

# 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 of Computer Engineering at Padua.

The project consists in an implementation of a toy compiler for a toy language for which we are free to design the syntax however we like.

- **Lettura del sorgente.** In particolare si può supporre una tradizionale lettura da un file.
- 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 prevede di generare un ibrido Assembly/C/3AC come semplice esempio di gestione di generazione

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

## Intermediate Code Generation: 3AC

### Problem analysis

...

### Program design

...

### Evaluation of the program

...

### Process description

...

### Conclusions

...

## Appendix A: Structure of the Intermediate Code

### Appendix: program text

```
1 let a: int[] = 10;  
2 let s = "Hello world";  
3 {  
4  
5 }
```

```
1 int main() {  
2  
3 }  
4 char **argv;
```

## Appendix B: Example Program Iterative Merge Sort

```
1  let array = [  
2      15, 59, 61, 75, 12, 71, 5, 35, 44,  
3      6, 98, 17, 81, 56, 53, 31, 20, 11,  
4      45, 80, 8, 34, 71, 83, 64, 28, 3,  
5      88, 50, 48, 80, 5  
6  ];  
7  
8  
9  for (let curr_size = 1; curr_size < len; curr_size = 2 * curr_size) {  
10     for (let left_start = 0; left_start < len - 1; left_start = left_start + 2 * curr_size) {  
11         let mid = len - 1;  
12  
13         if ((left_start + curr_size - 1) < len - 1) {  
14             mid = left_start + curr_size - 1;  
15         }  
16  
17         let right_end = len - 1;  
18  
19         if ((left_start + 2 * curr_size - 1) < len - 1) {  
20             right_end = left_start + 2 * curr_size - 1;  
21         }  
22  
23         {  
24             let l = left_start;  
25             let m = mid;  
26             let r = right_end;  
27             let n1 = m - l + 1;  
28             let n2 = r - m;  
29  
30             let L: int[1024];  
31             let R: int[1024];  
32  
33             for (let i = 0; i < n1; i++) {  
34                 L[i] = array[l + i];  
35             }  
36  
37             for (let i = 0; i < n2; i++) {  
38                 R[i] = array[m + 1 + i];  
39             }  
40  
41  
42             let i = 0;  
43             let j = 0;  
44             let k = l;  
45  
46             while (i < n1 && j < n2) {  
47                 if (L[i] <= R[j]) {  
48                     array[k++] = L[i++];  
49                 } else {  
50                     array[k++] = R[j++];  
51                 }  
52             }  
53  
54             while (i < n1) {  
55                 array[k++] = L[i++];  
56             }  
57             while (j < n2) {  
58                 array[k++] = R[j++];  
59             }  
60         }  
61     }  
62 }  
63  
64 print("Sorted array\n");  
65 print(array);
```