ARM Compiler

SVR ADITYA REDDY IMT2014047

PRASHANTHI SK MT2016520

K DEEPIKA RAJ MT2016529

Problem Statement

- Read expressions and statements from a text file
- Generate ARM assembly instructions that can be run directly on Keil uVision IDE. The assembly instructions are specific to a generic ARM Cotrex M4 device/board.

Tools used

Langauge	:	Python
Package	:	PLY
Lexer	:	Lex
Parser	:	Yacc

Implementation

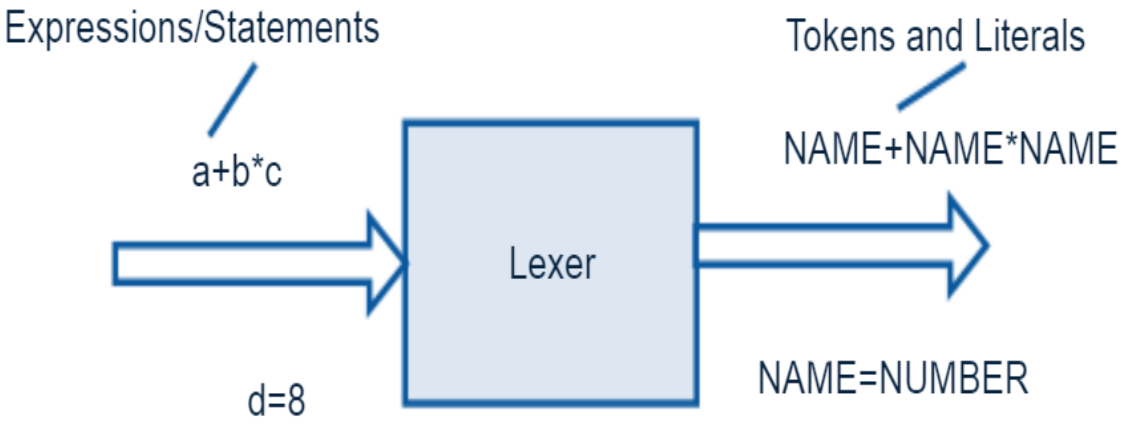
- PLY stands for Python Lex and Yacc.

Why PLY?

- A powerful, easy-to-use tool
- PLY provides extensive diagnostics[1]
- PLY provides most of the standard lex/yacc features including support for empty productions, precedence rules, error recovery, and support for ambiguous grammars.[1]
- It uses LR-parsing which is reasonably efficient and well suited for larger grammars.

Lexing

- Tokens are defined either using regular expressions or functions in the code for the lexer.
- The lexer splits up the input file into these tokens.



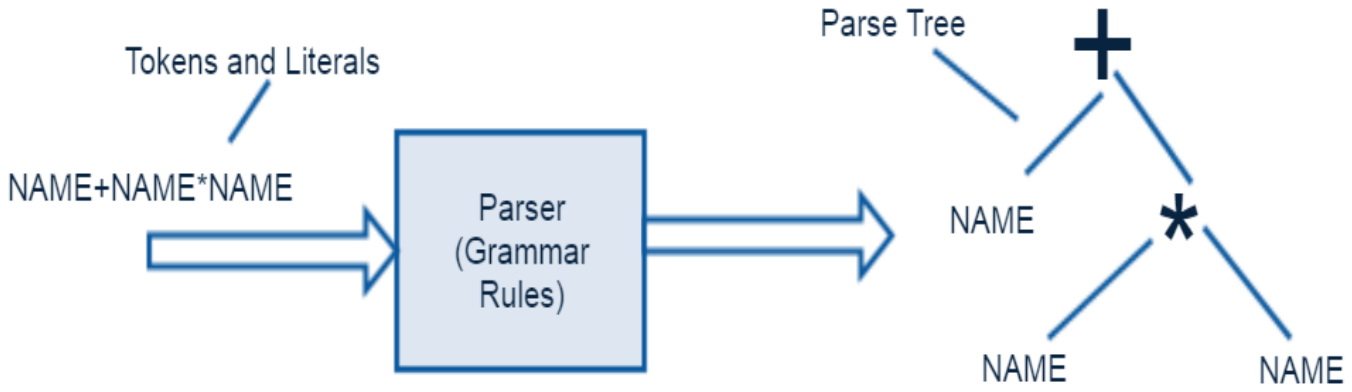
Lexing--Example

```
t_NAME = r’[a-zA-Z_][a-zA-Z0-9_]*’


def t_NUMBER(t):
r’\d+’ t.
value = int(t.value)
return t
```

Parsing

- Grammar rules are defined within functions.
- Tokens are imported from the lexer.
- PLY uses LR parsing aka Shift Reduce parsing.
- Results propagate up through the grammar in a bottom up fashion.



Parsing--Example

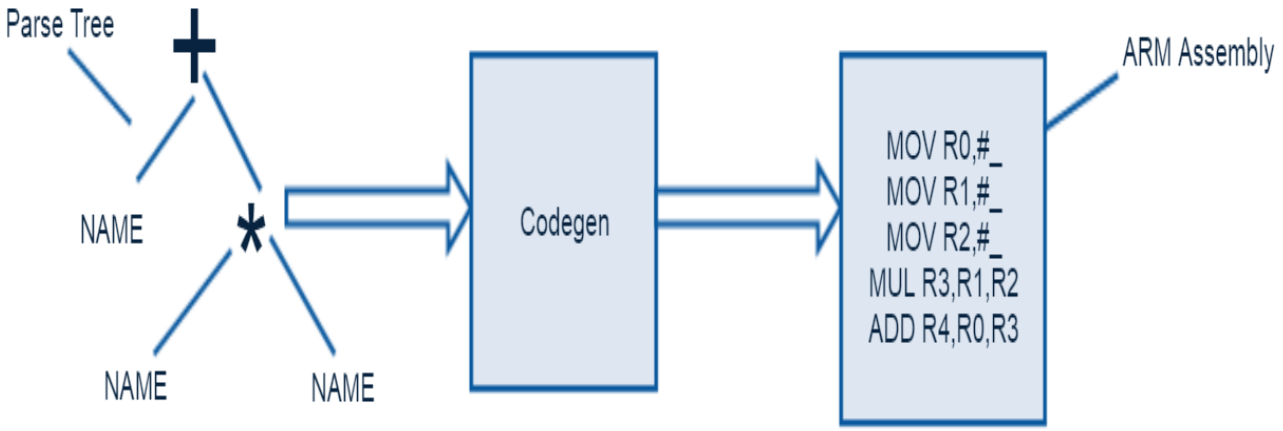
assign : NAME EQUALS expr

expr :NUMBER

```
def p_expr(p):
'''expr : expr PLUS expr | expr MINUS expr '''
```

Code Generation

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



References

[1] http://www.dabeaz.com/ply/PLYTalk.pdf

[2] https://github.com/dabeaz/ply/