

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

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



**BÁO CÁO**

**Final projects Course 2019-02**

**Môn học: Thực hành Kiến trúc máy tính – IT3238**

***Đề tài: 1 + 10***

*Giảng viên hướng dẫn:*

ThS. Lê Bá Vui

*Sinh viên thực hiện:*

*Nhóm 16:*

Kiều Đăng Nam - 20176830

Lê Minh Quang - 20176856

*Mã lớp:* 113834

Hà Nội, tháng 5 năm 2020

**Mục Lục**

[**I. Bài 1** 4](#_Toc43557546)

[**1.** **Đề bài** 4](#_Toc43557547)

[**2.** **Phân tích cách thực hiện** 4](#_Toc43557548)

[**3.** **Ý nghĩa của các thanh ghi** 5](#_Toc43557549)

[**4.** **Ý nghĩa các chương trình con** 5](#_Toc43557550)

[**5.** **Kết quả chạy chương trình.** 5](#_Toc43557551)

[**II.Bài 12** 7](#_Toc43557552)

[**1.** **Phân tích cách thực hiện** 7](#_Toc43557553)

[**2.** **Ý nghĩa các thanh ghi** 9](#_Toc43557554)

[**3.** **Ý nghĩa các chương trình con** 9](#_Toc43557555)

[**4.** **Kết quả chạy chương trình** 10](#_Toc43557556)

**Báo cáo Final - project giữa kỳ 20192**

**Môn học: Thực hành Kiến trúc máy tính**

Trong báo cáo này , nhóm em xin được trình bày nội dụng thực hiện các chủ đề đã được phân công như sau:

Phần 1: Bài 1 do Kiều Đăng Nam thực hiện.

Phần 2: Bài 10 do Lê Minh Quang thực hiện.

# **I. Bài 1**

1. **Đề bài**

Curiosity Mars Bot: Xe tự hành Curiository MarsBot chay jtreen sao Hỏa, được vận hành từ xa bởi các lập trình viên trên Trái Đất bằng cách gửi các mã điều khiển từ một bàn phím ma trận. Các mã điều khiển quá trình di chuyển của MarsBot như sau:

|  |  |
| --- | --- |
| **Mã điều khiển** | **Ý nghĩa** |
| 1b4 | Marbot bắt đầu chuyển động |
| c68 | Marbot đứng im |
| 444 | Rẽ trái 90\* so vơi phương chuyển động gần đây và giữ hướng mới |
| 666 | Rẽ phải 90\* so vơi phương chuyển động gần đây và giữ hướng mới |
| dad | Bắt đầu để lại vết trên đường |
| cbc | Chấm dứt để lại vết trên đường |
| 999 | Tự động quay trở lại theo lộ trình ngược lại. Không vẽ vết, không nhận mã khác cho tới khi kết thúc lộ trình ngược. Mô tả: Marsbot được lập trình để nhớ lại toàn bộ lịch sử các mã điều khiển và khoảng thời gian giữa các lần đổi mã. Vì vậy, nó có thể đảo ngược lại lộ trình để quay về điểm xuất phát (dù có thể lệch một chút do hàm syscall sleep không thực sự thời gian thực) |

Sau khi nhận mã điều khiển, Curioisity Marsbot sẽ không xử lý ngay, mà phải đợi lệnh kích hoạt mã từ bàn phím Keyboard & Display MMIO Simulator. Có 2 lệnh như vậy:

|  |  |
| --- | --- |
| **Kích hoạt mã** | **Ý nghĩa** |
| Phím Enter | Kết thúc nhập mã và yêu cầu Marbot thực thi. |
| Phím Del | Xóa toàn bộ mã điều khiển đang nhập |

Đồng thời bổ sung thêm tính năng: mỗi khi gửi một mã điều khiển cho Marsbot, hiển thị mã đó lên màn hình console để người xem có thể giám sát lộ trình của xe

1. **Phân tích cách thực hiện**

Ý tưởng thực hiện: bài tập được lấy ý tưởng từ việc tổng hợp của 3 home assignment trong các bài thực hành week 10, 11. (Home Assignment 3, 4 – Week 10, Home Assignment 3 – Week 11)

* Mỗi khi người dùng nhập một ký vào trong Data Lab Sim sẽ tạp ra interrupt để lưu ký tự được nhập vào bộ nhớ, tạo nên CodeControl (Mã điều khiển)
* Tiếp theo là kiểm tra liên tục xem ký tự Enter/Delete có được nhập trong cửa sổ Keyboard & Display MMIO Simulator hay không.

+ Trường hợp là enter, chương trình sẽ tiến hành kiểm tra xem mã Code được nhập có hợp lệ hay không, về độ dài (gồm 3 ký tự), nếu không sẽ thông báo lỗi WRONG CODE. Nếu thỏa mãn về độ dài sẽ tiế tục kiểm tra xem mã code đó có trùng khớp với các mã code điều khiển đã được quy định trước đó. Nếu không thì thông báo WRONG CODE. Ngược lại sẽ chuyển đến hàm thực hiện các hành động của bot tương ứng với mã code điều khiển.

+ Trường hợp Delete, chương trình sẽ xóa toàn bộ mã điều khiển đang nhập.

* In ra console code điều khiển.

1. **Ý nghĩa của các thanh ghi**

|  |  |
| --- | --- |
| **Thanh ghi** | **Ý nghĩa** |
| $a0 | Địa chỉ của ký tự đầu tiên của xâu: string[0]. |
| $t0 | Số lượng các ký tự khác nhau có trong xâu. |
| $t1 | Giá trị biến chạy i để duyệt xâu từ đầu đến cuối. |
| $s1 | Giá trị của ký tự kết thúc xâu ( = 10, do kết thúc xâu = Enter) |
| $t2 | Địa chỉ của string[i] – ký tự hiện tại |
| $t3 | Giá trị của string[i] |
| $s0 | Biến check xem ký tự đó đã xuất hiện trước đó hay chưa |
| $t4 | Giá trị j (biến chạy để duyệt xâu từ đầu đến vị trí i ) |
| $t5 | Địa chỉ của string[j] |
| $t6 | Giá trị của string[j] |

1. **Ý nghĩa các chương trình con**
   1. Các hàm thực hiện di chuyển của Marsbot
2. storePath

* Chức năng: Lưu lại thông tin về đường đi của Marsbot vào mảng path, mỗi phần tử được lưu dưới dạng cấu trúc (x, y, z) với x,y là điểm tạo độ điểm đầu tiên, z là hướng đi của cạnh đó.
* Input: nowHeading, lengthPath.

1. goBack

* Chức năng: Marsbot đi ngược lại theo lộ trình nó đã đi, về điểm xuất phát. Mỗi lần quay ngược lại và đi về điểm đầu tiên của một cạnh trên đường đi, chương trình lấy hướng đi cạnh đó và đi ngược lại đến khi gặp điểm có tọa độ như đã lưu thì kế thúc việc đi ngược lại trên cạnh và tiếp tục trên các cạnh khác
* Input: mảng path lưu thông tin về đường đi, biến lengthPath lưu kích cỡ của mảng Path ( =12bytes)

1. goRight, goLeft

* Chức năng: Điều khiển Marsbot di chuyển quay sang phải (trái) một góc 900.
* Input: biến nowHeading. Khi di chuyển sang phải ta chỉ cần tăng biến nowHeading lên 900 và bên trái sẽ giảm 900. Và gọi hàm ROTATE để thực hiện.

1. ROTATE

* Chức năng: quay Marsbot theo hướng có góc được lưu trong nowHeading.
* Input: biến nowHeading.

1. TRACK, UNTRACK

* Chức năng: điều khiển Marsbot bắt đầu để lại vết (TRACK) hoặc dừng việc ghi vết (UNTRACK).
* Thực hiện: Load 1 vào địa chỉ LEAVETRACK nếu muốn để lại vết và 0 nếu kết thúc ghi vết.

1. GO, STOP

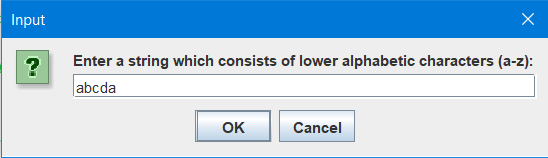
* Chức năng: điều khiển Marsbot bắt đầu chuyển động (GO) hoặc dừng lại (STOP).
* Thực hiện: Load 1 vào địa chỉ MOVING nếu muốn đi và 0 khi dừng lại
  1. Các hàm kiểm tra

1. isEqualString

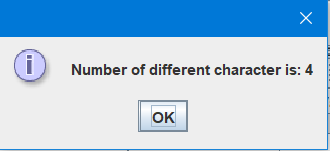
* Chức năng: Kiểm tra mã code điều khiển người dung nhập (lưu trong $s1) có trùng với các mã code đã được định nghĩa sẵn (lưu trong $s3) hay không.

Nếu 2 xâu này bằng nhau thì thanh ghi $s0 = 1, ngược lại = 0.

1. **Kết quả chạy chương trình.**
2. Nhập xâu đầu vào.



1. Kết quả.



# **II.Bài 12**

***Đề bài***: Ticket numbers usually consist of an even number of digits. A ticket number is considered lucky if the sum of the first half of the digits is equal to the sum of the second half. Given a ticket number n, determine if it islucky or not.

1. **Phân tích cách thực hiện**

* Thực hiện vòng lặp chia dần số đã cho cho 10 cho đến khi thương = 0. Số ký tự chính bằng số vòng lặp. Chia đôi ra được số ký tự mỗi bên
* Thực hiện vòng lặp chia số đã cho cho 10 cho đến khi số vòng lặp bằng số ký tự mỗi bên. Khi đó tổng số dư = tổng các chữ số vế phải. Đặt lại tổng số dư bằng không thực hiện vòng lặp tiếp, tính được tổng các chữ số vế trái
* Thực hiện so sánh 2 vế
* Mã nguồn chương trình theo MIPs:

.data

Message: .asciiz "Number: "

Message1: .asciiz "Lucky"

Message2: .asciiz "Not lucky"

Message3: .asciiz "The end"

.text

li $v0, 51

la $a0, Message

beq $a1, -1, Done # Validate input value

syscall

init:

add $t0, $zero, $a0 # input value n

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

# Loop: Calculate the number of characters saved in $t3

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

main:

add $s1,$t0, $zero # The variable temporarily stores the value of n

addi $s2, $zero, 10 # the dividend = 10

jal countCharacter # $t3 = number of character

nop

addi $s3, $zero, 2

divu $t3, $s3

mfhi $t1

mflo $t6 # $t6: number of characters on each side

bne $t1, $zero, Done # check number of character is even ?

jal totalOneSide

add $t5,$t2,$zero # right value

add $t2, $zero,$zero

jal totalOneSide

add $t4,$t2, $zero # left value

sub $t6, $t4, $t5 # if (left - right) == 0 ?

beq $t6, $zero, print1

j print2

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

# Count number of character

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

countCharacter:

Loop:

divu $s1,$s2 # n / 10

mflo $a1 # The result is saved to $a1

add $s1,$zero, $a1 # Update the value to n

add $t3, $t3, 1 # Increase count variable to 1

bne $a1, $zero, Loop # if result = 0 exit loop

jr $ra

end\_count:

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

# Total one side

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

totalOneSide:

add $a2, $zero, $zero # bien dem

loop:

divu $t0, $s2 # n / 10

mflo $a1 # result

mfhi $t1 # surplus

add $t0, $zero, $a1 # update n

add $t2, $t2, $t1 # sum = sum + surplus

addi $a2, $a2, 1 # count

slt $s4, $a2, $t6 # the variable count = (total character) / 2 ?

bne $s4, $zero, loop

jr $ra

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

# Print mesage

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

print1:

li $v0, 4

la $a0, Message1

syscall

li $v0, 10

syscall

print2:

li $v0, 4

la $a0, Message2

syscall

li $v0, 10

syscall

Done:

li $v0, 4

la $a0, Message3

syscall

li $v0, 10

**syscall**

1. **Ý nghĩa các thanh ghi**

|  |  |
| --- | --- |
| **Thanh ghi** | **Ý nghĩa** |
| $t0 | Lưu số nhập vào |
| $t1 | Số dư khi chia |
| $t2 | Tổng các chữ số mỗi bên |
| $t3 | Số các chữ số |
| $t4 | Tổng các chữ số vế trái |
| $t5 | Tổng các chữ số vế phải |
| $t6 | Số các chữ số mỗi bên |
| $t7 | Số dư khi chia số các chữ số cho 2(Kiểm tra xem số lượng chữ số chẵn hay không) |
| $s1 | Biến tạm thời lưu giá trị nhập vào |
| $s2 | 10 (số chia) |
| $s3 | 2 (số chia) |
| $a1 | Thương |
| $a2 | Biến đếm |

1. **Ý nghĩa các chương trình con**

* ***countCharacte***r: Tính số lượng các chữ số lưu kết quả vào thanh ghi $t3
* ***totalOneSide***: Tính tổng các chữ số vế phải , vế trái. Kết quả lần lượt lưu vào thanh ghi $t4, $t5

1. **Kết quả chạy chương trình**

Cho đầu vào: 1607. Kết quả:



Trong đó vế trái = vế phải = 7



Cho đầu vào: 1604. Kết quả:



Trong đó vế trái = 7, vế phải = 4

