Organização de Computadores 1

5 – CONJUNTO DE INSTRUÇÕES

Prof. Luiz Gustavo A. Martins

Introdução

- O que é um conjunto de instruções?
  - Coleção completa das instruções que a CPU é capaz de executar (entende).
  - **Linguagem de máquina**.
  - É uma **linguagem numérica** (seqüência de bits).

- Linguagem de montagem (*Assembly*) é uma **representação simbólica** do conj. instruções.
  - Tradução é feita por um **programa montador** (**1:1**).

- Linguagens de alto-nível devem ser convertidas em linguagem de máquina para sua execução.
  - Tradução por **compiladores ou interpretadores** (**1:N**).

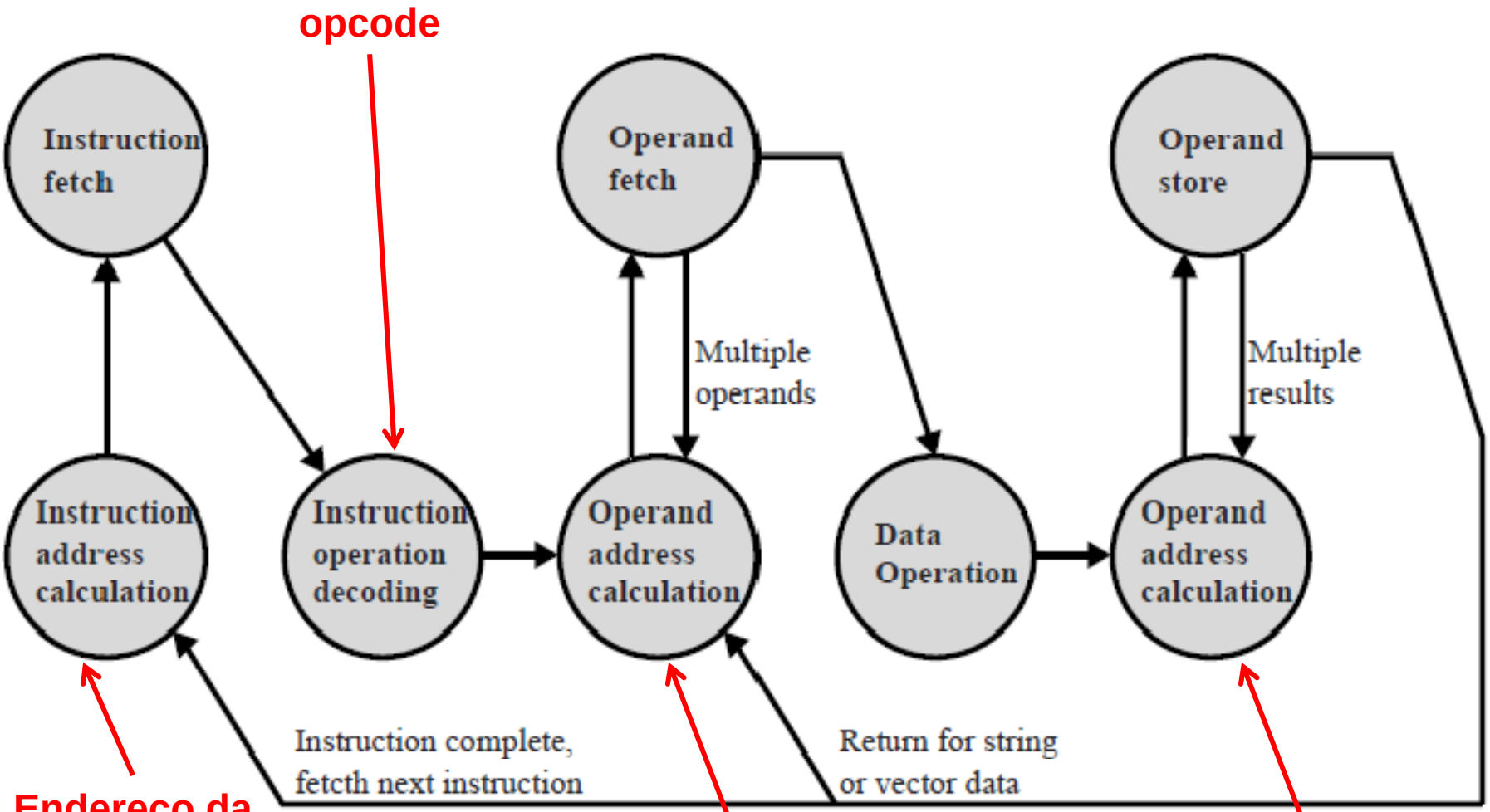
Introdução

- Conjunto de instruções da máquina constitui o **limite entre o projetista de computador e o projetista de compiladores (programador)**.
  - **Projetista:** CPU deve ser concebida para executar as operações do conjunto de instruções.
  - **Programador:** compilador deve converter adequadamente programas em códigos de máquina.
- Um bom conjunto de instruções visa **facilitar a implementação dos projetistas**.

Elementos de uma Instrução

- Toda instrução deve conter as informações necessárias para que a CPU possa executá-la.
  - **Código de operação (*opcode*)**
  - **Referência ao operando-fonte**
  - **Referência ao operando-destino**
  - **Endereço da próxima instrução**

Ciclo de Instrução



Elementos de uma Instrução

- Operandos fonte e destino podem estar localizados nas seguintes áreas:
  - **Memória principal ou virtual**
  - **Registrador da CPU**
  - **Dispositivo de E/S**
- A **próxima instrução** pode estar na memória principal ou virtual.

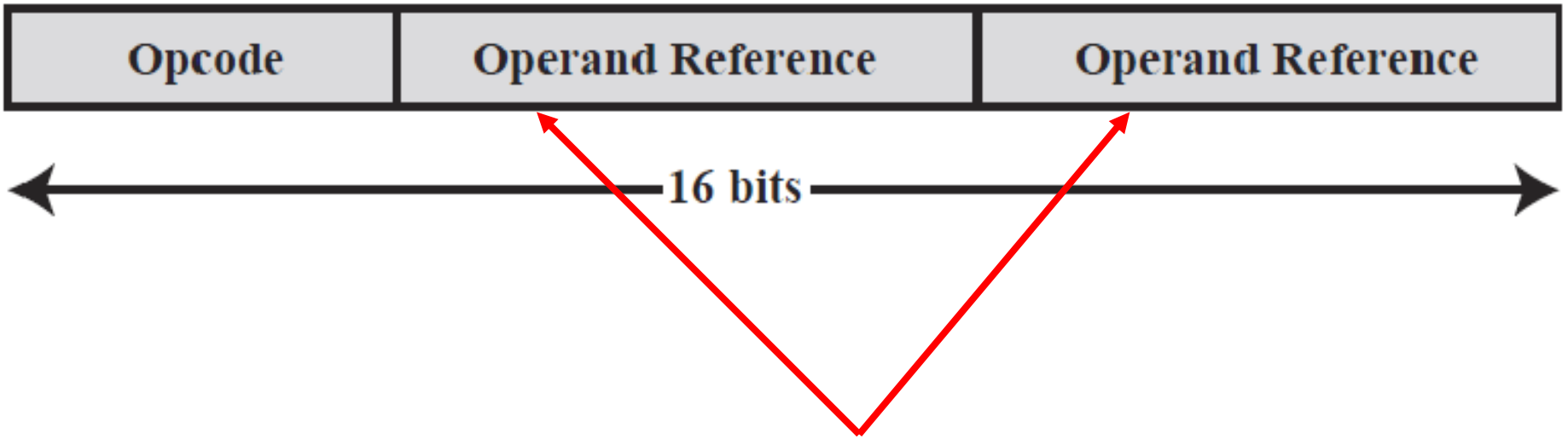
Representação de Instruções

- Cada instrução é representada como uma **seqüência de bits**.
  - Cada código de máquina tem um **padrão único de bit**.
  - Para melhor compreensão dos programadores é utilizada uma **representação simbólica**.

- Instruções são divididas em campos, de acordo com os seus elementos (**formato da instrução**):

Exemplo do Formato de Instrução

Identifica o que será feito (Qual operação)



Define os parâmetros de entrada e/ou saída (quantidade depende do tipo da operação)

Tipos de Instruções

- Conjunto de instruções deve **permitir formular qualquer tarefa** de processamento de dados.

- Instruções de máquina podem ser agrupadas em:
  - **Processamento de dados:** operações aritméticas e lógicas.
    - Operam somente sobre **dados armazenados em registradores**.
  - **Armazenamento de dados:** instruções de memória.
  - **Movimentação de dados:** instruções de E/S.
  - **Controle do fluxo de dados:** instruções de teste e desvio.

Número de Endereços

- Considerando os elementos de uma instrução, pode-se haver **até 4 endereços em um operação**:
  - Máximo de 2 endereços dos operandos de entrada.
  - 1 endereço do operando de saída.
  - 1 endereço da próxima instrução.

- Na prática, temos **instruções com 1, 2 ou 3 endereços**.
  - Endereço da próxima instrução está implícito.

- Instruções com **3 endereços**:
  - Especifica endereços **para 2 operandos e para o resultado**.
    - **Ex:** ADD A, B, C A = B + C
  - **Não são comuns**.

Número de Endereços

- Instruções com **2 endereços**:
  - Especifica endereços **para os 2 operandos de entrada**.
    - **Ex:** ADD A, B A = A + B
  - **Vantagem:** reduz tamanho da instrução.
  - **Desvantagem:** aumenta o nº de instruções/operação.

- Instruções com **1 endereço**:
  - Comum nos primeiros computadores (**ex:** IAC).
  - Especifica apenas o endereço de **1 dos operandos de entrada**.
    - **Endereço implícito para o outro operando e o resultado**.
    - **Ex:** ADD B ACC = ACC + B

- Instruções com **0 endereços**:
  - **Todos os endereços estão implícitos**.
    - **Ex1:** Operações baseadas em registradores ou pilha.

Número de Endereços: Exemplo Programas

Instruction		Comment
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) Three-address instructions

Instruction		Comment
MOVE	Y, A	Y ← A
SUB	Y, B	Y ← Y – B
MOVE	T, D	T ← D
MPY	T, E	T ← T × E
ADD	T, C	T ← T + C
DIV	Y, T	Y ← Y ÷ T

(b) Two-address instructions

Instruction		Comment
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 ÷ Y
STOR	Y	Y ← AC

(c) One-address instructions

Programas para executar a expressão:
Y = (A – B) ÷ (C + D × E)

Quantos Endereços?

- Questão importante de projeto:
  - **Mais endereços:**
    - Instruções maiores e mais complexas.
    - Mais registradores de propósito geral
    - Programas menores (menos instruções)
  - **Menos endereços:**
    - Instruções menores e mais primitivas.
    - Mais instruções por programa.
    - Ciclo de instrução mais rápido.
- Máquinas modernas usam 2 ou 3 endereços.

Projeto do Conjunto de Instruções

- Afeta diversos aspectos do sistema.
  - Efeito significativo sobre a implementação da CPU.
  - Deve considerar as necessidades do programador.

- **Decisões importantes do projeto:**
  - Repertório de operações.
  - Tipos de dados.
  - Formatos de instruções.
  - Registradores.
  - Endereçamento.

- **Questões altamente inter-relacionadas.**

- **CISC X RISC.**

Tipos de Operandos

- Instruções de máquina operam sobre dados.
- **Tipos mais importantes:**
  - **Endereços**.
  - **Dados numéricos**.
    - Inteiro e ponto flutuante.
    - Decimal (BCD).
  - **Caracteres**.
    - Representação por seqüência de bits (ex: códigos ASCII e EBCDIC).
  - **Dados lógicos**.
    - Unidade de *N* bits corresponde a *N* itens de dados com valores 0 ou 1.
- **Outros dados possíveis:**
  - Estruturas de dados (**listas**).
  - Seqüência de caracteres (***string***).

Tipos de Operandos

- Existe diferença entre o caractere 6 e o número 6?
  - Número 6: 00000110
  - Caractere 6: ASCII – 00110110 EBCDIC – 11110110
  - Conversão envolve operação com os bits
    - Operação lógica **AND**.
  - **Problema:** conversão de números com vários dígitos.
    - Ex: 374.

Tipo de Dados do Pentium

- **Agrupamento de dados:**
  - Byte: 8 bits.
  - Palavra: 16 bits.
  - Palavra dupla: 32 bits.
  - Palavra quádrupla: 64 bits.
- Endereçamento é feito por unidades de 8 bits (byte).
  - Disposição de múltiplos bytes: ***little-endian***.
- **Dados não precisam ser alinhados** na memória em endereços divisíveis por 4.

Disposição de Múltiplos Bytes (*Endianness*)

- ***Endianness:*** disposição dos bytes de valores escalares de **múltiplos bytes**.

- **Ex:** Palavra **12345678** (em hexadecimal)

Endereço	Valor 1	Valor 2
184	12	78
185	34	56
186	56	34
187	78	12
Disposição	***Little-endian***	***Big-endian***

- ***Bi-endian***: permite utilizar os 2 modos.

Disposição de Múltiplos Bytes (*Endianness*)

- **Qual padrão utilizar?**

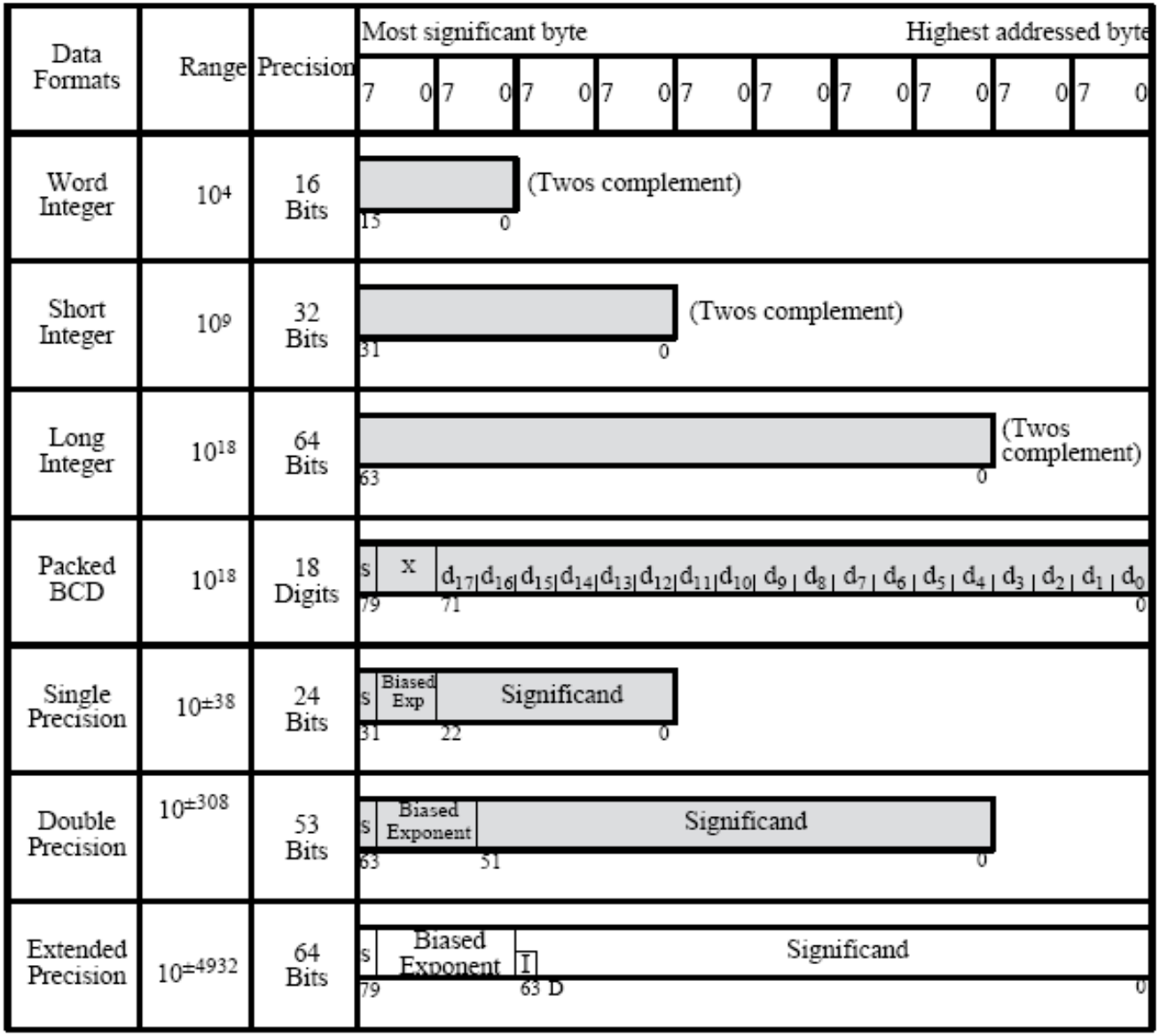
- Não existe um consenso.
  - Existem pontos positivos e negativos para ambos.
    - ***Big-endian:*** + rápido na comparação de seqüência de caracteres.
    - ***Little-endian:*** + fácil efetuar aritmética de alta precisão.
  - **Exemplos:**
    - Pentium (80x86), VAX são ***little-endian.***
    - IBM 370, Moterola 680x0 (Mac) e muitas máquinas RISC são ***big-endian.***

Tipos de Dados Específico

- Diferenciação é feita pelas operações de manipulação.

- **Tipos manipulados:**
  - **Gerais:** conteúdo binário arbitrário.
  - **Inteiros:** valor binário em complemento de 2.
  - **Ordinal:** número inteiro sem sinal.
  - **BCD desempacotado:** 1 dígito decimal por byte.
  - **BCD empacotado:** 2 dígitos decimais por byte.
  - **Ponteiro (*near pointer*) :** endereço efetivo de 32 bits.
    - Um endereço relativo ao início de um segmento.
    - Uma posição em uma memória não segmentada.
  - **Campo de bits:** seqüência contígua de bits.
    - Cada posição é considerada uma unidade independente.
  - **Seqüência de bytes:** seqüência de bytes, palavras ou palavras duplas.
  - **Ponto flutuante:** precisão simples, dupla ou estendida (IEEE 754).

Tipos de Dados Específico



Tipos de Operação

- **Qtde. de *opcode* pode variar** entre máquinas.

- **Classe de operações não varia**:
  - Transferência de dados
  - Aritméticas
  - Lógicas
  - Conversão
  - E/S
  - Controle de sistema
  - Transferência de controle

Operações de Transferência de Dados

- Tipo de operação **mais fundamental**.

- **Especifica:**
  - Origem e destino dos dados.
  - Tamanho dos dados a serem transferidos.
  - Modo de endereçamento dos operandos.

- Quando envolve **endereço de memória**:
  - Determina o endereço.
  - Converte endereço de memória virtual e real.
  - Verifica memória cache.
  - Inicia leitura/escrita na memória.

Operações de Transferência de Dados

- O endereço pode ser indicado tanto pelo *opcode* quanto pela especificação do operando.
  - Pode haver **instruções distintas para movimentos diferentes**.
    - **Ex:** IBM 370
  - Uma instrução e diferentes endereços.
    - **Ex:** VAX

Operações Aritméticas

- **Envolve:**
  - **Transferência de dados** antes e/ou depois da operação.
  - Execução da operação na **ULA**.
  - Atualização dos **códigos de condição** (*flags*).

- **Operações básicas:**
  - Soma
  - Subtração
  - Multiplicação
  - Divisão

- **Outras possíveis operações** (unárias):
  - Incremento (a++) ou decremento (a--) do operando.
  - Negação do operando (-a).
  - Valor absoluto do operando (|a|).

Operações Lógicas

- **Manipulam bits individuais** de qualquer unidade endereçável (**ex:** palavra).

- **Operações básicas:**
  - **NOT** (NÃO)
  - **AND** (E)
  - **OR** (OU)
  - **XOR** (OU-exclusivo)
  - **EQUAL** (teste de igualdade binária)

Operações Lógicas

- Tabela Verdade:

P	Q	NOT P	P AND Q	P OR Q	P XOR Q	P = Q
0	0	1	0	0	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	0
1	1	0	1	1	0	1

Deslocamentos e Rotações

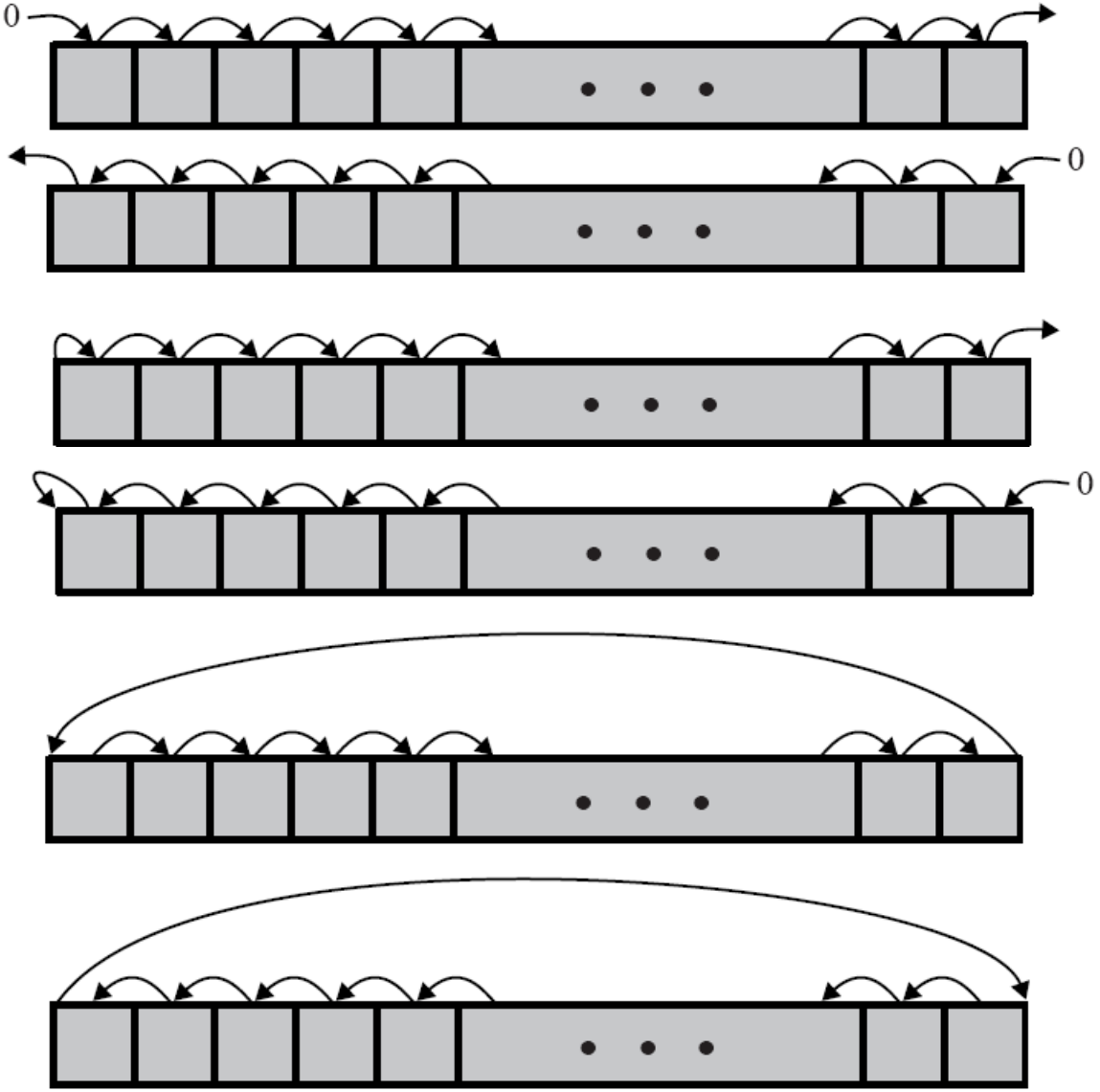
- **Outras operações:**
  - **Deslocamento de bits:** movimentação uma casa para direita ou esquerda.
    - **Lógico:** bit deslocado é perdido e um 0 é inserido na outra ponta
    - **Aritmético:** trata o dado como inteiro com sinal.
  - **Rotação (deslocamento cíclico):** gira os bits para esquerda ou para direita.
    - Mantém todos os bits da palavra.
    - Pode ser usado para teste dos bits de uma palavra.

Deslocamentos e Rotações

Deslocamento Lógico

Deslocamento Aritmético

Rotação



Exercício

- Como transmitir caracteres de dados para um dispositivo de E/S (um por vez), dada uma palavra de memória de 16 bits (2 caracteres/palavra)?

Solução:

1. Carregue a palavra em dois registradores (R1 e R2).
2. Execute a instrução AND entre o registrador R1 e o valor 1111111100000000 (mascarar o caracter à direita).
3. Desloque 8 posições o conteúdo do registrador R1 para a direita (deslocamento lógico).
4. Efetue a E/S.
5. Execute a instrução AND entre o registrador R2 e o valor 0000000011111111 (mascarar o caracter à esquerda).
6. Efetue a E/S.

Operações de Conversão e E/S

- **Operações de conversão** são aquelas que mudam ou operam sobre o formato de dados.
  - **Ex:** conversão de um número decimal para binário.

- **Operações de E/S** realizam a transferência de dados com os dispositivos de E/S.
  - Emite comando para um módulo de E/S.
  - E/S mapeada em memória:
    - Instruções de movimentação de dados.
      - Grande variedade de instruções.
  - E/S mapeada independentemente:
    - Poucos comandos especiais de E/S.
  - Pode ser feita por um controlador separado (DMA).

Operações de Controle do Sistema

- **Instruções privilegiadas**.
  - CPU em estado privilegiado (**modo de núcleo**).
  - CPU executando um programa na área especial (privilegiada) da memória.
- Para **uso pelo Sistema Operacional**.
- **Exemplos:**
  - Leitura ou alteração de um registrador de controle.
  - Leitura ou alteração de uma chave de proteção da memória.
  - Acesso a blocos de controle de processos (multiprogramação).

Operações de Transferência de Controle

- **Altera a seqüência de execução** das instruções.
  - CPU atualiza PC com o endereço de outra instrução da memória.

- **Motivos para o desvio:**
  - **Executar + de 1 vez** um conjunto de instruções.
    - Laço de repetição ou *loop*.
  - **Tomada de decisão**.
    - Executa uma seqüência se a condição for satisfeita ou outra caso contrário (*if*).
  - **Modularização de programas.**
    - Dividir o programa em partes menores.
      - Cada parte Pode ser programada separadamente.

Instruções de Desvio

- Um dos operandos é o **endereço da próxima instrução**.
- Desvio pode ser **para frente** ou **para trás**.
- Permite desvio condicional e incondicional.
  - **Condicional:** desvio só ocorre se condição for satisfeita.
  - **Incondicional:** desvio sempre ocorre.
- **Ex:** BRZ X ; Desvia para instruçao de endereço X se resultado última operação for zero.

Instruções de Salto

- Incluem endereço de desvio implícito.
- Área destinada ao endereço pode ser utilizada para outra finalidade (ex: variável de controle).
  - **Ex:** Incrementa e salta se zero (ISZ)

    301 xxxxxx

    …

    309 ISZ Register1

    310 Branch 301

    311 ADD A

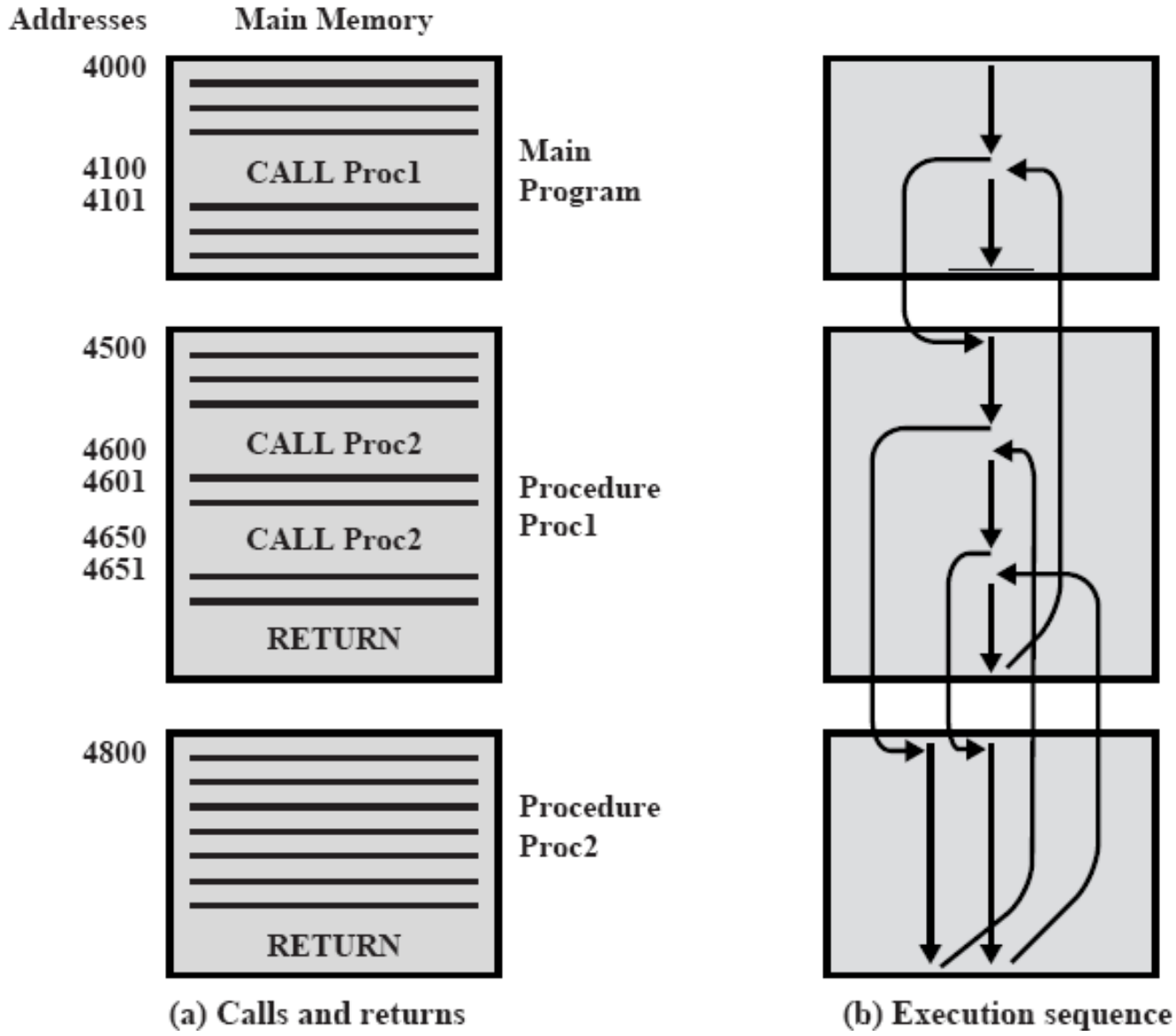
Instruções de Chamada de Subrotina

- **Subrotina ou procedimento:** subprograma incorporado em um programa maior.

- Uma subrotina pode ser chamada de qualquer ponto do programa.
  - **Chamada:** executar subrotina e retornar à instrução seguinte a chamada.

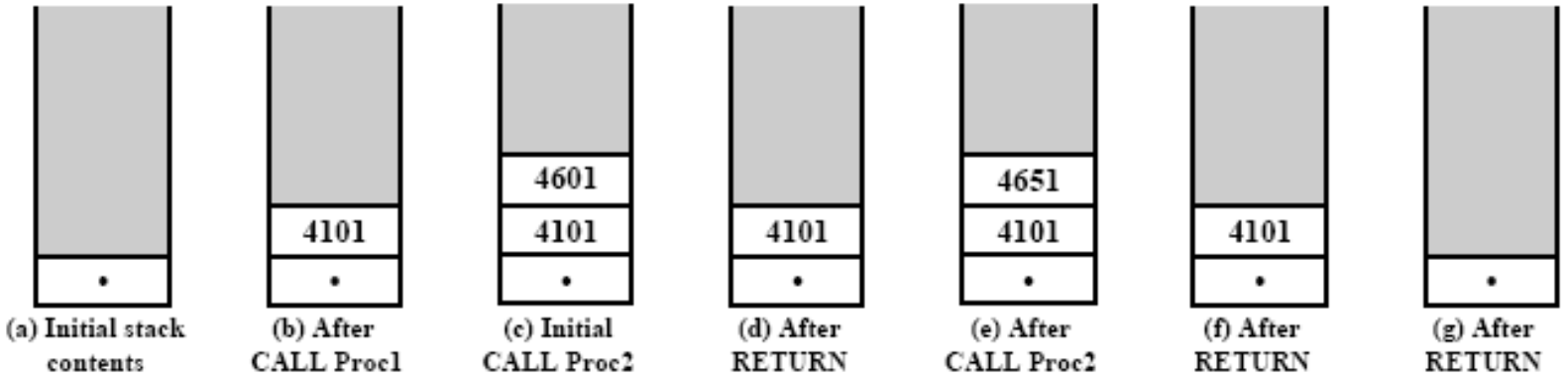
- Vantagens de subrotina:
  - **Economia:** elimina repetição do código
  - **Modularidade:** divisão de grandes tarefas em unidades menores.

Aninhamento de Subrotinas



Instruções de Chamada de Subrotina

- Envolve 2 instruções básicas:
  - **Instrução de chamada:** desvia a execução da instrução corrente para o início da subrotina.
  - **Instrução de retorno:** retorna a execução para o endereço que ocorreu a chamada.

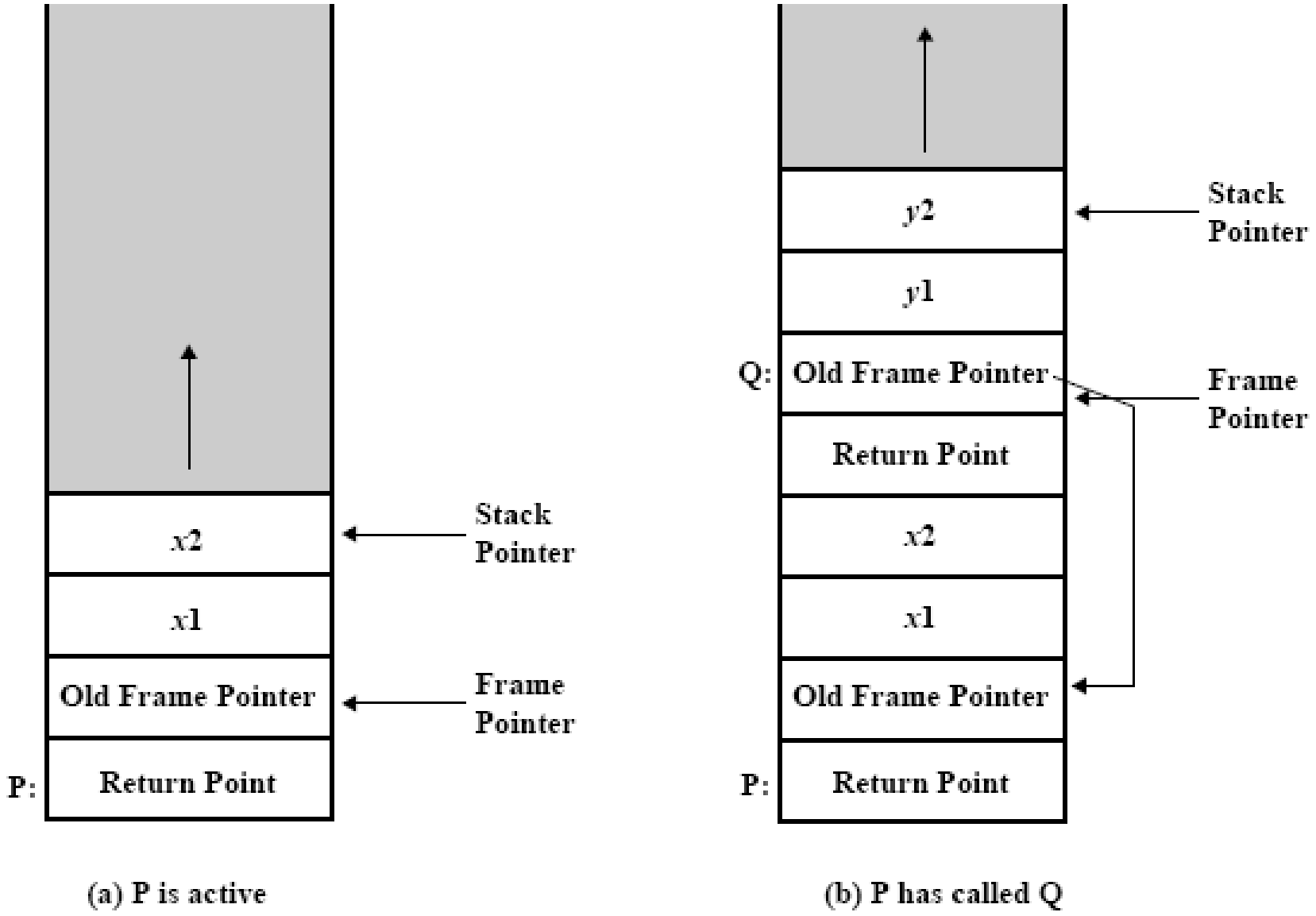


- **Considerações:**
  - Procedimento pode ser **chamado de + de 1 pto** do programa.
  - Chamada de procedimento ocorre dentro de uma subrotina (aninhamento de procedimentos).
  - **Cada chamada corresponde a um retorno**.

Instruções de Chamada de Subrotina

- Endereço de retorno deve ser salvo:
  - **Registrador:** RN = PC+Δ
    PC = X
    - **Problema:** uso de subrotinas aninhadas.
  - **Início da área de memória do procedimento:**
    X = PC+Δ
    PC = X + 1
    - **Problema:** uso de subrotinas reentrantes (ex: recursivo).
  - **Pilha:** armazena PC no topo da pilha.
- **Passagem de parâmetro:**
  - **Registradores e memória**
  - **Pilha (registro de ativação)**

Crescimento da Pilha



Exemplo de Instruções

TABLE 9.3 Common Instruction Set Operations

Type	*Operation Name*	*Description*
Data Transfer	Move (transfer)	Transfer word or block from source to destination
	Store	Transfer word from processor to memory
	Load (fetch)	Transfer word from memory to processor
	Exchange	Swap contents of source and destination
	Clear (reset)	Transfer word of 0s to destination
	Set	Transfer word of 1s to destination
	Push	Transfer word from source to top of stack
	Pop	Transfer word from top of stack to destination
Arithmetic	Add	Computer sum of two operands
	Subtract	Compute difference of two operands
	Multiply	Compute product of two operands
	Divide	Compute quotient of two operands
	Absolute	Replace operand by its absolute value
	Negate	Change sign of operand
	Increment	Add 1 to operand
	Decrement	Subtract 1 from operand
Logical	AND	Perform the specified logical operation bitwise
	OR	
	NOT (Complement)	
	Exclusive-OR	
	Test	Test specified condition; set flag(s) based on outcome
	Compare	Make logical or arithmetic comparison of two or more operands; set flag(s) based on outcome
	Set Control Variables	Class or instructions to set controls for protection purposes, interrupt handling, timer control, etc.
	Shift	Left (right) shift operand, introducing constants at end
	Rotate	Left (right) shift operand, with wraparound end
	Jump (branch)	Unconditional transfer; load PC with specified address
	Jump Conditional	Test specified condition; either load PC with specified address or do nothing, based on condition

Modos de Endereçamento

- Instruções possuem campos de endereço **pequenos**.
- **Referência a grande quantidade de posições** requer o uso de técnicas de endereçamento.
  - Quase todas arquiteturas fornecem **+ de 1** modo de endereçamento.
  - **Questão:** Como UC determina qual modo de endereçamento a ser usado em uma determinada instrução?
  - **Possíveis soluções:**
    - Definição pelo **código de operação** (*opcode*).
    - Inclusão de um **campo de modo de endereçamento** (1 ou + bits) no formato da instrução.
- Escolha envolve algumas **decisões que se contrapõem**:
  - Quantidade de posições endereçáveis.
  - Flexibilidade de endereçamento ao número de referências de cada instrução.
  - Complexidade no cálculo de endereçamento.

Endereçamento Imediato

- Forma **mais simples** de endereçamento.
- Campo de endereço contém o **valor do operando**.
  - **Operando = A**
- Usado para **valores constantes** ou **inicialização de variáveis**.
- **Vantagem: não requer acesso à memória** para buscar o operando.
- **Desvantagem**: **limitação do tamanho** do operando.

Diagrama de Endereçamento Imediato

Instrução

Opcode	**Operando**

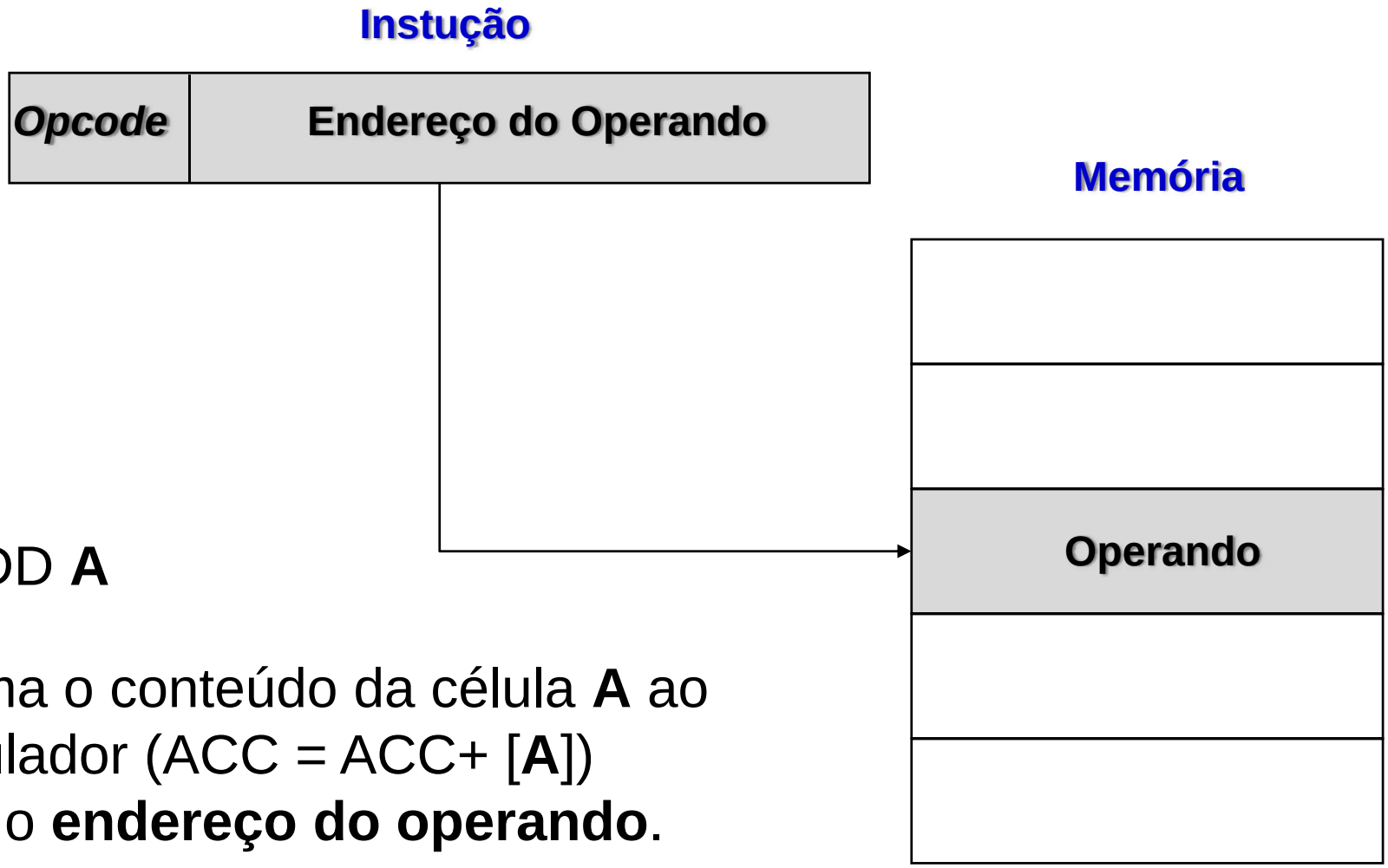
Ex: ADD **5**

- Soma 5 ao acumulador (ACC = ACC+5)
- **5** é o **valor do operando**.

Endereçamento Direto

- Campo de endereço contém o **endereço do operando**.
  - **EA = A**
    - **EA:** endereço efetivo.
    - **A:** conteúdo do campo de endereço do operando.
- Acesso ao conteúdo do operando através de **uma única referência a memória**.
- **Vantagem: simplicidade** do endereçamento.
- **Desvantagem: espaço de endereçamento limitado**.

Diagrama do Endereçamento Direto



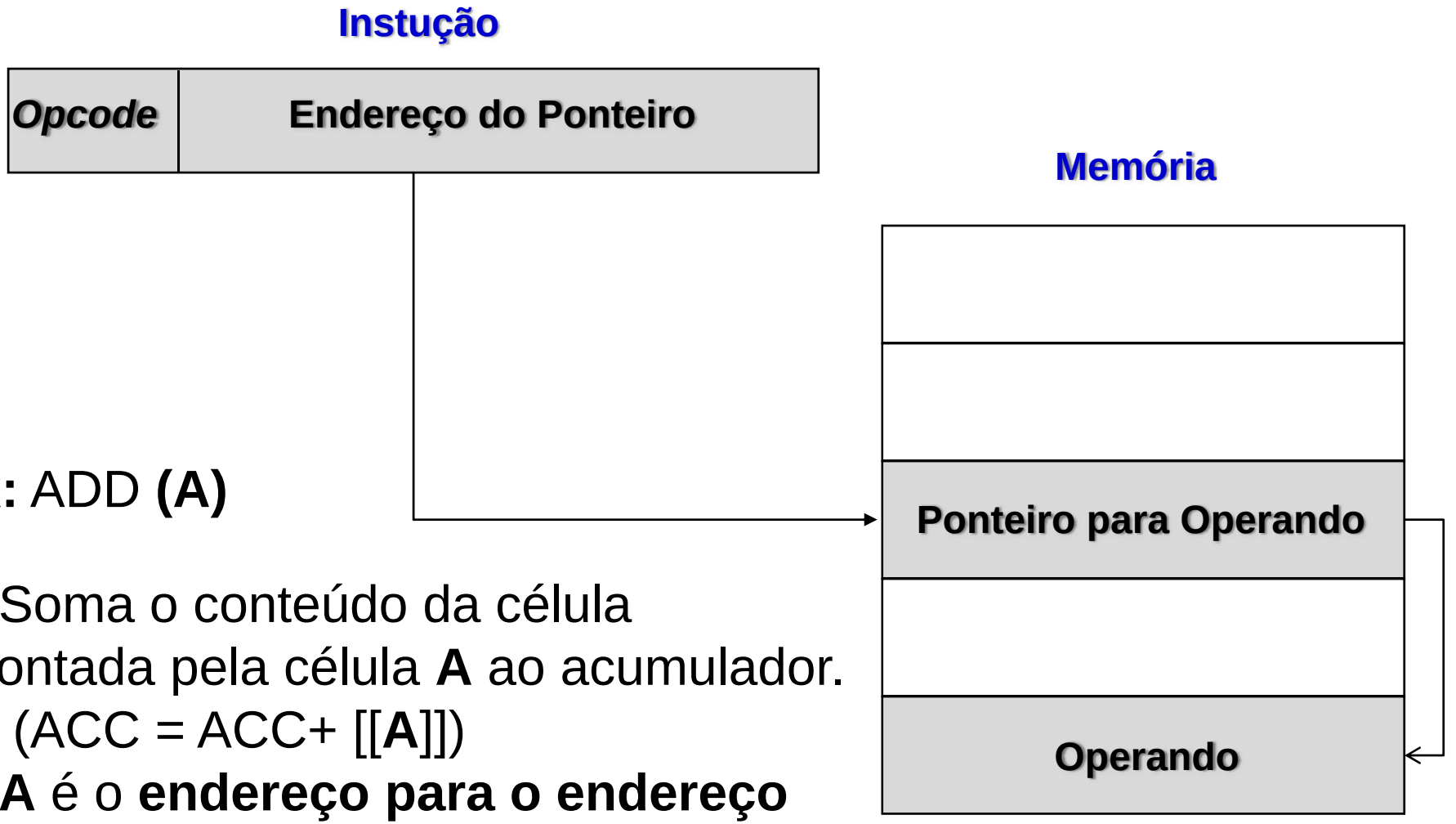
Ex: ADD **A**

- Soma o conteúdo da célula **A** ao acumulador (ACC = ACC+ [**A**])
- **A** é o **endereço do operando**.

Endereçamento Indireto

- Campo de endereço contém um **ponteiro para o endereço do operando**.
  - **EA = (A)**
    - **(A):** conteúdo da célula de memória apontada pelo conteúdo de A.
- **Vantagem:** grande espaço de endereçamento.
  - 2^N onde N = tamanho da célula de memória.
- **Desvantagem:** necessita de 2 acessos a memória.
  - 1° para obter o **endereço** do operando.
  - 2° para obter o **valor** do operando.
- Pode ser empregado multiníveis de endereçamento.
  - **Ex: EA = (...((A))...)**
  - **Uso de bit especial**.

Diagrama do Endereçamento Indireto



Ex: ADD **(A)**

- Soma o conteúdo da célula apontada pela célula **A** ao acumulador.
  (ACC = ACC+ [[**A**]])
- **A** é o **endereço para o endereço do operando**.

Endereçamento por Registrador

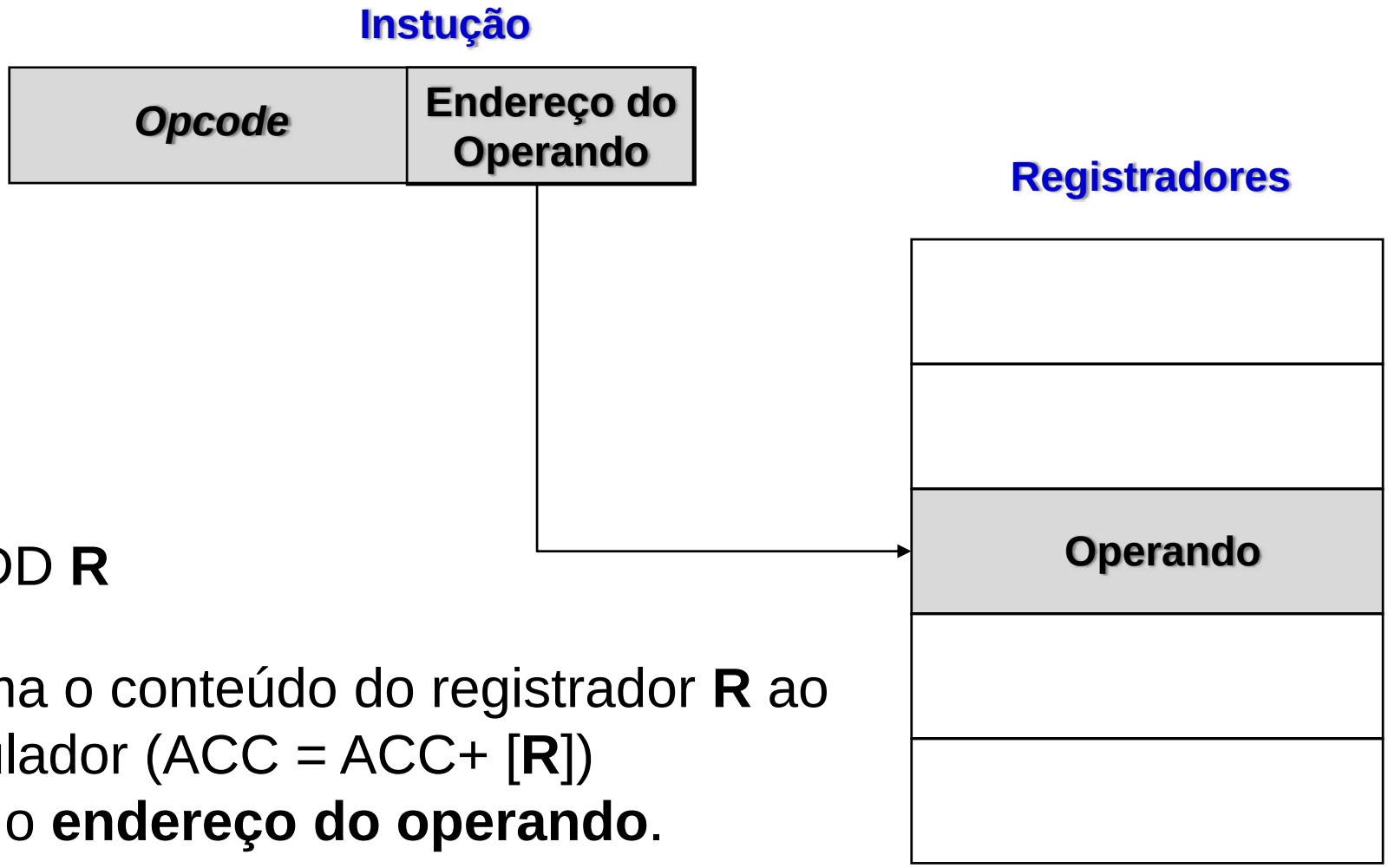
- **Similar ao endereçamento direto**.
  - **Diferença:** campo de endereço **referencia um registrador**.
  - **EA = R**
    - **R:** conteúdo do registrador.

- **Vantagens:**
  - Campo de endereço pequeno.
  - Não requer acesso à memória.

- **Desvantagem:** espaço de endereço muito limitado.

- Múltiplos registradores ajudam o desempenho.
  - Requer boa programação assembly ou compilador eficiente.
  - Programador define quais valores colocar nos registradores e na MP.

Diagrama de Endereçamento por Registrador



Ex: ADD **R**

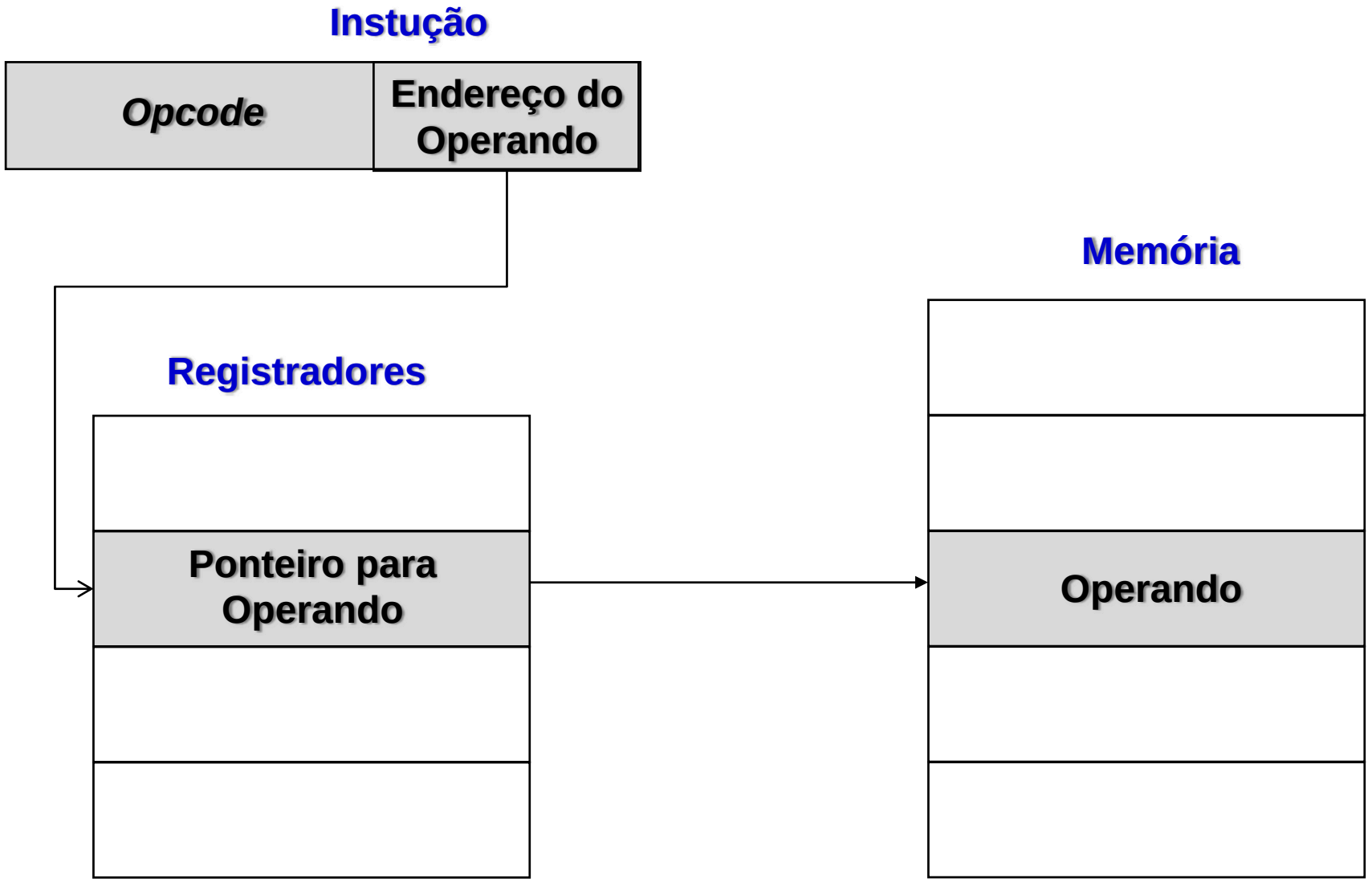
- Soma o conteúdo do registrador **R** ao acumulador (ACC = ACC+ [**R**])
- **R** é o **endereço do operando**.

Endereçamento Indireto por Registrador

- **Similar ao endereçamento indireto**.
  - **Diferença:** campo de endereço **referencia um registrador**.
  - **EA = (R)**
    - **(R):** conteúdo da célula de memória apontada pelo conteúdo do registrador **R**.

- Vantagens e desvantagens também similares ao endereçamento indireto.
  - Requer **um acesso a menos** à memória.

Diagrama do End. Indireto por Registrador



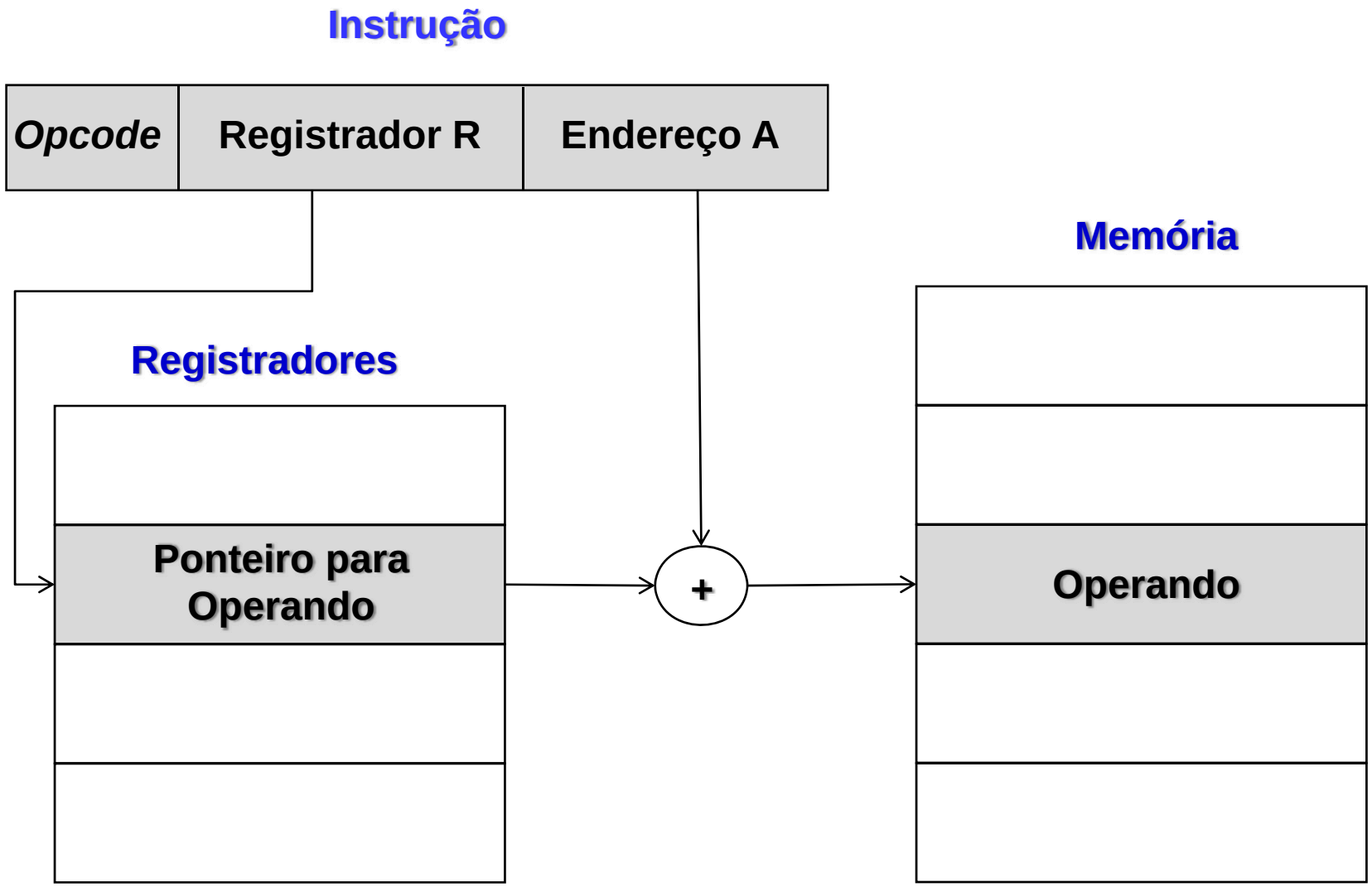
Endereçamento por Deslocamento

- Campo de endereço formado por 2 referências:
  - **A** = Posição de memória.
  - **R** = Registrador.
  - **Pelo menos 1 deve ser explícito**.

- Combina o **endereçamento direto** e o **endereçamento indireto por registrador**.
  - **EA = A + (R)**

- **Vantagem:** flexibilidade de referência.
- **Desvantagem:** complexidade no cálculo do **EA**.

Diagrama do End. por Deslocamento



Tipos de Endereçamento por Deslocamento

- **Endereçamento relativo:** registrador é referenciado implicitamente.
  - **R = PC** (Contador de Programa).
  - **EA = A + (PC)**
  - **EA** é um deslocamento relativo ao endereço da instrução.
- **Endereçamento via registrador-base:**
  - **R** contém **ponteiro para endereço-base**.
  - **A** contém **deslocamento** relativo ao endereço-base
  - Forma de **implementar a segmentação** de memória.
- **Endereçamento indexado:**
  - **A** contém o **endereço-base** na MP.
  - **R** contém um **deslocamento positivo** (**reg. índice**).
  - **EA = A + R**
  - Eficiente para implementar **acesso a vetores**.
  - Alguns sistemas dispõem de **auto-indexação**.

Combinações

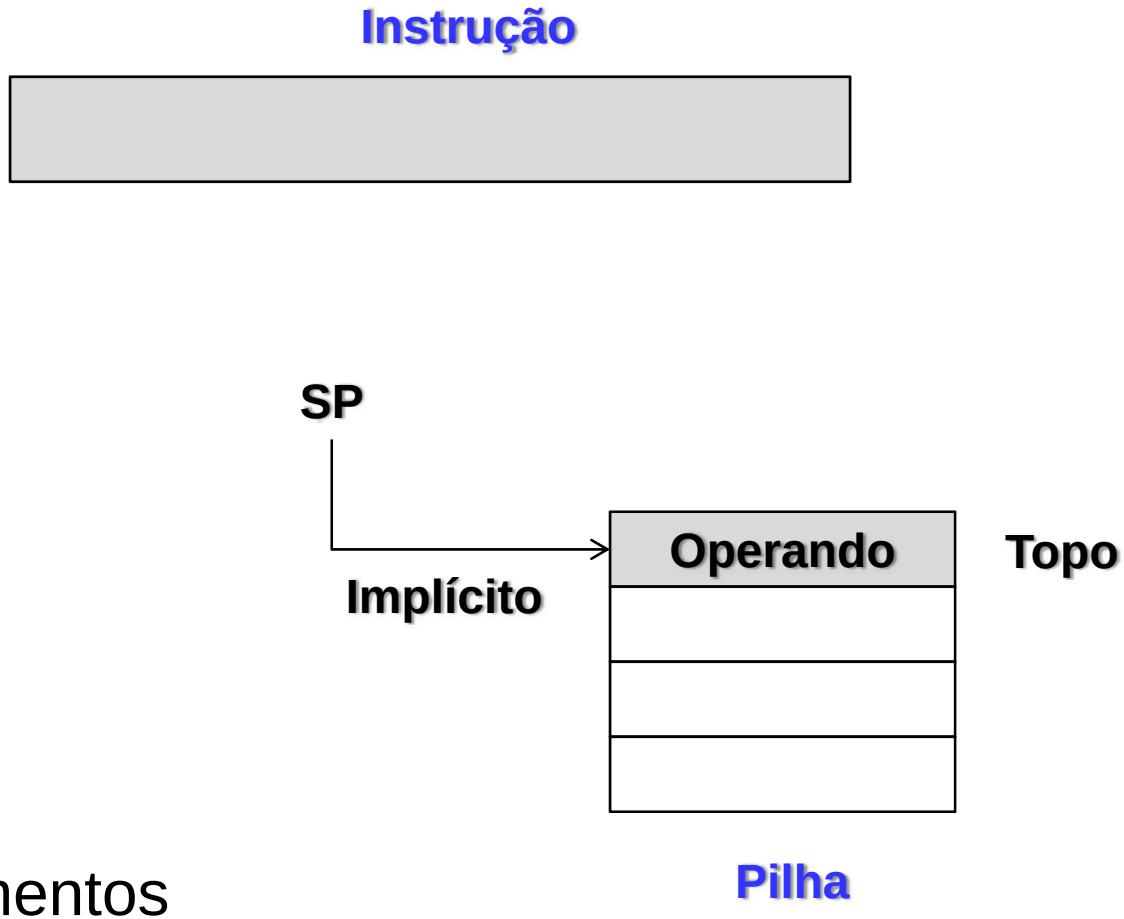
- Uso combinado do endereçamento indireto e indexado:
  - **Pós-indexação:** indexação **após** o endereçamento indireto.
    - **EA = (A) + (R)**
  - **Pré-indexação:** indexação **antes** do endereçamento indireto.
    - **EA = (A + (R))**

Endereçamento a Pilha

- **Pilha:** estrutura do tipo **LIFO** (*Last In, First Out*).
  - Seqüência linear de posições de memória.
  - Possui um registrador que aponta para o topo da pilha (**SP – *Stack Pointer***).

- Operando está sempre no topo da pilha (**implícito**).
  - Como **SP** é um registrador, este modo é um tipo de endereçamento indireto via registrador.

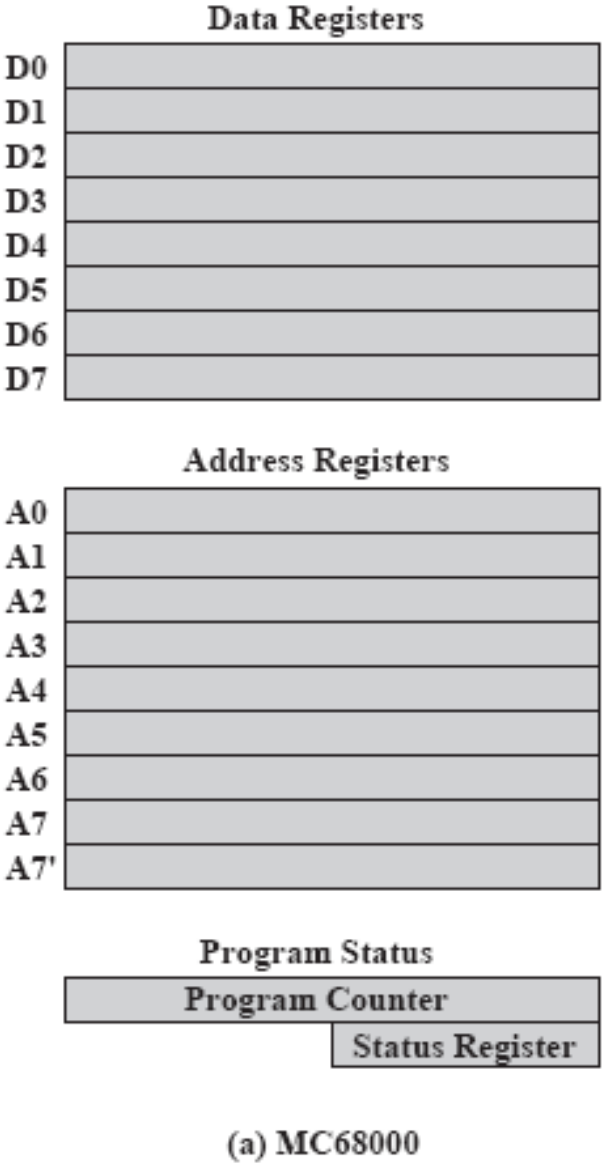
Endereçamento a Pilha



Ex: ADD

- soma os 2 elementos retirados do topo da pilha.

Exemplo da Organização dos Registradores



Formatos de Instrução

- Determina a **disposição dos bits** da instrução.
- Conjunto de instruções tem **+ de 1 formato** de instrução.
- Questões fundamentais do projeto incluem:
  - **Tamanho da instrução**.
  - **Alocação de bits**.

Tamanho de Instrução

- Afeta e é afetada pelo:
  - Tamanho e organização da memória.
  - Estrutura do barramento.
  - Complexidade e velocidade da CPU.

- Conflito entre **repertório de instruções poderosas** e necessidade **de economizar espaço**.

- Considerações:
  - Deve ser **igual ou múltiplo** do tamanho da **unidade de transferência de dados** da MP.
  - Deve ser **múltiplo** do tamanho de um **caractere** e de um número de ponto fixo (**inteiro**).

Alocação de Bits

- Define a relação entre **n° de *opcodes* e capacidade de endereçamento**.
- Fatores que determinam o uso dos bits:
  - **N° de modos de endereçamento.**
  - **N° de operandos.**
  - **Memória ou registrador.**
  - **N° de conjunto de registradores.**
  - **Faixa de endereços.**
  - **Granularidade de endereçamento.**
    - Referenciar **byte ou palavra**.