CS335 Project
Milestone 4
Group ID - 31

Kartavya Damor(200492) Soruabh Mina(200996) Kevalkumar Solanki(200991)

Instructions

The following tools were used in the project :-

- Flex : flex 2.6.4
- Bison : bison (GNU Bison) 3.8.2
- Dot : dot - graphviz version 2.43.0 (0)
- g++: g++ (Ubuntu 11.3.0-1ubuntu1 22.04) 11.3.0

The MakeFile contains the compiling codes for the flex bison and c++ program,

```
compile:
  bison -dtv parser.y
  flex scanner.l
  g++ -o milestone4 parser.tab.c lex.yy.c AST.cpp symbol_table.cpp typecheck.cpp 3ac.cpp
    codegen.cpp
```

Milestone 4:
For Generating Symbol Tables (.csv), 3AC (.txt) and x86_64 code (.s):-

- make

To run a single testcases from the test folder:-

- ./milestone4 −−input ../tests/<TestFileName>.java

The output CSVs, TXTs and assembly code will be generated in a folder (with the same name as the test file).
For your convenience, we have included a bash script, which runs on all our test cases and provides the output in the respective test name folders.
For executing the bash script,

- ./runall.sh

After running the latter command the code is ready to be tested.

Implementation level details
Modifier contains limited keywords (public, private, static)
Since there are C-type array declarations, the function's return value cannot be java style array. For example, "public int[8] fun()" will not work.
To reduce conflicts, we have simplified the grammar to support only basic for loop.

3AC Functions

1. param: Pushes the parameters into the stack from right to left.
2. stackpointer– x : Removes x bytes from the stack
3. popreturn: returns the function's return value from its corresponding location (offset) stack.
4. popparam: Emits parameters from the stack in the correct order. (First emitted value will be the base pointer)
5. beginfunc x : Allocates x bytes (for local and temporary variables) in the stack for the called function.
6. call func x : Calls the function (whose x parameters were previously pushed onto the stack.)
7. call_alloc x : Allocates x memory for the object and return the object.

Functions used to generate x86_64 .s code

1. Register Allocation (getReg): Allocates registers according to the algorithm, discussed in the class.
2. Memory Location (getmemlocation): Returns the location of the given input
3. Different Function for all operators:-
    - Addition (Addop)
    - Subtraction (Subop)
    - Division (Divop)
    - Multiplication (Multop)
    - Modulo (Modop)
    - Bitwise (Bitwiseop)
    - Shift (Shiftop)
    - Relational Operators
    - Logical Operators
4. Call Function (callfunc): Writes the assembly code for a function call
5. Assigning the leaders (findleaders): Assigns the leader lines as per the algorithm described in the book.
6. Updating the register values (updateregval): Updates the value stored in the register as per the given arguements.

Contribution

S. No.	Name	Roll No	Email	Contribution
1	Kartavya Damor	200492	kartavyad20@iitk.ac.in	33.33%
2	Solanki Kevalkumar	200991	solankikb20@iitk.ac.in	33.33%
3	Sourabh Mina	200996	sourabhm20@iitk.ac.in	33.34%

References

1. Java Language Specifications: https://docs.oracle.com/javase/specs/jls/se17/html/jls-19.html
2. https://github.com/mohitmo/CS335-Project (got insights for AST and dot generation procedures)
3. A.Aho, R.Sethi, and J.Ullman. Compilers: Principles, Techniques, and Tools, 1st edition.
4. Testcases code generated from GFG, javaviz etc websites.

:)