Stage 2: Unary Operators

Team Assembly

Project Manager

Néstor Martínez Ostoa

Project Manager

- New experiences
- Challenges
- Achievements

Architect

Mario Garrido Czacki

Architect

- New experiences
- Challenges
- Improvements added to the system
- New features added

```
def parse(otl, gast) do
    ps_m = generate_possible_structure_map(gast)
    root_AST = generate_root_ast()
    {result_token, oast, tl, error_cause} = my_structu
    if result_token === :ok and tl === [] do
        {:ok,oast,tl,error_cause}
    else
        if result_token === :error do
            {:token_missing_error,oast,tl,error_cause}
```

Integrator

Enrique Hernández Zamora

Integrator

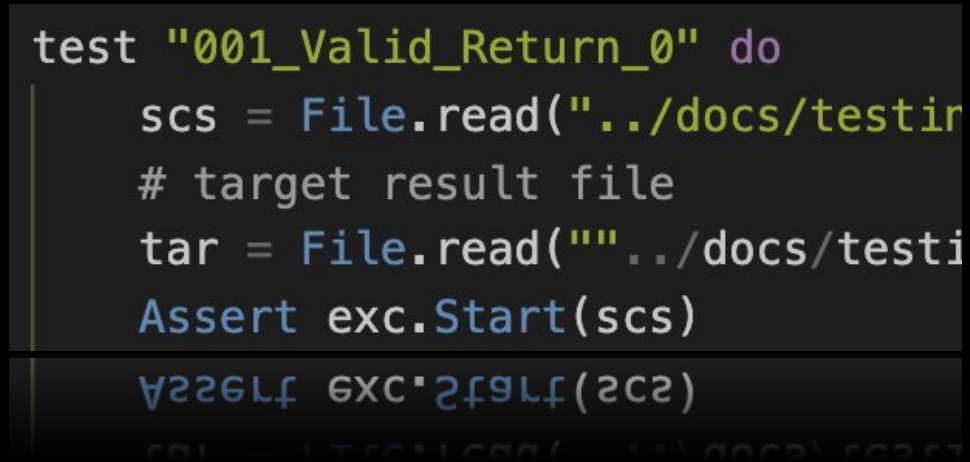
- New experiences
- Challenges
- Integrations
  - Lexer modification
  - Parser modification
  - Code Generator modification
  - Corrections proposed by the teacher

Tester

Alejandro Bondi

Tester

- New experiences
- Test goals
- Challenges
- Risk and mistakes



exc > test > code_generator_test.exs

```elixir
defmodule CodeGeneratorTest do
    use ExUnit.Case
    doctest CodeGenerator

    test "001_S1_Valid_Return0" do
        verbose = false
        c_tokens_path = "./specification_files/c_tokens.xml"
        c_structures_path = "./specification_files/c_structures.xml"
        file_path = "./examples/test.c"

        cg = Reader.read_code_and_tokens(file_path, c_tokens_path, verbose)
        |> Lexer.tokenize()
        |> Filter.filter_lexer_output(file_path, verbose)
        |> Parser.parse(Reader.read_general_ast(c_structures_path))
        |> Filter.filter_parser_output(file_path, verbose)
        |> CodeGenerator.generate_code(verbose)

        assert cg == "    .section        __TEXT,__text,regular,pure_instructions\n    .p2
    end

    test "015_S2_Valid_Negative" do
        verbose = false
        c_tokens_path = "./specification_files/c_tokens.xml"
        c_structures_path = "./specification_files/c_structures.xml"
        file_path = "./examples/stage_2/015_S2_Valid_Negative.c"

        cg = Reader.read_code_and_tokens(file_path, c_tokens_path, verbose)
        |> Lexer.tokenize()
        |> Filter.filter_lexer_output(file_path, verbose)
        |> Parser.parse(Reader.read_general_ast(c_structures_path))
        |> Filter.filter_parser_output(file_path, verbose)
        |> CodeGenerator.generate_code(verbose)

        assert cg == "    .section        __TEXT,__text,regular,pure_instructions\n    .p2
    end
end
```

Test Example

Code Generator

Test Example

Code Generator

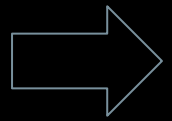
```
alejandro@MacBook-Pro exc % mix compile
alejandro@MacBook-Pro exc % mix test test/code_generator_test.exs
..

Finished in 0.06 seconds
2 tests, 0 failures

Randomized with seed 397809
alejandro@MacBook-Pro exc %
```

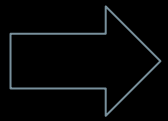
So … what changed ?

Compiler’s name

assembly  exc

Big Code Refactoring

```
def start(file_path \\ "examples/test.c") do
  {scs, gtl} = Reader.load(file_path)
  {otl, status} = Lexer.tokenize({scs, gtl})
  if status == :error do
    Hps.ErrorDetecter.lexer_error(otl, file_path)
    :error
  else
    gast = Reader.load_gast()
    {result_token,oast,tl,error_cause} = Parser.parse(otl, gast)
    if result_token === :ok do
      CodeGenerator.generate_code(oast)
      |> Writer.write_file
      |> Invoker.invoke_gcc
      :ok
    else
      Hps.ErrorDetecter.parser_error(result_token, tl, error_cause, file_path)
      :error
  end
  end
end
```



```
def start_compilation(file_path, verbose)                    do
  Reader.read_code_and_tokens(file_path, @c_tokens_path, verbose)
  |> Lexer.tokenize()
  |> Filter.filter_lexer_output(file_path, verbose)
  |> Parser.parse(Reader.read_general_ast(@c_structures_path))
  |> Filter.filter_parser_output(file_path, verbose)
  |> CodeGenerator.generate_code(verbose)
  |> Writer.write_file()
  |> Invoker.invoke_gcc()
end
```

New Modules

Filter https://hexdocs.pm/exc/Helpers.ASTTraveler.html

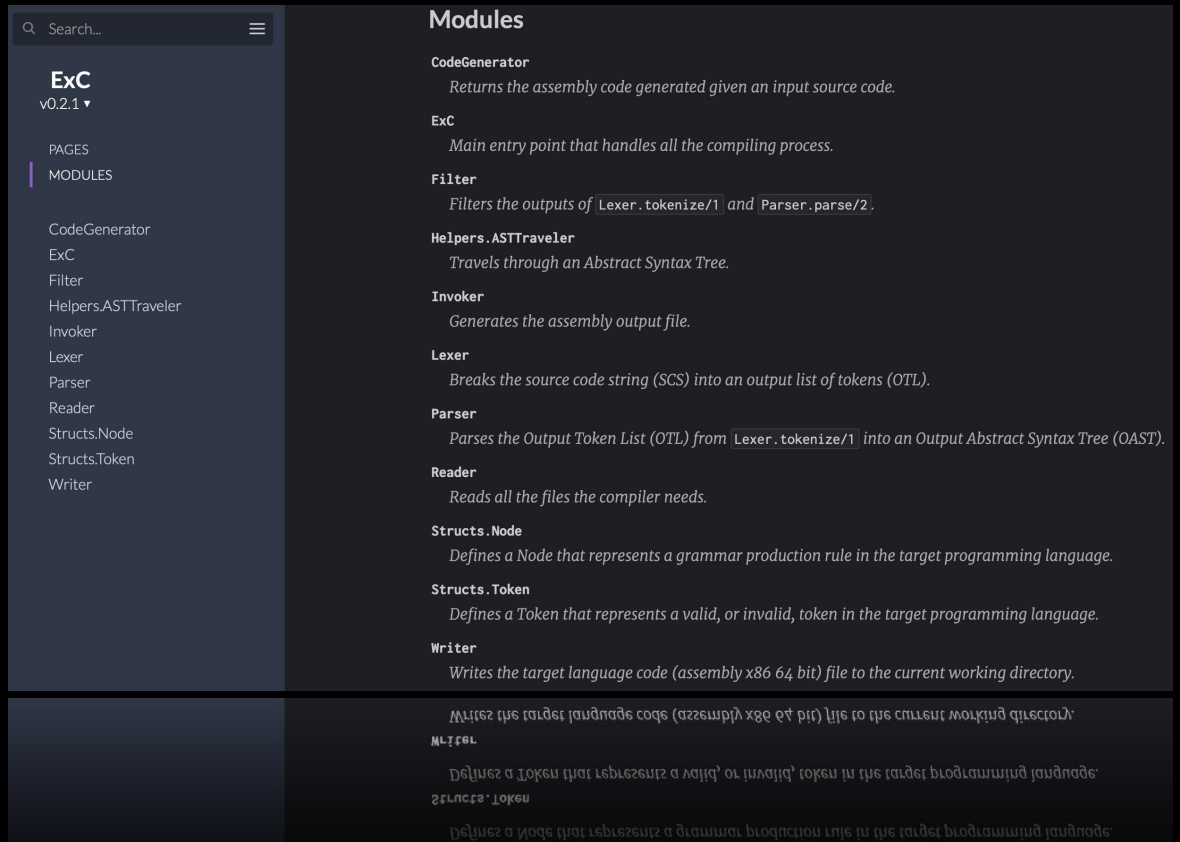
ASTTraveler https://hexdocs.pm/exc/Filter.html

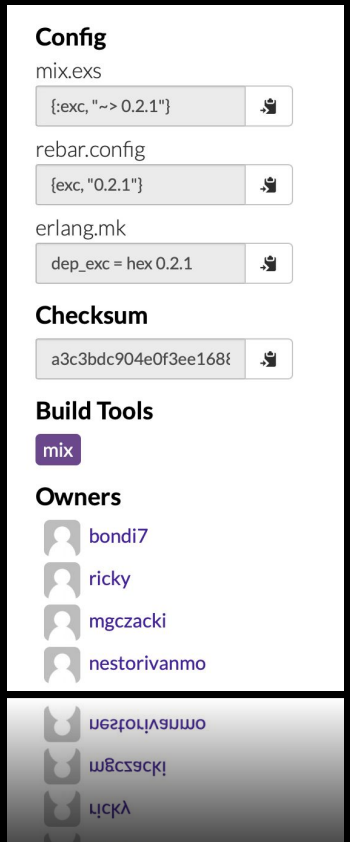
HexDocs

https://hex.pm/packages/exc

https://hexdocs.pm/exc/api-reference.html

HexDocs


Search...
ExC
v0.2.1
PAGES
MODULES
CodeGenerator
ExC
Filter
Helpers.ASTTraveler
Invoker
Lexer
Parser
Reader
Structs.Node
Structs.Token
Writer
Modules
CodeGenerator
Returns the assembly code generated given an input source code.
ExC
Main entry point that handles all the compiling process.
Filter
Filters the outputs of Lexer.tokenize/1 and Parser.parse/2.
Helpers.ASTTraveler
Travels through an Abstract Syntax Tree.
Invoker
Generates the assembly output file.
Lexer
Breaks the source code string (SCS) into an output list of tokens (OTL).
Parser
Parses the Output Token List (OTL) from Lexer.tokenize/1 into an Output Abstract Syntax Tree (OAST).
Reader
Reads all the files the compiler needs.
Structs.Node
Defines a Node that represents a grammar production rule in the target programming language.
Structs.Token
Defines a Token that represents a valid, or invalid, token in the target programming language.
Writer
Writes the target language code (assembly x86 64 bit) file to the current working directory.


Config
mix.exs
{:exc, "~> 0.2.1"}
rebar.config
{exc, "0.2.1"}
erlang.mk
dep_exc = hex 0.2.1
Checksum
a3c3bdc904e0f3ee168
Build Tools
mix
Owners
bondi7
ricky
mgczacki
nestorivanmo

HexDocs

Parser

Parses the Output Token List (OTL) from `Lexer.tokenize/1` into an Output Abstract Syntax Tree (OAST).

Summary

Functions

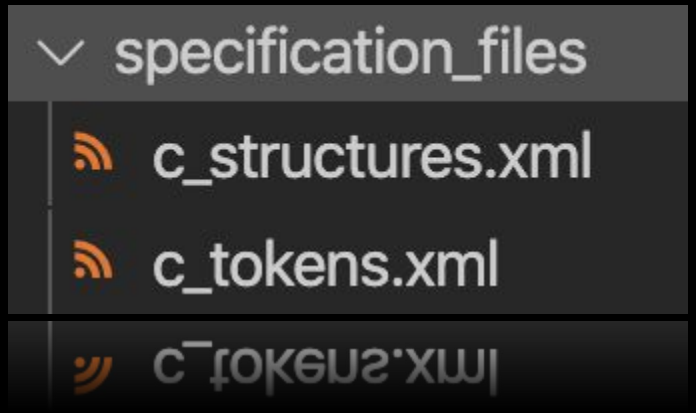
parse(otl, gast)

Returns a tuple containing a status token, the Output Abstract Syntax Tree (OAST), a token list and a possible error cause.

Functions

```
parse(otl, gast)
```

Specification Files



Specification Files

```
<token tag="minus">
    <expression>
        -
    </expression>
</token>
<token tag="negation">
    <expression>
        !
    </expression>
</token>
<token tag="complement">
    <expression>
        ~
    </expression>
</token>
```

```
    <structure tag="negative-operation">
        <token></token>
        <substructure tag="operator">
            <class>negative-operator</class>
        </substructure>
        <substructure tag="evaluation">
            <class>evaluation</class>
        </substructure>
        <class>evaluation</class>
        <asm>
            neg %:1
movl %:1, %:r
```

Code Generator

Before

```
defmodule CodeGenerator do
  def generate_code(oast) do…
  end
  def get_node_with_tag(root, str_tag) do…
  end
end
```

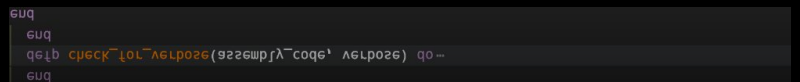
Code Generator

After

```
defmodule CodeGenerator do
  @moduledoc """
  Returns the assembly code generated given an input source code.
  """

  @doc """
  Returns a string of the assembly code generated given a source code.
  ## Specs
  ```abstract_syntax_tree``` : abstract sytnax tree generated by the `Parser.parse/2`.

  ```verbose``` a boolean value indicating if the compiler should output all of its steps.
  """
  def generate_code(abstract_syntax_tree, verbose) do…
  end
  defp generate_raw_string_code(abstract_syntax_tree, incoming_free_context \\ get_available_r
  end
  defp print_next_children([], incoming_siblings_context, incoming_free_context, _sibling_numb
  end
  defp print_next_children(children_list, incoming_siblings_context, incoming_free_context, si
  end
  defp get_available_registers() do…
  end
  defp contextualize_asm(code, incoming_children_context, incoming_free_context, sibling_numbe
  end
  defp fetch_registers([], _) do…
  end
  defp fetch_registers(candidates, free_registers) do…
  end
  defp generate_ram_variables(_number_needed) do…
  end
  defp cleanup(raw_asm_code) do…
  end
  defp assembly(cleaned_asm_code) do …
  end
  defp check_for_verbose(assembly_code, verbose) do…
  end
end
```



Code Generator

Input

```
nestorivanmo@Nestor-MBP exc % cat examples/test.c
int main() {
        return !(~!(-(4)));
}
}
```

Code Generator

Code Generated

```
Raw Code
    .section        __TEXT,__text,regular,pure_instructions
    .p2align        4, 0x90
    .globl _main
  _main:

movl $4, %ebx
movl %ebx, %ecx
neg %ecx
movl %ecx, %ebx
movl %ebx, %ecx
notl %ecx
movl %ecx, %ebx
not %ebx
movl %ebx, %ecx
movl %ecx, %ebx
notl %ebx
movl %ebx, %ecx
movl %ecx, %ebx
movl %ebx, %ecx
movl %ecx, %eax

    ret
```

Compiling Process

git clone https://github.com/hiphoox/c202-assembly.git

```
nestorivanmo@Nestor-MBP Desktop % git clone https://github.com/hiphoox/c202-asse
mbly.git
Cloning into 'c202-assembly'...
remote: Enumerating objects: 464, done.
remote: Counting objects: 100% (464/464), done.
remote: Compressing objects: 100% (259/259), done.
remote: Total 1485 (delta 284), reused 356 (delta 186), pack-reused 1021
Receiving objects: 100% (1485/1485), 5.70 MiB | 4.04 MiB/s, done.
Resolving deltas: 100% (902/902), done.
```

Compiling Process

cd c202-assembly/exc
mix deps.get

```
nestorivanmo@Nestor-MBP Desktop % cd c202-assembly/exc
nestorivanmo@Nestor-MBP exc % mix deps.get
Resolving Hex dependencies...
Dependency resolution completed:
Unchanged:
  earmark 1.4.3
  elixir_xml_to_map 0.1.2
  erlsom 1.5.0
  ex_doc 0.21.3
  makeup 1.0.1
  makeup_elixir 0.14.0
  nimble_parsec 0.5.3
* Getting elixir_xml_to_map (Hex package)
* Getting ex_doc (Hex package)
* Getting earmark (Hex package)
* Getting makeup_elixir (Hex package)
* Getting makeup (Hex package)
* Getting nimble_parsec (Hex package)
* Getting erlsom (Hex package)
```

Compiling Process

mix escript.build

```
nestorivanmo@Nestor-MBP exc % mix escript.build
==> makeup_elixir
Compiling 4 files (.ex)
Generated makeup_elixir app
==> ex_doc
Compiling 20 files (.ex)
Generated ex_doc app
==> exc
Compiling 16 files (.ex)
Generated exc app
Generated escript exc with MIX_ENV=dev
```

Compiling Process

./exc ./examples/test.c -v

```
nestorivanmo@Nestor-MBP exc % ./exc ./examples/test.c -v
```

Compiling Process

Source Code String (SCS)

```
SCS
int main() {  return -23; }
```

Compiling Process

Generic Token List (GTL)

```
GTL
[
  %Structs.Token{expression: "int\\s+", pos_x: nil, pos_y: nil, tag: "int"},
  %Structs.Token{expression: "main", pos_x: nil, pos_y: nil, tag: "main"},
  %Structs.Token{
    expression: "\\(",
    pos_x: nil,
    pos_y: nil,
    tag: "parenthesis-open"
  },
```

Compiling Process

Generic Token List (GTL) : continuation

```
  %Structs.Token{
    expression: "\\)",
    pos_x: nil,
    pos_y: nil,
    tag: "parenthesis-close"
  },
  %Structs.Token{expression: "{", pos_x: nil, pos_y: nil, tag: "bracket-open"},
  %Structs.Token{expression: "}", pos_x: nil, pos_y: nil, tag: "bracket-close"}
  %Structs.Token{
    expression: "return\\s+",
    pos_x: nil,
    pos_y: nil,
    tag: "return"
  },
  %Structs.Token{expression: "[0-9]+", pos_x: nil, pos_y: nil, tag: "literal"},
  %Structs.Token{expression: ";", pos_x: nil, pos_y: nil, tag: "semicolon"},
  %Structs.Token{expression: "-", pos_x: nil, pos_y: nil, tag: "minus"},
  %Structs.Token{expression: "!", pos_x: nil, pos_y: nil, tag: "negation"},
  %Structs.Token{expression: "~", pos_x: nil, pos_y: nil, tag: "complement"},
  %Structs.Token{expression: "\\S+", pos_x: nil, pos_y: nil, tag: "error"}
]
```

Compiling Process

Output Token List (OTL)

```
OTL
[
  %Structs.Token{expression: "int", pos_x: nil, pos_y: nil, tag: "int"},
  %Structs.Token{expression: "main", pos_x: nil, pos_y: nil, tag: "main"},
  %Structs.Token{
    expression: "(",
    pos_x: nil,
    pos_y: nil,
    tag: "parenthesis-open"
  },
```

Compiling Process

Output Token List (OTL): continuation

```
  %Structs.Token{
    expression: ")",
    pos_x: nil,
    pos_y: nil,
    tag: "parenthesis-close"
  },
  %Structs.Token{expression: "{", pos_x: nil, pos_y: nil, tag: "bracket-open"},
  %Structs.Token{expression: "return", pos_x: nil, pos_y: nil, tag: "return"},
  %Structs.Token{expression: "-", pos_x: nil, pos_y: nil, tag: "minus"},
  %Structs.Token{expression: "23", pos_x: nil, pos_y: nil, tag: "literal"},
  %Structs.Token{expression: ";", pos_x: nil, pos_y: nil, tag: "semicolon"},
  %Structs.Token{expression: "}", pos_x: nil, pos_y: nil, tag: "bracket-close"}
]
```

Compiling Process

Output Abstract Syntax Tree (OAST)

```
OAST

        {function => }
                {int-data-type => int}
                {main-function-name => main}
                {evaluator-open => (}
                {evaluator-close => )}
                {section-open => {}
                {operation => }
                        {return-word => return}
                        {negative-operation => }
                                {minus => -}
                                {literal => 23}
                        {semicolon => ;}
                {section-close => }}
```

Compiling Process

Code Generated

```
Raw Code
    .section        __TEXT,__text,regular,pure_instructions
    .p2align        4, 0x90
    .globl _main
  _main:

movl $23, %ebx
neg %ebx
movl %ebx, %ecx
movl %ecx, %ebx
movl %ebx, %ecx
movl %ecx, %eax

    ret
```

Compiling Process

mix test

```
nestorivanmo@Nestor-MBP exc % mix test
............

Finished in 0.1 seconds
12 tests, 0 failures

Randomized with seed 96498
```