



**Universidade Estadual de Santa Cruz – UESC**

## **Relatórios de Implementações de p-code Machine para o Proj1c**

**Docente César Alberto Bravo Pariente**

**Discente Matheus Miranda Brandão**

**Matrícula 201820065**

**Disciplina Compiladores**

**Curso Ciência da Computação**

**Semestre 2022.2**

**Ilhéus – BA  
2022**

# Índice

---

<b>P-Code Machine</b>	<b>3</b>
Comandos válidos:	<b>3</b>
Tabela de operações:	<b>4</b>
<b>Compilando e Executando</b>	<b>5</b>
<b>Exercícios e Testes</b>	<b>6</b>
Fibonacci(5):	<b>6</b>
Fatorial(4):	<b>8</b>
<b>Link para download</b>	<b>10</b>
<b>Referências</b>	<b>11</b>

# P-Code Machine

---

O projeto consiste na implementação em C de um algoritmo feito em Pascal code. A execução do código segue a regra dos comandos da p-code machine e sua tabela de operações.

## **Comandos válidos:**

LIT 0, a : carrega uma constante a.

OPR 0, a : executa uma operação delimitada entre os intervalos [0,13]..

LOD l, a : Carrega uma variável para o nível l

STO l, a : Armazena uma variável no nível l

CAL l, a : Chama um procedimento no nível l;

INT 0, a : Incrementa o registrador t em a;

JMP 0, a : Pula para a instrução a;

JPC 0, a : Pulo condicional para a instrução a (Se '0' pular, senão ignorar).

## Tabela de operações:

Foi considerada a seguinte codificação de operações

Código	Símbolo	Semântica
0	Return	Realiza o retorno de uma subrotina
1	Negate	$x = \text{pop}(); \text{push}(-x)$
2	Add	$x = \text{pop}(); y = \text{pop}(); \text{push}(y+x).$
3	Subtract	$x = \text{pop}(); y = \text{pop}(); \text{push}(y-x).$
4	Multiply	$x = \text{pop}(); y = \text{pop}(); \text{push}(y*x).$
5	Divide	$x = \text{pop}(); y = \text{pop}(); \text{push}(y/x).$
6	Odd?	Testa se o valor no topo da pilha é ímpar.
7	Equal?	$x = \text{pop}(); y = \text{pop}(); \text{push}(y==x).$
8	Not equal?	$x = \text{pop}(); y = \text{pop}(); \text{push}(y!=x).$
9	Less then?	$x = \text{pop}(); y = \text{pop}(); \text{push}(y<x).$
10	Bigger or equal then?	$x = \text{pop}(); y = \text{pop}(); \text{push}(y>=x).$
11	Bigger then?	$x = \text{pop}(); y = \text{pop}(); \text{push}(y>x)$
12	Less or equal then?	$x = \text{pop}(); y = \text{pop}(); \text{push}(y<=x)$

# Compilando e Executando

---

Para a execução não é necessário o uso de nenhuma dependência, basta compilá-lo normalmente.

```
$ gcc p-code.c -o p-code
```

Ao executar é necessário digitar o nome do arquivo destino contendo as instruções, caso contrário resultará em erro.

Exemplo:

```
$ ./p-code arquivo.txt
```

# Exercícios e Testes

---

## **Fibonacci(5):**

Abaixo será apresentado o código, em p-code, que encontra-se no diretório './examples/fib\_5' e trechos de seu output.

Input:

INT 0 3	STO 0 1
LIT 0 1	LIT 0 5
STO 0 0	LOD 0 2
LIT 0 1	OPR 0 11
STO 0 1	JPC 0 22
LIT 0 3	LOD 0 2
STO 0 2	LIT 0 1
LOD 0 0	OPR 0 2
LOD 0 1	STO 0 2
OPR 0 2	JMP 0 7
LOD 0 1	LOD 0 1
STO 0 0	OPR 0 0

Output:

Instruction	Level	Argument	StackPtr	ProgCounter	Stack
INT	0	3	0	1	0 0 0
LIT	0	1	1	2	0 0 0 1
STO	0	0	0	3	1 0 0
LIT	0	1	1	4	1 0 0 1
STO	0	1	0	5	1 1 0
LIT	0	3	3	6	1 1 0 3
STO	0	2	3	7	1 1 3
LOD	0	0	1	8	1 1 3 1
LOD	0	1	1	9	1 1 3 1 1
OPR	0	2	2	10	1 1 3 2
LOD	0	1	1	11	1 1 3 2 1
STO	0	0	2	12	1 1 3 2
STO	0	1	3	13	1 2 3
LIT	0	5	5	14	1 2 3 5
LOD	0	2	3	15	1 2 3 5 3
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...
LIT	0	5	5	14	3 5 5 5
LOD	0	2	5	15	3 5 5 5 5
OPR	0	11	0	16	3 5 5 0
JPC	0	22	5	22	3 5 5
LOD	0	1	5	23	3 5 5 5

Output: 5

### **Fatorial(4):**

Abaixo será apresentado o código, em p-code, que encontra-se no diretório './examples/fact\_4' e trechos de seu output.

Input:

INT 0 3	OPR 0 2
LIT 0 1	STO 0 1
STO 0 0	LOD 0 1
LIT 0 2	LOD 0 2
STO 0 1	OPR 0 4
LOD 0 0	STO 0 2
LOD 0 1	LOD 0 1
OPR 0 4	LIT 0 4
STO 0 2	OPR 0 7
LOD 0 0	JPC 0 9
LOD 0 1	OPR 0 0



Output:

Instruction	Level	Argument	StackPtr	ProgCounter	Stack
INT	0	3	0	1	0 0 0
LIT	0	1	1	2	0 0 0 1
STO	0	0	0	3	1 0 0
LIT	0	2	2	4	1 0 0 2
STO	0	1	0	5	1 2 0
LOD	0	0	1	6	1 2 0 1
LOD	0	1	2	7	1 2 0 1 2
OPR	0	4	2	8	1 2 0 2
STO	0	2	2	9	1 2 2
LOD	0	0	1	10	1 2 2 1
LOD	0	1	2	11	1 2 2 1 2
OPR	0	2	3	12	1 2 2 3
STO	0	1	2	13	1 3 2
LOD	0	1	3	14	1 3 2 3
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...
OPR	0	4	24	16	1 4 6 24
STO	0	2	24	17	1 4 24
LOD	0	1	4	18	1 4 24 4
LIT	0	4	4	19	1 4 24 4 4
OPR	0	7	1	20	1 4 24 1
JPC	0	9	24	21	1 4 24

Output: 24

# Link para download

---

Código fonte e exemplos encontram-se para download no seguinte link:

<https://github.com/MatBrands/Compiladores/tree/master/Atividade%202>

# Referências

---

[https://en.wikipedia.org/wiki/P-code\\_machine](https://en.wikipedia.org/wiki/P-code_machine)

<https://homepages.cwi.nl/~steven/pascal/book/10pcode.html>

<https://blackmesatech.com/2011/12/pl0/pl0.xhtml>