



Compiler Construction

CS-471

By Laeeq Khan Niazi

Teacher Introduction

- I am **Muhammad Laeeq Khan Niazi** from Mianwali
- I have done BS and MS in computer science
- Currently doing PhD in computer Science
- I am a professional software developer and work with various technologies
- Email: m.laeeq.niazi@gmail.com
- LinkedIn: <https://www.linkedin.com/in/laeeqkhanniazi/>
- GitHub : <https://github.com/Laeeq-Khan>

CLOs

- Understand and Apply Mathematical Formalisms like Regular Expressions, Grammars, FAs & PDAs.
- Understand the working principles of various phases of a compiler and how they are integrated to produce a working.
- Compare Various Implementations Techniques of Phases of Compiler, in particular Lexical Analyser, Parser
- Be able to work in a team (preferably alone) to design, develop, test, and deliver analysis phases of a compiler

Books



- Compiler – Principles, Techniques and Tools by Aho, Sethi and Ullman
- Engineering a Compiler – Keith D. Cooper & Linda Torczon 2nd Edition

Students Introduction



How to pass this course?

Be regular in classes and do all practical assignments



Does this course involve programming?



What is difficulty level of this course

Difficulty level of this course is medium. Its not that easy not that hard



Course Structure



- Theory 40%
- Assignments 15%
- Project 40%
- Attendance 5% (full marks if attendance is more than 90%)

Outcome of this course

- Understanding theory behind the compiler
- Implementation of your own compiler



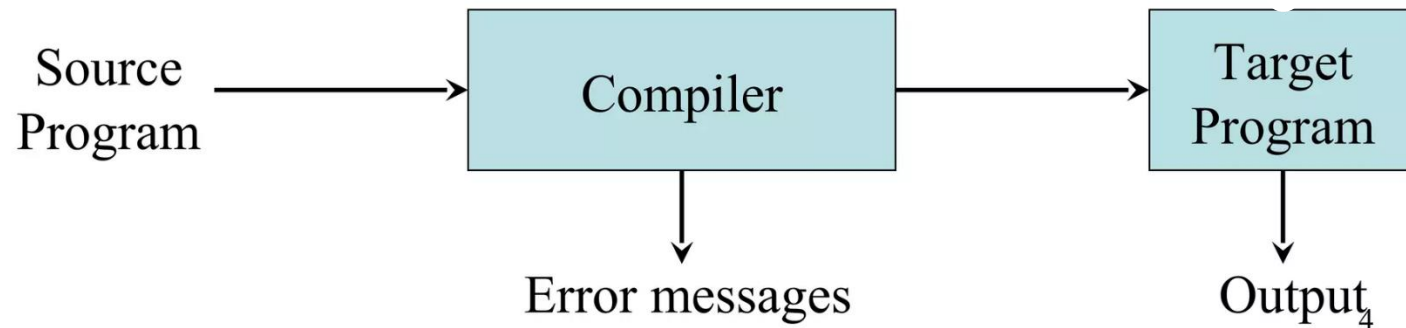


Why study compilers?

- Application of a wide range of theoretical techniques
 - Data Structures
 - Theory of Computation
 - Algorithms
 - Computer Architecture
- Good Software Engineering Experience
- Better understanding of programming languages

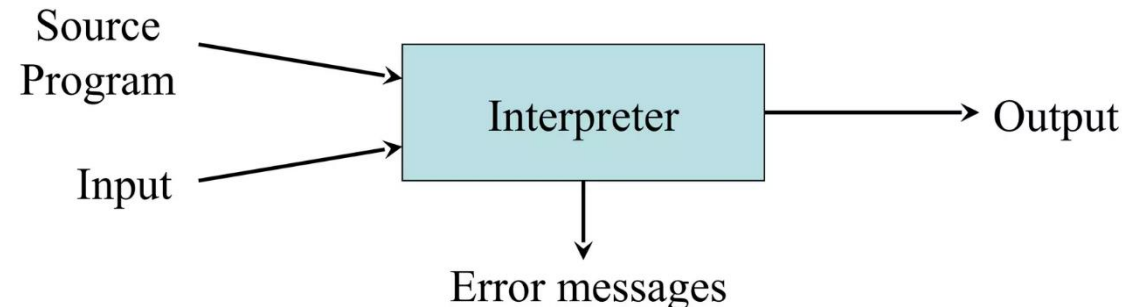
What is Compiler?

- A compiler translates a program from a source language into an equivalent program in a target language, such as machine code. It also checks for and reports any errors in the source code. C++ is a language that uses a compiler.
- C++ uses compiler



What is Interpreter?

- An interpreter is a program that reads a source program (often written in a high-level language) and directly executes the instructions without producing a separate target program (like machine code). Instead of generating an intermediate or final machine code, the interpreter performs the operations specified by the source program line by line or statement by statement.
- Python uses the interpreter



Just-In-Time (JIT) compilation

- It is a hybrid approach that combines elements of both compilers and interpreters.
- **Compilation Aspect:** Like a compiler, JIT translates code from a high-level language into machine code. However, this translation occurs at runtime, just before the code is executed, rather than ahead of time.
- **Interpretation Aspect:** Similar to an interpreter, JIT compiles and executes code on-the-fly, during the execution of the program.



Some Random Questions

- With C and C++, we generate an .exe file. Does it contain machine code or intermediate code?
- With C#, we also generate an .exe file. Does it contain intermediate code or direct machine code?
- What about the output file created by Java? Is it a .exe file? Is it directly executable by the CPU?

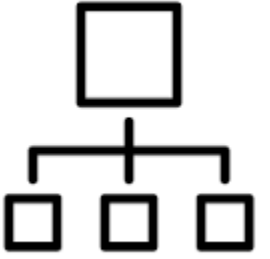
Answers



- **Answer 1:** The .exe file generated by C and C++ contains machine code. This machine code is directly executable by the CPU without further translation.
- **Answer 2:** The .exe file generated by C# typically contains Intermediate Language (IL) code, not direct machine code. The IL code is JIT-compiled into machine code by the .NET runtime (CLR) at runtime. In some scenarios, Ahead-Of-Time (AOT) compilation may be used, which can include machine code in the file.
- **Answer 3:** Yes, the output file created by Java is typically a .jar (Java ARchive) file. A .jar file is an archive that can contain compiled Java bytecode (.class files), which is intermediate code. This bytecode is executed by the Java Virtual Machine (JVM), which interprets or JIT-compiles it into machine code at runtime.

Classification of Compiler

- Single Pass Compiler
- Two Pass Compiler
- Multipass Compiler



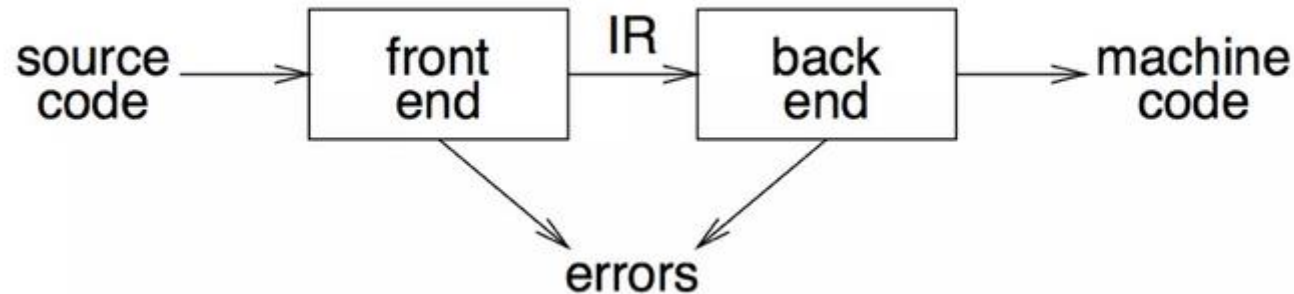
Single Pass Compiler

- Source code directly transforms into machine code

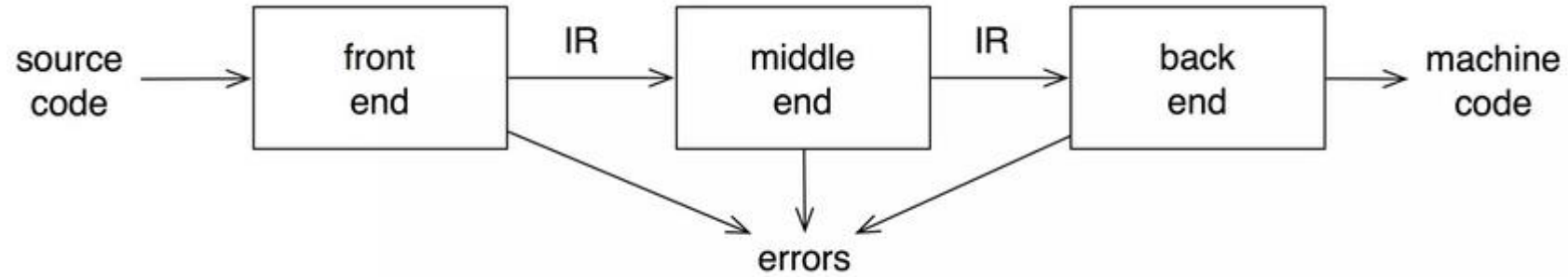


Two Pass Compiler

- Intermediate representation
- Frontend maps legal code into IR
- Backend maps IR onto target machine
- Simplify retargeting
- Allows multiple frontends



Multipass Compiler



Phases of Compiler

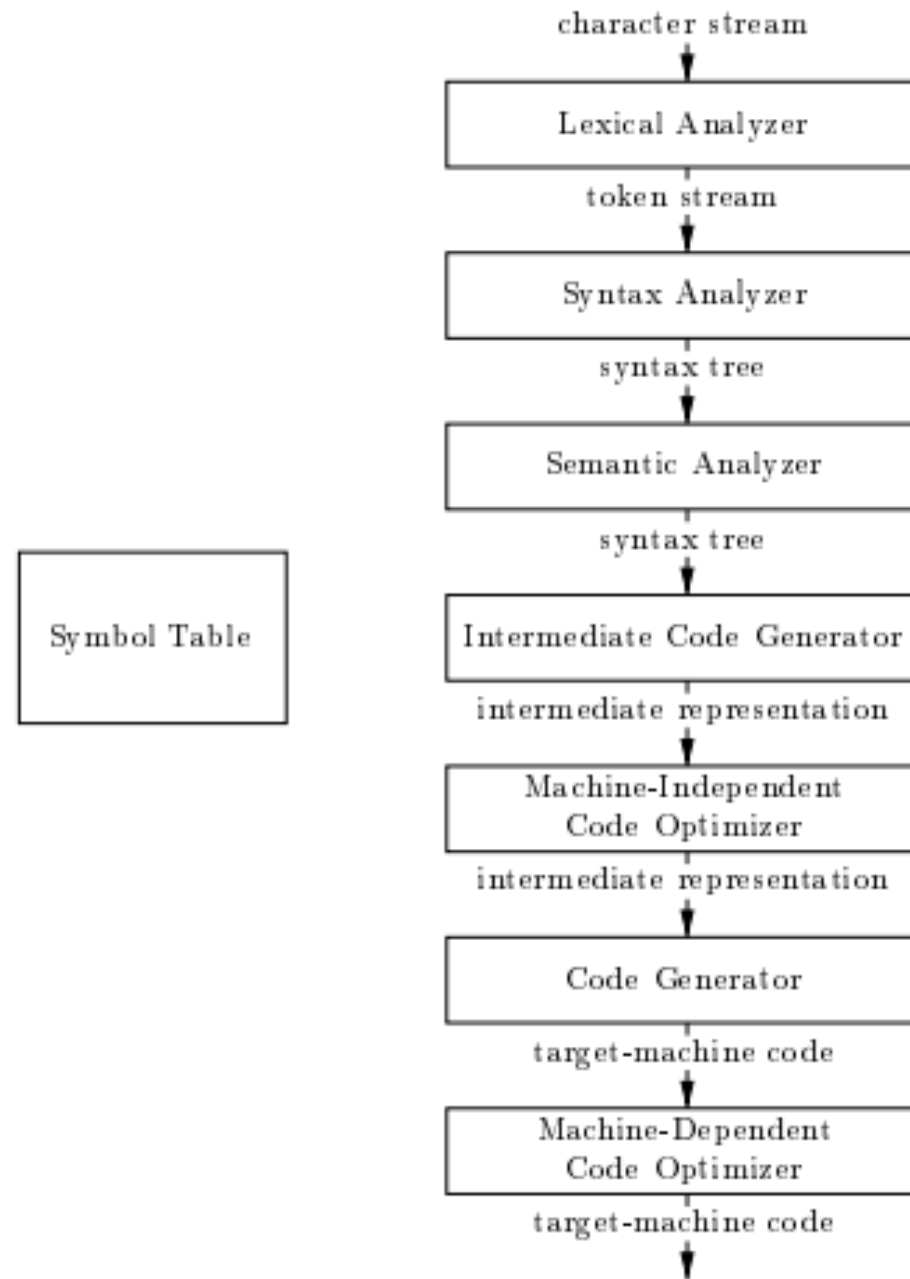


Figure 1.6: Phases of a compiler

Do C++ compilers like GCC or MinGW include only the compiler, or do they also include the assembler?

- Both contain a comprehensive suite that includes the compiler, assembler, linker, and other tools. When you install GCC, it usually includes the GNU Assembler (as), which is the assembler used in the GCC toolchain.
- The assembler is automatically invoked by the compiler as part of the build process, ensuring that the entire compilation and linking workflow is handled seamlessly.

C++ Code

```
void myfunction(int x){  
    int output = x*x*x + 2 ( x/3);  
    return output;  
}
```

Assembly Code

```
myfunction:
    # Prologue
    pushl    %ebp                # Save base pointer
    movl     %esp, %ebp         # Establish a new base pointer

    # Function body
    movl     8(%ebp), %eax       # Move the argument x into register %eax
    imull    %eax, %eax          # Compute x*x, store in %eax
    imull    8(%ebp), %eax       # Compute x*x*x, store in %eax

    movl     8(%ebp), %edx       # Move x into register %edx
    movl     $0x3, %ecx         # Load constant 3 into %ecx
    cdq                      # Sign extend %eax to %edx
    idivl    %ecx               # Divide %edx:%eax by 3, quotient in %eax, remainder in %edx

    addl     %eax, %eax          # Double the result, i.e., 2*(x/3)
    addl     %eax, %edx          # Add x*x*x + 2*(x/3)

    # Epilogue
    movl     %edx, %eax          # Move the result to %eax (return value)
    leave    # Clean up stack and restore base pointer
    ret                        # Return to caller
```


ToDo

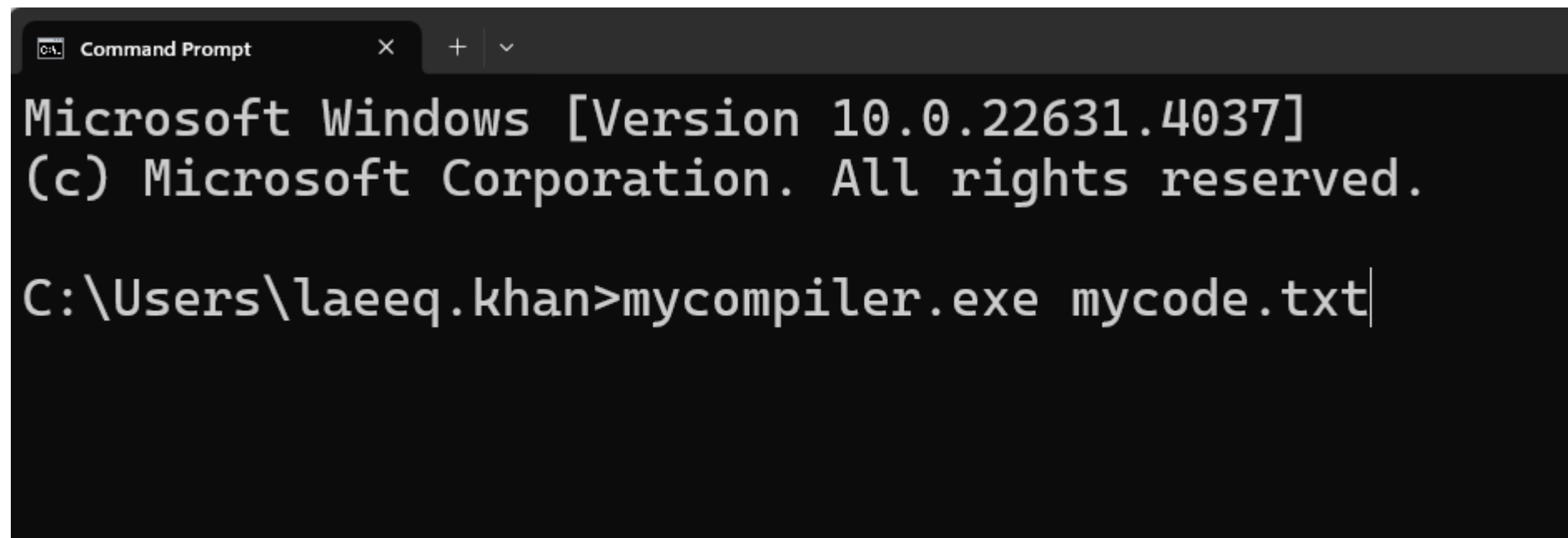
- Setup C++ Compiler
- Install IDE VS Code or any of your choice
- Write a C++ program, compile and execute it

Lab Task1

- What is the logical difference between a compiler and an interpreter, and which one is better in different situations?
- What is JIT, and in what conditions does it work better than a compiler and when is it not as good as a compiler?

Lab Task2

- Make your mycompiler.exe file it will take the filename.txt file as argument from cmd and it will read all the text available in this file and display output on the screen



```
Command Prompt
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\laeeq.khan>mycompiler.exe mycode.txt
```