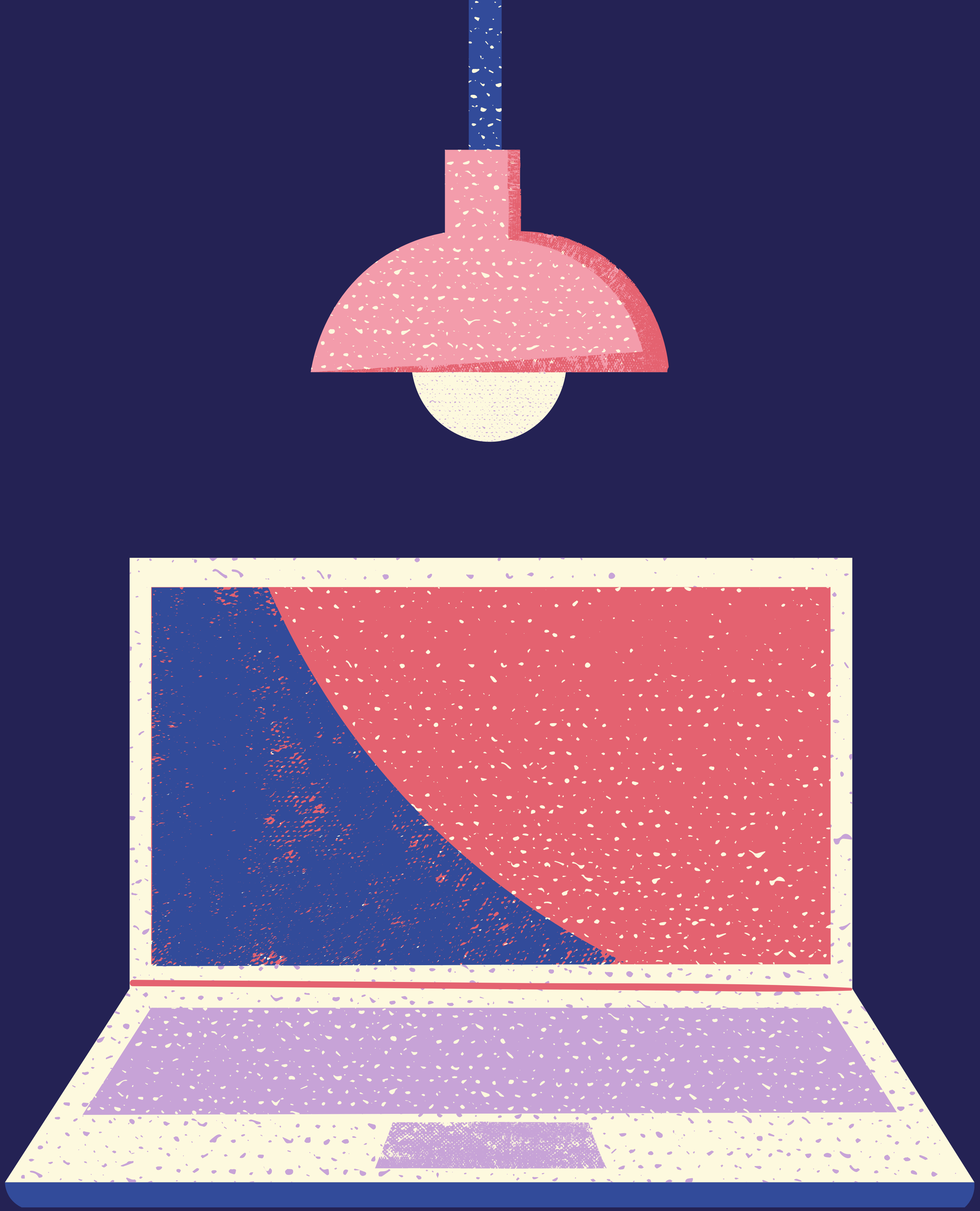


ALPHAX compiler

First delivery

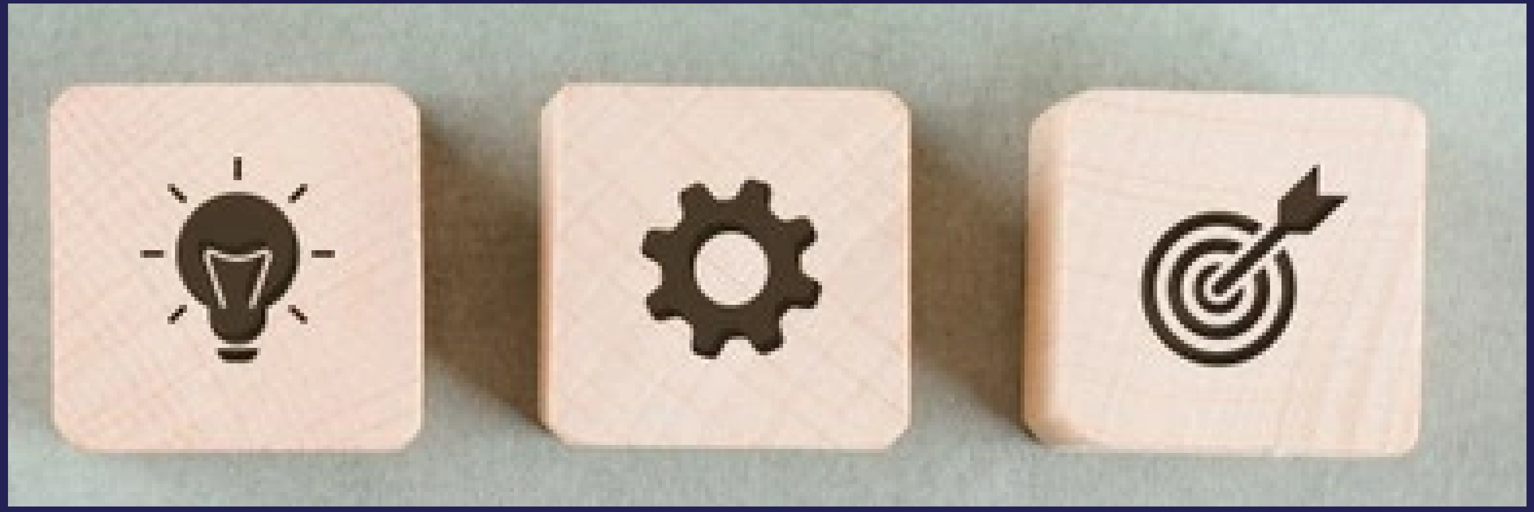
DEVELOPERS

ALPHAX Team

- Flores Constantino Diego
- Rojas Castañeda Karen Arleth

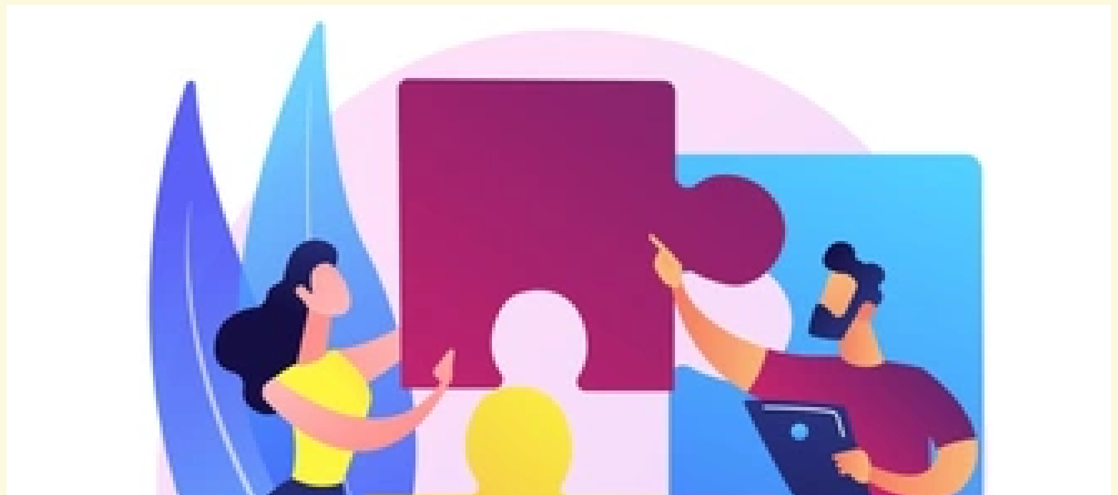
Objective.

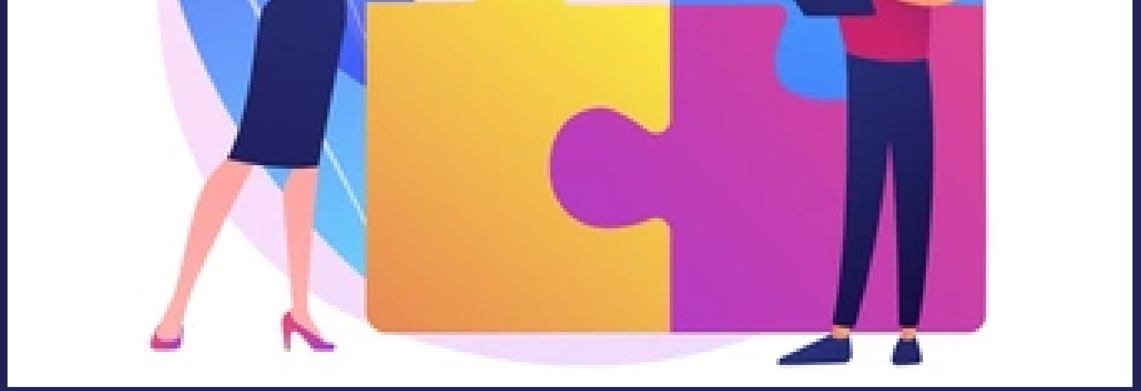
Develop a C compiler to accoplish the requeriments of the client Norberto Ortigoza, this compiler will be developed in Elixir programming language.



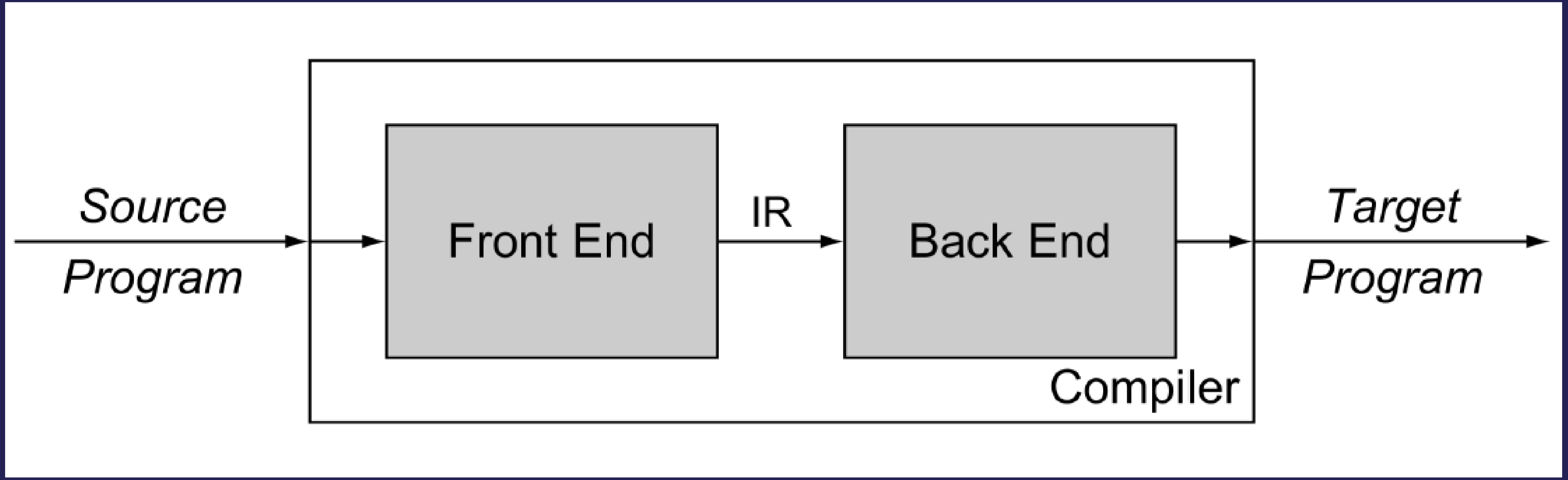
Use of github

For the version control we used github repository



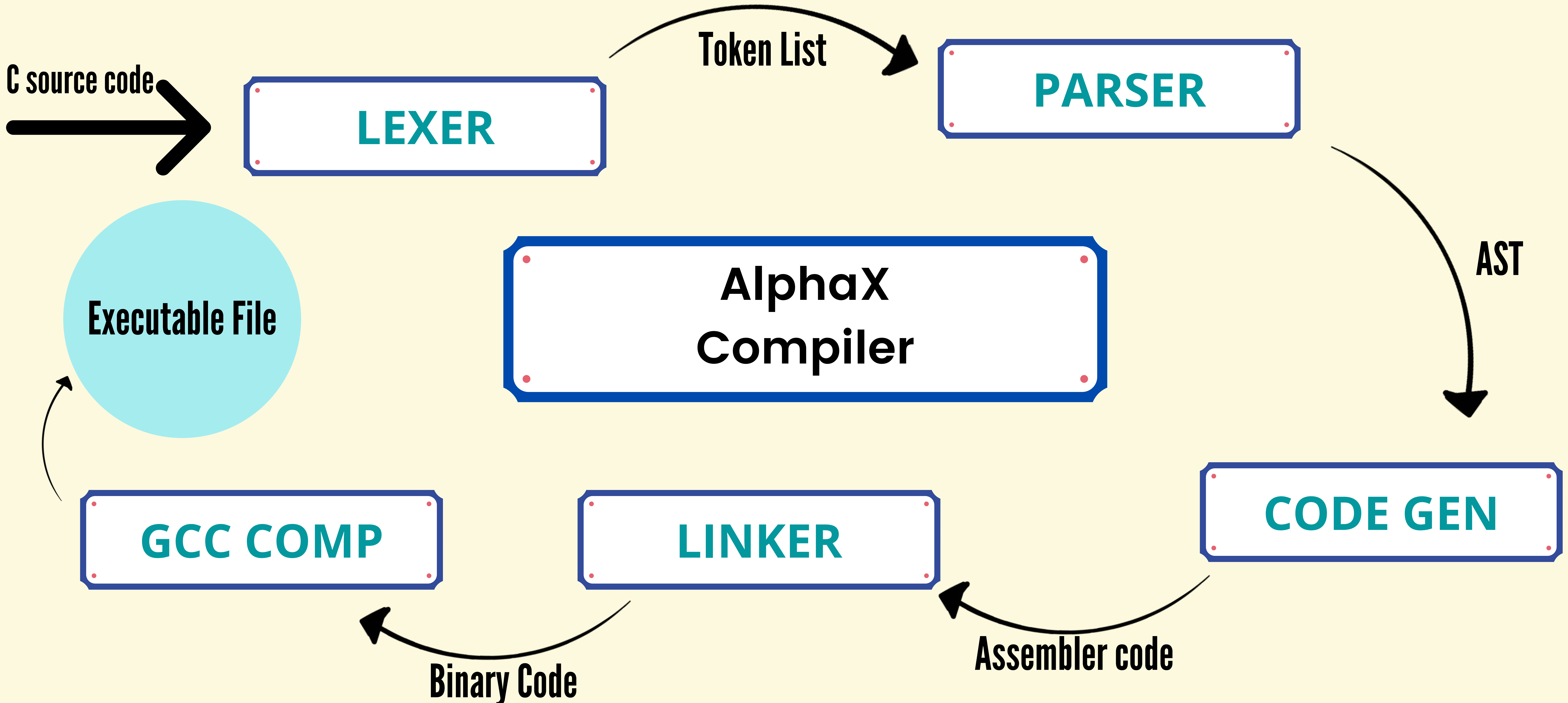


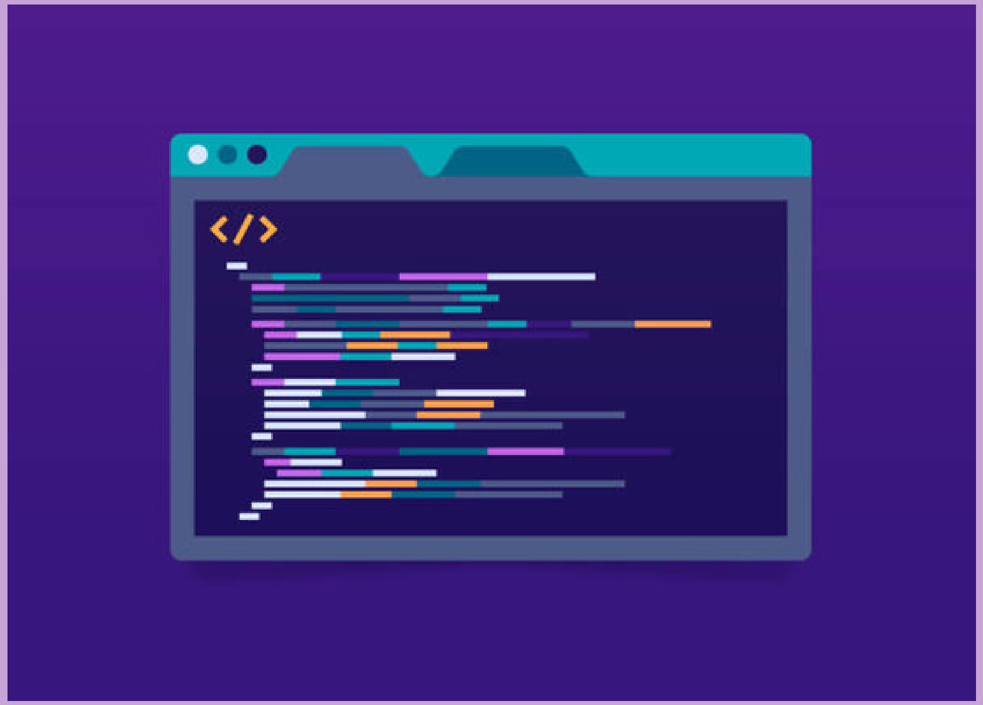
Structure



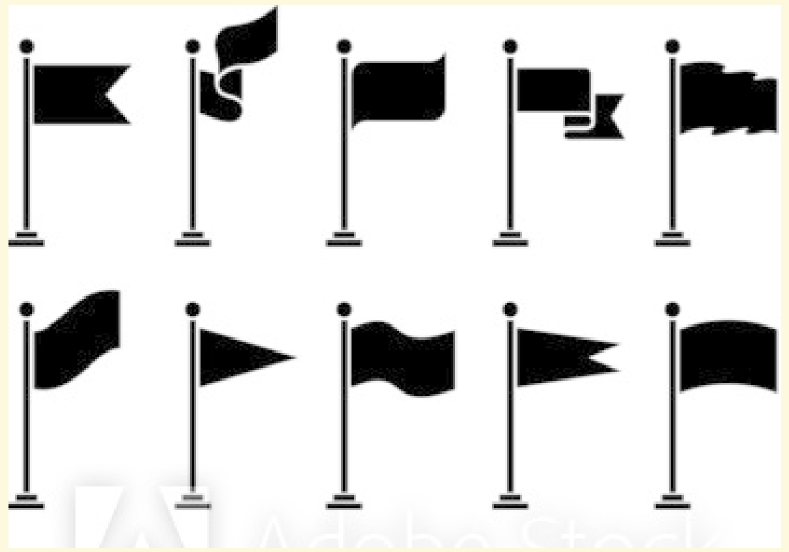
Our compiler has two main branches, frontend and backend. We developed frontend, it has many functions, check syntaxis or semantic rules following significance in source code.

Architecture

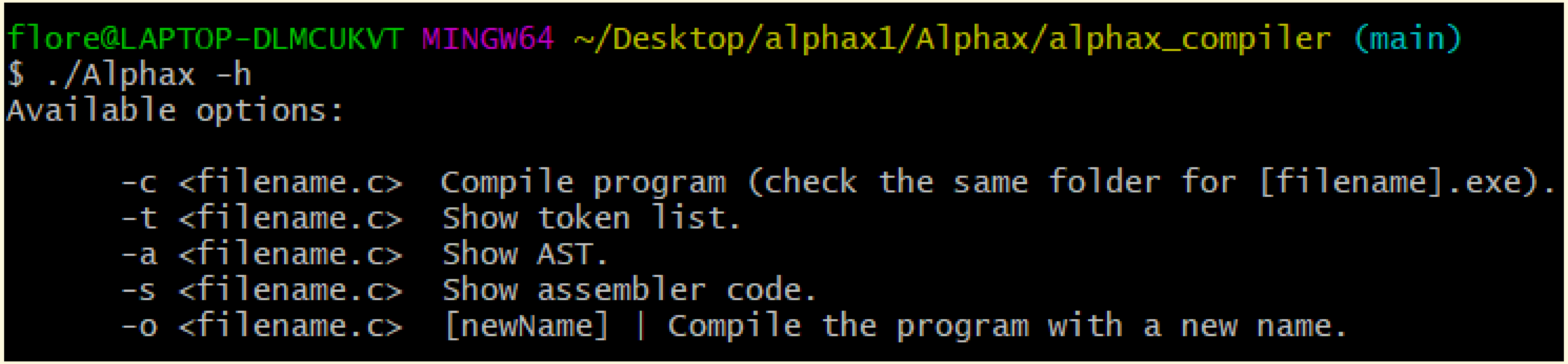




Implementation

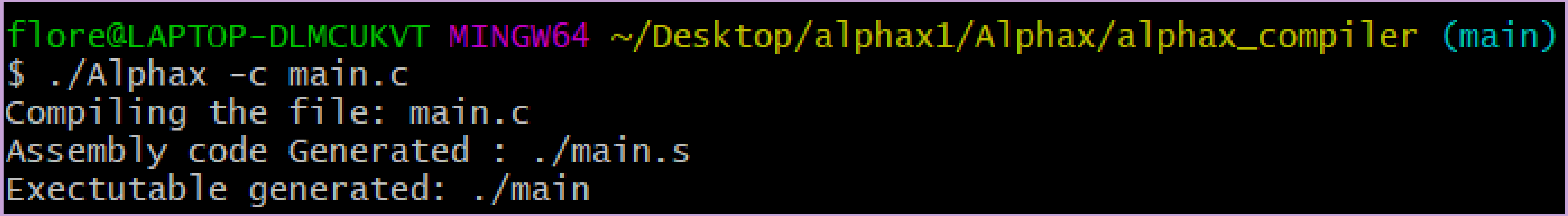


Flags input



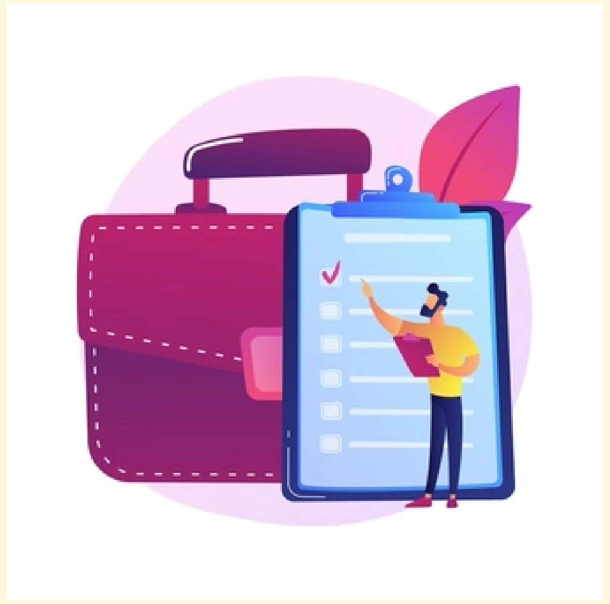
```
flore@LAPTOP-DLMCUKVT MINGW64 ~/Desktop/alphax1/Alphax/alphax_compiler (main)
$ ./Alphax -h
Available options:

      -c <filename.c>  Compile program (check the same folder for [filename].exe).
      -t <filename.c>  Show token list.
      -a <filename.c>  Show AST.
      -s <filename.c>  Show assembler code.
      -o <filename.c>  [newName] | Compile the program with a new name.
```

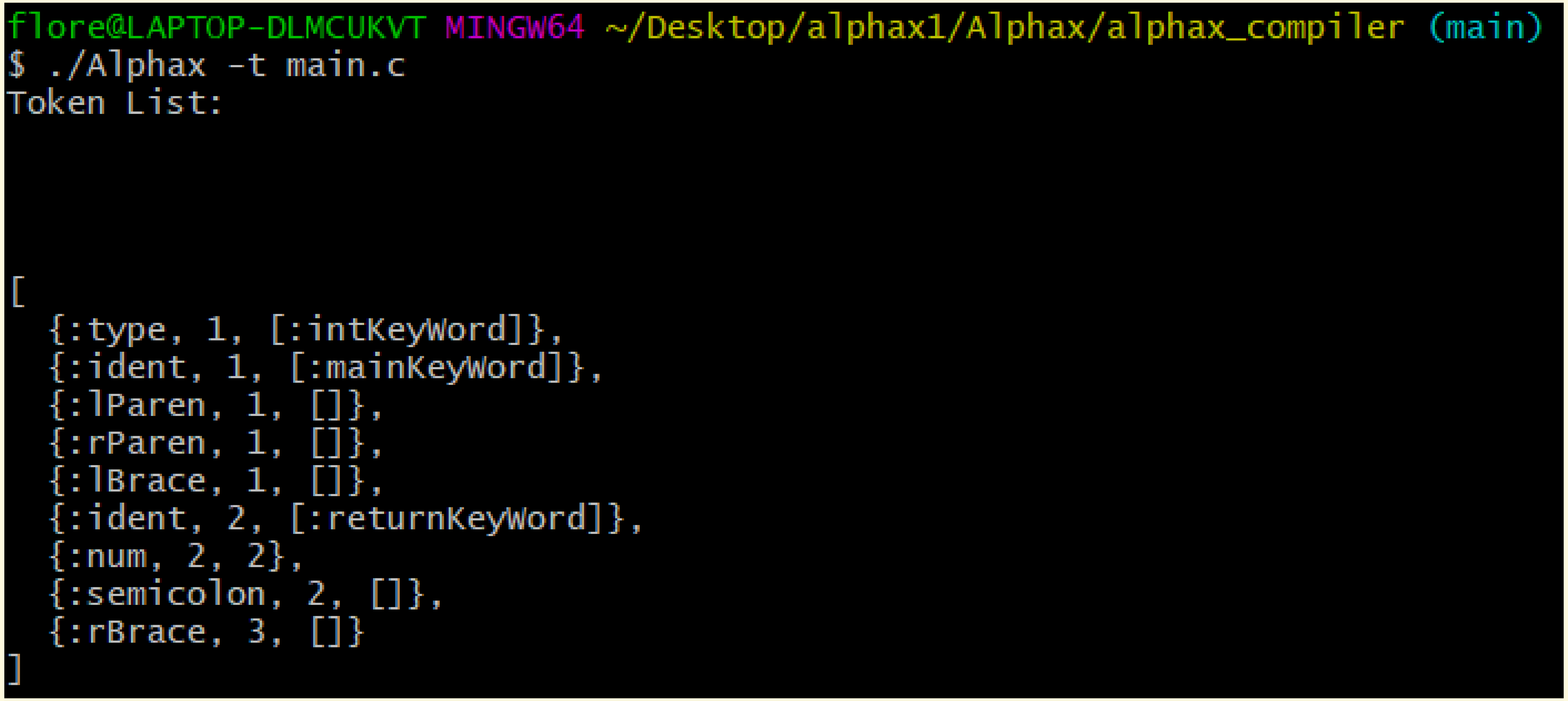
Basic Compilation



```
flore@LAPTOP-DLMCUKVT MINGW64 ~/Desktop/alphax1/Alphax/alphax_compiler (main)
$ ./Alphax -c main.c
Compiling the file: main.c
Assembly code Generated : ./main.s
Exectutable generated: ./main
```

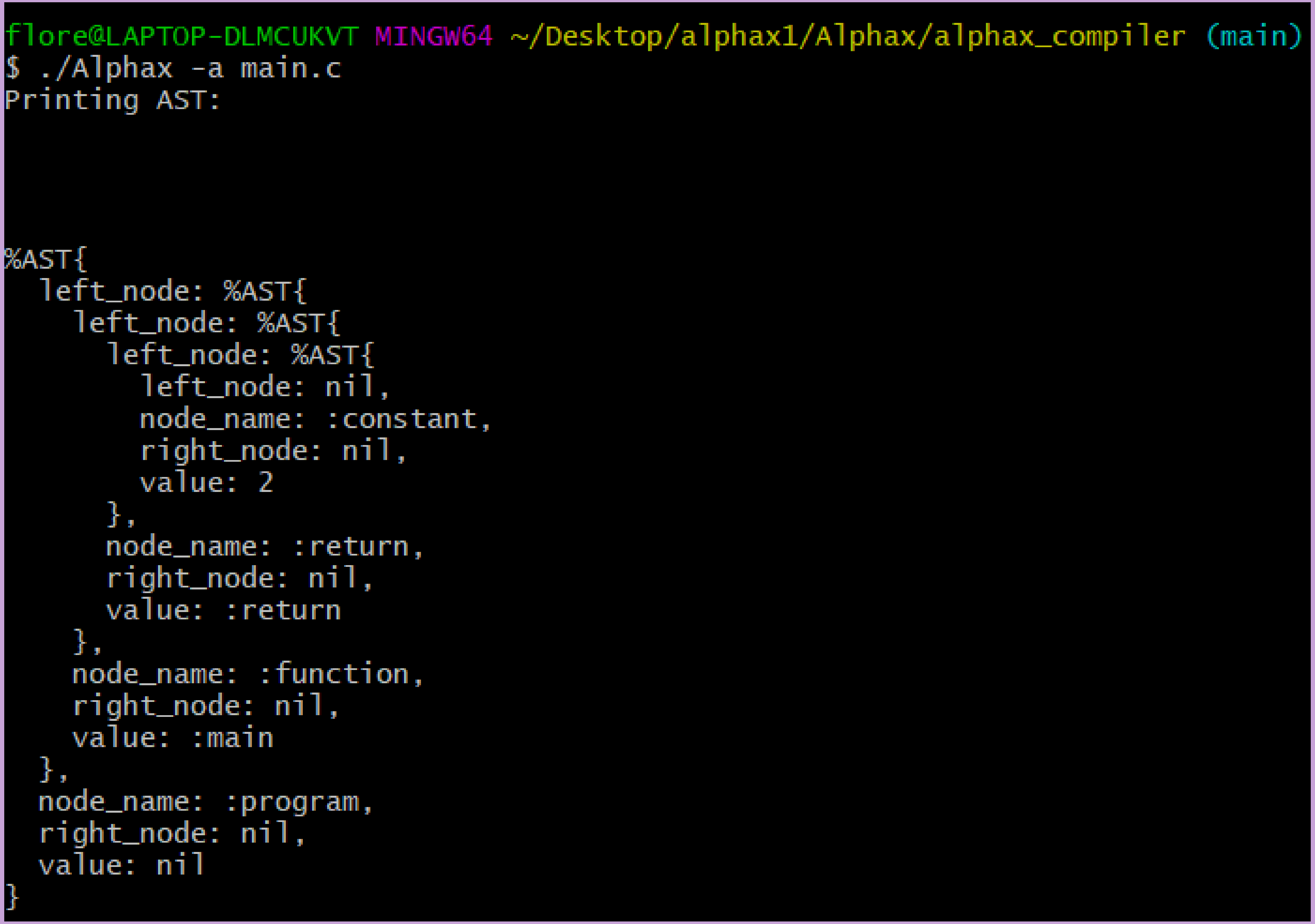
Token List





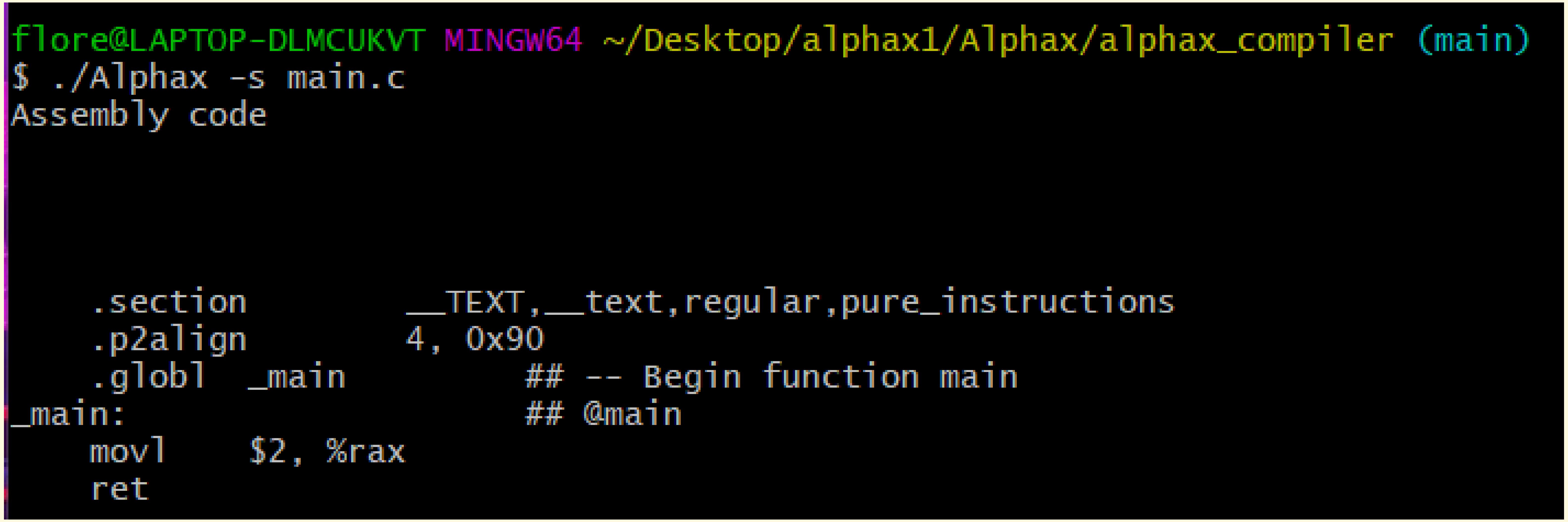
```
flore@LAPTOP-DLMCUKVT MINGW64 ~/Desktop/alphax1/Alphax/alphax_compiler (main)
$ ./Alphax -t main.c
Token List:





[
  {:type, 1, [:intKeyWord]},
  {:ident, 1, [:mainKeyWord]},
  {:lParen, 1, []},
  {:rParen, 1, []},
  {:lBrace, 1, []},
  {:ident, 2, [:returnKeyWord]},
  {:num, 2, 2},
  {:semicolon, 2, []},
  {:rBrace, 3, []}
]
```

Abstract Syntax Tree



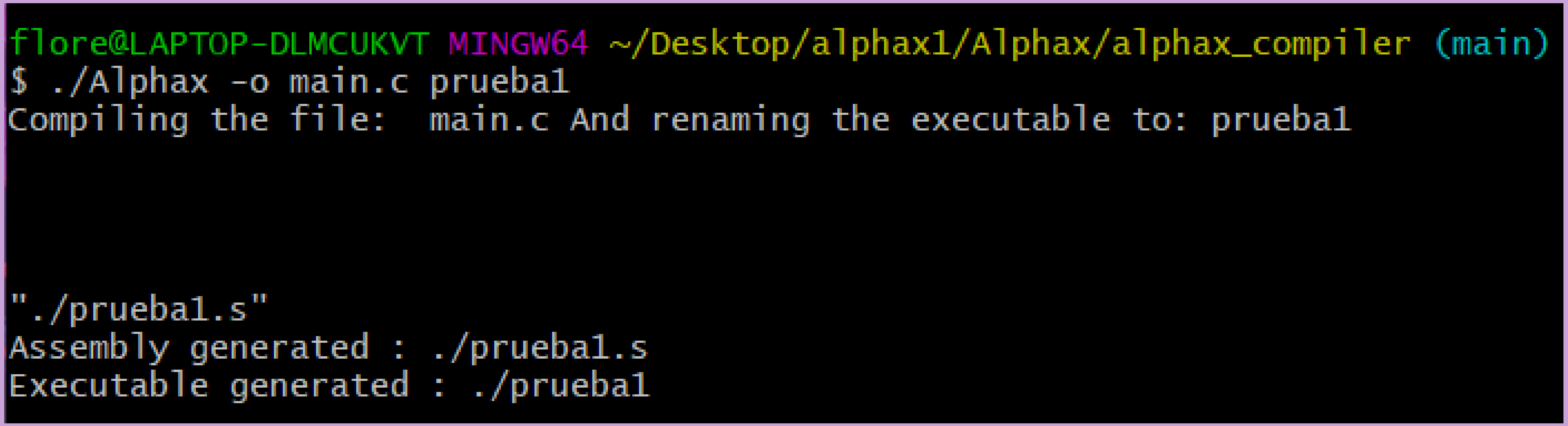
```
flore@LAPTOP-DLMCUKVT MINGW64 ~/Desktop/alphax1/Alphax/alphax_compiler (main)
$ ./Alphax -a main.c
Printing AST:




%AST{
  left_node: %AST{
    left_node: %AST{
      left_node: %AST{
        left_node: nil,
        node_name: :constant,
        right_node: nil,
        value: 2
      },
      node_name: :return,
      right_node: nil,
      value: :return
    },
    node_name: :function,
    right_node: nil,
    value: :main
  },
  node_name: :program,
  right_node: nil,
  value: nil
}
```

Assembler Code



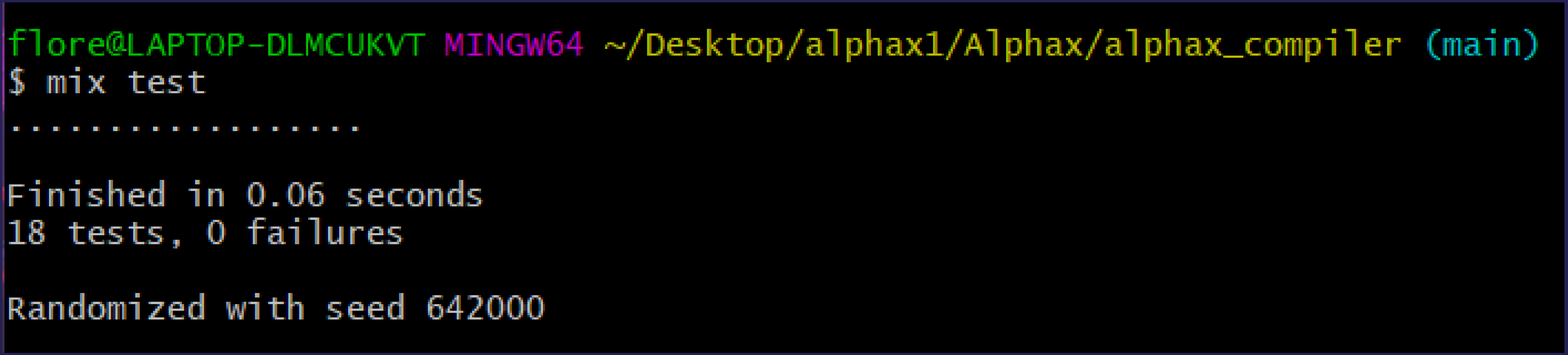
```
flore@LAPTOP-DLMCUKVT MINGW64 ~/Desktop/alphax1/Alphax/alphax_compiler (main)
$ ./Alphax -s main.c
Assembly code




        .section        __TEXT,__text,regular,pure_instructions
        .p2align        4, 0x90
        .globl  _main           ## -- Begin function main
_main:                          ## @main
        movl    $2, %rax
        ret
```

Compile and save with a new file name



```
flore@LAPTOP-DLMCUKVT MINGW64 ~/Desktop/alphax1/Alphax/alphax_compiler (main)
$ ./Alphax -o main.c prueba1
Compiling the file:  main.c And renaming the executable to: prueba1




"./prueba1.s"
Assembly generated : ./prueba1.s
Executable generated : ./prueba1
```

Test Plan



```
flore@LAPTOP-DLMCUKVT MINGW64 ~/Desktop/alphax1/Alphax/alphax_compiler (main)
$ mix test
..................

Finished in 0.06 seconds
18 tests, 0 failures

Randomized with seed 642000
```

CONCLUSIONS & LEARNED LESSONS