

# Introdução à programação

INF1005 -- Programação I -- 2016.1

Prof. Roberto Azevedo

razevedo@inf.puc-rio.br



DEPARTAMENTO  
DE INFORMÁTICA  
PUC-RIO

# roteiro

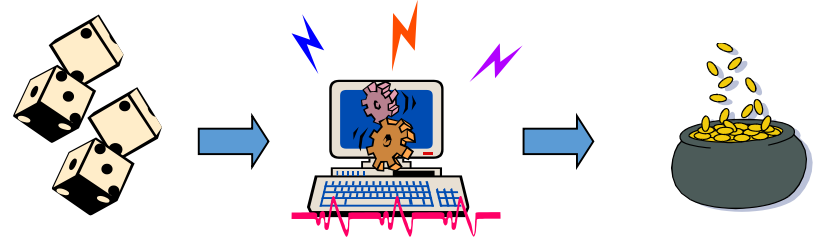
## tópicos

- conceitos básicos
- o que é um programa
- um programa na memória
  - Representação de dados
  - Representação do programa
- decifrando um código

## referência

- Capítulo 1 da apostila

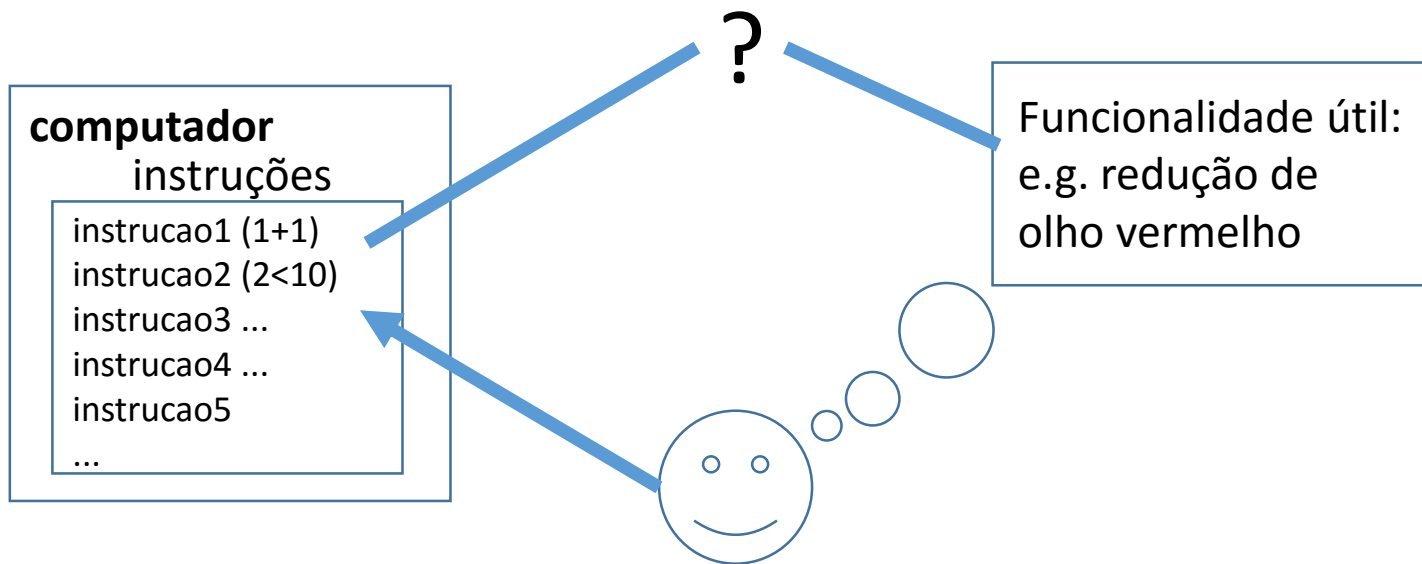
# conceitos básicos



- Um **computador** é uma máquina capaz de manipular informações processando uma sequência de instruções.
- As sequências de instruções definem um **programa**.
  - Programas são escritos para resolver problemas ou realizar tarefas no computador.
- **Programação de computador** é o processo de desenvolver e implementar programas para habilitar o computador a realizar uma determinada tarefa.

# conceitos básicos

## Programação de computadores

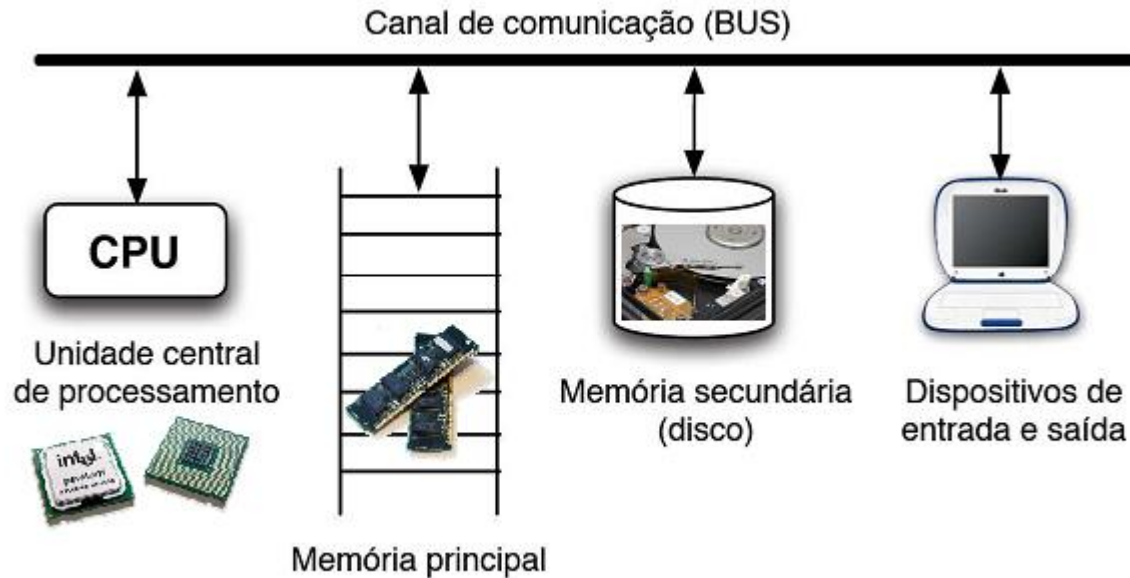


# programação de computadores

- Programadores traduzem soluções ou tarefas em uma determinada linguagem que o computador consegue entender.
- Enquanto escrevemos programas, nós devemos ter em mente que **o computador só faz aquilo que ele é instruído a fazer!**
- Por causa disso, devemos ser bastante cuidadosos e completos com as nossas instruções.

# modelo de um computador

- *hardware* – componentes físicos



- *software* - programas

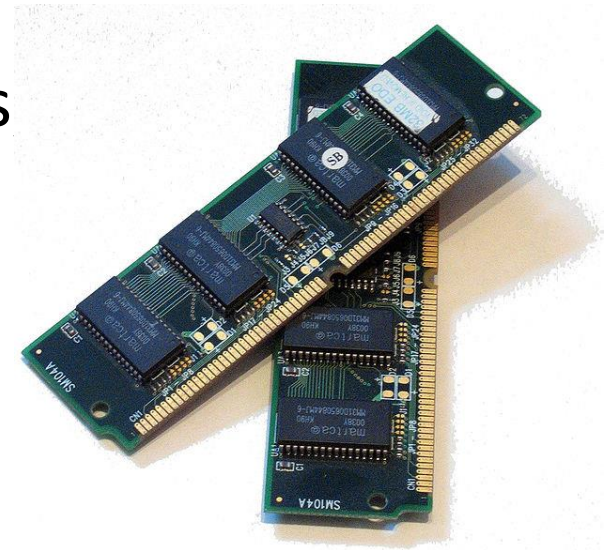
# CPU: Unidade Central de Processamento

- Principal componente de um computador digital.
- **Localiza** e **executa** as instruções de um programa.
- Capaz de executar operações simples com grande rapidez.



# memória principal

- Memória volátil (não-permanente) usada para armazenar **dados** e **programas**.
- Conteúdo modificável pelas instruções dos programas.
- Consiste de células elementares identificada por um endereço.
- Permite acesso rápido e aleatório
  - RAM: *Randomic Acess Memory*





## memória secundária (disco)

- Geralmente representada por meios magnéticos
  - Ex. Disco Rígido (HD)
- Acesso aos dados **bem mais lento** do que a memória principal.
- Tem a vantagem de ser **permanente**.
- Para serem processados pela CPU, dados armazenados no disco devem antes ser transferido para a memória principal.

# dispositivos de entrada/saída

## dispositivos de entrada

- permitem que os usuários forneçam dados para o programa
- Exemplos: Teclado, mouse, *touch screen*, etc.

## dispositivos de saída

- permitem que um programa exiba resultados computados
- Exemplos: Monitores, Impressoras, etc.

# instruções de máquina

- Para que uma máquina seja capaz de realizar várias operações, é preciso que ela seja de algum modo instruída para **identificar** cada uma delas e, depois de identificá-las, saber como **realizá-las**. Essas instruções são denominadas **instruções de máquina**.
- Podem existir diferentes instruções para diferentes modelos/tipos de CPUs
  - **Quais as operações reconhecidas pelos exemplos da última aula (Lightbot, Jogo de Raciocínio, Torres de Hanói etc.)?**

# computador hipotético

- Memória
  - posições com endereços 0 a n
  - registrador (*register*)
    - armazena resultado de operação
- Instruções

Endereço	Valor
<b>register</b>	
0	
1	
2	
3	
...	

Instruções	Descrição
<b>read</b> <i>pos</i>	Lê um número do teclado e grava-o no endereço <i>pos</i>
<b>write</b> <i>pos</i>	Escreve na tela o número que está em <i>pos</i>
<b>storeconst</b> <i>num pos</i>	Grava <i>num</i> em <i>pos</i>
<b>add</b> <i>pos1 pos2</i>	Calcula $pos1 + pos2$ e grava o resultado em <i>pos</i>
<b>sub</b> <i>pos1 pos2</i>	Calcula $pos1 - pos2$ e grava resultado em <i>register</i>
<b>mul</b> <i>pos1 pos2</i>	Calcula $pos1 \times pos2$ e grava resultado em <i>register</i>
<b>div</b> <i>pos1 pos2</i>	Calcula $pos1 / pos2$ e grava o resultado em <i>register</i>
<b>store</b> <i>pos</i>	Grava o número que está em <i>register</i> em <i>pos</i>

# O que é um programa?

- Sequência de instruções
  - Exemplo: programa que escreve na tela o resultado da soma de 2.5 mais um número lido do teclado

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 2
0 2 5 0 0 0 0 1
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 3
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 7
0 0 0 0 0 0 0 2
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 2
```

```
read 0
storeconst 2.5 1
add 0 1
store 2
write 2
```

```
variáveis
  valor1, valor2, valor3
início
  leia valor1
  valor2 = 2.5
  valor3 = valor1 + valor2
  escreva valor3
fim
```

# O que é um programa?

- Sequência de instruções

- Exemplo: programa que escreve na tela o resultado da soma de 2.5 mais um número lido do teclado

```
1.read 0
2.storeconst 2.5 1
3.add 0 1
4.store 2
5.write 2
```

>

# O que é um programa?

- Sequência de instruções

- Exemplo: programa que escreve na tela o resultado da soma de 2.5 mais um número lido do teclado



```
1.read 0  
2.storeconst 2.5 1  
3.add 0 1  
4.store 2  
5.write 2
```

Suponha que o usuário digite o valor 3.

read 0

> 3

# O que é um programa?

- Sequência de instruções

- Exemplo: programa que escreve na tela o resultado da soma de 2.5 mais um número lido do teclado

```
1.read 0  
2.storeconst 2.5 1  
3.add 0 1  
4.store 2  
5.write 2
```

Suponha que o usuário digite o valor 3.

read 0

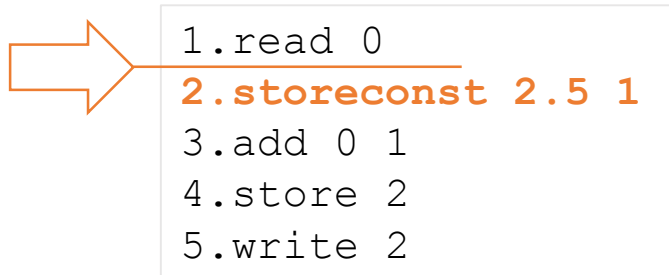
> 3	end.	val.
	reg.	???
	0	3
	1	???
	2	???
	3	???
	...	???



# O que é um programa?

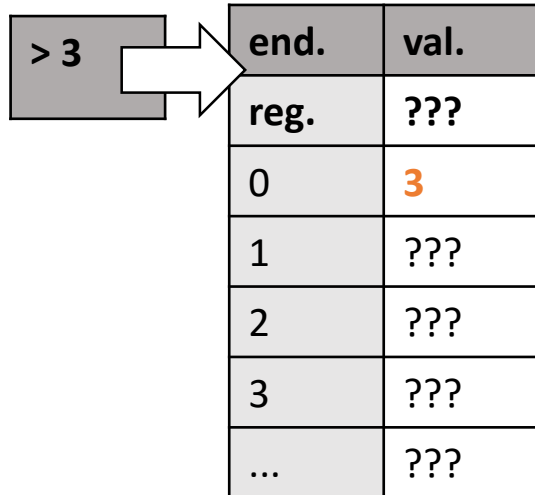
- Sequência de instruções

- Exemplo: programa que escreve na tela o resultado da soma de 2.5 mais um número lido do teclado



```
1.read 0
2.storeconst 2.5 1
3.add 0 1
4.store 2
5.write 2
```

read 0                      storeconst 2.5 1



> 3

end.	val.
reg.	???
0	3
1	???
2	???
3	???
...	???

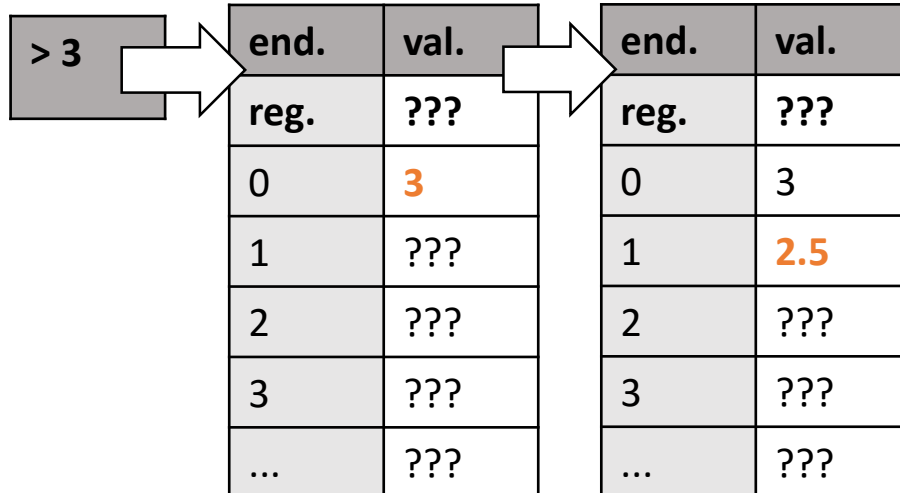
# O que é um programa?

- Sequência de instruções

- Exemplo: programa que escreve na tela o resultado da soma de 2.5 mais um número lido do teclado

```
1.read 0  
2.storeconst 2.5 1  
3.add 0 1  
4.store 2  
5.write 2
```

read 0                      storeconst 2.5 1



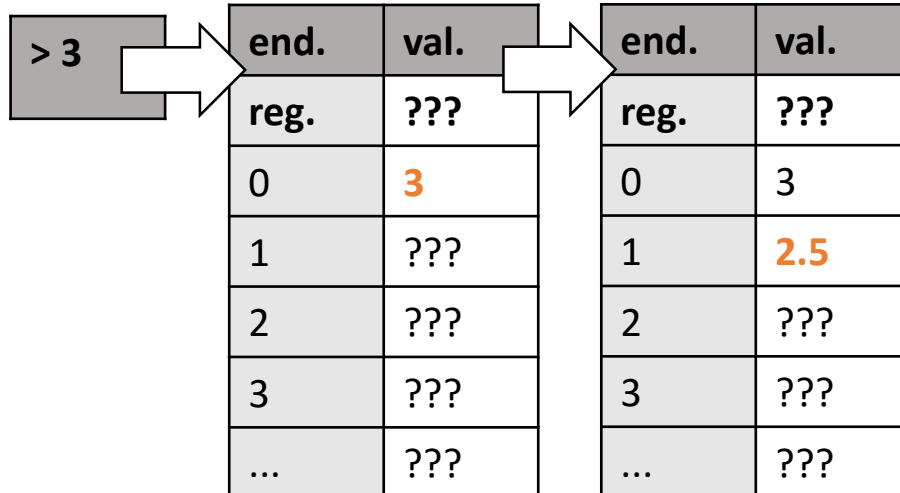
# O que é um programa?

- Sequência de instruções

- Exemplo: programa que escreve na tela o resultado da soma de 2.5 mais um número lido do teclado

```
1.read 0  
2.storeconst 2.5 1  
3.add 0 1  
4.store 2  
5.write 2
```

read 0                      storeconst 2.5 1                      add 0 1



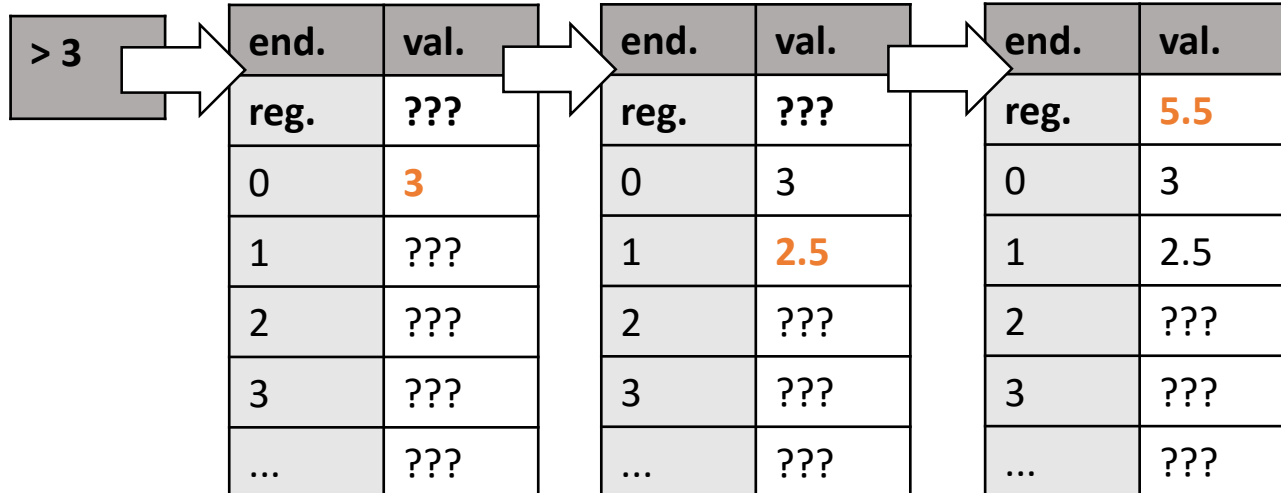
# O que é um programa?

- Sequência de instruções

- Exemplo: programa que escreve na tela o resultado da soma de 2.5 mais um número lido do teclado

```
1.read 0
2.storeconst 2.5 1
3.add 0 1
4.store 2
5.write 2
```

read 0                      storeconst 2.5 1                      add 0 1



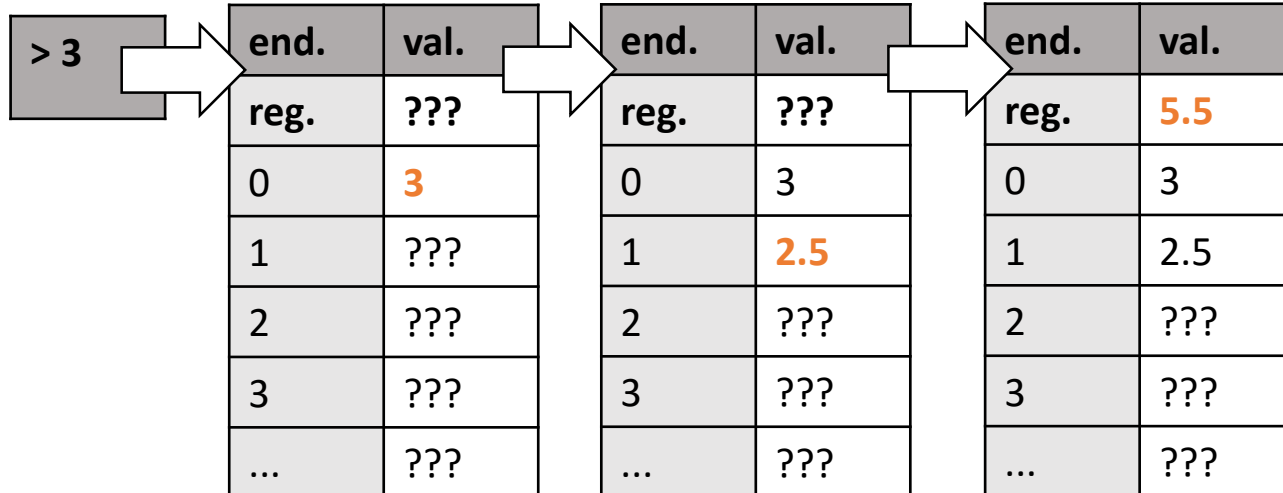
# O que é um programa?

- Sequência de instruções

- Exemplo: programa que escreve na tela o resultado da soma de 2.5 mais um número lido do teclado

```
1.read 0
2.storeconst 2.5 1
3.add 0 1
4.store 2
5.write 2
```

read 0                      storeconst 2.5 1                      add 0 1                      store 2



# O que é um programa?

- Sequência de instruções

- Exemplo: programa que escreve na tela o resultado da soma de 2.5 mais um número lido do teclado

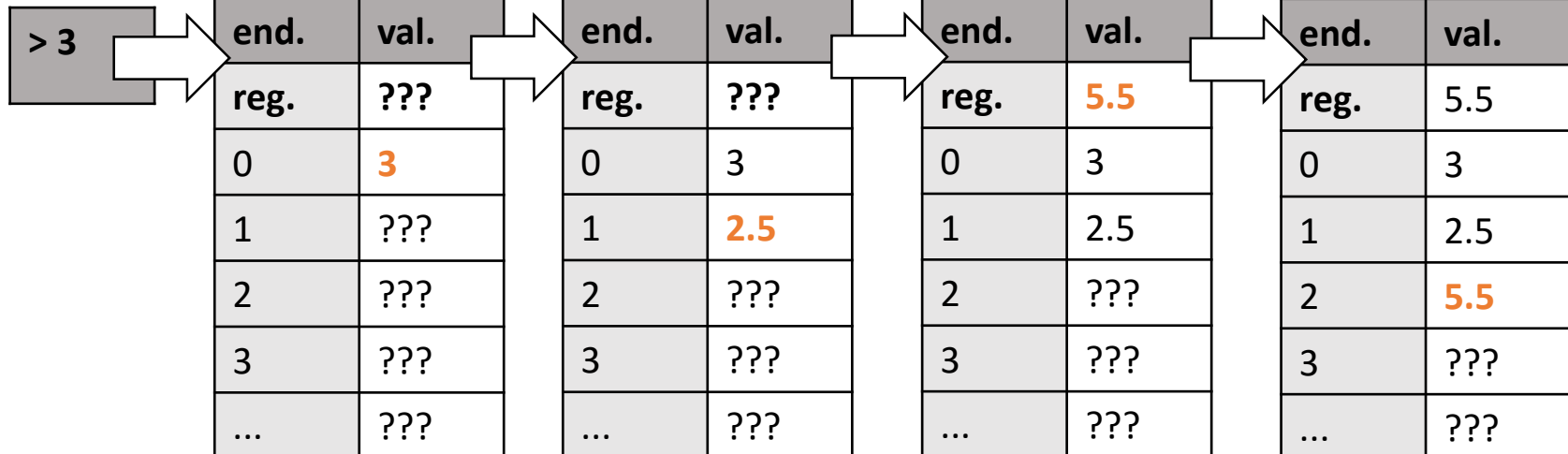
```
1.read 0
2.storeconst 2.5 1
3.add 0 1
4.store 2
5.write 2
```

read 0

storeconst 2.5 1

add 0 1

store 2



# O que é um programa?

- Sequência de instruções

- Exemplo: programa que escreve na tela o resultado da soma de 2.5 mais um número lido do teclado

```
1.read 0
2.storeconst 2.5 1
3.add 0 1
4.store 2
5.write 2
```

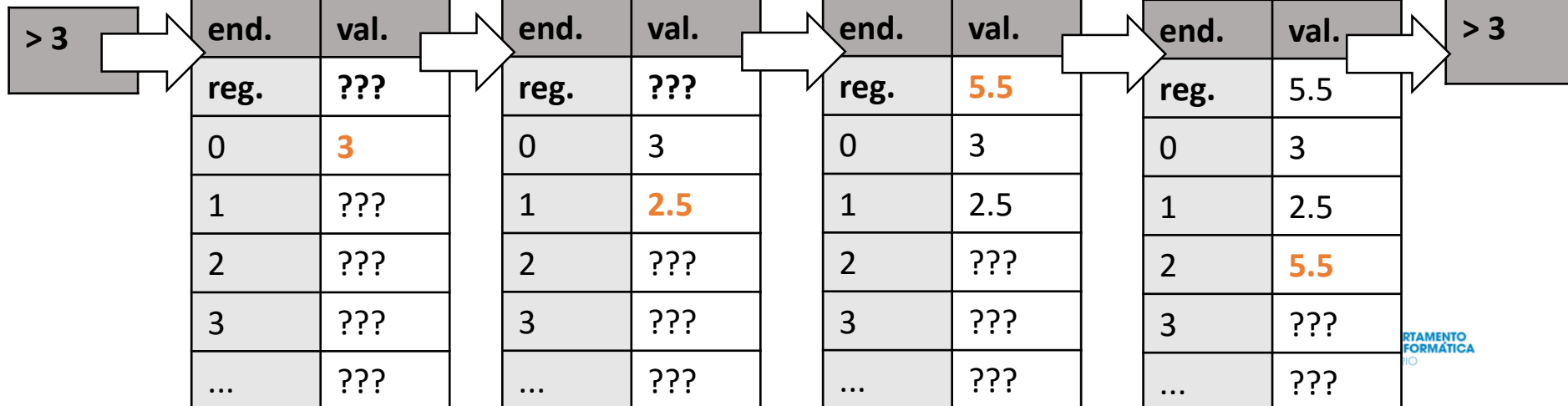
read 0

storeconst 2.5 1

add 0 1

store 2

write 2

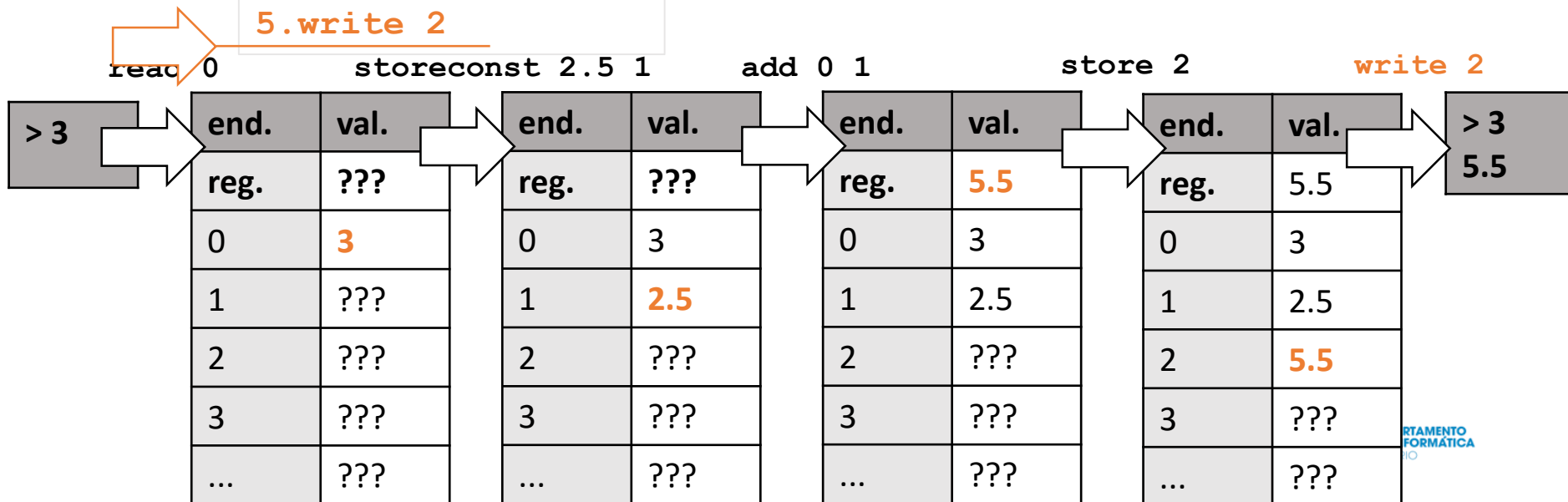


# O que é um programa?

- Sequência de instruções

- Exemplo: programa que escreve na tela o resultado da soma de 2.5 mais um número lido do teclado

```
1.read 0
2.storeconst 2.5 1
3.add 0 1
4.store 2
5.write 2
```





# Dúvidas?

Prof. Roberto Azevedo  
razevedo@inf.puc-rio.br



# O que é um programa?

- Sequência de instruções

- Exemplo: programa que escreve na tela o resultado da soma de 2.5 mais um número lido do teclado

Programa visto anteriormente:

```
1.read 0
2.storeconst 2.5 1
3.add 0 1
4.store 2
5.write 2
```

Este programa também funciona?

```
1.read 0
2.storeconst 2.5 1
3.add 0 1
4.store 0
5.write 0
```

- sim
- não

Este programa também funciona?

```
1.read 0
2.storeconst 2.5 1
3.add 0 1
4.store 0
5.write 2
```

- sim
- não

# Como fica um programa na memória?

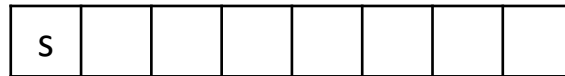
- Suponha que, no nosso computador hipotético:
  - Cada posição de memória tem 8 subseções
  - Cada subseção pode armazenar um algarismo de 0 a 9
- Como representar um número inteiro
  - 0 é representado por 

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---
  - Considerando apenas inteiros positivos, o maior número seria: 

9	9	9	9	9	9	9	9
---	---	---	---	---	---	---	---

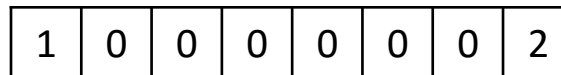
# Como fica um programa na memória?

- Como representar um número negativo?
  - Reservando uma subseção para o sinal (por exemplo, a primeira)

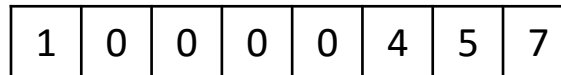


- Se 0 indica positivo e 1 indica negativo

- -2 representado por



- -457 representado por

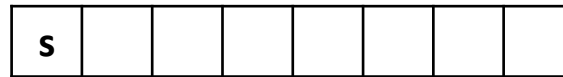


# Como fica um programa na memória?

- Como representar um número fracionário?

- Reservando subseções para (ponto fixo):

- Sinal



- Parte inteira



- Parte fracionário



- Qual é o maior número que poderíamos representar usando essa convenção?



- E o menor número positivo?



# Como fica um programa na memória?

- Como representar um número fracionário?

- Assumindo a seguinte notação científica:

- $-257.4 = -0.2574 \times 10^3$

- Reservando subseções para (ponto flutuante)

- Mantissa (com sinal)

Ms	M	M	M	M			
----	---	---	---	---	--	--	--

- Expoente (com sinal)

					Es	E	E
--	--	--	--	--	----	---	---

- Segundo essa convenção:

- -257.4 é representado por

1	2	5	7	2	0	0	3
---	---	---	---	---	---	---	---

- Qual é o maior número?

0	9	9	9	9	0	9	9
---	---	---	---	---	---	---	---

- E o menor positivo?

0	0	0	0	1	1	9	9
---	---	---	---	---	---	---	---

# Como fica um programa na memória?

- E se o número não couber?
  - $0.9999 \times 10^9 + 1$  : overflow
  - 2.9375: perda de precisão -> 2.937

0	2	9	3	7	0	0	1
---	---	---	---	---	---	---	---

# Como fica um programa na memória?

- Como representar instruções?
  - Associando um código a cada instrução
    - read: 0; write 1; storeconst 2; add: 3; sub: 4; mul: 5; div: 6; store: 7
  - Dependendo da instrução, os valores seguintes indicam parâmetros
    - read **pos**; write **pos**; storeconst **val pos**; add **pos1 pos2**; ...; store **pos**:

```
read 0
storeconst 2.5 1
add 0 1
store 2
write 2
```

```
0 0 0 0 0 0 0 0 # código de 'read' : 0
0 0 0 0 0 0 0 0 # 0
0 0 0 0 0 0 0 2 # código de 'storeconst' : 2
0 2 5 0 0 0 0 1 # 2.5 (valor real)
0 0 0 0 0 0 0 1 # 1
0 0 0 0 0 0 0 3 # código de 'add' : 3
0 0 0 0 0 0 0 0 # 0
0 0 0 0 0 0 0 1 # 1
0 0 0 0 0 0 0 7 # código de 'store' : 7
0 0 0 0 0 0 0 2 # 2
0 0 0 0 0 0 0 1 # código de 'write' : 1
0 0 0 0 0 0 0 2 # 2
```



# Dúvidas?

Prof. Roberto Azevedo  
razevedo@inf.puc-rio.br



# Decifrando o código

- Considerando a representação numérica e o código utilizado, o que faz o programa a seguir?

read: 0; write: 1; storeconst: 2; add: 3; sub: 4; mul: 5; div: 6; store: 7

código equivalente		memória a cada passo							
0 0 0 0 0 0 0 0	-----	end	P1	P2	P3	P4	P5	P6	P7
0 0 0 0 0 0 0 0	-----	...							
0 0 0 0 0 0 0 0	-----	3							
0 0 0 0 0 0 0 1	-----	2							
0 0 0 0 0 0 0 5	-----	1							
0 0 0 0 0 0 0 0	-----	0							
0 0 0 0 0 0 0 1	-----	reg							
0 0 0 0 0 0 0 7	-----								
0 0 0 0 0 0 0 2	-----								
0 0 0 0 0 0 0 1	-----								
0 0 0 0 0 0 0 2	-----								

descrição em português

-----  
-----  
-----

# Decifrando o código

- Considerando a representação numérica e o código utilizado, o que faz o programa a seguir?

read: 0; write: 1; storeconst: 2; add: 3; sub: 4; mul: 5; div: 6; store: 7

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 5
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 7
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 2
0 3 1 4 2 0 0 1
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 5
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 7
0 0 0 0 0 0 0 2
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 2
```

código equivalente

```
-----
-----
-----
-----
-----
-----
-----
```

descrição em português

```
-----
-----
-----
```

memória a cada passo

end	P1	P2	P3	P4	P5	P6	P7
...							
3							
2							
1							
0							
reg							

# Dúvidas?

Prof. Roberto Azevedo  
razevedo@inf.puc-rio.br

