CMSC 124

DESIGN AND IMPLEMENTATION OF PROGRAMMING LANGUAGES

CNM PERALTA

LANGUAGE IMPLEMENTATION METHODS

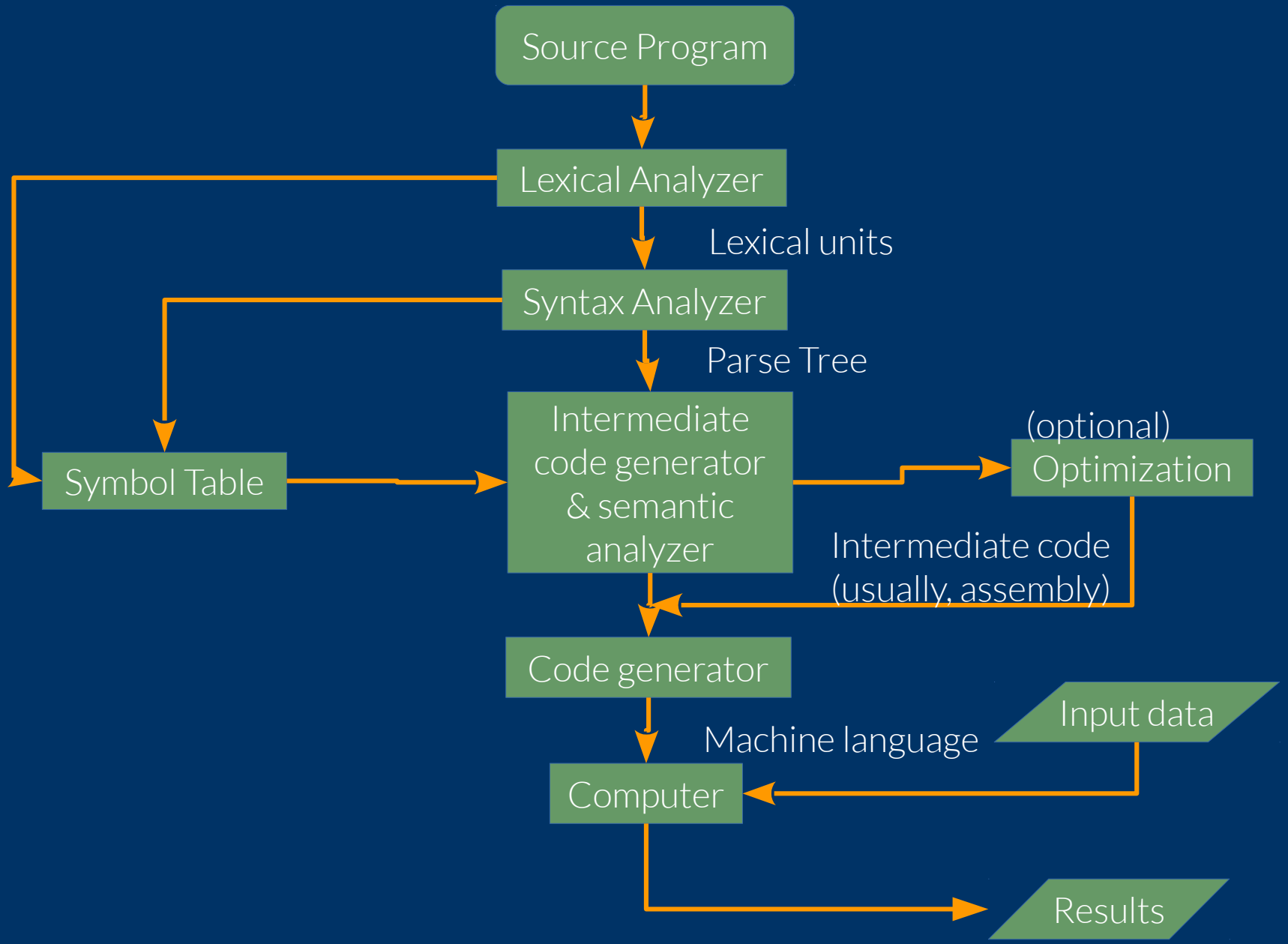
There are three general methods for implementing programming languages.

1.

Compilation

Programs are translated to machine language.

The **advantage** of compilation is that it is the **fastest** of the three methods of implementation.


Source Program
Lexical Analyzer
Lexical units
Syntax Analyzer
Parse Tree
Intermediate code generator & semantic analyzer
(optional)
Optimization
Symbol Table
Intermediate code (usually, assembly)
Code generator
Input data
Machine language
Computer
Results

Linkers

Link (or load) user programs to required operating system programs during compilation by inserting their addresses into the user program.

Links can also be made to other user programs, e.g., program libraries.

The most prominent problem in compilation is the von Neumann bottleneck.

von Neumann Bottleneck

Connection between the memory and the processor; instructions are executed faster than they can be moved to the processor.

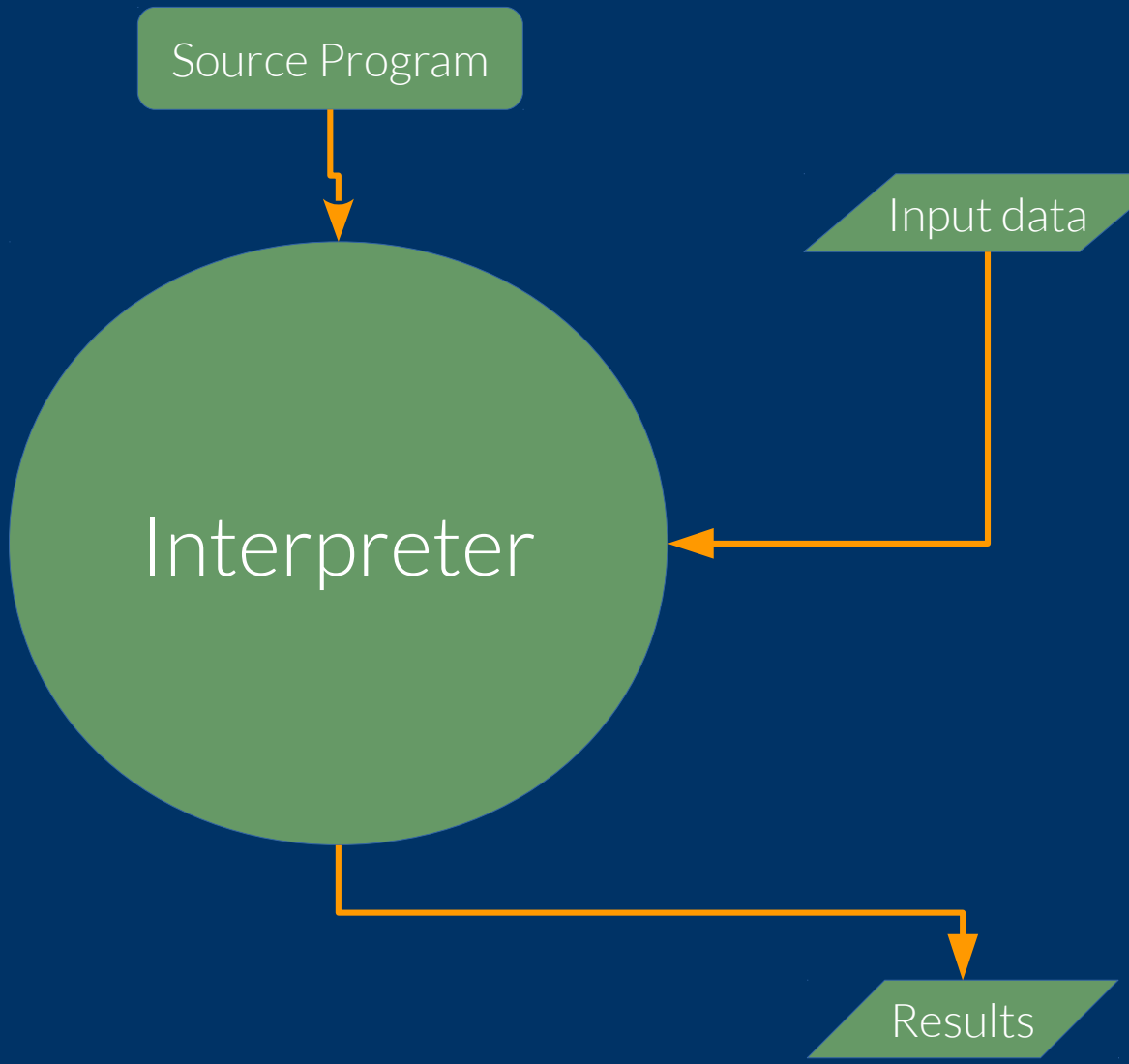
In general, it is the primary limiting factor of von Neumann architecture computers.

Examples of compiled are languages C, C++, FORTRAN, etc.

2.

Interpretation

Programs are interpreted by an interpreter, and no translation of the user program is done.


Source Program
Input data
Interpreter
Results

In interpretation, programs are directly executed statement by statement, allowing source-level debugging.

However, **execution time** is **10 to 100 times slower** than compiled languages.

The bottleneck in pure interpretation is the decoding of high-level language statements.

EXAMPLE

Loops may require the execution of the same statements many times:

```
while(someCondition) {
  print “What is your grade in ”, course;
  grade = <>;
}
```

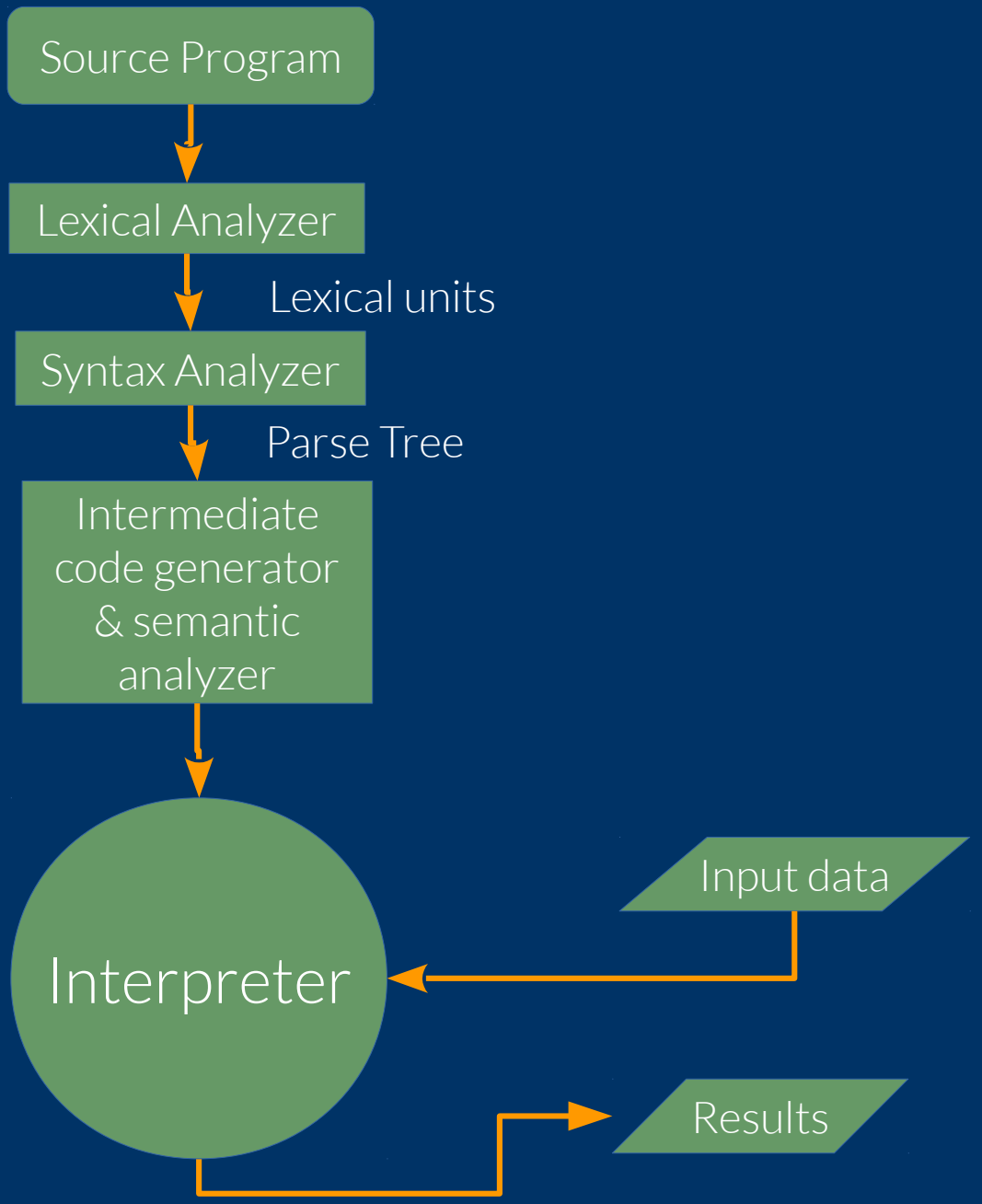
Interpreted languages also have the added disadvantage of needing more space to store the symbol table; moreover, code optimization can't be done.

Examples of interpreted languages include LISP, APL, SNOBOL, Javascript, PHP, etc.

3.

Hybrid Implementation Systems

Compromise between compilation and pure interpretation.


Source Program
Lexical Analyzer
Lexical units
Syntax Analyzer
Parse Tree
Intermediate code generator & semantic analyzer
Interpreter
Input data
Results

Source programs are translated to an intermediate language that can easily be interpreted.

Definitely faster then pure interpretation because high-level statements only need to be decoded once.

Examples of languages that use hybrid implementation systems are Perl and early implementations of Java.

Preprocessors

Programs that process a program before it is compiled.

EXAMPLE

```
#include “myLib.h”
```

The contents of myLib.h are added to the program before it is compiled.

EXAMPLE

```
#define max(A,B) ((A)>(B)) ? (A):
(B))
```

…

```
x = max(2,3);
```

will become `x = ((2)>(3)) ? (2):(3)`