

# MC404

---

## ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

2010

**Prof. Paulo Cesar Centoducatte**

**Prof. Mario Lúcio Côrtes**

**Prof. Ricardo Pannain**

# MC404

---

## ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

### “Conceitos Básicos”

# Programa do Curso e Conceitos Básicos

## Sumário

---

- **Porque um Curso de Linguagem de Montagem**
- **Conceitos Básicos**
  - Processadores
  - Bits e Bytes
  - Little Endian e Big Endian
  - Memória
  - Representação de números com e sem sinal
  - Conversão entre bases numéricas

# Porque um Curso de Linguagem de Montagem

---

- Permite compreender o funcionamento de uma CPU
- Utilizado na:
  - Programação de máquinas baseadas em micro-controladores.
  - Programação de sistemas embarcados (embedded systems)
  - Programação de trechos críticos (tempo e/ou memória)
  - Acesso a recursos não disponíveis em alto nível
- **OBS.:** A linguagem de montagem é absolutamente ligada ao hardware, depende de cada máquina específica (diferentemente das linguagens de alto nível)

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

---

## Conceitos Básicos

# Processadores

---

- **Máquinas para manipular informações**
- **Como representar as informações em um processador**
  - **Associando-as a uma grandeza física que possamos gerar, manipular, armazenar, ler etc**
  - **Exemplos de grandezas: comprimento, posição angular, força, pressão, fluxo, **tensão**, corrente etc**
- **Máq. Analógicas X Máq. Digitais**
  - **Analógicas: Informação associada a uma grandeza continua**
  - **Digital: Informação associada a uma grandeza discreta**
    - **Normalmente um pequeno conjunto de valores (ex. 2), da grandeza escolhida para representar a informação, distintos e facilmente controláveis e distinguíveis.**

# Processadores

---

- A informação é representada de forma digitalizada, ié representada como um número na base B, igual ao número de estados distintos, usados na representação da informação.
- Utiliza-se tantos componentes digitais quantos necessários para representar a informação desejada.
  - Ex. Para representar os valores de 0 a 1000, usando base 2 são necessários 10 componentes digitais – são necessários 1001 combinações diferentes dos estados dos componentes digitais
- **OBS.:**
  - Base: número de símbolos distintos usados para representar a informação (1, 2, 8, 10, 16 etc)
  - Símbolos normalmente usados:
    - Base até 10: dígitos ou algarismos arábicos
    - Base maior que 10: acrescenta-se letras (de preferencia maiúsculas), ex Hexadecimal: 0 1 2 3 4 5 6 7 8 9 A B C D E F

# Conceitos Básicos

## BITS e BYTES

- Bit = BInary digiT = vale sempre 0 ou 1. Elemento básico de informação
- Byte = 8 bits processados em paralelo (ao mesmo tempo)
- Word = n bytes (depende do processador em questão)
- Double word = 2 words
- Nibble = 4 bits (útil para BCD)
  
- Posição dos bits:

Para 1 byte:

7	6	5	4	3	2	1	0
0	1	0	1	0	1	0	1

Para 1 word (de 16 bits):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

high byte | low byte

# Conceitos Básicos

---

## Little Endian X Big Endian

Words são armazenados em bytes consecutivos, em memórias de largura de 8 bits.

Exemplo:

$$1025_{10} = 00000000\ 00000000\ 00000100\ 00000001_2$$

Endereço	Representação Big-Endian (MOTOROLA)	Representação Little-Endian (INTEL)
00	00000000	00000001
01	00000000	00000100
02	00000100	00000000
03	00000001	00000000

# Conceitos Básicos

## Memória

- **Memória:** local do computador (hardware) onde se armazenam temporária ou definitivamente dados (números, caracteres e instruções)
- **Posição de memória ou endereço:** localidade física da memória onde se encontra o dado.
- **Organização da memória:**

Endereço	Conteúdo
...	...
4MB	10110101
...	...
1048576	01001010
...	...
1765	01001101
...	...
4	01010000
3	11111111
2	11101001
1	11011010
0	01100100

# Conceitos Básicos

## Representação binária de números não sinalizados

Qualquer número em qualquer base  $\rightarrow N = \sum_{i=0}^{n-1} d_i \times \text{base}^i$

a) 1 byte

$$\begin{aligned} 00100111_2 &= 0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 0 + 0 + 32 + 0 + 0 + 4 + 2 + 1 = 39_{10} \\ &= 27_{16} \end{aligned}$$

b) 1 word

$$\begin{aligned} 0101011101101110_2 &= 0 \times 2^{15} + 1 \times 2^{14} + \dots + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 22382_{10} \\ &= 576E_{16} \text{ (mais fácil de representar!)} \\ \text{high byte} &= 0101\ 0111b = 57_{16} \\ \text{low byte} &= 0110\ 1110b = 6E_{16} \end{aligned}$$

# Conceitos Básicos

---

## Representação binária de números não sinalizados

Qualquer número em qualquer base  $\rightarrow N = \sum_{i=0}^{n-1} d_i \times base^i$

- **Caso Particulares de Bases:**
  - **B = 1 - Unário - tem utilidade ?**
  - **B = 2 - Binário**
  - **B = 8 - Octal**
  - **B = 10 - Decimal**
  - **B = 16 - Hexadecimal**

# Conceitos Básicos

## Conversão entre bases numéricas

<b>Tipo de conversão</b>	<b>Procedimento</b>
<b>Decimal → Binário</b>	<b>Divisões sucessivas por 2 até se obter zero no quociente. Leitura dos dígitos binários de baixo para cima.</b>
<b>Binário → Decimal</b>	<b>Soma de potências de 2 cujo expoente é a posição do bit e cujo coeficiente é o próprio bit.</b>
<b>Hexadecimal → Binário</b>	<b>Expandir cada dígito hexa em quatro dígitos binários segundo seu valor.</b>
<b>Binário → Hexadecimal</b>	<b>Compactar cada quatro dígitos binários em um único dígito hexa segundo seu valor.</b>
<b>Decimal → Hexadecimal</b>	<b>Divisões sucessivas por 16 até se obter zero no quociente; leitura dos dígitos de baixo para cima.</b>
<b>Hexadecimal → Decimal</b>	<b>Soma de potências de 16 cujo expoente é a posição do dígito e cujo coeficiente é o valor do próprio dígito hexa.</b>

# Conceitos Básicos

## Representação binária de números sinalizados

- Representação com sinal e magnitude
  - O bit mais significativo é o sinal do número → se for 1 o número é negativo se for 0 o número é positivo

Exemplo 1:  $01110001_2$

$$\begin{aligned}\text{valor não sinalizado} &= 0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + \\ &+ 0 \times 2^1 + 1 \times 2^0 = \\ &= 64 + 32 + 16 + 1 = 113_{10}\end{aligned}$$

$$\begin{aligned}\text{valor sinalizado} \quad \text{bit de sinal} = 0 &\Rightarrow \text{" + " (positivo)} \\ &= 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + \\ &= 0 \times 2^1 + 1 \times 2^0 = \\ &= 64 + 32 + 16 + 1 = 113_{10} \rightarrow \text{logo} = +113_{10}\end{aligned}$$

# Conceitos Básicos

---

Exemplo 2:  $10110001_2$

valor não sinalizado

$$\begin{aligned} &= 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + \\ &+ 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = \\ &= 128 + 32 + 16 + 1 = 177_{10} \end{aligned}$$

valor sinalizado

bit de sinal = 1  $\Rightarrow$  " - " (negativo)

$$\begin{aligned} &= 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + \\ &+ 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 32 + 16 + 1 = 49_{10} \rightarrow \text{logo} = -49_{10} \end{aligned}$$

# Conceitos Básicos

---

**Exemplo 3:**

$$70FF_{16} = 0111000011111111_2$$

$$\text{valor não sinalizado} = 0 \times 2^{15} + 1 \times 2^{14} + \dots + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

**valor sinalizado** → bit de sinal = 0 ⇒ " + " (positivo)

$$= + (0 \times 2^{15} + 1 \times 2^{14} + \dots + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)$$

**Exemplo 4:**

$$C777_{16} = 1100011101110111_2$$

$$\text{valor não sinalizado} = 1 \times 2^{15} + 1 \times 2^{14} + \dots + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

**valor sinalizado** → bit de sinal = 1 ⇒ " - " (negativo)

$$= - (1 \times 2^{14} + \dots + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)$$

# Conceitos Básicos

---

## Representações possíveis de números sinalizados

- Complemento de 1

-  $X = (2^n - 1) - X$      $n$  é o número de bits utilizados na representação

- Complemento de 2

-  $X = 2^n - X$      $n$  é o número de bits utilizados na representação

# Conceitos Básicos

---

- Representações possíveis de números sinalizados

Sinal e Magnitude	Complemento de 1	Complemento de 2
000 = +0	000 = +0	000 = +0
001 = +1	001 = +1	001 = +1
010 = +2	010 = +2	010 = +2
011 = +3	011 = +3	011 = +3
100 = -0	100 = -3	100 = -4
101 = -1	101 = -2	101 = -3
110 = -2	110 = -1	110 = -2
111 = -3	111 = -0	111 = -1

- Representação em Complemento de 2 → utilizada pois temos apenas uma representação para o zero e podemos fazer a soma e subtração com apenas um circuito.

# Conceitos Básicos

- Números sinalizados de 32 bits, em Complemento de 2:

$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = 0_{10}$$

$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = +1_{10}$$

$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2 = +2_{10}$$

...

$$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = +2,147,483,646_{10}$$

$$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = +2,147,483,647_{10}$$

$$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = -2,147,483,648_{10}$$

$$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = -2,147,483,647_{10}$$

$$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2 = -2,147,483,646_{10}$$

...

$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_2 = -3_{10}$$

$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = -2_{10}$$

$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = -1_{10}$$

— *maxint*

— *minint*

# Conceitos Básicos

---

## Representação em Complemento de 2 de um número:

- Partindo-se da representação do negativo do valor a ser achado, nega-se este número (negar  $\rightarrow$  inverter) e somar 1

### Exemplo 1:

-5 em Complemento de 2 (usando-se 5 bits para a sua representação)

Partindo-se da representação do  $5_{10} = 00101_2 \rightarrow$  (invertendo os bits) =  $11010 \rightarrow$  (somando 1) =  $11011_2 = -5$  em Complemento de 2

### Exemplo 2:

+5 em Complemento de 2 (usando-se 5 bits para a sua representação)

Partindo-se da representação do  $-5_{10} = 11011_2 \rightarrow$  (invertendo os bits) =  $00100_2 \rightarrow$  (somando 1) =  $00101_2 = +5$  em Complemento de 2

# Conceitos Básicos

---

- **Conversão de números com n bits em números com mais que n bits:**
  - **copiar o bit mais significativo (bit de sinal) nos outros bits (extensão do sinal):**

**Exemplo:**

**0010 → 0000 0010**

**1010 → 1111 1010**

# Conceitos Básicos

## Operações de soma e adição binárias

- Como aprenderam no primeiro grau: (vai-um/vem-um)

$$\begin{array}{r} 0111 \text{ (7)} \\ + 0110 \text{ (6)} \\ \hline 1101 \text{ (13)} \end{array} \quad \begin{array}{r} 0111 \text{ (7)} \\ - 0110 \text{ (6)} \\ \hline 0001 \text{ (1)} \end{array} \quad \begin{array}{r} 0110 \text{ (6)} \\ - 0101 \text{ (5)} \\ \hline 0001 \text{ (1)} \end{array}$$

- Subtração em complemento de 2 é feito como se fosse uma soma ( $A - B = A + (-B)$ ):
  - subtração usando adição de números negativos

$$\begin{array}{r} 0111 \text{ (=+7)} \\ + 1010 \text{ (= -6)} \\ \hline 1| 0001 \text{ (=1)} \end{array}$$

# Conceitos Básicos

---

## Overflow

- **Overflow (resultado maior (menor) que a palavra do computador pode representar):**

**Exemplo:**

- **Quando na operação abaixo ocorre e quando não ocorre overflow ???**

$$\begin{array}{r} 0111 \text{ (7) ou (+7)} \\ + 0001 \text{ (1) ou (+1)} \\ \hline 1000 \end{array}$$

# Conceitos Básicos

---

## Detecção de Overflow

- Não existe overflow quando adicionamos um número positivo e um negativo
- Não existe overflow quando os sinais dos números são os mesmos na subtração
- Ocorre overflow quando os valores afetam o sinal:
  - Somando dois números positivos dá um número negativo
  - Somando dois números negativos dá um número positivo
  - Subtrai um número negativo de um positivo e dá negativo
  - Subtrai um número positivo de um negativo e dá positivo

### Exercício

- Considere as operações  $A + B$  e  $A - B$ 
  - Pode ocorrer overflow se  $B = 0$  ?
  - Pode ocorrer overflow se  $A = 0$  ?

# Conceitos Básicos

---

## Multiplicação Binária

- Exemplo:

1010 X 101

```
      1 0 1 0
      X 1 0 1
      -----
      1 0 1 0
      0 0 0 0
      1 0 1 0
      -----
      1 1 0 0 1 0
```

# Conceitos Básicos

---

## Divisão Binária

- Exemplo:

1 1 0 0 1 0 / 1 0 1

$$\begin{array}{r} 110010 \quad \overline{)101} \\ - 101 \phantom{0000} \\ \hline 00101 \\ - \phantom{0}101 \\ \hline 0000 \end{array}$$

# Conceitos Básicos

## Representação de Caracteres Alfanuméricos

- Tabela ASCII (American Standard Code Interchange Information)

Exemplo:

64	@		96	`
65	A		97	a
66	B		98	b
67	C		99	c
68	D		100	d
69	E		101	e
70	F		102	f
71	G		103	g
72	H		104	h
73	I		105	i

48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9

# Conceitos Básicos

---

## Linguagem de programação

- Linguagem de Alto Nível - próximo ao ser humano, escrita de forma textual.
  - Ex: `if (a==b) a=b+c;`
- Linguagem de Montagem (Assembly) - próximo à linguagem de máquina, escrita em códigos (mnemônicos)
  - Ex: `ADD AX,BX;`
- Linguagem de Máquina - linguagem que o computador consegue executar - códigos binários
  - Ex: `01010001`

# Conceitos Básicos

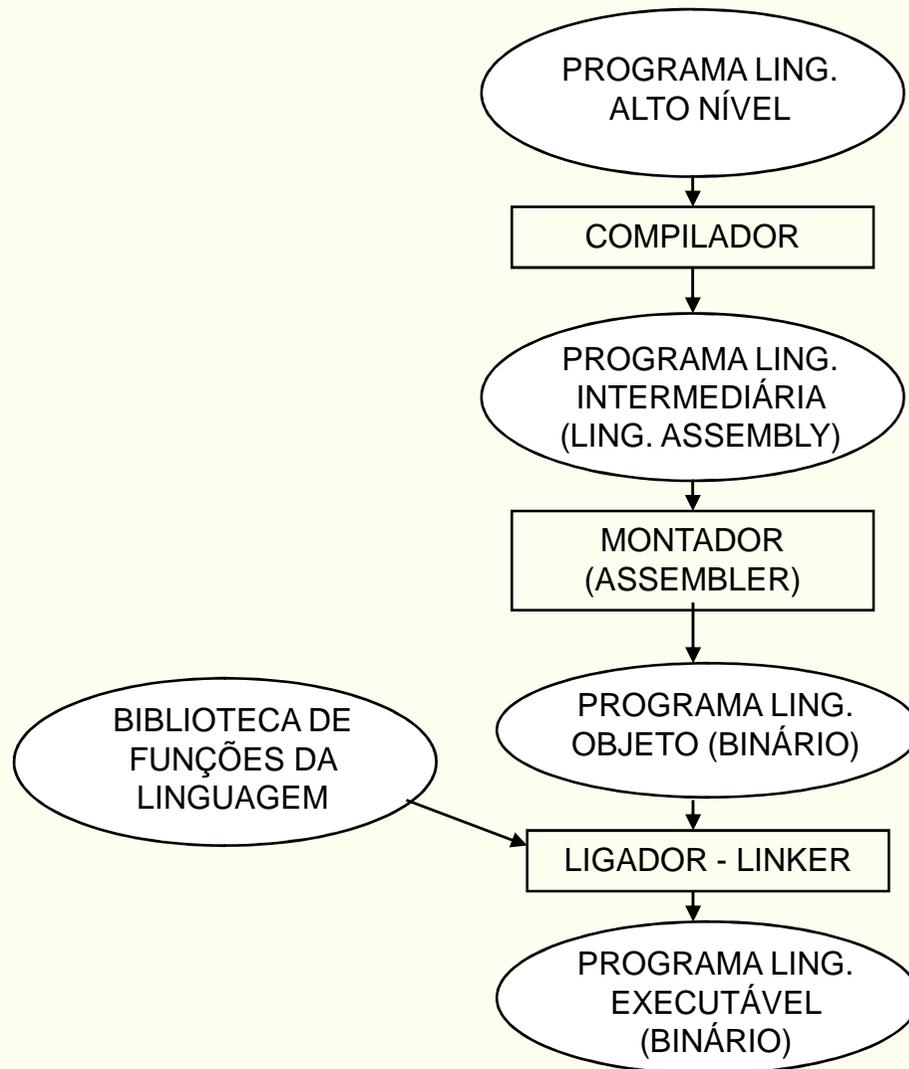
---

## Execução de um programa

- **Um programa escrito em linguagem de alto nível, para ser executado ele deve:**
  - **Ser traduzido para linguagem de máquina (compiladores, montadores, ligadores);**
  - **Ter seus endereços realocados, conforme posição onde será carregado na memória (loaders);**
  - **Ser alocado em um região da memória (loaders).**

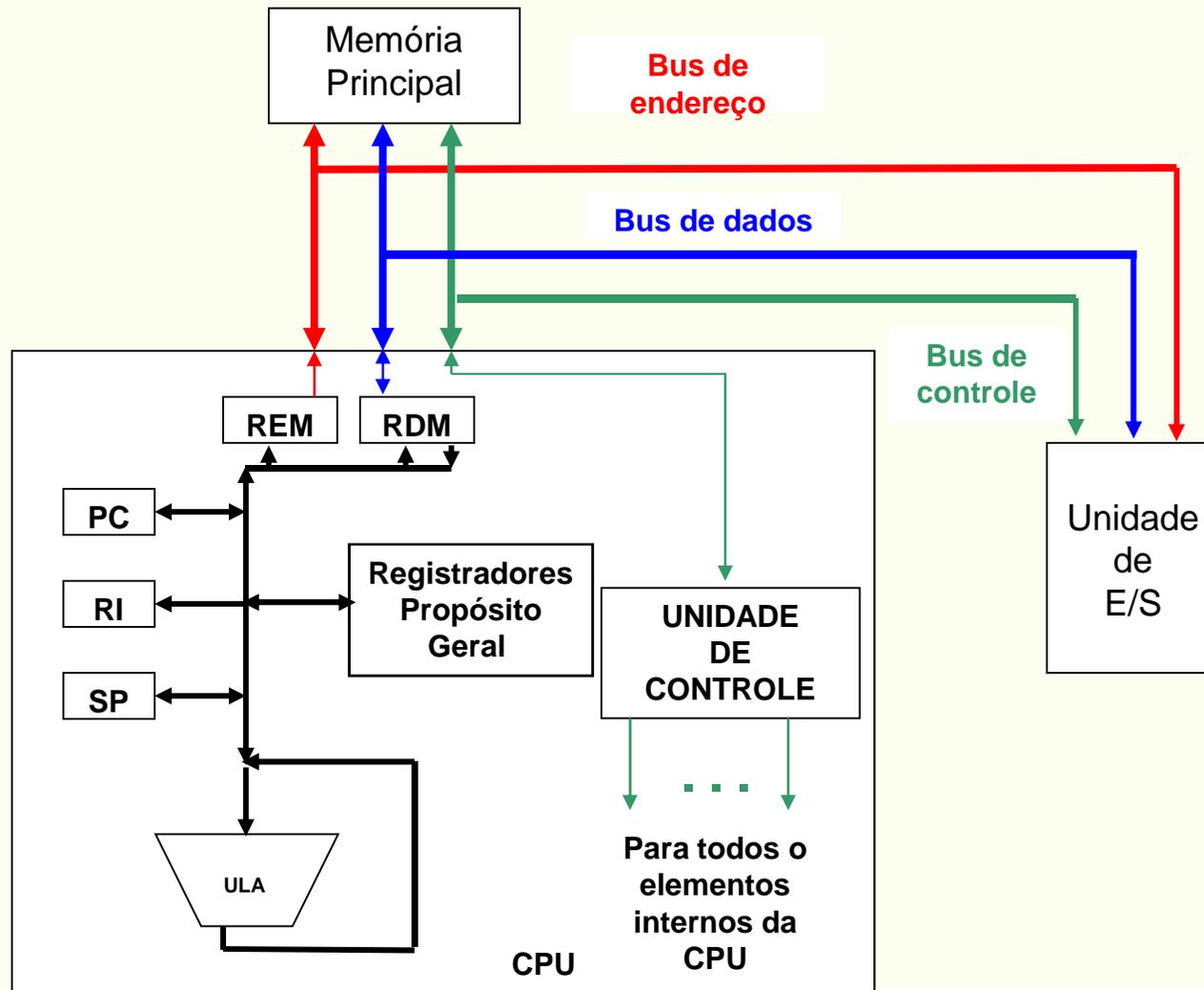
# Conceitos Básicos

## Processo de tradução de um programa em linguagem de alto nível



# Conceitos Básicos

## Organização Básica de um Computador Digital



# Conceitos Básicos

## Organização Básica de um Computador Digital

- **Unidade Central de Processamento – CPU:**
  - Unidade de Controle – UC;
  - Unidade Lógica e Aritmética – ULA;
  - Registradores de Propósito Geral – GPR;
  - Registradores Específicos.
- **Unidade de Memória → hierarquia de memória:**
  - Memória Principal;
  - Memória Secundária;
- **Unidade de Entrada e Saída:**
  - Interfaces;
  - Canais de E/S;
  - Processadores E/S.
- **Barramentos:**
  - Barramento de Endereços;
  - Barramento de Dados;
  - Barramento de Controle.

# Conceitos Básicos

---

## Organização Básica de um Computador Digital

- **Unidade Central de Processamento – CPU**
  - **Responsável por todo o processamento (execução de programas) no sistema**
    - **Unidade de Controle: circuito que gera os sinais de controle responsáveis pelo gerenciamento (controle) de todas as atividades do computador.**
    - **Unidade Lógica e Aritmética – ULA: circuito responsável por efetuar todas as operações lógicas e aritméticas.**
    - **Registradores de Propósito Geral – GPR: elementos de memória (circuitos) responsáveis por armazenar os dados que são utilizados durante a execução de um programa (instruções).**

# Conceitos Básicos

---

## Organização Básica de um Computador Digital

- **Unidade Central de Processamento – CPU (cont.)**
  - **Registradores Específicos:**
    - **Program Counter – PC:** armazena o endereço da próxima instrução a ser executada;
    - **Stack Pointer – SP:** armazena o endereço do topo da pilha;
    - **Registrador de Instrução – RI:** armazena a instrução que está sendo executada;
    - **Registrador de Dados de Memória – RDM:** armazena os dados que vem da memória (lidos) ou que vão para a memória (escritos);
    - **Registrador de Endereços de memória – REM:** armazena o endereço enviado para a memória, quando ocorrer um acesso à mesma (leitura ou escrita)

# Principais Componentes de um Sistema Computacional

## Sumário

---

- **Memórias**
- **CPUS**
- **Unidades de I/O (periféricos de entrada e saída)**

# Memória

---

- **Memórias: Dispositivo capaz de armazenar informação para posterior recuperação**
  - **OBS. Conceito bastante antigo**
    - Saco com pedrinhas ou sementes
    - Marcas em tijolos de argila
    - Escrita em papiros/papel
    - Marcas com “tintas” em cavernas
    - Etc
- **Anológicas/Digitais**
  - Contador de pontos em bilhar
  - Disco de vinil (som)
  - K7
  - CD
  - HD

# Memória

---

- **Memórias Digitais (bits) implementadas como circuitos eletrônicos integrados - “CI’s de memória”**
- **Operações suportadas por uma memória:**
  - **Entrada de dados: escrita (gravação, apagamento, programação) - write**
  - **Saída de dados: leitura – read**
- **Memória simples**
  - **Flip-Flop**
    - **Escrita de “1” - set**
    - **Escrita de “0” – reset**
    - **Leitura – verificar se o FF está set ou reset**

# Memória

---

- **Memórias (reais)**
  - Podem ser vistas como grandes vetores de bits
  - Capacidade
    - N bits
    - N/8 Bytes
  - Abreviações normalmente utilizadas para expressar a capacidade de uma memória:
    - K (kilo) =  $2^{10} = 1024 \sim 1000 = 10^2$
    - M (mega) =  $2^{20} = 1.048.578 \sim 10^6$
    - G (giga) =  $2^{30} \sim 10^9$
    - T (tera) =  $2^{40} \sim 10^{12}$

# Memória

---

- **OBS.:**
  - **Memórias normalmente utilizam  $2^N$  devido ao espaço de endereçamento das CPU's**
  - **HD's tendem a utilizarem  $10^N$**
  - **Quando falarmos em capacidade de memória usaremos  $2^N$**
- **Palavras de Memória (words)**
  - **Números de bits transferidos em paralelo, em uma operação (leitura ou escrita)**
  - **Expresso em bits ou Bytes -- largura da palavra (word size)**

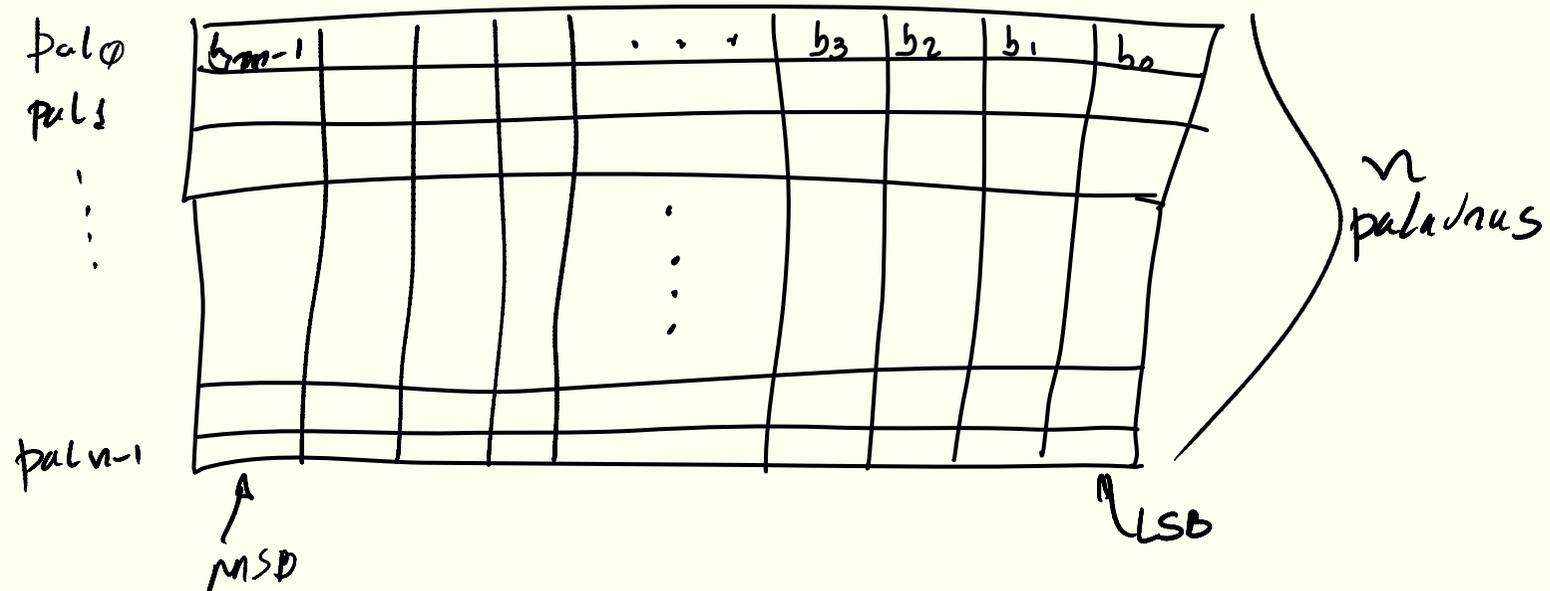
# Memória

---

- **Organização de uma Memória**
  - **Profundidade x Largura**
  - **Ex.: 64Kx8-bits = 64 KBytes**
  - **Obs.: Memórias com mesma capacidade podem ter organizações diferentes**
    - **Ex.: capacidade 512 K bits = 64 KBytes**
      - **64K x 8- bits**
      - **32K x 16-bits**

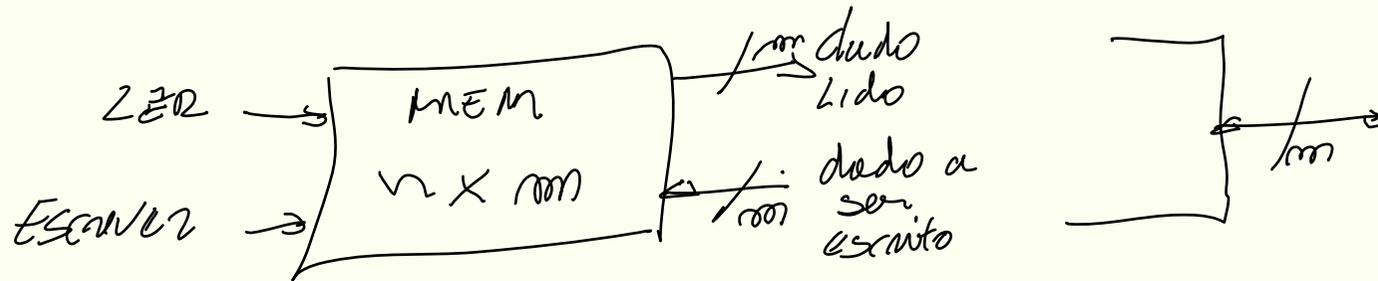
# Memória

- Memória  $n \times m$  (n palavras com m bits)



# Memória

- Uma operação (leitura ou escrita) em uma memória afeta uma e somente uma palavra e todos os bits da palavra são lidos ou escritos em paralelo.



- Obs.:
  - Em memórias com  $m \geq 8$ , como leitura e escritas são mutuamente exclusivas, frequentemente possuem somente um conjunto de linhas de dados bi-direcional.

# Memória

---

- **Forma de especificar a palavra que será lida/escrita**
  - **O método mais comum (e único a ser visto neste curso) :  
Endereçamento explícito**
    - **Associa-se a cada palavra um número inteiro  $\geq 0$  que a especifica univocamente : conhecido como ENDEREÇO da palavra**
    - **Normalmente, para usar toda a capacidade de expressão dos bits de endereço, usualmente o número de palavras de uma memória é uma potência de 2, ié  $n = 2^K$**
    - **O endereço de memória é um inteiro binário com K bits.**
  - **Obs.: Outras formas de endereçamento, utilizadas em aplicações especiais incluem: endereçamento implícito (pilha e fila) e endereçamento por conteúdo.**

# Memória

---

- **AVR ATmega88**
  - **Memória de Dados de 1Kx8bits (1K Bytes)**
    - 10 bits de endereço de dados
  - **512 Bytes de EEPROM**
  - **8K Bytes In-System Self-Programmable Flash**

# Memória

---

- **Mapa de Memória**
  - **Espaço de Endereçamento**
  - **Alocação dos Endereços**
- **Obs.:**
  - **Posição da Palavra de Memória**
  - **Conteúdo da Palavra de Memória**
- **Se um sistema computacional utiliza endereços com K bits para designar as palavras de memória de dados, dizemos que o espaço de endereçamento de dados deste sistema é de  $2^k$  palavras**
  - **Obs.: Nem sempre todo o espaço de endereçamento de memória está alocado. Pode haver buracos**
    - **Permite expansão da memória sem modificação do sistema**

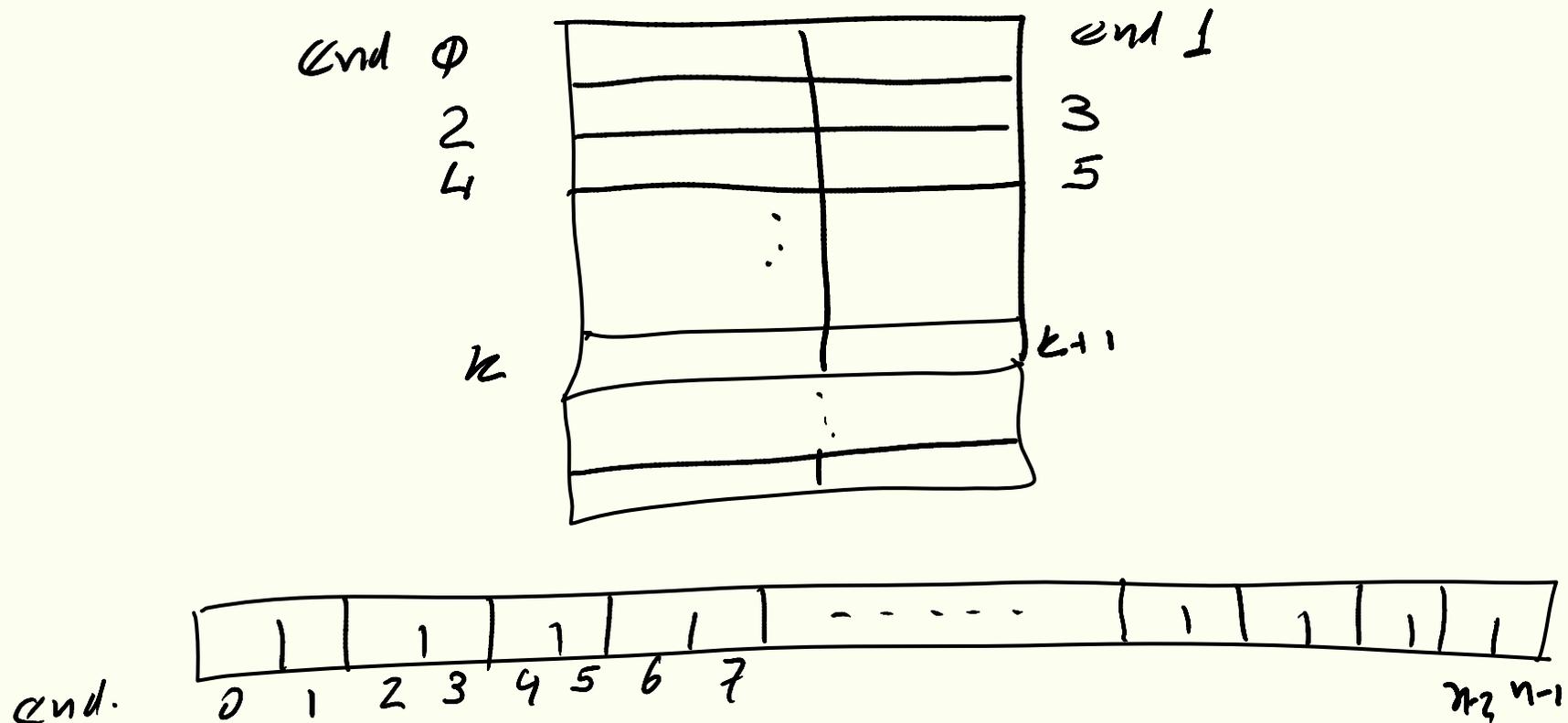
# Memória

---

- **Nível de Endereçamento**
  - **Para memórias com palavras maiores que 1 Byte (ex. 16 bits ou 32 bits ), é possível usar endereçamento com diferentes granularidades**
    - **1 endereço por byte**
    - **1 endereço por palavra**
    - **1 endereço por cada  $\frac{1}{2}$  palavra**
    - **Etc**

# Memória

- Memórias com palavras de 16 bits e endereçamento a Byte



# Memória

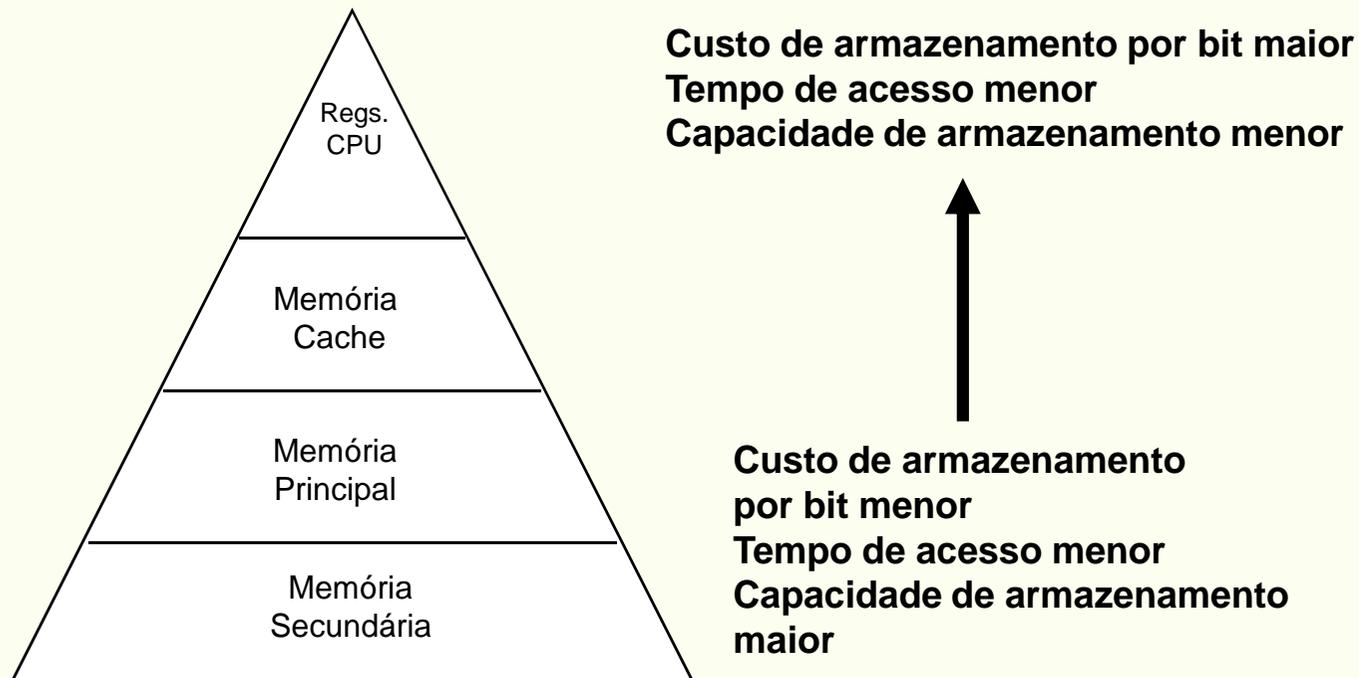
---

- **Memórias com palavras de 16 bits e endereçamento a Byte**
  - **Acesso alinhado**
    - **Só é permitido acesso a palavras cujo endereço é múltiplo do tamanho da palavra em bytes**
      - **Ex. 0 (bytes 0 e 1); 2 (bytes 2 e 3); 4 (bytes 4 e 5); 16 (bytes 16 e 17)**
  - **Acesso desalinhado**
    - **É permitido acesso a palavras a partir de qualquer endereço**
      - **Ex.: 0 (bytes 0 e 1); 1 (bytes 1 e 2);**

# Conceitos Básicos

## Organização Básica de um Computador Digital

- **Unidade de Memória**
  - **Hierarquia de Memória: sistema de memória com objetivo de melhorar o desempenho de um sistema computacional, diminuindo o tempo de acesso médio**



# Conceitos Básicos

---

## Organização Básica de um Computador Digital

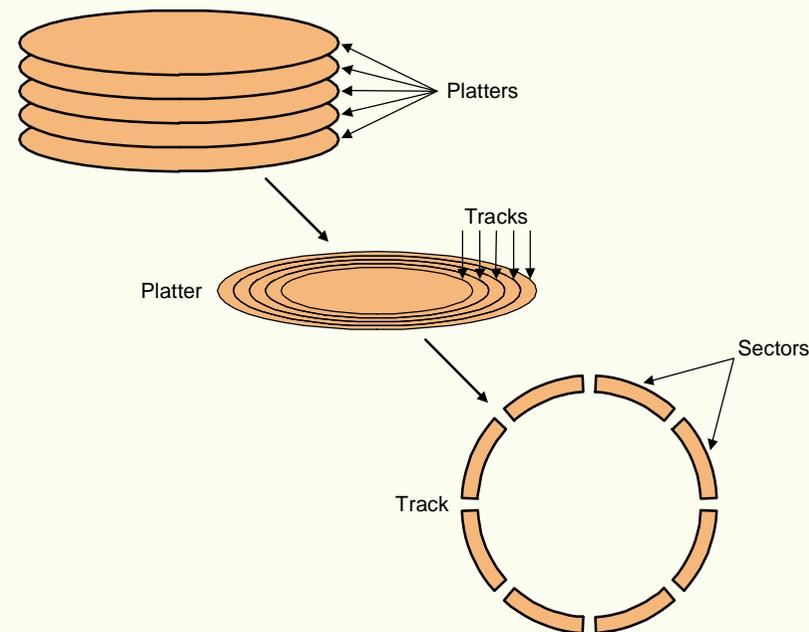
- **Memórias**
  - **Semicondutoras: fabricadas com materiais semicondutores (silício) – circuitos integrados.**
    - **RAM – Random Access Memory : memória de acesso aleatório, volátil.**
      - **SRAM – RAM estática: seu conteúdo só se altera quando se escreve nela ou quando se desliga a tensão de alimentação. Exemplo – registradores da CPU, memória cache.**
      - **DRAM – RAM dinâmica: periodicamente é necessário reescrever o seu conteúdo (refresh de memória) pois há diminuição de cargas elétricas.**  
Exemplo – memória principal.
    - **ROM – Read Only Memory: memória somente de leitura, não volátil.**
      - **ROM: gravação feita pelo fabricante da memória, não apagável;**
      - **PROM – Programmable ROM: programação feita pelo usuário, não apagável;**
      - **EPROM – Erasable PROM: programação feita pelo usuário, apagável através de luz ultra-violeta;**
      - **EEPROM – Electrical EPROM: programação feita pelo usuário, apagável eletricamente;**
    - **Flash – memória semicondutora, não volátil e de escrita e leitura, apagável.**

# Conceitos Básicos

## Organização Básica de um Computador Digital

- Memórias (continuação)
  - Magnéticas
    - Discos – Hard Disk – HDs
    - Ópticos – CD-ROM, DVD, etc.
    - Fitas – cartchos, rolos, etc.

Exemplo: memórias secundárias



**Disco Magnético →  
pratos, lados, trilhas  
e setores**

# Conceitos Básicos

---

## Organização Básica de um Computador Digital

- **Unidade de Entrada e Saída: responsável por gerenciar a ligação entre CPU-Memória-barramentos e os periféricos.**
  - **Interfaces – circuitos simples que apenas compatibilizam a comunicação (protocolo). O controle da transferência é feita pela CPU. Exemplo: interface serial RS232, interface paralela, interface USB;**
  - **Canais de E/S – circuitos que controlam e compatibilizam a comunicação. A CPU apenas inicia a transferência. Exemplo – Controlador de Acesso Direto à Memória (DMA – Direct Access Memory);**
  - **Processadores de E/S – são CPUs dedicadas a fazer E/S de dados. Iniciam e controlam a comunicação.**

# Conceitos Básicos

---

## Organização Básica de um Computador Digital

- **Barramentos: Conjunto de fios que fazem a ligação física entre as diversas unidades.**
  - **Barramento de Endereços: Por onde trafegam os endereços;**
  - **Barramento de Dados: Por onde trafegam os dados;**
  - **Barramento de Controle: por onde trafegam os sinais de controle;**
- **Observação:**

**Internamente à CPU, existe um barramento interno de dados que liga os registradores com a ULA e a UC, e um barramento interno de controle que liga a UC a todos os elementos da CPU.**

# Conceitos Básicos

---

## Organização Básica de um Computador Digital

- **Formato das Instruções**
  - **Tamanho (número de bits) e o significado de cada campo de bits de uma instrução de linguagem de máquina.**
  
- **Conjunto de Instruções**
  - **Cada processador tem o seu conjunto de instruções de linguagem de máquina (ISA – Instruction Set Architecture). Este conjunto contém todas as instruções, em linguagem de máquina, que o processador pode executar.**

# Conceitos Básicos

## Execução de uma instrução pela CPU

- Ciclos de execução de uma instrução:
  - Leitura da instrução da memória principal – Fetch da Instrução
    - REM  $\leftarrow$  PC
    - Read (sinal de controle)
    - PC  $\leftarrow$  PC atualizado
    - RDM  $\leftarrow$  [REM] (instrução lida)
  - Decodificação da instrução
    - RI  $\leftarrow$  RDM (instrução)
    - É feita a decodificação pela Unidade de Controle
  - Busca dos operandos da instrução na memória – se houver
    - REM  $\leftarrow$  End. Dado
    - Read (sinal de controle)
    - RDM  $\leftarrow$  [REM] (operando lido)
  - Execução da instrução – depende da instrução
  - Escrita no Banco de Registradores
- Obs – Quando usamos [..], significa que estamos acessando um conteúdo de memória, cujo endereço está dentro dos colchetes.

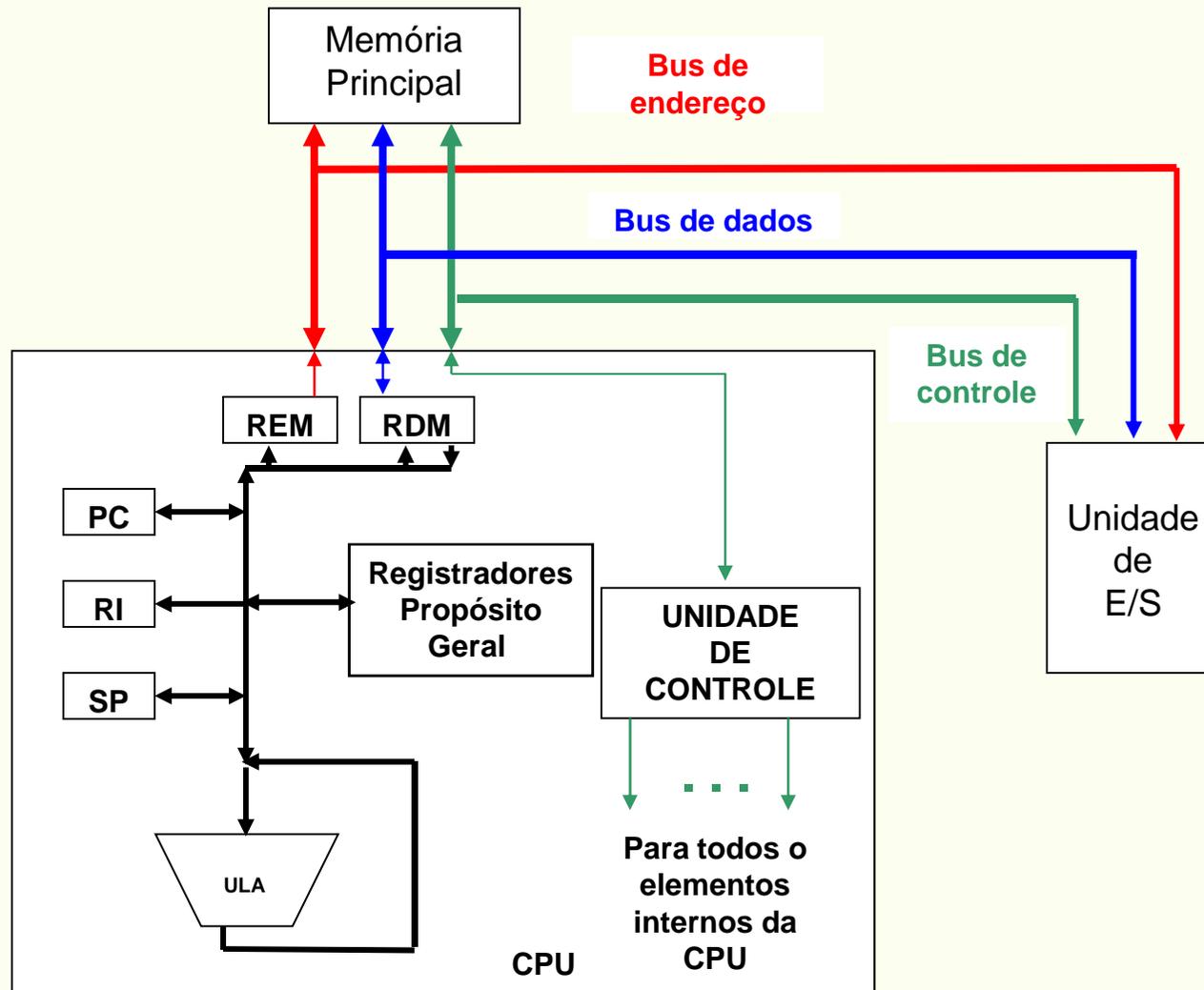
# Conceitos Básicos

## Execução de uma instrução pela CPU

- Ciclos de execução de uma instrução:
  - Leitura da instrução da memória principal – Fetch da Instrução
    - REM  $\leftarrow$  PC
    - Read (sinal de controle)
    - PC  $\leftarrow$  PC atualizado
    - RDM  $\leftarrow$  [REM] (instrução lida)
  - Decodificação da instrução
    - RI  $\leftarrow$  RDM (instrução)
    - É feita a decodificação pela Unidade de Controle
  - Busca dos operandos da instrução no banco de registradores
    - Read-Banco-Reg (sinal de controle)
    - ULA  $\leftarrow$  Operandos lidos
  - Execução da instrução – depende da instrução
  - Escrita no Banco de Registradores
- Obs – Quando usamos [..], significa que estamos acessando um conteúdo de memória, cujo endereço está dentro dos colchetes.

# Conceitos Básicos

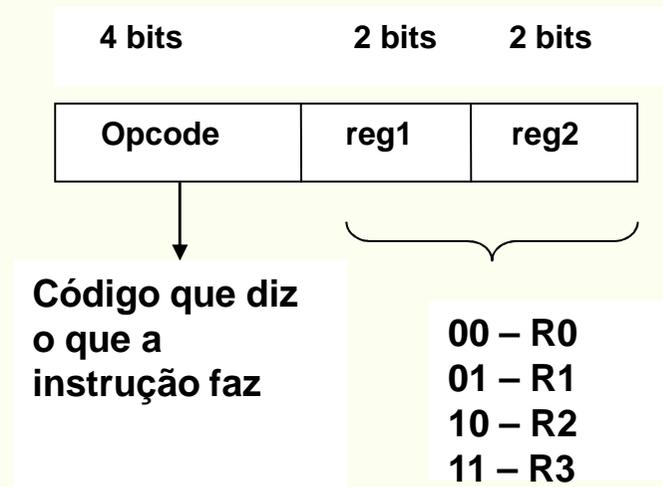
## Organização Básica de um Computador Digital



# Conceitos Básicos

## ESTUDO DE CASO - CPU HIPOTÉTICA

- **Formatos das instruções da CPU HIPOTÉTICA:**
  - **Formato tipo I – Uma palavra de 8 bits, com os seguintes campos:**



**Exemplo: MOV R0,R1 ; R0 ← R1**



# Conceitos Básicos

Mnemônico	Operandos	Opcode	Significado
<b>Instruções de Movimentação de Dados</b>			
MOV	Reg1,Reg2	0000	Reg1 $\leftarrow$ Reg2
MOV	Reg,imed	1000	Reg $\leftarrow$ imed
MOV	Reg,[end]	1001	Reg $\leftarrow$ [end]
MOV	[end],Reg	1010	[end] $\leftarrow$ Reg
<b>Instruções Aritméticas e Lógicas</b>			
ADD	Reg1,Reg2	0001	Reg1 $\leftarrow$ Reg1 + Reg2
ADD	Reg,imed	1011	Reg $\leftarrow$ Reg + imed
SUB	Reg1,Reg2	0010	Reg1 $\leftarrow$ Reg1 - Reg2
SUB	Reg,imed	1100	Reg $\leftarrow$ Reg - imed
AND	Reg1,Reg2	0011	Reg1 $\leftarrow$ Reg1 <u>e</u> Reg2
AND	Reg,imed	1101	Reg $\leftarrow$ Reg <u>e</u> imed
OR	Reg1,Reg2	0100	Reg1 $\leftarrow$ Reg1 <u>ou</u> Reg2
<b>Instruções de Manipulação de Pilha</b>			
PUSH	Reg	0101	SP-- , [SP] $\leftarrow$ Reg
POP	Reg	0110	Reg $\leftarrow$ [SP], SP++
<b>Instruções de Controle de Fluxo de Execução</b>			
JMP	end	1110	PC $\leftarrow$ end
CALL	end	1111	SP-- , [SP] $\leftarrow$ PC , PC $\leftarrow$ end
RET	---	0111	PC $\leftarrow$ [SP] , SP++

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

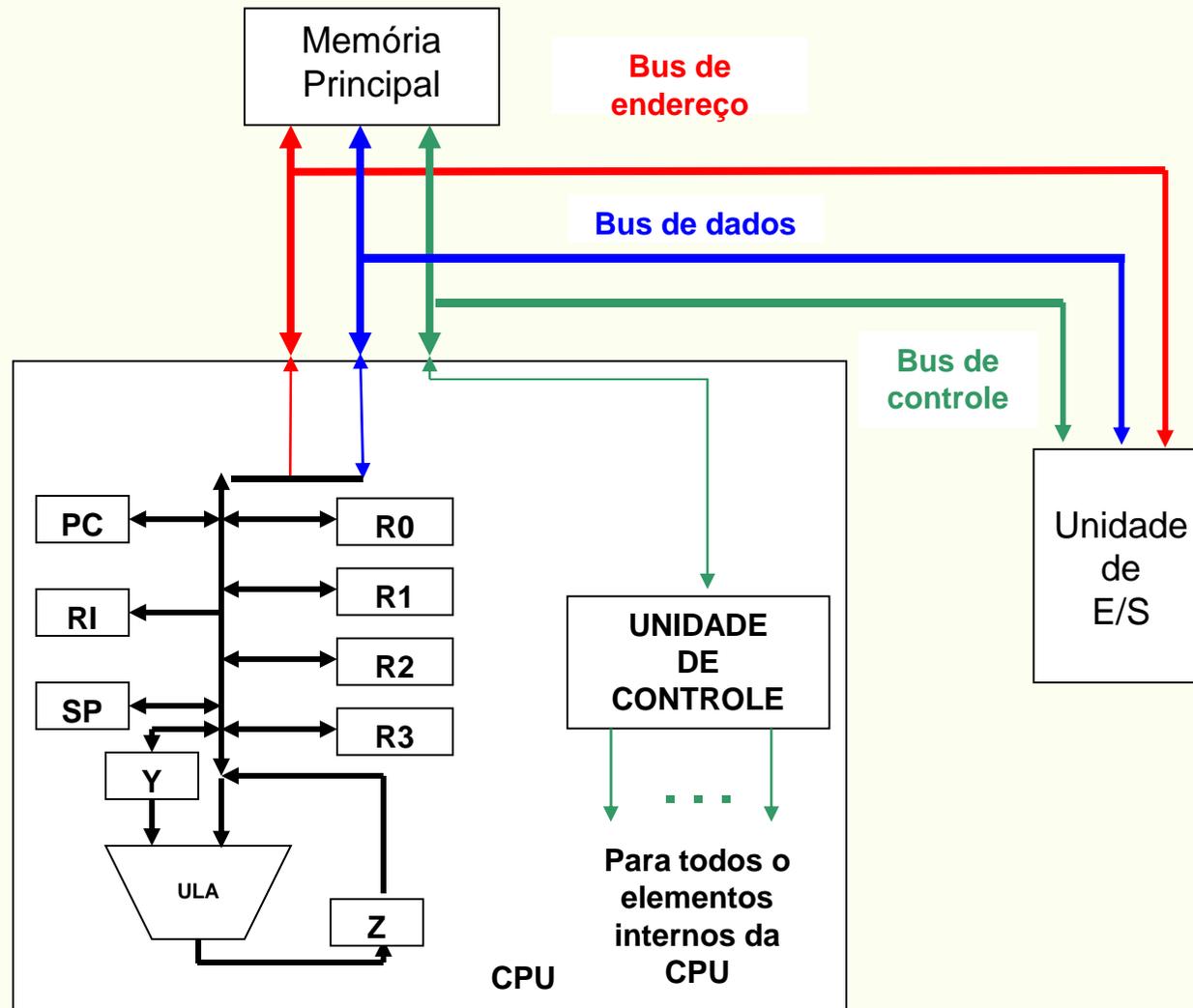
---

## Exercícios

- **Mostrar o ciclo de execução de instruções para todas as instruções do ISA das CPUs Hipotéticas 1 e 2.**

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

## ESTUDO DE CASO 2 - CPU HIPOTÉTICA

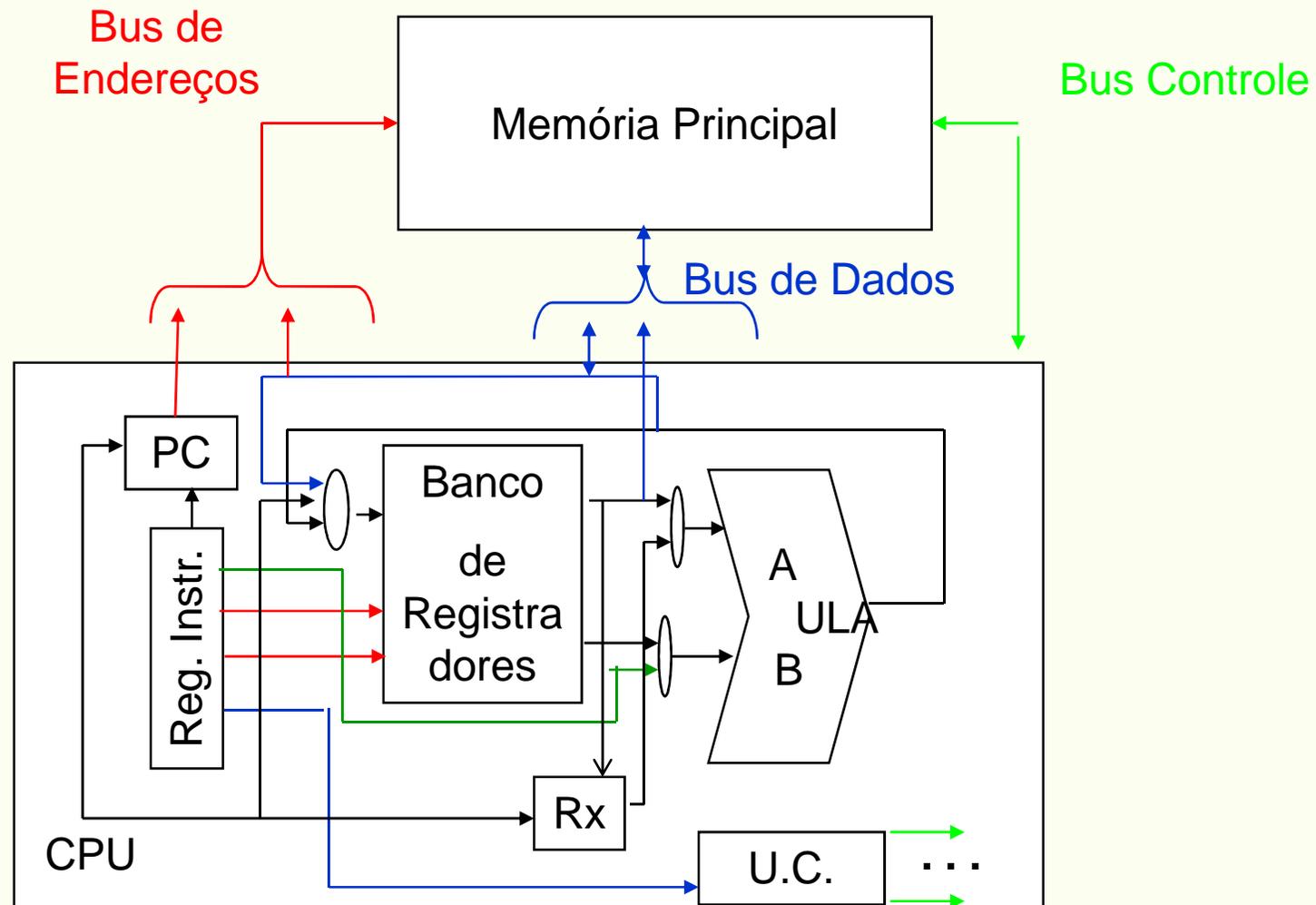


# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Mnemônico	Operandos	Opcode	Significado
<b>Instruções de Movimentação de Dados</b>			
MOV	Reg1,Reg2	0000	Reg1 $\leftarrow$ Reg2
MOV	Reg,imed	1000	Reg $\leftarrow$ imed
MOV	Reg,[end]	1001	Reg $\leftarrow$ [end]
MOV	[end],Reg	1010	[end] $\leftarrow$ Reg
<b>Instruções Aritméticas e Lógicas</b>			
ADD	Reg1,Reg2	0001	Reg1 $\leftarrow$ Reg1 + Reg2
ADD	Reg,imed	1011	Reg $\leftarrow$ Reg + imed
SUB	Reg1,Reg2	0010	Reg1 $\leftarrow$ Reg1 - Reg2
SUB	Reg,imed	1100	Reg $\leftarrow$ Reg - imed
AND	Reg1,Reg2	0011	Reg1 $\leftarrow$ Reg1 <u>e</u> Reg2
AND	Reg,imed	1101	Reg $\leftarrow$ Reg <u>e</u> imed
OR	Reg1,Reg2	0100	Reg1 $\leftarrow$ Reg1 <u>ou</u> Reg2
<b>Instruções de Manipulação de Pilha</b>			
PUSH	Reg	0101	SP-- , [SP] $\leftarrow$ Reg
POP	Reg	0110	Reg $\leftarrow$ [SP], SP++
<b>Instruções de Controle de Fluxo de Execução</b>			
JMP	end	1110	PC $\leftarrow$ end
CALL	end	1111	SP-- , [SP] $\leftarrow$ PC , PC $\leftarrow$ end
RET	---	0111	PC $\leftarrow$ [SP] , SP++

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

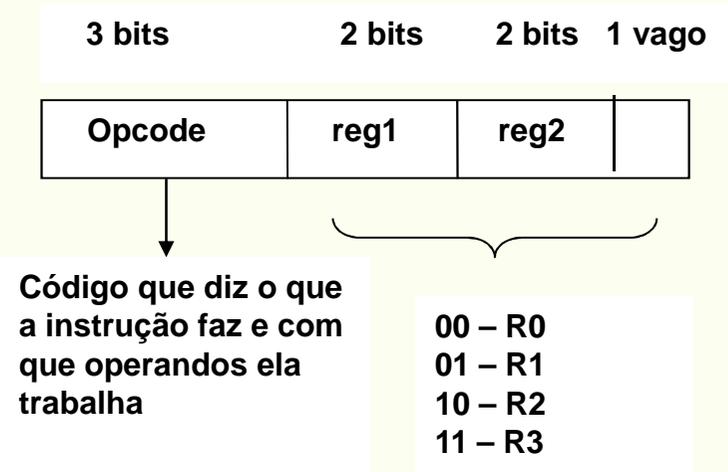
## CPU HIPOTÉTICA 3



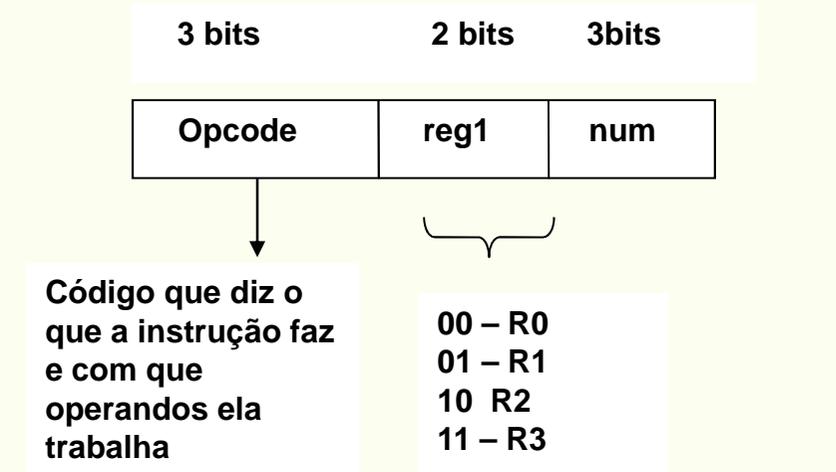
# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

## ESTUDO DE CASO - CPU HIPOTÉTICA 3 – Formato de Instruções

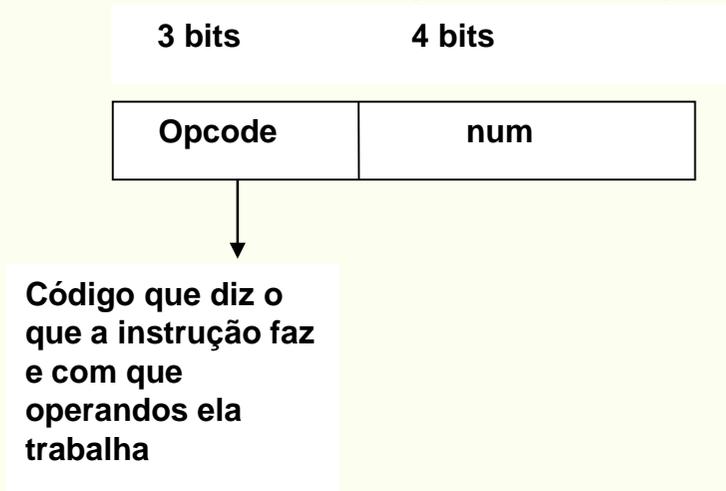
### – Formato tipo R – Registrador



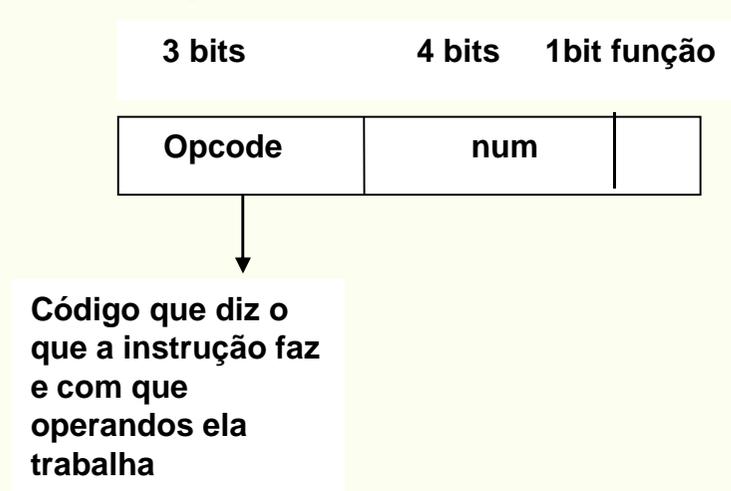
### – Formato I – Imediato



### – Formato tipo J – Jump



### – Formato tipo S - subrotina



# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

<b>Conjunto de Instruções – CPU Hipotética 3</b>			
<b>Mnemônico</b>	<b>Operandos</b>	<b>Opcode</b>	<b>Significado</b>
		<b>Instrução especial</b>	
<b>MV</b>	<b>Rx,reg</b>	<b>000</b>	<b>Rx ← Reg</b>
		<b>Instruções de load e store</b>	
<b>LW</b>	<b>Reg,num</b>	<b>001</b>	<b>Reg ← [Rx + num]</b>
<b>SW</b>	<b>Reg,num</b>	<b>010</b>	<b>[Rx + num] ← Reg</b>
		<b>Instruções Aritméticas e Lógicas</b>	
<b>ADD</b>	<b>Reg1,Reg2</b>	<b>011</b>	<b>Reg1 ← Reg1 + Reg2</b>
<b>SUB</b>	<b>Reg1,Reg2</b>	<b>100</b>	<b>Reg1 ← Reg1 - Reg2</b>
<b>AND</b>	<b>Reg1,Reg2</b>	<b>101</b>	<b>Reg1 ← Reg1 <u>e</u> Reg2</b>
		<b>Instruções de Controle de Fluxo de Execução</b>	
<b>JMP</b>	<b>num</b>	<b>110</b>	<b>PC ← num</b>
<b>JAL</b>	<b>num</b>	<b>111 0</b>	<b>Rx ← PC , PC ← num</b>
<b>RET</b>	<b>---</b>	<b>111 1</b>	<b>PC←Rx</b>

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

---

## Exercícios

- **Mostrar o ciclo de execução de instruções para todas as instruções do ISA da CPU Hipotética 3.**