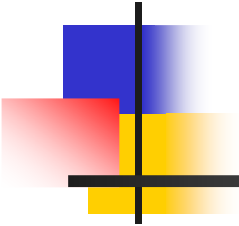
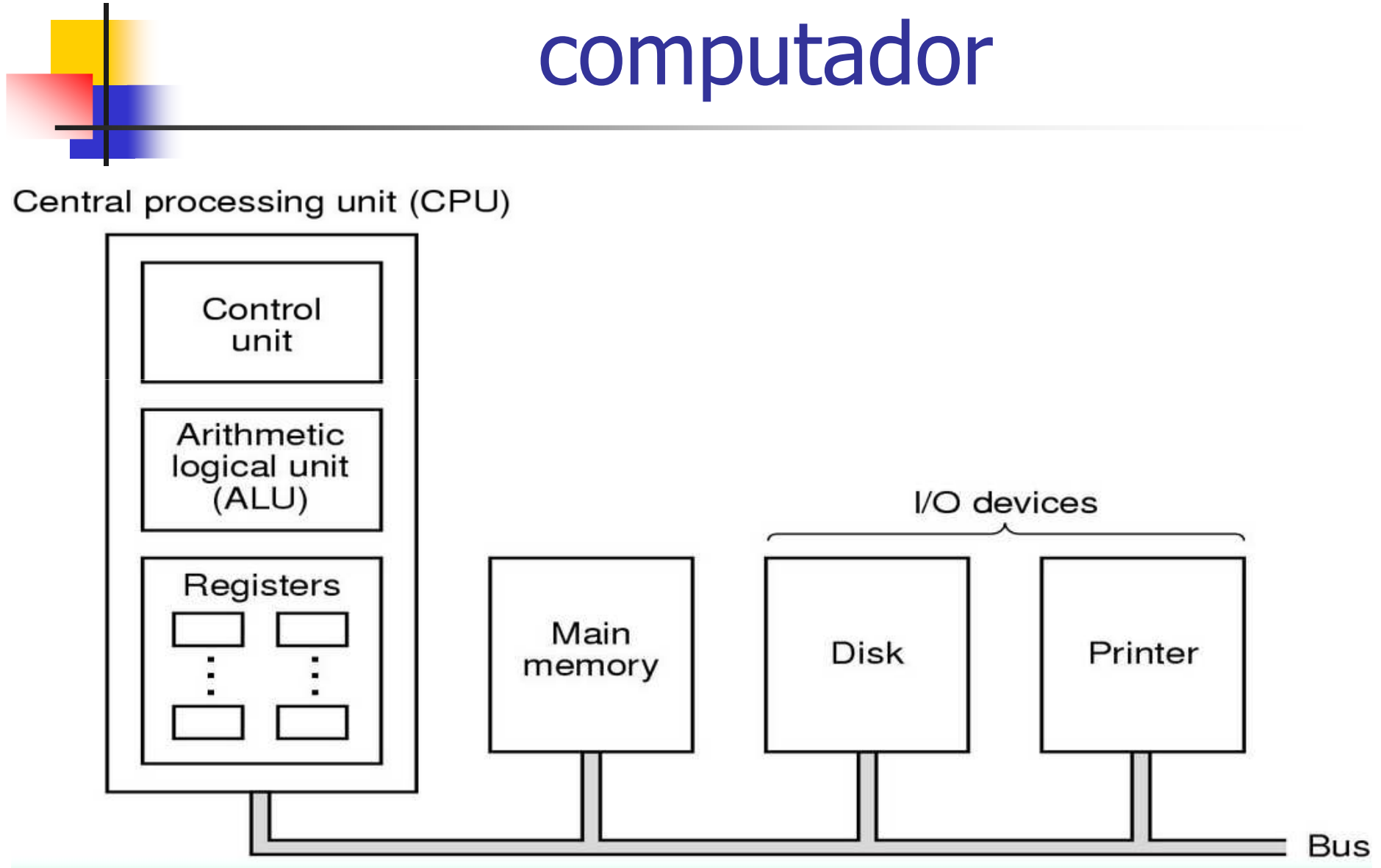


CPU (Processadores)



Prof. Tiago Gonçalves Botelho

Organização Geral de um computador



Processador

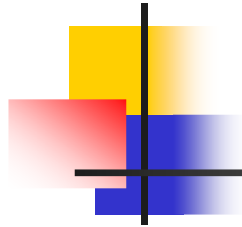
- **Conceito conotativo:** É o cérebro do computador;
- **Função:** Executar os programas armazenados na memória principal; **Como?** Buscando cada instrução na memória, examinando-a e executando uma após outra.





Barramentos

- **Conceito:** Conjuntos de fios paralelos que permite a transmissão de dados, endereços, sinais de controle e instruções;
- **Tipos:** Externos ou internos a CPU



Componentes do processador

- **1. Unidade de controle:**
busca instruções na memória principal e define seu tipo;
- **2. Unidade Lógica Aritmética (U.L.A.):**
realiza as operações necessárias a execução das instruções;



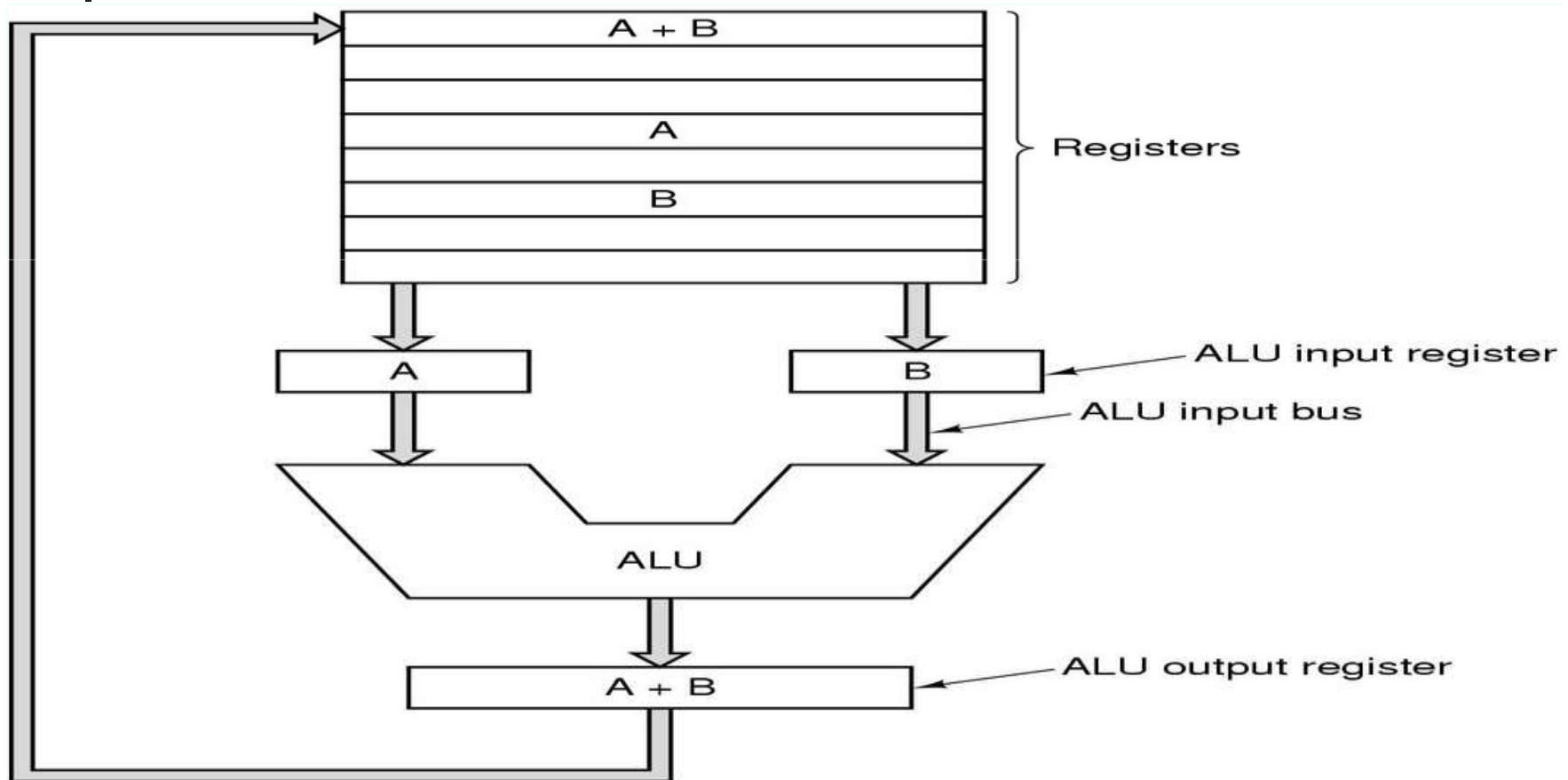
Componentes do processador

■ 3. Registradores:

- memória pequena de alta velocidade;
- em geral todos de tamanhos iguais;
- *Program Counter* (PC): armazena o endereço da próxima instrução;
- *Instruction Register* (IR): armazena instrução que está sendo executada

Organização do processador

Caminho de dados para uma máquina típica de Von Neumann.





Organização do processador

Caminho de dados:

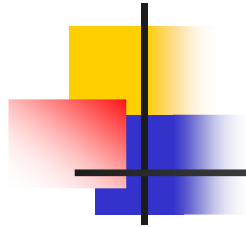
- Parte do processador com registradores (1 a 32), ALU e barramentos;
- Dois registradores armazenam as duas entradas (A e B) da ALU;
- A saída da ALU é conectada a 1 dos registradores;
- Existem 2 categorias de instruções:
 - *instrução registrador-memória*: que permite que uma palavra de memória seja armazenada no registrador, e vice-versa;
 - *instrução registrador-registrador*: instrução que opera sobre 2 registradores e coloca a saída em outro registrador (ciclo de caminho de dados)

Importante: *A velocidade do caminho de dados determina a velocidade do computador*



Execução de instrução

- Ciclo BUSCA/DECODIFICA/EXECUTA
 1. Busca próxima instrução na memória e armazena no IR;
 2. Atualiza Contador de instrução PC para apontar para a próxima instrução;
 3. Determina tipo de instrução armazenada no IR;
 4. Determina endereço dos dados na memória, se a instrução requer dados adicionais;
 5. Busca palavras (dados) na memória, caso a instrução precisar, e armazena-as em outros registradores;
 6. Executa instrução;
 7. Volta ao passo 1.



Qual arquitetura utilizar?

- Alto custo: Uma implementação em hardware puro (sem interpretação de instrução);
- Baixo custo: Implementação com interpretador de software (por software).



Vantagens do interpretador em relação ao hardware puro

- Capacidade de corrigir no campo eventuais erros na implementação de instruções;
- Oportunidade de incorporar novas instruções nas máquinas já existentes;
- Projeto estruturado que permite o desenvolvimento, teste e documentação de instruções complexas de maneira muito eficiente. Pode inclusive substituir implementações antigas de instruções;
- Armazenamento das microinstruções do interpretador em memórias read-only (ROM), chamadas de memória de controle, muito mais rápidas do que as memórias convencionais.



Decadência da implementação via hardware

- Criação de um conjunto muito grande de instruções difíceis e caras para implementação via hardware.



RISC *versus* CISC

❖ RISC

- Nova tecnologia para máquinas de alta performance;
- Máquina com conjunto reduzido de instruções básicas em hardware;
- Uso de chips processadores VLSI (Very Large Scale Integration) sem interpretação;
- Demais instruções eram geradas por combinação das instruções básicas de hardware.



RISC *versus* CISC

❖ CISC

- Tecnologia mais antiga e usada para famílias de computadores compatíveis a nível de software;
- Número maior de instruções;
- Uso extensivo de interpretação.



Porque então a tecnologia RISC não suplantou a CISC ?

- Problemas de compatibilidade com máquinas antigas com software já desenvolvido;
- Aparecimento de soluções híbridas: Por exemplo, a INTEL usa RISC para instruções de uso mais frequentes (*Núcleo RISC*) e interpretação para instruções mais complexas e de uso menos frequentes.



Princípios de Projeto para computadores Modernos

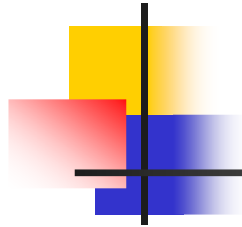
❖ Projeto RISC

- Todas as instruções são diretamente executadas por hardware;
- Maximizar a Taxa à qual as instruções são executadas;
- As instruções precisam ser facilmente decodificadas;
- Somente as Instruções de Load e Store devem referenciar a Memória;
- Projetar uma máquina com muitos registradores (≥ 32).
- Uso de paralelismo.



Paralelismo ao Nível de Instruções

- Pipeline: divide a execução de instruções em várias partes, cada uma das quais tratada por um hardware dedicado exclusivamente a ela.



Paralelismo ao Nível de Instruções

- **Funcionamento de um pipeline de 5 estágios:**
- O estágio S1 busca a instrução da memória e armazena num buffer até chegar a hora de executá-la;
- No estágio S2 ocorre a decodificação da instrução, determinando tipo e operandos;
- No estágio S3 ocorre a busca dos operandos na memória ou nos registradores;
- No estágio S4 temos a execução - passagem pelo caminho de dados;
- No estágio S5 o resultado do processamento é escrito num registrador.

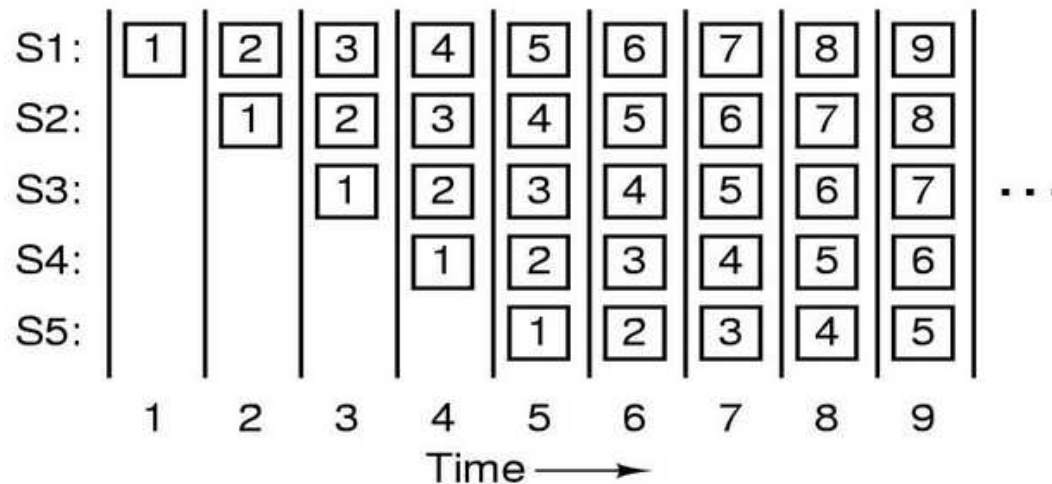
Paralelismo ao Nível de Instruções

(a) Pipeline de 5 estágios.

(b) Estado de cada um dos estágios como função do tempo. Estão ilustrados nove períodos de clock.



(a)

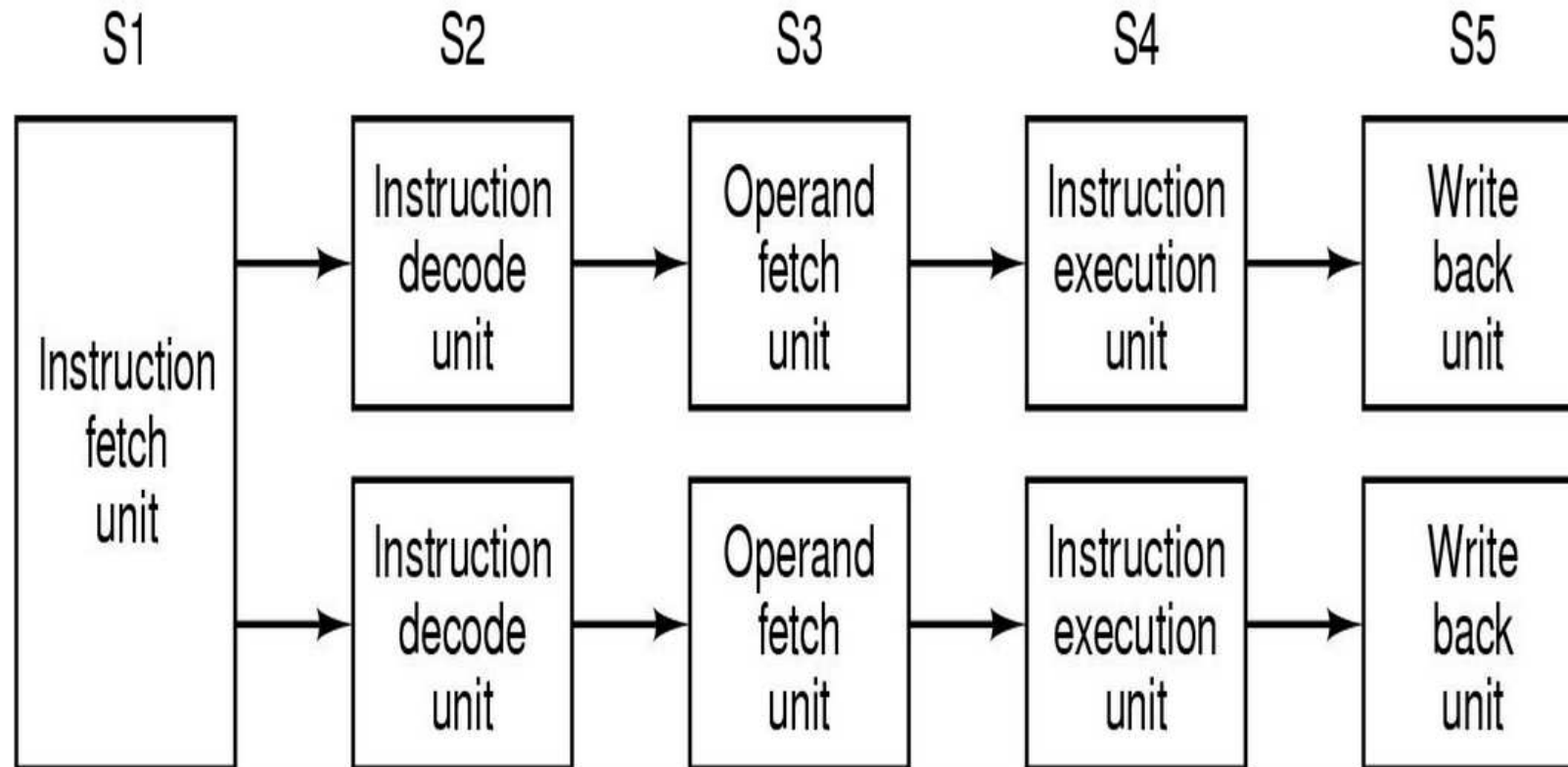


(b)

Arquiteturas Superescalares

- Pipelines duplos:

Caso 1 - Dois pipelines de 5 estágios com uma unidade de busca de instruções comum a ambos.





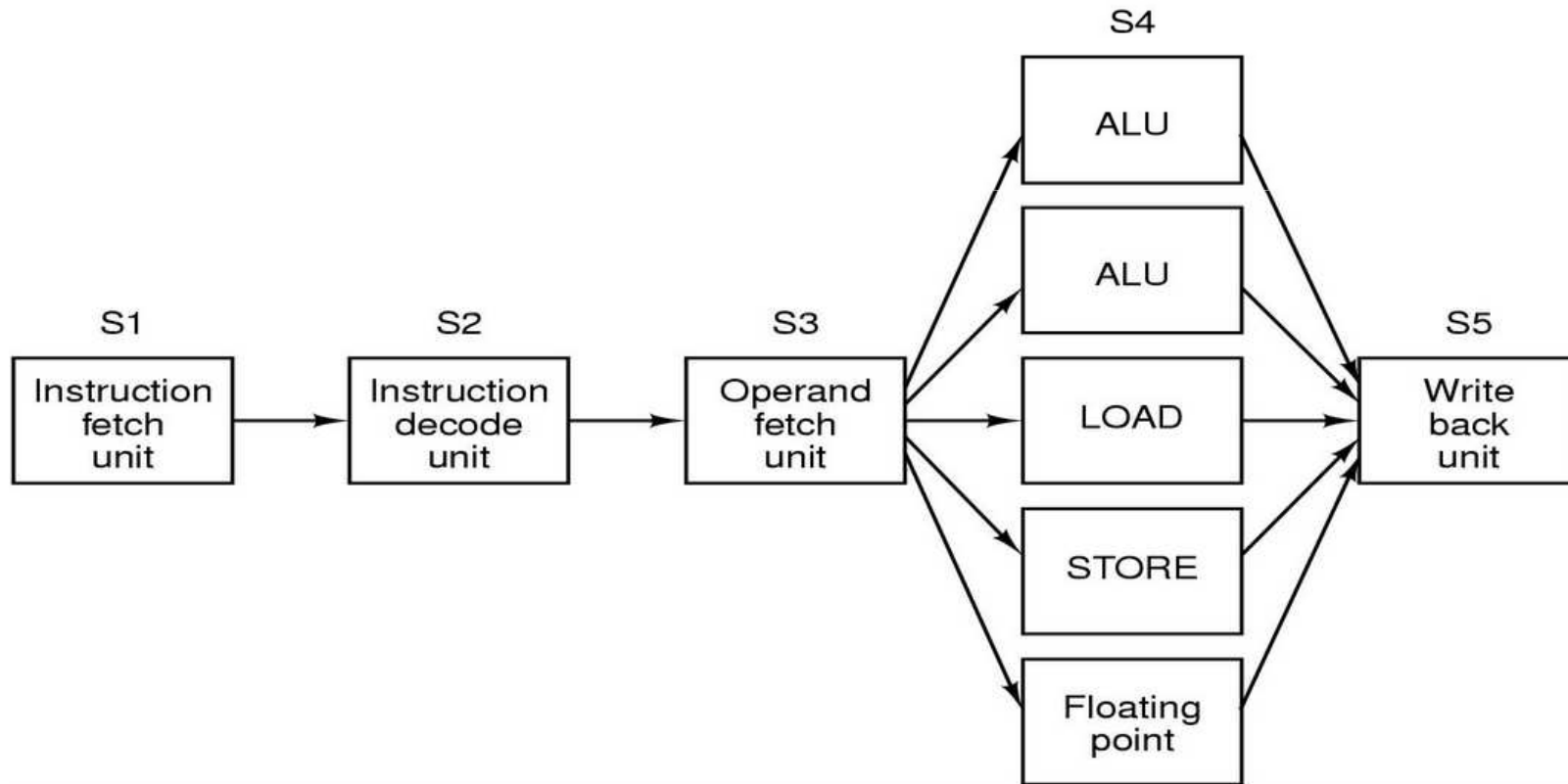
Arquiteturas Superescalares

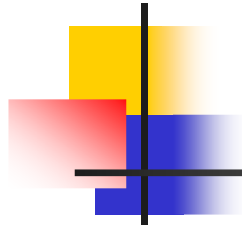
- Restrições para implementação de pipelines duplos:
 - não pode haver conflitos pelo uso de recursos (mesmo registro, por exemplo);
 - o resultado de uma instrução não pode depender do resultado da outra;
 - pode se pensar em pipelines com leitura inicial de 3 ou mais instruções porém hardware fica complexo.

Arquiteturas Superescalares

- Pipeline com diversas unidades funcionais:

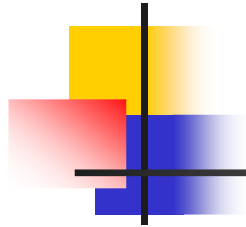
Caso 2 - O estágio 3 distribuindo instruções para o estágio 4.





Paralelismo ao Nível do Processador

- À medida que os processadores vão ficando mais rápidos, surgem limitações:
 - Ordem física – velocidade da luz;
 - Dissipação de energia – produção de calor pelo chip.



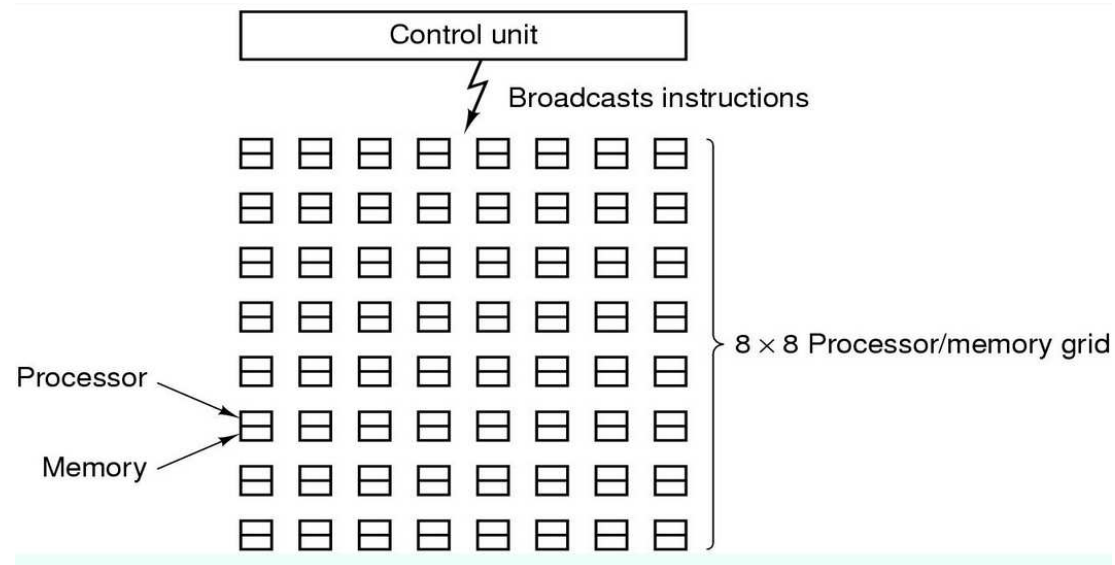
Paralelismo ao Nível do Processador

Método	Velocidade
1 Processador em pipeline ou superescalar	5 a 10 vezes mais rápido
+ de 1 processador	50 a 100 vezes mais rápido

Paralelismo ao Nível do Processador

■ Processador matricial

- composto de grande número de processadores idênticos;
- cada processador executa a *mesma sequência de instruções* sobre *diferentes conjuntos de dados*;
- tem uma única unidade de controle;
- tem uma ULA para cada processador.





Paralelismo ao Nível do Processador

- Processador Vetorial
 - operações aritméticas são executadas numa única ULA, que opera em pipeline;
 - operandos são colocados em um *registro vetorial* para serem processados na ULA.



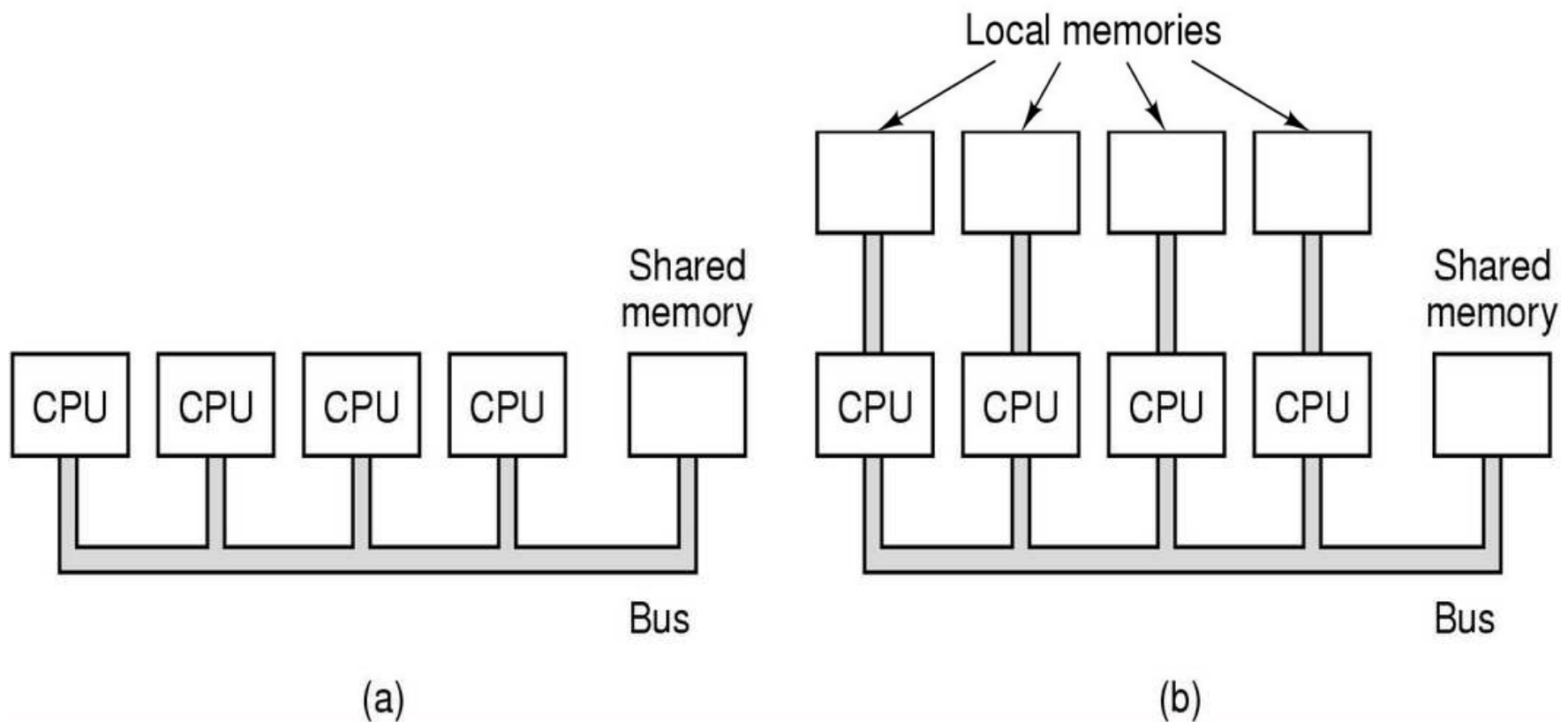
Paralelismo ao Nível do Processador

❖ **Multiprocessadores**

- é composto de vários processadores independentes;
- compartilham uma mesma memória por um barramento principal (a);
- ou compartilham uma memória e tem memórias locais (b):
 - executam processamentos locais
 - liberam tráfego do barramento principal
 - é necessário gerenciar conflitos

Paralelismo ao Nível do Processador

❖ Multiprocessadores





Paralelismo ao Nível do Processador

❖ **Multicomputadores**

- Sistemas com um grande número de computadores interconectados
- Não existe nenhum tipo de memória comum sendo compartilhada
- Comunicação entre computadores é feita através de troca de mensagens a uma velocidade bem alta
- Computador pode não precisa estar ligados diretamente com todos os outros (uso de topologias em árvore, anéis, etc..)
- Mensagens são roteadas do computador fonte para o destino (usando computadores intermediários)



Bibliografia:

- Tanenbaum, A. S. Organização Estruturada de Computadores. 5 ed – Editora Pearson, 2007.