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

Batch No. :

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS**

**Compiler Construction (CS F363)**

Group Number

12

**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:\_2016A7TS082P\_\_\_Name:\_\_\_\_\_Arpan Parikh\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_2017A7PS0153P\_\_Name:\_\_\_\_\_Pranav Panchumarthi \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_2017A7PS1179P\_\_Name:\_\_\_\_\_Asrita Venkata Mandalam\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_\_\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

1. ast.c 7. lexer.c 13. t\_nont.h 19\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. codegen.c 8. lexer.h 14\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. driver.c 9. new.txt 15\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 21\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. grammar.c 10. newparser.c 16\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 22\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. grammar.txt 11. parserdef.h 17\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 23\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. keywords.h 12. symboltable.c 18\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 24\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Total number of submitted files: \_\_\_\_\_\_\_\_\_\_\_ (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 \_\_\_\_ as per the details uploaded)
   7. Code Generator (Yes/No):\_\_\_\_\_Yes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. **Execution Status**:
   1. Code generator produces code.asm (Yes/ No):\_Yes (text file but asm struct)\_\_\_
   2. code.asm produces correct output using NASM for testcases (C#.txt, #:1-11): \_\_\_\_\_\_\_\_
   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):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   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):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
6. **Data Structures** (Describe in maximum 2 lines and avoid giving C definition of it)
   1. AST node structure: Each node contains a pointer to child, next and parent. It also contains a pointer to its respective symbol table and tree node. It has the rule number, type, visited and token number.
   2. Symbol Table structure:Each node contains name, table entry matrix, number of entries, pointers to next and previous symbol tables and the offset.
   3. array type expression structure: Has been stored in symbol table. The table entry’s type is 1 if it is an array
   4. Input parameters type structure:Has been stored in symbol table. The table entry’s type is 1 if it is an array, 2 if num, 3 if rnum and 4 if boolean.
   5. Output parameters type structure:Has been stored in symbol table. The table entry’s type is 1 if it is an array, 2 if num, 3 if rnum and 4 if boolean.
   6. Structure for maintaining the three address code(if created) :We’ve used a combination of ast, symbol table, module and tableentry structures for this.
7. **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 :Entry not found in symbol table(using lookup function) when undeclared variable is defined. Returned an error.
   2. Multiple declarations:Entry found in symbol table (using lookup function). Error returned.
   3. Number and type of input and output parameters:Stored in modules structure. If too many or too less given, error is thrown.
   4. assignment of value to the output parameter in a function: used a set of globally available registers (easier to access from recursive switch case structure of program
   5. function call semantics: \_taken care while in symbol table creation\_
   6. static type checking :\_\_taken care while in symbol table creation\_
   7. return semantics:\_\_\_\_\_\_\_\_\_Taken care in module structure\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   8. Recursion :\_Handling it with error returned\_
   9. module overloading:\_taken care in modules structure\_\_\_\_\_\_
   10. 'switch' semantics :each case has new symbol table node\_
   11. 'for' and 'while' loop semantics: entry in symbol table and taken care in code generation
   12. handling offsets for nested scopes:\_\_\_\_\_\_new node in symbol table structure\_\_\_\_\_
   13. handling offsets for formal parameters:\_\_\_\_\_\_\_symbol table structure\_\_\_\_\_\_\_\_\_\_\_\_\_
   14. handling shadowing due to a local variable declaration over input parameters:\_\_\_\_\_\_symbol table structure\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   15. array semantics and type checking of array type variables: \_\_\_symbol table entry\_\_\_

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

* 1. Scope of variables and their visibility :\_\_\_\_\_\_\_\_\_symbol table entry\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  2. computation of nesting depth:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

* For base address of an activation record:didn’t use an activation record
* for stack pointer:no stack
* others (specify):we used a switch case with globally available registers
  1. Mention the physical sizes of the integer, real and boolean data as used in your code generation module

size(integer): \_\_\_\_5\_\_\_\_\_(in words/ locations), \_\_\_\_5\_\_\_\_\_(in bytes)

size(real): \_\_\_\_\_\_\_\_\_\_8\_\_\_\_\_\_\_\_(in words/ locations), \_\_\_\_8\_\_\_\_\_\_(in bytes)

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

* 1. How did you implement functions calls?(write 3-5 lines describing your model of implementation) We moved the input parameters to globally available registers. After calling the function, we wrote the output registers to the memory location of the output list. Each module had its own designated input and output registers.
  2. Specify the following:
     + Caller's responsibilities: Push input values to specified registers. Call the function. Copy the values from output registers to respective output list memory locations
     + Callee's responsibilities:Take the data from the registers, output the values to the specified registers.
  3. How did you maintain return addresses? (write 3-5 lines):

We stored the address of the output list variables.

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

We wrote the value of the parameter’s address to the specified register which was then used by the callee.

* 1. How is a dynamic array parameter receiving its ranges from the caller?

We send the value from the symbol table.

* 1. What have you included in the activation record size computation? (local variables, parameters, both): We didn’t use activation records
  2. register allocation (your manually selected heuristic) :each register (based on its designation such as output or regular) has the same start appended by a specified number.
  3. Which primitive data types have you handled in your code generation module?(Integer, real and boolean):Integer, Real and Boolean
  4. Where are you placing the temporaries in the activation record of a function? We made ours without the activation records

1. **Compilation Details**:
   1. Makefile works (yes/No):Yes
   2. Code Compiles (Yes/ No):Yes
   3. Mention the .c files that do not compile:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   4. Any specific function that does not compile:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   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) \_\_\_\_\_Yes\_\_\_\_ and (in seconds) \_\_\_\_\_\_0.01\_\_\_\_\_\_\_\_\_
     2. t2.txt (in ticks) \_\_\_\_\_Yes\_\_\_\_ and (in seconds) \_\_\_\_\_\_0.01\_\_\_\_\_\_\_\_\_\_
     3. t3.txt (in ticks) \_\_\_\_\_Yes\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_0.01\_\_\_\_\_\_\_\_\_
     4. t4.txt (in ticks) \_\_\_\_\_Yes\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_0.01\_\_\_\_\_\_\_\_
     5. t5.txt (in ticks) \_\_\_\_\_Yes\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_0.01\_\_\_\_\_\_\_\_
     6. t6.txt (in ticks) \_\_\_\_\_Yes\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_0.01\_\_\_\_\_\_\_\_
     7. t7.txt (in ticks) \_\_\_\_\_Yes\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_0.01\_\_\_\_\_\_\_\_\_
     8. t8.txt (in ticks) \_\_\_\_\_Yes\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_0.01\_\_\_\_\_\_\_\_
     9. t9.txt (in ticks) \_\_\_\_\_Yes\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_0.01\_\_\_\_\_\_\_
     10. t10.txt (in ticks) \_\_\_\_\_Yes\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_0.01\_\_\_\_\_\_\_\_\_
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)

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

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
2. Any other point you wish to mention: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

1. Declaration: We, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Arpan Parikh\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Pranav Panchumarthi\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_Asrita Venkata Mandalam\_\_\_\_\_\_\_\_\_\_ (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:\_2016A7TS082P\_\_\_Name:\_\_\_\_\_Arpan Parikh\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_2017A7PS0153P\_\_Name:\_\_\_\_\_Pranav Panchumarthi \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_2017A7PS1179P\_\_Name:\_\_\_\_\_Asrita Venkata Mandalam\_\_\_\_\_\_\_\_\_\_\_\_

ID\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_20/4/2020\_\_\_\_

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

Should not exceed 6 pages.