**Esercizio 1**
Considerare il seguente frammento di codice.

a) Indicare il contenuto (in base 10) dei registri x20 e x21 dopo l'esecuzione di ogni singola istruzione:

```
mov x20, #2
lsr x21, x20, #1
sub x20, x20, x21
add x20, x20, x21, lsl #2
```

b) Eseguire il programma "esercizio1.s" per verificare se i valori indicati coincidono con quelli ottenuti. Si noti che il programma fa uso delle due seguenti macro per stampare tramite ***printf***, il contenuto dei due registri dopo ogni singola istruzione.

```
.section .rodata
fmt: .asciz "%d\n"

.macro print_x20
    adr x0, fmt
    mov x1, x20
    bl printf
.endm

.macro print_x21
    adr x0, fmt
    mov x1, x21
    bl printf
.endm
```

**Soluzione**

| | ***Contenuto x20*** | ***Contenuto x21*** |
|---|---|---|
| `mov x20, #2` | 2 | Imprevedibile, non ancora impostato |
| `lsr x21, x20, #1` | 2 | x20>>1=1 |
| `sub x20, x20, x21` | x20-x21=1 | 1 |
| `add x20, x20, x21, lsl #2` | x20+x21<<2=x20+4=5 | 1 |

**Esercizio 2**
Considerare il seguente frammento di codice.

a) Indicare i numeri contenuti nei registri x20 e x21. in base 10 e in base 2 (in complemento a 2), dopo l'esecuzione di ogni singola istruzione:

```
mov x20, #0b00000001
mov x21, #0b00000011
and x21, x21, x20
bic x20, x20, x21
orr x20, x20, x21, lsl #1
orn x20, x20, x21
eor x20, x20, x21, lsl #2
eon x20, x20, x21
```

b) Eseguire il programma “esercizio2.s” per verificare se i valori indicati coincidono con quelli ottenuti. Si noti che il programma fa uso due stesse macro riportate per il punto b) dell’esercizio 1 per stampare tramite ***printf***, il contenuto dei due registri dopo ogni singola istruzione.

**Soluzione**

| | ***Contenuto x20*** | ***Contenuto x21*** |
|---|---|---|
| `mov x20, #1` | $1_{10}$ = $00000001_2$ | Imprevedibile, non ancora impostato |
| `mov x21, #1` | $1_{10}$ = $00000001_2$ | $3_{10}$ = $00000011_2$ |
| `and x21, x21, x20` | $1_{10}$ = $00000001_2$ | $1_{10}$ = $00000001_2$ |
| `bic x20, x20, x21` | $0_{10}$ = $00000000_2$ | $1_{10}$ = $00000001_2$ |
| `orr x20, x20, x21, lsl #1` | $2_{10}$ = $00000010_2$ | $1_{10}$ = $00000001_2$ |
| `orn x20, x20, x21` | $-2_{10}$ = $11111110_2$ | $1_{10}$ = $00000001_2$ |
| `eor x20, x20, x21, lsl #2` | $-6_{10}$ = $11111010_2$ | $1_{10}$ = $00000001_2$ |
| `eon x20, x20, x21` | $4_{10}$ = $00000100_2$ | $1_{10}$ = $00000001_2$ |

**Esercizio 3**

Supponendo che il registro x0 contenga il byte 01001011 che corrisponde a $75_{10}$ e il registro x1 contenga 00001111 che corrisponde a $15_{10}$, indicare il contenuto del registro x2 in base 10 dopo ciascuna delle seguenti istruzioni. Si indichi inoltre come vengono impostati i flag NZCV di PSTATE.

a) `adds x2, x1, x0`
b) `subs x2, x1, x0`

Al punto b) si consiglia di determinare prima $-75_{10}$ rappresentato in complemento a due e poi di calcolare il risultato come $15_{10}$ + $(-75)_{10}$. Eseguire quindi il programma “esercizio3.s” per verificare se i valori indicati coincidono con quelli ottenuti. Si noti che il programma usa la macro ***print*** per stampare il contenuto dei flag NZCV di PSTATE.

**Soluzione**

Dopo istruzione a): N = 0, Z = 0, C = 0, V = 0, x2 contiene 90

Dopo istruzione b): N = 1, Z = 0, C = 1, V = 0, x2 contiene -60

**Esercizio 4**

a) Partendo dal programma riportato nel file “esercizio4.s”, scrivere un programma che calcoli l’or esclusivo (eor) tra tutti gli elementi dell’array A. Sia N il risultato, il programma deve quindi impostare al valore N l’elemento in posizione 0 di A.
**Esempio**. Essendo A = [0,1,2,3], al termine del programma l’array A rimane tale.
b) Cosa cambierebbe se al posto dell’or esclusivo si usasse quello inclusivo?

**Soluzione**

a) Vedi esercizio4.s in “Lab05Soluzioni.zip”
b) Semplicemente usiamo orr al posto eor. Nell’esempio si otterrebbe A = [3,1,2,3].

**Esercizio 5**

a) Partendo dal frammento di codice riportato in "esercizio5.s", scrivere un programma che letto da input un numero X, incrementi ogni valore nell'array A di X unità. Si noti che la macro `scan` serve proprio a leggere un numero da input. Per leggere il numero X è sufficiente invocare la macro come segue:
```
scan fmt_scan, x
```
**Esempio**. Se X fosse 3, l'array A diventerebbe: [10, 3, 6, 4].

b) Estendere il programma ottenuto al punto precedente, per fare in modo che, letto un secondo numero da input Y, il programma decrementi ogni valore nell'array A di Y unità. Per leggere il numero Y utilizziamo sempre la macro `scan` come segue:
```
scan fmt_scan, y
```
**Esempio**. Dopo il punto a) l'array A è diventato [10, 3, 6, 4], quindi ad esempio, se Y fosse 3, l'array A ritornerebbe: [7, 0, 3, 1].

c) Estendere ulteriormente il programma ottenuto al punto precedente, per fare in modo che, ogni elemento di A sia moltiplicato per 2: si utilizzi a tal fine un appropriato shift.
**Esempio**. Dopo i punti a) e b) l'array A è diventato[7, 0, 3, 1], quindi dopo il punto c) diventerebbe: [14, 0, 6, 2].

**Soluzione**

Vedi esercizio5.s in "Lab05Soluzioni.zip"

**Esercizio 6**

Partendo dal frammento di codice riportato in "esercizio6.s" in cui sono presenti due array di interi A e B, scrivere un programma che modifichi l'array A facendo in modo che:

a) Il primo elemento nell'array A sia il risultato della somma tra il primo elemento nell'array A e il primo elemento nell'array B, ovvero:

$$A[0] = A[0] + B[0]$$

b) Il secondo elemento nell'array A sia il risultato della sottrazione tra il secondo elemento nell'array A e il secondo elemento nell'array B, ovvero:

$$A[1] = A[1] - B[1]$$

**Soluzione**

Vedi esercizio6.s in "Lab05Soluzioni.zip"

**Esercizio 7**

Partendo dal frammento di codice riportato in "esercizio7.s" in cui sono presenti due array di byte A e B scrivere un programma che modifichi l'array A facendo in modo che:

a) Il primo byte nell'array A sia il risultato dell'AND tra il primo byte nell'array A e il primo byte nell'array B, ovvero:

A[0] = A[0] and B[0]

b) Il secondo byte nell'array A sia il risultato dell'OR (inclusivo) tra il secondo byte nell'array A e il secondo byte nell'array B, ovvero:

A[1] = A[1] or B[1]

(**Suggerimento**: dal momento che il programma manipola dei byte, occorre utilizzare opportunamente i modificatori b/sb nelle istruzioni ldr/str ;)

**Soluzione**

Vedi esercizio7.s in "Lab05Soluzioni.zip"