

# clear\_arreglo\_vs\_punteros.s# 2016/06/02

# Autor: Jorge Ayala

# Ejemplo Array vs Pointer

#

.data

Arreglo: .word 3, 0, 1, 2, 6, -2, 4, 7, 3, 7

Arreglo2: .word 3, 0, 1, 2, 6, -2, 4, 7, 3, 7

tamano: .word 10

msg: .asciiz "Fin"

#Declaro main como una funcion global

.globl main

.text

# La etiqueta main representa el punto de partida

main:

#ARGUMENTOS DE FUNCION

la $a0, Arreglo

lw $a1, tamano

jal clearArreglo

la $a0, Arreglo2

jal clearPuntero

li $v0, 4 # syscall 4 (print\_str)

la $a0, msg # argument: string

syscall

j salir

clearArreglo:

addi $sp, $sp, -16

sw $t0, 12($sp)

sw $t1, 8($sp)

sw $t2, 4($sp)

sw $t3, 0($sp)

move $t0, $zero # i = 0

loop1: sll $t1, $t0, 2 # $t1 = i \* 4

add $t2, $a0, $t1 # $t2 = direccion de array [i]

sw $zero, 0($t2) # array[i] = 0

addi $t0, $t0, 1 # i = i + 1

slt $t3, $t0, $a1 # $t3 = (i < size) slt set on less than

bne $t3, $zero, loop1 # si (i<size) ir a loop1

lw $t3, 0($sp)

lw $t2, 4($sp)

lw $t1, 8($sp)

lw $t0, 12($sp)

addi $sp,$sp,16

jr $ra

clearPuntero:

addi $sp, $sp, -16

sw $t0, 12($sp)

sw $t1, 8($sp)

sw $t2, 4($sp)

sw $t3, 0($sp)

move $t0, $a0

sll $t1,$a1,2

add $t2,$a0,$t1

loop2: sw $zero, 0($t0)

addi $t0, $t0, 4

slt $t3, $t0, $t2

bne $t3, $zero, loop2

lw $t3, 0($sp)

lw $t2, 4($sp)

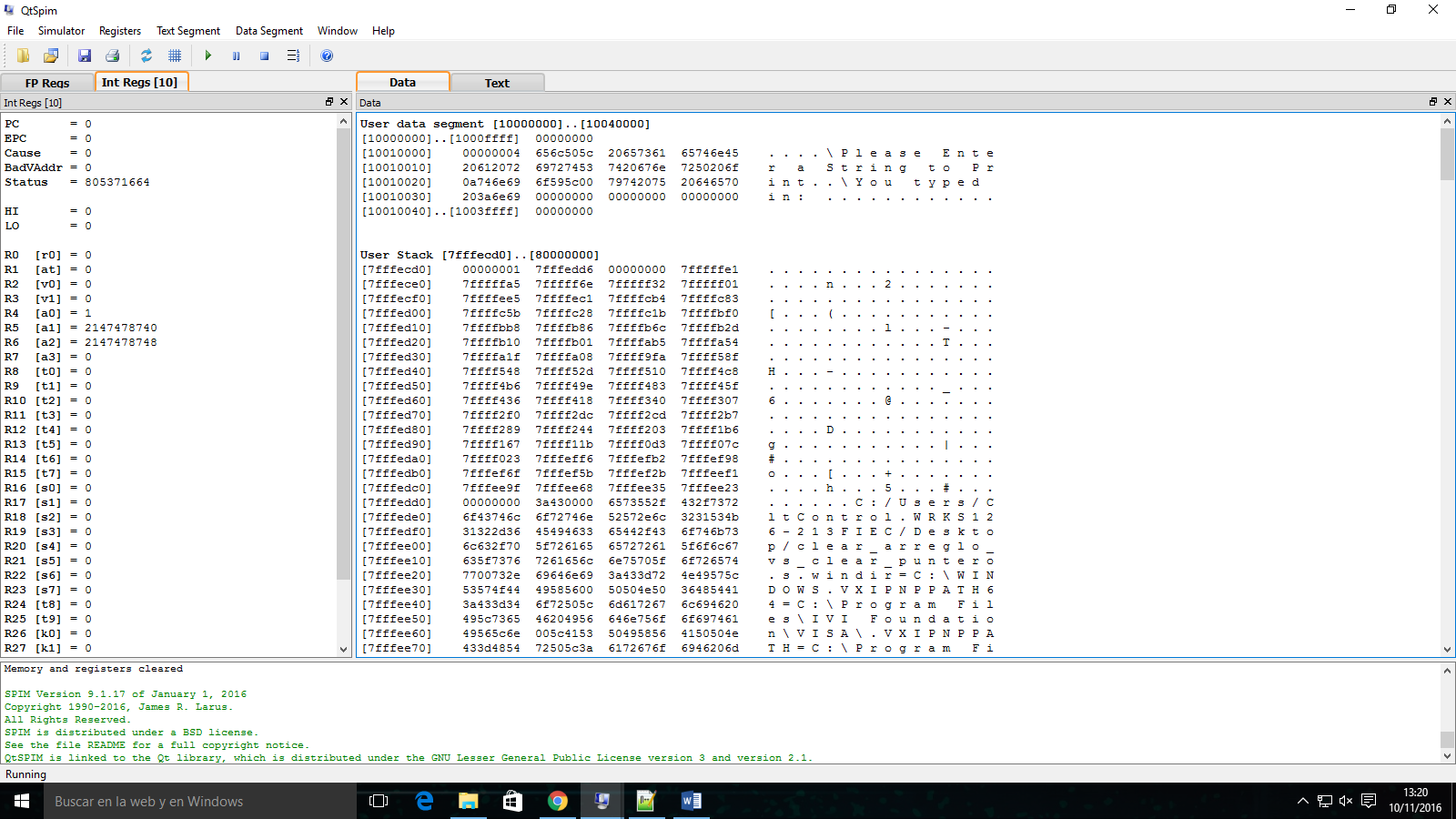
lw $t1, 8($sp)

lw $t0, 12($sp)

addi $sp,$sp,16

jr $ra

salir:



#A Program that asks for your input then outputs what you said.

#Author: Philip Matuskiewicz

#Mips Code

.data #let processor know we will be submitting data to program now

insert\_into:

.word 4 #make a 4 byte (32 bit) space in memory for a word with address insert\_into

Ask\_Input:

.asciiz "\Please Enter a String to Print\n" #in unused memory store this string with address Ask\_Input

Tell\_Output:

.asciiz "\You typed in: " #in unused memory store this string with address Tell\_Output

.text #enables text input / output, kind of like String.h in C++

main: #main function is always called in any mips program, so the program will start here with actual assembly code

la $a0, Ask\_Input #load address Ask\_Input from memory and store it into arguement register 0

li $v0, 4 #loads the value 4 into register $v0 which is the op code for print string

syscall #reads register $v0 for op code, sees 4 and prints the string located in $a0

la $a0, insert\_into #sets $a0 to point to the space allocated for writing a word

la $a1, insert\_into #gets the length of the space in $a1 so we can't go over the memory limit

li $v0, 8 #load op code for getting a string from the user into register $v0

syscall #reads register $v0 for op code, sees 8 and asks user to input a string, places string in reference to $a0

la $a0, Tell\_Output #load address Tell\_Output from memory and store it into arguement register 0

li $v0, 4 #loads the value 4 into register $v0 which is the op code for print string

syscall #reads register $v0 for op code, sees 4 and prints the string located in $a0

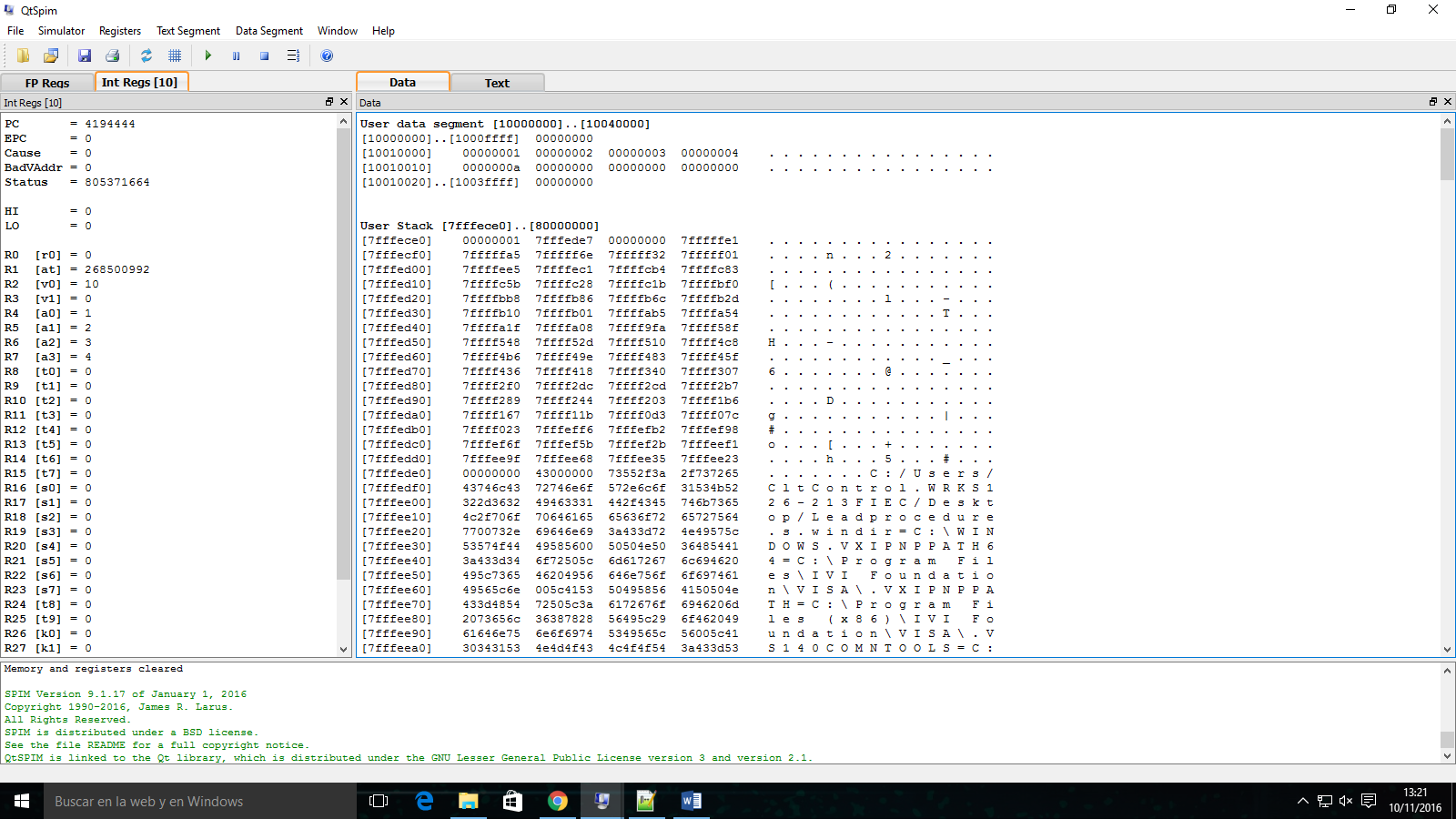
la $a0, insert\_into #load address insert\_into from memory and store it into arguement register 0

li $v0, 4 #loads the value 4 into register $v0 which is the op code for print string

syscall #reads register $v0 for op code, sees 4 and prints the string located in $a0

li $v0, 10 #loads op code into $v0 to exit program

syscall #reads $v0 and exits program



.data

var1: .word 1#g

var2: .word 2#h

var3: .word 3#i

var4: .word 4#j

varResultado: .word 0

#Declaro main como una funcion global

.globl main

.text

# La etiqueta main representa el punto de partida

main:

#ARGUMENTOS DE FUNCION

lw $a0, var1

lw $a1, var2

lw $a2, var3

lw $a3, var4

jal leafExample# LLAMO A LA FUNCION

sw $v0, varResultado

j exit

leafExample:

#addi $sp, $sp, –12 # adjust stack to make room for 3 items

addi $sp, $sp, -12

sw $t1, 8($sp) # save register $t1 for use afterwards

sw $t0, 4($sp) # save register $t0 for use afterwards

sw $s0, 0($sp)

#REALIZO LAS OPERACIONES CON REGISTROS TEMPORALES

add $t0, $a0, $a1

add $t1, $a2, $a3

add $s0,$t0,$t1 #

add $v0,$s0,$zero # returns f ($v0 = $s0 + 0)

lw $s0, 0($sp) # restore register $s0 for caller

lw $t0, 4($sp) # restore register $t0 for caller

lw $t1, 8($sp) # restore register $t1 for caller

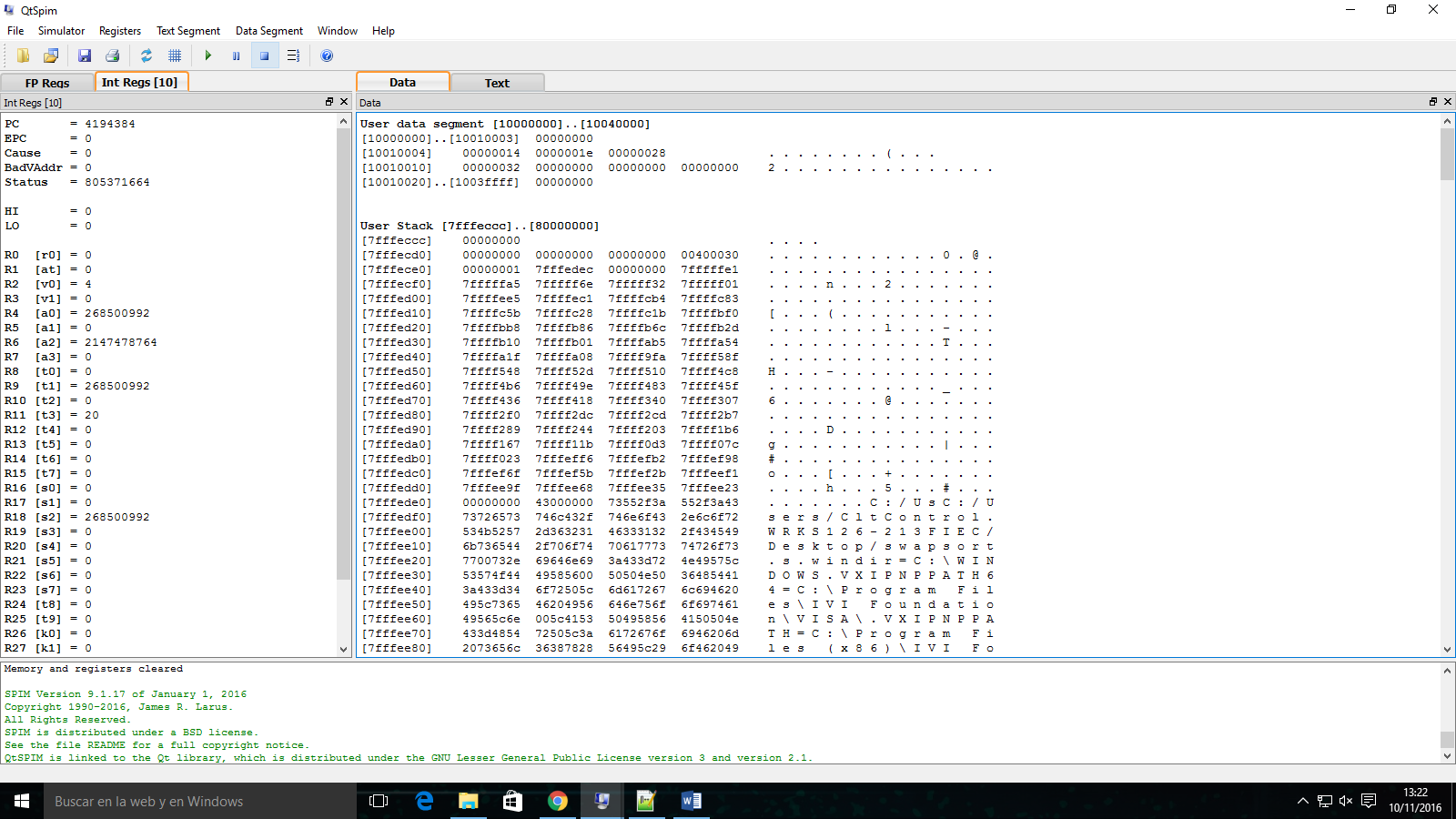
addi $sp,$sp,12 # adjust stack to delete 3 items

jr $ra

exit:

li $v0, 10 #loads op code into $v0 to exit program #El registro 0 lo utiliza el sistema para ejecutar una funcionalidad, por medio de syscall

syscall #reads $v0 and exits program



exit:.data

array: .word 40, 30, 50, 20

.globl main

.text

main:

la $a0, array

li $a1, 5

jal sort

sort:

addi $sp, $sp, -20

sw $ra, 16($sp)

sw $s3, 12($sp)

sw $s2, 8($sp)

sw $s1, 4($sp)

sw $s0, 0($sp)

move $s2, $a0

move $s3, $a1

move $s0, $zero

for1:

slt $t0, $s0, $s3

beq $t0, $zero, exit1

addi $s1, $s0, -1

for2:

slti $t0, $s1,0

bne $t0, $zero, exit2

sll $t1, $s1, 2

add $t2, $s2, $t1

lw $t3, 0($t2)

lw $t4, 4($t2)

slt $t0, $t4, $t3

beq $t0, $zero, exit2

move $a0, $s2

move $a1, $s1

jal swap

addi $s1, $s1, -1

j for2

exit2:

addi $s0, $s0, 1

j for1

exit1:

lw $s0, 0($sp)

lw $s1, 4($sp)

lw $s2, 8($sp)

lw $s3, 12($sp)

lw $ra, 16($sp)

addi $sp, $sp, 20

jr $ra

swap:

sll $t1, $a1, 2

add $t1, $a0, $t1

lw $t0, 0($t1)

lw $t2, 4($t1)

sw $t2, 0($t1)

sw $t0, 4($t1)

jr $ra