

Aula 12

Organização de Computadores

Processadores Paralelos - Superescalar

Profa. Débora Matos

Contextualização de Paralelismo em nível de instrução

Vamos considerar a possibilidade de despachar 2 instruções por ciclo. O que muda?

1) O despacho de duas instruções por ciclo no MIPS exigirá a decodificação de 64 bits de instruções.

Instruction type	Pipe stages							
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

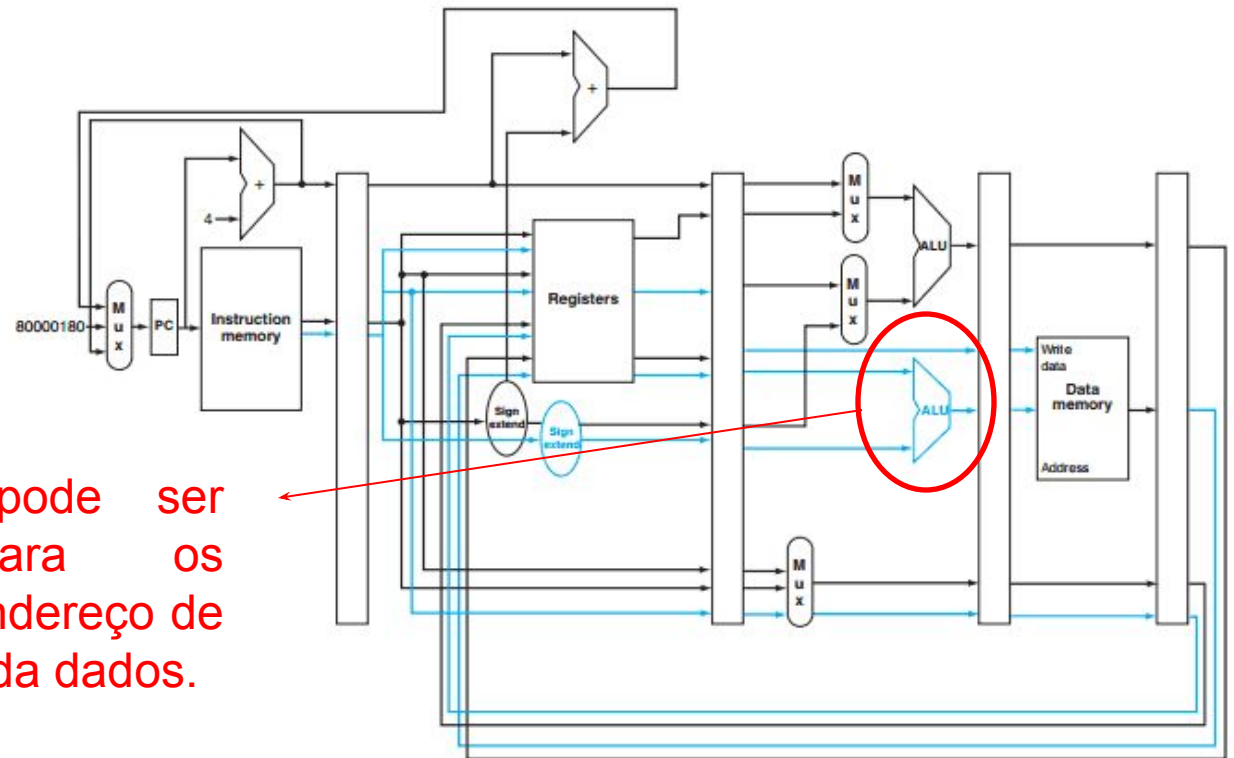
Contextualização de Paralelismo em nível de instrução

2) Para que as duas instruções sejam despachadas sem que ocorram hazards, o compilador ou um controle por hardware precisará removê-los.

3) É necessário adicionar portas extras ao banco de registradores.

4) É necessário adicionar um somador.

Esta ALU pode ser definida para os cálculos de endereço de transferência de dados.



Contextualização de Paralelismo em nível de instrução

A primeira ideia por trás do paralelismo em nível de instrução é a **adição de mais recursos de hardware**.

No entanto, verificaremos que uma série de necessidades e restrições precisam ser observadas e controladas nestas arquiteturas.

Exemplos:

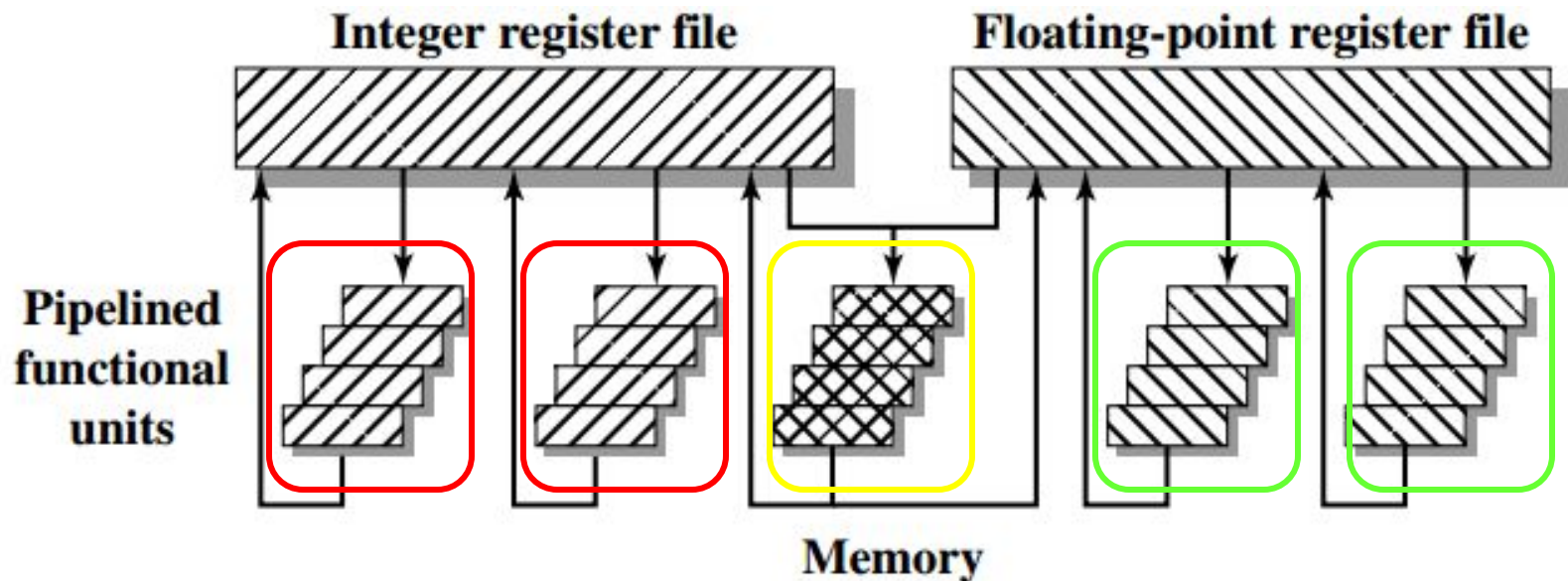
- Processadores superescalares
- VLIW – Very Long Instruction Word
- Processadores com superpipelines

Processadores superescalares

- Um processador superescalar utiliza múltiplos e independentes pipelines de instruções;
- Permite executar um número variado de instruções por ciclo de clock.
- Cada pipeline consiste de múltiplos estágios e pode lidar com múltiplas instruções ao mesmo tempo;
- Quando ocorrem dependências entre as instruções, o processador pode executá-las fora de ordem;

Duplicação de recursos

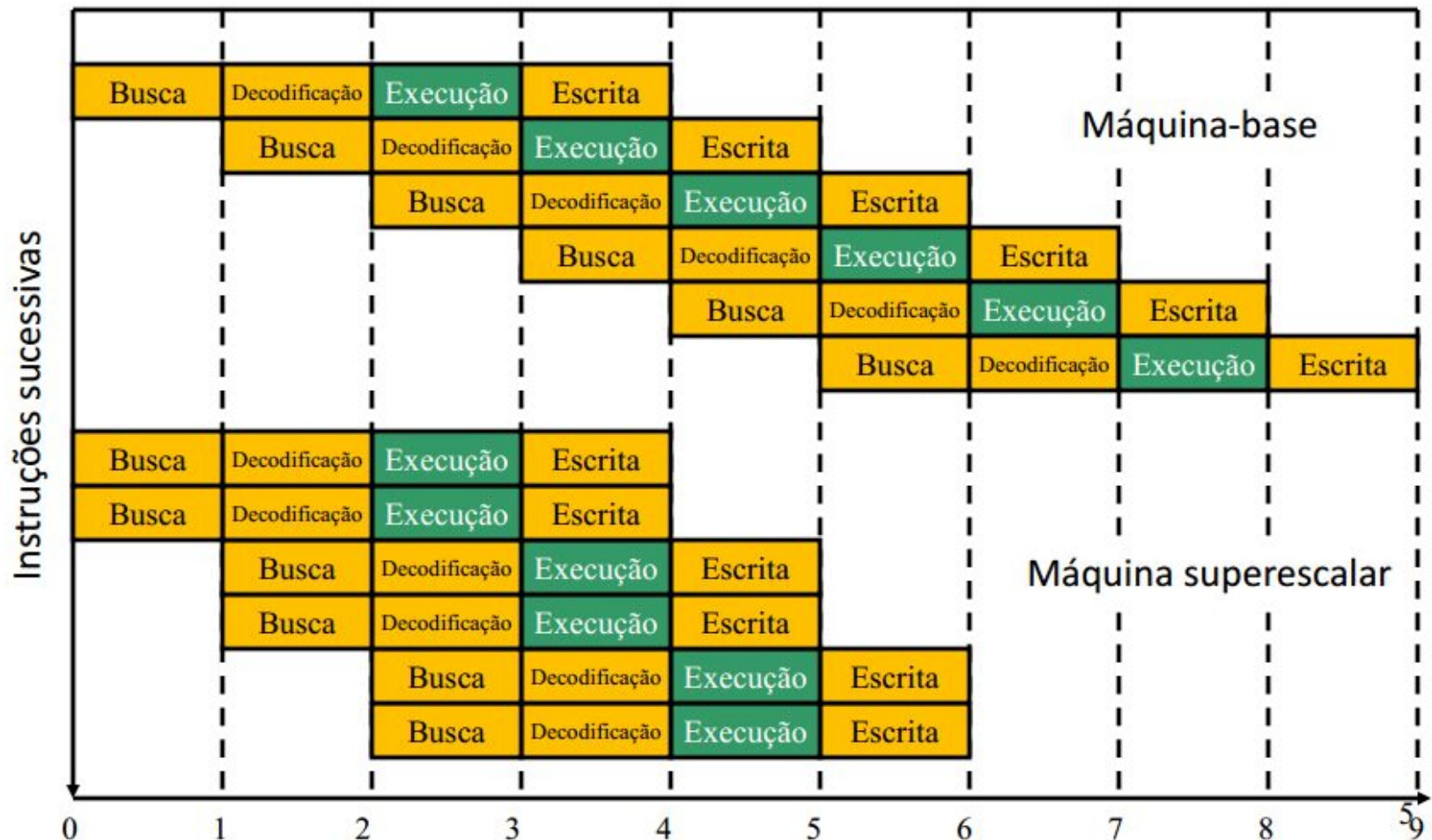
Modelo de arquitetura superescalar



- Cada unidade funcional implementa um pipeline, e cada pipeline permite a execução paralela de múltiplas instruções;

Processadores superescalares

Modelo de arquitetura superescalar

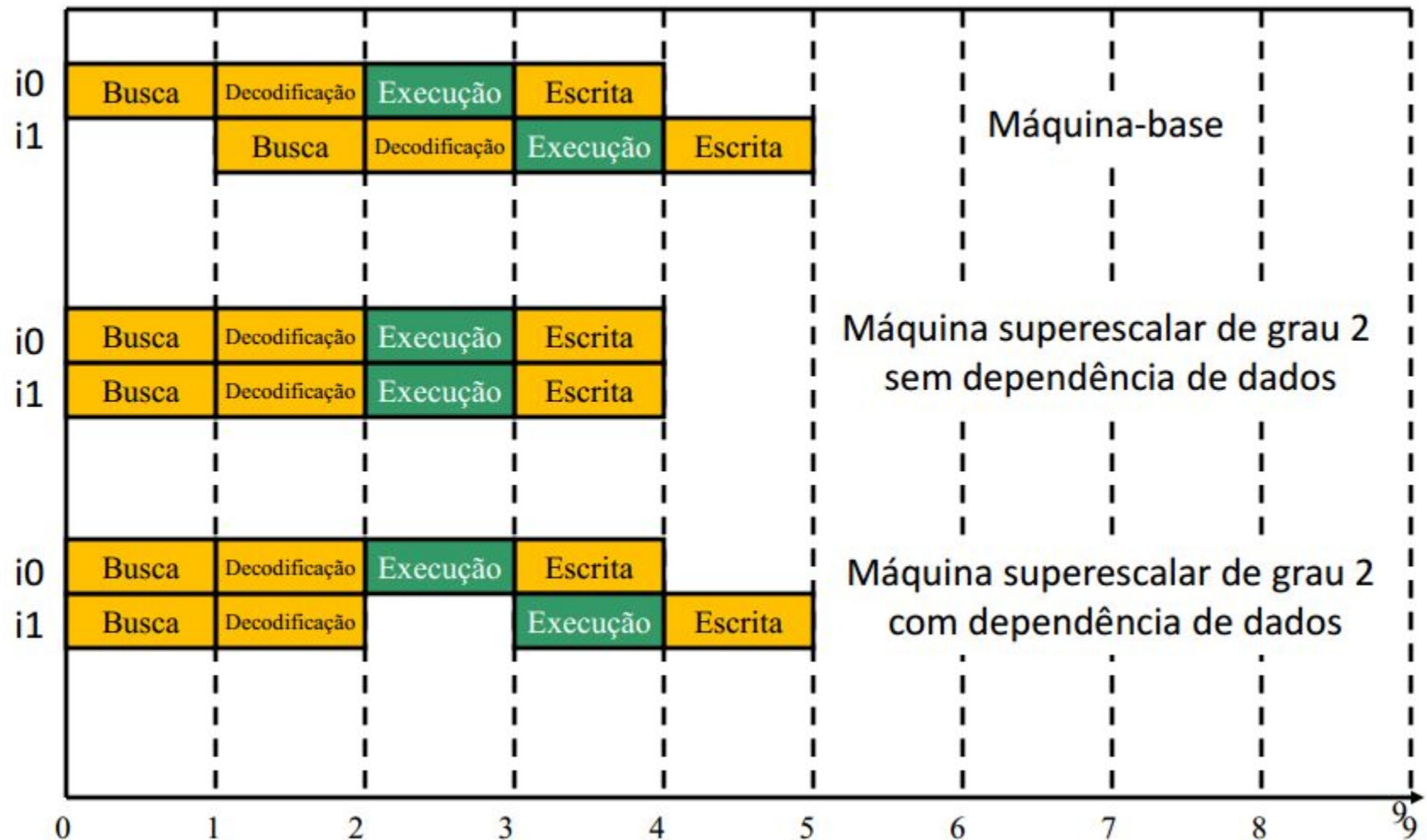


Processadores superescalares

No entanto, um processador superescalar precisa se preocupar com as seguintes situações:

- Dependência de dados
- Dependência procedural
- Conflitos de recursos
- Dependência de saída
- Antidependência

Processadores superescalares



Processadores superescalares

No entanto, um processador superescalar precisa se preocupar com as seguintes situações:

- Dependência de dados
- Dependência procedural
- Conflitos de recursos
- Dependência de saída
- Antidependência

Processadores superescalares

Dependência de saída:

Exemplo:

I1: $R3 \leftarrow R3 \text{ op } R5$

I2: $R4 \leftarrow R3 + 1$

I3: $R3 \leftarrow R5 + 1$

I4: $R7 \leftarrow R3 \text{ op } R4$

Processadores superescalares

Dependência de saída:

Exemplo:

I1: R3 <= R3 op R5

I2: R4 <= R3 + 1

I3: R3 <= R5 + 1

I4: R7 <= R3 op R4

- Instruções I1 e I3 escrevem em R3
- Atribuição da 1ª instrução não pode ser feita após a atribuição da 3ª
- Despacho da 3ª instrução precisa ser congelado.

Processadores superescalares

Antidependência:

Exemplo:

I1: $R3 \leftarrow R3 \text{ op } R5$

I2: $R4 \leftarrow R3 + 1$

I3: $R3 \leftarrow R5 + 1$

I4: $R7 \leftarrow R3 \text{ op } R4$

Processadores superescalares

Antidependência:

Exemplo:

I1: R3 <= R3 op R5

I2: R4 <= R3 + 1

I3: R3 <= R5 + 1

I4: R7 <= R3 op R4

A instrução I3 não pode concluir sua execução antes da instrução I2 ter obtido seus operandos.

despacho da instrução I3 precisa ser congelado até que a instrução I2 tenha lido valor de R3.

Processadores superescalares

Com todas essas limitações, qual seria uma possível solução para otimizar os recursos do pipeline?

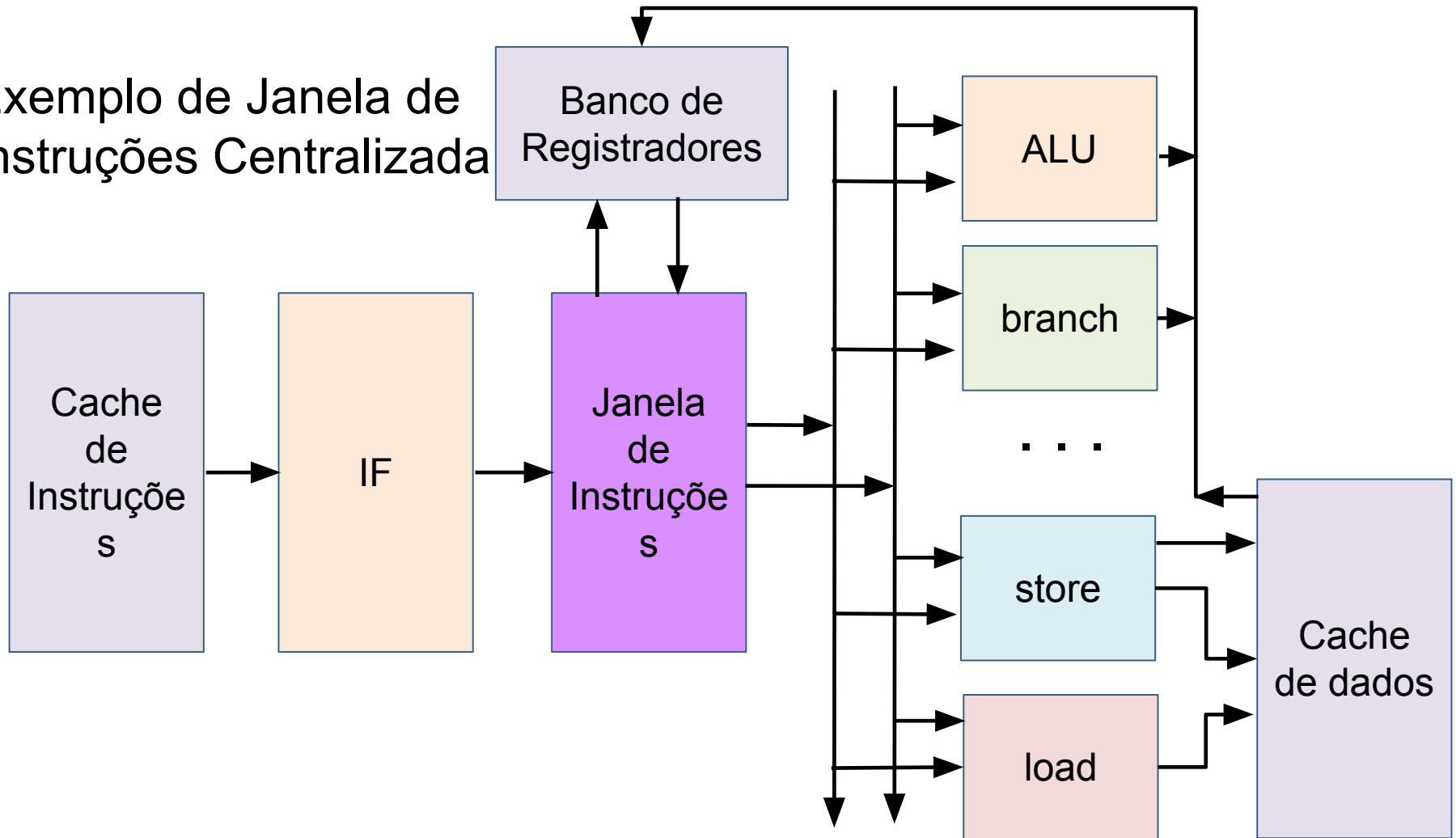
- Uma solução é alterar a ordem de execução das instruções;
- A restrição é de que o resultado da execução deve ser igual ao resultado obtido com a execução sequencial.
- No entanto, exige controles mais complexos para testar as dependências.

Execuções fora de ordem

- A **janela de instruções** é uma solução para permitir a execução fora-de-ordem com conclusão fora-de-ordem;
- Sempre que uma instrução **não apresenta dependências**, o processador pode decodificá-la e **colocá-la na janela de instruções**.
- Quando houver uma unidade funcional livre, uma instrução da janela de instruções pode ser emitida para o estágio de execução.

Execução fora de ordem

Exemplo de Janela de Instruções Centralizada



Renomeação de Registradores

- Quando as instruções são executadas fora de ordem, os valores dos registradores podem não ser conhecidos, considerando-se apenas a sequência de instruções do programa.
- As antidependências e dependências de saída são exemplos comuns de conflitos no uso de registradores em arquiteturas superescalares.
- Uma solução para este problema é utilizar a técnica de **Renomeação de Registradores**.

Renomeação de Registradores

Exemplo:

I1: $R3 \leftarrow R3 \text{ op } R5$

I2: $R4 \leftarrow R3 + 1$

I3: $R3 \leftarrow R5 + 1$

I4: $R7 \leftarrow R3 \text{ op } R4$

São utilizados registradores internos escolhidos dinamicamente.

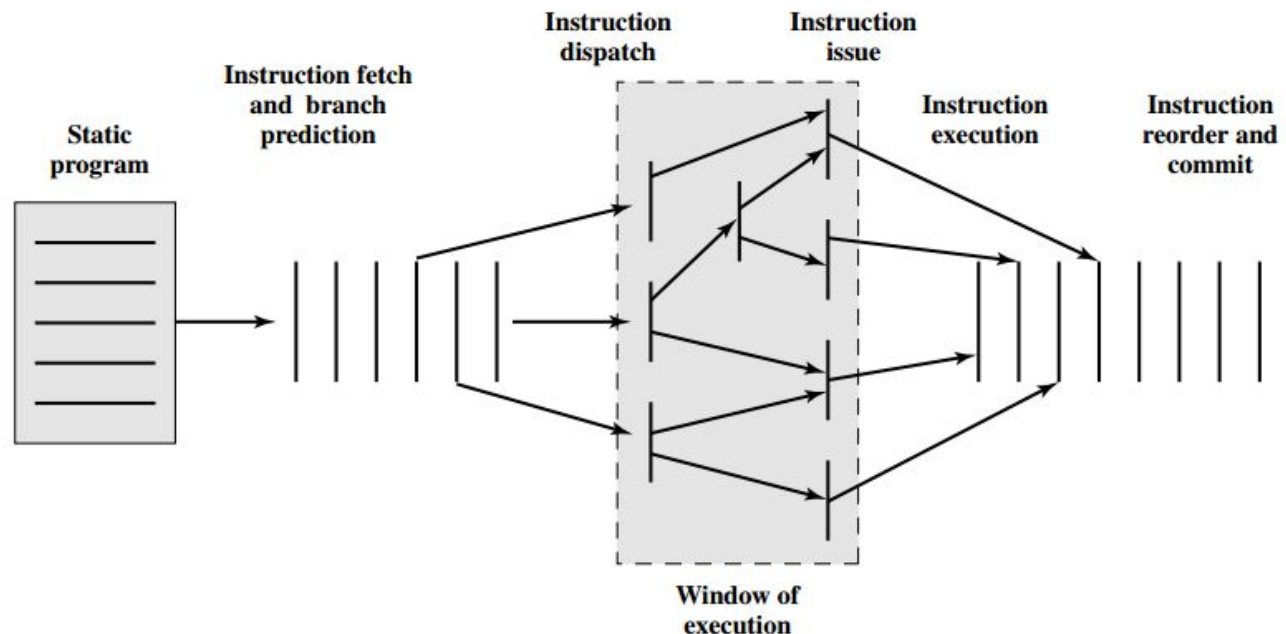
Elimina antidependência e dependência de saída

Previsão de Desvios

- Arquiteturas superescalares retomaram as técnicas de **previsão de desvio** empregadas antes da arquitetura RISC;
- Processadores mais simples empregam previsões estáticas, enquanto processadores mais sofisticados usam previsão dinâmica, baseando-se em históricos de desvios.

Projeto de processadores superescalares

- O processo de busca de instruções (incluindo previsão de desvio) constrói um **fluxo dinâmico de instruções**, onde o processador identifica e tenta eliminar dependências;
- As instruções são despachadas para uma janela de execução, onde **não formam mais um fluxo sequencial**;
- As instruções são **reordenadas de acordo com as dependências** de dados verdadeiras e recolocadas em um novo fluxo sequencial.



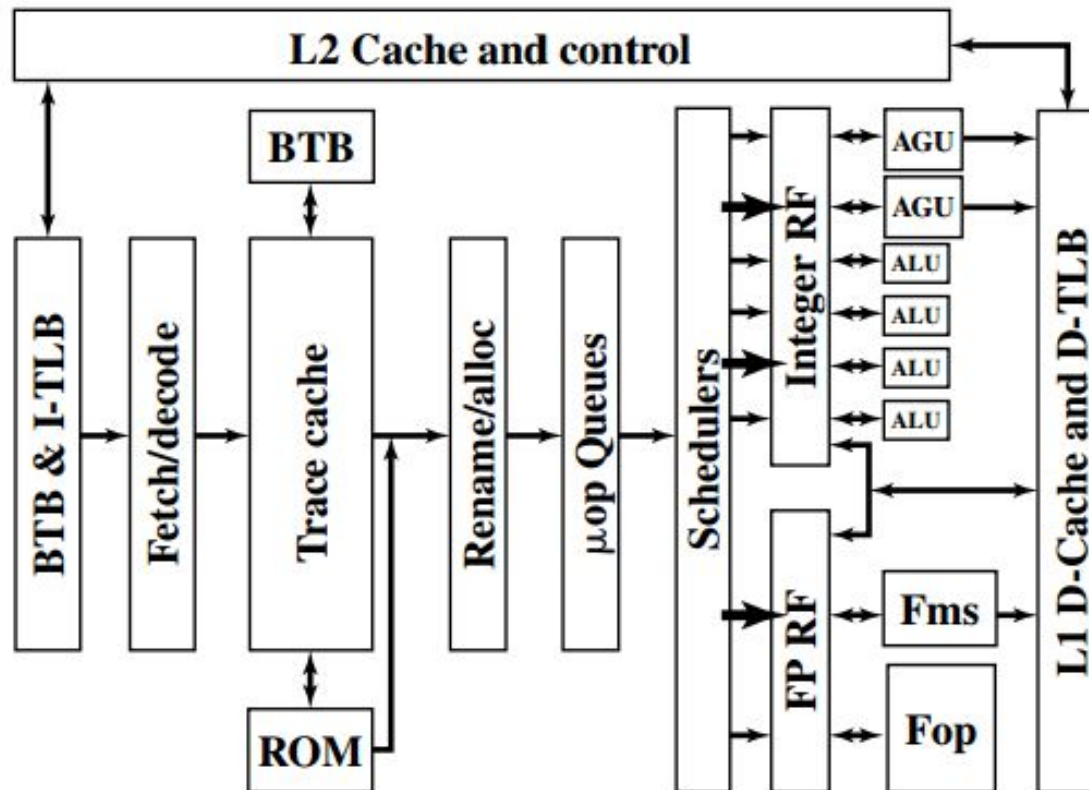
Projeto de processadores superescalares

Requerem:

- Uso de múltiplos estágios de busca e decodificação e uma lógica de **previsão de desvio**.
- Lógica para determinar dependências verdadeiras, envolvendo registradores e mecanismos para transferir os valores (**renomeação de registradores**);
- Mecanismos para emitir múltiplas instruções em paralelo;
- **Recursos adicionais de HW**: memória, unidades funcionais, registradores...
- Mecanismo para concluir o estado do processo na ordem correta (**buffer de reordenamento**).

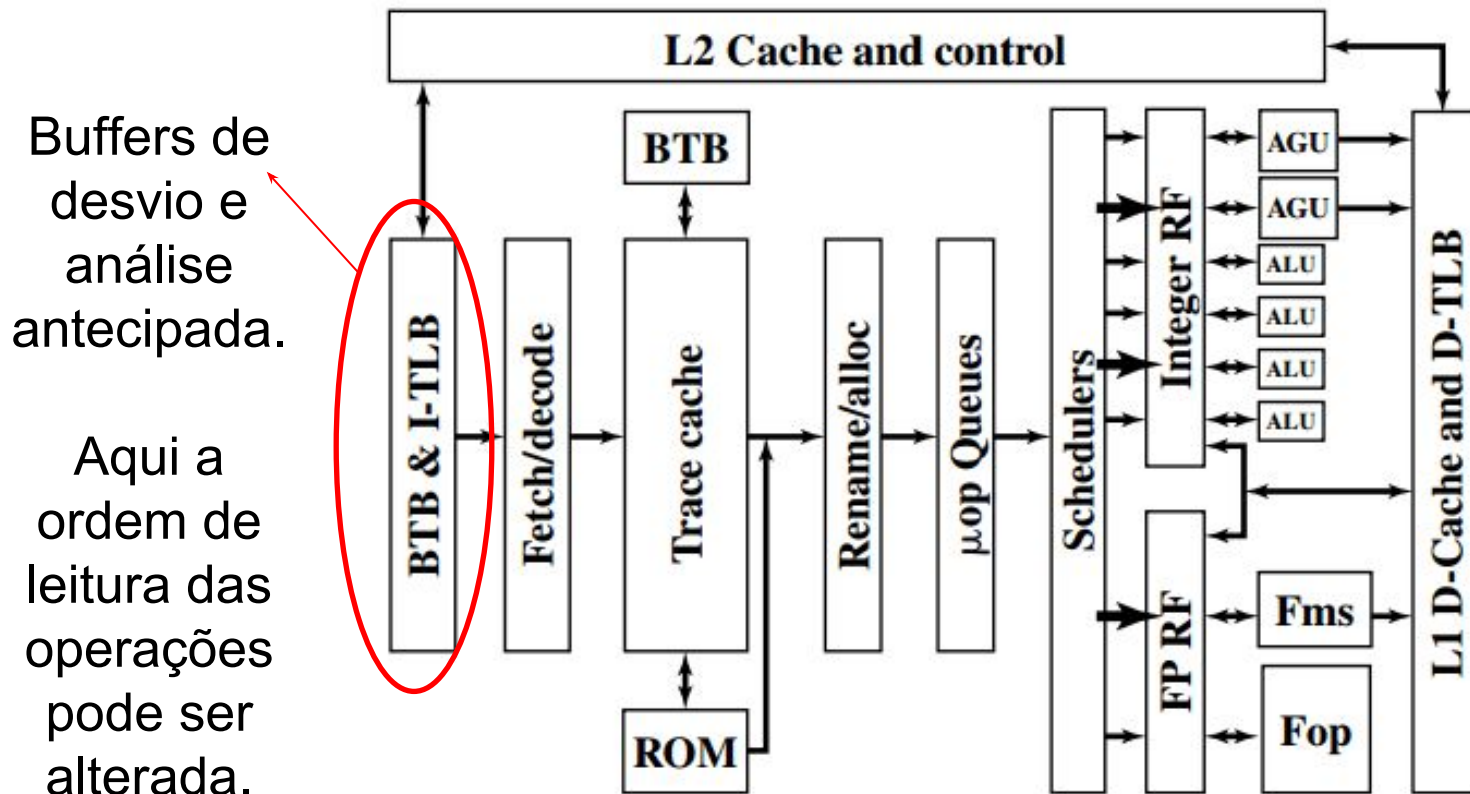
Projeto de processadores superescalares

Exemplo de arquitetura superescalar: Pentium 4



Projeto de processadores superescalares

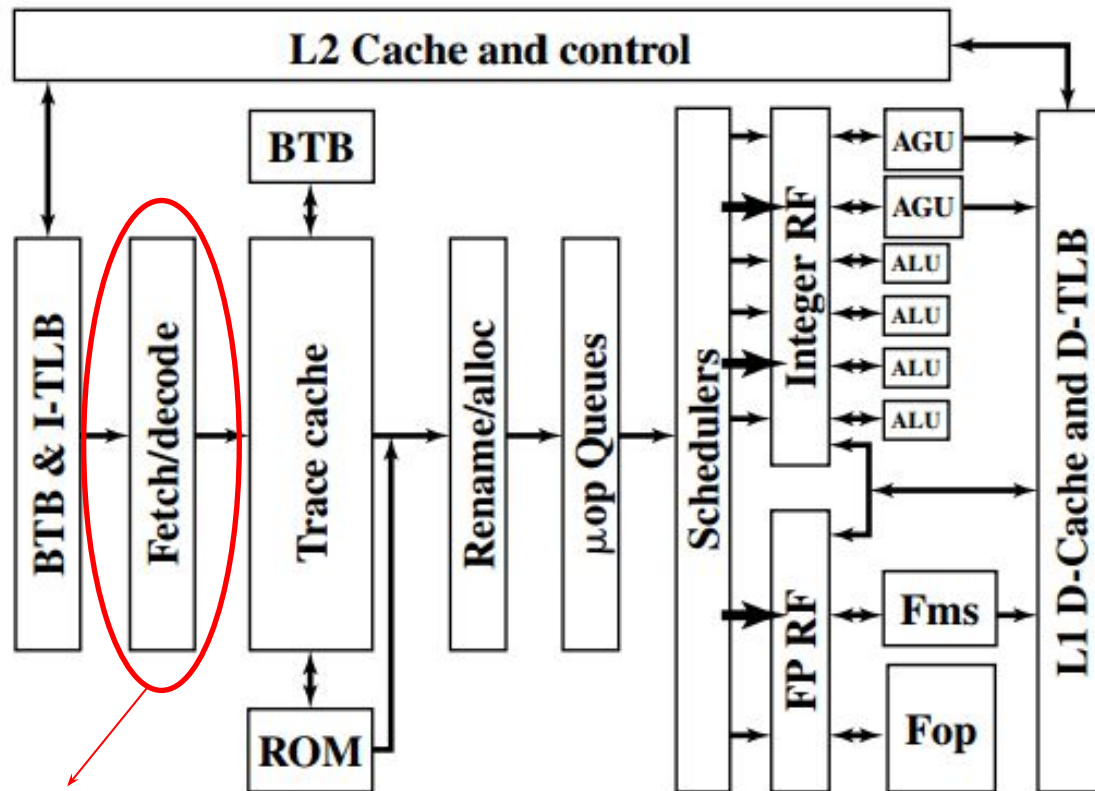
Exemplo de arquitetura superescalar: Pentium 4



O Pentium 4 usa uma estratégia dinâmica de previsão de desvios baseada em histórico. O campo de histórico é composto por 4 bits.

Projeto de processadores superescalares

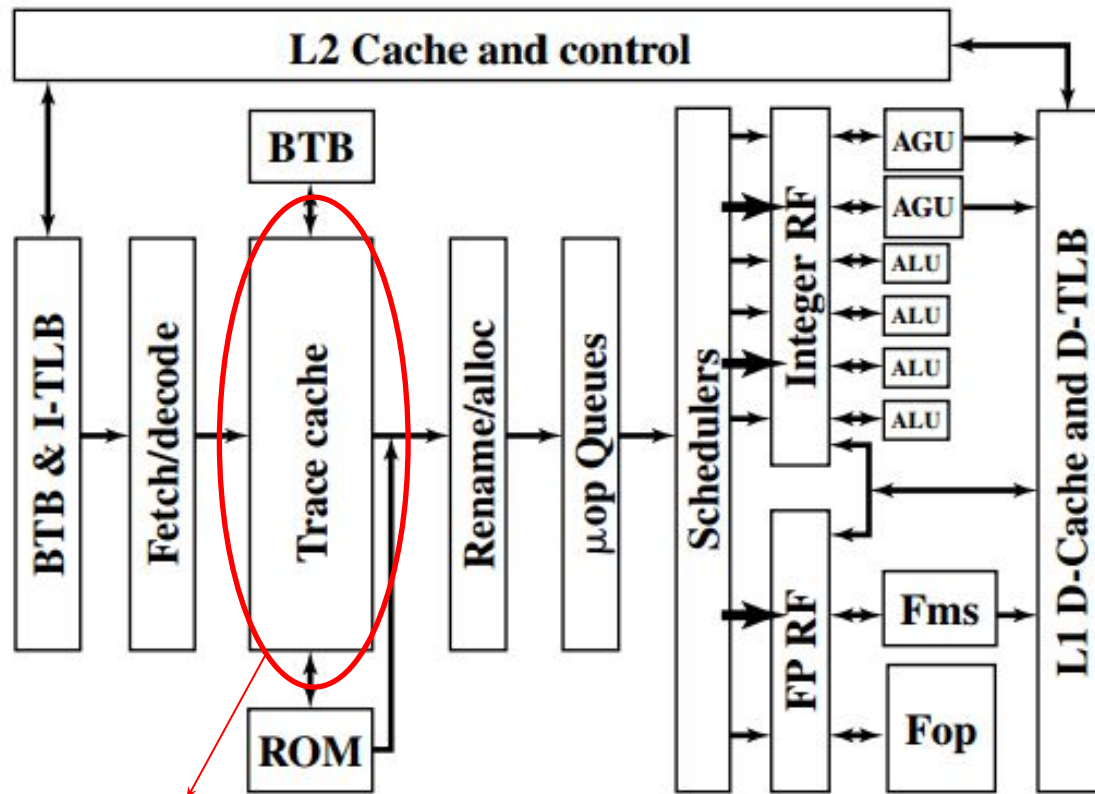
Exemplo de arquitetura superescalar: Pentium 4



O decodificador traduz cada instrução para até 4 micro-ops (instruções RISC, cada uma com 118 bits)

Projeto de processadores superescalares

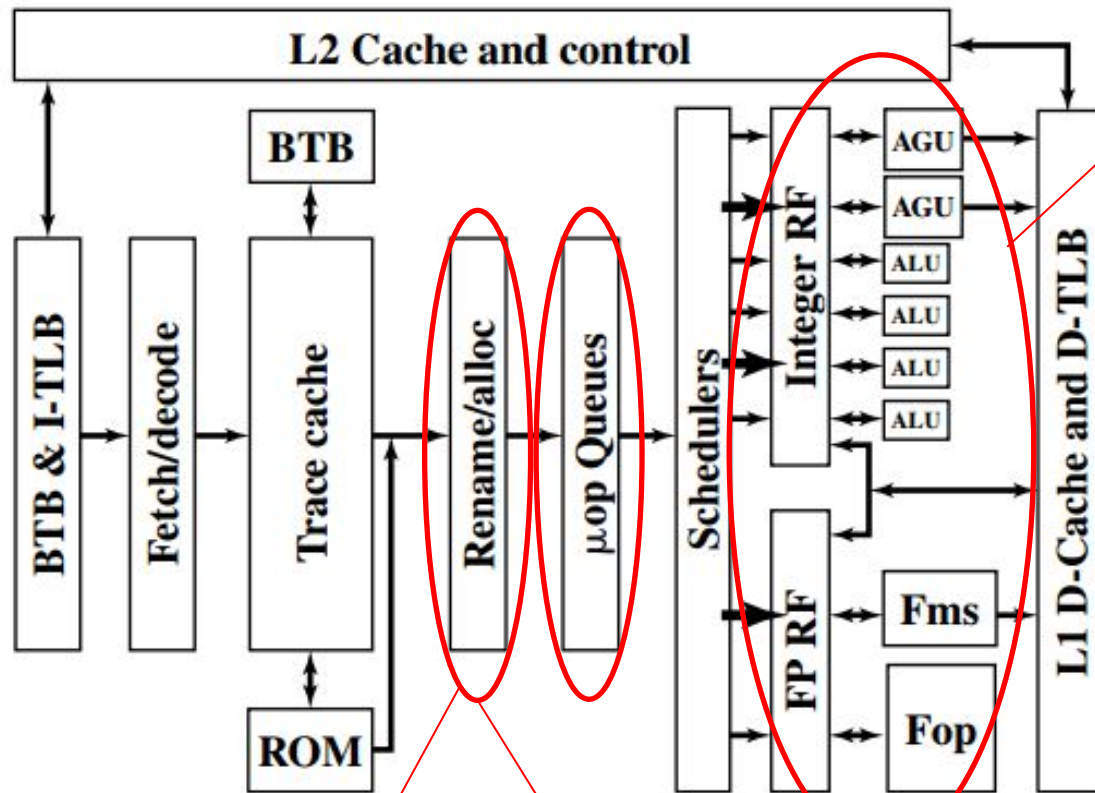
Exemplo de arquitetura superescalar: Pentium 4



As micro-ops geradas são guardadas no trace cache.

Projeto de processadores superescalares

Exemplo de arquitetura superescalar: Pentium 4



Após a verificação se a micro-op possui todos os **operandos** requeridos e se a **unidade de execução** estiver **disponível**, a micro-op é **despachada**.

Verifica se um recurso está indisponível para alguma das micro-ops que chegam no alocador.

O alocador é ainda responsável por alocar registradores necessários para a execução das micro-ops. (ex. renomeação de registradores)

Revisão de arquiteturas VLIW

- Várias operações (o que seriam instruções em uma máquina normal) são codificadas em uma mesma instrução;
- A palavra de instrução é bastante longa, podendo conter várias operações (que operam sobre vários operandos) independentes;
- A posição de cada operação dentro da palavra VLIW determina a unidade funcional que será usada;
- O hardware de despacho é simples quando comparado ao superescalar;



Apresente pelo menos 3 características
que diferem um projeto de um
superscalar de um VLIW:

VLIW x Superscalar

	VLIW	Superscalar
Número de instruções por clock	Utilizam um número fixo de instruções	Variam o número de instruções por clock
Dependências	Resolvidas em tempo de compilação pelo compilador	Resolvidas em tempo de execução por um hardware dedicado
Consumo de energia	Baixo	Alto
Tamanho do código	Maior	Menor
Compatibilidade através de gerações de HW	Não	Sim

Exercício

- 1) Supondo um processador superescalar com a seguinte configuração: 4 unidades funcionais, 1 multiplicador, 1 load/store
- Pode executar 4 instruções por ciclo em cada estágio do pipeline
 - Latências: Somador – 1 ciclo, Multiplicador – 2 ciclos, Load/ store – 2 ciclos

Deve ser executado o seguinte programa:


```
ADD R1, R2, R3
LW R10, 100 (R5)
ADD R5, R1, R6
MUL R7, R4, R8
ADD R2, R7, R3
ADD R9, R4, R10
ADD R11, R4, R6
```

Verificar as dependências e classificá-las

Exercício

ciclo	somador 1	somador 2	multiplicador	load/store
1	$R1 = R2 + R3$	$R11 = R4 + R6$	$R7 = R4 * R8$	$R10 = \text{mem}(R5+100)$
2	$R5 = R1 + R6$		$R7 = R4 * R8$	$R10 = \text{mem}(R5+100)$
3	$R9 = R4 + R10$	$R2 = R7 + R3$		

ADD R1, R2, R3
 LW R10, 100(R5)
 ADD R5, R1, R6
 MUL R7, R4, R8
 ADD R2, R7, R3
 ADD R9, R4, R10
 ADD R11, R4, R6

 dependências verdadeiras

 dependências falsas