Exercícios sobre instruções adicionais AARCH64 (ARMV8)

① a)

```
.text
.global f3e1a
.type f3e1a, %function

f3e1a:      mov X2,#0
            mov X4,#0
ciclo:      ldr X3,[X0]
            add X4, X4, X3
            add X2, X2, #1
            cmp X2, X1
            b.eq fim
            add X0, X0,#4
            b ciclo
fim:        mov X0, X4
ret
```

b)

(tudo igual)

exceto:

⟷ add X0, X0, #8

② a)

```
.text
.global f3e2a
.type f3e2a, %function

f3e2a:      stp X29, X30,[SP,-16]!
mov W2,#0 ← mov X29, SP
            cbz W1, fim
            ldr W2,[X0]
            sub W1, W1, #1

ciclo:      cbz W1, fim
            add X0, X0, #4
            ldr W3, [X0]
            sub W1, W1, #1
            cmp W2, W3
            csel W2, W2, W3, ge
            b ciclo.

fim:        mov W0, W2
            ldp X29, X30,[SP], 16

ret
```

b) (tudo igual ao a) mas W passam a X e no ciclo:

```
add X0, X0, #8
csel X2, X2, X3, le
```

c) (tudo igual ao a, mas fica:

```
ldrsh W3, [X0], 2
```

(não é preciso add).

d)

```
stp X29, X30,[SP,-16]!
mov X29, SP
mov W4, W1
mov X5, X30
bl f3e1b
mov X30, X5
sdiv W0, W0, W4
ldp X29, X30, [SP],16
ret
```

e)
```
        .text
        .global f3e2e
        .type f3e2e, %function

f3e2e:  stp X29, X30, [SP, -16]!
        mov X29, SP
        mov X4, XZR
ciclo:  cbz W1, fim
        ldr X5, [X0], 8
        sub W1, W1, 1
        cmp X5, X2
        b.lt ciclo
        cmp X5, X3
        cinc X4, X4, le
        b ciclo
fim:    mov X0, X4
        ldp X29, X30, [SP], 16
        ret
```

③ a)
```
        .text
        .global f3e3a
        .type f3e3a, %function

f3e3a:  stp X29, X30, [SP, -16]!
        mov X29, SP
        mov X1, XZR
ciclo:  ldrb W2, [X0], 1
        cbz W2, fim
        add X1, X1, 1
        b ciclo
fim:    mov X0, X1
        ldp X29, X30, [SP], 16
        ret
```

b)
```
        .text
        .global f3e3b
        .type f3e3b, %function

f3e3b:  stp X29, X30, [SP, -16]!
        mov X29, SP
        mov X2, XZR
ciclo:  ldrb W3, [X0], 1
        cbz W3, fim
        cmp W3, W1
        cinc X2, X2, eq
        b ciclo
fim:    mov X0, X2
        ldp X29, X30, [SP], 16
        ret
```

c)
```
        .text
        .global f3e3c
        .type f3e3c, %function

f3e3c:  stp X29, X30, [SP, -16]!
        mov X29, SP
        mov X2, XZR
ciclo:  ldrb W1, [X0], 1
        cbz W1, fim
        cmp W1, 'A'
        cinc X2, X2, eq
        b.eq ciclo
        cmp W1, 'a'
        cinc X2, X2, eq
        b.eq ciclo
        cmp W1, 'B'
        cinc X2, X2, eq
        b.eq ciclo
        (...)
fim     mov X0, X2
        ldp X29, X30, [SP], 16
        ret
```

(para as poupar todos.) ← (...)

d)
```
        .text
        .global f3e3d
        .type f3e3d, %function

f3e3d:  stp X29, X30, [SP, -16]!
        mov X29, SP
        mov X1, XZR
ciclo:  ldrb W2, [X0], 1
        cbz W2, fim
        cmp W2, 'A'
        b.lt ciclo
        cmp W2, 'Z'
        cinc X1, X1, le
        b ciclo
fim:    mov X0, X1
        ldp X29, X30, [SP], 16
        ret
```

a)
```
mov X5,X0
bl f3e3a
mov X3,#1
sub X0,X0,#1
add X2,X5,X0
ciclo: ldrb X6,[X5],#1
ldrb X7,[X0],#-1
cmp X5,X2
B.eq fim
sub X8,X0,X5 (incadrat)
cmp X8,#1
b.eq fim
Bic W5,W6,0x20
Bic W7,W7,0x20
cmp W6,W7
```

b).
```
.text
.global f3,e3b
.type f3e3b, % function
f3e3b: stp X29,X30,[SP,-16]!
mov X29,SP
mov X1, 0
ciclo: ldrb W2,[X0],1
cbz W2, fim
cmp W2,'.'
cinc, X1,X1, eq
b ciclo
fim add X1,X1,1
mov X0,X1
ret.
```

④ a).
```
.text
.global f3e4a
.type f3e4a, % function
f3e4a: stp X29,X30,[SP,-16]!
mov X29,SP
ciclo: cbz X2, fim
ldrb W3,[X0],1
str X3,[X1],8
sub X2,X2,1
b ciclo
fim: ret
```
X0 → nr. elemente

b).
```
.text
.global f3e4b
.type f3e4b, % function
f3e4b: stp X29 X30,[SP,-16]!
mov X29,SP
ciclo: cbz X2, fim
ldrsw X3,[X0],4
str X3,[X1],8
sub X2,X2,1
b ciclo
fim: ret
```

⑤ a).
```
.text
.global POS1msb
.type POS1msb, % function
POS1msb: stp X29,X30,[SP,-16]!
mov X29,SP
clz X1,X0
mov X2,63
sub X0,X2,X1
ldp X29,X30,[SP],16
ret
```

b).
```
.text
.global PAL8
.type PAL8, % function
PAL8: stp X29,X30,[SP,-16]!
mov X29,SP
ldr X1,[X0]
rev X0,X1
cmp X0,X1
cset W0,eq
ldp X29,X30,[SP],16
ret
```

e).
```
.text
.global NCAP
.type NCAP, % function
NCAP: stp X29,X30,[SP,-16]!
mov X29,SP
rbit W1,W0
cmp W1,W0
cset W0,eq
ldp X29,X30,[SP],16
ret
```

⑥ a) bfi X10, X12, #8, #56
ubfx X10, XZR, #0, #7

b) ubfx X10, X12, #8, #56

c) ubfx W13, W14, #0, #3
ubfx W14, W14, #3, #61
bfi W14, W13, #61, #3

⑦ a).
```
.text
.global f3e7a
.type f3e7a, %function
f3e7a:  cbz X2, fim
        ldr X3, [X0], #4
        ldr X4, [X1], #4
        add X5, X3, X4
        str X5, [X6], #4
        sub X2, X2, #1
        b f3e7a
fim:    ret
```

b).
```
.text
.global f3e7b
.type f3e7b, %function
f3e7b:  cbz X2, fim
        ldr X3, [X0], #4
        ldr X4, [X1], #4
        adds X5, X3, X4
        b.vs overflow
continua: str X5, [X6], #4
        sub X2, X2, #1
        b f3e7b
overflow: adds X5, X3, X4
        b.lt negativo
        mov X10, #-1
        lsr X10, X10, #1
        mov X6, X10
        b continua
negativo: mov X9, #1
        ror X9, X9, #1
        mov X6, X9
        b continua
fim:    ret
```

c).
```
.text
.global f3e7c
.type f3e7c, %function
f3e7c:  cbz X2, fim
        ldr X3, [X0], #4
        mul X4, X1, X3
        str X4, [X8, #-4]
        sub X2, X2, #1
        b f3e7c
fim:    ret
```

d).
```
.text
.global f3e7d
.type f3e7d, %function
f3e7d:  mov X3, #0
ciclo:  cbz X2, fim
        ldr X4, [X0], #4
        ldr X5, [X1], #4
        sub X2, X2, #1
        smaddl X3, X4, X5, X3
        b ciclo
fim:    mov X0, X3
```

e).
```
.text
.global f3e7e
.type f3e7e, %function
f3e7e:  mov X3, #0
ciclo:  cbz X0, fim
        ldr W4, [X0], #4
        ldr W5, [X1], #4
        sub X2, X2, #1
        smadd X6, W4, W5
        adds X3, X3, X6
        b.vs overflow
        b ciclo
overflow: csel X3, X_min (se d), X_max (se d), PL
fim:    mov X0, X3
        ret
```

⑧ a) W0 = 0x66666666 W1 = 0xF000000F

0110 0110 0110 0110 0110 0110 0110 0110 > and
1111 0000 0000 0000 0000 0000 0000 1111
W0 = 0110 0000 0000 0000 0000 0000 0000 0110
W0 = 0x60000006

0x90000009

EOR: W0 = 1001 0000 0000 0000 0000 0000 0000 1001
ORR: 1111 0110 0110 0110 0110 0110 0110 1111
W0 = 0xF666666F

b) W0 = 0x0000BEEF

0000 0000 0000 0000 1011 1110 1110 1111 > add
0000 0000 0000 0000 1000 0000 0000 0000
0000 0000 0000 0001 0011 1110 1110 1111
W1 = 0x0001 3EEF

0000 0000 0000 0000 0011 1110 1110 1110 > SBC
0000 0000 0000 0001 0000 0000 0000 0001
1 > do carry
0000 0000 0000 0001 0000 0000 0000 0000 ← ADC
0000 0000 0000 0010 0000 0000 0000 0000 (carry bit)
R: W0 = 0x00020000

⑨ a) W1 = 0xABCD0000 W0 = 0x12345678

UBFX → W2 = 0x00000012
BFI → W1 = 0xAB120000

A S B:
0x00007856

b) REV → W1 = 0x78563412
AND → W1 = 0x00003012
REV → W1 = 0x12300000
SUB → W1 = 0x00045678

W0 − W1:
0001 0010 0011 0100 0101 0110 0111 1000
− 0001 0010 0011 0000 0000 0000 0000
0 0 0 4 5678

c) EON → X1 = 0xFFFFFFFF
SXTB → W0 = 0x00000078
lsl → W0 = 0x00000780
ADD →

#4 → (lsl)

R: X1 = 0x0000077F

W0+W1: 1111 1111 1111 1111
0000 0111 1000 0
0000 0111 0111 F

(10) a) X0 apresenta o módulo (valor absoluto) do valor inicial de X0.

b) X0 apresenta o maior valor de entre os valores guardados nos registos X1, X2 e X3.

(11)
```
.text
.global B3e11
.type B3e11, %function

B3e11 stp X29, X30, [SP, #-16]!
      mov X29, SP
      clz X0, X0
      sub X0, X0, #64
      neg X0, X0
      csinc X0, X0, XZR, ne
      ldp X29, X30, [SP], #16
      ret
```