**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS**

**Compiler Construction (CS F363)**

**II Semester 2019-20**

**Compiler Project (Stage-2 Submission)**

**Coding Details**

**(April 20, 2020)**

*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: 2017A7PS0006P Name: PIKLU PAUL

ID: 2017A7PS0007P Name: RAJABABU SAIKIA

ID: 2017A7PS0090P Name: SAURAV VIRMANI

ID: 2017A7PS0111P Name: SIDDHANT KHARBANDA

ID: 2017A7PS0275P Name: SREYAS RAVICHANDRAN

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

1. lexer.c 7. adt.c 13. utils.c 20. ast.c

2. lexerDef.h 8. adt.h 14. utils.h 21. ast.h

3. lexer.h 9. hash\_table.c 15. driver.c 22. codeGen.c

4. parser.c 10. hash\_table.h 16. makefile 23. codeGen.h

5. parserDef.h 11. parseRules.c 17. rules.txt 24. ir.c

6. parser.h 12. parseRules.h 18. nonterminals.txt 25. ir.h

19. keywords.txt 26. parseTable.c 27. parseTable.h 28. terminals.txt

29. performa.docx 30. symbol\_table.c 31. symbol\_table.c 32. semCheck.c

33. semCheck.h 34. stage1.exe

1. Total number of submitted files: **34** (All files should be in **ONE** folder named exactly as Group number)
2. 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]
3. Have you compressed the folder as specified in the submission guidelines? (yes/no) **YES**
4. **Status of Code development**: Mention 'Yes' if you have developed the code for the given module, else mention 'No'.
   1. Lexer (Yes/No): **YES**
   2. Parser (Yes/No):**YES**
   3. Abstract Syntax tree (Yes/No):**YES**
   4. Symbol Table (Yes/ No):**YES**
   5. Type checking Module (Yes/No): **YES**
   6. Semantic Analysis Module (Yes/ no): **YES** (reached LEVEL \_4\_ as per the details uploaded)
   7. Code Generator (Yes/No):**YES**
5. **Execution Status**:
   1. Code generator produces code.asm (Yes/ No): yes
   2. code.asm produces correct output using NASM for testcases (C#.txt, #:1-11): no
   3. Semantic Analyzer produces semantic errors appropriately (Yes/No): yes
   4. Static Type Checker reports type mismatch errors appropriately (Yes/ No): yes
   5. Dynamic type checking works for arrays and reports errors on executing code.asm (yes/no): no
   6. Symbol Table is constructed (yes/no) \_\_yes\_\_\_and printed appropriately (Yes /No): yes
   7. AST is constructed (yes/ no) \_\_yes\_\_\_\_\_and printed (yes/no) \_yes\_\_\_
   8. Name the test cases out of 21 as uploaded on the course website for which you get the segmentation fault (t#.txt ; # 1-10 and c@.txt ; @:1-11):\_ None\_\_
6. **Data Structures** (Describe in maximum 2 lines and avoid giving C definition of it)
   1. AST node structure\_\_\_\_\_as an n-arry tree where each node has data of union type of possible structures.

\_\_\_\_\_\_

* 1. Symbol Table structure:\_\_\_\_as an n-arry tree where each node has a hash table(for variables) and a few meta data

\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. array type expression structure:\_\_\_\_\_implemented as a structure having basetype, elementtype, and low and high pointers. \_\_\_\_\_\_\_\_\_\_\_\_\_
  2. Input parameters type structure:\_\_\_hash table of input variables where each hash item has data corresponding to input variable \_\_\_\_\_\_
  3. Output parameters type structure:\_\_\_\_ hash table of output variables where each hash item has data corresponding to output variable \_\_\_\_\_\_\_\_
  4. Structure for maintaining the three address code(if created) :\_ implemented as a linked list where each node has a three address code. Each address can be of 3 types : Constant, Variable, Label.

\_\_\_\_\_\_\_\_\_

1. **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.]
   1. Variable not Declared :\_\_\_\_variable not found in hierarchy of scopes\_\_\_\_\_\_\_\_\_\_
   2. Multiple declarations: \_\_\_\_already undeclared variable found at Line number, lexeme(hash table).present in the current symbol table(hash table).\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   3. Number and type of input and output parameters: \_traversal of parameters structure with tables for input and output lists\_\_\_\_
   4. assignment of value to the output parameter in a function \_\_\_total output parameters allotted value\_\_\_\_\_
   5. function call semantics:\_\_\_\_\_\_\_\_\_checking params in corresponding func struct which is present as a hash item in hash table of functions \_\_\_\_
   6. static type checking :\_\_\_\_\_\_\_\_\_\_\_matching type at every level of every node in expression in ast\_\_\_\_\_\_
   7. return semantics:\_\_\_\_output variable mask check(to check value assigned), type matching of output vars, \_\_\_\_\_\_\_\_
   8. Recursion :\_\_\_\_\_\_\_checking whether function calls itself within its scope \_\_\_\_\_\_\_\_\_\_\_\_
   9. module overloading:\_\_\_\_\_checking in function hash table (no duplicates) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   10. 'switch' semantics :\_\_\_\_\_check value type, one scope for entire scope, no duplicate cases etc \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   11. 'for' and 'while' loop semantics: \_handling scope and modification to looping variables, compulsory modification required for atleast one loop variable

For : low<high for the values given as low..high

While : short circuiting (done wherever Boolean expressions exist), type evaluation of condition as boolean.\_\_\_\_\_

* 1. handling offsets for nested scopes:\_\_\_\_as a parameter in struct of symbol table tree\_\_\_\_\_\_\_\_\_\_\_\_
  2. handling offsets for formal parameters:\_\_\_\_\_\_offset start for input list and ends at output list\_\_\_\_\_\_\_\_\_\_
  3. handling shadowing due to a local variable declaration over input parameters: \_\_maintaining a different hash table for input variables in the function scope\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  4. array semantics and type checking of array type variables: \_types are defined as an enum where array is a composite type of base type, element type,low and high
  5. Scope of variables and their visibility :\_\_\_\_\_decendents of symbol table node denotes the scope \_\_\_\_\_\_\_
  6. computation of nesting depth:\_\_\_\_\_\_as a parameter in the symbol table tree struct \_\_\_\_\_\_\_\_\_\_\_

1. Code Generation:
   1. NASM version as specified earlier used (Yes/no):\_\_\_\_\_yes\_\_\_\_\_\_\_\_\_\_
   2. Used 32-bit or 64-bit representation:\_\_\_\_\_\_\_\_\_\_\_\_\_64\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   3. For your implementation: 1 memory word = \_\_\_\_\_\_\_\_\_1\_\_\_\_\_\_\_\_\_\_\_(in bytes)
   4. Mention the names of major registers used by your code generator:

* For base address of an activation record: \_\_\_\_\_bp\_\_\_\_\_\_\_\_\_\_\_\_\_
* for stack pointer:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_sp\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* others (specify):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_bx for indirect addressing\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  1. Mention the physical sizes of the integer, real and boolean data as used in your code generation module

size(integer): \_\_\_\_\_\_\_\_\_\_\_2\_\_\_\_\_\_\_\_\_\_\_\_(in words/ locations), \_\_2\_\_\_\_\_\_\_\_\_\_\_\_\_(in bytes)

size(real): \_\_\_\_\_\_\_\_\_\_\_\_\_\_4\_\_\_\_\_\_\_\_\_\_\_\_(in words/ locations), \_\_\_\_\_\_\_4\_\_\_\_\_\_\_\_(in bytes)

size(booelan): \_\_\_\_\_\_1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(in words/ locations), \_\_1\_\_\_\_\_\_\_\_\_\_\_\_\_(in bytes)

* 1. How did you implement functions calls?(write 3-5 lines describing your model of implementation) \_pushin all function parameters on top of stack and then using positive indexes to access those\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Specify the following:
     + Caller's responsibilities:\_\_push space of output and input parameters\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     + Callee's responsibilities:\_\_\_\_\_\_\_fill the output parameters, allocate space for variables\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  2. How did you maintain return addresses? (write 3-5 lines): \_\_\_\_\_we used call so it automatically took care of it\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. How have you maintained parameter passing? How were the statically computed offsets of the parameters used by the callee? \_\_\_\_offsets +8 for return address\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  2. How is a dynamic array parameter receiving its ranges from the caller? \_\_\_\_\_\_\_start of array\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  3. What have you included in the activation record size computation? (local variables, parameters, both): \_\_\_\_\_\_\_\_\_\_\_\_\_local variables\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  4. register allocation (your manually selected heuristic) :\_\_\_\_\_\_\_deallocate only if not used further\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  5. Which primitive data types have you handled in your code generation module?(Integer, real and boolean):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  6. Where are you placing the temporaries in the activation record of a function? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Compilation Details**:
   1. Makefile works (yes/No):\_\_yes\_\_\_\_
   2. Code Compiles (Yes/ No):\_\_\_yes\_\_\_\_\_\_
   3. Mention the .c files that do not compile:\_\_\_\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   4. Any specific function that does not compile:\_\_\_\_\_\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   5. Ensured the compatibility of your code with the specified versions [GCC, UBUNTU, NASM] (yes/no)\_\_\_yes\_\_\_\_
2. Execution time for compiling the test cases [lexical, syntax and semantic analyses including symbol table creation, type checking and code generation] :
   * 1. t1.txt (in ticks) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     2. t2.txt (in ticks) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     3. t3.txt (in ticks) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     4. t4.txt (in ticks) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     5. t5.txt (in ticks) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     6. t6.txt (in ticks) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     7. t7.txt (in ticks) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     8. t8.txt (in ticks) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     9. t9.txt (in ticks) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     10. t10.txt (in ticks) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. **Driver Details**: Does it take care of the **TEN** options specified earlier?(yes/no):\_\_\_\_yes\_\_
4. Specify the language features your compiler is not able to handle (in maximum one line)

\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Are you availing the lifeline (Yes/No): \_\_No\_\_\_\_\_
2. 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]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **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 : None striked.
2. Any other point you wish to mention: \_\_Sample execution command:

make clean

make

./stage2exe c1.txt tree.txt

1. Declaration: We **PIKLU PAUL, RAJABABU SAIKIA, SAURAV VIRMANI, SIDDHANT KHARBANDA, SREYAS RAVICHANDRAN**, 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: 2017A7PS0006P Name: PIKLU PAUL

ID: 2017A7PS0007P Name: RAJABABU SAIKIA

ID: 2017A7PS0090P Name: SAURAV VIRMANI

ID: 2017A7PS0111P Name: SIDDHANT KHARBANDA

ID: 2017A7PS0275P Name: SREYAS RAVICHANDRAN

Date: 20-04-2020

---------------------------------------------------------------------------------------------------------------------------------------------

Should not exceed 6 pages.