HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION AND TECHNOLOGY

---○🕮○---

Final Report

COMPUTER ARCHITECTURE LAB

Supervior:Lê Bá Vui
Student: Nguyễn Kim Tuyến – Lương Nam Khánh
Group: 9
Class:135411

Project 1: Curiosity Marsbot

I.Đề bài

Xe tự hành Curiosity 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 độ so vơi phương chuyển động gần đây và giữ hướng mới
666	Rẽ phải 90 độ 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

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.
Phím Space	Lặp lại lệnh đã thực hiện trước đó

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

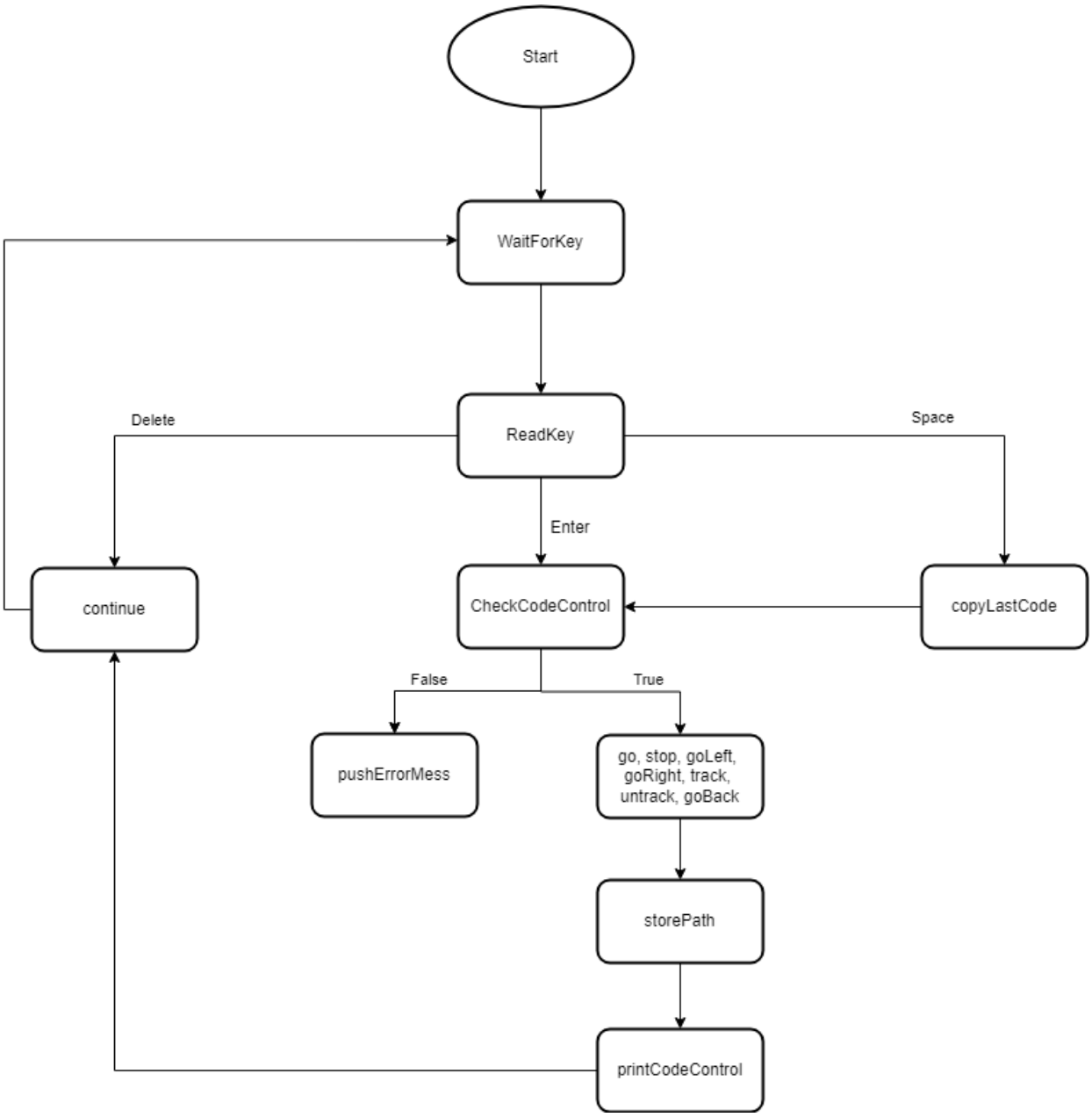
Đồng thời bổ sung thêm tính năng: mỗi khi gửi một mã điều khiển cho Marsbot, hiển thị mã đó lên màn hình console để người xem có thể giám sát lộ trình của xe

II.Phân tích cách thực hiện

1. Mô tả

- Bước 1: Mỗi khi người dùng nhập 1 kí tự từ Digital Lab Sim sẽ lưu kí tự được nhập vào bộ nhớ (***inputControlCode*** và ***latestCode***) .
- Bước 2: Kiểm tra liên tục xem kí tự Enter có được nhập ở Keyboard & Display MMIO Simulator hay không. Nếu người dùng nhập Del, chuyển tới bước 5. Nếu người dùng nhập Enter, chuyển tới bước 4. Nếu người dùng nhập Space, chuyển tới bước 3.
- Bước 3: Sao chép ***latestCode*** vào ***inputControlCode***. Chuyển tới bước 4.
- Bước 4: 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 5. Nếu hợp lệ thì chuyển sang bước 6.
- Bước 5: Xóa toàn bộ mã điều khiển đang nhập. Quay trở lại bước 1
- Bước 6: Marbot thực hiện các yêu cầu theo lệnh. Lưu lại đường đi (nếu cần)
- Bước 7: In ra console code điều khiển đã nhập, đồng thời xóa luôn mã điều khiển. Quay trở lại bước 1

2. **Lưu đồ thuật toán**



III.Ý nghĩa các hàm trong mã nguồn

1. **CheckControlCode**

Ý nghĩa: Kiểm tra ***inputControlString*** có trùng với các Control Code hay không:

Bước 1: Kiểm tra ***lengthControlCode***, nếu bằng 3, chuyển tới bước 2. Ngược lại, hiển thị lỗi "Wrong Control Code"

Bước 2: Kiểm tra nội dung ***inputControlCode*** có trùng với các Control Code (địa chỉ lưu trong thanh ghi $s3) hay không bằng hàm ***isEqualString***. Nếu có, thực thi các hàm điều khiển Marsbot tương ứng. Ngược lại, thông báo lỗi

2. isEqualString

Ý nghĩa: so sánh 2 xâu

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

3. pushErrorMess

Ý nghĩa: hiện thông báo dialog khi người dùng nhập code điều khiển không đúng.

Sử dụng các hàm syscall 4, 55.

4. printControlCode

Ý nghĩa: 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

5. removeControlCode

Ý nghĩa: xóa ***inputControlString***

Sử dụng vòng lặp, lần lượt gán các kí tự từ 0 tới ***lengthControlCode*** (độ dài của xâu hiện tại) bằng '\0'. Sau đó update ***lengthControlCode*** = 0

6. copyLastCode

Ý nghĩa: Copy các ký tự của ***latestCode*** vào trong ***InputControlCode***. Được gọi khi ấn phím Space để lặp lại lệnh đã thực hiện trước đó

7. GO và STOP

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

Load 1 vào địa chỉ MOVING (0xffff8050) nếu muốn Marsbot chuyển động và load 0 nếu muốn Marsbot dừng lại.

8. goRight và goLeft

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

Đầu vào: biến ***nowHeading***

Muốn di chuyển sang phải 90 độ so với hướng hiện tại ta chỉ cần tăng biến nowHeading thêm 90 dộ, đối với bên trái là giảm đi 90 độ.
Sau đó gọi hàm ***ROTATE*** để thực hiện thay đổi.

9. ROTATE

Ý nghĩa: quay Marsbot theo hướng được lưu trong ***nowHeading***

Đầu vào: biến ***nowHeading***

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

10. TRACK và 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.

11. goBack

Ý nghĩa: điều khiển Marsbot đi ngược lại theo lộ trình nó đã đi và không để lại vết

Đầu vào: mảng path lưu thông tin đường đi, biến ***lengthPath*** lưu kích cỡ của mảng history theo byte.

Mảng path: lưu thông tin về cạnh đường đi. Mỗi cạnh gồm 3 số nguyên: tọa độ x và y và hướng đi z. Do đó mỗi thông tin đường đi sẽ chiếm 12 byte (3 words x 4 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 đảo ngược hướng đã thực hiện (bằng cách tăng thêm 180 độ) và di chuyển đến khi nào gặp điểm có tọa độ đã lưu thì kết thúc việc đi ngược trên cạnh đó, tiếp tục trên cạnh khác. Dừng lại khi Marbot quay lại vị trí xuất phát

12. storePath

Ý nghĩa: Lưu lại thông tin về đường đi của Marsbot vào mảng path.

Đầu vào: biến ***nowHeading***, ***lengthPath***, ***WHEREX***, ***WHEREY***.

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 (***WHEREX***) và y (***WHEREY***) của điểm đầu tiên, z (***nowHeading***) là hướng đi của cạnh đó.

13. Các thao tác trong phần xử lí interrupt (địa chỉ cố định 0x80000180)

Lần lượt quét các hàng của Digital Lab Sim để xem phím nào được bấm bằng cách so sánh với các giá trị được lưu trong ***Key value***. Tiếp đó dựa vào mã được trả về ghi kí tự tương ứng vào bộ nhớ.

Sau khi kết thúc chương trình ngắt, sử dụng lệnh eret để quay trở lại chương trình chính. Lệnh eret sẽ gán nội dung thanh ghi PC bằng giá trị trong thanh ghi $14 (epc)

Vì thanh ghi PC vẫn chứa địa chỉ của lệnh mà ngắt xảy ra, tức là lệnh đã thực hiện xong, chứ không chứa địa chỉ của lệnh kế tiếp. Bởi vậy cần lập trình để tăng địa chỉ chứa trong thanh ghi epc.

IV.Mã Nguồn

Trong file n01_g09_luongnamkhanh.asm

V.Mô Phỏng

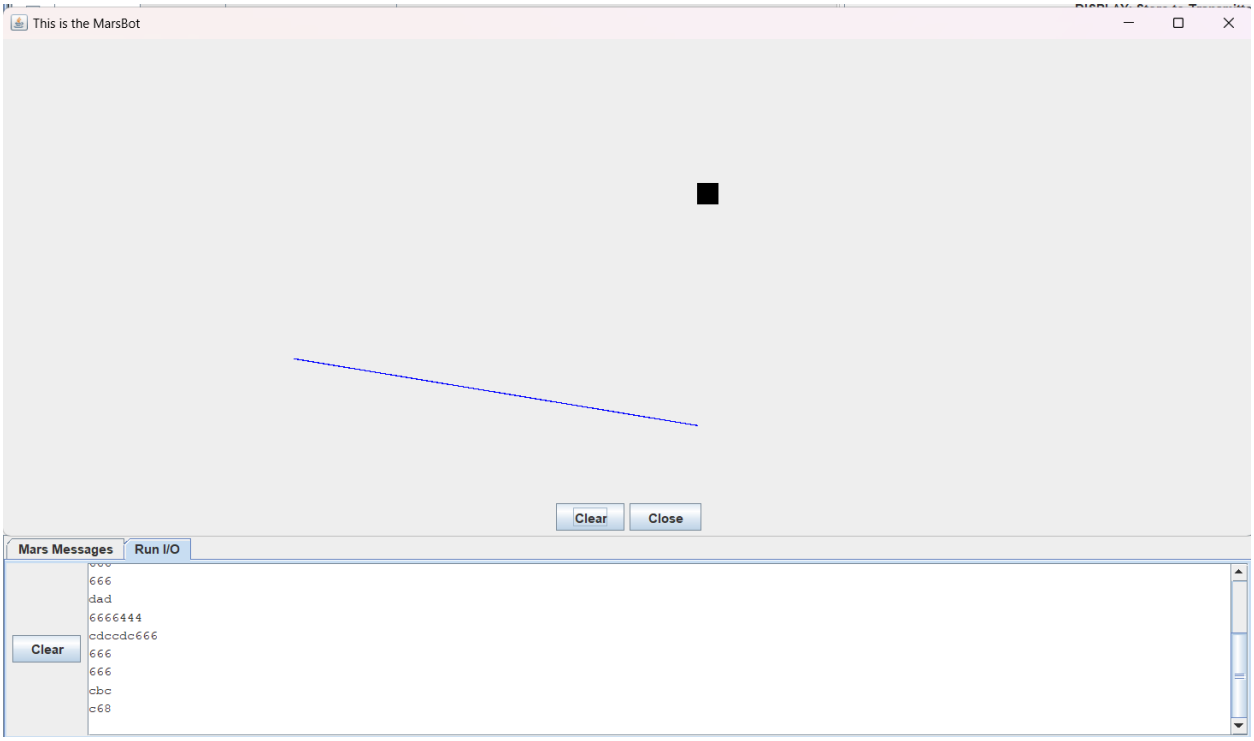


Figure 1. Điều khiển Marsbot bằng các câu lệnh và in các câu lệnh ra console.

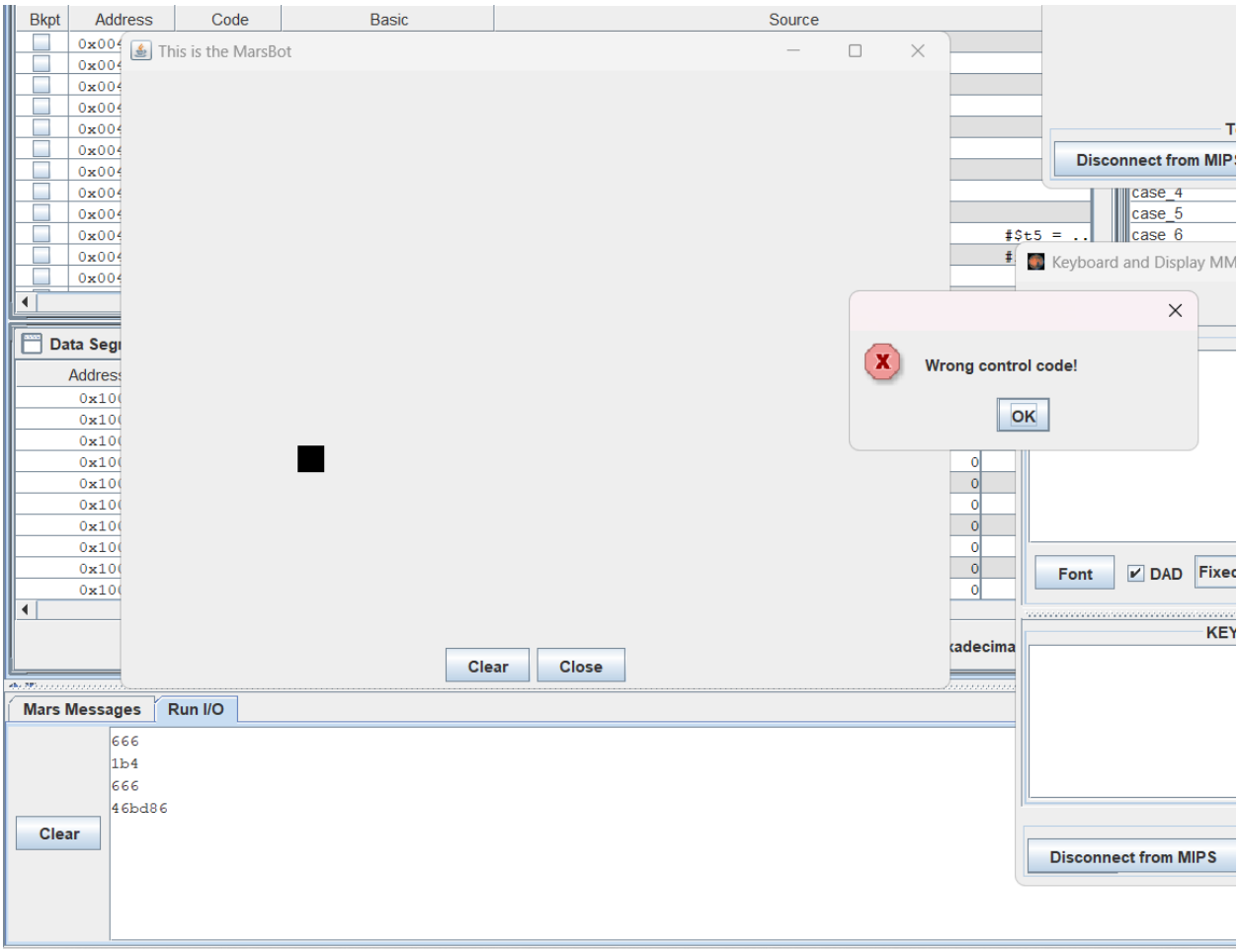


Figure 2. In ra dialog thông báo control code nhập vào bị lỗi

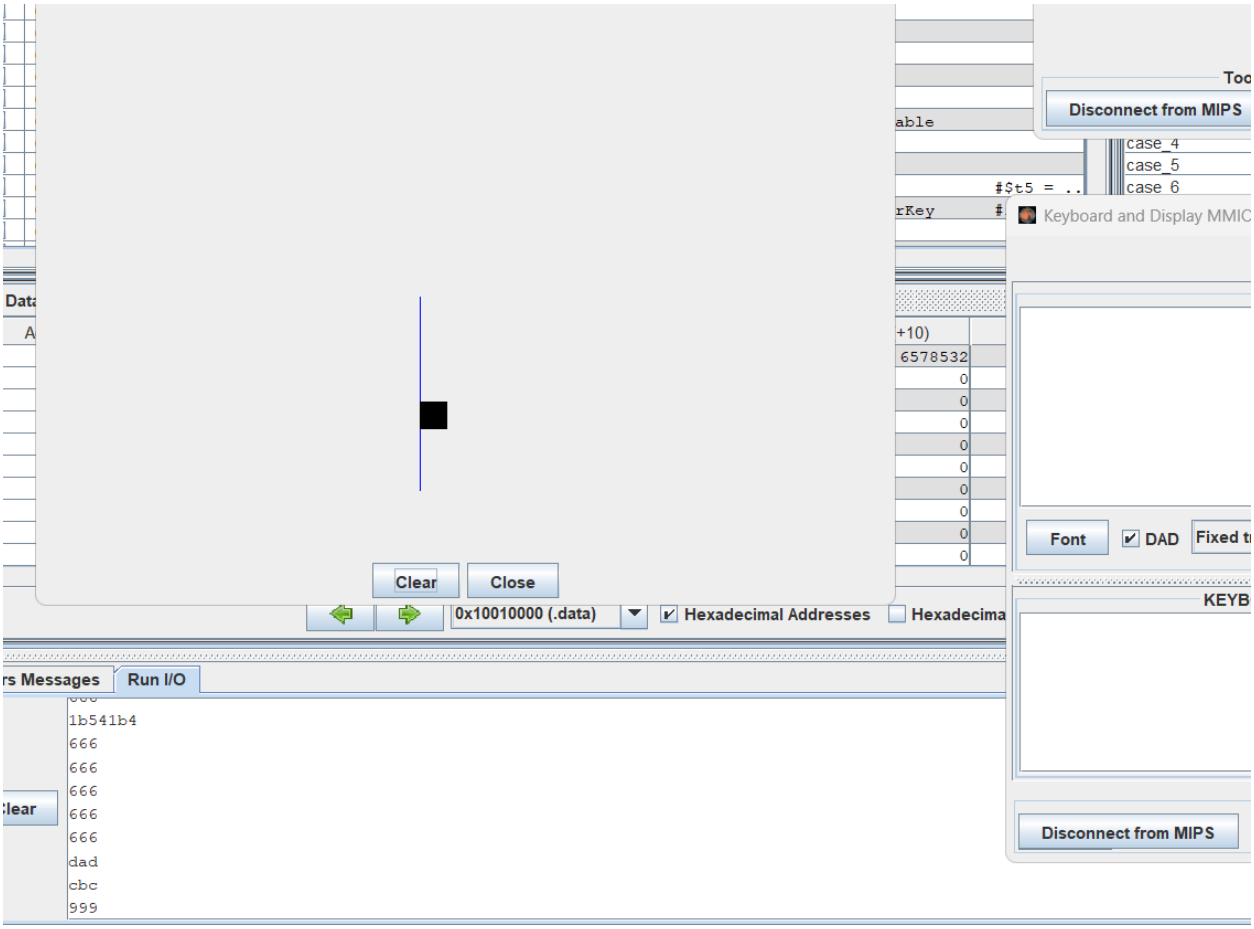


Figure 3. Marsbot đi ngược lại trở về vị trí ban đầu

PROJECT 3: Kiểm tra tốc độ và độ chính xác gõ phím

I. Đề bài

Chương trình sau sẽ đo tốc độ gõ bàn phím và hiển thị kết quả bằng 2 đèn LED 7 đoạn. Nguyên tắc:

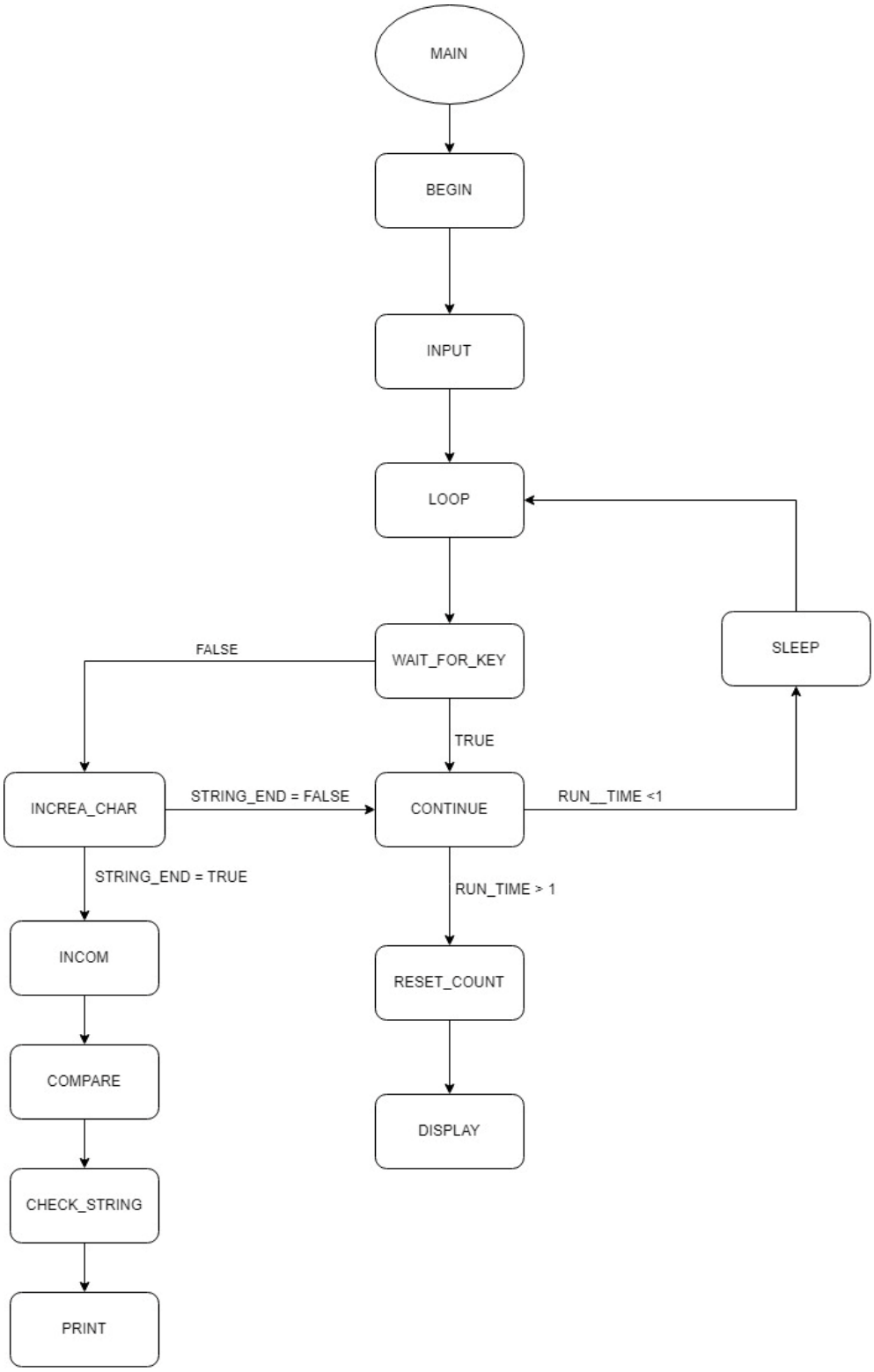
- Cho một đoạn văn bản mẫu, cố định sẵn trong mã nguồn.
- Sử dụng bộ định thời Timer để tạo ra khoảng thời gian để đo. Đây là thời gian giữa 2 lần ngắt, chu kỳ ngắt.
- Trong khoảng thời gian đó, người dùng nhập các ký tự từ bàn phím. Chương trình cần phải đếm số kí tự đúng mà người dùng đã gõ và hiển thị lên các đèn LED.

II. Phân tích cách thực hiện

1. Ý tưởng

- Bước 1: Đếm các ký tự được nhập vào vùng input.
- Bước 2: Cứ mỗi 1 chu kỳ thì sẽ tổng hợp lại và đưa ra tốc độ gõ phím.
- Bước 3: Nếu người dùng nhập phím enter sẽ xuất hiện hộp thoại hỏi người dùng muốn tiếp tục thực hiện chương trình hay thoát đồng thời in ra tốc độ gõ phím trung bình trong 1s và hiển thị lên đèn LED 7 đoạn số lượng từ người dùng gõ đúng.

2. Lưu đồ thuật toán



III. Ý nghĩa các hàm trong mã nguồn

1. ***MAIN***

- Hàm đẩy các hằng vào thanh ghi và gọi đến các hàm con bên dưới.

2. ***INPUT***

- Hàm dùng để đọc các ký tự nhập từ bàn phím

3. ***LOOP***

- Trong hàm sẽ nhận ký tự, tăng biến đếm khi đếm hết thời gian sẽ tự động đếm lại từ đầu.

4. ***WAIT_FOR_KEY***

- Hàm sẽ kiểm tra trong input trong 1 chu kỳ có ký tự nào được nhập vào không, nếu có sẽ nhảy đến INCRE_CHAR, nếu không nhảy đến CONTINUE.

5. ***CONTINUE***

- Kiểm tra xem đã đủ 1s chưa, nếu chưa đủ sẽ nhảy đến SLEEP, nếu đủ sẽ nhảy đến RESETCOUNT

6. ***SLEEP***

- Bản chất chính là gọi syscall của hệ thống nhằm treo máy và nhảy ngược về LOOP.

7. ***RESETCOUNT***

- Tính số thời gian gõ, số ký tự đã nhập trong 1s sau đó nhảy đến DISPLAY để in nó ra Digital Lab Sim.

8. ***DISPLAY***

- Dùng để hiển thị số lên 2 đèn LED (Có 2 hàm con để in ra LED bên trái và phải là SHOW_7SEG_LEFT và SHOW_7SEG_RIGHT)

9. ***COMPARE***

- Kiểm tra chuỗi nhận được và chuỗi gốc xem đã giống nhau chưa.

10. ***PRINT***

- In ra màn hình số ký tự nhập đúng và tốc độ trung bình.

11. ***ASK***

- Hỏi người dùng muốn chạy lại chương trình không. Nếu có reset các biến và lặp lại, nếu không kết thúc chương trình.

12. Interrupt subroutine

- Kiểm soát ngắt của hệ thống để đảm bảo CPU thực hiện đọc giá trị liên tục.

IV. Mã nguồn

Trong file n03_g09_NguyenKimTuyen.asm

V. Mô phỏng