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

Batch No. :

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS**

**Compiler Construction (CS F363)**

**II Semester 2022-23**

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

**Coding Details**

**(April 12, 2023)**

**Group number 51**

*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: 2019B3A70411P Name: Aviral Omar

ID: 2019B2A70966P Name: Arnav Agarwal

ID: 2019B4A70634P Name: Chandra Sekhar Reddy E

ID: 2019B5A70697P Name: Vatsal Pattani

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

1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 7\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 13\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 19\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 8\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 14\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 9\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 15\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 21\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 10\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 16\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 22\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 11\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 17\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 23\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 12\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 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)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
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): No
5. **Execution Status**:
   1. Code generator produces code.asm (Yes/ No): NA
   2. code.asm produces correct output using NASM for testcases (C#.txt, #:1-11): NA
   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): NA
   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: Contains node type, number of children and and a dynamic array of children, parent pointer and a pointer to next node of the same type (like a linked list of IDs), pointer to symbol table and a pointer to token info struct (if applicable) from stage 1.
   2. Symbol Table structure: Separate structs for function entries and symbol table entries. FunctionTable entries have information needing for activation record(like size and list of input and output parameters) and symbol table entries have offset, width, type information and the outer symbol table struct has hash table array and pointer to parent symbol table for nesting.
   3. array type expression structure: Type of array, if it is static, lower and upper bound sign and union for lower and upper bounds which may be num or IDs.
   4. Input parameters type structure: Contains name and type information(may have array type info) and pointer to next param
   5. Output parameters type structure: Same as above
   6. Structure for maintaining the three address code(if created) : NA
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: No symbol table entry found corresponding to name
   2. Multiple declarations: Symbol table entry with same name exists
   3. Number and type of input and output parameters: Count retrieved from function table entry
   4. assignment of value to the output parameter in a function: Type Info matched with output parameter list in function table entry
   5. function call semantics: Type Info matched with input parameter list in function table entry
   6. static type checking : Type of expression computed using right and left node of operators
   7. return semantics: All return parameters present in assign statement or function call output parameters
   8. Recursion : Function name not same as current function in call statement
   9. module overloading: Module already defined
   10. 'switch' semantics: Default node is the last child or last child is null
   11. 'for' and 'while' loop semantics: For iterator not in assign or function call output params, Some ID in while condition expression is assigned
   12. handling offsets for nested scopes: Offset is size in function table entry which is incremented appropriately
   13. handling offsets for formal parameters: Computed in same way before other declare statements at root level symbol table
   14. handling shadowing due to a local variable declaration over input parameters: Allowed as local variable is at nesting level below parameters
   15. array semantics and type checking of array type variables: Bound checking done for static arrays and bounds and type both matched for type matching
   16. Scope of variables and their visibility: Redeclaration allowed in nested scopes and entries retrieved from bottom to top level symbol tables
   17. computation of nesting depth: Level of parent symbol table + 1
8. Code Generation:
   1. NASM version as specified earlier used (Yes/no): NA
   2. Used 32-bit or 64-bit representation: NA
   3. For your implementation: 1 memory word = NA (in bytes)
   4. Mention the names of major registers used by your code generator: NA

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

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

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

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

* 1. How did you implement functions calls?(write 3-5 lines describing your model of implementation) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

* 1. Specify the following:
     + Caller's responsibilities:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     + Callee's responsibilities:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  2. How did you maintain return addresses? (write 3-5 lines): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

* 1. How have you maintained parameter passing? How were the statically computed offsets of the parameters used by the callee? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  2. How is a dynamic array parameter receiving its ranges from the caller? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  3. What have you included in the activation record size computation? (local variables, parameters, both): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  4. register allocation (your manually selected heuristic) :\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  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: None
   4. Any specific function that does not compile: None
   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 (Has 2 extra options at the end)
4. Specify the language features your compiler is not able to handle (in maximum one line)

All handled

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]

NA

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, Aviral Omar, Vatsal Pattani, Arnav Agarwal, Chandra Sekhar Reddy E (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: 2019B3A70411P Name: Aviral Omar

ID: 2019B2A70966P Name: Arnav Agarwal

ID: 2019B4A70634P Name: Chandra Sekhar Reddy E

ID: 2019B5A70697P Name: Vatsal Pattani

Date: 12/04/23 Group number: 51

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

Should not exceed 6 pages.