ELTD03z
Microcontroladores/Microprocessadores

Teoria_03a1_2b

Prof. Enio R. Ribeiro

Universidade Federal de Itajubá - UNIFEI

T3.0.1 Modos de endereçamento: Revisão!

STM32F103C8
MODELO FUNCIONAL

R0	000022AA
R1	00005511
R2	
R3	000077BB
R4	
R5	
R6	
R7	
R8	
R9	
R10	
R11	
R12	
R13	
R14	
R15	
PSR	

```
     export   __main
     area     m_prog,code,readonly
__main
     mov  r0,#0x22aa  ;end.imediato
     mov  r1,#0x5511
     add  r3,r0,r1    ;end.inerente
     nop
aki  b    aki
     end
```

```
                    __main:
0x080002BC F24220AA  MOVW      r0,#0x22AA
0x080002C0 F2455111  MOVW      r1,#0x5511
0x080002C4 EB000301  ADD       r3,r0,r1
0x080002C8 BF00      NOP
0x080002CA E7FE      B         0x080002CA
```

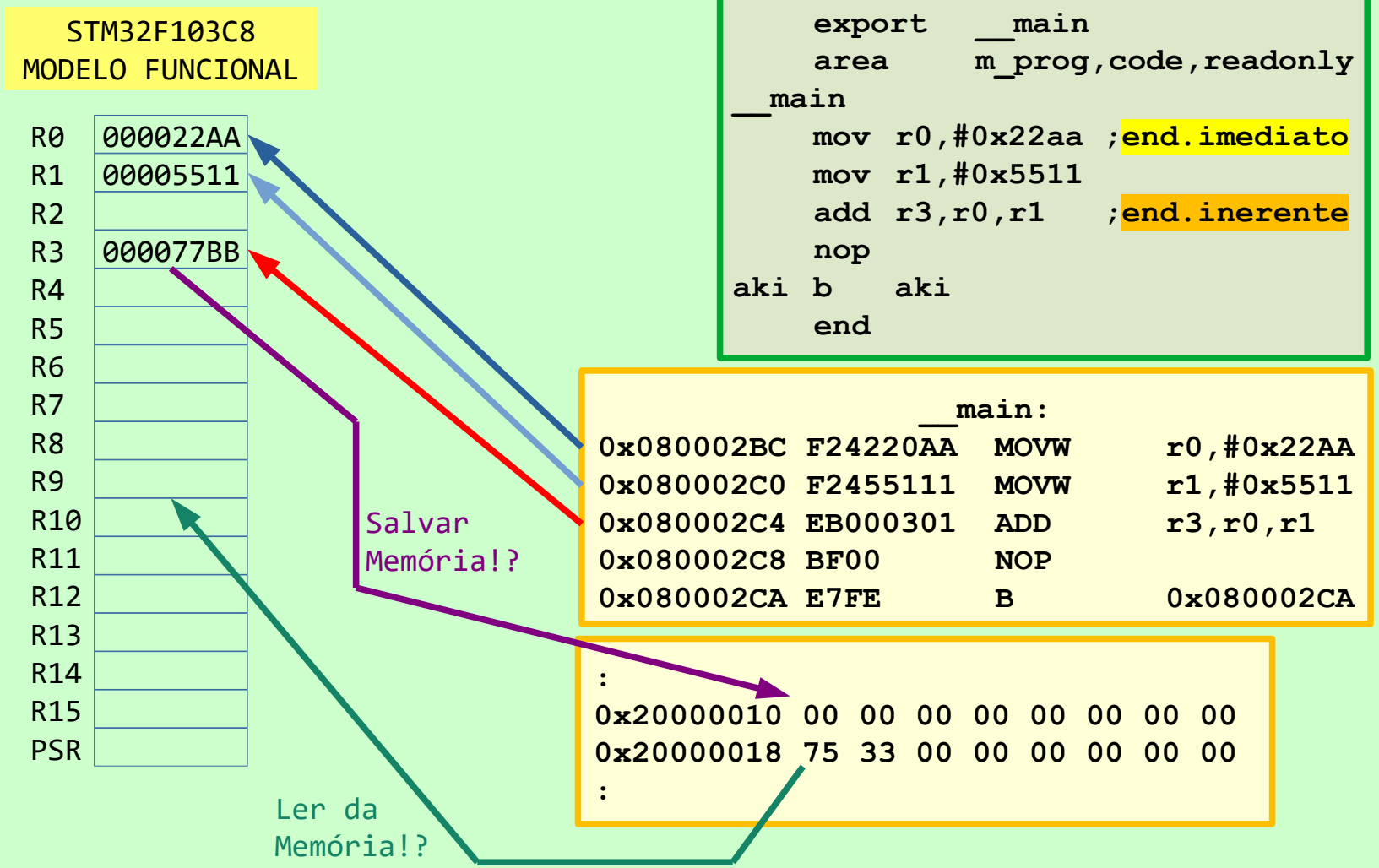
Salvar
Memória ?

```
:
0x20000010 00 00 00 00 00 00 00 00
0x20000018 75 33 00 00 00 00 00 00
:
```

Salvar memória => escrever, armazenar, gravar (na memória).

Instrução: “SAVE R0, 0x20000010” ?!?!

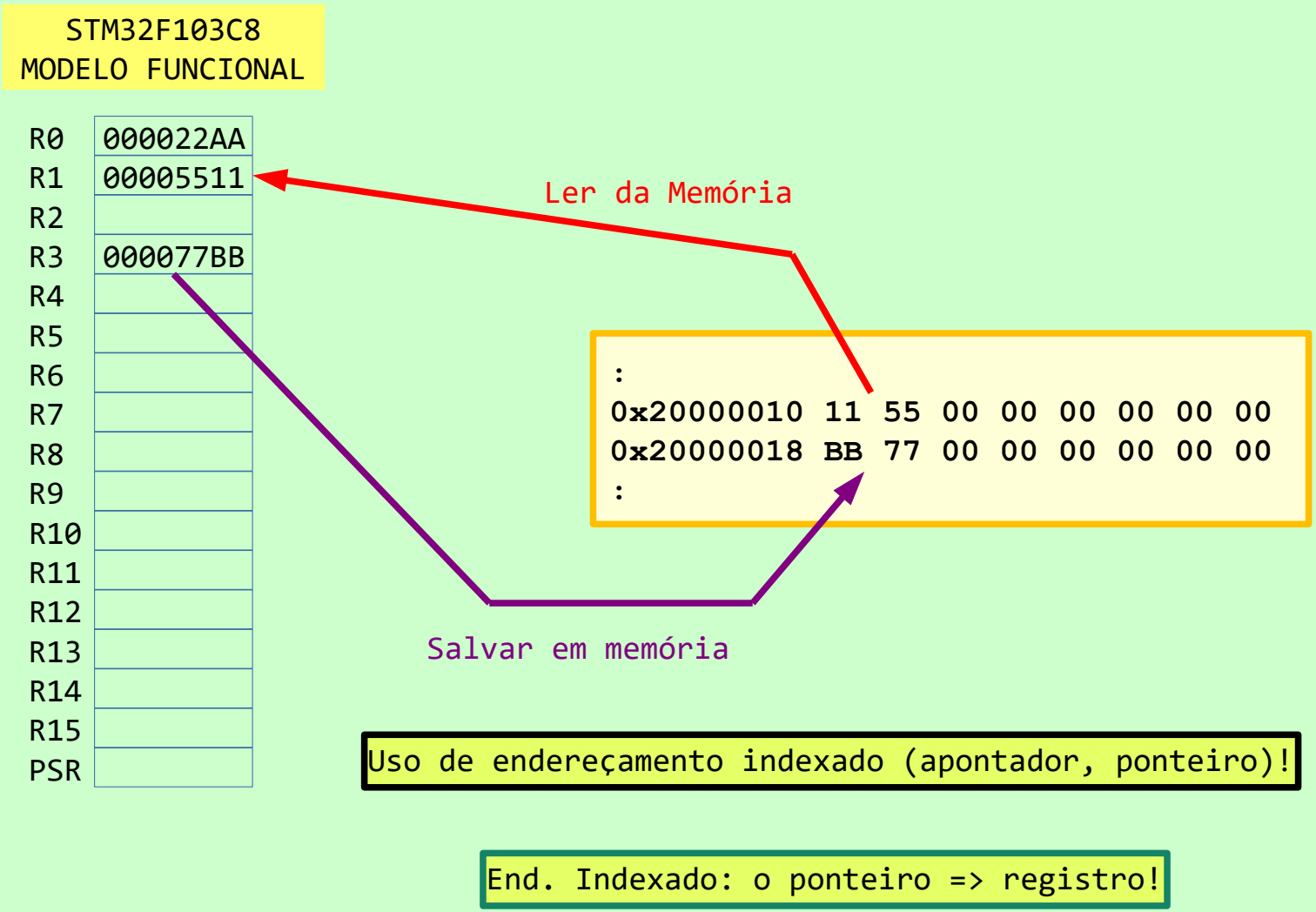
T3.0.1 Modos de endereçamento: Revisão!



Ler da memória => carregar, copiar da memória.

Instrução: “LOAD R0, 0x20000018” ?!?!

T3.0.1 Modos de endereçamento: INDEXADO



T3.0.1 Modo de endereçamento: INDEXADO

Endereçamento indexado:

1) inicializar o registro apontador (valor inicial do registro) ;
2) usar as instruções de endereçamento indexado.

Criar address pointer (apontador de endereço) with ARM load "pseudo-op" (pseudo-instrução)

- Load a **32-bit constant** (data, address, etc.) into a register

- Possibilidade: `mov rd,#constant_LO; movt rd,#constant_HI`
  NÃO É USADO, POIS O ENDEREÇO É, GERALMENTE, DESCONHECIDO!

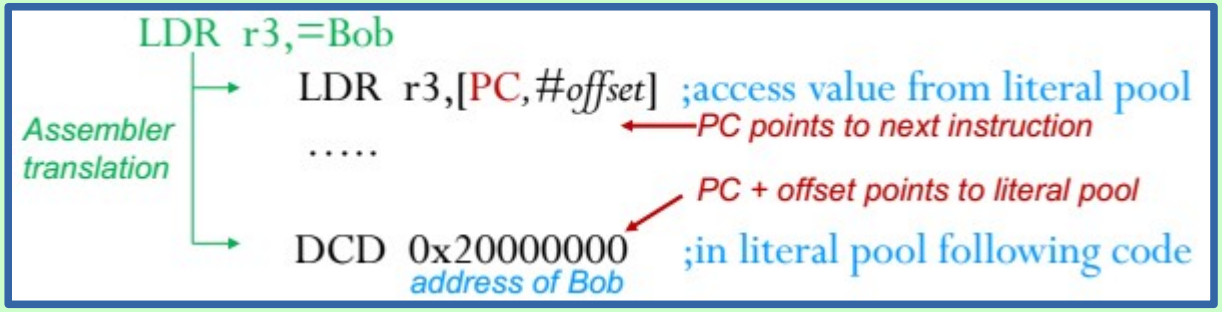
- Use a ***pseudo-operation*** ***(pseudo-op)***, a qual é convertida by the assembler to one or more actual ARM instructions

T3.0.1 Modo de endereçamento: INDEXADO

Exemplo de apontador de endereço (Address pointer)

```
    AREA D1,DATA,READWRITE ;assume D1 starts at 0x20000000
Bob DCD &a3f4eedd
    :
    AREA C1,CODE,READONLY
    :
    LDR r3,=Bob ;Load address of Bob into r3
    LDR r2,[r3] ;Load value of variable Bob into r2
    :
```

- Assembler stores address of Bob (constant 0x20000000) in the "literal pool" in code area C1
- Constant loaded from literal pool address [PC,#offset]



T5.7a) MODO ENDEREÇAMENTO INDEXADO -> possibilidades

Instruction: **LDR/STR**

LDR instructions load one (or two) register(s) with a value from memory. STR instructions store one (or two) register value(s) to memory.

Syntax

```
op{type}{cond} Rt, [Rn {, #offset}]    ; immediate offset
```

- '*op* ' is either LDR (load register) or STR (store register);
- '*type* ' is one of the following: **B**, **SB**, **H**, **SH** or omit, for word;
- '*cond* ' is an optiional conditional code;
- '*Rt* ' is the register to load or store;
- '*Rn* ' is the register on which the memory address is based;
- '*offset* ' is an offset from *Rn*. If *offset* is omitted, the address is the contents of *Rn*.

(memory to/from register) – **não é pseudo-instrução**

- **LDRH** – load halfword (16 bit unsigned #) / zero-extend to 32 bits
- **LDRSH** – load signed halfword / sign-extend to 32 bits
- **LDRB** – load byte (8 bit unsigned #) / zero-extend to 32 bits
- **LDRSB** – load signed byte / sign-extend to 32 bits
- **STRH** – store 16-bit halfword (right-most16 bits of register)
- **STRB** - store 8-bit byte (right-most 8 bits of register)

T5.7b) MODO ENDEREÇ. INDEXADO OFFSET IMMEDIATE -> offset fixo

Operation

Load and store instructions with immediate offset can use the following addressing modes:

• Offset addressing (or Offset indexed) => **END. INDEXADO COM OFFSET IMEDIATO (FIXO)**

The offset value is added to or subtracted from the address obtained from the register *Rn*. The result is used as the address for the memory access. The register *Rn* is unaltered. The assembly language syntax for this mode is:

[*Rn*, *#offset*]

Table 25. Immediate offset ranges

Instruction type	Immediate offset
Word, halfword, signed halfword, byte or signed byte	-255 to 4095

Fonte da “Table 25”: **cd00228163-stm32f10xxx-cortex-m3-programming-manual-stmicroelectronics.pdf**

T5.8.1a) MODO END. INDEX. COM OFFSET IMMEDIATE (fixo)

O offset é um número sinalizado a ser adicionado ao conteúdo do registro base (ou a ser subtraído do conteúdo do registro base). O resultado é usado como endereço para acesso à memória. **O registro Rn não é alterado (Rn é o registro base)**.

Immediate offset indexed addressing é útil para acessar elementos que estão a certa distância do local indicado/apontado pelo registro base (base/index/pointer register), por exemplo, as stack offsets and input/output registers.

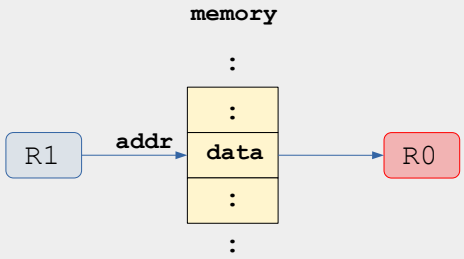
Syntax:

Load data from memory: `LDR{type} {cond} Rt, [Rn {, #imm }]`

Save data to memory: `STR{type} {cond} Rt, [Rn {, #imm }]`

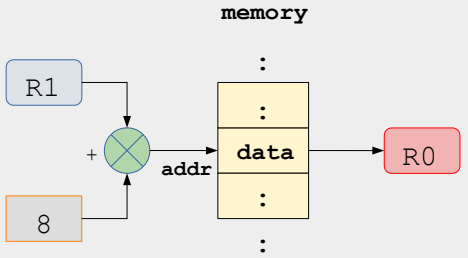
Load with zero immediate offset

```
LDR   R0, [R1]    ;R0<=[R1]
LDR   R0, [R1,#0] ;R0<=[R1]
```



Load with immediate offset (addressed by R1+8)

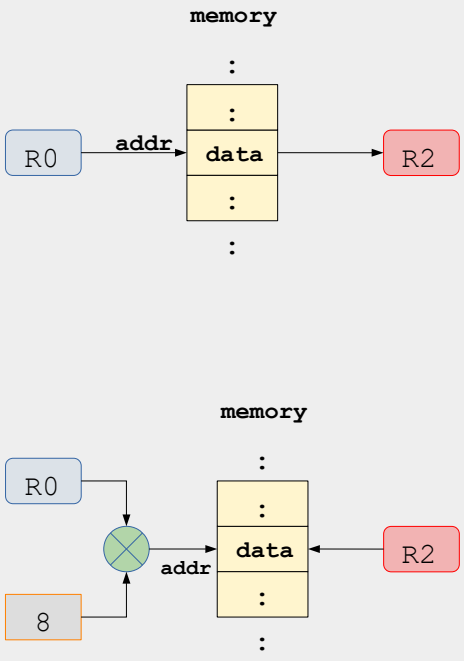
```
LDR       R0,[R1,#8]    ;R0<=[R1+8]
```



T5.8.1b) MODO END. INDEX. COM OFFSET IMMEDIATE (fixo)

Ex. 1d – O vetor vt1 tem 8 bytes. Faça um programa para copiar os bytes de vt1 no vetor vt2. O programa é cíclico. Faça as designações e alocações necessárias (FDAN). (O usuário calcula o offset entre os vetores!).

```
;---Ex. 1d - End.Index. Offset imediato
     export   __main
;=== diretiva area - dados (sram)
     area     dds1, data, readwrite
vt1  space    8
vt2  space    8
;=== diretiva area - prog. (flash)
     area     m_prog, code, readonly
__main
     ldr      r0,=vt1  ;load pointer vt1
     mov      r1,#8    ;counter
pk0  ldrsb    r2,[r0]  ;ler vt1(i)
     strb     r2,[r0,#8];salvar em vt2(i)
     add      r0,#1    ;inc.pointer
     subs     r1,#1    ;
     bne      pk0      ;prox.elem.
     b        __main   ;
     end
```



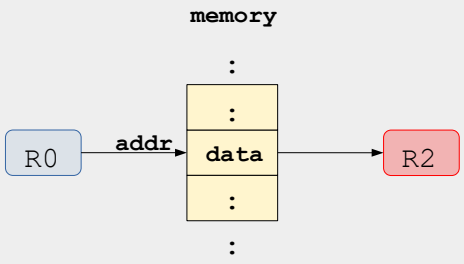
Ex. 1d_1 – Refaça o (Ex.1d) carregando o registro r0 com o endereço de vt2. Acrescente, entre vt1 e vt2, um vetor de 4 bytes. (O usuário calcula o offset entre os vetores!).

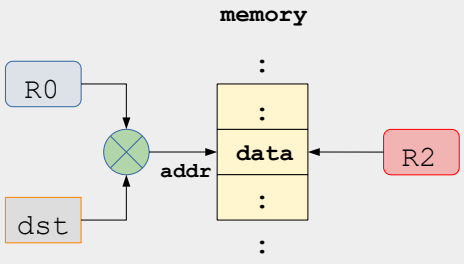
```
                __main:
0x0800024C 4805      LDR      r0,[pc,#20]   ;@0x08000264
0x0800024E F04F0108  MOV      r1,#0x08
0x08000252 F9902000  LDRSB    r2,[r0,#0x00]
0x08000256 7202      STRB     r2,[r0,#0x08]
0x08000258 F1000001  ADD      r0,r0,#0x01
0x0800025C 3901      SUBS     r1,r1,#0x01
0x0800025E D1F8      BNE      0x08000252
0x08000260 E7F4      B        0x0800024C __main
```

T5.8.1c) MODO END. INDEX. COM OFFSET IMMEDIATE (fixo)

Ex. 1e – O vetor vt1 tem 8 bytes. Faça um programa para copiar os bytes de vt1 no vetor vt2. O programa é cíclico. Faça as designações e alocações necessárias. O cálculo do offset entre os vetores deve ser dinâmico (pelo software).

```
;---Ex. 1e - End.Index. Offset imediato
     export   __main
;=== dir. equate
dst  equ      vt2-vt1  ;(final-inicial)
;=== diretiva area - dados (sram)
     area     dds1, data, readwrite
vt1  space    8
vtw  dcb      &a7,&3b,&5e
vtz  dcw      &39af,&b287
vt2  space    8
;=== diretiva area - prog. (flash)
     area     m_prog, code, readonly
__main
     ldr      r0,=vt1  ;load pointer vt1
     mov      r1,#8    ;counter
pk0  ldrsb    r2,[r0]  ;ler vt1(i)
;    strb     r2,[r0,#?];salvar em vt2(i)
     strb     r2,[r0,#dst];salvar em vt2(i)
     add      r0,#1    ;inc.pointer
     subs     r1,#1    ;
     bne      pk0      ;
     b        __main   ;
     end
```

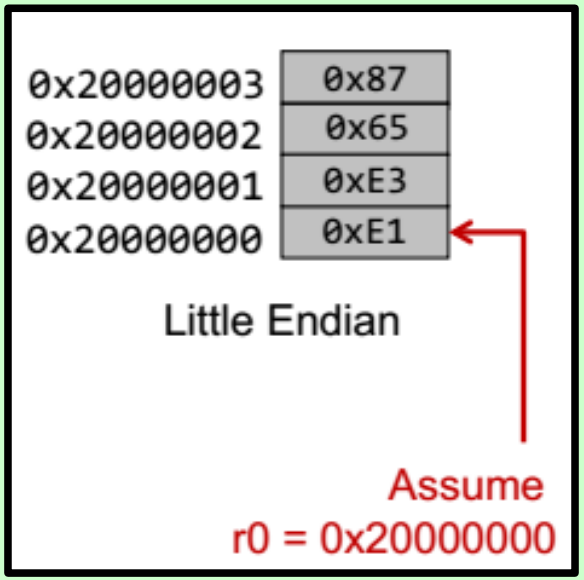




Ex.1e_1 – Refaça o (Ex.1e) iniciando a cópia dos bytes de vt1 a partir do último byte de vt1.

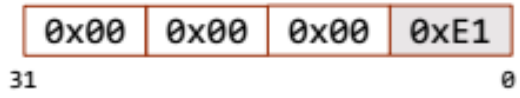
```
                    __main:
0x0800024C 4805      LDR      r0,[pc,#20]   ; @0x08000264
0x0800024E F04F0108  MOV      r1,#0x08
0x08000252 F9902000  LDRSB    r2,[r0,#0x00]
0x08000256 7402      STRB     r2,[r0,#0x10]
0x08000258 F1000001  ADD      r0,r0,#0x01
0x0800025C 3901      SUBS     r1,r1,#0x01
0x0800025E D1F8      BNE      0x08000252
0x08000260 E7F4      B        0x0800024C __main
```

T3.0.2 Load and Store instructions: INDEXADO (REGISTRO (pointer))



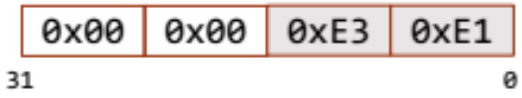
Load a Byte

```
LDRB r1, [r0]
```



Load a Halfword

```
LDRH r1, [r0]
```



Load a Word

```
LDR r1, [r0]
```