**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

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

A picture containing text, sign

Description automatically generated**---🙠**🕮**🙢---**

**BÁO CÁO BÀI TẬP LỚN IT3280E - 131102**

**ASSEMBLY LANGUAGE AND COMPUTER ARCHITECTURE LAB**

**Nhóm 13**

***Giáo viên hướng dẫn: Đỗ Công Thuần***

***Sinh viên thực hiện: Trần Đức Huy -20194780***

***Nguyễn Ngọc Tú-20194873***

**Năm học 2021 - 2022**

Mục Lục:

[I. Project 4: Postscript CNC Marsbot 3](#_Toc109323377)

[1. Mô tả project 3](#_Toc109323378)

[2. Phân tích 4](#_Toc109323379)

[3. Thuật toán 5](#_Toc109323380)

[4. Mô phỏng 5](#_Toc109323383)

[5. Hình ảnh kết quả mô phỏng 6](#_Toc109323384)

[II. Project 5: Biểu thức trung tố hậu tố 8](#_Toc109323385)

[1. Mô tả project 8](#_Toc109323386)

[2. Phân tích 8](#_Toc109323387)

[3. Thuật toán 9](#_Toc109323388)

[4. Mô phỏng 11](#_Toc109323389)

[5. Hình ảnh kết quả mô phỏng 12](#_Toc109323390)

# Project 4: Postscript CNC Marsbot

## Mô tả project

Máy gia công cơ khí chính xác CNC Marsbot được dùng để cắt tấm kim loại theo các đường nét được qui định trước. CNC Marsbot có một lưỡi cắt dịch chuyển trên tấm kim loại, với giảđịnh rằng:

-Nếu lưỡi cắt dịch chuyển nhưng không cắt tấm kim loại, tức là Marsbot di chuyển nhưng không đểlại vết (Track)

-Nếu lưỡi cắt dịch chuyển và cắt tấm kim loại, tức là Marsbot di chuyển và có đểlại vết.Đểđiều khiển Marsbot cắt đúng như hình dạng mong muốn, người ta nạp vào Marsbot một mảng cấu trúc gồm 3 phần tử:

-<Góc chuyển động>, <Thời gian>, <Cắt/Không cắt>

-Trong đó <Góc chuyển động> là góc của hàm HEADING của Marsbot

-<Thời gian> là thời gian duy trì quá trình vận hành hiện tại

-<Cắt/Không cắt> thiết lập lưu vết/không lưu vết

Hãy lập trình đểCNC Marsbot có thể:

-Thực hiện cắt kim loại như đã mô tả

-Nội dung postscript được lưu trữ cố định bên trong mã nguồn

-Mã nguồn chứa 3 postscript và người dùng sử dụng 3 phím 0, 4, 8 trên bàn phím Key Matrix để chọn postscript nào sẽ được gia công.-Một postscript chứa chữ DCE cần gia công. Hai script còn lại sinh viên tự đề xuất (tối thiểu 10 đường cắt)

A picture containing graphical user interface

Description automatically generated Text

Description automatically generated

## Phân tích

Theo đề bài, có hai yêu cầu chính cần thực hiện:

* Gia công script DCE
* Gia công thêm 2 script (tự đề xuất) tối thiếu 10 đường cắt)

Sử dụng Digital Lab Sim and Mars Bot để hỗ trợ thực hiện chương trình.

Sử dụng phương pháp Polling để phát hiện phím được nhấn trong Digital Lab Sim. Để phát hiện một khóa trong ma trận khóa, chỉ số của mỗi hàng sẽ được ghi vào byte tại địa chỉ 0xFFFF0012 và sau đó là byte tại địa chỉ 0xFFFF0014 sẽ được đọc để biết phím nào đã được nhấn.

Bằng cách đặt các giá trị thích hợp trong HEADING, LEAVETRACK và MOVE các vị trí, MarsBot có thể thay đổi hướng, bật / tắt bản vẽ đường và

tạm dừng hoặc tiếp tục chuyển động của nó và tạo thành đường vẽ mong muốn.

## Thuật toán

### Thuật toán sử dụng

-Phương pháp Polling

Polling:

store $t4, 0(0xFFFF0012)

load $a0, 0(0xFFFF0014)

If ($a0 == NUM\_0)

Print postscript 1

Else if($a0 == NUM\_4)

Print postscript 2

Else if($a0 == NUM\_8)

Print postscript 3

Else if($a0 == NUM\_C)

Print postscript 4

Else

Back to polling

=> khi các số 0,4,8,C không được chọn thì quay lại đọc tiếp

-Xử lí marsbot theo từng giá trị Rotate, Time, Track hay Untrack:  
 + Xử lí theo từng kí tự qua 3 hàm:

Hàm SOLVE\_ROTATE: đọc số đo góc, gặp dấu ‘,’ thì xuống SOLVE \_TIME

Hàm SOLVE \_TIME: đọc thời gian di chuyển, gặp dấu ‘,’ thì xuống

SOLVE\_TRACK

Hàm SOLVE\_TRACK: đọc kiểm tra ghi

+ INCREASE\_2\_BIT: sau khi kiểm tra ghi xong tăng 2 bit để bỏ qua dấu ’;’

### Ý nghĩa của 1 số thanh ghi

$a1 lưu địa chỉ của postscript

$t0 lưu giá trị rotate

$t1 lưu giá trị của time

$t2 IN\_ADRESS\_HEXA\_KEYBOARD

$t3, OUT\_ADRESS\_HEXA\_KEYBOARD

$t4 đọc từng kí tự (số, dấu phẩy, dấu chấm phẩy)

$t5 giá trị địa chỉ đang trỏ tới $t4

$t6 số bit đã dịch

## Mô phỏng

N04\_g13\_TranDucHuy.asm

## Hình ảnh kết quả mô phỏng Graphical user interface Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

A picture containing graphical user interface

Description automatically generated

# Project 5: Biểu thức trung tố hậu tố

## Mô tả project

Viết chương trình tính giá trị biểu thức bất kỳ bằng phương pháp duyệt biểu thức hậu tố.

Các yêu cầu cụ thể:

1. Nhập vào biểu thức trung tố, ví dụ: 9 + 2 + 8 \* 6

2. In ra biểu thức ở dạng hậu tố, ví dụ: 9 2 + 8 6 \* +

3. Tính ra giá trị của biểu thức vừa nhập

Các hằng số là số nguyên, trong phạm vi từ 0 → 99.

Toán tử bao gồm phép cộng, trừ, nhân, chia lấy thương

## Phân tích

Theo đề bài, có hai yêu cầu chính cần thực hiện:

* Chuyển biểu thức nhập vào từ dạng trung tố sang dạng hậu tố
* Tính giá trị của biểu thức vừa nhập

Sử dụng Keyboard and Display MMIO Simulator để hỗ trợ thực hiện chương trình.

Dùng một stack để lưu giá trị của toán tử và một stack để lưu giá trị của toán hạng. Tạo stack từ array.  Coi stack1 là stack để lưu toán tử, stack2 là stack để lưu các toán hạng

Khi người dùng nhập ký tự từ bàn phím vào khu vực input của MMIO, ký tự vừa nhập vào sẽ được phân loại

* Nếu là ‘(‘ sẽ được đưa vào stack1
* Nếu là ‘)’ thì các toán tử chứa trong stack1 sẽ được đưa ra khỏi stack1 và in ra màn hình cho đến khi gặp toán tử ‘(’
* Nếu là ‘+’,’-’: Kiểm tra toán tử trên cùng của stack1, nếu là ‘(‘ thì đẩy toán tử hiện tại vào toán tử, các trường hợp còn lại thì đưa toán tử đó ra khỏi stack1, in ra màn hình và đẩy toán tử hiện tại vào stack
* Nếu là ‘\*’,’/’: Kiểm tra toán tử trên cùng của stack1, nếu là ‘\*’,’/’  thì đưa toán tử đó ra khỏi stack1, in ra màn hình và đẩy toán tử hiện tại vào stack, các trường hợp còn lại thì chỉ đưa toán tử hiện tại vào stack1.
* Nếu là các số từ 0 đến 9 thì đẩy các số vào stack2. Nếu nhập là 2 số liên tiếp ( số có 2 chữ số) thì số hạng trên đỉnh stack2 sẽ được đưa ra khỏi stack2, nhân 10 và cộng với số hạng vừa nhập sau đó được đẩy lại vào stack2.
* Nếu là enter thì duyệt toàn bộ stack, đưa tất cả các toán tử ra khỏi stack và in ra màn hình theo thứ tự cho đến khi stack rỗng, reset và bắt đầu lại chương trình
* Các trường hợp còn lại bỏ qua.

## Thuật toán

$s7: dùng để kiểm tra nếu giá trị nhập vào là số có 2 chữ số

Default: $s7 = 0

$s7 = 1 -> 1 chữ số

$s7 = 2 -> 2 chữ số

$t0 : giá trị được nhập vào từ MMIO

Switch( $t0):

case ‘(‘:

$t3.push($t0)

case ‘)’:

while( $s6 != ‘)’ )

$t3.pop

Print $s6

$s3 = $s3 - 1

case ‘+’, ‘ – ‘:

$s7 = $s7-1

if $s6 == ‘)’ :

$t3.push($t0)

else:

$t3.pop()

print $s6

$t3.push($t0)

case ‘\*’, ’/ ’:

$s7 = $s7 -1

if( $s6 == ‘\*’ || $s6 == ‘/’)

$t3.pop

Print $s6

$t3.push( $t0)

case ‘0’…’9’:

print $t0

$s7 = $s7+1

$s2 = $t0 ( in decimal)

If $s7 == 1: số một chữ số

$t4 = $s2

$f0 = $t4 ( in single float)

$t5.push ( $f0)

If $s7 ==2: số có 2 chữ số

$s7 = $s7 -1

$t4 = $t4\*10 + $s2

$f0 = $t4 (in single float)

$t5.pop()

$t5.push ( $f0)

case press enter:

if( $s3 == -1)

print the result

jump to init

else:

while( $s3!= -1)

$t3.pop

Print $s6

Print the result

Jump to init

default:

$t3: char[] :chứa địa chỉ của stack dùng để lưu giá trị của toán tử

$s3: index i, con trỏ trỏ đến giá trị trên cùng của stack

$s6: giá trị tại con trỏ i đang trỏ đến, cũng là top của stack

Push:

$s3 = $s3+1

$s6 = $t0

Pop:

$f1 = $t5.pop

$f2 = $t5.pop

Switch ( $s6):

Case ‘+’:

$f0 = $f1 + $f2

$t5.push( $f0)

$s3 = $s3 – 1

Return

Case ‘-‘:

$f0 = $f2 - $f1

$t5.push ( $f0)

$s3 = $s3 – 1

return

Case ‘\*’:

$f0 = $f2 \* $f1

$t5.push ( $f0)

$s3 = $s3 – 1

return

Case ‘/’:

$f0 = $f2 / $f1

$t5.push ( $f0)

$s3 = $s3 – 1

return

$s3 = $s3 -1

return

$t5:  float[] :chứa địa chỉ của stack  dùng để lưu giá trị của toán hạng

$t7: index i, con trỏ trỏ đến giá trị trên cùng của stack

$f0: top

Push:

$t7 = $t7 + 1

$f0 = $f0

Pop:

$t7 = $t7 – 1

## Mô phỏng

N05\_g13\_NguyenNgocTu.asm

## Hình ảnh kết quả mô phỏng

Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated