Organização de Computadores

Professor Marcus Vinícius Midena Ramos

Colegiado de Engenharia de Computação

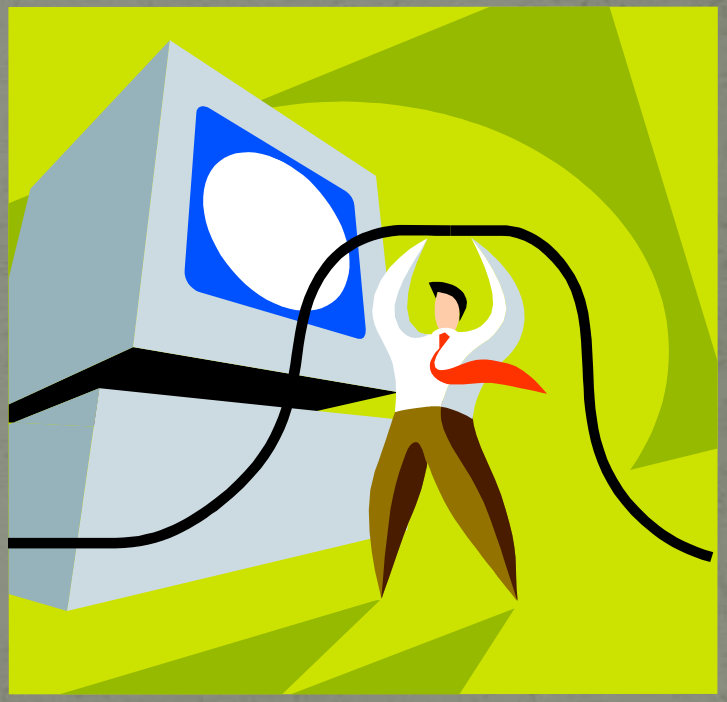
(74)3614.1936

marcus.ramos@univasf.edu.br

www.univasf.edu.br/~marcus.ramos

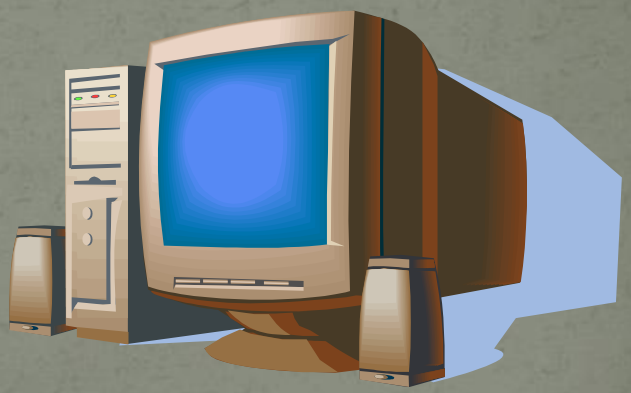
Computador

- Ferramenta indispensável;
- Faz parte das nossas vidas;
- Por si só não faz nada de útil;
- Grande capacidade de resolução de problemas;
- Necessita ser instruído.



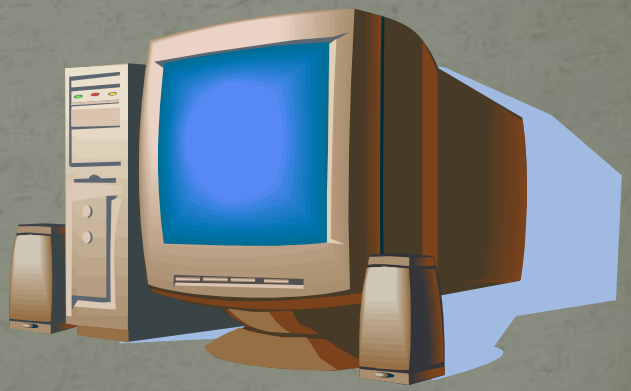
Computador

- Capaz apenas de excutar poucas tarefas básicas distintas, todas muito simples;
- É extremamente rápido;
- Possui um comportamento previsível;
- É excelente para reproduzir “roteiros” pré-concebidos;
- Não se cansa e pode ser usado à exaustão.



Computador = Hardware *(corpo)* + Software *(alma)*

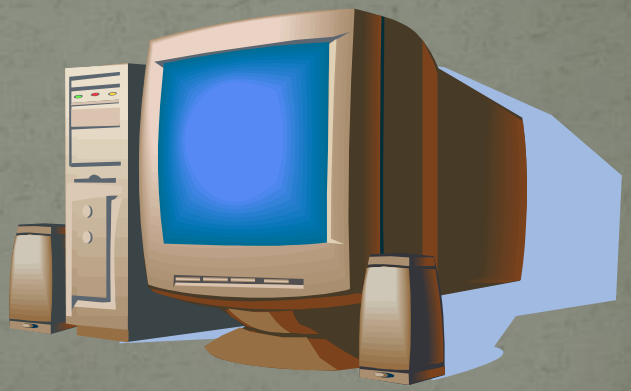
- ✓ O corpo fornece suporte para a alma.
- ✓ O corpo procura suprir as necessidades da alma.
- ✓ O corpo pode criar novas possibilidades para a alma, ou então estabeldcer limitações.
- ✓ A alma se expressa através do corpo.
- ✓ A alma usa os recursos do corpo.



Computador

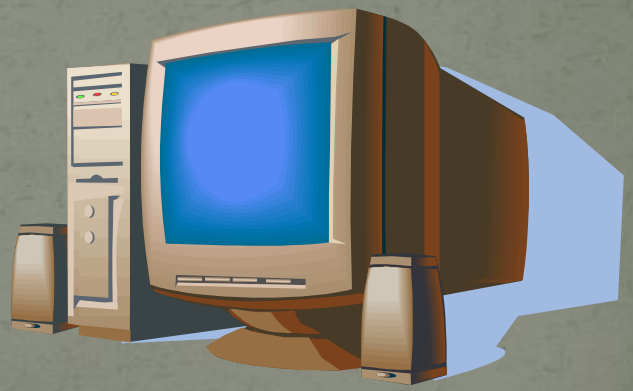
- Hardware:

  - ✓ Parte física: placas, periféricos, circuitos, cabos e componentes.
  - ✓ Quanto mais usado, mais propenso à falhas.
  - ✓ Sozinho, não serve para nada.
  - ✓ Vem pronto da fábrica.



Computador

- Software:
  - ✓ Parte intangível: conhecimentos e idéias que fazem o hardware exibir um certo comportamento.
  - ✓ Quanto mais usado, menos propenso à falhas.
  - ✓ Confere funcionalidade ao hardware.
  - ✓ Pode ser adquirido ou desenvolvido.

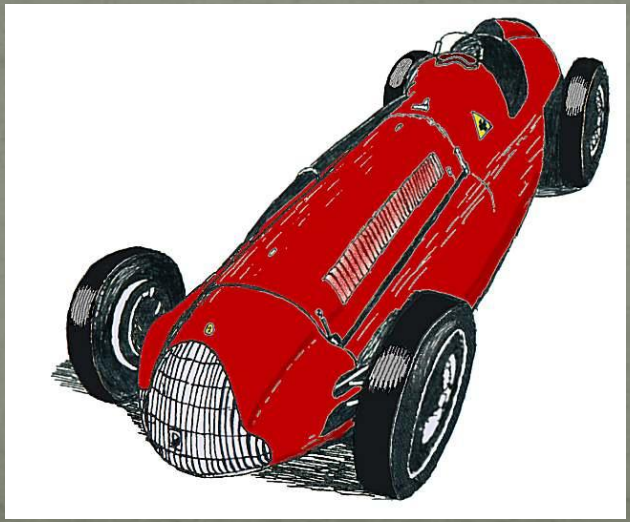


Modelo de von Neumann

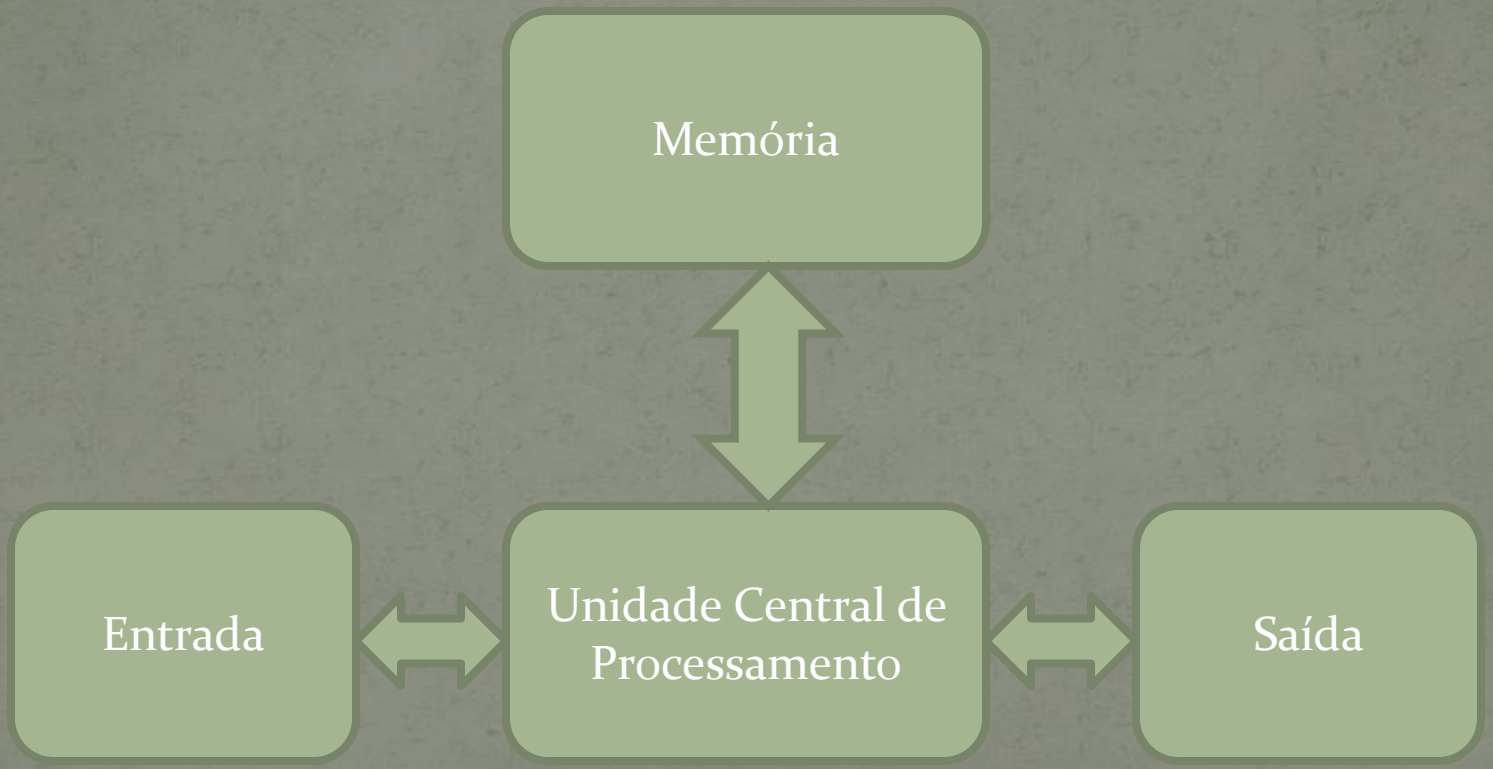
- Matemático húngaro;
- Propôs um modelo de arquitetura composto por quatro partes distintas;
- Nesse modelo, o programa a ser executado pelo computador ficava armazenado na memória do mesmo;
- Grande flexibilidade e rapidez, pois o hardware não precisa ser modificado e não precisava ser lido em cartões perfurados;
- A maioria dos computadores modernos adota essa arquitetura.

Modelo de von Neumann

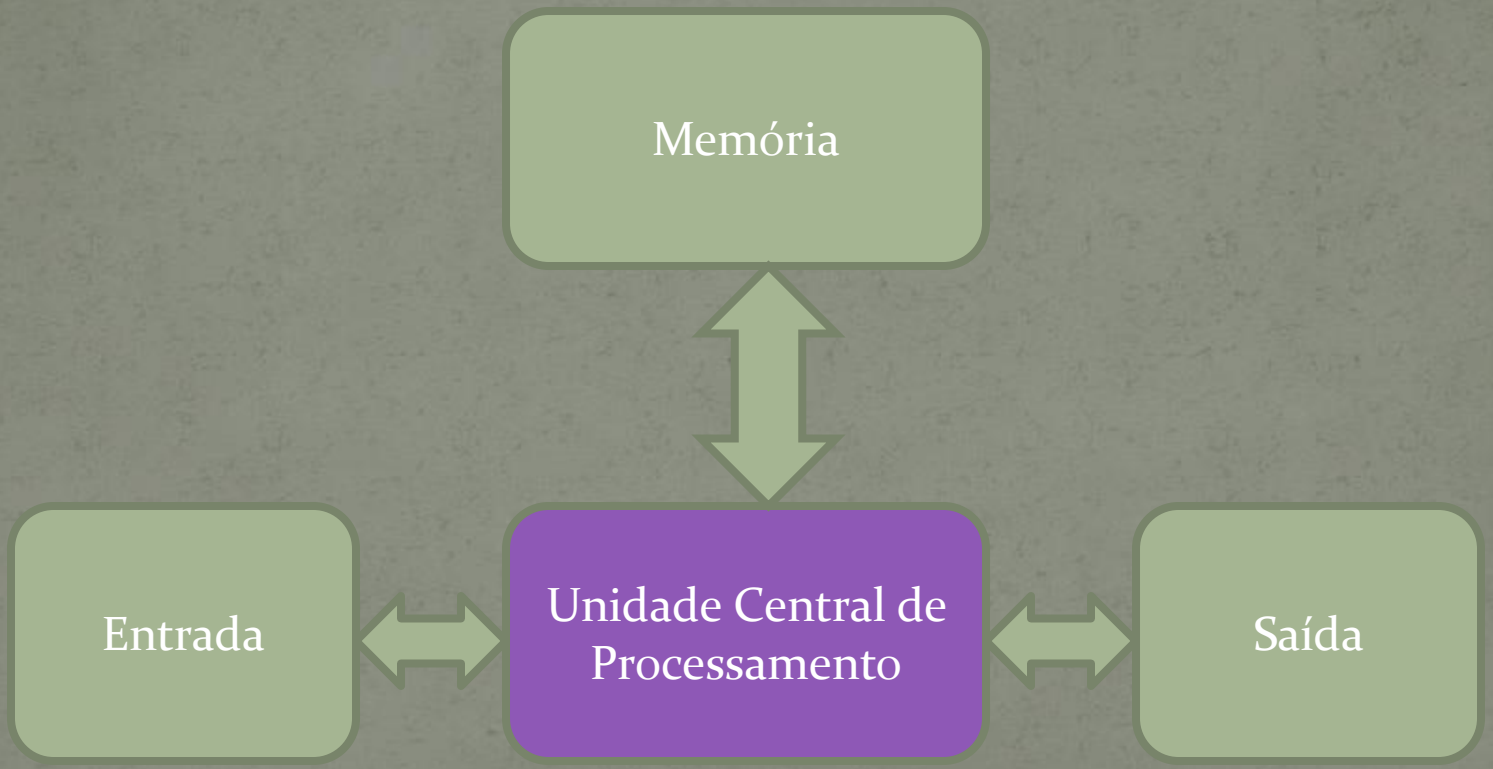
- Ciência x Tecnologia
- Evolução científica x tecnológica
- Exemplo do automóvel



Modelo de von Neumann



Modelo de von Neumann

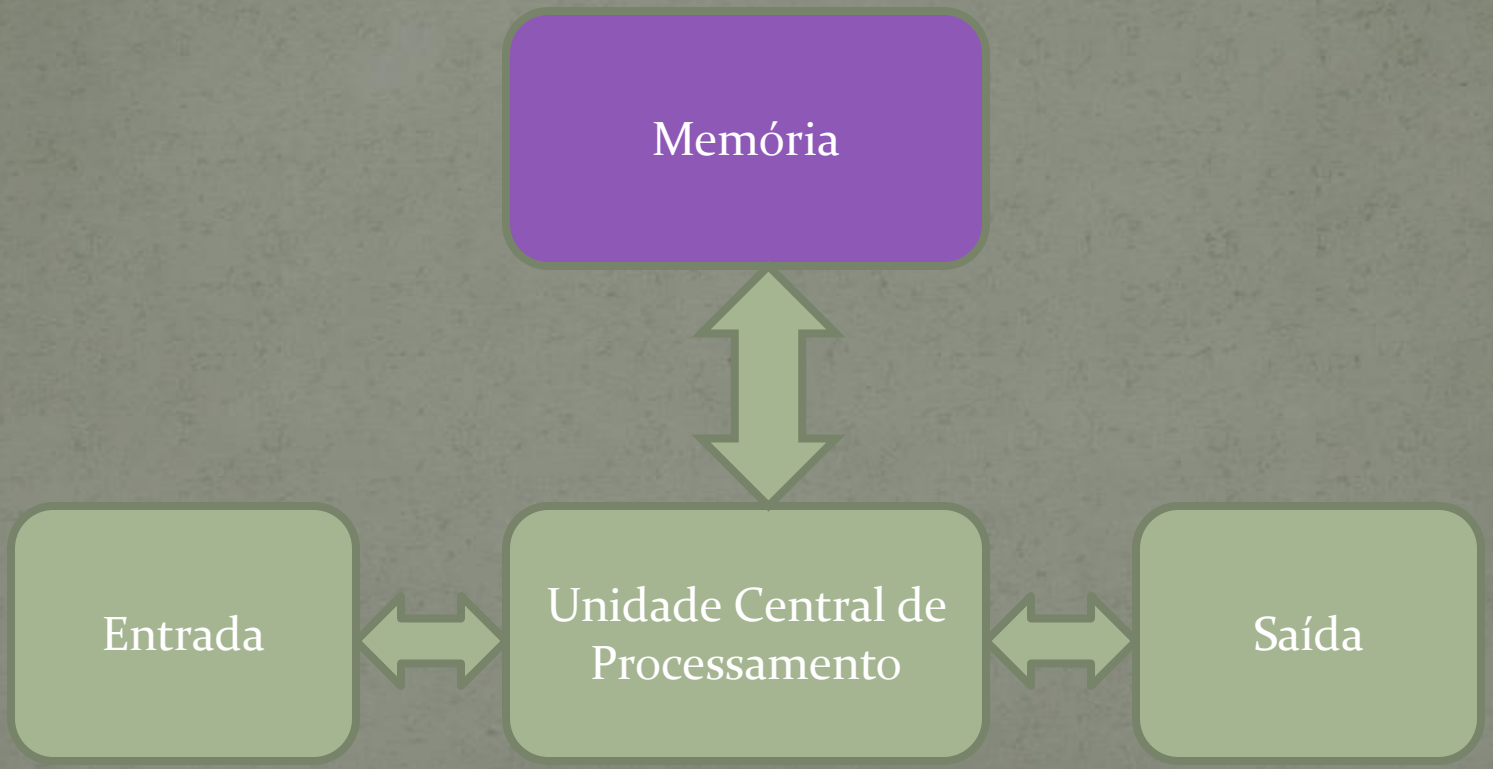


Modelo de von Neumann

Unidade Central de Processamento (UCP ou CPU):

- Lê, decodifica e executa as instruções armazenadas na memória;
- Realiza operações lógicas e aritméticas;
- Pode ler modificar o valor de alguma posição de memória;
- Pode transferir valores da entrada para a memória e da memória para a saída;
- Executa as instruções seqüencialmente, até o término do programa ou até encontrar algum desvio explícito.

Modelo de von Neumann

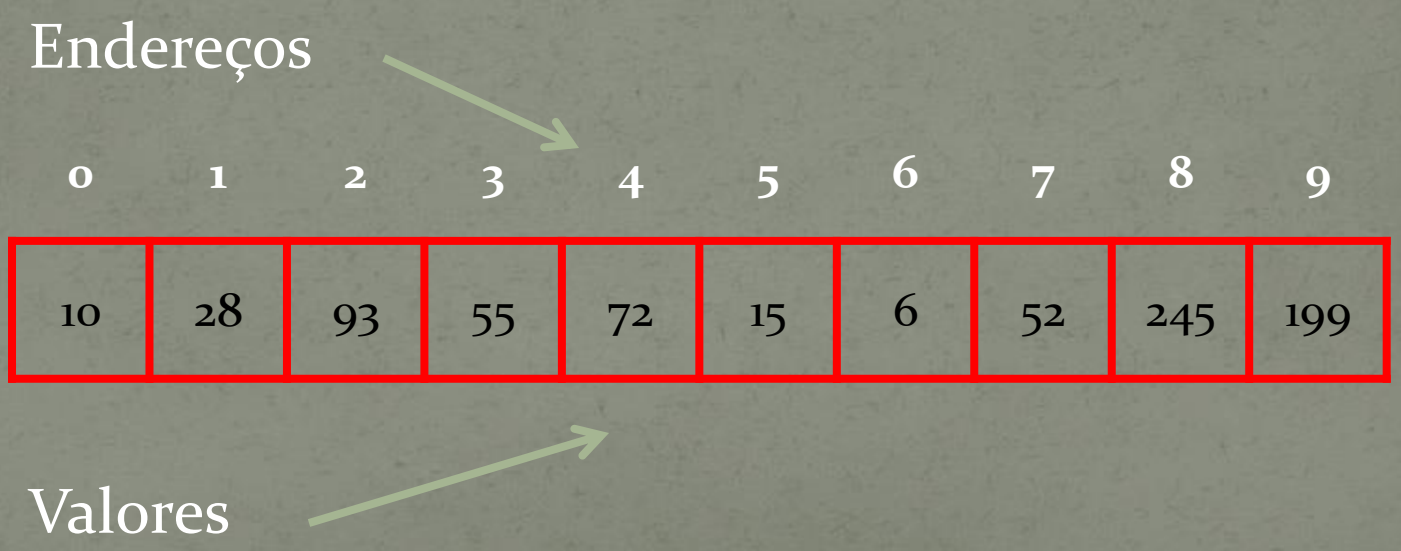


Modelo de von Neumann

Memória:

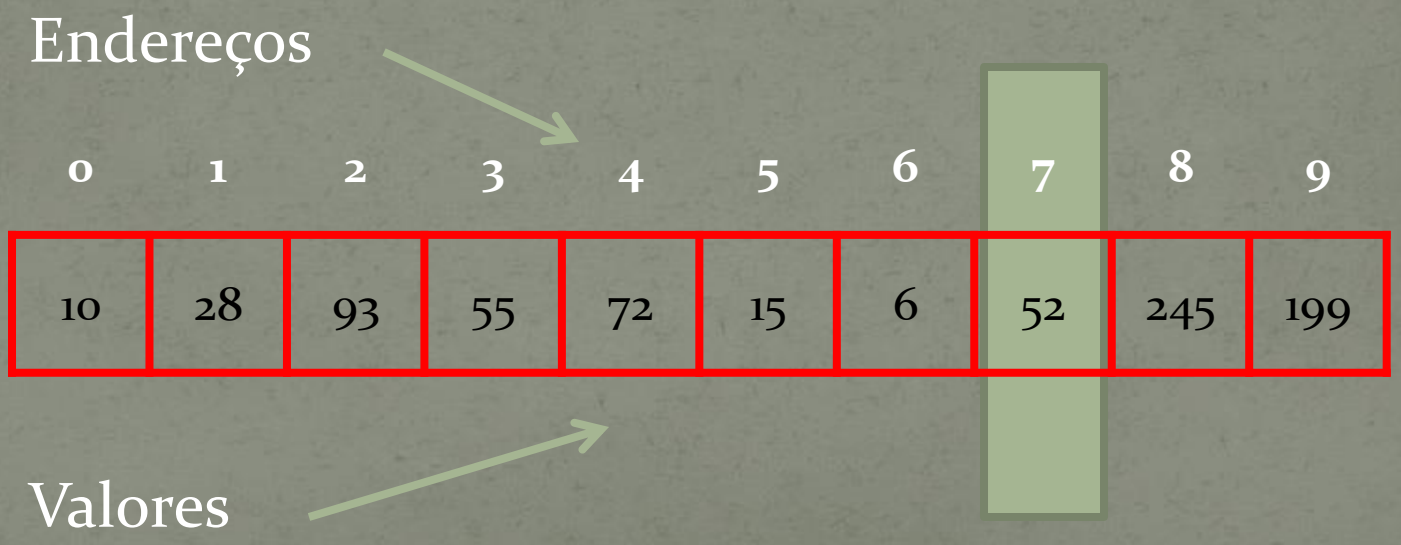
- É dividida em células (bytes)
- Cada célula armazena um valor e possui um endereço;
- É comandada pela CPU, que informa em qual endereço deseja executar uma operação (de leitura ou escrita);
- Operações de leitura não modificam os valores armazenados;
- Operações de escrita apagam os valores anteriores, substituindo-os pelos novos valores;

- Exemplo de memória com 10 posições;
- As posições são numeradas 0 a 9;
- Valores permanecem até que um novo valor seja armazenado na mesma posição ou até que a memória seja desligada.



Operação de leitura

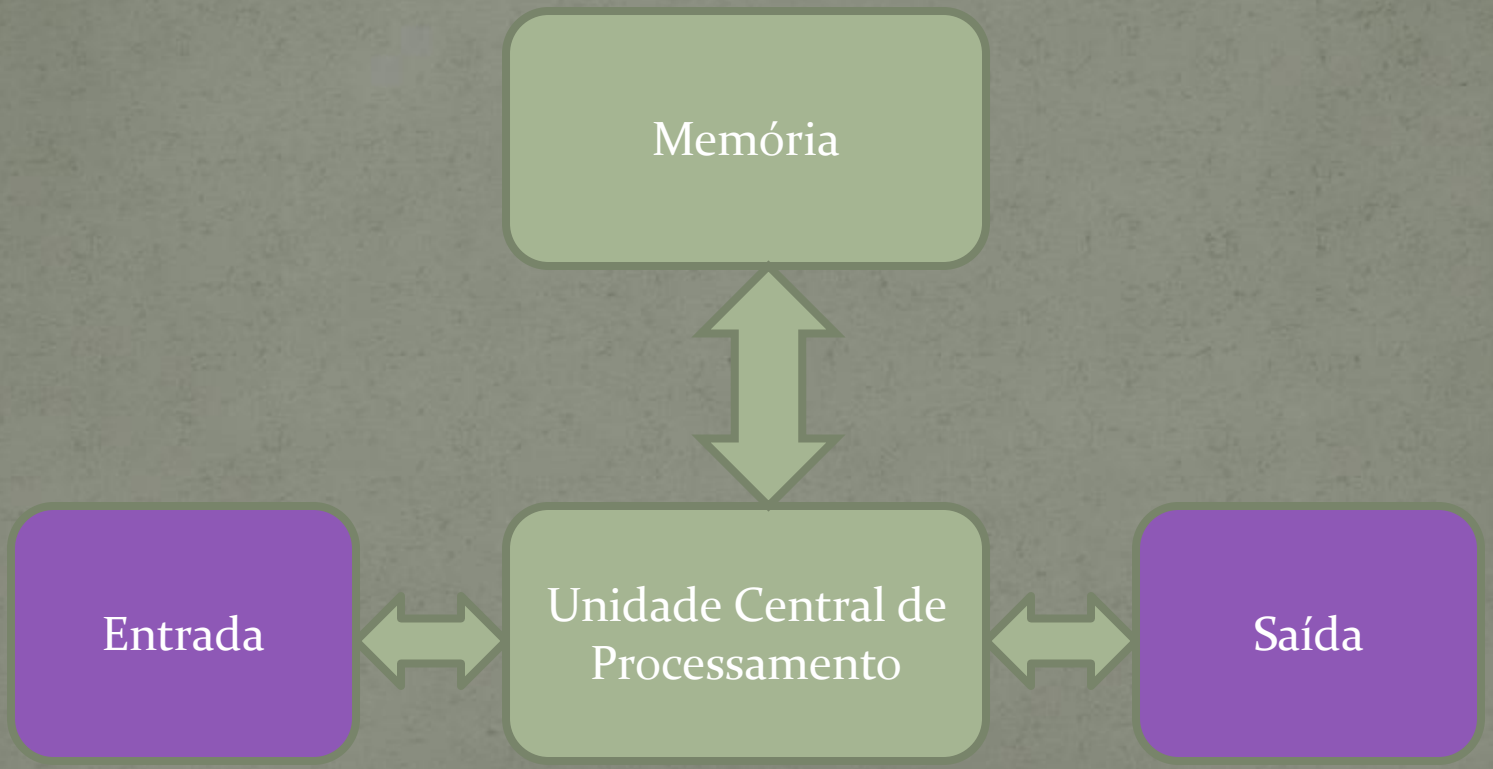
- Obter o valor armazenado na posição (endereço) 7
- Resultado: 52



Operação de escrita

- Armazenar o valor 31 na posição (endereço) 7

Modelo de von Neumann



Modelo de von Neumann

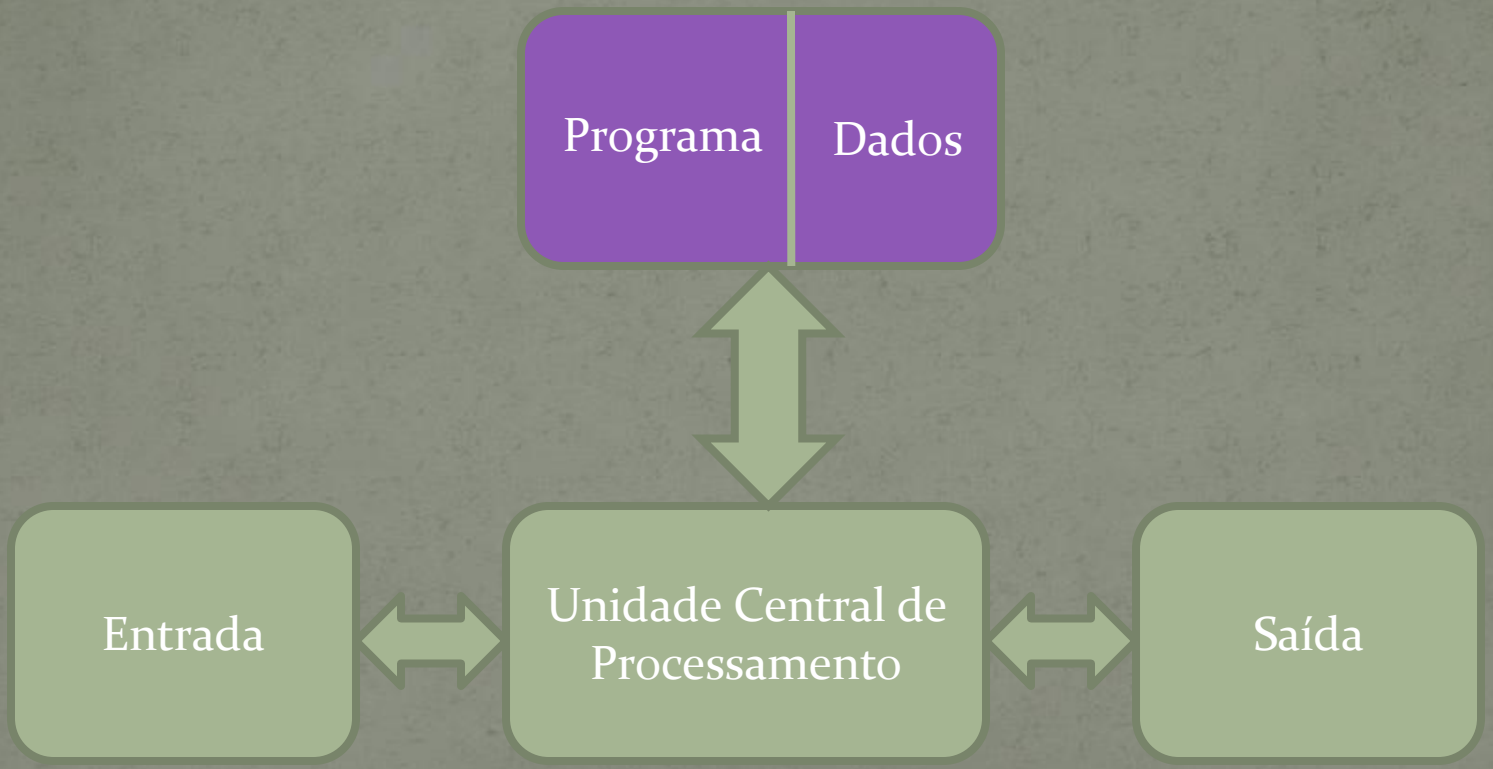
Entrada e Saída:

- Sob comando da CPU, é possível ler valores de um dispositivo de entrada, armazenando-os na memória;
- Sob comando da CPU, é possível ler valores da memória, enviando-os para um dispositivo de saída;
- Existe uma grande variedade de dispositivos de entrada e saída;
- Conceitualmente, no entanto, eles são passivos e obedecem ao comando da CPU.

Modelo de von Neumann

- Os dados e os programas são armazenados na memória, em regiões distintas;
- Um programa é composto por uma coleção de instruções que são lidas e executadas em seqüência.

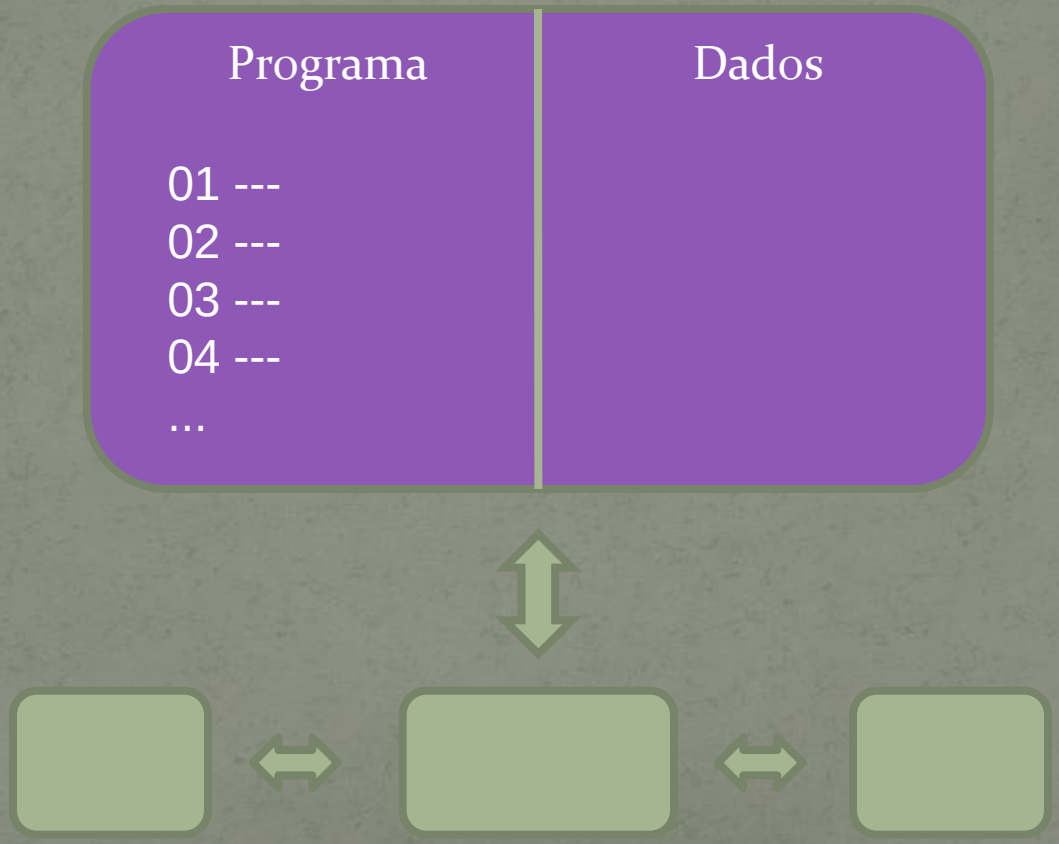
Modelo de von Neumann



Modelo de von Neumann

- Os programas são formados, essencialmente, por comandos (instruções sobre o que fazer);
- As instruções são muito simples e executadas muito rapidamente;

Modelo de von Neumann



Modelo de von Neumann

- As instruções são lidas seqüencialmente da memória, uma após a outra;
- Uma instrução pode (i) ler um valor da entrada; (ii) enviar um valor para a saída; (iii) gerar um novo valor; (iv) indicar o endereço da próxima instrução a ser executada.

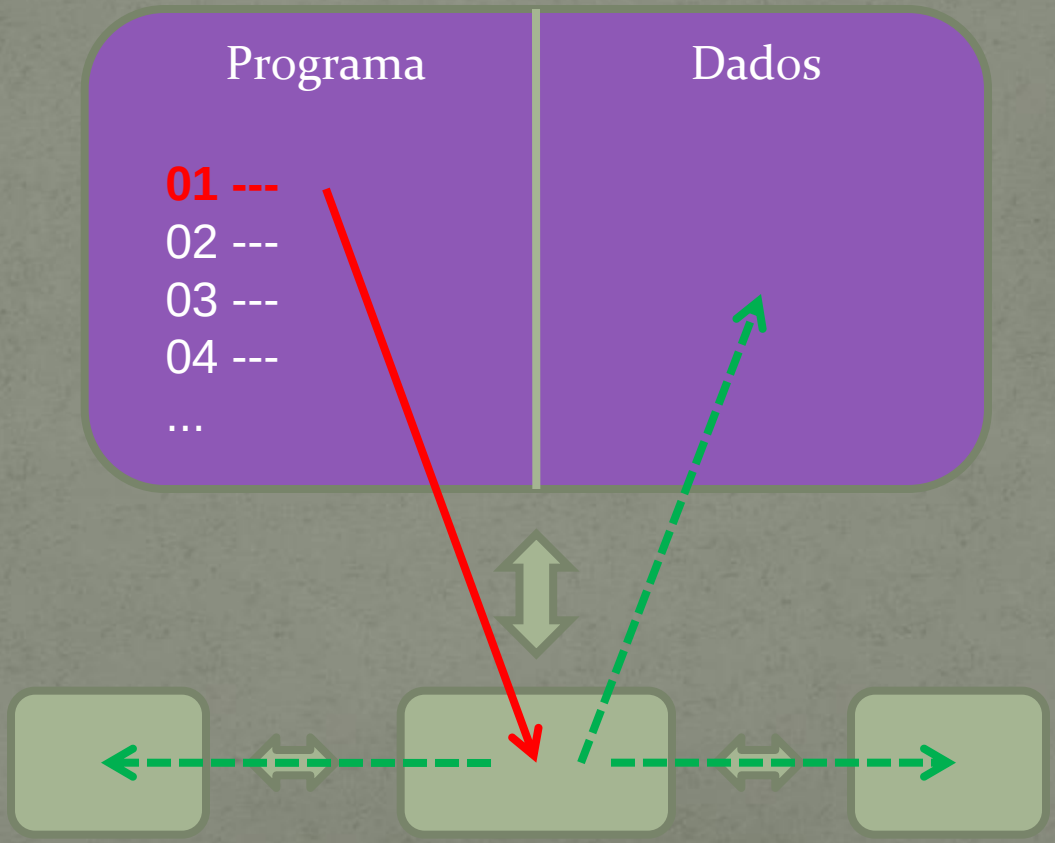
Modelo de von Neumann

- A execução de um novo comando inicia apenas depois que a execução do anterior tiver terminado (execução seqüencial);
- Se não houver nenhuma indicação explícita, a instrução armazenada no endereço de memória seguinte é executada.

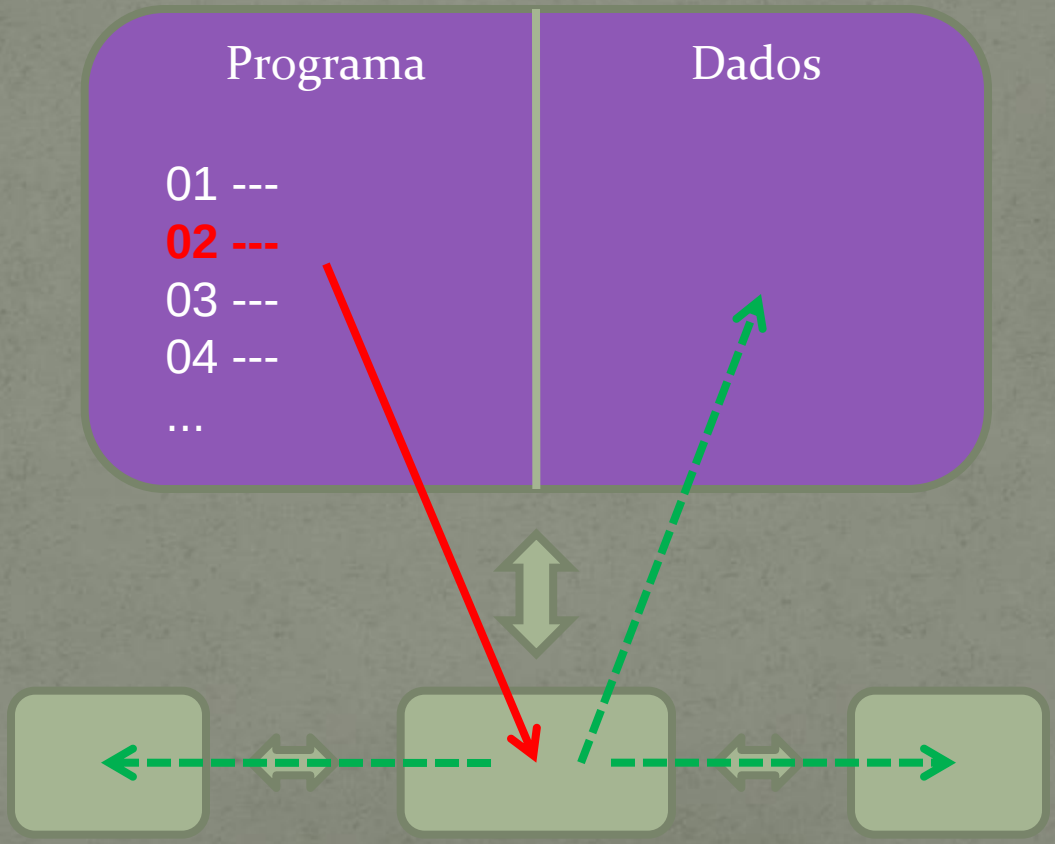
Modelo de von Neumann

- Eventualmente, um comando pode modificar o valor de um dado existente na memória, solicitar novos dados ao usuário ou enviar dados para a saída.

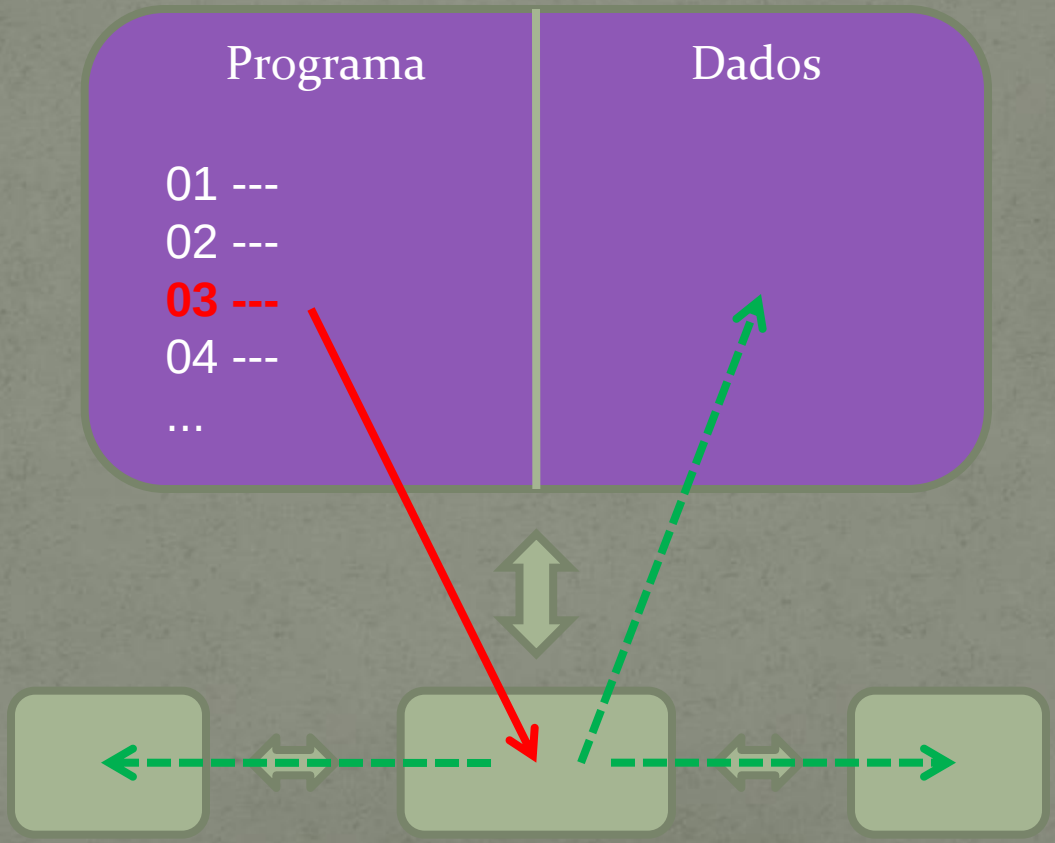
Modelo de von Neumann



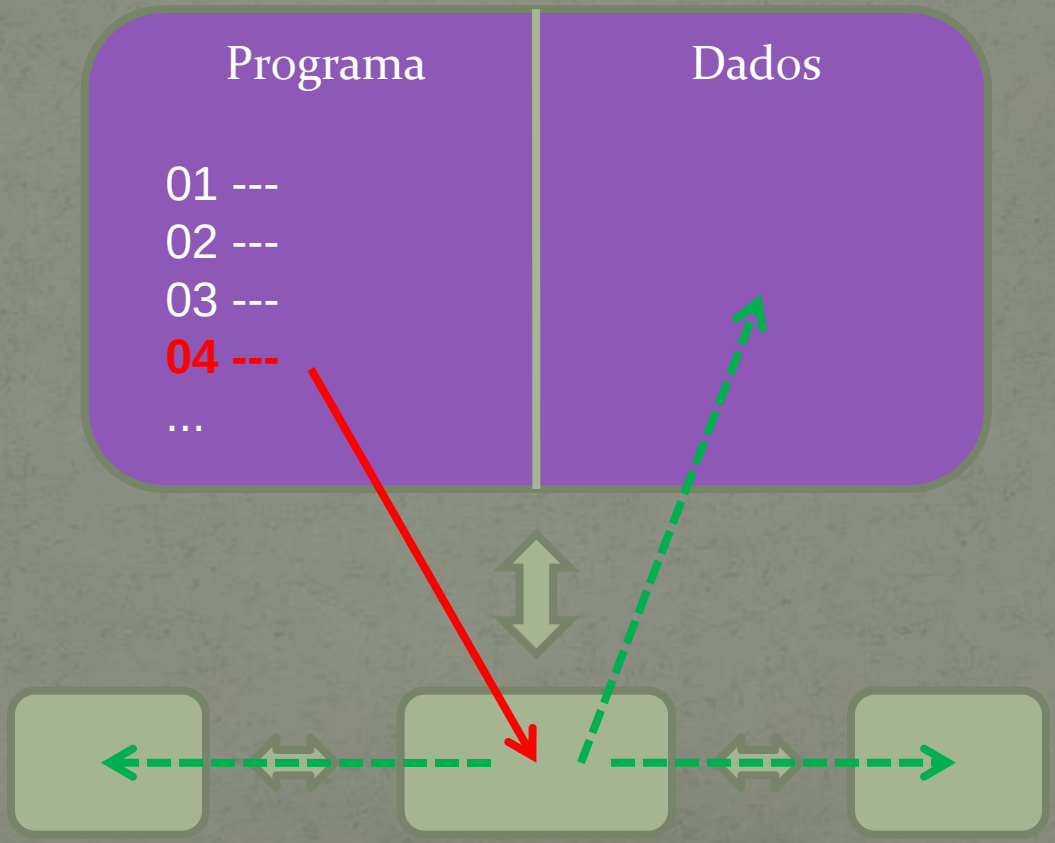
Modelo de von Neumann



Modelo de von Neumann



Modelo de von Neumann



Modelo de von Neumann

Exemplos de evolução tecnológica:

- Velocidade da CPU;
- Quantidade de registradores;
- Quantidade de bits maniupulados simultaneamente;
- Quantidade de posições de memória;
- Velocidade de acesso à memória;
- Tamanho do computador;
- Consumo de energia;
- Preço do computador;
- Possibilidade de expansão do computador;
- Diversificação dos dispositivos de entrada e saída.

Unidades

- Sistema binário
- 1 bit = 0 ou 1 (menor unidade de informação)
- 1 byte = 8 bits
- 1 KiloByte = 1KB = 1.024 bytes
- $1.024 = 2^{10}$
- 1 MegaByte = 1MB = 1.024 KiloBytes = 1.024*1.024 bytes ~ 1 milhão de bytes
- 1 GigaByte = 1 GB = 1.024 MegaBytes = 1.024*1.024*1.024 bytes ~ 1 bilhão de bytes
- Bits, Bytes, Herz, ...
- Kilo, Mega, Giga, Tera...

Sistemas de numeração

- Sistema decimal:
  xyz =
  $x*100 + y*10 + z*1 =$
  $x*10^2 + x*10^1 + x*10^0$

Sistema binário:
  xyz =
  $x*4 + y*2 + z*1 =$
  $x*2^2 + x*2^1 + x*2^0$

Sistemas de numeração

- Sistema decimal:
  xyz =
  $x*100 + y*10 + z*1 =$
  $x*10^2 + x*10^1 + x*10^0$

Sistema binário:
  xyz =
  $x*4 + y*2 + z*1 =$
  $x*2^2 + x*2^1 + x*2^0$

Sistemas de numeração

- Sistema decimal:

  518_{10} =

  $5*100 + 1*10 + 8*1$ =

  $5*10^2 + 1*10^1 + 8*10^0$

Sistema binário:

101_2 =

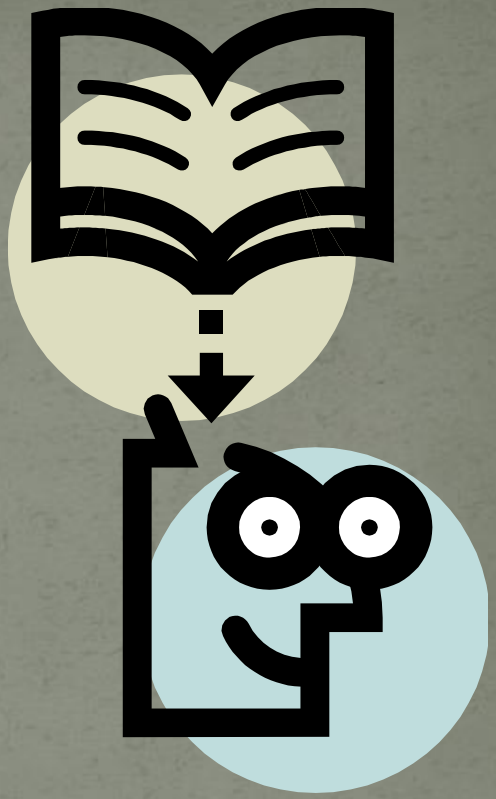
$1*4 + 0*2 + 1*1$ =

$1*2^2 + 0*2^1 + 1*2^0 = 5_{10}$

Objetivos:

- Entender a natureza do software e a sua relação com o hardware;
- Conhecer o processo básico de desenvolvimento de software;
- Entender a diferença entre linguagens de baixo e alto nível;
- Conhecer superficialmente as principais ferramentas de processamento de linguagens.

Nosso objetivo



- Desenvolver software

- ✓ Organização de idéias;
- ✓ Modelo de funcionamento do computador;
- ✓ Conceitos básicos de programação;
- ✓ Transcrição para linguagens apropriadas;
- ✓ Comunicação e interação com o computador;
- ✓ Obtenção dos resultados pretendidos;
- ✓ Prática em laboratório.

Desenvolver software

- Roteiro

1. Problema;
2. Solução;
3. Algoritmo;
4. Programa;
5. Resultados.

Problema

- ✓ Precisa ser conhecido em todos os seus aspectos;
- ✓ É necessário ter resposta para todas as perguntas que dele possam suscitar;
- ✓ É fundamental considerar todas as situações adversas;
- ✓ Nenhuma faceta deve ser omitida.

Solução

- ✓ Existe solução para o problema?
- ✓ Qual o custo da sua implementação?
- ✓ Qual o custo da sua execução?
- ✓ Como iremos representá-la?

Algoritmo

- ✓ Representação de uma solução para um problema, com algumas características:
  - Seqüência finita de etapas;
  - Individualmente, existe realização possível para cada uma das etapas consideradas;
  - Termina após um tempo finito.

Algoritmo

- ✓ Representação :
  - Linguagem natural;
  - Pseudocódigo (linguagem textual com poucos símbolos e regras, que são simples);
  - Fluxograma (linguagem visual composta por poucos símbolos e regras)
- ✓ Um algoritmo expressa uma solução para um problema.

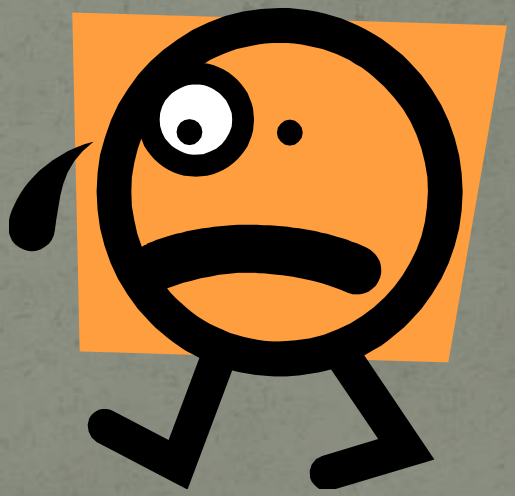
Terminou?

Não!!!

Acontece que...

Computadores não entendem (normalmente, ou pelo menos da forma como nós precisamos):

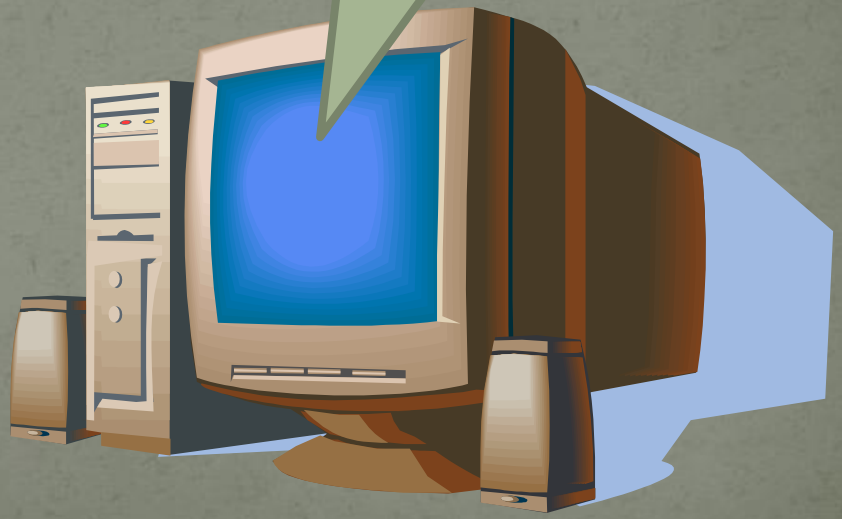
- Linguagens naturais;
- Pseudocódigos;
- Fluxogramas.



Precisamos ir além...

O que o computador entende afinal?

Faça isso. Depois aquilo. Se OK, então pare, senão refaça tudo.



01101010110010101101010100101010001011111101010010000101111110101011001111111?????

Temos um problema de comunicação.

Solução?

Melhorar um pouco as coisas prá ele (computador) sem piorar tanto prá nós (humanos).

- ⇒ Escrever um “programa” de computador, a partir do algoritmo.
- ⇒ Para isso, vamos usar uma “linguagem de programação”.
- ⇒ Um pouco mais complexas do que as linguagens usadas para representar algoritmos;
- ⇒ Mas mais fáceis de serem entendidas pelo computador.

Java?
C?
C++?
Delphi?
Pascal?
HTML?
Perl?
Python?
Ruby?
Fortran?
Assembly?
PHP?
Cobol?
SQL?
Lisp?
Prolog?

Java!!
C!!
C++!!
Delphi!!
Pascal!!
HTML!!
Perl!!
Python!!
Ruby!!
Fortran!!
Assembly!!
PHP!!
Cobol!!
SQL!!
Lisp!!
Prolog!!

Programação de baixo nível

- Utiliza linguagem de máquina (numérica) ou de montagem (com nomes, geralmente mnemônicos) atribuídos às instruções;
- Principais características:
  - Difícil legibilidade;
  - Dependente da arquitetura do computador (baixa portabilidade);
  - Difícil manipulação (números, mnemônicos, endereços...);
  - Baixa confiabilidade;
  - Baixa produtividade do programador;
  - Apenas especialistas.

Exemplo de programa

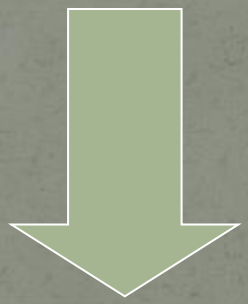
Linguagem de montagem		Linguagem de máquina	
INPUT	**25**	**71**	**25**
INPUT	26	71	26
LOAD	25	161	25
ADD	26	50	26
ADDI	01	44	01
STORE	27	160	27
OUTPUT	27	72	27
HALT		00	

Programação de alto nível

- Utiliza linguagem cuja sintaxe é mais próxima da linguagem natural;
- Principais características:
  - Alta legibilidade;
  - Independente da arquitetura do computador (alta portabilidade);
  - Fácil manipulação (uso de abstrações);
  - Maior confiabilidade;
  - Maior produtividade do programador;
  - Uso por não-especialistas

```
10 REM RESOLVE EQUACAO DO SEGUNDO GRAU
20 READ A,B,C
25 IF A=0 THEN GOTO 410
30 LET D=B*B-4*A*C
40 IF D<0 THEN GOTO 430
50 PRINT "SOLUCAO"
60 IF D=0 THEN GOTO 100
70 PRINT "PRIMEIRA SOLUCAO",(-B+SQR(D))/(2*A)
80 PRINT "SEGUNDA SOLUCAO",(-B-SQR(D))/(2*A)
90 GOTO 20
100 PRINT "SOLUCAO UNICA",(-B)/(2*A)
200 GOTO 20
410 PRINT "A DEVE SER DIFERENTE DE ZERO"
420 GOTO 20
430 PRINT "NAO HA SOLUCOES REAIS"
440 GOTO 20
490 DATA 10,20,1241,123,22,-1
500 END
```

Linguagem de alto nível: confere produtividade ao programador e legibilidade, confiabilidade e portabilidade aos programas desenvolvidos.



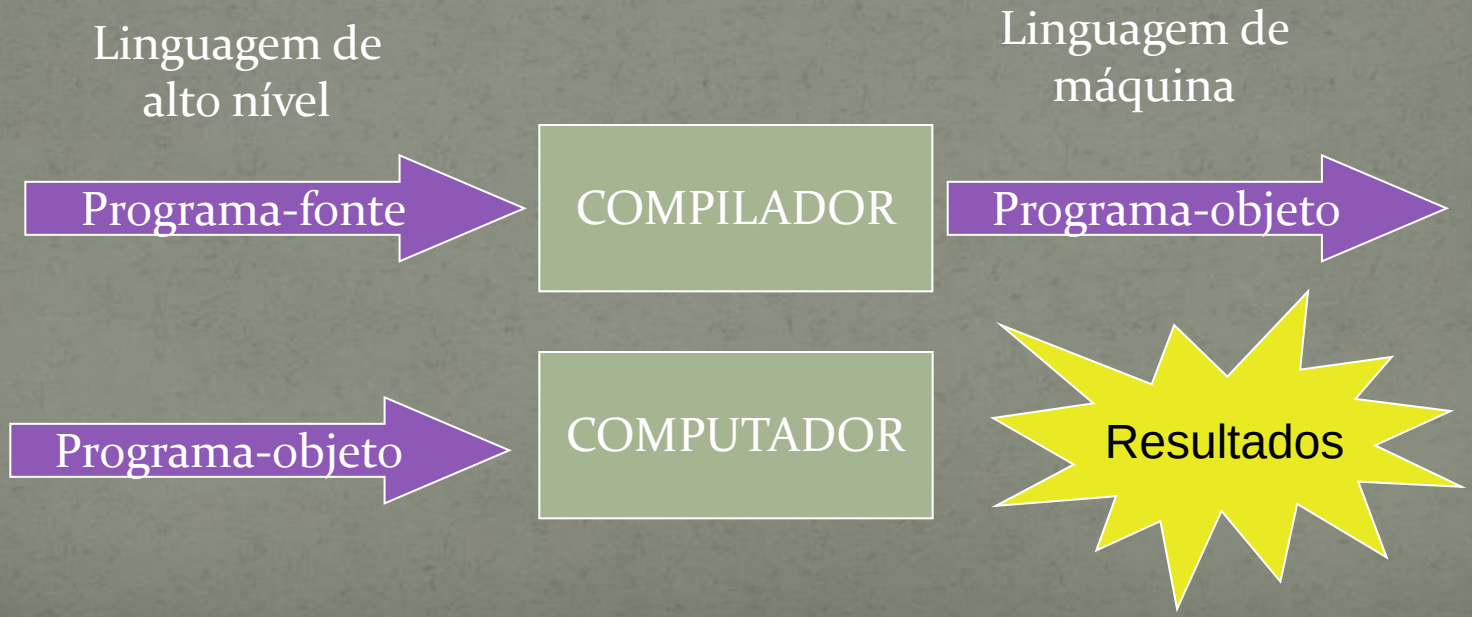
Tradução

Linguagem de máquina: a linguagem que o computador entende diretamente

Linguagem de montagem: usa mnemônicos para facilitar um pouco o trabalho do programador, mas tem as mesmas características fundamentais da linguagem de montagem

Compilação

O programa escrito na linguagem de alto nível é convertido para um programa equivalente (que desempenha a mesma função), porém escrito na linguagem da máquina na qual ele vai ser executado.



Compilação

Desvantagens:

- Processo burocrático (o programa precisa primeiro ser compilado para depois ser executado);
- Baixo nível de interação com o usuário;
- Dificuldade de depuração;
- Dependência do hardware / sistema operacional.

Vantagens:

- Gera programas mais rápidos e menores;
- Não precisa que o interpretador esteja residente na memória;

Linguagens compiladas

- C / C++
- Delphi
- Pascal

Linguagens interpretadas

- HTML
- BASIC
- JavaScript

Linguagens híbridas

- Java

Interpretação

O programa escrito na linguagem de alto nível é executado diretamente, gerando os resultados para o usuário imediatamente. Nenhum programa equivalente em linguagem de máquina é gerado.

Interpretação

Desvantagens:

- Execução lenta;
- Precisa que o interpretador esteja residente na memória;

Vantagens:

- Processo direto (o programa é executado diretamente);
- Alto nível de interação com o usuário;
- Facilidade de depuração;
- Independência do hardware / sistema operacional.

E…?

Sim, vamos precisar traduzir algoritmos para programas.

Sim, precisaremos conhecer (pelo menos)duas linguagens.

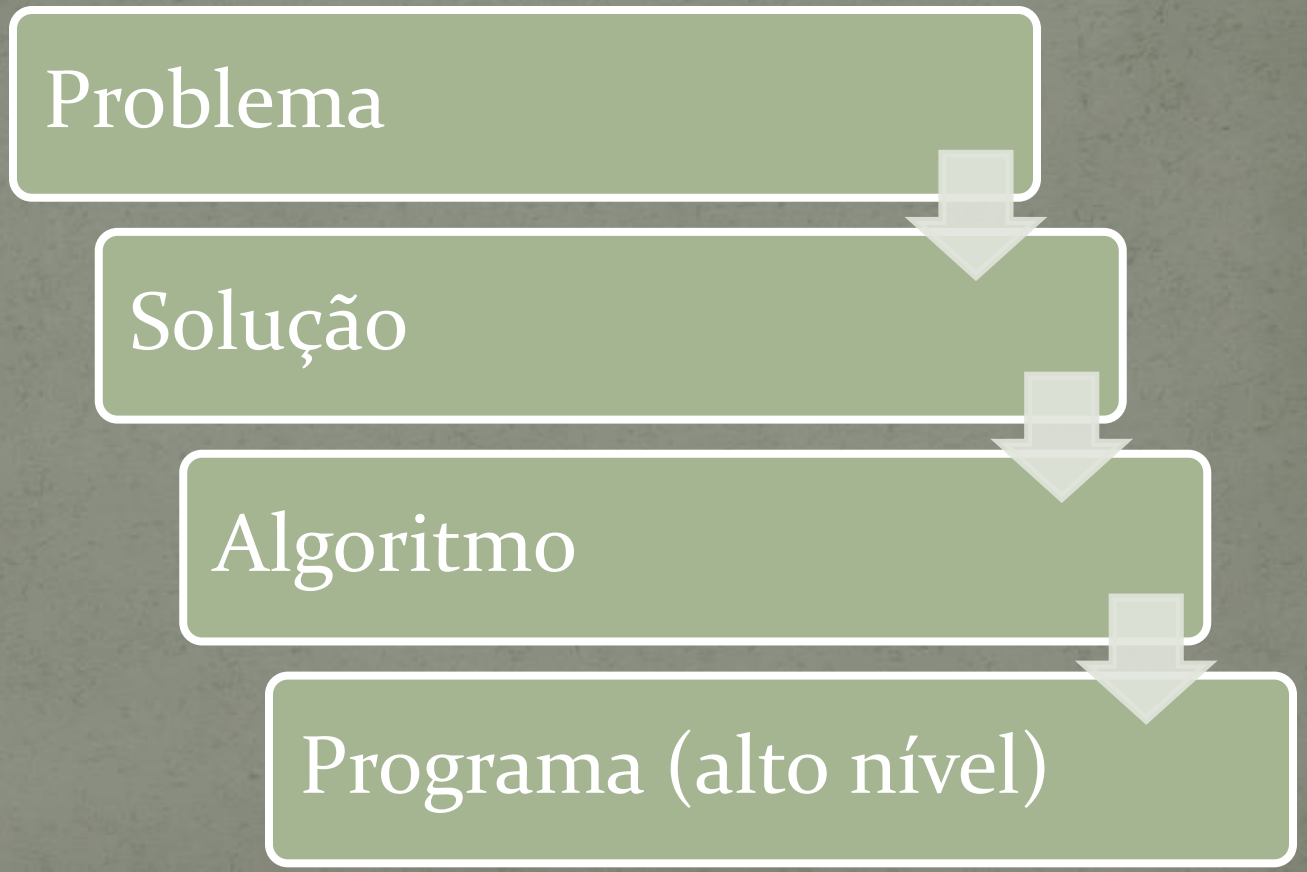
Sim, cometeremos erros nas traduções.

C´est la vie…

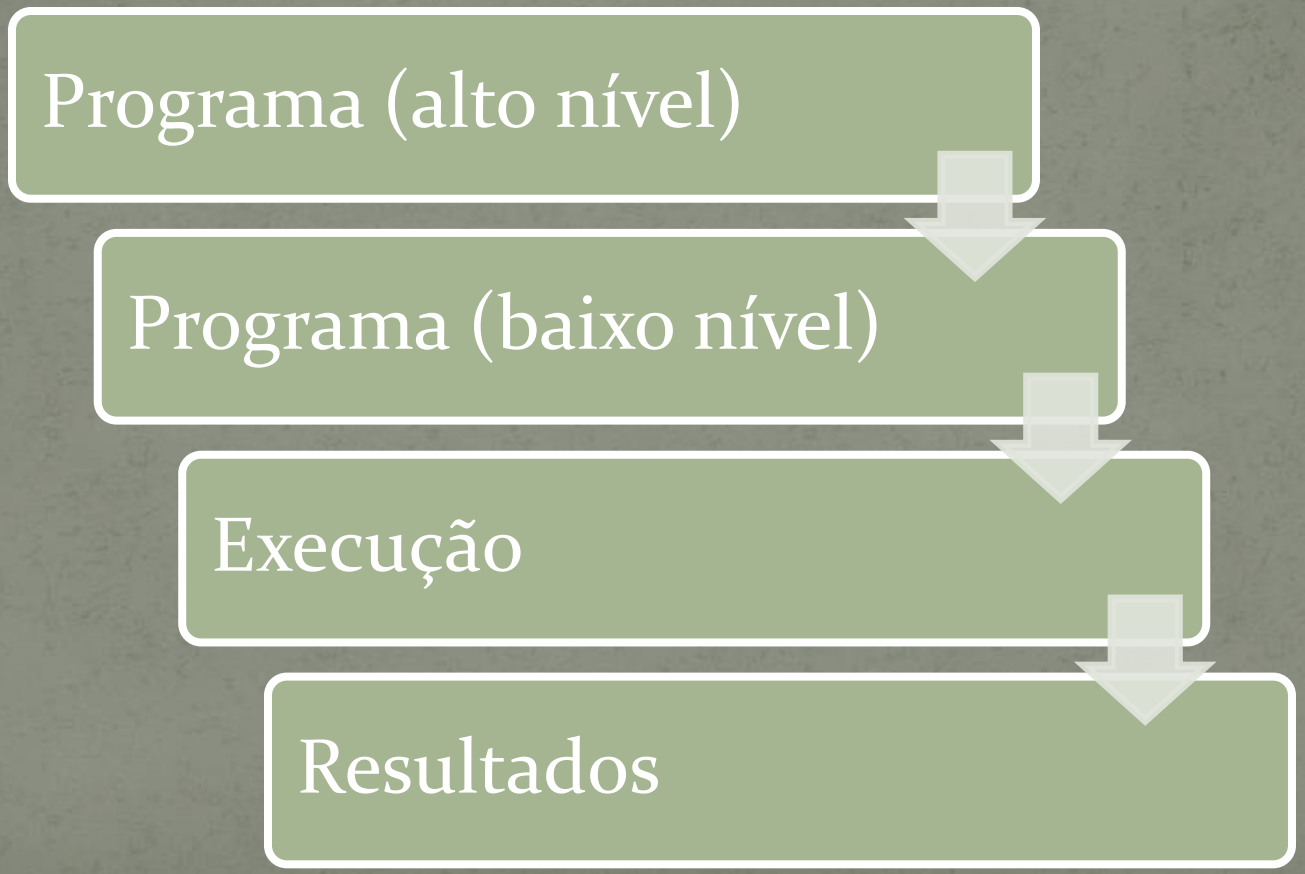
Eu não existo.

Como ficamos então?

Sua parte:



Parte do computador (com a sua supervisão...):



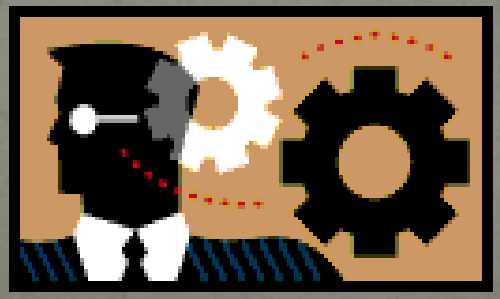
⇒Deu errado?

⇒Não era bem isso que você queria?

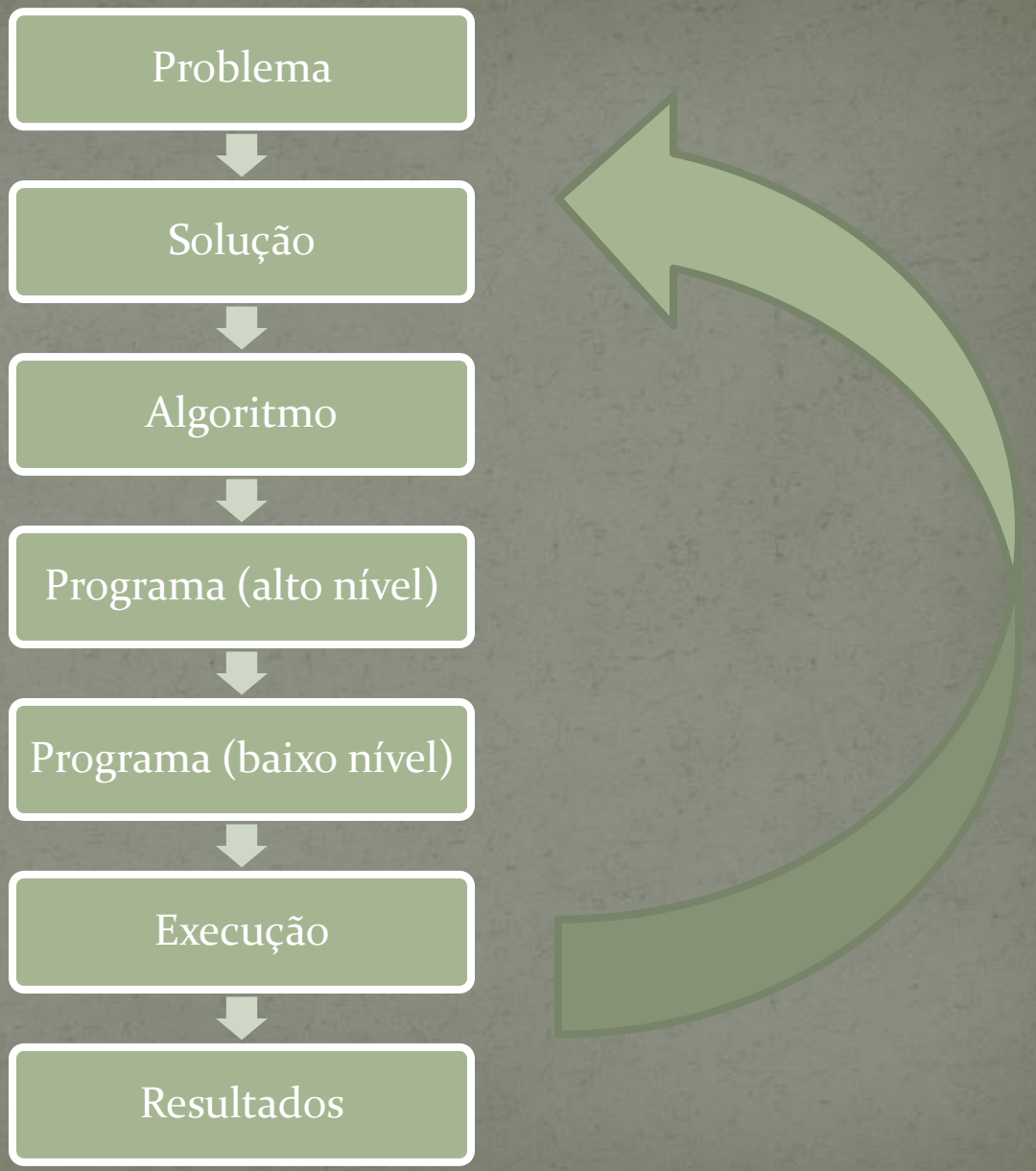
Não tem problema.

Volte à prancheta...

E descubra onde está o erro.



Ciclo de desenvolvimento

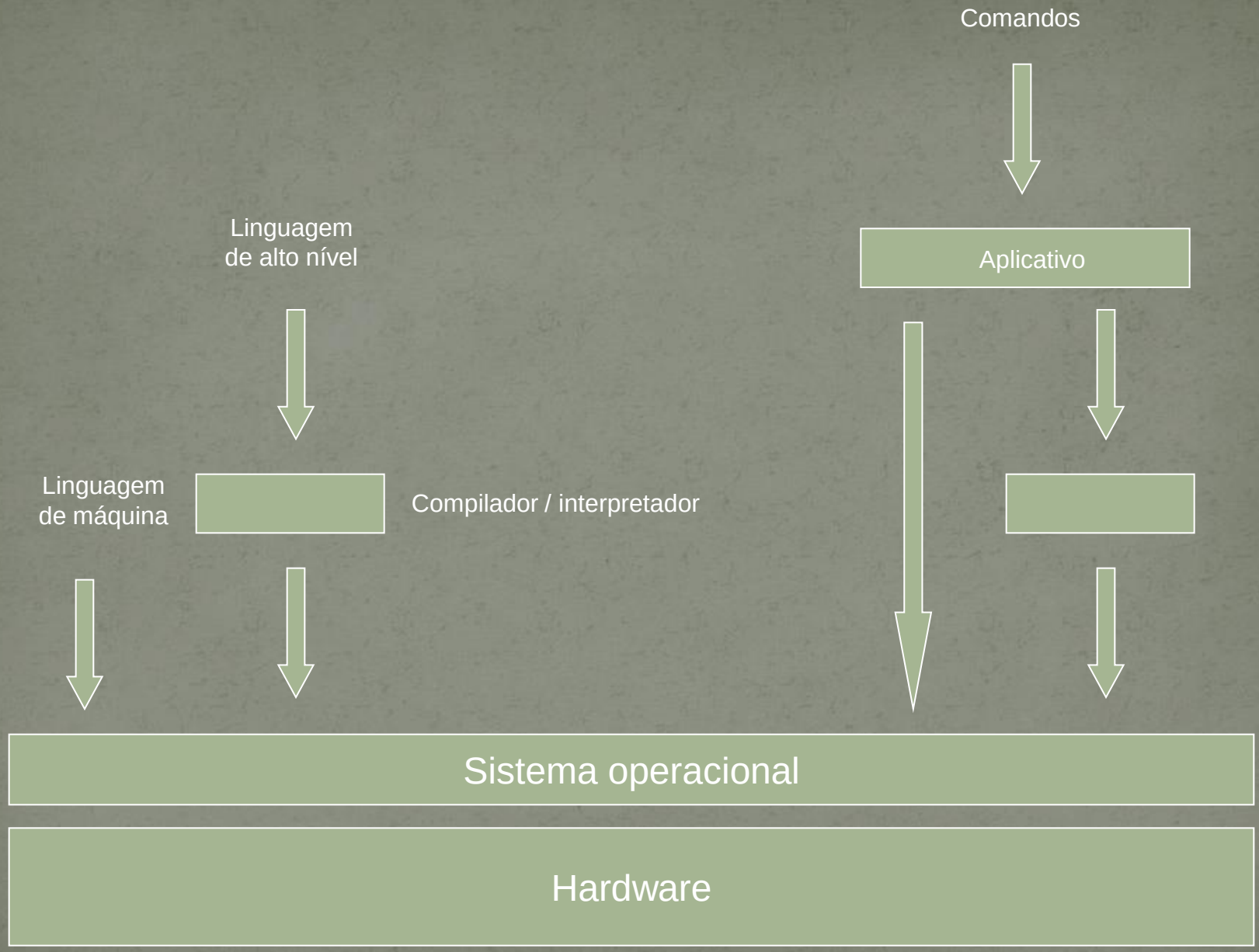


Desenvolvedor

x

Usuário

- Linguagens de programação, mesmo de alto nível, ainda não são acessíveis para a maioria das pessoas;
- Programas aplicativos permitem que usuários leigos usem o computador, com alto grau de sofisticação, sem a necessidade de conhecer qualquer linguagem de programação tradicional;
- Eles podem, ou não, gerar saídas que depois são compiladas e/ou interpretadas pelo computador;
- Apresentam grande facilidade de uso e são especializados em determinadas funções (edição de textos, cálculos numéricos etc).


Comandos
Linguagem
de alto nível
Aplicativo
Linguagem
de máquina
Compilador / interpretador
Sistema operacional
Hardware

Sistema operacional

Ambientes típicos de computação:

- Vários programas na mesma máquina
- Vários usuários na mesma máquina
- Vários tipos de dispositivo na mesma máquina

Necessidade de garantir:

- Integridade e privacidade dos dados
- Desempenho dos programas
- Inexistência de interferências indevidas
- Melhor utilização de todos os recursos da máquina

Desenvolvimento

Resultados desejáveis:

- Custos previsíveis
- Prazos respeitados
- Qualidade (funcionalidade e desempenho)

Desenvolvimento

Características do processo:

- Trabalho individual x trabalho em equipe;
- Poucas linhas de código x centenas de milhares de linhas de código;
- Aplicações críticas (risco de vida, valores elevados envolv idos);
- Hardware dedicado ou multiplataforma;
- Complexidade crescente das aplicações.

Programação

Necessidades:

- Recursos humanos altamente especializados
- Ferramentas adequadas
- Planejamento
- Gerenciamento
- Metodologia de desenvolvimento
- Testes
- Documentação
- Treinamento
- Atualização