Curso de Informática
DCC/IM – UFRJ

Arquitetura de Computadores II

Arquiteturas Paralelas

Gabriel P. Silva

Bibliografia

- **Advanced Computer Architecture**
  - Dezsõ Sima, Terence Fountain, Péter Kacsuk
  - Addison-Wesley, 1997
- **Computer Architecture: A Quantitative Approach**
  - John L. Henessy e David A. Patterson
  - Morgan Kaufman Publishers, 3 edição, 2003

Aplicações Científicas

- Genoma Humano
- Turbulência dos Fluidos
- Dinâmica de Veículos
- Circulação de Oceanos
- Dinâmica de Fluidos Viscosos
- Modelagem de Supercondutores
- Cromodinâmica Quântica
- Visão por Computador
- Farmacêutica
- Biologia Estrutural
- Previsão do Tempo (+ 72 hs)

Conceitos Básicos

- Permite a execução das tarefas em menor tempo, através da execução em paralelo de diversas tarefas.
- O paralelismo pode ser obtido em diversos níveis, com ou sem o uso de linguagens de programação paralela.
- Arquiteturas de diversos tipos, elaboradas para aplicações específicas, podem ser utilizadas para acelerar a execução dessas aplicações.

Conceitos Básicos

- **Processo:**
  - Um processo é criado para execução de um programa pelo sistema operacional.
- **A criação de um processo envolve os seguintes passos:**
  - Preparar o descritor de processo;
  - Reservar um espaço de endereçamento;
  - Carregar um programa no espaço reservado;
  - Passar o descritor de processo para o escalonador.
- **Nos modernos S.O.s, um processo pode gerar cópias, chamadas de processos "filhos".**

Conceitos Básicos

- **Execução concorrente está associada a idéia de um servidor atendendo a vários clientes através de uma política de escalonamento no tempo.**
- **Execução paralela está associada ao modelo de vários servidores atendendo a vários clientes simultaneamente no tempo.**
- **As linguagens de programação podem então ser classificadas como seqüenciais, concorrentes e paralelas.**
- **Seqüenciais:**
  - C, Pascal, Fortran
- **Concorrentes:**
  - Ada, Pascal Concorrente, Modula-2, PROLOG Concorrente
- **Paralelas:**
  - Occam-2, 3L Parallel C, Strand-88

Medidas de Desempenho

- **Velocidade: tempo de resposta, vazão e utilização:**
  - **Vazão/Taxa *(Throughput):* taxa na qual os pedidos são atendidos (servidos) pelo sistema.**
  - **Utilização: fração do tempo em que o recurso permanece ocupado atendendo os pedidos dos usuários**
  - **Tempo de resposta: tempo decorrido entre o pedido e o início/conclusão da realização do serviço (latência).**
- **Confiabilidade**
  - **Probabilidade de erro**
  - **Intervalo entre erros**
- **Disponibilidade**
  - **Duração da falha**
  - **Intervalo entre falhas**

Vazão

- **Taxa na qual os pedidos são atendidos (servidos) pelo sistema.**
- **Exemplos:**
  - **Sistemas em lotes: jobs por segundo;**
  - **Sistemas interativos: pedidos por segundo;**
  - **CPUs: MIPs ou MFLOPs;**
  - **Redes: pacotes por segundo (pps) ou bits por segundo (bps);**
  - **Sistemas de Processamento de Transações: Transações por segundo (TPS);**

Medidas de Desempenho em Processamento Paralelo

- **Speed-up:**
  - Mede a razão entre o tempo gasto para execução de um algoritmo ou aplicação em um único processador e o tempo gasto na execução com *n* processadores

$$S(n) = T(1)/T(n)$$

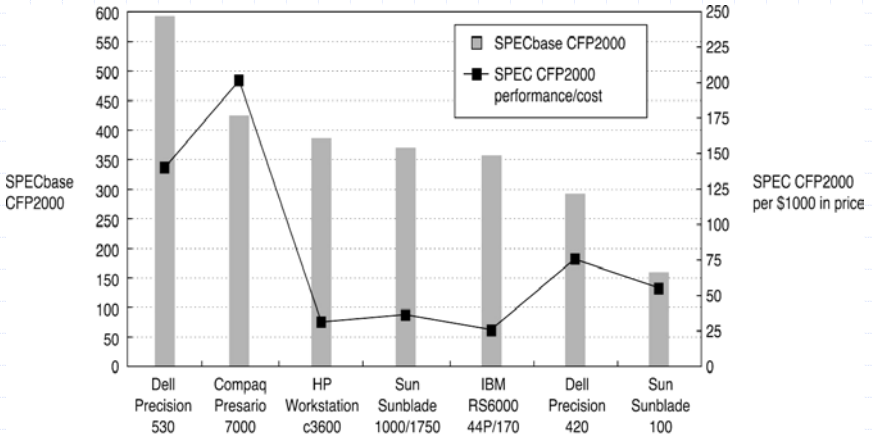
- **Eficiência:**

$$E(n) = S(n)/n$$

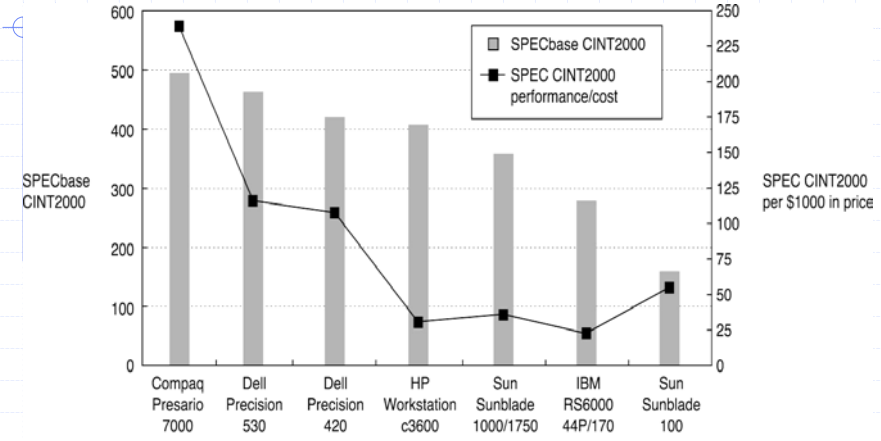
Medidas de Desempenho

- **Escalabilidade:**
  - Um sistema é dito *escalável* quando sua eficiência se mantém constante à medida que o número de processadores (*n*) aplicado à solução do problema cresce.
  - Se o tamanho do problema é mantido constante e o número de processadores aumenta, o "overhead" de comunicação tende a crescer e a eficiência a diminuir.
  - A análise da escalabilidade considera a possibilidade de se aumentar proporcionalmente o tamanho do problema a ser resolvido à medida que *n* cresce de forma a contrabalançar o natural aumento do "overhead" de comunicação quando *n* cresce.

Medidas de Desempenho



Medidas de Desempenho



Técnicas de Avaliação

- Medição
- Modelagem Analítica
- Simulação

Medição

- Para efetuarmos medições (com os "*benchmarks*" – programas de avaliação) é preciso termos à disposição ao menos um protótipo do sistema;
- Normalmente é difícil comparar alternativas;
- O método utilizado é compilar os "*benchmarks*" e executá-los no sistema que se deseja avaliar.
- Os programas devem ter características de acordo com os parâmetros que se seja avaliar (CPU, E/S, Gráfico, etc.).

Modelagem Analítica

- Teoria das filas
- Filas associadas a recursos
- Caracterização:
  - Processo de chegada
  - Processo de atendimento
  - Número de servidores
  - Tamanho máximo da fila
  - Política de atendimento da fila

Modelagem Analítica

- É uma técnica aproximada, ou seja, aproxima a realidade por um modelo;
- Se o modelo for simples e a aproximação boa, é possível avaliar facilmente compromissos entre alternativas;

Simulação

- Simulação de eventos discretos;
- Cada evento (ex.: chegada de usuário, término de serviço, etc.) é tratado quando do instante de sua ocorrência;
- Simula o comportamento de um sistema real;
- Em geral, é possível construir um modelo muito mais próximo da realidade do que com a teoria das filas;

Técnicas de Avaliação

Critério	Modelagem Analítica	Simulação	Medição
Estágio	Qualquer	Qualquer	Protótipo
Tempo Necessário	Pouco	Médio	Variado
Ferramentas	Analistas	Linguagens de Programação	Instrumen-tação
Precisão	Pouca	Moderada	Variada
Avaliação de Compromissos	Fácil	Moderada	Difícil
Custo	Baixo	Médio	Alto
Escalabilidade	Baixa	Média	Alta

Programas de Avaliação

- **É o processo de comparação entre dois ou mais sistemas através de medições em que são utilizados programas de avaliação (*benchmarks*) como cargas de trabalho *(workloads).* Podem ser de vários tipos:**
  - **Aplicações Reais: compiladores, processadores de texto e imagem são alguns exemplos de programas utilizados. Sofrem de problemas de portabilidade.**
  - **Aplicações Modificadas: E/S removida. "Scripts" são utilizados para simular interatividade.**
  - **Kernels: pequenos trechos de código de programas reais são extraídos de programas reais. Ex: Livermoore loops e Linpack.**

Programas de Avaliação

- ***Benchmarks* de brinquedo: tipicamente entre 10 e 100 linhas de código para produzir um resultado previamente conhecido. Exemplos são o "Crivo de Eratóstenes", Quicksort e Puzzle. São fáceis de digitar e executam em quase qualquer computador.**

- ***Benchmarks* sintéticos: tentam imitar a freqüência de execução média das instruções esperada em uma aplicação de um determinado tipo, mas não são extratos de programas reais. Whetstone (ponto flutuante) e Dhrystone (inteiro) são os exemplos mais populares.**

Programas de Avaliação

- **Suítes de Avaliação:** coleção de programas de avaliação que tentam medir o desempenho dos processadores para uma variedade de aplicações.
- O exemplo de maior sucesso é o SPEC, que possui versões inteiras e de ponto flutuante, com edições em 1989, 1992, 1995 e 2000. Estão divididos em duas grandes categorias: inteiros e de ponto flutuante.
- O SPEC CPU2000 é constituído por 11 programas de avaliação inteiros (CINT2000) e 14 de ponto-flutuante (CFP2000).
  - http://www.spec.org
  - http://www.top500.org

Paralelismo

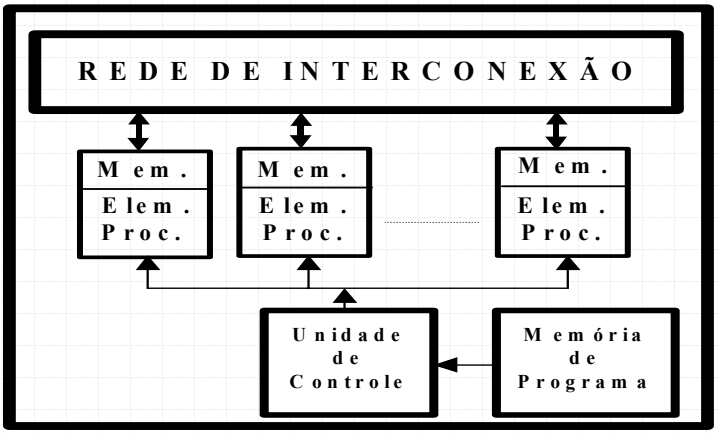
- Níveis de paralelismo :
  - nível de instrução (granulosidade fina): arquiteturas pipelined, superescalares e VLIW
  - nível de "thread" (granulosidade média): arquiteturas multithreading, SMT
  - nível de processo (granulosidade grossa): multiprocessadores e multicomputadores

Paralelismo

- Classificação de Flynn (1966):
  - Single Instruction Stream (SI)
  - Multiple Instruction Streams(MI)
  - Single Data Stream (SD)
  - Multiple Data Streams (MD)
- Categorias de Arquiteturas
  - SISD (Processadores Convencionais)
  - SIMD (Processadores Vetoriais)
  - MIMD (Multiprocessadores)

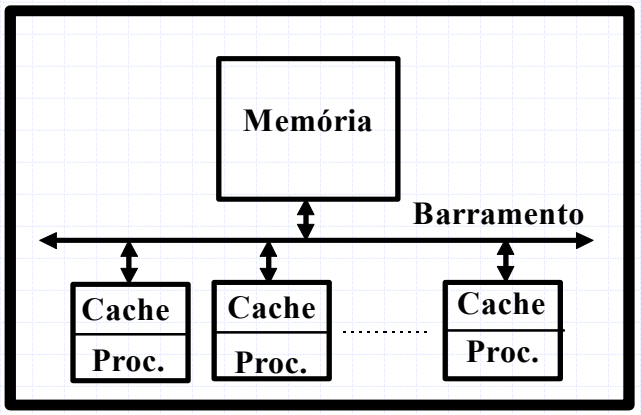
Arquitetura SIMD

- Arquitetura SIMD Básica:



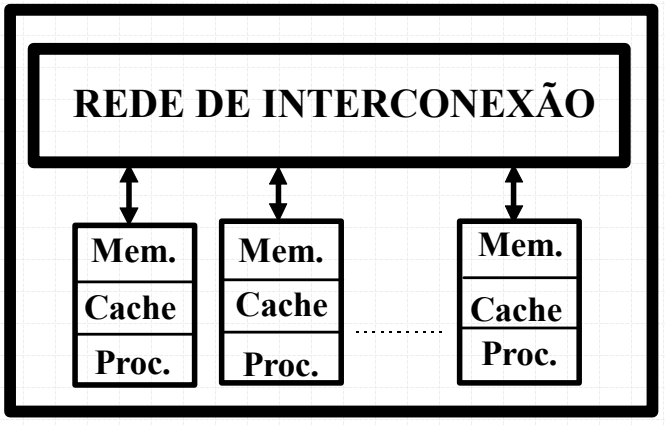
Arquitetura MIMD

- Arquitetura de Memória Centralizada



Arquitetura MIMD

- Arquitetura de Memória Distribuída



Modelos de Programação

- Modelos Básicos de Programação:
  - Memória Compartilhada:
    - Todos os processos/threads compartilham de um espaço de endereçamento comum;
    - Comunicação através da memória;
    - Uso de semáforos para sincronização;
    - Linguagens: Pascal e "C" Concorrente.
  - Memória Distribuída:
    - Os processos/threads não dispõem de um espaço comum para sincronização e comunicação por troca de mensagens;
    - Linguagens: PVM, MPI, Occam-3.

Arquiteturas SIMD

- Processadores Vetoriais
- Arquiteturas SIMD
- Arquiteturas Sistólicas

Processadores Vetoriais

- Os processadores vetoriais são arquiteturas "pipelined" do tipo SIMD, ou seja, uma única instrução opera sobre vários dados, no caso, um vetor.
- Um processador vetorial possui instruções para operar em vetores: *um conjunto linear de valores.*
- Uma operação vetorial típica pode adicionar dois vetores com 64 elementos em notação ponto flutuante para obter um único vetor de resultado com 64 elementos.

Processadores Vetoriais

- A instrução vetorial é equivalente a um "loop" inteiro, onde cada iteração computa um dos 64 elementos do resultado, atualiza os índices e retorna para o início.
- As **instruções vetoriais** possuem as seguintes características:
  - Cada instrução equivale a um **loop**
  - O cálculo de cada resultado não depende de resultados anteriores ➔ **é possível haver pipelines profundos sem a ocorrência de dependências de dados**
  - O padrão de acesso à memória para a busca dos operandos é conhecido e regular ➔ **benéfico utilizar memória com interleaving**

Processadores Vetoriais

- Podem ser de dois tipos:
  - Memória-Memória
    - Arquiteturas mais antigas
    - Todas as operações vetoriais manipulam operandos na memória
  - Registrador-Registrador
    - Arquitetura usadas em todos os processadores vetoriais mais recentes
    - Todas as operações vetoriais, com exceção de load e store, trabalham com registradores
- Unidades funcionais são organizadas como pipelines profundos

Processadores Vetoriais

- Exemplo:

  Vetor C ← Vetor A * Vetor B

- Estágios do Pipeline:

Lê elementos dos vetores A e B	→ LA, LB
Soma expoentes	→ SE
Multipica mantissas	→ MM
Normaliza resultado	→ NR
Armazena resultado em C	→ AC

Exemplo de Processamento Vetorial

LA/LB	1	2	3	4	5	6	7	8	9	10
SE		1	2	3	4	5	6	7	8	9
MM			1	2	3	4	5	6	7	8
NR				1	2	3	4	5	6	7
AC					1	2	3	4	5	6

Exemplo de Processamento Vetorial

Tempo de execução → T = [5 + (N-1)] * t

N → Número de elementos do vetor

t → Perído do Clock

- **Supondo que os vetores A e B possuam cada um 10.000 elementos e que a freqüência do clock dos pipelines é de 200 MHz, temos, que o tempo total gasto para executar a operação de cálculo de todos os elementos do Vetor C é dado por:**
- ***T = (5 + (10.000 - 1)) x 5 ns = 50 μs (aprox.)***
- **200 MFLOPS**

Problemas para a vetorização

- **Tamanho dos vetores:**
  - Quando o tamanho dos vetores é maior do que o dos registradores vetoriais, o compilador deve quebrar a operação vetorial em uma seqüência de operações com vetores de tamanho igual ou menor do que o dos registradores vetoriais ➔ "overhead" introduzido pelo esvaziamento dos pipelines e operações adicionais de load/store.

Problemas para a vetorização

- **Stride**
  - Em operações de multiplicação de matrizes, por exemplo, necessariamente o acesso ao vetor que armazena os elementos de uma das matrizes não será feito de forma seqüencial, já que as matrizes são armazenadas na memória de forma ordenada, ou por linha ou por coluna.
  - O salto entre os endereços de dois elementos consecutivos a serem utilizados é chamado ***stride.***
  - Em geral estão disponíveis instruções vetoriais do tipo *load/store* com a definição do valor do *stride*.

Problemas para a vetorização

- A existência de desvios condicionais em "loops":

```
for (i=1; i<256; i++)
        if (a[i] < 0)
                a[i] = a[i] + b[i];
```

Problemas para a vetorização

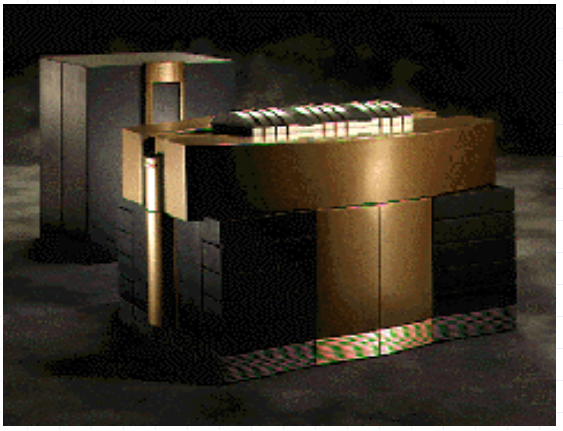
- Para que operações como a de soma do exemplo anterior possam ser vetorizadas, as arquiteturas de processamento vetorial devem suportar os chamados ***vetores de máscara*** associados às operações vetoriais, indicando sobre que elementos dos vetores devem ser efetuadas as operações, em função do resultado do teste relativo ao desvio condicional.

Supercomputadores Vetoriais Comerciais

- **NEC-SX5**
  - Até 16 processadores
  - Período de Clock: 4 ns
  - Desempenho de Pico: 128 GFLOPS (123)
- **Cray T90 ***
  - Até 32 processadores
  - Período de Clock: 2.2 ns
  - Desempenho de Pico: 56 GFLOPS (36.6)
- **Fujitsu VPP-5500U**
  - Um processador
  - Período de clock: 3,3 ns
  - Desempenho de Pico: 9,6 GFLOPS (8,7)

Supercomputadores Vetoriais Comerciais

- **Cray T90**



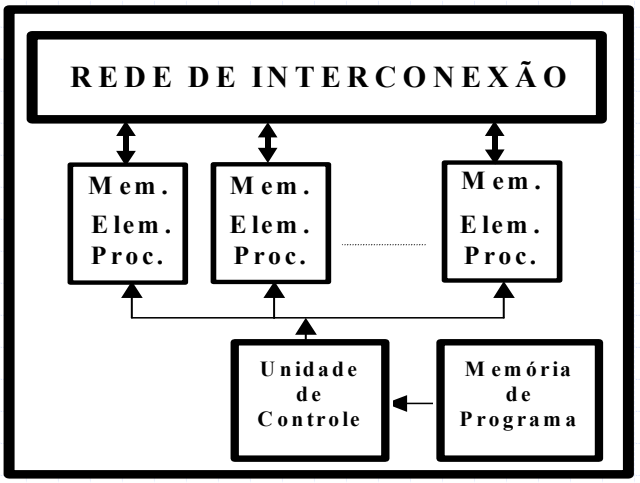
Arquiteturas SIMD

- É uma classe importante de processadores, devido a fatores como:
  - Simplicidade de conceitos e programação
  - Regularidade da estrutura
  - Facilidade de escalabilidade em tamanho e desempenho
  - Aplicação direta em uma série de aplicações que requerem paralelismo para obter o desempenho necessário.

Arquiteturas SIMD

- Processadores executam sincronizadamente a mesma instrução sobre dados diferentes.
- Utiliza vários processadores especiais muito mais simples, organizados em geral de forma matricial.
- Muito eficiente em aplicações onde cada processador pode ser associado a uma sub-matriz independente de dados (processamento de imagens, algoritmos matemáticos, etc.).

Arquitetura Básica



Opções Arquiteturais

- **Granulosidade**
  - É a relação entre o número de elementos processadores e o paralelismo do conjunto de dados sobre os quais opera.
  - Um sistema com poucos elementos de dados por elemento processador é qualificado como granulosidade fina.
  - Não é prático termos apenas um elemento processador por elemento de dados nas arquiteturas SIMD.
- **Conectividade**
  - Vizinhança (Malha ou Toro)
  - Árvore
  - Pirâmide
  - Hipercubo

Opções Arquiteturais

- Complexidade do Processador

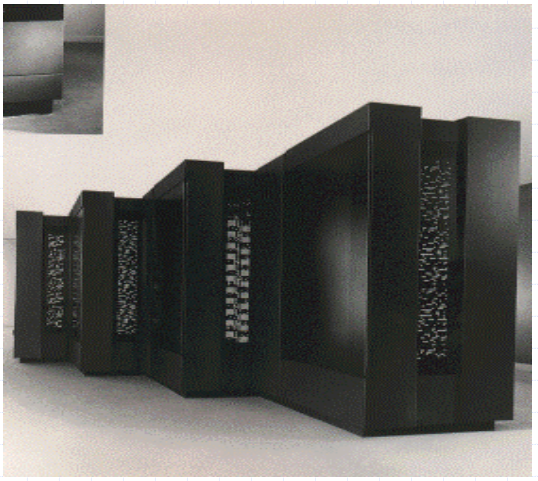
Single-Bit	Granulosidade fina. Usualmente utilizados para processamento de imagem
Inteiro	Uma solução de compromisso. Utilizados para visão ou computação genérica
Ponto Flutuante	Granulosidade grossa. Usualmente utilizados para computação científica.

Exemplos

- **Illiac IV**
  - Matriz 8 x8 (1968)
- **MPP ***
  - Matriz 128 x128 (1983)
- **Connection Machine ***
  - Hipercubo (16K x 4) processadores (1985)
- **MasPar**
  - Multi-level crossbar switch (1990)
  - 16K processadores

Thinking Machines

- CM-5



Thinking Machines

- CM5 fat tree network

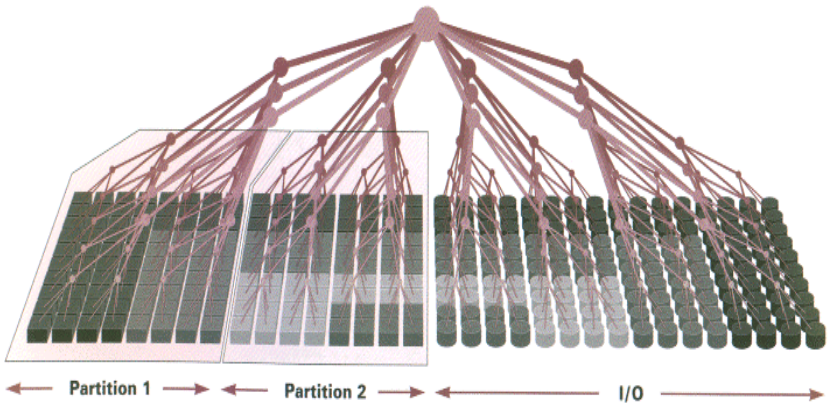


Figure 2

Arquiteturas Sistólicas

- Termo utilizado por H.T. Kung, em 1978, em analogia com o funcionamento do coração.
- Uma aplicação típica é a multiplicação de matrizes:

$$C_{mn} = \sum_{ij} A_{in} B_{mj}$$

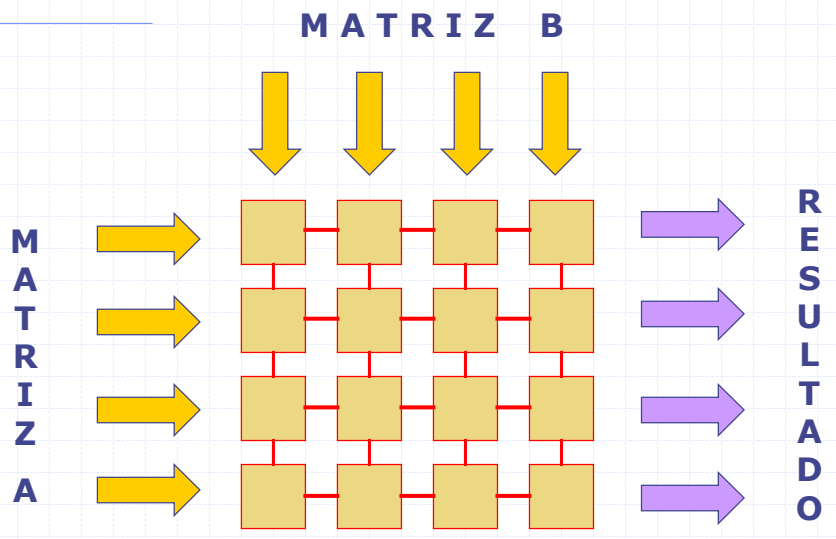
Características Básicas

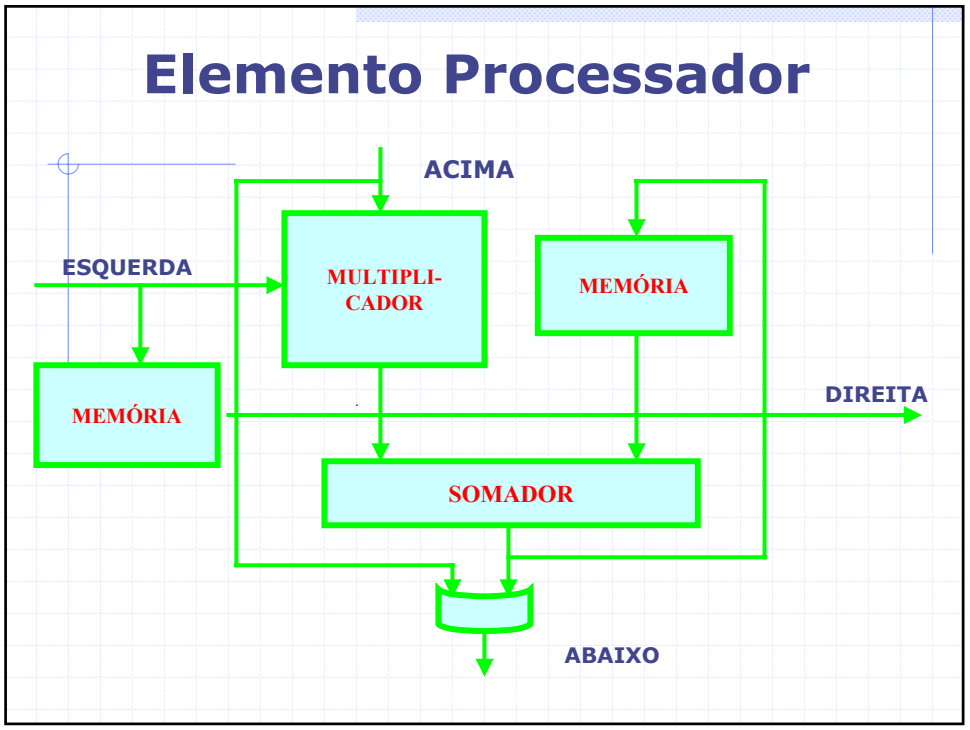
- Possui as seguintes características:
  - Cada elemento do arranjo computa um função simples e única
  - Cada elemento está conectado apenas aos seus vizinhos mais próximos através apenas de um caminho de dados uni-direcional
  - O único sinal de controle enviado através do arranjo é um pulso de relógio, utilizado para sincronizar as operações
  - Os dados de entrada são inseridos nos dois lados da matriz e passam através do arranjo em duas direções ortogonais
  - Os dados de saída são extraídos como uma série de elementos de um dos dois lados restantes da matriz

Características Básicas

- **Dimensão:**
  - 1, 2 ou mais dimensões
- **Precisão:**
  - 1 bit ou multi-bit
  - Inteira ou ponto-flutuante
- **Conectividade**
  - Linear, matriz, hexagonal, etc.
  - Uni-direcional ou bi-direcional
- **Sincronicidade**
  - Fixa ou programável

Organização da Matriz




Elemento Processador
ACIMA
ESQUERDA
MULTIPLI-
CADOR
MEMÓRIA
MEMÓRIA
DIREITA
SOMADOR
ABAIXO