BÁO CÁO THỨC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 5

Họ và tên: Vũ Đức Hoàng Anh

MSSV: 20235658

1. Assignment 1

- **Nhập chương trình:**

```
# Laboratory Exercise 5, Home Assignment 1
.data
        test: .asciz "Hello World"          # Khởi tạo chuỗi kí tự
.text
        li  a7, 4       # 4 là số hiệu dịch vụ in chuỗi kí tự
        la  a0, test    # Lấy địa chỉ của chuỗi kí tự
ecall
```

- **Các bước thực hiện của chương trình:**

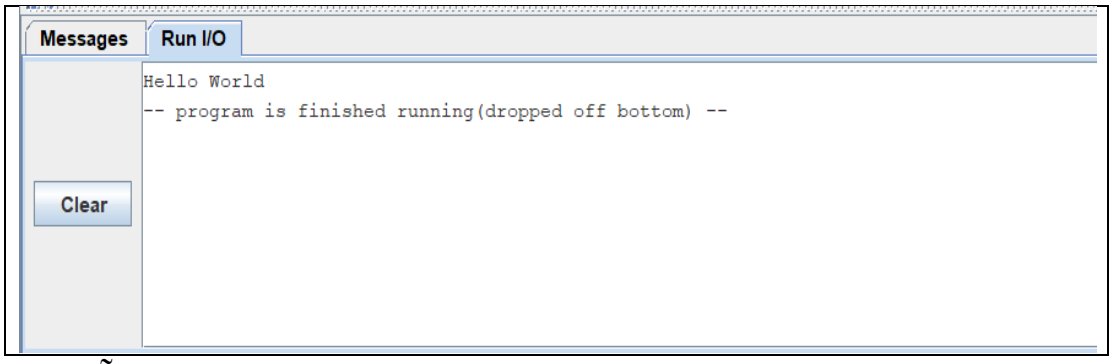
  + Câu lệnh test: .asciz “Hello World” để khởi tạo giá trị cho chuỗi kí tự “Hello World”.

  + 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.

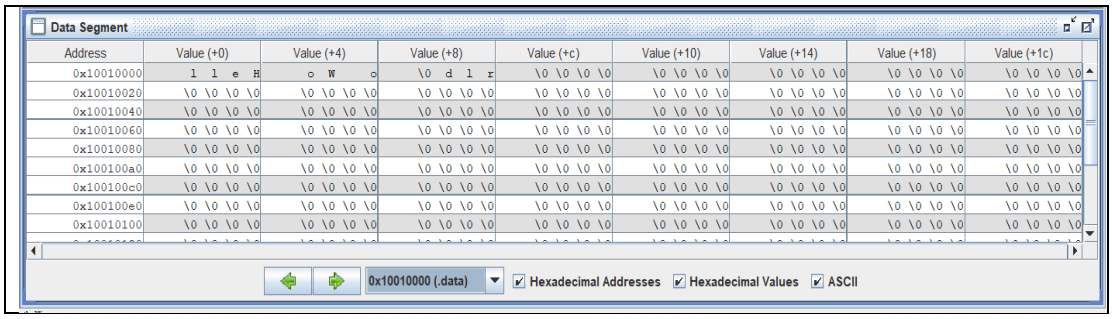
  + Câu lệnh ecall là lời gọi đến hệ thống, trao quyền cho hệ điều hành thực hiện chương trình.

- **Kết quả của chương trình:**



+ Chuỗi kí tự “Hello World được in ra màn hình

- **Dữ liệu ở bảng Data Segment:**



+ 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 lược "l l e H", "o W _ o", "\0 d l r" (kí tự '_' để mô tả thay thế dấu cách, \0 là kí tự kết thúc chuỗi kí tự).

2. Assignment 2

- **Nhập chương trình**

```
 .data
        msg1: .asciz "The sum of "
        msg2: .asciz " and "
        msg3: .asciz " is "
.text
# Gán giá trị số vào thanh ghi
            li s1, 10        # s1 = 10
            li s2, 20        # s2 = 20
            add s0, s1, s2    # s0 = s1 + s2
# In chuỗi "The sum of "
            li a7, 4
            la a0, msg1
            ecall
# In số s1
            li a7, 1
            mv a0, s1
            ecall
# In chuỗi " and "
            li a7, 4
            la a0, msg2
            ecall
# In số s2
            li a7, 1
```

```
            mv a0, s2
            ecall
# In chuỗi " is "
            li a7, 4
            la a0, msg3
            ecall
# In kết quả tổng
            li a7, 1
            mv a0, s0
            ecall
# Kết thúc chương trình
            li a7, 10
    ecall
```

- **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

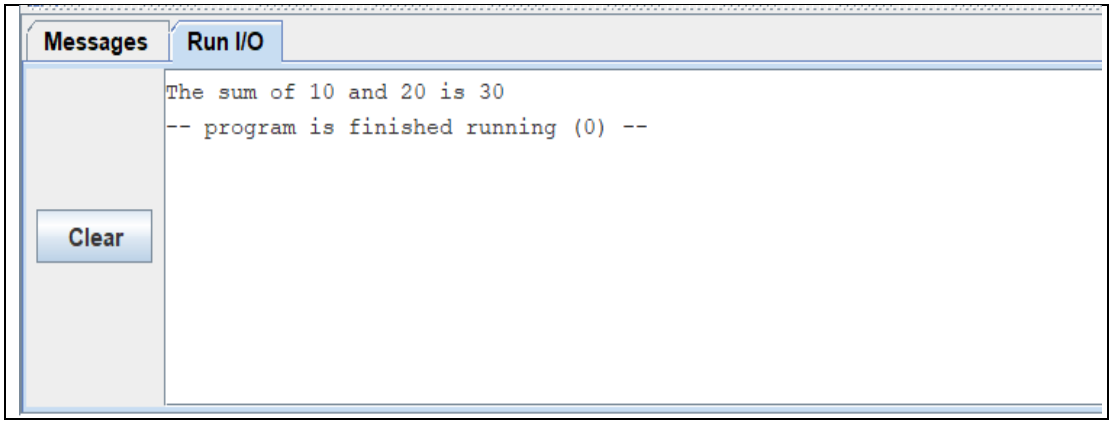
  + Cụm gán giá trị vào thanh ghi để gán giá trị cho các thanh ghi s1, s2 và tính tổng 2 thanh ghi

  + 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

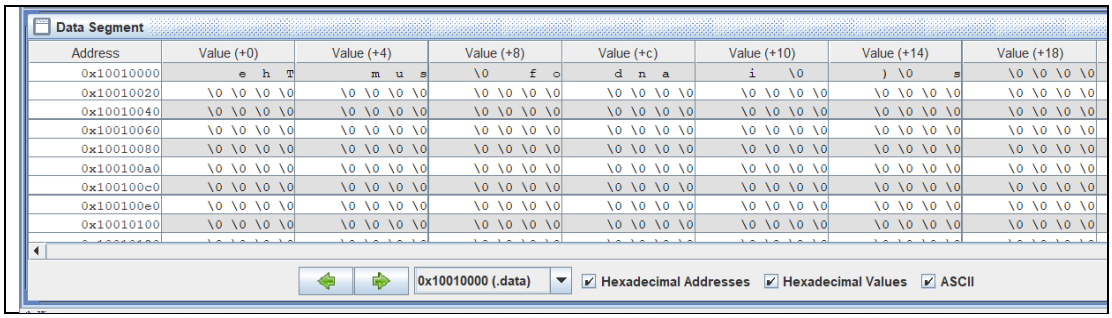
  + Các cụm in số và tổng sử dụng chung cú pháp li a7, 1 để in giá trị thuộc kiểu int. Sau đó sao chép giá trị các thanh ghi vào thanh ghi a0 để thực hiện ecall in ra màn hình.

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

- **Kết quả của chương trình:**



+ Dữ liệu lưu ở bảng data segment :



3. Assignment 3

- **Nhập chương trình:**

```
.data
        x: .space 32 # Chuỗi đích x, khởi tạo là buffer rỗng
        y: .asciz "Hello" # Chuỗi nguồn y
.text
        strcpy:
                add s0, zero, zero # s0 = i = 0
                la a0, x # load addr x to a0
                la a1, y # y to a0
        L1:
                add t1, s0, a1
                lb t2, 0(t1)
                add t3, s0, a0
                sb t2, 0(t3)
                beq t2, zero, end_of_strcpy
                addi s0, s0, 1
                j L1
end_of_strcpy:
```

- **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 đại 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 lênh 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

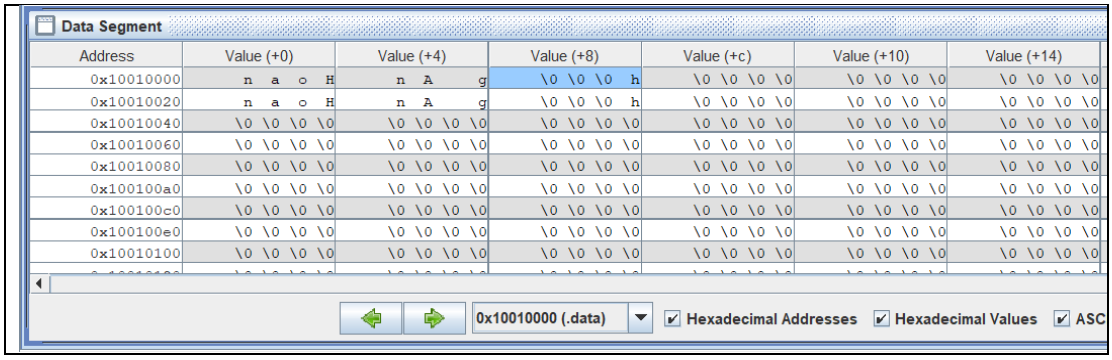
⇨ Sử dụng vòng lặp để đưa từng byte dữ liệu của y vào x đến khi không còn giá trị nào của y để đẩy vào x thì kết thúc vòng lặp.

- **Kết quả của chương trình**

+ Kết quả lý thuyết

Gán giá trị cho y = “Hoang Anh” thì dữ liệu đầu ra của x =”Hoang Anh” và được lưu ở của sổ data segment dưới dạng “n a o H”, ”n A _ g”, “\0 \0 \0 h” (Với ‘_’ thay thế cho kí tự dấu cách)

+ Kết quả hiện thị ở khung data segment



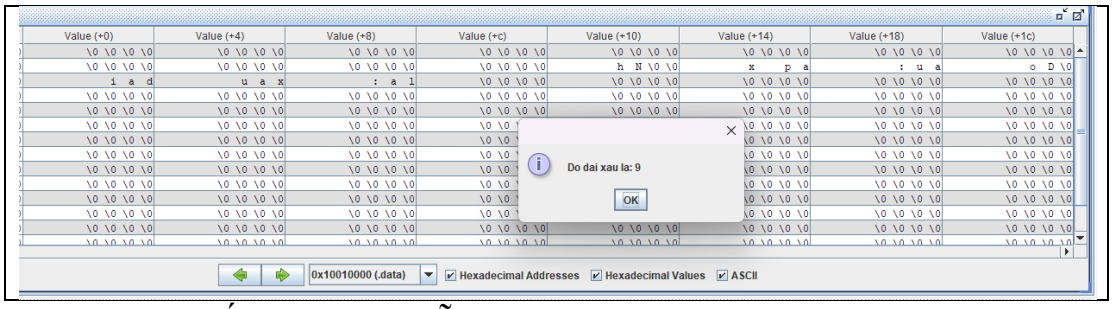
4. Assignment 4

- **Nhập chương trình:**

```
.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:
```

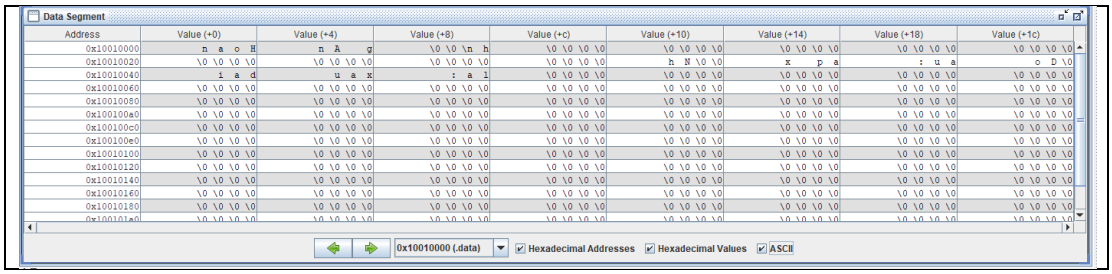
```
        la a0, string # a0 = address(string[0])
        li t0, 0 # t0 = i = 0
check_char:
        add t1, a0, t0 # t1 = a0 + t0 = address(string[0]+i)
        lb t2, 0(t1) # t2 = string[i]
        beq t2, zero, end_of_str # Is null char?
        beq t2, a3, end_of_str # Is newline char?
        addi 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
```

- **Các bước thực hiện:**
  + Cụm câu lệnh .data để để khai báo dữ liệu trong bộ nhớ, với string là ô nhớ trống, message 1,2 là các dãy kí tự thiết lập sẵn
  + Cụm câu lệnh tiếp theo khởi tạo giá trị của các thanh ghi. Câu lệnh li a7, 54 là lời gọi đến hệ thống in chuỗi ra màn hình, a0 là địa chỉ của message 1, a1 là địa chỉ của string, khởi tạo giá trị a2 = 50
  + Cụm câu lệnh tiếp theo để tìm độ dài chuỗi: Khởi tạo giá của a3 = 0xa tương ứng với kí tự \n, lấy a0 là địa chỉ của string, t0 = 0 là biến đếm
  + Cụm vòng lặp để kiểm tra từng kí tự: t1 để lưu địa chỉ của byte tiếp theo, t2 lấy địa chỉ của byte tại vị trí địa chỉ t1, sau đó kiểm tra t2 có phải là kí tự rỗng hay không, nếu là kí tự rỗng thì nhảy đến thẻ end_of_str, sau đó kiểm tra t2 có phải kí tự xuống dòng không, nếu là kí tự xuống dòng thì nhảy đến thẻ end_of_str và kết thúc vòng lặp, tiếp tục tăng biến đếm và nhảy đến thẻ check_char để quay lại vòng lặp.
  + Cụm câu lệnh in độ dài ra màn hình: Gọi a7 = 56 là in số nguyên ra màn hình, và in giá trị của a1 tương ứng với t0 và in ra màn hình
- **Kết quả thực hiện chương trình:**
  + Với string nhập vào là Hoang Anh



⇨ Đúng với kết quả của chuỗi Hoang Anh.

- **Kết quả ở của sổ Data segment:**



5. Assignment 5

- **Nhập chương trình:**

```
.data
buffer: .space 21 # 20 characters + \0
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
```

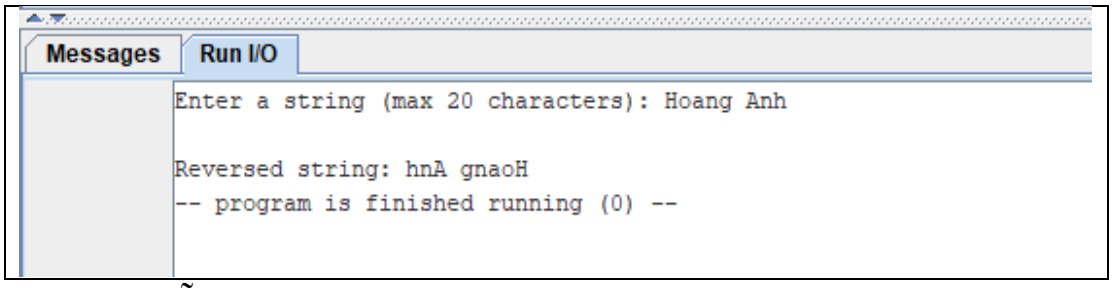
- **Các bước thực hiện:**

  + Cụm data khai báo các dữ liệu ở bộ nhớ. Khai báo bộ nhớ trống, khai báo các chuỗi kí tự để suất ra màn hình, và bộ nhớ lưu chuỗi đảo ngược
  + Cụm ecall đầu tiên để hiện cụm " Enter a string (max 20 characters):"
  + Cụm ecall thứ 2 để nhập chuỗi kí tự từ bàn phím với độ dài tối đa 20 kí tự. Kết quả được lưu vào bufer.
  + Tiếp sau đó kiểm tra kí tự tiếp theo của chuỗi nhập vào: nêis là \n thì chương trình báo lỗi và kết thúc.
  + Sau đó gọi hàm reverse_string để đảo ngược chuỗi: Gọi hàm reverse_string để đảo ngược chuỗi kết quả được lưu vào inverted_string, tiếp đó in thông báo của inverted_string ra ngoài màn hình.
  + Cuối cùng là in chuỗi đảo ngược ra màn hình thông qua li a7, 10
  + Các câu lệnh trong thẻ end_error để in ra màn hình câu lệnh "Entered string í empty"
  + Các câu lệnh trong thẻ reverse_string để lưu địa chỉ của buffer và đếm số kí tự chuỗi
  + Câu lệnh trong thẻ find_length để duyệt và tìm để dài của chuỗi dừng lại khi gặp kí tự \n
  + Câu lệnh trong thẻ skip_last_char để giảm độ dài của xâu khi kí tự cuối là kí tự \n

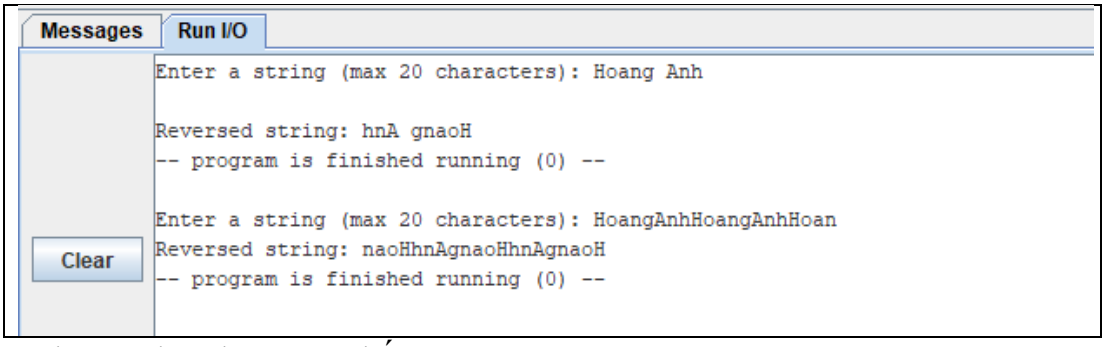
+ Câu lệnh trong thẻ reverse để đảo ngược kí tự từ cuối lên đầu và lưu vào inverted_string

- Kết quả thực hiện chương trình:

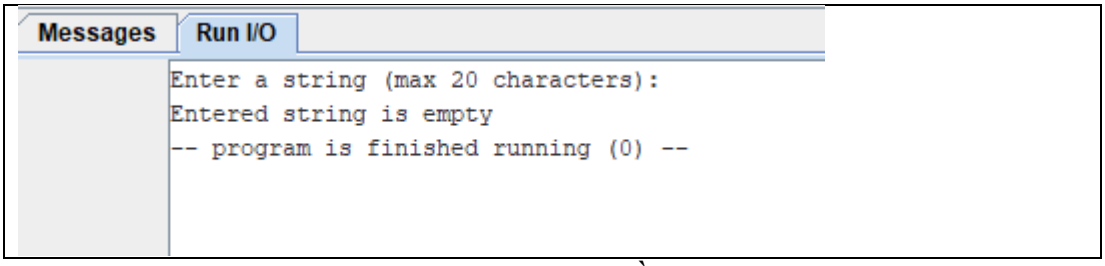
+ Nhập chuỗi kí tự Hoang Anh (có số kí tự < 20)



+ Nhập chuỗi kí tự có độ dài 20: HoangAnhHoangAnhHoan



+ Không nhập kí tự và nhấn enter:



⇨ Chương trình hoạt động đúng với yêu cầu.