SIL (SIMPLE IMPERATIVE LANGUAGE)

TEAM 29:

ABHIJITH KRISHNAN RADHAKRISHNA KURUP	1211577910
AKSHAY JAIN	1211223907
KAPIL GUPTA	1211259046
UPENDRA SINGH	1210882397

OBJECTIVES:

- Imperative Programming Paradigm
- Loosely typed Language
- Simple Syntax
- Von Neumann architecture
- Simple runtime
- Able to handle complex expressions
- Support for loops (while), condition statement(if-else)

OVERVIEW

- Language SIL is a high level, user friendly, imperative language.
- It has a simple syntax.
- Supports operators, conditional statements, data types, iterations.
- The compiler of SIL is very intelligent and does maximum processing, syntax checking.
- The runtime is very simple, fast and efficient.

INSPIRATION

01

Matlab

02

ADA

03

Python

TOOLS USED

Tools used to build the compiler and runtime interpreter are:

- **Lexical Analysis and Parsing:** ANTLR 4.7
- **Intermediate code generation:** ANTLR 4.7 and Java 8
- **Runtime environment:** Java 8

FEATURES

Our language contains the following features:

- Variables
  - The compiler can easily identify the variables and it is loosely typed.

- Data types: The following primitive data types are supported.
  - Integer
  - Boolean

- Operators: Following operators are supported in the order of the priority
  - Assignment: =
  - Arithmetic: *, /, +, -
  - Conditional: ==, ~=, >, <, >=, <=

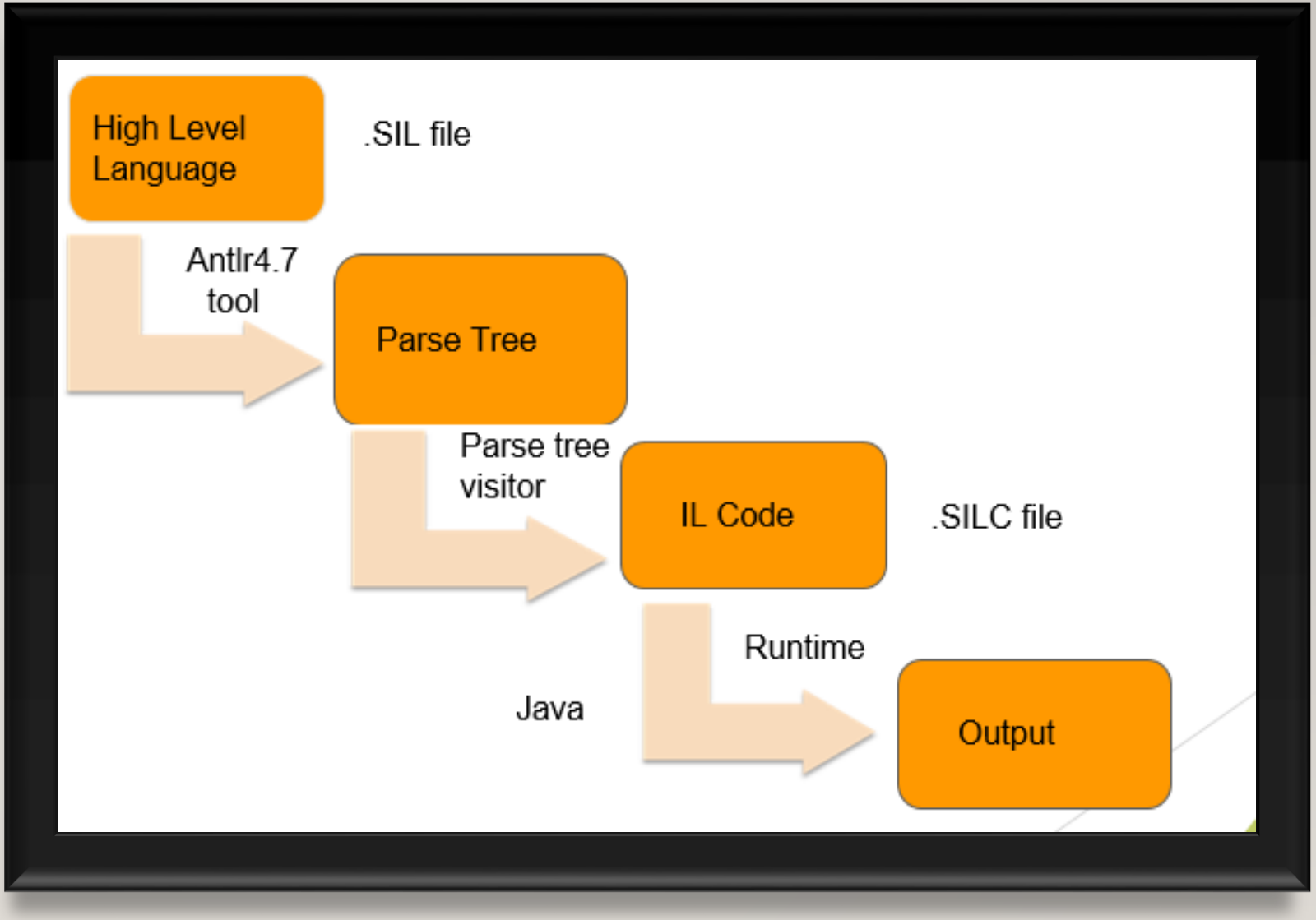
FEATURES (CONTINUED…)

- Complex expressions
  - Handle complex expressions such as : a = a * b + a / b - c

- Conditional operations
  - Implements conditional statement *if* , *nested if* , *else*
  - Handles dangling else problem
  - Else Statement is not mandatory
  - Condition checks in *IF* can be expressions, variables or boolean literals like *true* and *false*

- Iterative operations
  - Implements a loop similar to *while* loop in Java.

WORKING PROCEDURE:

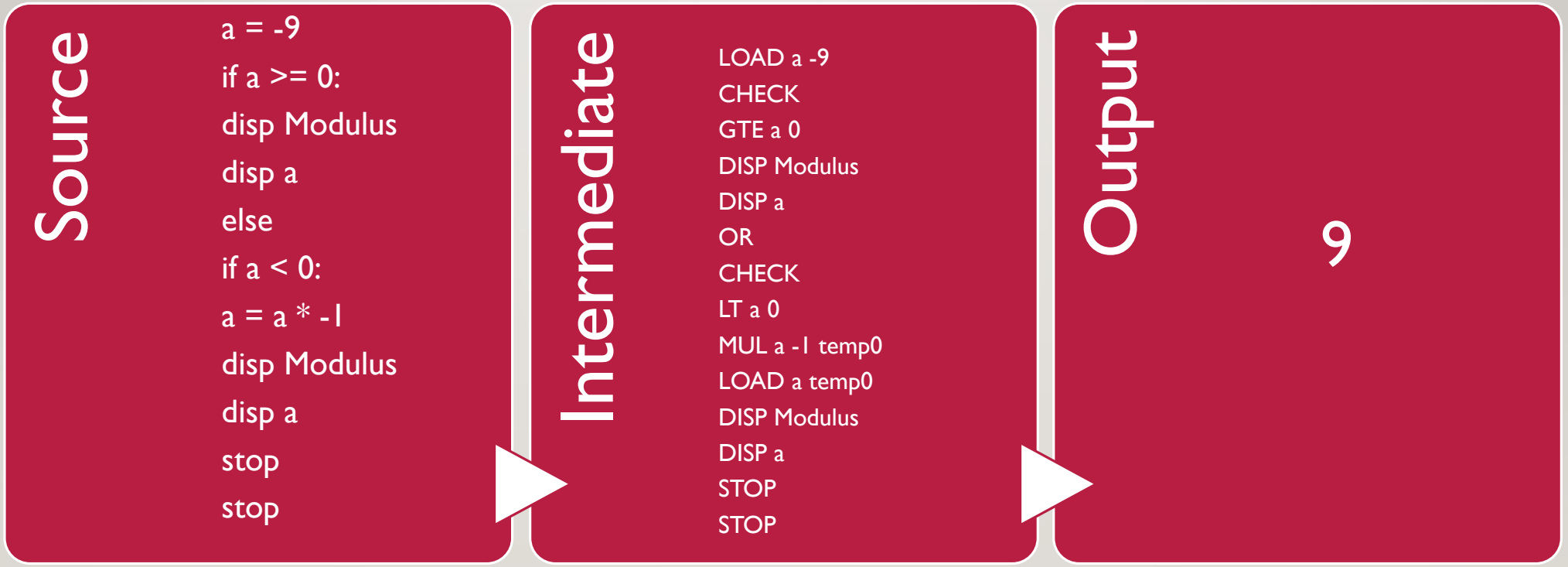


GRAMMAR

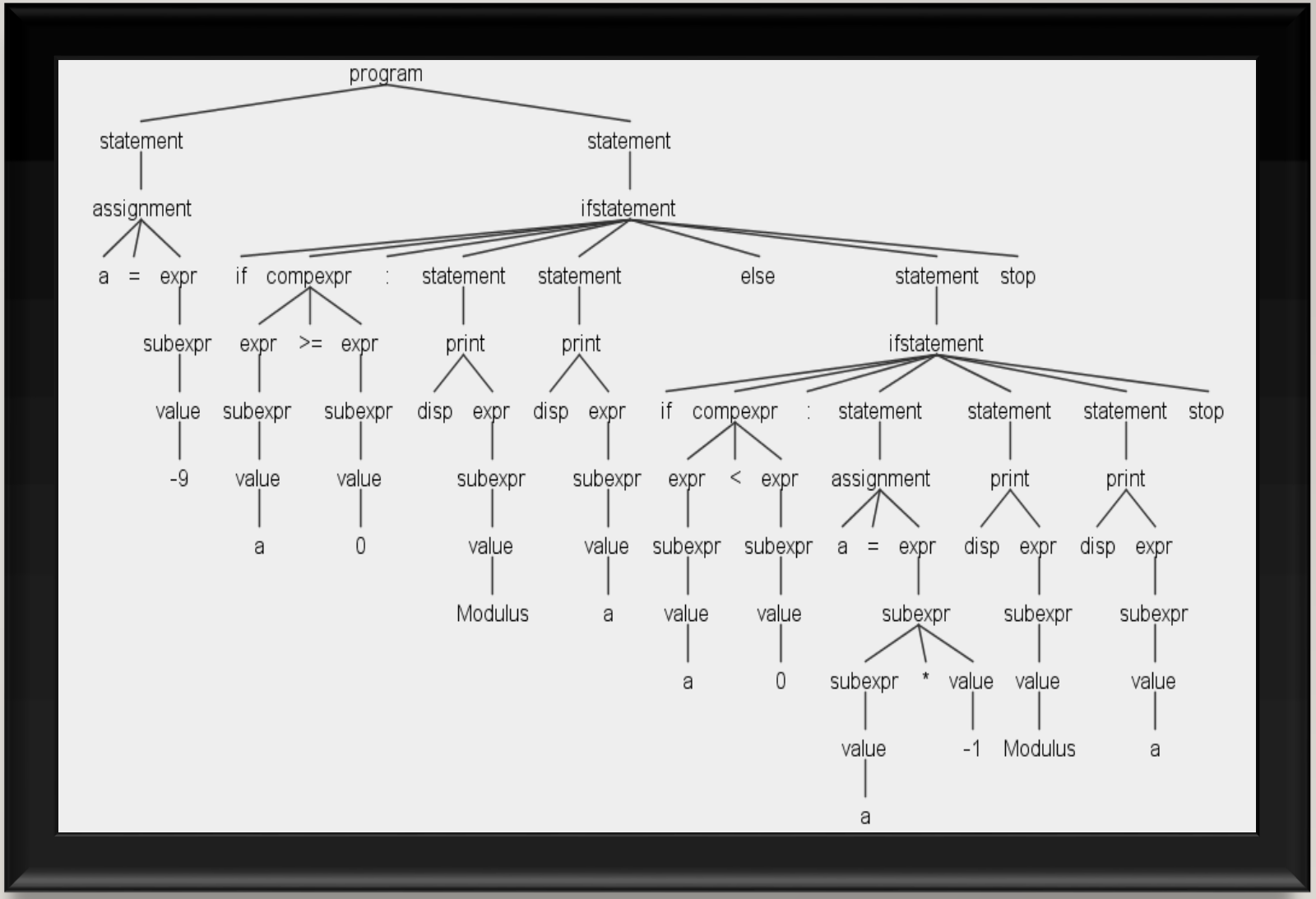
- Grammar is written in Extended Backus-Naur form (EBNF).
- SIL follows the basic grammar shown below

Program → Statement
Statement → IF Stmt | WHILE Stmt | Assignment Stmt | Print Stmt
If Stmt → IF Condition: Statement+ (ELSE Statement+)? STOP
WHILE Statement → WHILE Condition: Statement+ END
Assignment Stmt → Variable = Expression
Print Stmt → DISP Expression
Condition → Expression
Expression → Number

TRANSITION OF CODE



PARSE TREE


program
statement
statement
assignment
ifstatement
a = expr
if compexpr : statement statement else statement stop
subexpr
expr >= expr
print
print
ifstatement
value
subexpr
subexpr
disp expr
disp expr
if compexpr : statement statement statement stop
-9
value
value
subexpr
subexpr
expr < expr
assignment
print
print
a
0
value
value
subexpr
subexpr
a = expr
disp expr
disp expr
Modulus
a
value
value
subexpr
subexpr
subexpr
a
0
subexpr * value
value
value
value
-1
Modulus
a
a

SIL COMPILER

- The Input .sil file is fed into the compiler
- ANTLR performs lexical analysis and parsing using the grammar file which then generates the parse tree on the basis of the defined grammar.
- A visitor class is used to traverse through the nodes of the tree.
- For each syntactic rules, created by ANTLR in the SILBaseVisitor class.
- We have overridden the SILBaseVisitor class methods in MyVisitor class to generate the intermediate code.
- Once all the nodes are traversed, the intermediate file .silc will be generated.

SIL RUNTIME

- Runtime is written in Java
- .silc intermediate file is given as input which generates the final output.
- The intermediate language is read line by line.
- Each line of intermediate code is divided into lexical tokens.
- The first token in a line determines the actions to be performed.
  Based on it, appropriate function call is made.
- Symbol table is used for value storage.

CHALLENGES

- Learning Antlr4 tool was one of the major challenge we faced.
- Another major challenge was to build the entire runtime from scratch.
- Implementation of nested condition and condition within iteration

REFERENCES

- http://www.antlr.org/
- http://stackoverflow.com/questions/tagged/antlr
- http://www.oursland.net/tutorials/antlr/AntlrEclipse.html
- http://www.antlr2.org/doc/sor.html


THANK
YOU
return