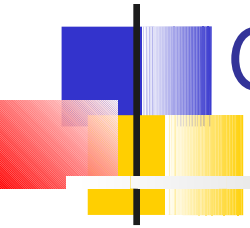


Organização e Arquitetura de Computadores

Material de apoio

CPU: INSTRUÇÕES

Tópico 32

Grupo de Professores OAC

2010

Esclarecimentos

- Esse material é de apoio para as aulas da disciplina e não substitui a leitura da bibliografia básica.
- Os professores da disciplina irão focar alguns dos tópicos da bibliografia assim como poderão adicionar alguns detalhes não presentes na bibliografia, com base em suas experiências profissionais.
- O conteúdo de slides com o título “Comentário” seguido de um texto, se refere a comentários adicionais ao slide cujo texto indica e tem por objetivo incluir alguma informação adicional aos conteúdo do slide correspondente.
- Bibliografia básica:
  - PATTERSON, A.D.E.; HENNESSY, L.J.. Organização e projetos de computadores: a interface hardware/software. São Paulo: Campus, 2005.;
  - MONTEIRO, Mário A.. Introdução à organização de computadores. 5.ed. Rio de Janeiro: LTC, 2007.
  - STALLINGS, William. Arquitetura e organização de computadores : projeto para o desempenho. São Paulo: Pearson Education, 2005.

Instruções de Máquina

A operação de uma UCP é determinada pelas instruções que ela executa, conhecidas como **instruções de máquina**. A coleção das diferentes instruções que a UCP é capaz de executar é conhecida como o **conjunto de instruções** do processador, o qual pode variar de um fabricante para outro e mesmo de um modelo para outro de processador.

Cada instrução deve conter toda a informação necessária para permitir sua execução pela UCP. Essa informação necessária compreende:

- Código de operação: especifica a operação a ser realizada
- Operando fonte*: operandos que constituem dados de entrada para a operação
- Operando de destino*: resultado que pode ser produzido pela operação
- Endereço da próxima instrução: local onde deve ser buscada a instrução seguinte (em alguns casos), após o término da corrente

* Os operandos fonte e de destino podem estar localizados na memória principal, em registradores da UCP, ou em dispositivos de E/S.

Instruções de Máquina

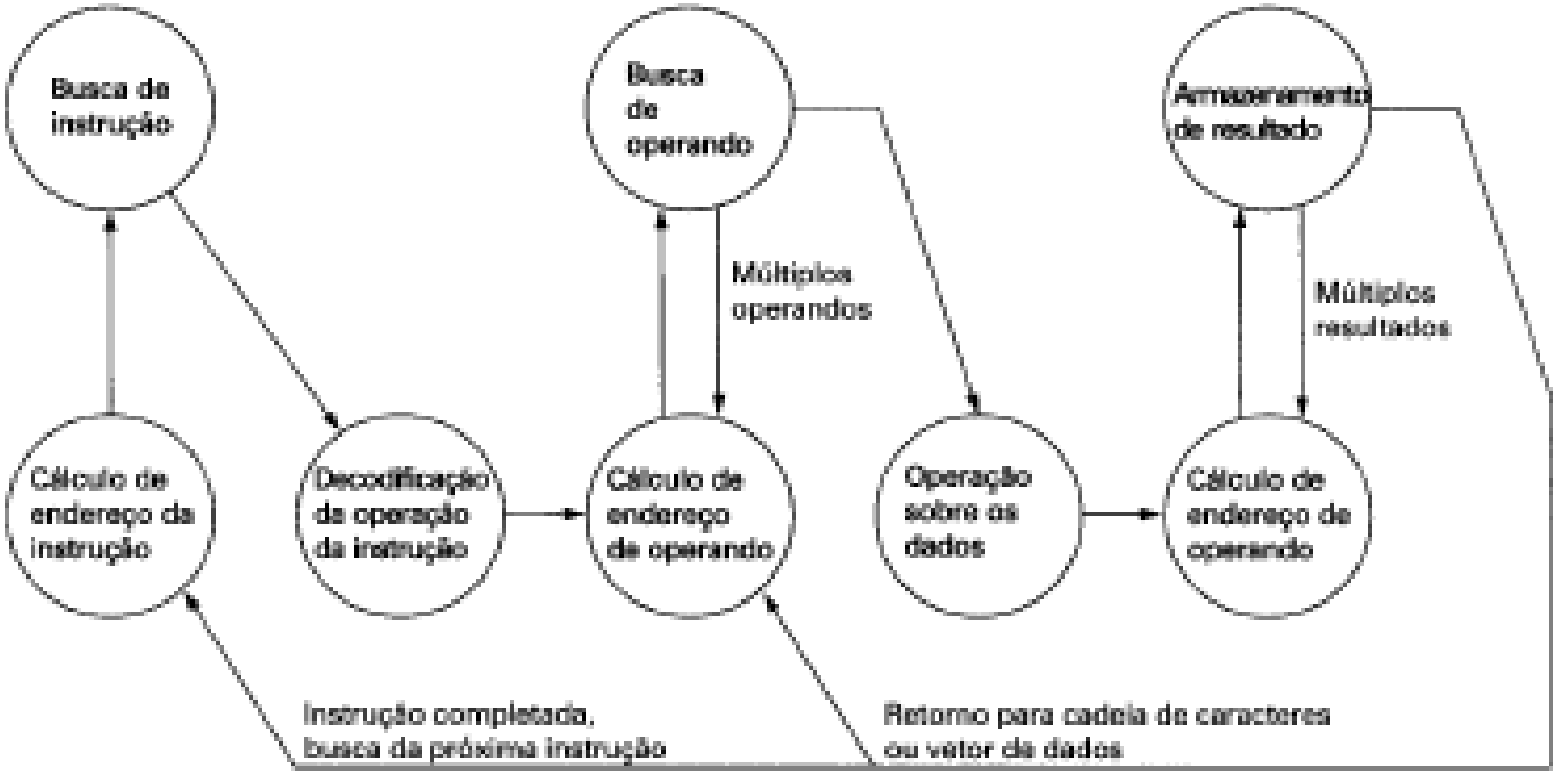


Diagrama de estados do ciclo de instruções

Instruções de Máquina

O projeto de um processador é condicionado ao conjunto de instruções de máquina as quais se deseja que ele execute. Uma das mais fundamentais análises e decisões do projeto envolve o tamanho e a complexidade do conjunto de instruções.

Num processador, quanto menor e mais simples for o conjunto de instruções, mais rápido será o seu ciclo de tempo.

Atualmente há duas tecnologias de projeto de processadores empregadas pelo mercado:

- Sistema com conjuntos de instruções complexo ***(Complex Instruction Set Computer - CISC);***
- Sistema com conjunto de instruções reduzido ***(Reduced Instruction Set Computer - RISC).***

Formato das Instruções

Cada instrução consiste num grupo de bits que pode ser dividido em duas partes:

- a primeira parte indica o que é a instrução e como será executada, sendo constituída de um só campo;
- a segunda parte refere-se ao(s) dado(s) que será(ão) manipulado(s) na operação, podendo ser constituída por mais de um campo.

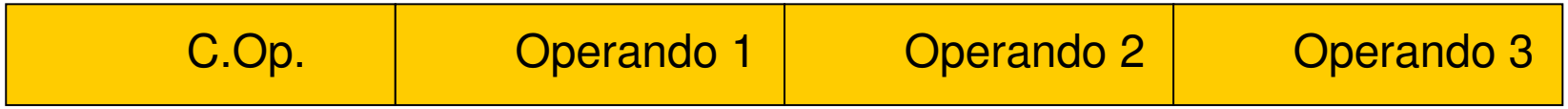
Assim, cada uma das instruções é composta pelos seguintes campos:

- um campo (subgrupo de bits) denominado código de operação – **C.Op.** – cujo valor binário é a identificação (código) da operação a ser realizada, o qual servirá de dado de entrada para o decodificados da área de controle;

- o grupo restante de bits, denominado campo do operando – **Op.** - ou operando, cujo valor binário indica a localização do dado (ou dados) que será(ão) manipulado(s) durante a execução da operação.

Exemplos

Instrução para adição de dois valores (operandos 1 e 2), indicando o endereço (operando 3) a ser armazenado o resultado: OP3 ← OP1 + OP2



Instrução para adição de dois valores (operandos 1 e 2), armazenando-se o resultado no local do operando 1: OP1 ← OP1 + OP2

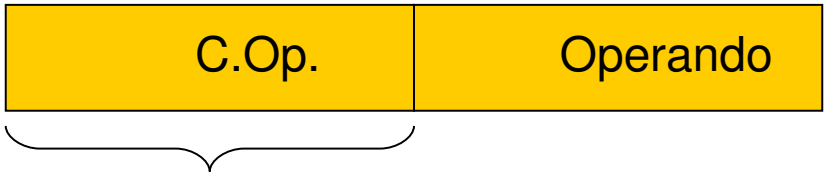
Utilizando-se o acumulador para armazenar inicialmente um dos operandos e depois armazenar o resultado da soma: ACC ← ACC + OP

Tamanho das Instruções

A definição dos códigos de operação do conjunto de instruções de um processador pode ser feita por duas maneiras:

- instruções com C.Op. de tamanho fixo;
- instruções com C.Op. de tamanho variável.

No primeiro caso, todas as instruções têm um C.Op. com a mesma quantidade de bits. A implementação das instruções e sua manipulação durante a execução de um programa são facilitadas, ao passo que o tamanho do C.Op., e da própria instrução tende a aumentar, influenciando no aumento do tamanho ocupado pelo programa na MP.



C.Op. Fixo

8 bits

O número de bits do C.Op. Também indica o número máximo de instruções passíveis de implementação (no exemplo, 256 instruções no máximo).

Tamanho das Instruções

No caso de implementação de instruções com um C.Op. de tamanho variável, há a possibilidade de redução de espaço ocupado na MP, já que permite a codificação de um número maior de instruções usando uma menor quantidade de bits.

Esse tipo de implementação permite maior versatilidade entre as quantidades de bits do código de operação e as dos campos operandos, objetivando criar um conjunto de instruções com maior número de instruções, com quantidades diferentes de operandos, sem aumentar em demasia o tamanho total das instruções.

Tipos de Instruções

Um computador deve ter um conjunto de instruções que permita ao usuário formular qualquer tarefa de processamento de dados. Outra maneira de determinar esse conjunto de instruções é considerar os comandos disponíveis em uma linguagem de alto nível (Basic, Pascal, Fortran, C, etc.), partindo-se da premissa de que qualquer programa em uma linguagem de alto nível deve ser traduzido para uma linguagem de máquina para que possa ser executado.

Portanto, o conjunto de instruções da máquina deve ser suficiente para expressar qualquer comando de uma linguagem de alto nível.

Podemos, então, classificar as instruções de máquina nos seguintes tipos, dependendo da sua função:

- **Processamento de dados:** instruções aritméticas e lógicas;
- **Armazenamento de dados:** instruções de memória;
- **Movimentação de dados:** instruções de E/S;
- **Controle:** instruções de teste e desvio.

Tipos de Instruções

Instruções aritméticas são aquelas que fornecem a capacidade computacional para processamento de dados numéricos.

Instruções lógicas (ou booleanas) operam sobre bits de uma palavra, na condição de bits e não de números, oferecendo, portanto, a capacidade de processar qualquer outro tipo de dado (quantitativo ou qualitativo) que o usuário possa desejar empregar, sem o estabelecimento de relações matemáticas ou algébricas entre esses dados.

Instruções de memória são aquelas utilizadas para mover dados entre a memória e os registradores da UCP, uma vez que operações aritméticas e lógicas são executadas sobre dados armazenados nesses registradores.

Instruções de E/S são necessárias para transferir programas e dados para a memória (provenientes de fontes externas à UCP) e para transferir resultados de processamentos computacionais de volta para o usuário.

Instruções de teste são aquelas utilizadas para testar o valor de uma palavra de dados ou o estado de uma etapa de processamento computacional.

Instruções de desvio são usadas para desviar a execução do programapara uma nova instrução, muitas vezes em função do resultado de um teste.

Tipos de Operandos

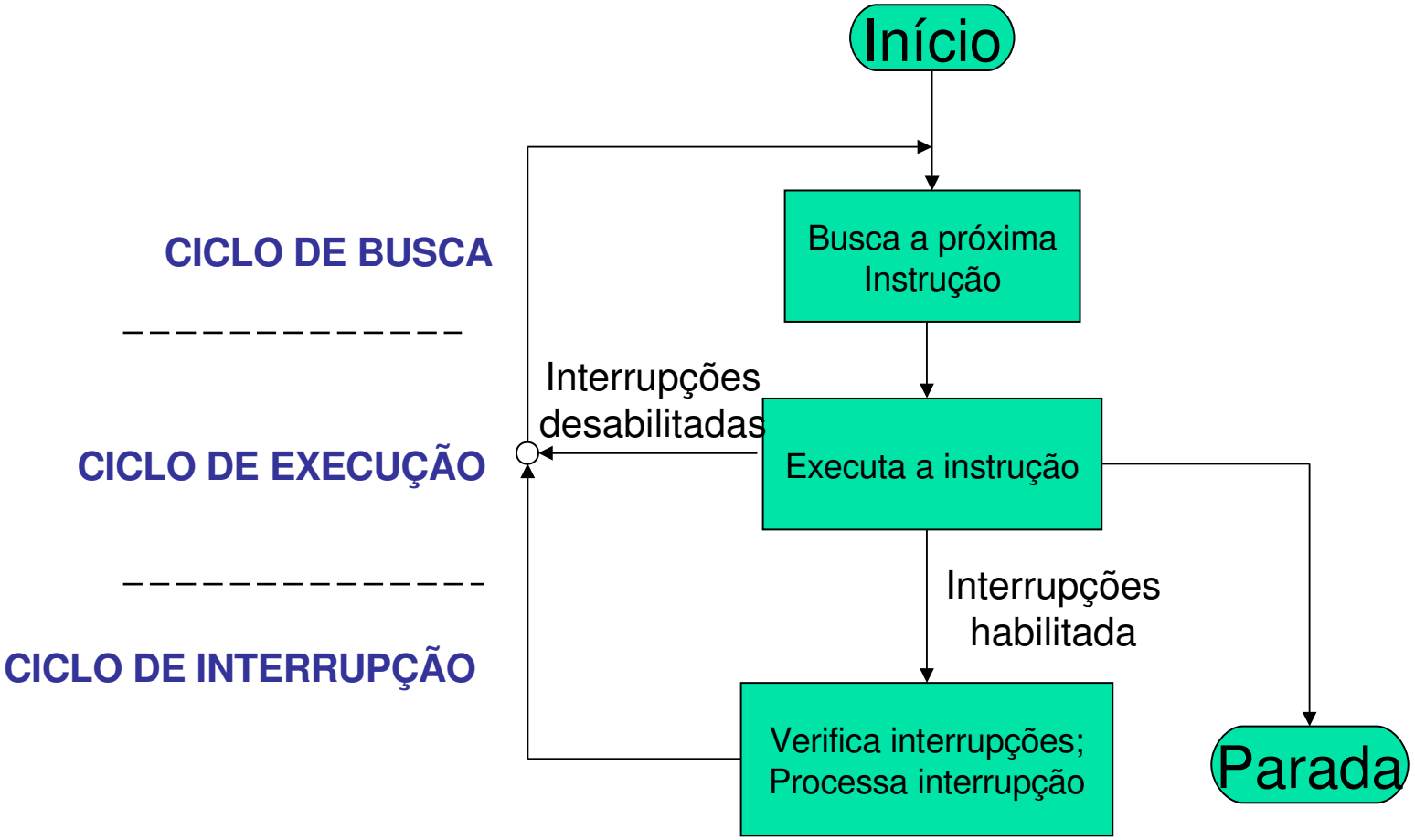
Endereços: são a representação de uma origem (registrador ou posição de memória) ou destino de onde um dado deve ser buscado ou para onde deve ser levado e armazenado.

Números: são utilizados no processamento de dados numéricos e não numéricos, tais como valores de contadores, tamanhos de campos, etc. (embora não sujeitos a processamento numérico, são representados por números). Em sistemas computacionais há duas limitações: quanto à magnitude de representação e quanto à precisão. São comuns três tipos de dados numéricos, números decimais, números inteiros ou de ponto fixo, e números de ponto flutuante, os quais estão sujeitos a efeitos do arredondamento, do *overflow* e do *underflow*.

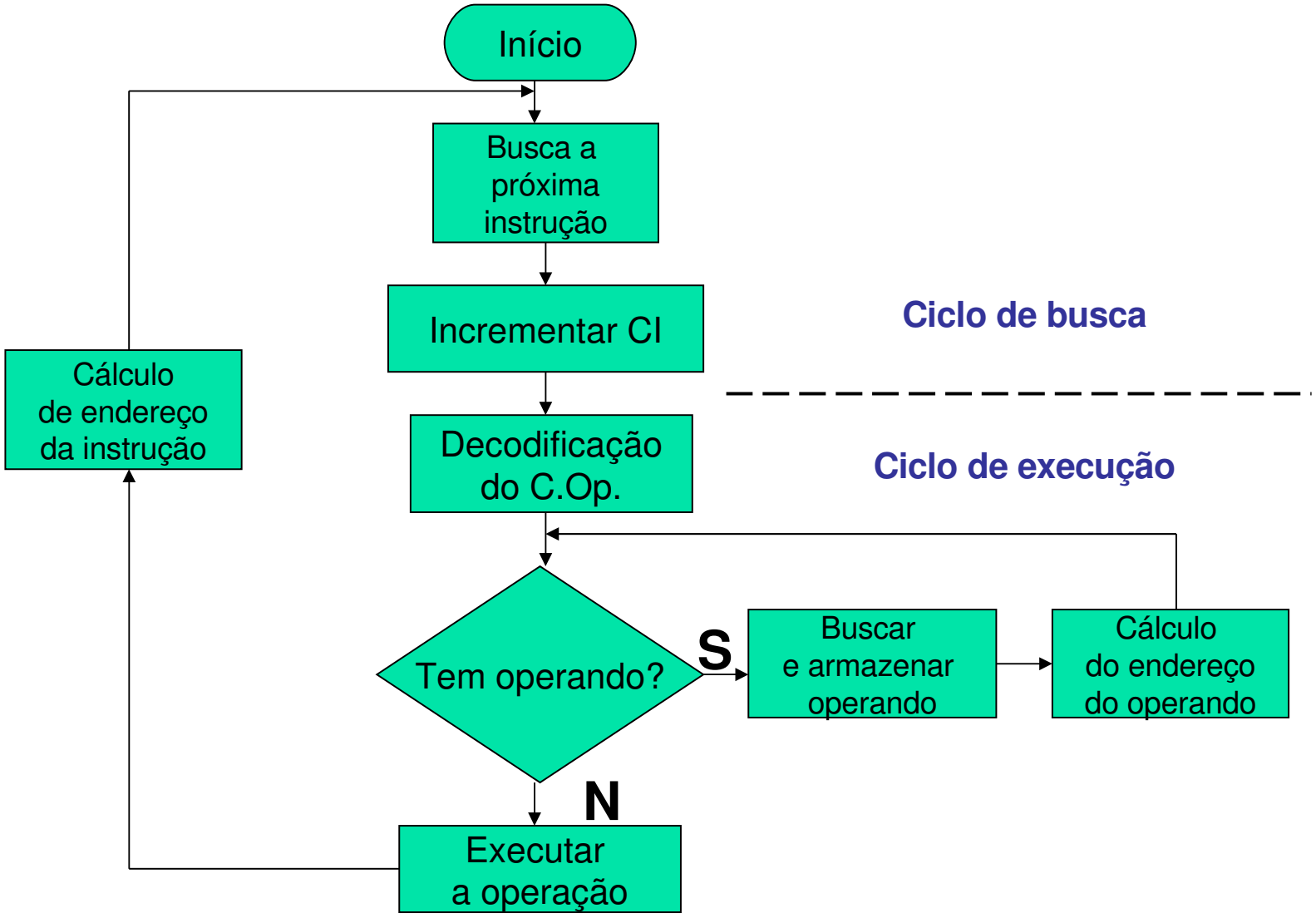
Caracteres constituem a representação de dados textuais, os quais são convertidos em sequências de bits para manipulação em sistemas computacionais. O código mais utilizado é o ASCII (American Standard Code for Information Interchange), onde cada caractere é representado por uma sequência de 7 bits, o que permite a representação de um conjunto de 128 caracteres diferentes.

Dados lógicos são a consideração de uma sequência de n bits como sendo uma sequência de n dados lógicos independentes, que podem assumir os valores 0 ou 1, para a execução de operações booleanas.

Ciclo de Instrução



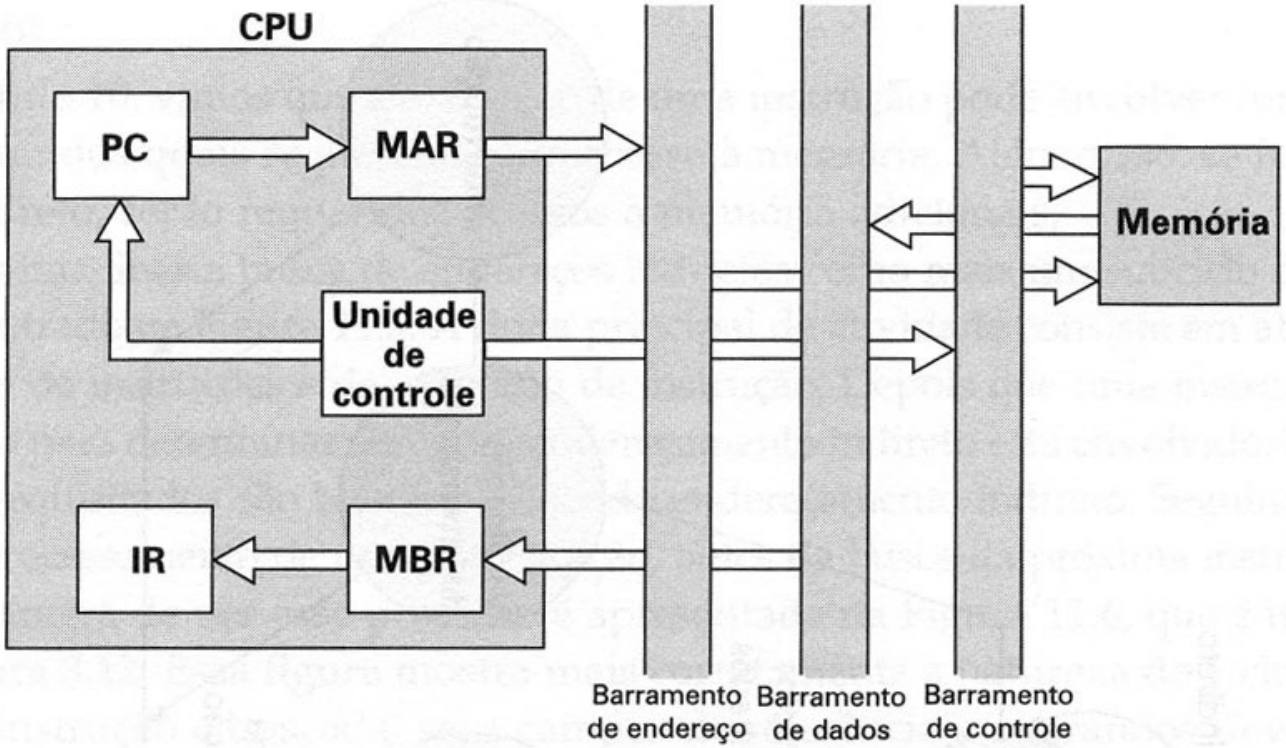
Ciclo de Instrução sem Interrupção



Ciclo de Instrução

- **Busca (leitura)**
  - Lê a próxima instrução da memória para a UCP;
- **Decodifiação**
  - Interpreta o código de operação e seus operandos (se existir).
- **Execução**
  - Efetua a operação indicada.
- **Interrupção**
  - Se as interrupções estão habilitadas e ocorreu uma interrupção, salva o estado do processo atual e processa a interrupção.
- ***Exemplo de execução de um ciclo de instrução:***
  - Iniciar
  - RI←(CI) Busca a instrução, cujo endereço está no CI
  - (CI) ← (CI)+1 Atualiza o CI para a próxima instrução;
  - Interpretar o C.Op.
  - Buscar Op. (Se houver)
  - Executar a instrução
  - Retornar

Ciclo de Busca



MBR = Registrador de armazenamento temporário de dados

MAR = Registrador de endereço de memória

IR = Registrador de instruções

PC = Contador de programa

Fluxo de dados no ciclo de busca.

Ciclo de Busca

No ciclo de busca, temos os seguintes registradores envolvidos:

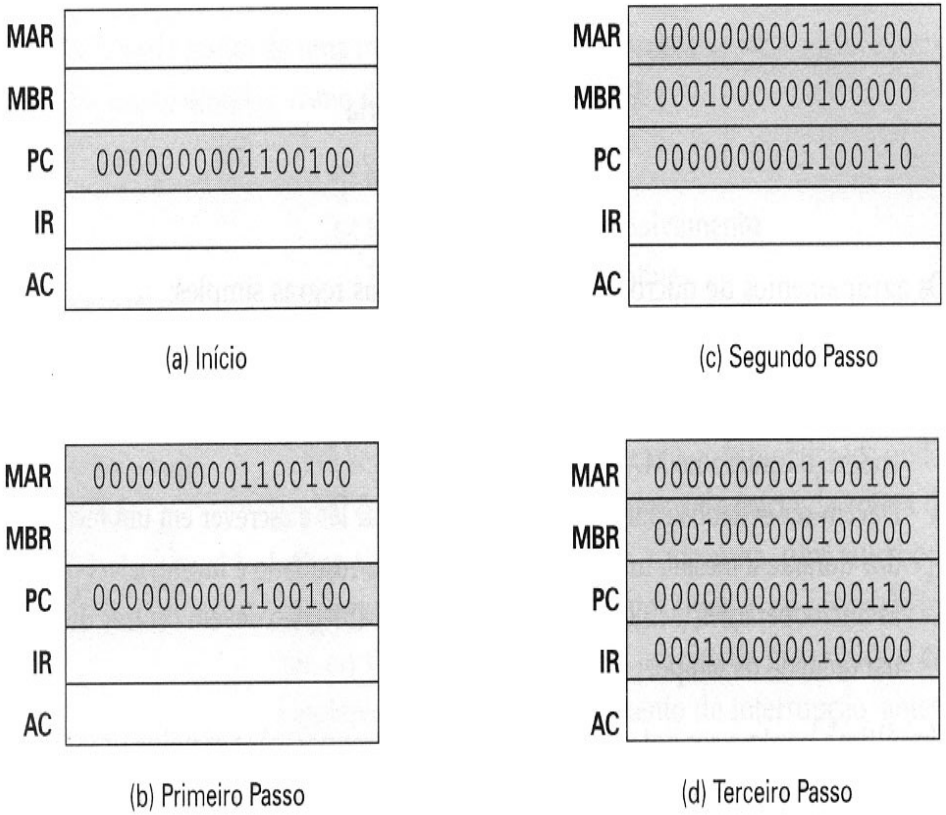
Registrador de Endereço de Memória (MAR – Memory Address Register): **Especifica o endereço de memória para leitura ou escrita.**

Registrador de Armazenamento Temporário de Dados (MBR – Memory Data/Buffer Register): **Contém um valor a ser armazenado na memória ou o último valor lido da memória.**

Contador de Programa (PC – Program Counter): **Mantém o endereço da próxima instrução a ser buscada na memória.**

Registrador de instruções (IR – Instruction Register): **mantém a última instrução buscada na memória.**

Ciclo de Busca



Seqüência de eventos do ciclo de busca.

- **(a) PC contém o endereço da próxima instrução a ser executada.**
- **(b) MAR recebe o endereço de PC, pois ele é o único que tem acesso às linhas de endereço do barramento de sistemas.**
- **(c) A UC gera um comando de leitura, os dados disponibilizados no barramento de dados são copiados para a MBR.**
- **(c) Neste mesmo passo é efetuado o incremento do PC (duas ações que não interferem uma na outra podem ser executadas simultaneamente).**
- **(d) Move o conteúdo da MBR para a IR.**

Ciclo de Interrupção

Ao final do ciclo de execução é feito um teste para verificar se ocorreu alguma interrupção habilitada. Caso tenha ocorrido é executado um ciclo de interrupção. A natureza deste ciclo varia muito de uma máquina para outra. Então vamos apresentar uma seqüência de eventos simples.

t_1: MBR ← (PC)
t_2: MAR ← Endereço de salvamento
PC ← Endereço de rotina
t_3: Memória ← (MBR)

No primeiro passo MBR recebe o conteúdo de PC para o retorno da interrupção.
No segundo passo MAR recebe o endereço onde o conteúdo de PC deve ser salvo e o PC recebe o endereço de início da rotina de tratamento de interrupções.
No terceiro passo a memória recebe do MBR o valor antigo do PC.

Ciclo de Execução

No ciclo de execução podemos ter diversos tipos de microoperações. Vejamos alguns exemplos:

• ADD R1, X

Soma o conteúdo de uma posição de memória X ao registrador R1.

t_1: MAR ← (IR(endereço))

t_2: MBR ← Memória

t_3: R1 ← (R1) + (MBR)

• ISZ X

A posição da memória X é incrementada de 1. Se o resultado for 0 a próxima instrução será saltada.

t_1: MAR ← (IR(endereço))

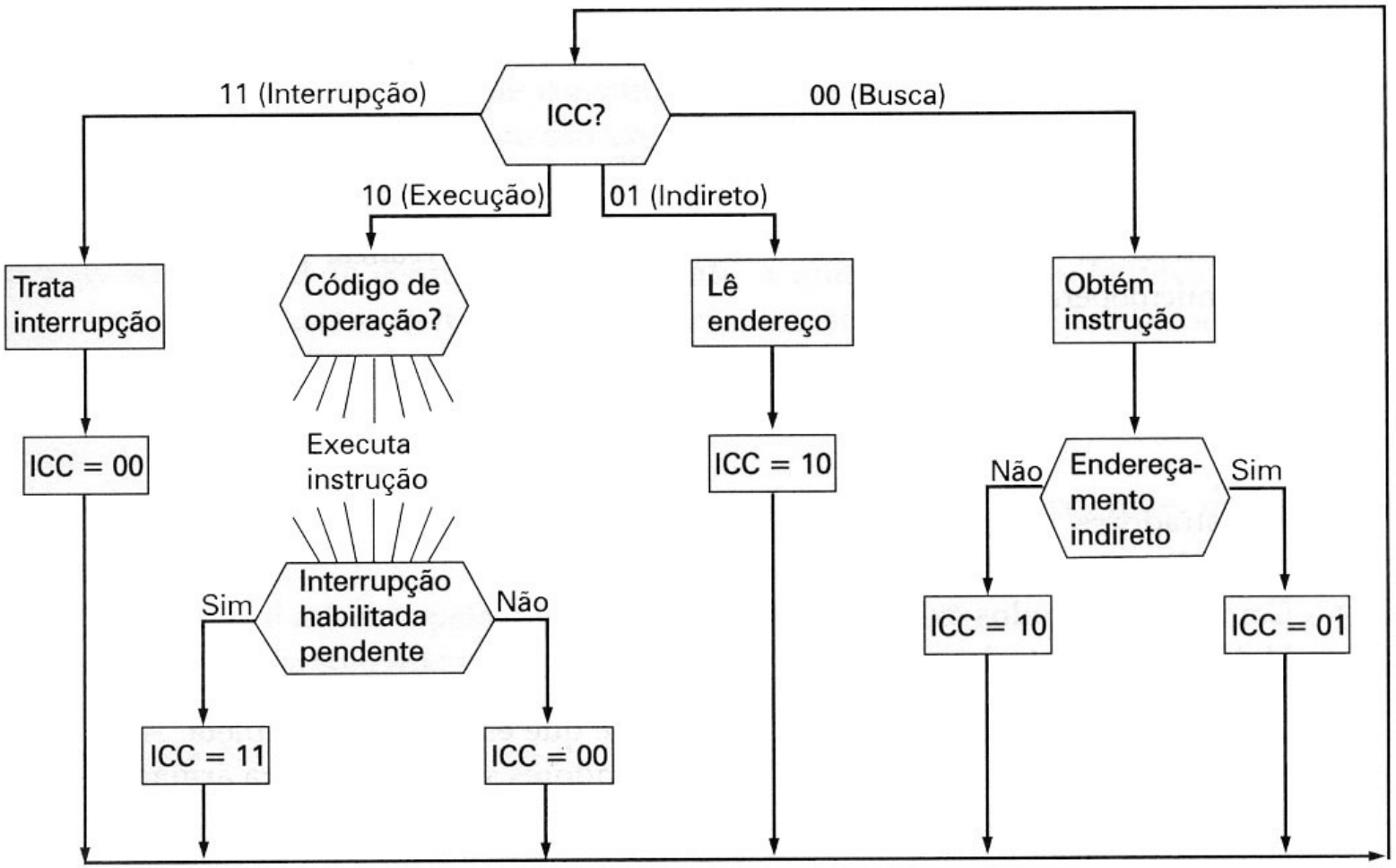
t_2: MBR ← Memória

t_3: MBR ← (MBR) + 1

t_4: Memória ← (MBR)

If ((MBR) = 0) then (PC ← (PC) + I)

Ciclo de Instrução



Fluxograma do ciclo de instrução.

Modos de Endereçamento de Instruções

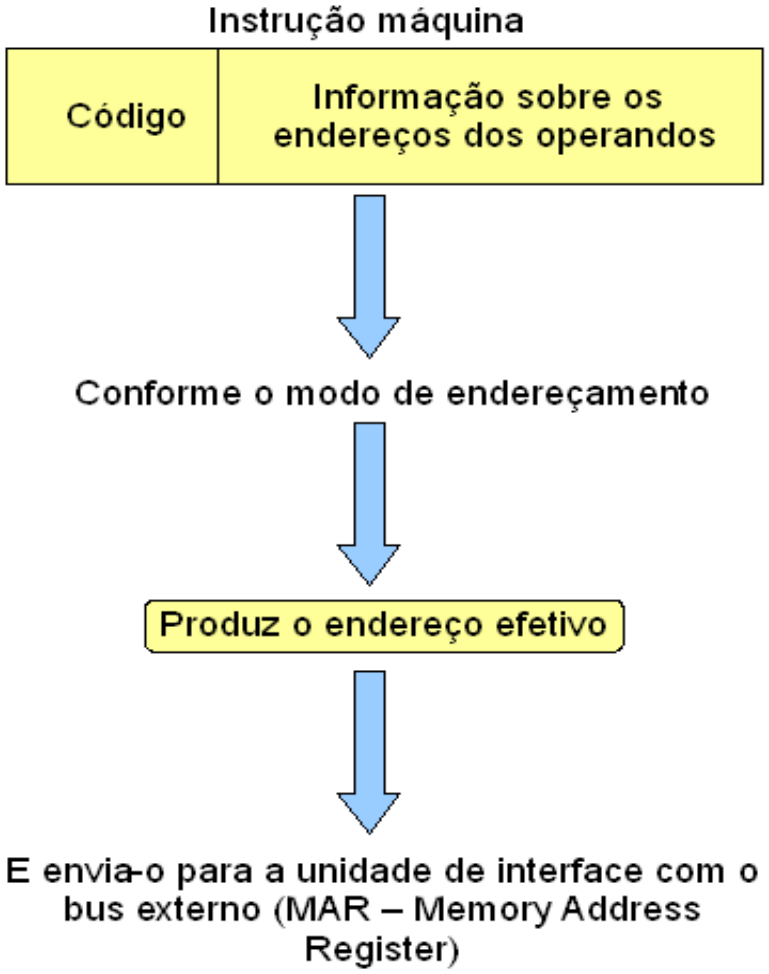
As instruções podem especificar o modo de acesso às células de memória, sendo o endereço dessa célula calculado na própria execução da instrução. Os principais objetivos dos modos de endereçamento de memória são:

- Possibilitar um acesso mais eficiente a certas estruturas de dados, como matrizes, listas e pilhas;
- Especificar um endereço completo de memória utilizando um número menor de bits no campo de endereço da instrução;
- Calcular endereços relativos à posição da instrução corrente ou a outra posição de base, permitindo assim que um programa fique independente da zona de memória central onde é carregado pelo Sistema de Operação.

Modos de Endereçamento de Instruções

Endereço Efetivo

Dado um endereço, sabemos que bytes acessar na memória. Os modos de endereçamento especificam constantes e registradores, além de posições na memória. Quando uma posição de memória é utilizada, o endereço de memória real especificado pelo modo de endereçamento é chamado de **endereço efetivo.**



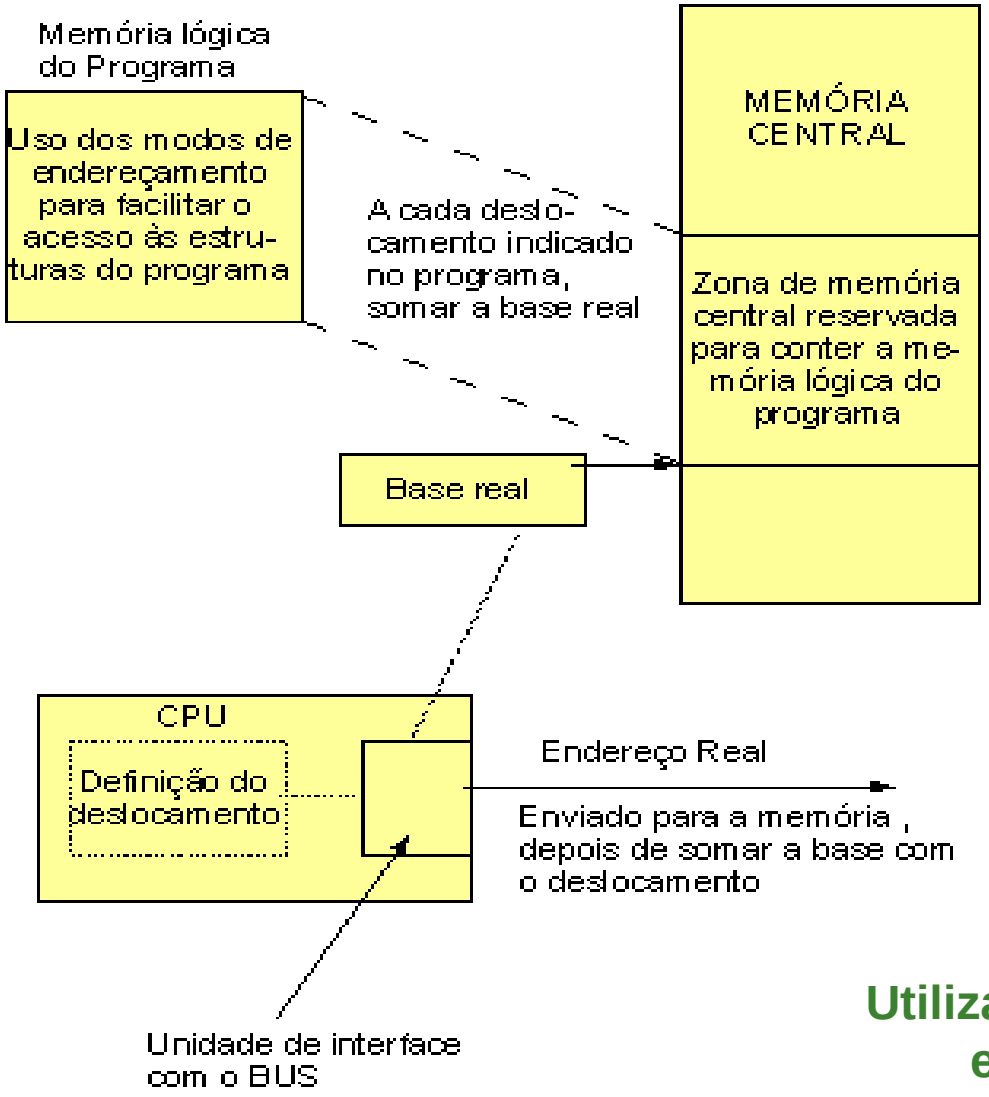
Modos de Endereçamento de Instruções

Há, habitualmente, uma grande variedade de modos de endereçamento, com o objetivo de facilitar o acesso às constantes, às variáveis simples, às estruturas, bem como a geração de endereços relativos, que permitam ao programa ficar independente da posição real de memória onde é carregado.

Podemos distinguir dois aspectos principais:

- o primeiro aspecto refere-se aos modos de endereçamento tal como são utilizados pelo programador ou pelo compilador, mas ainda assumindo uma memória cujo endereço inicial é 0;
- o segundo aspecto refere-se ao modo como o Sistema de Operação, uma vez determinado o endereço real inicial do programa em memória, permite que, a partir do momento em que carrega o programa em memória, as suas instruções possam ser executadas sem ter de alterar o programa.

Modos de Endereçamento de Instruções



Utilização dos modos de endereçamento

Endereçamento Imediato

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

OPERANDO = A

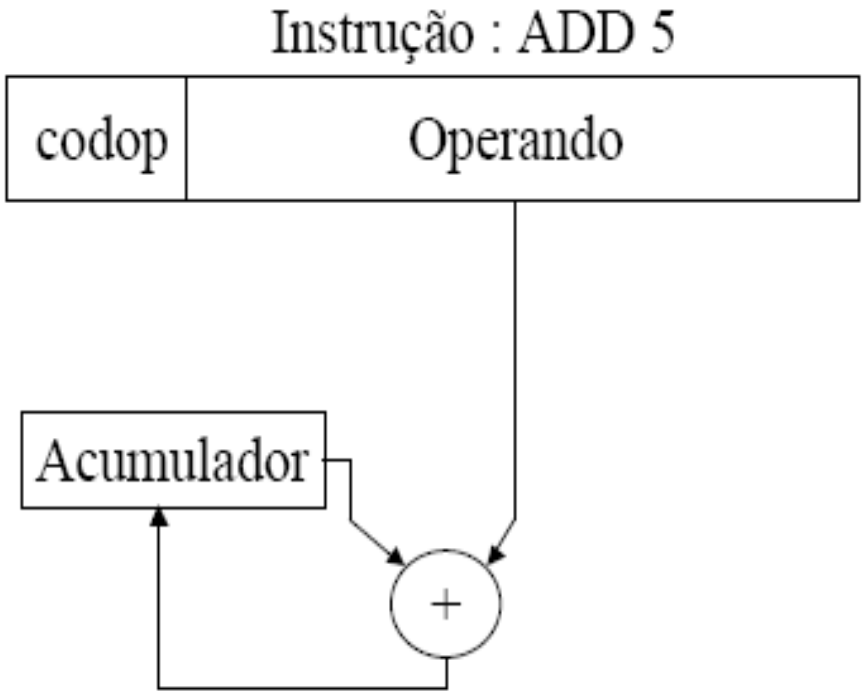
Este modo pode ser usado para definir e usar constantes ou atribuir valores iniciais às variáveis. Tipicamente, o número é guardado em complemento de dois; o bit mais à esquerda do campo do operando é usado como bit de sinal. Quando o operando é carregado num registro de dados, o bit de sinal estende-se para a esquerda até ao máximo tamanho da palavra de dados.

Endereçamento Imediato

A vantagem do endereçamento imediato é que não há nenhuma referência à memória além da extração da instrução necessária para obter o operando, dessa forma, poupando o ciclo de memória ou de *cache* no ciclo de instrução. A desvantagem é que o tamanho do campo de endereçamento é restringido ao tamanho do campo de endereço, o qual, em grande parte dos jogos de instruções é pequeno quando comparado com o tamanho da palavra.

```
-------------------------------
|  Código   |  Operando  |
-------------------------------
```

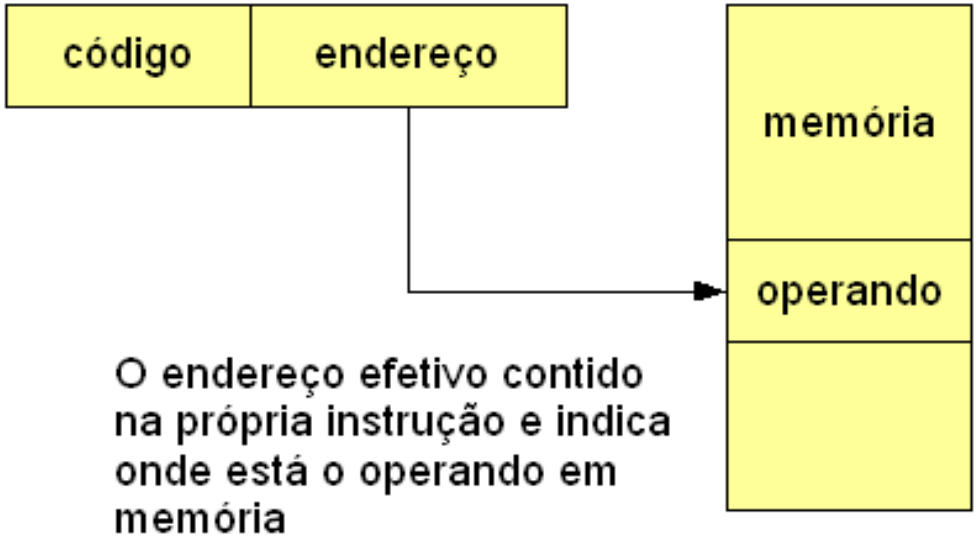
Endereçamento Imediato



Exemplo

Exemplo de endereçamento imediato. Adicionar 5 ao conteúdo do acumulador

Endereçamento Direto

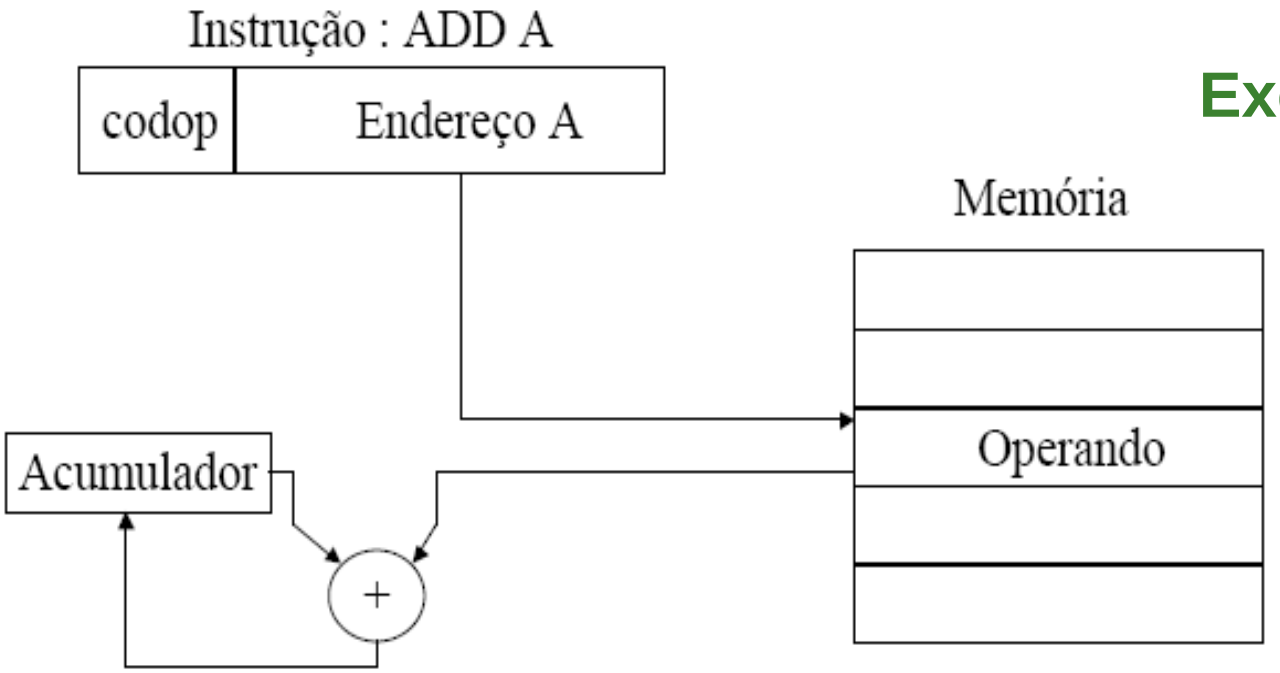


Se o Endereço indicado na instrução é o próprio endereço real do operando, então este modo de endereçamento é também designado por *absoluto* e o programa fica vinculado a só poder ser executado se for carregado na zona de memória que ele referencia. Por outro lado, este modo exige que o campo Endereço tenha um número de bits suficiente para endereçar um domínio razoável de posições de memória, o que pode originar formatos de instruções muito longas.

Endereçamento Direto

Por exemplo, no caso extremo do modo direto e absoluto, em que o campo Endereço tivesse 32 bits, numa máquina com um bus de endereços de 32 linhas, a instrução ocuparia mais de 4 bytes. Uma forma de diminuir o tamanho da instrução, sem prejudicar o alcance do endereço, é utilizar outro modo de endereçamento, que permite que, na instrução, apenas se especifique uma parte do endereço. Este modo de endereçamento também implica que, quando se escreve o programa, se saiba o endereço onde estará localizado o operando. Em muitos casos, tal endereço só se conhece durante a própria execução do programa.

Endereçamento Direto

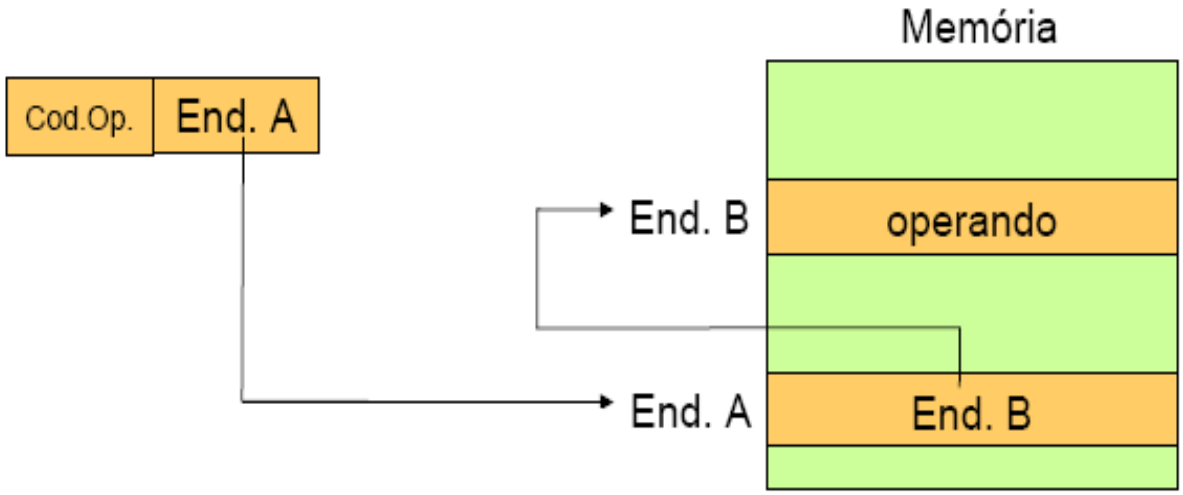


Exemplo

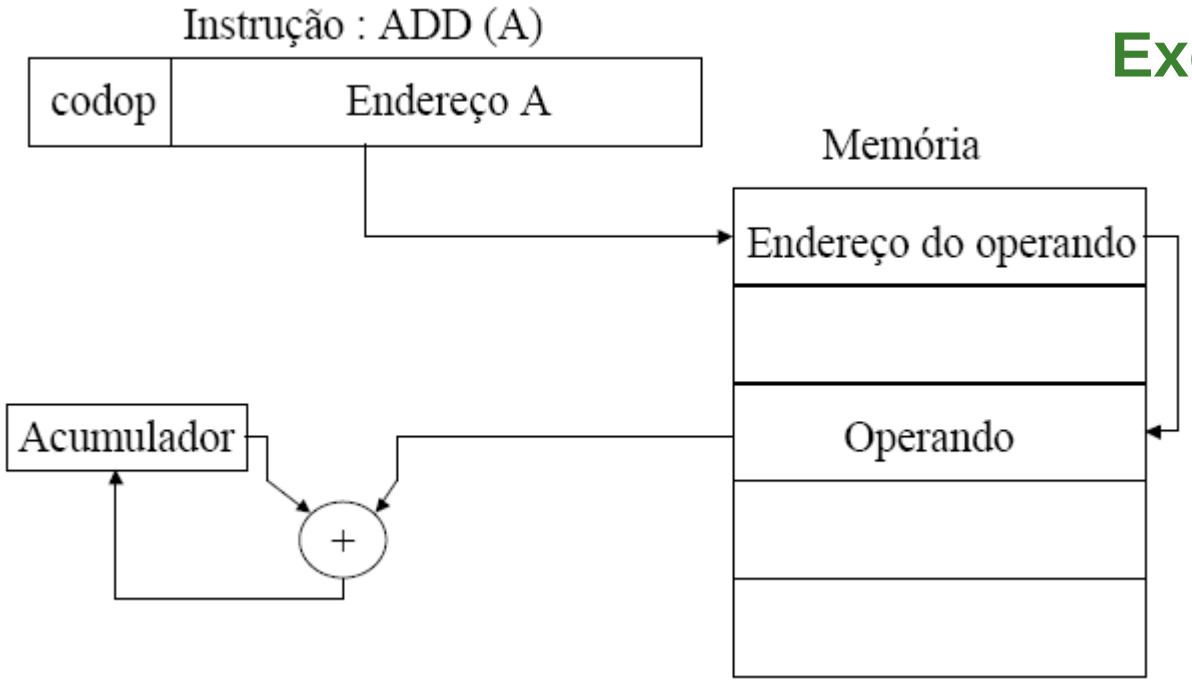
No exemplo acima, se procura pelo operando na posição A da memória e adiciona o conteúdo da posição A da memória no acumulador.

Endereçamento Indireto

Nesse modo de endereçamento, o campo de endereço da instrução contém um endereço da memória cujo conteúdo é o endereço do operando na memória.
Portanto, há um duplo endereçamento. O endereço intermediário é chamado ponteiro (“*pointer*”)



Endereçamento Indireto



Exemplo

Busca em A, encontra o endereço do operando (A) e Busca em (A) pelo operando

- Adiciona o conteúdo do endereço efetivo ao acumulador

Endereçamento Indireto

- **Principais vantagens:**
  - Espaço de endereçamento grande;
  - Permite implementar estruturas de organização de dados mais complexas, mais sofisticadas;
  - Elimina a limitação de células endereçáveis.

- **Principais desvantagens:**
  - Acessos múltiplos à memória
  - Requer maior quantidade de acessos à MP para completar o ciclo de execução da instrução, acarretando que o tempo requerido para a execução da instrução é maior.

Endereçamento via Registrador

Existem outros modos de endereçamento, que usam registradores para indicar a posição onde estão os dados. Os modos de endereçamento direto e indireto por registrador funcionam de forma semelhante aos modos de endereçamento direto e indireto vistos anteriormente (em que o operando aponta para uma posição de memória), porém o operando aponta para um registrador (onde está o dado - endereçamento direto - ou então faz referência à memória - endereçamento indireto).

Vantagens

Maior velocidade / rapidez de execução - o acesso ao registrador é muito mais rápido que o acesso à memória.

Economia de espaço de armazenamento de instrução (o tamanho da instrução é menor porque como são poucos registradores, são menos bits para seus endereços).

Desvantagens

Não são adequados para transferência de variáveis da MP para ULA.

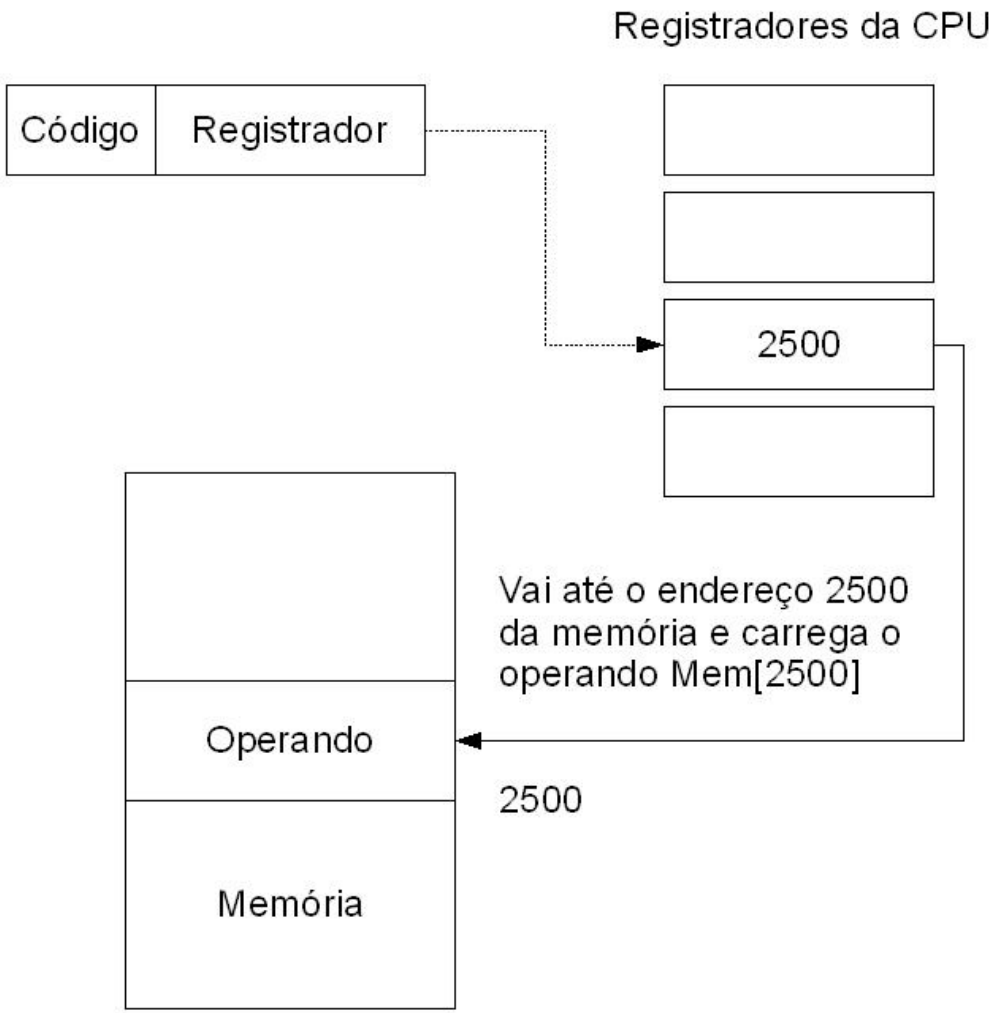
Pequeno número de registradores - se forem muitos os dados endereçados por registrador, os registradores disponíveis podem não ser suficientes.

Endereçamento via Registrador - Direto

A figura abaixo ilustra este modo, que permite que o operando esteja contido num dos registradores da CPU. Assim, na instrução só se indica o número do registrador e durante a execução do programa é que se carregará aquele registrador com o operando. A instrução é carregada diretamente ao operando, dentro da CPU, tornando o acesso muito rápido. No entanto, por haver um número limitado de registadores na CPU, este modo só possibilita o acesso, a cada momento, a um número pequeno de registradores.

Endereçamento via Registrador - Indireto

Neste modo, um registrador da CPU contém o endereço efetivo, sendo utilizado como apontador para a estrutura a ser carregada na memória.
Pode-se observar no exemplo que o registrador aponta para um endereço da memória.



Endereçamento via Registrador - Indireto

Neste modo também, pode ser conveniente dispor de uma função automática de incremento ou decremento do registrador, antes ou depois de usado. Em alguns processadores esta função pode aplicar-se a qualquer registrador usado como apontador, enquanto em outros processadores esta função pode apenas aplicar-se a certos registradores e só quando estes são usados em certas instruções:

Por exemplo, no processador 8086, o uso dos registadores SI e DI, nas instruções de tipo 'move string';

Ou também a função da flag DF 'direction flag', para definir o incremento ou decremento daqueles registradores.