Aula 09: Arquitetura de Computadores – Conjuntos de instruções: características e funções & modos e formatos de endereçamento

Prof. Hugo Puertas de Araújo
hugo.puertas@ufabc.edu.br
Sala: 509-2 (5º andar / Torre 2)

Arquitetura de Computadores

Objetivos de aprendizagem

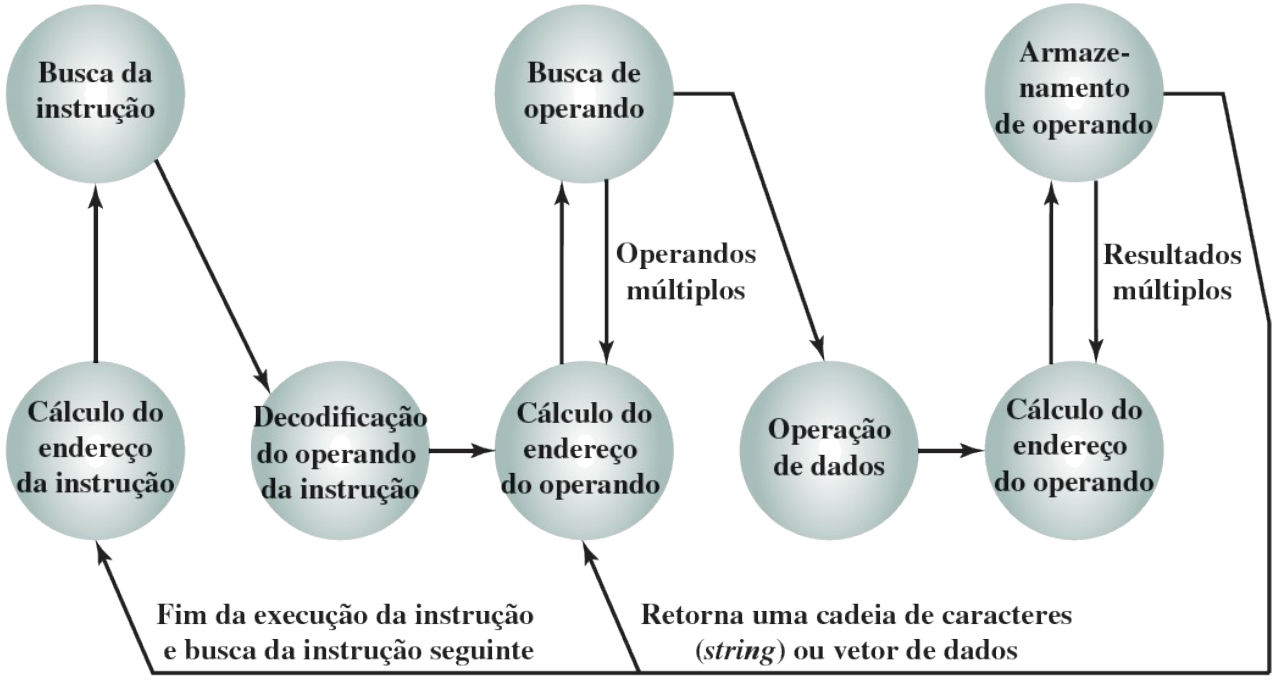
- Apresentar as características das instruções de máquina.
- Descrever os tipos de operandos usados nos conjuntos típicos de instruções de máquina.
- Apresentar uma visão geral dos tipos de dados de x86 e ARM.
- Descrever os tipos de operandos aceitos pelos conjuntos típicos de instruções de máquina.
- Compreender as diferenças entre big-endian, little-endian e bi-endian.

Objetivos de aprendizagem (cont.)

- Descrever os vários tipos de modos de endereçamento comuns nos conjuntos de instruções.
- Apresentar uma visão geral dos modos de endereçamento do x86 e do ARM.
- Resumir as questões e relações envolvidas no desenvolvimento de um formato de instrução.
- Compreender a diferença entre a linguagem de máquina e a linguagem de montagem.

Características das instruções de máquina

- A figura abaixo mostra as etapas envolvidas na execução da instrução e, por consequência, define os elementos de uma instrução de máquina:



Características das instruções de máquina

- Esses elementos são os seguintes:
  - **Código de operação**: especifica a operação a ser realizada (por exemplo, ADD, E/S).
  - **Referência a operando fonte**: a operação pode envolver um ou mais operandos fontes.
  - **Referência a operando de resultado**: a operação deve produzir um resultado.
  - **Referência à próxima instrução**: isso diz ao processador onde buscar a próxima instrução depois que a execução dessa instrução estiver completa.

Características das instruções de máquina

- Operandos fonte e resultado podem estar em uma destas áreas:
  - **Memória principal ou virtual**: o endereço da memória principal ou virtual deve ser fornecido.
  - **Registradores do processador**: com raras exceções, um processador contém um ou mais registradores que podem ser referenciados por instruções de máquina.
  - **Imediato**: o valor do operando está contido em um campo na instrução sendo executada.
  - **Dispositivo de E/S**: a instrução precisa especificar o módulo e o dispositivo de E/S para a operação.

Representação da instrução

- Os **opcodes** são representados por abreviações, chamadas mnemônicos, que indicam a operação.
- Alguns exemplos comuns são:
  - **ADD**: Adição
  - **SUB**: Subtração
  - **MUL**: Multiplicação
  - **DIV**: Divisão
  - **LOAD**: Carrega dados da memória
  - **STOR**: Armazena dados na memória

Tipos de instrução

- Se considerarmos um conjunto simples de instruções de máquina, essa operação poderia ser feita com três instruções:
  i. Carregue um registrador com o conteúdo do local de memória 513.
  ii. Some o conteúdo do local de memória 514 ao registrador.
  iii. Armazene o conteúdo do registrador no local de memória 513.
- Como podemos ver, uma única instrução em BASIC pode exigir três instruções de máquina.

Tipos de instrução

- As **instruções aritméticas** oferecem capacidades de cálculo para o processamento de **dados numéricos**.
- As **instruções lógicas** (booleanas) oferecem capacidades de processamento de **qualquer outro tipo de dado** que o usuário possa querer empregar.
- As **instruções de memória** existem para mover dados entre a **memória** e os **registradores**.
- As **instruções de E/S** são necessárias para transferir **programas e dados** para a **memória** e os resultados de cálculos **de volta ao usuário**.

Tipos de instrução

- As **instruções de teste** são usadas para testar o valor de uma palavra de dados ou o estado de um cálculo.
- As **instruções de desvio** são então usadas para desviar para um conjunto de instruções diferente, dependendo da decisão tomada.
- O **conjunto de instruções** é o meio de o programador controlar o processador.
- Dessa maneira, os requisitos do programador devem ser considerados no projeto do conjunto de instruções.

Projeto do conjunto de instruções

- As questões básicas mais importantes de projeto são as seguintes:
  - **Repertório de operações**: quantas e quais operações oferecer.
  - **Tipos de dados**: os diversos tipos de dados sobre os quais as operações são realizadas.
  - **Formato de instrução**: tamanho da instrução, número de endereços, tamanho dos diversos campos, e assim por diante.
  - **Registradores**: número de registradores do processador que podem ser referenciados pelas instruções e seu uso.
  - **Endereçamento**: o modo ou os modos pelos quais o endereço de um operando é especificado.

Projeto do conjunto de instruções

- As questões básicas mais importantes de projeto são as seguintes:
  - **Repertório de operações**: quantas e quais operações oferecer.
  - **Tipos de dados**: os diversos tipos de dados sobre os quais as operações são realizadas.
  - **Formato de instrução**: tamanho da instrução, número de endereços, tamanho dos diversos campos, e assim por diante.
  - **Registradores**: número de registradores do processador que podem ser referenciados pelas instruções e seu uso.
  - **Endereçamento**: o modo ou os modos pelos quais o endereço de um operando é especificado.

E.: Arquitetura ou Organização do computador?

Tipos de operandos

- As instruções de máquina operam sobre dados.
- As categorias gerais de dados mais importantes são:
  i. **Endereços.**
  ii. **Números.**
  iii. **Caracteres.**
  iv. **Dados lógicos.**

Números

- Todas as linguagens de máquina incluem tipos de dados numéricos.
- Até mesmo no processamento de dados não numéricos, existe a necessidade de os números atuarem como contadores, tamanhos de campo e assim por diante.
- Três tipos de dados numéricos são comuns nos computadores:
  i. **Inteiros binários ou ponto fixo binário.**
  ii. **Ponto flutuante binário.**
  iii. **Decimal.**

Caracteres

- Uma forma de dado comum é o texto, ou strings de caracteres.
- Diversos códigos foram elaborados, nos quais os caracteres são representados por uma sequência de bits.
- Talvez o exemplo comum mais antigo seja o **código Morse**.
- Hoje, o código de caracteres mais utilizado é o International Reference Alphabet (**IRA**), mais conhecido como ASCII (versão americana).
- Cada caractere nesse código é representado por um padrão exclusivo de 7 bits; dessa maneira, 128 caracteres diferentes podem ser representados.

Dados lógicos

- Em geral, cada palavra ou outra unidade endereçável é tratada como uma única unidade de dados.
- Todavia, às vezes é útil considerar que uma unidade de n bits consista em n itens de dados de 1 bit, com cada item tendo o valor 0 ou 1.
- Quando os dados são vistos dessa forma, eles são considerados **dados lógicos**.
- O “tipo” de uma unidade de dados é determinado pela operação que está sendo realizada sobre ele.

Tipos de dados do Intel x86 e do ARM

- Tipos de dados do x86:

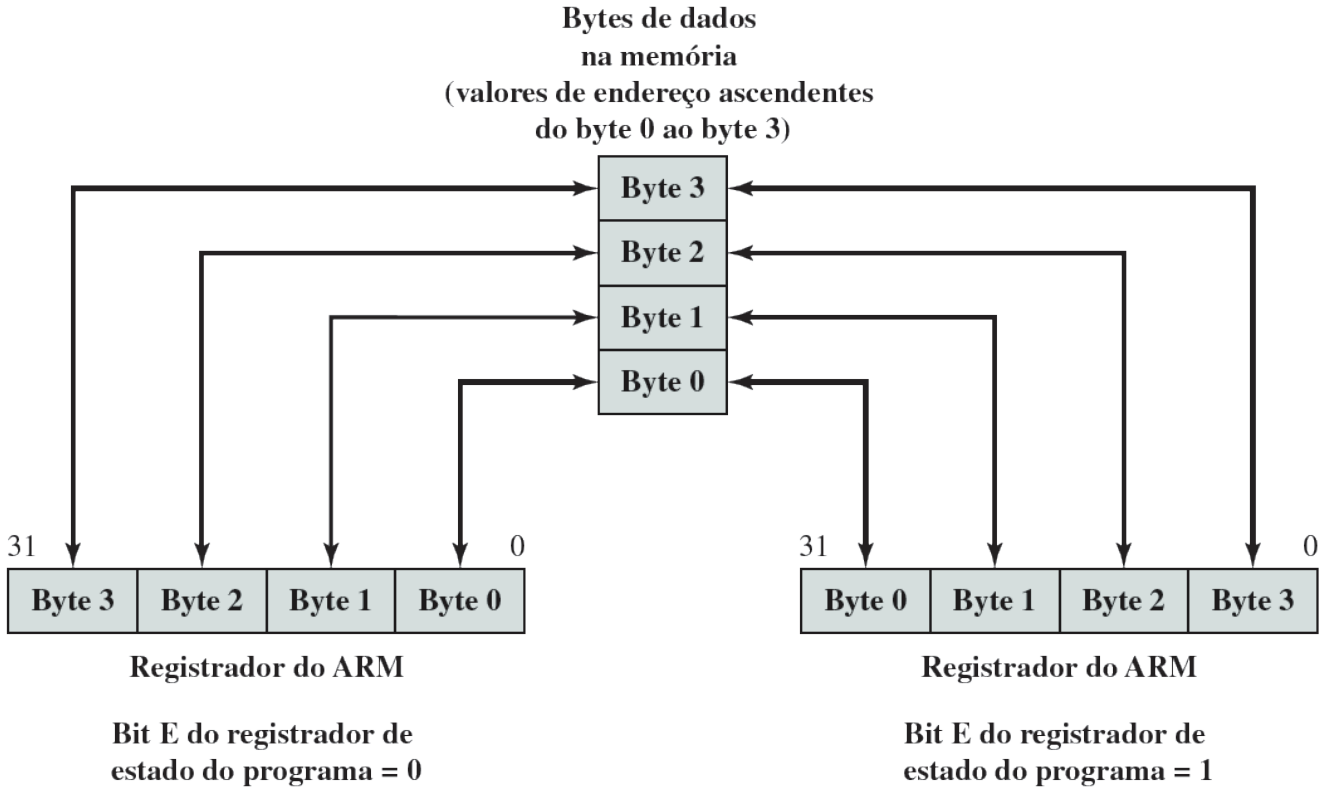
Tipo de dados	Descrição
Geral	Byte, palavra (16 bits), palavras duplas (32 bits), quatro palavras (64 bits) e quatro palavras duplas (128 bits) com conteúdo binário arbitrário.
Inteiros	Um valor binário com sinal, contido em um byte, palavra ou palavras duplas, usando a representação de complemento de dois.
Ordinais	Um inteiro sem sinal contido em um byte, palavra ou palavras duplas.
Números em BCD (*Binary Coded Decimal*) não empacotado	Uma representação de um dígito BCD no intervalo de 0 a 9, com um dígito em cada byte.
BCD empacotado	Representação de byte empacotado 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, em que 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 uma unidade independente. Uma *string* de bits pode começar em qualquer posição de bit de qualquer byte e pode conter até 32 bits.
String de bits	Uma sequência contígua de bits, contendo de zero a $2^{23} - 1$ bits.
String de bytes	Uma sequência contígua de bytes, palavras ou palavras duplas, contendo de zero a $2^{23} - 1$ bytes.
Ponto flutuante	Ver Figura 12.4
SIMD empacotada (do inglês, *Single Instruction*, *Multiple Data* — única instrução, múltiplos dados)	Tipos de dados de 64 e 128 bits agrupados.

Tipos de dados do Intel x86 e do ARM

- **Processadores ARM** admitem tipos de dados de 8 (**byte**), 16 (meia-palavra — **halfword**) e 32 (palavra — **word**) bits de tamanho.
- O acesso de meia-palavra deve ser alinhado por meia-palavra e os acessos de palavra precisam ser alinhados por palavra.
- Para os três tipos de dados, uma interpretação sem sinal é admitida, em que o valor representa um inteiro sem sinal e não negativo.
- Eles também podem ser usados para os **inteiros com sinal** em **complemento de dois**.

Tipos de dados do Intel x86 e do ARM

- Suporte a endian no ARM — Load/Store de palavra com o bit E:



Tipos de operações

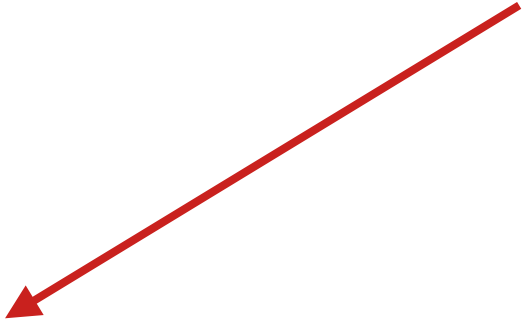
- Os mesmos tipos gerais de operações são encontrados em todas as máquinas.
- Uma categorização prática e comum é a seguinte:
  i. **Transferência de dados.**
  ii. **Aritmética.**
  iii. **Lógica.**
  iv. **Conversão.**
  v. **E/S.**
  vi. **Controle de sistema.**
  vii. **Transferência de controle.**

Tipos de operações

- Os mesmos tipos gerais de operações são encontrados em todas as máquinas.
- Uma categorização prática e comum é a seguinte:
  i. Transferência de dados.
  ii. Aritmética.
  iii. Lógica.
  iv. Conversão.
  v. E/S.
  vi. Controle de sistema.
  vii. Transferência de controle.

Exercício

E.: A que isso se refere? Opera sobre qual registrador?



Tipos de operações

- Operações comuns do conjunto de instruções:

Tipo	Nome da operação	Descrição
Transferência de dados	Move (transferência)	Transfere palavra ou bloco da origem ao destino
	Store (armazenamento)	Transfere palavra do processador para a memória
	Load (busca)	Transfere palavra da memória para o processador
	Exchange (troca)	Troca o conteúdo da origem e do destino
	Clear (reset)	Transfere palavra de 0s para o destino
	Set	Transfere palavra de 1s para o destino
	Push	Transfere palavra da origem para o topo da pilha
	Pop	Transfere palavra do topo da pilha para o destino

Tipos de operações

- Operações comuns do conjunto de instruções:

Tipo	Nome da operação	Descrição
Aritmética	Soma	Calcula a soma de dois operandos
	Subtração	Calcula a diferença de dois operandos
	Multiplicação	Calcula o produto de dois operandos
	Divisão	Calcula o quociente de dois operandos
	Absoluto	Substitui o operando pelo seu valor absoluto
	Negativo	Troca o sinal do operando
	Incremento	Soma 1 ao operando
	Decremento	Subtrai 1 do operando

Tipos de operações

- Operações comuns do conjunto de instruções:

Tipo	Nome da operação	Descrição
Lógica	AND	Realiza o AND lógico
	OR	Realiza o OR lógico
	NOT (complemento)	Realiza o NOT lógico
	Exclusive-OR	Realiza o XOR lógico
	Test	Testa condição especificada; define flag(s) com base no resultado
	Compare	Faz comparação lógica ou aritmética de dois ou mais operandos; define flag(s) com base no resultado
	Definir variáveis de controle	Classe de instruções para definir controles para fins de proteção, tratamento de interrupção, controle de tempo etc.
	Shift	Desloca o operando para a esquerda (direita), introduzindo constantes na extremidade
	Rotate	Desloca ciclicamente o operando para a esquerda (direita), de uma extremidade à outra

Tipos de operações

- Operações comuns do conjunto de instruções:

Tipo	Nome da operação	Descrição
Transferência de controle	Jump (desvio)	Transferência incondicional; carrega PC com endereço especificado
	Jump condicional	Testa condição especificada; ou carrega PC com endereço especificado ou não faz nada, com base na condição
	Jump para sub-rotina	Coloca informação do controle do programa atual em local conhecido; salta para endereço especificado
	Return	Substitui conteúdo do PC por outro registrador de local conhecido
	Execute	Busca operando do local especificado e executa como instrução; não modifica o PC
	Skip	Incrementa o PC para saltar para a próxima instrução
	Skip condicional	Testa condição especificada; ou salta ou não faz nada, com base na condição
	Halt	Termina a execução do programa
	Wait (hold)	Termina a execução do programa; testa condição especificada repetidamente; retoma a execução quando a condição for satisfeita
	No operation	Nenhuma operação é realizada, mas a execução do programa continua

Tipos de operações

- Operações comuns do conjunto de instruções:

Tipo	Nome da operação	Descrição
Entrada/ saída	Input (leitura)	Transfere dados da porta de E/S ou dispositivo especificado para o destino (por exemplo, memória principal ou registrador do processador)
	Output (escrita)	Transfere dados da origem especificada para porta de E/S ou dispositivo
	Start I/O	Transfere instruções para o processador de E/S para iniciar operação de E/S
	Test I/O	Transfere informações de estado do sistema de E/S para destino especificado

Tipos de operações

- Operações comuns do conjunto de instruções:

Tipo	Nome da operação	Descrição
Conversão	Translate	Traduz valores em uma seção da memória com base em uma tabela de correspondências
	Convert	Converte o conteúdo de uma palavra de uma forma para outra (por exemplo, decimal empacotado para binário)

Tipos de operações

- Operações comuns do conjunto de instruções:

Tipo	Nome da operação	Descrição
Conversão	Translate	Traduz valores em uma seção da memória com base em uma tabela de correspondências
	Convert	Converte o conteúdo de uma palavra de uma forma para outra (por exemplo, decimal empacotado para binário)

E.: O que é isso?

Tipos de operações

- Ações do processador para diversos tipos de operação:

Transferência de dados	Transfere dados de um local para outro
	Se a memória estiver envolvida: Determina o endereço da memória Realiza transformação de endereço de memória virtual para real Verifica cache Inicia leitura/escrita da memória
Aritmética	Pode envolver transferência de dados, antes e/ou depois
	Realiza função na ALU
	Define códigos de condição e flags
Lógica	O mesmo que aritmética
Conversão	Semelhante à aritmética e lógica. Pode envolver lógica especial para realizar conversão
Transferência de controle	Atualiza contador de programa. Para chamada/retorno de sub-rotina, gerencia passagem de parâmetros e ligação
E/S	Envia comando para módulo de E/S
	Se E/S mapeada na memória, determina o endereço mapeado na memória

Tipos de operação do Intel x86 e do ARM

- O x86 oferece um **conjunto complexo** de tipos de **operação**, incluindo uma série de instruções especializadas.
- O x86 oferece quatro instruções para dar suporte à chamada ou ao retorno de procedimento:
  i. **CALL,**
  ii. **ENTER,**
  iii. **LEAVE,**
  iv. **RETURN.**

Tipos de operação do Intel x86 e do ARM

- Quando um novo procedimento é chamado, o seguinte deverá ser realizado na entrada do novo procedimento:
  i. Levar o ponto de retorno para a pilha.
  ii. Levar o ponteiro do frame atual para a pilha.
  iii. Copiar o ponteiro de pilha como o novo valor do ponteiro de frame.
  iv. Ajustar o ponteiro de pilha para alocar um frame.
- Outro conjunto de instruções especializadas lida com a segmentação da memória.

Tipos de operação do Intel x86 e do ARM

- Os flags de estado são bits em registradores especiais que podem ser definidos por certas operações e usados em instruções de desvio condicional.
- O termo código de condição refere-se às configurações de um ou mais flags de estado.
- A tabela a seguir lista os flags de estado usados no x86.
- A tabela seguinte mostra os códigos de condição (combinações de valores de flag de estado) para os quais os opcodes de salto foram definidos.

Tipos de operação do Intel x86 e do ARM

- Flags de estado do x86:

Bit de estado	Nome	Descrição
C	*Carry*	Indica a existência do bit de transporte ou empréstimo (*carry bit* — *vai um*) na posição do bit mais à esquerda após uma operação aritmética. Também modificado por algumas das operações de deslocamento e rotação.
P	Paridade	Paridade do byte menos significativo do resultado de uma operação aritmética ou lógica. 1 indica paridade par; 0 indica paridade ímpar.
A	*Carry* auxiliar	Representa a existência do bit de transporte ou empréstimo (*carry bit* — *vai um*) na posição entre dois bytes após uma operação aritmética ou lógica de 8 bits. Usado na aritmética BCD.
Z	Zero	Indica que o resultado de uma operação aritmética ou lógica é 0.
S	Sinal	Indica o sinal do resultado de uma operação aritmética ou lógica.
O	*Overflow*	Indica um *overflow* aritmético após uma adição ou subtração em aritmética de complemento de dois.

Tipos de operação do Intel x86 e do ARM

- Códigos de condição do x86 para instruções de salto condicional e SETcc:

Símbolo	Condição testada	Comentário
A, NBE	C = 0 AND Z = 0	Acima; Não abaixo ou igual (maior que, sem sinal)
AE, NB, NC	C = 0	Acima ou igual; Não abaixo (maior que ou igual, sem sinal); Sem *carry*
B, NAE, C	C = 1	Abaixo; Não acima ou igual (menor que, sem sinal); *Carry* definido
BE, NA	C = 1 OR Z = 1	Abaixo ou igual; Não acima (menor que ou igual, sem sinal)
E, Z	Z = 1	Igual; Zero (com ou sem sinal)
G, NLE	[(S = 1 AND O = 1) OR (S = 0 AND O = 0)]AND[Z = 0]	Maior que; Não menor que ou igual (com sinal)
GE, NL	(S = 1 AND O = 1) OR (S = 0 AND O = 0)	Maior que ou igual; Não menor que (com sinal)
L, NGE	(S = 1 AND O = 0) OR (S = 0 AND O = 0)	Menor que; Não maior que ou igual (com sinal)
LE, NG	(S = 1 AND O = 0) OR (S = 0 AND O = 1) OR (Z = 1)	Menor que ou igual; Não maior que (com sinal)
NE, NZ	Z = 0	Não igual; Não zero (com ou sem sinal)
NO	O = 0	Sem *overflow*
NS	S = 0	Sem sinal (não negativo)
NP, PO	P = 0	Sem paridade; Paridade ímpar
O	O = 1	*Overflow*
P	P = 1	Paridade; Paridade par
S	S = 1	Sinal (negativo)

Tipos de operação do Intel x86 e do ARM

- A arquitetura ARM oferece uma grande variedade de tipos de operação.
- A seguir estão as principais categorias:
  i. **Instruções load e store**: na arquitetura ARM, somente instruções load e store acessam locais da memória.
  ii. **Instruções de desvio**: o ARM admite uma instrução de desvio que permite um desvio condicional para a frente ou para trás em até 32 MB.

Tipos de operação do Intel x86 e do ARM

- **Instruções de processamento de dados**: essa categoria inclui instruções lógicas (AND, OR, XOR), instruções de adição e subtração, e instruções de teste e comparação.
- **Instruções de multiplicação**: as instruções de multiplicação de inteiros operam sobre operandos de uma palavra ou de meia-palavra e podem produzir resultados normais ou grandes.
- **Instruções paralelas de adição e subtração**: existe um conjunto de instruções paralelas de adição e subtração, em que partes dos dois operandos são operadas em paralelos.

Tipos de operação do Intel x86 e do ARM

- **Instruções de extensão**: existem várias instruções para desagrupar dados, estendendo por sinal ou com zeros, de bytes para meias-palavras ou palavras, ou de meias-palavras para palavras.
- **Instruções de acesso do registrador de estado**: o ARM oferece a capacidade de ler e também escrever em partes do registrador de estado.
- A arquitetura ARM define quatro flags de condição que são armazenados no registrador de estado do programa: N, Z, C e V (Negativo, Zero, Carry e oVerflow).

Tipos de operação do Intel x86 e do ARM

- Condições do ARM para execução de instrução condicional:

Código	Símbolo	Condição testada	Comentário
0000	EQ	Z = 1	Igual
0001	NE	Z = 0	Não igual
0010	CS/HS	C = 1	*Carry* em um/acima ou igual sem sinal
0011	CC/LO	C = 0	*Carry* zerado/abaixo sem sinal
0100	MI	N = 1	Menos/negativo
0101	PL	N = 0	Mais/positivo ou zero
0110	VS	V = 1	*Overflow*
0111	VC	V = 0	Sem *overflow*
1000	HI	C = 1 AND Z = 0	Acima sem sinal
1001	LS	C = 0 OR Z = 1	Abaixo ou igual sem sinal
1010	GE	N = V [(N = 1 AND V = 1) OR (N = 0 AND V = 0)]	Sinalizado maior que ou igual
1011	LT	N ≠ V [(N = 1 AND V = 0) OR (N = 0 AND V = 1)]	Sinalizado menor que
1100	GT	(Z = 0) AND (N = V)	Sinalizado maior que
1101	LE	(Z = 1) OR (N ≠ V)	Sinalizado menor que ou igual
1110	AL	—	Sempre (incondicional)
1111	—	—	Esta instrução só pode ser executada incondicionalmente

Modos de endereçamento

- Modos básicos de endereçamento:

Modo	Algoritmo	Principal vantagem	Principal desvantagem
Imediato	Operando = A	Nenhuma referência à memória	Magnitude de operando limitada
Direto	EA = A	Simples	Espaço de endereçamento limitado
Indireto	EA = (A)	Espaço de endereçamento grande	Múltiplas referências à memória
Por registrador	EA = R	Nenhuma referência à memória	Espaço de endereçamento limitado
Indireto por registrador	EA = (R)	Espaço de endereçamento grande	Referência extra de memória
Por deslocamento	EA = A + (R)	Flexibilidade	Complexidade
De pilha	EA = topo da pilha	Nenhuma referência à memória	Aplicabilidade limitada

Modos de endereçamento

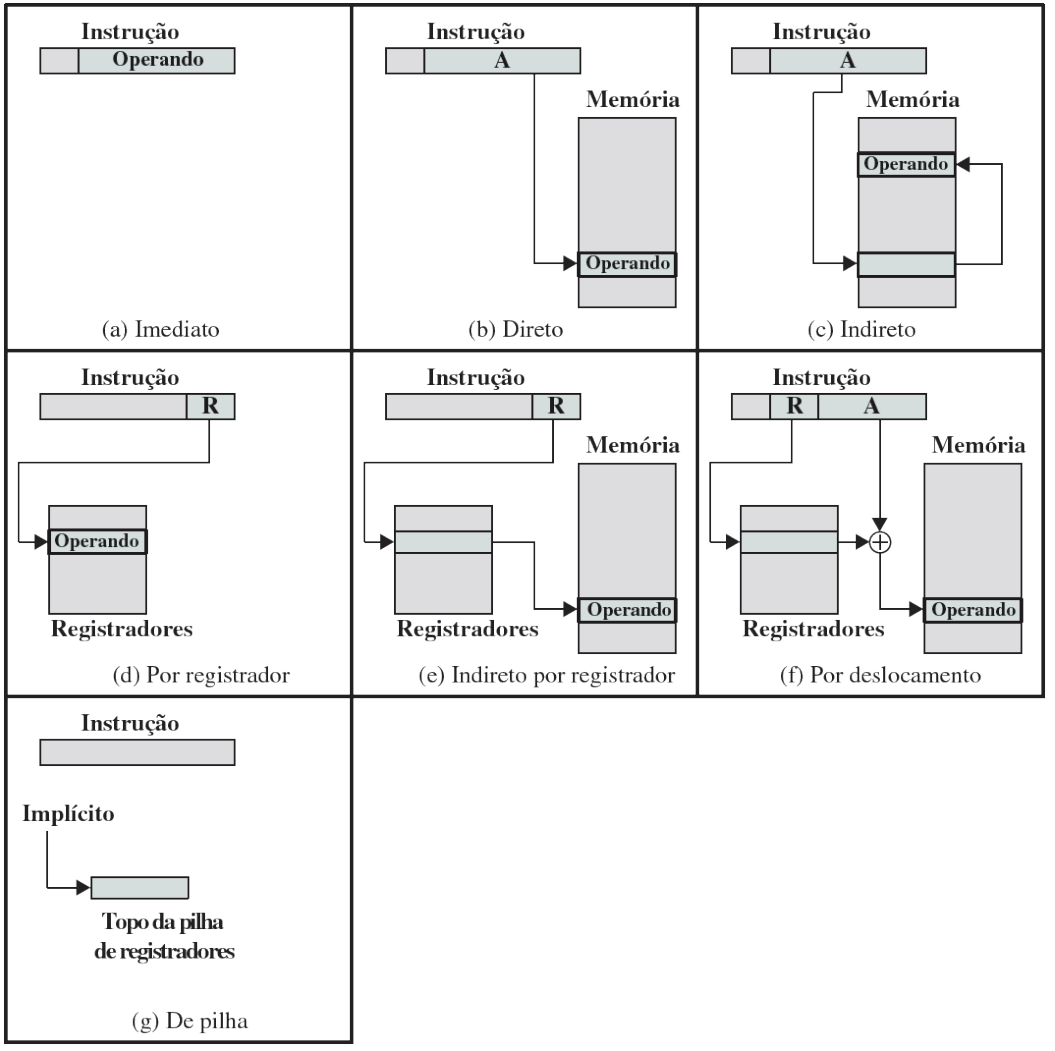
Exercício

- Modos básicos de endereçamento:

E.: Por quê?

Modo	Algoritmo	Principal vantagem	Principal desvantagem
Imediato	Operando = A	Nenhuma referência à memória	Magnitude de operando limitada
Direto	EA = A	Simples	Espaço de endereçamento limitado
Indireto	EA = (A)	Espaço de endereçamento grande	Múltiplas referências à memória
Por registrador	EA = R	Nenhuma referência à memória	Espaço de endereçamento limitado
Indireto por registrador	EA = (R)	Espaço de endereçamento grande	Referência extra de memória
Por deslocamento	EA = A + (R)	Flexibilidade	Complexidade
De pilha	EA = topo da pilha	Nenhuma referência à memória	Aplicabilidade limitada

Modos de endereçamento



Endereçamento imediato

- A forma mais simples de endereçamento é o endereçamento imediato, no qual o valor do operando está presente na instrução:

  Operando = A

- A vantagem do endereçamento imediato é que nenhuma referência de memória é necessária para obter o operando, economizando dessa forma um ciclo de memória ou de cache dentro do ciclo da instrução.
- A desvantagem é que o tamanho do número é limitado ao tamanho do campo de endereço.

Endereçamento direto

- Uma forma muito simples de endereçamento é o endereçamento direto, onde o campo de endereço contém o endereço efetivo do operando:

$$EA = A$$

- A técnica era comum nas primeiras gerações dos computadores.
- Ela requer apenas uma referência à memória e nenhum cálculo especial.
- A limitação óbvia é que ela oferece um espaço de endereços limitado.

Endereçamento indireto

- Para ter um campo de endereço se referindo ao endereço de uma palavra na memória, que, por sua vez, contém o endereço completo do operando, usamos uma técnica conhecida como endereçamento indireto:

$$EA = (A)$$

- A vantagem óbvia é que, para o tamanho N de uma palavra, um espaço de endereçamento de 2N estará disponível.
- A desvantagem é que a execução da instrução requer duas referências à memória para obter o operando: uma para obter seu endereço e outra para obter seu valor.

Endereçamento por registradores

- Endereçamento por registradores é semelhante ao endereçamento direto.
- A única diferença é que o campo de endereço se refere a um registrador em vez de um endereço da memória principal:

$$EA = R$$

- As vantagens de endereçamento por registradores são:
  i. apenas um pequeno campo de endereço é necessário,
  ii. nenhuma referência à memória que consome tempo é necessária.

Endereçamento indireto por registradores

- Assim como o endereçamento por registradores é análogo ao endereçamento direto, o endereçamento indireto por registradores é análogo ao endereçamento indireto.
- Em ambos os casos, a única diferença é se o campo de endereço referencia um local de memória ou um registrador.
- Assim, temos para endereçamento indireto de registradores:

$$EA = (R)$$

- As vantagens e as limitações são basicamente as mesmas do endereçamento indireto.

Endereçamento por deslocamento

- Uma forma muito poderosa de endereçamento combina as capacidades do endereçamento direto e do endereçamento indireto por registradores.
- Ela é conhecida como endereçamento por deslocamento:

$$EA = A + (R)$$

- Os três usos mais comuns são:
  i. Endereçamento relativo
  ii. Endereçamento por registrador base
  iii. Indexação

Endereçamento de pilha

- Uma pilha é um array linear de locais.
- A pilha é um bloco reservado de locais.
- Associado à pilha, temos um ponteiro cujo valor é o endereço do topo da pilha.
- O **modo de endereçamento de pilha** é uma forma de endereçamento implícito.
- As instruções da máquina não precisam incluir uma referência de memória, e sim operar no topo da pilha.

Modos de endereçamento do x86 e do ARM

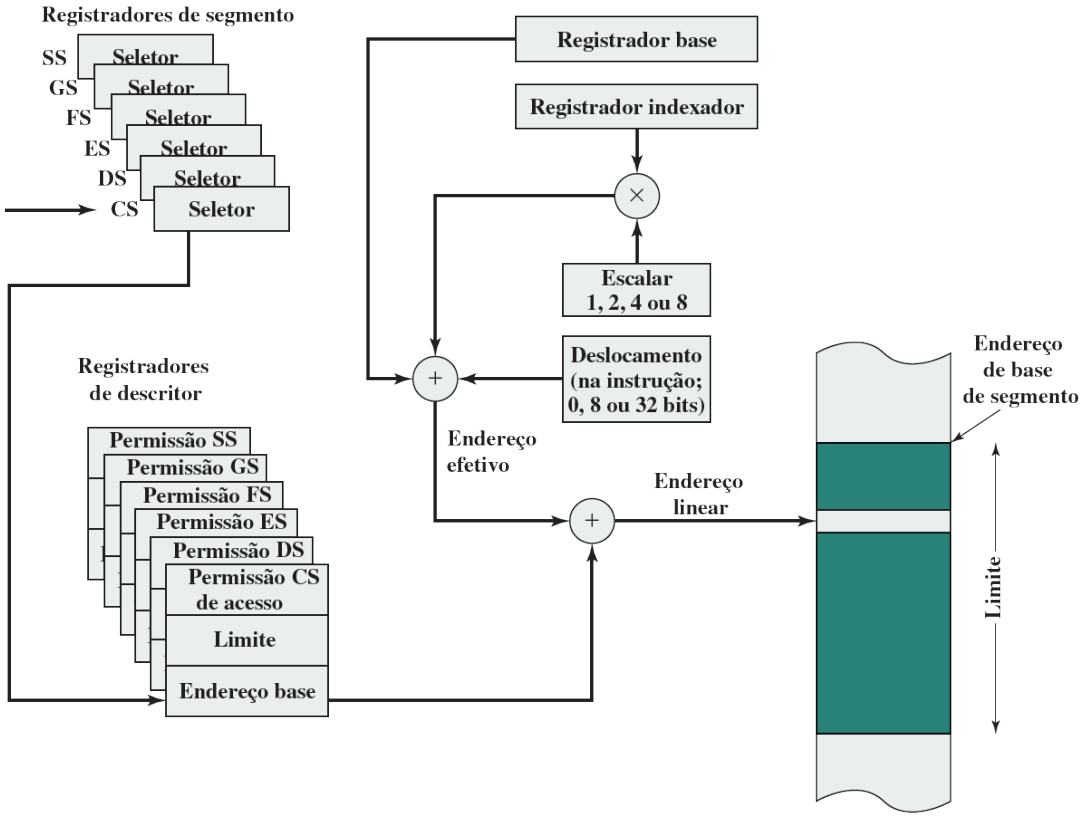
- Modos de endereçamento x86:

Modo	Algoritmo
Imediato	Operando = A
Operando em registrador	LA = R
Deslocamento	LA = (SR) + A
Base	LA = (SR) + (B)
Base com deslocamento	LA = (SR) + (B) + A
Índice escalado com deslocamento	LA = (SR) + (I) × S + A
Base com índice e deslocamento	LA = (SR) + (B) + (I) + A
Base com índice escalado e deslocamento	LA = (SR) + (I) × S + (B) + A
Relativo	LA = (PC) + A

LA = endereço linear
R = registrador
(X) = conteúdos de X
B = registrador base
SR = registrador de segmento
I = registrador indexador
PC = contador de programa
S = fator de escala
A = conteúdos de um campo de endereço dentro da instrução.

Modos de endereçamento do x86 e do ARM

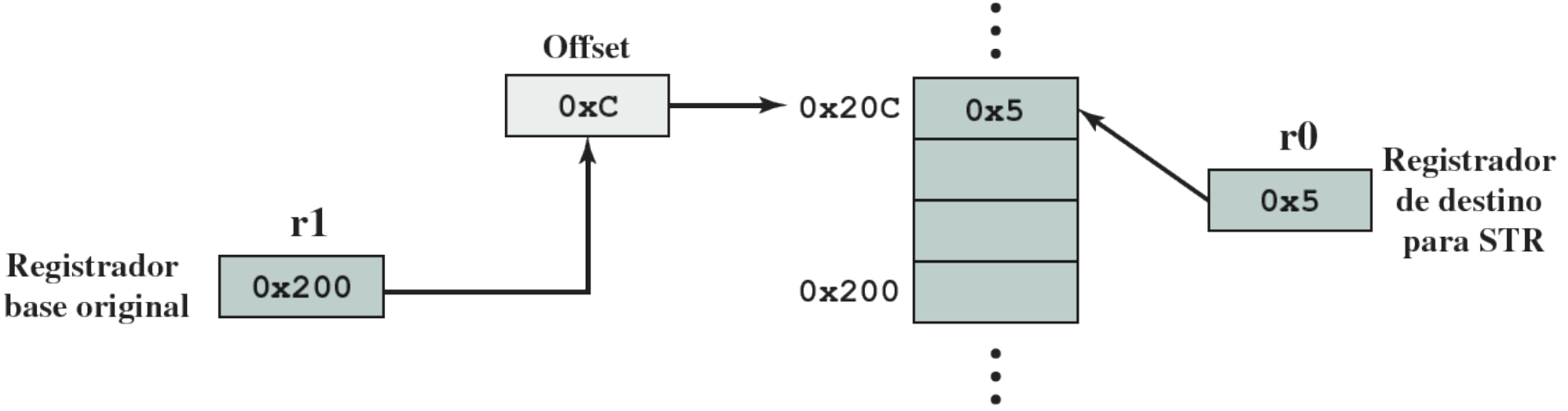
- Cálculo do modo de endereçamento do x86:



Modos de endereçamento do x86 e do ARM

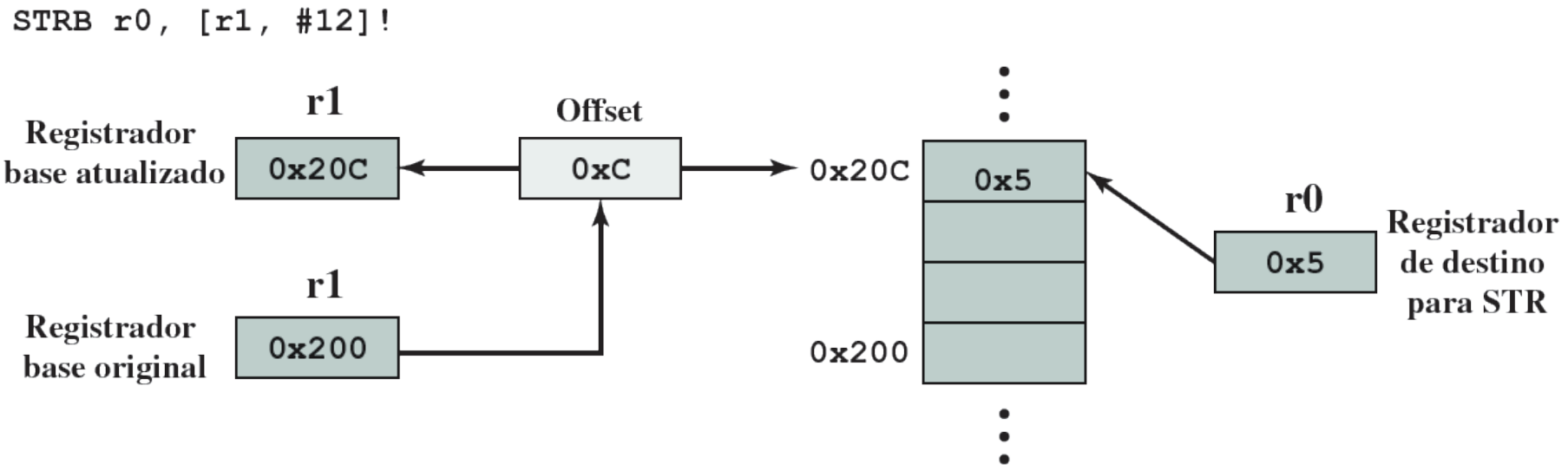
- Métodos de indexação ARM – Offset:

```
STRB r0, [r1, #12]
```



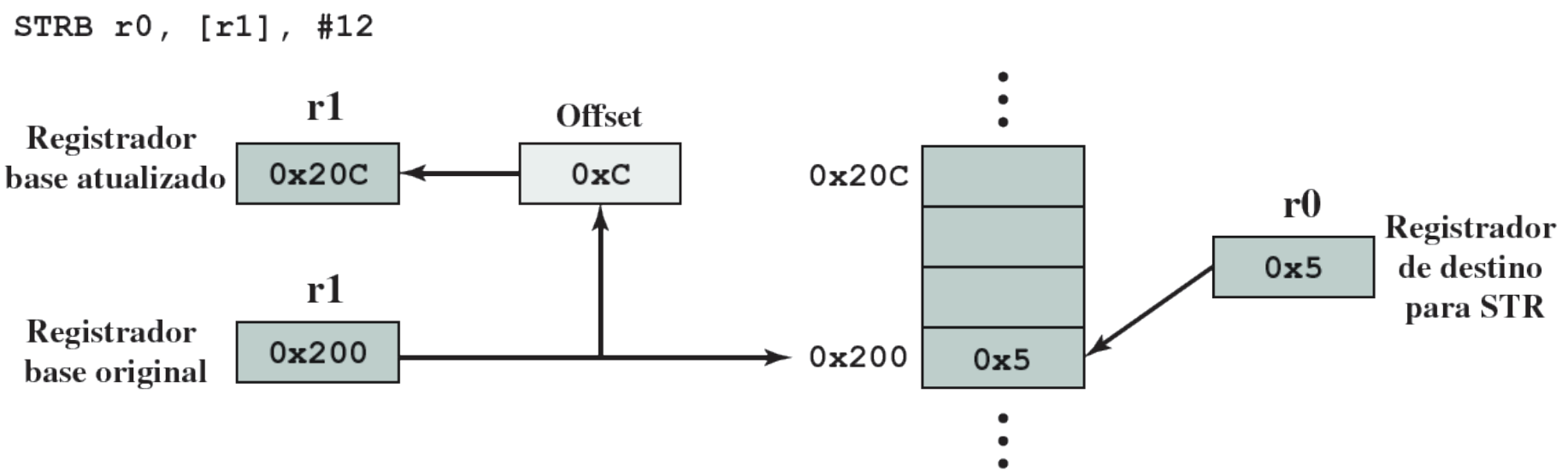
Modos de endereçamento do x86 e do ARM

- Métodos de indexação ARM – Pré-indexação:



Modos de endereçamento do x86 e do ARM

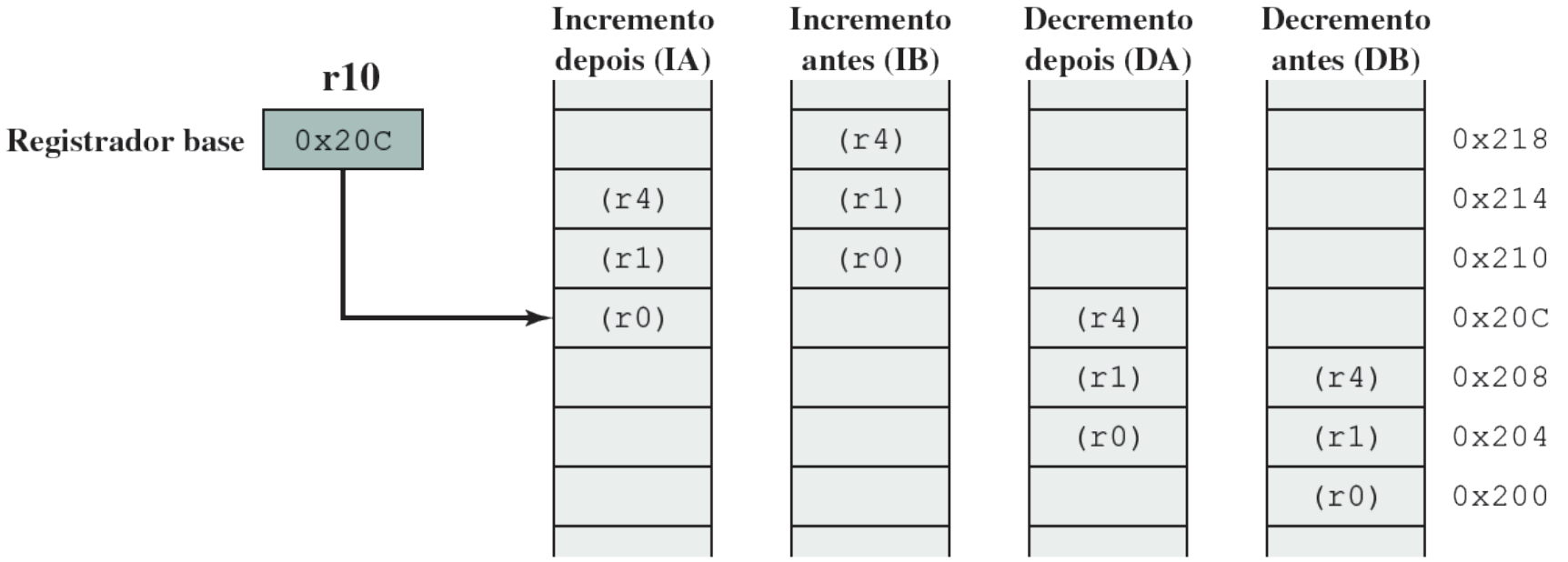
- Métodos de indexação ARM – Pós-indexação:



Modos de endereçamento do x86 e do ARM

- Endereçamento de carga/armazenamento múltiplo ARM:

```
LDMxx r10, {r0, r1, r4}
STMxx r10, {r0, r1, r4}
```



Formatos de instruções

- O formato de instrução define o **layout de bits de uma instrução**, no que diz respeito aos campos que a constituem.
- Um formato de instrução tem que incluir um **opcode** e, implícita ou explicitamente, **zero ou mais operandos**.
- O formato deve, implícita ou explicitamente, indicar o **modo de endereçamento para cada operando**.
- Na maioria dos conjuntos de instruções, mais do que um formato de instrução é usado.

Tamanho da instrução

- A decisão sobre o tamanho do formato da instrução afeta, e é afetada, pelo tamanho da memória, organização da memória, estrutura do barramento, complexidade e velocidade do processador.
- Tal decisão determina a riqueza e a flexibilidade da máquina do ponto de vista do programador de linguagem de montagem.
- A relação aqui está entre o desejo por um conjunto poderoso de instruções e a necessidade de economizar o espaço.
- Uma instrução de 64 bits ocupa o dobro de espaço de uma de 32 bits, mas provavelmente não é duas vezes mais útil.

Alocação de bits

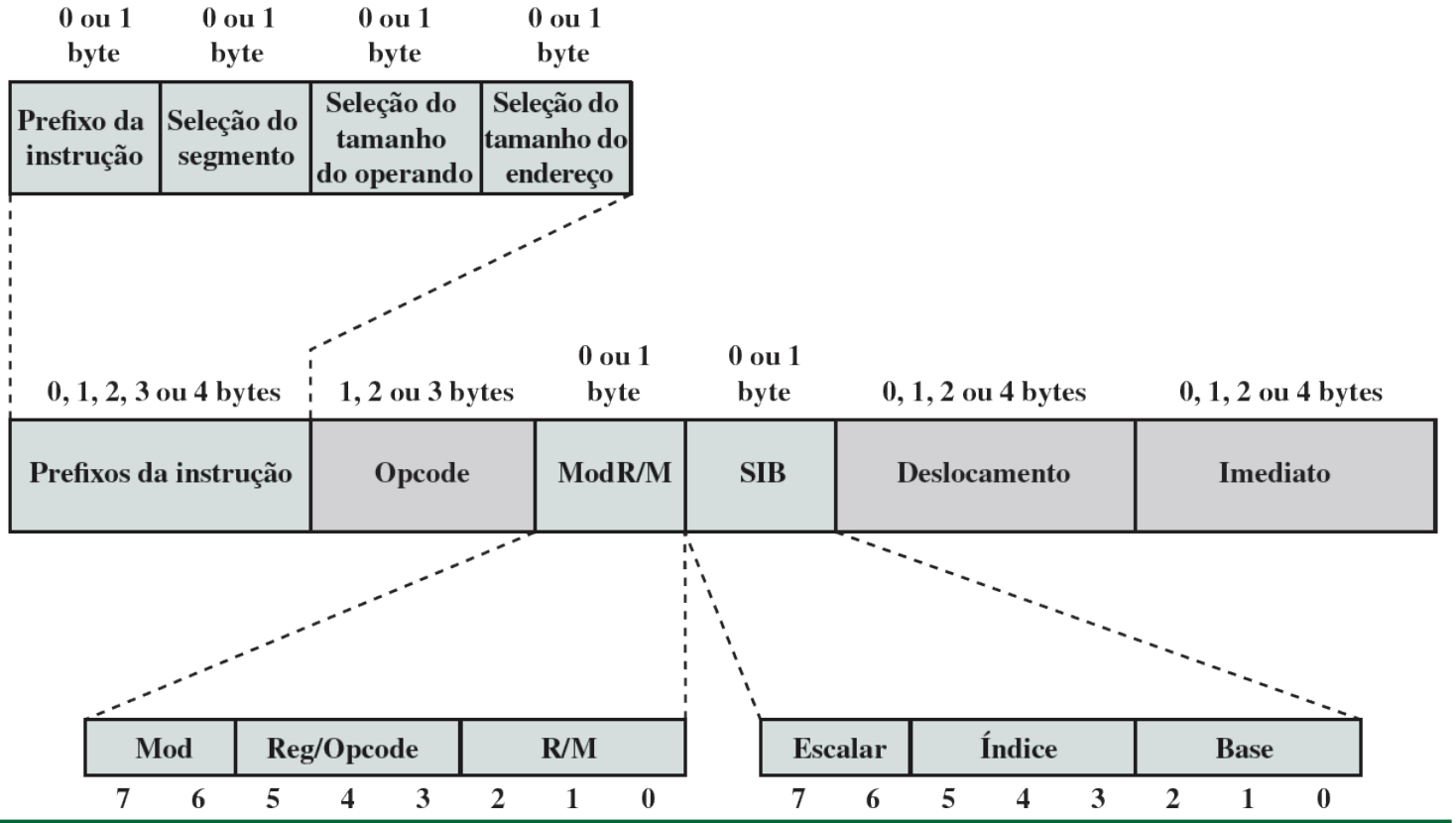
- Os seguintes fatores relacionados são importantes para determinar o uso de bits de endereçamento:
  - Número de modos de endereçamento
  - Número de operandos
  - Registrador versus memória
  - Número de conjuntos de registradores
  - Intervalo de endereços
  - Granularidade do endereço
- Dessa forma, o desenvolvedor tem de considerar e equilibrar diversos fatores.

Instruções de tamanho variável

- O endereçamento pode ser mais flexível com várias combinações de referências aos registradores e à memória e com modos de endereçamento.
- Com instruções de tamanho variável, essas variações podem ser fornecidas de uma forma eficiente e compacta.
- **O principal preço a ser pago pelas instruções de tamanho variável é o aumento na complexidade do processador.**
- O uso de instruções de tamanho variável não acaba com a necessidade de relacionar integralmente todas as extensões das instruções com o tamanho da palavra.

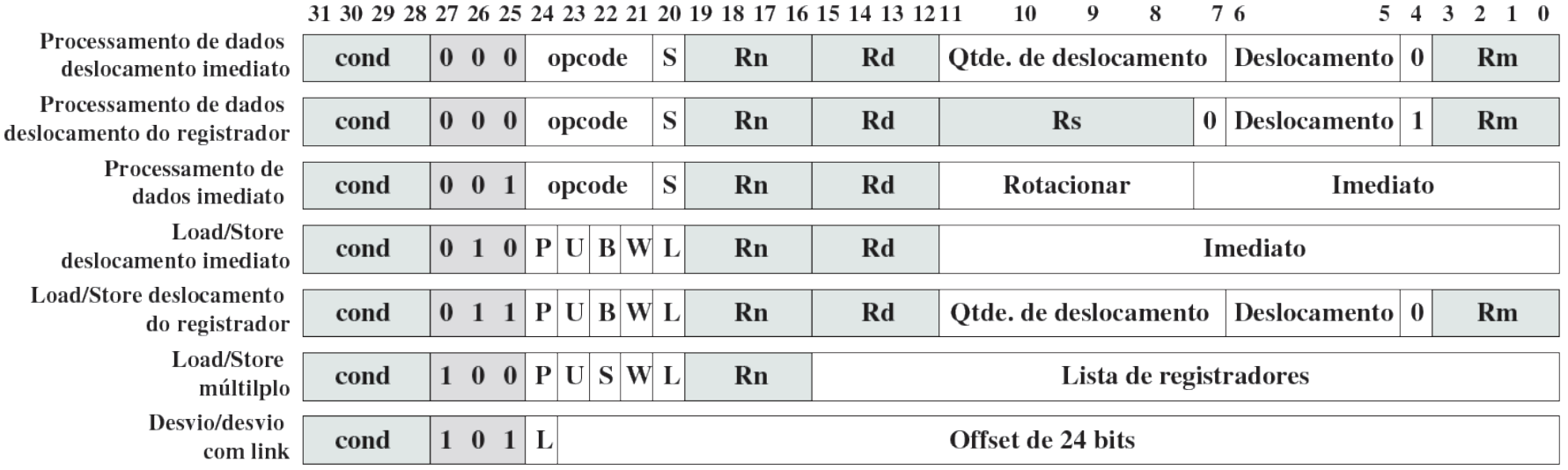
Formatos de instruções do x86 e do ARM

- Formato da instrução do x86:



Formatos de instruções do x86 e do ARM

- Formatos das instruções do ARM:



S = Para instruções de processamento de dados, significa que a instrução atualiza os códigos de condição.

S = Para instruções de múltiplo load/store, significa se a instrução em execução é restrita ao modo supervisor.

P, U, W = bits que distinguem os diferentes tipos de modos de endereçamentos.

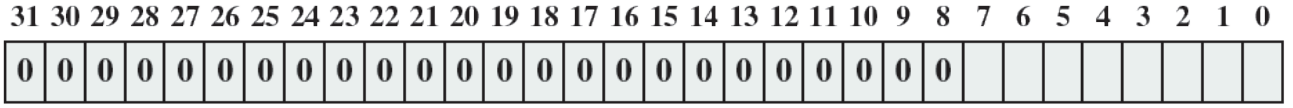
B = Distingue entre um acesso a um byte (B==1) e uma palavra (B==0) sem sinal.

L = Para instruções load/store, distingue entre um carregamento (L==1) e um armazenamento (L==0).

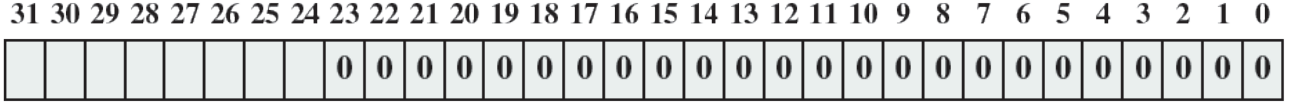
L = Para instruções de desvios, determina se um endereço de retorno está armazenado no registrador de ligação (*link register*).

Formatos de instruções do x86 e do ARM

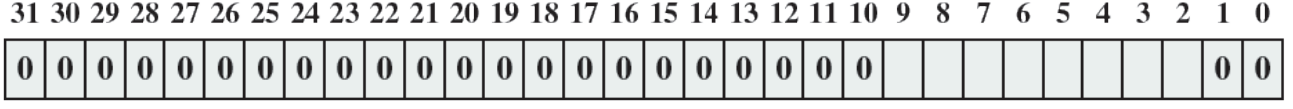
- Exemplos de uso de constantes imediatas ARM:



ror #0—range 0 through 0x000000FF—step 0x00000001



ror #8—range 0 through 0xFF000000—step 0x01000000



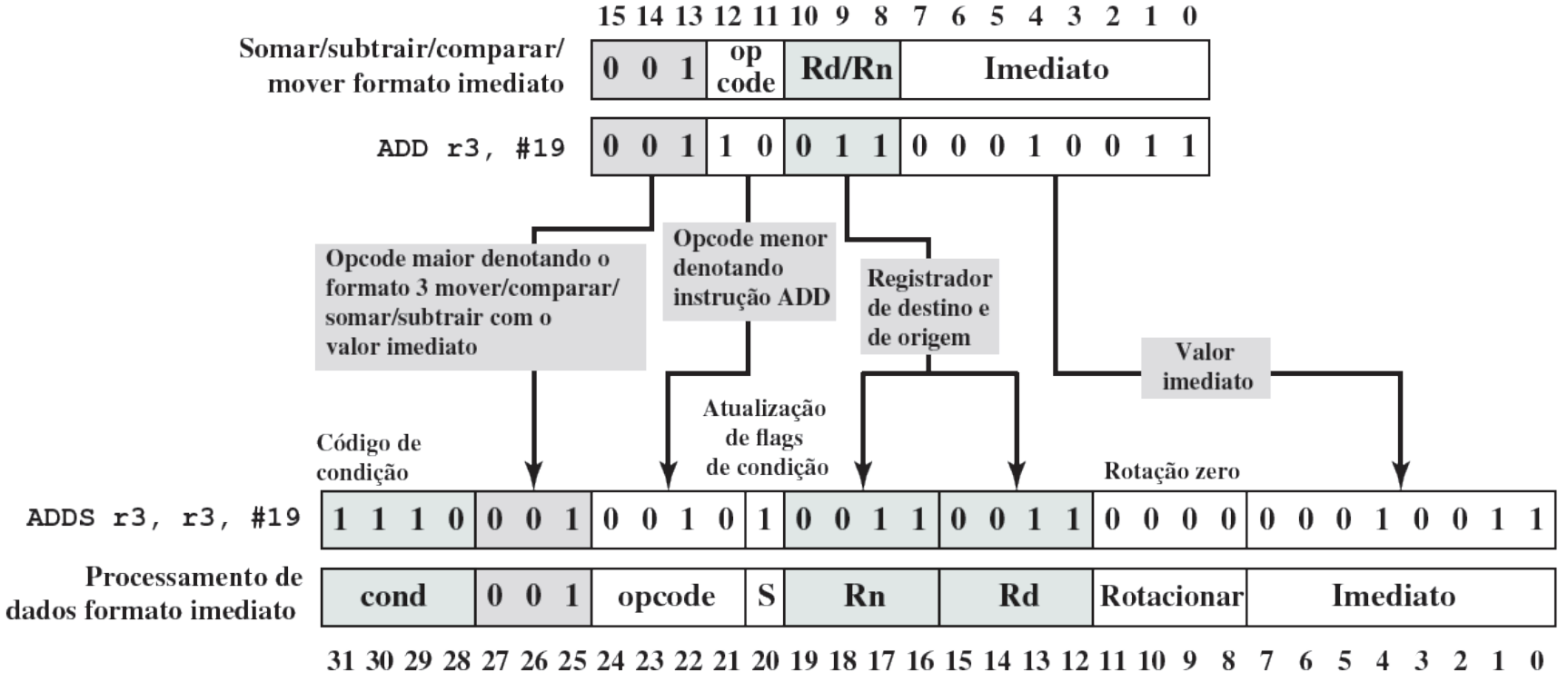
ror #30—range 0 through 0x000003FC—step 0x00000004

Formatos de instruções do x86 e do ARM

- O **conjunto de instruções Thumb** é um subconjunto recodificado do conjunto de instruções do ARM.
- As instruções Thumb não são condicionais, então o campo de código da condição não é usado.
- O Thumb possui apenas um subconjunto de operações do conjunto de instruções completo e usa apenas um campo de opcode de 2 bits mais um campo de tipo de 3 bits.
- A economia restante de 9 bits vem da redução na especificação do operando.

Formatos de instruções do x86 e do ARM

- Expansão de uma instrução Thumb ADD em seu ARM equivalente:

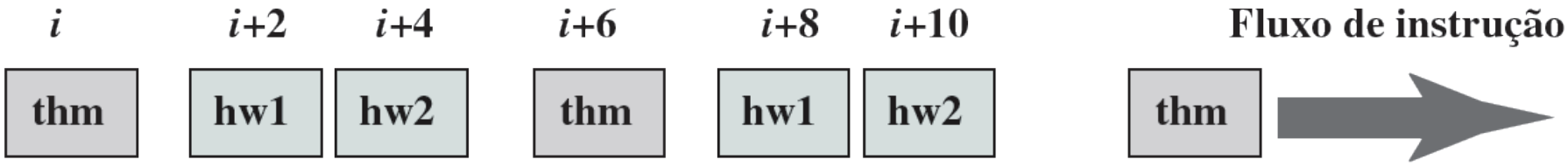


Formatos de instruções do x86 e do ARM

- O Thumb-2 é a principal melhoria na arquitetura do conjunto de instruções (ISA) do Thumb.
- Ele introduz instruções de 32 bits que podem ser intermixadas livremente com as instruções de 16 bits antigas.
- Essas novas instruções de 32 bits cobrem quase toda a funcionalidade do conjunto de instruções ARM.
- A diferença mais importante entre Thumb ISA e ARM ISA é que a maioria das instruções Thumb de 32 bits é incondicional, ao passo que quase todas as instruções ARM podem ser condicionais.

Formatos de instruções do x86 e do ARM

- Codificação Thumb-2:



Meia-palavra1 [15:13]	Meia-palavra1 [12:11]	Tamanho	Funcionalidade
Não 111	xx	16 bits (1 meia-palavra)	Instrução Thumb de 16-bit
111	00	16 bits (1 meia-palavra)	Instrução de desvio incondicional
111	Não 00	32 bits (2 meias-palavras)	Instruções Thumb-2 de 32 bits

Linguagem de montagem

- Um processador pode entender e executar **instruções de máquina**.
- Essas instruções são simples **números binários** armazenados no computador.
- Se o programador quisesse programar diretamente na linguagem de máquina, então seria necessário entrar com o programa como dados binários.
- Considere a simples instrução BASIC:

$$N = I + J + K$$

Linguagem de montagem

- Um ótimo sistema, normalmente utilizado, é usar endereços simbólicos.
- Cada linha ainda consiste em três campos.
- O primeiro campo ainda é para endereço, porém um símbolo é usado no lugar de um endereço numérico absoluto.
- Algumas linhas não possuem endereço, o que implica que o endereço dessa linha é um a mais do que o endereço da linha anterior.
- Para instruções que referenciam memória, o terceiro campo também contém um endereço simbólico.

Linguagem de montagem

- Programa em linguagem de montagem (assembly):

Rótulo	Operação	Operando
FORMUL	LDA	I
	ADD	J
	ADD	K
	STA	N
I	DATA	2
J	DATA	3
K	DATA	4
N	DATA	0