William Stallings
Arquitetura e Organização de Computadores
8ª Edição

Capítulo 10
Conjuntos de instruções: Características e funções

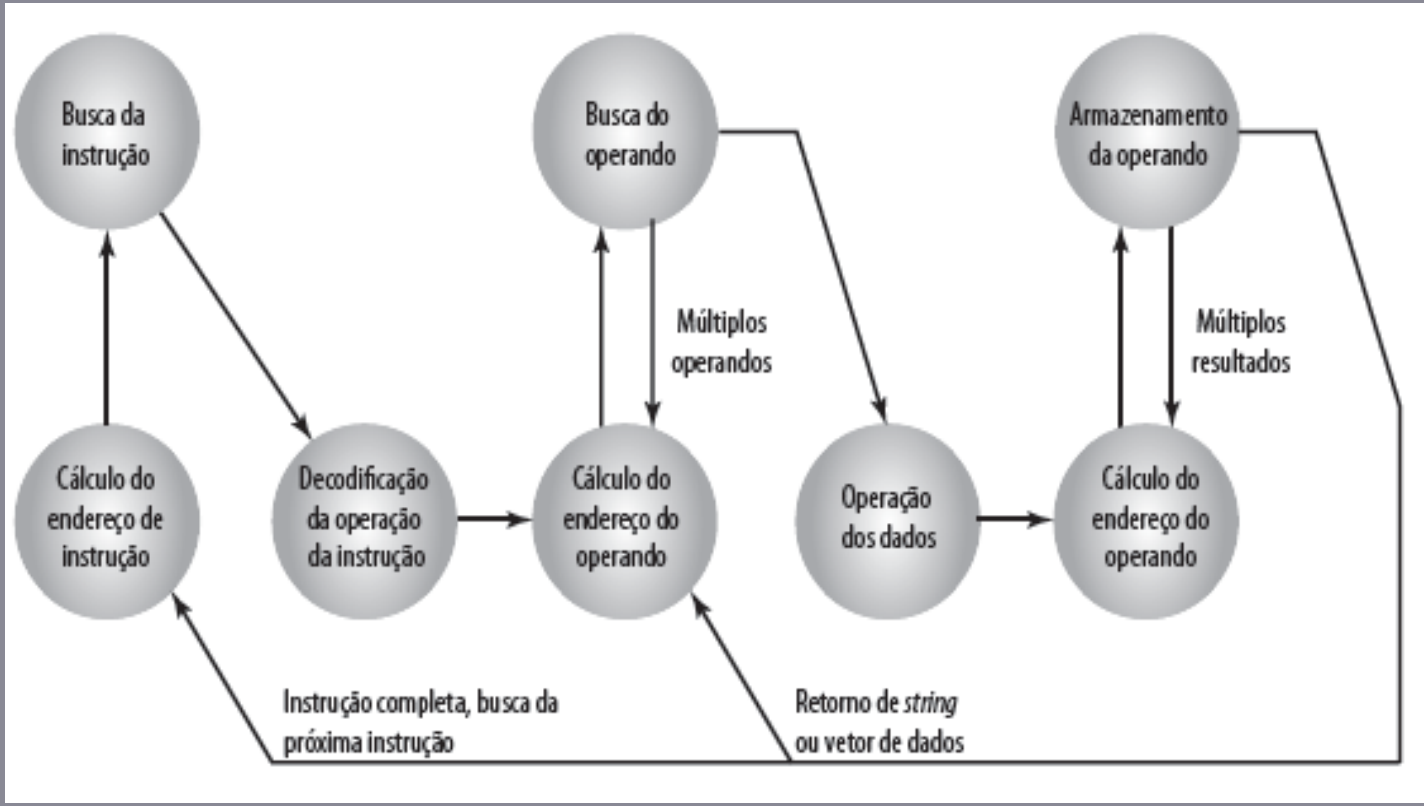
O que é um conjunto de instruções?

- A coleção completa de instruções que são entendidas por uma CPU.
- Código de máquina.
- Binário.
- Normalmente, representado por códigos em *assembly*.

Elementos de uma instrução

- Código de operação (Op code):
  - Faça isto.
- Referência a operando fonte:
  - Nisto.
- Referência a operando de destino:
  - Coloque a resposta aqui.
- Referência à próxima instrução:
  - Quando tiver feito isso, faça isto...

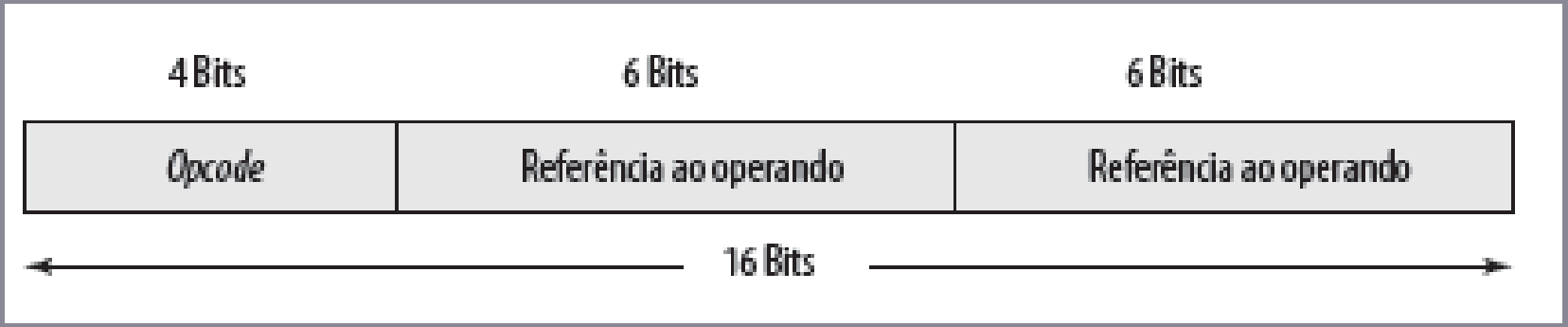
Diagrama de estado do ciclo de instrução



Representação da instrução

- Em código de máquina, cada instrução tem um padrão de bits exclusivo.
- Para consumo humano (bem, para programadores), uma representação simbólica é utilizada.
  - P.e., ADD, SUB, LOAD.
- Operandos também podem ser representados desta maneira:
  - ADD A,B.

Formato de instrução simples



Tipos de instrução

- Processamento de dados.
- Armazenamento de dados (memória principal).
- Movimentação de dados (E/S).
- Controle de fluxo do programa.

Número de endereços (a)

- 3 endereços:
  - Operando 1, Operando 2, Resultado.
  - a = b + c.
  - Pode ser uma instrução for-next (normalmente implícita).
  - Não é comum.
  - Precisa de palavras muito longas para manter tudo.

Número de endereços (b)

- 2 endereços:
  - Um endereço servindo como operando e resultado.
  - a = a + b.
  - Reduz tamanho da instrução.
  - Requer algum trabalho extra.
    - Armazenamento temporário para manter alguns resultados.

Número de endereços (c)

- 1 endereço:
  - Segundo endereço implícito.
  - Normalmente, um registrador (acumulador).
  - Comum nas primeiras máquinas.

Número de endereços (d)

- 0 (zero) endereços:
  - Todos os endereços implícitos.
  - Usa uma pilha.
  - p.e. push a.
  - push b.
  - add.
  - pop c.

  - c = a + b.

Quantos endereços

- Mais endereços:
  - Instruções mais complexas (poderosas?).
  - Mais registradores.
    - Operações entre registradores são mais rápidas.
  - Menos instruções por programa.
- Menos endereços:
  - Instruções menos complexas (poderosas?).
  - Mais instruções por programa.
  - Busca/execução de instruções mais rápida.

Quantos endereços

Figura 10.3 Programas para executar $Y = \frac{A - B}{C + (D \times E)}$

Instrução	Comentário
SUB Y, A, B	Y ← A − B
MPY T, D, E	T ← D × E
ADD T, T, C	T ← T + C
DIV Y, Y, T	Y ← Y ÷ T

(a) Instruções com três endereços

Instrução	Comentário
MOVE Y, A	Y ← A
SUB Y, B	Y ← Y − B
MOVE Y, D	T ← D
MPY T, E	T ← T × E
ADD T, C	T ← T + C
DIV Y, T	Y ← Y ÷ T

(b) Instruções de dois endereços

Instrução	Comentário
LOAD D	AC ← D
MPY E	AC ← AC × E
ADD C	AC ← AC + C
STOR Y	Y ← AC
LOAD A	AC ← A
SUB B	AC ← AC − B
DIV Y	AC ← AC ÷ T
STOR Y	Y ← AC

(c) Instruções de um endereço

Decisões de projeto

- Repertório de operações:
  - Quantas operações?
  - O que elas podem fazer?
  - Qual a complexidade delas?
- Tipos de dados.
- Formatos de instrução:
  - Tamanho do campo de código de operação.
  - Número de endereços.

- Registradores:
  - Número de registradores da CPU disponíveis.
  - Quais operações podem ser realizadas sobre quais registradores?
- Modos de endereçamento (mais adiante…).

- RISC v CISC.

Tipos de operando

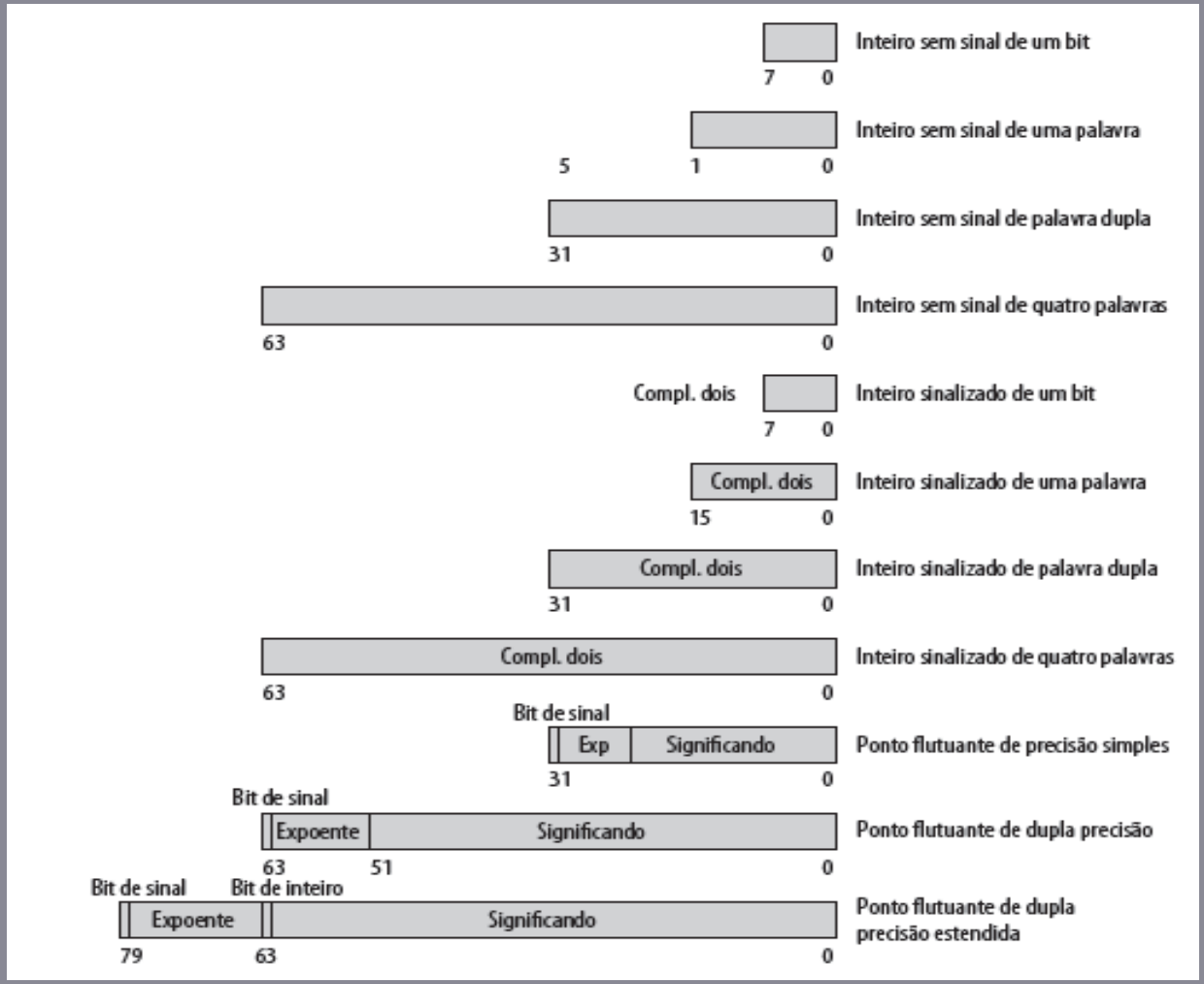
- Endereços.
- Números:
  - Inteiro/ponto flutuante.
- Caracteres:
  - ASCII etc.
- Dados lógicos:
  - Bits ou flags

Tipos de dados do x86

Tabela 10.2 Tipos de dados x86

Tipo de dados	Descrição
Geral	Locais de byte, palavra (16 bits), palavras duplas (32 bits), quatro palavras (64 bits) e quatro palavras duplas (128 bits) com conteúdo binário arbitrário.
Números inteiros	Um valor binário com sinal, contido em um byte, palavra ou palavras duplas, usando a representação de complemento a dois.
Números ordinais	Um inteiro sem sinal contido em um byte, palavra ou palavras duplas.
Números em BCD (*Binary coded decimal*) (BCD) não agrupado	Uma representação de um dígito BCD no intervalo de 0 a 9, com um dígito em cada byte.
Agrupado BCD	Representação de byte agrupado de dois dígitos BCD; valor no intervalo de 0 a 99.
Ponteiro near	Um endereço efetivo de 16, 32 ou 64 bits, que representa o deslocamento dentro de um segmento. Usado para todos os ponteiros em uma memória não segmentada e para referências dentro de um segmento em uma memória segmentada.
Ponteiro far	Um endereço lógico consistindo em um seletor de segmento de 16 bits e um deslocamento de 16, 32 ou 64 bits. Ponteiros far são usados para referência à memória em um modelo de memória segmentado, onde a identidade de um segmento sendo acessado precisa ser especificada explicitamente.
Campo de bits	Uma sequência contígua de bits em que a posição de cada bit é considerada como uma unidade independente. Uma *string* de bits pode começar em qualquer posição de bit de qualquer byte e pode conter até 32 bits.
Cadeia de bits	Uma sequência contígua de bits, contendo de zero a $2^{32} - 1$ bits.
Cadeia de bytes	Uma sequência contígua de bytes, palavras ou *doublewords*, contendo de zero a $2^{32} - 1$ bytes.
Ponto flutuante	Ver Figura 10.4.
SIMD agrupada (do inglês *single instruction, multiple data* — única instrução, múltiplos dados)	Tipos de dados de 64 e 128 bits agrupados.

Formatos de dados numéricos do x86



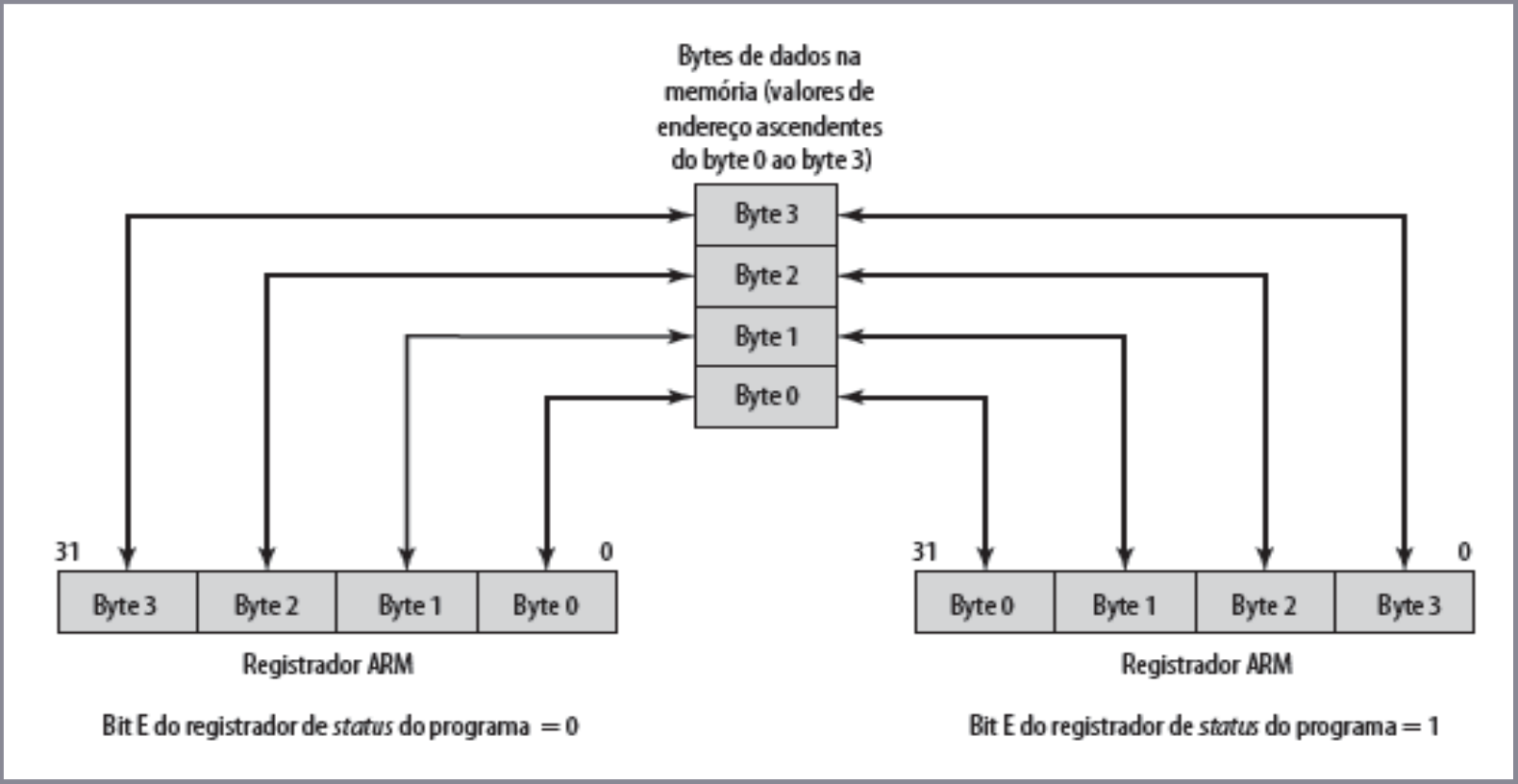
Tipos de dados do ARM

- 8 (byte), 16 (meia-palavra), 32 (palavra) bits.
- Acessos de meia-palavra e palavra devem ser alinhados por palavra.
- Alternativas de acesso não alinhado.
  - Default:
    - Tratado como truncado.
    - Bits[1:0] tratado como zero para word.
    - Bit[0] tratado como zero para meia-palavra.
    - Carrega instruções de única palavra, gira para direita dados alinhados por palavra transferidos por endereço não alinhado por palavra com um, dois ou três bytes.
    - Verificação de alinhamento.
  - Sinal de abortar dados indica falta de alinhamento para tentar acesso desalinhado.
  - Acesso desalinhado.
  - Processador usa um ou mais acessos à memória para gerar transferência de bytes adjacentes de forma transparente ao programador.

- Interpretação de inteiro desalinhado aceita para todos os tipos.
- Interpretação de inteiro com sinal de complemento a dois aceita para todos os tipos.
- Maioria das implementações não oferece hardware de ponto flutuante.
  - Economiza energia e superfície.
  - Aritmética de ponto flutuante implementada no software.
  - Coprocessador de ponto flutuante.
  - Tipos de dados de ponto flutuante IEEE 754 de precisão simples e dupla.

Suporte a endian no ARM

- Bit E no registrador de controle do sistema.
- Sob controle do programa.



Tipos de operação

- Transferência de dados.
- Aritmética.
- Lógica.
- Conversão.
- E/S.
- Controle do sistema.
- Transferência de controle.

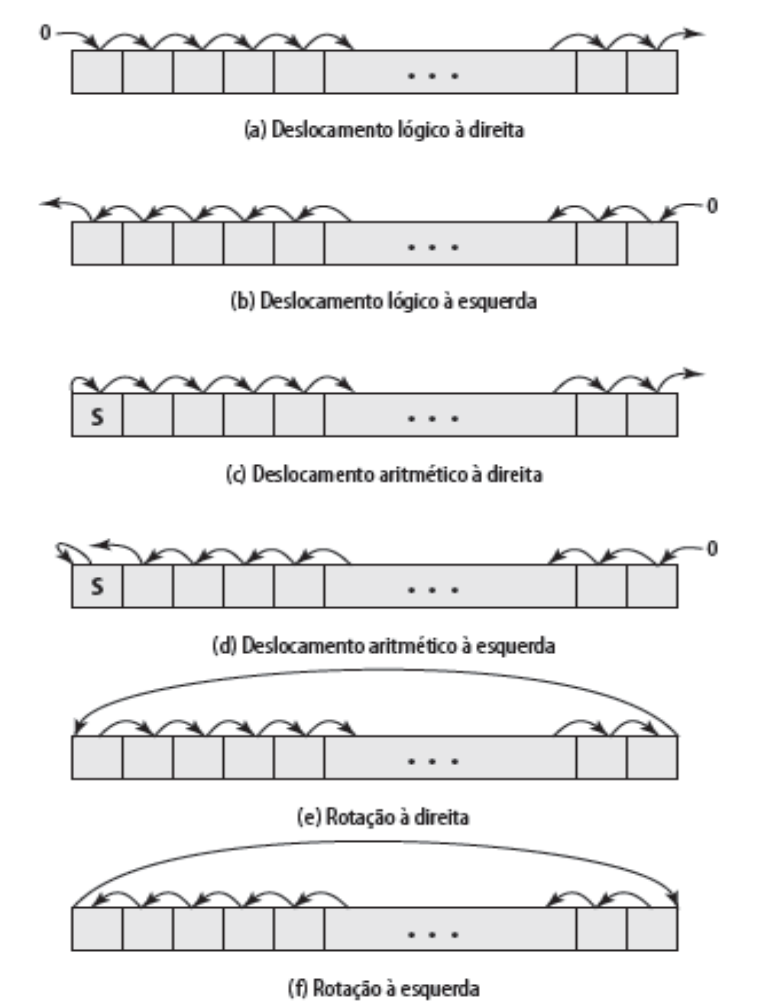
Transferência de dados

- Especificam:
  - Origem.
  - Destino.
  - Quantidade de dados.
- Podem ser instruções diferentes para diferentes movimentações.
  - P.e., IBM 370.
- Ou uma instrução e diferentes endereços.
  - P.e., VAX.

Aritmética

- Adição, Subtração, Multiplicação, Divisão.
- Inteiro com sinal.
- Ponto flutuante?
- Pode incluir:
  - Incremento (a++).
  - Decremento (a--).
  - Negação (-a).

Operações de deslocamento e rotação



Lógica

- Operações bit a bit.
- AND, OR, NOT.

Conversão

- P.e., binário para decimal.

Entrada/saída

- Podem ser instruções específicas.
- Pode ser feita usando instruções de movimentação de dados (mapeadas na memória).
- Pode ser feita por um controlador separado (DMA).

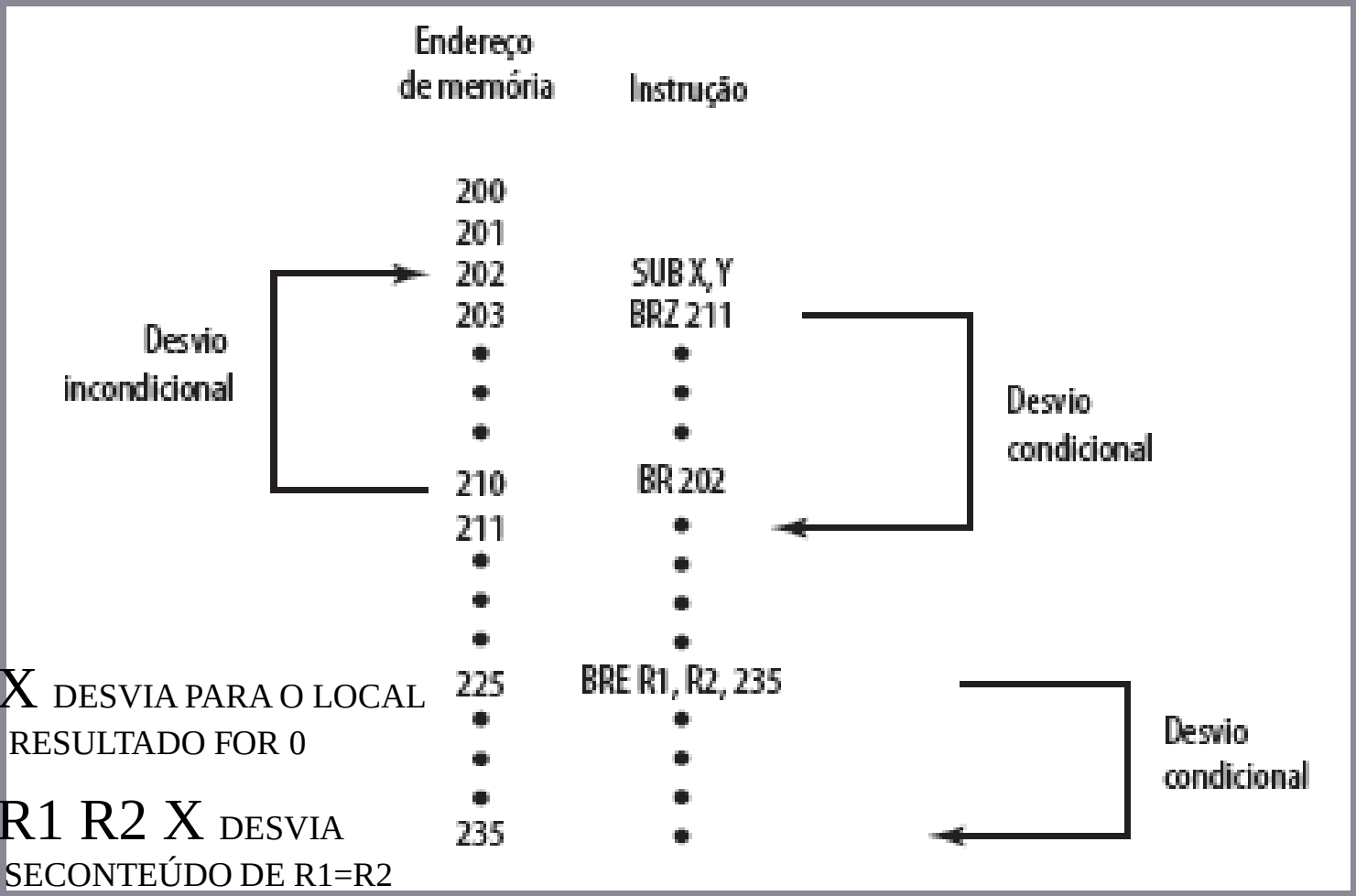
Controle do sistema

- Instruções privilegiadas.
- CPU precisa estar em estado específico:
  - Anel 0 no 80386+.
  - Modo kernel.
- Para uso dos sistemas operacionais.

Transferência de controle

- Desvio:
  - P.e., desvio para x se resultado for zero.
- Salto:
  - P.e., incrementa e salta se for zero.
  - Desvia xxxx.
  - ADD A.
- Chamada de sub-rotina:
  - chamada de interrupção.

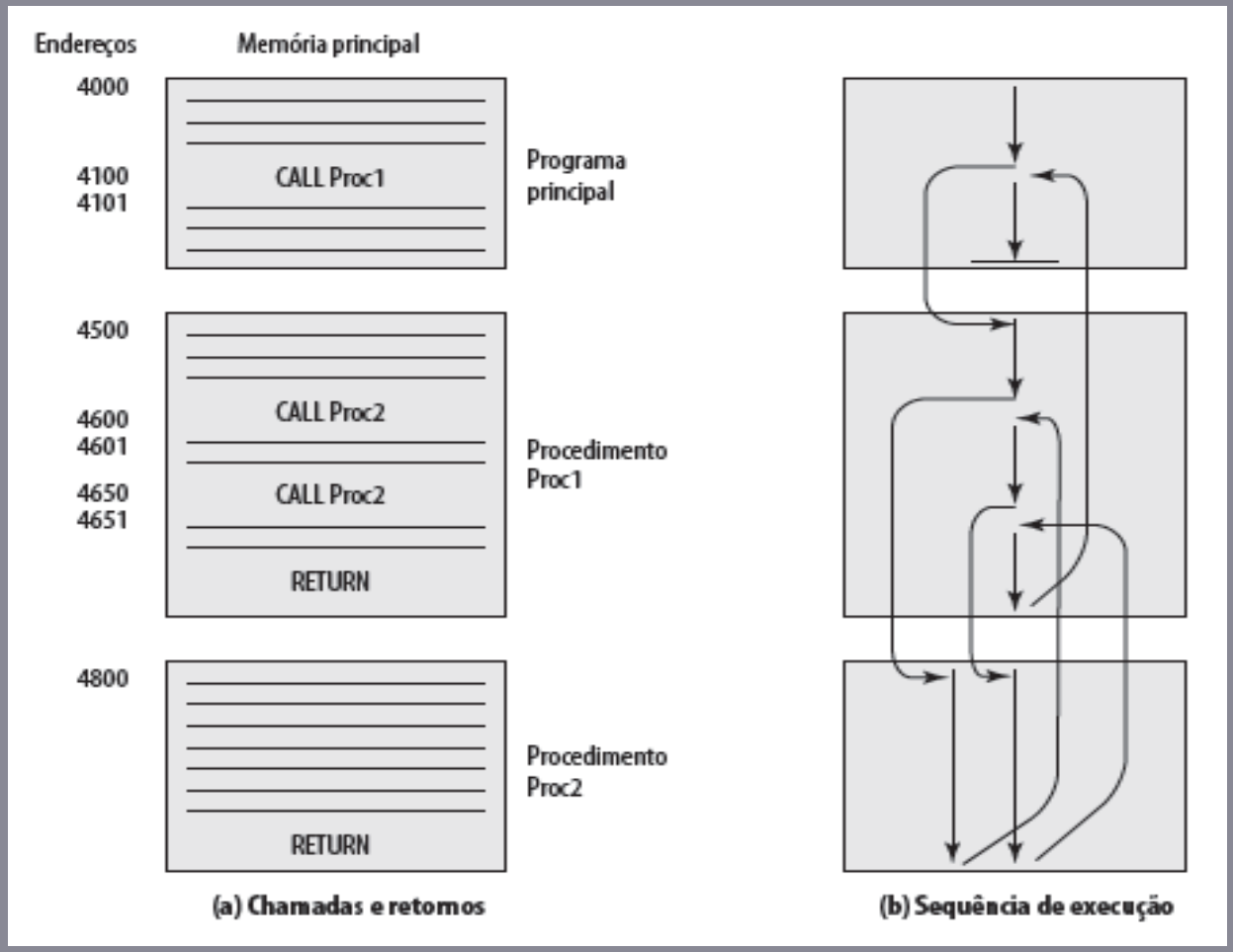
Instrução de desvio



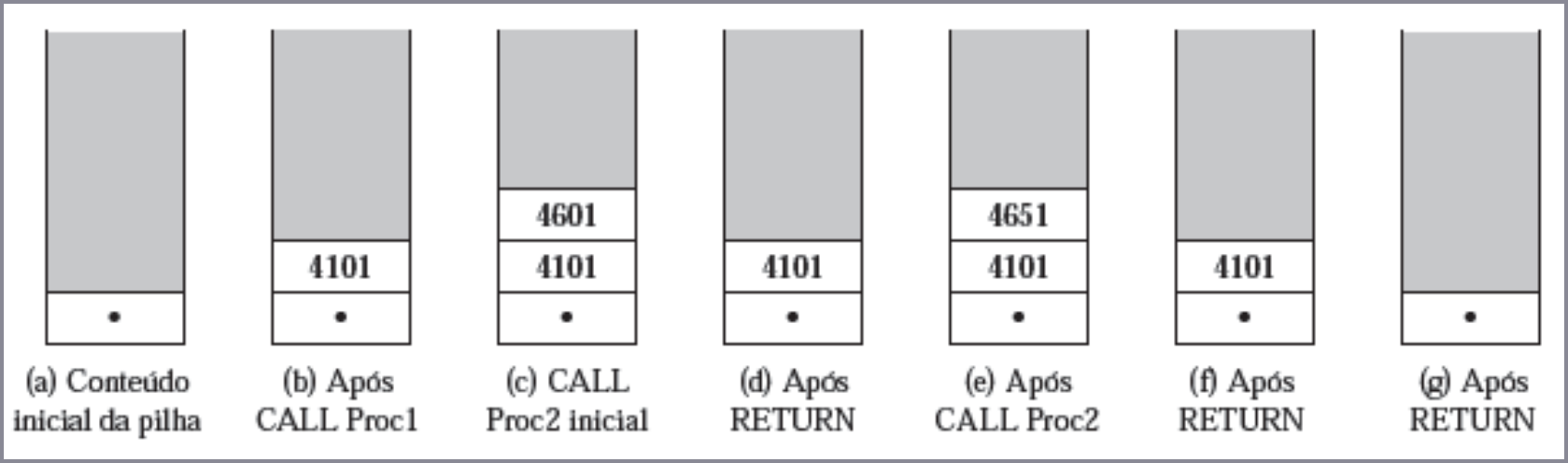
BRZ X DESVIA PARA O LOCAL X SE RESULTADO FOR 0

BRE R1 R2 X DESVIA PARA X SECONTEÚDO DE R1=R2

Chamadas de procedimento aninhadas



Uso da pilha



(a) Conteúdo inicial da pilha | (b) Após CALL Proc1 | (c) CALL Proc2 inicial | (d) Após RETURN | (e) Após CALL Proc2 | (f) Após RETURN | (g) Após RETURN

Crescimento do frame de pilha usando procedimentos de exemplo P e Q

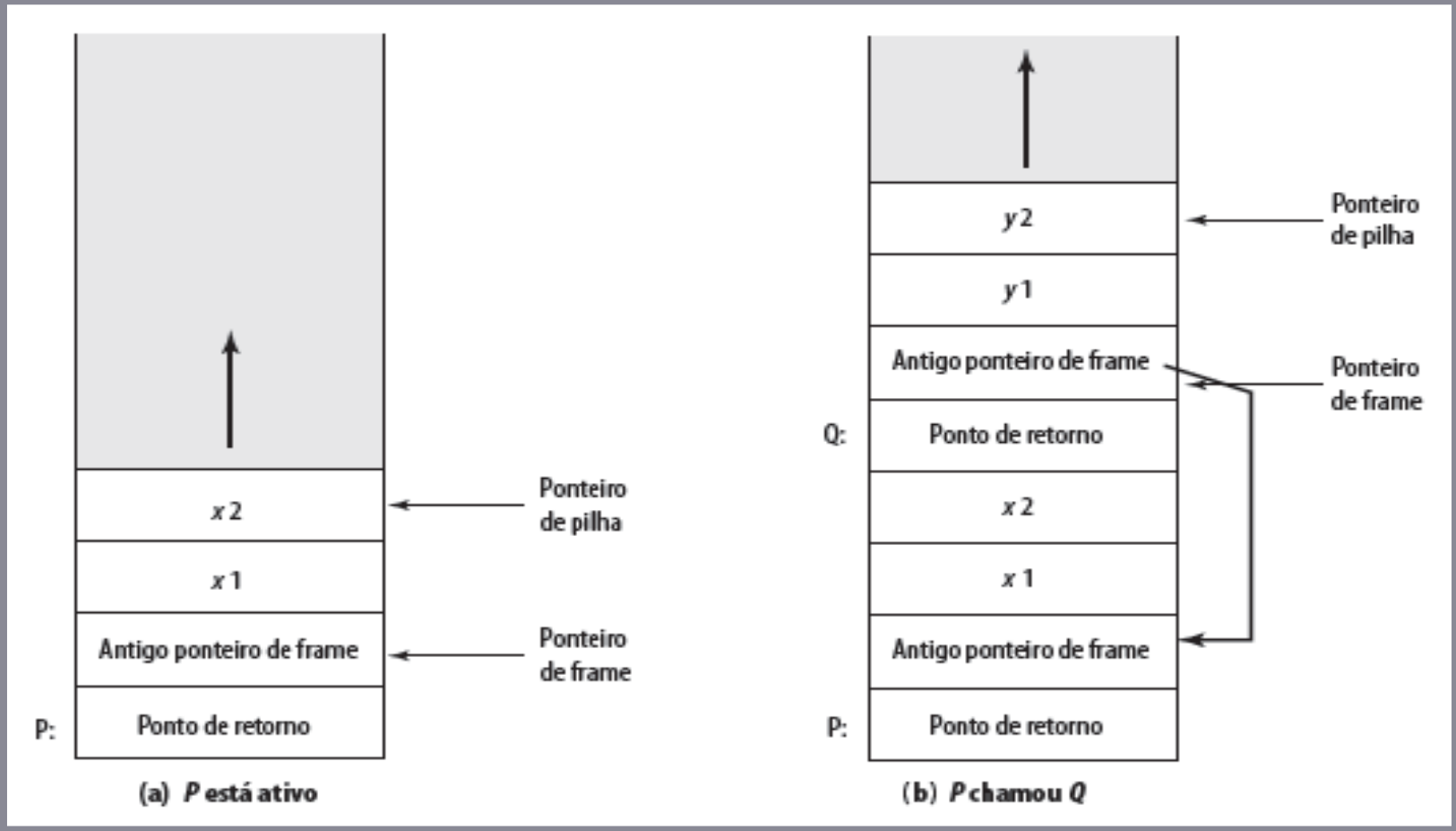
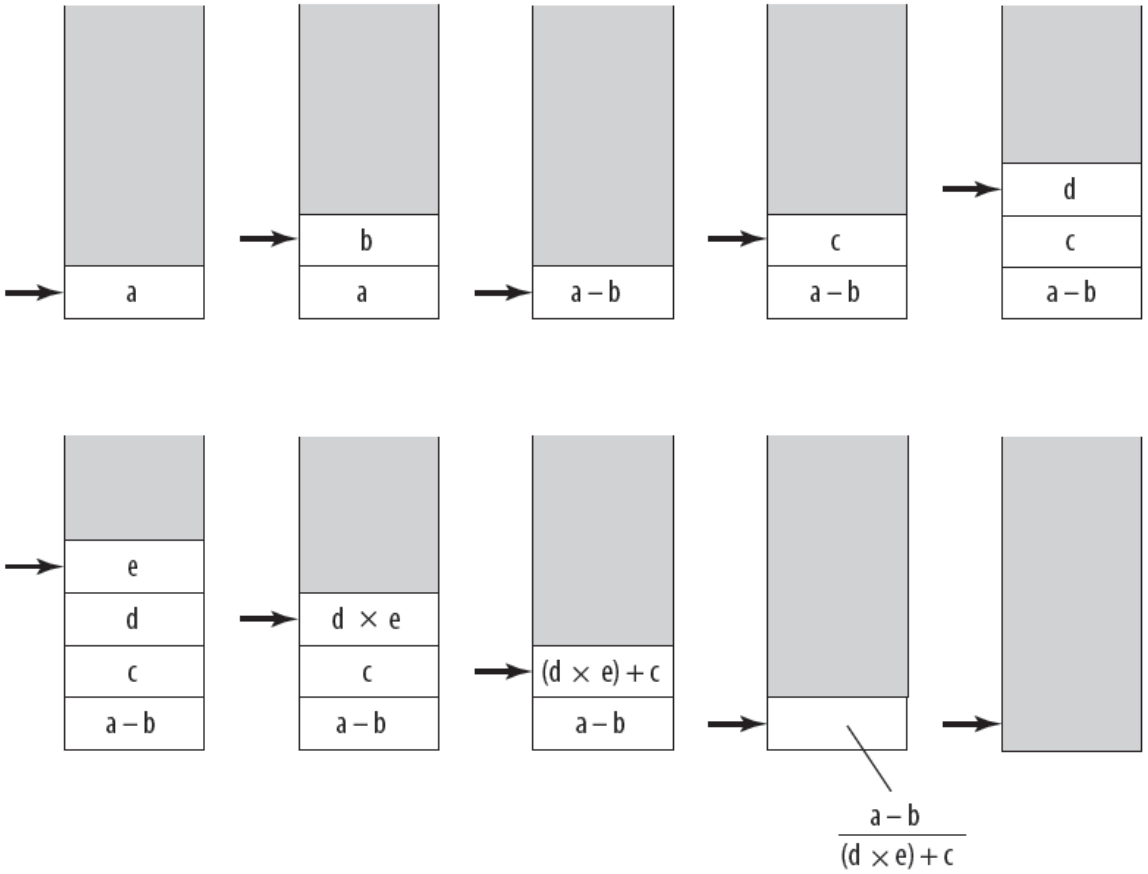


Figura 10.16 Uso da pilha para calcular f = (a – b)/[(d × e) + c]



Tipos de operação x86

Tabela 10.8 Tipos de operação do x86 (com exemplos de operações típicas)

Instrução	Descrição
Movimentação de dados	
MOV	Operação de movimentação entre registradores ou entre registrador e memória.
PUSH	Coloca operando na pilha.
PUSHA	Coloca todos os registradores na pilha.
MOVSX	Move byte, palavra, palavra dupla, com extensão de sinal. Move um byte para uma palavra, ou uma palavra para uma palavra dupla, com extensão de sinal para complemento de dois.
LEA	Carrega endereço efetivo. Carrega o deslocamento do operando de origem, em vez do seu valor, ao operando de destino.
XLAT	Tradução de tabela de pesquisa. Substitui um byte em AL por um byte de uma tabela de tradução codificada pelo usuário. Quando XLAT é executada, AL deverá ter um índice sem sinal para a tabela. XLAT muda o conteúdo de AL do índice de tabela para a entrada de tabela.
IN, OUT	Operando de entrada, saída do espaço de E/S.
Aritméticas	
ADD	Adiciona operandos.
SUB	Subtrai operandos.
MUL	Multiplicação de inteiro sem sinal, com operandos de byte, palavra ou palavra dupla, e resultado de palavra, palavra dupla ou quatro palavras.
IDIV	Divisão com sinal.
Lógicas	
AND	AND dos operandos.
BTS	Teste e definição de bit. Opera sobre um operando de campo de bit. A instrução copia o valor atual de um bit para o flag CF e define o bit original como 1.
BSF	Varredura de bit adiante. Varre uma palavra ou palavra dupla para o bit 1 e armazena o número do primeiro bit 1 em um registrador.
SHL/SHR	Deslocamento lógico à esquerda ou à direita.
SAL/SAR	Deslocamento aritmético à esquerda ou à direita.
ROL/ROR	Rotação à esquerda ou à direita.
SETcc	Define um byte como zero ou 1, dependendo de qualquer uma das 16 condições definidas pelos flags de status.

(*Continua*)

(*Continuação*)

Transferência de controle	
JMP	Salto incondicional.
CALL	Transfere o controle para outro local. Antes da transferência, o endereço da instrução após o CALL é colocado na pilha.
JE/JZ	Salto se igual/zero.
LOOPE/LOOPZ	Loop se igual/zero. Esse é um salto condicional usando um valor armazenado no registrador ECX. A primeira instrução decrementa ECX antes de testá-lo para a condição de desvio.
INT/INTO	Interrompe/Interrompe se houver *overflow*. Transfere o controle para uma rotina de serviço de interrupção.
Operações de *string*	
MOVS	Move *string* de byte, palavra, palavra dupla. A instrução opera sobre um elemento de uma *string*, indexado pelos registradores ESI e EDI. Após cada operação de *string*, os registradores são automaticamente incrementados ou decrementados para apontarem para o próximo elemento da *string*.
LODS	Carrega byte, palavra ou palavra dupla de *string*.
Suporte a linguagem de alto nível	
ENTER	Cria um frame de pilha que pode ser usado para implementar as regras de uma linguagem de alto nível estruturada em bloco.
LEAVE	Reverte a ação de um ENTER anterior.
BOUND	Verifica limites de *array*. Verifica se o valor no operando 1 está dentro dos limites inferior e superior. Os limites estão em dois locais de memória adjacentes referenciados pelo operando 2. Uma interrupção ocorre se o valor estiver fora dos limites. Essa instrução é usada para verificar um índice de *array*.
Controle de flag	
STC	Define flag de *carry*.
LAHF	Carrega registro AH a partir dos flags. Copia bits SF, ZF, AF, F e CF para o registrador A.

(*Continua*)

(*Continuação*)

Registrador de segmento	
LDS	Carrega ponteiro para DS e para outro registrador.
Controle do sistema	
HLT	Termina a execução do programa.
LOCK	Ativa uma suspensão na memória compartilhada de modo que o Pentium tenha uso exclusivo dela durante a instrução que vem imediatamente após o LOCK.
ESC	Escape para uma extensão do processador. Um código de escape que indica que as instruções seguintes devem ser executadas por um processador numérico que admite cálculos com inteiro e ponto flutuante de alta precisão.
WAIT	Espera até BUSY# negado. Suspende a execução do programa no Pentium até que o processador detecte que o pino BUSY esteja inativo, indicando que o coprocessador numérico terminou a execução.
Proteção	
SGDT	Armazena tabela de descritor global.
LSL	Carrega limite de segmento. Carrega um registrador especificado pelo usuário com um limite de segmento.
VERR/VERW	Verifica segmento para leitura/gravação.
Gerenciamento de cache	
INVD	Esvazia a memória cache interna.
WBINVD	Esvazia a memória cache interna após gravar linhas modificadas na memória.
INVLPG	Invalida uma entrada no *translation lookaside buffer* (TLB).

INSTRUÇÕES MMX

Tabela 10.11 Conjunto de instruções MMX

Categoria	Instrução	Descrição
Aritmética	PADD [B, W, D]	Adição paralela de oito pacotes de bits, quatro palavras de 16 bits ou duas palavras duplas de 32 bits, com contorno.
	PADDS [B, W]	Adição com saturação.
	PADDUS [B, W]	Adição sem sinal com saturação.
	PSUB [B, W, D]	Subtração com contorno.
	PSUBS [B, W]	Subtração com saturação.
	PSUBUS [B, W]	Subtração sem sinal com saturação.
	PMULHW	Multiplicação paralela de quatro palavras de 16 bits com sinal, com 16 bits de alta ordem do resultado de 32 bits escolhidos.
	PMULLW	Multiplicação paralela de quatro palavras de 16 bits com sinal, com 16 bits de baixa ordem do resultado de 32 bits escolhidos.
	PMADDWD	Multiplicação paralela de quatro palavras de 16 bits com sinal; soma pares adjacentes de resultados de 32 bits
Comparação	PCMPEQ [B, W, D]	Comparação paralela de igualdade; resultado é máscara de 1s se verdadeiro ou 0s se falso.
	PCMPGT [B, W, D]	Comparação paralela de maior; resultado é máscara de 1s se verdadeiro ou 0s se falso.

(*Continua*)

(*Continuação*)

Conversão	PACKUSWB	Agrupa palavras em bytes com saturação sem sinal.
	PACKSS [WB, DW]	Agrupa palavras em bytes, ou palavras duplas em palavras, com saturação com sinal.
	PUNPCKH [BW, WD, DQ]	Desagrupa em paralelo (mesclagem intervalada) bytes, palavras ou palavras duplas de alta ordem do registrador MMX.
	PUNPCKL [BW, WD, DQ]	Desagrupa em paralelo (mesclagem intervalada) bytes, palavras ou palavras duplas de baixa ordem do registrador MMX.
Lógica	PAND	AND lógico bit a bit com 64 bits.
	PNDN	AND NOT lógico bit a bit com 64 bits.
	POR	OR lógico bit a bit com 64 bits.
	PXOR	XOR lógico bit a bit com 64 bits.
Deslocamento	PSLL [W, D, Q]	Deslocamento lógico paralelo à esquerda de pacotes de palavra, palavras duplas ou quatro palavras pela quantidade especificada no registrador MMX ou valor imediato.
	PSRL [W, D, Q]	Deslocamento lógico paralelo à direita de pacotes de palavra, palavras duplas ou quatro palavras agrupadas.
	PSRA [W, D]	Deslocamento aritmético paralelo à direita de pacotes de palavra, palavras duplas ou quatro palavras.
Transferência de dados	MOV [D, Q]	Move palavras duplas ou quatro palavras de/para registrador MMX.
Ger. de estado	EMMS	Esvazia estado MMX (esvazia bits de tag dos registradores FP).

Nota: se uma instrução aceitar vários tipos de dados [byte (B), palavra (W), *doubleword* (D), *quadword* (Q)], os tipos de dados são indicados entre colchetes.

Exercício para o leitor

- Descubra sobre o conjunto de instruções do Pentium e ARM.
- Comece com Stallings.
- Visite *Web sites*.