Organização e Arquitetura de Computadores - OAC

Parte 2

"A experiência é resultado da prática".

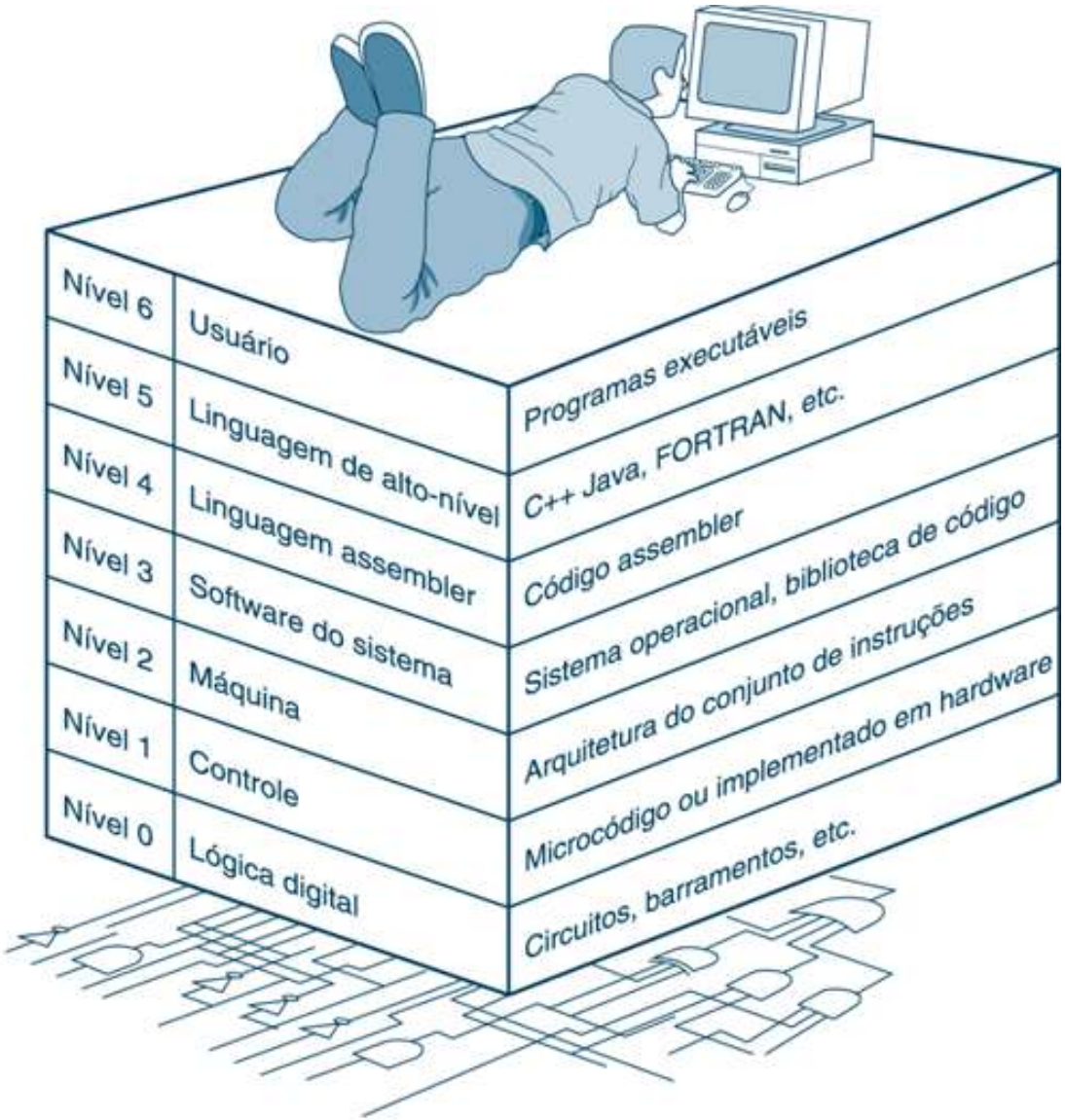
"A vontade de vencer é importante, mas a vontade de se preparar é vital".

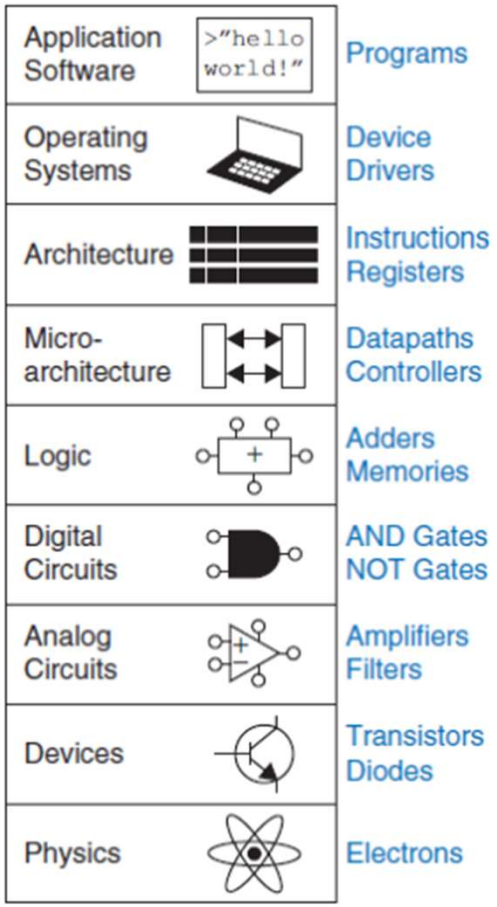
Prof. Misael Morais

moraiscg.uepb@gmail.com

https://sites.google.com/site/moraiscg/misael/moraiscg_oac

Níveis abstratos da computação





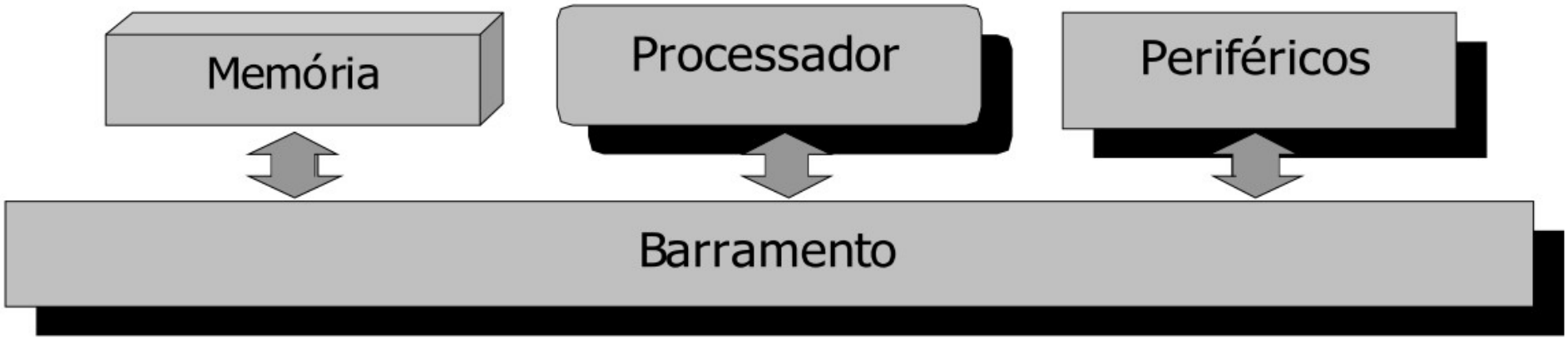
Além da abstração e da disciplina, utiliza-se para gerir a complexidade os treis princípios abaixo:

- A *hierarquia* implica a divisão de um sistema em módulos, em seguida, deve-se ainda subdividir cada um destes módulos até que as peças sejam fáceis de entender.

- A *modularidade* afirma que os módulos têm funções e interfaces bem definidas, para que se possam interligar facilmente sem imprevistos efeitos colaterais.

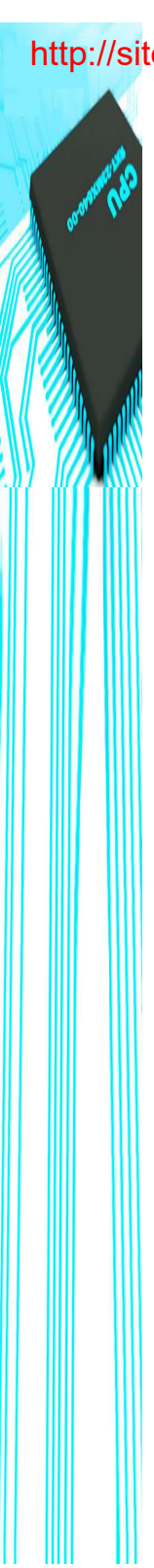
- *A regularidade* procura uniformidade entre os módulos. Os módulos mais comuns são reutilizados muitas vezes, reduzindo o número de módulos distintos que devem ser concebidos.

Componentes Básicos de um Computador



- O **processador** (ou microprocessador) é responsável pelo tratamento de informações armazenadas em memória (programas em código de máquina e dos dados).
- A **memória** é responsável pela armazenagem dos programas e dos dados.
- **Periféricos**, que são os dispositivos responsáveis pelas entradas e saídas de dados do computador, ou seja, pelas interações entre o computador e o mundo externo. Exemplos de periféricos são o monitor, teclados, *mouses*, impressoras, etc.
- **Barramento**, que liga todos estes componentes e é uma via de comunicação de alto desempenho por onde circulam os dados tratados pelo computador

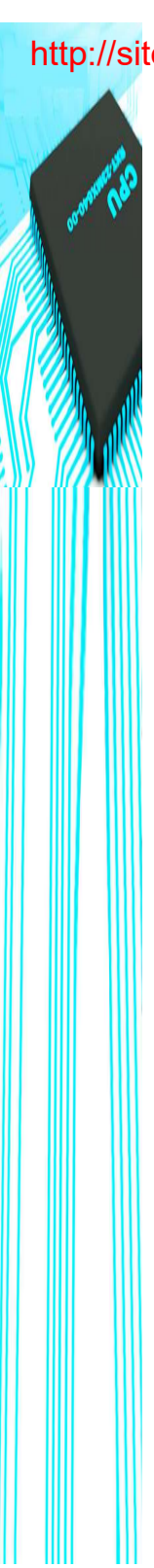
Prof. Misael Morais (moraiscg.uepb@gmail.com)



Processador

- A CPU é composta basicamente de três elementos: unidade de controle, unidade lógica e aritmética e registradores.
- **Unidade Lógica e Aritmética** (ALU) - Assume todas as tarefas relacionadas às operações lógicas (ou, e, negação, etc.) e aritméticas (adições, subtrações, etc...) a serem realizadas no contexto de uma tarefa.

Prof. Misael Morais (moraiscg.uepb@gmail.com)

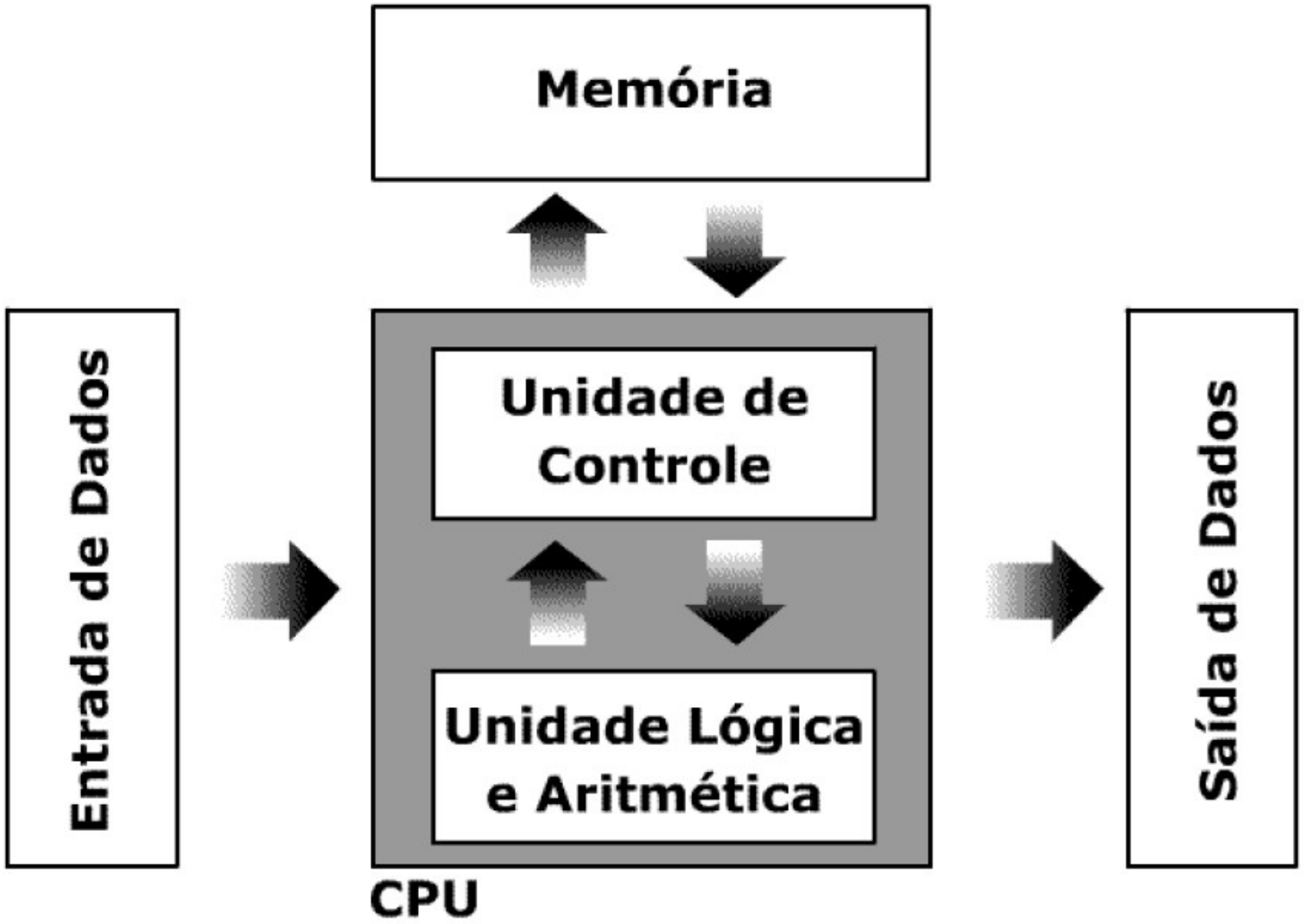


Processador

- **Unidade de Controle (UC)** - assume toda a tarefa de controle das ações a serem realizadas pelo computador, comandando todos os demais componentes de sua arquitetura.
- **Registradores** - são utilizados para assegurar o armazenamento temporário de informações importantes para o processamento de uma dada instrução.

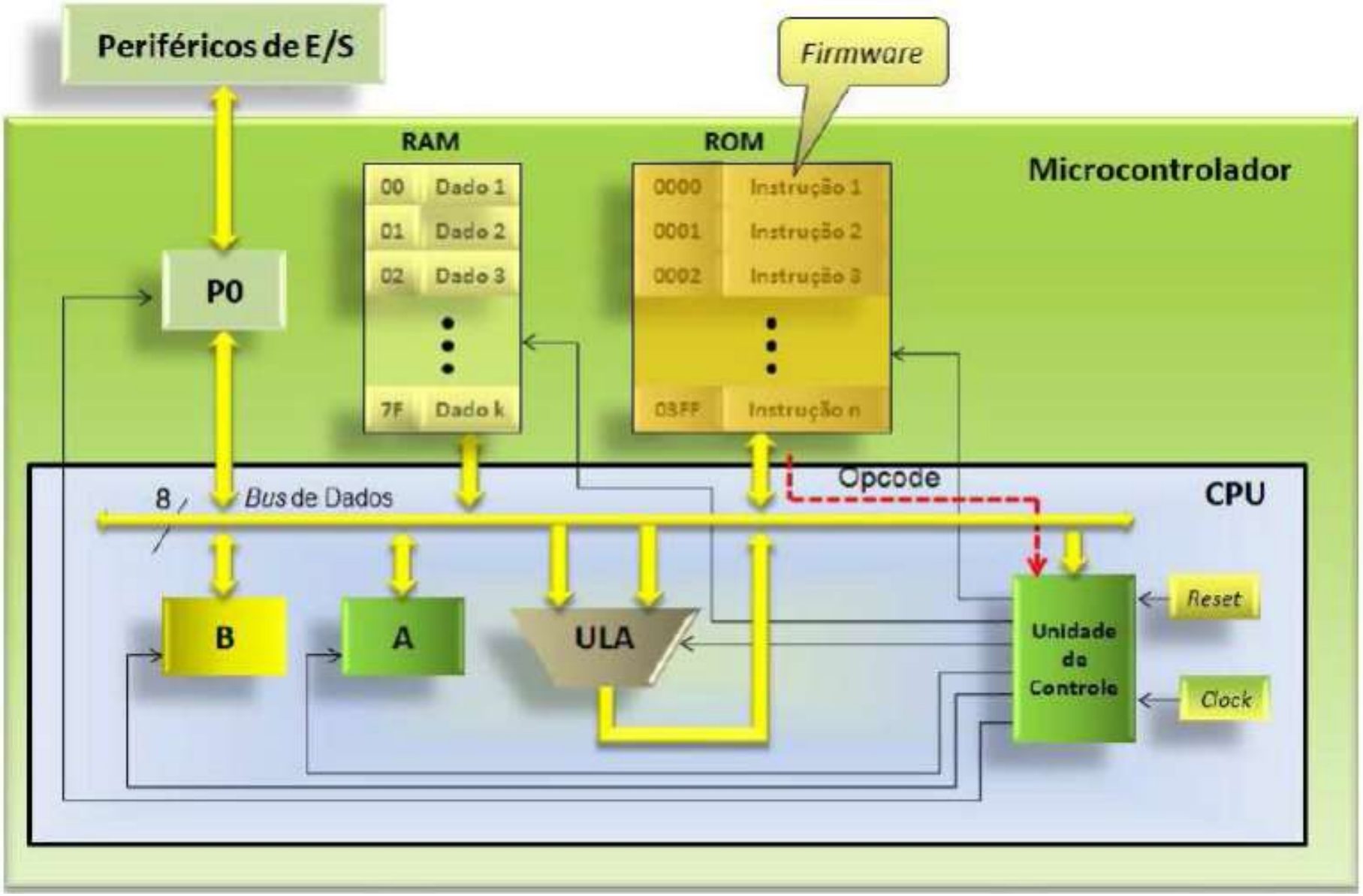
Prof. Misael Morais (moraiscg.uepb@gmail.com)

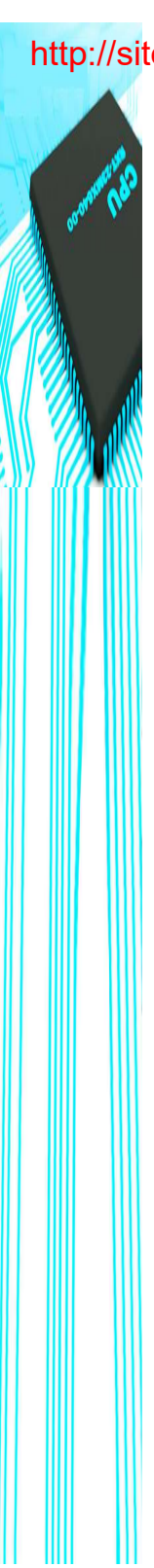
Processador



Prof. Misael Morais (moraiscg.uepb@gmail.com)

Modelo Hipotético



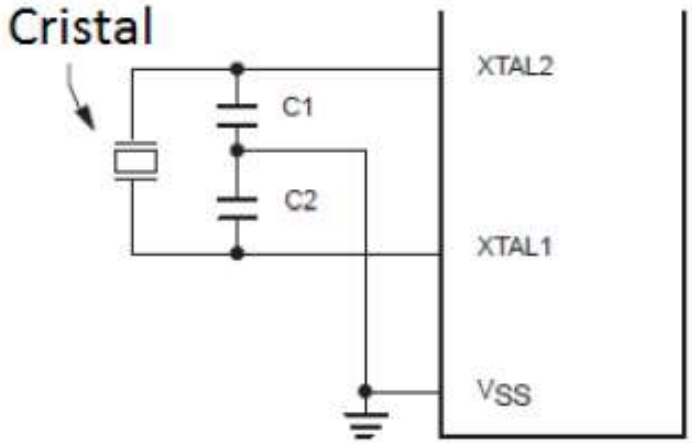


Clock

- Tem como função gerar um sinal oscilante entre nível 0 e 1 com freqüência e fase constante. Como o processador é basicamente uma máquina síncrona este gerará todos os seus sinais sincronizados com os níveis alto, baixos, subidas e descidas do sinal de clock.
- O clock pode ser gerado por osciladores internos de valor fixo, drives internos através de um cristal externo ou o uso de um gerador externo de clock. Limites máximos e mínimos devem ser respeitados.

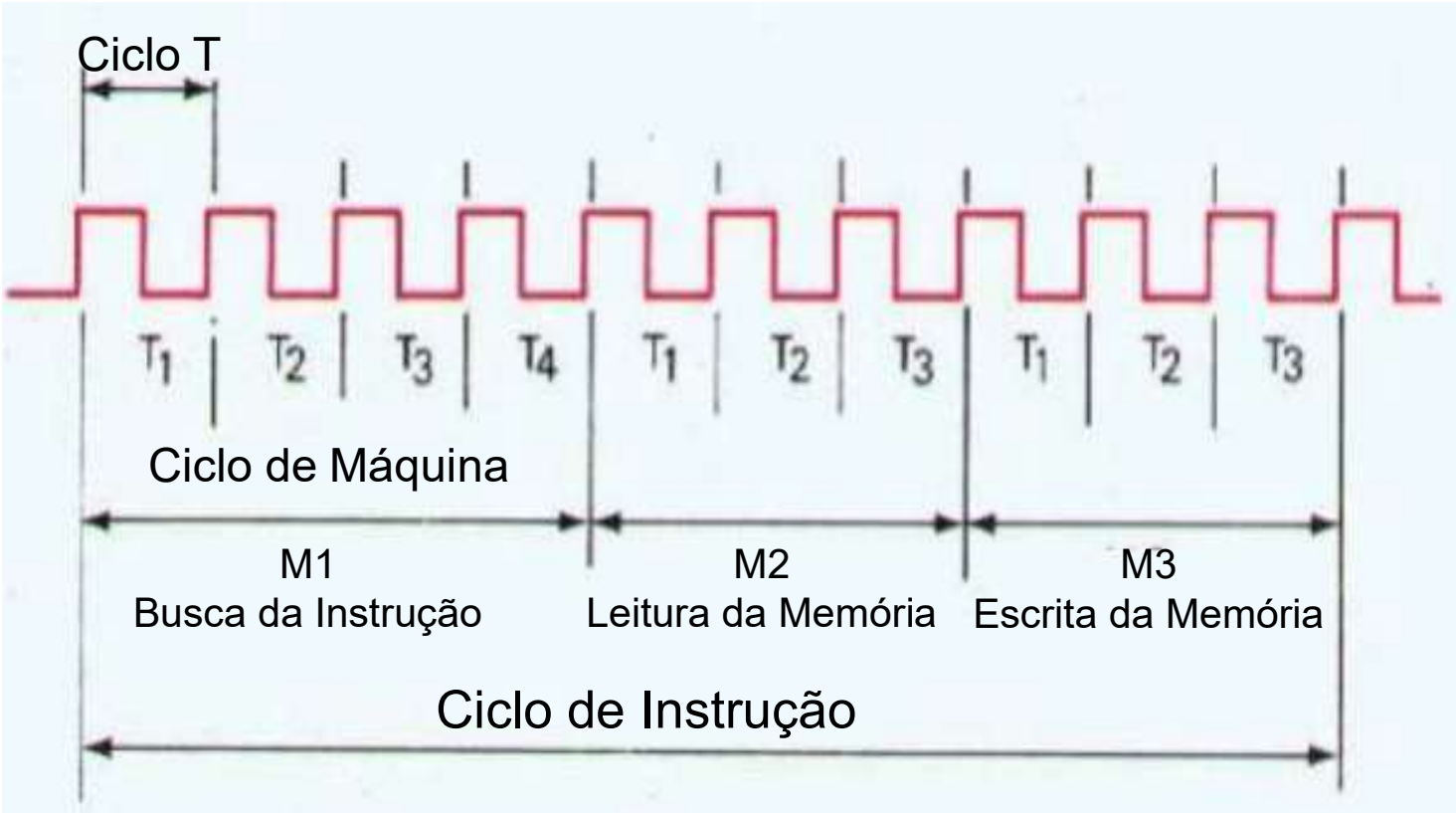
Clock

- Clock é um circuito oscilador que tem a função de sincronizar e ditar a medida de velocidade de transferência de dados no computador, por exemplo, entre o processador e a memória principal. Esta freqüência é medida em ciclos por segundo, ou Hertz.



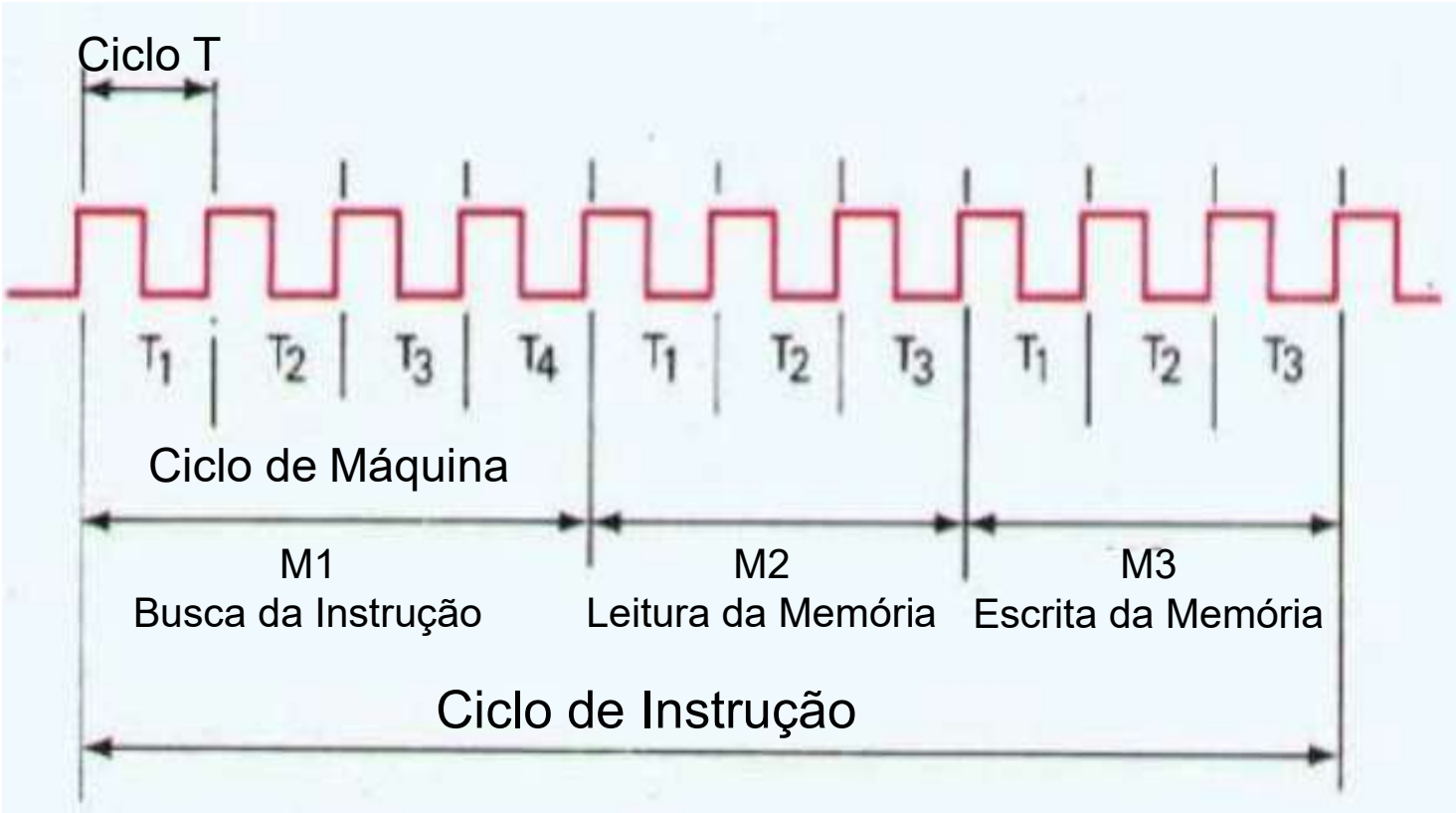
Prof. Misael Morais (moraiscg.uepb@gmail.com)

Clock



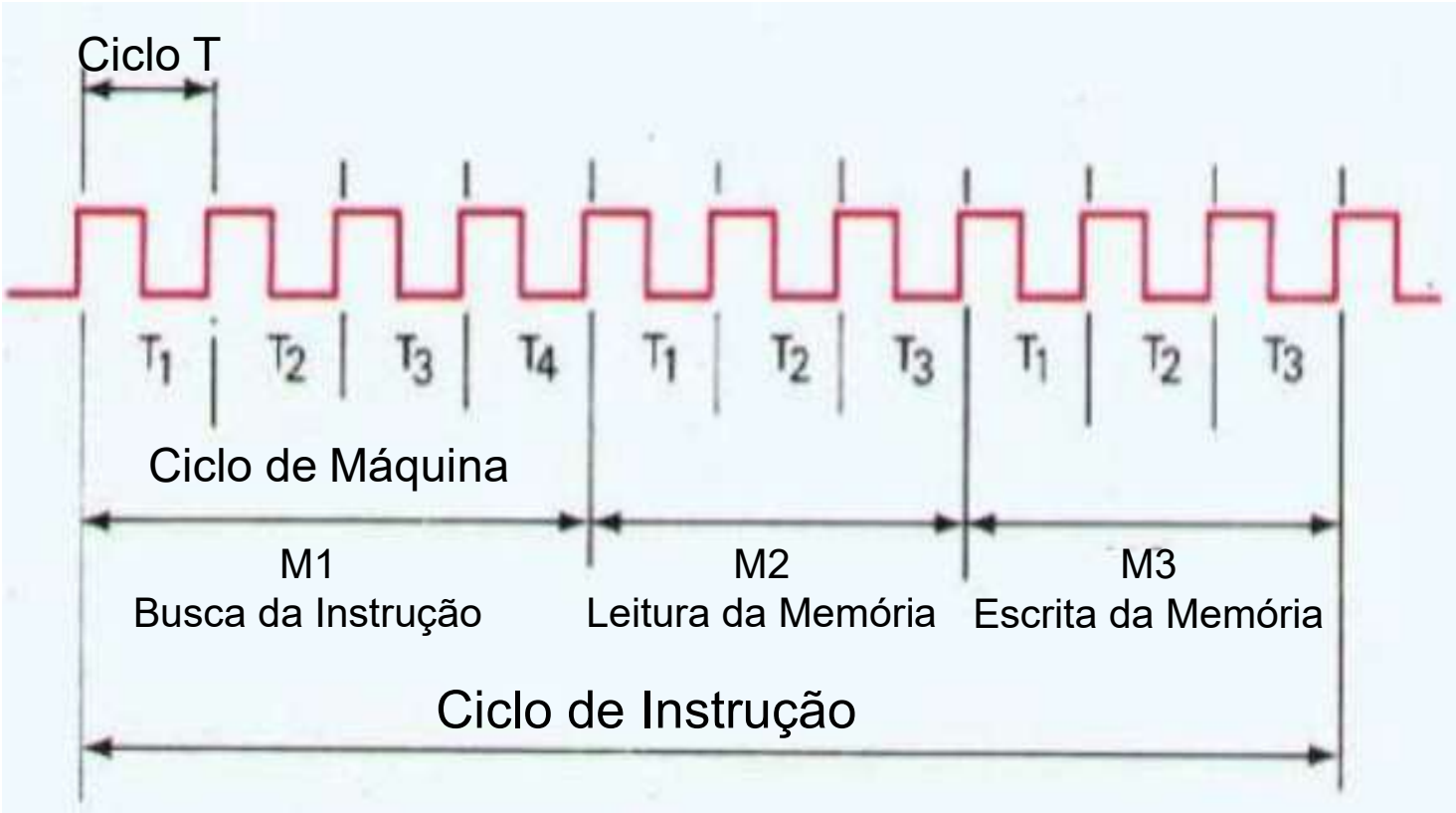
Ciclo de Clock – consiste de um período do clock, baseado no qual o processador executa cada passo interno.

Clock



Ciclo de Máquina – é cada uma das operações básicas que o µP pode executar. Sua execução é determinada pela instrução que está no momento sendo executada. Sua duração varia de entre microprocessadores.

Clock



Ciclo de Instrução – é o conjunto de ciclos de máquina necessários para a busca e execução de uma instrução. Varia de acordo com a complexidade da instrução.

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Ciclos de Máquinas

Opcode Fetch – Chamado de busca de instrução, é o primeiro ciclo após o reset, e sempre o primeiro ciclo de cada instrução. Coloca nos pinos de endereço o valor de PC e seleciona a leitura de memória de programa. Em seguida copia o valor presente nos pinos de dados (instrução) para ser decodificado.

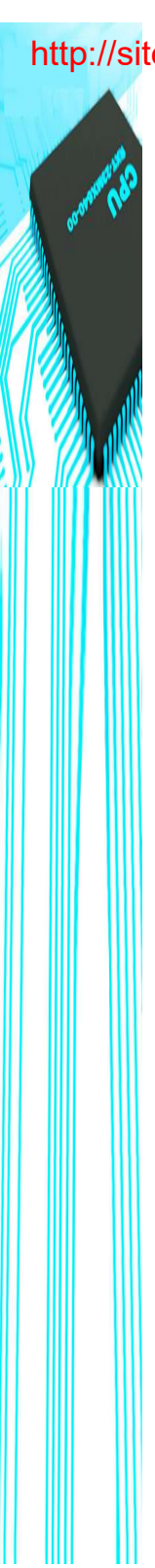
Memory Read – Operação de leitura em memória similar ao Opcode Fetch. As diferenças são que o endereço colocado no barramento foi gerado pela instrução atual, e que o dado lido será usado como operando pela instrução atual e não decodificado.

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Ciclos de Máquinas

Memory Write – Operação de escrita em memória onde o endereço foi gerado pela instrução atual e o dado será fornecido para a memória pelo uP. O dado foi fornecido como operando da instrução ou obtido como resultado da execução da mesma.

Prof. Misael Morais (moraiscg.uepb@gmail.com)

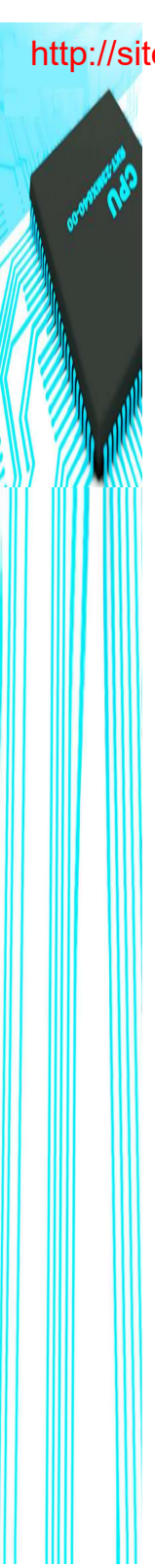


Barramento

Em Arquitetura de Computadores, um **barramento** é um conjunto de linhas de comunicação (condutor elétrico ou fibra ótica) que permitem a interligação entre dispositivos de um sistema de computação (CPU; Memória Principal; HD e outros periféricos), ou entre vários sistemas de computação.

- Um barramento, ou bus, nada mais é do que um caminho comum pelo qual os dados trafegam dentro do computador.
- O tamanho de um barramento é importante pois ele determina quantos dados podem ser transmitidos em uma única vez. Por exemplo, um barramento de 16 bits pode transmitir 16 bits de dado, e um barramento de 32 bits pode transmitir 32 bits de dados a cada vez.

Prof. Misael Morais (moraiscg.uepb@gmail.com)

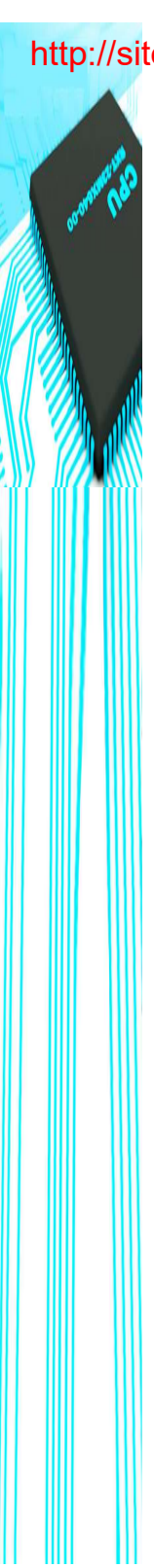


Funções do Barramento

Barr. de Endereços – Conjunto de pinos/sinais físicos unidirecionais que são gerados pelo uP para endereçar a posição de memória ou periférico a ser acessado. O número de pinos está ligado a capacidade de memória do uP.

Barr. de Dados – Conjunto de pinos bidirecionais que permitem que as informações endereçadas das memórias ou periféricos possam sair ou chegar ao uP, dependendo se a instrução é de escrita ou leitura.

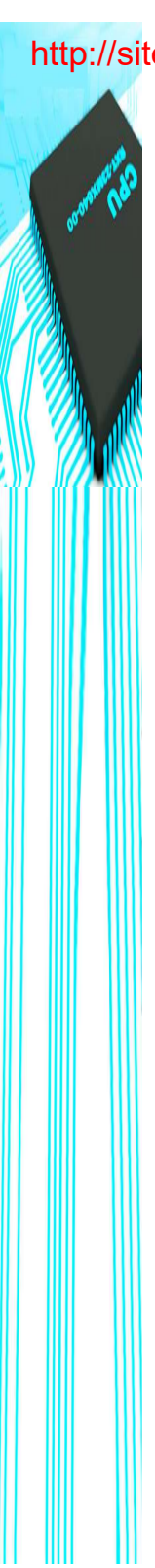
Barr. de Controle – Sinais de saída do uP que servem para indicar aos blocos externos que tipo de evento está ocorrendo no momento. Quando de entrada indica que um serviço está sendo solicitado ao uP.



Barramentos

- **Barramentos Internos:** ligam a CPU (processador) aos equipamentos que ficam dentro do gabinete.
- Existem diversos tipos de barramentos específicos para equipamentos diferentes:
  - IDE
  - ISA
  - PCI
  - AGP
  - SCSI

Prof. Misael Morais (moraiscg.uepb@gmail.com)



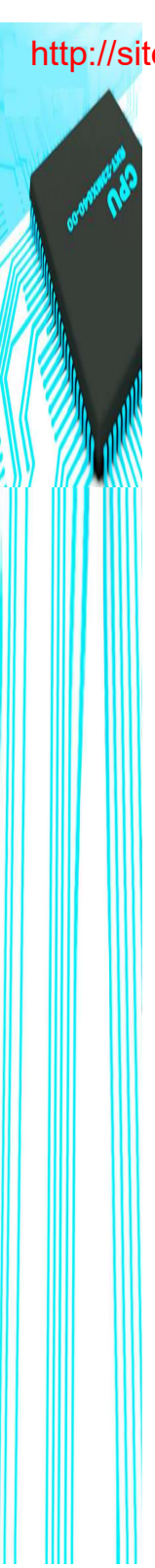
Microcontrolador

Def.: É um microcomputador encapsulado em um CI que pode conter, além da CPU, memórias RAM, ROM, Interfaces de E/S, circuitos de clock e reset, etc.

Conseqüências da compactação:

- **Diminuição do custo.**
- **Capacidade de processamento menor.**
- **Ênfase em minimizar recursos de externos.**
- **Aplicação mais específica.**

Prof. Misael Morais (moraiscg.uepb@gmail.com)



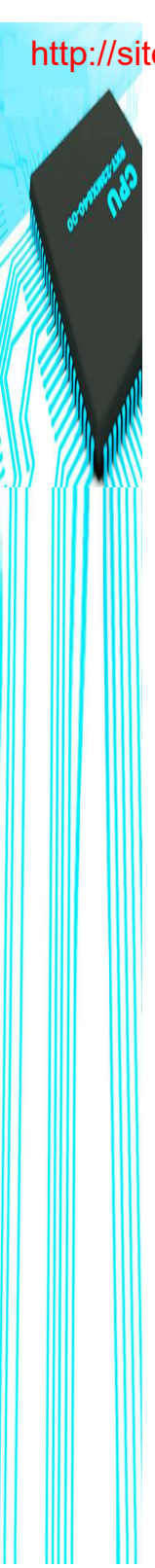
Microcontrolador

Def.: São utilizados em aplicações nas quais não há a necessidade da flexibilidade de interfaceamento e é executado um único software (firmware).

Componentes que podem ser incorporados:

- **Conversores A/D e D/A.**
- **PWM.**
- **Contadores e Temporizadores.**
- **Interfaces Seriais.**
- **Memórias: ROM, EPROM, EEPROM, FLASH, RAM.**

Prof. Misael Morais (moraiscg.uepb@gmail.com)



Família de Microcontroladores

MCS51 - Intel e outras empresas.

M68HC11 – Motorola.

Z8 – Zilog.

COP8 – National.

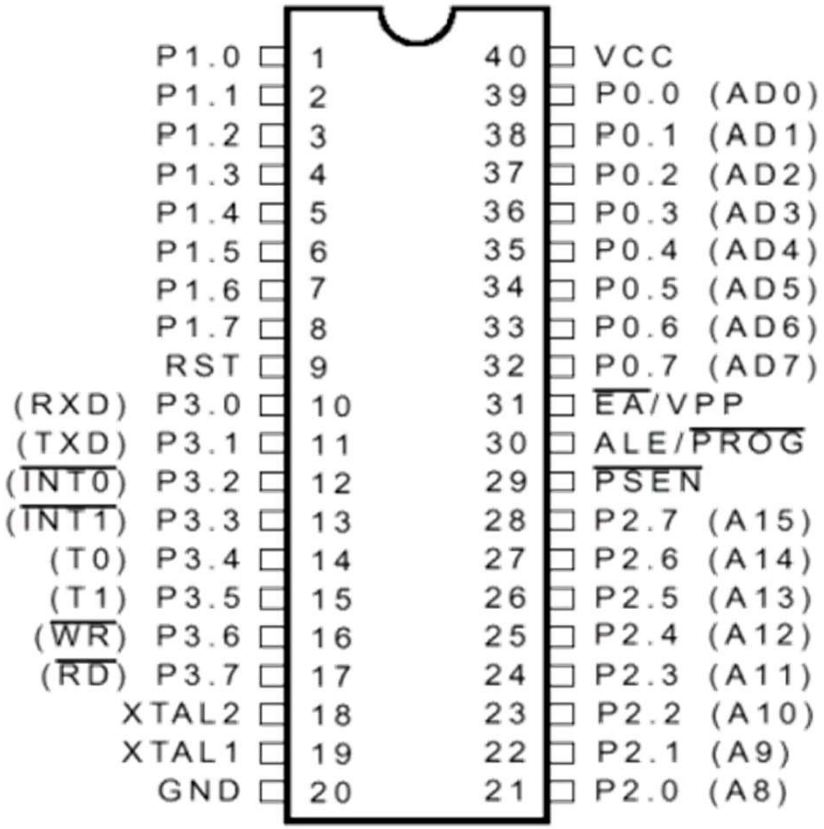
PIC – Microchip.

AVR – Atmel.

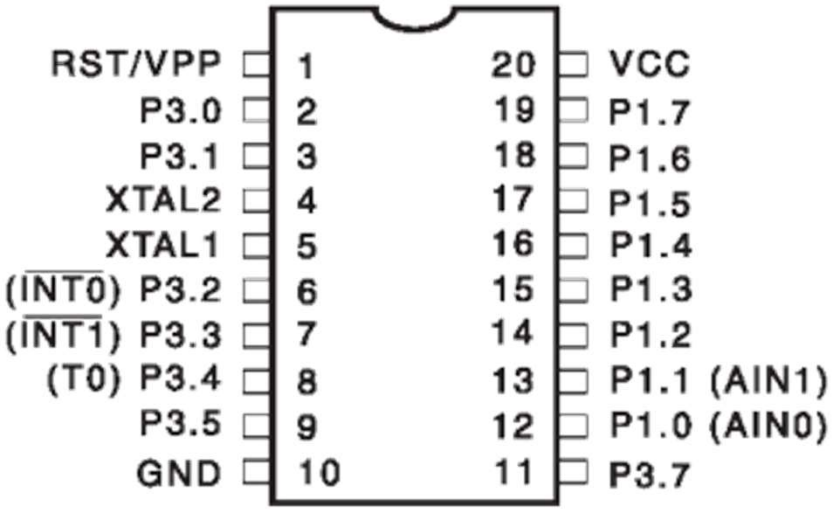
Prof. Misael Morais (moraiscg.uepb@gmail.com)

Exemplo de Microcontroladores

AT89C51(DIP40)

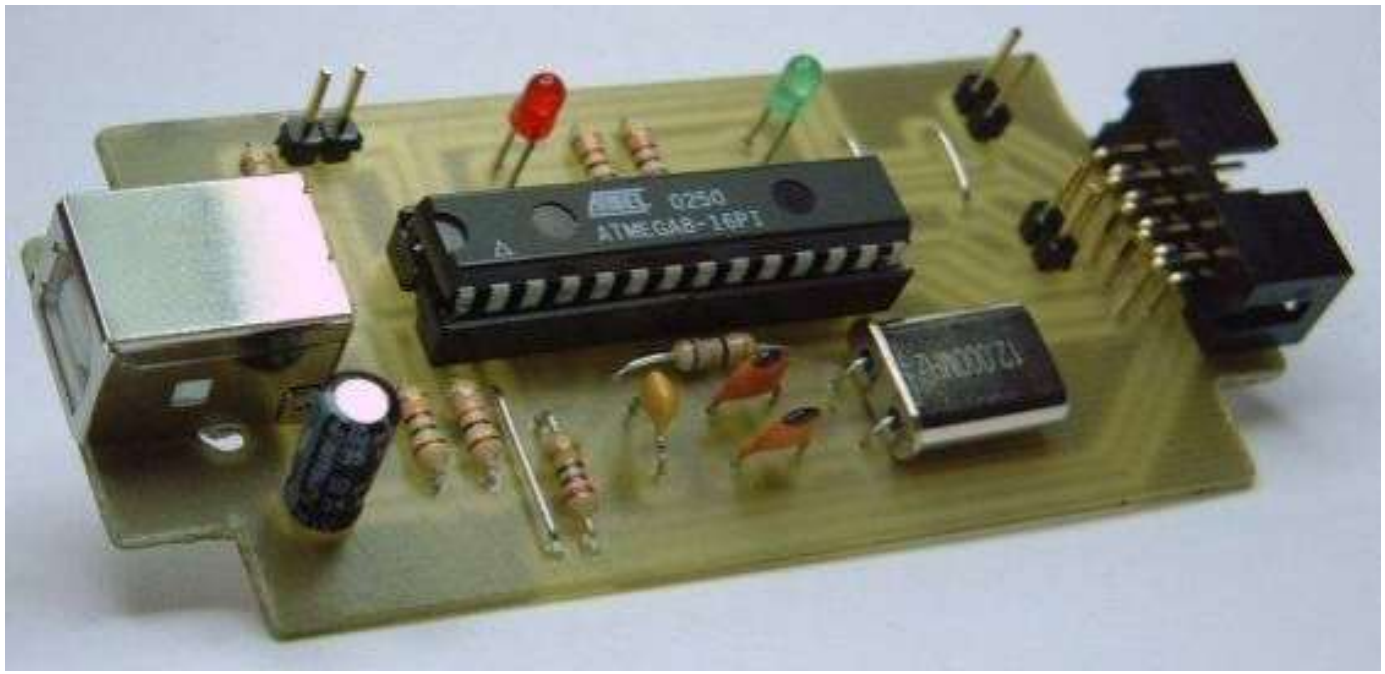


AT89C1051(DIP20)



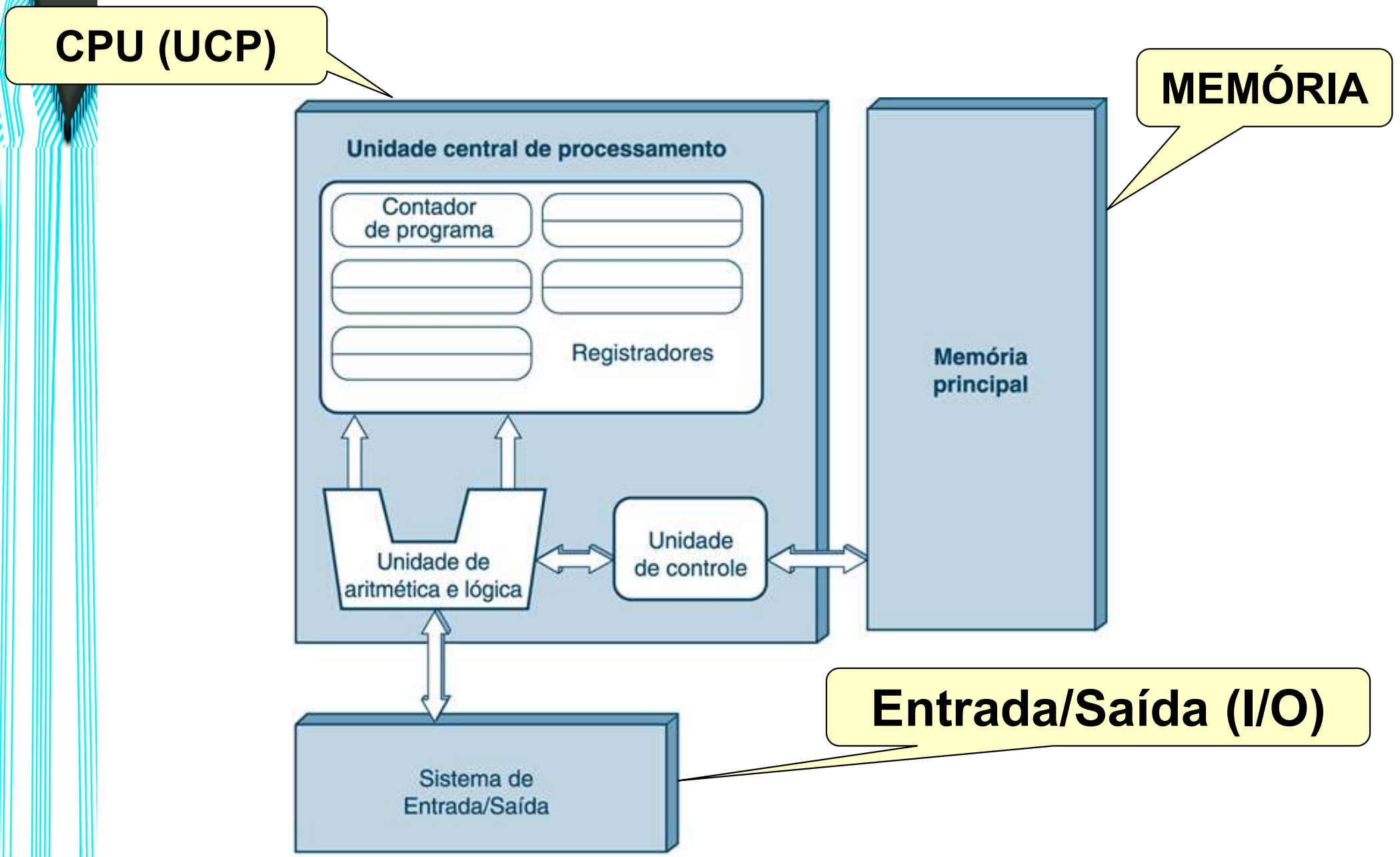
Prof. Misael Morais (moraiscg.uepb@gmail.com)

Exemplo de Microcontroladores



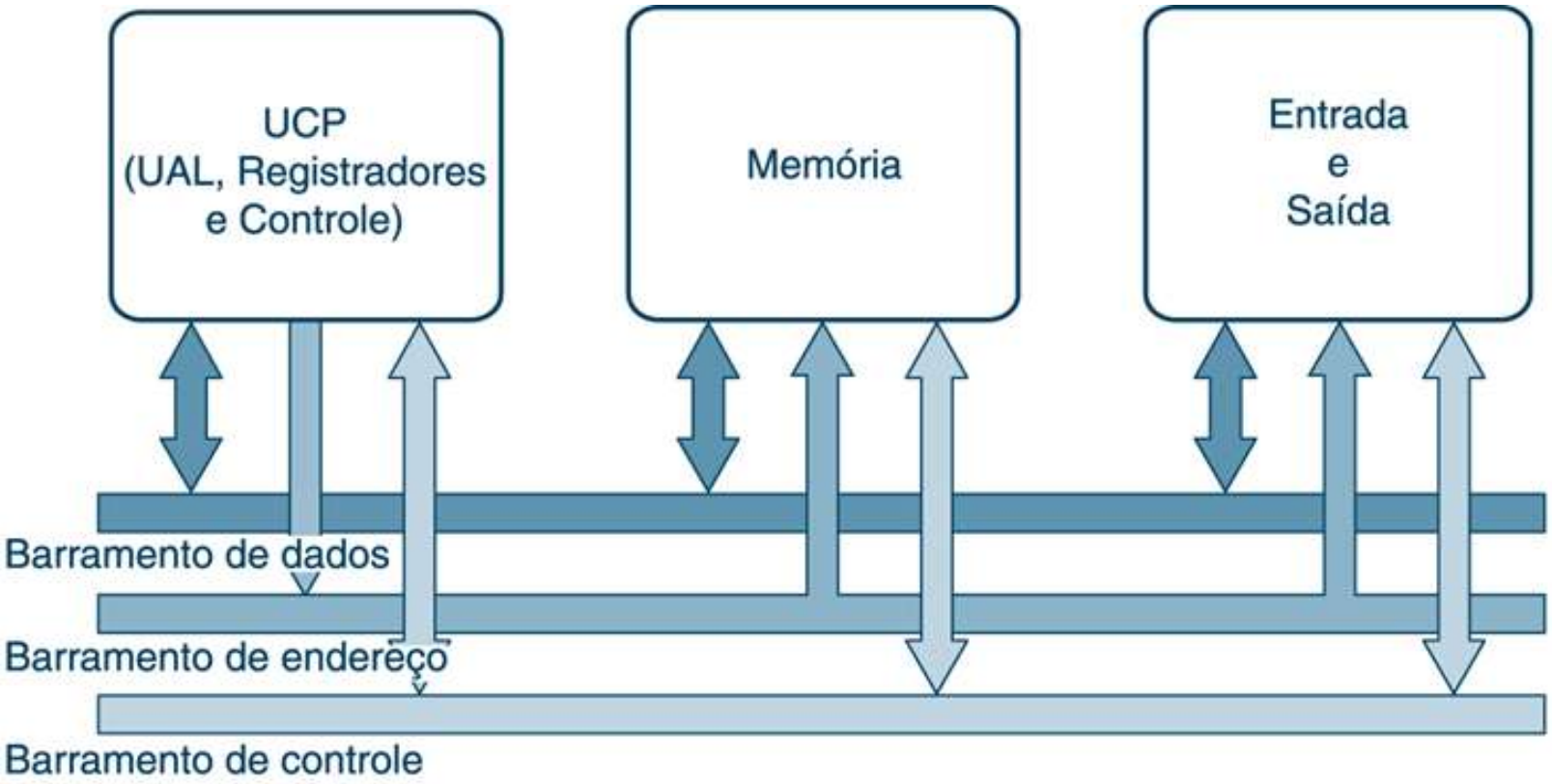
Prof. Misael Morais (moraiscg.uepb@gmail.com)

Modelo: Arquitetura de von Neumann



Prof. Misael Morais (moraiscg.uepb@gmail.com)

Arquitetura de von Neumann com barramento do sistema



Prof. Misael Morais (moraiscg.uepb@gmail.com)

Arquitetura

- O termo 'Arquitetura de um computador' refere-se aos atributos de um sistema que são visíveis para o programador.
- Os atributos que tem impacto direto sobre a lógica de um programa.
- Exemplos:
  - número de bits de um tipo de dados;
  - Endereçamento de memória;

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Organização

- Já o termo 'Organização de um computador' refere-se as unidades operacionais e suas interconexões que implementam as especificações da sua arquitetura.
- Exemplos:
  - Sinais emitidos por uma interface;
  - Controle do barramento;

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Estruturação e Função

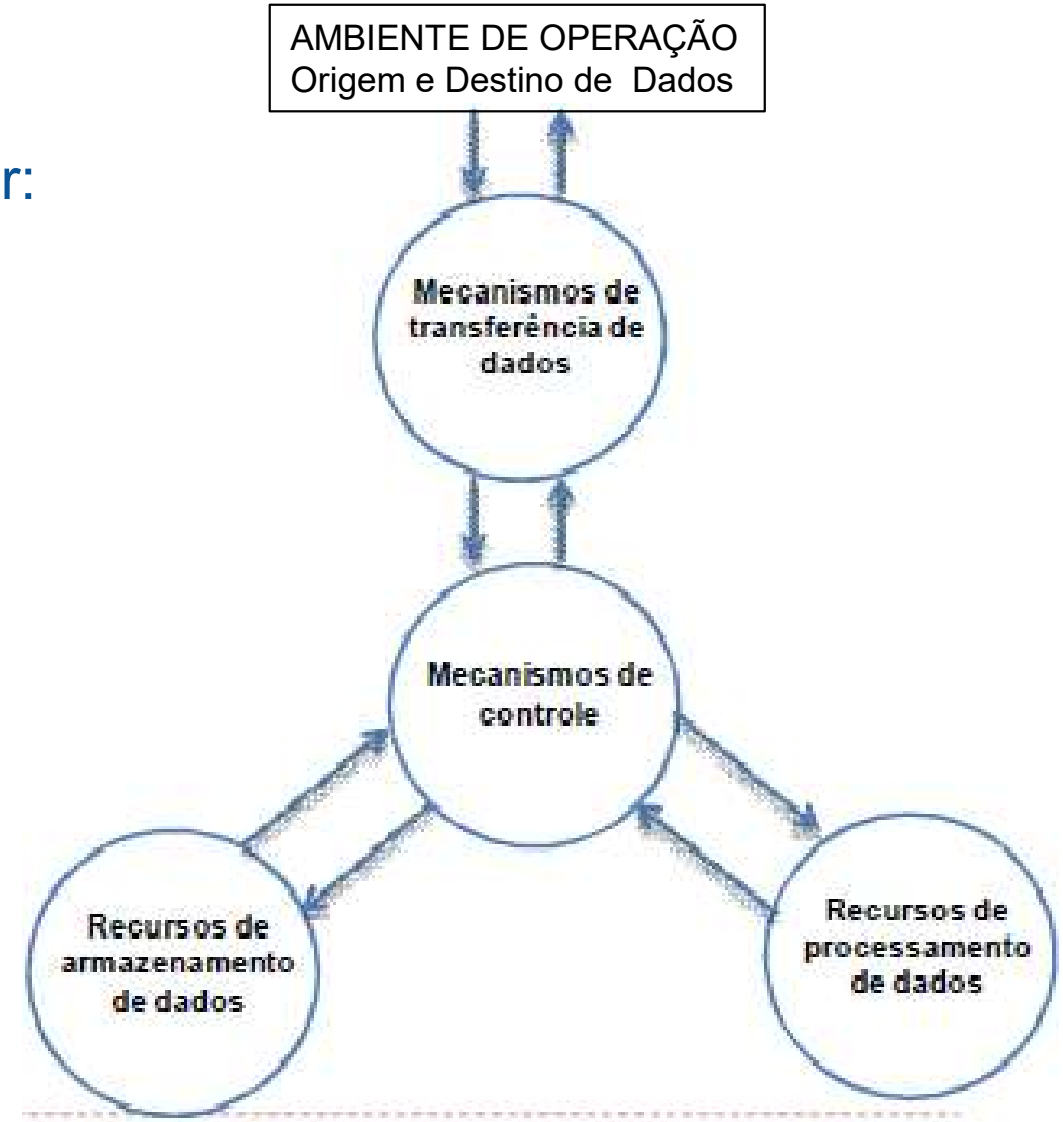
- O comportamento de cada nível depende apenas de uma caracterização abstrata;
- Em cada nível o projetista deve levar em consideração:
  - Estrutura: O modo como os componentes estão inter-relacionados;
  - Função: a operação de cada componente individual;

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Visão Funcional de um Computador

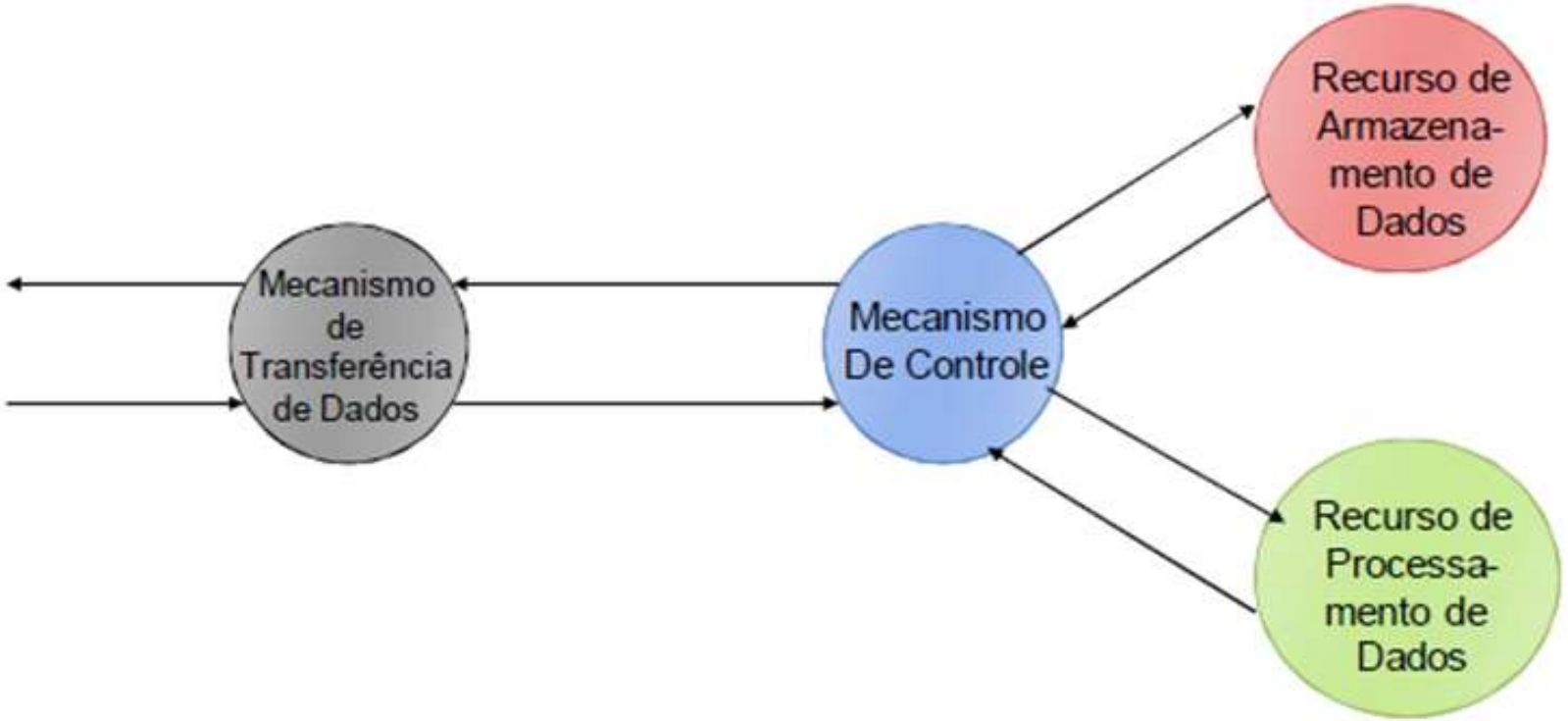
São 4 funções básicas que um computador pode desempenhar:

- Processamento de dados
- Armazenamento de dados
- Transferência de dados
- Controle

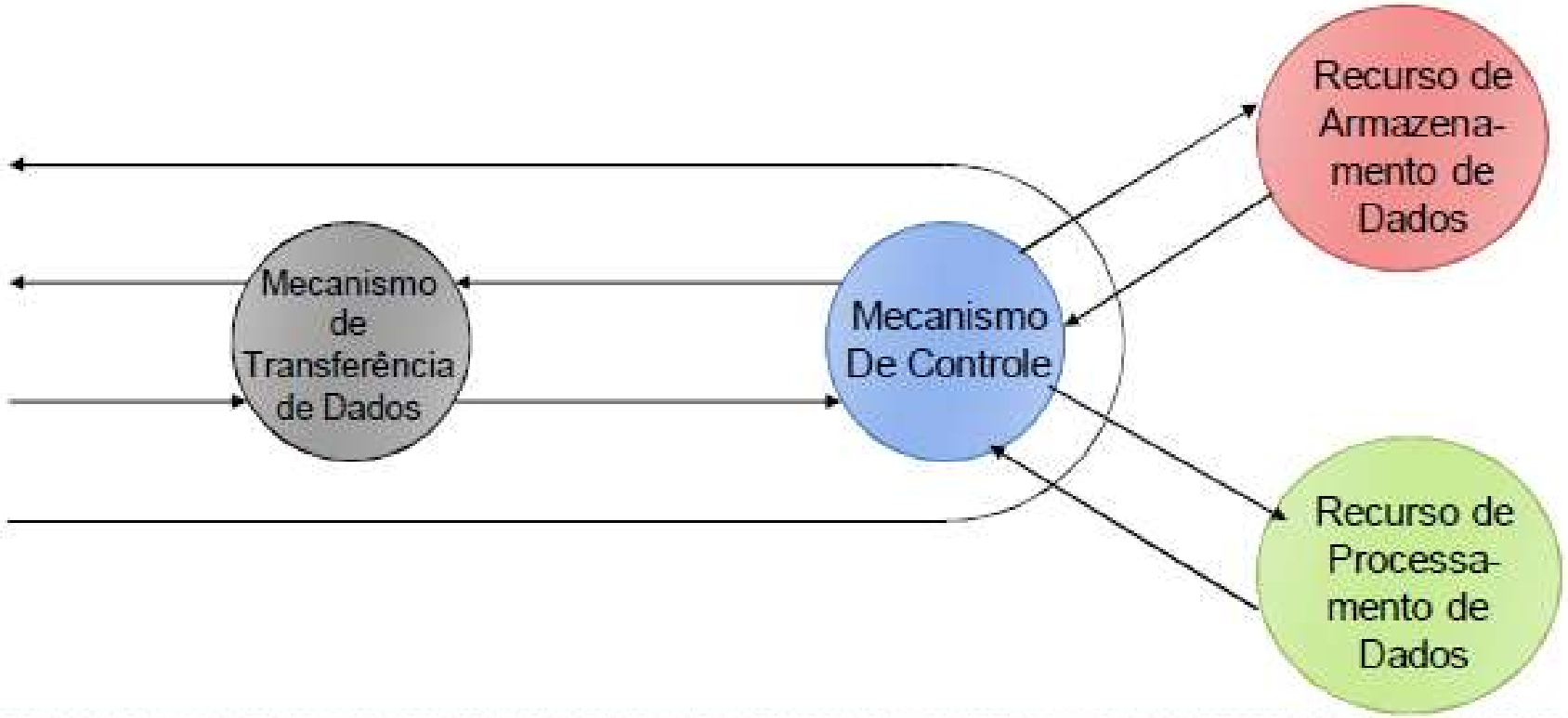


Prof. Misael Morais (moraiscg.uepb@gmail.com)

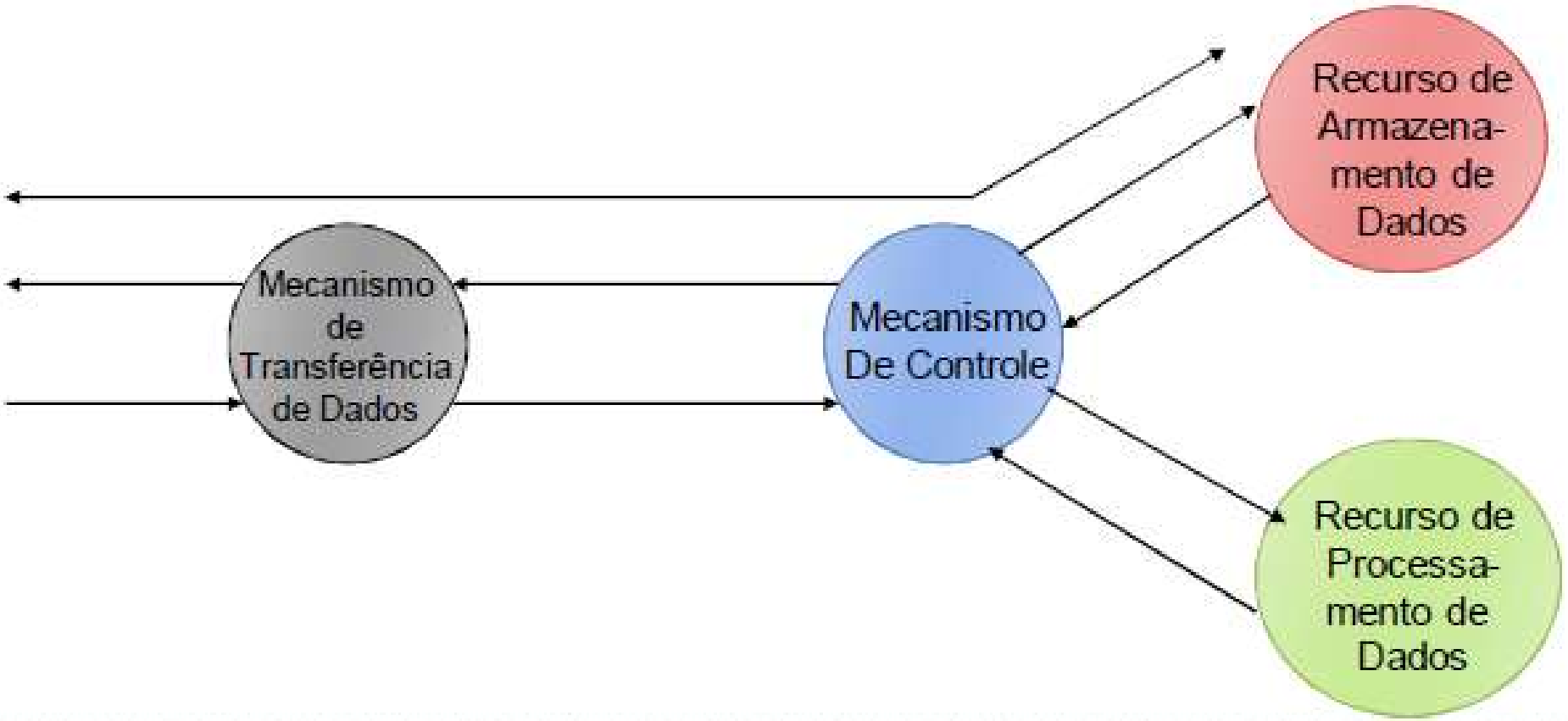
Visão Funcional de um Computador



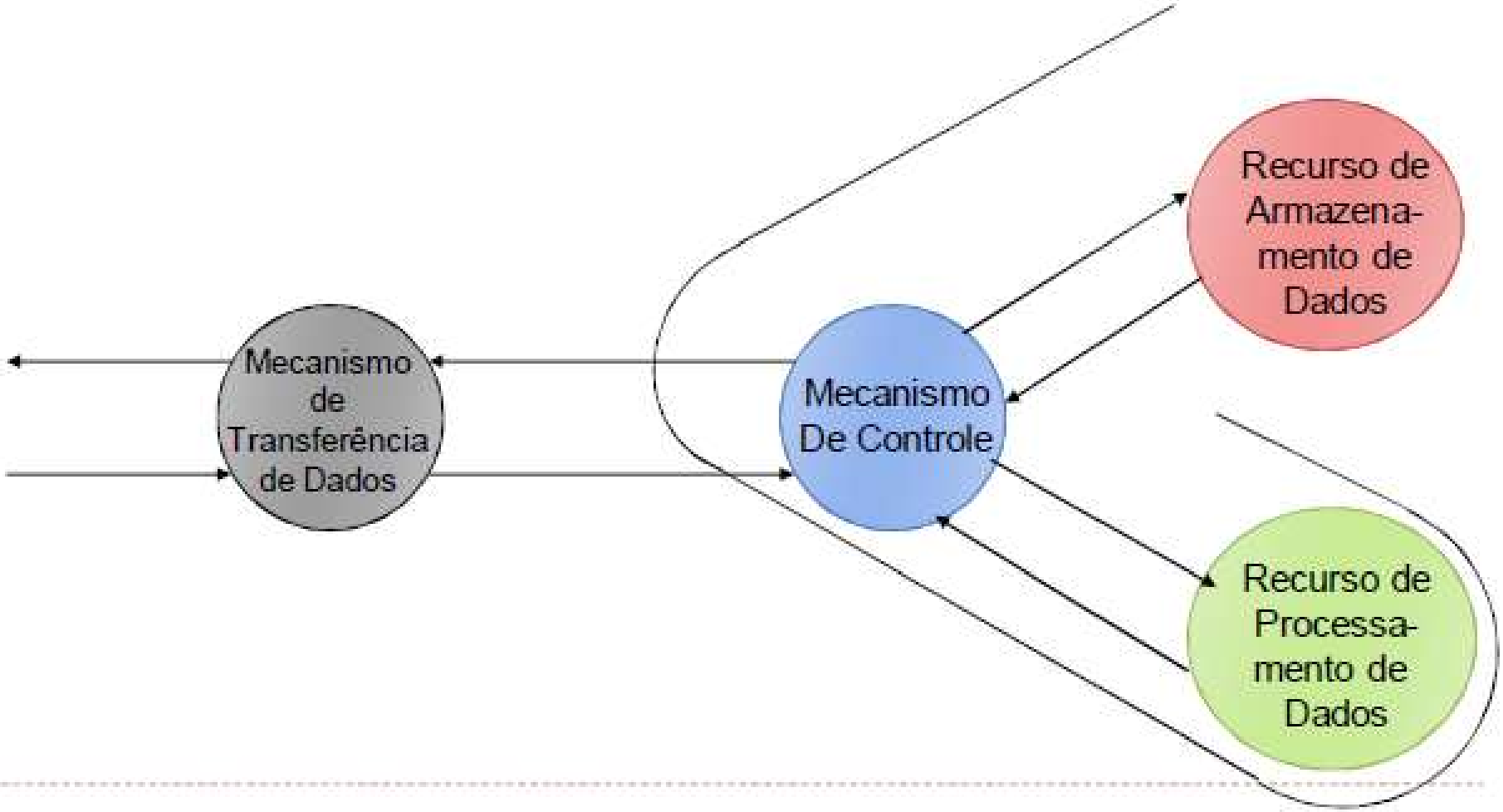
Operações


Recurso de Armazena-mento de Dados
Mecanismo de Transferência de Dados
Mecanismo De Controle
Recurso de Processa-mento de Dados

Armazenamento



Operação


Recurso de Armazena-mento de Dados
Mecanismo de Transferência de Dados
Mecanismo De Controle
Recurso de Processa-mento de Dados

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Memória

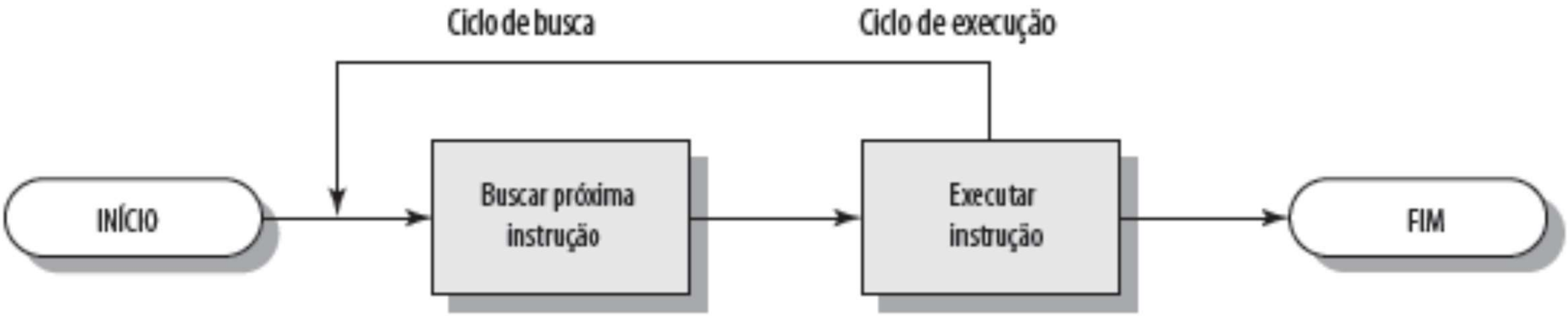
- Memória é uma unidade de armazenamento de dados, seja temporária ou permanente.
- Temos dois tipos de memória:
  - Memória interna (Memória Principal);
  - Memória externa (Memória Secundária);

Dispositivos de Entrada/Saída

- São dispositivos que através do barramento são utilizados pelo computador para interação com o mundo exterior ou como apoio as funções básicas

Ciclo de Instrução

- Duas etapas:
  - Busca
  - Execução



Prof. Misael Morais (moraiscg.uepb@gmail.com)

Exemplo como funciona um computador

- **Programa que faz cálculos:**
  - **Usuário digita: 15 + 35*3 (por exemplo)**
  - **O computador realiza:**
    - UC recebe este comando e os dados
    - UC decodifica e verifica o que precisa ser processado
    - UC envia para a ULA
    - ULA realiza o cálculo e retorna o valor 120
    - UC armazena na memória
    - UC mostra o resultado num dispositivo de saída

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Ex.: Formato de instruções (Campos)

Código de operação (opcode)	(identificação do operando)

(mnemônica)

MOV AC,XXX

No exemplo anterior:

- **0001 -> carregar AC a partir do endereço de memória especificado .**
- **0010 -> armazena o valor contido em AC no endereço de memória especificado**
- **0101 -> acrescenta ao valor contido em AC o valor contido no endereço de memória especificado**

(mnemônica)

MOV XXX,AC

(mnemônica)

ADD AC,XXX

Por exemplo:

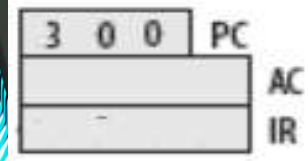
- **Operando imediato**
- **Endereço do operando**

- **Instruções de E/S específicas, exige barramento e *opcode* de E/S, neste caso 0 dispositivo não é mapeado na memória**

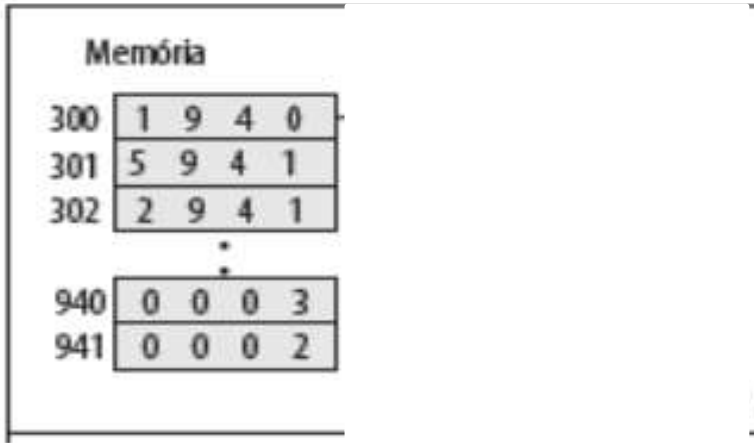
Prof. Misael Morais (moraiscg.uepb@gmail.com)

Exemplo de execução de programa

Inicio



Busca

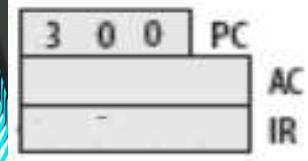


Execução

Prof. Misael Morais (moraiscg.uepb@gmail.com)

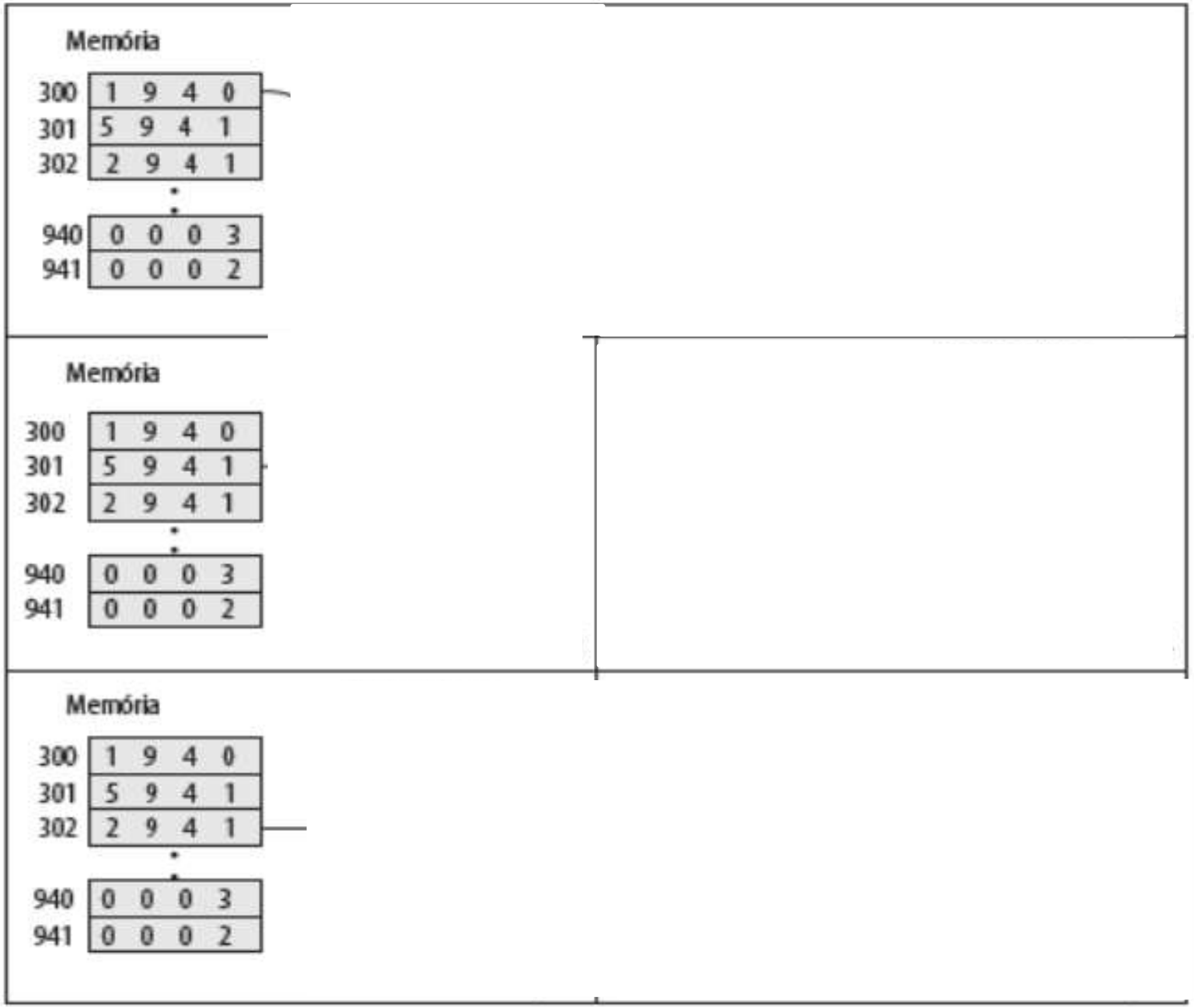
Exemplo de execução de programa

Inicio



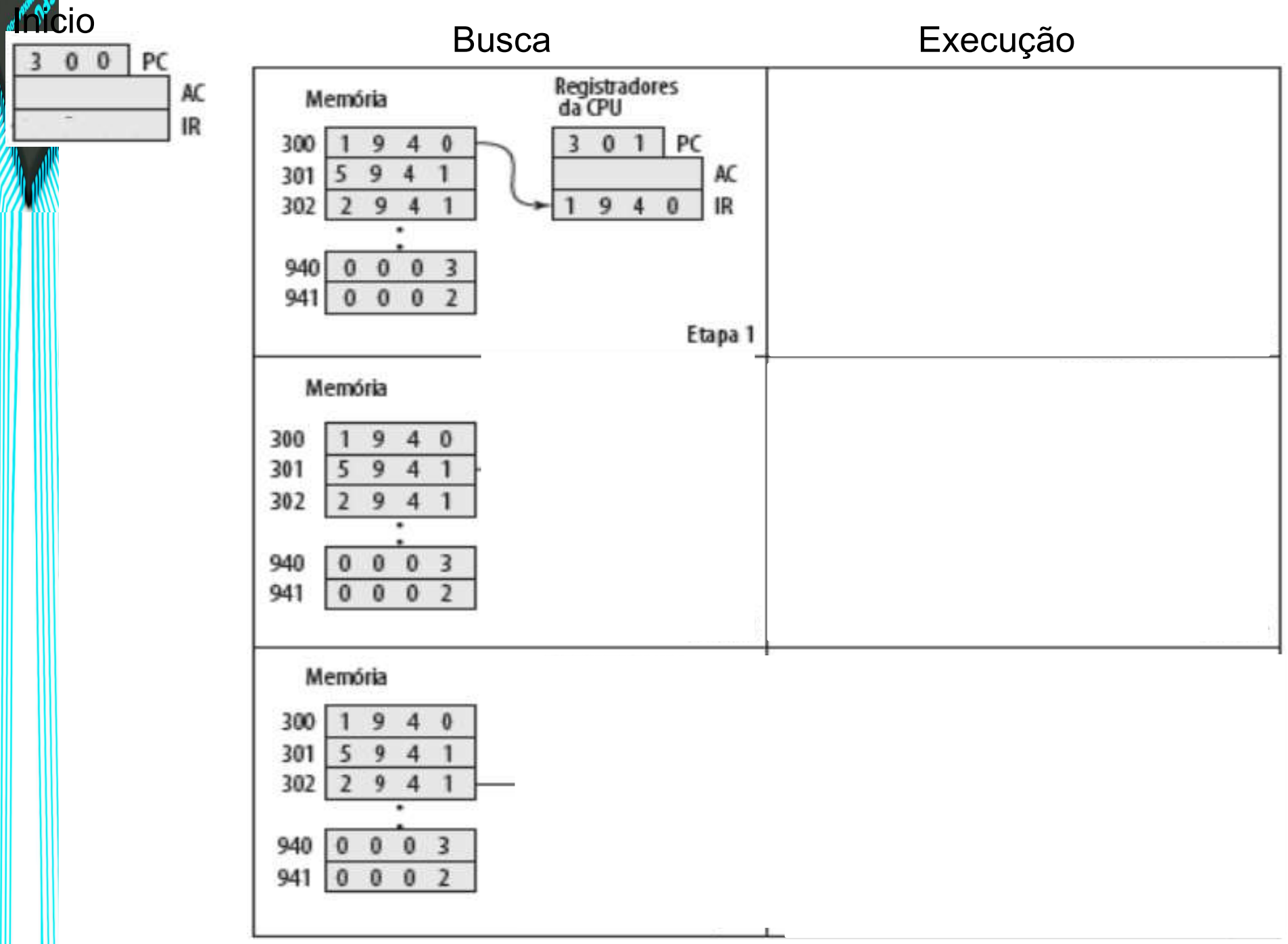
Busca

Execução



Prof. Misael Morais (moraiscg.uepb@gmail.com)

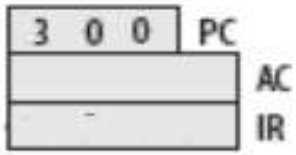
Exemplo de execução de programa



Prof. Misael Morais (moraiscg.uepb@gmail.com)

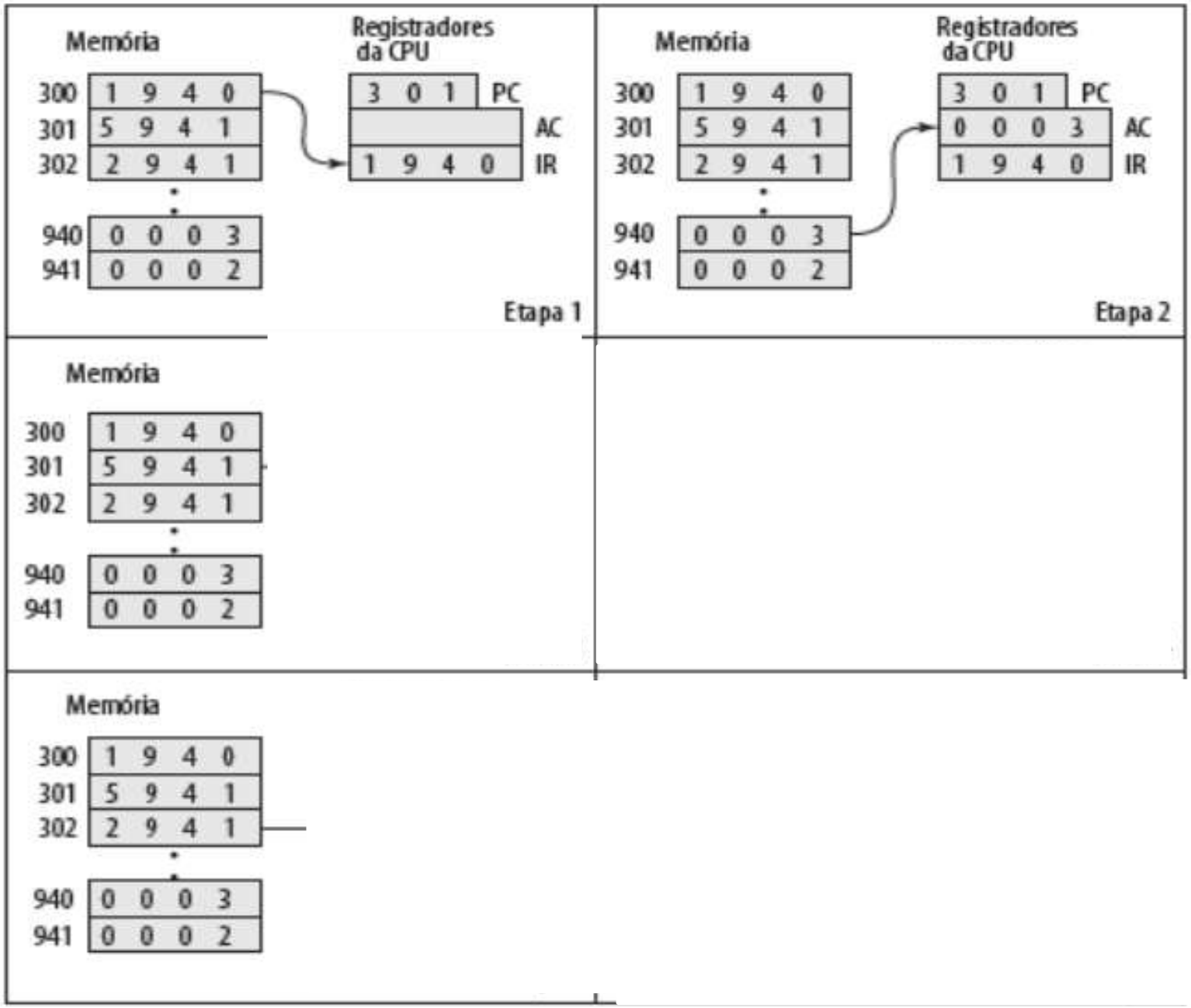
Exemplo de execução de programa

Inicio



Busca

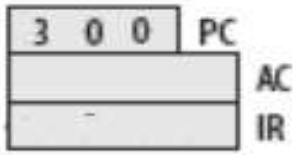
Execução



Prof. Misael Morais (moraiscg.uepb@gmail.com)

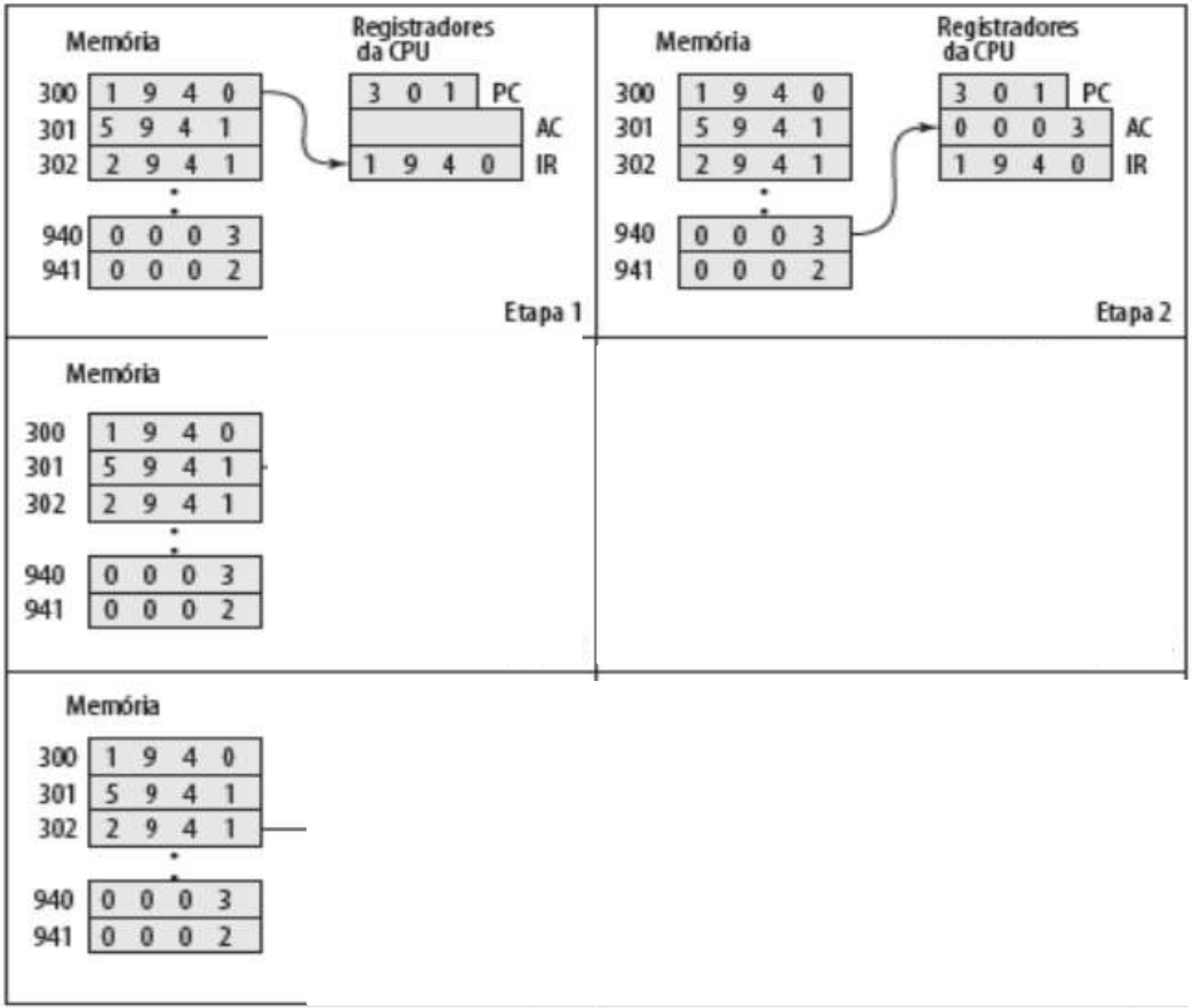
Exemplo de execução de programa

Inicio



Busca

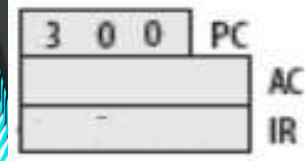
Execução



Prof. Misael Morais (moraiscg.uepb@gmail.com)

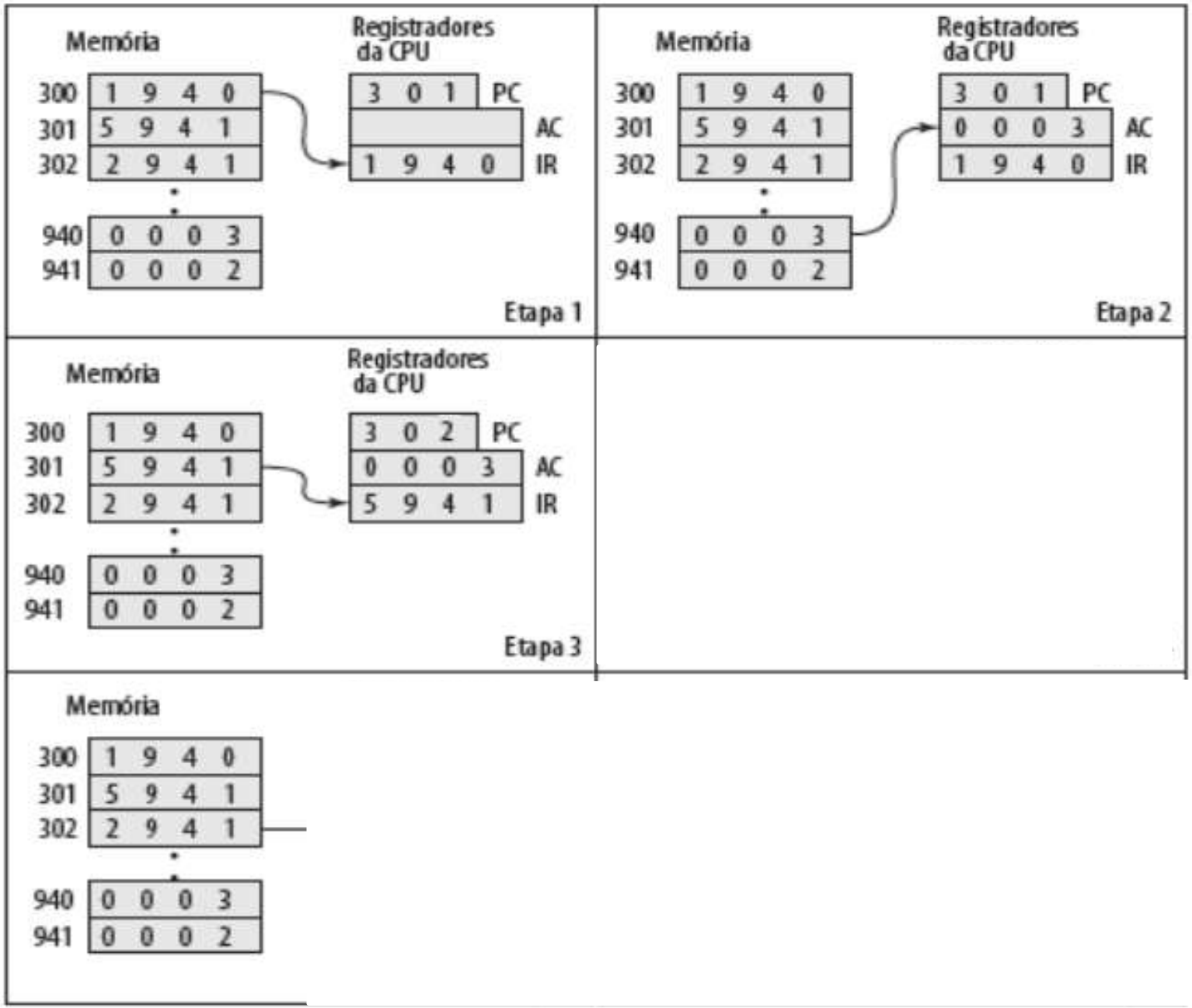
Exemplo de execução de programa

Inicio



Busca

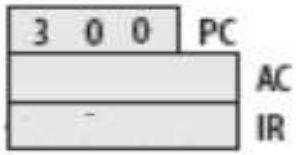
Execução



Prof. Misael Morais (moraiscg.uepb@gmail.com)

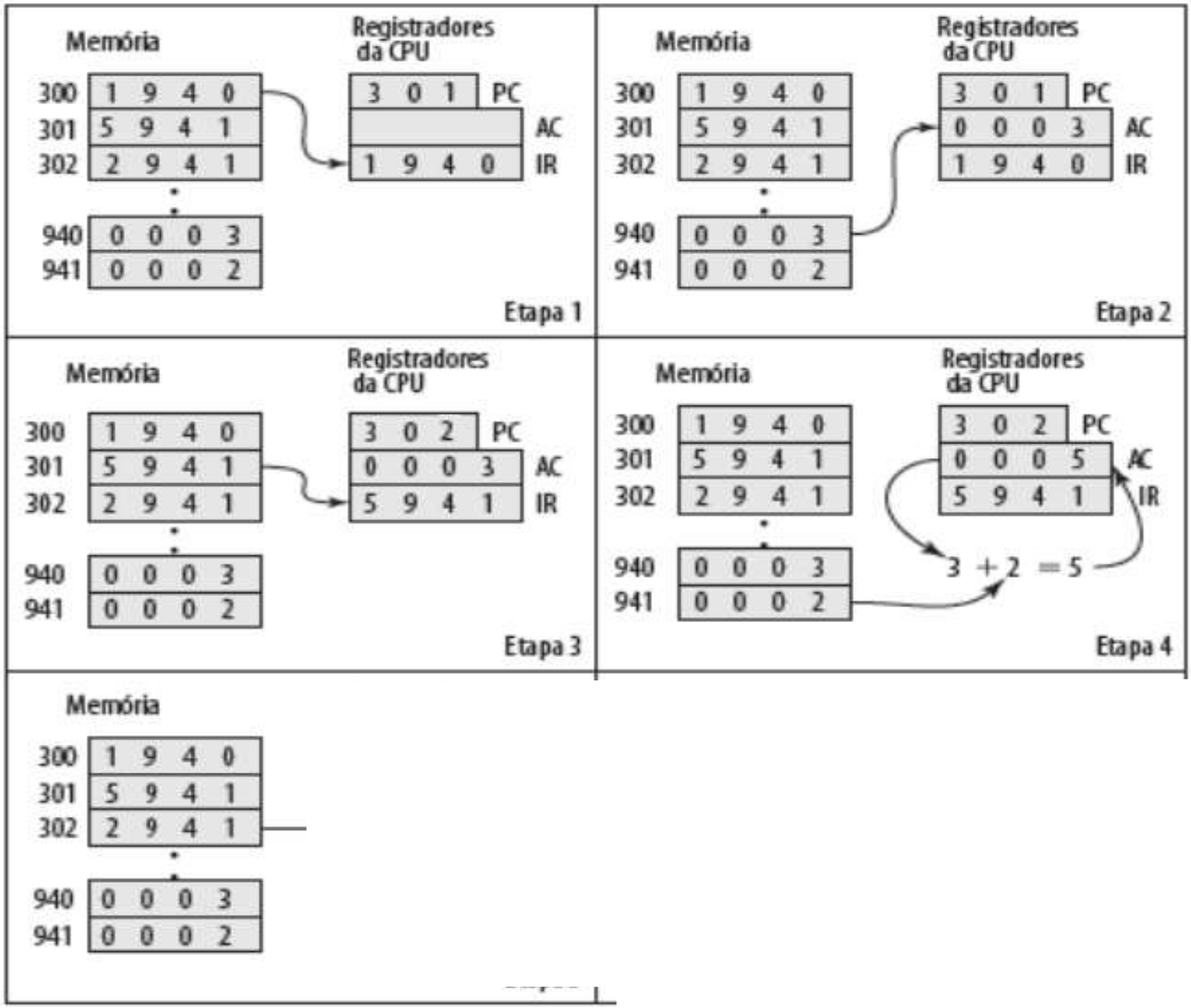
Exemplo de execução de programa

Inicio



Busca

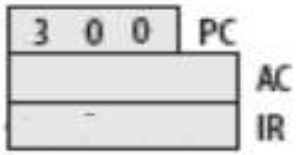
Execução



Prof. Misael Morais (moraiscg.uepb@gmail.com)

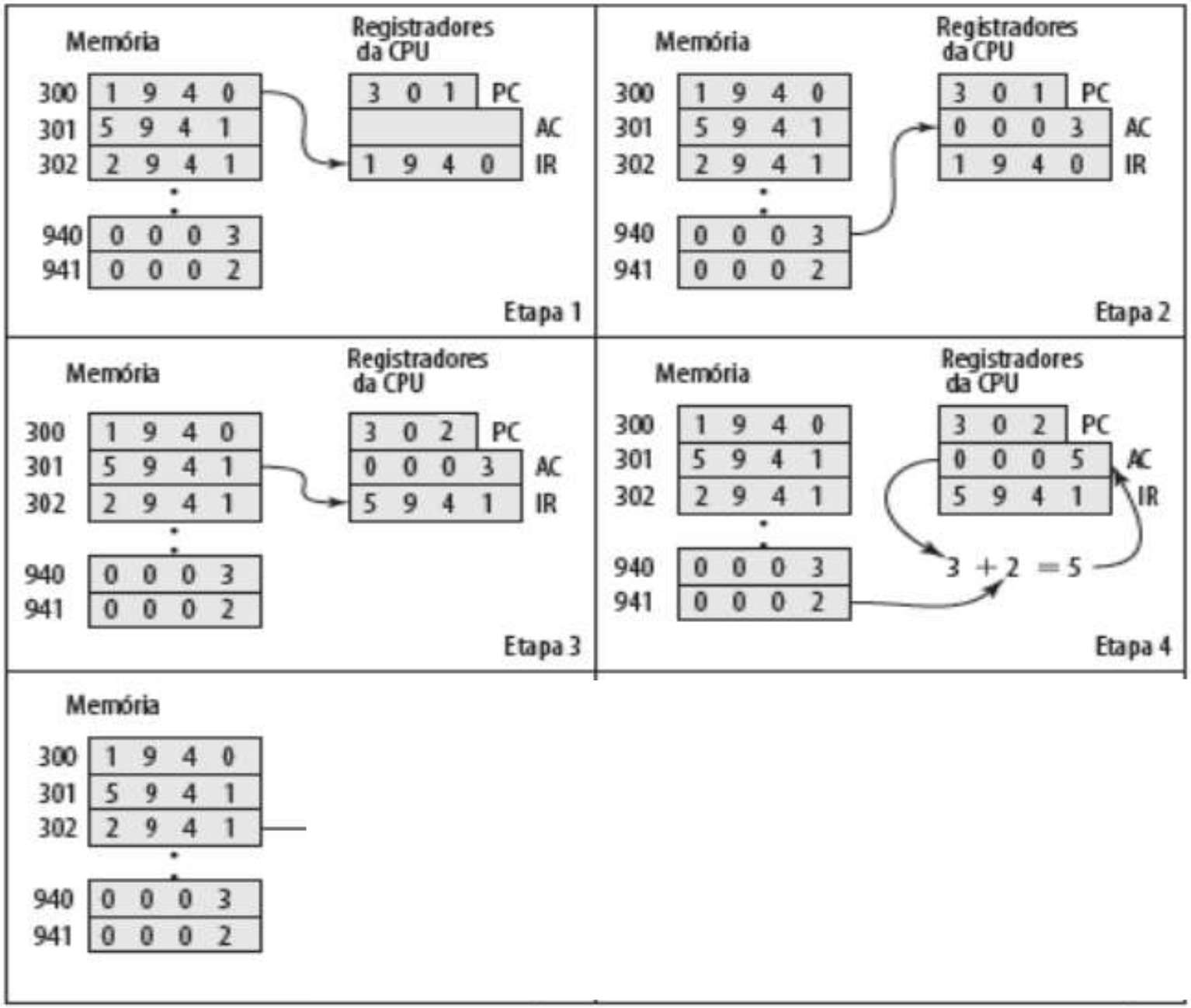
Exemplo de execução de programa

Inicio



Busca

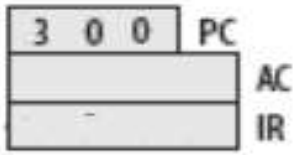
Execução



Prof. Misael Morais (moraiscg.uepb@gmail.com)

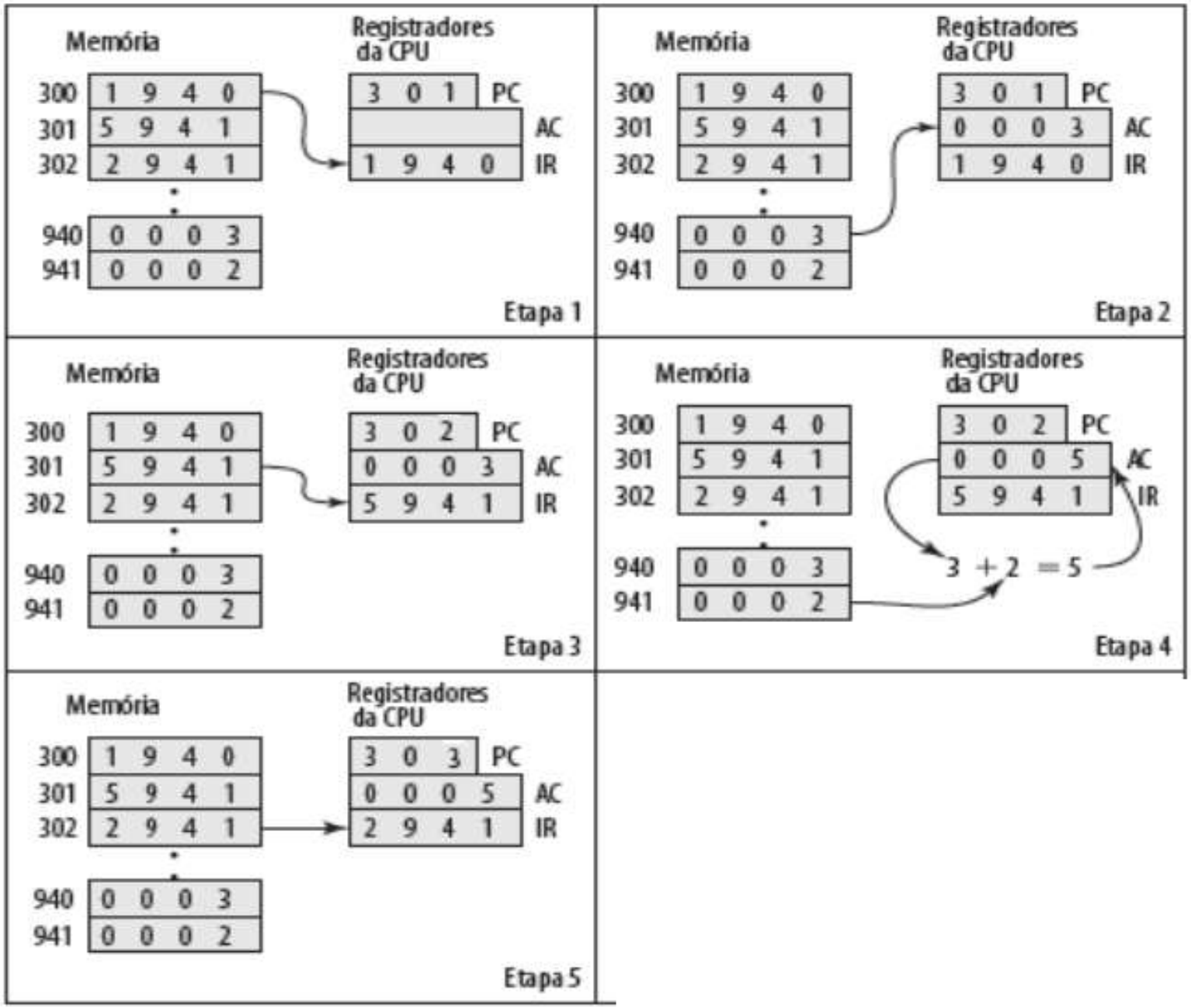
Exemplo de execução de programa

Inicio



Busca

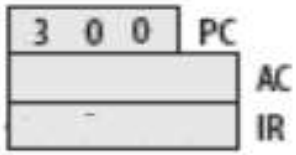
Execução



Prof. Misael Morais (moraiscg.uepb@gmail.com)

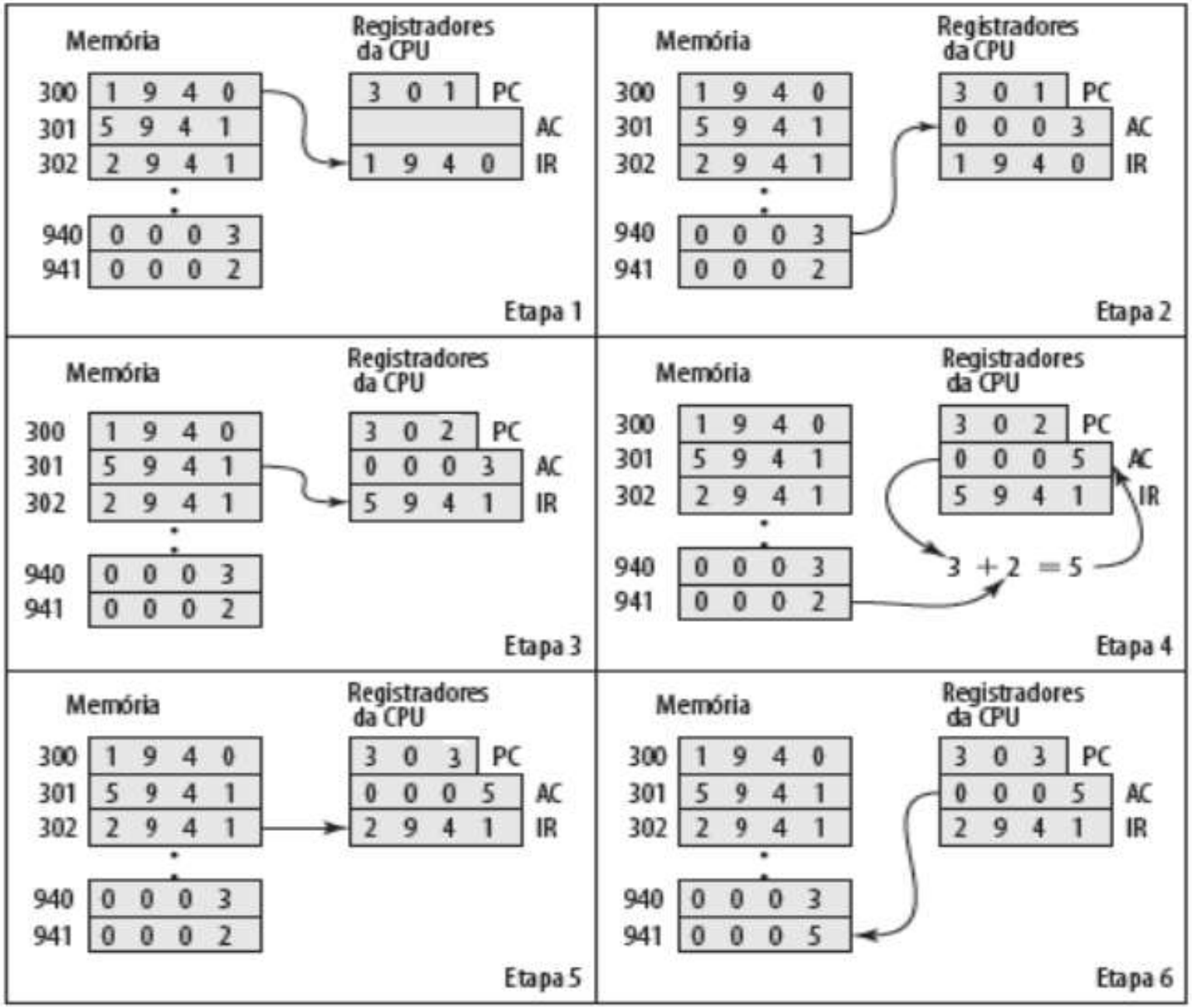
Exemplo de execução de programa

Inicio



Busca

Execução



Prof. Misael Morais (moraiscg.uepb@gmail.com)

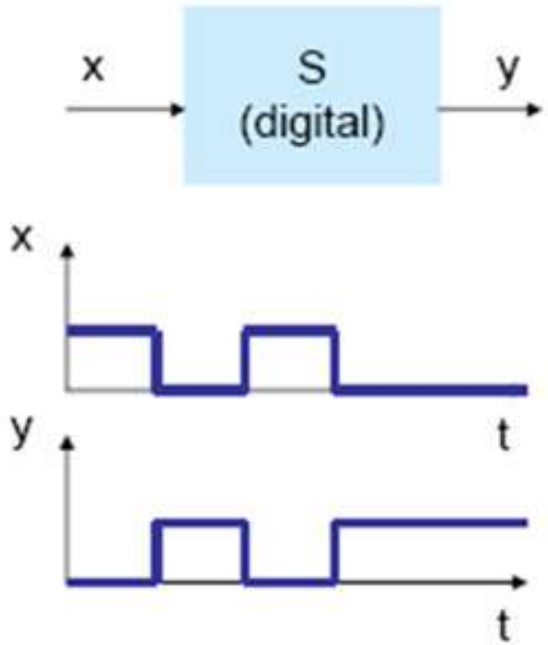
Representação da Informação em Sistemas de Computação

- **Informação** – representada por grandezas físicas.

- **Sistemas digitais**
  - Variáveis assumem valores discretos
  - Representação usual: sistema binário (máquina projetada para operar com 0 e 1)

Conceito – Sistema Digital

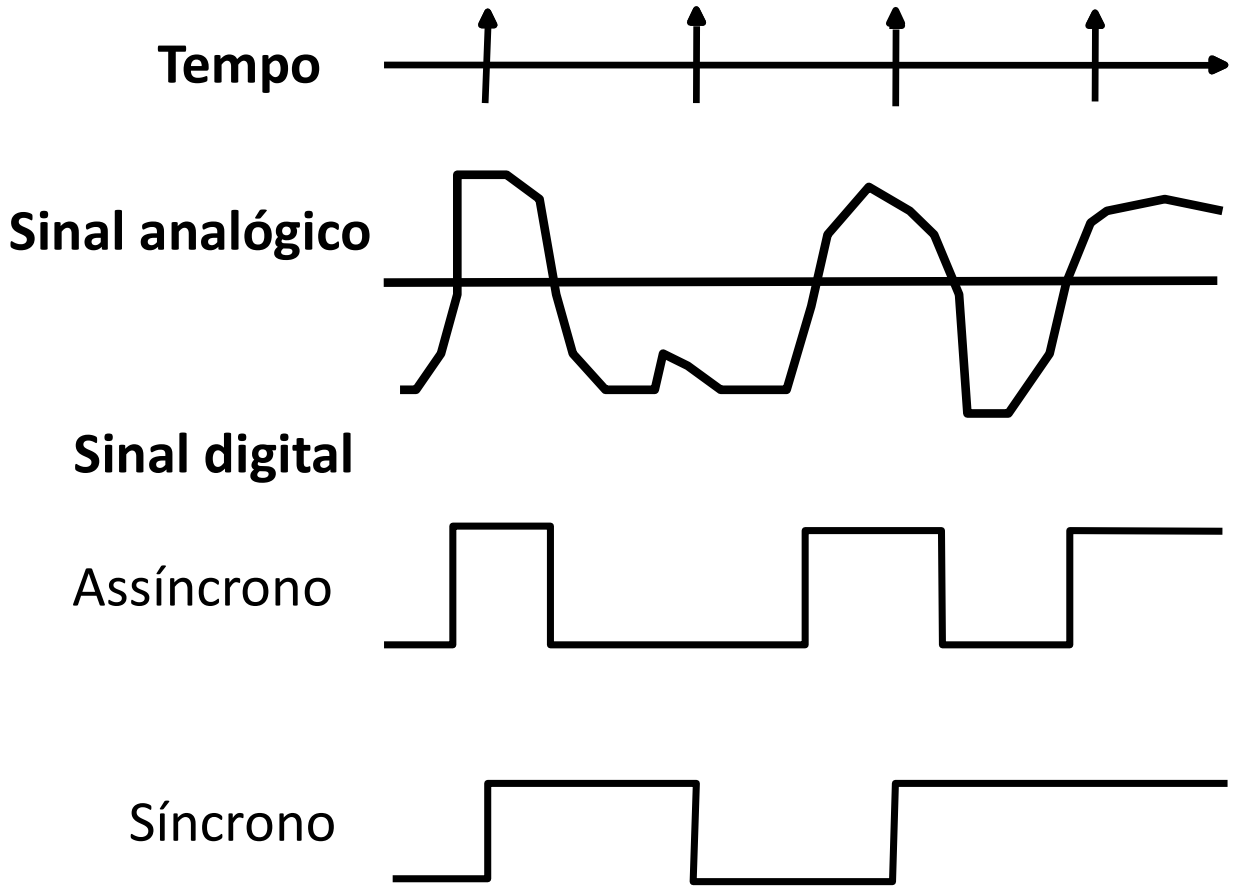
- **Sistema Digital** – sistema no qual os sinais têm um número finito de valores discretos na entrada (estímulos – $X = \{x_1, x_2, ..., x_n\}$) e na saída (respostas – $Y = \{y_1, y_2, ... y_m\}$).



Prof. Misael Morais (moraiscg.uepb@gmail.com)

Representação da Informação

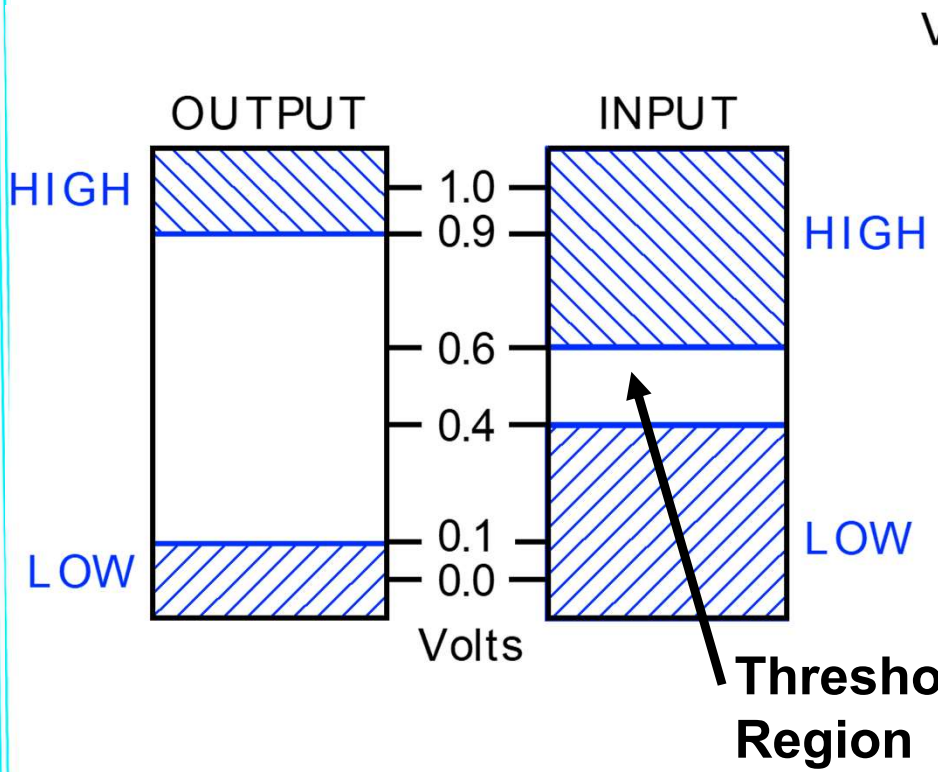
- Exemplos de sinais no domínio do tempo



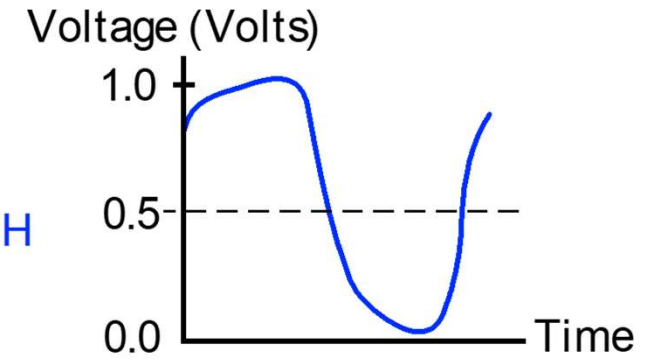
Prof. Misael Morais (moraiscg.uepb@gmail.com)

Representação da Informação

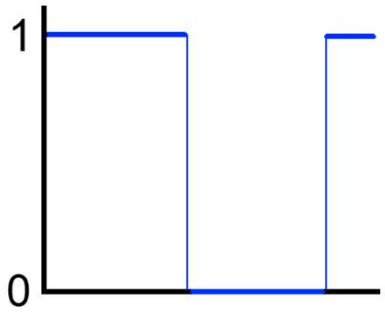
- Exemplos de sinais: quantidade física tensão



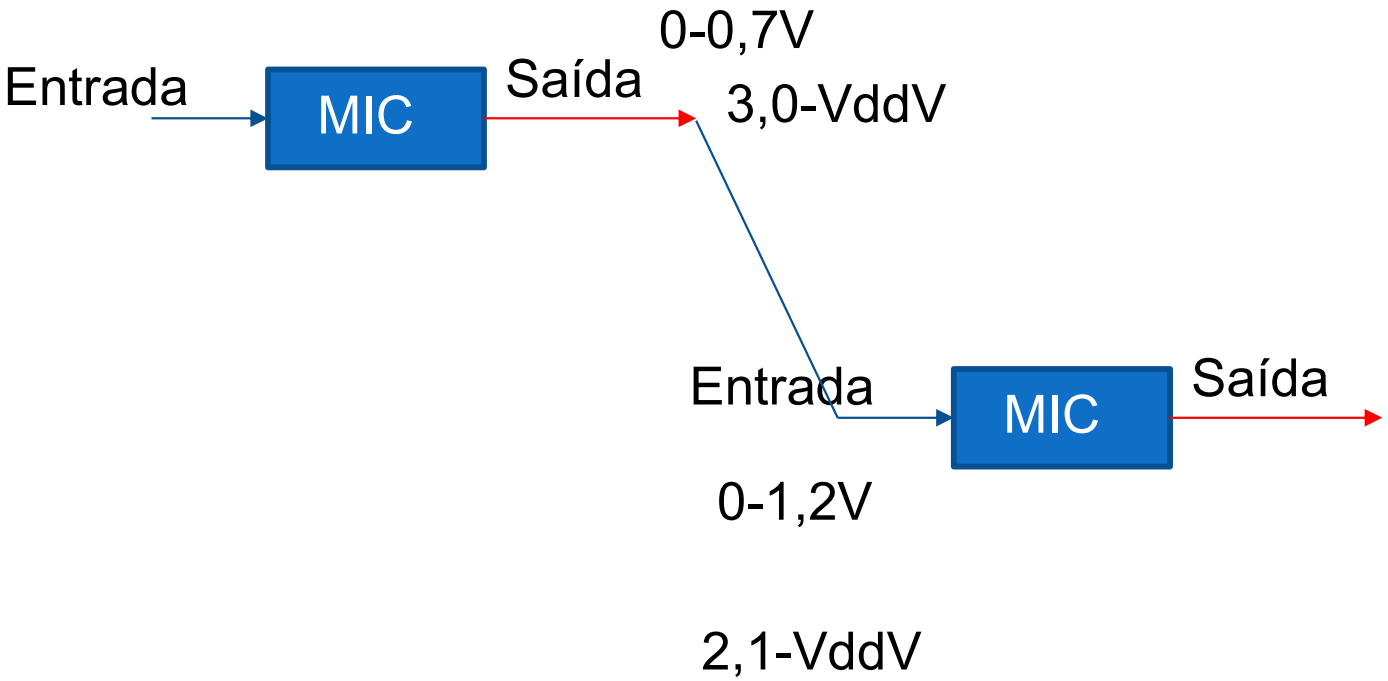
(a) Example voltage ranges



(b) Time-dependent Voltage



(c) Binary model of time-dependent voltage


Entrada
MIC
Saída
0-0,7V
3,0-VddV
Entrada
MIC
Saída
0-1,2V
2,1-VddV

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Representação da Informação

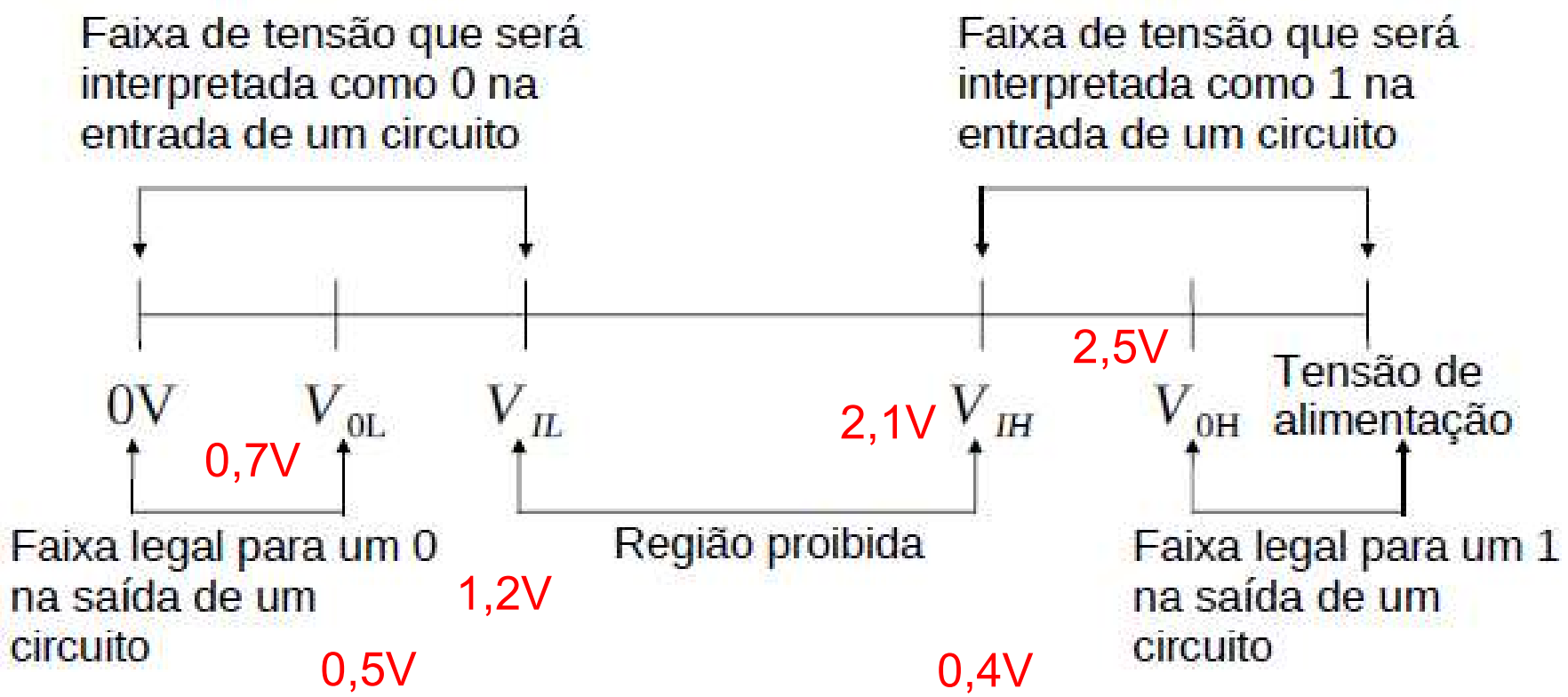
- Outros exemplos de sinais representados por “0” e “1”
  - CPU ⇨ Tensão
  - Disco ⇨ Direção do campo magnético
  - CD ⇨ Superfície Pits/Light
  - RAM Dinâmica ⇨ Carga elétrica

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Interpretação de dados eletronicamente

- De elétrons a bits

A convenção de sinais de um sistema digital determina como os sinais elétricos analógicos são interpretados como valores digitais



Convenções

- V_{DD} (ou V_{CC}) tensão de alimentação.
- V_{OL} tensão mais alta de saída como **zero** lógico.
- V_{OH} tensão mais baixa de saída como **um** lógico.
- V_{IL} tensão mais alta aceita na entrada como **zero** lógico
- V_{IH} tensão mais baixa de aceita na entrada como **um** lógico.
- Os intervalos entre V_{OL} e V_{IL} (ou V_{OH} e V_{IH}) determinam as margens de ruído do sistema digital

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Exercício

Suponha que um sistema digital tenha

V_{DD}=3,3V , V_{IL}=1,2V , V_{0L}=0,7V , V_{IH} 2,1V , V_{0H}=3,0V.

Qual é a margem de ruído para esta convenção de sinais ?

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Sistema de Numeração Posicionais

(O valor atribuído a um símbolo depende da sua posição)

- Base de um Sistema de Numeração Posicional

Quantidade de algarismos disponíveis na representação

Tipos de Sistemas:

- Sistema decimal
- Sistema octal
- Sistema binário
- Sistema hexadecimal

Base 10 – {0,1,2,3,4,5,6,7,8,9,0}
Base 8 – {0,1,2,3,4,5,6,7}
Base 2 – {0,1}
Base 16 – {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

- Dados tratados:
  - 8 bits são chamados de **byte**.
  - 4 bits é chamada de **nibble**.
  - **word**. **double word**. **quad word**.
  - **Overflow, underflow**

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Sistema de Numeração Posicional

$$a = \sum_{i=0}^{n-1} (x_i B^i)$$

$$\mathbf{a} = x_{p-1} \times B^{p-1} + x_{p-2} \times B^{p-2} + x_{p-1} \times B^{p-1} + \ldots + x_0 \times B^o + x_{-1} \times B^{-1} + \ldots + x_{-q} \times B^{-q}$$

Onde:

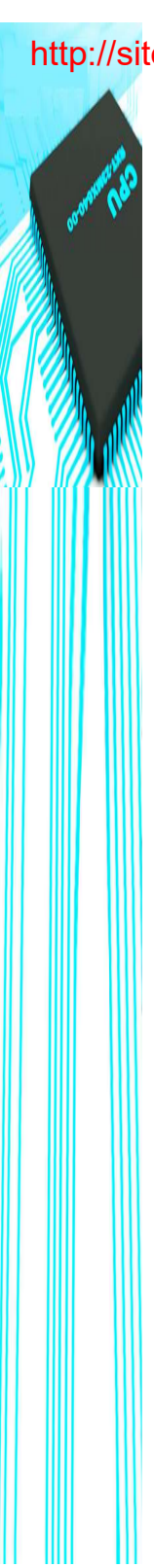
x´s são dígitos do sistema numérico e

B representa a base numérica do sistema

Com:

$$(B - 1) \geq d_u \geq 0$$

Prof. Misael Morais (moraiscg.uepb@gmail.com)



Conversão de bases

- Decimal → Binário
- Binário → Decimal
- Binário → Octal
- Octal → Binário

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Medidas em Byte

Medida	Sigla		Caracteres	
Byte		2^0	1 (8 bits)	1 byte
Kilobyte	**KB**	2^{10}	1.024	1.024 bytes
Megabyte	**MB**	2^{20}	1.048.576	1.024 KBytes
Gigabyte	**GB**	2^{30}	1.073.741.824	1.024 MBytes
Terabyte	**TB**	2^{40}	1.099.511.627.776	1.024 GBytes
Pentabyte	**PB**	2^{50}	1.125.899.906.842.624	1.024 TBytes
Hexabyte	**HB**	2^{60}	1.152.921.504.606.846.976	1.024 PBytes
Yotabyte	**YB**	2^{80}	1.208.925.819.614.630.000.000.000	1.024 Hexabyte

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Conversão de bases

- Conversão de base
  - **$13_{10} = 1101_2$ (conversão)**
  - **$13_{10} \Leftrightarrow$ 0001|0011 (codificação)**
  - **$110\ 010_2 = 62_8$ (**converter entre bases que são potência de dois)
- Converter frações (**pontos da base**)
- Representação sem sinal e com sinal (magnitude com sinal e complementos)
- Complemento -> Complemento é a diferença entre cada algarismo do número e o maior algarismo possível na base. (A representação de números positivos em complemento não tem qualquer alteração, isto é, é idêntica à representação em sinal e magnitude)

Conversão de bases

- Método das divisões

$$a / B = x_{n-1}.B^{n-2} + x_{n-2}.B^{n-3} + \ldots + x_2.B^1 + x_1.B^0$$

com resto igual a x_0

Divide-se pela base. Depois organiza o resto da divisão numa ordem invertida.

Conversão de bases

- Método da substituição direta
  - Conversão de base 2 para base potencia de 2.
  - Conversão pelo peso dos dígitos
  - Base binária para octal ou hexadecimal

$$\underbrace{101}_{5}\underbrace{111}_{7}\underbrace{011}_{3}\underbrace{101}_{5}{}_{2} = 5735_{8}$$

$$\underbrace{1011}_{B}\underbrace{1101}_{D}\underbrace{1101}_{D}{}_{2} = BDD_{16}$$

Conversão de bases

- Método das subtrações

$$a = x_{n-1}.B^{n-1} + x_{n-2}.B^{n-2} + \ldots + x_2.B^2 + x_1.B^1 + x_0.B^0$$

$$a = a - x_{n-1}.B^{n-1} = x_{n-2}.B^{n-2} + \ldots + x_2.B^2 + x_1.B^1 + x_0.B^0$$

- Exemplo: $101101_2 = ?_{10}$ Onde, B = 2, n =6

Portanto:

$$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 32 + 8 + 4 + 1 = 45_{10}$$

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Conversão de bases

Exemplo: $27_8 = ?_{10}$ Onde, B = 8, n = 2

Portanto:

$$2 \times 8^1 + 7 \times 8^0 = 23_{10}$$

Exemplo: $2A5_{16} = ?_{10}$ Onde, B = 16, n = 3

Portanto:

$$2 \times 16^2 + 10 \times 16^1 + 5 \times 16^0 = 512 + 16^0 + 5 = 677^{10}$$

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Aritmética Computacional

- **Aritmética não-decimal**
  - Operação Aritmética sem sinal
  - Operação Aritmética com sinal
  - Inteiros e fracionários

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Aritmética Computacional

- Regra básica:

Soma

- 0 + 0 = 0
- 1 + 0 = 1
- 0 + 1 = 1
- 1 + 1 = 0 e “vai um”

Subtração

- 0 - 0 = 0
- 1 - 0 = 1
- 0 - 1 = 1 e “pede um”
- 1 - 1 = 0

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Aritmética Computacional

- Regra básica:

Multiplicação

- 0 . 0 = 0
- 1 . 0 = 0
- 0 . 1 = 1
- 1 . 1 = 1

Divisão

- 0 / 0 = “erro”
- 1 / 0 = “erro”
- 0 / 1 = 0
- 1 / 1 = 1

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Aritmética Computacional

- Overflow:

  Acontece quando o resultado de uma soma não poder ser armazenada no número de bits disponíveis.

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Aritmética Computacional

- **REPRESENTAÇÃO DE NÚMEROS NEGATIVOS EM COMPLEMENTO A (BASE - 1)**
  - A representação dos números inteiros negativos é obtida efetuando-se: (base - 1) menos cada algarismo do número.
- **REPRESENTAÇÃO DE NÚMEROS NEGATIVOS EM COMPLEMENTO A (BASE - 2)**
  - calcular o complemento a (base -1) e depois soma 1 ao resultado.

Aritmética Computacional

- Sinal magnitude

Sinal	Magnitude

1 = "-"
0 = "+"

$$a = S(a)M(a)$$

Determinação do sinal:

- 1 representa o sinal -
- 0 representa o sinal +

Determinação da magnitude:

- $+7_{10}$(em 4 bits) tem-se: 0 1 1 1 — Sinal: 0; Magnitude: 1 1 1
- -7_{10}(em 4 bits) tem-se: 1 0 0 0 — Sinal: 1; Magnitude: 0 0 0; $B\text{-}1\text{-}x_i$

FAIXA DE REPRESENTAÇÃO

- A representação na base **b** em complemento a base com *n* bits possui b^n representações e permite representar b^n valores.

- A faixa de representação de uma representação na base **2** em complemento a base com *n* bits pode ser calculada como:

  2^n valores (entre -2^{n-1} e $+2^{n-1}-1$), sendo 2 a base.

  O maior valor inteiro positivo será então $+(2^{n-1}-1)$ e o menor valor inteiro negativo será $-(2^{n-1})$.

Prof. Misael Morais (moraiscg.uepb@gmail.com)

RESUMO: Números Inteiros Representados em Complemento 2

- Bit mais significativo ➡ informação de sinal
  (0 = positivo e 1 = negativo)

- números inteiros positivos
  - igual à forma aprendida na conversão decimal/binário

- números inteiros negativos
  - Complemento 1 +1
  - mantém-se os bits menos significativos da direita para a esquerda até à ocorrência do primeiro bit igual a 1 (inclusive), sendo os bits restantes complementados de 1.

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Bit de Paridade

- **Características**
  - Tipos:
    - **Paridade Par** – o número de bits iguais a 1 é par
    - **Paridade Ímpar** – o número de bits iguais a 1 é ímpar
  - Pode detectar erro em um bit (ou múltiplos bits)
  - Exemplo de representação de um dado de 3 bits:
    - Palavra-código **“1111”**: **paridade par**
    - Palavra-código **“1110”: paridade ímpar**
  - MSB, LSB

Número de bits necessários

- **Dados *M* elementos a serem representados por um código binário, o número mínimo de bits, *n*, necessários, satisfaz a relação:**

$$2^n \geq M > 2^{(n-1)}$$

- **Exemplo:** Quantos bits são necessários para representar dígitos decimais em um código binário?

Estouro

- Números em computador são limitados
- Somas (Overflow):
  - Dois operandos tiverem sinais diferente, nunca há overflow.
  - Dois operandos tiverem sinais iguais, e este sinal for diferente do sinal obtido para o resultado.

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Representação de Números Reais

Ponto Flutuante

$$n = f \times 10^{e}$$

- ***f*** - fração ou significando (ou mantissa)
- ***e*** - expoente (inteiro positivo ou negativo)

- Qualquer número (inteiro ou fracionário) pode ser expresso no formato **número x base^{expoente}**, podendo-se variar a posição da vírgula e o expoente.

- Denominação (computacional): **representação em ponto flutuante** (o ponto varia sua posição, modificando, em conseqüência, o valor representado).

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Notação em Ponto Flutuante

Tradicional	Científica	Característica	Mantissa	Expoente
0,234	2,34E-1	2	34	-1
0,054334	5,4334E-2	5	4334	-2
1	1,0E0	1	0	0
10	1,0E1	1	0	1
100	1,0E2	1	0	2
1000	1,0E3	1	0	3
125	1,25E2	1	25	2
...	1,25E56	1	25	56
...	1,25E-56	1	25	-56

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Representação de Números Reais (ponto-flutuante)

Ponto Flutuante

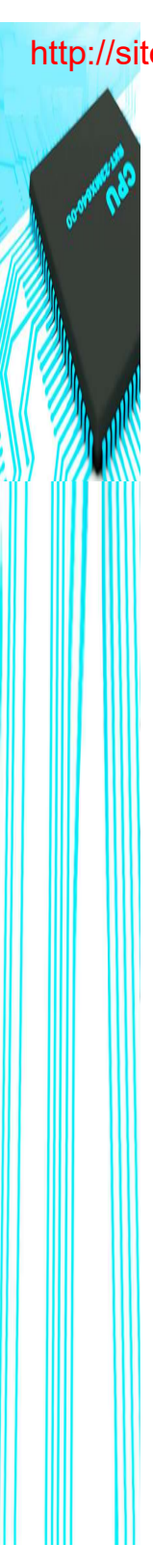
- Representação pode variar ("**flutuar**") a posição da vírgula, ajustando a potência da base.

- **Exemplos:**
  - $3{,}14 = 0{,}314 \times 10^{-1} = 3{,}14 \times 10^{0}$
  - $0{,}000001 = 0{,}1 \times 10^{-5} = 1{,}0 \times 10^{-6}$
  - $1941 = 0{,}1941 \times 10^{4} = 1{,}941 \times 10^{3}$

$$mantissa \times 2^{expoente}$$

- A **faixa de variação** dos números é determinada pela quantidade de dígitos do expoente e a **precisão** é determinada pela quantidade de dígitos do significando.

Prof. Misael Morais (moraiscg.uepb@gmail.com)



PONTO FLUTUANTE EM BINÁRIO

Notação em Ponto Flutuante

- Será que isso funciona com binários?
- 10110000000000000000000000000000b
- Pode ser reescrito assim:

$$\mathbf{1{,}011b * 2^{+31}}$$

- O que significa $\mathbf{* 2^{+31}}$?
- Significa que tenho que mudar a vírgula de lugar para obter o número real
- No caso, andar 31 casas binárias **à direita**

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Notação em Ponto Flutuante

- Será que isso funciona com binários?
- 0,00000000000000000000000001011b
- Pode ser reescrito assim:

$$\mathbf{1{,}011b * 2^{-31}}$$

- O que significa $\mathbf{* 2^{-31}}$?
- Significa que tenho que mudar a vírgula de lugar para obter o número real
- No caso, andar 31 casas binárias **à esquerda**

Notação em Ponto Flutuante

- $1{,}011b * 2^{+31}$
- $1{,}011b * 2^{-31}$

- Os nomes permanecem os mesmos
- Característica
- Mantissa
- Expoente

Notação em Ponto Flutuante

- Vejamos alguns exemplos

Tradicional	Científica	Mantissa	Expoente
100b	1,00b *		2
101b	1,01 * 2		2
11,101b	1,1101		1
1b	1,0		0
0,1001b	1,001b * 2		-1
...	1,0101b * 2^5	01b	56
...	1,0101b * 2^-5	101b	-56

Se é sempre 1b não precisa ser armazenado

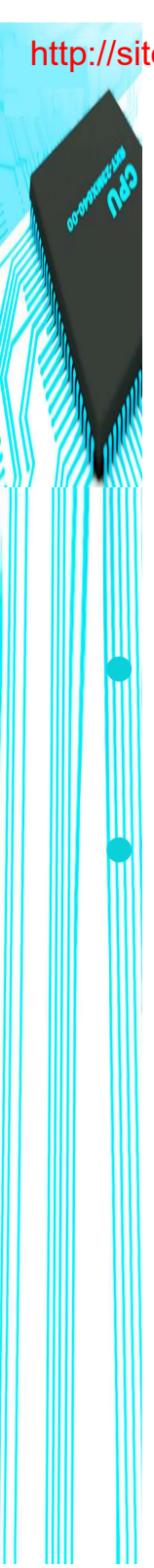
- O que tem de estranho aí?

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Notação em Ponto Flutuante

Fazemos a característica ser sempre 1b, isto é normalizamos

Tradicional	Científica	Mantissa	Expoente
100b	$1,00b * 2^{2}$	00b	2
101b	$1,01 * 2^{2}$	01b	2
11,101b	$1,1101b * 2^{1}$	1101b	1
1b	$1,0b * 2^{0}$	0b	0
0,1001b	$1,001b * 2^{-1}$	001b	-1
...	$1,0101b * 2^{56}$	0101b	56
...	$1,0101b * 2^{-56}$	0101b	-56



Notação em Ponto Flutuante

- **Forma normalizada:** usa um único dígito antes da vírgula, diferente de zero (*).
- Na representação computacional de números em ponto flutuante, a representação normalizada é, em geral, melhor que a não-normalizada.
  - **Forma normalizada**: só existe uma forma de representar um número.
  - **Forma não normalizada**: um mesmo número pode ser representado de diversas maneiras.

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Representação em ponto fixo

- Como representar estes números na memória?
- Em ponto fixo, reservamos um bit para o sinal... E os demais bits para a magnitude

Bit	7 (Sinal)	6	5	4	3	2	1	0
Dígito Binário	0	1	1	0	0	0	0	1

+ ou -
Bit 7: Sinal

0 a 127
Bits 0 a 6: Magnitude

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Representação em ponto-flutuante

- Precisão Simples (32 bits)

Sinal	Sinal do Expoente	Expoente	Mantissa
Bit 31	Bit 30	Bit 29 ~ Bit 23	Bit 22 ~ Bit0

- 1 bit para o sinal do número
- 8 bits para o expoente (incluindo sinal)
- 23 bits para a mantissa

Representação em ponto-flutuante

- Precisão Dupla (64 bits)

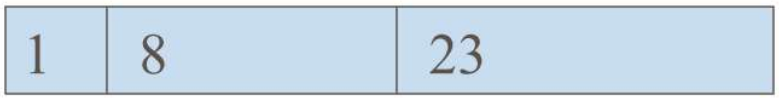
Sinal	Sinal do Expoente	Expoente	Mantissa
Bit 63	Bit 62	Bit 61 ~ Bit 52	Bit 51 ~ Bit0

- 1 bit para o sinal do número
- 11 bits para o expoente (incluindo sinal)
- 52 bits para a mantissa

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Representação em ponto-flutuante

- IEEE 754 especifica diferentes larguras de bits para números em PF.
- Precisão simples



Sinal **Expoente** **Fração**

- Precisão dupla

Sinal **Expoente** **Fração**

(*) Padrão IEEE 754 para números em ponto flutuante – **significando normalizado** – começa com um bit 1, seguido de um ponto (vírgula) binário e pelo resto do significando (número = ± 1,_ _ ... x 2^{exp})

Mantissa normalizada - começa com o ponto (vírgula) binário seguido por um bit 1 e pelo resto da mantissa (bit antes da vírgula igual a zero).

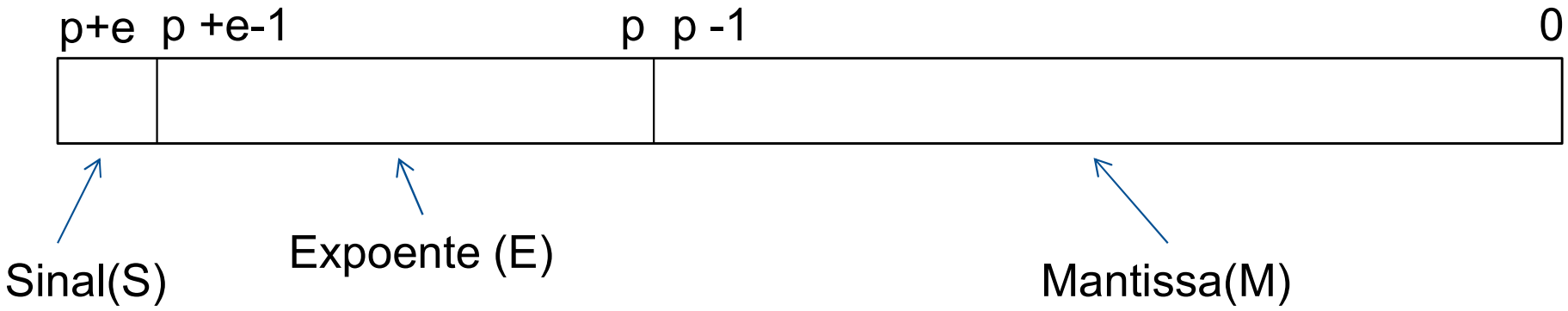
Representação em ponto-flutuante

- **Exemplos:**
  - No número **1,10101x**$(10)^{\mathbf{101}}$:
    - **1,10101 = significando**
    - **101 = expoente**

- **OBS:**
  - A **base binária** não precisa ser explicitada
  - O **"1"** antes da vírgula, na representação normalizada, se esta for adotada, também pode ficar implícito, economizando um bit ("**bit escondido**")

Representação de Números Reais - IEEE

IEEE (*Institute of Electrical and Eletronic Engineers)*



S = Sinal

E = e+excesso 127

E = e+excesso 1023 (precisão dupla)

M = fração normalizada

Norma IEEE 754 – ponto flutuante

- O padrão IEEE 754 para ponto (vírgula) flutuante é a representação mais comum para **números reais** em computadores de hoje, incluindo PC's compatíveis com Intel, Macintosh, e a maioria das plataformas Unix/Linux.

- **O padrão IEEE 754 define três formatos**:
  - **Precisão simples** (32 bits)
  - **Precisão dupla** (64 bits)
  - Precisão estendida (80 bits)

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Padrão IEEE 754 – ponto-flutuante

Precisão	Sinal	Expoente(+/-)	Significando
Simples (32bits)	**1** [bit31]	**8** [bits30-23]	**23** [bits22-00]
Dupla (64 bits)	**1** [bit63]	**11** [bits62-52]	**52** [bits51-00]

- **Sinal**: 0 = + e 1 = -
- **Combinações**: Sinal + Expoente + Significando
- **Notação em excesso de 127**: **precisão simples.**
- **Notação em excesso de 1023**: **precisão dupla.**

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Padrão IEEE 754 – ponto flutuante

- **Exemplo:**

Realize as conversões abaixo:

a) $10{,}875_{10}$ = (?)$_2$ (IEEE 754, com 32 bits)

b) $11000001110100000000000000000000_2$ (IEEE 754, com 32 bits) = (?)$_{10}$

Padrão IEEE 754 – ponto flutuante

Solução:

a) **$10{,}875_{10}$** = $1010{,}111_2$ = $1{,}010111 \times 2^3$

sinal: 0

expoente: $3_{10} + 127_{10} = x_{10}$, $x_{10} = 130_{10} = 10000010_2$

significando: $01011100000000000000000_2$

Resposta:

Número (IEEE 754, com 32 bits):

$01000001001011100000000000000000_2$

Padrão IEEE 754 – ponto flutuante

Solução:

b) $\mathbf{11000001101000000000000000000000_2}$

(IEEE 754, com 32 bits)

sinal: 1

expoente: $10000011_2 = 131_{10}$, e $+127_{10} = 131_{10}$, e = 4_{10}

significando: $10100000000000000000000_2$ =

Resposta:

Número:(negativo) $1{,}101_2 \times 2^4 = 11010_2 = \mathbf{-26_{10}}$

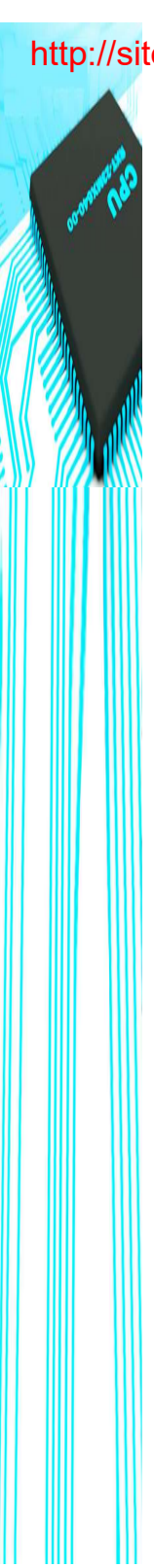
Padrão IEEE 754 – ponto flutuante

- Expoentes na **precisão dupla**
  - **01111111111** (1023_{10}) = expoente zero
  - **00000000001** = menor expoente = –1022 (abaixo de zero)
  - **11111111110** = maior expoente = +1023 (acima de zero)
- **Menor número positivo** (lembre-se do bit escondido e não normalizada)
  - **0 00000000000 00…01** = $2^{-1022} \times 2^{-52} = 2^{-1074}$
- **Maior número positivo** (lembre do bit escondido)
  - **0 11111111110 11...11** = $2^{1023} \times (2-2^{-52})$

Padrão IEEE 754 – ponto flutuante

- A representação em ponto flutuante tem limites de **alcance** e de **precisão**.
- O **alcance** é limitado pelo número de bits do **expoente**.
- A **precisão** é determinada pelo número de bits do **significando**.

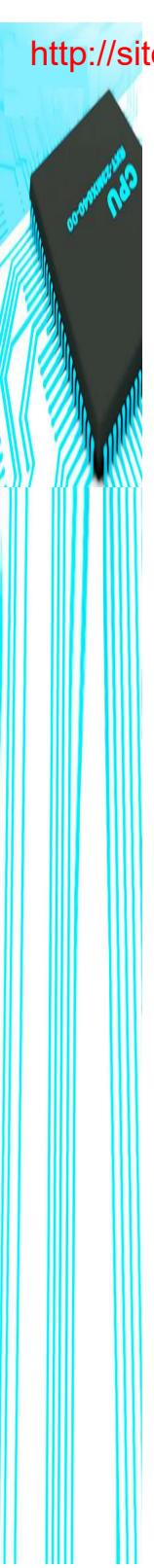
Prof. Misael Morais (moraiscg.uepb@gmail.com)



Representação em Ponto fixo

- **Esse método consiste na determinação de uma posição fixa para a vírgula** (ou ponto).

- Todos os valores representados em ponto fixo para uma determinada operação possuem a mesma quantidade de algarismos inteiros, bem como a mesma quantidade de algarismos fracionários.
  - **Exemplo**: 1101,101 1110,001 0011,110



Representação BCD

- A representação de números reais em ponto flutuante é perfeitamente adequada para fazer cálculos matemáticos, científicos, etc.

- Na representação em ponto flutuante pode-se ter perda de precisão do número representado ou mesmo haverá números que não podem ser representados por ***overflow***.

- Para representação de números em que é necessário manter precisão até o último algarismo, não é admissível erro por aproximação.

- **Solução**: usar a representação **BCD** ou *Binary Coded Decimal* (Decimal Representado em Binário).

Representação BCD

- A idéia do **BCD** é representar, em binário, cada algarismo de forma que o número original seja integralmente preservado.

- A codificação **BCD** não possui extensão fixa, possibilitando representar números com precisão variável - quanto maior o número de bits, maior será a precisão.

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Representação BCD

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Representação BCD

- **Exemplo: 239_{10} = (?) BCD**
  - $2 = 0010_2$
  - $3 = 0011_2$ e
  - $9 = 1001_2$, logo: **239 = 001000111001 (BCD).**

Representação BCD

- A codificação de um dígito em BCD requer 4 bits.
- Como a utilização de apenas 4 bits por byte não é eficiente, normalmente são armazenados 2 dígitos BCD em um só byte. Esta representação é chamada **BCD comprimido ou compactado** ("***packed BCD***").
- Exemplo: **14239_{10} = (?) BCD**

1	42	39	número decimal
xxxx0001	**01000010**	**00111001**	representação BCD comprimido
a+2	a+1	a	endereço

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Representação BCD

- Com esta representação ainda há um desperdício de códigos; como BCD usa 4 bits (16 representações possíveis) para representar 10 algarismos, 6 (ou 4) códigos não são utilizados.

- Portanto, essa representação é menos eficiente em relação à utilização dos recursos do computador que a **representação em ponto flutuante**.

Representação Interna de Caracteres

- ASCII
- Código de 6 *bits*.
- Código de 7 *bits* (ASCII).
- EBCDIC (*Extended Binary Coded Decimal Interchange Code*).
- ASCII Estendido.
- ISO Latin-1.
- Caracteres ANSI.
- Caracteres UNICODE.
- Código Gray.....

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Representação Interna de Caracteres

- **Na maioria das linguagens de programação e nos sistemas de computação (e os compiladores da maior parte das linguagens de programação) emprega-se**:
  - a representação de números em ponto fixo para indicar valores inteiros (a vírgula fracionária é assumida na posição mais à direita do número);
  - números fracionários são representados apenas em ponto flutuante.

Códigos

- Alguns exemplos:

Decimal	8,4,2,1	Excesso de 3	8,4,-2,-1	Gray
0	0000	0011	0000	0000
1	0001	0100	0111	0001
2	0010	0101	0110	0011
3	0011	0110	0101	0010
4	0100	0111	0100	0110
5	0101	1000	1011	0111
6	0110	1001	1010	0101
7	0111	1010	1001	0100
8	1000	1011	1000	1100
9	1001	1100	1111	1000

Representação por Código Gray

- É uma forma de código binário
- O código Gray é utilizado para evitar erros quando ocorrem transições para o próximo estado.
- Na passagem de um valor para outro sucessivo ou antecedente apenas um bit ou dígito muda.

Exemplo: Código Gray

Decimal	Código Gray
0	0 0 0
1	0 0 1
2	0 1 1
3	0 1 0
4	1 1 0
5	1 1 1
6	1 0 1
7	1 0 0

Todas as codificações consecutivas são adjacentes

Prof. Misael Morais (moraiscg.uepb@gmail.com)

Representação de caracteres

- UTF: Unicode Transformation Format
  - UTF-8: 256 caracteres
  - UTF-16: 65536 caracteres
  - UTF-32: 4 bilhões de caracteres
- UTF-8 é compatível com os 128 primeiros caracteres do ASCII
- UTF-16 é compatível com UTF-8
- UTF-32 é compatível com UTF-16

Outras representações (Códigos)

- ***ASCII*** *(American Standard Code for Information Interchange)*
  - Padrão definido pela organização ANSI.
  - Código de 7 bits (128 combinações de caracteres).
  - ASCII Estendido (utiliza outros 128 códigos para símbolos gráficos, e línguas diferentes do inglês).
    - $0\text{-}9_{10}$ ➡ $30 - 39_{16}$
    - A-Z ➡ $41 - 5A_{16}$
    - a-z $61 - 7A_{16}$

- ***UNICODE***
  - Oferece 2 bytes para a representação de símbolos (mais de 65.000 símbolos).

ASCII - controle

Decimal	Binário	Hex	Referência
0	00000000	00	Null - NUL
1	00000001	01	Start of Heading - SOH
2	00000010	02	Start of Text - STX
3	00000011	03	End of Text - ETX
4	00000100	04	End of Transmission - EOT
5	00000101	05	Enquiry - ENQ
6	00000110	06	Acknowledge - ACK
7	00000111	07	Bell, rings terminal bell - BEL
8	00001000	08	BackSpace - BS
9	00001001	09	Horizontal Tab - HT
10	00001010	0A	Line Feed - LF
11	00001011	0B	Vertical Tab - VT
12	00001100	0C	Form Feed - FF
13	00001101	0D	Enter - CR
14	00001110	0E	Shift-Out - SO
15	00001111	0F	Shift-In - SI
16	00010000	10	Data Link Escape - DLE
17	00010001	11	Device Control 1 - D1
18	00010010	12	Device Control 2 - D2
19	00010011	13	Device Control 3 - D3

Decimal	Binário	Hex	Referência
20	00010100	14	Device Control 4 - D4
21	00010101	15	Negative Acknowledge - NAK
22	00010110	16	Synchronous idle - SYN
23	00010111	17	End Transmission Block - ETB
24	00011000	18	Cancel line - CAN
25	00011001	19	End of Medium - EM
26	00011010	1A	Substitute - SUB
27	00011011	1B	Escape - ESC
29	00011101	1D	Group Separator - GS
30	00011110	1E	Record Separator - RS
31	00011111	1F	Unit Separator - US

Prof. Misael Morais (moraiscg.uepb@gmail.com)

ASCII – Normal

Decimal	Binário	Hex	Referência
32	00100000	20	Space - SPC
33	00100001	21	!
34	00100010	22	"
35	00100011	23	#
36	00100100	24	$
37	00100101	25	%
38	00100110	26	&
39	00100111	27	'
40	00101000	28	(
41	00101001	29	)
42	00101010	2A	*
43	00101011	2B	+
44	00101100	2C	,
45	00101101	2D	-
46	00101110	2E	.
47	00101111	2F	/
48	00110000	30	0
49	00110001	31	1
50	00110010	32	2
51	00110011	33	3
52	00110100	34	4
53	00110101	35	5
54	00110110	36	6
55	00110111	37	7
56	00111000	38	8
57	00111001	39	9

Decimal	Binário	Hex	Referência
58	00111010	3A	:
59	00111011	3B	;
60	00111100	3C	<
61	00111101	3D	=
62	00111110	3E	>
63	00111111	3F	?
64	01000000	40	@
65	01000001	41	A
66	01000010	42	B
67	01000011	43	C
68	01000100	44	D
69	01000101	45	E
70	01000110	46	F
71	01000111	47	G
72	01001000	48	H
73	01001001	49	I
74	01001010	4A	J
75	01001011	4B	K
76	01001100	4C	L
77	01001101	4D	M
78	01001110	4E	N
79	01001111	4F	O
80	01010000	50	P
81	01010001	51	Q
82	01010010	52	R
83	01010011	53	S

Prof. Misael Morais (moraiscg.uepb@gmail.com)

ASCII – Normal (cont.)

Decimal	Binário	Hex	Referência
83	01010011	53	S
84	01010100	54	T
85	01010101	55	U
86	01010110	56	V
87	01010111	57	W
88	01011000	58	X
89	01011001	59	Y
90	01011010	5A	Z
91	01011011	5B	[
92	01011100	5C	\
93	01011101	5D	]
94	01011110	5E	^
95	01011111	5F	_
96	01100000	60	`
97	01100001	61	a
98	01100010	62	b
99	01100011	63	c
100	01100100	64	d
101	01100101	65	e
102	01100110	66	f
103	01100111	67	g
104	01101000	68	h
106	01101010	6A	j
107	01101011	6B	k
108	01101100	6C	l
109	01101101	6D	m
110	01101110	6E	n

Decimal	Binário	Hex	Referência
111	01101111	6F	o
112	01110000	70	p
113	01110001	71	q
114	01110010	72	r
115	01110011	73	s
116	01110100	74	t
117	01110101	75	u
118	01110110	76	v
119	01110111	77	w
120	01111000	78	x
121	01111001	79	y
122	01111010	7A	z
123	01111011	7B	{
124	01111100	7C	|
125	01111101	7D	}
126	01111110	7E	~
127	01111111	7F	Delete
128	10000000	80	Ç
129	10000001	81	ü
130	10000010	82	é
131	10000011	83	â
132	10000100	84	ä
133	10000101	85	à
134	10000110	86	å
135	10000111	87	ç
136	10001000	88	ê
137	10001001	89	ë