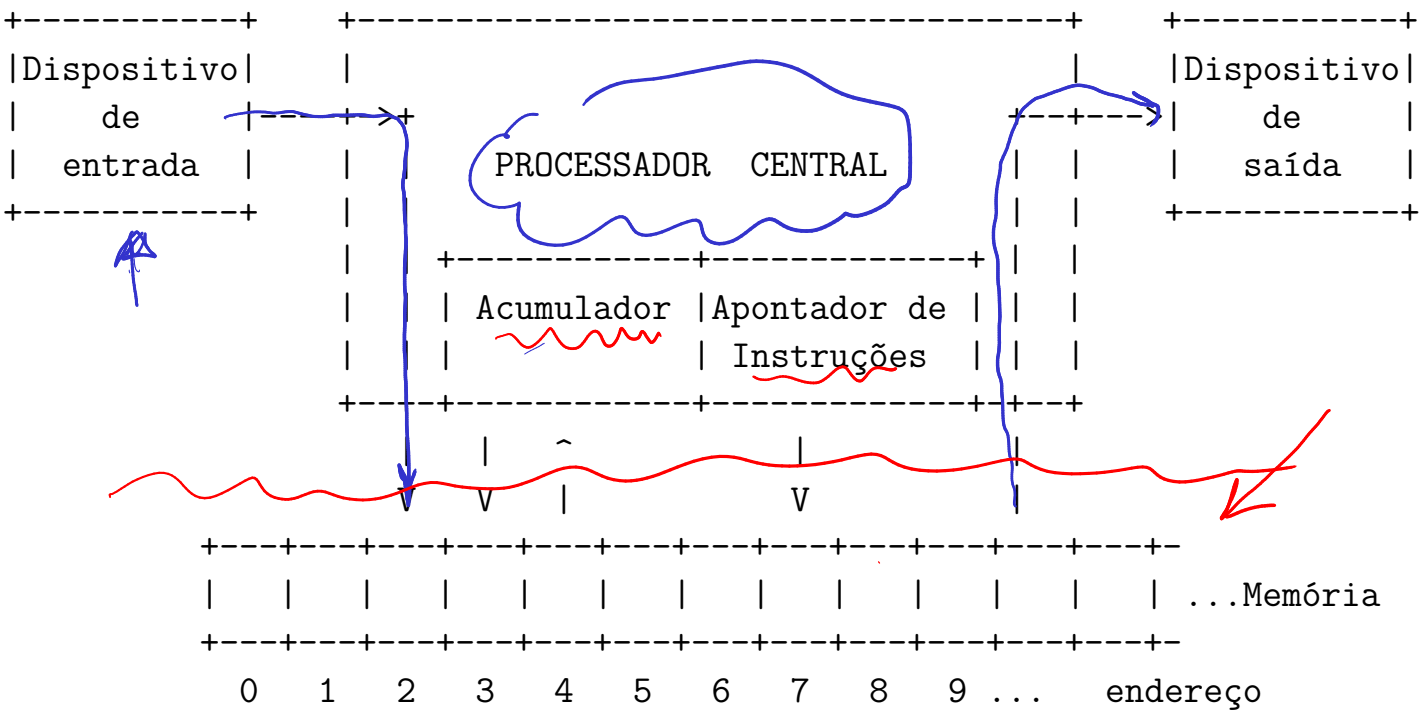
1.1 Aula 01: 01/SET/2020

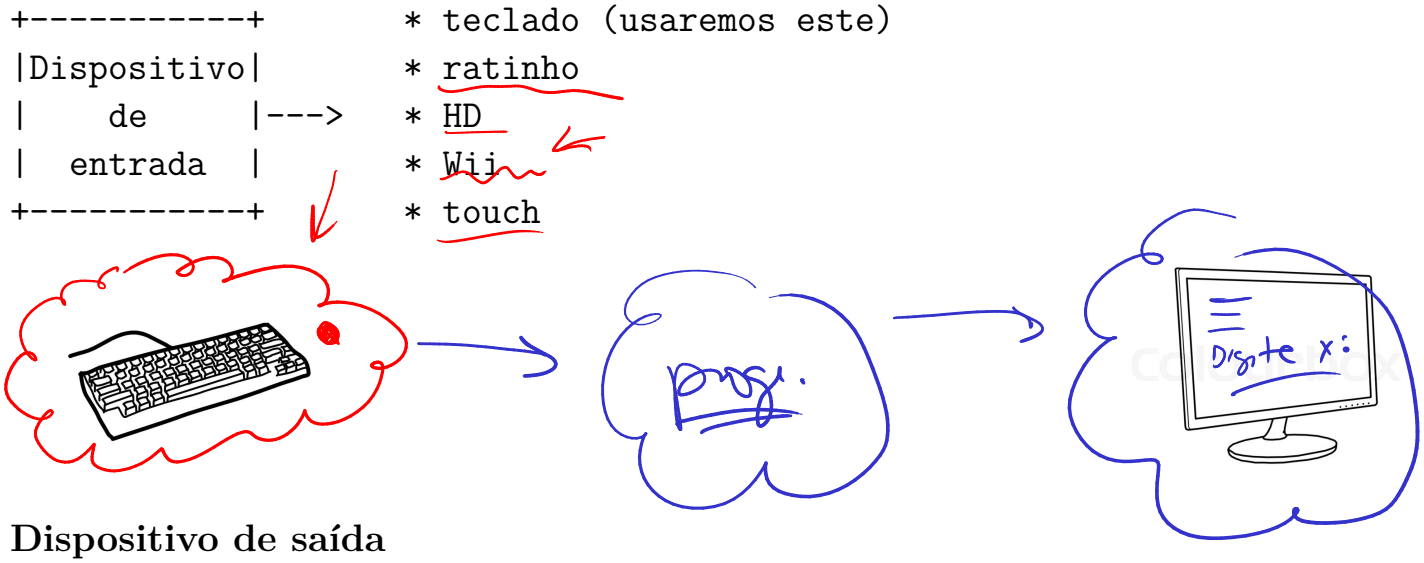
Hoje:

- computador simples
- programa simples no computador simples
- tradução do programa simples para um programa mais simples
- alguns conceitos

1.2 Computador simples



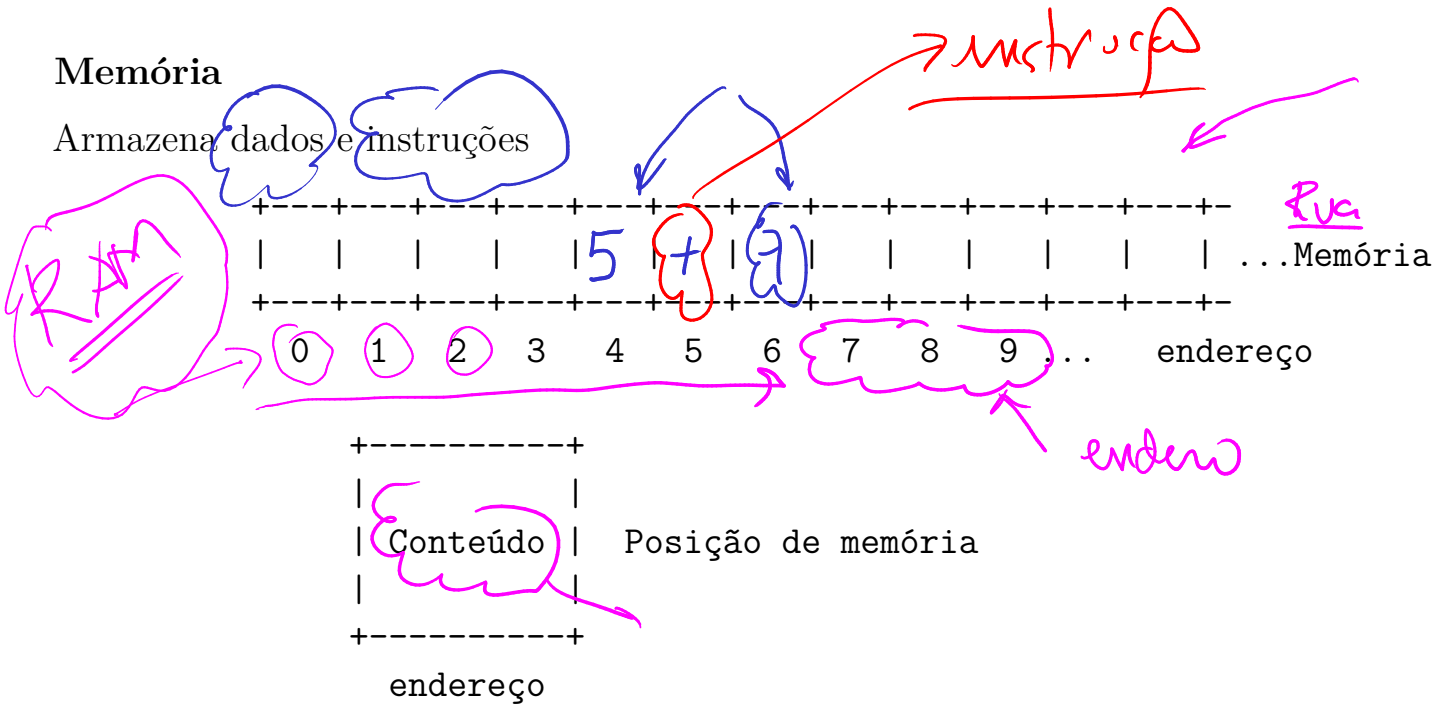
Dispositivo de entrada



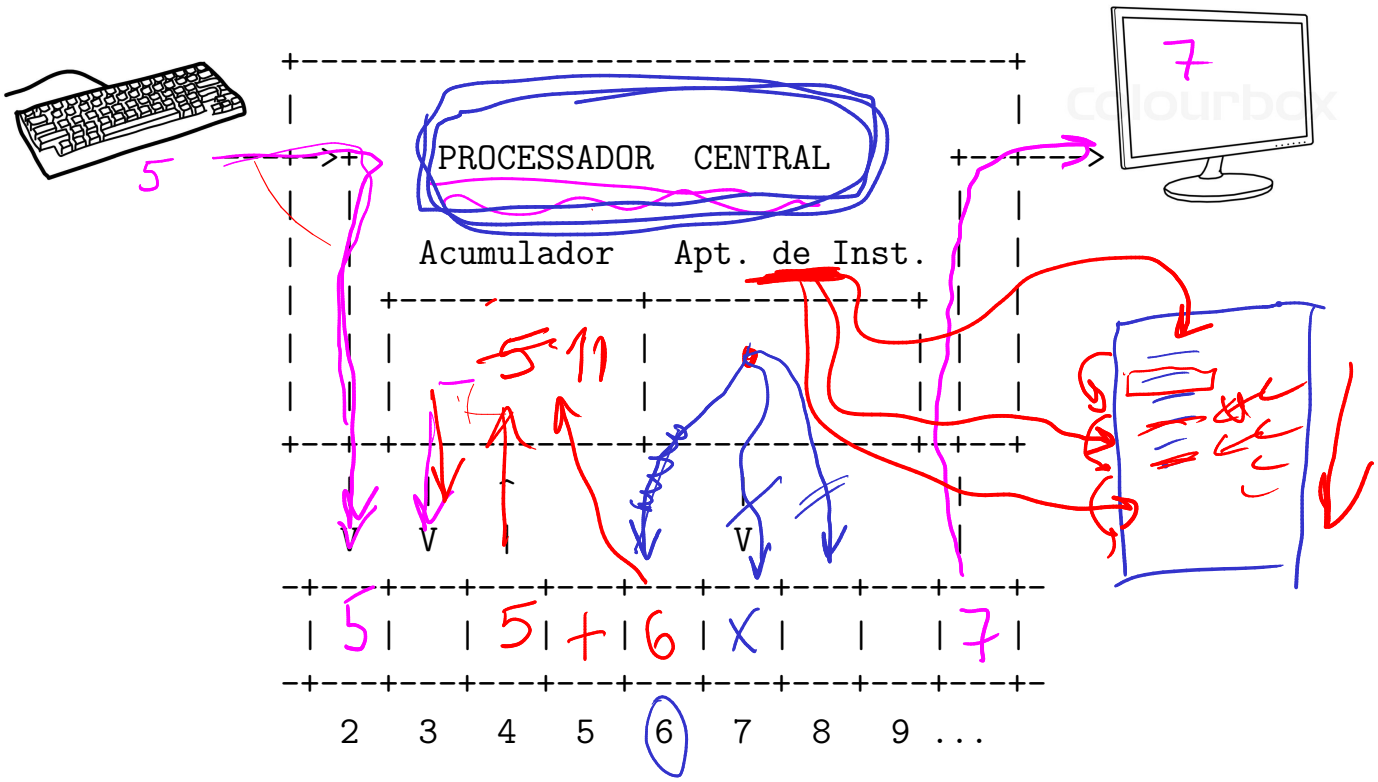
Dispositivo de saída

Memória

Armazena dados e instruções



Processador central



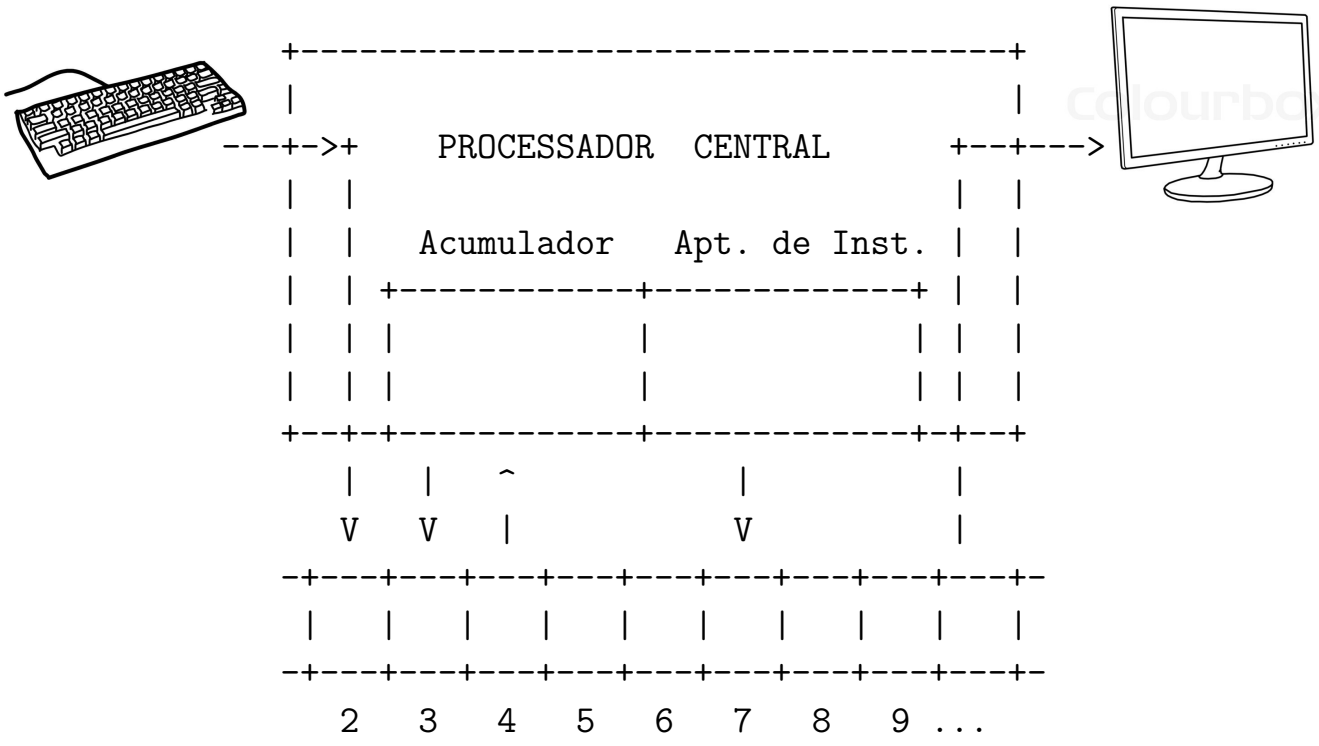
Pentium, Motorola,

Coordena o funcionamento do computador:

- analisa e executa cada instrução;
- obtém da memória os dados necessário para executar instruções e coloca resultados na memória;
- quando copia informações da memória, não as destrói, elas podem ser utilizadas novamente
- ativa equipamentos de entrada e saída

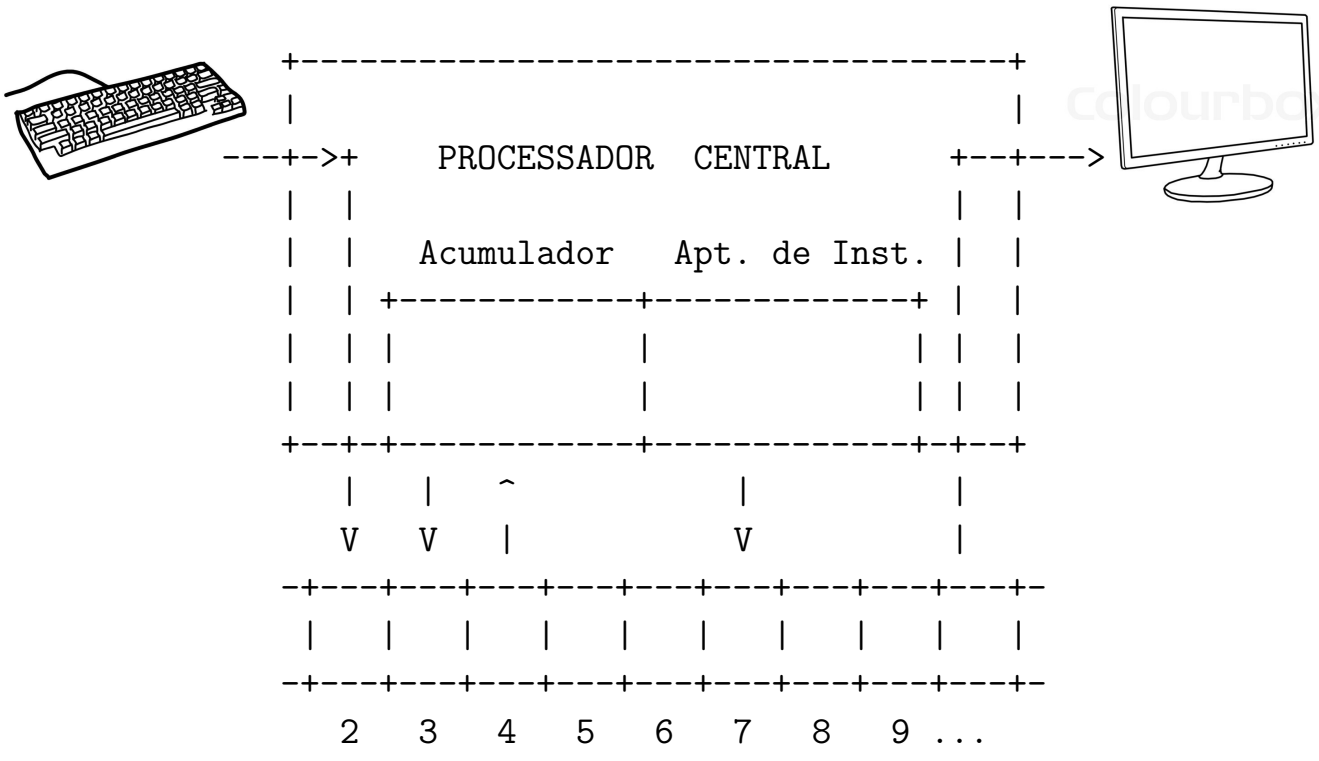
Acumulador

Quando uma operação aritmética (como `+`, `-`, `*`, `/`) é executada, um dos operandos deve estar no acumulador e o resultado da operação é colocado no acumulador.



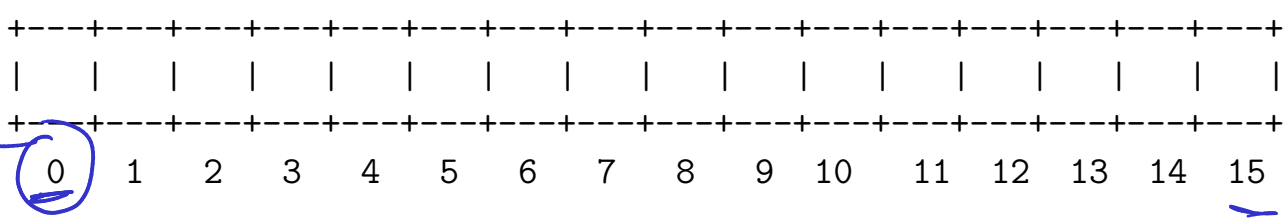
Apontador de instruções

Indica a posição da memória onde está a próxima instrução a ser executada



1.3 Programa simples em um computador simples

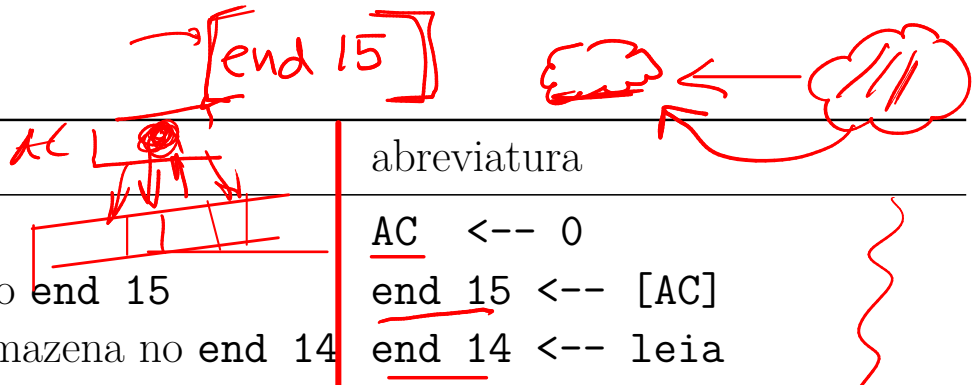
Nosso computador terá 16 posições de memória: 0, 1, . . . , 15



Abreviaturas

- `AC`: acumulador
- `end EE`: endereço `EE` (`EE` = `00`, `01`, `02`, . . . , `15`)
- `[AC]`: conteúdo do `AC`
- `[end EE]`: conteúdo do EE

Programa



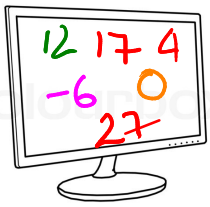
end	conteúdo	abreviatura
00	**carregue** 0 no AC	AC <-- 0
01	**armazene** o [AC] no end 15	end 15 <-- [AC]
02	**leia** um número e armazena no **end 14**	end 14 <-- leia
03	**escreva** [end 14]	**escreva** [end 14]
04	**carregue** no AC o [end 14]	AC <-- [end 14]
05	**desvie** para end 12 se [AC]=0	**desvie** para end 12 se [AC]=0
06	**carregue** no AC o [end 15]	AC <-- [end 15]
07	**some** [AC] ao [end 14] e **armazene** resultado no AC	AC <-- [AC] + [end 14]
08	**armazene** [AC] no end 15	end 15 <-- [AC]
09	**leia** um número e armazene no **end 14**	end 14 <-- **leia**
10	**escreva** [end 14]	**escreva** [end 14]
11	**desvie** para o end 04	**desvie** para o end 04
12	**escreva** [end 15]	**escreva** [end 15]
13	**pare**	**pare**
14	??????	
15	??????	

Entrada:

12 17 4 -6 0

```
          |
          V
+----------------------+
|  PROCESSADOR         |
|        Acumulador    |
|   +----------------+ |
|   |                | |
|   +----------------+ |
|   |                | |
|   +----------------+ |
|       Apt. de Instr  |
|                      |
+----------------------+
          |
          V
```

Saída:



Memória

```
+-------------------------------+
|  0 | AC  <-- 0                |
|  1 | end 15 <-- [AC]          |
|  2 | end 14 <-- leia          |
|  3 | escreva  [end 14]        |
|  4 | AC <-- [end 14]          |
|  5 | desvie para end 12 se [AC]=0 |
|  6 | AC <-- [end 15]          |
|  7 | AC <-- [AC] + [end 14]   |
|  8 | end 15 <-- [AC]          |
|  9 | end 14 <-- leia          |
| 10 | escreva [end 14]         |
| 11 | desvie para o end 04     |
| 12 | escreva [end 15]         |
| 13 | pare                     |
+-------------------------------+
| 14 |                          |
+-------------------------------+
| 15 |                          |
+-------------------------------+
```

Variáveis

Perguntas

O que faz o programas?

Qual o papel do número 0?

Qual o papel do `end 14`?

Qual o papel do `end 15`?

1.4 Tradução ...

Problema: Dada uma sequência de números inteiros diferentes de zero, terminada por um zero, calcular a sua soma. Por exemplo, para a sequência

```
12    17    4    -6    8    0
```

o seu programa deve escrever o número 35.

end	conteúdo	tradução para um programa mais simples
00	**carregue** 0 no AC	soma ← 0
01	**armazene** o [AC] no end 15	
02	**leia** um número e armazena no end 14	num ← leia()
03	**escreva** [end 14]	escreva(num)
04	**carregue** no AC o [end 14]	enquanto num ≠ 0 faça
05	**desvie** para end 12 se [AC]=0	
06	**carregue** no AC o [end 15]	soma ← soma + num
07	**some** [AC] ao [end 14] e **armazene** resultado no AC	num ← leia()
08	**armazene** [AC] no end 15	escreva(num)
09	**leia** um número e armazene no end 14	
10	**escreva** [end 14]	
11	**desvie** para o end 04	escreve(soma)
12	**escreva** [end 15]	
13	**pare**	
14		
15		

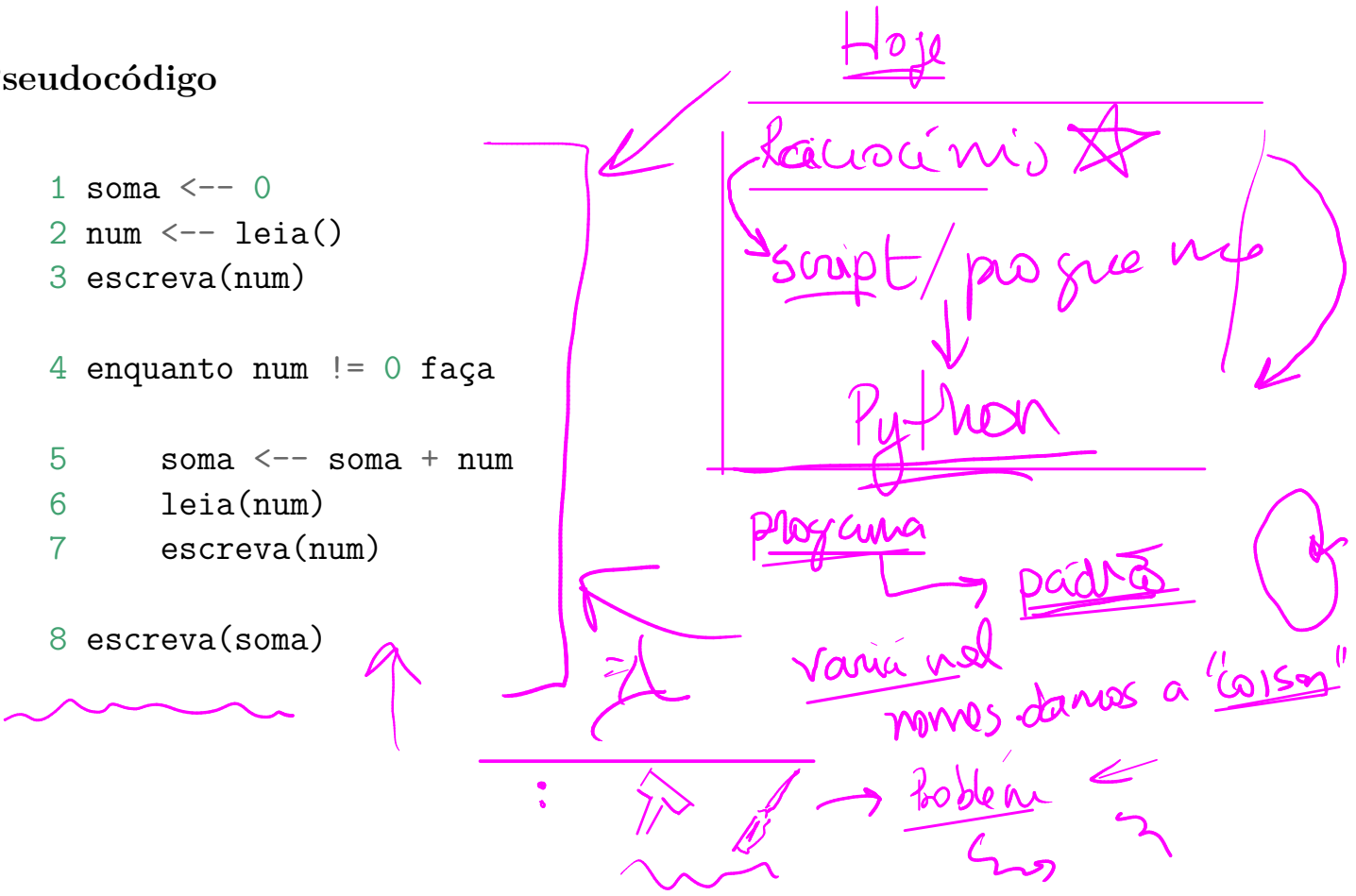
Pseudocódigo

```
1 soma <-- 0
2 num <-- leia()
3 escreva(num)

4 enquanto num != 0 faça

5     soma <-- soma + num
6     leia(num)
7     escreva(num)

8 escreva(soma)
```



Simulação

numero	soma	comando	linha
	0		1
10			2
			3 escreve 10
			4
	10		5
-3			6
			7 escreve -3
			4
	7		5
12			6
			7 escreve 12
			4
	19		5
0			6
			7 escreve 0
			4
			8 escreve 19