**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

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

**\*\*\*\*\*\*\*\*\*\*\*\*\***

****

**BÁO CÁO**

**THỰC HÀNH KIẾN TRÚC MÁY TÍNH**

Giáo viên hướng dẫn: **Lê Bá Vui**

Sinh viên thực hiện:

**Lương Đức Minh - MSSV: 20176821**

**Nguyễn Thanh Hà - MSSV: 20176742**

***Hà Nội, 05 – 2020***

**Mục lục**

Table of Contents

[Thông tin đề tài 3](#_Toc40695786)

[Đề số 18 4](#_Toc40695787)

[**Phương hướng giải quyết** 4](#_Toc40695788)

[**Source code (mã nguồn)** 6](#_Toc40695789)

[**Ý nghĩa các thanh ghi và hàm** 11](#_Toc40695790)

[**Kết quả demo** 11](#_Toc40695791)

[Đề số 19 13](#_Toc40695792)

[**Phương hướng giải quyết** 13](#_Toc40695793)

[**Source code (mã nguồn)** 14](#_Toc40695794)

[**Ý nghĩa các thanh ghi và hàm** 16](#_Toc40695795)

[**Kết quả demo** 17](#_Toc40695796)

[**Xử lý ngoại lệ** 18](#_Toc40695797)

# **Thông tin đề tài**

**Đề số 18:**

Define a word as a sequence of consecutive English letters. Find the longest word from the given string. Example For text = "Ready, steady, go!", the output should be longestWord(text) = "steady".

**Đề số 19:**

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

Example:

For n = 152, the output should be deleteDigit(n) = 52;

For n = 1001, the output should be deleteDigit(n) = 101.

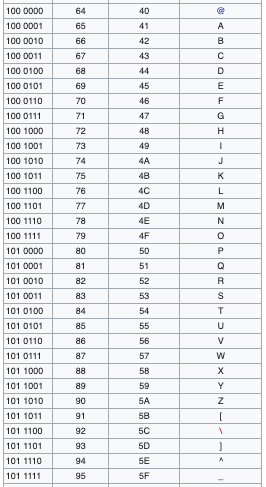
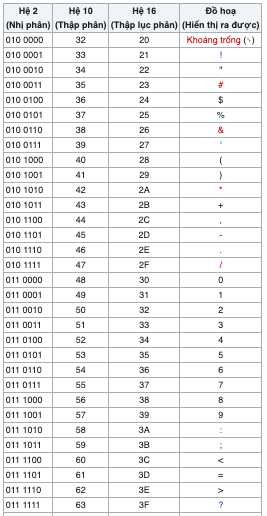
# **Đề số 18**

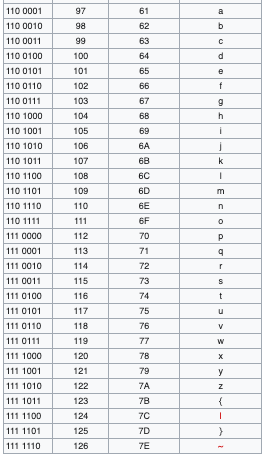
## **Phương hướng giải quyết**

**VD:** String:"Ready, steady, go!"

**Bước 1**: Đọc một kí tự và kiểm tra xem nó là chữ cái hay là số, dấu câu. (Ta chỉ cần quan tâm các chữ cái)

Dựa vào bảng mã ASCII (Nguồn: Wikipedia)





Xét theo hệ 16:

* Các chữ cái in hoa nằm trong khoảng 65 – 90
* Các chữ cái thường nằm trong khoảng từ 97 - 122

**Bước 2**: Tìm độ dài lớn nhất của các từ trong câu đã cho.

* Khai báo độ dài lớn nhất max = 0, duyệt lần lượt toàn bộ string.
* Tất cả các chữ cái đọc được cho đến khi gặp một kí tự đặc biệt hoặc một số thì được tính là một từ.
* Mỗi khi đọc được một từ mới, kiểm tra độ dài của từ này so với giá trị max hiện tại

Ta được độ dài max = 6

**Bước 3**: In ra màn hình những từ có độ dài bằng giá trị max trên.

* Duyệt lại string từ đầu
* Lưu từ vừa đọc được vào biến tạm thời
* Từ nào có độ dài = 6 thì in ra màn hình

**Bước 4**: Cho phép người dùng nhập nhiều lần.

* Sau khi kết thúc một chu trình, yêu cầu người dùng nhấn enter nếu muốn duyệt string mới.
* Nhấn các phím khác thì coi như kết thúc.

## **Source code (mã nguồn)**

|  |  |  |
| --- | --- | --- |
|  | | .data |
|  | | tmp: .space 32 # save current word that we have just read |
|  | | input: .space 100 # save user input |
|  | | start: .asciiz "\n\n----------- Let's start -----------" |
|  | | request: .asciiz "\nInput: " |
|  | | message1: .asciiz "\nThe longest length is: " |
|  | | message2: .asciiz "\nLongest words is: " |
|  | | newline: .asciiz "\n\t" |
|  | | contPrompt: .asciiz "\nPress ENTER to continue..." |
|  | |
|  | | .text |
|  | | li $v0, 4 # print\_string |
|  | | la $a0, start # start message |
|  | | syscall |
|  | |
|  | | init: li $v0, 4 # print\_string call number |
|  | | la $a0, request |
|  | | syscall |
|  | |
|  | | li $v0, 8 |
|  | | la $a0, input # pointer to string in memory |
|  | | la $a1, 100 |
|  | | syscall |
|  | |
|  | | li $s0, 0 # i = 0 run through all text |
|  | | li $s2, 0 # j = 0 is the index of tmp |
|  | | li $s4, 0 # current longest length of result |
|  | | li $s6, 0 # k = 0: index to clear tmp |
|  | | la $a2, tmp # load tmp address |
|  | |
|  | | main: jal checkNonAlpha # loop through input string |
|  | |
|  | | li $v0, 4 # print\_string |
|  | | la $a0, message1 # message to print longest length |
|  | | syscall |
|  | |
|  | | li $v0, 1 # print\_integer |
|  | | add $a0, $s4, $zero # longest length |
|  | | syscall |
|  | |
|  | | li $v0, 4 # print\_string |
|  | | la $a0, message2 # message to print longest words |
|  | | syscall |
|  | |
|  | | la $a1, input # load input address into $a1 |
|  | | j printWord # print all words with max length |
|  | |
|  | | exit: li $v0, 10 # syscall to terminate |
|  | | syscall |
|  | | ################################################################################# |
|  | | # checkNonAlpha: if current character is non-alphabet 🡪 we got a word # |
|  | | ################################################################################# |
|  | | checkNonAlpha: |
|  | | add $t4, $s0, $a0 # address of A[i] in $t4 |
|  | | lb $s1, 0($t4) # load value of A[i] |
|  | |  |
|  | | slti $t1, $s1, 65 # if ascii code is less than 48 |
|  | | bne $t1, $zero, checkLength # get a word |
|  | |
|  | | slti $t1, $s1, 91 # if ascii code is greater than 90 |
|  | | # and |
|  | | slti $t2, $s1, 97 # if ascii code is less than 97 |
|  | | slt $t3, $t1, $t2 |
|  | | bne $t3, $zero, checkLength # get a word |
|  | |  |
|  | | slti $t1, $s1, 123 # if ascii character is greater than 122 |
|  | | beq $t1, $zero, checkLength # get a word |
|  | |  |
|  | | addi $s0, $s0, 1 # i = i + 1 |
|  | | addi $s2, $s2, 1 # j = j + 1 |
|  | | j checkNonAlpha # go to checkNonAlpha |
|  | | ################################################################################# |
|  | | # checkLength: if current word's length > max 🡪 we have new max length # |
|  | | ################################################################################# |
|  | | checkLength: |
|  | | slt $t3, $s2, $s4 # if this word is not longer than current max |
|  | | bnez $t3, next # reset j and move to the next one |
|  | | add $s4, $zero, $s2 # else, we have new max length |
|  | | next: beq $s1, 10, done # if A[i] = '\n' -> done |
|  | | addi $s0, $s0, 1 # i = i + 1 |
|  | | li $s2, 0 # j = 0 |
|  | | j checkNonAlpha # proceed next character |
|  | | done: li $s0, 0 # reset i = 0 |
|  | | li $s2, 0 # reset j = 0 |
|  | | jr $ra # return to main |
|  | | ################################################################################# |
|  | | # printWord: second loop to find word for printing # |
|  | | ################################################################################# |
|  | | printWord: |
|  | | add $t4, $s0, $a1 # address of A[i] in $t4 |
|  | | lb $s1, 0($t4) # load value of A[i] |
|  | |
|  | | slti $t1, $s1, 65 # if ascii code is less than 65 |
|  | | bne $t1, $zero, compare # get a word |
|  | |
|  | | slti $t1, $s1, 91 # if ascii code is greater than 90 |
|  | | # and |
|  | | slti $t2, $s1, 97 # if ascii code is less than 97 |
|  | | slt $t3, $t1, $t2 |
|  | | bne $t3, $zero, compare # get a word |
|  | |
|  | | slti $t1, $s1, 123 # if ascii character is greater than 122 |
|  | | beq $t1, $zero, compare # get a word |
|  | |
|  | | add $t5, $s2, $a2 # address of tmp[i] in $t5 |
|  | | sb $s1, 0($t5) # store current character to tmp[i] |
|  | | addi $s0, $s0, 1 # i = i + 1 |
|  | | addi $s2, $s2, 1 # j = j + 1 |
|  | | j printWord # go to checkNonAlpha |
|  | | ################################################################################# |
|  | | # compare: if current word's length = max length -> print # |
|  | | ################################################################################# |
|  | | compare: |
|  | | beq $s2, $s4, print # if length = max 🡪 print |
|  | | j reset # else: reset j and move to the next one |
|  | | print: |
|  | | li $v0, 4 # print\_word |
|  | | la $a0, newline # line break and tab |
|  | | syscall |
|  | | li $v0, 4 # print\_string |
|  | | la $a0, tmp # word with max length |
|  | | syscall |
|  | | reset: |
|  | | addi $s0, $s0, 1 # i = i + 1 |
|  | | li $s6, 0 # k = 0 |
|  | | clear\_tmp: |
|  | | beq $s6, $s2, return # if k = j end |
|  | | add $t4, $s6, $a2 # loop through tmp string |
|  | | sb $0, 0($t4) # turn tmp[k] into 0 |
|  | | addi $s6, $s6, 1 # k = k + 1 |
|  | | j clear\_tmp # continue to clear |
|  | | return: |
|  | | li $s2, 0 # j = 0 |
|  | | beq $s1, 10, readEnter # if A[i] = '\n' 🡪 done. Enter to continue |
|  | | j printWord # proceed next character |
|  | | ################################################################################# |
|  | | # Enter to continue # |
|  | | ################################################################################# |
|  | | readEnter: # Ask user to press ENTER to continue |
|  | | li $v0, 4 |
|  | | la $a0, contPrompt |
|  | | syscall |
|  | | li $v0, 12 # sys code for readchar |
|  | | syscall |
|  | |
|  | | move $t0, $v0 |
|  | |
|  | | bne $t0, 10, exit # exit if not ENTER |
|  | |
|  | | j init # return to the start point |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

## **Ý nghĩa các thanh ghi và hàm**

* **Vùng nhớ .data**

|  |
| --- |
| tmp: Mảng lưu 1 từ vừa đọc được trong string. Tối đa 32 kí tự |
| input: Mảng tối đa 100 kí tự lưu string mà người dùng nhập vào. |

* **Thanh ghi**

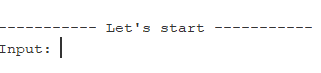
|  |
| --- |
| $s0 # biến i: duyệt qua tất cả chữ cái trong string. |
| $s2 # biến j: duyệt qua tất cả chữ cái trong 1 từ. |
| $s4 # độ dài lớn nhất hiện tại của 1 từ trong string. |
| $s6 # biến k: duyệt qua các chữ cái trong tmp để print hoặc reset. |
| $t0 # giả sử độ dài input tối đa = 32. |
| $a2 # địa chỉ bắt đầu vùng nhớ tmp (để lưu từng từ trong string). |
| $s1 # ascii code của chữ cái hiện tại. |
| $a0 # địa chỉ bắt đầu của string người dùng vừa nhập.  $a1 # copy địa chỉ của $a0 để chạy vòng lặp lần 2. |

* **Hàm**

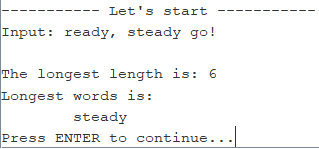
|  |
| --- |
| checkNonAlpha : Duyệt lần đầu để tìm độ dài lớn nhất trong string. |
| checkLength: Cập nhật độ dài lớn nhất nếu tìm được từ dài hơn. |
| printWord: Duyệt lại lần 2 để in ra màn hình. |
| compare: Từ vừa đọc được có dài bằng max không? Có thì print. |
| clear\_tmp: Reset vùng nhớ tmp để lưu từ mới, tránh tình trạng  tmp cũ = ‘gogogo’, từ vừa đọc = ‘minh’ 🡪 tmp = ‘minhgo’ |
| readEnter: Dùng syscall 12 yêu cầu người dùng ấn Enter để  tiếp tục. So sánh kí tự đó với ‘\n’ (ASCII: 10). |

## **Kết quả demo**

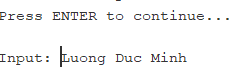
* Nhập vào một chuỗi bất kì:

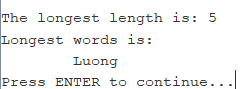


* In ra từ có độ dài lớn nhất và độ dài của nó.



* Ấn enter để nhập xâu tiếp theo:





* Kết thúc chương trình:



# **Đề số 19**

## **Phương hướng giải quyết**

**VD:** Cho số 245 tìm số lớn nhất khi xoá đi một chữ số.

245 -> tách thành 3 số 24, 45 và 25. -> Số lớn nhất là 45.

**Bước 1**: Tách số vừa nhập. VD 245 -> 2, 4, 5. Rồi lưu vào một mảng n phần tử.

Lấy ví dụ số 245: Đầu tiên cần tách 3 số vào 1 mảng

245 % 10 = 5 --> lưu số 5 vào mảng

245 / 10 = 24

24 % 10 = 4 --> lưu số 4 vào mảng

24 / 10 = 2

2 % 10 = 2 --> lưu số 2 vào mảng

**Bước 2**: Ghép từng cặp n-1 chữ số với nhau và lưu vào biến tạm tmp để xét max.

Mã giả:

result = 0

for i = 0 -> array\_size:

tmp = 0

for j = array\_size -> 0:

if (i != j): tmp = tmp \* 10 + array[j]

result = max(result, tmp)

VD: *i = 0* 🡪 *array[0] = 5*

*(j = 2) != i* 🡪 *tmp = 2*

*(j = 1) != i* 🡪 *tmp = 24* 🡪 *Tìm được cặp số đầu tiên*

**Bước 3**: In ra màn hình số lớn nhất vừa tìm được.

**Bước 4**:

* Cho phép người dùng nhập một số bất kì nhiều lần vào và in ra màn hình.
* Ấn enter để tiếp tục chương trình. Hoặc ấn 1 kí tự bất kì để thoát khỏi chương trình

## **Source code (mã nguồn)**

|  |  |
| --- | --- |
|  | .data |
|  | # input: .space 32 　　　　# Không dùng đến (nếu muốn xử lý lỗi thì yêu  # cầu người dùng nhập string) |
|  | message1: .asciiz "\n\n----------- Let's start -----------" |
|  | toolarge: .asciiz "\nThe number is too large" |
|  | message: .asciiz "\nInput: " |
|  | result: .asciiz "\nResult is: " |
|  | array: .word 0:10 　　　　# split num into individual digit  # and store in array |
|  | contPrompt: .asciiz "\nPress ENTER to continue..." |
|  | .text |
|  | li $v0, 4 # print start message |
|  | la $a0, message1 # pointer to start message in memory |
|  | syscall |
|  |  |
|  | start: li $v0, 4 # print input message |
|  | la $a0, message # pointer to input message in memory |
|  | syscall |
|  | li $v0, 5 # read an integer |
|  | syscall |
|  |  |
|  | li $t0, 10 # modulo 10 |
|  | add $s0, $zero, $v0 # load input to $s0 |
|  | la $s1, array # load array address to $s1 |
|  | li $s2, 0 # array index |
|  | main: |
|  | jal split |
|  | li $v0, 4 # print result message |
|  | la $a0, result # pointer to result message in memory |
|  | syscall |
|  | move $a0, $s3 # save result to $v0 for printing |
|  | li $v0, 1 # print result |
|  | syscall |
|  | j readEnter # enter to continue |
|  | exit: |
|  | li $v0, 10 # exit |
|  | syscall |
|  | # Tách số rồi lưu vào mảng |
|  | split: |
|  | slt $t2, $s2, $t0 # if i < 10, continue |
|  | beqz $t2, error # else: error int number too large (>10 digits) |
|  | div $s0, $t0 # 245 % 10 = 24(quotient) + 5(remainder) |
|  | mfhi $t1 # remainder |
|  | sll $t2, $s2, 2 # 4 \* i |
|  | add $t2, $s1, $t2 # load array[i] address |
|  | sw $t1, 0($t2) # save remainder to array[i] |
|  | mflo $s0 # quotient |
|  | beqz $s0, init # if quotient = 0 (2 % 10 = 0) -> next step |
|  | addi $s2, $s2, 1 # else i = i + 1 |
|  | j split |
|  | # After this step, array = [5, 4, 2] |
|  | # Array size = $s2 = 3 |
|  | error: |
|  | li $v0, 4 # print error string |
|  | la $a0, toolarge # pointer to error message in memory |
|  | syscall |
|  | j exit |
|  | # Tìm số lớn nhất |
|  | init: |
|  | li $s3, 0 # result |
|  | li $s4, 0 # loop index i = 0 |
|  | addi $t3, $s2, 1 # array\_size |
|  | loop\_i: |
|  | add $s5, $s2, $zero # nested loop index j = array\_size - 1 = 2 |
|  | slt $t2, $s4, $t3 # if i < array\_size continue |
|  | beqz $t2, end # else -> end |
|  | li $s6, 0 # tmp = 0 |
|  | j loop\_j # execute loop\_j |
|  | loop\_j: |
|  | slt $t2, $s5, $zero # if j < 0 |
|  | bnez $t2, check # exit loop\_j |
|  | bne $s4, $s5, update\_tmp # if i != j, update tmp |
|  | addi $s5, $s5, -1 # else j = j - 1 |
|  | j loop\_j |
|  | update\_tmp: |
|  | mult $s6, $t0 # tmp \* 10 |
|  | mflo $s6 # tmp = tmp \* 10 |
|  | sll $t2, $s5, 2 # j = j \* 4 |
|  | add $t2, $t2, $s1 # load array[j] address |
|  | lb $t2, 0($t2) # load array[j] value |
|  | add $s6, $s6, $t2 # tmp = tmp + array[j] |
|  | addi $s5, $s5, -1 # j = j - 1 |
|  | j loop\_j |
|  | check: slt $t2, $s3, $s6 # if result < tmp |
|  | bnez $t2, swap # result = tmp |
|  | addi $s4, $s4, 1 # else i = i + 1 |
|  | j loop\_i |
|  | swap: add $s3, $zero, $s6 # result = tmp |
|  | addi $s4, $s4, 1 # i = i + 1 |
|  | j loop\_i |
|  | end: |
|  | jr $ra # return to main |
|  | # Enter to continue |
|  | readEnter: # Ask user to press ENTER to continue |
|  | li $v0, 4 |
|  | la $a0, contPrompt |
|  | syscall |
|  | li $v0, 12 # sys code for readchar |
|  | syscall |
|  |  |
|  | move $t0, $v0 |
|  |  |
|  | bne $t0, 10, exit # exit if not ENTER |
|  |  |
|  | j start # return to the start point |

## **Ý nghĩa các thanh ghi và hàm**

* **Vùng nhớ .data**

|  |
| --- |
| array: Khởi tạo mảng 10 phần tử để lưu các kí tự số khi split |
| # tối đa 10 kí tự vì max unsigned int = 4294967295  # Tham khảo cách khởi tạo array ở link sau:  # <http://pages.cs.wisc.edu/~smoler/cs354/onyourown/arrays.html> |

* **Thanh ghi**

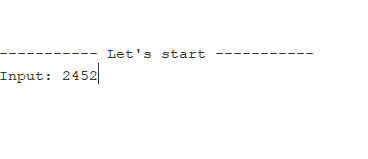
|  |
| --- |
| $t0 # lưu số 10, để thực hiện phép modulo |
| $s0 # số mà người dùng nhập từ bàn phím |
| $s1 # địa chỉ bắt đầu của mảng (lưu các kí tự của 1 số) |
| $s2 # chỉ số của mảng / số đếm |
| $s3 # kết quả |
| $t1 # phần dư (remainder) khi thực hiện phép chia modulo 10 |
| $s4 # index i của vòng lặp bên ngoài |
| $s5 # index j của vòng lặp bên trong  $s6 # giá trị biến tmp |

* **Hàm**

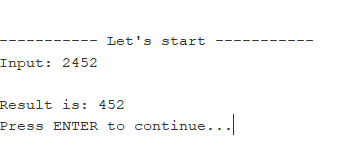
|  |
| --- |
| split: Tách số thành các kí tự rồi lần lượt lưu vào mảng  VD: 245 🡪 Array = [5, 4, 2]. |
| loop\_i: Vòng lặp ngoài cho mã giả ở bước 2. |
| Loop\_j: Vòng lặp trong cho mã giả ở bước 2. |
| update\_tmp: Cập nhật giá trị của cặp số vừa ghép được. |
| check: Kiểm tra giá trị tmp đó có lớn hơn max không? |
| swap: Nếu lớn hơn thì thay max = giá trị cặp số đó. |
| readEnter: Tương tự bài 18. Đọc kí tự ‘\n’ để restart chương trình. |

## **Kết quả demo**

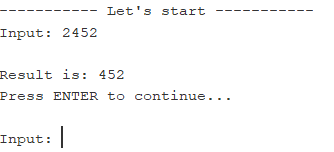
* Nhập vào 1 số bất kì:



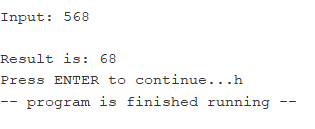
* Kết quả:



* Nếu tiếp tục nhấn enter:



* Kết thúc chương trình:

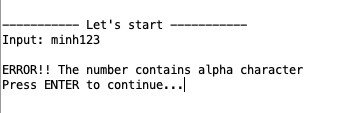


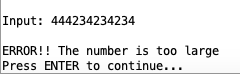
## **Xử lý ngoại lệ**

* Cho phép nhập String bằng lệnh syscall 8 thay vì nhập integer
* Hàm check\_input kiểm tra string nhập vào có chứa chữ cái không? Có thì báo lỗi integer
* Kiểm tra độ dài string có lớn hơn 10 hay không? Có thì báo lỗi số quá dài
* Chữ số 0-9 có mã ASCII từ 48-57.
  + VD Hàm atoi(character): Để chuyển từ kí tự ‘0’ sang int 0: trừ 48

|  |  |
| --- | --- |
|  | .data |
|  | input: .space 100 |
|  |
|  | message1: .asciiz "\n\n----------- Let's start -----------" |
|  | str\_err: .asciiz "\nERROR!! The number contains alpha character" |
|  | toolarge: .asciiz "\nERROR!! The number is too large" |
|  | message: .asciiz "\nInput: " |
|  | result: .asciiz "\nResult is: " |
|  | contPrompt: .asciiz "\nPress ENTER to continue..." |
|  | .text |
|  | li $v0, 4 # print start message |
|  | la $a0, message1 # pointer to start message in memory |
|  | syscall |
|  |  |
|  | start: li $v0, 4 # print input message |
|  | la $a0, message # pointer to input message in memory |
|  | syscall |
|  | li $v0, 8 # read a string |
|  | la $a0, input # pointer to input message in memory |
|  | la $a1, 100 # max readable 100 characters |
|  | syscall |
|  |  |
|  | li $t0, 10 # modulo 10 |
|  | li $s2, 0 # String index i |
|  | check\_input: |
|  | slt $t1, $s2, $t0 # if i < 10, continue |
|  | beqz $t1, error2 # else -> error: int number is too large |
|  | add $t1, $s2, $a0 # load string[i] address |
|  | lb $t2, 0($t1) # $t2 = string[i] |
|  | beq $t2, 10, input\_ok # read until last element '\n' |
|  |  |
|  | slti $t3, $t2, 48 # if ascii code is less than 48 |
|  | bne $t3, $zero, error1 # it's a character |
|  | slti $t3, $t2, 58 # if ascii code is greater than 57 |
|  | beq $t3, $zero, error1 # it's a character |
|  |  |
|  | addi $s2, $s2, 1 # i = i + 1 |
|  | j check\_input |
|  | # After this step, array = [2, 4, 5] |
|  | # Array size = $s2 = 3 |
|  | input\_ok: |
|  | add $s0, $zero, $a0 # load input to $s0 |
|  | jal init |
|  |  |
|  | li $v0, 4 # print result message |
|  | la $a0, result # pointer to result message in memory |
|  | syscall |
|  | move $a0, $s3 # save result to $v0 for printing |
|  | li $v0, 1 # print result |
|  | syscall |
|  | j readEnter # enter to continue |
|  | error1: |
|  | li $v0, 4 # print error string |
|  | la $a0, str\_err # pointer to error message in memory |
|  | syscall |
|  | j readEnter |
|  | error2: |
|  | li $v0, 4 # print error string |
|  | la $a0, toolarge # pointer to error message in memory |
|  | syscall |
|  | j readEnter |
|  | exit: |
|  | li $v0, 10 # exit |
|  | syscall |
|  | ############################# Nested loop ############################### |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  | init: |
|  | add $s1, $t2, $zero |
|  | li $s3, 0 # result |
|  | add $s4, $s2, -1 # nested loop index j = array\_size - 1 = 2 |
|  | loop\_i: |
|  | li $s5, 0 # nested loop j = 0 |
|  | slt $t2, $s4, $zero # if i < 0 |
|  | bnez $t2, end # exit j loop |
|  | li $s6, 0 # tmp = 0 |
|  | j loop\_j # execute j loop |
|  | loop\_j: |
|  | slt $t2, $s5, $s2 # if j < array\_size continue |
|  | beqz $t2, check # exit j loop |
|  | bne $s4, $s5, update\_tmp # if i != j, update tmp |
|  | addi $s5, $s5, 1 # else j = j + 1 |
|  | j loop\_j |
|  | update\_tmp: |
|  | mult $s6, $t0 # tmp \* 10 |
|  | mflo $s6 # tmp = tmp \* 10 |
|  | add $t2, $s5, $s0 # load string[j] address |
|  | lb $t2, 0($t2) # load string[j] value |
|  | addi $t2, $t2, -48 # atoi(string[i]) |
|  | add $s6, $s6, $t2 # tmp = tmp + string[j] |
|  | addi $s5, $s5, 1 # j = j + 1 |
|  | j loop\_j |
|  | check: slt $t2, $s3, $s6 # if result < tmp |
|  | bnez $t2, swap # result = tmp |
|  | addi $s4, $s4, -1 # else i = i - 1 |
|  | j loop\_i |
|  | swap: add $s3, $zero, $s6 # result = tmp |
|  | addi $s4, $s4, -1 # i = i - 1 |
|  | j loop\_i |
|  | end: |
|  | jr $ra # return to main |
|  | ################################################################################# |
|  | # Enter to continue |
|  | ################################################################################# |
|  | readEnter: # Ask user to press ENTER to continue |
|  | li $v0, 4 |
|  | la $a0, contPrompt |
|  | syscall |
|  | li $v0, 12 # sys code for readchar |
|  | syscall |
|  |  |
|  | move $t0, $v0 |
|  |  |
|  | bne $t0, 10, exit # exit if not ENTER |
|  |  |
|  | j start # return to the calling subroutine |

* **Demo xử lý ngoại lệ**

****

****