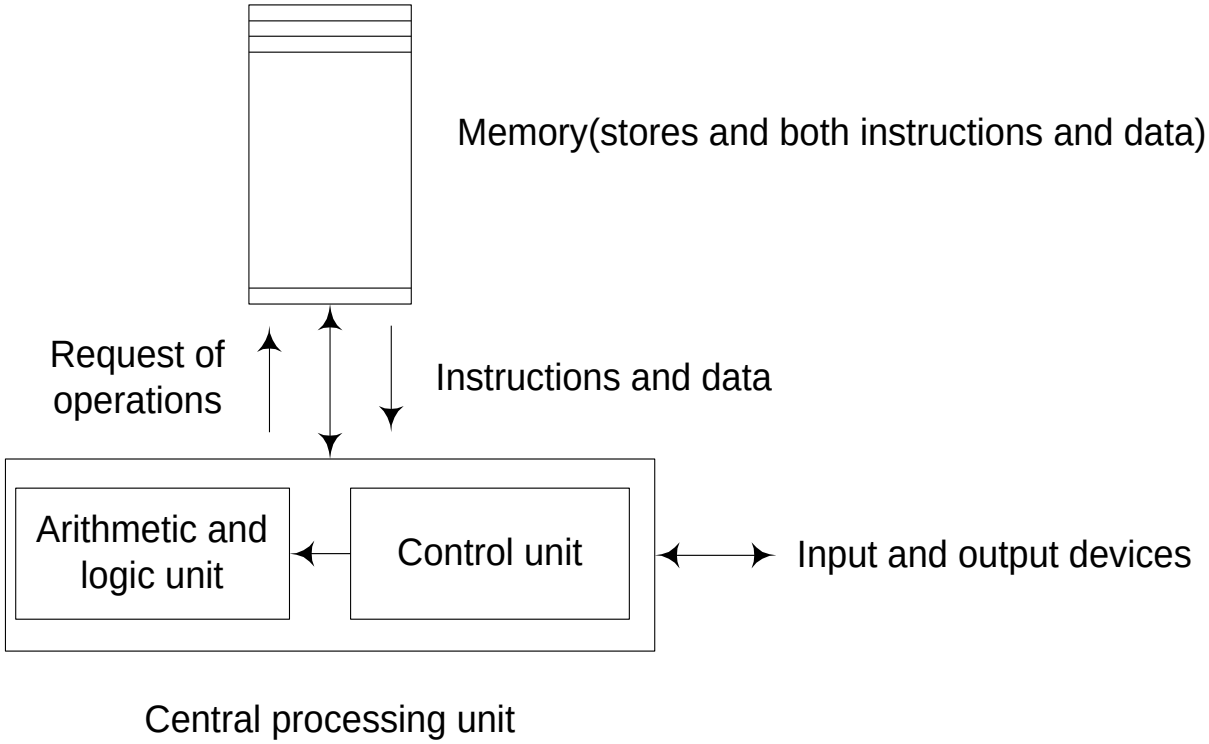
The most important factors

- computer architecture
- programming design methodologies

Computer architecture

- Von Neumann architecture
  - CPU and memory
  - Imperative language
    - variable, assignment statement, repetition
- Non von Neumann architecture
  - massively parallel machine
  - functional programming language

Von Neumann Computer



Program Design Methodologies

- Assembly programming
- High level language
  - Numeric data processing oriented: FORTRAN
  - Business data processing oriented: COBOL
- Structured programming
  - Structure FORTAN, ALGOL, PL/I…
- System programming
  - C for UNIX

Program Design Methodologies (cont.)

- Object oriented programming
  - abstraction: Smalltalk, C++, Ada
- Artificial intelligence programming
  - LISP, PROLOG
- Network transparent programming
  - JAVA
- What next?

Language Categories

- Imperative
- Functional
- Logic
- Object-oriented
  - closely related to imperative
- Markup

Language Design Trade-offs

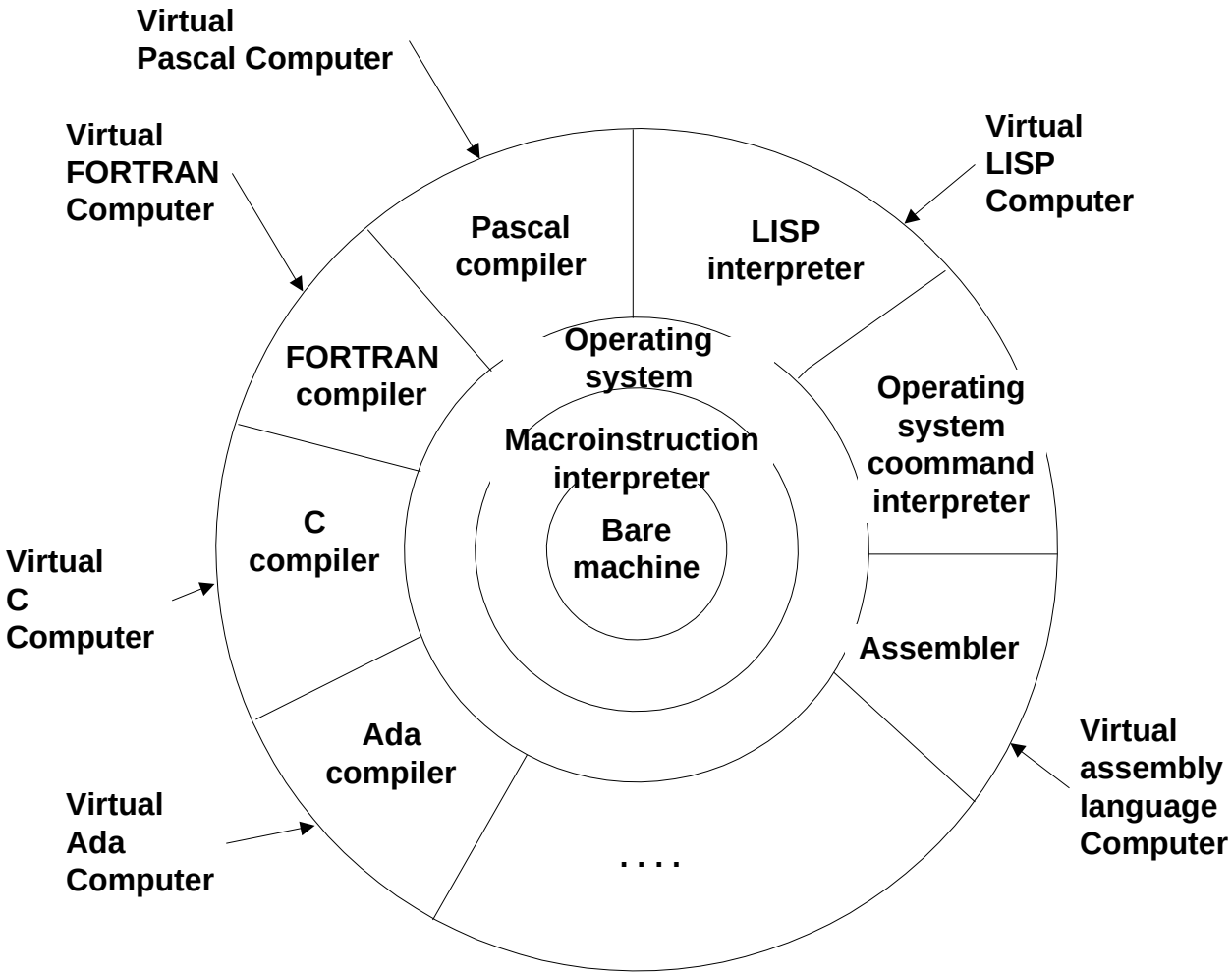
There are so many important but conflicting criteria, that their reconciliation and satisfaction is a major engineering task

- Reliability *vs.* cost of execution
- Readability *vs.* Writability
- Flexibility *vs.* Safety

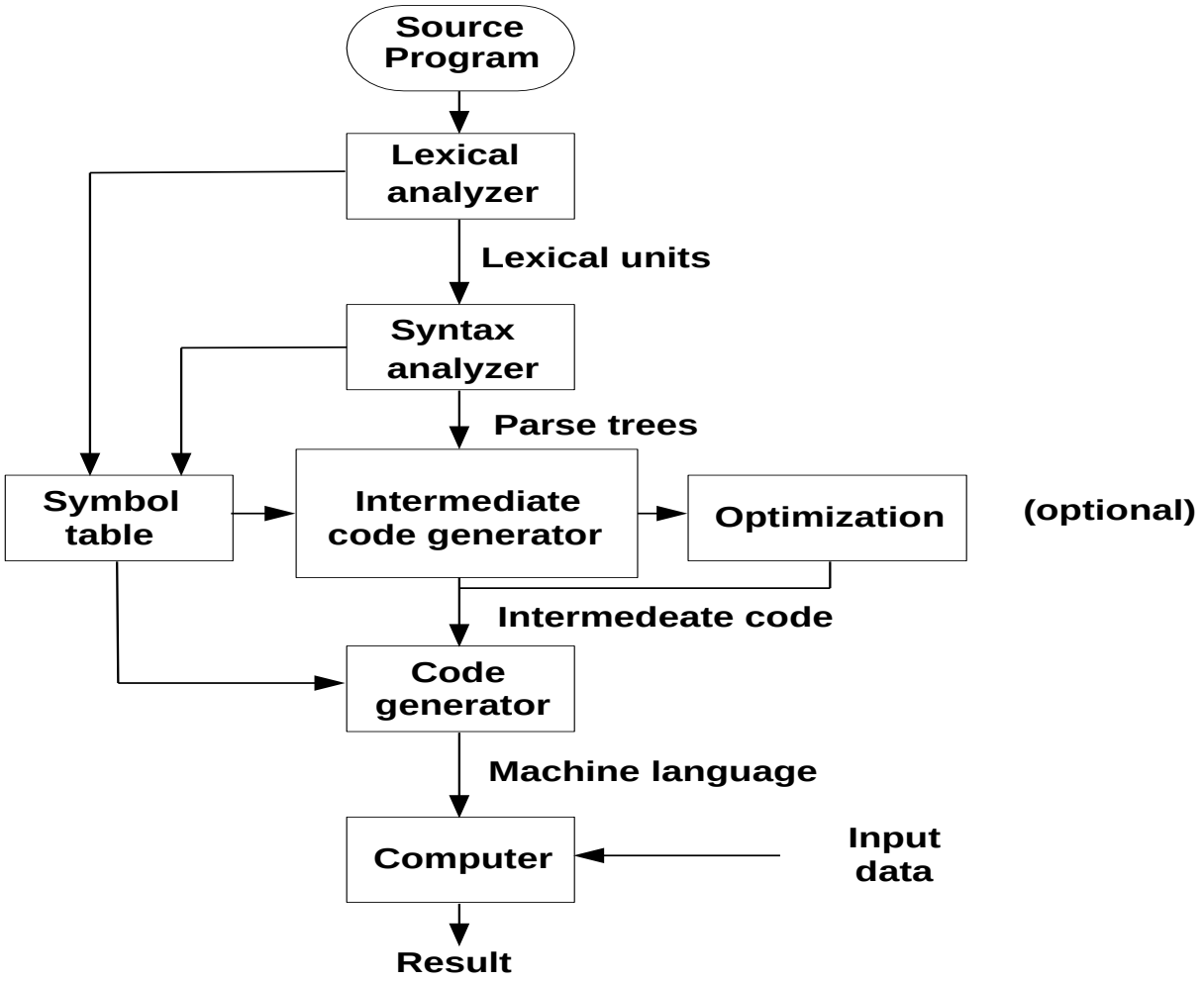
Implementation Methods

- Compilation and Execution
  - translate high-level program to machine code
  - slow translation, fast execution
- Interpretation
  - no translation
  - slow execution

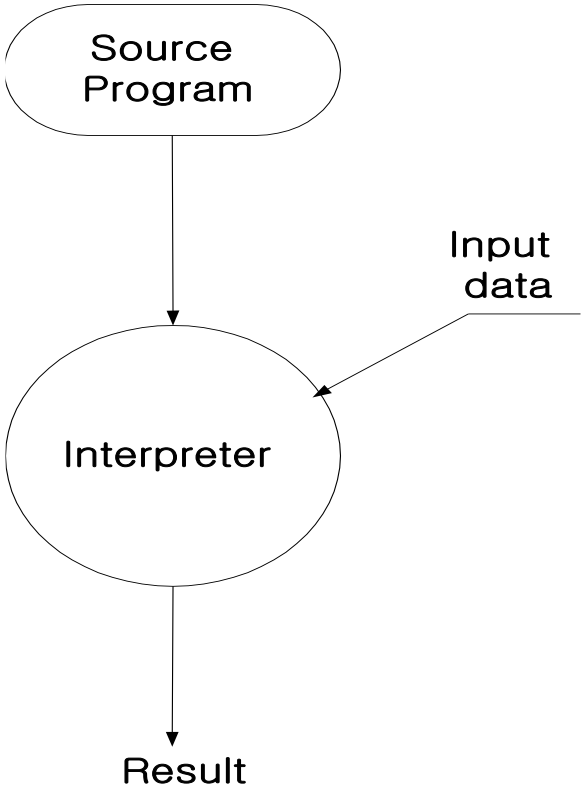
Compilation



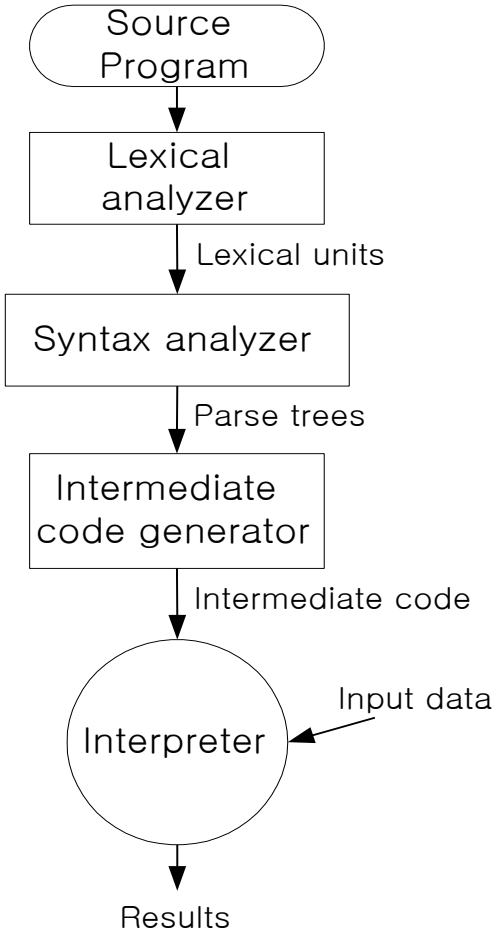
Compilation Process



Interpretation



Hybrid Implementation System



Preprocessors

- pcc
  - #include, #define, ..
- Ratfor
- Language extension

Programming Environments

Software development tools

- editor
- compiler or interpreter
- linker
- debugger
- software library