Final Report

Team Members:

Hajar El Boutahiri

Youssef Yousfi

Hiba Aqqaoui

Final Report

Professor

Violetta Cavalli-Sforza

CSC 3315 02

Final Report

Documentation:

The teammates are:

- Youssef Yousfi
- Hajar El Boutahiri
- Hiba Aqqaoui

Instructions to run the code:

In order to run the code: you need to have first all files in one folder:

In other words:

- Start by unzipping the folder submitted on Canvas:
- Copy the address of the folder “Deliverable6” from your computer to run all files at the same time.
- Copy the wanted code into “file.txt”
- Open cmd to run the code
- Write cd + the address folder
- Copy the Command Python: Generator.py

Summary:

Note: Our language is case-sensitive but accept spaces.

1. Interpreter:

In this part, we implemented a virtual machine (VM) whose language (VML). The used programming language was C; we were able to: Load instructions from a file, create the

appropriate label table and symbol table, and Execute Instructions.

2. Language Design

In this part, we have designed our language; we worked on the lexical description (reserved words, punctuation, operators, built-in-functions, user-defined identifiers, Positive Decimal Integer Literals, Boolean literals, Strings constants, IDs…) and the syntactic description using EBNF; and we made sure that it is suitable for an LL1 parser: unambiguous, not left recursive, and that each rule's RHSs are pairwise disjoint.

3. Lexer

In this part of the project, from a given program, which is satisfying our grammar rules, we identified the lexemes, their line code, and converted them into stream of tokens, we also have identified unacceptable lexemes and have displayed errors, and we built the symbol and label table. We passed to the parser a stream of lexemes.

4. Parser & Static Semantics

In this part, using the stream of tokens and line codes array taken from the lexer phase, we handled all possible errors, we have created an acceptor, built the concrete syntax tree (CST) and abstract syntax tree (AST), code to ensure that all referenced identifiers appear in the code with their definitions before they are referenced, an indicator of scope in the Symbol Table is existing, possibility to distinguish global variables from local variables/parameters, actual used parameters in function calls match the formal parameters of function definitions in number…

5. Generator

In this part, we worked on generating the corresponding VML for expressions, assignments, control structures and input/output instructions, designing our next steps (ways

to implement the stack and related controls, size of memory, ways to modify the instruction sets…) and finally implementing a restricted design as an extra credit.

Overall:

Thanks to this project we have learned a lot; when it comes to developing personal, and professional skills. Some of them were time management, effective communication, group work, organizational, problem solving, active listening, conflict resolution skills … Thanks to it again, we were productive, have learned a lot, and have enjoyed the class more and more.