

PRÁCTICA I – ASA II

“Codificación de datos e instrucciones”

A. SEGMENTO DE TEXTO

1. ¿Cuál es la longitud de las instrucciones?

En la totalidad del programa se ejecutan 6 instrucciones con 4 bytes de longitud por cada una de ellas. Esto se puede apreciar mediante la dirección de memoria en el apartado “Text Segment” del programa Mars. En el registro PC también puedo visualizarlo a medida que voy continuando con las instrucciones.

2. ¿Cuál es la dirección de la segunda instrucción?

La dirección de memoria de la segunda instrucción en hexadecimal es la 0x00400004

3. ¿Qué tipo de ordenación se utiliza? ¿Cómo lo sabes?

Se utiliza la ordenación de tipo “Little Endian” ya que el byte menos significativo se encuentra en la posición más significativa. Esto se puede ver ya que las palabras, una vez están codificadas en código ASCII, aparecen al revés.

4. Teniendo en cuenta los campos de cada una de las instrucciones, ¿es correcta la codificación en hexadecimal?

Realizaré la codificación con la instrucción “addiu” que necesita de tres parámetros: un registro temporal, un RS y un inmediato.

Addition immediate (without overflow)

addiu rt, rs, imm	9	rs	rt	imm
	6	5	5	16

Son necesarios 6bits + 5bits + 5bits + 16bits = 32bits (4 bytes)

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	4194304	0x3c011001	lui \$1,0x00001001	4: main: la \$a0, str
<input type="checkbox"/>	4194308	0x34240000	ori \$4,\$1,0x00000000	
<input type="checkbox"/>	4194312	0x24020004	addiu \$2,\$0,0x00000004	5: li \$v0, 4
<input type="checkbox"/>	4194316	0x0000000c	syscall	6: syscall
<input type="checkbox"/>	4194320	0x2402000a	addiu \$2,\$0,0x0000000a	7: li \$v0, 10
<input type="checkbox"/>	4194324	0x0000000c	syscall	8: syscall

Luego verifico que el valor del rt es 2, el valor del rs es 0 y el inmediato es 4. Procedemos a ubicar cada uno de esos valores en hexadecimal y tendré:

6BITS	5BITS	5BITS	16BITS
001001	00000	00010	0000000000000100

Empiezo a separar de Der. a Izq. Cada 4 bits, entonces:

0x24020004

Hagamos lo mismo pero para la instrucción *ori \$4,\$1,0Xhex0*

OR immediate

ori rt, rs, imm	Oxd	rs	rt	imm
	6	5	5	16

Put the logical OR of register rs and the zero-extended immediate into register rt.

6BITS	5BITS	5BITS	16BITS
001101	00001	00100	0000000000000000

Empiezo a separar de Der. a Izq. Cada 4 bits, entonces:

0x34240000

5. Clasifica cada una de las instrucciones según su tipo, I, R o J.

LUI = LOAD UPPER IMMEDIATE (TIPO I)

ORI = OR IMMEDIATE (TIPO I)

ADDIU = ADDITION IMMEDIATE (TIPO I)

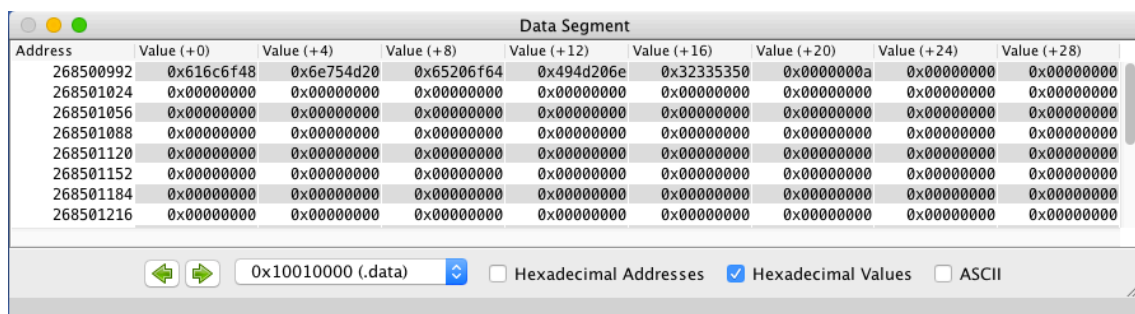
SYSCALL = SISTEM CALLING

6. Indicar el modo de direccionamiento empleado por cada una de las instrucciones.

Existe un direccionamiento con desplazamiento.

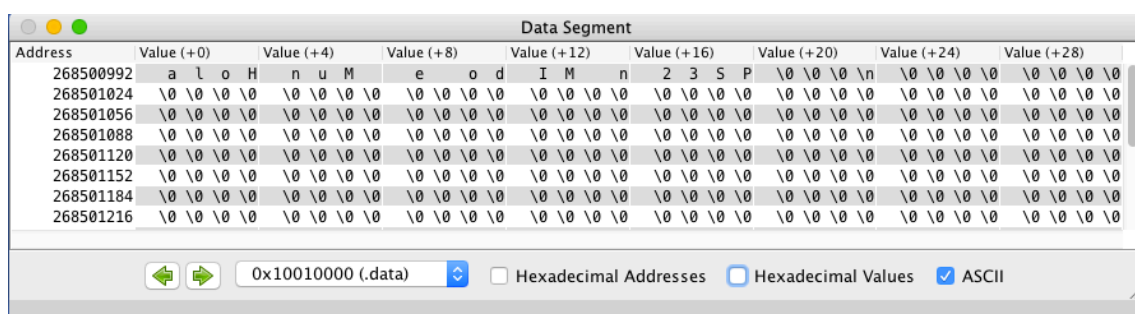
B. SEGMENTO DE DATOS

1. ¿Cuál es la longitud de una dirección de memoria?



Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	0x616c6f48	0x6e754d20	0x65206f64	0x494d206e	0x32335350	0x0000000a	0x00000000	0x00000000
268501024	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
268501056	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
268501088	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
268501120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
268501152	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
268501184	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
268501216	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Valores de direcciones de memoria del segmento de datos en Hexadecimal



Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	a	l	o	H	n	u	M	e
268501024	\0	\0	\0	\0	\0	\0	\0	\0
268501056	\0	\0	\0	\0	\0	\0	\0	\0
268501088	\0	\0	\0	\0	\0	\0	\0	\0
268501120	\0	\0	\0	\0	\0	\0	\0	\0
268501152	\0	\0	\0	\0	\0	\0	\0	\0
268501184	\0	\0	\0	\0	\0	\0	\0	\0
268501216	\0	\0	\0	\0	\0	\0	\0	\0

Valores de direcciones de memoria del segmento de datos en ASCII

Tienen una longitud de 32 bits.

2. ¿Cuál es la dirección del dato str?

Es la dirección 0x10010000, esto lo sé al mirar la primera posición de memoria.

3. ¿Qué tipo de ordenación se utiliza? ¿Cómo lo sabes?

Little-Endian. Debido a lo que explicado anteriormente, al codificar con ASCII se lee de manera invertida.

4. ¿cuántos bytes emplea la variable str? ¿cuántas palabras ocupa en memoria?

Emplea 21 bytes (cada palabra ocupa 4 bytes) ocupando 5 palabras en memoria.

C. MÁS TIPOS DE DATOS

1. ¿ Cuántos bytes emplea la variable de tipo .asciiz? ¿cuántas palabras ocupa en memoria?

Emplea 16 bytes. Cada palabra ocupa un total de 4 bytes, por ende ocupa un total de 4 palabras en memoria.

2. ¿Cuántos bytes emplea la variable de tipo .byte? ¿a qué estructura de datos te recuerda?

Emplea 11 Bytes.

D. ALMACENAMIENTO DE BYTES

1.¿Cómo se almacenarían en memoria los siguientes datos (BYTES)?

Exactamente igual.

E. ARRAYS Y CARGA DE MEMORIA A REGISTRO

MFLO = Move from Lo