Computer architecture and Programming language implementation with history

A Slecture by Abu Sadat Md. Sayem, Hasib Mahmud & Md. Oli Ullah

Course: CSE 425 Summer'19

What you will learn from this slecture?

- Programming language's history and implementation methods
- Von Neumann Computer Architecture

Programming Language:

A programming language is a set of instructions using which we can instruct computer easily instead of writing machine code by ourselves. The translation from a programming language to machine code is done usually by a compiler or interpreter.

Why do we use programming languages?

Since, computer only understands binary code and writing binary code is error prone and time consuming, we invented programming language which helps us to read, write and maintain program for computer easily without working directly with binary code.

Why are there so many programming languages?

We have created and are still creating new programming languages to meet up our needs.A few reasons that often thrives our quest of designing new languages are as follows:

- **Evolution:** Everyday we are learning better ways of doing things.
- **Pleasure:** To make the implementation of language pleasant to use.
- **Purpose:** To meet special purposes. Example of some of these special purposes are:
    - System programming
        - Languages: C, C++
    - Numerical computations
        - Languages: Fortran, A Programming Language (APL)
    - Data manipulation
        - Languages: Lisp, SQL
    - Network oriented programs
        - Languages: Java, C, Python
    - Embedded systems
        - Languages: Rust, C
- **Hardware:** Since our hardwares are improving to perform much faster day by day and to work with a specific hardware, new programming languages may be created.

Why should we study programming languages?

Studying programming languages helps us to make the right choice of programming language to work to our specific purpose and if we know what a programming language does behind the scene, we can write efficient code which will reduce implementation and maintenance costs.

Implementation Methods

Programming Language Implementation describes the method for how your code (such as Java) as an example is converted to a language that the machine (processor etc) understand. We refer to this as machine code.

There are three approaches for programming language implementation:

1. Compilation
2. Pure interpretation
3. Hybrid implementation

1. Compilation

Compiler is a software or a program that process all source code and translate into machine understandable code. This process is known as Compilation.

Advantage

1. Much faster than other implementation.
2. Better error detection mechanisms.
3. More secured

Disadvantage

1. Hardware and operating system (OS) dependent
2. Debugging is comparatively hard
3. Generates intermediate object code, hence requires more memory.

A compiler basically executes four major steps

1. Lexical analyzer
2. Syntax analyzer
3. Intermediate code generator and semantic analyzer
4. Code generator

Lexical analyzer: The compiler converts the sequence of characters that appear in the source code into a series of strings of characters (known as tokens), which are associated with a specific rule by a program called a lexical analyzer. A symbol table is used by the lexical analyzer to store the words in the source code that correspond to the token generated.

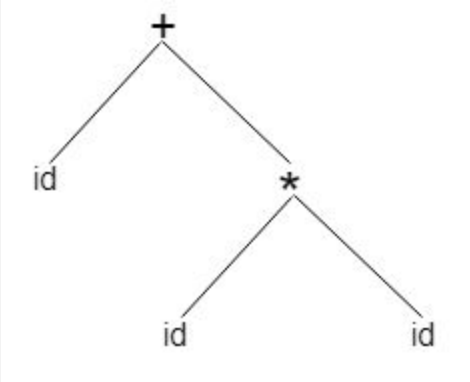
Output: Lexical unit.

```
int x = y + z * 2;
```

Token names	Sample of tokens
keyword	int
identifier	x, y, z
separator	;
operator	=, +
literal	2

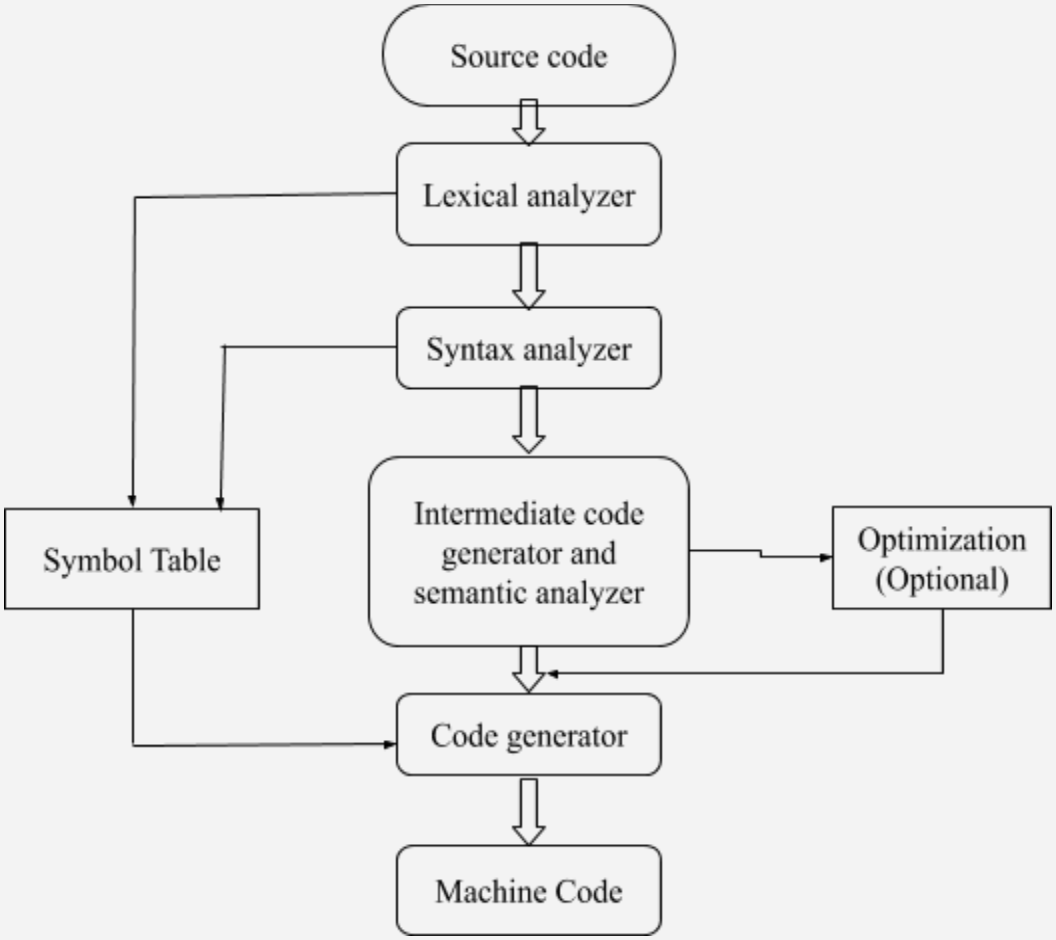
Table 1: Tokens

Syntax analyzer: This is the second phase of the compiler. In this it checks if the given input is in the correct syntax of the programming language. It is known as the **Parse Tree or Syntax Tree**. This phase use context free grammar (CFG).
Semantic analyzer: Semantics help interpret symbols, their types, and their relations with each other. Semantic analysis judges whether the syntax structure constructed in the source program derives any meaning or not.
A statement id + id * id would have the following syntax tree:



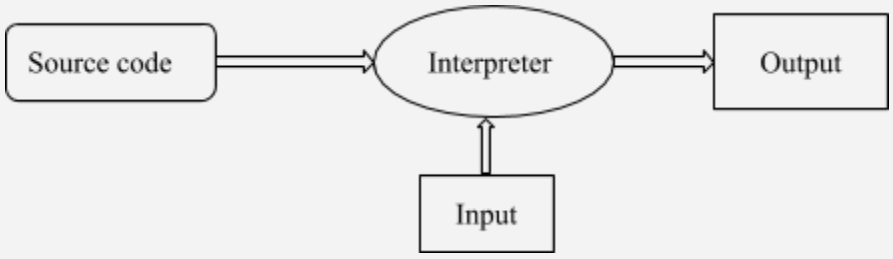
Code generation: In this phase compiler generate machine understandable code (Machine Code) that can be readily executed by a computer.

Diagram of compilation process:



2. Pure interpretation

Pure interpretation is another method of implementation. With this approach source codes or programs are interpreted by another program called an interpreter.



Advantage

1. Easy to debug.

2. No intermediate code generate hence memory efficient.
3. Support dynamic typing.
4. The interpreter reads a single statement and shows the error if any. You must correct the error to interpret next line.

Disadvantage

1. Comparatively slower than compiler.
2. Interpreters can be susceptible to code injection attacks (a code injection attack is that the attacker tricks the application into running some code statements that are not part of the intended functions of that application).

3. Hybrid implementation

In a hybrid implementation system, compiler and pure interpreter are used together where a high level programming language is translated to intermediate language (for instance: Java Bytecode) which is very close to machine code hence allows easy interpretation. This implementation is faster than pure interpretation implementation because the source language is compiled only once.

Examples: Python, Java, C++ 11, Swift, Perl.

Advantages of hybrid implementation:

- **Portability:** Portable to any machine that has a bytecode interpreter. Different interpreter for different hardwares. For instance, Java Virtual Machine (JVM).
- **Platform independent:** It can be run in different OS since the intermediate code is not platform specific.
- **Error detection:** It provides better error detection mechanism. Error is detected during compilation period. Hence, if compiled code is error free, interpreter can simply interpret the code. Thus, it is making the interpreter more simple.
- **Abstraction:** This provides an abstraction to hardware and programmers, which makes programmers worry less about the machine code.

Example of hybrid implementation using Java code taken from Wikipedia:

Consider the following Java Code:

```
outer:
for (int i = 2; i < 1000; i++) {
      for (int j = 2; j < i; j++) {
      if (i % j == 0)
```

```
            continue outer;
        }
        System.out.println (i);
}
```

A Java compiler might translate the above Java code into Bytecode as follows:

```
 0:  iconst_2
 1:  istore_1
 2:  iload_1
 3:  sipush  1000
 6:  if_icmpge   44
 9:  iconst_2
 10:  istore_2
 11:  iload_2
 12:  iload_1
 13:  if_icmpge   31
 16:  iload_1
 17:  iload_2
 18:  irem
 19:  ifne   25
 22:  goto   38
 25:  iinc   2, 1
 28:  goto   11
 31:  getstatic
 34:  iload_1
 35:  invokevirtual
 38:  iinc   1, 1
 41:  goto   2
 44:  return
```

As seen, the bytecode already looks very simple than the Java code. So, interpretation of bytecode will be much easier than the actual high level Java code.

Diagram of a Hybrid implementation system:

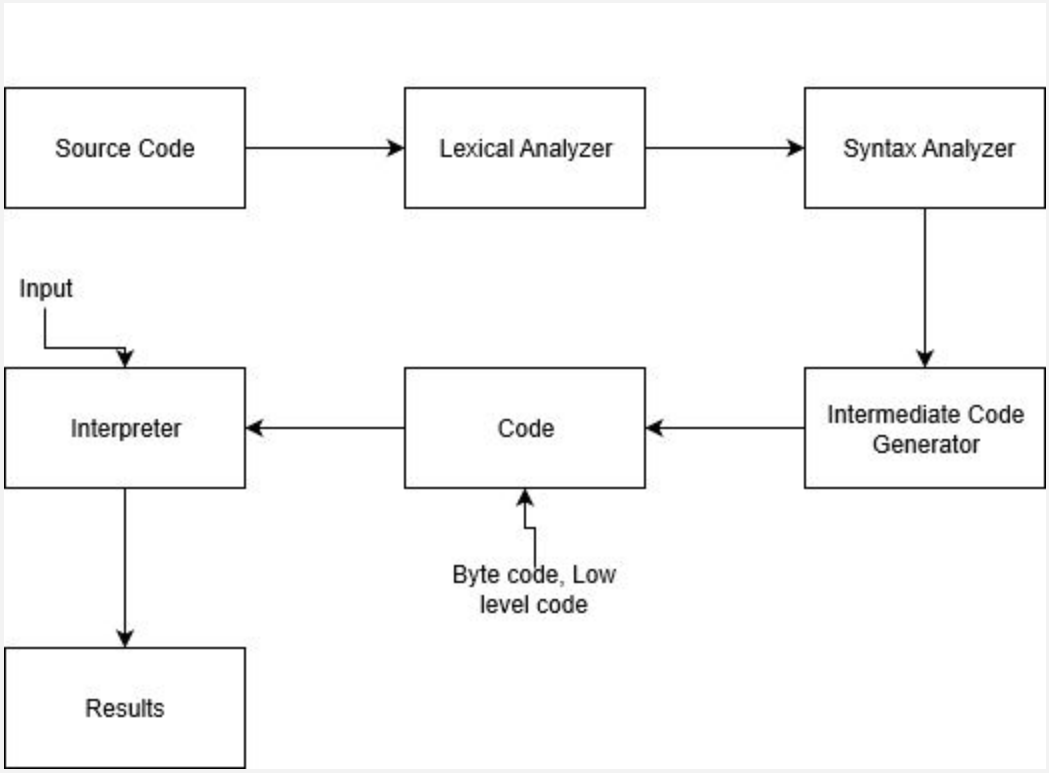


Table of Programming language

Programming language	Inventor	Year	Use	Specialty	Limitations
Assembly language	Kathleen Booth	1947	General	Understanding Hardware	Hardware dependent, No SDK
Fortran	John Backus	1957	Application, numerical computing	Faster for numerical calculations	Poor string handling, Data scoping is limited.
ALGOL	Backus, Bauer, Green, MORE...	1960	Application	Came up with first commercial application	Hard to learn
COBOL	Howard Bromberg, Howard Discount, MORE...	1969	Application, business	Designed for business use, English-like syntax,	Compilation time of a COBOL program might be greater than other machine-oriented programming language

Lisp(Scheme)	John McCarthy	1975	General	Unified, simple and elegant way of representing both code and data.	Poor readability, Unintuitive syntax
C	Dennis Ritchie	1972	Application, system, general purpose, low-level operations	C combines the features of both high level and low level languages.	Hard to debug, Doesn't perform runtime data type checking
C++	Bjarne Stroustrup	1985	Application, system	Portable language, Much faster and efficient, Support namespace	Lack of garbage collection
Python	Guido van Rossum	1990	Application, general, web, scripting, artificial intelligence, scientific computing	Portable, Extensive support libraries, Dynamically typed language	Slow, memory consumption is high,
Java	James Gosling	1995	Application, business, client-side, general, mobile development, server-side, web	Platform-Independent , Multithreaded, easy to learn	Slower performance, Memory consumption is high
PHP	Rasmus Lerdorf	1995	Server-side, web application	Work with databases more efficiently, Easy to learn	Poor error handling
Javascript	Brendan Eich	1995	Client-side, server-side, web	Faster, Easy to learn	Lack of client-side security

Table 2: Programming language

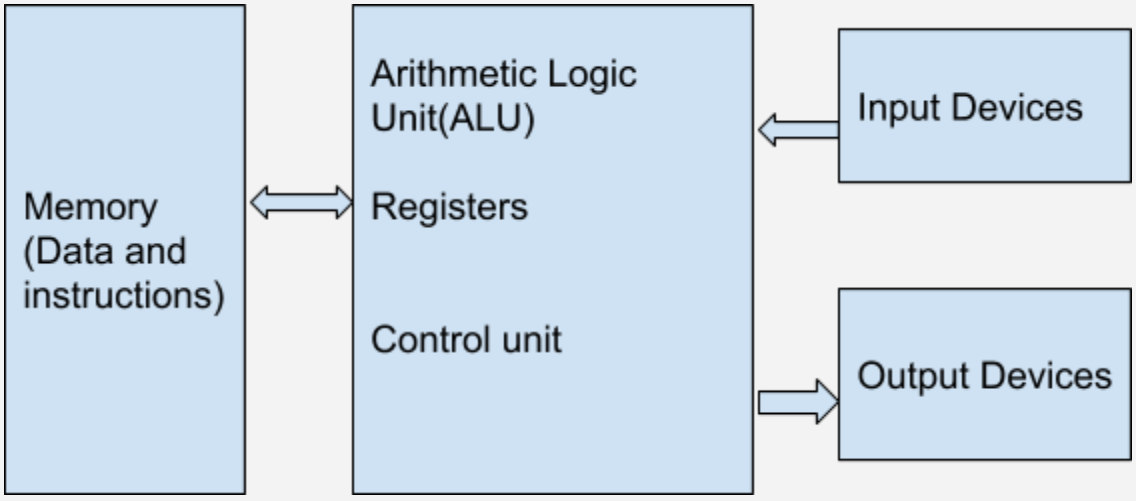
Computer Architecture: Historically there have been two types of computers:

Fixed program computer: There function is very specific and they couldn't be programmed.

Stored program Computer: These can be programmed to carry out different tasks,applications are stored on them hence the name

The modern computers are based on a stored-program concept introduced by John Von Neumann. In this stored program concept, programs and data are stored in a separate storage unit called memories and are treated the same. This novel idea meant that a computer build with this architecture would be much easier to reprogram. Von Neumann architecture was first published

by John Von Neumann in 1945. His computer architecture design consists of a Control Unit, Arithmetic and logic unit(ALU), Memory unit, Registers and Inputs/Outputs. This design is still used in most computers produced today.



Central Processing Unit:
The Central processing unit is the electronic circuit responsible for executing the instructions of a computer program. It is sometimes referred to as the microprocessor or processor. The Central Processing Unit (CPU) contains Arithmetic & logic unit (ALU), Control Unit (CU) and variety of registers.

Registers:
Registers are high speed storage areas in Central Processing Unit. All data must be stored in a register before it can be processed.

Control Unit:
The control unit controls the operation of the computer's Arithmetic Logic Unit, Memory and Input/Output devices, telling them how to respond to the program instructions it has just read and interpreted from the memory unit. The Control Unit also provides the timing and control signals required by other computer components.

Buses:
Buses are components by which data is transmitted from one part of a computer to another,connecting all major internal components to the CPU and memory. A standard CPU bus is comprised of a control bus, data bus and address bus.

Address Bus: Carries the addresses of data (but not the data) between the processor and memory.

Data Bus: Carries data between the processor ,the memory unit and the input/output devices.

Control Bus: Carries control signals/commands from the CPU (and status signals from other devices) in order to control and coordinate all the activities within the computer.

Memory Unit:
The memory unit consists of Random Access Memory (RAM), sometimes referred to as primary or main memory. Unlike a hard drive (Secondary memory) this memory is fast and also directly accessible by the CPU. RAM is split into partitions. Each partition consists of an address and its contents (both i binary form). The address will uniquely identify every location in the memory. Loading data from permanent memory (hard drive) into the faster and directly accessible temporary memory (RAM), allows the CPU to operate much quicker.

Advantages of Von Neumann Architecture

1. Less physical space is required than Harvard.
2. Handling just one memory block is simpler and easier to achieve.
3. Data can be retrieved in the same manner.

History of Programming Languages:

The first programming language was created in 1883, when a woman named Ada Lovelace worked with Charles Babbage on his very early mechanical computer, the Analytical Engine.

Assembly Language (1949): First widely used in the electronic delay storage automatic calculator, assembly language is a type of low-level computer programming language that simplifies the language of machine code, the specific instruction needed to tell the computer what to do.

Fortran (1957): A computer programming language created by John Backus for complicated scientific, mathematical, and statistical work ,Fortran stands for formula translation.it is one of the programming languages still used today.

Algol (1958): Created by a committee for scientific use, Algol stands for Algorithmic Language. Algol served as a starting point in the development of languages such as Pascal, C, C++, and Java**.**

LISP (1959): Created by John McCarthy of MIT, LISP is still in use. It stands for LISt Processing language. It was originally created for artificial intelligence research but today can be used in situations where Ruby or Python are used.

Pascal (1970): Developed by Niklaus wirth , Pascal was named in honor of the French mathematician, physicist, and Philosopher Blaise Pascal.It is easy to learn and was originally created as a teaching tool of computer programming.Pascal was the main language used for software development in Apple's early year.

C (1972): Developed by Dennis Ritchie at Bell Labs, C is considered by many as high-level language.A high-level computer programming language is closer to human language and more removed from the machine code.C was created so that an operating system called UNIX could be used on many different types of computers.It has influenced many other languages including Ruby,c#, Go, java , Java script, Perl, PHP, python.

Objective-C (1983): created by Brad cox and Tom Love, Objective C is the main programming language used when writing software for macOSand iOS Apple's operating system.

C++ (1983): C++ is an extension of the C language and was developed by Bjarne Stroustrup.It is one of the most widely used languages in the world.C++ used in game engines and high-performance software like Adobe Photoshop.Most packaged software is still written in C++.

Python (1991): Designed by Guido Van Rossum, Python is easier to read and requires fewer lines of code than many other computer programming languages. It was named after the British comedy group Monty Python. Popular sites like Instagram use frameworks that are written in Python.

Java (1995): Originally called Oak, Java was developed by Sun Microsystems. It was intended for cable boxes and hand-held devices but was later enhanced so it could be used to deliver information on the World Wide Web. Java is everywhere, from computers to smartphones to parking meters. Three billion devices run Java!

PHP (1995): Created by Rasmus Lerdorf, PHP is used mostly for Web development and is usually run on Web servers. It originally stood for Personal Home Page, as it was used by Lerdorf to manage his own online information. PHP is now widely used to build websites and blogs. WordPress, a popular website creation tool, is written using PHP.

Ruby (1995): Ruby was created by Yukihiro "Matz" Matsumoto, who combined parts of his favorite languages to form a new general-purpose computer programming language that can perform many programming tasks. It is popular in Web application development. Ruby code executes more slowly, but it allows for computer programmers to quickly put together and run a program.

C# (2000): Developed by Microsoft with the goal of combining the computing ability of C++ with the simplicity of Visual Basic, C# is based on C++ and is similar to Java in many aspects. It is used in almost all Microsoft products and is primarily used for developing desktop applications.

Swift (2014): Developed by Apple as a replacement for C, C++, and Objective-C, Swift is supposed to be easier to use and allows less room for mistakes. It is versatile and can be used for desktop and mobile apps and cloud services.

Most computer programming languages were inspired by or built upon the concepts from previous computer programming languages. Today, while older languages still serve as a strong foundation for new ones, newer computer programming languages make programmers' work simpler.

References:

1. https://en.wikipedia.org/wiki/Java_bytecode
2. https://getrevising.co.uk/grids/von-neumann-architecture
3. https://en.wikipedia.org/wiki/Comparison_of_programming_languages