Esercizio 1
Considerare il seguente frammento di codice.

a) Indicare il contenuto dei registri x20, w21 e x22 dopo l'esecuzione di ogni singola istruzione:

```
mov x20, #5
ror w21, w20, #2
umnegl x22, w20, w21
neg x20, x20
sdiv x22, x22, x20
```

b) Eseguire il programma "esercizio1.s" riportato in "Lab06Esercizi.zip" per verificare se i valori indicati coincidono con quelli ottenuti.

Esercizio 2
Considerare il seguente frammento di codice.

a) Indicare il contenuto del registro w1 dopo l'esecuzione delle seguenti istruzioni `csel` e `cset`:

```
mov w20, #20
mov w21, #10
cmp w20, w21
csel w1, w20, w21, lt
sub w20, w20, w21
cmp w20, w21
cset w1, eq
```

b) Eseguire il programma "esercizio2.s" riportato in "Lab06Esercizi.zip" per verificare se i valori indicati coincidono con quelli ottenuti.

Esercizio 3

Partendo dal frammento di codice riportato in "esercizio3.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 terzo elemento nell'array A sia il numero maggiore tra il terzo elemento nell'array A e il terzo elemento nell'array B (utilizzare a tal fine **csel**):

```
if B[2] > A[2]:
  A[2]=B[2]
```

b) Il quarto elemento nell'array A sia impostato ad 1 se il quarto elemento nell'array A è maggiore del quarto elemento nell'array B, a 0 altrimenti (utilizzare a tal fine **cmp** e **cset**):

```
if A[3] > B[3]:
    A[3]=1
else:
    A[3]=0
```

Esercizio 4

Partendo dal frammento di codice riportato in "esercizio4.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 terzo byte nell'array A sia uguale al terzo byte nell'array B incrementato di 1 se il terzo byte nell'array A è maggiore del terzo byte nell'array B (utilizzare a tal fine **cmp** e **csinc**):

```
if A[2] > B[2]:
    A[2]=B[2]+1
```

b) Il quarto byte nell'array A sia impostato al valore assoluto del risultato dell'AND tra il quarto byte nell'array A ed il quarto byte nell'array B. Utilizzare l'operazione di confronto **tst** e l'operazione condizionale **csneg** per accertarsi che il risultato sia in valore assoluto.

$$A[3] = | A[3] \text{ and } B[3] |$$

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

Per capire il funzionamento di **tst**, a titolo esemplificativo, considerare le seguenti istruzioni:

mov w1, 0b11111111
tst w1, 0b10000000
Nel registro w1 copiamo la sequenza di bit **11111111** (tramite l'istruzione **mov**). Successivamente, l'istruzione **tst** effettua l'AND bit a bit tra **11111111** e **10000000** e imposta in base al risultato i flag N e Z. In questo caso il risultato dell'AND sarebbe:

1 1 1 1 1 1 1 1 AND
1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0

Il risultato dunque è **10000000** che viene interpretato in complemento a due e dunque corrisponde a -128. Pertanto il flag Z sarebbe impostato ad 1.
Ulteriori informazioni su tst sono disponibili sulla documentazione ufficiale di ARM 64:
https://developer.arm.com/documentation/dui0068/b/ARM-Instruction-Reference/ARM-general-data-processing-instructions/TST-and-TEQ

Esercizio 5

Partendo dallo scheletro di programma riportato nel file "esercizio5.s" in "Lab06Esercizi.zip", che nella sezione .data contiene l'array A così definito:

```
.data
A: .word 13, 4, 5, 4, 1, 0, -3, 10
```

realizzare quattro programmi differenti che determinino e stampino:

1. Il minimo di A,

2. Il massimo di A,
3. La somma degli elementi in A,
4. La media degli elementi in A (utilizzando la divisione intera),

Per stampare utilizzare la macro **print** già presente nello scheletro di programma fornito. Per usare correttamente la macro è necessario che il valore da stampare, ovvero minimo, massimo, somma o media, sia contenuto nel registro w1 prima che la macro sia invocata.

Per il quarto programma e dunque per il calcolo della media, è sufficiente determinare il risultato della divisione tra la somma degli elementi in A e la sua dimensione.

Per la dimensione si può utilizzare la costanze A_size già definita nel programma come segue:

```
.equ A_size, (. - A) / 4
```

Si dovrà ottenere che il minimo è -3, il massimo è 13, la somma è 34 e la media è 4.