

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

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

- Data segment:

| Data Segment | | | |
|--------------|-------------|-------------|-------------|
| Address | Value (+0) | Value (+4) | Value (+8) |
| 0x10010000 | l l e H | o W o | \0 d l r |
| 0x10010004 | \0 \0 \0 \0 | \0 \0 \0 \0 | \0 \0 \0 \0 |

- Kết quả chạy chương trình:

| Messages | Run I/O |
|------------------|---|
| <div>Clear</div> | Hello World -- program is finished running (dropped off bottom) -- |

- 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 little-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)

2. 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 | - | - | - | - |

3. Assignment 3

```

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

```

- Trước khi strcpy

| Data Segment | | | |
|--------------|-------------|-------------|--|
| Address | Value (+0) | Value (+4) | |
| 0x10010000 | \0 \0 \0 \0 | \0 \0 \0 \0 | |
| 0x10010020 | l l e H | \0 \0 \0 o | |

- Sau khi strcpy

| Data Segment | | | |
|--------------|------------|------------|--|
| Address | Value (+0) | Value (+4) | |
| 0x10010000 | l l e H | \0 \0 \0 o | |
| 0x10010020 | l l e H | \0 \0 \0 o | |

| 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 rộng 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`

4. Assignment 4

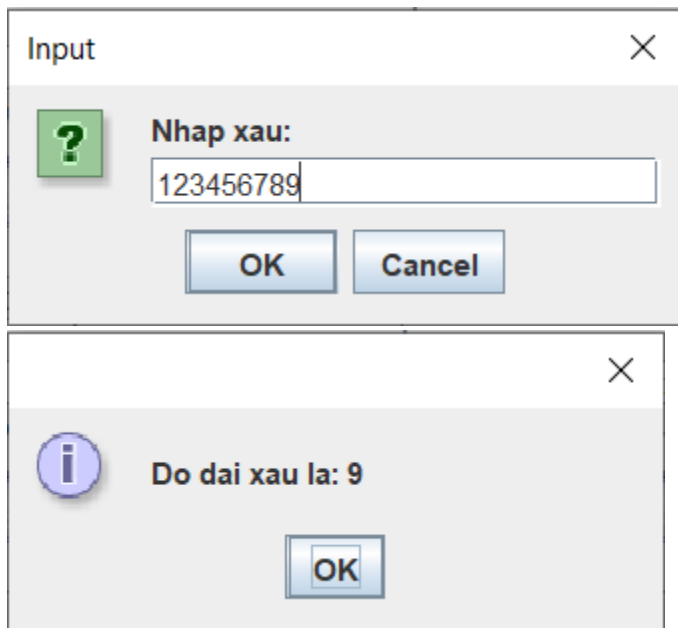
```
.data
string: .space 50
message1: .asciz "Nhap xau: "
message2: .asciz "Do dai xau
la: "

.text
main:
get_string:
    li a7, 54
    la a0, message1
    la a1, string
    li a2, 50
    ecall

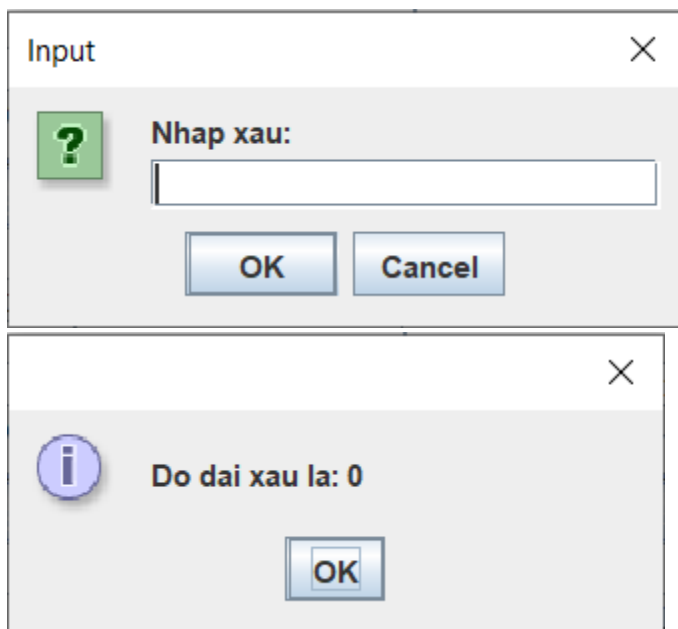
    li a3, 0xa #value of newline
in ascii
get_length:
    la a0, string # a0 =
address(string[0])
    li t0, 0 # t0 = i = 0
```

```
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:

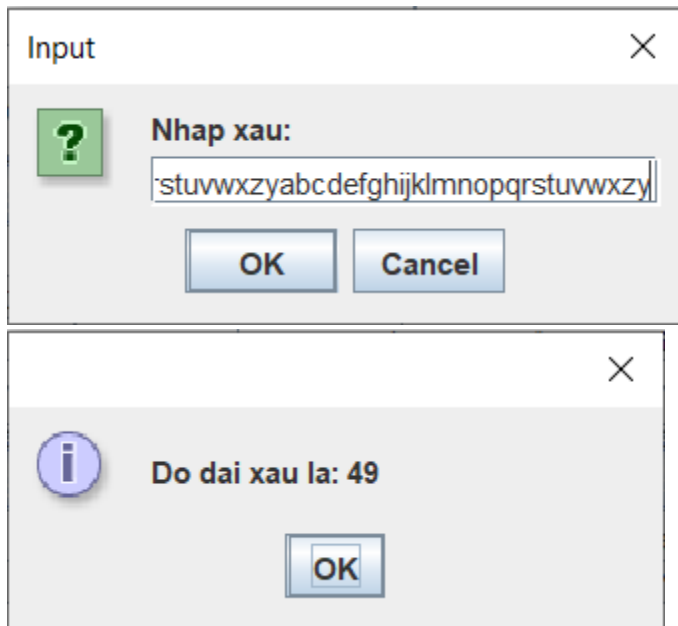


- TH không nhập gì:



- TH chuỗi nhập vào có len > 50

(ở đây len(abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz)=52)



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

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
|------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 0x10010000 | d c b a | h g f e | l k j i | p o n m | t s r q | x w v u | b a y z | f e d c |
| 0x10010020 | j i h g | n m l k | r q p o | v u t s | h N \0 w | x p a | : u a | o D \0 |
| 0x10010040 | i a d | u a x | : a l | \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 |
| 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 |
| 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 |
| 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 |
| 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 |
| 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 |

5.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

```

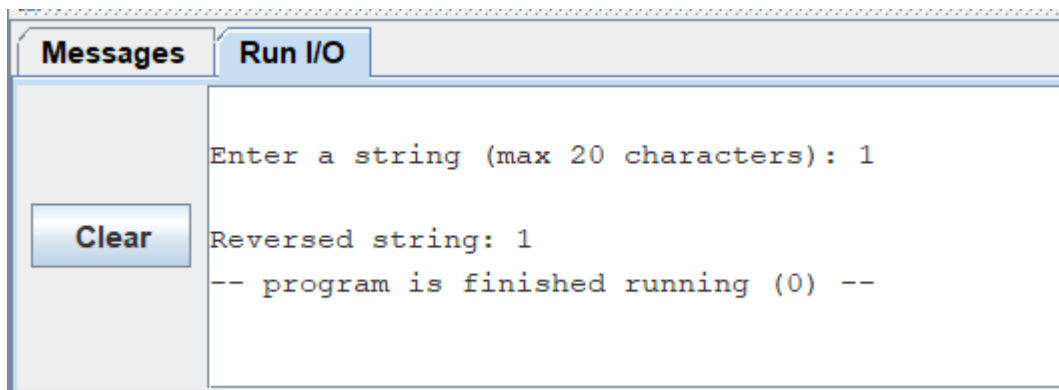

reverse:

```
lbu t2, 0(t0)
sb t2, 0(a1)
addi t0, t0, -1
```

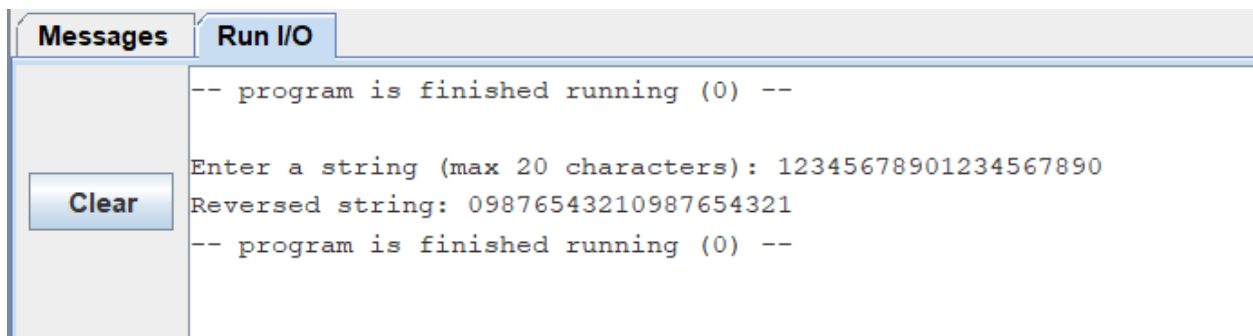
```
addi a1, a1, 1
addi t1, t1, -1
bnez t1, reverse
sb zero, 0(a1)
jr ra
```

- 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)

