Arquitetura de Computadores II
1COP012

Paralelismo em nível de instruções e processadores superescalares

Paralelismo em nível de instruções e processadores superescalares

Objetivos

- Explicar a diferença entre abordagens superescalares e de superpipelines.
- Definir o paralelismo em nível de instrução.
- Discutir dependências e conflitos de recursos como limitações ao paralelismo em nível de instrução.
- Comparar e contrastar as técnicas para melhorar o desempenho de pipeline em máquinas RISC e máquinas superescalares.

Paralelismo em nível de instruções e processadores superescalares

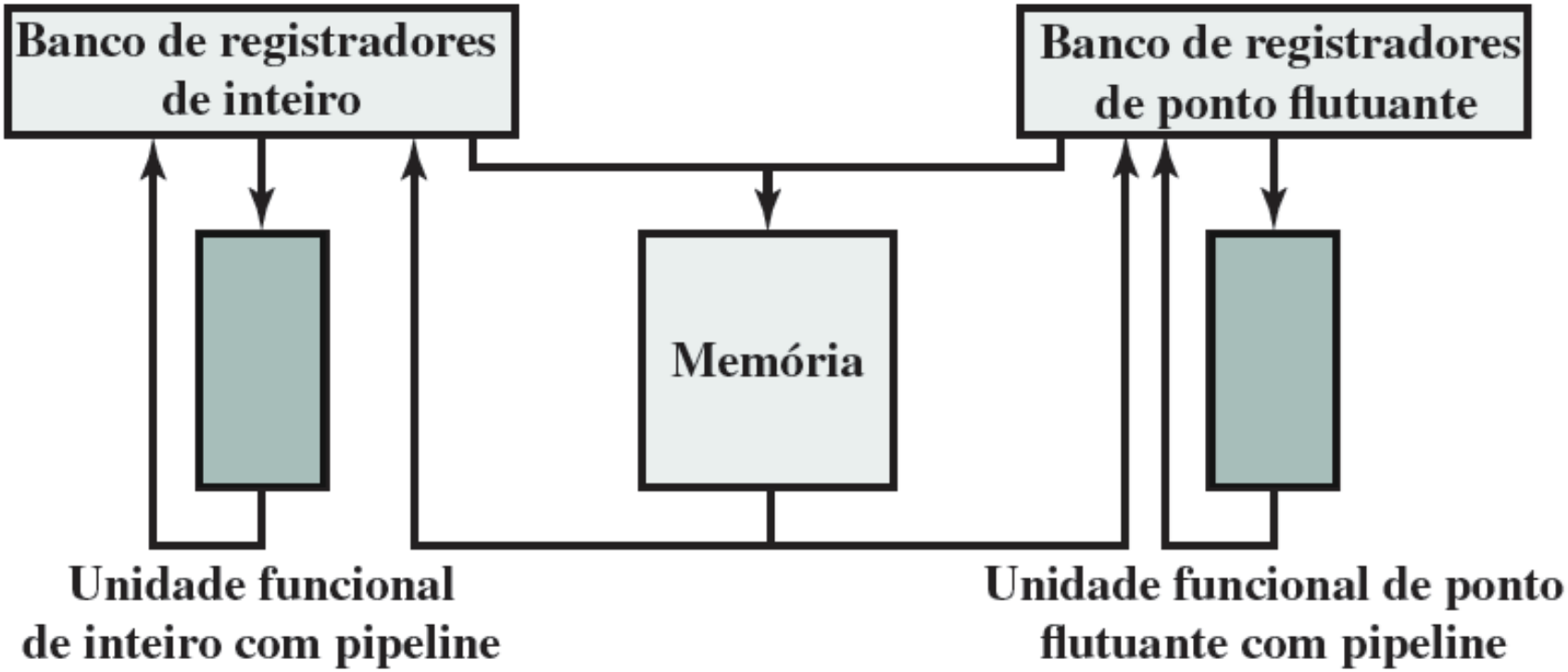
Visão geral

- A essência da abordagem superescalar é a habilidade de executar instruções independente e concorrentemente em pipelines diferentes.
- O conceito pode ser explorado permitindo que as instruções sejam executadas em uma ordem diferente da do programa.
- Muitos pesquisadores investigaram processadores do tipo superescalar e suas pesquisas indicam que algum grau de melhoria de desempenho é possível.
- As figuras que seguem comparam, em termos gerais, as abordagens escalares e superescalares.

Paralelismo em nível de instruções e processadores superescalares

Visão geral

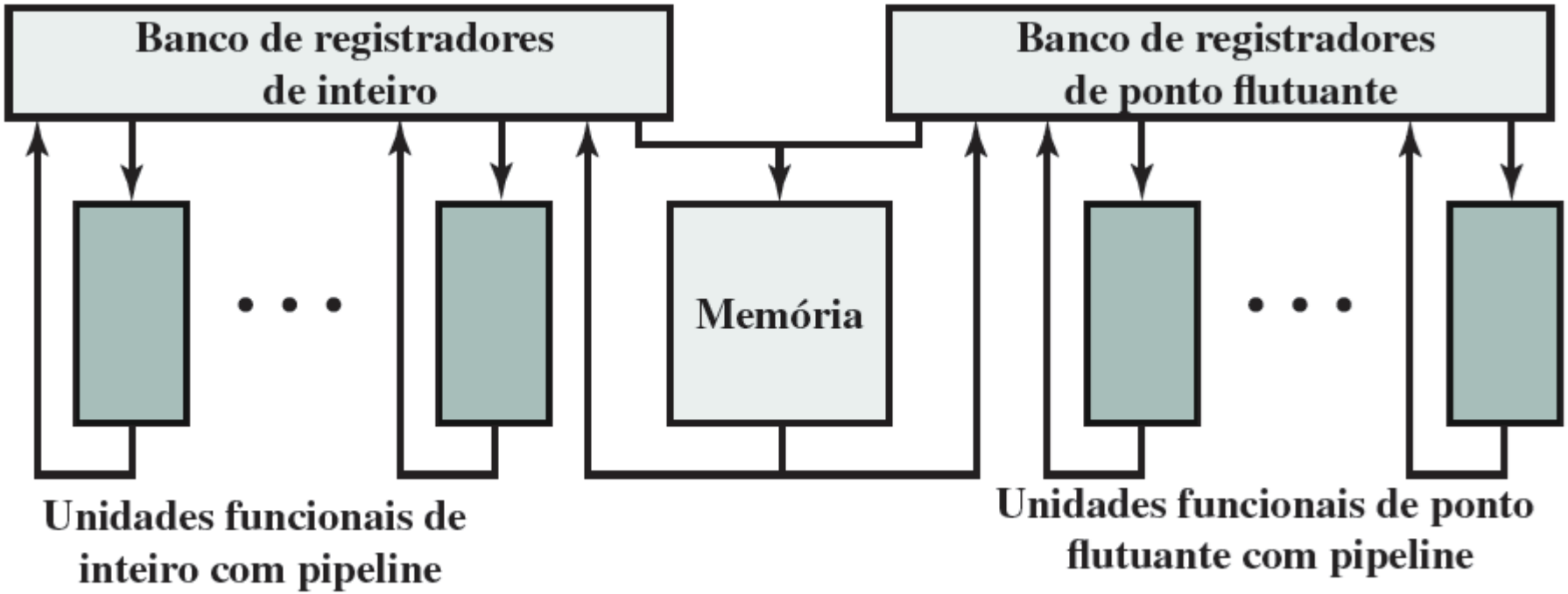
- Organização escalar:



Paralelismo em nível de instruções e processadores superescalares

Visão geral

- Organização superescalar:



Paralelismo em nível de instruções e processadores superescalares

Visão geral

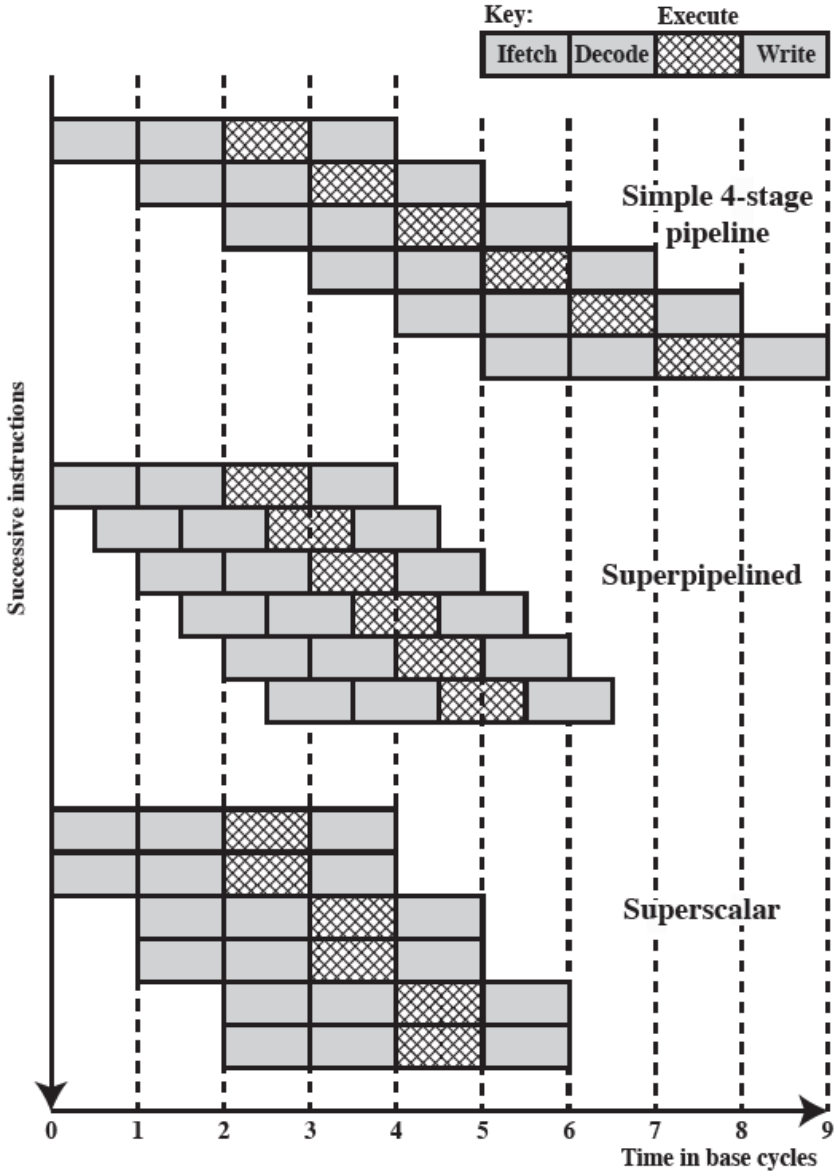
- A abordagem superescalar depende da habilidade de executar múltiplas instruções em paralelo em pipelines diferentes.
- O termo **paralelismo em nível de instruções** refere-se **ao grau em que**, **em média**, as instruções de um programa podem ser executadas em paralelo.
- Uma combinação de otimização baseada em compilador e técnicas de hardware pode ser usada para maximizar o paralelismo em nível de instruções.

Superescalar x Superpipeline

- O superpipeline explora o fato de que muitos estágios de pipeline executam tarefas que requerem menos do que metade de um ciclo de *clock*.
- Desse modo, a velocidade interna de *clock* dobrada possibilita o desempenho de duas tarefas em um ciclo de clock externo.

Paralelismo em nível de instruções e processadores superescalares

Superescalar x Superpipeline



Paralelismo em nível de instruções e processadores superescalares

Limitações

- O paralelismo em nível de instruções deve lidar com cinco limitações:

1. Dependência verdadeira de dados
2. Dependência procedural
3. Conflito de recursos
4. Dependência de saída
5. Antidependência

Paralelismo em nível de instruções e processadores superescalares

Limitações

- O paralelismo em nível de instruções deve lidar com cinco limitações:

1. Dependência verdadeira de dados (leitura após escrita RAW)

```
ADD EAX, ECX    ; carrega o registrador EAX com o conteúdo de ECX mais o conteúdo de EAX
MOV EBX, EAX    ; carrega EBX  com o conteúdo de EAX
```

Paralelismo em nível de instruções e processadores superescalares

Limitações

- O paralelismo em nível de instruções deve lidar com cinco limitações:

2. Dependências procedurais

As instruções que vem depois de um desvio (tomado ou não) possuem uma dependência procedural com o desvio e não podem ser executadas até que este seja executado.

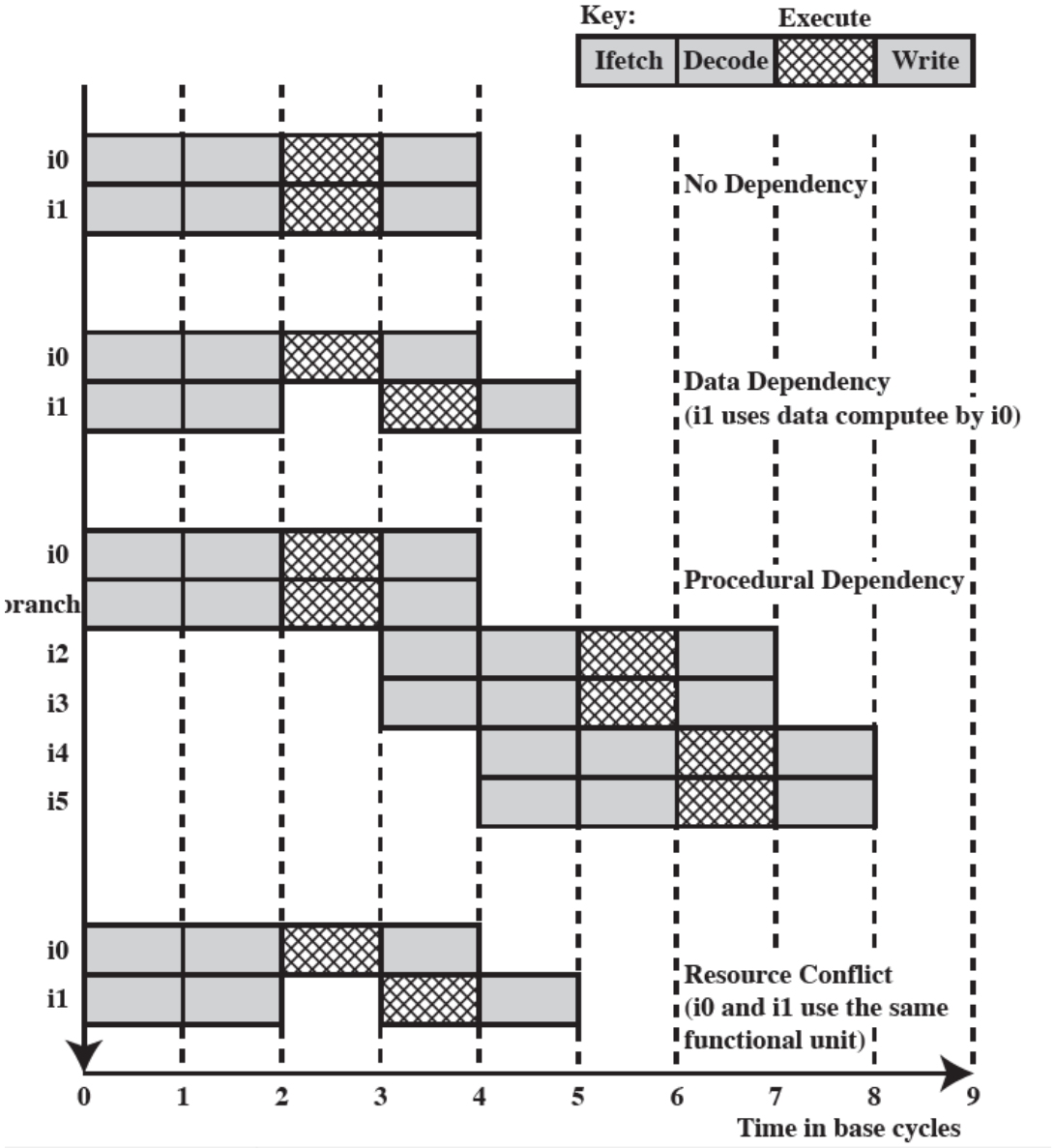
Limitações

- O paralelismo em nível de instruções deve lidar com cinco limitações:

3. Conflito de recursos

É uma concorrência de duas ou mais instruções pelo mesmo recurso e ao mesmo tempo. Exemplos de recursos incluem memória, cache, barramentos, entradas para banco de registradores e unidades funcionais (ex. somador da ALU).

Efeito de dependências



Paralelismo em nível de instruções e processadores superescalares

Aspectos de projeto – Paralelismo de instrução x Paralelismo de máquina

- O **paralelismo em nível de instrução** existe quando as instruções de uma sequência são independentes e, assim, podem ser executadas em paralelo por sobreposição.

- Como um exemplo do conceito de paralelismo, considere dois pedaços de código a seguir (JOUPPI, 1989b):

```
Load R1 ← R2          Add R3 ← R3, "1"
Add R3 ← R3, "1"      Add R4 ← R3, R2
Add R4 ← R4, R2       Store [R4] ← R0
```

Paralelismo em nível de instruções e processadores superescalares

Aspectos de projeto

- O **paralelismo de máquina** é uma medida da habilidade do processador para obter vantagem do paralelismo em nível de instruções.
- Ele é determinado pelo número de instruções que podem ser obtidas e executadas ao mesmo tempo (o número de pipelines paralelos) e pela velocidade e sofisticação dos mecanismos que o processador usa para localizar instruções independentes.
- O paralelismo em nível de instruções e o de máquina são fatores importantes para melhorar o desempenho.

Política sobre emissão de instruções

- Em geral, podemos dizer que a emissão da instrução ocorre quando **a instrução é movida do estágio de decodificação para o primeiro estágio de execução do pipeline**.

- Essencialmente, o processador está tentando olhar para a frente do ponto atual de execução para localizar instruções que podem ser trazidas para o pipeline e executadas.

- Três tipos de ordenação são importantes nessa consideração:

1. A ordem em que as instruções são lidas.
2. A ordem em que as instruções são executadas.
3. A ordem em que as instruções atualizam o conteúdo dos registradores e as posições de memória.

Política sobre emissão de instruções

- Quanto mais sofisticado for o processador, menos ele estará sujeito a esta estrita relação entre essas ordens.

- Em termos gerais, podemos agrupar políticas de emissão de instruções superescalares nas seguintes categorias:

1. Emissão em ordem com conclusão em ordem.
2. Emissão em ordem com conclusão fora de ordem.
3. Emissão fora de ordem com conclusão fora de ordem.

Política sobre emissão de instruções

1. Emissão em ordem com conclusão em ordem

✓ A política mais simples de emissão de instruções é a ordem exata em que seria alcançada pela execução sequencial (emissão em ordem) e pela escrita do resultado na mesma ordem (conclusão em ordem).

Política sobre emissão de instruções

2. Emissão em ordem com conclusão fora de ordem

- ✓ A conclusão fora de ordem é usada em processadores RISC escalares para melhorar o desempenho das instruções que requerem múltiplos ciclos.
- ✓ Com conclusão fora de ordem, qualquer número de instruções pode estar no estágio de execução em qualquer tempo até o nível máximo do paralelismo de máquina através de todas as unidades funcionais.
- ✓ A emissão de instrução é parada por um conflito de recursos, uma dependência de dados ou uma dependência procedural.
- ✓ A conclusão fora de ordem requer uma lógica de emissão de instruções mais complexa do que a conclusão em ordem.

Política sobre emissão de instruções

3. Emissão fora de ordem com conclusão fora de ordem

- ✓ Para permitir emissão fora de ordem, é necessário separar os estágios de decodificação e execução do pipeline.
- ✓ A separação dos estágios é feita com um buffer conhecido como janela de instruções.
- ✓ Com essa organização, depois que o processador termina de codificar uma instrução ela é colocada na janela de instruções.
- ✓ Enquanto esse buffer não estiver cheio, o processador pode continuar a obter e decodificar novas instruções.

Política sobre emissão de instruções

3. Emissão fora de ordem com conclusão fora de ordem

- ✓ Quando uma unidade funcional se torna disponível no estágio de execução, uma instrução da janela de instruções pode ser emitida para o estágio de execução.
- ✓ Qualquer instrução pode ser emitida considerando que: (1) ela precisa de uma unidade funcional particular que esteja disponível; (2) nenhum conflito ou dependência bloqueie essa instrução.
- ✓ O resultado dessa organização é que o processador tem capacidade de olhar para a frente, o que permite que ele identifique instruções independentes que podem ser trazidas para o estágio de execução.

Política sobre emissão de instruções

3. Emissão fora de ordem com conclusão fora de ordem

- ✓ As instruções são emitidas a partir da janela de instruções sem se preocupar com a ordem do programa original.

Paralelismo em nível de instruções e processadores superescalares

Política de emissão e conclusão de instruções superescalares

Decode		Execute			Write		Cycle
I1	I2						1
I3	I4	I1	I2				2
I3	I4	I1					3
	I4			I3	I1	I2	4
I5	I6			I4			5
	I6		I5		I3	I4	6
			I6				7
					I5	I6	8

(a) In-order issue and in-order completion

- I1 requer dois ciclos para executar
- I2 e I4 competem pela mesma unidade funcional
- I5 depende do valor produzido por I4
- I5 e I6 competem pela mesma unidade funcional

Paralelismo em nível de instruções e processadores superescalares

Política de emissão e conclusão de instruções superescalares

Decode		Execute			Write		Cycle
I1	I2						1
I3	I4	I1	I2				2
	I4	I1		I3	I2		3
I5	I6			I4	I1	I3	4
	I6		I5		I4		5
			I6		I5		6
					I6		7

(b) In-order issue and out-of-order completion

- I1 requer dois ciclos para executar
- I2 e I4 competem pela mesma unidade funcional
- I5 depende do valor produzido por I4
- I5 e I6 competem pela mesma unidade funcional

Paralelismo em nível de instruções e processadores superescalares

Política de emissão e conclusão de instruções superescalares

Decode		Window	Execute			Write		Cycle
I1	I2							1
I3	I4	*I1,I2*	I1	I2				2
I5	I6	*I3,I4*	I1		I3	I2		3
		I4,I5,I6		I6	I4	I1	I3	4
		I5		I5		I4	I6	5
						I5		6

(c) Out-of-order issue and out-of-order completion

- I1 requer dois ciclos para executar
- I2 e I4 competem pela mesma unidade funcional
- I5 depende do valor produzido por I4
- I5 e I6 competem pela mesma unidade funcional

Renomeação de registradores

- Quando as instruções são emitidas e completadas na sequência, é possível especificar o conteúdo de cada registrador em cada ponto da execução.
- Quando as técnicas fora de ordem são usadas, os valores dos registradores não podem ser totalmente conhecidos em cada momento apenas considerando a sequência de instruções definidas pelo programa.
- A ocorrência de antidependências e dependências de saída são exemplos de conflito de armazenamento.
- Múltiplas instruções competem pelo uso de mesmas posições de registradores, gerando restrições de pipeline que retardam o desempenho.

Renomeação de registradores

- Uma maneira de enfrentar os conflitos de armazenamento é baseada em uma solução tradicional de conflitos de recursos: duplicação de recursos.
- Nesse contexto, a técnica é conhecida como **renomeação de registradores**.
- Basicamente, registradores são alocados dinamicamente pelo hardware do processador e são associados com os valores usados pelas instruções em vários pontos do tempo.
- Quando um novo valor de registrador é criado, um novo registrador é alocado para esse valor.

Renomeação de registradores

- As instruções subsequentes que acessam esse valor como operando de origem nesse registrador têm de passar pelo processo de renomeação:

- As referências de registradores nessas instruções precisam ser revisadas para que se refiram ao registrador que contém o valor necessário.

- Desse modo, a mesma referência do registrador original em várias instruções diferentes pode se referir a registradores reais diferentes, se valores diferentes são pretendidos.

Paralelismo de máquina

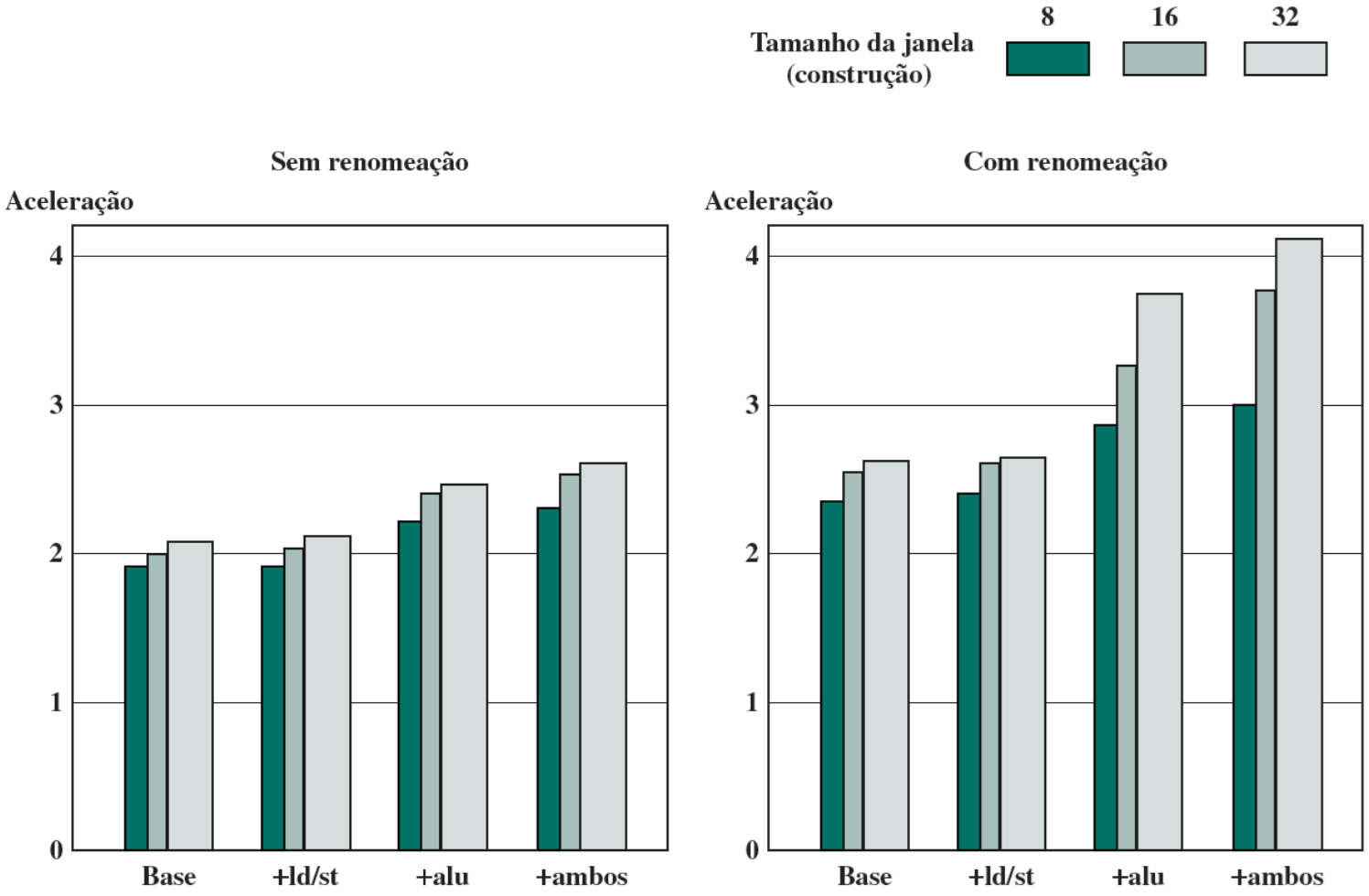
- Técnicas de hardware que podem ser usadas em um processador superescalar para melhorar o desempenho:

1. Duplicação de recursos
2. Emissão fora de ordem
3. Renomeação

Paralelismo em nível de instruções e processadores superescalares

Paralelismo de máquina

Acelerações de várias organizações de máquinas sem dependências procedurais:

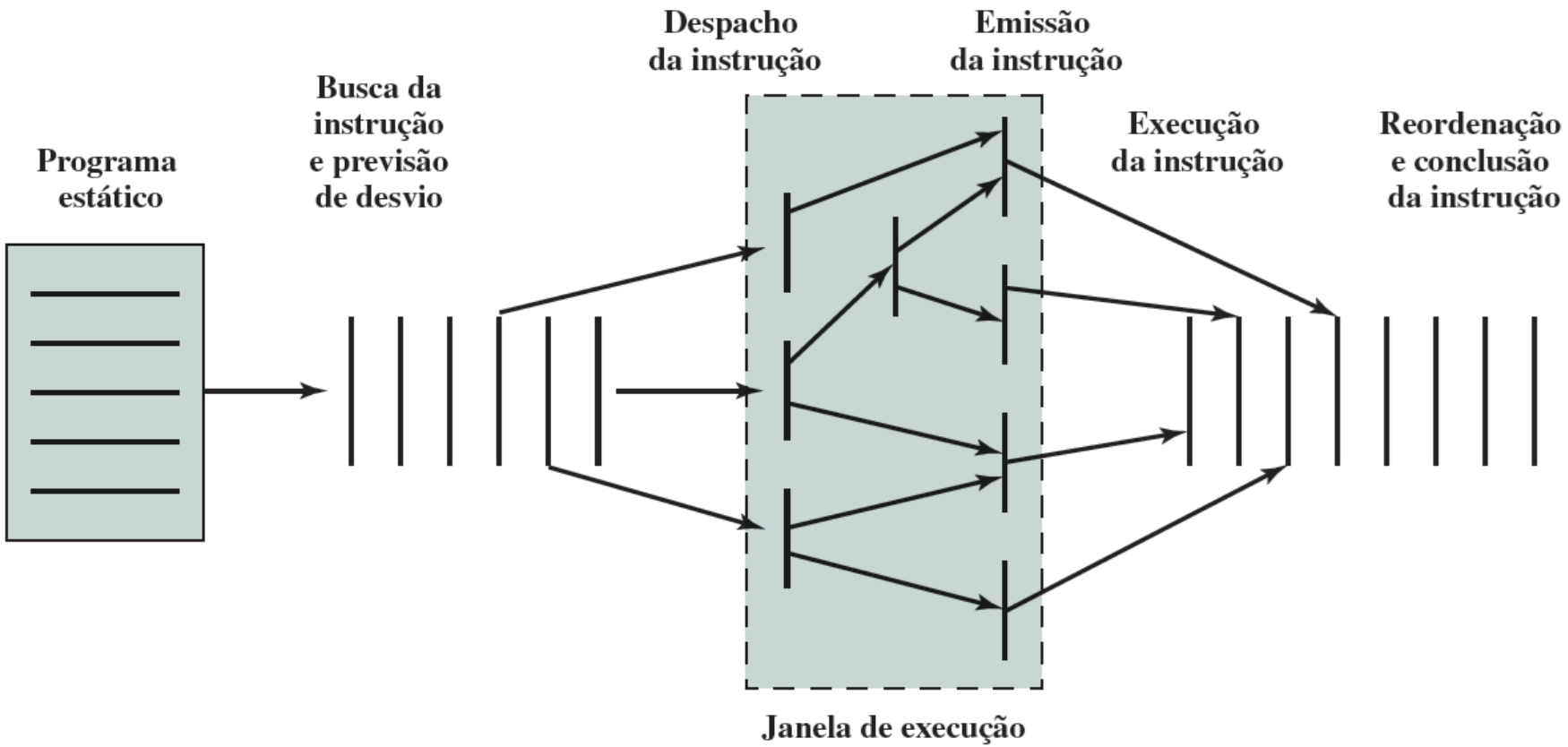


Previsão de desvio

- Com a chegada das máquinas RISC, a estratégia de desvio atrasado (*delayed branch*) foi explorada.
- Com o desenvolvimento de máquinas superescalares, a estratégia de desvio atrasado tem menos apelo.
- O motivo é que múltiplas instruções precisam ser executadas no slot de atraso (*delay slot*), trazendo vários problemas relacionados com as dependências das instruções.
- Desse modo, as máquinas superescalares retornaram às técnicas de previsão de desvios pré-RISC.

Execução superescalar

- Ilustração conceitual de processamento superescalar:



Paralelismo em nível de instruções e processadores superescalares

Execução superescalar

- O programa a ser executado consiste em uma sequência linear de instruções.
- Esse é o programa estático conforme escrito pelo programador ou gerado pelo compilador.
- O processo de **obter instrução**, o que inclui a previsão de desvio, é usado para formar um fluxo dinâmico de instruções.
- Esse fluxo é examinado para dependências e o processador pode remover as dependências artificiais.
- Ele então **despacha** as instruções para uma janela de execução.

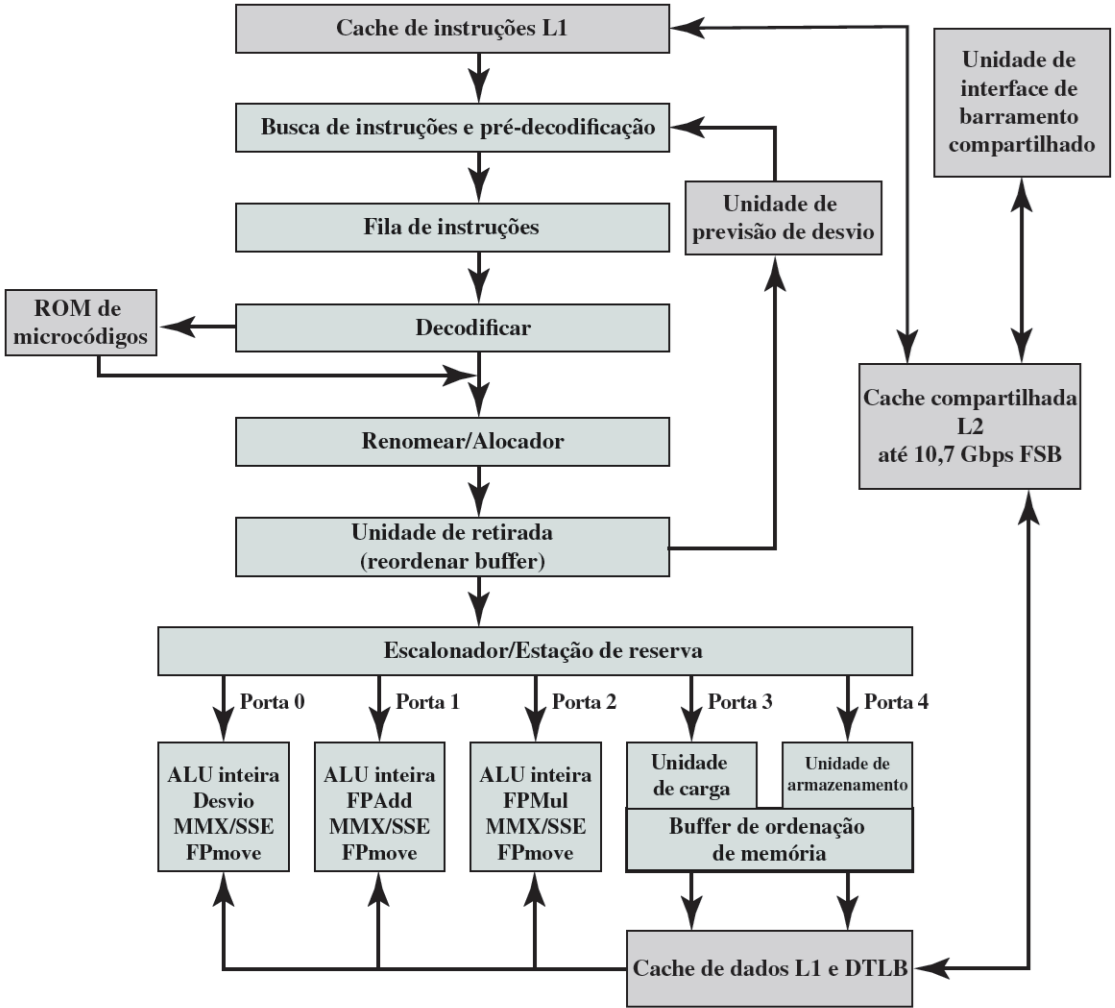
Execução superescalar

- Nessa janela, as instruções não formam mais um fluxo sequencial, mas são estruturadas de acordo com suas dependências verdadeiras de dados.
- O processador efetua o estágio de execução de cada instrução numa ordem determinada pelas dependências verdadeiras de dados e pela disponibilidade de recursos de hardware.
- Finalmente, as instruções são conceitualmente colocadas de volta na ordem sequencial e seus resultados são reordenados.
- O passo final, acima mencionado, é conhecido como **concluir**, ou **retirar**, a instrução.

Paralelismo em nível de instruções e processadores superescalares

Microarquitetura Intel Core

- Versão atual da arquitetura x86 com pipeline:



Paralelismo em nível de instruções e processadores superescalares

Microarquitetura Intel Core

- Parâmetros de cache/memória e desempenho dos processadores com base na microarquitetura Intel Core.

Nível de cache	Capacidade	Associatividade (formas)	Tamanho da linha (bytes)	Política de atualização
L1 dados	32 kB	8	64	*Write back*
L1 instruções	32 kB	8	N/A	N/A
L2 (compartilhada)[1]	2, 4 MB	8 ou 16	64	*Write back*
L2 (compartilhada)[2]	3, 6 MB	12 ou 24	64	*Write back*
L3 (compartilhada)[2]	8, 12, 16 MB	15	64	*Write back*

Obs.:
1. Intel Core Microarchitecture
2. Enhanced Intel Core Microarchitecture

Paralelismo em nível de instruções e processadores superescalares

Microarquitetura Intel Core

- Parâmetros de cache/memória e desempenho dos processadores com base na microarquitetura Intel Core.

Localidade de dados	Carga		Armazenamento	
	Latência	Taxa de transferência	Latência	Taxa de transferência
Cache de dados L1	3 ciclos de *clock*	1 ciclo de *clock*	2 ciclos de *clock*	3 ciclos de *clock*
Cache de dados L1 de outro *core* no estado modificado	14 ciclos de *clock* + 5,5 ciclos de barramento	14 ciclos de *clock* + 5,5 ciclos de barramento	14 ciclos de *clock* + 5,5 ciclos de barramento	N/A
Controle de pipeline da cache L2	14	3	14	3
Memória	14 ciclos de *clock* + 5,5 ciclos de barramento + latência de memória	Depende do protocolo de leitura de barramento	14 ciclos de *clock* + 5,5 ciclos de barramento + latência de memória	Depende do protocolo de leitura de barramento

Microarquitetura Intel Core

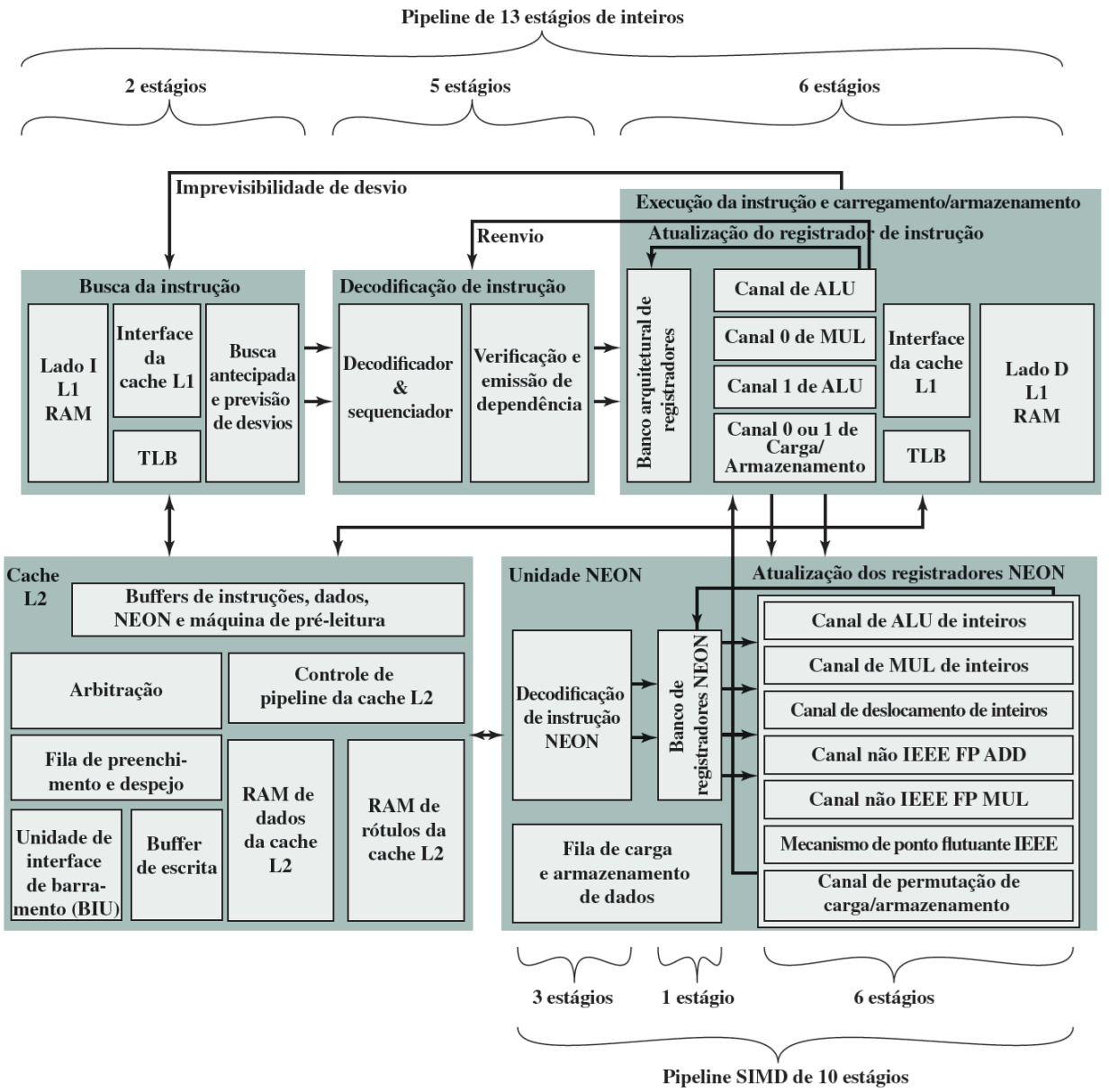
O pipeline da microarquitetura Intel Core contém:

- Um *front end* de emissão em ordem que busca fluxo de instruções a partir da memória.
- Um core de execução superescalar fora de ordem que pode emitir até seis micro-ops por ciclo e reordenar micro-ops para executar assim que as fontes estiverem prontas.
- Uma unidade de retirada em ordem que assegure que os resultados de execução dos micro-ops sejam processados e execução dos micro-ops sejam processados e o estado da arquitetura e o conjunto do registrador do processador atualizados **de acordo com a ordem do programa original**.

Paralelismo em nível de instruções e processadores superescalares

ARM Cortex-A8

O Cortex-A8 é um processador da família ARM, referido como processador de aplicação:



Unidade de busca de instruções

- A unidade de busca das instruções pode ler até quatro instruções por ciclo e passa pelos seguintes estágios:
- F0: a unidade de geração de endereços (AGU — do inglês, Address Generation Unit) gera um novo endereço virtual.
- F1: o endereço calculado é usado para obter instruções da cache de instruções L1.
- F3: dados da instrução são colocados na fila de instruções.
- A unidade de busca das instruções pode obter e enfileirar até 12 instruções.

Unidade de decodificação de instruções

- A preocupação principal do pipeline de decodificação de instruções é a prevenção de hazards RAW.

- Cada instrução passa por cinco estágios de processamento:

1. D0: instruções Thumb são descompactadas em instruções ARM de 32 bits.

2. D1: a função de decodificação de instruções é finalizada.

3. D2: esse estágio escreve e lê instruções da estrutura de fila pendente/reenvio.

Unidade de decodificação de instruções

4. D3: esse estágio contém a lógica de agendamento de instruções.

5. D4: efetua a decodificação final para todos os sinais de controle requeridos pelas unidades de execução de inteiros e de carga/armazenamento.

- Para lidar com os atrasos, um mecanismo de recuperação é usado para esvaziar todas as instruções subsequentes no pipeline de execução e reemiti-las (reenvio).

Unidade de execução de inteiros

A unidade de execução de instruções:

- Executa todas as operações de inteiros da ALU e de multiplicação, incluindo geração de flags.
- Gera os endereços virtuais para carga e armazenamento e o valor base de atualização, quando necessário.
- Fornece dados formatados para armazenamento e encaminha os dados e flags para a frente.
- Processa desvios e outras mudanças do fluxo de instruções e avalia os códigos condicionais das instruções.

Unidade de execução de inteiros

Para instruções da ALU, cada pipeline pode ser usado, consistindo nos seguintes estágios:

- E0: acessar o banco de registradores.
- E1: o registrador de deslocamento efetua a sua função, se necessário.
- E2: a unidade de ALU efetua a sua função.
- E3: se necessário, esse estágio completa saturação aritmética usada por algumas instruções ARM de processamento de dados.

Unidade de execução de inteiros

- E4: qualquer alteração no fluxo de controle, incluindo falhas na previsão de desvios, exceções e reenvios do sistema de memória são priorizados e processados.

- E5: resultados das instruções ARM são atualizados no banco de registradores.

Pipeline de carga/armazenamento ocorre em paralelo ao pipeline de inteiros.

Os estágios são os seguintes:

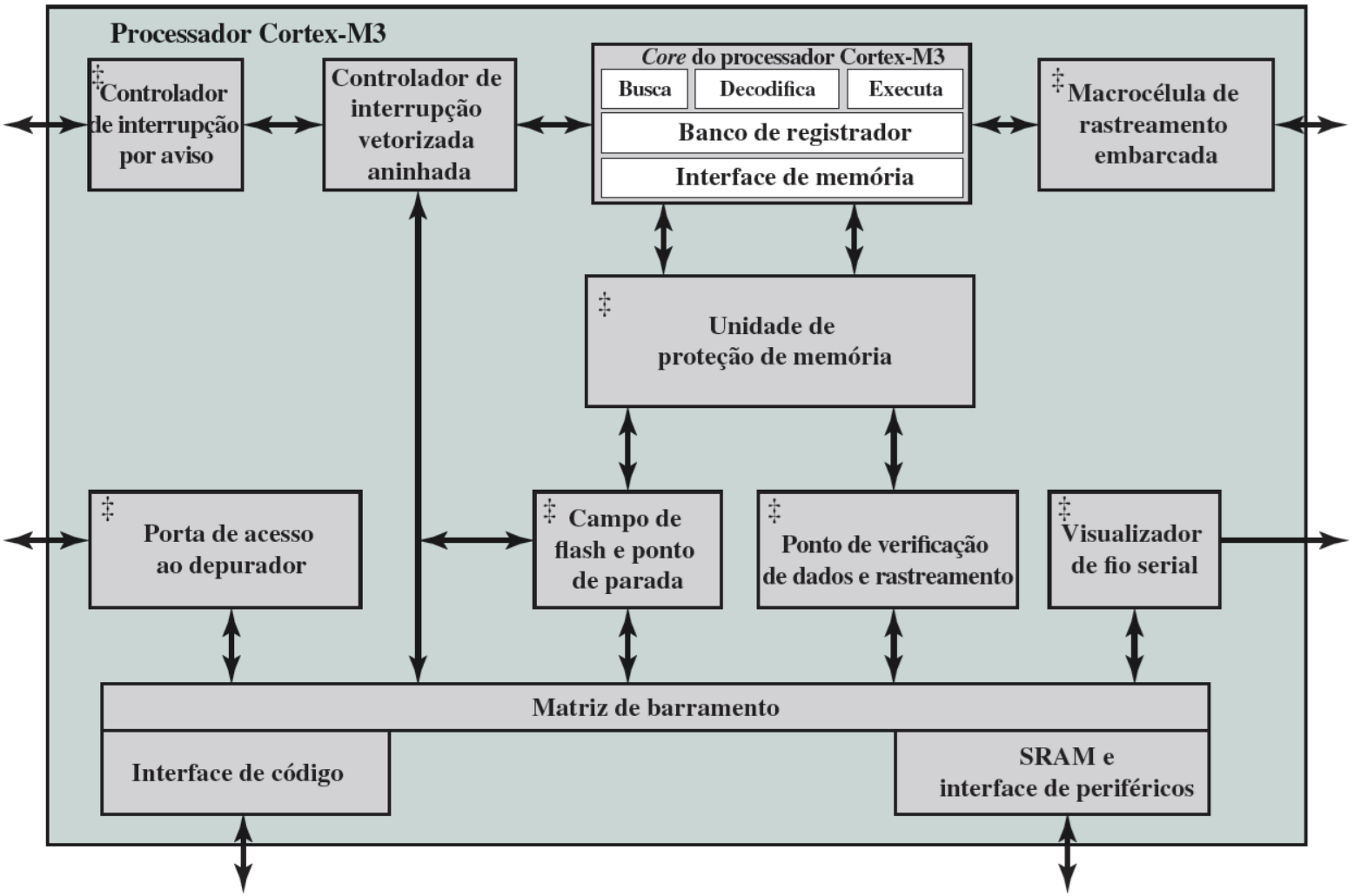
Unidade de execução de inteiros

- E1: o endereço de memória é gerado a partir do registrador base e indexador.
- E2: o endereço é aplicado a arrays da cache.
- E3: no caso de uma carga, os dados são retornados e formatados para serem encaminhados para a unidade ALU ou MUL. No caso de um armazenamento, os dados são formatados e prontos para serem escritos na cache.
- E4: atualiza a cache L2, se necessário.
- E5: resultados das instruções ARM são atualizados no banco de registradores.

Paralelismo em nível de instruções e processadores superescalares

ARM Cortex-M3

- Diagrama de bloco ARM Cortex-M3:



Estrutura de pipeline

- Durante o estágio de busca, uma palavra de 32 bits é buscada por vezes e carregada em um buffer de 3 palavras.

- A palavra de 32 bits consiste em:

- duas instruções Thumb;
- uma instrução de Thumb-2 alinhada por palavra; ou
- a meia-palavra inferior ou superior de uma instrução Thumb-2 alinhada por meia-palavra com:

- — uma instrução Thumb; ou
- — uma meia-palavra inferior/superior e uma instrução Thumb-2 alinhada por meia-palavra.

Estrutura de pipeline

- O estágio de decodificação apresenta três funções-chave:
- Decodificação de instrução e leitura de registrador: decodifica instruções Thumb e Thumb-2.
- Geração de endereço: a unidade de geração de endereço (AGU) cria endereços da memória principal para a unidade de carga/armazenamento.
- Desvio: apresenta desvio baseado no deslocamento imediato em instrução de desvio ou um retorno baseado nos conteúdos do registrador de ligação (registrador R14).

Estrutura de pipeline

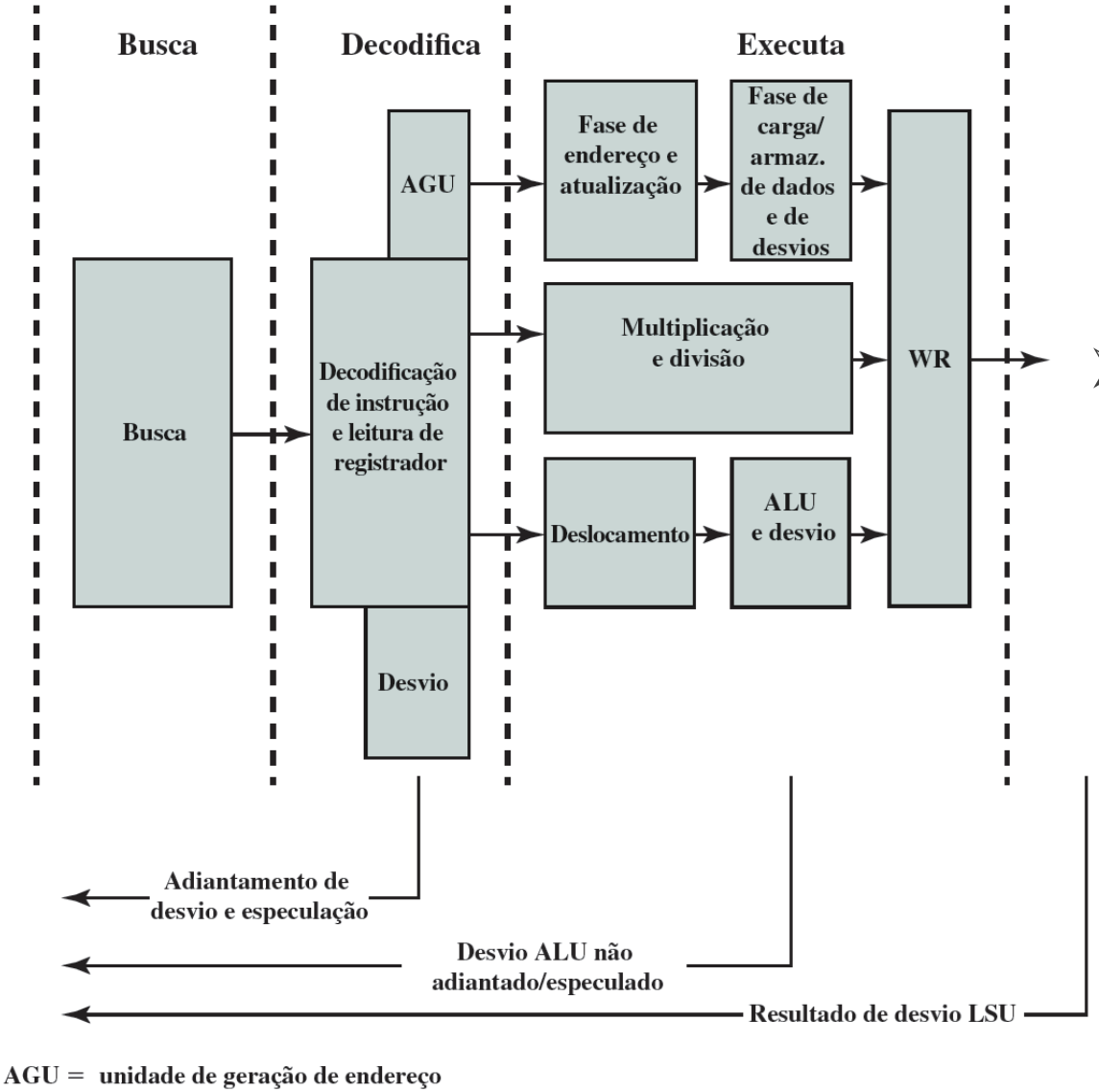
- Para manter o processador tão simples quanto possível, o processador Cortex-M3 usa técnicas simples de adiantamento e especulação de desvio, definidas como segue:

- Adiantamento de desvio: o termo adiantamento se refere a apresentar um endereço de instrução a ser buscado na memória.

- **Especulação de desvio**: para desvios condicionais, o endereço de instrução é apresentado de modo especulativo, de modo que a instrução seja buscada a partir da memória antes que se saiba se a instrução será executada.

Paralelismo em nível de instruções e processadores superescalares

Estrutura de pipeline



AGU = unidade de geração de endereço

- Pipeline de ARM Cortex-M3