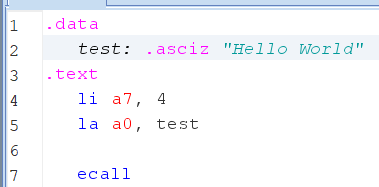
BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 5

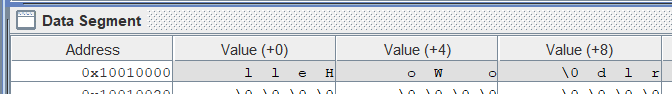
Họ và tên: Nguyễn Quý Đức MSSV: 20235682

Mã HP: IT3280 Mã lớp: 156788

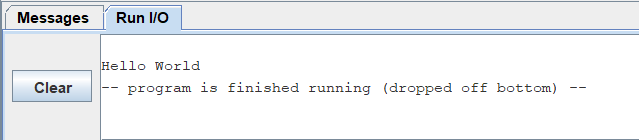
1. Assignment 1



- Data segment:



- Kết quả chạy chương trình:



- Trạng thái các thanh ghi:

|  |  |  |  |
| --- | --- | --- | --- |
| Trạng thái | a7 | a0 | pc |
| Ban đầu | 0x00000000 | 0x00000000 | 0x00400000 |
| li a7, 4 | 0x00000004 | - | 0x00400004 |
| la a0, test | - | 0x10010000 | 0x0040000c |
| ecall | - | - | 0x00400010 |

- Chuỗi "Hello world\0" được lưu trong bộ nhớ với kiểu litle-endian: "Hell" thành “l l e H”, "o Wo" thành “o W \_ o”, và "rld\0" thành “\0 d l r”. (Chuỗi xuất hiện ‘\0’ ở cuối do sử dụng lệnh .asciz lưu chuỗi và tự động thêm \0 vào cuối)

1. Assignment 2

.data

msg1: .asciz "The sum of "

msg2: .asciz " and "

msg3: .asciz " is "

msg4: .asciz "\n"

.text

main:

li s0, 10

li s1, 20

li a7, 4

la a0, msg1

ecall # print "The sum of "

li a7, 1

mv a0, s0

ecall # print value of s0

li a7, 4

la a0, msg2

ecall # print " and "

li a7, 1

mv a0, s1

ecall # print value of s1

li a7, 4

la a0, msg3

ecall # print " is "

add t0, s0, s1

li a7, 1

mv a0, t0

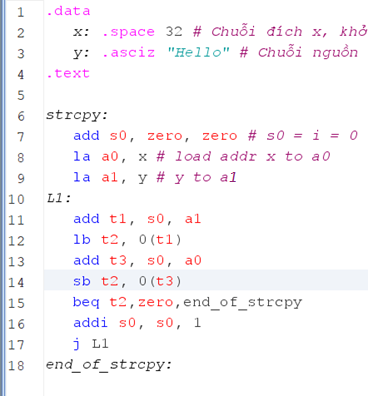
ecall # print sum

li a7, 10

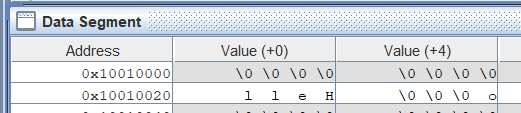
ecall

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lệnh | a7 (hex) | a0 (hex) | s0 (hex) | s1 (hex) |
| Trạng thái ban đầu | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| li s0, 10 | - | - | 0x0000000A | - |
| li s1, 20 | - | - | - | 0x00000014 |
| li a7, 4 | 0x00000004 | - | - | - |
| la a0, msg1 | - | 0x10010000 | - | - |
| ecall | - | - | - | - |
| li a7, 1 | 0x00000001 | - | - | - |
| mv a0, s0 | - | 0x0000000A | - | - |
| ecall | - | - | - | - |
| li a7, 4 | 0x00000004 | - | - | - |
| la a0, msg2 | - | 0x1001000C | - | - |
| ecall | - | - | - | - |
| li a7, 1 | 0x00000001 | - | - | - |
| mv a0, s1 | - | 0x00000014 | - | - |
| ecall | - | - | - | - |
| li a7, 4 | 0x00000004 | - | - | - |
| la a0, msg3 | - | 0x10010012 | - | - |
| ecall | - | - | - | - |
| add t0, s0, s1 | - | - | - | - |
| li a7, 1 | 0x00000001 | - | - | - |
| mv a0, t0 | - | 0x0000001E | - | - |
| ecall | - | - | - | - |
| li a7, 10 | 0x0000000A | - | - | - |
| ecall | - | - | - | - |

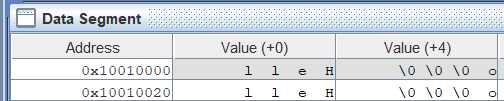
1. Assignment 3



- Trước khi strcpy



- Sau khi strcpy



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Trạng thái** | **a0** | **a1** | **s0** | **t1** | **t2** | **t3** |
| **Ban đầu** | 0x10010000 | 0x10010020 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| **Loop 1** | - | - | 0x00000000 | 0x10010020 | 0x00000048(H) | 0x10010000 |
| **Loop 2** | - | - | 0x00000001 | 0x10010021 | 0x00000065(e) | 0x10010001 |
| **Loop 3** | - | - | 0x00000002 | 0x10010022 | 0x0000006C(l) | 0x10010002 |
| **Loop 4** | - | - | 0x00000003 | 0x10010023 | 0x0000006C(l) | 0x10010003 |
| **Loop 5** | - | - | 0x00000004 | 0x10010024 | 0x0000006F(o) | 0x10010004 |
| **Loop 6** | - | - | 0x00000005 | 0x10010025 | 0x00000000(\0) | 0x10010005 |
| **Kết thúc** | - | - | 0x00000005 | - | - | - |

Khởi tạo x là buffer rỗng có kích thước 32 byte, y là chuỗi "Hello".

Gán địa chỉ x vào a0 và địa chỉ y vào a1.

Khởi tạo s0 = 0 để làm biến đếm vị trí (index) của ký tự đang xử lý.

Ở mỗi vòng lặp:

t1 = s0 + a1: Trỏ đến địa chỉ của y[i].

t2 = lb 0(t1): Lấy giá trị ký tự y[i] từ bộ nhớ.

t3 = s0 + a0: Trỏ đến địa chỉ x[i].

sb t2, 0(t3): Sao chép ký tự từ y[i] sang x[i].

Khi t2 == 0 ('\0'), chương trình nhảy đến end\_of\_strcpy và kết thúc.

Nếu t2 != 0, tăng s0 để lặp lại cho ký tự tiếp theo.

Kết thúc strcpy

1. Assignment 4

.data

string: .space 50

message1: .asciz "Nhap xau: "

message2: .asciz "Do dai xau la: "

.text

main:

get\_string:

li a7, 54

la a0, message1

la a1, string

li a2, 50

ecall

li a3, 0xa #value of newline in ascii

get\_length:

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

li t0, 0 # t0 = i = 0

check\_char:

add t1, a0, t0 # t1 = a0 + t0 = address(string[0]+i)

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

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

beq t2, a3, end\_of\_str # Is newline char?

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

j check\_char

end\_of\_str:

end\_of\_get\_length:

print\_length:

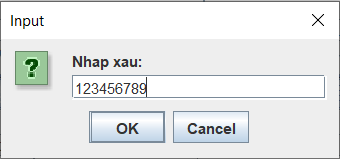
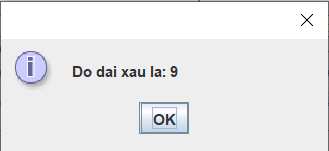
la a0, message2

li a7, 56

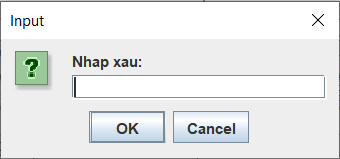
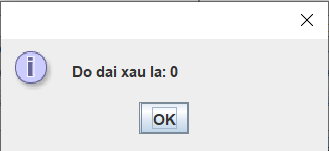
mv a1, t0

ecall

- Kết quả:

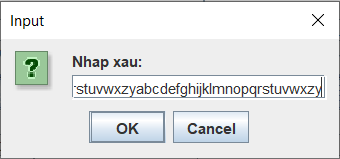
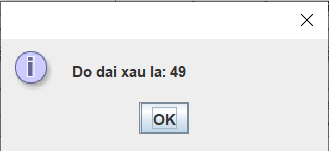
 

- TH không nhập gì:

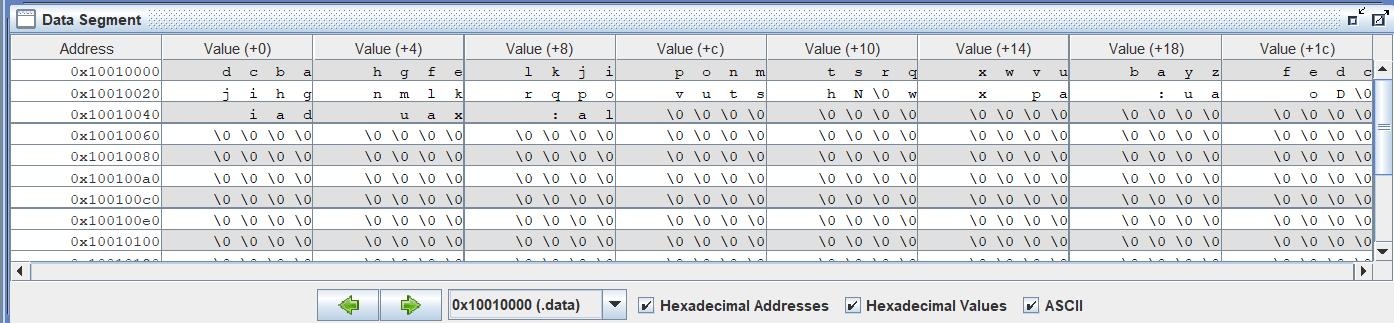
 

- TH chuỗi nhập vào có len > 50

(ở đây len(abcdefghijklmnopqrstuvwxzyabcdefghijklmnopqrstuvwxzy)=52)

(kí tự cuối ở địa chỉ 0x10010031 mặc định là ‘\0’ null terminator)



1. Assignment 5

.data

buffer: .space 21 # 20 characters + \0 + 1 byte tránh overflow

ask: .asciz "Enter a string (max 20 characters): "

ans: .asciz "\nReversed string: "

inverted\_string: .space 21 # Space for the inverted string

err: .asciz "Entered string is empty"

.text

.globl main

main:

li a7, 4

la a0, ask

ecall

li a7, 8

la a0, buffer

li a1, 21 # Maximum len 20 char + \0

ecall

lb t0, 0(a0) # Load first character of buffer

li t1, 10 # '\n' (ASCII 10)

beq t0, t1, end\_error # If first character is '\n', jump to end.error

la a1, inverted\_string

jal ra, reverse\_string

la a0, ans

li a7, 4

ecall

la a0, inverted\_string

li a7, 4

ecall

end:

li a7, 10

ecall

end\_error:

li a7, 4

la a0, err

ecall

li a7, 10

ecall

reverse\_string:

mv t0, a0

li t1, 0

find\_length:

lbu t2, 0(t0)

beq t2, zero, length\_found

addi t0, t0, 1

addi t1, t1, 1

j find\_length

length\_found:

addi t0, t0, -1

lbu t2, 0(t0) # Kiểm tra ký tự cuối

li t3, 10 # '\n'

beq t2, t3, skip\_last\_char

j reverse

skip\_last\_char:

addi t0, t0, -1

addi t1, t1, -1

reverse:

lbu t2, 0(t0)

sb t2, 0(a1)

addi t0, t0, -1

addi a1, a1, 1

addi t1, t1, -1

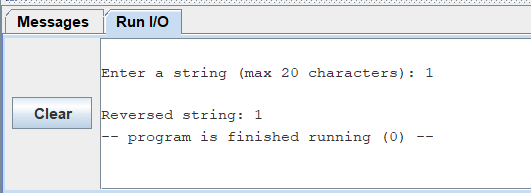
bnez t1, reverse

sb zero, 0(a1)

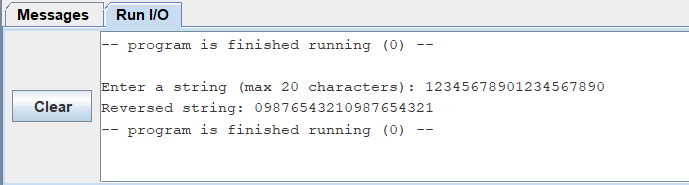
jr ra

- Kết quả:

+ TH nhập < 20 kí tự (kí tự cuối không tính \0 sẽ là \n nên inverted\_string sẽ có \n ở vị trí 0, như trong hình “1\n\0” -> “\n1”)



+ TH nhập đến kí tự thứ 20



+ TH không nhập gì (lúc này chuỗi nhập vào sẽ là “\n”, lúc đó lệnh beq t0, t1, end\_error sẽ so sánh t0 = str[0] với t1 = ‘\n’, do kết quả t0 == t1 nên sẽ branch ra label end\_error)

