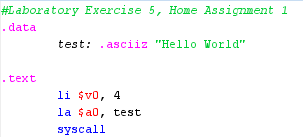
**BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH (IT3280) TUẦN 5**

***Họ và tên: Phạm Đức Long***

***MSSV: 20225737***

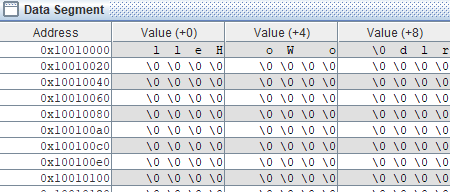
Assignment 1:



* Kết quả thu được:



* Data Segment:



* Nhận xét: Ta thấy chuỗi “Hello World” được lưu vào các ô nhớ theo thứ tự ngược lại ứng với mỗi Value.
* Sự thay đổi giá trị trên thanh ghi:

|  |  |  |  |
| --- | --- | --- | --- |
|  | $v0 | $a0 | pc |
| Ban đầu | 0x00000000 | 0x00000000 | 0x00400000 |
| Sau lệnh li | 0x00000004 | - | 0x00400004 |
| Sau lệnh la | - | 0x10010000 | 0x0040000c |
| Sau lệnh syscall | - | - | 0x00400010 |

**🡪** Mỗi kí tự khi được lưu trong thanh ghi sẽ tốn 1 byte trong thanh ghi, do đó cứ một ký tự sẽ được lưu vào một địa chỉ của thanh ghi. Theo đó, các ký tự từ trái sang phải (Từ string[0] đến string[n – 1]) sẽ được lưu trữ bắt đầu bằng việc chuyển sang hệ thập lục phân (16), bắt đầu từ ô Value (+0), nạp từ 2 vị trí cuối của thanh ghi cho đến khi lưu trữ đủ 4 ký tự.. Khi Value (+0) đầy thì chuyển sang các cột Value (+4), Value (+8),… cho đến khi lưu trữ đầy đủ các ký tự, ở đây là “Hello World”.

A screenshot of a computer

Description automatically generated

* Lấy ví dụ như ở Value (+0), ta có ‘H’ trong bảng mã ascii là 48, vậy ta điền 48 vào 2 bit cuối, tương tự có ‘e’ là 65, ‘l’ là 6c => 0x6c6c6548 = l/ l/ e/ H
* Tương tự, ta có 0x6f57206f = o/ W/ (space)/ o . 0x00646c72 = 0/ d/ l/ r .
* Kết luận:
* Chuỗi được lưu trữ như trên, mỗi Value sẽ lưu trữ tối đa 4 ký tự của chuỗi (tương ứng với việc tối đa 4 ký tự sẽ được đóng gói trong 1 cột Value), bắt đầu từ Value (+0) cho đến khi hết chuỗi.
* Giả sử nếu chuỗi có n ký tự, khi đó sẽ cần [n/4] + 1 cột (với [x] là phần nguyên của x).

Assignment 2:

* Code:

**.data**

**m1: .asciiz "First num: "**

**m2: .asciiz "Second num: "**

**m3: .asciiz "The sum of "**

**m4: .asciiz " and "**

**m5: .asciiz " is "**

**.text**

**# Nhap so dau tien:**

**li $v0, 4**

**la $a0, m1**

**syscall**

**li $v0, 5**

**syscall**

**move $a1, $v0 # Cho so dau tien vao thanh ghi $a1**

**# Nhap so thu hai:**

**li $v0, 4**

**la $a0, m2**

**syscall**

**li $v0, 5**

**syscall**

**move $a2, $v0 # Cho so thu hai vao thanh ghi $a1**

**# Tinh tong hai so vua nhap:**

**add $a3, $a1, $a2 # Cho tong hai so vao thanh ghi $a3**

**# In ra:**

**# In "The sum of ":**

**li $v0, 4**

**la $a0, m3**

**syscall**

**li $v0, 1**

**addi $a0, $a1, 0**

**syscall**

**# In " and "**

**li $v0, 4**

**la $a0, m4**

**syscall**

**li $v0, 1**

**addi $a0, $a2, 0**

**syscall**

**# In " is: "**

**li $v0, 4**

**la $a0, m5**

**syscall**

**li $v0, 1**

**addi $a0, $a3, 0**

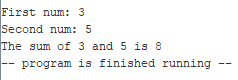
**syscall**

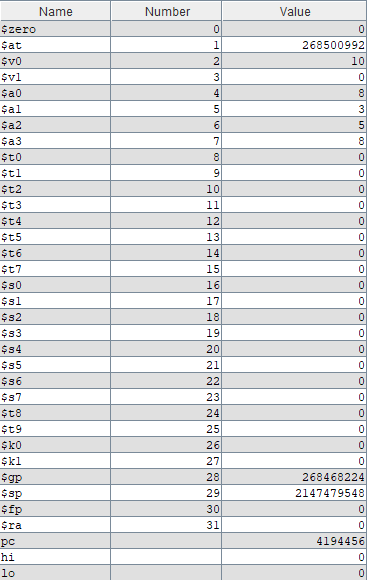
**# Ket thuc chuong trinh:**

**li $v0, 10**

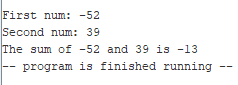
**syscall**

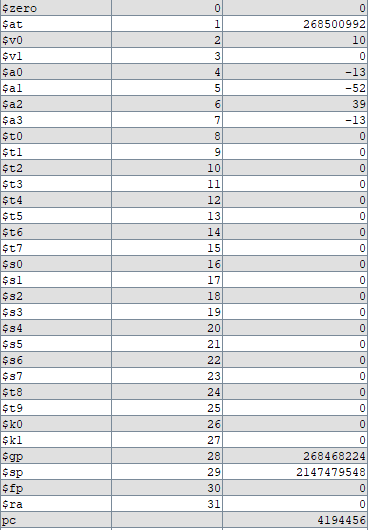
* Kiểm tra kết quả:
* Số đầu = 3, số hai = 5:





* Số đầu = -59, số hai = 39:





* Giải thích chương trình: Để có được output như format trong đề bài, ta thực hiện bằng cách nạp đầu ra theo thứ tự : m1 → m2 → m3 → giá trị $a1 → m4 → Giá trị $a2 → m5 → tổng của 2 giá trị trên là giá trị $a3. Bằng việc dùng print string, print number, read number lần lượt đan xen theo thứ tự như trên để có đầu ra mong muốn.
* Kết luận: Chương trình đã chạy đúng như yêu cầu đề bài.

Assignment 3:

* Code:

**#Laboratory Exercise 5, Home Assignment 2**

**.data**

**m1: .asciiz "Truoc khi copy, x se la: \n"**

**m2: .asciiz "Sau khi copy, x se la: "**

**x: .space 32 # destination string x, empty**

**y: .asciiz "Hello" # source string y**

**.text**

**la $a0, x**

**la $a1, y**

**li $v0, 4**

**la $a0, m1**

**syscall**

**li $v0, 4**

**la $a0, x**

**syscall**

**strcpy:**

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

**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:**

**# In ra man hinh**

**li $v0, 4**

**la $a0, m2**

**syscall**

**li $v0, 4**

**la $a0, x**

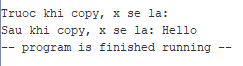
**syscall**

**# Ket thuc chuong trinh:**

**li $v0, 10**

**syscall**

* Giải thích chương trình:
* Đầu tiên, ta sẽ có không gian lưu trữ cho xâu x và lưu xâu cho biến y. Tiếp đến hàm main, ta sẽ lưu địa chỉ của các biến trên vào từng thanh ghi $a0, $a1 tương ứng. Đến hàm strcpy, ta sẽ khởi tạo chỉ số (index) của ký tự trong xâu, bắt đầu từ 0.
* Bắt đầu L1: hàm add thứ nhất (add $t1, $s0, $a1) sẽ gắn giá trị cho thanh ghi $t1 bằng địa chỉ của ký tự trong string y (hay là y[i]), với $a1 sẽ là mặc định là y[0], hay ký tự đầu tiên của chuỗi. Khi muốn có địa chỉ của y[i], ta sẽ có bằng cách cộng giá trị của I vào địa chỉ của y[0].
* Tiếp đó, ta lưu giá trị của y[i] bằng cách dùng hàm lb, lưu giá trị của y[i] vào thanh ghi $t2, với thanh ghi $t1 sẽ là thanh ghi lưu địa chỉ gốc, thanh ghi $t2 sẽ là thanh ghi được nạp vào. Số 0 là hằng số nguyên, cộng vào giá trị của thanh ghi $t1 để có được địa chỉ nạp vào.
* Tiếp đến, lệnh add thứ 2 (add $t3, $s0, $a0) sẽ tương tự với lệnh add thứ 1, dùng để gán địa chỉ của x[i].
* Lệnh sb sẽ giúp ta lưu trữ giá trị của thanh ghi $t2 vào thanh ghi $t3, hay nạp vào x[i].
* Hàm beq để xác định nếu giá trị $t2 bằng 0 (tức đã lấy hết ký tự trong chuỗi ban đầu) thì sẽ kết thúc chương trình strcpy, nếu không ta sẽ tiếp tục chạy các lệnh sau. (Trong hệ mã ASCII, ký tự kết thúc chuỗi ‘\0’ được biểu diễn bằng giá trị 0)
* Nop có tác dụng trễ thời gian, tránh sự ảnh hưởng của Delayed branching khiến cho vòng lặp bị lặp vô hạn lần trong một số trường hợp.
* Lệnh addi sẽ là lệnh tăng giá trị của i lên 1, hay tăng chỉ số của ký tự trong chuỗi x lên 1 đơn vị, để tiếp tục việc lấy ký tự tiếp theo trong chuỗi x.
* j L1 thực hiện việc vòng lặp L1. Cuối cùng, end\_of\_strcpy sẽ là nơi in trên màn hình ký tự x sau khi được copy từ y và kết thúc chương trình.
* Kết quả thu được:





* Như vậy, chương trình trên đã chạy đúng.

Assignment 4:

* Code:

**#Laboratory Exercise 5, Home Assignment 3**

**.data**

**string: .space 50**

**Message1: .asciiz "Nhap xau: \n"**

**Message2: .asciiz "Do dai xau la: "**

**.text**

**main:**

**get\_string:**

**# Nhap sau:**

**li $v0, 54**

**la $a0, Message1**

**la $a1, string**

**la $a2, 50**

**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 $s5, 1 # gan $s5 = 1**

**sub $t0, $t0, $s5 # $t0 = $t0 - 1 de lay ra do dai cua chuoi vua nhap**

**# In output:**

**li $v0, 56**

**la $a0, Message2**

**move $a1, $t0**

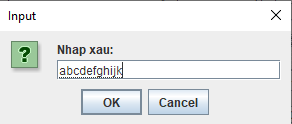
**syscall**

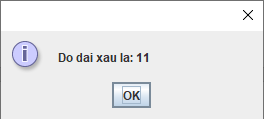
**# Ket thuc chuong trinh:**

**li $v0, 10**

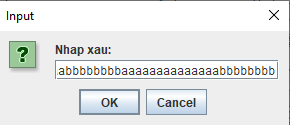
**syscall**

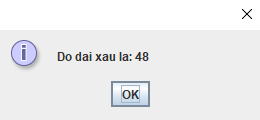
* Kiểm thử chương trình:
* TH1: Số ký tự trong khoảng space:





* TH2: Số ký tự ngoài khoảng space:





* Giải thích: Do đã đặt điều kiện không gian lưu trữ ở đầu là string: .space 50 nên ta không thể nhập quá không gian này. Độ dài xâu là 48 bởi vì máy phải dành không gian lưu trữ cho ký tự ‘\0’ khi kết thúc chuỗi và ký tự ‘\n’ khi ta bấm vào phím ENTER để kết thúc việc nhập đầu vào, vì thế chương trình bị mất đi 2 vị trí lưu trữ.
* Chương trình trên đã chạy đúng.

Assignment 5:

* Code:

**.data**

**string: .space 21 # Toi da 21 ky tu boi vi phai tinh them ca ki tu**

**# ket thuc chuoi ('\0') o cuoi.**

**reverse: .space 21**

**Mssg: .asciiz "Nhap chuoi: "**

**Input: .asciiz "\nChuoi ban vua nhap la: "**

**Output: .asciiz "\nChuoi dao nguoc la: "**

**Error: .asciiz "Chuoi khong co phan tu nao!"**

**.text**

**# Message:**

**li $v0, 4**

**la $a0, Mssg**

**syscall**

**# Lay dia chi cua x[0] va y[0]:**

**la $a0, string**

**la $a1, reverse**

**# Khoi tao cac gia tri:**

**li $s0, 0 # i = 0**

**li $s1, 0 # j = 0**

**# Nhap chuoi:**

**CheckLength:**

**# Neu do dai chuoi i = 20 thi dung vong lap va in ket qua**

**beq $s0, 20, Sub**

**In:**

**# Nhap ki tu:**

**li $v0, 12**

**syscall**

**CheckEnter:**

**# Neu user nhap enter thi dung vong lap**

**beq $v0, 10, CheckEmpty**

**load:**

**add $t1, $s0, $a0 # $t1 = i + x[0]**

**sb $v0, 0($t1) # x[i] = $v0 = ki tu vua nhap**

**nop**

**addi $s0, $s0, 1 # i += 1**

**j CheckLength**

**nop**

**CheckEmpty:**

**# Kiem tra neu chuoi la trong (empty)**

**bgtz $s0, Sub # Neu i > 0 thi chuoi khong trong**

**li $v0, 55 # Hien dialog thong bao chuoi trong**

**la $a0, Error**

**la $a1, 0**

**syscall**

**j out**

**Sub:**

**sub $s0, $s0, 1 # i -= 1 de lay gia tri chu cai hien tai,**

**# tuc la chu cai o cuoi string.**

**makeReverse:**

**sub $s2, $s0, $s1 # Dat $s2 = t = i - j**

**bltz $s2, out # Neu $s2 = t < 0 thi out**

**add $t1, $s2, $a0 # $t1 = t + x[0]**

**lb $t2, 0($t1) # $t2 = x[t], t hien tai dang co gia tri string.size() - 1**

**add $t3, $s1, $a1 # $t3 = j + y[0]**

**sb $t2, 0($t3) # y[j] = $t2 = x[t]**

**nop**

**addi $s1, $s1, 1 # j += 1**

**j makeReverse**

**out: # In ra output:**

**# Chuoi vua nhap:**

**li $v0, 4**

**la $a0, Input**

**syscall**

**li $v0, 4**

**la $a0, string**

**syscall**

**# Chuoi dao nguoc:**

**li $v0, 4**

**la $a0, Output**

**syscall**

**li $v0, 4**

**la $a0, reverse**

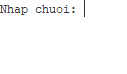
**syscall**

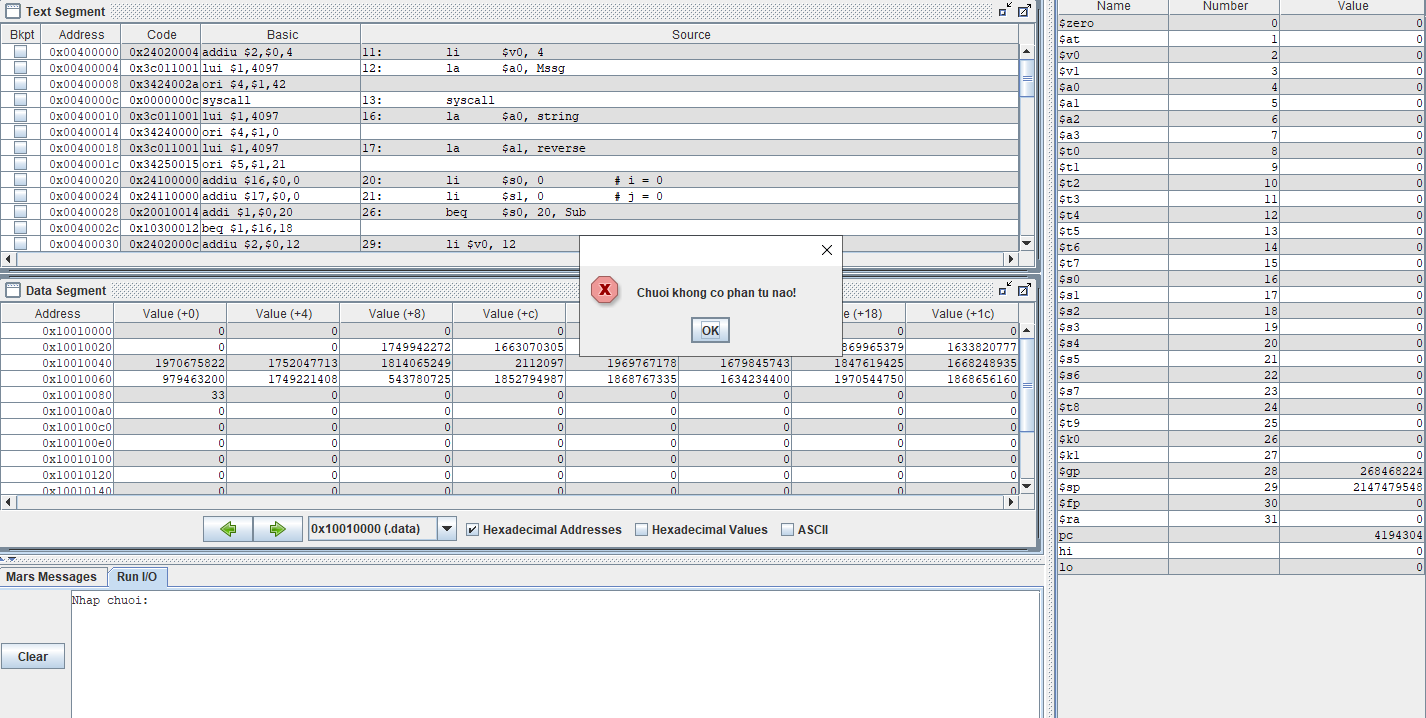
**# Ket thuc:**

**li $v0, 10**

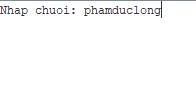
**syscall**

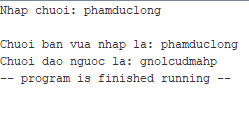
* Ta xét 03 trường hợp:
* TH1: string rỗng:



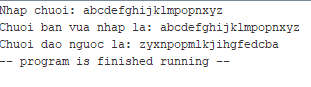


* TH2: string có độ dài nhỏ hơn 20:





* TH3: string có độ dài quá 20:



* Chương trình trên đã chạy đúng như mong muốn.
* **Kết luận**:
* Khác biệt chính giữa chuỗi trong C và Java là ở cách triển khai và xử lý. Trong C, chuỗi được biểu diễn dưới dạng mảng các ký tự (Ví dụ char s[100]) và kết thúc bằng ký tự null ('\0'). Vì thế, việc xử lý chuỗi trong C yêu cầu quản lý bộ nhớ thủ công và sử dụng các hàm rõ ràng cho các phép toán như nối chuỗi (strcat), so sánh (strcmp) và sao chép (strcpy),… trong thư viện <string.h>. Trong khi đó, trong Java, chuỗi có thể được biểu diễn dưới dạng đối tượng của lớp String. Java còn cung cấp nhiều phương thức tích hợp sẵn để xử lý chuỗi, như nối chuỗi bằng toán tử +, so sánh bằng phương thức equals(), và sao chép bằng phương thức substring(),... Ngoài ra, Java cũng tự động quản lý bộ nhớ cho chuỗi.
* Trong C, kích thước của char thường là 1 byte. Do đó, với 8 byte bộ nhớ, ta có thể lưu trữ 8 ký tự.
* Trong Java, kích thước của char luôn là 2 byte. Do đó, với 8 byte bộ nhớ, ta có thể lưu trữ 4 ký tự (Java sử dụng mã hóa UTF-16, trong đó mỗi ký tự được biểu diễn bằng 2 byte).