C-COMPILER

- Tanay Srivastava
- Viswesh Bhaskara
- Venkatesh Babu Jagarlamudi
- Amey Kulkarni

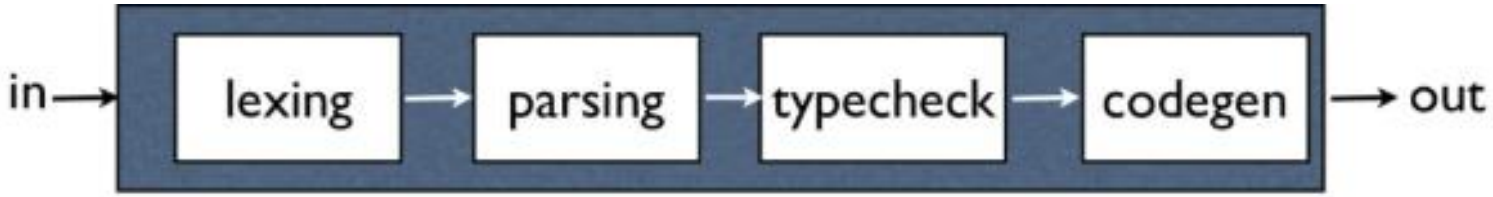
What is a Compiler?

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

Steps

- Language statements are written in a language (C or C++)one line at a time using an editor.
- The file that is created contains what are called the source statements.
- The programmer then runs the appropriate language compiler, specifying the name of the file that contains the source statements.

Compiler Design



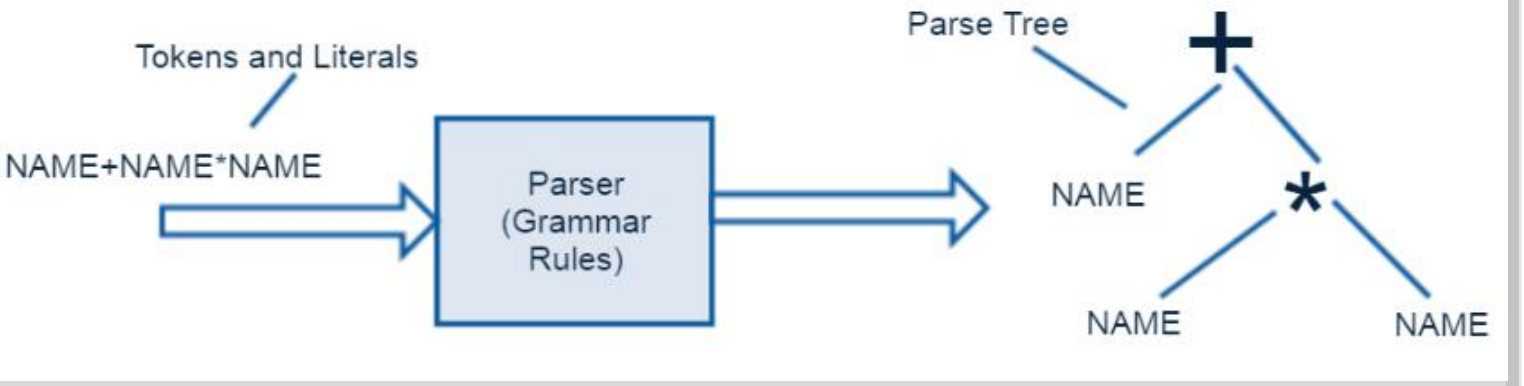
- **Compiler broken into stages.**
- **Lexing /Parsing are input related.**
- **Type Checking is error validation.**
- **Code generation generates machine understandable code.**


```
b = 40 + 20*(2+3)/37.5
NAME = NUM + NUM * ( NUM + NUM ) / FLOAT
```

```
b = 40 * $5
        ^
Illegal Character
```

Lexing

- Splits input text into tokens and detects illegal symbols.




```
b = 40 + "hello"      (Syntax OK)
b = 3 * 4 7 /         (Syntax error)
```

```
b = 40 + "hello"      (???)
```

Parsing

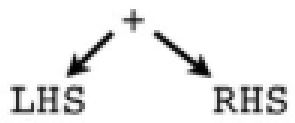
- Grammar rules are defined within functions.
- Tokens are imported from the lexer.
- Detects Syntax Errors
- If a program parses, it is at least well-formed
- Still don't know if program is correct or not.

Type Checking

- Enforces underlying rules

```
b = 40 + 20*(2+3)/37.5      (OK)
c = 3 + "hello"             (TYPE ERROR)
d[4.5] = 4                  (BAD INDEX)
```

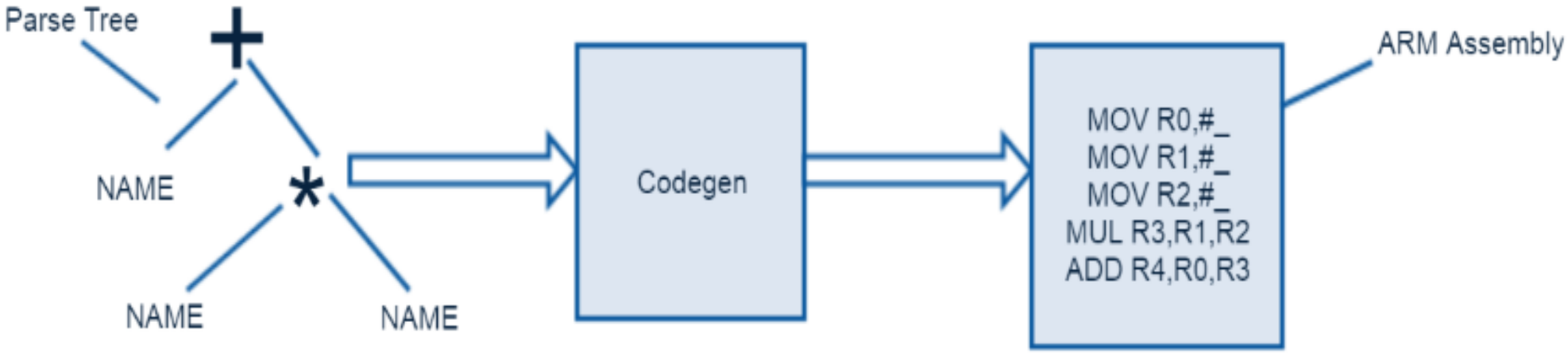
- Ex:+ operator



1. LHS and RHS must be the same type
2. If different types, must be convertible to same type

Code Generation

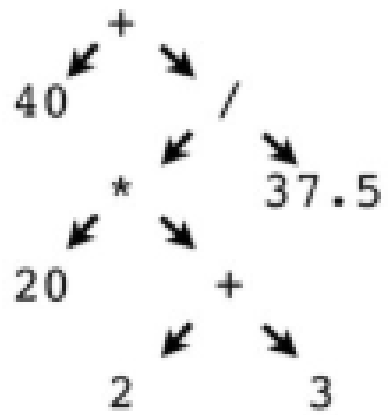
- Assembly instructions specific to ARM Cortex M4 are generated and written to an assembly file



Traversing of a parse tree

```
b = 40 + 20*(2+3)/37.5

LOAD R1, 40
LOAD R2, 20
LOAD R3, 2
LOAD R4, 3
ADD  R3, R4, R3   ; R3 = (2+3)
MUL  R2, R3, R2   ; R2 = 20*(2+3)
LOAD R3, 37.5
DIV  R2, R3, R2   ; R2 = 20*(2+3)/37.5
ADD  R1, R2, R1   ; R1 = 40+20*(2+3)/37.5
STORE R1, "b"
```



Tools used &Implementation:

Type	Name
Langauge	Python
Package	PLY
Lexer	Lex
Parser	Yacc

PLY stands for Python Lex and Yacc. It is a Python version of Lex and Yacc that has the same functionality as Lex and Yacc but has a different interface with ample support for debugging. Simply put, it provides an easy way to write a compiler.

Usage:

```
python compiler.py input.txt output
```

Here input.txt file is the text file containing the C Expressions output is the name of the file where the result must be generated. The file generated will be a '.s' file so you need not add .s to the target file

THANK YOU