Introdução

Prof. Max Vieira Santiago

O mundo hoje

Computadores querem ser úteis...

- Computadores foram criados por uma razão - fazer coisas para nós
- Mas precisamos falar a língua deles para dizer o que queremos que seja feito.
- Usuários comuns têm comodidade - alguém já colocou muitos programas diferentes (instruções) dentro do computador.
- Os usuários só escolhem quais programas querem usar.

Usuários vs. Programadores

- Usuários veem computadores como um conjunto de ferramentas: navegadores Web, processadores de texto, planilhas de cálculos, mapas, lista de tarefas, etc.
- Programadores aprendem como o computador funciona e a linguagem que o computador entende.
- Programadores têm ferramentas que os ajudam a criar novas ferramentas.

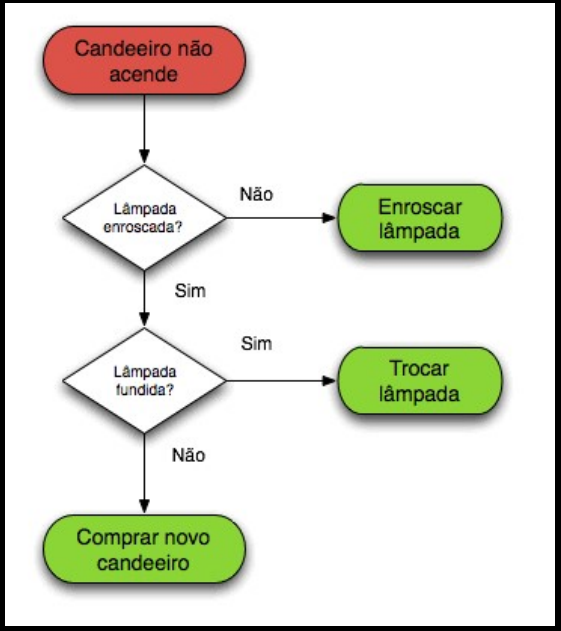
Por que ser um programador?

- Criar algo para outros usarem.
- Limpar e analisar dados de pesquisa.
- Resolver um problema de desempenho em um software.
- Adicionar um mecanismo de busca em um site.
- ... o céu é o limite

Algoritmo

- Sequência finita de instruções que levam à solução de um problema em um número finito de passos.
- Exemplo: Receita de bolo

1. Adicione 3 ovos
2. Adicione 500ml de leite
3. Bata todos os ingredientes em uma batedeira
4. ...



O que é código? Software? Programas?

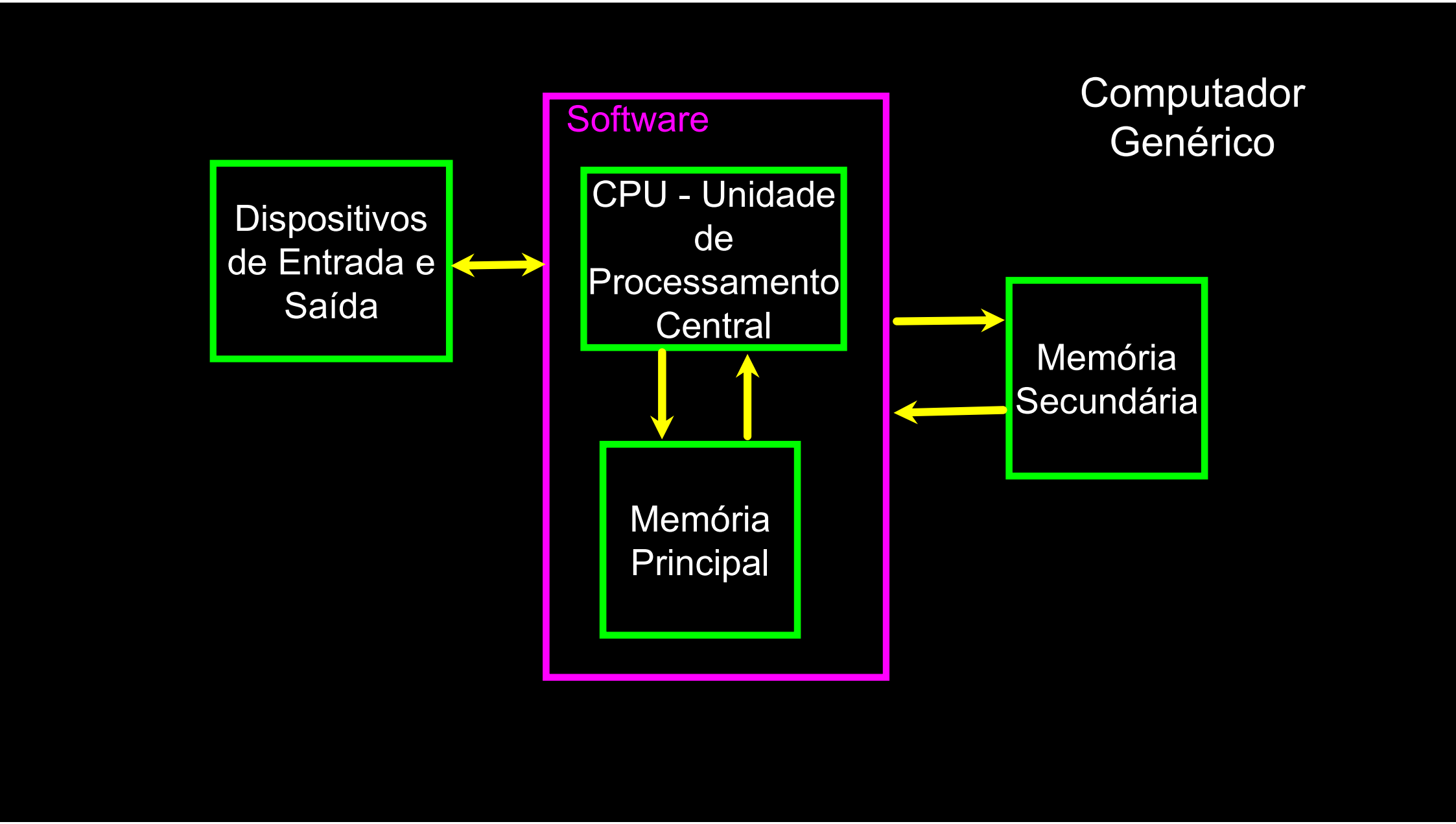
- Um conjunto de instruções armazenadas.
- É um pequeno pedaço de nossa inteligência, dentro do computador.
- É um pequeno pedaço de nossa inteligência que podemos dar os outros.
  - Podemos imaginar algo útil, escrever isso e dar para outras pessoas de forma a lhes poupar tempo e energia resolvendo o mesmo problema.
- Um pedaço de arte criativa - particularmente quando fazemos um bom trabalho na experiência do usuário.

```
name = input('Enter file:')
handle = open(name, 'r')
text = handle.read()
words = text.split()

counts = dict()
for word in words:
   counts[word] = counts.get(word,0) + 1
bigcount = None
bigword = None

for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count
print(bigword, bigcount)
```

```
python words.py
Enter file: words.txt
to 16
```

```
python words.py
Enter file: clown.txt
the 7
```

Hardware Architecture
Arquitetura de Hardware


Computador
Genérico
Software
Dispositivos
de Entrada e
Saída
CPU - Unidade
de
Processamento
Central
Memória
Principal
Memória
Secundária

Definições

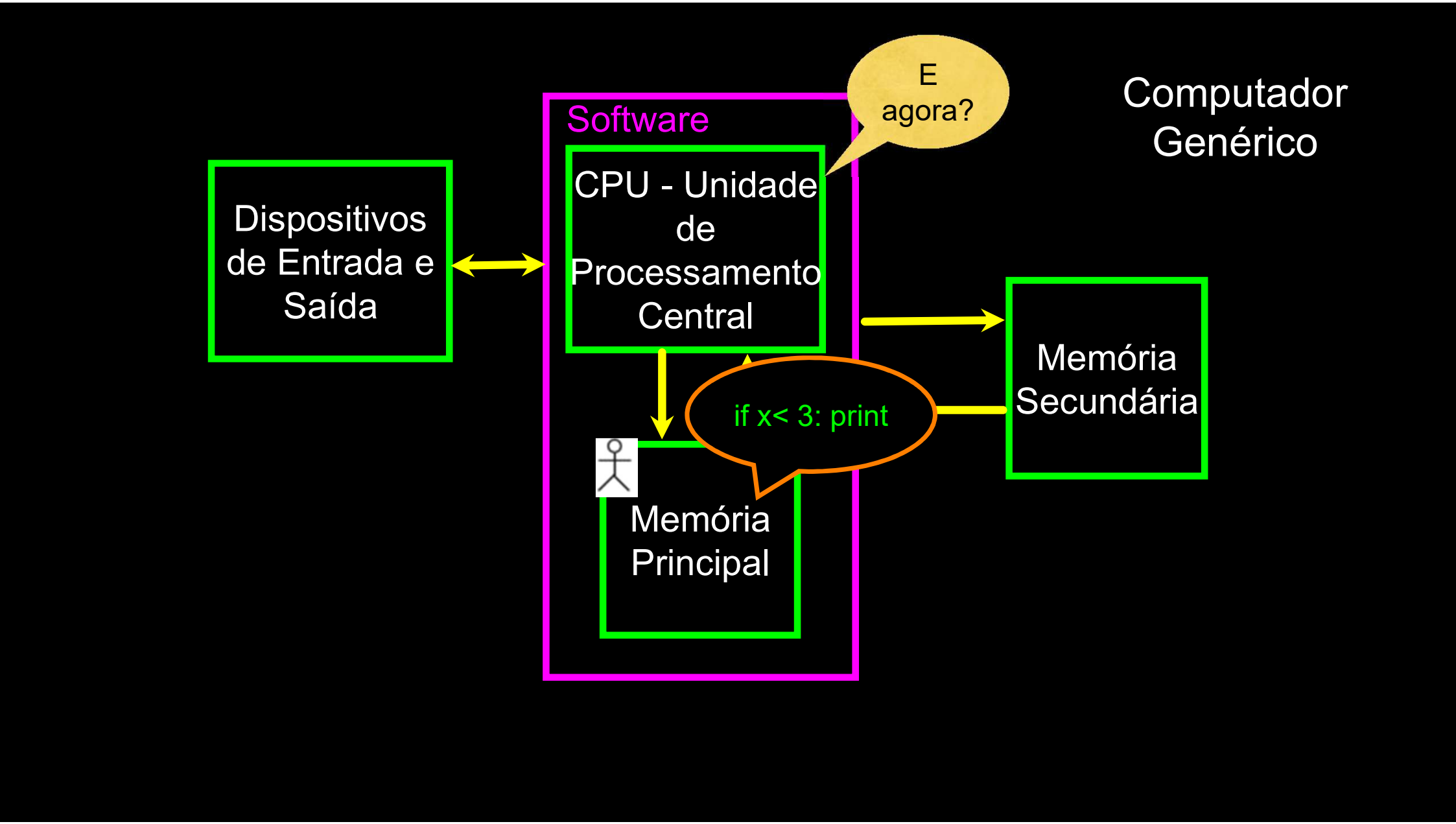
- Central Processing Unit(Unidade de processamento central):
  Roda o programa - A CPU está sempre se perguntando: “O que fazer agora”? Não exatamente um cérebro - muito “burro”, mas muito rápido

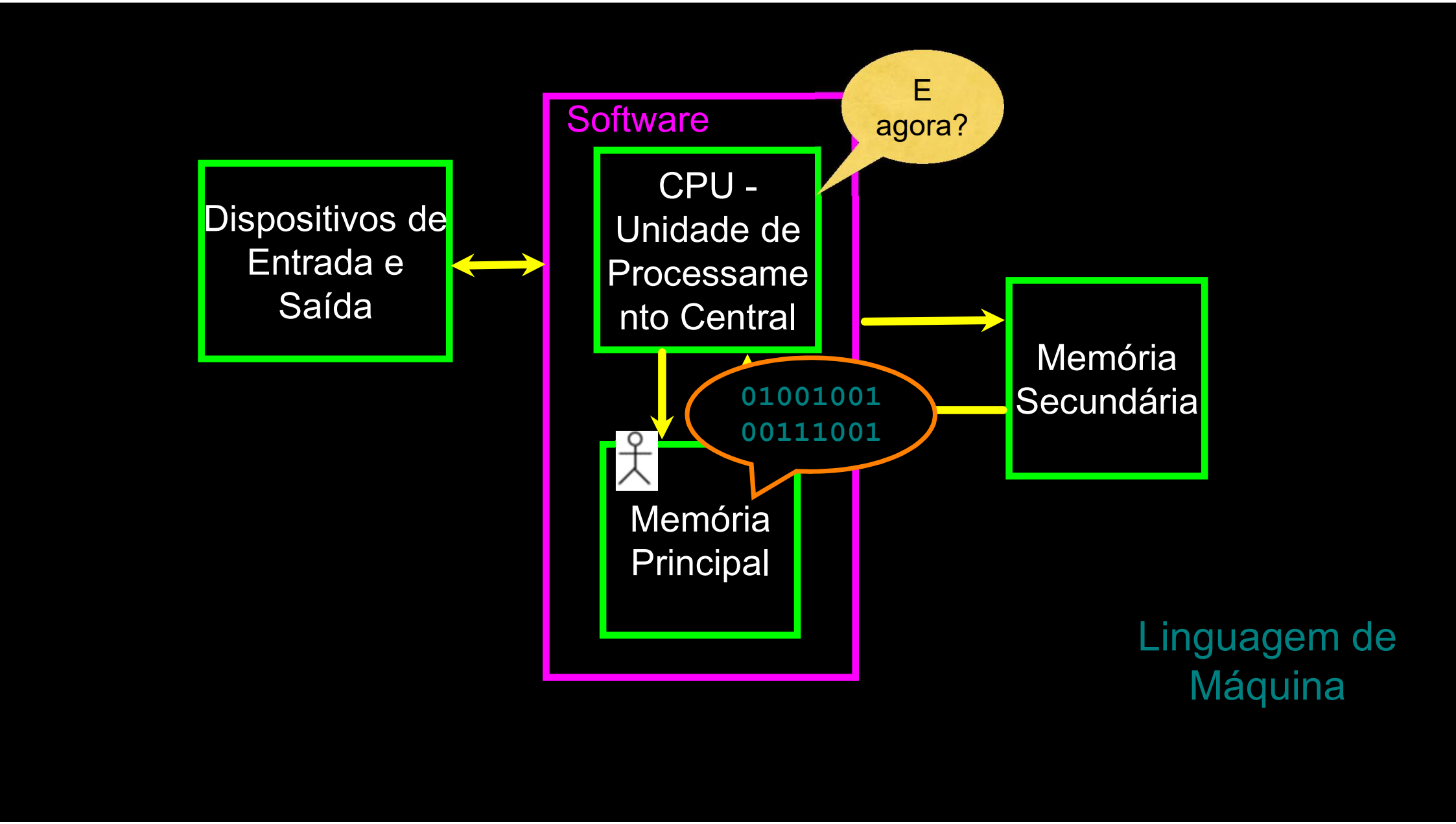
- Input Devices(Dispositivos de entrada): Teclado, Mouse, Tela sensível ao toque.

- Output Devices(Dispositivos de saída): Monitor, Auto-falantes, Impressora, Gravador de DVD

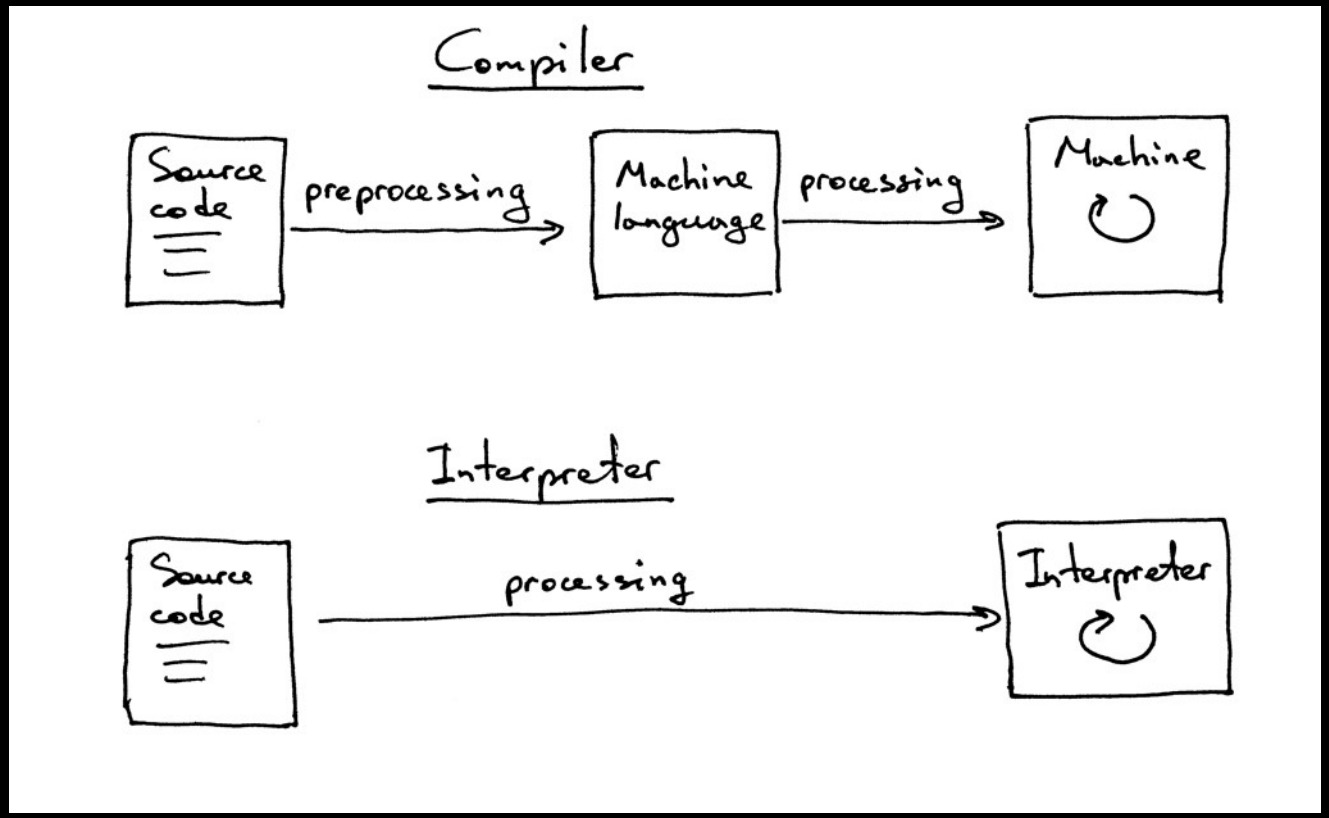
- Main Memory(Memória principal): Rápido e pequeno armazenamento temporário - perdida em reiniciação - também conhecida por RAM (Memória de acesso aleatório)

- Secondary Memory(Memória secundária): Devagar, grande e armazenamento permanente - dura até ser deletado - unidade de disco / cartão de memória


Computador Genérico
Software
E agora?
CPU - Unidade de Processamento Central
Dispositivos de Entrada e Saída
Memória Secundária
if x< 3: print
Memória Principal


Software
E agora?
Dispositivos de Entrada e Saída
CPU - Unidade de Processamento Central
Memória Secundária
01001001
00111001
Memória Principal
Linguagem de Máquina

Compilação x Interpretação



Python como uma linguagem

- Linguagem de alto nível
- Interpretada

Erros de Sintaxe x Erros de Lógica

- Nós precisamos aprender a linguagem Python para que possamos comunicar nossas instruções para o Interpretador Python

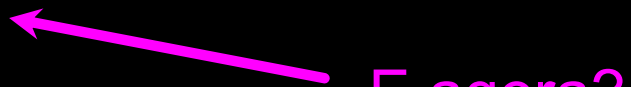
- Quando você erra, o computador não quer saber se você é “sensível”. Ele diz “syntax error” *(erro de sintaxe)*

- *Erros de lógica* são mais difíceis de encontrar
  - O Interpretador Python não vai te avisar que você está cometendo um erro
  - Ao executar o seu programa, o resultado será diferente do esperado
  - Mas onde está o erro?

Modos de utilizar o Python

Python no modo Interativo

```
csev$ python
Python 2.5 (r25:51918, Sep 19 2006, 08:49:13)
[GCC 4.0.1 (Apple Computer, Inc. build 5341)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```



```
csev$ python
Python 2.5 (r25:51918, Sep 19 2006, 08:49:13)
[GCC 4.0.1 (Apple Computer, Inc. build 5341)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> x = 1
>>> print(x)
1
>>> x = x + 1
>>> print(x)
2
>>> exit()
```

Esse é um bom teste para ter certeza que você tem o Python instalado corretamente. Perceba que exit() serve para encerrar a sessão interativa.

Python no modo Script

- Python no modo interativo é bom para experimentos e códigos de 3-4 linhas.
- A maioria dos códigos são bem maiores.
- Logo, escrevemos o código em um arquivo e dizemos ao Python para executar o código contido no arquivo.
- Por convenção, nós adicionamos “.py” como um sufixo no final desses arquivos para indicar que o arquivo contem código em Python.

Escrevendo um programa simples

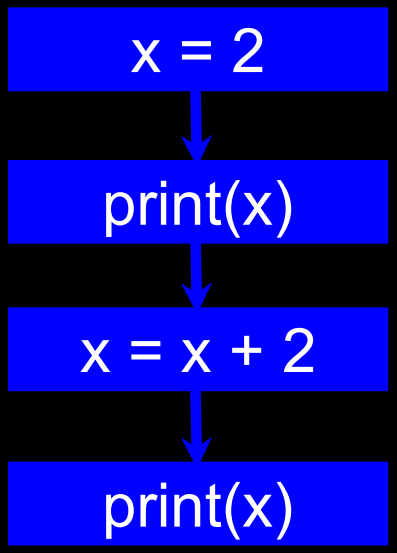
Interativo vs. Script

- Interativo
  - Você digita diretamente no Python uma linha de cada vez e ele responde
- Script
  - Você escreve uma sequência de declarações (linhas) em um arquivo usando um editor de texto e diz ao Python para executar essas declarações no arquivo.

Programar em passos

- Como uma receita ou manual de instalação, um programa é uma sequência de passos para ser feito em ordem.
- Alguns passos são condicionais - eles podem ser pulados.
- Às vezes, um passo ou um grupo de passos tem que ser repetidos.
- Às vezes, nós salvamos um conjunto de passos para serem usados várias vezes quanto for necessário em vários locais do programa.

Passos sequenciais



Programa:

```
x = 2
print(x)
x = x + 2
print(x)
```

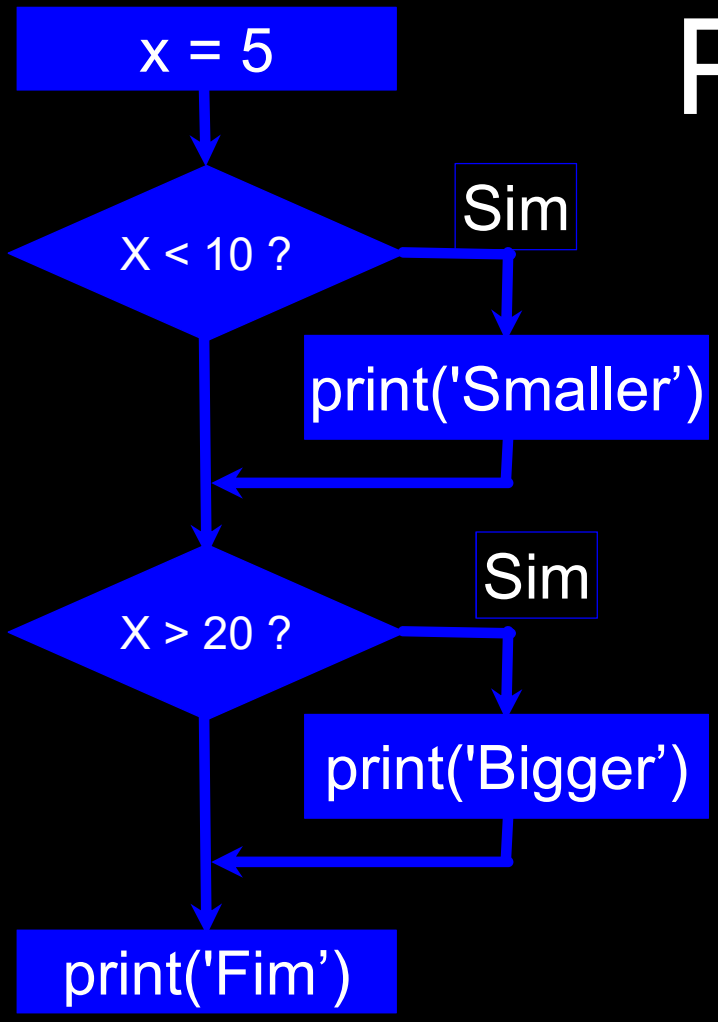
Resultado:

```
2
4
```

Quando um programa está sendo executado, ele vai de um passo para o próximo. Como programadores, nós criamos "paths"(caminhos) para o programa seguir.

Capítulo 2

Passos Condicionais

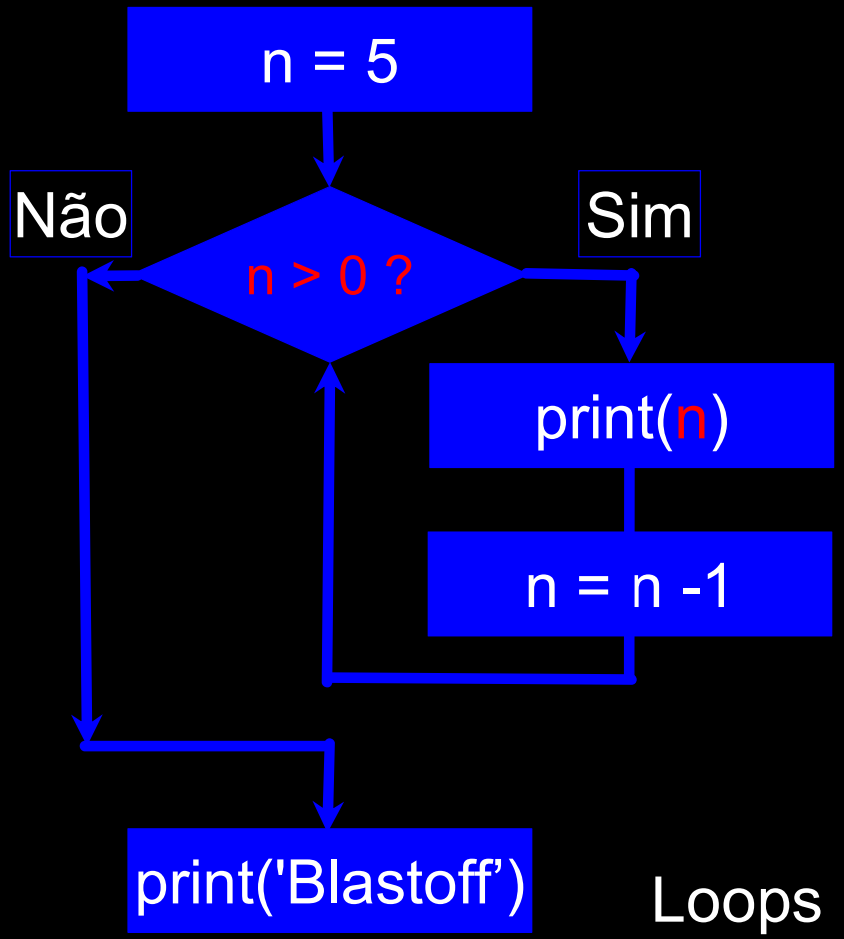


Programa:

```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')

print('Fim')
```

Resultado:

```
Smaller
Fim
```



Passos Repetidos

Programa:

```
n = 5
while n > 0 :
    print(n)
    n = n – 1
print('Blastoff!’)
```

Resultado:

```
5
4
3
2
1
Blastoff!
```

Loops (passos repetidos) tem variáveis de iteração que mudam a cada vez que passam por um loop. Às vezes essas variáveis de iteração vão por uma sequência de números.

Sumário

- Por que programar é importante?
- Visão geral do que será visto no curso.
- Aprofundaremos esses conceitos durante o curso.

Agradecimentos / Contribuições

Initial Development: Charles Severance, University of Michigan School Of Information

Translation: Jonathan Cardozo, Luis Otávio, Silvio L Carvalho, Maria A. Pysklevicz Sakiyama.

Desenvolvimento inicial: Charles Severance, da Universidade de Michigan School of Information

Tradução: Jonathan Cardozo, Luis Otávio, Silvio L Carvalho, Maria A. Pysklevicz Sakiyama.