DPCC: DParo's Own C-Alike Compiler

Davide Paro

December 2020

Project Description

This project is the implementation of an assignment for a course on **Compilers** at the department of Computer Engineering Master Degree Padova (ITA).

The assignment consists in implementing a toy compiler (mostly the frontend side) for a toy language at our liking.

The assignment specs out the how the compiler should be composed. We can in fact distinguish these macro components.

- **Input Stage** deals with the input byte stream that composes the source of the program.
- **Lexer/Scanner** has the purpose of grouping characters (lexical analysis) together to compose compunded structures (called tokens). For the project assignment we can use **Flex** to aid in the code generation for the scanner.
- **Parser** for performing the syntax analysis. It is what defines the look & fell (grammar) of the language. For the project assignment we can use **Bison** to ad in the implementation of a good parser.
- **Intermediate Code Generator**. The ultimate purpose of a compiler is to produce something useful. In this project assignment we are not requested to implement a proper backend. Thus, we need to emit a 3AC representation of our input program. More in this later.

In particular the final Intermediate representation that we need to emit is based on Three Address Code (3AC), that is each statement can only have 1 operand at the left hand side of the assignment, and 2 operands at the left hand side of the assignment, and an operator driving the operation that should be performed.

You can view more about 3AC at the following wikipedia link.

In practice the emitted 3AC code is on itself a partially valid C program, it's only missing variable declarations at the top for the temporary variables.

For the specification of the Intermdiate Code that is generated please refer to appendix A

So the project requires to produce this kind of 3AC / C hybrid. Control flow is allowed to be implemented trough the usage of C labels and simple if conditional followed by a goto statement. Inside the if conditional there can only be a single element composing the expression.

The assignment requires the following features from the programming langugage that we should develop:

- Variables declaration, initialization and assignment
- Handling of variable scopes. Variable names can be reused if out of scope. Variable shadowing may or may not warn/fail/pass depending on the design choices.
- Only 2 types of variables: integers, booleans
- Assignment statements, print statemnts, if statements, and at least 1 loop statement at will (while, for, ...)
- Handling of simple mathematical expressions that we can encounter in common programming languages, addition, subtraction, multiplication, division, modulo, etc ...
- **Function definition, function calls, and custom user definable types are not required**

Intermediate Code Generation: 3AC

The prospsoed implementation

- **Input Stage**. The compiler only supports file. The input stream is implemented in the following way. The file is loaded and entirely copied into a memory buffer. This is more than adequate for what it's necessary to do for the project assignment.
- Un **Lexer/Scanner** per la tokenizzazione del sorgente da caratteri a tipi di dati strutturati. La scelta ricade su **Flex** per la gestione dell'analisi lessicale
- Un **Parser** per implementare la sintassi del linguaggio e gestire l'analisi sintattica. La scelta ricade su **Bison** per la gestione di questa componente
- Un semplice **generator di codice intermedio**. Il progetto provede di generare un ibrido Assembly/C/3AC come semplice esempio di gestione di generazione

Hello world: More hello world

Problem analysis

...

Program design

...

Evaluation of the program

...

Process description

...

Conclusions

...

Appendix A: Structure of the Intermediate Code

Appendix: program text

```
let a: int[] = 10;
let s = "Hello world";
{

}
```

```
int main() {

}
char **argv;
```

Appendix B: Example Program Iterative Merge Sort

```
let array = [
    15, 59, 61, 75, 12, 71, 5, 35, 44,
    6, 98, 17, 81, 56, 53, 31, 20, 11,
    45, 80, 8, 34, 71, 83, 64, 28, 3,
    88, 50, 48, 80, 5
];


for (let curr_size = 1; curr_size < len; curr_size = 2 * curr_size) {
    for (let left_start = 0; left_start < len - 1; left_start = left_start + 2 * curr_size) {
        let mid = len - 1;

        if ((left_start + curr_size - 1) < len - 1) {
            mid = left_start + curr_size - 1;
        }

        let right_end = len - 1;

        if ((left_start + 2 * curr_size - 1) < len - 1) {
            right_end = left_start + 2 * curr_size - 1;
        }

        {
            let l = left_start;
            let m = mid;
            let r = right_end;
            let n1 = m - l + 1;
            let n2 = r - m;

            let L: int[1024];
            let R: int[1024];

            for (let i = 0; i < n1; i++) {
                L[i] = array[l + i];
            }

            for (let i = 0; i < n2; i++) {
                R[i] = array[m + 1 + i];
            }


            let i = 0;
            let j = 0;
            let k = l;

            while (i < n1 && j < n2) {
                if (L[i] <= R[j]) {
                    array[k++] = L[i++];
                } else {
                    array[k++] = R[j++];
                }
            }

            while (i < n1) {
                array[k++] = L[i++];
            }
            while (j < n2) {
                array[k++] = R[j++];
            }
        }
    }
}

print("Sorted array\n");
print(array);
```