North South University

Slecture 01

Group: Red Keep

Course: CSE 425 – Concepts of Programming Languages

Section: 3

Faculty Initial: MSK1

Faculty Name: Md. Shahriar Karim

Semester: Summer 2019

Date of Submission: 25th June 2019

Group Members:

1. **Arshi Siddiqui Promiti – ID: 1611236042**
2. **Shamima Haque Priya – ID: 1620069042**
3. **Bakhtiyar Habib – ID: 1531038642**

Review of Von Neumann Architecture

1. **Von Neumann Architecture:**

- PC design created in 1945 by the mathematician and physicist John Von Neumann.
- The architecture was created to store program instructions in the memory along with the data on which those instructions will be executed upon
- The architecture is composed of three distinct components (Sub-system). They are:
  - (a) CPU (Central Processing Unit)
  - (b) Memory
  - (c) I/O (input/output) interfaces

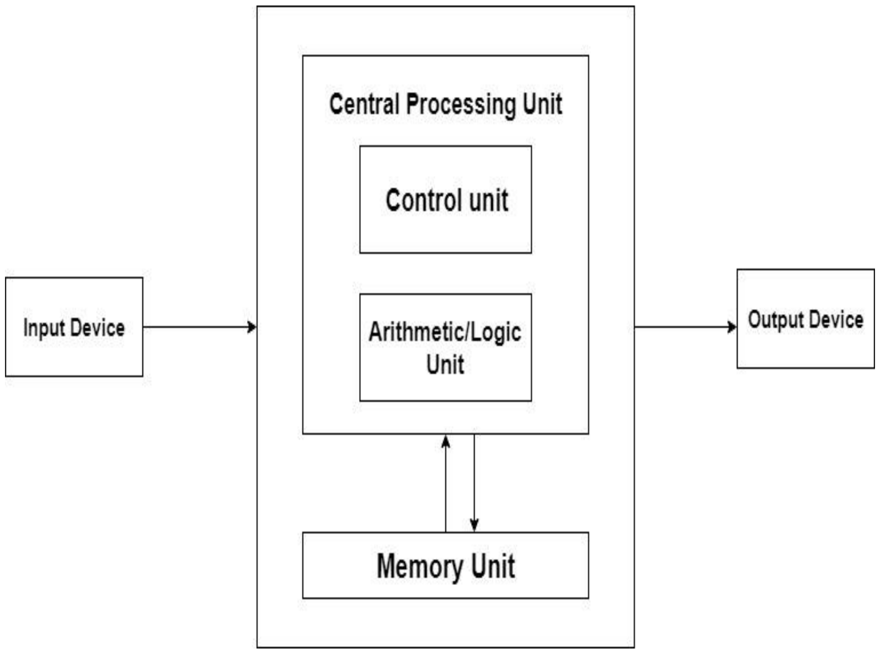


Figure 1: Von Neumann Architecture

- The three subsystems are connected to each other. The CPU, also called the central processing unit consists of the three major components of the architecture: The control unit, the arithmetic/logic unit (ALU) and different memory registers, making the memory unit.
- The control unit determines the order of operations to be executed as it controls the flow of instructions moving around the system.
- Arithmetic and logic units are responsible for mathematical and Boolean operations
- In the memory unit, there are registers present. Registers are temporary storage locations where the data and the instructions to be executed are stored.

- The memory unit is responsible to store data and instructions that needs to be executed. There are two types of memory present in this unit:
  (a) RAM (Random Access Memory) :
    1. Used in the normal operations of a computer, once the computer has started
    2. Can store multiple gigabytes of data.
    3. As they are temporary, they can be used in different devices to store information
    4. Writing data to a RAM is very fast

  (b) ROM (Read-Only Memory) :
    1. Used in the startup process of a computer
    2. Can store multiple megabytes of data
    3. As it is permanent, it stores the program needed for the initial computer startup process.
    4. Writing data to a ROM is very slow.
- The I/O interfaces receives and sends information from the computer's memory to the output devices. They also communicate with the user and the secondary storage (tape devices, pen drives etc.)
- Another subsystem that should be included is BUS. Bus is responsible to transfer data within the computer components and with the other computers. They are connected to the CPU by three different forms:
  (a) Control Bus: allows the CPU to communicate with the memory and the input and output devices or ports.
  (b) Data Bus: bidirectional bus that sends data to and from the chosen component.
  (c) Address Bus: after getting the instructions from the control bus, allocates the address of the memory or any input/output devices.

2. Machine Language:

Machine language, also known as a low-level programming language is the only form of language which is understood by the computers. Humans cannot communicate just with numbers. As machine language is a communication using numbers, hence high level programming languages were created. The high level programming languages are coded in English where later they are converted into values of 1 and 0 for the computer to understand.

- The codes are series of 1s, or 0s ("bits"), which are much of the time changed over both from and to hexadecimal (base 16) for human review and adjustment.
- Its codes change from PC to PC.

- It is easily executable by computers but they are hard for humans to understand.
- Machine language helps to understand the internal architecture of a computer.
- Some examples of machine language are provided below:

Machine Instruction	**Machine Operation**
00000000	Stop Program
00000001	Turn bulb fully on
00000010	Turn bulb fully off
00000100	Dim bulb by 10%
00001000	Brighten bulb by 10%
00010000	If bulb is fully on, skip over next instruction
00100000	If bulb is fully off, skip over next instruction
01000000	Go to start of program (address 0)

Table 1: Machine instruction and their corresponding operation for a light bulb

Programming Language Implementation

Programming Language Implementation: System of executing programs is called programming language implementation. The source code is translated into executable machine code through programming language implementation. There are three ways for programming language implementation:

1) Compilation
2) Pure interpretation
3) Hybrid implementation system

1. **Compilation:**

- The compilation process consists of a series of steps, which is primarily divided into the analysis and synthesis.
- Analysis phase is also called the front-end of the compiler.
- The analysis phase generates an intermediate code from the source code, undergoing several steps.
- Synthesis phase is called the back end of the compiler.
- The intermediate code is translated to final machine code in the synthesis phase.
- The compilation process is shown below:

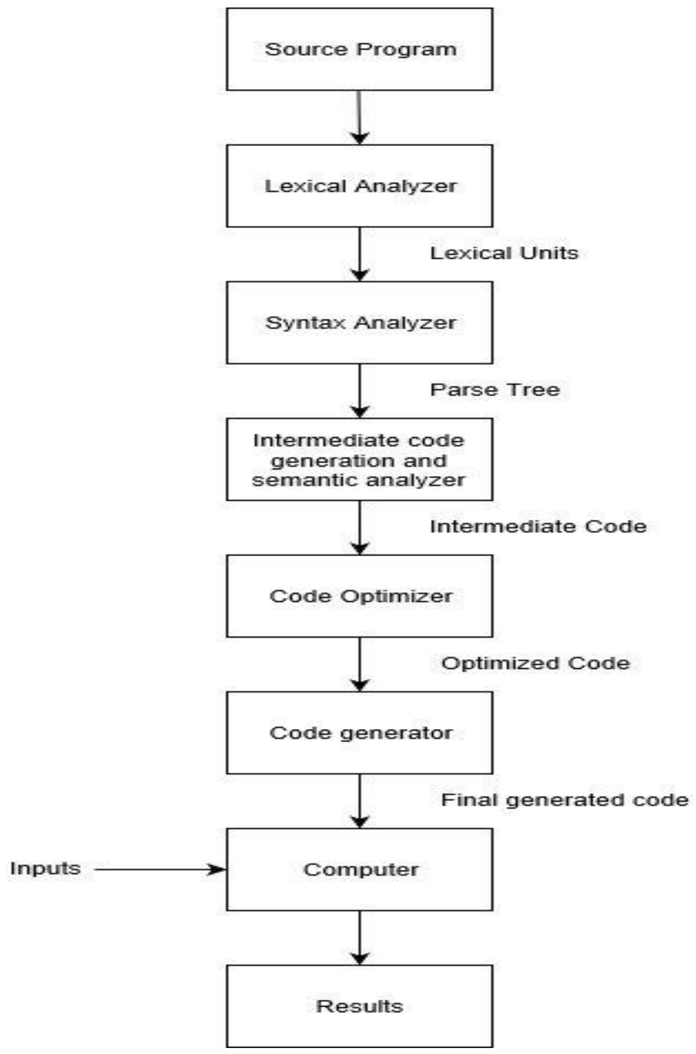


Figure 2: Compilation process

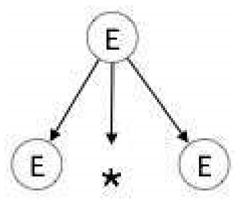
- The first step of the compilation process is lexical analysis, and is performed by lexical analyzer. A lexical analyzer tokenizes the whole source program. The characters from the source program are broken into several tokens, or, lexical units, known as lexemes. Identifiers, specific words, operators, punctuation symbols- all these characters are tokenized, while any comments are ignored. For example:

  Consider the following code:

  ```
  int main (){
  int a; //Comments will be ignored
  a=10;
  return 0;
  }
  ```

  For this code, the tokens are: ‘int’, ‘main’, ‘(‘, ‘)’, ‘{‘, ‘int’, ‘;’, ‘=’, ‘;’, ‘}’ .

- Next, the syntax analyzer uses the lexical units and forms a parse tree. This is the syntax analysis.
  For example:
  Consider the following input string:
  E→E*E
  For this input string, the following parse tree can be formed:



- Then, semantic analyzer carries out semantic analysis, which checks for both typing errors and grammars. Examples of semantic errors include undeclared variables, mismatch of data types etc.
- The intermediate code generator then generates an intermediate level code; that is, a code which is in between the source code and the machine code. It is a code which is a bit higher level than the assembly language, but lower level than the source code. For example, byte code is an intermediate form of code in Java.
- The code optimizer, then, optimizes the intermediate code to remove redundancy. Code optimization gets rid of unnecessary and redundant code, and generates an optimized code which reduces execution time.
- Finally, the code generator generates the machine code. Inputs may be needed at the time of execution.

Compilers are used by programming languages like C, C++.

Advantages of compiler:

- Compilers have better error detection mechanisms. Errors are detected in several stages, such as, lexical analysis, syntax analysis and also in semantic analysis.
- Compilers are fasters and have lower execution time. This is because compilers process the whole code and then execute it, instead of executing it line by line.

Disadvantages of compiler:

- Compilation takes up more memory space. This is because intermediate code is generated which consume extra memory space.
- Compilers are hardware specific.
- It is difficult to debug.

2. **Pure Interpretation:**

- In pure interpretation, another program interprets the source program/code.
- The source code is decoded by the interpreter, as it interprets statements line by line.
- If it faces any error in any of the statements, then it stops the interpretation and shows error.
- Pure interpreters are used by languages such as Ruby, PHP.

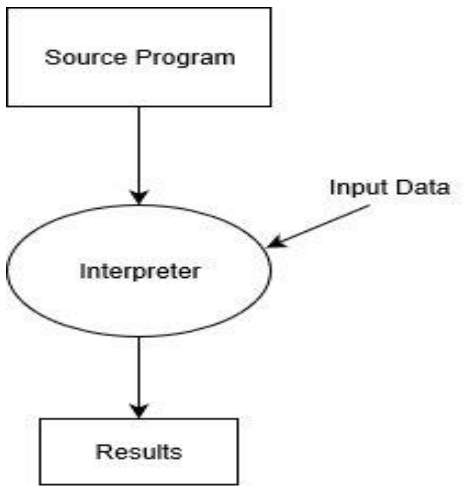


Figure 3: Pure Interpretation

Advantages of pure interpretation:

- Debugging is easier in pure interpretation as it executes statements line by line, so the specific location of error can be known for easy debugging.

Disadvantages of pure interpretation:

- The execution time is higher in pure interpretation. Interpreter executes the code line by line, so it takes longer to run in interpreter than it does in compiler.
- Takes up greater memory space. This is because during interpretation, symbol table must be present, which takes up more space.

3. **Hybrid Implementation System:** Hybrid implementation process is a combination of both compilation and interpretation. The source code is first translated to an intermediate form of code, which then, undergoes interpretation, which interprets the intermediate code. The steps of hybrid implementation are:

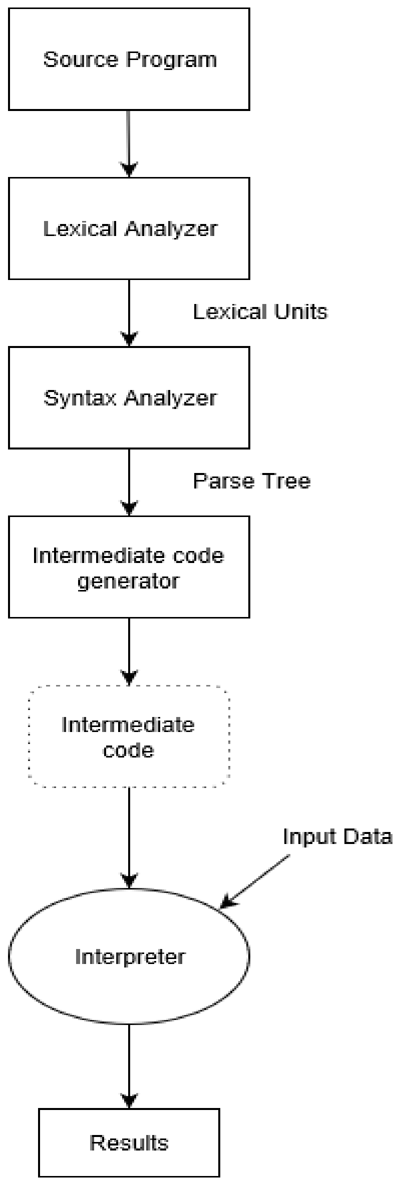


Figure 4: Hybrid Interpretation system

- The hybrid interpretation uses the steps followed in compilation to generate the intermediate code from the source code.
- Then, the intermediate code is interpreted using an interpreter, which then generates the results. Intermediate code is a form of code in between machine code and source code.
- For example, byte code is a form of intermediate code which can be executed by a virtual machine. Initially, Java used a hybrid interpretation approach of implementation, where the intermediate code was byte code.

Python is an example of a language that uses hybrid interpretation system

History of Programming Languages

Von Neumann Architecture: Famous mathematician John von Neumann, in 1945, conducted a thorough study of computer architecture that demonstrated that a computer could have a simple, fixed structure, but still be able to execute any kind of computation given properly programmed control without the need for hardware modification.

In short, he formed a new understanding of how practical fast computers should be created and organized. These ideas are referred to as the stored-program technique.

Stored-program technique:

- The stored-program technique was fundamental for future generations of high-speed digital computers and were universally adopted.
- The major advancement was the invention of a special type of machine instruction called conditional control transfer.
- Conditional control transfer allows a program sequence to be interrupted and reinitiated at any point – and by storing all instruction programs together with data in the same memory unit, so that, when desired, instructions could be arithmetically modified in the same way as data.
- As a result of these techniques offered by Von Neumann Architecture, computer programming became faster, and more efficient, with the block-code instructions in subroutines being used for several computational work.
- Frequently used subroutines had no need to be changed for each new program, but could be kept intact in "libraries" and read into memory when needed.
- Therefore, majority of a computer program could be created from the subroutine library. After the advantages of these techniques became known and understandable, the techniques soon became standard practice.
- The first generation of modern programmed electronic computers to make use of these improvements were launched in 1947.

Example of Library files (in C programming language):

To use the printf() function (to display a statement) , the header file <stdio.h> should be included at the start of the program. Other library file examples include math.h etc.

Short Code:

- Short Code was one of the earliest High-Level Languages developed for computers in 1949.
- It was the first computer language that represented mathematic expressions rather than a machine instruction, to instruct electronic devices.

- Initially Short Code represented expressions, the representation itself was not direct and required a process of manual conversion.
- Shortly after, Grace Murray Hopper developed the first compiler ever. The compiler allowed programs written in words and symbols to be easily converted to machine language.

Assembly Language:

- Assembly language is a machine specific low-level language.
- It is the language that the processor 'directly' understands.
- The assembly language uses mnemonic symbols for instructions and memory locations.
- Programs that are written in assembly languages are interpreted by an assembler, which translates the code written in assembly language into machine language form.

Some examples of mnemonic symbols used in Assembly Language:

- MOV – move data to and from memory and registers
- LD – load to the specified location

FORTRAN:

- FORTRAN is short for Formula Translation.
- It was formed in 1957 by John Backus to help shorten the process of programming.
- Invention of FORTRUN allowed computer programming to be more accessible.
- FORTRAN allowed the fast writing of computer programs that executed almost as efficiently as programs that had been laboriously hand coded in machine language.
- The formation of an efficient higher-level (or natural) language, allowed computer programming to move beyond a small scope.
- It was updated a several times (FORTRAN I, FORTRAN 77, FORTRUN 90 etc.) in the 1950s and 1960s in order to remain competitive with newer programming languages.

LISP:

- LISP (short for List Processing) is the second oldest high-level programming language, which is still in use today.
- Many dialects of LISP has existed since its inception, including Clojure, Common Lisp, and Scheme.
- It soon became the favored programming language for research on artificial intelligence.
- All LISP program codes are written as s-expressions, or parenthesized lists.

- A function call or syntactic form is written as a list with the function or operator's name first, and the arguments following.

Data Structure:

- Pure LISP has two different types of data structure: Atoms and Lists.
- Atoms are symbols or identifiers.
- Lists are stored as single linked list structures.

For example:

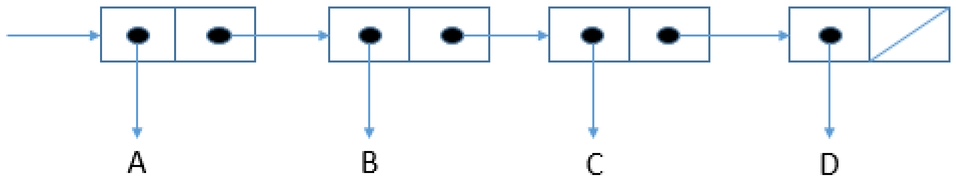


Figure 5: Representing List (A, B, C, D)

ALGOL:

- ALGOL (short for algorithmic language) was originally invented in the late 1950s, by a joint committee of American and European computer scientists.
- A second release by the group, known as ALGOL 60, became the standard version of the language for common use.
- It was created due to limitations of other contemporary languages, such as lack of portability, and difficulties in algorithmic communication.
- ALGOL’s syntax and structure directly influenced a number of other languages, including C.

Table of Programming Languages:

Programming Language Name	**Inventor**	**Year of invention**	**Use**	**Limitations**
LISP	John McCarthy	1950	It is mostly used in programming of artificial intelligence (AI)	LISP is a highly functional language, and so, it is quite difficult to learn.
Fortran	John Backus	1957	Used in industrial projects like design of airplane structures, factory automation, scientific data analysis etc.	Difficult to debug and detect errors. It is not a simple language, as it does not have several control structures and data structures, which aid in the simplification of programming languages.
ALGOL	Friedrich L. Bauer, Hermann Bottenbruch, Heinz Rutishauser, and Klaus Samelson	1958	ALGOL is used by computer scientists in research and scientific computing. It is also used as a standard method of creating algorithms by ACM in textbooks.	ALGOL has very complex syntax and structures. It is comparatively slower and less memory efficient. Due to the absence of its own variables, and input/output statements, it does not have efficient compilation.
C	Dennis M. Ritchie	1972	C programming is used in operating systems, robots, micro-controllers etc.	It is difficult to debug, and so error detection is difficult. C Compilers are unable to handle exceptions (run-time errors). C does not support Object-Oriented Programming

				features of encapsulation, inheritance etc.
Python	Guido van Rossum	1980	Python is used in the back-end development of many popular software.	Slower than other modern programming languages. Consumes higher memory space; not very suitable for mobile development.
Java	James Gosling	1995	Used to develop android apps.	Slow, with high execution time as the byte-code interpretation is slow. Consumes more memory than a lot of other languages.
C++	Bjarne Stroustrup	1998	Used in advanced software such as radar processing, flight simulator, and in operating systems and office applications.	Very weak in terms of security, and offers very low security. Very difficult to debug, not easy to find errors.
Swift	Chris Lattner	2014	Used for the development of IOS applications.	Swift can only be used for IOS and Linux. It is not compatible for all operating systems. It also creates compatibility issues with certain versions of IOS. Its speed and performance does not match the standards set by Apple.

References:

1. “Six Phases of the Compilation Process,” The Tech Pro, 09-Feb-2017. [Online]. Available: https://www.kttpro.com/2017/02/09/six-phases-of-the-compilation-process/. [Accessed: 24-Jun-2019].
2. “Compiler vs Interpreter: Complete Difference Between Compiler and Interpreter,” Meet Guru99 - Free Training Tutorials & Video for IT Courses. [Online]. Available: https://www.guru99.com/difference-compiler-vs-interpreter.html. [Accessed: 23-Jun-2019].
3. Ripunjay Tiwari, “Describe hybrid implementation system of a programming language.,” MP Study, 03-Jun-2016. [Online]. Available: http://mpstudy.com/describe-hybrid-implementation-system-of-a-programming-language/. [Accessed: 24-Jun-2019].
4. “Top Programming Languages and Their Real World Applications,” IEEE Transmitter, 01-Sep-2017. [Online]. Available: https://transmitter.ieee.org/top-programming-languages-real-world-applications/. [Accessed: 24-Jun-2019].
5. The Von Neumann Machine. [Online]. Available: http://www.ict.griffith.edu.au/~johnt/1004ICT/lectures/lecture07/Cragon-pp1-13.html. [Accessed: 23-Jun-2019].
6. T. E. of E. Britannica, “FORTRAN,” Encyclopædia Britannica, 17-May-2019. [Online]. Available: https://www.britannica.com/technology/FORTRAN. [Accessed: 24-Jun-2019].
7. “What is Assembly Language?,” Computer Hope, 04-Nov-2017. [Online]. Available: https://www.computerhope.com/jargon/a/al.htm. [Accessed: 24-Jun-2019].
“Informit”, InformIT. [Online]. Available: http://www.informit.com/articles/article.aspx?p=1077906. [Accessed: 24-Jun-2019].
8. V. Beal, “machine language,” What is Machine Language? Webopedia Definition. [Online]. Available: https://www.webopedia.com/TERM/M/machine_language.html. [Accessed: 23-Jun-2019].
9. “CSE 505 Lecture Notes:” Algol. [Online]. Available: https://courses.cs.washington.edu/courses/cse505/99au/imperative/algol.html. [Accessed: 24-Jun-2019].
10. Tutorialspoint.com, “Compiler Design - Syntax Analysis,” *www.tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/compiler_design/compiler_design_syntax_analysis. [Accessed: 20-Jul-2019].