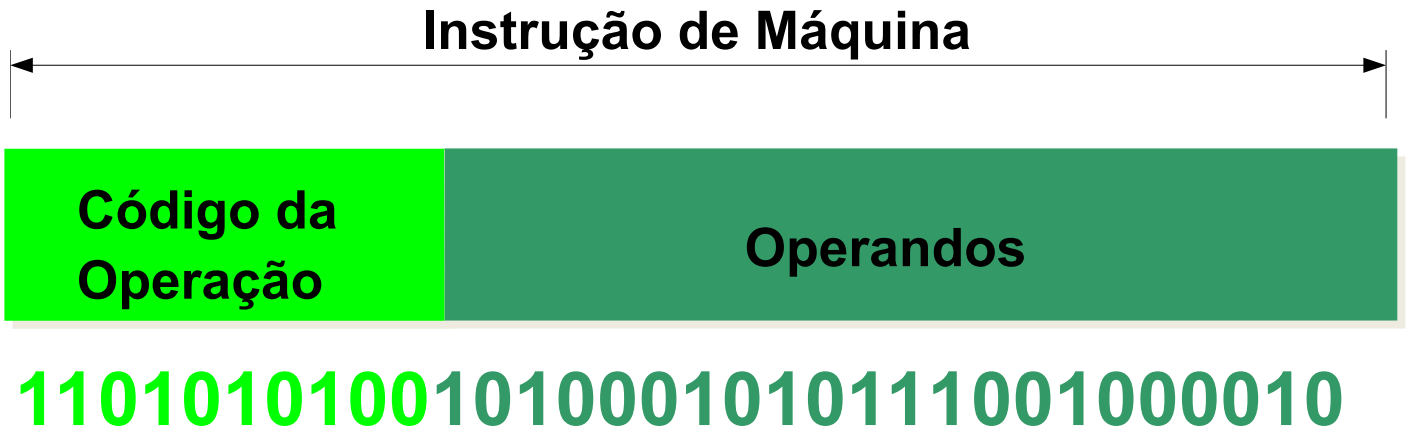
Universidade Federal da Fronteira Sul
Curso de Ciência da Computação
Campus Chapecó

Arquitetura do Conjunto de Instruções e Formato de Representação das Instruções

Prof. Luciano L. Caimi
lcaimi@uffs.edu.br

Arquitetura do Conjunto de Instruções



- A **instrução de máquina** é um conjunto de bits contendo dois campos:
  - Código da Operação (opcode): o que a instrução faz
    Exemplos: add; lw; jump; beq
  - Operandos (operand): onde estão os dados a serem manipulados
    exemplos: constante; registrador, memória

Classes de instruções

Instruções lógicas e aritméticas

- ADD, SUB, AND, XOR, etc...

Instruções de desvio condicional

- BNE, BEQ, BLT, etc...

Instruções de desvio incondicional

- JUMP, CALL, etc...

Instruções de movimentação de dados

- MOV, LOAD, STORE, etc...

Instruções de controle

- NOP, EN_INT,

Arquitetura do Conjunto de Instruções

►A Instrução Assembly

As instruções assembly são definidas a partir da sua sintaxe e semântica

A sintaxe refere-se ao conjunto de regras que regem a formação do "*texto*".

A semântica refere-se ao significado ou a interpretação deste "*texto*".

sintaxe	Semântica
push op;	$pilha_{topo} \leftarrow (op)$

onde: ( ) indica acesso a memória

Arquitetura do Conjunto de Instruções

Arquivo ASM

```
        .data
txt_1   ascii   "Organização de Computadores"
var_A   word    -40
cnt     equ     5       # comentário

        .text
main:
        addi    a4,s0,cnt
        add     a5,a4,a5
        j       L2
L3:
        lw      a5,-20(s0)
        slli    a5,a5,2
        addi    a4,s0,-16
        lw      a5,-20(s0)
        li      a5,99
        ble     a4,a5, L3

L2:
        lw      a4,-20(s0)
        li      a5,99
```

```
txt_1   .data   "Organização de Computadores"
var_A   word    -40
cnt     equ     5       # comentário

        .text
main:
        addi    a4,s0,cnt
        add     a5,a4,a5
        j       L2
L3:
        lw      a5,-20(s0)
        slli    a5,a5,2
        addi    a4,s0,-16
        lw      a5,-20(s0)
        li      a5,99
        ble     a4,a5, L3

L2:
        lw      a4,-20(s0)
        li      a5,99
```

Arquitetura do Conjunto de Instruções

```c
int main() {
    int num1[100];
    int num2[100];
    int result[100];
    for(int x = 0; x < 100; x++){
        result[x] = num1[x] + num2[x];
    }
}
```

```
main:
        addi    sp,sp,-1232
        sw      s0,1228(sp)
        addi    s0,sp,1232
        sw      zero,-20(s0)
        j       .L2
.L3:
        lw      a5,-20(s0)
        slli    a5,a5,2
        addi    a4,s0,-16
        add     a5,a4,a5
        lw      a4,-404(a5)
        lw      a5,-20(s0)
        slli    a5,a5,2
        addi    a3,s0,-16
        add     a5,a3,a5
        lw      a5,-804(a5)
        add     a4,a4,a5
        lw      a5,-20(s0)
        slli    a5,a5,2
        addi    a3,s0,-16
        add     a5,a3,a5
        sw      a4,-1204(a5)
        lw      a5,-20(s0)
        addi    a5,a5,1
        sw      a5,-20(s0)
.L2:
        lw      a4,-20(s0)
        li      a5,99
        ble     a4,a5,.L3
        li      a5,0
        mv      a0,a5
        lw      s0,1228(sp)
        addi    sp,sp,1232
        jr      ra
```

```
00000000 <main>:
    0:      b3010113        addi x2 x2 -1232
    4:      4c812623        sw x8 1228 x2
    8:      4d010413        addi x8 x2 1232
    c:      fe042623        sw x0 -20 x8
   10:      0500006f        jal x0 80 <L2>

00000014 <L3>:
   14:      fec42783        lw x15 -20 x8
   18:      00279793        slli x15 x15 2
   1c:      ff040713        addi x14 x8 -16
   20:      00f707b3        add x15 x14 x15
   24:      e6c7a703        lw x14 -404 x15
   28:      fec42783        lw x15 -20 x8
   2c:      00279793        slli x15 x15 2
   30:      ff040693        addi x13 x8 -16
   34:      00f687b3        add x15 x13 x15
   38:      cdc7a783        lw x15 -804 x15
   3c:      00f70733        add x14 x14 x15
   40:      fec42783        lw x15 -20 x8
   44:      00279793        slli x15 x15 2
   48:      ff040693        addi x13 x8 -16
   4c:      00f687b3        add x15 x13 x15
   50:      b4e7a623        sw x14 -1204 x15
   54:      fec42783        lw x15 -20 x8
   58:      00178793        addi x15 x15 1
   5c:      fef42623        sw x15 -20 x8

00000060 <L2>:
   60:      fec42703        lw x14 -20 x8
   64:      06300793        addi x15 x0 99
   68:      fae7d6e3        bge x15 x14 -84 <L3>
   6c:      00000793        addi x15 x0 0
   70:      00078513        addi x10 x15 0
   74:      4cc12403        lw x8 1228 x2
   78:      4d010113        addi x2 x2 1232
   7c:      00008067        jalr x0 x1 0
```

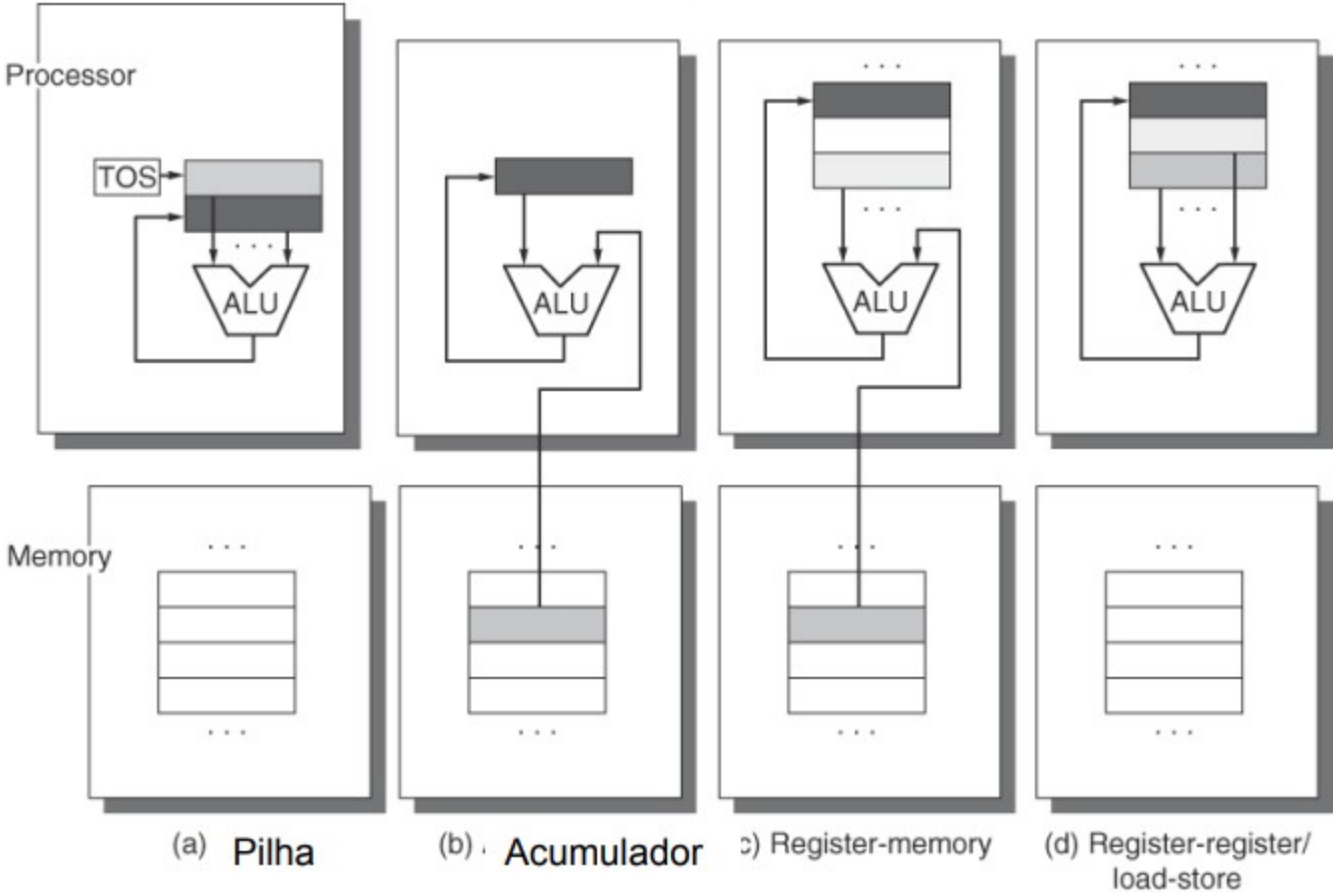
```
10110011000000010000000100010011
01001100100000010010011000100011
01001101000000010000010000010011
11111110000001000010011000100011
00000101000000000000000001101111

11111110110001000010011110000011
00000000001001111001011110010011
11111111000001000000011100010011
00000000111101110000011110110011
11100110110001111010011100000011
11111110110001000010011110000011
00000000001001111001011110010011
11111111000001000000011010010011
00000000111101101000011110110011
11001101110001111010011110000011
00000000111101110000011100110011
11111110110001000010011110000011
00000000001001111001011110010011
11111111000001000000011010010011
00000000111101101000011110110011
10110100111001111010011000100011
11111110110001000010011110000011
00000000000010111100011110010011
11111110111101000010011000100011

11111110110001000010011100000011
00000110001100000000011110010011
11111010111001111101011011100011
00000000000000000000011110010011
00000000000001111000010100010011
01001100110000010010010000000011
01001101000000010000000100010011
00000000000000001000000001100111
```

Arquitetura do Conjunto de Instruções

- **Classificação quanto ao tipo de armazenamento interno:**
  - Arquitetura de Pilha
  - Arquitetura de Acumulador
  - Arquitetura de Registradores de Propósito Geral
    - *arquitetura register-memory*
    - *arquitetura load-store* (*register-register*)
  - Arquitetura Memória-Memória (obsoleto)

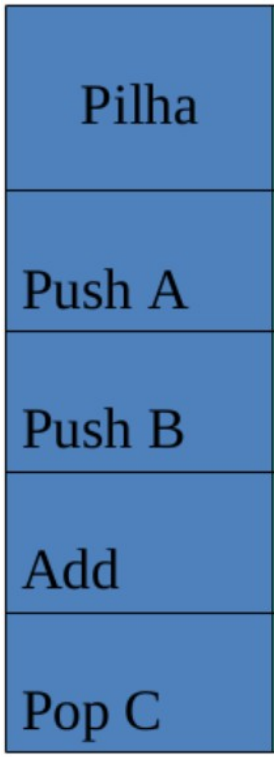
Arquitetura do Conjunto de Instruções

- **Classificação quanto ao tipo de armazenamento interno:**



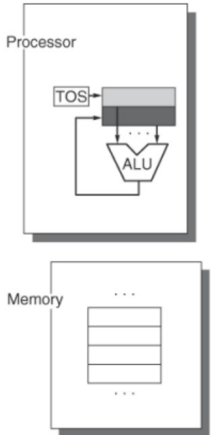
Arquitetura do Conjunto de Instruções

Exemplo: C = A+B



Pilha

```
Pop  op; (op) ← topo
Push op; topo ← (op)
Add    ; topo ← topo + topo-1
Sub    ; topo ← topo - topo-1
Mul    ; topo ← topo * topo-1
Div    ; topo ← topo / topo-1
```

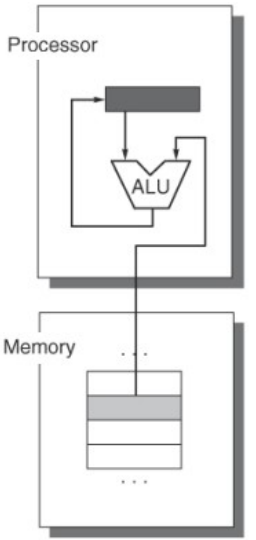


Arquitetura do Conjunto de Instruções

Exemplo: C = A+B

Acumulador
Load A
Add B
Store C

```
Movmw op; w ← (op)
Movwm op; (op) ← w
Add   op; w ← w + (op)
Sub   op; w ← w - (op)
Mul   op; w ← w * (op)
Div   op; w ← w / (op)
```

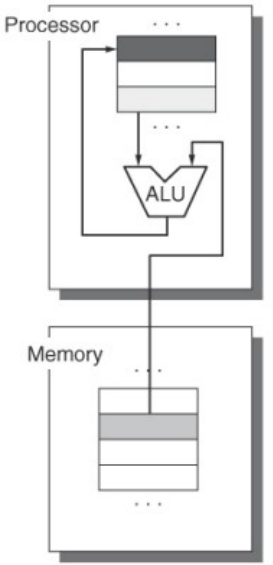


Arquitetura do Conjunto de Instruções

Exemplo: C = A+B

Registrador (register-memory)
Load R1, A
Add R1, B
Store C, R1

```
Load  R, op   ; R ← (op)
Store op, R   ; (op) ← R
Add R,op      ; R ← R + (op)
Sub R,op      ; R ← R - (op)
Mul R,op      ; R ← R * (op)
Div R,op      ; R ← R / (op)
```

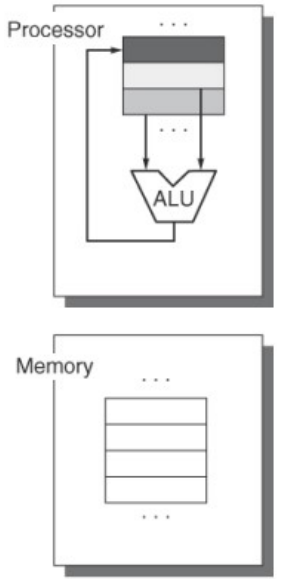


Arquitetura do Conjunto de Instruções

Exemplo: C = A+B

Registrador (load-store)
Load R1, A
Load R2, B
Add, R3, R1, R2
Store C, R3

```
Load  R, op     ; R ← (op)
Store op, R     ; (op) ← R
Add rd,rs1,rs2  ; rd ← rs1 + rs2
Sub rd,rs1,rs2  ; rd ← rs1 - rs2
Mul rd,rs1,rs2  ; rd ← rs1 * rs2
Div rd,rs1,rs2  ; rd ← rs1 / rs2
```



Arquitetura do Conjunto de Instruções

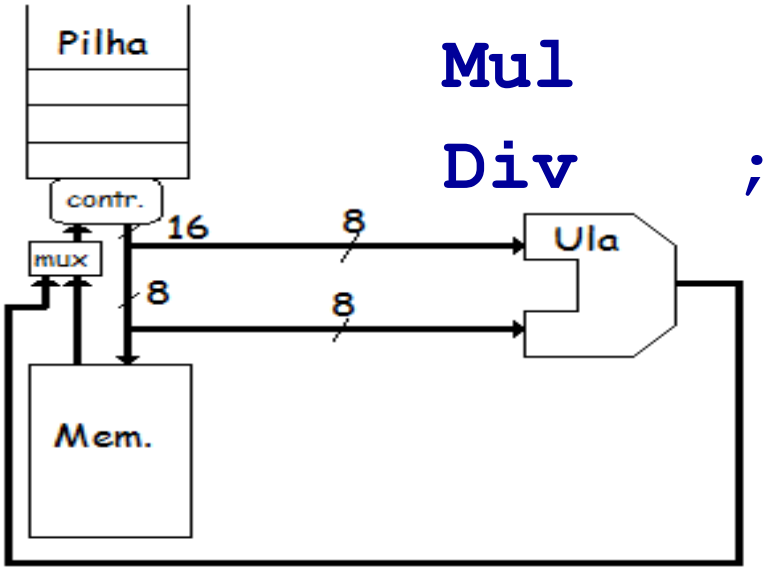
Pilha

```
Pop  op; (op) ← topo
Push op; topo ← (op)
Add    ; topo ← topo + topo-1
Sub    ; topo ← topo - topo-1
Mul    ; topo ← topo * topo-1
Div    ; topo ← topo / topo-1
```

Arquitetura do Conjunto de Instruções

Pilha

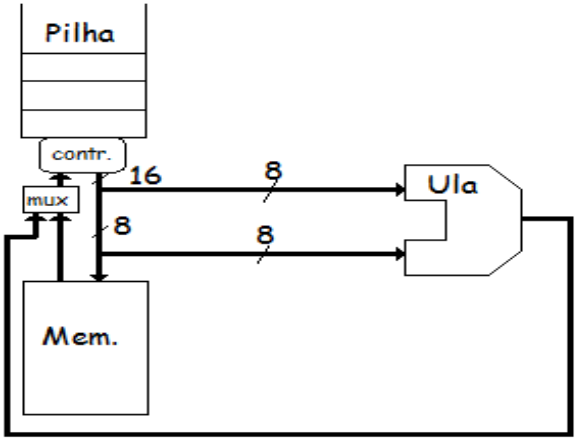
```
Pop  op; (op) ← topo
Push op; topo ← (op)
Add    ; topo ← topo + topo-1
Sub    ; topo ← topo - topo-1
Mul    ; topo ← topo * topo-1
Div    ; topo ← topo / topo-1
```



Arquitetura do Conjunto de Instruções

Pilha:

$$S = \frac{(A - C) * D}{(C - B)}$$



Pilha

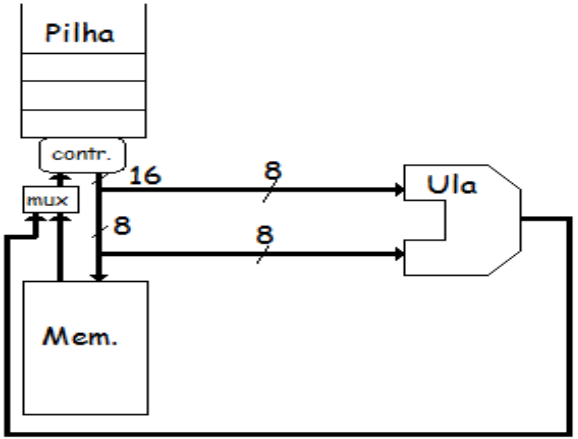
```
Pop  op; (op) ← topo
Push op; topo ← (op)
Add    ; topo ← topo + topo-1
Sub    ; topo ← topo - topo-1
Mul    ; topo ← topo * topo-1
Div    ; topo ← topo / topo-1
```

Arquitetura do Conjunto de Instruções

Arquitetura do Conjunto de Instruções

Pilha:

$$S = \frac{(A - C) * D}{(C - 5)}$$



Pilha

```
Pop  op; (op) ← topo
Push op; topo ← (op)
Add    ; topo ← topo + topo-1
Sub    ; topo ← topo - topo-1
Mul    ; topo ← topo * topo-1
Div    ; topo ← topo / topo-1
```

Arquitetura do Conjunto de Instruções

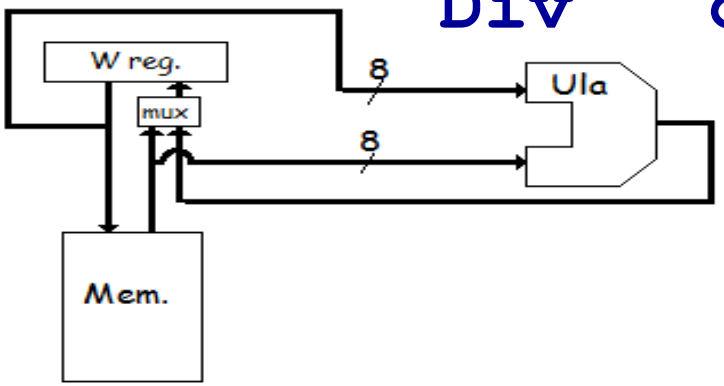
Acumulador (work register)

```
Movmw op; w ← (op)
Movwm op; (op) ← w
Add   op; w ← w + (op)
Sub   op; w ← w - (op)
Mul   op; w ← w * (op)
Div   op; w ← w / (op)
```

Arquitetura do Conjunto de Instruções

Acumulador (work register)

```
Movmw op; w ← (op)
Movwm op; (op) ← w
Add   op; w ← w + (op)
Sub   op; w ← w - (op)
Mul   op; w ← w * (op)
Div   op; w ← w / (op)
```

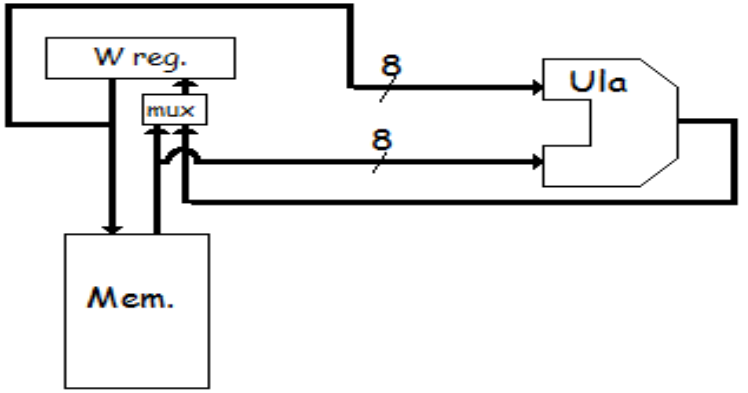


Arquitetura do Conjunto de Instruções

Acumulador (work register)

$$S = \frac{(A - C) * D}{(C - B)}$$

```
Movmw op; w ← (op)
Movwm op; (op) ← w
Add   op; w ← w + (op)
Sub   op; w ← w - (op)
Mul   op; w ← w * (op)
Div   op; w ← w / (op)
```



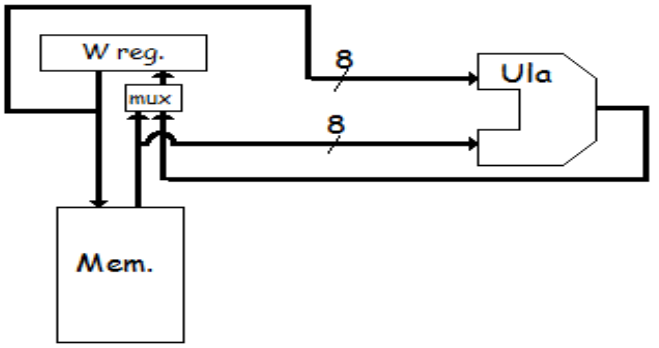
Arquitetura do Conjunto de Instruções

Arquitetura do Conjunto de Instruções

Acumulador (work register)

$$S = \frac{(A - C) * D}{(C - 5)}$$

```
Movmw op; w ← (op)
Movwm op; (op) ← w
Add   op; w ← w + (op)
Sub   op; w ← w - (op)
Mul   op; w ← w * (op)
Div   op; w ← w / (op)
```



Arquitetura do Conjunto de Instruções

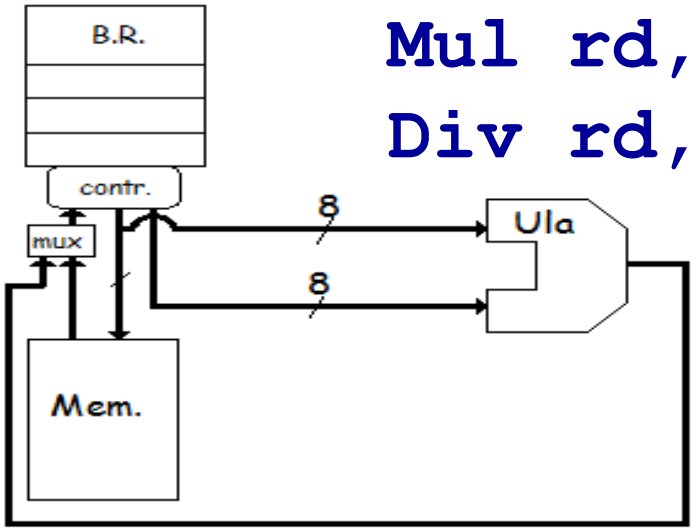
Load-Store

```
Load  op, R      ; R ← (op)
Store op, R      ; (op) ← R
Add rd,rs1,rs2   ; rd ← rs1 + rs2
Sub rd,rs1,rs2   ; rd ← rs1 - rs2
Mul rd,rs1,rs2   ; rd ← rs1 * rs2
Div rd,rs1,rs2   ; rd ← rs1 / rs2
```

Arquitetura do Conjunto de Instruções

Load-Store

```
Load  op, R     ; R ← (op)
Store op, R     ; (op) ← R
Add rd,rs1,rs2  ; rd ← rs1 + rs2
Sub rd,rs1,rs2  ; rd ← rs1 - rs2
Mul rd,rs1,rs2  ; rd ← rs1 * rs2
Div rd,rs1,rs2  ; rd ← rs1 / rs2
```



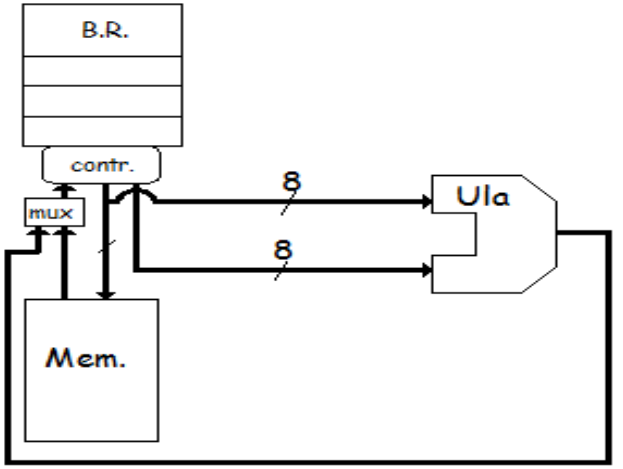
Arquitetura do Conjunto de Instruções

Arquitetura do Conjunto de Instruções

Load-Store

$$S = \frac{(A - C) * D}{(C - B)}$$

```
Load  op, R      ; R ← (op)
Store op, R      ; (op) ← R
Add rd,rs1,rs2   ; rd ← rs1 + rs2
Sub rd,rs1,rs2   ; rd ← rs1 - rs2
Mul rd,rs1,rs2   ; rd ← rs1 * rs2
Div rd,rs1,rs2   ; rd ← rs1 / rs2
```



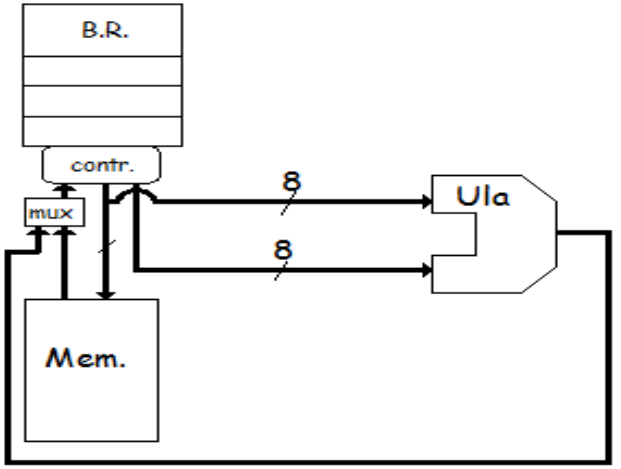
Arquitetura do Conjunto de Instruções

Arquitetura do Conjunto de Instruções

Load-Store

$$S = \frac{(A - C) * D}{(C - 5)}$$

```
Load  op, R      ; R ← (op)
Store op, R      ; (op) ← R
Add rd,rs1,rs2   ; rd ← rs1 + rs2
Sub rd,rs1,rs2   ; rd ← rs1 - rs2
Mul rd,rs1,rs2   ; rd ← rs1 * rs2
Div rd,rs1,rs2   ; rd ← rs1 / rs2
```



Arquitetura do Conjunto de Instruções

Exercício:

1) Considerando a equação:

$$S = \left(A^2 - \left(\frac{C + B}{A}\right) \right) * \left(\frac{D - B}{2^C}\right)$$

a) Defina um conjunto de instruções (ISA) com armazenamento interno tipo pilha para implementar a equação

b) Apresente o programa assembly utilizando o ISA definido acima para a equação

Arquitetura do Conjunto de Instruções

Exercício:

1) Considerando a equação:

$$S = \left(A^2 - \left(\frac{C + B}{A}\right) \right) * \left(\frac{D - B}{2^C}\right)$$

a) Defina um conjunto de instruções (ISA) com armazenamento interno tipo acumulador para implementar a equação

b) Apresente o programa assembly utilizando o ISA definido acima para a equação

Arquitetura do Conjunto de Instruções

Arquitetura do Conjunto de Instruções

▶ Tipos de Ordenação de dados

Refere-se a ordem que os dados são armazenados na memória

Imagine o valor hexadecimal 0x12345678 a ser armazenado a partir do endereço 0x120

Endereço	Big Endian	Little Endian
120	12	78
121	34	56
122	56	34
123	78	12

Little Endian: Intel; Risc-V

Big Endian: Motorola 68XX

Bi Endian: PowerPC; ARM (geralmente Little)

Arquitetura do Conjunto de Instruções

▶ Tipos de Ordenação de dados

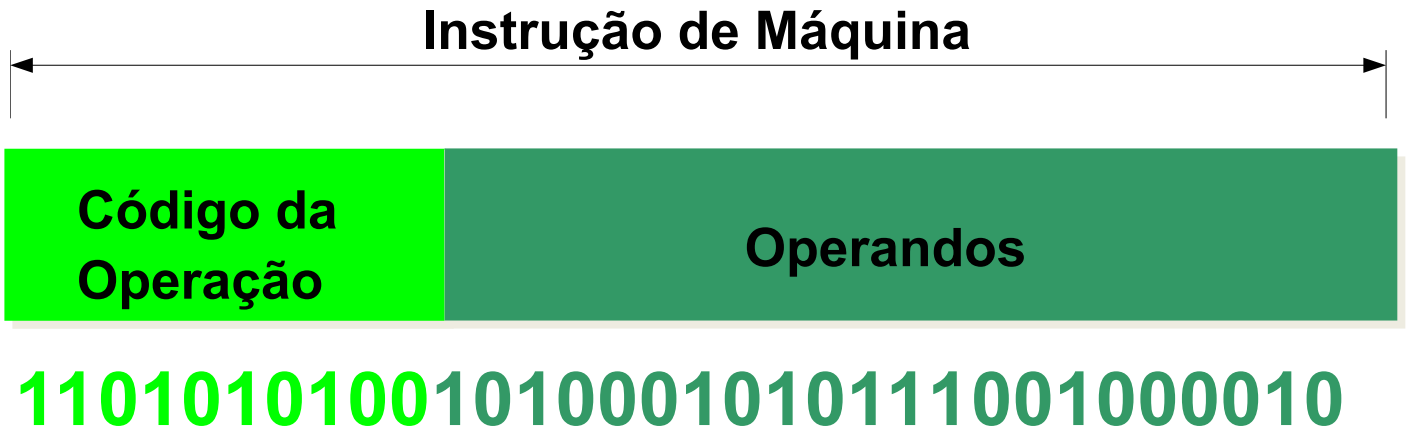
Refere-se a ordem que os dados são armazenados na memória

Endereço	Big Endian	Little Endian
120	12	78
121	34	56
122	56	34
123	78	12

Little Endian: O endereço de um dado (0x120) é o endereço do byte menos significativo (0x78)

Big Endian: O endereço de um dado (0x120) é o endereço do byte mais significativo (0x12)

Arquitetura do Conjunto de Instruções



- A **instrução de máquina** é um conjunto de bits contendo dois campos:
  - Código da Operação (opcode): o que a instrução faz
    Exemplos: add; lw; jump; beq
  - Operandos (operand): onde estão os dados a serem manipulados
    exemplos: constante; registrador, memória

Arquitetura do Conjunto de Instruções

▶ O campo operando pode variar quanto a:

Quantidade de operandos:

- 3 operandos - SUB R1,R2,R3; R1 ← R2 - R3
- 2 operandos - MOV R1,R2; R1 ← R2
- 1 operando - ADD X; W ← W + (X)
- 0 operandos NOP

Modos de Endereçamento: como interpretar o campo operando no que diz respeito a onde se encontra o dado utilizado pela instrução

Arquitetura do Conjunto de Instruções

▶ Quantidade de Operandos

3 Operandos

(Op1) ← (Op2) operação (Op3)

ADD Op1, Op2, Op3; (Op1) ← (Op2) + (Op3)

SUB Op1, Op2, Op3; (Op1) ← (Op2) – (Op3)

MUL Op1, Op2, Op3; (Op1) ← (Op2) * (Op3)

DIV Op1, Op2, Op3; (OP1) ← (Op2) / (Op3)

Instrução é completa pois permite especificar os dois operandos de origem e destino explicitamente

A quantidade de operandos e a quantidade de bits ocupada pelos mesmos pode limitar o espaço de endereçamento ou valor representado

Arquitetura do Conjunto de Instruções

▶ Quantidade de Operandos

2 Operandos

(Op1) ← (Op1) operação (Op2)

ADD Op1, Op2; (Op1) ← (Op1) + (Op2)

SUB Op1, Op2; (Op1) ← (Op1) – (Op2)

MUL Op1, Op2; (Op1) ← (Op1) * (Op2)

DIV Op1, Op2; (Op1) ← (Op1) / (Op2)

MOV Op1, Op2; (Op1) ← (Op2)

- Instrução não é completa (operando Op1usado implicitamente)
- Instrução MOV deve ser implementada

Arquitetura do Conjunto de Instruções

▶ Quantidade de Operandos

1 Operando

W ← W operação (Op)

ADD Op;	W ← W + (Op)
SUB Op;	W ← W – (Op)
MUL Op;	W ← W * (Op)
DIV Op;	W ← W / (Op)
LDA Op;	W ← (Op)
STR Op	(Op) ← W

- Utilização de um registrador especial (W); As operações ocorrem entre o operando na memória e este registrador
- Instruções LDA e STR devem ser implementadas

Arquitetura do Conjunto de Instruções

▶ Quantidade de Operandos

0 Operandos

Utilizado em casos em que a própria instrução indica o dado/operando manipulado, ou quando a instrução não exige dado ou operando;

Exemplos:
```
NOP    ; NO Operation
CLRW   ; W ← 0
```

Arquitetura do Conjunto de Instruções

- **Modos de Endereçamento:** como interpretar o campo operando no que diz respeito a <u>onde se encontra o dado utilizado pela instrução</u>

  - Imediato
  - Direto
  - Indireto
  - Registrador Direto
  - Registrador Indireto
  - Base + Deslocamento

  ...

Arquitetura do Conjunto de Instruções

- **Modo Imediato:** o dado a ser manipulado está indicado no próprio campo operando da instrução;

Utilizadas na inicialização de variáveis e ponteiros; operações com constantes e desvios;

- Vantagem: poucos acessos a memória
- Desvantagem: limitação do campo operando restringe o valor máximo manipulado

Exemplo:

ADDI R1, R2, 3; R1 ← R2 + 3

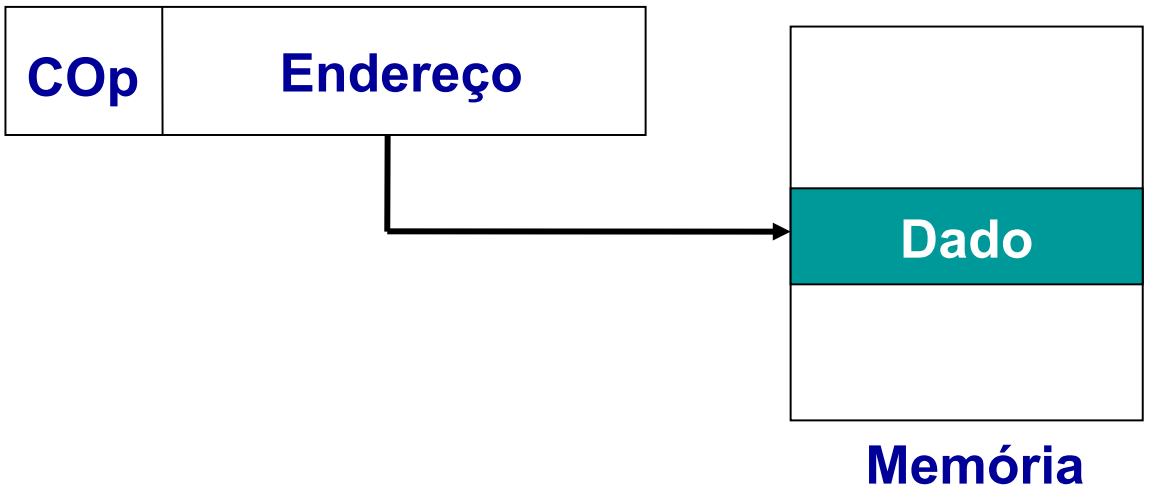
Arquitetura do Conjunto de Instruções

Modo Direto

- O valor contido no campo operando indica o endereço de memória onde se localiza o dado a ser manipulado
- Um dos formatos naturais de implementar as variáveis do programa. Cada variável representa um endereço de memória

Exemplo:

LDA X; W ← (X)

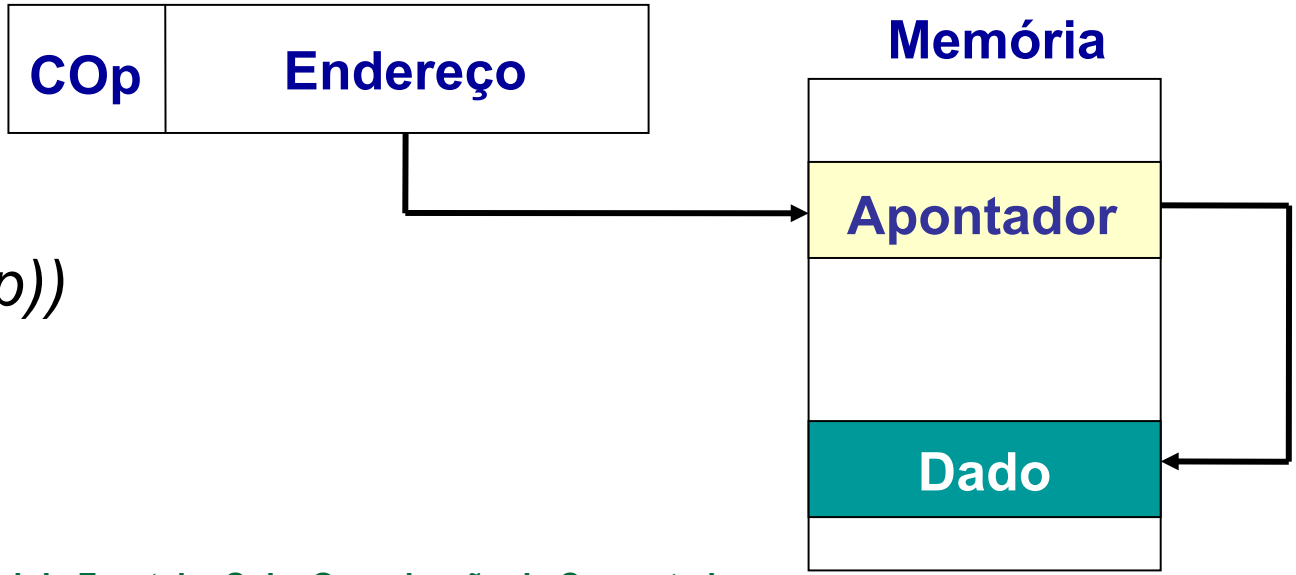


Arquitetura do Conjunto de Instruções

Arquitetura do Conjunto de Instruções

Modo Indireto

- O campo operando representa uma célula de memória, entretanto o valor contido neste endereço não é o dado e sim o endereço onde o dado se encontra
- Utilizado na implementação de ponteiros



Exemplo:

LDAIn Op; W ← ((Op))

LDAin ptr;

Arquitetura do Conjunto de Instruções

Registrador Direto

- O campo operando representa o número do registrador onde o dado se encontra
- Vantagens:
  - acesso aos registradores é mais rápida que o acesso a memória
  - número menor de bits para endereçar registradores
- Principal modo de endereçamento nas arquiteturas RISC

Exemplo:

ADD R1, R2, R3; R1← R2 + R3

Arquitetura do Conjunto de Instruções

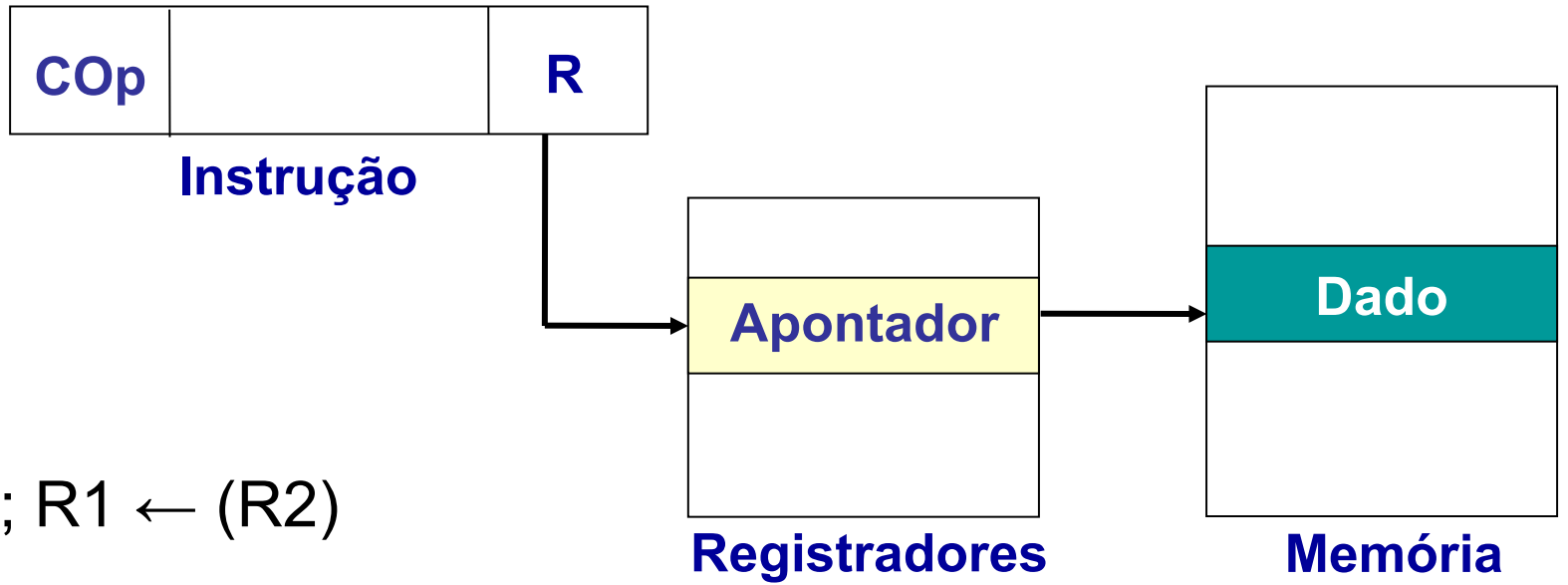
Registrador Indireto

- O operando (registrador) aponta para o endereço de memória onde o dado se encontra;
- Pode ser implementado de várias maneiras; os modos a seguir são casos especiais deste modo:
  - Indireto
  - Base+deslocamento

Arquitetura do Conjunto de Instruções

Registrador Indireto

- Indireto: operando é um registrador cujo conteúdo aponta para um endereço de memória



- Exemplo:

LDAr R1,R2; R1 ← (R2)

Arquitetura do Conjunto de Instruções

Indireto: Base + Deslocamento

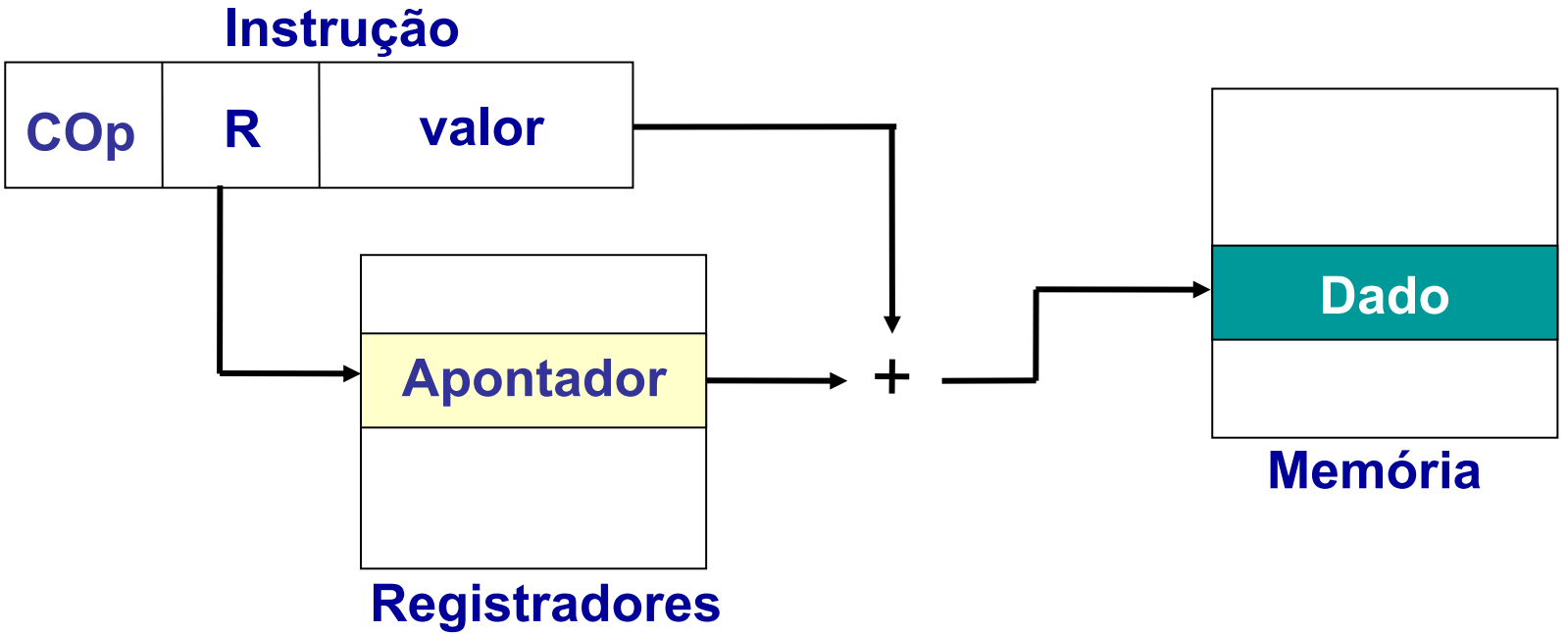
- Neste modo o registrador base aponta para o início de um bloco e o deslocamento informa qual é o deslocamento dentro daquele bloco
- O endereço onde o dado se encontra é obtido a partir da soma entre o valor contido no registrador base e no deslocamento (valor imediato contido na instrução)

Arquitetura do Conjunto de Instruções

Indireto: Base + Deslocamento

Exemplo:

LW R1, 4(R2) ; R1 ← (R2 + 4)



Arquitetura do Conjunto de Instruções

Relativo ao PC

- Há uma referência implícita ao contador de programa (PC)
- Utilizado para realizar desvios no programa em instruções de desvio condicional
- Explora o conceito de localidade de referências

Exemplo:

BEQ R1, R2, 12; if (R1 == R2) then
PC ← PC + 12
else
PC ← PC + 4