Organização e Arquitetura de Computadores

Paralelismo

Prof. Msc. Francisco de Assis de Souza Rodrigues

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Objetivos e Motivação

Conhecer um processo para otimizar o tempo de execução de instruções



O que é Pipeline?

- Uma forma da CPU buscar mais de uma instrução a ser executada
- Implementada em Hardware
- Instruções são colocadas em uma fila para serem executadas
- Forma de implementar o paralelismos na execução das instruções

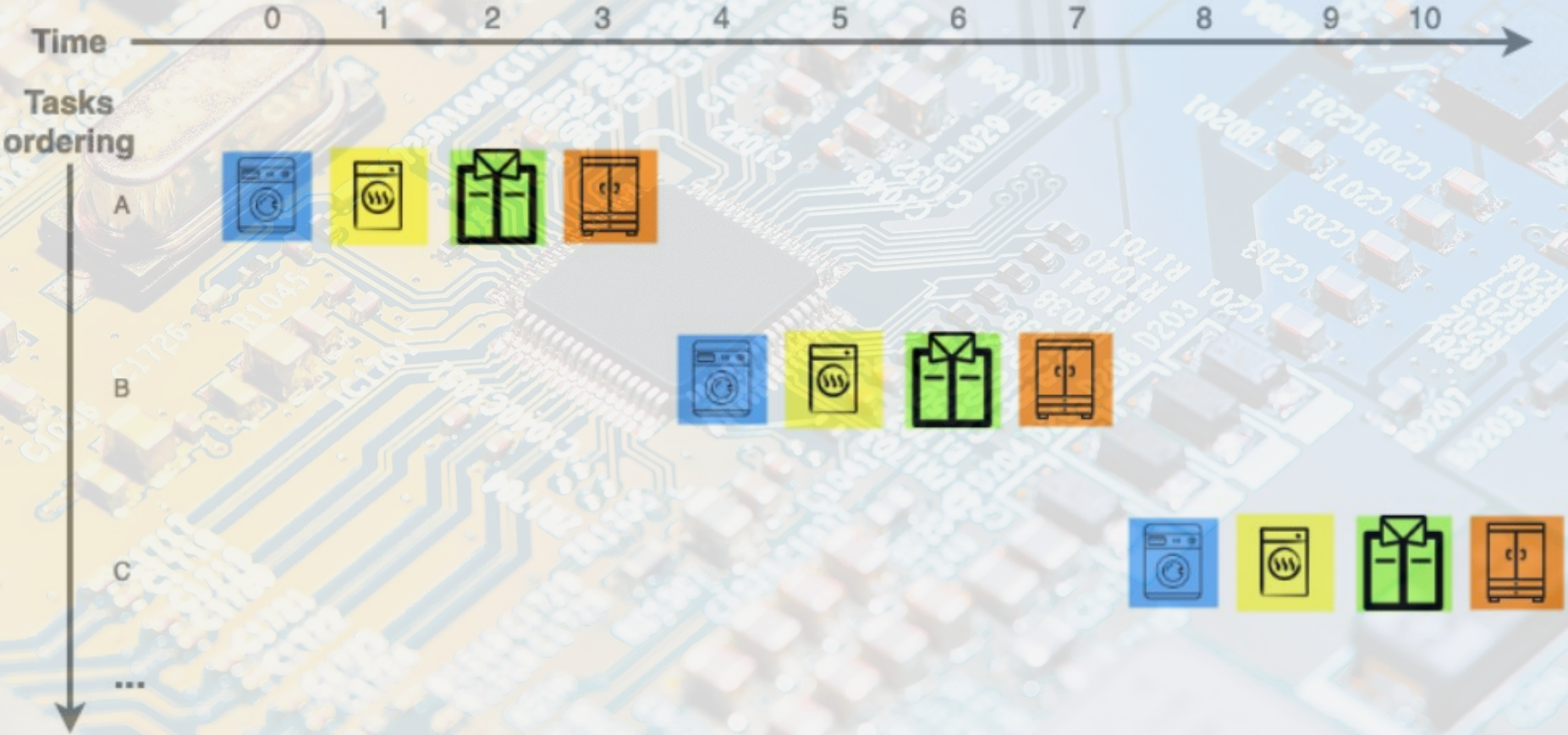


Para que serve

- Acelerar a execução das instruções
  - As próximas instruções não precisam ser buscadas em memória.

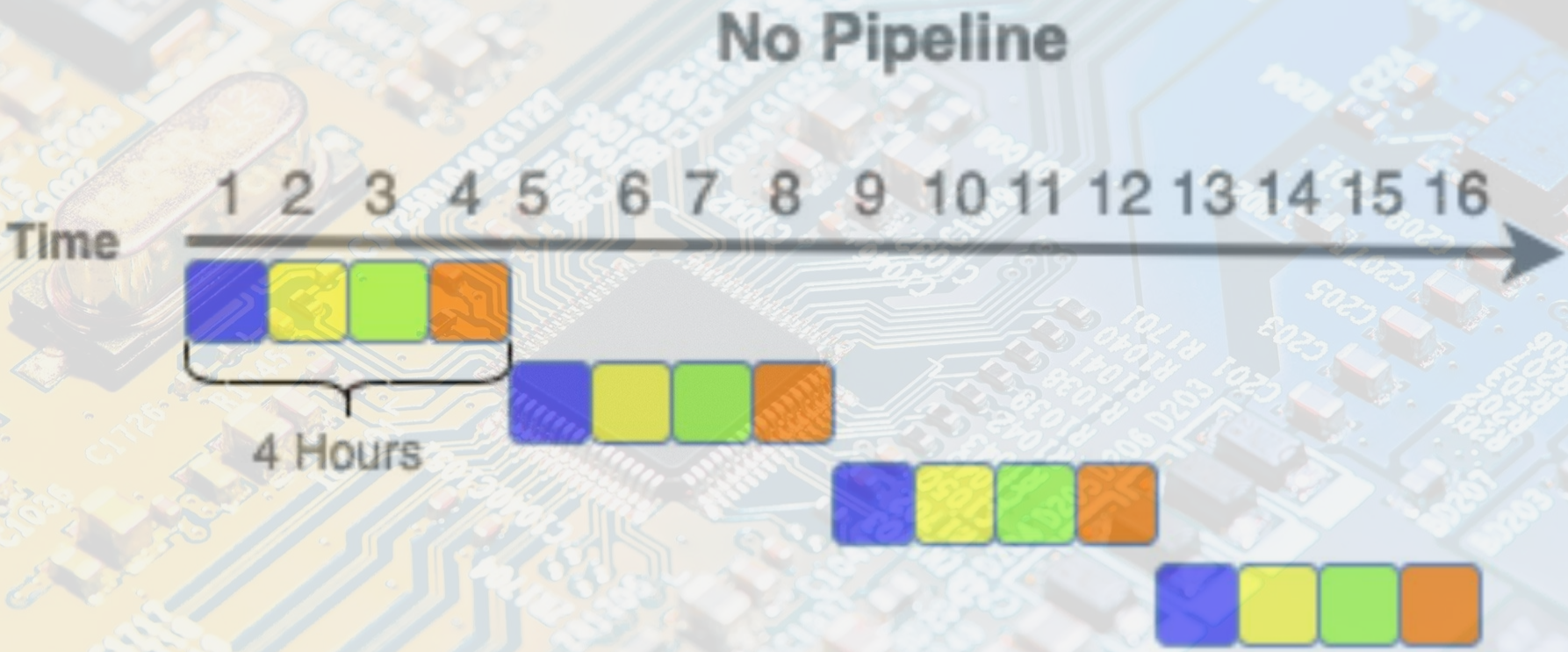


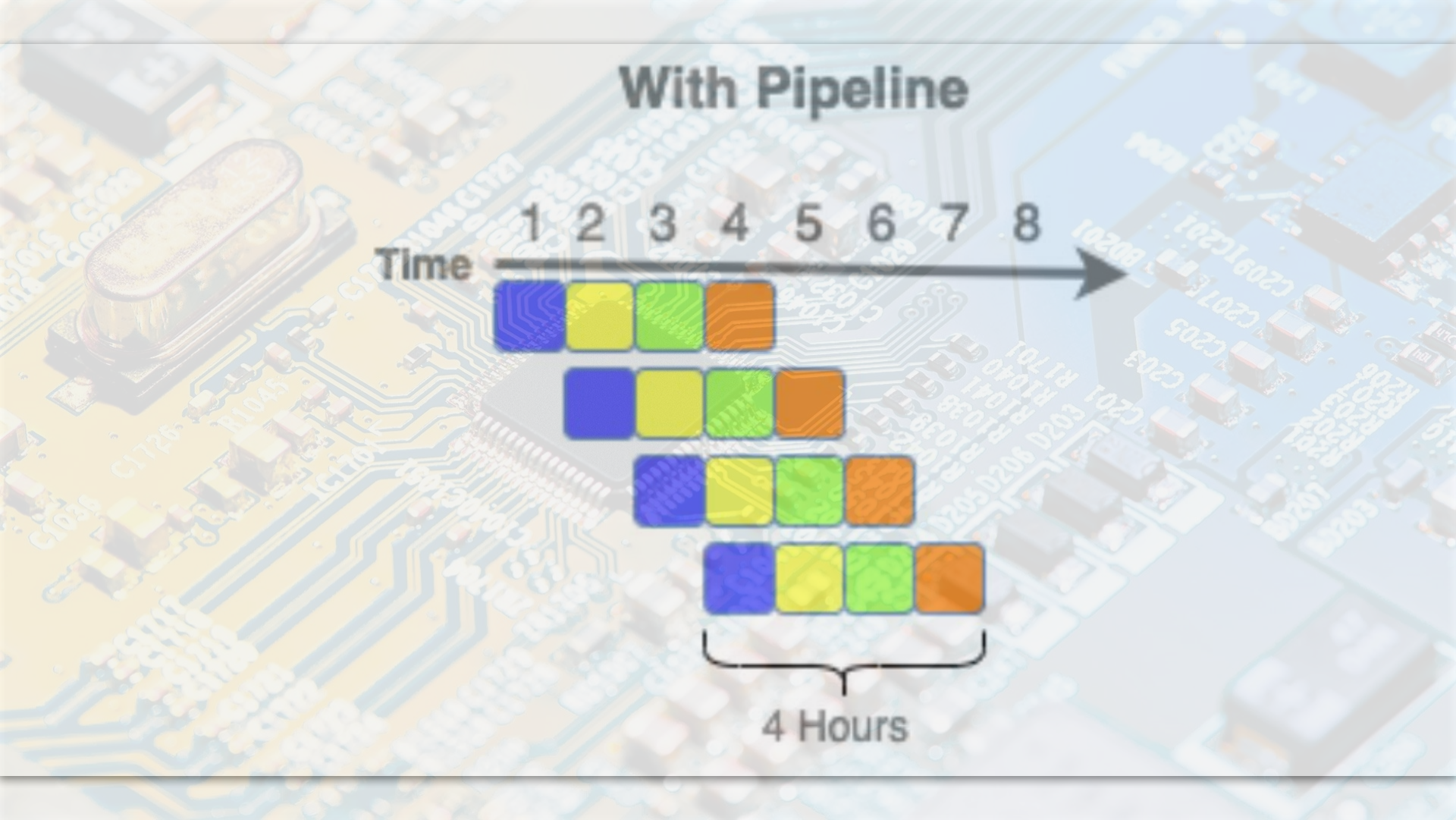
Exemplo Clássico



Exemplo Clássico




No Pipeline
Time
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
4 Hours


With Pipeline
1 2 3 4 5 6 7 8
Time
4 Hours

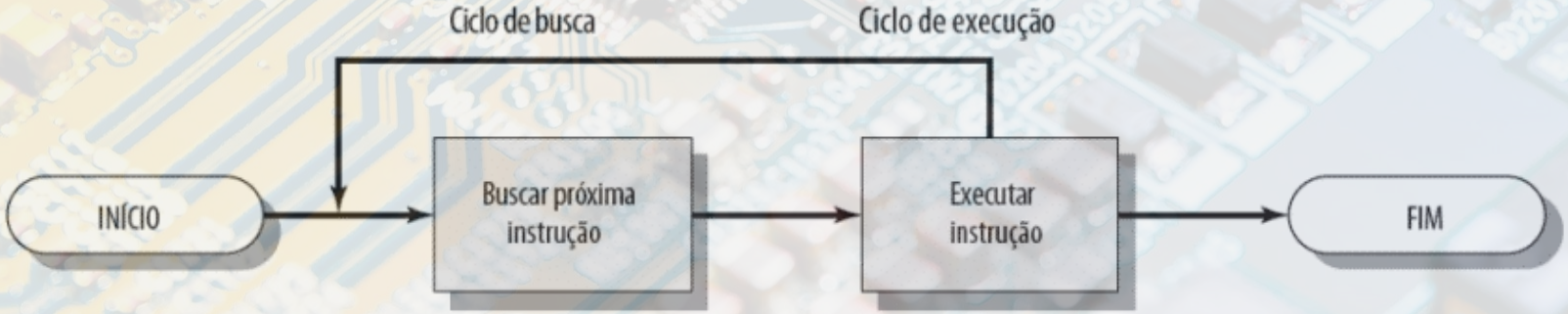
Pipeline

- No método tradicional cada instrução (“roupa”) espera a outra terminar para ser executada
- Com uso de pipelines temos um trabalho simultâneo então ganhamos tempo



Ciclo de execução das instruções

1. Busca Instrução na memória
2. Leitura dos registradores e decodificação das instruções
3. Execução de operação ou cálculo de endereço
4. Acesso a operando na memória
5. Escrita do resultado em registrador



Considerações

- Obviamente que os passos de cada instruções não tem tempos iguais, mas de modo geral ainda temos ganhos com o uso de pipelines
- O Pipeline não melhora o tempo de execução da tarefa individual, o ganho só é percebido com o conjunto de tarefas.

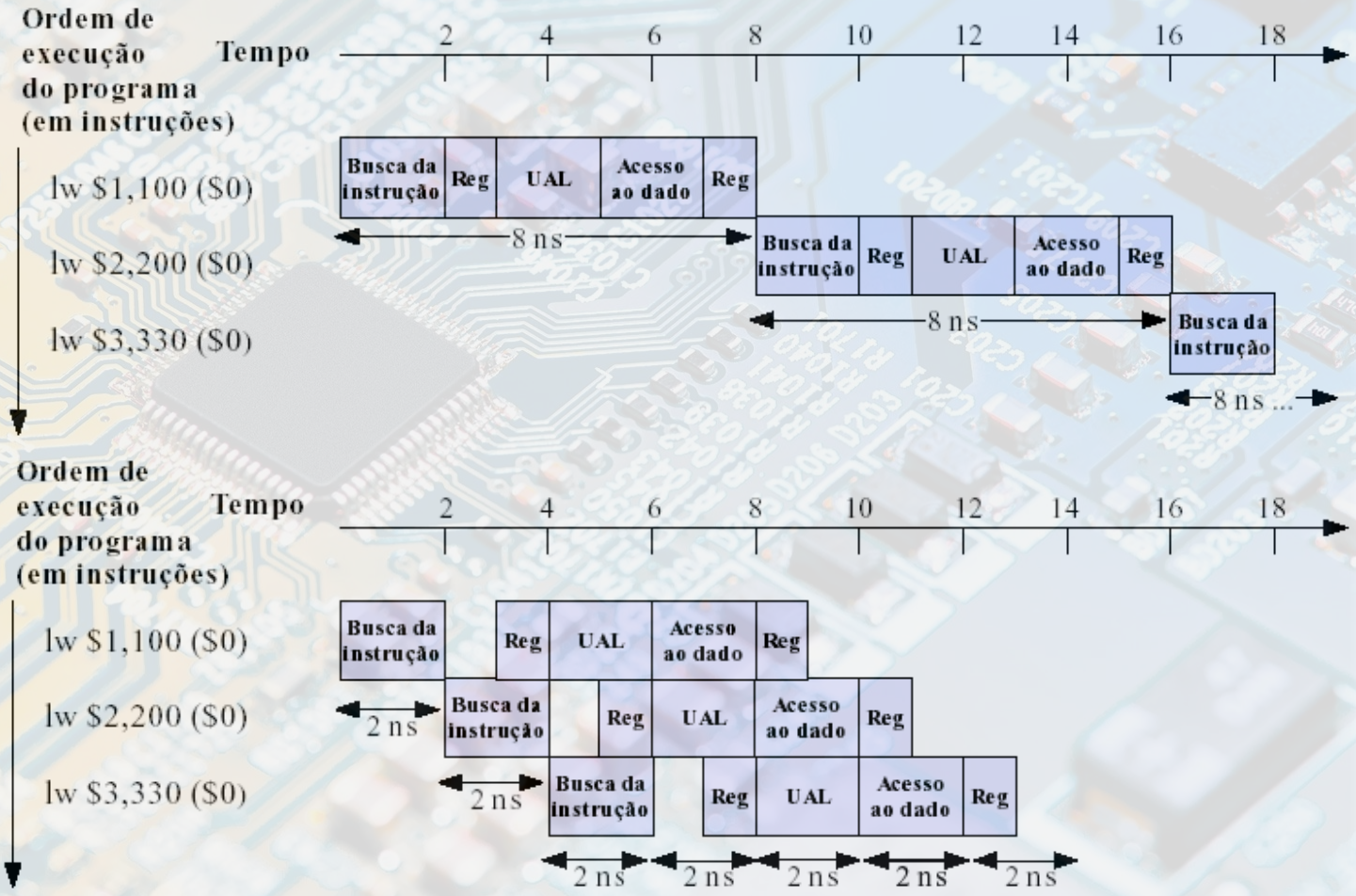
Problemas

- As instruções podem ter tempos diferentes de execução
  - Procurar dividir as instruções de forma a terem tempos mais uniformes
- Dois processos paralelos podem requerer o mesmo recurso
  - O primeiro na execução tem prioridade
- A instrução depender da antecessora

Exemplo

Classe de instrução	Busca de instrução	Leitura registro	Operação da ULA	Acesso a dado	Escrita registro	Tempo Total
Load word (lw)	2 ns	1 ns	2 ns	2 ns	1 ns	8 ns
Store word (sw)	2 ns	1 ns	2 ns	2 ns		7 ns
R-format (add,sub,and,or,slt)	2 ns	1 ns	2 ns		1 ns	6 ns
Branch (beq)	2 ns	1 ns	2 ns			5 ns

Exemplo


Ordem de execução do programa (em instruções)
Tempo
2
4
6
8
10
12
14
16
18
lw $1,100 ($0)
lw $2,200 ($0)
lw $3,330 ($0)
Busca da instrução
Reg
UAL
Acesso ao dado
Reg
8 ns
8 ns ...
2 ns

Conflitos

- Ocorrem quando temos um problema no paralelismo das instruções
  - Estruturais
  - Controle
  - Dados

Conflitos Estruturais

- Em analogia ao modelo da lavanderia seria se tivéssemos uma máquina com a função de lavar e secar.
- Acessos concorrentes a memória
  - Solução: Uso de memórias com acessos independentes
- Leitura de instrução e leitura/escrita de dados simultâneos à memória
  - Solução: Uso de cache de dados e instruções separados
- Acesso simultâneo ao banco de registradores
  - Solução: Uso de banco de registradores separados

Conflitos de Dados

- Voltando ao exemplo da lavanderia, se antes de ser guardadas as peças precisassem formar kits ou pares com resultados de outras lavagens
- Ou seja um processo precisa do resultado de outro.
  1. add $1, $2, $3
  2. sub $4, $1, $6
- Não tem uma solução fácil a estratégia geral é resolver a dependência antes de seguir para o próximo passo, ou posso seguir com resultados temporários mas isso pode causar erros.

Conflitos de Controle

- Imaginem que precisássemos do resultado da primeira amostra da lavanderia para direcionar os parâmetros das próximas lavagens.
- A estratégia é esperar o primeiro resultado ou uma amostra deles para iniciar o pipeline definitivo
- Precisaríamos de um momento de PARADA

Objetivos e Motivação

Estudar como podemos aumentar o desempenho do processamento usando técnicas de paralelismo



Paralelismo

- Aumentar o desempenho dos processadores simplesmente incrementando o clock é inviável
- Para melhorar o desempenho é executando duas ou mais operações ao mesmo tempo



Pipelines

- Somente pipelines também não resolvem pois é uma estratégia para aproveitar melhor recursos ociosos



Paralelismo

- Executar mais de um processamento.
- Se apresenta de duas maneiras
  - No nível das **instruções** - dentro das instruções individuais
  - No nível do **processador** - vários processadores trabalham juntos para resolver o mesmo problema.



Condições para Paralelismo

- Três condições essenciais para que o paralelismo possa ser implementado:
  - Disponibilidade de modelos de execução, evoluídos e adaptados ao paralelismo
  - Disponibilidade de mecanismos de comunicação entre processadores e/ou processos seguros e eficientes
  - Disponibilidade de ambientes de programação paralela evoluídos

Dependências

- Para que duas tarefas possam ser executadas em paralelo, elas devem ser independentes, a execução de uma das tarefas não influencie a outra.
  - Dependência de Dados – quando uma instrução necessita de informações geradas por outra
  - Dependências de Controle – a ordem de execução importa
  - Dependências de Recursos – existem recursos que não podem ser duplicados

Granularidade E Latência

- Granularidade é o tamanho das tarefas que o programa tem que executar
- Latência é o tempo de comunicação entre dois subsistemas
  - *Latência de memória* - tempo de acessar um endereço na memória.
  - *Latência de sincronização* - tempo para que dois processos sejam sincronizados.

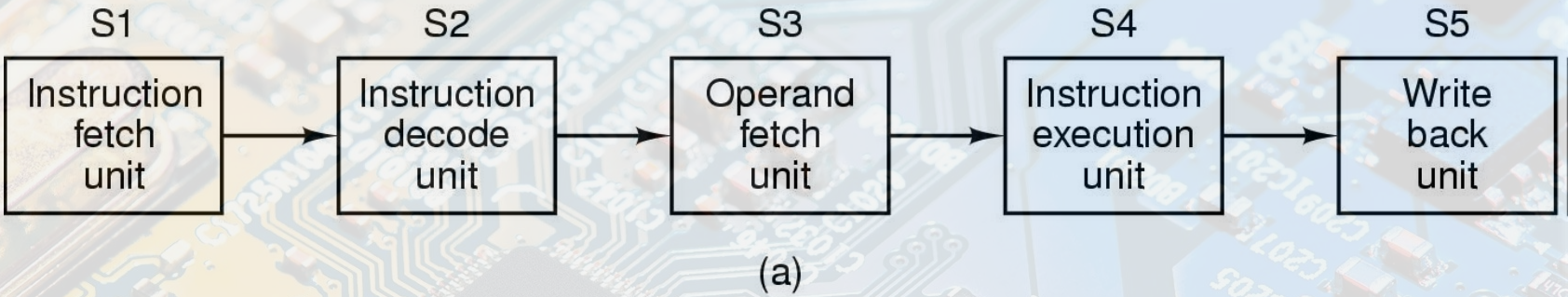
Deve haver um equilíbrio entre essas duas características, a tendência é aumentar a granularidade para ganhar paralelismo mas ganhamos latência na comunicação dos processos



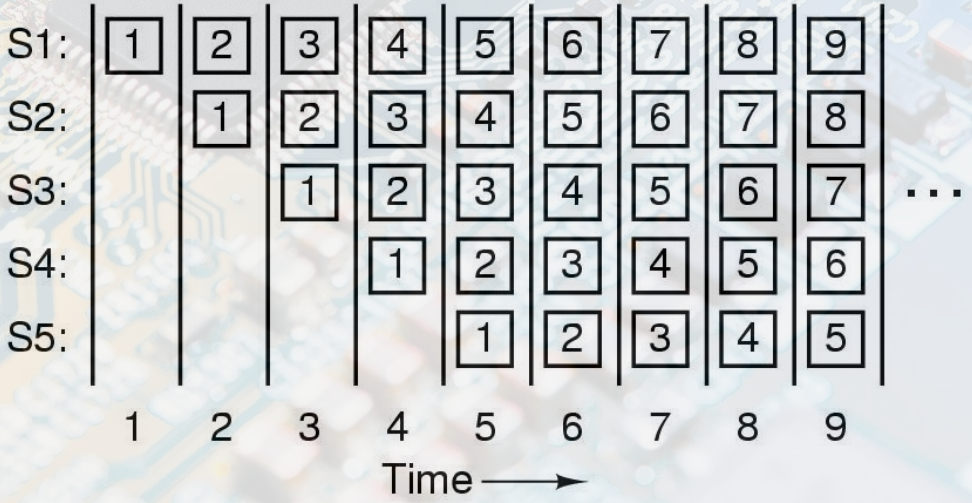
Pipelines

- O *pipeline* é uma técnica de implementação em que múltiplas instruções são sobrepostas na execução.
- Paralelismo no nível da instrução
- Principais blocos de um *pipeline*
  - Busca de instrução (S1)
  - Decodificação da instrução (S2)
  - Busca de Operando (S3)
  - Execução da instrução (S4)
  - Write back ou Unidade de Escrita (S5)

Pipelines



(a)



(b)

Paralelismo do Processador

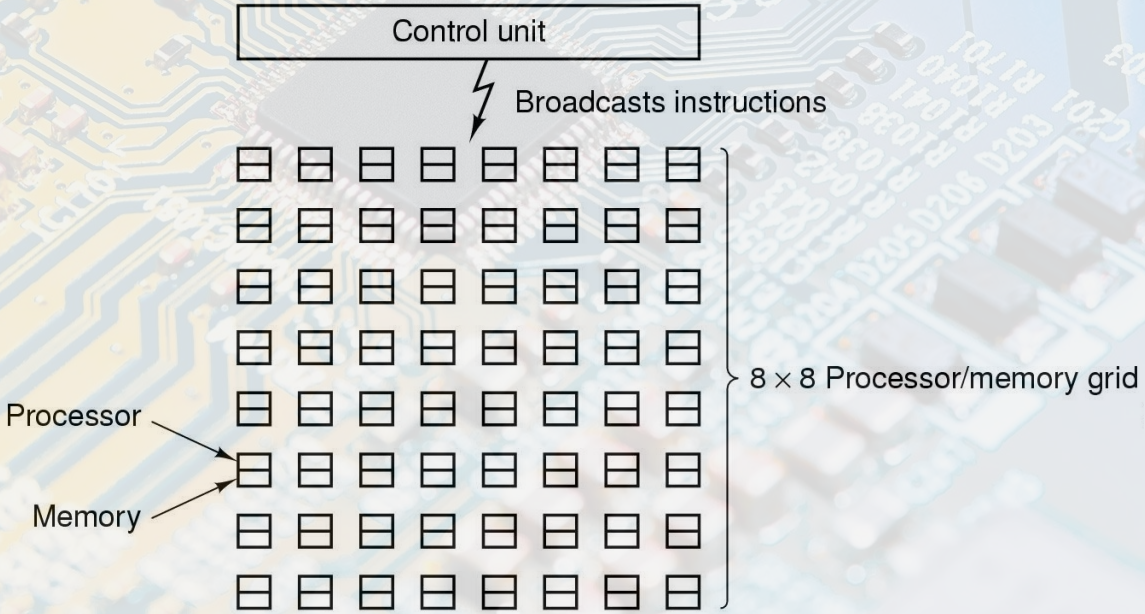
- O paralelismo ao nível das instruções dá ganhos de 5 a 10 vezes na velocidade as
- Para conseguir uma performance ainda maior, é necessário projetar computadores com mais de um processador.
- Esse tipo de organização de computadores pode ser subdividido em Computadores Matriciais, Multiprocessadores e Multicomputadores.

Computadores Matriciais

- Utilizados em cálculos científicos
- Trabalham com matrizes
- Diversos conjuntos de dados ao mesmo tempo.
- Duas metodologias que são usadas de maneira eficiente na resolução de problemas científicos:
  - Processamento Matricial
  - Processamento Vetorial.
- Um processador matricial é composto de vários processadores idênticos, que executam a mesma seqüência de instruções, mas em conjuntos de dados diferentes

Processamento Matricial

- Executavam a mesma operação em conjunto de dados diferentes



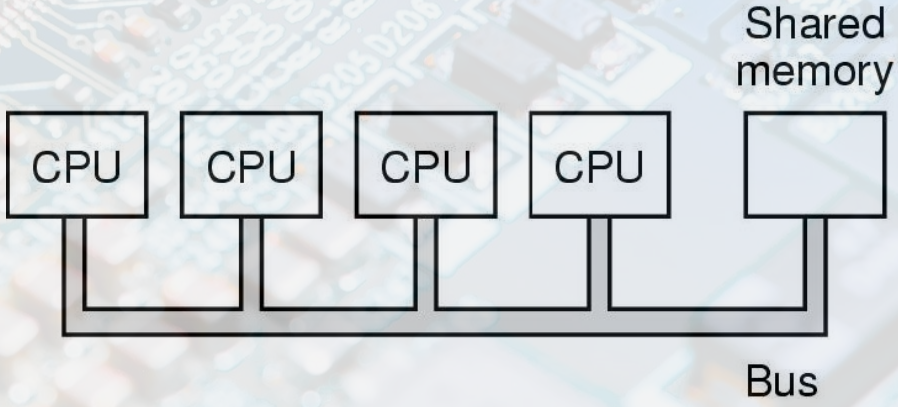
Multiprocessadores

- No computador matricial, os processadores não são independentes pois compartilham a mesma unidade de controle.
- Em um sistema multi processador, cada processador é independente.



Multiprocessadores

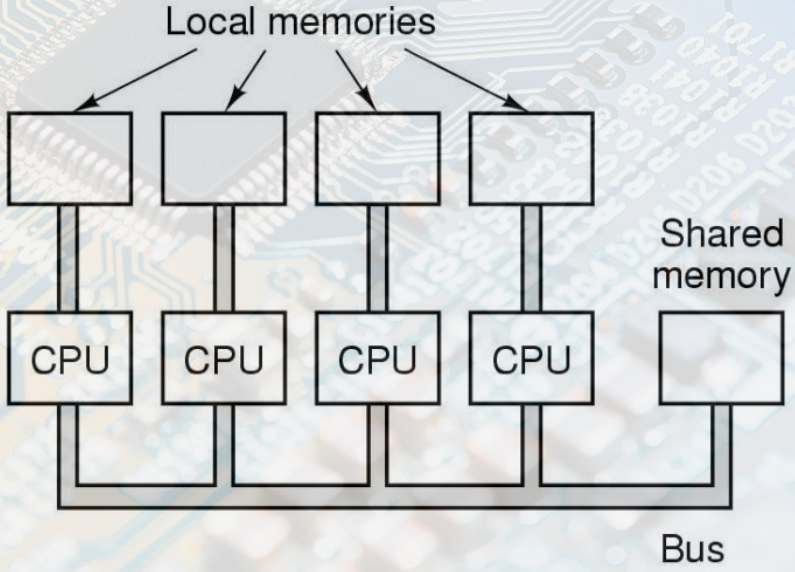
- Apenas um barramento,
- Vários processadores e uma memória compartilhada.
- Problema: conflito na transmissão e no armazenamento dos dados.



(a)

Multiprocessadores

- Para resolver o problema foi desenvolvida uma arquitetura na qual cada processador tem uma memória local, além da memória que é compartilhada por todos os processadores.
- Para acessar essa memória local, não é necessário utilizar o barramento, diminuindo assim o tráfego sobre ele, e também a possibilidade de conflitos.



(b)

Multicomputadores

- Apesar de os sistemas multiprocessados terem um bom desempenho, possuem limitações no acesso a memória compartilhada
- Foram desenvolvidos sistemas chamados multicomputadores compostos de vários computadores interconectados e sem memória comum.
  - vários computadores completos e autônomos
  - trabalham em conjunto
  - comunicam através de uma rede



Arquiteturas Vetoriais

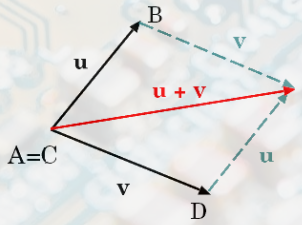
- Resolver problemas com cálculos vetoriais, são necessários instruções vetoriais (operações) e vetores (operandos).
- Em computação, vetor é um conjunto de dados, todos do mesmo tipo, armazenados em memória de forma ordenada.
- Um processador vetorial possui recursos apropriados a execução de instruções vetoriais

Computador Vetorial

- Os computadores para lidar com cálculos científicos complexos

  Ex.: aerodinâmica, sismologia, meteorologia

- Um computador vetorial é caracterizado por poder realizar operações em vetores completos através de uma simples instrução.



Computador Vetorial

- Esses problemas necessitam de alta precisão numérica e programas que efetuem repetidamente operações matemáticas de ponto flutuante em grande vetores de números.
- Então foram desenvolvidos os supercomputadores, máquinas capazes de efetuar centenas de milhões de operações de ponto flutuante por segundo.

Arquitetura Superescalar

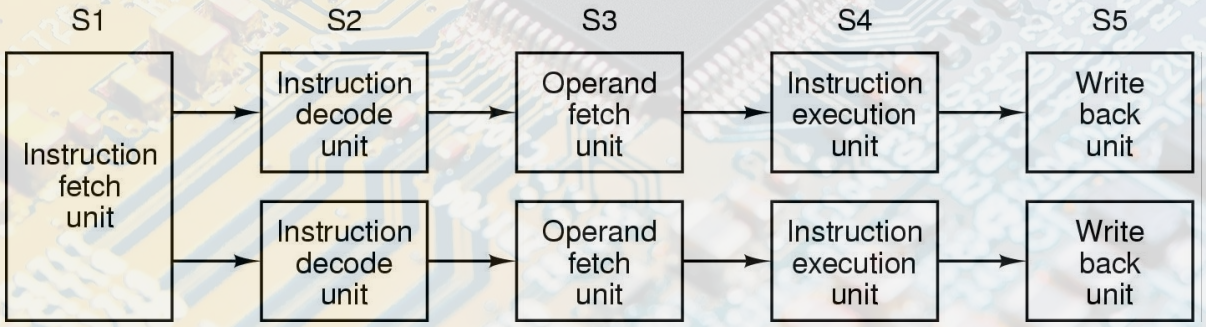
- Assim como nos Pipelines é uma forma de paralelismo no nível das instruções
- Objetivo aumentar o desempenho de um processador executando mais de uma **instrução** ao mesmo tempo.
- Múltiplas unidades funcionais dentro de um único processador dando-lhe a capacidade de trabalhar em paralelo.
- Ex.: (A+B)+(C+D)
- (A+B)+C não podem ser executadas em paralelo

Detecção de Paralelismo

- Deve existir um algoritmo de detecção de paralelismo pois nem todas as instruções podem ser executadas em paralelo.
- A alternativa mais barata e eficiente é que a detecção seja feita por **software**.
- Porém, existe a desvantagem de que todo o software executado neste computador deva ser específico.
- A detecção de paralelismo por **hardware** deve ser implantada dentro do processador, ou seja, o processador deve detectar as instruções que podem ser executadas em paralelo em tempo real.

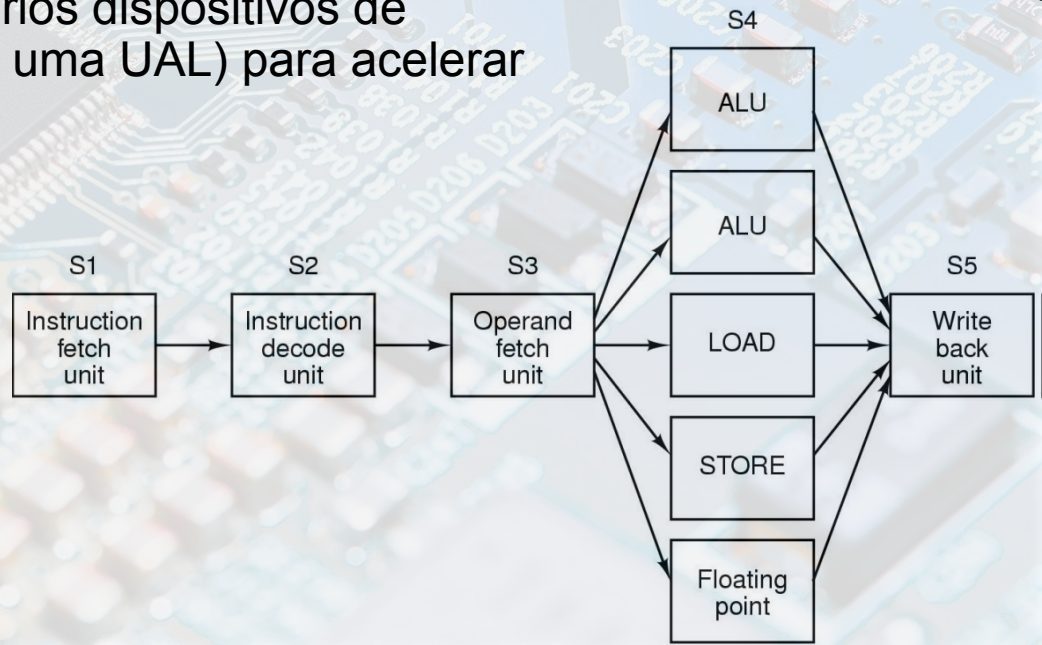
Arquitetura Superescalar

- Processador com dois pipeline,
- uma única unidade de busca de instrução busca pares de instruções juntas e coloca cada uma delas em seu próprio pipeline, permitindo, portanto, a execução de instruções em paralelo.
- Máquinas de alta performance usam outra metodologia, a arquitetura Superescalar.



Arquitetura Superescalar

- A idéia básica é ter um único pipeline, com diversas unidades funcionais
- O estágio 3 pode distribuir instruções a uma velocidade consideravelmente mais alta do que o estagio 4 pode executá-las. Este estágio usa vários dispositivos de hardware (inclusive mais do que uma UAL) para acelerar o processamento neste estágio.



Paralelismo

- Aumentar o desempenho dos processadores simplesmente incrementando o clock é inviável
- Para melhorar o desempenho é executando duas ou mais operações ao mesmo tempo



Paralelismo

- Executar mais de um processamento.
- Aumentar a escalabilidade.
- Divisão de tarefas
- Se apresenta de duas maneiras
  - No nível das **instruções** - dentro das instruções individuais.
    - Ex.: Pipelines, vetoriais e superescalares
  - No nível do **processador** - vários processadores trabalham juntos para resolver o mesmo problema.
    - Ex.: Matriciais, multiprocessadores, clusters, SMP



Condições para Paralelismo

- Três condições essenciais para que o paralelismo possa ser implementado:
  - Disponibilidade de modelos de execução, evoluídos e adaptados ao paralelismo
  - Disponibilidade de mecanismos de comunicação entre processadores e/ou processos seguros e eficientes
  - Disponibilidade de ambientes de programação paralela evoluídos

Dependências

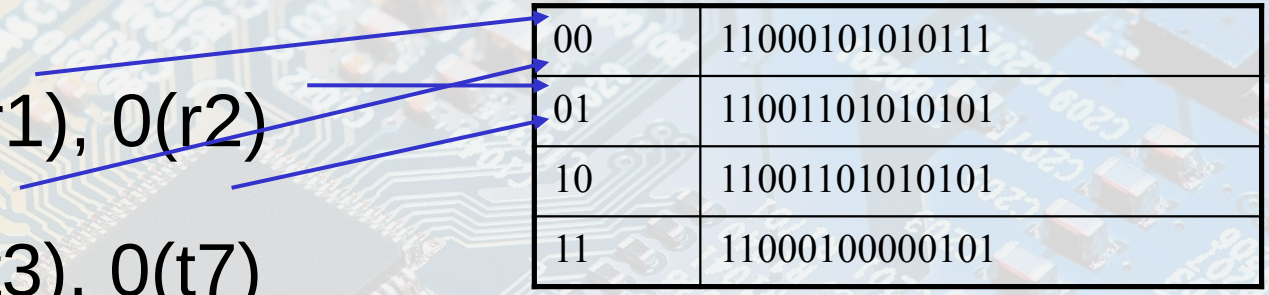
- Para determinar quanto paralelismo tem em um pprograma precisamos determinar qual o nível de dependência das instruções
- Para que duas tarefas possam ser executadas em paralelo, elas devem ser independentes, a execução de uma das tarefas não influencie a outra.
  - Dependência de Dados – quando uma instrução necessita de informações geradas por outra
  - Dependências de Controle – a ordem de execução importa
  - Dependências de Recursos – existem recursos que não podem ser duplicados

Dependência de dados

- LD f0, 0(r1)
- ADD f4, f0, f2
- SD f4, 0(r1)
- SUB r1,r1,8
- BNE r1,r2, loop

Dependências

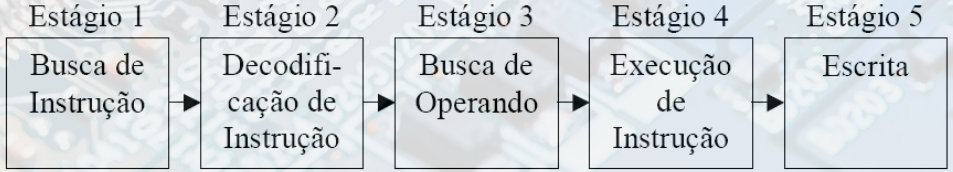
- ADD f4, 0(r1), 0(r2)
- SUB f5, 0(t3), 0(t7)



00	11000101010111
01	11001101010101
10	11001101010101
11	11000100000101

Pipelines

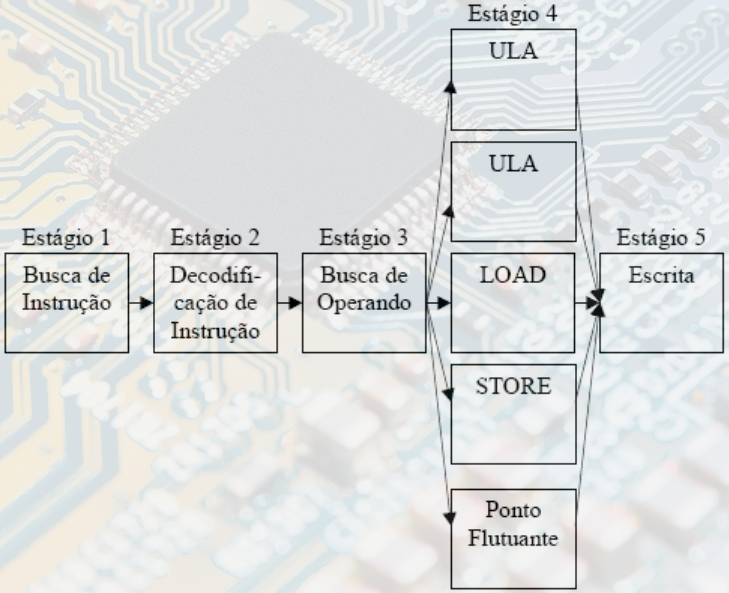
- 1985 – Idéia da linha de montagem
- Gera redução do tempo médio de execução
- Funciona melhor com processadores RISC
- Somente pipelines também não resolvem pois é uma estratégia para aproveitar melhor recursos ociosos



Clock→ Estágio	1	2	3	4	5	6	7
E1	1	2	3	4	5	6	7
E2		1	2	3	4	5	6
E3			1	2	3	4	5
E4				1	2	3	4
E5					1	2	3

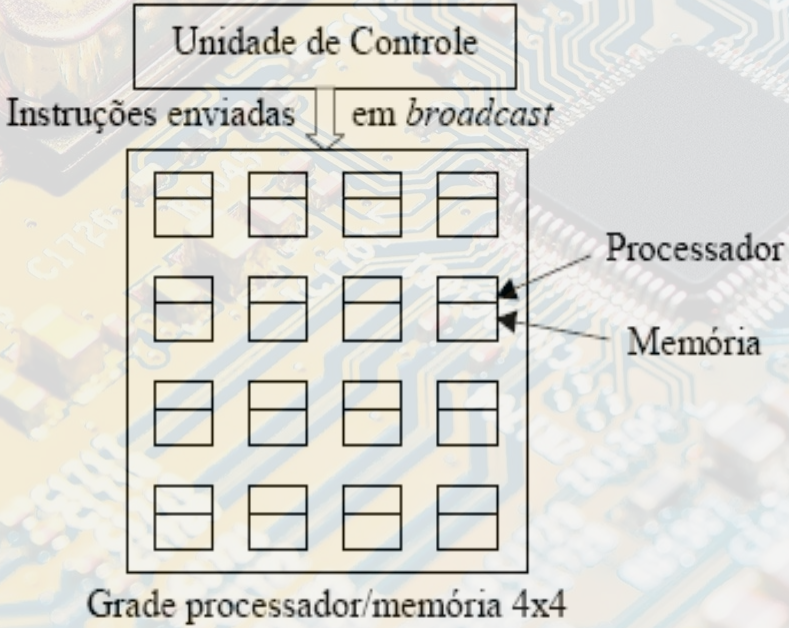
Superescalar

- A fase de execução das instruções [e mais demorada
- Pipeline único dotado de vários estágios de execução paralelo

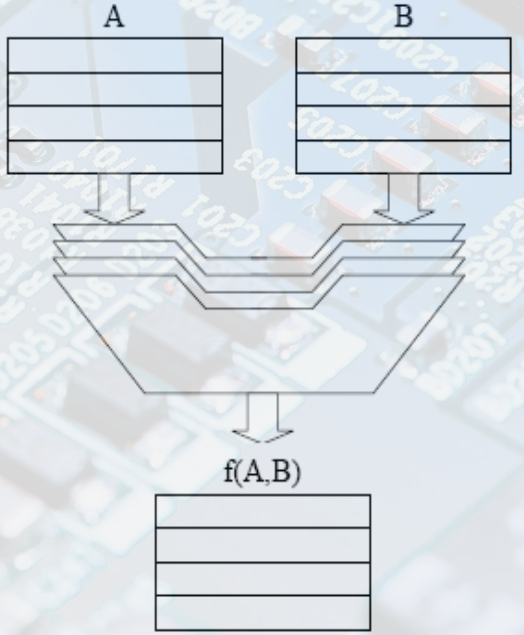


Matriciais e Vetoriais

Matricial



ULA Vetorial



Vetorial

- A maneira mais simples de aumentar o paralelismo é através de loops

```
For(i=1; i<=100; i++)
x[i] = y[i] + x[i];
```

Classificação de Flynn

- Segundo Flynn o processo computacional deve ser visto como uma relação entre fluxos de instruções e fluxos de dados. Um fluxo de instruções equivale a uma seqüência de instruções executadas sobre um fluxo de dados .
- **SISD** (Single Instruction Stream, Single Data Stream): computadores seqüenciais
- **SIMD** (Single Instruction Stream, Multiple Data Streams): computadores vetoriais e matriciais
- **MISD** (Multiple Instruction Streams, Single Data Stream): não existem
- **MIMD** (Multiple Instruction Streams, Multiple Data Streams): arquiteturas com múltiplos processadores independentes

SISD

- Um único fluxo de instruções opera sobre um único fluxo de dados
- Corresponde ao processamento seqüencial característico da arquitetura de Von Neumann.
- PC e Estações de trabalho.



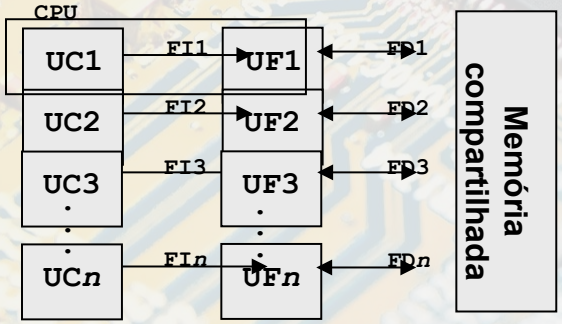
SIMD

- Um único fluxo de instruções opera sobre múltiplos fluxos de dados.
- Executa simultaneamente e a mesma instrução sobre conjuntos de dados distintos.

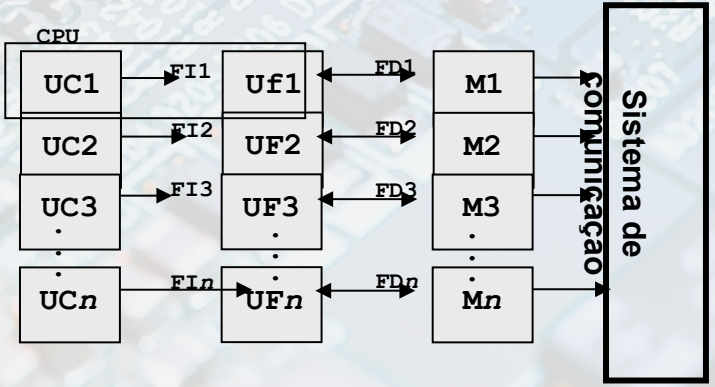
MIMD

- Múltiplos fluxos de instruções e múltiplos fluxos de dados
- Base da maioria das arquiteturas paralelas da atualidade
- Pode utilizar memória compartilhada ou distribuída

MIMD compartilhada

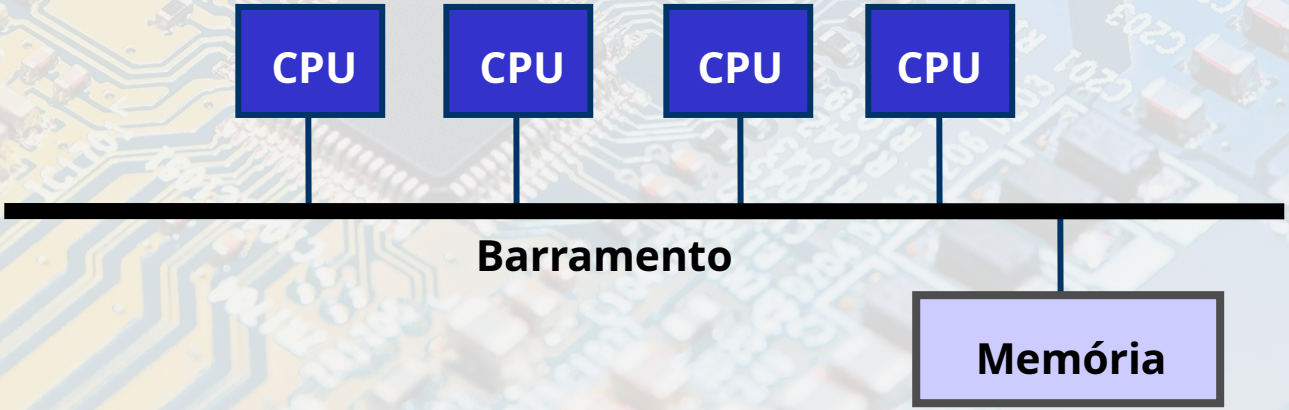


MIMD distribuída



SMP

- MIMD
- Multiprocessador simétrico com memória compartilhada

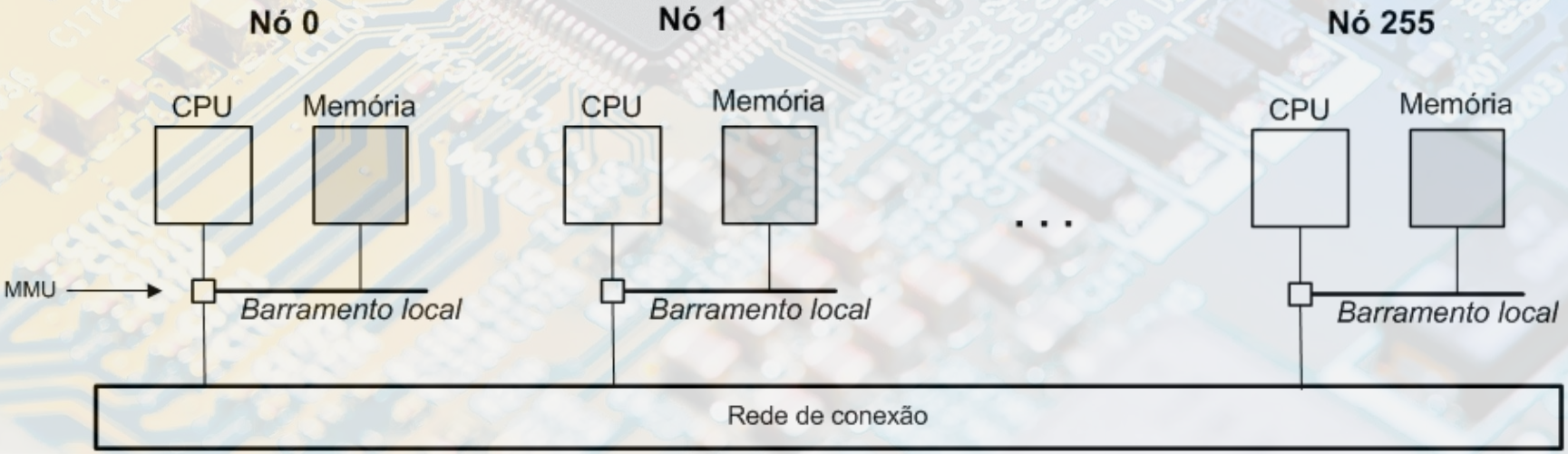


SMP

- Dois ou mais processadores compartilhando a mesma memória e dispositivos I/O
- Único sistema operacional que torna transparente ao usuário a existência de vários processadores.
  - Ex.: *nix e MS-Windows
- Todos os processadores podem desempenhar as mesmas funções
- O tempo de acesso a memória é o mesmo - é uma arquitetura UMA - *Uniform Memory Access*.
- Estruturas bastante similar a de um único processador
- Arbitragem do barramento
- Funciona bem até 3 processadores (o barramento compartilhado se torna um gargalo)

NUMA

- MIMD
- Semelhante ao SMP
- Acesso não uniforme a memória
- Todos os processadores podem acessar todas as posições de memória mas o tempo pode variar



Cluster

- MIMD
- Conjunto de computadores com memória distribuída
- Trabalham em conjunto e mantém a ilusão de uma única máquina
- Conectados com uma interface de alto desempenho
- Escalabilidade, disponibilidade, custo/benefício

Cluster



Objetivos e Motivação

- Conhecer os princípios de funcionamento de sistemas paralelos em cluster



Cluster

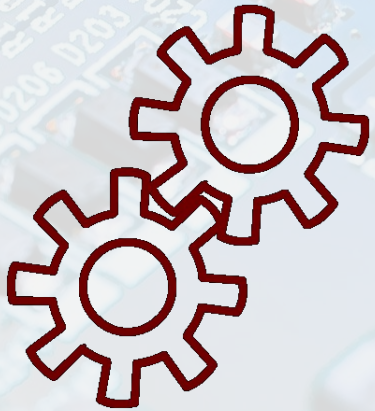
- MIMD
- Conjunto de computadores autônomos
- Trabalham em conjunto e mantém a ilusão de uma única máquina
- Conectados com uma interface de alto desempenho (rede)
- Escalabilidade, disponibilidade, custo/benefício

Cluster

- Cada computador chamado nó
- Facilidade se tiverem o mesmo SO
- Configuração dos nós: Os nós podem ser PCs ou mesmo SMPs
- Escalabilidade absoluta - clusters muito grandes com capacidade várias vezes maior que a capacidade da máquina individual
- Escalabilidade incremental
- Disponibilidade - uma falha em um nó não implica necessariamente na perda total de serviço

Porque usar?

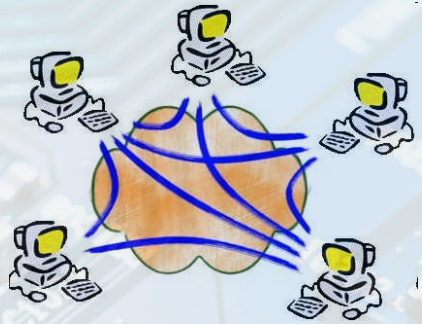
- Diminuir a **ociosidade** – se você tem dois computadores é possível que em algum instante um deles esteja sem fazer nada
- Espalhar a carga em todos os computadores



Cluster

- **Melhor Relação custo/desempenho** - facilidade de construir um cluster a partir PCs comercialmente disponíveis, é possível obter um cluster com poder de computação igual ou maior que uma única máquina paralela de grande porte, por um custo muito menor.
- Aplicação para grandes processamentos
- Pode-se desenvolver **aplicações específicas** para resolver problemas de maneira concorrente que cuidam de todos os aspectos de processamento, balanceamento de carga, comunicação e sincronização.
- Pode ser utilizado como um recurso único, de maneira **transparente** para os usuários. Para isto, é necessário a utilização de escalonadores que cuidam da distribuição de carga entre os servidores.

Escolha da rede



- Capital disponível;
- Do tipo de aplicação que será executada no cluster.
  - Aplicações I/O-bound necessitam de redes de alto desempenho
  - Aplicações CPU-bound podem utilizar redes mais lentas

Tipos

- Tolerante a falhas
- Balanceamento de carga
- Alto Desempenho



Tolerante a falha

- Dois ou mais computadores conectados com um software de monitoramento – heart-beat
- Se uma falhar a outra assume

Balanceamento de carga

- É um tipo de tolerante a falha na qual as duas são usadas para distribuição de carga



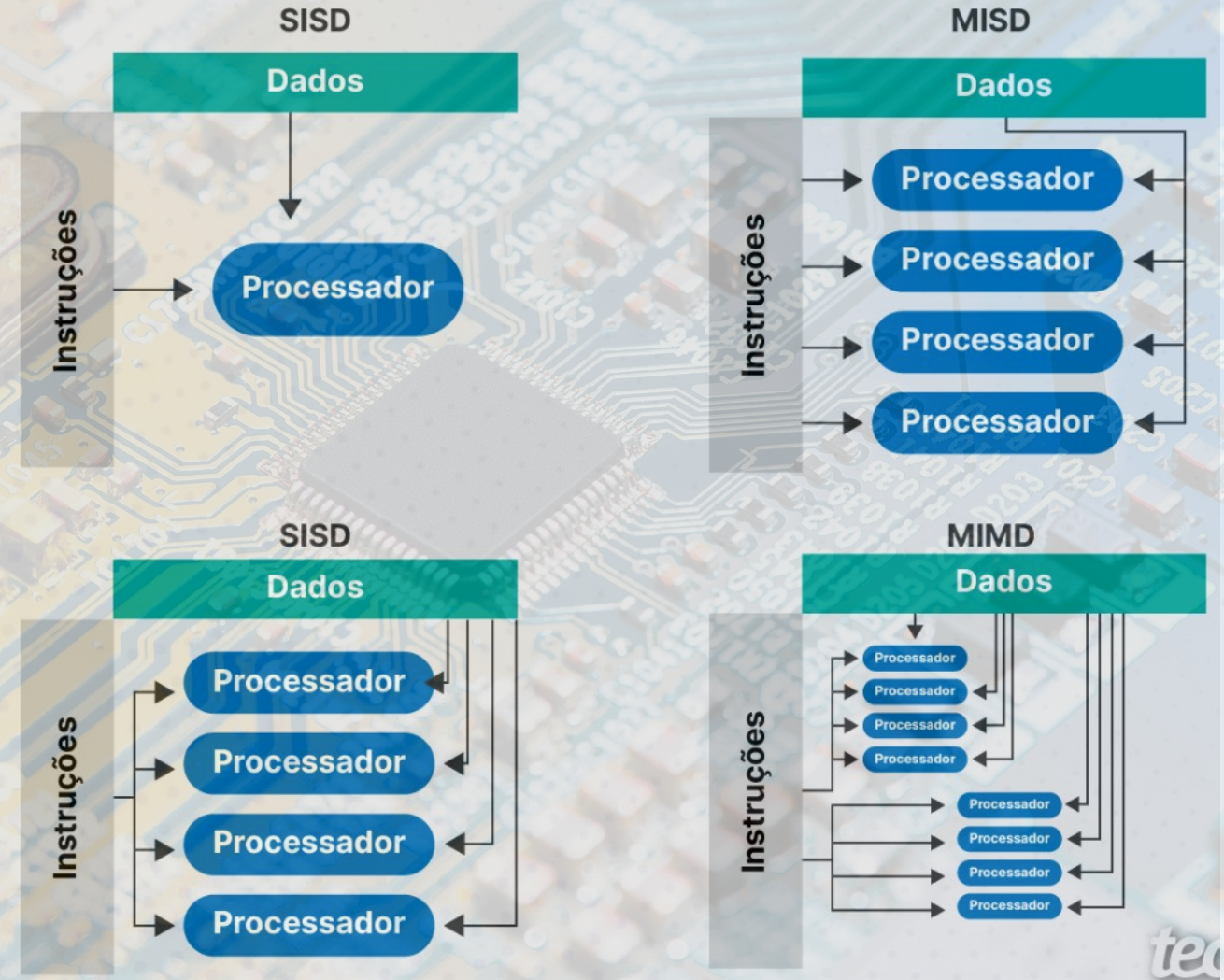
Alto desempenho

- Criar a ilusão de uma máquina de alto poder computacional
- Divisão de tarefas no nível de subprocessos e threads



Desafios

- Segurança
- Aplicabilidade
  - Data mining, servidores web
  - Química quântica, oceanografia, previsão do tempo
- Gerenciamento


SISD
Dados
Instruções
Processador
MISD
Dados
Instruções
Processador
Processador
Processador
Processador
SISD
Dados
Instruções
Processador
Processador
Processador
Processador
MIMD
Dados
Instruções
Processador
Processador
Processador
Processador
Processador
Processador
Processador
Processador

Eclesiastes 12:1 – "Lembra-te do teu Criador nos dias da tua mocidade."

A juventude é o melhor tempo para firmar valores que guiarão sua vida.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO