MAC2166 – Introdução à Computação
Aula 1
Como Funciona um Computador

Paulo Meirelles[1]
paulormm@ime.usp.br

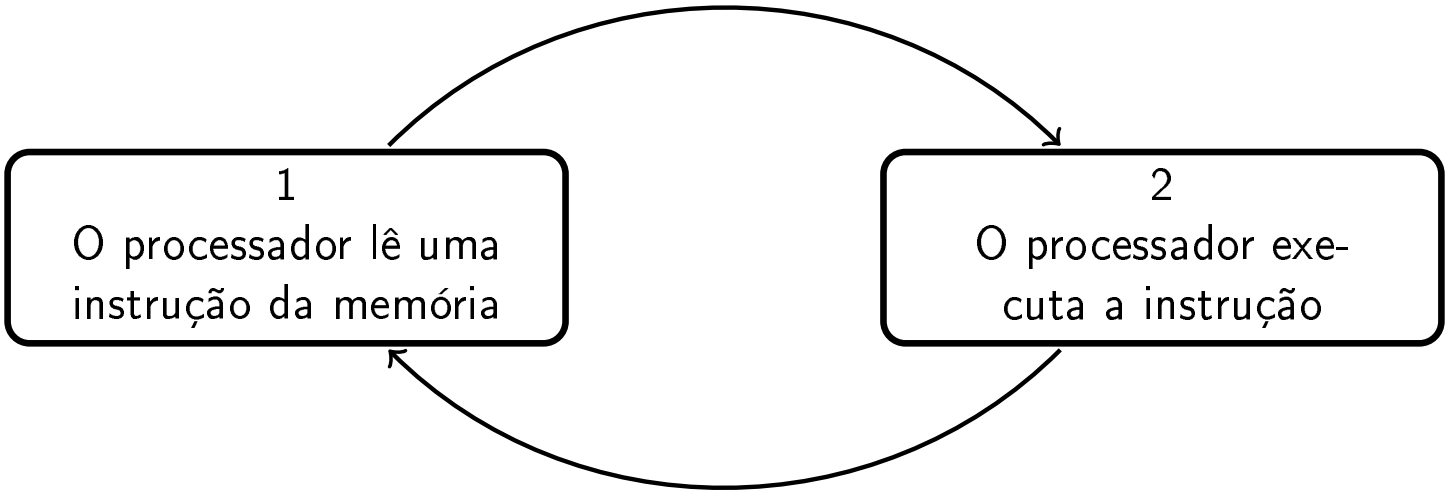
Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

Março de 2023

[1] *slides da Profa. Kelly Rosa Braghetto (IME-USP)*

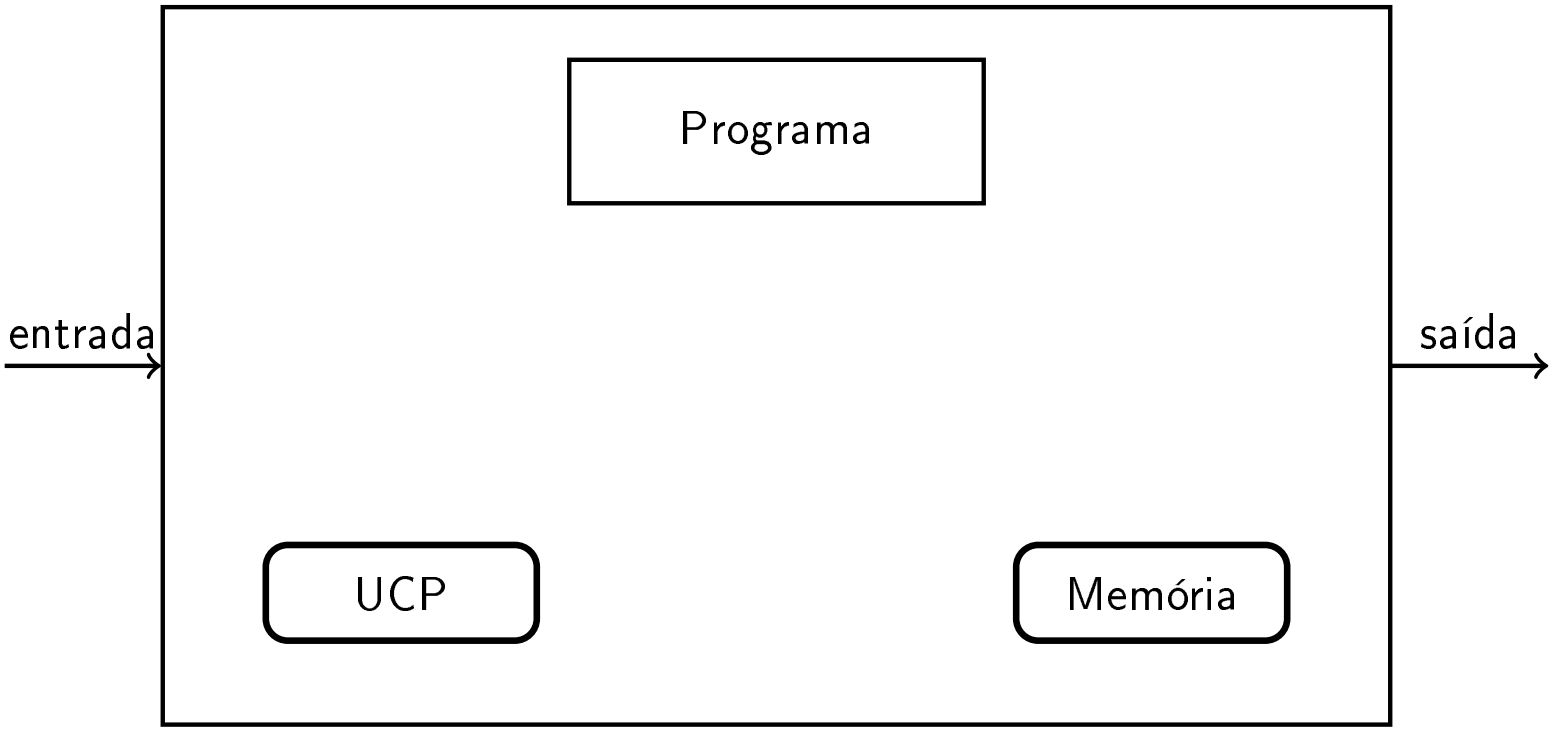
O ciclo de busca e execução

- **Programa**: lista de instruções
- Processador efetua uma computação por meio do **ciclo de busca e execução**:



Arquitetura de computadores – primórdios

- Primeiros computadores eletrônicos (como o ENIAC, de 1945): não armazenavam programas; cada novo cálculo exigia que plugues e cabos fossem movidos para reprogramá-lo

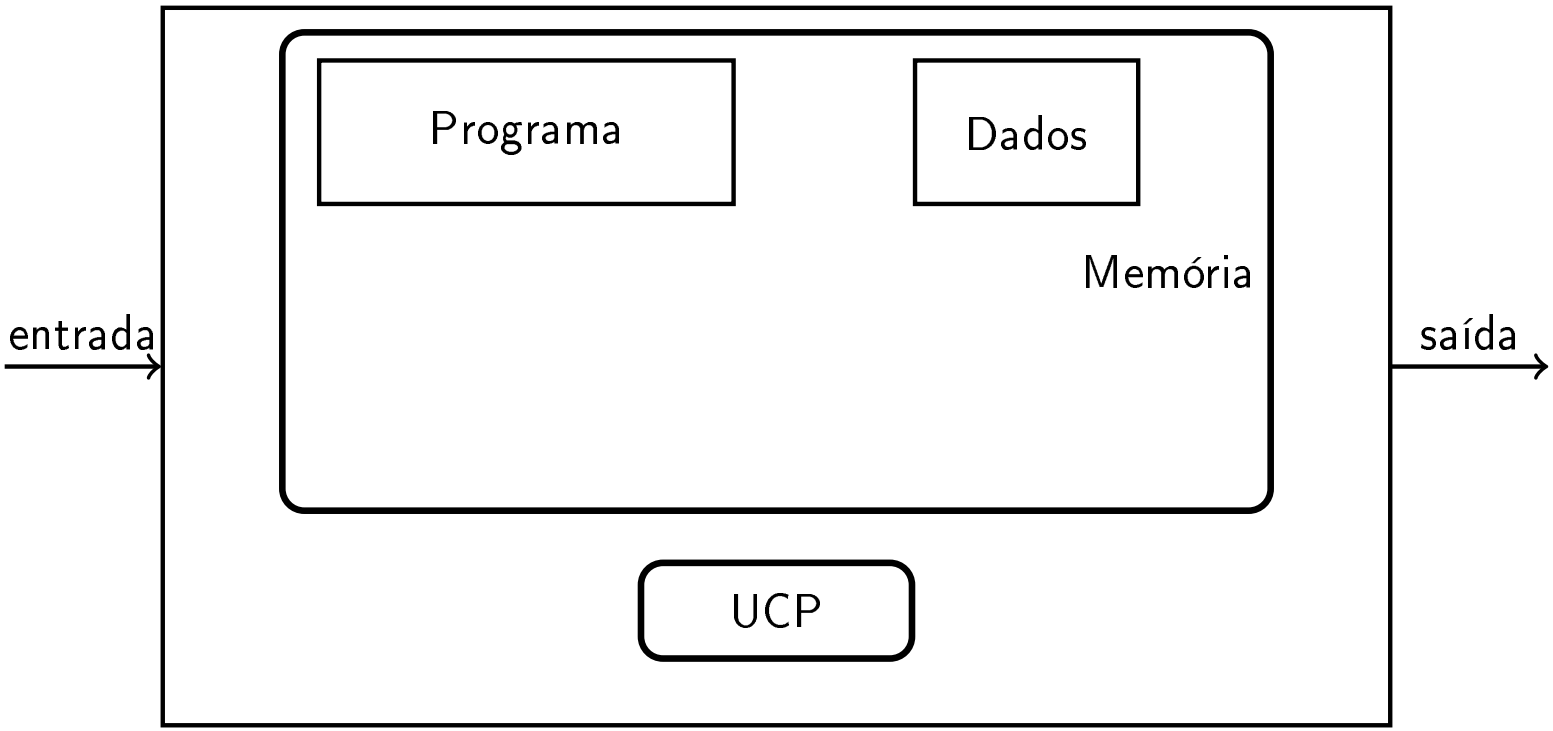


Arquitetura do ENIAC

UCP = Unidade Central de Processamento

Arquitetura de von Neumann (usada nos computadores atuais)

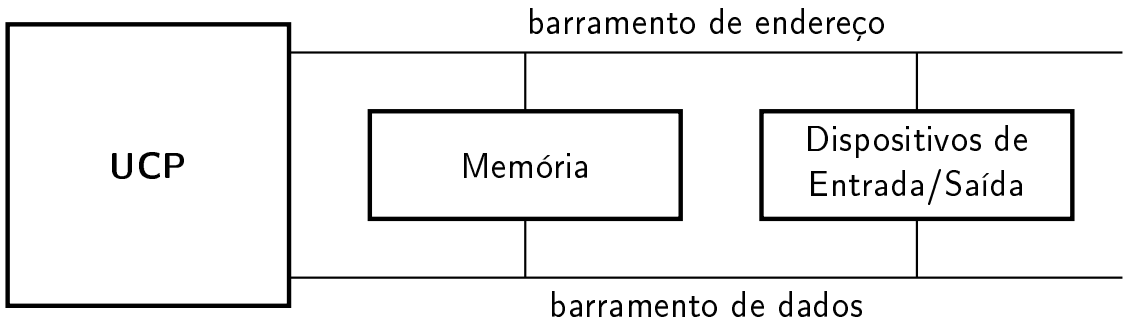
- John von Neumann (matemático consultor do projeto ENIAC) publicou o conceito de "programa armazenado" em 1945

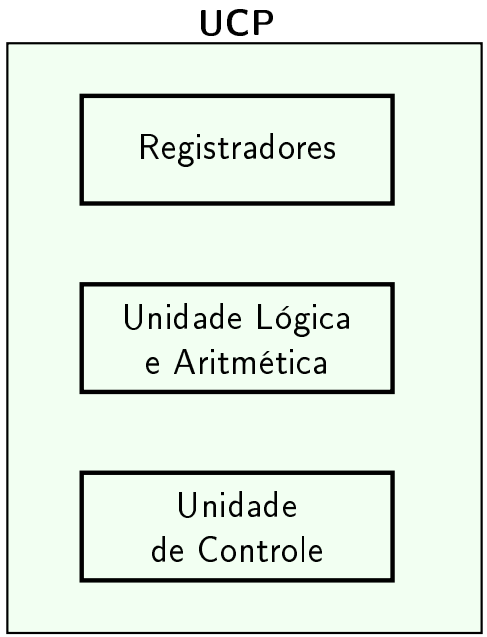


Arquitetura de von Neumann

UCP = Unidade Central de Processamento

Modelo simplificado de um computador





Dentro de uma UCP (processador), temos...

Unidade Lógica e Aritmética (ULA)

- Realiza todas as tarefas relacionadas a operações aritméticas (adições, subtrações, etc.) e a operações de comparação (como *igual* ou *maior que*)

Unidade de Controle (UC)

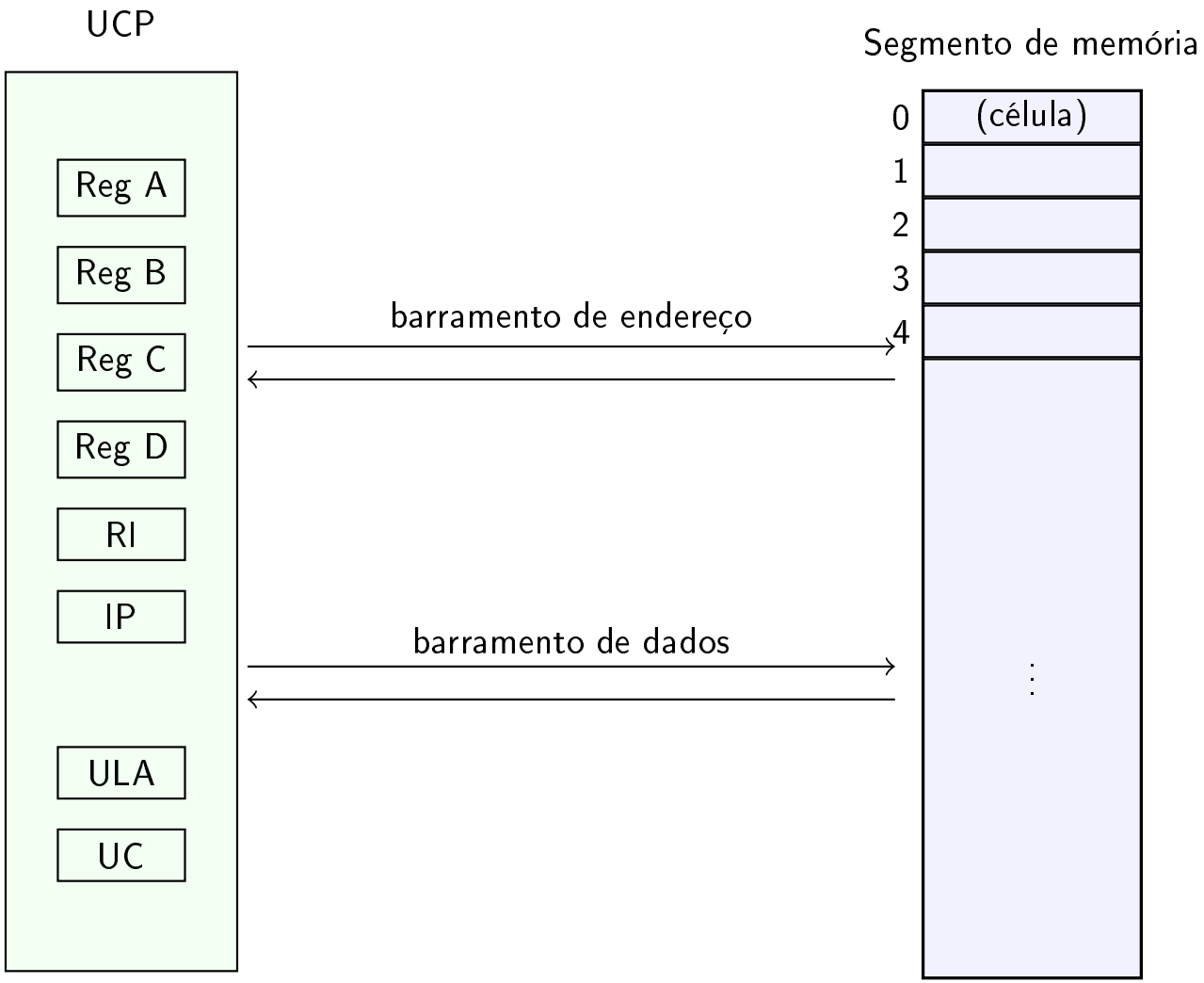
- Controla as ações realizadas pelo computador, comandando todos os demais componentes de sua arquitetura
  1. Lê dados ou instruções da memória ou dos dispositivos de entrada
  2. Decodifica as instruções
  3. Alimenta a ULA com as entradas corretas de acordo com as instruções
  4. Envia os resultados à memória ou aos dispositivos de saída

Dentro de uma UCP (processador), temos...

Registradores

- Um **registrador** é uma coleção de circuitos que armazenam **bits**
- Os registradores de um processador não precisam armazenar uma mesma quantidade de bits (mas é mais fácil de se lidar com eles quando eles são assim)
- A quantidade de bits que se pode armazenar em um registrador típico do processador é um dos atributos que determinam sua classificação (Ex.: processador de 32-bits, ou de 64-bits, etc.)

Registradores



Dentro de uma UCP (processador), temos...

Registradores

- Cada registrador possui uma função própria. Exemplos:
  - **Apontador de instrução (IP, de *instruction pointer* ou contador de programa** (PC, de *program counter*) – aponta para a próxima instrução a executar
  - **Registrador de instrução** (IR, de *instruction register*) – armazena a instrução em execução
  - Armazenamento de resultados intermediários

Conjunto de instruções de um processador

- As **instruções** são as operações que um processador é capaz de realizar; elas são a parte do processador que é "visível" para os programadores
- Cada processador possui o seu próprio conjunto finito de instruções, que pode variar de fabricante para fabricante
- Mas processadores com arquiteturas internas diferentes podem ter um mesmo conjunto de instruções (ex.: Intel* e AMD*)
- Um processador executa instruções em uma dada frequência; exemplos de frequências comuns atualmente:
  - 3,6 GHz, para um computador de mesa (*desktop*)
  - 2,8 GHz, para um computador portátil (*notebook*)
  - 1,9 Ghz, para um computador de mão (*smartphone*)

  Obs.: GHz = bilhões de instruções por segundo

Conjunto de instruções de um processador

Operações

As instruções de um processador se relacionam às seguintes funcionalidades:

- operações matemáticas e lógicas
- movimentação de dados (transferência de dados da memória para os registradores e vice-versa)
- operações de entrada/saída (leitura ou escrita de dados em dispositivos de entrada e saída)
- controle do fluxo de execução (desvios condicionais ou incondicionais)

Memória

- A memória do computador pode ser vista como uma lista de **células**
- Cada célula pode armazenar uma quantidade fixa e pequena de informação. Uma informação pode ser:
  - uma **instrução** – que diz ao computador o que fazer
  - **dados** – a serem processados pelo processador usando as instruções
- Cada célula tem um **endereço** numérico; células contíguas na memória possuem números de endereços sequenciais

Tipos de memória

Memórias voláteis

- São chamadas de memória do tipo RAM – *Random Access Memory*
- Precisam de energia para manter seu conteúdo (= só funcionam com o computador ligado)

Memória não voláteis (ou permanentes)

- Mantém as informações de forma permanente
- São mais baratas que as memórias voláteis e possuem maior capacidade de armazenamento, mas são muito mais lentas
- Exemplo: disco rígido (HD – *Hard Disc*)

$\Rightarrow$ Um computador pessoal mediano da atualidade possui cerca de 4 8GB de RAM e 500GB de HD.

Um "parênteses" sobre bytes e bits...

- Um **bit** (de *binary digit*) é a menor unidade de informação que pode ser armazenada ou transmitida
- Um bit pode assumir somente 2 valores: 0 ou 1 (corte ou passagem de energia, respectivamente)
- Cada célula de uma memória é composta por bits
- Todas as células de uma mesma memória contêm a mesma quantidade de bits
- Geralmente, uma célula contém 8 bits
- **8 bits = 1 byte**
- 1 GB = 10^9 bytes = 1 bilhão de bytes

Dispositivos de Entrada e Saída – E/S

- Definem como o computador recebe informação do mundo exterior e como ele devolve informação para o mundo exterior
- Exemplos de dispositivos de entrada: teclado, *mouse*, *scanner*, microfone e câmera
- Exemplos de dispositivos de saída: monitor, impressora, projetor de vídeo, caixa de som

O ciclo de busca e execução (revisitado)

A UCP executa cada instrução por meio de uma série de pequenos passos:

1. Lê a próxima instrução na memória e a armazena no registrador de instrução (RI)
2. Muda o registrador Apontador de instrução (IP), para que ele aponte para a instrução seguinte
3. Determina o tipo da instrução que acabou de ser lida
4. Se a instrução usa algum dado da memória, determina onde ele está
5. Carrega o dado, se necessário, em um registrador da UCP
6. Executa a instrução
7. Volta para o passo 1, para começar a execução da instrução seguinte

O processador executa esse ciclo infinitas vezes (ou até que o computador seja desligado).

Programas e Algoritmos

- Um **programa** é uma sequência de instruções que ficam armazenadas na memória.
- Quem define como instruções e dados podem ser usados para produzir um programa útil são os **algoritmos**.

Definição informal:

Um algoritmo é uma estratégia para resolver um problema.

Um algoritmo deve ter as seguintes propriedades:

1. Ser **claro** e estar definido de forma **não ambígua**
2. Ser **efetivo**, ou seja, todos os seus passos devem ser exequíveis
3. Ser **finito**, ou seja, deve terminar depois de um número limitado de passos

Algoritmos

- **Algoritmos** são como **receitas de bolo**:
  - Uma receita de bolo determina como utensílios e equipamentos de cozinha (como recipientes, fornos, etc.) devem ser usados para transformar ingredientes em um bolo
- Um **algoritmo** determina como recursos computacionais (memória, processador, dispositivos de E/S, etc.) devem ser usados para processar dados de entrada e produzir a saída desejada

Como os programas são escritos?

- O algoritmo é uma entidade **abstrata**, que define uma ideia
- Um programa é uma **realização** de um algoritmo
- Um mesmo algoritmo pode ser escrito como programas diferentes, usando linguagens diferentes
  - Da mesma forma que uma receita pode ser escrita de várias formas

Algoritmo (exemplo)

- Como ensinar um computador a obter o quociente q da divisão a/b de dois números inteiros positivos a e b quando ele só sabe realizar instruções simples, como somar, subtrair e comparar números?

[PAUSA PARA VOCÊ PENSAR A RESPEITO]

Algoritmo (exemplo)

- Como ensinar um computador a obter o quociente q da divisão a/b de dois números inteiros positivos a e b quando ele só sabe realizar instruções simples, como somar, subtrair e comparar números?

Um exemplo de solução:

```
1  Obtenha os valores para a e b
2  Comece q com o valor 0: q = 0
3  Enquanto a >= b repita
4      Incremente o valor de q: q = q+1
5      Subtraia o valor b de a: a = a-b
6  Imprima o valor de q
7  Pare
```

Programa (exemplo)

Um programa que implementa o algoritmo do slide anterior:

```
1  Leia um numero e armazene-o em RegA
2  Leia um numero e armazene-o em RegB
3  Carregue o valor 0 em RegC
4  Se RegA < RegB desvie para 8
5  Incremente o valor em RegC: RegC = RegC+1
6  Subtraia o valor b de RegA: RegA = RegA-RegB
7  Desvie incondicionalmente para 4
8  Imprima o valor armazenado em RegC
9  Pare
```

Programa do quociente

Em Python (uma linguagem de alto nível):

```
1  a = int(input())
2  b = int(input())
3  q = 0
4  while a >= b:
5      q = q + 1
6      a = a - b
7  print(q)
```

"Lição de casa"

Descubra o que o seguinte programa faz

Pos. memória	Instrução
01	Carregue o RegA com [30]
02	Armazene [RegA] em 40
03	Leia um número e armazene-o em 45
04	Exiba numericamente [45]
05	Carregue o RegA com [45]
06	Se [RegA] < 0 desvie para 11
07	Carregue o RegA com [40]
08	Adicione ao RegA [45]
09	Armazene [RegA] em 40
10	Desvie incondicionalmente para 03
11	Imprima[40]
12	Pare

- [X] denota o valor armazenado na posição de memória X
- Considere que, no início da execução, [30] = 0

Programas e linguagens de programação

- Um programa escrito em uma linguagem de programação de alto nível não pode ser executado diretamente pelo computador
- Ele precisa ser *convertido* em um programa em uma **linguagem de baixo nível** (= **linguagem de máquina**) para que possa ser executado
- Essa conversão é feita por meio de dois tipos de programas: os **compiladores** e os **interpretadores**

Linguagem de alto nível → linguagem de máquina

Compilador

- O **compilador** lê o programa e o **traduz completamente** antes que o programa comece a ser executado
- O programa escrito em linguagem de alto nível é chamado de **código fonte**
- O programa traduzido é chamado de **código objeto** ou **executável**
- Uma vez que um programa é compilado, ele pode ser executado repetidamente, sem que uma nova tradução seja necessária
- Exemplo de linguagem de programação que gera programas que precisam ser compilados: **Linguagem C**

Linguagem de alto nível → linguagem de máquina

Interpretador

- O interpretador lê um programa escrito em linguagem de alto nível e o executa
- Ele processa o programa um pouco de cada vez, alternadamente: ora lendo uma(s) linha(s) do código fonte, ora realizando computações
- Exemplo de linguagem de programação que gera programas que precisam ser interpretados: **Python**

E já que falamos de código fonte, código objeto ...

... vale a pena também falar do **Sistema de Arquivos**

- Tanto o código fonte quanto o código objeto são armazenados no computador como **arquivos**
- Todas as informações que armazenamos nos dispositivos de memória não volátil são armazenadas na forma de arquivos
- Há 3 tipos básicos de arquivos:
  - **aplicativos** – podem ser executados pelo computador
  - **dados** – são usados como entrada e/ou saída para os aplicativos
  - **diretórios** (ou **pastas**) – são arquivos que contêm outros arquivos; são usados para organização

Cenas dos próximos capítulos...

Na próxima aula veremos:

- O "esqueleto" de um programa em Python
- Comandos de entrada, saída, atribuição e repetição
- Introdução ao uso de ambientes de desenvolvimento

Veja também:

- Uma breve introdução à História da Computação:
  `https://www.ime.usp.br/~macmulti/historico/`

Bibliografia

- "Capítulo 1 – Como Funciona um Computador" da apostila "Introdução à Ciência da Computação Usando a Linguagem C"
  `http://www.ime.usp.br/~hitoshi/introducao/`
- Livro *Structured Computer Organization*, de A. S. Tanenbaum

Entre em contato

- E-mail: paulormm@ime.usp.br
- Sala: 228, CCSL (Centro de Competência em Software Livre), IME-USP, Bloco C