Report of ARM Architecture Course Project

ARM Compiler

Guided by
Prof. Girish Kumar

Submitted by
SVR Aditya Reddy IMT2014047
Prashanthi SK MT2016520
K Deepika Raj MT2016529

Problem statement

To read expressions and statements from a text file and 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.

Abstract

Expressions from the input source are read. Lexical analysis is performed on the expressions to obtain the appropriate tokens. Thereafter, parsing is done using pre-defined grammar rules to generate the specific assembly instructions. These instructions are written to a .s file and can be simulated/debugged on uVision.

Tools used

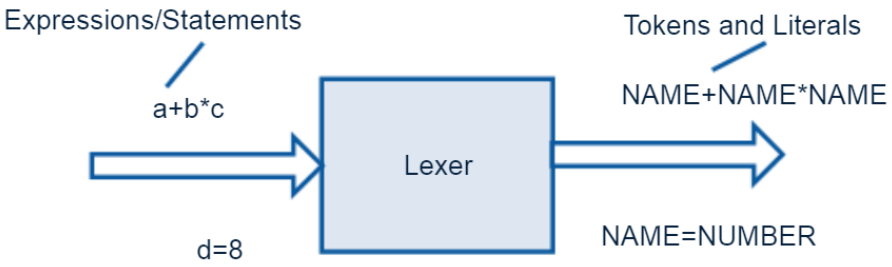
Langauge	:	Python
Package	:	PLY
Lexer	:	Lex
Parser	:	Yacc

Implementation

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.

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.



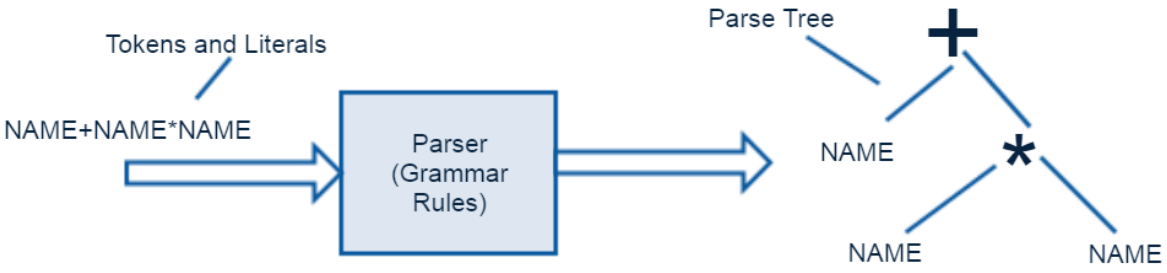
Example:

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.

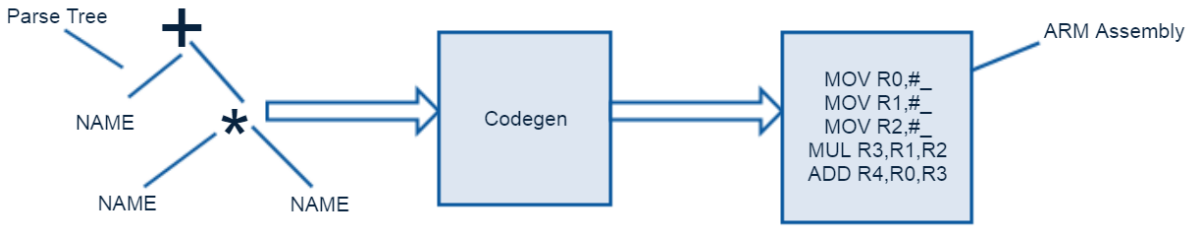


Example:
assign : NAME EQUALS expr
expr :NUMBER

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/