Pipelining Avançado

Prof. Rômulo Calado Pantaleão Camara

Carga Horária: 2h/60h

Introdução

- ✓ A técnica de *pipelining* explora o paralelismo entre as instruções → **Paralelismo em Nível de Instrução (ILP)**.

- ✓ Métodos para aumentar a quantidade de ILP:

  - Aumentar a "profundidade" do *pipeline* - Sobrepor mais instruções → **Superpipeline**.

  - Replicar os componentes internos do computador – Iniciar várias instruções em cada estágio → **Despacho múltiplo**.

Superpipeline

- ✓ **Exemplo:**
  - Considere que o ciclo da lavadora fosse maior do que os outros. Poderíamos dividir a lavadora em três máquinas que lavam, enxáguam e centrifugam. Então, o *pipeline* passaria de quatro para seis estágios.
    - A quantidade de paralelismo aumentaria, pois existem mais operações sendo sobrepostas.
    - O desempenho é potencialmente maior, pois o ciclo de *clock* pode ser encurtado.

Despacho Múltiplo

- ✓ **Exemplo:**
  - Substituir a lavadora e a secadora por, três lavadoras e três secadoras.
    - Seria necessário ter mais pessoas para passar e guardar três vezes a quantidade de roupa no mesmo período.

Despacho Múltiplo

- ✓ Considere um processador de despacho múltiplo quádruplo – processador com quatro unidades por estágio do *pipeline.*
  - Quantas instruções por ciclo de *clock* esse processador pode executar?
    - R.: quatro instruções.
  - Considerando um *pipeline* de cinco estágios, qual o número máximo de instruções executadas em um determinado instante? Desenhe um esquema.
    - R.: vinte instruções.

Despacho Múltiplo

- ✓ Os microprocessadores mais potentes de hoje tentam despachar de três a oito instruções a cada ciclo de clock.

- ✓ **Dificuldades:**

  - Restrições relacionadas aos tipos de instruções que podem ser executadas simultaneamente.

  - Consequências geradas pela ocorrência de dependências.

Despacho Múltiplo

- ✓ Técnicas para implementar um processador de despacho múltiplo:
  - **Técnica de despacho múltiplo estático.**
  - **Técnica de despacho múltiplo dinâmico.**
- ✓ A principal diferença entre elas é a **divisão de trabalho** entre o compilador e o *hardware*.
  - **Divisão de trabalho** → Indica se as decisões estão sendo feitas estaticamente (durante compilação) ou dinamicamente (durante execução).

Despacho Múltiplo

- ✓ Existem duas tarefas principais que precisam ser tratadas em um *pipeline* de despacho múltiplo:
  - **Empacotar as instruções em *slots* de despacho;**
    - Como o processador determina quantas instruções e quais instruções podem ser despachadas em um determinado ciclo de *clock*?
  - **Lidar com *hazards* de dados e controle.**
    - *Forwarding, stall...*

Despacho Múltiplo

- ✓ **Empacotar as instruções em *slots* de despacho.**
  - **Processadores de despacho múltiplo estático.**
    - É tratado parcialmente pelo compilador.
  - **Processadores de despacho múltiplo dinâmico.**
    - É tratado durante a execução pelo processador, com algum auxílio prévio do compilador, que coloca as instruções em uma ordem benéfica.

Despacho Múltiplo

- ✓ **Lidar com *hazards* de dados e controle.**
  - **Processadores de despacho múltiplo estático.**
    - Alguns ou todos os problemas dos *hazards* de dados e controle são tratados estaticamente pelo **compilador**.
  - **Processadores de despacho múltiplo dinâmico.**
    - Tenta aliviar pelo menos algumas classes de *hazards* usando técnicas de **hardware** operando durante a execução.

Despacho Múltiplo Estático

- ✓ O compilador é utilizado para tratar:
  - Empacotamento das instruções;
  - Hazards;
  - Prever estaticamente desvios e escalonamento de código para reduzir ou impedir todos os hazards.
- ✓ Um conjunto de instruções é despachado junto em um ciclo de clock – **Pacote de despacho**.

Despacho Múltiplo Estático

- ✓ **Pacote de despacho** → Podemos imaginar como sendo uma única instrução que executa várias operações.
  - Essa técnica é chamada de **VLIW (*Very Long Instruction Word*) – Processadores VLIW**.
  - **Exemplos:**
    - Arquitetura Intel IA-64 → EPIC (*Explicitly Parallel Instruction Computer*).
      - Computadores Itanium e Itanium 2.

Despacho Múltiplo Dinâmico

- ✓ **Processadores superescalares.**
  - As instruções são despachadas em ordem.
  - O processador decide quantas instruções podem ser despachadas em um determinado ciclo de clock.
  - A velocidade de despacho das instruções depende do compilador – Escalonar as instruções para separar as dependências.
  - **Exemplos:**
    - Pentium Pro, Pentium II, III e IV, Power PC.

Analisando *Pipelines*

- ✓ Tipos de *pipelines* a serem analisados:
  - Superpipeline.
  - Superescalar.
  - Super-super.

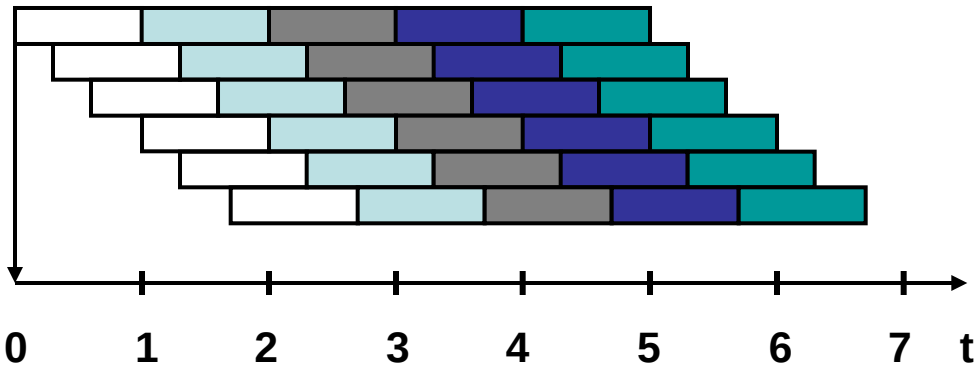
Analisando *Pipelines*

- ✓ **Superpipeline.**
  - Processador divide os estágios do pipeline de instrução em **sub-estágios**.
  - **Grau do superpipeline** = número de sub-estágio → três na figura abaixo.

– IPC (Instruções prontas por ciclo).

$$IPC = Grau\ do\ superpipeline$$

$$IPC = 3$$



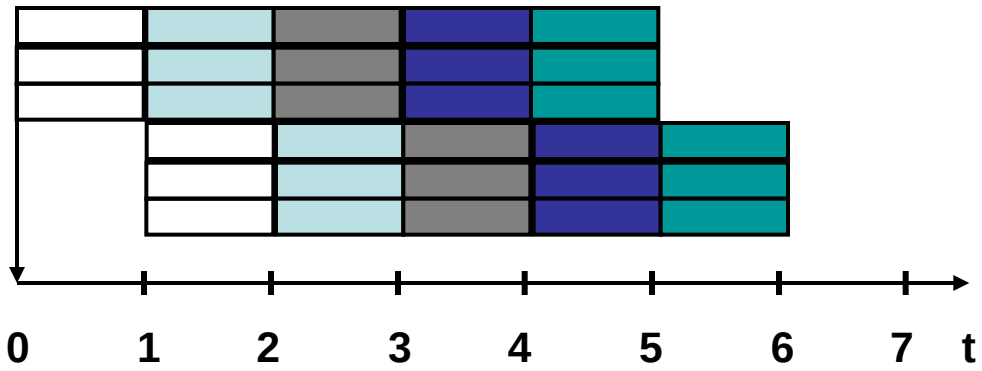
Analisando *Pipelines*

- ✓ **Superescalar.**
  - Processador possui vários *pipelines* de instrução replicados.
  - **Grau do superescalar** = número de *pipelines* replicados → três na figura abaixo.

– IPC (Instruções prontas por ciclo).

$$IPC = Grau\ do\ \text{sup}erescalar$$

$$IPC = 3$$



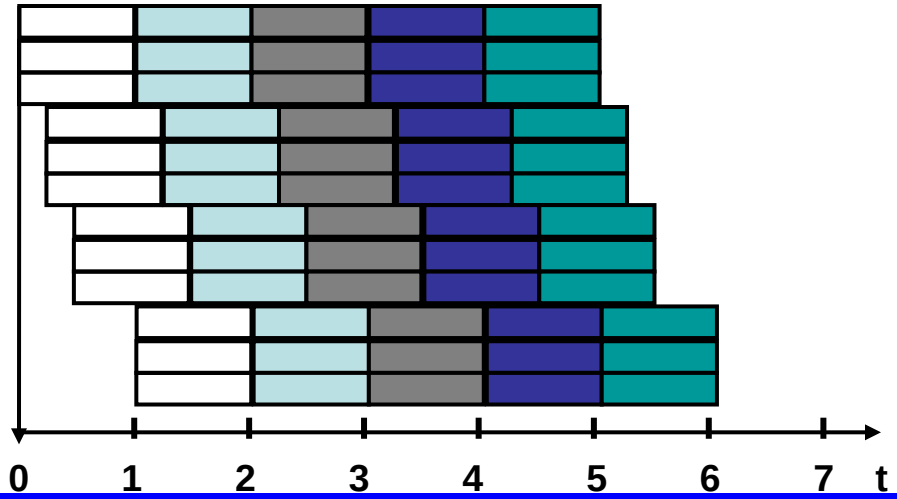
Analisando *Pipelines*

- ✓ **Super-super.**
  - Processador possui vários pipelines de instrução replicados e cada um deles é um superpipeline.
  - **Grau do Super-super** → número de superpipelines replicados * número de sub-estágios → 3*3 = 9 na figura abaixo.

– IPC (Instruções prontas por ciclo).

$$IPC = Grau\ do\ \text{super}er - \text{super}er$$

$$IPC = 9$$



Localização e Exploração de ILP

Especulação

- ✓ **Especulação** → Método importante para localizar e explorar ILP.
- ✓ **Definição** → Técnica que permite que o compilador ou o processador "adivinhem" as propriedades de uma instrução (especulada), de modo que outras instruções dependentes da instrução especulada comecem a executar.

Especulação

- ✓ **Exemplos:**
  - Especular sobre o resultado de um desvio, de modo que as instruções após o desvio pudessem ser executadas mais cedo.
  - Especular que um *store* anterior a um *load* não se refere ao mesmo endereço, o que permitiria que o *load* fosse executado antes do *store*.

Especulação

- ✓ **Problemas da especulação:**
  - Erro na especulação.
  - Necessidade de um método para verificar se a escolha foi certa, e um método para retornar os efeitos das instruções executadas de forma especulativa.
  - Métodos para retornar os efeitos das instruções aumentam a complexidade do processador.

Especulação

- ✓ **Especulação em *software* (compilador).**
  - O compilador insere instruções adicionais que verificam a precisão da especulação e oferecem uma rotina de reparo para usar quando a especulação for incorreta.

Especulação

- ✓ **Especulação em *hardware* (processador).**
  - O processador coloca os resultados especulativos em um *buffer* até saber que eles não são mais especulativos.
    - **Especulação correta** → As instruções são concluídas, permitindo que o conteúdo dos *buffers* seja escrito nos registradores ou na memória.
    - **Especulação incorreta** → O *hardware* faz um *flush* nos *buffers* e executa novamente, mas na seqüência de instruções corretas.

Exemplo de Despacho Múltiplo Estático em Processadores MIPS

Despacho Múltiplo Estático - MIPS

- ✓ Considere um processador MIPS capaz de despachar duas instruções por ciclo, sendo que uma delas pode ser uma operação da ULA ou desvio, e a outra um load (lw) ou store (sw), nesta ordem.

Tipo da instrução	Estágio do pipeline							
ALU or branch instruction	IF	ID	EX	MEM	WB			
Load or store instruction	IF	ID	EX	MEM	WB			
ALU or branch instruction		IF	ID	EX	MEM	WB		
Load or store instruction		IF	ID	EX	MEM	WB		
ALU or branch instruction			IF	ID	EX	MEM	WB	
Load or store instruction			IF	ID	EX	MEM	WB	
ALU or branch instruction				IF	ID	EX	MEM	WB
Load or store instruction				IF	ID	EX	MEM	WB

- Se uma instrução do par não puder ser usada, ela deve ser substituída por um NOP (*No OPeration*).

Despacho Múltiplo Estático - MIPS

- ✓ Como o loop abaixo seria escalonado de modo a explorar o paralelismo com eficiência?

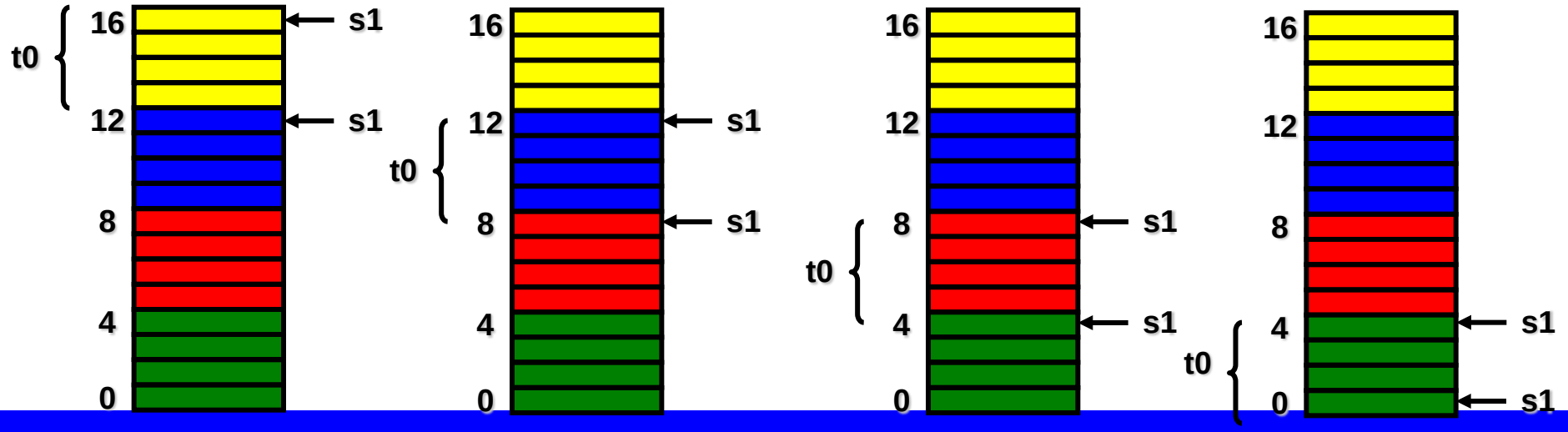
```
Loop: lw     $t0, 0($s1)        #$t0 = elemento do array
      addu   $t0, $t0, $s2      #add escalar em $s2
      sw     $t0, 0($s1)        #resultado do store
      addi   $s1, $s1, -4       #decrementa ponteiro
      bne    $s1, $zero, Loop   #desvia se $s1 != 0
```

  - As três primeiras instruções possuem dependências, assim como as duas últimas.
  - Considere que os desvios são previstos e os *hazards* são tratados pelo *hardware*.
  - **TAREFA:** reordene essas instruções de forma a evitar stalls.

Despacho Múltiplo Estático - MIPS

```
Loop:   lw    $t0, 0($s1)      #$t0 = elemento do array
        addu  $t0, $t0, $s2    #add escalar em $s2
        sw    $t0, 0($s1)      #resultado do store
        addi  $s1, $s1, -4     #decrementa ponteiro
        bne   $s1, $zero, Loop #desvia se $s1 != 0
```

- ✓ Neste exemplo, podemos considerar que o loop possui quatro iterações e que $S1 aponta para o endereço 16.

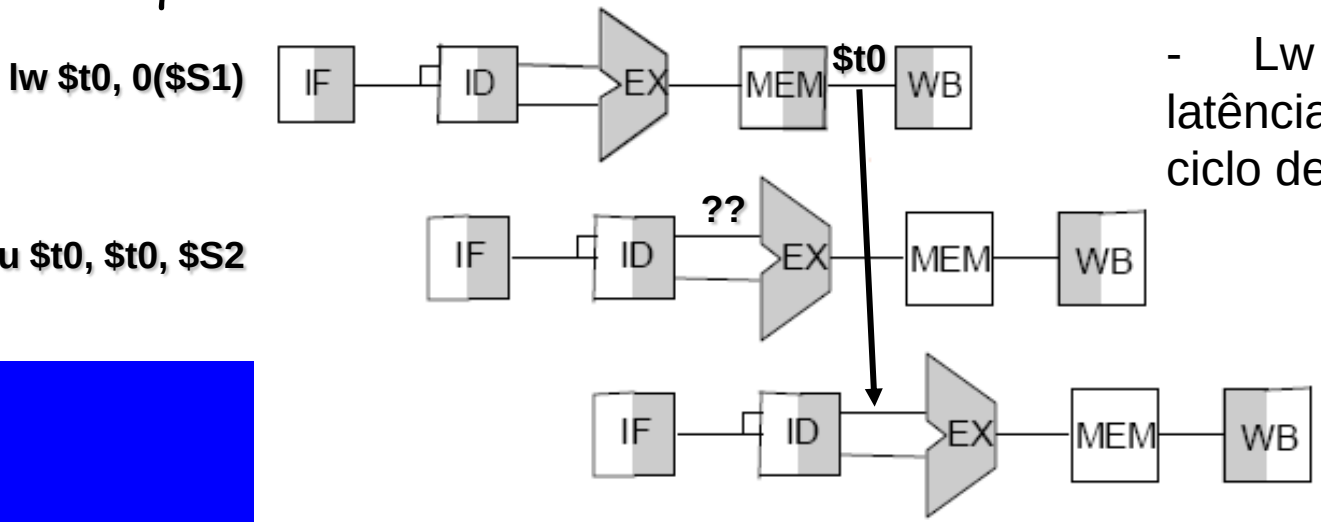


Despacho Múltiplo Estático - MIPS

- ✓ **Resultado após o escalonamento.**

	Instruções ALU ou branchs	Instr. data transfer	Ciclo de clock

- ✓ Posso executar addu no primeiro ciclo de clock? E no segundo?
  - NÃO!! Porque os dados em $t0 só estão disponível após o quarto estáaio → Seria necessário inserir stall.



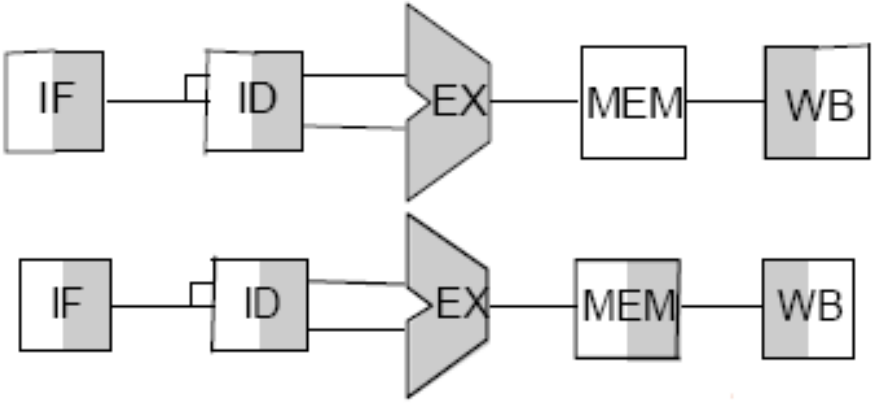
- Lw possui uma latência de uso de 1 ciclo de clock.

Despacho Múltiplo Estático - MIPS

- ✓ **Resultado após o escalonamento.**

	Instruções ALU ou branchs	Instr. data transfer	Ciclo de clock
Loop:		lw $t0, 0($s1)	1
	addi $s1,$s1,-4		2
	addu $t0,$t0,$s2		3
	bne $s1,$zero,Loop	sw $t0, 4($s1)	4

- ✓ Porque não executei addu e sw ou addi e sw no mesmo ciclo?



Despacho Múltiplo Estático - MIPS

- ✓ **Resultado após o escalonamento.**

	Instruções ALU ou branchs	Instr. data transfer	Ciclo de clock
Loop:		lw $t0, 0($s1)	1
	addi $s1,$s1,-4		2
	addu $t0,$t0,$s2		3
	bne $s1,$zero,Loop	sw $t0, 4($s1)	4

- ✓ Posso executar addi e lw no mesmo ciclo de clock (no primeiro)?
  - Apesar de não existir dependências de dados, se isto ocorresse, não teríamos instruções a serem executadas no segundo ciclo de clock – Seria um stall → **NÃO POSSO!!**

Despacho Múltiplo Estático - MIPS

- ✓ **Resultado após o escalonamento.**

	Instruções ALU ou branchs	Instr. data transfer	Ciclo de clock
Loop:		lw $t0, 0($s1)	1
	addi $s1,$s1,-4		2
	addu $t0,$t0,$s2		3
	bne $s1,$zero,Loop	sw $t0, 4($s1)	4

- ✓ IPC (Instruções Por Ciclo de Clock) = 5/4 = 1,25.
  - Melhor caso → IPC = 8/4 = 2,0.
- ✓ CPI (Ciclo de Clock Por Instrução) = 4/5 = 0,8.
  - Melhor caso → CPI = 4/8 = 0,5.

Desdobramento de Loop e Escalonamento Dinâmico em Pipeline

Desdobramento de Loop

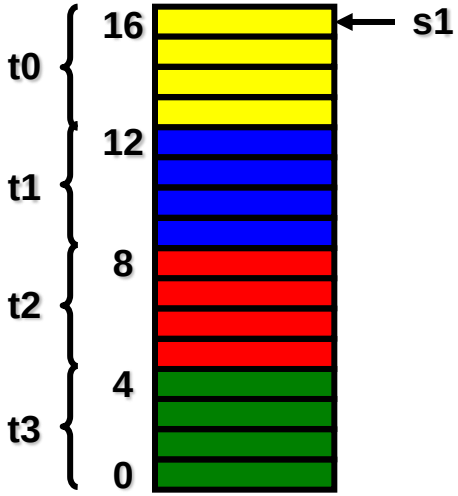
- ✓ Como melhorar o desempenho de loops?
- ✓ Técnica de compilador → **Desdobramento de loop**.
- ✓ **Funcionamento** → São feitas várias cópias do corpo do loop. Após isso, haverá mais ILP disponível pela sobreposição de instruções de diferentes iterações.
- ✓ Considerando o exemplo anterior.
  - É necessário fazer quatro cópias do corpo do loop.
  - Eliminar as instruções desnecessárias.
  - O loop terá quatro cópias de lw, addu e sw, um addi e um bne.

Desdobramento de Loop

- ✓ Desdobrando o loop.

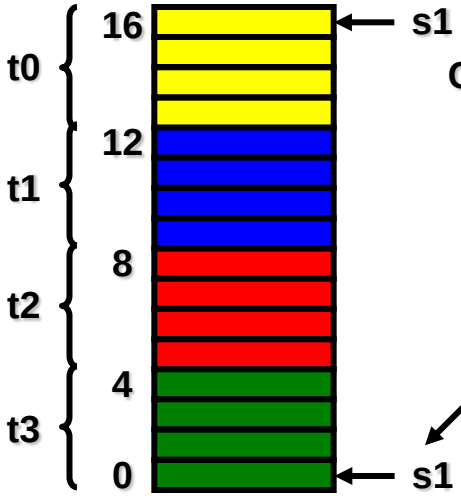
Renomeação de registradores (compilador).

```
Loop:  lw    $t0, 0($s1)     #$t0 = elemento do array
       lw    $t1, 4($s1)     #$t1 = elemento do array
       lw    $t2, 8($s1)     #$t2 = elemento do array
       lw    $t3, 12($s1)    #$t3 = elemento do array
       addu  $t0, $t0, $s2   #add escalar em $s2
       addu  $t1,  $t1, $s2  #add escalar em $s2
       addu  $t2, $t2, $s2   #add escalar em $s2
       addu  $t3, $t3, $s2   #add escalar em $s2
       sw    $t0, 0($s1)     #resultado do store
       sw    $t1, 4($s1)     #resultado do store
       sw    $t2, 8($s1)     #resultado do store
       sw    $t3, 12($s1)    #resultado do store
       addi  $s1, $s1, -16   #decrementa ponteiro
       bne   $s1, $zero, Loop  #desvia se $s1 != 0
```



Desdobramento de Loop

- ✓ **Objetivo da renomeação de registradores** → Eliminar dependências que não são dependências de dados verdadeiras (**interdependência ou dependência de nome**), mas que poderiam conduzir à hazards ou impedir que o compilador escalonasse o código de forma flexível.



Código desdobrado e escalonado – inicialmente $S1 aponta para o endereço 16.

	Instruções ALU ou branchs	Instr. data transfer	Ciclo de clock
Loop:	addi $s1,$s1,-16	lw $t0, 0($s1)	1
		lw $t1, 12($s1)	2
	addu $t0,$t0,$s2	lw $t2, 8($s1)	3
	addu $t1,$t1,$s2	lw $t3, 4($s1)	4
	addu $t2,$t2,$s2	sw $t0, 16($s1)	5
	addu $t3,$t3,$s2	sw $t1, 12($s1)	6
		sw $t2, 8($s1)	7
	bne $s1,$zero,Loop	sw $t3, 4($s1)	8

Desdobramento de Loop

- ✓ **Análise de desempenho.**
  - IPC = 14/8 = **1,75**, contra IPC = **1,25** (sem desdob.).
  - Melhor caso → IPC = 16/8 ou 8/4 = **2,0**.
  - CPI = 8/14 = **0,57**, contra CPI = **0,8** (sem desdob.).
  - Melhor caso → CPI = 8/16 ou 4/8 = **0,5**.
- ✓ O custo da melhoria de desempenho foi o uso de quatro registradores temporários ao invés de um, além do aumento no código (mais memória).

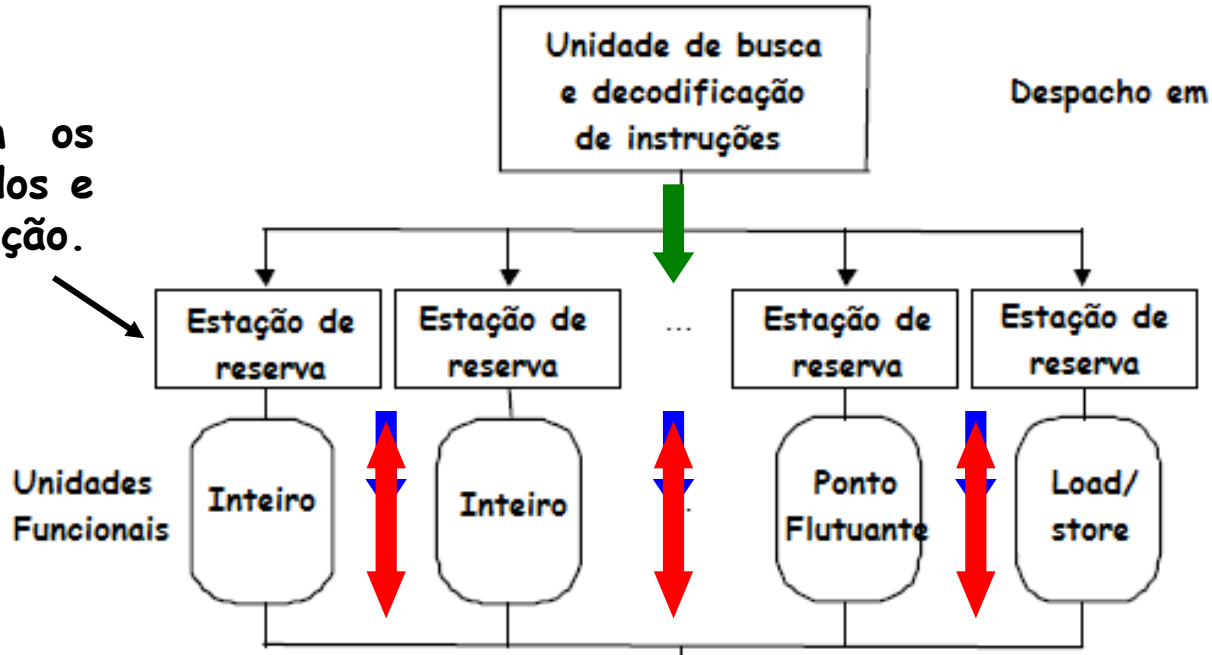
Escalonamento Dinâmico em Pipeline – Processadores Superescalares

- ✓ Processadores superescalares (despacho múltiplo dinâmico) simples:
  - As instruções são despachadas em ordem;
  - O processador decide quantas instruções podem ser despachadas em determinado ciclo de clock.
- ✓ Em alguns superescalares, a estrutura das decisões de despacho é estendida para incluir **escalonamento dinâmico em pipeline**.

Escalonamento Dinâmico em Pipeline – Processadores Superescalares

- ✓ **Escalonamento dinâmico em pipeline** → Determina dinamicamente quais instruções serão executadas em determinado ciclo de clock.
  - O hardware modifica a ordem de execução das instruções para evitar hazards e stall.
- ✓ Nesses processadores, o pipeline é dividido em três unidades principais:
  - Uma unidade de busca e despacho de instruções;
  - Várias unidades funcionais;
  - Uma unidade commit.

Escalonamento Dinâmico em Pipeline – Processadores Superescalares



As três unidade principais de um pipeline escalonado dinamicamente.

Escalonamento Dinâmico em Pipeline – Processadores Superescalares

- ✓ **Renomeação de registradores.**
  - A combinação de operandos em buffers (estações de reserva) e os resultados no buffer de reordenação resulta em uma forma de **renomeação de registradores**.
  - **Como?**
    - Seja lá onde estiver os operandos (banco de registradores, buffer de reordenação ou não existir ainda), em algum momento ele será armazenado na estação de reserva.
      - Sempre teremos registradores livres para armazenar novas variáveis - Não é necessário utilizar mais registradores.

Arquiteturas Paralelas

Introdução

- ✓ O conjunto de arquiteturas computacionais pode ser dividido em três grupos básicos:
  - **Arquiteturas seqüenciais;**
  - **Arquiteturas com mecanismos de ILP;**
    - Pipeline, superpipeline, superescalar...
  - **Arquiteturas paralelas.**

Arquitetura Paralela

- ✓ **Definição.**
  - Estrutura de alto nível para o desenvolvimento de soluções utilizando o processamento paralelo através de múltiplos processadores, que cooperam para resolver problemas através de execução concorrente.

Arquitetura Paralela

- ✓ **Motivação.**
  - Necessidade de maior desempenho computacional.
  - Restrições físicas, como a velocidade finita da luz, tornam difícil o aumento de velocidade em um único processador.
  - O desenvolvimento tecnológico possibilitou a construção de microprocessadores de alto desempenho que, agrupados, possibilitam um ganho significativo de poder computacional.
  - Tolerância a falhas – Redundância de hardware.
  - Muitas aplicações necessitam de paralelismo real.

Arquitetura Paralela

- ✓ **Desvantagens.**
  - Programação complexa.
  - Necessidade de técnicas de balanceamento de carga para melhor distribuição dos processos nos vários processadores.
  - Há necessidade de sincronismo e comunicação entre os processos, o que gera certa complexidade.
  - A sobrecarga gerada no sistema (por exemplo, na comunicação entre processos) impede que se tenha um *speedup* ideal.

Taxonomia de Flynn

- ✓ Classificação de arquiteturas paralelas.
- ✓ **Flynn (1972)** → Classificou um processo computacional baseado nos **fluxos de instruções** e **fluxos de dados**.
  - **Fluxo de instruções** → Seqüência de instruções (programas).
  - **Fluxo de dados** → Operandos. Se relacionam com as instruções.

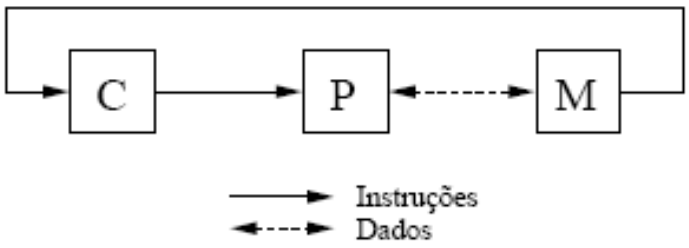
Classificação de Flynn

- ✓ Flynn propôs quatro categorias de sistemas de computação:

1. **Única instrução, único dado (SISD);**
2. **Única instrução, múltiplos dados (SIMD);**
3. **Múltiplas instruções, único dado (MISD);**
4. **Múltiplas instruções, múltiplos dados (MIMD).**

Classificação de Flynn

1. **Única instrução, único dado (SISD)** → Um único processador executa seqüencialmente um conjunto de instruções, usando dados armazenados em uma única memória. Ex.: uniprocessador (computadores pessoais).



- O fluxo de instruções alimenta uma unidade de controle (C) que ativa a unidade central de processamento (P).

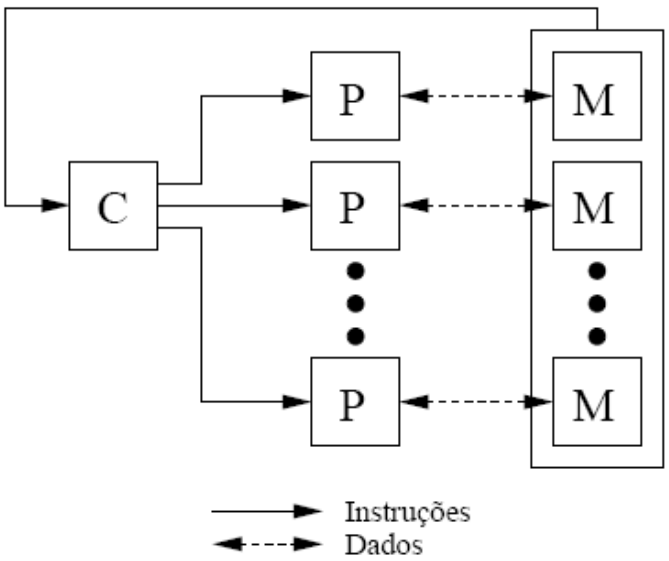
- A unidade P, por sua vez, atua sobre um único fluxo de dados, que é lido, processado e reescrito na memória (M).

Classificação de Flynn

2. **Múltiplas instruções, único dado (MISD)** → Vários processadores executam diferentes instruções operando no mesmo dado.
   - Categoria incomum.
   - **Não existem arquiteturas MISD.**

Classificação de Flynn

3. **Única instrução, múltiplos dados (SIMD)** → Uma única instrução é executada ao mesmo tempo sobre múltiplos dados.



Ex.: Processadores vetoriais e matriciais.

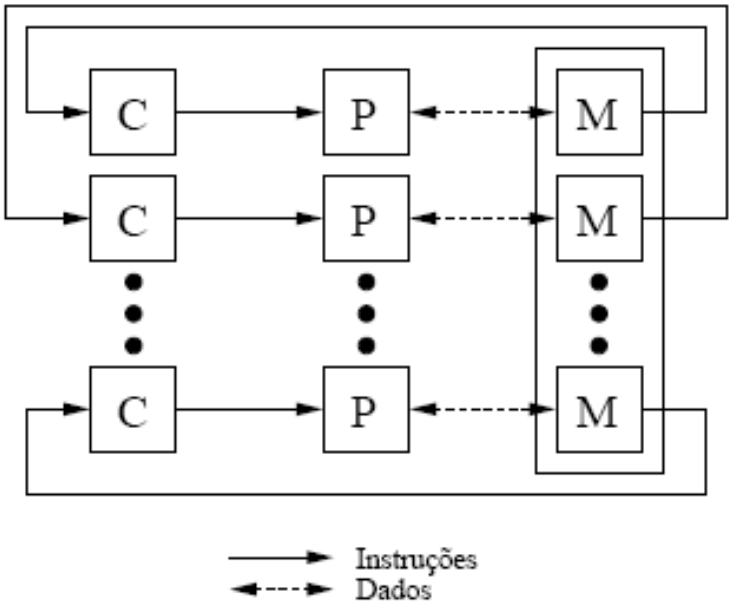
- O processamento é controlado por uma única unidade de controle (C), alimentada por um único fluxo de instruções. A mesma instrução é enviada para os diversos processadores (P).

- Todos os processadores executam suas instruções em paralelo de forma **síncrona** sobre diferentes fluxos de dados. **Na prática** → O mesmo programa está sendo executado sobre diferentes dados.

- Devido o processamento em paralelo, a unidade de memória M não pode ser implementada como um único módulo.

Classificação de Flynn

4. **Múltiplas instruções, múltiplos dados (MIMD)** → Um conjunto de processadores executa simultaneamente seqüencias diferentes de instruções, sobre conjuntos de dados distintos.



- Cada processador executa o seu próprio programa sobre seus próprios dados de forma **assíncrona.**

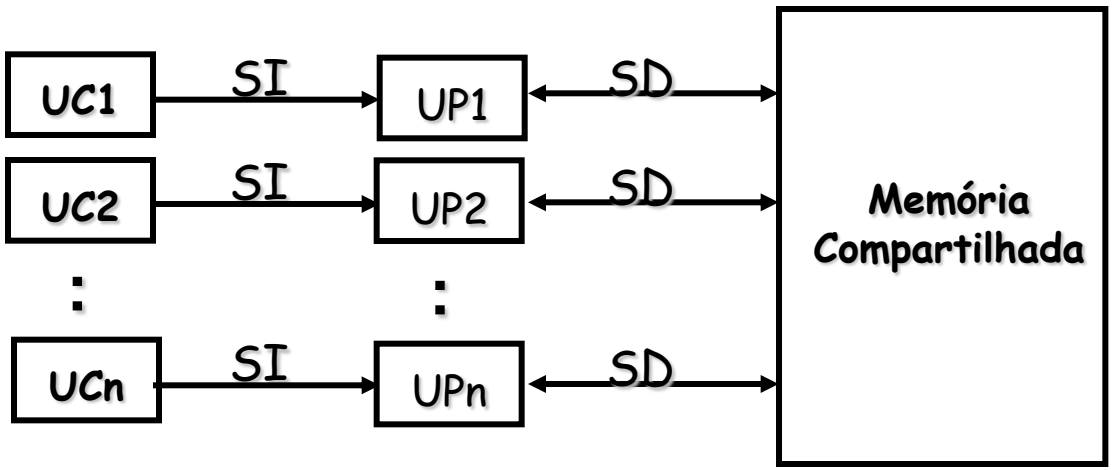
- Ex.: multiprocessadores e multicomputadores.

MIMD

- ✓ Sistemas MIMD são divididos de acordo com a forma de comunicação entre os processadores:
  - **Memória compartilhada**
  - **Memória distribuída.**

MIMD

- ✓ **Memória compartilhada**
  - Os programas e dados de cada processador são armazenados nessa memória.
  - A comunicação entre os processadores é realizada por meio desta memória.

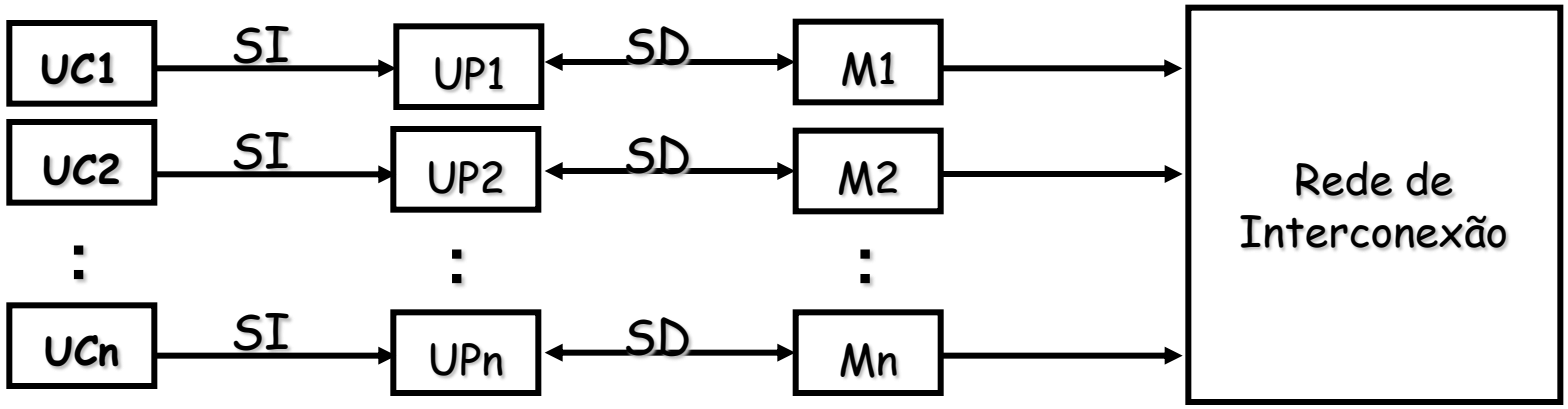


MIMD

- ✓ **Memória compartilhada**
  - A forma mais comum desse sistema é chamada de **multiprocessador simétrico (SMP)**.
    - Em um SMP, vários processadores compartilham uma única memória ou conjunto de memórias por meio de um barramento compartilhado ou algum outro tipo de mecanismo de interconexão.
    - O tempo de acesso a qualquer região da memória é aproximadamente o mesmo para cada processador.
      - Recentemente, uma nova organização chamada de **acesso não-uniforme à memória (NUMA)** foi desenvolvida – O tempo de acesso a diferentes regiões da memória pode ser diferente.

MIMD

- ✓ **Memória distribuída**
  - Um conjunto de uniprocessadores ou de SMPs independentes conectados via uma rede de interconexão → Forma um **cluster**.
    - A comunicação entre os computadores pode ser por meio de caminhos fixos ou de uma rede.



Taxonomia de Arquiteturas