

# PCS3432 - Laboratório de Processadores

---

## Planejamento - E4

Bruno Mariz - 11261826

---

B.1) Descreva o conteúdo do registrador R13 ou sp depois que as seguintes instruções forem executadas, assumindo que a memória contenha os valores mostrados abaixo. O registrador R0 contém 0x24, e o sistema de memória é little-endian (o menos significativo é colocado no endereço mais baixo).

Endereço	Conteúdo
0x24	0x06
0x25	0xFC
0x26	0x03
0x27	0xFF

```
LDRSB sp, [r0] @ -> sp = 0x00000006
LDRSH sp, [r0] @ -> sp = 0xFFFFFC06
LDR sp, [r0] @ -> sp = 0xFF03FC06
LDRB sp, [r0] @ -> sp = 0x00000006
```

B.2) Indique se as seguintes instruções usam o modo pré ou pós indexado de endereçamento:

STR r6, [r4,#4] → pre-indexado

LDR r3, [r12], #6 → pos-indexado

LDRB r4, [r3,r2]! → pre-indexado (com writeback)

LDRSH r12, [r6] → pos-indexado

B.3) Calcule o endereço efetivo das seguintes instruções se o registrador r3 = 0x4000 e o registrador r4 = 0x20

STRB r9, [r3,r4] → endereço: 0x4020 LDRB r8,[r3,r4,LSL #3] → endereço: 0x4100 LDR r7, [r3], r4 → endereço: 0x4000 STRB r6, [r3], r4, ASR #2 → endereço: 0x4000

B.4) O que há de errado na seguinte instrução? Veja "incorrect example" em: <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0068b/Chdbifed.html>

```
LDRSB r1, [r6], r3, LSL #4
```

### 4.1.3 C - escreva código

Escreva o código em Assembly que faça:

```
for (i=0; i<8; i++) {  
    a[i] = b[7-i];  
}
```

instalar o gnuarm na máquina de vocês e testar. Procurem usar as seguintes instruções em seu código:

- LDR ou ADR (isto é: declarem os dados na memória e leiam de lá; por exemplo, onde começa o array b e o array a).
- BGE (usem instruções que façam o desvio condicional, não necessariamente BGE).
- RSB (para o 7-i)
- STR (isto é: armazene de fato o dado na memória).

Vejam: <https://stackoverflow.com/questions/42503417/arm-assembly-arrays/53391341#53391> ou informações aqui na aula de hoje sobre como alocar uma array na memória.

Código utilizado no exercício:

```
.data  
array_a: .word 0,0,0,0,0,0,0,0  
array_b: .word 0x10,0x20,0x30,0x40,0x50,0x60,0x70,0x80  
  
.text  
.globl main  
  
main:  
    MOV r4, #0 @ dados temporarios  
    MOV r5, #0 @ i  
    MOV r6, #0 @ 7-i  
    LDR r0, =array_b @ carrega endereço do array b  
    LDR r1, =array_a @ carrega endereço do array a  
    BL transfer @ transfere valores de b para o segundo enunciado  
  
    MOV r0, #0x18  
    LDR r1, =0x20026  
    SWI 0x0  
  
transfer:  
    RSB r6, r5, #7 @ r6 = 7 - i  
    LDR r4, [r0, r5, LSL #2] @ r4 = b[i]
```

```

STR r4, [r1, r6, LSL #2] @ a[7 - i] = r4
ADD r5, r5, #1
CMP r5, #8
BLT transfer
MOV pc, lr

```

Primeiramente, foi escrito um código que cria os arrays, e carrega seus endereços nos registradores r0 e r1 utilizando a instrução ADR, seguindo o link indicado no enunciado, com a modificação de retirar a seção .data, que impediu o uso da instrução ADR (seria necessário utilizar LDR) e o conteúdo dos arrays foi observado conforme na imagem a seguir:

Register group: general

r0	0x21044	135236	r1	0x21024	135204
r2	0xfffec2c	-70612	r3	0x103c8	66504
r4	0x0	0	r5	0x0	0
r6	0x0	0	r7	0x0	0
r8	0x0	0	r9	0x0	0
r10	0xff7ee000	-8462336	r11	0x0	0
r12	0xfffeeb48	-70840	sp	0xfffeead0	0xfffeead0

4-1-3-C.s

```

B+ 10      MOV r4, #0 @ dados temporarios
11         MOV r5, #0 @ i
12         MOV r6, #0 @ 7-i
13         LDR r0, =array_b @ carrega endereco do array b
14         LDR r1, =array_a @ carrega endereco do array a
>15        BL  transfer @ transfere valores de b para a segundo enunciado
16
17         MOV r0, #0x18

```

remote Thread 1.79570 In: main L15 PC: 0x103dc

```

(gdb) s
(gdb) s
(gdb) x/x $r0
0x21044: 0x00000010
(gdb) x/x $r1
0x21024: 0x00000000
(gdb) x/x $r0+4*7
0x21060: 0x00000080
(gdb)

```

A seguir, foi desenvolvida uma função que transfere o conteúdo de um array para o outro conforme o enunciado. Após a primeira iteração, é possível observar o array a sendo preenchido:

Register group: general

r0	0x21044	135236	r1	0x21024	135204
r2	0xfffec2c	-70612	r3	0x103c8	66504
r4	0x10	16	r5	0x0	0
r6	0x7	7	r7	0x0	0
r8	0x0	0	r9	0x0	0
r10	0xff7ee000	-8462336	r11	0x0	0
r12	0xfffeeb48	-70840	sp	0xfffeead0	0xfffeead0

4-1-3-C.s

```

22         RSB r6, r5, #7 @ r6 = 7 - i
23         LDR r4, [r0, r5, LSL #2] @ r4 = b[i]
24         STR r4, [r1, r6, LSL #2] @ a[7 - i] = r4
>25        ADD r5, r5, #1
26         CMP r5, #8
27         BLT transfer
28         MOV pc, lr

```

remote Thread 1.79570 In: transfer L25 PC: 0x103f8

```

(gdb) x/x $r1
0x21024: 0x00000000
(gdb) x/x $r1+4*7
0x21040: 0x00000010

```

Ao fim das iterações, o conteúdo do array a foi observado conforme na imagem a seguir:

```
Register group: general
r0      0x21044      135236      r1      0x21024      135204
r2      0xfffeec2c   -70612      r3      0x103c8      66504
r4      0x80         128         r5      0x8          8
r6      0x0          0           r7      0x0          0
r8      0x0          0           r9      0x0          0
r10     0xff7ee000   -8462336    r11     0x0          0
r12     0xfffeeb48   -70840      sp      0xfffeead0    0xfffeead0
lr      0x103e0      66528      pc      0x103fc     0x103fc <transfer+16>

4-1-3-C.s
22      RSB r6, r5, #7 @ r6 = 7 - i
23      LDR r4, [r0, r5, LSL #2] @ r4 = b[i]
24      STR r4, [r1, r6, LSL #2] @ a[7 - i] = r4
25      ADD r5, r5, #1
>26     CMP r5, #8
27      BLT transfer
28      MOV pc, lr

remote Thread 1.79570 In: transfer L26 PC: 0x103fc
(gdb) x/x $r1
0x21024: 0x00000080
0x21028: 0x00000070
0x2102c: 0x00000060
0x21030: 0x00000050
0x21034: 0x00000040
0x21038: 0x00000030
0x2103c: 0x00000020
0x21040: 0x00000010
```

Foi então também observado o array b para comparação:

```
Register group: general
r0      0x21044      135236      r1      0x21024      135204
r2      0xfffeec2c   -70612      r3      0x103c8      66504
r4      0x80         128         r5      0x8          8
r6      0x0          0           r7      0x0          0
r8      0x0          0           r9      0x0          0
r10     0xff7ee000   -8462336    r11     0x0          0
r12     0xfffeeb48   -70840      sp      0xfffeead0    0xfffeead0
lr      0x103e0      66528      pc      0x103fc     0x103fc <transfer+16>

4-1-3-C.s
22      RSB r6, r5, #7 @ r6 = 7 - i
23      LDR r4, [r0, r5, LSL #2] @ r4 = b[i]
24      STR r4, [r1, r6, LSL #2] @ a[7 - i] = r4
25      ADD r5, r5, #1
>26     CMP r5, #8
27      BLT transfer
28      MOV pc, lr

remote Thread 1.79570 In: transfer L26 PC: 0x103fc
(gdb) x/x $r0
0x21044: 0x00000010
0x21048: 0x00000020
0x2104c: 0x00000030
0x21050: 0x00000040
0x21054: 0x00000050
0x21058: 0x00000060
0x2105c: 0x00000070
0x21060: 0x00000080
```