Programming Languages

Bibhas Adhikari

IIT Kharagpur

September 4, 2020

Programming Language

Programming language is a set of of notations for the description of algorithms and data structures. In particular, this should be utilized and implemented on a machine or device to perform certain desired tasks.

Programming language design methodology and its implementation have evolved continuously since the earliest high-level languages appeared in the 1950s. In the 1960s and 1970s, new languages were often developed as part of major software development projects.

- 1950s: FORTRAN and LISP
- 1970s: Ada, C, Pascal, Prolog, and Smalltalk
- 1980s: C++, ML, Perl, and Postscript
- 1990s: Java ...
- ..... almost 700 PLs are there today

Software architecture

- Hardware that supports a language has a great impact on language design
- The external environment supporting the execution of a program is termed as operating or target environment
- The host environment in which a program is designed, coded, tested, and debugged may be different from the operating environment in which the program ultimately is used

Computing industry in different eras

- Mainframe Era
- Personal Computer Era
- Networking Era

Languages used in different domains

Era	Application	Major Language	Other language
1960s	Business	COBOL	Assembler
	Scientific	FORTRAN	ALGOL, BASIC, APL
	System	Assembler	JOVIAL, Forth
	Artificial Intelligence	LISP	SNOBOL
Today	Business	COBOL, C++, Java, spreadsheet	C, PL/I, 4GLs
	Scientific	FORTRAN, C, C++, Java	BASIC
	System	C, C++, JAVA	Ada, BASIC, Modula
	Artificial Intelligence	LISP, Prolog	
	Publishing	TeX, Postscript, word processing	

A bit details about applications of PLs

In 1960s

- Business processing: Reading in large amounts of historical data on multiple tape drives, reading in a smaller set of recent transactions, and writing out a new set of historical data.
- Scientific calculations: Solution of various mathematical equations.
- Systems programming: For building operating systems and implementing compilers.
- Artificial-intelligence: Search through large data spaces.

A bit details about applications of PLs

21st century

- Business processing: E-commerce
- Scientific calculations: Numerical Mathematics, Data Analysis
- Systems programming: microprocessors running cars, microwave ovens, video games, and digital watches
- Artificial-intelligence
- Publishing LaTex

Influences on programming language development

Years	Influences and New Technology
1951-55	Hardware: Vacuum-tube computers; mercury delay line memories Methods: Assembly languages; foundation concepts: subprograms, data structures Languages: Experimental use of expression compilers
1956-60	Hardware: Magnetic tape storage; core memories; transistor circuits Methods: Early compiler technology; BNF grammars; code optimization; interpreters; dynamic storage methods and list processing Languages: FORTRAN, ALGOL 58, ALGOL 60, LISP

Influences on programming language development.

Years	Influences and New Technology
1961-65	Hardware: Families of compatible architectures; magnetic disk storage Methods: Multiprogramming operating systems; syntax-directed compilers Languages: COBOL, ALGOL 60 (revised), SNOBOL, JOVIAL
1966-70	Hardware: Increasing size and speed and decreasing cost; integrated circuits Methods: Time-sharing systems; optimizing compilers; translator writing systems Languages: APL, FORTRAN 66, COBOL 65, ALGOL 68, SNOBOL4, BASIC, PL/I, SIMULA 67, ALGOL-W

Influences on programming language development.

Years	Influences and New Technology
1971-75	Hardware: Small mass storage systems; semiconductor memories Methods: Program verification; structured programming; software engineering Languages: Pascal, COBOL 74, PL/I (standard), C, Scheme, Prolog
1976-80	Hardware: Mass storage systems; distributed computing Methods: Data abstraction; formal semantics; concurrent, embedded, and realtime programming techniques Languages: Smalltalk, Ada, FORTRAN 77, ML

Influences on programming language development.

Years	Influences and New Technology
1981-85	Hardware: Personal computers; workstations; video games; local-area networks; ARPANET Methods: Object-oriented programming; interactive environments syntax- directed editors Languages: Turbo Pascal, Smalltalk-80, use of Prolog, Ada 83, Postscript
1986-90	Hardware: Age of microcomputer; engineering workstation; RISC architectures; Internet Methods: Client/server computing Languages: FORTRAN 90, C++, SML (Standard ML)

Influences on programming language development.

Years	Influences and New Technology
1991-95	Hardware: Very fast inexpensive workstations and microcomput massively parallel architectures; voice, video, fax, multimedia Methods: Open systems; environment frameworks Languages: Ada 95, Process languages (TCL, PERL), HTML
1996-2000	Hardware: Computers as inexpensive appliances; Personal digital assistants; World wide web; Cable-based home networking; Gigabyte disk storage Methods: E-commerce Languages: Java, Javascript, XML

Good Language

- Clarity, simplicity, and unity
- Naturalness for the application
- Support for abstraction
- Ease of program verification
- Programming environment
- Cost of use
  - Cost of program execution
  - Cost of program translation
  - Cost of program creation, testing, and use
  - Cost of program maintenance

A Computational Machine and Compiler

1. A computation device or machine is an abstraction of the concept of a physical computer
2. Let $\mathcal{L}$ be a programming language which is used to write a program or a collection of commands or instructions (finite) that helps the device to understand the procedure to execute a job
3. A program in $\mathcal{L}$ is a a finite collection of instructions

Definition: Computational Machine

A computational machine for $\mathcal{L}$, denoted by $\mathcal{M}_{\mathcal{L}}$ (for now onwards just $\mathcal{M}$), is any set of data structures and algorithms which can perform the storage and execution of programs written in $\mathcal{L}$.

Computational machine

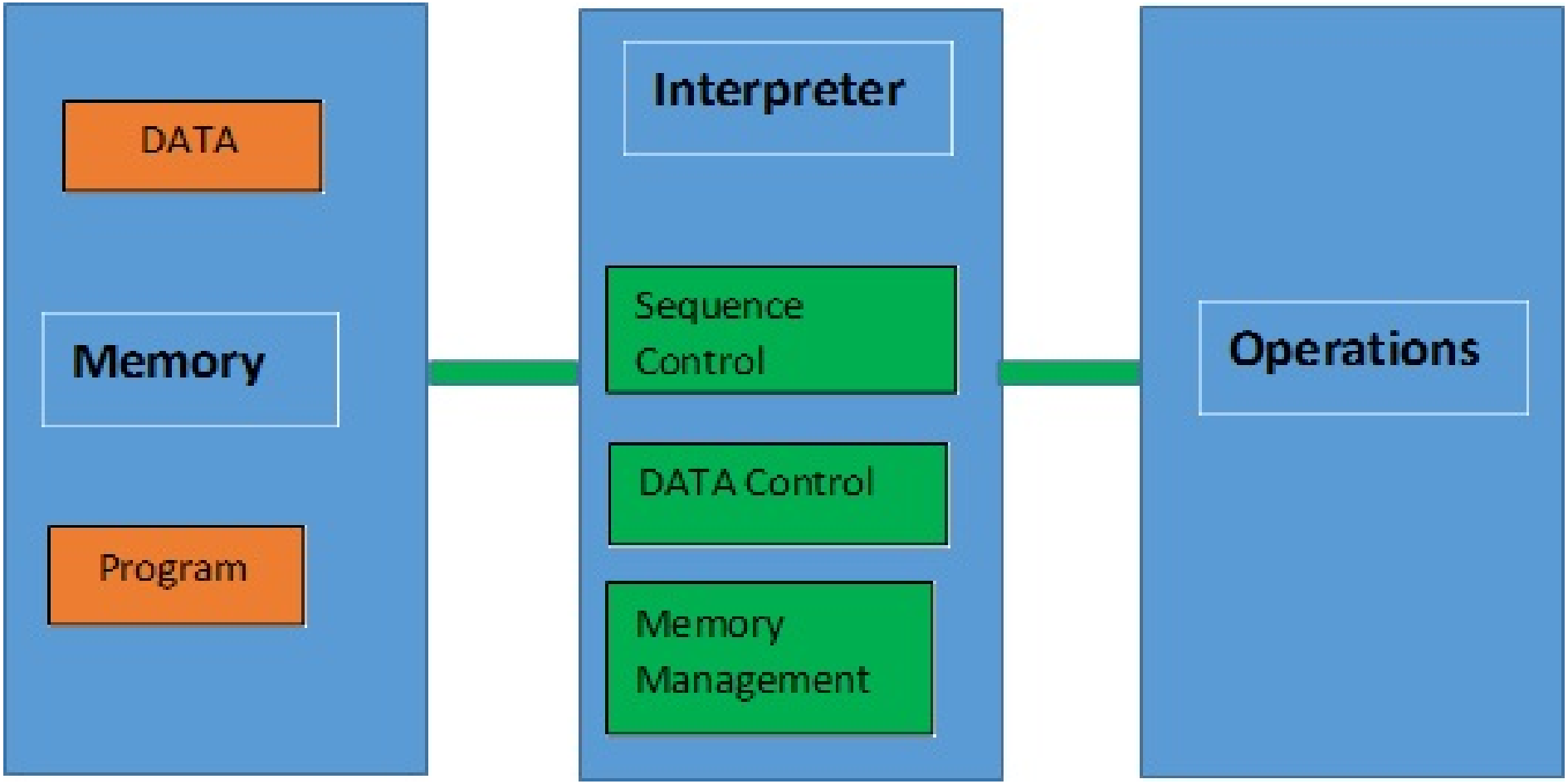

Figure: Structure of a computational machine

The memory serves to store data and programs while the interpreter is the component that executes the instructions contained in programs.

Interpreter

1. Interpreter performs the operations that are specific to the language $\mathcal{L}$ it is interpreting
2. The type of operations executed by the interpreter and associated data structures, can be characterized into the following categories:
   - Operations for processing primitive data : For example, numbers (integer or real) are almost always primitive data, arithmetic operations are primitive operations
   - Operations and data structures for controlling the sequence of execution of operations. For example, to hold the address of the next instruction to execute, to update the address of the next instruction to execute
   - Operations and data structures for controlling data transfers. For example, to control how operands and data is to be transferred from memory to the interpreter and vice versa
   - Operations and data structures for memory management. For example, to allocate data and programs in memory