Linguagens de Montagem
Capítulo 4 - Transferência de dados

Ricardo Anido
Instituto de Computação
Unicamp

Instruções de transferência de dados

- Permitem a transferência de dados entre dois registradores do processador ou entre a memória e um registrador.
- Já vimos uma: carrega constante em registrador.

Copia registrador

Copia o valor de um registrador (chamado registrador fonte) para um outro registrador (chamado registrador destino).

MOV

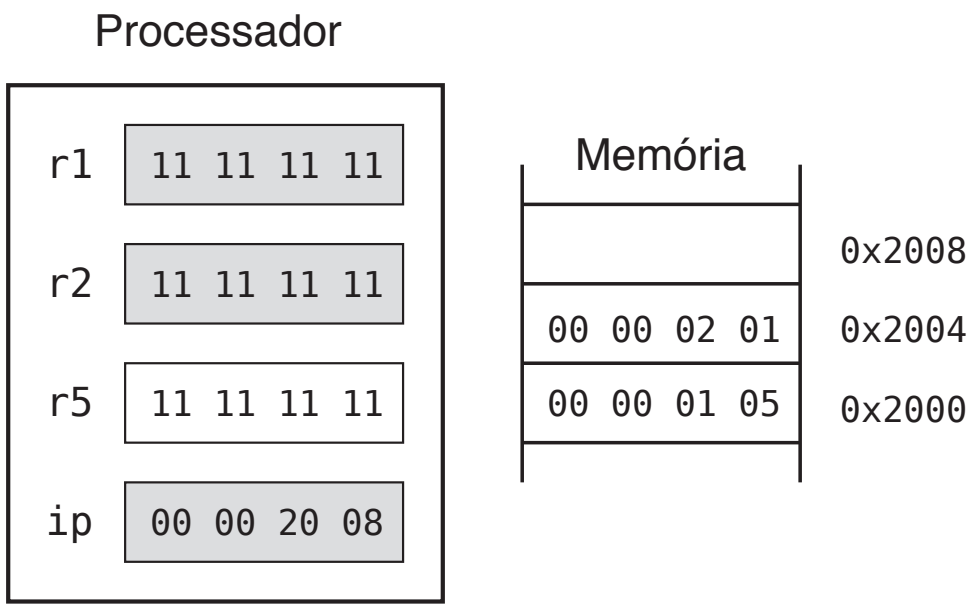
Copia Registrador			
Sintaxe	Operação	Flags	Codificação
mov *rd*, *rf*	$rd \leftarrow rf$	–	31 … 0: 0x00 | – | *rd* | *rf*

Figura: Descritor da instrução MOV.

Copia registrador

```
Endereço     Código                          Programa

                          |@ exemplos de instrução mov
00002000 [00 00 01 05] |   mov   r1,r5     @ copia valor de r5 para r1
00002004 [00 00 02 01] |   mov   r2,r1     @ copia valor de r1 para r2
```



Carrega registrador

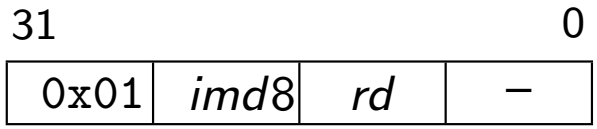
- Carrega um registrador com valor da memória.
- Diversas formas, com diferentes *modos de endereçamento*.

Endereçamento imediato

- no endereçamento imediato o valor do operando faz parte do próprio código da instrução.
- Na instrução *Carrega registrador com endereçamento imediato* o registrador destino é carregado com o valor do operando, dado com endereçamento imediato.
- o valor carregado no registrador é sempre o mesmo!
- utilizada quando se deseja carregar um valor constante no registrador destino.

Carrega registrador com endereçamento imediato

SET

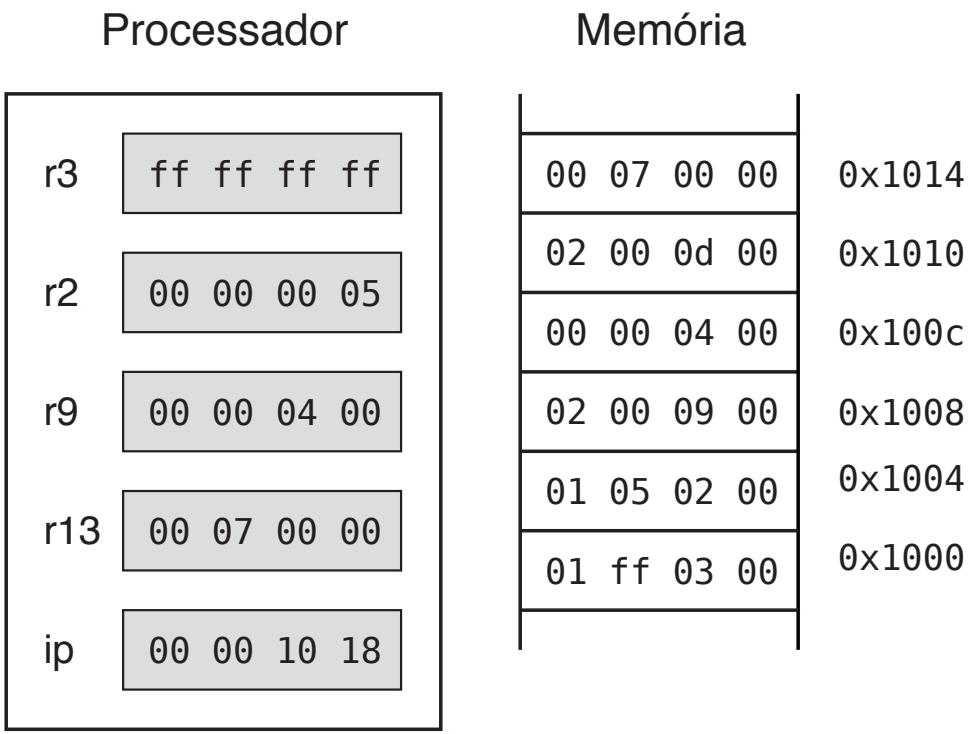
Carrega registrador com endereçamento imediato			
Syntax	Operação	Flags	Codificação
set *rd*, *expr*8	*rd* ← *ext*32(*imd*8)	–	
set *rd*, *expr*32	*rd* ← *imd*32	–	

Carrega registrador com endereçamento imediato

```
Endereço     Código                          Programa

                          |@ exemplos de instrução set (ender. imediato)
00001000 [01 ff 03 00] |    set   r3,-1       @ instrução é codificada
                          |                    @ em uma palavra
00001004 [01 05 02 00] |    set   r2, 5       @ instrução é codificada
                          |                    @ em uma palavra
00001008 [02 00 09 00] |    set   r9,0x400    @ valor não pode ser
         [00 00 04 00] |                      @ representado em 8 bits
                          |                    @ instrução tem 2 palavras
00001010 [02 00 0d 00] |    set   r13,var1    @ carrega endereço de
         [00 07 00 00] |                      @ var1, usa duas palavras
                          |    ...             @ outras instruções
                          |                    @ não mostradas
                          |    .org   0x70000
                          |var1:
00070000 [00 00 00 02] |    .byte  2          @ uma variável associada
                          |                    @ ao endereço 0x70000
```

Carrega registrador com endereçamento imediato

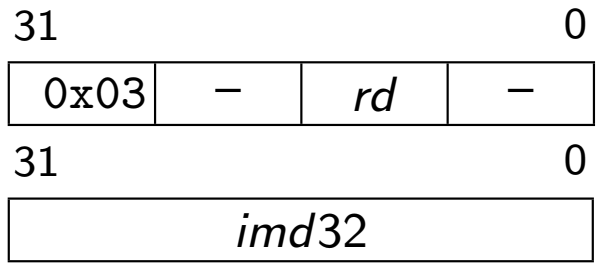


Endereçamento direto

- No endereçamento direto, a a instrução contém o *endereço* da posição de memória cujo valor se deseja acessar.
- A instrução *Carrega registrador com endereçamento direto* (LD) carrega um registrador com o valor de uma posição de memória, cujo endereço é especificado na instrução.

Carrega registrador com endereçamento direto

LD

Carrega registrador com endereçamento direto			
Syntax	Operação	Flags	Codificação
ld *rd*, *expr*32	*rd* ← *mem*[*imd*32]	–	

Carrega registrador com endereçamento direto

Para executar uma instrução LD com endereçamento direto o processador faz três acessos à memória:

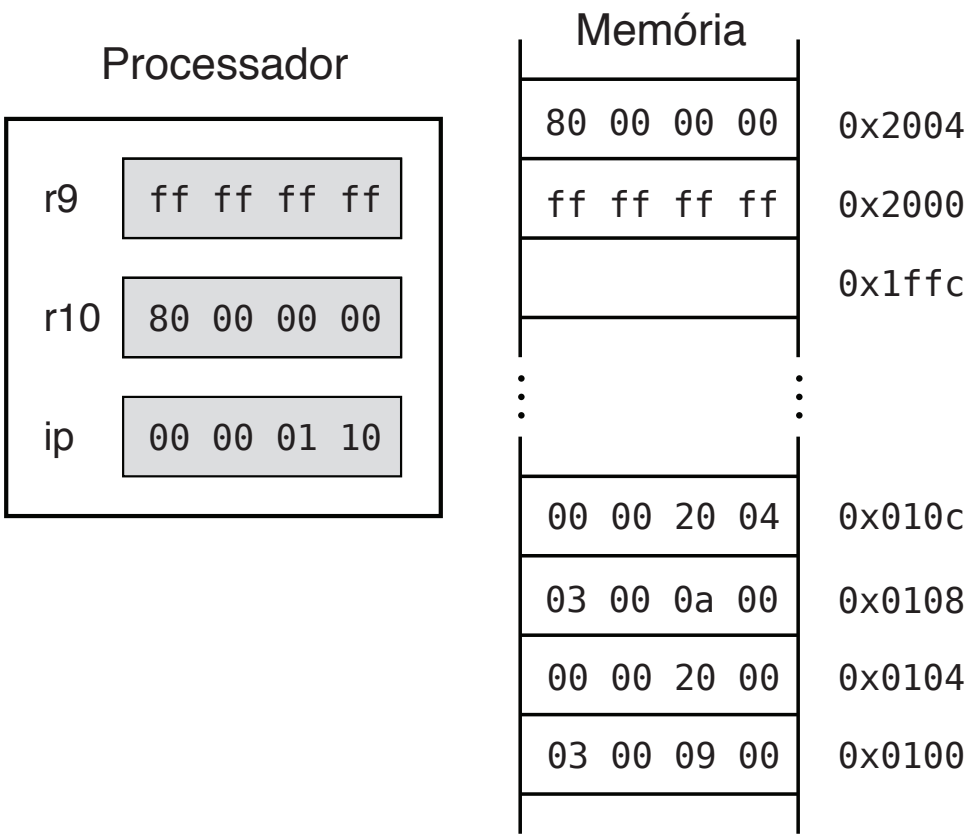
- um acesso no endereço apontado pelo registrador `ip`, para a busca da primeira palavra da instrução;
- outro acesso no endereço `ip+4`, para a busca da segunda palavra da instrução (*imd*32), que especifica o endereço do operando; e
- ainda outro acesso para a busca do valor operando (no endereço obtido no segundo acesso). O valor obtido nesse último acesso é armazenado no registrador *rd*.

Carrega registrador com endereçamento direto

```
Endereço     Código                          Programa

                          | @ exemplos de instrução ld (ender. direto)
                          |
00000100 [03 00 09 00]    |       ld   r9,cont      @ carrega conteúdo do
         [00 00 20 00]    |                         @ endereço cont
00000108 [03 00 0a 00]    |       ld   r10,cont+4   @ carrega conteúdo do
         [00 00 20 04]    |                         @ endereço cont+4
                          |       ...
                          |
                          |       .org 0x2000
                          | cont:                   @ uma variável
00002000 [ff ff ff ff]    |       .word -1          @ associada
00002004 [80 00 00 00]    |       .word 0x80000000  @ ao endereço 0x2000
```

Carrega registrador com endereçamento direto



Endereçamento indireto por registrador

- endereço do operando é dado em um registrador, ao invés de ser um valor constante, codificado na instrução, como no endereçamento direto.
- Em linguagem de montagem usa o mesmo comando LD já utilizado anteriormente, mas indicaremos o modo de endereçamento distinto pela grafia do operando fonte.

Endereçamento indireto por registrador

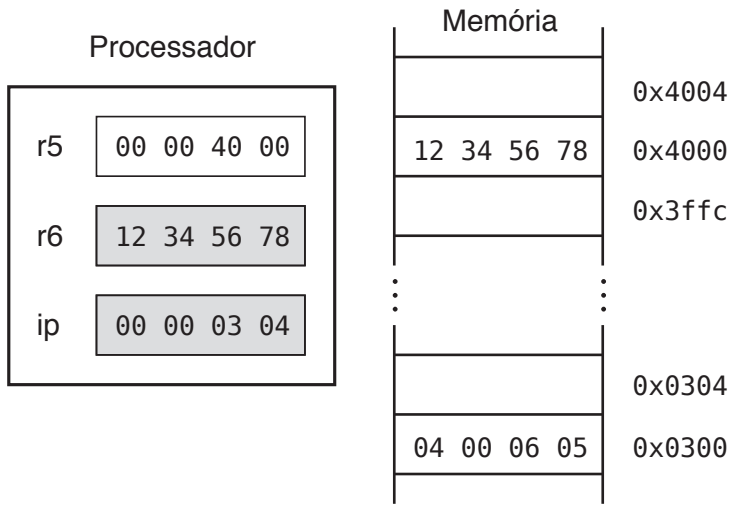
LD

Carrega registrador com endereçamento indireto por registrador			
Sintaxe	Operação	Flags	Codificação
ld $rd, [rf]$	$rd \leftarrow mem[rf]$	–	31 … 0: 0x04 | – | rd | rf

Carrega registrador com endereçamento indireto por registrador

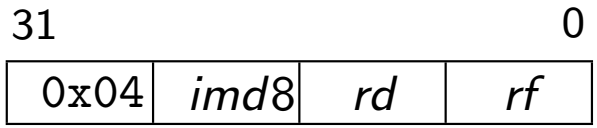
```
Endereço     Código                          Programa

                          |@ exemplo instrução ld (indir. por registrador)
00000300 [04 00 06 05] |     ld  r6, [r5]       @ instrução ld
```



Carrega registrador com endereçamento indireto por registrador mais constante

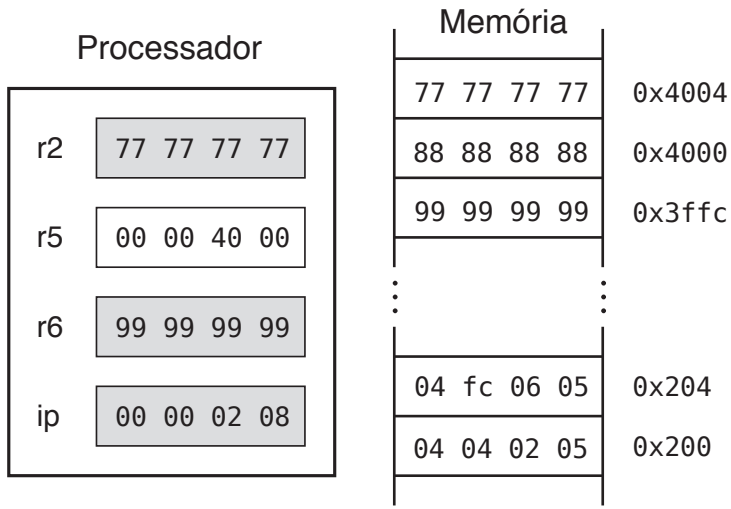
LD

Carrega registrador com ender. indireto por registrador mais constante			
Sintaxe	Operação	Flags	Codificação
ld $rd, [rf + expr8]$	$rd \leftarrow mem[rf+ext(imd8)]$	–	

Carrega registrador com endereçamento indireto por registrador mais constante

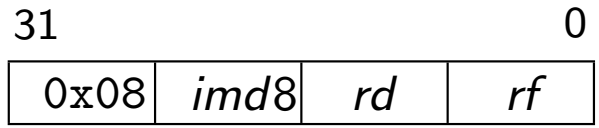
```
Endereço     Código                          Programa

                        |@ instrução ld (endereçamento indireto por
                        |@ registrador mais constante)
00000200 [04 04 02 05] |     ld  r2,[r5+4]
00000204 [04 fc 06 05] |     ld  r6,[r5-4]
```



Carrega registrador com byte

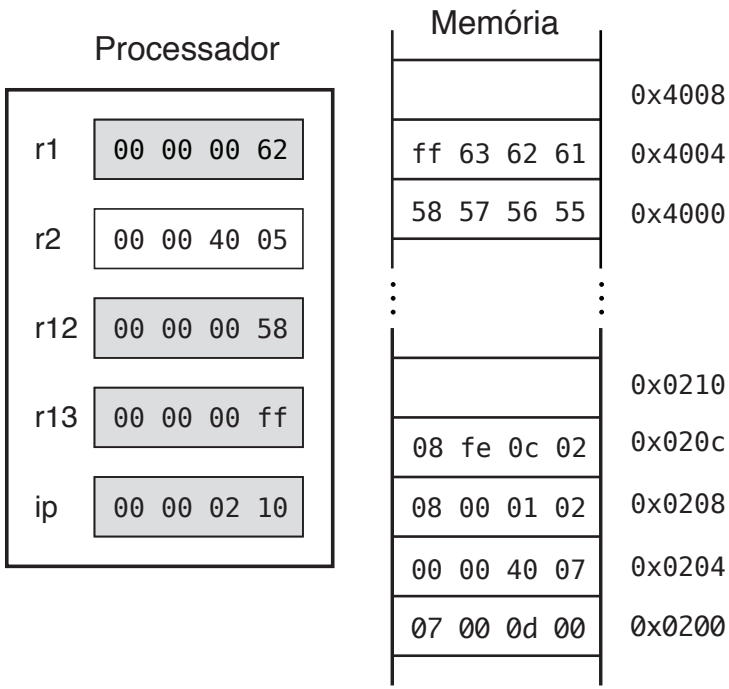
LDB

Carrega registrador com byte			
Syntax	Operação	Flags	Codificação
ldb *rd*, *expr*32	$rd \leftarrow mem8[imd32]$	–	
ldb *rd*, *[rf+expr8]*	$rd \leftarrow mem8[rf + ext(imd8)]$	–	

Carrega byte em registrador

```
                        | @ exemplos de instrução ldb
                        |
00000200 [07 00 0d 00]  |     ldb  r13,carac+7  @ carrega byte cujo
         [00 00 40 07]  |                       @ endereço é carac+7
                        |
00000208 [08 00 01 02]  |     ldb  r1,[r2]      @ r2 tem endereço do
                        |                       @ byte que queremos
                        |                       @ carregar em r1
                        |
0000020c [08 fe 0c 02]  |     ldb  r12,[r2-2]   @ r2-2 tem endereço do
                        |                       @ byte que queremos
                        |                       @ carregar em r12
                        | ...
                        |      .org 0x4000
                        | carac:                @ uma variável que
                        |                       @ contém uma sequência
                        |                       @ de bytes
00004000 [58 57 56 55]  |      .byte  'U', 'V', 'W', 'X'
00004004 [ff 63 62 61]  |      .byte  'a', 'b', 'c', 0xff
```

Carrega byte em registrador

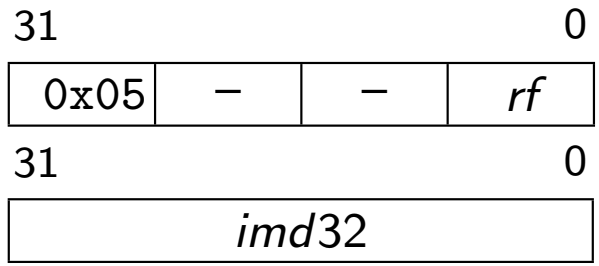
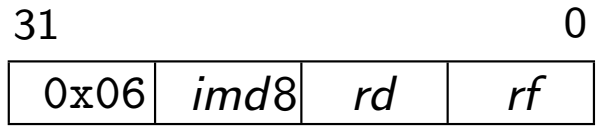


Armazena registrador

- A instrução *Armazena registrador* (ST, do inglês *store*, armazenar), efetua a operação inversa à operação LD, armazenando na memória o valor de um registrador.
- Duas variantes, com endereçamento direto e indireto por registrador mais constante.

Armazena registrador

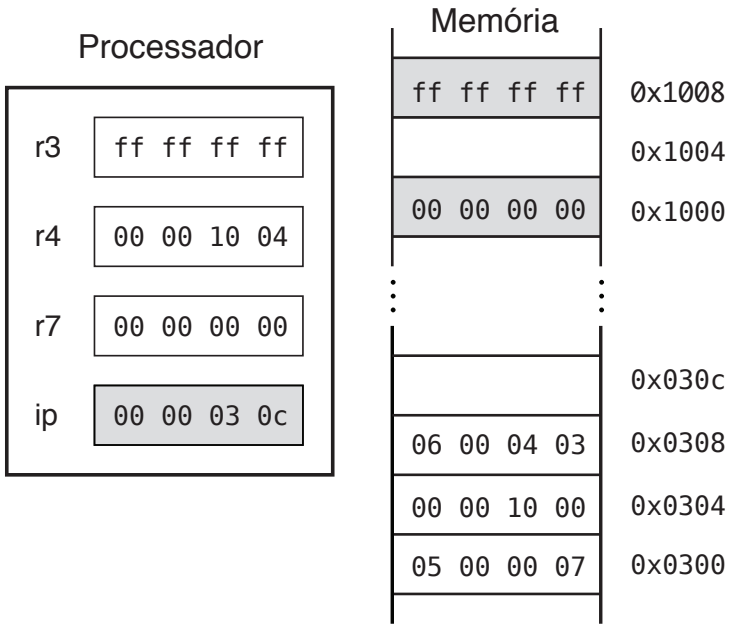
ST

Armazena registrador em memória			
Syntax	Operação	Flags	Codificação
st *expr*32, *rf*	*mem*32[*imed*32] ← *rf*	–	
st [*rd* + *expr*8], *rf*	*mem*32[*rd* + *ext*(*imd*8)] ← *rf*	–	

Armazena registrador

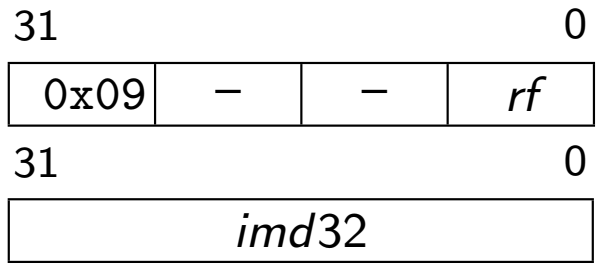
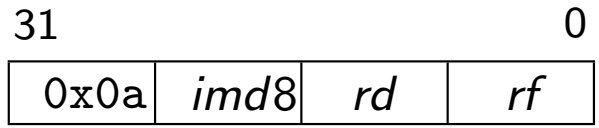
```
                        |@ exemplos de instrução st
                        |
00000300 [05 00 00 07]  |      st  var,r7          @ armazena registrador
         [00 00 10 00]  |                          @ r7 na variável var
00000308 [06 04 04 03]  |      st  [r4+4],r3       @ r4+4 tem endereço
                        |                          @ onde queremos
                        |                          @ armazenar r3
                        |       .org 0x1000
                        |var:                      @ uma variável montada
00001000                |       .skip 4            @ no endereço 0x1000
```

Armazena registrador



Armazena byte de registrador

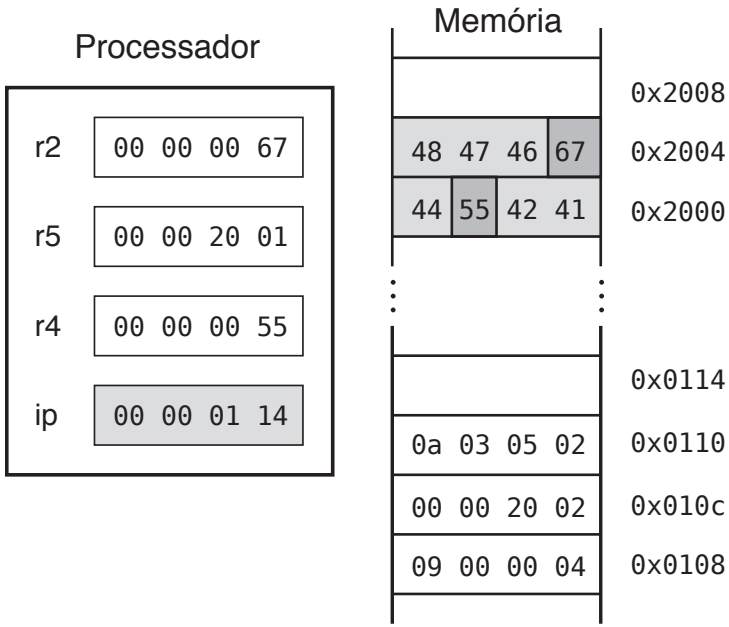
STB

Armazena byte de registrador em memória			
Syntax	Operação	Flags	Codificação
stb *expr*32, *rf*	*mem*8[*imed*32] ← *rf*	–	
stb [*rd* + *expr*8]], *rf*	*mem*8[*rd* + *ext*(*imd*8)] ← *rf*	–	

Armazena byte de registrador

```
                          |@ exemplo de instruções stb
                          |
00000108 [09 00 00 04]    |      stb  carac+2,r4    @ armazena byte de r4
         [00 00 20 02]    |                         @ no endereço carac+2
00000110 [0a 03 05 02]    |      stb  [r5+3],r2     @ r5 tem endereço onde
                          |                         @ byte em r2
                          |                         @ instrução ocupa uma
                          |                         @ palavra
                          |
                          |       .org 0x2000
                          | carac:                  @ uma variável montada
00002000                  |       .byte 'ABCDEFGH'  @ no endereço 0x2000
                          |                         @ que representa uma
                          |                         @ sequência de oito
                          |                         @ caracteres
```

Armazena byte de registrador



Usos de modos de endereçamento

- Os diferentes modos de endereçamento fornecidos por um processador podem ser utilizados para implementar diferentes conceitos em linguagens de programação.
- O modo de endereçamento direto pode ser usado para carregar e armazenar valores de variáveis que podem ser acessadas diretamente, através dos rótulos associados às variáveis.

Exemplo

```
// Trecho de programa em C
#define TAMANHO 256

int a,b;
char c,d=1;
...

a = TAMANHO;
b = a;
c = 'a';
```

Exemplo

```
@ definição de uma constante
TAMANHO .equ 256

@ reserva espaço para as variáveis
    .org  0x400
a:  .skip 4                 @ variável inteira, quatro bytes
b:  .skip 4                 @ variável inteira, quatro bytes
c:  .skip 1                 @ variável char, reserva um byte
d:  .byte 1                 @ variável char, inicializada, valor é 1

    .org  0x1000
    set   r0,TAMANHO        @ r0 usado como auxiliar
    st    a,r0              @ armazena valor na variável a
    st    b,r0              @ armazena valor na variável b
    set   r1,'a'            @ r1 usado como auxiliar; note
                            @ que r0 poderia ser reutilizado
                            @ ao invés de utilizar r1
    stb   c,r1              @ armazena caractere na variável c
```

Usos de modos de endereçamento

- O modo de endereçamento indireto por registrador pode ser usado para implementar o conceito de apontadores em linguagens de alto nível: carregamos o endereço de uma variável em um registrador, e usamos esse registrador para acessar essa variável.
- Se a variável é uma coleção de valores, por exemplo um vetor de inteiros, podemos *percorrer* os elementos da variável incrementando ou decrementando o registrador para acessar diferentes elementos.

Exemplo

```
// Trecho de programa em C

int x=1,vet[100],*p;
...

p = &vet[0];
*p = x;
```

Exemplo

```
@ reserva espaço para as variáveis
      .org  0x400
x:    .word 1           @ variável inteira, valor 1
vet:  .skip 4*100       @ variável vetor, inteira, 100 elementos
p:    .skip 4           @ variável apoontador, quatro bytes

...
      set   r0,vet      @ r0 usado como apontador p, tem endereço de vet
      st    p,r0        @ armazena valor na variável p
      ld    r1,x        @ r1 tem valor da variável x
      st    [r0],r1     @ armazena valor de x no elemento 0 de vet
```

Usos de modos de endereçamento

- O modo de endereçamento indireto pode ser usado para acessar uma variável do tipo estrutura, como o tipo `struct` em C, que pode armazenar uma coleção valores de tipos distintos.

Exemplo

```
// Trecho de programa em C

struct exemplo {
  int r;
  char s;
  char t;
};

struct exemplo exe;
...

exe.r = 0;
exe.s = 'a';
exe.t = 0;
```

Exemplo

```
@ reserva espaço para as variáveis
      .org  0x400
exe:  .skip   6             @ variável ocupa seis bytes

...
      set r0,exe            @ r0 usado como apontador, tem endereço de exe
      set r1,0              @ vamos usar para atribuir o valor
      st  [r0],r1           @ zero ao elemento r da variável exe
      set r2,0x41           @ vamos usar r2 para atribuir o valor
      stb [r0+4],r2         @ 'a' ao elemento s da variável exe
      stb [r0+5],r1         @ armazena valor zero no elemento t
```