TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

****

BÁO CÁO FINAL-PROJECT THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Họ tên	MSSV	Đề tài
Trịnh Huy Bằng	20194483	Đề 2
Nguyễn Tiến Đạt	20194504	Đề 8

GVHD: ThS. Lê Bá Vui

1. Project1: (Đề 2)

1.1. Yêu cầu:

Viết một chương trình sử dụng MIPS để vẽ một quả bóng 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.

- Thiết lập màn hình ở kích thước 512x512. Kích thước 1 pixel 1x1.
- Quả bóng là một đường tròn
- 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 (Z), giảm tốc (X) trong bộ giả lập Keyboard and Display MMIO Simulator). Vị trí bóng ban đầu ở giữa màn hình.

1.2. Ý tưởng bài toán:

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.

1.3. Phân tích cách thực hiện:

- Khởi tạo 3 tập lệnh bó macro gồm: **delay**, **branchIfLessOrEqual** và **setColorAndDrawCirle.** Trong đó tập lệnh **delay** dùng để delay chương trình, ảnh hưởng đến tốc độ di chuyển của hình tròn, với tham số truyền vào là %r - thanh ghi dùng để lưu trữ thời gian delay với đơn vị (ms). Tập lệnh **branchIfLessOrEqual** có tham số truyền vào là %r1 , %r2 và %branch, so sánh nếu %r1 <= %r2 thì nhảy đến nhãn %branch. Tập lệnh **setColorAndDrawCirle** dùng để cài đặt màu cho hình tròn và nhảy về nhãn “drawCircle”.

- Hàm **circleInit** dùng để tạo mảng dữ liệu và lưu trữ tọa độ các điểm trên đường tròn.

- Hàm **readKeyboard** dùng để đọc dữ liệu người dùng nhập vào từ bàn phím. Đầu tiên, kiểm tra xem đã có ký tự nào được nhập vào hay chưa? Nếu chưa nhập thì cho phép người dùng nhập vào từ bàn phím. Nếu đã nhập thì nhảy xuống hàm **positionCheck** rồi lần lượt vào các hàm **checkRightEdge**, **checkLeftEdge**, **checkTopEdge** và **checkBottomEdge** để kiểm tra xem đường tròn đã chạm các mép màn hình hay chưa?

- Nếu các điều kiện trên không thỏa mãn nghĩa là đường tròn chưa chạm mép nào thì nhảy xuống hàm **draw**, đổi màu đường tròn hiện tại sang màu nền, cập nhập vị trí mới và đổi màu đường tròn sang màu vàng rồi thực hiện kiểm tra các kí tự nhập vào là gì để nhảy đến các nhãn tương ứng thực hiện việc điều khiển hướng di chuyển hoặc tốc độ của đường tròn.

- Hàm **drawCircle** dùng để vẽ đường tròn, ta dùng hàm **drawCirclePoint** để vẽ các điểm ảnh tạo nên một đường tròn.

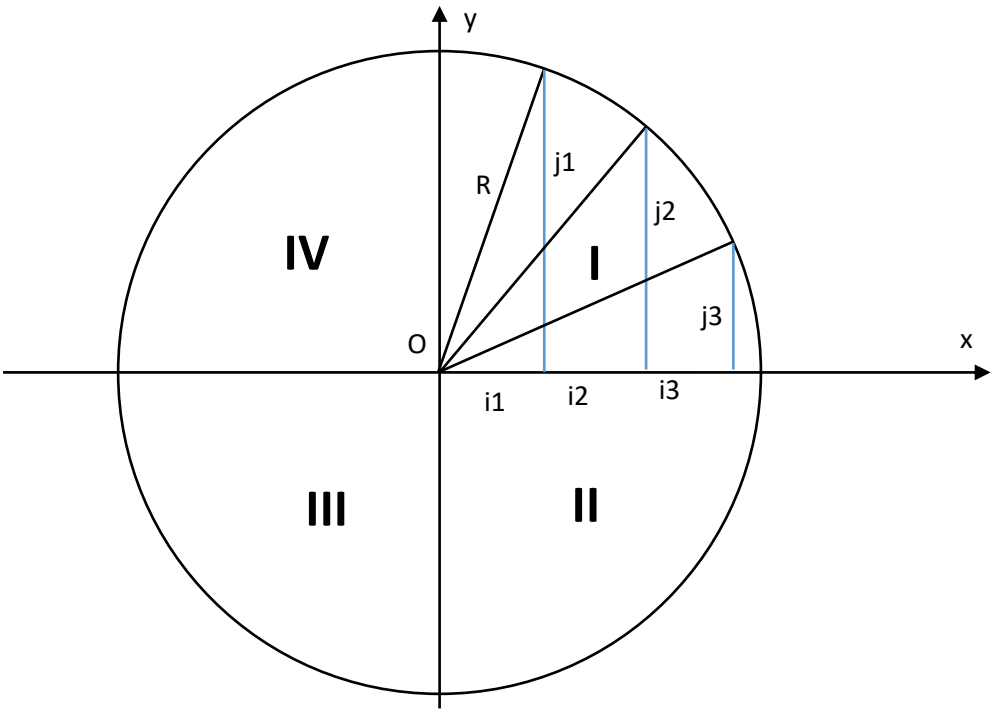
- Hàm **sqrt** dùng để tính căn bậc hai

- Giải thích cách vẽ đường tròn:

+ Đầu tiên thực hiện tính các giá trị j1,j2,….jn tương ứng với i1,i2,…in (i chạy từ 0 -> R) rồi lưu vào $t5. Với j^2 = R^2 – i^2.

+ Trên mỗi ¼ đường tròn vẽ đồng thời 2 điểm ảnh ứng với từng cặp (i , j) :

- Phần tư thứ I: (X0 + i, Y0 + j) và (Xo + j, Yo + i)
- Phần tư thứ II: (X0 + i, Y0 - j) và (Xo + j, Yo - i)
- Phần tư thứ III: (X0 - i, Y0 - j) và (Xo - j, Yo - i)
- Phần tư thứ IV: (X0 - i, Y0 + j) và (Xo - j, Yo + i)



- Các thanh ghi sử dụng:

+ $s0 : lưu trữ tọa độ X của tâm đường tròn

+ $s1 : lưu trữ tọa độ Y của tâm đường tròn

+ $s2 : Lưu trữ bán kính của đường tròn

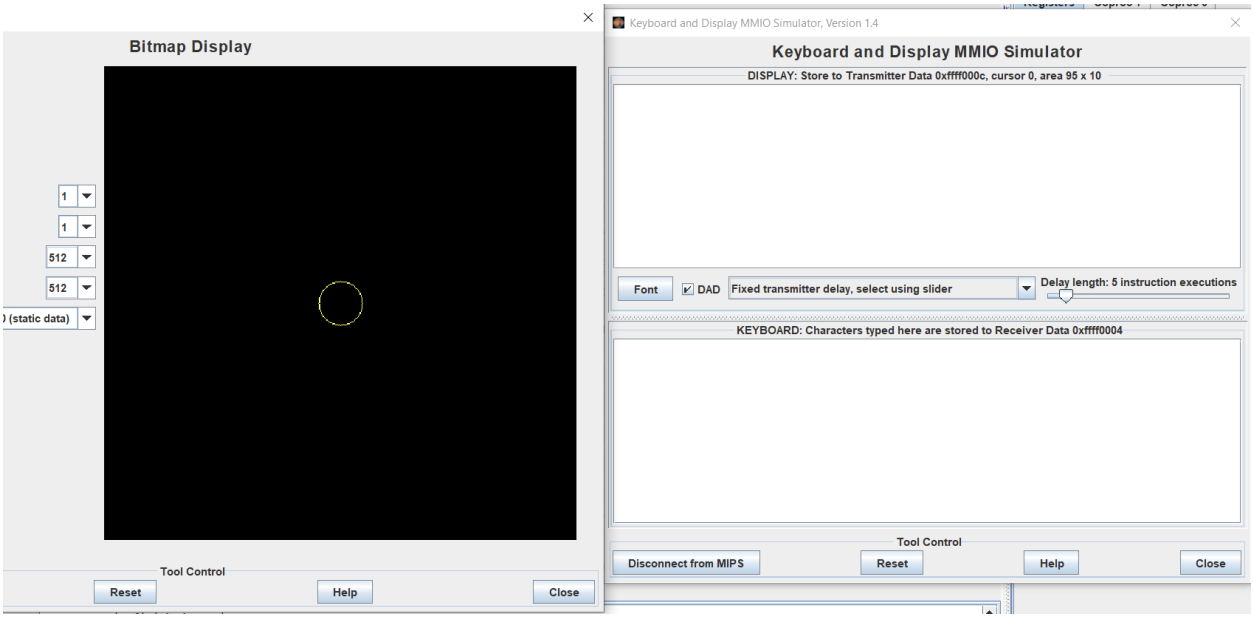
+ $s3 : Lưu trữ độ rộng của màn hình

+ $s4 : Lưu trữ chiều cao của màn hình

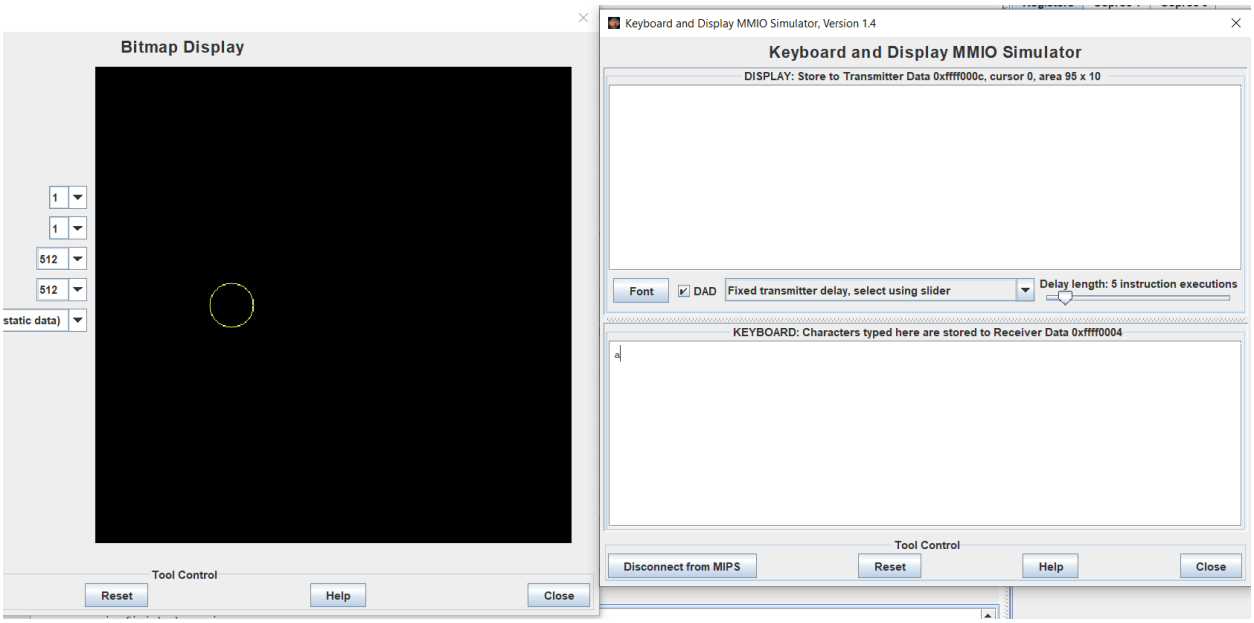
+ $s5 : Lưu trữ màu sắc của màn hình
+ $t6 : Lưu trữ độ lớn khoảng nhảy giữa mỗi lần vẽ đường tròn
+ $s7 : dx
+ $t8 : dy
+ $t9 : Lưu trữ thời gian delay (khởi tạo là 100 (ms))

1.4. Kết quả:

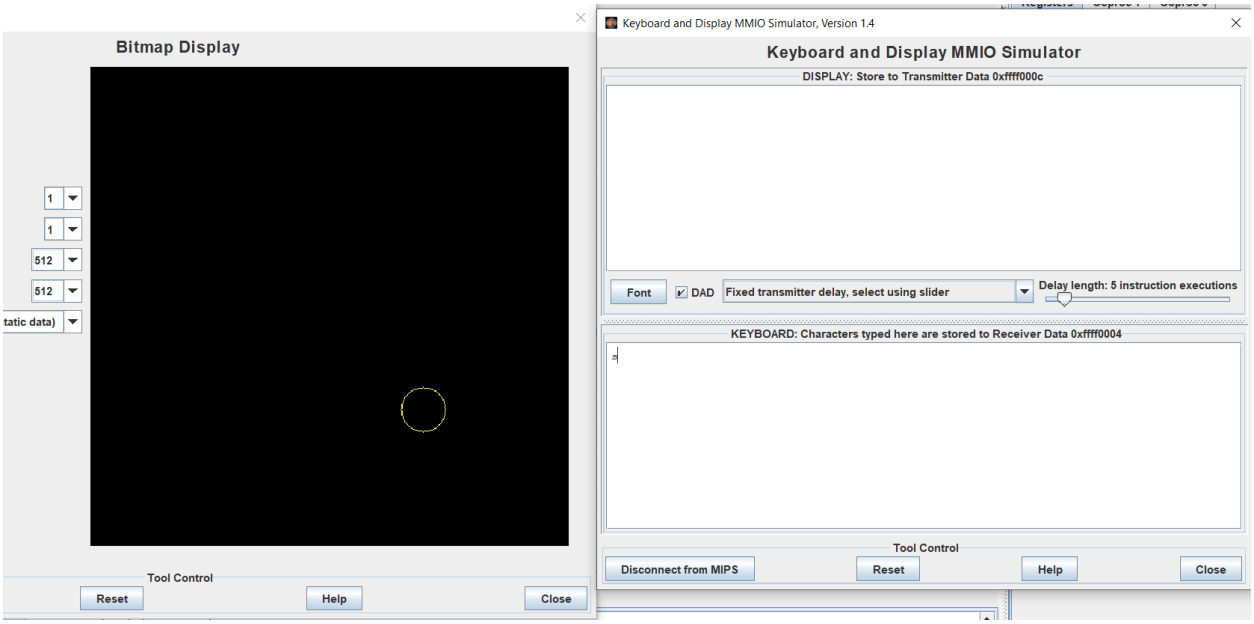
- Chưa nhập gì:



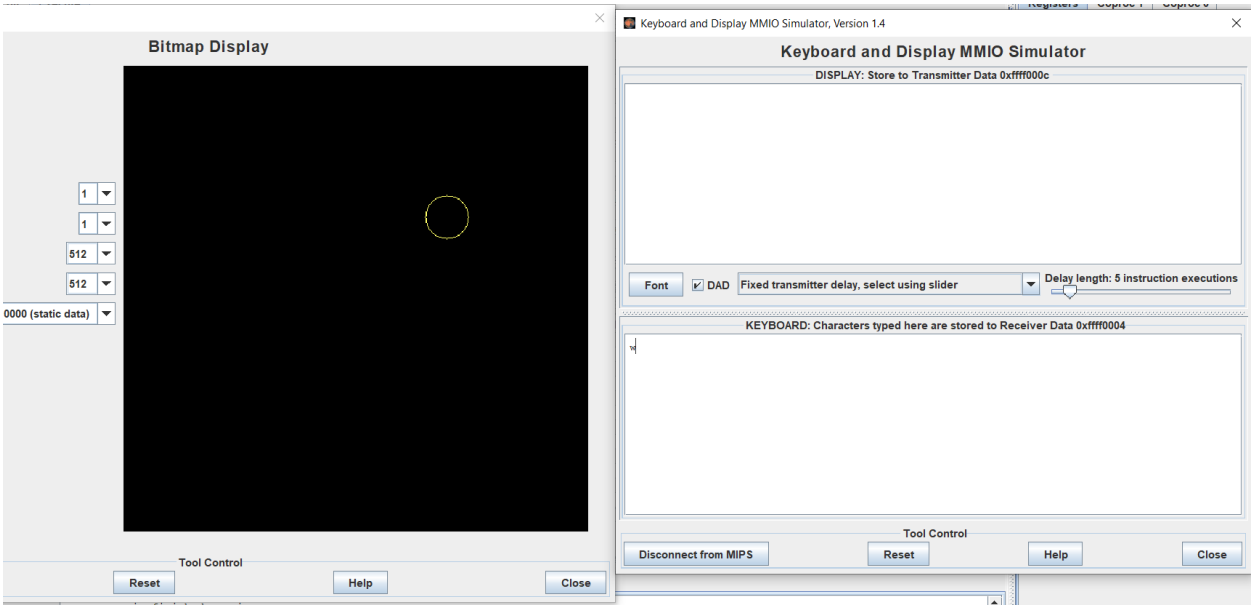
- Nhập vào a:


Bitmap Display
1
1
512
512
static data)
Tool Control
Reset
Help
Close
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

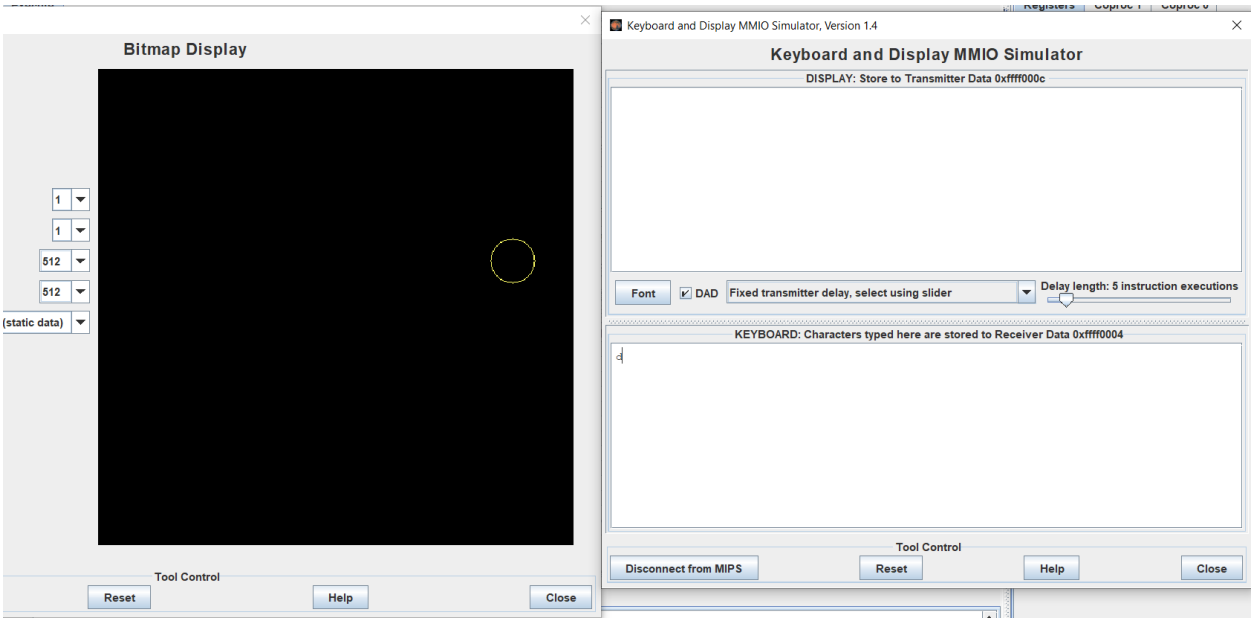
- Nhập vào s:


Bitmap Display
1
1
512
512
tatic data)
Tool Control
Reset
Help
Close
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c
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

- Nhập vào w:



- Nhập vào d:



1.5. Mã nguồn:

```
.eqv SCREEN  0x10010000
.eqv YELLOW  0x00FFFF66
.eqv BACKGROUND 0x00000000
.eqv KEY_A      0x00000061
.eqv KEY_S      0x00000073
.eqv KEY_D      0x00000064
.eqv KEY_W      0x00000077
.eqv KEY_Z      0x0000007A # tang toc (giam thoi gian delay hoac tang do lon khoang nhay)
.eqv KEY_X      0x00000078 # Giam toc ( tang thoi gian delay hoac giam do lon khoang nhay)
.eqv KEY_ENTER      0x0000000A
.eqv DELTA      15
.eqv KEY_CODE       0xFFFF0004
.eqv KEY_READY      0xFFFF0000

#-----------------------------------------------------------------------------------------
# Delay chuong trinh
# Khoang thoi gian delay giua cac lan di chuyen cua hinh tron (ms)

.macro delay(%r)       # %r thanh ghi chua gia tri thoi gian delay
    addi $a0,%r,0
    li $v0, 32
    syscall
.end_macro
```

```
.macro branchIfLessOrEqual(%r1, %r2, %branch) # Tap lenh dung de so sanh neu r1 <= r2 ghi nhay den nhan branch
    sle $v0, %r1, %r2  # $v0 = 1 neu %r1 <= %r2 nguoc lai $v0 = 0
    bnez $v0, %branch        # neu $v0 != 0 nhay den nhan %branch
.end_macro

.macro setColorAndDrawCirle(%color)
    li $s5, %color              #       Dat mau den cho duong tron de
    jal drawCircle              #       xoa duong tron cu.
.end_macro

.kdata
    CIRCLE_DATA:    .space 512
.text
    li $s0, 256    # Xo = 256          Toa do X cua tam duong tron
    li $s1, 256    # Yo = 256          Toa do Y cua tam duong tron
    li $s2, 30     # R = 24            Ban kinh cua duong tron
    li $s3, 512    # SCREEN_WIDTH = 512        Do rong man hinh
    li $s4, 512    # SCREEN_HEIGHT = 512       Chieu cao man hinh
    li $s5, YELLOW #  Mau sac duong tron la mau vang
    li $t6, DELTA        # Khoang nhay giua cac hinh tron
    li $s7, 0       #       dx = 0
    li $t8, 0       #       dy = 0
    li $t9, 100    # Thanh ghi luu tru thoi gian delay ( khoi tao la 100 (ms) )

 #-------------------------------------------------------------------------------------------
 # Ham khoi dong duong tron
```

```
# Tao mang du lieu luu toa do cac diem cua duong tron

circleInit:
    li $t0, 0               # i = 0
    la $t5, CIRCLE_DATA     # tro vao dia chi cua noi luu du lieu duong tron
loop:       slt $v0, $t0, $s2     # for loop i -> R
    beqz $v0, end_circleInit
    mul $s6, $s2, $s2  # R^2
    mul $t3, $t0, $t0     # i^2
    sub $t3, $s6, $t3    # $t3 = R^2 - i^2
    move $v0, $t3
    jal sqrt

    sw $a0, 0($t5)              # Luu j = sqrt(R^2 - i^2) vao mang du lieu
    addi $t0, $t0, 1            # i++
    add $t5, $t5, 4             # Di den vi tri tiep theo luu du lieu cua
CIRCLE_DATA
    j loop
end_circleInit:

#-----------------------------------------------------------------------------------------
# Ham nhap du lieu tu ban phim

start:
readKeyboard:
    lw $k1, KEY_READY # kiem tra da nhap ki tu nao chua
```

```
    beqz $k1, positionCheck  # Neu $k1 != 0 tuc da nhan duoc ki tu nhap tu ban phim thi bat dau kiem tra da va cham canh nao chua
    lw $k0, KEY_CODE         # $k0 luu gia tri ki tu nhap vao, kiem tra voi tung truong hop
    beq $k0, KEY_A, case_a  # Dieu khien qua trai
    beq $k0, KEY_S, case_s# Dieu khien xuong duoi
    beq $k0, KEY_D, case_d # DIeu khien qua phai
    beq $k0, KEY_W, case_w # Dieu khien len tren
    beq $k0, KEY_X, case_x # Giam toc do
    beq $k0, KEY_Z, case_z # Tang toc do
    beq $k0, KEY_ENTER, case_enter # Dung chuong trinh
    j positionCheck
    nop
case_a:
    jal moveToLeft
    j positionCheck
case_s:
    jal moveToDown
    j positionCheck
case_d:
    jal moveToRight
    j positionCheck
case_w:
    jal moveToUp
    j positionCheck
#-----------------------------------------
# Dieu chinh toc do bang khoang nhay DELTA
```

```
#case_z:
#  addi $t6,$t6,5
#  j positionCheck
#case_x:
#  subi $t6,$t6,5
#  j positionCheck
# ----------------------------------------


#----------------------------------------
# Dieu chinh toc do bang thoi gian delay
case_z:
    subi $t9,$t9,50
    j positionCheck
case_x:
    addi $t9,$t9,50
    j positionCheck
 #----------------------------------------


case_enter:
    j endProgram


positionCheck:
checkRightEdge:
    add $v0, $s0, $s2  # Xo + R
    add $v0, $v0,$s7            # If Xo + R + DELTA > SCREEN_WIDTH Then
moveToLeft
```

```
    branchIfLessOrEqual($v0, $s3, checkLeftEdge)    # else check left edge
    jal moveToLeft
    nop
 checkLeftEdge:
    sub $v0, $s0, $s2
    add $v0, $v0, $s7  # If Xo - R + DELTA < 0 then moveToRight
    branchIfLessOrEqual($zero, $v0, checkTopEdge)  # else check top edge
    jal moveToRight
    nop
 checkTopEdge:
    sub $v0, $s1, $s2
    add $v0, $v0, $t8  # If Yo - R + DELTA < 0 then moveToDown
    branchIfLessOrEqual($zero, $v0, checkBottomEdge) # else check bottom
edge
    jal moveToDown
    nop
 checkBottomEdge:
    add $v0, $s1, $s2
    add $v0, $v0, $t8  # If Yo + R + DELTA > SCREEN_HEIGHT then
moveToUp
    branchIfLessOrEqual($v0, $s4, draw)         # else all condition eligible,
draw circle
    jal moveToUp
    nop


#-------------------------------------------------------------------------------------
# Ham ve duong tron
```

```
draw:
    setColorAndDrawCirle(BACKGROUND) # Ve duong tron trung mau nen
    add $s0, $s0, $s7           # Cap nhat toa do moi cua duong tron
    add $s1, $s1, $t8

    setColorAndDrawCirle(YELLOW)       # Ve duong tron moi
    delay($t9)                         # Dung chuong trinh 1 khoang
    j start

endProgram:
    li $v0, 10
    syscall

#---------------------------------------------------------------------------------------------
# Ham ve duong tron
# Su dung du lieu o mang CIRCLE_DATA tao boi Circle_Init

 drawCircle:
    add $sp, $sp, -4
    sw $ra, 0($sp)
    li $t0, 0                # khoi tao bien i = 0
 loop_drawCircle:
    slt $v0, $t0, $s2     # i -> R
    beqz $v0,  end_drawCircle # Neu i = R thi dung

    sll $t5, $t0, 2
    lw $t3, CIRCLE_DATA($t5) # Load j to $t3
```

```
    move $a0, $t0               # i = $a0
    move $a1, $t3               # j = $a1
    jal drawCirclePoint# Ve 2 diem (Xo + i, Yo + j), (Xo + j, Yo + i) tren phan tu thu I
    sub $a1, $zero, $t3
    jal drawCirclePoint# Ve 2 diem (Xo + i, Yo - j), (Xo + j, Yo - i) tren phan tu thu II
    sub $a0, $zero, $t0
    jal drawCirclePoint# Ve 2 diem (Xo - i, Yo - j), (Xo - j, Yo - i) tren phan tu thu III
    add $a1, $zero, $t3
    jal drawCirclePoint# Ve 2 diem (Xo - i, Yo + j), (Xo - j, Yo + i) tren phan tu thu IV


    addi $t0, $t0, 1
    j loop_drawCircle
  end_drawCircle:
    lw $ra, 0($sp)
    add $sp, $sp, 0
    jr $ra


#-------------------------------------------------------------------------------------
#   Ham ve diem tren duong tron
#   Ve dong thoi 2 diem (X0 + i, Y0 +j ) va (Xo + j, Yo + i)
#   i = $a0, j = $a1
#   Xi =$t1, Yi = $t4
```

```
 drawCirclePoint:


    add $t1, $s0, $a0  # Xi = X0 + i
    add $t4, $s1, $a1  # Yi = Y0 + j
    mul $t2, $t4, $s3   # Yi * SCREEN_WIDTH
    add $t1, $t1, $t2   # Yi * SCREEN_WIDTH + Xi (Toa do 1 chieu cua diem anh)
    sll $t1, $t1, 2                # Dia chi tuong doi cua diem anh
    sw $s5, SCREEN($t1)    # Draw anh
    add $t1, $s0, $a1  # Xi = Xo + j
    add $t4, $s1, $a0  # Yi = Y0 + i
    mul $t2, $t4, $s3   # Yi * SCREEN_WIDTH
    add $t1, $t1, $t2   # Yi * SCREEN_WIDTH + Xi (Toa do 1 chieu cua diem anh)
    sll $t1, $t1, 2                # Dia chi tuong doi cua diem anh
    sw $s5, SCREEN($t1)    # Draw anh


    jr $ra


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

# Cac ham di chuyen


moveToLeft:
    sub $s7, $zero, $t6
    li $t8, 0
    jr $ra
moveToRight:
```

```
    add $s7, $zero, $t6
    li $t8, 0
    jr $ra
moveToUp:
    li $s7, 0
    sub $t8, $zero, $t6
    jr $ra
moveToDown:
    li $s7, 0
    add $t8, $zero, $t6
    jr $ra

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

# Square Root
# de su dung floating point thi phai chuyen sang coprocessor

# $v0 = S, $a0 = sqrt(S)

sqrt:
    mtc1 $v0, $f0 # dua tu $v0 vao $f0
    cvt.s.w $f0, $f0 # Chuyen ve int 32 bit
    sqrt.s $f0, $f0 # Tinh can bac hai cua %f0
    cvt.w.s $f0, $f0 # Chuyen lai ve word
    mfc1 $a0, $f0 # dua lai tu $f0 vao $a0
    jr $ra
```

2.Project2: (Đề 8)

2.1. Yêu cầu:

Viết chương trình mô phỏng hoạt động của RAID 5 với 3 ổ đĩa:

- Giả định mỗi block dữ liệu có 4 kí tự
- Giới hạn chuỗi kí tựnhập vào có độ dài là bội của 8.
- Block 4 byte đầu tiên sẽ được lưu trên Disk 1
- Block 4 byte tiếp theo sẽ lưu trên Disk 2
- Dữ liệu trên Disk 3 sẽ là 4 byte parity được tính từ 2 block đầu tiên với mã ASCII

Giả sử chuỗi kí tự nhập vào từ bàn phím là (DCE.****ABCD1234HUSTHUST)

Trong Disk3 ta có: 6e=’D’ xor ‘*’ ; 69=’C’ xor ‘*’; 6f=’E’ xor ‘*’; 04=’.’ xor ‘*’

```
Nhap chuoi ki tu : DCE.****ABCD1234HUSTHUST
     Disk 1                Disk 2                Disk 3
 --------------        --------------        --------------
|      DCE.      |      |      ****      |      [[ 6e,69,6f,04]]
|      ABCD      |      [[ 70,70,70,70]]      |      1234      |
[[ 00,00,00,00]]      |      HUST      |      |      HUST      |
 --------------        --------------        --------------
```

2.2. Ý tưởng bài toán:

Xử lí xâu nhập vào từ bàn phím để đọc và ghi các block tương ứng vào các Disk rồi sau đó dựa trên 2 Block đã tìm được để tính toán 4 byte parity và ghi vào Disk còn lại.

2.3. Phân tích cách thực hiện:

- Khi nhập một xâu từ bàn phím ta sẽ thực hiện kiểm tra độ dài của xâu bằng hàm(check_char) . Độ dài xâu sẽ được lưu vào thanh ghi $t3.

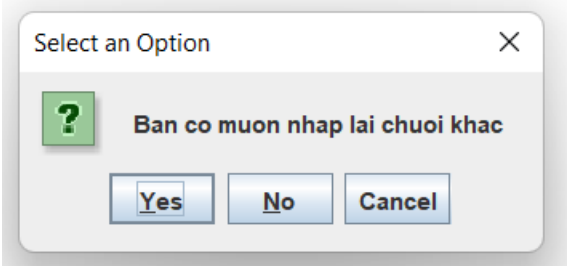
- Hàm (test_length) sẽ thực hiện kiểm tra xem độ dài của xâu có phải là bội của 8 hay không bằng cách: kiểm tra chữ số hexa cuối của giá trị thanh ghi $t3(thanh ghi lưu độ dài xâu) ở dạng hexa. Nếu chữ số đó là 0 hoặc 8 thì độ dài xâu là bội của 8. Nếu không thỏa mãn thì sẽ yêu cầu nhập lại. Nếu thỏa mãn thì sẽ bắt đầu xử lí xâu
- Ta xử lí xâu theo từng block 8 byte. Trong từng block đó ta lại chia thành 2 block 4 byte như yêu cầu bài toán. Mỗi lần xử lí xong một kí tự trong xâu thì giá trị lưu ở $t3 sẽ giảm 1. Khi giá trị giảm về 0 thì việc xử lí xâu kết thúc. Ta quy ước mỗi vòng lặp sẽ xử lí gồm 3 block 8 byte:
  - Block 8 byte thứ 1:
    - Block 4 byte thứ nhất sẽ ghi vào Disk1, Block 4 byte thứ 2 sẽ ghi vào Disk2.
    - Hàm HEX sẽ thực hiện tính mã parity từ 2 block đã đọc được ở trên. Kết quả tính được sẽ ghi vào Disk3.
    - Nếu giá trị $t3 bằng 0 thì sẽ kết thúc đọc xâu. Nếu không thì sẽ chuyển sang xử lí Block 8 byte thứ 2.
  - Block 8 byte thứ 2:
    - Block 4 byte thứ nhất sẽ ghi vào Disk1, Block 4 byte thứ 2 sẽ ghi vào Disk3.
    - Hàm HEX sẽ thực hiện tính mã parity từ 2 block đã đọc được ở trên. Kết quả tính được sẽ ghi vào Disk2.
    - Nếu giá trị $t3 bằng 0 thì sẽ kết thúc đọc xâu. Nếu không thì sẽ chuyển sang xử lí Block 8 byte thứ 3.
  - Block 8 byte thứ 3:
    - Block 4 byte thứ nhất sẽ ghi vào Disk2, Block 4 byte thứ 2 sẽ ghi vào Disk3.
    - Hàm HEX sẽ thực hiện tính mã parity từ 2 block đã đọc được ở trên. Kết quả tính được sẽ ghi vào Disk1.
    - Nếu giá trị $t3 bằng 0 thì sẽ kết thúc đọc xâu. Nếu không thì quay lại vòng lặp xử lí Block 8 byte thứ 1.
- Xử lí xâu hoàn tất khi giá trị $t3 bằng 0. Lúc đó chương trình sẽ hỏi người dùng có muốn nhập xâu mới hay không. Nếu đồng ý sẽ quay lại vòng lặp nhập một xâu mới. Nếu không thì chương trình kết thúc.

2.4. Kết quả:

- Nhập xâu độ dài không phải bội của 8

```
Nhap vao chuoi ky tu : sdse
Do dai chuoi khong hop le! Moi nhap lai!
Nhap vao chuoi ky tu : 2134567tyrfsccfvgds
Do dai chuoi khong hop le! Moi nhap lai!
```

- Cho phép nhập lại xâu mới



- Nhập xâu gồm 2 Block 8 byte

```
Nhap vao chuoi ky tu : 1234abcd5678HGFD
      Disk 1                Disk 2               Disk 3
-----------------     -----------------     -----------------
|      1234      |     |      abcd      |     [[ 50,50,50,50]]
|      5678      |     [[ 7d,71,71,7c]]     |      HGFD      |
-----------------     -----------------     -----------------
```

- Nhập xâu gồm 3 Block 8 byte

```
Nhap vao chuoi ky tu : 1111aaaa2222bbbb3333cccc
      Disk 1                Disk 2               Disk 3
-----------------     -----------------     -----------------
|      1111      |     |      aaaa      |     [[ 50,50,50,50]]
|      2222      |     [[ 00,00,00,00]]     |      bbbb      |
[[ 50,50,50,50]]     |      3333      |     |      cccc      |
-----------------     -----------------     -----------------
```

- Nhập xâu gồm 4 Block 8 byte

```
Nhap vao chuoi ky tu : 12345678qwerasdf1234dfghcder4567
      Disk 1                Disk 2                Disk 3
----------------      ----------------      ----------------
|     1234     |      |     5678     |      [[ 04,04,04,0c]]
|     qwer     |      [[ 00,00,00,00]]      |     asdf     |
[[ 55,54,54,5c]]      |     1234     |      |     dfgh     |
|     cder     |      |     4567     |      [[ 57,51,53,45]]
----------------      ----------------      ----------------
```

- Nhập xâu gồm 5 Block 8 byte

```
Nhap vao chuoi ky tu : 1234dsfd6784dfyb1784fhdb4762sertyutr5419
      Disk 1                Disk 2                Disk 3
----------------      ----------------      ----------------
|     1234     |      |     dsfd     |      [[ 55,41,55,50]]
|     6784     |      [[ 00,00,00,00]]      |     dfyb     |
[[ 57,5f,5c,56]]      |     1784     |      |     fhdb     |
|     4762     |      |     sert     |      [[ 47,52,44,46]]
|     yutr     |      [[ 00,00,00,00]]      |     5419     |
----------------      ----------------      ----------------
```

2.5. Mã nguồn:

```
.data
nhap: .asciiz "Nhap vao chuoi ky tu : "
hex:   .byte '0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f'
d1:    .space 4
d2:    .space 4
d3:    .space 4
```

```
array: .space 32
string:.space 10000
enter: .asciiz "\n"
m1:     .asciiz "      Disk 1                Disk 2              Disk 3\n"
m2:    .asciiz "----------------       ----------------       ----------------\n"
m3:    .asciiz "|     "
m4:    .asciiz "     |       "
m5:    .asciiz "[[ "
m6:    .asciiz "]]       "
comma: .asciiz ","



m7:    .asciiz "Ban co muon nhap lai chuoi khac"
error_length: .asciiz "Do dai chuoi khong hop le! Moi nhap lai!\n"

.text
        la $s1, d1
        la $s2, d2
        la $s3, d3
        la $a2, array               # dia chi mang chua ma parity

input: li $v0, 4              # nhap chuoi ki tu
        la $a0, nhap
        syscall
```

```
        li $v0, 8
        la $a0, string
        li $a1, 10000
        syscall



# Kiem tra do dai co phai la boi cua 8 khong
length: move $s0, $a0          # s0 chua dia chi xau moi nhap

       addi $t3, $zero, 0 # t3 = length
        addi $t0, $zero, 0         # i = 0

check_char:
       add $t1, $s0, $t0  # t1 = address of string[i]
        lb $t2, 0($t1)                # t2 = string[i]
        nop
        beq $t2,10,test_length # t2 = '\n' ket thuc xau
        nop
        addi $t3, $t3, 1    # length++
        addi $t0, $t0, 1    # index++
        j check_char
        nop
test_length:
        and $t1, $t3, 0x0000000f              # giu lai chu so hexa cuoi
```

```
        bne $t1, 0, test1               # chu so hexa cuoi bang 0 hoac 8
thi length chia het cho 8
        j start
test1:  beq $t1, 8, start               # kiem tra co phai bang 8 ?
        j error1
error1:         li $v0, 4
        la $a0, error_length            # do dai xau khong hop le
        syscall
        j input
# Ket thuc kiem tra do dai

# Xu li ma parity
HEX:
        add  $t9,$t8,$0
        andi $t8,$t8,0x0000000f       # lay chu so hexa ben phai
        srl  $t9,$t9,4
        andi $t9,$t9,0x0000000f       # lay chu so hexa ben trai
        la $t5,hex
        move $t6,$t5
        add $t5,$t5,$t9
        add $t6,$t6,$t8
        lb $a0,0($t5)               # print ma parity
        li $v0,11
        syscall
        lb $a0,0($t6)
        li $v0,11
```

```
        syscall
        jr $ra
# Ket thuc xu li ma parity

#------------------------------mo phong RAID 5------------------------------
#----------------------xet 6 khoi dau----------------------
#----------------lan 1: luu vao 2 khoi 1,2; xor vao 3-------
start:
        li $v0, 4
        la $a0, m1
        syscall
        li $v0, 4
        la $a0, m2
        syscall
# Xet nhom gom 2 block 4 byte thu nhat
split1:         addi $t0, $zero, 0
        addi $t9, $zero, 0
        addi $t8, $zero, 0
        la $s1, d1          # disk 1
        la $s2, d2          # disk 2
        la $a2, array       # ma parity
print11:li $v0, 4
        la $a0, m3
        syscall
b11:  lb $t1, ($s0)
```

```
        addi $t3, $t3, -1
        sb $t1, ($s1)       # luu vao disk1
b21:    add $s5, $s0, 4
        lb $t2, ($s5)
        addi $t3, $t3, -1
        sb $t2, ($s2)       # luu vao disk2
b31:    xor $a3, $t1, $t2
        sw $a3, ($a2)       # luu vao array
        addi $a2, $a2, 4
        addi $t0, $t0, 1
        addi $s0, $s0, 1
        addi $s1, $s1, 1
        addi $s2, $s2, 1
        bgt $t0, 3, reset   # doc xong split1
        j b11
reset:  la $s1, d1
        la $s2, d2
print12:lb $a0, ($s1)       # print noi dung disk
        li $v0, 11
        syscall
        addi $t9, $t9, 1
        addi $s1, $s1, 1
        bgt $t9, 3, next11
        j print12
next11:     li $v0, 4
```

```
        la $a0, m4
        syscall
        li $v0, 4
        la $a0, m3
        syscall
print13:lb $a0, ($s2)
        li $v0, 11
        syscall
        addi $t8, $t8, 1
        addi $s2, $s2, 1
        bgt $t8, 3, next12
        j print13
next12:     li $v0, 4
        la $a0, m4
        syscall
        li $v0, 4
        la $a0, m5
        syscall

        la $a2, array
        addi $t7, $zero, 0
print14:lw $t8, 0($a2)
        jal HEX
        addi $t7, $t7, 1
        addi $a2, $a2, 4
```

```
        bgt $t7, 3, end1

       li $v0, 4
        la $a0, comma
        syscall

        j print14
end1:
        li $v0, 4
        la $a0, m6
        syscall
        li $v0, 4
        la $a0, enter
        syscall
        beq $t3, 0, exit1           # kiem tra da doc het xau chua?
# Xet nhom gom 2 block 4 byte thu 2
split2:         la $a2, array
        la $s1, d1
        la $s3, d3
        addi $s0, $s0, 4
        addi $t0, $zero, 0
print21:li $v0, 4
        la $a0, m3
        syscall
b12:  lb $t1, ($s0)
```

```
        addi $t3, $t3, -1
        sb $t1, ($s1)
b32:    add $s5, $s0, 4
        lb $t2, ($s5)
        addi $t3, $t3, -1
        sb $t2, ($s3)
b22:    xor $a3, $t1, $t2
        sw $a3, ($a2)
        addi $a2, $a2, 4
        addi $t0, $t0, 1
        addi $s0, $s0, 1
        addi $s1, $s1, 1
        addi $s3, $s3, 1
        bgt $t0, 3, reset2
        j b12
reset2:     la $s1, d1
        la $s3, d3
        addi $t9, $zero, 0
print22:lb $a0, ($s1)
        li $v0, 11
        syscall
        addi $t9, $t9, 1
        addi $s1, $s1, 1
        bgt $t9, 3, next21
        j print22
```

```
next21:     li $v0, 4
        la $a0, m4
        syscall
        addi $t7, $zero, 0
        li $v0, 4
        la $a0, m5
        syscall
print23:lw $t8, 0($a2)
        jal HEX
        addi $t7, $t7, 1
        addi $a2, $a2, 4
        bgt $t7, 3, next22
        li $v0, 4
        la $a0, comma
        syscall

        j print23
next22:
        li $v0, 4
        la $a0, m6
        syscall
        li $v0, 4
        la $a0, m3
        syscall
        addi $t8, $zero, 0
```

```
print24:lb $a0, ($s3)
        li $v0, 11
        syscall
        addi $t8, $t8, 1
        addi $s3, $s3, 1
        bgt $t8, 3, end2
        j print24

end2:   li $v0, 4
        la $a0, m4
        syscall
        li $v0, 4
        la $a0, enter
        syscall
        beq $t3, 0, exit1
# Xet nhom gom 2 block 4 byte thu 3
split3:         la $a2, array
        la $s2, d2
        la $s3, d3
        addi $s0, $s0, 4
        addi $t0, $zero, 0
print31:li $v0, 4
        la $a0, m5
        syscall
b23:  lb $t1, ($s0)
```

```
        addi $t3, $t3, -1
        sb $t1, ($s2)
b33:    add $s5, $s0, 4
        lb $t2, ($s5)
        addi $t3, $t3, -1
        sb $t2, ($s3)
b13:    xor $a3, $t1, $t2
        sw $a3, ($a2)
        addi $a2, $a2, 4
        addi $t0, $t0, 1
        addi $s0, $s0, 1
        addi $s2, $s2, 1
        addi $s3, $s3, 1
        bgt $t0, 3, reset3
        j b23
reset3:     la $s2, d2
        la $s3, d3
        la $a2, array
        addi $t7, $zero, 0
print32:lw $t8, 0($a2)
        jal HEX
        addi $t7, $t7, 1
        addi $a2, $a2, 4
        bgt $t7, 3, next31
        li $v0, 4
```

```
        la $a0, comma
        syscall

        j print32
next31:
        li $v0, 4
        la $a0, m6
        syscall
        li $v0, 4
        la $a0, m3
        syscall
        addi $t9, $zero, 0
print33:lb $a0, 0($s2)
        li $v0, 11
        syscall
        addi $t9, $t9, 1
        addi $s2, $s2, 1
        bgt $t9, 3, next32
        j print33
next32:     li $v0, 4
        la $a0, m4
        syscall
        li $v0, 4
        la $a0, m3
        syscall
```

```
        addi $t9, $zero, 0
print34:lb $a0, ($s3)
        li $v0, 11
        syscall
        addi $t9, $t9, 1
        addi $s3, $s3, 1
        bgt $t9, 3, end3
        j print34

end3:   li $v0, 4
        la $a0, m4
        syscall
        li $v0, 4
        la $a0, enter
        syscall
        beq $t3, 0, exit1

       addi $s0, $s0, 4
        j split1

exit1:  li $v0, 4
        la $a0, m2
        syscall
        j ask
# ket thuc mo phong RAID 5
```

```
# nhap xau moi
ask:  li $v0, 50
      la $a0, m7
      syscall
      beq $a0, 0, input
      nop
      j exit
      nop

exit: li $v0, 10
      syscall
```