Iniziato	martedì, 19 dicembre 2023, 14:49
Stato	Completato
Terminato	martedì, 19 dicembre 2023, 15:39
Tempo impiegato	50 min.
Valutazione	**0,5** su un massimo di 10,0 (**5**%)

Domanda **1**

Parzialmente corretta

Punteggio ottenuto 0,5 su 5,0

Scrivere una procedura chiamata `tetrahedral(n)` che restituisce 1 se il numero intero (double-word) n è un numero Numero Tetraedrico, 0 altrimenti.

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

Il seguente codice in C implementa `tetrahedral` (convertilo in RISC-V):

```
// long long (in C) equivale a double-word (in RISC-V)
long long tetrahedral(long long n) {
    long long i = 1;
    long long t = 0;

    while (t < n) {
        t = i * (i + 1) * (i + 2) / 6;
        i++;
    }
    return (t == n);
}
```

Attenzione:

- Incollare solo la funzione `tetrahedral` (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
        n:   .dword 56

    .text
    _start:
        # chiama tetrahedral
        la   a0, n
        ld   a0, 0(a0)
        jal  ra, tetrahedral

        # Risultato atteso su a0: 1

        #exit
        li   a7, 10
        ecall

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

Answer:

Reset answer

```
5       li t3,6
6   loop:
```

```
    bge t2,a0,uguale
    add t2,t2,t1
    addi t0,t1,1
    mul t2,t2,t0
    addi t0,t1,2
    mul t2,t2,t0
    div t2,t2,t3
    addi t1,t1,1
    add t2,t2,zero
    j loop
uguale:
    beq t2,a2,return1
return0:
    li a0,0
    ret
return1:
    li a0,1
    ret

```

Your code failed one or more hidden tests.
Your code must pass all tests to earn any marks. Try again.

Question author's solution (Asm):

```
################################################################################
# Procedure tetrahedral(n)
# a0 -> N
# return 1 if N is tetrahedral, 0 otherwise
################################################################################
tetrahedral:
    li  t0, 1 # t0 => i
    li  t1, 0 # t1 => t
    li  a1, 6

tetrahedral_loop:
    bge  t1, a0, tetrahedral_exit
    addi t2, t0, 1 # t2 => i + 1
    addi t1, t0, 2 # t1 => t => i + 2
    mul  t1, t1, t0 # t => i * (i + 2)
    mul  t1, t1, t2 # t => i * (i + 1) * (i + 2)
    div  t1, t1, a1
    addi t0, t0, 1
    j tetrahedral_loop

tetrahedral_exit:
    bne t1, a0, tetrahedral_false
```

Parzialmente corretta
Punteggio di questo invio: 0,0/5,0.

Commento:
- perchè si confronta con a2?

- calcolo di t non corretto

Domanda **2**
Risposta errata
Punteggio ottenuto 0,0 su 5,0

Scrivere una procedura chiamata `bar_odd(array, size)` che ritorna la somma del risultato dell'applicazione della funzione `bar` a tutti gli elementi nelle posizioni dispari dell'array di word `array`.
La funzione `bar` è già implementata in RISC-V e non deve essere riscritta.
Il valore di ritorno deve essere lasciato nel registro `a0`.

Il seguente codice in C implementa `bar_odd` (convertirlo in RISC-V):

```
// il codice RISC-V che realizza bar è disponibile sotto
// int (in C) equivale a word (in RISC-V)

int bar_odd(int array[], int size) {
    int sum = 0;
    for (int i = 1; i < size; i=i+2) {
        sum += bar(array[i]);
    }
    return sum;
}
```

La funzione `bar_odd` deve utilizzare la funzione `bar`. Soluzioni che non utilizzano la funzione `bar` verranno considerate invalide.

Attenzione:

- Incollare solo la funzione `bar_odd` (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 8,5,3,7,2,6,4,1
        size:  .word 8

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

        # Risultato atteso su a0: 38

        #exit
        li   a7, 10
        ecall

    #*********************************************************
    # Implementazione di bar: Non incollarla nella risposta
    #*********************************************************
    bar:
        li  t0, 2
        mul a0, a0, t0
        ret

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

Answer:

Reset answer

```
1  bar_odd:
2      addi sp,sp,-16
3      sd ra,0(sp)
4      sd a0,8(sp)
5      sd a1,16(sp)
6
7      li t0,0
8      li t1,1
9
10     bge t1,a1
11
```

Testing was aborted due to error.
Your code must pass all tests to earn any marks. Try again.

Question author's solution (Asm):

```
1
2  ###############################################################################
3  # Procedure bar_odd(array, size)
4  # a0 -> address of array
5  # a1 -> size
6  # apply bar to all elements in odd positions of the array, return the sur
7  ###############################################################################
8  bar_odd:
9      addi sp, sp, -40
10     sd   ra, 0(sp)
11     sd   s1, 8(sp)
12     sd   s2, 16(sp)
13     sd   s3, 24(sp)
14     sd   s4, 32(sp)
15
16     li   s1, 1      # i
17     mv   s2, a0     # address
18     mv   s3, a1     # size
19     li   s4, 0      # sum
20
21 bar_odd_loop:
22     bge  s1, s3, bar_odd_return
```

Risposta errata
Punteggio di questo invio: 0,0/5,0.

Commento:
- soluzione incompleta