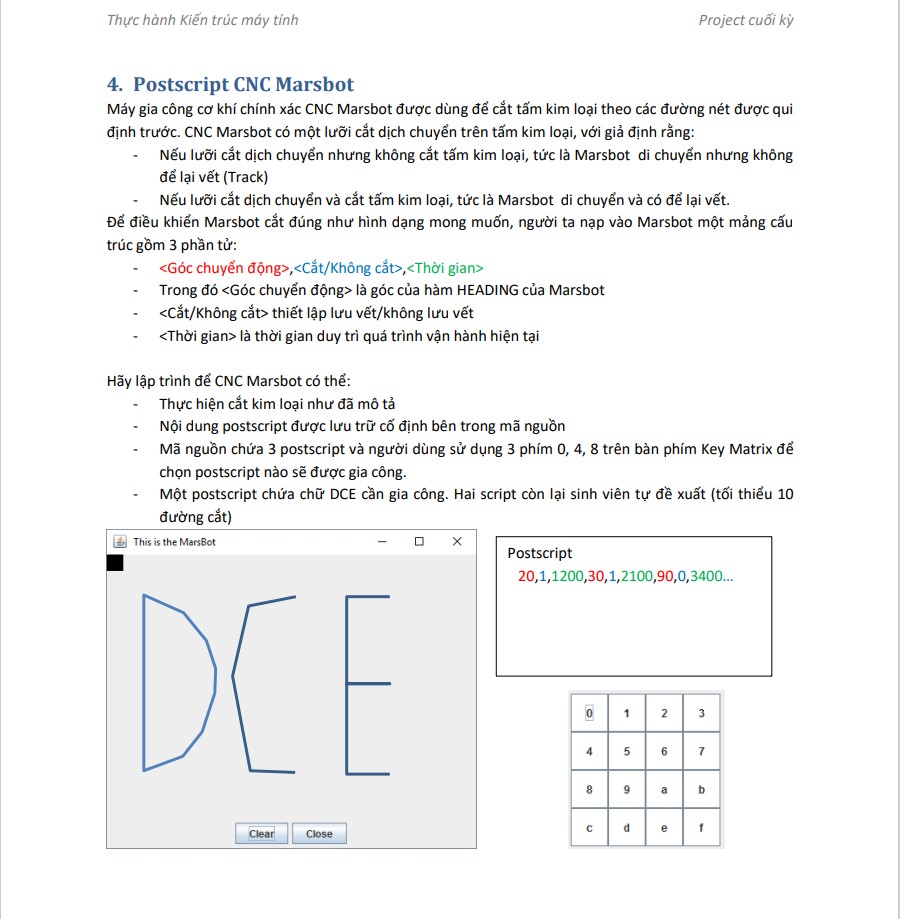
# Project 4: Postscripts CNC MarsBot

**YÊU CẦU BÀI TOÁN**



1. **Giới thiệu**

Chương trình được viết bằng ngôn ngữ MIPS Assembly nhằm điều khiển một robot di chuyển theo các hướng dẫn được lưu trữ trong các đoạn mã (postscript). Mỗi postscript chứa thông tin về góc quay, trạng thái vẽ và thời gian di chuyển của robot. Chương trình cho phép người dùng chọn một trong ba postscript thông qua bàn phím và robot sẽ thực hiện các hướng dẫn tương ứng.

1. **Cấu trúc dữ liệu**

* **Postscript**: Mỗi postscript là một mảng các giá trị bao gồm ba phần tử: góc quay (degree), trạng thái vẽ (leave track), và thời gian di chuyển (duration).
* **postscript1, postscript2, postscript3**: Các mảng chứa các giá trị hướng dẫn cho robot.
* **postscript1\_length, postscript2\_length, postscript3\_length**: Độ dài của mỗi mảng postscript.

1. **Biến và hằng số**

* **IN\_ADDRESS\_HEXA\_KEYBOARD**: Địa chỉ nhận đầu vào từ bàn phím
* **OUT\_ADDRESS\_HEXA\_KEYBOARD**: Địa chỉ của đầu ra bàn phím
* **HEADING**: Địa chỉ để nhận góc quay của robot.
* **LEAVETRACK**: Địa chỉ để điều khiển trạng thái vẽ của robot (0 hoặc 1)
* **WHEREX, WHEREY**: Địa chỉ để lấy vị trí hiện tại của robot.
* **MOVING**: Địa chỉ để điều khiển trạng thái di chuyển của robot.

1. **Ý tưởng**

Đầu tiên phải tạo mảng để chứa postscript, các postscript đã chọn ngoài DCE là

• KHAI(là 27 nét vẽ với K là 3 nét vẽ, H là 3 nét vẽ, A là 20 nét vẽ, I là 1 nét)

• HUY (là 26 nét vẽ với H là 3 nét vẽ U là 20 nét vẽ và Y là 3 nét vẽ)

• Xử lí Key Matrix – sử dụng Digital Lab Sim để có bộ 16 key tương ứng hình ảnh ở yêu cầu đề bài

• Do cơ chế cơ bản của MarsBot nên ta sẽ tạo 1 vòng lặp để xử lí từng đường cắt một được lấy từ postscript (xử lí 3 phần tử của mảng mỗi lần)

• Và khi vẽ xong cả 3 postscript đều được hoàn thành -> kết thúc chương trình.

**SOURCE CODE**

.data

postscript1:    .word       180,1,6000, 90,1,1500, 60,1,1000, 0,1,5000, 300,1,1000, 270,1,1500, 90,0,6500, 270,1,1500, 240,1,1000, 180,1,5000, 120,1,1000, 90,1,1500, 90,0,2500, 270,1,1500, 0,1,3000, 90,1,1500, 270,0,1500, 0,1,3000, 90,1,1500

postscript1\_length: .word   57

postscript2:    .word       180,1,8600, 0, 0, 4300, 135,1,6080, 315,0,6080, 45,1,6080, 225,0,6080, 0,0,4300, 90,0,6000, 180,1,8600, 90,0,4060, 0,1,8600, 180,0,4300, 270,1,4060, 180,0,4300, 90,0,6000, 0,1,6100, 10,1,470, 20,1,470, 30,1,470, 40,1,470, 50,1,470, 60,1,470, 70,1,470, 80,1,470, 90,1,20, 100,1,470, 110,1,470, 120,1,470, 130,1,470, 140,1,470, 150,1,470, 160,1,470, 170,1,470, 180,1,6100, 0,0,4300, 270,1,5000, 180,0,4300, 90,0,7000, 0,1,8600

postscript2\_length: .word   117

postscript3:    .word       180,1,8600, 90,0,4600, 0,1,8600, 180,0,4300, 270,1,4600, 90,0,6000, 0,0,4310, 180,1,6100, 170,1,470, 160,1,470, 150,1,470, 140,1,470, 130,1,470, 120,1,470, 110,1,470, 100,1,470, 90,1,20, 80,1,470, 70,1,470, 60,1,470, 50,1,470, 40,1,470, 30,1,470, 20,1,470,10,1,470, 0,1,6100, 90,0,1500,150,1,5000,30,1,5000,210,0,5000,180,1,4300,180,0,3000

postscript3\_length: .word   96

Pleasechoose:   .asciiz     "Press 0 to print DCE\nPress 4 to print KHAI\nPress 8 to print HUY\n\n"

Messenger1: .asciiz     "You choose to print DCE!\n"

enter:  .asciiz     "\n"

Messenger2: .asciiz     "You choose to print KHAI!\n"

Messenger3:     .asciiz     "You choose to print HUY!\n"

space:  .asciiz

# declaring table

# postscript length = numberOfLines\*3

.eqv    IN\_ADDRESS\_HEXA\_KEYBOARD    0xFFFF0012

.eqv    OUT\_ADDRESS\_HEXA\_KEYBOARD   0xFFFF0014

.eqv    HEADING     0xffff8010  # Integer: An angle between 0 and 359

                    # 0 : North (up)

                    # 90: East (right)

                    # 180: South (down)

                    # 270: West (left)

.eqv    LEAVETRACK  0xffff8020  # Boolean (0 or non-0):

                    # whether or not to leave a track

.eqv    WHEREX      0xffff8030  # Integer: Current x-location of MarsBot

.eqv    WHEREY      0xffff8040  # Integer: Current y-location of MarsBot

.eqv    MOVING      0xffff8050  # Boolean: whether or not to move

.text

    addi    $s5, $0, 0      # count number of successful postscript

    addi    $t4, $0, 0      # $t4 là biến check xem postscript1 đã được vẽ chưa

    addi    $t5, $0, 0      # $t5 là biến check xem postscript2 đã được vẽ chưa

    addi    $t6, $0, 0      # $s6 là biến check xem postscript3 đã được vẽ chưa

                    # 0 - chưa vẽ, 1 - đã vẽ => $s5 += 1

                    # $s5 === 3 => cả 3 postscript đều đã được vẽ => complete

    addi    $s6, $0, 3      # $s6 = 3 lần vẽ

polling:

row1:

    li  $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

    li  $t2, OUT\_ADDRESS\_HEXA\_KEYBOARD

    li  $t3, 0x01               # check row 1 with key 0, 1, 2, 3

    sb  $t3, 0($t1)                 # must reassign expected row

    lb  $a0, 0($t2)                 # read scan code of key button

    bne $a0, 0x00000011, row2           # check trùng với 0 thì vẽ postscript1 nếu không thì check row2

draw0:  li  $v0, 4

    la  $a0, Messenger1

    syscall

    la  $t8, postscript1

check0: bne $t4, 0, postscript1\_already\_done    # nếu poscript1 đã được vẽ 1 lần trước đó rồi thì $t5=1 và không tăng $t4 nữa

    li  $t4, 1                  # postscript1 sẽ done

    addi    $s5, $s5, 1             # done 1/3 postscript

postscript1\_already\_done:

    la  $t7, postscript1\_length         # gán $t7=length của mảng

    lw  $t7, 0($t7)

    j   main

    nop

row2:

    li  $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

    li  $t2, OUT\_ADDRESS\_HEXA\_KEYBOARD

    li  $t3, 0x02               # check row 2 with key 4, 5, 6, 7

    sb  $t3, 0($t1)                 # must reassign expected row

    lb  $a0, 0($t2)                 # read scan code of key button

    bne $a0, 0x00000012, row3           # 4 - postscript2

draw4:  li  $v0, 4

    la  $a0, Messenger2

    syscall

    la  $t8, postscript2

check4: bne     $t5, 0, postscript2\_already\_done    # nếu poscript2 đã được vẽ 1 lần trước đó rồi thì $t6=1 và không tăng $t4 nữa

    li  $t5, 1                  # postscript2 done

    addi    $s5, $s5, 1             # done 1 postscript

postscript2\_already\_done:

    la  $t7, postscript2\_length         # gán $t7=length của mảng

    lw  $t7, 0($t7)

    j   main

    nop

row3:

    li  $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

    li  $t2, OUT\_ADDRESS\_HEXA\_KEYBOARD

    li  $t3, 0x04               # check row 3 with key 8, 9, A, B

    sb  $t3, 0($t1)                 # must reassign expected row

    lb  $a0, 0($t2)                 # read scan code of key button

    bne $a0, 0x00000014, invalid        # 8 - postscript3

draw8:  li  $v0, 4

    la  $a0, Messenger3

    syscall

    la  $t8, postscript3

check8: bne $t6, 0, postscript3\_already\_done    # nếu poscript3 đã được vẽ 1 lần trước đó rồi thì $s5=1 và không tăng $t4 nữa

    li  $t6, 1                  # postscript3 done

    addi    $s5, $s5, 1             # done 1 postscript

postscript3\_already\_done:

    la  $t7, postscript3\_length         # gán $t7=length của mảng

    lw  $t7, 0($t7)

    j   main

    nop

invalid:

    li  $v0, 4

    la  $a0, Pleasechoose

    syscall

sleep\_wait:

    li  $a0, 1000               # đợi 1000ms

    li  $v0, 32

    syscall

    j   polling

main:

# Go to cut area

    jal     UNTRACK         # no draw track line

    addi    $s2, $zero, 135     # Marsbot rotates given radius and start

start\_running:

    jal     ROTATE

    jal     GO

start\_sleep:

    addi    $v0,$zero, 32       # Keep running by sleeping in 5000 ms

    addi    $a0,$zero, 5000

    syscall

    jal     UNTRACK         # keep old track

    li  $s0, 0          # Set index counter for postscript array

loop:

    beq $s0, $t7, end\_loop  # if i == numberOfLines\*3 then quit

    sll     $s1, $s0, 2     # s1 = 4i

    add $s1, $s1, $t8       # s1 = A[i]'s address

    lw  $s2, 0($s1)     # s2 = A[i]'s value - <Góc chuyển động>

    addi    $s1, $s1, 4

    lw  $s3, 0($s1)     # s3 = A[i+1]'s value - <Cắt/Không cắt>

    addi    $s1, $s1, 4

    lw  $s4, 0($s1)     # s4 = A[i+2]'s value - <Thời gian chạy>

    jal     TRACK\_UNTRACK       # draw track line/ or not

running:

    jal     ROTATE

    jal     GO

sleep:

    addi    $v0,$zero,32        # Keep running by sleeping bằng <Thời gian chạy>

    add     $a0,$zero,$s4

    syscall

    jal     UNTRACK         # keep old track

    addi    $s0, $s0, 3     # tăng lên 3 phần tử của mảng tiếp theo

    j   loop

end\_loop:

    jal STOP

    beq $s5, $s6, end\_main  # Nếu $s4 = $s6 thì dừng chương trình (với $s6 define bằng 3 ở ban đầu)

    j   polling         # Nếu chưa vẽ đủ 3 postscript thì tiếp tục polling

end\_main:

    li  $v0, 10

    syscall

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

# GO procedure, to start running

# param[in] none

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

GO:

    li  $at, MOVING         # change MOVING port

    addi    $k0, $zero,1        # to logic 1,

    sb  $k0, 0($at)         # to start running

    jr  $ra

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

# STOP procedure, to stop running

# param[in] none

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

STOP:

    li  $at, MOVING         # change MOVING port to 0

    sb  $zero, 0($at)       # to stop

    jr  $ra

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

# TRACK procedure, to start drawing line

# param[in] none

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

TRACK\_UNTRACK:

    li  $at, LEAVETRACK     # change LEAVETRACK port

    sb  $s3, 0($at)         # to start tracking/ or not

    jr  $ra

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

# UNTRACK procedure, to stop drawing line

# param[in] none

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

TRACK:

    li  $at, LEAVETRACK     # change LEAVETRACK port

    addi    $k0, $zero,1        # to logic 1,

    sb  $k0, 0($at)         # to start tracking

    jr  $ra

UNTRACK:

    li  $at, LEAVETRACK     # change LEAVETRACK port to 0

    sb  $zero, 0($at)       # to stop drawing tail

    jr  $ra

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

# ROTATE procedure, to rotate the robot

# param[in] $a0, An angle between 0 and 359

# 0 : North (up)

# 90: East (right)

# 180: South (down)

# 270: West (left)

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

ROTATE:

    li  $at, HEADING        # change HEADING port

    sw  $s2, 0($at)         # to rotate robot

    jr  $ra

**KẾT QUẢ**

* 1. A screenshot of a computer

     Description automatically generated**Postscript 1: DCE**

* 1. **Postscript 2: KHAI**

A screenshot of a computer

Description automatically generated

* 1. **Postscript 3: HUY**

**A screenshot of a computer

Description automatically generated**

**Giải thích code sơ bộ hay lưu đồ hoạt động**

1. **Khởi tạo các biến đếm**:
   * $s5: Đếm số postscript đã được thực hiện.
   * $t4, $t5, $t6: Các biến để kiểm tra xem postscript1, postscript2, postscript3 đã được vẽ hay chưa.
   * $s6: Tổng số postscript cần vẽ (3).
2. **Polling (chờ phím)**:
   * Chương trình liên tục kiểm tra đầu vào từ bàn phím để xác định người dùng đã nhấn phím nào.
   * Dựa vào giá trị đầu vào, chương trình quyết định sẽ vẽ postscript nào.
3. **Xử lý phím nhấn**:
   * Nếu phím 0 được nhấn: Thực hiện postscript1.
   * Nếu phím 4 được nhấn: Thực hiện postscript2.
   * Nếu phím 8 được nhấn: Thực hiện postscript3.
   * Nếu phím khác: Hiển thị thông báo chọn phím hợp lệ và tiếp tục chờ phím.
4. **Thực hiện postscript**:
   * Kiểm tra xem postscript tương ứng đã được thực hiện trước đó hay chưa.
   * Nếu chưa, đánh dấu đã thực hiện và tăng biến đếm $s5 khi $s5=3 là khi đã vẽ đủ 3 postscript thì dừng chương trình
   * Đọc độ dài của postscript và gọi hàm main để thực hiện.
5. **Hàm main**:
   * Lặp qua từng hướng dẫn trong postscript và điều khiển robot theo từng bước duyệt mỗi lần 3 phần tử của mảng để
     + **ROTATE**: Đặt góc quay cho robot (góc vẽ từ 0-359­o)
     + **GO**: Bắt đầu di chuyển robot.
     + **TRACK\_UNTRACK**: Thiết lập vẽ hoặc không vẽ (0 hoặc 1)
     + **SLEEP**: Tạm dừng trong khoảng thời gian được chỉ định.
6. **Kiểm tra hoàn tất**:
   * Sau khi thực hiện xong postscript, kiểm tra xem tất cả các postscript đã được thực hiện hay chưa nếu $s5=3 thì dừng chương trình nếu không thì tiếp tục vẽ đến khi nào vẽ đủ 3 postscript thì dừng

**Phân tích chi tiết**

1. **Polling**: Phần polling là một vòng lặp vô hạn kiểm tra đầu vào từ bàn phím và quyết định hành động tương ứng. Điều này đảm bảo chương trình luôn sẵn sàng nhận lệnh từ người dùng.
2. **Xử lý phím nhấn và kiểm tra postscript đã thực hiện**: Các biến $t4, $t5, $t6 giúp theo dõi trạng thái của mỗi postscript để tránh thực hiện lại các postscript đã được vẽ.
3. **Vòng lặp thực hiện postscript**: Mỗi postscript được thực hiện bằng cách lặp qua từng hướng dẫn trong mảng và thực hiện các hành động tương ứng. Điều này cho phép linh hoạt trong việc thêm, sửa hoặc xóa các hướng dẫn mà không cần thay đổi cấu trúc chương trình.

➢ **CÁC VẤN ĐỀ VÀ HẠN CHẾ**

* + - Việc **điều kiện thoát hẳn chương trình** (đến end\_main) sẽ chỉ đạt khi người dùng vẽ đủ 3 postscripts, vậy nên nếu người dùng cứ vẽ lại 1 postscript hoặc 2 postscripts thì chương trình sẽ không thoát mà phải chạy lại bằng tay –> hạn chế. Mặt khác người dùng có thể vẽ nhiều lần 1 postscript.
    - Mỗi lần vẽ xong 1 postscript người dùng sẽ phải **tự clear khu vực đã vẽ / cắt của MarsBot thủ công**, đồng thời **khi ấn 1 key** nào đó hợp lệ (Ví dụ: 0) thì phải **nhớ tắt key đó luôn** bởi nếu để quên thì nó sẽ tiếp tục nhận key đó tiếp và vẽ lại postscript dẫn đến phải tắt chương trình thủ công và chạy lại.