

Construção de um Compilador de códigos Java para Parrot Virtual Machine

Frederico Franco Calhau

`fred_ffc@hotmail.com`

Faculdade de Computação
Universidade Federal de Uberlândia

18 de abril de 2017

Lista de Figuras

1.1	Instalando e testando OCaml	8
1.2	Instalando e testando Parrot	8

Lista de Tabelas

Lista de Listagens

1.1	Output Simples em Parrot Assembly Language	9
1.2	Output Simples em Parrot Intermediate Representation	9
1.3	Programa nano 01 em Java	10
1.4	Programa nano 01 em PASM	10
1.5	Programa nano 02 em Java	10
1.6	Programa nano 02 em PASM	11
1.7	Programa nano 03 em Java	11
1.8	Programa nano 03 em PASM	11
1.9	Programa nano 04 em Java	11
1.10	Programa nano 04 em PASM	11
1.11	Programa nano 05 em Java	12
1.12	Programa nano 05 em PASM	12
1.13	Programa nano 06 em Java	12
1.14	Programa nano 06 em PASM	12
1.15	Programa nano 07 em Java	13
1.16	Programa nano 07 em PASM	13
1.17	Programa nano 08 em Java	13
1.18	Programa nano 08 em PASM	13
1.19	Programa nano 09 em PASM	14
1.20	Programa nano 10 em Java	14
1.21	Programa nano 10 em PASM	15
1.22	Programa nano 11 em Java	15
1.23	Programa nano 11 em PASM	15
1.24	Programa nano 12 em Java	16
1.25	Programa nano 12 em PASM	16
1.26	Programa micro 01 em Java	17
1.27	Programa Micro 01 em PASM	17
1.28	Programa micro 02 em Java	18
1.29	Programa Micro 02 em PASM	18
1.30	Programa micro 03 em Java	19
1.31	Programa Micro 03 em PASM	19
1.32	Programa micro 04 em Java	20
1.33	Programa Micro 04 em PASM	21
1.34	Programa micro 05 em Java	22
1.35	Programa Micro 05 em PASM	22
1.36	Programa micro 06 em Java	23
1.37	Programa Micro 06 em PASM	24
1.38	Programa micro 07 em Java	25
1.39	Programa Micro 07 em PASM	25
1.40	Programa micro 08 em Java	26

1.41 Programa Micro 08 em PASM	26
1.42 Programa micro 09 em Java	27
1.43 Programa Micro 09 em PASM	28
1.44 Programa micro 10 em Java	29
1.45 Programa Micro 10 em PASM	29
1.46 Programa micro 11 em Java	30
1.47 Programa Micro 11 em PASM	30

Sumário

Lista de Figuras	2
Lista de Tabelas	3
1 Introdução	7
1.1 Instalação dos componentes via Homebrew	7
1.1.1 Instalando Ocaml	7
1.1.2 Instalação da Parrot VM	8
1.2 Máquina Virtual Parrot	8
1.3 Parrot Assembly Language (PASM)	10
1.3.1 Códigos Java e PASM	10
1.3.2 Micro Programas	17

Capítulo 1

Introdução

Este relatório possui o propósito de documentar as atividades realizadas ao longo da disciplina de Construção de Compiladores, a qual tem como objetivo - que também pode ser facilmente deduzido pelo seu nome – de construir uma versão simples de um compilador.

Assim, esse trabalho consiste em criar tal compilador que seja capaz de compilar códigos escritos em Mini Java (um subconjunto da linguagem Java) e "transformá-los" na linguagem PASM (Parrot Assembly), a qual poderá ser interpretada pela máquina virtual Parrot. Para construir esse compilador, utilizou-se a linguagem de programação OCaml, e o sistema operacional macOS (10.12.2).

A seguir, encontra-se algumas explicações das tecnologias e termos descritos acima.

1.1 Instalação dos componentes via Homebrew

Homebrew é um gerenciador de pacotes para macOS, 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)"
```

1.1.1 Instalando Ocaml

Novamente através do homebrew, basta digitar:

```
$ brew install ocaml
```

Resultado:

Figura 1.1: *Instalando e testando OCaml*

```

oliveira~ oliveira$ brew install ocaml
=> Downloading https://homebrew.bintray.com/bottles/ocaml-4.03.0.el_capitan.bot
##### 100.0%
=> 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 = ()
#

```

1.1.2 Instalação da Parrot VM

Digitar no Terminal:

```
$ brew install parrot
```

Resultado:

Figura 1.2: *Instalando e testando Parrot*

```

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: 708 files, 28.4M
oliveira~ oliveira$ parrot
parrot -[aEhrtVwy.] -[Ddt][HEXFLAGS] [-O[level] [-[LIX] path] [-R runcore] [-o
FILE] <file> <args>
oliveira~ oliveira$

```

1.2 Máquina Virtual Parrot

Parrot é uma máquina virtual (VM) projetada para atender as necessidades de linguagens tipadas dinamicamente (como por exemplo Perl e Python), e para prover interoperabilidade entre essas linguagens aceitas.

Ela é uma máquina baseada em registradores, e há 4 tipos desses: inteiros (I), números (N), palavras (S), e PMCs (P). A quantidade de registradores necessários é determinada para cada sub-rotina em tempo de compilação. Os registradores serão nomeados da seguinte maneira "XN" onde 'X' seria uma das letras que representa o tipo do registrador, e 'N' um número entre 0 e a quantidade máxima de registradores daquele tipo. Assim, o quinto registrador do tipo inteiro se chamaria: "I4".

Os registradores do tipo PMCs (Polymorphic Container) representam qualquer tipo ou estrutura de dados complexa (classes e objetos), incluindo agregações de tipos (vetores, hash tables, etc). Eles podem implementar seu próprio comportamento para operações aritméticas, lógicas, e que envolvam palavras, oferecendo comportamento específico para cada linguagem. Podem também ser carregados dinamicamente quando forem requisitados, ao invés de serem montados estaticamente junto ao executável Parrot.

Atualmente, a VM Parrot aceita instruções descritas de 4 formas diferentes, as quais serão descritas a seguir em ordem de abstração — da mais abstrata (high-level) para a mais próxima da linguagem da máquina (low-level):

- PIR (Parrot Intermediate Representation) é a forma padrão. Como o próprio nome diz, ela é uma linguagem intermediária que esconde alguns detalhes low-level do usuário, mas que será convertida para PASM;
- PASM (Parrot Assembly) é um assembly customizado para a máquina Parrot. Utilizaremos essa linguagem;
- PAST (Parrot Abstract Syntax Tree) linguagem útil para construção de compiladores porque permite receber como entrada uma árvore sintática abstrata;
- PBC (Parrot Bytecode) é a linguagem de máquina que pode ser executada imediatamente. Todas as outras linguagens serão primeiro convertidas em PBC para assim poderem ser executadas eficientemente pela máquina Parrot.

Ao longo deste relatório, a linguagem PASM será utilizada como linguagem alvo da nossa compilação, uma vez que ela se assemelha mais com as linguagens (assembly) aceitas pelas outras plataformas estudadas por outros alunos. No entanto, infelizmente, os compiladores de Parrot disponíveis atualmente conseguem apenas compilar para a linguagem PIR.

Fazendo alguns testes com PASM e PIR:

Listagem 1.1: Output Simples em Parrot Assembly Language

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

Para executar o código:

```
$ parrot news.txt
```

Listagem 1.2: 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.txt
```

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.txt
```

De acordo com a documentação oficial, o Compilador Intermediário de Parrot é capaz de traduzir códigos PIR para PASM através do comando:

```
$ parrot -o output.txt input.txt
```

Mas, infelizmente, essa execução resultou em um código bytecode (PBC), ao invés do assembly (PASM). Por isso, para analisar os códigos em PASM, será necessário "compilar" manualmente os arquivos fontes.

Apesar da documentação oficial enfatizar que a linguagem intermediária PIR ser mais recomendada e utilizada no desenvolvimento de ferramentas para Parrot, o alvo será a linguagem Assembly PASM.

1.3 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.

Para melhor entender o funcionamento da linguagem PASM, compilaremos programas simples em Mini Java para PASM. No entanto, infelizmente, essa compilação será feita manualmente devido ao problema comentado anteriormente em que os compiladores disponíveis para a plataforma Parrot não geram mais códigos escritos em PASM, apenas em PIR.

1.3.1 Códigos Java e PASM

Nano Programas

Nano 01

Listagem 1.3: Programa nano 01 em Java

```
1 public class nano01
2 {
3     public static void main(String[] args)
4     {
5
6     }
7 }
```

Listagem 1.4: Programa nano 01 em PASM

```
1 # Modulo Minimo
2 end
```

Nano 02

Listagem 1.5: Programa nano 02 em Java

```
1 public class nano02
```

1.3

```
2 {  
3   public static void main(String[] args)  
4   {  
5       int n;  
6   }  
7 }
```

Listagem 1.6: Programa nano 02 em PASM

```
1 # Declarando uma variavel  
2  
3 end
```

Nano 03

Listagem 1.7: Programa nano 03 em Java

```
1 public class nano03  
2 {  
3   public static void main(String[] args)  
4   {  
5       int n;  
6       n=1;  
7   }  
8 }
```

Listagem 1.8: Programa nano 03 em PASM

```
1 # Atribuição de um inteiro a uma variavel  
2  
3 set I1, 1  
4 end
```

Nano 04

Listagem 1.9: Programa nano 04 em Java

```
1 public class nano04  
2 {  
3   public static void main(String[] args)  
4   {  
5       int n;  
6       n=1+2;  
7   }  
8 }
```

Listagem 1.10: Programa nano 04 em PASM

```
1 # Atribuição de uma soma de inteiros a uma variavel  
2 set I1, 1  
3 set I2, 2  
4 add I3, I1, I2  
5 end
```

Nano 05

Listagem 1.11: Programa nano 05 em Java

```

1 public class nano05
2 {
3     public static void main(String[] args)
4     {
5         int n;
6         n=2;
7         System.out.print(n);
8     }
9 }

```

Listagem 1.12: Programa nano 05 em PASM

```

1 # Inclusão do comando de impressão
2 set I1, 2
3 print I1
4 print "\n"
5
6 end

```

Saída:

2

Nano 06

Listagem 1.13: Programa nano 06 em Java

```

1 public class nano06
2 {
3     public static void main(String[] args)
4     {
5         int n;
6         n= 1 -2;
7         System.out.print(n);
8     }
9 }

```

Listagem 1.14: Programa nano 06 em PASM

```

1 # Atribuição de uma subtração de inteiros a uma variável
2 set I1, 1
3 set I2, 2
4 sub I3, I1, I2
5
6 print I3
7 print "\n"
8
9 end

```

Saída:

-1

Nano 07

Listagem 1.15: Programa nano 07 em Java

```

1 public class nano07
2 {
3     public static void main(String[] args)
4     {
5         int n;
6         n= 1 ;
7         if(n == 1){
8             System.out.print(n);
9         }
10    }
11 }

```

Listagem 1.16: Programa nano 07 em PASM

```

1 # Inclusão do comando condicional
2
3 set      I1, 1 # atribuição
4 eq       I1, 1, VERDADEIRO
5 branch   FIM
6
7 VERDADEIRO:
8 print    I1
9 print    "\n"
10
11 FIM:
12 end

```

Saída:

1

Nano 08

Listagem 1.17: Programa nano 08 em Java

```

1 public class nano08
2 {
3     public static void main(String[] args)
4     {
5         int n;
6         n= 1 ;
7         if(n == 1){
8             System.out.print(n);
9         }
10        else{
11            System.out.print(0);
12        }
13    }
14 }

```

Listagem 1.18: Programa nano 08 em PASM

```

1 # Inclusão do comando condicional senão
2
3 set      I1, 1
4 eq       I1, 1, VERDADEIRO

```

```

5 print    "0\n"
6 branch  FIM
7
8 VERDADEIRO:
9 print    I1
10 print   "\n"
11
12 FIM:
13 end

```

Saída:

1

Nano 09

Listagem 1.19: Programa nano 09 em PASM

```

1 # Atribuição de duas operações aritmeticas sobre inteiros a uma variável
2
3 set      I1, 1
4 set      I2, 2
5 div      I3, I1, I2
6 add      I4, I1, I3
7
8 eq       I4, 1, VERDADEIRO
9 print    "0\n"
10 branch  FIM
11
12 VERDADEIRO:
13 print    I4
14 print   "\n"
15
16 FIM:
17 end

```

Saída:

1

Nano 10

Listagem 1.20: Programa nano 10 em Java

```

1 public class nano10
2 {
3     public static void main(String[] args)
4     {
5         int n,m;
6         n=1;
7         m=2;
8         if(n == m){
9             System.out.print(n);
10        }
11        else{
12            System.out.print(0);

```

1.3

```
13     }  
14 }  
15 }
```

Listagem 1.21: Programa nano 10 em PASM

```
1 # Atribuição de duas variáveis inteiras  
2  
3 set     I1, 1  
4 set     I2, 2  
5  
6 eq I1, I2, VERDADEIRO  
7 print   "0\n"  
8 branch  FIM  
9  
10 VERDADEIRO:  
11 print   I1  
12 print   "\n"  
13  
14 FIM:  
15 end
```

Saída:

0

Nano 11

Listagem 1.22: Programa nano 11 em Java

```
1 public class nano11  
2 {  
3     public static void main(String[] args)  
4     {  
5         int n,m,x;  
6         n=1;  
7         m=2;  
8         x=5;  
9         while (x >n)  
10        {  
11            n = n +m;  
12            System.out.print(n);  
13        }  
14    }  
15 }
```

Listagem 1.23: Programa nano 11 em PASM

```
1 # Introdução do comando de repetição enquanto  
2  
3 set     I1, 1 # n  
4 set     I2, 2 # m  
5 set     I3, 5 # x  
6  
7 TESTE:  
8 gt      I3, I1, LOOP # gt = greater then  
9 branch  FIM
```

```

10
11 LOOP:
12 add      I1, I1, I2
13 print    I1
14 print    "\n"
15 branch   TESTE
16
17 FIM:
18 end

```

Saída:

```

3
5

```

Nano 12

Listagem 1.24: Programa nano 12 em Java

```

1 public class nano12
2 {
3     public static void main(String[] args)
4     {
5         int n,m,x;
6         n=1;
7         m=2;
8         x=5;
9         while (x >n)
10        {
11            if(n==m)
12            {
13                System.out.print(n);
14            }
15            else
16            {
17                System.out.print(0);
18            }
19            x = x -1;
20        }
21    }
22 }

```

Listagem 1.25: Programa nano 12 em PASM

```

1 # Comando condicional aninhado com um de repeticao
2
3 set      I1, 1
4 set      I2, 2
5 set      I3, 5
6
7 TESTE_ENQUANTO:
8 gt       I3, I1, LOOP
9 branch   FIM
10
11 LOOP:
12 eq       I1, I2, VERDADEIRO
13 print    "0\n"
14 branch   POS_CONDICIONAL

```


1.3

```
15
16 VERDADEIRO:
17 print    I1
18 print    "\n"
19
20 POS_CONDICIONAL:
21 dec      I3                # decrementa I3 (x)
22 branch   TESTE_ENQUANTO
23
24 FIM:
25 end
```

Saída:

```
0
0
0
0
```

1.3.2 Micro Programas

Micro 01

Listagem 1.26: Programa micro 01 em Java

```
1 import java.util.Scanner;
2
3 public class micro01
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         float c, f;
9         System.out.println("Celsius -> Fahrenheit");
10        System.out.print("Digite a temperatura em Celsius: ");
11        c = s.nextFloat();
12        f = (9 * c + 160) / 5;
13        System.out.println("A nova temperatura é:" + f + "F");
14    }
15 }
```

Listagem 1.27: Programa Micro 01 em PASM

```
1 # Converte graus Celsius para Fahrenheit
2 .loadlib 'io_ops'          # Para fazer IO
3
4 set      S1, "Celsius -> Fahrenheit\n"
5 set      S2, "Digite a Temperatura em Celsius: "
6 set      S3, "A nova temperatura e: "
7 set      S4, " graus F."
8
9 print    S1
10 print    S2
11 read     S10, 5
12 set      I1, S10
13
14 mul      I1, I1, 9
```

```

15 add      I1, I1, 160
16 div      I1, I1, 5
17
18 print     S3
19 print     I1
20 print     S4
21 print     "\n"
22
23 end

```

Celsius -> Fahrenheit
 Digite a Temperatura em Celsius: 20
 A nova temperatura e: 68 graus F.

Micro 02

Listagem 1.28: Programa micro 02 em Java

```

1 import java.util.Scanner;
2
3 public class micro02
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int num1 , num2 ;
9         System.out.print("Digite o primeiro numero: ");
10        num1 = s.nextInt();
11        System.out.print("Digite o segundo numero: ");
12        num2 = s.nextInt();
13        if(num1 > num2)
14            System.out.print("O primeiro numero "+num1+" e maior que o segundo "
15                               +num2);
16        else
17            System.out.print("O segundo numero "+num2+" e maior que o primeiro "
18                               +num1);
19    }
20 }

```

Listagem 1.29: Programa Micro 02 em PASM

```

1 # Ler dois inteiros e decidir qual e maior
2 .loadlib 'io_ops'
3
4 set      S1, "Digite o primeiro numero: "
5 set      S2, "Digite o segundo numero: "
6 set      S3, "o primeiro numero"
7 set      S4, "o segundo numero"
8 set      S5, " e maior que "
9
10 print    S1
11 read     S10, 3
12 set      I1, S10
13 print    S2
14 read     S11, 3
15 set      I2, S11
16

```

1.3

```
17 gt      I1, I2, VERDADEIRO
18 print   S4
19 print   S5
20 print   S3
21 print   "\n"
22 branch  FIM
23
24 VERDADEIRO:
25 print   S3
26 print   S5
27 print   S4
28 print   "\n"
29
30 FIM:
31 end
```

```
Digite o primeiro numero: 10
Digite o segundo numero: 20
o segundo numero e maior que o primeiro numero
Digite o primeiro numero: 20
Digite o segundo numero: 10
o primeiro numero e maior que o segundo numero
```

Micro 03

Listagem 1.30: Programa micro 03 em Java

```
1 import java.util.Scanner;
2
3 public class micro03
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int numero;
9         System.out.print("Digite um numero: ");
10        numero = s.nextInt();
11
12        if(numero >= 100)
13        {
14            if(numero <=200)
15                System.out.println("O numero esta no intervalo entre 100 e 200");
16            else
17                System.out.println("O numero nao esta no intervalo entre 100 e 200");
18        }
19        else
20            System.out.println("O numero nao esta no intervalo entre 100 e 200");
21    }
22 }
```

Listagem 1.31: Programa Micro 03 em PASM

```
1 # Le um numero e verifica se ele esta entre 100 e 200
2 .loadlib 'io_ops'
3
4 set      S1, "Digite um numero: "
```

```

5 set      S2, "O numero esta no intervalo entre 100 e 200\n"
6 set      S3, "O numero nao esta no intervalo entre 100 e 200\n"
7
8 print     S1
9 read      S10, 3
10 set      I1, S10
11
12 ge        I1, 100, MAIOR_QUE_100
13 branch    NAO_ESTA_NO_INTERVALO
14
15 MAIOR_QUE_100:
16 le        I1, 200, MENOR_QUE_200
17
18 NAO_ESTA_NO_INTERVALO:
19 print     S3
20 branch    FIM
21
22 MENOR_QUE_200:
23 print     S2
24
25 FIM:
26 end

```

```

Digite um numero: 5
O numero nao esta no intervalo entre 100 e 200

Digite um numero: 150
O numero esta no intervalo entre 100 e 200

Digite um numero: 201
O numero nao esta no intervalo entre 100 e 200

```

Micro 04

Listagem 1.32: Programa micro 04 em Java

```

1 import java.util.Scanner;
2
3 public class micro04
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int x=0,num=0,intervalo =0;
9
10        for (x=0;x<5;x++){
11            System.out.print("Digite o numero: ");
12            num = s.nextInt();
13            if( num >=10)
14                if (num <=150)
15                    intervalo = intervalo +1;
16        }
17
18        System.out.println("Ao total, foram digitados "+intervalo+" numeros no
19                           intervalo entre 10 e 150");
20    }
21 }

```

Listagem 1.33: Programa Micro 04 em PASM

```

1 # Le numeros e informa quais estao entre 10 e 150
2 .loadlib 'io_ops'
3
4 set      S1, "Digite um numero: "
5 set      S2, "Ao total foram digitados "
6 set      S3, " numeros no intervalo entre 10 e 150."
7
8 set      I1, 1                                # x
9 set      I2, 0                                # intervalo
10
11 LOOP_TESTE:
12 le      I1, 5, INICIO_LOOP
13 branch  FIM
14
15 INICIO_LOOP:
16 print   S1
17 read    S10, 3
18 set     I10, S10
19
20 ge      I10, 10, MAIOR_QUE_10
21 branch  FIM_LOOP
22
23 MAIOR_QUE_10:
24 le      I10, 150, MENOR_QUE_150
25 branch  FIM_LOOP
26
27 MENOR_QUE_150:
28 inc     I2
29
30 FIM_LOOP:
31 inc     I1
32 branch  LOOP_TESTE
33
34
35 FIM:
36 print   S2
37 print   I2
38 print   S3
39 print   "\n"
40 end

```

```

Digite um numero: 50
Digite um numero: 50
Digite um numero: 50
Digite um numero: 50
Digite um numero: 50
Ao total foram digitados 5 numeros no intervalo entre 10 e 150.

Digite um numero: 02
Digite um numero: 03
Digite um numero: 25
Digite um numero: 60
Digite um numero: 160
Ao total foram digitados 2 numeros no intervalo entre 10 e 150.

```

Micro 05

Listagem 1.34: Programa micro 05 em Java

```

1 import java.util.Scanner;
2
3 public class micro05
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int x=0,h=0,m =0;
9         String nome, sexo;
10
11         for (x=0;x<5;x++) {
12             System.out.print("Digite o nome: ");
13             nome = s.nextLine();
14             System.out.print("H - Homem ou M - Mulher");
15             sexo = s.nextLine();
16
17             switch(sexo) {
18                 case "H":
19                     h = h +1;
20                     break;
21                 case "M":
22                     m = m +1;
23                     break;
24                 default:
25                     System.out.println("Sexo so pode ser H ou M!");
26             }
27         }
28
29         System.out.println("Foram inseridos "+h+" Homens");
30         System.out.println("Foram inseridas "+m+" Mulheres");
31     }
32 }

```

Listagem 1.35: Programa Micro 05 em PASM

```

1 # Le strings e caracteres
2 .loadlib 'io_ops'
3
4 set      S2, "H - Homem ou M - Mulher: "
5 set      S3, "Sexo so pode ser H ou M!\n"
6 set      S4, "Foram inseridos "
7 set      S5, "Foram inseridas "
8 set      S6, " homens"
9 set      S7, " mulheres"
10
11 set      I1, 1                # x
12 set      I2, 0                # homens
13 set      I3, 0                # mulheres
14
15 LOOP_TESTE:
16 le       I1, 5, INICIO_LOOP
17 branch   FIM
18
19 INICIO_LOOP:
20 print    S2
21 read     S11, 2
22
23 eq       S11, "H\n", HOMEM

```

1.3

```
24 eq          S11, "M\n", MULHER
25
26 print       S3
27 branch      FIM_LOOP
28
29 HOMEM:
30 inc         I2
31 branch      FIM_LOOP
32
33 MULHER:
34 inc         I3
35
36 FIM_LOOP:
37 inc         I1
38 branch      LOOP_TESTE
39
40 FIM:
41 print       S4
42 print       I2
43 print       S6
44 print       "\n"
45
46 print       S5
47 print       I3
48 print       S7
49 print       "\n"
50 end
```

```
H - Homem ou M - Mulher: H
H - Homem ou M - Mulher: M
H - Homem ou M - Mulher: H
H - Homem ou M - Mulher: M
H - Homem ou M - Mulher: M
Foram inseridos 2 homens
Foram inseridas 3 mulheres
```

Micro 06

Listagem 1.36: Programa micro 06 em Java

```
1 import java.util.Scanner;
2
3 public class micro06
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int numero=0;
9         System.out.print("Digite um numero de 1 a 5: ");
10        numero = s.nextInt();
11        switch(numero)
12        {
13            case 1:
14                System.out.println("Um");
15                break;
16            case 2:
17                System.out.println("Dois");
18                break;
```

```

19     case 3:
20         System.out.println("Tres");
21         break;
22     case 4:
23         System.out.println("Quatro");
24         break;
25     case 5:
26         System.out.println("Cinco");
27         break;
28     default:
29         System.out.println("Numero Invalido");
30     }
31 }
32 }
33 }

```

Listagem 1.37: Programa Micro 06 em PASM

```

1 # Escrever um numero por extenso
2 .loadlib 'io_ops'
3
4 print      "Digite um numero de 1 a 5: "
5 read      S1, 2
6 set       I1, S1
7
8 eq         I1, 1, UM
9 eq         I1, 2, DOIS
10 eq        I1, 3, TRES
11 eq        I1, 4, QUATRO
12 eq        I1, 5, CINCO
13
14 print      "Numero invalido!!!"
15 branch    FIM
16
17 CINCO:
18 print      "Cinco"
19 branch    FIM
20
21 QUATRO:
22 print      "Quatro"
23 branch    FIM
24
25 TRES:
26 print      "Tres"
27 branch    FIM
28
29 DOIS:
30 print      "Dois"
31 branch    FIM
32
33 UM:
34 print      "Um"
35
36 FIM:
37 print      "\n"
38 end

```

```

Digite um numero de 1 a 5: 3
Tres

```


Micro 07

Listagem 1.38: Programa micro 07 em Java

```

1 import java.util.Scanner;
2
3 public class micro07
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int numero=0, programa=1;
9         char opc;
10        while( programa ==1){
11            System.out.print("Digite um número: ");
12            numero = s.nextInt();
13
14            if (numero>0)
15                System.out.println("Positivo");
16            else
17            {
18                if (numero==0)
19                    System.out.println("O numero e igual a 0");
20                if (numero <0)
21                    System.out.println("Negativo");
22            }
23            System.out.print("Deseja Finalizar? (S/N) ");
24            opc = s.next().charAt(0);
25            if (opc == 'S')
26                programa = 0;
27        }
28    }
29 }

```

Listagem 1.39: Programa Micro 07 em PASM

```

1 # Decide se os numeros sao positivos, zeros ou negativos
2 .loadlib 'io_ops'
3
4 LOOP:
5 print      "Digite um numero: "
6 read      S1, 3
7 set       I1, S1
8
9 # Testar se e maior que 0
10 gt        I1, 0, POSITIVO
11 eq        I1, 0, ZERO
12 lt        I1, 0, NEGATIVO
13
14 POSITIVO:
15 print     "Positivo!\n"
16 branch    FINALIZAR
17
18 ZERO:
19 print     "Zero!\n"
20 branch    FINALIZAR
21
22 NEGATIVO:
23 print     "Negativo!\n"

```

```

24
25 # Parte de DESEJA FINALIZAR?
26 FINALIZAR:
27 print      "Deseja finalizar? (S/N): "
28 read      S10, 2
29 eq        S10, "S\n", FIM
30 branch    LOOP
31
32 FIM:
33 end

```

```

Digite um numero: 5
Positivo!
Deseja finalizar? (S/N): N
Digite um numero: -5
Negativo!
Deseja finalizar? (S/N): N
Digite um numero: 0
Zero!
Deseja finalizar? (S/N): S

```

Micro 08

Listagem 1.40: Programa micro 08 em Java

```

1 import java.util.Scanner;
2
3 public class micro08
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int numero = 1;
9         while (numero < 0 || numero > 0){
10             System.out.print("Digite o numero");
11             numero = s.nextInt();
12             if (numero > 10)
13                 System.out.println("O numero "+numero+" e maior que 10");
14             else
15                 System.out.println("O numero "+numero+" e menor que 10");
16         }
17     }
18 }

```

Listagem 1.41: Programa Micro 08 em PASM

```

1 # Decide se um número é maior ou menor que 10
2 .loadlib 'io_ops'
3
4 set      I1, 1                # variavel numero
5
6 TESTE_LOOP:
7 ne      I1, 0, LOOP
8 branch  FIM
9
10 LOOP:
11 print   "Digite um numero: "
12 read    S10, 3

```

1.3

```
13 set      I1, S10
14
15 gt       I1, 10, MAIOR
16 print    "O numero "
17 print    I1
18 print    " e menor que 10.\n"
19 branch   TESTE_LOOP
20
21 MAIOR:
22 print    "O numero "
23 print    I1
24 print    " e maior que 10.\n"
25 branch   TESTE_LOOP
26
27 FIM:
28 end
```

```
Digite um numero: 50
O numero 50 e maior que 10.
Digite um numero: 5
O numero 5 e menor que 10.
Digite um numero: 0
O numero 0 e menor que 10.
```

Micro 09

Listagem 1.42: Programa micro 09 em Java

```
1 import java.util.Scanner;
2
3 public class micro09
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         double preco, venda, novopreco=0 ;
9
10        System.out.print("Digite o preco: ");
11        preco = s.nextDouble();
12        System.out.print("Digite a venda: ");
13        venda = s.nextDouble();
14
15        if (venda < 500.0 || preco <30.0){
16            novopreco = preco + 10.0/100.0 *preco;
17        }
18        else if ((venda >= 500.0 && venda <1200.0) || (preco >= 30.0 && preco
19            <80.0)){
20            novopreco = preco + 15.0/100.0 * preco;
21        }
22        else if (venda >=1200.0 || preco >=80.0){
23            novopreco = preco - 20.0/100.0 * preco;
24        }
25
26        System.out.println("O novo preco e: "+novopreco);
27    }
28 }
```

Listagem 1.43: Programa Micro 09 em PASM

```

1 # Calculo de precos
2 .loadlib 'io_ops'
3
4
5 print          "Digite o preco (max. 2 digitos): "
6 read          S1, 3
7 set           N1, S1
8 print          "Digite a venda (max. 4 digitos): "
9 read          S1, 5
10 set           N2, S1
11
12 lt            N2, 500, AUMENTAR_10_PORCENTO
13 ge            N1, 30, FALSO1
14
15 AUMENTAR_10_PORCENTO:
16 mul           N3, 10, N1
17 div           N3, N3, 100
18 add           N3, N3, N1
19 branch        FIM
20
21 FALSO1:
22 lt            N2, 500, SEGUNDO_TESTE
23 lt            N2, 1200, AUMENTAR_15_PORCENTO
24 SEGUNDO_TESTE:
25 lt            N1, 30, FALSO2
26 ge            N1, 80, FALSO2
27
28 AUMENTAR_15_PORCENTO:
29 mul           N3, N1, 15
30 div           N3, N3, 100
31 add           N3, N3, N1
32 branch        FIM
33
34 FALSO2:
35 ge            N2, 1200, DIMINUIR_20_PORCENTO
36 lt            N1, 80, FIM
37
38 DIMINUIR_20_PORCENTO:
39 mul           N3, 20, N1
40 div           N3, N3, 100
41 sub           N3, N1, N3
42
43 FIM:
44 print          "O novo preco e: "
45 print          N3
46 print          "\n"
47 end

```

```

Digite o preco: 10
Digite a venda: 10
O novo preco e: 11

```

```

Digite o preco: 40
Digite a venda: 600
O novo preco e: 46

```

```

Digite o preco: 90
Digite a venda: 1500

```

O novo preço é: 72

Micro 10

Listagem 1.44: Programa micro 10 em Java

```

1 import java.util.Scanner;
2
3 public class micro10
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int numero=0, fat;
9         System.out.print("Digite um numero: ");
10        numero = s.nextInt();
11        fat = fatorial(numero);
12        System.out.println("O fatorial de "+numero+" é "+fat);
13
14
15    }
16
17    public static int fatorial(int n){
18        if(n <= 0) return 1;
19        else return n* fatorial(n-1);
20    }
21
22
23 }
```

Listagem 1.45: Programa Micro 10 em PASM

```

1 # Calcula o fatorial de um numero
2 .loadlib 'io_ops'
3
4 print      "Digite um numero: "
5 read      S1, 2
6 set       I1, S1
7 set       I10, S1
8
9 branch    FATORIAL
10 RETURN:
11 print     "O fatorial de "
12 print     I1
13 print     " e: "
14 print     I10
15 print     "\n"
16
17 end
18
19
20 FATORIAL:
21 set       I11, I10
22 dec       I11
23
24 TESTE:
25 eq        I11, 0, RETURN
26 mul       I10, I10, I11
```

```

27 dec          I11
28 branch       TESTE

```

```

Digite um numero: 5
O fatorial de 5 e: 120

```

Micro 11

Listagem 1.46: Programa micro 11 em Java

```

1 import java.util.Scanner;
2
3 public class microll
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int numero=0, x;
9         System.out.print("Digite um numero: ");
10        numero = s.nextInt();
11        x = verifica(numero);
12        if(x ==1) System.out.println("Numero Positivo");
13        else if (x==0) System.out.println("Zero");
14        else System.out.println("Numero Negativo");
15
16    }
17
18    public static int verifica(int n){
19        int res;
20        if(n>0) res =1;
21        else if (n<0) res = -1;
22        else res = 0;
23
24
25        return res;
26    }
27
28
29 }

```

Listagem 1.47: Programa Micro 11 em PASM

```

1 # Decide se um numero e positivo, zero ou negativo com auxilio de uma
   subrotina
2 .loadlib 'io_ops'
3
4 print          "Digite um numero: "
5 read          S1, 3
6 set           I1, S1
7
8 set           I2, 0                # variavel que tera o resultado
9 branch       VERIFICA
10 RETORNO:
11
12 eq           I2, 1,  POSITIVO
13 eq           I2, 0,  ZERO
14 print        "Negativo\n"
15 branch      FIM

```

1.3

```
16
17 ZERO:
18 print      "Zero\n"
19 branch     FIM
20
21 POSITIVO:
22 print      "Positivo\n"
23
24 FIM:
25 end
26
27 VERIFICA:
28 gt         I1, 0, MAIOR
29 lt         I1, 0, MENOR
30 branch     FIM_SUB
31
32 MENOR:
33 set        I2, -1
34 branch     FIM_SUB
35
36 MAIOR:
37 set        I2, 1
38
39 FIM_SUB:
40 branch     RETORNO
```

```
Digite um numero: 5
Positivo
```

```
Digite um numero: -5
Negativo
```

```
Digite um numero: 0
Zero
```