Arquitetura de Computadores

Prof. Fábio M. Costa
Instituto de Informática – UFG
1S/2004

ISA: Arquitetura de Conjunto de Instruções

Roteiro

Introdução

Classificação de conjuntos de instruções

Endereçamento de memória

Tipos de operações

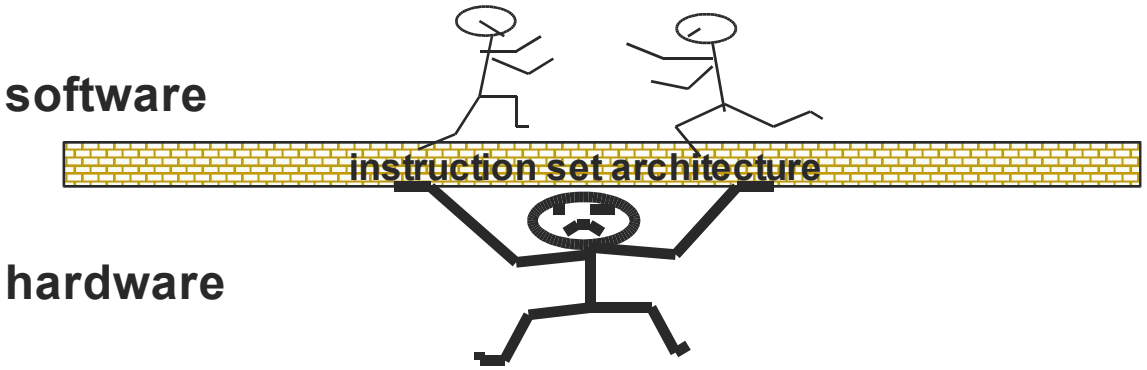
Tipos de dados (operandos)

Formatos de instruções

Controle de fluxo (procedimentos e sub-rotinas, interrupções)

Introdução

ISA: Definição



A parte do computador visível ao programador ou ao implementador de compiladores

Interface entre o Hardware e o Software

Componentes do nível ISA

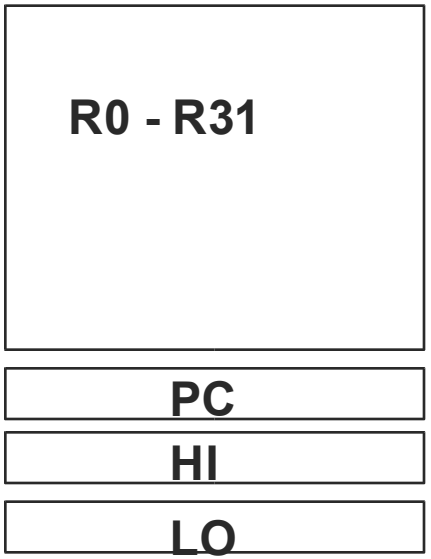
- Conjunto de instruções
- Conjunto de registradores
- Tipos de dados nativos
- Modos de endereçamento da memória
- Esquemas de E/S

Exemplo: ISA do MIPS R3000

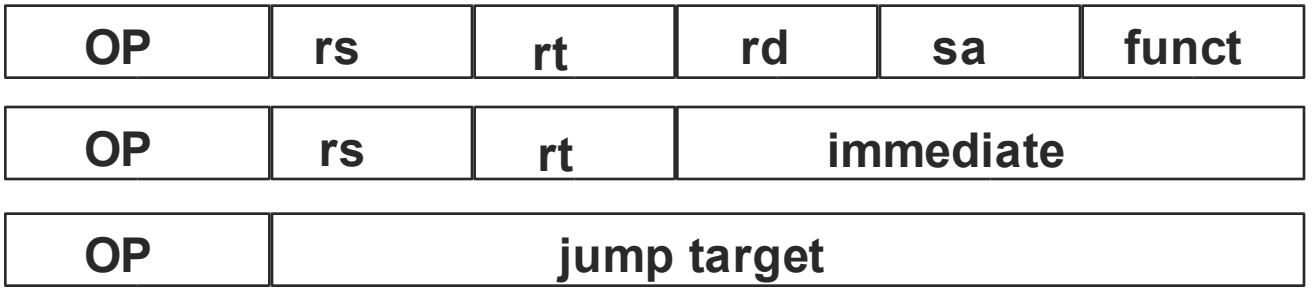
Categorias de instruções

- Load/Store
- Computacionais
- Desvio
- Ponto flutuante
  - co-processador
- Gerenciamento de memória
- Especiais

Registradores

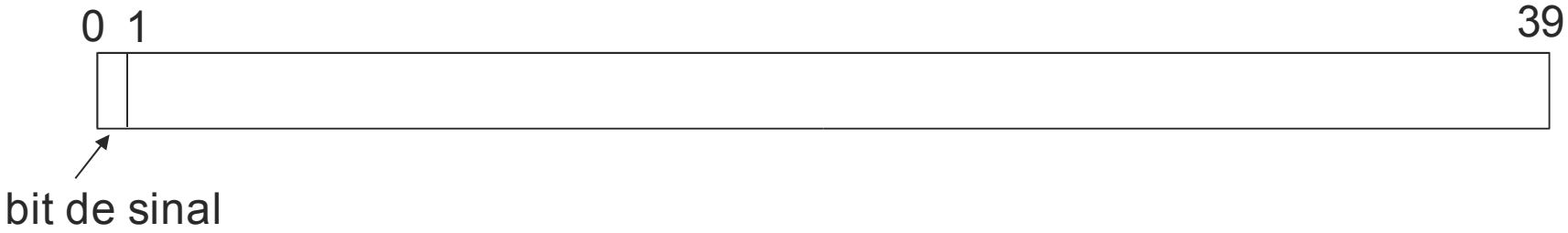


3 Formatos de Instruções: todos com largura de 32 bits

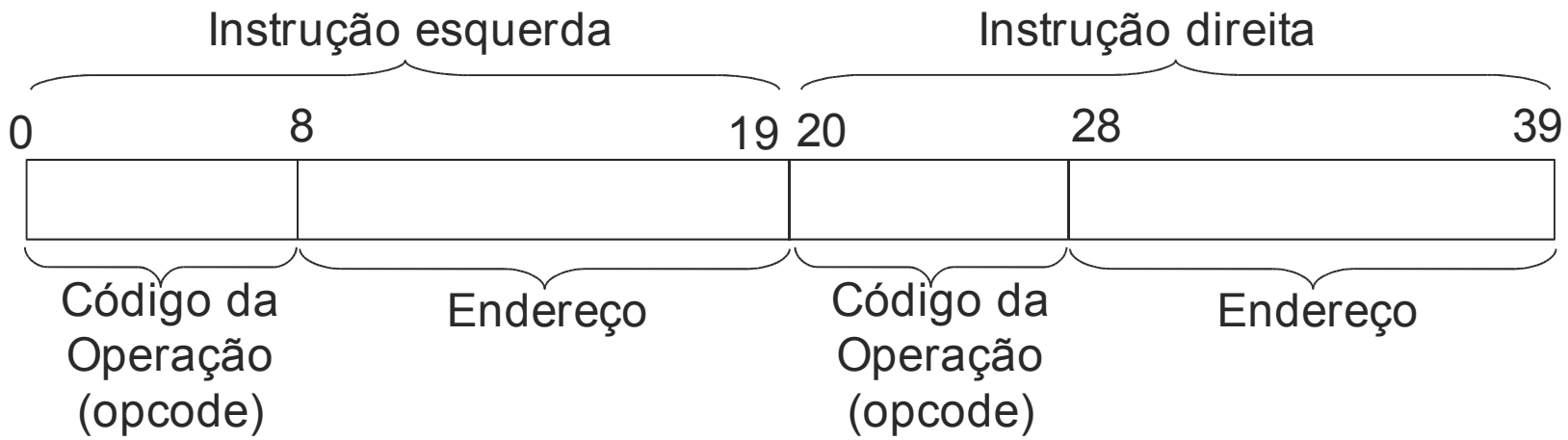


Outro exemplo: IAS (bem antigo... mas dá uma boa idéia)

Formato de dados (números inteiros)



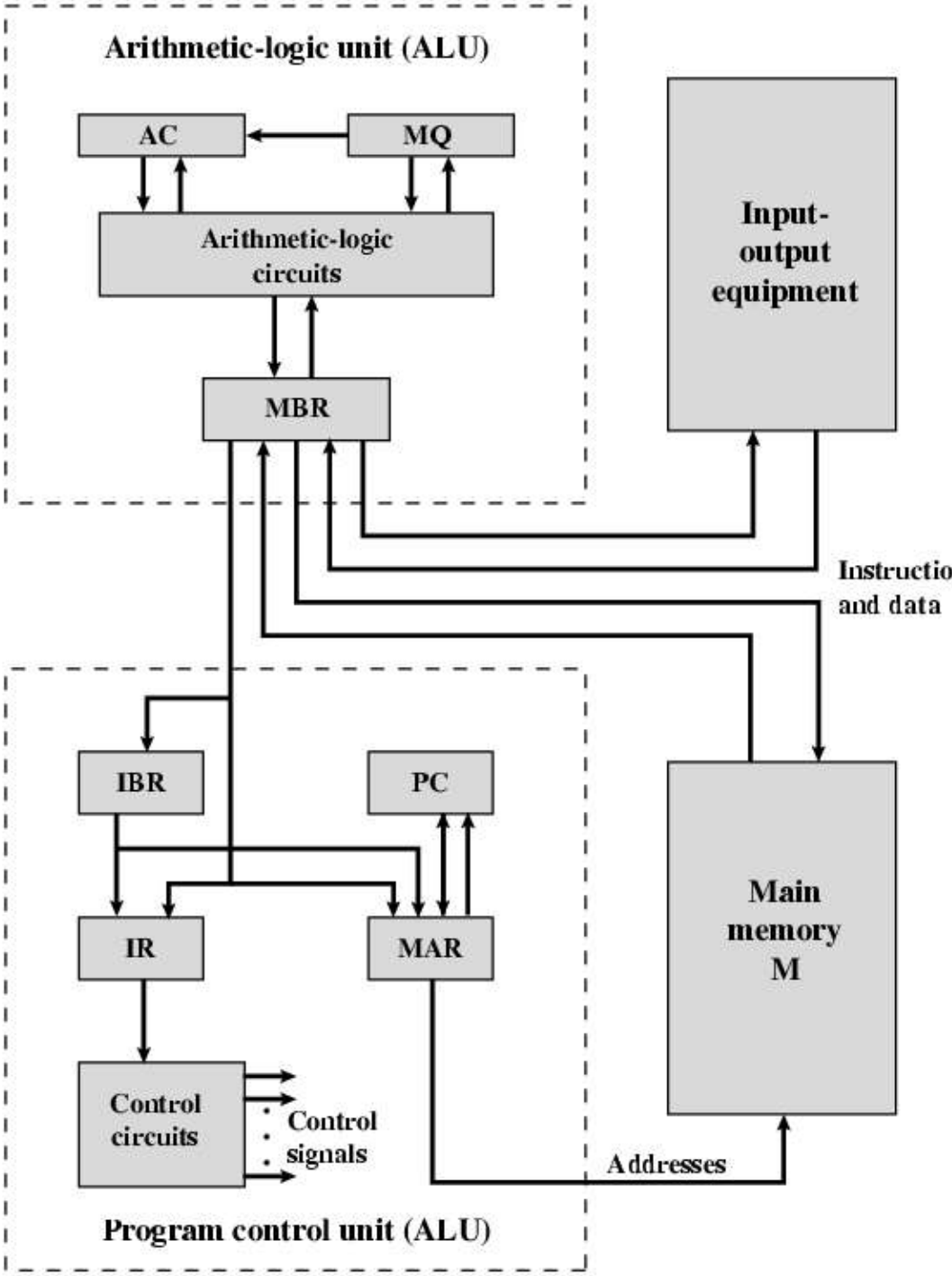
Formato de instruções



Organização do IAS

Observe os registradores AC e MQ

•eles são usados como operandos em toda operação aritimética ou lógica executada



Conjunto de instruções do IAS (muito simples)

Instruções de transferência de dados

opcode	mnemônico	significado
00001010	LOAD MQ	AC ← MQ
00001001	LOAD MQ,M(X)	MQ ← mem(X)
00100001	STOR M(X)	mem(X) ← AC
00000001	LOAD M(X)	AC ← mem(X)
00000010	LOAD – M(X)	AC ← –mem(X)
00000011	LOAD |M(X)|	AC ← abs(mem(X))
00000100	LOAD –|M(X)|	AC ← -abs(mem(X))

Conjunto de instruções do IAS (muito simples)

Instruções de desvio incondicional

opcode	mnemônico	significado
00001101	JUMP M(X,0:19)	próx. instr.: metade esq. de mem(X)
00001110	JUMP M(X,20:39)	próx. instr.: metade dir. de mem(X)

Conjunto de instruções do IAS (muito simples)

Instruções de desvio condicional

opcode	mnemônico	significado
00001111	JUMP +(X,0:19)	se AC >= 0, próx. instr.: metade esq. de mem(X)
00010000	JUMP +M(X,20:39)	se AC >= 0, próx. instr.: metade dir. de mem(X)

Conjunto de instruções do IAS (muito simples)

Instruções aritiméticas

opcode	mnemônico	significado
00000101	ADD M(X)	AC ← AC + mem(X)
00000111	ADD |M(X)|	AC ← AC + |mem(X)|
00000110	SUB M(X)	AC ← AC – mem(X)
00001000	SUB |M(X)|	AC ← AC – |mem(X)|
00001011	MUL M(X)	AC:MQ ← MQ * mem(X)
00001100	DIV M(X)	MQ:AC ← AC / mem(X)
00010100	LSH	AC ← AC * 2 (shift esq.)
00010101	RSH	AC ← AC / 2 (shift dir.)

Conjunto de instruções do IAS (muito simples)

Instruções de alteração de endereço

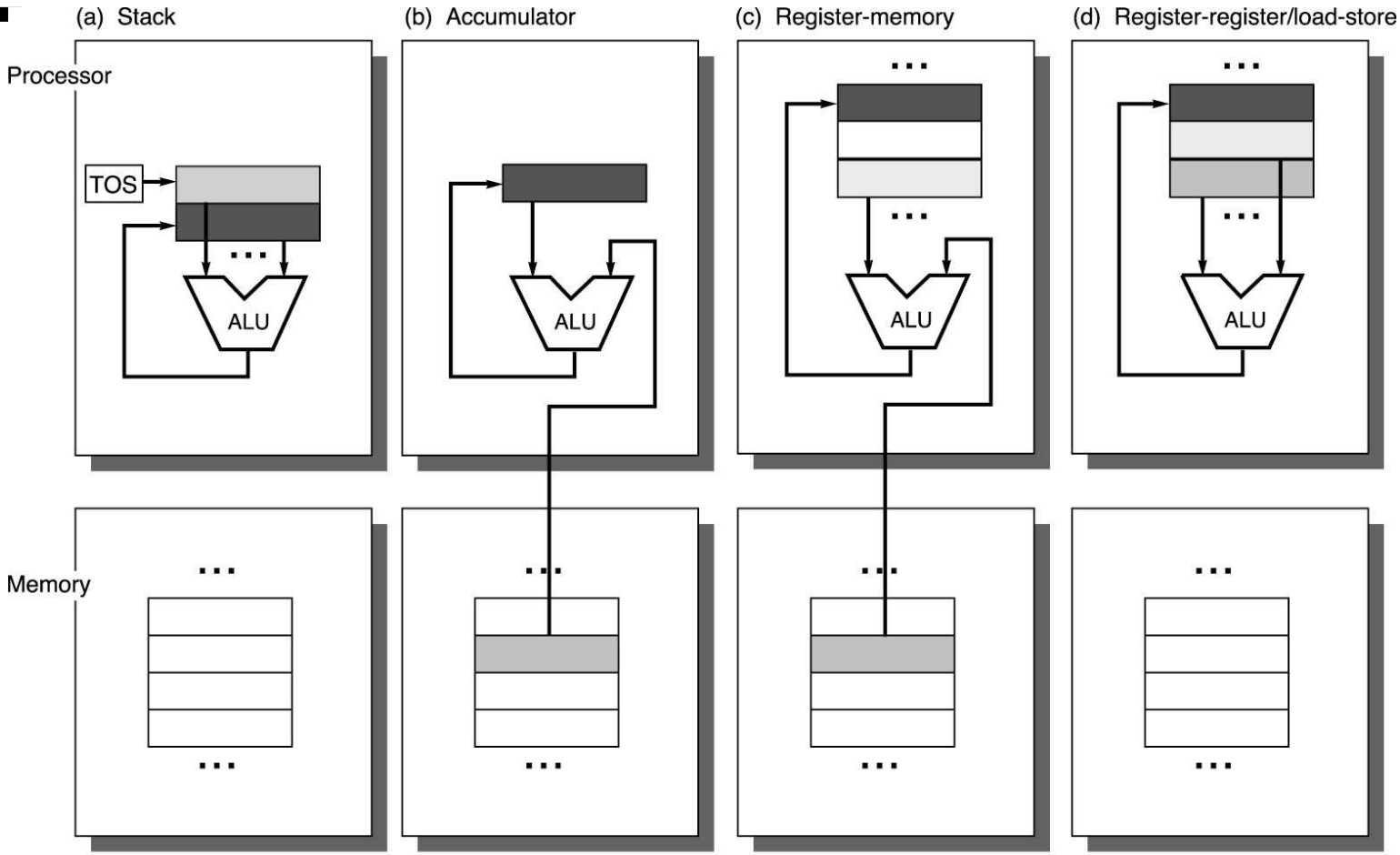
opcode	mnemônico	significado
00010010	STOR M(X,8:19)	substitui o campo de endereço à esq. de mem(X) pelos 12 bits mais à dir. de AC
00010011	STOR M(X,28:39)	subst. o campo de endereço à dir. de mem(X) pelos 12 bits mais à esq. de AC

Classificação de Conjuntos de Instruções

Tipos de arquiteturas comuns

- Arquiteturas de pilha
  - JVM
- Arquiteturas de acumulador
  - IAS, Z80 (algum exemplo recente???)
- Arquiteturas registrador-registrador
  - load-store
  - computadores RISC típicos (MIPS, SPARC,...)
- Arquiteturas registrador-memória
  - 80x86

Os 4 tipos de arquiteturas



C = A + B...

Pilha	Acumulador	Registrador-Memória	load-store
Push A	Load A	Load R1, A	Load R1, A
Push B	Add B	Add R3, R1, B	Load R2, B
Add	Store C	Store R3, C	Add R3, R1, R2
Pop C			Store R3, C

Registradores

Espaço de armazenamento de rápido acesso interno ao processador

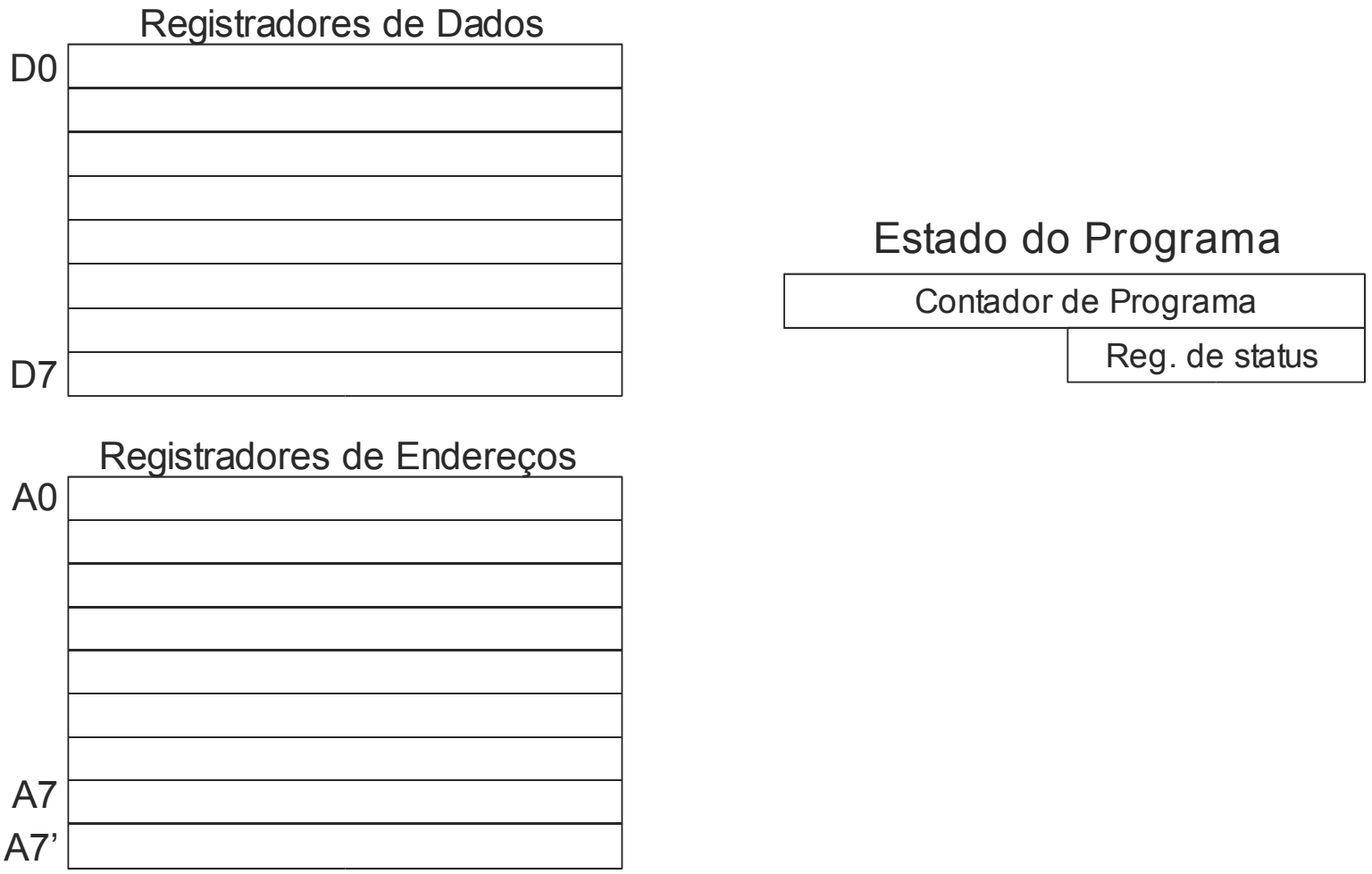
Usos:

- avaliação de expressões
- passagem de parâmetros
- armazenamento de variáveis muito usadas

Arquiteturas de registrador: dois tipos

- registradores de propósito geral
- registradores com finalidades específicas

Exemplos de arquiteturas de registradores: MC68000



Exemplos de arquiteturas de registradores: 8086

Exemplos de arquiteturas de registradores: 80386 – P II

Registradores genéricos

EAX		AX
EBX		BX
ECX		CX
EDX		DX

Registradores de índice

ESP		SP
EBP		BP
ESI		SI
EDI		DI

Estado do Programa

PC	Contador de programa
flags	Flags de status

Operandos de uma instrução: classificação

Número de endereços de memória	Número máximo de operandos	Tipos de arquitetura	Exemplos
0	3	Reg-Reg (load-store)	Alpha, ARM, MIPS, PowerPC, SPARC, SuperH, Trimedia, TM5200
1	2	Reg-Mem	IBM 360/370, **80x86**, Motorola 68K, TI TMS320C54x
2	2	Mem-Mem	VAX
3	3	Mem-Mem	VAX

Comparação

Tipo	Vantagens	Desvantagens
Reg-Reg (0,3)	instruções simples e de tamanho fixo toda instrução consome o mesmo número de ciclos de clock	contagem de instruções mais alta programas maiores
Reg-Mem (1,2)	dados podem ser acessados sem antes fazer uma operação de carga (load) separada resulta em programas menores	operandos não são equivalentes (um deles é fonte-resultado) codificação reduz o número de registradores CPI variável
Mem-Mem (2,2) ou (3,3)	programas mais compactos não gasta registradores com itens temporários	grande variação no tamanho de instruções acessos freqüentes à memoria: gargalo obsoleto

RISC vs. CISC

RISC (*Reduced Instruction Set Computer*)	**CISC (*Complex Instruction Set Computer*)**
instruções simples	instruções mais complexas
poucas instruções	muitas instruções
geralmente load-store	geralmente registrador-memória
instruções de tamanho fixo	instruções de tamanho variável
mais oportunidades de otimização (e.g., pipelines)	mais difícil otimizar
instruções executadas diretamente pelo hardware	instruções interpretadas por meio de microprograma

Tendência atual: arquiteturas híbridas (e.g., P4)

Endereçamento de Memória

Interpretação de endereços de memória

Memória endereçada em bytes

Quantidades endereçadas:

- byte (8 bits)
- meia palavra (16 bits)
- palavra (32 bits)
- palavra dupla (64 bits)

Interpretação de endereços de memória: Ordem dos bytes

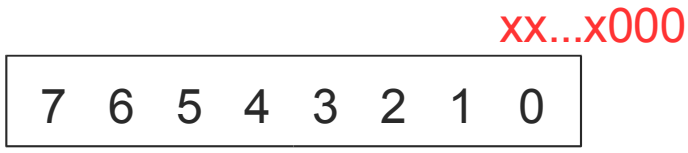
Little-Endian

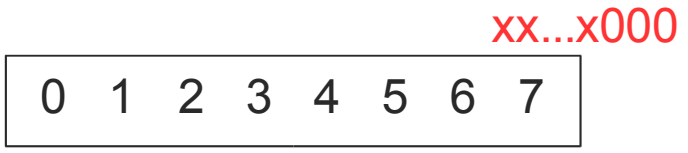
- bytes armazenados da direita para a esquerda
  - i.e., byte **menos** significativo fica no endereço menor
- Ex.:
  - palavra dupla com endereço xx...x000

X

Big-Endian

- bytes armazenados da esquerda para a direita
  - i.e., byte **mais** significativo fica no endereço menor
- Ex.:
  - palavra dupla com endereço xx...x000





Interpretação de endereços de memória: Alinhamento

Em muitos computadores, os acessos a objetos maiores que um byte devem ser através de endereços “alinhados”

Um acesso a um objeto de tamanho *s* bytes no endereço de byte *A* está alinhado se

$A \bmod s = 0$

i.e., se o endereço é divisível pelo tamanho do objeto

Acessos alinhados geralmente são mais eficientes

Modos de endereçamento

Registrador
Imediato
Deslocamento
Indireto de registrador
Indexado
Direto ou absoluto
Indireto de memória
Auto-incremento
Autodecremento
Escalonado

Modo Registrador

Exemplo:

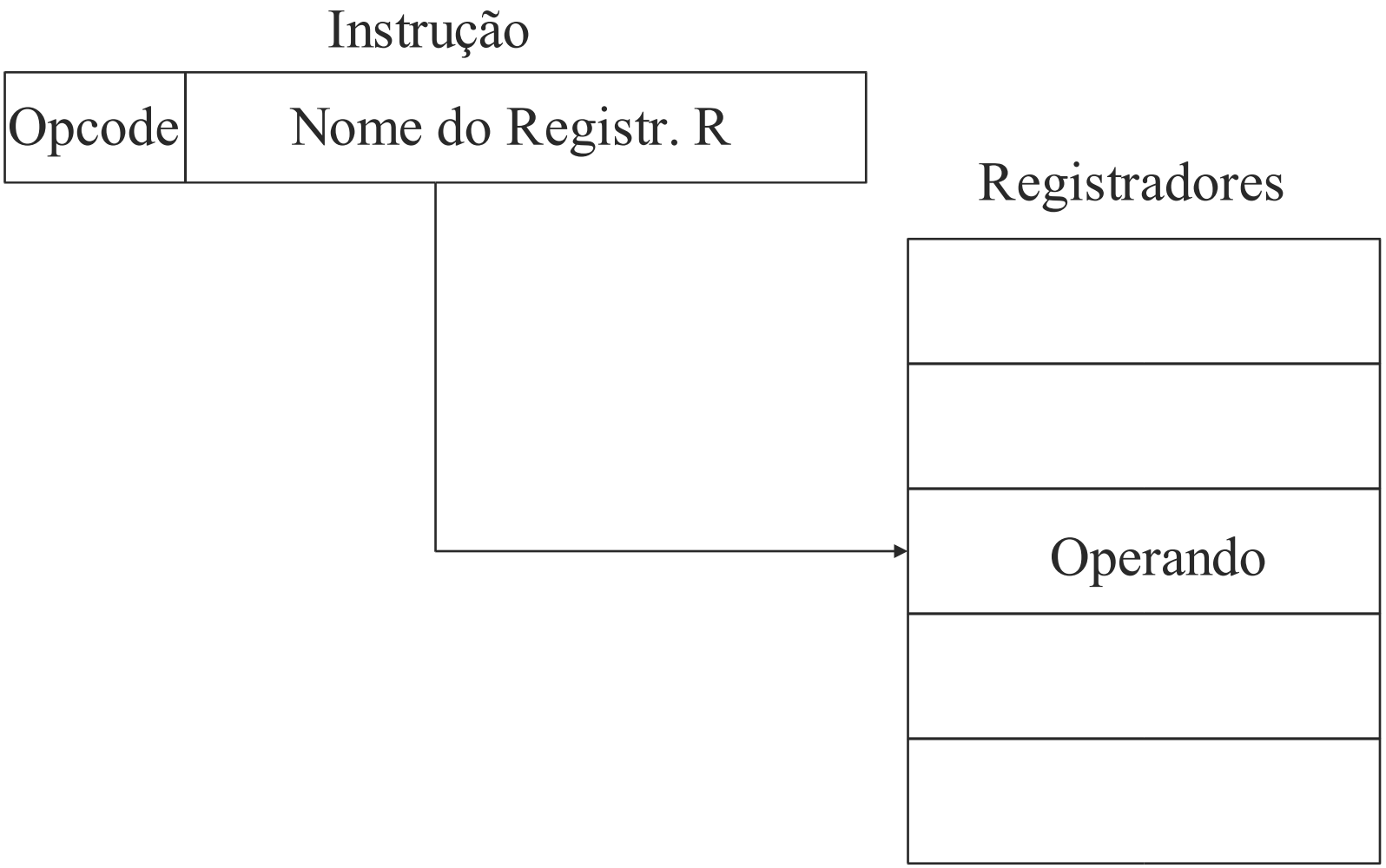
```
Add R4, R3
```

Significado:

```
Regs[R4] ← Regs[R4] + Regs[R3]
```

Usando quando um ou mais operandos está em registradores (inclusive o operando de destino)

Modo Registrador (2)



Modo Imediato

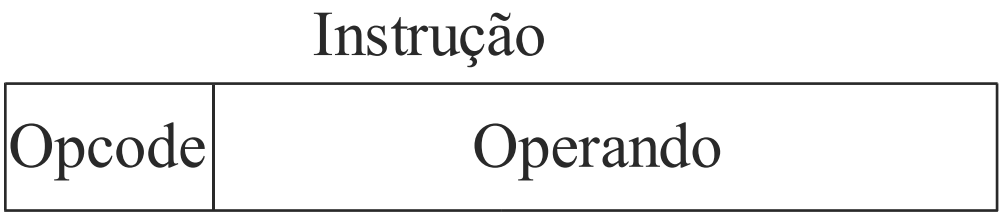
Exemplo:

```
Add R4, #3
```

Significado:

```
Regs[R4] ← Regs[R4] + 3
```

Usando quando um dos operandos é uma constante (codificada no programa)



Modo Direto ou Absoluto

Exemplo:

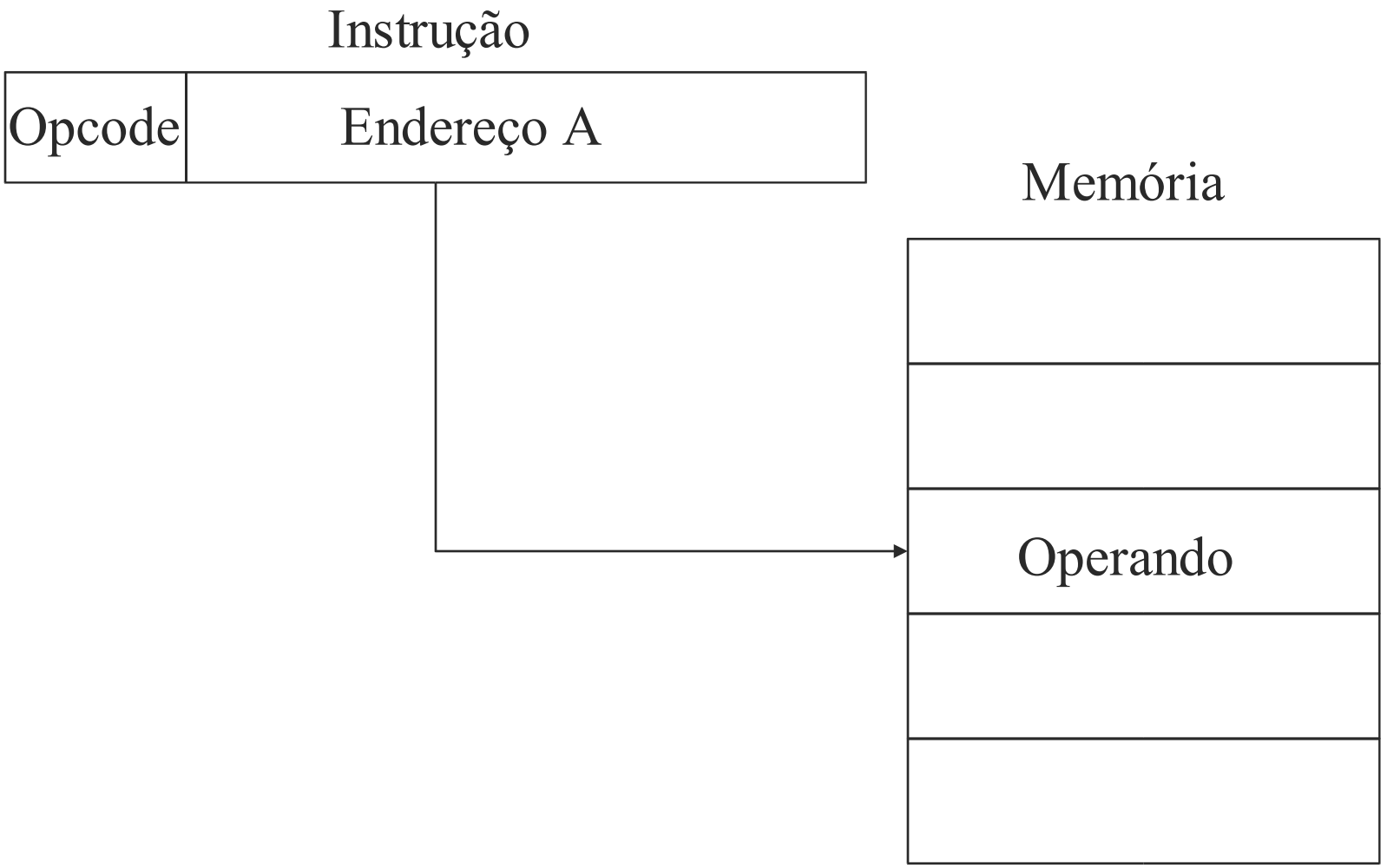
```
Add R1, (1001)
```

Significado:

```
Regs[R1] ← Regs[R1] +
           Mem[1001]
```

Usando no acesso a dados alocados estaticamente na memória (endereço constante)

Modo Direto ou Absoluto (2)



Modo Indireto de Memória

Exemplo:

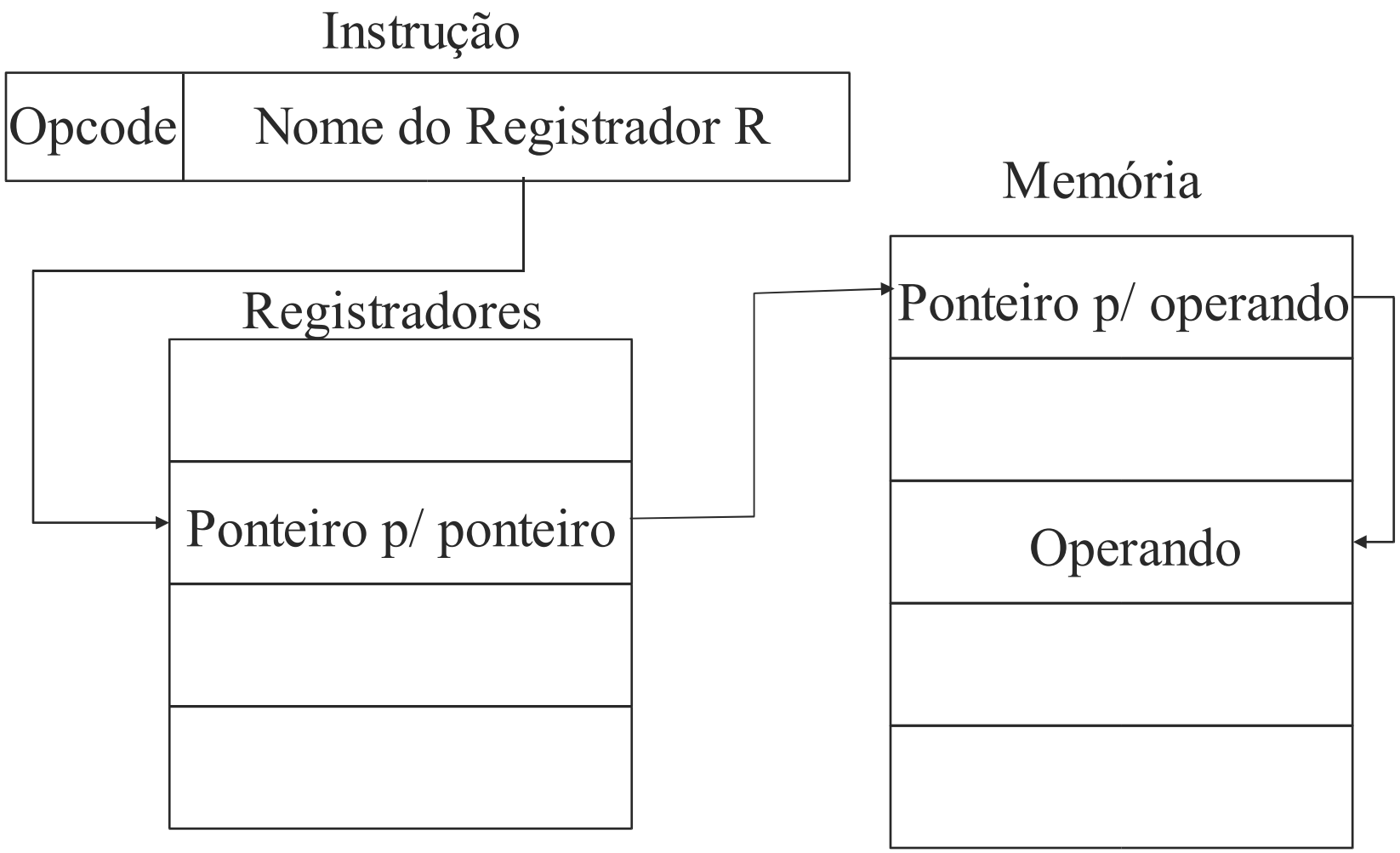
```
Add R1, @(R3)
```

Significado:

```
Regs[R1] ← Regs[R1] +
                Mem[Mem[Regs[R3]]
```

Usando para de-referenciar um ponteiro

Modo Indireto de Memória (2)



Modo Indireto Registrador

Exemplo:

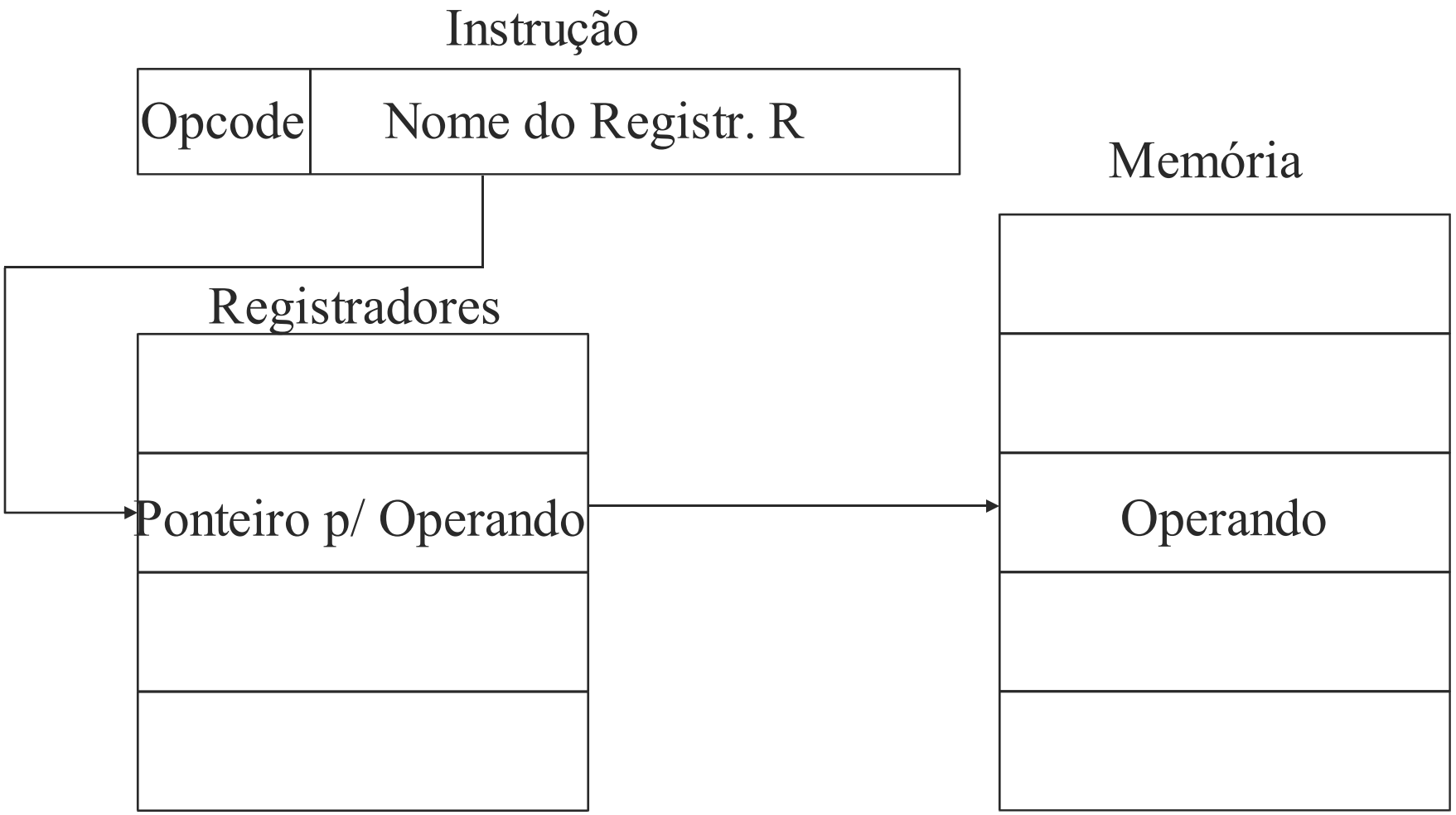
```
Add R4, (R1)
```

Significado:

```
Regs[R4] ← Regs[R4] +
           Mem[Regs[R1]]
```

Usando à memória com uso de ponteiros para cálculo de endereço

Modo Indireto Registrador (2)



Modo Deslocamento

Exemplo:

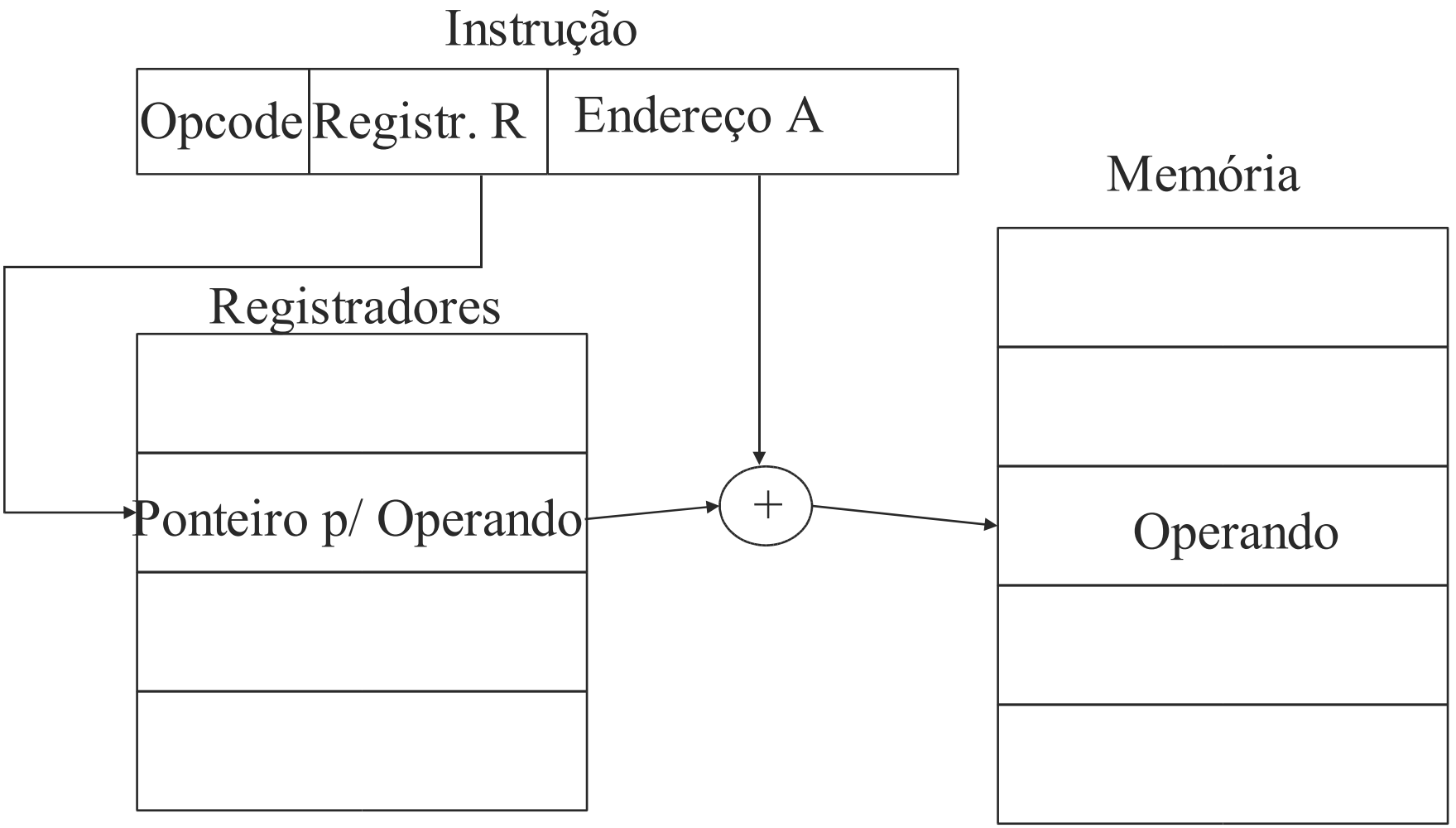
```
Add R4, 100(R1)
```

Significado:

```
Regs[R4] ← Regs[R4] +
                Mem[100+Regs[R1]]
```

Usando para acesso a variáveis locais

Modo Deslocamento (2)



Modo Indexado

Exemplo:

```
Add R3, (R1+R2)
```

Significado:

```
Regs[R3] ← Regs[R3] +
           Mem[Regs[R1]+Regs[R2]]
```

Usando no acesso aos elementos de um vetor (R1=base, R2=índice)

Modo de Auto-Incremento

Exemplo:

```
Add R1, (R2)+
```

Significado:

```
Regs[R1] ← Regs[R1] +
                Mem[Regs[R2]]
Regs[R2] ← Regs[R2] + d
```

Útil para percorrer arrays em loop

- R2 é inicializado com o endereço do primeiro elemento do array
- R2 incrementado de *d* unidades a cada iteração

Modo de Autodecremento

Exemplo:

```
Add R1, -(R2)
```

Significado:

```
Regs[R2] ← Regs[R2] - d
Regs[R1] ← Regs[R1] +
                Mem[Regs[R2]]
```

Idem

R2 é decrementado de *d* unidades a cada iteração

Modo Escalonado

Exemplo:

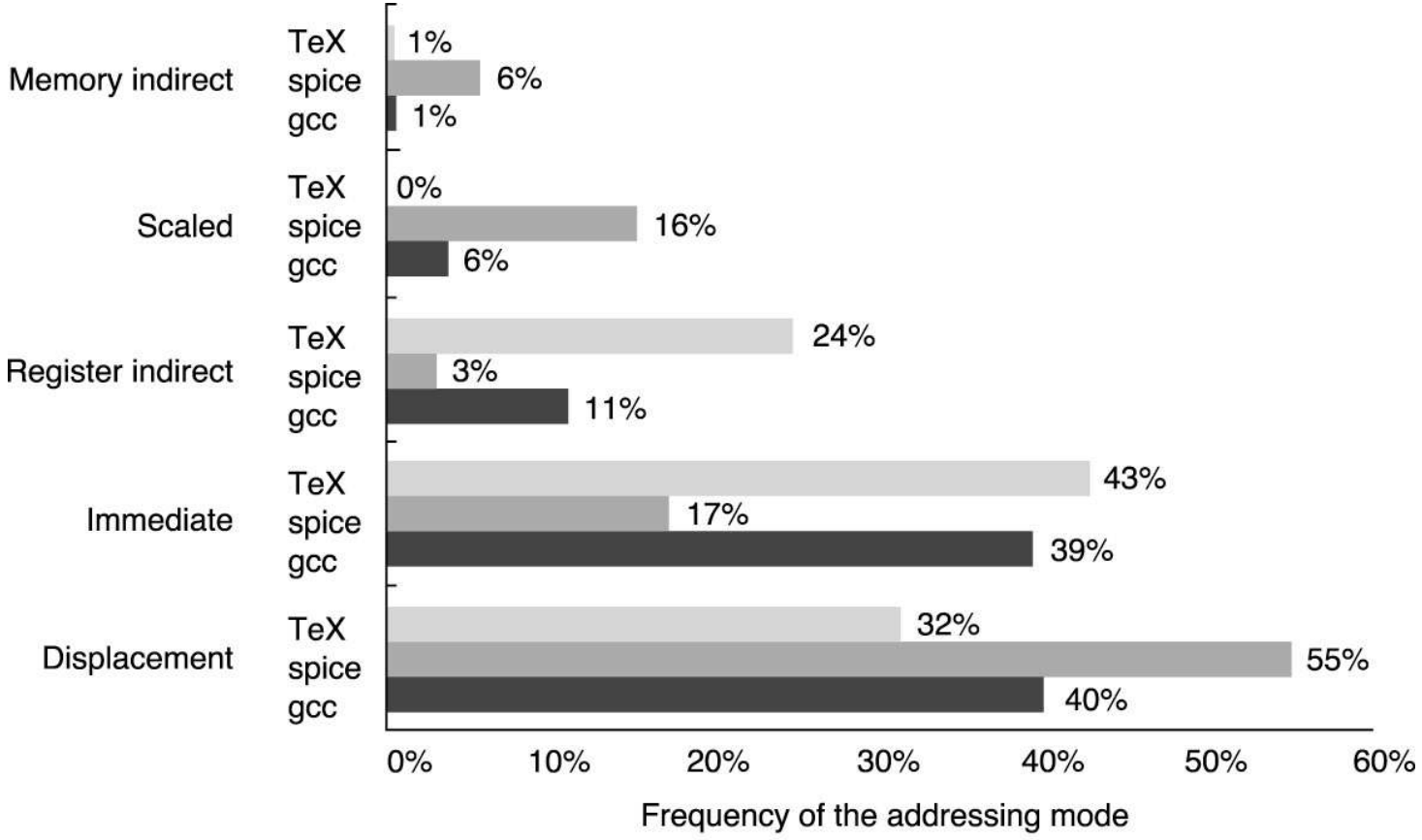
```
Add R1, 100(R2)[R3]
```

Significado:

```
Regs[R1] ← Regs[R1] +
                Mem[100 + Regs[R2] +
                    Regs[R3] * d]
```

Usado para indexar arrays cujos elementos tenham tamanhos não convencionais

Modos de endereçamento: Estatísticas de uso



Modo Registrador (direto): 50% do total

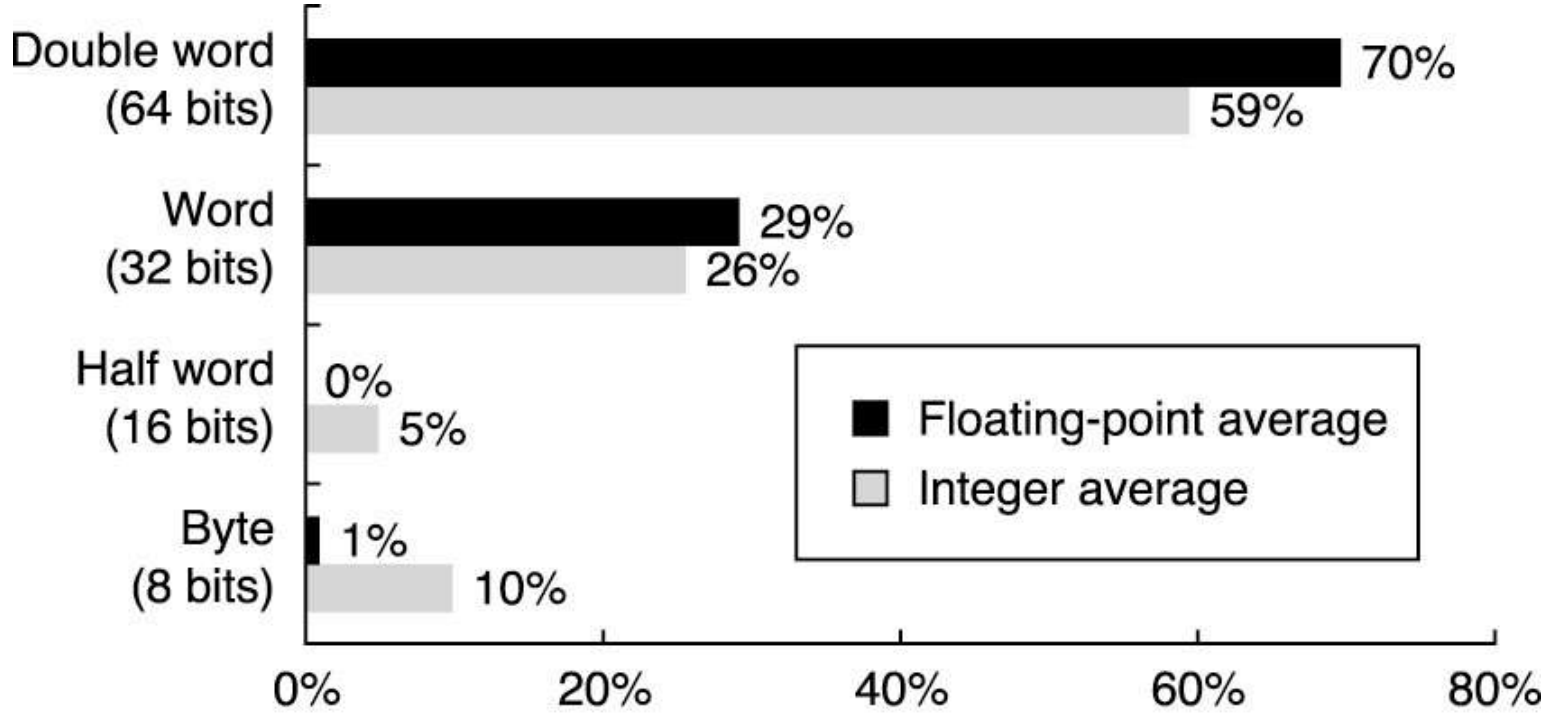
Tipo e Tamanho de Operandos

- Especificação:
  - como parte do opcode
  - marcador (tag) junto ao operando
- Na memória: apenas bits
- Tipo e tamanho dependem da interpretação
- Instruções apropriadas para cada tipo de dados

Tipos e Tamanhos Comuns de Operandos

Tipo	Tamanho
caractere	8 bits
caractere Unicode	16 bits
inteiro	32 bits
inteiro curto	16 bits
ponto flutuante prec. simples	32 bits
ponto flutuante prec. dupla	64 bits
decimal codificado em binário (BCD)	4 bits
strings	nX8bits

Tamanhos de Operandos: Estatísticas de uso



Tipos de Operações

Tipo de operação	Exemplos
Aritiméticas e lógicas	Adição, subtração, e, ou, multiplicação, divisão
Transferência de dados	Operações de carga e armazenamento
Controle	Desvio, salto, chamada e retorno de procedimento, traps
Ponto flutuante	Adição, multiplicação, divisão, comparação
Decimal	Adição, multiplicação, conversões de formato
String	Movimentação, comparação e pesquisa de strings
Gráficos	Transformações com pixels e vértices, (des)compactação

Exemplo: Instruções mais comuns na família Intel x86

Instrução	Freqüência para inteiros
load	22%
desvio condicional	20%
comparação	16%
store	12%
add	8%
and	6%
sub	5%
trasf. registrador-registrador	4%
call (chamada de procedimento)	1%
return (retorno de procedimento)	1%
Total	**96%**

Instruções de Controle do Fluxo de Execução

Desvios condicionais

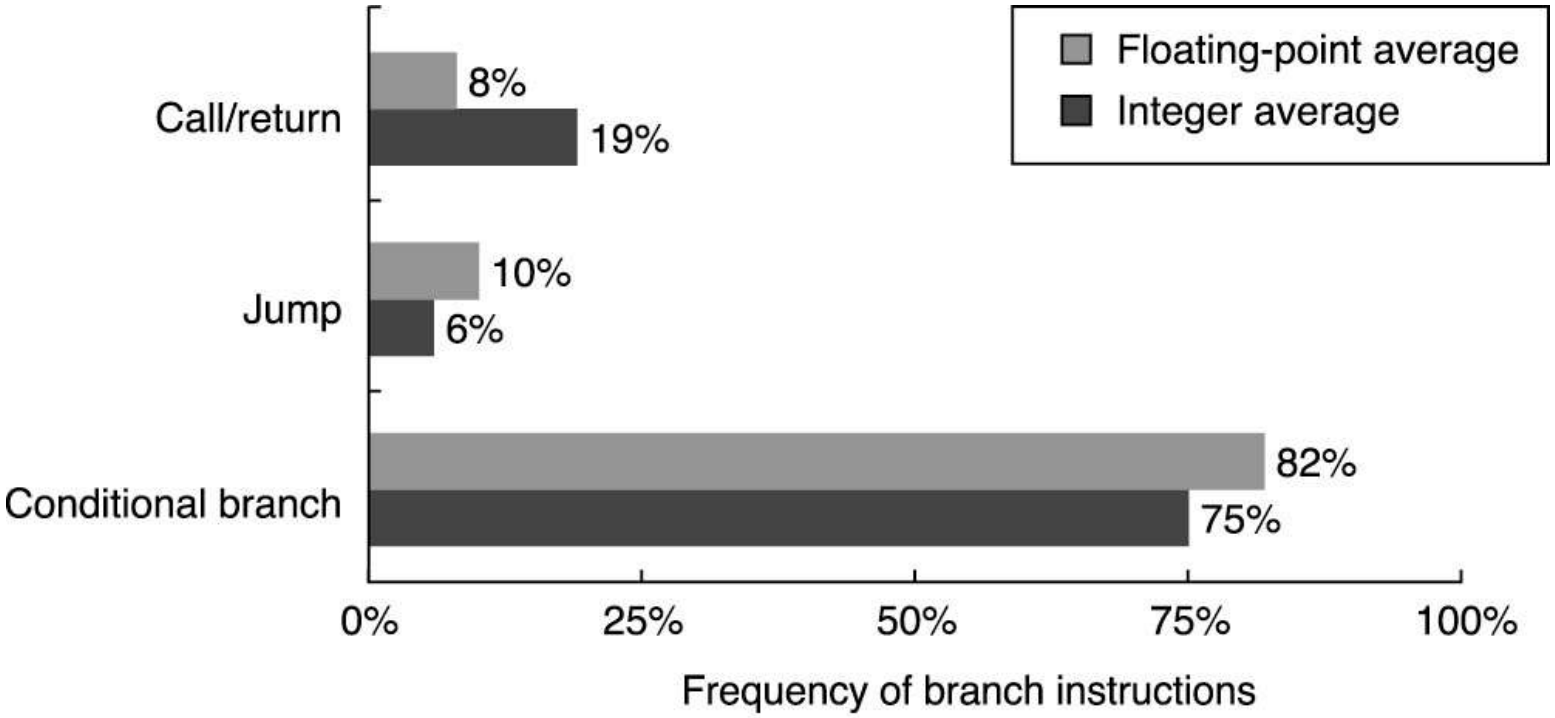
Saltos (incondicionais)

Chamadas de procedimentos

Retorno de procedimentos

Interrupções ou traps

Instruções de Controle de Fluxo: Freqüência



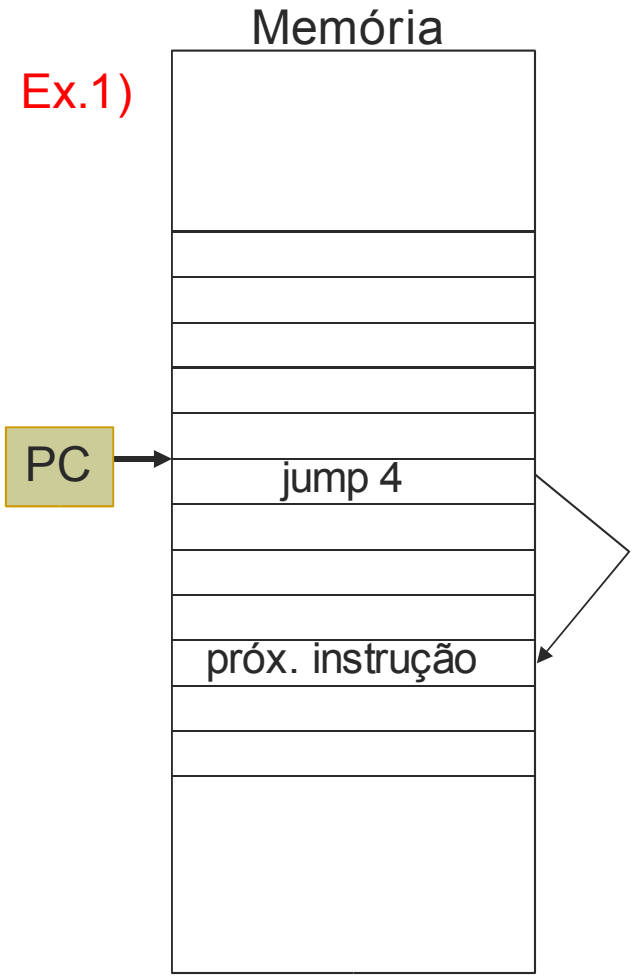
Controle de Fluxo: Especificando o endereço alvo

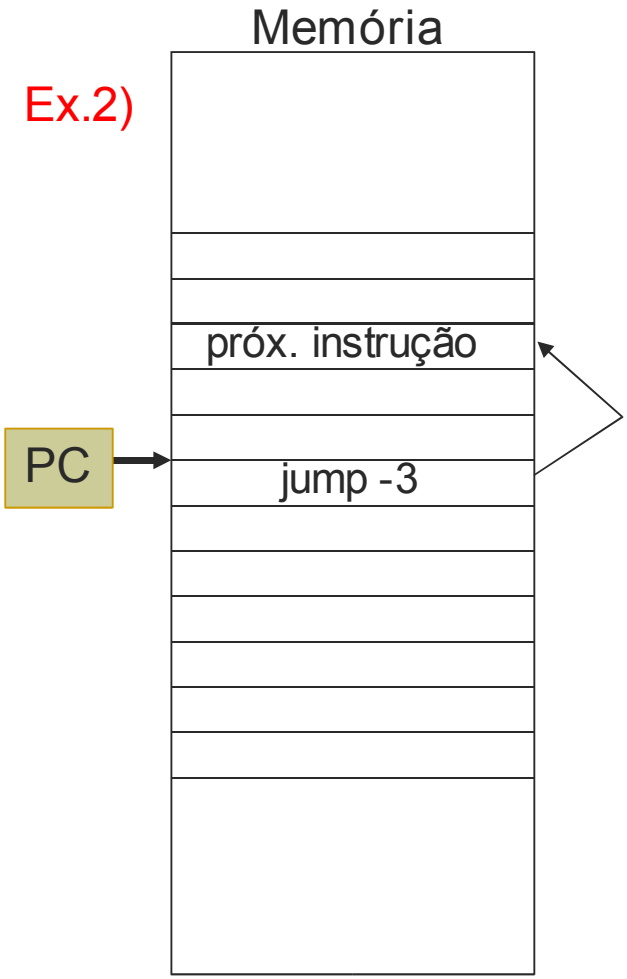
- Na própria instrução (imediato)
  - geralmente com o uso de rótulos
  - estático
- Em um registrador (indireto)
  - dinâmico (endereço não é conhecido *a priori*)
- Em uma posição de memória (indireto)
  - geralmente na pilha
  - dinâmico (endereço não é conhecido *a priori*)

Usos de instruções de controle de fluxo

- Desvios simples, para uma instrução conhecida estaticamente
- Desvios dinâmicos, com endereço calculado pelo programa
  - case/switch, funções e métodos virtuais, ponteiros para funções em C, bibliotecas de carga dinâmica
- Chamada de procedimentos
- Retorno de procedimentos (dinâmico)

Desvios relativos ao PC

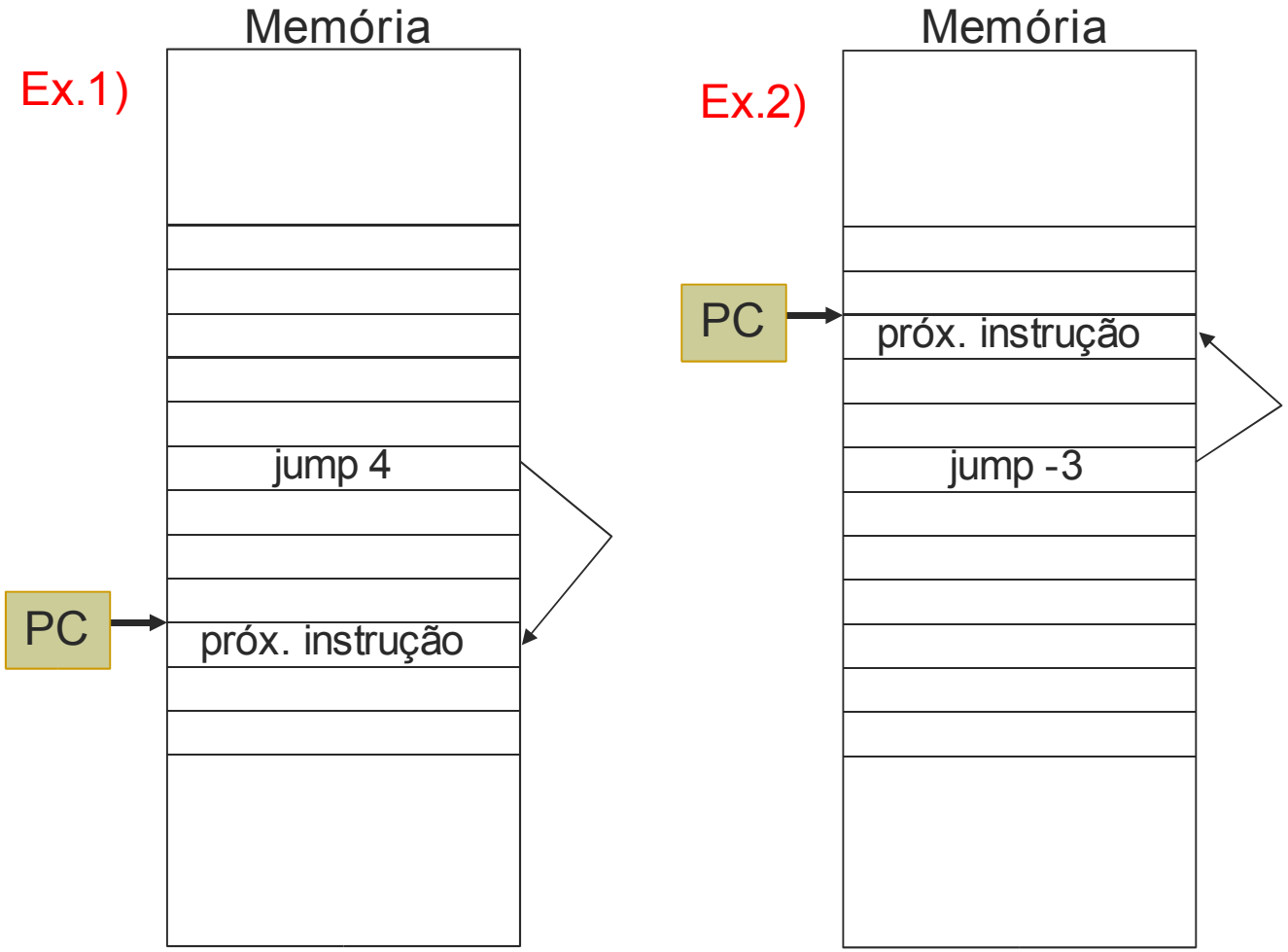




O endereço de desvio especificado na instrução é somado ao (subtraído do) PC

PC: Contador de Programa
→ determina a próxima instrução a ser executada

Desvios relativos ao PC



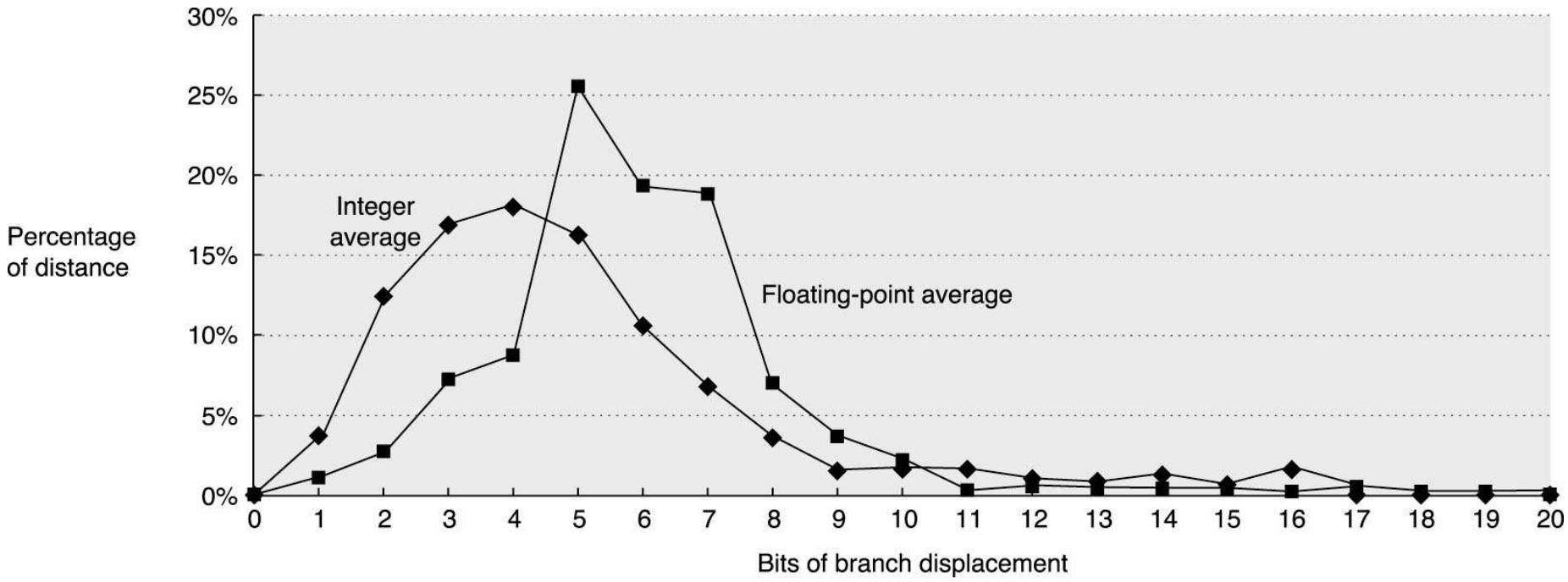
O endereço de desvio especificado na instrução é somado ao (subtraído do) PC

PC: Contador de Programa
→ determina a próxima instrução a ser executada

Desvios relativos ao PC

- Vantagem:
  - Economiza bits no campo de endereço da instrução
- Maioria dos desvios são para instruções próximas à instrução atual
  - Impacto direto no tamanho das instruções de desvio

Distribuição das distâncias de desvios



Eixo x: número de bits necessários para o campo de endereço de destino

Desvios Condicionais

Desvio baseado na satisfação de uma condição lógica

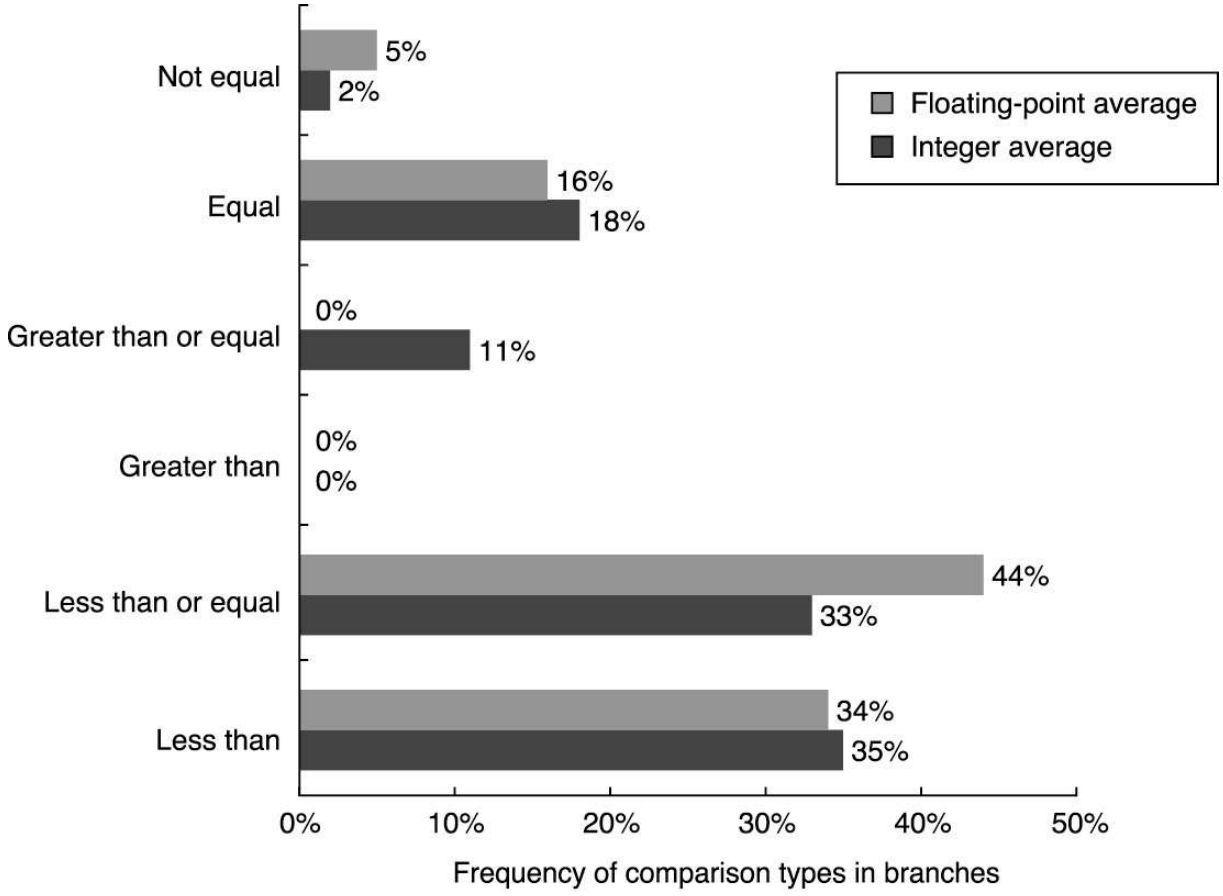
Três opções para especificar a condição:

- códigos condicionais (flags)
- registrador de condição

Com base no resultado da última operação executada pela ALU

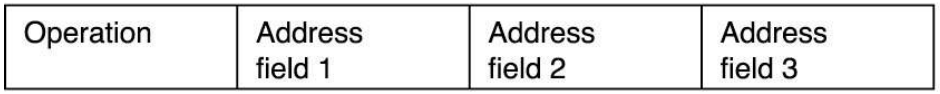
- instruções de comparação e desvio

Desvios condicionais: Freqüência de uso de condições

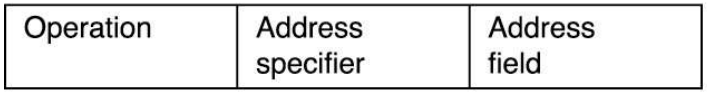


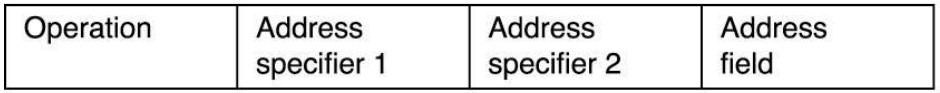
Codificação de Conjuntos de Instruções

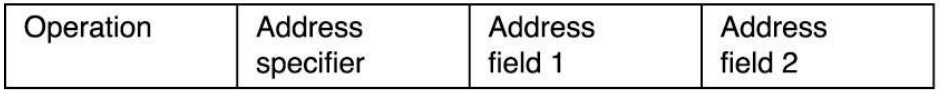
(a) Variable (e.g., VAX, Intel 80x86)



(b) Fixed (e.g., Alpha, ARM, MIPS, PowerPC, SPARC, SuperH)

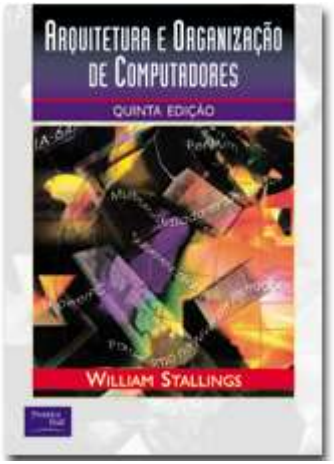






(c) Hybrid (e.g., IBM 360/70, MIPS16, Thumb, TI TMS320C54x)

Referências



Capítulos 9 e 10

Capítulo 2

Capítulo 3