

BÁO CÁO THỰC HÀNH GIỮA KỲ 2024.2

Họ và tên: Nguyễn Thành Duy

MSSV: 20235696

Bài tập được giao: A-9, B-3, C-12

Assignment A-9: Nhập số nguyên dương N có từ 2 chữ số trở lên. In ra màn hình chữ số nhỏ nhất của N.

Code:

```
.data
message_1: .asciz "Nhap so nguyen duong N co tu 2 chu so tro len: "
message_2: .asciz "Chu so nho nhat cua N: "
message_3: .asciz "Nhap sai, vui long thuc hien lai !"
space: .ascii " "
.text
main:
#----- Yeu cau nguoi dung nhap vao mot chu so -----
# In ra message_1
li a7, 4
la a0, message_1
ecall
# Nhap N
li a7, 5
ecall
mv s0, a0 # s0 = N
# Kiem tra chu so nhap vao tu nguoi dung
bgt s0, zero, find_min
# Neu N < 0 thi in ra message_3
li a7, 55
la a0, message_3
li a1, 0
ecall
j end_main
#----- Tim chu so nho nhat co trong N -----
find_min:
li a2, 10 # a2 = 10
li s2, 0x7fffffff # s2 duoc gan gia tri duong lon nhat
loop:
beqz s0, exit # Lap cho toi khi s0 = 0
rem s1, s0, a2 # s1 = s0 % 10
div s0, s0, a2 # s0 = s0 / 10
```

```

bge s2, s1, update_min # s2 > s1 thì cap nhat
j loop
# Kiem tra va cap nhat gia tri nho nhat
update_min:
mv s2, s1 # Cap nhat gia tri min
j loop
# ----- In ra chu so nho nhat va ket thuc chuong trinh -----
exit:
# In ra message_2
li a7, 4
la a0, message_2
ecall
# In ra chu so nho nhat
li a7, 1
mv a0, s2
ecall
# Ket thuc chuong trinh
end_main:
li a7, 10
ecall

```

Phân tích:

- Chương trình yêu cầu nhập số nguyên dương có từ 2 chữ số trở lên.
- +) Nếu người dùng nhập chữ số âm hoặc số 0 thì báo lỗi và yêu cầu thực hiện lại
- +) Nếu người dùng nhập số có 1 chữ số dương thì chữ số nhỏ nhất của số đó chính là số đó.

Giải thích:

Ý tưởng giải quyết:

- Nhận vào số nguyên N
- Duyệt qua từng chữ số N, giữ lại chữ số nhỏ nhất
- In ra kết quả.

Giải thích đoạn chương trình:

- Đoạn chương trình sau nhận số N từ người dùng.

```

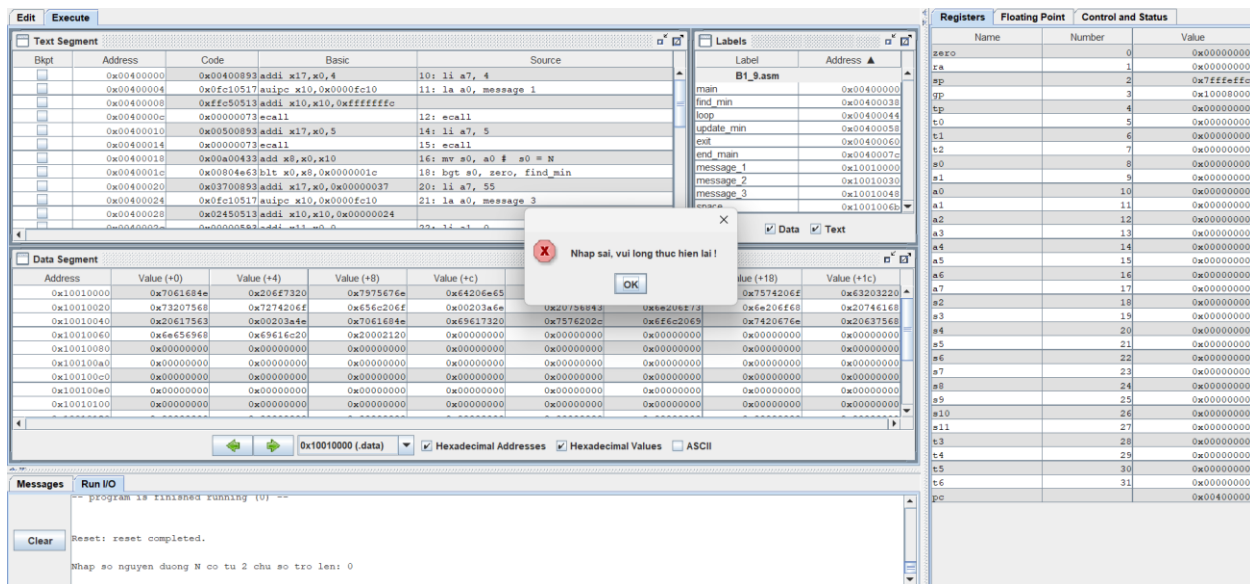
main:
#----- Yeu cau nguoi dung nhap vao mot chu so -----
# In ra message_1
li a7, 4
la a0, message_1
ecall
# Nhap N
li a7, 5
ecall
mv s0, a0 # s0 = N
# Kiem tra chu so nhap vao tu nguoi dung
bgt s0, zero, find_min
# Neu N < 0 thi in ra message_3
li a7, 55
la a0, message_3
li a1, 0
ecall
j end_main

```

Nếu như người dùng nhập sai yêu cầu đề bài ví dụ như nhập số âm hoặc số 0 thì sẽ báo lỗi và nhảy tới nhãn kết thúc chương trình:

The screenshot shows a debugger window with the following components:

- Text Segment:** A table showing assembly instructions with columns for Bkpt, Address, Code, Basic, and Source. Instructions include `addi x17,x0,4`, `sw x10,0x0000fc10`, `addi x10,x10,0xffffffffc`, `ecall`, `addi x17,x0,5`, `ecall`, `add x8,x0,x10`, `blt x0,x8,0x0000001c`, `addi x17,x0,0x00000037`, `sw x10,0x0000fc10`, `addi x10,x10,0x00000024`, and `addi x11,x0,0`.
- Data Segment:** A table showing memory addresses and their corresponding values in hexadecimal. The values are mostly zeros, with some non-zero values like `0x7061684e`, `0x73207568`, `0x20617563`, `0x6e556968`, `0x9616c20`, `0x20002120`, `0x2074206f`, `0x656c206f`, `0x7061684e`, `0x69617320`, `0x7576202c`, `0x666c2069`, `0x7420676e`, and `0x20637568`.
- Registers:** A table showing the state of registers. The `zero` register is set to 0. Other registers like `ra`, `sp`, `gp`, `tp`, `t0`, `t1`, `t2`, `a0`, `a1`, `a2`, `a3`, `a4`, `a5`, `a6`, `a7`, `s0`, `s1`, `s2`, `s3`, `s4`, `s5`, `s6`, `s7`, `s8`, `s9`, `s10`, `s11`, `s12`, `s13`, `s14`, `s15`, `s16`, `s17`, `s18`, `s19`, `s20`, `s21`, `s22`, `s23`, `s24`, `s25`, `s26`, `s27`, `s28`, `s29`, `s30`, `s31`, and `pc` are also listed.
- Messages:** A section at the bottom showing a message: "Reset: reset completed." and another message: "Nhập số nguyên dương N có từ 2 chu số trở lên: -543".



- Đoạn chương trình sau để tìm chữ số nhỏ nhất bằng cách duyệt qua từng chữ số của N kiểm tra và cập nhật lại chữ số nhỏ nhất.

```
#----- Tìm chu so nho nhat co trong N -----
find_min:
li a2, 10 # a2 = 10
li s2, 0x7fffffff # s2 duoc gan gia tri duong lon nhat
loop:
beqz s0, exit # Lap cho toi khi s0 = 0
rem s1, s0, a2 # s1 = s0 % 10
div s0, s0, a2 # s0 = s0 / 10
bge s2, s1, update_min # s2 > s1 thi cap nhat

j loop
# Kiem tra va cap nhat gia tri nho nhat
update_min:
mv s2, s1 # Cap nhat gia tri min
j loop
# ----- In ra chu so nho nhat va ket thuc chuong trinh -----
exit:
```

- Đoạn chương trình sau in ra chữ số nhỏ nhất của số N:

```

# ----- In ra chu so nho nhat va ket thuc chuong trinh -----
exit:
# In ra message_2
li a7, 4
la a0, message_2
ecall

# In ra chu so nho nhat
li a7, 1
mv a0, s2
ecall

# Ket thuc chuong trinh
end_main:
li a7, 10
ecall

```

Kết quả:

- Khi nhập số: 58279321
- Kết quả mong muốn: 1

The screenshot displays a debugger window with three main panes:

- Text Segment:** Shows assembly instructions with their addresses and codes. The instructions include loading values into registers, moving values, and calling functions.
- Labels:** A list of labels used in the code, such as `main`, `find_min`, `loop`, `update_min`, `exit`, `end_main`, `message_1`, `message_2`, `message_3`, and `space`.
- Registers:** A table showing the state of various registers. The `zero` register is highlighted, showing a value of `0x00000000`.

At the bottom, the **Messages** pane shows the output of the program:

```

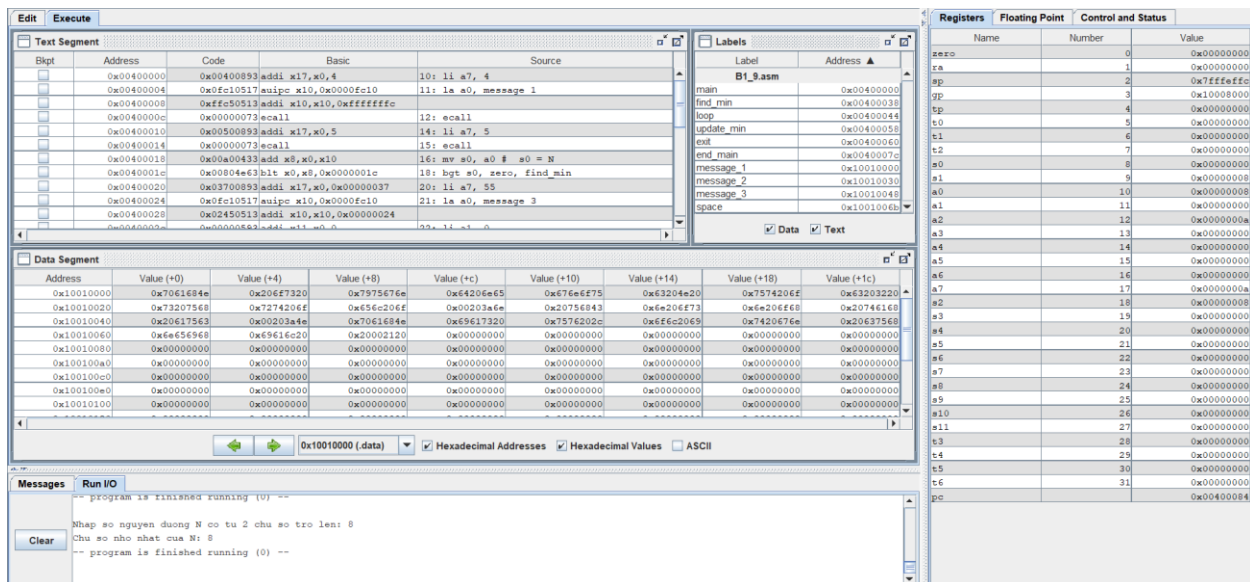
-- program is finished running (0) --
Nhập số nguyên dương N có từ 2 chu số trở lên: 58279321
Chu số nhỏ nhất của N: 1
-- program is finished running (0) --

```

Kết quả chương trình đúng như mong muốn.

Khi nhập chữ số có 1 chữ số thì chữ số nhỏ nhất là chính nó.

VD: N = 8 -> Chữ số nhỏ nhất của N: 8



Assignment B-3: Nhập mảng số nguyên từ bàn phím. In ra màn hình cặp phần tử liên kề có tích lớn nhất. Ví dụ: Nhập mảng [3, 6, -2, -5, 7, 3], cặp phần tử liên kề có tích lớn nhất là 7 và 3.

Code:

```
.data
A: .space 100
message_1: .asciz "Nhap so phan tu cua mang A: "
message_2: .asciz "Cap phan tu lien ke co tích lon nhat: "
space: .asciz " "
.text
main:
# In ra yeu cau nhap mang A
    li a7, 4
    la a0, message_1
    ecall

# Nhap so phan tu cua mang A
    li a7, 5
    ecall
    mv s0, a0 # Luu so phan tu vao s0

# Nhap mang A
    la s1, A # Con tro dau mang
    li t0, 0 # Bien dem
```

loop:

bge t0, s0, done # Neu nhap du so phan tu thi thoat

li a7, 5 # Nhap so nguyen

ecall

slli s2, t0, 2 # $t0 * 4$

add s2, s2, s1 # Dia chi phan tu t0 trong A

sw a0, 0(s2) # Lưu giá trị vào mảng

addi t0, t0, 1 # Tang bien dem

j loop

done:

Tim cap co tich lon nhat

li t0, 0 # Bien dem

li a5, -2147483648 # Gia tri tich nho nhat co the

li s2, 0 # Luu phan tu thu nhat

li s5, 0 # Luu phan tu thu hai

la s1, A # Con tro dau mang

find_max:

addi t1, t0, 1 # $t1 = t0 + 1$

bge t1, s0, exit # Neu duyet het thi thoat

lw a3, 0(s1) # Lay phan tu t0

lw a4, 4(s1) # Lay phan tu t0 + 1

mul s3, a3, a4 # Tinh tich

Neu tich lon hon a5, cap nhat gia tri

bge s3, a5, update

Cap nhat con tro va bien dem

next:

addi s1, s1, 4 # Di chuyen con tro den phan tu tiep theo

addi t0, t0, 1 # Tang bien dem

j find_max

update:

mv a5, s3 # Cap nhat tich lon nhat

mv s2, a3 # Cap nhat phan tu thu nhat

mv s5, a4 # Cap nhat phan tu thu hai

```
j next
```

```
exit:
```

```
# In kết quả và kết thúc chương trình
```

```
li a7, 4
```

```
la a0, message_2
```

```
ecall
```

```
li a7, 1
```

```
mv a0, s2 # In phần tử thứ nhất
```

```
ecall
```

```
li a7, 4
```

```
la a0, space
```

```
ecall
```

```
li a7, 1
```

```
mv a0, s5 # In phần tử thứ hai
```

```
ecall
```

```
li a7, 10 # Kết thúc chương trình
```

```
ecall
```

Phân tích:

- Nhập số phần tử của mảng A
- Duyệt qua từng phần tử i và tính tích phần tử $A[i]$ và phần tử $A[i + 1]$ nếu lớn hơn max hiện tại thì cập nhật max và 2 phần tử kề.

Giải thích:

- Đoạn chương trình sau in ra message_1 và yêu cầu người dùng nhập số phần tử của mảng A

```
main:
```

```
# In ra yêu cầu nhập mảng A
```

```
li a7, 4
```

```
la a0, message_1
```

```
ecall
```

```
# Nhập số phần tử của mảng A
```

```
li a7, 5
```

```
ecall
```

```
mv s0, a0 # Lưu số phần tử vào s0
```


- Nhập từng phần tử và lưu vào mảng A tới khi bằng số phần tử đã nhập ở trên:

```
# Nhập mảng A
la s1, A # Con trỏ đầu mảng
li t0, 0 # Biến đếm

loop:
    bge t0, s0, done # Nếu nhập đủ số phần tử thì thoát

    li a7, 5 # Nhập số nguyên
    ecall

    slli s2, t0, 2 # t0 * 4
    add s2, s2, s1 # Địa chỉ phần tử t0 trong A
    sw a0, 0(s2) # Lưu giá trị vào mảng

    addi t0, t0, 1 # Tăng biến đếm
    j loop
```

- Đoạn chương trình sau duyệt qua từng phần tử và tính tích phần tử đó và phần tử kế tiếp rồi cập nhật để tìm tích lớn nhất

```
done:
# Tìm cặp có tích lớn nhất
li t0, 0 # Biến đếm
li a5, -2147483648 # Giá trị tích nhỏ nhất có thể
li s2, 0 # Lưu phần tử thu nhất
li s5, 0 # Lưu phần tử thu hai

la s1, A # Con trỏ đầu mảng

find_max:
    addi t1, t0, 1 # t1 = t0 + 1
    bge t1, s0, exit # Nếu duyệt hết thì thoát

    lw a3, 0(s1) # Lấy phần tử t0
    lw a4, 4(s1) # Lấy phần tử t0 + 1

    mul s3, a3, a4 # Tính tích

# Nếu tích lớn hơn a5, cập nhật giá trị
    bge s3, a5, update

# Cập nhật con trỏ và biến đếm
next:
    addi s1, s1, 4 # Di chuyển con trỏ đến phần tử tiếp theo
    addi t0, t0, 1 # Tăng biến đếm
    j find_max
```

```

update:
    mv a5, s3 # Cap nhat tich lon nhat
    mv s2, a3 # Cap nhat phan tu thu nhat
    mv s5, a4 # Cap nhat phan tu thu hai
    j next

```

- In ra cặp phần tử có tích lớn nhất:

```

exit:
# In kết quả và kết thúc chương trình
    li a7, 4
    la a0, message_2
    ecall

    li a7, 1
    mv a0, s2 # In phan tu thu nhat
    ecall

    li a7, 4
    la a0, space
    ecall

    li a7, 1
    mv a0, s5 # In phan tu thu hai
    ecall

    li a7, 10 # Ket thuc chuong trinh
    ecall

```

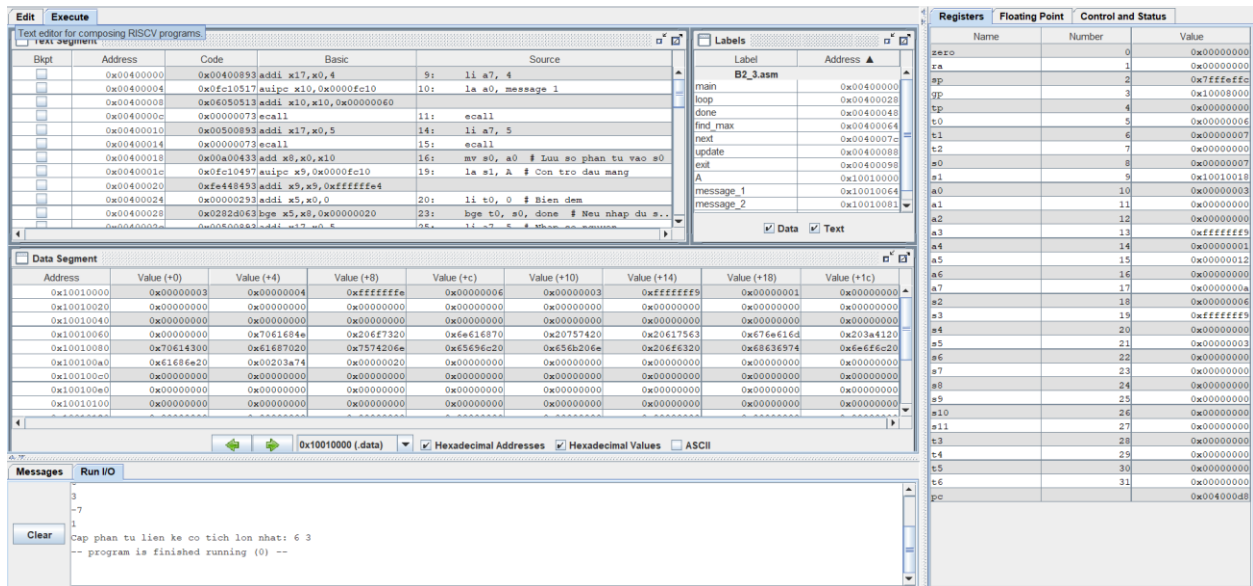
Kết quả:

Input: n = 7

3 4 -2 6 3 -7 1

Kết quả mong muốn: 6 3

Kết quả khi chạy chương trình:



Đúng như mong muốn: 6 3

Assignment C-12: Nhập vào 2 chuỗi ký tự A và B. In ra màn hình các ký tự chữ cái thường chỉ có trong A và không có trong B.

Code:

```

.data
A: .space 100
B: .space 100
message_1: .asciz "INPUT: \n"
message_2: .asciz "OUTPUT: \n"
A_input: .asciz "Nhập vào A: "
B_input: .asciz "Nhập vào B: "
space: .ascii " "

.text
main:

# INPUT

li a7, 4
la a0, message_1
ecall

#-----Nhập vào 1-----
# In ra thông báo nhập vào A
la a0, A_input
ecall

# Nhập vào A:
li a7, 8

```

```

la a0, A
li a1, 100
ecall
#-----Nhap xau 2-----
# In ra thong bao nhap xau B
li a7, 4
la a0, B_input
ecall
# Nhap xau B:
li a7, 8
la a0, B
li a1, 100
ecall

#-----

# OUTPUT
li a7, 4
la a0, message_2
ecall

la s1, A # Lay dia chi xau A
li s10, 10
check_A:
lb t0, 0(s1) # Lay gia tri tai dia chi s1
beq t0, s10, done # Neu gia tri la ky tu xuong dong tuc da duyet xong xau
beqz t0, done # Neu gia tri la ky tu rong tuc da duyet xong xau
li t1, 'a'
li t2, 'z'
blt t0, t1, next_A # Neu nhu nam ngoai a-z thi chuyen toi phan tu tiep theo
bgt t0, t2, next_A
# Kiem tra ky tu co trong xau B
la s2, B # Lay dia chi xau B
check_B:
lb t3, 0(s2) # Lay gia tri tai dia chi s2
beq t0, t3, next_A # Neu co bo qua xet ky tu dang sau
beq t3, s10, print # Neu gap ky tu xuong dong thi da duyet het xau
beqz t3, print # Neu duyet het ma khong thay thi in ra man hinh
addi s2, s2, 1 # s2 = s2 + 1
j check_B
next_A:
addi s1, s1, 1 # s1 = s1 + 1
j check_A
# In ky tu thuong khong xuat hien trong B

```

```

print:
li a7, 11
mv a0, t0
ecall
# In dấu cách
li a7, 4
la a0, space
ecall

j next_A
# Kết thúc chương trình
done:
li a7, 10
ecall

```

Phân tích:

- Nhập vào 2 chuỗi A, B
- Lặp qua từng ký tự của chuỗi A, với mỗi ký tự duyệt qua chuỗi B nếu như ký tự thường của A nằm trong B thì xét tới ký tự tiếp theo còn duyệt hết chuỗi B mà không thấy thì in ra màn hình.

Giải thích:

- Đoạn chương trình sau yêu cầu người dùng nhập chuỗi A, B

```

# INPUT

li a7, 4
la a0, message_1
ecall
#-----Nhập chuỗi 1-----
# In ra thông báo nhập chuỗi A
la a0, A_input
ecall
# Nhập chuỗi A:
li a7, 8
la a0, A
li a1, 100
ecall
#-----Nhập chuỗi 2-----
# In ra thông báo nhập chuỗi B
li a7, 4
la a0, B_input
ecall
# Nhập chuỗi B:
li a7, 8
la a0, B
li a1, 100
ecall

```

- Xét từng ký tự của mảng A, ứng với mỗi phần tử duyệt qua mảng B để kiểm tra ký tự thường đó có nằm trong mảng B không

```
# Xét các ký tự
li a7, 4 # In ra message_2
la a0, message_2
ecall

la s1, A # Lay dia chi xau A
li s10, 10
check_A:
lb t0, 0(s1) # Lay gia tri tai dia chi s1
beq t0, s10, done # Neu gia tri la ky tu xuong dong tuc da duyet xong xau
beqz t0, done # Neu gia tri la ky tu rong tuc da duyet xong xau
li t1, 'a'
li t2, 'z'
blt t0, t1, next_A # Neu nhu nam ngoai a-z thi chuyen toi phan tu tiep theo
bgt t0, t2, next_A
# Kiem tra ky tu co trong xau B
la s2, B # Lay dia chi xau B
check_B:
lb t3, 0(s2) # Lay gia tri tai dia chi s2
beq t0, t3, next_A # Neu co bo qua xet ky tu dang sau
beq t3, s10, print # Neu gap ky tu xuong dong thi da duyet het xau
beqz t3, print # Neu duyet het ma khong thay thi in ra man hinh
addi s2, s2, 1 # s2 = s2 + 1
j check_B
next_A:
addi s1, s1, 1 # s1 = s1 + 1
j check_A
```

- Với các phần tử thuộc xâu A mà không thuộc xâu B thì in ra màn hình:

```
# OUTPUT
# In ky tu thuong khong xuat hien trong B
print:
li a7, 11
mv a0, t0
ecall
# In dau cach
li a7, 4
la a0, space
ecall

j next_A
# Ket thuc chuong trinh
done:
li a7, 10
ecall
```

Kết quả:

INPUT: A: hanoiUNIVERCITY

B: bachKhoahaNOI

Ta thấy ký tự n, i của xâu A không có trong xâu B nên output mong muốn là: n i

Kết quả khi chạy chương trình:

The screenshot displays a debugger window with three main panes. The top-left pane shows assembly code with columns for Bkpt, Address, Code, Basic, and Source. The top-right pane shows a list of labels with their addresses. The bottom-left pane shows a data segment with columns for Address, Value (+0), Value (+4), Value (+8), Value (+c), Value (+10), Value (+14), Value (+18), and Value (+1c). The bottom-right pane shows the registers, floating point, and control and status.

Assembly Code:

Bkpt	Address	Code	Basic	Source
	0x00400000	0x00400893	addi w17,w0,4	14: li a7, 4
	0x00400004	0x0fc10517	auipc w10,0x0000fc10	15: la a0, message_1
	0x00400008	0x0e450513	addi w10,w10,0x000000c4	
	0x0040000c	0x00000073	ecall	16: ecall
	0x00400010	0x0fc10517	auipc w10,0x0000fc10	19: la a0, A input
	0x00400014	0x0cb50513	addi w10,w10,0x000000cb	
	0x00400018	0x00000073	ecall	20: ecall
	0x0040001c	0x00800893	addi w17,w0,8	22: li a7, 8
	0x00400020	0x0fc10517	auipc w10,0x0000fc10	23: la a0, A
	0x00400024	0x0e050513	addi w10,w10,0xffffffe0	
	0x00400028	0x0e400593	addi w11,w0,0x00000064	24: li a1, 100
	0x00400030	0x00000073	ecall	25: ecall

Labels:

Label	Address
B2_lasm	
main	0x00400000
check_A	0x00400070
check_B	0x00400054
nest_A	0x004000ac
print	0x004000b4
done	0x004000d4
A	0x10010000
B	0x100100e4
message_1	0x100100c8
message_2	0x100100d1

Data Segment:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0xeff6e6168	0x494e5565	0x43524556	0x0a555445	0x00000000	0x00000000	0x00000000	0x00000000
0x10010004	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010008	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001000c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010010	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010014	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010018	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001001c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010024	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010028	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001002c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010030	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010034	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010038	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001003c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010044	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010048	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001004c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010050	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010054	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010058	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001005c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010064	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010068	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001006c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010070	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010074	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010078	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001007c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010084	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010088	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001008c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010090	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010094	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010098	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001009c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100ac	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100b0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100b4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100b8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100bc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100cc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100d0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100d4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100d8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100dc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100ec	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100f0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100f4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100f8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100fc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Registers:

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffefc
gp	3	0x10000000
tp	4	0x00000000
t0	5	0x0000000a
t1	6	0x00000061
t2	7	0x0000007a
a0	8	0x00000000
a1	9	0x100100e4
a2	10	0x100100c8
a3	11	0x00000064
a4	12	0x00000000
a5	13	0x00000000
a6	14	0x00000000
a7	15	0x00000000
a8	16	0x00000000
a9	17	0x0000000a
a10	18	0x10010071
a11	19	0x00000000
a12	20	0x00000000
a13	21	0x00000000
a14	22	0x00000000
a15	23	0x00000000
a16	24	0x00000000
a17	25	0x00000000
a18	26	0x00000000
a19	27	0x00000000
a20	28	0x0000000a
a21	29	0x00000000
a22	30	0x00000000
a23	31	0x00000000
pc		0x004000d4

Messages:

```
-- program is finished running (0) --  
INPUT:  
Nhap xau A: hanoiUNIVERSITY  
Nhap xau B: bachKhoahaNOI  
OUTPUT:  
n i
```

Kết quả như mong muốn.