

Construção de um compilador de Python para LLVM usando Objective Caml

Tatiane Fernanda de Souza Silva

`tatianefx@gmail.com`

Faculdade de Computação
Universidade Federal de Uberlândia

18 de dezembro de 2018

Lista de Listagens

2.1	Terminal	9
2.2	Terminal	9
2.3	Terminal	9
2.4	Terminal	10
2.5	Terminal	10
3.1	Terminal	11
3.2	Terminal	11
3.3	Terminal	11
3.4	Terminal	12
3.5	nano01.py	12
3.6	nano01.c	12
3.7	nano01.ll	12
3.8	nano02.py	13
3.9	nano02.c	13
3.10	nano02.ll	13
3.11	nano03.py	13
3.12	nano03.c	14
3.13	nano03.ll	14
3.14	nano04.py	14
3.15	nano04.c	15
3.16	nano04.ll	15
3.17	nano05.py	15
3.18	nano05.c	15
3.19	nano05.ll	16
3.20	nano06.py	16
3.21	nano06.c	17
3.22	nano06.ll	17
3.23	nano07.py	18
3.24	nano07.c	18
3.25	nano07.ll	18
3.26	nano08.py	19
3.27	nano08.c	19
3.28	nano08.ll	19
3.29	nano09.py	20
3.30	nano09.c	21
3.31	nano09.ll	21
3.32	nano10.py	22
3.33	nano10.c	22
3.34	nano10.ll	22
3.35	nano11.py	24

3.36	nano11.c	24
3.37	nano11.ll	24
3.38	nano12.py	25
3.39	nano12.c	25
3.40	nano12.ll	26
3.41	micro01.py	27
3.42	micro01.c	28
3.43	micro01.ll	28
3.44	micro02.py	29
3.45	micro02.c	29
3.46	micro02.ll	30
3.47	micro03.py	31
3.48	micro03.c	31
3.49	micro03.ll	32
3.50	micro04.py	33
3.51	micro04.c	33
3.52	micro04.ll	34
3.53	micro05.py	35
3.54	micro05.c	36
3.55	micro05.ll	36
3.56	micro06.py	39
3.57	micro06.c	39
3.58	micro06.ll	39
3.59	micro07.py	41
3.60	micro07.c	42
3.61	micro07.ll	42
3.62	micro08.py	44
3.63	micro08.c	45
3.64	micro08.ll	45
3.65	micro09.py	47
3.66	micro09.c	47
3.67	micro09.ll	47
3.68	micro10.py	50
3.69	micro10.c	50
3.70	micro10.ll	50
3.71	micro11.py	52
3.72	micro11.c	52
3.73	micro11.ll	53
4.1	Terminal	56
4.2	lexico.mll	56
4.3	pre _{processador} .ml	61
4.4	carregador.ml	62
4.5	Terminal	63
4.6	sintatico.mly	64
4.7	ast.ml	67
4.8	Terminal	68
4.9	testeSintatico.txt	68
4.10	semantico.ml	71
4.11	sast.ml	78

4.12	tast.ml	78
4.13	Terminal	79
4.14	testeSemantico.txt	79
4.15	interprete.ml	80
4.16	ambInterp.ml	86
4.17	Terminal	87
4.18	testeInterprete.txt	87
4.19	codigo.ml	88
4.20	codend.ml	88
4.21	codendtest.ml	94
4.22	teste.py	99
4.23	Terminal	99
4.24	gerador.ml	100
4.25	teste.py	104
4.26	Terminal	105

Sumário

1	Introdução	7
1.1	Sistema Operacional e Hardware	7
1.2	OCaml	7
1.3	Low Level Virtual Machine	7
1.4	Python	8
2	Instalações	9
2.1	Homebrew	9
2.2	Instalação do OCaml	9
2.3	Instalação LLVM	9
2.4	Instalação Menhir	10
3	Execução dos programas	11
3.1	Código feito em Python	11
3.2	Clang	11
3.3	LLVM	11
3.4	Nano Programas	12
3.4.1	Estrutura básica	12
3.4.2	Declaração de variável	13
3.4.3	Atribuição de valor a variável	13
3.4.4	Soma de valores inteiros	14
3.4.5	Impressão de inteiro	15
3.4.6	Subtração e impressão do resultado	16
3.4.7	Estrutura condicional e impressão	18
3.4.8	Estrutura condicional e impressão	19
3.4.9	Adição com divisão, estrutura condicional e impressão	20
3.4.10	Condição de igualdade	22
3.4.11	Estrutura de repetição "while" com impressão	24
3.4.12	Estruturas de repetição e condicional com impressão de valores	25
3.5	Micro Programas	27
3.5.1	Converte graus Celsius para Fahrenheit	27
3.5.2	Ler dois inteiros e decide qual é maior	29
3.5.3	Lê um número e verifica se ele está entre 100 e 200	31
3.5.4	Lê números e informa quais estão entre 10 e 150	33
3.5.5	Lê strings e caracteres	35
3.5.6	Escreve um número lido por extenso	39
3.5.7	Decide se os números são positivos, zeros ou negativos	41
3.5.8	Decide se um número é maior ou menor que 10	44
3.5.9	Cálculo de preços	47

3.5.10	Calcula o fatorial de um número	50
3.5.11	Decide se um número é positivo, zero ou negativo com auxílio de uma função	52
4	Compilador	56
4.1	Analizador Léxico	56
4.1.1	Teste	63
4.2	Analizador Sintático	64
4.2.1	Teste	68
4.3	Analizador Semântico	71
4.3.1	Testes	79
4.4	Interprete	80
4.4.1	Teste	87
4.5	Código de 3 Endereços	88
4.5.1	Teste	99
4.6	Gerador	100
4.6.1	Teste	104
4.7	Trabalhos Futuros	106

Capítulo 1

Introdução

Este documento foi escrito como relatório do desenvolvimento de um compilador da linguagem Python que utiliza como assembler a máquina virtual LLVM (Low Level Virtual Machine) para gerar códigos binários executáveis.

O intuito dessas tarefas é aumentar a familiaridade com a linguagem Python, melhorar o entendimento do que é e como é um código gerado para a máquina virtual LLVM e também de como utilizá-la.

1.1 Sistema Operacional e Hardware

O Sistema Operacional (SO) utilizado para realização dos experimentos será o macOS High Sierra 10.13.6. O computador usado para execução das tarefas é um MacBook Air (13-inch, 2017) com processador 1,8 GHz Intel Core i5, 8 GB 1600 MHz DDR3 de memória RAM e 128 GB de espaço em disco SSD.

1.2 OCaml

OCaml é uma linguagem de programação de foco industrial e propósito geral, com ênfase em expressividade e segurança. É a tecnologia escolhida em empresas onde um único erro pode custar milhões. Possui uma comunidade ativa que desenvolveu um rico conjunto de bibliotecas. É também uma linguagem de ensino amplamente usada.

1.3 Low Level Virtual Machine

A *Low Level Virtual Machine* (LLVM) é uma infraestrutura de compilador escrita em C++, desenvolvida para otimizar em tempos de compilação, ligação e execução de programas escritos em linguagens de programação variadas. Implementada originalmente para C e C++, sua arquitetura permitiu a expansão para outras linguagens posteriormente, incluindo

Objective-C, Fortran, Ada, Haskell, bytecode Java, Python, Ruby, ActionScript, GLSL, Julia, entre outras.

1.4 Python

Python é uma linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por Guido van Rossum em 1991. Atualmente possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos Python Software Foundation. Apesar de várias partes da linguagem possuírem padrões e especificações formais, a linguagem como um todo não é formalmente especificada. O padrão de fato é a implementação CPython.

Capítulo 2

Instalações

2.1 Homebrew

O Homebrew auxilia na instalação de outras dependências. Para instalá-lo digitamos o seguinte comando no terminal:

Listagem 2.1: Terminal

```
1 $ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install  
/master/install)"
```

2.2 Instalação do OCaml

Para efetuar a instalação o OCaml, basta digitar os seguintes comandos no terminal:

Listagem 2.2: Terminal

```
1 $ brew install ocaml  
2 $ brew install opam  
3 $ eval $(opam env)
```

Para verificar se tudo ocorreu bem com a instalação, basta digitar o comando a seguir e verificar sua saída.

Listagem 2.3: Terminal

```
1 $ ocaml -version  
2 The OCaml toplevel, version 4.07.0
```

2.3 Instalação LLVM

Para a instalação da plataforma LLVM, basta digitar no terminal o seguinte comando:

brewinstall --with --toolchainllvm

Além da plataforma LLVM, é necessária a instalação do CLANG que irá realizar a conversão do código da linguagem C para o assembly do LLVM. Para a instalação do CLANG, basta digitar no terminal o seguinte comando:

Listagem 2.4: Terminal

```
1 $(brew --prefix llvm)/bin  
2 $ echo 'export PATH="/usr/local/opt/llvm/bin:$PATH"' >> ~/.bash_profile
```

2.4 Instalação Menhir

Para a instalação do Menhir, basta digitar no terminal o seguinte comando:

Listagem 2.5: Terminal

```
1 $ opam install menhir  
2 $ brew install menhir
```

Capítulo 3

Execução dos programas

Depois de tudo instalado é possível executar os programas. A seguir temos os programas em Python seguidos por sua tradução para assembly LLVM.

3.1 Código feito em Python

Para executar um código feito em Python, basta executar o seguinte comando:

Listagem 3.1: Terminal

```
1 $ python file.py
2 $ python
3 > import file
```

3.2 Clang

Para compilar um arquivo .c para .ll, basta executar o seguinte comando:

Listagem 3.2: Terminal

```
1 $ clang -S -emit-llvm file.c -o file.ll
```

3.3 LLVM

Para compilar um arquivo .ll utilizando o llvm e executá-lo, basta executar os seguintes comandos:

Listagem 3.3: Terminal

```
1 $ llvm-as file.ll
2 $ lli file.bc
```

Para descompilar arquivo binário para geração de código em assembly:

Listagem 3.4: Terminal

```
1 $ llvm-dis file.bc
```

3.4 Nano Programas

Nesta seção teremos todos os nano programas escritos em Pascal, Java e Jasmin.

3.4.1 Estrutura básica

Listagem 3.5: nano01.py

```
1 def nano01():
2     return
3
4 nano01()
```

Listagem 3.6: nano01.c

```
1 #import<stdio.h>
2
3 int main(){
4     return 0;
5 }
```

Listagem 3.7: nano01.ll

```
1 ; ModuleID = 'nano01.c'
2 source_filename = "nano01.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 ; Function Attrs: noinline nounwind optnone uwtable
7 define i32 @main() #0 {
8     %1 = alloca i32, align 4
9     store i32 0, i32* %1, align 4
10    ret i32 0
11 }
12
13 attributes #0 = { noinline nounwind optnone uwtable "correctly-rounded-
    divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-
    -fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-elim-
    non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-nans-
    fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-math"="
    false" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-
    features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-
    soft-float"="false" }
```

```
14
15 !llvm.module.flags = !{!0}
16 !llvm.ident = !{!1}
17
```

```

18 !0 = !{i32 1, !"wchar_size", i32 4}
19 !1 = !{i32 7, !"PIC Level", i32 2}
20 !2 = !{"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.4.2 Declaração de variável

Listagem 3.8: nano02.py

```

1 def nano02():
2     n = (int)
3
4 nano02()

```

Listagem 3.9: nano02.c

```

1 #import<stdio.h>
2
3 int main(){
4     int n;
5     return 0;
6 }

```

Listagem 3.10: nano02.ll

```

1 ; ModuleID = 'nano02.c'
2 source_filename = "nano02.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 ; Function Attrs: noinline nounwind optnone uwtable
7 define i32 @main() #0 {
8     %1 = alloca i32, align 4
9     %2 = alloca i32, align 4
10    store i32 0, i32* %1, align 4
11    ret i32 0
12 }
13
14 attributes #0 = { noinline nounwind optnone uwtable "correctly-rounded-
    divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-
    fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-elim-
    non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-nans-
    fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-math"="
    false" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-
    features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-
    soft-float"="false" }
15
16 !llvm.module.flags = !{!0}
17 !llvm.ident = !{!1}
18
19 !0 = !{i32 1, !"wchar_size", i32 4}
20 !1 = !{i32 7, !"PIC Level", i32 2}
21 !2 = !{"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.4.3 Atribuição de valor a variável

Listagem 3.11: nano03.py

```

1 def nano03():
2     n = (int)
3     n = 1
4
5 nano03()

```

Listagem 3.12: nano03.c

```

1 #import<stdio.h>
2
3 int main(){
4     int n;
5     n = 1;
6     return 0;
7 }

```

Listagem 3.13: nano03.ll

```

1 ; ModuleID = 'nano03.c'
2 source_filename = "nano03.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 ; Function Attrs: noinline nounwind optnone uwtable
7 define i32 @main() #0 {
8     %1 = alloca i32, align 4
9     %2 = alloca i32, align 4
10    store i32 0, i32* %1, align 4
11    store i32 1, i32* %2, align 4
12    ret i32 0
13 }
14
15 attributes #0 = { noinline nounwind optnone uwtable "correctly-rounded-
    divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-
    -fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-elim-
    non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-nans-
    fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-math"="
    false" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-
    features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-
    soft-float"="false" }
16
17 !llvm.module.flags = !{!0}
18 !llvm.ident = !{!1}
19
20 !0 = !{i32 1, !"wchar_size", i32 4}
21 !1 = !{i32 7, !"PIC Level", i32 2}
22 !2 = !{!"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.4.4 Soma de valores inteiros

Listagem 3.14: nano04.py

```

1 def nano04():
2     n = (int)
3     n = 1 + 2
4
5 nano04()

```

Listagem 3.15: nano04.c

```

1 #import<stdio.h>
2
3 int main() {
4     int n;
5     n = 1 + 2;
6     return 0;
7 }

```

Listagem 3.16: nano04.ll

```

1 ; ModuleID = 'nano04.c'
2 source_filename = "nano04.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 ; Function Attrs: noinline nounwind optnone uwtable
7 define i32 @main() #0 {
8     %1 = alloca i32, align 4
9     %2 = alloca i32, align 4
10    store i32 0, i32* %1, align 4
11    store i32 3, i32* %2, align 4
12    ret i32 0
13 }
14
15 attributes #0 = { noinline nounwind optnone uwtable "correctly-rounded-
    divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-
    fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-elim-
    non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-nans-
    fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-math"="
    false" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-
    features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-
    soft-float"="false" }
16
17 !llvm.module.flags = !{!0}
18 !llvm.ident = !{!1}
19
20 !0 = !{i32 1, !"wchar_size", i32 4}
21 !1 = !{i32 7, !"PIC Level", i32 2}
22 !2 = !{!"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.4.5 Impressão de inteiro

Listagem 3.17: nano05.py

```

1 def nano05():
2     n = 2
3     print(n)
4
5 nano05()

```

Listagem 3.18: nano05.c

```

1 #import<stdio.h>
2
3 int main() {
4     int n;

```

```

5  n = 2;
6  printf("%d", n);
7  return 0;
8  }

```

Listagem 3.19: nano05.ll

```

1 ; ModuleID = 'nano05.c'
2 source_filename = "nano05.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
7
8 ; Function Attrs: noinline nounwind optnone uwtable
9 define i32 @main() #0 {
10     %1 = alloca i32, align 4
11     %2 = alloca i32, align 4
12     store i32 0, i32* %1, align 4
13     store i32 2, i32* %2, align 4
14     %3 = load i32, i32* %2, align 4
15     %4 = call i32 @printf(i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8],
16         [3 x i8]* @.str, i32 0, i32 0), i32 %3)
17     ret i32 0
18 }
19 declare i32 @printf(i8*, ...) #1
20
21 attributes #0 = { noinline nounwind optnone uwtable "correctly-rounded-
    divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-
    -fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-elim-
    non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-nans-
    fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-math"="
    false" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-
    features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-
    soft-float"="false" }
22 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
    -tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
    elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
    "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
    trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
    x86-64" "target-features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"
    ="false" "use-soft-float"="false" }
23
24 !llvm.module.flags = !{!0}
25 !llvm.ident = !{!1}
26
27 !0 = !{i32 1, !"wchar_size", i32 4}
28 !1 = !{i32 7, !"PIC Level", i32 2}
29 !2 = !{!"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.4.6 Subtração e impressão do resultado

Listagem 3.20: nano06.py

```

1 def nano06():
2     n = 1 - 2
3     print(n)

```



```

4
5 nano06()

```

Listagem 3.21: nano06.c

```

1 #import<stdio.h>
2
3 int main() {
4     int n;
5     n = 1 - 2;
6     printf("%d", n);
7     return 0;
8 }

```

Listagem 3.22: nano06.ll

```

1 ; ModuleID = 'nano06.c'
2 source_filename = "nano06.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
7
8 ; Function Attrs: noinline nounwind optnone uwtable
9 define i32 @main() #0 {
10     %1 = alloca i32, align 4
11     %2 = alloca i32, align 4
12     store i32 0, i32* %1, align 4
13     store i32 -1, i32* %2, align 4
14     %3 = load i32, i32* %2, align 4
15     %4 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([3 x i8],
        [3 x i8]* @.str, i32 0, i32 0), i32 %3)
16     ret i32 0
17 }
18
19 declare i32 @printf(i8*, ...) #1
20
21 attributes #0 = { noinline nounwind optnone uwtable "correctly-rounded-
    divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-
    fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-elim-
    non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-nans-
    fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-math"="
    false" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-
    features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-
    soft-float"="false" }
22 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
    tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
    elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
    "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
    trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
    x86-64" "target-features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"
    ="false" "use-soft-float"="false" }
23
24 !llvm.module.flags = !{!0}
25 !llvm.ident = !{!1}
26
27 !0 = !{i32 1, !"wchar_size", i32 4}
28 !1 = !{i32 7, !"PIC Level", i32 2}
29 !2 = !{!"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.4.7 Estrutura condicional e impressão

Listagem 3.23: nano07.py

```

1 def nano07():
2     n = 1
3     if n == 1:
4         print(n)
5
6 nano07()

```

Listagem 3.24: nano07.c

```

1 #import<stdio.h>
2
3 int main(){
4     int n;
5     n = 1;
6     if (n == 1) {
7         printf("%d", n);
8     }
9     return 0;
10 }

```

Listagem 3.25: nano07.ll

```

1 ; ModuleID = 'nano07.c'
2 source_filename = "nano07.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
7
8 ; Function Attrs: noinline nounwind optnone uwtable
9 define i32 @main() #0 {
10     %1 = alloca i32, align 4
11     %2 = alloca i32, align 4
12     store i32 0, i32* %1, align 4
13     store i32 1, i32* %2, align 4
14     %3 = load i32, i32* %2, align 4
15     %4 = icmp eq i32 %3, 1
16     br i1 %4, label %5, label %8
17
18 ; <label>:5:                                ; preds = %0
19     %6 = load i32, i32* %2, align 4
20     %7 = call i32 @__printf(i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8],
        [3 x i8]* @.str, i32 0, i32 0), i32 %6)
21     br label %8
22
23 ; <label>:8:                                ; preds = %5, %0
24     ret i32 0
25 }
26
27 declare i32 @__printf(i8*, ...) #1
28
29 attributes #0 = { noinline nounwind optnone uwtable "correctly-rounded-
    divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-
    -fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-elim-
    non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-nans-

```

```

fp-math="false" "no-signed-zeros-fp-math"="false" "no-trapping-math"="
false" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-
features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-
soft-float"="false" }
30 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable
-tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
"no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
x86-64" "target-features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"
="false" "use-soft-float"="false" }
31
32 !llvm.module.flags = !{!0}
33 !llvm.ident = !{!1}
34
35 !0 = !{i32 1, !"wchar_size", i32 4}
36 !1 = !{i32 7, !"PIC Level", i32 2}
37 !2 = !{!"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.4.8 Estrutura condicional e impressão

Listagem 3.26: nano08.py

```

1 def nano08():
2     n = 1
3     if n == 1:
4         print(n)
5     else:
6         print(0)
7
8 nano08()

```

Listagem 3.27: nano08.c

```

1 #import<stdio.h>
2
3 int main(){
4     int n;
5     n = 1;
6     if (n == 1) {
7         printf("%d", n);
8     } else {
9         printf("0");
10    }
11    return 0;
12 }

```

Listagem 3.28: nano08.ll

```

1 ; ModuleID = 'nano08.c'
2 source_filename = "nano08.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
7 @.str.1 = private unnamed_addr constant [2 x i8] c"0\00", align 1
8

```

```

 9 ; Function Attrs: noinline nounwind optnone uwtable
10 define i32 @main() #0 {
11     %1 = alloca i32, align 4
12     %2 = alloca i32, align 4
13     store i32 0, i32* %1, align 4
14     store i32 1, i32* %2, align 4
15     %3 = load i32, i32* %2, align 4
16     %4 = icmp eq i32 %3, 1
17     br i1 %4, label %5, label %8
18
19 ; <label>:5:                                     ; preds = %0
20     %6 = load i32, i32* %2, align 4
21     %7 = call i32 (@i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8],
22     [3 x i8]* @.str, i32 0, i32 0), i32 %6)
23     br label %10
24
25 ; <label>:8:                                     ; preds = %0
26     %9 = call i32 (@i8*, ...) @printf(i8* getelementptr inbounds ([2 x i8],
27     [2 x i8]* @.str.1, i32 0, i32 0))
28     br label %10
29
30 ; <label>:10:                                    ; preds = %8, %5
31     ret i32 0
32 }
33
34 declare i32 @printf(i8*, ...) #1
35
36 attributes #0 = { noinline nounwind optnone uwtable "correctly-rounded-
37     divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-
38     -fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-elim-
39     non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-nans-
40     fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-math"="
41     false" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-
42     features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-
43     soft-float"="false" }
44
45 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
46     -tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
47     elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
48     "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
49     trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
50     x86-64" "target-features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"
51     ="false" "use-soft-float"="false" }
52
53 !0 = !{i32 1, !"wchar_size", i32 4}
54 !1 = !{i32 7, !"PIC Level", i32 2}
55 !2 = !{"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.4.9 Adição com divisão, estrutura condicional e impressão

Listagem 3.29: nano09.py

```

1 def nano09():
2     n = 1 + (1 / 2)
3     if n == 1:
4         print(n)

```

3.4

```
5     else:
6         print(0)
7
8 nano09()
```

Listagem 3.30: nano09.c

```
1 #import<stdio.h>
2
3 int main(){
4     int n;
5     n = 1 + (1 / 2);
6     if (n == 1) {
7         printf("%d", n);
8     } else {
9         printf("0");
10    }
11    return 0;
12 }
```

Listagem 3.31: nano09.ll

```
1 ; ModuleID = 'nano09.c'
2 source_filename = "nano09.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
7 @.str.1 = private unnamed_addr constant [2 x i8] c"0\00", align 1
8
9 ; Function Attrs: noinline nounwind optnone uwtable
10 define i32 @main() #0 {
11     %1 = alloca i32, align 4
12     %2 = alloca i32, align 4
13     store i32 0, i32* %1, align 4
14     store i32 1, i32* %2, align 4
15     %3 = load i32, i32* %2, align 4
16     %4 = icmp eq i32 %3, 1
17     br i1 %4, label %5, label %8
18
19 ; <label>:5:                                ; preds = %0
20     %6 = load i32, i32* %2, align 4
21     %7 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([3 x i8],
22     [3 x i8]* @.str, i32 0, i32 0), i32 %6)
23     br label %10
24
25 ; <label>:8:                                ; preds = %0
26     %9 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([2 x i8],
27     [2 x i8]* @.str.1, i32 0, i32 0))
28     br label %10
29
30 ; <label>:10:                               ; preds = %8, %5
31     ret i32 0
32 }
33
34 declare i32 @printf(i8*, ...) #1
35
36 attributes #0 = { noinline nounwind optnone uwtable "correctly-rounded-
37     divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise
```

```

    -fpmad="false" "no-frame-pointer-elim"="true" "no-frame-pointer-elim-
    non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-nans-
    fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-math"="
    false" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-
    features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-
    soft-float"="false" }
35 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable
    -tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
    elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
    "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
    trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
    x86-64" "target-features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"
    ="false" "use-soft-float"="false" }
36
37 !llvm.module.flags = !{!0}
38 !llvm.ident = !{!1}
39
40 !0 = !{i32 1, !"wchar_size", i32 4}
41 !1 = !{i32 7, !"PIC Level", i32 2}
42 !2 = !{"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.4.10 Condição de igualdade

Listagem 3.32: nano10.py

```

1 def nano10():
2     n = 1
3     m = 2
4     if n == m:
5         print(n)
6     else:
7         print(0)
8
9 nano10()

```

Listagem 3.33: nano10.c

```

1 #import<stdio.h>
2
3 int main(){
4     int n, m;
5     n = 1;
6     m = 2;
7     if (n == m) {
8         printf("%d", n);
9     } else {
10        printf("0");
11    }
12    return 0;
13 }

```

Listagem 3.34: nano10.ll

```

1 ; ModuleID = 'nano10.c'
2 source_filename = "nano10.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"

```

3.4

```

5
6 @.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
7 @.str.1 = private unnamed_addr constant [2 x i8] c"0\00", align 1
8
9 ; Function Attrs: noline nounwind optnone uwtable
10 define i32 @main() #0 {
11     %1 = alloca i32, align 4
12     %2 = alloca i32, align 4
13     %3 = alloca i32, align 4
14     store i32 0, i32* %1, align 4
15     store i32 1, i32* %2, align 4
16     store i32 2, i32* %3, align 4
17     %4 = load i32, i32* %2, align 4
18     %5 = load i32, i32* %3, align 4
19     %6 = icmp eq i32 %4, %5
20     br i1 %6, label %7, label %10
21
22 ; <label>:7:                                ; preds = %0
23     %8 = load i32, i32* %2, align 4
24     %9 = call i32 @printf(i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8],
25         [3 x i8]* @.str, i32 0, i32 0), i32 %8)
26     br label %12
27
28 ; <label>:10:                                ; preds = %0
29     %11 = call i32 @printf(i8*, ...) @printf(i8* getelementptr inbounds ([2 x i8],
30         [2 x i8]* @.str.1, i32 0, i32 0))
31     br label %12
32
33 ; <label>:12:                                ; preds = %10, %7
34     ret i32 0
35 }
36
37 declare i32 @printf(i8*, ...) #1
38
39 attributes #0 = { noline nounwind optnone uwtable "correctly-rounded-
40     divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-
41     fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-elim-
42     non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-nans-
43     fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-math"="
44     false" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-
45     features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-
46     soft-float"="false" }
47
48 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
49     tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
50     elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
51     "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
52     trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
53     x86-64" "target-features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"
54     ="false" "use-soft-float"="false" }
55
56 !0 = !{i32 1, !"wchar_size", i32 4}
57 !1 = !{i32 7, !"PIC Level", i32 2}
58 !2 = !{"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.4.11 Estrutura de repetição "while" com impressão

Listagem 3.35: nano11.py

```

1 def nano11():
2     n = 1
3     m = 2
4     x = 5
5     while x > n:
6         n = n + m
7         print(n)
8
9 nano11()

```

Listagem 3.36: nano11.c

```

1 #import<stdio.h>
2
3 int main(){
4     int n, m, x;
5     n = 1;
6     m = 2;
7     x = 5;
8
9     while (x > n) {
10         n = n + m;
11         printf("%d", n);
12     }
13     return 0;
14 }

```

Listagem 3.37: nano11.ll

```

1 ; ModuleID = 'nano11.c'
2 source_filename = "nano11.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
7
8 ; Function Attrs: noinline nounwind optnone uwtable
9 define i32 @main() #0 {
10     %1 = alloca i32, align 4
11     %2 = alloca i32, align 4
12     %3 = alloca i32, align 4
13     %4 = alloca i32, align 4
14     store i32 0, i32* %1, align 4
15     store i32 1, i32* %2, align 4
16     store i32 2, i32* %3, align 4
17     store i32 5, i32* %4, align 4
18     br label %5
19
20 ; <label>:5:                                     ; preds = %9, %0
21     %6 = load i32, i32* %4, align 4
22     %7 = load i32, i32* %2, align 4
23     %8 = icmp sgt i32 %6, %7
24     br i1 %8, label %9, label %15
25
26 ; <label>:9:                                     ; preds = %5

```



```

27 %10 = load i32, i32* %2, align 4
28 %11 = load i32, i32* %3, align 4
29 %12 = add nsw i32 %10, %11
30 store i32 %12, i32* %2, align 4
31 %13 = load i32, i32* %2, align 4
32 %14 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([3 x i8],
    [3 x i8]* @.str, i32 0, i32 0), i32 %13)
33 br label %5
34
35 ; <label>:15:                                ; preds = %5
36 ret i32 0
37 }
38
39 declare i32 @printf(i8*, ...) #1
40
41 attributes #0 = { noinline nounwind optnone uwtable "correctly-rounded-
    divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-
    -fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-elim-
    non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-nans-
    fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-math"="
    false" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-
    features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-
    soft-float"="false" }
42 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
    -tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
    elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
    "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
    trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
    x86-64" "target-features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"
    ="false" "use-soft-float"="false" }
43
44 !llvm.module.flags = !{!0}
45 !llvm.ident = !{!1}
46
47 !0 = !{i32 1, !"wchar_size", i32 4}
48 !1 = !{i32 7, !"PIC Level", i32 2}
49 !2 = !{!"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.4.12 Estruturas de repetição e condicional com impressão de valores

Listagem 3.38: nano12.py

```

1 def nano12():
2     n = 1
3     m = 2
4     x = 5
5     while x > n:
6         if n == m:
7             print(n)
8         else:
9             print(0)
10            x = x - 1
11
12 nano12()

```

Listagem 3.39: nano12.c

```

1 #import<stdio.h>
2
3 int main(){
4     int n, m, x;
5     n = 1;
6     m = 2;
7     x = 5;
8
9     while (x > n) {
10         if (n == m) {
11             printf("%d", n);
12         } else {
13             printf("0");
14             x = x - 1;
15         }
16     }
17     return 0;
18 }

```

Listagem 3.40: nano12.ll

```

1 ; ModuleID = 'nano12.c'
2 source_filename = "nano12.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
7 @.str.1 = private unnamed_addr constant [2 x i8] c"0\00", align 1
8
9 ; Function Attrs: noinline nounwind optnone uwtable
10 define i32 @main() #0 {
11     %1 = alloca i32, align 4
12     %2 = alloca i32, align 4
13     %3 = alloca i32, align 4
14     %4 = alloca i32, align 4
15     store i32 0, i32* %1, align 4
16     store i32 1, i32* %2, align 4
17     store i32 2, i32* %3, align 4
18     store i32 5, i32* %4, align 4
19     br label %5
20
21 ; <label>:5:                                     ; preds = %20, %0
22     %6 = load i32, i32* %4, align 4
23     %7 = load i32, i32* %2, align 4
24     %8 = icmp sgt i32 %6, %7
25     br i1 %8, label %9, label %21
26
27 ; <label>:9:                                     ; preds = %5
28     %10 = load i32, i32* %2, align 4
29     %11 = load i32, i32* %3, align 4
30     %12 = icmp eq i32 %10, %11
31     br i1 %12, label %13, label %16
32
33 ; <label>:13:                                    ; preds = %9
34     %14 = load i32, i32* %2, align 4
35     %15 = call i32 @i8*, ... @printf(i8* getelementptr @inbounds ([3 x i8],
36                                     [3 x i8]* @.str, i32 0, i32 0), i32 %14)
37     br label %20

```

```

38 ; <label>:16:                                ; preds = %9
39 %17 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([2 x i8],
    [2 x i8]* @.str.1, i32 0, i32 0))
40 %18 = load i32, i32* %4, align 4
41 %19 = sub nsw i32 %18, 1
42 store i32 %19, i32* %4, align 4
43 br label %20
44
45 ; <label>:20:                                ; preds = %16, %13
46 br label %5
47
48 ; <label>:21:                                ; preds = %5
49 ret i32 0
50 }
51
52 declare i32 @printf(i8*, ...) #1
53
54 attributes #0 = { noinline nounwind optnone uwtable "correctly-rounded-
    divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-
    -fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-elim-
    non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-nans-
    fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-math"="
    false" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-
    features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-
    soft-float"="false" }
55 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
    -tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
    elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
    "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
    trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
    x86-64" "target-features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"
    ="false" "use-soft-float"="false" }
56
57 !llvm.module.flags = !{!0}
58 !llvm.ident = !{!1}
59
60 !0 = !{i32 1, !"wchar_size", i32 4}
61 !1 = !{i32 7, !"PIC Level", i32 2}
62 !2 = !{!"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.5 Micro Programas

Nesta seção teremos todos os micro programas escritos em Pascal e Jasmin.

3.5.1 Converte graus Celsius para Fahrenheit

Listagem 3.41: micro01.py

```

1 def micro01():
2     cel , far = 0.0 , 0.0
3     print(" Tabela de conversao: Celsius -> Fahrenheit")
4     print("Digite a temperatura em Celsius: ")
5     cel = input()
6     far = (9 * cel + 160) / 5
7     print("A nova temperatura e:" + str(far) + "F")

```

```

8
9 micro01()

```

Listagem 3.42: micro01.c

```

1 #import<stdio.h>
2
3 int main(){
4     float cel, far;
5     cel = 0.0;
6     far = 0.0;
7     printf("Tabela de conversao: Celsius -> Fahrenheit");
8
9     printf("Digite a temperatura em Celsius: ");
10    scanf("%f", &cel);
11
12    far = (9.0 * cel + 160.0) / 5.0;
13    printf("%f", far);
14
15    return 0;
16 }

```

Listagem 3.43: micro01.ll

```

1 ; ModuleID = 'micro01.c'
2 source_filename = "micro01.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [43 x i8] c"Tabela de conversao:
7     Celsius -> Fahrenheit\00", align 1
8 @.str.1 = private unnamed_addr constant [34 x i8] c"Digite a temperatura
9     em Celsius: \00", align 1
10 @.str.2 = private unnamed_addr constant [3 x i8] c"%f\00", align 1
11
12 ; Function Attrs: noinline nounwind optnone ssp uwtable
13 define i32 @main() #0 {
14     %1 = alloca i32, align 4
15     %2 = alloca float, align 4
16     %3 = alloca float, align 4
17     store i32 0, i32* %1, align 4
18     store float 0.000000e+00, float* %2, align 4
19     store float 0.000000e+00, float* %3, align 4
20     %4 = call i32 @__printf(i8*, ...) @printf(i8* getelementptr inbounds ([43 x i8],
21         [43 x i8]* @.str, i32 0, i32 0))
22     %5 = call i32 @__printf(i8*, ...) @printf(i8* getelementptr inbounds ([34 x i8],
23         [34 x i8]* @.str.1, i32 0, i32 0))
24     %6 = call i32 @__scanf(i8* getelementptr inbounds ([3 x i8], [3
25         x i8]* @.str.2, i32 0, i32 0), float* %2)
26     %7 = load float, float* %2, align 4
27     %8 = fpext float %7 to double
28     %9 = fmul double 9.000000e+00, %8
29     %10 = fadd double %9, 1.600000e+02
30     %11 = fdiv double %10, 5.000000e+00
31     %12 = fptrunc double %11 to float
32     store float %12, float* %3, align 4
33     %13 = load float, float* %3, align 4
34     %14 = fpext float %13 to double

```

3.5

```
30  %15 = call i32 @i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8],
    [3 x i8]* @.str.2, i32 0, i32 0), double %14)
31  ret i32 0
32 }
33
34 declare i32 @printf(i8*, ...) #1
35
36 declare i32 @scanf(i8*, ...) #1
37
38 attributes #0 = { noinline nounwind optnone ssp uwtable "correctly-rounded
    -divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-
    precise-fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-
    elim-non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-
    nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-
    math"="false" "stack-protector-buffer-size"="8" "target-cpu"="penryn" "
    target-features"="+cxl6,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+ssse3,+x87
    " "unsafe-fp-math"="false" "use-soft-float"="false" }
39 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable
    -tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
    elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
    "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
    trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
    penryn" "target-features"="+cxl6,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+
    ssse3,+x87" "unsafe-fp-math"="false" "use-soft-float"="false" }
40
41 !llvm.module.flags = !{!0, !1}
42 !llvm.ident = !{!2}
43
44 !0 = !{i32 1, !"wchar_size", i32 4}
45 !1 = !{i32 7, !"PIC Level", i32 2}
46 !2 = !{!"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}
```

3.5.2 Ler dois inteiros e decide qual é maior

Listagem 3.44: micro02.py

```
1 def micro02():
2     num1, num2 = 0 , 0
3     print("Digite o primeiro numero: ")
4     num1 = int(input())
5     print("Digite o segundo numero: ")
6     num2 = int(input())
7
8     if num1 > num2:
9         print("O primeiro numero " + str(num1) + " e maior que o segundo "
            + str(num2))
10    else:
11        print("O segundo numero " + str(num2) + " e maior que o primeiro "
            + str(num1))
12
13 micro02()
```

Listagem 3.45: micro02.c

```
1 #import<stdio.h>
2
3 int main(){
4     int num1, num2;
```

```

5  printf("Digite o primeiro numero: ");
6  scanf("%d", &num1);
7  printf("Digite o segundo numero: ");
8  scanf("%d", &num2);
9
10 if(num1 > num2) {
11     printf("O primeiro numero %d e maior que o segundo %d.", num1, num2);
12 } else {
13     printf("O primeiro numero %d e maior que o segundo %d.", num2, num1);
14 }
15
16 return 0;
17 }

```

Listagem 3.46: micro02.ll

```

1 ; ModuleID = 'micro02.c'
2 source_filename = "micro02.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [27 x i8] c"Digite o primeiro numero
   : \00", align 1
7 @.str.1 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
8 @.str.2 = private unnamed_addr constant [26 x i8] c"Digite o segundo
   numero: \00", align 1
9 @.str.3 = private unnamed_addr constant [47 x i8] c"O primeiro numero %d e
   maior que o segundo %d.\00", align 1
10
11 ; Function Attrs: noinline nounwind optnone ssp uwtable
12 define i32 @main() #0 {
13     %1 = alloca i32, align 4
14     %2 = alloca i32, align 4
15     %3 = alloca i32, align 4
16     store i32 0, i32* %1, align 4
17     %4 = call i32 @__printf(i8* getelementptr inbounds ([27 x i8],
   [27 x i8]* @.str, i32 0, i32 0))
18     %5 = call i32 @__scanf(i8* getelementptr inbounds ([3 x i8], [3
   x i8]* @.str.1, i32 0, i32 0), i32* %2)
19     %6 = call i32 @__printf(i8* getelementptr inbounds ([26 x i8],
   [26 x i8]* @.str.2, i32 0, i32 0))
20     %7 = call i32 @__scanf(i8* getelementptr inbounds ([3 x i8], [3
   x i8]* @.str.1, i32 0, i32 0), i32* %3)
21     %8 = load i32, i32* %2, align 4
22     %9 = load i32, i32* %3, align 4
23     %10 = icmp sgt i32 %8, %9
24     br i1 %10, label %11, label %15
25
26 ; <label>:11:                                ; preds = %0
27     %12 = load i32, i32* %2, align 4
28     %13 = load i32, i32* %3, align 4
29     %14 = call i32 @__printf(i8* getelementptr inbounds ([47 x i8],
   [47 x i8]* @.str.3, i32 0, i32 0), i32 %12, i32 %13)
30     br label %19
31
32 ; <label>:15:                                ; preds = %0
33     %16 = load i32, i32* %3, align 4
34     %17 = load i32, i32* %2, align 4
35     %18 = call i32 @__printf(i8* getelementptr inbounds ([47 x i8],

```

```

[47 x i8]* @.str.3, i32 0, i32 0), i32 %16, i32 %17)
36 br label %19
37
38 ; <label>:19:                                ; preds = %15, %11
39 ret i32 0
40 }
41
42 declare i32 @printf(i8*, ...) #1
43
44 declare i32 @scanf(i8*, ...) #1
45
46 attributes #0 = { noinline nounwind optnone ssp uwtable "correctly-rounded-
-divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-
precise-fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-
elim-non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-
nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-
math"="false" "stack-protector-buffer-size"="8" "target-cpu"="penryn" "
target-features"="+cx16,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+ssse3,+x87
" "unsafe-fp-math"="false" "use-soft-float"="false" }
47 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
"no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
penryn" "target-features"="+cx16,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+
ssse3,+x87" "unsafe-fp-math"="false" "use-soft-float"="false" }
48
49 !llvm.module.flags = !{!0, !1}
50 !llvm.ident = !{!2}
51
52 !0 = !{i32 1, !"wchar_size", i32 4}
53 !1 = !{i32 7, !"PIC Level", i32 2}
54 !2 = !{!"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.5.3 Lê um número e verifica se ele está entre 100 e 200

Listagem 3.47: micro03.py

```

1 def micro03():
2     numero = 0
3     print("Digite um numero: ")
4     numero = int(input())
5     if numero >= 100:
6         if numero <= 200:
7             print("O numero esta no intervalo entre 100 e 200")
8         else:
9             print("O numero nao esta no intervalo entre 100 e 200")
10    else:
11        print("O numero nao esta no intervalo entre 100 e 200")
12
13 micro03()

```

Listagem 3.48: micro03.c

```

1 #import<stdio.h>
2
3 int main(){
4     int numero;

```

```

5  printf("Digite um numero: ");
6  scanf("%d", &numero);
7
8  if (numero >= 100) {
9      if (numero <= 200) {
10         printf("O numero esta no intervalo entre 100 e 200");
11     } else {
12         printf("O numero nao esta no intervalo entre 100 e 200");
13     }
14 } else {
15     printf("O numero nao esta no intervalo entre 100 e 200");
16 }
17
18 return 0;
19 }

```

Listagem 3.49: micro03.ll

```

1 ; ModuleID = 'micro03.c'
2 source_filename = "micro03.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [19 x i8] c"Digite um numero: \00",
    align 1
7 @.str.1 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
8 @.str.2 = private unnamed_addr constant [43 x i8] c"O numero esta no
    intervalo entre 100 e 200\00", align 1
9 @.str.3 = private unnamed_addr constant [47 x i8] c"O numero nao esta no
    intervalo entre 100 e 200\00", align 1
10
11 ; Function Attrs: noinline nounwind optnone ssp uwtable
12 define i32 @main() #0 {
13     %1 = alloca i32, align 4
14     %2 = alloca i32, align 4
15     store i32 0, i32* %1, align 4
16     %3 = call i32 @__printf(i8*, ...) @printf(i8* getelementptr inbounds ([19 x i8],
        [19 x i8]* @.str, i32 0, i32 0))
17     %4 = call i32 @__scanf(i8*, ...) @scanf(i8* getelementptr inbounds ([3 x i8], [3
        x i8]* @.str.1, i32 0, i32 0), i32* %2)
18     %5 = load i32, i32* %2, align 4
19     %6 = icmp sge i32 %5, 100
20     br i1 %6, label %7, label %15
21
22 ; <label>:7:                                     ; preds = %0
23     %8 = load i32, i32* %2, align 4
24     %9 = icmp sle i32 %8, 200
25     br i1 %9, label %10, label %12
26
27 ; <label>:10:                                     ; preds = %7
28     %11 = call i32 @__printf(i8*, ...) @printf(i8* getelementptr inbounds ([43 x i8],
        [43 x i8]* @.str.2, i32 0, i32 0))
29     br label %14
30
31 ; <label>:12:                                     ; preds = %7
32     %13 = call i32 @__printf(i8*, ...) @printf(i8* getelementptr inbounds ([47 x i8],
        [47 x i8]* @.str.3, i32 0, i32 0))
33     br label %14
34

```


3.5

```
35 ; <label>:14:                                ; preds = %12, %10
36   br label %17
37
38 ; <label>:15:                                ; preds = %0
39   %16 = call i32 @i8*, ... @printf(i8* getelementptr @inbounds ([47 x i8],
40     [47 x i8]* @.str.3, i32 0, i32 0))
41   br label %17
42 ; <label>:17:                                ; preds = %15, %14
43   ret i32 0
44 }
45
46 declare i32 @printf(i8*, ...) #1
47
48 declare i32 @scanf(i8*, ...) #1
49
50 attributes #0 = { noinline nounwind optnone ssp uwtable "correctly-rounded-
  divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-
  precise-fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-
  elim-non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-
  nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-
  math"="false" "stack-protector-buffer-size"="8" "target-cpu"="penryn" "
  target-features"="+cxl6,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+ssse3,+x87
  " "unsafe-fp-math"="false" "use-soft-float"="false" }
51 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
  tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
  elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
  "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
  trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
  penryn" "target-features"="+cxl6,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+
  ssse3,+x87" "unsafe-fp-math"="false" "use-soft-float"="false" }
52
53 !llvm.module.flags = !{!0, !1}
54 !llvm.ident = !{!2}
55
56 !0 = !{i32 1, !"wchar_size", i32 4}
57 !1 = !{i32 7, !"PIC Level", i32 2}
58 !2 = !{!"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}
```

3.5.4 Lê números e informa quais estão entre 10 e 150

Listagem 3.50: micro04.py

```
1 def micro04():
2     x, num, intervalo = 0,0,0
3     for x in range(5):
4         print("Digite o numero: ")
5         num = int(input())
6         if num >= 10:
7             if num <= 150:
8                 intervalo = intervalo + 1
9         print("Ao total, foram digitados " + str(intervalo) + " numeros no
10           intervalo entre 10 e 150")
11 micro04()
```

Listagem 3.51: micro04.c

```

1 #import<stdio.h>
2
3 int main(){
4     int x, num, intervalo;
5
6     for (x = 0; x < 5; x++) {
7         printf("Digite um numero: ");
8         scanf("%d", &num);
9         if(num >= 10) {
10             if (num <= 150) {
11                 intervalo = intervalo + 1;
12             }
13         }
14     }
15
16     printf("Ao total, foram digitados %d numeros no intervalo entre 10 e 150.",
17           intervalo);
18
19     return 0;
20 }

```

Listagem 3.52: micro04.ll

```

1 ; ModuleID = 'micro04.c'
2 source_filename = "micro04.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [19 x i8] c"Digite um numero: \00",
7     align 1
8 @.str.1 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
9 @.str.2 = private unnamed_addr constant [63 x i8] c"Ao total, foram
10     digitados %d numeros no intervalo entre 10 e 150.\00", align 1
11
12 ; Function Attrs: noinline nounwind optnone ssp uwtable
13 define i32 @main() #0 {
14     %1 = alloca i32, align 4
15     %2 = alloca i32, align 4
16     %3 = alloca i32, align 4
17     %4 = alloca i32, align 4
18     store i32 0, i32* %1, align 4
19     store i32 0, i32* %2, align 4
20     br label %5
21
22 ; <label>:5:                                     ; preds = %21, %0
23     %6 = load i32, i32* %2, align 4
24     %7 = icmp slt i32 %6, 5
25     br i1 %7, label %8, label %24
26
27 ; <label>:8:                                     ; preds = %5
28     %9 = call i32 @__printf(i8*, ...) @printf(i8* getelementptr inbounds ([19 x i8],
29     [19 x i8]* @.str, i32 0, i32 0))
30     %10 = call i32 @__scanf(i8* getelementptr inbounds ([3 x i8],
31     [3 x i8]* @.str.1, i32 0, i32 0), i32* %3)
32     %11 = load i32, i32* %3, align 4
33     %12 = icmp sge i32 %11, 10
34     br i1 %12, label %13, label %20
35
36 ; <label>:13:                                    ; preds = %8

```

3.5

```
33 %14 = load i32, i32* %3, align 4
34 %15 = icmp sle i32 %14, 150
35 br i1 %15, label %16, label %19
36
37 ; <label>:16:                                ; preds = %13
38 %17 = load i32, i32* %4, align 4
39 %18 = add nsw i32 %17, 1
40 store i32 %18, i32* %4, align 4
41 br label %19
42
43 ; <label>:19:                                ; preds = %16, %13
44 br label %20
45
46 ; <label>:20:                                ; preds = %19, %8
47 br label %21
48
49 ; <label>:21:                                ; preds = %20
50 %22 = load i32, i32* %2, align 4
51 %23 = add nsw i32 %22, 1
52 store i32 %23, i32* %2, align 4
53 br label %5
54
55 ; <label>:24:                                ; preds = %5
56 %25 = load i32, i32* %4, align 4
57 %26 = call i32 @i8*, ... @printf(i8* getelementptr @inbounds ([63 x i8],
    [63 x i8]* @.str.2, i32 0, i32 0), i32 %25)
58 ret i32 0
59 }
60
61 declare i32 @printf(i8*, ...) #1
62
63 declare i32 @scanf(i8*, ...) #1
64
65 attributes #0 = { noline nounwind optnone ssp uwtable "correctly-rounded-
    -divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-
    precise-fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-
    -elim-non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-
    nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-
    math"="false" "stack-protector-buffer-size"="8" "target-cpu"="penryn" "
    target-features"="+cxl6,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+ssse3,+x87
    " "unsafe-fp-math"="false" "use-soft-float"="false" }
66 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
    -tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
    elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
    "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
    trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
    penryn" "target-features"="+cxl6,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+
    ssse3,+x87" "unsafe-fp-math"="false" "use-soft-float"="false" }
67
68 !llvm.module.flags = !{!0, !1}
69 !llvm.ident = !{!2}
70
71 !0 = !{i32 1, !"wchar_size", i32 4}
72 !1 = !{i32 7, !"PIC Level", i32 2}
73 !2 = !{"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}
```

3.5.5 Lê strings e caracteres

Listagem 3.53: micro05.py

```

1 def micro05():
2     x,m,h = 0,0,0
3     nome,sexo = "", ""
4     for x in range(5):
5         print("Digite o nome: ")
6         nome = input()
7         print("H - Homem ou M - Mulher")
8         sexo = input()
9         if sexo == 'H':
10             h = h+1
11         elif sexo == 'M':
12             m = m+1
13         else:
14             print("Sexo so pode ser H ou M!")
15
16     print("Foram inseridos " + str(h) + " Homens")
17     print("Foram inseridas " + str(m) + " Mulheres")
18
19 micro05()

```

Listagem 3.54: micro05.c

```

1 #import<stdio.h>
2 #include <string.h>
3
4 int main(int argc, char *argv[]){
5     int x, m, h;
6     char nome, sexo;
7
8     for (x = 0; x < 5; x++) {
9         printf("Digite o nome: ");
10        scanf("%c", &nome);
11
12        printf("H - Homem ou M - Mulher: ");
13        scanf("%c", &sexo);
14
15        switch (sexo) {
16            case 'H':
17                h = h + 1;
18                break;
19            case 'M':
20                m = m + 1;
21                break;
22            default:
23                printf("Sexo só pode ser H ou M!");
24                break;
25        }
26    }
27
28    printf("Foram inseridos %d Homens", h);
29    printf("Foram inseridas %d Mulheres", m);
30
31    return 0;
32 }

```

Listagem 3.55: micro05.ll

3.5

```

1 ; ModuleID = 'micro05.c'
2 source_filename = "micro05.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [16 x i8] c"Digite o nome: \00",
    align 1
7 @.str.1 = private unnamed_addr constant [3 x i8] c"%c\00", align 1
8 @.str.2 = private unnamed_addr constant [26 x i8] c"H - Homem ou M -
    Mulher: \00", align 1
9 @.str.3 = private unnamed_addr constant [26 x i8] c"Sexo s\C3\B3 pode ser
    H ou M!\00", align 1
10 @.str.4 = private unnamed_addr constant [26 x i8] c"Foram inseridos %d
    Homens\00", align 1
11 @.str.5 = private unnamed_addr constant [28 x i8] c"Foram inseridas %d
    Mulheres\00", align 1
12
13 ; Function Attrs: noinline nounwind optnone ssp uwtable
14 define i32 @main(i32, i8**) #0 {
15     %3 = alloca i32, align 4
16     %4 = alloca i32, align 4
17     %5 = alloca i8**, align 8
18     %6 = alloca i32, align 4
19     %7 = alloca i32, align 4
20     %8 = alloca i32, align 4
21     %9 = alloca i8, align 1
22     %10 = alloca i8, align 1
23     store i32 0, i32* %3, align 4
24     store i32 %0, i32* %4, align 4
25     store i8** %1, i8*** %5, align 8
26     store i32 0, i32* %6, align 4
27     br label %11
28
29 ; <label>:11:                                ; preds = %30, %2
30     %12 = load i32, i32* %6, align 4
31     %13 = icmp slt i32 %12, 5
32     br i1 %13, label %14, label %33
33
34 ; <label>:14:                                ; preds = %11
35     %15 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([16 x i8],
        [16 x i8]* @.str, i32 0, i32 0))
36     %16 = call i32 (i8*, ...) @scanf(i8* getelementptr inbounds ([3 x i8],
        [3 x i8]* @.str.1, i32 0, i32 0), i8* %9)
37     %17 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([26 x i8],
        [26 x i8]* @.str.2, i32 0, i32 0))
38     %18 = call i32 (i8*, ...) @scanf(i8* getelementptr inbounds ([3 x i8],
        [3 x i8]* @.str.1, i32 0, i32 0), i8* %10)
39     %19 = load i8, i8* %10, align 1
40     %20 = sext i8 %19 to i32
41     switch i32 %20, label %27 [
42         i32 72, label %21
43         i32 77, label %24
44     ]
45
46 ; <label>:21:                                ; preds = %14
47     %22 = load i32, i32* %8, align 4
48     %23 = add nsw i32 %22, 1
49     store i32 %23, i32* %8, align 4
50     br label %29

```

```

51
52 ; <label>:24:                                     ; preds = %14
53   %25 = load i32, i32* %7, align 4
54   %26 = add nsw i32 %25, 1
55   store i32 %26, i32* %7, align 4
56   br label %29
57
58 ; <label>:27:                                     ; preds = %14
59   %28 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([26 x i8],
60     [26 x i8]* @.str.3, i32 0, i32 0))
61   br label %29
62 ; <label>:29:                                     ; preds = %27, %24, %21
63   br label %30
64
65 ; <label>:30:                                     ; preds = %29
66   %31 = load i32, i32* %6, align 4
67   %32 = add nsw i32 %31, 1
68   store i32 %32, i32* %6, align 4
69   br label %11
70
71 ; <label>:33:                                     ; preds = %11
72   %34 = load i32, i32* %8, align 4
73   %35 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([26 x i8],
74     [26 x i8]* @.str.4, i32 0, i32 0), i32 %34)
75   %36 = load i32, i32* %7, align 4
76   %37 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([28 x i8],
77     [28 x i8]* @.str.5, i32 0, i32 0), i32 %36)
78   ret i32 0
79 }
80
81 declare i32 @printf(i8*, ...) #1
82
83 declare i32 @scanf(i8*, ...) #1
84
85 attributes #0 = { noinline nounwind optnone ssp uwtable "correctly-rounded-
86   -divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-
87   precise-fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-
88   elim-non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-
89   nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-
90   math"="false" "stack-protector-buffer-size"="8" "target-cpu"="penryn" "
91   target-features"="+cxl16,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+ssse3,+x87
92   " "unsafe-fp-math"="false" "use-soft-float"="false" }
93
94 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
95   tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
96   elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
97   "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
98   trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
99   penryn" "target-features"="+cxl16,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+
100   ssse3,+x87" "unsafe-fp-math"="false" "use-soft-float"="false" }
101
102
103 !0 = !{i32 1, !"wchar_size", i32 4}
104 !1 = !{i32 7, !"PIC Level", i32 2}
105 !2 = !{"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.5.6 Escreve um número lido por extenso

Listagem 3.56: micro06.py

```

1 def micro06():
2     numero = 0
3     print("Digite um numero de 1 a 5: ")
4     numero = int(input())
5     if numero == 1:
6         print("Um")
7     elif numero == 2:
8         print("Dois")
9     elif numero == 3:
10        print("Tres")
11    elif numero == 4:
12        print("Quatro")
13    elif numero == 5:
14        print("Cinco")
15    else:
16        print("Numero Invalido!!!")
17
18 micro06()

```

Listagem 3.57: micro06.c

```

1 #import<stdio.h>
2
3 int main(){
4     int numero;
5     printf("Digite um numero de 1 a 5: ");
6     scanf("%d", &numero);
7
8     if (numero == 1) {
9         printf("Um");
10    } else if(numero == 2) {
11        printf("Dois");
12    } else if(numero == 3) {
13        printf("Tres");
14    } else if(numero == 4) {
15        printf("Quatro");
16    } else if(numero == 5) {
17        printf("Cinco");
18    } else {
19        printf("Numero Invalido!!!");
20    }
21
22    return 0;
23 }

```

Listagem 3.58: micro06.ll

```

1 ; ModuleID = 'micro06.c'
2 source_filename = "micro06.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [28 x i8] c"Digite um numero de 1 a
5: \00", align 1
7 @.str.1 = private unnamed_addr constant [3 x i8] c"%d\00", align 1

```

```

8 @.str.2 = private unnamed_addr constant [3 x i8] c"Um\00", align 1
9 @.str.3 = private unnamed_addr constant [5 x i8] c"Dois\00", align 1
10 @.str.4 = private unnamed_addr constant [5 x i8] c"Tres\00", align 1
11 @.str.5 = private unnamed_addr constant [7 x i8] c"Quatro\00", align 1
12 @.str.6 = private unnamed_addr constant [6 x i8] c"Cinco\00", align 1
13 @.str.7 = private unnamed_addr constant [19 x i8] c"Numero Invalido!!!\00"
    , align 1
14
15 ; Function Attrs: noinline nounwind optnone ssp uwtable
16 define i32 @main() #0 {
17     %1 = alloca i32, align 4
18     %2 = alloca i32, align 4
19     store i32 0, i32* %1, align 4
20     %3 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([28 x i8],
        [28 x i8]* @.str, i32 0, i32 0))
21     %4 = call i32 @i8*, ... @scanf(i8* getelementptr inbounds ([3 x i8], [3
        x i8]* @.str.1, i32 0, i32 0), i32* %2)
22     %5 = load i32, i32* %2, align 4
23     %6 = icmp eq i32 %5, 1
24     br i1 %6, label %7, label %9
25
26 ; <label>:7:                                ; preds = %0
27     %8 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([3 x i8],
        [3 x i8]* @.str.2, i32 0, i32 0))
28     br label %35
29
30 ; <label>:9:                                ; preds = %0
31     %10 = load i32, i32* %2, align 4
32     %11 = icmp eq i32 %10, 2
33     br i1 %11, label %12, label %14
34
35 ; <label>:12:                               ; preds = %9
36     %13 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([5 x i8],
        [5 x i8]* @.str.3, i32 0, i32 0))
37     br label %34
38
39 ; <label>:14:                               ; preds = %9
40     %15 = load i32, i32* %2, align 4
41     %16 = icmp eq i32 %15, 3
42     br i1 %16, label %17, label %19
43
44 ; <label>:17:                               ; preds = %14
45     %18 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([5 x i8],
        [5 x i8]* @.str.4, i32 0, i32 0))
46     br label %33
47
48 ; <label>:19:                               ; preds = %14
49     %20 = load i32, i32* %2, align 4
50     %21 = icmp eq i32 %20, 4
51     br i1 %21, label %22, label %24
52
53 ; <label>:22:                               ; preds = %19
54     %23 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([7 x i8],
        [7 x i8]* @.str.5, i32 0, i32 0))
55     br label %32
56
57 ; <label>:24:                               ; preds = %19
58     %25 = load i32, i32* %2, align 4
59     %26 = icmp eq i32 %25, 5

```


3.5

```
60 br i1 %26, label %27, label %29
61
62 ; <label>:27:                                ; preds = %24
63 %28 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([6 x i8],
64 [6 x i8]* @.str.6, i32 0, i32 0))
65 br label %31
66
66 ; <label>:29:                                ; preds = %24
67 %30 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([19 x i8],
68 [19 x i8]* @.str.7, i32 0, i32 0))
69 br label %31
70
70 ; <label>:31:                                ; preds = %29, %27
71 br label %32
72
73 ; <label>:32:                                ; preds = %31, %22
74 br label %33
75
76 ; <label>:33:                                ; preds = %32, %17
77 br label %34
78
79 ; <label>:34:                                ; preds = %33, %12
80 br label %35
81
82 ; <label>:35:                                ; preds = %34, %7
83 ret i32 0
84 }
85
86 declare i32 @printf(i8*, ...) #1
87
88 declare i32 @scanf(i8*, ...) #1
89
90 attributes #0 = { noline nounwind optnone ssp uwtable "correctly-rounded-
-divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-
precise-fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-
-elim-non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-
nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-
math"="false" "stack-protector-buffer-size"="8" "target-cpu"="penryn" "
target-features"="+cxl6,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+ssse3,+x87
" "unsafe-fp-math"="false" "use-soft-float"="false" }
91 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
-tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
"no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
penryn" "target-features"="+cxl6,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+
ssse3,+x87" "unsafe-fp-math"="false" "use-soft-float"="false" }
92
93 !llvm.module.flags = !{!0, !1}
94 !llvm.ident = !{!2}
95
96 !0 = !{i32 1, !"wchar_size", i32 4}
97 !1 = !{i32 7, !"PIC Level", i32 2}
98 !2 = !{!"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}
```

3.5.7 Decide se os números são positivos, zeros ou negativos

```

1 def micro07():
2     numero ,programa = 0,1
3     opc = ""
4     while programa == 1:
5         print("Digite um numero: ")
6         numero = int(input())
7         if numero > 0:
8             print("Positivo")
9         else:
10            if numero == 0:
11                print("O numero e igual a 0")
12            elif numero < 0:
13                print("Negativo")
14
15        print("Deseja Finalizar? (S/N) ")
16        opc = input()
17        if opc == "S":
18            programa = 0
19
20 micro07()

```

Listagem 3.60: micro07.c

```

1 #import<stdio.h>
2
3 int main() {
4     int numero = 0;
5     int programa = 1;
6     char opc;
7
8     while (programa == 1) {
9         printf("Digite um numero: ");
10        scanf("%d", &numero);
11
12        if(numero > 0) {
13            printf("Positivo");
14        } else {
15            if (numero == 0) {
16                printf("O numero e igual a 0");
17            } else if (numero < 0) {
18                printf("Negativo");
19            }
20        }
21        printf("Deseja Finalizar? (S/N) ");
22        scanf("%c", &opc);
23        if (opc == 'S') {
24            programa = 0;
25        }
26    }
27    return 0;
28 }

```

Listagem 3.61: micro07.ll

```

1 ; ModuleID = 'micro07.c'
2 source_filename = "micro07.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5

```

3.5

```

6 @.str = private unnamed_addr constant [19 x i8] c"Digite um numero: \00",
    align 1
7 @.str.1 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
8 @.str.2 = private unnamed_addr constant [9 x i8] c"Positivo\00", align 1
9 @.str.3 = private unnamed_addr constant [21 x i8] c"O numero e igual a
    0\00", align 1
10 @.str.4 = private unnamed_addr constant [9 x i8] c"Negativo\00", align 1
11 @.str.5 = private unnamed_addr constant [25 x i8] c"Deseja Finalizar? (S/N
    ) \00", align 1
12 @.str.6 = private unnamed_addr constant [3 x i8] c"%c\00", align 1
13
14 ; Function Attrs: noinline nounwind optnone ssp uwtable
15 define i32 @main() #0 {
16     %1 = alloca i32, align 4
17     %2 = alloca i32, align 4
18     %3 = alloca i32, align 4
19     %4 = alloca i8, align 1
20     store i32 0, i32* %1, align 4
21     store i32 0, i32* %2, align 4
22     store i32 1, i32* %3, align 4
23     br label %5
24
25 ; <label>:5:                                ; preds = %34, %0
26     %6 = load i32, i32* %3, align 4
27     %7 = icmp eq i32 %6, 1
28     br i1 %7, label %8, label %35
29
30 ; <label>:8:                                ; preds = %5
31     %9 = call i32 (@i8*, ...) @printf(i8* getelementptr inbounds ([19 x i8],
        [19 x i8]* @.str, i32 0, i32 0))
32     %10 = call i32 (@i8*, ...) @scanf(i8* getelementptr inbounds ([3 x i8],
        [3 x i8]* @.str.1, i32 0, i32 0), i32* %2)
33     %11 = load i32, i32* %2, align 4
34     %12 = icmp sgt i32 %11, 0
35     br i1 %12, label %13, label %15
36
37 ; <label>:13:                                ; preds = %8
38     %14 = call i32 (@i8*, ...) @printf(i8* getelementptr inbounds ([9 x i8],
        [9 x i8]* @.str.2, i32 0, i32 0))
39     br label %27
40
41 ; <label>:15:                                ; preds = %8
42     %16 = load i32, i32* %2, align 4
43     %17 = icmp eq i32 %16, 0
44     br i1 %17, label %18, label %20
45
46 ; <label>:18:                                ; preds = %15
47     %19 = call i32 (@i8*, ...) @printf(i8* getelementptr inbounds ([21 x i8],
        [21 x i8]* @.str.3, i32 0, i32 0))
48     br label %26
49
50 ; <label>:20:                                ; preds = %15
51     %21 = load i32, i32* %2, align 4
52     %22 = icmp slt i32 %21, 0
53     br i1 %22, label %23, label %25
54
55 ; <label>:23:                                ; preds = %20
56     %24 = call i32 (@i8*, ...) @printf(i8* getelementptr inbounds ([9 x i8],
        [9 x i8]* @.str.4, i32 0, i32 0))

```

```

57  br label %25
58
59 ; <label>:25:                                ; preds = %23, %20
60  br label %26
61
62 ; <label>:26:                                ; preds = %25, %18
63  br label %27
64
65 ; <label>:27:                                ; preds = %26, %13
66  %28 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([25 x i8],
    [25 x i8]* @.str.5, i32 0, i32 0))
67  %29 = call i32 @i8*, ... @scanf(i8* getelementptr inbounds ([3 x i8],
    [3 x i8]* @.str.6, i32 0, i32 0), i8* %4)
68  %30 = load i8, i8* %4, align 1
69  %31 = sext i8 %30 to i32
70  %32 = icmp eq i32 %31, 83
71  br i1 %32, label %33, label %34
72
73 ; <label>:33:                                ; preds = %27
74  store i32 0, i32* %3, align 4
75  br label %34
76
77 ; <label>:34:                                ; preds = %33, %27
78  br label %5
79
80 ; <label>:35:                                ; preds = %5
81  ret i32 0
82 }
83
84 declare i32 @printf(i8*, ...) #1
85
86 declare i32 @scanf(i8*, ...) #1
87
88 attributes #0 = { noinline nounwind optnone ssp uwtable "correctly-rounded-
    -divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-
    precise-fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-
    -elim-non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-
    nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-
    math"="false" "stack-protector-buffer-size"="8" "target-cpu"="penryn" "
    target-features"="+cxl6,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+ssse3,+x87
    " "unsafe-fp-math"="false" "use-soft-float"="false" }
89 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
    -tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
    elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
    "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
    trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
    penryn" "target-features"="+cxl6,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+
    ssse3,+x87" "unsafe-fp-math"="false" "use-soft-float"="false" }
90
91 !llvm.module.flags = !{!0, !1}
92 !llvm.ident = !{!2}
93
94 !0 = !{i32 1, !"wchar_size", i32 4}
95 !1 = !{i32 7, !"PIC Level", i32 2}
96 !2 = !{"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.5.8 Decide se um número é maior ou menor que 10

Listagem 3.62: micro08.py

```

1 def micro08():
2     numero = 1
3     while numero < 0 or numero > 0:
4         print("Digite o numero")
5         numero = int(input())
6         if numero > 10:
7             print("O numero "+str(numero)+" e maior que 10")
8         else:
9             print("O numero "+str(numero)+" e menor que 10")
10
11 micro08()

```

Listagem 3.63: micro08.c

```

1 #import<stdio.h>
2
3 int main() {
4     int numero = 1;
5
6     while (numero < 0 || numero > 0) {
7         printf("Digite um numero: ");
8         scanf("%d", &numero);
9
10        if(numero > 10) {
11            printf("O numero %d e maior que 10", numero);
12        } else {
13            printf("O numero %d e menor que 10", numero);
14        }
15    }
16    return 0;
17 }

```

Listagem 3.64: micro08.ll

```

1 ; ModuleID = 'micro08.c'
2 source_filename = "micro08.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [19 x i8] c"Digite um numero: \00",
    align 1
7 @.str.1 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
8 @.str.2 = private unnamed_addr constant [27 x i8] c"O numero %d e maior
    que 10\00", align 1
9 @.str.3 = private unnamed_addr constant [27 x i8] c"O numero %d e menor
    que 10\00", align 1
10
11 ; Function Attrs: noinline nounwind optnone ssp uwtable
12 define i32 @main() #0 {
13     %1 = alloca i32, align 4
14     %2 = alloca i32, align 4
15     store i32 0, i32* %1, align 4
16     store i32 1, i32* %2, align 4
17     br label %3
18
19 ; <label>:3:                                ; preds = %22, %0
20     %4 = load i32, i32* %2, align 4

```

```

21  %5 = icmp slt i32 %4, 0
22  br i1 %5, label %9, label %6
23
24 ; <label>:6:                                ; preds = %3
25  %7 = load i32, i32* %2, align 4
26  %8 = icmp sgt i32 %7, 0
27  br label %9
28
29 ; <label>:9:                                ; preds = %6, %3
30  %10 = phi i1 [ true, %3 ], [ %8, %6 ]
31  br i1 %10, label %11, label %23
32
33 ; <label>:11:                               ; preds = %9
34  %12 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([19 x i8],
    [19 x i8]* @.str, i32 0, i32 0))
35  %13 = call i32 @i8*, ... @scanf(i8* getelementptr inbounds ([3 x i8],
    [3 x i8]* @.str.1, i32 0, i32 0), i32* %2)
36  %14 = load i32, i32* %2, align 4
37  %15 = icmp sgt i32 %14, 10
38  br i1 %15, label %16, label %19
39
40 ; <label>:16:                               ; preds = %11
41  %17 = load i32, i32* %2, align 4
42  %18 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([27 x i8],
    [27 x i8]* @.str.2, i32 0, i32 0), i32 %17)
43  br label %22
44
45 ; <label>:19:                               ; preds = %11
46  %20 = load i32, i32* %2, align 4
47  %21 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([27 x i8],
    [27 x i8]* @.str.3, i32 0, i32 0), i32 %20)
48  br label %22
49
50 ; <label>:22:                               ; preds = %19, %16
51  br label %3
52
53 ; <label>:23:                               ; preds = %9
54  ret i32 0
55 }
56
57 declare i32 @printf(i8*, ...) #1
58
59 declare i32 @scanf(i8*, ...) #1
60
61 attributes #0 = { noinline nounwind optnone ssp uwtable "correctly-rounded-
    -divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-
    -precise-fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-
    -elim-non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-
    -nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-
    -math"="false" "stack-protector-buffer-size"="8" "target-cpu"="penryn" "
    target-features"="+cx16,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+ssse3,+x87
    " "unsafe-fp-math"="false" "use-soft-float"="false" }
62 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
    -tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
    -elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
    "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
    trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
    penryn" "target-features"="+cx16,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+
    ssse3,+x87" "unsafe-fp-math"="false" "use-soft-float"="false" }

```

```

63
64 !llvm.module.flags = !{!0, !1}
65 !llvm.ident = !{!2}
66
67 !0 = !{i32 1, !"wchar_size", i32 4}
68 !1 = !{i32 7, !"PIC Level", i32 2}
69 !2 = !{"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.5.9 Cálculo de preços

Listagem 3.65: micro09.py

```

1 def micro09():
2     preco, venda, novopreco = 0.0, 0.0, 0.0
3     print("Digite o preco: ")
4     preco = int(input())
5     print("Digite a venda: ")
6     venda = int(input())
7     if venda < 500 or preco < 30:
8         novopreco = preco + 10/100 * preco
9     elif (venda >= 500 and venda < 1200) or (preco >= 30 and preco < 80):
10        novopreco = preco + 15/100 * preco
11    elif venda >= 1200 or preco >= 80:
12        novopreco = preco - 20/100 * preco
13
14    print("O novo preco e: "+str(novopreco))
15
16 micro09()

```

Listagem 3.66: micro09.c

```

1 #import<stdio.h>
2
3 int main() {
4     float preco, venda, novopreco;
5
6     printf("Digite o preco: ");
7     scanf("%f", &preco);
8
9     printf("Digite o venda: ");
10    scanf("%f", &venda);
11
12    if(venda < 500 || preco < 30) {
13        novopreco = preco + 10.0 / 100.0 * preco;
14    } else if((venda >= 500 && venda < 1200) || (preco >= 30 && preco < 80)) {
15        novopreco = preco + 15.0 / 100.0 * preco;
16    } else {
17        novopreco = preco + 20.0 / 100.0 * preco;
18    }
19
20    printf("O novo preco e %f", novopreco);
21    return 0;
22 }

```

Listagem 3.67: micro09.ll

```

1 ; ModuleID = 'micro09.c'

```

```

2 source_filename = "micro09.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [17 x i8] c"Digite o preco: \00",
    align 1
7 @.str.1 = private unnamed_addr constant [3 x i8] c"%f\00", align 1
8 @.str.2 = private unnamed_addr constant [17 x i8] c"Digite o venda: \00",
    align 1
9 @.str.3 = private unnamed_addr constant [18 x i8] c"O novo preco e %f\00",
    align 1
10
11 ; Function Attrs: noinline nounwind optnone ssp uwtable
12 define i32 @main() #0 {
13     %1 = alloca i32, align 4
14     %2 = alloca float, align 4
15     %3 = alloca float, align 4
16     %4 = alloca float, align 4
17     store i32 0, i32* %1, align 4
18     %5 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([17 x i8],
19         [17 x i8]* @.str, i32 0, i32 0))
20     %6 = call i32 @i8*, ... @scanf(i8* getelementptr inbounds ([3 x i8], [3
21         x i8]* @.str.1, i32 0, i32 0), float* %2)
22     %7 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([17 x i8],
23         [17 x i8]* @.str.2, i32 0, i32 0))
24     %8 = call i32 @i8*, ... @scanf(i8* getelementptr inbounds ([3 x i8], [3
25         x i8]* @.str.1, i32 0, i32 0), float* %3)
26     %9 = load float, float* %3, align 4
27     %10 = fcmp olt float %9, 5.000000e+02
28     br i1 %10, label %14, label %11
29
30 ; <label>:11:                                ; preds = %0
31     %12 = load float, float* %2, align 4
32     %13 = fcmp olt float %12, 3.000000e+01
33     br i1 %13, label %14, label %22
34
35 ; <label>:14:                                ; preds = %11, %0
36     %15 = load float, float* %2, align 4
37     %16 = fpext float %15 to double
38     %17 = load float, float* %2, align 4
39     %18 = fpext float %17 to double
40     %19 = fmul double 1.000000e-01, %18
41     %20 = fadd double %16, %19
42     %21 = fptrunc double %20 to float
43     store float %21, float* %4, align 4
44     br label %51
45
46 ; <label>:22:                                ; preds = %11
47     %23 = load float, float* %3, align 4
48     %24 = fcmp oge float %23, 5.000000e+02
49     br i1 %24, label %25, label %28
50
51 ; <label>:25:                                ; preds = %22
52     %26 = load float, float* %3, align 4
53     %27 = fcmp olt float %26, 1.200000e+03
54     br i1 %27, label %34, label %28
55
56 ; <label>:28:                                ; preds = %25, %22
57     %29 = load float, float* %2, align 4

```


3.5

```

54  %30 = fcmp oge float %29, 3.000000e+01
55  br i1 %30, label %31, label %42
56
57 ; <label>:31:                                ; preds = %28
58  %32 = load float, float* %2, align 4
59  %33 = fcmp olt float %32, 8.000000e+01
60  br i1 %33, label %34, label %42
61
62 ; <label>:34:                                ; preds = %31, %25
63  %35 = load float, float* %2, align 4
64  %36 = fpext float %35 to double
65  %37 = load float, float* %2, align 4
66  %38 = fpext float %37 to double
67  %39 = fmul double 1.500000e-01, %38
68  %40 = fadd double %36, %39
69  %41 = fptrunc double %40 to float
70  store float %41, float* %4, align 4
71  br label %50
72
73 ; <label>:42:                                ; preds = %31, %28
74  %43 = load float, float* %2, align 4
75  %44 = fpext float %43 to double
76  %45 = load float, float* %2, align 4
77  %46 = fpext float %45 to double
78  %47 = fmul double 2.000000e-01, %46
79  %48 = fadd double %44, %47
80  %49 = fptrunc double %48 to float
81  store float %49, float* %4, align 4
82  br label %50
83
84 ; <label>:50:                                ; preds = %42, %34
85  br label %51
86
87 ; <label>:51:                                ; preds = %50, %14
88  %52 = load float, float* %4, align 4
89  %53 = fpext float %52 to double
90  %54 = call i32 @i8*, ... @printf(i8* getelementptr @inbounds ([18 x i8],
    [18 x i8]* @.str.3, i32 0, i32 0), double %53)
91  ret i32 0
92 }
93
94 declare i32 @printf(i8*, ...) #1
95
96 declare i32 @scanf(i8*, ...) #1
97
98 attributes #0 = { noinline nounwind optnone ssp uwtable "correctly-rounded-
    -divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-
    -precise-fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-
    -elim-non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-
    -nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-
    -math"="false" "stack-protector-buffer-size"="8" "target-cpu"="penryn" "
    target-features"="+cx16,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+ssse3,+x87
    " "unsafe-fp-math"="false" "use-soft-float"="false" }
99 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-
    -tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
    -elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
    "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
    trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
    penryn" "target-features"="+cx16,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+

```

```

    ssse3,+x87" "unsafe-fp-math"="false" "use-soft-float"="false" }
100
101 !llvm.module.flags = !{!0, !1}
102 !llvm.ident = !{!2}
103
104 !0 = !{i32 1, !"wchar_size", i32 4}
105 !1 = !{i32 7, !"PIC Level", i32 2}
106 !2 = !{"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.5.10 Calcula o fatorial de um número

Listagem 3.68: micro10.py

```

1 def micro10():
2     numero = 0
3     fat = 0
4     print("Digite um numero: ")
5     numero = int(input())
6     fat = fatorial(numero)
7     print("O fatorial de " + str(numero) + " e " + str(fat))
8
9 def fatorial(n):
10     if n <= 0:
11         return 1
12     else:
13         return (n * fatorial(n-1))
14
15 micro10()

```

Listagem 3.69: micro10.c

```

1 #import<stdio.h>
2
3 int fatorial(int n) {
4     if (n <= 0) {
5         return 1;
6     } else {
7         return (n * fatorial(n - 1));
8     }
9 }
10
11 int main() {
12     int numero, fat;
13
14     printf("Digite um numero: ");
15     scanf("%d", &numero);
16
17     fat = fatorial(numero);
18     printf("O fatorial de %d e %d", numero, fat);
19
20     return 0;
21 }

```

Listagem 3.70: micro10.ll

```

1 ; ModuleID = 'micro10.c'
2 source_filename = "micro10.c"

```

3.5

```

3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [19 x i8] c"Digite um numero: \00",
    align 1
7 @.str.1 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
8 @.str.2 = private unnamed_addr constant [22 x i8] c"O fatorial de %d e %d
    \00", align 1
9
10 ; Function Attrs: noinline nounwind optnone ssp uwtable
11 define i32 @fatorial(i32) #0 {
12     %2 = alloca i32, align 4
13     %3 = alloca i32, align 4
14     store i32 %0, i32* %3, align 4
15     %4 = load i32, i32* %3, align 4
16     %5 = icmp sle i32 %4, 0
17     br i1 %5, label %6, label %7
18
19 ; <label>:6:                                ; preds = %1
20     store i32 1, i32* %2, align 4
21     br label %13
22
23 ; <label>:7:                                ; preds = %1
24     %8 = load i32, i32* %3, align 4
25     %9 = load i32, i32* %3, align 4
26     %10 = sub nsw i32 %9, 1
27     %11 = call i32 @fatorial(i32 %10)
28     %12 = mul nsw i32 %8, %11
29     store i32 %12, i32* %2, align 4
30     br label %13
31
32 ; <label>:13:                               ; preds = %7, %6
33     %14 = load i32, i32* %2, align 4
34     ret i32 %14
35 }
36
37 ; Function Attrs: noinline nounwind optnone ssp uwtable
38 define i32 @main() #0 {
39     %1 = alloca i32, align 4
40     %2 = alloca i32, align 4
41     %3 = alloca i32, align 4
42     store i32 0, i32* %1, align 4
43     %4 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([19 x i8],
        [19 x i8]* @.str, i32 0, i32 0))
44     %5 = call i32 (i8*, ...) @scanf(i8* getelementptr inbounds ([3 x i8], [3
        x i8]* @.str.1, i32 0, i32 0), i32* %2)
45     %6 = load i32, i32* %2, align 4
46     %7 = call i32 @fatorial(i32 %6)
47     store i32 %7, i32* %3, align 4
48     %8 = load i32, i32* %2, align 4
49     %9 = load i32, i32* %3, align 4
50     %10 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([22 x i8],
        [22 x i8]* @.str.2, i32 0, i32 0), i32 %8, i32 %9)
51     ret i32 0
52 }
53
54 declare i32 @printf(i8*, ...) #1
55
56 declare i32 @scanf(i8*, ...) #1

```

```

57
58 attributes #0 = { noline nounwind optnone ssp uwtable "correctly-rounded
    -divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-
    precise-fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-
    -elim-non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-
    nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-
    math"="false" "stack-protector-buffer-size"="8" "target-cpu"="penryn" "
    target-features"="+cx16,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+ssse3,+x87
    " "unsafe-fp-math"="false" "use-soft-float"="false" }
59 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable
    -tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
    elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
    "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
    trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
    penryn" "target-features"="+cx16,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+
    ssse3,+x87" "unsafe-fp-math"="false" "use-soft-float"="false" }
60
61 !llvm.module.flags = !{!0, !1}
62 !llvm.ident = !{!2}
63
64 !0 = !{i32 1, !"wchar_size", i32 4}
65 !1 = !{i32 7, !"PIC Level", i32 2}
66 !2 = !{"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

3.5.11 Decide se um número é positivo, zero ou negativo com auxílio de uma função

Listagem 3.71: micro11.py

```

1 def micro11():
2     numero,x = 0,0
3     print("Digite um numero: ")
4     numero = int(input())
5     x = verifica(numero)
6     if x ==1:
7         print("Numero Positivo")
8     elif x ==0:
9         print("Zero")
10    else:
11        print("Negativo")
12
13 def verifica(n):
14     res = 0
15     if n > 0:
16         res = 1
17     elif n < 0:
18         res = -1
19     else:
20         res = 0
21
22     return res
23
24 micro11()

```

Listagem 3.72: micro11.c

```

1 #import<stdio.h>

```

```

2
3 int verifica(int n) {
4     int res;
5     if (n > 0) {
6         res = 1;
7     } else if (n < 0) {
8         res = -1;
9     } else {
10        res = 0;
11    }
12    return res;
13 }
14
15 int main() {
16     int numero, x;
17
18     printf("Digite um numero: ");
19     scanf("%d", &numero);
20
21     x = verifica(numero);
22     if(x == 1) {
23         printf("Numero Positivo");
24     } else if(x == 0) {
25         printf("Zero");
26     } else {
27         printf("Negativo");
28     }
29
30     return 0;
31 }

```

Listagem 3.73: micro11.ll

```

1 ; ModuleID = 'micro11.c'
2 source_filename = "micro11.c"
3 target datalayout = "e-m:o-i64:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-apple-macosx10.13.0"
5
6 @.str = private unnamed_addr constant [19 x i8] c"Digite um numero: \00",
    align 1
7 @.str.1 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
8 @.str.2 = private unnamed_addr constant [16 x i8] c"Numero Positivo\00",
    align 1
9 @.str.3 = private unnamed_addr constant [5 x i8] c"Zero\00", align 1
10 @.str.4 = private unnamed_addr constant [9 x i8] c"Negativo\00", align 1
11
12 ; Function Attrs: noinline nounwind optnone ssp uwtable
13 define i32 @verifica(i32) #0 {
14     %2 = alloca i32, align 4
15     %3 = alloca i32, align 4
16     store i32 %0, i32* %2, align 4
17     %4 = load i32, i32* %2, align 4
18     %5 = icmp sgt i32 %4, 0
19     br i1 %5, label %6, label %7
20
21 ; <label>:6:                                ; preds = %1
22     store i32 1, i32* %3, align 4
23     br label %13
24

```

```

25 ; <label>:7:                                ; preds = %1
26   %8 = load i32, i32* %2, align 4
27   %9 = icmp slt i32 %8, 0
28   br i1 %9, label %10, label %11
29
30 ; <label>:10:                                ; preds = %7
31   store i32 -1, i32* %3, align 4
32   br label %12
33
34 ; <label>:11:                                ; preds = %7
35   store i32 0, i32* %3, align 4
36   br label %12
37
38 ; <label>:12:                                ; preds = %11, %10
39   br label %13
40
41 ; <label>:13:                                ; preds = %12, %6
42   %14 = load i32, i32* %3, align 4
43   ret i32 %14
44 }
45
46 ; Function Attrs: noinline nounwind optnone ssp uwtable
47 define i32 @main() #0 {
48   %1 = alloca i32, align 4
49   %2 = alloca i32, align 4
50   %3 = alloca i32, align 4
51   store i32 0, i32* %1, align 4
52   %4 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([19 x i8],
53     [19 x i8]* @.str, i32 0, i32 0))
54   %5 = call i32 @i8*, ... @scanf(i8* getelementptr inbounds ([3 x i8], [3
55     x i8]* @.str.1, i32 0, i32 0), i32* %2)
56   %6 = load i32, i32* %2, align 4
57   %7 = call i32 @verifica(i32 %6)
58   store i32 %7, i32* %3, align 4
59   %8 = load i32, i32* %3, align 4
60   %9 = icmp eq i32 %8, 1
61   br i1 %9, label %10, label %12
62
63 ; <label>:10:                                ; preds = %0
64   %11 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([16 x i8],
65     [16 x i8]* @.str.2, i32 0, i32 0))
66   br label %20
67
68 ; <label>:12:                                ; preds = %0
69   %13 = load i32, i32* %3, align 4
70   %14 = icmp eq i32 %13, 0
71   br i1 %14, label %15, label %17
72
73 ; <label>:15:                                ; preds = %12
74   %16 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([5 x i8],
75     [5 x i8]* @.str.3, i32 0, i32 0))
76   br label %19
77
78 ; <label>:17:                                ; preds = %12
79   %18 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([9 x i8],
80     [9 x i8]* @.str.4, i32 0, i32 0))
81   br label %19
82
83 ; <label>:19:                                ; preds = %17, %15

```

```

79  br label %20
80
81  ; <label>:20:                                ; preds = %19, %10
82  ret i32 0
83 }
84
85 declare i32 @printf(i8*, ...) #1
86
87 declare i32 @scanf(i8*, ...) #1
88
89 attributes #0 = { noinline nounwind optnone ssp uwtable "correctly-rounded
-divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-
precise-fpmad"="false" "no-frame-pointer-elim"="true" "no-frame-pointer-
elim-non-leaf" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-
nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-trapping-
math"="false" "stack-protector-buffer-size"="8" "target-cpu"="penryn" "
target-features"="+cx16,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+ssse3,+x87
" "unsafe-fp-math"="false" "use-soft-float"="false" }
90 attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable
-tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false"
"no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false" "no-
trapping-math"="false" "stack-protector-buffer-size"="8" "target-cpu"="
penryn" "target-features"="+cx16,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+
ssse3,+x87" "unsafe-fp-math"="false" "use-soft-float"="false" }
91
92 !llvm.module.flags = !{!0, !1}
93 !llvm.ident = !{!2}
94
95 !0 = !{i32 1, !"wchar_size", i32 4}
96 !1 = !{i32 7, !"PIC Level", i32 2}
97 !2 = !{!"Apple LLVM version 9.1.0 (clang-902.0.39.2)"}

```

Capítulo 4

Compilador

Um compilador traduz um programa de uma linguagem textual facilmente entendida por um ser humano para uma linguagem de máquina, específica para um processador e sistema operacional.

Para compilar fazemos:

Listagem 4.1: Terminal

```
1 $ ocamllex lexico.mll
2 $ ocamlc -c lexico.ml
3 $ ocamlc -c pre_processador.ml
4 $ rlwrap ocaml
5 # #use "carregador.ml";;
6 # lex "file.py";;
```

4.1 Analisador Léxico

Listagem 4.2: lexico.mll

```
1 {
2   open Lexing
3   open Printf
4
5   type token =
6   | LITINT of (int)
7   | LITFLOAT of (float)
8   | LITSTRING of (string)
9   | ID of (string)
10  | APAR
11  | ACOL
12  | ACHA
13  | FPAR
14  | FCOL
15  | FCHA
16  | VIRG
17  | SOMA
18  | SUBTRAI
19  | DIVIDE
```


4.1

```
20 | MULTIPLICA
21 | DOIS_PONTOS
22 | MENOR
23 | MAIOR
24 | MENOR_IGUAL
25 | MAIOR_IGUAL
26 | IGUAL
27 | DIFERENTE
28 | AND
29 | OR
30 | IF
31 | ELSE_IF
32 | ELSE
33 | NOT
34 | TIPO_INT
35 | TIPO_FLOAT
36 | TIPO_STRING
37 | TIPO_BOOL
38 | TIPO_VOID
39 | SETA
40 | TRUE
41 | FALSE
42 | FOR
43 | WHILE
44 | IN
45 | IS
46 | FROM
47 | RANGE
48 | PRINT
49 | INPUT
50 | INT_PARSE
51 | E
52 | ATRIB
53 | ARTIB_SOMA
54 | ARTIB_SUB
55 | ARTIB_MULT
56 | ARTIB_DIV
57 | ARTIB_MOD
58 | MODULO
59 | VIRGULA
60 | PONTO_VIRGULA
61 | PONTO
62 | RETURN
63 | DEF
64 | EOF
65 (* Os tokens a seguir são importantes para o pré processador *)
66 | Linha of (int * int * token list)
67 | INDENTA
68 | DEDENTA
69 | NOVALINHA
70
71 (* contador de nivel de parentizacao - utilizado para identacao *)
72 let nivel_par = ref 0
73
74 (* incrementa a contagem de linhas *)
75 let incr_nlinha lexbuf =
76   let pos = lexbuf.lex_curr_p in
77   lexbuf.lex_curr_p <- { pos with
78     pos_lnum = pos.pos_lnum + 1;
```

```

79             pos_bol = pos.pos_cnum;
80         }
81
82     (* imprime mensagem de erro *)
83     let msg_erro lexbuf c =
84         let pos = lexbuf.lex_curr_p in
85         let lin = pos.pos_lnum
86         and col = pos.pos_cnum - pos.pos_bol - 1 in
87         sprintf "%d-%d: Caracter Desconhecido %c" lin col c
88
89
90     (* cria tabela hasg *)
91     let cria_tab_hash iniciais =
92         let tbl = Hashtbl.create (List.length iniciais) in
93         List.iter (fun (chave, valor) -> Hashtbl.add tbl chave valor)
94             iniciais;
95         tbl
96
97     (* palavras reservadas *)
98     let plv_res =
99         cria_tab_hash
100         [
101             ("def", DEF);
102             ("else", ELSE );
103             ("for", FOR);
104             ("if", IF);
105             ("in", IN);
106             ("not", NOT);
107             ("and", AND);
108             ("or", OR);
109             ("is", IS);
110             ("from", FROM);
111             ("return", RETURN);
112             ("while", WHILE);
113             ("range", RANGE);
114             ("print", PRINT);
115             ("int", TIPO_INT);
116             ("None", TIPO_VOID);
117             ("str", TIPO_STRING);
118             ("float", TIPO_FLOAT);
119             ("bool", TIPO_BOOL);
120             ("raw_input", INPUT);
121             ("int_parse", INT_PARSE)
122         ]
123
124     (* Valores booleanos sao armazenados como 1 para true e 0 para false. *)
125     (* Operacoes com booleanos sao transformadas em operacoes com inteiros
126        *)
127     let booleano nbool =
128         match nbool with
129         | "True" -> 1
130         | "False" -> 0
131         | _ -> failwith "Erro: nao eh valor booleano"
132     }
133
134     (* definicoes *)
135     let digito = ['0' - '9']
136     let letra = ['a'-'z' 'A'-'Z']
137     let id = letra ( letra | digito | '_' )*)

```

4.1

```
137 let comentario = '#' [^ '\n']*
138 let comentario2 = '"' ['\n']* '|' ['\n']* [^ '\n']*
139 let linha_em_branco = [' '\t']* comentario | [' '\t']* comentario
140 let restante = [^ '\t' '\n'] [^ '\n']+
141 let boolean = "True" | "False"
142 let strings = '"' id* digito* '"' | '"' id* digito* '"'
143 let floats = digito+ '.' digito+
144 let neg = '-' digito+
145
146 (* regras para identificar indentacao para gerar tokens de abre e fecha
    escopo *)
147 rule preprocessor indentacao = parse
148   linha_em_branco { preprocessor 0 lexbuf } (* ignora brancos *)
149   | [' '\t']+ '\n' { incr_nlinha lexbuf;
150                     preprocessor 0 lexbuf } (* ignora brancos *)
151   | ' ' { preprocessor (indentacao + 1) lexbuf }
152   | '\t' { let nova_ind = indentacao + 8 - (indentacao mod
153                                             8) in
154             preprocessor nova_ind lexbuf }
154   | '\n' { incr_nlinha lexbuf;
155             preprocessor 0 lexbuf }
156   | eof { nivel_par := 0; EOF }
157   | restante as linha {
158                       let rec tokenize lexbuf =
159                         let tok = token lexbuf in
160                         match tok with
161                         EOF -> []
162                         | _ -> tok :: tokenize lexbuf in
163                         let toks = tokenize (Lexing.from_string linha)
164                           in
165                         Linha(indentacao,!nivel_par, toks)
166                       }
167 (* identificacao dos tokens *)
168 and token = parse
169 | ' '
170 | '\t'
171 | comentario { token lexbuf }
172 | comentario2 { comentario_bloco 0 lexbuf; }
173 | digito+ as numint { let num = int_of_string numint in LITINT num }
174 | neg as numNeg { let num = int_of_string numNeg in LITINT num }
175 | digito+ '.' digito+ as numfloat { let num = float_of_string numfloat in
176                                     LITFLOAT num }
177 | boolean as nbool { LITINT (booleano nbool) }
178 | floats as numfloat { LITFLOAT (float_of_string numfloat) }
179 | id as palavra { try Hashtbl.find plv_res palavra
180                  with Not_found -> ID (palavra) }
181 | '"' { let buffer = Buffer.create 1 in
182          LITSTRING (cadeia buffer lexbuf) }
183 | '=' { ATRIB }
184 | '+' { SOMA }
185 | '-' { SUBTRAI }
186 | '/' { DIVIDE }
187 | '*' { MULTIPLICA }
188 | ':' { DOIS_PONTOS }
189 | '(' { APAR }
190 | ')' { FPAR }
191 | ';' { PONTO_VIRGULA }
```

```

192 | ','          { VIRGULA }
193 | "=="        { IGUAL }
194 | '>'         { MAIOR }
195 | '<'         { MENOR }
196 | ">="       { MAIOR_IGUAL }
197 | "<="       { MENOR_IGUAL }
198 | "!="       { DIFERENTE }
199 | "and"      { AND }
200 | "or"       { OR }
201 | "if"       { IF }
202 | "else"     { ELSE }
203 | "not"      { NOT }
204 | "int"      { TIPO_INT }
205 | "float"    { TIPO_FLOAT }
206 | "str"      { TIPO_STRING }
207 | "bool"     { TIPO_BOOL }
208 | "None"     { TIPO_VOID }
209 | "def"      { DEF }
210 | "->"       { SETA } (* tipo funcao *)
211 | "True"     { TRUE }
212 | "False"    { FALSE }
213 | "for"      { FOR }
214 | "while"    { WHILE }
215 | "in"       { IN }
216 | "range"    { RANGE }
217 | "int_parse" { INT_PARSE }
218 | "print"    { PRINT }
219 | "+="       { ARTIB_SOMA }
220 | "-="       { ARTIB_SUB }
221 | "*="       { ARTIB_MULT }
222 | "/="       { ARTIB_DIV }
223 | "%="       { ARTIB_MOD }
224 | "%"        { MODULO }
225 | '('        { incr(nivel_par); APAR }
226 | '['        { incr(nivel_par); ACOL }
227 | '{'        { incr(nivel_par); ACHA }
228 | '}'        { decr(nivel_par); FPAR }
229 | ']'        { decr(nivel_par); FCOL }
230 | '}'        { decr(nivel_par); FCHA }
231 | ':'        { DOIS_PONTOS }
232 | ','        { VIRG }
233 | ';'        { PONTO_VIRGULA }
234 | '.'        { PONTO }
235 | eof        { EOF }
236 | _ as c     { failwith (msg_erro lexbuf c); }
237
238 (* para criar comentario de bloco *)
239 and comentario_bloco n = parse
240 | ''' ''' ''' | """ """ """ { if n=0 then token lexbuf
241 |                               else comentario_bloco (n - 1) lexbuf }
242 | ''' ''' ''' | """ """ """ { comentario_bloco (n + 1) lexbuf; }
243 | _ { comentario_bloco n lexbuf }
244 | eof { failwith "Comentário não fechado" }
245
246 (* para criar cadeias de strings *)
247 and cadeia buffer = parse
248 | ''' { Buffer.contents buffer }
249 | "\\t" { Buffer.add_char buffer '\t'; cadeia buffer lexbuf }
250 | "\\n" { Buffer.add_char buffer '\n'; cadeia buffer lexbuf }

```

```

251 | '\\\'' '\"' { Buffer.add_char buffer '"'; cadeia buffer lexbuf }
252 | '\\\'' '\\\'' { Buffer.add_char buffer '\\\''; cadeia buffer lexbuf }
253 | eof      { failwith "string não foi fechada" }
254 | _ as c   { Buffer.add_char buffer c; cadeia buffer lexbuf }

```

Listagem 4.3: *pre_processador.ml*

```

1 open Lexico
2 open Printf
3
4 (* Pré processa o arquivo gerando os tokens de indenta e dedenta *)
5
6 let preprocessa lexbuf =
7   let pilha = Stack.create ()
8   and npars = ref 0 in
9   let _ = Stack.push 0 pilha in
10  let off_side toks nivel =
11    let _ = printf "Nivel: %d\n" nivel in
12    if !npars != 0 (* nova linha entre parenteses *)
13    then toks      (* nao faz nada *)
14    else if nivel > Stack.top pilha
15         then begin
16           Stack.push nivel pilha;
17           INDENTA :: toks
18         end
19    else if nivel = Stack.top pilha
20         then toks
21    else begin
22      let prefixo = ref toks in
23      while nivel < Stack.top pilha do
24        ignore (Stack.pop pilha);
25        if nivel > Stack.top pilha
26        then failwith "Erro de indentacao"
27        else prefixo := DEDENTA :: !prefixo
28      done;
29      !prefixo
30    end
31  in
32
33  let rec dedenta sufixo =
34    if Stack.top pilha != 0
35    then let _ = Stack.pop pilha in
36         dedenta (DEDENTA :: sufixo)
37    else sufixo
38  in
39  let rec get_tokens () =
40    let tok = Lexico.preprocessador 0 lexbuf in
41    match tok with
42    | Linha(nivel,npars,toks) ->
43      let new_toks = off_side toks nivel in
44      npars := npars;
45      new_toks @ (if npars = 0
46                  then NOVALINHA :: get_tokens ()
47                  else get_tokens ())
48    | _ -> dedenta []
49  in get_tokens ()
50
51
52 (* Chama o analisador léxico *)

```

```

53 let lexico =
54   let tokbuf = ref None in
55   let carrega lexbuf =
56     let toks = preprocessa lexbuf in
57     (match toks with
58      tok::toks ->
59        tokbuf := Some toks;
60        tok
61      | [] -> print_endline "EOF";
62        EOF)
63   in
64   fun lexbuf ->
65   match !tokbuf with
66   Some tokens ->
67     (match tokens with
68      tok::toks ->
69        tokbuf := Some toks;
70        tok
71      | [] -> carrega lexbuf)
72   | None -> carrega lexbuf

```

Listagem 4.4: carregador.ml

```

1 (* Para compilar:
2   ocamllex lexico.mll
3   ocamlc -c lexico.ml
4   ocamlc -c pre_processador.ml
5
6   Para usar:
7   rlwrap ocaml
8
9   #use "carregador.ml";;
10  lex "teste.py";;
11 *)
12
13 #load "lexico.cmo"
14 #load "pre_processador.cmo"
15
16 type nome_arq = string
17 type tokens = Lexico.token list
18
19 let rec tokens lexbuf =
20   let tok = Pre_processador.lexico lexbuf in
21   match tok with
22   | Lexico.EOF -> ([Lexico.EOF]:tokens)
23   | _ -> tok :: tokens lexbuf
24 ;;
25
26 let lexico str =
27   let lexbuf = Lexing.from_string str in
28   tokens lexbuf
29 ;;
30
31 let lex (arq:nome_arq) =
32   let ic = open_in arq in
33   let lexbuf = Lexing.from_channel ic in
34   let toks = tokens lexbuf in
35   let _ = close_in ic in
36   toks

```

4.1.1 Teste

Para compilar fazemos:

Listagem 4.5: Terminal

```

1 $ ocamllex lexico.mll
2 $ ocamlc -c lexico.ml
3 $ ocamlc -c pre_processador.ml
4 $ rlwrap ocaml
5      OCaml version 4.07.0
6
7 # #use "carregador.ml";;
8 type nome_arq = string
9 type tokens = Lexico.token list
10 val tokens : Lexing.lexbuf -> tokens = <fun>
11 val lexico : string -> tokens = <fun>
12 val lex : nome_arq -> tokens = <fun>
13 # lex "tests/micro10.py";;
14 Linha(identacao=0,nivel_par=0)
15 Nivel: 0
16 Linha(identacao=4,nivel_par=0)
17 Nivel: 4
18 Linha(identacao=4,nivel_par=0)
19 Nivel: 4
20 Linha(identacao=4,nivel_par=0)
21 Nivel: 4
22 Linha(identacao=4,nivel_par=0)
23 Nivel: 4
24 Linha(identacao=4,nivel_par=0)
25 Nivel: 4
26 Linha(identacao=4,nivel_par=0)
27 Nivel: 4
28 Linha(identacao=0,nivel_par=0)
29 Nivel: 0
30 Linha(identacao=4,nivel_par=0)
31 Nivel: 4
32 Linha(identacao=8,nivel_par=0)
33 Nivel: 8
34 Linha(identacao=4,nivel_par=0)
35 Nivel: 4
36 Linha(identacao=8,nivel_par=0)
37 Nivel: 8
38 EOF
39 - : tokens =
40 [Lexico.DEF; Lexico.ID "micro10"; Lexico.APAR; Lexico.FPAR; Lexico.DPONTOS
    ;
41 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "numero"; Lexico.ATRIB;
42 Lexico.LITINT 0; Lexico.NOVALINHA; Lexico.ID "fat"; Lexico.ATRIB;
43 Lexico.LITINT 0; Lexico.NOVALINHA; Lexico.PRINT; Lexico.APAR;
44 Lexico.LITSTRING "Digite um numero: "; Lexico.FPAR; Lexico.NOVALINHA;
45 Lexico.ID "numero"; Lexico.ATRIB; Lexico.INT; Lexico.APAR; Lexico.INPUT;
46 Lexico.APAR; Lexico.FPAR; Lexico.FPAR; Lexico.NOVALINHA; Lexico.ID "fat";
47 Lexico.ATRIB; Lexico.ID "fatorial"; Lexico.APAR; Lexico.ID "numero";
48 Lexico.FPAR; Lexico.NOVALINHA; Lexico.PRINT; Lexico.APAR;
49 Lexico.LITSTRING "O fatorial de "; Lexico.MAIS; Lexico.STR; Lexico.APAR;
50 Lexico.ID "numero"; Lexico.FPAR; Lexico.MAIS; Lexico.LITSTRING " e ";
51 Lexico.MAIS; Lexico.STR; Lexico.APAR; Lexico.ID "fat"; Lexico.FPAR;
52 Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.DEF;

```

```

53 Lexico.ID "fatorial"; Lexico.APAR; Lexico.ID "n"; Lexico.FPAR;
54 Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.INDENTA; Lexico.IF; Lexico.ID "n
    ";
55 Lexico.MENORIGUAL; Lexico.LITINT 0; Lexico.DPONTOS; Lexico.NOVALINHA;
56 Lexico.INDENTA; Lexico.RETURN; Lexico.LITINT 1; Lexico.NOVALINHA;
57 Lexico.DEDENTA; Lexico.ELSE; Lexico.DPONTOS; Lexico.NOVALINHA;
58 Lexico.INDENTA; Lexico.RETURN; Lexico.APAR; Lexico.ID "n"; Lexico.VEZES;
59 Lexico.ID "fatorial"; Lexico.APAR; Lexico.ID "n"; Lexico.MENOS;
60 Lexico.LITINT 1; Lexico.FPAR; Lexico.FPAR; Lexico.NOVALINHA; Lexico.
    DEDENTA;
61 Lexico.DEDENTA; Lexico.EOF]

```

4.2 Analisador Sintático

Listagem 4.6: sintatico.mly

```

1 %{
2   open Ast
3   open Sast
4 %}
5
6 %token <int * int * token list> Linha
7 %token <float * Lexing.position> LITFLOAT
8 %token <string *Lexing.position > ID
9 %token <string *Lexing.position > LITSTRING
10 %token <int * Lexing.position> LITINT
11 %token <bool * Lexing.position> LITBOOL
12 %token <Lexing.position> DEF SETA DPONTOS
13 %token <Lexing.position> VIRG
14 %token <Lexing.position> ATRIB MAIOR MAIORIGUAL MENOR MENORIGUAL DIFERENTE
    IGUALDADE
15 %token <Lexing.position> OU E NOT MAIS MENOS DIVIDIDO VEZES MODULO
16 %token <Lexing.position> APAR FPAR
17 %token <Lexing.position> PRINT
18 %token <Lexing.position> INPUTI INPUTF INPUTS
19 %token <Lexing.position> WHILE FOR IN RANGE
20 %token <Lexing.position> IF ELIF ELSE
21 %token <Lexing.position> RETURN
22 %token <Lexing.position> NONE
23 %token <Lexing.position> STR
24 %token <Lexing.position> INT
25 %token <Lexing.position> FLOAT
26 %token <Lexing.position> BOOL
27 %token INDENTA DEDENTA NOVALINHA EOF
28
29 %left OU
30 %left E
31 %left IGUALDADE DIFERENTE
32 %left MAIOR MAIORIGUAL MENOR MENORIGUAL
33 %left MAIS MENOS
34 %left VEZES DIVIDIDO MODULO
35
36 %nonassoc unary_minus
37
38 %start <Sast.expressao Ast.programa> programa
39

```


4.2

```
40 %%
41
42 programa: ins=instrucao*
43     EOF
44     {Programa ins }
45
46 funcao:
47     DEF nome= ID
48     APAR args = separated_list(VIRG, parametro) FPAR
49     SETA retorno = tipo DPONTOS NOVALINHA
50     INDENTA
51     cmd = comandos
52     DEDENTA
53     {
54         Funcao {
55             fn_nome = nome;
56             fn_tiporet = retorno;
57             fn_formais = args;
58             fn_corpo = cmd
59         }
60     }
61
62
63 parametro:
64     | id = ID DPONTOS tp = tipo { (id,tp) }
65
66
67 /*esse eh o meu stm_block */
68 instrucao:
69     | func = funcao          { func }
70     | cmd = comando          { ACMD(cmd) }
71
72
73 comandos:
74     cmd = comando+ { cmd }
75
76 /*esse eh o meu stm_list*/
77 comando:
78     | stm = atribuicao          { stm }
79     | stm = chamadafuncao      { stm }
80     | stm = loopWhile          { stm }
81     | stm = condicaoIF          { stm }
82     | stm = loopFOR            { stm }
83     | stm = print              { stm }
84     | stm = retorno            { stm }
85     | stm = leiai NOVALINHA    { stm }
86     | stm = leiaf NOVALINHA    { stm }
87     | stm = leias NOVALINHA    { stm }
88     ;
89
90 retorno:
91     | RETURN expr = exprLogicoAritmetica? NOVALINHA { RETORNO(expr) }
92     ;
93
94 print:
95     | PRINT exprla = exprLogicoAritmetica NOVALINHA {PRINT(exprla) }
96     ;
97 /*a sacada eh emcapsular tudo dentro de expressao*/
98
```

```

99 chamadafuncao:
100   | exp=chamada NOVALINHA { CHAMADADEFUNCAO(exp) }
101   ;
102
103 chamada : nome=ID APAR args=separated_list(VIRG, exprLogicoAritmetica)
         FPAR { EXPCALL (nome, args) }
104
105 condicaoIF:
106   | IF exprla= exprLogicoAritmetica DPONTOS NOVALINHA
107     INDENTA stm=comandos DEDENTA
108     cee = condicaoELIFELSE?
109     { CONDICAIF(exprla,stm,cee) }
110
111
112 condicaoELIFELSE:
113   | ELIF exprla = exprLogicoAritmetica DPONTOS NOVALINHA INDENTA stm =
         comandos DEDENTA condee = condicaoELIFELSE? { CONDICAIF (exprla,stm,
         condee) }
114   | ELSE DPONTOS NOVALINHA INDENTA stm=comandos DEDENTA {CONDICAOelifElse(
         stm ) }
115   ;
116
117 atribuicao: id = ID ATRIB exprla = exprLogicoAritmetica NOVALINHA {
         ATRIBUICAO (EXPVAR id , exprla) }
118
119 leiai: INPUTI exp=exprLogicoAritmetica { LEIAI exp }
120 leiaf: INPUTF exp=exprLogicoAritmetica { LEIAF exp }
121 leias: INPUTS exp=exprLogicoAritmetica { LEIAS exp }
122
123 loopFOR:
124   | FOR expid=exprLogicoAritmetica IN RANGE APAR exprcomeco =
         exprLogicoAritmetica VIRG exprfim = exprLogicoAritmetica FPAR DPONTOS
         NOVALINHA INDENTA stm = comandos DEDENTA { FORLOOP(expid,exprcomeco
         ,exprfim,stm)}
125   ;
126
127 loopWhile: WHILE exprla = exprLogicoAritmetica DPONTOS NOVALINHA INDENTA
         stm = comandos DEDENTA { WHILELOOP(exprla,stm) }
128
129 exprLogicoAritmetica:
130   | f = chamada { f }
131   | id = ID { EXPVAR(id) }
132   | i = LITINT { EXPINT(i) }
133   | s = LITSTRING { EXPSTRING(s) }
134   | f = LITFLOAT { EXPFLOAT(f) }
135   | b = LITBOOL { EXPBOOL (b) }
136   | op=opU e=exprLogicoAritmetica %prec unary_minus { EXPOPU (op,e)
         }
137   | el=exprLogicoAritmetica op = opB e2 = exprLogicoAritmetica { EXPOPB (
         op,el,e2) }
138   | APAR e=exprLogicoAritmetica FPAR { e }
139   ;
140
141 tipo:
142   | BOOL { BOOLEAN }
143   | INT { INTEIRO }
144   | FLOAT { REAL }
145   | NONE { NONE }
146   | STR { STRING }

```

```

147 ;
148
149 %inline opB:
150 | pos = MAIS          { (ADICAO, pos) }
151 | pos = MENOS         { (SUBTRACAO, pos) }
152 | pos = VEZES         { (MULTIPLICACAO, pos) }
153 | pos = DIVIDIDO      { (DIVISAO, pos) }
154 | pos = MODULO        { (MOD, pos) }
155 | pos = IGUALDADE     { (EHIGUAL, pos) }
156 | pos = MAIOR         { (MAIORQ, pos) }
157 | pos = MAIORIGUAL    { (MAIORIGUALQ, pos) }
158 | pos = MENOR         { (MENORQ, pos) }
159 | pos = MENORIGUAL    { (MENORIGUALQ, pos) }
160 | pos = DIFERENTE     { (EHDIFERENTE, pos) }
161 | pos = E             { (AND, pos) }
162 | pos = OU            { (OR, pos) }
163 ;
164
165 %inline opU:
166 | pos = NOT           { (NEGACAO, pos) }
167 | pos = MENOS         { (SUBTRACAO, pos) }
168 ;

```

Listagem 4.7: ast.ml

```

1 (* The type of the abstract syntax tree (AST). *)
2
3 type identificador = string
4 (*posicao no arquivo*)
5 type 'a pos = 'a * Lexing.position
6
7 type 'expr programa = Programa of 'expr instrucoes
8 and 'expr comandos = 'expr comando list
9 and 'expr instrucoes = 'expr instrucao list
10 and 'expr expressoes = 'expr list
11 and 'expr instrucao =
12   Funcao of 'expr decfn
13   | ACMD of 'expr comando
14 and 'expr decfn = {
15   fn_nome: identificador pos;
16   fn_tiporet: tipo;
17   fn_formais: (identificador pos * tipo) list;
18   fn_corpo: 'expr comandos
19 }
20
21 and tipo =
22   BOOLEAN
23   | INTEIRO
24   | REAL
25   | NONE
26   | STRING
27
28 and 'expr comando =
29   ATRIBUICAO of 'expr * 'expr
30   | CONDICAOWIF of 'expr * ('expr comando) list * ('expr comando
31   option)
32   | CONDICAOWElse of 'expr comandos
33   | WHILELOOP of 'expr * ('expr comando) list
34   | FORLOOP of 'expr * 'expr * 'expr * ('expr comando) list

```

```

34         | PRINT of 'expr
35         | RETORNO of 'expr option
36         | LEIAI of 'expr
37         | LEIAF of 'expr
38         | LEIAS of 'expr
39         | CHAMADADEFUNCAO of 'expr
40
41 and operador =
42     ADICAO
43     | SUBTRACAO
44     | MULTIPLICACAO
45     | DIVISAO
46     | MOD
47     | EHIGUAL
48     | MAIORQ
49     | MAIORIGUALQ
50     | MENORQ
51     | MENORIGUALQ
52     | EHDIFERENTE
53     | AND
54     | OR
55     | NEGACAO

```

4.2.1 Teste

Para compilar fazemos:

Listagem 4.8: Terminal

```

1 $ ocamlbuild -use-ocamlfind -use-menhir -menhir "menhir --table" -package
   menhirLib sintaticoTeste.byte
2 Finished, 40 targets (37 cached) in 00:00:00.
3 $ rlwrap ocaml
4     OCaml version 4.07.0
5
6 Findlib has been successfully loaded. Additional directives:
7   #require "package";;          to load a package
8   #list;;                      to list the available packages
9   #camlp4o;;                   to load camlp4 (standard syntax)
10  #camlp4r;;                   to load camlp4 (revised syntax)
11  #predicates "p,q,...";;      to set these predicates
12  Topfind.reset();;           to force that packages will be reloaded
13  #thread;;                   to enable threads
14
15 /Users/tatianefx/.opam/default/lib/menhirLib: added to search path
16 /Users/tatianefx/.opam/default/lib/menhirLib/menhirLib.cmo: loaded
17 # parse_arq "tests/micro10.py";;

```

Listagem 4.9: testeSintatico.txt

```

1 EOF
2 - : Sast.expressao Ast.programa option option =
3 Some
4   (Some
5     (Programa
6       [Funcao
7         {fn_nome =
8           ("main",

```

4.2

```
9      {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum =
10        4});
11  fn_tiporet = NONE; fn_formais = [];
12  fn_corpo =
13    [ATRIBUICAO
14      (Sast.EXPVAR
15        ("numero",
16          {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
17            pos_cnum = 0}),
18        Sast.EXPINT
19          (0,
20            {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum =
21              8})));
22    ATRIBUICAO
23      (Sast.EXPVAR
24        ("fat",
25          {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
26            pos_cnum = 0}),
27        Sast.EXPINT
28          (0,
29            {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum =
30              6})));
31    PRINT
32      (Sast.EXPSTRING
33        ("Digite um numero: ",
34          {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
35            pos_cnum = 25})));
36    LEIAI
37      (Sast.EXPVAR
38        ("numero",
39          {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
40            pos_cnum = 7})));
41    ATRIBUICAO
42      (Sast.EXPVAR
43        ("fat",
44          {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
45            pos_cnum = 0}),
46        Sast.EXPCALL
47          (("fatorial",
48            {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
49              pos_cnum = 6}),
50          [Sast.EXPVAR
51            ("numero",
52              {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
53                pos_cnum = 15}]]));
54    PRINT
55      (Sast.EXPSTRING
56        ("O fatorial eh ",
57          {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
58            pos_cnum = 21})));
59    PRINT
60      (Sast.EXPVAR
61        ("fat",
62          {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
63            pos_cnum = 6}]]]);
64  Funcao
65    {fn_nome =
66      ("fatorial",
67        {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum =
```

```

        4});
65 fn_tiporet = INTEIRO;
66 fn_formais =
67 [ ("n",
68   {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum =
        13}),
69   INTEIRO)];
70 fn_corpo =
71 [CONDICAOIF
72   (Sast.EXPOPB
73     ((MENORIGUALQ,
74       {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
75        pos_cnum = 5}),
76     Sast.EXPVAR
77       ("n",
78       {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
79        pos_cnum = 3}),
80     Sast.EXPINT
81       (0,
82       {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
83        pos_cnum = 8})),
84   [RETORNO
85     (Some
86       (Sast.EXPINT
87         (1,
88         {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
89          pos_cnum = 7}))),],
90   Some
91     (CONDICAOElifElse
92       [RETORNO
93         (Some
94           (Sast.EXPOPB
95             ((MULTIPLICACAO,
96               {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
97                pos_cnum = 9}),
98             Sast.EXPVAR
99               ("n",
100              {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
101               pos_cnum = 7}),
102             Sast.EXPCALL
103               (("fatorial",
104                {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
105                 pos_cnum = 11}),
106              [Sast.EXPOPB
107                ((SUBTRACAO,
108                  {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol =
109                   0;
110                   pos_cnum = 22}),
111                Sast.EXPVAR
112                  ("n",
113                  {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol =
114                   0;
115                   pos_cnum = 20}),
116                Sast.EXPINT
117                  (1,
118                  {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol =
119                   0;
120                   pos_cnum = 24}})))])))]))]]);

```

```

119         (CHAMADADEFUNCAO
120         (Sast.EXPCALL
121         (("main",
122         {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum =
123         0})),
124         [ ]))))))

```

4.3 Analisador Semântico

Listagem 4.10: semantico.ml

```

1 module Amb = Ambiente
2 module A = Ast
3 module S = Sast
4 module T = Tast
5
6 let rec posicao exp =
7   let open S in
8   match exp with
9   | EXPVAR      (_,pos)      -> pos
10  | EXPINT      (_,pos)      -> pos
11  | EXPSTRING   (_,pos)      -> pos
12  | EXPBOOL     (_,pos)      -> pos
13  | EXPFLOAT    (_,pos)      -> pos
14  | EXPOPB      ((_,pos),_,_) -> pos
15  | EXPOPU      ((_,pos),_,_) -> pos
16  | EXPCALL     ((_,pos),_,_) -> pos
17
18
19 type classe_op = Aritmetico | Relacional | Logico
20
21 let classifica op =
22   let open A in
23   match op with
24   | ADICAO
25   | SUBTRACAO
26   | MULTIPLICACAO
27   | DIVISAO
28   | MOD          -> Aritmetico
29   | MAIORQ
30   | MENORQ
31   | MAIORIGUALQ
32   | MENORIGUALQ
33   | EHIGUAL
34   | EHDIFERENTE -> Relacional
35   | AND
36   | NEGACAO
37   | OR          -> Logico
38
39 let msg_erro_pos pos msg =
40   let open Lexing in
41   let lin = pos.pos_lnum
42   and col = pos.pos_cnum - pos.pos_bol - 1 in
43   Printf.sprintf "Semantico -> linha %d, coluna %d: %s" lin col msg
44
45 (* argumento nome é do tipo S.tipo *)

```

```

46 let msg_erro nome msg =
47   let pos = snd nome in
48   msg_erro_pos pos msg
49
50 let nome_tipo t =
51   let open A in
52   match t with
53     INTEIRO      -> "inteiro"
54   | STRING       -> "string"
55   | BOOLEAN      -> "booleano"
56   | REAL         -> "real"
57   | NONE         -> "vazio"
58
59 let mesmo_tipo pos msg tinf tdec =
60   if tinf <> tdec then
61     let msg = Printf.sprintf msg (nome_tipo tinf) (nome_tipo tdec) in
62     failwith (msg_erro_pos pos msg)
63
64 let rec infere_exp amb exp =
65   match exp with
66
67 | S.EXPINT i -> (T.EXPINT (fst i, A.INTEIRO ), A.INTEIRO )
68 | S.EXPSTRING s -> (T.EXPSTRING (fst s, A.STRING ), A.STRING )
69 | S.EXPBOOL b -> (T.EXPBOOL (fst b, A.BOOLEAN ), A.BOOLEAN )
70 | S.EXPFLOAT f -> (T.EXPFLOAT (fst f, A.REAL), A.REAL)
71 | S.EXPVAR variavel ->
72   let nome = fst variavel in
73   (try begin
74     (match (Amb.busca amb nome) with
75     | Amb.EntVar tipo -> (T.EXPVAR (nome, tipo), tipo)
76     | Amb.EntFun _ ->
77       let msg = "Nome de funcao usado como nome de variavel: "^
78         nome in
79       failwith (msg_erro variavel msg))
80   end with Not_found ->
81     let msg = "Variavel "^nome^" nao declarada" in
82     failwith (msg_erro variavel msg))
83 | S.EXPOPB (op, exp_esq, exp_dir) ->
84   let (esq, tesq) = infere_exp amb exp_esq
85   and (dir, tdir) = infere_exp amb exp_dir in
86   let verifica_aritmetico () =
87     (match tesq with
88     | A.INTEIRO
89     | A.REAL ->
90       let _ = mesmo_tipo (snd op)
91         "Operando esquerdo do tipo %s, mas o tipo do direito
92         eh %s"
93         tesq tdir
94       in tesq (* Tipo inferido para a operação *)
95   | demais ->
96     let msg = "O tipo "^
97       (nome_tipo demais)^
98       " nao eh valido em um operador aritmético" in
99     failwith (msg_erro op msg))
100   and verifica_relacional () =
101     (match tesq with
102     | A.INTEIRO
103     | A.STRING
104     | A.BOOLEAN

```



```

103 | A.REAL ->
104 |   (let _ = mesmo_tipo (snd op)
105 |     "Operando esquerdo do tipo %s, mas o tipo do direito
106 |       eh %s"
107 |     tesq tdir
108 |     in A.BOOLEAN) (* Tipo inferido para a operação *)
109 | demais ->
110 |   (let msg = "O tipo " ^
111 |     (nome_tipo demais) ^
112 |     " nao eh valido em um operador relacional" in
113 |     failwith (msg_erro op msg)))
114 and verifica_logico () =
115 |   (match tesq with
116 |   | A.BOOLEAN ->
117 |     (let _ = mesmo_tipo (snd op)
118 |       "Operando esquerdo do tipo %s, mas o tipo do direito
119 |         eh %s"
120 |       tesq tdir
121 |       in A.BOOLEAN) (* Tipo inferido para a operação *)
122 |   | demais ->
123 |     (let msg = "O tipo " ^
124 |       (nome_tipo demais) ^
125 |       " nao eh valido em um operador logico" in
126 |       failwith (msg_erro op msg))
127 |   in
128 |   let oper = fst op in
129 |   let tinf =
130 |     (match (classifica oper) with
131 |     | Aritmetico -> verifica_aritmetico ()
132 |     | Relacional -> verifica_relacional ()
133 |     | Logico -> verifica_logico ())
134 |   in (T.EXPOPB ((oper, tinf), (esq, tesq), (dir, tdir)), tinf)
135 | S.EXPOPU (op, exp) ->
136 |   let (exp, texp) = infere_exp amb exp in
137 |   let verifica_not () =
138 |     match texp with
139 |     | A.BOOLEAN ->
140 |       (let _ = mesmo_tipo (snd op)
141 |         "O operando eh do tipo %s, mas espera-se um %s"
142 |         texp A.BOOLEAN
143 |         in A.BOOLEAN)
144 |     | demais ->
145 |       (let msg = "O tipo " ^
146 |         (nome_tipo demais) ^
147 |         " nINTEIROao eh valido para o operador not" in
148 |         failwith (msg_erro op msg))
149 |   and verifica_negativo () =
150 |     match texp with
151 |     | A.REAL ->
152 |       (let _ = mesmo_tipo (snd op)
153 |         "O operando eh do tipo %s, mas espera-se um %s"
154 |         texp A.REAL
155 |         in A.REAL)
156 |     | A.INTEIRO ->
157 |       (let _ = mesmo_tipo (snd op)
158 |         "O operando eh do tipo %s, mas espera-se um %s"
159 |         texp A.INTEIRO
160 |         in A.INTEIRO)
161 |     | demais ->

```

```

160         let msg = "O tipo " ^
161             (nome_tipo demais) ^
162             " nao eh valido para o operador menos" in
163             failwith (msg_erro op msg)
164     in
165     let oper = fst op in
166     let tinf =
167         let open A in
168             match oper with
169             | NEGACAO -> verifica_not ()
170             | SUBTRACAO -> verifica_negativo ()
171             | demais->
172                 let msg = "Operador unario indefinido"
173                 in failwith (msg_erro op msg)
174     in (T.EXPOPU ((oper, tinf), (exp, texp)), tinf)
175 | S.EXPCALL (nome, args) ->
176     let rec verifica_parametros ags ps fs =
177         match (ags, ps, fs) with
178         | (a::ags), (p::ps), (f::fs) ->
179             let _ = mesmo_tipo (posicao a)
180                 "O parametro eh do tipo %s mas deveria ser do tipo %s"
181                 p f
182             in verifica_parametros ags ps fs
183         | [], [], [] -> ()
184         | _ -> failwith (msg_erro nome "Numero incorreto de parametros")
185     in
186     let id = fst nome in
187     try
188         begin
189             let open Amb in
190                 match (Amb.busca amb id) with
191                 | Amb.EntFun {tipo_fn; formais} ->
192                     let targs = List.map (infere_exp amb) args
193                     and tformais = List.map snd formais in
194                     let _ = verifica_parametros args (List.map snd targs)
195                         tformais in
196                     (T.EXPCALL (id, (List.map fst targs), tipo_fn), tipo_fn)
197                 | Amb.EntVar _ -> (* Se estiver associada a uma variável,
198                                     falhe *)
199                     let msg = id ^ " eh uma variavel e nao uma funcao" in
200                     failwith (msg_erro nome msg)
201             end
202         with Not_found ->
203             let msg = "Nao existe a funcao de nome " ^ id in
204             failwith (msg_erro nome msg)
205     let rec verifica_cmd amb tiporet cmd =
206         let open A in
207             match cmd with
208             | CHAMADADEFUNCAO exp -> let (exp,tinf) = infere_exp amb exp in
209                 CHAMADADEFUNCAO exp
210             | PRINT exp -> let expt = infere_exp amb exp in PRINT (fst expt)
211             | WHILELOOP (cond, cmds) ->
212                 let (expCond, expT ) = infere_exp amb cond in
213                 let comandos_tipados =
214                     (match expT with
215                     | A.BOOLEAN -> List.map (verifica_cmd amb tiporet) cmds
216                     | _ -> let msg = "Condicao deve ser tipo Bool" in
217                         failwith (msg_erro_pos (posicao cond) msg))

```

```

216     in WHILELOOP (expCond,comandos_tipados)
217 | LEIAI exp ->
218     (match exp with
219     S.EXPVAR (id,pos) ->
220     (try
221     begin
222         (match (Amb.busca amb id) with
223         Amb.EntVar tipo ->
224             let expt = infere_exp amb exp in
225             let _ = mesmo_tipo pos
226                 "inputi com tipos diferentes: %s = %s"
227                 tipo (snd expt) in
228             LEIAI (fst expt)
229         | Amb.EntFun _ ->
230             let msg = "nome de funcao usado como nome de
231                 variavel: " ^ id in
232             failwith (msg_erro_pos pos msg) )
233     end
234     with Not_found ->
235         let _ = Amb.insere_local amb id A.INTEIRO in
236         let expt = infere_exp amb exp in
237         LEIAI (fst expt) )
238 )
239 | LEIAF exp ->
240     (match exp with
241     S.EXPVAR (id,pos) ->
242     (try
243     begin
244         (match (Amb.busca amb id) with
245         Amb.EntVar tipo ->
246             let expt = infere_exp amb exp in
247             let _ = mesmo_tipo pos
248                 "Inputf com tipos diferentes: %s = %s"
249                 tipo (snd expt) in
250             LEIAF (fst expt)
251         | Amb.EntFun _ ->
252             let msg = "nome de funcao usado como nome de
253                 variavel: " ^ id in
254             failwith (msg_erro_pos pos msg) )
255     end
256     with Not_found ->
257         let _ = Amb.insere_local amb id A.REAL in
258         let expt = infere_exp amb exp in
259         LEIAF (fst expt) )
260 )
261 | LEIAS exp ->
262     (match exp with
263     S.EXPVAR (id,pos) ->
264     (try
265     begin
266         (match (Amb.busca amb id) with
267         Amb.EntVar tipo ->
268             let expt = infere_exp amb exp in
269             let _ = mesmo_tipo pos
270                 "Inputs com tipos diferentes: %s = %s"
271                 tipo (snd expt) in
272             LEIAS (fst expt)

```

```

273         | Amb.EntFun _ ->
274             let msg = "nome de funcao usado como nome de
275                 variavel: " ^ id in
276                 failwith (msg_erro_pos pos msg) )
277         end
278     with Not_found ->
279         let _ = Amb.insere_local amb id A.STRING in
280         let expt = infere_exp amb exp in
281         LEIAS (fst expt) )
282     | _ -> failwith "Falha Inputs"
283 )
284 | ATRIBUICAO (elem, exp) ->
285     let (var1, tdir) = infere_exp amb exp in
286     ( match elem with
287     | S.EXPVAR (id,pos) ->
288         (try
289             begin
290                 (match (Amb.busca amb id) with
291                 | Amb.EntVar tipo ->
292                     let _ = mesmo_tipo pos
293                     "Atribuicao com tipos diferentes: %s = %s"
294                     tipo tdir in
295                     ATRIBUICAO (T.EXPVAR (id, tipo), var1)
296                 | Amb.EntFun _ ->
297                     let msg = "nome de funcao usado como nome de
298                         variavel: " ^ id in
299                     failwith (msg_erro_pos pos msg) )
300                 end
301             with Not_found ->
302                 let _ = Amb.insere_local amb id tdir in
303                 ATRIBUICAO (T.EXPVAR (id, tdir), var1))
304         | _ -> failwith "Falha CmdAtrib"
305     )
306 | RETORNO exp ->
307     (match exp with
308     (* Se a função não retornar nada, verifica se ela foi declarada como
309     void *)
310     None ->
311         let _ = mesmo_tipo (Lexing.dummy_pos)
312         "O tipo retornado eh %s mas foi declarado como %s"
313         NONE tiporet
314     in RETORNO None
315     | Some e ->
316         (* Verifica se o tipo inferido para a expressão de retorno confere
317         com o *)
318         (* tipo declarado para a função. *)
319         let (e1,tinf) = infere_exp amb e in
320         let _ = mesmo_tipo (posicao e)
321         "O tipo retornado eh %s mas foi declarado
322         como %s"
323         tinf tiporet
324         in RETORNO (Some e1)
325     )
326 | CONDICAOelifElse comandos ->
327     let comandos = List.map (verifica_cmd amb tiporet) comandos in
328     CONDICAOelifElse comandos

```

```

326 | CONDICAIOIF (teste, entao, senao) ->
327   let (testel,tinf) = infere_exp amb teste in
328   let _ = mesmo_tipo (posicao teste)
329       "O teste do if deveria ser do tipo %s e nao %s"
330       BOOLEAN tinf in
331   let entao1 = List.map (verifica_cmd amb tiporet) entao in
332   let senao1 =
333       match senao with
334       | None          -> None
335       | Some bloco -> let c = verifica_cmd amb tiporet bloco in Some c
336   in CONDICAIOIF (testel, entao1, senao1)
337 | FORLOOP (idt, int_de,int_ate,bloco) ->
338   let (idt1,tinf) = infere_exp amb idt in
339   let (int_del,tinf1) = infere_exp amb int_de in
340   let (int_atel,tinf2) = infere_exp amb int_ate in
341   (* O tipo inferido para o identificador deve ser int *)
342   let _ = mesmo_tipo (posicao idt)
343       "A variável deveria ser do tipo %s e nao %s"
344       INTEIRO tinf in
345   (* O tipo inferido para os ints devem ser inteiros *)
346   let _ = mesmo_tipo (posicao int_de)
347       "O comando DE deveria ser do tipo %s e nao %s"
348       INTEIRO tinf1 in
349   let _ = mesmo_tipo (posicao int_ate)
350       "O comando DE deveria ser do tipo %s e nao %s"
351       INTEIRO tinf2 in
352   (* Verifica a validade de cada comando do bloco *)
353   let blocol = List.map (verifica_cmd amb tiporet) bloco in
354   FORLOOP (idt1, int_del,int_atel,blocol)
355
356 and verifica_fun amb ast =
357   let open A in
358   match ast with
359   | Funcao {fn_nome; fn_tiporet; fn_formais; fn_corpo} ->
360     (* Estende o ambiente global, adicionando um ambiente local *)
361     let ambfn = Amb.novo_escopo amb in
362     (* Insere os parâmetros no novo ambiente *)
363     let insere_parametro (v,t) = Amb.insere_param ambfn (fst v) t in
364     let _ = List.iter insere_parametro fn_formais in
365     (* Verifica cada comando presente no corpo da função usando o novo
366        ambiente *)
367     let corpo_tipado = List.map (verifica_cmd ambfn fn_tiporet) fn_corpo
368     in
369     Funcao {fn_nome; fn_tiporet; fn_formais; fn_corpo = corpo_tipado}
370 | ACMD _ -> failwith "Instrução invalida"
371
372 let rec verifica_dup xs =
373   match xs with
374   | [] -> []
375   | (nome,t)::xs ->
376     let id = fst nome in
377     if (List.for_all (fun (n,t) -> (fst n) <> id) xs)
378     then (id, t) :: verifica_dup xs
379     else let msg = "Parametro duplicado " ^ id in
380          failwith (msg_erro nome msg)
381
382 let insere_declaracao_fun amb dec =
383   let open A in
384   match dec with

```

```

383 | Funcao {fn_nome; fn_tiporet; fn_formais; fn_corpo} ->
384   let formais = verifica_dup fn_formais in
385   let nome = fst fn_nome in
386   Amb.insere_fun amb nome formais fn_tiporet
387 | ACMD _ -> failwith "Instrucao invalida"
388
389 let fn_predefs =
390   let open A in
391     ("inputi", [("x", INTEIRO)], NONE);
392     ("inputs", [("x", STRING)], NONE);
393     ("inputf", [("x", REAL)], NONE)
394
395 let declara_predefinidas amb =
396   List.iter (fun (n,ps,tr) -> Amb.insere_fun amb n ps tr) fn_predefs
397
398 let semantico ast =
399   let amb_global = Amb.novo_amb [] in
400   let _ = declara_predefinidas amb_global in
401   let A.Programa instr = ast in
402   let decs_funs = List.filter (fun x ->
403     (match x with
404     | A.Funcao _ -> true
405     | _ -> false)) instr in
406   let _ = List.iter (insere_declaracao_fun amb_global) decs_funs in
407   let decs_funs = List.map (verifica_fun amb_global) decs_funs in
408   (A.Programa decs_funs, amb_global)

```

Listagem 4.11: sast.ml

```

1 open Ast
2
3 type expressao =
4   | EXPOPB of operador pos * expressao * expressao
5   | EXPOPU of operador pos * expressao
6   | EXPVAR of identificador pos
7   | EXPINT of int pos
8   | EXPSTRING of string pos
9   | EXPFLOAT of float pos
10  | EXPBOOL of bool pos
11  | EXPCALL of identificador pos * (expressao expressoes)

```

Listagem 4.12: tast.ml

```

1 open Ast
2
3 type expressao =
4   | EXPOPB of (operador * tipo) * (expressao * tipo) * (expressao *
5     tipo)
6   | EXPOPU of (operador * tipo) * (expressao * tipo)
7   | EXPCALL of identificador * (expressao expressoes) * tipo
8   | EXPINT of int * tipo
9   | EXPSTRING of string * tipo
10  | EXPFLOAT of float * tipo
11  | EXPBOOL of bool * tipo
12  | EXPVAR of identificador * tipo
13  | EXPNONE

```

4.3.1 Testes

Para compilar fazemos:

Listagem 4.13: Terminal

```

1 $ ocamlbuild -use-ocamlfind -use-menhir -menhir "menhir --table" -package
   menhirLib semanticoTeste.byte
2 Finished, 40 targets (37 cached) in 00:00:00.
3 $ rlwrap ocaml
4     OCaml version 4.07.0
5
6 Findlib has been successfully loaded. Additional directives:
7 #require "package";;          to load a package
8 #list;;                       to list the available packages
9 #camlp4o;;                    to load camlp4 (standard syntax)
10 #camlp4r;;                   to load camlp4 (revised syntax)
11 #predicates "p,q,...";;      to set these predicates
12 Topfind.reset();;            to force that packages will be reloaded
13 #thread;;                    to enable threads
14
15 /Users/tatianefx/.opam/default/lib/menhirLib: added to search path
16 /Users/tatianefx/.opam/default/lib/menhirLib/menhirLib.cmo: loaded
17 # verifica_tipos "tests/micro10.py";;
```

Listagem 4.14: testeSemantico.txt

```

1 EOF
2 - : Tast.expressao Ast.programa * Ambiente.t =
3 (Programa
4   [Funcao
5     {fn_nome =
6       ("main",
7        {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum = 4});
8     fn_tiporet = NONE; fn_formais = []};
9   fn_corpo =
10    [ATRIBUICAO (Tast.EXPVAR ("numero", INTEIRO), Tast.EXPINT (0,
11      INTEIRO));
12    ATRIBUICAO (Tast.EXPVAR ("fat", INTEIRO), Tast.EXPINT (0, INTEIRO))
13    ;
14    PRINT (Tast.EXPSTRING ("Digite um numero: ", STRING));
15    LEIAI (Tast.EXPVAR ("numero", INTEIRO));
16    ATRIBUICAO (Tast.EXPVAR ("fat", INTEIRO),
17      Tast.EXPCALL ("fatorial", [Tast.EXPVAR ("numero", INTEIRO)],
18      INTEIRO));
19    PRINT (Tast.EXPSTRING ("O fatorial eh ", STRING));
20    PRINT (Tast.EXPVAR ("fat", INTEIRO))];];
21 Funcao
22 {fn_nome =
23   ("fatorial",
24    {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum = 4});
25   fn_tiporet = INTEIRO;
26   fn_formais =
27     [(("n",
28      {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum =
29        13}),
30      INTEIRO)];
31   fn_corpo =
32     [CONDICAOIF
```

```

29      (Tast.EXPOPB ((MENORIGUALQ, BOOLEAN),
30        (Tast.EXPVAR ("n", INTEIRO), INTEIRO),
31        (Tast.EXPINT (0, INTEIRO), INTEIRO)),
32      [RETORNO (Some (Tast.EXPINT (1, INTEIRO)))],
33      Some
34        (CONDICAOelifElse
35          [RETORNO
36            (Some
37              (Tast.EXPOPB ((MULTIPLICACAO, INTEIRO),
38                (Tast.EXPVAR ("n", INTEIRO), INTEIRO),
39                (Tast.EXPCALL ("fatorial",
40                  [Tast.EXPOPB ((SUBTRACAO, INTEIRO),
41                    (Tast.EXPVAR ("n", INTEIRO), INTEIRO),
42                    (Tast.EXPINT (1, INTEIRO), INTEIRO))],
43                  INTEIRO),
44                  INTEIRO))))))]),
45  <abstr>)

```

4.4 Interprete

Listagem 4.15: interprete.ml

```

1  module Amb = AmbInterp
2  module A = Ast
3  module S = Sast
4  module T = Tast
5
6  exception Valor_de_retorno of T.expressao
7
8  let obtem_nome_tipo_var exp = let open T in
9    match exp with
10     | EXPVAR (nome,tipo) -> (nome,tipo)
11     | _ -> failwith "obtem_nome_tipo_var1: nao eh
        variavel"
12
13  let pega_int exp =
14    match exp with
15     | T.EXPINT (i,_) -> i
16     | _ -> failwith "pega_int: nao eh inteiro"
17
18  let pega_float exp = match exp with
19     | T.EXPFLOAT (f,_) -> f
20     | _ -> failwith "pega_float: nao eh inteiro"
21
22  let pega_str exp =
23    match exp with
24     | T.EXPSTRING (s,_) -> s
25     | _ -> failwith "pega_string: nao eh string"
26
27  let pega_bool exp =
28    match exp with
29     | T.EXPBOOL (b,_) -> b
30     | _ -> failwith "pega_bool: nao eh booleano"
31
32  type classe_op = Aritmetico | Relacional | Logico
33

```



```

34 let classifica op =
35   let open A in
36   match op with
37     OR
38   | NEGACAO
39   | AND -> Logico
40   | MENORQ
41   | MAIORQ
42   | MAIORIGUALQ
43   | MENORIGUALQ
44   | EHIGUAL
45   | EHDIFERENTE -> Relacional
46   | ADICAO
47   | SUBTRACAO
48   | MULTIPLICACAO
49   | MOD
50   | DIVISAO -> Aritmetico
51
52
53 let rec interpreta_exp amb exp =
54   let open A in
55   let open T in
56   match exp with
57   | EXPFLOAT _
58   | EXPINT _
59   | EXPSTRING _
60   | EXPBOOL _ -> exp
61   | EXPVAR (nome, tipo) ->
62     (match (Amb.busca amb nome) with
63      | Amb.EntVar (_, v) ->
64        (match v with
65         | Some valor -> valor
66         | None -> failwith "variável nao inicializada: ")
67      | _ -> failwith "interpreta_exp: expvar")
68   | EXPOPB ((op,top), (esq, tesq), (dir,tdir)) ->
69     let vesq = interpreta_exp amb esq
70     and vdir = interpreta_exp amb dir in
71
72     let interpreta_aritmetico () =
73       (
74         match tesq with
75       | INTEIRO ->
76         (
77           match op with
78           | ADICAO -> EXPINT (pega_int vesq + pega_int vdir, top)
79           | SUBTRACAO -> EXPINT (pega_int vesq - pega_int vdir, top)
80           | MULTIPLICACAO -> EXPINT (pega_int vesq * pega_int vdir, top)
81           | DIVISAO -> EXPINT (pega_int vesq / pega_int vdir, top)
82           | _ -> failwith "interpreta_aritmetico"
83         )
84       | _ -> failwith "interpreta_aritmetico"
85     )
86
87     and interpreta_relacional () =
88       (match tesq with
89       | INTEIRO ->
90         (match op with
91          | MAIORIGUALQ -> EXPBOOL (pega_int vesq >= pega_int vdir,

```

```

    top)
93 | MENORIGUALQ -> EXPBOOL (pega_int vesq <= pega_int vdir,
    top)
94 | MENORQ      -> EXPBOOL (pega_int vesq <  pega_int vdir,
    top)
95 | MAIORQ      -> EXPBOOL (pega_int vesq >  pega_int vdir,
    top)
96 | EHIGUAL     -> EXPBOOL (pega_int vesq == pega_int vdir,
    top)
97 | EHDIFERENTE -> EXPBOOL (pega_int vesq != pega_int
    vdir, top)
98 | _           -> failwith "interpreta_relacional"
99 )
100 | STRING ->
101   (match op with
102   | MAIORIGUALQ -> EXPBOOL (pega_str vesq >= pega_str vdir,
    top)
103   | MENORIGUALQ -> EXPBOOL (pega_str vesq <= pega_str vdir,
    top)
104   | MENORQ      -> EXPBOOL (pega_str vesq <  pega_str vdir,
    top)
105   | MAIORQ      -> EXPBOOL (pega_str vesq >  pega_str vdir,
    top)
106   | EHIGUAL     -> EXPBOOL (pega_str vesq == pega_str vdir,
    top)
107   | EHDIFERENTE -> EXPBOOL (pega_str vesq != pega_str
    vdir, top)
108   | _           -> failwith "interpreta_relacional"
109 )
110 | BOOLEAN ->
111   (match op with
112   | MAIORIGUALQ -> EXPBOOL (pega_bool vesq >= pega_bool vdir,
    top)
113   | MENORIGUALQ -> EXPBOOL (pega_bool vesq <= pega_bool vdir,
    top)
114   | MENORQ      -> EXPBOOL (pega_bool vesq <  pega_bool vdir,
    top)
115   | MAIORQ      -> EXPBOOL (pega_bool vesq >  pega_bool vdir,
    top)
116   | EHIGUAL     -> EXPBOOL (pega_bool vesq == pega_bool vdir,
    top)
117   | EHDIFERENTE -> EXPBOOL (pega_bool vesq != pega_bool
    vdir, top)
118   | _           -> failwith "interpreta_relacional"
119 )
120 | REAL ->
121   (match op with
122   | MAIORIGUALQ -> EXPBOOL (pega_float vesq == pega_float vdir
    , top)
123   | MENORIGUALQ -> EXPBOOL (pega_float vesq == pega_float vdir
    , top)
124   | MENORQ      -> EXPBOOL (pega_float vesq <  pega_float vdir
    , top)
125   | MAIORQ      -> EXPBOOL (pega_float vesq >  pega_float vdir
    , top)
126   | EHIGUAL     -> EXPBOOL (pega_float vesq == pega_float
    vdir, top)
127   | EHDIFERENTE -> EXPBOOL (pega_float vesq != pega_float
    vdir, top)

```

```

128         | _          -> failwith "interpreta_relacional"
129     )
130     | _ -> failwith "interpreta_relacional"
131 )
132
133 and interpreta_logico () =
134     (match tesq with
135     | BOOLEAN ->
136         (match op with
137         | OR -> EXPBOOL (pega_bool vesq || pega_bool vdir, top)
138         | AND -> EXPBOOL (pega_bool vesq && pega_bool vdir, top)
139         | _ -> failwith "interpreta_logico"
140         )
141     | _ -> failwith "interpreta_logico"
142     )
143
144 in
145 let valor = (match (classifica op) with
146     Aritmetico -> interpreta_aritmetico ()
147     | Relacional -> interpreta_relacional ()
148     | Logico -> interpreta_logico ()
149     )
150 in
151     valor
152
153 | EXPOPU ((op, top), (exp, texp)) ->
154     let vexp = interpreta_exp amb exp in
155     let interpreta_not () =
156         (match texp with
157         | A.BOOLEAN -> EXPBOOL (not (pega_bool vexp), top)
158         | _ -> failwith "Operador unario indefinido")
159     and interpreta_negativo () =
160         (match texp with
161         | A.INTEIRO -> EXPINT (-1 * pega_int vexp, top)
162         | A.REAL -> EXPFLOAT (-1.0 *. pega_float vexp, top)
163         | _ -> failwith "Operador unario indefinido")
164     in
165     let valor =
166         (match op with
167         | NEGACAO -> interpreta_not ()
168         | SUBTRACAO -> interpreta_negativo ()
169         | _ -> failwith "Operador unario indefinido")
170     in valor
171 | EXPCALL (id, args, tipo) ->
172     let open Amb in
173         (match (Amb.busca amb id) with
174         | Amb.EntFun {tipo_fn; formais; corpo} ->
175             let vargs = List.map (interpreta_exp amb) args in
176             let vformais = List.map2 (fun (n,t) v -> (n, t, Some v))
177                 formais vargs
178             in interpreta_fun amb vformais corpo
179         | _ -> failwith "interpreta_exp: expchamada"
180         )
181 | EXPNONE -> T.EXPNONE
182
183 and interpreta_cmd amb cmd =
184     let open A in
185     let open T in
186     match cmd with

```

```

186 RETORNO exp ->
187 (* Levantar uma exceção foi necessária pois, pela semântica do comando
    de *)
188 (* retorno, sempre que ele for encontrado em uma função, a computação
    *)
189 (* deve parar retornando o valor indicado, sem realizar os demais
    comandos. *)
190 (match exp with
191   (* Se a função não retornar nada, então retorne ExpVoid *)
192   | None -> raise (Valor_de_retorno EXPNONE)
193   | Some e ->
194     (* Avalia a expressão e retorne o resultado *)
195     let e1 = interpreta_exp amb e in
196     raise (Valor_de_retorno e1)
197 | CONDICAOF (teste, entao, senao) ->
198   let testel = interpreta_exp amb teste in
199   (match testel with
200     | EXPBOOL (true, _) ->
201       (* Interpreta cada comando do bloco 'então' *)
202       List.iter (interpreta_cmd amb) entao
203     | _ ->
204       (* Interpreta cada comando do bloco 'senão', se houver *)
205       (match senao with
206         | None -> ()
207         | Some bloco -> interpreta_cmd amb bloco))
208 | CONDICAOelifElse comandos ->
209   List.iter (interpreta_cmd amb) comandos
210 | ATRIBUICAO (elem, exp) ->
211   let resp = interpreta_exp amb exp in
212   (match elem with
213     | T.EXPVAR (id, tipo) ->
214       (try
215         begin
216           match (Amb.busca amb id) with
217             | Amb.EntVar (t, _) -> Amb.atualiza_var amb id tipo (
                Some resp)
218             | Amb.EntFun _ -> failwith "falha na atribuicao"
219           end
220         with Not_found ->
221           let _ = Amb.insere_local amb id tipo None in
222           Amb.atualiza_var amb id tipo (Some resp))
223     | _ -> failwith "Falha CmdAtrib"
224   )
225 | CHAMADADEFUNCAO exp -> ignore( interpreta_exp amb exp )
226 | LEIAI exp
227 | LEIAF exp
228 | LEIAS exp ->
229   (* Obtem os nomes e os tipos de cada um dos argumentos *)
230   let nt = obtem_nome_tipo_var exp in
231   let leia_var (nome, tipo) =
232     let _ =
233       (try
234         begin
235           match (Amb.busca amb nome) with
236             | Amb.EntVar (_, _) -> ()
237             | Amb.EntFun _ -> failwith "falha no input"
238           end
239         with Not_found ->
240           let _ = Amb.insere_local amb nome tipo None in

```

```

241     )
242     in
243     let valor =
244         (match tipo with
245         | INTEIRO -> T.EXPINT (read_int () , tipo)
246         | STRING -> T.EXPSTRING (read_line () , tipo)
247         | REAL -> T.EXPFLOAT (read_float () , tipo)
248         | _ -> failwith "Fail input")
249     in Amb.atualiza_var amb nome tipo (Some valor)
250 in leia_var nt
251 | PRINT exp ->
252     let resp = interpreta_exp amb exp in
253     (match resp with
254     | T.EXPINT (n,_) -> print_int n
255     | T.EXPFLOAT (n,_) -> print_float n
256     | T.EXPSTRING (n,_) -> print_string n
257     | T.EXPBOOL (b,_) ->
258         let _ = print_string (if b then "true" else "false")
259         in print_string " "
260     | _ -> failwith "Fail print"
261     )
262 | WHILELOOP (cond, cmds) ->
263     let rec laco cond cmds =
264         let condResp = interpreta_exp amb cond in
265         (match condResp with
266         | EXPBOOL (true,_) ->
267             (* Interpreta cada comando do bloco 'então' *)
268             let _ = List.iter (interpreta_cmd amb) cmds in
269             laco cond cmds
270         | _ -> ())
271     in laco cond cmds
272 | FORLOOP (idt, int_de ,int_ate, bloco) ->
273     let (elem1,tipo) = obtem_nome_tipo_var idt in
274     let rec executa_para amb int_de int_ate bloco elem1 tipo =
275         if (int_de) <= (int_ate)
276         then begin
277             (*Executa o bloco de código: *)
278             List.iter (interpreta_cmd amb) bloco;
279             (*Atualiza o valor da variavel: *)
280             Amb.atualiza_var amb elem1 tipo (Some ( EXPINT( (int_de
281                 + 1 ),INTEIRO) ) );
281             (*Chamada recursiva:*)
282             executa_para amb (int_de + 1) int_ate bloco elem1 tipo;
283         end in
284     executa_para amb (pega_int int_de) (pega_int int_ate) bloco elem1 tipo
285
286 and interpreta_fun amb fn_formais fn_corpo =
287     let open A in
288     (* Estende o ambiente global, adicionando um ambiente local *)
289     let ambfn = Amb.novo_escopo amb in
290     (* Associa os argumento
291     s aos parâmetros e insere no novo ambiente *)
292     let insere_parametro (n,t,v) = Amb.insere_param ambfn n t v in
293     let _ = List.iter insere_parametro fn_formais in
294     (* Interpreta cada comando presente no corpo da função usando o novo
295     *)
296     (* ambiente
297     *)
298     try

```

```

297         let _ = List.iter (interpreta_cmd ambfn) fn_corpo in T.EXPNONE
298     with
299         Valor_de_retorno expret -> expret
300
301 let insere_declaracao_fun amb dec =
302     let open A in
303         match dec with
304         | Funcao {fn_nome; fn_tiporet; fn_formais; fn_corpo} ->
305             let nome = fst fn_nome in
306             let formais = List.map (fun (n,t) -> ((fst n), t)) fn_formais in
307             Amb.insere_fun amb nome formais fn_tiporet fn_corpo
308         | _ -> failwith "Erro de declaracao de funcao"
309
310
311 let fn_predefs = let open A in [
312     ("inputi", [("x", INTEIRO )], NONE, []);
313     ("inputf", [("x", REAL )], NONE, []);
314     ("inputs", [("x", STRING )], NONE, []);
315
316 ]
317
318 (* insere as funções pré definidas no ambiente global *)
319 let declara_predefinidas amb =
320     List.iter (fun (n,ps,tr,c) -> Amb.insere_fun amb n ps tr c) fn_predefs
321
322 let interprete ast =
323     let open Amb in
324     let amb_global = Amb.novo_amb [] in
325     let _ = declara_predefinidas amb_global in
326     let A.Programa instr = ast in
327     let decs_funs = List.filter (fun x ->
328         (match x with
329         | A.Funcao _ -> true
330         | _ -> false)) instr in
331     let _ = List.iter (insere_declaracao_fun amb_global) decs_funs in
332     (try begin
333         (match (Amb.busca amb_global "main") with
334         | Amb.EntFun { tipo_fn ; formais ; corpo } ->
335             let vformais = List.map (fun (n,t) -> (n, t, None)) formais
336             in
337             let _ = interpreta_fun amb_global vformais corpo in
338             ()
339         | _ -> failwith "variavel declarada como 'main'")
340     end with Not_found -> failwith "Funcao main nao declarada ")

```

Listagem 4.16: ambInterp.ml

```

1 module Tab = Tabsimb
2 module A = Ast
3 module T = Tast
4
5 type entrada_fn = {
6     tipo_fn: A.tipo;
7     formais: (A.identificador * A.tipo) list;
8     corpo: T.expressao A.comandos
9 }
10
11 type entrada = EntFun of entrada_fn
12             | EntVar of A.tipo * (T.expressao option)

```

```

13
14
15 type t = {
16   ambv : entrada Tab.tabela
17 }
18
19 let novo_amb xs = { ambv = Tab.cria xs }
20
21 let novo_escopo amb = { ambv = Tab.novo_escopo amb.ambv }
22
23 let busca amb ch = Tab.busca amb.ambv ch
24
25 let atualiza_var amb ch t v =
26   Tab.atualiza amb.ambv ch (EntVar (t,v))
27
28 let insere_local amb nome t v =
29   Tab.insere amb.ambv nome (EntVar (t,v))
30
31 let insere_param amb nome t v =
32   Tab.insere amb.ambv nome (EntVar (t,v))
33
34 let insere_fun amb nome params resultado corpo =
35   let ef = EntFun { tipo_fn = resultado;
36                     formais = params;
37                     corpo = corpo }
38   in Tab.insere amb.ambv nome ef

```

4.4.1 Teste

Para compilar fazemos:

Listagem 4.17: Terminal

```

1 $ ocamlbuild -use-ocamlfind -use-menhir -menhir "menhir --table" -package
   menhirLib interpreteTeste.byte
2 Finished, 40 targets (37 cached) in 00:00:00.
3 $ rlwrap ocaml
4     OCaml version 4.07.0
5
6 Findlib has been successfully loaded. Additional directives:
7 #require "package";;      to load a package
8 #list;;                   to list the available packages
9 #camlp4o;;                 to load camlp4 (standard syntax)
10 #camlp4r;;                 to load camlp4 (revised syntax)
11 #predicates "p,q,...";;   to set these predicates
12 Topfind.reset();;         to force that packages will be reloaded
13 #thread;;                 to enable threads
14
15 /Users/tatianefx/.opam/default/lib/menhirLib: added to search path
16 /Users/tatianefx/.opam/default/lib/menhirLib/menhirLib.cmo: loaded
17 # interprete "tests/micro10.py";;

```

Listagem 4.18: testeInterprete.txt

```

1 EOF
2 Digite um numero: 5
3 O fatorial eh 120- : unit = ()

```

4.5 Código de 3 Endereços

Listagem 4.19: codigo.ml

```

1 open Ast
2 open Tast
3
4 type endereco =
5     Nome of string
6     | ConstInt of int
7     | ConstFloat of float
8     | ConstBool of bool
9     | Temp of int
10 and instrucao =
11     AtribBin of endereco * endereco * opBin * endereco (* x = y op z *)
12     | AtribUn of endereco * opUn * endereco (* x = op y *)
13     | Copia of endereco * endereco (* x = y *)
14     | Goto of instrucao (* goto L *)
15     | If of endereco * instrucao (* if x goto L *)
16     | IfFalse of endereco * instrucao (* ifFalse x goto
17         L *)
18     | IfRelgoto of endereco * opRel * endereco * instrucao (* if x oprel y goto
19         L *)
20     | Call of string * (endereco * Ast.tipo) list * Ast.tipo (* call p, [(x,
21         t)], t *)
22     | Recebe of string * Ast.tipo
23     | Local of string * Ast.tipo
24     | Global of string * Ast.tipo
25     | CallFn of endereco * string * (endereco * Ast.tipo) list * Ast.tipo
26         (* x = call p,n,t *)
27     | Return of endereco option
28     | Imprime of endereco
29     | BeginFun of string * int * int (* beginFun p,nparam, nlocais *)
30     | EndFun
31     | Rotulo of string
32
33 and opBin = Ast.operador * Ast.tipo
34
35 and opUn = Ast.operador * Ast.tipo
36
37 and opRel = Ast.operador * Ast.tipo

```

Listagem 4.20: codend.ml

```

1 open Printf
2
3 open Ast
4 open Tast
5 open Codigo
6
7 module Amb = AmbInterp
8
9 exception Valor_de_retorno of expressao
10
11 let conta_temp = ref 0
12 let conta_rotulos = ref (Hashtbl.create 5)
13
14 let zera_contadores () =

```



```

15  begin
16      conta_temp := 0;
17      conta_rotulos := Hashtbl.create 5
18  end
19
20  let novo_temp () =
21      let numero = !conta_temp in
22      let _ = incr conta_temp in
23      Temp numero
24
25  let novo_rotulo prefixo =
26      if Hashtbl.mem !conta_rotulos prefixo
27      then
28          let numero = Hashtbl.find !conta_rotulos prefixo in
29          let _ = Hashtbl.replace !conta_rotulos prefixo (numero + 1) in
30          Rotulo (prefixo ^ (string_of_int numero))
31      else
32          let _ = Hashtbl.add !conta_rotulos prefixo 1 in
33          Rotulo (prefixo ^ "0")
34
35  (*Codigo para impressao *)
36
37  let endr_to_str = function
38      | Nome s -> s
39      | ConstInt n -> string_of_int n
40      | ConstFloat n -> string_of_float n
41      | Temp n -> "t" ^ string_of_int n
42
43  let tipo_to_str t =
44      match t with
45      | INTEIRO -> "inteiro"
46      | STRING -> "string"
47      | BOOLEAN -> "bool"
48      | NONE -> "void"
49      | REAL -> "float"
50
51  let op_to_str op =
52      match op with
53      | ADICAO -> "+"
54      | SUBTRACAO -> "-"
55      | MULTIPLICACAO -> "*"
56      | DIVISAO -> "/"
57      | MOD -> "%"
58      | MENORQ -> "<"
59      | MENORIGUALQ -> "<="
60      | EHIGUAL -> "="
61      | EHDIFERENTE -> "!="
62      | MAIORQ -> ">"
63      | MAIORIGUALQ -> ">="
64      | AND -> "&&"
65      | OR -> "||"
66      | NEGACAO -> "not"
67
68  let rec args_to_str ats =
69      match ats with
70      | [] -> ""
71      | [(a,t)] ->
72          let str = sprintf "(%s,%s)" (endr_to_str a) (tipo_to_str t) in
73          str

```

```

74 | (a,t) :: ats ->
75 | let str = sprintf "(%s,%s)" (endr_to_str a) (tipo_to_str t) in
76 | str ^ ", " ^ args_to_str ats
77
78 let rec escreve_cod3 c =
79 | match c with
80 | AtribBin (x,y,op,z) ->
81 |   sprintf "%s := %s %s %s\n" (endr_to_str x)
82 |   (endr_to_str y) (op_to_str (fst op)) (
83 |   endr_to_str z)
84 | AtribUn (x,op,y) ->
85 |   sprintf "%s := %s %s\n" (endr_to_str x) (op_to_str (fst op)) (
86 |   endr_to_str y)
87 | Copia (x,y) ->
88 |   sprintf "%s := %s\n" (endr_to_str x) (endr_to_str y)
89 | Goto l ->
90 |   sprintf "goto %s\n" (escreve_cod3 l)
91 | If (x,l) ->
92 |   sprintf "if %s goto %s\n" (endr_to_str x) (escreve_cod3 l)
93 | IfFalse (x,l) ->
94 |   sprintf "ifFalse %s goto %s\n" (endr_to_str x) (escreve_cod3 l)
95 | IfRelgoto (x,oprel,y,l) ->
96 |   sprintf "if %s %s %s goto %s\n" (endr_to_str x) (op_to_str (fst
97 |   oprel))
98 |   (endr_to_str y) (escreve_cod3 l)
99 | Call (p,ats,t) -> sprintf "call %s(%s): %s\n" p (args_to_str ats) (
100 |   tipo_to_str t)
101 | Recebe (x,t) -> sprintf "recebe %s,%s\n" x (tipo_to_str t)
102 | Local (x,t) -> sprintf "local %s,%s\n" x (tipo_to_str t)
103 | Global (x,t) -> sprintf "global %s,%s\n" x (tipo_to_str t)
104 | CallFn (x,p,ats,t) ->
105 |   sprintf "%s := call %s(%s): %s\n" (endr_to_str x) p (args_to_str ats
106 |   ) (tipo_to_str t)
107 | Return x ->
108 |   (match x with
109 |   | None -> "return\n"
110 |   | Some x -> sprintf "return %s\n" (endr_to_str x) )
111 | Imprime x -> sprintf "printf %s\n" (endr_to_str x)
112 | BeginFun (id,np,nl) -> sprintf "beginFun %s(%d,%d)\n" id np nl
113 | EndFun -> "endFun\n\n"
114 | Rotulo r -> sprintf "%s: " r
115
116
117
118 let rec escreve_codigo cod =
119 | match cod with
120 | [] -> printf "\n"
121 | c::cs -> printf "%s" (escreve_cod3 c);
122 |   escreve_codigo cs
123
124
125
126 (* Código do tradutor para código de 3 endereço *)
127
128 let pega_tipo exp =
129 | match exp with
130 | EXPINT (n, t) -> t
131 | EXPVAR (v, t) -> t
132 | EXPOPB ((op,t),_,_) -> t
133 | EXPCALL (id, args, t) -> t
134 | _ -> failwith "pega_tipo: não implementado"

```

```

128
129 let rec traduz_exp amb exp =
130   match exp with
131   | EXPFLOAT (n, REAL) ->
132     let t = novo_temp () in
133     (t, [Copia (t, ConstFloat n)])
134   | EXPSTRING (n, STRING) ->
135     let t = novo_temp () in
136     (t, [Copia (t, Nome n)])
137   | EXPBOOL (n, BOOLEAN) ->
138     let t = novo_temp () in
139     (t, [Copia (t, ConstBool n)])
140   | EXPINT (n, INTEIRO) ->
141     let t = novo_temp () in
142     (t, [Copia (t, ConstInt n)])
143   | EXPVAR (v, tipo) ->
144     (match v with
145      VarSimples nome ->
146        let id = fst nome in
147        (match (Amb.busca amb id) with
148         | Amb.EntVar (_, va) ->
149           (match va with
150            | Some valor -> valor; ((Nome id), [])
151            | None       -> failwith "variável não inicializada: ")
152         | _ -> failwith "interpreta_exp: expvar"
153        )
154      | _ -> failwith "traduz_exp: não implementado"
155     )
156   | EXPOPB (op, exp1, exp2) ->
157     let (endr1, codigo1) = let (e1,t1) = exp1 in (traduz_exp amb) e1
158     and (endr2, codigo2) = let (e2,t2) = exp2 in (traduz_exp amb) e2
159     and t = novo_temp () in
160     let codigo = codigo1 @ codigo2 @ [AtribBin (t, endr1, op, endr2)] in
161     (t, codigo)
162
163
164   | EXPCALL (id, args, tipo_fn) ->
165     let (enderecos, codigos) = List.split (List.map (traduz_exp amb)
166      args) in
167     let tipos = List.map pega_tipo args in
168     let endr_tipos = List.combine enderecos tipos
169     and t = novo_temp () in
170     let codigo = (List.concat codigos) @
171      [CallFn (t, id, endr_tipos, tipo_fn)]
172     in
173     (t, codigo)
174   | _ -> failwith "traduz_exp: não implementado"
175
176 let rec traduz_cmd amb cmd =
177   match cmd with
178   | RETORNO exp ->
179     (match exp with
180      | None -> [Return None]
181      | Some e ->
182        let (endr_exp, codigo_exp) = (traduz_exp amb) e in
183        codigo_exp @ [Return (Some endr_exp)]
184      )
185   | PRINT exp ->
186     let (endr_exp, codigo_exp) = (traduz_exp amb) exp in

```

```

186     codigo_exp @ [Imprime endr_exp]
187 | ATRIBUICAO (elem, EXPINT (n, INTEIRO)) ->
188     let (endr_elem, codigo_elem) = (traduz_exp amb) elem
189     in codigo_elem @ [Copia (endr_elem, ConstInt n)]
190 | ATRIBUICAO (elem, EXPFLOAT (n, REAL)) ->
191     let (endr_elem, codigo_elem) = (traduz_exp amb) elem
192     in codigo_elem @ [Copia (endr_elem, ConstFloat n)]
193 | ATRIBUICAO (elem, EXPSTRING (n, STRING)) ->
194     let (endr_elem, codigo_elem) = (traduz_exp amb) elem
195     in codigo_elem @ [Copia (endr_elem, Nome n)]
196 | ATRIBUICAO (elem, EXPBOOL (n, BOOLEAN)) ->
197     let (endr_elem, codigo_elem) = (traduz_exp amb) elem
198     in codigo_elem @ [Copia (endr_elem, ConstBool n)]
199 | ATRIBUICAO (elem, exp) ->
200     (* let (var1, tdir) = traduz_exp amb exp in
201     ( match elem with
202     EXPVAR (v, t) ->
203     (match v with
204     VarSimples nome ->
205     let id = fst nome
206     and pos = snd nome in
207     (try
208     begin
209     (match (Amb.busca amb id) with
210     Amb.EntVar (tipo, texp) ->
211     ATRIBUICAO (EXPVAR (VarSimples nome, tipo), var1)
212     | Amb.EntFun _ ->
213     failwith "traduz_cmd: não implementado" )
214     end
215     with Not_found ->
216     let _ = Amb.insere_local amb id tdir in
217     ATRIBUICAO (EXPVAR (VarSimples nome, tdir), var1))
218     | _ -> failwith "Falha ATRIBUICAO"
219     )
220     *)
221     let (endr_exp, codigo_exp) = (traduz_exp amb) exp
222     and (endr_elem, codigo_elem) = (traduz_exp amb) elem in
223     let codigo = codigo_exp @ codigo_elem @ [Copia (endr_elem, endr_exp)]
224     in codigo
225 | CONDICAIF (teste, entao, senao) ->
226     let (endr_teste, codigo_teste) = (traduz_exp amb) teste
227     and codigo_entao = traduz_cmds amb entao
228     and rotulo_falso = novo_rotulo "L" in
229     (match senao with
230     | None -> codigo_teste @
231     [IfFalse (endr_teste, rotulo_falso)] @
232     codigo_entao @
233     [rotulo_falso]
234     | Some cmds ->
235     let codigo_senao = traduz_cmds amb [cmds]
236     and rotulo_fim = novo_rotulo "L" in
237     codigo_teste @
238     [IfFalse (endr_teste, rotulo_falso)] @
239     codigo_entao @
240     [Goto rotulo_fim] @
241     [rotulo_falso] @ codigo_senao @
242     [rotulo_fim]
243     )
244 | CONDICAOfElseElse comandos ->

```

```

245     traduz_cmds amb comandos
246 | CHAMADADEFUNCAO (EXPCALL (id, args, tipo_fn)) ->
247     let (enderecos, codigos) = List.split (List.map (traduz_exp amb)
248         args) in
249     let tipos = List.map pega_tipo args in
250     let endr_tipos = List.combine enderecos tipos in
251     (List.concat codigos) @
252     [Call (id, endr_tipos, tipo_fn)]
253 | WHILELOOP (teste, cmd) ->
254     let (endr_teste, codigo_teste) = traduz_exp amb teste
255     and codigo_verdadeiro = traduz_cmds amb cmd
256     and rotulo_comeco = novo_rotulo "L"
257     and rotulo_falso = novo_rotulo "L" in
258     [rotulo_comeco] @ codigo_teste @ [IfFalse (endr_teste, rotulo_falso)
259         ] @ codigo_verdadeiro @ [Goto rotulo_comeco] @
260     [rotulo_falso]
261 | CmdSaida args ->
262     let (enderecos, codigos) = List.split (List.map (traduz_exp amb)
263         args) in
264     let tipos = List.map pega_tipo args in
265     let endr_tipos = List.combine enderecos tipos in
266     (List.concat codigos) @
267     [Call ("print", endr_tipos, NONE)]
268
269
270 and traduz_cmds amb cmds =
271     match cmds with
272     | [] -> []
273     | cmd :: cmds ->
274         let codigo = traduz_cmd amb cmd in
275         codigo @ traduz_cmds amb cmds
276
277 let traduz_fun amb ast =
278     let trad_local x =
279         match x with
280         | DecVar ((id,pos),t) -> Local (id,t)
281     in
282     (* Estende o ambiente global, adicionando um ambiente local *)
283     let ambfn = Amb.novo_escopo amb in
284     (* Associa os argumento
285     s aos parâmetros e insere no novo ambiente *)
286     let insere_parametro (n,t,v) = Amb.insere_param ambfn n t v in
287     match ast with
288     | Funcao {fn_nome; fn_tiporet; fn_formais; fn_locais; fn_corpo} ->
289         let nome = fst fn_nome
290         and formais = List.map (fun ((id,pos),tipo) -> Recebe (id,tipo))
291             fn_formais
292         and nformais = List.length fn_formais
293         and locais = List.map trad_local fn_locais
294         and nlocais = List.length fn_locais
295         and corpo = traduz_cmds ambfn fn_corpo in
296         [BeginFun (nome,nformais,nlocais)] @ formais @ locais @ corpo @ [
297             EndFun]
298
299 let insere_declaracao_fun amb dec =
300     match dec with

```

```

299     | Funcao {fn_nome; fn_tiporet; fn_formais; fn_locais; fn_corpo} ->
300         let nome = fst fn_nome in
301         let formais = List.map (fun (n,t) -> ((fst n), t)) fn_formais in
302         Amb.insere_fun amb nome formais fn_tiporet fn_corpo
303     | _ -> failwith "Erro de declaacao de funcao"
304
305 let fn_predefs = [
306     ("inputi", [("x", INTEIRO )], NONE, []);
307     ("inputf", [("x", REAL    )], NONE, []);
308     ("inputs", [("x", STRING  )], NONE, []);
309 ]
310
311 (* insere as funções pré definidas no ambiente global *)
312 let declara_predefinidas amb =
313     List.iter (fun (n,ps,tr,c) -> Amb.insere_fun amb n ps tr c) fn_predefs
314
315 let tradutor ast_tipada =
316     let open Amb in
317     let amb_global = Amb.novo_amb [] in
318     let _ = declara_predefinidas amb_global in
319     let Programa instr = ast_tipada in
320     let decs_funs = List.filter (fun x ->
321         (match x with
322         | Funcao _ -> true
323         | _ -> false)) instr in
324     let _ = List.iter (insere_declaracao_fun amb_global) decs_funs in
325     (try begin
326         (match (Amb.busca amb_global "main") with
327         | Amb.EntFun { tipo_fn ; formais ; corpo } ->
328             let vformais = List.map (fun (n,t) -> (n, t, None)) formais
329             in
330             let corpos      = List.map (traduz_fun amb_global) decs_funs
331             in
332             List.flatten corpos
333         | _ -> failwith "variavel declarada como 'main'")
334     end with Not_found -> failwith "Funcao main nao declarada ")

```

Listagem 4.21: codendtest.ml

```

1 open Printf
2 open Lexing
3
4 open Ast
5 exception Erro_Sintatico of string
6
7 module S = MenhirLib.General (* Streams *)
8 module I = Sintatico.MenhirInterpreter
9
10 open Semantico
11 open Codigo
12 open Cod3End
13
14 let message =
15     fun s ->
16         match s with
17         | 0 ->
18             "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
19         | 1 ->
20             "<YOUR SYNTAX ERROR MESSAGE HERE>\n"

```

```

21 | 34 ->
22 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
23 | 35 ->
24 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
25 | 36 ->
26 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
27 | 72 ->
28 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
29 | 47 ->
30 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
31 | 48 ->
32 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
33 | 49 ->
34 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
35 | 51 ->
36 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
37 | 52 ->
38 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
39 | 55 ->
40 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
41 | 56 ->
42 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
43 | 57 ->
44 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
45 | 58 ->
46 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
47 | 61 ->
48 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
49 | 62 ->
50 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
51 | 63 ->
52 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
53 | 64 ->
54 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
55 | 73 ->
56 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
57 | 74 ->
58 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
59 | 95 ->
60 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
61 | 89 ->
62 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
63 | 97 ->
64 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
65 | 98 ->
66 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
67 | 99 ->
68 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
69 | 65 ->
70 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
71 | 66 ->
72 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
73 | 53 ->
74 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
75 | 67 ->
76 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
77 | 68 ->
78 |     "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
79 | 59 ->

```

```

80         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
81 | 60 ->
82         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
83 | 42 ->
84         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
85 | 41 ->
86         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
87 | 70 ->
88         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
89 | 75 ->
90         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
91 | 77 ->
92         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
93 | 76 ->
94         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
95 | 105 ->
96         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
97 | 84 ->
98         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
99 | 43 ->
100        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
101 | 85 ->
102        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
103 | 86 ->
104        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
105 | 45 ->
106        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
107 | 46 ->
108        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
109 | 102 ->
110        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
111 | 103 ->
112        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
113 | 81 ->
114        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
115 | 3 ->
116        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
117 | 2 ->
118        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
119 | 6 ->
120        "estado 6: esperava um tipo. Exemplo:\n    x : inteiro;\n"
121 | 7 ->
122        "estado 7: esperava a definicao de um campo. Exemplo:\n    i:
            registro\n                parte_real: inteiro;\n                parte_imag:
            inteiro;\n                fim registro;\n                "
123 | 8 ->
124        "estado 8: esperava ':'. Exemplo:\n    x: inteiro;\n    "
125 | 9 ->
126        "estado 9: esperava um tipo. Exemplo:\n    x: inteiro;\n"
127 | 25 ->
128        "estado 25: esperva um ';'.'. \n"
129 | 26 ->
130        "estado 26: uma declaracao foi encontrada. Para continuar era\n
            esperado uma outra declara\195\167\195\163o ou a palavra '
            inicio'. \n"
131 | 29 ->
132        "estado 29: espera a palavra 'registro'. Exemplo:\n    i: registro\
            n                parte_real: inteiro;\n                parte_imag: inteiro;\n
            fim registro;\n"

```



```

133 | 31 ->
134 |     "estado 31: esperava um ';'. \n"
135 | 107 ->
136 |     "estado 107: uma declaracao foi encontrada. Para continuar era\n
        esperado uma outra declara\195\167\195\163o ou a palavra '
        inicio'. \n"
137 | 13 ->
138 |     "estado 13: esperava um '['. Exemplo:\n    arranjo [1..10] de
        inteiro;\n"
139 | 14 ->
140 |     "estado 14: esperava os limites do vetor. Exemplo:\n    arranjo
        [1..10] de inteiro;\n"
141 | 15 ->
142 |     "estado 15: esperava '..'. Exemplo:\n    1 .. 10\n"
143 | 16 ->
144 |     "estado 16: esperava um numero inteiro. Exemplo:\n    1 .. 10\n"
145 | 18 ->
146 |     "estado 18: esperava um ']'. Exemplo\n    arranjo [1..10] de
        inteiro;\n"
147 | 19 ->
148 |     "estado 19: esperava a palavra reservada 'de'. Exemplo:\n
        arranjo [1..10] de inteiro;\n"
149 | 20 ->
150 |     "estado 20: esperava um tipo. Exemplo\n    arranjo [1..10] de
        inteiro;\n"
151 | _ ->
152 |     raise Not_found
153
154 open Semantico
155 let posicao lexbuf =
156     let pos = lexbuf.lex_curr_p in
157     let lin = pos.pos_lnum
158     and col = pos.pos_cnum - pos.pos_bol - 1 in
159     sprintf "linha %d, coluna %d" lin col
160
161 (* [pilha checkpoint] extrai a pilha do autômato LR(1) contida em
    checkpoint *)
162
163 let pilha_checkpoint =
164     match checkpoint with
165     | I.HandlingError amb -> I.stack amb
166     | _ -> assert false (* Isso não pode acontecer *)
167
168 let estado_checkpoint : int =
169     match Lazy.force (pilha_checkpoint) with
170     | S.Nil -> (* O parser está no estado inicial *)
171         0
172     | S.Cons (I.Element (s, _, _, _), _) ->
173         I.number s
174
175 let sucesso v = Some v
176
177 let falha lexbuf (checkpoint : (Sast.expressao Ast.programa) I.checkpoint)
    =
178     let estado_atual = estado_checkpoint in
179     let msg = message estado_atual in
180     raise (Erro_Sintatico (Printf.sprintf "%d - %s.\n"
181                                     (Lexing.lexeme_start lexbuf) msg))
182

```

```

183 let loop lexbuf resultado =
184   let fornecedor = I.lexer_lexbuf_to_supplier Pre_processador.lexico
      lexbuf in
185   I.loop_handle sucesso (falha lexbuf) fornecedor resultado
186
187
188 let parse_com_erro lexbuf =
189   try
190     Some (loop lexbuf (Sintatico.Incremental.programa lexbuf.lex_curr_p))
191   with
192   | Lexico.Erro msg ->
193     printf "Erro lexico na %s:\n\t%s\n" (posicao lexbuf) msg;
194     None
195   | Erro_Sintatico msg ->
196     printf "Erro sintático na %s %s\n" (posicao lexbuf) msg;
197     None
198
199 let parse s =
200   let lexbuf = Lexing.from_string s in
201   let ast = parse_com_erro lexbuf in
202   ast
203
204 let parse_arq nome =
205   let ic = open_in nome in
206   let lexbuf = Lexing.from_channel ic in
207   let ast = parse_com_erro lexbuf in
208   let _ = close_in ic in
209   ast
210
211 let verifica_tipos nome =
212   let ast = parse_arq nome in
213   match ast with
214   | Some (Some ast) -> semantico ast
215   | _ -> failwith "Nada a fazer!\n"
216
217 let traduz nome =
218   let arv, amb = verifica_tipos nome in
219   tradutor arv
220
221 let imprime_traducao cod =
222   let _ = printf "\n" in
223   escreve_codigo cod
224
225
226 (* Para compilar:
227     ocamlbuild -use-ocamlfind -use-menhir -menhir "menhir --table" -
      package menhirLib cod3endTest.byte
228
229     Para usar, entre no ocaml
230
231     rlwrap ocaml
232
233     e se desejar ver apenas a árvore sintática que sai do analisador sintá
      tico, digite
234
235     parse_arq "exemplos/Tipos/ex8.tip";;
236
237     Depois, para ver a saída do analisador semântico já com a árvore
      anotada com

```

```

238   o tipos, digite:
239
240   verifica_tipos "exemplos/Tipos/ex8.tip";;
241
242   Note que o analisador semântico está retornando também o ambiente
      global. Se
243   quiser separá-los, digite:
244
245   let (arv, amb) = verifica_tipos "exemplos/Tipos/ex8.tip";;
246
247   Para ver o código de 3 endereços:
248   traduz "exemplos/Tipos/ex8.tip";;
249
250   ou se quiser ver em um formato mais legível:
251
252   let cod = traduz "exemplos/Tipos/ex8.tip" in imprime_traducao cod;;
253
254
255
256
257  *)

```

4.5.1 Teste

Listagem 4.22: teste.py

```

1 def main() -> int:
2     print("Teste")
3     while 1 == 1:
4         if 1 > 2:
5             return 1
6         elif 1 == 3:
7             return 4
8         else:
9             return 2
10    return 0

```

Para compilar fazemos:

Listagem 4.23: Terminal

```

1 $ ocamlbuild -use-ocamlfind -use-menhir -menhir "menhir --table" -package
   menhirLib cod3endTest.byte -r
2 Finished, 40 targets (40 cached) in 00:00:00.
3 $ rlwrap ocaml
4     OCaml version 4.07.1
5
6 Findlib has been successfully loaded. Additional directives:
7 #require "package";;      to load a package
8 #list;;                  to list the available packages
9 #camlp4o;;               to load camlp4 (standard syntax)
10 #camlp4r;;               to load camlp4 (revised syntax)
11 #predicates "p,q,...";;  to set these predicates
12 Topfind.reset();;        to force that packages will be reloaded
13 #thread;;                to enable threads
14
15 /Users/tatianefx/.opam/default/lib/menhirLib: added to search path
16 /Users/tatianefx/.opam/default/lib/menhirLib/menhirLib.cmo: loaded

```

```

17 # traduz "tests/teste.py";;
18
19 - : Codigo.instrucao list =
20 [BeginFun ("main", 0, 0); Copia (Temp 0, Nome "Teste"); Imprime (Temp 0);
21 Rotulo "L4"; Copia (Temp 1, ConstInt 1); Copia (Temp 2, ConstInt 1);
22 AtribBin (Temp 3, Temp 1, (EHIGUAL, BOOLEAN), Temp 2);
23 IfFalse (Temp 3, Rotulo "L5"); Copia (Temp 4, ConstInt 1);
24 Copia (Temp 5, ConstInt 2);
25 AtribBin (Temp 6, Temp 4, (MAIORQ, BOOLEAN), Temp 5);
26 IfFalse (Temp 6, Rotulo "L0"); Copia (Temp 7, ConstInt 1);
27 Return (Some (Temp 7)); Goto (Rotulo "L3"); Rotulo "L0";
28 Copia (Temp 8, ConstInt 1); Copia (Temp 9, ConstInt 3);
29 AtribBin (Temp 10, Temp 8, (EHIGUAL, BOOLEAN), Temp 9);
30 IfFalse (Temp 10, Rotulo "L1"); Copia (Temp 11, ConstInt 4);
31 Return (Some (Temp 11)); Goto (Rotulo "L2"); Rotulo "L1";
32 Copia (Temp 12, ConstInt 2); Return (Some (Temp 12)); Rotulo "L2";
33 Rotulo "L3"; Copia (Temp 13, ConstInt 0); Return (Some (Temp 13));
34 Goto (Rotulo "L4"); Rotulo "L5"; EndFun]
35 # let cod = traduz "tests/teste.py" in imprime_traducao cod;;
36
37
38 beginFun main(0,0)
39 t14 := Teste
40 printf t14
41 L10: t15 := 1
42 t16 := 1
43 t17 := t15 = t16
44 ifFalse t17 goto L11:
45 t18 := 1
46 t19 := 2
47 t20 := t18 > t19
48 ifFalse t20 goto L6:
49 t21 := 1
50 return t21
51 goto L9:
52 L6: t22 := 1
53 t23 := 3
54 t24 := t22 = t23
55 ifFalse t24 goto L7:
56 t25 := 4
57 return t25
58 goto L8:
59 L7: t26 := 2
60 return t26
61 L8: L9: t27 := 0
62 return t27
63 goto L10:
64 L11: endFun
65
66
67 - : unit = ()
68 #

```

4.6 Gerador

Listagem 4.24: gerador.ml

```
1 open Printf
```

4.6

```

2 open Ast
3 open Tast
4 open Codigo
5
6 let num_bytes t =
7     match t with
8         INTEIRO -> 4
9         | STRING -> 4
10        | BOOLEAN -> 4
11        | REAL -> 4
12        | NONE -> 4
13
14
15 let emite_cabecalho oc arq =
16     fprintf oc " .file \"%s\"\\n" arq
17
18 let rec emite_global oc cod =
19     match cod with
20     | Global (n,t) :: cod ->
21         let _ = fprintf oc " .comm %s,%d,4\\n" n (num_bytes t) in
22         emite_global oc cod
23     | _ ->
24         let _ = fprintf oc " .text\\n" in
25         cod
26
27 let emite_prologo oc nome =
28     fprintf oc "
29     .globl %s
30     .type %s, @function
31 %s:
32     pushq %%rbp
33     movq %%rsp, %%rbp
34 "
35     nome nome nome
36
37 let emite_epilogo oc nome =
38     fprintf oc " .size %s, .-\\n" nome nome
39
40 let rec move_parametros oc tbl cod regs =
41     match cod with
42     | Recebe (n,tipo) :: cod ->
43         let deslocamento = Hashtbl.find tbl (Nome n) in
44         let _ = fprintf oc "     movl    %%s, %d(%%rbp)          # %s\\n"
45             (List.hd regs) deslocamento n
46         in move_parametros oc tbl cod (List.tl regs)
47     | _ -> cod
48
49
50 let rec emite_corpo oc tbl cod =
51     match cod with
52     | Local _ :: cod -> emite_corpo oc tbl cod
53     | Copia (Nome n, ConstInt i) :: cod ->
54         begin
55             try
56                 let deslocamento = Hashtbl.find tbl (Nome n) in
57                 let _ = fprintf oc "     movl    $%d, %d(%%rbp)          # %s <- %d\\n"
58                     i deslocamento n i
59                 in emite_corpo oc tbl cod
60             with

```

```

61     Not_found -> (* Assume que é uma variável global *)
62     let _ = fprintf oc " movl  %d, %s(%rip)          # %s <- %d\n"
63                   i n n i
64     in emite_corpo oc tbl cod
65 end
66 | If (endr, inst) :: cod ->
67     let _ =
68         (match endr with
69         | Temp n -> let _ = fprintf oc "  if %d goto" n
70                   in emite_corpo oc tbl [inst]
71         | _ -> failwith "emite_corpo: endereco nao implementado"
72         )
73     in emite_corpo oc tbl cod
74 | IfFalse (endr, inst) :: cod ->
75     let _ =
76         (match endr with
77         | Temp n -> let _ = fprintf oc "  ifFalse %d goto" n
78                   in emite_corpo oc tbl [inst]
79         | _ -> failwith "emite_corpo: endereco nao implementado"
80         )
81     in emite_corpo oc tbl cod
82 | Goto inst :: cod ->
83     let _ = fprintf oc "goto "
84     and _ = emite_corpo oc tbl [inst]
85     in emite_corpo oc tbl cod
86 | Rotulo str :: cod ->
87     let _ = fprintf oc "%s:\n" str
88     in emite_corpo oc tbl cod
89 | Return opcao :: cod ->
90     let _ =
91         (match opcao with
92         | None -> ()
93         | Some endr ->
94             (match endr with
95             | Nome n ->
96                 begin
97                     try
98                         let deslocamento = Hashtbl.find tbl endr in
99                         fprintf oc "  movl  %d(%rip), %%eax          # EAX <- %s\n"
100                               deslocamento n
101                     with
102                         Not_found -> (* Assume que é uma variável global *)
103                         fprintf oc "  movl  %s(%rip), %%eax          # EAX <- %s\n"
104                               n n
105                     end
106                 | ConstInt n ->
107                     begin
108                         try
109                             let deslocamento = Hashtbl.find tbl endr in
110                             fprintf oc "  movl  %d(%rip), %%eax          # EAX <- %d\n"
111                                   deslocamento n
112                         with
113                             Not_found -> (* Assume que é uma variável global *)
114                             fprintf oc "  movl  %d(%rip), %%eax          # EAX <- %d\n"
115                                   n n
116                         end
117                     | Temp n ->

```

```

118         begin
119             try
120                 let deslocamento = Hashtbl.find tbl endr in
121                 fprintf oc " movl %d(%%rbp), %%eax          # EAX <- %d\n"
122                     deslocamento n
123             with
124                 Not_found -> (* Assume que é uma variável global *)
125                     fprintf oc " movl %d(%%rip), %%eax          # EAX <- %d
126                                     \n"
127                                     n n
128             end
129         | _ -> failwith "emite_corpo: endereco nao implementado Return"
130     )
131     in emite_corpo oc tbl cod
132 | EndFun :: cod ->
133     let _ = fprintf oc " leave\n ret\n"
134     in cod
135 | Imprime endr :: cod ->
136     let _ =
137         (match endr with
138         | Temp n -> fprintf oc " printf %d\n" n
139         | _ -> failwith "emite_corpo: endereco nao implementado
140                     Imprime"
141         )
142     in emite_corpo oc tbl cod
143 | [] -> []
144
145
146
147 let emite_quadro_de_pilha oc cod nargs locais =
148     let cria_quadro_de_pilha tbl params locais =
149         let rec aloca_na_pilha inicio fim xs =
150             match xs with
151             | (x,nb) :: restante ->
152                 let endereco = inicio - nb in
153                 if endereco >= fim
154                 then let _ = Hashtbl.add tbl (Nome x) endereco in
155                     aloca_na_pilha endereco fim restante
156                 else aloca_na_pilha fim (fim - 16) xs
157             | [] -> fim
158         in
159             let fim_locais = aloca_na_pilha 0 (-16) locais in
160             let fim_params = aloca_na_pilha fim_locais (fim_locais - 16)
161                 params in
162             fprintf oc " subq %d, %%rsp\n" (- fim_params)
163         in
164         let rec separa_parametros n params cod =
165             if n <= 0 then (List.rev params, cod)
166             else match cod with
167             | Recebe (nome,tipo) :: cod ->
168                 let nb = num_bytes tipo in
169                 separa_parametros (n-1) ((nome,nb) :: params) cod
170             | _ -> (List.rev params, cod)
171         and separa_locais n locais cod =
172             if n <= 0 then (locais, cod)
173             else match cod with
174             | Local (nome,tipo) :: cod ->

```

```

174         let nb = num_bytes tipo in
175         separa_locais (n-1) ((nome,nb) :: locais) cod
176         | _ -> (locais, cod)
177     in
178     let (params, cod) = separa_parametros nargs [] cod in
179     let (locais, cod) = separa_locais nlocais [] cod in
180     let tbl_de_alocacao = Hashtbl.create 5 in
181     let _ = cria_quadro_de_pilha tbl_de_alocacao params locais in
182     tbl_de_alocacao
183
184 let emite_funcao oc cod nome nargs nlocais =
185     let _ = emite_prologo oc nome in
186     let tbl = emite_quadro_de_pilha oc cod nargs nlocais in
187     let registradores = ["edi"; "esi"; "edx"; "ecx"; "r8d"; "r9d"] in
188     let cod = move_parametros oc tbl cod registradores in
189     let cod = emite_corpo oc tbl cod in
190     let _ = emite_epilogo oc nome in
191     cod
192
193 let emite_rodape oc =
194     fprintf oc "
195     .ident  \"GCC: (Ubuntu 6.3.0-12ubuntu2) 6.3.0 20170406\"
196     .section .note.GNU-stack,\"\",@progbits
197 "
198
199 let rec emite_codigo oc cod =
200     match cod with
201     | BeginFun (nome, nargs, nlocais) :: cod ->
202         let cod = emite_funcao oc cod nome nargs nlocais in
203         emite_codigo oc cod
204     | _ -> ()
205
206 let gerador oc cod arq =
207     let _ = emite_cabecalho oc arq in
208     let cod = emite_global oc cod in
209     let _ = emite_codigo oc cod in
210     emite_rodape oc
211
212 let compila arq =
213     let oc = open_out (Filename.chop_suffix arq ".py" ^ ".s") in
214     let cod = traduz arq in
215     gerador stdout cod arq;
216     close_out oc

```

4.6.1 Teste

Listagem 4.25: teste.py

```

1 def main() -> int:
2     print("Teste")
3     while 1 == 1:
4         if 1 > 2:
5             return 1
6         elif 1 == 3:
7             return 4
8         else:
9             return 2
10    return 0

```


Para compilar fazemos:

Listagem 4.26: Terminal

```

1 $ ocamlbuild -use-ocamlfind -use-menhir -menhir "menhir --table" -package
   menhirLib cod3endTest.byte -r
2 Finished, 40 targets (40 cached) in 00:00:00.
3 $ rlwrap ocaml
4     OCaml version 4.07.1
5
6 Findlib has been successfully loaded. Additional directives:
7 #require "package";;      to load a package
8 #list;;                  to list the available packages
9 #camlp4o;;              to load camlp4 (standard syntax)
10 #camlp4r;;              to load camlp4 (revised syntax)
11 #predicates "p,q,...";;  to set these predicates
12 Topfind.reset();;        to force that packages will be reloaded
13 #thread;;               to enable threads
14
15 /Users/tatianefx/.opam/default/lib/menhirLib: added to search path
16 /Users/tatianefx/.opam/default/lib/menhirLib/menhirLib.cmo: loaded
17 # #use "gerador.ml";;
18 val num_bytes : Ast.tipo -> int = <fun>
19 val emite_cabecalho : out_channel -> string -> unit = <fun>
20 val emite_global :
21   out_channel -> Codigo.instrucao list -> Codigo.instrucao list = <fun>
22 val emite_prologo : out_channel -> string -> unit = <fun>
23 val emite_epilogo : out_channel -> string -> unit = <fun>
24 val move_parametros :
25   out_channel ->
26   (Codigo.endereco, int) Hashtbl.t ->
27   Codigo.instrucao list -> string list -> Codigo.instrucao list = <fun>
28 val emite_corpo :
29   out_channel ->
30   (Codigo.endereco, int) Hashtbl.t ->
31   Codigo.instrucao list -> Codigo.instrucao list = <fun>
32 val emite_quadro_de_pilha :
33   out_channel ->
34   Codigo.instrucao list -> int -> int -> (Codigo.endereco, int) Hashtbl.t
   =
35   <fun>
36 val emite_funcao :
37   out_channel ->
38   Codigo.instrucao list -> string -> int -> int -> Codigo.instrucao list =
39   <fun>
40 val emite_rodape : out_channel -> unit = <fun>
41 val emite_codigo : out_channel -> Codigo.instrucao list -> unit = <fun>
42 val gerador : out_channel -> Codigo.instrucao list -> string -> unit = <
   fun>
43 val compila : string -> unit = <fun>
44 # compila "tests/teste.py";;
45
46 .file "tests/teste.py"
47 .text
48
49 .globl main
50 .type main, @function
51 main:
52   pushq %rbp
53   movq  %rsp, %rbp

```

```

54  subq  $32, %rsp
55  printf $28
56 L16:
57  ifFalse $31 gotoL17:
58  ifFalse $34 gotoL12:
59  movl  35(%rip), %eax      # EAX <- 35
60 goto L15:
61 L12:
62  ifFalse $38 gotoL13:
63  movl  39(%rip), %eax      # EAX <- 39
64 goto L14:
65 L13:
66  movl  40(%rip), %eax      # EAX <- 40
67 L14:
68 L15:
69  movl  41(%rip), %eax      # EAX <- 41
70 goto L16:
71 L17:
72  leave
73  ret
74  .size main, .-main
75
76  .ident  "GCC: (Ubuntu 6.3.0-12ubuntu2) 6.3.0 20170406"
77  .section .note.GNU-stack,"",@progbits
78 - : unit = ()
79 #

```

4.7 Trabalhos Futuros

Como os módulos de *Código de 3 Endereços* e *Gerador* não estão completos, é preciso implementar a parte atribuição de valor às variáveis no arquivo **Cod3End.ml**.

Atualmente eles funcionam somente com constantes, como no exemplo do teste utilizando o arquivo **teste.py**.