Es1

Scrivere una procedura chiamata $lessthan(array, x, y)$ che restituisca 1 se il valore in $array[x]$ è minore del valore in $array[y]$. Altrimenti, la funzione deve restituire 0. Il valore di ritorno deve essere inserito nel registro $a0$.

Il seguente codice in C implementa $lessthan$ (convertilo in RISC-V):

```
// int (C) equivale a word (RISC-V)
void lessthan(int array[], int x, int y) {
    if (array[x] < array[y]) {
        return 1;
    } else {
        return 0;
    }
}
```

Attenzione:

- Incollare solo la funzione $lessthan$ (in RISC-V) nel campo sottostante
- Attenzione alle convenzioni di chiamata!
- Usare il seguente codice "main" per lo sviluppo e il debugging nel simulatore RARS

```
    .globl _start
    .data
        array: .word 1,5,3,7,2,6,4,8
        x:     .word 0
        y:     .word 1

    .text
    _start:
        # chiama lessthan
        la   a0, array
        la   a1, x
        lw   a1, 0(a1)
        la   a2, y
        lw   a2, 0(a2)
        jal  ra, lessthan

        #exit
        li   a7, 10
        ecall

    #************************************************
    # completare la funzione lessthan nel campo di sotto
```

Soluzione:

```
################################################################################
# Procedure lessthan(array, x, y)
# a0 -> address of array
# a1 -> index x
# a2 -> index y
# return 1 if array[x] < array[y], 0 otherwise
################################################################################
lessthan:
    # load array[x] in t0
    slli t1, a1, 2
    add  t1, a0, t1
    lw   t0, 0(t1)

    # load array[y] in t1
    slli t1, a2, 2
    add  t1, a0, t1
    lw   t1, 0(t1)

    # compare t0 and t1
    slt  a0, t0, t1

    # return
    jr   ra
```

Es2

Scrivere una procedura chiamata $smaller\_numbers(array, size)$ che restituisca il numero di elementi di un array di interi che sono minori del primo elemento dell'array.

Il valore di ritorno deve essere inserito nel registro $a0$.

Assumere che $size$ sia maggiore di 1.

Il seguente codice in C implementa $smaller\_numbers$ (convertilo in RISC-V):

```
// int (C) equivale a word (RISC-V)
void smaller_numbers(int array[], int size) {
    int i, count = 0;
    for (i = 1; i < size; i++) {
        count += lessthan(array, i, 0);
    }
    return count;
}
```

La funzione deve uttilizzare la funzione $lessthan$ sviluppata nell'esercizio precedente. Soluzioni che non utilizzano la funzione $lessthan$ verranno considerate invalide.

Attenzione:

- Incollare solo la funzione $smaller\_numbers$ (in RISC-V) nel campo sottostante
- Attenzione alle convenzioni di chiamata!
- Usare il seguente codice "main" per lo sviluppo e il debugging nel simulatore RARS

```
    .globl _start
    .data
        array: .word 8,5,3,7,2,6,4,8
        size:  .word 8

    .text
    _start:
        # chiama smaller_numbers
        la   a0, array
        la   a1, size
        lw   a1, 0(a1)
        jal  ra, smaller_numbers

        #exit
        li   a7, 10
        ecall

    #******************************************************
    # completare la funzione smaller_numbers nel campo di sotto
```

Soluzione:

```
###############################################################################
# Procedure smaller_numbers(array, size)
# a0 -> address of array
# a1 -> size
# return the number of elements in array that are smaller than the first element
###############################################################################
smaller_numbers:
    addi sp, sp, -40
    sd   ra, 0(sp)
    sd   a0, 8(sp)
    sd   s1, 16(sp)
    sd   s2, 24(sp)
    sd   s3, 32(sp)

    li   s1, 1
    mv   s2, a1
    li   s3, 0

loop:
    bge  s1, s2, return

    ld   a0, 8(sp)
    add  a2, zero, zero
    add  a1, zero, s1
    jal  ra, lessthan
    add  s3, s3, a0
    addi s1, s1, 1
    j    loop

return:
    mv   a0, s3

    ld   ra, 0(sp)
    ld   s1, 16(sp)
    ld   s2, 24(sp)
    ld   s3, 32(sp)
    addi sp, sp, 40
    jr   ra
```