Cairo University

Faculty of Engineering

Department of Computer Engineering

Languages and Compilers

Project Phase 2
Team 4

Remonda Talaat Eskarous
SEC:1, BN:19

Mohamed Shawky Zaky
SEC:2, BN:15

Mohamed Ahmed Mohamed Ahmed
SEC:2, BN:10

Ahmed Mohamed Zakaria
SEC:1, BN:3

1 Project Overview

In this project, a simple `C/C++` compiler is built using basic constructs. `Lex`, `Yacc`, `C` and `C++` are used to build the product. Our tool takes a code file as an input, parses it and outputs the corresponding assembly code, a list of syntax and semantic errors and the corresponding symbol table. Moreover, our tool is developed and tested on *Ubuntu*.

2 Utilized Tools and Technologies

- **Lexer :** `Flex`
- **Parser :** `Bison`
- **Compilation and symbol table :** `C/C++`
- **GUI :** `Python`, `PyQT5`

3 Bonus Features

We implemented the following feature as bonus :

- Nested scopes and block structures.

4 List of Tokens

Token	Description
IF / ELSE	keywords of if/else statements
WHILE / DO / FOR	keywords of loops statements
BREAK	breaking out of a loop
SWITCH / CASE / DEFAULT	keywords for switch statements
RETURN	return from functions
INT_TYPE / FLOAT_TYPE / STRING_TYPE / CHAR_TYPE / BOOLEAN_TYPE	data types tokens
CONST	constant token
VOID	no return type token for functions
EQEQ	==
NOTEQ	!=
G	>
L	<
GE	>=
LE	<=
AND	&&
OR	||
NOT	!
ASSIGNMENT	=
PLUS	+
MINUS	-
MUL	*
DIV	/
MOD	%
BOOLEAN_TRUE	true boolean value
BOOLEAN_FALSE	false boolean value
VARIABLE	identifier name token
STRING	string value token
CHAR	character value token
INTEGER	integer value token
FLOAT	float value token

5 List of Language Production Rules

- **type :**
  - INT_TYPE
  - FLOAT_TYPE
  - CHAR_TYPE
  - BOOLEAN_TYPE
  - STRING_TYPE
- **stmt :**
  - expr ';'
  - type VARIABLE ';'
  - type VARIABLE ASSIGNMENT expr ';'
  - CONST type VARIABLE ASSIGNMENT expr ';'
  - VARIABLE ASSIGNMENT expr ';'
  - WHILE '(' expr ')' stmt
  - DO stmt WHILE '(' expr ')'
  - FOR '(' VARIABLE ASSIGNMENT expr ';' expr ';' VARIABLE ASSIGNMENT expr ')' stmt
  - IF '(' expr ')' stmt %prec IFX
  - IF '(' expr ')' stmt ELSE stmt
  - SWITCH '(' VARIABLE ')' '{' case_list case_default '}'
  - BREAK ';'
  - type VARIABLE func_list '{' func_stmt_list '}'
  - VOID VARIABLE func_list '{' stmt_list '}'
  - VOID VARIABLE func_list '{' '}'
  - '{' stmt_list '}'
  - '{' '}'
- **stmt_list :**
  - stmt
  - stmt_list stmt
- **case_list :**
  - case_list CASE INTEGER ':' stmt_list
  - case_list CASE CHAR ':' stmt_list
  - case_list CASE STRING ':' stmt_list
  - case_list CASE BOOLEAN_FALSE ':' stmt_list

- case_list CASE BOOLEAN_TRUE ':' stmt_list

- **case_default :**
  - DEFAULT ':' stmt_list

- **expr :**
  - INTEGER
  - FLOAT
  - CHAR
  - STRING
  - BOOLEAN_TRUE
  - BOOLEAN_FALSE
  - VARIABLE
  - MINUS expr %prec UMINUS
  - NOT expr
  - expr PLUS expr
  - expr MINUS expr
  - expr MUL expr
  - expr DIV expr
  - expr MOD expr
  - expr L expr
  - expr G expr
  - expr GE expr
  - expr LE expr
  - expr NOTEQ expr
  - expr EQEQ expr
  - expr AND expr
  - expr OR expr
  - VARIABLE call_list
  - '(' expr ')'

- **func_stmt_list :**
  - RETURN expr ';'
  - stmt func_stmt_list

- **func_var_list :**
  - type VARIABLE
  - type VARIABLE ',' func_var_list

- **func_list :**
  - '(' func_var_list ')'
  - '(' ')'
- **call_var_list :**
  - expr
  - call_var_list ',' expr
- **call_list :**
  - '(' call_var_list ')'
  - '(' ')'

6 List of Quadruples

Quadruple	Description