BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
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 10

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

IDs and Names of team members

1. ID: 2019B5A70688P Name: Abhijith Kannan

   ID: 2020A7PS1687P Name: Khushi Shah

   ID: 2020A7PS0003P Name: Anushka Bhattacharjee

   ID: 2020A7PS0148P Name: Deep Pandya

   ID: 2020A7PS0119P Name: Jash Ranipa

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

1	AST.c	10	makefile	19	symbolTable.c	28	t6.txt	37	c5.txt
2	AST.h	11	parser.c	20	symbolTable.h	29	t7.txt	38	c6.txt
3	driver.c	12	parser.h	21	treeADT_1.c	30	t8.txt	39	c7.txt
4	grammar.txt	13	parserDef.h	22	treeADT_1.h	31	t9.txt	40	c8.txt
5	hashtableADT.c	14	semantic_analyser.c	23	t1.txt	32	t10.txt	41	c9.txt
6	hashtableADT.h	15	semantic_analyser.h	24	t2.txt	33	c1.txt	42	c10.txt
7	lexer.c	16	setup.c	25	t3.txt	34	c2.txt	43	c11.txt
8	lexer.h	17	stackADT.c	26	t4.txt	35	c3.txt	44	
9	lexerDef.h	18	stackADT.h	27	t5.txt	36	c4.txt	45	

3. Total number of submitted files: 43 (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. Type checking Module (Yes/No): Yes
   d. Semantic Analysis Module (Yes/ no): Yes (reached LEVEL ____ as per the details uploaded)
   e. Code Generator (Yes/No):_____Yes_____

7. **Execution Status**:

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

b. code.asm produces correct output using NASM for testcases (C#.txt, #:1-11): _____1,2_____

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

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

e. Dynamic type checking works for arrays and reports errors on executing code.asm (yes/no): Yes

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 21 as uploaded on the course website for which you get the segmentation fault (t#.txt ; # 1-10 and c@.txt ; @:1-11): NA

8. **Data Structures** (Describe in maximum 2 lines and avoid giving C definition of it)
   a. AST node structure: synthesized address and inherited address stored as a Tree Node, and all original properties of Parse Tree Node are retained.

   b. Symbol Table structure:_____

   _____

   c. array type expression structure: Dummy nodes having label “ARRAY”, “L”, and “R” are used. Their children are [ID, “L”, “R”], the left bound, and right bound successfully.

   d. Input parameters type structure: Dummy node with label “IPARAM” is used, and its children are ID and the datatype of the parameter.

   e. Output parameters type structure: Dummy node with label “OPARAM” is used, and its child is ID, corresponding to the parameter.

   f. Structure for maintaining the three address code(if created) : quadruples structure used, it contains the operator, argument1, argument2, result, and a reference pointer to the symbol table.

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 :_Symbol table entry is NULL

   b. Multiple declarations: Existing symbol table entry is not present

   c. Number and type of input and output parameters:Module reuse statement doesnt match symbol table entry

   d. assignment of value to the output parameter in a function Boolean to check if its output

   e. function call semantics:Entry identifier should match list node identifier

   f. static type checking : character flag used to represent type & compared for errors

   g. return semantics:

   h. Recursion :String compare with label of module reuse statement

   i. module overloading: checking for existing entry

   j. 'switch' semantics : symbol table entry for that variable is null

   k. 'for' and 'while' loop semantics:String compare used for iterator

   l. handling offsets for nested scopes:Check parent ST entry

   m. handling offsets for formal parameters:By default set to 0 & widths specified

n. handling shadowing due to a local variable declaration over input parameters:Create nested ST to represent local variable

o. array semantics and type checking of array type variables: Use of identifier to find array and comparison with usage ST entry

p. Scope of variables and their visibility :Implemented using parent ST pointer

q. computation of nesting depth:Increment when going lower

10. Code Generation:

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

b. Used 32-bit or 64-bit representation:_____64_____

c. For your implementation: 1 memory word = _____8_____(in bytes)

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

- For base address of an activation record: _____
- for stack pointer:_____RBP_____
- others (specify):____RAX, RBX, RCX, RDX, RDI_____

e. Mention the physical sizes of the integer, real and boolean data as used in your code generation module
size(integer): _____1_____(in words/ locations), _____8_____(in bytes)
size(real): _____1_____(in words/ locations), _____8_____(in bytes)
size(booelan): _____1_____(in words/ locations), _____8_____(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: AST Generation
- Callee's responsibilities: Populate code.asm with corresponding NASM code

h. How did you maintain return addresses? (write 3-5 lines): Label is used as an intermediate to return to address. These labels are generated by a helper function.

i. How have you maintained parameter passing? How were the statically computed offsets of the parameters used by the callee? Unary operator is used to store the parameters. Offsets are used to compute the exact location of that parameter.

j. How is a dynamic array parameter receiving its ranges from the caller? Array dummy label is used to denote that the range is placed below.

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

l. register allocation (your manually selected heuristic) :_____
_____

m. Which primitive data types have you handled in your code generation module?(Integer, real and boolean): all primitive data types

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

11. **Compilation Details**:

a. Makefile works (yes/No): Yes

b. Code Compiles (Yes/ No): Yes

c. Mention the .c files that do not compile: NA

d. Any specific function that does not compile: NA

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

12. Execution time for compiling the test cases [lexical, syntax and semantic analyses including symbol table creation, type checking and code generation] :
    i. t1.txt (in ticks) _____530.000_____ and (in seconds) _0.000530_
    ii. t2.txt (in ticks) _____345.000_____ and (in seconds) _0.000530_
    iii. t3.txt (in ticks) _____1424.000_____ and (in seconds) 0.001424_
    iv. t4.txt (in ticks) _____2665.000_____ and (in seconds) __0.002665_
    v. t5.txt (in ticks) _____3696.000_____ and (in seconds) _0.003696_
    vi. t6.txt (in ticks) _____1690.000_____ and (in seconds) __0.001690_
    vii. t7.txt (in ticks) _____1232.000_____ and (in seconds) _____0.001590_
    viii. t8.txt (in ticks) _____1000.000_____ and (in seconds) _____0.001600_
    ix. t9.txt (in ticks) _____1800.000_____ and (in seconds) _0.001890_
    x. t10.txt (in ticks) _____890.000_____ and (in seconds) _____0.001490_

13. **Driver Details**: Does it take care of the **TEN** options specified earlier?(yes/no):_Yes_____
14. Specify the language features your compiler is not able to handle (in maximum one line)
    NA
15. Are you availing the lifeline (Yes/No): Yes
16. 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]
    nasm-felf64 code.asm && gcc-no-pie code.o-ocode && ./code
17. **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
18. Any other point you wish to mention: NA
19. Declaration: We, Anushka Bhattacharjee, Deep Pandya, Abhijith Kannan, Jash Ranipa, and Khushi Shah

    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: 2019B5A70688P Name: Abhijith Kannan

ID: 2020A7PS1687P Name: Khushi Shah

ID: 2020A7PS0003P Name: Anushka Bhattacharjee

ID: 2020A7PS0148P Name: Deep Pandya

ID: 2020A7PS0119P Name: Jash Ranipa

Date: 12-04-2023 Group number 10

Should not exceed 6 pages.