Primera Entrega Compilador

Barrientos Veana Luis Mauricio.

González Pacheco Leonardo Alonso.

Martínez Matías Joan Eduardo.

Rosales Romero Ricardo.

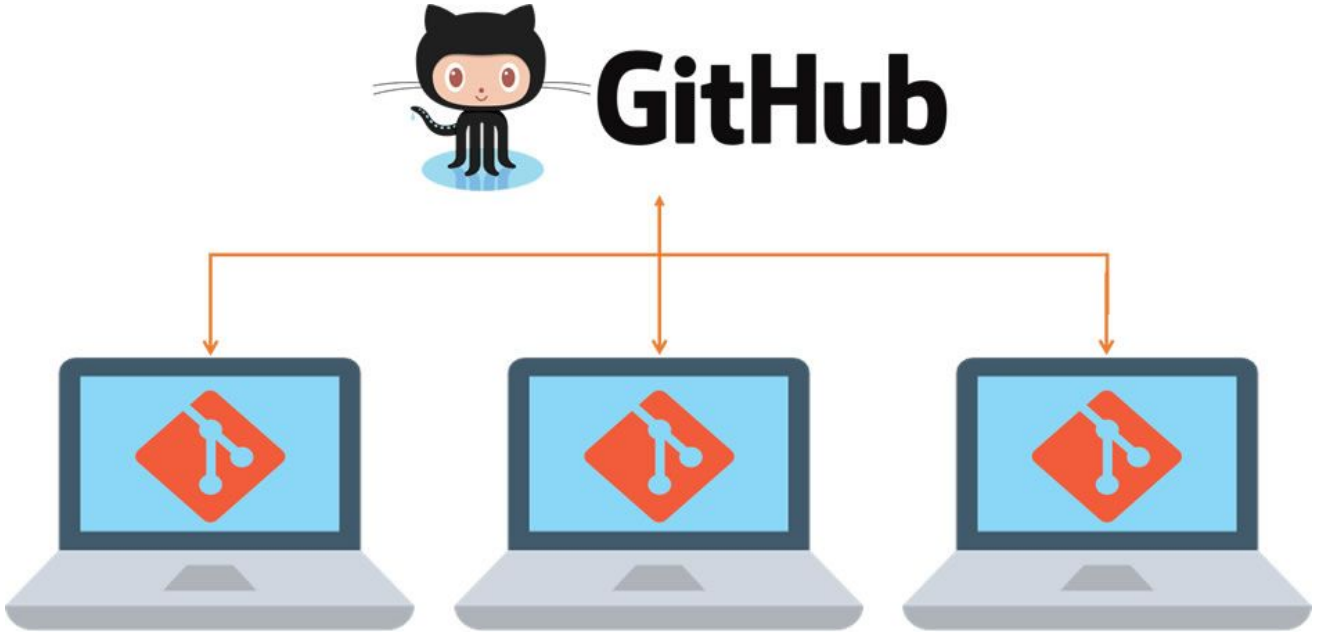
Objetivo:

Desarrollar un compilador en C que cumpla con los requerimientos del cliente Norberto Jesús Ortigoza Márquez ;el proyecto se desarrollara en elixir. Las fechas de entrega a continuación serán proporcionadas en el plan de trabajo.

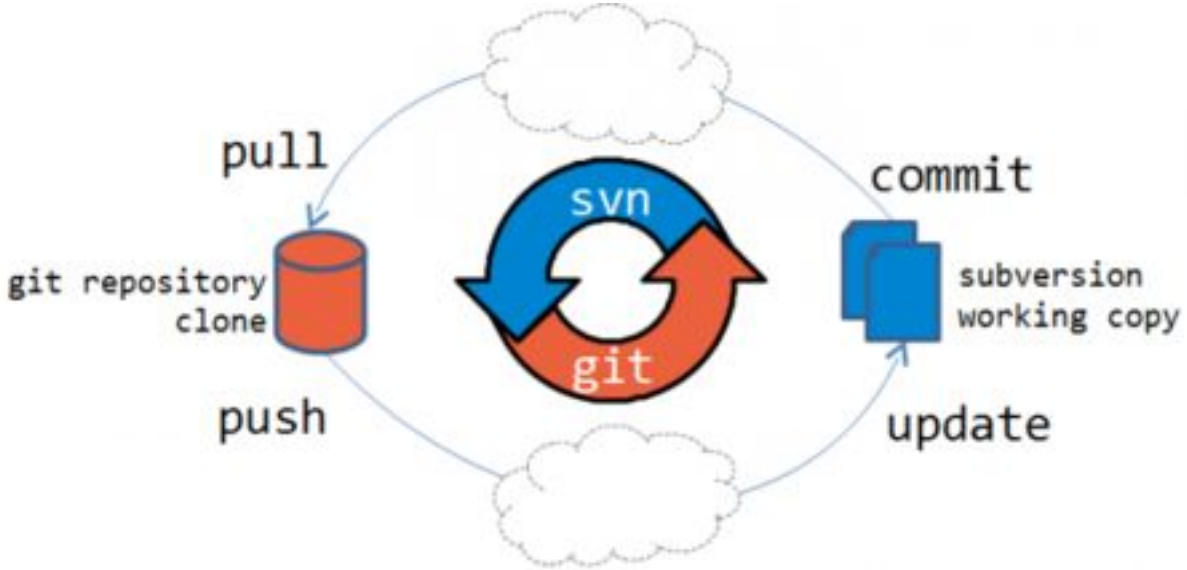
Gremlins Working P8lan

# Activity	Activity	February / 3-7 / 2020 (1)	February / 10-14 / 2020 (2)	February / 17-21 / 2020 (3)	February / 24-28 / 2020 (4)	March / 2-6 / 2020 (5)	March / 9-13 / 2020 (6)	March / 16-20 / 2020 (7)	23-27 / marzo / 2020 (8)	February / 30 - April / 3 / 2020 (9)	April / 6-10 / 2020 (10)	April / 13-17 / 2020 (11)	April / 20-24 / 2020 (12)	April / 27 - May / 1 / 2020 (13)	May / 4-8 / 2020 (14)	May / 11-15 / 2020 (15)	May / 18-22 / 2020 (16)	Resourses
1	Read Elixir Manual	Tue	Thu															All
2	Requirements	Tue		Tue	Tue													Project Manayer
3	Validate the Requirements	Tue																All
4	Analysis		Mon	Thu	Thu													All
5	Architecture Design		Thu		Fri													System Architect
6	Lexico Analysis			Sun		Sun												All
7	Global Funtion (Main)																	All
8	Lifting of Requirements					Mon												Project Manayer
9	First Delivery						Tue	Thu										All

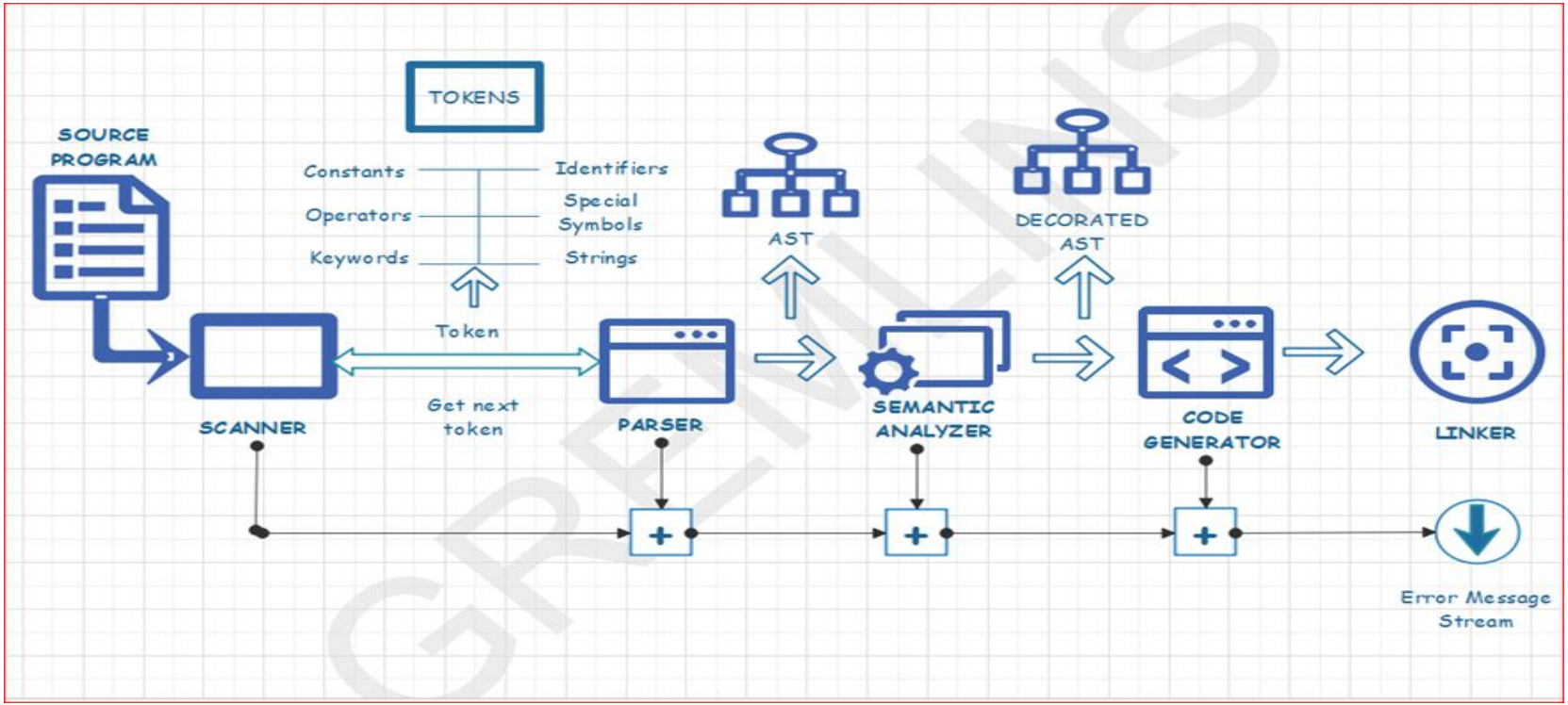
Uso de git y github



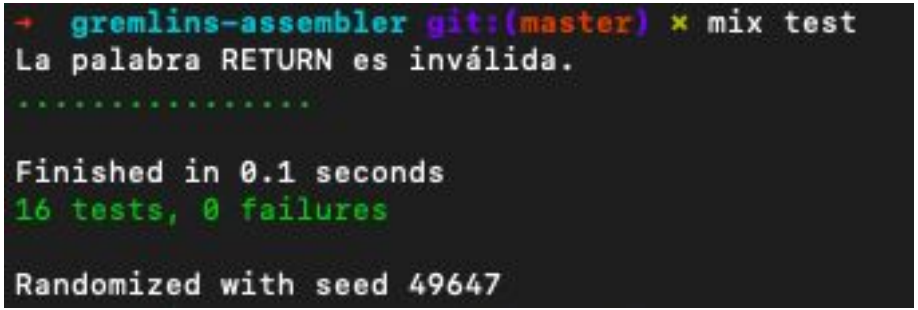
Uso de git y github



Arquitectura (Pipe-filter pattern)



Pruebas


gremlins-assembler git:(master) × mix test
La palabra RETURN es inválida.
.....
Finished in 0.1 seconds
16 tests, 0 failures
Randomized with seed 49647

Complicaciones que se tuvieron

Hubo problemas con github ya que en un comienzo no se aceptaron las invitaciones proporcionadas por el profesor.

La planeación en las fechas de entregas y reuniones se complicó un poco por los tiempos de cada quien.

Para la realización del código se tuvieron que entender correctamente la funcionalidad de todas las partes del compilador.

Second Delivery

Barrientos Veana Luis Mauricio.

González Pacheco Leonardo Alonso.

Martínez Matías Joan Eduardo.

Rosales Romero Ricardo.

Aspects that were improved.

Organization.

Handling tools.

Planning for more realistic and achievable goals.

Changes

3 unary operators added.

Negation
Complementary bit
Logical negation

Activities.

Code update.

Changes in git and Github.

Documentation was improved and the objectives of the second installment were explained more clearly.

Coupling Architecture

```
def manager(file, path, opt) do
#Utilizando "with" se procesa el archivo. Si hay error deja de hacer la compilación.
with  {:ok, tok} <- Lexer.scan_word(file, opt),
      {:ok , ast} <- Parser.parse_token_list(tok, opt),
      {:ok, asm} <- Generador.code_gen(ast, opt, path),
      {:ok, _}  <- Linker.outputBin(asm, opt, path)
      do
      IO.puts("Finalizó la compilación de forma exitosa.")
else
#Se muestra el motivo del error o la salida de la opción seleccionada al compilar
      {:error, error} -> IO.puts(error)
      {:only_tokens, _} -> IO.puts("Lista de tokens.")
      {:only_ast, _} -> IO.puts("Árbol Sintáctico.")
      {:only_asm, path_asm} ->IO.puts(path_asm)
```

Adding Unary Operators to Lexer

```
def lex_raw_tokens(program) when program != "" do #Búsq
  {token, resto} =
  case program do
      "{" <> resto -> {:open_brace, resto}
      "}" <> resto -> {:close_brace, resto}
      "(" <> resto -> {:open_par, resto}
      ")" <> resto-> {:close_par, resto}
      ";" <> resto -> {:semicolon, resto}
      "return" <> resto -> {:return_Reserveword, resto}
      "int" <> resto -> {:int_Reserveword, resto}
      "main" <> resto -> {:main_Reserveword, resto}
      "-" <> resto -> {:negation_Reserveword, resto}
      "!" <> resto -> {:logicalNeg, resto}
      "~" <> resto -> {:bitewise_Reserveword, resto}
```

Adding Unary Operators to Parser

We are created new function , that benefits the controll and order to the process for creating nodes :

```
def pars_factor(tokens) do
  #Parseando con operador unario
  if List.first(tokens) == :negation_Reserveword or List.first(tokens) == :bitewise_Reserveword  or List.first(tokens) == :logicalNeg
```

Adding Unary Operators to Code Generator

```
def codigo_gen(:negation_Reserveword, _, codigo, _) do
  codigo <> """
    neg  %eax
  """
end

def codigo_gen(:logicalNeg, _, codigo, _) do
  codigo <> """
    cmp     $0, %rax
    mov     $0, %rax
    sete    %al
  """
end
```

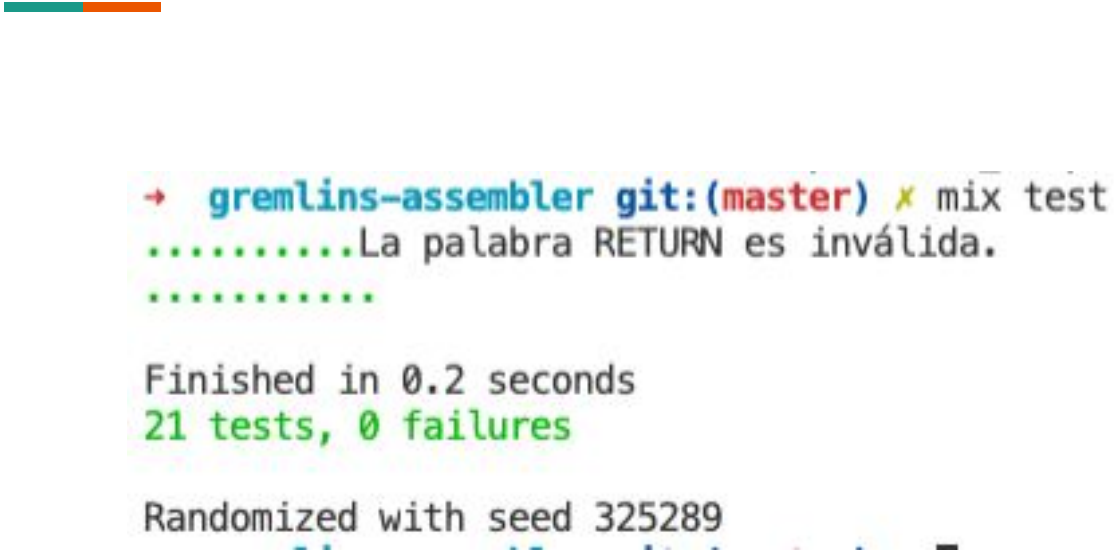
```
def codigo_gen(:bitewise_Reserveword, _, codigo,post_stack) do
  if List.first(post_stack) == "return" do
    codigo <> """
      not     %rax
    """
  else
    codigo <> """
      not     %rax
      push    %rax
    """
  end

end
```

Tests

```
test "Prueba 6 de Nora Sandler: Operador unario, complemento bit a bit de 0" do
  token_list = Lexer.scan_word(File.read!("test/bitwise_zero.c"), :no_output);
  assert  Parser.parse_token_list(elem(token_list, 1), :no_output) ==
    {:ok, {:program, "program",
           {:function, "main",
            {:return_Reserveword, "return",
             {:bitewise_Reserveword, "~", {:constant, 0, {}, {}}, {}}, {}}, {}}, {}}, {}}}
end
```

```
test "Prueba 7 de Nora Sandler: Operador unario, negación" do
   token_list = Lexer.scan_word(File.read!("test/negacion.c"), :no_output);
   assert  Parser.parse_token_list(elem(token_list, 1), :no_output) ==
     {:ok, {:program, "program",
            {:function, "main",
             {:return_Reserveword, "return",
              {:negation_Reserveword, "-", {:constant, 5, {}, {}}, {}}, {}}, {}}, {}}, {}}}
 end
```


gremlins-assembler git:(master) ✗ mix test
.....La palabra RETURN es inválida.
.....
Finished in 0.2 seconds
21 tests, 0 failures
Randomized with seed 325289

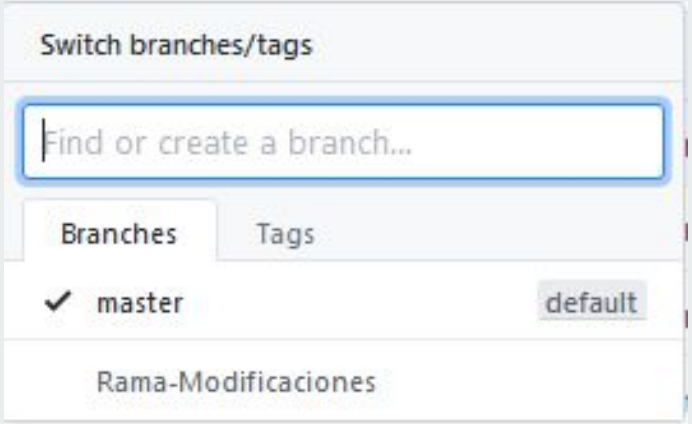
Changes in git

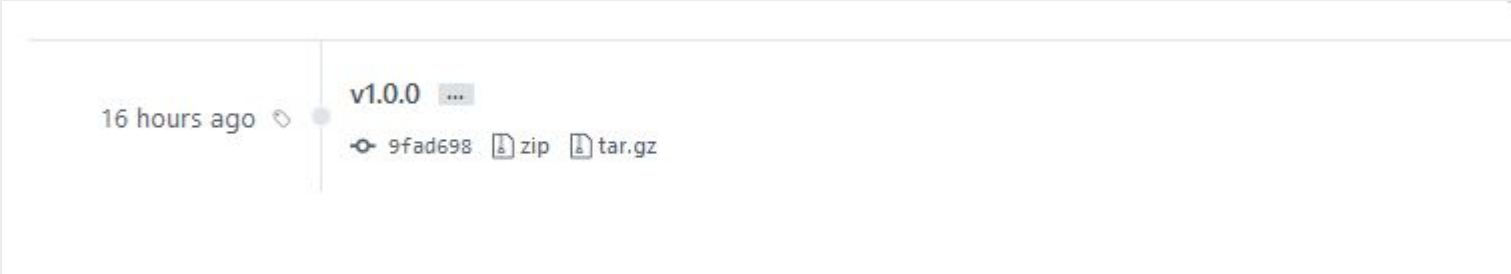
-Update in the master branch.

-Label creation in github called V1.0.0.

-Branch creation for changes and modifications.

Changes in git





Conclusions.

Deliverables and assigned dates for modifications need to be further improved.

Communication with the client must be more efficient and we must show more interest in knowing their points of view.

Improve the technical part for understanding elixir and the other tools used