Universidad de La Laguna

Principios de Computadores

Utilización del emulador QTSpim y representación de la información

Aday Padilla Amaya

March 1, 2019

Índice

1.- Introducción 2

2.- Ejercicios de la práctica

a) Identificación en el emulador

I.- El segmento de Datos 2

II.- El segmento de Instrucciones 4

III.- Registros enteros 6

IV.- Registros en punto flotante 6

V.- La consola del sistema 8

b) Editar con un editor de texto plano 8

c) Explorar el segmento de datos

I.- Apartado I. 9

II.- Apartado II. 10

III.- Apartado III. 11

IV.- Apartado IV. 12

V.- Apartado V. 13

VI.- Apartado VI. 13

VII.- Apartado VII. 13

d) Reinicia la máquina y vuelve a cargar el programa en el QtSpim.

I.- Apartado I. 13

II.- Apartado II. 14

III.- Apartado III. 14

IV.- Apartado IV. 15

3.- Bibliografía 16

1 Introducción

En esta práctica aprenderemos a interpretar y manejar un simulador independiente que ejecuta programas en MIPS32. Lee y ejecuta programas en lenguaje ensamblador escritos para su procesador. Además, contamos con un *debugger* y un conjunto mínimo de servicios de sistema operativo.

Utilizaremos anotaciones como IEEE-754 tanto en 32 bits como en 64 bits, además de la anotación hexadecimal.

2 Ejercicios de la práctica

a) Identificar en el emulador los siguientes elementos.

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

Para Usuario:

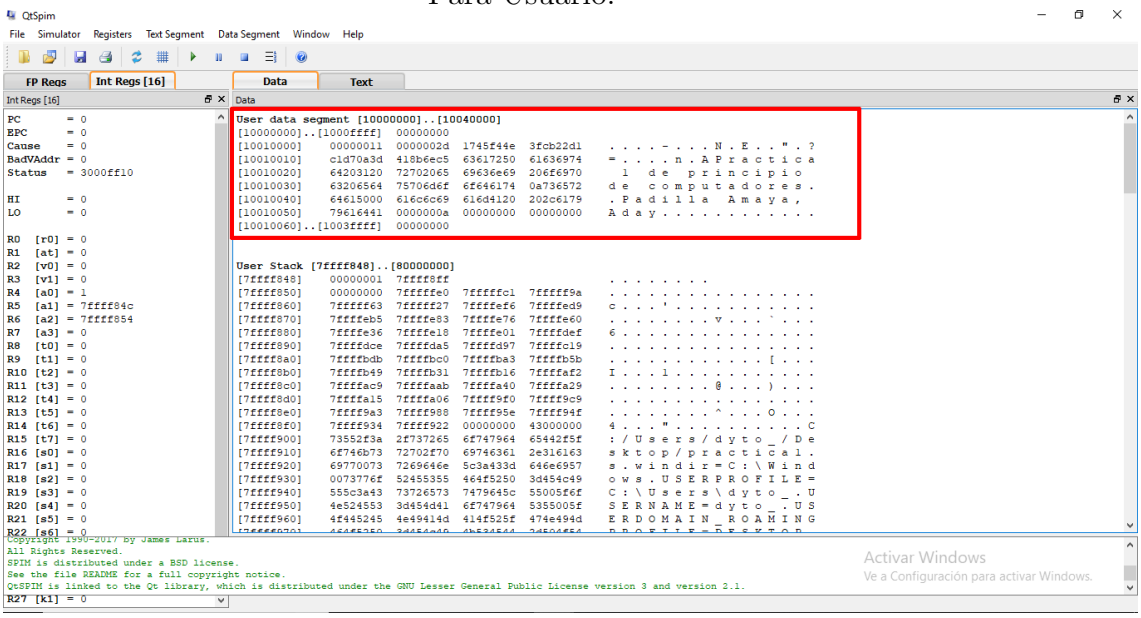

Figura 1: User Data Segment

¿Porqué tiene esa dirección?

Los sistemas basados en procesadores MIPS comúnmente dividen la memoria en tres partes, como podemos ver en la imagen siguiente.

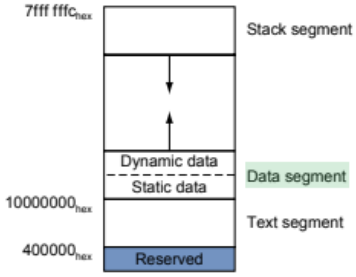

Figura 2:Disposición de la memoria

Ahora entendemos por qué comienza en la posición 10000000_{HEX} los datos de Usuario en el simulador.Además, también podemos ver que en la Figura 1 hay varias columnas donde se indica: la dirección en hexadecimal en el registro,el contenido en hexadecimal de la celda y la representación de los datos en ASCII (de izquierda a derecha respectivamente).

Para Kernel:

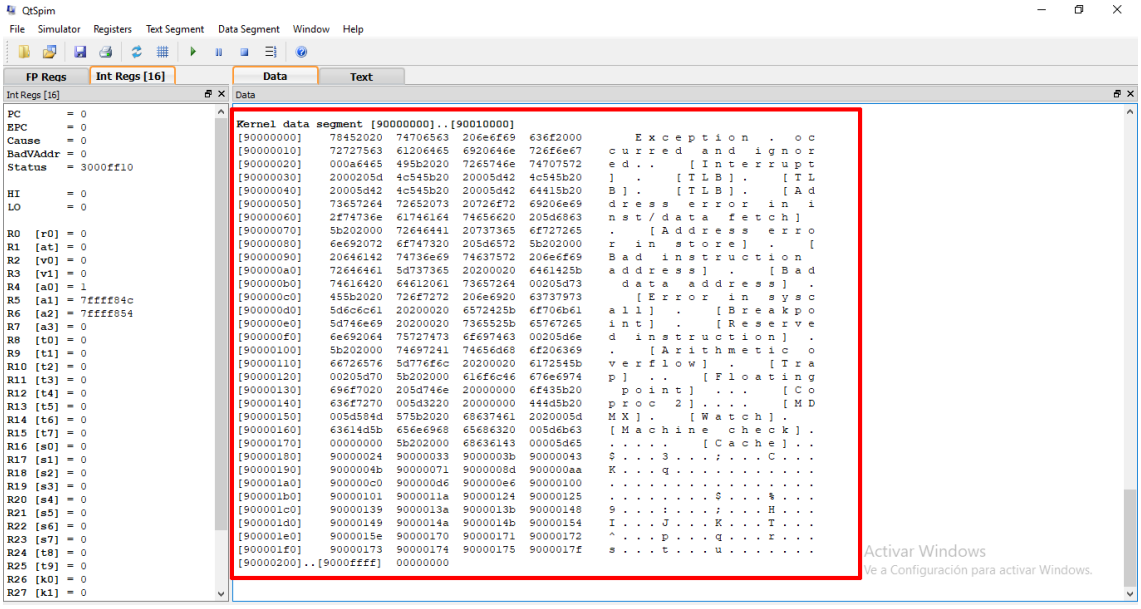

Figura 3: Kernel Data Segment

En el recuadro rojo podemos ver los datos en el kernel. Su bloque de datos acapara desde la posición 90000000_{HEX} hasta la 90010000_{HEX}.

Para la Pila:

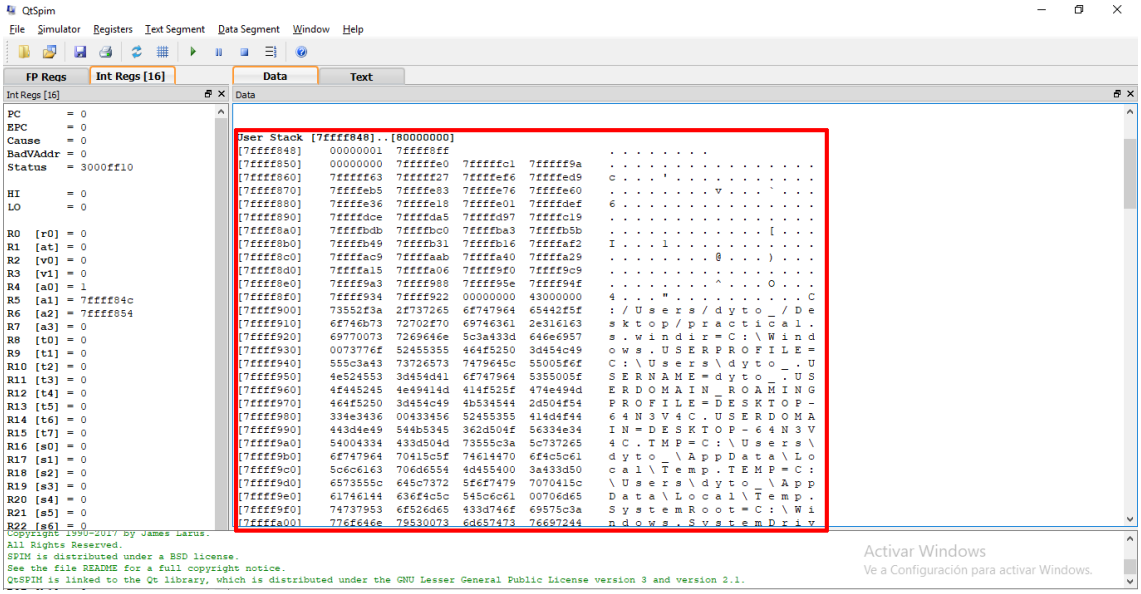

Figura 4: Stack Data Segment

El bloque de la Pila es grande (en comparación con los otros dos) y requeriría de mas de tres imágenes para ilustrar todo su contenido. Sin embargo, podemos ver que comienza en la dirección $7ffff848_{\text{HEX}}$ y acaba en la posición 80000000_{HEX}. Si restamos esas cantidades, obtenemos como resultado que nos ocupa $7B8_{\text{HEX}}$ (1976_{DEC}) direcciones para los bloques de memoria.

II. El segmento de Instrucciones (Usuario y Kernel)

Ahora veremos en el simulador donde se pueden observar las instrucciones cargadas en él y en qué dirección de registro se guardan. Al igual que el segmento de datos, disponemos de varias columnas las cuales indican; 1 direcciones, 2 Opcodes, 3 Mnemónicos y 4 Pseudo-Instrucciones (Figura 5). La primera instrucción se encuentra en la posición $0x00400000_{\text{HEX}}$ de la memoria, siendo la primera celda de las no reservadas (Figura 2).

Para el Usuario:

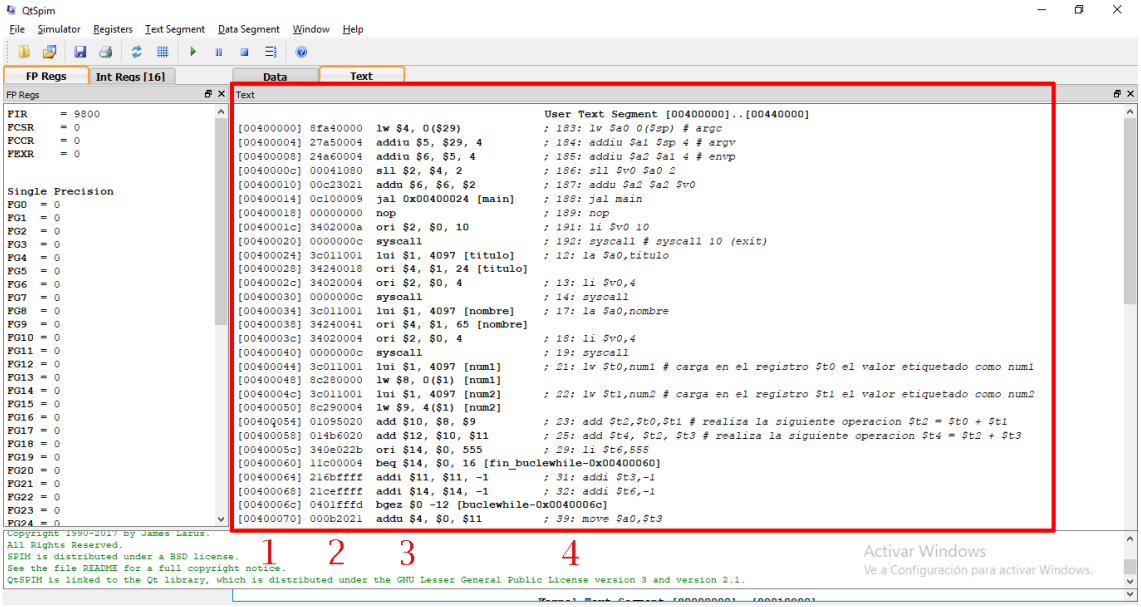

Figura 5: User Text Segment

Para el Kernel:

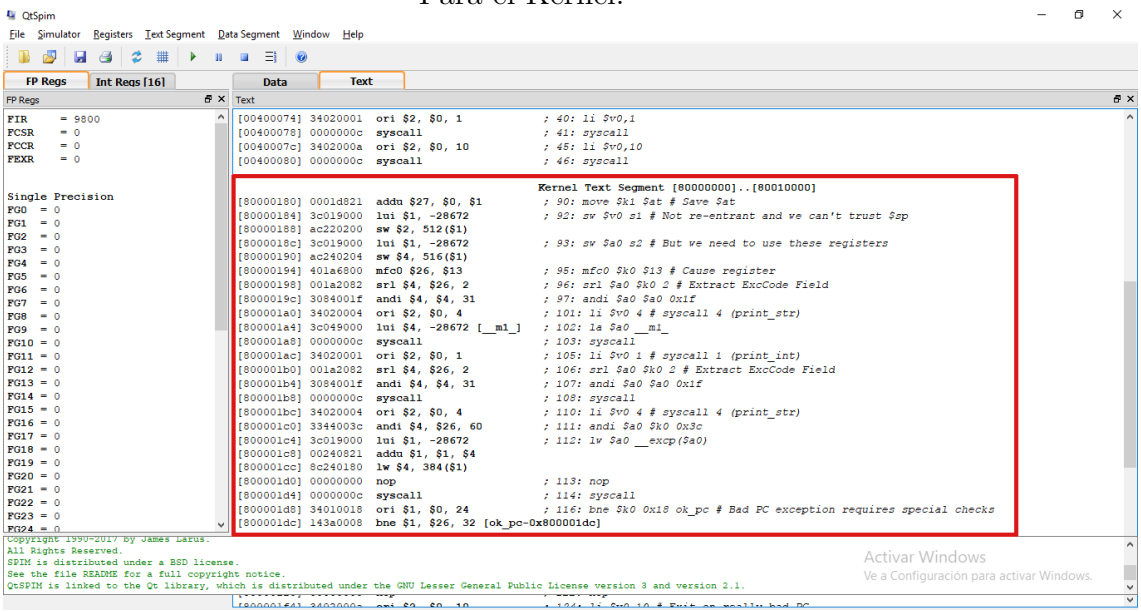

Figura 6: Kernel Text Segment

Comienza al acabar el segmento de instrucciones del usuario.

III. El contenido de los Registros Enteros

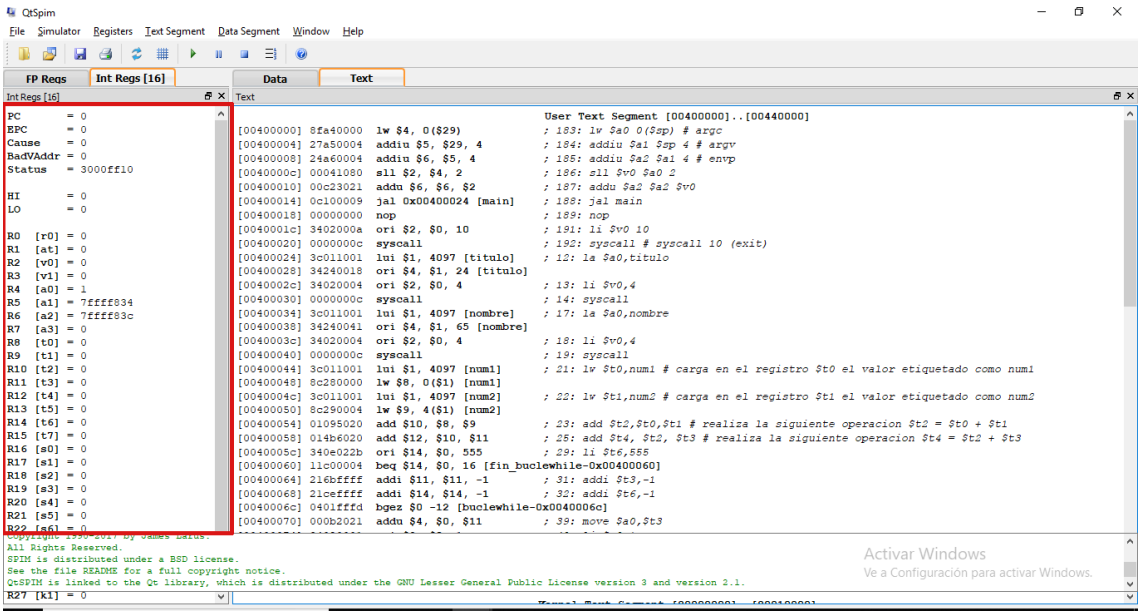

Figura 7: Integer Registers

En la pestaña podemos ver qué valores guardan las variables enteras, desde el PC (Program Counter) hasta las *k0-k1* (Reservadas para el Kernel OS) pasando por las *a0-a3,t0-t7,s0,s7...* Todos ellos pueden ser modificados directamente por nosotros en cualquier momento.

IV. El contenido de los Registros en Punto Flotante

En la Figura 8 se muestran los registros de formato en coma flotante de precisión simple y en la Figura 9 en precisión doble. No olvidar que el simulador los expresa en hexadecimal (tanto los registros, como el segmento de datos y el de instrucciones se pueden cambiar a binario o a decimal si lo prefiere).

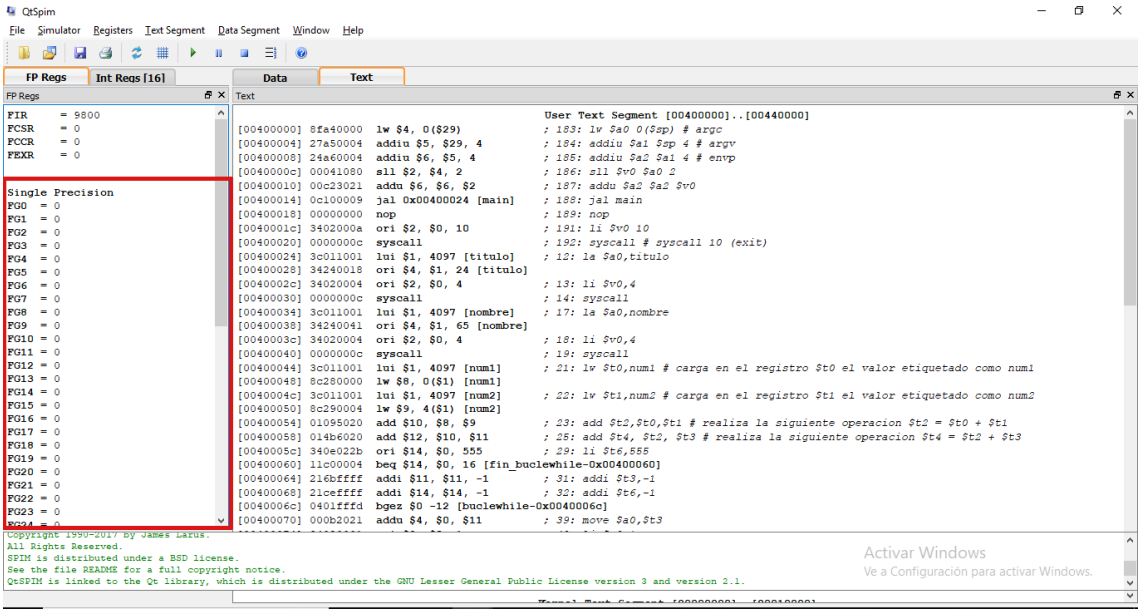

Figura 8: Single Precision Registers

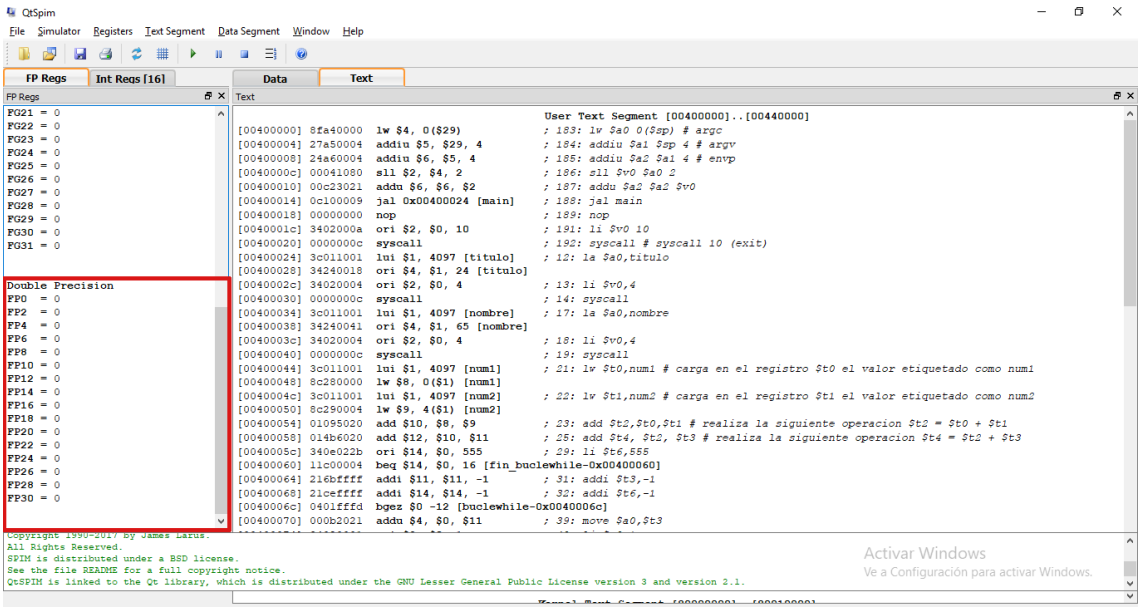

Figura 9: Double Precision Registers

V.La consola del sistema

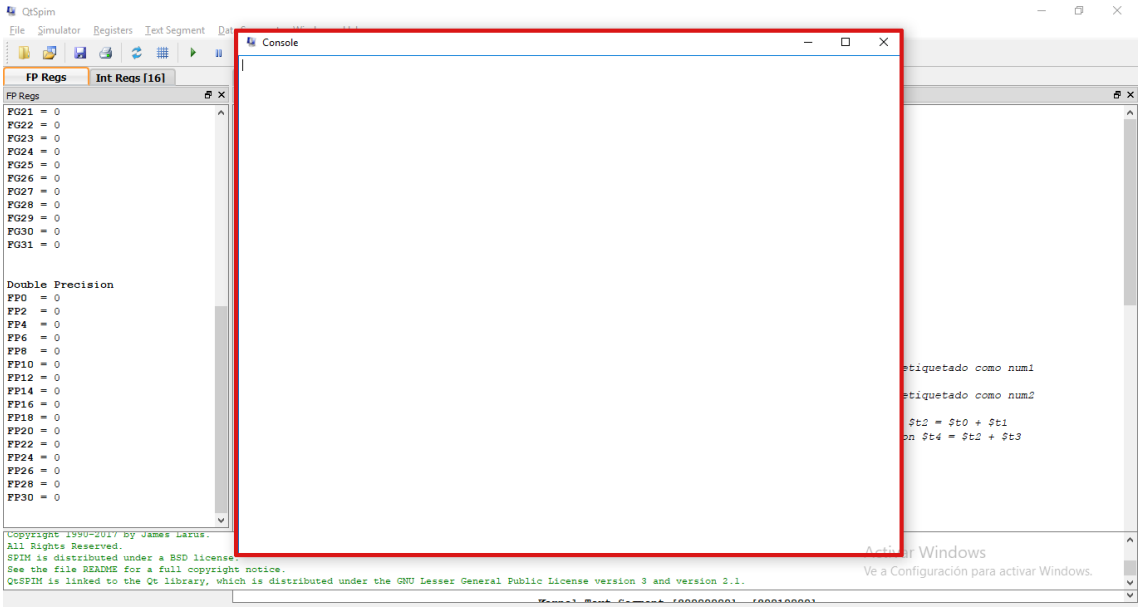

Figura 10: Console window

Si el programa realiza una salida o entrada será dirigida a la consola.

b) Edita con un editor de textos plano.

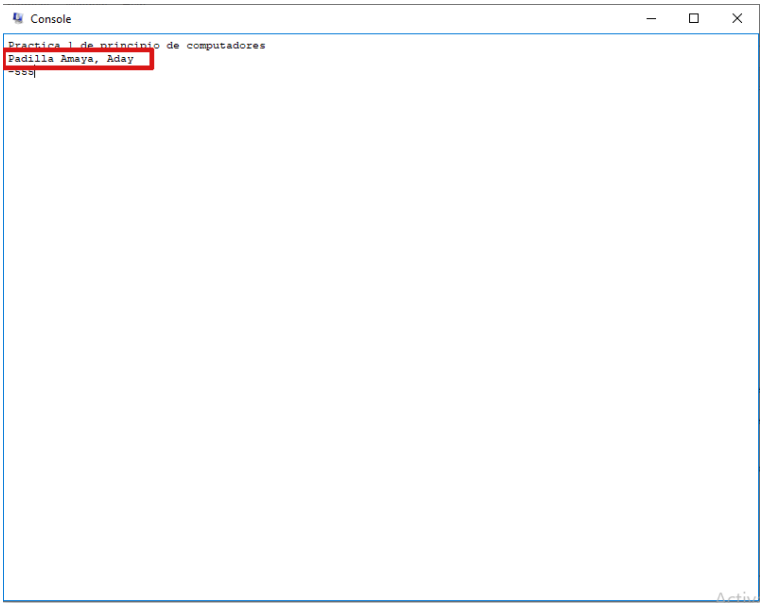

Figura 11: Console window

c) Explora el segmento de datos.

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

Ahora explicaré cómo podemos interpretar el segmento de datos de usuario. En la Figura 12, en la posición 10010040_{HEX} vemos todo el contenido de la memoria en la celda. Empezando desde la primera columna de la izquierda, con el contenido *64615000*, el byte con la dirección de memoria mas baja es el primero de la derecha, es decir, el *00*.

```
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]    64615000  616c6c69  616d4120  202c6179    . P a d i l l a   A m a y a ,
[10010050]    79616441  0000000a  00000000  00000000    A d a y . . . . . . . . . . . .
[10010060]..[1003ffff]  00000000
```

Figura 12: Data Segment

Para una mejor visualización, la Figura 13 muestra como se distribuye cada contenido de la celda con los datos cargados al programa (el color azul representa el valor en hexadecimal y en naranja la posición). Por ejemplo, en la posición '0' vemos que esta guardado el valor 00_{HEX} (que contiene 1 *byte*, es decir, 8 *bits*). No olvidar la posición F_{HEX}, que aunque no muestre nada, ahí hay un byte, porque como dimos en clase, también hay caracteres no imprimibles (tabulador, enter,...). Entonces, la dirección que ocupa la ultima palabra de mi primer apellido (a) sería la 7, es decir, 0x10010047.

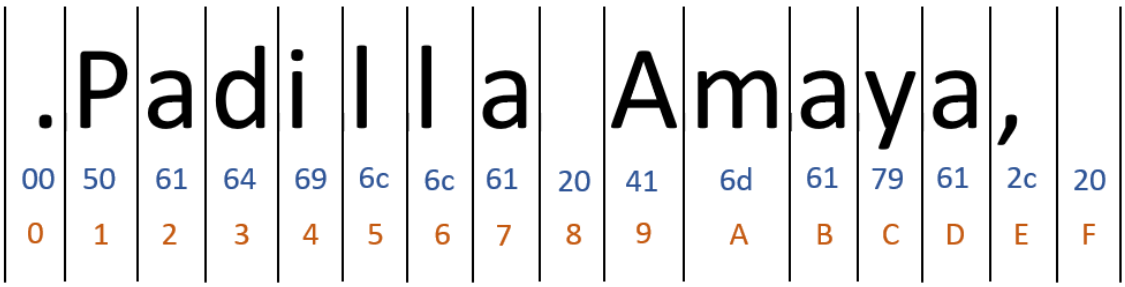

Figura 13: Data Segment with Address

II.¿Qué carácter es y qué representación tiene en hexadecimal?

El carácter correspondiente es la *a* con la representación 0x61.

```
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]    64615000  616c6c69  616d4120  202c6179    . P a d i l l a   A m a y a ,
[10010050]    79616441  0000000a  00000000  00000000    A d a y . . . . . . . . . . .
[10010060]..[1003ffff]  00000000
```

Figura 14: Hexadecimal Data

Podemos comprobarla en la tabla de caracteres US-ASCII.

USASCII code chart

b_7 b_6 b_5 → / b_4 b_3 b_2 b_1 ↓	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Column → / Row ↓	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

Figura 15: US-ASCII Table

III.Busca en el segmento de datos de QtSpim el número que se encuentra en la dirección etiquetada como num3.

Primero miramos qué valor tiene la variable *num3* en nuestro código.

$$\text{num3} = 0\text{x}1745\text{F}44\text{E}$$

Ahora pasamos el valor 0x1745F44E a los caracteres que correspondan:

$17_{\text{HEX}} = ETB$ (END-OF-TRANSMISSION-BLOCK, No imprimible)

$45_{\text{HEX}} = E$

$\text{F}4_{\text{HEX}} = \textit{Out of Range}$ (No imprimible)

$4\text{E}_{\text{HEX}} = N$

USASCII code chart

Bits	b_7 0	0	0	0	1	1	1	1
	b_6 0	0	1	1	0	0	1	1
b_4	b_3	b_2	b_1	Row ↓ Column →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Figura 16: US-ASCII Table

Ahora buscamos el bloque que contenga el valor $1745\text{F}44\text{E}_{\text{HEX}}$ en el segmento de datos de usuario:

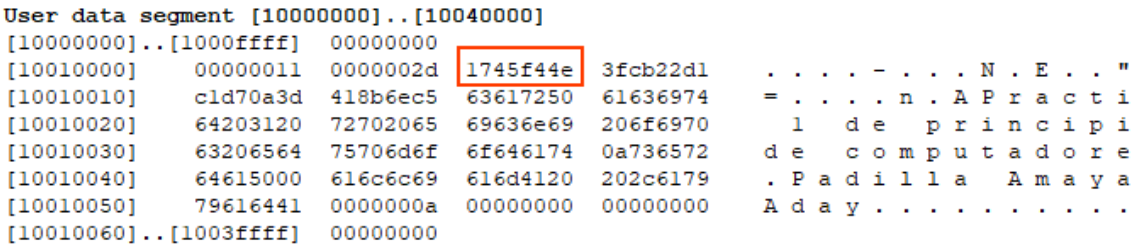

Figura 17: User Data Segment

Por lo que el valor *num3* está entre 0x10010008 y 0x1001000B

IV.Convierte el número 1.587 a formato IEE-754 para 32 bits. Busca ahora este número en el segmento de datos.

El número 1.587 corresponde a $3fcb22d1_{HEX}$. Ahora, como el apartado anterior, los pasamos a los caracteres que corresponda de la tabla ASCII y los buscamos en el segmento de datos de usuario:

$3F_{HEX}$ = *?*

cb_{HEX} = *Out of Range* (No imprimible)

22_{HEX} = ”

dl_{HEX} = *Out of Range* (No imprimible)

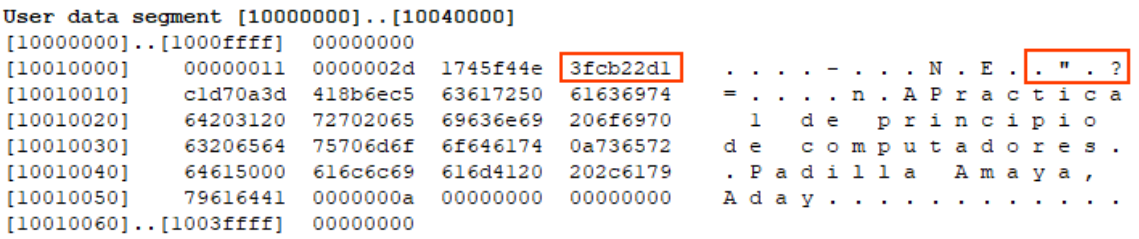

Figura 18: User Data Segment

V.¿En qué dirección empieza el número 1.587?

Como podemos ver en la Figura 18, comienza en la posición 0x100100C y acaba en la 0x100100F.

VI.Convierte el número 57530552.23 a formato IEE-754 para 64 bits.Busca ahora este número en el segmento de datos.

El número 57530552.23_{DEC} equivale a $418B6EC5C1D70A3D_{\text{HEX}}$. Se encuentra en:

```
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]    64615000  616c6c69  616d4120  202c6179    . P a d i l l a   A m a y a ,
[10010050]    79616441  0000000a  00000000  00000000    A d a y . . . . . . . . . . . .
[10010060]..[1003ffff]  00000000
```

Figura 19: User Data Segment

VII.¿En qué dirección empieza el número 57530552.23?.

El número 57530552.23_{DEC} empieza en la posición del registro 0x10010010.

d) Reinicia la máquina y vuelve a cargar el programa en el QtSpim.

I.Ejecuta paso a paso el programa.

Contiene el valor $3E_{\text{HEX}}$ y equivale a 62_{DEC}. Lo podemos corroborar observando el código fuente, ya que *num3 = num2 + num1*, donde *num1 = 17* y *num2 = 45*, luego *num3 = 62*.

II.Cuando hayas terminado de ejecutar esta instrucción, modifica a mano el valor del registro t3.

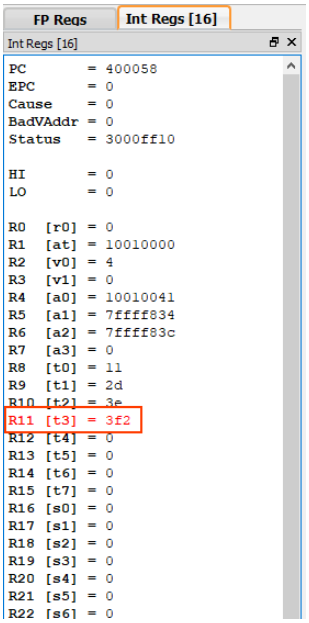

Figura 20: Integer Registers

III.A continuación sigue ejecutando paso a paso hasta terminar de ejecutar la instrucción add t4,t2,t3.

Contiene el valor 430_{HEX}=1072_{DEC}.

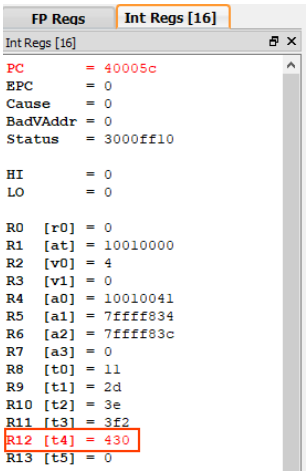

Figura 21: Integer Registers

IV.A continuación establece un punto de ruptura.

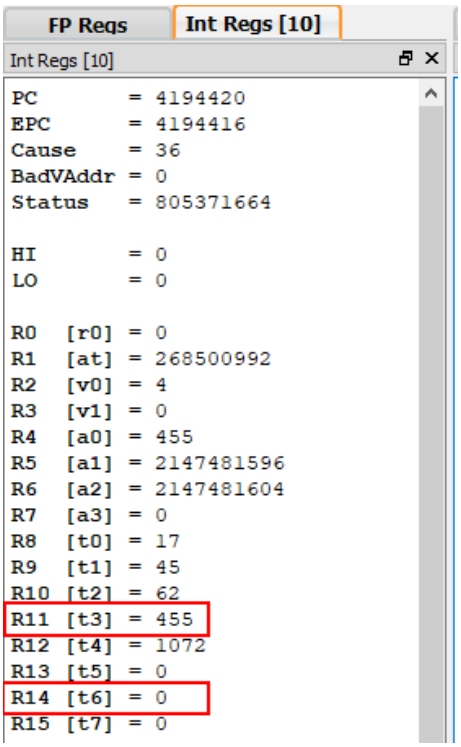

Figura 22: Integer Registers

El resultado es $t3{=}1c7_{HEX}{=}455_{DEC}$ y $t6{=}0_{HEX}{=}0_{DEC}$.

¿Porqué tienen esos valores?

Como en el apartado anterior cambiamos el valor de la variable $t3{=}1010_{DEC}$, el bucle se parará cuando *t3* o *t6* lleguen a *0*. Sabiendo que $t6{=}555_{DEC}$, entonces llegará antes a *0*, así que *t3* será la diferencia de ambos.

$$t3{=}1010 \text{ - } 555 = 455_{DEC}$$

3. Bibliografía

$https://drive.google.com/file/d/1Zye1G-6Sri2KYBAAAze6Sjfn3yJgr_PJ/view$

$https://drive.google.com/file/d/1_64OxIFydYLyI58HECCw1hOC_Yk50nHt/view$

$https://drive.google.com/file/d/1_69X7LQSDRbkkP5XLkl6BXQagF9EJq-v/view$

$Estructuradecomputadores, programaciondelprocesadorMIPSysuensamblador$

$Fundamentosyestructuradecomputadores/JoseM^aAnguloUsategui, JavierGarciaZubia$

$http://spimsimulator.sourceforge.net/$

$http://spimsimulator.sourceforge.net/spim.pdf$

$http://spimsimulator.sourceforge.net/xspim.pdf$

$http://spimsimulator.sourceforge.net/PCSpim.pdf$

$http://spimsimulator.sourceforge.net/SPIM_command-line.pdf$

$https://www.binaryhexconverter.com/hex-to-decimal-converter$

$https://babbage.cs.qc.cuny.edu/ieee-754.old/decimal.html$

$https://www.h-schmidt.net/FloatConverter/IEEE754.html$

$https://www.adobe.com/es/products/photoshopfamily.html$

$https://www.latex-project.org/$

$https://wiki.archlinux.org/index.php/locale$

$https://wiki.archlinux.org/index.php/TeX_Live$