Team 20

Osama Magdy 1 14

Reem Emad 1 33

Kareem Taha 2 9

Yousef Gamal 2 39

Project Description

This project is a compiler for a language called "Kak++" which is a subset of the Go language. The compiler is written in C and it generates assembly code.

Tools used

- Flex
- Bison
- C
- Makefile
- Bash
- Python
- PyQt5

How to run

To run the test.txt file

1. chmod u+x ./test.sh
2. ./test.sh

To run a custom file

1. make
2. ./comp `<yourfile>`

Language Descriptions

Tokens

1- Variable and constant Declaration

```
val x: Int = 10;

var x: Int = 10;

val x: Int;
var x: Int;
```

- int
  - double
  - string
  - bool
  - void

2- Mathematical and Logical expressions

- Mathematical operator
  - +, -, *, /, **, %
- Logical operator
  - <, >, <=, >=, ==, !=
  - and , or , not ,

3 - Assignment statement

```
x = 5;
x = "Hello";
x = 5.5;
x = true;
```

4 - If else statement

```
if (x < 5 and y > 10) {
  x = x + 1;
} else {
};
```

```
if 1 {
  x = x +1 ;
} else {
  if (x < 5 and y > 10) {
    x = x + 2;
  } else {
    x = x + 3
```

```
  };
};
```

```
if (x < 5 and y > 10) {
  x = x + 2;
} else {
  x = x + 3
};
```

5 - While loop

```
while (1) {
  x = x + 1;
};
```

6 - Repeat until

```
until (x == 5){
  x = x + 1;
};
```

7 - For loop

```
for i:(x,2,4) {
  x = x + 1;
};
```

8 - Switch case

```
  switch x {
      1:{

      }
      1:{

      }
      2:{

      }
      else:{

      }

  };
```

9 - Function declaration

```
fun func1(int x, int y) int {
  var z: Int = x + y;

  z;
}
```

```
fun func2(int x) void {
  x = x + 1;
}
```

```
fun func3() void {
  x = x + 1;
}
```

10 - Function call

```
  x = func1(5, 10);
  func2(5, 10,
  566,
```

```
    568,);
    func3();
```

11 - Block structure

```
    do {
        do{
            do{

            };
        };
    };
```

14- Enum

```
type BOOL = TRUE | FALSE | OTHER;
```

15 - Type declaration (optional)

```
type Bool = MkInt {str:int, kak:KKK}
          | MkBool{str:int, kak:KKK}
          ;
```

List of Quadrables

- "_start" --> start of the program
- "PUSH 1" --> Is to push the number 1 to the stack
- "CALL *FUNCTION_int_to_double*" --> Is to call the functions named "*FUNCTION_int_to_double*"
- "STORE 7" $--> Istostorethelastvalueonthestackinthememorylocation7$
- "*FUNCTION_fib*:" --> Is the label when you want to jump

- "LOAD $7" --> Is to load from the memory location and push it into the stack
- "JMPF *if_12*" --> Jump false which takes the value of the stack and check if it's false. It jumps to the label "*if_12*"
- "JMP *if_12*" --> Will jump anyways to the label "*if_12*"
- "RETURN" --> Return from the function you're in to the address of the pointer on the stack
- "SUB" --> Operation that will take the last two values in the stack and compute the result (subtraction) and push it again on the stack.
    - Same with "ADD", "LT"(less than), "GT", "EQ", "NEQ", "MUL", "DIV", "MOD", "EXP", "OR", "LEQ", "GEQ", "NOT" (this only takes one value not two)