Estrutura e função do processador

Estrutura e Função do Processador

Tópicos a serem abordados

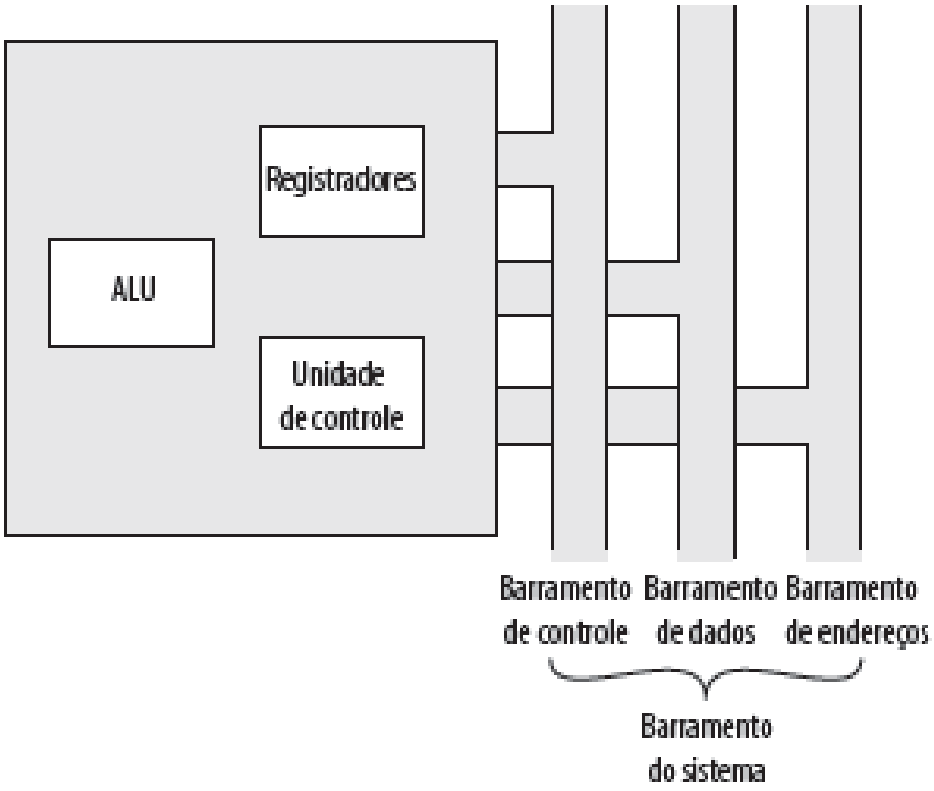
- Resumo sobre a organização do processador
- Registradores
- Ciclo da instrução
- Pipeline de instruções
- Aspectos das organizações x86 e ARM

Estrutura da CPU

- CPU precisa:
  - Buscar instruções (registrador, cache, memória principal)
  - Interpretar instruções (decodificação da instrução)
  - Obter dados (leitura de dados da memória ou de um dispositivo de E/S)
  - Processar dados (alguma operação aritmética ou lógica com os dados)
  - Gravar dados (para a memória ou em um módulo de E/S)

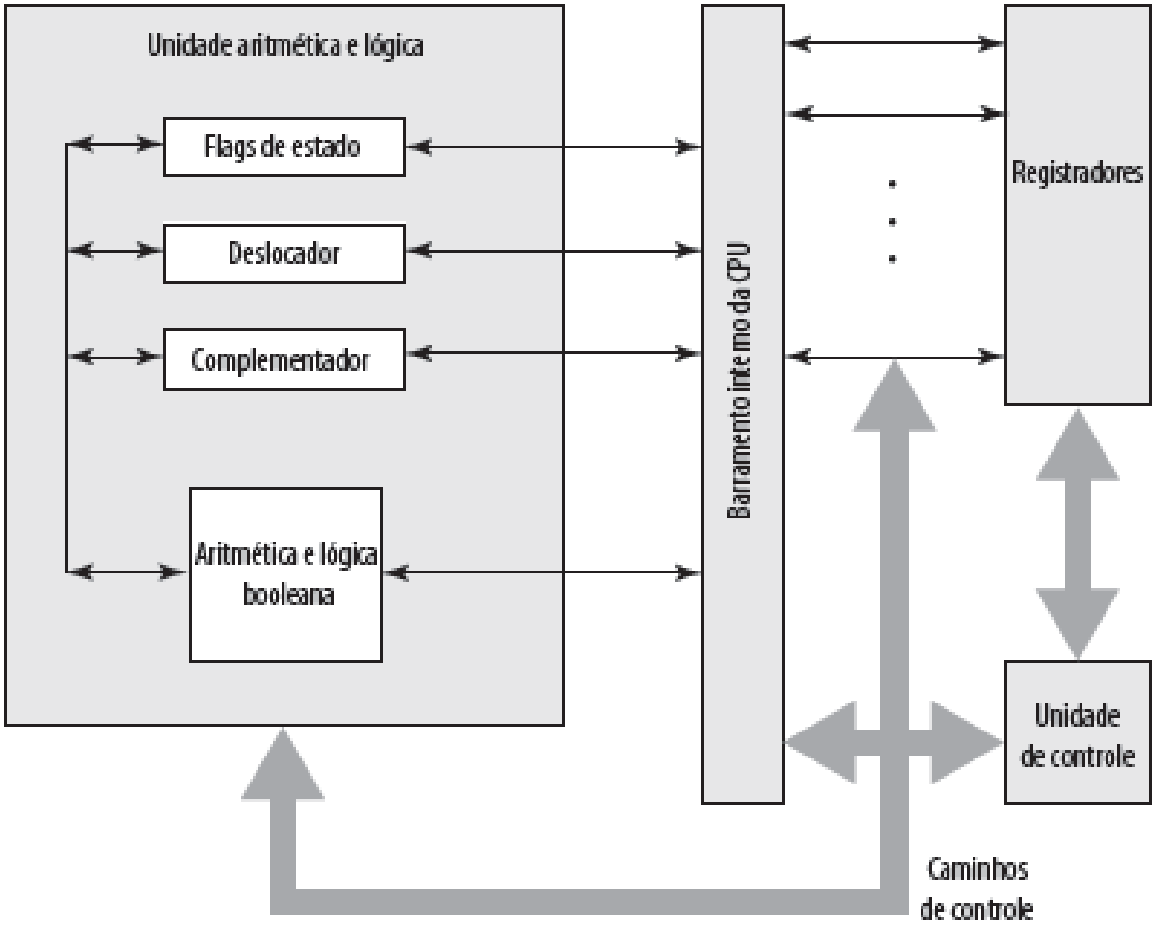
CPU com barramento de sistemas

Visão simplificada



Estrutura interna da CPU

Visão mais detalhada



Registradores

Organização dos Registradores

- CPU precisa ter algum espaço de trabalho (armazenamento temporário)
- Registradores
- Número e função variam entre projetos de processadores (uma das principais decisões de projeto)
- Alto nível de hierarquia de memória (acima da memória principal e da cache)

Registradores

Organização dos Registradores

- Registradores visíveis ao usuário: possibilitam que o programador de linguagem de máquina ou assembly minimize as referências à memória, pela otimização do uso de registradores.

- Registradores de controle e estado: usados pela unidade de controle para controlar a operação do processador e por programas privilegiados do Sistema Operacional (SO) para controlar a execução de programas.

Registradores visíveis ao usuário

- Registrador visível ao usuário é aquele que pode ser referenciado pelos recursos da linguagem de máquina que o processador executa. Dividem-se em:
  - Uso geral
  - Dados
  - Endereços
  - Códigos condicionais

Registradores de uso geral

Organização dos Registradores

- Podem ser de propósito geral verdadeiro
- Podem ser restritos (exs. registradores dedicados para ponto flutuante, operações de pilha)
- Podem ser usados para dados ou endereçamento.
- Dados (somente para armazenar dados, não pode ser utilizado para calcular o endereço de um operando)
  - Acumulador
- Endereçamento (de uso geral ou dedicados para um modo de endereçamento em particular)
  - Segmento (exs. ponteiros de segmentos, registradores de índice, ponteiros de pilha).

Registradores de uso geral

Questão de projeto: registradores somente de propósito geral ou especializados?

- Torne-os de uso geral:
  - Aumente flexibilidade e opções do programador
  - Aumente tamanho de instrução e complexidade

- Torne-os especializados:
  - Instruções menores (mais rápidas)
  - Menos flexibilidade

Quantos registradores de uso geral?

Questão de projeto: número de registradores.

- Entre 8 −32
- Menos= mais referências à memória
- Mais = não reduz as referências à memória e ocupa espaço no processador

De qual tamanho?

Questão de projeto: tamanho dos registradores.

- Grande o suficiente para manter endereço completo
- Grande o suficiente para manter palavra completa
- Normalmente, é possível combinar dois registradores de dados
  - Programação C
  - Double int a
  - Long int a

Registradores de código condicional

- Conjuntos de bits individuais
  - Ex. resultado da última operação foi zero
- Podem ser lidos (implicitamente) por programas
  - Ex. Jump if zero
- Não podem (normalmente) ser alterados por programas

Registradores de código condicional

Organização dos Registradores

Códigos condicionais

Vantagens	Desvantagens
1. Como os códigos condicionais são definidos por instruções normais aritméticas ou de movimentação de dados, eles devem reduzir o número de instruções de comparação e teste (COMPARE, TEST) necessárias.	1. Códigos condicionais acrescentam complexidade, tanto para hardware como para software. Os bits dos códigos condicionais são frequentemente modificados de maneiras diferentes por instruções diferentes, tornando a vida do microprogramador e do projetista de compiladores mais difícil.
2. Instruções condicionais, como BRANCH, são simplificadas em relação a instruções compostas como TEST AND BRANCH.	2. Códigos condicionais são irregulares; normalmente eles não fazem parte do caminho principal de dados e, por isso, requerem conexões extras de hardware.
3. Códigos condicionais facilitam desvios múltiplos. Por exemplo, uma instrução TEST pode ser seguida de dois desvios, um para menor ou igual a zero e outro para maior que zero.	3. Frequentemente, máquinas com códigos condicionais precisam adicionar instruções especiais que não usam códigos condicionais para situações especiais de qualquer forma, como verificação de bits, controle de laços e operações atômicas de semáforos.
	4. Em uma implementação de pipeline, códigos condicionais requerem sincronização especial para evitar conflitos.

Registradores de controle e estado

- Contador de programa (PC): contém o endereço de uma instrução a ser lida.
- Registrador da instrução (IR): contém a instrução lida mais recentemente.
- Registrador de endereço de memória (MAR): contém o endereço de uma posição de memória.
- Registrador de buffer de memória (MBR): contém uma palavra de dados par ser escrita na memória ou a palavra lida mais recentemente.

Palavra de estado do programa (PSW)

- Um conjunto de bits
- Inclui *flags* (códigos condicionais)
- Sinal - contém o bit de sinal do resultado da última operação aritmética
- Zero - marcado quando o resultado é zero
- *Carry - marcado se uma operação resultou em transportar (adição) para empréstimo (subtração) de um bit de ordem maior.*
- Igual – marcado se uma comparação lógica resultou em igualdade
- *Overflow – usado para indicar sobrecarga aritmética*
- Habilitar/desabilitar interrupção
- Supervisor – indica se o processador esta executando no modo supervisor ou usuário.

Modo supervisor

- Intel anel zero
  - Há quatro níveis de privilégio, numerados de 0 (o de maior privilégio) a 3 (o de menor privilégio), e três recursos principais sendo protegidos: memória, portas I/O, e a possibilidade de executar certas instruções da máquina.
- Modo kernel
- Permite execução de instruções privilegiadas
- Usado pelo sistema operacional
- Não disponível aos programas do usuário

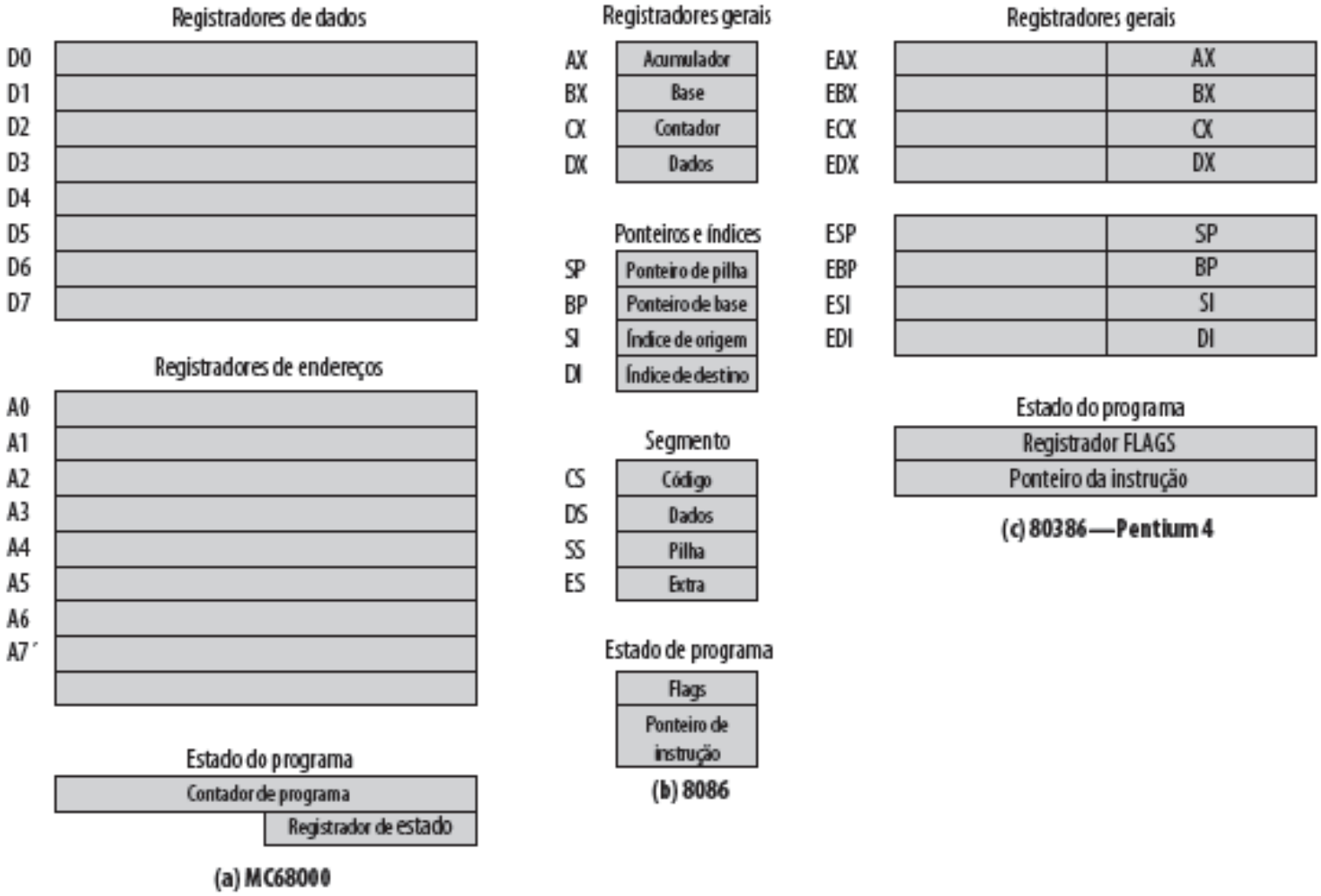
Outros registradores

- Podem ter registradores apontando para:
  - Blocos de controle de processo (S/O)
  - Vetores de interrupção (S/O)

- Projeto da CPU e projeto do sistema operacional são bastante interligados.

Exemplo de organizações de registradores

Organização dos Registradores



Ciclo de instrução

- Revisão

- Ciclo de instrução (3 estágios)

1. Buscar: lê a próxima instrução da memória para dentro do processador.
2. Executar: interpreta *opcode* e efetua a operação indicada.
3. Interromper: se as interrupções estão habilitadas e uma interrupção ocorre, salva o estado do processo atual a atender a interrupção.

Ciclo indireto

- Pode exigir acesso à memória para obter operandos
- Endereçamento indireto requer mais acessos à memória
- Pode ser imaginado como subciclo de instrução adicional

Ciclo de instrução com indireção

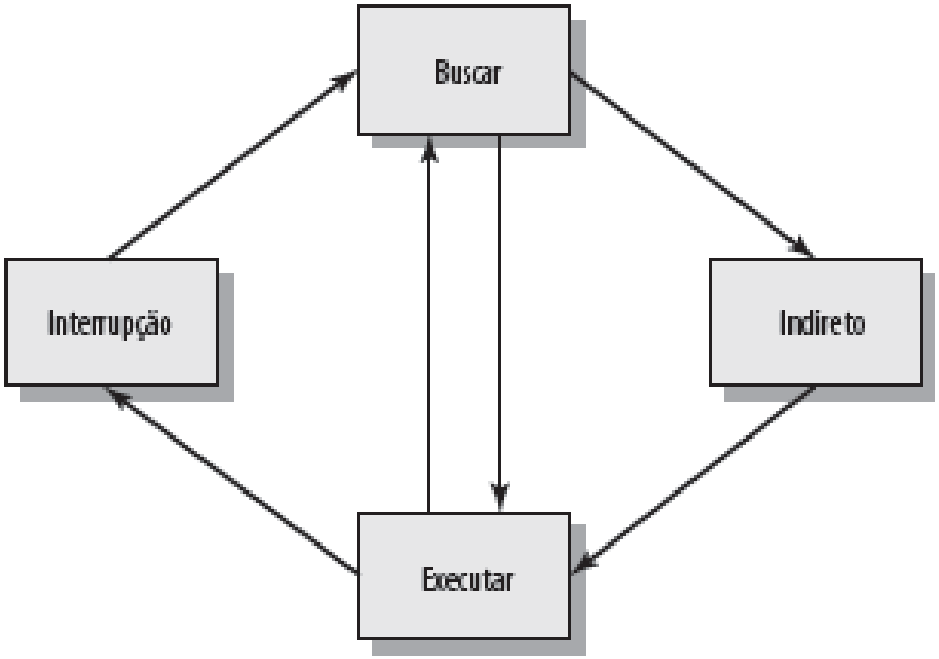
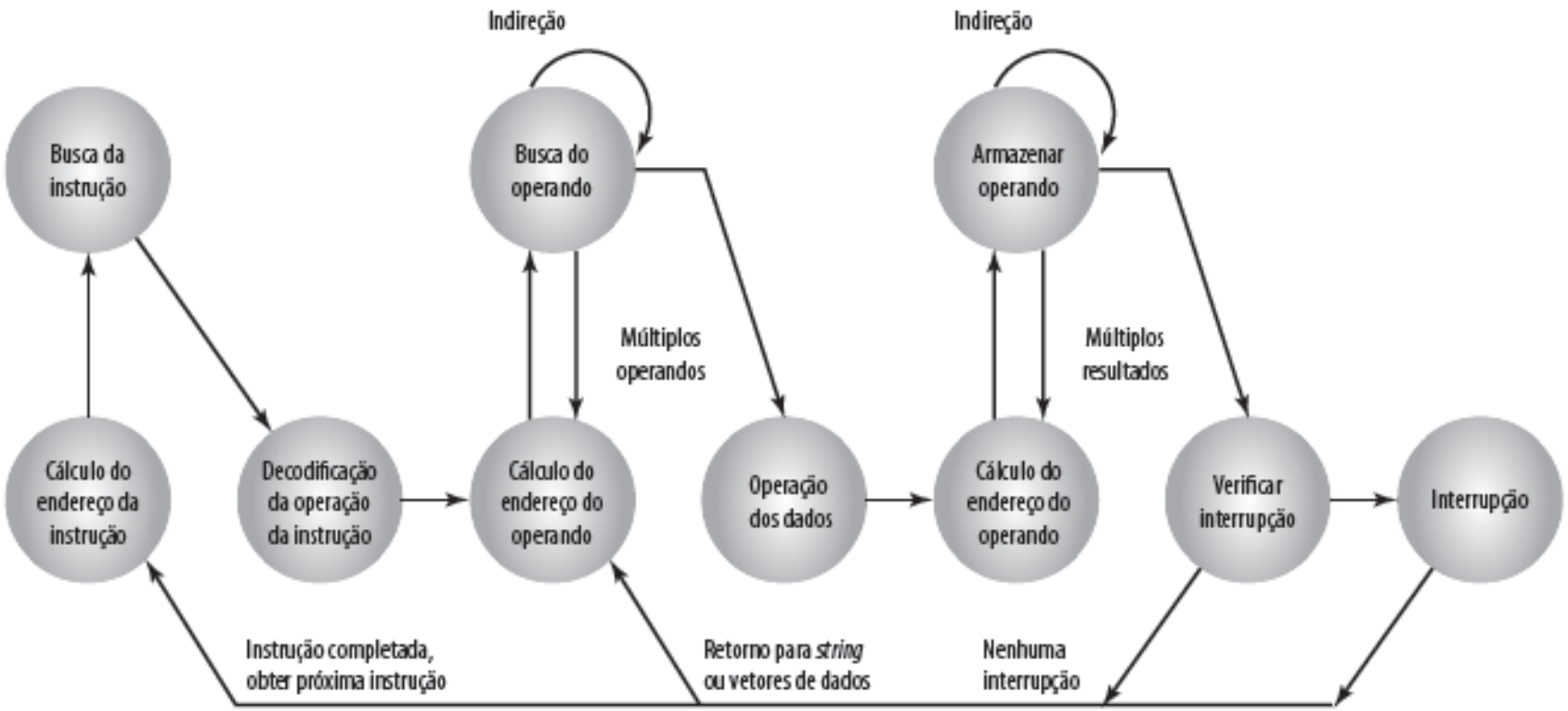


Diagrama de estado do ciclo da instrução

Ciclo da instrução



Fluxo de dados (busca de instrução)

- Depende do projeto da CPU
- Em geral:

- Busca:
  - PC contém endereço da próxima instrução
  - Endereço movido para MAR
  - Endereço colocado no barramento de endereço
  - Unidade de controle solicita leitura de memória
  - Resultado colocado no barramento de dados, copiado para MBR, depois para IR
  - Enquanto isso, PC incrementado em 1

Fluxo de dados (busca de dados)

- IR é examinado
- Se endereçamento indireto, ciclo indireto é realizado
  - N bits da extrema direita do MBR transferidos para MAR
  - Unidade de controle solicita leitura de memória
  - Resultado (endereço do operando) movido para MBR

Fluxo de dados (execução)

Ciclo da instrução

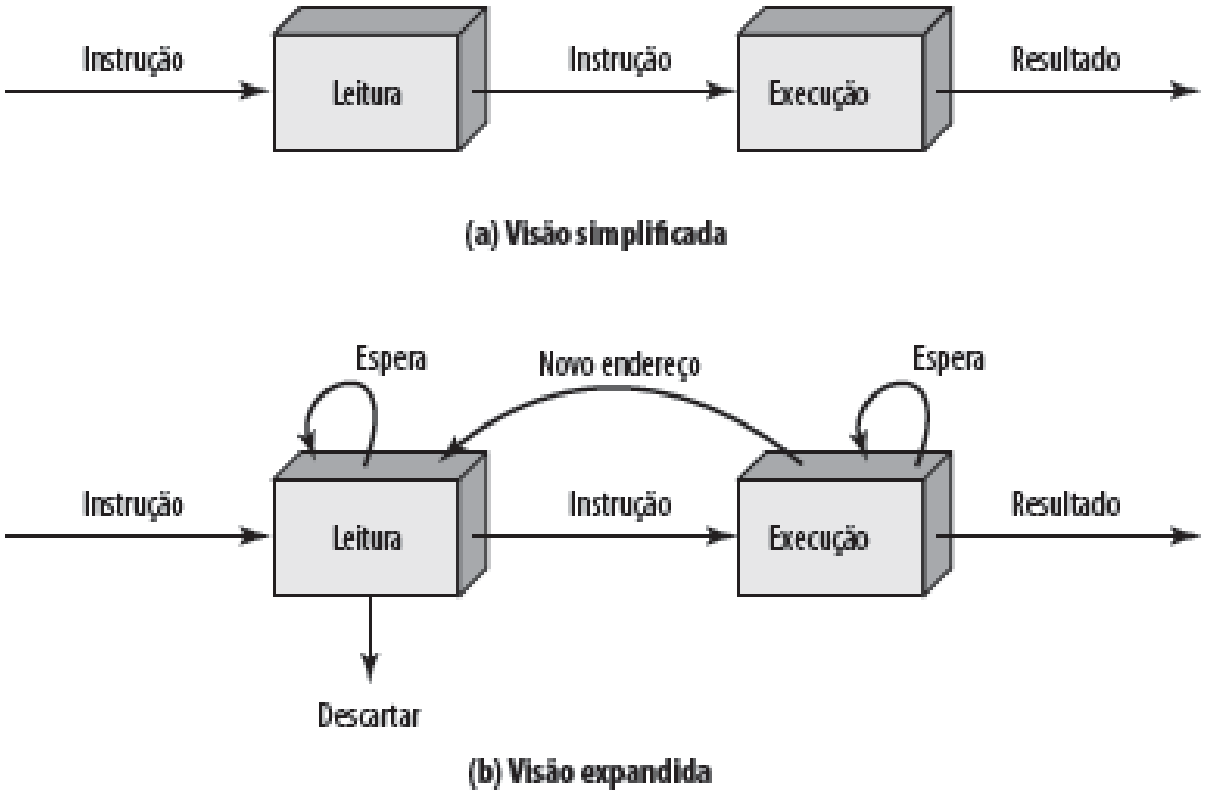
- Pode tomar muitas formas
- Depende da instrução sendo executada
- Pode incluir:
  - Leitura/escrita da memória
  - Entrada/saída
  - Transferências de registradores
  - Operações da ALU

Fluxo de dados (interrupção)

Ciclo da instrução

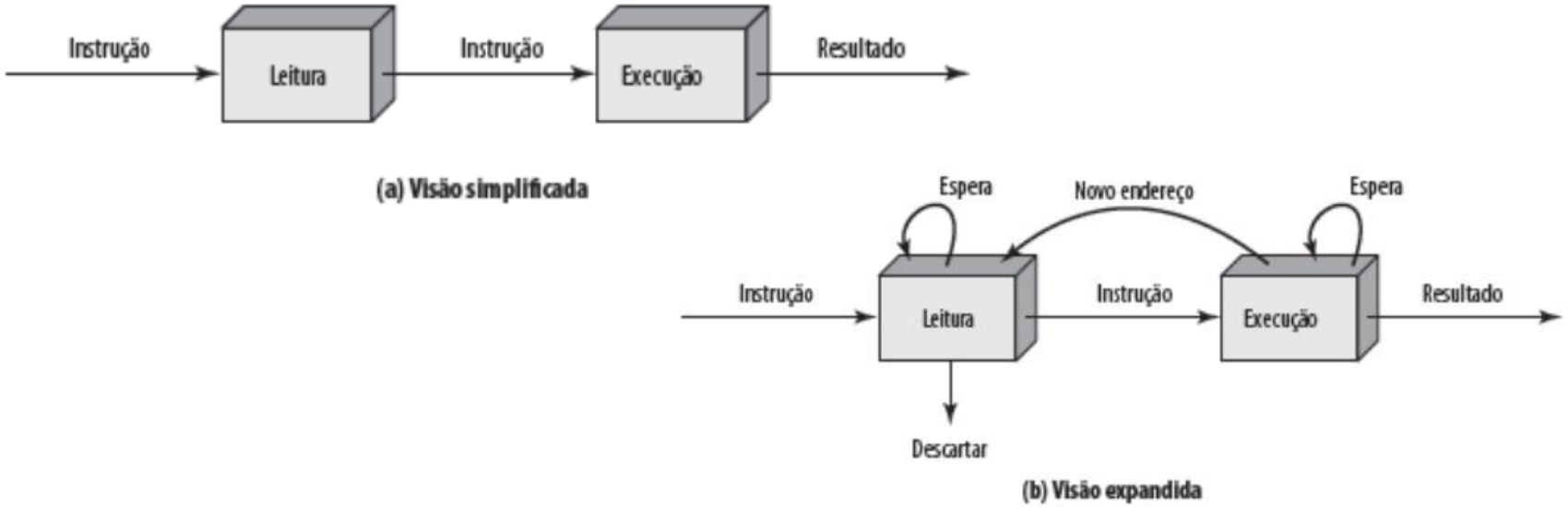
- Simples
- Previsível
- PC atual salvo para permitir retomada após interrupção
- Conteúdo do PC copiado para MBR
- Local especial da memória (Ex. ponteiro de pilha) carregado no MAR
- MBR gravado na memória
- PC carregado com endereço da rotina de tratamento de interrupção
- Próxima instrução (primeira do tratador de interrupção) pode ser obtida

Pipeline de instrução de dois estágios



Busca antecipada (prefetch)

Pipeline de instruções



- Busca acessando memória principal.
- Execução normalmente não acessa memória principal.
- Pode buscar próxima instrução durante execução da instrução atual.
- Chamada busca antecipada da instrução.

Desempenho melhorado

- Mas, não dobrado:
  - Busca normalmente mais curta que a execução
  - Busca antecipada de mais de uma instrução?
  - Qualquer salto ou desvio significa que as instruções com busca antecipada não são as instruções solicitadas
- Acrescente mais estágios para melhorar o desempenho

Pipelining

Pipeline de instruções

- Pipeline de 6 estágios

- Buscar instrução: ler a próxima instrução esperada em um buffer (FI)
- Decodificar instrução: determinar o opcode e os especificadores dos operandos (DI)
- Calcular operandos: calcula o endereço efetivo de cada operando de origem. Isto pode envolver endereçamento por deslocamento, registrador indireto ou outras formas de cálculo de endereço (CO)

Pipelining

Pipeline de instruções

- Pipeline de 6 estágios

- Obter operandos: obter cada operando da memória. Operandos que estão nos registradores não precisam ser lidos da memória. (FO)
- Executar instrução: efetuar a operação indicada e armazenar o resultado, se houver, na posição do operando de destino especificado (EI)
- Escrever operando: armazenar o resultado na memória (WO)

- Sobrepor estas operações

Diagrama de tempo para a operação do pipeline da instrução

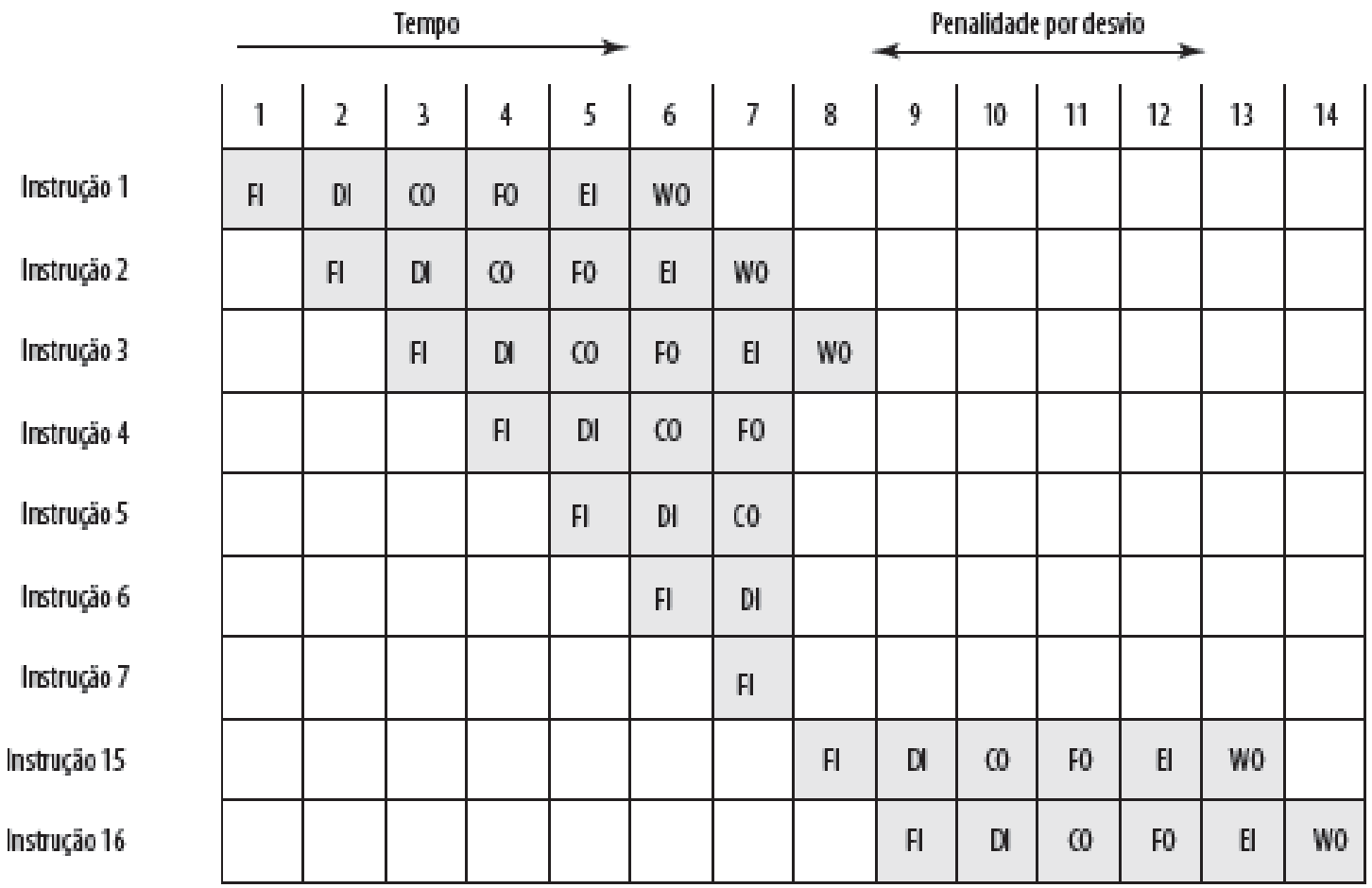
Pipeline de instruções – Diagrama de temo para operação do pipeline de instrução

Tempo →

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instrução 1	FI	DI	CO	FO	EI	WO								
Instrução 2		FI	DI	CO	FO	EI	WO							
Instrução 3			FI	DI	CO	FO	EI	WO						
Instrução 4				FI	DI	CO	FO	EI	WO					
Instrução 5					FI	DI	CO	FO	EI	WO				
Instrução 6						FI	DI	CO	FO	EI	WO			
Instrução 7							FI	DI	CO	FO	EI	WO		
Instrução 8								FI	DI	CO	FO	EI	WO	
Instrução 9									FI	DI	CO	FO	EI	WO

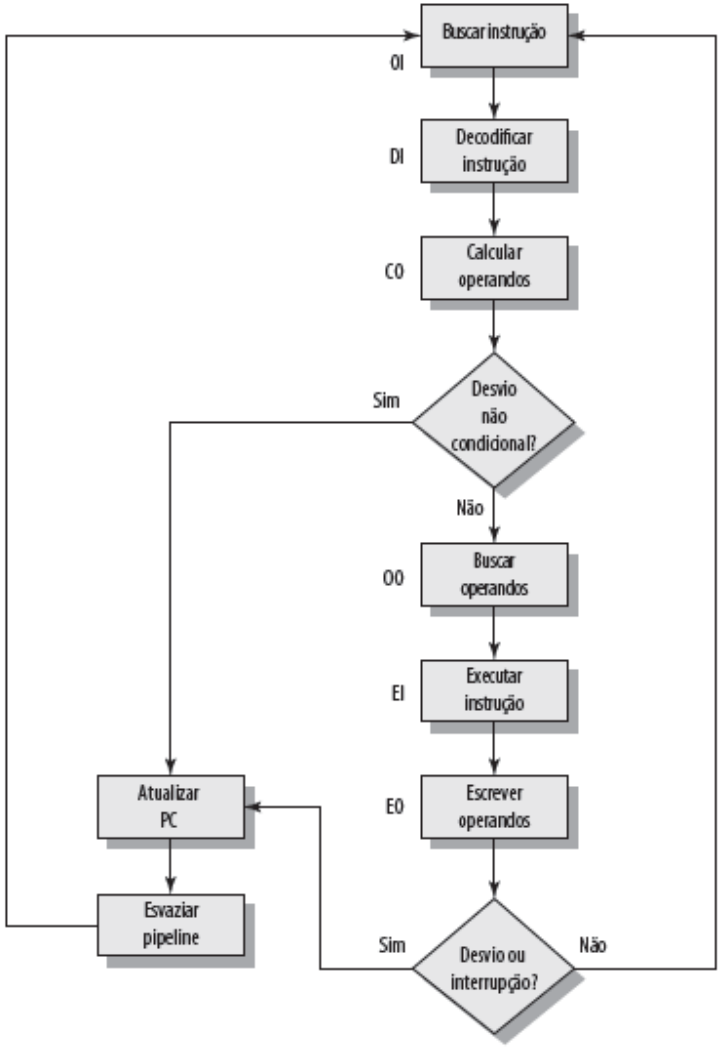
Efeito de desvio condicional na operação do pipeline na instrução

Pipeline de instruções – Efeito de um desvio condicional na operação do pipeline de instrução



Pipeline de instrução de seis estágios

Pipeline de instruções de uma CPU de seis estágios



Descrição alternativa de um pipeline

Tempo

	FI	DI	CO	FO	EI	WO
1	I1					
2	I2	I1				
3	I3	I2	I1			
4	I4	I3	I2	I1		
5	I5	I4	I3	I2	I1	
6	I6	I5	I4	I3	I2	I1
7	I7	I6	I5	I4	I3	I2
8	I8	I7	I6	I5	I4	I3
9	I9	I8	I7	I6	I5	I4
10		I9	I8	I7	I6	I5
11			I9	I8	I7	I6
12				I9	I8	I7
13					I9	I8
14						I9

(a) Sem desvios

	FI	DI	CO	FO	EI	WO
1	I1					
2	I2	I1				
3	I3	I2	I1			
4	I4	I3	I2	I1		
5	I5	I4	I3	I2	I1	
6	I6	I5	I4	I3	I2	I1
7	I7	I6	I5	I4	I3	I2
8	I15					I3
9	I16	I15				
10		I16	I15			
11			I16	I15		
12				I16	I15	
13					I16	I15
14						I16

(b) Com desvios condicionais

Desempenho do pipeline

- Medições simples de desempenho do pipeline e a correspondente melhoria da velocidade (Hwang, 1993)

- Tempo de ciclo de uma instrução do pipeline τ : é o tempo necessário para que a instrução avance um estágio dentro do pipeline. O tempo de ciclo pode ser determinado como:

$$\tau = \max[\tau_i] + d = \tau_m + d \qquad 1 \le i \le k$$

τ_i = tempo de demora de respota do circuito no estágio i do pipeline

τ_m = tempo de demora máximo no estágio (demora do estágio que apresenta o maior temp o de demora de resposta)

k = número de estágios na instrução do pipeline

d = tempo de resposta de um ciclo necessário para avançar sinais e dados de um estágio para o próximo.

Desempenho do pipeline

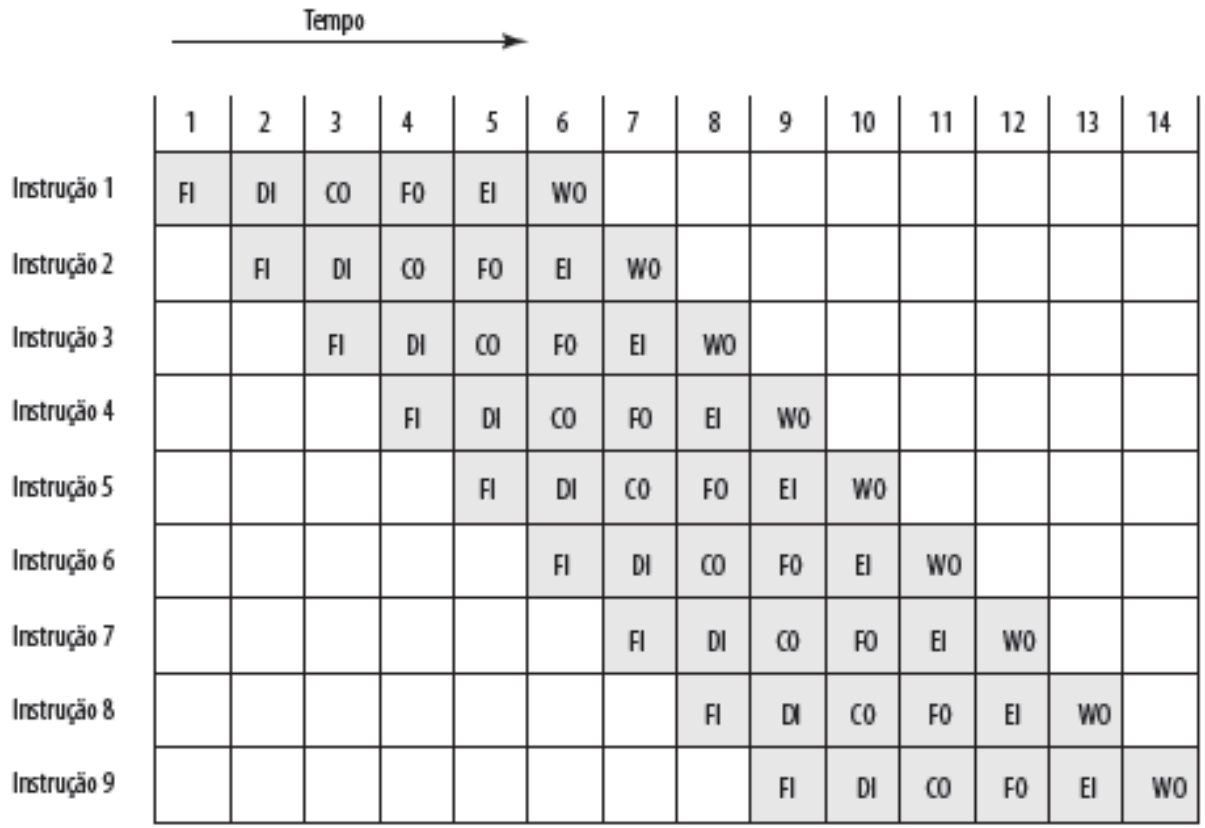
- Em geral o tempo de resposta d é equivalente a um pulso de clock e $\tau_m >> d$. Suponha agora que n instruções são processadas, sem desvios. Seja $T_{k,n}$ o tempo total necessário para que um pipeline de k estágios processe n instruções, então:

$$T_{k,n} = [k + (n-1)]\tau$$

- Um total de k ciclos é necessário para completar a execução da primeira instrução e o restante (n-1) instruções requerem (n-1) ciclos. Esta equação é facilmente verificada na figura a seguir:

Desempenho do pipeline

Pipeline de instruções



- A nona instrução completa no ciclo de tempo 14: 14 = [6+(9-1)]

Desempenho do pipeline

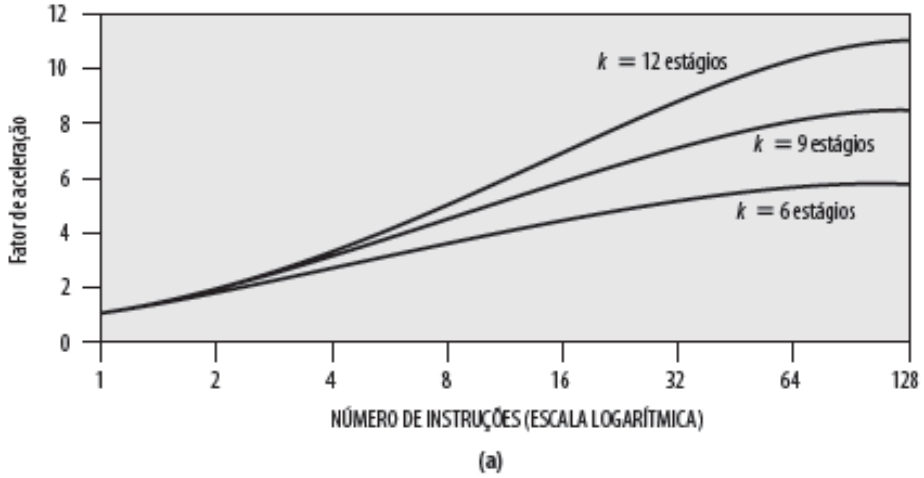
Pipeline de instruções

- Consideremos agora um processador com funções equivalentes, mas sem pipeline, e vamos supor que o tempo de ciclo de instrução seja $k.\tau$.

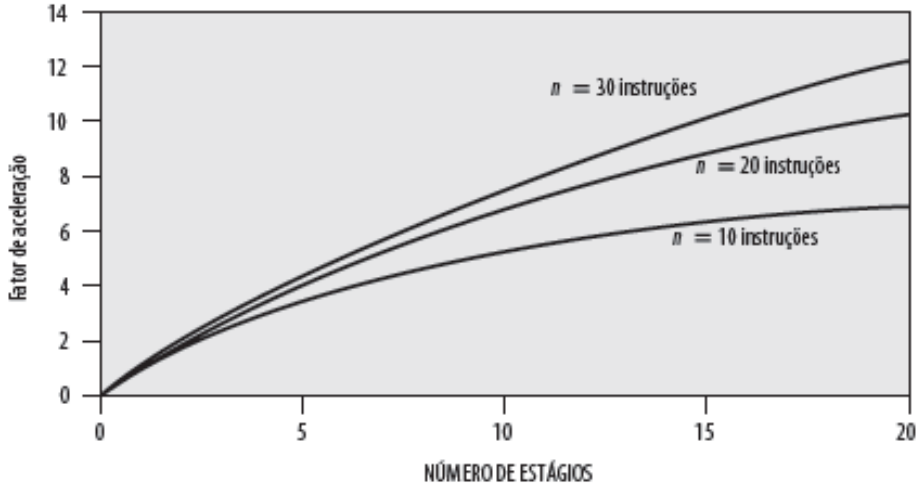
  O fator de aceleração para a instrução do pipeline comparado com a execução sem pipeline é definido como:

$$S_k = \frac{T_{1,n}}{T_{k,n}} = \frac{nk\tau}{[k+(n-1)]\tau} = \frac{nk}{k+(n-1)}$$

Fatores de aceleração com pipeline da instrução



(a)



(b)

Exercícios

1) Um microprocessador fornece uma instrução capaz de mover uma cadeia de bytes de uma área de memória para outra. A leitura e decodificação inicial da instrução leva 10 ciclos de clock. Depois demora 15 ciclos de clock para transferir cada byte. O microprocessador possui um clock de 10 GHz.

a) Determine o tamanha do ciclo da instrução para o caso de uma cadeia de 64 bytes.

b) Qual é o pior atraso para aceitar uma interrupção se a instrução não puder ser interrompida?

c) Repita a parte (b) assumindo que a instrução possa ser interrompida no começo da transferência de cada byte.

Exercícios

2) Um processador pipeline tem uma taxa de clock de 2,5 GHz e executa um programa com 1,5 milhões de instruções. O pipeline possui cinco estágios e as instruções são emitidas numa taxa de uma por ciclo de clock. Ignore penalidades por causa das instruções de desvio e execuções fora de ordem.

a) Qual a diferença de velocidade deste processador para este programa comparado a um processador sem pipeline?

b) Qual o rendimento (em MIPS) do processador com pipeline?

Exercícios

3) Um processador sem pipeline tem um taxa de clock de 2.5 GHz e um CPI (ciclo por instrução) médio de 4. Uma atualização no processador introduz um pipeline de cinco estágios. No entanto, por causa dos atrasos internos do pipeline, a taxa de clock do novo processador deve ser reduzida para 2 GHz.

a) Qual o aumento de velocidade obtido para um programa típico?

b) Qual a taxa MIPS para cada processador?

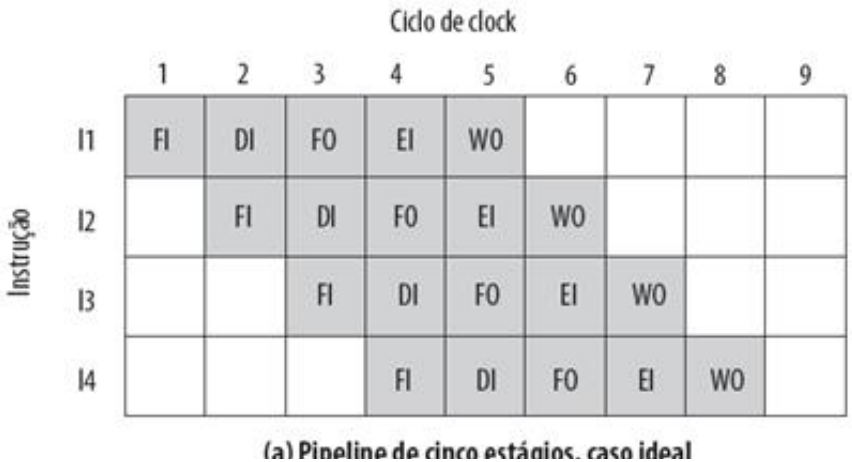
Hazards do pipeline

- Pipeline, ou alguma parte do pipeline, precisa parar.
- Também conhecida como *bolha de pipeline*.
- Tipos de hazards:
  - Recursos
  - Dados
  - Controle

Hazards de recursos

- Duas (ou mais) instruções no pipeline precisam do mesmo recurso.
- As instruções precisam ser executadas em série, e não em paralelo.
- Também chamado *hazard estrutural*.
- P.ex., considere pipeline simplificado em 5 estágios.
  - Cada estágio usa um ciclo de clock.
- No caso ideal, cada nova instrução entra no pipeline a cada ciclo de clock.
- Suponha que a memória principal tenha única porta.
- Considere buscas de instrução e leituras e escritas de dados uma por vez.
- Ignore a cache.
- Leitura ou escrita de operando não podem ser realizadas em paralelo com busca de instrução.
- Estágio de busca de instrução fica ocioso por um ciclo buscando I3.
- P.ex., várias instruções prontas para entrar na fase de execução de instrução.
- Única ALU.
- Uma solução: aumentar recursos disponíveis.
  - Múltiplas portas da memória principal.
  - Múltiplas ALUs.

Hazards de recursos



(a) Pipeline de cinco estágios, caso ideal

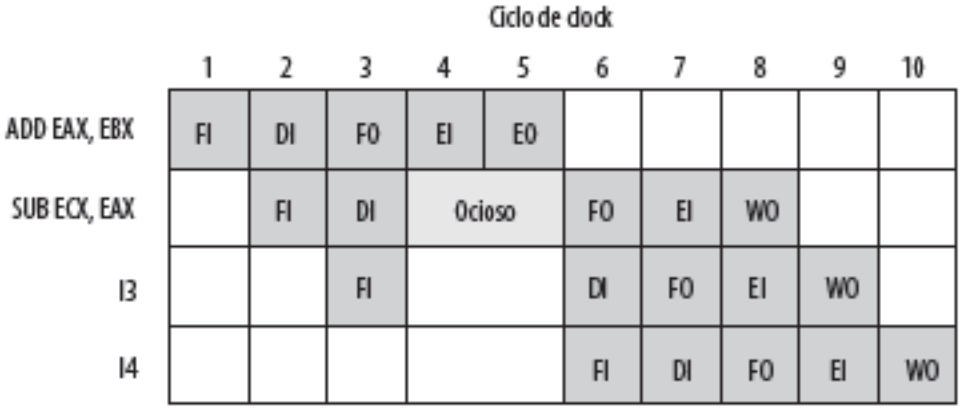
(b) Operando de origem de I1 na memória

Hazards de dados

- Conflito no acesso de um local de operando.
- Duas instruções a serem executadas em sequência.
- Ambas acessam uma memória em particular ou operando do registrador.
- Se na sequência estrita, não ocorre problema.
- Se em um pipeline, valor do operando poderia ser atualizado para produzir resultado diferente da execução sequencial estrita.
- P.ex., sequência de instruções de máquina do x86:

- ADD EAX, EBX /* EAX = EAX + EBX
- SUB ECX, EAX /* ECX = ECX – EAX

Exemplo de hazard de dados



- ADD EAX, EBX /* EAX = EAX + EBX
- SUB ECX, EAX /* ECX = ECX – EAX

- Instrução ADD não atualiza EAX até o fim do estágio 5, no ciclo de clock 5.
- Instrução SUB precisa do valor no início do seu estágio 2, no ciclo de clock 4.
- Pipeline precisa parar por dois ciclos de clock.
- Sem hardware especial e algoritmos de impedimento específicos, resulta em uso ineficaz do pipeline.

Tipos de hazard de dados

- Leitura após escrita (RAW), ou **dependência verdadeira**:
  - Uma instrução modifica um registrador ou local de memória.
  - Instrução seguinte lê dados nesse local.
  - Hazard ocorre **se leitura** ocorre antes do término da escrita.
- Escrita após leitura (WAR), ou **antidependência**:
  - Uma instrução lê um registrador ou local da memória.
  - Instrução seguinte escreve no local.
  - Hazard ocorre **se escrita** termina antes que ocorra a leitura.
- Escrita após escrita (WAW), ou **dependência de saída**:
  - Duas instruções escrevem no mesmo local.
  - Hazard ocorre **se a escrita** ocorre na ordem contrária à sequência intencionada.
- Exemplo anterior é um hazard RAW.

Hazard de controle

- Também conhecido como *hazard de desvio*.
- Pipeline toma decisão errada sobre previsão de desvio.
- Traz para o pipeline instruções que precisam ser descartadas subsequentemente.

- Lidando com desvios: (5 técnicas)
  - Múltiplos fluxos.
  - Busca antecipada do alvo do desvio.
  - Buffer de laço de repetição.
  - Previsão de desvio.
  - Desvio atrasado.

Múltiplos fluxos (1)

- Dois pipelines (replicar as partes iniciais do pipeline a fim de permitir que o pipeline obtenha duas instruções)
- Usa pipeline apropriado.

- Problemas com esta abordagem:
  - Ocasiona disputa por barramento e registrador (atraso no acesso aos registradores e à memória)
  - Múltiplos desvios (novas instruções de desvio) exigem um fluxo adicional.

Busca antecipada do alvo do desvio (2)

- Quando um desvio condicional é reconhecido, o alvo do desvio é buscado antecipadamente além das instruções após o desvio.
- Mantém alvo (salvo) até que o desvio seja executado.

Buffer de laço de repetição (3)

- Um buffer de laço de repetição é uma memória pequena e extremamente rápida mantida pelo estágio do pipeline de busca da instrução que contém as n instruções mais recentemente lidas na sequência.

- Se um desvio está para ser tomado, o hardware primeiro verifica se o alvo do desvio já está no buffer. Se estiver, a próxima instrução é obtida do buffer.

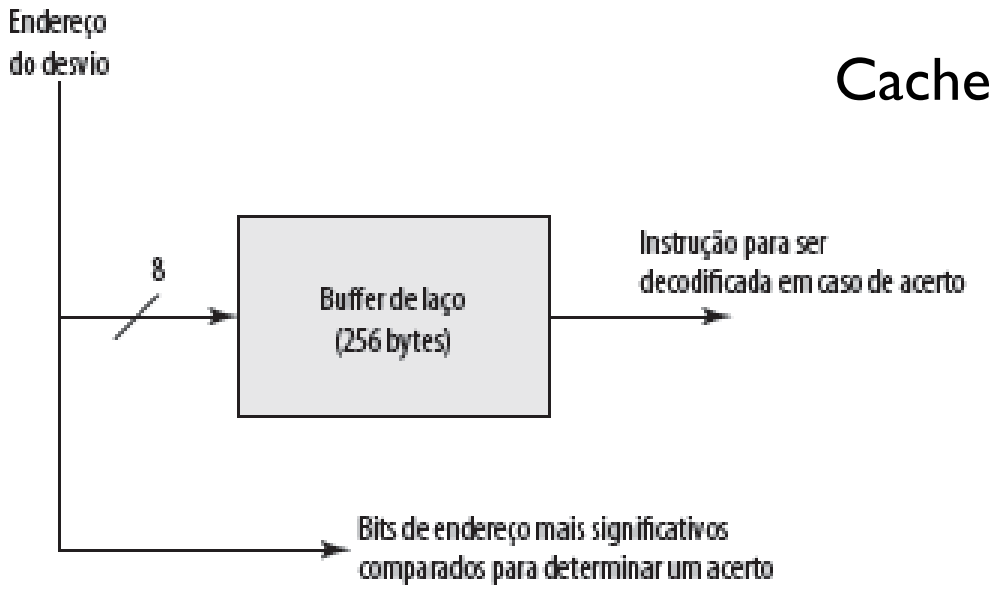
Buffer de laço de repetição (3)

Vantagens

- Com o uso de busca antecipada, o buffer de laço conterá algumas instruções em sequência na frente do endereço da instrução atual. Assim, as instruções obtidas na sequência estarão disponíveis sem o tempo usual de acesso à memória.
- Se um desvio para um alvo estiver apenas algumas posições à frente do endereço da instrução de desvio, o alvo já estará no buffer. (IF THEN – IF THEN ELSE)
- Se o buffer de laço de repetição for suficientemente grande para conter todas as instruções de um laço, então essas instruções precisam ser obtidas da memória apenas uma vez, na primeira iteração. Para iterações subsequentes, todas as instruções necessárias já estão no buffer.

Buffer de laço de repetição (3)

- Memória muito rápida.
- Mantido pelo estágio de busca do pipeline.
- Verifica buffer antes de buscar da memória.
- Muito bom para laços ou saltos pequenos.



Cache ?

Previsão de desvio (4)

- Previsão nunca tomada:
  - Assume que salto não acontecerá.
  - Sempre busca próxima instrução.
  - 68020 & VAX 11/780.
- Previsão sempre tomada:
  - Assume que salto acontecerá.
  - Sempre busca instrução alvo.

Previsão de desvio (5)

- Previsão por *opcode:*
  - Algumas instruções são mais prováveis de resultar em um salto do que outras.
  - Pode chegar até 75% de sucesso (Lilja, 1988)
- Chave tomada/não tomada:
  - Baseada no histórico de desvio.
  - Boa para laços.
  - Refinada pelo histórico de desvio com dois níveis ou baseado em correlação.
- Baseado em correlação:
  - Nos desvios de laço, o histórico é uma boa forma de previsão.
  - Em estruturas mais complexas, a direção do desvio é correlacionada com a direção de desvios condicionados.
    - Também usa histórico de desvios recentes.

Previsão de desvio (5)

- Desvio atrasado:
  - Não salta até que você realmente precise.
  - Reorganiza instruções (rearranjo automático de instruções dentro de um programa, para que as instruções ocorram depois do que realmente desejado).

Pipeline de Intel 80486 (5 estágios)

- Leitura:
  - Da cache ou da memória externa.
  - Colocadas em um de 2 buffers de busca antecipada de 16 bits.
  - Preenche buffer com novos dados quando antigos são consumidos pelo decodificador de instrução.
  - Em média, 5 instruções lidas por carga.
  - Opera independente de outros estágios para manter buffers cheios.

- Estágio de decodificação 1:
  - *Opcode* e informação de modo de endereçamento.
  - No máximo 3 primeiros bytes da instrução.
  - Pode direcionar estágio D2 para obter restante da instrução (dados imediatos e deslocamento)

Pipeline de Intel 80486

- Estágio de decodificação 2:
  - Expande *opcode* para sinais de controle para ALU.
  - Cálculo de modos de endereçamento complexos.
- Execução:
  - Operações da ALU, acesso a cache, atualização de registrador.
- Escrita:
  - Atualiza registradores e flags de estado modificados durante o processo de execução anterior.
  - Resultados enviados à cache e buffers de escrita da interface de barramento.

Processador ARM

- ARM é um sistema RISC.
- Principais características:
  - Possui um conjunto moderado de registradores uniformes, mais do que são encontrados em alguns sistemas CISC, porém menos do que encontrados em muitos sistemas RISC.
  - Modelo *load/store* de processamento de dados, no qual as operações executam apenas com os operandos nos registradores e não diretamente na memória. Todos os dados precisam ser carregados em registradores antes que uma operação possa ser efetuada; o resultado pode ser usado para o processamento posterior ou armazenado em memória.
  - Instrução uniforme de tamanho fixo de 32 bits par o conjunto padrão e 16 bits para o conjunto de instruções Thumb (diminui tamanho do código e aumenta a economia de energia)
  - Para tornar cada instrução de processamento de dados mais flexível, um deslocamento ou rotação pode pré-processar um dos registradores de origem. Para suportar esse recurso eficientemente, a ALU e unidades de deslocamento são separadas.

Processador ARM

- Um número pequeno de modos de endereçamento com todos os endereços de carga/armazenamento sendo determinados a partir dos registradores e campos da instrução. Endereçamento indireto ou indexado envolvendo valores na memória não é utilizado.

- Modos de endereçamento com autoincremento e autodecremento são usados para melhorar a operação de laços de repetição dos programas.

- Execução condicional das instruções minimiza a necessidade das instruções de desvios condicionais, melhorando assim a eficiência do pipeline, porque a limpeza do pipeline é reduzida.

Organização do processador ARM

- Muitas variações dependem da versão do ARM.
- Dados trocados entre processador e memória através de barramento de dados.
- Item de dados (load/store) ou instrução (leitura).
- Instruções passam por decodificador antes da execução.
- Pipeline e geração de sinal de controle na unidade de controle.
- Dados vão para arquivo de registrador:
  - Conjunto de registradores de 32 bits.
  - Um byte ou meia palavra tratados como complemento de dois estendidos com sinal até 32 bits.
- Normalmente dois registradores de origem e um resultado.
- Rotação ou deslocamento antes da ALU.

Modos do processador

- É comum um processador suportar apena um número pequeno de modos do processador. Muitos sistemas operacionais usam apenas dois modos: modo usuário e modo *kernel* (utilizado para executar software de sistemas privilegiado).

- A arquitetura ARM permite uma base flexível para que os sistemas operacionais possam impor uma variedade de políticas de proteção.
  - **Modos do processador**
    - Modo supervisor
    - Modo de abortamento
    - Modo indefinido
    - Modo interrupção rápido
    - Modo de interrupção

Modos do Processador - Modos privilegiados

- Supervisor:
  - Normalmente é o modo em que executa o SO. Ele é ativado quando o processador encontra uma interrupção de software. Interrupções de software são um jeito padrão para invocar os serviços do sistema operacional no ARM.
- Modo de abortamento:
  - Ativado como resposta a falhas de memória.
- Modo indefinido:
  - Ativado quando o processador tenta executar uma instrução que não é suportada nem pelo núcleo principal nem por um dos coprocessadores.

Modos do Processador - Modos privilegiados

- Modo de interrupção rápido:
  - Ativado sempre que o processador recebe um sinal de interrupção a partir de uma fonte designada de interrupção rápida. Uma interrupção rápida não pode ser interrompida, porém uma interrupção rápida pode interromper uma interrupção normal.

- Modo de interrupção.
  - Ativado sempre que o processador recebe um sinal de interrupção a partir de qualquer outra origem de interrupção (diferente da interrupção rápida). Uma interrupção somente pode ser interrompida por uma interrupção rápida

- Modo do sistema
  - Este modo não é ativado por nenhuma exceção e usa os mesmos registradores disponíveis no modo usuário. O modo de sistema é usado para executar certas tarefas privilegiadas do SO. As tarefas de modo de sistema podem ser interrompidas por qualquer uma das cinco categorias de exceção.

Organização dos registradores do ARM

Modos						
	Modos privilegiados					
		Modos de exceção				
Usuário	Sistema	Supervisor	Abortamento	Indefinido	Interrupção	Interrupção rápida
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8_fiq
R9	R9	R9	R9	R9	R9	R9_fiq
R10	R10	R10	R10	R10	R10	R10_fiq
R11	R11	R11	R11	R11	R11	R11_fiq
R12	R12	R12	R12	R12	R12	R12_fiq
R13(SP)	R13(SP)	R13_svc	R13_abtR	13_und	R13_irq	R13_fiq
R14(LR)	R14(LR)	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
R15(PC)	R15(PC)	R15(PC)	R15(PC)	R15(PC)	R15(PC)	R15(PC)
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

Sombreado indica que o registrador normal usado pelo modo usuário ou de sistema foi substituído por um registrador específico para modo de exceção.

SP = ponteiro de pilha
LR = registrador de ligação
PC = contador de programa

CPSR = registrador de estado de programa corrente
SPSR = registrador de estado de programa salvo

Organização dos registradores do ARM

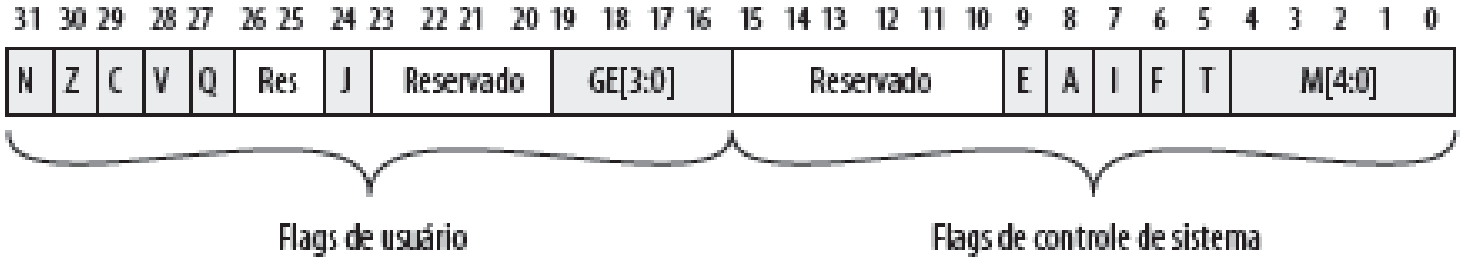
- 37 registradores de 32 bits.
- 31 registradores de uso geral.
  - Alguns têm propósitos especiais.
  - P.ex., contadores de programa.
- Seis registradores de *status* de programa.
- Registradores em bancos parcialmente sobrepostos.
  - Modo processador determina banco.
- 16 registradores numerados e um ou dois registradores de *status* de programa visíveis.

Registradores de propósito geral

- R13– normalmente ponteiro de pilha (SP).
  - Cada modo de exceção tem seu próprio R13.
- R14 – registrador de ligação (LR).
  - Endereço de retorno da sub-rotina e retornos do modo de exceção.
- R15 contador de programa.

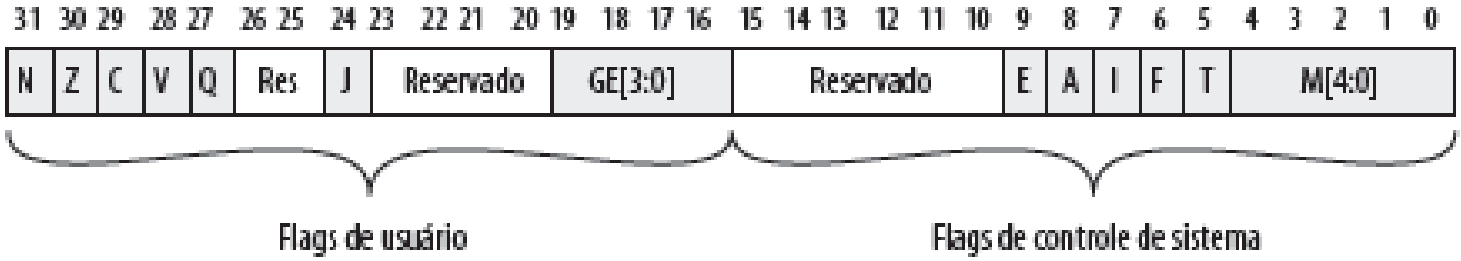
CPSR (*Current program status register*)

- CPSR processa registrador de estado.
  - Modos de exceção têm SPSR dedicado.
- 16 bits mais significativos flags do usuário.
  - Códigos de condição (N,Z,C,V).
  - Q– estouro ou saturação em instruções SMID.
  - J– instruções Jazelle (8 bits). (DBX)
  - GE[3:0] SMID usam bits [19:16] como flag de maior ou igual.



CPSR (*Current program status register*)

- 16 bits menos significativos contêm flags para modo privilegiado.
  - E– endian (armazenamento *endianness*)
  - A, I e F - Desabilitar interrupção.
  - T– instrução normal ou Thumb.
  - Modo (indica o modo do processador)



Endianess

Big Endian

Supondo unidade de endereçamento byte (8 bits):

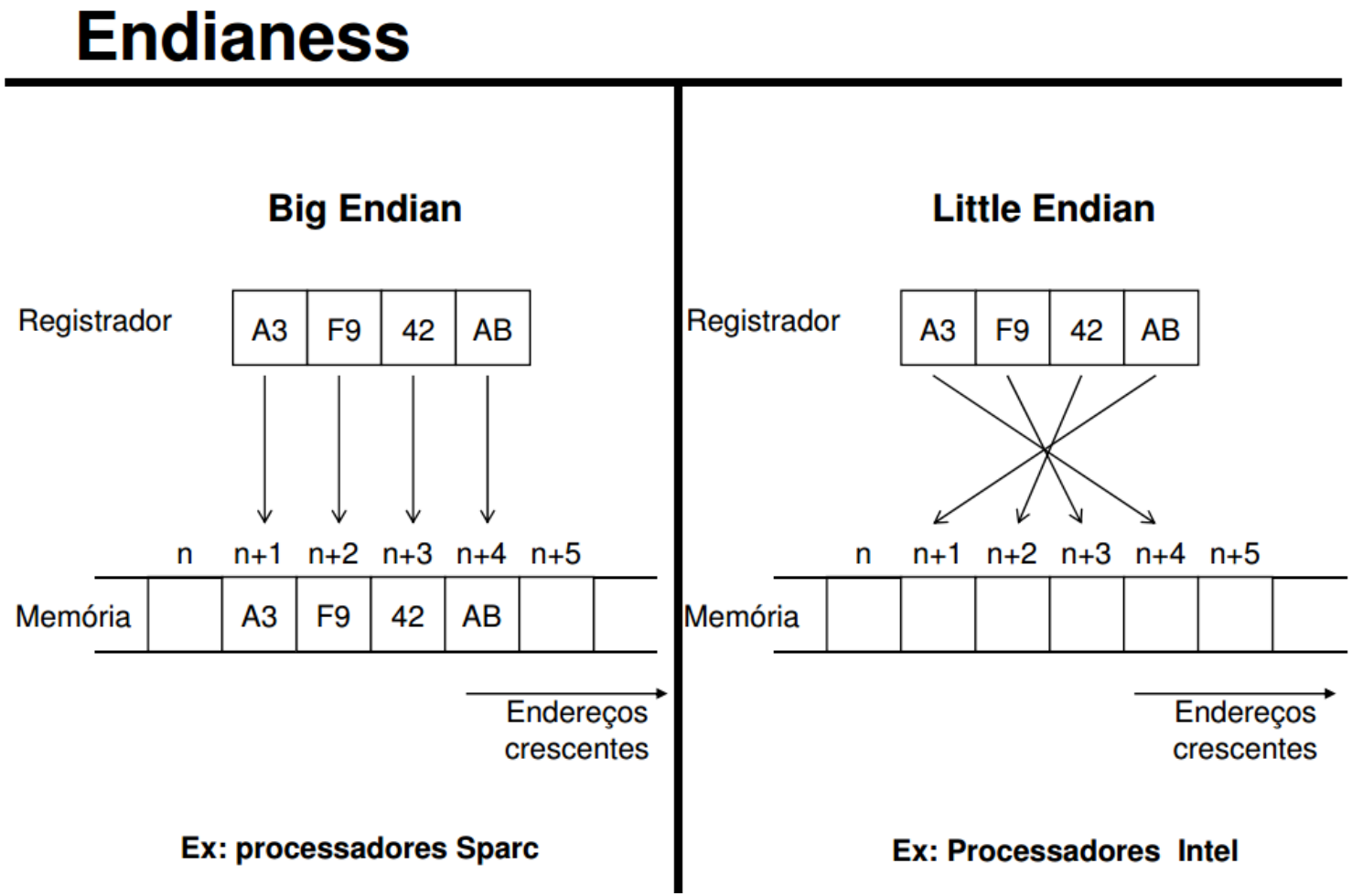
O valor do byte mais significativo (Most Significant Byte – MSB) é armazenado na posição de memória de menor endereço. O próximo byte é armazenado na próxima posição, e assim por diante.

Little Endian

Supondo unidade de endereçamento byte (8 bits):

O valor do byte menos significativo (Least Significant Byte – LSB) é armazenado na posição de memória de menor endereço.

Endianess


Endianess
Big Endian
Registrador
A3
F9
42
AB
n
n+1
n+2
n+3
n+4
n+5
Memória
A3
F9
42
AB
Endereços crescentes
Ex: processadores Sparc
Little Endian
Registrador
A3
F9
42
AB
n
n+1
n+2
n+3
n+4
n+5
Memória
Endereços crescentes
Ex: Processadores Intel

Processamento de interrupção (exceção) ARM

- Mais de uma exceção permitida (geradas por fontes internas ou externas).
- Sete tipos de exceções são suportadas.
- Execução forçada por vetores de exceção.
- Múltiplas exceções tratadas em ordem de prioridade.
- Processador para a execução após instrução atual.
- Estado do processador preservado no SPSR para exceção:
  - Endereço da instrução a executar colocado no registrador de ligação.
  - Retorna movendo SPSR p/ CPSR e R14 p/ PC.