Safi Nassar - assignment p6

Usage:

- `make clean && make testclean && make`
- `make test`

Executing the test will generate output file of the formated program.

Prerequisites & dependencies:

- OpenJDK 11
- Java Cup
- JLex
- LLVM
- SPIM interpreter: for testing generated MIPS code http://pages.cs.wisc.edu/~larus/spim.html https://shawnzhong.github.io/JsSpim/

Tests documentation:

- use SPIM tool to verify assembly correct execution. Run the generated assembly program to test results with the expected input: e.g. run `testExample.s` with input shown in `testExample.expectedResult`
- use the Linux utility diff to compare your file of error messages with the expected files. (e.g. `diff test.expected.s test.s`)

MIPS architecture assembly code generation for C–:

- [x] Not required code generation for:
  - structs or anything struct-related (like dot-accesses)
  - repeat statement
- Semantic clarifications:
  - All parameters should be passed by value.
  - and/or operators (&& and ||) are short circuited (right operands are only evaluated if necessary).
    - If the left operand of "&&" evaluates to false, then the right operand is not evaluated (and the value of the whole expression is false); similarly, if the left operand of "||" evaluates to true, then the right operand is not evaluated (and the value of the whole expression is true).
    - for all of the other binary operators, both operands are always evaluated.

  - Use Codegen.java class for generating template code.
  - [NOT REQUIRED] ~In C– (as in C++ and Java), two string literals are considered equal if they contain the same sequence of characters. (e.g. “abc” == “abc” is true)~
  - `seq` opcode useful for ints and bools comparison
  - Boolean values should be output as 1 for true and 0 for false. Boolean values should also be input using 1 for true and 0 for false.

Tasks:

- [x] write codeGen method for various types of AST nodes.
- [x] main program:
  - [x] compiler phase errors.
  - [x] code generator should wirte code to file named by 2nd cli arg.
  - [x] remove unparse call and reporting.
  - [x] initialize Codegen class’s PrintWriter p with output file.
- [x] To allow SPIM simulator to recognize main function:
  - [NOTE REQUIED ANYMORE] ~add `__start:` to main preamble on line after `main:`~
  - [x] function exit for main: instead of `jr $ra` issue a syscall to exit with: `li $v0, 10     syscall`
- [x] Add to name analyzer or type checker wheather the program contians a function named main.
- [x] Add global / local differentiation
- [x] Add a new “offset” field to the TSym class (or to the appropriate subclass(es) of TSym). Change the name analyzer to compute offsets for each function’s parameters and local variables (i.e., where in the function’s Activation Record they will be stored at runtime) and to fill in the new offset field.
- [x] WriteStmtNode field for holding type of expression being written.
- [x] test code generation for the other kinds of statements and the expressions by writing a program that computes and prints a value
- Implement code generation for each of the following:
  - [x] global variable declarations, function entry, and function exit
  - [x] int and bool literals (just push the value onto the stack), string literals, and WriteStmtNode
  - [x] IdNode (code that pushes the value of the id onto the stack, and code that pushes the address of the id onto the stack) and assignments of the form id=literal and id=id (test by assigning then writing)
  - [x] expressions other than calls
  - [x] statements other than calls and returns
  - [x] call statements and expressions, return statements (to implement a function call, you will need a third code-generation method for the IdNode class: one that is called only for a function name and that generates a jump-and-link instruction)

Submission:

- [x] Create pdf from markdown: `pandoc README.md -o <lastname.firstname.Pn.pdf>`
  - [ ] generate markdown from javadoc and remove redundant comments or
  - [ ] generate javadoc to extract method headers: `find . -type f -name "*.java" | xargs javadoc -d ../javadoc`
- [ ] Add headers for each file
- [x] Verify code format
- [ ] Verify code execution on CSL machines
- [x] lastname.firstname.lastname.firstname.P6.zip
  - `tar -czvf src.tar.gz ./src` +—+ deps/ +—+ ast.java +—+ cminusminus.cup +—+ cminusminus.jlex +—+ Codegen.java +—+ DuplicateSymException.java +—+ EmptySymTableException.java +—+ ErrMsg.java +—+ Makefile +—+ P6.java +—+ TSym.java +—+ SymTable.java +—+ Type.java +—+ lastname.firstname.lastname.firstname.P6.pdf