PARTE 1

1.1 Preguntas
1.1.1 Antes de ejecutar el codigo

1. Estudiar las directivas e instrucciones que aparecen en el codigo ¿Cuantas directivas tiene el codigo?

R= Tiene 10 Directivas.

2. Seleccione la pestana llamada Register en la ventana ubicada en la derecha del MARS y note que la mayoria de los registros est´an en cero excepto Stack Pointer ($sp), el Global Pointer ($gp) y el Program Counter ($pc) ¿Cuales son sus valores?

MIPS Register	Value
gp(hexadecimal)	0x10008000
sp(hexadecimal)	0x7fffeffc
pc(hexadecimal)	0x00400000

3. Rellene una tabla como la que se muestra a continuacion, indicando las etiquetas del programa. ¿Cuales son los nombres y direcciones (hexadecimal) de las etiquetas del segmento de programa(Text Segment) y cuales la del segmento datos(Data Segment)?

Text Segment		Data Segment	
Name	Address	Name	Address
main	0x00400000	Valor Caracid	0x10010000 hasta 0x1001000c 0x10010014 hasta 0x1001001c 0x10010020

4. ¿Numero de instrucciones escritas del programa y numero de instrucciones generadas por MARS?

R= Hay 13 instrucciones escritas en el programa y MARS genera 22

5. Identificar la instruccion numero 24 del programa y diga en cuantas instrucciones de lenguaje maquina se transformo?.

R= La instruccion es “lw $s0,Valor($zero)” y se transformo en 3 instrucciones distintas de lenguaje maquina

6. ¿Cuantos bytes de memoria ocupan los datos del programa?.

R= Ocupan 36 bytes

7. ¿Cual es el contenido de la direccion de memoria 0x10010018?

Caracter	P		I		M		,	
Hexadecimal	0x5	0	4	9	4	d	2	c

8.Identifique la instruccion numero 31 de su programa. Escriba los contenidos de las columnas Address, Code y Basic.

Address(hexadecimal)	Code(hexadecimal)	Basic(ensamblador)
0x0040002c	0x7296a802	Mul $21, $20, $22

Haga la conversion de la columna Code a Binario.

Hexadecimal	7				2				9				6				a				8				0				2			
Binary	0	1	1	1	0	0	1	0	1	0	0	1	0	1	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0

1.1.2 Despues de ejecutar el codigo

Seleccione Run->Go desde el Menu de la parte superior de MARS y observe los cambios

1. Estudiar como se cargan en registros las distintas posiciones de un arreglo de datos que esta en memoria. Que operaciones aritmeticas son necesarias realizar (y porque). Observar los valores que han tomado algunos de ellos. Para explicar esos valores, observar las operaciones descritas por las instrucciones de nuestro codigo. ¿Cuantos registros han cambiado de valor?. Rellene una tabla como la que se muestra indicando los cambios que han ocurrido en dichos registros

R= Para cargar las posiciones de un arreglo en la memoria debemos ir aumentando en 4 la direccion en memoria, ya que aunque los indices del arreglo avanzan de 1 en 1, cada uno de esos indices ocupa 4 bytes de memoria. Para esta operacion definimos una constante en un registro que tenga un valor de 4, y luego multiplicaremos el contenido de este registro por el contenido de otro registro en el cual tendremos almacenado el "indice" del arreglo sobre el cual estamos parados. A medida que aumentemos el indice adiccionandole 1, al multiplicarlo por la constante "4" obtendremos cada posicion en la memoria donde se encuentran los valores del arreglo.Han cambiado 12 registros.

Register		Value	
Name	Number	Hexadecimal	Decimal
$at	1	0x00000000	268501004
$v0	2	0x0000000a	10
$s0	16	0x00000008	8
$s1	17	0x00000009	9
$s2	18	0x0000000a	10
$s3	19	0x0000000b	11
$s4	20	0x00000003	3
$s5	21	0x0000000c	12
$s6	22	0x00000004	4
pc		0x00400058	4194392
lo		0x0000000c	12

2. La memoria de datos ha sufrido algun cambio? Muestre el estado del conjunto de direcciones de memoria.

Memory Address	Contenido
0x10010000	0x00000008
0x10010004	0x00000009
0x10010008	0x0000000a
0x1001000c	0x0000000b
0x10010010	0x000a0b1a
0x10010014	0x616c6f48
0x10010018	0x50494d2c
0x1001001c	0x00000053
0x10010020	0x00000001

3. Describa lo que hace el programa :

R= La función del programa es recorrer un arreglo de enteros creado por el programador y guardar cada uno de los componentes del arreglo en un registro distinto.

4. ¿En que direccion se detiene el programa?

R= La ultima instruccion esta en la direccion 0x00400054 y en el registro $v0.

1.1.3 Modificacion del codigo

1. Reeemplace en el codigo las instrucciones 28,31 y 34 (mul $s5,$s4,$s6) por sll $s5,$s4,2 y vuelva a ensamblar el programa y vea que pasa. Ejecute de nuevo el programa. ¿Que ha cambiado?.

R= El registro cambia de 0x00000000 a 0x0000000c en el programa original, pero en el codigo modificado no.

2. Puede identificar si alguna instruccion sobra o no cumple con alguna
funcion?. Identifiquela y diga que hace?

*R= La instrucción **addi $s6,$zero,4**, ya que el shift hace el dezplazamiento dentro del arreglo.*

PARTE 2 SERVICIOS DE ENTRADA Y SALIDA

2.1 Ensamble y Ejecute

1. Ensamble y ejecute el programa paso a paso (Run->Step). Asegurece que la opcion “popup dialog for input syscalls” del Settings en el menu principal, este seleccionado. Fijese en la ventana de mensajes,la pestana Run I/O y escriba las salidas que se van obteniendo.

```
Esto es una cadena
99**** user input : 2
**** user input : fran
fran
44
-- program is finished running  --
```

2. Diga como cambia la memoria de datos durante la ejecucion.

Memory Address	Contenido antes ejecutar	Contenido despues ejecutar
0x10010000	t o s e	n a r f
0x10010004	s e	s \0 \n
0x10010008	a n u	a n u
0x1001000c	e d a c	e d a c
0x10010010	\0 \n a n	\0 \n a n
0x10010014	\0 \0 \0 c	\0 \0 \0 c
0x10010018	\0 \0 \0 v	\0 \0 \0 v
0x1001001c	\0 \0 \0 .	\0 \0 \0 .
0x10010020	\0 \0 \0 !	\0 \0 \0 !
0x10010024	\0 \0 \0 ,	\0 \0 \0 ,
0x10010028	\0 \0 \0 7	\0 \0 \0 7
0x10010034	\0 \0 \0 X	\0 \0 \0 X
0x10010038	\0 \0 \0 c	\0 \0 \0 c

3. Modifique el codigo cambiando la instruccion li $a1,9 por li $a1,28. Y vuelva a ejecutar el programa e introduzca un string de al menos 28 caracteres. Diga como cambia la memoria de datos.

Memory Address	Contenido antes ejecutar	Contenido despues ejecutar
0x10010000	t o s e	4 3 2 1
0x10010004	s e	8 7 6 5
0x10010008	a n u	2 1 0 9
0x1001000c	e d a c	6 5 4 3
0x10010010	\0 \n a n	0 9 8 7
0x10010014	\0 \0 \0 c	4 3 2 1
0x10010018	\0 \0 \0 v	\0 7 6 5
0x1001001c	\0 \0 \0 .	\0 \0 \0 .
0x10010020	\0 \0 \0 !	\0 \0 \0 !
0x10010024	\0 \0 \0 ,	\0 \0 \0 ,
0x10010028	\0 \0 \0 7	\0 \0 \0 7
0x10010034	\0 \0 \0 X	\0 \0 \0 X
0x10010038	\0 \0 \0 c	\0 \0 \0 c