



**UNIVERSITÀ  
DEGLI STUDI  
DI TRIESTE**

# Introduzione a QtSpim

**Prof.ssa Giulia Cisotto**

[giulia.cisotto@units.it](mailto:giulia.cisotto@units.it)

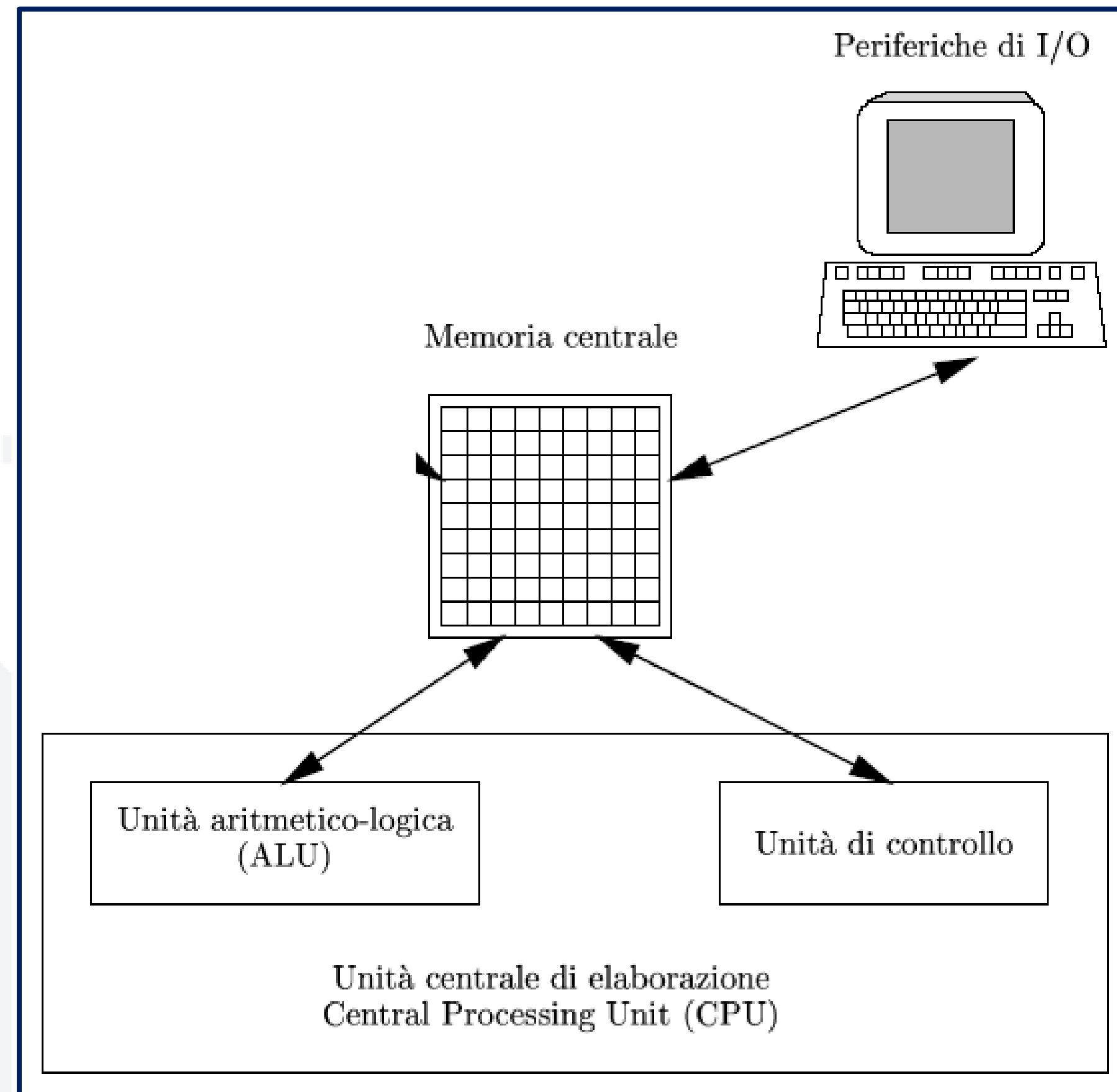
Trieste, 3 marzo 2025



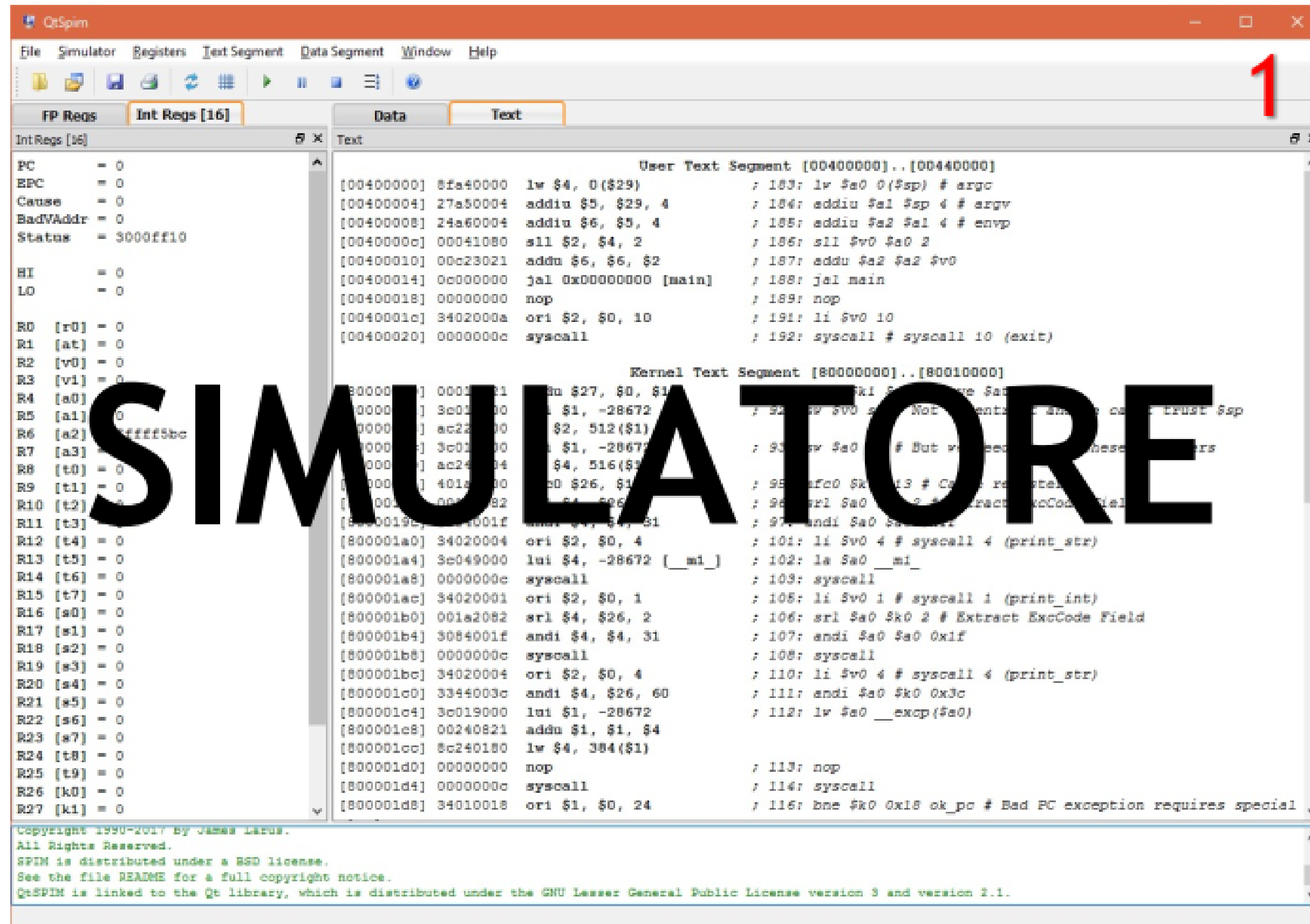
**QtSpim**

<https://spimsimulator.sourceforge.net/>

# COSA SIMULA?



# L'interfaccia del simulatore QtSpim è composta da due parti principali



**SIMULATORE**

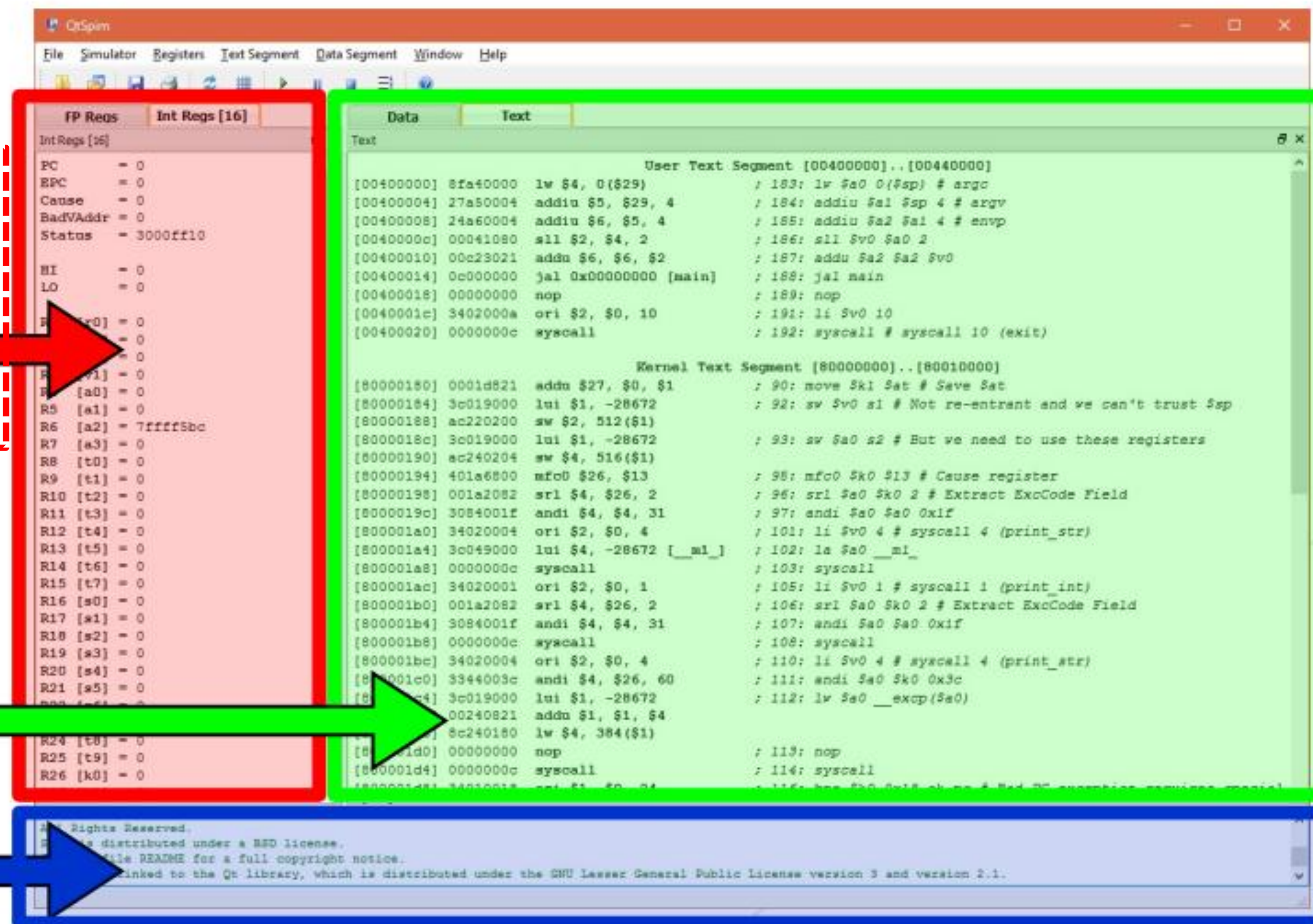
# REGISTRI e MEMORIA

LA FINESTRA PRINCIPALE È SUDDIVISA IN TRE PARTI

IL PANNELLO SULLA SINISTRA VISUALIZZA I REGISTRI FLOATING POINT oppure I REGISTRI INTERI, A SECONDA DELLA SCHEDA IN PRIMO PIANO

IL PANNELLO SULLA DESTRA VISUALIZZA IL TEXT SEGMENT (CONTIENE LE ISTRUZIONI MACCHINA) ED IL DATA SEGMENT (CONTIENE I DATI IN MEMORIA), A SECONDA DELLA SCHEDA IN PRIMO PIANO

IL PANNELLO INFERIORE CONTIENE I MESSAGGI DEL SIMULATORE



Abbiamo a disposizione 32 registri INTERI e 32 registri FLOATING POINT (codifica IEEE 754)

# ESEMPIO DI INTERFACCIA

The screenshot displays the QtSpim MIPS simulator interface. The window title is "QtSpim" and the menu bar includes "File", "Simulator", "Registers", "Text Segment", "Data Segment", "Window", and "Help". The interface is divided into several panes:

- Registers:** Shows the state of various registers. For example, PC, EPC, Cause, BadVAddr, and Status are all 0. HI and LO are also 0. R0-R20 are shown with their names and values (mostly 0). R4 [a0] is 111, R5 [a1] is 111111111111111110110, and R6 [a2] is 111111111111111110110.
- Data Segment:** Shows memory contents. The "User data segment" is at [10000000]..[10040000] with value 00000000. The "User Stack" is at [7ffff62c]..[80000000]. The stack contains assembly code, including ".l i n k e r / m e a n \_ l i b", ".w o r d . a s m . c o n . A s", and ".s e m b l y . Q t S p i m . - .".
- Assembly Code:** The bottom pane shows the assembly code being executed, including instructions like "w o v 1", "i t s", "e p d", "o m", "a t n", and "-".

At the bottom of the window, there is a status bar with the following text:

```

Memory and registers cleared
SPIM Version 9.1.24 of August 1, 2023 (final)
Copyright 1990-2023 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.
    
```



# QtSpim

In questa sezione della memoria verranno memorizzate i dati dell'utente (che si carica, all'avvio di QtSpim, per poterlo eseguire)



QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [2] Data Text

Int Regs [2]

PC = 0  
EPC = 0  
Cause = 0  
BadVAddr = 0  
Status = 110000000000001111111110  
0010000

HI = 0  
LO = 0

R0 [r0] = 0  
R1 [at] = 0  
R2 [v0] = 0  
R3 [v1] = 0  
R4 [a0] = 111  
R5 [a1] = 111111111111111110110  
00110000  
R6 [a2] = 111111111111111110110  
01010000  
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

User data segment [10000000]..[10040000]  
[10000000]..[1003ffff] 00000000

User Stack [7ffff62c]..[80000000]  
[7ffff62c] 00000007  
[7ffff630] 7ffff740 7ffff730 7ffff72e 7ffff727 . . . .  
[7ffff640] 7ffff71e 7ffff71a 7ffff701 00000000 @ . . . 0 . . . . ' . . .  
[7ffff650] 7fffffe1 7fffffba 7fffff89 7fffff4d . . . . . . . . M . . .  
[7ffff660] 7fffff1c 7ffffeff 7ffffedb 7ffffea9 . . . . . . . . . . . .  
[7ffff670] 7ffffe9d 7ffffe6c 7ffffe44 7ffffe37 . . . . l . . . D . . . 7 . . .  
[7ffff680] 7ffffe21 7ffffc75 7ffffc4b 7ffffc2d ! . . . u . . . K . . . - . . .  
[7ffff690] 7ffffc15 7ffffbf4 7ffffbe6 7ffffa66 . . . . . . . . . . f . . .  
[7ffff6a0] 7ffffa28 7ffffa0b 7ffff9c2 7ffff9b0 ( . . . . . . . . . . . . . . . .  
[7ffff6b0] 7ffff998 7ffff97d 7ffff95f 7ffff936 . . . . } . . . \_ . . . 6 . . .  
[7ffff6c0] 7ffff918 7ffff8ad 7ffff896 7ffff882 . . . . . . . . . . . . . . . .  
[7ffff6d0] 7ffff873 7ffff85d 7ffff836 7ffff810 s . . . ] . . . 6 . . . . . . . .  
[7ffff6e0] 7ffff7f5 7ffff7cb 7ffff7bc 7ffff7a1 . . . . . . . . . . . . . . . .  
[7ffff6f0] 7ffff78f 7ffff77b 00000000 00000000 . . . { . . . . . . . . . . . . . . . .  
[7ffff700] 6e696c00 2f72656b 6e61656d 62696c5f . l i n k e r / m e a n \_ l i b  
[7ffff710] 726f775f 73612e64 6f63006d 7341006e \_ w o r d . a s m . c o n . A s  
[7ffff720] 626d6573 5100796c 69705374 002d006d \_ s e m b l y . Q t S p i m . - .  
[7ffff730] 33323032 3230322d 614c2f34 00342362 2 0 2 3 - 2 0 2 4 / L a b # 4 .  
[7ffff740] 552f3a43 73726573 7569672f 4f2f696c C : / U s e r s / g i u l i / O  
[7ffff750] 7244656e 2f657669 75636f44 746e656d n e D r i v e / D o c u m e n t  
[7ffff760] 49442f69 54544144 2f414349 6d696e55 i / D I D A T T I C A / U n i m  
[7ffff770] 412f6269 2f686372 5a006465 455f5345 i b / A r c h / e d . Z E S \_ E  
[7ffff780] 4c42414e 59535f45 4e414d53 7700313d N A B L E \_ S Y S M A N = 1 . w  
[7ffff790] 69646e69 3a433d72 4e49575c 53574f44 i n d i r = C : \ W I N D O W S  
[7ffff7a0] 45535500 4f525052 454c4946 5c3a433d . U S E R P R O F I L E = C : \  
[7ffff7b0] 72657355 69675c73 00696c75 52455355 U s e r s \ g i u l i . U S E R  
[7ffff7c0] 454d414e 7569673d 5500696c 44524553 N A M E = g i u l i . U S E R D  
[7ffff7d0] 49414d4f 4f525f4e 4e494d41 4f525047 O M A I N \_ R O A M I N G P R O  
[7ffff7e0] 454c4946 5549473d 5f41494c 4f4e454c F I L E = G I U L I A \_ L E N O  
[7ffff7f0] 31584f56 45535500 4d4f4a52 3d4c4941 W O V 1 I I S E P D O M A T N -

Memory and registers cleared

SPIM Version 9.1.24 of August 1, 2023 (final)  
Copyright 1990-2023 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.

# QtSpim



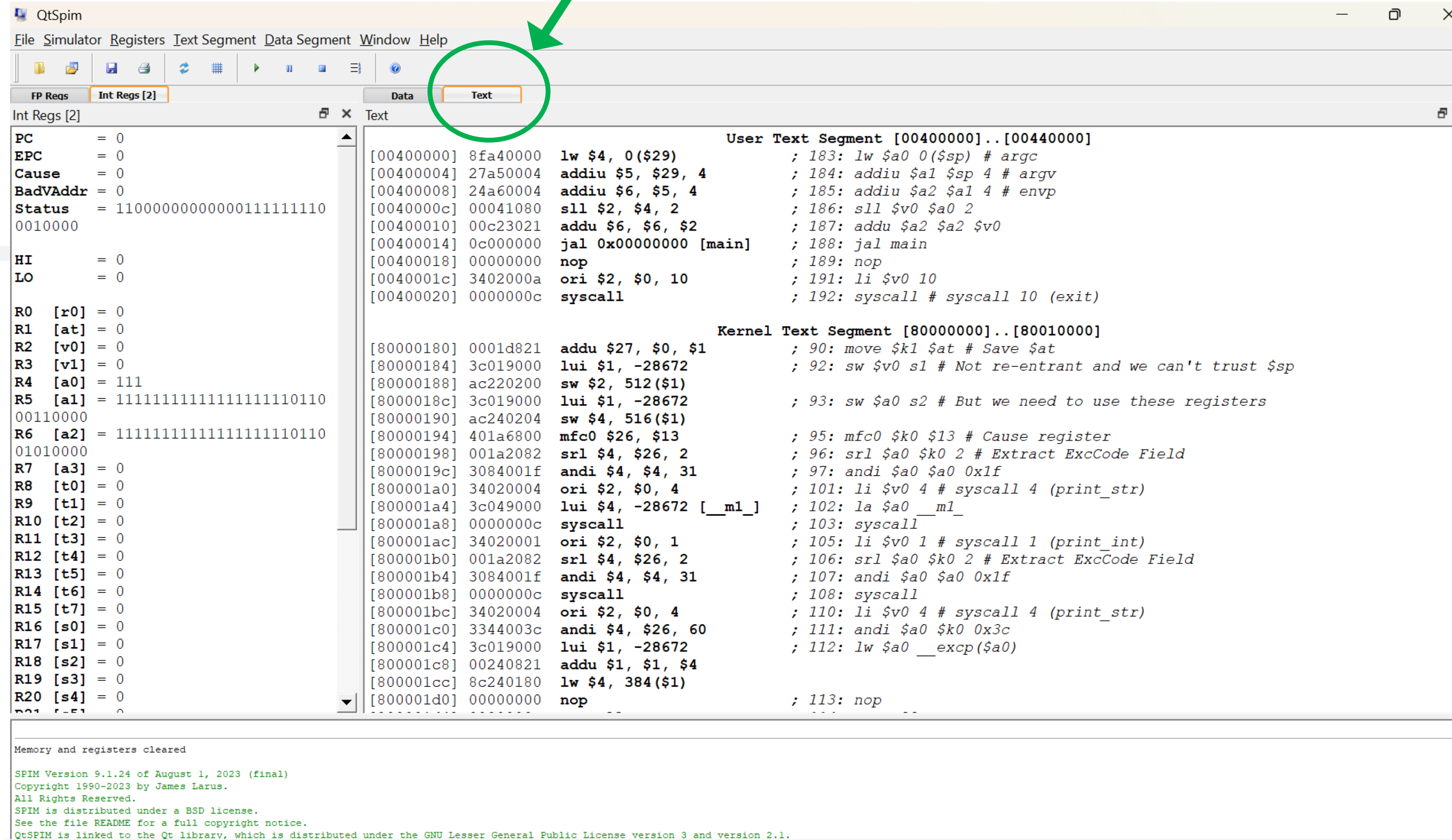
UNIVERSITÀ  
DEGLI STUDI  
DI TRIESTE

# QtSpim

In questa sezione della memoria verranno memorizzate le **istruzioni associate al programma dell'utente** (che si carica, all'avvio di QtSpim, per poterlo eseguire)



## QtSpim



```
QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Reas Int Regs [2] Data Text
Int Regs [2]
PC = 0
EPC = 0
Cause = 0
BadVAddr = 0
Status = 11000000000000111111110
0010000
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 0
R3 [v1] = 0
R4 [a0] = 111
R5 [a1] = 11111111111111110110
00110000
R6 [a2] = 11111111111111110110
01010000
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

User Text Segment [00400000]..[00440000]
[00400000] 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
[00400010] 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0
[00400014] 0c000000 jal 0x00000000 [main] ; 188: jal main
[00400018] 00000000 nop ; 189: nop
[0040001c] 3402000a ori $2, $0, 10 ; 191: li $v0 10
[00400020] 0000000c syscall ; 192: syscall # syscall 10 (exit)

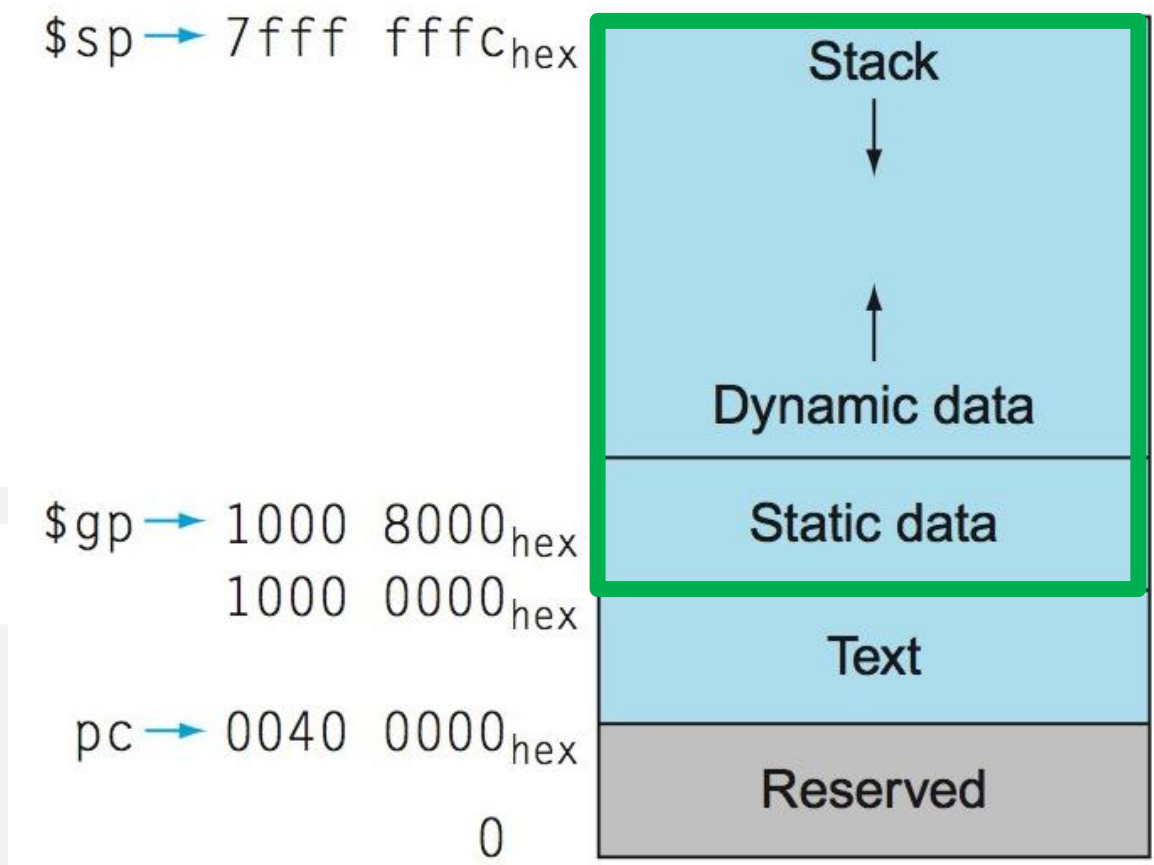
Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821 addu $27, $0, $1 ; 90: move $k1 $at # Save $at
[80000184] 3c019000 lui $1, -28672 ; 92: sw $v0 $1 # Not re-entrant and we can't trust $sp
[80000188] ac220200 sw $2, 512($1)
[8000018c] 3c019000 lui $1, -28672 ; 93: sw $a0 $2 # But we need to use these registers
[80000190] ac240204 sw $4, 516($1)
[80000194] 401a6800 mfc0 $26, $13 ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082 srl $4, $26, 2 ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f andi $4, $4, 31 ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004 ori $2, $0, 4 ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000 lui $4, -28672 [__m1_] ; 102: la $a0 __m1_
[800001a8] 0000000c syscall ; 103: syscall
[800001ac] 34020001 ori $2, $0, 1 ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082 srl $4, $26, 2 ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f andi $4, $4, 31 ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c syscall ; 108: syscall
[800001bc] 34020004 ori $2, $0, 4 ; 110: li $v0 4 # syscall 4 (print_str)
[800001c0] 3344003c andi $4, $26, 60 ; 111: andi $a0 $k0 0x3c
[800001c4] 3c019000 lui $1, -28672 ; 112: lw $a0 __excp($a0)
[800001c8] 00240821 addu $1, $1, $4
[800001cc] 8c240180 lw $4, 384($1)
[800001d0] 00000000 nop ; 113: nop

Memory and registers cleared

SPIM Version 9.1.24 of August 1, 2023 (final)
Copyright 1990-2023 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.
```



# QtSpim



```

Data  Text
Text
[00400000] 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
[00400010] 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0
[00400014] 0c000000 jal 0x00000000 [main] ; 188: jal main
[00400018] 00000000 nop ; 189: nop
[0040001c] 3402000a ori $2, $0, 10 ; 191: li $v0 10
[00400020] 0000000c syscall ; 192: syscall # syscall 10 (exit)

Kernel Text Segment [80000000]..[80010000]
[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) ; 93: sw $a0 $2 # But we need to use these registers
[8000018c] 3c019000 lui $1, -28672
[80000190] ac240204 sw $4, 516($1)
[80000194] 401a6800 mfc0 $26, $13 ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082 srl $4, $26, 2 ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f andi $4, $4, 31 ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004 ori $2, $0, 4 ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000 lui $4, -28672 [__ml_] ; 102: la $a0 __ml_
[800001a8] 0000000c syscall ; 103: syscall
[800001ac] 34020001 ori $2, $0, 1 ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082 srl $4, $26, 2 ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f andi $4, $4, 31 ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c syscall ; 108: syscall
[800001bc] 34020004 ori $2, $0, 4 ; 110: li $v0 4 # syscall 4 (print_str)
[800001c0] 3344003c andi $4, $26, 60 ; 111: andi $a0 $k0 0x3c
[800001c4] 3c019000 lui $1, -28672 ; 112: lw $a0 __exc($a0)
[800001c8] 00240821 addu $1, $1, $4
[800001cc] 8c240180 lw $4, 384($1)
[800001d0] 00000000 nop ; 113: nop
  
```

```

Data
User data segment [10000000]..[10040000]
[10000000]..[003fffff] 00000000

User Stack [7ffff62c]..[80000000]
[7ffff62c] 00000007
[7ffff630] 7ffff740 7ffff730 7ffff72e 7ffff727 . . . .
[7ffff640] 7ffff71e 7ffff71a 7ffff701 00000000 @ . . . 0 . . . . . ! . . .
[7ffff650] 7fffffe1 7fffffba 7fffff89 7fffff4d . . . . . M . . .
[7ffff660] 7fffff1c 7ffffeff 7ffffedb 7ffffea9 . . . . .
[7ffff670] 7ffffe9d 7ffffe6c 7ffffe44 7ffffe37 . . . . l . . . D . . . 7 . . .
[7ffff680] 7ffffe21 7ffffc75 7ffffc4b 7ffffc2d ! . . . u . . . K . . . - . . .
[7ffff690] 7ffffc15 7ffffbf4 7ffffbe6 7ffffa66 . . . . . f . . .
[7ffff6a0] 7ffffa28 7ffffa0b 7ffff9c2 7ffff9b0 ( . . . . .
[7ffff6b0] 7ffff998 7ffff97d 7ffff95f 7ffff936 . . . . . ) . . . _ . . . 6 . . .
[7ffff6c0] 7ffff918 7ffff8ad 7ffff896 7ffff882 . . . . .
[7ffff6d0] 7ffff873 7ffff85d 7ffff836 7ffff810 s . . . . ] . . . 6 . . . . .
[7ffff6e0] 7ffff7f5 7ffff7cb 7ffff7bc 7ffff7a1 . . . . .
[7ffff6f0] 7ffff78f 7ffff77b 00000000 00000000 . . . . { . . . . .
[7ffff700] 6e696c00 2f72656b 6e61656d 62696c5f . l i n k e r / m e a n _ l i b
[7ffff710] 726f775f 73612e64 6f63006d 7341006e _ w o r d . a s m . c o n . A s
[7ffff720] 626d6573 5100796c 69705374 002d006d s e m b l y . Q t S p i m . - .
[7ffff730] 3323032 3230322d 614c2f34 00342362 2 0 2 3 - 2 0 2 4 / L a b # 4 .
[7ffff740] 552f3a43 73726573 7569672f 4f2f696c C : / U s e r s / g i u l i / O
[7ffff750] 7244656e 2f657669 75636f44 746e656d n e D r i v e / D o c u m e n t
[7ffff760] 49442f69 54544144 2f414349 6d696e55 i / D I D A T T I C A / U n i m
[7ffff770] 412f6269 2f686372 5a006465 455f5345 i b / A r c h / e d . Z E S _ E
[7ffff780] 4c42414e 59535f45 4e414d53 7700313d N A B L E _ S Y S M A N = 1 . w
[7ffff790] 69646e69 3a433d72 4e49575c 53574f44 i n d i r = C : \ W I N D O W S
[7ffff7a0] 45535500 4f525052 454c4946 5c3a433d . U S E R P R O F I L E = C : \
[7ffff7b0] 72657355 69675c73 00696c75 52455355 U s e r s \ g i u l i . U S E R
[7ffff7c0] 454d414e 7569673d 5500696c 44524553 N A M E = g i u l i . U S E R D
[7ffff7d0] 49414d4f 4f525f4e 4e494d41 4f525047 O M A I N _ R O A M I N G P R O
[7ffff7e0] 454c4946 5549473d 5f41494c 4f4e454c F I L E = G I U L I A _ L E N O
[7ffff7f0] 31594f56 45535500 424f4452 3d4c4941 v o y 1 U S E R P R O M A T N -
  
```

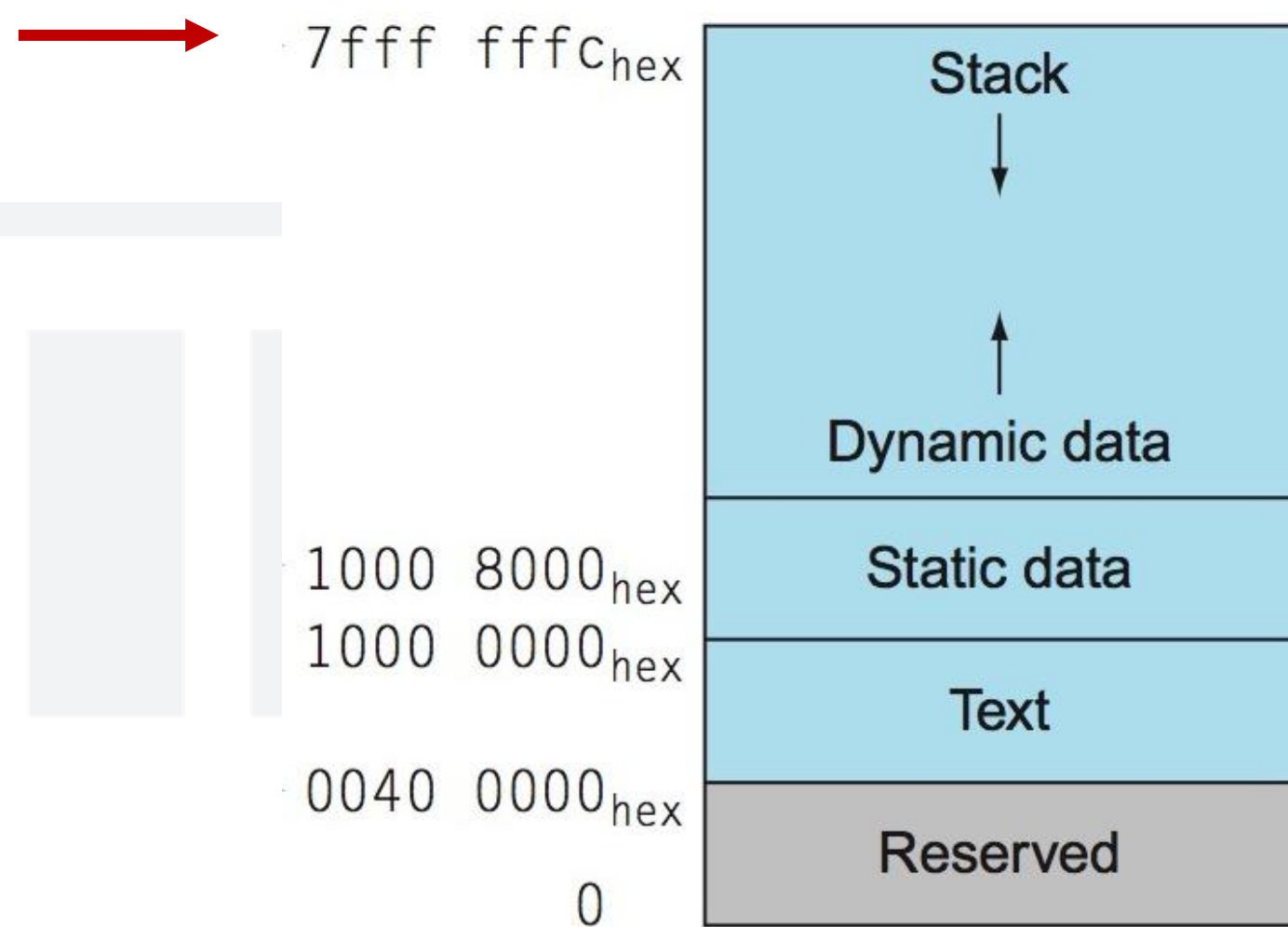
In questa porzione di memoria verranno memorizzate i **dati dell'utente** (caricati in memoria centrale), le **variabili temporanee** che servono durante l'esecuzione di **procedure e altro** (si faccia riferimento alle prossime lezioni).



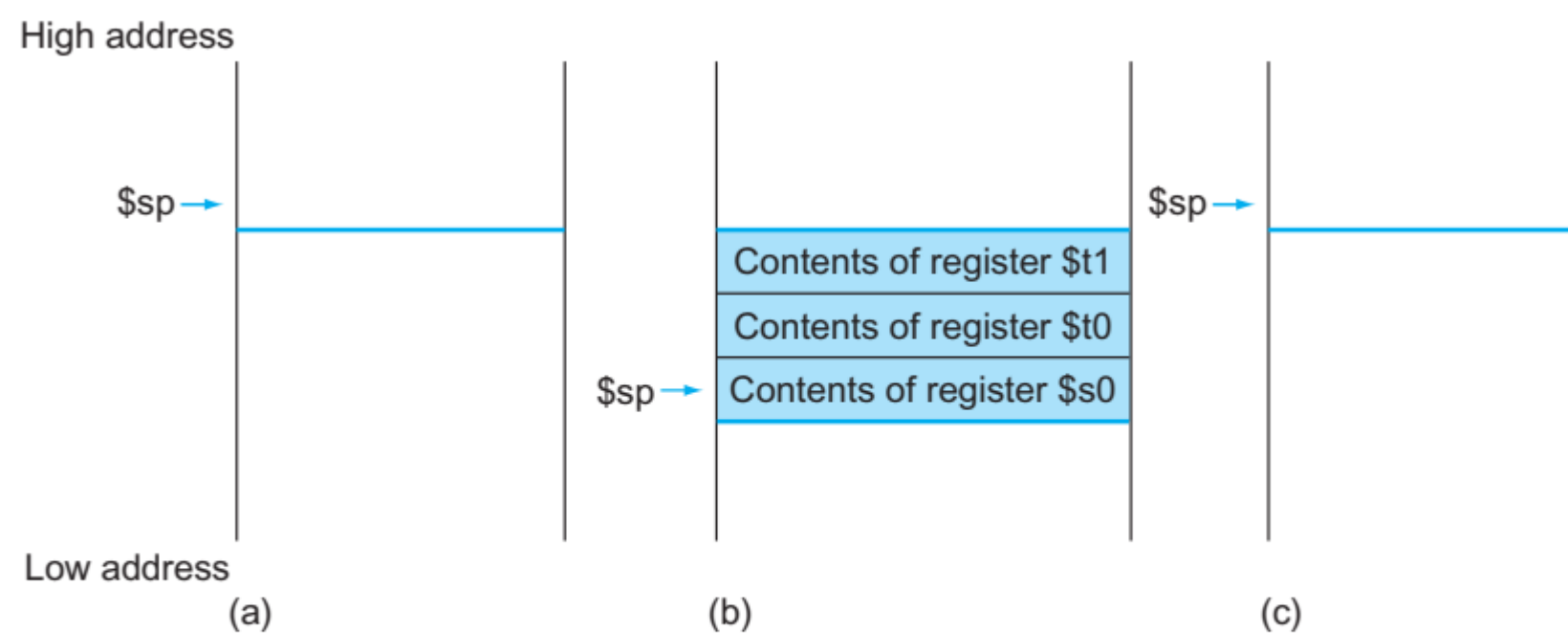
# QtSpim

# STACK POINTER (\$sp)

Lo **Stack Pointer** è un registro (in MIPS è \$sp) che indica l'**indirizzo della cima** dello stack in memoria.



Lo stack è un'area di memoria utilizzata per memorizzare temporaneamente dati (ad esempio variabili locali, indirizzi di ritorno di funzioni, parametri di funzioni, ecc.).



**FIGURE 2.10** The values of the stack pointer and the stack (a) before, (b) during, and (c) after the procedure call. The stack pointer always points to the “top” of the stack, or the last word in the stack in this drawing.

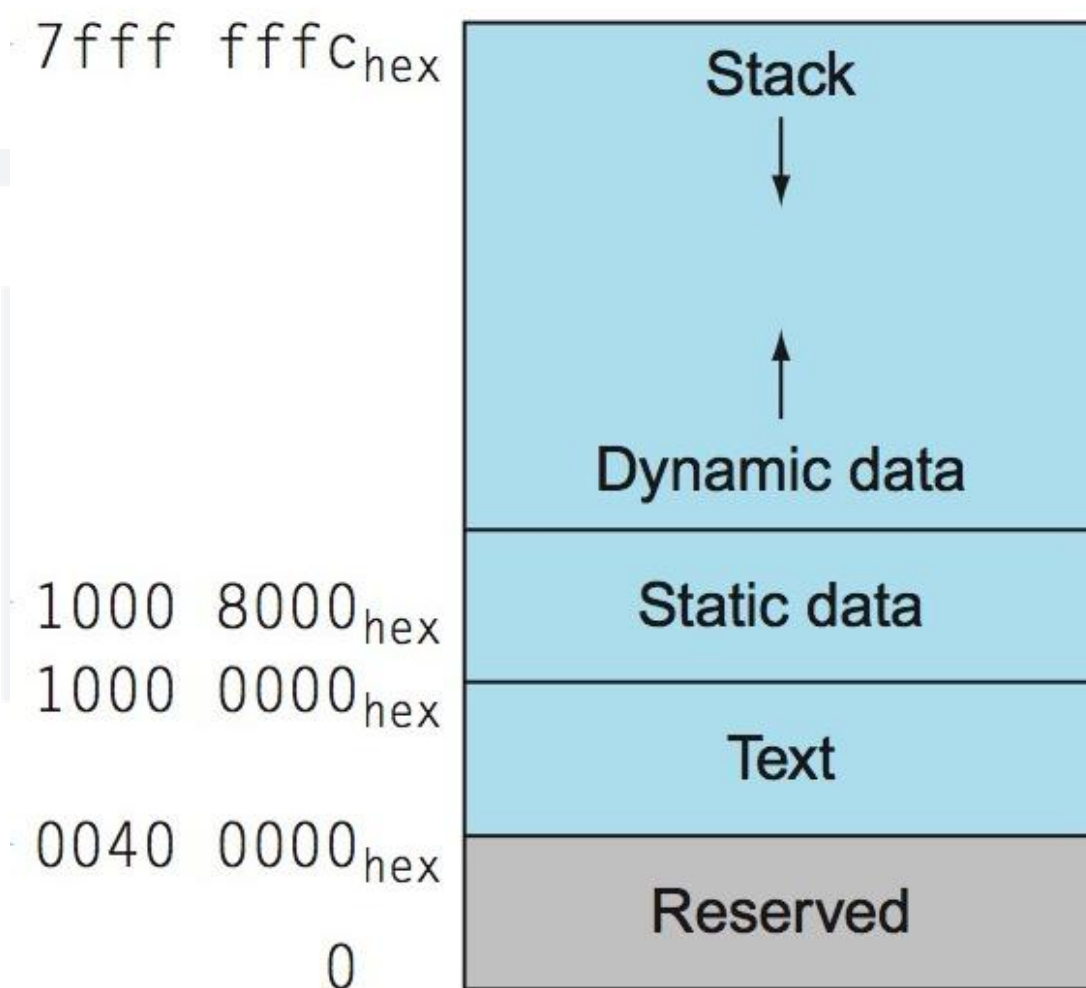
```

QtSpim
File Simulator Registers Text Segment Data Segment Window
FP Regs Int Regs [2] Data
Int Regs [2]
R2 [v0] = 0
R3 [v1] = 0
R4 [a0] = 111
R5 [a1] = 111111111111111110110
00110000
R6 [a2] = 111111111111111110110
01010000
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
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 1000000000000100000000
000000
R29 [sp] = 111111111111111110110
00101100
R30 [s8] = 0
R31 [ra] = 0
    
```

**Cresce verso indirizzi più bassi:** ogni volta si aggiunge qualcosa allo stack, \$sp viene decrementato.

# GLOBAL POINTER (\$gp)

È un registro speciale in MIPS32 che contiene un **indirizzo di riferimento** per accedere rapidamente alle variabili globali in memoria.



È utile per accedere a dati statici/globali con **indirizzamento breve e veloce**.

Solitamente è inizializzato a un indirizzo centrale nel segmento dati, permettendo di accedere facilmente (con un offset piccolo) alle variabili globali più comuni.

The screenshot shows the QtSpim simulator's register file. The registers are listed as follows:

Register	Value
R2 [v0]	= 0
R3 [v1]	= 0
R4 [a0]	= 111
R5 [a1]	= 1111111111111111111011000110000
R6 [a2]	= 11111111111111111110110001010000
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
R23 [s7]	= 0
R24 [t8]	= 0
R25 [t9]	= 0
R26 [k0]	= 0
R27 [k1]	= 0
R28 [gp]	= 100000000000010000000000000000
R29 [sp]	= 1111111111111111111011000101100
R30 [s8]	= 0
R31 [ra]	= 0

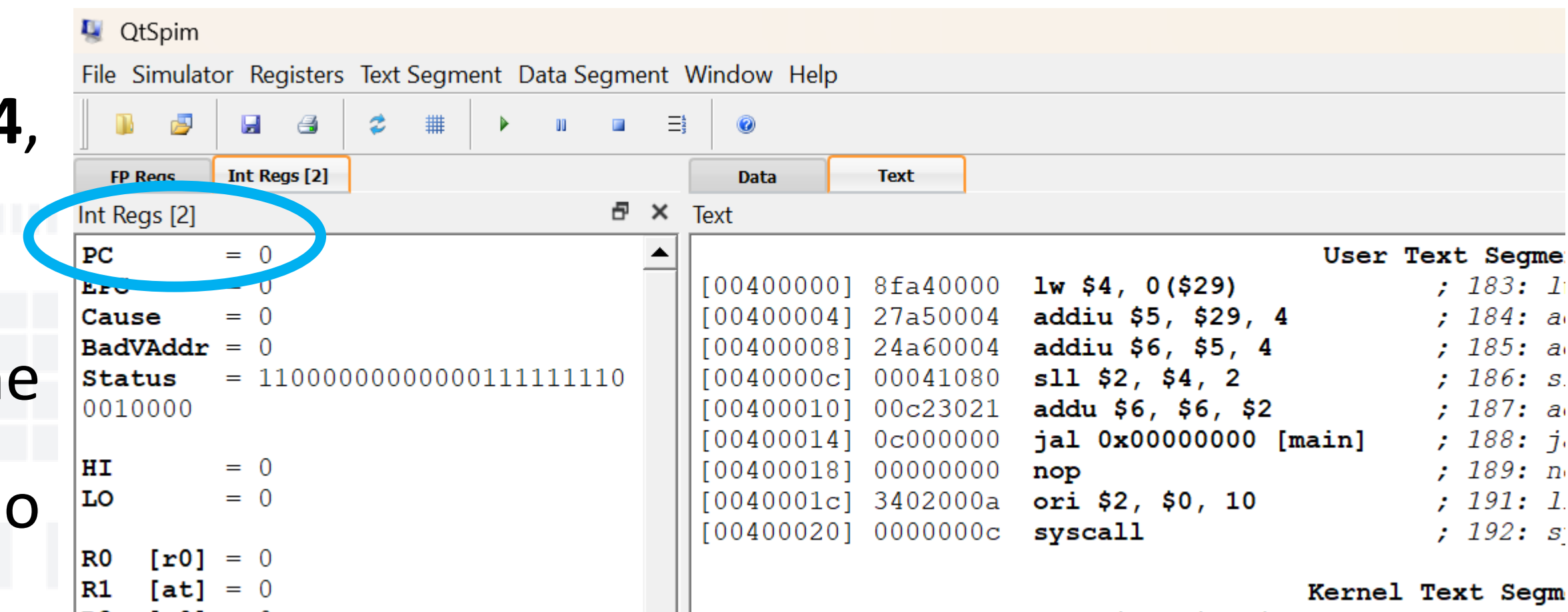
The register `R28 [gp]` is circled in green, indicating its value, which is the base address of the static data segment.

Accesso alle variabili globali più efficiente rispetto all'indirizzamento assoluto.

# PROGRAM COUNTER (PC)

Il **Program Counter (PC)** in MIPS32 è un registro speciale della CPU che ha il compito di tenere traccia dell'**indirizzo della prossima istruzione** da eseguire.

- È un registro a **32 bit**.
- L'indirizzo in esso contenuto è sempre **multiplo di 4**, dato che ogni istruzione MIPS32 occupa **4 byte**.
- Dopo che un'istruzione viene letta (*fetch*), il PC viene automaticamente incrementato di **4**, passando all'istruzione successiva.



```
QtSpim
File Simulator Registers Text Segment Data Segment Window Help
Int Regs [2]
PC = 0
Cause = 0
BadVAddr = 0
Status = 1100000000000000111111110
0010000
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 0
Text
[00400000] 8fa40000 lw $4, 0($29) ; 183: l
[00400004] 27a50004 addiu $5, $29, 4 ; 184: a
[00400008] 24a60004 addiu $6, $5, 4 ; 185: a
[0040000c] 00041080 sll $2, $4, 2 ; 186: s
[00400010] 00c23021 addu $6, $6, $2 ; 187: a
[00400014] 0c000000 jal 0x00000000 [main] ; 188: j
[00400018] 00000000 nop ; 189: n
[0040001c] 3402000a ori $2, $0, 10 ; 191: l
[00400020] 0000000c syscall ; 192: s
Kernel Text Segm
```

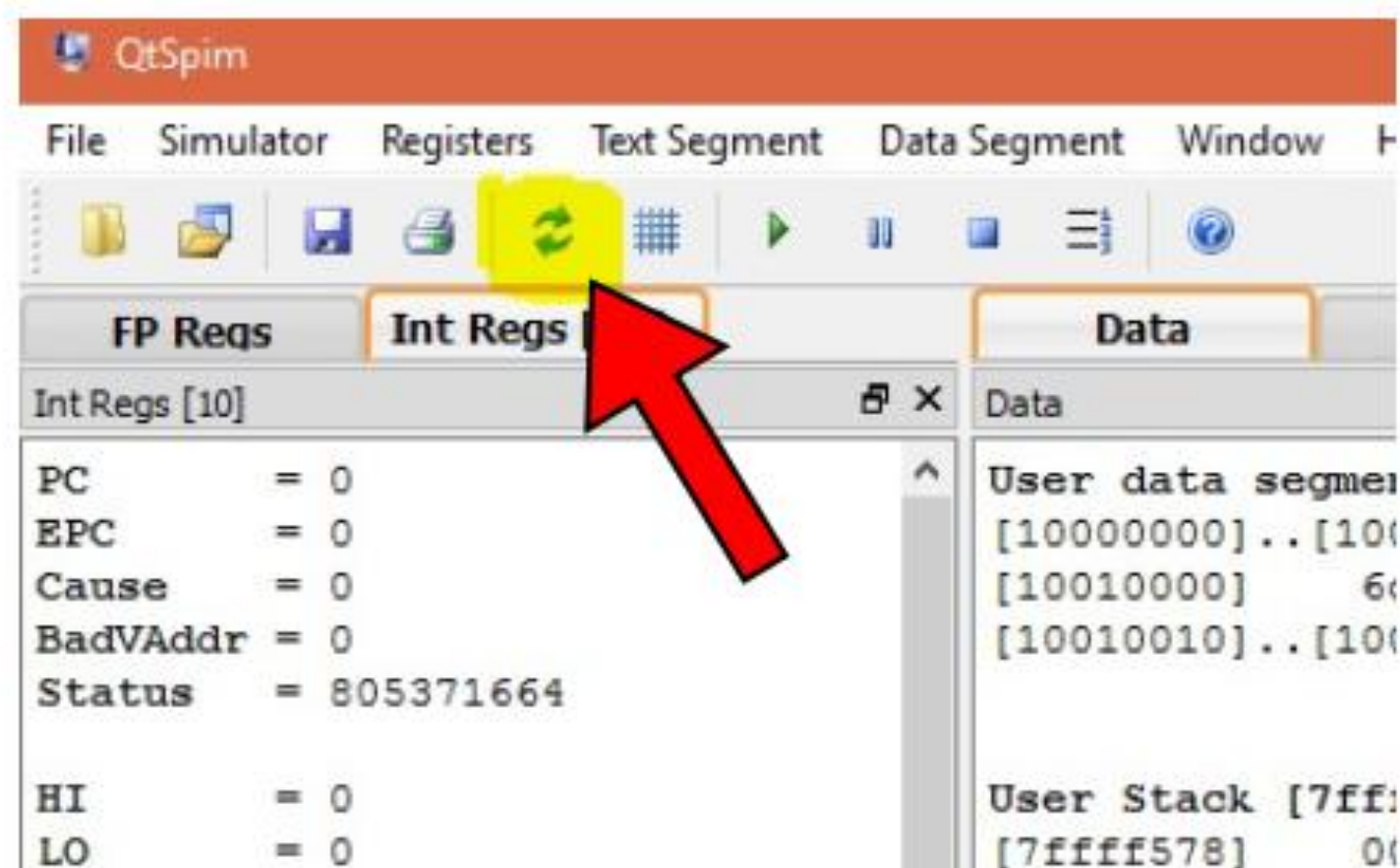
**Istruzioni sequenziali (la maggior parte):**  $PC = PC + 4$

**Istruzioni speciali** (branch, jump, chiamate funzioni) modificano il PC diversamente.

# QtSpim

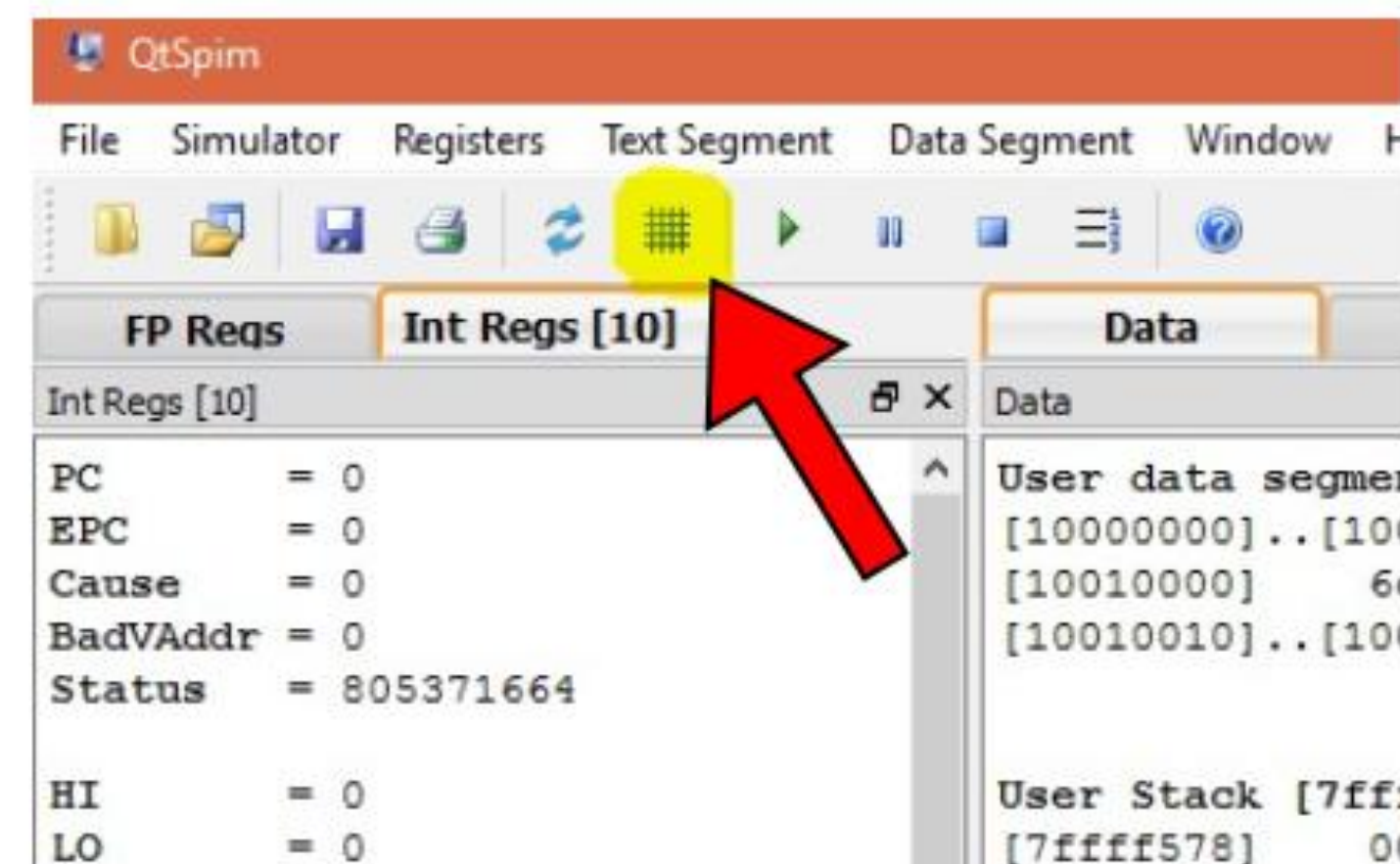


## QtSpim



**PULSANTE  
«CLEAR REGISTER»**

**CANCELLA IL CONTENUTO DEI  
REGISTRI, MA NON  
LA MEMORIA (AREA DATI)**



**PULSANTE  
«REINITIALIZE SIMULATOR»**

**REIMPOSTA LA MEMORIA  
(AREA DATI), E  
I REGISTRI**

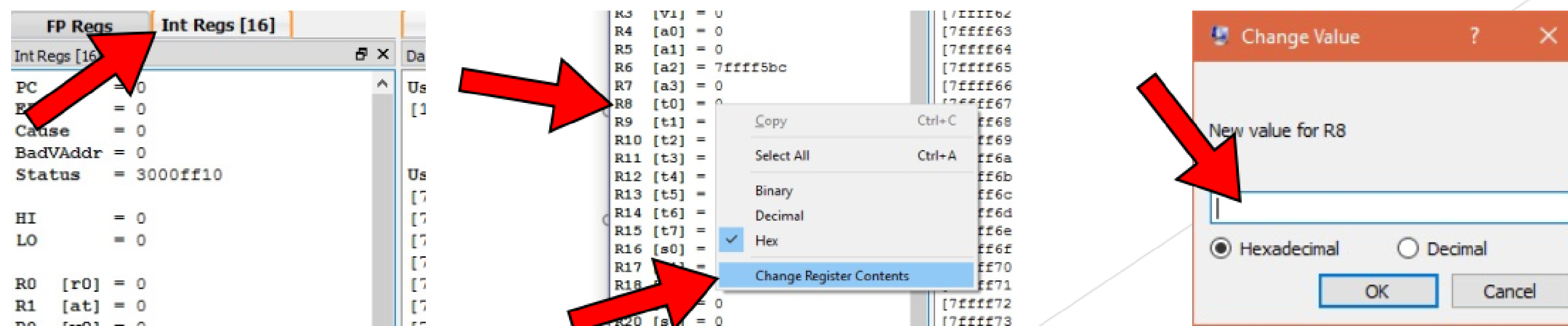
# QtSpim



## QtSpim

### COME CAMBIARE IL VALORE DI UN REGISTRO (e.g. R8)?

- SELEZIONARE LA SCHEDA DEI REGISTRI DEGLI INTERI «INT REGS [16]» ED INDIVIDUARE IL REGISTRO R8 (Register8)
- FARE CLICK CON IL PULSANTE DESTRO IN CORRISPONDENZA DEL REGISTRO R8
- SELEZIONARE «CHANGE REGISTER CONTENTS»
- INSERIRE IL VALORE (IN FORMATO HEX oppure DEC)



# QtSpim



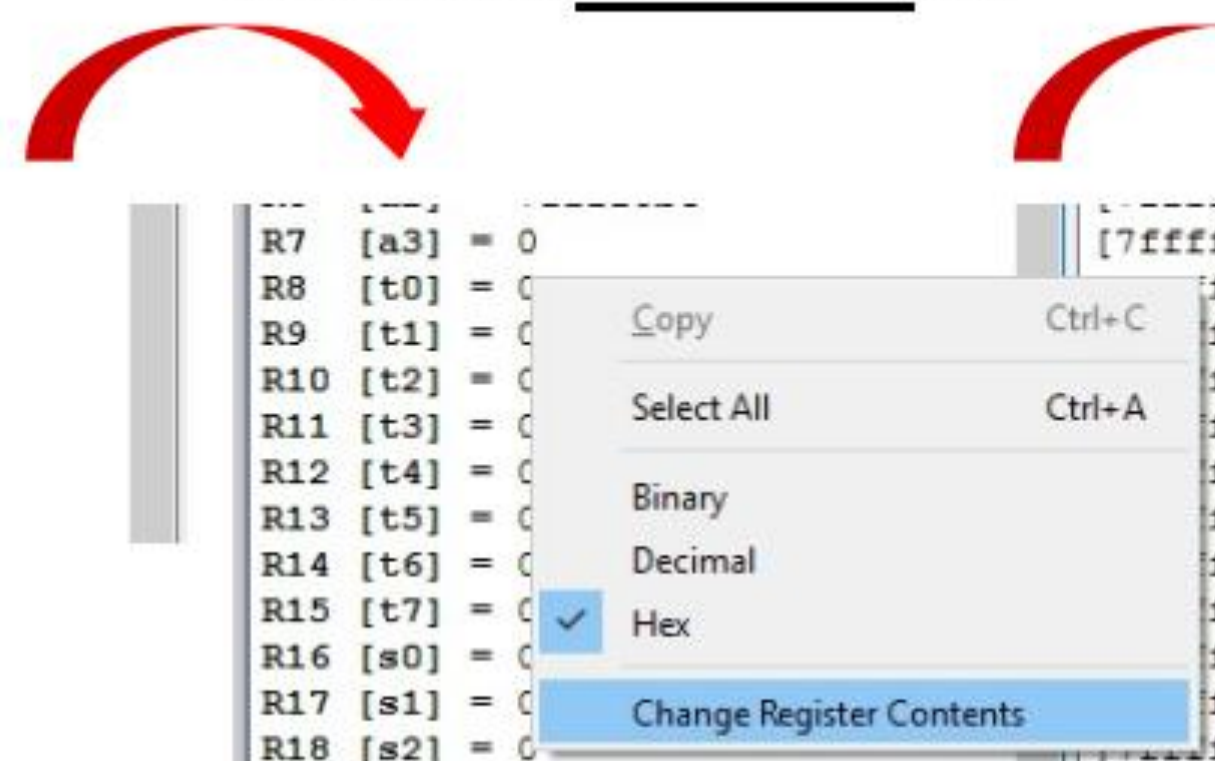
**QtSpim**

## ESEMPIO: IMPOSTARE R8 CON IL VALORE 0x3

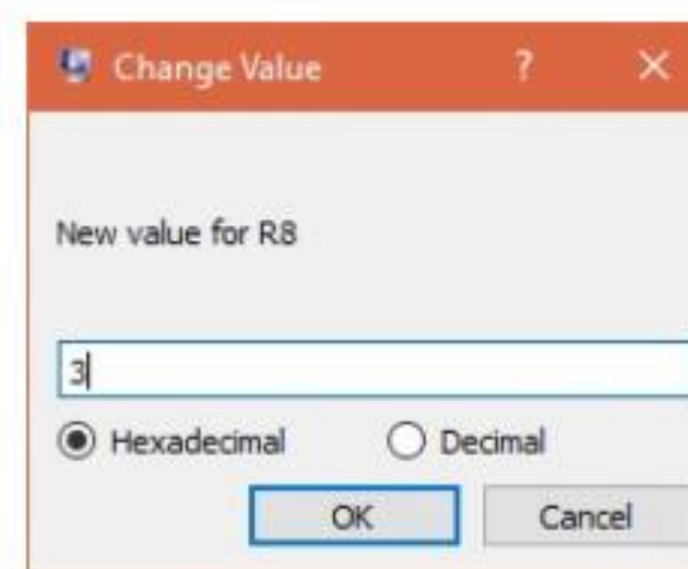
1  
INDIVIDUO R8

```
R4 [a0] = 0
R5 [a1] = 0
R6 [a2] = 7ffff5bc
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
```

2  
CLICK DESTRO +  
«CHANGE REGISTER...»



3  
SCRIVO IL NUMERO



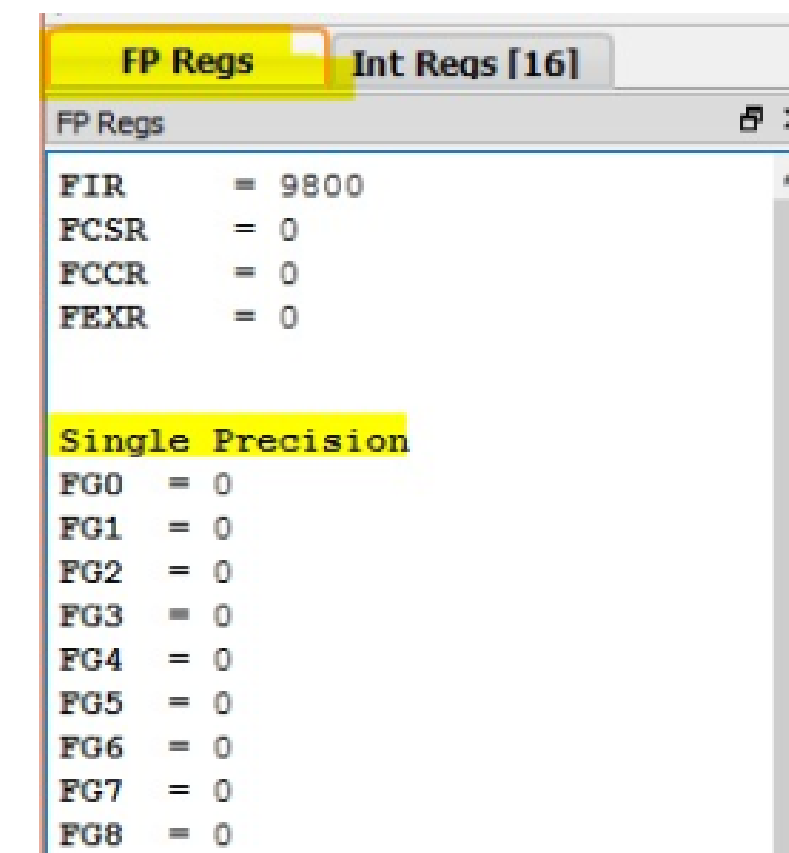
4  
RISULTATO

```
R4 [a0] = 0
R5 [a1] = 0
R6 [a2] = 7ffff5bc
R7 [a3] = 0
R8 [t0] = 3
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
```

# QtSpim

## REGISTRI FLOATING POINT SINGLE PRECISION (VIRGOLA MOBILE, SINGOLA PRECISIONE)

- SI VISUALIZZANO ABILITANDO LA SCHEDA «FP Regs»
- SI MODIFICANO ALLO STESSO MODO DEI REGISTRI INTERI
- LA RAPPRESENTAZIONE DEI NUMERI SEGUE LO STANDARD IEEE754



```

FP Regs  Int Regs [16]
FP Regs
FIR      = 9800
FCSR     = 0
FCCR     = 0
FEXR     = 0

Single Precision
FG0      = 0
FG1      = 0
FG2      = 0
FG3      = 0
FG4      = 0
FG5      = 0
FG6      = 0
FG7      = 0
FG8      = 0
  
```

N.B. il contenuto del registro è visualizzato a partire dalla prima cifra diversa da 0



# QtSpim