

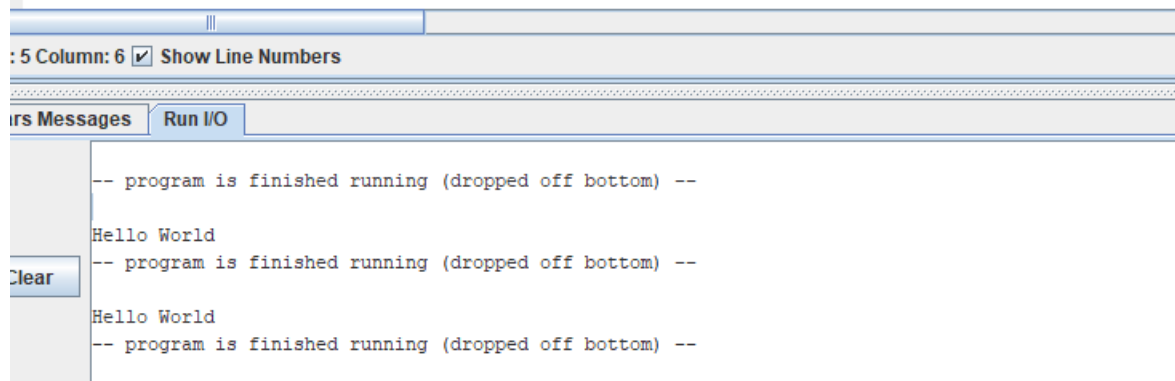
# **BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH (IT3280) TUẦN 5**

*Họ và tên: Trịnh Minh Đức*

*MSSV: 20225813*

Assignment 1:

```
#Laboratory Exercise 5, Home Assignment 1
.data
test: .asciiz "Hello World"
.text
li $v0, 4
la $a0, test
syscall
```



```
5 Column: 6 ☒ Show Line Numbers
rs Messages Run I/O
-- program is finished running (dropped off bottom) --
Hello World
-- program is finished running (dropped off bottom) --
Hello World
-- program is finished running (dropped off bottom) --
```

Kết quả thu được:

- Data Segment:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	l l e H	o W o	\0 d l r	\0 \0 \0 \0
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

-->Ta thấy chuỗi “Hello World” được lưu vào các ô nhớ theo thứ tự ngược lại ứng với mỗi Value.

→ 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. 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”.

Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x6c6c6548	0x6f57206f	0x00646c72
0x10010020	0x00000000	0x00000000	0x00000000

- 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

- Kết luận:

- Chuỗi được lưu trữ như trên, mỗi Value sẽ lưu trữ tối đa 4 ký tự của chuỗi ,bắt đầu từ Value (+0) cho đến khi hết chuỗi.

Assignment 2:

- Code:

**.data**

**a1: .asciiz "So thu nhat: "**

**a2: .asciiz "So thu hai: "**

**a3: .asciiz "The some of "**

**a4: .asciiz " and "**

**a5: .asciiz " is "**

**.text**

**# Nhap so dau tien:**

**li \$v0, 4**

**la \$a0, a1**

**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, a2**

**syscall**

**li \$v0, 5**

**syscall**

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

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

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

**# In ra:**

**# In "The some of ":**

**li \$v0, 4**

**la \$a0, a3**

**syscall**

**li \$v0, 1**

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

**syscall**

**# In " and "**

**li \$v0, 4**

**la \$a0, a4**

**syscall**

**li \$v0, 1**

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

**syscall**

**# In " is: "**

**li \$v0, 4**

**la \$a0, a5**

**syscall**

**li \$v0, 1**

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

**syscall**

**# Ket thuc chuong trinh:**

**li \$v0, 10**

**syscall**

- Kiểm tra kết quả:

- Số đầu = 2, số hai = 3:

```

Số thu nhất: **** user input : 2
Số thu hai: **** user input : 3
The sum of 2 and 3 is 5
-- program is finished running --

```

\$a0	4	0x00000005
\$a1	5	0x00000002
\$a2	6	0x00000003
\$a3	7	0x00000005
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000

- Số đầu = -10, số hai = 3:

```

Số thu nhất: **** user input : -10
Số thu hai: **** user input : 3
The sum of -10 and 3 is -7
-- program is finished running --

```

\$a0	4	-7
\$a1	5	-10
\$a2	6	3
\$a3	7	-7
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0

- Kết luận: Chương trình đã chạy đúng như yêu cầu đề bài.

### Assignment 3:

- Code:



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

**.data**

**a1: .asciiz "Truoc copy, x la: \n"**

**a2: .asciiz "Sau i copy, x 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, a1**

**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, a2**

**syscall**

**li    \$v0, 4**

**la    \$a0, x**

**syscall**

**# Ket thuc chuong trinh:**

**li    \$v0, 10**

**syscall**

- Giải thích chương trình:

- Ta có không gian lưu trữ cho chuỗi x và lưu chuỗi cho biến y. Tiếp đến hàm main, ta sẽ lưu địa chỉ của các biến trên vào từng thanh ghi \$a0, \$a1 tương ứng. Ở hàm strcpy, ta sẽ khởi tạo chỉ số (index) của ký tự trong chuỗi, bắt đầu từ 0.
- Bắt đầu L1: hàm add thứ nhất (add \$t1, \$s0, \$a1) sẽ gán 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],. 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 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
- j L1 thực hiện việc vòng lặp L1. Cuối cùng, end\_of\_memcpy 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:

```
So thu nhât: **** user input : -10
So thu hai: **** user input : 3
The sum of -10 and 3 is -7
-- program is finished running --

Truoc copy, x la:
Sau i copy, x la : Hello
-- program is finished running --
```

268500992	o u r I	o c c	, y p	a l x	\0 \n :
268501024	a l x	\0 :	l l e H	\0 \0 \0 o	\0 \0 \0 \0

➔ 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 sâu:**

**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 lấy độ dài  
của chuỗi vừa nhập**

**# In output:**

**li \$v0, 56**

**la \$a0, Message2**

**move \$a1, \$t0**

**syscall**

**# Ket thuc chuong trinh:**

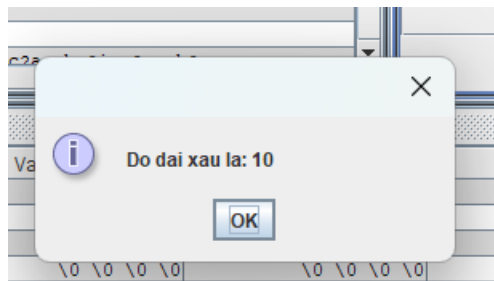
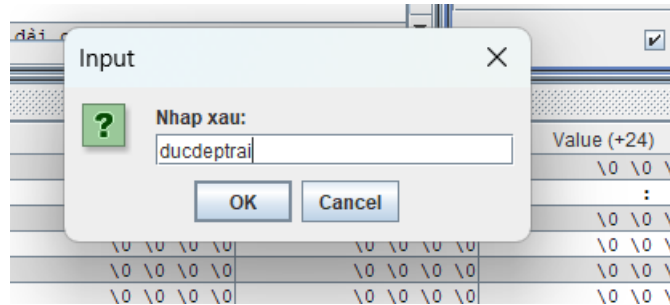
**li \$v0, 10**

**syscall**

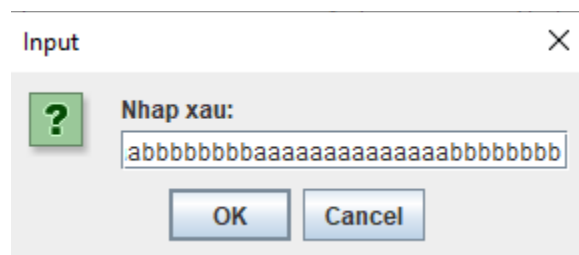


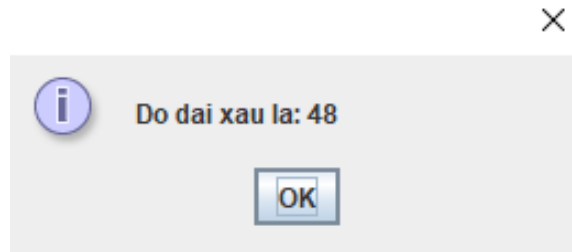
- Kiểm thử chương trình:

- TH1: Số ký tự trong khoảng space:



- TH2: Số ký tự ngoài khoảng space:





- 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
x:  .space 21    # Toi da 21 ky tu
                        # vi co ket thuc "\0"cuoi.
y:  .space 21
Message:      .ascii "Nhap chuoi: "
Input:        .ascii "\nChuoi ban vua nhap
la: "
Output:       .ascii "\nChuoi dao nguoc la: "
Error:        .ascii "Chuoi khong co phan
tu nao!"
```

.text

# Message:

li \$v0, 4

la \$a0, Message

syscall

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

la \$a0, x

la \$a1, y

# 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 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 rong  
bgtz \$s0, Sub            # Neu i > 0 thi  
chuoi khong rong  
li \$v0, 55            # Hien dialog thong bao  
chuoi rong  
la \$a0, Error  
la \$a1, 0

```
syscall
j    out
```

Sub:

sub \$s0, \$s0, 1 # i -= 1 de lay gia tri  
chu cai hien , 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, x  
syscall
```

```
# Chuoi dao nguoc:  
li $v0, 4  
la $a0, Output  
syscall
```

```
li $v0, 4  
la $a0, y  
syscall
```

```
# Ket thuc:  
li $v0, 10  
syscall
```

các trường hợp xảy ra

Nhap chuo:

Chuoi ban vua nhap la:

Chuoi dao nguoc la:

```
-- program is finished running --
```

The screenshot shows the WinDbg interface with the following components:

- Assembly View:** Displays assembly code for a function. The code includes instructions like `syscall`, `la $a0, x`, `la $a1, y`, `li $s0, 0 # i = 0`, `li $s1, 0 # j = 0`, `beq $s0, 20, Sub`, `li $v0, 12`, `syscall`, `beq $v0, 10, CheckEmpty`, and `add $t1, $a0, $a0 # $t1 = i + x[0]`.
- Registers View:** Shows the state of registers, with `$s0` and `$s1` both set to 0.
- Memory Dump:** Displays a memory dump with columns for values at offsets +8, +12, +16, +20, and +24. The dump shows a string of characters: `a l p a i o u n o h k o c g a h p u t r`.
- Dialog Box:** A modal dialog box is overlaid on the memory dump. It has a red 'X' icon and the text `Chuoi không có phần tử nào!` (String has no elements!). There is an `OK` button.
- Bottom Panel:** Shows the current address `0x10010000 (.data)` and checkboxes for `Hexadecimal Addresses`, `Hexadecimal Values`, and `ASCII`.

## chuỗi rỗng

```
Nhap chuoai: ducdeptraiquaaa
```

```
Chuoi ban vua nhap la: ducdeptraiquaaa
```

```
Chuoi dao nguoc la: aaauqiartpedcud
```

```
-- program is finished running --
```

```
Nhap chuoai: dfejfnewfnkndfnkfdm
```

```
Chuoi ban vua nhap la: dfejfnewfnkndfnkfdm
```

```
Chuoi dao nguoc la: mdfknfdnknfwenfjefd
```

---

chuỗi khi nhập dưới 20 kí tự và khi nhập  
thừa 20 kí tự . khi nhập quá 20 kí tự nó tự  
động kết thúc

->Kết luận

- Trong C, chuỗi được biểu diễn dưới dạng mảng các ký tự 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 lưu trữ 8 ký tự.
- Trong Java, kích thước của char luôn là 2 byte. Do đó, với 8 byte bộ nhớ lưu trữ 4 ký tự