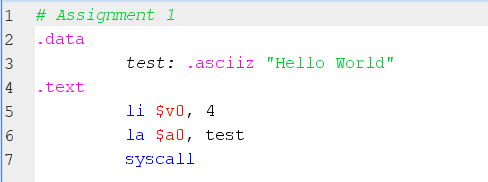
**BÀI THỰC HÀNH TUẦN 5**

**KIẾN TRÚC MÁY TÍNH**

Họ và tên: Đinh Huy Dương

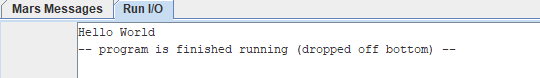
MSSV: 20215020

**Bài 1:**

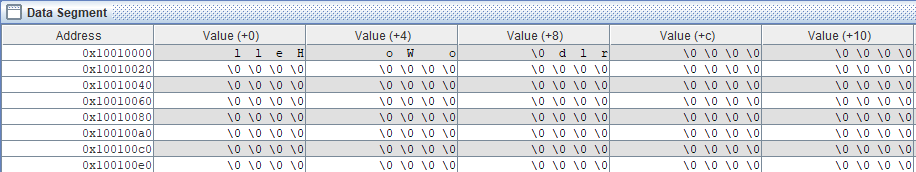


- Chương trình thực hiện việc in ra màn hình chuỗi xâu “Hello World” ra màn hình qua dịch vụ syscall 4

- Kết quả:



- Vùng nhớ của biến “test” ở Data Segment:



Khác với mảng (kiểu word, mỗi phần tử chiếm 4 byte), mỗi phần tử của xâu sẽ được lưu trữ ở kiểu “byte” chỉ chiếm 1 byte, nên một ô nhớ 32 bit có thể lưu trữ được 4 phần tử của xâu. Phần tử đứng trước sẽ được lưu vào byte thấp hơn và sẽ lưu đến phần tử NULL (\0).

**Bài 2:**

#Assignment 2

.data

message1: .asciiz "The sum of "

message2: .asciiz " and "

message3: .asciiz " is "

.text

addi $s0, $zero, 32 # $s0 = 32

addi $s1, $zero, 64 # $s1 = 64

add $s2, $s0, $s1 # $s2 = $s0 + $s1

li $v0, 4

la $a0, message1

syscall

li $v0, 1

add $a0, $zero, $s0

syscal

li $v0, 4

la $a0, message2

syscall

li $v0, 1

add $a0, $zero, $s1

syscall

li $v0, 4

la $a0, message3

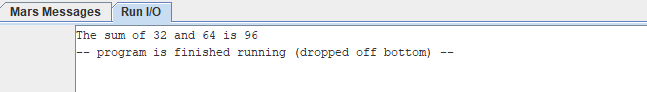
syscall

li $v0, 1

add $a0, $zero, $s2

syscall

Kết quả:



**Bài 3:**

# Assignment 3

.data

x: .space 32 # destination string x, empty

y: .asciiz "LigmaBallz" # source string y

.text

strcpy:

add $s0,$zero,$zero # $s0 = i = 0

la $a0, x # $a0 = the address of x

la $a1, y # $a1 = the address of y

L1:

add $t1,$s0,$a1 # $t1 = $s0 + $a1 = i + y[0]

# = address of y[i]

lb $t2,0($t1) # $t2 = value at $t1 = y[i]

add $t3,$s0,$a0 # $t3 = $s0 + $a0 = i + x[0]

# = address of x[i]

sb $t2,0($t3) # x[i]= $t2 = y[i]

beq $t2,$zero,end\_of\_strcpy # if y[i] == 0, exit

nop

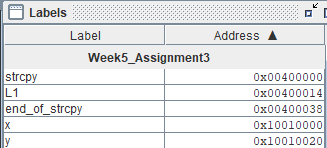
addi $s0,$s0,1 # $s0 = $s0 + 1 <-> i = i + 1

j L1 # next character

nop

end\_of\_strcpy:

Địa chỉ của x và y:



Gán địa chỉ của x và y vào trong thanh ghi $a0 và $a1 qua lệnh “la”, khởi tạo giá trị của i =0 tại thanh $s0

Khi đó, ta sẽ thực hiện vòng lặp với nhãn “L1”, cộng địa chỉ ban đầu với giá trị i để ra được địa chỉ của phần tử y[i] được lưu vào $t1.

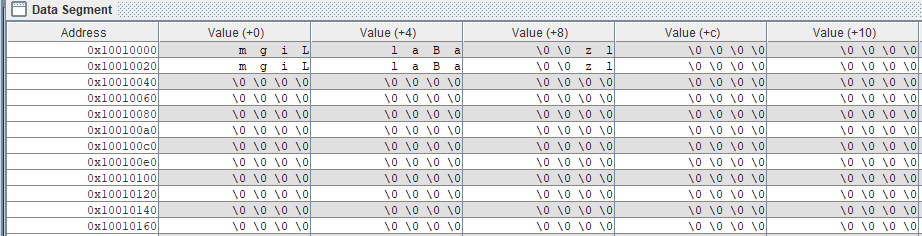
- Sử dụng lệnh “lb” để lưu giá trị tại địa chỉ của y[i] ở $t1 vào trong $t2. Lý do sử dụng “lb” mà không phải “lw” vì 1 phần tử của xâu y chỉ sử dụng 1 byte, kiểu byte, trong khi word sử dụng 4 byte.

- Lấy địa chỉ của x[i] tương tự với y[i] vào thanh ghi $t3. Và sử dụng lệnh “sb” để gán giá trị tại thanh $t2 vào trong thanh có địa chỉ chứa tại $t3.

- So sánh giá trị của y[i] với 0 = NULL, nếu bằng thì nhảy đến kết thúc. Nếu không, tăng i lên 1 và tiếp tục lặp.

- Ở đây có sử dụng lệnh “nop” là chỉ thị không để làm gì cả. Cách giải thích ở đây có thể là làm trống bộ nhớ khi chương trình được thực hiện dưới kiến trúc đường ống (pipeline), khi nhiều lệnh được sử dụng đồng thời tại cùng 1 chu kỳ xung nhịp và làm quá tải máy.

- Kết thúc chương trình, ta nhận được giá trị của y tại địa chỉ bộ nhớ của x:



**Bài 4:**

.data

string: .space 50

Message1: .asciiz "Nhap xau: "

Message2: .asciiz "Do dai xau la: "

.text

main:

get\_string:

li $v0,54

la $a0, Message1

la $a1, string # Input the string

la $a2, 50 # Maxmimum length of the string

syscall

get\_length:

la $a0, string # $a0 = address(string[0])

add $t0, $zero, $zero # $t0 = i = 0

check\_char:

add $t1, $a0, $t0 # $t1 = $a0 + $t0

# = address(string[i])

lb $t2, 0($t1) # $t2 = string[i]

beq $t2, $zero, end\_of\_str # is null char?

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

j check\_char

end\_of\_str:

end\_of\_get\_length:

print\_length:

li $v0, 56

la $a0, Message2

add $a1,$zero, $t0 # the interger to be printed is $t0

syscall # execute

Graphical user interface, application

Description automatically generated

Kết quả:

Graphical user interface, application, Word

Description automatically generated

Giải thích: Mặc dù xâu có 4 ký tự, nhưng MIPS sau khi nhập xong xâu, kết thúc sẽ cho thêm ký tự \n (xuống dòng) ở cuối:



**Bài 5:**

# Assignment 5

.data

string: .space 21

rvstr: .space 21

Message1: .asciiz "Insert the character of the string: "

Message2: .asciiz "\nThe reverse string: "

.text

get\_string:

li $v0,4

la $a0, Message1

syscall

la $a0, string

la $a1, rvstr

add $t0, $zero, $zero # $t0 = i = 0

add $t2, $zero, 0xa # $t2 = '\n'

addstr:

beq $t0, 20, eof\_addstr # finish input if the number of char >20

add $t1, $a0, $t0 # $t1 = address of string[i]

li $v0, 12

syscall

sb $v0, 0($t1)

beq $v0, $t2, eof\_addstr # finish input if string[i] = '\n'

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

j addstr

eof\_addstr:

beq $v0, $t2, reverse # if string[n] = \n, jump to ”reverse”

addi $t1, $t1,1 # $t1++ so it can loop properly on “loop”

reverse:

add $s0,$zero,$zero # $s0 = i = 0

loop:

subi $t1, $t1, 1 # $t1 = $t1 -1

lb $t2,0($t1) # $t2 = value at $t1 = y[i]

add $t3,$s0,$a1 # $t3 = $s0 + $a0 = i + x[0]

# = address of x[i]

sb $t2,0($t3) # x[i]= $t2 = y[i]

beq $t2,$zero,end\_reverse # if y[i] == 0, exit

nop

addi $s0,$s0,1 # $s0 = $s0 + 1 <-> i = i + 1

j loop # next character

nop

end\_reverse:

printstring:

li $v0, 4

la $a0, Message2

syscall

li $v0, 4

la $a0, rvstr

syscall

Sử dụng syscall 12 để nhập từng ký tự ở trong vòng lặp “addstr”. Các ký tự được nhập sẽ được lưu trong $v0. Khi đó ta có thể so sánh $v0 với ‘\n’ = 0xa ở mã ASCII được lưu ở $t2. Bên cạnh đó, ta đặt lệnh “beq” ở đầu để rẽ nhánh khi đã nhập đến phần tử thứ 20. Kết thúc nhập, nếu kết thúc ở ‘\n’, ta nhảy thẳng qua “reverse”, nếu không ta tăng 1 cho địa chỉ của ký tự cuối tại $t1. Chúng ta làm vậy bởi trong nhãn “reverse”, ta dùng lệnh “subi” ngay đầu để trừ dần ngược về địa chỉ đầu. Nếu ta nhập 20 ký tự, phần tử kết thúc sẽ không phải là ‘\n’, khiến cho khi đảo xâu, ta sẽ bị mất đi ký tự thứ 20, trong khi đó nếu nhập ‘\n’, thì trừ 1 cho giá trị địa chỉ sẽ bỏ qua ký tự’\n’, vẫn thực hiên được phép đảo.

Sau đó sử dụng lại bài 3 để có thể sao chép ngược xâu vào xâu “rvstr”

Từ đó ta sử dụng syscall 4 để có thể in ra xâu “rvstr”

Kết quả:

* Trường hợp kết thúc bằng ‘\n’:

Graphical user interface, text, email

Description automatically generated

* Trường hợp kết thúc khi nhập quá 20 ký tự:

