

Ejercicios de la práctica 1.

a) Identificar en el emulador los siguientes elementos. (Prepara en el informe el número de capturas de pantalla que necesites para señalar mediante un cuadro rojo cada una de ellas):

I. El segmento de Datos (Usuario, Kernel y Pila)

1ºCuadro = USUARIO// 2ºCuadro = PILA // 3ºCuadro = KERNEL

The image displays three screenshots of an emulator's memory viewer, each showing a different segment of memory. The first screenshot shows the 'User data segment' from address [10000000] to [10040000]. The second screenshot shows the 'User Stack' from address [7ffff670] to [80000000]. The third screenshot shows the 'Kernel data segment' from address [90000000] to [90010000]. Each screenshot has a red box highlighting the memory content.

User data segment [10000000]..[10040000]

```
[10000000]..[1003ffff] 00000000
```

User Stack [7ffff670]..[80000000]

```
[7ffff670] 00000001 7ffff730 00000000 7fffffe1 . . . . 0 . . . . .
[7ffff680] 7fffffb0 7ffff78a 7fffffe4 7fffffd1 . . . . . N . . . . .
[7ffff690] 7fffff00 7ffffedc 7fffffaa 7fffffe7 . . . . . y . . . . .
[7ffff6a0] 7ffffe51 7ffffe44 7ffffde2 7ffffde2 Q . . . . D . . . . / . . . . .
[7ffff6b0] 7ffffdb9 7ffffd9b 7ffffd3b 7ffffd23 . . . . . ; . . . . # . . . .
[7ffff6c0] 7ffffd03 7ffffcf5 7ffffa81 7ffffa43 . . . . . C . . . . .
[7ffff6d0] 7ffffa26 7ffff9e0 7ffff9cd 7ffff9b5 & . . . . .
[7ffff6e0] 7ffff99a 7ffff97c 7ffff953 7ffff935 . . . . . | . . . . S . . . . 5 . . . .
[7ffff6f0] 7ffff8ca 7ffff8b3 7ffff89f 7ffff890 . . . . .
[7ffff700] 7ffff87a 7ffff854 7ffff82f 7ffff814 z . . . . T . . . . / . . . . .
[7ffff710] 7ffff7ea 7ffff7dc 7ffff7c2 7ffff7a0 . . . . .
[7ffff720] 7ffff766 7ffff754 00000000 00000000 f . . . . T . . . . .
[7ffff730] 552f3a43 73726573 4641522f 6f442f49 C : / U s e r s / R A F I / D o
[7ffff740] 6f6c6e77 2f736461 63617270 61636974 w n l o a d s / p r a c t i c a
[7ffff750] 00732e31 646e6977 433d7269 49575c3a l . s . w i n d i r = C : \ W I
[7ffff760] 574f444e 42560053 4d5f584f 495f4953 N D O W S . V B O X _ M S I _ I
[7ffff770] 4154534e 505f4c4c 3d485441 505c3a43 N S T A L L _ P A T H = C : \ P
[7ffff780] 72676f72 46206d61 73656c69 61724f5c r o g r a m _ F i l e s \ O r a
[7ffff790] 5c656c63 74726956 426c6175 005c786f c l e \ V i r t u a l B o x \ .
[7ffff7a0] 584f4256 5657485f 45545249 47495f58 V B O X _ H W V I R T E X _ I G
[7ffff7b0] 45524f4e 4d56535f 5f4e495f 3d455355 N O R E _ S V M _ I N _ U S E =
[7ffff7c0] 53550031 52505245 4c49464f 3a433d45 l . U S E R P R O F I L E = C :
[7ffff7d0] 6573555c 525c7372 00494641 52455355 \ U s e r s \ R A F I . U S E R
[7ffff7e0] 454d414e 4641523d 53550049 4f445245 N A M E = R A F I . U S E R D O
[7ffff7f0] 4e49414d 414f525f 474e494d 464f5250 M A I N _ R O A M I N G P R O F
[7ffff800] 3d454c49 4b534544 2d504f54 44394737 I L E = D E S K T O P - 7 G 9 D
[7ffff810] 00455436 52455355 414d4f44 443d4e49 6 T E . U S E R D O M A I N = D
[7ffff820] 544b5345 372d504f 36443947 54004554 E S K T O P - 7 G 9 D 6 T E . T
[7ffff830] 433d504d 73555c3a 5c737265 49464152 M P = C : \ U s e r s \ R A F I
[7ffff840] 7070415c 61746144 636f4c5c 545c6c61 \ A p p D a t a \ L o c a l \ T
[7ffff850] 00706d65 504d4554 5c3a433d 72657355 e m p . T E M P = C : \ U s e r
[7ffff860] 41525c73 415c4946 61447070 4c5c6174 s \ R A F I \ A p p D a t a \ L
[7ffff870] 6c61636f 6d65545c 79530070 6d657473 o c a l \ T e m p . S y s t e m
[7ffff880] 746f6f52 5c3a433d 444e4957 0053574f R o o t = C : \ W I N D O W S .
[7ffff890] 74737953 72446d65 3d657669 53003a43 S y s t e m D r i v e = C : . S
```

Kernel data segment [90000000]..[90010000]

```
[90000000] 78452020 74706563 206e6f69 636f2000 E x c e p t i o n . o c
[90000010] 72727563 61206465 6920646e 726f6e67 c u r r e d a n d i g n o r
[90000020] 000a6465 495b2020 7265746e 74707572 e d . . [ I n t e r r u p t
[90000030] 2000205d 4c545b20 20005d42 4c545b20 ] . [ T L B ] . [ T L
[90000040] 20005d42 4c545b20 20005d42 64415b20 B ] . [ T L B ] . [ A d
[90000050] 73657264 72652073 20726f72 69206e69 d r e s s e r r o r i n i
[90000060] 2f74736e 61746164 74656620 205d6863 n s t / d a t a f e t c h ]
[90000070] 5b202000 72646441 20737365 6f727265 . [ A d d r e s s e r r o
[90000080] 6e692072 6f747320 205d6572 5b202000 r i n s t o r e ] . [
[90000090] 20646142 74736e69 74637572 206e6f69 B a d i n s t r u c t i o n
[900000a0] 72646461 5d737365 20200020 6461425b a d d r e s s ] . [ B a d
[900000b0] 74616420 64612061 73657264 00205d73 d a t a a d d r e s s ] .
[900000c0] 455b2020 726f7272 206e6920 63737973 [ E r r o r i n s y s c
[900000d0] 5d6c6c61 20200020 6572425b 6f706b61 a l l ] . [ B r e a k p o
[900000e0] 5d746e69 20200020 7365525b 65767265 i n t ] . [ R e s e r v e
[900000f0] 6e692064 75727473 6f697463 00205d6e d i n s t r u c t i o n ] .
[90000100] 5b202000 74697241 74656d68 6f206369 . [ A r i t h m e t i c o
[90000110] 6e726576 5d776f6c 20200020 6172545b v e r f l o w ] . [ T r a
[90000120] 00205d70 5b202000 616f6c46 676e6974 p ] . . [ F l o a t i n g
[90000130] 696f7020 205d746e 20000000 6f435b20 p o i n t ] . . . . [ C o
[90000140] 636f7270 005d3220 20000000 444d5b20 p r o c 2 ] . . . . [ M D
[90000150] 005d594d 575b2020 68637461 2020005d M X ] . [ W a t c h ] .
[90000160] 63614d5b 656e6968 65686320 005d6b63 [ M a c h i n e c h e c k ] .
[90000170] 00000000 5b202000 68636143 00005d65 . . . . . [ C a c h e ] . .
[90000180] 90000024 90000033 9000003b 90000043 $ . . . . 3 . . . . ; . . . . C . . . .
[90000190] 9000004b 90000071 9000008d 900000aa K . . . . q . . . . .
[900001a0] 900000c0 900000d6 900000e6 90000100 . . . . .
[900001b0] 90000101 9000011a 90000124 90000125 . . . . . $ . . . . % . . . .
[900001c0] 90000139 9000013a 9000013b 90000148 9 . . . . : . . . . ; . . . . H . . . .
[900001d0] 90000149 9000014a 9000014b 90000154 I . . . . J . . . . K . . . . T . . . .
[900001e0] 9000015e 90000170 90000171 90000172 ^ . . . . p . . . . q . . . . r . . . .
[900001f0] 90000173 90000174 90000175 9000017f s . . . . t . . . . u . . . . .
[90000200]..[9000ffff] 00000000
```

II. El segmento de Instrucciones (Usuario y Kernel)

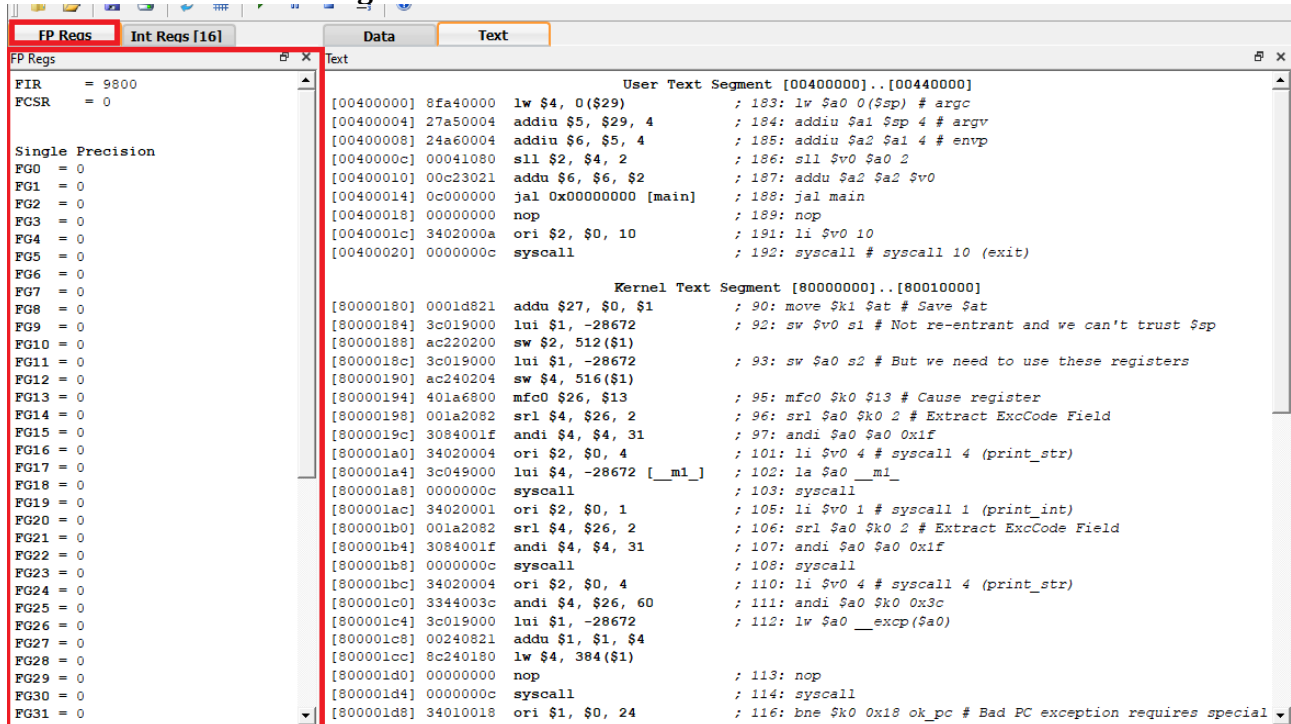
1º Cuadro = USUARIO // 2º Cuadro = KERNEL

Data	Text
Text	
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
[800001d4] 0000000c	syscall ; 114: syscall
[800001d8] 34010018	ori \$1, \$0, 24 ; 116: bne \$k0 0x18 ok_pc # Bad PC exception requires special

III. El contenido de los Registros Enteros

FP Regs	Int Regs [16]	Data	Text
Text			
User Text Segment [00400000]..[00440000]			
PC = 0		[00400000] 8fa40000	lw \$4, 0(\$29) ; 183: lw \$a0 0(\$sp) # argc
EPC = 0		[00400004] 27a50004	addiu \$5, \$29, 4 ; 184: addiu \$a1 \$sp 4 # argv
Cause = 0		[00400008] 24a60004	addiu \$6, \$5, 4 ; 185: addiu \$a2 \$a1 4 # envp
BadVAddr = 0		[0040000c] 00041080	sll \$2, \$4, 2 ; 186: sll \$v0 \$a0 2
Status = 3000fff0		[00400010] 00c23021	addu \$6, \$6, \$2 ; 187: addu \$a2 \$a2 \$v0
HI = 0		[00400014] 0c000000	jal 0x00000000 [main] ; 188: jal main
LO = 0		[00400018] 00000000	nop ; 189: nop
R0 [r0] = 0		[0040001c] 3402000a	ori \$2, \$0, 10 ; 191: li \$v0 10
R1 [at] = 0		[00400020] 0000000c	syscall ; 192: syscall # syscall 10 (exit)
R2 [v0] = 0		Kernel Text Segment [80000000]..[80010000]	
R3 [v1] = 0		[80000180] 0001d821	addu \$27, \$0, \$1 ; 90: move \$k1 \$at # Save \$at
R4 [a0] = 1		[80000184] 3c019000	lui \$1, -28672 ; 92: sw \$v0 \$1 # Not re-entrant and we can't trust \$sp
R5 [a1] = 7ffff674		[80000188] ac220200	sw \$2, 512(\$1)
R6 [a2] = 7ffff67c		[8000018c] 3c019000	lui \$1, -28672 ; 93: sw \$a0 \$2 # But we need to use these registers
R7 [a3] = 0		[80000190] ac240204	sw \$4, 516(\$1)
R8 [t0] = 0		[80000194] 401a6800	mfc0 \$26, \$13 ; 95: mfc0 \$k0 \$13 # Cause register
R9 [t1] = 0		[80000198] 001a2082	srl \$4, \$26, 2 ; 96: srl \$a0 \$k0 2 # Extract ExcCode Field
R10 [t2] = 0		[8000019c] 3084001f	andi \$4, \$4, 31 ; 97: andi \$a0 \$a0 0x1f
R11 [t3] = 0		[800001a0] 34020004	ori \$2, \$0, 4 ; 101: li \$v0 4 # syscall 4 (print_str)
R12 [t4] = 0		[800001a4] 3c049000	lui \$4, -28672 [__m1_] ; 102: la \$a0 __m1_
R13 [t5] = 0		[800001a8] 0000000c	syscall ; 103: syscall
R14 [t6] = 0		[800001ac] 34020001	ori \$2, \$0, 1 ; 105: li \$v0 1 # syscall 1 (print_int)
R15 [t7] = 0		[800001b0] 001a2082	srl \$4, \$26, 2 ; 106: srl \$a0 \$k0 2 # Extract ExcCode Field
R16 [s0] = 0		[800001b4] 3084001f	andi \$4, \$4, 31 ; 107: andi \$a0 \$a0 0x1f
R17 [s1] = 0		[800001b8] 0000000c	syscall ; 108: syscall
R18 [s2] = 0		[800001bc] 34020004	ori \$2, \$0, 4 ; 110: li \$v0 4 # syscall 4 (print_str)
R19 [s3] = 0		[800001c0] 3344003c	andi \$4, \$26, 60 ; 111: andi \$a0 \$k0 0x3c
R20 [s4] = 0		[800001c4] 3c019000	lui \$1, -28672 ; 112: lw \$a0 __excp(\$a0)
R21 [s5] = 0		[800001c8] 00240821	addu \$1, \$1, \$4
R22 [s6] = 0		[800001cc] 8c240180	lw \$4, 384(\$1)
R23 [s7] = 0		[800001d0] 00000000	nop ; 113: nop
R24 [t8] = 0		[800001d4] 0000000c	syscall ; 114: syscall
R25 [t9] = 0		[800001d8] 34010018	ori \$1, \$0, 24 ; 116: bne \$k0 0x18 ok_pc # Bad PC exception requires special
R26 [k0] = 0			
R27 [k1] = 0			

IV. El contenido de los Registros en Punto Flotante.



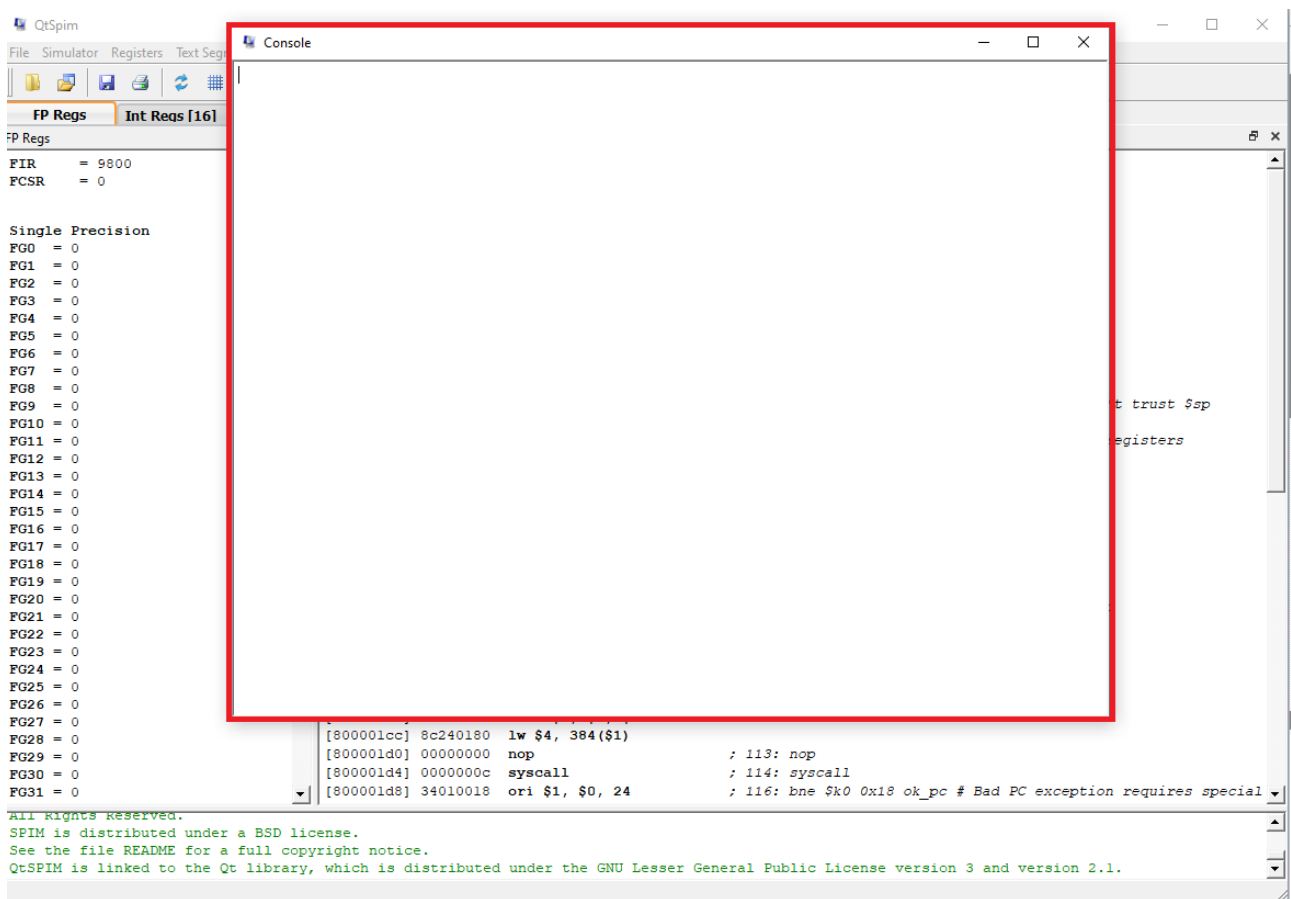
The screenshot displays the QtSpim simulator interface. The 'FP Regs' window is highlighted with a red box, showing the following registers:

- FIR = 9800
- FCSR = 0
- Single Precision registers FG0 through FG31, all set to 0.

The 'Text' window shows assembly code for two segments:

- 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) ; 93: sw \$a0 \$2 # But we need to use these registers
 - [80000190] ac240204 sw \$4, 516(\$1) ; 95: mfc0 \$k0 \$13 # Cause register
 - [80000194] 401ae800 mfc0 \$26, \$13 ; 96: srl \$a0 \$k0 2 # Extract ExcCode Field
 - [80000198] 001a2082 srl \$4, \$26, 2 ; 97: andi \$a0 \$a0 0x1f
 - [8000019c] 3084001f andi \$4, \$4, 31 ; 101: li \$v0 4 # syscall 4 (print_str)
 - [800001a0] 34020004 ori \$2, \$0, 4 ; 102: la \$a0 __ml__
 - [800001a4] 3c049000c lui \$4, -28672 [__ml_] ; 103: syscall
 - [800001a8] 0000000c syscall ; 105: li \$v0 1 # syscall 1 (print_int)
 - [800001ac] 34020001 ori \$2, \$0, 1 ; 106: srl \$a0 \$k0 2 # Extract ExcCode Field
 - [800001b0] 001a2082 srl \$4, \$26, 2 ; 107: andi \$a0 \$a0 0x1f
 - [800001b4] 3084001f andi \$4, \$4, 31 ; 108: syscall
 - [800001b8] 0000000c syscall ; 110: li \$v0 4 # syscall 4 (print_str)
 - [800001bc] 34020004 ori \$2, \$0, 4 ; 111: andi \$a0 \$k0 0x3c
 - [800001c0] 3344003c andi \$4, \$26, 60 ; 112: lw \$a0 __excp(\$a0)
 - [800001c4] 3c019000c lui \$1, -28672 ; 113: nop
 - [800001c8] 00240821 addu \$1, \$1, \$4 ; 114: syscall
 - [800001cc] 8c240180 lw \$4, 384(\$1) ; 116: bne \$k0 0x18 ok_pc # Bad PC exception requires special
 - [800001d0] 00000000 nop ; 113: nop
 - [800001d4] 0000000c syscall ; 114: syscall
 - [800001d8] 34010018 ori \$1, \$0, 24 ; 116: bne \$k0 0x18 ok_pc # Bad PC exception requires special

V. La consola del sistema



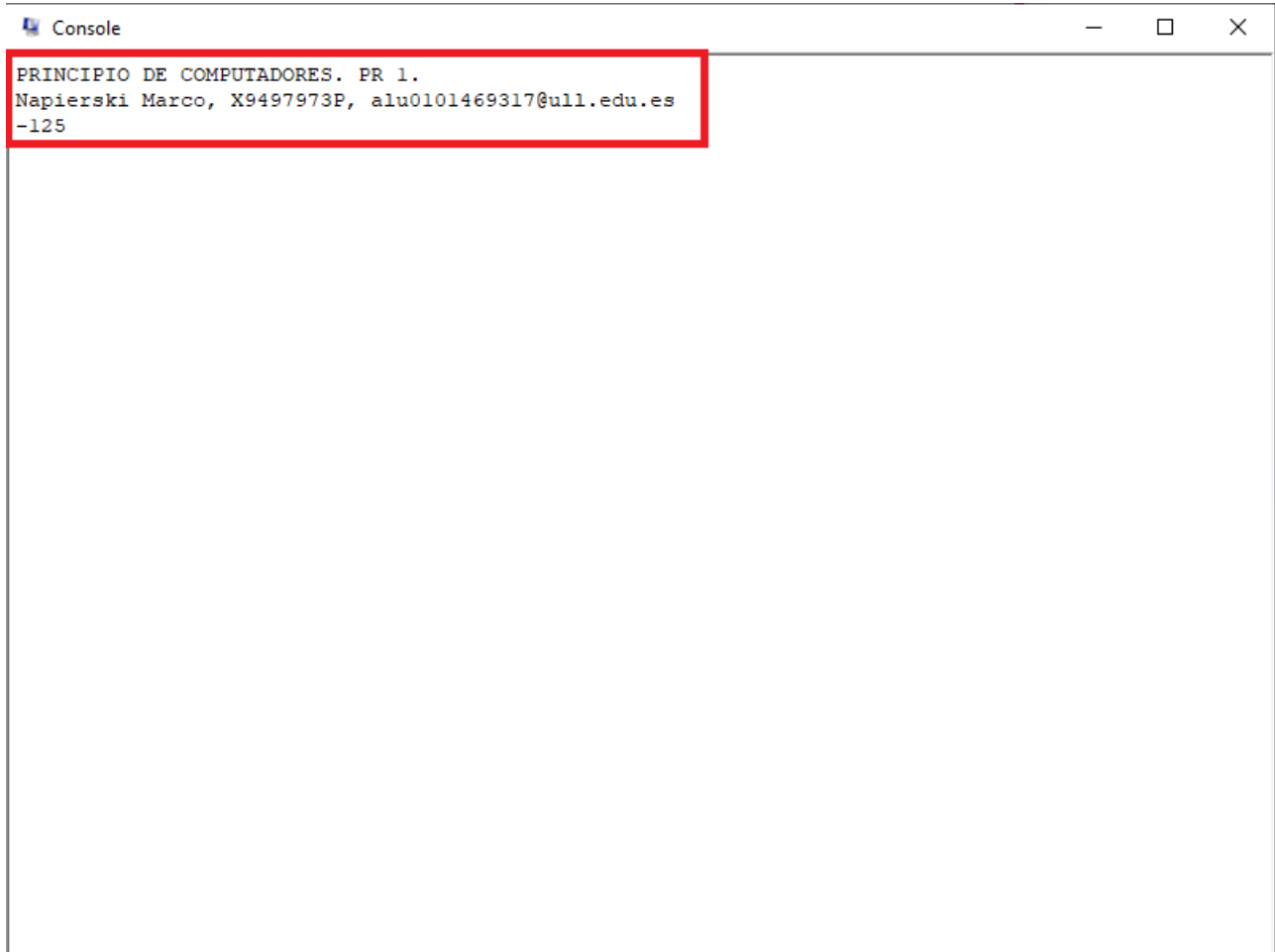
The screenshot displays the QtSpim simulator interface. The 'Console' window is highlighted with a red box, showing assembly code for the Kernel Text Segment:

- [800001cc] 8c240180 lw \$4, 384(\$1)
- [800001d0] 00000000 nop ; 113: nop
- [800001d4] 0000000c syscall ; 114: syscall
- [800001d8] 34010018 ori \$1, \$0, 24 ; 116: bne \$k0 0x18 ok_pc # Bad PC exception requires special

The 'FP Regs' window is also visible in the background, showing the same register values as in the previous screenshot.

b) Edita con un editor de textos plano (vi, vim, gedit, kate, visual studio code o el que prefieras) el fichero practica1.s y sustituye la cadena "apellido1 apellido2 nombre, NIF, alu123456789@ull.edu.es \n" con tu dirección de correo, nombre, apellidos y NIF (NIE o pasaporte), y graba el fichero. A continuación, carga el programa en QtSpim y ejecuta el programa de una sola vez. Saca un pantallazo de la consola y marca mediante un cuadro rojo la impresión de tus datos.

CONSOLA

A screenshot of a QtSpim console window. The window has a title bar with the text "Console" and standard window controls (minimize, maximize, close). The console area contains the following text: "PRINCIPIO DE COMPUTADORES. PR 1.", "Napierski Marco, X9497973P, alu0101469317@ull.edu.es", and "-125". A red rectangular box is drawn around the first three lines of text, highlighting the program's output.

```
PRINCIPIO DE COMPUTADORES. PR 1.  
Napierski Marco, X9497973P, alu0101469317@ull.edu.es  
-125
```

c) Explora el segmento de datos. (NOTA: Al sacar los pantallazos debes incluir las direcciones y los valores del User Data Segment completo, ya que los valores y las direcciones dependerán del nombre de cada alumno y serán necesarios para poder realizar la corrección).

Comprueba que el segmento de datos está representado en Hexadecimal (en el menú Data Segment marca el checkbox correspondiente a Hex). Con las indicaciones que te de tu profesor en la práctica responde a las siguientes preguntas:

I. ¿Qué dirección de memoria (expresa la dirección en hexadecimal) ocupa el primer carácter de tu nombre? (p.ej: en "Martin Galan Carlos, 43777444Z, alu123456789@ull.edu.es\n" el carácter en cuestión es "C")?

[1001002C]

II. ¿Qué carácter es y qué representación tiene en hexadecimal? Saca un pantallazo de User data Segment y marca con un cuadro rojo el byte correspondiente a ese carácter.

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff] 00000000
[10010000] 4e495250 49504943 4544204f 4d4f4320 P R I N C I P I O D E C O M
[10010010] 41545550 45524f44 50202e53 2e312052 P U T A D O R E S . P R l .
[10010020] 614e000a 72656970 20696b73 6372614d . . N a p i e r s k i M a r c
[10010030] 58202c6f 37393439 50333739 6c61202c o , X 9 4 9 7 9 7 3 P , a l
[10010040] 30313075 39363431 40373133 2e6c6c75 u 0 1 0 1 4 6 9 3 1 7 @ u l l .
[10010050] 2e756465 000a7365 490a3d71 4180f159 e d u . e s . . q = . I Y . . A
[10010060] 00000015 0000002c 3bafcl0c 4089alcb . . . . , . . . . . ; . . . @
[10010070]..[1003ffff] 00000000
```

III. Recuerda que estás en hexadecimal. Busca en el segmento de datos de qtspim el número que se encuentra en la dirección etiquetada como num3. Saca un pantallazo y marca con un recuadro en rojo la palabra correspondiente.

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff] 00000000
[10010000] 4e495250 49504943 4544204f 4d4f4320 P R I N C I P I O D E C O M
[10010010] 41545550 45524f44 50202e53 2e312052 P U T A D O R E S . P R l .
[10010020] 614e000a 72656970 20696b73 6372614d . . N a p i e r s k i M a r c
[10010030] 58202c6f 37393439 50333739 6c61202c o , X 9 4 9 7 9 7 3 P , a l
[10010040] 30313075 39363431 40373133 2e6c6c75 u 0 1 0 1 4 6 9 3 1 7 @ u l l .
[10010050] 2e756465 000a7365 490a3d71 4180f159 e d u . e s . . q = . I Y . . A
[10010060] 00000015 0000002c 3bafcl0c 4089alcb . . . . , . . . . . ; . . . @
[10010070]..[1003ffff] 00000000
```

IV. Convierte el número 4.301 a formato IEE-754 para 32 bits (usa los apuntes del profesor o utiliza una calculadora online). Busca ahora este número en el segmento de datos, saca un pantallazo y márcalo con un cuadro en rojo.

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff] 00000000
[10010000] 4e495250 49504943 4544204f 4d4f4320 P R I N C I P I O D E C O M
[10010010] 41545550 45524f44 50202e53 2e312052 P U T A D O R E S . P R l .
[10010020] 614e000a 72656970 20696b73 6372614d . . N a p i e r s k i M a r c
[10010030] 58202c6f 37393439 50333739 6c61202c o , X 9 4 9 7 9 7 3 P , a l
[10010040] 30313075 39363431 40373133 2e6c6c75 u 0 1 0 1 4 6 9 3 1 7 @ u l l .
[10010050] 2e756465 000a7365 490a3d71 4180f159 e d u . e s . . q = . I Y . . A
[10010060] 00000015 0000002c 3bafcl0c 4089alcb . . . . , . . . . . ; . . . @
[10010070]..[1003ffff] 00000000
```

V. ¿En qué dirección empieza el número 4.301? expresa la dirección en hexadecimal.

[1001006C]

VI. Convierte el número 35531561.13 a formato IEE-754 para 64 bits (usa los apuntes del profesor o utiliza una calculadora online). Busca ahora este número en el segmento de datos, saca un pantallazo y márcalo con un cuadro en rojo.

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff] 00000000
[10010000] 4e495250 49504943 4544204f 4d4f4320 P R I N C I P I O   D E   C O M
[10010010] 41545550 45524f44 50202e53 2e312052 P U T A D O R E S .   P R   l .
[10010020] 614e000a 72656970 20696b73 6372614d . . N a p i e r s k i   M a r c
[10010030] 58202c6f 37393439 50333739 6c61202c o ,   X 9 4 9 7 9 7 3 P ,   a l
[10010040] 30313075 39363431 40373133 2e6c6c75 u 0 1 0 1 4 6 9 3 1 7 @ u l l .
[10010050] 2e756465 000a7365 490a3d71 4180f159 e d u . e s . . q = . I Y . . A
[10010060] 00000015 0000002c 3baafc10c 4089a1cb . . . . , . . . . . ; . . . @
[10010070]..[1003ffff] 00000000
```

VII. ¿En qué dirección empieza el número 35531561.13? expresa la dirección en hexadecimal. [10010058]

d) Reinicia la máquina y vuelve a cargar el programa en el QtSpim. Visualiza el banco de registros enteros y flotantes en hexadecimal. Menú Registers opción Hex). Recuerda, que las instrucciones de tu programa pueden ser convertidas en una o más instrucciones en QtSpim, por lo que tendrás que buscar la instrucción original de tu programa en los comentarios de la parte derecha.

I. Ejecuta paso a paso el programa hasta que hayas encontrado la instrucción add \$t2,\$t0,\$t1 Una vez se haya ejecutado saca un pantallazo del banco de registros enteros y pon un cuadro rojo sobre el registro \$t2. ¿Qué valor contiene? ¿sabrías expresarlo en decimal?

The screenshot shows the QtSpim MIPS simulator. The 'Registers' window is open, displaying the integer register bank. Register \$t2 is highlighted with a red box and contains the value 41. The 'Text' window shows the assembly code, with the instruction 'add \$t2, \$t0, \$t1' highlighted, corresponding to the value in \$t2.

Tiene valor 41 en hexadecimal

Sí, es 65 en decimal. (4*16 + 1*1 = 65)

II. Cuando hayas terminado de ejecutar esta instrucción, modifica a mano el valor del registro \$t3 (pulsas con el botón derecho del ratón sobre el registro correspondiente en el banco de registro y selecciona “Change Register Contents”, allí puedes seleccionar el formato y el valor). Deberás introducir un valor 1200 en formato decimal. Una vez lo hayas hecho saca un pantallazo y marca con un cuadro en rojo el registro correspondiente.

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Reqs Int Regs [16] Data Text

nt Regs [16]

PC = 400058
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000fff10

HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 10010000
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010022
R5 [a1] = 7ffff674
R6 [a2] = 7ffff67c
R7 [a3] = 0
R8 [t0] = 15
R9 [t1] = 2c
R10 [t2] = 41
R11 [t3] = 4b0
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

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.

4b0 = 1200 (4*16*16 + 11 * 16 = 1200)

III. A continuación sigue ejecutando paso a paso hasta terminar de ejecutar la instrucción add \$t4,\$t2,\$t3. ¿Qué valor tiene el registro \$t4 en hexadecimal? ¿y en decimal?

HEXADECIMAL: 4f1

DECIMAL : 1265

FP Reqs Int Regs [16] Data Text

Int Regs [16]

PC = 40005c
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000fff10

HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 10010000
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010022
R5 [a1] = 7ffff674
R6 [a2] = 7ffff67c
R7 [a3] = 0
R8 [t0] = 15
R9 [t1] = 2c
R10 [t2] = 41
R11 [t3] = 4b0
R12 [t4] = 4f1
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

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.

IV. A continuación establece un punto de ruptura “breakpoint” sobre la instrucción `move $a0,$t3` (sobre la instrucción correspondiente, pulsa en el botón derecho del ratón y selecciona “Set Breakpoint”. Después ejecuta todo el código (no paso a paso) y observarás que la ejecución se para en esta instrucción saltándose el bucle que hemos puesto. En este punto. ¿Qué valor tiene `$t3` (expresado en hexadecimal y también en decimal)? ¿y qué valor tiene `$t6`?

`$t3`: HEX = 433 // DEC = 1075

`$t6`: HEX = 0 // DEC = 0

Hexadecimal:

FP Regs	Int Regs [16]	Data	Text
Int Regs [16]			
PC = 400080		[00400018] 00000000	nop ; 189: nop
EPC = 400070		[0040001c] 3402000a	ori \$2, \$0, 10 ; 191: li \$v0 10
Cause = 24		[00400020] 0000000c	syscall ; 192: syscall # syscall 10 (exit)
BadVAddr = 0		[00400024] 3c011001	lui \$1, 4097 [titulo] ; 6: la \$a0,titulo
Status = 3000fff10		[00400028] 34240000	ori \$4, \$1, 0 [titulo]
		[0040002c] 34020004	ori \$2, \$0, 4 ; 7: li \$v0,4
HI = 0		[00400030] 0000000c	syscall ; 8: syscall
LO = 0		[00400034] 3c011001	lui \$1, 4097 [alumno]
		[00400038] 34240022	ori \$4, \$1, 34 [alumno]
		[0040003c] 34020004	ori \$2, \$0, 4 ; 12: li \$v0,4
R0 [r0] = 0		[00400040] 0000000c	syscall ; 13: syscall
R1 [at] = 10010000		[00400044] 3c011001	lui \$1, 4097 [num1]
R2 [v0] = a		[00400048] 8c280060	lw \$8, 96(\$1) [num1]
R3 [v1] = 0		[0040004c] 8c280060	lw \$8, 96(\$1) [num1]
R4 [a0] = 433		[00400050] 3c011001	lui \$1, 4097 [num2]
R5 [a1] = 7ffff674		[00400054] 8c290064	lw \$9, 100(\$1) [num2]
R6 [a2] = 7ffff67c		[00400058] 01095020	add \$10, \$8, \$9 ; 17: add \$t2,\$t0,\$t1 # realiza la siguiente operacion \$t2 = \$t0 + \$t1
R7 [a3] = 0		[0040005c] 014b6020	add \$12, \$10, \$11 ; 19: add \$t4, \$t2, \$t3 # realiza la siguiente operacion \$t4 = \$t2 + \$t3
R8 [t0] = 15		[00400060] 340e007d	ori \$14, \$0, 125 ; 23: li \$t6,125
R9 [t1] = 2c		[00400064] 11c00004	beq \$14, \$0, 16 [fin_buclewhile-0x00400060]
R10 [t2] = 41		[00400068] 216bffff	addi \$11, \$11, -1 ; 25: addi \$t3,-1
R11 [t3] = 433		[0040006c] 21ceffff	addi \$14, \$14, -1 ; 26: addi \$t6,-1
R12 [t4] = 4f1		[00400070] 0401ffff	bgez \$0 -12 [buclewhile-0x0040006c]
R13 [t5] = 0		[x0040007] x000b202	addu \$4, \$0, \$11 ; 32: move \$a0,\$t3
R14 [t6] = 0		[00400074] 34020001	ori \$2, \$0, 1 ; 33: li \$v0,1
R15 [t7] = 0		[00400078] 0000000c	syscall ; 34: syscall
R16 [s0] = 0		[0040007c] 3402000a	ori \$2, \$0, 10 ; 37: li \$v0,10
R17 [s1] = 0		[00400080] 0000000c	syscall ; 38: syscall
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			

Decimal:

FP Regs	Int Regs [10]	Data	Text
Int Regs [10]			
PC = 4194432		[00400018] 00000000	nop ; 189: nop
EPC = 4194416		[0040001c] 3402000a	ori \$2, \$0, 10 ; 191: li \$v0 10
Cause = 36		[00400020] 0000000c	syscall ; 192: syscall # syscall 10 (exit)
BadVAddr = 0		[00400024] 3c011001	lui \$1, 4097 [titulo]
Status = 805371664		[00400028] 34240000	ori \$4, \$1, 0 [titulo]
		[0040002c] 34020004	ori \$2, \$0, 4 ; 7: li \$v0,4
HI = 0		[00400030] 0000000c	syscall ; 8: syscall
LO = 0		[00400034] 3c011001	lui \$1, 4097 [alumno]
		[00400038] 34240022	ori \$4, \$1, 34 [alumno]
		[0040003c] 34020004	ori \$2, \$0, 4 ; 12: li \$v0,4
R0 [r0] = 0		[00400040] 0000000c	syscall ; 13: syscall
R1 [at] = 268500992		[00400044] 3c011001	lui \$1, 4097 [num1]
R2 [v0] = 10		[00400048] 8c280060	lw \$8, 96(\$1) [num1]
R3 [v1] = 0		[0040004c] 8c280060	lw \$8, 96(\$1) [num1]
R4 [a0] = 1075		[00400050] 3c011001	lui \$1, 4097 [num2]
R5 [a1] = 2147481204		[00400054] 8c290064	lw \$9, 100(\$1) [num2]
R6 [a2] = 2147481212		[00400058] 01095020	add \$10, \$8, \$9 ; 17: add \$t2,\$t0,\$t1 # realiza la siguiente operacion \$t2 = \$t0 + \$t1
R7 [a3] = 0		[0040005c] 014b6020	add \$12, \$10, \$11 ; 19: add \$t4, \$t2, \$t3 # realiza la siguiente operacion \$t4 = \$t2 + \$t3
R8 [t0] = 21		[00400060] 340e007d	ori \$14, \$0, 125 ; 23: li \$t6,125
R9 [t1] = 44		[00400064] 11c00004	beq \$14, \$0, 16 [fin_buclewhile-0x00400060]
R10 [t2] = 65		[00400068] 216bffff	addi \$11, \$11, -1 ; 25: addi \$t3,-1
R11 [t3] = 1075		[0040006c] 21ceffff	addi \$14, \$14, -1 ; 26: addi \$t6,-1
R12 [t4] = 1265		[00400070] 0401ffff	bgez \$0 -12 [buclewhile-0x0040006c]
R13 [t5] = 0		[x0040007] x000b202	addu \$4, \$0, \$11 ; 32: move \$a0,\$t3
R14 [t6] = 0		[00400074] 34020001	ori \$2, \$0, 1 ; 33: li \$v0,1
R15 [t7] = 0		[00400078] 0000000c	syscall ; 34: syscall
R16 [s0] = 0		[0040007c] 3402000a	ori \$2, \$0, 10 ; 37: li \$v0,10
R17 [s1] = 0		[00400080] 0000000c	syscall ; 38: syscall
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			