1. Calcola la rappresentazione di -43_{10} in complemento a 2 su 8 bits.

Risposta: $1101\ 0101_2$

2. Effettua la seguente sottrazione in binario:
$0111\ 1111\ 0110\ 1010\ 1111_2 - 0101\ 1101\ 1110\ 1101\ 0111_2$

Risposta: $0010\ 0001\ 0111\ 1101\ 1000_2$

3. Rappresenta il numero decimale 0.1 nella codifica dello standard IEEE754

Risposta: $0011\ 1101\ 1100\ 1100\ 1100\ 1100\ 1100\ 1101_2$

4. Il registro `x8` contiene il valore `x8` = $1111\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111_2$. Per tutti i registri, i 32 bits più significativi sono omessi perché non influenzati dalle seguenti operazioni. Nota: il suffisso "w" indica operazioni a 32-bits. Quale sarà il valore contenuto da `x8` dopo l'esecuzione delle seguenti istruzioni in assembly RISC-V?

```
slliw      x5, x8, 4
srliw      x8, x8, 8
or         x8, x8, x5
```

Risposta: `x8` = $0000\ 0000\ 1111\ 0000\ 0000\ 0000\ 1111\ 0000_2$

5. Quale delle seguenti affermazioni è CORRETTA riguardo le CPU CISC?

Riposta: CISC è un acronimo per Complex Instruction Set Computer

6. La funzione `sumV` prende in input l'indirizzo di tre vettori di interi `u`, `v`, `z`, e la `dimensione` degli array, l'obiettivo è di effettuare delle operazioni sugli elementi i-esimi di `u` , `z` e `v`, e di salvare il risultato nell'i-esimo elemento di `u`. Quale delle seguenti implementazioni di questa funzione in assembly RISC-V è corretta?

```
void sumV(int * u, int * v, int* z, unsigned int size){
      for (unsigned int i = 0; i < size; i++) {
            *(u+i) = *(v+i) - *(u+i) + *(z+i) + 12;
      }
      return;
}
```

Answer:

```
sumV:
     beqz    a3, END       # return if size is 0
     slli    a3, a3, 32    # shift a sinistra di 32 bits
     srli    a3, a3, 16    # shift a destra di 16 bits
     srli    a3, a3, 16    # shift a destra di 16 bits
FOR:
     lw     a4, 0(a0)      # carica il valore di u[i]
     lw     a5, 0(a1)      # carica il valore di v[i]
     lw     a6, 0(a2)      # carica il valore di z[i]
     sub    a4, a5, a4     # esegui v[i] - u[i]
     add    a5, a4, a6     # esegui (v[i] - u[i]) + z[i]
     addi   a6, a5, 12     # esegui ((v[i] - u[i]) + z[i]) + 12
     sw     a6, 0(a0)      # salva il risultato in u[i]
     addi    a0, a0, 4     #
     addi    a1, a1, 4     #
     addi    a2, a2, 4     #
     addi    a3, a3, -1    # incrementa i
     bnez    a3, FOR       # controlla se (i < size)
END:
     ret                   # return
```