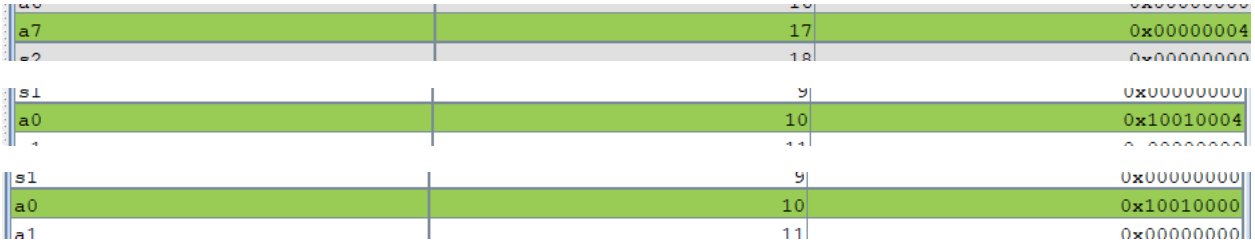
Nguyễn Đình Dương – 20225966

Assignment 1

Code:

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

Output:



Trong phân đoạn Data: chuỗi test (là "Hello World") được lưu tuần tự trong bộ nhớ dưới dạng các giá trị ASCII, với mỗi ký tự chiếm 1 byte. Chuỗi bắt đầu tại địa chỉ bộ nhớ 0x10010000 và được lưu liên tiếp theo cách sau:

H	0x10010000
e	0x10010001
l	0x10010002
l	0x10010003
o	0x10010004
space	0x10010005
W	0x10010006
o	0x10010007
r	0x10010008
l	0x10010009
d	0x1001000A

Chuỗi được kết thúc bằng một ký tự null (\0) tại địa chỉ 0x1001000B (giá trị là 0x00), đánh dấu kết thúc của chuỗi trong bộ nhớ. Ký tự null này cần thiết để chỉ định điểm kết thúc của chuỗi trong bộ nhớ.

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
0x10010040	\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

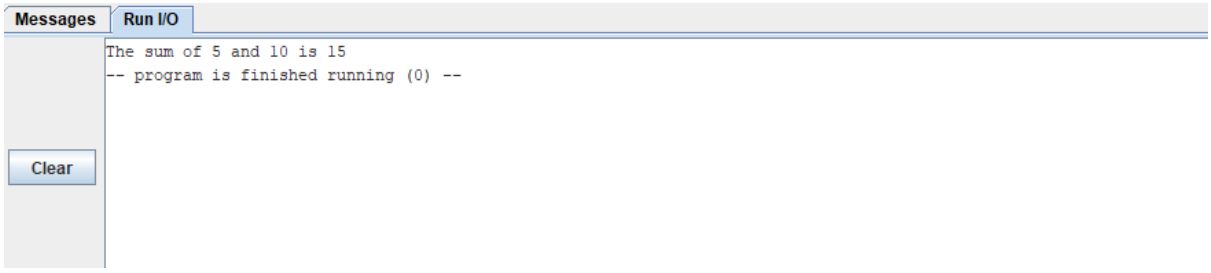
Assignment 2

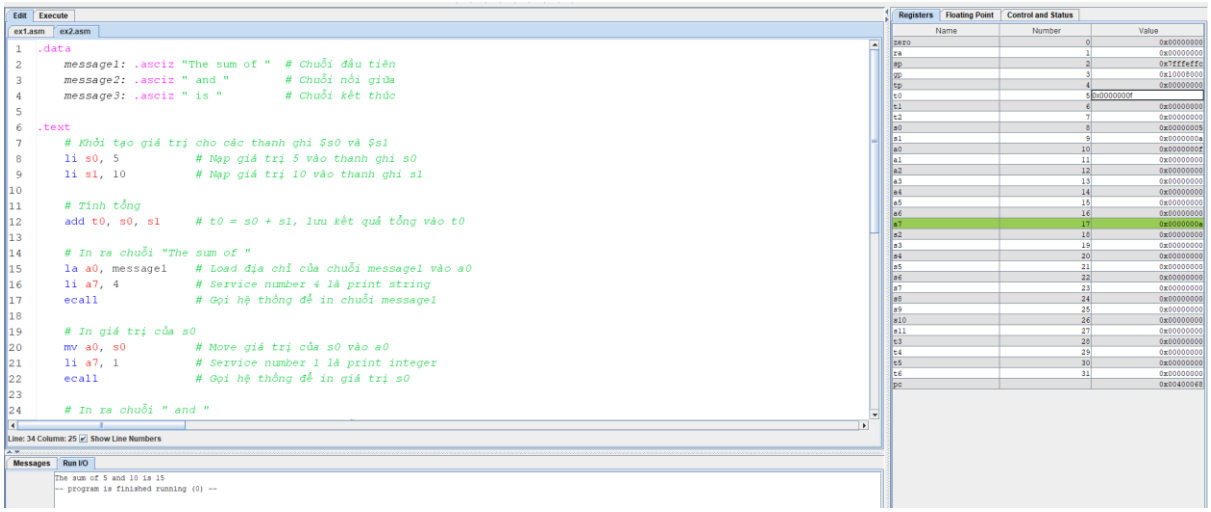
Code

```
.data
   message1: .asciz "The sum of "  # Chuỗi đầu tiên
   message2: .asciz " and "        # Chuỗi nối giữa
   message3: .asciz " is "         # Chuỗi kết thúc

.text
   # Khởi tạo giá trị cho các thanh ghi $s0 và $s1
   li s0, 5          # Nạp giá trị 5 vào thanh ghi s0
   li s1, 10         # Nạp giá trị 10 vào thanh ghi s1

   # Tính tổng
   add t0, s0, s1     # t0 = s0 + s1, lưu kết quả tổng vào t0

   # In ra chuỗi "The sum of "
   la a0, message1    # Load địa chỉ của chuỗi message1 vào a0
   li a7, 4           # Service number 4 là print string
   ecall              # Gọi hệ thống để in chuỗi message1

   # In giá trị của s0
   mv a0, s0          # Move giá trị của s0 vào a0
   li a7, 1           # Service number 1 là print integer
   ecall              # Gọi hệ thống để in giá trị s0

   # In ra chuỗi " and "
   la a0, message2    # Load địa chỉ của chuỗi message2 vào a0
   li a7, 4           # Service number 4 là print string
   ecall              # Gọi hệ thống để in chuỗi message2

   # In giá trị của s1
   mv a0, s1          # Move giá trị của s1 vào a0
   li a7, 1           # Service number 1 là print integer
   ecall              # Gọi hệ thống để in giá trị s1

   # In ra chuỗi " is "
   la a0, message3    # Load địa chỉ của chuỗi message3 vào a0
   li a7, 4           # Service number 4 là print string
   ecall              # Gọi hệ thống để in chuỗi message3
```

```
# In kết quả tổng
mv a0, t0          # Move giá trị tổng từ t0 vào a0
li a7, 1           # Service number 1 là print integer
ecall              # Gọi hệ thống để in kết quả tổng

# Thoát chương trình
li a7, 10          # Service number 10 là thoát chương trình
ecall
```

Output:





Giải thích chi tiết:

1. **Phần .data:**
    - Đây là nơi các chuỗi ký tự như "The sum of ", " and ", và " is " được lưu trữ để sử dụng khi in ra màn hình.
2. **Phần .text:**
    - li s0, 5 và li s1, 10 dùng để gán giá trị 5 vào thanh ghi s0 và giá trị 10 vào thanh ghi s1.
    - add t0, s0, s1 tính tổng của s0 và s1, sau đó lưu kết quả vào thanh ghi t0.
    - la a0, message1 nạp địa chỉ của chuỗi message1 vào thanh ghi a0, sau đó gọi hệ thống với lệnh ecall để in chuỗi.
    - Các lệnh tiếp theo tương tự để in giá trị của các thanh ghi và các chuỗi khác theo định dạng yêu cầu.

3. **Dịch vụ ECALL:**
    - li a7, 4: Dịch vụ số 4 dùng để in chuỗi ra màn hình.
    - li a7, 1: Dịch vụ số 1 dùng để in số nguyên ra màn hình.
    - li a7, 10: Dịch vụ số 10 dùng để thoát khỏi chương trình.

Mỗi bước sẽ in theo thứ tự từng sâu ký tự cho trước → Đúng với lý thuyết

Assignment 3

Code:

```
.data
   x: .space 32          # Chuỗi đích x, chưa có dữ liệu
   y: .asciz "Duong"     # Chuỗi nguồn y, chứa "Hello"
   newline: .asciz "\n"  # Ký tự xuống dòng

.text
   # Nạp địa chỉ của chuỗi đích x và chuỗi nguồn y vào thanh ghi
   la a0, x              # Nạp địa chỉ của chuỗi x vào a0
   la a1, y              # Nạp địa chỉ của chuỗi y vào a1

strcpy:
   add s0, zero, zero   # Khởi tạo s0 = 0 (dùng làm chỉ số i)
L1:
   add t1, s0, a1       # t1 = s0 + a1 = địa chỉ y[i]
   lb t2, 0(t1)         # t2 = giá trị tại y[i]
   add t3, s0, a0       # t3 = s0 + a0 = địa chỉ x[i]
   sb t2, 0(t3)         # x[i] = t2 = y[i]
   beq t2, zero, end_of_strcpy  # Nếu y[i] == 0 (kết thúc chuỗi), thoát
   addi s0, s0, 1       # s0 = s0 + 1 <-> i = i + 1
   j L1                 # Lặp lại cho ký tự tiếp theo
end_of_strcpy:

   # In chuỗi x ra màn hình để kiểm tra kết quả
   la a0, x             # Nạp địa chỉ chuỗi x vào a0
   li a7, 4             # Dịch vụ in chuỗi
   ecall                # Gọi hệ thống để in chuỗi x

   # In ký tự xuống dòng
   la a0, newline       # Nạp địa chỉ của newline
   li a7, 4             # Dịch vụ in chuỗi
   ecall                # Gọi hệ thống để in newline

   # Thoát chương trình
   li a7, 10            # Dịch vụ thoát chương trình
   ecall
```

Output:

```
-- program is finished running (0) --

Duong
```

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x10010025
t2	7	0x00000067
s0	8	0x00000005
s1	9	0x00000000
a0	10	0x10010000
a1	11	0x10010020
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x10010004
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400018

Các bước chạy:

1. **Khai báo chuỗi**:
   - Chuỗi đích x có 32 byte trống, chuỗi nguồn y chứa "Duong".
2. **Sao chép chuỗi**:
   - Hàm strcpy lặp qua từng ký tự trong chuỗi y, sao chép chúng vào chuỗi x.
   - Vòng lặp dừng khi gặp ký tự null (\0), tức là kết thúc chuỗi.
3. **In kết quả**:

- Sau khi sao chép, chương trình in chuỗi x ra màn hình, tiếp theo là ký tự xuống dòng (\n).
- Cuối cùng, chương trình thoát

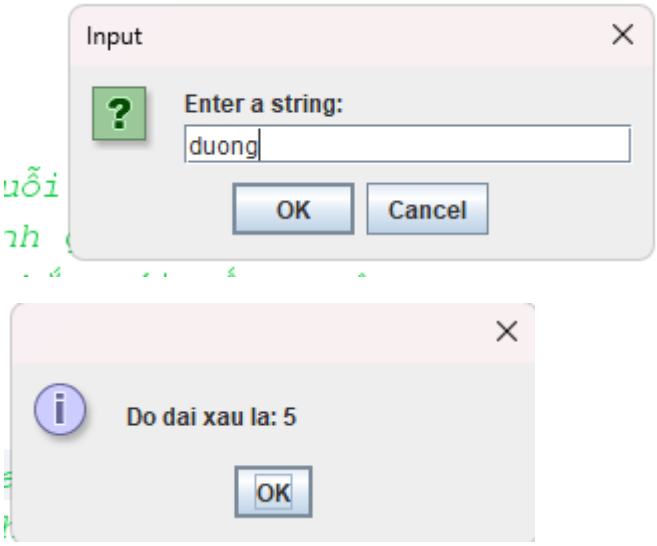
Assignment 4

Code:

```
.data
string: .space 50               # Dành không gian cho chuỗi nhập vào
message1: .asciz "Nhap xau: "    # Thông báo yêu cầu nhập chuỗi
message2: .asciz "Do dai xau la: "  # Thông báo độ dài chuỗi
newline: .byte 10               # Giá trị của ký tự newline (ASCII 10)

.text
main:
get_string:
   # Hiển thị hộp thoại để người dùng nhập chuỗi
   la a0, message1             # Nạp địa chỉ của thông báo yêu cầu nhập chuỗi
   la a1, string               # Nạp địa chỉ của chuỗi vào buffer
   li a2, 50                   # Giới hạn số ký tự nhập là 50
   li a7, 54                   # Mã ECALL để hiển thị hộp thoại nhập chuỗi
   ecall                       # Gọi ECALL hiển thị hộp thoại

   # Loại bỏ ký tự newline nếu có
   la a0, string               # Nạp địa chỉ của chuỗi vào a0
   li t0, 0                    # Đặt biến đếm i = 0

remove_newline:
   add t1, a0, t0              # Tính địa chỉ của string[i]
   lb t2, 0(t1)                # Lấy giá trị string[i]
   beq t2, zero, end_remove_nl  # Nếu gặp ký tự null, kết thúc vòng lặp
   li t3, 10                   # Giá trị ASCII của ký tự newline là 10
   beq t2, t3, set_null        # Nếu ký tự là newline, thay thế bằng null
   addi t0, t0, 1              # Tăng biến đếm i
   j remove_newline            # Lặp lại vòng lặp

set_null:
   sb zero, 0(t1)              # Thay newline bằng ký tự null
end_remove_nl:

get_length:
   li t0, 0                    # Đặt biến đếm i = 0

check_char:
   add t1, a0, t0              # Tính địa chỉ của string[i]
   lb t2, 0(t1)                # Lấy giá trị string[i] (t2 = string[i])
   beq t2, zero, end_of_str      # Nếu gặp ký tự null, kết thúc vòng lặp
   addi t0, t0, 1              # Tăng biến đếm i
   j check_char                # Lặp lại vòng lặp
```

```
end_of_str:
end_of_get_length:

print_length:
    # Hiển thị độ dài chuỗi ra màn hình bằng hộp thoại thông báo
    la a0, message2             # Nạp địa chỉ của thông báo "Do dai xau la: "
    mv a1, t0                   # Chuyển giá trị độ dài chuỗi vào thanh ghi a1
    li a7, 56                   # Mã ECALL để hiển thị hộp thoại thông báo độ dài
    ecall                       # Gọi ECALL hiển thị hộp thoại

    # Thoát chương trình
    li a7, 10                   # Mã ECALL để thoát chương trình
    ecall
```

Output:



Giải thích:

1. **Nhập chuỗi từ hộp thoại**:
    - Sử dụng **ECALL (mã 54)** để hiển thị hộp thoại nhập chuỗi và lưu chuỗi vào biến string.
2. **Loại bỏ ký tự xuống dòng (newline)**:
    - Vòng lặp remove_newline kiểm tra từng ký tự của chuỗi.
    - Nếu phát hiện ký tự xuống dòng (mã ASCII 10), nó sẽ thay thế bằng ký tự null (\0).
3. **Tính độ dài chuỗi**:
    - Vòng lặp check_char duyệt qua chuỗi từ đầu đến khi gặp ký tự null (\0) để đếm số ký tự.

4. **Hiển thị độ dài chuỗi**:

- Sử dụng **ECALL (mã 56)** để hiển thị độ dài chuỗi trong hộp thoại tin nhắn.

5. **Thoát chương trình**:

- Sử dụng **ECALL (mã 10)** để thoát chương trình sau khi hiển thị kết quả.

Assignment 5

```
    .data
string:     .space 21             # Cấp phát bộ nhớ cho chuỗi (20 ký tự + ký tự null)
newline:    .byte 0x0A            # Ký tự newline (Enter)
message:    .asciz "Chuoi dao nguoc: "
count_msg:  .asciz "\nSo ky tu da nhap: "

    .text
    .globl _start
_start:
    li      t0, 0               # t0 giữ chỉ số hiện tại của chuỗi
    la      t1, string          # t1 trỏ đến vị trí bắt đầu của chuỗi

input_loop:
    li      a7, 12              # Hàm đọc ký tự
    ecall                       # Lệnh gọi hệ thống (ECALL) để đọc ký tự
    sb      a0, 0(t1)           # Lưu ký tự vào bộ nhớ tại vị trí hiện tại
    la      t2, newline         # Tải giá trị của newline (Enter) vào t2
    lb      t2, 0(t2)           # Đọc giá trị thực tế của newline từ bộ nhớ vào t2
    beq     a0, t2, end_input   # Nếu nhập Enter, kết thúc nhập
    addi    t0, t0, 1           # Tăng chỉ số lên
    addi    t1, t1, 1           # Di chuyển con trỏ đến vị trí tiếp theo
    li      t2, 20              # Kiểm tra nếu vượt quá 20 ký tự
    bge     t0, t2, end_input   # Nếu >= 20 ký tự, dừng nhập
    j       input_loop          # Quay lại vòng lặp nhập

end_input:
    sb      zero, 0(t1)         # Thêm ký tự null để kết thúc chuỗi

    # Đảo ngược chuỗi
    la      t1, string          # Tải địa chỉ của chuỗi vào t1
    addi    t2, t1, -1          # t2 trỏ tới ký tự cuối cùng của chuỗi
    addi    t3, t0, -1          # t3 giữ chỉ số để đảo ngược

reverse_loop:
    bge     t3, zero, print_reverse    # Nếu chỉ số < 0, dừng
    lb      a0, 0(t2)           # Đọc ký tự từ chuỗi
    li      a7, 11              # Hàm in ký tự
    ecall                       # In ký tự
    addi    t2, t2, -1          # Di chuyển về ký tự trước
```

```
    addi    t3, t3, -1              # Giảm chỉ số
    j       reverse_loop            # Quay lại vòng lặp

print_reverse:
    # In chuỗi đã nhập
    la      a0, string          # Đưa địa chỉ chuỗi vào a0
    li      a7, 4               # Hàm in chuỗi
    ecall

    # In thông báo về số ký tự đã nhập
    la      a0, count_msg       # Đưa thông báo vào a0
    li      a7, 4               # Hàm in chuỗi
    ecall

    # In số lượng ký tự
    mv      a0, t0              # Chuyển giá trị số ký tự đã nhập vào a0
    li      a7, 1               # Hàm in số nguyên
    ecall

    # Thoát chương trình
    li      a7, 10              # Hàm thoát chương trình
    ecall
```

Output:

```
Reset: reset completed.

duongdeptrai
duongdeptrai
So ky tu da nhap: 12
-- program is finished running (0) --
```

1. **Khởi tạo**:
    - Cấp phát bộ nhớ cho chuỗi (string), khởi tạo biến để theo dõi số ký tự đã nhập (t0).
2. **Vòng lặp nhập ký tự**:
    - Sử dụng ECALL 12 để đọc từng ký tự từ bàn phím.
    - Lưu ký tự vào chuỗi. Nếu người dùng nhấn Enter hoặc nhập quá 20 ký tự, kết thúc quá trình nhập.
3. **Kết thúc nhập**:
    - Thêm ký tự null (\0) để kết thúc chuỗi.
4. **Đảo ngược chuỗi**:

- Duyệt chuỗi từ cuối về đầu và in từng ký tự bằng ECALL 11.

5. **In chuỗi đã nhập**:
    - In chuỗi đã nhập sử dụng ECALL 4.
6. **In số lượng ký tự**:
    - In thông báo và số lượng ký tự đã nhập (lưu trong t0) bằng ECALL 1.
7. **Thoát chương trình**:
    - Sử dụng ECALL 10 để thoát chương trình.

Chương trình thực hiện nhập chuỗi, đảo ngược và in chuỗi cùng với số lượng ký tự.