

BÁO CÁO THÚC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 5

Họ và tên: Phan Khánh Vũ

MSSV: 20235880

1. Assignment 1:

Nhập chương trình:

The screenshot shows a text editor window titled "Lab05_1.asm*". The "Edit" tab is selected at the top. The code in the editor is as follows:

```
1 # Laboratory Exercise 5, Home Assignment 1
2 .data
3 test: .asciz "Hello World"
4 .text
5 li a7, 4
6 la a0, test
7 ecall
```

Giải thích code:

Các bước thực hiện của chương trình:

- ✓ Câu lệnh test: .asciz “Hello World” để khởi tạo giá trị cho chuỗi kí tự “Hello World”.
- ✓ Câu lệnh li a7, 4 là gán giá trị 4 cho a7 là số hiệu dịch vụ in chuỗi kí tự
- ✓ Câu lệnh la a0, test để lấy địa chỉ của chuỗi kí tự.
- ✓ Câu lệnh ecall là lời gọi đến hệ thống, trao quyền cho hệ điều hành thực hiện chương trình.

Quan sát kết quả trên cửa sổ Run I/O:

The screenshot shows the "Run I/O" window. The "Messages" tab is selected at the top. The output window displays the following text:

```
Hello World
-- program is finished running (dropped off bottom) --
```

Below the output window is a "Clear" button.

Quan sát cửa sổ Data Segment:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x6c6c6548	0x6f57206f	0x00646c72	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Dữ liệu được lưu ở data segment theo từng ô 4 byte và được lưu theo thứ tự từ phải sang trái. Ở đây chia “Hello World” thành 3 cụm 4 byte và được lưu lần lượt “l l e H”, “o W _ o”, “\0 d l r” (kí tự ‘_’ để mô tả thê dấu cách, \0 là kí tự kết thúc chuỗi kí tự).

Phân tích truyền tham số trong RISC-V

Trong RISC-V, tham số được truyền vào các thanh ghi a0, a1, a2, ...a7 theo thứ tự:

- ✓ a0: Đối số đầu tiên (thường là địa chỉ chuỗi format của printf).
- ✓ a1, a2, a3, ...: Các tham số tiếp theo (có thể là số nguyên, địa chỉ chuỗi, số thực, v.v.).

Nhận xét:

Đoạn code là ví dụ việc giao tiếp với hệ điều hành thông qua syscall, cụ thể là in chuỗi ký tự ra màn hình. Bằng cách sử dụng các thanh ghi để truyền tham số và mã syscall, chương trình đã tận dụng hiệu quả cơ chế xử lý của hệ thống RISC-V.

2. Assignment 2:

Format: “The sum of (s0) and (s1) is (result)”

Nhập chương trình:

<pre> 1 .data 2 msg1: .asciz "The sum of " 3 .asciz " and " 4 .asciz " is " 5 .text 6 # Gán giá trị số vào thanh ghi 7 li s1, 10 # s1 = 10 8 li s2, 20 # s2 = 20 9 add s0, s1, s2 # s0 = s1 + s2 10 # In chuỗi "The sum of " 11 li a7, 4 12 la a0, msg1 13 ecall 14 # In số s1 15 li a7, 1 16 mv a0, s1 17 ecall 18 # In chuỗi " and " 19 li a7, 4 20 la a0, msg2 21 ecall 22 # In số s2 23 li a7, 1 24 mv a0, s2 25 ecall 26 # In chuỗi " is " 27 li a7, 4 28 la a0, msg3 29 ecall 30 # In kết quả tổng 31 li a7, 1 32 mv a0, s0 33 ecall 34 # Kết thúc chương trình 35 li a7, 10 36 ecall 37 38 </pre>	<pre> 12 la a0, msg1 13 ecall 14 # In số s1 15 li a7, 1 16 mv a0, s1 17 ecall 18 # In chuỗi " and " 19 li a7, 4 20 la a0, msg2 21 ecall 22 # In số s2 23 li a7, 1 24 mv a0, s2 25 ecall 26 # In chuỗi " is " 27 li a7, 4 28 la a0, msg3 29 ecall 30 # In kết quả tổng 31 li a7, 1 32 mv a0, s0 33 ecall 34 # Kết thúc chương trình 35 li a7, 10 36 ecall 37 38 </pre>
---	--

Giải thích chương trình:

✓ **Các giá trị khởi tạo:**

+ Khởi tạo giá trị cho s1 = 10, s2 = 20;

✓ **Các bước thực hiện chương trình:**

+ Cụm .data để gán các chuỗi kí tự vào các msg

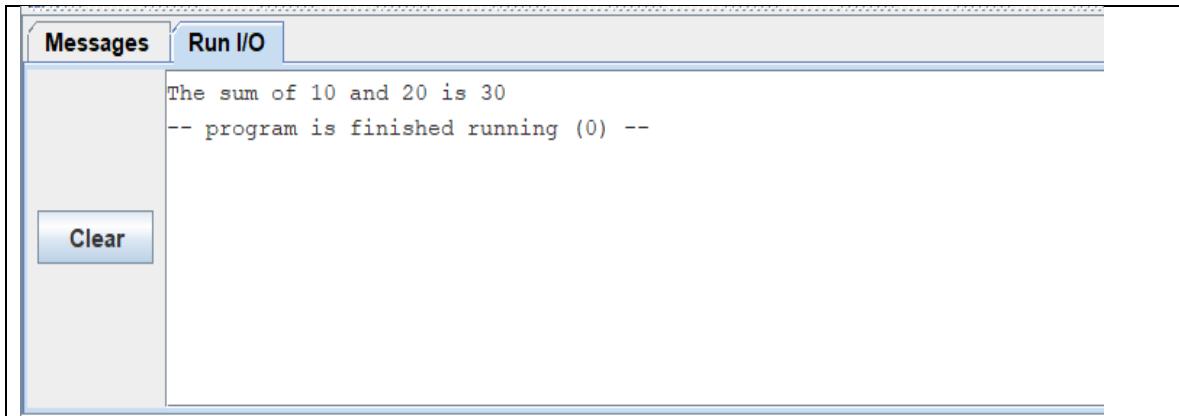
+ Cụm gán giá trị vào thanh ghi để gán giá trị cho các thanh ghi s1, s2 và tính tổng 2 thanh ghi

+ Các cụm in chuỗi kí tự sử dụng chung cú pháp li a7, 4 để gán giá trị cho a7 = 4 tương ứng với dịch vụ in chuỗi kí tự. Sau đó lấy địa chỉ của chuỗi kí tự thông qua thanh ghi a0 và thực hiện ecall để trao quyền cho hệ thống làm việc

+ Các cụm in số và tổng sử dụng chung cú pháp li a7, 1 để in giá trị thuộc kiểu int. Sau đó sao chép giá trị các thanh ghi vào thanh ghi a0 để thực hiện ecall in ra màn hình.

+ Câu lệnh li a7, 10 để thực hiện thoát khỏi hệ thống

✓ **Kết quả của chương trình:**



+ Dữ liệu lưu ở bảng data segment :

Data Segment									
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+22)	Value (+26)
0x10010000	e h T	m u s	\0 f o	d n a	i \0) \0	s	\0 \0 \0 \0	\0 \0 \0 \0
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100e0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010100	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

3. Assignment 3:

Nhập chương trình:

```

1 .data
2         x: .space 32 # Chuỗi đích x, khởi tạo là buffer rỗng
3 y: .asciz "Hello" # Chuỗi nguồn y
4 .text
5 strcpy:
6     add s0, zero, zero # s0 = i = 0
7     la a0, x # load addr x to a0
8     la a1, y # y to a0
9 L1:
10    add t1, s0, a1
11    lb t2, 0(t1)
12    add t3, s0, a0
13    sb t2, 0(t3)
14    beq t2, zero, end_of_strcpy
15    addi s0, s0, 1
16    j L1
17 end_of_strcpy:
18

```

✓ Các bước thực hiện chương trình:

- + Cụm .data để khởi tạo chuỗi kí tự và khởi tạo buffer rỗng để lưu chuỗi kí tự sao chép
- + Cụm các câu lệnh khởi tạo s0 = 0, la a0, 0 để lấy địa chỉ của x vào a, la a1, y để gán địa chỉ của y vào a1
- + Câu lệnh add t1, s0, a1 để lấy địa chỉ byte tiếp theo của y lưu vào t1

- + Câu lệnh lb t2, 0(t1) để lấy dữ liệu từ byte có địa chỉ t1
 - + Câu lệnh add t3, s0, a0 để lấy địa chỉ byte tiếp theo của x
 - + Câu lệnh sb t2, 0(t3) để đưa giá trị của t2 vào byte t3
 - + Câu lệnh beq t2, zero, end_of_strcpy để so sánh nếu t2 = 0 thì nhảy đến end_of_strcpy để kết thúc vòng lặp
 - + Câu lệnh addi s0, s0, 1 để tăng giá trị biến đếm
 - + Câu lệnh j L1 để nhảy đến thẻ L1 tiếp tục vòng lặp
- ⇒ Sử dụng vòng lặp để đưa từng byte dữ liệu của y vào x đến khi không còn giá trị nào của y để đẩy vào x thì kết thúc vòng lặp.

✓ Kết quả của chương trình

Theo lí thuyết: Gán giá trị cho y = "Hello" thì dữ liệu đầu ra của x = "Hello" và được lưu ở cửa sổ Data Segment dưới dạng "l1 e H", "\0 \0 o".

Chạy từng bước quan sát sự thay đổi bộ nhớ nhằm kiểm tra xem chương trình có copy paste không:

Name	Number	Value
zero	0	0
r1	1	0
sp	2	2147475540
fp	3	268468224
tp	4	0
t0	5	0
t1	6	0
t2	7	0
s0	8	0
s1	9	0
a0	10	268500992
a1	11	268501000
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
a2	18	0
a3	19	0
a4	20	0
a5	21	0
a6	22	0
a7	23	0
a8	24	0
a9	25	0
a10	26	0
a11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0
pc		4194316

Name	Number	Value
zero	0	0
r1	1	0
sp	2	214747958
fp	3	268469224
tp	4	0
t0	5	0
t1	6	0
t2	7	0
s0	8	0
s1	9	0
a0	10	268500992
a1	11	268501024
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
a2	18	0
a3	19	0
a4	20	0
a5	21	0
a6	22	0
a7	23	0
a8	24	0
a9	25	0
a10	26	0
a11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0
pc		4194320

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400004	0x0fc10597;auipc x11,x10,0	7: la a0, x	# Địa chỉ chuỗi nguồn y
	0x00400008	0xfc0f10597;auipc x11,e4528	8: la a1, y	# Địa chỉ chuỗi nguồn y
	0x0040000c	0x01e58593;addi x11,x11,24		
	0x00400010	0x01000ef;jal x11,28	9: jal strcpy	# Gọi hàm strcpy
	0x00400014	0x00400893;addi x17,x0,4	11: li a7, 4	# Syscall in chuỗi
	0x00400018	0xfc0f10597;auipc x10,e4528	12: la a0, x	# In chuỗi x sau khi ma..
	0x0040001c	0x0fe50513;addi x10,x10,-24		
	0x00400020	0x00000073;ecall	13: ecall	
	0x00400024	0x00000073;ecall	15: li a7, 10	# Thoát chương trình
	0x00400028	0x00000073;ecall	16: ecall	
	0x0040002c	0x00000043;add x8,x0,x0	19: add \$0, zero, zero	# \$0 = i = 0

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	1819043144	111	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
...

Buttons: **0x10010000 (.data)** Hexadecimal Addresses Hexadecimal Values ASCII

Registers Floating Point Control and Status

Name	Number	Value
zero	0	0
ra	1	4194324
sp	2	2147479548
gp	3	268460224
tp	4	0
t0	5	0
t1	6	0
t2	7	0
t3	8	0
t4	9	0
t5	10	0
t6	11	268501024
t7	12	0
t8	13	0
t9	14	0
t10	15	0
t11	16	0
t12	17	0
t13	18	0
t14	19	0
t15	20	0
t16	21	0
t17	22	0
t18	23	0
t19	24	0
t20	25	0
t21	26	0
t22	27	0
t23	28	0
t24	29	0
t25	30	0
t26	31	0
pc		4194324

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400004	0x0fc10597;addi x11,x10,0	7: la a1, y	# Địa chỉ chuỗi nguồn y
	0x00400008	0xfc0f10597;auipc x11,e4528	8: la a1, y	# Địa chỉ chuỗi nguồn y
	0x0040000c	0x01e58593;addi x11,x11,24		
	0x00400010	0x01000ef;jal x11,28	9: jal strcpy	# Gọi hàm strcpy
	0x00400014	0x00400893;addi x17,x0,4	11: li a7, 4	# Syscall in chuỗi
	0x00400018	0xfc0f10597;auipc x10,e4528	12: la a0, x	# In chuỗi x sau khi ma..
	0x0040001c	0x0fe50513;addi x10,x10,-24		
	0x00400020	0x00000073;ecall	13: ecall	
	0x00400024	0x00000073;ecall	15: li a7, 10	# Thoát chương trình
	0x00400028	0x00000073;ecall	16: ecall	
	0x0040002c	0x00000043;add x8,x0,x0	19: add \$0, zero, zero	# \$0 = i = 0
	0x00400030	0x000b4033;add x6,x8,x11	21: add t1, \$0, a1	# Lấy địa chỉ y[i]

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	1819043144	111	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
...

Buttons: **0x10010000 (.data)** Hexadecimal Addresses Hexadecimal Values ASCII

Registers Floating Point Control and Status

Name	Number	Value
zero	0	0
ra	1	4194324
sp	2	2147479548
gp	3	268460224
tp	4	0
t0	5	0
t1	6	0
t2	7	0
t3	8	0
t4	9	0
t5	10	268501024
t6	11	268501024
t7	12	0
t8	13	0
t9	14	0
t10	15	0
t11	16	0
t12	17	0
t13	18	0
t14	19	0
t15	20	0
t16	21	0
t17	22	0
t18	23	0
t19	24	0
t20	25	0
t21	26	0
t22	27	0
t23	28	0
t24	29	0
t25	30	0
t26	31	0
pc		4194324

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400004	0x0fc10597;addi x11,x10,0	7: la a1, y	# Địa chỉ chuỗi nguồn y
	0x00400008	0xfc0f10597;auipc x11,e4528	8: la a1, y	# Địa chỉ chuỗi nguồn y
	0x0040000c	0x01e58593;addi x11,x11,24		
	0x00400010	0x01000ef;jal x11,28	9: jal strcpy	# Gọi hàm strcpy
	0x00400014	0x00400893;addi x17,x0,4	11: li a7, 4	# Syscall in chuỗi
	0x00400018	0xfc0f10597;auipc x10,e4528	12: la a0, x	# In chuỗi x sau khi ma..
	0x0040001c	0x0fe50513;addi x10,x10,-24		
	0x00400020	0x00000073;ecall	13: ecall	
	0x00400024	0x00000073;ecall	15: li a7, 10	# Thoát chương trình
	0x00400028	0x00000073;ecall	16: ecall	
	0x0040002c	0x00000043;add x8,x0,x0	19: add \$0, zero, zero	# \$0 = i = 0
	0x00400030	0x000b4033;add x6,x8,x11	21: add t1, \$0, a1	# Lấy địa chỉ y[i]
	0x00400034	0x00003038;lb x7,0(\$x6)	22: lb t2, 0(\$t1)	# Đọc ký tự y[i]

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	1819043144	111	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
...

Buttons: **0x10010000 (.data)** Hexadecimal Addresses Hexadecimal Values ASCII

Registers Floating Point Control and Status

Name	Number	Value
zero	0	0
ra	1	4194324
sp	2	2147479548
gp	3	268460224
tp	4	0
t0	5	0
t1	6	268501024
t2	7	0
t3	8	0
t4	9	0
t5	10	268501024
t6	11	268501024
t7	12	0
t8	13	0
t9	14	0
t10	15	0
t11	16	0
t12	17	0
t13	18	0
t14	19	0
t15	20	0
t16	21	0
t17	22	0
t18	23	0
t19	24	0
t20	25	0
t21	26	0
t22	27	0
t23	28	0
t24	29	0
t25	30	0
t26	31	0
pc		4194324

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400020	0x00000073_ecall		
	0x00400024	0x0000093.addi x17,x0,10	13: ecall	
	0x00400028	0x0000073_ecall	15: li a7, 10 # Thoát chương trình	
	0x0040002c	0x0000433.add x8,x0,x0	16: ecall	
	0x00400030	0x00b40333.add x6,x8,x11	19: add s0, zero, zero # s0 = i = 0	
	0x00400034	0x00030383.lb x7,0,(x6)	21: add t1, s0, a1 # Lấy địa chỉ y[i]	
	0x00400038	0x00a4e033.add x28,x8,x10	22: lb t2, 0(t1) # Đọc ký tự y[i]	
	0x0040003c	0x07e00233.beq x7,0,(x28)	23: add t3, s0, a0 # Lấy địa chỉ x[i]	
	0x00400040	0x00038633.beq x7,x0,12	24: sb t2, 0(t3) # Gán x[i] = y[i]	
	0x00400044	0x0140413.addi x8,x0,1	25: beg t2, zero, end of strcpy # Dùng mèu g...	
	0x00400048	0xfeffff6f.jal x0,-24	26: addi s0, s0, 1 # i++	
	0x0040004c	0x00008067.jalr x0,x1,0	27: j L1	
			28: jr ra # Quay về ngay sau jalr	
			30:	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	1819043144	111	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0

0x10010000 .(data) Hexadecimal Addresses Decimal Values ASCII

Registers Floating Point Control and Status

Name	Number	Value
zero	0	0
ra	1	4194324
sp	2	214749548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	268501024
t2	7	72
s0	8	0
s1	9	0
a0	10	268500992
a1	11	268501024
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0
pc		4194360

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400020	0x00000073_ecall		
	0x00400024	0x0000093.addi x17,x0,10	13: ecall	
	0x00400028	0x0000073_ecall	15: li a7, 10 # Thoát chương trình	
	0x0040002c	0x0000433.add x8,x0,x0	16: ecall	
	0x00400030	0x00b40333.add x6,x8,x11	19: add s0, zero, zero # s0 = i = 0	
	0x00400034	0x00030383.lb x7,0,(x6)	21: add t1, s0, a1 # Lấy địa chỉ y[i]	
	0x00400038	0x00a4e033.add x28,x8,x10	22: lb t2, 0(t1) # Đọc ký tự y[i]	
	0x0040003c	0x07e00233.beq x7,0,(x28)	23: add t3, s0, a0 # Lấy địa chỉ x[i]	
	0x00400040	0x00038633.beq x7,x0,12	24: sb t2, 0(t3) # Gán x[i] = y[i]	
	0x00400044	0x0140413.addi x8,x0,1	25: beg t2, zero, end of strcpy # Dùng mèu g...	
	0x00400048	0xfeffff6f.jal x0,-24	26: addi s0, s0, 1 # i++	
	0x0040004c	0x00008067.jalr x0,x1,0	27: j L1	
			28: jr ra # Quay về ngay sau jalr	
			30:	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	1819043144	111	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0

0x10010000 .(data) Hexadecimal Addresses Decimal Values ASCII

Registers Floating Point Control and Status

Name	Number	Value
zero	0	0
ra	1	4194324
sp	2	214749548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	268501024
t2	7	72
s0	8	0
s1	9	0
a0	10	268500992
a1	11	268501024
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	268500992
t4	29	0
t5	30	0
t6	31	0
pc		4194364

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400020	0x00000073_ecall		
	0x00400024	0x0000093.addi x17,x0,10	13: ecall	
	0x00400028	0x0000073_ecall	15: li a7, 10 # Thoát chương trình	
	0x0040002c	0x0000433.add x8,x0,x0	16: ecall	
	0x00400030	0x00b40333.add x6,x8,x11	19: add s0, zero, zero # s0 = i = 0	
	0x00400034	0x00030383.lb x7,0,(x6)	21: add t1, s0, a1 # Lấy địa chỉ y[i]	
	0x00400038	0x00a4e033.add x28,x8,x10	22: lb t2, 0(t1) # Đọc ký tự y[i]	
	0x0040003c	0x07e00233.beq x7,0,(x28)	23: add t3, s0, a0 # Lấy địa chỉ x[i]	
	0x00400040	0x00038633.beq x7,x0,12	24: sb t2, 0(t3) # Gán x[i] = y[i]	
	0x00400044	0x0140413.addi x8,x0,1	25: beg t2, zero, end of strcpy # Dùng mèu g...	
	0x00400048	0xfeffff6f.jal x0,-24	26: addi s0, s0, 1 # i++	
	0x0040004c	0x00008067.jalr x0,x1,0	27: j L1	
			28: jr ra # Quay về ngay sau jalr	
			30:	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	72	0	0	0	0	0	0	0
0x10010020	1819043144	111	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0

0x10010000 .(data) Hexadecimal Addresses Decimal Values ASCII

Registers Floating Point Control and Status

Name	Number	Value
zero	0	0
ra	1	4194324
sp	2	214749548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	268501024
t2	7	72
s0	8	0
s1	9	0
a0	10	268500992
a1	11	268501024
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	268500992
t4	29	0
t5	30	0
t6	31	0
pc		4194368

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x04000020	0x00000073_ecall	13: ecall	
	0x04000024	0x0a000933_adci x17,x0,10	15: li a7, 10	# Thoát chương trình
	0x04000028	0x00000073_ecall	16: ecall	
	0x0400002c	0x00000433_ad x8,x0,x0	19: add s0, zero, zero	# s0 = i = 0
	0x04000030	0x0b403333_ad x8,x8,x11	21: add t1, s0, a1	# Lấy địa chỉ y[i]
	0x04000034	0x000303831b_x7,0_(x6)	22: lb t2, 0(t1)	# Duyệt ký tự y[i]
	0x04000038	0x00040e0333_ad x28,x8,x10	23: add t3, s0, a0	# Lấy địa chỉ x[i]
	0x0400003c	0x07a00233_sb x7,0_(x20)	24: sb t2, 0(t3)	# Gán x[i] = y[i]
	0x04000040	0x00038663_beq x7,x0,12	25: beq t2, zero, end of strcpy # Dừng nếu g..	
	0x04000044	0x01404133_adci x8,x0,1	26: addi s0, s0, 1	# i++
	0x04000048	0xfe9ff06f_jal x0,-24	27: j L1	
	0x0400004c	0x000008067_jalr x0,x1,0	30: jr ra	# Quay về ngay sau jalr

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	72	0	0	0	0	0	0	0
0x10010020	1819043144	111	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0

Registers **Floating Point** **Control and Status**

Name	Number	Value
zero	0	0
ra	1	4194324
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	268501024
t2	7	72
s0	8	1
s1	9	0
a0	10	268500992
a1	11	268501024
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	268500992
t4	29	0
t5	30	0
t6	31	0
pc		4194376

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x04000020	0x00000073_ecall	13: ecall	
	0x04000024	0x0a000933_adci x17,x0,10	15: li a7, 10	# Thoát chương trình
	0x04000028	0x00000073_ecall	16: ecall	
	0x0400002c	0x00000433_ad x8,x0,x0	19: add s0, zero, zero	# s0 = i = 0
	0x04000030	0x0b403333_ad x8,x8,x11	21: add t1, s0, a1	# Lấy địa chỉ y[i]
	0x04000034	0x000303831b_x7,0_(x6)	22: lb t2, 0(t1)	# Duyệt ký tự y[i]
	0x04000038	0x00040e0333_ad x28,x8,x10	23: add t3, s0, a0	# Lấy địa chỉ x[i]
	0x0400003c	0x07a00233_sb x7,0_(x20)	24: sb t2, 0(t3)	# Gán x[i] = y[i]
	0x04000040	0x00038663_beq x7,x0,12	25: beq t2, zero, end of strcpy # Dừng nếu g..	
	0x04000044	0x01404133_adci x8,x0,1	26: addi s0, s0, 1	# i++
	0x04000048	0xfe9ff06f_jal x0,-24	27: j L1	
	0x0400004c	0x000008067_jalr x0,x1,0	30: jr ra	# Quay về ngay sau jalr

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	72	0	0	0	0	0	0	0
0x10010020	1819043144	111	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0

Registers **Floating Point** **Control and Status**

Name	Number	Value
zero	0	0
ra	1	4194324
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	268501024
t2	7	72
s0	8	1
s1	9	0
a0	10	268500992
a1	11	268501024
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	268500992
t4	29	0
t5	30	0
t6	31	0
pc		4194352

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x04000014	0x04000893_adci x17,x0,4	11: li a7, 4	# Syscall in chuỗi
	0x04000018	0x0fc10517_auipe x10,64528	12: la a0, x	# In chuỗi x sau khi sa..
	0x0400001c	0xfe850513_adci x10,x10,-24		
	0x04000020	0x00000073_ecall	13: ecall	
	0x04000024	0x0a000933_adci x17,x0,10	15: li a7, 10	# Thoát chương trình
	0x04000028	0x00000073_ecall	16: ecall	
	0x0400002c	0x00000433_ad x8,x0,x0	19: add s0, zero, zero	# s0 = i = 0
	0x04000030	0x0b403333_ad x8,x8,x11	21: add t1, s0, a1	# Lấy địa chỉ y[i]
	0x04000034	0x000303831b_x7,0_(x6)	22: lb t2, 0(t1)	# Duyệt ký tự y[i]
	0x04000038	0x00040e0333_ad x28,x8,x10	23: add t3, s0, a0	# Lấy địa chỉ x[i]
	0x0400003c	0x07a00233_sb x7,0_(x20)	24: sb t2, 0(t3)	# Gán x[i] = y[i]
	0x04000040	0x00038663_beq x7,x0,17	25: beq t2, zero, end of strcpy # Dừng nếu g..	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1819043144	111	0	0	0	0	0	0
0x10010020	1819043144	111	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0

Registers **Floating Point** **Control and Status**

Name	Number	Value
zero	0	0
ra	1	4194324
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	268501025
t2	7	0
s0	8	1
s1	9	0
a0	10	268500992
a1	11	268501024
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	10
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	268500997
t4	29	0
t5	30	0
t6	31	0
pc		4194348

⇒ Kết quả: Quan sát trong Data Segment, x đã chứa "Hello\0", giống y.

4. Assignment 4:

Nhập chương trình:

Edit Execute

Lab05_1.asm riscv2.asm Lab05_3.asm **Lab05_4.asm**

```
1 .data
2 buffer: .space 100
3 message1: .asciz "Nhập xau: "
4 message2: .asciz "Do dai xau la: "
5 message3: .asciz "Xau vuot qua ki tu hoac khong ton tai"
6 .text
7 li a7, 54
8 la a0, message1
9 la a1, buffer
10 li a2, 100
11 ecall
12
13 get_length:
14 la a0, buffer # a0 = address(string[0])
15 li t0, 0 # t0 = i = 0
16 check_char:
17 add t1, a0, t0 # t1 = a0 + t0 = address(string[0]+i)
18 lb t2, 0(t1) # t2 = string[i]
19 beq t2, zero, end_of_str # Nếu là ký tự NULL thì kết thúc
20 addi t0, t0, 1 # t0 = t0 + 1 -> i = i + 1
21 j check_char
22 end_of_str:
23 end_of_get_length:
24 print_length:
25 blt a1,zero,no_string      # neu a1 < 0 thi se khong co xau nao duoc nhap
26 addi t1,zero,1
27 sub a1,t0,t1
```

Edit Execute

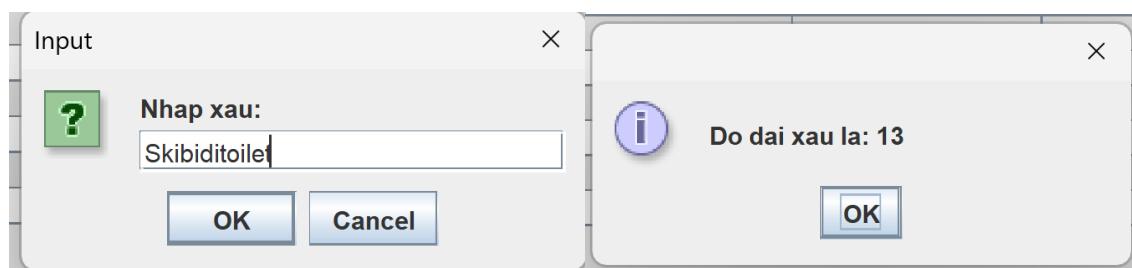
Lab05_1.asm riscv2.asm Lab05_3.asm **Lab05_4.asm**

```
14 la a0, buffer # a0 = address(string[0])
15 li t0, 0 # t0 = i = 0
16 check_char:
17 add t1, a0, t0 # t1 = a0 + t0 = address(string[0]+i)
18 lb t2, 0(t1) # t2 = string[i]
19 beq t2, zero, end_of_str # Nếu là ký tự NULL thì kết thúc
20 addi t0, t0, 1 # t0 = t0 + 1 -> i = i + 1
21 j check_char
22 end_of_str:
23 end_of_get_length:
24 print_length:
25 blt a1,zero,no_string      # neu a1 < 0 thi se khong co xau nao duoc nhap
26 addi t1,zero,1
27 sub a1,t0,t1
28 j have_string
29 no_string:
30 li a7, 59                  # in ra message 3
31 la a0, message3
32
33 ecall
34 j end
35 have_string:
36 li a7,56                  # in ra gia tri do dai xau khong tinh null
37 la a0,message2
38 ecall
39 end:
```

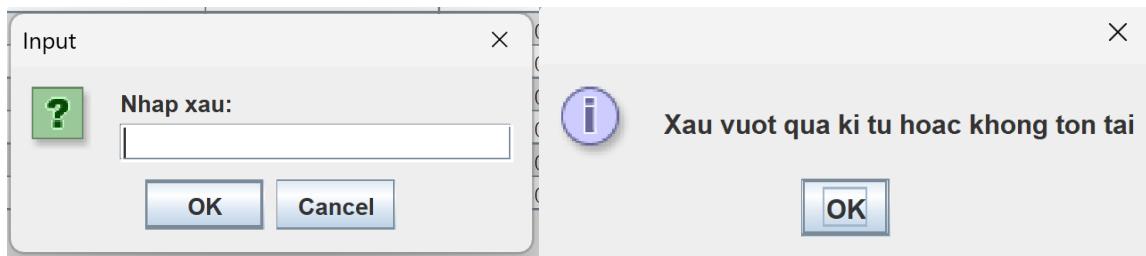
Giải thích code:

- + Cụm .data để khởi tạo chuỗi ký tự và buffer rỗng để lưu chuỗi nhập vào: Khởi tạo buffer (100 byte) để lưu chuỗi nhập, message1, message2, message3 là các chuỗi thông báo.
- + Cụm các câu lệnh syscall 54 để nhập chuỗi: Gán syscall 54 để nhập chuỗi, a0 là thông báo "Nhập xau: ", a1 là buffer, a2 là giới hạn ký tự (100).
- + Câu lệnh khởi tạo t0 = 0 và la a0, buffer để lấy địa chỉ của buffer.
- + Câu lệnh add t1, a0, t0 để lấy địa chỉ byte tiếp theo trong buffer.
- + Câu lệnh lb t2, 0(t1) để lấy dữ liệu từ byte có địa chỉ t1.
- + Câu lệnh beq t2, zero, end_of_str để kiểm tra nếu gặp ký tự NULL (\0) thì kết thúc vòng lặp.
- + Câu lệnh addi t0, t0, 1 để tăng giá trị biến đếm.
- + Câu lệnh j check_char để quay lại vòng lặp và kiểm tra ký tự tiếp theo.
- + Kiểm tra nếu chuỗi rỗng hoặc không hợp lệ: blt a1, zero, no_string.
- + Câu lệnh sub a1, t0, t1 để tính độ dài thực của chuỗi (không tính ký tự NULL).
- + In ra độ dài chuỗi: Gọi syscall 56 để in "Do dai xau la: ".
=> Sử dụng vòng lặp để duyệt qua từng byte trong buffer, đếm số ký tự cho đến khi gặp ký tự NULL (\0). Nếu chuỗi rỗng, in thông báo lỗi. Nếu chuỗi hợp lệ, in ra độ dài chuỗi (không tính ký tự NULL).

✓ **In ra xâu bình thường:**



✓ **In ra xâu quá dài hoặc rỗng:**



5. Assignment 5:

❖ Ý tưởng chương trình:

Hiển thị thông báo "Nhập chuoi: " để yêu cầu người dùng nhập dữ liệu.

- Đọc từng ký tự một bằng syscall 12 (đọc ký tự từ bàn phím).

- ✓ Lưu từng ký tự vào mảng str.
- ✓ Giới hạn tối đa 20 ký tự – nếu đạt đến 20 ký tự thì tự động dừng.
- ✓ Dừng nhập nếu gặp phím Enter (\n).

- Ghi ký tự NULL ('\0') vào cuối chuỗi để đánh dấu kết thúc.

- Đảo ngược chuỗi bằng cách:

- ✓ Duyệt từ cuối về đầu của str.
- ✓ Sao chép từng ký tự vào mảng reversed.

- Hiển thị chuỗi đã đảo ngược bằng syscall 4.

Mã code:

```
.data
    str:      .space 21          # Dự phòng 1 ký tự NULL ('\0')
    reversed: .space 21
    message1: .asciz "Nhập chuoi: "
    message2: .asciz "Chuoi dao nguoc: "

.text
.globl main
main:
    # In thông báo nhập chuỗi
```

```

    li a7, 4
    la a0, message1
    ecall

    # Đọc từng ký tự bằng syscall 12 (giới hạn 20 ký tự)
    la t0, str          # t0 trả đến str
    li t1, 0            # t1 là biến đếm độ dài

read_loop:
    li a7, 12          # Syscall 12: Đọc 1 ký tự
    ecall
    li t6, 10
    beq a0, t6, end_input # Nếu gặp Enter ('\n'), kết thúc nhập
    sb a0, 0(t0)        # Lưu ký tự vào str
    addi t0, t0, 1        # Tăng lên 1 ô nhớ
    addi t1, t1, 1        # Tăng biến đếm độ dài chuỗi
    li t2, 20            # Giới hạn 20 ký tự
    beq t1, t2, end_input # Nếu đã nhập đủ 20 ký tự, dừng nhập
    j read_loop

end_input:
    sb zero, 0(t0)      # Kết thúc chuỗi bằng NULL ('\0')

    # Đảo ngược chuỗi

reverse_string:
    la t2, str          # t2 trả đến str
    add t2, t2, t1        # Dời t2 đến ký tự cuối của chuỗi
    la t3, reversed       # t3 trả đến reversed

```

```

reverse_loop:

    blt t1, zero, show_result    # Khi hết ký tự thì dừng

    lb t4, -1(t2)      # Lấy ký tự cuối cùng từ str

    sb t4, 0(t3)       # Gán vào reversed

    addi t2, t2, -1     # Lùi về trước trong str

    addi t3, t3, 1      # Tiên lên trong reversed

    addi t1, t1, -1

    j reverse_loop


show_result:

    sb zero, 0(t3)        # Thêm ký tự kết thúc chuỗi vào reversed

    li a7, 4               # Print thông báo

    la a0, message2

    ecall

    li a7, 4               # Print reversed

    la a0, reversed

    ecall


exit:

    li a7, 10              # Syscall 10: Thoát

    ecall

```

Nhập chương trình:

Lab05_1.asm riscv2.asm Lab05_3.asm Lab05_4.asm Lab05_5.asm*

```

34 # Đảo ngược chuỗi
35 reverse_string:
36     la t2, str          # t2 trỏ đến str
37     add t2, t2, t1      # Dời t2 đến ký tự cuối của chuỗi
38     la t3, reversed    # t3 trỏ đến reversed
39
40 reverse_loop:
41     blt t1, zero, show_result # Khi hết ký tự thì dừng
42     lb t4, -1(t2)        # Lấy ký tự cuối cùng từ str
43     sb t4, 0(t3)        # Gán vào reversed
44     addi t2, t2, -1      # Lùi về trước trong str
45     addi t3, t3, 1       # Tiến lên trong reversed
46     addi t1, t1, -1
47     j reverse_loop
48
49 show_result:
50     sb zero, 0(t3)      # Thêm ký tự kết thúc chuỗi vào reversed
51     li a7, 4             # Print thông báo
52     la a0, message2
53     ecall
54     li a7, 4             # Print reversed
55     la a0, reversed
56     ecall
57
58 exit:
59     li a7, 10            # Syscall 10: Thoát
60     ecall

```

Edit Execute

Lab05_1.asm riscv2.asm Lab05_3.asm Lab05_4.asm Lab05_5.asm*

```

1 .data
2     str:      .space 21          # Dự phòng 1 ký tự NULL ('\0')
3     reversed: .space 21
4     message1: .asciz "Nhập chuỗi: "
5     message2: .asciz "Chuỗi đảo ngược: "
6
7 .text
8 .globl main
9 main:
10    # In thông báo nhập chuỗi
11    li a7, 4
12    la a0, message1
13    ecall
14
15    # Đọc từng ký tự bằng syscall 12 (giới hạn 20 ký tự)
16    la t0, str          # t0 trỏ đến str
17    li t1, 0             # t1 là biến đếm độ dài
18
19 read_loop:
20    li a7, 12            # Syscall 12: Đọc 1 ký tự
21    ecall
22    li t6, 10
23    beq a0, t6, end_input # Nếu gặp Enter ('\n'), kết thúc nhập
24    sb a0, 0(t0)         # Lưu ký tự vào str
25    addi t0, t0, 1        # Tiến lên 1 ô nhớ
26    addi t1, t1, 1        # Tăng biến đếm độ dài chuỗi
27    li t2, 20            # Giới hạn 20 ký tự

```

Chạy thử một vài trường hợp

TH1: độ dài xâu nhỏ hơn 20:

Edit Execute

Text Segment

Program Arguments:

Bkpt	Address	Code	Basic	Source
	0x00400000	0x00400893	addi x17,x0,4	11: li a7, 4
	0x00400004	0x0fc10517	auipc x10,0x0000fc10	12: la a0, message1
	0x00400008	0x02650513	addi x10,x10,0x00000026	
	0x0040000c	0x00000073	ecall	13: ecall
	0x00400010	0x0fc10297	auipc x5,0x0000fc10	16: la t0, str # t0 trả đến str
	0x00400014	0xff028293	addi x5,x5,0xffffffff	
	0x00400018	0x00000313	addi x6,x0,0	17: li t1, 0 # t1 là biến đếm độ dài
	0x0040001c	0x00c00893	addi x17,x0,12	20: li a7, 12 # Syscall 12: Đọc 1 ký..
	0x00400020	0x00000073	ecall	21: ecall
	0x00400024	0x0aa00f93	addi x31,x0,10	22: li t6, 10

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x62696b53	0x00696469	0x1 32-bit value stored 8 bytes beyond base address for its row.	0x69646900	0x536b6962	0x00000000		
0x10010020	0x00000000	0x00000000	0x684e0000	0x63207061	0x696f7568	0x4300203a	0x696f7568	0x6f616420
0x10010040	0x75676e20	0x203a636f	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Messages Run I/O

-- program is finished running (dropped off bottom) --

Clear

Nhap chuoi: Skibidi
Chuoi dau nguoc: idibikS
-- program is finished running (0) --

⇒ Kết quả trả về xâu đảo ngược như kỳ vọng.

TH2: Thủ nhập xâu dài hơn 20 ký tự:

Edit Execute

Text Segment

Program Arguments:

Bkpt	Address	Code	Basic	Source
	0x00400000	0x00400893	addi x17,x0,4	11: li a7, 4
	0x00400004	0x0fc10517	auipc x10,0x0000fc10	12: la a0, message1
	0x00400008	0x02650513	addi x10,x10,0x00000026	
	0x0040000c	0x00000073	ecall	13: ecall
	0x00400010	0x0fc10297	auipc x5,0x0000fc10	16: la t0, str # t0 trả đến str
	0x00400014	0xff028293	addi x5,x5,0xffffffff	
	0x00400018	0x00000313	addi x6,x0,0	17: li t1, 0 # t1 là biến đếm độ dài
	0x0040001c	0x00c00893	addi x17,x0,12	20: li a7, 12 # Syscall 12: Đọc 1 ký..
	0x00400020	0x00000073	ecall	21: ecall
	0x00400024	0x0aa00f93	addi x31,x0,10	22: li t6, 10

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x64636261	0x68676665	0x616c6a69	0x6c616e64	0x64616e64	0x6e616400	0x6e616c64	0x6a6c6164
0x10010020	0x66676869	0x62636465	0x68000061	0x63207061	0x696f7568	0x4300203a	0x696f7568	0x6f616420
0x10010040	0x75676e20	0x203a636f	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Messages Run I/O

Nhap chuoi: Skibidi
Chuoi dau nguoc: idibikS
-- program is finished running (0) --

Clear

Nhap chuoi: abcdefghijklabcdabc
Chuoi dau nguoc: dandlandaljihgfedcba
-- program is finished running (0) --

⇒ Kết quả là ngay sau khi nhập ký tự thứ 20 thì chương trình ngay lập tức kết thúc và trả về xâu đảo ngược như kỳ vọng.

