

Construção de um compilador de Lua para Parrot Virtual Machine usando Objective Caml

Guilherme Pacheco de Oliveira

`guilherme.061@gmail.com`

Faculdade de Computação
Universidade Federal de Uberlândia

15 de agosto de 2016

Lista de Figuras

2.1	Instalando e testando LUA	8
2.2	Instalando e testando OCaml	9
2.3	Instalando e testando Parrot	10

Lista de Tabelas

Lista de Listagens

2.1	Output Simples em Parrot Assembly Language	10
2.2	Output Simples em Parrot Intermediate Representation	10
3.1	Programa nano 01 em Lua	13
3.2	Programa nano 01 em Ruby	13
3.3	Programa nano 01 em PIR	13
3.4	Programa nano 02 em Lua	14
3.5	Programa nano 02 em Ruby	14
3.6	Programa nano 02 em PIR	14
3.7	Programa nano 03 em Lua	15
3.8	Programa nano 03 em Ruby	15
3.9	Programa nano 03 em PIR	15
3.10	Programa nano 04 em Lua	16
3.11	Programa nano 04 em Ruby	16
3.12	Programa nano 04 em PIR	16
3.13	Programa nano 05 em Lua	17
3.14	Programa nano 05 em Ruby	17
3.15	Programa nano 05 em PIR	18
3.16	Programa nano 06 em Lua	19
3.17	Programa nano 06 em Ruby	19
3.18	Programa nano 06 em PIR	19
3.19	Programa nano 07 em Lua	20
3.20	Programa nano 07 em Ruby	21
3.21	Programa nano 07 em PIR	21
3.22	Programa nano 08 em Lua	22
3.23	Programa nano 08 em Ruby	23
3.24	Programa nano 08 em PIR	23
3.25	Programa nano 09 em Lua	25
3.26	Programa nano 09 em Ruby	26
3.27	Programa nano 10 em Lua	26
3.28	Programa nano 10 em Ruby	26
3.29	Programa nano 10 em PIR	26
3.30	Programa nano 11 em Lua	29
3.31	Programa nano 11 em Ruby	29
3.32	Programa nano 11 em PIR	29
3.33	Programa nano 12 em Lua	32
3.34	Programa nano 12 em Ruby	32
3.35	Programa nano 12 em PIR	33
3.36	Programa micro 01 em Lua	36
3.37	Programa micro 01 em Ruby	36
3.38	Programa micro 01 em PIR	37

3.39 Programa micro 02 em Lua	39
3.40 Programa micro 02 em Ruby	39
3.41 Programa micro 02 em PIR	40
3.42 Programa micro 03 em Lua	44
3.43 Programa micro 03 em Ruby	45
3.44 Programa micro 03 em PIR	45
3.45 Programa micro 04 em Lua	49
3.46 Programa micro 04 em Ruby	49
3.47 Programa micro 04 em PIR	50
3.48 Programa micro 05 em Lua	54
3.49 Programa micro 05 em Ruby	55
3.50 Programa micro 05 em PIR	55
3.51 Programa micro 06 em Lua	61
3.52 Programa micro 06 em Ruby	62
3.53 Programa micro 06 em PIR	62
3.54 Programa micro 07 em Lua	68
3.55 Programa micro 07 em Ruby	68
3.56 Programa micro 07 em PIR	69
3.57 Programa micro 08 em Lua	74
3.58 Programa micro 08 em Ruby	74
3.59 Programa micro 08 em PIR	75
3.60 Programa micro 09 em Lua	79
3.61 Programa micro 09 em Ruby	80
3.62 Programa micro 10 em Lua	80
3.63 Programa micro 10 em Ruby	80
3.64 Programa micro 10 em PIR	81
3.65 Programa micro 11 em Lua	85
3.66 Programa micro 11 em Ruby	86
3.67 Programa micro 11 em PIR	86

Sumário

Lista de Figuras	2
Lista de Tabelas	3
1 Introdução	7
2 Instalação dos componentes	8
2.1 Homebrew	8
2.2 Lua	8
2.2.1 Instalação e Teste	8
2.2.2 Informações sobre a linguagem Lua	9
2.3 Ocaml	9
2.3.1 Instalação e Teste	9
2.3.2 Informações sobre a linguagem OCaml	9
2.4 Parrot Virtual Machine	9
2.4.1 Instalação e Teste	9
2.4.2 Informações sobre a Parrot Virtual Machine	10
2.4.3 Parrot Assembly Language (PASM)	11
2.4.4 Parrot Intermediate Representation (PIR)	11
3 Estudando a geração de PIR por um compilador	12
3.1 Compilador e Utilização	12
3.1.1 Nano Programas	12
3.1.2 Micro Programas	36
4 Referências	93

Capítulo 1

Introdução

Este documento foi escrito para documentar o processo de instalação de todas as ferramentas necessárias para a construção de um compilador da Linguagem Lua para a máquina virtual Parrot, utilizando a linguagem Ocaml para fazer a implementação.

Um segundo objetivo é mostrar uma série de programas simples na linguagem Lua e sua versão na linguagem PASM, que é a linguagem assembly utilizada pela Parrot, afim de estabelecer um guia sobre a saída dos programas que passarão pelo compilador.

Outro objetivo é adquirir conhecimento sobre a linguagem Lua, ter um contato inicial com OCaml e conhecer como funciona a máquina virtual Parrot, suas linguagens de Assembly e bytecode e de compiladores já existentes

O Sistema Operacional utilizado é OS X El Capitan 10.11.6

Capítulo 2

Instalação dos componentes

2.1 Homebrew

Homebrew é um gerenciador de pacotes para Mac OS X, escrito em Ruby, e é responsável por instalar pacotes nos diretórios adequados e fazer adequadamente a configuração desses pacotes, instalá-lo facilita todo o processo de instalação dos componentes necessários.

Para instalar o homebrew basta digitar no terminal:

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

2.2 Lua

2.2.1 Instalação e Teste

Para instalar Lua através do homebrew, basta digitar no terminal:

```
\$ brew install lua
```

Resultado:

Figura 2.1: *Instalando e testando LUA*

```
oliveira@lua oliveira$ brew install lua
=> Downloading https://homebrew.bintray.com/bottles/lua-5.2.4.3.el_capitan.bott
##### 100.0%
=> Pouring Lua-5.2.4.3.el_capitan.bottle.tar.gz
=> Caveats
Please be aware due to the way Luarocks is designed any binaries installed
via Luarocks-5.2 AND 5.1 will overwrite each other in /usr/local/bin.

This is, for now, unavoidable. If this is troublesome for you, you can build
rocks with the --tree= command to a special, non-conflicting location and
then add that to your $PATH.
=> Summary
[0] /usr/local/Cellar/lua/5.2.4.3: 143 files, 697.3K
oliveira@lua oliveira$ lua
Lua 5.2.4 Copyright (C) 1994-2015 Lua.org, PUC-Rio
> print("Hello World")
Hello World
> ^D
oliveira@lua oliveira$
```


2.2.2 Informações sobre a linguagem Lua

A principal referência para Lua é a documentação em seu site oficial [1]. Lua é uma linguagem de programação de extensão, projetada para dar suporte à outras linguagens de programação procedimental e planejada para ser usada como uma linguagem de script leve e facilmente embarcável, é implementada em C.

2.3 Ocaml

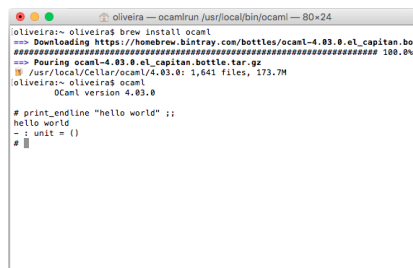
2.3.1 Instalação e Teste

Novamente através do homebrew, basta digitar:

```
\$ brew install ocaml
```

Resultado:

Figura 2.2: *Instalando e testando OCaml*



```
oliveira~ oliveira$ brew install ocaml
=> Downloading https://homebrew.bintray.com/bottles/ocaml-4.03.0.el_capitan.bot
=> Pouring ocaml-4.03.0.el_capitan.bottle.tar.gz
=> /usr/local/Cellar/ocaml/4.03.0: 1,641 files, 173.7M
oliveira~ oliveira$ ocaml
OCaml version 4.03.0

# print_endline "hello world" ;;
hello world
- : unit = ()
#
```

2.3.2 Informações sobre a linguagem OCaml

A documentação oficial do OCaml [2] possui manuais, licenças, documentos e algumas dicas sobre como programar adequadamente na linguagem. OCaml é uma linguagem de programação funcional, imperativa e orientada à objetos.

2.4 Parrot Virtual Machine

2.4.1 Instalação e Teste

Digitar no Terminal:

```
\$ brew install parrot
```

Resultado:

Figura 2.3: Instalando e testando Parrot

```

oliveira ~ - bash -- 80x24
Last login: Tue Aug 9 19:01:04 on ttys000
oliveira:~ oliveira$ brew install parrot
=> Downloading https://homebrew.bintray.com/bottles/parrot-8.1.0.el_capitan.bot
##### 100.0%
=> Pouring parrot-8.1.0.el_capitan.bottle.tar.gz
  /usr/local/Cellar/parrot/8.1.0: 706 files, 28.4M
oliveira:~ oliveira$ parrot
parrot -[acOhrtWxy.] -[Ddt][HEXFLAGS] [-O[level]] [-[LIX] path] [-R runcore] [-o
FILE] <file> -args
oliveira:~ oliveira$

```

2.4.2 Informações sobre a Parrot Virtual Machine

A máquina virtual Parrot é utilizada principalmente para linguagens dinâmicas como Perl, Python, Ruby e PHP, seu design foi originalmente feito para trabalhar com a versão 6 de Perl, mas seu uso foi expandido como uma máquina virtual dinâmica e de propósito geral, apta a lidar com qualquer linguagem de programação de alto nível. [3]

Parrot pode ser programada em diversas linguagens, os dois mais utilizados são: Parrot Assembly Language (PASM): É a linguagem de mais baixo nível utilizada pela Parrot, muito similar a um assembly tradicional. Parrot Intermediate Representation(PIR): De mais alto nível que PASM, também um pouco mais facil de se utilizar e mais utilizada.

Fazendo alguns testes com PASM e PIR:

Listagem 2.1: Output Simples em Parrot Assembly Language

```

1 say "Here are the news about Parrots."
2 end

```

Para executar o código:

```
\$ parrot news.pasm
```

Listagem 2.2: Output Simples em Parrot Intermediate Representation

```

1 .sub main :main
2   print "No parrots were involved in an accident on the M1 today...\n"
3 .end

```

Para executar o código:

```
\$ parrot hello.pir
```

Os arquivos PASM e PIR são convertidos para Parrot Bytecode (PBC) e somente então são executados pela máquina virtual, é possível obter o arquivo .pbc através comando:

```
\$ parrot -o output.pbc input.pasm
```

Apesar da documentação oficial enfatizar que PIR é mais utilizado e mais recomendado para o desenvolvimento de compiladores para Parrot, o alvo será a linguagem Assembly PASM.

2.4.3 Parrot Assembly Language (PASM)

A linguagem PASM é muito similar a um assembly tradicional, com exceção do fato de que algumas instruções permitem o acesso a algumas funções dinâmicas de alto nível do sistema Parrot.

Parrot é uma máquina virtual baseada em registradores, há um número ilimitado de registradores que não precisam ser instanciados antes de serem utilizados, a máquina virtual se certifica de criar os registradores de acordo com a sua necessidade, tal como fazer a reutilização e se livrar de registradores que não estão mais sendo utilizados, todos os registradores começam com o símbolo "\$" e existem 4 tipos de dados, cada um com suas regras:

Strings: Registradores de strings começam com um S, por exemplo: "\$S10"

Inteiros: Registradores de inteiros começam com um I, por exemplo: "\$I10"

Número: Registradores de números de ponto flutuante, começam com a letra N, por exemplo: "\$N10"

PMC: São tipos de dados utilizados em orientação a objetos, podem ser utilizados para guardar vários tipos de dados, começam com a letra P, por exemplo: "\$P10"

Para mais referências sobre PASM, consultar [4].

2.4.4 Parrot Intermediate Representation (PIR)

A maioria dos compiladores possuem como alvo o PIR, inclusive o que será utilizado para estudar qual o comportamento um compilador deve ter ao gerar o assembly. A própria máquina virtual Parrot possui um módulo intermediário capaz de interpretar a linguagem PIR e gerar o bytecode ou o próprio assembly (PASM), além disso, existem compiladores capazes de realizar a mesma tarefa.

PIR é de nível mais alto que assembly mas ainda muito próximo do nível de máquina, o principal benefício é a facilidade em programar em PIR em comparação com a programação em PASM, além disso, ela foi feita para compiladores de linguagens de alto nível gerarem código PIR para trabalhar com a máquina Parrot. Mais informações sobre PIR e sua sintaxe podem ser encontradas em [5].

Capítulo 3

Estudando a geração de PIR por um compilador

3.1 Compilador e Utilização

O compilador que será utilizado será o Cardinal [6], é um compilador da linguagem Ruby para a máquina virtual Parrot capaz de gerar código o código intermediário (PIR) como saída.

A documentação do compilador é simples e clara, para baixar o compilador basta digitar no terminal:

```
\$ git clone git://github.com/parrot/cardinal.git
```

Entre as várias opções de instalação, é possível fazê-la utilizando do próprio parrot, para isso basta entrar na pasta onde foi baixado o Cardinal e digitar:

```
\$ winxed setup.winxed build
```

Para compilar é necessário estar na pasta de instalação e o comando é:

```
\$ parrot cardinal.pbc [arquivo].rb
```

Sendo o arquivo o diretório do arquivo Ruby que se deseja executar, para gerar o PIR o comando é:

```
\$ parrot cardinal.pbc -o [output].pir --target=pir [arquivo].rb
```

Sendo output o diretório onde será salvo o arquivo PIR.

3.1.1 Nano Programas

Nano 01

Listagem 3.1: Programa nano 01 em Lua

```
1 -- Listagem 1: Mo dulo mi nimo que caracteriza um programa
```

Listagem 3.2: Programa nano 01 em Ruby

```
1 # modulo minimo
```

Compilação do Código Ruby:

```
\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/nano01.pir --target=pir nano01.rb
```

Listagem 3.3: Programa nano 01 em PIR

```
1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471301651.1019")
6     .param pmc param_1002 :optional :named("!BLOCK")
7     .param int has_param_1002 :opt_flag
8 .annotate 'file', "nano01.rb"
9 .annotate 'line', 0
10     .const 'Sub' $P1004 = "11_1471301651.1019"
11     capture_lex $P1004
12 .annotate 'line', 1
13     if has_param_1002, optparam_13
14         new $P100, "Undef"
15         set param_1002, $P100
16     optparam_13:
17         .lex "!BLOCK", param_1002
18         .return ()
19 .end
20
21
22 .HLL "cardinal"
23
24 .namespace []
25 .sub "" :load :init :subid("post12") :outer("10_1471301651.1019")
26 .annotate 'file', "nano01.rb"
27 .annotate 'line', 0
28     .const 'Sub' $P1001 = "10_1471301651.1019"
29     .local pmc block
30     set block, $P1001
31 .end
32
33
34 .HLL "parrot"
35
36 .namespace []
37 .sub "_block1003" :init :load :anon :subid("11_1471301651.1019") :outer("
    10_1471301651.1019")
38 .annotate 'file', "nano01.rb"
39 .annotate 'line', 0
40 $P0 = compreg "cardinal"
41 unless null $P0 goto have_cardinal
42 load_bytecode "cardinal.pbc"
43 have_cardinal:
```

```

44     .return ()
45 .end

```

Nano 02

Listagem 3.4: Programa nano 02 em Lua

```

1 -- Listagem 2: Declarac a o de uma varia vel
2
3 -- Em Lua, declaração de variaveis limitam apenas seu escopo
4 -- As variaveis podem ser local ou global
5 -- local: local x = 10 - precisam ser inicializadas
6 -- global: x = 10      - não precisam ser inicializadas
7 -- local x             é um programa aceito em lua (declaração de uma
    variavel local)
8 -- x                   não é um programa aceito em lua

```

Listagem 3.5: Programa nano 02 em Ruby

```

1 # declaracao de variavel ...

```

Compilação do Código Ruby:

```

\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/nano02.pir --target=pir nano02.rb

```

Listagem 3.6: Programa nano 02 em PIR

```

1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471302648.57833")
6     .param pmc param_1002 :optional :named("!BLOCK")
7     .param int has_param_1002 :opt_flag
8 .annotate 'file', "nano02.rb"
9 .annotate 'line', 0
10    .const 'Sub' $P1004 = "11_1471302648.57833"
11    capture_lex $P1004
12 .annotate 'line', 1
13    if has_param_1002, optparam_13
14    new $P100, "Undef"
15    set param_1002, $P100
16    optparam_13:
17    .lex "!BLOCK", param_1002
18    .return ()
19 .end
20
21
22 .HLL "cardinal"
23
24 .namespace []
25 .sub "" :load :init :subid("post12") :outer("10_1471302648.57833")
26 .annotate 'file', "nano02.rb"
27 .annotate 'line', 0
28    .const 'Sub' $P1001 = "10_1471302648.57833"
29    .local pmc block

```

3.1

```
30     set block, $P1001
31 .end
32
33
34 .HLL "parrot"
35
36 .namespace []
37 .sub "_block1003" :init :load :anon :subid("11_1471302648.57833") :outer("
    10_1471302648.57833")
38 .annotate 'file', "nano02.rb"
39 .annotate 'line', 0
40 $P0 = compreg "cardinal"
41 unless null $P0 goto have_cardinal
42 load_bytecode "cardinal.pbc"
43 have_cardinal:
44     .return ()
45 .end
```

Nano 03

Listagem 3.7: Programa nano 03 em Lua

```
1 -- Atribuicao de um inteiro a uma variavel
2 n = 1
```

Listagem 3.8: Programa nano 03 em Ruby

```
1 # atribuicao
2 n = 1
```

```
\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/nano03.pir --target=pir nano03.rb
```

Listagem 3.9: Programa nano 03 em PIR

```
1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471302675.85006")
6     .param pmc param_1002 :optional :named("!BLOCK")
7     .param int has_param_1002 :opt_flag
8 .annotate 'file', "nano03.rb"
9 .annotate 'line', 0
10     .const 'Sub' $P1005 = "11_1471302675.85006"
11     capture_lex $P1005
12 .annotate 'line', 1
13     if has_param_1002, optparam_13
14         new $P100, "Undef"
15         set param_1002, $P100
16     optparam_13:
17         .lex "!BLOCK", param_1002
18 .annotate 'line', 2
19     new $P101, "CardinalInteger"
20     assign $P101, 1
21     set $P1003, $P101
22     .lex "n", $P1003
```

```

23 .annotate 'line', 1
24     .return ($P1003)
25 .end
26
27
28 .HLL "cardinal"
29
30 .namespace []
31 .sub "" :load :init :subid("post12") :outer("10_1471302675.85006")
32 .annotate 'file', "nano03.rb"
33 .annotate 'line', 0
34     .const 'Sub' $P1001 = "10_1471302675.85006"
35     .local pmc block
36     set block, $P1001
37 .end
38
39
40 .HLL "parrot"
41
42 .namespace []
43 .sub "_block1004" :init :load :anon :subid("11_1471302675.85006") :outer("
    10_1471302675.85006")
44 .annotate 'file', "nano03.rb"
45 .annotate 'line', 0
46 $P0 = compreg "cardinal"
47 unless null $P0 goto have_cardinal
48 load_bytecode "cardinal.pbc"
49 have_cardinal:
50     .return ()
51 .end

```

Nano 04

Listagem 3.10: Programa nano 04 em Lua

```

1 -- Atribuição de uma soma de inteiros a uma variável
2 n = 1 + 2

```

Listagem 3.11: Programa nano 04 em Ruby

```

1 # atribuição de soma de inteiros a uma variável
2 n = 1 + 2

```

Compilação do Código Ruby:

```

\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/nano04.pir --target=pir nano04.rb

```

Listagem 3.12: Programa nano 04 em PIR

```

1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471302769.88451")
6     .param pmc param_1002 :optional :named("!BLOCK")
7     .param int has_param_1002 :opt_flag

```


3.1

```
8 .annotate 'file', "nano04.rb"
9 .annotate 'line', 0
10 .const 'Sub' $P1005 = "11_1471302769.88451"
11 capture_lex $P1005
12 .annotate 'line', 1
13 if has_param_1002, optparam_13
14 new $P100, "Undef"
15 set param_1002, $P100
16 optparam_13:
17 .lex "!BLOCK", param_1002
18 .annotate 'line', 2
19 new $P101, "CardinalInteger"
20 assign $P101, 1
21 new $P102, "CardinalInteger"
22 assign $P102, 2
23 $P103 = "infix:+"($P101, $P102)
24 set $P1003, $P103
25 .lex "n", $P1003
26 .annotate 'line', 1
27 .return ($P1003)
28 .end
29
30
31 .HLL "cardinal"
32
33 .namespace []
34 .sub "" :load :init :subid("post12") :outer("10_1471302769.88451")
35 .annotate 'file', "nano04.rb"
36 .annotate 'line', 0
37 .const 'Sub' $P1001 = "10_1471302769.88451"
38 .local pmc block
39 set block, $P1001
40 .end
41
42
43 .HLL "parrot"
44
45 .namespace []
46 .sub "_block1004" :init :load :anon :subid("11_1471302769.88451") :outer("
10_1471302769.88451")
47 .annotate 'file', "nano04.rb"
48 .annotate 'line', 0
49 $P0 = compreg "cardinal"
50 unless null $P0 goto have_cardinal
51 load_bytecode "cardinal.pbc"
52 have_cardinal:
53 .return ()
54 .end
```

Nano 05

Listagem 3.13: Programa nano 05 em Lua

```
1 -- Inclusa o do comando de impressa o
2 n = 2
3 print(n)
```

Listagem 3.14: Programa nano 05 em Ruby

```

1 # Inclusao do comando de impressao
2 n = 2
3 puts(n)

```

Compilação do Código Ruby:

```

\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/nano05.pir --target=pir nano05.rb

```

Listagem 3.15: Programa nano 05 em PIR

```

1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471302779.23036")
6     .param pmc param_1002 :optional :named("!BLOCK")
7     .param int has_param_1002 :opt_flag
8 .annotate 'file', "nano05.rb"
9 .annotate 'line', 0
10     .const 'Sub' $P1005 = "11_1471302779.23036"
11     capture_lex $P1005
12 .annotate 'line', 1
13     if has_param_1002, optparam_13
14         new $P100, "Undef"
15         set param_1002, $P100
16     optparam_13:
17         .lex "!BLOCK", param_1002
18 .annotate 'line', 2
19         new $P101, "CardinalInteger"
20         assign $P101, 2
21         set $P1003, $P101
22         .lex "n", $P1003
23 .annotate 'line', 3
24         get_hll_global $P102, "puts"
25         unless_null $P102, vivify_14
26         new $P102, "Undef"
27     vivify_14:
28         find_lex $P103, "n"
29         unless_null $P103, vivify_15
30         new $P103, "Undef"
31     vivify_15:
32         $P104 = $P102($P103)
33 .annotate 'line', 1
34     .return ($P104)
35 .end
36
37
38 .HLL "cardinal"
39
40 .namespace []
41 .sub "" :load :init :subid("post12") :outer("10_1471302779.23036")
42 .annotate 'file', "nano05.rb"
43 .annotate 'line', 0
44     .const 'Sub' $P1001 = "10_1471302779.23036"
45     .local pmc block
46     set block, $P1001
47 .end
48

```

3.1

```
49
50 .HLL "parrot"
51
52 .namespace []
53 .sub "_block1004" :init :load :anon :subid("11_1471302779.23036") :outer("
    10_1471302779.23036")
54 .annotate 'file', "nano05.rb"
55 .annotate 'line', 0
56 $P0 = compreg "cardinal"
57 unless null $P0 goto have_cardinal
58 load_bytecode "cardinal.pbc"
59 have_cardinal:
60     .return ()
61 .end
```

Nano 06

Listagem 3.16: Programa nano 06 em Lua

```
1 -- Listagem 6: Atribuição de uma subtração de inteiros a uma
    variável
2
3 n = 1 - 2
4 print(n)
```

Listagem 3.17: Programa nano 06 em Ruby

```
1 # Subtração de inteiros
2
3 n = 1 - 2
4 puts(n)
```

Compilação do Código Ruby:

```
\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/nano06.pir --target=pir nano06.rb
```

Listagem 3.18: Programa nano 06 em PIR

```
1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471302788.2895")
6     .param pmc param_1002 :optional :named("!BLOCK")
7     .param int has_param_1002 :opt_flag
8 .annotate 'file', "nano06.rb"
9 .annotate 'line', 0
10     .const 'Sub' $P1005 = "11_1471302788.2895"
11     capture_lex $P1005
12 .annotate 'line', 1
13     if has_param_1002, optparam_13
14         new $P100, "Undef"
15         set param_1002, $P100
16     optparam_13:
17         .lex "!BLOCK", param_1002
18 .annotate 'line', 3
```

```

19     new $P101, "CardinalInteger"
20     assign $P101, 1
21     new $P102, "CardinalInteger"
22     assign $P102, 2
23     $P103 = "infix:-"($P101, $P102)
24     set $P1003, $P103
25     .lex "n", $P1003
26 .annotate 'line', 4
27     get_hll_global $P104, "puts"
28     unless_null $P104, vivify_14
29     new $P104, "Undef"
30     vivify_14:
31         find_lex $P105, "n"
32         unless_null $P105, vivify_15
33         new $P105, "Undef"
34     vivify_15:
35         $P106 = $P104($P105)
36 .annotate 'line', 1
37     .return ($P106)
38 .end
39
40
41 .HLL "cardinal"
42
43 .namespace []
44 .sub "" :load :init :subid("post12") :outer("10_1471302788.2895")
45 .annotate 'file', "nano06.rb"
46 .annotate 'line', 0
47     .const 'Sub' $P1001 = "10_1471302788.2895"
48     .local pmc block
49     set block, $P1001
50 .end
51
52
53 .HLL "parrot"
54
55 .namespace []
56 .sub "_block1004" :init :load :anon :subid("11_1471302788.2895") :outer("
    10_1471302788.2895")
57 .annotate 'file', "nano06.rb"
58 .annotate 'line', 0
59 $P0 = compreg "cardinal"
60 unless null $P0 goto have_cardinal
61 load_bytecode "cardinal.pbc"
62 have_cardinal:
63     .return ()
64 .end

```

Nano 07

Listagem 3.19: Programa nano 07 em Lua

```

1 -- Listagem 7: Inclusa o do comando condicional
2 n = 1
3 if (n == 1)
4 then
5     print(n)
6 end

```

Listagem 3.20: Programa nano 07 em Ruby

```

1 # Inclusao de condicional
2 n = 1
3 if n == 1
4   puts(n)
5 end

```

Compilação do Código Ruby:

```

\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/nano07.pir --target=pir nano07.rb

```

Listagem 3.21: Programa nano 07 em PIR

```

1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471302800.66042")
6   .param pmc param_1002 :optional :named("!BLOCK")
7   .param int has_param_1002 :opt_flag
8   .annotate 'file', "nano07.rb"
9   .annotate 'line', 0
10    .const 'Sub' $P1009 = "12_1471302800.66042"
11    capture_lex $P1009
12    .const 'Sub' $P1006 = "11_1471302800.66042"
13    capture_lex $P1006
14 .annotate 'line', 1
15   if has_param_1002, optparam_14
16     new $P100, "Undef"
17     set param_1002, $P100
18   optparam_14:
19     .lex "!BLOCK", param_1002
20 .annotate 'line', 2
21   new $P101, "CardinalInteger"
22   assign $P101, 1
23   set $P1003, $P101
24   .lex "n", $P1003
25 .annotate 'line', 3
26   find_lex $P103, "n"
27   unless_null $P103, vivify_15
28   new $P103, "Undef"
29   vivify_15:
30     new $P104, "CardinalInteger"
31     assign $P104, 1
32     $P105 = "infix:=="($P103, $P104)
33     if $P105, if_1004
34       set $P102, $P105
35       goto if_1004_end
36   if_1004:
37 .annotate 'line', 4
38   .const 'Sub' $P1006 = "11_1471302800.66042"
39   capture_lex $P1006
40   $P110 = $P1006()
41   set $P102, $P110
42   if_1004_end:
43 .annotate 'line', 1
44   .return ($P102)

```

```

45 .end
46
47
48 .HLL "cardinal"
49
50 .namespace []
51 .sub "" :load :init :subid("post13") :outer("10_1471302800.66042")
52 .annotate 'file', "nano07.rb"
53 .annotate 'line', 0
54     .const 'Sub' $P1001 = "10_1471302800.66042"
55     .local pmc block
56     set block, $P1001
57 .end
58
59
60 .HLL "parrot"
61
62 .namespace []
63 .sub "_block1008" :init :load :anon :subid("12_1471302800.66042") :outer("
    10_1471302800.66042")
64 .annotate 'file', "nano07.rb"
65 .annotate 'line', 0
66 $P0 = compreg "cardinal"
67 unless null $P0 goto have_cardinal
68 load_bytecode "cardinal.pbc"
69 have_cardinal:
70     .return ()
71 .end
72
73
74 .HLL "cardinal"
75
76 .namespace []
77 .sub "_block1005" :anon :subid("11_1471302800.66042") :outer("10
    _1471302800.66042")
78     .param pmc param_1007 :optional :named("!BLOCK")
79     .param int has_param_1007 :opt_flag
80 .annotate 'file', "nano07.rb"
81 .annotate 'line', 4
82     if has_param_1007, optparam_16
83     new $P106, "Undef"
84     set param_1007, $P106
85 optparam_16:
86     .lex "!BLOCK", param_1007
87     get_hll_global $P107, "puts"
88     unless_null $P107, vivify_17
89     new $P107, "Undef"
90 vivify_17:
91     find_lex $P108, "n"
92     unless_null $P108, vivify_18
93     new $P108, "Undef"
94 vivify_18:
95     $P109 = $P107($P108)
96     .return ($P109)
97 .end

```

Listagem 3.22: Programa nano 08 em Lua

```

1 -- Listagem 8: Inclusa o do comando condicional com parte sena o
2
3 n = 1
4 if(n == 1)
5 then
6   print(n)
7 else
8   print("0")
9 end

```

Listagem 3.23: Programa nano 08 em Ruby

```

1 # Inclusao do comando condicional com parte senao
2
3 n = 1
4 if n == 1
5   puts(n)
6 else
7   puts("0")
8 end

```

Compilação do Código Ruby:

```

\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/nano08.pir --target=pir nano08.rb

```

Listagem 3.24: Programa nano 08 em PIR

```

1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471302808.29346")
6   .param pmc param_1002 :optional :named("!BLOCK")
7   .param int has_param_1002 :opt_flag
8 .annotate 'file', "nano08.rb"
9 .annotate 'line', 0
10  .const 'Sub' $P1012 = "13_1471302808.29346"
11  capture_lex $P1012
12  .const 'Sub' $P1009 = "12_1471302808.29346"
13  capture_lex $P1009
14  .const 'Sub' $P1006 = "11_1471302808.29346"
15  capture_lex $P1006
16 .annotate 'line', 1
17   if has_param_1002, optparam_15
18     new $P100, "Undef"
19     set param_1002, $P100
20   optparam_15:
21     .lex "!BLOCK", param_1002
22 .annotate 'line', 3
23   new $P101, "CardinalInteger"
24   assign $P101, 1
25   set $P1003, $P101
26   .lex "n", $P1003
27 .annotate 'line', 4
28   find_lex $P103, "n"
29   unless_null $P103, vivify_16

```

```

30     new $P103, "Undef"
31   vivify_16:
32     new $P104, "CardinalInteger"
33     assign $P104, 1
34     $P105 = "infix:=="($P103, $P104)
35     if $P105, if_1004
36   .annotate 'line', 6
37     .const 'Sub' $P1009 = "12_1471302808.29346"
38     capture_lex $P1009
39     $P115 = $P1009()
40     set $P102, $P115
41   .annotate 'line', 4
42     goto if_1004_end
43   if_1004:
44   .annotate 'line', 5
45     .const 'Sub' $P1006 = "11_1471302808.29346"
46     capture_lex $P1006
47     $P110 = $P1006()
48     set $P102, $P110
49   if_1004_end:
50   .annotate 'line', 1
51     .return ($P102)
52 .end
53
54
55 .HLL "cardinal"
56
57 .namespace []
58 .sub "" :load :init :subid("post14") :outer("10_1471302808.29346")
59 .annotate 'file', "nano08.rb"
60 .annotate 'line', 0
61     .const 'Sub' $P1001 = "10_1471302808.29346"
62     .local pmc block
63     set block, $P1001
64 .end
65
66
67 .HLL "parrot"
68
69 .namespace []
70 .sub "_block1011" :init :load :anon :subid("13_1471302808.29346") :outer("
    10_1471302808.29346")
71 .annotate 'file', "nano08.rb"
72 .annotate 'line', 0
73 $P0 = compreg "cardinal"
74 unless null $P0 goto have_cardinal
75 load_bytecode "cardinal.pbc"
76 have_cardinal:
77     .return ()
78 .end
79
80
81 .HLL "cardinal"
82
83 .namespace []
84 .sub "_block1008" :anon :subid("12_1471302808.29346") :outer("10
    _1471302808.29346")
85     .param pmc param_1010 :optional :named("!BLOCK")
86     .param int has_param_1010 :opt_flag

```


3.1

```
87 .annotate 'file', "nano08.rb"
88 .annotate 'line', 6
89     if has_param_1010, optparam_17
90         new $P111, "Undef"
91         set param_1010, $P111
92     optparam_17:
93         .lex "!BLOCK", param_1010
94 .annotate 'line', 7
95     get_hll_global $P112, "puts"
96     unless_null $P112, vivify_18
97     new $P112, "Undef"
98     vivify_18:
99         new $P113, "CardinalString"
100         assign $P113, "0"
101         $P114 = $P112($P113)
102 .annotate 'line', 6
103     .return ($P114)
104 .end
105
106
107 .HLL "cardinal"
108
109 .namespace []
110 .sub "_block1005" :anon :subid("11_1471302808.29346") :outer("10
    _1471302808.29346")
111     .param pmc param_1007 :optional :named("!BLOCK")
112     .param int has_param_1007 :opt_flag
113 .annotate 'file', "nano08.rb"
114 .annotate 'line', 5
115     if has_param_1007, optparam_19
116         new $P106, "Undef"
117         set param_1007, $P106
118     optparam_19:
119         .lex "!BLOCK", param_1007
120         get_hll_global $P107, "puts"
121         unless_null $P107, vivify_20
122         new $P107, "Undef"
123     vivify_20:
124         find_lex $P108, "n"
125         unless_null $P108, vivify_21
126         new $P108, "Undef"
127     vivify_21:
128         $P109 = $P107($P108)
129         .return ($P109)
130 .end
```

Nano 09

Listagem 3.25: Programa nano 09 em Lua

```
1 -- Listagem 9: Atribuição de duas operações aritméticas sobre
    inteiros a uma variável
2
3 n = 1 + 1 / 2
4 if (n == 1)
5 then
6     print(n)
7 else
```

```

8   print("0")
9 end

```

Listagem 3.26: Programa nano 09 em Ruby

```

1 # Atribuicao de duas operacoes aritmeticas sobre inteiros a uma variavel
2
3 n = 1 + 1 / 2
4
5 if n == 1
6   puts(n)
7 else
8   puts("0")
9 end

```

Compilação do Código Ruby:

```

\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/nano09.pir --target=pir nano09.rb

```

ERRO!

Nano 10

Listagem 3.27: Programa nano 10 em Lua

```

1 -- Listagem 10: Atribuicao de duas variaveis inteiras
2 n = 1
3 m = 2
4
5 if(n == m)
6 then
7   print(n)
8 else
9   print("0")
10 end

```

Listagem 3.28: Programa nano 10 em Ruby

```

1 # atribuicao de duas variaveis inteiras
2 n = 1
3 m = 2
4
5 if n == m
6   puts(n)
7 else
8   puts("0")
9 end

```

Compilação do Código Ruby:

```

\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/nano10.pir --target=pir nano10.rb

```

Listagem 3.29: Programa nano 10 em PIR

3.1

```
1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471302878.09803")
6   .param pmc param_1002 :optional :named("!BLOCK")
7   .param int has_param_1002 :opt_flag
8   .annotate 'file', "nano10.rb"
9   .annotate 'line', 0
10    .const 'Sub' $P1013 = "13_1471302878.09803"
11    capture_lex $P1013
12    .const 'Sub' $P1010 = "12_1471302878.09803"
13    capture_lex $P1010
14    .const 'Sub' $P1007 = "11_1471302878.09803"
15    capture_lex $P1007
16  .annotate 'line', 1
17    if has_param_1002, optparam_15
18    new $P100, "Undef"
19    set param_1002, $P100
20  optparam_15:
21    .lex "!BLOCK", param_1002
22  .annotate 'line', 2
23    new $P101, "CardinalInteger"
24    assign $P101, 1
25    set $P1003, $P101
26    .lex "n", $P1003
27  .annotate 'line', 3
28    new $P102, "CardinalInteger"
29    assign $P102, 2
30    set $P1004, $P102
31    .lex "m", $P1004
32  .annotate 'line', 5
33    find_lex $P104, "n"
34    unless_null $P104, vivify_16
35    new $P104, "Undef"
36  vivify_16:
37    find_lex $P105, "m"
38    unless_null $P105, vivify_17
39    new $P105, "Undef"
40  vivify_17:
41    $P106 = "infix:=="($P104, $P105)
42    if $P106, if_1005
43  .annotate 'line', 7
44    .const 'Sub' $P1010 = "12_1471302878.09803"
45    capture_lex $P1010
46    $P116 = $P1010()
47    set $P103, $P116
48  .annotate 'line', 5
49    goto if_1005_end
50  if_1005:
51  .annotate 'line', 6
52    .const 'Sub' $P1007 = "11_1471302878.09803"
53    capture_lex $P1007
54    $P111 = $P1007()
55    set $P103, $P111
56  if_1005_end:
57  .annotate 'line', 1
58    .return ($P103)
59 .end
```

```

60
61
62 .HLL "cardinal"
63
64 .namespace []
65 .sub "" :load :init :subid("post14") :outer("10_1471302878.09803")
66 .annotate 'file', "nano10.rb"
67 .annotate 'line', 0
68     .const 'Sub' $P1001 = "10_1471302878.09803"
69     .local pmc block
70     set block, $P1001
71 .end
72
73
74 .HLL "parrot"
75
76 .namespace []
77 .sub "_block1012" :init :load :anon :subid("13_1471302878.09803") :outer("
    10_1471302878.09803")
78 .annotate 'file', "nano10.rb"
79 .annotate 'line', 0
80 $P0 = compreg "cardinal"
81 unless null $P0 goto have_cardinal
82 load_bytecode "cardinal.pbc"
83 have_cardinal:
84     .return ()
85 .end
86
87
88 .HLL "cardinal"
89
90 .namespace []
91 .sub "_block1009" :anon :subid("12_1471302878.09803") :outer("10
    _1471302878.09803")
92     .param pmc param_1011 :optional :named("!BLOCK")
93     .param int has_param_1011 :opt_flag
94 .annotate 'file', "nano10.rb"
95 .annotate 'line', 7
96     if has_param_1011, optparam_18
97     new $P112, "Undef"
98     set param_1011, $P112
99     optparam_18:
100     .lex "!BLOCK", param_1011
101 .annotate 'line', 8
102     get_hll_global $P113, "puts"
103     unless_null $P113, vivify_19
104     new $P113, "Undef"
105     vivify_19:
106     new $P114, "CardinalString"
107     assign $P114, "0"
108     $P115 = $P113($P114)
109 .annotate 'line', 7
110     .return ($P115)
111 .end
112
113
114 .HLL "cardinal"
115
116 .namespace []

```

3.1

```
117 .sub "_block1006" :anon :subid("11_1471302878.09803") :outer("10
    _1471302878.09803")
118     .param pmc param_1008 :optional :named("!BLOCK")
119     .param int has_param_1008 :opt_flag
120     .annotate 'file', "nano10.rb"
121     .annotate 'line', 6
122     if has_param_1008, optparam_20
123     new $P107, "Undef"
124     set param_1008, $P107
125     optparam_20:
126     .lex "!BLOCK", param_1008
127     get_hll_global $P108, "puts"
128     unless_null $P108, vivify_21
129     new $P108, "Undef"
130     vivify_21:
131     find_lex $P109, "n"
132     unless_null $P109, vivify_22
133     new $P109, "Undef"
134     vivify_22:
135     $P110 = $P108($P109)
136     .return ($P110)
137 .end
```

Nano 11

Listagem 3.30: Programa nano 11 em Lua

```
1 -- Listagem 11: Introduc a o do comando de repetiç a o enquanto
2 n = 1
3 m = 2
4 x = 5
5
6 while(x > n)
7 do
8     n = n + m
9     print(n)
10 end
```

Listagem 3.31: Programa nano 11 em Ruby

```
1 # Introducao do comando de repeticao enquanto
2 n = 1
3 m = 2
4 x = 5
5
6 while x > n
7     n = n + m
8     puts(n)
9 end
```

Compilação do Código Ruby:

```
\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/nano11.pir --target=pir nano11.rb
```

Listagem 3.32: Programa nano 11 em PIR

```

1
2 .HLL "cardinal"
3
4 .namespace []
5 .include "except_types.pasm"
6 .sub "_block1000" :load :main :anon :subid("10_1471302890.93109")
7     .param pmc param_1002 :optional :named("!BLOCK")
8     .param int has_param_1002 :opt_flag
9 .annotate 'file', "nanoll.rb"
10 .annotate 'line', 0
11     .const 'Sub' $P1011 = "12_1471302890.93109"
12     capture_lex $P1011
13     .const 'Sub' $P1007 = "11_1471302890.93109"
14     capture_lex $P1007
15 .annotate 'line', 1
16     if has_param_1002, optparam_14
17     new $P100, "Undef"
18     set param_1002, $P100
19     optparam_14:
20     .lex "!BLOCK", param_1002
21 .annotate 'line', 2
22     new $P101, "CardinalInteger"
23     assign $P101, 1
24     set $P1003, $P101
25     .lex "n", $P1003
26 .annotate 'line', 3
27     new $P102, "CardinalInteger"
28     assign $P102, 2
29     set $P1004, $P102
30     .lex "m", $P1004
31 .annotate 'line', 4
32     new $P103, "CardinalInteger"
33     assign $P103, 5
34     set $P1005, $P103
35     .lex "x", $P1005
36 .annotate 'line', 6
37     new $P114, 'ExceptionHandler', [.CONTROL_LOOP_NEXT;.CONTROL_LOOP_REDO
38         ;.CONTROL_LOOP_LAST]
39     set_label $P114, loop1009_handler
40     push_eh $P114
41 loop1009_test:
42     find_lex $P104, "x"
43     unless_null $P104, vivify_15
44     new $P104, "Undef"
45 vivify_15:
46     find_lex $P105, "n"
47     unless_null $P105, vivify_16
48     new $P105, "Undef"
49 vivify_16:
50     $P106 = "infix:>($P104, $P105)
51     unless $P106, loop1009_done
52 loop1009_redo:
53 .annotate 'line', 7
54     .const 'Sub' $P1007 = "11_1471302890.93109"
55     capture_lex $P1007
56     $P1007()
57 loop1009_next:
58     goto loop1009_test
59 loop1009_handler:

```

3.1

```
59     .local pmc exception
60     .get_results (exception)
61     pop_upto_eh exception
62     getattribute $P115, exception, 'type'
63     eq $P115, .CONTROL_LOOP_NEXT, loop1009_next
64     eq $P115, .CONTROL_LOOP_REDO, loop1009_redo
65     loop1009_done:
66         pop_eh
67     .annotate 'line', 1
68     .return ($P106)
69 .end
70
71
72 .HLL "cardinal"
73
74 .namespace []
75 .sub "" :load :init :subid("post13") :outer("10_1471302890.93109")
76 .annotate 'file', "nano11.rb"
77 .annotate 'line', 0
78     .const 'Sub' $P1001 = "10_1471302890.93109"
79     .local pmc block
80     set block, $P1001
81 .end
82
83
84 .HLL "parrot"
85
86 .namespace []
87 .sub "_block1010" :init :load :anon :subid("12_1471302890.93109") :outer("
    10_1471302890.93109")
88 .annotate 'file', "nano11.rb"
89 .annotate 'line', 0
90 $P0 = compreg "cardinal"
91 unless null $P0 goto have_cardinal
92 load_bytecode "cardinal.pbc"
93 have_cardinal:
94     .return ()
95 .end
96
97
98 .HLL "cardinal"
99
100 .namespace []
101 .sub "_block1006" :anon :subid("11_1471302890.93109") :outer("10
    _1471302890.93109")
102     .param pmc param_1008 :optional :named("!BLOCK")
103     .param int has_param_1008 :opt_flag
104 .annotate 'file', "nano11.rb"
105 .annotate 'line', 7
106     if has_param_1008, optparam_17
107     new $P107, "Undef"
108     set param_1008, $P107
109 optparam_17:
110     .lex "!BLOCK", param_1008
111     find_lex $P108, "n"
112     unless_null $P108, vivify_18
113     new $P108, "Undef"
114 vivify_18:
115     find_lex $P109, "m"
```

```

116     unless_null $P109, vivify_19
117     new $P109, "Undef"
118     vivify_19:
119         $P110 = "infix:+"($P108, $P109)
120         store_lex "n", $P110
121     .annotate 'line', 8
122         get_hll_global $P111, "puts"
123         unless_null $P111, vivify_20
124         new $P111, "Undef"
125     vivify_20:
126         find_lex $P112, "n"
127         unless_null $P112, vivify_21
128         new $P112, "Undef"
129     vivify_21:
130         $P113 = $P111($P112)
131     .annotate 'line', 7
132     .return ($P113)
133 .end

```

Nano 12

Listagem 3.33: Programa nano 12 em Lua

```

1 -- Listagem 12: Comando condicional aninhado em um comando de
   repetição
2 n = 1
3 m = 2
4 x = 5
5
6 while (x > n)
7 do
8     if (n == m)
9     then
10         print(n)
11     else
12         print("0")
13     end
14     x = x - 1
15 end

```

Listagem 3.34: Programa nano 12 em Ruby

```

1 # Comando condicional aninhado em um comando de repetição
2 n = 1
3 m = 2
4 x = 5
5
6 while x > n
7     if n == m
8         puts(n)
9     else
10        puts("0")
11    end
12    x = x - 1
13 end

```

Compilação do Código Ruby:


```
\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/nano12.pir --target=pir nano12.rb
```

Listagem 3.35: Programa nano 12 em PIR

```
1
2 .HLL "cardinal"
3
4 .namespace []
5 .include "except_types.pasm"
6 .sub "_block1000" :load :main :anon :subid("10_1471302899.27719")
7   .param pmc param_1002 :optional :named("!BLOCK")
8   .param int has_param_1002 :opt_flag
9 .annotate 'file', "nano12.rb"
10 .annotate 'line', 0
11   .const 'Sub' $P1018 = "14_1471302899.27719"
12   capture_lex $P1018
13   .const 'Sub' $P1007 = "11_1471302899.27719"
14   capture_lex $P1007
15 .annotate 'line', 1
16   if has_param_1002, optparam_16
17     new $P100, "Undef"
18     set param_1002, $P100
19   optparam_16:
20     .lex "!BLOCK", param_1002
21 .annotate 'line', 2
22   new $P101, "CardinalInteger"
23   assign $P101, 1
24   set $P1003, $P101
25   .lex "n", $P1003
26 .annotate 'line', 3
27   new $P102, "CardinalInteger"
28   assign $P102, 2
29   set $P1004, $P102
30   .lex "m", $P1004
31 .annotate 'line', 4
32   new $P103, "CardinalInteger"
33   assign $P103, 5
34   set $P1005, $P103
35   .lex "x", $P1005
36 .annotate 'line', 6
37   new $P122, 'ExceptionHandler', [.CONTROL_LOOP_NEXT;.CONTROL_LOOP_REDO
38     ;.CONTROL_LOOP_LAST]
39   set_label $P122, loop1016_handler
40   push_eh $P122
41 loop1016_test:
42   find_lex $P104, "x"
43   unless_null $P104, vivify_17
44   new $P104, "Undef"
45 vivify_17:
46   find_lex $P105, "n"
47   unless_null $P105, vivify_18
48   new $P105, "Undef"
49 vivify_18:
50   $P106 = "infix:>"($P104, $P105)
51   unless $P106, loop1016_done
52 loop1016_redo:
53 .annotate 'line', 7
```

```

53     .const 'Sub' $P1007 = "11_1471302899.27719"
54     capture_lex $P1007
55     $P1007()
56     loop1016_next:
57         goto loop1016_test
58     loop1016_handler:
59         .local pmc exception
60         .get_results (exception)
61         pop_upto_eh exception
62         getattribute $P123, exception, 'type'
63         eq $P123, .CONTROL_LOOP_NEXT, loop1016_next
64         eq $P123, .CONTROL_LOOP_REDO, loop1016_redo
65     loop1016_done:
66         pop_eh
67     .annotate 'line', 1
68     .return ($P106)
69 .end
70
71
72 .HLL "cardinal"
73
74 .namespace []
75 .sub "" :load :init :subid("post15") :outer("10_1471302899.27719")
76 .annotate 'file', "nano12.rb"
77 .annotate 'line', 0
78     .const 'Sub' $P1001 = "10_1471302899.27719"
79     .local pmc block
80     set block, $P1001
81 .end
82
83
84 .HLL "parrot"
85
86 .namespace []
87 .sub "_block1017" :init :load :anon :subid("14_1471302899.27719") :outer("
    10_1471302899.27719")
88 .annotate 'file', "nano12.rb"
89 .annotate 'line', 0
90 $P0 = compreg "cardinal"
91 unless null $P0 goto have_cardinal
92 load_bytecode "cardinal.pbc"
93 have_cardinal:
94     .return ()
95 .end
96
97
98 .HLL "cardinal"
99
100 .namespace []
101 .sub "_block1006" :anon :subid("11_1471302899.27719") :outer("10
    _1471302899.27719")
102     .param pmc param_1008 :optional :named("!BLOCK")
103     .param int has_param_1008 :opt_flag
104 .annotate 'file', "nano12.rb"
105 .annotate 'line', 7
106     .const 'Sub' $P1014 = "13_1471302899.27719"
107     capture_lex $P1014
108     .const 'Sub' $P1011 = "12_1471302899.27719"
109     capture_lex $P1011

```

3.1

```
110     if has_param_1008, optparam_19
111     new $P107, "Undef"
112     set param_1008, $P107
113   optparam_19:
114     .lex "!BLOCK", param_1008
115     find_lex $P108, "n"
116     unless_null $P108, vivify_20
117     new $P108, "Undef"
118   vivify_20:
119     find_lex $P109, "m"
120     unless_null $P109, vivify_21
121     new $P109, "Undef"
122   vivify_21:
123     $P110 = "infix:=="($P108, $P109)
124     if $P110, if_1009
125   .annotate 'line', 9
126     .const 'Sub' $P1014 = "13_1471302899.27719"
127     capture_lex $P1014
128     $P1014()
129     goto if_1009_end
130   if_1009:
131   .annotate 'line', 8
132     .const 'Sub' $P1011 = "12_1471302899.27719"
133     capture_lex $P1011
134     $P1011()
135   if_1009_end:
136   .annotate 'line', 12
137     find_lex $P119, "x"
138     unless_null $P119, vivify_27
139     new $P119, "Undef"
140   vivify_27:
141     new $P120, "CardinalInteger"
142     assign $P120, 1
143     $P121 = "infix:--"($P119, $P120)
144     store_lex "x", $P121
145   .annotate 'line', 7
146     .return ($P121)
147 .end
148
149
150 .HLL "cardinal"
151
152 .namespace []
153 .sub "_block1013" :anon :subid("13_1471302899.27719") :outer("11
154   _1471302899.27719")
155   .param pmc param_1015 :optional :named("!BLOCK")
156   .param int has_param_1015 :opt_flag
157   .annotate 'file', "nano12.rb"
158   .annotate 'line', 9
159     if has_param_1015, optparam_22
160     new $P115, "Undef"
161     set param_1015, $P115
162   optparam_22:
163     .lex "!BLOCK", param_1015
164   .annotate 'line', 10
165     get_hll_global $P116, "puts"
166     unless_null $P116, vivify_23
167     new $P116, "Undef"
168   vivify_23:
```

```

168     new $P117, "CardinalString"
169     assign $P117, "0"
170     $P118 = $P116($P117)
171     .annotate 'line', 9
172     .return ($P118)
173 .end
174
175
176 .HLL "cardinal"
177
178 .namespace []
179 .sub "_block1010" :anon :subid("12_1471302899.27719") :outer("11
    _1471302899.27719")
180     .param pmc param_1012 :optional :named("!BLOCK")
181     .param int has_param_1012 :opt_flag
182     .annotate 'file', "nano12.rb"
183     .annotate 'line', 8
184     if has_param_1012, optparam_24
185     new $P111, "Undef"
186     set param_1012, $P111
187     optparam_24:
188     .lex "!BLOCK", param_1012
189     get_hll_global $P112, "puts"
190     unless_null $P112, vivify_25
191     new $P112, "Undef"
192     vivify_25:
193     find_lex $P113, "n"
194     unless_null $P113, vivify_26
195     new $P113, "Undef"
196     vivify_26:
197     $P114 = $P112($P113)
198     .return ($P114)
199 .end

```

3.1.2 Micro Programas

Micro 01

Listagem 3.36: Programa micro 01 em Lua

```

1 -- Listagem 13: Converte graus Celsius para Fahrenheit
2
3 -- [[ Func a o: Ler uma temperatura em graus Celsius e apresenta -la
    convertida em graus Fahrenheit. A fo rmula de conversao é : F=(9*C
    +160) / 5, sendo F a temperatura em Fahrenheit e C a temperatura em
    Celsius. --]]
4
5 print("Tabela de Conversão: Celsius -> Fahrenheit")
6 print("Digite a Temperatura em Celsius: ")
7 cel = io.read("*number")
8 far = (9*cel+160)/5
9 print("A nova temperatura é:", far)

```

Listagem 3.37: Programa micro 01 em Ruby

```

1 # conversao de graus celsius para fahrenheit
2

```

3.1

```
3 puts("Tabela de Conversao: Celsius -> Fahrenheit")
4 puts("Digite a temperatura em Celsius: ")
5 cel = gets.chomp.to_i
6 far = (9*cel+160)/5
7 puts("A nova temperatura e: #{far}")
```

Compilação do Código Ruby:

```
\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/micro01.pir --target=pir micro01.rb
```

Listagem 3.38: Programa micro 01 em PIR

```
1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471303683.49484")
6   .param pmc param_1002 :optional :named("!BLOCK")
7   .param int has_param_1002 :opt_flag
8   .annotate 'file', "micro01.rb"
9   .annotate 'line', 0
10  .const 'Sub' $P1009 = "12_1471303683.49484"
11  capture_lex $P1009
12  .const 'Sub' $P1006 = "11_1471303683.49484"
13  capture_lex $P1006
14 .annotate 'line', 1
15   if has_param_1002, optparam_14
16     new $P100, "Undef"
17     set param_1002, $P100
18   optparam_14:
19     .lex "!BLOCK", param_1002
20 .annotate 'line', 3
21   get_hll_global $P101, "puts"
22   unless_null $P101, vivify_15
23   new $P101, "Undef"
24 vivify_15:
25   new $P102, "CardinalString"
26   assign $P102, "Tabela de Conversao: Celsius -> Fahrenheit"
27   $P101($P102)
28 .annotate 'line', 4
29   get_hll_global $P103, "puts"
30   unless_null $P103, vivify_16
31   new $P103, "Undef"
32 vivify_16:
33   new $P104, "CardinalString"
34   assign $P104, "Digite a temperatura em Celsius: "
35   $P103($P104)
36 .annotate 'line', 5
37   get_hll_global $P105, "gets"
38   unless_null $P105, vivify_17
39   new $P105, "Undef"
40 vivify_17:
41   $P106 = $P105."chomp"()
42   $P107 = $P106."to_i"()
43   set $P1003, $P107
44   .lex "cel", $P1003
45 .annotate 'line', 6
46   new $P108, "CardinalInteger"
```

```

47     assign $P108, 9
48     find_lex $P109, "cel"
49     unless_null $P109, vivify_18
50     new $P109, "Undef"
51   vivify_18:
52     $P110 = "infix:*"($P108, $P109)
53     new $P111, "CardinalInteger"
54     assign $P111, 160
55     $P112 = "infix:+"($P110, $P111)
56     $P113 = "circumfix:( )"($P112)
57     new $P114, "CardinalInteger"
58     assign $P114, 5
59     $P115 = "infix:/"($P113, $P114)
60     set $P1004, $P115
61     .lex "far", $P1004
62   .annotate 'line', 7
63     get_hll_global $P116, "puts"
64     unless_null $P116, vivify_19
65     new $P116, "Undef"
66   vivify_19:
67     new $P117, "CardinalString"
68     assign $P117, "A nova temperatura e: "
69     .const 'Sub' $P1006 = "11_1471303683.49484"
70     capture_lex $P1006
71     $S100 = $P1006()
72     concat $P120, $P117, $S100
73     $P121 = $P116($P120)
74   .annotate 'line', 1
75     .return ($P121)
76 .end
77
78
79 .HLL "cardinal"
80
81 .namespace []
82 .sub "" :load :init :subid("post13") :outer("10_1471303683.49484")
83 .annotate 'file', "micro01.rb"
84 .annotate 'line', 0
85     .const 'Sub' $P1001 = "10_1471303683.49484"
86     .local pmc block
87     set block, $P1001
88 .end
89
90
91 .HLL "parrot"
92
93 .namespace []
94 .sub "_block1008" :init :load :anon :subid("12_1471303683.49484") :outer("
    10_1471303683.49484")
95 .annotate 'file', "micro01.rb"
96 .annotate 'line', 0
97 $P0 = compreg "cardinal"
98 unless null $P0 goto have_cardinal
99 load_bytecode "cardinal.pbc"
100 have_cardinal:
101     .return ()
102 .end
103
104

```

3.1

```
105 .HLL "cardinal"
106
107 .namespace []
108 .sub "_block1005" :anon :subid("11_1471303683.49484") :outer("10
    _1471303683.49484")
109     .param pmc param_1007 :optional :named("!BLOCK")
110     .param int has_param_1007 :opt_flag
111 .annotate 'file', "micro01.rb"
112 .annotate 'line', 7
113     if has_param_1007, optparam_20
114     new $P118, "Undef"
115     set param_1007, $P118
116     optparam_20:
117     .lex "!BLOCK", param_1007
118     find_lex $P119, "far"
119     unless_null $P119, vivify_21
120     new $P119, "Undef"
121     vivify_21:
122     .return ($P119)
123 .end
```

Micro 02

Listagem 3.39: Programa micro 02 em Lua

```
1 -- Listagem 14: Ler dois inteiros e decide qual é maior
2
3 --[[ Func a o : Escrever um algoritmo que leia dois valores inteiro
    distintos e informe qual e o maior --]]
4
5 print("Escreva o primeiro número:")
6 num1 = io.read("*number")
7 print("Escreva o segundo número:")
8 num2 = io.read("*number")
9
10 if (num1 > num2)
11 then
12     print("O primeiro número", num1, "é maior que o segundo", num2)
13 else
14     print("O segundo número", num2, "é maior que o primeiro", num1)
15 end
```

Listagem 3.40: Programa micro 02 em Ruby

```
1 # ler dois inteiros e decidir qual e maior
2
3 puts("Escreva o primeiro numero:")
4 num1 = gets.chomp.to_i
5 puts("Escreva o segundo numero:")
6 num2 = gets.chomp.to_i
7
8
9 if num1 > num2
10     puts("O primeiro numero, #{num1}, e maior que o segundo, #{num2}")
11 else
12     puts("O segundo numero, #{num2}, e maior que o primeiro, #{num1}")
13 end
```

Compilação do Código Ruby:

```
\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/micro02.pir --target=pir micro02.rb
```

Listagem 3.41: Programa micro 02 em PIR

```
1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471303693.22907")
6   .param pmc param_1002 :optional :named("!BLOCK")
7   .param int has_param_1002 :opt_flag
8 .annotate 'file', "micro02.rb"
9 .annotate 'line', 0
10  .const 'Sub' $P1025 = "17_1471303693.22907"
11  capture_lex $P1025
12  .const 'Sub' $P1016 = "14_1471303693.22907"
13  capture_lex $P1016
14  .const 'Sub' $P1007 = "11_1471303693.22907"
15  capture_lex $P1007
16 .annotate 'line', 1
17   if has_param_1002, optparam_19
18   new $P100, "Undef"
19   set param_1002, $P100
20   optparam_19:
21   .lex "!BLOCK", param_1002
22 .annotate 'line', 3
23   get_hll_global $P101, "puts"
24   unless_null $P101, vivify_20
25   new $P101, "Undef"
26   vivify_20:
27   new $P102, "CardinalString"
28   assign $P102, "Escreva o primeiro numero:"
29   $P101($P102)
30 .annotate 'line', 4
31   get_hll_global $P103, "gets"
32   unless_null $P103, vivify_21
33   new $P103, "Undef"
34   vivify_21:
35   $P104 = $P103."chomp"()
36   $P105 = $P104."to_i"()
37   set $P1003, $P105
38   .lex "num1", $P1003
39 .annotate 'line', 5
40   get_hll_global $P106, "puts"
41   unless_null $P106, vivify_22
42   new $P106, "Undef"
43   vivify_22:
44   new $P107, "CardinalString"
45   assign $P107, "Escreva o segundo numero:"
46   $P106($P107)
47 .annotate 'line', 6
48   get_hll_global $P108, "gets"
49   unless_null $P108, vivify_23
50   new $P108, "Undef"
51   vivify_23:
52   $P109 = $P108."chomp"()
53   $P110 = $P109."to_i"()
```


3.1

```
54     set $P1004, $P110
55     .lex "num2", $P1004
56 .annotate 'line', 9
57     find_lex $P112, "num1"
58     unless_null $P112, vivify_24
59     new $P112, "Undef"
60 vivify_24:
61     find_lex $P113, "num2"
62     unless_null $P113, vivify_25
63     new $P113, "Undef"
64 vivify_25:
65     $P114 = "infix:>"($P112, $P113)
66     if $P114, if_1005
67 .annotate 'line', 11
68     .const 'Sub' $P1016 = "14_1471303693.22907"
69     capture_lex $P1016
70     $P140 = $P1016()
71     set $P111, $P140
72 .annotate 'line', 9
73     goto if_1005_end
74 if_1005:
75 .annotate 'line', 10
76     .const 'Sub' $P1007 = "11_1471303693.22907"
77     capture_lex $P1007
78     $P127 = $P1007()
79     set $P111, $P127
80 if_1005_end:
81 .annotate 'line', 1
82     .return ($P111)
83 .end
84
85
86 .HLL "cardinal"
87
88 .namespace []
89 .sub "" :load :init :subid("post18") :outer("10_1471303693.22907")
90 .annotate 'file', "micro02.rb"
91 .annotate 'line', 0
92     .const 'Sub' $P1001 = "10_1471303693.22907"
93     .local pmc block
94     set block, $P1001
95 .end
96
97
98 .HLL "parrot"
99
100 .namespace []
101 .sub "_block1024" :init :load :anon :subid("17_1471303693.22907") :outer("
    10_1471303693.22907")
102 .annotate 'file', "micro02.rb"
103 .annotate 'line', 0
104 $P0 = compreg "cardinal"
105 unless null $P0 goto have_cardinal
106 load_bytecode "cardinal.pbc"
107 have_cardinal:
108     .return ()
109 .end
110
111
```

```

112 .HLL "cardinal"
113
114 .namespace []
115 .sub "_block1015" :anon :subid("14_1471303693.22907") :outer("10
    _1471303693.22907")
116     .param pmc param_1017 :optional :named("!BLOCK")
117     .param int has_param_1017 :opt_flag
118 .annotate 'file', "micro02.rb"
119 .annotate 'line', 11
120     .const 'Sub' $P1022 = "16_1471303693.22907"
121     capture_lex $P1022
122     .const 'Sub' $P1019 = "15_1471303693.22907"
123     capture_lex $P1019
124     if has_param_1017, optparam_26
125     new $P128, "Undef"
126     set param_1017, $P128
127     optparam_26:
128     .lex "!BLOCK", param_1017
129 .annotate 'line', 12
130     get_hll_global $P129, "puts"
131     unless_null $P129, vivify_27
132     new $P129, "Undef"
133     vivify_27:
134     new $P130, "CardinalString"
135     assign $P130, "O segundo numero, "
136     .const 'Sub' $P1019 = "15_1471303693.22907"
137     capture_lex $P1019
138     $S102 = $P1019()
139     concat $P133, $P130, $S102
140     new $P134, "CardinalString"
141     assign $P134, ", e maior que o primeiro, "
142     concat $P135, $P133, $P134
143     .const 'Sub' $P1022 = "16_1471303693.22907"
144     capture_lex $P1022
145     $S103 = $P1022()
146     concat $P138, $P135, $S103
147     $P139 = $P129($P138)
148 .annotate 'line', 11
149     .return ($P139)
150 .end
151
152
153 .HLL "cardinal"
154
155 .namespace []
156 .sub "_block1018" :anon :subid("15_1471303693.22907") :outer("14
    _1471303693.22907")
157     .param pmc param_1020 :optional :named("!BLOCK")
158     .param int has_param_1020 :opt_flag
159 .annotate 'file', "micro02.rb"
160 .annotate 'line', 12
161     if has_param_1020, optparam_28
162     new $P131, "Undef"
163     set param_1020, $P131
164     optparam_28:
165     .lex "!BLOCK", param_1020
166     find_lex $P132, "num2"
167     unless_null $P132, vivify_29
168     new $P132, "Undef"

```

3.1

```
169   vivify_29:
170     .return ($P132)
171 .end
172
173
174 .HLL "cardinal"
175
176 .namespace []
177 .sub "_block1021" :anon :subid("16_1471303693.22907") :outer("14
    _1471303693.22907")
178   .param pmc param_1023 :optional :named("!BLOCK")
179   .param int has_param_1023 :opt_flag
180 .annotate 'file', "micro02.rb"
181 .annotate 'line', 12
182   if has_param_1023, optparam_30
183     new $P136, "Undef"
184     set param_1023, $P136
185   optparam_30:
186     .lex "!BLOCK", param_1023
187     find_lex $P137, "num1"
188     unless_null $P137, vivify_31
189     new $P137, "Undef"
190   vivify_31:
191     .return ($P137)
192 .end
193
194
195 .HLL "cardinal"
196
197 .namespace []
198 .sub "_block1006" :anon :subid("11_1471303693.22907") :outer("10
    _1471303693.22907")
199   .param pmc param_1008 :optional :named("!BLOCK")
200   .param int has_param_1008 :opt_flag
201 .annotate 'file', "micro02.rb"
202 .annotate 'line', 10
203   .const 'Sub' $P1013 = "13_1471303693.22907"
204   capture_lex $P1013
205   .const 'Sub' $P1010 = "12_1471303693.22907"
206   capture_lex $P1010
207   if has_param_1008, optparam_32
208     new $P115, "Undef"
209     set param_1008, $P115
210   optparam_32:
211     .lex "!BLOCK", param_1008
212     get_hll_global $P116, "puts"
213     unless_null $P116, vivify_33
214     new $P116, "Undef"
215   vivify_33:
216     new $P117, "CardinalString"
217     assign $P117, "O primeiro numero, "
218     .const 'Sub' $P1010 = "12_1471303693.22907"
219     capture_lex $P1010
220     $S100 = $P1010()
221     concat $P120, $P117, $S100
222     new $P121, "CardinalString"
223     assign $P121, ", e maior que o segundo, "
224     concat $P122, $P120, $P121
225     .const 'Sub' $P1013 = "13_1471303693.22907"
```

```

226     capture_lex $P1013
227     $$S101 = $P1013()
228     concat $P125, $P122, $$S101
229     $P126 = $P116($P125)
230     .return ($P126)
231 .end
232
233
234 .HLL "cardinal"
235
236 .namespace []
237 .sub "_block1009" :anon :subid("12_1471303693.22907") :outer("11
    _1471303693.22907")
238     .param pmc param_1011 :optional :named("!BLOCK")
239     .param int has_param_1011 :opt_flag
240 .annotate 'file', "micro02.rb"
241 .annotate 'line', 10
242     if has_param_1011, optparam_34
243     new $P118, "Undef"
244     set param_1011, $P118
245     optparam_34:
246     .lex "!BLOCK", param_1011
247     find_lex $P119, "num1"
248     unless_null $P119, vivify_35
249     new $P119, "Undef"
250     vivify_35:
251     .return ($P119)
252 .end
253
254
255 .HLL "cardinal"
256
257 .namespace []
258 .sub "_block1012" :anon :subid("13_1471303693.22907") :outer("11
    _1471303693.22907")
259     .param pmc param_1014 :optional :named("!BLOCK")
260     .param int has_param_1014 :opt_flag
261 .annotate 'file', "micro02.rb"
262 .annotate 'line', 10
263     if has_param_1014, optparam_36
264     new $P123, "Undef"
265     set param_1014, $P123
266     optparam_36:
267     .lex "!BLOCK", param_1014
268     find_lex $P124, "num2"
269     unless_null $P124, vivify_37
270     new $P124, "Undef"
271     vivify_37:
272     .return ($P124)
273 .end

```

Micro 03

Listagem 3.42: Programa micro 03 em Lua

```

1 -- Lê um número e verifica se ele está entre 100 e 200
2 --[[ Função: Faça um algoritmo que receba um número e diga se este número
    está no intervalo entre 100 e 200 --]]

```

3.1

```
3
4 print("Digite um número:")
5 numero = io.read("*number")
6
7 if(numero >= 100)
8 then
9     if(numero <= 200)
10    then
11        print("O número está no intervalo entre 100 e 200")
12    else
13        print("O número não está no intervalo entre 100 e 200")
14    end
15 else
16    print("O número não está no intervalo entre 100 e 200")
17 end
```

Listagem 3.43: Programa micro 03 em Ruby

```
1 # le um numero e verifica se ele esta entre 100 e 200
2
3 puts("Digite um numero:")
4 numero = gets.chomp.to_i
5
6 if numero >= 100
7     if numero <= 200
8         puts("O numero esta no intervalo entre 100 e 200")
9     else
10        puts("O numero nao esta no intervalo entre 100 e 200")
11    end
12 else
13    puts("O numero nao esta no intervalo entre 100 e 200")
14 end
```

Compilação do Código Ruby:

```
\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/micro03.pir --target=pir micro03.rb
```

Listagem 3.44: Programa micro 03 em PIR

```
1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471303701.96532")
6     .param pmc param_1002 :optional :named("!BLOCK")
7     .param int has_param_1002 :opt_flag
8 .annotate 'file', "micro03.rb"
9 .annotate 'line', 0
10    .const 'Sub' $P1019 = "15_1471303701.96532"
11    capture_lex $P1019
12    .const 'Sub' $P1016 = "14_1471303701.96532"
13    capture_lex $P1016
14    .const 'Sub' $P1006 = "11_1471303701.96532"
15    capture_lex $P1006
16 .annotate 'line', 1
17     if has_param_1002, optparam_17
18         new $P100, "Undef"
```

```

19     set param_1002, $P100
20     optparam_17:
21         .lex "!BLOCK", param_1002
22     .annotate 'line', 3
23         get_hll_global $P101, "puts"
24         unless_null $P101, vivify_18
25         new $P101, "Undef"
26     vivify_18:
27         new $P102, "CardinalString"
28         assign $P102, "Digite um numero:"
29         $P101($P102)
30     .annotate 'line', 4
31         get_hll_global $P103, "gets"
32         unless_null $P103, vivify_19
33         new $P103, "Undef"
34     vivify_19:
35         $P104 = $P103."chomp"()
36         $P105 = $P104."to_i"()
37         set $P1003, $P105
38         .lex "numero", $P1003
39     .annotate 'line', 6
40         find_lex $P107, "numero"
41         unless_null $P107, vivify_20
42         new $P107, "Undef"
43     vivify_20:
44         new $P108, "CardinalInteger"
45         assign $P108, 100
46         $P109 = "infix:>="($P107, $P108)
47         if $P109, if_1004
48     .annotate 'line', 12
49         .const 'Sub' $P1016 = "14_1471303701.96532"
50         capture_lex $P1016
51         $P130 = $P1016()
52         set $P106, $P130
53     .annotate 'line', 6
54         goto if_1004_end
55     if_1004:
56     .annotate 'line', 7
57         .const 'Sub' $P1006 = "11_1471303701.96532"
58         capture_lex $P1006
59         $P125 = $P1006()
60         set $P106, $P125
61     if_1004_end:
62     .annotate 'line', 1
63         .return ($P106)
64     .end
65
66
67     .HLL "cardinal"
68
69     .namespace []
70     .sub "" :load :init :subid("post16") :outer("10_1471303701.96532")
71     .annotate 'file', "micro03.rb"
72     .annotate 'line', 0
73         .const 'Sub' $P1001 = "10_1471303701.96532"
74         .local pmc block
75         set block, $P1001
76     .end
77

```

3.1

```
78
79 .HLL "parrot"
80
81 .namespace []
82 .sub "_block1018" :init :load :anon :subid("15_1471303701.96532") :outer("
    10_1471303701.96532")
83 .annotate 'file', "micro03.rb"
84 .annotate 'line', 0
85 $P0 = compreg "cardinal"
86 unless null $P0 goto have_cardinal
87 load_bytecode "cardinal.pbc"
88 have_cardinal:
89     .return ()
90 .end
91
92
93 .HLL "cardinal"
94
95 .namespace []
96 .sub "_block1015" :anon :subid("14_1471303701.96532") :outer("10
    _1471303701.96532")
97     .param pmc param_1017 :optional :named("!BLOCK")
98     .param int has_param_1017 :opt_flag
99 .annotate 'file', "micro03.rb"
100 .annotate 'line', 12
101     if has_param_1017, optparam_21
102     new $P126, "Undef"
103     set param_1017, $P126
104     optparam_21:
105     .lex "!BLOCK", param_1017
106 .annotate 'line', 13
107     get_hll_global $P127, "puts"
108     unless_null $P127, vivify_22
109     new $P127, "Undef"
110     vivify_22:
111     new $P128, "CardinalString"
112     assign $P128, "O numero nao esta no intervalo entre 100 e 200"
113     $P129 = $P127($P128)
114 .annotate 'line', 12
115     .return ($P129)
116 .end
117
118
119 .HLL "cardinal"
120
121 .namespace []
122 .sub "_block1005" :anon :subid("11_1471303701.96532") :outer("10
    _1471303701.96532")
123     .param pmc param_1007 :optional :named("!BLOCK")
124     .param int has_param_1007 :opt_flag
125 .annotate 'file', "micro03.rb"
126 .annotate 'line', 7
127     .const 'Sub' $P1013 = "13_1471303701.96532"
128     capture_lex $P1013
129     .const 'Sub' $P1010 = "12_1471303701.96532"
130     capture_lex $P1010
131     if has_param_1007, optparam_23
132     new $P110, "Undef"
133     set param_1007, $P110
```

```

134 optparam_23:
135   .lex "!BLOCK", param_1007
136   find_lex $P112, "numero"
137   unless_null $P112, vivify_24
138   new $P112, "Undef"
139   vivify_24:
140     new $P113, "CardinalInteger"
141     assign $P113, 200
142     $P114 = "infix:<="($P112, $P113)
143     if $P114, if_1008
144 .annotate 'line', 9
145   .const 'Sub' $P1013 = "13_1471303701.96532"
146   capture_lex $P1013
147   $P124 = $P1013()
148   set $P111, $P124
149 .annotate 'line', 7
150   goto if_1008_end
151   if_1008:
152 .annotate 'line', 8
153   .const 'Sub' $P1010 = "12_1471303701.96532"
154   capture_lex $P1010
155   $P119 = $P1010()
156   set $P111, $P119
157   if_1008_end:
158 .annotate 'line', 7
159   .return ($P111)
160 .end
161
162
163 .HLL "cardinal"
164
165 .namespace []
166 .sub "_block1012" :anon :subid("13_1471303701.96532") :outer("11
    _1471303701.96532")
167   .param pmc param_1014 :optional :named("!BLOCK")
168   .param int has_param_1014 :opt_flag
169 .annotate 'file', "micro03.rb"
170 .annotate 'line', 9
171   if has_param_1014, optparam_25
172   new $P120, "Undef"
173   set param_1014, $P120
174   optparam_25:
175     .lex "!BLOCK", param_1014
176 .annotate 'line', 10
177   get_hll_global $P121, "puts"
178   unless_null $P121, vivify_26
179   new $P121, "Undef"
180   vivify_26:
181     new $P122, "CardinalString"
182     assign $P122, "O numero nao esta no intervalo entre 100 e 200"
183     $P123 = $P121($P122)
184 .annotate 'line', 9
185     .return ($P123)
186 .end
187
188
189 .HLL "cardinal"
190
191 .namespace []

```


3.1

```
192 .sub "_block1009" :anon :subid("12_1471303701.96532") :outer("11
    _1471303701.96532")
193     .param pmc param_1011 :optional :named("!BLOCK")
194     .param int has_param_1011 :opt_flag
195     .annotate 'file', "micro03.rb"
196     .annotate 'line', 8
197     if has_param_1011, optparam_27
198     new $P115, "Undef"
199     set param_1011, $P115
200     optparam_27:
201     .lex "!BLOCK", param_1011
202     get_hll_global $P116, "puts"
203     unless_null $P116, vivify_28
204     new $P116, "Undef"
205     vivify_28:
206     new $P117, "CardinalString"
207     assign $P117, "O numero esta no intervalo entre 100 e 200"
208     $P118 = $P116($P117)
209     .return ($P118)
210 .end
```

Micro 04

Listagem 3.45: Programa micro 04 em Lua

```
1 -- Listagem 16: Lê números e informa quais estão entre 10 e 150
2
3 --[[ Função: Ler 5 números e ao final informar quantos números estão no
    intervalo entre 10 (inclusive) e 150(inclusive) --]]
4
5 intervalo = 0
6
7 for x=1,5,1
8 do
9     print("Digite um número")
10    num = io.read("*number")
11    if(num >= 10)
12    then
13        if(num <= 150)
14        then
15            intervalo = intervalo + 1
16        end
17    end
18 end
19
20 print("Ao total, foram digitados",intervalo,"números no intervalo entre 10
    e 150")
```

Listagem 3.46: Programa micro 04 em Ruby

```
1 # Le numero e informa quais estao entre 10 e 150
2
3 intervalo = 0
4 for i in 1..5
5     puts("Digite um numero:")
6     num = gets.chomp.to_i
7     if num >= 10
8         if num <= 150
```

```

9         intervalo = intervalo + 1
10     end
11 end
12 end
13
14 puts("Ao total, foram digitados #{intervalo} numeros no intervalo entre 10
    e 150")

```

Compilação do Código Ruby:

```

\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/micro04.pir --target=pir micro04.rb

```

Listagem 3.47: Programa micro 04 em PIR

```

1
2 .HLL "cardinal"
3
4 .namespace []
5 .include "except_types.pasm"
6 .sub "_block1000" :load :main :anon :subid("10_1471303711.8864")
7     .param pmc param_1002 :optional :named("!BLOCK")
8     .param int has_param_1002 :opt_flag
9 .annotate 'file', "micro04.rb"
10 .annotate 'line', 0
11     .const 'Sub' $P1022 = "15_1471303711.8864"
12     capture_lex $P1022
13     .const 'Sub' $P1019 = "14_1471303711.8864"
14     capture_lex $P1019
15     .const 'Sub' $P1005 = "11_1471303711.8864"
16     capture_lex $P1005
17 .annotate 'line', 1
18     if has_param_1002, optparam_17
19         new $P100, "Undef"
20         set param_1002, $P100
21     optparam_17:
22         .lex "!BLOCK", param_1002
23 .annotate 'line', 3
24     new $P101, "CardinalInteger"
25     assign $P101, 0
26     set $P1003, $P101
27     .lex "intervalo", $P1003
28 .annotate 'line', 4
29     new $P103, "CardinalInteger"
30     assign $P103, 1
31     new $P104, "CardinalInteger"
32     assign $P104, 5
33     $P105 = "infix:.."($P103, $P104)
34     defined $I100, $P105
35     unless $I100, for_undef_18
36         iter $P102, $P105
37         new $P129, 'ExceptionHandler', [.CONTROL_LOOP_NEXT;.CONTROL_LOOP_REDO
            ;.CONTROL_LOOP_LAST]
38         set_label $P129, loop1017_handler
39         push_eh $P129
40     loop1017_test:
41         unless $P102, loop1017_done
42             shift $P106, $P102
43     loop1017_redo:

```

3.1

```
44 .annotate 'line', 5
45     .const 'Sub' $P1005 = "11_1471303711.8864"
46     capture_lex $P1005
47     $P1005($P106)
48     loop1017_next:
49         goto loop1017_test
50     loop1017_handler:
51         .local pmc exception
52         .get_results (exception)
53         pop_upto_eh exception
54         getattribute $P130, exception, 'type'
55         eq $P130, .CONTROL_LOOP_NEXT, loop1017_next
56         eq $P130, .CONTROL_LOOP_REDO, loop1017_redo
57     loop1017_done:
58         pop_eh
59     for_undef_18:
60 .annotate 'line', 14
61     get_hll_global $P131, "puts"
62     unless_null $P131, vivify_28
63     new $P131, "Undef"
64     vivify_28:
65     new $P132, "CardinalString"
66     assign $P132, "Ao total, foram digitados "
67     .const 'Sub' $P1019 = "14_1471303711.8864"
68     capture_lex $P1019
69     $S100 = $P1019()
70     concat $P135, $P132, $S100
71     new $P136, "CardinalString"
72     assign $P136, "numeros no intervalo entre 10 e 150"
73     concat $P137, $P135, $P136
74     $P138 = $P131($P137)
75 .annotate 'line', 1
76     .return ($P138)
77 .end
78
79
80 .HLL "cardinal"
81
82 .namespace []
83 .sub "":load:init:subid("post16"):outer("10_1471303711.8864")
84 .annotate 'file', "micro04.rb"
85 .annotate 'line', 0
86     .const 'Sub' $P1001 = "10_1471303711.8864"
87     .local pmc block
88     set block, $P1001
89 .end
90
91
92 .HLL "parrot"
93
94 .namespace []
95 .sub "_block1021":init:load:anon:subid("15_1471303711.8864"):outer("10_1471303711.8864")
96 .annotate 'file', "micro04.rb"
97 .annotate 'line', 0
98 $P0 = compreg "cardinal"
99 unless null $P0 goto have_cardinal
100 load_bytecode "cardinal.pbc"
101 have_cardinal:
```

```

102     .return ()
103 .end
104
105
106 .HLL "cardinal"
107
108 .namespace []
109 .sub "_block1004" :anon :subid("11_1471303711.8864") :outer("10
    _1471303711.8864")
110     .param pmc param_1007 :optional
111     .param int has_param_1007 :opt_flag
112     .param pmc param_1006 :optional :named("!BLOCK")
113     .param int has_param_1006 :opt_flag
114 .annotate 'file', "micro04.rb"
115 .annotate 'line', 5
116     .const 'Sub' $P1011 = "12_1471303711.8864"
117     capture_lex $P1011
118     if has_param_1006, optparam_19
119     new $P107, "Undef"
120     set param_1006, $P107
121     optparam_19:
122     .lex "!BLOCK", param_1006
123 .annotate 'line', 4
124     if has_param_1007, optparam_20
125     new $P108, "Undef"
126     set param_1007, $P108
127     optparam_20:
128     .lex "i", param_1007
129 .annotate 'line', 5
130     get_hll_global $P109, "puts"
131     unless_null $P109, vivify_21
132     new $P109, "Undef"
133     vivify_21:
134     new $P110, "CardinalString"
135     assign $P110, "Digite um numero:"
136     $P109($P110)
137 .annotate 'line', 6
138     get_hll_global $P111, "gets"
139     unless_null $P111, vivify_22
140     new $P111, "Undef"
141     vivify_22:
142     $P112 = $P111."chomp"()
143     $P113 = $P112."to_i"()
144     set $P1008, $P113
145     .lex "num", $P1008
146 .annotate 'line', 7
147     find_lex $P115, "num"
148     unless_null $P115, vivify_23
149     new $P115, "Undef"
150     vivify_23:
151     new $P116, "CardinalInteger"
152     assign $P116, 10
153     $P117 = "infix:>="($P115, $P116)
154     if $P117, if_1009
155     set $P114, $P117
156     goto if_1009_end
157     if_1009:
158 .annotate 'line', 8
159     .const 'Sub' $P1011 = "12_1471303711.8864"

```

3.1

```
160     capture_lex $P1011
161     $P128 = $P1011()
162     set $P114, $P128
163     if_1009_end:
164     .annotate 'line', 5
165     .return ($P114)
166 .end
167
168
169 .HLL "cardinal"
170
171 .namespace []
172 .sub "_block1010" :anon :subid("12_1471303711.8864") :outer("11
    _1471303711.8864")
173     .param pmc param_1012 :optional :named("!BLOCK")
174     .param int has_param_1012 :opt_flag
175     .annotate 'file', "micro04.rb"
176     .annotate 'line', 8
177     .const 'Sub' $P1015 = "13_1471303711.8864"
178     capture_lex $P1015
179     if has_param_1012, optparam_24
180     new $P118, "Undef"
181     set param_1012, $P118
182     optparam_24:
183     .lex "!BLOCK", param_1012
184     find_lex $P120, "num"
185     unless_null $P120, vivify_25
186     new $P120, "Undef"
187     vivify_25:
188     new $P121, "CardinalInteger"
189     assign $P121, 150
190     $P122 = "infix:<="($P120, $P121)
191     if $P122, if_1013
192     set $P119, $P122
193     goto if_1013_end
194     if_1013:
195     .annotate 'line', 9
196     .const 'Sub' $P1015 = "13_1471303711.8864"
197     capture_lex $P1015
198     $P127 = $P1015()
199     set $P119, $P127
200     if_1013_end:
201     .annotate 'line', 8
202     .return ($P119)
203 .end
204
205
206 .HLL "cardinal"
207
208 .namespace []
209 .sub "_block1014" :anon :subid("13_1471303711.8864") :outer("12
    _1471303711.8864")
210     .param pmc param_1016 :optional :named("!BLOCK")
211     .param int has_param_1016 :opt_flag
212     .annotate 'file', "micro04.rb"
213     .annotate 'line', 9
214     if has_param_1016, optparam_26
215     new $P123, "Undef"
216     set param_1016, $P123
```

```

217 optparam_26:
218   .lex "!BLOCK", param_1016
219   find_lex $P124, "intervalo"
220   unless_null $P124, vivify_27
221   new $P124, "Undef"
222 vivify_27:
223   new $P125, "CardinalInteger"
224   assign $P125, 1
225   $P126 = "infix:+"($P124, $P125)
226   store_lex "intervalo", $P126
227   .return ($P126)
228 .end
229
230
231 .HLL "cardinal"
232
233 .namespace []
234 .sub "_block1018" :anon :subid("14_1471303711.8864") :outer("10
    _1471303711.8864")
235   .param pmc param_1020 :optional :named("!BLOCK")
236   .param int has_param_1020 :opt_flag
237 .annotate 'file', "micro04.rb"
238 .annotate 'line', 14
239   if has_param_1020, optparam_29
240   new $P133, "Undef"
241   set param_1020, $P133
242 optparam_29:
243   .lex "!BLOCK", param_1020
244   find_lex $P134, "intervalo"
245   unless_null $P134, vivify_30
246   new $P134, "Undef"
247 vivify_30:
248   .return ($P134)
249 .end

```

Micro 05

Listagem 3.48: Programa micro 05 em Lua

```

1 -- Listagem 17: Lê strings e caracteres
2 --[ Função: Escrever um algoritmo que leia o nome e o sexo de 56 pessoas
   e informe o nome e se ela é homem ou mulher. No final informe o total
   de homens e mulheres --]
3
4 h = 0
5 m = 0
6 for x=1,5,1
7 do
8   print("Digite o nome: ")
9   nome = io.read()
10  print("H - Homem ou M - Mulher")
11  sexo = io.read()
12  if(sexo == 'H') then h = h + 1
13  elseif (sexo == 'M') then m = m + 1
14  else print("Sexo só pode ser H ou M!")
15  end
16 end
17

```

3.1

```
18 print("Foram inseridos",h,"homens")
19 print("Foram inseridas",m,"mulheres")
```

Listagem 3.49: Programa micro 05 em Ruby

```
1 # Le strings e caracteres
2
3 h = 0
4 m = 0
5
6 for i in 1..5
7   puts("Digite o nome:")
8   nome = gets.chomp
9   puts("H - homem ou M - mulher")
10  sexo = gets.chomp
11
12  if(sexo == 'H')
13    h = h + 1
14  elsif (sexo == 'M')
15    m = m + 1
16  else
17    puts("Sexo so pode ser H ou M!")
18  end
19 end
20
21 puts("Foram inseridos #{h} homens.")
22 puts("Foram inseridas #{m} mulheres.")
```

Compilação do Código Ruby:

```
\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/micro05.pir --target=pir micro05.rb
```

Listagem 3.50: Programa micro 05 em PIR

```
1
2 .HLL "cardinal"
3
4 .namespace []
5 .include "except_types.pasm"
6 .sub "_block1000" :load :main :anon :subid("10_1471303721.47548")
7   .param pmc param_1002 :optional :named("BLOCK")
8   .param int has_param_1002 :opt_flag
9 .annotate 'file', "micro05.rb"
10 .annotate 'line', 0
11   .const 'Sub' $P1030 = "17_1471303721.47548"
12   capture_lex $P1030
13   .const 'Sub' $P1027 = "16_1471303721.47548"
14   capture_lex $P1027
15   .const 'Sub' $P1024 = "15_1471303721.47548"
16   capture_lex $P1024
17   .const 'Sub' $P1006 = "11_1471303721.47548"
18   capture_lex $P1006
19 .annotate 'line', 1
20   if has_param_1002, optparam_19
21     new $P100, "Undef"
22     set param_1002, $P100
23   optparam_19:
```

```

24     .lex "!BLOCK", param_1002
25 .annotate 'line', 3
26     new $P101, "CardinalInteger"
27     assign $P101, 0
28     set $P1003, $P101
29     .lex "h", $P1003
30 .annotate 'line', 4
31     new $P102, "CardinalInteger"
32     assign $P102, 0
33     set $P1004, $P102
34     .lex "m", $P1004
35 .annotate 'line', 6
36     new $P104, "CardinalInteger"
37     assign $P104, 1
38     new $P105, "CardinalInteger"
39     assign $P105, 5
40     $P106 = "infix:.."($P104, $P105)
41     defined $I100, $P106
42     unless $I100, for_undef_20
43     iter $P103, $P106
44     new $P143, 'ExceptionHandler', [.CONTROL_LOOP_NEXT;.CONTROL_LOOP_REDO
45         ;.CONTROL_LOOP_LAST]
46     set_label $P143, loop1022_handler
47     push_eh $P143
48 loop1022_test:
49     unless $P103, loop1022_done
50     shift $P107, $P103
51 loop1022_redo:
52 .annotate 'line', 7
53     .const 'Sub' $P1006 = "11_1471303721.47548"
54     capture_lex $P1006
55     $P1006($P107)
56 loop1022_next:
57     goto loop1022_test
58 loop1022_handler:
59     .local pmc exception
60     .get_results (exception)
61     pop_upto_eh exception
62     getattribute $P144, exception, 'type'
63     eq $P144, .CONTROL_LOOP_NEXT, loop1022_next
64     eq $P144, .CONTROL_LOOP_REDO, loop1022_redo
65 loop1022_done:
66     pop_eh
67 for_undef_20:
68 .annotate 'line', 21
69     get_hll_global $P145, "puts"
70     unless_null $P145, vivify_35
71     new $P145, "Undef"
72 vivify_35:
73     new $P146, "CardinalString"
74     assign $P146, "Foram inseridos "
75     .const 'Sub' $P1024 = "15_1471303721.47548"
76     capture_lex $P1024
77     $$S100 = $P1024()
78     concat $P149, $P146, $$S100
79     new $P150, "CardinalString"
80     assign $P150, "homens."
81     concat $P151, $P149, $P150
82     $P145($P151)

```


3.1

```
82 .annotate 'line', 22
83     get_hll_global $P152, "puts"
84     unless_null $P152, vivify_38
85     new $P152, "Undef"
86     vivify_38:
87         new $P153, "CardinalString"
88         assign $P153, "Foram inseridas "
89         .const 'Sub' $P1027 = "16_1471303721.47548"
90         capture_lex $P1027
91         $S101 = $P1027()
92         concat $P156, $P153, $S101
93         new $P157, "CardinalString"
94         assign $P157, "mulheres."
95         concat $P158, $P156, $P157
96         $P159 = $P152($P158)
97     .annotate 'line', 1
98     .return ($P159)
99 .end
100
101
102 .HLL "cardinal"
103
104 .namespace []
105 .sub "" :load :init :subid("post18") :outer("10_1471303721.47548")
106 .annotate 'file', "micro05.rb"
107 .annotate 'line', 0
108     .const 'Sub' $P1001 = "10_1471303721.47548"
109     .local pmc block
110     set block, $P1001
111 .end
112
113
114 .HLL "parrot"
115
116 .namespace []
117 .sub "_block1029" :init :load :anon :subid("17_1471303721.47548") :outer("
    10_1471303721.47548")
118 .annotate 'file', "micro05.rb"
119 .annotate 'line', 0
120 $P0 = compreg "cardinal"
121 unless null $P0 goto have_cardinal
122 load_bytecode "cardinal.pbc"
123 have_cardinal:
124     .return ()
125 .end
126
127
128 .HLL "cardinal"
129
130 .namespace []
131 .sub "_block1005" :anon :subid("11_1471303721.47548") :outer("10
    _1471303721.47548")
132     .param pmc param_1008 :optional
133     .param int has_param_1008 :opt_flag
134     .param pmc param_1007 :optional :named("!BLOCK")
135     .param int has_param_1007 :opt_flag
136 .annotate 'file', "micro05.rb"
137 .annotate 'line', 7
138     .const 'Sub' $P1020 = "14_1471303721.47548"
```

```

139     capture_lex $P1020
140     .const 'Sub' $P1017 = "13_1471303721.47548"
141     capture_lex $P1017
142     .const 'Sub' $P1013 = "12_1471303721.47548"
143     capture_lex $P1013
144     if has_param_1007, optparam_21
145     new $P108, "Undef"
146     set param_1007, $P108
147     optparam_21:
148     .lex "!BLOCK", param_1007
149 .annotate 'line', 6
150     if has_param_1008, optparam_22
151     new $P109, "Undef"
152     set param_1008, $P109
153     optparam_22:
154     .lex "i", param_1008
155 .annotate 'line', 7
156     get_hll_global $P110, "puts"
157     unless_null $P110, vivify_23
158     new $P110, "Undef"
159     vivify_23:
160     new $P111, "CardinalString"
161     assign $P111, "Digite o nome:"
162     $P110($P111)
163 .annotate 'line', 8
164     get_hll_global $P112, "gets"
165     unless_null $P112, vivify_24
166     new $P112, "Undef"
167     vivify_24:
168     $P113 = $P112."chomp"()
169     set $P1009, $P113
170     .lex "nome", $P1009
171 .annotate 'line', 9
172     get_hll_global $P114, "puts"
173     unless_null $P114, vivify_25
174     new $P114, "Undef"
175     vivify_25:
176     new $P115, "CardinalString"
177     assign $P115, "H - homem ou M - mulher"
178     $P114($P115)
179 .annotate 'line', 10
180     get_hll_global $P116, "gets"
181     unless_null $P116, vivify_26
182     new $P116, "Undef"
183     vivify_26:
184     $P117 = $P116."chomp"()
185     set $P1010, $P117
186     .lex "sexo", $P1010
187 .annotate 'line', 12
188     find_lex $P119, "sexo"
189     unless_null $P119, vivify_27
190     new $P119, "Undef"
191     vivify_27:
192     new $P120, "CardinalString"
193     assign $P120, "H"
194     $P121 = "infix:=="($P119, $P120)
195     $P122 = "circumfix:( )"($P121)
196     if $P122, if_1011
197 .annotate 'line', 14

```

3.1

```
198     find_lex $P129, "sexo"
199     unless_null $P129, vivify_28
200     new $P129, "Undef"
201   vivify_28:
202     new $P130, "CardinalString"
203     assign $P130, "M"
204     $P131 = "infix:=="($P129, $P130)
205     $P132 = "circumfix:( )"($P131)
206     if $P132, if_1015
207   .annotate 'line', 16
208     .const 'Sub' $P1020 = "14_1471303721.47548"
209     capture_lex $P1020
210     $P142 = $P1020()
211     set $P128, $P142
212   .annotate 'line', 14
213     goto if_1015_end
214   if_1015:
215   .annotate 'line', 15
216     .const 'Sub' $P1017 = "13_1471303721.47548"
217     capture_lex $P1017
218     $P137 = $P1017()
219     set $P128, $P137
220   if_1015_end:
221   .annotate 'line', 12
222     set $P118, $P128
223     goto if_1011_end
224   if_1011:
225   .annotate 'line', 13
226     .const 'Sub' $P1013 = "12_1471303721.47548"
227     capture_lex $P1013
228     $P127 = $P1013()
229     set $P118, $P127
230   if_1011_end:
231   .annotate 'line', 7
232     .return ($P118)
233 .end
234
235
236 .HLL "cardinal"
237
238 .namespace []
239 .sub "_block1019" :anon :subid("14_1471303721.47548") :outer("11
    _1471303721.47548")
240   .param pmc param_1021 :optional :named("!BLOCK")
241   .param int has_param_1021 :opt_flag
242   .annotate 'file', "micro05.rb"
243   .annotate 'line', 16
244     if has_param_1021, optparam_29
245     new $P138, "Undef"
246     set param_1021, $P138
247   optparam_29:
248     .lex "!BLOCK", param_1021
249   .annotate 'line', 17
250     get_hll_global $P139, "puts"
251     unless_null $P139, vivify_30
252     new $P139, "Undef"
253   vivify_30:
254     new $P140, "CardinalString"
255     assign $P140, "Sexo so pode ser H ou M!"
```

```

256     $P141 = $P139($P140)
257 .annotate 'line', 16
258     .return ($P141)
259 .end
260
261
262 .HLL "cardinal"
263
264 .namespace []
265 .sub "_block1016" :anon :subid("13_1471303721.47548") :outer("11
    _1471303721.47548")
266     .param pmc param_1018 :optional :named("!BLOCK")
267     .param int has_param_1018 :opt_flag
268 .annotate 'file', "micro05.rb"
269 .annotate 'line', 15
270     if has_param_1018, optparam_31
271     new $P133, "Undef"
272     set param_1018, $P133
273     optparam_31:
274     .lex "!BLOCK", param_1018
275     find_lex $P134, "m"
276     unless_null $P134, vivify_32
277     new $P134, "Undef"
278     vivify_32:
279     new $P135, "CardinalInteger"
280     assign $P135, 1
281     $P136 = "infix:+"($P134, $P135)
282     store_lex "m", $P136
283     .return ($P136)
284 .end
285
286
287 .HLL "cardinal"
288
289 .namespace []
290 .sub "_block1012" :anon :subid("12_1471303721.47548") :outer("11
    _1471303721.47548")
291     .param pmc param_1014 :optional :named("!BLOCK")
292     .param int has_param_1014 :opt_flag
293 .annotate 'file', "micro05.rb"
294 .annotate 'line', 13
295     if has_param_1014, optparam_33
296     new $P123, "Undef"
297     set param_1014, $P123
298     optparam_33:
299     .lex "!BLOCK", param_1014
300     find_lex $P124, "h"
301     unless_null $P124, vivify_34
302     new $P124, "Undef"
303     vivify_34:
304     new $P125, "CardinalInteger"
305     assign $P125, 1
306     $P126 = "infix:+"($P124, $P125)
307     store_lex "h", $P126
308     .return ($P126)
309 .end
310
311
312 .HLL "cardinal"

```

3.1

```
313
314 .namespace []
315 .sub "_block1023" :anon :subid("15_1471303721.47548") :outer("10
    _1471303721.47548")
316     .param pmc param_1025 :optional :named("!BLOCK")
317     .param int has_param_1025 :opt_flag
318 .annotate 'file', "micro05.rb"
319 .annotate 'line', 21
320     if has_param_1025, optparam_36
321     new $P147, "Undef"
322     set param_1025, $P147
323 optparam_36:
324     .lex "!BLOCK", param_1025
325     find_lex $P148, "h"
326     unless_null $P148, vivify_37
327     new $P148, "Undef"
328 vivify_37:
329     .return ($P148)
330 .end
331
332
333 .HLL "cardinal"
334
335 .namespace []
336 .sub "_block1026" :anon :subid("16_1471303721.47548") :outer("10
    _1471303721.47548")
337     .param pmc param_1028 :optional :named("!BLOCK")
338     .param int has_param_1028 :opt_flag
339 .annotate 'file', "micro05.rb"
340 .annotate 'line', 22
341     if has_param_1028, optparam_39
342     new $P154, "Undef"
343     set param_1028, $P154
344 optparam_39:
345     .lex "!BLOCK", param_1028
346     find_lex $P155, "m"
347     unless_null $P155, vivify_40
348     new $P155, "Undef"
349 vivify_40:
350     .return ($P155)
351 .end
```

Micro 06

Listagem 3.51: Programa micro 06 em Lua

```
1 -- Escreve um número lido por extenso
2
3 --[[ Função: Faça um algoritmo que leia um número de 1 a 5 e o escreva por
    extenso. Caso o usuário digite um número que não esteja nesse
    intervalo, exibir mensagem: número invalido --]]
4
5 print("Digite um número de 1 a 5")
6 numero = io.read("*number")
7 if(numero == 1) then print("Um")
8 elseif (numero == 2) then print("Dois")
9 elseif (numero == 3) then print("Três")
10 elseif (numero == 4) then print("Quatro")
```

```

11 elsif (numero == 5) then print("Cinco")
12 else print("Número Inválido!!!")
13 end

```

Listagem 3.52: Programa micro 06 em Ruby

```

1 # Escreve um numero lido por extenso
2
3 puts("Digite um numero 1 a 5:")
4 numero = gets.chomp.to_i
5
6 case numero
7 when 1
8   puts("Um")
9 when 2
10  puts("Dois")
11 when 3
12  puts("Tres")
13 when 4
14  puts("Quatro")
15 when 5
16  puts("Cinco")
17 else
18   puts("Numero invalido!!!")
19 end

```

Compilação do Código Ruby:

```

\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/micro06.pir --target=pir micro06.rb

```

Observação: O código primeiramente foi implementado utilizando case/when, que é uma funcionalidade similar ao switch/case do C, porém o compilador apresentou erro ao interpretar essa funcionalidade da linguagem Ruby.

Listagem 3.53: Programa micro 06 em PIR

```

1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471304120.2282")
6   .param pmc param_1002 :optional :named("!BLOCK")
7   .param int has_param_1002 :opt_flag
8 .annotate 'file', "micro06.rb"
9 .annotate 'line', 0
10  .const 'Sub' $P1028 = "17_1471304120.2282"
11  capture_lex $P1028
12  .const 'Sub' $P1025 = "16_1471304120.2282"
13  capture_lex $P1025
14  .const 'Sub' $P1022 = "15_1471304120.2282"
15  capture_lex $P1022
16  .const 'Sub' $P1018 = "14_1471304120.2282"
17  capture_lex $P1018
18  .const 'Sub' $P1014 = "13_1471304120.2282"
19  capture_lex $P1014
20  .const 'Sub' $P1010 = "12_1471304120.2282"
21  capture_lex $P1010

```

3.1

```
22     .const 'Sub' $P1006 = "11_1471304120.2282"
23     capture_lex $P1006
24 .annotate 'line', 1
25     if has_param_1002, optparam_19
26     new $P100, "Undef"
27     set param_1002, $P100
28 optparam_19:
29     .lex "!BLOCK", param_1002
30 .annotate 'line', 3
31     get_hll_global $P101, "puts"
32     unless_null $P101, vivify_20
33     new $P101, "Undef"
34 vivify_20:
35     new $P102, "CardinalString"
36     assign $P102, "Digite um numero 1 a 5:"
37     $P101($P102)
38 .annotate 'line', 4
39     get_hll_global $P103, "gets"
40     unless_null $P103, vivify_21
41     new $P103, "Undef"
42 vivify_21:
43     $P104 = $P103."chomp"()
44     $P105 = $P104."to_i"()
45     set $P1003, $P105
46     .lex "numero", $P1003
47 .annotate 'line', 6
48     find_lex $P107, "numero"
49     unless_null $P107, vivify_22
50     new $P107, "Undef"
51 vivify_22:
52     new $P108, "CardinalInteger"
53     assign $P108, 1
54     $P109 = "infix:=="($P107, $P108)
55     if $P109, if_1004
56 .annotate 'line', 8
57     find_lex $P116, "numero"
58     unless_null $P116, vivify_23
59     new $P116, "Undef"
60 vivify_23:
61     new $P117, "CardinalInteger"
62     assign $P117, 2
63     $P118 = "infix:=="($P116, $P117)
64     if $P118, if_1008
65 .annotate 'line', 10
66     find_lex $P125, "numero"
67     unless_null $P125, vivify_24
68     new $P125, "Undef"
69 vivify_24:
70     new $P126, "CardinalInteger"
71     assign $P126, 3
72     $P127 = "infix:=="($P125, $P126)
73     if $P127, if_1012
74 .annotate 'line', 12
75     find_lex $P134, "numero"
76     unless_null $P134, vivify_25
77     new $P134, "Undef"
78 vivify_25:
79     new $P135, "CardinalInteger"
80     assign $P135, 4
```

```

81     $P136 = "infix:=="($P134, $P135)
82     if $P136, if_1016
83 .annotate 'line', 14
84     find_lex $P143, "numero"
85     unless_null $P143, vivify_26
86     new $P143, "Undef"
87     vivify_26:
88     new $P144, "CardinalInteger"
89     assign $P144, 5
90     $P145 = "infix:=="($P143, $P144)
91     if $P145, if_1020
92 .annotate 'line', 16
93     .const 'Sub' $P1025 = "16_1471304120.2282"
94     capture_lex $P1025
95     $P155 = $P1025()
96     set $P142, $P155
97 .annotate 'line', 14
98     goto if_1020_end
99     if_1020:
100 .annotate 'line', 15
101     .const 'Sub' $P1022 = "15_1471304120.2282"
102     capture_lex $P1022
103     $P150 = $P1022()
104     set $P142, $P150
105     if_1020_end:
106 .annotate 'line', 6
107     set $P133, $P142
108 .annotate 'line', 12
109     goto if_1016_end
110     if_1016:
111 .annotate 'line', 13
112     .const 'Sub' $P1018 = "14_1471304120.2282"
113     capture_lex $P1018
114     $P141 = $P1018()
115     set $P133, $P141
116     if_1016_end:
117 .annotate 'line', 6
118     set $P124, $P133
119 .annotate 'line', 10
120     goto if_1012_end
121     if_1012:
122 .annotate 'line', 11
123     .const 'Sub' $P1014 = "13_1471304120.2282"
124     capture_lex $P1014
125     $P132 = $P1014()
126     set $P124, $P132
127     if_1012_end:
128 .annotate 'line', 6
129     set $P115, $P124
130 .annotate 'line', 8
131     goto if_1008_end
132     if_1008:
133 .annotate 'line', 9
134     .const 'Sub' $P1010 = "12_1471304120.2282"
135     capture_lex $P1010
136     $P123 = $P1010()
137     set $P115, $P123
138     if_1008_end:
139 .annotate 'line', 6

```


3.1

```
140     set $P106, $P115
141     goto if_1004_end
142   if_1004:
143   .annotate 'line', 7
144     .const 'Sub' $P1006 = "11_1471304120.2282"
145     capture_lex $P1006
146     $P114 = $P1006()
147     set $P106, $P114
148   if_1004_end:
149   .annotate 'line', 1
150     .return ($P106)
151 .end
152
153
154 .HLL "cardinal"
155
156 .namespace []
157 .sub "" :load :init :subid("post18") :outer("10_1471304120.2282")
158 .annotate 'file', "micro06.rb"
159 .annotate 'line', 0
160     .const 'Sub' $P1001 = "10_1471304120.2282"
161     .local pmc block
162     set block, $P1001
163 .end
164
165
166 .HLL "parrot"
167
168 .namespace []
169 .sub "_block1027" :init :load :anon :subid("17_1471304120.2282") :outer("
170     10_1471304120.2282")
171 .annotate 'file', "micro06.rb"
172 .annotate 'line', 0
173 $P0 = compreg "cardinal"
174 unless null $P0 goto have_cardinal
175 load_bytecode "cardinal.pbc"
176 have_cardinal:
177     .return ()
178 .end
179
180 .HLL "cardinal"
181
182 .namespace []
183 .sub "_block1024" :anon :subid("16_1471304120.2282") :outer("10
184     _1471304120.2282")
185     .param pmc param_1026 :optional :named("!BLOCK")
186     .param int has_param_1026 :opt_flag
187 .annotate 'file', "micro06.rb"
188 .annotate 'line', 16
189     if has_param_1026, optparam_27
190     new $P151, "Undef"
191     set param_1026, $P151
192   optparam_27:
193     .lex "!BLOCK", param_1026
194 .annotate 'line', 17
195     get_hll_global $P152, "puts"
196     unless_null $P152, vivify_28
197     new $P152, "Undef"
```

```

197   vivify_28:
198       new $P153, "CardinalString"
199       assign $P153, "Numero invalido!!!"
200       $P154 = $P152($P153)
201   .annotate 'line', 16
202   .return ($P154)
203 .end
204
205
206 .HLL "cardinal"
207
208 .namespace []
209 .sub "_block1021" :anon :subid("15_1471304120.2282") :outer("10
    _1471304120.2282")
210     .param pmc param_1023 :optional :named("!BLOCK")
211     .param int has_param_1023 :opt_flag
212 .annotate 'file', "micro06.rb"
213 .annotate 'line', 15
214     if has_param_1023, optparam_29
215         new $P146, "Undef"
216         set param_1023, $P146
217     optparam_29:
218         .lex "!BLOCK", param_1023
219         get_hll_global $P147, "puts"
220         unless_null $P147, vivify_30
221         new $P147, "Undef"
222     vivify_30:
223         new $P148, "CardinalString"
224         assign $P148, "Cinco"
225         $P149 = $P147($P148)
226         .return ($P149)
227 .end
228
229
230 .HLL "cardinal"
231
232 .namespace []
233 .sub "_block1017" :anon :subid("14_1471304120.2282") :outer("10
    _1471304120.2282")
234     .param pmc param_1019 :optional :named("!BLOCK")
235     .param int has_param_1019 :opt_flag
236 .annotate 'file', "micro06.rb"
237 .annotate 'line', 13
238     if has_param_1019, optparam_31
239         new $P137, "Undef"
240         set param_1019, $P137
241     optparam_31:
242         .lex "!BLOCK", param_1019
243         get_hll_global $P138, "puts"
244         unless_null $P138, vivify_32
245         new $P138, "Undef"
246     vivify_32:
247         new $P139, "CardinalString"
248         assign $P139, "Quatro"
249         $P140 = $P138($P139)
250         .return ($P140)
251 .end
252
253

```

3.1

```
254 .HLL "cardinal"
255
256 .namespace []
257 .sub "_block1013" :anon :subid("13_1471304120.2282") :outer("10
    _1471304120.2282")
258     .param pmc param_1015 :optional :named("!BLOCK")
259     .param int has_param_1015 :opt_flag
260 .annotate 'file', "micro06.rb"
261 .annotate 'line', 11
262     if has_param_1015, optparam_33
263     new $P128, "Undef"
264     set param_1015, $P128
265 optparam_33:
266     .lex "!BLOCK", param_1015
267     get_hll_global $P129, "puts"
268     unless_null $P129, vivify_34
269     new $P129, "Undef"
270 vivify_34:
271     new $P130, "CardinalString"
272     assign $P130, "Tres"
273     $P131 = $P129($P130)
274     .return ($P131)
275 .end
276
277
278 .HLL "cardinal"
279
280 .namespace []
281 .sub "_block1009" :anon :subid("12_1471304120.2282") :outer("10
    _1471304120.2282")
282     .param pmc param_1011 :optional :named("!BLOCK")
283     .param int has_param_1011 :opt_flag
284 .annotate 'file', "micro06.rb"
285 .annotate 'line', 9
286     if has_param_1011, optparam_35
287     new $P119, "Undef"
288     set param_1011, $P119
289 optparam_35:
290     .lex "!BLOCK", param_1011
291     get_hll_global $P120, "puts"
292     unless_null $P120, vivify_36
293     new $P120, "Undef"
294 vivify_36:
295     new $P121, "CardinalString"
296     assign $P121, "Dois"
297     $P122 = $P120($P121)
298     .return ($P122)
299 .end
300
301
302 .HLL "cardinal"
303
304 .namespace []
305 .sub "_block1005" :anon :subid("11_1471304120.2282") :outer("10
    _1471304120.2282")
306     .param pmc param_1007 :optional :named("!BLOCK")
307     .param int has_param_1007 :opt_flag
308 .annotate 'file', "micro06.rb"
309 .annotate 'line', 7
```

```

310     if has_param_1007, optparam_37
311     new $P110, "Undef"
312     set param_1007, $P110
313   optparam_37:
314     .lex "!BLOCK", param_1007
315     get_hll_global $P111, "puts"
316     unless_null $P111, vivify_38
317     new $P111, "Undef"
318   vivify_38:
319     new $P112, "CardinalString"
320     assign $P112, "Um"
321     $P113 = $P111($P112)
322     .return ($P113)
323 .end

```

Micro 07

Listagem 3.54: Programa micro 07 em Lua

```

1 -- Listagem 19: Decide se os números são positivos, zeros ou negativos
2
3 --[[ Função: Faça um algoritmo que receba N números e mostre positivo,
   negativo ou zero para cada número --]]
4
5 programa = 1
6 while (programa == 1)
7 do
8   print("Digite um numero: ")
9   numero = io.read()
10  numero = tonumber(numero)
11
12  if(numero > 0)
13  then print("Positivo")
14  elseif(numero == 0)
15  then print("O número é igual a 0")
16  elseif(numero < 0)
17  then print("Negativo")
18  end
19
20
21  print("Deseja Finalizar? (S/N) ")
22  opc = io.read("*line")
23
24  if(opc == "S")
25  then programa = 0
26  end
27 end

```

Listagem 3.55: Programa micro 07 em Ruby

```

1 # Decide se os numeros sao positivos, zero ou negativos
2
3 programa = 1
4 while programa == 1
5   puts("Digite um numero: ")
6   numero = gets.chomp.to_i
7
8   if numero > 0

```

3.1

```
9     puts("Positivo")
10  elsif numero == 0
11     puts("O numero e igual a 0")
12  elsif numero < 0
13     puts("Negativo")
14  end
15
16  puts("Deseja finalizar? (S/N)")
17  opc = gets.chomp
18
19  if opc == 'S'
20     programa = 0
21  end
22 end
```

Compilação do Código Ruby:

```
\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/micro07.pir --target=pir micro07.rb
```

Listagem 3.56: Programa micro 07 em PIR

```
1
2 .HLL "cardinal"
3
4 .namespace []
5 .include "except_types.pasm"
6 .sub "_block1000" :load :main :anon :subid("10_1471303775.64184")
7     .param pmc param_1002 :optional :named("!BLOCK")
8     .param int has_param_1002 :opt_flag
9 .annotate 'file', "micro07.rb"
10 .annotate 'line', 0
11     .const 'Sub' $P1027 = "16_1471303775.64184"
12     capture_lex $P1027
13     .const 'Sub' $P1005 = "11_1471303775.64184"
14     capture_lex $P1005
15 .annotate 'line', 1
16     if has_param_1002, optparam_18
17         new $P100, "Undef"
18         set param_1002, $P100
19     optparam_18:
20         .lex "!BLOCK", param_1002
21 .annotate 'line', 3
22     new $P101, "CardinalInteger"
23     assign $P101, 1
24     set $P1003, $P101
25     .lex "programa", $P1003
26 .annotate 'line', 4
27     new $P143, 'ExceptionHandler', [.CONTROL_LOOP_NEXT;.CONTROL_LOOP_REDO
28         ;.CONTROL_LOOP_LAST]
29     set_label $P143, loop1025_handler
30     push_eh $P143
31 loop1025_test:
32     find_lex $P102, "programa"
33     unless_null $P102, vivify_19
34     new $P102, "Undef"
35 vivify_19:
36     new $P103, "CardinalInteger"
37     assign $P103, 1
```

```

37     $P104 = "infix:=="($P102, $P103)
38     unless $P104, loop1025_done
39     loop1025_redo:
40     .annotate 'line', 5
41         .const 'Sub' $P1005 = "11_1471303775.64184"
42         capture_lex $P1005
43         $P1005()
44     loop1025_next:
45         goto loop1025_test
46     loop1025_handler:
47         .local pmc exception
48         .get_results (exception)
49         pop_upto_eh exception
50         getattribute $P144, exception, 'type'
51         eq $P144, .CONTROL_LOOP_NEXT, loop1025_next
52         eq $P144, .CONTROL_LOOP_REDO, loop1025_redo
53     loop1025_done:
54         pop_eh
55     .annotate 'line', 1
56         .return ($P104)
57 .end
58
59
60 .HLL "cardinal"
61
62 .namespace []
63 .sub "" :load :init :subid("post17") :outer("10_1471303775.64184")
64 .annotate 'file', "micro07.rb"
65 .annotate 'line', 0
66     .const 'Sub' $P1001 = "10_1471303775.64184"
67     .local pmc block
68     set block, $P1001
69 .end
70
71
72 .HLL "parrot"
73
74 .namespace []
75 .sub "_block1026" :init :load :anon :subid("16_1471303775.64184") :outer("
    10_1471303775.64184")
76 .annotate 'file', "micro07.rb"
77 .annotate 'line', 0
78 $P0 = compreg "cardinal"
79 unless null $P0 goto have_cardinal
80 load_bytecode "cardinal.pbc"
81 have_cardinal:
82     .return ()
83 .end
84
85
86 .HLL "cardinal"
87
88 .namespace []
89 .sub "_block1004" :anon :subid("11_1471303775.64184") :outer("10
    _1471303775.64184")
90     .param pmc param_1006 :optional :named("!BLOCK")
91     .param int has_param_1006 :opt_flag
92 .annotate 'file', "micro07.rb"
93 .annotate 'line', 5

```

3.1

```
94     .const 'Sub' $P1023 = "15_1471303775.64184"
95     capture_lex $P1023
96     .const 'Sub' $P1018 = "14_1471303775.64184"
97     capture_lex $P1018
98     .const 'Sub' $P1014 = "13_1471303775.64184"
99     capture_lex $P1014
100    .const 'Sub' $P1010 = "12_1471303775.64184"
101    capture_lex $P1010
102    if has_param_1006, optparam_20
103    new $P105, "Undef"
104    set param_1006, $P105
105    optparam_20:
106    .lex "!BLOCK", param_1006
107    get_hll_global $P106, "puts"
108    unless_null $P106, vivify_21
109    new $P106, "Undef"
110    vivify_21:
111    new $P107, "CardinalString"
112    assign $P107, "Digite um numero: "
113    $P106($P107)
114    .annotate 'line', 6
115    get_hll_global $P108, "gets"
116    unless_null $P108, vivify_22
117    new $P108, "Undef"
118    vivify_22:
119    $P109 = $P108."chomp"()
120    $P110 = $P109."to_i"()
121    set $P1007, $P110
122    .lex "numero", $P1007
123    .annotate 'line', 8
124    find_lex $P111, "numero"
125    unless_null $P111, vivify_23
126    new $P111, "Undef"
127    vivify_23:
128    new $P112, "CardinalInteger"
129    assign $P112, 0
130    $P113 = "infix:>"($P111, $P112)
131    if $P113, if_1008
132    .annotate 'line', 10
133    find_lex $P118, "numero"
134    unless_null $P118, vivify_24
135    new $P118, "Undef"
136    vivify_24:
137    new $P119, "CardinalInteger"
138    assign $P119, 0
139    $P120 = "infix:=="($P118, $P119)
140    if $P120, if_1012
141    .annotate 'line', 12
142    find_lex $P125, "numero"
143    unless_null $P125, vivify_25
144    new $P125, "Undef"
145    vivify_25:
146    new $P126, "CardinalInteger"
147    assign $P126, 0
148    $P127 = "infix:<"($P125, $P126)
149    unless $P127, if_1016_end
150    .annotate 'line', 13
151    .const 'Sub' $P1018 = "14_1471303775.64184"
152    capture_lex $P1018
```

```

153     $P1018()
154   if_1016_end:
155     .annotate 'line', 8
156     goto if_1012_end
157   if_1012:
158     .annotate 'line', 11
159     .const 'Sub' $P1014 = "13_1471303775.64184"
160     capture_lex $P1014
161     $P1014()
162   if_1012_end:
163     .annotate 'line', 8
164     goto if_1008_end
165   if_1008:
166     .annotate 'line', 9
167     .const 'Sub' $P1010 = "12_1471303775.64184"
168     capture_lex $P1010
169     $P1010()
170   if_1008_end:
171     .annotate 'line', 16
172     get_hll_global $P132, "puts"
173     unless_null $P132, vivify_32
174     new $P132, "Undef"
175   vivify_32:
176     new $P133, "CardinalString"
177     assign $P133, "Deseja finalizar? (S/N)"
178     $P132($P133)
179   .annotate 'line', 17
180     get_hll_global $P134, "gets"
181     unless_null $P134, vivify_33
182     new $P134, "Undef"
183   vivify_33:
184     $P135 = $P134."chomp"()
185     set $P1020, $P135
186     .lex "opc", $P1020
187   .annotate 'line', 19
188     find_lex $P137, "opc"
189     unless_null $P137, vivify_34
190     new $P137, "Undef"
191   vivify_34:
192     new $P138, "CardinalString"
193     assign $P138, "S"
194     $P139 = "infix:=="($P137, $P138)
195     if $P139, if_1021
196     set $P136, $P139
197     goto if_1021_end
198   if_1021:
199   .annotate 'line', 20
200     .const 'Sub' $P1023 = "15_1471303775.64184"
201     capture_lex $P1023
202     $P142 = $P1023()
203     set $P136, $P142
204   if_1021_end:
205   .annotate 'line', 5
206     .return ($P136)
207 .end
208
209
210 .HLL "cardinal"
211

```


3.1

```
212 .namespace []
213 .sub "_block1017" :anon :subid("14_1471303775.64184") :outer("11
    _1471303775.64184")
214     .param pmc param_1019 :optional :named("!BLOCK")
215     .param int has_param_1019 :opt_flag
216 .annotate 'file', "micro07.rb"
217 .annotate 'line', 13
218     if has_param_1019, optparam_26
219     new $P128, "Undef"
220     set param_1019, $P128
221 optparam_26:
222     .lex "!BLOCK", param_1019
223     get_hll_global $P129, "puts"
224     unless_null $P129, vivify_27
225     new $P129, "Undef"
226 vivify_27:
227     new $P130, "CardinalString"
228     assign $P130, "Negativo"
229     $P131 = $P129($P130)
230     .return ($P131)
231 .end
232
233
234 .HLL "cardinal"
235
236 .namespace []
237 .sub "_block1013" :anon :subid("13_1471303775.64184") :outer("11
    _1471303775.64184")
238     .param pmc param_1015 :optional :named("!BLOCK")
239     .param int has_param_1015 :opt_flag
240 .annotate 'file', "micro07.rb"
241 .annotate 'line', 11
242     if has_param_1015, optparam_28
243     new $P121, "Undef"
244     set param_1015, $P121
245 optparam_28:
246     .lex "!BLOCK", param_1015
247     get_hll_global $P122, "puts"
248     unless_null $P122, vivify_29
249     new $P122, "Undef"
250 vivify_29:
251     new $P123, "CardinalString"
252     assign $P123, "O numero e igual a 0"
253     $P124 = $P122($P123)
254     .return ($P124)
255 .end
256
257
258 .HLL "cardinal"
259
260 .namespace []
261 .sub "_block1009" :anon :subid("12_1471303775.64184") :outer("11
    _1471303775.64184")
262     .param pmc param_1011 :optional :named("!BLOCK")
263     .param int has_param_1011 :opt_flag
264 .annotate 'file', "micro07.rb"
265 .annotate 'line', 9
266     if has_param_1011, optparam_30
267     new $P114, "Undef"
```

```

268     set param_1011, $P114
269   optparam_30:
270     .lex "!BLOCK", param_1011
271     get_hll_global $P115, "puts"
272     unless_null $P115, vivify_31
273     new $P115, "Undef"
274   vivify_31:
275     new $P116, "CardinalString"
276     assign $P116, "Positivo"
277     $P117 = $P115($P116)
278     .return ($P117)
279 .end
280
281
282 .HLL "cardinal"
283
284 .namespace []
285 .sub "_block1022" :anon :subid("15_1471303775.64184") :outer("11
    _1471303775.64184")
286     .param pmc param_1024 :optional :named("!BLOCK")
287     .param int has_param_1024 :opt_flag
288 .annotate 'file', "micro07.rb"
289 .annotate 'line', 20
290     if has_param_1024, optparam_35
291     new $P140, "Undef"
292     set param_1024, $P140
293   optparam_35:
294     .lex "!BLOCK", param_1024
295     new $P141, "CardinalInteger"
296     assign $P141, 0
297     store_lex "programa", $P141
298     .return ($P141)
299 .end

```

Micro 08

Listagem 3.57: Programa micro 08 em Lua

```

1 -- Listagem 20: Decide se um numero e maior ou menor que 10
2
3 numero = 1
4 while(numero ~= 0)
5 do
6   print("Escreva um numero: ")
7   numero = tonumber(io.read())
8
9   if(numero > 10)
10  then print("O numero",numero,"e maior que 10")
11  else print("O numero",numero,"e menor que 10")
12  end
13 end

```

Listagem 3.58: Programa micro 08 em Ruby

```

1 # Decide se um numero e maior ou menor que 10
2
3 numero = 1
4 while numero != 0

```

3.1

```
5 puts("Escreva um numero:")
6 numero = gets.chomp.to_i
7
8 if numero > 10
9   puts("O numero #{numero} e maior que 10")
10 else
11   puts("O numero #{numero} e menor que 10")
12 end
13 end
```

Compilação do Código Ruby:

```
\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/micro08.pir --target=pir micro08.rb
```

Listagem 3.59: Programa micro 08 em PIR

```
1
2 .HLL "cardinal"
3
4 .namespace []
5 .include "except_types.pasm"
6 .sub "_block1000" :load :main :anon :subid("10_1471303784.49866")
7   .param pmc param_1002 :optional :named("!BLOCK")
8   .param int has_param_1002 :opt_flag
9   .annotate 'file', "micro08.rb"
10  .annotate 'line', 0
11    .const 'Sub' $P1022 = "16_1471303784.49866"
12    capture_lex $P1022
13    .const 'Sub' $P1005 = "11_1471303784.49866"
14    capture_lex $P1005
15  .annotate 'line', 1
16    if has_param_1002, optparam_18
17      new $P100, "Undef"
18      set param_1002, $P100
19    optparam_18:
20      .lex "!BLOCK", param_1002
21  .annotate 'line', 3
22    new $P101, "CardinalInteger"
23    assign $P101, 1
24    set $P1003, $P101
25    .lex "numero", $P1003
26  .annotate 'line', 4
27    new $P135, 'ExceptionHandler', [.CONTROL_LOOP_NEXT;.CONTROL_LOOP_REDO
      ;.CONTROL_LOOP_LAST]
28    set_label $P135, loop1020_handler
29    push_eh $P135
30  loop1020_test:
31    find_lex $P102, "numero"
32    unless_null $P102, vivify_19
33    new $P102, "Undef"
34  vivify_19:
35    new $P103, "CardinalInteger"
36    assign $P103, 0
37    $P104 = "infix:!="($P102, $P103)
38    unless $P104, loop1020_done
39  loop1020_redo:
40  .annotate 'line', 5
41    .const 'Sub' $P1005 = "11_1471303784.49866"
```

```

42     capture_lex $P1005
43     $P1005()
44     loop1020_next:
45         goto loop1020_test
46     loop1020_handler:
47         .local pmc exception
48         .get_results (exception)
49         pop_upto_eh exception
50         getattribute $P136, exception, 'type'
51         eq $P136, .CONTROL_LOOP_NEXT, loop1020_next
52         eq $P136, .CONTROL_LOOP_REDO, loop1020_redo
53     loop1020_done:
54         pop_eh
55 .annotate 'line', 1
56     .return ($P104)
57 .end
58
59
60 .HLL "cardinal"
61
62 .namespace []
63 .sub "" :load :init :subid("post17") :outer("10_1471303784.49866")
64 .annotate 'file', "micro08.rb"
65 .annotate 'line', 0
66     .const 'Sub' $P1001 = "10_1471303784.49866"
67     .local pmc block
68     set block, $P1001
69 .end
70
71
72 .HLL "parrot"
73
74 .namespace []
75 .sub "_block1021" :init :load :anon :subid("16_1471303784.49866") :outer("
    10_1471303784.49866")
76 .annotate 'file', "micro08.rb"
77 .annotate 'line', 0
78 $P0 = compreg "cardinal"
79 unless null $P0 goto have_cardinal
80 load_bytecode "cardinal.pbc"
81 have_cardinal:
82     .return ()
83 .end
84
85
86 .HLL "cardinal"
87
88 .namespace []
89 .sub "_block1004" :anon :subid("11_1471303784.49866") :outer("10
    _1471303784.49866")
90     .param pmc param_1006 :optional :named("!BLOCK")
91     .param int has_param_1006 :opt_flag
92 .annotate 'file', "micro08.rb"
93 .annotate 'line', 5
94     .const 'Sub' $P1015 = "14_1471303784.49866"
95     capture_lex $P1015
96     .const 'Sub' $P1009 = "12_1471303784.49866"
97     capture_lex $P1009
98     if has_param_1006, optparam_20

```

3.1

```
99     new $P105, "Undef"
100     set param_1006, $P105
101   optparam_20:
102     .lex "!BLOCK", param_1006
103     get_hll_global $P106, "puts"
104     unless_null $P106, vivify_21
105     new $P106, "Undef"
106   vivify_21:
107     new $P107, "CardinalString"
108     assign $P107, "Escreva um numero:"
109     $P106($P107)
110   .annotate 'line', 6
111     get_hll_global $P108, "gets"
112     unless_null $P108, vivify_22
113     new $P108, "Undef"
114   vivify_22:
115     $P109 = $P108."chomp"()
116     $P110 = $P109."to_i"()
117     store_lex "numero", $P110
118   .annotate 'line', 8
119     find_lex $P112, "numero"
120     unless_null $P112, vivify_23
121     new $P112, "Undef"
122   vivify_23:
123     new $P113, "CardinalInteger"
124     assign $P113, 10
125     $P114 = "infix:>"($P112, $P113)
126     if $P114, if_1007
127   .annotate 'line', 10
128     .const 'Sub' $P1015 = "14_1471303784.49866"
129     capture_lex $P1015
130     $P134 = $P1015()
131     set $P111, $P134
132   .annotate 'line', 8
133     goto if_1007_end
134   if_1007:
135   .annotate 'line', 9
136     .const 'Sub' $P1009 = "12_1471303784.49866"
137     capture_lex $P1009
138     $P124 = $P1009()
139     set $P111, $P124
140   if_1007_end:
141   .annotate 'line', 5
142     .return ($P111)
143 .end
144
145
146 .HLL "cardinal"
147
148 .namespace []
149 .sub "_block1014" :anon :subid("14_1471303784.49866") :outer("11
    _1471303784.49866")
150   .param pmc param_1016 :optional :named("!BLOCK")
151   .param int has_param_1016 :opt_flag
152   .annotate 'file', "micro08.rb"
153   .annotate 'line', 10
154     .const 'Sub' $P1018 = "15_1471303784.49866"
155     capture_lex $P1018
156     if has_param_1016, optparam_24
```

```

157     new $P125, "Undef"
158     set param_1016, $P125
159     optparam_24:
160         .lex "!BLOCK", param_1016
161     .annotate 'line', 11
162         get_hll_global $P126, "puts"
163         unless_null $P126, vivify_25
164         new $P126, "Undef"
165     vivify_25:
166         new $P127, "CardinalString"
167         assign $P127, "O numero "
168         .const 'Sub' $P1018 = "15_1471303784.49866"
169         capture_lex $P1018
170         $S101 = $P1018()
171         concat $P130, $P127, $S101
172         new $P131, "CardinalString"
173         assign $P131, "e menor que 10"
174         concat $P132, $P130, $P131
175         $P133 = $P126($P132)
176     .annotate 'line', 10
177         .return ($P133)
178 .end
179
180
181 .HLL "cardinal"
182
183 .namespace []
184 .sub "_block1017" :anon :subid("15_1471303784.49866") :outer("14
    _1471303784.49866")
185     .param pmc param_1019 :optional :named("!BLOCK")
186     .param int has_param_1019 :opt_flag
187 .annotate 'file', "micro08.rb"
188 .annotate 'line', 11
189     if has_param_1019, optparam_26
190     new $P128, "Undef"
191     set param_1019, $P128
192     optparam_26:
193         .lex "!BLOCK", param_1019
194         find_lex $P129, "numero"
195         unless_null $P129, vivify_27
196         new $P129, "Undef"
197     vivify_27:
198         .return ($P129)
199 .end
200
201
202 .HLL "cardinal"
203
204 .namespace []
205 .sub "_block1008" :anon :subid("12_1471303784.49866") :outer("11
    _1471303784.49866")
206     .param pmc param_1010 :optional :named("!BLOCK")
207     .param int has_param_1010 :opt_flag
208 .annotate 'file', "micro08.rb"
209 .annotate 'line', 9
210     .const 'Sub' $P1012 = "13_1471303784.49866"
211     capture_lex $P1012
212     if has_param_1010, optparam_28
213     new $P115, "Undef"

```

3.1

```
214     set param_1010, $P115
215 optparam_28:
216     .lex "!BLOCK", param_1010
217     get_hll_global $P116, "puts"
218     unless_null $P116, vivify_29
219     new $P116, "Undef"
220 vivify_29:
221     new $P117, "CardinalString"
222     assign $P117, "O numero "
223     .const 'Sub' $P1012 = "13_1471303784.49866"
224     capture_lex $P1012
225     $$100 = $P1012()
226     concat $P120, $P117, $$100
227     new $P121, "CardinalString"
228     assign $P121, "e maior que 10"
229     concat $P122, $P120, $P121
230     $P123 = $P116($P122)
231     .return ($P123)
232 .end
233
234
235 .HLL "cardinal"
236
237 .namespace []
238 .sub "_block1011" :anon :subid("13_1471303784.49866") :outer("12
    _1471303784.49866")
239     .param pmc param_1013 :optional :named("!BLOCK")
240     .param int has_param_1013 :opt_flag
241     .annotate 'file', "micro08.rb"
242     .annotate 'line', 9
243     if has_param_1013, optparam_30
244     new $P118, "Undef"
245     set param_1013, $P118
246 optparam_30:
247     .lex "!BLOCK", param_1013
248     find_lex $P119, "numero"
249     unless_null $P119, vivify_31
250     new $P119, "Undef"
251 vivify_31:
252     .return ($P119)
253 .end
```

Micro 09

Listagem 3.60: Programa micro 09 em Lua

```
1 -- Listagem 21: Calculo de Precos
2
3 print("Digite o preco: ")
4 preco = tonumber(io.read())
5 print("Digite a venda: ")
6 venda = tonumber(io.read())
7
8 if ((venda < 500) or (preco < 30))
9 then novo_preco = preco + (10/100 * preco)
10 elseif ((venda >= 500 and venda < 1200) or (preco >= 30 and preco < 80))
11 then novo_preco = preco + (15/100 * preco)
12 elseif (venda >= 1200 or preco >= 80)
```

```

13 then novo_preco = preco - (20/100 * preco)
14 end
15
16 print("O novo preco e: ", novo_preco)

```

Listagem 3.61: Programa micro 09 em Ruby

```

1 # Calculo de Precos
2
3 puts("Digite o preco:")
4 preco = gets.chomp.to_f
5 puts("Digite a venda:")
6 venda = gets.chomp.to_f
7
8 if venda < 500 or preco < 30
9   novo_preco = (preco + (10/100.to_f * preco))
10 elsif ((venda >= 500 and venda < 1200) or (preco >= 30 and preco < 80))
11   novo_preco = preco + (15/100.to_f * preco)
12 elsif venda >= 1200 or preco >= 80
13   puts("Oi3")
14   novo_preco = preco - (20/100.to_f * preco)
15 end
16
17 puts("O novo preco e #{novo_preco}")

```

Compilação do Código Ruby:

```

\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/micro09.pir --target=pir micro09.rb

```

ERRO !

Micro 10

Listagem 3.62: Programa micro 10 em Lua

```

1 --Listagem 22: Calcula o fatorial de um numero
2
3 --[[ Função: recebe um número e calcula recursivamente o fatorial desse nú
   mero --]]
4
5 function fatorial(n)
6   if(n <= 0)
7     then return 1
8   else return (n* fatorial(n-1))
9   end
10 end
11
12 print("Digite um numero: ")
13 numero = tonumber(io.read())
14 fat = fatorial(numero)
15
16 print("O fatorial de", numero, "e: ", fat)

```

Listagem 3.63: Programa micro 10 em Ruby

```

1 # Calcula o fatorial de um numero

```


3.1

```
2
3 def fatorial(n)
4   if n <= 0
5     return 1
6   else
7     return (n * fatorial(n-1))
8   end
9 end
10
11 puts("Digite um numero:")
12 numero = gets.chomp.to_i
13 fat = fatorial(numero)
14
15 puts("O fatorial de #{numero} e: #{fat}")
```

Compilação do Código Ruby:

```
\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/micro10.pir --target=pir micro10.rb
```

Listagem 3.64: Programa micro 10 em PIR

```
1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471303809.1351")
6   .param pmc param_1002 :optional :named("!BLOCK")
7   .param int has_param_1002 :opt_flag
8 .annotate 'file', "micro10.rb"
9 .annotate 'line', 0
10   .const 'Sub' $P1022 = "16_1471303809.1351"
11   capture_lex $P1022
12   .const 'Sub' $P1019 = "15_1471303809.1351"
13   capture_lex $P1019
14   .const 'Sub' $P1016 = "14_1471303809.1351"
15   capture_lex $P1016
16   .const 'Sub' $P1003 = "11_1471303809.1351"
17   capture_lex $P1003
18 .annotate 'line', 1
19   if has_param_1002, optparam_18
20     new $P100, "Undef"
21     set param_1002, $P100
22   optparam_18:
23     .lex "!BLOCK", param_1002
24 .annotate 'line', 11
25   get_hll_global $P119, "puts"
26   unless_null $P119, vivify_26
27   new $P119, "Undef"
28   vivify_26:
29     new $P120, "CardinalString"
30     assign $P120, "Digite um numero:"
31     $P119($P120)
32 .annotate 'line', 12
33   get_hll_global $P121, "gets"
34   unless_null $P121, vivify_27
35   new $P121, "Undef"
36   vivify_27:
37     $P122 = $P121."chomp"()
```

```

38     $P123 = $P122."to_i"()
39     set $P1013, $P123
40     .lex "numero", $P1013
41 .annotate 'line', 13
42     get_global $P124, "fatorial"
43     unless_null $P124, vivify_28
44     new $P124, "Undef"
45     vivify_28:
46         find_lex $P125, "numero"
47         unless_null $P125, vivify_29
48         new $P125, "Undef"
49     vivify_29:
50         $P126 = $P124($P125)
51         set $P1014, $P126
52         .lex "fat", $P1014
53 .annotate 'line', 15
54     get_hll_global $P127, "puts"
55     unless_null $P127, vivify_30
56     new $P127, "Undef"
57     vivify_30:
58         new $P128, "CardinalString"
59         assign $P128, "O fatorial de "
60         .const 'Sub' $P1016 = "14_1471303809.1351"
61         capture_lex $P1016
62         $S100 = $P1016()
63         concat $P131, $P128, $S100
64         new $P132, "CardinalString"
65         assign $P132, "e: "
66         concat $P133, $P131, $P132
67         .const 'Sub' $P1019 = "15_1471303809.1351"
68         capture_lex $P1019
69         $S101 = $P1019()
70         concat $P136, $P133, $S101
71         $P137 = $P127($P136)
72 .annotate 'line', 1
73     .return ($P137)
74 .end
75
76
77 .HLL "cardinal"
78
79 .namespace []
80 .sub "" :load :init :subid("post17") :outer("10_1471303809.1351")
81 .annotate 'file', "micro10.rb"
82 .annotate 'line', 0
83     .const 'Sub' $P1001 = "10_1471303809.1351"
84     .local pmc block
85     set block, $P1001
86 .end
87
88
89 .HLL "parrot"
90
91 .namespace []
92 .sub "_block1021" :init :load :anon :subid("16_1471303809.1351") :outer("
    10_1471303809.1351")
93 .annotate 'file', "micro10.rb"
94 .annotate 'line', 0
95 $P0 = compreg "cardinal"

```

3.1

```
96 unless null $P0 goto have_cardinal
97 load_bytecode "cardinal.pbc"
98 have_cardinal:
99     .return ()
100 .end
101
102
103 .HLL "cardinal"
104
105 .namespace []
106 .sub "fatorial" :subid("11_1471303809.1351") :outer("10_1471303809.1351")
107     .param pmc param_1004
108     .param pmc param_1005 :optional :named("!BLOCK")
109     .param int has_param_1005 :opt_flag
110 .annotate 'file', "micro10.rb"
111 .annotate 'line', 1
112     .const 'Sub' $P1011 = "13_1471303809.1351"
113     capture_lex $P1011
114     .const 'Sub' $P1008 = "12_1471303809.1351"
115     capture_lex $P1008
116 .annotate 'line', 3
117     .lex "n", param_1004
118     if has_param_1005, optparam_19
119     new $P101, "Undef"
120     set param_1005, $P101
121     optparam_19:
122     .lex "!BLOCK", param_1005
123 .annotate 'line', 4
124     find_lex $P103, "n"
125     unless_null $P103, vivify_20
126     new $P103, "Undef"
127     vivify_20:
128     new $P104, "CardinalInteger"
129     assign $P104, 0
130     $P105 = "infix:<="($P103, $P104)
131     if $P105, if_1006
132 .annotate 'line', 6
133     .const 'Sub' $P1011 = "13_1471303809.1351"
134     capture_lex $P1011
135     $P118 = $P1011()
136     set $P102, $P118
137 .annotate 'line', 4
138     goto if_1006_end
139     if_1006:
140 .annotate 'line', 5
141     .const 'Sub' $P1008 = "12_1471303809.1351"
142     capture_lex $P1008
143     $P108 = $P1008()
144     set $P102, $P108
145     if_1006_end:
146 .annotate 'line', 1
147     .return ($P102)
148 .end
149
150
151 .HLL "cardinal"
152
153 .namespace []
154 .sub "_block1010" :anon :subid("13_1471303809.1351") :outer("11
```

```

    _1471303809.1351")
155     .param pmc param_1012 :optional :named("!BLOCK")
156     .param int has_param_1012 :opt_flag
157 .annotate 'file', "micro10.rb"
158 .annotate 'line', 6
159     if has_param_1012, optparam_21
160     new $P109, "Undef"
161     set param_1012, $P109
162     optparam_21:
163     .lex "!BLOCK", param_1012
164 .annotate 'line', 7
165     find_lex $P110, "n"
166     unless_null $P110, vivify_22
167     new $P110, "Undef"
168     vivify_22:
169     get_hll_global $P111, "fatorial"
170     unless_null $P111, vivify_23
171     new $P111, "Undef"
172     vivify_23:
173     find_lex $P112, "n"
174     unless_null $P112, vivify_24
175     new $P112, "Undef"
176     vivify_24:
177     new $P113, "CardinalInteger"
178     assign $P113, 1
179     $P114 = "infix:-"($P112, $P113)
180     $P115 = $P111($P114)
181     $P116 = "infix:*"($P110, $P115)
182     $P117 = "circumfix:( )"($P116)
183     .return($P117)
184 .annotate 'line', 6
185     .return ()
186 .end
187
188
189 .HLL "cardinal"
190
191 .namespace []
192 .sub "_block1007" :anon :subid("12_1471303809.1351") :outer("11
    _1471303809.1351")
193     .param pmc param_1009 :optional :named("!BLOCK")
194     .param int has_param_1009 :opt_flag
195 .annotate 'file', "micro10.rb"
196 .annotate 'line', 5
197     if has_param_1009, optparam_25
198     new $P106, "Undef"
199     set param_1009, $P106
200     optparam_25:
201     .lex "!BLOCK", param_1009
202     new $P107, "CardinalInteger"
203     assign $P107, 1
204     .return($P107)
205     .return ()
206 .end
207
208
209 .HLL "cardinal"
210
211 .namespace []

```

3.1

```
212 .sub "_block1015" :anon :subid("14_1471303809.1351") :outer("10
    _1471303809.1351")
213     .param pmc param_1017 :optional :named("!BLOCK")
214     .param int has_param_1017 :opt_flag
215 .annotate 'file', "micro10.rb"
216 .annotate 'line', 15
217     if has_param_1017, optparam_31
218     new $P129, "Undef"
219     set param_1017, $P129
220 optparam_31:
221     .lex "!BLOCK", param_1017
222     find_lex $P130, "numero"
223     unless_null $P130, vivify_32
224     new $P130, "Undef"
225 vivify_32:
226     .return ($P130)
227 .end
228
229
230 .HLL "cardinal"
231
232 .namespace []
233 .sub "_block1018" :anon :subid("15_1471303809.1351") :outer("10
    _1471303809.1351")
234     .param pmc param_1020 :optional :named("!BLOCK")
235     .param int has_param_1020 :opt_flag
236 .annotate 'file', "micro10.rb"
237 .annotate 'line', 15
238     if has_param_1020, optparam_33
239     new $P134, "Undef"
240     set param_1020, $P134
241 optparam_33:
242     .lex "!BLOCK", param_1020
243     find_lex $P135, "fat"
244     unless_null $P135, vivify_34
245     new $P135, "Undef"
246 vivify_34:
247     .return ($P135)
248 .end
```

Micro 11

Listagem 3.65: Programa micro 11 em Lua

```
1 -- Listagem 23: Decide se um número é positivo, zero ou negativo com o
    auxílio de uma função.
2
3 --[[ Função: recebe um número e verifica se o número é positivo, nulo ou
    negativo com o auxílio de uma função --]]
4
5 function verifica(n)
6     if(n > 0)
7     then res = 1
8     elseif (n < 0)
9     then res = -1
10    else res = 0
11    end
12
```

```

13     return res
14 end
15
16 print("Escreva um numero: ")
17 numero = tonumber(io.read())
18 x = verifica(numero)
19
20 if(x==1)
21 then print("Numero positivo")
22 elseif(x==0)
23 then print("Zero")
24 else print("Numero negativo")
25 end

```

Listagem 3.66: Programa micro 11 em Ruby

```

1 # Decide se um numero e positivo, zero ou negativo com auxilio de uma
   funcao
2
3 def verifica(n)
4   if n > 0
5     res = 1
6   elsif n < 0
7     res = -1
8   else
9     res = 0
10  end
11
12  return res
13 end
14
15 puts("Escreva um numero:")
16 numero = gets.chomp.to_i
17
18 x = verifica(numero)
19
20 if x == 1
21   puts("Numero positivo")
22 elsif x == 0
23   puts("Zero")
24 else
25   puts("Numero Negativo")
26 end

```

Compilação do Código Ruby:

```

\$ parrot /Users/oliveira/cardinal/cardinal/cardinal.pbc -o
../parrot/micro11.pir --target=pir micro11.rb

```

Listagem 3.67: Programa micro 11 em PIR

```

1
2 .HLL "cardinal"
3
4 .namespace []
5 .sub "_block1000" :load :main :anon :subid("10_1471303817.60888")
6   .param pmc param_1002 :optional :named("!BLOCK")
7   .param int has_param_1002 :opt_flag

```

3.1

```
8 .annotate 'file', "microll.rb"
9 .annotate 'line', 0
10   .const 'Sub' $P1034 = "18_1471303817.60888"
11   capture_lex $P1034
12   .const 'Sub' $P1031 = "17_1471303817.60888"
13   capture_lex $P1031
14   .const 'Sub' $P1028 = "16_1471303817.60888"
15   capture_lex $P1028
16   .const 'Sub' $P1024 = "15_1471303817.60888"
17   capture_lex $P1024
18   .const 'Sub' $P1003 = "11_1471303817.60888"
19   capture_lex $P1003
20 .annotate 'line', 1
21   if has_param_1002, optparam_20
22     new $P100, "Undef"
23     set param_1002, $P100
24   optparam_20:
25     .lex "!BLOCK", param_1002
26 .annotate 'line', 15
27   get_hll_global $P115, "puts"
28   unless_null $P115, vivify_28
29   new $P115, "Undef"
30   vivify_28:
31     new $P116, "CardinalString"
32     assign $P116, "Escreva um numero:"
33     $P115($P116)
34 .annotate 'line', 16
35   get_hll_global $P117, "gets"
36   unless_null $P117, vivify_29
37   new $P117, "Undef"
38   vivify_29:
39     $P118 = $P117."chomp"()
40     $P119 = $P118."to_i"()
41     set $P1020, $P119
42     .lex "numero", $P1020
43 .annotate 'line', 18
44   get_global $P120, "verifica"
45   unless_null $P120, vivify_30
46   new $P120, "Undef"
47   vivify_30:
48     find_lex $P121, "numero"
49     unless_null $P121, vivify_31
50     new $P121, "Undef"
51   vivify_31:
52     $P122 = $P120($P121)
53     set $P1021, $P122
54     .lex "x", $P1021
55 .annotate 'line', 20
56   find_lex $P124, "x"
57   unless_null $P124, vivify_32
58   new $P124, "Undef"
59   vivify_32:
60     new $P125, "CardinalInteger"
61     assign $P125, 1
62     $P126 = "infix:=="($P124, $P125)
63     if $P126, if_1022
64 .annotate 'line', 22
65     find_lex $P133, "x"
66     unless_null $P133, vivify_33
```

```

67     new $P133, "Undef"
68   vivify_33:
69     new $P134, "CardinalInteger"
70     assign $P134, 0
71     $P135 = "infix:=="($P133, $P134)
72     if $P135, if_1026
73 .annotate 'line', 24
74     .const 'Sub' $P1031 = "17_1471303817.60888"
75     capture_lex $P1031
76     $P145 = $P1031()
77     set $P132, $P145
78 .annotate 'line', 22
79     goto if_1026_end
80   if_1026:
81 .annotate 'line', 23
82     .const 'Sub' $P1028 = "16_1471303817.60888"
83     capture_lex $P1028
84     $P140 = $P1028()
85     set $P132, $P140
86   if_1026_end:
87 .annotate 'line', 20
88     set $P123, $P132
89     goto if_1022_end
90   if_1022:
91 .annotate 'line', 21
92     .const 'Sub' $P1024 = "15_1471303817.60888"
93     capture_lex $P1024
94     $P131 = $P1024()
95     set $P123, $P131
96   if_1022_end:
97 .annotate 'line', 1
98     .return ($P123)
99 .end
100
101
102 .HLL "cardinal"
103
104 .namespace []
105 .sub "" :load :init :subid("post19") :outer("10_1471303817.60888")
106 .annotate 'file', "microll.rb"
107 .annotate 'line', 0
108     .const 'Sub' $P1001 = "10_1471303817.60888"
109     .local pmc block
110     set block, $P1001
111 .end
112
113
114 .HLL "parrot"
115
116 .namespace []
117 .sub "_block1033" :init :load :anon :subid("18_1471303817.60888") :outer("
    10_1471303817.60888")
118 .annotate 'file', "microll.rb"
119 .annotate 'line', 0
120 $P0 = compreg "cardinal"
121 unless null $P0 goto have_cardinal
122 load_bytecode "cardinal.pbc"
123 have_cardinal:
124     .return ()

```


3.1

```
125 .end
126
127
128 .HLL "cardinal"
129
130 .namespace []
131 .sub "verifica" :subid("11_1471303817.60888") :outer("10_1471303817.60888
    ")
132     .param pmc param_1004
133     .param pmc param_1005 :optional :named("!BLOCK")
134     .param int has_param_1005 :opt_flag
135     .annotate 'file', "microll.rb"
136     .annotate 'line', 1
137     .const 'Sub' $P1017 = "14_1471303817.60888"
138     capture_lex $P1017
139     .const 'Sub' $P1013 = "13_1471303817.60888"
140     capture_lex $P1013
141     .const 'Sub' $P1008 = "12_1471303817.60888"
142     capture_lex $P1008
143     .annotate 'line', 3
144     .lex "n", param_1004
145     if has_param_1005, optparam_21
146     new $P101, "Undef"
147     set param_1005, $P101
148     optparam_21:
149     .lex "!BLOCK", param_1005
150     .annotate 'line', 4
151     find_lex $P102, "n"
152     unless_null $P102, vivify_22
153     new $P102, "Undef"
154     vivify_22:
155     new $P103, "CardinalInteger"
156     assign $P103, 0
157     $P104 = "infix:>"($P102, $P103)
158     if $P104, if_1006
159     .annotate 'line', 6
160     find_lex $P107, "n"
161     unless_null $P107, vivify_23
162     new $P107, "Undef"
163     vivify_23:
164     new $P108, "CardinalInteger"
165     assign $P108, 0
166     $P109 = "infix:<"($P107, $P108)
167     if $P109, if_1011
168     .annotate 'line', 8
169     .const 'Sub' $P1017 = "14_1471303817.60888"
170     capture_lex $P1017
171     $P1017()
172     goto if_1011_end
173     if_1011:
174     .annotate 'line', 7
175     .const 'Sub' $P1013 = "13_1471303817.60888"
176     capture_lex $P1013
177     $P1013()
178     if_1011_end:
179     .annotate 'line', 4
180     goto if_1006_end
181     if_1006:
182     .annotate 'line', 5
```

```

183     .const 'Sub' $P1008 = "12_1471303817.60888"
184     capture_lex $P1008
185     $P1008()
186   if_1006_end:
187     .annotate 'line', 12
188     get_hll_global $P114, "res"
189     unless_null $P114, vivify_27
190     new $P114, "Undef"
191   vivify_27:
192     .return($P114)
193   .annotate 'line', 1
194   .return ()
195 .end
196
197
198 .HLL "cardinal"
199
200 .namespace []
201 .sub "_block1016" :anon :subid("14_1471303817.60888") :outer("11
    _1471303817.60888")
202   .param pmc param_1018 :optional :named("!BLOCK")
203   .param int has_param_1018 :opt_flag
204   .annotate 'file', "micro11.rb"
205   .annotate 'line', 8
206   if has_param_1018, optparam_24
207     new $P112, "Undef"
208     set param_1018, $P112
209   optparam_24:
210     .lex "!BLOCK", param_1018
211   .annotate 'line', 9
212   new $P113, "CardinalInteger"
213   assign $P113, 0
214   set $P1019, $P113
215   .lex "res", $P1019
216   .annotate 'line', 8
217   .return ($P1019)
218 .end
219
220
221 .HLL "cardinal"
222
223 .namespace []
224 .sub "_block1012" :anon :subid("13_1471303817.60888") :outer("11
    _1471303817.60888")
225   .param pmc param_1014 :optional :named("!BLOCK")
226   .param int has_param_1014 :opt_flag
227   .annotate 'file', "micro11.rb"
228   .annotate 'line', 7
229   if has_param_1014, optparam_25
230     new $P110, "Undef"
231     set param_1014, $P110
232   optparam_25:
233     .lex "!BLOCK", param_1014
234     new $P111, "CardinalInteger"
235     assign $P111, -1
236     set $P1015, $P111
237     .lex "res", $P1015
238     .return ($P1015)
239 .end

```

3.1

```
240
241
242 .HLL "cardinal"
243
244 .namespace []
245 .sub "_block1007" :anon :subid("12_1471303817.60888") :outer("11
    _1471303817.60888")
246     .param pmc param_1009 :optional :named("!BLOCK")
247     .param int has_param_1009 :opt_flag
248 .annotate 'file', "micro11.rb"
249 .annotate 'line', 5
250     if has_param_1009, optparam_26
251     new $P105, "Undef"
252     set param_1009, $P105
253 optparam_26:
254     .lex "!BLOCK", param_1009
255     new $P106, "CardinalInteger"
256     assign $P106, 1
257     set $P1010, $P106
258     .lex "res", $P1010
259     .return ($P1010)
260 .end
261
262
263 .HLL "cardinal"
264
265 .namespace []
266 .sub "_block1030" :anon :subid("17_1471303817.60888") :outer("10
    _1471303817.60888")
267     .param pmc param_1032 :optional :named("!BLOCK")
268     .param int has_param_1032 :opt_flag
269 .annotate 'file', "micro11.rb"
270 .annotate 'line', 24
271     if has_param_1032, optparam_34
272     new $P141, "Undef"
273     set param_1032, $P141
274 optparam_34:
275     .lex "!BLOCK", param_1032
276 .annotate 'line', 25
277     get_hll_global $P142, "puts"
278     unless_null $P142, vivify_35
279     new $P142, "Undef"
280 vivify_35:
281     new $P143, "CardinalString"
282     assign $P143, "Numero Negativo"
283     $P144 = $P142($P143)
284 .annotate 'line', 24
285     .return ($P144)
286 .end
287
288
289 .HLL "cardinal"
290
291 .namespace []
292 .sub "_block1027" :anon :subid("16_1471303817.60888") :outer("10
    _1471303817.60888")
293     .param pmc param_1029 :optional :named("!BLOCK")
294     .param int has_param_1029 :opt_flag
295 .annotate 'file', "micro11.rb"
```

```

296 .annotate 'line', 23
297     if has_param_1029, optparam_36
298     new $P136, "Undef"
299     set param_1029, $P136
300     optparam_36:
301         .lex "!BLOCK", param_1029
302         get_hll_global $P137, "puts"
303         unless_null $P137, vivify_37
304         new $P137, "Undef"
305     vivify_37:
306         new $P138, "CardinalString"
307         assign $P138, "Zero"
308         $P139 = $P137($P138)
309         .return ($P139)
310 .end
311
312
313 .HLL "cardinal"
314
315 .namespace []
316 .sub "_block1023" :anon :subid("15_1471303817.60888") :outer("10
    _1471303817.60888")
317     .param pmc param_1025 :optional :named("!BLOCK")
318     .param int has_param_1025 :opt_flag
319 .annotate 'file', "micro11.rb"
320 .annotate 'line', 21
321     if has_param_1025, optparam_38
322     new $P127, "Undef"
323     set param_1025, $P127
324     optparam_38:
325         .lex "!BLOCK", param_1025
326         get_hll_global $P128, "puts"
327         unless_null $P128, vivify_39
328         new $P128, "Undef"
329     vivify_39:
330         new $P129, "CardinalString"
331         assign $P129, "Numero positivo"
332         $P130 = $P128($P129)
333         .return ($P130)
334 .end

```

Capítulo 4

Referências

- [1] Documentação Lua - <https://www.lua.org/docs.html>
Documentação OCaml - <https://ocaml.org/docs/>
Wikibooks, Parrot Virtual Machine - https://en.wikibooks.org/wiki/Parrot_virtual_machine
Wikibooks, PASM Reference—https : //en.wikibooks.org/wiki/Parrot_virtual_machine/PASM_Reference
Wikibooks, Parrot Intermediate Representation (PIR)—https : //en.wikibooks.org/wiki/Parrot_virtual_machine/PIR
– Cardinal, Github – [https : //github.com/parrot/cardinal/](https://github.com/parrot/cardinal/)