TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

- - - - - - - -□ - - - - - - - -

BÁO CÁO BÀI TẬP
MÔN KIẾN TRÚC MÁY TÍNH
ĐỀ TÀI SỐ 1, 6

Giảng viên hướng dẫn: **ThS. Nguyễn Đức Tiến**

Sinh viên : **Nguyễn Trí Minh – 20132599**

Nguyễn Việt Toàn - 20134023

Nhóm: **21** - Lớp: **Việt Nhật B K58**

Contents

A.BÀI TẬP 1. - 1 -
I.ĐỀ BÀI - 1 -
II. Định hướng - 2 -
III. Ý nghĩa các hàm trong mã nguồn - 2 -
1. storePath - 2 -
2. goBack - 2 -
3. goRight và goLeft - 3 -
4. ROTATE - 3 -
5. TRACK, UNTRACK - 3 -
6. GO, STOP - 3 -
7. pushErrorMess - 3 -
8. isEqualString - 3 -
9. removeControlCode - 3 -
10. Các thao tác trong phần xử lí interrupt - 3 -
IV. MÃ NGUỒN - 4 -
V. HÌNH ẢNH MÔ PHỎNG - 4 -
B. BÀI TẬP 6 - 6 -
I. ĐỀ BÀI - 6 -
II. ĐỊNH HƯỚNG - 6 -
IV. MÃ NGUỒN - 7 -

A.BÀI TẬP 1.

I.ĐỀ BÀI

Curiosity Marsbot

Xe tự hành Curioisity Marsbot chạy trên sao Hỏa, được vận hành từ xa bởi các lập trình viên trên Trái Đất. Bằng cách gửi đi các mã điều khiển từ một bàn phím ma trận, lập trình viên điều khiển quá trình di chuyển của Marbot 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 dở dang.

Hãy lập trình để Marsbot có thể hoạt động như đã mô tả.
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.

II. Định hướng

- B1: Mỗi khi người dùng nhập 1 kí tự từ Digital Lab Sim sẽ tạo ra interrupt để lưu kí tự được nhập vào bộ nhớ, tạo nên đoạn code điều khiển.
- B2: Kiểm tra liên tục xem kí tự Enter có được nhập ở Keyboard & Display MMIO Simulator hay không. Khi kí tự Enter được nhập, sẽ kiểm tra xem đoạn code điều khiển có hợp lệ không (gồm 3 kí tự), nếu không sẽ thông báo code lỗi và sang bước 4. Nếu có thì chuyển sang bước 3.
- B3: Lần lượt kiểm tra xem code điều khiển được nhập vào có trùng với các đoạn code điều khiển đã quy định sẵn. Nếu không thì thông báo đoạn code bị lỗi. Ngược lại thực hiện thao tác theo quy định sẵn.
- B4: In ra console code điều khiển đã nhập và xóa lưu trữ trong bộ nhớ.

III. Ý nghĩa các hàm trong mã nguồn

1. **storePath**

   Ý nghĩa: Lưu lại thông tin về đường đi của Marsbot vào mảng path.
   Đầu vào: biến ***nowHeading***, ***lengthPath***.

   Mảng ***path*** lưu thông tin về đường đi hay đúng hơn là thông tin về các cạnh của đường đi của Marsbot. Mỗi một cạnh gồm 3 thông tin: tọa độ x và y của điểm đầu tiên, hướng đi của cạnh đó.

2. **goBack**

   Ý nghĩa: điều khiển Marsbot đi ngược lại theo lộ trình nó đã đi và về điểm xuất phát
   Đầu vào: mảng ***path*** lưu thông tin đường đi, biến ***lengthPath*** lưu kích cỡ của mảng ***path*** theo byte.

   Mảng ***path*** lưu thông tin đường đi. Mỗi thông tin về 1 cạnh gồm tọa độ x và y và hướng đi - 3 số nguyên. Do đó mỗi thông tin đường đi sẽ chiếm 12 byte. Do đó ***lengthPath*** sẽ có giá trị là bội của 12.
   Mỗi khi muốn quay ngược lại và đi về điểm đầu tiên của 1 cạnh trên đường đi, ta sẽ lấy hướng đi của cạnh đó và đi ngược lại, đến khi nào gặp điểm có tọa độ như đã lưu thì kết thúc việc đi ngược trên cạnh đó, tiếp tục trên cạnh khác.

3. **goRight và goLeft**

Ý nghĩa: điều khiển Marsbot quay và di chuyển sang phải (với hàm goRight) hoặc trái (goLeft) một góc 90*.

Đầu vào: biến ***nowHeading***

Muốn di chuyển sang phải 90* ta chỉ cần tăng biến ***nowHeading*** lên 90*, đối với bên trái là giảm đi 90*.

Sau đó gọi hàm ***ROTATE*** để thực hiện.

4. **ROTATE**

Ý nghĩa: quay Marsbot theo hướng có số độ lưu trong ***nowHeading***

Đầu vào: biến ***nowHeading***

Load biến ***nowHeading*** và lưu vào địa chỉ ***HEADING*** (0xffff8010) để Marsbot chuyển hướng.

5. **TRACK, UNTRACK**

Ý nghĩa: điều khiển Marsbot bắt đầu để lại vết (TRACK) hoặc kết thúc để lại vết (UNTRACK)

Load 1 vào địa chỉ ***LEAVETRACK*** (***0xffff8020***) nếu muốn để lại vết và load 0 nếu muốn kết thúc vết.

6. **GO, STOP**

Ý nghĩa: điều khiển Marsbot bắt đầu chuyển động (GO) hoặc dừng lại (STOP)

Load 1 vào địa chỉ ***MOVING (0xffff8050)*** nếu muốn để lại vết và load 0 nếu muốn kết thúc vết.

7. **pushErrorMess**

Ý nghĩa: hiện thông báo dialog khi người dùng nhập code điều khiển không đúng. Sử dụng các hàm syscall 4, 55.

8. **isEqualString**

Ý nghĩa: so sánh code điều khiển mà người dùng nhập vào với 1 code điều khiển nào đó có địa chỉ được đặt trong $s3

Lần lượt so sánh các kí tự trong 2 xâu này. Nếu 2 xâu bằng nhau thanh ghi ***$t0*** có giá trị là 1, ngược lại là 0.

9. **removeControlCode**

Ý nghĩa: xóa xâu ***inputControlCode*** (nơi lưu trữ code điều khiển nhập vào).

Lần lượt gán các kí tự trong xâu bằng ***'\0'***.

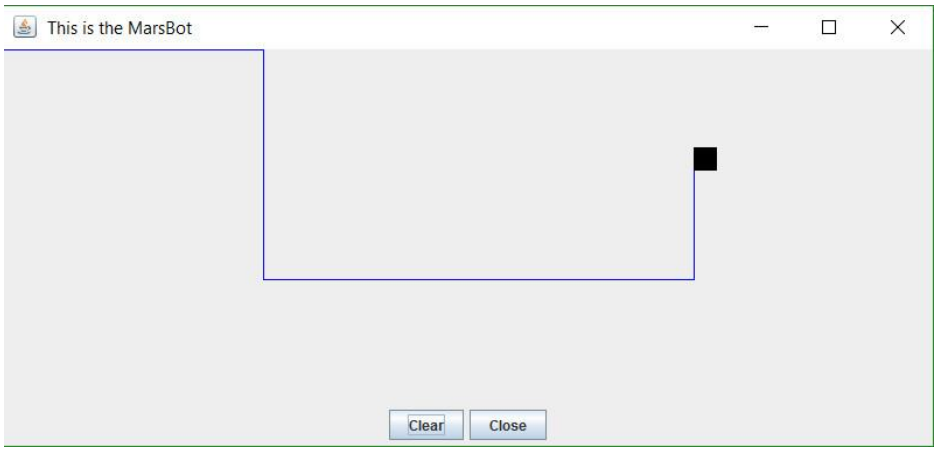
10. **Các thao tác trong phần xử lí interrupt**

Lần lượt quét các hàng của Digital Lab Sim để xem phím nào được bấm. Tiếp đó dựa vào mã được trả về ghi kí tự tương ứng vào bộ nhớ, cụ thể là ghi vào cuối xâu ***inputControlCode.***

IV. MÃ NGUỒN

Lưu trong file n01_g21_NguyenTriMinh.asm đi kèm.

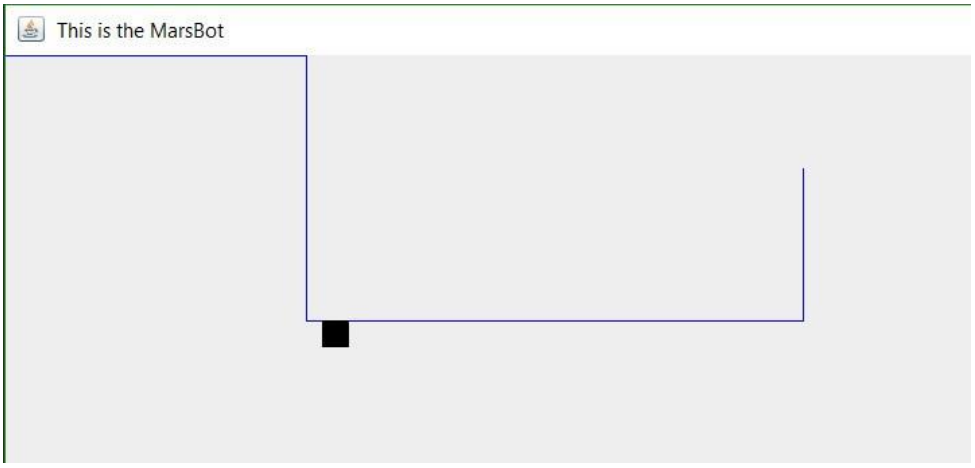
V. HÌNH ẢNH MÔ PHỎNG

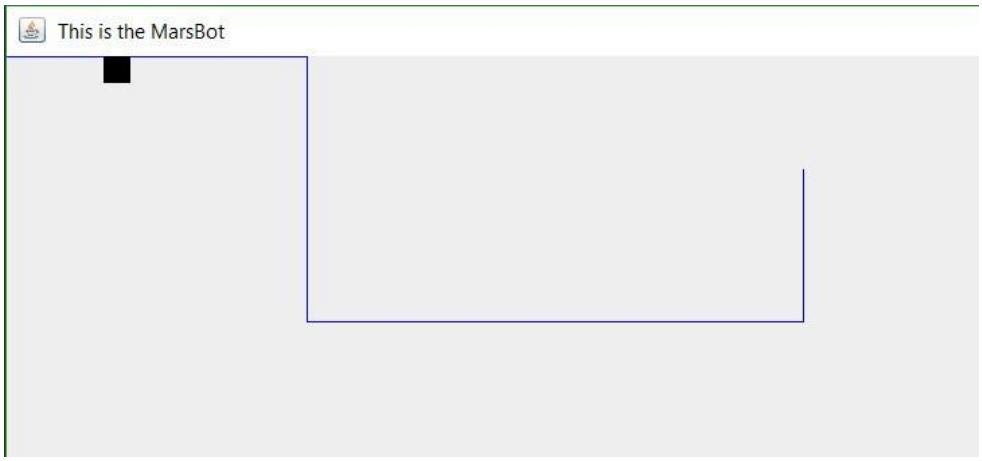


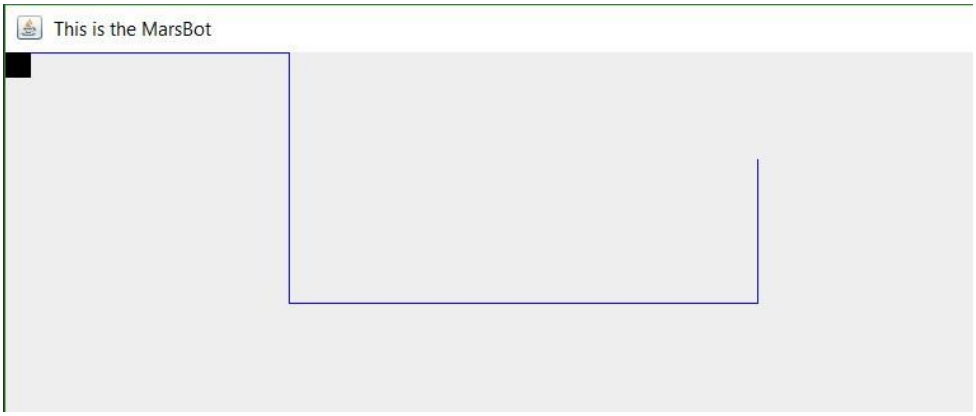
Điều khiển Marsbot bằng các câu lệnh.

In các câu lệnh ra console.

In ra dialog thông báo control code nhập vào bị lỗi







Marsbot đi ngược lại trở về vị trí ban đầu.

B. BÀI TẬP 6

I. ĐỀ BÀI

Bài 6:

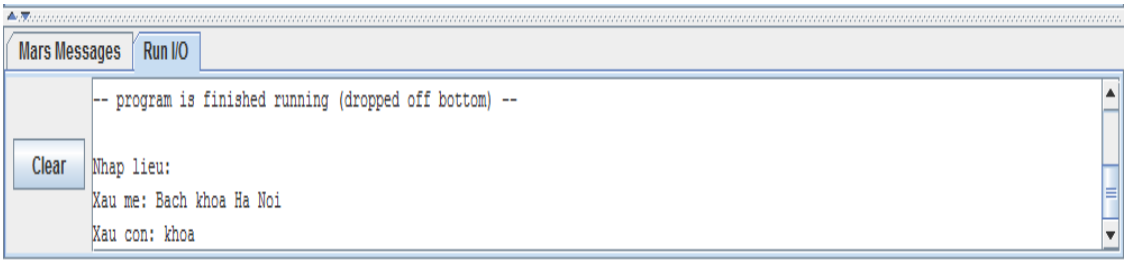
Viết chương trình sử dụng MIPS để tìm các vị trí xuất hiện của xâu kí tự con trong xâu kí tự lớn. Chương trình sẽ yêu cầu người dùng nhập vào một xâu và một xâu cần tìm kiếm (cả 2 xâu đều là xâu ASCII). Đầu ra của chương trình sẽ là chỉ số nơi mà xâu cần tìm kiếm xuất hiện trong xâu gốc.

Hàm tìm kiếm cho phép tìm theo các cách
1. Không phân biệt chữ hoa, chứ thường
2. Phân biệt chữ hoa, thường

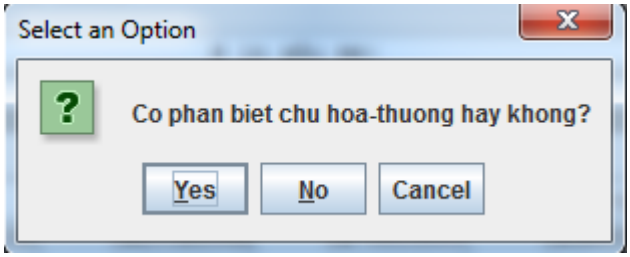
Ví dụ:
Input:
string1: DAI HOC BACH KHOA LA dai hoc da nganh
string2: HOC không phân biệt hoa thường
Output:
4, 25

II. ĐỊNH HƯỚNG

- B1: Nhập liệu
  - Nhập xâu mẹ ở hàm InputString
  - Nhập xâu con cần tìm ở hàm InputSubString



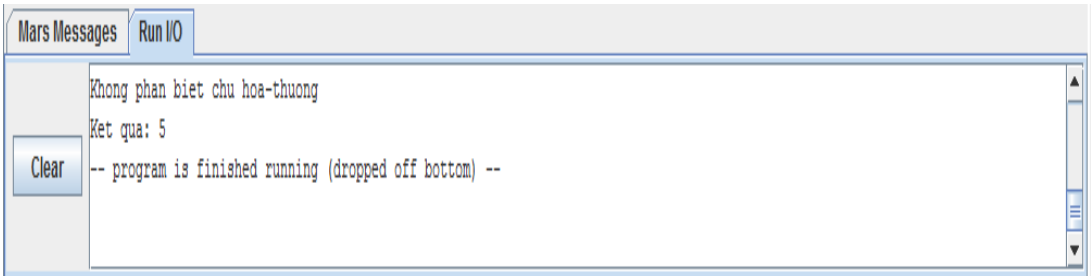
- B2: Thuật toán thực hiện:
  - Chương trình hỏi người dung có muốn phấn biệt chữ hoa chữ thường hay không:
    YES: có
    NO: không



- Nếu chọn YES: -> notUpCase1

  Duyệt xâu mẹ từ kí tự đầu tiên:
  +Nếu gặp kí tự trùng với kí tự đầu của xâu cần tìm thì bắt đầu so sánh. So sánh đến khi nào đủ xâu con thì lưu giá trị vị trí bắt đầu của xâu cần tìm trong xâu mẹ
  +Nếu không trùng kí tự đầu của xâu cần tìm thì duyệt đến kí tự tiếp theo
  +Cứ như vậy đến khi duyệt hết xau mẹ thì in ra kết quả
- Nếu chọn NO: -> UpCase

  Đưa toàn bộ xâu mẹ và xâu cần tìm thành chư hoa rồi tìm với thuật toán giống như YES.



IV. MÃ NGUỒN

Lưu trong file n06_g21_NguyenVietToan.asm đi kèm.