

Relazione del progetto di laboratorio

Architetture degli elaboratori

Esercizio 1 - Analisi di stringhe

TESTO

Utilizzando QtSpim, scrivere e provare un programma che prende in input una stringa qualsiasi di dimensione massima di 100 caratteri (es, "uno due ciao 33 tree tre uno Uno di eci"), e traduce ogni sua sotto-sequenza di caratteri separati da almeno uno spazio (nell'esempio, le sotto-sequenze "uno", "due", "ciao", "tree", "tre", "uno", "Uno", "di", "eci") applicando le seguenti regole:

- ogni sotto-sequenza "uno" si traduce nel carattere '1';
- ogni sotto-sequenza "due" si traduce nel carattere '2';
- ogni sotto-sequenza "nove" si traduce nel carattere '9';
- qualsiasi altra sotto-sequenza si traduce nel carattere '?'.

Nell'esempio considerato, se "uno due ciao 33 tree tre uno Uno di eci" è la stringa di input inserita da tastiera, a seguito della traduzione il programma dovrà stampare su console la seguente stringa di output:

- Risultato della traduzione: 1 2 ? ? ? ? 1 ? ? ?

DESCRIZIONE DELLA SOLUZIONE

Il programma chiede immediatamente in input la stringa da analizzare (salvata nello spazio di memoria **user**), con una dimensione massima di 100 caratteri, e ne carica l'indirizzo di memoria nel registro **\$s0** per futuri utilizzi. A questo punto, il blocco di istruzioni identificato dall'etichetta **strcmp** si occupa di caricare negli appositi registri gli indirizzi di memoria delle precedentemente dichiarate stringhe "uno" (stringa **one**), "due" (stringa **two**) e "nove" (stringa **nine**). I registri in questione sono **\$t1** per **one**, **\$t2** per **two** e **\$t3** per **nine**; mentre i caratteri di tali stringhe vengono rispettivamente salvati nei registri **\$t4**, **\$t5** e **\$t6**. Sempre in **strcmp** avviene il controllo alla base del programma, ovvero l'estrazione della prima lettera della parola attualmente analizzata della stringa **user** (salvata nel registro **\$t0**), per confrontarla con la prima lettera delle stringhe **one**, **two** e **nine**. Se la prima lettera della stringa utente coincide con esattamente una delle prime lettere delle stringhe **one**, **two** o **nine**, allora si salta al blocco di istruzioni chiamato **cmp_one**, se tale lettera è 'u', **cmp_two**, se tale lettera è 'd', oppure **cmp_nine**, se tale lettera è 'n'. Nel caso in cui la stringa da analizzare fosse finita, ovvero si raggiunge il carattere di fine stringa, il programma salta alla procedura **exit** di ritorno al sistema operativo, mentre se i confronti sopra descritti non avessero avuto successo, il programma salterebbe direttamente a **print_question** per stampare un punto interrogativo, in quanto la parola analizzata non corrisponderebbe né a "uno", né a "due", né a "nove".

Le funzioni **cmp_one**, **cmp_two** e **cmp_nine** si occupano di analizzare in modo parallelo i caratteri, successivi al primo, delle stringhe **one** oppure **two** oppure **nine** e della parola attualmente analizzata della stringa utente. Se almeno uno dei caratteri estratti non coincide, allora salta alla funzione **print_question**; se invece tutti i caratteri estratti coincidono, salta alla funzione **check_one**, da **cmp_one**, alla funzione **check_two**, da **cmp_two**, oppure alla funzione **check_nine**, da **cmp_nine**. Nelle funzioni di "check" si va a controllare il carattere successivo all'ultimo estratto ('o' per "uno", 'e' per "due", 'e' per "nove"), per verificare che tale carattere sia effettivamente uno spazio oppure il carattere di fine stringa. Tale controllo viene effettuato per evitare situazioni del tipo "unociao", ovvero situazioni in cui alla parola corretta "uno", "due" o "nove" viene concatenata una qualunque altra parola.

Se i controlli effettuati dalle funzioni di “check” hanno successo si procede a stampare il numero corrispondente alla parola analizzata (1 per “uno”, 2 per “due” e 9 per “nove”), altrimenti si stampa un punto interrogativo. A questo punto viene stampato uno spazio con le istruzioni identificate dall’etichetta **print_space** e con la funzione **next_word** si fa scorrere il puntatore alla stringa utente fino a che non raggiunge uno spazio. Se la funzione **next_word** dovesse raggiungere il carattere di fine stringa, salterebbe alla funzione **exit**. Dopo aver raggiunto una nuova sotto-sequenza della stringa **user**, è possibile richiamare la funzione iniziale di smistamento **strcmp**. Questi passaggi verranno ripetuti fino a che non verrà analizzata tutta la stringa, ovvero finché non si raggiungerà il carattere di fine stringa.

NOTE: I controlli necessari a verificare il raggiungimento del carattere di fine stringa sono stati effettuati confrontando il carattere estratto con il carattere corrispondente al codice ASCII 10 (notazione decimale), ovvero il carattere di “Line Feed”.

SIMULAZIONE

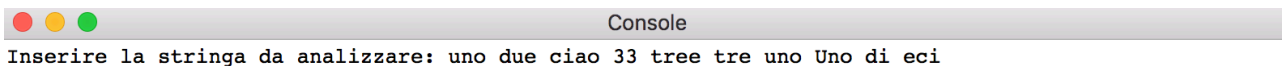
Poiché lo “user stack” e lo “user data segment” non variano durante l’esecuzione del codice, di seguito vengono riportate due simulazioni-tipo dell’esecuzione del programma.

1. Simuliamo su QtSpim, utilizzando la stringa d’esempio fornita dal testo dell’esercizio:
 - Il programma chiede in input la stringa da analizzare



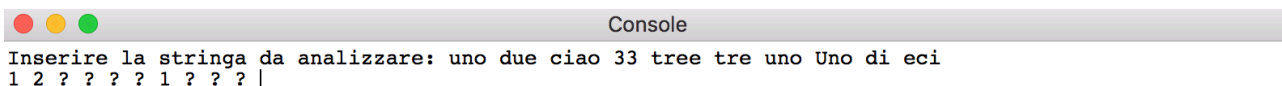
```
Console
Inserire la stringa da analizzare: |
```

- Si inserisce la stringa che si desidera analizzare, per un massimo di 100 caratteri



```
Console
Inserire la stringa da analizzare: uno due ciao 33 tree tre uno Uno di eci
```

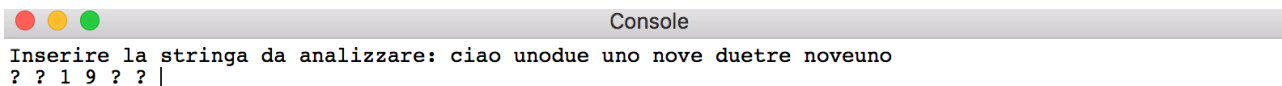
- Dopo aver premuto invio, il programma restituisce il risultato della traduzione



```
Console
Inserire la stringa da analizzare: uno due ciao 33 tree tre uno Uno di eci
1 2 ? ? ? ? 1 ? ? ? |
```

2. Simuliamo su QtSpim, utilizzando una stringa qualunque:

- Dopo aver inserito la stringa da analizzare e aver premuto invio, il programma restituisce il risultato della traduzione



```
Console
Inserire la stringa da analizzare: ciao unodue uno nove duetre noveuno
? ? 1 9 ? ? |
```

CODICE MIPS

```
# Esercizio 1
# Nome: Alessio Falai
# Matricola: 6134275
# Email: alessio.falai@stud.unifi.it
# Data di consegna: 02/07/2017

.data
    prompt: .asciiz "Inserire la stringa da analizzare: "
    user: .space 100
    one: .asciiz "uno"
    two: .asciiz "due"
    nine: .asciiz "nove"

.text
.globl main
main:
    # Stampa della stringa di inserimento
    la $a0, prompt
    li $v0, 4
    syscall

    # Input della stringa da analizzare
    la $a0, user
    li $a1, 100
    li $v0, 8
    syscall

    # Carica la stringa utente
    la $s0, user

strcmp:
    la $t1, one           # Carica nel registro $t1 l'indirizzo
della stringa "uno"
    la $t2, two           # Carica nel registro $t2 l'indirizzo
della stringa "due"
    la $t3, nine          # Carica nel registro $t3 l'indirizzo
della stringa "nove"
    lb $t0, 0($s0)        # Carica nel registro $t0 il primo
carattere della nuova parola della stringa da analizzare
    lb $t4, 0($t1)        # Carica nel registro $t4 il primo
carattere della stringa "uno"
    lb $t5, 0($t2)        # Carica nel registro $t5 il primo
carattere della stringa "due"
    lb $t6, 0($t3)        # Carica nel registro $t6 il primo
carattere della stringa "nove"
    addi $t7, $zero, 1    # Inizializza il contatore del numero
di caratteri di ogni parola
    beq $t0, 10, exit     # Se la stringa da analizzare è finita
esce dal programma
    beq $t0, $t4, cmp_one # Se il primo carattere della nuova
parola della stringa da analizzare è uguale a 'u' salta a cmp_one
```

```
    beq $t0, $t5, cmp_two # Se il primo carattere della nuova
parola della stringa da analizzare è uguale a 'd' salta a cmp_two
    beq $t0, $t6, cmp_nine # Se il primo carattere della nuova
parola della stringa da analizzare è uguale a 'n' salta a cmp_nine
    j print_question      # Se i controlli precedenti falliscono
stampa un punto interrogativo

# Controlla se la parola analizzata corrisponde a "uno"
cmp_one:
    beq $t7, 3, check_one # Se $t7 è uguale a 3, significa che
abbiamo analizzato tutta la parola "uno"
    addi $t1, $t1, 1      # Incrementa il puntatore all'indirizzo
della stringa "uno"
    addi $s0, $s0, 1      # Incrementa il puntatore all'indirizzo
della stringa da analizzare
    lb $t0, 0($s0)        # Carica il byte successivo della
stringa da analizzare
    lb $t4, 0($t1)        # Carica il byte successivo della
stringa "uno"
    addi $t7, $t7, 1      # Incrementa il numero di caratteri
analizzati
    beq $t0, $t4, cmp_one # Se i due caratteri estratti
coincidono, passa al controllo dei byte successivi
    j print_question      # Altrimenti stampa un punto
interrogativo

# Controlla il carattere successivo a 'o' della stringa da
analizzare
check_one:
    addi $s0, $s0, 1      # Incrementa il puntatore all'indirizzo
della stringa da analizzare
    lb $t0, 0($s0)        # Carica il carattere successivo della
stringa da analizzare
    beq $t0, 10, print_one # Se il carattere estratto è il
carattere di fine riga, stampa 1
    beq $t0, 32, print_one # Se il carattere estratto è uno
spazio, stampa 1
    j print_question      # Altrimenti stampa un punto
interrogativo

# Controlla se la parola analizzata corrisponde a "due"
cmp_two:
    beq $t7, 3, check_two # Se $t7 è uguale a 3, significa che
abbiamo analizzato tutta la parola "due"
    addi $t2, $t2, 1      # Incrementa il puntatore all'indirizzo
della stringa "due"
    addi $s0, $s0, 1      # Incrementa il puntatore all'indirizzo
della stringa da analizzare
    lb $t0, 0($s0)        # Carica il byte successivo della
stringa da analizzare
    lb $t5, 0($t2)        # Carica il byte successivo della
stringa "due"
```

```
    addi $t7, $t7, 1      # Incrementa il numero di caratteri
analizzati
    beq $t0, $t5, cmp_two # Se i due caratteri estratti
coincidono, passa al controllo dei byte successivi
    j print_question      # Altrimenti stampa un punto
interrogativo

# Controlla il carattere successivo a 'e' della stringa da
analizzare
check_two:
    addi $s0, $s0, 1      # Incrementa il puntatore all'indirizzo
della stringa da analizzare
    lb $t0, 0($s0)        # Carica il carattere successivo della
stringa da analizzare
    beq $t0, 10, print_two # Se il carattere estratto è il
carattere di fine riga, stampa 2
    beq $t0, 32, print_two # Se il carattere estratto è uno
spazio, stampa 2
    j print_question      # Altrimenti stampa un punto
interrogativo

# Controlla se la parola analizzata corrisponde a "nove"
cmp_nine:
    beq $t7, 4, check_nine # Se $t7 è uguale a 4, significa che
abbiamo analizzato tutta la parola "nove"
    addi $t3, $t3, 1      # Incrementa il puntatore all'indirizzo
della stringa "nove"
    addi $s0, $s0, 1      # Incrementa il puntatore all'indirizzo
della stringa da analizzare
    lb $t0, 0($s0)        # Carica il byte successivo della
stringa da analizzare
    lb $t6, 0($t3)        # Carica il byte successivo della
stringa "nove"
    addi $t7, $t7, 1      # Incrementa il numero di caratteri
analizzati
    beq $t0, $t6, cmp_nine # Se i due caratteri estratti
coincidono, passa al controllo dei byte successivi
    j print_question      # Altrimenti stampa un punto
interrogativo

# Controlla il carattere successivo a 'e' della stringa da
analizzare
check_nine:
    addi $s0, $s0, 1      # Incrementa il puntatore all'indirizzo
della stringa da analizzare
    lb $t0, 0($s0)        # Carica il carattere successivo della
stringa da analizzare
    beq $t0, 10, print_nine # Se il carattere estratto è il
carattere di fine riga, stampa 9
    beq $t0, 32, print_nine # Se il carattere estratto è uno
spazio, stampa 9
```

```
j print_question      # Altrimenti stampa un punto
interrogativo

# Stampa il numero 1
print_one:
    addi $s0, $s0, -1    # Decrementa il puntatore della stringa
da analizzare in seguito ai controlli di check_one
    li $a0, 1
    li $v0, 1
    syscall

j print_space

# Stampa il numero 2
print_two:
    addi $s0, $s0, -1    # Decrementa il puntatore della stringa
da analizzare in seguito ai controlli di check_two
    li $a0, 2
    li $v0, 1
    syscall

j print_space

# Stampa il numero 9
print_nine:
    addi $s0, $s0, -1    # Decrementa il puntatore della stringa
da analizzare in seguito ai controlli di check_nine
    li $a0, 9
    li $v0, 1
    syscall

j print_space

# Stampa uno spazio
print_space:
    li $a0, 32
    li $v0, 11
    syscall

j next_word

# Seleziona la parola successiva della stringa
next_word:
    addi $s0, $s0, 1
    lb $t0, 0($s0)
    beq $t0, 10, exit
    bne $t0, 32, next_word
    addi $s0, $s0, 1

j strcmp
```

```
# Stampa un punto interrogativo
print_question:
    li $a0, 63
    li $v0, 11
    syscall

    j print_space

# Esce dal programma
exit:
    li $v0, 10
    syscall
```

Esercizio 2 - Procedure annidate e ricorsive

TESTO

Siano G e F due procedure definite come segue (in pseudo-linguaggio di alto livello):

```
Procedure  $G(n)$ 
begin
     $b := 0$ 
    for  $k := 0, 1, 2, \dots, n$  do
        begin
             $u := F(k)$ 
             $b := b^2 + u$ 
        end
    end
    return  $b$ 
end

Procedure  $F(n)$ 
begin
    if  $n = 0$  then return 1
    else return  $2 * F(n - 1) + n$ 
end
```

Utilizzando QtSpim, scrivere e provare un programma che legga inizialmente un numero naturale n compreso tra 1 e 8, e che visualizzi su console:

- il valore restituito dalla procedura $G(n)$, implementando G e F come descritto precedentemente. Le chiamate alle due procedure G ed F devono essere realizzate utilizzando l'istruzione `jal` (jump and link).
- la traccia con la sequenza delle chiamate annidate (con argomento fra parentesi) ed i valori restituiti dalle varie chiamate annidate (valore restituito fra parentesi), sia per G che per F .

Mostrare e discutere nella relazione l'evoluzione dello stack nel caso specifico in cui $n = 3$.

Esempio di output su console nel caso in cui $n = 1$:

Valore restituito dalla procedura: $G(1) = 4$

Traccia:

$G(1) \rightarrow F(0) \rightarrow F:\text{return}(1) \rightarrow F(1) \rightarrow F(0) \rightarrow F:\text{return}(1) \rightarrow F:\text{return}(3) \rightarrow G:\text{return}(4)$

Infatti: $G(1)$ richiama $F(0)$, che ritorna il valore 1; quindi viene richiamata $F(1)$, che a sua volta richiama $F(0)$; $F(0)$ ritorna il valore 1, quindi $F(1)$ ritorna il valore 3, ed infine $G(1)$ ritorna il valore 4 che è il risultato finale.

DESCRIZIONE DELLA SOLUZIONE (LINGUAGGIO AD ALTO LIVELLO C)

```
#include <stdio.h>
```

```
#include <iostream>
```

```
using namespace std;
```

```
int F(int n) {
    if(n == 0) {
        cout << "F:return(" << 1 << ")-->";
        return 1;
    }
    else {
        cout << "F(" << n - 1 << ")-->";
        int x = 2 * F(n - 1) + n;
        cout << "F:return(" << x << ")-->";
        return x;
    }
}
```

```
int G(int n) {
    int b = 0;
    for(int k = 0; k <= n; k++) {
        cout << "F(" << k << ")-->";
        int u = F(k);
        b = (b * b) + u;
    }
    cout << "G:return(" << b << ")";
    return b;
}
```

```
int main() {
    int n = 0;
    cin >> n;
    if(n < 1 || n > 8) {
        cout << "Errore.";
        return 0;
    }
    cout << "G(" << n << ")-->";
    int r = G(n);
    return 0;
}
```


DESCRIZIONE DELLA SOLUZIONE (LINGUAGGIO NATURALE)

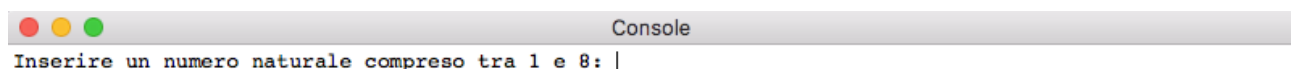
Il programma chiede immediatamente in input il numero naturale n , che viene salvato nel registro **\$s0**. Nel caso in cui il numero inserito non sia compreso tra 1 e 8, viene effettuato un salto al blocco di istruzioni identificato dall'etichetta **print_error**, che si occupa di stampare un errore e ritornare alla funzione principale **main** per chiedere nuovamente in input il numero n . Se invece il numero inserito rientrasse nei limiti consentiti, allora verrebbe chiamata la procedura **g**, con il parametro n inserito da tastiera, dopo aver effettuato la stampa della chiamata di funzione attraverso la procedura **print_call**. Nella procedura **g** viene inizialmente preparato lo stack per fare spazio a due word, una per l'indirizzo di ritorno della funzione e una per l'argomento passato alla funzione, che inizialmente è proprio n ; e vengono inizializzate le due variabili **b** e **k** nei registri **\$s1** e **\$s2**. Dopodiché, per **k** che va da 0 a n , viene richiamata la funzione **f** con parametro **k**, dopo aver effettuato la stampa della chiamata di funzione attraverso la procedura **print_call**, e viene salvato il valore di ritorno di tale chiamata nel registro **\$s3** (variabile **u**), oltre a salvare in **\$s1** la somma di **b** al quadrato e **u**. A questo punto viene stampata la stringa di ritorno della funzione **g** attraverso la procedura **print_return** e vengono ripristinati i valori dei registri effettuando due estrazioni dallo stack. Nella procedura **f**, come nella procedura **g**, viene inizialmente preparato lo stack per fare spazio a due word, una per l'indirizzo di ritorno della funzione e una per l'argomento passato alla funzione. Dopodiché viene controllato il valore del parametro x passato alla funzione; se x è 0 la funzione stampa la stringa di ritorno della funzione **f** con parametro 1 e ritorna 1, altrimenti salva nello stack il valore di x , richiama ricorsivamente se stessa con $x - 1$, dopo aver stampato la chiamata di procedura della funzione **f** con parametro $x - 1$, e somma x al valore di ritorno della chiamata ricorsiva, dopo aver estratto il valore originario di x dallo stack. A questo punto, una volta effettuate tutte le chiamate ricorsive nella procedura **f**, vengono estratti i valori inizialmente inseriti nello stack per ripristinare i registri e ritornare al chiamante. In conclusione, appena terminate le chiamate ricorsive e annidate delle funzioni **f** e **g**, è possibile saltare al blocco di istruzioni **exit** per terminare l'esecuzione del programma e ritornare al sistema operativo.

NOTE: L'unico argomento accettato dalle funzioni **f**, **g**, **print_call** e **print_return** viene passato alla funzione attraverso il registro **\$a3**, mentre il valore di ritorno delle procedure **f** e **g** viene salvato nel registro **\$v1**.

SIMULAZIONE

1. Simuliamo su QtSpim, utilizzando il valore $n = 1$ fornito dal testo dell'esercizio:

- Il programma chiede in input il numero naturale n

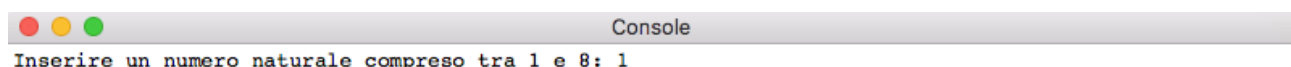


```

Console
Inserire un numero naturale compreso tra 1 e 8: |

```

- Si inserisce il numero naturale n di cui si desidera analizzare il comportamento

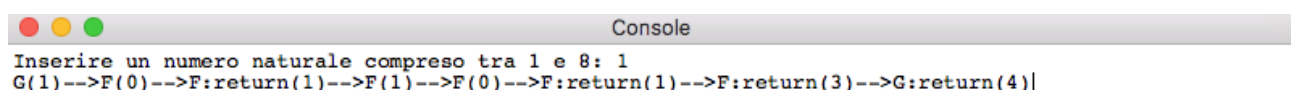


```

Console
Inserire un numero naturale compreso tra 1 e 8: 1

```

- Il programma restituisce la traccia delle chiamate ricorsive e il valore restituito dalla funzione **G**, con parametro n



```

Console
Inserire un numero naturale compreso tra 1 e 8: 1
G(1)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->G:return(4)|

```

2. Simuliamo su QtSpim, utilizzando il valore $n = 3$, mostrando l'evoluzione dello stack pointer, tramite le operazioni che lo modificano e l'indirizzo con il relativo contenuto:

```
lw $4, 0($29)           ; 183: lw $a0 0($sp) # argc
addiu $5, $29, 4         ; 184: addiu $a1 $sp 4 # argv
[7ffffdbc] 0000000003

addi $29, $29, -8        ; 58: addi $sp, $sp, -8
sw $31, 4($29)           ; 59: sw $ra, 4($sp)
sw $7, 0($29)            ; 60: sw $a3, 0($sp)
[7ffffdb4] 0000000003 0004194404 0000000003
```

Console
Inserire un numero naturale compreso tra 1 e 8: 3

```
lw $7, 0($29)           ; 77: lw $a3, 0($sp)
lw $31, 4($29)          ; 78: lw $ra, 4($sp)
addi $29, $29, 8        ; 79: addi $sp, $sp, 8
[7ffffdbc] 0000000003

addi $29, $29, -8        ; 174: addi $sp, $sp, -8
sw $31, 4($29)           ; 175: sw $ra, 4($sp)
sw $7, 0($29)            ; 176: sw $a3, 0($sp)
[7ffffdb4] 0000000003 0004194408 0000000003
```

Console
Inserire un numero naturale compreso tra 1 e 8: 3
G(3)-->|

```
addi $29, $29, -8        ; 58: addi $sp, $sp, -8
sw $31, 4($29)           ; 59: sw $ra, 4($sp)
sw $7, 0($29)            ; 60: sw $a3, 0($sp)
[7ffffdac] 0000000000
```

Console
Inserire un numero naturale compreso tra 1 e 8: 3
G(3)-->F(0)-->

```
lw $7, 0($29)           ; 77: lw $a3, 0($sp)
lw $31, 4($29)          ; 78: lw $ra, 4($sp)
addi $29, $29, 8        ; 79: addi $sp, $sp, 8
[7ffffdb4] NON SPECIFICATO

addi $29, $29, -8        ; 121: addi $sp, $sp, -8
sw $31, 4($29)           ; 122: sw $ra, 4($sp)
sw $7, 0($29)            ; 123: sw $a3, 0($sp)
[7ffffdac] 0000000000
```

```

addi $29, $29, -8          ; 84: addi $sp, $sp, -8
sw $31, 4($29)           ; 85: sw $ra, 4($sp)
sw $7, 0($29)            ; 86: sw $a3, 0($sp)
[7ffffda4] 0000000001 0004194780 0000000000

```

```

lw $7, 0($29)            ; 114: lw $a3, 0($sp)
lw $31, 4($29)          ; 115: lw $ra, 4($sp)
addi $29, $29, 8         ; 116: addi $sp, $sp, 8
[7ffffdac] NON SPECIFICATO

```

```

lw $31, 4($29)          ; 166: lw $ra, 4($sp)
addi $29, $29, 8         ; 167: addi $sp, $sp, 8
[7ffffdb4] NON SPECIFICATO

```

```

Console
Inserire un numero naturale compreso tra 1 e 8: 3
G(3)-->F(0)-->F:return(1)-->|

```

```

addi $29, $29, -8          ; 58: addi $sp, $sp, -8
sw $31, 4($29)           ; 59: sw $ra, 4($sp)
sw $7, 0($29)            ; 60: sw $a3, 0($sp)
[7ffffdac] 0000000001

```

```

Console
Inserire un numero naturale compreso tra 1 e 8: 3
G(3)-->F(0)-->F:return(1)-->F(1)-->|

```

```

addi $29, $29, -8          ; 121: addi $sp, $sp, -8
sw $31, 4($29)           ; 122: sw $ra, 4($sp)
sw $7, 0($29)            ; 123: sw $a3, 0($sp)
[7ffffdac] 0000000001

```

```

addi $29, $29, -8          ; 58: addi $sp, $sp, -8
sw $31, 4($29)           ; 59: sw $ra, 4($sp)
sw $7, 0($29)            ; 60: sw $a3, 0($sp)
[7ffffda4] 0000000000 0004194692 0000000001

```

```

lw $7, 0($29)            ; 77: lw $a3, 0($sp)
lw $31, 4($29)          ; 78: lw $ra, 4($sp)
addi $29, $29, 8         ; 79: addi $sp, $sp, 8
[7ffffdac] NON SPECIFICATO

```

```

Console
Inserire un numero naturale compreso tra 1 e 8: 3
G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->|

```

```

addi $29, $29, -8          ; 121: addi $sp, $sp, -8
sw $31, 4($29)           ; 122: sw $ra, 4($sp)
sw $7, 0($29)            ; 123: sw $a3, 0($sp)
[7ffffda4] 0000000000 0004194696 0000000001

```

```

addi $29, $29, -8           ; 84: addi $sp, $sp, -8
sw $31, 4($29)             ; 85: sw $ra, 4($sp)
sw $7, 0($29)              ; 86: sw $a3, 0($sp)
[7ffffd9c]      0000000001

```

Console

Inserire un numero naturale compreso tra 1 e 8: 3
 G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->|

```

lw $7, 0($29)               ; 114: lw $a3, 0($sp)
lw $31, 4($29)             ; 115: lw $ra, 4($sp)
addi $29, $29, 8           ; 116: addi $sp, $sp, 8
[7ffffda4]      NON SPECIFICATO

```

```

lw $31, 4($29)             ; 166: lw $ra, 4($sp)
addi $29, $29, 8           ; 167: addi $sp, $sp, 8
[7ffffdac]      NON SPECIFICATO

```

```

addi $29, $29, -8           ; 84: addi $sp, $sp, -8
sw $31, 4($29)             ; 85: sw $ra, 4($sp)
sw $7, 0($29)              ; 86: sw $a3, 0($sp)
[7ffffda4]      0000000003  0004194740  0000000001

```

Console

Inserire un numero naturale compreso tra 1 e 8: 3
 G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->|

```

lw $7, 0($29)               ; 114: lw $a3, 0($sp)
lw $31, 4($29)             ; 115: lw $ra, 4($sp)
addi $29, $29, 8           ; 116: addi $sp, $sp, 8
[7ffffdac]      NON SPECIFICATO

```

```

lw $31, 4($29)             ; 150: lw $ra, 4($sp)
addi $29, $29, 8           ; 151: addi $sp, $sp, 8
[7ffffdb4]      NON SPECIFICATO

```

```

addi $29, $29, -8           ; 58: addi $sp, $sp, -8
sw $31, 4($29)             ; 59: sw $ra, 4($sp)
sw $7, 0($29)              ; 60: sw $a3, 0($sp)
[7ffffdac]      0000000002

```

Console

Inserire un numero naturale compreso tra 1 e 8: 3
 G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->|

```

lw $7, 0($29)               ; 77: lw $a3, 0($sp)
lw $31, 4($29)             ; 78: lw $ra, 4($sp)
addi $29, $29, 8           ; 79: addi $sp, $sp, 8
[7ffffdb4]      NON SPECIFICATO

```

```


addi $29, $29, -8          ; 121: addi $sp, $sp, -8
sw $31, 4($29)           ; 122: sw $ra, 4($sp)
sw $7, 0($29)            ; 123: sw $a3, 0($sp)
[7ffffdac]      0000000002

```

```

addi $29, $29, -8          ; 58: addi $sp, $sp, -8
sw $31, 4($29)           ; 59: sw $ra, 4($sp)
sw $7, 0($29)            ; 60: sw $a3, 0($sp)
[7ffffda4]      0000000001  0004194692  0000000002

```



Console

Inserire un numero naturale compreso tra 1 e 8: 3
 G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->

```

lw $7, 0($29)             ; 77: lw $a3, 0($sp)
lw $31, 4($29)           ; 78: lw $ra, 4($sp)
addi $29, $29, 8          ; 79: addi $sp, $sp, 8
[7ffffdac]      NON SPECIFICATO

```

```


addi $29, $29, -8          ; 121: addi $sp, $sp, -8
sw $31, 4($29)           ; 122: sw $ra, 4($sp)
sw $7, 0($29)            ; 123: sw $a3, 0($sp)
[7ffffda4]      0000000001  0004194696  0000000002

```

```

addi $29, $29, -8          ; 58: addi $sp, $sp, -8
sw $31, 4($29)           ; 59: sw $ra, 4($sp)
sw $7, 0($29)            ; 60: sw $a3, 0($sp)
[7ffffd9c]      0000000000

```



Console

Inserire un numero naturale compreso tra 1 e 8: 3
 G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->F(0)-->

```

lw $7, 0($29)             ; 77: lw $a3, 0($sp)
lw $31, 4($29)           ; 78: lw $ra, 4($sp)
addi $29, $29, 8          ; 79: addi $sp, $sp, 8
[7ffffda4]      NON SPECIFICATO

```

```


addi $29, $29, -8          ; 121: addi $sp, $sp, -8
sw $31, 4($29)           ; 122: sw $ra, 4($sp)
sw $7, 0($29)            ; 123: sw $a3, 0($sp)
[7ffffd9c]      0000000000

```

```

addi $29, $29, -8          ; 84: addi $sp, $sp, -8
sw $31, 4($29)           ; 85: sw $ra, 4($sp)
sw $7, 0($29)            ; 86: sw $a3, 0($sp)
[7ffffd94]      0000000001  0004194780  0000000000

```



Console

Inserire un numero naturale compreso tra 1 e 8: 3
 G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->F(0)-->
 >F:return(1)-->

```

lw $7, 0($29)           ; 114: lw $a3, 0($sp)
lw $31, 4($29)          ; 115: lw $ra, 4($sp)
addi $29, $29, 8        ; 116: addi $sp, $sp, 8
[7ffffd9c]             NON SPECIFICATO

lw $31, 4($29)          ; 166: lw $ra, 4($sp)
addi $29, $29, 8        ; 167: addi $sp, $sp, 8
[7ffffda4]             NON SPECIFICATO

addi $29, $29, -8       ; 84: addi $sp, $sp, -8
sw $31, 4($29)          ; 85: sw $ra, 4($sp)
sw $7, 0($29)           ; 86: sw $a3, 0($sp)
[7ffffd9c]             0000000003

```

Console

Inserire un numero naturale compreso tra 1 e 8: 3
 G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->

```

lw $7, 0($29)           ; 114: lw $a3, 0($sp)
lw $31, 4($29)          ; 115: lw $ra, 4($sp)
addi $29, $29, 8        ; 116: addi $sp, $sp, 8
[7ffffda4]             NON SPECIFICATO

lw $31, 4($29)          ; 150: lw $ra, 4($sp)
addi $29, $29, 8        ; 151: addi $sp, $sp, 8
[7ffffdac]             NON SPECIFICATO

addi $29, $29, -8       ; 84: addi $sp, $sp, -8
sw $31, 4($29)          ; 85: sw $ra, 4($sp)
sw $7, 0($29)           ; 86: sw $a3, 0($sp)
[7ffffda4]             0000000008  0004194740  0000000002

```

Console

Inserire un numero naturale compreso tra 1 e 8: 3
 G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F:return(8)-->

```

lw $7, 0($29)           ; 114: lw $a3, 0($sp)
lw $31, 4($29)          ; 115: lw $ra, 4($sp)
addi $29, $29, 8        ; 116: addi $sp, $sp, 8
[7ffffdac]             NON SPECIFICATO

lw $31, 4($29)          ; 150: lw $ra, 4($sp)
addi $29, $29, 8        ; 151: addi $sp, $sp, 8
[7ffffdb4]             NON SPECIFICATO

addi $29, $29, -8       ; 58: addi $sp, $sp, -8
sw $31, 4($29)          ; 59: sw $ra, 4($sp)
sw $7, 0($29)           ; 60: sw $a3, 0($sp)
[7ffffdac]             0000000003

```

```

Console
Inserire un numero naturale compreso tra 1 e 8: 3
G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->F(0)--
>F:return(1)-->F:return(3)-->F:return(8)-->F(3)-->

```

```

lw $7, 0($29)           ; 77: lw $a3, 0($sp)
lw $31, 4($29)          ; 78: lw $ra, 4($sp)
addi $29, $29, 8        ; 79: addi $sp, $sp, 8
[7ffffdb4]             NON SPECIFICATO

addi $29, $29, -8       ; 121: addi $sp, $sp, -8
sw $31, 4($29)          ; 122: sw $ra, 4($sp)
sw $7, 0($29)           ; 123: sw $a3, 0($sp)
[7ffffdac]             0000000003

addi $29, $29, -8       ; 58: addi $sp, $sp, -8
sw $31, 4($29)          ; 59: sw $ra, 4($sp)
sw $7, 0($29)           ; 60: sw $a3, 0($sp)
[7ffffda4]             0000000002  0004194692  0000000003

lw $7, 0($29)           ; 77: lw $a3, 0($sp)
lw $31, 4($29)          ; 78: lw $ra, 4($sp)
addi $29, $29, 8        ; 79: addi $sp, $sp, 8
[7ffffdac]             NON SPECIFICATO

```

```

Console
Inserire un numero naturale compreso tra 1 e 8: 3
G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->F(0)--
>F:return(1)-->F:return(3)-->F:return(8)-->F(3)-->F(2)-->F(1)-->

```

```

addi $29, $29, -8       ; 121: addi $sp, $sp, -8
sw $31, 4($29)          ; 122: sw $ra, 4($sp)
sw $7, 0($29)           ; 123: sw $a3, 0($sp)
[7ffffda4]             0000000002  0004194696  0000000003

addi $29, $29, -8       ; 58: addi $sp, $sp, -8
sw $31, 4($29)          ; 59: sw $ra, 4($sp)
sw $7, 0($29)           ; 60: sw $a3, 0($sp)
[7ffffd9c]             0000000003

lw $7, 0($29)           ; 77: lw $a3, 0($sp)
lw $31, 4($29)          ; 78: lw $ra, 4($sp)
addi $29, $29, 8        ; 79: addi $sp, $sp, 8
[7ffffda4]             NON SPECIFICATO

```

```

Console
Inserire un numero naturale compreso tra 1 e 8: 3
G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->F(0)--
>F:return(1)-->F:return(3)-->F:return(8)-->F(3)-->F(2)-->F(1)-->

```

```

addi $29, $29, -8       ; 121: addi $sp, $sp, -8
sw $31, 4($29)          ; 122: sw $ra, 4($sp)
sw $7, 0($29)           ; 123: sw $a3, 0($sp)
[7ffffd9c]             0000000001

```



```

addi $29, $29, -8           ; 58: addi $sp, $sp, -8
sw $31, 4($29)             ; 59: sw $ra, 4($sp)
sw $7, 0($29)              ; 60: sw $a3, 0($sp)
[7ffffd94] 0000000001 0004194692 0000000001

```

Console

Inserire un numero naturale compreso tra 1 e 8: 3
 G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F:return(8)-->F(3)-->F(2)-->F(1)-->F(0)-->|

```

lw $7, 0($29)              ; 77: lw $a3, 0($sp)
lw $31, 4($29)            ; 78: lw $ra, 4($sp)
addi $29, $29, 8          ; 79: addi $sp, $sp, 8
[7ffffd9c] NON SPECIFICATO

```

```

addi $29, $29, -8         ; 121: addi $sp, $sp, -8
sw $31, 4($29)           ; 122: sw $ra, 4($sp)
sw $7, 0($29)            ; 123: sw $a3, 0($sp)
[7ffffd94] 0000000000 0004194696 0000000001

```

```

addi $29, $29, -8         ; 84: addi $sp, $sp, -8
sw $31, 4($29)           ; 85: sw $ra, 4($sp)
sw $7, 0($29)            ; 86: sw $a3, 0($sp)
[7ffffd8c] 0000000001

```

Console

Inserire un numero naturale compreso tra 1 e 8: 3
 G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F:return(8)-->F(3)-->F(2)-->F(1)-->F(0)-->F:return(1)-->

```

lw $7, 0($29)              ; 114: lw $a3, 0($sp)
lw $31, 4($29)            ; 115: lw $ra, 4($sp)
addi $29, $29, 8          ; 116: addi $sp, $sp, 8
[7ffffd94] NON SPECIFICATO

```

```

lw $31, 4($29)            ; 166: lw $ra, 4($sp)
addi $29, $29, 8          ; 167: addi $sp, $sp, 8
[7ffffd9c] NON SPECIFICATO

```

```

addi $29, $29, -8         ; 84: addi $sp, $sp, -8
sw $31, 4($29)           ; 85: sw $ra, 4($sp)
sw $7, 0($29)            ; 86: sw $a3, 0($sp)
[7ffffd94] 0000000003 0004194740 0000000001

```

Console

Inserire un numero naturale compreso tra 1 e 8: 3
 G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F:return(8)-->F(3)-->F(2)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->

```


lw $7, 0($29)              ; 114: lw $a3, 0($sp)
lw $31, 4($29)            ; 115: lw $ra, 4($sp)
addi $29, $29, 8          ; 116: addi $sp, $sp, 8
[7ffffd9c] NON SPECIFICATO

```



```
lw $31, 4($29)          ; 150: lw $ra, 4($sp)
addi $29, $29, 8        ; 151: addi $sp, $sp, 8
[7ffffda4]             NON SPECIFICATO

addi $29, $29, -8       ; 84: addi $sp, $sp, -8
sw $31, 4($29)          ; 85: sw $ra, 4($sp)
sw $7, 0($29)           ; 86: sw $a3, 0($sp)
[7ffffd9c]             0000000001
```




Console

```
Inserire un numero naturale compreso tra 1 e 8: 3
G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->F(0)--
>F:return(1)-->F:return(3)-->F:return(8)-->F(3)-->F(2)-->F(1)-->F(0)-->F:return(1)-->F:return(3)--
>F:return(8)-->
```

```
lw $7, 0($29)           ; 114: lw $a3, 0($sp)
lw $31, 4($29)          ; 115: lw $ra, 4($sp)
addi $29, $29, 8        ; 116: addi $sp, $sp, 8
[7ffffda4]             NON SPECIFICATO
```

```
lw $31, 4($29)          ; 150: lw $ra, 4($sp)
addi $29, $29, 8        ; 151: addi $sp, $sp, 8
[7ffffdac]             NON SPECIFICATO
```

```
addi $29, $29, -8       ; 84: addi $sp, $sp, -8
sw $31, 4($29)          ; 85: sw $ra, 4($sp)
sw $7, 0($29)           ; 86: sw $a3, 0($sp)
[7ffffda4]             0000000019  0004194740  0000000003
```




Console

```
Inserire un numero naturale compreso tra 1 e 8: 3
G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->F(0)--
>F:return(1)-->F:return(3)-->F:return(8)-->F(3)-->F(2)-->F(1)-->F(0)-->F:return(1)-->F:return(3)--
>F:return(8)-->F:return(19)-->
```

```
lw $7, 0($29)           ; 114: lw $a3, 0($sp)
lw $31, 4($29)          ; 115: lw $ra, 4($sp)
addi $29, $29, 8        ; 116: addi $sp, $sp, 8
[7ffffdac]             NON SPECIFICATO
```

```
lw $31, 4($29)          ; 150: lw $ra, 4($sp)
addi $29, $29, 8        ; 151: addi $sp, $sp, 8
[7ffffdb4]             NON SPECIFICATO
```

```
addi $29, $29, -8       ; 84: addi $sp, $sp, -8
sw $31, 4($29)          ; 85: sw $ra, 4($sp)
sw $7, 0($29)           ; 86: sw $a3, 0($sp)
[7ffffdac]             0000000595
```



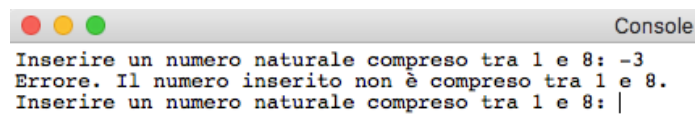
Console

```
Inserire un numero naturale compreso tra 1 e 8: 3
G(3)-->F(0)-->F:return(1)-->F(1)-->F(0)-->F:return(1)-->F:return(3)-->F(2)-->F(1)-->F(0)--
>F:return(1)-->F:return(3)-->F:return(8)-->F(3)-->F(2)-->F(1)-->F(0)-->F:return(1)-->F:return(3)--
>F:return(8)-->F:return(19)-->G:return(595)|
```

```
lw $7, 0($29)           ; 114: lw $a3, 0($sp)
lw $31, 4($29)           ; 115: lw $ra, 4($sp)
addi $29, $29, 8         ; 116: addi $sp, $sp, 8
[7ffffdb4]              NON SPECIFICATO
```

```
lw $31, 4($29)           ; 205: lw $ra, 4($sp)
addi $29, $29, 8         ; 206: addi $sp, $sp, 8
[7ffffdbc]              NON SPECIFICATO
```

3. Simuliamo su QtSpim, utilizzando un valore di n non compreso tra 1 e 8:
- Dopo aver inserito il numero n richiesto in input, il programma controlla che sia compreso tra 1 e 8. Se tale proprietà non è verificata, come in questo caso, il programma restituisce un errore e richiede l'inserimento di un nuovo valore



The screenshot shows a console window titled "Console" with a standard macOS-style title bar (red, yellow, green buttons). The text inside the console reads: "Inserire un numero naturale compreso tra 1 e 8: -3", "Errore. Il numero inserito non è compreso tra 1 e 8.", and "Inserire un numero naturale compreso tra 1 e 8: |".

```
Inserire un numero naturale compreso tra 1 e 8: -3
Errore. Il numero inserito non è compreso tra 1 e 8.
Inserire un numero naturale compreso tra 1 e 8: |
```

CODICE MIPS

```
# Esercizio 2
# Nome: Alessio Falai
# Matricola: 6134275
# Email: alessio.falai@stud.unifi.it
# Data di consegna: 02/07/2017

.data
    prompt: .asciiz "Inserire un numero naturale compreso tra 1 e
8: "
    error: .asciiz "Errore. Il numero inserito non è compreso tra 1
e 8."
    str_arrow: .asciiz "-->"
    str_return: .asciiz ":return"
    char_f: .asciiz "F"
    char_g: .asciiz "G"
    char_openbr: .asciiz "("
    char_closebr: .asciiz ")"

.text
.globl main
main:
    # Stampa la stringa per l'inserimento del numero 'n'
    li $v0, 4
    la $a0, prompt
    syscall

    # Prende in input l'intero 'n'
    li $v0, 5
    syscall

    # Controlla che n sia compreso tra 1 e 8
    add $s0, $v0, $zero
    bgt $s0, 8, print_error
    blt $s0, 1, print_error

    # Chiama la procedura 'g' con il parametro n inserito
    li $v0, 4
    la $a0, char_g
    syscall
    add $a3, $s0, $zero
    jal print_call
    jal g

    j exit

# Se il numero n inserito non è compreso tra 1 e 8 stampa un
errore
print_error:
    li $v0, 4
    la $a0, error
    syscall
```

```
li $a0, 10
li $v0, 11
syscall

j main

# Procedura di stampa della chiamata della funzione 'f' oppure 'g'
print_call:
    addi $sp, $sp, -8          # Prepara due word di spazio nello
stack
    sw $ra, 4($sp)            # Salva l'indirizzo di ritorno nello
stack
    sw $a3, 0($sp)            # Salva l'argomento passato alla
funzione nello stack

    # Stampa una parentesi aperta, l'argomento della funzione ('f'
oppure 'g'), una parentesi chiusa e una freccia
    li $v0, 4
    la $a0, char_openbr
    syscall
    lw $a0, 0($sp)
    li $v0, 1
    syscall
    li $v0, 4
    la $a0, char_closebr
    syscall
    li $v0, 4
    la $a0, str_arrow
    syscall

    # Estrae i valori dallo stack per ripristinare i registri e
ritorna al chiamante
    lw $a3, 0($sp)
    lw $ra, 4($sp)
    addi $sp, $sp, 8
    jr $ra

# Procedura di stampa del valore di ritorno della funzione 'f'
oppure 'g'
print_return:
    addi $sp, $sp, -8          # Prepara due word di spazio nello
stack
    sw $ra, 4($sp)            # Salva l'indirizzo di ritorno nello
stack
    sw $a3, 0($sp)            # Salva l'argomento passato alla
funzione nello stack

    # Stampa la stringa di return, una parentesi aperta,
l'argomento della funzione ('f' oppure 'g') e una parentesi chiusa
    li $v0, 4
    la $a0, str_return
```

```

syscall
li $v0, 4
la $a0, char_openbr
syscall
lw $a0, 0($sp)
li $v0, 1
syscall
li $v0, 4
la $a0, char_closebr
syscall

# Stampa una freccia se non si sta processando l'ultima
chiamata
bne $a1, 1, print_arrow
j end_pr

# Stampa una freccia
print_arrow:
    li $v0, 4
    la $a0, str_arrow
    syscall

# Estrae i valori dallo stack per ripristinare i registri e
ritorna al chiamante
end_pr:
    lw $a3, 0($sp)
    lw $ra, 4($sp)
    addi $sp, $sp, 8
    jr $ra

# Procedura 'f'
f:
    addi $sp, $sp, -8          # Prepara due word di spazio nello
stack                          # stack
    sw $ra, 4($sp)            # Salva l'indirizzo di ritorno nello
stack                          # stack
    sw $a3, 0($sp)            # Salva l'argomento passato alla
funzione nello stack
    bne $a3, $zero, core_f    # Se $a3 non contiene zero, ritorna 2 *
f(n - 1) + n
    j end_f                  # Altrimenti ritorna 1

# Ritorna 2 * f(n - 1) + n
core_f:
    # Calcola (n - 1), stampa la chiamata di procedura f(n - 1)
e richiama 'f' con parametro (n - 1)
    addi $a3, $a3, -1
    li $v0, 4
    la $a0, char_f
    syscall
    jal print_call
    jal f

```

```

        lw $a3, 0($sp)      # Ripristina il valore originario di
'n'
        mul $v1, $v1, 2     # Calcola il doppio del valore
ritornato dalla procedura f(n - 1)
        add $v1, $a3, $v1   # Calcola la somma di 2 * f(n - 1) e
'n'

        # Stampa il valore ritornato dalla procedura 'f'
        li $v0, 4
        la $a0, char_f
        syscall
        add $a3, $v1, $zero
        add $a1, $zero, $zero
        jal print_return

        # Estrae i valori dallo stack per ripristinare i registri e
ritorna al chiamante
        lw $ra, 4($sp)
        addi $sp, $sp, 8
        jr $ra

        # Ritorna 1
end_f:
        # Stampa la stringa di ritorno della procedura 'f' con
parametro 1
        li $v0, 4
        la $a0, char_f
        syscall
        addi $a3, $zero, 1
        add $a1, $zero, $zero
        jal print_return
        li $v1, 1

        # Estrae i valori dallo stack per ripristinare i registri e
ritorna al chiamante
        lw $ra, 4($sp)
        addi $sp, $sp, 8
        jr $ra

# Procedura 'g'
g:
        add $s1, $zero, $zero # Inizializzazione variabile 'b'
        add $s2, $zero, $zero # Inizializzazione variabile 'k'
        addi $sp, $sp, -8     # Prepara due word di spazio nello
stack
        sw $ra, 4($sp)       # Salva l'indirizzo di ritorno nello
stack
        sw $a3, 0($sp)       # Salva l'argomento passato alla
funzione nello stack

```

```
# Per 'k' che va da 0 a 'n', esegue questo blocco di
istruzioni
loop:
    # Chiama la funzione 'f' con parametro 'k' e stampa la
chiamata di procedura
    li $v0, 4
    la $a0, char_f
    syscall
    move $a3, $s2
    jal print_call
    jal f

    move $s3, $v1      # $s3 contiene la variabile 'u'
    mul $s1, $s1, $s1  # $s1 contiene 'b' al quadrato
    add $s1, $s1, $s3  # $s1 contiene la somma di 'b' al
quadrato e 'u'
    addi $s2, $s2, 1   # Incrementa la variabile 'k'
    blt $s2, $s0, loop # Se 'k' è minore di 'n' continua a
ciclare
    beq $s2, $s0, loop # Se 'k' è uguale a 'n' continua a
ciclare

    # Stampa la stringa di ritorno della funzione 'g' con parametro
'b'
    move $v1, $s1
    li $v0, 4
    la $a0, char_g
    syscall
    add $a3, $v1, $zero
    addi $a1, $zero, 1
    jal print_return

    # Estrae i valori dallo stack per ripristinare i registri e
ritorna al chiamante
    lw $ra, 4($sp)
    addi $sp, $sp, 8
    jr $ra

# Esce dal programma
exit:
    li $v0, 10
    syscall
```

Esercizio 3 - Operazioni fra matrici

TESTO

Utilizzando QtSpim, scrivere e provare un programma che visualizza all'utente un menù di scelta con le seguenti cinque opzioni *a*, *b*, *c*, *d*, *e*:

- a) **Inserimento di matrici.** Il programma richiede di inserire da tastiera un numero intero $0 < n < 5$, e richiede quindi l'inserimento di due matrici quadrate, chiamate A e B, di dimensione $n \times n$, contenenti numeri interi. Quindi si ritorna al menù di scelta.

Facoltativo. Le matrici A e B dovranno essere allocate dinamicamente in memoria. Si consiglia l'utilizzo della system call 'sbrk' del MIPS.

Ogni volta che si seleziona l'opzione a) del menu, i nuovi valori inseriti di A e B dovranno essere salvati nella stessa area di memoria in cui erano stati salvati i vecchi valori: i nuovi valori sovrascriveranno quelli vecchi.

Facoltativo: Si dovrà allocare (con la 'sbrk') uno spazio aggiuntivo di memoria solo se le due nuove matrici dovessero richiedere più spazio di memoria rispetto a quello già allocato in precedenza.

Esempio di interfaccia per l'inserimento delle due matrici:

Dimensione matrici: 3x3

Matrice A:

Riga1: -2 44 5

Riga2: 1 1 1

Riga3: 3 0 1

Matrice B:

Riga1: 0 0 10

Riga2: -1 1 -1

Riga3: 1 0 0

- b) **Somma di matrici.** Il programma effettua la somma fra le due matrici A e B, e visualizza su console il risultato $A+B$. Quindi si ritorna al menù di scelta.

Ad esempio, se A e B sono state inserite come riportato nell'esempio al punto a), il programma dovrà visualizzare su console:

Risultato di $A+B$:

Riga1: -2 44 15

Riga2: 0 2 0

Riga3: 4 0 1

- c) **Sottrazione di matrici.** Il programma effettua la sottrazione fra le due matrici A e B, e visualizza su console il risultato $A-B$. Quindi si ritorna al menù di scelta.

Ad esempio, se A e B sono state inserite come riportato nell'esempio al punto a), il programma dovrà visualizzare su console:

Risultato di $A-B$:

Riga1: -2 44 -5
Riga2: 2 0 2
Riga3: 2 0 1

- d) **Prodotto di matrici.** Il programma effettua il prodotto fra le due matrici A e B, e visualizza su console il risultato $A*B$. Quindi si ritorna al menù di scelta.

Ad esempio, se A e B sono state inserite come riportato nell'esempio al punto a), il programma dovrà visualizzare su console:

Risultato di $A*B$:
Riga1: -39 44 -64
Riga2: 0 1 9
Riga3: 1 0 30

- e) **Uscita.** Stampa un messaggio di uscita ed esce dal programma.

Alle opzioni *a*, *b*, *c*, *d* corrisponderanno le chiamate alle opportune procedure, e quindi il programma dovrà tornare disponibile per selezionare una nuova opzione. Alla scelta *e* corrisponderà la terminazione del programma.

Mostrare e discutere nella relazione l'evoluzione della memoria dinamica (heap) in alcuni casi di interesse.

DESCRIZIONE DELLA SOLUZIONE

Il programma stampa immediatamente il menù e richiede l'inserimento di una lettera fra *a*, *b*, *c*, *d*, oppure *e* per svolgere la funzione descritta e saltare rispettivamente al blocco di istruzioni identificato da **case_a** per inserire elementi nelle matrici, a quello identificato da **case_b** per effettuare la somma tra matrici, a quello identificato da **case_c** per effettuare la sottrazione tra matrici, a quello identificato da **case_d** per effettuare il prodotto tra matrici oppure a quello identificato da **exit** per uscire dal programma. Nel caso in cui l'utente provasse a inserire un carattere non presente nel menù il programma ristamperebbe l'intero menù e chiederebbe di inserire nuovamente una lettera corretta. Descriviamo di seguito le operazioni svolte dai vari "case":

- **case_a:** Prima di tutto viene richiesto l'inserimento della dimensione delle matrici (la richiesta è unica in quanto le matrici considerate sono quadrate e della stessa dimensione), fino a che il numero naturale n inserito non risulta compreso tra 0 e 5. Dopo aver salvato la dimensione delle matrici nel registro **\$s0**, è necessario calcolare il numero totale di elementi delle matrici, che è pari a $n \times n$ e viene salvato in **\$s1**, e l'occupazione totale delle due matrici, che è invece pari a $n \times n \times 4$ e viene salvata in **\$t0**. Questi calcoli, e in particolare l'ultimo, vengono utilizzati per allocare dinamicamente le due matrici, utilizzando la chiamata di sistema "sbrk", i cui indirizzi vengono salvati nei registri **\$s2** e **\$s3**. A questo punto, grazie alle istruzioni identificate dalle etichette **fill** e **insert**, all'utente viene richiesto di inserire $n \times n$ elementi per la prima matrice e, dopo averla stampata, lo stesso procedimento viene ripetuto per la seconda matrice. I registri di appoggio utilizzati per queste operazioni sono **\$t2**, che viene utilizzato per scorrere le matrici, **\$t1**, che contiene il numero di elementi già inseriti, **\$t3**, che contiene il numero n di elementi presenti in ogni riga, e **\$t7**, che contiene un valore utilizzato per distinguere l'inserimento degli elementi della prima matrice da quello degli elementi della seconda. Una volta completata la procedura di inserimento il programma ritorna alla visualizzazione del menù di scelta.

- **case_b**: Prima di tutto vengono inizializzati i registri di appoggio necessari alla procedura, ovvero **\$t1** per contare il numero di elementi già processati, **\$t3** per contare il numero di elementi già processati per ogni riga, **\$t2** per puntare alla prima matrice, il cui indirizzo è in **\$s2**, e **\$t4** per puntare alla seconda matrice, il cui indirizzo è in **\$s3**. A questo punto, con il blocco di istruzioni identificato da **sum_row** viene estratto un elemento dalla prima matrice e salvato in **\$a0**, viene estratto un elemento dalla seconda matrice e salvato in **\$a1** e viene salvata in **\$a0** la somma di **\$a0** e **\$a1**. Dopo aver stampato il contenuto del registro **\$a0** è possibile incrementare i puntatori e i contatori sopra-descritti e controllare se sono stati processati tutti gli elementi, e in tal caso con un salto a **end_sum** terminerebbe la procedura di somma; oppure se sono stati processati tutti gli elementi della riga corrente, e in tal caso con un salto a **new_sum** si andrebbe a stampare un carattere di “a capo” per poi tornare a svolgere le somme; oppure, se le due condizioni sopra-descritte non fossero verificate sarebbe necessario continuare a sommare gli elementi della riga corrente. Una volta completata la procedura di somma, ovvero quando si effettua il salto a **end_sum**, il programma ritorna alla visualizzazione del menù di scelta.
- **case_c**: Il principio di funzionamento è totalmente identico a quello descritto nel **case_b**, ad eccezione dell'operazione effettuata tra i valori contenuti nei registri **\$a0** e **\$a1**, che viene sostituita con una sottrazione. Inoltre, all'etichetta **sum_row** viene sostituita l'etichetta **sub_row**, a **new_sum** viene sostituita **new_sub** e a **end_sum** viene sostituita **end_sub**. Una volta completata la procedura di sottrazione, ovvero quando si effettua il salto a **end_sub**, il programma ritorna alla visualizzazione del menù di scelta.
- **case_d**: Descriviamo le funzionalità principali della procedura, suddividendole in base all'etichetta:
 - **multiply**: Vengono estratti i due valori puntati dalla prima e dalla seconda matrice e moltiplicati fra di loro. Il prodotto viene aggiunto alla somma parziale relativa all'elemento esaminato, contenuta nel registro **\$t7**. Dopodiché vengono incrementati i puntatori e i contatori di appoggio e se sono state fatte tutte le iterazioni necessarie a calcolare il valore dell'elemento esaminato (n iterazioni) si salta all'etichetta **next_element**, altrimenti si continua con il processo di moltiplicazione.
 - **next_element**: Prima di tutto viene stampato l'elemento processato dalle appena terminate n iterazioni della procedura **multiply**. Dopodiché vengono azzerati i registri di appoggio utilizzati in **multiply** e viene calcolato l'offset necessario a posizionare correttamente i puntatori agli elementi delle due matrici, per procedere con l'analisi dell'elemento successivo. In particolare viene calcolato lo scostamento necessario a posizionare il puntatore alla seconda matrice alla giusta colonna. A questo punto, se sono state effettuate tutte le iterazioni ($n \times n \times n$) il programma salta a **end_mul**, mentre se abbiamo completato una riga della matrice prodotto, il programma salta a **next_row**, altrimenti salta a **multiply**.
 - **next_row**: Viene utilizzato il numero di riga corrente, salvato in **\$t9**, per calcolare l'incremento necessario a cambiare riga, salvato in **\$t8**, partendo dall'inizio della prima matrice ($\$t8 = \$t9 * n * 4$). Dopo aver aggiunto l'offset calcolato al puntatore alla prima matrice è possibile iniziare nuovamente il processo di **multiply**.
 - **end_mul**: Il programma ritorna alla visualizzazione del menù di scelta.
- **exit**: Stampa un messaggio d'uscita, termina l'esecuzione del programma e ritorna al sistema operativo.

SIMULAZIONE

1. Simuliamo su QtSpim, utilizzando l'esempio fornito dal testo dell'esercizio:

- Il programma stampa il menù e attende l'inserimento di una scelta fra a , b , c , d , e

```

a) Inserimento di matrici
b) Somma di matrici
c) Sottrazione di matrici
d) Prodotto di matrici
e) Uscita
|

```

- Scegliendo l'opzione a , all'utente viene chiesto di inserire la dimensione della matrici

```

a) Inserimento di matrici
b) Somma di matrici
c) Sottrazione di matrici
d) Prodotto di matrici
e) Uscita
a

```

Inserire la dimensione delle matrici quadrate: |

- Dopo aver inserito la dimensione n , all'utente viene chiesto di inserire $n \times n$ elementi e viene stampata la prima matrice

```

Inserire la dimensione delle matrici quadrate: 3
Inserire il prossimo elemento: -2
Inserire il prossimo elemento: 44
Inserire il prossimo elemento: 5
Inserire il prossimo elemento: 1
Inserire il prossimo elemento: 1
Inserire il prossimo elemento: 1
Inserire il prossimo elemento: 3
Inserire il prossimo elemento: 0
Inserire il prossimo elemento: 1

```

```

-2 44 5
1 1 1
3 0 1

```

- Dopo aver stampato la prima matrice, all'utente viene chiesto di inserire altri $n \times n$ elementi e viene stampata la seconda matrice

```

Inserire il prossimo elemento: 0
Inserire il prossimo elemento: 0
Inserire il prossimo elemento: 10
Inserire il prossimo elemento: -1
Inserire il prossimo elemento: 1
Inserire il prossimo elemento: -1
Inserire il prossimo elemento: 1
Inserire il prossimo elemento: 0
Inserire il prossimo elemento: 0

```

```

0 0 10
-1 1 -1
1 0 0

```

- A questo punto si ripresenta il menù e scegliendo l'opzione b otteniamo

```

a) Inserimento di matrici
b) Somma di matrici
c) Sottrazione di matrici
d) Prodotto di matrici
e) Uscita
b

```

```

-2 44 15
0 2 0
4 0 1

```

- Scegliendo l'opzione c otteniamo

```

a) Inserimento di matrici
b) Somma di matrici
c) Sottrazione di matrici
d) Prodotto di matrici
e) Uscita
c

```

```

-2 44 -5
2 0 2
2 0 1

```

- Scegliendo l'opzione *d* otteniamo

```
a) Inserimento di matrici
b) Somma di matrici
c) Sottrazione di matrici
d) Prodotto di matrici
e) Uscita
d
```

```
-39 44 -64
0 1 9
1 0 30
```

- Infine, scegliendo l'opzione *e* otteniamo

```
a) Inserimento di matrici
b) Somma di matrici
c) Sottrazione di matrici
d) Prodotto di matrici
e) Uscita
e
```

```
Programma terminato.
```

2. Simuliamo su QtSpim, mostrando l'evoluzione delle memoria dinamica (heap):

- Nel caso dell'inserimento degli elementi mostriamo l'evoluzione della memoria dinamica, tramite l'indirizzo con il relativo contenuto

```
Inserire la dimensione delle matrici quadrate: 2
Inserire il prossimo elemento: 5
```

```
[10040000] 0000000005 0000000000 0000000000 0000000000
[10040010]..[1004001f] 0000000000
```

```
Inserire la dimensione delle matrici quadrate: 2
Inserire il prossimo elemento: 5
Inserire il prossimo elemento: 6
```

```
[10040000] 0000000005 0000000006 0000000000 0000000000
[10040010]..[1004001f] 0000000000
```

```
Inserire la dimensione delle matrici quadrate: 2
Inserire il prossimo elemento: 5
Inserire il prossimo elemento: 6
Inserire il prossimo elemento: -3
```

```
[10040000] 0000000005 0000000006 -3 0000000000
[10040010]..[1004001f] 0000000000
```

```
Inserire la dimensione delle matrici quadrate: 2
Inserire il prossimo elemento: 5
Inserire il prossimo elemento: 6
Inserire il prossimo elemento: -3
Inserire il prossimo elemento: 14
```

```
[10040000] 0000000005 0000000006 -3 0000000014
[10040010]..[1004001f] 0000000000
```

```
Inserire la dimensione delle matrici quadrate: 2
Inserire il prossimo elemento: 5
Inserire il prossimo elemento: 6
Inserire il prossimo elemento: -3
Inserire il prossimo elemento: 14
```

```
5 6
-3 14
```

```
Inserire il prossimo elemento: -2
```

```
[10040000] 0000000005 0000000006 -3 0000000014
[10040010] -2 0000000000 0000000000 0000000000
```

```

Inserire la dimensione delle matrici quadrate: 2
Inserire il prossimo elemento: 5
Inserire il prossimo elemento: 6
Inserire il prossimo elemento: -3
Inserire il prossimo elemento: 14

5 6
-3 14

Inserire il prossimo elemento: -2
Inserire il prossimo elemento: 9
[10040000] 0000000005 0000000006      -3 0000000014
[10040010]      -2 0000000009 0000000000 0000000000

```

```

Inserire la dimensione delle matrici quadrate: 2
Inserire il prossimo elemento: 5
Inserire il prossimo elemento: 6
Inserire il prossimo elemento: -3
Inserire il prossimo elemento: 14

5 6
-3 14

Inserire il prossimo elemento: -2
Inserire il prossimo elemento: 9
Inserire il prossimo elemento: 12
[10040000] 0000000005 0000000006      -3 0000000014
[10040010]      -2 0000000009 0000000012 0000000000

```

```

Inserire la dimensione delle matrici quadrate: 2
Inserire il prossimo elemento: 5
Inserire il prossimo elemento: 6
Inserire il prossimo elemento: -3
Inserire il prossimo elemento: 14

5 6
-3 14

Inserire il prossimo elemento: -2
Inserire il prossimo elemento: 9
Inserire il prossimo elemento: 12
Inserire il prossimo elemento: 21
[10040000] 0000000005 0000000006      -3 0000000014
[10040010]      -2 0000000009 0000000012 0000000021

```

3. Simuliamo su QtSpim, mostrando il funzionamento dell'algoritmo in corrispondenza di input sbagliati:

- Se non si sono ancora inseriti valori all'interno delle due matrici, qualunque opzione venga inserita il programma salta alla procedura di inserimento

```

Console
a) Inserimento di matrici
b) Somma di matrici
c) Sottrazione di matrici
d) Prodotto di matrici
e) Uscita
t
Inserire la dimensione delle matrici quadrate:

```

- Finché non si inserisce un valore compreso tra 0 e 5 per la dimensione delle matrici, il programma chiederà di inserire un nuovo valore che soddisfi tali requisiti

```

Inserire la dimensione delle matrici quadrate: 8
Inserire la dimensione delle matrici quadrate: 9
Inserire la dimensione delle matrici quadrate: 3
Inserire il prossimo elemento: |

```

- Se invece le due matrici sono già state riempite e l'opzione inserita non corrisponde a nessuna opzione del menù, il programma ristampa il menù e richiede l'inserimento di una nuova scelta

```
a) Inserimento di matrici
b) Somma di matrici
c) Sottrazione di matrici
d) Prodotto di matrici
e) Uscita
m
```

```
a) Inserimento di matrici
b) Somma di matrici
c) Sottrazione di matrici
d) Prodotto di matrici
e) Uscita
|
```

- Ogni carattere inserito come elemento, al posto di un numero, verrà interpretato dal programma come uno zero

```
Inserire il prossimo elemento: b
Inserire il prossimo elemento: c
Inserire il prossimo elemento: d
Inserire il prossimo elemento: e
Inserire il prossimo elemento: f
Inserire il prossimo elemento: g
Inserire il prossimo elemento: h
Inserire il prossimo elemento: i
Inserire il prossimo elemento: l
```

```
0 0 0
0 0 0
0 0 0
```

CODICE MIPS

```
# Esercizio 3
# Nome: Alessio Falai
# Matricola: 6134275
# Email: alessio.falai@stud.unifi.it
# Data di consegna: 02/07/2017

.data
    str_dim: .asciiz "Inserire la dimensione delle matrici
quadrate: "
    str_insert: .asciiz "Inserire il prossimo elemento: "
    str_exit: .asciiz "Programma terminato."
    str_menu: .asciiz "a) Inserimento di matrici\nb) Somma di
matrici\nc) Sottrazione di matrici\nd) Prodotto di matrici\ne)
Uscita\n"
    newline: .asciiz "\n"
    double_newline: .asciiz "\n\n"
    space: .asciiz " "

.text
.globl main
main:
    # Stampa il menù
    la $a0, str_menu
    li $v0, 4
    syscall

    # Chiamata di sistema per la lettura di un carattere
    li $v0, 12
    syscall

    add $s5, $zero, $zero # Registro per il controllo della
dimensione delle matrici

    beq $s0, $zero, case_a # Se le due matrici sono vuote, salta
all'inserimento
    beq $v0, 97, case_a     # Inserimento delle matrici
    beq $v0, 98, case_b     # Somma delle matrici
    beq $v0, 99, case_c     # Sottrazione delle matrici
    beq $v0, 100, case_d    # Prodotto delle matrici
    beq $v0, 101, exit      # Esci dal programma

    # Stampa due linee bianche
    la $a0, double_newline
    li $v0, 4
    syscall

    j main # Se il carattere inserito non è tra quelli presenti
nel menù, ristampa tutto

# Inserimento delle matrici
case_a:
```

```
# Stampa due linee bianche
la $a0, double_newline
li $v0, 4
syscall

# Inserimento della dimensione delle matrici
matrix_dimension:
    # Stampa la stringa per l'inserimento della dimensione
    delle matrici
    la $a0, str_dim
    li $v0, 4
    syscall

    # Salva la vecchia dimensione delle matrici
    move $s5, $s0

    # Chiamata di sistema per leggere un intero (Dimensione
    delle matrici)
    li $v0, 5
    syscall

    # Controllo della dimensione ( $0 < n < 5$ ), salvata in $s0
    ble $v0, 0, matrix_dimension
    bgt $v0, 4, matrix_dimension
    move $s0, $v0

    mul $s1, $s0, $s0    # $s1 contiene il numero di elementi delle
    matrici
    mul $t0, $s1, 4      # $t0 contiene la dimensione totale (in
    word) della matrice

    beq $s5, $s0, allocated

# Allocazione dinamica (sbrk) delle due matrici
alloc:
    li $v0, 9
    move $a0, $t0
    syscall
    move $s2, $v0

    li $v0, 9
    move $a0, $t0
    syscall
    move $s3, $v0

# Memoria già allocata, procede al caricamento delle matrici
allocated:
    add $t7, $zero, $zero    # $t7 viene utilizzato per
    distinguere l'inserimento della prima matrice dalla seconda
    move $t2, $s2            # $t2 viene utilizzato per scorrere
    gli elementi delle matrici
    fill:
```



```

    add $t1, $zero, $zero # $t1 conta il numero di
    elementi inseriti
    addi $t7, $t7, 1
    insert:
        # Stampa la stringa di inserimento di un elemento
        la $a0, str_insert
        li $v0, 4
        syscall

        # Inserimento di un elemento da tastiera
        li $v0, 5
        syscall

        sw $v0, 0($t2) # Salva il valore inserito
nella matrice
        addi $t2, $t2, 4 # Seleziona la posizione
successiva della matrice
        addi $t1, $t1, 1 # Incremento il numero di
elementi inseriti
        bne $t1, $s1, insert # Inserisco finchè la
matrice non è piena

        # Stampa una linea vuota
        la $a0, newline
        li $v0, 4
        syscall

        # Inizializza i registri per la stampa della matrice
        add $t1, $zero, $zero # $t1 contiene il
numero di elementi stampati
        add $t3, $zero, $zero # $t3 contiene il
numero di elementi presenti in ogni riga
        move $t2, $s2 # $t2 contiene il
puntatore alla matrice
        beq $t7, 2, select_second_matrix # Controlla quale
delle due matrici si deve stampare
        j print_row

        # Seleziona la seconda matrice per la stampa
        select_second_matrix:
            move $t2, $s3

        # Stampa una riga della matrice
        print_row:
            lw $a0, 0($t2)
            li $v0, 1
            syscall
            la $a0, space
            li $v0, 4
            syscall

            addi $t1, $t1, 1

```

```

        addi $t3, $t3, 1
        addi $t2, $t2, 4
        beq $t1, $s1, end_pr    # Se il numero di elementi
stampati è uguale al numero di elementi totali, esci
        beq $t3, $s0, new_row   # Se la riga attuale è stata
completamente stampata, seleziona la successiva
        j print_row            # Altrimenti continua a
stampare gli elementi della riga attuale

        # Seleziona una nuova riga della matrice
new_row:
        add $t3, $zero, $zero
        la $a0, newline
        li $v0, 4
        syscall
        j print_row

        # Fine della stampa
end_pr:
        # Stampa due linee bianche
        la $a0, double_newline
        li $v0, 4
        syscall

        beq $t7, 2, main    # Se si sono processate
entrambe le matrici torna al main
        move $t2, $s3      # Altrimenti carica l'indirizzo
della seconda matrice in $t2
        j fill              # E salta alla procedura di
inserimento degli elementi

# Somma delle matrici
case_b:
        # Stampa due linee bianche
        la $a0, double_newline
        li $v0, 4
        syscall

        add $t1, $zero, $zero    # $t1 contiene il numero totale di
elementi processati
        add $t3, $zero, $zero    # $t3 contiene il numero di
elementi processati per ogni riga
        move $t2, $s2            # $t2 contiene il puntatore alla
prima matrice
        move $t4, $s3            # $t4 contiene il puntatore alla
seconda matrice

        # Calcola la somma fra elementi delle due matrici
sum_row:
        # Somma i due elementi e stampa il risultato
        lw $a0, 0($t2)
        lw $a1, 0($t4)

```

```
    add $a0, $a0, $a1
    li $v0, 1
    syscall
    la $a0, space
    li $v0, 4
    syscall

    addi $t1, $t1, 1      # Incrementa il numero totale di
elementi processati
    addi $t3, $t3, 1      # Incrementa il numero di elementi
processati nella riga corrente
    addi $t2, $t2, 4      # Punta all'elemento successivo
della prima matrice
    addi $t4, $t4, 4      # Punta all'elemento successivo
della seconda matrice
    beq $t1, $s1, end_sum # Se sono stati processati tutti
gli elementi, esci
    beq $t3, $s0, new_sum # Se sono stati processati tutti
gli elementi della riga corrente, seleziona la riga successiva
    j sum_row             # Altrimenti continua a lavorare
sulla stessa riga

# Seleziona una nuova riga delle due matrici
new_sum:
    add $t3, $zero, $zero # Inizializza il contatore del
numero di elementi processati della nuova riga
    la $a0, newline
    li $v0, 4
    syscall
    j sum_row

# Fine della somma
end_sum:
    la $a0, double_newline
    li $v0, 4
    syscall
    j main

# Sottrazione delle matrici
case_c:
    # Stampa due linee bianche
    la $a0, double_newline
    li $v0, 4
    syscall

    add $t1, $zero, $zero      # $t1 contiene il numero totale di
elementi processati
    add $t3, $zero, $zero      # $t3 contiene il numero di
elementi processati per ogni riga
    move $t2, $s2              # $t2 contiene il puntatore alla
prima matrice
```

```
    move $t4, $s3                # $t4 contiene il puntatore alla
seconda matrice

# Calcola la differenza fra elementi delle due matrici
sub_row:
    # Sottrae i due elementi e stampa il risultato
    lw $a0, 0($t2)
    lw $a1, 0($t4)
    sub $a0, $a0, $a1
    li $v0, 1
    syscall
    la $a0, space
    li $v0, 4
    syscall

    addi $t1, $t1, 1             # Incrementa il numero totale di
elementi processati
    addi $t3, $t3, 1             # Incrementa il numero di elementi
processati nella riga corrente
    addi $t2, $t2, 4             # Punta all'elemento successivo
della prima matrice
    addi $t4, $t4, 4             # Punta all'elemento successivo
della seconda matrice
    beq $t1, $s1, end_sub        # Se sono stati processati tutti
gli elementi, esci
    beq $t3, $s0, new_sub        # Se sono stati processati tutti
gli elementi della riga corrente, seleziona la riga successiva
    j sub_row                    # Altrimenti continua a lavorare
sulla stessa riga

# Seleziona una nuova riga delle due matrici
new_sub:
    add $t3, $zero, $zero        # Inizializza il contatore del
numero di elementi processati della nuova riga
    la $a0, newline
    li $v0, 4
    syscall
    j sub_row

# Fine della sottrazione
end_sub:
    la $a0, double_newline
    li $v0, 4
    syscall
    j main

# Prodotto delle matrici
case_d:
    # Stampa due linee bianche
    la $a0, double_newline
    li $v0, 4
    syscall
```

```

    add $s4, $zero, $zero # $s4 contiene il numero di iterazioni
    necessarie a cambiare riga
    add $t0, $zero, $zero
    mul $t0, $s0, $s1      # $t0 contiene il numero di iterazioni
    (n x n x n)
    add $t1, $zero, $zero # $t1 conta il numero totale di
    iterazioni (fino ad arrivare a $t0)
    add $t3, $zero, $zero # $t3 conta il numero di passi
    necessari a calcolare ogni elemento
    add $t5, $zero, $zero # $t5 mantiene il conto del cambio di
    colonna
    add $t6, $zero, $zero
    mul $t6, $s0, 4        # $t6 viene utilizzato per il movimento
    sulla colonna oppure per il cambio di riga
    add $t7, $zero, $zero # $t7 contiene la somma parziale di
    ogni elemento della matrice prodotto
    add $t8, $zero, $zero # $t8 è un registro di appoggio, che
    viene utilizzato per salvare risultati di operazioni
    addi $t9, $zero, 0     # $t9 contiene il numero di riga
    corrente
    move $t2, $s2          # $t2 contiene il puntatore alla prima
    matrice
    move $t4, $s3          # $t4 contiene il puntatore alla
    seconda matrice

    # Calcola il prodotto tra una riga della prima matrice e una
    colonna della seconda matrice
    multiply:
        lw $a0, 0($t2)     # $a0 contiene il valore estratto dalla
    prima matrice
        lw $a1, 0($t4)     # $a1 contiene il valore estratto dalla
    seconda matrice
        mul $a0, $a0, $a1  # $a0 contiene il prodotto tra i due
    elementi considerati
        add $t7, $t7, $a0  # $t7 viene aggiornato, aggiungendo il
    prodotto appena calcolato

        addi $s4, $s4, 1   # Incrementa il contatore del numero di
    iterazioni necessarie a cambiare riga
        addi $t1, $t1, 1   # Incrementa il numero di iterazioni
    effettuate
        addi $t3, $t3, 1   # Incrementa il numero di passi
    effettuati per l'elemento attuale
        addi $t2, $t2, 4   # Incrementa il puntatore della prima
    matrice (Prossimo elemento della stessa riga)
        add $t4, $t4, $t6  # Incrementa il puntatore della seconda
    matrice (Prossimo elemento della stessa colonna)

        beq $t3, $s0, next_element # Se l'elemento è stato
    completamente processato si passa al successivo

```

```
        j multiply                # Altrimenti si continua con il
calcolo della somma parziale

    # Prepara le matrici in ingresso per calcolare l'elemento
    successivo della matrice prodotto
    next_element:
        # Stampa l'elemento processato
        move $a0, $t7
        li $v0, 1
        syscall
        la $a0, space
        li $v0, 4
        syscall

        addi $t5, $t5, 4          # Incrementa il contatore di cambio
di colonna
        add $t3, $zero, $zero    # Azzera il numero di passi
effettuati per l'elemento attuale
        add $t7, $zero, $zero    # Azzera la somma parziale
        move $t2, $s2            # Ricarica il puntatore iniziale
della prima matrice
        move $t4, $s3            # Ricarica il puntatore iniziale
della seconda matrice
        mul $t8, $t9, $t6        # Calcola l'offset necessario a
selezionare la giusta riga
        add $t2, $t2, $t8        # Aggiunge lo scostamento calcolato
al puntatore alla prima matrice
        add $t4, $t4, $t5        # Aggiunge l'offset necessario a
selezionare la giusta colonna al puntatore alla seconda matrice

        beq $t1, $t0, end_mul    # Se sono state effettuate tutte le
iterazioni, esci
        beq $s4, $s1, next_row   # Controlla se è necessario
cambiare riga
        j multiply                # Altrimenti continua a lavorare
sulla stessa riga

    # Seleziona la riga successiva della prima matrice
    next_row:
        # Stampa due linee bianche
        la $a0, newline
        li $v0, 4
        syscall

        add $s4, $zero, $zero    # Azzera il numero di iterazioni
necessarie a cambiare riga
        add $t5, $zero, $zero    # Azzera il contatore del cambio di
colonna
        addi $t9, $t9, 1         # Incrementa il numero di riga
        mul $t8, $t9, $t6        # Calcola l'incremento necessario a
cambiare riga
```

```
        move $t2, $s2           # Ricarica il puntatore iniziale
della prima matrice
        add $t2, $t2, $t8       # Passa alla riga successiva
        move $t4, $s3           # Ricarica il puntatore iniziale
della seconda matrice

        j multiply              # Inizia a calcolare il prodotto,
partendo dalla nuova riga

# Fine della moltiplicazione
end_mul:
        la $a0, double_newline
        li $v0, 4
        syscall
        j main

# Esci dal programma
exit:
        la $a0, double_newline
        li $v0, 4
        syscall
        la $a0, str_exit
        li $v0, 4
        syscall

        li $v0, 10
        syscall
```