ALEJANDRO REYES ALBILLAR 45931406-S

PRÁCTICA 3 ESTRUCTURA DE LOS COMPUTADORES

1º) Se nos pide que completemos una serie de tablas sobre el ejercicio propuesto en la práctica 3.

1. Tras implementar el código especificado y haberlo guardarlo con el nombre “ejercicio propuesto 1.asm” se ha comprobado que compilaba correctamente y la ejecución producía los resultados correctos que son mostrados en los sucesivos apartados del ejercicio.

|  |  |  |
| --- | --- | --- |
| **Dirección de Memoria** | **Contenido INICIAL** | **Contenido FINAL** |
| 0x10000000 | 0x00000000 | 0x11112222 |
| 0x10000004 | 0x00000000 | 0x88884444 |
| 0x10000008 | 0x00000000 | 0x43211234 |
| 0x1000000c | 0x00000000 | 0x00007bf8 |
| 0x10000010 | 0x00000000 | 0x11112222 |

1. Se nos pide mostrar los valores antes y después de la ejecución del programa en una tabla, la cual mostramos a continuación:

Hay que tener en cuenta que estos resultados solamente se muestran si se realiza la ejecución paso a paso, no directamente sobre el simulador de MARS debido a que esta opción realiza un limpiado de memoria al finalizar la ejecución.

1. Los registros van cambiando según avanza la ejecución del programa y cuando termina dicha ejecución sus valores son los siguientes:

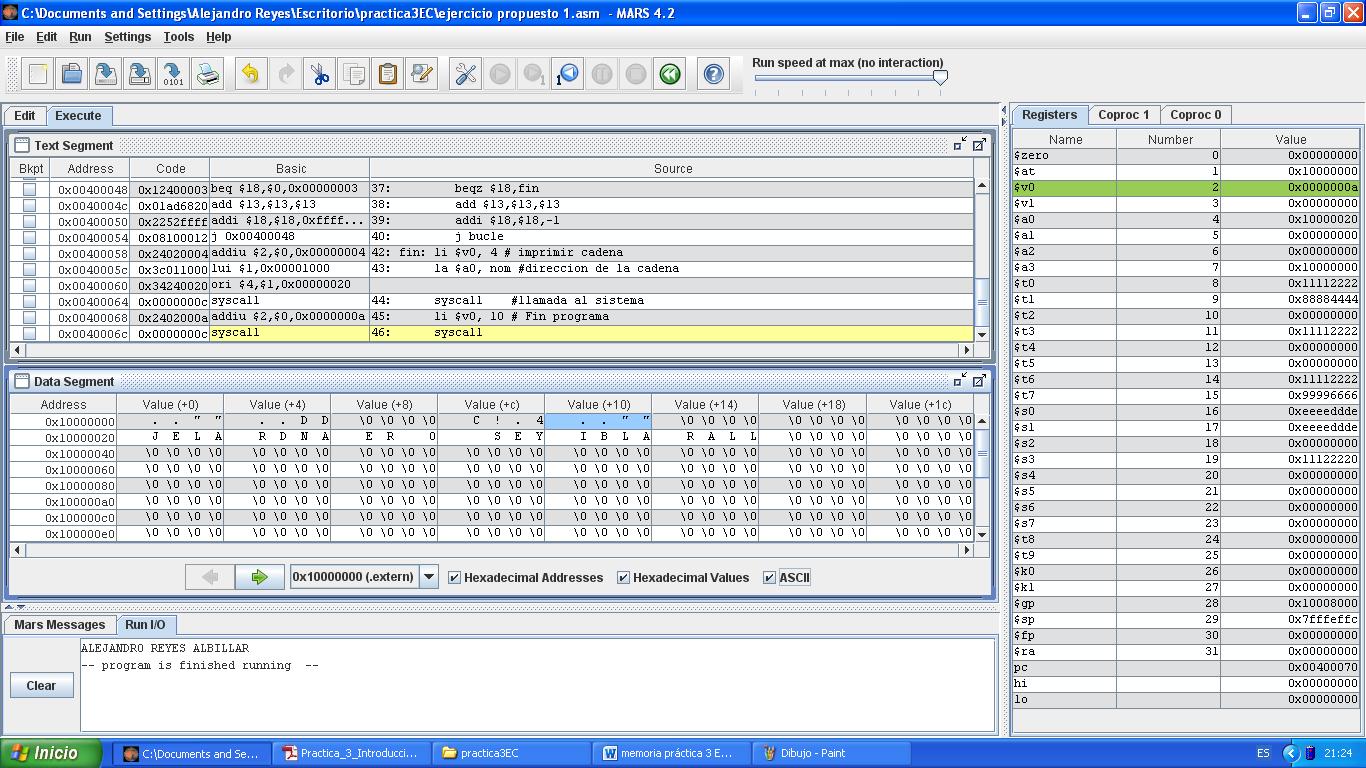
|  |  |  |
| --- | --- | --- |
| **Registro** | **Contenido INICIAL** | **Contenido FINAL** |
| $at | 0x00000000 | 0x10000000 |
| $a3 | 0x00000000 | 0x10000000 |
| $t0 | 0x00000000 | 0x11112222 |
| $t1 | 0x00000000 | 0x88884444 |
| $t2 | 0x00000000 | 0x43211234 |
| $t3 | 0x00000000 | 0x11112222 |
| $t4 | 0x00000000 | 0x00001234 |
| $t5 | 0x00000000 | 0x000001a0 |
| $t6 | 0x00000000 | 0x11113456 |
| $t7 | 0x00000000 | 0x99996666 |
| $s0 | 0x00000000 | 0x320ff012 |
| $s1 | 0x00000000 | 0xeeeef012 |
| $s2 | 0x00000000 | 0x00000000 |
| $s3 | 0x00000000 | 0x23030160 |
| Hi | 0x00000000 | 0x00000003 |
| Lo | 0x00000000 | 0x777aeee8 |
| pc | 0x00400000 | 0x00400060 |
| $v0 | 0x00000000 | 0x0000000a |

Los valores mostrados en esta tabla son los obtenidos tras la última iteración del programa, lo que quiere decir que los valores de los registros no son los mismos que los mostrados en la tabla durante toda la ejecución.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instrucción Ensamblador** | **Instrucción código máquina** | **Tipo** | **opcode**  **(6)** | **rs**  **(5)** | **rt**  **(5)** | **rd**  **(5)** | **immediate (16)** | **instr\_index (26)** | **sa**  **(5)** | **function**  **(6)** |
| la $7,0x10000000 | 0x3c011000 | I | 001111 | 00000 | 00001 | - | 0001000000000000 | - | - | - |
|  | 0x34270000 | R | 001101 | 00001 | 00111 | 00000 | - | - | 00000 | 000000 |
| lw $8, 0($7) | 0x8ce80000 | I | 100011 | 00111 | 01000 | - | 0000000000000000 | - | - | - |
| lw $9, 4($7) | 0x8ce90004 | I | 100011 | 00111 | 01001 | - | 0000000000000100 | - | - | - |
| lw $10, 8($7) | 0x8cea0008 | I | 100011 | 00111 | 01010 | - | 0000000000001000 | - | - | - |
| move $11, $8 | 0x00085821 | I | 000000 | 00000 | 01000 | - | 0101100000100001 | - | - | - |
| lh $12, 8($7) | 0x84ec0008 | I | 100001 | 00111 | 01100 | - | 0000000000001000 | - | - | - |
| lb $13, 8($7) | 0x80ed0008 | I | 100001 | 00111 | 01101 | - | 0000000000001000 | - | - | - |
| sw $11, 16($7) | 0xaceb0010 | I | 101011 | 00111 | 01011 | - | 0000000000010000 | - | - | - |
| add $14, $11, $12 | 0x016c7020 | R | 000000 | 01011 | 01100 | 01110 | - | - | 00000 | 100000 |
| add $15, $9, $11 | 0x012b7820 | R | 000000 | 01001 | 01011 | 01111 | - | - | 00000 | 100000 |
| sub $16, $10, $11 | 0x014b8022 | R | 000000 | 01010 | 01011 | 10000 | - | - | 00000 | 100010 |
| sub $17, $12, $11 | 0x018b8822 | R | 000000 | 01100 | 01011 | 10001 | - | - | 00000 | 100010 |
| divu $8, $12 | 0x010c001b | I | 000000 | 01000 | 01100 | - | 0000000000011011 | - | - | - |
| mult $8, $13 | 0x010d0018 | I | 000000 | 01000 | 01101 | - | 0000000000011000 | - | - | - |
| mfhi $18 | 0x00009010 | J | 000000 | - | - | - | - | 00000000001001000000010000 | - | - |
| xor $19, $10, $11 | 0x014b9826 | R | 000000 | 01010 | 01011 | 10011 | - | - | 00000 | 100110 |
| sll $19, $19, 4 | 0x00139900 | R | 000000 | 00000 | 10011 | 10011 | - | - | 00100 | 000000 |
| beqz $18, fin | 0x12400003 | I | 000100 | 10010 | 00000 | - | 0000000000000011 | - | - | - |
| add $13, $13, $13 | 0x01ad6820 | R | 000000 | 01101 | 01101 | 01101 | - | - | 00000 | 100000 |
| addi $18, $18, -1 | 0x2252ffff | R | 001000 | 10010 | 10010 | 11111 | - | - | 11111 | 111111 |
| jbucle | 0x08100012 | J | 000010 | - | - | - | - | 00000100000000000000010010 | - | - |
| fin: li $v0, 10 | 0x2402000a | I | 010010 | 00000 | 00100 | - | 0000000000001010 | - | - | - |
| syscall | 0x0000000c | J | 000000 | - | - | - | - | 00000000000000000000001100 | - | - |

1. En la siguiente tabla se muestra la codificación de cada una de las instrucciones del código visto en código máquina, indicando su tipo y sus diferentes campos según el mismo. La sucesión de unos y ceros de las columnas a la derecha de la columna tipo son la traducción a binario de la instrucción en código máquina expresada en forma hexadecimal.

Hay que tener en cuenta que la instrucción “la” del ensamblador es interpretada como dos instrucciones diferentes en código máquina, más específicamente como las instrucciones “lui” y “ori”, cuyas traducciones a código máquina están escritas respectivamente en las dos primeras filas de la tabla anterior.

1. Se nos pide que modifiquemos el código que nos proporciona el enunciado para que, cuando se llegue a la etiqueta “fin” se muestre el nombre del alumno y que, para ello, utilicemos la instrucción “syscall” que es una llamada al sistema.

Una vez realizada la modificación necesaria para que imprima por pantalla el nombre del alumno podemos observar en la siguiente imagen cómo la ejecución es la deseada:

Para evitar que se solapara la información hemos tenido que redireccionar el valor de la cadena ASCII, ya que en caso de no hacerlo las 2 primeras letras del nombre del alumno se perdían mostrándose unos cuadros en su lugar.

A continuación se muestra el código modificado para que funcione de la manera solicitada por el ejercicio:

.data 0x10000000

num1: .word 0x11112222

.word 0x88884444

.word 0

.half 0x1234

.half 0x4321

.byte -8

.byte 123

.data 0x10000020 #dirección para evitar errores

nom: .asciiz "ALEJANDRO REYES ALBILLAR"

.align 2

.text

main:

la $7,0x10000000

lw $8,0($7)

lw $9,4($7)

lw $10,8($7)

move $11,$8

lh $12,8($7)

lb $13,8($7)

sw $11,16($7)

add $14,$11,$12

add $15,$9,$11

sub $16,$10,$11

sub $17,$12,$11

divu $8,$12

mult $8,$13

mfhi $18

xor $19,$10,$11

sll $19,$19,4

bucle:

beqz $18,fin

add $13,$13,$13

addi $18,$18,-1

j bucle

fin: li $v0, 4 # imprimir cadena

la $a0, nom #direccion de la cadena

syscall #llamada al sistema

li $v0, 10 # Fin programa

syscall

**EL CÓDIGO DEL EJERCICIO ASÍ COMO LA CAPTURA DE PANTALLA SE ENCUENTRAN DENTRO DEL ZIP QUE CONTIENE ESTA MEMORIA**