SER 502
Compiler and Virtual Machine for a Programming Language

Team 1
Design Document

“ESCN”

Team members:
Namratha Olety Venkatesh(noletyve@asu.edu)
Rachana Kashyap(rnkashya@asu.edu)
Rhythm Sharma(rsharm58@asu.edu)
Shreyas Hosahalli Govindaraja(shosahal@asu.edu)

Introduction:

This project intends to design, implement, and demonstrate a simple yet efficient language called the “ESCN”. Initial step for building escn is- defining the grammar and performing lexical analysis on it. We have used Antlr4 as parser.

ESCN is a high-level language which can be compiled, interpreted and executed. In addition to above features, ESCN generates intermediate file code as well. Reading of intermediate code file is line-by-line and, executed in JAVA runtime environment.

ESCN has simple, easy to learn syntaxes. It supports all basic arithmetic and relational operators. Data types supported by the language are *integer* and *boolean*. Conditional operators and looping constructs are also supported by the language.

Operators

Arithmetic operators		
Operators	**High Level Language Symbol**	**Intermediate code Symbol**
Addition	+	ADD
Subtraction	-	SUB
Multiplication	*	MUL
Division	/	DIV

Logical operators		
Operators	**High Level Language Symbol**	**Intermediate code Symbol**
Greater than	>	GT
Greater than equal to	>=	GTE
Less than	<	LT
Less than equal to	<=	LTE
Equal to	==	EQ
Not Equal to	!=	NEQ

Conditional Operators		
Operators	**High Level Language Symbol**	**Intermediate code Symbol**
If	if	IF
else	else	ELSE
else If	else if	ELIF

Iterators		
Operators	**High Level Language Symbol**	**Intermediate code Symbol**
while	while	ALA(As long as)

Data Types		
Operators	**High Level Language Symbol**	**Intermediate code Symbol**
integer	int	NUM
boolean	bool	BOOL

Miscellaneous		
Operators	**High Level Language Symbol**	**Intermediate code Symbol**
Assignment	=	SET
Print	print	OUT

Grammar:

```
grammar escn ;

program : 'start' block 'stop' ;

block : declaration control ;

declaration : declarationStmt+ ;

declarationStmt : dataType identifier ';' ;

control : controlStmt+ ;

controlStmt : assignStmt | ifStmt | whileStmt | printStmt ;

assignStmt : identifier '=' (identifier | expr | bool) ';';

ifStmt : ifSection elseSection? 'endif' ;

ifSection : 'if' logicalStmt 'then' block ;

elseSection : 'else' block ;

whileStmt : 'while' logicalStmt 'do' block 'endwhile' ;

block : assignStmt | printStmt ;

logicalStmt : logicalStmt '<=' logicalStmt
 | logicalStmt '>=' logicalStmt
 | logicalStmt '<' logicalStmt
 | logicalStmt '>' logicalStmt
 | logicalStmt '==' logicalStmt
 | logicalStmt '!=' logicalStmt
 | identifier
 | bool
 ;

printStmt : 'print' '(' (expr | logicalStmt) ')' ';' ;

expr : term '+' expr | term '-' expr | term ;

term : factor '*' term | factor '/' term | factor ;

factor : identifier | num ;

identifier : Alphabet identifier* ;
```

Alphabet : [a-zA-Z] ;

num : NUMBER ;

NUMBER : Digit ;

fragment Digit : [1-9][0-9]* | [0] ;

dataType : 'int' | 'boolean' ;

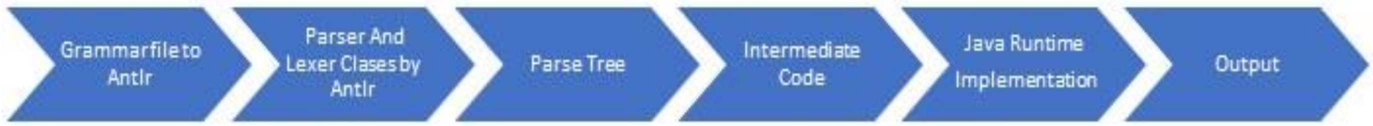
bool : 'true' | 'false' ;

WhiteSpace : [\r\n\t\u000C]+ -> skip ;

LineComment : '//' ~[\n\r]* -> skip ;

BlockComment : '/*' .*? '*' -> skip ;

Compiling and Running of ESCN



The grammar is given as input to ANTLR and the tool will perform lexical and semantic analysis on it. The top-down approach has been followed for the analysis. Lexer, Parser, Listener classes along with tokens will get generated by the tool. The ‘Lexer’ file contains the definition for the generated scanner and the ‘parser’ file contains the definition for the generated parser. Along with the semantic analysis, we will be having the Parse Tree generated. To generate the intermediate code ‘.icf’ file, we will walk through the tree. The baseListener class generated has entry and exit methods for all our parse rules specified in ‘escn’ grammar. Walking through the Tree nodes with the help of an inherited class, from BaseListener class, will be helpful every time a rule is encountered. The intermediate code file will be the input to the java runtime. And, the java interpreter will provide us the output.

The runtime for ESCN language is in JAVA. All the grammar supported keywords and their corresponding meaning are maintained in a symbol table in the form of key - value pair. This is done by using HashMap. Stack is used to maintain the decision-based and loop-based constructs. An ArrayList holds the tokens provided as input to the compiler.

Intermediate Code:

The form we are using to represent intermediate code is Prefix Notation.
For instance, the tree for:

a * (6 + k) can be written as *a+6k.

Using prefix operators makes it much easier to translate to a format that is suitable for direct execution. This type of format can be used directly for tree processing. Infix notations can be ambiguous if we do not know the precedence and associativity. Prefix notations do not need parenthesis and are completely unambiguous.

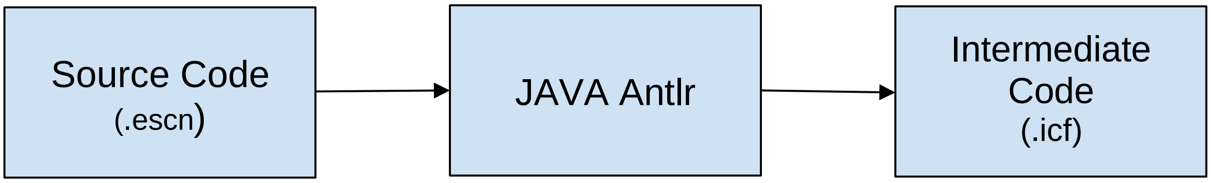


Fig: Representation how JAVA Antlr converts *.escn* file to *.icf* file

Sample Codes:

1. **Program to demonstrate Arithmetic Operation**

ESCN File	Intermediate Code File

start
int a;
int b;
int c;
a = 5;
b = 10;
c = a + b;
print (c);
stop
	NUM a
	NUM b
	NUM c
	SET a,5
	SET b,10
	NUM temp0
	SET temp0,a
	ADD temp0,b
	SET c,temp0
	OUT c

OUTPUT:
15

2. **Program to demonstrate Complex Arithmetic Operation**

ESCN File	Intermediate Code File

start
int a;
int b;
int c;
int d;
int e;
a=6;
b=3;
c=7;
d=2;
e=a*b+c/d-b;
print(e);
stop
	NUM a
	NUM b
	NUM c
	NUM d
	NUM e
	SET a,6
	SET b,3
	SET c,7
	SET d,2
	NUM temp0
	SET temp0,a
	MUL temp0,b
	NUM temp1
	SET temp1,c
	DIV temp1,d
	NUM temp2
	SET temp2,temp1
	SUB temp2,b
	NUM temp3
	SET temp3,temp0
	ADD temp3,temp2
	SET e,temp3
	OUT e

OUTPUT:
18

3. Program to demonstrate simple if statement

ESCN File	Intermediate Code File

start
int a;
int b;
a=6;
b=3;
if a>b
then
print(a);
endif
stop
	NUM a
	NUM b
	SET a,6
	SET b,3
	NUM ifTemp0
	SET ifTemp0,a
	GT ifTemp0,b
	IF ifTemp0
	OUT a
	ENDIF

OUTPUT:

```
6
```

4. Program to demonstrate if else statement

ESCN File	Intermediate Code File

start
int a;
int b;
a=6;
b=3;
if a<b
then
print(a);
else
print(b);
endif
stop
	NUM a
	NUM b
	SET a,6
	SET b,3
	NUM ifTemp0
	SET ifTemp0,a
	LT ifTemp0,b
	IF ifTemp0
	OUT a
	ELSE
	OUT b
	ENDIF

OUTPUT:

```
3
```

5. Program to demonstrate logical expressions in if else statement

ESCN File	Intermediate Code File

start
int a;
int b;
int c;
a=6;
b=3;
if a>b==true
then
c = a+b*a;
print(c);
else
print(a);
endif
stop
	NUM a
	NUM b
	NUM c
	SET a,6
	SET b,3
	NUM ifTemp0
	SET ifTemp0,a
	GT ifTemp0,b
	BOOL ifTemp1
	SET ifTemp1,true
	EQ ifTemp1,ifTemp0
	IF ifTemp1
	NUM temp3
	SET temp3,b
	MUL temp3,a
	NUM temp4
	SET temp4,a
	ADD temp4,temp3
	SET c,temp4
	OUT c
	ELSE
	OUT a
	ENDIF

OUTPUT:
24

6. Program to demonstrate while loop

ESCN File	Intermediate Code File

start
int a;
int b;
a=6;
b=3;
while a>b
do
a = a-1;
print(a);
endwhile
stop
	NUM a
	NUM b
	SET a,6
	SET b,3
	NUM whileTemp0
	SET whileTemp0,a
	GT whileTemp0,b
	ALA whileTemp0
	NUM temp2
	SET temp2,a
	SUB temp2,1
	SET a,temp2
	OUT a
	NUM whileTemp3
	SET whileTemp3,a
	GT whileTemp3,b
	GOBACK

OUTPUT:

```
6
5
4
```