Instituto de Matemática
Departamento de Ciência da Computação

Arquitetura de Computadores

Computadores com conjunto reduzido de instruções (RISC)

Prof. Marcos E Barreto

Tópicos

- Características da execução de instruções
- Banco de registradores
- Otimização baseada em compiladores
- Arquitetura RISC
- Pipeline no RISC
- Estudo de casos

- Referências:
  - William Stallings. Arquitetura e organização de computadores. 8 ed. Cap. 13.
  - David Patterson; John Hennessy. Organização e projeto de computadores – interface hardware/software. 3 ed. Apêndice D.

Contextualização

- O projeto de arquiteturas RISC (*reduced instructions set computer*) está baseado no estudo do comportamento da execução dos programas de linguagem de alto nível.
  - Instruções de atribuição são predominantes, o que leva à otimização da movimentação de dados entre memória e registradores.
  - Instruções LOOP e IF também levam à otimizações no mecanismo de controle de sequência para permitir pipeline eficiente.
  - Estudos sobre padrões de referência de operandos sugerem melhorias no desempenho através do uso de um grande número de registradores para armazenamento de operandos.

Contextualização (2)

- Principais características das máquinas RISC:
  - Conjunto de instruções limitado e com formato fixo
  - Número grande de registradores OU uso de compilador que otimize a utilização dos registradores
  - Ênfase na otimização do pipeline de instruções.

- Conjunto de instruções simples leva a um pipeline mais eficiente – menos instruções por operação e elas são mais previsíveis.
- Empregam a técnica de desvio atrasado, na qual as instruções são rearranjadas com outras instruções para melhorar a eficiência do pipeline.

Inovações na arquitetura e organização

- Computadores de programa armazenado (1950)
- Conceito de família
  - IBM System/360 (1964) e DEC PDP-8
  - Separa a arquitetura de uma máquina da sua implementação.
  - Conjunto de computadores com a mesma arquitetura, mas com diferentes características de preço e desempenho.
- Unidade de controle microprogramada
  - Sugerida em 1951 por Wilkes e introduzida no IBM System/360 (1964)
  - Fornece suporte para o conceito de família
  - Facilita a tarefa de projetar e implementar a unidade de controle

Inovações na arquitetura e organização

- Memória cache
  - IBM System/360 Model 85 (1968)
  - Melhoria de desempenho na hierarquia de memória
- RAM de estado sólido
- Pipeline
  - Introdução de paralelismo na natureza essencialmente sequencial de um programa de instruções de máquina
  - Exemplos: pipeline de instruções e processamento vetorial.
- Múltiplos processadores
- Arquiteturas RISC

	Computadores com conjuntos de instruções complexos (CISC)			Computadores com conjuntos de instruções reduzidos (RISC)		Superescalares		
Característica	IBM 370/168	VAX 11/780	Intel 80486	SPARC	MIPS R4000	Power PC	Ultra SPARC	MIPS R10000
Ano de desenvolvimento	1973	1978	1989	1987	1991	1993	1996	1996
Número de instruções	208	303	235	69	94	225		
Tamanho de instrução em bytes	2–6	2–57	1–11	4	4	4	4	4
Modos de endereçamento	4	22	11	1	1	2	1	1
Número de registradores de uso geral	16	16	8	40–520	32	32	40–520	32
Tamanho da memória de controle (Kb)	420	480	246	—	—	—	—	—
Tamanho da cache (KB)	64	64	8	32	128	16–32	32	64

Características da execução de instruções

- Uma das evoluções mais importantes associadas aos computadores é o surgimento das linguagens de programação.
- A evolução trouxe o problema da **diferença semântica**: operações oferecidas nas linguagens de alto nível X operações suportadas pela arquitetura (processador).
  - Ineficiência de execução, tamanho excessivo dos programas e complexidade dos compiladores.
- Arquiteturas para tratamento dessa diferença
  - Grandes conjuntos de instruções
  - Dúzias de modos de endereçamento
  - Instruções da linguagem de alto nível implementadas em hardware (ex. CASE no VAX).

Facilitar a programação de compiladores, melhorar a eficiência da execução, fornecer suporte para linguagens de programação.

Características da execução de instruções

- Aspectos de interesse computacional
  - Operações efetuadas: funções a serem efetuadas pelo processador e sua interação com a memória.
  - Operandos usados: tipos de operandos e frequência de uso determinam a organização da memória para armazená-los e os modos de endereçamento para acessá-los.
  - Sequência de execução: determina a organização e o controle do pipeline.

Operações

- Número médio de instruções de máquina e as referências à memória por tipo de instrução de alto nível (Patterson, 1982).

Frequência relativa de ocorrência

Colunas 2 e 3 X número de instruções de máquina produzidas pelo compilador. Valores normalizados de acordo com peso relativo à ocorrência

Colunas 2 e 3 X número de acessos à memória causados por cada instrução.

	Ocorrência dinâmica		Avaliação das instruções de máquina		Avaliação de referências de memória	
	Pascal	C	Pascal	C	Pascal	C
ATRIBUIÇÃO	45%	38%	13%	13%	14%	15%
LOOP	5%	3%	42%	32%	33%	26%
CHAMADA	15%	12%	31%	33%	44%	45%
IF	29%	43%	11%	21%	7%	13%
GOTO	—	3%	—	—	—	—
OUTROS	6%	1%	3%	1%	2%	1%
	1	2	3	4	5	6	7

Operandos

- Frequência dinâmica da ocorrência de classes de variáveis, independente da arquitetura subjacente.
- Predominância no acesso a variáveis escalares simples e de escopo local (ao procedimento).

	Pascal	**C**	**Média**
Constante inteira	16%	23%	20%
Variável escalar	58%	53%	55%
Array e estrutura	26%	24%	25%

Chamada de procedimentos

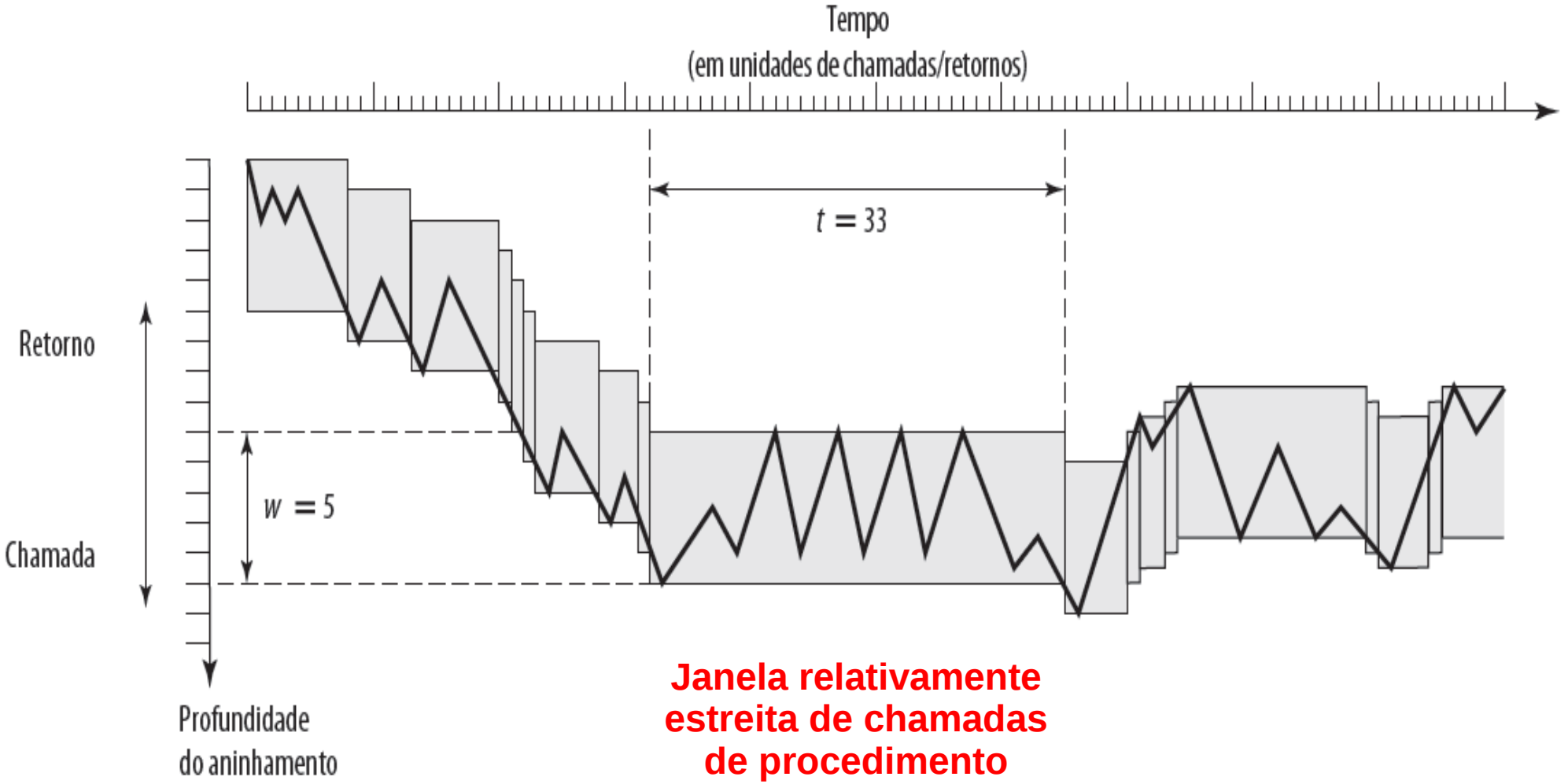
- Chamadas e retornos de procedimentos são as operações que consomem mais tempo em programas compilados de alto nível.
  - Exige eficiência na implementação dessas operações
    - Número de parâmetros e variáveis presentes nos procedimentos
    - Profundidade de aninhamento
- Tanembaum (1978): 98% das chamadas de procedimento usam menos do que 6 argumentos e 92% dos procedimentos usam menos do que 6 variáveis locais.

Berkeley RISC (1983)

% de chamadas de procedimentos executadas com	Compiladores, interpretadores e editores de texto	Pequenos programas não numéricos
> 3 argumentos	0-7%	0-5%
> 5 argumentos	0-3%	0%
> 8 palavras de argumentos e escalares locais	1-20%	0-6%
> 12 palavras de argumentos e escalares locais	1-6%	0-3%

Chamada de procedimentos

- Padrão de chamadas e retorno de procedimentos
  - É raro haver uma grande sequência ininterrupta de chamadas de procedimento seguida pela correspondente sequência de retornos.



Janela relativamente estreita de chamadas de procedimento

Implicações

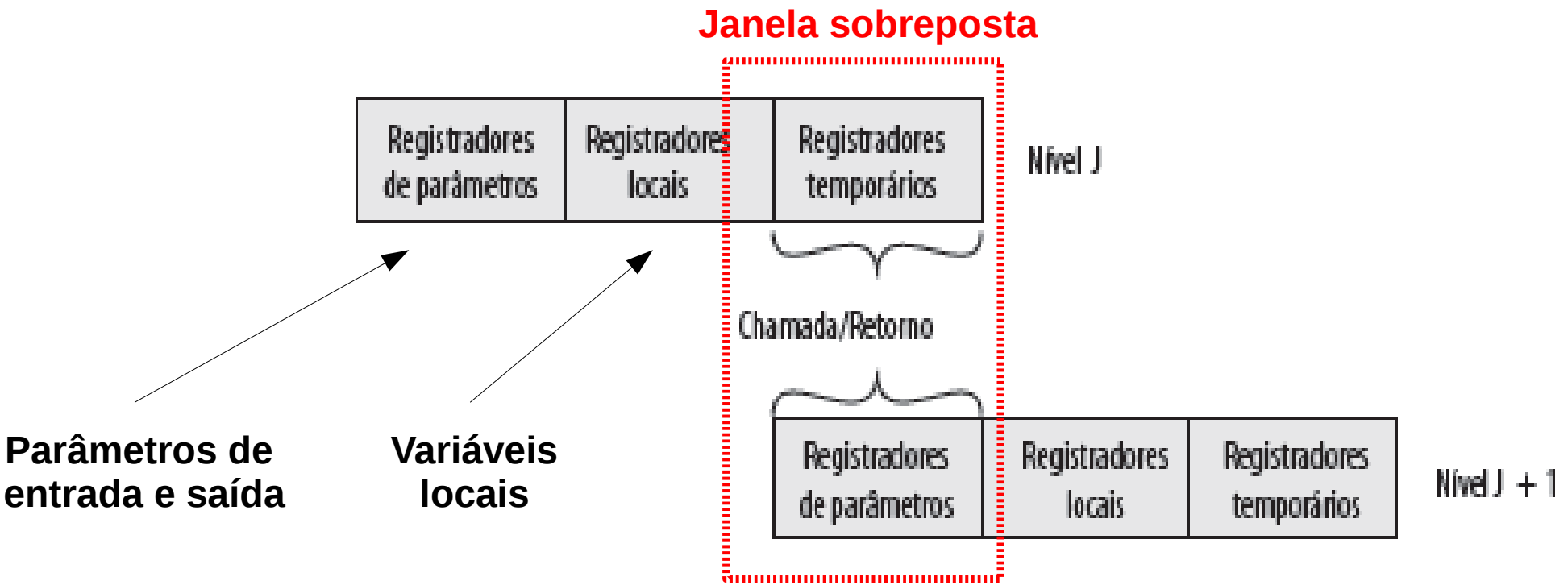
- Criar uma arquitetura com um conjunto de instruções parecido com as linguagens de alto nível **não** é uma estratégia eficaz.
- Melhor suporte para linguagens de alto nível é dado otimizando recursos mais usados e mais demorados.
- Elementos que caracterizam as arquiteturas RISC:
  - Uso de um grande número de registradores ou de um compilador para otimizar o uso de registradores
    - Devido à grande proporção de instruções de atribuição, aliado com a localidade e predominância de variáveis escalares.
    - Objetivo é melhorar o desempenho através da redução de acessos à memória, passando a acessar os registradores.
  - Atenção especial ao projeto de pipelines de instruções
    - Devido ao grande número de instruções de desvio e chamadas de procedimento.
  - Uso de um conjunto reduzido de instruções

Grande número de registradores

- Grande proporção de instruções de atribuição nas linguagens de alto nível, do tipo A ← B.
- Maior parte dos acessos é para variáveis escalares locais => forte dependência de armazenamento em registradores.
- O banco de registradores é fisicamente pequeno, geralmente presente no mesmo chip da ULA e da unidade de controle. Emprega endereços bem menores do que os endereços usados em memória cache e principal.
- Duas estratégias:
  - Baseada em software (compilador) para otimizar o uso de registradores => alocar registradores para aquelas variáveis usadas com maior frequência num dado momento.
  - Baseada em hardware, através do uso de mais registradores para armazenamento de variáveis.

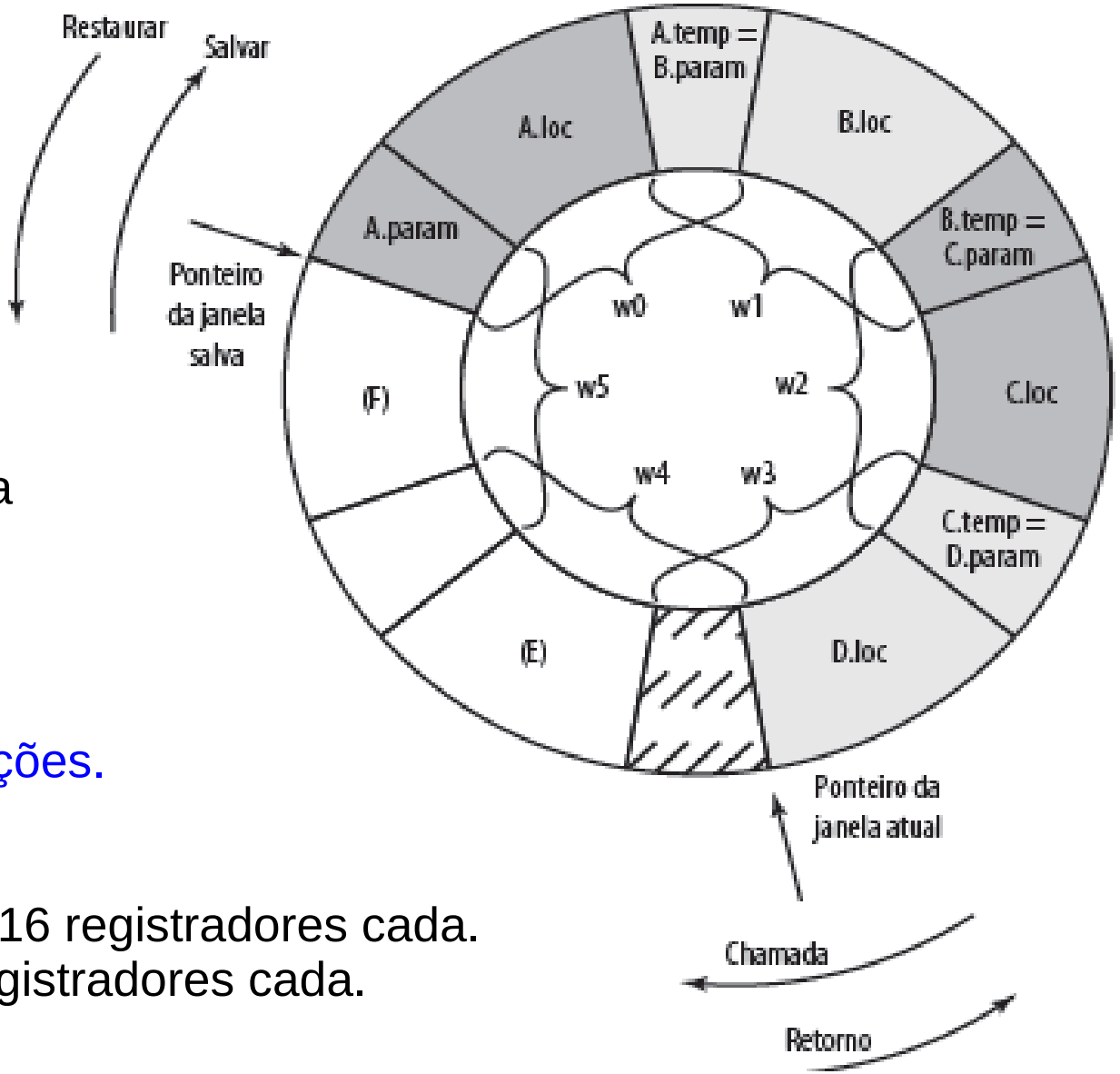
Janela de registradores

- Conjunto pequeno de registradores, cada um atribuído a um procedimento.
- Janelas para procedimentos adjacentes são sobrepostas para permitir passagem de parâmetros.



Janela de registradores

- Buffer circular de janelas sobrepostas
  - Sun SPARC e Intel Itanium 64



- Buffer de 6 janelas
- 4 janelas ativas (A, B, C e D)
- D pode chamar E, ativando a quinta janela
- Se E chamar F, vai causar uma Interrupção, visto que a janela sobreporia a janela A.
Nesse caso, a janela A é salva.

N janelas armazenam N-1 ativações.

Na prática:
- RISC Berkeley: 8 janelas com 16 registradores cada.
- Pyramid: 16 janelas com 32 registradores cada.

Variáveis globais

- O esquema de janelas sobrepostas não serve para o armazenamento de variáveis globais (acessadas por dois ou mais procedimentos).
- Duas abordagens:
  - Variáveis globais são alocadas em posições “globais” de memória, referenciadas por todas as instruções que acessam tais variáveis. Pode ser ineficiente, dependendo da frequência de acesso.
  - Uso de registradores globais, numerados, por exemplo, de 0 a 7. Os endereços de 8 para cima poderiam ser usados para determinar o deslocamento dentro da janela. Precisa que o compilador determine quais variáveis globais serão alocadas em tais registradores. Além disso, impõe sobrecarga ao hardware para tratar a divisão de endereçamento.

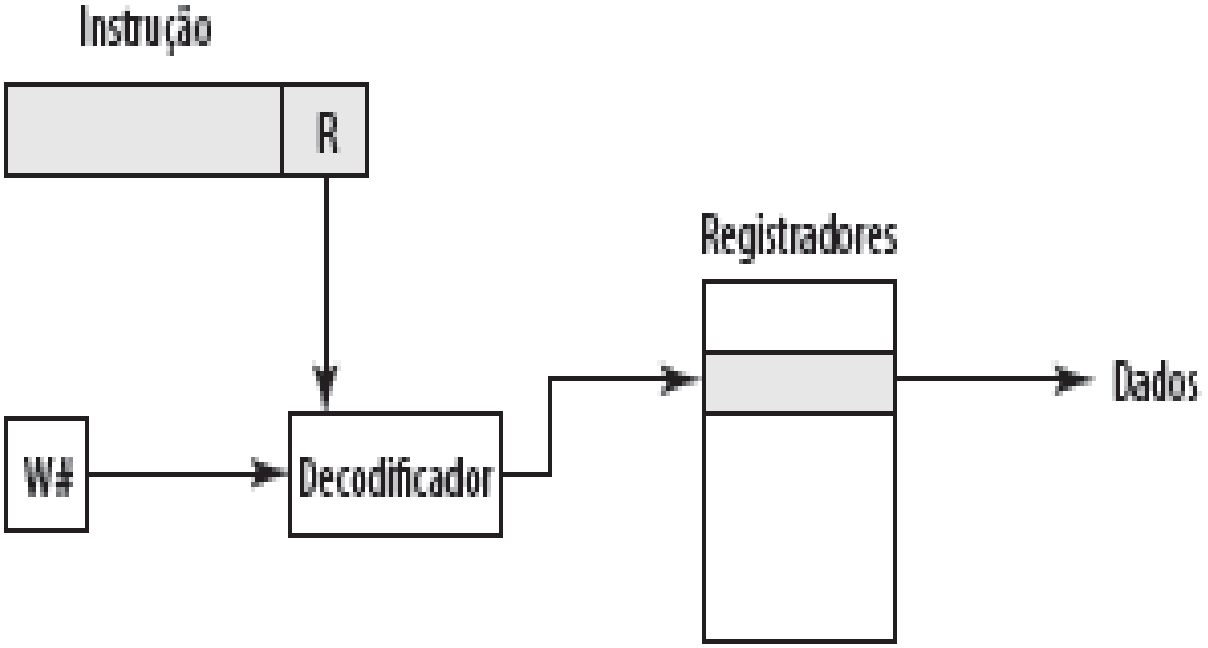
Grande banco de registradores X cache

- A janela de registradores age como um pequeno buffer para armazenamento das variáveis usadas com mais frequência => semelhante à memória cache.
- Questão: seria mais simples usar uma cache ou um banco de registradores pequeno e tradicional?

Grandes arquivos de registradores	Cache
Todas variáveis locais escalares	Variáveis locais recentemente usadas
Variáveis individuais	Blocos de memória
Variáveis globais assinaladas pelo compilador	Variáveis globais recentemente usadas
Salvar/restaurar baseados na profundidade de aninhamento do procedimento	Salvar/restaurar baseado em algoritmos de atualização da cache
Endereçamento de registrador	Endereço de memória

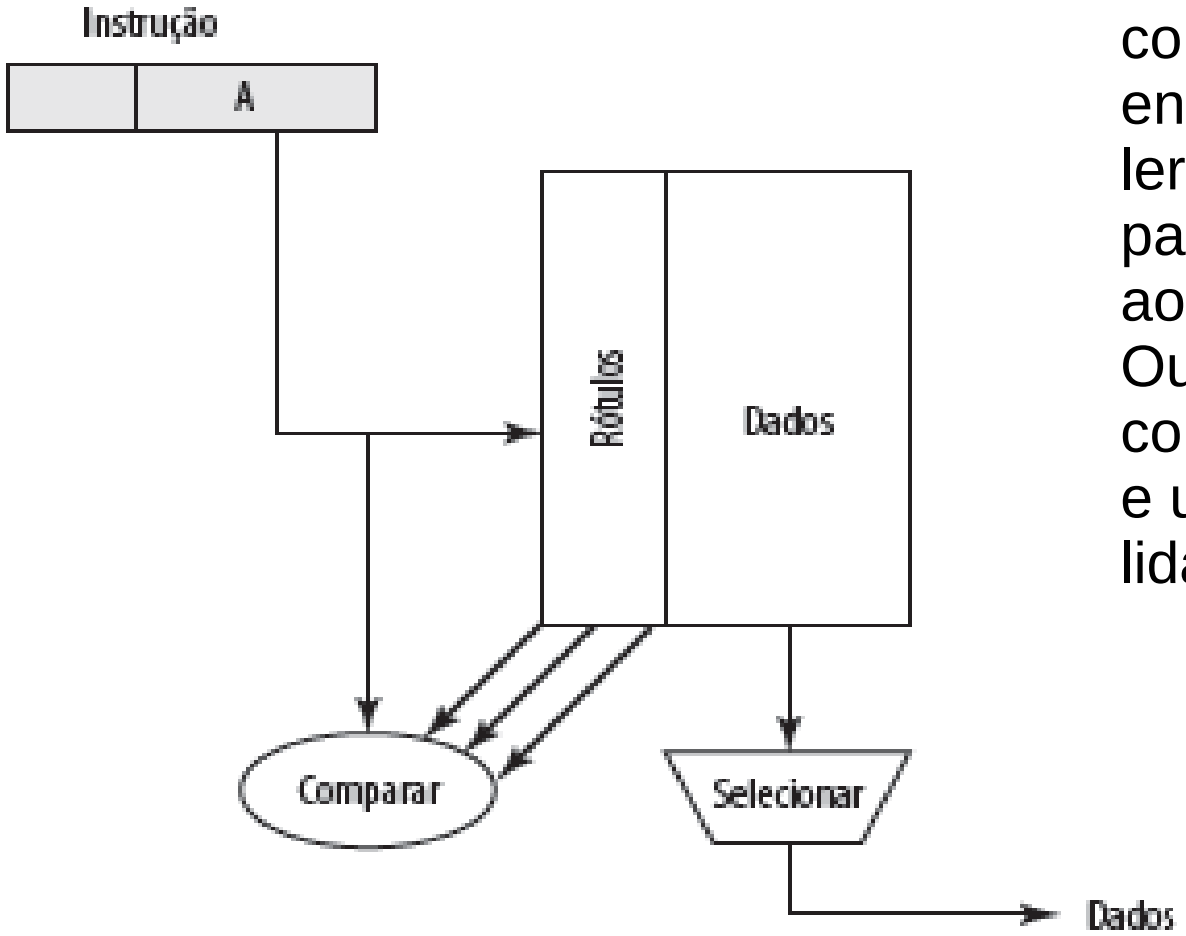
Grande banco de registradores X cache

- Para referenciar um escalar local num arquivo de registradores baseado em janelas, um número de registrador virtual (R) e um número de janela (W) são usados.
- Eles podem passar por um decodificar para gerar o endereço do registrador físico.



Grande banco de registradores X cache

- Para referenciar uma posição na cache, um endereço de memória de tamanho completo deve ser gerado.
- A complexidade depende do modo de endereçamento.



- Em cache associativa por conjunto, uma parte do endereço é usada para ler um número de palavras e rótulos iguais ao tamanho do conjunto. Outra parte do endereço é comparada com as marcações e uma das palavras que foram lidas é selecionada.

Otimização baseada em compiladores

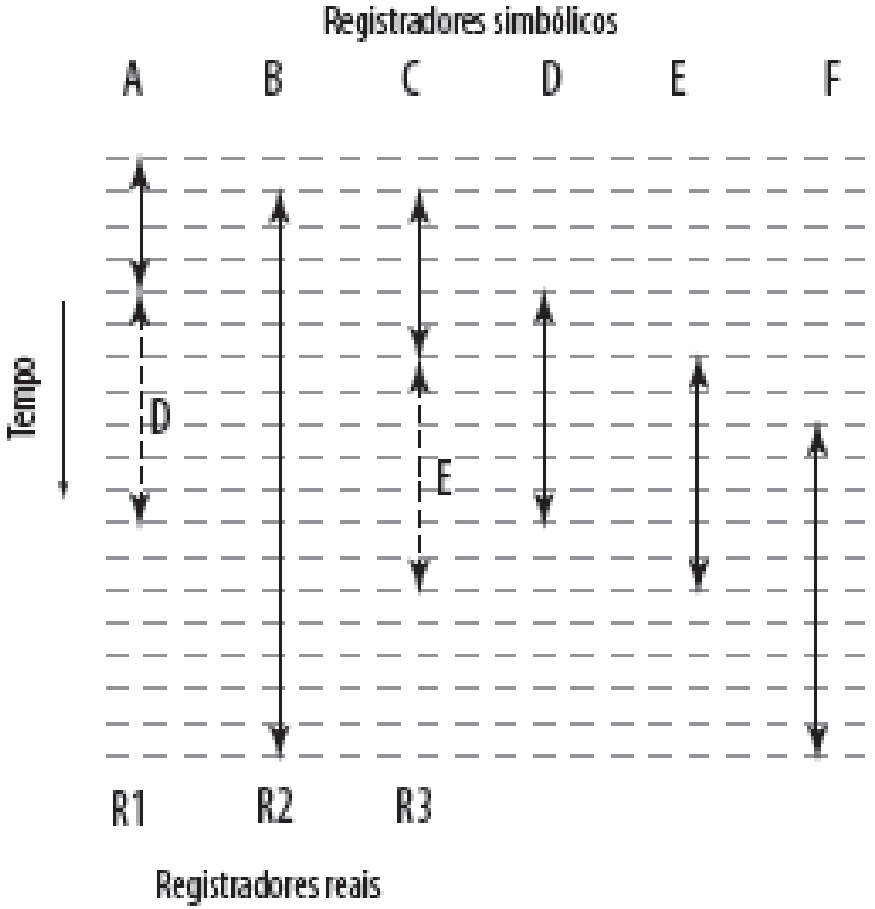
- O objetivo do compilador é guardar os operandos em registradores durante o máximo de instruções possível, reduzindo o número de acessos à memória.
- A abordagem adotada utiliza-se de registradores simbólicos (virtuais), os quais são usadas para mapear as variáveis do programa. Cada registrador simbólico é, por sua vez, mapeado para um registrador real.
- Quantidade de registradores simbólicos X quantidade de registradores reais
  - As variáveis que sobram (ou seja, que não são alocadas em registradores reais) são alocadas em memória.

Otimização baseada em compiladores

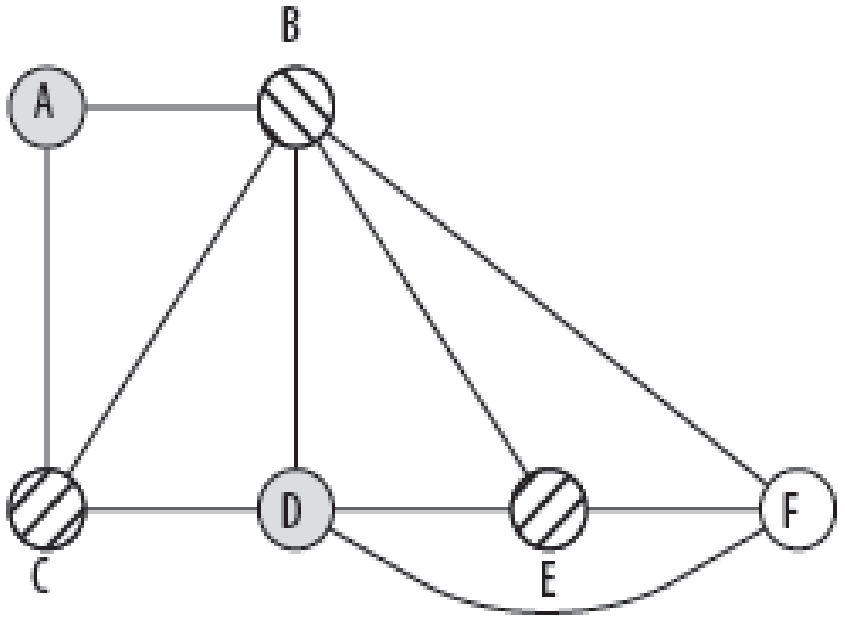
- Questão: quais variáveis devem ser atribuídas para registradores em uma determinada parte do programa?
- Solução baseada em coloração de grafos
  - *Dado um grafo que consiste de nós e arestas, atribuir cores para os nós de tal forma que nós adjacentes tenham cores diferentes e que seja usada a menor quantidade possível de cores.*
  - Adaptação
    - Programa é analisado para montar um grafo de interferência entre registradores.
    - Os nós do grafo são registradores simbólicos.
    - Se dois registradores simbólicos estão “ativos” durante o mesmo fragmento de programa, então eles são unidos por uma aresta.
    - Uma tentativa é feita para colorir o grafo com N cores, onde N é o número de registradores.
    - Nós que compartilham a mesma cor podem ser atribuídos para o mesmo registrador. Nós que não puderem ser coloridos, devem ser alocados em memória.

Otimização baseada em compiladores

- 6 registradores simbólicos => 3 registradores reais
- Registradores A e D são mapeados para R1
- Registradores C e E são mapeados para R3



(a) Sequência de tempo do uso ativo de registradores



(b) Grafo de interferência de registradores

Por que CISC?

- Objetivo: simplificar os compiladores e melhorar o desempenho.
- Estratégia: projetar máquinas que fornecem melhor suporte para linguagens de alto nível.
  - Simplificar o compilador => gerar uma sequência de instruções de máquina para cada instrução de alto nível => ideal é que as instruções se assemelhem.
    => na prática, instruções complexas são difíceis de serem exploradas pelo compilador => quando usar?
  - Expectativa de que um CISC gere programas menores e mais rápidos.
    - Programas menores => menos memória, menos instrução e menor ocupação de páginas.
    - Programas mais rápidos => expectativa de que uma instrução complexa de alto nível seja executada mais rápida sendo uma única instrução complexa de baixo nível.

Por que CISC?

- Um programa CISC pode ser mais curto (menos instruções), mas ocupa mais bits de memória.
  - Tamanho de programas C compilados em várias arquiteturas, incluindo RISC I e máquinas CISC.
  - Pouca ou nenhuma economia ao se usar CISC no lugar de RISC.
  - VAX (conjunto de instruções muito mais complexo) tem pouca economia referente ao PDP-11.

	Patterson e Sequin (1982) 11 programas C	**Katevenis (1983) 12 programas C**	**Heath (1984) 5 programas C**
RISC I	1,0	1,0	1,0
VAX-11/780	0,8	0,67	
M68000	0,9		0,9
Z8002	1,2		1,12
PDP-11/70	0,9	0,71	

Características de arquiteturas RISC

- Várias abordagens RISC, mas consenso aborda:
  - Uma instrução de máquina por ciclo de máquina
    - Ciclo de máquina: busca de dois operandos + executar operação na ULA + escrita de resultados.
    - Tendência de que instruções RISC sejam mais menos complicadas do que instruções CISC.
    - Pouca ou nenhuma necessidade de microcódigo => instruções podem ser embutidas no hardware.

Características de arquiteturas RISC

- Operações registrador-para-registrador
  - Maioria das operações sendo registrador-para-registrador, com apenas operações simples de LOAD e STORE acessando a memória => simplifica o conjunto de instruções e, portanto, a unidade de controle.
  - Máquinas CISC também fornecem tais instruções, mas também fornecem instruções memória-memória e memória-registrador.

A = B + C

8	16	16	16
ADD	B	C	A

memória para memória
I = 56, D= 96, M = 152

8	4	16
LOAD	RB	B
LOAD	RC	B
ADD	RA	RB RC
STORE	RA	A

registrador para memória
I = 104, D= 96, M = 200

8	16	16	16
ADD	B	C	A
ADD	A	C	B
SUB	B	D	D

memória para memória
I = 168, D= 288, M = 456

8	4	4	4
ADD	RA	RB	RC
ADD	RB	RA	RC
SUB	RD	RD	RB

registrador para memória
I = 60, D= 0, M = 60

A = B + C
B = A + C
D = D – B

I = # bytes ocupados por instruções
D = # bytes ocupados por dados
M = tráfego de memória (I + D)

Características de arquiteturas RISC

- Modos de endereçamento simples
  - Quase todas as instruções RISC usam endereçamento de registradores simples.
  - Outros modos, tais como deslocamento relativo ao PC, podem ser incluídos.
  - Modos mais complexos são sintetizados por software.

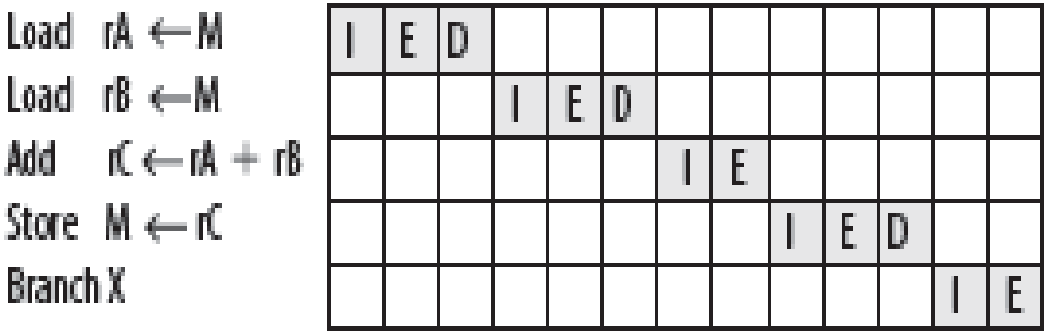
- Formato de instruções simples
  - Um ou poucos formatos diferentes são usados.
  - Tamanho da instrução é fixo e ajustado dentro do limite da palavra.
  - A posição dos campos é fixa, o que permite a decodificação do opcode e o acesso a registradores simultaneamente.

Características CISC x RISC

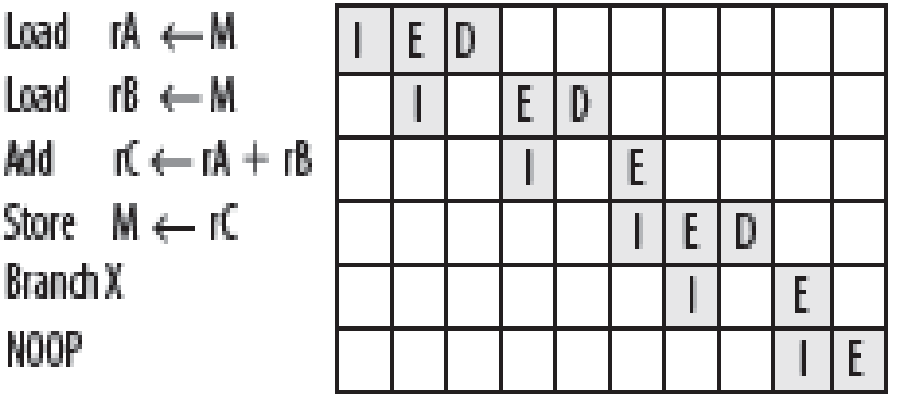
- Projetos recentes adotam uma abordagem híbrida
  - Arquiteturas RISC (ex. PowerPC) não são mais “puramente RISC” e arquiteturas CISC (ex. Pentium II e demais) incorporam características RISC.
  - Itens típicos de uma arquitetura RISC:
    - Tamanho único de instrução (normalmente, 4 bytes)
    - Número menor de modos de endereçamento => difícil parametrizar!!
    - Nenhum endereçamento indireto
    - Nenhuma instrução que combine leitura/escrita com aritmética
    - Não mais do que um operando endereçado em memória por instrução
    - Sem alinhamento arbitrário de dados para operações de leitura/escrita
    - Número máximo de usos de unidade de gerenciamento de memória (MMU) para um endereço de dados em uma instrução
    - 5 ou mais bits para endereçamento de registradores inteiros => no mínimo, 32 registradores
    - 4 ou mais bits para endereçamento de registradores de ponto flutuante => no mínimo, 16 registradores.

Pipeline no RISC

- Pipeline com instruções regulares
  - A maioria das instruções é registrador-para-registrador.
  - Um ciclo de instruções tem dois estágios:
    - I: busca da instrução e E: execução (operação na ULA com entrada e saída de registradores)
  - Operações LOAD/STORE têm três estágios
    - I: busca da instrução; E: execução (calcular o endereço de memória) e D: memória (operação registrador-memória ou memória-registrador)



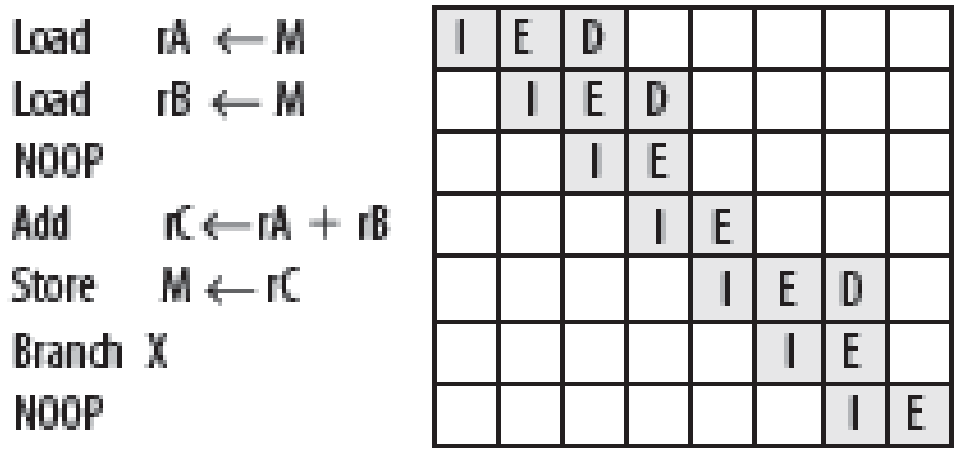
(a) Execução sequencial



(b) Tempo do pipeline de dois estágios

Pipeline no RISC

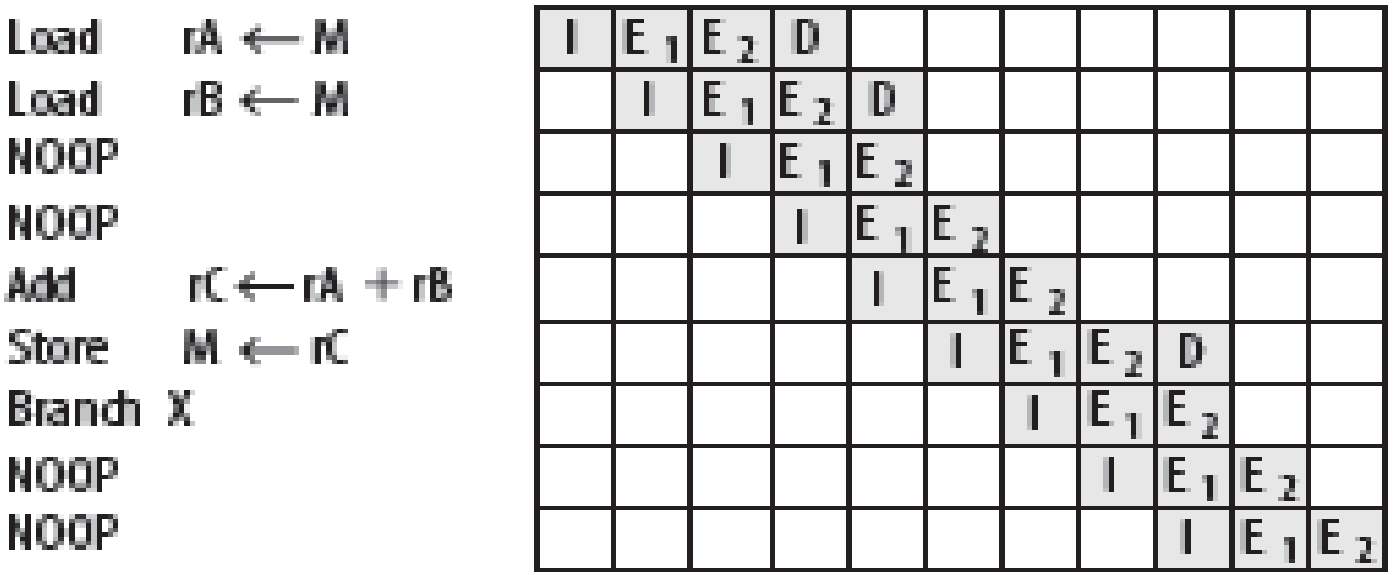
- Pipeline com instruções regulares
  - Melhoria pode ser obtida permitindo dois acessos à memória por estágio.
  - Problemas referentes à dependência de dados
    - Uso de NOOP para introduzir espera nos estágios enquanto a dependência existir.



(c) Tempo do pipeline de três estágios

Pipeline no RISC

- Pipeline com instruções regulares
  - O estágio E geralmente é mais demorado, pois envolve operações na ULA.
  - Pode ser dividido em dois subestágios:
    - E1: leitura do banco de registradores
    - E2: operação da ULA e escrita em registrador



(d) Tempo do pipeline de quatro estágios

Pipeline no RISC

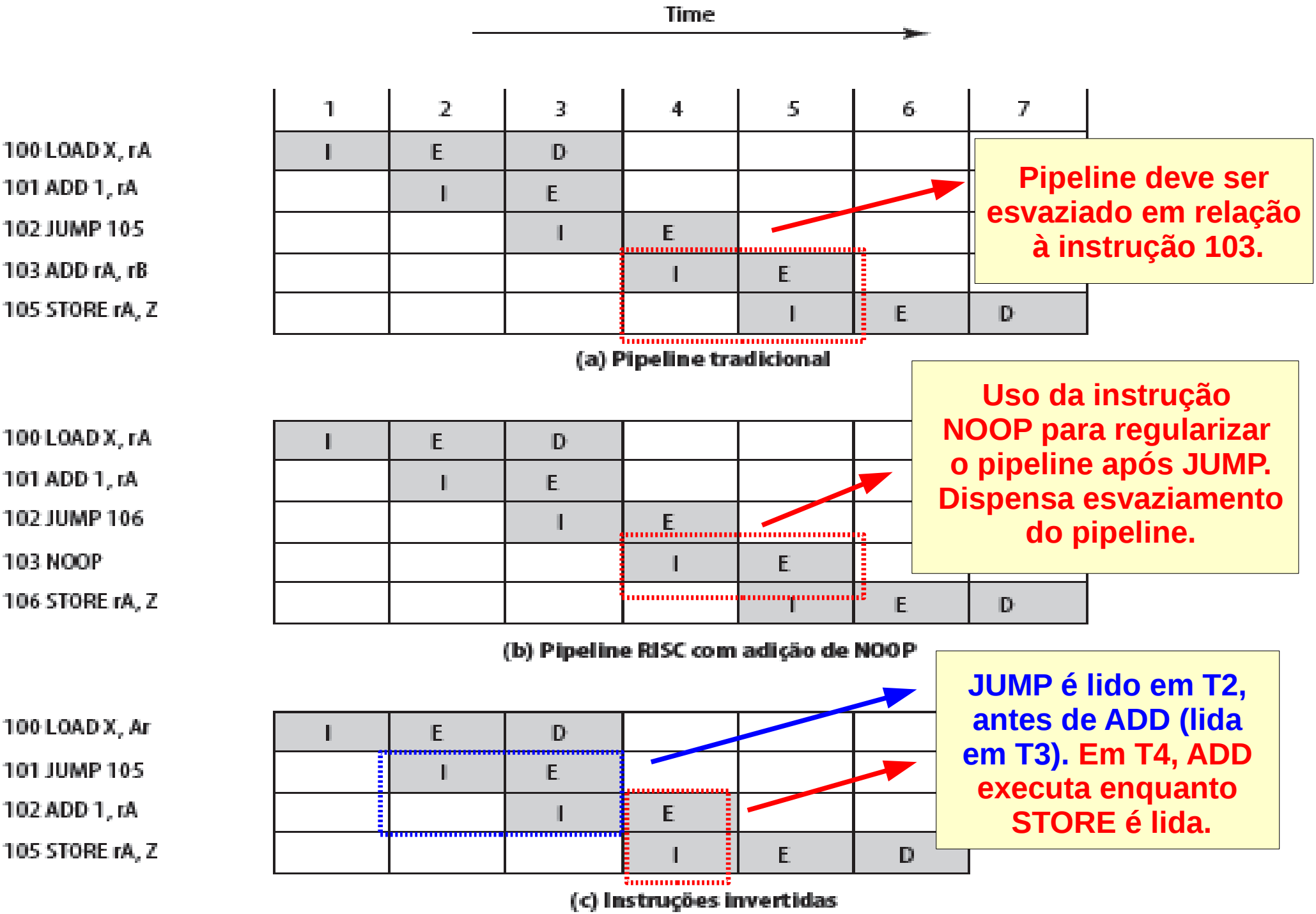
- Otimização de pipeline
  - A natureza regular e simples das instruções RISC favorece o emprego de pipelines eficientes.
  - Problema está na dependência de dados e nas instruções de devio.
  - Técnicas usadas:
    - Desvio atrasado
    - Leitura atrasada
    - Laço desenrolado

Pipeline no RISC

- Otimização de pipeline - desvio atrasado
  - Faz uso de um desvio que não toma efeito até depois da execução da instrução seguinte
  - Posição da instrução imediatamente após o desvio é chamada *delay slot*.

Endereço	Desvio normal	Desvio atrasado	Desvio atrasado otimizado
100	LOAD X, rA	LOAD X, rA	LOAD X, rA
101	ADD 1, rA	ADD 1, rA	JUMP 105
102	JUMP 105	JUMP 106	ADD 1, rA
103	ADD rA, rB	NOOP	ADD rA, rB
104	SUB rC, rB	ADD rA, rB	SUB rC, rB
105	STORE rA, Z	SUB rC, rB	STORE rA, Z
106		STORE rA, Z	

- Otimização de pipeline - desvio atrasado



Pipeline no RISC

- Otimização de pipeline – leitura atrasada
  - Tática usada em instruções LOAD (delay load).
  - Em instruções LOAD, o registrador que é o alvo da leitura é bloqueado pelo processador.
  - O processador, então, continua a execução até que alcance uma instrução que precise deste registrador, ponto no qual ele fica ocioso até que a leitura esteja completada.
  - Se houver o rearranjo das instruções para que o trabalho útil possa ser feito enquanto ocorre leitura dentro do pipeline, a eficiência será aumentada.

Pipeline no RISC

- Otimização de pipeline – laço desenrolado
  - Técnica que replica o corpo de um laço em um número de vezes chamado fator de desenrolar (*u*), e faz a iteração pelo passo *u* em vez de passo 1.

```
LAÇO ORIGINAL
do i=2, n-1
a[i]= a[i] + a[i-1] * a[i+1]
end do
```

```
LAÇO DESENROLADO DUAS VEZES
do i=2, n-2, 2
a[i]= a[i] + a[i-1] * a[i+1]
a[i+1]= a[i+1] + a[i] * a[i+2]
end do
if (mod(n-2, 2) = i) then
a[n-1]= a[n-1] + a[n-2] * a[n]
end if
```

A **sobrecarga do laço é reduzida** pela metade porque duas iterações são executadas antes do teste.

Aumenta o paralelismo de instruções porque a segunda atribuição pode ser realizada enquanto os resultados da primeira estão sendo armazenados, e as variáveis do laço sendo atualizadas.

Se os elementos do vetor são atribuídos aos registradores, o **posicionamento dos registradores vai melhorar** porque a[i] e a[i+1] são usados duas vezes no corpo do laço, reduzindo o número de leituras por iteração.

Estudo de casos

- Sugestão de leitura do Apêndice D do livro do Patterson e Hennessy sobre diversas arquiteturas RISC.

  Uma Visão Geral das Arquiteturas RISCs para Computadores Desktop, Servidores e Embutidos

- Material disponível na página da disciplina no Moodle.