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

BÁO CÁO CUỐI KỲ
MÔN THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Giáo viên hướng dẫn: Lê Bá Vui
Sinh viên thực hiện: Trần Phạm Minh Đức (20226077)
Nguyễn Trung Hiếu (20226082)

Nhóm: 7
Lớp: Thực hành Kiến trúc máy tính
Mã lớp: 147789
Kỳ học: 2023.2

Table of Contents

Phần I: Vẽ hình trên màn hình Bitmap

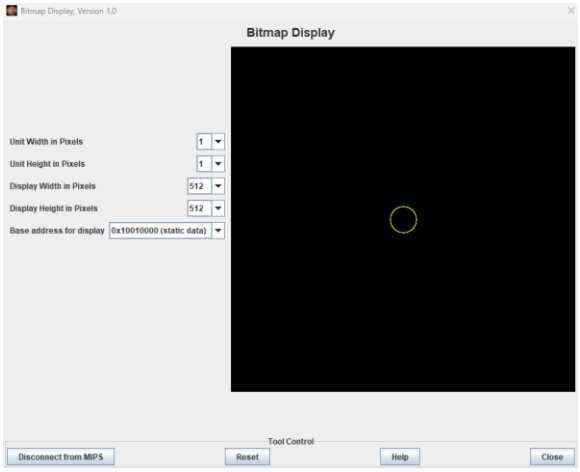
Viết chương trình vẽ một quả bóng hình tròn di chuyển trên màn hình mô phỏng Bitmap của Mars. Nếu đối tượng đập vào cạnh của màn hình thì sẽ di chuyển theo chiều ngược lại.

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.



Code:

Phần định nghĩa hằng số và dữ liệu

```
.eqv KEY_CODE 0xFFFF0004  # ASCII code to show, 1 byte

.data

LEFT:   .asciiz "a"

RIGHT : .asciiz "d"

UP:     .asciiz "w"

DOWN:         .asciiz "s"


.text
```

```
li $k0, KEY_CODE                    #Enter the key to direct ball

.eqv YELLOW 0x00FFFF00
.eqv MONITOR_SCREEN 0x10010000
Phần mã lệnh
.text
#Draw the circle at the center of screen
# Center point is (x0, y0)
#s0 = x0
#s1 = y0
#s2 = color
#s3 = radius

li $v1, MONITOR_SCREEN
#Set the first value
        li $s0, 256         # set x_center point at center of screen
        li $s1, 256         # set y_center point at center of screen
        li $s3, 20          # value of radius
        li $s2, YELLOW
        li $s4, 1
        addi $s7, $0, 512   #save the large to $s7
        jal DrawCircle
        nop
```

Giải thích:

- Phần này để vẽ hình tròn ở tâm chính giữa của Bitmap.
- Khởi tạo giá trị cho các thanh ghi $s1,$s2,$s3,$s4,$s7.
- nhảy đến hàm DrawCircle để vẽ hình tròn .
- nop nhằm không thực hiện gì, đảm bảo lệnh “jal” hoàn thành.

Phần di chuyển hình tròn:

```
#-----------------------------------------------------------------------------------------------------------
-------------
MMove:                          #Moving the circle
  Readkey:
        lw $t0, 0($k0)
        beq $t0, 97, left       #$t0 = 'a'
        beq $t0, 100, right     #$t0 = 'd'
        beq $t0, 115, down      #$t0 = 's'
        beq $t0, 119, up        #$t0 = 'w'
        beq $t0, 120, slow
        beq $t0, 122, speed
        j Readkey
  EndReadkey:
                #-------------------------------------------------------------------------#
  left:
        addi $t0, $0, 97
        sw $t0, 0($k0)
        bltu  $s0, 20, right            # If go the the left margin, back right
        li $s2, 0x00000000
        jal DrawCircle
        li $s2, YELLOW
        sub $s0, $s0, $s4
        jal DrawCircle
        jal Readkey
 end_left:
                        #-----------------------------------------#
```

```
 right:
        addi $t0, $0, 100
        sw $t0, 0($k0)
        bgtu  $s0, 492, left
        li $s2, 0x00000000
        jal DrawCircle
        li $s2, YELLOW
        add $s0, $s0, $s4
        jal DrawCircle
        jal Readkey
end_right:
                        #-----------------------------------------#
up:
        addi $t0, $0, 119
        sw $t0, 0($k0)
        bltu  $s1, 20, down
        li $s2, 0x00000000
        jal DrawCircle
        li $s2, YELLOW
        sub $s1, $s1, $s4
        jal DrawCircle
        jal Readkey
end_up:
                        #-----------------------------------------#
down:
        addi $t0, $0, 115
        sw $t0, 0($k0)
```

```
        bgtu $s1, 492, up
        li $s2, 0x00000000
        jal DrawCircle
        li $s2, YELLOW
        addu $s1, $s1, $s4
        jal DrawCircle
        jal Readkey
end_down:
                        #----------------------------------------#
speed:
        sw $0, 0($k0)
        sll $s4, $s4, 1
        bgt $s4, 8, update_speed
        j Readkey
end_speed:

update_speed:
        addi $s4, $0, 8
        j Readkey
end_update_speed:

slow:
        sw $0, 0($k0)
        srl $s4, $s4, 1
        blt $s4, 1, update_slow
        j Readkey
end_slow:
```

```
update_slow:
        addi $s4, $0, 1
        j Readkey
end_update_slow:
end_MMove:
```

Giải thích:

- Phần này để đọc phím bấm từ địa chỉ “KEY_CODE” vào thanh ghi $t0
- Với mỗi $t0 khác nhau (giá trị tùy theo mã acsii của các ký tự a,w,s,d,z,x) thì hình tròn sẽ di chuyển tương ứng
- Phần này cần lưu ý về Rebound, đó là khi hình tròn chạm biên sẽ nảy ngược lại
- Thuật toán vẽ hình tròn được hiểu đơn giản như sau:

    + Lưu màu đen vào thanh ghi $s2

    + Vẽ hình tròn màu đen ở vị trí tương ứng của hình tròn cũ (Xóa hình tròn)

    + Lưu lại màu vàng vào thanh ghi $s2

    + Dịch chuyển vị trí của tâm hình tròn (bằng cách thay đổi $s0 hoặc $s1 với $s4)

    + Vẽ hình tròn mới dựa trên vị trí tâm mới

- Hàm speed và update_speed

    + Tăng tốc độ của hình tròn bằng cách dịch trái $s4

    + Nếu $s4 lớn hơn 8 thì nhảy đến update_speed để cập nhật tốc độ tối đa là 8

- Hàm slow và update_slow

    + Tương tự như speed và update_speed

Phần vẽ hình tròn (DrawCircle)

```
#-----------------------------------------------------------------------------------------------#
DrawCircle:                         #Using Midpoint Circle Algo
```

```
#CREATE STACK TO STRORE DATAL POINT, COLOR, ...
        addi    $sp, $sp, -20           #Make room on stack for 1 words
        sw      $ra, 0($sp)             #Store $ra on element 0 of stack
        sw      $s0, 4($sp)             #Store $a0 on element 1 of stack
        sw      $s1, 8($sp)             #Store $a1 on element 2 of stack
        sw      $s2, 12($sp)                    #Store $a2 on element 3 of stack
        sw      $s3, 16($sp)                    #Store $a3 on element 4 of stack

        #VARIABLES
        move    $t0, $s0                #x0
        move    $t1, $s1                #y0
        move    $t2, $s3                #radius
        addi    $t3, $t2, 0                     #x
        li      $t4, 0                  #y

        li      $t7, 0                  #Err

        #While(x >= y)

circleLoop:
        blt     $t3, $t4, skipCircleLoop    #If x < y, skip circleLoop
        #s5 = x, s6 = y
        #Draw Dot (x0 + x, y0 + y)
        addu    $s5, $t0, $t3
        addu    $s6, $t1, $t4
        lw      $s2, 12($sp)
        jal     drawDot         #Jump to drawDot
```

```
#Draw Dot (x0 + y, y0 + x)
addu      $s5, $t0, $t4
addu      $s6, $t1, $t3
lw     $s2, 12($sp)
jal    drawDot                    #Jump to drawDot
#Draw Dot (x0 - y, y0 + x)
subu      $s5, $t0, $t4
addu      $s6, $t1, $t3
lw     $s2, 12($sp)
jal    drawDot                    #Jump to drawDot
#Draw Dot (x0 - x, y0 + y)
subu      $s5, $t0, $t3
addu      $s6, $t1, $t4
lw     $s2, 12($sp)
jal    drawDot                    #Jump to drawDot
#Draw Dot (x0 - x, y0 - y)
subu      $s5, $t0, $t3
subu      $s6, $t1, $t4
lw     $s2, 12($sp)
jal    drawDot                    #Jump to drawDot
#Draw Dot (x0 - y, y0 - x)
subu      $s5, $t0, $t4
subu      $s6, $t1, $t3
lw     $s2, 12($sp)
jal    drawDot                    #Jump to drawDot
#Draw Dot (x0 + y, y0 - x)
addu      $s5, $t0, $t4
```

```
        subu    $s6, $t1, $t3
        lw    $a2, 12($sp)
        jal    drawDot                    #Jump to drawDot
        #Draw Dot (x0 + x, y0 - y)
        addu    $s5, $t0, $t3
        subu    $s6, $t1, $t4
        lw    $s2, 12($sp)
        jal    drawDot                    #Jump to drawDot
            #If (err <= 0)
            bgtz    $t7, doElse
            addi    $t4, $t4, 1            #y++
            sll $t8, $t4, 1                #Bitshift y left 1
            addi $t8, $t8, 1                    #2y + 1
            addu $t7, $t7, $t8             #Add  e + (2y + 1)
            j    circleContinue                 #Skip else stmt

            #Else If (err > 0)
            doElse:
            addi    $t3, $t3, -1                #x--
            sll $t8, $t3, 1                #Bitshift x left 1
            addi $t8, $t8, 1                    #2x + 1
            subu $t7, $t7, $t8             #Subtract e - (2x + 1)

j circleContinue

circleContinue:
            #LOOP
```

```
    j     circleLoop
    #CONTINUE
    skipCircleLoop:
    #RESTORE $RA
    lw     $ra, 0($sp)              #Restore $ra from stack
    addiu       $sp, $sp, 20        #Readjust stack
    jr $ra
    nop
```

Giải thích:

- Hàm này dựa trên logic của thuật toán Midpoint Circle Drawing Algorithm
- Chia hình tròn thành 8 phần bằng nhau, mỗi một circleloop sẽ lần lượt vẽ 8 điểm trên hình tròn, trong đó ở lần lặp đầu tiên mỗi điểm bị lặp 2 lần nên chỉ có 4 điểm được vẽ.
- được gọi ra từ câu lệnh "jal DrawCircle" nên ở cuối vòng lặp jr $ra để trả về đi chỉ trước khi thực hiện hàm.

Hàm vẽ điểm (CircleDot)

```
drawDot:
    #li $a2, YELLOW
    add $at, $s6, $0
    sll    $at, $at, 9              # calculate offset in $at: at = y_pos * 512
    add    $at, $at, $s5                  # at = y_pos * 512 + x_pos = "index"
    sll    $at, $at, 2              # at = (y_pos * 512 + x_pos)*4 = "offset"
    add    $at, $at, $v1                  # at = v1 + offset
    sw     $s2, ($at)               # draw it!
    jr $ra
```

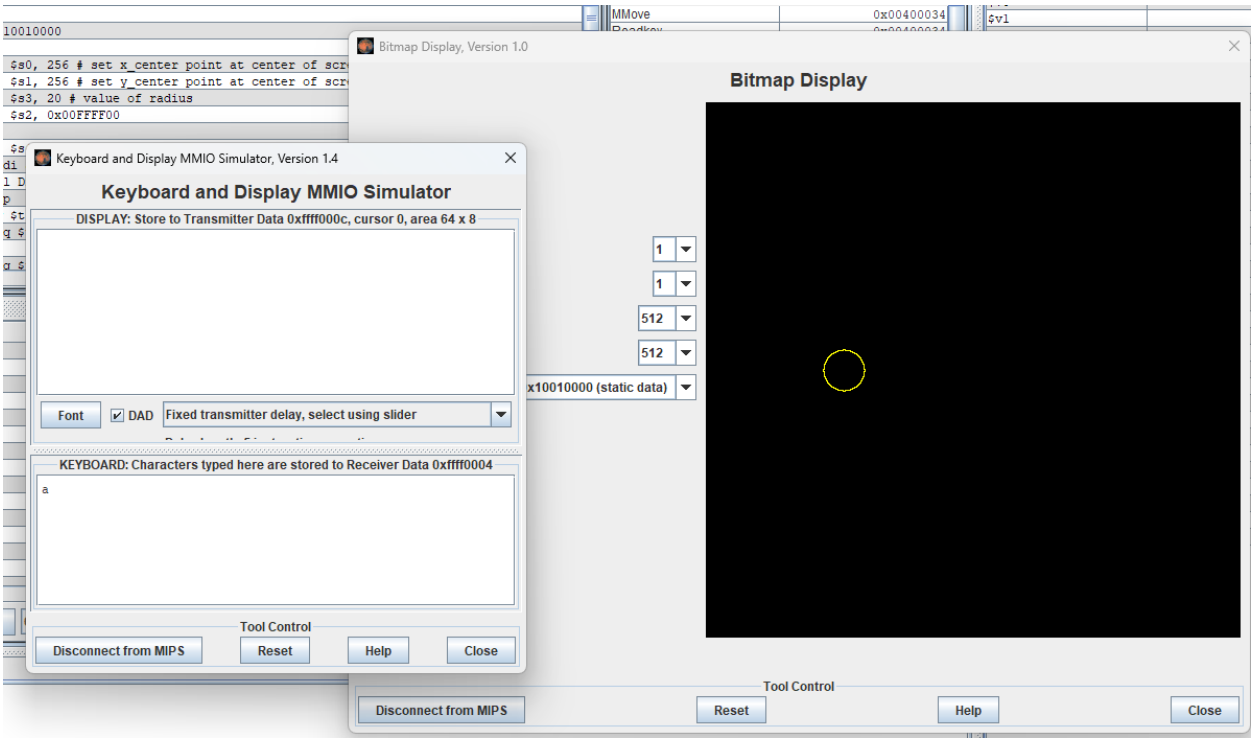
Giải thích:

- Sao chép giá trị tọa độ y vào $at

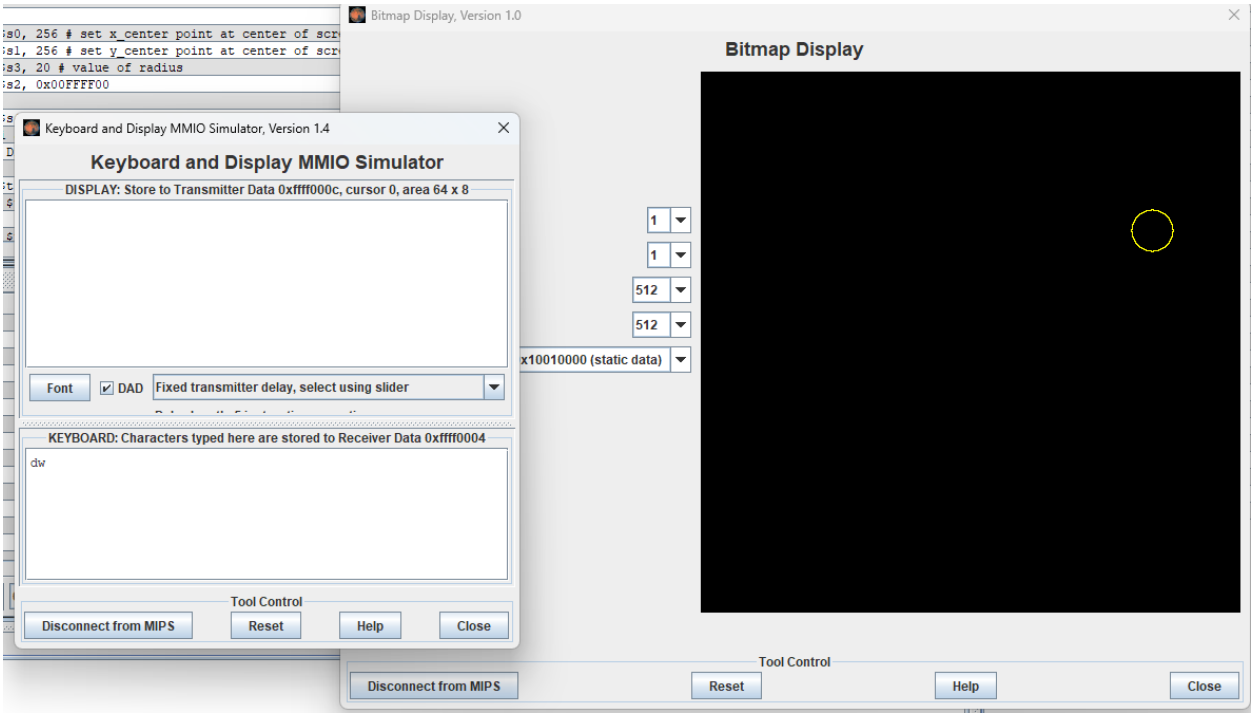
- Nhân tọa độ y với 512 (nhằm chuyển tọa độ y thành vị trí dòng trong bộ nhớ màn hình. Mỗi dòng có 512 điểm ảnh (pixel), do đó dịch trái 9 bit sẽ cho chúng ta vị trí bắt đầu của dòng y trong bộ nhớ.)
- Cộng tọa độ x vào (Sau lệnh này, $at chứa chỉ số của điểm ảnh (pixel) trong bộ nhớ, tương ứng với tọa độ (x, y).)
- Nhân với 4 để tính toán vị trí bộ nhớ vì mỗi điểm ảnh (pixel) được biểu diễn bằng 4 byte (32 bit), việc nhân với 4 sẽ chuyển chỉ số điểm ảnh thành địa chỉ byte trong bộ nhớ.
- Thêm giá trị MONITOR_SCREEN để tính toán địa chỉ chính xác
- Lưu màu vào địa chỉ đã tính toán

Kết quả:

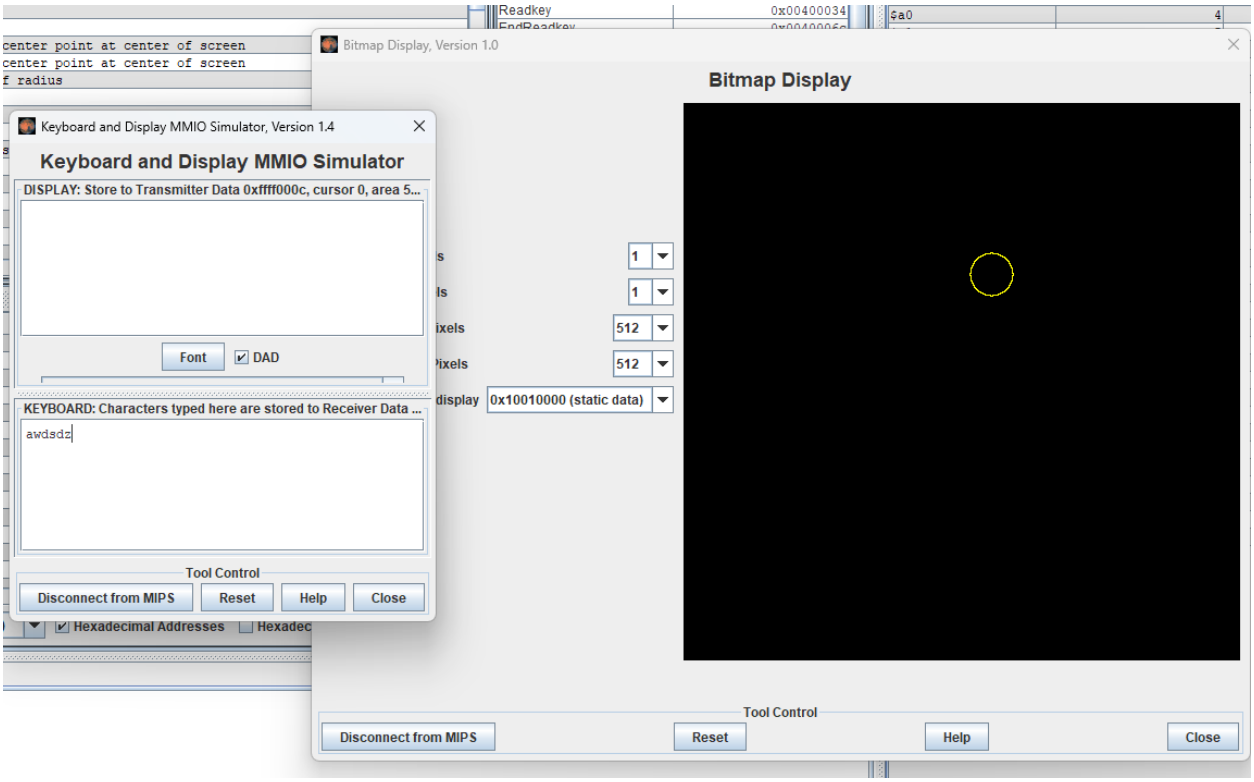
Di chuyển sang trái



Di chuyển sang phải và lên trên đồng thời



Kết hợp một chuỗi các lệnh



Minh họa cho việc nhập “a” vào KEY_CODE:

Bước 1: Thiết lập ban đầu

- Chương trình khởi tạo các giá trị cho vị trí trung tâm của hình tròn và bán kính.
- Hình tròn được vẽ tại vị trí (256, 256) với bán kính 20 và màu vàng (YELLOW).

Bước 2: Chương trình bắt đầu vòng lặp MMove

- Chương trình đi vào nhãn MMove để bắt đầu vòng lặp di chuyển hình tròn.

Bước 3: Đọc phím (Readkey)

- Chương trình đọc giá trị từ địa chỉ KEY_CODE (0xFFFF0004) vào thanh ghi $t0.

Bước 4: So sánh giá trị phím (beq)

- Chương trình kiểm tra giá trị của $t0 để xác định phím nào được nhấn:
    - **beq $t0, 97, left**: Nếu giá trị là 97 (ASCII của 'a'), nhảy đến nhãn left.

Bước 5: Xử lý phím 'a' (left)

- **addi $t0, $0, 97**: Đặt giá trị 97 vào thanh ghi $t0.
- **sw $t0, 0($k0)**: Lưu giá trị này vào địa chỉ KEY_CODE.
- **bltu $s0, 20, right**: Nếu tọa độ x của hình tròn ($s0) nhỏ hơn 20, nhảy đến nhãn right (không thực hiện vì $s0 = 256).
- **li $s2, 0x00000000**: Đặt màu là màu đen để xóa hình tròn cũ.
- **jal DrawCircle**: Gọi hàm DrawCircle để xóa hình tròn cũ.

Bước 6: Xóa hình tròn cũ (DrawCircle)

- Hàm DrawCircle được gọi với màu đen để xóa hình tròn tại vị trí (256, 256).
- Các điểm của hình tròn được tính toán và màu đen được đặt vào các vị trí tương ứng trong bộ nhớ màn hình.

Bước 7: Thiết lập lại màu vàng cho hình tròn (YELLOW)

- **li $s2, YELLOW**: Đặt lại màu vàng cho hình tròn.
- **sub $s0, $s0, $s4**: Giảm tọa độ x của hình tròn đi một khoảng $s4 (tức là di chuyển hình tròn sang trái).
    - Nếu $s4 = 1, tọa độ x mới là 256 - 1 = 255.

Bước 8: Vẽ lại hình tròn tại vị trí mới (DrawCircle)

- Gọi lại hàm DrawCircle để vẽ hình tròn tại vị trí mới (255, 256) với màu vàng.

Bước 9: Quay lại đọc phím (Readkey)

- Chương trình quay lại nhãn Readkey để tiếp tục đọc phím mới từ người dùng.

Phần II: Kiểm tra tốc độ và độ chính xác khi gõ văn bản

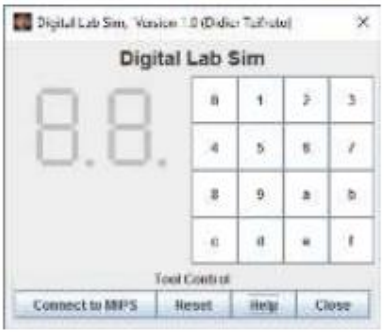
Thực hiện chương trình đo tốc độ gõ bàn phím và hiển thị kết quả bằng 2 đèn led 7 đoạn. Nguyên tắc:

- Cho một đoạn văn bản mẫu, cố định sẵn trong mã nguồn. Ví dụ “bo mon ky thuat may tinh”

- Sử dụng bộ định thời Timer (trong bộ giả lập Digital Lab Sim) để tạo ra khoảng thời gian để đo. Đây là thời gian giữa 2 lần ngắt, chu kì ngắt.

- Người dùng nhập các kí tự từ bàn phím. Ví dụ nhập “bo mOn ky 5huat may tinh”. Chương trình cần phải đếm số kí tự đúng (trong ví dụ trên thì người dùng gõ sai chữ O và 5) mà người dùng đã gõ và hiển thị lên các đèn led.

- Chương trình đồng thời cần tính được tốc độ gõ: thời gian hoàn thành và số từ trên một đơn vị thời gian.



Code

```
.eqv SEVENSEG_LEFT      0xFFFF0011    # Dia chi cua den led 7 doan trai
.eqv SEVENSEG_RIGHT     0xFFFF0010    # Dia chi cua den led 7 doan phai
```

```
.eqv IN_ADRESS_HEXA_KEYBOARD        0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD       0xFFFF0014
.eqv KEY_CODE                       0xFFFF0004    # ASCII code from keyboard, 1 byte
.eqv KEY_READY                      0xFFFF0000    # =1 if has a new keycode ?
.eqv MASK_CAUSE_KEYBOARD                0x0000034      # Keyboard Cause

.data
num_for_led:    .byte 63,6,91,79,102,109,125,7,127,111
String_space :  .space 1000                     #khoang trong de luu cac ky tu nhap tu ban phim.
stringsource :   .asciiz "bo mon ky thuat may tinh"
Mess1:              .asciiz "\n So ky tu nhap trong 1s la :  "
Mess2:              .asciiz  "\n So ky tu dung la: "
Mess3:              .asciiz "\n Ban co muon chay lai chuong trinh? "
Mess4:              .asciiz "\n Thoi gian nhap la:  "
Mess5:              .asciiz "\n Toc do go trung binh la(ky tu/1s): "
Mess6:              .asciiz "\n Vui long nhap ky tu!!!!!!"
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# MAIN
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.text
        li   $k0,  KEY_CODE
        li   $k1,  KEY_READY
Main:
        li $s2,0                    # s2 = so ky tu nhap vao
        li $s3,0                    # s3 = so vong lap da thuc hien (max = 200 ~ 1s)
        li $s4,10
        li $s5,200                  # s5 = max vong lap
        li $s6,0                    # s6 = so ky tu nhap duoc trong 1s
        li $s7,0                    #Bien danh dau het chuong trinh
```

```
        li $a1,0                    # a1 = thoi gian go
Loop:
WAIT_FOR_KEY:
        lw   $t1, 0($k1)            # $t1 = [$k1] = KEY_READY
        beq  $t1, $zero,Check       # if $t1 == 0 then Polling
MAKE_INTER:
        addi $s6,$s6,1              #Tang bien dem ky tu nhap duoc trong 1s len 1
        teqi $t1, 1                 # if $t1= 1 then raise an Interrupt
#-------------------------------------------------------
# Dem so ky tu go duoc trong 1s
#-------------------------------------------------------
Check:
        #Neu chua du 200 vong lap(1s) thi sleep.
        addi    $s3, $s3, 1         # Tang so vong lap len 1
        div $s3,$s5                 #Lay so vong lap chia cho 200 de xac dinh da duoc 1s hay
chua
        mfhi $t2                    #Luu phan du cua phep chia tren
        bnez $t2,Sleep              #Neu chua duoc 1s nhay den Sleep

        #Neu da duoc 1s thi nhay den nhan SetCount de thuc hien in ra man hinh
        addi $a1,$a1,1              #Tang thoi gian go len 1s
#=============================================================================
=========
# DA DU 1s (in ra so ky tu da nhap trong 1s o run I/O va LED 7 thanh)
#=============================================================================
=========
SetCount:
        li $s3,0            #Tao lai so vong lap cho 1s tiep
        li $v0,4            #In Mess1 : so ky tu trong 1s:
        la $a0,Mess1
```

```
        syscall
        li   $v0,1                      #In ra so ky tu trong 1s
        add  $a0,$zero,$s6
        syscall
DISPLAY_DIGITAL:
        div $s6,$s4                     #Lay so ky tu nhap duoc trong 1s chia cho 10
        mflo $t2                        #Lay phan nguyen de hien thi o Led trai
        la $t3,num_for_led              #Lay dia chi num_for_led
        add $t3,$t3,$t2                 # Chi toi dia chi gia tri can hien thi
        lb $a0,0($t3)                   #Lay gia tri cho vao $a0
        jal  SHOW_7SEG_LEFT             #Hien thi Led trai
#---------------------------------------------------------------------------------
        mfhi $t2                        #Lay phan du de hien thi o Led phai
        la $t3,num_for_led              #Lay dia chi num_for_led
        add $t3,$t3,$t2                 # Chi toi dia chi gia tri can hien thi
        lb $a0,0($t3)                   #Lay gia tri cho vao $a0
        jal  SHOW_7SEG_RIGHT            #Hien thi Led phai

        li $s6,0                        #Khoi tao lai so ky tu trong 1s cho 1s tiep
        beq $s7,1,Loop_for_another_input
#=================================================================================
# CHUA DU 1s
#=================================================================================
Sleep:
        li $v0,32
        li $a0,5                        #Sleep 5ms
        syscall
```

```
        nop
        j Loop                          #Quay lai Loop
#================================================================================
=========
# END_MAIN
#================================================================================
=========
End_Main:
        li $v0,10
        syscall
        nop
#================================================================================
===========
#In ra led 7 thanh
#================================================================================
===========
SHOW_7SEG_LEFT:
        li   $t0,  SEVENSEG_LEFT        # assign port's address
        sb   $a0,  0($t0)               # assign new value
        jr   $ra
SHOW_7SEG_RIGHT:
        li   $t0,  SEVENSEG_RIGHT       # assign port's address
        sb   $a0,  0($t0)               # assign new value
        jr   $ra
#================================================================================
=========
#XU LY NGAT
#================================================================================
=========
.ktext  0x80000180
        mfc0 $t1,$13                    #Phan_Nguyen nhan ngat
```

```
        li   $t2, MASK_CAUSE_KEYBOARD      #Ngyen nhan ngat do ban phim
        and  $at,$t1,$t2                   #So sanh nguyen nhan ngat
        beq  $at,$t2,CountKey              #Neu do ban phim thi nhay den CountKey
        j    End_Inter_Process             #Khong thi nhay den End_Inter_Process
CountKey:
        lb   $t0, 0($k0)                   # $t0 = [$k0] = KEY_CODE
        la $t5,String_space                #Dia chi luu xau duoc nhap
        add $t5,$t5,$s2                    #Nhay den dia can ghi cua chi ki vua nhap
        sb $t0,0($t5)                      #Luu ki tu vua nhap
        addi $s2,$s2,1                     #Tang so ki tu len 1
        bne $t0,10,End_Inter_Process       #Neu khong phai ky tu la enter nhay den
End_Inter_Process
        beq $s2,1,Error                    #Neu la xau rong nhay den Error
        bnez $a1,End                       #Neu thoi gian hoan thanh >=1 thi nhay den End
        addi $a1,$a1,1                     #Neu khong thi lam tron roi nhay
        j End
End_Inter_Process:
NEXT_PC:
        mfc0 $at, $14                      # $at <= Coproc0.$14 = Coproc0.epc
        addi $at, $at, 4                   # $at = $at + 4 (next instruction)
        mtc0 $at, $14                      # Coproc0.$14 = Coproc0.epc <= $at
RETURN:
        eret                               #Tro ve lenh tiep theo trong Main
#================================================================================
========================
End:
        li $v0,11
        li $a0,'\n'                        #In xuong dong
        syscall
```

```
        li $t1,0                        #Dem so ki tu da xet
        li $t2,0                        #Dem so ky tu dung
        li $t3,24                       #Do dai xau cua ma nguon
        slt $t4,$s2,$t3                 #So sanh do dai 2 xau
        #Xau nao nho hon thi duyet theo xau do
        beqz $t4,Check_String
        add $t3,$zero,$s2               #Gan $t3=so ky tu da nhap de xet
        add $t3,$t3,-1                  #Bo qua ky tu enter
Check_String:
        la $t7 String_space             #Lay dia chi xau da nhap
        add $t7,$t7,$t1                 #Nhay den dia chi ky tu dang xet
        li $v0,11                       #In ra ky tu dang xet
        lb $t4,0($t7)                   #Lay ky tu dang xet de in
        add $a0,$zero,$t4
        syscall
        la $t5,stringsource             #Lay dia chi xau nguon
        add $t5,$t5,$t1                 #Nhay den dia chi ky tu dang xet
        lb $t6,0($t5)                   #Lay ky tu can xet de so sanh
        bne $t4,$t6,Continue            #Neu khac thi nhay den Continue
        add $t2,$t2,1                   #Neu giong thi tang bien dem len 1
Continue:
        addi $t1,$t1,1                  #Tang bien dem de xet ky tu tiep theo
        beq $t1,$t3,Print               #Neu da duyet het ky tu thi nhay den Print
        j Check_String                  #Neu chua thi quay lai check tiep
Print:
        li $v0,4                        #In ra Mess2
        la $a0,Mess2
        syscall
        li $v0,1                        #In ra so ky tu dung
```

```
        add $a0,$zero,$t2
        syscall
        li $v0,4                        #In Mess4
        la $a0,Mess4
        syscall
        li $v0,1                        #In thoi gian hoan thanh
        add $a0,$zero,$a1
        syscall
        jal So_Ky_Tu_Tren_1s            #Tinh toc do go trung binh
        li $s7,1                        #Danh dau ket thuc chuong trinh
        add $s6,$zero,$t2               #Cho $s6=$t2 de hien thi tren led
        b DISPLAY_DIGITAL
Loop_for_another_input:
        li $v0,50                       #In Mess3
        la $a0,Mess3
        syscall
        beqz $a0,Main
        b End_Main
Error:
        li $v0,4                        #In Mess6 : Vui long nhap ki tu
        la $a0,Mess6
        syscall
        b Main
So_Ky_Tu_Tren_1s:
        li $v0,4                        #In Mess5: toc do go trung binh
        la $a0,Mess5
        syscall
        div $s2,$a1                     #Lay so ky tu chia cho thoi gian go
        li $t8,0                        #Bien dem so chu so sau dau phay
```

```
        li $a2,0                        #Phan nguyen cua phep chia
Phan_Nguyen:
        mfhi $a3                        #Lay phan du de kiem tra
        beqz $a3,Show                   #Neu chia het thi in ra man hinh
        mflo $a0                        #Neu khac 0 thi xet phan nguyen
Show:
        mflo $a2
        li $v0,1                        #In phan nguyen
        add $a0,$zero,$a2
        syscall
        li $v0,11
        li $a0,'.'                      #In dau '.'
        syscall
Phan_Thap_Phan:
        beq $t8,2,End_Print             #Lay 2 so sau dau phay
        mulo $a3,$a3,$s4                #Nhan phan du truoc voi 10
        div $a3,$a1                     #Chia tiep de lay phan thap phan
        mflo $a3                        #Lay phan nguyen de in
        li $v0,1                        #In ra phan nguyen
        add $a0,$zero,$a3
        syscall
        mfhi $a3                        #Lay phan du de kiem tra va cho vong lap sau
        beqz $a3,End_Print              #Neu phan du =0 => Ket thuc
        addi $t8,$t8,1                  #Neu khong thi tang bien dem len 1 va lap tiep
        j Phan_Thap_Phan
End_Print:      jr $ra
```

Giải thích:

Đề bài: Kiểm tra tốc độ và độ chính xác khi gõ văn bản

Thực hiện chương trình đo tốc độ gõ bàn phím và hiển thị kết quả bằng 2 đèn led 7 đoạn. Nguyên tắc:

- Cho một đoạn văn bản mẫu, cố định sẵn trong mã nguồn. Ví dụ “bo mon ky thuat may tinh”

- Sử dụng bộ định thời Timer (trong bộ giả lập Digital Lab Sim) để tạo ra khoảng thời gian để đo. Đây là

thời gian giữa 2 lần ngắt, chu kì ngắt.

- Người dùng nhập các kí tự từ bàn phím. Ví dụ nhập “bo mOn ky 5huat may tinh”. Chương trình cần phải

đếm số kí tự đúng (trong ví dụ trên thì người dùng gõ sai chữ O và 5) mà người dùng đã gõ và hiển thị lên

các đèn led.

- Chương trình đồng thời cần tính được tốc độ gõ: thời gian hoàn thành và số từ trên một đơn vị thời gian.

* Cách làm:

Lưu giá trị:

$s2 là tổng số ký tự đã nhập vào

$s3 là số vòng lặp đã thực hiện

$s5 là số vòng lặp cần hoàn thành để đủ 1s

$s6 là số ký tự đã nhập trong 1s

$a1 là thời gian gõ kí tự

$s7 là cờ xem chương trình đã hoàn thành chưa

Chương trình tính thời gian bằng cách chạy vòng lặp, mỗi vòng lặp sẽ sleep 5ms, nếu chạy đủ 200 vòng lặp sẽ tính là 1s. So sánh với chuỗi cho trước bằng cách lưu chuỗi mới nhập vào String_space

- Sử dụng vòng lặp vô hạn để kiểm tra xem có 1 kí tự nào được nhập vào bàn phím chưa,

+ nếu có thì $t1 = 0($k1)=1 tại KEY_READY , và $s6(số kí tự đã nhập trong 1s) tăng thêm 1 chương trình sẽ xảy ra ngắt và nhảy xuống phần xử lý ngắt

+ Nếu chưa thì sẽ tiếp tục chạy vòng lặp để tính thời gian

-Tại phần xử lý ngắt (.ktext): chương trình sẽ kiểm tra xem nguyên nhân ngắt là do đâu bằng cách lấy giá trị trong Coproc0.cause($13) và so sánh với MASK_CAUSE_KEYBOARD

+ Nếu đúng do bàn phím chương trình sẽ nhảy xuống CountKey để lưu ký tự vùa nhập vào space_string tăng tổng số kí tự đã nhập($s2) lên 1, nếu là enter chương trình hỏi có muộn thực hiện lại không và dừng

, nếu không phải thì sẽ kết thúc xử lý ngắt và quay chở lại vòng lặp tính thời gian

-Tại vòng lặp tính thời gian đã chôi qua (s):

+ Chương trình sẽ đếm số vòng lặp đã thực hiện bằng cách lấy $s3 chia cho $s5(200) và lấy phần dư, nếu dư khác 0 thì nhảy xuống sleep và sleep 5ms và quay trở lại Loop ban đầu để chờ kí tự nhập tiếp theo,

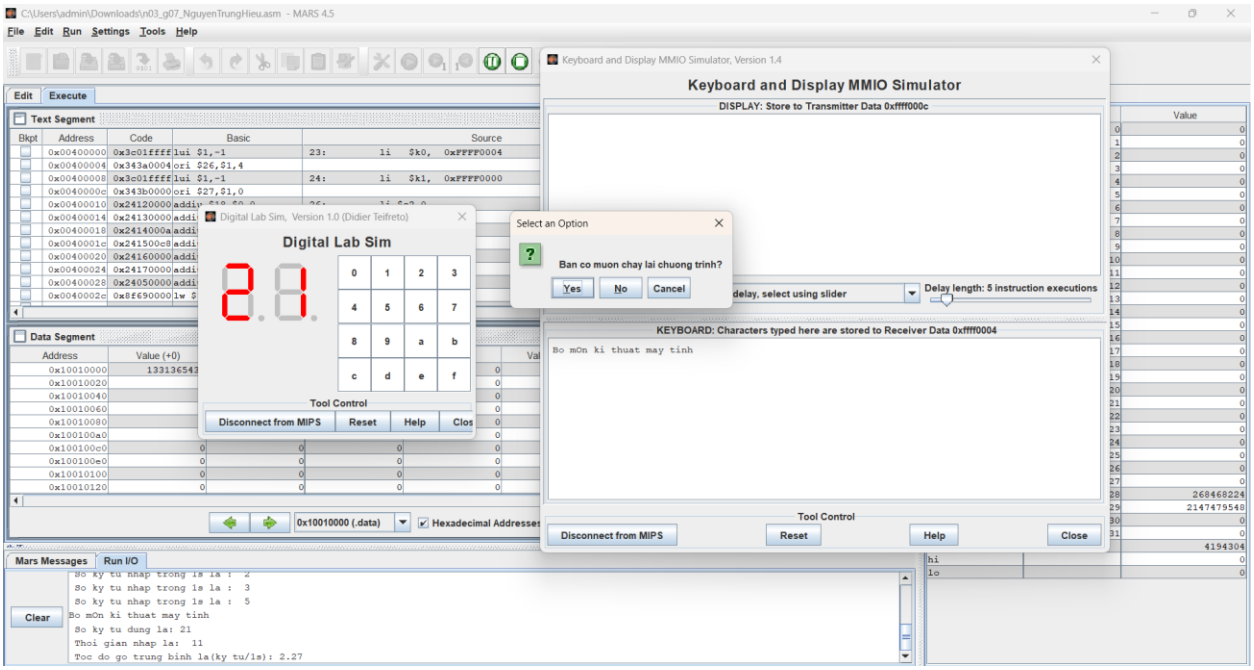
+ Nếu dư bằng 0 có nghĩa là đã thực hiện đủ 200 vòng lặp là 1s thì chương trình sẽ in ra số từ đã nhập trong 1s ở Run I/O và LED 7 thanh và cập nhật lại số ký tự đã nhập trong 1s($s6) cũng như số vòng lặp đã thực hiện ($s3) về 0

-Chương trình sẽ ngừng đếm khi nhân Enter, và in ra màn hình kết quả :

+Kiểm tra lần lượt từng phần tử trong String_space so với string_source và đếm số phần tử đúng

+Tính tốc độ gõ trung bình bằng cách chia tổng số ký tự đã gõ($s2) cho thời gian gõ($a1) lấy phần nguyên và phần thập phân

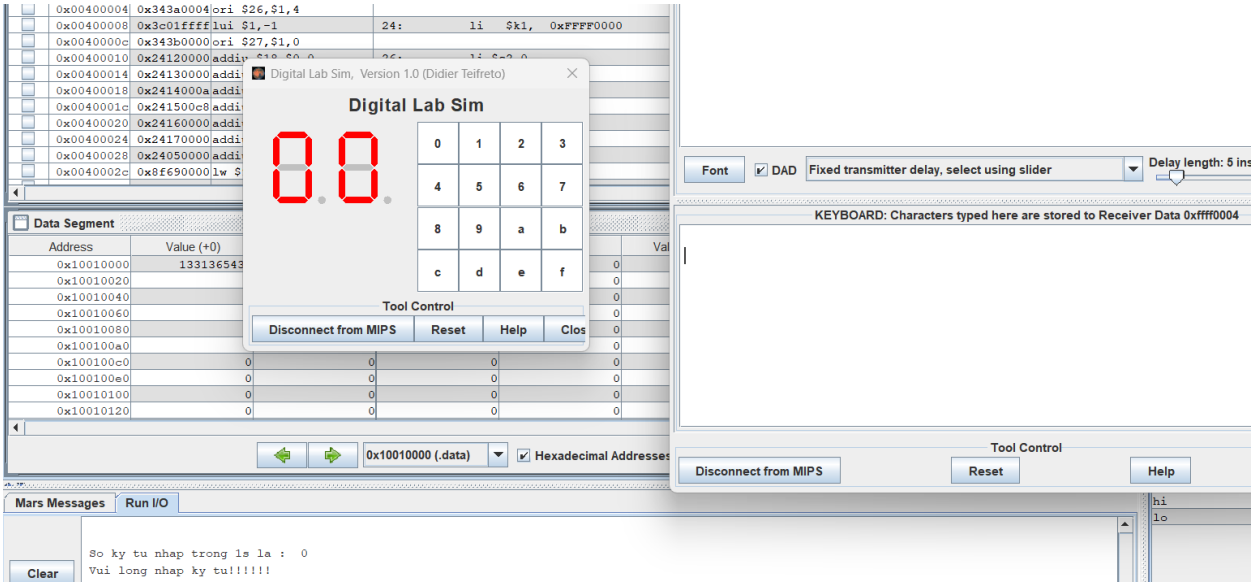
Kết quả



Khi nhập từ keyboard chuỗi “Bo mOn ki thuat may tinh” đúng 21 kí tự(tính cả khoảng trống so với chuỗi gốc “bo mon ky thuat may tinh” (24 kí tự)

Chương trình in ra thời gian gõ là 11s và tốc độ gõ trung bình là 2.27 ký tự /s

Bấm No để kết thúc chương trình, Yes để thực hiện lại



Khi nhấn enter luôn, chương trình báo lỗi, yêu cầu người dùng nhập ký tự vào