Insper

Elementos de Sistemas

Aula – Máquina Virtual

"Programadores são criadores de universos da qual só eles são responsáveis. Universos de complexidade praticamente ilimitada podem ser criados sob a forma de programas de computador".
"Programmers are creators of universes for which they alone are responsible. Universes of virtually unlimited complexity can be created in the form of computer programs."
Joseph Weizenbaum (1923-2008), Cientista da Computação
apud Nisan, N. & Schocken, S. 2005. Elements of Computing Systems

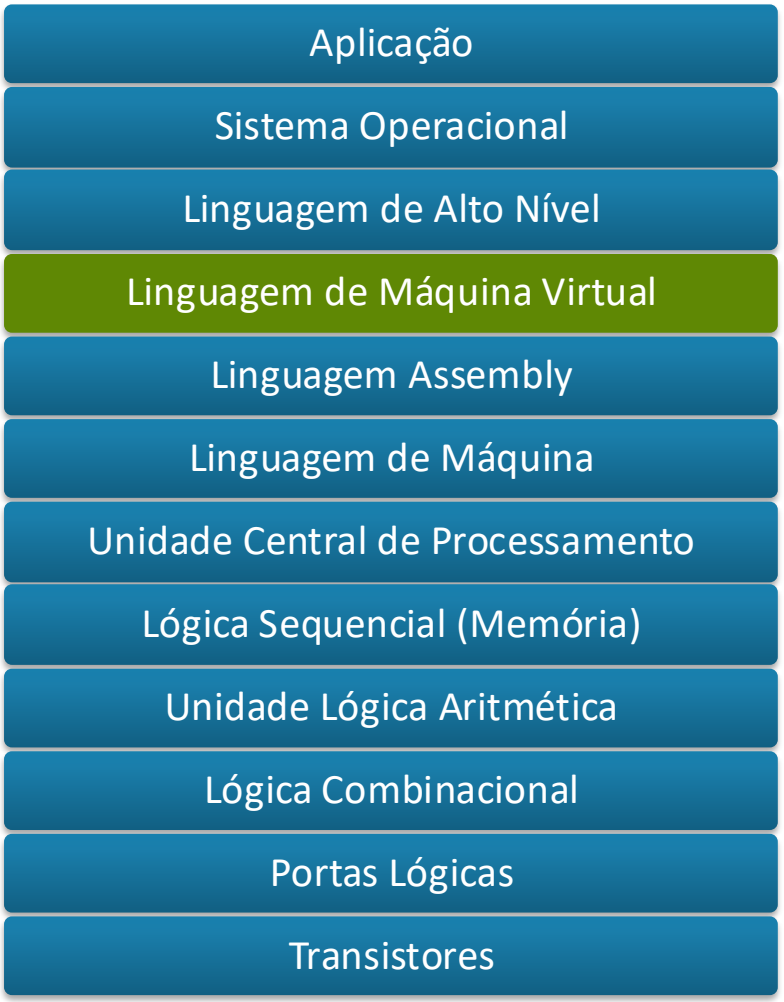
Engenharia - Luciano Soares <lpsoares@insper.edu.br>

Objetivos de Aprendizado da Aula

- Distinguir Máquinas Virtuais;
- Trabalhar com Máquina a Pilha.

Conteúdo(s): Interrupções; Autômatos de Pilha;

Níveis de Abstração

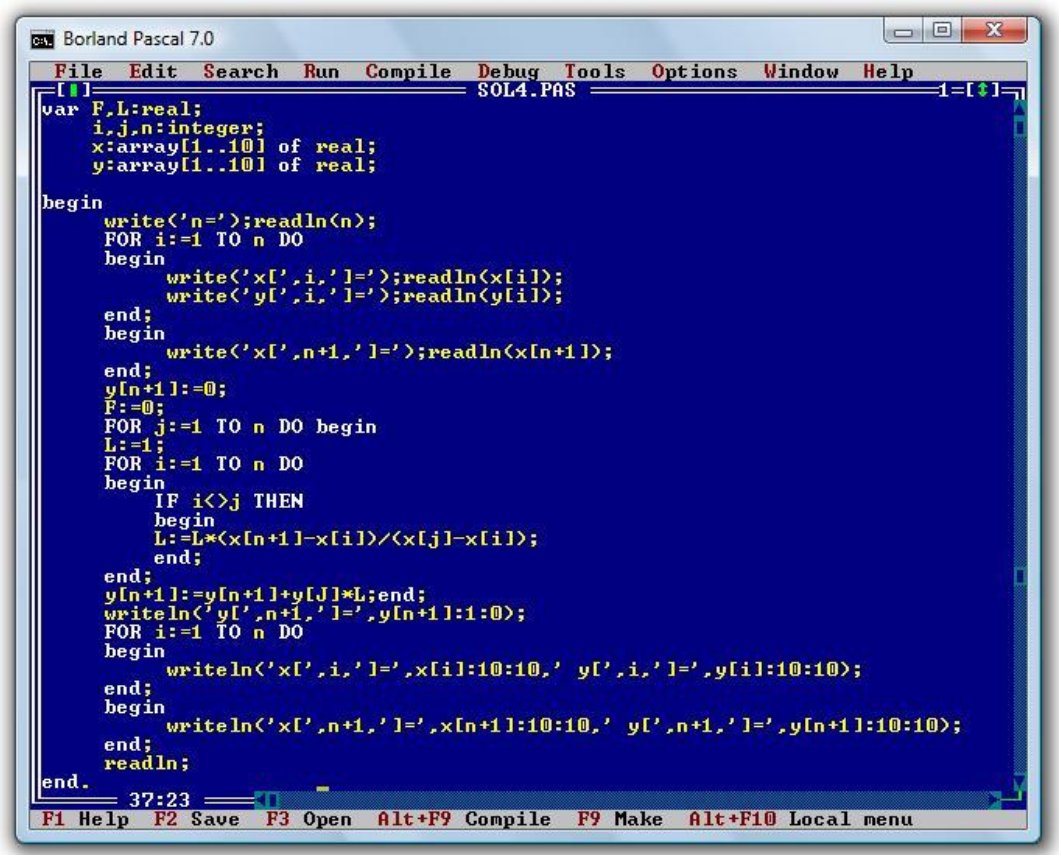


Java	C#
JRE	.NE
java	T
c	csc
JVM	CLR

Insper

P-Code

Linguagem para máquina virtual usada pelo Pascal-P.



p-code

```
L 3
 ENT 1 L 4
 ENT 2 L 5
 LODI 0 5
 LDCI 1
 ADI
 STRI 0 5
 RETP
L 4= 6
L 5= 7
L 6
I 10
ENT 1 L 7
ENT 2 L 8
 MST 0
 CUP 0 L 3
 RETP
L 7= 9
L 8= 5
 Q
 I0
 MST 0
 CUP 0 L 6
 STP
 Q
```


Insper

Java

Linguagem orientada a objetos que roda sobre um Maquina Virtual.

```
01 // First, log in
02 LoginResult loginResult=null;
03 SoapBindingStub sfdc=null;
04 sfdc = (SoapBindingStub) new SforceServiceLocator().getSoap();
05 // login
06 loginResult = sfdc.login("username","password");
07
08 // The set up some security related items
09 // Reset the SOAP endpoint to the returned server URL
10 sfdc._setProperty(SoapBindingStub.ENDPOINT_ADDRESS_PROPERTY,loginResult.getServerUrl());
11 // Create a new session header object
12 // add the session ID returned from the login
13 SessionHeader sh=new SessionHeader();
14 sh.setSessionId(loginResult.getSessionId());
15 sfdc.setHeader(new SforceServiceLocator().getServiceName().getNamespaceURI(),
16   "SessionHeader",sh);
17
18 // now that we're logged in, make some calls - retrieve information about the user
19 GetUserInfoResult userInfo = sfdc.getUserInfo();
20
21 // create a new account object locally
22 Account account = new Account();
23 account.setAccountNumber("002DF99ELK9");
24 account.setName("My New Account");
25 account.setBillingCity("Glasgow")
26
27
28 SObject[] sObjects = new SObject[2];
29 sObjects[0] = account;
30
31 // persist the object
32 SaveResult[] saveResults = sfdc.create(sObjects);
```

bytecode

```
iconst_0      // 03
istore_0      // 3b
iinc 0, 1     // 84 00 01
iload_0       // 1a
iconst_2      // 05
imul          // 68
istore_0      // 3b
goto -7       // a7 ff f9
```

.NET

O framework .NET permite o desenvolvimento em um conjunto de linguagens para rodar um uma máquina virtual.

```
private void btnOpen_Click(object sender, EventArgs e)
{
    string file_name = "C:\\test1.txt";
    string textLine = "";

    System.IO.StreamReader objReader;
    objReader = new System.IO.StreamReader(file_name);

    do
    {
        textLine = textLine + objReader.ReadLine() + "\r\n";

    } while (objReader.Peek() != -1);

    textBox1.Text = textLine;

    objReader.Close();
}
```

CIL

```
0.method private hidebysig
static void  'add'() cil
managed
{
  .maxstack  2
  .locals init ([0] int32 value1,
           [1] int32 value2,
           [2] int32 value3)
  IL_0000:  nop
  IL_0001:  ldc.i4.s   10
  IL_0003:  stloc.0
  IL_0004:  ldc.i4.s   20
  IL_0006:  stloc.1 0
  IL_0007:  ldloc.0
  IL_0008:  ldloc.1
  IL_0009:  add
  IL_000a:  stloc.2
  IL_000b:  ret
}
```

Geração de Código

Alto Nível

- Front end
- Tradutor (Compilador + StdLib)

Código Intermediário

- Back end
- Tradução/Interpretação de Código Intermediário

Linguagem de Máquina

Virtual Machine

Dois tipos de máquinas virtuais:

- System Virtual Machine
- Process Virtual Machine
  (ou Language Virtual Machine)
  (ou Application Virtual Machine)

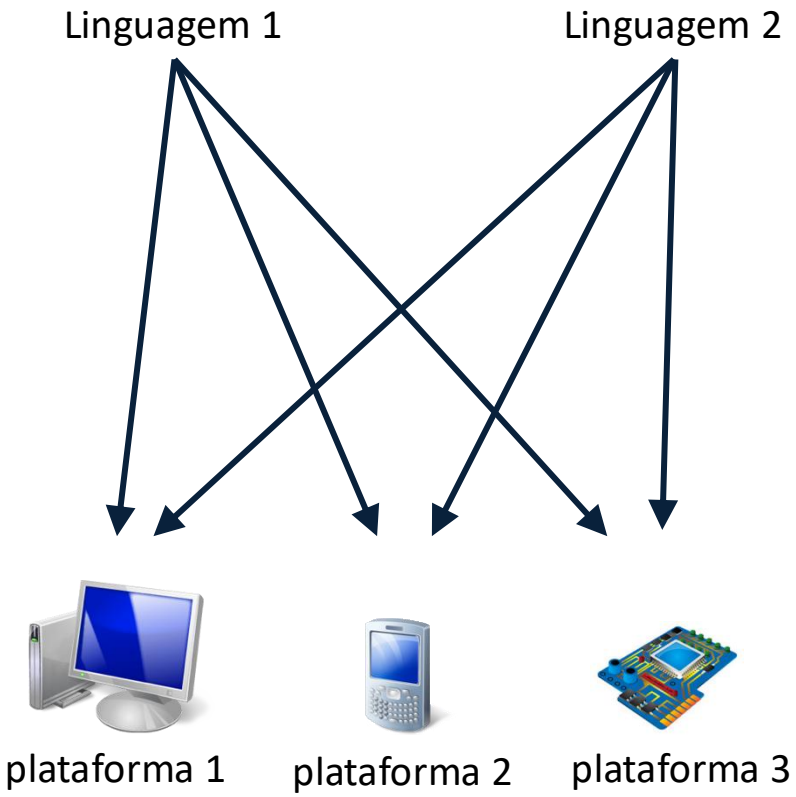
Compiladores

Traduzem programas de alto-nível.

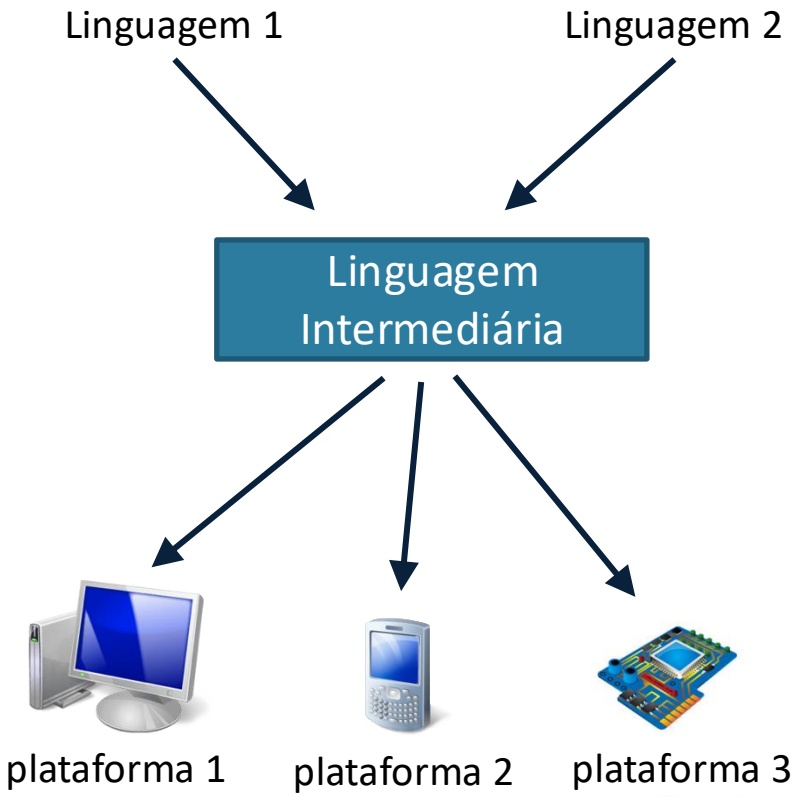
- Diretamente: tem uma forte dependência entre linguagem e arquitetura.
- Por uma linguagem intermediária: Facilita o código ser transportado para outras arquiteturas.

Modelos de Compilação

Compilação Direta (direct compilation)



Compilação de 2 níveis (2-tier compilation)



Insper

Compartilhando Backend

Códigos em diferentes linguagem podem compartilhar um backend. Por exemplo:

JVM:

Java, Scala e Clojure

.NET Framework:

C#, F#, Visual Basic.NET

Operações da Linguagem

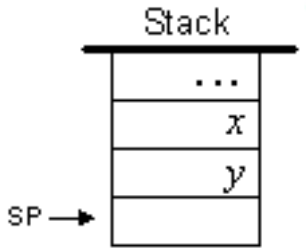
- Aritméticas
- Acesso a memória
- Fluxo de execução
- Chamada de sub-rotinas

Operações de Pilha

Para o funcionamento de pilhas existem duas operações principais:

- PUSH - insere um elemento no topo da pilha
- POP - retira elemento do topo da pilha.

Comandos



Principais comandos em linguagem de pilha:

Comando	Operação	Comentário
add	x + y	complemento de 2
sub	x - y	complemento de 2
neg	- y	complemento de 2
eq	x == y	igualdade
gt	x > y	maior que
lt	x < y	menor que
and	x & y	bit-wise
or	x | y	bit-wise
not	~x	bit-wise

RPN (Reverse Polish Notation)

A Notação Polonesa Inversa, ou notação pós-fixada é usada para calcular operações aritméticas com os operadores após os operandos.

Operação	Notação convencional	Notação Polonesa	Notação Polonesa Inversa
$a + b$	a+b	+ a b	a b +
$\frac{a+b}{c}$	(a+b)/c	/ + a b c	a b + c /
$\frac{a \cdot b - c \cdot d}{e \cdot f}$	((a*b)-(c*d))/(e*f)	/ - * a b * c d * e f	a b * c d * - e f * /

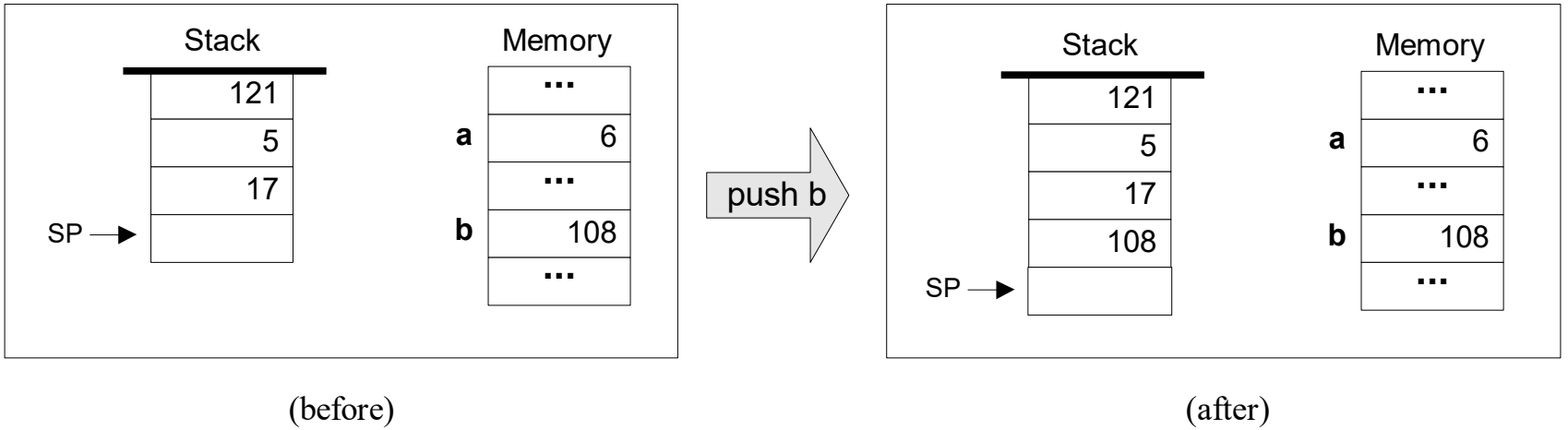
Calculadoras HP

A HP produziu várias calculadores/computadores com a notação polonesa reversa.

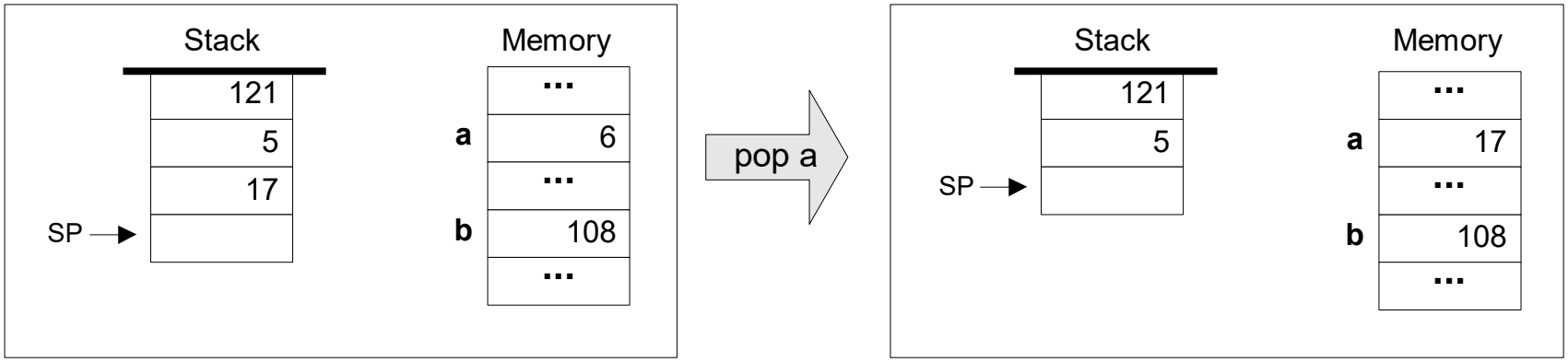
https://epxx.co/ctb/hp12c.html

Insper

Exemplo de Acesso a Memória

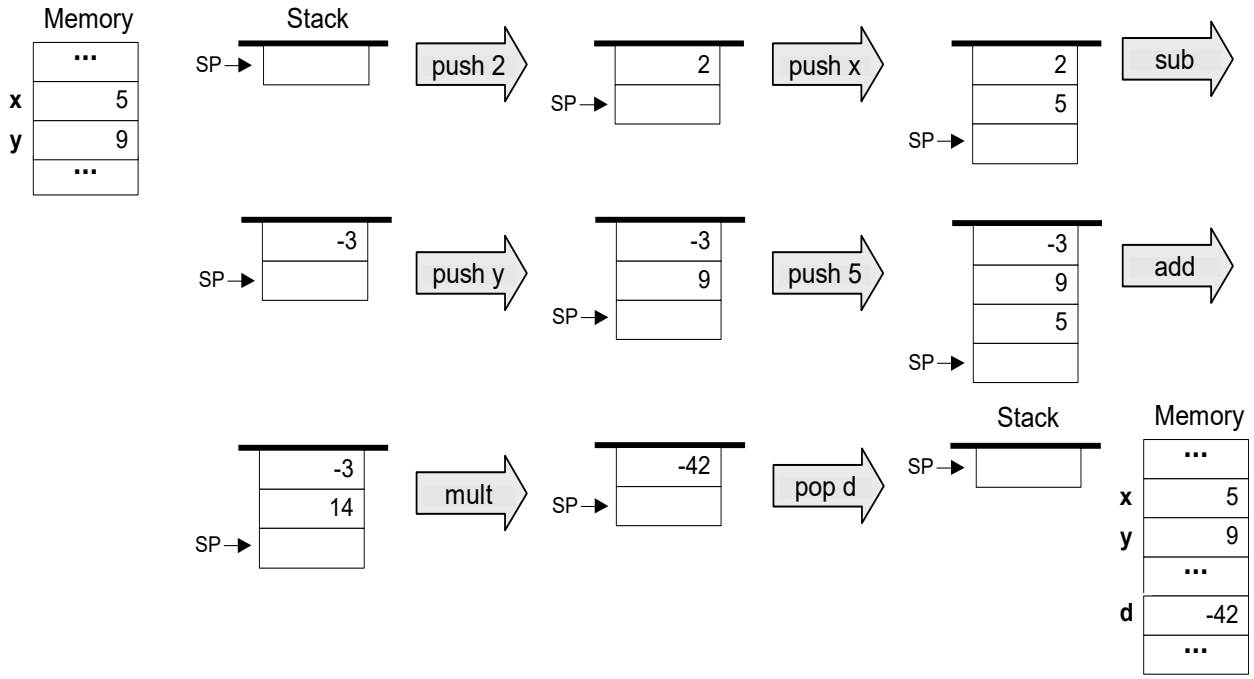


(before) (after)



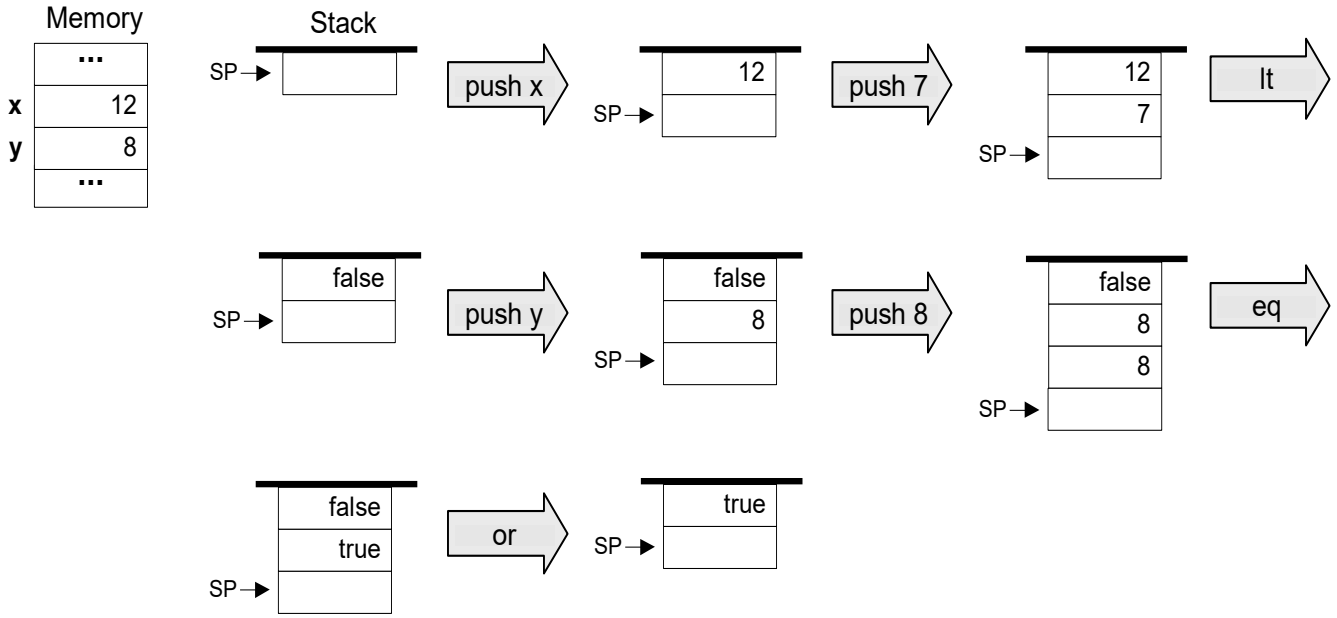
Calculando Expressões Aritméticas

```
// d=(2-x)*(y+5)
push 2
push x
sub
push y
push 5
add
mult
pop d
```



Processando Expressões Booleanas

```
// if (x<7) or (y=8)
push x
push 7
lt
push y
push 8
eq
or
```



Exercício Exemplo

Converta a seguinte equação para a Notação Polonesa Inversa e implemente na máquina de pilha na linguagem VM do Z0:

-3 + (5 + 6) - (4 + 3)

RPN = -3 5 6 + + 4 3 + -

```
push constant 3
neg
push constant 5
push constant 6
add
add
push constant 4
push constant 3
add
sub
```

Insper

Exercício Exemplo

Converta a seguinte expressão booleana para operações de pilha e implemente na máquina de pilha na linguagem VM do Z0:

(10 > 2) OR (5 == 6)

```
10 2 > 5 6 eq or
```

```
push constant 10
push constant 2
gt
push constant 5
push constant 6
eq
or
```

Insper

www.insper.edu.br