CSB 353: Compiler Design

CS C++ Mini Compiler

PROJECT SYNOPSIS

Submitted To:

Dr. Shelly Sachdeva

NATIONAL INSTITUTE OF TECHNOLOGY DELHI

Department of Computer Science and Engineering

2023

CS C++ Mini Compiler

PROJECT SYNOPSIS

Title

The intended goal of this project is to create a c++ compiler that could interpret conditional statements like If-Else statements and Switch-Case constructs and output the results at each stage of compilation before producing 8086 assembly code as a final output.

Introduction

A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or 'code' that a computer's processor uses.

The intended goal of this project is to create a c++ compiler that could interpret conditional statements like If-Else statements and Switch-Case constructs and output the results at each stage of compilation before producing 8086 assembly code as a final output.

In order to increase the project's success, it is necessary to analyze the methods and tactics presently being implemented.

Team Members

Deepak Sharma: 201210014@nitdelhi.ac.in
Anavi Somani: 201210005@nitdelhi.ac.in
Atul Goyal: 201210011@nitdelhi.ac.in

FUNCTIONALITY

INPUT(C++ Code)

```cpp
int age;
if(age < 18)
{
    cout<<"Child\n";
}
else
{
    cout<<"Adult\n";
}
```

CS C++ Mini Compiler

OUTPUT
(Assembly language Code)

```asm
_start:
    ; load age into register eax
    mov eax, [age]

    ; compare age to 18
    cmp eax, 18

    ; if age < 18, jump to label "child"
    jl child

    ; otherwise, jump to label "adult"
    jmp adult

child:
    ; print "Child" to standard output
    mov eax, 4
    mov ebx, 1
    mov ecx, child_msg
    mov edx, 6
    int 0x80

    ; exit the program
    mov eax, 1
    xor ebx, ebx

adult:
    ; print "Adult" to standard output
    mov eax, 4
    mov ebx, 1
    mov ecx, adult_msg
    mov edx, 6
    int 0x80

    ; exit the program
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

Functionality

1. The compiler will go through several phases of compilation, starting with the lexical analysis, where the input code will be broken down into a series of tokens. These tokens will then be processed by the syntax analyzer, which will check whether they conform to the rules of C++ syntax.
2. After the syntax analysis, the compiler will perform semantic analysis to ensure that the program is semantically valid. This phase will involve type-checking, variable scoping, and other checks to ensure that the program is free of errors that could result in runtime crashes or incorrect output.
3. Once the program passes the semantic analysis phase, the compiler will generate the intermediate code, which is a low-level representation of the input code. This intermediate code will be optimized to improve the efficiency of the compiled program.
4. Finally, the compiler will generate the machine code that can be executed by the target machine. The machine code will be optimized to reduce the amount of memory and processing power required to run the program.

Throughout the development of the compiler, we will ensure that it can handle a wide range of input code and produce output that is consistent with the behavior of standard C++ compilers. Additionally, we will focus on producing high-quality documentation and testing to ensure that the compiler is reliable and easy to use.

Requirements

Hardware Requirements:

- 64 BIT MACHINE, 800 MHz PROCESSOR 1.66GHz
- WINDOWS/LINUX OS
- 128 MB RAM

Software Requirements:

- LEX TOOL
- YACC PARSER TOOL
- TURBO C/GCC

Feasibility and Future Scope

With the development of technology, productivity is prioritized but the foundation of programming and coding remains the same although many have moved away from C, C++, and other programming languages in favor of python, etc

An automatic tool for translating computer programmes from one language to another is called a compiler. It converts the source code provided as input into executable machine code. The input language is frequently one that a particular computer cannot directly execute, for instance, because the language is intended to be human-readable. The output language is frequently one that a certain machine can perform immediately.

A compiler is never required to run a programme. Converting a HighLevel Language application to machine executable code is only a middle step.

The project development will begin immediately, but we will need to analyze it every month to ensure that we can all achieve the success peak of this development project together.