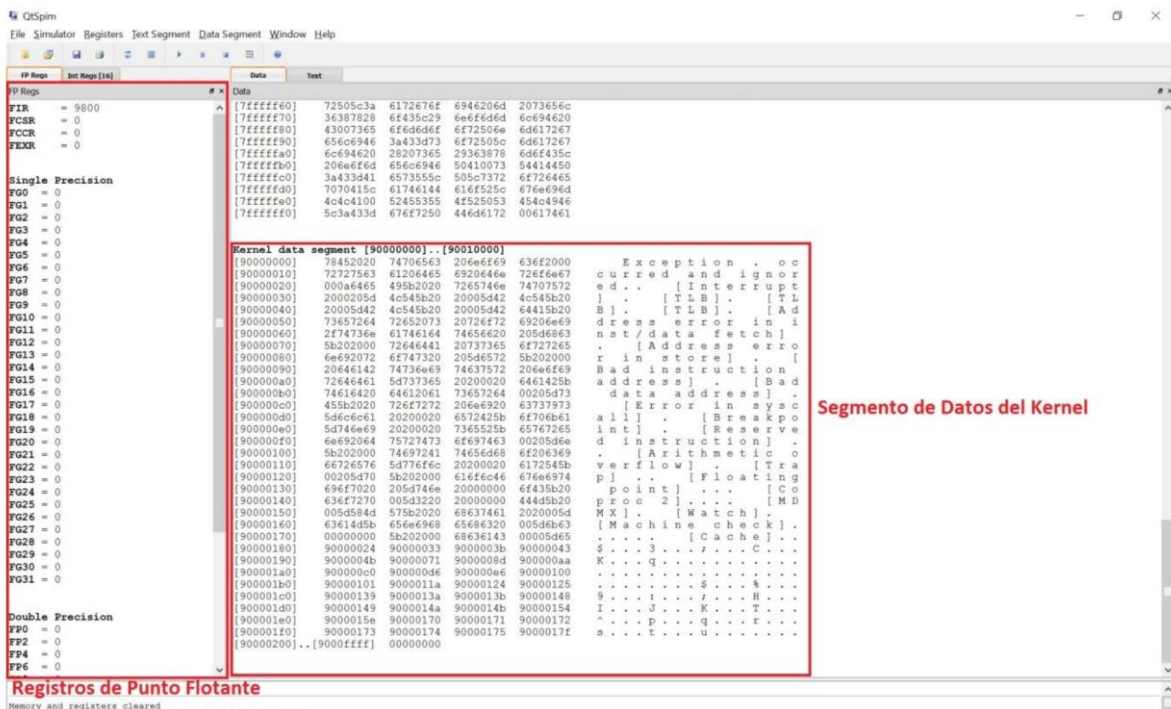
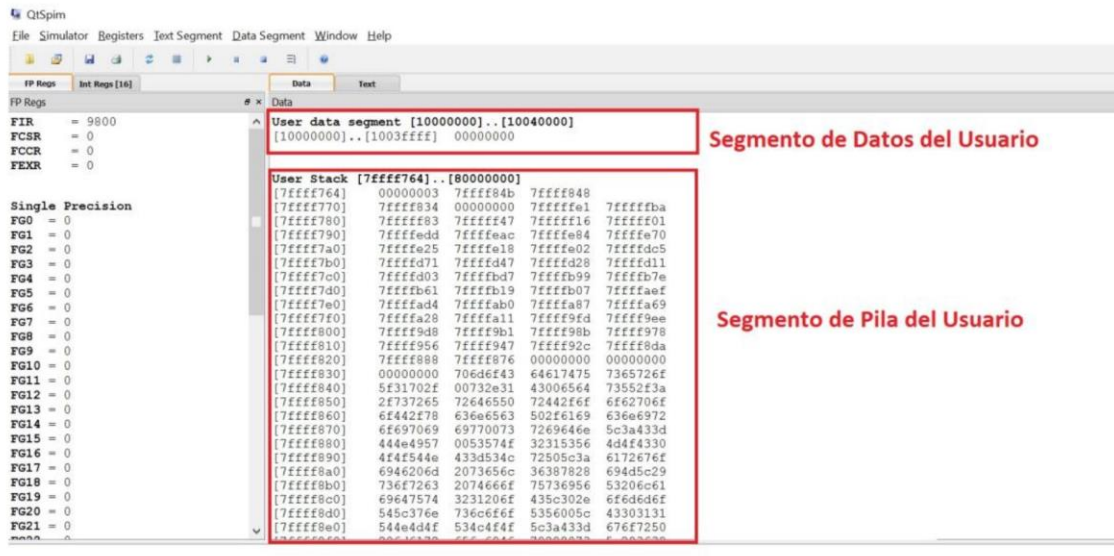


EJEMPLO DE RESOLUCIÓN DE INFORME.

- Identificar en el emulador los siguientes elementos
- I. El segmento de Datos (Usuario, Kernel y Pila)
- II. El segmento de Instrucciones (Usuario y Kernel)
- III. El contenido de los Registros Enteros
- IV. El contenido de los Registros en Punto Flotante.
- V. La consola del sistema



QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Text

Int Regs [16]

PC = 0
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000fff0
HI = 0
LO = 0
R0 [r0] = 0
R1 [a1] = 0
R2 [v0] = 0
R3 [v1] = 0
R4 [a0] = 3
R5 [a1] = 7ffff768
R6 [a2] = 7ffff778
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 [s8] = 0
R25 [s9] = 0
R26 [s10] = 0
R27 [s11] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff764
R30 [s8] = 0
R31 [ra] = 0

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 $a0 # Save $a0
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 ; 95: mfc0 $k0 $13 # Cause register
80000190 ac240204 sw $4, 516($1) ; 96: srl $a0 $k0 2 # Extract ExcCode Field
80000194 401a6800 mfc0 $26, $13 ; 97: andi $a0 $a0 0x1f
80000198 001a2082 srl $4, $26, 2 ; 101: li $v0 4 # syscall 4 (print_str)
8000019c 3084001f andi $4, $4, 31 ; 102: la $a0 __m1_
800001a0 34020004 ori $2, $0, 4 ; 103: syscall
800001a4 3c049000 lui $4, -28672 [__m1_] ; 105: li $v0 1 # syscall 1 (print_int)
800001a8 0000000c syscall ; 106: srl $a0 $k0 2 # Extract ExcCode Field
800001ac 34020001 ori $2, $0, 1 ; 107: andi $a0 $a0 0x1f
800001b0 001a2082 srl $4, $26, 2 ; 108: syscall
800001b4 3084001f andi $4, $4, 31 ; 110: li $v0 4 # syscall 4 (print_str)
800001b8 0000000c syscall ; 111: andi $a0 $k0 0x3c
800001bc 34020004 ori $2, $0, 4 ; 112: lw $a0 __excp($a0)
800001c0 3344003c andi $4, $26, 60 ; 113: nop
800001c4 3c019000 lui $1, -28672 ; 114: syscall
800001c8 00240821 addu $1, $1, $4 ; 115: bne $k0 0x18 ok_pc # Bad PC exception requires special checks
800001cc 8c240180 lw $4, 384($1) ; 117: nop
800001d0 00000000 nop ; 119: mfc0 $a0 $14 # EPC
800001d4 0000000c syscall ; 120: andi $a0 $a0 0x3 # Is EPC word-aligned?
800001d8 34010018 ori $1, $0, 24 ; 122: nop
800001dc 143a0008 bne $1, $26, 32 [ok_pc-0x800001dc] ; 124: li $v0 10 # Exit on really bad PC
800001e0 00000000 nop ; 125: syscall
800001e4 40047000 mfc0 $4, $14 ; 128: li $v0 4 # syscall 4 (print_str)
800001e8 30840003 andi $4, $4, 3 ; 129: la $a0 __m2_
800001ec 10040004 bne $0, $4, 16 [ok_pc-0x800001ec] ; 
800001f0 00000000 nop ; 
800001f4 3402000a ori $2, $0, 10 ; 
800001f8 0000000c syscall ; 
800001fc 34020004 ori $2, $0, 4 ; 
80000200 3c019000 lui $1, -28672 [__m2_] ; 
80000204 3424000d ori $4, $1, 13 [__m2_] ;
```

Registros Enteros

Memory and registers cleared

Segmento de Instrucciones del Usuario

Segmento de Instrucciones del Kernel



b) Edita con un editor de textos plano (vi, vim, gedit, kate o el que prefieras) el fichero practica1.s y sustituye la cadena “nombre apellido1 apellido2\n” con tu nombre y tus apellidos y graba el fichero. A continuación carga el programa en QtSpim y ejecuta el programa de una sola vez. Sacar un pantallazo de la consola y marca mediante un cuadro rojo la impresión de tu nombre.



c)

I. ¿Qué dirección de memoria (expresa la dirección en hexadecimal) ocupa el último carácter de tu nombre (p.ej: en “Simpson, Homer” el carácter en cuestión es “n”)?

En este caso estoy buscando la dirección del carácter ‘o’ que es [1001004E]
Que corresponde con el carácter hexadecimal 0x6F que en la tabla ascii es ‘o’

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

Data					
User data segment [10000000]..[10040000]					
[10000000]..[1000ffff]	00000000				
[10010000]	00000011	0000002d	1745f44e	3fcb22d1 - . . . N . E . . " . ?
[10010010]	c1d70a3d	418b6ec5	63617250	61636974	= n . A P r a c t i c a
[10010020]	64203120	72702065	69636e69	206f6970	l d e p r i n c i p i o
[10010030]	63206564	75706d6f	6f646174	0a736572	d e c o m p u t a d o r e s .
[10010040]	6c6f5400	206f6465	676c6544	2c6f6461	. T o l e d o D e l g a d o ,
[10010050]	64655020	41206f72	6e6f746e	000a6f69	P e d r o A n t o n i o . .
[10010060]..[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.

Data					
User data segment [10000000]..[10040000]					
[10000000]..[1000ffff]	00000000				
[10010000]	00000011	0000002d	1745f44e	3fcb22d1 - . . . N . E . . " . ?
[10010010]	c1d70a3d	418b6ec5	63617250	61636974	= n . A P r a c t i c a
[10010020]	64203120	72702065	69636e69	206f6970	l d e p r i n c i p i o
[10010030]	63206564	75706d6f	6f646174	0a736572	d e c o m p u t a d o r e s .
[10010040]	6c6f5400	206f6465	676c6544	2c6f6461	. T o l e d o D e l g a d o ,
[10010050]	64655020	41206f72	6e6f746e	000a6f69	P e d r o A n t o n i o . .
[10010060]..[1003ffff]	00000000				

IV. Convierte el número 1.578 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.

1.578 en IEEE754 32 bits es 0x3FCB22D1

Data					
User data segment [10000000]..[10040000]					
[10000000]..[1000ffff]	00000000				
[10010000]	00000011	0000002d	1745f44e	3fcb22d1 - . . . N . E . . " . ?
[10010010]	c1d70a3d	418b6ec5	63617250	61636974	= n . A P r a c t i c a
[10010020]	64203120	72702065	69636e69	206f6970	l d e p r i n c i p i o
[10010030]	63206564	75706d6f	6f646174	0a736572	d e c o m p u t a d o r e s .
[10010040]	6c6f5400	206f6465	676c6544	2c6f6461	. T o l e d o D e l g a d o ,
[10010050]	64655020	41206f72	6e6f746e	000a6f69	P e d r o A n t o n i o . .
[10010060]..[1003ffff]	00000000				

V. ¿En qué dirección empieza el número 1.578? expresa la dirección en hexadecimal. [1001000C] (empieza en la dirección 1001000C y su último byte está en la 1001000F)

VI. Convierte el número 57530552.23 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.

57530552.23 en IEEE754 64bits es 0x418B6EC5C1D70A3D

Data					
User data segment [10000000]..[10040000]					
[10000000]..[1000ffff]	00000000				
[10010000]	00000011	0000002d	1745f44e	3fcb22d1 - . . . N . E . . " . ?
[10010010]	c1d70a3d	418b6ec5	63617250	61636974	= n . A P r a c t i c a
[10010020]	64203120	72702065	69636e69	206f6970	l d e p r i n c i p i o
[10010030]	63206564	75706d6f	6f646174	0a736572	d e c o m p u t a d o r e s .
[10010040]	6c6f5400	206f6465	676c6544	2c6f6461	. T o l e d o D e l g a d o ,
[10010050]	64655020	41206f72	6e6f746e	000a6f69	P e d r o A n t o n i o . .
[10010060]..[1003ffff]	00000000				

VII. ¿En qué dirección empieza el número 53125322.45? expresa la dirección en hexadecimal.

[10010010]] (empieza en la dirección 10010010 y su último byte está en la 10010017)

d) Ejecución del programa:

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?

Contiene el valor `0x3E` que es $16 \times 3 + 14 = 62$

Int Regs [16]	#	Text
PC = 400058	[00400004]	addiu \$5, \$29, 4 ; 184: addiu \$a1 \$sp 4 # argv
EPC = 0	[00400008]	addiu \$6, \$5, 4 ; 185: addiu \$a2 \$a1 4 # envp
Cause = 0	[0040000c]	sll \$2, \$4, 2 ; 186: sll \$v0 \$a0 2
BadVAddr = 0	[00400010]	addu \$6, \$6, \$2 ; 187: addu \$a2 \$a2 \$v0
Status = 3000fff10	[00400014]	jal 0x00400024 [main] ; 188: jal main
	[00400018]	nop ; 189: nop
HI = 0	[0040001c]	ori \$2, \$0, 10 ; 191: li \$v0 10
LO = 0	[00400020]	syscall ; 192: syscall # syscall 10 (exit)
	[00400024]	lui \$1, 4097 [titulo] ; 12: la \$a0,titulo
	[00400028]	ori \$4, \$1, 24 [titulo] ; 13: li \$v0,4
R0 [r0] = 0	[0040002c]	ori \$2, \$0, 4 ; 14: syscall
R1 [a0] = 10010000	[00400030]	syscall ; 17: la \$a0,nombre
R2 [v0] = 4	[00400034]	lui \$1, 4097 [nombre] ; 18: li \$v0,4
R3 [v1] = 0	[00400038]	ori \$4, \$1, 65 [nombre] ; 19: syscall
R4 [a0] = 10010041	[0040003c]	ori \$2, \$0, 4 ; 21: lw \$t0,num1 # carga en el registro \$t0 el valor etiquetado como num1
R5 [a1] = 7ffff46c	[00400040]	syscall ; 22: lw \$t1,num2 # carga en el registro \$t1 el valor etiquetado como num2
R6 [a2] = 7ffff474	[00400044]	lui \$1, 4097 [num1] ; 23: add \$t2,\$t0,\$t1 # realiza la siguiente operacion \$t2 = \$t0 + \$t1
R7 [a3] = 0	[00400048]	lw \$8, 0(\$1) [num1] ; 25: add \$t4, \$t2, \$t3 # realiza la siguiente operacion \$t4 = \$t2 + \$t3
R8 [t0] = 11	[0040004c]	lui \$1, 4097 [num2] ; 29: li \$t6,555
R9 [t1] = 2d	[00400050]	lw \$9, 4(\$1) [num2] ; 31: addi \$t3,-1
R10 [t2] = 3e	[00400054]	add \$10, \$8, \$9 ; 32: addi \$t6,-1
R11 [t3] = 3e	[00400058]	add \$12, \$10, \$11
R12 [t4] = 3e	[0040005c]	ori \$14, \$0, 555
R13 [t5] = 0	[00400060]	beq \$14, \$0, 16 [fin_buclewhile-0x00400060]
R14 [t6] = 0	[00400064]	addi \$11, \$11, -1
R15 [t7] = 0	[00400068]	addi \$14, \$14, -1
R16 [s0] = 0		

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 seleccionas “Change Register Contents”, allí puedes seleccionar el formato y el valor). Deberás introducir un valor 1010 en formato decimal. Una vez lo hayas hecho saca un pantallazo y marca con un cuadro en rojo el registro correspondiente.

Se queda con el valor `0x3f2` que es 1010 en decimal

FP Regs	Int Regs [16]	Data	Text
PC = 400058	[00400004]	27a50004	addiu \$5, \$29, 4 ; 184: addiu \$a1 \$sp 4 # argv
EPC = 0	[00400008]	24a60004	addiu \$6, \$5, 4 ; 185: addiu \$a2 \$a1 4 # envp
Cause = 0	[0040000c]	00041080	sll \$2, \$4, 2 ; 186: sll \$v0 \$a0 2
BadVAddr = 0	[00400010]	00c23021	addu \$6, \$6, \$2 ; 187: addu \$a2 \$a2 \$v0
Status = 3000fff10	[00400014]	0c100009	jal 0x00400024 [main] ; 188: jal main
	[00400018]	00000000	nop ; 189: nop
HI = 0	[0040001c]	3402000a	ori \$2, \$0, 10 ; 191: li \$v0 10
LO = 0	[00400020]	0000000c	syscall ; 192: syscall # syscall 10 (exit)
	[00400024]	3c011001	lui \$1, 4097 [titulo] ; 12: la \$a0,titulo
	[00400028]	34240018	ori \$4, \$1, 24 [titulo] ; 13: li \$v0,4
R0 [r0] = 0	[0040002c]	34020004	ori \$2, \$0, 4 ; 14: syscall
R1 [a0] = 10010000	[00400030]	0000000c	syscall ; 17: la \$a0,nombre
R2 [v0] = 4	[00400034]	3c011001	lui \$1, 4097 [nombre] ; 18: li \$v0,4
R3 [v1] = 0	[00400038]	34240041	ori \$4, \$1, 65 [nombre] ; 19: syscall
R4 [a0] = 10010041	[0040003c]	34020004	ori \$2, \$0, 4 ; 21: lw \$t0,num1 # carga en el registro \$t0 el valor etiquetado como num1
R5 [a1] = 7ffff46c	[00400040]	0000000c	syscall ; 22: lw \$t1,num2 # carga en el registro \$t1 el valor etiquetado como num2
R6 [a2] = 7ffff474	[00400044]	3c011001	lui \$1, 4097 [num1] ; 23: add \$t2,\$t0,\$t1 # realiza la siguiente operacion \$t2 = \$t0 + \$t1
R7 [a3] = 0	[00400048]	8c280000	lw \$8, 0(\$1) [num1] ; 25: add \$t4, \$t2, \$t3 # realiza la siguiente operacion \$t4 = \$t2 + \$t3
R8 [t0] = 11	[0040004c]	3c011001	lui \$1, 4097 [num2] ; 29: li \$t6,555
R9 [t1] = 2d	[00400050]	8c290004	lw \$9, 4(\$1) [num2] ; 31: addi \$t3,-1
R10 [t2] = 3e	[00400054]	01095020	add \$10, \$8, \$9 ; 32: addi \$t6,-1
R11 [t3] = 3f2	[00400058]	014b6020	add \$12, \$10, \$11
R12 [t4] = 3e	[0040005c]	340e022b	ori \$14, \$0, 555
R13 [t5] = 0	[00400060]	11c00004	beq \$14, \$0, 16 [fin_buclewhile-0x00400060]
R14 [t6] = 0	[00400064]	216bffff	addi \$11, \$11, -1
R15 [t7] = 0	[00400068]	21ceffff	addi \$14, \$14, -1
R16 [s0] = 0			

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?

En hexadecimal tiene el valor `0x430`

En decimal tiene el valor 1072

(se puede comprobar cambiando el modo de visualización en el menú Registers)

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 en hexadecimal tiene un 0x1c7 y en decimal 455

\$t6 tiene un cero (que es igual en los dos formatos)

FP Regs	Int Regs [10]	Data	Text
Int Regs [10]	# x		
BadVAddr = 0			
Status = 805371664			
HI = 0			
LO = 0			
R0 [r0] = 0			
R1 [at] = 268500992			
R2 [v0] = 4			
R3 [v1] = 0			
R4 [a0] = 455			
R5 [a1] = 2147480684			
R6 [a2] = 2147480692			
R7 [a3] = 0			
R8 [t0] = 17			
R9 [t1] = 45			
R10 [t2] = 62			
R11 [t3] = 455			
R12 [t4] = 1072			
R13 [t5] = 0			
R14 [t6] = 0			
R15 [t7] = 0			
R16 [s0] = 0			
R17 [s1] = 0			
R18 [s2] = 0			
R19 [s3] = 0			
			Kernel Text Segment [80000000]..[80010000]
FP Regs	Int Regs [16]	Data	Text
Int Regs [16]	# x		
BadVAddr = 0			
Status = 3000ff10			
HI = 0			
LO = 0			
R0 [r0] = 0			
R1 [at] = 10010000			
R2 [v0] = 4			
R3 [v1] = 0			
R4 [a0] = 1c7			
R5 [a1] = 7ffff46c			
R6 [a2] = 7ffff474			
R7 [a3] = 0			
R8 [t0] = 11			
R9 [t1] = 2d			
R10 [t2] = 3e			
R11 [t3] = 1c7			
R12 [t4] = 439			
R13 [t5] = 0			
R14 [t6] = 0			
R15 [t7] = 0			
R16 [s0] = 0			
R17 [s1] = 0			
R18 [s2] = 0			
R19 [s3] = 0			
			Kernel Text Segment [80000000]..[80010000]