# ESTRUCTURA DE LOS COMPUTADORES

Práctica 3 - Mars

Alejandro Bernabeu Calatayud

abc4.onil4@gmail.com Grupo 40 I2ADE 48788949-S 01/05/18

## Índice

Ejercicio 1	2
Ejercicio 2	
Ejercicio 3	
Ejercicio 4	
Ejercicio 5	
Ejercicio 6	
Ejercicio 7	

#### **Ejercicio 1**

1. Ejecútalo y verifica los datos de la posición de memoria 0x10010020.

2. Cambia la directiva ascii por asciiz y vuelve a compilar el código. ¿Hay diferencias en los valores de dicha posición?

Sí, se puede apreciar como ahora acaba en valor nulo.

Data Segment												
Address	Value (+	-0)			Valu	e (+	4)					
0x10010000	b	0	r	P	32-bit	tval	пе	stor	ed 4	1 by	rtes	s be
0x10010020		\0	s	е	02 01	VŪ	VŪ	\U	Ţυ	,		, ,,
0x10010040	\0	10	\0	\0		\0	\0	\0	\0			
0x10010060	\0	\0	\0	\0		\٥	\0	\0	\0			

3. Sustituye octeto: .byte 0Xf0 por palabra: .word 15. ¿Qué sucede? ¿Por qué? ¿Como solucionarías el problema?

Que el punto final se va a la siguiente dirección de memoria. Con el space 8 para reservar n bytes de espacio en el segmento de datos

segment							
Address	Value (+0)	Value (+4)	Value (+8)				
0x10010000	borP	odna	s a l				
0x10010020	\0 \0 s e	\0 \0 \0 .	\0 \0 \0 \0				
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0				
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0				

4. Mediante el uso de la directiva .half genera la cadena de caracteres HOLA de modo que se almacene en una única palabra ubicada en la posición de memoria0x10010028.

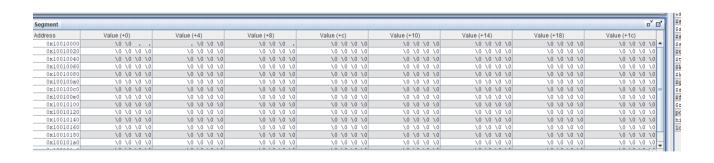
```
mips1.asm

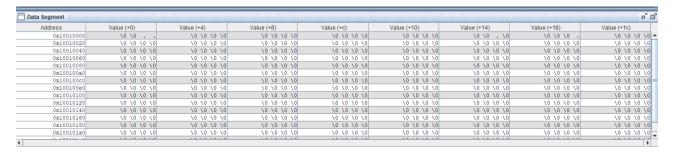
1  # Ejercicio 1
2  .data # Segmento de datos
3  cadena: .asciiz"Probando las cadenas de caracteres" # defino string
4  octeto: .word 15
5
6  cadena2: .ascii "HOLA"
7  plaabra: .half
```

a Segment			
Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	b o r P	o d n a	<u>s a l</u>
0x10010020	\0 \0 s e	\0 \0 \0 .	A L O H
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

#### **Ejercicio 2**

1. Ejecuta el código anterior sin la directiva .align y observa la colocación de los datos en memoria. Vuelve a ejecutar el programa incluyendo la directiva.





.align Alinea el siguiente dato sobre un límite de 2<sup>n</sup> byte, como hemos introducido 4 lo alinea 32 bytes después

2. Comenta como ha afectado a la alineación de half, espacio, byte y palabra en memoria.

Half, Almacena 0xABCD cantidades de 16 bits en posiciones consecutivas de memoria.

Espacio, ha reservado 5 bytes en el segmento de datos Byte, Almacena 0x11(hexadecimal) cantidades de 8 bits en posiciones consecutivas de memoria.

Palabra, Almacena 22 cantidades de 32 bits en posiciones consecutivas de memoria

3. ¿Se podría utilizar la memoria reservada a espacio para guardar una palabra? Razónalo.

Sí, esta memoria se puede reservar para poder utilizarla después.

4. Indica las diferentes alternativas de almacenamiento que tendríamos en espacio para almacenar byte, half y palabra.

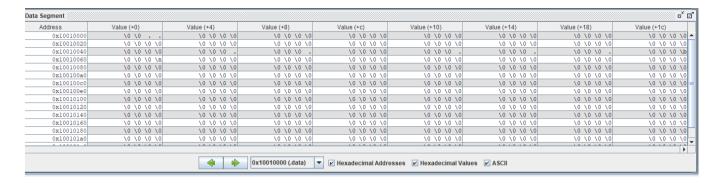
Almacena palabra (sw), Almacena media palabra (sh) y Almacena Byte (sb). La principal diferencia es el tamaño que ocupa lo que guardan.

#### Ejercicio 3

1. Ejecuta el programa paso a paso y comprueba cómo cambian los valores de los registros y de las posiciones de memoria. Cambia el valor de align por 6 y space por 4. ¿Qué sucede?

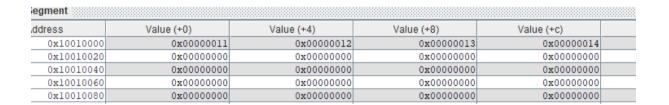
ata Segment								<b>់</b> 🗹
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	\0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 . \0	\0 \0 \0 .	\0 \0 \0 \0
0x10010020	\0 \0 \0 .	\0 \0 \0 .	\0 \0 \0 .	d/ 0/ 0/ 0/	\0 \0 \n	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100e0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010100	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010140	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010160	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010180	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100101a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
- 100101								

#### Con los valores cambiados:



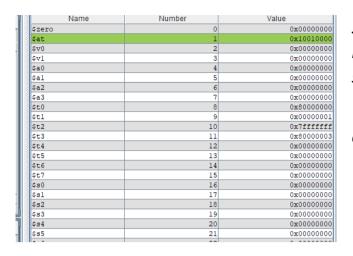
Como hemos cambiado la alineación con .align y la cantidad de bytes que hemos reservado, simplemente los valores están en una posición de memoria diferente, más espaciados ahora.

2. Realiza un programa que almacene a partir de la dirección de memoria 0x10010000 los siguientes datos 0x11, 0x12, 0x13 y 0x14. El programa convertirá cada uno de estos cuatro bytes en cuatro palabras almacenas a partir de la posición de memoria 0x10010020 manteniendo su valor.



#### **Ejercicio 4**

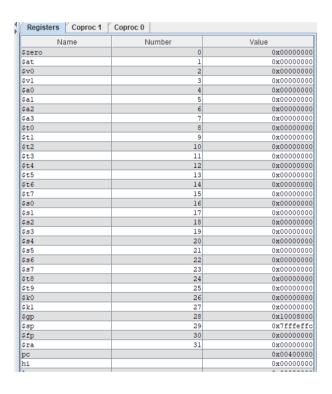
1. Ejecuta el programa paso a paso y comprueba como cambian los valores de los registros y de las posiciones de memoria. Es correcto el resultado mostrado. Razónalo.



Sí, es correcto, tenemos la resta de 800000-1 en t2 y la suma de 80000 + 3 en t3.

Y después las guardamos en otras posiciones de memoria.

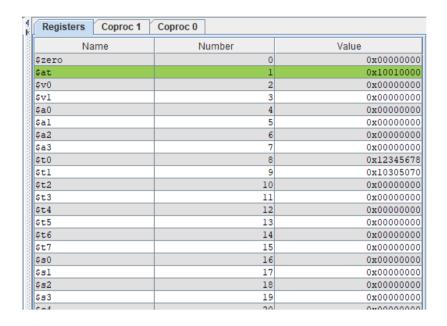
2. Cambia la instrucciones subiu por subi y ejecuta el programa paso a paso. ¿Qué sucede? ¿Por qué?



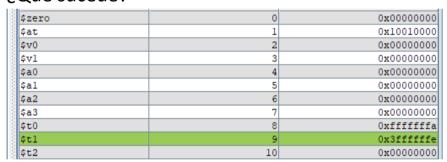
Ahora la resta se realiza teniendo en cuenta el signo por lo que el resultado y la posición de memoria son diferentes.

#### **Ejercicio 5**

 Ejecuta el programa paso a paso y comprueba cómo cambian los valores de los registros y de las posiciones de memoria.

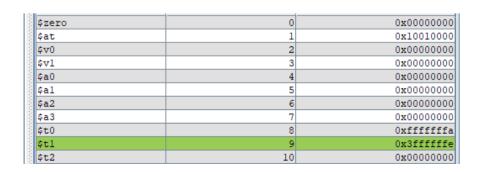


2. Explica que sucede con el número almacenado en el registro \$t1. ¿Que bits ha añadido a la izquierda?¿Qué operación matemática se ha producido? Cambia sla por srl. ¿Qué sucede?



Que se ve desplazado aritméticamente a la derecha.

3. Si cambiamos sra por srl el desplazamiento es lógico y no aritmético.



### **Ejercicio 6**

Ejecuta el programa y comprueba el resultado.
 Modifícalo para que muestre la cadena de texto
 introducida. Analiza que sucede si la cadena tiene más
 de 10 caracteres. Elimina los caracteres \n de string2.
 ¿Qué sucede? Añade un mensaje que nos indique que
 introduzcamos el mensaje "Finalizar programa (S/N)?"

2. y lea dicho carácter que posteriormente mostraremos.