BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS

Group Number

13

Compiler Construction (CS F363)
II Semester 2021-22
Compiler Project (Stage-2 Submission)
Coding Details
(April 16, 2022)

Instruction: Write the details precisely and neatly. Places where you do not have anything to mention, please write NA for Not Applicable.

1. IDs and Names of team members

ID 2018B5A70423P	Name Ashwin Kiran Godbole
ID 2018B2A70362P	Name Samarth Krishna Murthy

2. Mention the names of the Submitted files (Include Stage-1 and Stage-2 both)

1 grammar.txt
2 coding details stage 2.pdf
3 lexerDef.h
4 lexer.h
5 lexer.c
6 parserDef.h
7 parser.h
8 parser.c
9 makefile
10 semantic.h
11 semantic.c
12 semanticDef.h
13 symbolTable.c
14 driver.c
15 testcase1.txt
16 testcase5.txt
17 testcase6.txt
18 testcase7.txt
19 testcase8.txt
20 grp13_td.pdf
21 grp13_ff.pdf
22 SemanticRules_Group13.pdf

3. Total number of submitted files: 22 (All files should be in **ONE** folder named exactly as Group number)
4. Have you mentioned names and IDs of all team members at the top of each file (and commented well)? (Yes/ no) Yes [Note: Files without names will not be evaluated]
5. Have you compressed the folder as specified in the submission guidelines? (yes/no): yes

6. **Status of Code development**: Mention 'Yes' if you have developed the code for the given module, else mention 'No'.
   a. Lexer (Yes/No): Yes
   b. Parser (Yes/No): Yes
   c. Abstract Syntax tree (Yes/No): Yes
   d. Symbol Table (Yes/ No): Yes
   e. Type checking Module (Yes/No): No
   f. Semantic Analysis Module (Yes/ no): No (reached LEVEL ____ as per the details uploaded)
   g. Code Generator (Yes/No): No

7. **Execution Status**:
   a. Code generator produces code.asm (Yes/ No): No
   b. code.asm produces correct output using NASM for testcases (C#.txt, #:1-11): No
   c. Semantic Analyzer produces semantic errors appropriately (Yes/No): No
   d. Static Type Checker reports type mismatch errors appropriately (Yes/ No): No
   e. Dynamic type checking works for variant records with tagged union and reports errors on executing code.asm (yes/no): No
   f. Symbol Table is constructed (yes/no) yes and printed appropriately (Yes /No): Yes
   g. AST is constructed (yes/ no) yes and printed (yes/no) Yes

h. Name the test cases out of 17 as uploaded on the course website for which you get the segmentation fault (p#.txt ; # 1-4, s$.txt; $ 1-5, and c@.txt ; @:1-8):____

8. **Data Structures** (Describe in maximum 2 lines and avoid giving C definition of it)
   a. AST node structure : AST node contains 3 pointers (parent, leftchild, rightSibling), the name of the node, the depth of the node and a pointer to a structure which contains other info about the node.

   b. Symbol Table structure: Symbol table contains name of scope, pointer to symbol table entries, and offset of the table.

   c. Record type expression structure: A linked list that contains the type and a pointer to the next node which contains the next type.

   d. Data structure for global variables: N/A
   e. Variant record type expression structure: N/A
   f. Input parameters type structure: N/A
   g. Output parameters type structure: N/A
   h. Structure for maintaining the three address code(if created) :________________________________
   i. Any other interesting data structures used :____

9. **Semantic Checks:** Mention your scheme NEATLY for testing the following major checks (in not more than 5-10 words)[Hint: You can use simple phrases such as 'symbol table entry empty', 'symbol table entry already found populated', 'traversal of linked list of parameters and respective types' etc.]
   a. Variable not Declared :____

   b. Multiple declarations: ____

   c. Number and type of input and output parameters:____

   d. assignment of value to the output parameter in a function ____________________________________

   e. function call semantics:____

   f. static type checking :____

   g. return semantics:____

   h. Recursion :____

   i. module overloading:____

   j. if-then-else semantics :____

   k. handling offsets for local variables (starting with 0, integer size =2, real size =4 for symbol table purpose):____
   ____

   l. handling offsets for formal parameters:____

   m. handling global variable declaration over local variables and input-output parameters:____________
   ____

   n. Record semantics and static type checking: ____

   ____

   o. Variant record semantics and dynamic type checking: __________________________________

   ____

   p. Scope of variables and their visibility :____

q. handling nesting depth of variables in Boolean expression in while loop for assignment of an expression to one of the guard variables:____
____

10. Compiler passes description (Mention the details of information collected/populated/worked upon at each traversal of the whole AST):

a. Pass 1: ____
b. Pass 2: ____
c. Pass 3: ____
d. Pass 4: ____

11. Code Generation:

a. NASM version as specified earlier used (Yes/no): No
b. Used 32-bit or 64-bit representation:____
c. For your implementation: 1 memory word = ____________________(in bytes)
d. Mention the names of major registers used by your code generator:

- For base address of an activation record: __________________
- for stack pointer:_______________________________________
- others (specify):_______________________________________

e. Mention the physical sizes of the integer and real data as used in your code generation module
size(integer): _______________________(in words/ locations), _______________(in bytes)
size(real): ___________________________(in words/ locations), _______________(in bytes)

f. How did you implement functions calls?(write 3-5 lines describing your model of implementation)
____
____
____

g. Specify the following:

- Caller's responsibilities:____
- Callee's responsibilities:____

h. How did you maintain return addresses? (write 3-5 lines): ____________________________________
____
____
____

i. How have you maintained parameter passing? How were the statically computed offsets of the parameters used by the callee? ____

j. What have you included in the activation record size computation? (local variables, parameters, both):
____

k. Choice of registers (your manually selected heuristic only) ____________________________________
____

l. Which primitive data types have you handled in your code generation module?(Integer and real):
____

m. Where are you placing the temporaries in the activation record of a function? __________________
____

n. Write your method of code generation for dynamic type checking for tagged union data type. _____
____

12. **Compilation Details**:

a. Makefile works (yes/No): yes

b. Code Compiles (Yes/ No): yes

c. Mention the .c files that do not compile: N/A

d. Any specific function that does not compile: N/A

e. Ensured the compatibility of your code with the specified versions [GCC, UBUNTU, NASM] (yes/no) GCC&UBUNTU: yes

13. Execution time for compiling the test cases [type checking (p1-p4.txt), semantic analyses including symbol table creation (s1-s5.txt), and code generation (c1-c8.txt)] :

i. p1.txt (in ticks) ______________________ and (in seconds) _____________________

ii. p2.txt (in ticks) ______________________ and (in seconds) _____________________

iii. p3.txt (in ticks) ______________________ and (in seconds) _____________________

iv. p4.txt (in ticks) ______________________ and (in seconds) _____________________

v. s1.txt (in ticks) ______________________ and (in seconds) _____________________

vi. s2.txt (in ticks) ______________________ and (in seconds) _____________________

vii. s3.txt (in ticks) ______________________ and (in seconds) _____________________

viii. s4.txt (in ticks) ______________________ and (in seconds) _____________________

ix. s5.txt (in ticks) ______________________ and (in seconds) _____________________

x. c1.txt (in ticks) ______________________ and (in seconds) _____________________

xi. c2.txt (in ticks) ______________________ and (in seconds) _____________________

xii. c3.txt (in ticks) ______________________ and (in seconds) _____________________

xiii. c4.txt (in ticks) ______________________ and (in seconds) _____________________

xiv. c5.txt (in ticks) ______________________ and (in seconds) _____________________

xv. c6.txt (in ticks) ______________________ and (in seconds) _____________________

xvi. c7.txt (in ticks) ______________________ and (in seconds) _____________________

xvii. c8.txt (in ticks) ______________________ and (in seconds) _____________________

14. **Driver Details**: Does it take care of the **ELEVEN** options specified earlier?(yes/no): yes

15. Specify the language features your compiler is not able to handle (in maximum one line)
____

16. Are you availing the lifeline (Yes/No): No

17. Write exact command you expect to be used for executing the code.asm using NASM simulator [We will use these directly while evaluating your NASM created code]
____
____

18. **Strength of your code**(Strike off where not applicable): (a) correctness ~~(b) completeness~~ ~~(c) robustness (d) Well documented~~ (e) readable (f) strong data structure (f) Good programming style (indentation, avoidance of goto stmts etc) (g) modular (h) space and time efficient
19. Any other point you wish to mention: ____

    ____

    ____

20. Declaration: We, Ashwin Kiran Godbole and Samarth Krishna Murthy(your names) declare that we have put our genuine efforts in creating the compiler project code and have submitted the code developed only by our group. We have not copied any piece of code from any source. If our code is found plagiarized in any form or degree, we understand that a disciplinary action as per the institute rules will be taken against us and we will accept the penalty as decided by the department of Computer Science and Information Systems, BITS, Pilani. [Write your ID and names below]

    ID 2018B5A70423P Name: Ashwin Kiran Godbole
    ID 2018B2A70362P Name: Samarth Krishna Murthy

    Date: 16th April, 2022

    Should not exceed 6 pages.