

# Relazione progetto AE 2016-2017

Autori:

- Federico Moncini, 5936828, [federico.moncini@stud.unifi.it](mailto:federico.moncini@stud.unifi.it)
- Lorenzo Mungai, 5962693, [lorenzo.mungai@stud.unifi.it](mailto:lorenzo.mungai@stud.unifi.it)
- Tommaso Capecchi, 5943118, [tommaso.capecchi@stud.uni.it](mailto:tommaso.capecchi@stud.uni.it)

-----20/05/2017-----

## 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 sottosequenza di caratteri separati da almeno uno spazio (nell'esempio, le sottosequenze "uno", "due", "ciao", "tree", "tre", "uno", "Uno", "di", "eci") applicando le seguenti regole:

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

## Descrizione della soluzione:

La nostra soluzione prevede un array lungo 100 caratteri massimo. L'utente deve passare in input una serie di caratteri che andranno a popolare l'array. Ogni word è separata da uno spazio vuoto. L'inserimento avviene nella procedura main. Inoltre viene inizializzato un contatore nel registro \$t0 per effettuare un ciclo while per scorrere tutti i caratteri dell'array. Ogni carattere viene confrontato grazie ad uno switch statement che ha il seguente funzionamento:

- Se il carattere è uno spazio vuoto, il contatore viene incrementato fino al carattere successivo diverso da uno spazio.
- Se il carattere è uguale a 'u' lo switch ci rimanda nel case1 dove verranno effettuati dei controlli successivi per verificare che la word corrisponda a 'uno'; in tal caso controlla che non sia seguita da altri caratteri e infine viene stampato il valore 1; altrimenti verrà chiamata la funzione "notFound" che stamperà tramite la relativa procedura il carattere '?'.  
?
- Se il carattere è uguale a 'd' lo switch ci rimanda nel case2 dove verranno effettuati dei controlli successivi per verificare che la word corrisponda a 'due'; in tal caso controlla che non sia seguita da altri caratteri e infine viene stampato il valore 2; altrimenti verrà stampato il carattere '?'.  
?
- Se il carattere è uguale a 'n' lo switch ci rimanda nel case3 dove verranno effettuati dei controlli successivi per verificare che la word corrisponda a 'nove'; in tal caso controlla che non sia seguita da altri caratteri e infine viene stampato il valore 9, altrimenti '?'.  
?

Nel caso in cui il carattere trovato sia diverso da i casi precedentemente descritti, si stamperà il punto di domanda e si incrementa il contatore fino a che non verrà trovata una nuova word. Una volta arrivato alla fine dell'array verrà chiamata la procedura di uscita e il programma terminerà.

Abbiamo deciso di non implementare una stack perché non avevamo bisogno di preservare il valore precedente ad una chiamata successiva.

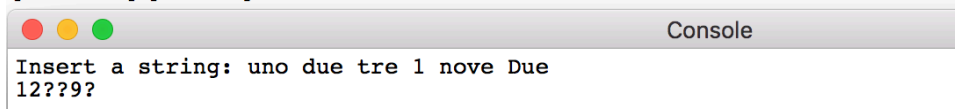
# Simulazione:

Stato iniziale dell' user data prima dell'inserimento degli input.

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff] 00000000
[10010000] 65736e49 61207472 72747320 3a676e69 Insert a string:
[10010010] 00000020 00000000 00000000 00000000 .....
[10010020]..[1003ffff] 00000000
```

Esempio di inserimento e relativo output finale.

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff] 00000000
[10010000] 65736e49 61207472 72747320 3a676e69 Insert a string:
[10010010] 6e750020 7564206f 72742065 20312065 .uno due tre 1
[10010020] 65766f6e 65754420 0000000a 00000000 nove Due.....
[10010030]..[1003ffff] 00000000
```



## 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, . . . , n do

Begin

u := F(k)

b := b<sup>2</sup> + u

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  $F$ .

## Descrizione della soluzione:

La nostra soluzione prevede l'utilizzo della stack per preservare i valori delle varie funzioni ricorsive. Inizialmente facciamo un controllo sul valore " $n$ " inserito, che deve essere compreso tra 1 e 8. All'interno della procedura  $G$  allochiamo 12 byte (tre parole) nella stack: nella prima posizione carichiamo l'indirizzo di ritorno, nella seconda salviamo il valore  $b$ , ovvero la variabile da stampare contenente il risultato delle due procedure, mentre nell'ultima posizione ci salviamo la  $n$  data in input. Dopo di che, tramite il ciclo "loop" si richiama la funzione  $F$ ; tramite l'istruzione "`bgt`" (branch greater than) si controlla che il contatore  $k$  sia minore stretto di  $n$ . Se ciò risulta falso si richiama la label "`end`" nella quale si dealloca la stack e ritorna al main. Altrimenti avviene la chiamata della funzione  $F$  tramite l'istruzione "`jal`". Al suo interno è necessario riallocare la stack poiché è una funzione ricorsiva, dunque salviamo l'indirizzo di ritorno e l'input della funzione  $F(n)$ . Se questo valore è uguale a 0, restituisce 1 e ritorna all'indirizzo chiamante (specificato da  $\$ra$ ) altrimenti si effettua l'operazione " $2 * F(n - 1) + n$ " che necessita l'istruzione "`mflo $t4`" per salvare il prodotto nel registro  $\$t4$ . Infine si ritorna alla funzione  $G$ , salviamo nel registro  $\$t2$  il valore di ritorno della funzione  $F$  (contenuto in

\$v0) necessario ad effettuare l'operazione "b^2 + u". Alla fine del ciclo viene restituito il valore b.

## Simulazione:

Stato iniziale:

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC = 0  
EPC = 0  
Cause = 0  
BadVAddr = 0  
Status = 3000fff10  
HI = 0  
LO = 0  
R0 [r0] = 0  
R1 [at] = 0  
R2 [v0] = 0  
R3 [v1] = 0  
R4 [a0] = 1  
R5 [a1] = 7ffff7e0  
R6 [a2] = 7ffff7e8  
R7 [a3] = 0  
R8 [t0] = 0  
R9 [t1] = 0  
R10 [t2] = 0  
R11 [t3] = 0  
R12 [t4] = 0  
R13 [t5] = 0  
R14 [t6] = 0  
R15 [t7] = 0  
R16 [s0] = 0  
R17 [s1] = 0  
R18 [s2] = 0  
R19 [s3] = 0  
R20 [s4] = 0  
R21 [s5] = 0  
R22 [s6] = 0

User data segment [10000000]..[10040000]  
[10000000]..[1000ffff] 00000000  
[10010000] 70206c49 72676f72 616d6d61 6c616320 I l p r o g r a m m a c a l  
[10010010] 616c6f63 73206e20 6e6f6365 6c206f64 c o l a n s e c o n d o l  
[10010020] 72702065 6465636f 20657275 696e6e61 e p r o c e d u r e a n n i  
[10010030] 65746164 49000a0a 7265736e 69637369 d a t e . . . I n s e r i s c i  
[10010040] 0a206e20 00000000 00000000 00000000 n . . . . .  
[10010050]..[1003ffff] 00000000

User Stack [7ffff7dc]..[80000000]  
[7ffff7dc] 00000001  
[7ffff7e0] 7ffff89f 00000000 7fffffe1 7fffffbb . . . . .  
[7ffff7f0] 7fffffa2 7fffffb6 7fffff2f 7ffffefe . . . . k . . . / . . . .  
[7ffff800] 7fffffee1 7ffffebd 7ffffe8c 7ffffe64 . . . . . d . . .  
[7ffff810] 7fffffe31 7ffffe24 7ffffe0f 7ffffde6 1 . . . \$ . . .  
[7ffff820] 7ffffdc8 7ffffd7c 7ffffd65 7ffffd45 . . . . | . . . e . . . E . . .  
[7ffff830] 7ffffd37 7ffffc4e 7ffffc10 7ffffbf5 7 . . . N . . . . .  
[7ffff840] 7ffffbd8 7ffffb90 7ffffb7e 7ffffb66 . . . . . f . . .  
[7ffff850] 7ffffb4b 7ffffb27 7ffffafe 7ffffae0 K . . . ' . . . .  
[7ffff860] 7ffffa75 7ffffa5e 7ffffa4a 7ffffa3b u . . . ^ . . . J . . . ; . . .  
[7ffff870] 7ffffa25 7ffff9ff 7ffff9da 7ffff9bf % . . . . .  
[7ffff880] 7ffff995 7ffff987 7ffff96d 7ffff933 . . . . . m . . . 3 . . .  
[7ffff890] 7ffff8e1 7ffff8cf 00000000 43000000 . . . . . C . . . . .  
[7ffff8a0] 73552f3a 2f737265 72657355 776f442f : / U s e r s / U s e r / D o w  
[7ffff8b0] 616f6c6e 452f7364 63726573 6f697a69 n l o a d s / E s e r c i z i o  
[7ffff8c0] 34322d33 30323330 612e3731 77006d73 3 - 2 4 0 3 2 0 1 7 . a s m . w  
[7ffff8d0] 69646e69 3a433d72 6e69575c 73776f64 i n d i r = C : \ W i n d o w s  
[7ffff8e0] 31535600 4f433034 4f544e4d 3d534c4f . V S 1 4 0 C O M M T O O L S =  
[7ffff8f0] 505c3a43 72676f72 46206d61 73656c69 C : \ P r o g r a m F i l e s  
[7ffff900] 38782820 4d5c2936 6f726369 74666f73 ( x 8 6 ) \ M i c r o s o f t  
[7ffff910] 73606e20 206e6175 64757a59 31206f60 W i n d o w s S e r v i c e 1

Dopo aver inserito 1 come input abbiamo salvato nella stack l'indirizzo di ritorno, il valore di b iniziale e la n.

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC = 40009c  
EPC = 0  
Cause = 0  
BadVAddr = 0  
Status = 3000fff10  
HI = 0  
LO = 0  
R0 [r0] = 0  
R1 [at] = 10010000  
R2 [v0] = 1  
R3 [v1] = 0  
R4 [a0] = 1  
R5 [a1] = 7ffff7e8  
R6 [a2] = 7ffff7f0  
R7 [a3] = 0  
R8 [t0] = 0  
R9 [t1] = 8  
R10 [t2] = 0  
R11 [t3] = 1  
R12 [t4] = 0  
R13 [t5] = 0  
R14 [t6] = 0  
R15 [t7] = 0  
R16 [s0] = 0  
R17 [s1] = 0  
R18 [s2] = 0  
R19 [s3] = 0  
R20 [s4] = 0  
R21 [s5] = 0  
R22 [s6] = 0

User data segment [10000000]..[10040000]  
[10000000]..[1000ffff] 00000000  
[10010000] 70206c49 72676f72 616d6d61 6c616320 I l p r o g r a m m a c a l  
[10010010] 616c6f63 73206e20 6e6f6365 6c206f64 c o l a n s e c o n d o l  
[10010020] 72702065 6465636f 20657275 696e6e61 e p r o c e d u r e a n n i  
[10010030] 65746164 49000a0a 7265736e 69637369 d a t e . . . I n s e r i s c i  
[10010040] 0a206e20 00000000 00000000 00000000 n . . . . .  
[10010050]..[1003ffff] 00000000

User Stack [7ffff7d8]..[80000000]  
[7ffff7d8] 00400070 00000000 p . @ . . . . .  
[7ffff7e0] 00000001 00000001 7ffff8a8 00000000 . . . . .  
[7ffff7f0] 7fffffe1 7fffffbb 7fffffa2 7fffffb6 . . . . . k . . .  
[7ffff800] 7fffff2f 7ffffefe 7ffffee1 7ffffebd / . . . . .  
[7ffff810] 7ffffe8c 7ffffe64 7ffffe31 7ffffe24 . . . . . d . . . 1 . . . \$ . . .  
[7ffff820] 7ffffdc8 7ffffd7c 7ffffd65 7ffffd45 . . . . . e . . . E . . . 7 . . . N . . .  
[7ffff830] 7ffffbd8 7ffffb90 7ffffb7e 7ffffb66 . . . . . f . . . . .  
[7ffff840] 7ffffb4b 7ffffb27 7ffffafe 7ffffae0 . . . . . K . . . . .  
[7ffff850] 7ffffa75 7ffffa5e 7ffffa4a 7ffffa3b . . . . . u . . . . ^ . . . .  
[7ffff860] 7ffffa25 7ffff9ff 7ffff9da 7ffff9bf J . . . . . % . . . . .  
[7ffff870] 7ffff995 7ffff987 7ffff96d 7ffff933 . . . . . m . . . . 3 . . . . .  
[7ffff880] 7ffff8e1 7ffff8cf 00000000 43000000 . . . . . C : / U s e r s  
[7ffff890] 657352f2 6442f72 6f6c6e77 2f736461 / U s e r / D o w n l o a d s /  
[7ffff8a0] 72657345 697a6963 612e326f 77006d73 E s e r c i z i o 2 . a s m . w  
[7ffff8b0] 69646e69 3a433d72 6e69575c 73776f64 i n d i r = C : \ W i n d o w s  
[7ffff8c0] 31535600 4f433034 4f544e4d 3d534c4f . V S 1 4 0 C O M M T O O L S =  
[7ffff8d0] 505c3a43 72676f72 46206d61 73656c69 C : \ P r o g r a m F i l e s  
[7ffff8e0] 38782820 4d5c2936 6f726369 74666f73 ( x 8 6 ) \ M i c r o s o f t  
[7ffff8f0] 73606e20 206e6175 64757a59 31206f60 W i n d o w s S e r v i c e 1

F(n) dove n è uguale a 0:

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC = 40011c  
EPC = 0  
Cause = 0  
BadVAddr = 0  
Status = 3000fff10  
HI = 0  
LO = 0  
R0 [r0] = 0  
R1 [at] = 0  
R2 [v0] = 1  
R3 [v1] = 0  
R4 [a0] = 0  
R5 [a1] = 7ffff7e8  
R6 [a2] = 7ffff7f0  
R7 [a3] = 0  
R8 [t0] = 0  
R9 [t1] = 8  
R10 [t2] = 0  
R11 [t3] = 1  
R12 [t4] = 0  
R13 [t5] = 0  
R14 [t6] = 0  
R15 [t7] = 0  
R16 [s0] = 0  
R17 [s1] = 0  
R18 [s2] = 0  
R19 [s3] = 0  
R20 [s4] = 0  
R21 [s5] = 0  
R22 [s6] = 0  
R23 [s7] = 0  
R24 [s8] = 0  
R25 [s9] = 0  
R26 [s10] = 0  
R27 [k1] = 0

User data segment [10000000]..[10040000]  
[10000000]..[1000ffff] 00000000  
[10010000] 70206c49 72676f72 616d6d61 6c616320 I l p r o g r a m m a c a l  
[10010010] 616c6f63 73206e20 6e6f6365 6c206f64 c o l a n s e c o n d o l  
[10010020] 72702065 6465636f 20657275 696e6e61 e p r o c e d u r e a n n i  
[10010030] 65746164 49000a0a 7265736e 69637369 d a t e . . . I n s e r i s c i  
[10010040] 0a206e20 00000000 00000000 00000000 n . . . . .  
[10010050]..[1003ffff] 00000000

User Stack [7ffff7d0]..[80000000]  
[7ffff7d0] 004000b0 00000000 00400070 00000000 . . @ . . . . . p . @ . . . . .  
[7ffff7e0] 00000001 00000001 7ffff7a8 00000000 . . . . . k . . . . .  
[7ffff7f0] 7ffff7f1 7ffff7fb 7ffff7a2 7ffff7b6 / . . . . . l . . . \$ . . . . .  
[7ffff800] 7ffff7f2 7ffff7fe 7ffff7e1 7ffff7bd . . . . . | . . . . .  
[7ffff810] 7ffff7f3 7ffff7ff 7ffff7e2 7ffff7c4 . . . . . E . . . . 7 . . . N . . .  
[7ffff820] 7ffff7f4 7ffff7ff 7ffff7e3 7ffff7c5 . . . . . f . . . . K . . . . ' . . .  
[7ffff830] 7ffff7f5 7ffff7ff 7ffff7e4 7ffff7c6 . . . . . u . . . . ^ . . . .  
[7ffff840] 7ffff7f6 7ffff7ff 7ffff7e5 7ffff7c7 . . . . . % . . . . .  
[7ffff850] 7ffff7f7 7ffff7ff 7ffff7e6 7ffff7c8 . . . . . 3 . . . . .  
[7ffff860] 7ffff7f8 7ffff7ff 7ffff7e7 7ffff7c9 . . . . . C : / U s e r s  
[7ffff870] 7ffff7f9 7ffff7ff 7ffff7e8 7ffff7ca / U s e r / D o w n l o a d s /  
[7ffff880] 7ffff7fa 7ffff7ff 7ffff7e9 7ffff7cb E s e r c i z i o 2 . a s m . w  
[7ffff890] 7ffff7fb 7ffff7ff 7ffff7ea 7ffff7cc i n d i r = C : \ W i n d o w s  
[7ffff8a0] 7ffff7fc 7ffff7ff 7ffff7eb 7ffff7cd . V S 1 4 0 C O M M T O O L S =  
[7ffff8b0] 7ffff7fd 7ffff7ff 7ffff7ec 7ffff7ce C : \ P r o g r a m F i l e s  
[7ffff8c0] 7ffff7fe 7ffff7ff 7ffff7ed 7ffff7cf ( x 8 6 ) \ M i c r o s o f t  
[7ffff8d0] 7ffff7ff 7ffff7ff 7ffff7ee 7ffff7d0

Copyright 1990-2017 by James Larus.  
All Rights Reserved.  
SPIM is distributed under a BSD license.  
See the file README for a full copyright notice.  
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

R27 [k1] = 0

Stack deallocata in seguito a n = 0:

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC = 4000c8  
EPC = 0  
Cause = 0  
BadVAddr = 0  
Status = 3000fff10  
HI = 0  
LO = 0  
R0 [r0] = 0  
R1 [at] = 0  
R2 [v0] = 1  
R3 [v1] = 0  
R4 [a0] = 0  
R5 [a1] = 7ffff7e8  
R6 [a2] = 7ffff7f0  
R7 [a3] = 0  
R8 [t0] = 1  
R9 [t1] = 8  
R10 [t2] = 1  
R11 [t3] = 0  
R12 [t4] = 0  
R13 [t5] = 0  
R14 [t6] = 0  
R15 [t7] = 0  
R16 [s0] = 0  
R17 [s1] = 0  
R18 [s2] = 0  
R19 [s3] = 0  
R20 [s4] = 0  
R21 [s5] = 0  
R22 [s6] = 0  
R23 [s7] = 0  
R24 [s8] = 0  
R25 [s9] = 0  
R26 [s10] = 0  
R27 [k1] = 0

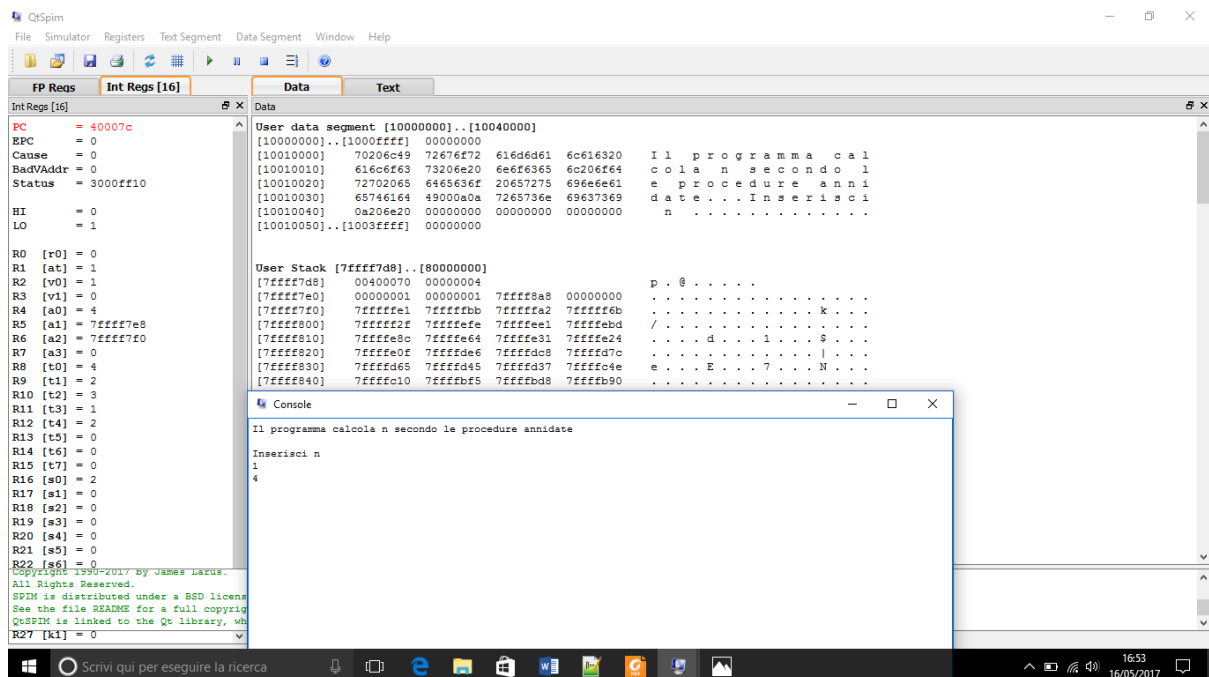
User data segment [10000000]..[10040000]  
[10000000]..[1000ffff] 00000000  
[10010000] 70206c49 72676f72 616d6d61 6c616320 I l p r o g r a m m a c a l  
[10010010] 616c6f63 73206e20 6e6f6365 6c206f64 c o l a n s e c o n d o l  
[10010020] 72702065 6465636f 20657275 696e6e61 e p r o c e d u r e a n n i  
[10010030] 65746164 49000a0a 7265736e 69637369 d a t e . . . I n s e r i s c i  
[10010040] 0a206e20 00000000 00000000 00000000 n . . . . .  
[10010050]..[1003ffff] 00000000

User Stack [7ffff7d8]..[80000000]  
[7ffff7d8] 00400070 00000001 7ffff7a8 00000000 p . @ . . . . .  
[7ffff7e0] 00000001 00000001 7ffff7a8 00000000 . . . . . k . . . . .  
[7ffff7f0] 7ffff7f1 7ffff7fb 7ffff7a2 7ffff7b6 / . . . . . l . . . \$ . . . . .  
[7ffff800] 7ffff7f2 7ffff7fe 7ffff7e1 7ffff7bd . . . . . | . . . . .  
[7ffff810] 7ffff7f3 7ffff7ff 7ffff7e2 7ffff7c4 . . . . . E . . . . 7 . . . N . . .  
[7ffff820] 7ffff7f4 7ffff7ff 7ffff7e3 7ffff7c5 . . . . . f . . . . K . . . . ' . . .  
[7ffff830] 7ffff7f5 7ffff7ff 7ffff7e4 7ffff7c6 . . . . . u . . . . ^ . . . .  
[7ffff840] 7ffff7f6 7ffff7ff 7ffff7e5 7ffff7c7 . . . . . % . . . . .  
[7ffff850] 7ffff7f7 7ffff7ff 7ffff7e6 7ffff7c8 . . . . . 3 . . . . .  
[7ffff860] 7ffff7f8 7ffff7ff 7ffff7e7 7ffff7c9 . . . . . C : / U s e r s  
[7ffff870] 7ffff7f9 7ffff7ff 7ffff7e8 7ffff7ca / U s e r / D o w n l o a d s /  
[7ffff880] 7ffff7fa 7ffff7ff 7ffff7e9 7ffff7cb E s e r c i z i o 2 . a s m . w  
[7ffff890] 7ffff7fb 7ffff7ff 7ffff7ea 7ffff7cc i n d i r = C : \ W i n d o w s  
[7ffff8a0] 7ffff7fc 7ffff7ff 7ffff7eb 7ffff7cd . V S 1 4 0 C O M M T O O L S =  
[7ffff8b0] 7ffff7fd 7ffff7ff 7ffff7ec 7ffff7ce C : \ P r o g r a m F i l e s  
[7ffff8c0] 7ffff7fe 7ffff7ff 7ffff7ed 7ffff7cf ( x 8 6 ) \ M i c r o s o f t  
[7ffff8d0] 7ffff7ff 7ffff7ff 7ffff7ee 7ffff7d0

Copyright 1990-2017 by James Larus.  
All Rights Reserved.  
SPIM is distributed under a BSD license.  
See the file README for a full copyright notice.  
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

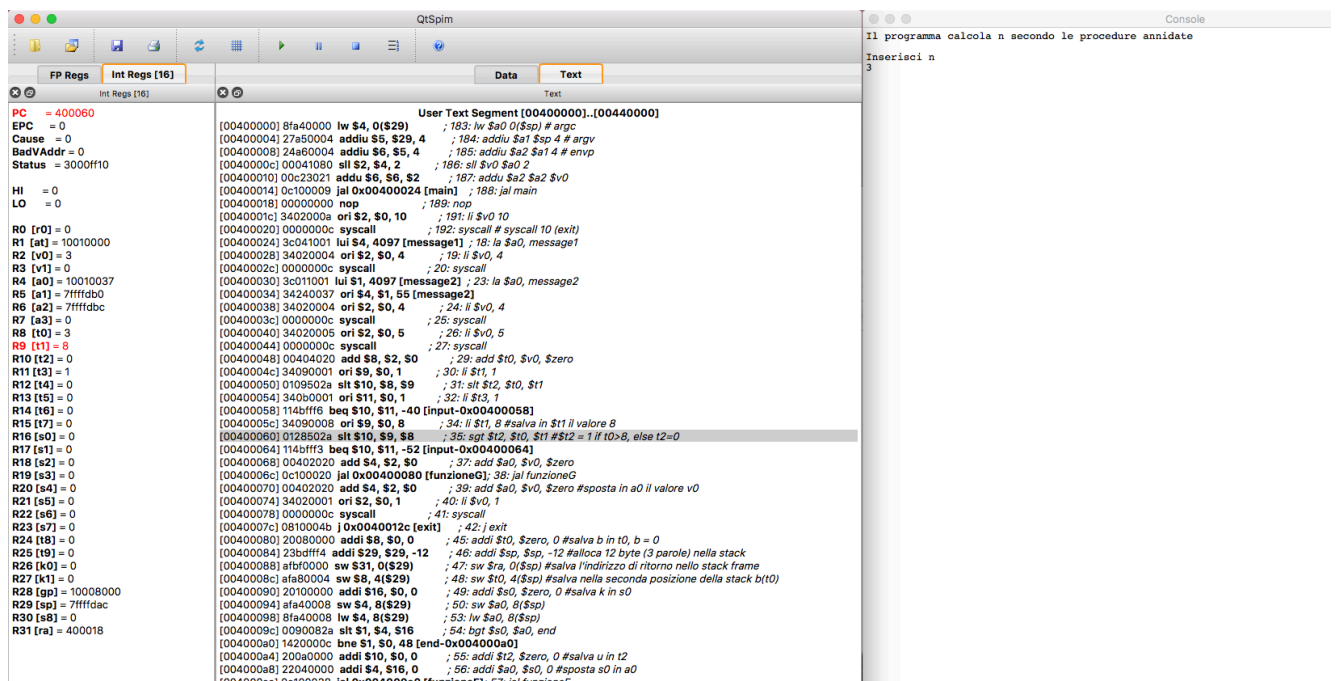
R27 [k1] = 0

Risultato dopo aver completato le chiamate ricorsive:



Esempio con input n = 3:

l'input n viene memorizzato nel registro \$t0.



Una volta effettuati i controlli (il numero in input deve essere compreso tra 1 e 8) viene richiamata la procedura “funzioneG” che alloca la stack con 3 byte,

dove vengono salvati in ordine di posizione, l'indirizzo di ritorno "\$ra", la variabile "b" e la "n" data in input.

The screenshot displays the QtSpim MIPS simulator interface. The top toolbar contains icons for file operations, execution, and debugging. Below the toolbar, there are tabs for 'FP Regs', 'Int Regs [16]', 'Data', and 'Text'. The 'Int Regs [16]' tab is active, showing a list of registers and their values. The 'Data' tab is also visible, showing memory data.

**Int Regs [16]**

```
PC = 4000ec
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000fff10

HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 3
R3 [v1] = 0
R4 [a0] = 0
R5 [a1] = 7ffffdb0
R6 [a2] = 7ffffdbc
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 8
R10 [t2] = 0
R11 [t3] = 1
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffffd98
R30 [s8] = 0
R31 [ra] = 4000b0
```

**User data segment [10000000]..[10040000]**

```
[10000000]..[1000ffff] 00000000
[10010000] 70206c49 72676f72 616d6d61 6c616320 il programma cal
[10010010] 616c6f63 73206e20 6e6f6365 6c206f64 cola n secondo l
[10010020] 72702065 6465636f 20657275 696e6e61 e procedure anni
[10010030] 65746164 49000a0a 7265736e 69637369 date...Inserisci
[10010040] 0a206e20 00000000 00000000 00000000 n .....
[10010050]..[1003ffff] 00000000
```

**User Stack [7ffffd98]..[80000000]**

```
[7ffffd98] 004000b0 00000000 ..@.....
[7ffffda0] 00400070 00000000 00000003 00000002 p.@.....
[7ffffdb0] 7ffffe33 7ffffdf4 00000000 7fffffc7 3.....
[7ffffdc0] 7fffffa1 7ffff91 7fffff74 7fffff28 .....t...(...)
[7ffffdd0] 7ffffee6 7ffffec3 7ffffea7 7ffffe82 .....
[7ffffde0] 7ffffe6c 7ffffe5e 00000000 00000000 l...^.....
[7ffffdf0] 00000000 6f72502f 74746567 2f45416f .../ProgettoAE/
[7ffffe00] 76697264 6f642d65 6f6c6e77 322d6461 drive-download-2
[7ffffe10] 30373130 54303235 39313331 2d5a3633 0170520T131936Z-
[7ffffe20] 2f313030 72657345 697a6963 612e326f 001/Esercizio2.a
[7ffffe30] 2f006d73 72657355 6f742f73 73616d6d sm./Users/tommas
[7ffffe40] 61635f6f 63636570 442f6968 746b7365 o_capecchi/Deskt
[7ffffe50] 552f706f 6576696e 74697372 50580061 op/Universita.XP
[7ffffe60] 4c465f43 3d534741 00307830 52455355 C_FLAGS=0x0.USER
[7ffffe70] 6d6f743d 6f73616d 7061635f 68636365 =tommaso_capecch
[7ffffe80] 50580069 45535f43 43495652 414e5f45 i.XPC_SERVICE_NA
[7ffffe90] 633d454d 6c2e6d6f 73757261 7374712e ME=com.larus.qts
[7ffffea0] 2e6d6970 31393732 4f4c0036 4d414e47 pim.27916.LOGNAM
[7ffffeb0] 6f743d45 73616d6d 61635f6f 63636570 E=tommaso_capecch
[7ffffec0] 50006968 3d485441 7273752f 6e69622f hi.PATH=/usr/bin
[7ffffed0] 69622f3a 752f3a6e 732f7273 3a6e6962 :/bin:/usr/sbin:
[7ffffee0] 6962732f 5353006e 55415f48 535f4854 /sbin.SSH_AUTH_S
[7ffffef0] 3d4b434f 6972702f 65746176 706d742f OCK=/private/tmp
[7fffff00] 6d6f632f 7070612e 6c2e656c 636e7561 /com.apple.launc
[7fffff10] 332e6468 44797138 4d513776 694c2f46 hd.38qyDv7QMF/Li
[7fffff20] 6e657473 00737265 6c707041 75505f65 steners.Apple_Pu
[7fffff30] 62755362 636f535f 5f74656b 646e6552 bSub_Socket_Rend
[7fffff40] 2f3d7265 76697270 2f657461 2f706d74 er=/private/tmp/
[7fffff50] 2e6d6f63 6c707061 616c2e65 68636e75 com.apple.launch
[7fffff60] 5a422e64 5045796a 6761324b 6e65522f d.BZjyEPK2ag/Ren
[7fffff70] 00726564 454d4f48 73552f3d 2f737265 der.HOME=/Users/
[7fffff80] 6d6d6f74 5f6f7361 65706163 69686363 tommaso_capecchi
[7fffff90] 45485300 2f3d4c4c 2f6e6962 68736162 .SHELL=/bin/bash
[7fffffa0] 435f5f00 53555f46 545f5245 5f545845 _._CF_USER_TEXT_
[7fffffb0] 4f434e45 474e4944 3178303d 303a3546 ENCODING=0x1F5:0
[7fffffc0] 303a3078 54003478 4944504d 762f3d52 x0:0x4.TMPDIR=/v
[7fffffd0] 662f7261 65646c6f 672f7372 637a2f33 ar/folders/g3/zc
[7fffffe0] 64666d63 71317373 6c72366e 77637763 cmfdss1qn6rlcwcw
```



Si richiama inoltre la procedura “funzioneF” che prende in input la variabile k, (ovvero il contatore del ciclo della procedura “funzioneG”). All’interno di tale procedura è necessario allocare di nuovo 8 byte nella stack, per memorizzare l’indirizzo di ritorno e la variabile k, questo passaggio è necessario per poter effettuare le eventuali chiamate ricorsive.

The screenshot displays the QtSpim MIPS simulator interface. The top toolbar includes icons for file operations, execution, and debugging. Below the toolbar, there are two main panels: 'FP Regs' and 'Int Regs [16]'. The 'Int Regs [16]' panel shows the current state of the integer registers, with the PC register highlighted in red. To the right of the registers, there are two tabs: 'Data' and 'Text'. The 'Data' tab is active, showing the memory segment starting at address 10000000. The memory contains a sequence of instructions and data, including a loop structure with a 'for' loop and a 'while' loop. The memory dump shows the following instructions and data:

```

User data segment [10000000]..[10040000]
[10000000]..[1000ffff] 00000000
[10010000] 70206c49 72676f72 616d6d61 6c616320 il programma cal
[10010010] 616c6f63 73206e20 6e6f6365 6c206f64 cola n secondo l
[10010020] 72702065 6465636f 20657275 696e6e61 e procedure anni
[10010030] 65746164 49000a0a 7265736e 69637369 date...Inserisci
[10010040] 0a206e20 00000000 00000000 00000000 n .....
[10010050]..[1003ffff] 00000000

User Stack [7ffffd98]..[80000000]
[7ffffd98] 004000b0 00000000 ..@.....
[7ffffda0] 00400070 00000000 00000003 00000002 p.@.....
[7ffffdb0] 7ffffe33 7ffffd4 00000000 7fffffc7 3.....
[7ffffdc0] 7fffffa1 7fffff91 7fffff74 7fffff28 .....t...(...)
[7ffffdd0] 7ffffee6 7ffffec3 7ffffea4 7ffff82 .....
[7ffffde0] 7ffffe6c 7ffffe5e 00000000 00000000 l...^.....
[7ffffdf0] 00000000 6f72502f 74746567 2f45416f ..../ProgettoAE/
[7ffffe00] 76697264 6f642d65 6f6c6e77 322d6461 drive-download-2
[7ffffe10] 30373130 54303235 39313331 2d5a3633 0170520T131936Z-
[7ffffe20] 2f313030 72657345 697a6963 612e326f 001/Esercizio2.a
[7ffffe30] 2f006d73 72657355 6f742f73 73616d6d sm./Users/tommas
[7ffffe40] 61635f6f 63636570 442f6968 746b7365 o_capecchii/Deskt
[7ffffe50] 552f706f 6576696e 74697372 50580061 op/Universita.XP
[7ffffe60] 4c465f43 3d534741 00307830 52455355 C_FLAGS=0x0.USER
[7ffffe70] 6d6f743d 6f73616d 7061635f 68636365 =tommaso_capecch
[7ffffe80] 50580069 45535f43 43495652 414e5f45 i.XPC_SERVICE_NA
[7ffffe90] 633d454d 6c2e6d6f 73757261 7374712e ME=com.larus.qts
[7ffffea0] 2e6d6970 31393732 4f4c0036 4d414e47 pim.27916.LOGNAM
[7ffffeb0] 6f743d45 73616d6d 61635f6f 63636570 E=tommaso_capecch
[7ffffec0] 50006968 3d485441 7273752f 6e69622f hi.PATH=/usr/bin
[7ffffed0] 69622f3a 752f3a6e 732f7273 3a6e6962 :/bin:/usr/sbin:
[7ffffee0] 6962732f 5353006e 55415f48 535f4854 /sbin.SSH_AUTH_S
[7ffffef0] 3d4b434f 6972702f 65746176 706d742f OCK=/private/tmp
[7fffff00] 6d6f632f 7070612e 6c2e656c 636e7561 /com.apple.launc
[7fffff10] 332e6468 44797138 4d513776 694c2f46 hd.38qyDv7QMF/Li
[7fffff20] 6e657473 00737265 6c707041 75505f65 steners.Apple_Pu
[7fffff30] 62755362 636f535f 5f74656b 646e6552 bSub_Socket_Rend
[7fffff40] 2f3d7265 76697270 2f657461 2f706d74 er=/private/tmp/
[7fffff50] 2e6d6f63 6c707061 616c2e65 68636e75 com.apple.launch
[7fffff60] 5a422e64 5045796a 6761324b 6e65522f d.BZjyEPK2ag/Ren
[7fffff70] 00726564 454d4f48 73552f3d 2f737265 der.HOME=/Users/
[7fffff80] 6d6d6f74 5f6f7361 65706163 69686363 tommaso_capecchi
[7fffff90] 45485300 2f3d4c4c 2f6e6962 68736162 .SHELL=/bin/bash
[7fffffa0] 435f5f00 53555f46 545f5245 5f545845 ...CF_USER_TEXT_
[7fffffb0] 4f434e45 474e4944 3178303d 303a3546 ENCODING=0x1F5:0
[7fffffc0] 303a3078 54003478 4944504d 762f3d52 x0:0x4.TMPDIR=/v
[7fffffd0] 662f7261 65646c6f 672f7372 637a2f33 ar/folders/g3/zc
[7fffffe0] 64666d63 71317373 6c72366e 77637763 cmfdss1qn6rlcwcw

```

In questo caso  $k = 0$ , quindi la funzione ritorna semplicemente il valore 1, salvato nel registro  $\$v0$ . Si ritorna quindi alla procedura “funzioneG” che potrà quindi salvare in  $\$t2$  il risultato contenuto in  $\$v0$  ed infine calcolare l’operazione “ $(b*b)+u$ ”. Tale risultato ( $0*0+1 = 1$ ) verrà scritto nella seconda posizione della stack. Si incrementa quindi la variabile “k” e si ripete il ciclo.

The screenshot shows the QtSpim MIPS simulator interface. The top toolbar contains icons for file operations, execution, and debugging. Below the toolbar, there are tabs for 'FP Regs', 'Int Regs [16]', 'Data', and 'Text'. The 'Int Regs [16]' tab is active, displaying the state of the integer registers. The 'Data' tab is also visible, showing the 'User data segment' and 'User Stack'.

**Int Regs [16]**

```

PC = 4000cc
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10

HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 1
R3 [v1] = 0
R4 [a0] = 0
R5 [a1] = 7ffffdb0
R6 [a2] = 7ffffdbc
R7 [a3] = 0
R8 [t0] = 1
R9 [t1] = 8
R10 [t2] = 1
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 1
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffffda0
R30 [s8] = 0
R31 [ra] = 4000b0
  
```

**User data segment [10000000]..[10040000]**

```

[10000000]..[1000ffff] 00000000
[10010000] 70206c49 72676f72 616d6d61 6c616320 il programma cal
[10010010] 616c6f63 73206e20 6e6f6365 6c206f64 cola n secondo l
[10010020] 72702065 6465636f 20657275 696e6e61 e procedure anni
[10010030] 65746164 49000a0a 7265736e 69637369 date...Inserisci
[10010040] 0a206e20 00000000 00000000 00000000 n .....
[10010050]..[1003ffff] 00000000
  
```

**User Stack [7ffffda0]..[80000000]**

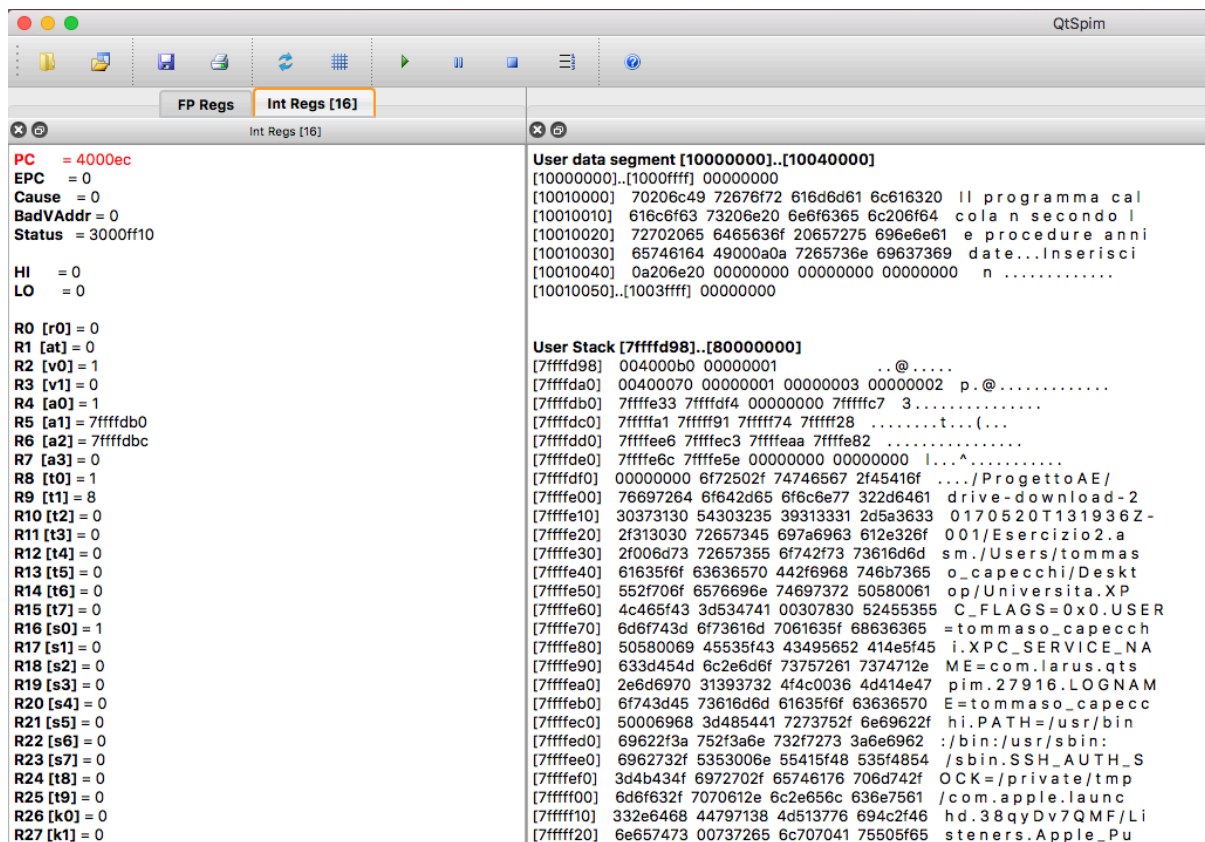
```

[7ffffda0] 00400070 00000001 00000003 00000002 p.@.....
[7ffffdb0] 7ffffe33 7ffffdf4 00000000 7fffffc7 3.....
[7ffffdc0] 7fffffa1 7fffff91 7fffff74 7fffff28 .....t...
[7ffffdd0] 7ffffee6 7ffffec3 7ffffea4 7ffffe82 .....
[7ffffde0] 7fffffc6 7fffffe5 00000000 00000000 l...^.....
[7ffffdf0] 00000000 6f72502f 74746567 2f45416f .../ProgettoAE/
[7ffffe00] 76697264 6f642d65 6f6c6e77 322d6461 drive-download-2
[7ffffe10] 30373130 54303235 39313331 2d5a3633 0170520T131936Z-
[7ffffe20] 2f313030 72657345 697a6963 612e326f 001/Esercizio2.a
[7ffffe30] 2f006d73 72657355 6f742f73 73616d6d sm./Users/tommas
[7ffffe40] 61635f6f 63636570 442f6968 746b7365 o_capecchi/Deskt
[7ffffe50] 552f706f 6576696e 74697372 50580061 op/Universita.XP
[7ffffe60] 4c465f43 3d534741 00307830 52455355 C_FLAGS=0x0.USER
[7ffffe70] 6d6f743d 6f73616d 7061635f 68636365 =tommaso_capecch
[7ffffe80] 50580069 45535f43 43495652 414e5f45 i.XPC_SERVICE_NA
[7ffffe90] 633d454d 6c2e6d6f 73757261 7374712e ME=com.larus.qts
[7ffffea0] 2e6d6970 31393732 4f4c0036 4d414e47 pim.27916.LOGNAM
[7ffffeb0] 6f743d45 73616d6d 61635f6f 63636570 E=tommaso_capecch
[7ffffec0] 50006968 3d485441 7273752f 6e69622f hi.PATH=/usr/bin
[7ffffed0] 69622f3a 752f3a6e 732f7273 3a6e6962 :/bin:/usr/sbin:
[7ffffee0] 6962732f 5353006e 55415f48 535f4854 /sbin.SSH_AUTH_S
[7ffffef0] 3d4b434f 6972702f 65746176 706d742f OCK=/private/tmp
[7fffff00] 6d6f632f 7070612e 6c2e656c 636e7561 /com.apple.launc
[7fffff10] 332e6468 44797138 4d513776 694c2f46 hd.38qyDv7QMF/Li
[7fffff20] 6e657473 00737265 6c707041 75505f65 steners.Apple_Pu
[7fffff30] 62755362 636f535f 5f74656b 646e6552 bSub_Socket_Rend
[7fffff40] 2f3d7265 76697270 2f657461 2f706d74 er=/private/tmp/
[7fffff50] 2e6d6f63 6c707061 616c2e65 68636e75 com.apple.launch
[7fffff60] 5a422e64 5045796a 6761324b 6e65522f d.BZjyEPK2ag/Ren
[7fffff70] 00726564 454d4f48 73552f3d 2f737265 der.HOME=/Users/
[7fffff80] 6d6d6f74 5f6f7361 65706163 69686363 tommaso_capecchi
[7fffff90] 45485300 2f3d4c4c 2f6e6962 68736162 .SHELL=/bin/bash
[7fffffa0] 435f5f00 53555f46 545f5245 5f545845 ._CF_USER_TEXT_
[7fffffb0] 4f434e45 474e4944 3178303d 303a3546 ENCODING=0x1F5:0
[7fffffc0] 303a3078 54003478 4944504d 762f3d52 x0:0x4.TMPDIR=/v
[7fffffd0] 662f7261 65646c6f 672f7372 637a2f33 ar/folders/g3/zc
[7fffffe0] 64666d63 71317373 6c72366e 77637763 cmfdss1qn6rlcwcw
[7ffffff0] 6a6b3267 30306d35 6e673030 002f542f g2kj5m0000gn/T/.
  
```

Infine viene preso in esame il comportamento delle varie chiamate ricorsive sempre con input  $n = 3$ , ma considerando che il contatore “k” del ciclo for della procedura “funzioneG” sia  $k = 1$  (si ricorda che k è memorizzato nel registro  $\$s0$  poichè il valore viene preservato):

QtSpim	
FP Regs	Int Regs [16]
Int Regs [16]	
<b>PC = 400098</b> <b>EPC = 0</b> <b>Cause = 0</b> <b>BadVAddr = 0</b> <b>Status = 3000fff0</b>  <b>HI = 0</b> <b>LO = 0</b>  <b>R0 [r0] = 0</b> <b>R1 [at] = 0</b> <b>R2 [v0] = 1</b> <b>R3 [v1] = 0</b> <b>R4 [a0] = 0</b> <b>R5 [a1] = 7ffffdb0</b> <b>R6 [a2] = 7ffffdbc</b> <b>R7 [a3] = 0</b> <b>R8 [t0] = 1</b> <b>R9 [t1] = 8</b> <b>R10 [t2] = 1</b> <b>R11 [t3] = 0</b> <b>R12 [t4] = 0</b> <b>R13 [t5] = 0</b> <b>R14 [t6] = 0</b> <b>R15 [t7] = 0</b> <b>R16 [s0] = 1</b> <b>R17 [s1] = 0</b> <b>R18 [s2] = 0</b> <b>R19 [s3] = 0</b> <b>R20 [s4] = 0</b> <b>R21 [s5] = 0</b> <b>R22 [s6] = 0</b> <b>R23 [s7] = 0</b> <b>R24 [t8] = 0</b> <b>R25 [t9] = 0</b> <b>R26 [k0] = 0</b> <b>R27 [k1] = 0</b>	<b>User data segment [10000000]..[10040000]</b> [10000000]..[1000ffff] 00000000 [10010000] 70206c49 72676f72 616d6d61 6c616320 il programma cal [10010010] 616c6f63 73206e20 6e6f6365 6c206f64 cola n secondo l [10010020] 72702065 6465636f 20657275 696e6e61 e procedure anni [10010030] 65746164 49000a0a 7265736e 69637369 date...Inserisci [10010040] 0a206e20 00000000 00000000 00000000 n ..... [10010050]..[1003ffff] 00000000  <b>User Stack [7ffffff0]..[80000000]</b> [7ffffff0] 00400070 00000001 00000003 00000002 p.@..... [7ffffffb] 7ffffe33 7ffffdf4 00000000 7fffffc7 3..... [7ffffffc] 7fffffa1 7fffff91 7fffff74 7fffff28 .....t...( [7ffffffd] 7ffffee6 7ffffec3 7ffffeaa 7ffffe82 ..... [7ffffffe] 7ffffe6c 7ffffe5e 00000000 00000000 l...^..... [7fffffff] 00000000 6f72502f 74746567 2f45416f ..../ProgettoAE/ [7fffffff] 76697264 6f642d65 6f6c6e77 322d6461 drive-download-2 [7fffffff] 30373130 54303235 39313331 2d5a3633 0170520T131936Z- [7fffffff] 2f313030 72657345 697a6963 612e326f 001/Esercizio2.a [7fffffff] 2f006d73 72657355 6f742f73 73616d6d sm./Users/tommas [7fffffff] 61635f6f 63636570 442f6968 746b7365 o_capecchi/Deskt [7fffffff] 552f706f 6576696e 74697372 50580061 op/Universita.XP [7fffffff] 4c465f43 3d534741 00307830 52455355 C_FLAGS=0x0.USER [7fffffff] 6d6f743d 6f73616d 7061635f 68636365 =tommaso_capecch [7fffffff] 50580069 45535f43 43495652 414e5f45 i.XPC_SERVICE_NA [7fffffff] 633d454d 6c2e6d6f 73757261 7374712e ME=com.larus.qts [7fffffff] 2e6d6970 31393732 4f4c0036 4d414e47 pim.27916.LOGNAM [7fffffff] 6f743d45 73616d6d 61635f6f 63636570 E=tommaso_capecc [7fffffff] 50006968 3d485441 7273752f 6e69622f hi.PATH=/usr/bin [7fffffff] 69622f3a 752f3a6e 732f7273 3a6e6962 :/bin:/usr/sbin: [7fffffff] 6962732f 5353006e 55415f48 535f4854 /sbin.SSH_AUTH_S [7fffffff] 3d4b434f 6972702f 65746176 706d742f OCK=/private/tmp [7fffffff] 6d6f632f 7070612e 6c2e656c 636e7561 /com.apple.launc [7fffffff] 332e6468 44797138 4d513776 694c2f46 hd.38qyDv7QMF/Li [7fffffff] 6e657473 00737265 6c707041 75505f65 steners.Apple_Pu [7fffffff] 62755362 636f535f 5f74656b 646e6552 bSub_Socket_Rend

Si effettua quindi la chiamata alla procedura “funzioneF” che allocherà nella stack l’indirizzo di ritorno (\$ra) e il valore della k, passata in argomento tramite il registro \$a0:



The screenshot shows the QtSpim MIPS simulator interface. The 'Int Regs [16]' window is active, displaying the following register values:

```

PC = 4000ec
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10

HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 1
R3 [v1] = 0
R4 [a0] = 1
R5 [a1] = 7ffffdb0
R6 [a2] = 7ffffdbc
R7 [a3] = 0
R8 [t0] = 1
R9 [t1] = 8
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 1
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0

```

The 'User data segment [10000000]..[10040000]' window shows the following memory contents:

```

[10000000]..[1000ffff] 00000000
[10010000] 70206c49 72676f72 616d6d61 6c616320 il programma cal
[10010010] 616c6f63 73206e20 6e6f6365 6c206f64 cola n secondo l
[10010020] 72702065 6465636f 20657275 696e6e61 e procedure anni
[10010030] 65746164 49000a0a 7265736e 69637369 date...Inserisci
[10010040] 0a206e20 00000000 00000000 00000000 n .....
[10010050]..[1003ffff] 00000000

```

The 'User Stack [7ffffd98]..[80000000]' window shows the following memory contents:

```

[7ffffd98] 004000b0 00000001 ..@.....
[7ffffda0] 00400070 00000001 00000003 00000002 p.@.....
[7ffffdb0] 7ffffe33 7ffffdf4 00000000 7fffffc7 3.....
[7ffffdc0] 7fffffa1 7fffff91 7fffff74 7fffff28 .....t...(..
[7ffffdd0] 7ffffee6 7ffffec3 7ffffeaa 7ffffe82 .....
[7ffffde0] 7ffffe6c 7ffffe5e 00000000 00000000 l...^.....
[7ffffdf0] 00000000 6f72502f 74746567 2f45416f ..../ProgettoAE/
[7ffffe00] 76697264 6f642d65 6f6c6e77 322d6461 drive-download-2
[7ffffe10] 30373130 54303235 39313331 2d5a3633 0170520T131936Z-
[7ffffe20] 2f313030 72657345 697a6963 612e326f 001/Esercizio2.a
[7ffffe30] 2f006d73 72657355 6f742f73 73616d6d sm./Users/tommas
[7ffffe40] 61635f6f 63636570 442f6968 746b7365 o_capecchi/Deskt
[7ffffe50] 552f706f 6576696e 74697372 50580061 op/Universita.XP
[7ffffe60] 4c465f43 3d534741 00307830 52455355 C_FLAGS=0x0.USER
[7ffffe70] 6d6f743d 6f73616d 7061635f 68636365 =tommaso_capecc
[7ffffe80] 50580069 45535f43 43495652 414e5f45 i.XPC_SERVICE_NA
[7ffffe90] 633d454d 6c2e6d6f 73757261 7374712e ME=com.larus.qts
[7ffffea0] 2e6d6970 31393732 4f4c0036 4d414e47 pim.27916.LOGNAM
[7ffffeb0] 6f743d45 73616d6d 61635f6f 63636570 E=tommaso_capecc
[7ffffec0] 50006968 3d485441 7273752f 6e69622f hi.PATH=/usr/bin
[7ffffed0] 69622f3a 752f3a6e 732f7273 3a6e6962 :/bin:/usr/sbin:
[7ffffee0] 6962732f 5353006e 55415f48 535f4854 /sbin.SSH_AUTH_S
[7ffffef0] 3d4b434f 6972702f 65746176 706d742f OCK=/private/tmp
[7fffff00] 6d6f632f 7070612e 6c2e656c 636e7561 /com.apple.launc
[7fffff10] 332e6468 44797138 4d513776 694c2f46 hd.38qyDv7QMF/Li
[7fffff20] 6e657473 00737265 6c707041 75505f65 steners.Apple_Pu

```



poiché  $k = 1$ , la “funzioneF” effettuerà l’operazione “ $2 \cdot F(n-1) + n$ ” (vale la pena chiarire che  $n$  è l’input della procedura “funzioneF” nonché lo stesso  $k$  passato come parametro). Questo comporta una chiamata ricorsiva: la procedura “funzioneF” richiama quindi se stessa con parametro  $F(n-1)$ , quindi riallocherà nuovamente la stack con il nuovo indirizzo di ritorno ed il nuovo valore del parametro  $n-1$  (che assume valore 0):

The screenshot shows a debugger window with two main panes. The left pane displays the state of the CPU registers, and the right pane shows a memory dump of the user data segment and stack.

**Left Pane: CPU Registers**

```

PC = 40011c
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10

HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 1
R3 [v1] = 0
R4 [a0] = 0
R5 [a1] = 7ffffdb0
R6 [a2] = 7ffffdbc
R7 [a3] = 0
R8 [t0] = 1
R9 [t1] = 8
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 1
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
  
```

**Right Pane: Memory Dump**

**User data segment [10000000]..[10040000]**

```

[10000000]..[1000ffff] 00000000
[10010000] 70206c49 72676f72 616d6d61 6c616320 Il programma cal
[10010010] 616c6f63 73206e20 6e6f6365 6c206f64 cola n secondo l
[10010020] 72702065 6465636f 20657275 696e6e61 e procedure anni
[10010030] 65746164 49000a0a 7265736e 69637369 date...Inserisci
[10010040] 0a206e20 00000000 00000000 00000000 n .....
[10010050]..[1003ffff] 00000000
  
```

**User Stack [7ffffd90]..[80000000]**

```

[7ffffd90] 004000f8 00000000 004000b0 00000001 ..@.....@.....
[7ffffda0] 00400070 00000001 00000003 00000002 p.@.....
[7ffffdb0] 7ffffe33 7ffffdf4 00000000 7fffffc7 3.....
[7ffffdc0] 7fffffa1 7fffff91 7fffff74 7fffff28 .....t...(..
[7ffffdd0] 7ffffee6 7ffffec3 7ffffea9 7ffffe82 .....
[7ffffde0] 7ffffe6c 7ffffe5e 00000000 00000000 l...^.....
[7ffffdf0] 00000000 6f72502f 74746567 2f45416f ..../ProgettoAE/
[7ffffe00] 76697264 6f642d65 6f6c6e77 322d6461 drive-download-2
[7ffffe10] 30373130 54303235 39313331 2d5a3633 0170520T131936Z-
[7ffffe20] 2f313030 72657345 697a6963 612e326f 001/Esercizio2.a
[7ffffe30] 2f006d73 72657355 6f742f73 73616d6d sm./Users/tommas
[7ffffe40] 61635f6f 63636570 442f6968 746b7365 o_capecchi/Desk
[7ffffe50] 552f706f 6576696e 74697372 50580061 op/Universita.XP
[7ffffe60] 4c465f43 3d534741 00307830 52455355 C_FLAGS=0x0.USER
[7ffffe70] 6d6f743d 6f73616d 7061635f 68636365 =tommaso_capecch
[7ffffe80] 50580069 45535f43 43495652 414e5f45 i.XPC_SERVICE_NA
[7ffffe90] 633d454d 6c2e6d6f 73757261 7374712e ME=com.larus.qts
[7ffffea0] 2e6d6970 31393732 4f4c0036 4d414e47 pim.27916.LOGNAM
[7ffffeb0] 6f743d45 73616d6d 61635f6f 63636570 E=tommaso_capecch
[7ffffec0] 50006968 3d485441 7273752f 6e69622f hi.PATH=/usr/bin
[7ffffed0] 69622f3a 752f3a6e 732f7273 3a6e6962 :/bin:/usr/sbin:
[7ffffee0] 6962732f 5353006e 55415f48 535f4854 /sbin.SSH_AUTH_S
[7ffffef0] 3d4b434f 6972702f 65746176 706d742f OCK=/private/tmp
[7fffff00] 6d6f632f 7070612e 6c2e656c 636e7561 /com.apple.launc
[7fffff10] 332e6468 44797138 4d513776 694c2f46 hd.38qyDv7QMF/Li
[7fffff20] 6e657473 00737265 6c707041 75505f65 steners.Apple_Pu
  
```

Sappiamo che  $F(0) = 1$ ; quindi grazie all'indirizzo di ritorno precedentemente salvato nella stack, possiamo tornare all'indirizzo di chiamata e concludere l'operazione " $2 * F(n-1) + n$ " dove " $F(n-1) = 1$ " quindi " $(2 * 1) + 1 = 3$ ", valore memorizzato in  $\$v0$ :

The screenshot shows the QtSpim MIPS simulator. On the left, the 'FP Regs' and 'Int Regs [16]' panels are visible. The 'Int Regs [16]' panel shows the following register values:

```

PC = 400118
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 2
R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 3
R3 [v1] = 0
R4 [a0] = 0
R5 [a1] = 7ffffdb0
R6 [a2] = 7ffffdb0
R7 [a3] = 0
R8 [t0] = 1
R9 [t1] = 2
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 2
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 1
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0

```

The main window displays assembly code with comments. The code is as follows:

```

[00400004] 81a00004 lw $t0, 4($29) ; 59: lw $t0, 4($sp) #estrae la k
[00400008] 01080018 mult $8, $8 ; 60: mult $t0, $t0 #moltiplica b con se stesso
[0040000c] 00005812 mflo $11 ; 61: mflo $t3 #salva il risultato di mult in t3
[00400010] 016a4020 add $8, $11, $10 ; 62: add $t0, $t3, $t2
[00400014] afa80004 sw $8, 4($29) ; 63: sw $t0, 4($sp)
[00400018] 22100001 addi $16, $16, 1 ; 64: addi $s0, $s0, 1
[0040001c] 08100026 j 0x00400098 [loop] ; 65: j loop
[00400020] 8fbf0000 lw $31, 0($29) ; 68: lw $ra, 0($sp) #carica indirizzo di ritorno in ra
[00400024] 8fa20004 lw $2, 4($29) ; 69: lw $v0, 4($sp) #estrae b finale dalla stack
[00400028] 23bd000c addi $29, $29, 12 ; 70: addi $sp, $sp, 12 #dealloca la stack
[0040002c] 03e00008 jr $31 ; 71: jr $ra #ritorna al main
[00400030] 23bdf0f8 addi $29, $29, -8 ; 74: addi $sp, $sp, -8 #rialloca la stack
[00400034] afbf0000 sw $31, 0($29) ; 75: sw $ra, 0($sp) #salva l'indirizzo di ritorno nello stack frame
[00400038] afa40004 sw $4, 4($29) ; 76: sw $a0, 4($sp) #salva k nello sf
[0040003c] 1004000c beq $0, $4, 48 [casoUgualea0-0x004000ec]
[00400040] 2084ffff addi $4, $4, -1 ; 78: addi $a0, $a0, -1
[00400044] 0c100038 jal 0x004000e0 [funzioneF]; 79: jal funzioneF
[00400048] 204c0000 addi $12, $2, 0 ; 80: addi $t4, $v0, 0 #salva in t4 il valore di F(n-1)
[0040004c] 34090002 ori $9, $0, 2 ; 81: li $t1, 2
[00400050] 01890018 mult $12, $9 ; 82: mult $t4, $t1 #moltiplica t4 per 2
[00400054] 00006012 mflo $12 ; 83: mflo $t4 #salva il risultato in t4
[00400058] 8fa80004 lw $8, 4($29) ; 84: lw $t0, 4($sp) #estrae la k dalla stack
[0040005c] 01881020 add $2, $12, $8 ; 85: add $v0, $t4, $t0 #2*F(n-1) + n
[00400060] 8fbf0000 lw $31, 0($29) ; 86: lw $ra, 0($sp) #estrae l'indirizzo di ritorno
[00400064] 23bd0008 addi $29, $29, 8 ; 87: addi $sp, $sp, 8 #dealloca la stack
[00400068] 03e00008 jr $31 ; 88: jr $ra
[0040006c] 20020001 addi $2, $0, 1 ; 91: addi $v0, $zero, 1
[00400070] 8fbf0000 lw $31, 0($29) ; 92: lw $ra, 0($sp)
[00400074] 23bd0008 addi $29, $29, 8 ; 93: addi $sp, $sp, 8
[00400078] 03e00008 jr $31 ; 94: jr $ra
[0040007c] 3402000a ori $2, $0, 10 ; 97: li $v0, 10
[00400080] 0000000c syscall ; 98: syscall

```

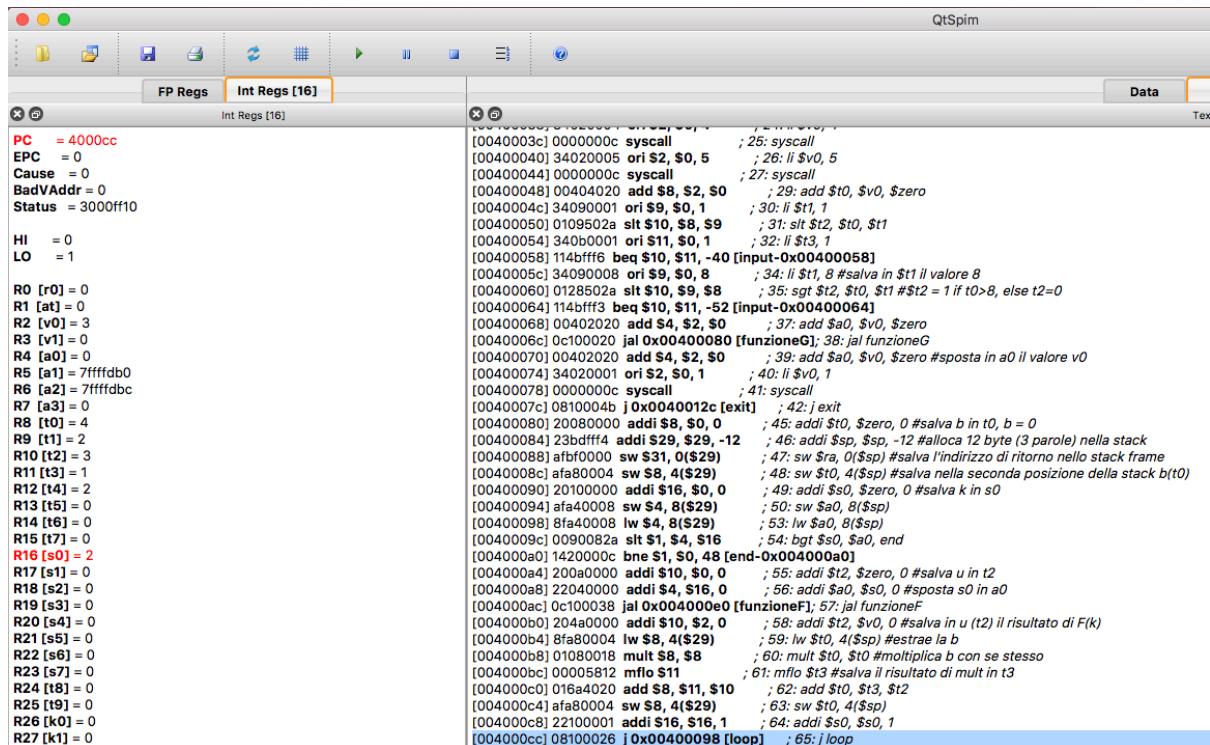
At the bottom, the 'Kernel Text Segment [80000180]' is shown with the following code:

```

[80000180] 0001d821 addu $27, $0, $1 ; 90: move $k1 $at # Save $at
[80000184] 3c019000 lui $1, -28672 ; 92: sw $v0 $1 # Not re-entrant and we can't trust $sp
[80000188] ac220200 sw $2, 512($1)
[8000018c] 3c019000 lui $1, -28672 ; 93: sw $a0 $2 # But we need to use these registers

```

Infine tramite l'istruzione "jr" ritorniamo alla procedura chiamante "funzioneG" dove possiamo finalmente aggiornare il risultato come segue:



The screenshot shows the QtSpim MIPS simulator interface. On the left, the 'Int Regs [16]' window displays the current state of the registers:

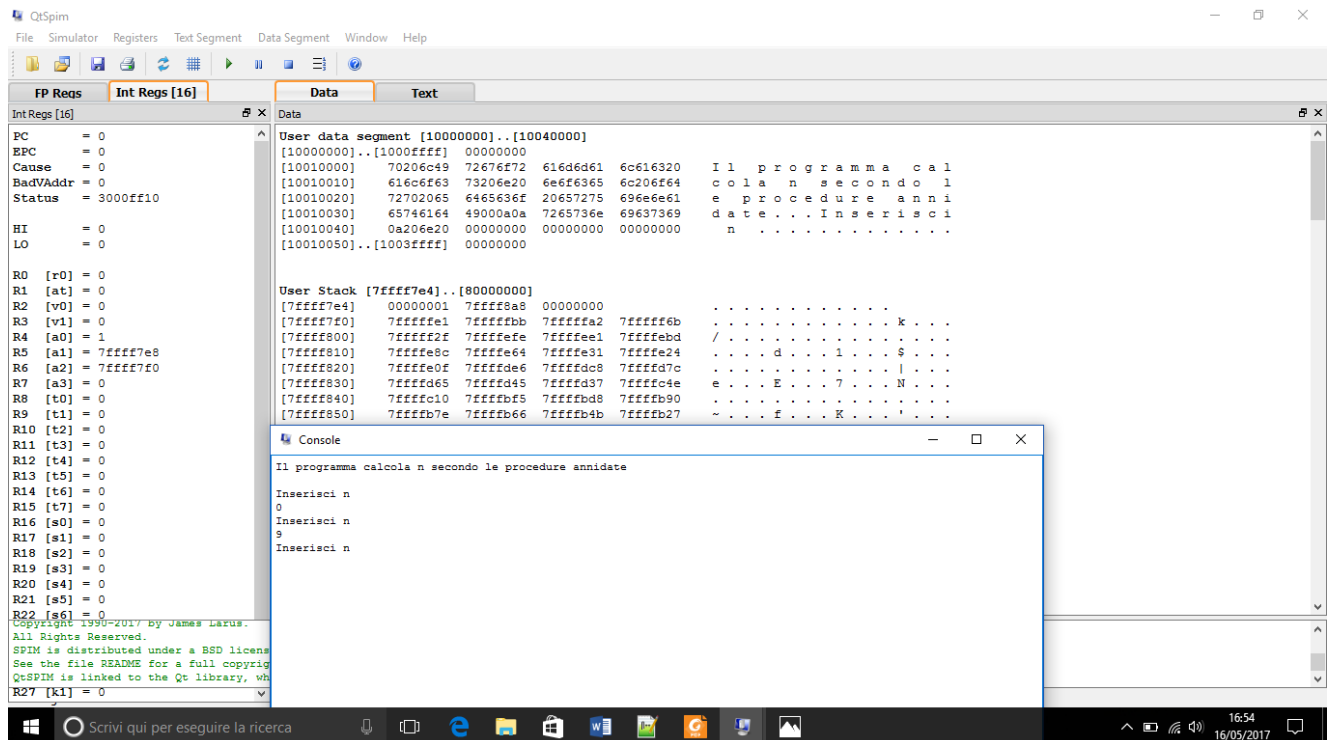
```
PC = 4000cc
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 1
R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 3
R3 [v1] = 0
R4 [a0] = 0
R5 [a1] = 7ffffdb0
R6 [a2] = 7ffffdbc
R7 [a3] = 0
R8 [t0] = 4
R9 [t1] = 2
R10 [t2] = 3
R11 [t3] = 1
R12 [t4] = 2
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 2
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

The main window displays the assembly code with comments. The instruction at address 0x00400026 is highlighted in blue:

```
[00400026] 08100026 jr $ra, $ra ; jr: jump register
```

Viene aggiornato il contatore k, ed il ciclo riprende la sua esecuzione fino a quando  $k > 3$ .

## Un esempio di input sbagliato:





## 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.

## Descrizione della soluzione:

Abbiamo realizzato il menù di scelta con una serie di procedure che stampano dei messaggi. Le operazioni disponibili sono: inserimento, somma, sottrazione, moltiplicazione e uscita, selezionabili dai seguenti caratteri, rispettivamente: "a", "b", "c", "d", ed "e". Tramite uno switch si verifica che effettivamente l'utente abbia inserito uno dei precedenti caratteri, che a loro volta ci manderanno nei rispettivi case:

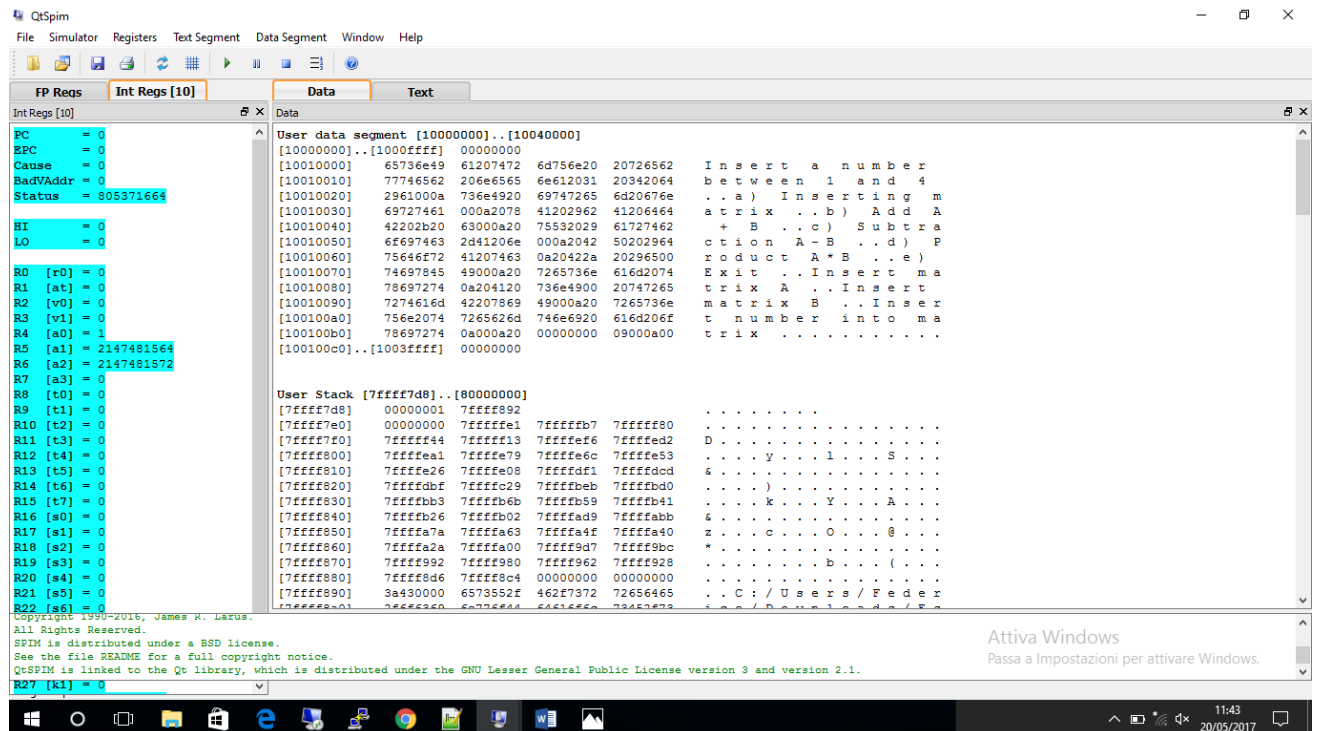
- **caseA:** richiama la procedura "input" grazie alla quale l'utente può inserire l'ordine delle matrici A e B (compreso tra 1 e 4 inclusi). L'inserimento dei valori avviene nella procedura "inserimenti" che prende in input quattro parametri, che sono la *n* appena inserita, l'eventuale *n* precedente (ordine matrice precedente), l'eventuale indirizzo della matrice A e l'eventuale indirizzo della matrice B. Si alloca la stack per salvare la *n*, la vecchia *n* e l'indirizzo di ritorno. Viene effettuato un salto condizionato che ci riporta alla procedura "changeValues" nel caso in cui le due *n* siano uguali poiché si sovrascrivono le vecchie matrici riutilizzando lo stesso spazio in memoria. Altrimenti si alloca lo spazio necessario per memorizzare la matrice nella memoria dinamica utilizzando la procedura "sbrk" (li \$v0, 9). Infine si inseriscono i valori degli elementi delle matrici. Tramite la procedura "exitFunction" si salvano i valori di ritorno della funzione (gli indirizzi in memoria di A e B), si dealloca la stack e si ritorna all'indirizzo di chiamata, che aggiornerà i vari registri preservati.
- **caseB:** questo caso effettua la somma tra i valori delle due matrici. Per poterla effettuare vengono utilizzati due cicli for annidati per tener traccia delle righe e delle colonne; poiché entrambe le matrici sono state allocate nella memoria dinamica, e conoscendo l'indirizzo iniziale, gli

elementi che sono posti l'uno in successione all'altro sono calcolati tramite la formula "row major ordering" ( $\text{indirizzo elemento} = \text{indirizzo base} + (2^2 * (n * i + j))$ ). Viene infine effettuata la somma dei vari elementi ed ogni elemento viene stampato a console.

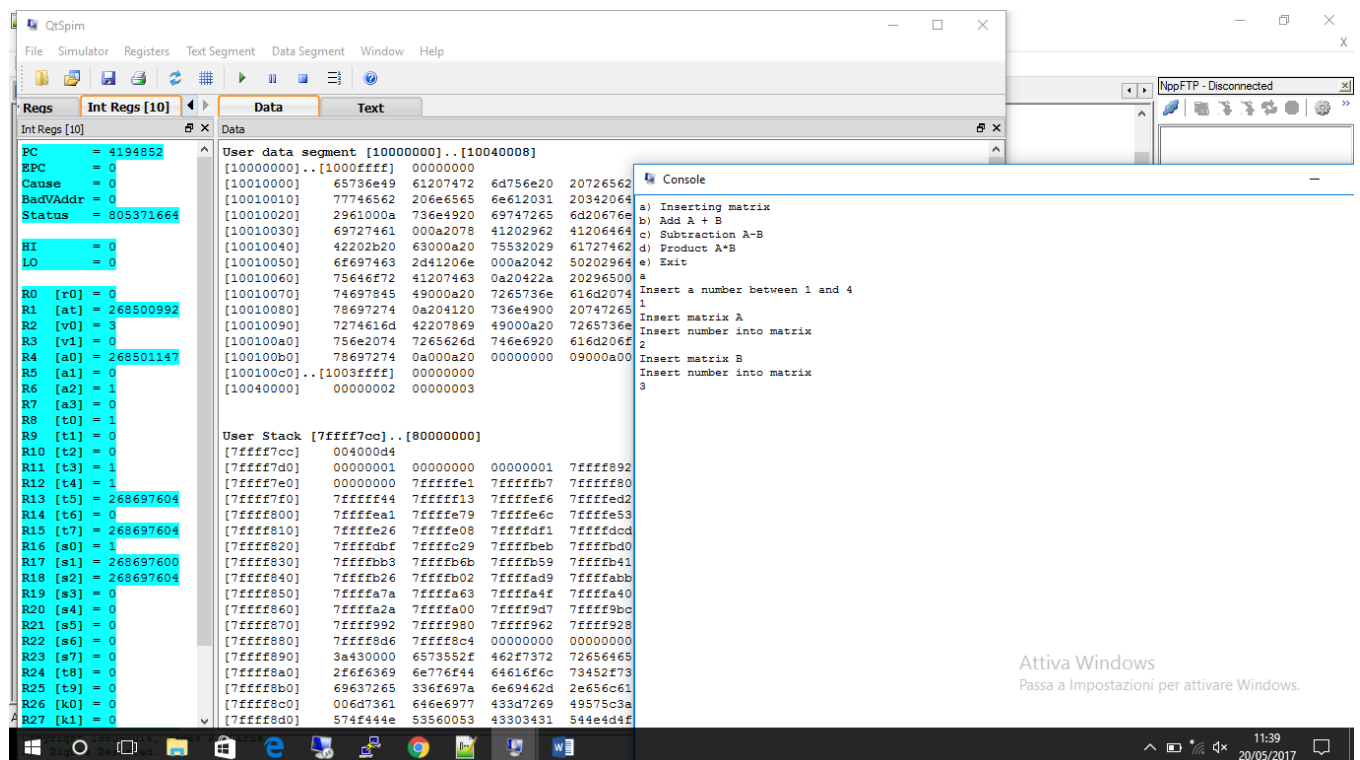
- caseC: si svolge in modo analogo al caso precedente per quanto riguarda la ricerca in memoria dinamica degli elementi delle matrici. Si effettua un salto condizionato per confrontare se l'input immesso dall'utente non sia il carattere relativo alla somma (sia somma che sottrazione appartengono alla stessa procedura), dopo di che si procede con la sottrazione dei valori. Infine vengono stampati i valori a console.
- caseD: si effettuano tre cicli for per realizzare l'operazione di prodotto riga \* colonna delle due matrici. Nella procedura "prodottoSemplice" si allocano quattro posizioni nella stack contenenti gli indici dei for e la somma temporanea. Dopo di che si effettua il calcolo "row major ordering" per reperire l'indirizzo dell'elemento in memoria dinamica; si calcola il prodotto tra gli elementi della matrice trovati e si aggiunge alla somma temporanea. Alla fine dei cicli interni si stampano i valori ottenuti.
- caseE: viene chiamata la procedura per terminare il programma.

# Simulazione:

Situazione “user data segment” prima di invocare la procedura “sbrk” per allocare lo spazio nella memoria dinamica:



Situazione “user data segment” dopo l’inserimento dei valori delle matrici di ordine 1:



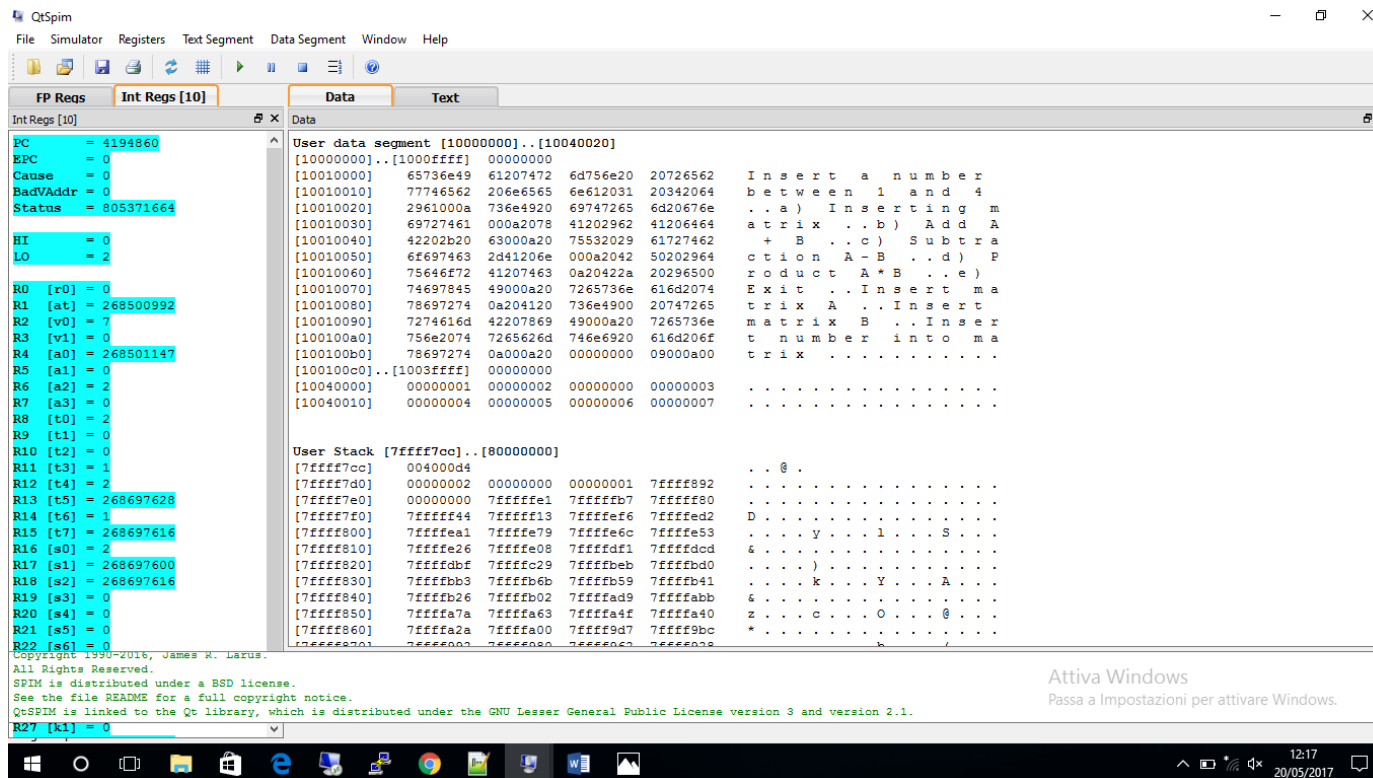
Il registro “\$s1” contiene l’indirizzo di base della matrice A, mentre il registro “\$s2” contiene l’indirizzo di base della matrice B. In “\$s0” abbiamo memorizzato l’ordine delle due matrici.

Per quanto riguarda la somma e la sottrazione, una volta trovato il risultato viene stampato a console direttamente, senza memorizzare il valore.

Per quanto riguarda invece la moltiplicazione usiamo la stack per memorizzare gli indici dei 3 for( il registro “\$t3” contiene l’indice i del primo for, il registro “\$t4” contiene l’indice j del secondo for, il registro “\$t1” contiene l’indice k del terzo for) e il risultato temporaneo contenuto nel registro “\$t8”, necessario per sommare i vari prodotti tra riga e colonna.

Il primo for scorre le colonne della 1° matrice, il secondo for scorre le righe della 2° matrice ed infine nel terzo for, si calcolano i prodotti e si sommano i valori.

Esempio di moltiplicazione tra matrici di ordine 2:



QtSPIM

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [10] Data Text

Int Regs [10]

PC = 4194860  
EPC = 0  
Cause = 0  
BadVAddr = 0  
Status = 805371664

HI = 0  
LO = 2

R0 [r0] = 0  
R1 [at] = 268500992  
R2 [v0] = 7  
R3 [v1] = 0  
R4 [a0] = 268501147  
R5 [a1] = 0  
R6 [a2] = 2  
R7 [a3] = 0  
R8 [t0] = 2  
R9 [t1] = 0  
R10 [t2] = 0  
R11 [t3] = 1  
R12 [t4] = 2  
R13 [t5] = 268697628  
R14 [t6] = 4  
R15 [t7] = 268697616  
R16 [s0] = 2  
R17 [s1] = 268697600  
R18 [s2] = 268697616  
R19 [s3] = 0  
R20 [s4] = 0  
R21 [s5] = 0  
R22 [s6] = 0

User data segment [10000000]..[10040020]

[10000000]..[1000ffff] 00000000  
[10010000] 65736e49 61207472 6d756e20 20726562 Insert a number  
[10010010] 77746562 206e6565 6e612031 20342064 between 1 and 4  
[10010020] 2961000a 736e4920 69747265 6d20676e . . a) Inserting m  
[10010030] 69727461 000a2078 41202962 41206464 atrix . . b) Add A  
[10010040] 42202b20 63000a20 75532029 61727462 + B . . c) Subtra  
[10010050] 6f697463 2d41206e 000a2042 50202964 ction A-B . . d) P  
[10010060] 75646e72 41207463 0a20422a 20296500 roduct A\*B . . e)  
[10010070] 74697845 49000a20 7265736e 616d2074 Exit . . Insert ma  
[10010080] 78697274 0a204120 736e4900 20747265 trix A . . Insert  
[10010090] 7274616d 42207869 49000a20 7265736e matrix B . . Inser  
[100100a0] 756e2074 7265626d 746e6920 616d206f t number into ma  
[100100b0] 78697274 0a000a20 00000000 09000a00 trix . . . . .  
[100100c0]..[1003ffff] 00000000  
[10040000] 00000001 00000002 00000000 00000003 . . . . .  
[10040010] 00000004 00000005 00000006 00000007 . . . . .

User Stack [7ffff700]..[80000000]

[7ffff700] 004000d4 . . @ .  
[7ffff7d0] 00000002 00000000 00000001 7ffff892 . . . . .  
[7ffff7e0] 00000000 7fffffe1 7fffffb7 7fffff80 . . . . .  
[7ffff7f0] 7fffffe4 7fffff13 7fffffe6 7ffffed2 D . . . . .  
[7ffff800] 7fffffe1 7ffff79 7fffffe6 7ffffe53 . . . . y . . . l . . . S . . .  
[7ffff810] 7ffffe26 7ffffe08 7ffffdf1 7ffffdc0 & . . . . .  
[7ffff820] 7ffffdbf 7ffffc29 7ffffbeb 7ffffbd0 . . . . ) . . . . .  
[7ffff830] 7ffffbb3 7ffffb6b 7ffffb59 7ffffb41 . . . . k . . . . Y . . . A . . .  
[7ffff840] 7ffffb26 7ffffb02 7ffffad9 7ffffabb & . . . . .  
[7ffff850] 7ffffa7a 7ffffa63 7ffffa4f 7ffffa40 z . . . . c . . . . O . . . . @ . . . .  
[7ffff860] 7ffffa2a 7ffffa00 7ffff9d7 7ffff9bc \* . . . . .

Copyright 1990-2016, James M. Latus.  
All Rights Reserved.  
SPIM is distributed under a BSD license.  
See the file README for a full copyright notice.  
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Attiva Windows  
Passa a Impostazioni per attivare Windows.

12:17  
20/05/2017

Matrice A: [1,2,0,3].

Matrice B: [4,5,6,7].

Ricavo gli indirizzi della memoria dinamica, degli elementi della matrice A[i,j] e della matrice B[j,i]:

The screenshot shows the QtSpim MIPS simulator interface. The 'Int Regs [10]' window on the left displays the following register values:

Register	Value
PC	4195272
EPC	0
Cause	0
BadVAddr	0
Status	805371664
HI	0
LO	0
R0 [r0]	0
R1 [at]	268500992
R2 [v0]	4
R3 [v1]	268697616
R4 [a0]	0
R5 [a1]	0
R6 [a2]	0
R7 [a3]	0
R8 [t0]	2
R9 [t1]	0
R10 [t2]	100
R11 [t3]	0
R12 [t4]	0
R13 [t5]	268697600
R14 [t6]	268697616
R15 [t7]	0
R16 [s0]	2
R17 [s1]	268697600
R18 [s2]	268697616
R19 [s3]	2
R20 [s4]	0
R21 [s5]	0
R22 [s6]	0

The 'Text' window on the right shows assembly code. The highlighted instruction is:

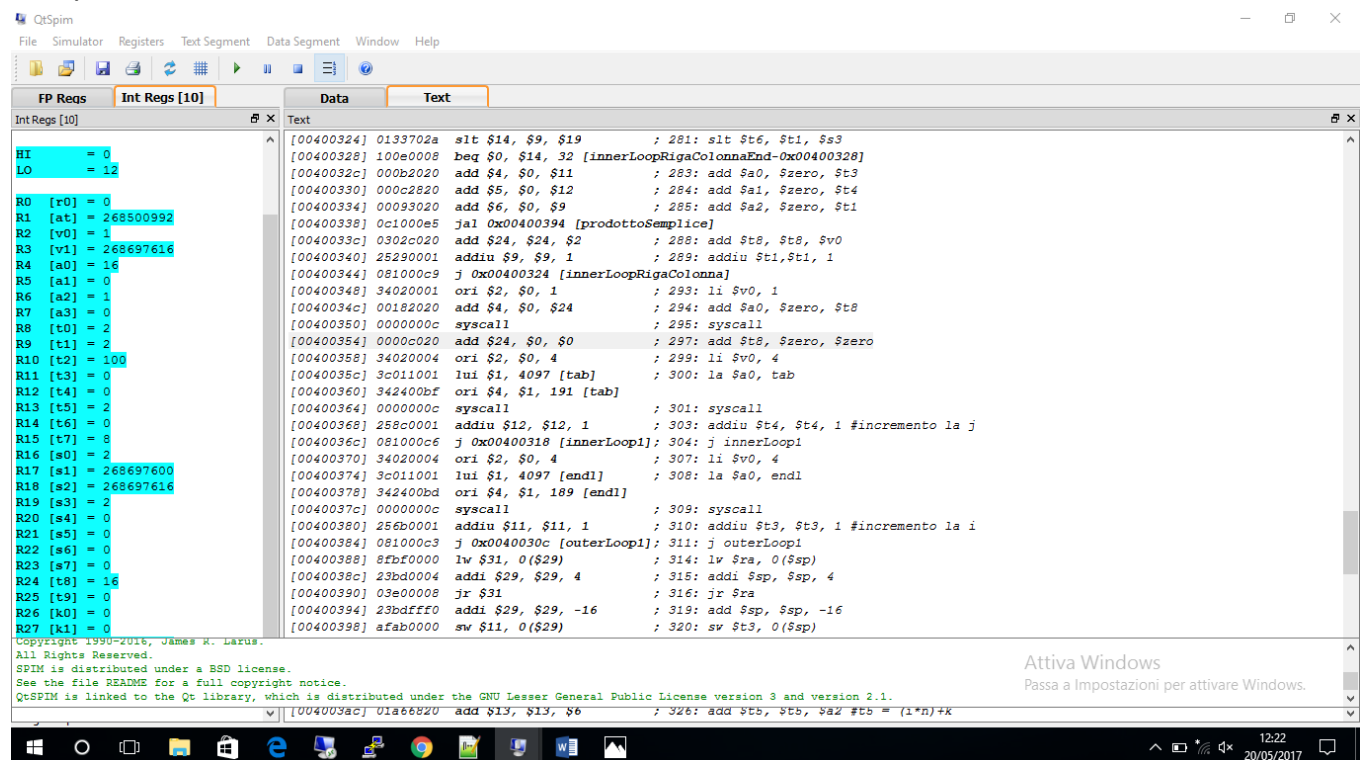
```
[004003d0] 71ae1002 mul $2, $13, $14 ; 337: mul $v0, $t5, $t6
```

The bottom status bar shows the date 20/05/2017 and the time 12:19.

Salvo i valori degli elementi delle matrici trovati:

This screenshot is identical to the one above, showing the QtSpim MIPS simulator interface with the same register values and assembly code. The highlighted instruction is the same: `[004003d0] 71ae1002 mul $2, $13, $14 ; 337: mul $v0, $t5, $t6`. The bottom status bar shows the date 20/05/2017 and the time 12:20.

Prodotto riga per colonna:  $(4 \times 1) + (2 \times 6) = 16 \rightarrow$  valore contenuto nel registro temporaneo “\$t8”:



I passaggi successivi sono identici, cambiano i valori.

Risultato finale della moltiplicazione tra le 2 matrici A e B:

```
a) Inserting matrix
b) Add A + B
c) Subtraction A-B
d) Product A*B
e) Exit
d
16          19
18          21
```