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

| Group Number |
| --- |
| 21 |

**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: 2019A7PS0087P Name: Raghava Kasyap Kristipati

ID: 2019A7PS0068P Name: Yadagiri Shiva Sai Sashank

ID: 2019A7PS0030P Name: K. V. S. Preetam

ID: 2019A7PS0028P Name: Shanmukh Chandra Yama

ID: 2019A7PS0083P Name: Uday Dheeraj Nulu

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

| | | | |
| --- | --- | --- | --- |
| 1 ast.c | 7 lexer.h | 13 symbol_table.h | 19 t5.txt |
| 2 ast.h | 8 lexerDef.h | 14 symbol_tableDef.h | 20 t6.txt |
| 3 astDef.h | 9 parser.c | 15 t1.txt | 21 type_checker.c |
| 4 driver.c | 10 parser.h | 16 t2.txt | 22 type_checker.h |
| 5 grammar.txt | 11 parserDef.h | 17 t3.txt | 23 type_checkerDef.h |
| 6 lexer.c | 12 symbol_table.c | 18 t4.txt | 24 p1.txt |

| | |
| --- | --- |
| 25 s1.txt | 31 p3.txt |
| 26 s2.txt | 32 p4.txt |
| 27 s3.txt | |
| 28 s4.txt | |
| 29 s5.txt | |
| 30 p2.txt | |

3 Total number of submitted files: 29 (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'.

6.a Lexer (Yes/No): _____YES_____

6.b Parser (Yes/No):_____YES_____

6.c Abstract Syntax tree (Yes/No):_____YES_____

6.d Symbol Table (Yes/ No):_____YES_____

6.e Type checking Module (Yes/No):_____NO_____

6.f Semantic Analysis Module (Yes/ no):_____NO_____(reached LEVEL ____ as per the details uploaded)

6.g Code Generator (Yes/No):_____NO_____

7 **Execution Status**:

7.a Code generator produces code.asm (Yes/ No):_____NA_____

7.b code.asm produces correct output using NASM for testcases (C#.txt, #:1-11): _____NA_____

7.c Semantic Analyzer produces semantic errors appropriately (Yes/No):_____NA_____

7.d Static Type Checker reports type mismatch errors appropriately (Yes/ No):_____NA_____

7.e Dynamic type checking works for variant records with tagged union and reports errors on executing code.asm (yes/no):_____NA_____

7.f Symbol Table is constructed (yes/no)____YES_____and printed appropriately (Yes /No):_____YES_____

7.g AST is constructed (yes/ no) _____YES_____and printed (yes/no) _____YES_____

7.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): we ran test cases t2-t5 for symbol table and ast as we haven't finished type-checker. It also did not give segmentation error in the files which have no variable declaration errors which are s1, s5, p1. Other files gave segmentation error while creating the symbol table as we have not handled errors.

8 **Data Structures** (Describe in maximum 2 lines and avoid giving C definition of it)

8.a AST node structure – It is a structure which has fields and a union which stores different type of information for different nodes.

8.b Symbol Table structure: Complex Structure which contains a hash table and different type of nodes for different types like variables, records etc.

8.c Record type expression structure: Linked List of basic or constructed types

8.d Data structure for global variables: Used a flag called isGlobal and have globalSymbolTable.

8.e Input parameters type structure: Present in function identifiers (Linked List)

8.f Output parameters type structure: Present in function Identifiers (Linked List)

8.g Structure for maintaining the three address code(if created) : Not created

8.h 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.]

9.a Variable not Declared :_____

9.b Multiple declarations: _____

9.c Number and type of input and output parameters:_____

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

9.e function call semantics:_____

9.f static type checking :_____

9.g return semantics:_____

9.h Recursion :_____

9.i module overloading:_____

9.j if-then-else semantics :_____

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

9.l handling offsets for formal parameters:_____

9.m handling global variable declaration over local variables and input-output parameters:_____

_____

9.n Record semantics and static type checking: _____

_____

9.o Variant record semantics and dynamic type checking: _____

_____

9.p Scope of variables and their visibility :_____

9.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):**

10.a Pass 1: Global variables are collected and added to symbol table

10.b Pass 2: All remaining things are added to respective symbol tables.

10.c Pass 3: Tried to implement type checker.

10.d Pass 4: _____

**11 Code Generation:**

a.a NASM version as specified earlier used (Yes/no):_____

a.b Used 32-bit or 64-bit representation:_____

a.c For your implementation: 1 memory word = _____(in bytes)

a.d Mention the names of major registers used by your code generator:

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

a.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)

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

_____

_____

_____

a.g Specify the following:

- Caller's responsibilities:_____
- Callee's responsibilities:_____

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

_____

_____

_____

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

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

_____

a.k Choice of registers (your manually selected heuristic only) _____

_____

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

_____

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

_____

_____

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

_____

12 **Compilation Details**:

12.a Makefile works (yes/No): No

12.b Code Compiles (Yes/ No): Yes

12.c Mention the .c files that do not compile: type_checker.c

12.d Any specific function that does not compile:_____NA_____

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

a.i p1.txt (in ticks) _____ and (in seconds) _____

a.ii p2.txt (in ticks) _____ and (in seconds) _____

a.iii p3.txt (in ticks) _____ and (in seconds) _____

a.iv p4.txt (in ticks) _____ and (in seconds) _____

a.v s1.txt (in ticks) _____ and (in seconds) _____

a.vi s2.txt (in ticks) _____ and (in seconds) _____

a.vii s3.txt (in ticks) _____ and (in seconds) _____

a.viii s4.txt (in ticks) _____ and (in seconds) _____

a.ix s5.txt (in ticks) _____ and (in seconds) _____

a.x c1.txt (in ticks) _____ and (in seconds) _____

a.xi c2.txt (in ticks) _____ and (in seconds) _____

a.xii c3.txt (in ticks) _____ and (in seconds) _____

a.xiii c4.txt (in ticks) _____ and (in seconds) _____

a.xiv c5.txt (in ticks) _____ and (in seconds) _____

a.xv c6.txt (in ticks) _____ and (in seconds) _____

a.xvi c7.txt (in ticks) _____ and (in seconds) _____

a.xvii c8.txt (in ticks) _____ and (in seconds) _____

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

15 Specify the language features your compiler is not able to handle (in maximum one line)
We haven't done semantic analyzer and code generator part and implemented type- checker till a certain extent.

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

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]
NA

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:

Our code is working for all the provided test cases , but only till the implementation of the symbol table. Error recovery was not done in symbol table implementation phase. So for test cases containing errors , symbol table throws segmentation error. For type checking phase we had handled most of the cases but we were not able to debug the code due to poor time management.

20 Declaration: We, Raghava Kasyap Kristipati, K.V.S Preetam, Yadagiri Shiva Sai Sashank, Uday Dheeraj Nulu, Shanmukh Chandra Yama (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 2019A7PS0087P | Name: Raghava Kasyap Kristipati |
| ID 2019A7PS0030P | Name: K.V.S Preetam |
| ID 2019A7PS0028P | Name: Shanmukh Chandra Yama |
| ID 2019A7PS0068P | Name: Yadagiri Shiva Sai Sashank |
| ID 2019A7PS0083P | Name: Uday Dheeraj Nulu |

Date: 17.04.2022

---

Should not exceed 6 pages.