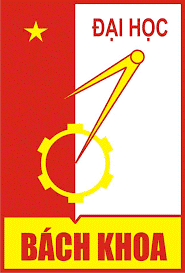
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



Mini project

**Kiến trúc máy tính**

**Nhóm: Thành Viên: Nguyễn Công Việt**

**Nguyễn Viết Trường**

# **Project 7:**

**Sinh viên thực hiện :Nguyễn Công Việt**

Some people are standing in a row in a park. There are trees between them which cannot be moved. Your task is to rearrange the people by their heights in a non-descending order without moving the trees.Example: For a = [-1, 150, 190, 170, -1, -1, 160, 180], the output should be sortByHeight(a) = [-1, 150, 160, 170, -1, -1, 180, 190].

1. Phân tích cách thực hiện

* Phân tích đề bài
* Input là 1 mảng số nguyên hệ thập phân.
* Giá trị -1 biểu thị cho tree (cây).
* Output hiển thị ra màn hình mảng input đã được sắp xếp theo thứ tự tăng dần nhưng vẫn giữ nguyên vị tí của các phần tử có giá trị bằng -1 trong mảng input.
* Chương trình chính được chia thành các phần như sau:
* main: người dùng nhập vào số lượng phần tử của mảng input, xử lý các lỗi input.
* loop: người dùng nhập vào các phần tử của mảng input.
* find\_height: lấy các giá trị là chiều cao thật của con người (khác -1) trong mảng input và cho vào mảng height.
* sort\_height: sắp xếp mảng height theo thứ tự không giảm.
* replace: thay thế những giá trị đã được sắp xếp trong mảng height vào mảng input.
* output: in ra mảng đã được sắp xếp ra màn hình console.
* Cách chạy chương trình:
* Nhập số lượng phần tử của mảng input.
* Nhập các phần tử của mảng input.
* Output sẽ hiển thị dãy số đã sắp xếp theo yêu cầu của đề bài tại cửa sổ console.

1. Ý nghĩa của các thanh ghi được sử dụng

* Trong thủ tục main:
* $v0: mode InputDialogInt của syscall
  + - $a0: địa chỉ của message "Enter length of input array", sau khi gọi syscall thì $a0 sẽ lưu giá trị mà người dùng nhập vào
    - $t5: giá trị -1
    - $t6: giá trị -2
    - $t7: giá trị -3
    - $s6: giá trị của status trả về sau khi người dùng nhập từ bàn phím
    - $t8: độ dài của mảng input do người dùng nhập vào (đã kiểm tra hợp lệ)
    - $t0: index của phần tử đầu tiên trong mảng input
    - $t2: địa chỉ của mảng input
* Trong thủ tục loop:
  + $t1: kết quả của phép set less than giữa $t0 và $t8
  + $v0: mode InputDialogInt của syscall
  + $a0: địa chỉ của message "Enter element of input array", sau khi gọi syscall thì $a0 lưu giá trị mà người dùng nhập vào
  + $s6: giá trị của status trả về sau khi người dùng nhập từ bàn phím
  + $t0: index của phần tử kế tiếp trong mảng input
  + $t2: địa chỉ của phần tử kế tiếp trong mảng input
  + $a1: địa chỉ của mảng height
  + $s0: giá trị -1 thể hiện tree
  + $s1: số phần tử của mảng input
  + $s2: số phần tử của mảng height
* Trong thủ tục find\_height:
  + $t0: thể hiện chỉ số i tương ứng với phần tử đang xét của mảng input
  + $t2: thể hiện chỉ số j tương ứng với phần tử đang xét của mảng height
  + $s3: giá trị của phần tử thứ i trong mảng input
  + $t3: địa chỉ của phần tử tiếp theo trong mảng height
  + $t4: kết quả của phép set less than giữa chỉ số i của mảng input và chiều dài mảng input
  + $s2: số lượng phần tử hiện thời của mảng height (tăng dần trong thủ tục find\_height)
* Trong thủ tục sort\_height:
  + $t0: thể hiện chỉ số i tương ứng với phần tử đang xét của mảng input
  + $t1: thể hiện chỉ số j tương ứng với phần tử đang xét của mảng height
  + $t6: kết quả của phép set less than giữa chỉ số i và số lượng phần tử của mảng height
  + $t3: địa chỉ của phần tử tiếp theo trong mảng height
  + $s3: giá trị của phần tử height[j]
  + $s4: giá trị của phần tử height[j+1]
  + $t4: kết quả của phép set less than giữa height[j] và height[j+1]
* Trong thủ tục replace:
  + $t0: thể hiện chỉ số i tương ứng với phần tử đang xét của mảng input
  + $t2: thể hiện chỉ số j tương ứng với phần tử đang xét của mảng height
  + $t1: địa chỉ của phần tử thứ i trong mảng input
  + $s3: giá trị của phần tử thứ i trong mảng input
  + $t3: địa chỉ của phần tử thứ j trong mảng height
  + $s4: dùng để reset giá trị của phần tử thứ j tỏng mảng height về 0
* Trong thủ tục output:
  + $t1: kết quả của phép set less than giữa index i và độ dài mảng input
  + $v0: mode khi gọi syscall
  + $a1: mode message
  + $t2: địa chỉ của phần tử tiếp theo được xét trong mảng input

1. Source code

#======================================================================

# Project 7: sortByHeight

# Task: Some people are standing in a row in a park. There are trees between them which cannot be moved.

# Your task is to rearrange the people by their heights in a non- descending order without moving the trees.

# People can be very tall!

#

# Example: For a = [-1, 150, 190, 170, -1, -1, 160, 180]

# the output should be sortByHeight(a) = [-1, 150, 160, 170, -1, # -1, 180, 190].

# =====================================================================

# =====================================================================

# Summary

# @input: A (input array)

# @note: -1 represent trees

# height: store height of person in input array

# @idea: Input array

# 1. @find\_height: find height of person in A array => store result in 'height' array

# 2. @sort\_height: sort the 'height' array in ascending order

# 3. @replace: replace values in A array by a value in 'height' arrays

# =====================================================================

.data

# Input array

message: .asciiz "Enter length of input array: "

message1: .asciiz "Enter element of input array: "

message\_error1: .asciiz "Input type error! Please enter integer input!"

message\_error2: .asciiz "Cancel was choosen! Please enter integer input!"

message\_error3: .asciiz "No data was input! Please enter integer input!"

message\_done: .asciiz "Sorted array: "

blank: .asciiz " "

message\_error: .ascii "Error input type!!! Please enter integer number!"

A: .word 0:100 # input array

height: .word # array to store height of each person

.text

main:

# constructor Array

li $v0, 51 # InputDialogInt

la $a0, message

syscall

addi $t5, $zero, -1 # t5 = -1: input error

addi $t6, $zero, -2 # t6 = -2: cancel error

addi $t7, $zero, -3 # t7 = -3: no data input error

add $s6, $zero, $a1 # s6 = a1 = status value of InputDialogInt

# if s6 == 0: OK

# if s6 == -1: input error -> needhandle

beq $s6, $t5, output\_error1 # input type error

beq $s6, $t6, output\_error2 # cancel error

beq $s6, $t7, output\_error3 # no data input error

add $t8, $a0, $zero # $t8 = length of input array

li $t0, 0 # $t0 = i = 0

la $t2, A # load address of A array

loop:

# Get input array from user

slt $t1, $t0, $t8 # if i < length

beqz $t1, end\_loop

li $v0, 51 # InputDialogInt

la $a0, message1

syscall

add $s6, $zero, $a1 # s6 = a1 = status value of InputDialogInt

# if s6 == 0: OK

# if s6 == -1: input error -> need handle

beq $s6, $t5, error1 # input type error

beq $s6, $t6, error2 # cancel error

beq $s6, $t7, error3 # no data input error

sw $a0, 0($t2) # save value of element in A[i]

addi $t2, $t2, 4

addi $t0, $t0, 1

j loop

end\_loop:

la $a0, A # load address of A array

la $a1, height # load address of height array

addi $s0, $zero, -1 # -1: represent tree in 'A' array

add $s1, $zero, $t8 # n: length of 'A' array

addi $s2, $zero, 0 # m: lenght of 'height' array

j find\_height

after\_find\_height:

j sort\_height

after\_sort:

j replace

after\_replace:

li $t0, 0 # $t0 = i = 0

la $t2, A # load address of A array

li $v0, 4

la $a0, message\_done # print stringn output

syscall

output:

loop\_output:

slt $t1, $t0, $t8 # if i < length

beqz $t1, end\_loop\_output

li $v0, 1 # InputDialogInt

lw $a0, 0($t2) # save value of element in A[i]

syscall

li $v0, 4

la $a0, blank # print blank

syscall

addi $t2, $t2, 4 # next element in A

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

j loop\_output

end\_loop\_output:

li $v0, 10 # terminate

syscall

output\_error1:

li $v0, 55

li $a1, 2 # warning message

la $a0, message\_error1

syscall

j main

output\_error2:

li $v0, 55

li $a1, 2 # warning message

la $a0, message\_error2

syscall

j main

output\_error3:

li $v0, 55

li $a1, 2 # warning message

la $a0, message\_error3

syscall

j main

error1:

li $v0, 55

li $a1, 2 # warning message

la $a0, message\_error1

syscall

j loop

error2:

li $v0, 55

li $a1, 2 # warning message

la $a0, message\_error2

syscall

j loop

error3:

li $v0, 55

li $a1, 2 # warning message

la $a0, message\_error3

syscall

j loop

end\_output:

end\_main:

#-----------------------------------------------------------

# 1. @find\_height: find height of person in A array => store result in 'height' array

# @input: A array (input array)

# @output: 'height' array

#-----------------------------------------------------------

find\_height:

# initialize i, j

addi $t0, $zero, 0 # i = 0

addi $t2, $zero, 0 # j = 0

fh\_loop:

sll $t1, $t0, 2 # $t1 = 4\*i

add $t1, $t1, $a0 # $t1 stores address of A[i]

lw $s3, 0($t1) # load value of A[i]

beq $s3, $s0, fh\_continue # if A[i] == -1 => continue(ignore tree)

sll $t3, $t2, 2 # $t3 = 4\*j

add $t3, $t3, $a1 # $t3 store address of height[j]

sw $s3, 0($t3) # store value of height[j] in $s3

addi $t2, $t2, 1 # j = j + 1

addi $s2, $s2, 1 # m = m + 1 (increase number of elements in 'height' array)

fh\_continue:

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

slt $t4, $t0, $s1 # if i < n => True: return 1; False: return 0

bne $t4, $zero, fh\_loop

fh\_end\_loop:

j after\_find\_height

#### Sort 'height' array in ascending order using BubbleSort

#-------------------------------------------------------------------------

# 2. @sort\_height: Sort 'height' array in ascending order

# @input: height - random order

# @output: height - sorted in ascending order

#-------------------------------------------------------------------------

sort\_height:

# Initialize index i of loop\_1 to 0

addi $t0, $zero, 0 # i = 0

loop\_1:

# Initialize index j of loop\_2 to 0

addi $t1, $zero, 0 # j = 0

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

sub $t2, $s2, $t0 # m - i - 1

# If i < m - 1

slt $t6, $t0, $s2

beq $t6, $zero, end\_loop\_1

loop\_2:

# If j < m - i - 1

slt $t5, $t1, $t2 # j < m - i - 1: True return 1; else return 0

beq $t5, $zero, end\_loop\_2

sll $t3, $t1, 2 # $t3 = 4\*j

add $t3, $t3, $a1 # $t3 stores address of height[j]

lw $s3, 0($t3) # load value of height[j] to $s3

lw $s4, 4($t3) # load value of height[j+1] to $s4

if:

slt $t4, $s3, $s4 # if A[j] < A[j+1] => True: return 1

# False: return 0

bne $t4, $zero, end\_if

# If height[j] > height[j + 1] => Swap height[j] and height[j+1]

sw $s4, 0($t3)

sw $s3, 4($t3)

end\_if:

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

j loop\_2

end\_loop\_2:

j loop\_1

end\_loop\_1:

j after\_sort

#-------------------------------------------------------------------------

# 3. @replace: replace values in A array by a value in 'height' array

# @input: A (input array)

# @output: A (sorted input array in ascending order)

# @note: Reset values of 'height' array = 0

#-------------------------------------------------------------------------

replace:

# Initialize i, j

addi $t0, $zero, 0 # i = 0

addi $t2, $zero, 0 # j = 0

i\_loop:

sll $t1, $t0, 2 # $t1 = 4\*i

add $t1, $t1, $a0 # $t1 stores address of A[i]

lw $s3, 0($t1) # load address of A[i]

beq $s3, $s0, i\_continue # if A[i] == -1 => continue

sll $t3, $t2, 2 # $t3 = 4\*j

add $t3, $t3, $a1 # $t3 stores address of height[j]

lw $s3, 0($t3) # load value of height[j]

sw $s3, 0($t1) # A[i] = height[j]

addi $s4, $zero, 0

sw $s4, 0($t3) # Reset value0 of height[j] = 0

addi $t2, $t2, 1 # j = j + 1

i\_continue:

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

slt $t4, $t0, $s1 # if i < n: True return: 1; False return: 0

bne $t4, $zero, i\_loop

i\_end\_loop:

j after\_replace

#-------------------------------------------------------------------------

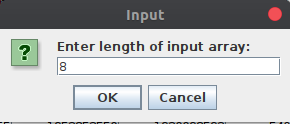
# END

#-------------------------------------------------------------------------

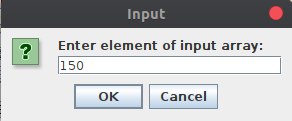
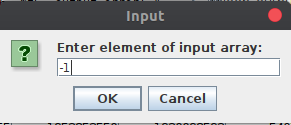
1. Kết quả:
2. Sắp xếp mảng đầu vào theo yêu cầu:

Đầu vào:

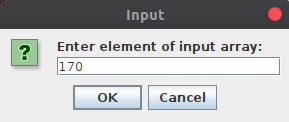
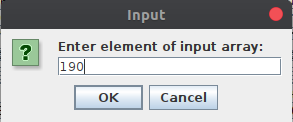
* Đầu tiên ta cần nhập số lượng phần tử của mảng: ở đây ta nhập là 8 (ảnh 1)
* Sau đó ta cần nhập các phần tử của mảng input. (lần lượt các ảnh 2 -> ảnh 9)



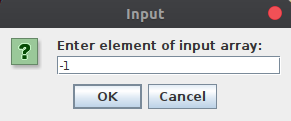
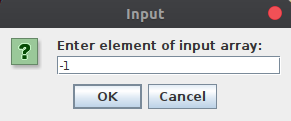
(Ảnh 1)



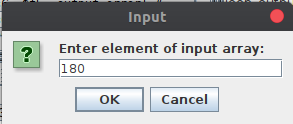
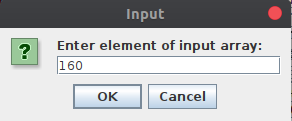
(Ảnh 2) (Ảnh 3)



(Ảnh 4) (Ảnh 5)

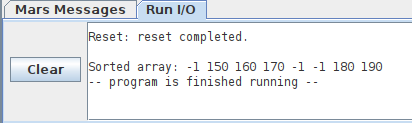


(Ảnh 6) (Ảnh 7)



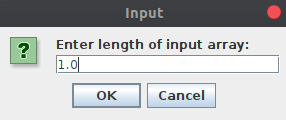
(Ảnh 8) (Ảnh 9)

Kết quả:

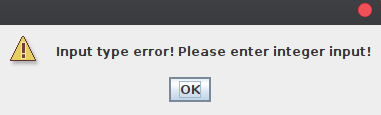


1. Một số ví dụ về xử lý lỗi input:

**Lỗi 1**: Nhập sai kiểu dữ liệu Integer -> Float

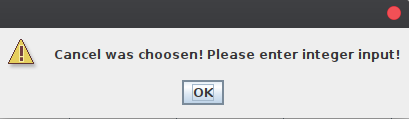


Chương trình báo lỗi và cho người dùng nhập lại:



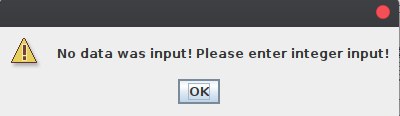
**Lỗi 2: Người dùng không nhập mà nhấn Cancel:**

Chương trình báo lỗi và cho người dùng nhập lại:



**Lỗi 3: Người dùng không nhập mà nhấn OK:**

Chương trình báo lỗi và cho người dùng nhập lại:



# **Project 19:**

**Sinh viên thực hiện : Nguyễn Viết Trường**

*Given some integer, find the maximal number you can obtain by deleting exactly one digit of the given number.*

*Example*

* *Forn = 152, the output should be 52*
* *Forn = 1001, the output should be 101*

**Ý tưởng:**

- Tìm và bỏ chữ số thứ i đầu tiên thỏa mãn N[i] < N[i + 1], trong trường hợp không tìm được thì loại bỏ chữ số cuối cùng. Ví dụ: 54345 có tại i = 3, N[i] = 3 < 4 = N[i + 1], ta bỏ chữ số 3, thu được 5445 là số lớn nhất thỏa mãn. 54321 không tìm được i, bỏ chữ số cuối cùng, 5432 là số lớn nhất tìm được.

- Trong trường hợp N là số âm, ta lấy giá trị tuyệt đối của N rồi tìm số nhỏ nhất thu được khi bỏ 1 chữ số (tìm và bỏ chữ số thứ i đầu tiên thỏa mãn N[i] > N[i+1], trong trường hợp không tìm được thì loại bỏ chữ số cuối cùng), rồi sau đó thêm lại dấu âm là nhận được kết quả cần tìm.

**Cách thực hiện:**

- Lưu các chữ số của N vào một mảng tối đa 10 phần tử bằng cách chia N cho 10 dần dần và lưu phần dư vào mảng, thứ tự các chữ số lưu trong mảng sẽ ngược với thứ tự các chữ số của N.

- Duyệt mảng theo chiều ngược (tức xuôi theo N) để tìm vị trí i cần xóa.

- Duyệt mảng theo chiều xuôi để ghép các chữ số thành số cần tìm (bỏ qua giá trị ở vị trí i).

**Ý nghĩa các thanh ghi được sử dụng:**

- $a0, $a1, $a2: Các tham số truyền vào hàm.

- $v0, $v1: Giá trị trả về của hàm.

- $s0: Chứa số N.

- $s1: Chứa địa chỉ của mảng.

- Các thanh ghi t: Các giá trị tạm thời, cụ thể nằm trong mã nguồn.

**Các chương trình con:**

**- PARSE\_INTEGER: Nhận tham số là N, địa chỉ mảng và trả về mảng chứa các chữ số của N cùng với vị trí kết thúc mảng**.

ý nghĩa thanh ghi trong hàm :

$a0 - Số nguyên đã cho

$a1 - Địa chỉ mảng đã khởi tạo

$v0 - Vị trí phần tử cuối cùng của mảng

$v1 - Mảng chứa các chữ số của số nguyên

$t0: 10

$t1: Iterator của $a1

$t2: Chứa giá trị của phần tử mà $t1 trỏ tới

**- FIND\_MAX: Nhận tham số là địa chỉ mảng, vị trí kết thúc mảng và trả về số lớn nhất ghép được.**

ý nghĩa thanh ghi trong hàm :

$a1 - Mảng chứa các phần tử là các chữ số của số nguyên đã cho

$a2 - Địa chỉ vị trí kết thúc mảng

$v0 - Giá trị lớn nhất nhận được khi xóa một chữ số

$v0: Giá trị lớn nhất tìm được và cũng là giá trị trả về

$t0: Iterator của mảng

$t1: Phần tử sau $t0

$t2: Giá trị của $t0

$t3: Giá trị của $t1

$t4: Ban đầu bằng 1, sau mỗi vòng lặp tăng lên 10 lần

$t5: value = 10

$t9: Vị trí chữ số cần xóa

**- FIND\_MIN: Nhận tham số là địa chỉ mảng, vị trí kết thúc mảng và trả về số nhỏ nhất ghép được (trong trường hợp N âm).**

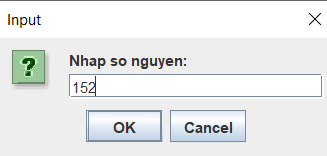
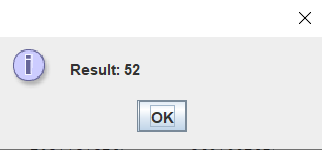
ý nghĩa thanh ghi trong hàm : tương tự FIND\_MAX nhưng ngược lại

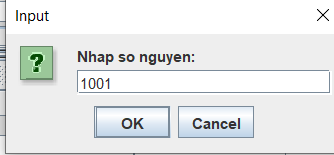
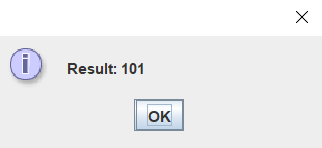
**Code:**

|  |
| --- |
| .data  Array: .word -1, -1, -1, -1, -1, -1, -1, -1, -1, -1  Message: .asciiz "Nhap so nguyen: "  Message1: .asciiz "Result: "  Message2: .asciiz "Khong hop le\n"  .text  li $v0, 51  la $a0, Message  syscall  addi $s0, $a0, 0  la $s1, Array  addi $t0, $0, 10  div $s0, $t0  mflo $t1  beqz $t1, ONE\_DIGIT\_NUMBER  bltz $s0, NEGATIVE\_INTEGER  addi $a0, $s0, 0  addi $a1, $s1, 0  jal PARSE\_INTEGER  addi $a1, $v1, 0  addi $a2, $v0, 0  jal FIND\_MAX  j EXIT  NEGATIVE\_INTEGER:  sub $s0, $0, $s0  addi $a0, $s0, 0  addi $a1, $s1, 0  jal PARSE\_INTEGER  addi $a1, $v1, 0  addi $a2, $v0, 0  jal FIND\_MIN  sub $v0, $0, $v0  j EXIT    EXIT:  addi $a1, $v0, 0  li $v0, 56  la $a0, Message1  syscall  li $v0, 10  syscall  ONE\_DIGIT\_NUMBER:  li $v0, 55  la $a0, Message2  syscall  li $v0, 10  syscall  PARSE\_INTEGER:  addi $t0, $0, 10  addi $t1, $a1, 0  WHILE:  beq $a0, $0, END\_WHILE  div $a0, $t0  mflo $a0  mfhi $t2  sw $t2, 0($t1)  addi $t1, $t1, 4  j WHILE  END\_WHILE:  addi $v0, $t1, -4  addi $v1, $a1, 0  jr $ra      FIND\_MAX:  addi $v0, $0, 0  addi $t0, $a2, 0 # $t0 = phần tử cuối mảng  addi $t1, $a2, -4  addi $t4, $0, 1  addi $t5, $0, 10  addi $t9, $a1, 0  # Duyệt mảng theo chiều ngược để tìm vị trí của chữ số cần xóa  # Vị trí i phù hợp là vị trí có A[i] < A[i+1]  WHILE2:  beq $t0, $a1, END\_WHILE2  lw $t2, 0($t0)  lw $t3, 0($t1)  bge $t2, $t3, NORMAL # A[i] >= A[i+1] -> Continue  addi $t9, $t0, 0 # Else -> Found  j END\_WHILE2    NORMAL:  addi $t0, $t0, -4  addi $t1, $t1, -4  j WHILE2    END\_WHILE2:  # Duyệt mảng theo chiều xuôi để ghép các chữ số lại  addi $t0, $a1, 0  addi $a2, $a2, 4  WHILE3:  beq $t0, $a2, END\_WHILE3  bne $t0, $t9, NORMAL1  addi $t0, $t0, 4 # Bỏ qua vị trí vừa tìm được  j WHILE3    NORMAL1:  lw $t2, 0($t0)  mult $t2, $t4  mflo $t2  add $v0, $v0, $t2 # Num = Num + $t2 \* $t4  mult $t4, $t5  mflo $t4 # $t4 = $t4 \* 10    addi $t0, $t0, 4  j WHILE3  END\_WHILE3:  jr $ra  FIND\_MIN:  addi $v0, $0, 0  addi $t0, $a2, 0  addi $t1, $a2, -4  addi $t3, $0, 0  addi $t4, $0, 1  addi $t5, $0, 10  addi $t9, $a1, 0  WHILE5:  beq $t0, $a1, END\_WHILE5  lw $t2, 0($t0)  lw $t3, 0($t1)  ble $t2, $t3, NORMAL2 # A[i] <= A[i+1] -> Continue  addi $t9, $t0, 0 # Else -> Found  j END\_WHILE5    NORMAL2:  addi $t0, $t0, -4  addi $t1, $t1, -4  j WHILE5    END\_WHILE5:  addi $t0, $a1, 0  addi $a2, $a2, 4  WHILE4:  beq $t0, $a2, END\_WHILE4  bne $t0, $t9, NORMAL3  addi $t0, $t0, 4  j WHILE4 # jump to WHILE3    NORMAL3:  lw $t2, 0($t0)  mult $t2, $t4  mflo $t2  add $v0, $v0, $t2 # Num = Num + $t2 \* $t4  mult $t4, $t5  mflo $t4 # $t4 = $t4 \* 10    addi $t0, $t0, 4  j WHILE4  END\_WHILE4:  jr $ra |

**Kết quả:**

1. khi nhập vào là một số dương:

** **

** **

1. khi nhập vào là một số âm:

