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

Báo cáo Thực hành Kiến trúc Máy tính

Giảng viên: THS.Lê Bá Vui

Sinh viên thực hiện:

Lê Thanh Giang - 20194541

Uông Hồng Minh - 20194625

MỤC LỤC

Bài 2

Sinh viên thực hiện: Lê Thanh Giang – 20194541

I. Đề bài

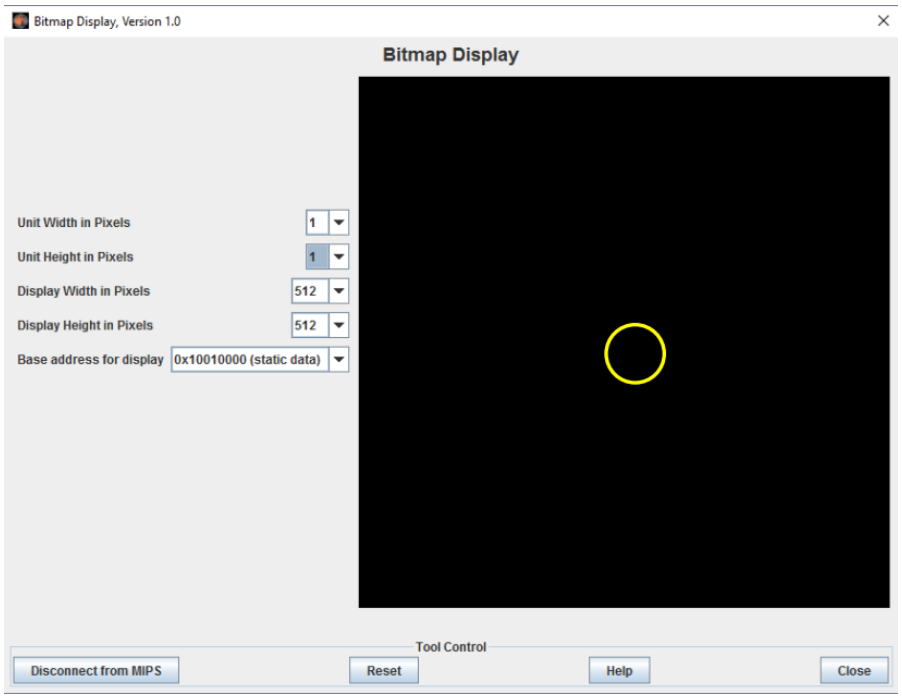
Viết chương trình vẽ một quả bóng hình tròn di chuyển trên màn hình mô phỏng Bitmap của Mars. Nếu đối tượng đập vào cạnh của màn hình thì sẽ di chuyển theo chiều ngược lại. Yêu cầu:

- Thiết lập màn hình ở kích thước 512x512. Kích thước pixel 1x1.

- Chiều di chuyển phụ thuộc vào phím người dùng bấm, gồm có (di chuyển lên (W), di chuyển xuống (S), sang trái (A), sang phải (D), tăng tốc độ (Z), giảm tốc độ (X) trong bộ giả lập Keyboard and Display MMIO Simulator).

- Vị trí bóng ban đầu ở giữa màn hình.

Gợi ý: Để làm một đối tượng di chuyển thì chúng ta sẽ xóa đối tượng ở vị trí cũ và vẽ đối tượng ở vị trí mới. Để xóa đối tượng chúng ta chỉ cần vẽ đối tượng đó với màu là màu nền.



II. Phân tích cách làm

Bài toán được chia thành 2 phần chính: Vẽ hình tròn và di chuyển bằng bàn phím

- Vẽ hình tròn:
  - + Hình tròn được vẽ dựa trên thuật toán Bresenham
  - + Mỗi điểm sẽ được tạo, tổng cộng là 8 điểm và được vẽ bằng Pixel
- Dịch chuyển
  - + Hình tròn được dịch chuyển khi mà tâm dịch chuyển và đồng thời xóa đường tròn đã vẽ lúc trước (tức là cho bằng màu nền) và sau đó lại thực hiện việc vẽ lại.

Chi tiết cụ thể:

- Thuật toán Bresenham code bằng ngôn ngữ C

```
void ve8diem(int x0,int y0,int x, int y, int color)
{
 putpixel( x0 + x , y0 + y ,color);
 putpixel( x0 - x , y0 + y ,color);
 putpixel( x0 + x , y0 - y ,color);
 putpixel( x0 - x , y0 - y ,color);
 putpixel( x0 + y , y0 + x ,color);
 putpixel( x0 - y , y0 + x ,color);
 putpixel( x0 + y , y0 - x ,color);
 putpixel( x0 - y , y0 - x ,color);
}
void circle(int x0,int y0,int r)
{
 int x=0;int y=r;
 int p=3-2*r;
 while (x<=y)
 {
  ve8diem(x0,y0,x,y,15);
  if(p<0) p=p+4*x+6;
  else
     {
        p=p+4*(x-y)+10;
        y=y-1;
     }
  x=x+1;
 }
}
```

- Dịch chuyển

  + Coi như màn hình được chia thành 4 ô tương ứng 2 trục x và y , với tâm là chính giữa màn hình tọa độ O(0,0)
  + Khi dịch chuyển sang trái, phải, trên, dưới chỉ cần thay đổi chiều của trục và tăng/ giảm đơn vị dịch chuyển tâm tương ứng dịch chuyển trên/dưới hoặc trái/phải

III. Mã nguồn

```
################################################################################
#                                    Ex 2                      #
################################################################################
#                      Le Thanh Giang                                  #
################################################################################
#                                                                              #
#      Bitmap Display Settings:                          #
#           Unit Width: 1                                                      #
#           Unit Height: 1                                                     #
#           Display Width: 512                                                 #
#           Display Height: 512                                                #
#           Base Address for Display: 0x10010000                    #
################################################################################

.eqv        INPUT_KEY 0xFFFF0004                    # ASCII code to show, 1 byte
.eqv        CHECK_KEY 0xFFFF0000                    # =1 if has a new keycode ?
                                                    # Auto clear after lw
.eqv        COLOR 0x00FFFF66                        # cobalt blue
.eqv        BLACK 0x00000000                        # black
.eqv        ENV 0x1001
.eqv FAST           10
.eqv NORMAL     100
.eqv SLOW           190
.eqv KEY_A          97
.eqv KEY_S          115
.eqv KEY_D          100
.eqv KEY_W          119
.eqv KEY_Z          122
.eqv KEY_X          120


.text
            li          $k0, INPUT_KEY              # Read_Input_Key key
            li          $k1, CHECK_KEY              # Check if any key has been entered
            addi        $s7, $zero, 512             # store the width in s7
            add         $t7, $t7, $zero

#-----------------------------------------------------------
```

```
#------------Circle detail-------------------------------
circle:
        addi    $a0, $zero, 256                 # x0 = 256
        addi    $a1, $zero, 256                 # y0 = 256
        addi    $a2, $zero, 20                  # r0 = 20 ban kinh cua hinh tron
        addi    $s0, $zero, COLOR
        jal     DrawCircle

max_right_or_down:
        sub     $s6, $s7, $a2

#-----------------------------------------------------------
#------------Controller on key----------------------------
Control:
        beq     $t0, KEY_A, left        # on-click A
        beq     $t0, KEY_D, right       # on-click D
        beq     $t0, KEY_S, down        # on-click S
        beq     $t0, KEY_W, up                  # on-click W
        beq     $t0, KEY_Z, fast        # on-click Z
        j       Read_Input_Key



        left:
                addi    $s0, $zero, BLACK
                jal     DrawCircle

                addi    $a0, $a0, -1
                add     $a1, $a1, $zero
                addi    $s0, $zero, COLOR
                jal     DrawCircle

                blt     $a0, $a2, reboundRight
                j       Read_Input_Key

        right:
                addi    $s0, $zero, BLACK
                jal     DrawCircle

                addi    $a0, $a0, 1
                add     $a1, $a1, $zero
                addi    $s0, $zero, COLOR
                jal     DrawCircle
```

```
        bgt     $a0,$s6,reboundLeft
        j       Read_Input_Key

up:
        addi    $s0, $zero, BLACK
        jal     DrawCircle

        addi    $a1, $a1, -1
        add     $a0, $a0, $zero
        addi    $s0, $zero, COLOR
        jal     DrawCircle

        blt     $a1, $a2, reboundDown
        j       Read_Input_Key

down:
        addi    $s0, $zero, BLACK
        jal     DrawCircle

        addi    $a1, $a1, 1
        add     $a0, $a0, $zero
        addi    $s0, $zero, COLOR
        jal     DrawCircle

        bgt     $a1, $s6, reboundUp
        j       Read_Input_Key

fast:
        addi    $a0, $a0, 5
        li      $v0, 30
        syscall
        j Read_Input_Key

reboundLeft:
        li      $t3, 97
        sw      $t3, 0($k0)
        j       Read_Input_Key

reboundRight:
        li      $t3, 100
        sw      $t3, 0($k0)
        j       Read_Input_Key
```

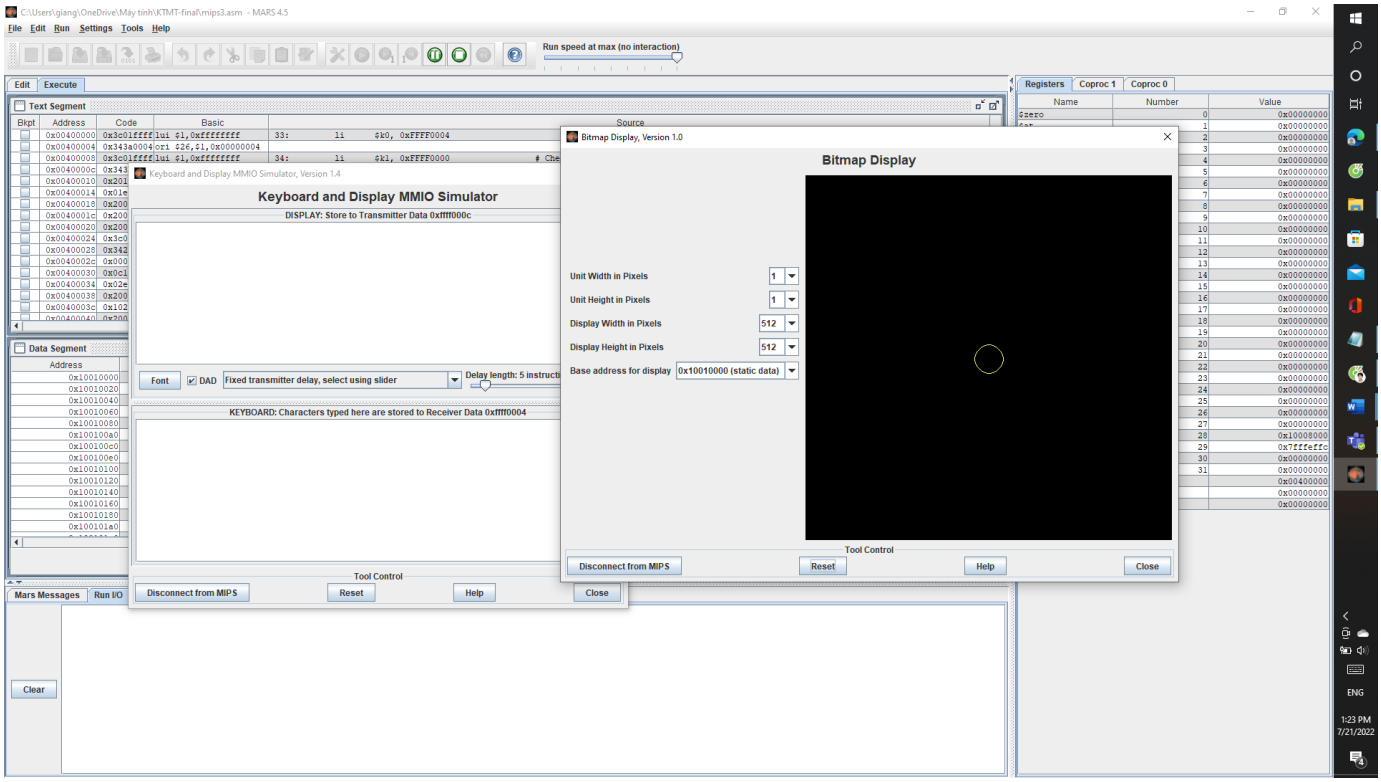
```
        reboundDown:
                li      $t3, 115
                sw      $t3,0($k0)
                j       Read_Input_Key

        reboundUp:
                li      $t3, 119
                sw      $t3,0($k0)
                j       Read_Input_Key

Done:



#-------------------------------------------------------
#------------Key Input-----------------------------------
Read_Input_Key:
        lw      $t0, 0($k0)                     # $t0 = [$k0] = INPUT_KEY chua ky tu nhap vao
        j       Control

#-------------------------------------------------------
#------------Ve duong tron lu bat dau-----------------------
DrawCircle:
        addi    $sp, $sp, -32                   # Khoi tao bien sp bat dau ve vong tron dung thuat toan Breseham
        sw      $ra, 28($sp)
        sw      $a0, 24($sp)
        sw      $a1, 20($sp)
        sw      $a2, 16($sp)
        sw      $s4, 12($sp)
        sw      $s3, 8($sp)
        sw      $s2, 4($sp)
        sw      $s0, 0($sp)



#-------------------------------------------------------
#------------Thuat toan ve duong tron----------------------
Bresenham_Alogorithm:                           # thuat toan ve duong tron
        add     $t0, $zero, $a0                 # x0
        add     $t1, $zero, $a1                 # y0
        add     $t2, $zero, $a2                 # r
        add     $s2, $zero, $zero       # x = 0
        add     $s3, $zero, $a2                 # y = r
        mul     $s4, $a2, -2                    # s4 = -2r
        addi    $s4, $s4, 3                     # p = 3 - 2r
```

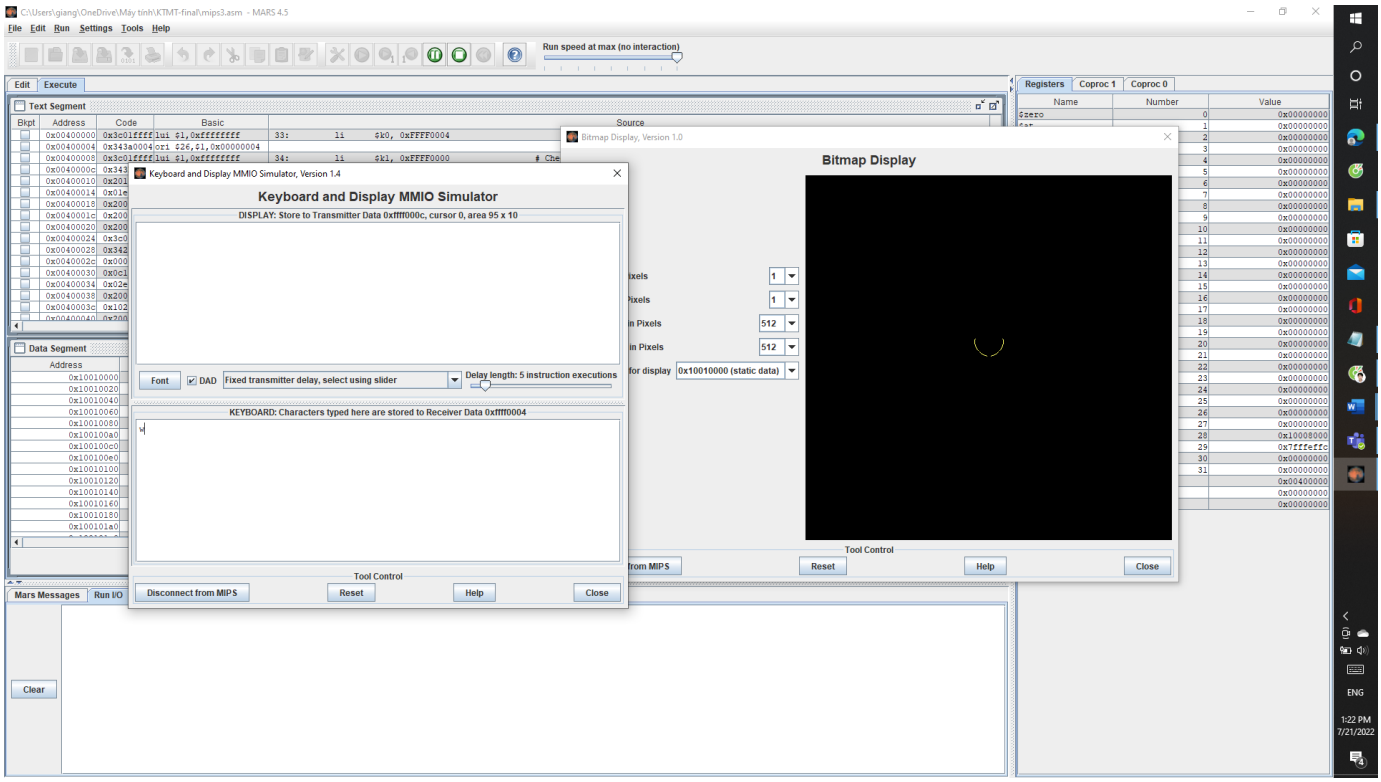
```
#-------------------------------------------------------
#------------Draw 8 point--------------------------------
DrawCircleCondition:
	bgt	$s2, $s3, exitDrawCircle		#if x > y, break the loop (while loop x <= y)

	# ve 8 diem voi cac toa do tuong ung

	# C1(x0+x,y0+y)
	add	$a0, $t0, $s2
	add	$a1, $t1, $s3
	jal	PutPixel			# xac dinh toa do va ve tren bitmap


	# C2(x0-x,y0+y)
	sub	$a0, $t0, $s2
	add	$a1, $t1, $s3
	jal	PutPixel


	# C3(x0+x,y0-y)
	add	$a0, $t0, $s2
	sub	$a1, $t1, $s3
	jal	PutPixel


	# C4(x0-x,y0-y)
	sub	$a0, $t0, $s2
	sub	$a1, $t1, $s3
	jal	PutPixel


	# C5(x0+y,y0+x)
	add	$a0, $t0, $s3
	add	$a1, $t1, $s2
	jal	PutPixel


	# C6(x0-y,y0+x)
	sub	$a0, $t0, $s3
	add	$a1, $t1, $s2
	jal	PutPixel
```

```
        # C7(x0+y,y0-x)
        add     $a0, $t0, $s3
        sub     $a1, $t1, $s2
        jal     PutPixel


        # C8(x0-y,y0-x)
        sub     $a0, $t0, $s3
        sub     $a1, $t1, $s2
        jal     PutPixel

        addi    $s2, $s2, 1

        # if p<0 Vong while check dieu kien
        bgez    $s4, Else
        sll     $t5, $s2, 2                 # 4x
        addi    $t5, $t5, 6                 # 4x + 6
        add     $s4, $s4, $t5               # p = p + 4x +6
        j       Cont

Else:   # re nhanh
        sub     $t3, $s2, $s3
        sll     $t5, $t3, 2                 # 4(x-y)
        addi    $t5, $t5, 10                # 4(x-y) + 1
        add     $s4, $t5, $s4               # p = p + 4(x-y) + 10
        addi    $s3, $s3, -1

Cont:
        j       DrawCircleCondition

exitDrawCircle:
        lw      $s0, 0($sp)                 # Xoa bo nho
        lw      $s2, 4($sp)
        lw      $s3, 8($sp)
        lw      $s4, 12($sp)
        lw      $a2, 16($sp)
        lw      $a1, 20($sp)
        lw      $a0, 24($sp)
        lw      $ra, 28($sp)
        addi    $sp, $sp, 32
        jr      $ra
```

```
#-------------------------------------------------------
#------------Draw on Bitmap Display----------------------
PutPixel:
        addiu   $sp, $sp, -20
        sw      $ra, 16($sp)
        sw      $s1, 12($sp)
        sw      $s0, 8($sp)
        sw      $a0, 4($sp)
        sw      $a1, 0($sp)

        lui     $s1, ENV                # starting address of the screen
        sll     $a0, $a0, 2                     # myltiply by the size of the pixels (4)  lay toa do tung va hoanh lay dia chi
        sll     $a1, $a1, 2                     # myltiply by the size of the pixels (4)
        add     $s1, $s1, $a0                   # x co-ord addded to pixel position
        mul     $a1, $a1, $s7                   # multiply by width
        add     $s1, $s1, $a1                   # add y co-ord to pixel position
        sw      $s0, 0($s1)                     # stores the value of colour into the pixels memory address 32x + 4y xuong 512 o thi dich den pixel do

        lw      $a1, 0($sp)
        lw      $a0, 4($sp)
        lw      $s0, 8($sp)
        lw      $s1, 12($sp)
        lw      $ra, 16($sp)
        addiu   $sp, $sp, 20
        jr      $ra
```

IV. Kết quả chạy chương trình




File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Data Segment
Registers Coproc 1 Coproc 0
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, cursor 0, area 95 x 10
Font
DAD
Fixed transmitter delay, select using slider
Delay length: 5 instruction executions
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
Tool Control
Disconnect from MIPS
Reset
Help
Close
Bitmap Display, Version 1.0
Bitmap Display
Tool Control
Reset
Help
Close
Mars Messages
Run I/O
Clear
ENG
1:22 PM
7/21/2022

BÀI 4:
Postscript CNC Marsbot

Sinh viên thực hiện: Uông Hồng Minh – 20194625

I. Đề bài

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 giả định rằng:

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

- Nếu lưỡi cắt dịch chuyển và cắt tấm kim loại, tức là Marsbot di chuyển và có để lại vết. Để điều 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>,<Cắt/Không cắt>,<Thời gian>

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

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

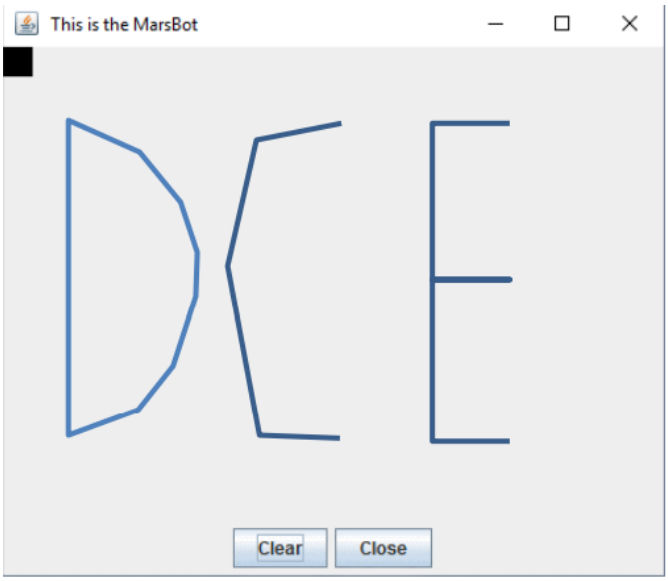
- <Thời gian> là thời gian duy trì quá trình vận hành hiện tại

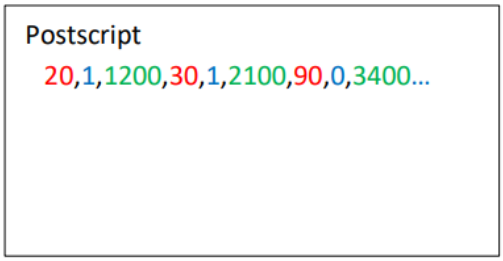
- Hãy lập trình để CNC Marsbot có thể:

- 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)





0	1	2	3
4	5	6	7
8	9	a	b
c	d	e	f

II. Phân tích cách làm

- B1: Kiểm tra người dùng nhập phím nào trên Key matrix
- B2: Đọc từng bộ ba số (góc, cắt/không cắt, thời gian) trong postscript
- B3: Marsbot chuyển động theo bộ 3 số vừa đọc. Lặp lại bước 2 và 3 trong postscript đến khi hết postscript đã chọn

III. Mã nguồn

```
.eqv HEADING 0xffff8010 # Integer: An angle between 0 and 359
# 0 : North (up)
# 90: East (right)
# 180: South (down)
# 270: West (left)
.eqv MOVING 0xffff8050 # Boolean: whether or not to move
.eqv LEAVETRACK 0xffff8020 # Boolean (0 or non-0):
# whether or not to leave a track
.eqv WHEREX 0xffff8030 # Integer: Current x-location ofMarsBot
.eqv WHEREY 0xffff8040 # Integer: Current y-location of
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.data
Postscript1: 180,0,2000, 90,0,1000, 180,1,10000, 75,1,2500, 50,1,2500, 25,1,2500, 0,1,2000 325,1,2500, 300,1,2500
275,1,1900, 90,0,12000, 250,1,2800, 200,1,4000, 160,1,4000, 110,1, 2800 ,90,0,6000, 270,1,2500, 0,1,10000, 90,1,2500,
180,0,5000, 270,1,2500
```

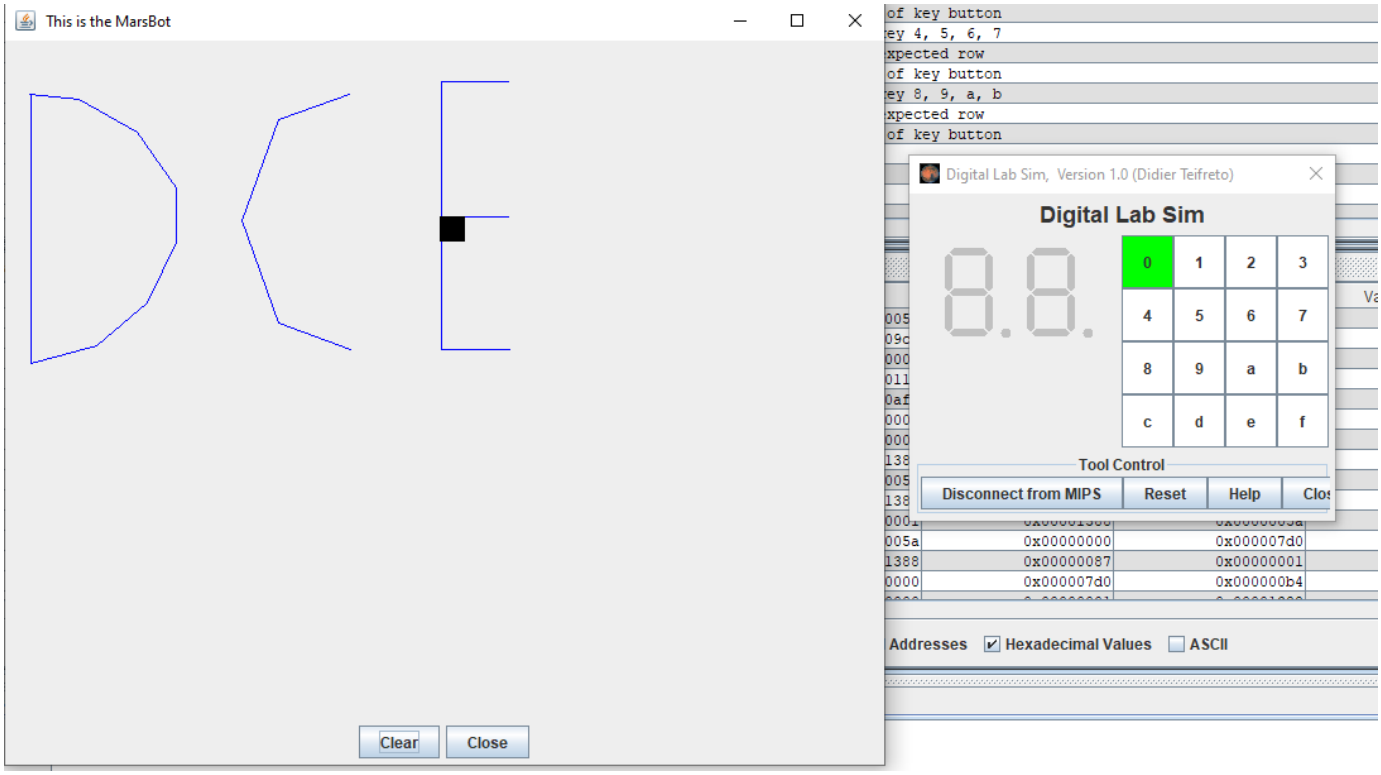
```
end_post1:  -1
Postscript2:        180,0,2000, 90,0,1000, 180,1,5000, 0,0,5000, 135,1,2500, 45,1,2500, 180,1,5000, 90,0,2000, 0,1,5000, 90,0,2000, 180,1,5000, 0,0,5000, 135,1,7000, 0,1,5000, 90,0,2000, 180,1,5000, 90,0,3000, 0,1,5000, 180,0,2500, 270,1,3000
end_post2:  -1
Postscript3:        180,0,2000, 90,0,1000, 180,1,5000, 90,1,2000, 45,1,1000, 0,0,1000, 90,0,1000, 270,1,2000, 90,0,1000, 180,1,2000, 0,0,5500, 270,1,2500, 90,0,5000, 180,1,5000, 90,0,1000, 15,1,6000, 165,1,6000, 345,0,3000, 270,1,1500, 0,0,2500, 90,0,3000, 180,1,5000, 0,0,5000, 150,1,5500, 0,1,5000, 90,0,1000, 180,1,5000, 90,1,2000, 45,1,1000, 0,0,1000, 90,0,1000, 270,1,2000, 90,0,1000, 180,1,2000, 0,0,5500, 270,1,2500
end_post3:  -1
message: .asciiz "Hay lua chon hinh dang muon cut bang cach bam phim 0, 4, 8\n"

.text
main:
check_post:         li $t1, IN_ADRESS_HEXA_KEYBOARD
                    li $t2, OUT_ADRESS_HEXA_KEYBOARD
polling:    li $t3, 0x1 # 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
                    li $t3, 0x2 # check row 2 with key 4, 5, 6, 7
                    sb $t3, 0($t1) # must reassign expected row
                    lb $a1, 0($t2) # read scan code of key button
                    li $t3, 0x4 # check row 3 with key 8, 9, a, b
                    sb $t3, 0($t1) # must reassign expected row
                    lb $a2, 0($t2) # read scan code of key button
                    beq $a0, 0x11, post1
                    beq $a1, 0x12, post2
                    beq $a2, 0x14, post3
                    j no_choice
post1:              la $t1, Postscript1 #luu phuong thuc cat DCE trong t1
                    j CNC
post2:              la $t1, Postscript2 #luu phuong thuc cat MINH trong t1
                    j CNC
post3:              la $t1, Postscript3 #luu phuong thuc cat GIANG trong t1
                    j CNC
no_choice:          li $v0, 4
                    la $a0, message
                    syscall
sleep:              li $a0, 100 # sleep 100ms
                    li $v0, 32
                    syscall
back_to_polling: j polling # continue polling
CNC:                jal GO                  #start mars bot
```

```
                    nop
CUT:                lw $a0, ($t1)               #load heading
                    slt $t2, $a0, $zero   # neu a0<0 thi ket thuc
                    bne $t2, $zero, end_main
                    add $t1, $t1, 4             #doc thong so tiep theo cuar postscript
                    jal ROTATE
                    nop
                    lw $a0, ($t1)               #load track or not
                    add $t1, $t1, 4
                    beq $a0, $zero, UNTRACKING          #neu khong cat thi nhay den untracking
                    jal TRACK
                    nop
                    lw $a0, ($t1)               #load time sleep
                    add $t1, $t1, 4             #doc thong so tiep theo cuar postscript
                    addi $v0,$zero,32   # Keep running by sleeping time in a0
                    syscall
                    jal UNTRACK
                    nop
                    j CUT                       #tiep tuc qua trinh chay may CNC
                    nop
UNTRACKING:         lw $a0, ($t1)               #load time sleep
                    add $t1, $t1, 4             #doc thong so tiep theo cuar postscript
                    addi $v0,$zero,32   # Keep running by sleeping time in a0
                    syscall
                    j CUT                       #tiep tuc qua trinh chay may CNC
                    nop
end_main:           jal STOP            #dung mars bot
                    nop
                    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
nop
jr $ra
nop
#---------------------------------------------------------------
# STOP procedure, to stop running
```

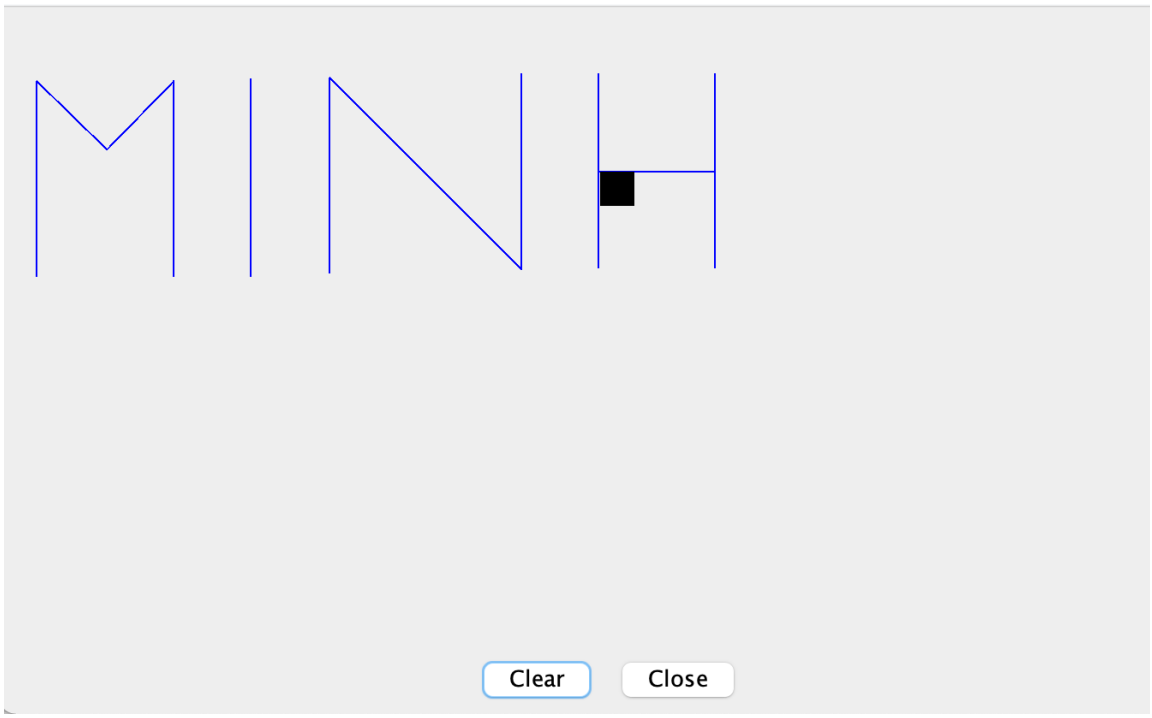
```
# param[in] none
#-------------------------------------------------------------Ha Noi University of Science and Technology

STOP: li $at, MOVING # change MOVING port to 0
sb $zero, 0($at) # to stop
nop
jr $ra
nop
#-------------------------------------------------------------
# TRACK procedure, to start 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
nop
jr $ra
nop
#-------------------------------------------------------------
# UNTRACK procedure, to stop drawing line
# param[in] none
#-------------------------------------------------------------
UNTRACK:li $at, LEAVETRACK # change LEAVETRACK port to 0
sb $zero, 0($at) # to stop drawing tail
nop
jr $ra
nop
#-------------------------------------------------------------
# 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 $a0, 0($at) # to rotate robot
nop
jr $ra
nop
```

IV. Kết quả thực hiện

Lựa chọn 0



Lựa chọn 4



Lựa chọn 8