

[< VOLTAR](#)

Tipos de operações - linguagem de montagem (assembly) - modos de endereçamento

Abordaremos nesta aula como são efetuadas as operações e a montagem em linguagem de máquina.

NESTE TÓPICO



Marcar
tópico



O conjunto de instruções é um dos pontos centrais na arquitetura de um processador.

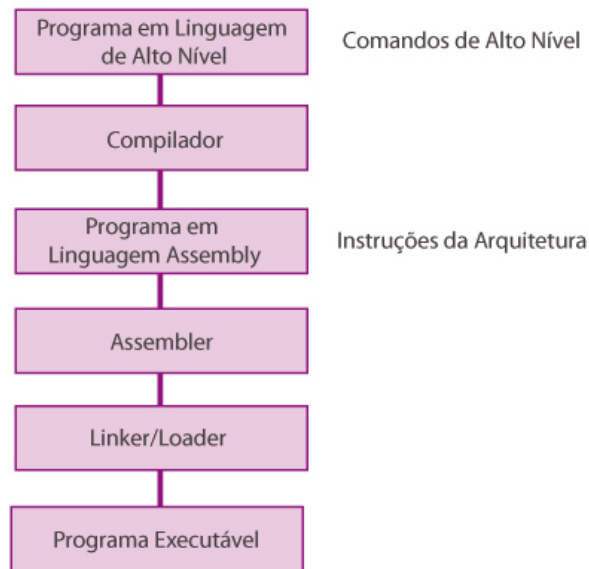
Vários aspectos na definição e implementação da arquitetura são influenciados pelas características do conjunto de instruções. Por exemplo, as operações realizadas pela unidade lógica e aritmética, o número e função dos registradores e a estrutura de interconexão dos componentes da seção de processamento. Além disso, as operações básicas que acontecem dentro da seção de processamento dependem das instruções que devem ser executadas.

O conjunto de instrução afeta o projeto da seção de controle. A sua estrutura e a sua complexidade são determinadas diretamente pelas características do conjunto de instruções. Esta aula discute os principais aspectos de um conjunto de instruções, como tipos de operações, operandos e modos de endereçamento.

1. Conjunto de instruções no contexto de software

A Figura 1 situa o conjunto de instruções do processador dentro dos diversos níveis de software existentes em um sistema de computação.





Em geral, os programas são desenvolvidos em uma linguagem de alto nível como FORTRAN, Pascal ou C. O compilador traduz o programa de alto nível em uma sequência de **instruções de processador**. O resultado dessa tradução é o programa em **linguagem de montagem** ou **linguagem de máquina** (*assembly language*). A linguagem de montagem é uma forma de representar textualmente as instruções oferecidas pela arquitetura. Cada arquitetura possui uma linguagem de montagem particular. No programa em linguagem de montagem, as instruções são representadas por meio de mnemônicos, que associam o nome da instrução à sua função, por exemplo, ADD ou SUB, isto é, soma e subtração, respectivamente.

O programa em linguagem de montagem é convertido para um programa em **código objeto** pelo **montador** (*assembler*). O montador traduz diretamente uma instrução da forma textual para a forma de código binário. É sob a forma binária que a instrução é carregada na memória e interpretada pelo processador.

Programas complexos são normalmente estruturados em módulos. Cada módulo é compilado separadamente e submetido ao montador, gerando diversos módulos em código objeto. Esses módulos são reunidos pelo **ligador** (*linker*), resultando finalmente no programa executável que é carregado na memória.

O conjunto de instruções de uma arquitetura se distingue por meio de diversas características. As principais características de um conjunto de instruções são: tipos de instruções e operandos, número e localização dos operandos em instruções aritméticas e lógicas, modos de endereçamento para acesso aos dados na memória e o formato dos códigos de instrução. Esses aspectos são analisados a seguir.

2. Tipos de instruções e de operandos

As instruções oferecidas por uma arquitetura podem ser classificadas em categorias, de acordo com o tipo de operação que realizam. Em geral, uma arquitetura fornece pelo menos três categorias de instruções básicas:

- **instruções aritméticas e lógicas**: são as instruções que realizam operações aritméticas sobre números inteiros (adição, subtração) e operações lógicas



bit a bit (*AND*, *OR*).

- **instruções de movimentação de dados:** instruções que transferem dados entre os registradores ou entre os registradores e a memória principal.
- **instruções de transferência de controle:** instruções de desvio e de chamada de rotina, que transferem a execução para uma determinada instrução dentro do código do programa.

Várias arquiteturas oferecem outras categorias de instruções, voltadas para operações especializadas. Dentre elas, podemos citar:



- **instruções de ponto flutuante:** instruções que realizam operações aritméticas sobre números com ponto flutuante.
- **instruções decimais:** instruções que realizam operações aritméticas sobre números decimais codificados em binário (BCD – *binary coded decimal*).
- **instruções de manipulação de bits:** instruções para testar ou atribuir o valor de um bit.
- **instruções de manipulação de *strings*:** instruções que realizam operações sobre cadeias de caracteres (*strings*), como movimentação e comparação, ou ainda procura de um caractere dentro de um *string*.

Existem muitas diferenças entre as arquiteturas quanto às categorias de instruções oferecidas. Arquiteturas de uso geral oferecem a maioria das categorias relacionadas anteriormente. Arquiteturas destinadas para uma aplicação específica podem oferecer outros tipos de instruções, especializadas para aquela aplicação. Um exemplo seria uma arquitetura voltada para processamento gráfico, que ofereceria instruções para realizar operações sobre *pixels*.

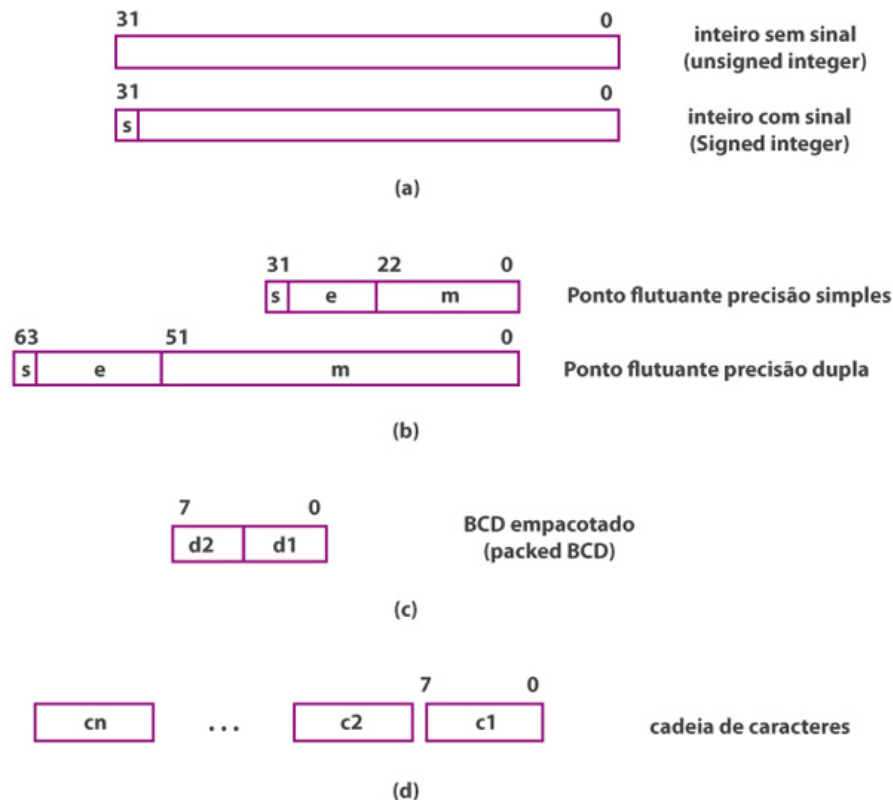
Os tipos de operandos que podem ser diretamente manipulados por uma arquitetura dependem, é claro, dos tipos de instruções oferecidas. A Figura 2 mostra como os principais tipos de dados são normalmente representados em uma arquitetura de uso geral.

A Figura 2(a) mostra a representação de inteiros, neste exemplo particular, inteiros com 32 bits. Números inteiros podem ser representados com ou sem sinal. Em um número inteiro com sinal, o bit mais significativo é reservado para indicar o estado do sinal (positivo ou negativo). Números inteiros sem sinal assumem apenas valores positivos. Algumas arquiteturas oferecem instruções específicas para aritmética com ou sem sinal. Essas instruções diferem no modo como são alterados os bits do registrador de estado associado a ALU. Algumas linguagens de programação tornam visível para o programador essa distinção entre inteiros com ou sem sinal. Na linguagem C, por exemplo, uma variável declarada do tipo *int* é representada por um inteiro com sinal. Ao contrário, variáveis do tipo *unsigned int* são representadas por inteiros sem sinal, sendo normalmente usadas para indexar elementos de vetores.

A Figura 2(b) mostra a representação de números com ponto flutuante, com precisão simples e dupla. A diferença entre precisões está no número de bits usados para representar a mantissa e o expoente. Atualmente, a maioria



das arquiteturas que operam números com ponto flutuante obedece a um padrão, denominado IEEE 754, que define a representação e um conjunto de operações aritméticas e lógicas para números com ponto flutuante.



A Figura 2(c) mostra a representação de números BCD empacotados (*Packed Binary Coded Decimal*). Nessa representação, dois dígitos decimais codificados em binário são representados dentro de um byte, cada dígito sendo codificado em quatro bits do byte. Finalmente, a Figura 2(d) mostra a representação de cadeias de caracteres, em que cada byte dentro de uma sequência de bytes codifica um caractere segundo certo padrão (por exemplo, o padrão ASCII).

3. Número e localização dos operandos

Outra característica de um conjunto de instruções é o número de operandos explicitamente indicados em uma instrução aritmética ou lógica. Em algumas arquiteturas, essas instruções referenciam explicitamente três operandos, dois **operandos-fonte** e um **operando-destino**, como em:

ADD R1, R2, R3

Em que R1 e R2 são os operandos-fonte e R3 é o operando-destino. Em outras arquiteturas, instruções aritméticas/lógicas especificam apenas dois operandos. Nesse caso, um dos operandos-fonte é também o operando-destino. Por exemplo, na instrução:

ADD R1, R2

R2 contém um dos operandos-fonte e também é usado como operando-destino.

Quanto à localização dos operandos especificados por uma instrução



aritmética/lógica, podemos encontrar arquiteturas em que podem ser realizados acessos aos operandos diretamente a partir da memória principal. Por exemplo, nessas arquiteturas podemos ter instruções como:

ADD M1, R1, R2

ADD M1, M2, R1

ADD M1, M2, M3

em que M1, M2 e M3 são endereços de locações de memória. Em outro extremo, existem arquiteturas em que todos os operandos encontram-se apenas em registradores. As instruções aritméticas/lógicas são todas do tipo:

ADD R1, R2, R3

ADD R1, R2

A partir do número de operandos explicitamente referenciados e da localização desses operandos, podemos classificar as arquiteturas nos seguintes tipos:

- **arquitetura memória-memória:** as instruções aritméticas/lógicas usam três operandos e todos os operandos podem estar na memória.
- **arquitetura registrador-memória:** as instruções aritméticas/lógicas usam dois operandos, e apenas um deles pode residir na memória.
- **arquitetura registrador-registrador:** as instruções aritméticas/lógicas usam três operandos, todos em registradores.

Nesse caso, apenas duas instruções acessam diretamente a memória: LOAD e STORE. A instrução LOAD carrega em um registrador um dado armazenado na memória, e a instrução STORE armazena na memória o conteúdo de um registrador.

Arquiteturas memória-memória e registrador-memória apresentam como vantagem um menor número de instruções no código do programa, já que não é necessário carregar previamente em registradores os operandos-fonte de uma instrução aritmética/lógica, como acontece em uma arquitetura registrador-registrador. Por outro lado, a existência de instruções aritméticas/lógicas mais poderosas torna mais complexa a implementação da arquitetura. As arquiteturas Intel 80x86 e Motorola MC680x0 são do tipo registrador-memória. Dentre as arquiteturas memória-memória, podemos citar o DEC VAX 11.

4. Modos de endereçamento

Os operandos de uma instrução podem encontrar-se em registradores, na memória principal ou ainda embutidos na própria instrução. O **modo de endereçamento** refere-se à maneira como uma instrução especifica a localização dos seus operandos. Existem três modos de endereçamento básicos:



- **modo registrador:** a instrução indica o número de um registrador de dados em que se encontra um operando (fonte ou destino).
- **modo imediato:** a instrução referencia um operando que se encontra dentro do próprio código da instrução.
- **modo implícito:** a localização do operando não está explicitamente indicada na instrução.



Por exemplo, nas chamadas **arquiteturas acumulador**, um dos operandos-fonte e o operando-destino nas instruções aritméticas/lógicas encontra-se sempre em um registrador especial, o **acumulador**. Assim, não é necessário que esse registrador seja explicitamente referenciado pela instrução.

A Figura 3 mostra exemplos de instruções que usam os modos de endereçamento implícito, registrador e imediato.

| Modo | Exemplo | Significado |
|-------------|------------|-------------------------|
| Implícito | ADD R1 | $Ac \leftarrow Ac + R1$ |
| Registrador | ADD R1, R2 | $R2 \leftarrow R1 + R2$ |
| Imediato | ADD R1, #4 | $R1 \leftarrow R1 + 4$ |

Os modos de endereçamento citados referenciam apenas operandos que se encontram em registradores ou na instrução. Existem ainda os modos de endereçamento usados para referenciar dados armazenados na memória principal. Entre as diferentes arquiteturas, existe uma enorme variedade de modos de endereçamento referentes à memória principal, e que formam, na realidade, uma classe de modos de endereçamento à parte.

Um modo de endereçamento referente à memória indica como deve ser obtido o endereço da locação de memória em que se encontra o dado que será acessado. Esse endereço é chamado **endereço efetivo**. Apesar da variedade mencionada, é possível identificar alguns modos de endereçamento referentes à memória que são oferecidos pela maioria das arquiteturas. Esses modos de endereçamento mais comuns estão relacionados na Figura 4.

| Modo | Exemplo | Significado | Uso |
|-----------------|------------------|--------------------------------|----------------------------------|
| Direto | ADD (100), R1 | $R1 \leftarrow M[100] + R1$ | acesso a variáveis estáticas |
| Indireto | ADD (R1), R2 | $R2 \leftarrow M[R1] + R2$ | acesso via ponteiros |
| Relativo à base | ADD 100 (R1), R2 | $R2 \leftarrow M[100+R1] + R2$ | acesso a elementos em estruturas |
| Indexado | ADD (R1+ R2), R3 | $R3 \leftarrow M[R1+R2] + R3$ | acesso a elementos em um vetor |

No **modo direto**, o endereço efetivo é um valor imediato contido no código da instrução. Por exemplo, na instrução ADD (100), R1, um dos operandos encontra-se na locação de memória com endereço 100. O modo de



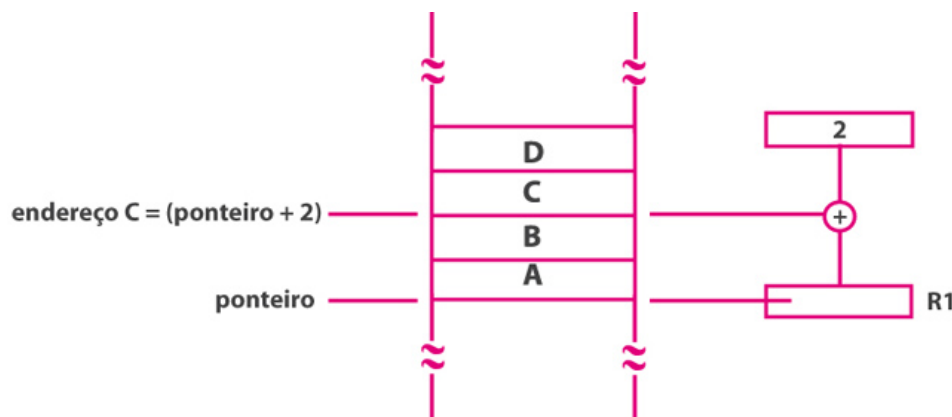
endereçamento direto é usado principalmente no acesso às **variáveis estáticas** de um programa, cujo endereço em memória pode ser determinado durante a compilação do programa.

No **modo indireto**, o endereço efetivo encontra-se em um registrador. Por exemplo, na instrução ADD (R1), R2, um dos operandos encontra-se na locação de memória cujo endereço está no registrador R1. Ou seja, o operando na memória é indicado indiretamente, por meio de um registrador que contém o endereço efetivo. Esse modo de endereçamento é usado no acesso a **variáveis dinâmicas**, cujo endereço na memória é conhecido apenas durante a execução do programa.



O acesso a uma variável dinâmica é realizado por meio de um ponteiro, que nada mais é do que o endereço da variável. Para realizar o acesso à variável dinâmica, o ponteiro é carregado em um registrador, e a instrução que acessa a variável usa esse registrador com o modo de endereçamento indireto.

No **modo relativo à base**, o endereço efetivo é a soma do conteúdo de um registrador, chamado **endereço-base**, com um valor imediato contido na instrução, chamado **deslocamento**. Por exemplo, na instrução ADD 100(R1), R2, R1 contém o endereço-base e 100 é o deslocamento. O endereço efetivo do operando em memória é a soma do conteúdo de R1 com o valor 100. O modo relativo à base é usado no acesso aos componentes de variáveis dinâmicas estruturadas (por exemplo, *record* em Pascal ou *struct* em C). A Figura 4 mostra como é calculado o endereço efetivo no modo de endereçamento relativo à base.

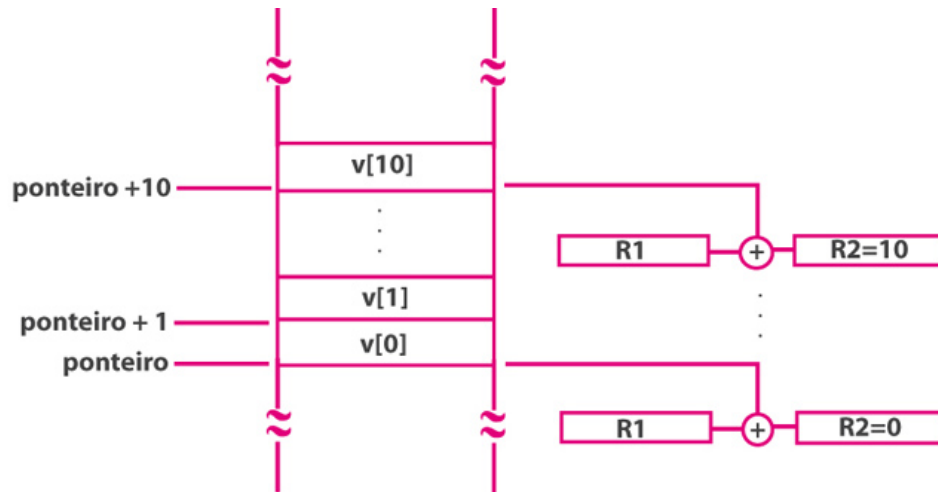


A figura mostra a localização na memória de uma estrutura com quatro campos, A, B, C e D. O endereço inicial da estrutura é indicado por um ponteiro, que se torna conhecido apenas durante a execução do programa.

No entanto, a posição de cada campo em relação ao início da estrutura é fixo, sendo conhecido durante a compilação. O endereço de um campo é obtido somando-se a posição do campo (o deslocamento) ao ponteiro que indica o início da estrutura (o endereço-base). Por exemplo, na Figura 5, para somar um valor ao campo C, o compilador pode usar a instrução ADD 2(R1), R2, precedida de uma instrução para carregar em R1 o endereço-base da estrutura.



No **modo indexado**, o endereço efetivo é dado pela soma de um **índice** com um endereço-base, ambos armazenados em registradores. Por exemplo, na instrução ADD (R1 + R2), R1 contém o endereço-base, e R2 o índice. O modo indexado é normalmente usado no acesso aos elementos de um vetor. A Figura 5 mostra como é calculado o endereço efetivo no modo de endereçamento indexado.



A Figura 6 representa a localização na memória de um vetor V. Um ponteiro indica o endereço-base do vetor, em que se encontra o primeiro elemento. O endereço de cada elemento é obtido somando o índice do elemento ao endereço-base. Para realizar o acesso sequencialmente aos elementos do vetor, o índice é inicialmente carregado no registrador com o valor 0. O índice é então incrementado dentro de um *loop* após o acesso a cada elemento. Por exemplo, para somar um valor em registrador aos elementos do vetor, o compilador pode usar as seguintes instruções em um *loop*:

ADD R1, (R2 + R3) ADD 1, R3

em que R1 contém o valor a ser somado, R2 contém o ponteiro para o vetor e R3 é o registrador com o índice, com valor inicial 0.

Referências

MACHADO, Francis B.; MAIA, Luiz P. *Arquitetura de sistemas operacionais*. 4. ed. Rio de Janeiro: LTC, 2007.

STALLINGS, Willian. *Arquitetura e organização de computadores*. 5. ed. Prentice Hall. São Paulo, 2006.

TANENBAUM. Andrew S. *Organização estruturada de computadores*. 5. ed. Rio de Janeiro: LTC, 2007.

WEBER, Raul Fernando. *Arquitetura de computadores pessoais*. 2. ed. Porto Alegre: Sagra Luzzatto, 2003.

_____. *Fundamentos de arquitetura de computadores*. 3. ed. Porto Alegre: Sagra Luzzatto, 2004.





Avalie este tópico





<

ANTERIOR

Arquitetura do conjunto de instruções (ISA); características de instruções de máquina; tipos de operandos

Índice

Biblioteca
(https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/) Portal Uninove (http://www.uninove.br) Mapa do Site

Ajuda?
(https://idCurso: ,



® Todos os direitos reservados

