

# Aula 1

## Organização de Computadores

Profa. Débora Matos

# Objetivos a disciplina

- Caracterização de desempenho das arquiteturas. Organização de processadores: bloco operacional e bloco de controle.
- Organização serial e paralela (pipeline) da CPU.
- Estudo de sistema de memória (hierarquia da memória, memória cache e memória virtual).
- Métodos para aumento de desempenho: organização de pipelines, máquinas super-escalares.
- Estudos de caso de processadores contemporâneos. Ferramentas para análise e projeto de organizações.

# Cronograma das aulas

Data	Assunto
06/08	Aula 1 – Apresentação da disciplina. Plano de Ensino. Revisão de conceitos. Introdução a microprocessadores: revisão dos conceitos de organização. Avaliação de desempenho.
13/08	Aula 2 – Lei de Amdahl. Análise de Potência. Como calcular o CPI, o MIPS e o tempo de CPI e análise de potência.
20/08	Aula 3 – Entendendo as instruções do MIPS e as unidades funcionais requeridas para execução das mesmas.
27/08	Aula 4 – Caminho de dados do MIPS, unidades funcionais, implementação de ciclo único e multiciclo.
03/09	Aula 5 – Implementação do MIPS com pipeline. Entendendo a melhora da performance com a técnica de pipeline.
10/09	Aula 6 – Tipos de conflitos de pipeline: Hazards de dados, hazards de controle. Soluções para os conflitos.
17/09	PROVA 1
24/09	Aula 7 – Hazards de controle. Implementação de previsão dinâmica de desvios. Implementação de stalls, forwarding e exceções. Definição do trabalho a ser desenvolvido em VHDL.

# Cronograma das aulas

01/10	Aula 8 – Hierarquia de memória, tipos de memória, tecnologia utilizada, capacidade de armazenamento, velocidade, memória cache.
08/10	Aula 9 – Hierarquia de memória, memória cache, tipos de mapeamentos da memória principal para a cache, técnicas de substituição de blocos, exercícios com memória cache.
22/10	Aula 10 – Correção dos exercícios sobre hierarquia de memória, cálculo do tamanho da cache, cálculo de CPI.
29/10	Aula 11 – Memória Virtual
05/11	Aula 12 – Processadores Paralelos, superpipeline, GPU
12/11	Aula 13 - Processadores Superescalares e VLIW
19/11	PROVA 2
26/11	Aula 14 - Entrega das notas das provas. Resolução da prova. Entrega por parte dos alunos da parte 1 do trabalho, orientação quanto a parte 2 do trabalho.
03/12	Aula 15 - Apresentação da parte 2 do trabalho. Entrega dos relatórios.
10/12	PROVA DE RECUPERAÇÃO
17/12	Divulgação dos conceitos finais.

# Avaliações

- PROVA 1 (P1)
- PROVA 2 (P2)
- Trabalhos (T) = Trabalhos de desenvolvimento de organizações de computadores em VHDL
- Nota final =  $(P1 + P2 + T)/3$

# Avaliações

- No caso do aluno não atingir a média mínima, será possível a realização da **recuperação** para substituir a nota de uma das áreas com prova;
- A recuperação será de todo o conteúdo e não substitui a **nota T**.
- A nota obtida na recuperação substituirá uma **das 2 notas com prova** e a média será recalculada.
- O aluno só pode realizar a prova se tiver **média maior ou igual a 4**, considerando todas as avaliações.

# Bibliografia

## Bibliografia Básica

- PATTERSON, David A; HENNESSY, John L. Organização e Projeto de Computadores: a Interface Hardware/Software. 3. Ed. Rio de Janeiro: Elsevier, 2005. xvii, 484 p.
- STALLINGS, W. Arquitetura e organização de computadores. 5° ed., São Paulo, 2010.

## Bibliografia Complementar

- TANENBAUM, Andrew, Organização estruturada de computadores, 4ª ed.
- Hennessy, John. L.; Patterson, David A. Arquitetura de Computadores: Uma abordagem quantitativa, 4ª Edição, 2007. Campus.

# Revisão de alguns conceitos



# Seguindo pedidos...

## Teremos na disciplina

o



# ORGJogo

## Regras:

- Formar equipes (4 a 5 alunos);
- Questões relacionadas ao conteúdo da disciplina ocorrerão ao longo do semestre;
- A equipe vencedora, ao final do semestre, terá **1 ponto a mais na média;**
- A equipe que ficar em segundo lugar, ao final do semestre, terá 0,5 ponto a mais na média;

# ORGjogo

## Regras:

- Quando a questão for colocada, a equipe deve se reunir para formalizar a resposta.
- Será dado em torno de 5 minutos para entregar a resposta escrita;
- Se a equipe responder corretamente, 1 ponto é computado;
- Se a equipe responder errado ou não entregar, nenhum ponto é computado.

# ORGjog

## Regras:

0

- As respostas serão lidas e primeiramente avaliadas pelas equipes;
- Se parte da resposta não estiver 100% correta, a equipe que respondeu ganha somente 0,5;
- Uma outra equipe pode corrigir a resposta de uma equipe e se corrigir corretamente, esta equipe ganha 0,5 a mais;

Exemplo

ORCJogo

Pergunta, pergunta, pergunta???

# Arquitetura X Organização

- **Arquitetura:**

refere-se aos atributos de um sistema que são visíveis para o programador, ou seja, aos atributos que tem impacto direto sobre a execução lógica de um programa.

Exemplos:

- ✓ conjunto de instruções,
- ✓ número de bits utilizados para representar os tipos de dados,
- ✓ mecanismos de entrada e saída
- ✓ as técnicas de endereçamento à memória.

# Arquitetura X Organização

- **Organização:** refere-se as unidades operacionais e suas interconexões que implementam as especificações da arquitetura.

Exemplos:

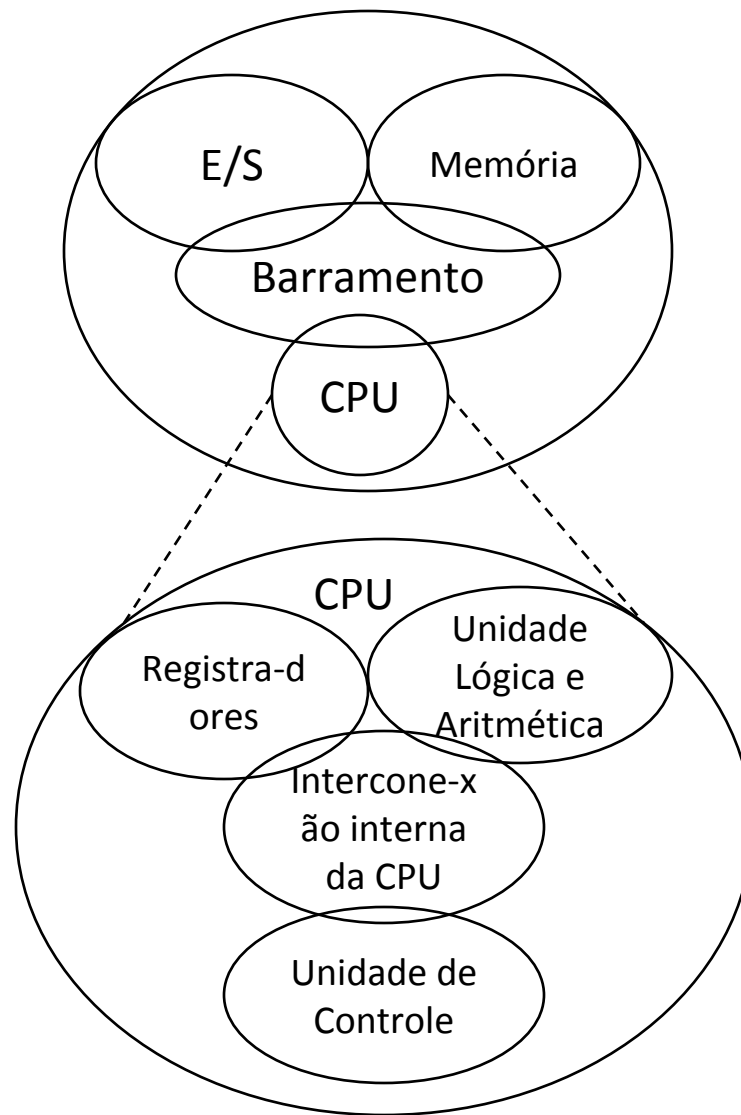
- ✓ Detalhes de hardware transparentes aos programador: sinais de controle, periféricos, tecnologia de memória utilizada, número de ULAS, interconexões entre os componentes, etc.

# Arquitetura X Organização

- Definir se um computador deve ou não ter uma instrução de multiplicação é uma decisão do projeto de sua **arquitetura**.
- Definir se essa instrução será implementada por uma unidade especial de multiplicação ou por um mecanismo que utiliza repetidamente sua unidade de soma é uma decisão do projeto da sua **organização**.



# Componentes de microcomputador



# **Microprocessadores: um breve histórico**

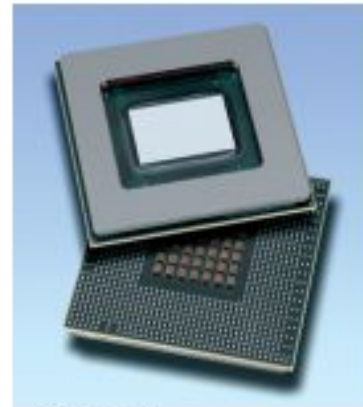
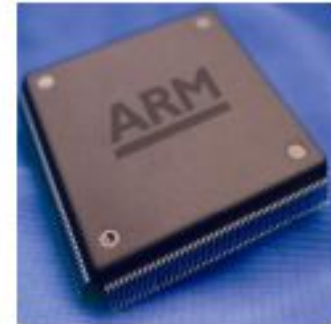
**Quais são alguns exemplos de fabricantes de processadores:**

# Microprocessadores: um breve histórico

- Microprocessadores vem sendo desenvolvidos e produzidos por um grande número de fabricantes:

- Intel, AMD, Motorola, Texas, IBM e várias outros
- Várias empresas já foram participantes ativos no mercado de microprocessadores (como Zilog) e desapareceram, perderam importância ou foram adquiridas pelos concorrentes

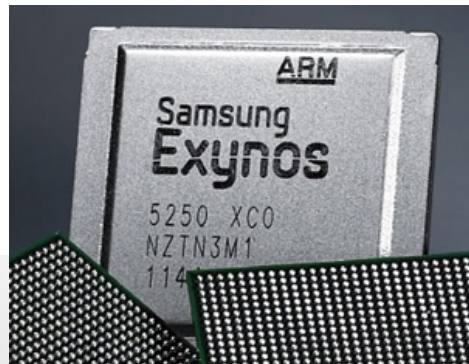
microprocessador  
ARM



IBM Cell

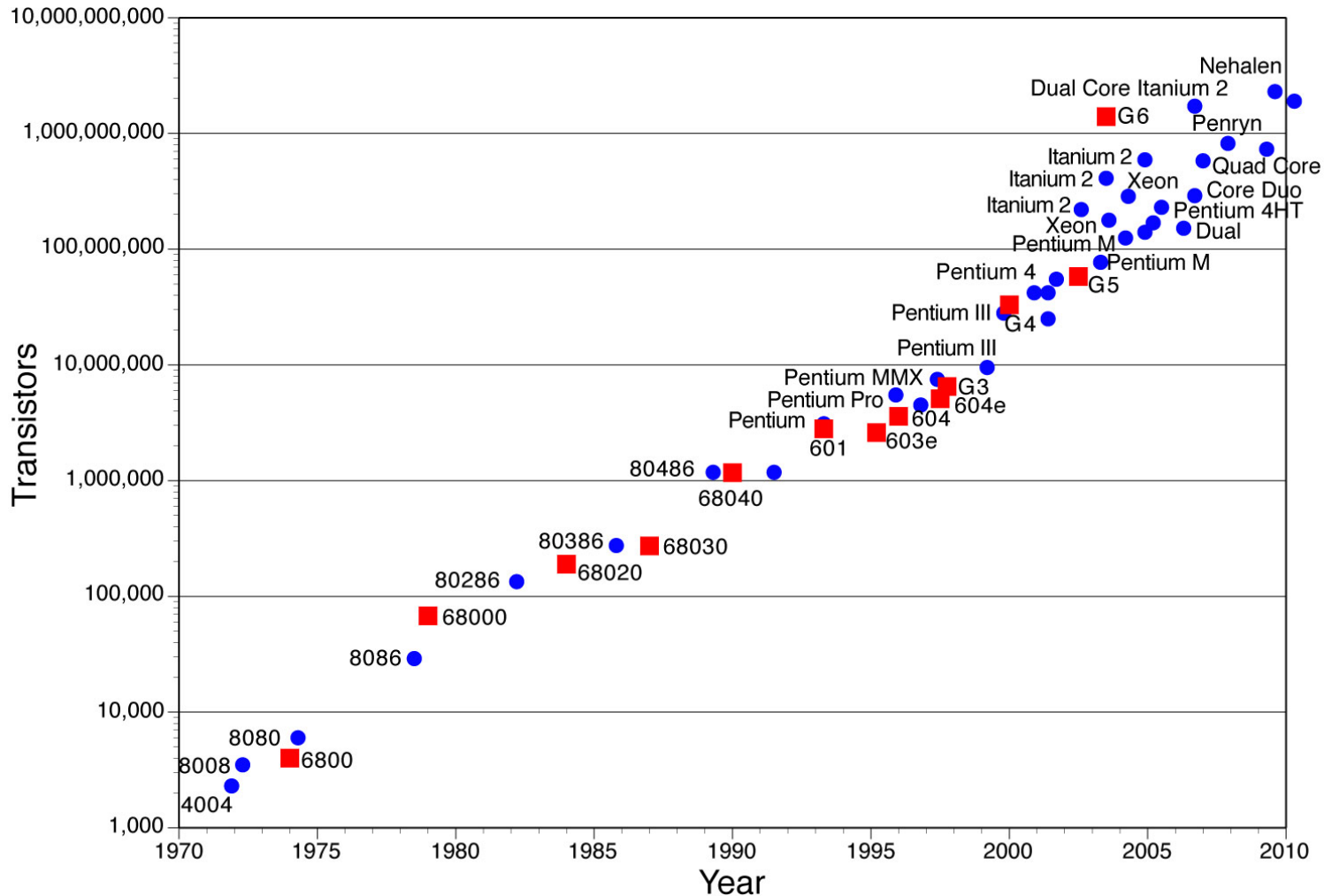


Motorola



# Lei de Moore

A lei de Moore diz que a quantidade de transistores em um circuito integrado dobra a cada 18 meses.



# Lei de Moore

- Um processador é um circuito integrado composto por milhões de transistores (ex. core i7 possui o número de transistores próximo a 1 bilhão);
- Estes transistores são agrupados para desempenharem determinadas funções (cálculos aritméticos, registro, condições lógicas...);

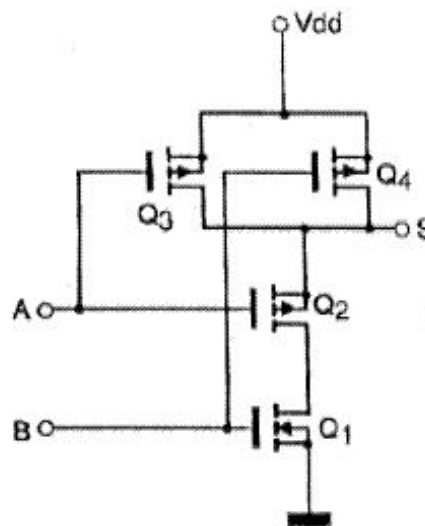
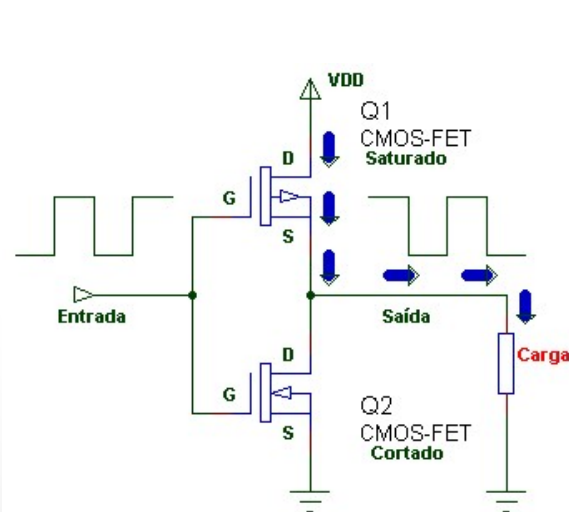
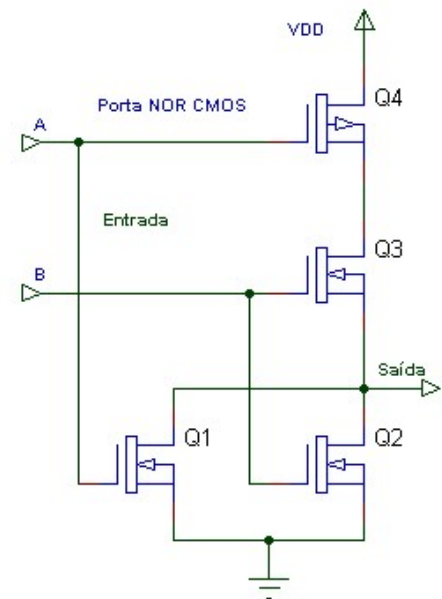


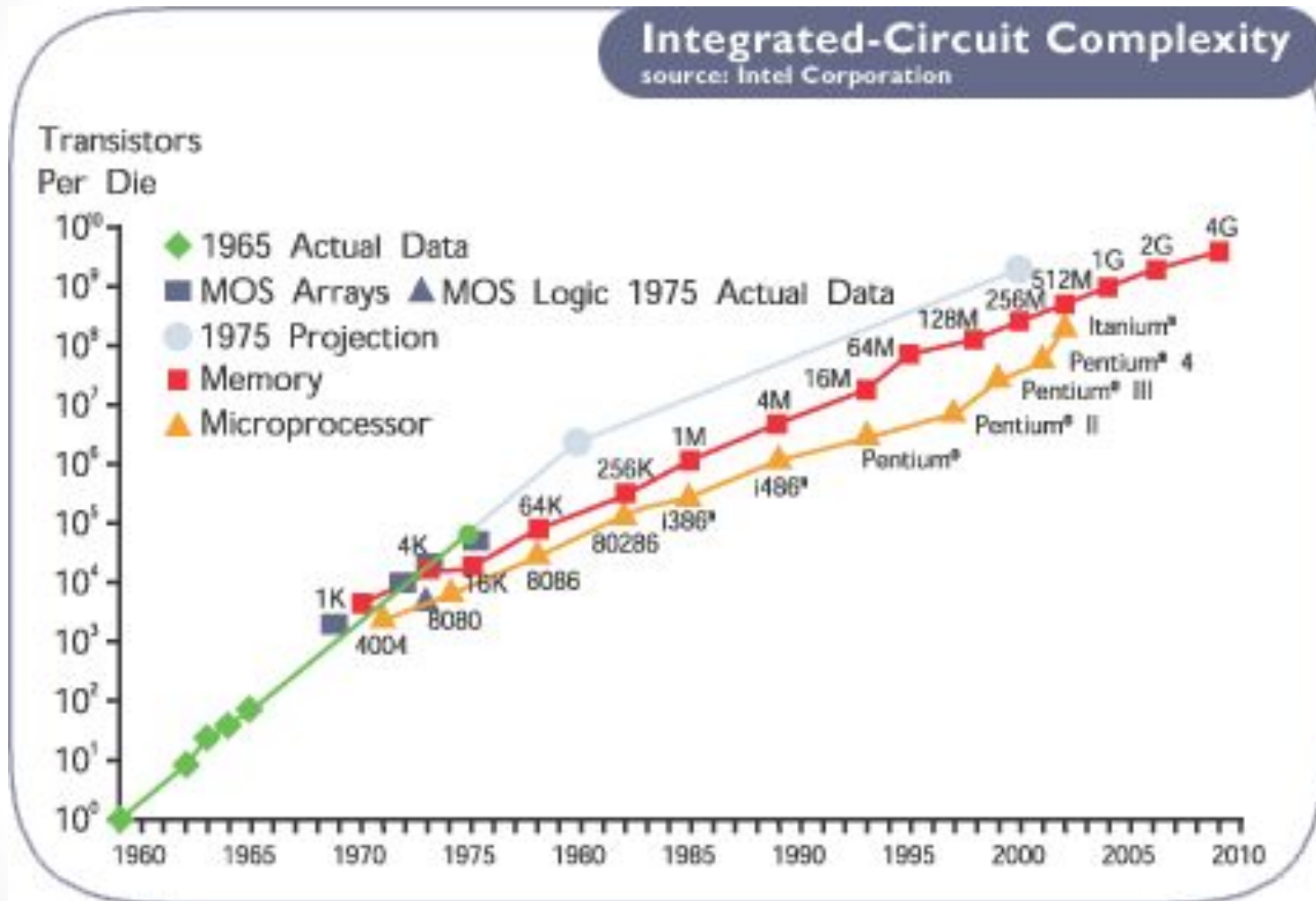
Figura 14 - Porta NAND CMOS.



# Lei de Moore

- Sustentar a taxa de progresso definida pela lei de Moore durante mais de 40 anos exigiu incríveis inovações tecnológicas nas técnicas de fabricação.

# Memória DRAM






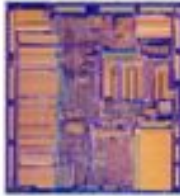

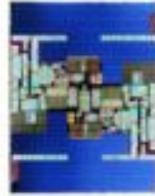
Capacidade por chip de DRAM ao longo do tempo



# Microprocessadores: um breve histórico

Tem evoluído, tornando-se menor e mais baratos

(<http://www.computerhistory.org/semiconductor/>)

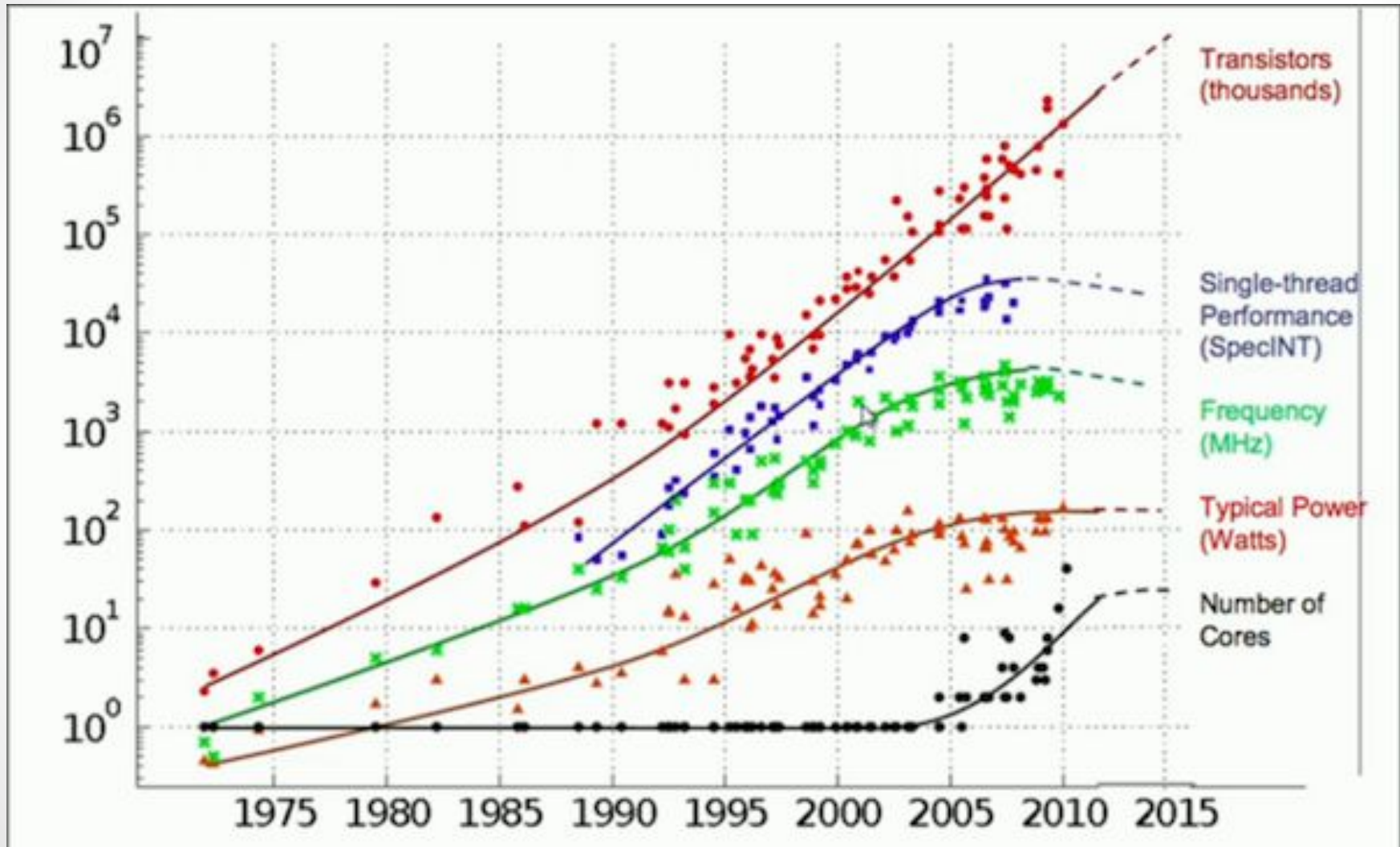
1950s	1960s	1970s	1980s	1990s	2000s
Silicon Transistor	TTL Quad Gate	8-bit Microprocessor	32-bit Microprocessor	32-bit Microprocessor	64-bit Microprocessor
					
<b>1</b> Transistor	<b>16</b> Transistors	<b>4500</b> Transistors	<b>275,000</b> Transistors	<b>3,100,000</b> Transistors	<b>592,000,000</b> Transistors

microprocessadores de 8 a 64 bits

8 a 64 indica o número de bits da palavra de dados. Quando maior o número de bits, maior a capacidade de representação de inteiros.



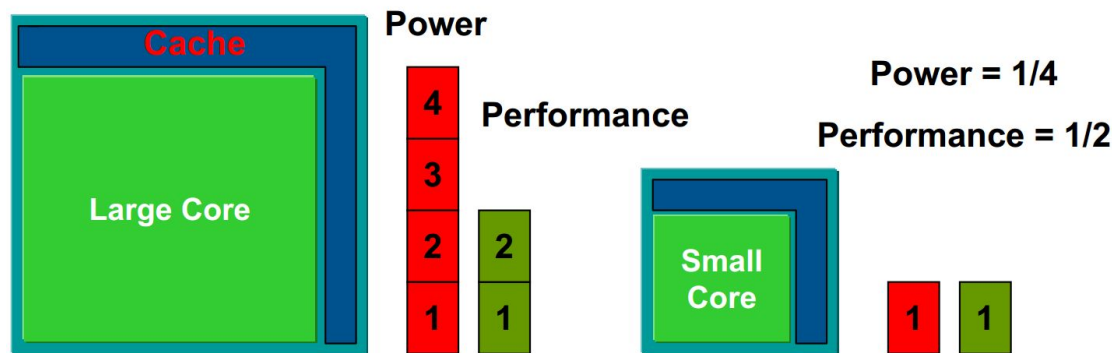
# Microprocessadores ao longo dos anos



Fonte: Data Processing in ExaScale-Class Computer Systems (C. Moore, April 2011).

# Microprocessadores ao longo dos anos

- Solução: substituição do aumento da frequência pelo aumento do número de cores.



Fonte: Keynote presentation (L. Benini, RSP 2010).

- MPSoC (Multiprocessor Sytem on Chip):** sistema composto por múltiplos cores (possivelmente heterogêneos), hierarquia de memória e componentes de entrada/saída (I/O) em um CI (Circuito Integrado).

# Microprocessadores ao longo dos anos

## Para entender a evolução...

DRAM		
Year	Size	Cycle Time
1980	64 Kb	250 ns
1983	256 Kb	220 ns
1986	1 Mb	190 ns
1989	4 Mb	165 ns
1992	16 Mb	145 ns
1995	64 Mb	120 ns

**1000:1!** (Size growth from 1980 to 1995)  
**2:1!** (Cycle Time reduction from 1980 to 1995)

Fonte: Presentation CS252 (Prof. Kurt Keutzer, 2000)

# Quais são alguns dos avanços que temos visto na arquitetura dos computadores?

- Aumento da capacidade de armazenamento
- Aumento da frequência de operação do processador
- Aumento do número de núcleos de processamento
- Aumento do paralelismo das organizações de computadores
- Processadores mais tolerante à falhas

# Qual o objetivo desses avanços?

- Acelerar as aplicações, aplicativos, softwares;
- Prover maior capacidade de armazenamento;
- Possibilitar o uso de mais recursos;
- Reduzir o consumo de energia;
- Aumentar a eficiência energética;
- Reduzir o número de falhas;

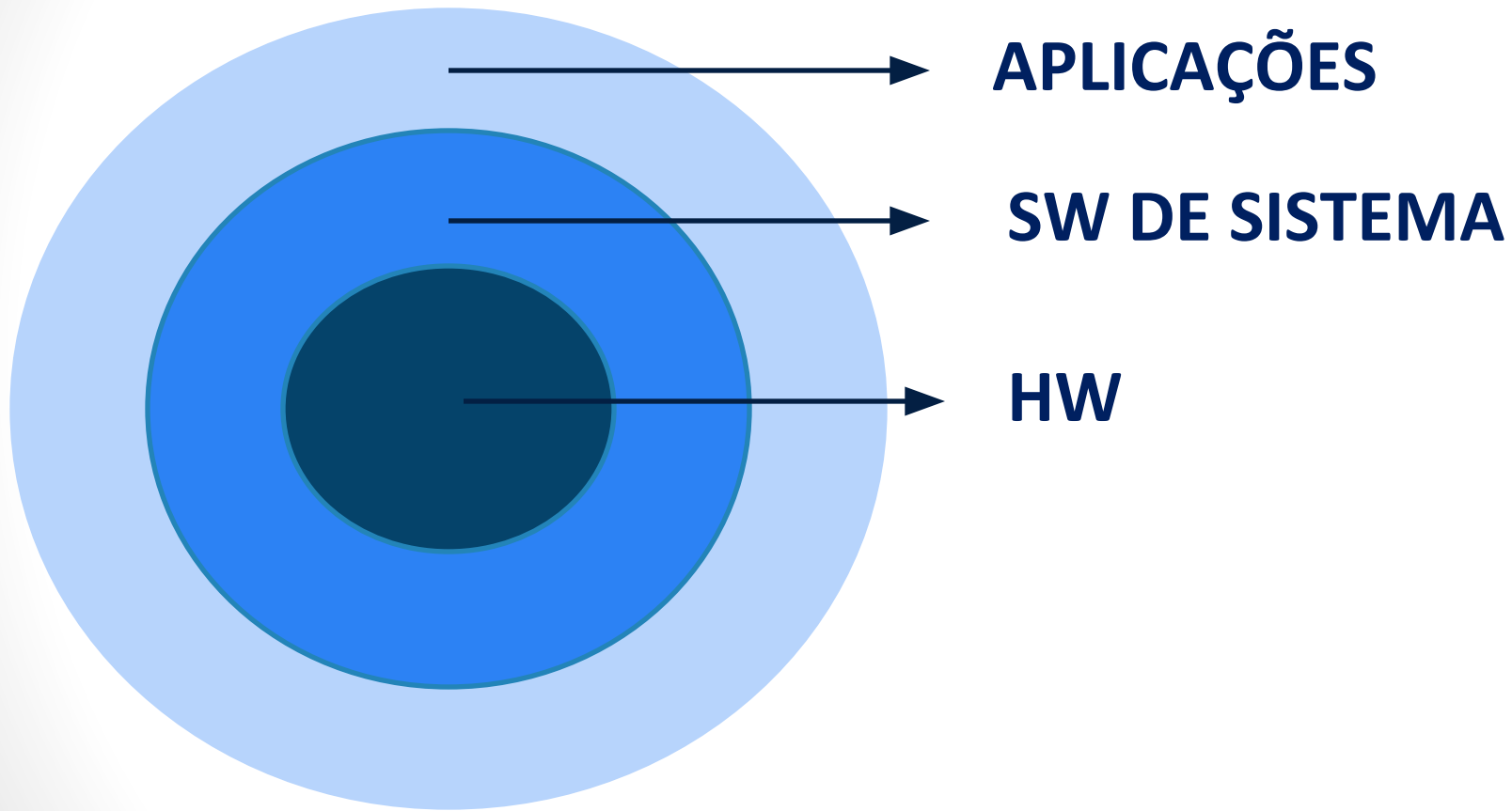
# O que afeta o desempenho de um computador?

- Requisitos de HW e SW
  - ✓ Algoritmos dos programas
  - ✓ Linguagem de programação e compiladores
  - ✓ Sistema operacional
  - ✓ O projeto do processador
  - ✓ Sistema de entrada e saída, dispositivos

# HW x SW

- O hardware de um computador só pode executar instruções de baixo nível extremamente simples.
- De uma aplicação complexa até as instruções simples, envolve várias camadas de SW que interpretam ou traduzem operações de alto nível em instruções simples de computador.

# HW x SW





# Exemplos de SW de sistema:

- ✓ Sistema Operacional
- ✓ Compilador
- ✓ Montadores
- ✓ Interpretador
- ✓ Drivers

# O que faz o Sistema Operacional?

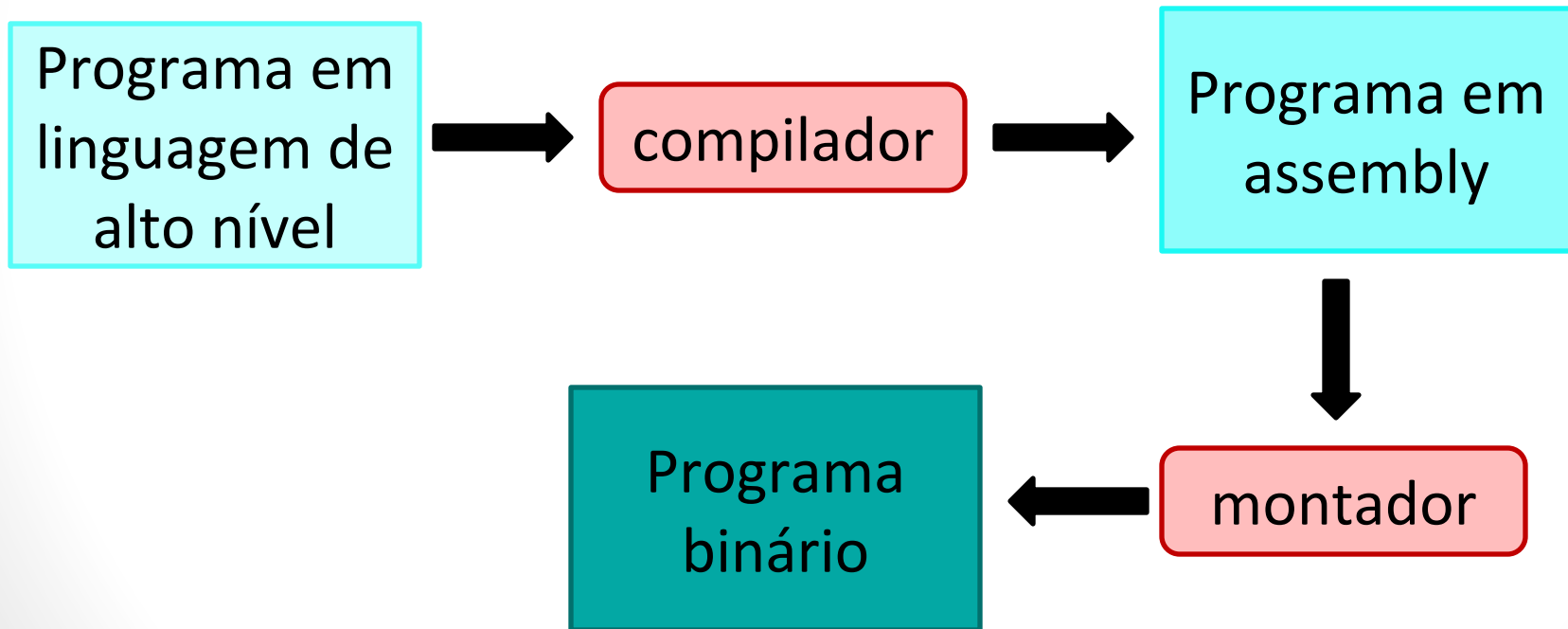
Gerencia os recursos do computador:

- manipula operações básicas de entrada e saída
- Aloca armazenamento e memória
- Possibilita e controla o compartilhamento do processador entre aplicações que rodam simultaneamente.

# O que fazem os compiladores?

Realizam a tradução de um programa descrito em linguagem de programação de alto nível em linguagem de máquina.

# Função dos compiladores e montadores



# Função dos compiladores e montadores

Programa em linguagem  
de alto nível (em C)

```
swap(int v[], int k)
{ int temp;
  temp = v[k];
  v[k] = v[k + 1];
  v[k + 1] = temp;
}
```

Compilador

Programa em linguagem  
assembly (para MIPS)

```
swap:    multi $2, $5, 4
         add $2, $4, $2
         lw $15, 0($2)
         lw $16, 4($2)
         sw $16, 0($2)
         sw $15, 4($2)
         jr $31
```

Assembler/  
Montador

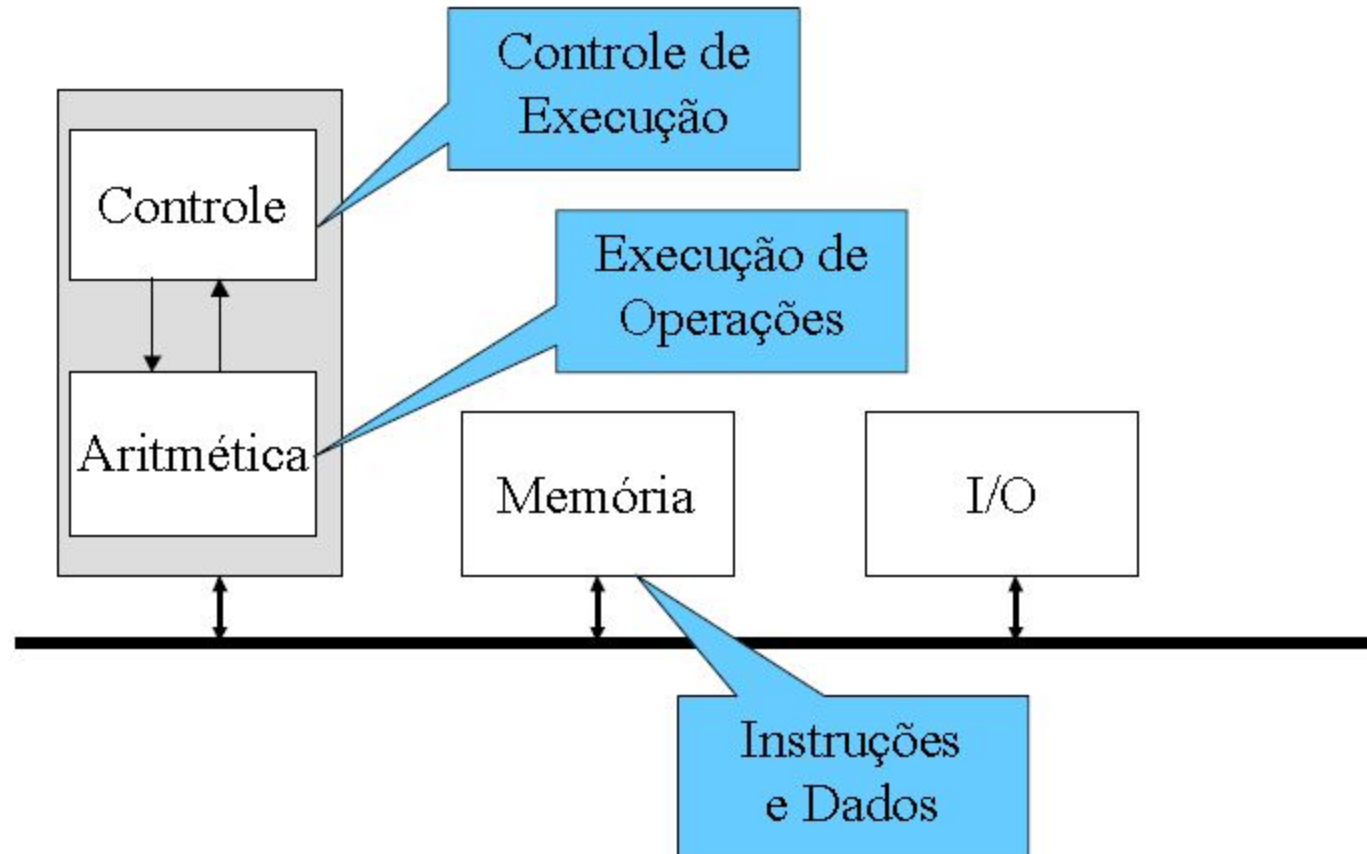
Programa em linguagem  
de máquina (para MIPS)

```
00001010110000110101111001010100
10111001001000111001001010100100
01110101101010100101110101001010
1010100101010101010111101011111
00010100010001010101101111110100
01010100010010101010010101001001
01010010111001011110010101101000
```

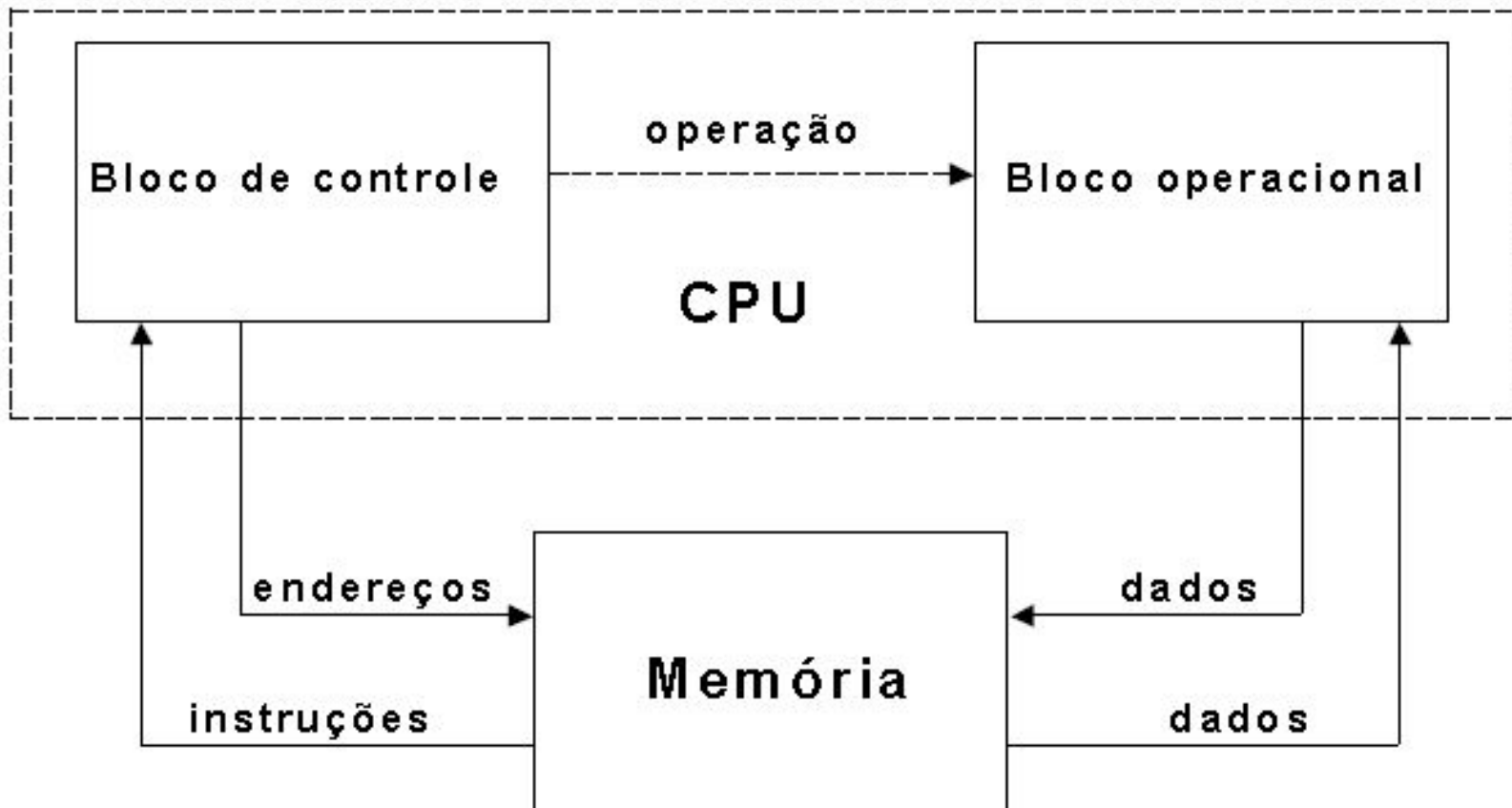
# Organização de um Computador

- Modelo Von Neumann (1945)
  - Conceito de programa armazenado
  - Separação da Unidade Aritmética e de Controle
  - Utilização de barramentos e registradores
  - Hardware de entrada e saída (I/O)

# Modelo Von Neumann



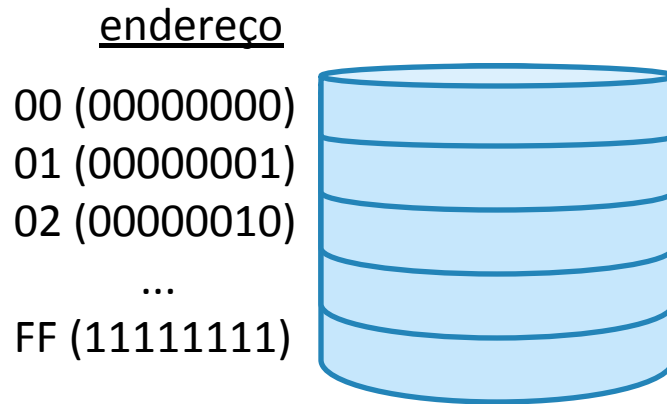
# Modelo Von Neumann





# Memória

A memória é organizada em posições:



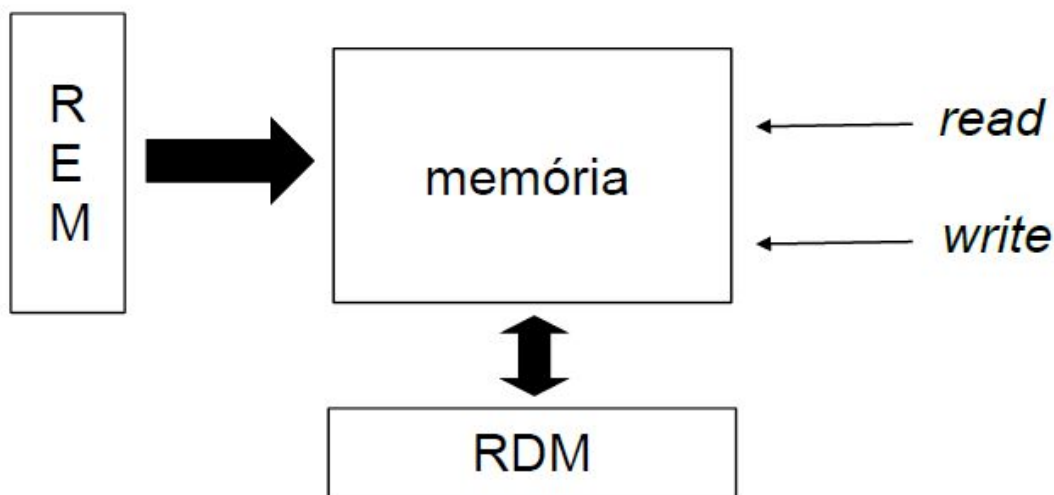
Cada elemento da memória é armazenado  
em um endereço

# Memória

- A memória de um sistema computacional tem a função de armazenar **dados** e **instruções**:
  - Organizada em posições;
  - Podem ser visualizadas como elementos de uma matriz;
  - Cada elemento tem um endereço.
- Então uma memória que tenha  $x$  posições:
  - Cada posição pode ser referenciada diretamente de acordo com a sua colocação na sequência;
  - Se uma memória tem 4096 posições existem posições de 0, ..., 4095;
  - Instruções são executadas em uma sequência determinada pela sua posição de memória

# Memória

- Memória é formada por elementos armazenadores de informações;
- É dividida em **palavras**;
- Cada palavra é identificada unicamente por **um endereço**;
- Conteúdo armazenado nas palavras da memória tanto pode representar **dados** como **instruções**;



# Princípios básicos

- O endereço representa uma posição particular na memória e pode ser formado de várias maneiras;
- A Unidade Lógica e Aritmética (ULA) é responsável por realizar ações indicadas nas instruções, executando operações numéricas (aritméticas) e não numéricas (lógica);
  - Preparação de informações de desvios do programa;
- O controle do programa e a ULA formam a unidade central de processamento, ou

# Busca-Decodificação-Execução de instruções

## ➤ **BUSCA:**

PC – (program counter) – contador de instruções contém sempre a posição da próxima instrução a ser executada. A instrução apontada pelo PC é trazida da memória para uma área de armazenamento chamada registrador de instruções.

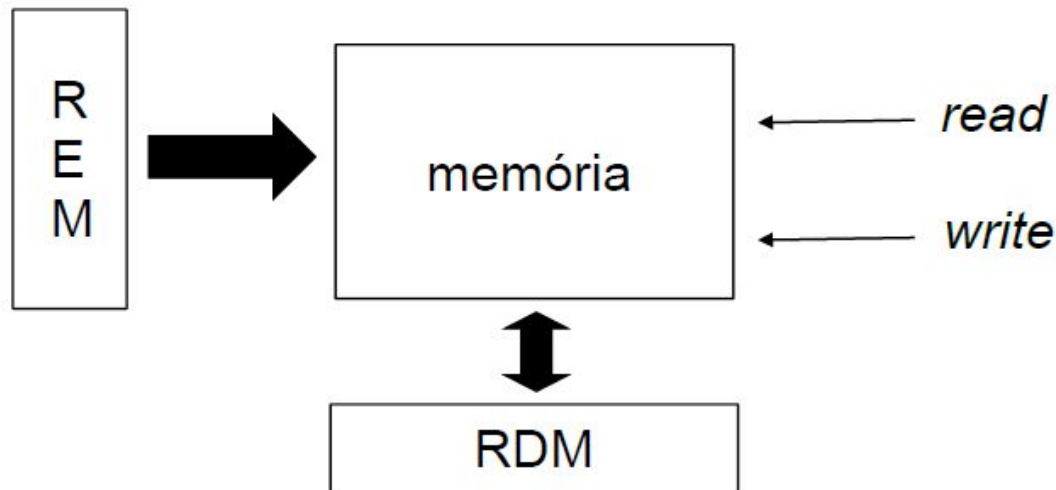
➤ **DECODIFICAÇÃO:** Sinais de controle são gerados de acordo com a informação presente no campo de operação.

➤ **EXECUÇÃO:** A execução da instrução se dá ao final da operação determinada após a decodificação. Ao término da execução da instrução, o ciclo é repetido.

# Busca-Decodificação-Execução de instruções

## ➤ BUSCA:

- Copia o apontador de programa (PC) para o registrador de programa (REM);
- Lê a instrução da memória;
- Copia o registrador de dados da memória (RDM) para o registrador de instruções (RI);
- Atualiza o apontador do programa (PC);



# Busca-Decodificação-Execução de instruções

- **DECODIFICAÇÃO:** Normalmente realizada por lógica combinacional.
- **EXECUÇÃO:** Exemplos de operações:
  - Cálculo do endereço de operandos;
  - Busca de operandos na memória;
  - Seleção de operação da ULA;
  - Carga de registradores;
  - Escrita de operandos na memória;
  - Atualização do PC para desvios.

O controle do ciclo busca-decodificação-execução é feito pela unidade de controle.

# Elementos funcionais básicos – unidade operacional

- **Unidade Operacional:**
  - Executa as transformações sobre **dados**, especificadas pelas **instruções do computador**;
  - Componentes: ULA, registradores de uso geral e específico, barramentos.
  - Cada organização possui:
    - número e tamanho de registradores variados em cada organização.
    - quantidade e tipo de operações variadas que a ULA realiza.



# Elementos funcionais básicos – unidade operacional

- **Unidade Lógica e Aritmética (ULA)**
  - Realiza operações lógicas e aritméticas;
  - Exemplo: Soma de dois operandos;
  - Negação de um operando;
  - Inversão de um operando;
  - Lógica de operando;
  - Rotação de um operando para a direita ou esquerda.
- As operações da ULA geralmente são bem simples;
- Funções complexas são realizadas pela ativação sequencial das várias operações básicas. Exemplo: multiplicação.

# Elementos funcionais básicos – unidade operacional

- A ULA fornece o resultado das operações e também algumas indicações sobre a operação realizada;
- Essas indicações são chamadas de códigos de condição;
- Exemplos:
  - Overflow
  - Sinal
  - Carry
  - Zero

Para que são utilizados os códigos de condição?

# Elementos funcionais básicos – unidade operacional

- **Acumulador**

- É um registrador e tem por função armazenar um operando e/ou um resultado fornecido pela ULA;
- Em computadores muito simples só são encontrados um acumulador;
- É ativado de acordo com o sinal de carga (*load*);
  - *Cada novo sinal faz perder o valor antigo*

# Elementos funcionais básicos – unidade de controle

- Serve para fornecer **sinais de controle**:
  - Gerenciar o fluxo interno de dados e o instante preciso em que ocorrem as transferências entre uma unidade e outra
- Cada unidade de controle comanda uma micro operação:
  - Responsável pela realização de uma carga em um registrador;
  - Seleção de dados para entrada;
  - Ativação de memória;
  - Seleção de uma operação da ULA.

# Elementos funcionais básicos – unidade de controle

- **Lógica combinacional:**
  - os sinais de saída são função exclusiva dos sinais de entrada.
- **Lógica sequencial:**
  - os sinais de saída são função dos sinais de entrada e do estado anterior. A unidade de controle utiliza máquinas de Estado Finitas (FSM).
- As máquinas de estados são circuitos sequenciais que utilizam flip-flops com sinal de clock para que as informações armazenadas possam ser atualizadas sincronamente a intervalos regulares (a cada pulso de clock).
- Existem várias formas de implementar a lógica sequencial. Porém duas são usuais:
  - Organização convencional
  - Organização microprogramada (por descrição de hardware).

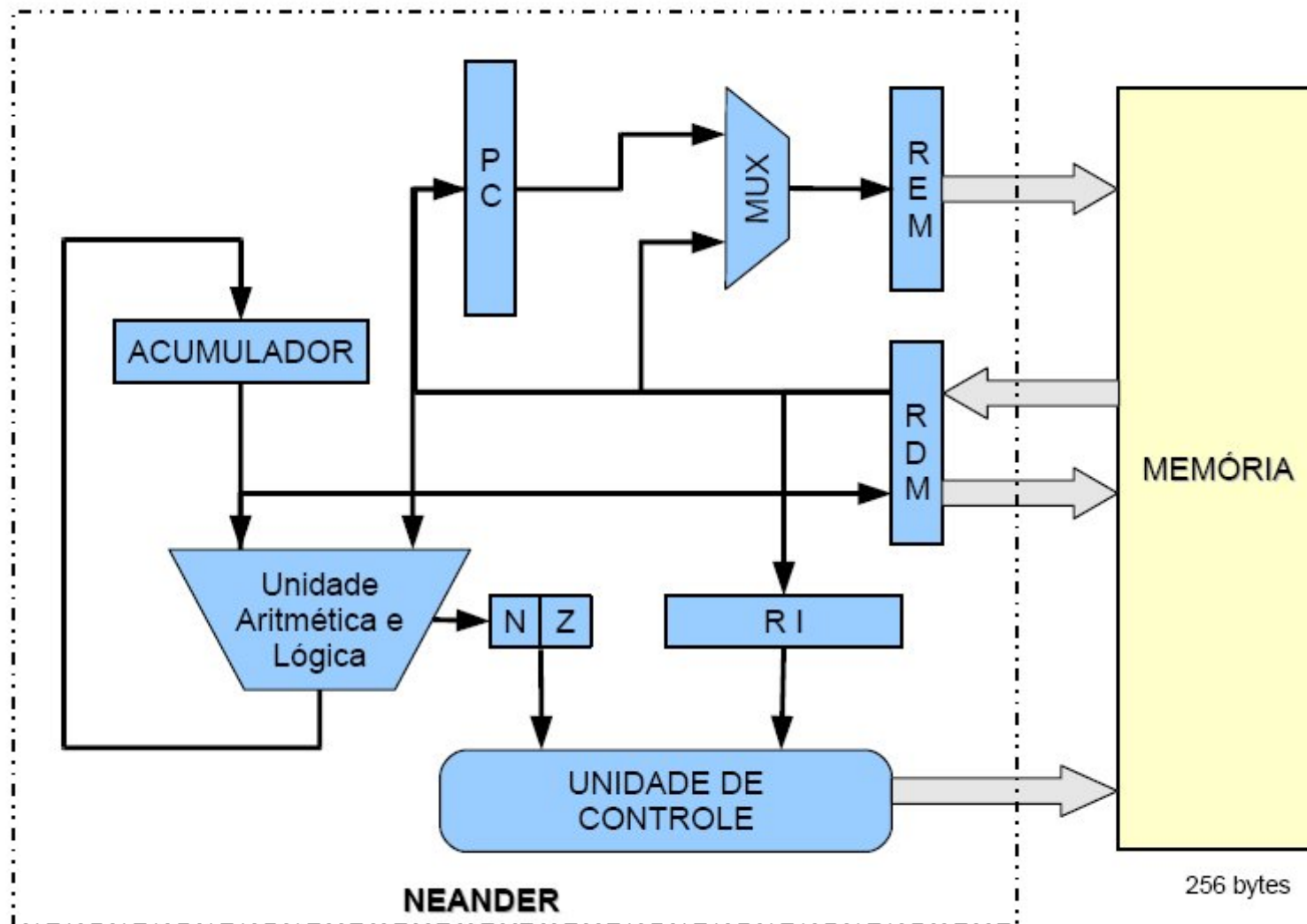
# Elementos funcionais básicos – registradores especiais

- Existem no computador alguns registradores com funções especiais. Depende da arquitetura e organização de cada máquina;
- Tipos:
  - Apontador de instruções (PC)
  - Registrador de instruções (IR)
  - Registrador de estado (RST)

# Elementos funcionais básicos – registradores especiais

- **Apontador de Instruções (PC)**
  - Tem como função manter atualizado o endereço de memória da próxima instrução;
- **Registrador de instrução (IR)**
  - Armazena a instrução que está sendo executada. De acordo com o conteúdo, a unidade de controle determina quais sinais deve ser gerados;
- **Registrador de estado (RST)**
  - Armazena códigos de condição gerados pela unidade lógica e aritmética;

# Organização do Neander





# Organização do Neander

## Instruções do Neander:

### Instrução STA

Busca:  $RI \leftarrow MEM(PC)$   
 $PC \leftarrow PC + 1$   
Execução:  $end \leftarrow MEM(PC)$   
 $PC \leftarrow PC + 1$   
 $MEM(end) \leftarrow AC$

### Instrução LDA

Busca:  $RI \leftarrow MEM(PC)$   
 $PC \leftarrow PC + 1$   
Execução:  $end \leftarrow MEM(PC)$   
 $PC \leftarrow PC + 1$   
 $AC \leftarrow MEM(end)$ ; atualiza N e Z

### Instrução ADD

Busca:  $RI \leftarrow MEM(PC)$   
 $PC \leftarrow PC + 1$   
Execução:  $end \leftarrow MEM(PC)$   
 $PC \leftarrow PC + 1$   
 $AC \leftarrow AC + MEM(end)$ ; atualiza N e Z

# Organização MIPS

