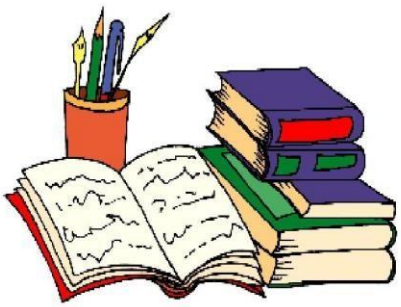
ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÁO CÁO CUỐI KỲ
THỰC HÀNH KIẾN TRÚC MÁY TÍNH

GIẢNG VIÊN HƯỚNG DẪN : **Ths. LÊ BÁ VUI**

SINH VIÊN : **BÙI TRUNG HÙNG** **MSSV: 20200255**

ĐINH TRỌNG HUY **MSSV: 20200269**



Hà Nội, Ngày 22 Tháng 7 Năm 2022

Bài 2: Vẽ hình trên màn hình Bitmap

Sinh viên thực hiện: Bùi Trung Hùng - 20200255

A. Đề bài yêu cầu:

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:

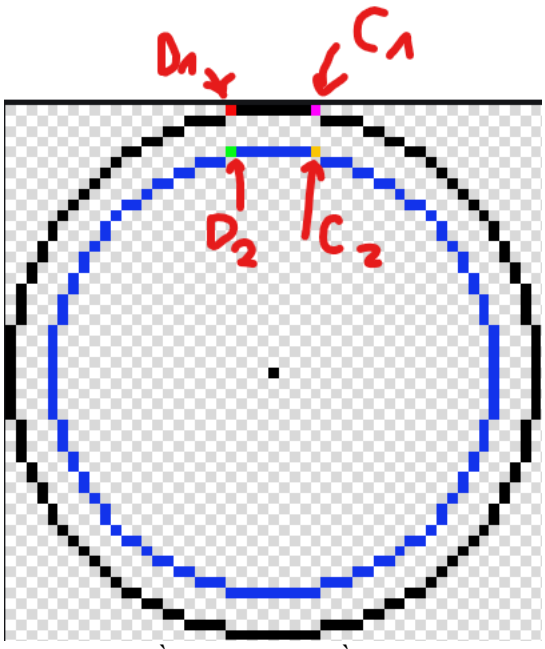
- Thiết lập màn hình ở kích thước 512x512. Kích thước pixel 1x1.
- Chiều di chuyển phụ thuộc vào phím người dùng bấm, gồm có (di chuyển lên (W), di chuyển xuống(S), sang trái(A), sang phải(D), tăng tốc độ(Z), giảm tốc độ(X) trong bộ giả lập Keyboard and Display MMIO Simulator).
- Vị trí bóng ở giữa màn hình.

B. Phân tích yêu cầu:

- Thiết lập màn hình ở kích thước 512x512. Kích thước pixel 1x1.
- Vẽ bóng và vẽ ở giữa màn hình.
- Kết nối để người dùng có thể sử dụng Keyboard and Display MMIO Simulator để di chuyển quả bóng.
- Bóng di chuyển đạp vào cạnh thì sẽ di chuyển theo chiều ngược lại.

C. Cách làm:

- Đầu tiên ta vẽ quả bóng, bằng cách tô màu đường tròn bằng màu vàng. Tức là ta tô màu một loạt các pixel trên màn hình mô phỏng Bitmap và tạo ra một đường tròn màu vàng.
    - Cách tạo đường tròn:



- Ta cần tô màu 2 viền tròn bên trên và các điểm bên trong màu bằng màu vàng, ta sẽ thu được đường tròn vàng
  - Vòng ngoài: từ dòng 1→51
  - Vòng trong: từ dòng 5→46
  - Trong hình ta thấy:
    - D1 là điểm đầu tiên bên trái của đường tròn ngoài.
    - C1 là điểm cuối cùng bên phải của đường tròn ngoài.
    - D2 là điểm đầu tiên bên trái của đường tròn trong.
    - C2 là điểm cuối cùng bên phải của đường tròn trong.
- Cách tô: Tô từ trên xuống , tô từng dòng, tô từng pixel từ trái qua phải
  - Từ 1→5: Tô từ D1 đến C1.
  - Từ 6→46: Tô từ D1 đến D2 và từ C2 đến C1.
  - Từ 47→51: Tô từ D1 đến C1.

- Khi người dùng nhập ký tự (w,a,s,d) vào trong Keyboard and Display MMIO Simulator thì bóng di chuyển theo hướng mà ký tự nhập quy định.
- Để quả bóng di chuyển thì ta sẽ vẽ quả bóng mới và xóa quả bóng ở vị trí cũ (tô màu quả bóng giống màu nền). Vị trí mới dựa trên người nhập vào Keyboard and Display MMIO Simulator:
  - W: Vị trí mới = Vị trí cũ -512. Tức là dịch lên một dòng .
  - S: Vị trí mới = Vị trí cũ +512. Tức là dịch xuống một dòng.
  - A: Vị trí mới = Vị trí cũ -1. Dịch sang trái một cột.
  - D: Vị trí mới = Vị trí cũ +1. Dịch sang phải một cột.
- Khi chạm viền thì di chuyển ngược lại.

D. Mã nguồn:

Code MIPS assembly và giải thích code:

```
1  .eqv MONITOR_SCREEN 0x10010000 #Dia chi bat dau cua bo nho man hinh
2
3  .eqv TopHead_1 118012  # D1 ban dau
4  .eqv TopTeal_1 118020  # C1 ban dau
5
6  .eqv TopHead_2 120060  # D2 ban dau
7  .eqv TopTeal_2 120068  # C2 ban dau
8
9
10 .eqv KEY_CODE 0xFFFF0004       # ASCII code from keyboard, 1 byte
11 .eqv KEY_READY 0xFFFF0000      # =1 if has a new keycode ?
12  # Auto clear after lw
13
14 .eqv DISPLAY_CODE 0xFFFF000C   # ASCII code to show, 1 byte
15 .eqv DISPLAY_READY 0xFFFF0008  # =1 if the display has already to do
16  # Auto clear after sw
```

Giải thích code bên trên:

- Khai báo các hằng và gán giá trị
- Dòng 1, 10, 11, 14, 15 : là các hằng mang giá trị địa chỉ đầu vào và ra của các công cụ bitmap, bộ giả lập MMIO
- Dòng 3, 4, 6, 7: là vị trí bắt đầu vẽ quả bóng

```
20 .text
21
22 # ---------------------------------------------------------------------------
23 # TAC DUNG CUA CAC THANH GHI TRONG CODE TAO HINH TRON
24 #----------------------------------------------------------------------------
25 # s0 : dem so dong
26 # s1, s2 : 2 bien trong ham tinh cal
27 # s3, s4 : 2 bien trong ham tinh calcu
28 # s5, s6 : 2 bien trong ham to mau
29 # t8: dia chi diem dau stack
30 # t9: dia chi diem cuoi stack
31 # ---------------------------------------------------------------------------
32
```

Giải thích code bên trên:

- Em thường comment lại những thanh ghi được sử dụng 1 chức năng xuyên suốt một đoạn code lớn lên trên đầu để dễ đối chiếu và xem xét.

```
33
34 # ---------------------------------------------------------------------------
35 #       CODE TAO HINH TRON
36 # ---------------------------------------------------------------------------
37         li $k0, MONITOR_SCREEN         # Nap dia chi bat dau cua man hinh
38         li $s0, 1              # So dong
39         add $t8, $sp, $zero    # Luu dia chi diem dau stack
40
```

Giải thích code bên trên:

- Gán k0 = địa chỉ bắt đầu của màn hình bitmap
- Cho biến số dòng s0 = 1
- Lưu địa chỉ điểm đầu stack vào t8

```
41
42  # Ham xet tung dong
43  main:
44
45  main_root:
46          # cac bien dau tien cho ham cal*
47          li $s1, TopHead_1
48          li $s2, TopTeal_1
49
50          # cac bien dau tien cua ham calcu*
51          li $s3, TopHead_2
52          li $s4, TopTeal_2
53
54          # To mau dong dau tien
55          li $s5, TopHead_1
56          li $s6, TopTeal_1
57          jal color_main
58
59          addi $s0, $s0, 1        # tang so dong
```

Giải thích code bên trên:

- Gán D1 vào s1, D2 vào s2, C1 vào s3, C2 vào s4
- Gán D1 vào s5, D2 vào S6 để tô màu từ trái (D1) sang phải (D2)
- Sau mỗi lần tô màu xong 1 dòng thì tăng giá trị biến s0 (biến đếm số dòng)

```
main_ele_circle_1:
        # dong 2
        beq $s0, 2, main_ele_1
        # dong 3 -> 5
        slti $t1, $s0, 6
        li $t3, 2
        slt $t2, $t3, $s0
        add $t4, $t1, $t2
        beq $t4, 2, main_ele_3
        # dong 6 -> 12
        slti $t1, $s0, 13
        li $t3, 5
        slt $t2, $t3, $s0
        add $t4, $t1, $t2
        beq $t4, 2, main_ele_4
        # dong 13
        beq $s0, 13, main_ele_5
        # dong 14
        beq $s0, 14, main_ele_4
        # dong 15
        beq $s0, 15, main_ele_5
        # dong 16
        beq $s0, 16, main_ele_4
        # dong 17
        beq $s0, 17, main_ele_5
        # dong 18
        beq $s0, 18, main_ele_4
        # dong 19 -> 21
        slti $t1, $s0, 22
        li $t3, 18
        slt $t2, $t3, $s0
        add $t4, $t1, $t2
        beq $t4, 2, main_ele_5
        # dong 22
        beq $s0, 22, main_ele_4
        # dong 23 -> 30
        slti $t1, $s0, 31
        li $t3, 22
        slt $t2, $t3, $s0
        add $t4, $t1, $t2
        beq $t4, 2, main_ele_5
```

```
        # dong 31
        beq $s0, 31, convert_1
main_ele_circle_1_back:
        beq $s0, 31, main_ele_4
        # dong 32 -> 34
        slti $t1, $s0, 35
        li $t3, 31
        slt $t2, $t3, $s0
        add $t4, $t1, $t2
        beq $t4, 2, main_ele_5
        # dong 35
        beq $s0, 35, main_ele_4
        # dong 36
        beq $s0, 36, main_ele_5
        # dong 37
        beq $s0, 37, main_ele_4
        # dong 38
        beq $s0, 38, main_ele_5
        # dong 39
        beq $s0, 39, main_ele_4
        # dong 40
        beq $s0, 40, main_ele_5
        # dong 41 -> 47
        slti $t1, $s0, 48
        li $t3, 40
        slt $t2, $t3, $s0
        add $t4, $t1, $t2
        beq $t4, 2, main_ele_4
        # dong 48 -> 50
        slti $t1, $s0, 51
        li $t3, 47
        slt $t2, $t3, $s0
        add $t4, $t1, $t2
        beq $t4, 2, main_ele_3
        # dong 51
        beq $s0, 51, main_ele_1
```

Giải thích code bên trên:

- **main_ele_circle_1**: Ở đây ta dùng biến đếm dòng s0 để xác định dòng cần tô màu và chuyển đến hàm xác định điểm D1, C1 của dòng cần tô màu sau đó chuyển đến hàm **main_ele_circle_2.**

```
main_ele_circle_2:
        # dong 6
        beq $s0, 6, main_element_1
        # dong 7
        beq $s0, 7, main_element_3
        # dong 8, 9
        beq $s0, 8, main_element_2
        beq $s0, 9, main_element_2
        # dong 10
        beq $s0, 10, main_element_1
        # dong 11
        beq $s0, 11, main_element_2
        # dong 12
        beq $s0, 12, main_element_1
        # dong 13
        beq $s0, 13, main_element_0
        # dong 14
        beq $s0, 14, main_element_1
        # dong 15
        beq $s0, 15, main_element_1
        # dong 16
        beq $s0, 16, main_element_0
        # dong 17
        beq $s0, 17, main_element_1
        # dong 18
        beq $s0, 18, main_element_0
        # dong 19
        beq $s0, 19, main_element_1
        # dong 20, 21
        beq $s0, 20, main_element_0
        beq $s0, 21, main_element_0
        # dong 22
        beq $s0, 22, main_element_1
        # dong 23 -> 30
        slti $t1, $s0, 31
        li $t3, 22
        slt $t2, $t3, $s0
        add $t4, $t1, $t2
        beq $t4, 2, main_element_0

        # dong 31
        beq $s0, 31, convert_2
main_ele_circle_2_back:
        beq $s0, 31, main_element_1
        # dong 32, 33
        beq $s0, 32, main_element_0
        beq $s0, 33, main_element_0
        # dong 34
        beq $s0, 34, main_element_1
        # dong 35
```

```
189         beq $s0, 35, main_element_0
190         # dong 36
191         beq $s0, 36, main_element_1
192         # dong 37
193         beq $s0, 37, main_element_0
194         # dong 38, 39
195         beq $s0, 38, main_element_1
196         beq $s0, 39, main_element_1
197         # dong 40
198         beq $s0, 40, main_element_0
199         # dong 41, 42
200         beq $s0, 41, main_element_1
201         beq $s0, 42, main_element_1
202         # dong 43
203         beq $s0, 43, main_element_2
204         # dong 44
205         beq $s0, 44, main_element_1
206         # dong 45, 46
207         beq $s0, 45, main_element_2
208         beq $s0, 46, main_element_2
209
```

Giải thích code bên trên:

- **main_ele_circle_2**: Ở đây ta dùng biến đếm dòng s0 để xác định dòng cần tô màu và chuyển đến hàm xác định điểm D2, C2 của dòng cần tô màu. Đến đây ta đã xác định được D1, C1, D2, C2 => chuyển đến hàm tô màu.

```
210 main_color:
211         # dong 1 -> 5
212         slti $t1, $s0, 6
213         li $t3, 0
214         slt $t2, $t3, $s0
215         add $t4, $t1, $t2
216         beq $t4, 2, main_color_1        # neu thuoc dong 1 -> 5 thi nhay den nhan
217         # dong 6 -> 46
218         slti $t1, $s0, 47
219         li $t3, 5
220         slt $t2, $t3, $s0
221         add $t4, $t1, $t2
222         beq $t4, 2, main_color_2        # neu thuoc dong 6 -> 46 thi nhay den nhan
223         # dong 47 -> 51
224         slti $t1, $s0, 52
225         li $t3, 46
226         slt $t2, $t3, $s0
227         add $t4, $t1, $t2
228         beq $t4, 2, main_color_1        # neu thuoc dong 47 -> 51 thi nhay den nhan
229
```

Giải thích code bên trên:

- **main_color:** dùng s0 để xác định dòng hiện tại cần tô màu và chuyển đến hàm tô màu thích hợp

+ Từ 1-> 5 và 47 -> 51: Chỉ cần tô màu từ D1 đến C1

+ Từ 6 -> 46: Cần tô từ D1 đến D2 và C2 đến C1. Vì khoảng trống ở bên trong đường tròn không cần tô, tức là không cần tô D2 đến C2

```
main_color_1:
        # To mau tu D1 -> C1
        add $s5, $s1, $zero     # s5 = s1
        add $s6, $s2, $zero     # s6 = s2
        jal color_main
        # Sau khi to mau xong thi tang so dong len
        j main_raise

```

Giải thích code bên trên:

- **main_color-1:** Tô màu từ D1 đến C1

  => Gán D1, C1 vào 2 biến s5, s6 (2 tham số của hàm tô màu) sau đó chuyển đến hàm tô màu

```
main_color_2:
        # To mau D1 -> D2
        add $s5, $s1, $zero     # s5 = s1
        add $s6, $s3, $zero     # s6 = s3
        jal color_main
        # To mau C2 -> C1
        add $s5, $s4, $zero     # s5 = s4
        add $s6, $s2, $zero     # s6 = s2
        jal color_main
        # Sau khi to mau xong thi tang so dong len
        j main_raise

```

Giải thích code bên trên:

- **main_color-2:** Tô màu từ D1 đến D2 và từ C2 đến C1

  => Gán D1, D2 vào 2 biến s5, s6 (2 tham số của hàm tô màu) sau đó chuyển đến hàm tô màu

  => Sau đó gán C2, C1 vào 2 biến s5, s6 (2 tham số của hàm tô màu) sau đó chuyển đến hàm tô màu

```
# Ham tang so dong
main_raise:
        beq $s0, 51, main_out  # neu den dong 51 thi thoat main
        addi $s0, $s0, 1       # tang so dong
        j main_ele_circle_1

```

Giải thích code bên trên:

- **main_raise:** Kiểm tra nếu dòng vừa tô là dòng 51 thì thoát ra khỏi hàm main nếu không thì tăng số dòng và tiếp tục tô màu.

```
257 # Ham mo rong nhay den Cal* va quay lai main_ele_circle_2
258 main_ele_1:
259         jal Cal_4
260         j main_ele_circle_2
261 main_ele_3:
262         jal Cal_2
263         j main_ele_circle_2
264 main_ele_4:
265         jal Cal_1
266         j main_ele_circle_2
267 main_ele_5:
268         jal Cal_0
269         j main_ele_circle_2
```

Giải thích code bên trên:

- **main_ele_1, main_ele_3, main_ele_4, main_ele_5:** Chuyển đến các cách tính D1, C1 tương ứng, sau đó nhảy đến **main_ele_circle_2**

```
271 # Ham mo rong nhay den Calcu* va quay lai main_raise de tang so dong
272 main_element_0:
273         jal Calcu_0
274         j main_color
275 main_element_1:
276         jal Calcu_1
277         j main_color
278 main_element_2:
279         jal Calcu_2
280         j main_color
281 main_element_3:
282         jal Calcu_3
283         j main_color
284
```

Giải thích code bên trên:

- **main_element_0, main_element_1, main_element_2, main_element_3:** Chuyển đến các cách tính D2, C2 tương ứng, sau đó nhảy đến **main_color**

```
# Ham tinh D = s1, C = s2
Cal_0:
        addi $s1, $s1, 512
        addi $s2, $s2, 512
        jr $ra

# Ham tinh D = s1, C = s2
Cal_1:
        addi $s1, $s1, 511
        addi $s2, $s2, 513
        jr $ra

# Ham tinh D = s1, C = s2
Cal_2:
        addi $s1, $s1, 510
        addi $s2, $s2, 514
        jr $ra

# Ham tinh D = s1, C = s2
Cal_4:
        addi $s1, $s1, 508
        addi $s2, $s2, 516
        jr $ra
```

Giải thích code bên trên:

- **Cal_0, Cal_1, Cal_2, Cal_4:** Tính D1, C1 của dòng cần tô màu tiếp theo

```
# Ham tinh D = s3, C = s4
Calcu_0:
        addi $s3, $s3, 512
        addi $s4, $s4, 512
        jr $ra

# Ham tinh D = s3, C = s4
Calcu_1:
        addi $s3, $s3, 511
        addi $s4, $s4, 513
        jr $ra

# Ham tinh D = s3, C = s4
Calcu_2:
        addi $s3, $s3, 510
        addi $s4, $s4, 514
        jr $ra

# Ham tinh D = s3, C = s4
Calcu_3:
        addi $s3, $s3, 509
        addi $s4, $s4, 515
        jr $ra
```

Giải thích code bên trên:

- **Calcu_0, Calcu_1, Calcu_2, Calcu_3:** Tính D2, C2 của dòng cần tô màu tiếp theo

```
# Ham to mau cac pixel tu s5 -> s6
color_main:
        slt $t5, $s5, $s6       # neu s5 > s6 thi can convert s5 va s6
        beqz $t5, convert_3
color_main_back:
        add $t1, $s5, $zero     # t1 = s5
color_ele:
        add $t6, $t1, $zero
        sw $t6, 0($sp)          # Loop gan toan bo cac diem pixel vao stack
        addi $sp, $sp, -4       # Den ngan tiep theo

        mul $t3, $t1, 4         # t3 = t1 * 4
        add $t4, $k0, $t3       # t4 = k0 + t1 * 4
        li $t2, 0x00FFFF00      # t2 = YELLOW
        sw $t2, 0($t4)          # k0 = YELLOW
        beq $t1, $s6, color_out         # neu t1 = s6 thi thoat khoi ham to mau
        addi $t1, $t1, 1        # t1 = t1 + 1
        j color_ele
color_out:
        jr $ra
```

Giải thích code bên trên:

- **color_main:** bắt đầu tô màu. Tô từng điểm từ s5 đến s6
    + Kiểm tra nếu s5 > s6 thì hoán đổi s5 và s6 cho nhau
- **color_main_back:** điểm quay lại sau khi hoán đổi
- **color_ele:** Bắt đầu tô màu từng pixel và lưu các pixel đó vào trong ngăn xếp.
- **color_out:** Thoát khỏi hàm tô màu

```

# Doi vi tri tu s1 thanh s2 va nguoc lai
convert_1:
        add $t1, $s1, $zero     # t1 = s1
        add $s1, $s2, $zero     # s1 = s2
        add $s2, $t1, $zero     # s2 = t1
        j main_ele_circle_1_back


# Doi vi tri tu s3 thanh s4 va nguoc lai
convert_2:
        add $t1, $s3, $zero     # t1 = s3
        add $s3, $s4, $zero     # s3 = s4
        add $s4, $t1, $zero     # s4 = t1
        j main_ele_circle_2_back

# Doi vi tri tu s5 thanh s6 va nguoc lai
convert_3:
        add $t1, $s5, $zero     # t1 = s3
        add $s5, $s6, $zero     # s3 = s4
        add $s6, $t1, $zero     # s4 = t1
        j color_main_back

```

Giải thích code bên trên:

- **convert_1:** Hàm hoán đổi s1 và s2
- **convert_2:** Hàm hoán đổi s3 và s4
- **convert_3:** Hàm hoán đổi s5 và s6

```
# Ham thoat khoi chuong trinh main
main_out:
        add $t9, $sp, $zero    # Luu dia chi diem cuoi stack
# ------------------------------------------------------------------------------------
```

Giải thích code bên trên:

- **main_out:** Sau khi vẽ xong dòng 51 thì lưu điểm cuối của stack và đến phần di chuyển quả bóng

```

#-----------------
#GIAI PHONG BO NHO
li $s1, 0
li $s2, 0
li $s3, 0
li $s4, 0
li $s5, 0
li $s6, 0
li $t1, 0
li $t2, 0
li $t3, 0
li $t4, 0
li $t6, 0
#----------------
```

Giải thích code bên trên:

- Giải phóng bộ nhớ với mong muốn code sẽ chạy nhanh hơn.

```
# ------------------------------------------------------------------------------------
# TAC DUNG CUA CAC THANH GHI TRONG CODE TAO CO CHE DI CHUYEN CUA HINH TRON
#-------------------------------------------------------------------------------------
# s0: gia tri diem pixel lay ra tu stack
# s1: gia tri tuong ung voi che do di chuyen (W, D, S, A)
#
# s7: bien toc do
# t8: dia chi diem dau stack
# t9: dia chi diem cuoi stack
# ------------------------------------------------------------------------------------


# ------------------------------------------------------------------------------------
#      CODE TAO CO CHE DI CHUYEN CUA HINH TRON
# ------------------------------------------------------------------------------------


      li $a3, KEY_CODE       # ASCII code from keyboard, 1 byte
      li $k1, KEY_READY      # =1 if has a new keycode ?
                             # Auto clear after lw

      li $a2, DISPLAY_CODE   # ASCII code to show, 1 byte
      li $a1, DISPLAY_READY  # =1 if the display has already to do
                             # Auto clear after sw

      li $s7, 0              # bien toc do cho

```

Giải thích code bên trên:

- Nạp địa chỉ nhận ký tự từ bàn phím (KEY_CODE) vào a3
- Nạp địa chỉ kiểm tra có ký tự mới được nhập không (KEY_READY) vào k1
- Nạp địa chỉ hiển thị ký tự vào a0
- Nạp địa chỉ sẵn sàng hiển thị vào a1

```
427  loop:
428          nop
429  #------------------------------------------------------
430  WaitForKey:
431          lw $t1, 0($k1) # $t1 = [$k1] = KEY_READY
432          nop
433          beq $t1, $zero, WaitForKey # if $t1 == 0 then Polling
434          nop
435  #------------------------------------------------------
436  ReadKey:
437          lw $t0, 0($a3) # $t0 = [$k0] = KEY_CODE
438          nop
439  #------------------------------------------------------
440  WaitForDis:
441          lw $t2, 0($a1) # $t2 = [$s1] = DISPLAY_READY
442          nop
443          beq $t2, $zero, WaitForDis # if $t2 == 0 then Polling
444          nop
445  #------------------------------------------------------
446  ShowKey:
447          sw $t0, 0($a0) # show key
448          nop
```

Giải thích code bên trên:

- Vòng lặp để kiểm tra người dùng có nhập ký tự vào không

```
452
453  CheckKey:
454          beq $t0, 'w', key_W
455          beq $t0, 'd', key_D
456          beq $t0, 's', key_S
457          beq $t0, 'a', key_A
458          beq $t0, 'z', key_Z
459          beq $t0, 'x', key_X
460          bne $s1, 0, check_border
461          j WaitForKey
```

Giải thích code bên trên:

- **CheckKey:** Kiểm tra xem ký tự vừa nhập là ký tự nào. Nếu là w, s, d, a thì tiến hành di chuyển đường tròn theo quy ước và sau đó kiểm tra điều kiện viền. Nếu là z thì tăng tốc, w thì giảm tốc. Nếu là ký tự khác thì quay lại chờ ký tự mới.

```
458  # Xac dinh huong di chuyen
459  # len
460  key_W:
461          li $s1, -512
462          j convert_color
463  # trai
464  key_A:
465          li $s1, -1
466          j convert_color
467  # phai
468  key_D:
469          li $s1, 1
470          j convert_color_2
471  # xuong
472  key_S:
473          li $s1, 512
474          j convert_color_2
```

```
481  # giam toc
482  key_X:
483          addi $s7, $s7, 2
484          beq $s1, -512, convert_color
485          beq $s1, -1, convert_color
486          beq $s1, 512, convert_color_2
487          beq $s1, 1, convert_color_2
488
489  # tang toc
490  key_Z:
491          addi $s7, $s7, -2
492          bltz $s7, key_Z_ex
493          j key_Z_cont
494  key_Z_ex:
495          li $s7, 0
496  key_Z_cont:
497          beq $s1, -512, convert_color
498          beq $s1, -1, convert_color
499          beq $s1, 512, convert_color_2
500          beq $s1, 1, convert_color_2
501
```

Giải thích code bên trên:

- **Key_W, Key_A, Key_D, Key_S:** Gán giá trị thích hợp cho s1, để khi cộng s1 vào vị trí cũ sẽ ra vị trí mới.
- **Key_Z, Key_X:** So sánh s1 với các giá trị 512, -512, 1, -1 để tăng, giảm tốc.

```
476  # Dao chieu di chuyen
477  convert_color_back:
478          mul $s1, $s1, -1        # dao chieu khi va phai bien
479          beq $s1, -512, convert_color
480          beq $s1, -1, convert_color
481          beq $s1, 512, convert_color_2
482          beq $s1, 1, convert_color_2
483
```

Giải thích code bên trên:

- **Convert_color_back:** Đảo chiều di chuyển khi gặp viền, chỉ cần nhân s1 với -1 ta có thể đảo ngược chiều di chuyển.

```
483
484 # Ham to mau cac pixel tu dau stack -> cuoi stack
485 convert_color:
486         add $sp, $t8, $zero     # lay diem dau stack
487
488 convert_color_ele:
489         lw $s0, 0($sp)          # lay vi tri pixel tu ngan xep
490         # Convert pixel vang -> den
491         mul $t3, $s0, 4         # t3 = s0 * 4
492         add $t4, $k0, $t3       # t4 = k0 + s0 * 4
493         li $t2, 0x0             # t2 = DARK
494         sw $t2, 0($t4)          # k0 = DARK
495
496         # Luu vao stack
497         add $s0, $s0, $s1       # Xac dinh pixel can to mau
498         add $s6, $s0, $zero
499         sw $s6, 0($sp)          # gan lai vao ngan stack vua lay
500
501         # convert pixel den -> vang
502         mul $t3, $s0, 4         # t3 = s0 * 4
503         add $t4, $k0, $t3       # t4 = k0 + s0 * 4
504         li $t2, 0x00FFFF00      # t2 = YELLOW
505         sw $t2, 0($t4)          # k0 = YELLOW
506
507         addi $sp, $sp, -4       # den ngan nho tiep theo
508         beq $sp, $t9, check_new_key    # neu het stack thi khong to mau nua
509         j convert_color_ele     # lap de to mau pixel tiep theo
```

Giải thích code bên trên:

- **convert_color:** Bắt đầu di chuyển quả bóng bằng cách đổi màu vị trí cũ về màu nền và màu vị trí mới bằng màu vàng. Đổi màu từ đầu ngăn xếp đến cuối ngăn xếp (từ trên xuống dưới của đường tròn)
- **convert_color_ele:** Lần lượt lấy các vị trí từ ngăn xếp ra, đổi màu về màu nền, cộng thêm s1 để tạo thành vị trí mới, lưu vào ngăn xếp và tô màu vàng cho vị trí mới. Tiếp tục cho đến khi hết ngăn xếp

```
511 # Ham to mau cac pixel tu cuoi stack -> dau stack
512 convert_color_2:
513         add $sp, $t9, $zero
514         addi $sp, $sp, 4
515
516 convert_color_ele_2:
517         lw $s0, 0($sp)          # lay vi tri pixel tu ngan xep
518         # Convert pixel vang -> den
519         mul $t3, $s0, 4         # t3 = s0 * 4
520         add $t4, $k0, $t3       # t4 = k0 + s0 * 4
521         li $t2, 0x0             # t2 = DARK
522         sw $t2, 0($t4)          # k0 = DARK
523
524         # Luu vao stack
525         add $s0, $s0, $s1       # Xac dinh pixel can to mau
526         add $s6, $s0, $zero
527         sw $s6, 0($sp)          # gan lai vao ngan stack vua lay
528
529         # convert pixel den -> vang
530         mul $t3, $s0, 4         # t3 = s0 * 4
531         add $t4, $k0, $t3       # t4 = k0 + s0 * 4
532         li $t2, 0x00FFFF00      # t2 = YELLOW
533         sw $t2, 0($t4)          # k0 = YELLOW
534
535         beq $sp, $t8, check_new_key    # neu het stack thi khong to mau nua
536         addi $sp, $sp, 4        # den ngan nho tiep theo
537         j convert_color_ele_2   # lap de to mau pixel tiep theo
```

Giải thích code bên trên:

- **convert_color_2:** Giống **convert_color** nhưng là tô từ cuối ngăn xếp đến đầu ngăn xếp (Tô từ dưới lên trên của đường tròn)
- **convert_color_ele_2:** Giống như **convert_color_ele.** Lần lượt lấy các vị trí từ ngăn xếp ra, đổi màu về màu nền, cộng thêm s1 để tạo thành vị trí mới, lưu vào ngăn xếp và tô màu vàng cho vị trí mới. Tiếp tục cho đến khi hết ngăn xếp

```
539 #check new key tu keyboard
540 check_new_key:
541         lw $t1, 0($k1) # $t1 = [$k1] = KEY_READY
542         nop
543         bne $t1, $zero, ReadKey # if $t1 == 0 then Polling
544         nop
545
```

Giải thích code bên trên:

- **check_new_key:** Kiểm tra xem có ký tự mới được nhập hay không

```
# bat dau check bien
check_border:
        add $sp, $t8, $zero     # con tro sp tro vao dau stack
        lw $s0, 0($sp)          # lay vi tri pixel tu ngan xep dau tien

        jal sleep
        beq $s1, -512, check_row_top
        beq $s1, 512, check_row_bottom
        beq $s1, -1, check_col_left
        beq $s1, 1, check_col_right
        beq $s1, 0, WaitForKey
```

Giải thích code bên trên:

- **check_border:** Lấy vị trí đầu tiên trên cùng bên trái của hình tròn để kiểm tra điều kiện viền (gọi là vị trí giới hạn biên s0)
- Sau đó tùy vào giá trị s1 mà xác định viền nào cần kiểm tra (viền trên, dưới, trái hay phải)

```
# check s0 co thuoc 22 -> 483
check_row_top:
        slti $t1, $s0, 484
        li $t3, 21
        slt $t2, $t3, $s0
        add $t4, $t1, $t2
        beq $t4, 2, convert_color_back # neu thuoc khoang kiem tra thi dao chieu di chuye
        j convert_color                # neu khong thi tiep tuc to mau

# check s0 co thuoc 236054 -> 236515
check_row_bottom:
        add $t5, $s0, $zero
        li $t6, -236000
        add $t5, $t5, $t6

        slti $t1, $t5, 516
        li $t3, 53
        slt $t2, $t3, $t5
        add $t4, $t1, $t2
        beq $t4, 2, convert_color_back # neu thuoc khoang kiem tra thi dao chieu di chuye
        j convert_color_2                      # neu khong thi tiep tuc to mau
```

```
# check bien ben phai
check_col_right:
        add $t5, $s0, $zero
        addi $t5, $t5, 29
        div $t5, $t5, 512
        mfhi $t6
        bne $t6, 0, convert_color_2    # neu so du khac 0 thi chua den bien

        slti $t1, $t5, 463
        li $t3, 0
        slt $t2, $t3, $t5
        add $t4, $t1, $t2
        beq $t4, 2, convert_color_back # neu thuoc khoang kiem tra thi dao chieu di chuye
        j convert_color_2              # neu khong thi tiep tuc to mau

# check bien ben trai
check_col_left:
        add $t5, $s0, $zero
        addi $t5, $t5, 490
        div $t5, $t5, 512
        mfhi $t6
        bne $t6, 0, convert_color      # neu du khac 0 thi chua den bien

        slti $t1, $t5, 463
        li $t3, 0
        slt $t2, $t3, $t5
        add $t4, $t1, $t2
        beq $t4, 2, convert_color_back # neu thuoc khoang kiem tra thi dao chieu di chuye
        j convert_color        # neu khong thi tiep tuc to mau
```

Giải thích code:

- Với những vị trí giới hạn biên của s0 được tính sẵn trong 4 trường hợp:
    + Trên: 22 -> 483
    + Dưới: 236054 -> 236515
    + Trái: Khi cộng thêm 29 thì sẽ chia hết cho 512
    + Phải: Khi cộng thêm 490 thì sẽ chia hết cho 512

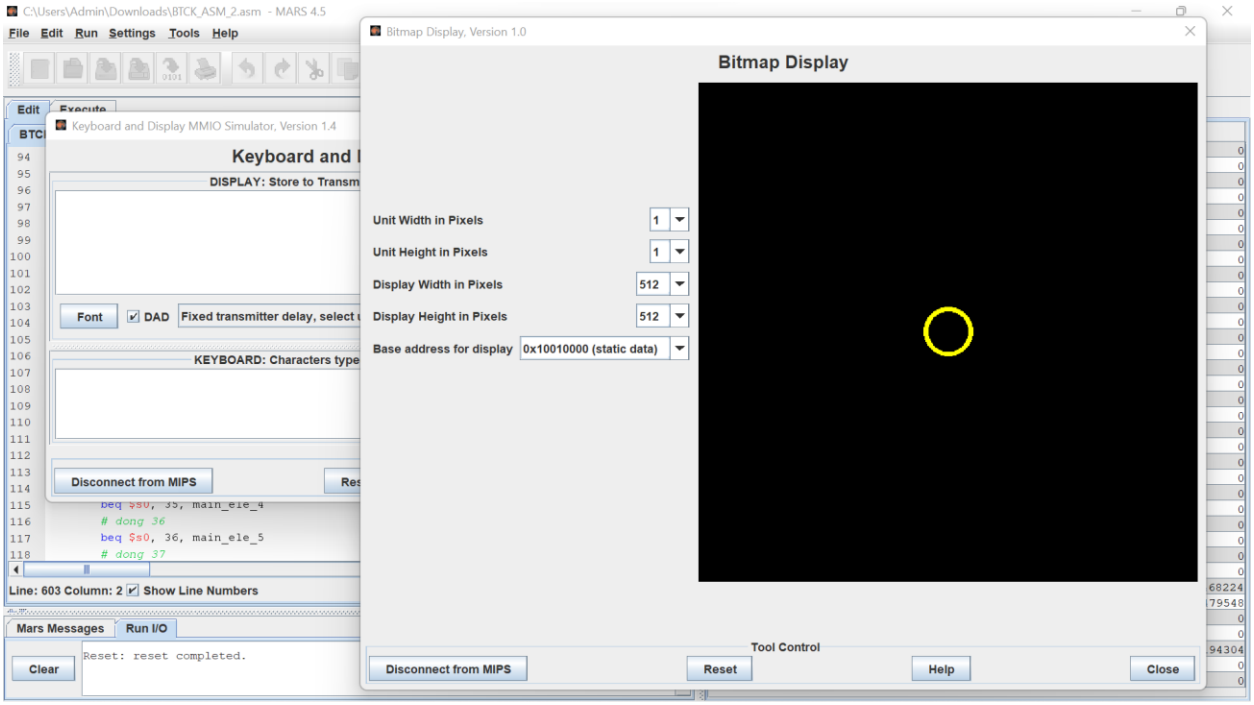
```
# sleep sau moi lan di chuyen
sleep:
        li $v0, 32
        add $a0, $s7, $0
        syscall
        jr $ra
```

Giải thích code:

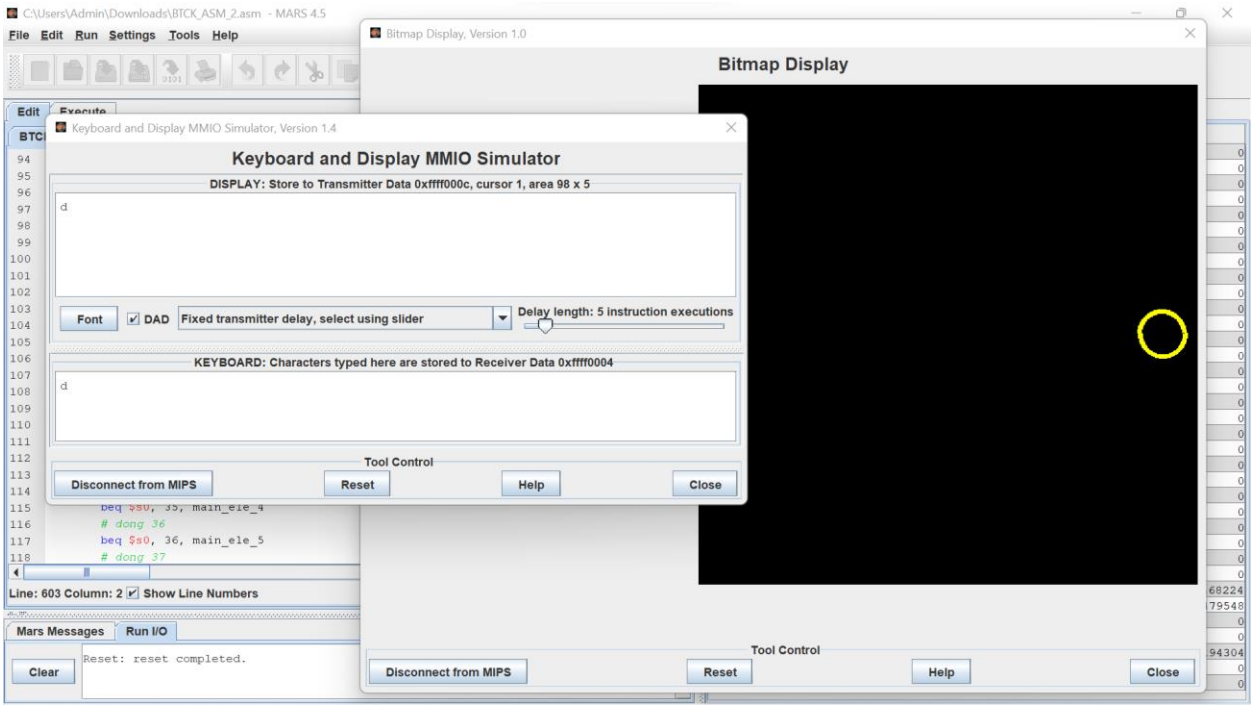
- **sleep:** phục vụ cho tăng tốc, giảm tốc.

E. Kết quả:

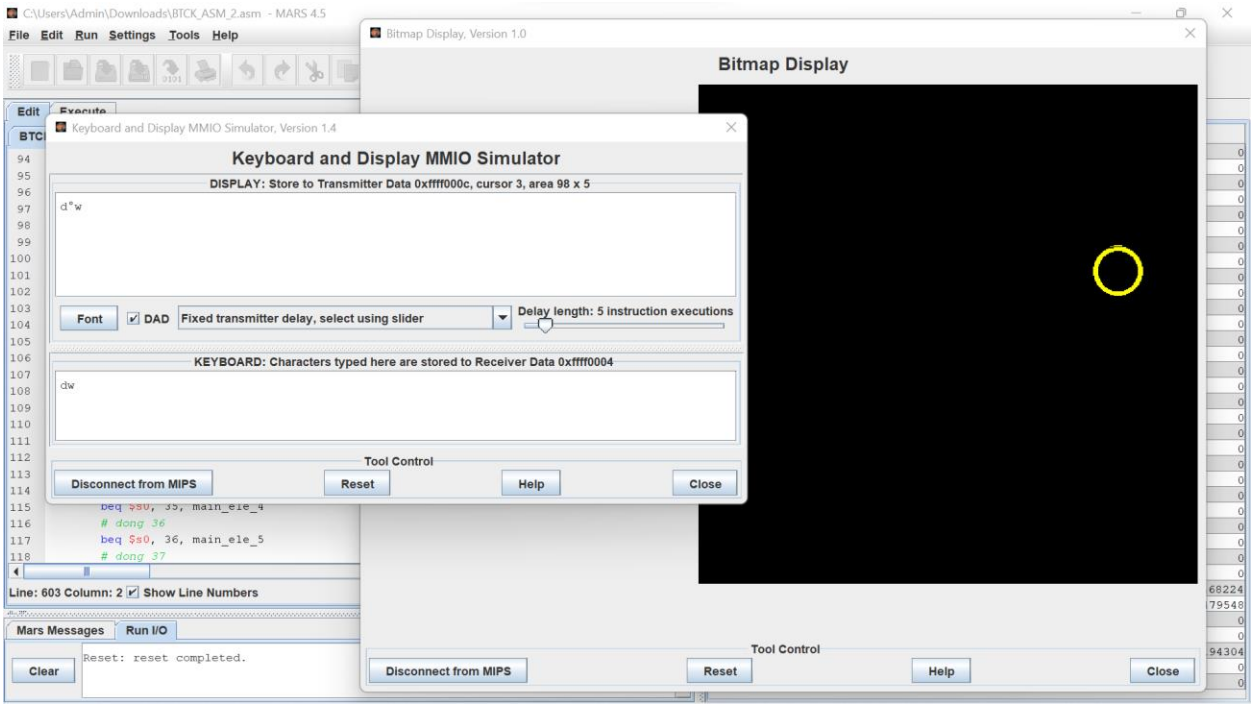
- Hình tròn ban đầu khởi tạo ở giữa màn hình:



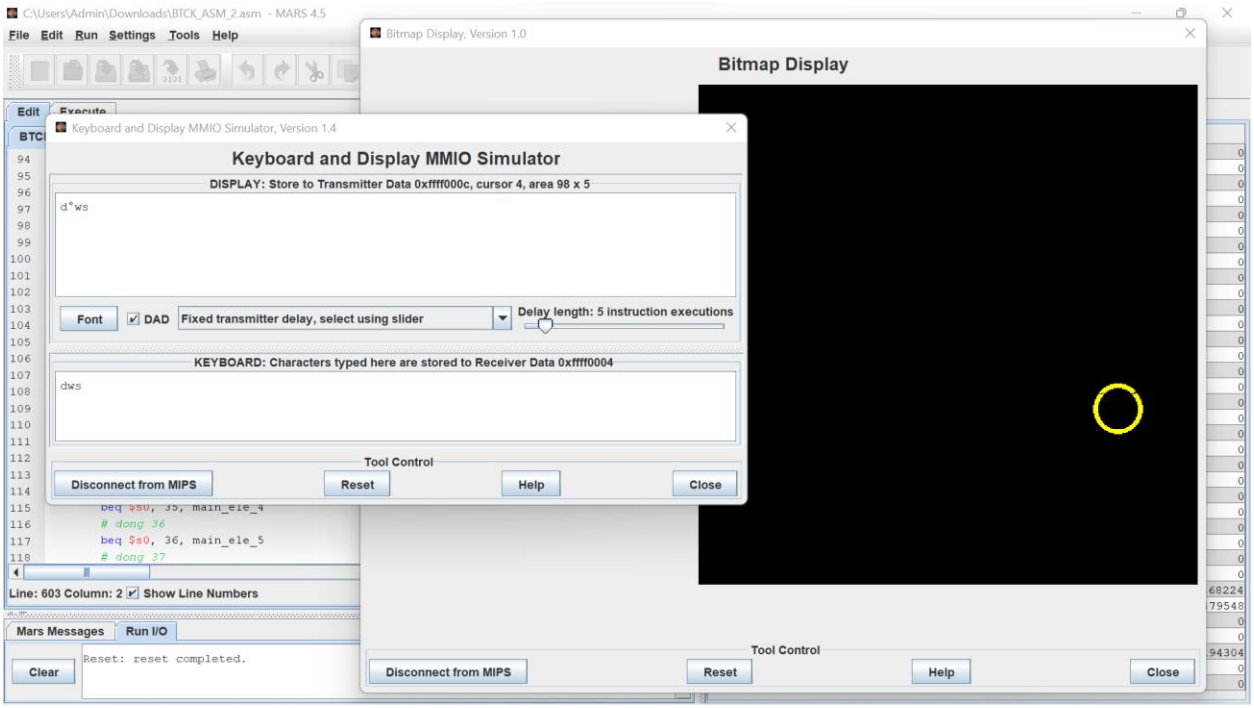
- Sang phải:



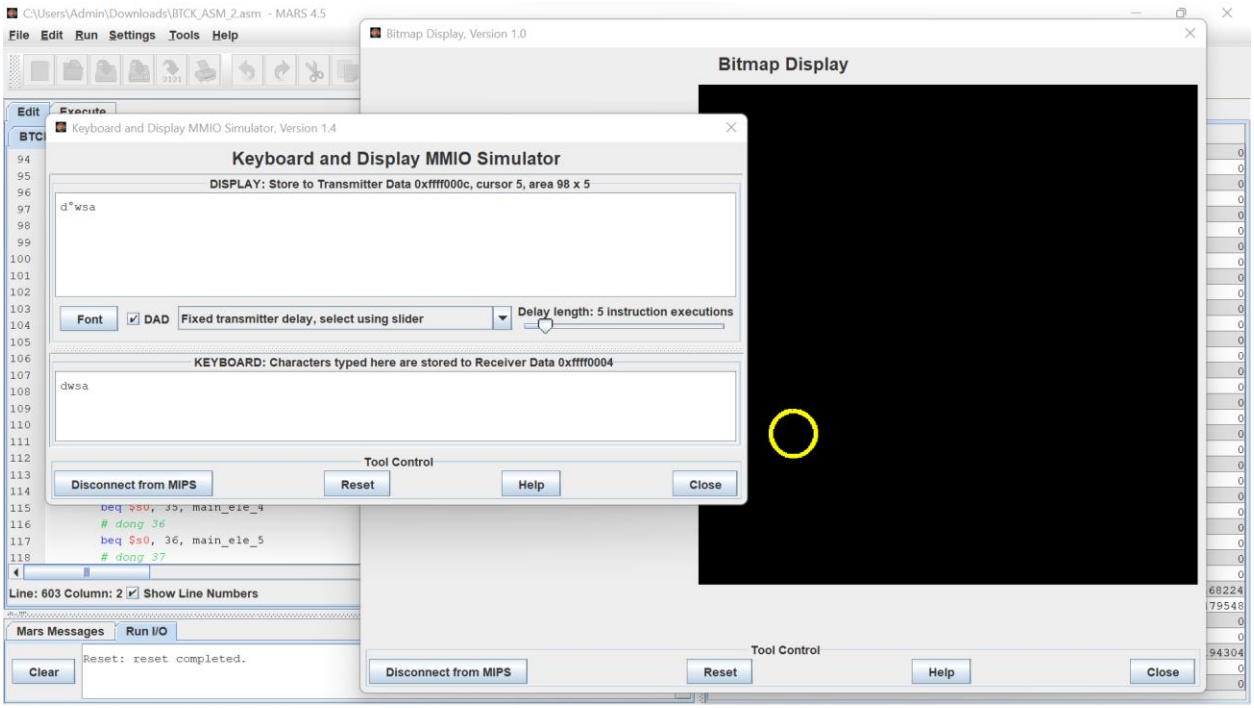
- Lên trên:


Bitmap Display
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, cursor 3, area 98 x 5
d°w
Font
DAD
Fixed transmitter delay, select using slider
Delay length: 5 instruction executions
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
dw
Tool Control
Disconnect from MIPS
Reset
Help
Close
Mars Messages
Run I/O
Clear
Reset: reset completed.

- Xuống dưới:


Bitmap Display
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, cursor 4, area 98 x 5
d°ws
Font
DAD
Fixed transmitter delay, select using slider
Delay length: 5 instruction executions
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
dws
Tool Control
Disconnect from MIPS
Reset
Help
Close
Mars Messages
Run I/O
Clear
Reset: reset completed.

- Sang trái:


C:\Users\Admin\Downloads\BTCK_ASM_2.asm - MARS 4.5
File Edit Run Settings Tools Help
Bitmap Display, Version 1.0
Bitmap Display
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, cursor 5, area 98 x 5
d°wsa
Font
DAD
Fixed transmitter delay, select using slider
Delay length: 5 instruction executions
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
dwsa
Tool Control
Disconnect from MIPS
Reset
Help
Close
beq $s0, 35, main_ele_4
dong 36
beq $s0, 36, main_ele_5
dong 37
Line: 603 Column: 2
Show Line Numbers
Mars Messages
Run I/O
Clear
Reset: reset completed.
Tool Control
Disconnect from MIPS
Reset
Help
Close

Bài 3: Kiểm tra tốc độ và độ chính xác khi gõ văn bản

Sinh viên thực hiện: Đinh Trọng Huy - 20200269

A. Cách làm và thuật toán

- Đầu tiên, sử dụng công cụ Keyboard and Display MMIO Simulator để đọc ký tự nhập vào từ bàn phím, đồng thời lặp liên tục hàm SLEEP với thời gian chạy 4ms và tăng biến đếm thời gian lên 4. Khi hàm này chạy được 250 lần thì đếm số lượng ký tự xuất hiện. Khi đó ta có thể đếm được số ký tự gõ được trong 1s (do 4ms*250=1000ms=1s)
- Cứ sau mỗi khi chạy hàm đếm số ký tự, ta cho hiển thị số ký tự đó trên đèn led (chia 10 để lấy 2 chữ số và cho chúng hiển thị ở led trái/phải)
- Khi nhập xong (KEY_READY=1) thì xử lý trap exception, ta sẽ in chuỗi, in tốc độ gõ trung bình và đếm số ký tự nhập đúng ở đây

  + In chuỗi: ta in chuỗi bằng cách quét qua tất cả các ký tự của nó và in lần lượt

  + In tốc độ gõ trung bình: lấy tổng số ký tự / biến đếm thời gian (sử dụng dấu phẩy động, chuyển từ ms->s)

  + Đếm số ký tự nhập đúng: đầu tiên, ta đếm số ký tự của chuỗi vừa nhập và chuỗi cho sẵn, chuỗi nào ít ký tự hơn thì duyệt theo chuỗi đó. Khi duyệt, ta so sánh từng vị trí và đếm số ký tự giống nhau của 2 chuỗi – chính là số ký tự nhập đúng. Sau đó hiển thị số ký tự nhập đúng lên đèn led
- Cuối cùng, hỏi người dùng có muốn quay lại chương trình hay không. Nếu có thì cho người dùng nhập tiếp và chạy lại từ bước đầu, xét với chuỗi ký tự ở lần nhập hiện tại (không phải ở chương trình trước). Nếu không thì thoát chương trình

B. Mã nguồn

```
.eqv SEVENSEG_LEFT    0xFFFF0011 # Dia chi cua den led 7 doan trai
                                 #Bit 0 = doan a
                                 #Bit 1 = doan b
                                 #Bit 7 = dau .
.eqv SEVENSEG_RIGHT   0xFFFF0010 # Dia chi cua den led 7 doan phai
```

```
.eqv KEY_READY  0xFFFF0000          # =1 if has a new keycode, auto clear
after lw
.eqv KEY_CODE   0xFFFF0004        # ascii của ký tự nhập từ bàn phím

.eqv DISPLAY_CODE   0xFFFF000C     # show ascii
.eqv DISPLAY_READY  0xFFFF0008   # =1 if the display has already to do
                              # Auto clear after sw
.eqv KEYBOARD_CAUSE   0x0000034     # Keyboard Cause

.data
bytehex     : .byte 63,6,91,79,102,109,125,7,127,111    # hiển thị led
                                          # chữ số từ 0->9
input_string : .space 1000     # lưu ký tự nhập từ bàn phím
string_origin : .asciiz "bo mon ky thuat may tinh"
Message1: .asciiz "\nSo ky tu trong 1s:  "
Message2: .asciiz "\nBan vua nhap: "
right_num: .asciiz  "\nSo ky tu nhap dung la: "
ask_return: .asciiz "\nBan co muon quay lai chuong trinh khong? "
speed1: .asciiz "\nToc do danh may trung binh: "
speed2: .asciiz " ky tu/giay\n"

.text
      li   $k0,  KEY_CODE
    li   $k1,  KEY_READY
    li   $s0, DISPLAY_CODE
```

```
    li   $s1, DISPLAY_READY
MAIN:
    li $s3, 0  # đếm số vòng lặp
    li $s4, 0  # đếm số ký tự nhập vào
    li $s5, 10   # để chia 10, lưu ở led trái
    li $s6, 250   # lưu số vòng lặp, chính là đơn vị thời gian để đo
(4*250=1000ms=1s)
    li $t4, 0     # đếm số ký tự nhập dc trong 1 khoảng thời gian
    li $t5, 0
    li $t6, 0    # đếm tổng thời gian
LOOP:
WAIT_KEY:
    lw $t1, 0($k1)   # $t1 = KEY_READY
    beqz $t1, KT_ky_tu      # $t1==1 then pooling
DEM:
    addi $t4,$t4,1                #tang bien dem ky tu nhap duoc trong 1s len 1
    teqi $t1, 1                   # nếu $t1 = 1 thì trap
KT_ky_tu:   # lặp 250 vòng xong thì xử lý ký tự
    addi $s3, $s3, 1     # tăng số vòng lặp
    div $s3, $s6
    mfhi $t7       # chia số vòng lặp cho 250, nếu dư=0 thì là được 1 vòng
    bnez $t7, SLEEP   # nếu chưa được 1 vòng thì sleep
# Nếu đã được 1 vòng thì in ra màn hình
PRINT_COUNT:
    li $s3, 0   # thiết lập lại cho lần đếm tiếp theo
    la $a0, Message1
```

```
    li $v0, 4   # in ra message
    syscall

    li $v0, 1   # in ra số ký tự trong 1 chu kỳ
    add $a0, $t4, $zero
    syscall
LED_DISPLAY:
    div $t4, $s5    # số ký tự nhập trong 1 chu kỳ chia 10
    mflo $t7        # lấy phần nguyên (để hiển thị ở led trái)
    la $s2, bytehex   # lấy mảng lưu giá trị từng chữ số đèn led
    add $s2, $s2, $t7    # lấy chữ số cần hiển thị
    lb $a0,0($s2)
    jal   SHOW_7SEG_LEFT    # hiện thị phần nguyên ở led trái

    mfhi $t7          # lấy phần dư (để hiển thị ở led phải)
    la $s2, bytehex   # lấy mảng lưu giá trị từng chữ số đèn led
    add $s2, $s2, $t7    # lấy chữ số cần hiển thị
    lb $a0,0($s2)
    jal   SHOW_7SEG_RIGHT    # hiện thị phần dư ở led phải

    li $t4, 0   # reset về 0 để bắt đầu chu kỳ mới
    beq $t5, 1, ASK_CONTINUE

SLEEP:
    addi $t6, $t6, 4
```

```
    addi $v0, $zero, 32
    li $a0, 4               # sleep 4 ms
    syscall
    nop
    b LOOP                  # trở lại Loop
END_MAIN:
    li $v0, 10
    syscall

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

# Xử lý trap
.ktext 0x80000180   # mips exception vector
    mfc0 $t1, $13   # examine cause register
    li $t2, KEYBOARD_CAUSE
    and   $at, $t1,$t2
    beq   $at,$t2, READ_KEYBOARD
    j    END_PROCESS
```

```
#COUNTER_KEYBOARD:
READ_KEYBOARD:  lb   $t0, 0($k0)          # $t0 = KEY_CODE
DISPLAY_WAIT:
    lw   $t2, 0($s1)              # $t2 = DISPLAY_READY
    beq  $t2, $zero, DISPLAY_WAIT
LOAD_KEY:
    sb $t0, 0($s0)                # load ký tự vừa nhập từ bàn phím
      la  $t7, input_string                    # $s7 là địa chỉ chuỗi nhập vào
      add $t7, $t7, $s4
      sb $t0, 0($t7)
      addi $s4, $s4, 1
      beq $t0, 10, END          # đến "\n" thì kết thúc, bắt đầu so sánh
END_PROCESS:
# Trap handler in the standard MIPS32 kernel text segment
TRAP_HANDLER:
    mfc0 $at,$14   # Coprocessor 0 register $14 has address of trapping
instruction
    addi $at,$at,4 # Add 4 to point to next instruction
    mtc0 $at,$14   # Store new address back into $14
    eret          # Error return; set PC to value in $14
END:
SPEED_COUNT:  # đếm tốc độ gõ phím
    mtc1 $t6, $f1    # f1 là tổng thời gian gõ
    cvt.s.w $f1, $f1
```

```
    mtc1 $s5, $f3    # f3 = 10
    cvt.s.w $f3, $f3

    mtc1 $s4, $f2    # f2 là tổng số ký tự
    cvt.s.w $f2, $f2

    # đổi từ ms -> s
    div.s $f1, $f1, $f3
    div.s $f1, $f1, $f3
    div.s $f1, $f1, $f3

    div.s $f2, $f2, $f1  # tổng ký tự / tổng thời gian gõ
COMPARE_LENGTH:
    li $v0, 11
    li $a0, '\n'    # xuống dòng
    syscall
    li $t1, 0       # đếm số ký tự được xét
    li $t3, 0       # đếm số ký tự nhập đúng
    li $t8, 24      # độ dài của string_origin

    slt $t7,$s4,$t8                    # so sánh độ dài xâu nhập từ bàn phím và
xâu ban đầu
                                       # xâu nào ngắn hơn thì duyệt theo xâu đó
    li $v0, 4
    la $a0, Message2
    syscall
```

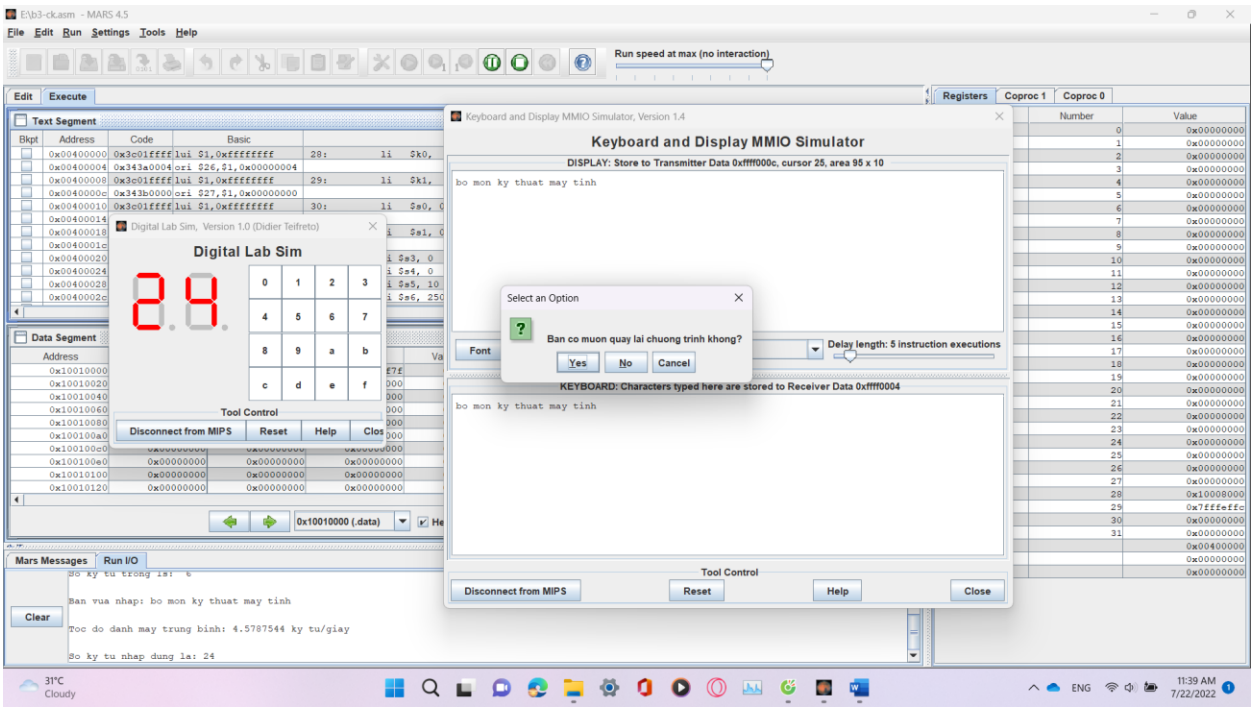
```
    bne $t7,1, PRINT_INPUT        # nếu $s4>$t8 thì check theo $t8
    add $t8, $zero, $s4           # nếu không thì xét theo $s4
    addi $t8, $t8, -1                    # trừ 1 vì không xét '\n'

PRINT_INPUT:   # in ra string nhập từ bàn phím
    la $t2, input_string
    add $t2, $t2, $t1
    lb $t9, 0($t2)

      li $v0, 11
      move $a0, $t9
      syscall

      addi $t1, $t1, 1
      bge $t1, $s4, PRINT_SPEED
      j PRINT_INPUT
PRINT_SPEED:
      # in tốc độ gõ phím trung bình
      li $v0,4
    la $a0,speed1
    syscall
    li $v0, 2
    mov.s $f12, $f2
    syscall
```

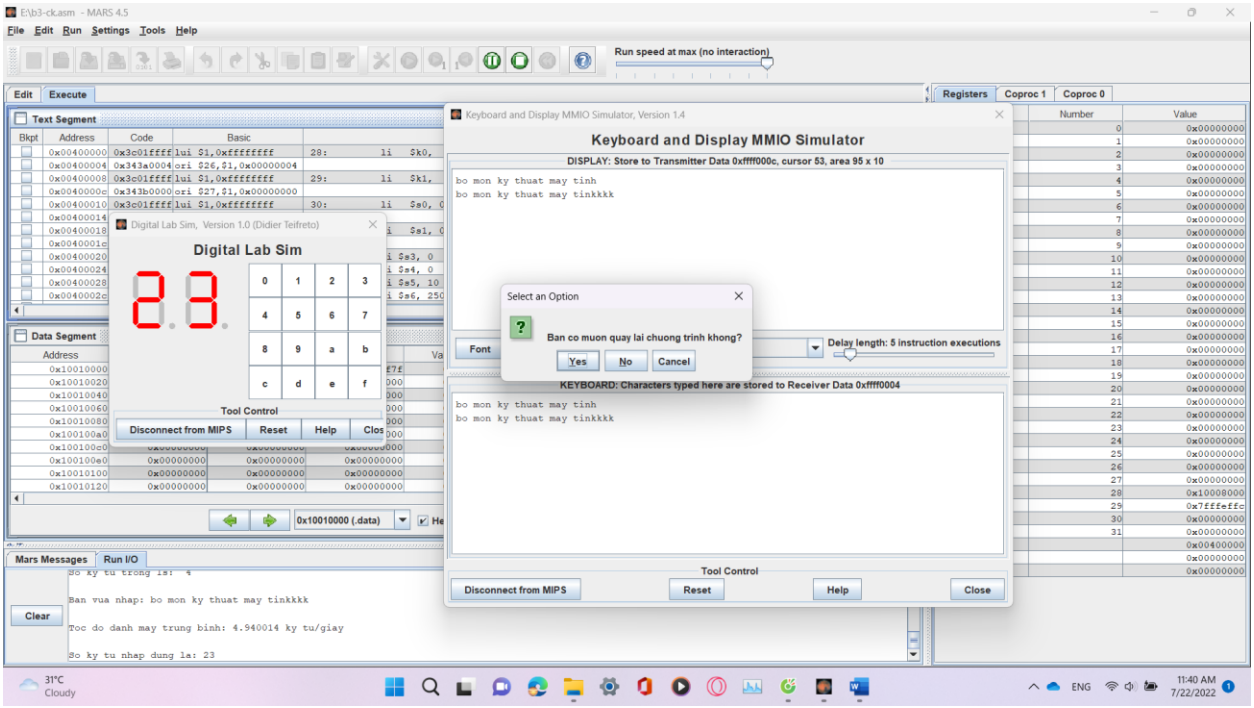
```
    li $v0,4
    la $a0,speed2
    syscall

RESET_1:
    li $t1, 0
CHECK_STRING:
    la $t2, input_string
    add $t2,$t2,$t1
    lb $t9, 0($t2)                  # lấy ký tự thứ $t1 trong input_string lưu vào $t9 để so sánh với ký tự thứ $t1 ở string_origin

    la $s7, string_origin
    add $s7, $s7, $t1
    lb $t4, 0($s7)              # lưu ký tự thứ $t1 trong string_origin lưu vào $t4

    bne $t4, $t9, CONTINUE      # nếu khác nhau thì vào CONTINUE, giống thì tăng $t3 rồi vào CONTINUE
    addi $t3, $t3, 1
CONTINUE:
    addi $t1, $t1, 1
    beq $t1, $t8, PRINT_RIGHT       # nếu duyệt hết thì print
    j CHECK_STRING

PRINT_RIGHT:
    li $v0, 4
```

```
    la $a0, right_num
    syscall

    li $v0, 1
    add $a0, $t3, $zero   # in số ký tự đúng
    syscall

    li $t5, 1
    li $t4, 0
    add $t4, $t3, $zero
    b LED_DISPLAY
ASK_CONTINUE:
    li $v0, 50
    la $a0, ask_return
    syscall
    beq $a0, 0, MAIN
    b EXIT
EXIT:
    li $v0, 10
    syscall
```

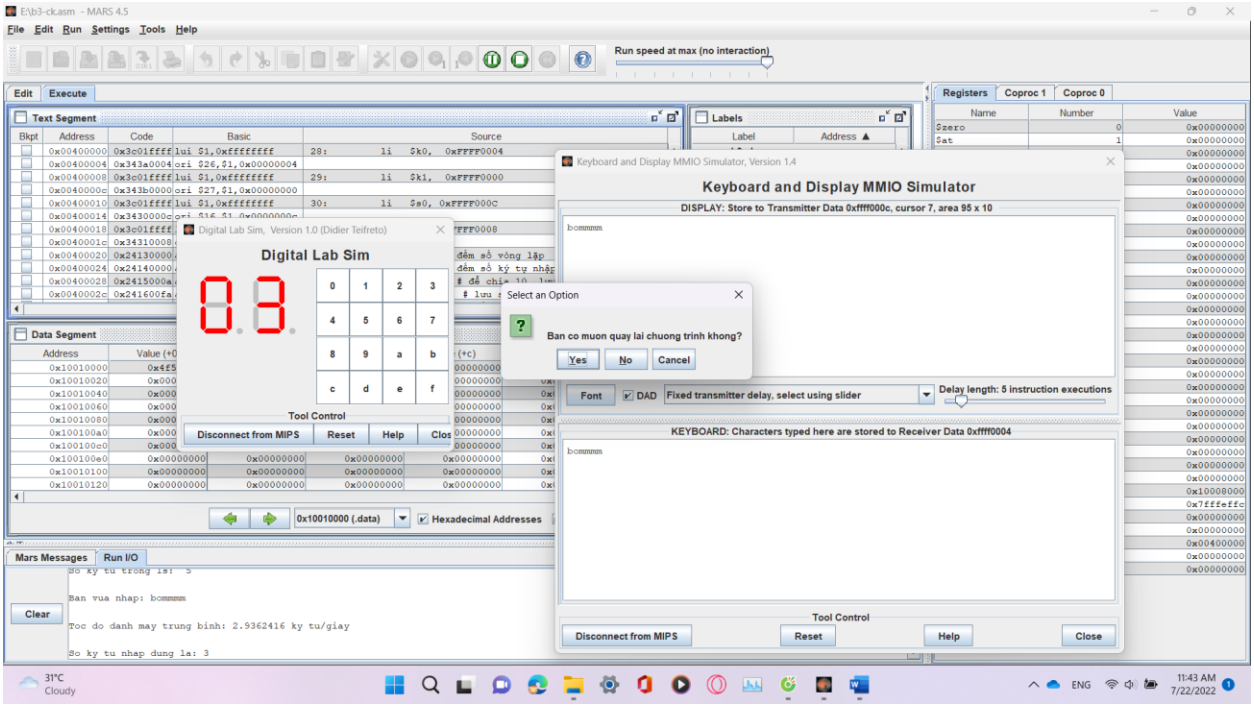
C. Kết quả chạy thử

- Lần 1: nhập "bo mon ky thuat may tinh" => trả về 24 ký tự đúng. Chọn "Yes" và nhập tiếp "bo mon ky thuat may tinkkkk" => trả về 23 ký tự đúng. Chọn "No" để dừng chương trình





- Lần 2: Nhập “bommmm” => trả về 3 ký tự đúng. Chọn “No” để dừng chương trình



- Lần 3: Nhập “hhghh7834;;k”=> trả về 0 ký tự đúng. Chọn “No” để dừng chương trình.