Final Project Report – Group 5

Mã lớp: 139363 – Kỳ: 2022.2

GVHD: Thầy Lê Bá Vui

I. Project 1 *(Người thực hiện: Nguyễn Văn Cường – 20215006)*

1. <u>Đề bài:</u> Curiosity Marsbot

- 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 với phương chuyển động gần đây và giữ hướng mới
666	Rẽ phải 90 độ so với 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 Marsbot thực thi
Phím Del	Xóa toàn bộ mã điều khiển đang nhập dở dang
Phím Space	Lặp lại lệnh đã thực hiện trước đó

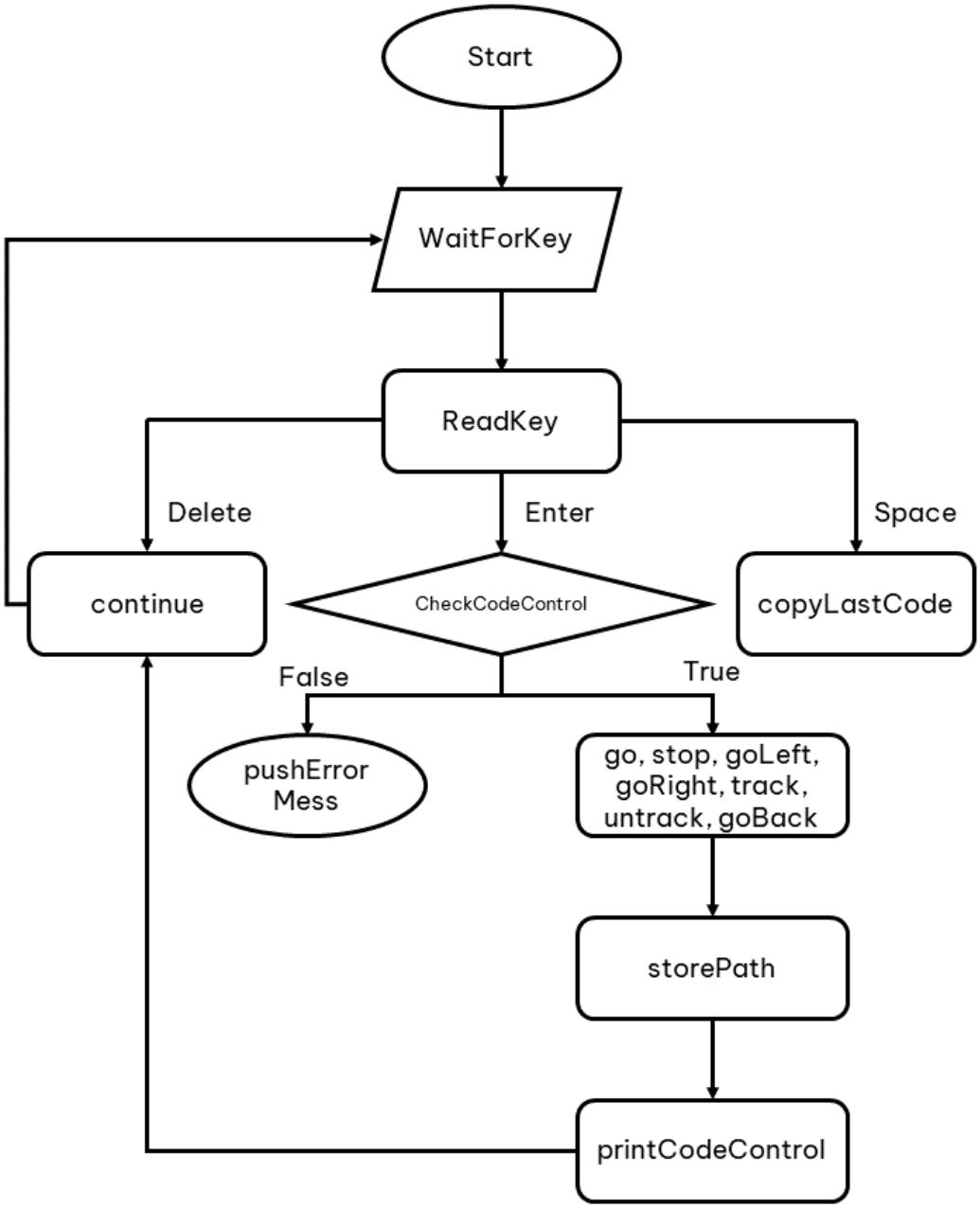
- Hãy lập trình để Marsbot có thể hoạt động như đã mô tả.
- Đồng thời bổ sung thêm tính năng: mỗi khi gửi một mã điều khiển cho Marsbot, hiển thị mã đó lên màn hình console để người xem có thể giám sát lộ trình của xe.

2. <u>Phân tích cách thực hiện:</u>

a) 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: Marsbot 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.

b) Lưu đồ thuật toán



3. <u>Ý nghĩa các hàm trong mã nguồn:</u>

a) 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.

b) *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.

c) *pushErrorMess*

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

Sử dụng các hàm syscall 4, 55.

d) *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.

e) *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

f) *copyLastCode*

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

g) *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.

h) *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.

i) *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.

j) 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.

k) 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.

l) 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.

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

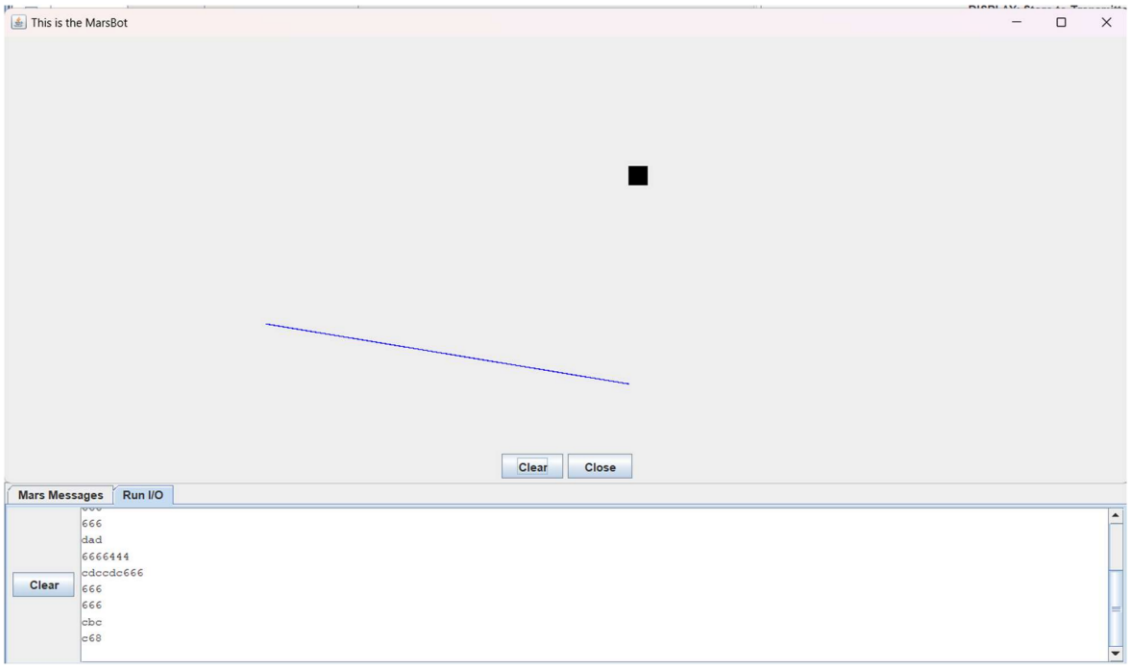
Lần lượt quét các hàng của Digital Lab Sim để xem phím nào được bấm bằng cách so sánh với các giá trị được lưu trong ***Key value***. Tiếp đó dựa vào mã được trả về ghi kí tự tương ứng vào bộ nhớ.

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).

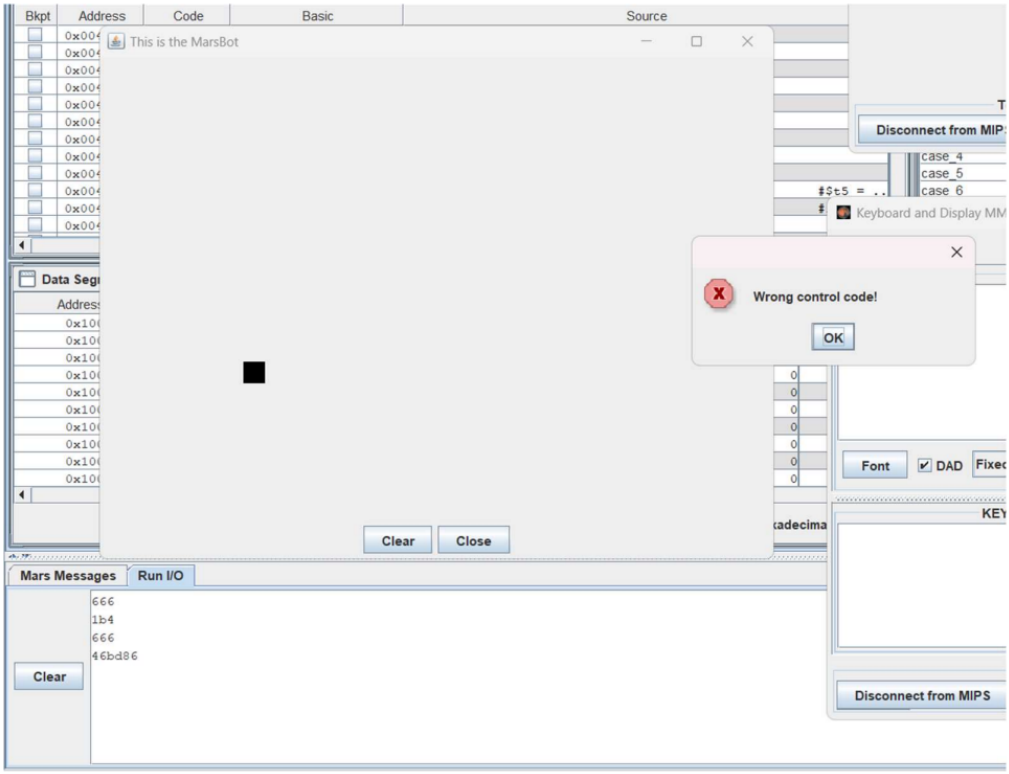
Vì thanh ghi PC vẫn chứa địa chỉ của lệnh mà ngắt xảy ra, tức là lệnh đã thực hiện xong, chứ không chứa địa chỉ của lệnh kế tiếp. Bởi vậy cần lập trình để tăng địa chỉ chứa trong thanh ghi epc.

4. <u>Mã nguồn:</u> nằm trong file ***n01_g05_NguyenVanCuong.asm***

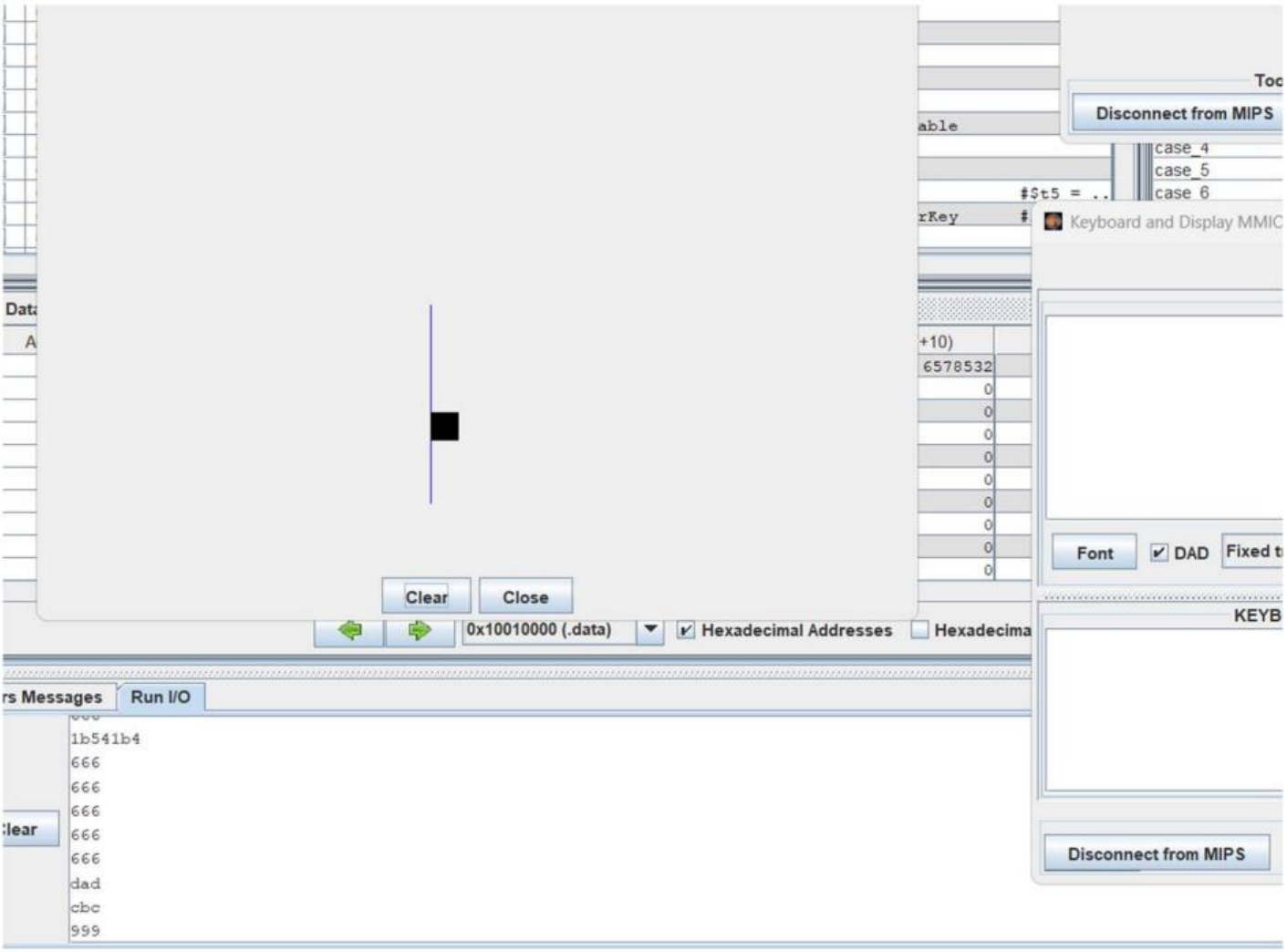
5. <u>Mô phỏng:</u>



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



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



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

II. Project 7 *(Người thực hiện: Bùi Anh Đức – 20210195)*

1. <u>Đề bài:</u> Chương trình kiểm tra cú pháp lệnh MIPS

Trình biên dịch của bộxử lý MIPS sẽ tiến hành kiểm tra cú pháp các lệnh hợp ngữ trong mã nguồn, xem có phù hợp về cú pháp hay không, rồi mới tiến hành dịch các lệnh ra mã máy. Hãy viết một chương trình kiểm tra cú pháp của 1 lệnh hợp ngữ MIPS bất kì (không làm với giả lệnh) như sau:

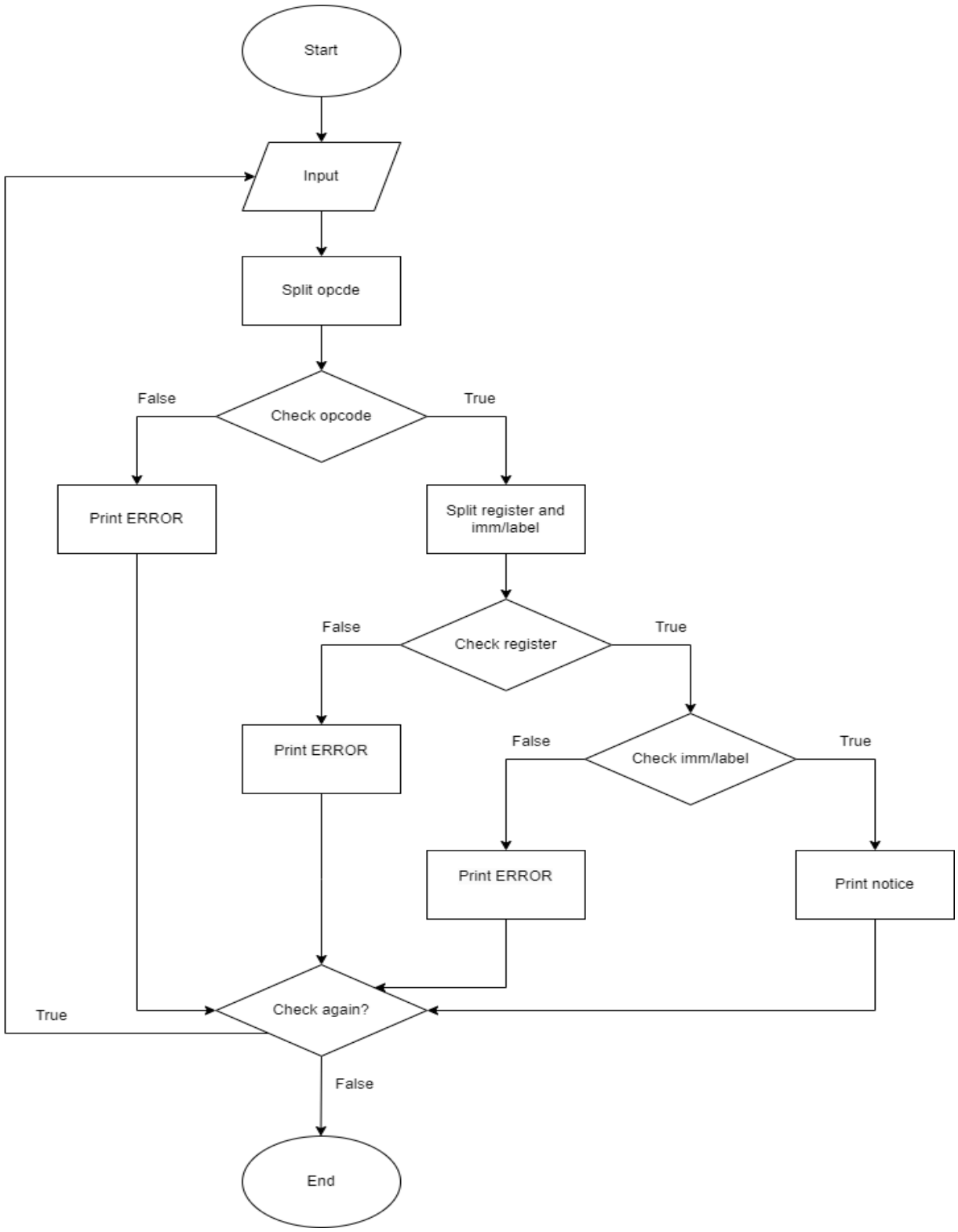
- Nhập vào từ bàn phím một dòng lệnh hợp ngữ. Ví dụ: *beq s1,31,t4*
- Kiểm tra xem mã opcode có đúng hay không? Trong ví dụ trên, opcode là beq là hợp lệ thì hiện thị thông báo *"opcode: beq, hợp lệ"*
- Kiểm tra xem tên các toán hạng phía sau có hợp lệ hay không? Trong ví dụ trên, *toán hạng s1 là hợp lệ, 31 là không hợp lệ, t4 thì khỏi phải kiểm tra nữa vì toán hạng trước đã bị sai rồi*.

2. Phân tích cách thực hiện:

a) Mô tả

- B1: Người dùng nhập một câu lệnh.
- B2: Tách câu lệnh ra để lấy opcode của câu lệnh này.
- B3: Kiểm tra opcode này có đúng không. Nếu đúng thì kiểm tra xem nó là loại opcode nào. Nếu sai thì in thông báo câu lệnh không chính xác ra màn hình.
- B4: Tách câu lệnh ra để lấy được thanh ghi thứ nhất và thanh ghi thứ 2/imm/label.
- B5: Kiểm tra thanh ghi đầu tiên có đúng không. Nếu đúng thì chuyển đến Bước 6, còn nếu sai thì in thông báo câu lệnh không chính xác ra màn hình.
- B6: Kiểm tra thanh ghi thứ 2/imm/label có đúng không. Nếu đúng thì in thông báo câu lệnh chính xác, còn nếu sai thì in thông báo câu lệnh không chính xác ra màn hình.
- B7: Kiểm tra xem người dùng có muốn tiếp tục kiểm tra hay không. Nếu có thì quay lại Bước 1, còn nếu không thì kết thúc chương trình.

b) Lưu đồ thuật toán



3. <u>Ý nghĩa các hàm trong mã nguồn:</u>

a) Split_opcode:

Ý nghĩa: chia một chuỗi (lệnh nhập vào) thành các chuỗi nhỏ (opcode, các thanh ghi, ...)

- Bước 1: khởi tạo các biến và thanh ghi

- Bước 2: Vòng lặp Loop1 lấy các ký tự từ chuỗi ban đầu (lệnh nhập vào) và kiểm tra xem nếu nó là NULL thì kết thúc vòng lặp, nếu nó là space thì bỏ qua và kiểm tra ký tự tiếp theo, còn nếu không thì tiếp tục chạy
- Bước 3: Vòng lặp Loop2 cũng tương tự như Loop 1 nhưng nó tiếp tục từ vị trí kết thúc Loop1 và kết thúc khi gặp newline.

b) Check_opcode:

Ý nghĩa: kiểm tra xem câu lệnh đầu vào có thuộc loại câu lệnh nào không (quy ước có các loại câu lệnh như: R, R1, R2, I, I1, J, J1, L, L1 và câu lệnh đặc biệt)

Lần lượt kiểm tra đầu vào có trùng với các opcode đúng không. Nếu có thì câu lệnh đầu vào thuộc loại đó, còn nếu không thì câu lệnh sai.

c) Check_Register_and_Number

Ý nghĩa: kiểm tra câu lệnh đầu vào có đúng cú pháp hay không

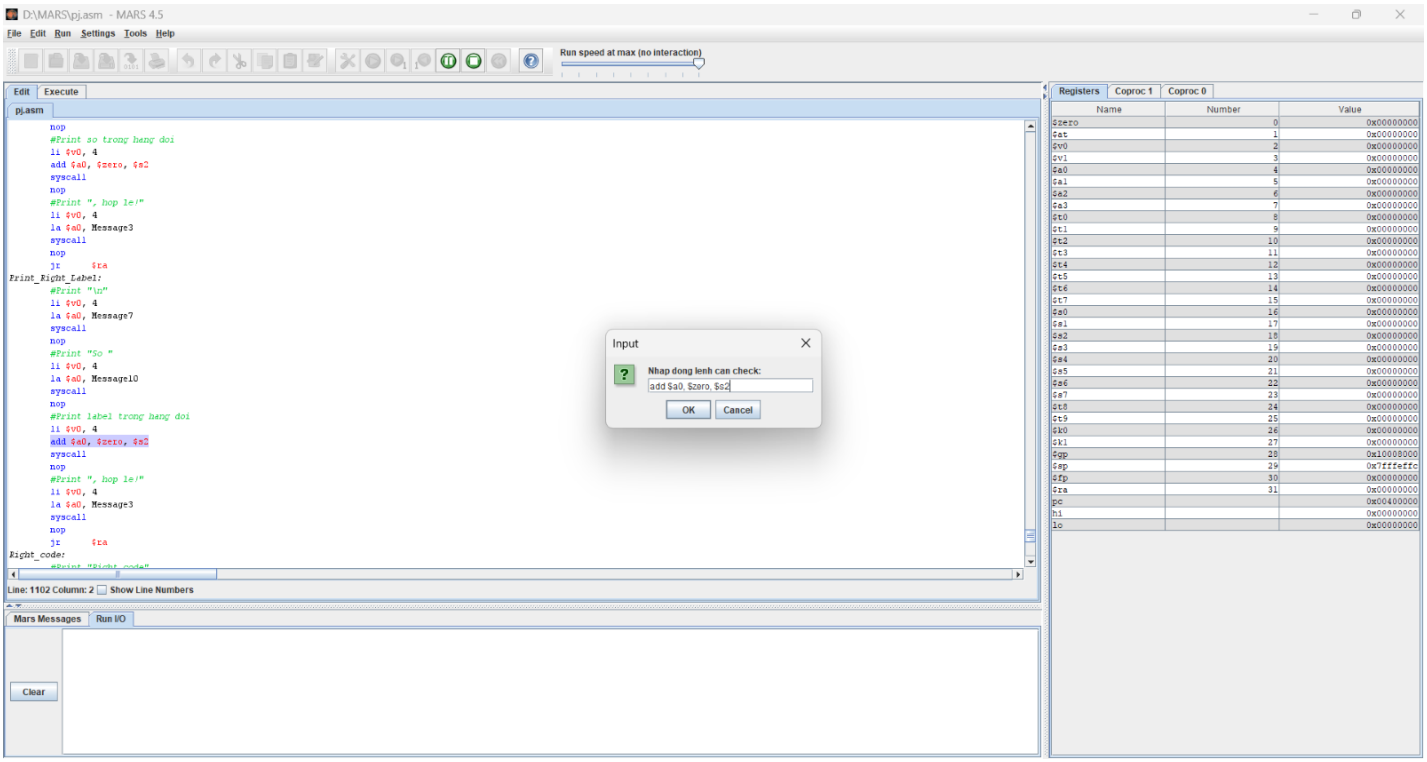
Nếu đã check_opcode và câu lệnh đầu vào thuộc loại nào, ta jump đến đoạn check của loại câu lệnh tương ứng, sau đó check các thanh ghi và imm/label sau đó. Nếu tất cả đều đúng thì câu lệnh đầu vào là đúng cú pháp, còn nếu không thì câu lệnh sai.

d) Kết thúc

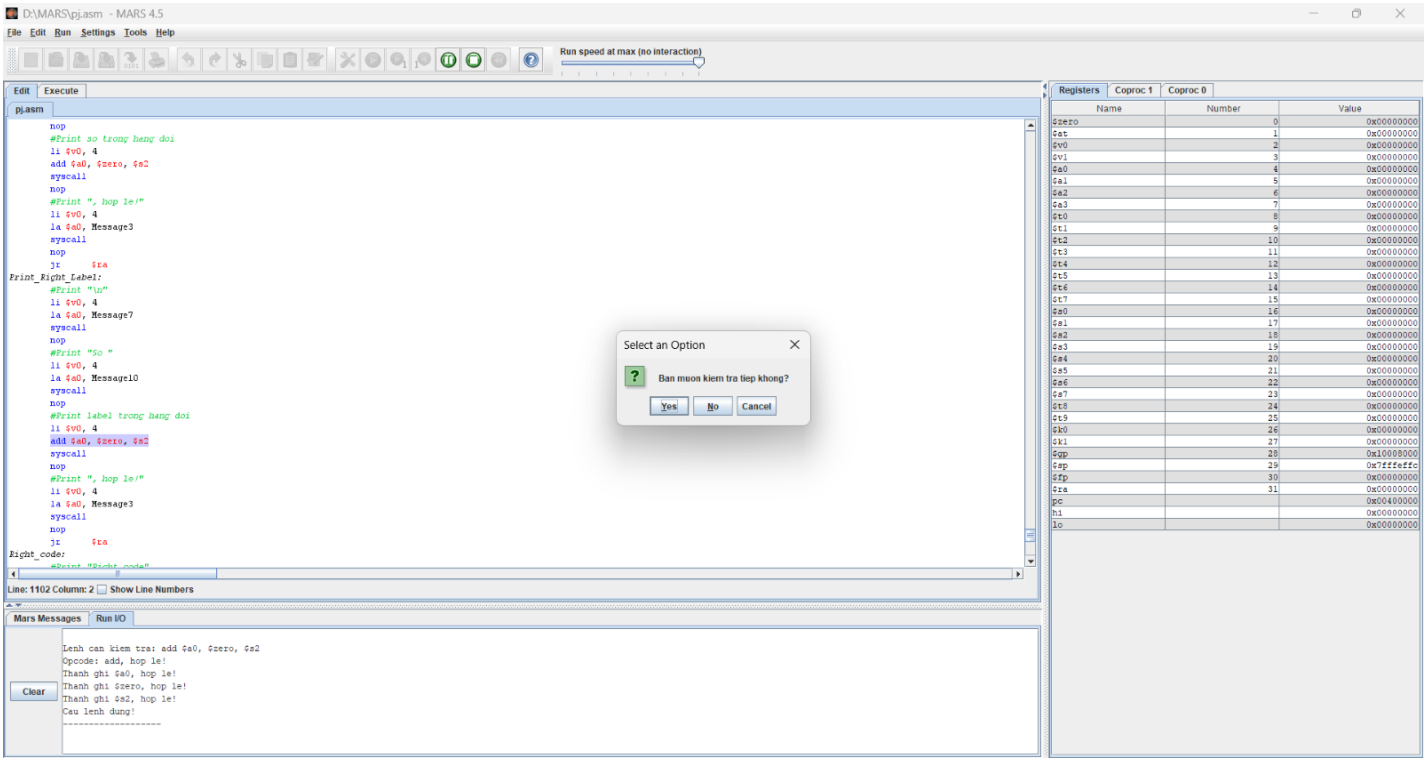
Ý nghĩa: In thông báo ứng với các trường hợp và hỏi xem người dùng có muốn kiểm tra câu lệnh tiếp theo hay không. Nếu có thì reset các thanh ghi và quay lại đầu chương trình, nếu không thì kết thúc chương trình.

4. <u>Mã nguồn:</u> nằm trong file ***n07_g05_BuiAnhDuc.asm***

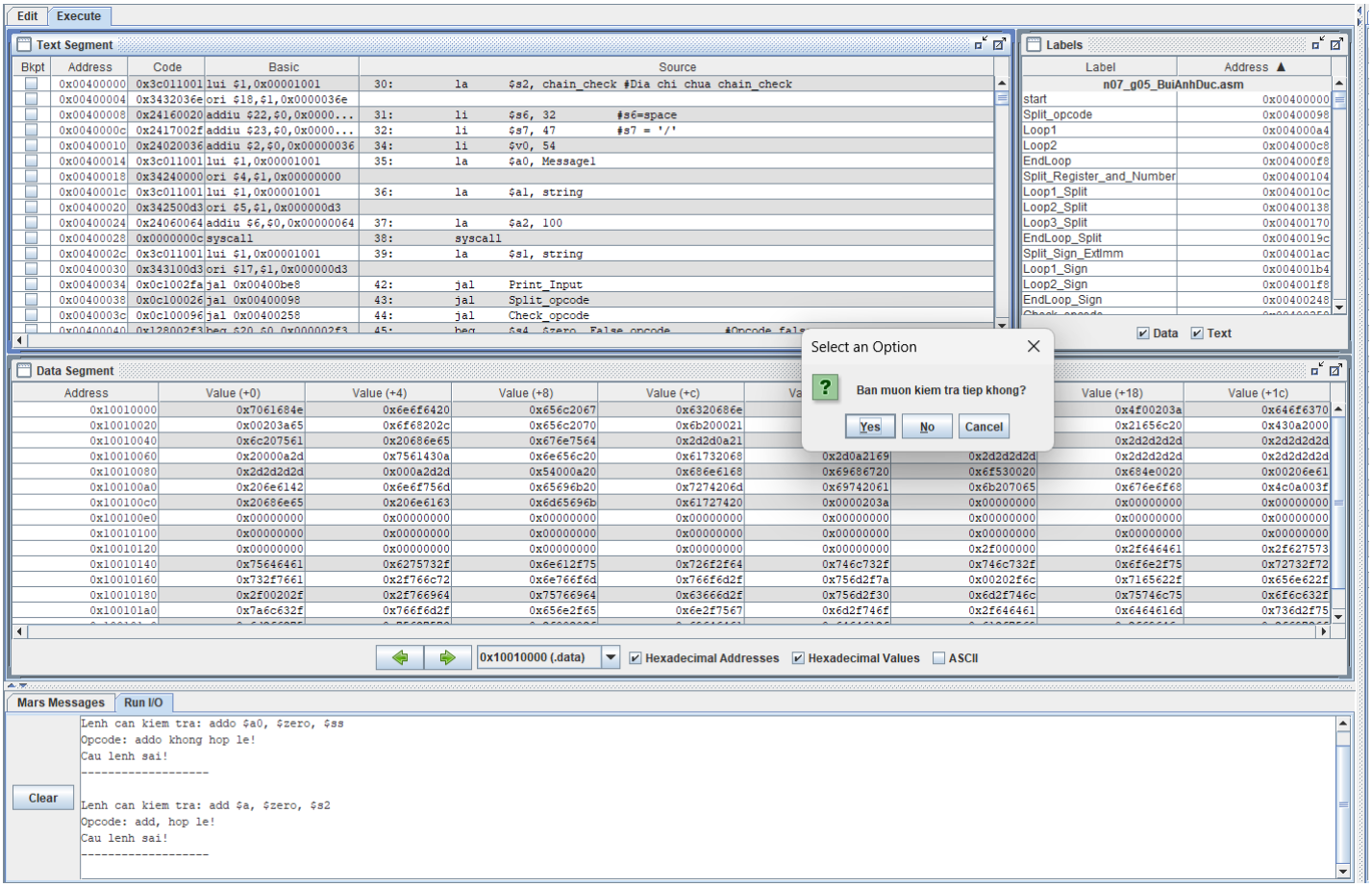
5. <u>Mô phỏng:</u>



Hình 1. Nhập câu lệnh



Hình 2. In kết quả và hỏi xem người dùng có kiểm tra tiếp không



Hình 3. Một vài trường hợp câu lệnh sai