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

--○🕮○--

BÁO CÁO

FINAL PROJECT

GV hướng dẫn: ThS. Lê Bá Vui

Nhóm sinh viên thực hiện:

Họ và tên	MSSV	Mã lớp
Nguyễn Thành Duy	20235696	156793
Đàm Vĩnh Hưng	20235736	156793

Hà Nội, ngày 26 tháng 5 năm 2025

I. Phân công công việc

a. Phân chia đề bài

- Project 7: Nguyễn Thành Duy phụ trách
- Project 16: Đàm Vĩnh Hưng phục trách

b. Trao đổi

- Sau khi cá nhân hoàn thành đề bài của mình, trao đổi phương pháp, ý tưởng, thuật toán cho người còn lại

c. Viết báo cáo

- Thiết kế bìa và trình bày nội dung project 7: Duy
- Căn chỉnh và trình bày nội dung project 16: Hưng

II. Project 7: Vẽ hình bằng kí tự ASCII

Đề bài:

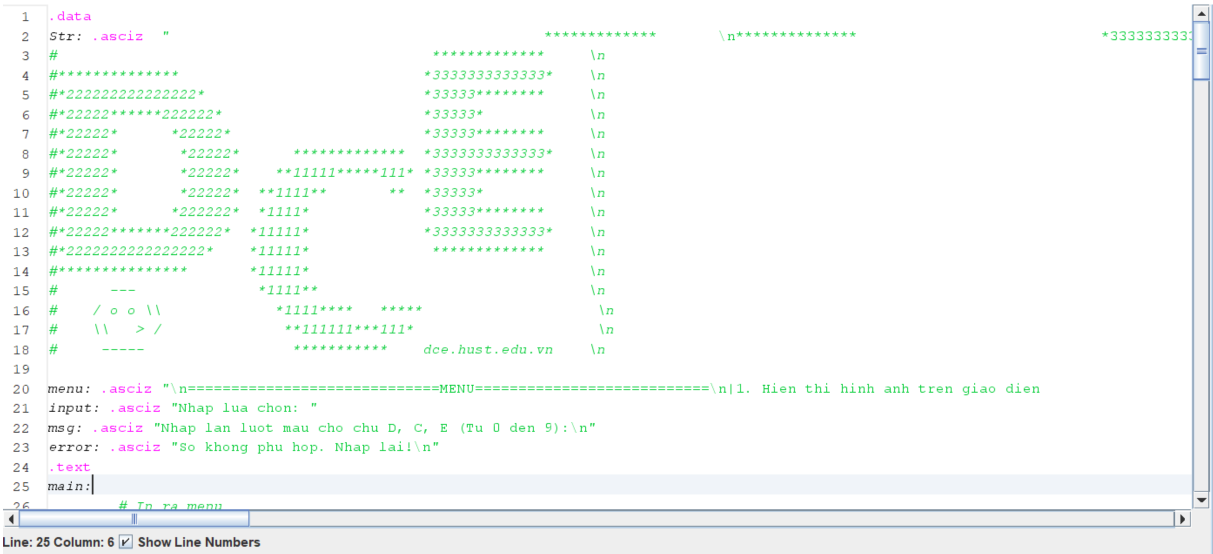
Cho hình ảnh đã được chuyển thành các kí tự ASCII như hình vẽ. Đây là hình của chữ DCE có viền * và màu là các con số.

- Hãy hiển thị hình ảnh trên lên giao diện console (hoặc giao diện Display trong công cụ giả lập Keyboard and Display MMIO Simulator)
- Hãy sửa ảnh để các chữ cái DCE chỉ còn lại viền, không còn màu số ở giữa, và hiển thị
- Hãy sửa ảnh để hoán đổi vị trí của các chữ, thành ECD, và hiển thị. Để đơn giản, các hoạ tiết đính kèm cũng được phép di chuyển theo.
- Hãy nhập từ bàn phím kí tự màu cho chữ D, C, E, rồi hiển thị ảnh trên với màu mới.

Chú ý: ngoài vùng nhớ lớn chứa ảnh được chứa sẵn trong code, không được tạo thêm vùng nhớ mới để chứa ảnh hiệu chỉnh.

1. Phân tích cách làm

- Để dễ dàng thao tác ta coi hình vẽ giống như một mảng 2 chiều với kích cỡ 64x16 nhưng được trình bày thành một mảng 1 chiều có kích cỡ 1024 kí tự (64 * 16) lưu vào trong chương trình kiểu xâu kí tự.
- Ta chia mảng ra thành 3 phần D, C, E tương ứng 21 ký tự và ký tự “\n”.



```
.data
Str: .asciz  "                                             *************       \n*************                                           *333333333
#                                 *************     \n
#**************                  *3333333333333*    \n
#*222222222222222*               *33333********     \n
#*22222******222222*             *33333*            \n
#*22222*      *22222*            *33333********     \n
#*22222*       *22222*      ************* *3333333333333*    \n
#*22222*       *22222*    **11111*****111* *33333********     \n
#*22222*       *22222*  **1111**       **  *33333*            \n
#*22222*      *222222*  *1111*             *33333********     \n
#*22222*******222222*  *11111*             *3333333333333*    \n
#*2222222222222222*    *11111*              *************     \n
#***************       *11111*                                \n
#      ---              *1111**                               \n
#    / o o \\             *1111****   *****                    \n
#    \\   > /              **111111***111*                     \n
#     -----                ***********    dce.hust.edu.vn     \n

menu: .asciz "\n============================MENU==========================\n|1. Hien thi hinh anh tren giao dien
input: .asciz "Nhap lua chon: "
msg: .asciz "Nhap lan luot mau cho chu D, C, E (Tu 0 den 9):\n"
error: .asciz "So khong phu hop. Nhap lai!\n"
.text
main:
        # In ra menu
```

Line: 25 Column: 6 Show Line Numbers

Lưu ý: Để hiển thị kí tự ‘\’ trong rars cần viết là ‘\\’

- Xây dựng giao diện menu để chọn chức năng cần hiển thị

menu: .asciz "\n============================MENU============================\n|1. Hien thi hinh anh tren giao dien |\n|2. Hien thi hinh anh chi con lai vien, khong co mau o giua|\n|3. Hien thi hinh anh sau khi hoan doi vi tri |\n|4. Nhap tu ban phim ki tu mau cho chu D, C, E roi hien thi|\n|5. Thoat |\n===\n"

- Với từng chức năng có phần xử lý riêng. Tạo vòng lặp duyệt từng chức năng để bắt được chức năng người dùng nhập đến khi người dùng nhập chức năng thoát.
- Khi người dùng nhập sai thì đưa ra thống báo, yêu cầu người dùng nhập lại

2. Thuật toán

2.1 Chức năng 1

- **Yêu cầu:** Hiển thị hình vẽ lên trên màn hình console.
- Khi người dùng nhập 1:

➔Chương trình nhảy tới nhãn ex1 để thực hiện

- Vì đã lưu hình vẽ vào xâu kí tự từ trước nên chỉ cần in xâu kí tự đó ra

```
67  #____________________Chuc nang 1_____________________________
68  ex1:
69          li a7, 4
70          la a0, Str # Lay dia chi xau Str
71          ecall
72          j main
```

2.2 Chức năng 2

- **Yêu cầu:** Sửa ảnh để các chữ cái DCE chỉ còn lại viền, không có màu số ở giữa, và hiển thị.
- Khi người dùng nhập 2:

➔Chương trình nhảy tới nhãn ex2 để thực hiện

```
73  #____________________Chuc nang 2_____________________
74  ex2:
75          li t0, 0 # Bien dem i = 0
76          la s0, Str # Lay dia chi xau Str
77          li t1, 1023 # Do dai xau
78          li s1, '0'
79          li s2, '9'
80  loop_ex2:
81          beq t0, t1, main
82          lb t2, 0(s0)
83
84          bgt t2, s2, print_ex2
85          bge t2, s1, number
86          j print_ex2
87  number:
88          addi t2, zero, 32 # Ky tu khoang trang co ma Ascii 32
89  print_ex2:
90          li a7, 11
91          mv a0, t2
92          ecall
93          addi t0, t0, 1 # i = i + 1
94          addi s0, s0, 1 # Dich chuyen sang ky tu tiep theo
95          j loop_ex2
```

- Vì các màu ở giữa là số nên ta duyệt qua từng kí tự trong xâu lưu hình vẽ và in ra màn hình tuy nhiên nếu ký tự là ‘0’ -> ‘9’ ta in ký tự khoảng trắng (‘ ‘) ra màn hình.

2.3 Chức năng 3

- Yêu cầu: Sửa ảnh để hoán đổi vị trí các chữ, thành ECD, và hiển thị. Các họa tiết đính kèm cũng được di chuyển theo.
- Khi người dùng nhập 3:

➔Chương trình nhảy tới nhãn ex3 để thực hiện

Với mỗi dòng in 21 kí tự phần của E -> 21 kí tự phần của C -> 21 kí tự phần của D -> kí tự '\n'

```
97  #____________________Chuc nang 3_____________________
98  ex3:
99          la s5, Str # Lay dia chi xau Str
100         li s6, -1
101         li s7, 16 # 16 dong
102         li s8, '\n'
103 loop_ex3:
104         addi s6, s6, 1
105         beq s6, s7, main # Duyet het 16 dong quay tro lai Menu
106         slli s1, s6, 6
107         add s1, s1, s5
108
109         addi s3, s1, 42 # Chu E bat dau tu ki tu 42 tren dong
110         jal print_21
111         addi s3, s1, 21 # Chu C bat dau tu ki tu 21 tren dong
112         jal print_21
113         addi s3, s1, 0 # Chu D bat dau tu dau dong
114         jal print_21
115
116 # In ky tu xuong dong
117         li a7, 11
118         mv a0, s8
119         ecall
120
121         j loop_ex3
122 # In 21 ky tu tren dong
123 print_21:
124         li s4, 0
125         li a5, 21
126 loop_2_ex3:
127         lb t2, 0(s3) # Luu gia tri phan tu dang xet tren Str
128         li a7, 11
129         mv a0, t2
130         ecall
131         addi s4, s4, 1 # s4 = s4 + 1
132         addi s3, s3, 1 # Chuyen sang ky tu tiep theo
133         bne s4, a5, loop_2_ex3
134         jr ra
135 j main
```

2.4 Chức năng 4

- Yêu cầu: Nhập từ bàn phím kí tự màu cho chữ D, C, E rồi hiển thị ảnh với màu mới.
- Khi người dùng nhập 4:

➔ Chương trình nhảy tới nhãn ex4 để thực hiện

```
#_____________________Chuc nang 4_____________________
ex4:
        li t0, 9
# In ra message: Nhap mau cho chu D, C, E
        li a7, 4
        la a0, msg
        ecall
# Nhap mau cho D: a1
        li a7, 5
        ecall
        bltz a0, ERROR
        bgt a0, t0, ERROR
        addi a1, a0, 48
# Nhap mau cho C: a2
        li a7, 5
        ecall
        bltz a0, ERROR
        bgt a0, t0, ERROR
        addi a2, a0, 48
# Nhap mau cho E: a3
        li a7, 5
        ecall
        bltz a0, ERROR
        bgt a0, t0, ERROR
        addi a3, a0, 48
# Mau goc: s5, s6, s7
        li s5, 50
        li s6, 49
        li s7, 51
        j next
ERROR:
        li a7, 4
      la a0, error
      ecall
      j ex4
next:
#_____________________________________________
        li t0, -1
        la s0, Str
        li t1, 1023
loop_ex4:
        beq t0, t1, main
        lb t2, 0(s0)

        beq t2, s5, print_1
        beq t2, s6, print_2
        beq t2, s7, print_3
        li a7, 11
        mv a0, t2
        ecall
```

```
195         j continue
196 print_1:
197         li a7, 11
198         mv a0, a1
199         ecall
200         j continue
201 print_2:
202         li a7, 11
203         mv a0, a2
204         ecall
205         j continue
206 print_3:
207         li a7, 11
208         mv a0, a3
209         ecall
210         j continue
211 continue:
212         addi t0, t0, 1
213         addi s0, s0, 1
214         j loop_ex4
215 j main
```

- Lần lượt nhập các chữ số 0->9 để thu được màu cho 3 chữ D, C, E cộng với 48 để thu được giá trị ASCII của chữ số đó.
- Nếu nhập sai yêu cầu nhập lại từ đầu
- Duyệt trên từng dòng và in lần lượt:
  21 kí tự D -> 21 kí tự C -> 21 kí tự E -> kí tự '\n'.
- Khi duyệt từng 21 kí tự của các chữ nếu gặp màu cũ (s5, s6, s7) của số thì thay tương ứng giá trị đó với màu mới vừa được nhập vào (a1, a2, a3)

2.5 Chức năng 5

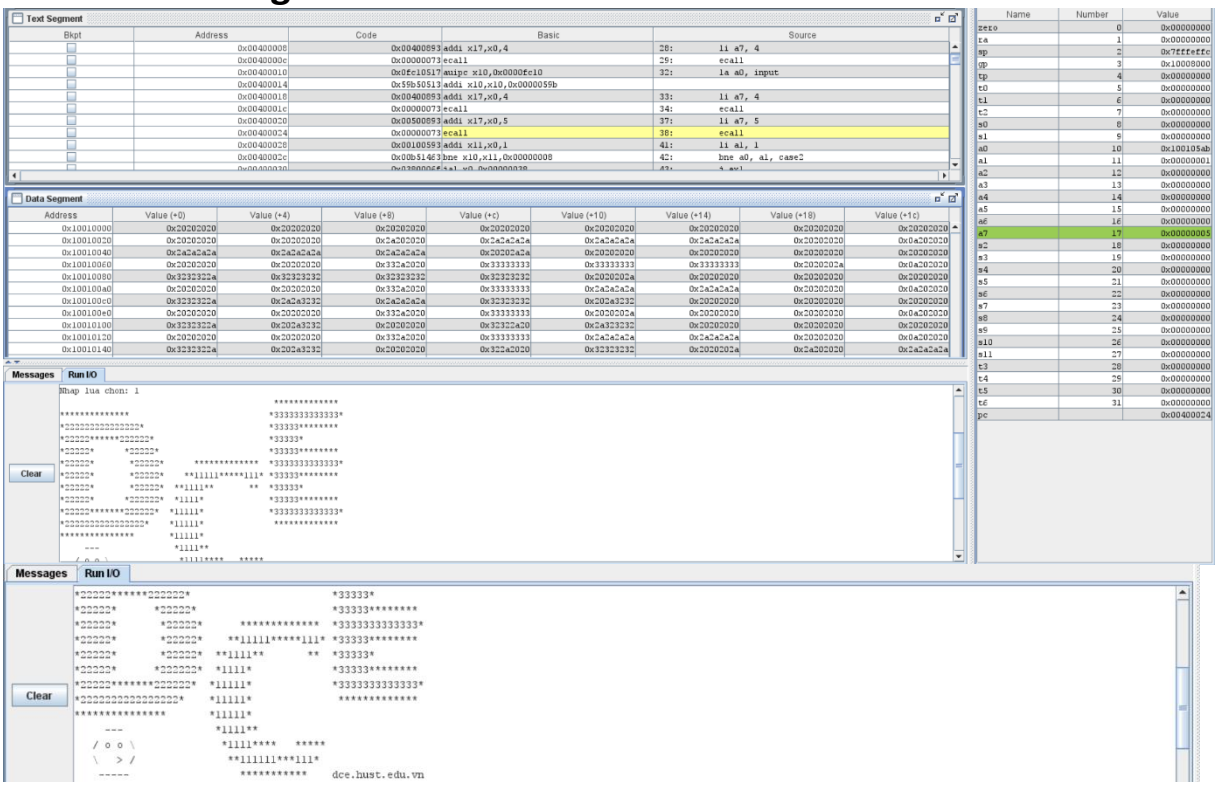
- Kết thúc đoạn chương trình

```
216 #____________________Chuc nang 5_____________________
217 ex5:
218 li a7, 10
219 ecall
220 j main
221
```

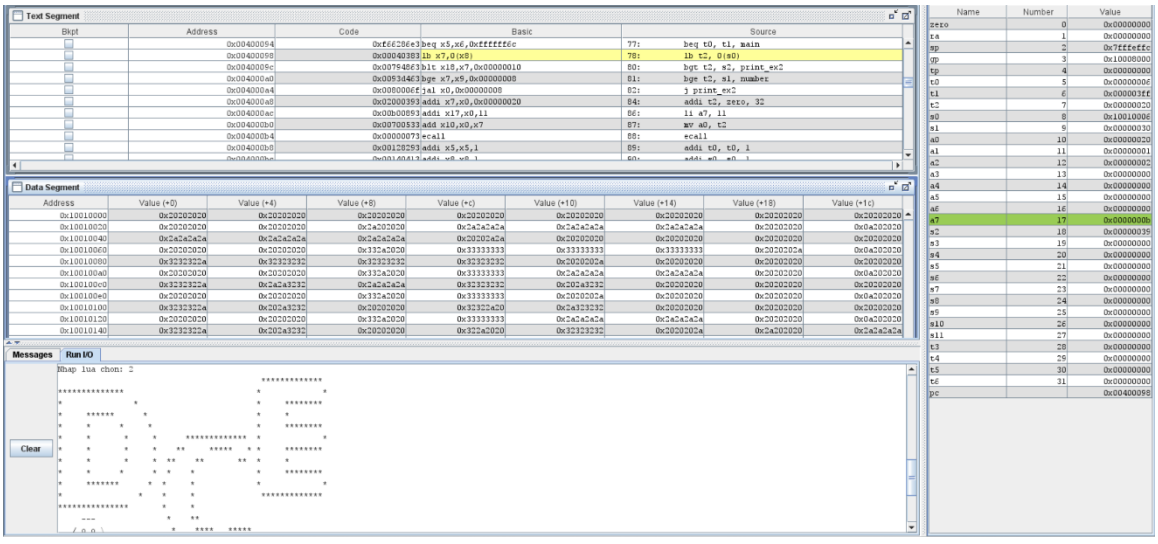
3. Mã nguồn

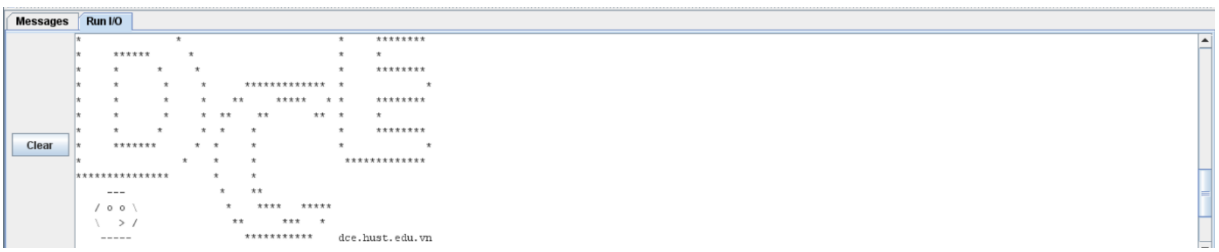
4. Kết quả

4.1 Chương trình 1

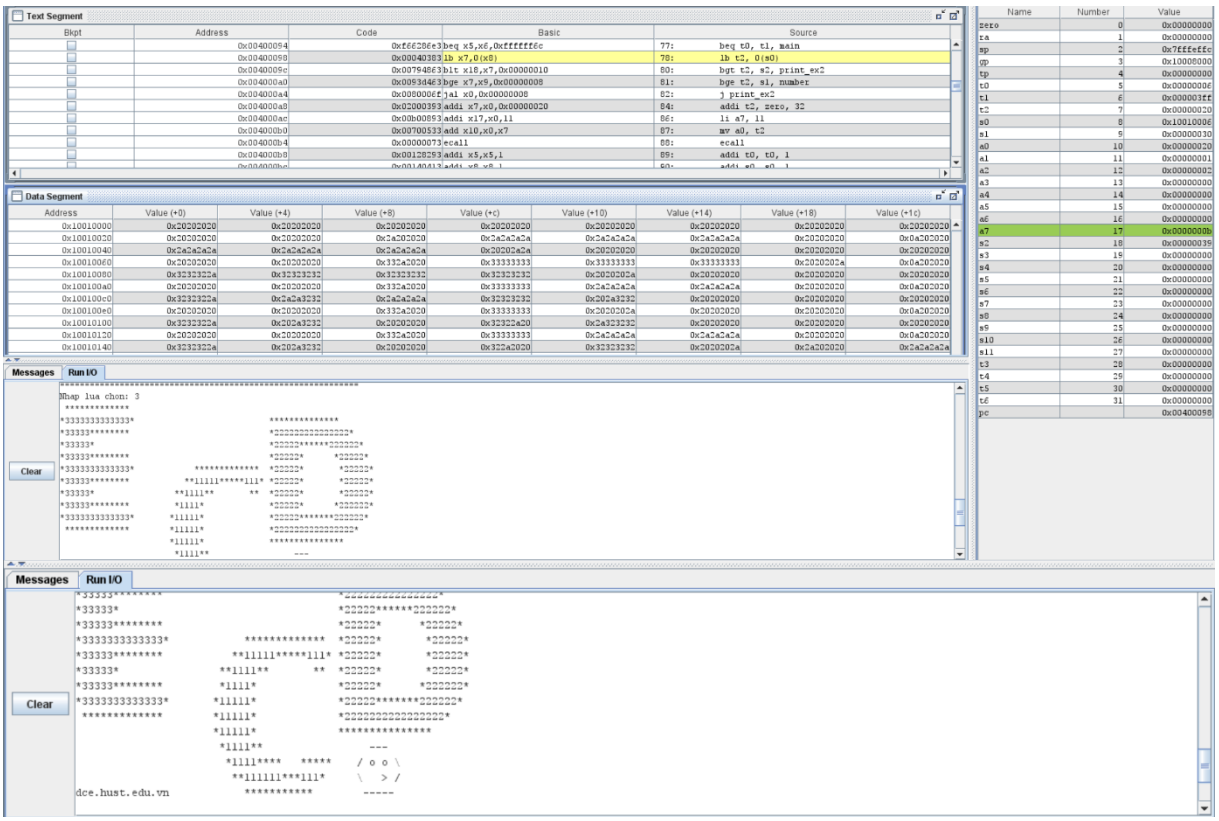


4.2 Chương trình 2

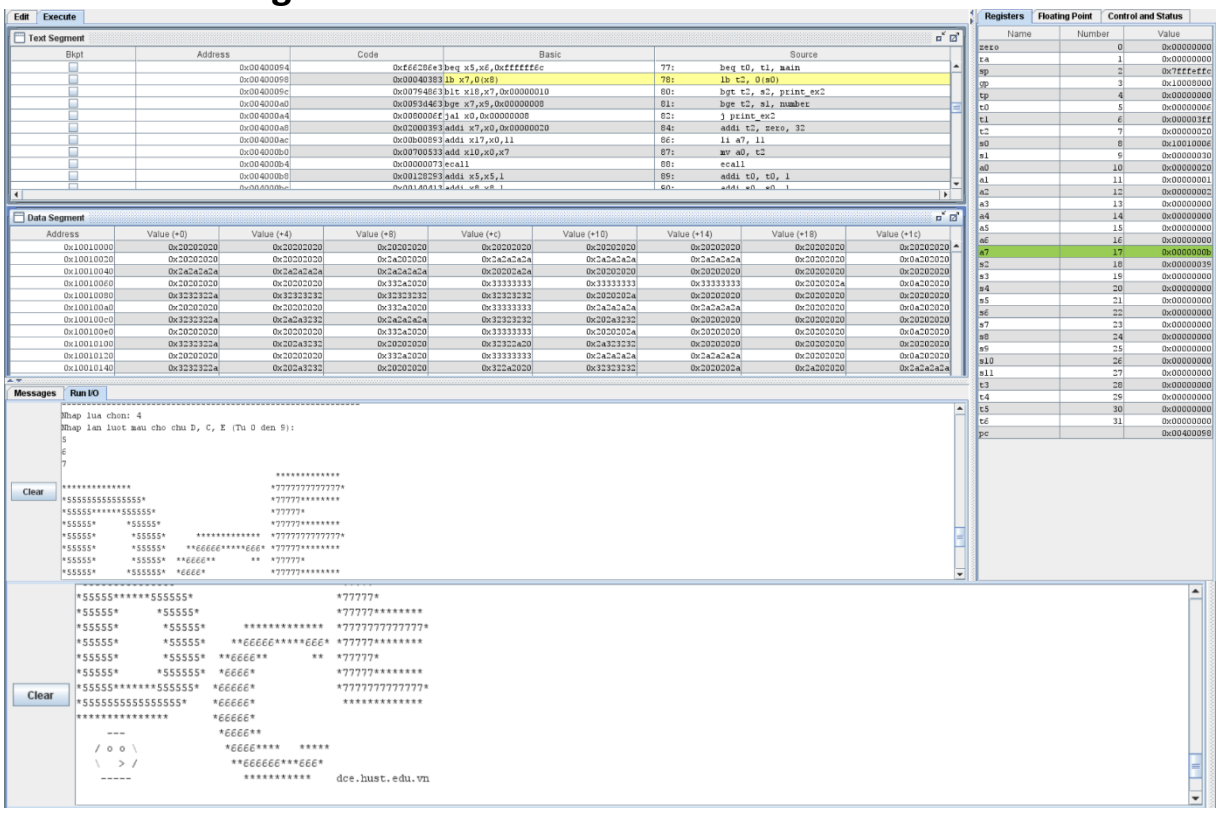




4.3 Chương trình 3



4.4 Chương trình 4



III. Project 7: Game Tic Tac Toe

16. Game Tic Tac Toe

+ Tham khảo Tic-Tac-Toe - Play retro Tic-Tac-Toe online for free
+ Lập trình game cho 2 người chơi với kích thước 4x4
+ Thiết lập kích thước Bitmap Display là 512x512
+ Vẽ lưới 4x4
+ Người chơi sử dụng bàn phím 4x4 để chọn vị trí
+ Vẽ X hoặc O vào vị trí người chơi đã chọn
+ Xác định trạng thái kết thúc ván chơi

1. PhÂN TÍCH CÁCH LÀM

1.1. Mô tả tổng quan

- Môi trường thực thi: Digital Lab Sim cung cấp bàn phím hexa 4x4 (16 phím, ánh xạ trực tiếp đến 16 ô của bàn cờ 4x4) và màn hình bitmap (512x512 pixel, mỗi pixel 4 byte).
- Tài nguyên sử dụng:
  - Bộ nhớ màn hình: Từ địa chỉ 0x10000000 (MONITOR_SCREEN) đến 0x10010000 (SCREEN_END).
  - Bàn phím hexa: Địa chỉ 0xFFFF0012 (IN_ADDRESS_HEXA_KEYBOARD) để thiết lập hàng quét, và 0xFFFF0014 (OUT_ADDRESS_HEXA_KEYBOARD) để đọc mã phím.

- Dữ liệu**:** Từ địa chỉ 0x10000000: Mảng board (4x4), biến turn (0 = O, 1 = X),biến start_turn (thứ tự đi đầu) và move_count để đếm nước đi.

1.2. Cấu trúc chương trình

a. Phần .data
   - turn: Lưu lượt chơi (0 = O, 1 = X).
   - board: Mảng con trỏ trỏ đến 4 hàng (line1, line2, line3, line4), mỗi hàng chứa 4 phần tử (0 = trống, 1 = X, 2 = O).
   - move_count: Đếm số nước đi (tối đa 16).
   - win_msg_X, win_msg_O, draw_msg: Chuỗi thông báo kết quả.

b. Phần .text
   - Khởi tạo: Vẽ lưới (luoi_cot, luoi_ngang) và thiết lập ngắt.
   - Chương trình chính (main): Thiết lập ngắt và chờ đầu vào.
   - Xử lý ngắt (handler): Quét bàn phím, xử lý nước đi, và kiểm tra trạng thái trò chơi.
   - Các hàm phụ: Vẽ ký hiệu (draw_X, draw_O), kiểm tra thắng (check_winner), và kiểm tra hòa (check_draw).

2. PHÂN TÍCH THUẬT TOÁN

2.1. Vẽ lưới

```
20  .text
21      li a0, MONITOR_SCREEN        # Nap dia chi bat dau cua man hinh
22      li t1, SCREEN_END            # Nap dia chi ket thuc
23  luoi_cot:
24      li t0, WHITE
25      sw t0, 0(a0)
26      addi a0, a0, 32
27      blt a0, t1, luoi_cot         # Neu a0 < SCREEN_END, tiep tuc vong lap
28
29      li a0, MONITOR_SCREEN
30      addi a0, a0, 896             # Dia chi dau hang ngang so 8
31      addi a1, a0, 128             # Dia chi dau hang ngang so 9
32  luoi_ngang:
33      li t0, WHITE
34      sw t0,0(a0)
35      addi a0, a0, 4
36      blt a0, a1, luoi_ngang
37  nhay_hang:
38      addi a0, a0, 896             # Dia chi dau hang ngang so 15, 22
39      addi a1, a0, 128             # Dia chi dau hang ngang so 16, 23
40      blt a0, t1, luoi_ngang       # Neu a0 < SCREEN_END, tiep tuc vong lap
```

- Mục đích: Tạo lưới 4x4 trên màn hình bitmap, chia thành 16 ô (mỗi ô 128x128 pixel).
- Đường dọc (luoi_cot):

- Bắt đầu từ MONITOR_SCREEN (0x10000000).
- Ghi màu trắng (0x00FFFFFF) vào các cột cách nhau 32 byte (8 pixel).
- Lặp đến khi chạm SCREEN_END (0x10010000).

- Đường ngang (luoi_ngang):
  - Vẽ tại các hàng 0, 8, 15, 22 (tương ứng offset 0, 896, 1792, 2688 byte từ MONITOR_SCREEN).
  - Mỗi đường dài 128 byte (32 pixel).
  - Sử dụng vòng lặp để ghi màu trắng vào các địa chỉ tương ứng.

2.2. Quét bàn phím

```
polling:
    li      t2, 0x1
get_key_code:
    li      t4, 0x80          # Kiem tra hang 4 va bat lai bit 7
    li      t1, IN_ADDRESS_HEXA_KEYBOARD
    add     t5, t4, t2
    sb      t5, 0(t1)         # Phai gan lai hang can kiem tra
    li      t1, OUT_ADDRESS_HEXA_KEYBOARD
    lbu     a0, 0(t1)
    beqz    a0, next_row      # Neu khong co nut nao bam, sang hai

    li      t0, 0
    li      s3, 16
cal_bit:
    # Lay gia tri hex tu bo nho
    add     t0, zero, a0
    # Tach chu so hang chuc (high nibble)
    srli    t1, t0, 4         # Dich phai 4 bit de lay phan chuc
    andi    t1, t1, 0xF       # Chi giu lai 4 bit thap
    # Tach chu so hang don vi (low nibble)
    andi    t2, t0, 0xF       # Lay 4 bit thap
    # Tim so mu co so 2 cao nhat cua hang chuc
    li      t3, -1            # Khoi tao so mu = -1 (neu t1 = 0)
    beqz    t1, skip_tens     # Neu t1 = 0, bo qua tim so mu
    li      t3, 0             # Khoi tao so mu = 0
```

```
find_tens_power:
    srli    t4, t1, 1         # Dich phai 1 bit
    beqz    t4, tens_done     # Neu = 0, da tim xong
    mv      t1, t4            # Cap nhat t1
    addi    t3, t3, 1         # Tang so mu
    j       find_tens_power
tens_done:
skip_tens:
    # Tim so mu co so 2 cao nhat cua hang don vi
    li      t5, -1            # Khoi tao so mu = -1 (neu t2 = 0)
    beqz    t2, skip_units    # Neu t2 = 0, bo qua tim so mu
    li      t5, 0             # Khoi tao so mu = 0
find_units_power:
    srli    t4, t2, 1         # Dich phai 1 bit
    beqz    t4, units_done    # Neu = 0, da tim xong
    mv      t2, t4            # Cap nhat t2
    addi    t5, t5, 1         # Tang so mu
    j       find_units_power
units_done:
skip_units:
    # Tinh dia chi: 34 + chuc*256 + donvi*8
    slli    t5, t5, 8         # t5 = t5 x 256
    slli    t3, t3, 3         # t3 = t3 * 8
    add     a0, t3, t5        # a0 = t3 + t5
    addi    a0, a0, 34
    slli    a0, a0, 2
    li      s0, MONITOR_SCREEN
    add     a0, s0, a0        # a0 = dia chi offset trong bitmap
```

```
invalid_key:
next_row:
    slli t2, t2, 1          # Dich trai de kiem tra hang tiep theo
    li t3, 0x10
    blt t2, t3, get_key_code
    j polling               # Quay lai quet ban phim
```

- Mục đích: Nhận mã phím từ bàn phím hexa và chuyển thành tọa độ.
- Thuật toán:

- Quét từng hàng (1, 2, 4, 8) bằng cách ghi giá trị 0x81, 0x82, 0x84, 0x88 vào IN_ADDRESS_HEXA_KEYBOARD.
- Đọc mã phím từ OUT_ADDRESS_HEXA_KEYBOARD.
- Nếu mã phím khác 0, phân tích:

+ Tách hàng chục (tens) và hàng đơn vị (units) từ mã phím.

+ Tính số mũ cao nhất của 2 cho tens và units (để ánh xạ phím thành tọa độ hàng/cột).

+ Tính offset trên bitmap: offset = 34 + tens*256 + units*8.

+ Chuyển thành địa chỉ bitmap: MONITOR_SCREEN + offset*4.

2.3. Vẽ kí hiệu X O

2.3.1. Trước khi vẽ X hoặc O, ta kiểm tra ô được bấm đã được điền chưa và lượt đi của X hay O.

```
121     # Kiem tra o da duoc dien chua
122     # t5 = row, t3 = col
123     srli t3,t3,3                # dua t3 ve lai gia tri col
124     srli t5,t5,8                # dua t5 ve lai gia tri col
125     la t1, board                # t1 = dia chi cua board
126     slli t2, t5, 2              # t2 = row * 4 (moi con tro 4 byte)
127     add t2, t1, t2              # t2 = dia chi cua board[row]
128     lw t1, 0(t2)                # t1 = dia chi cua lineX (hang row)
129     slli t2, t3, 2              # t2 = col * 4
130     add t1, t1, t2              # t1 = dia chi cua board[row][col]
131     lw t2, 0(t1)                # t2 = trang thai tai board[row][col]
132     # Neu o khong trong, quay lai quet ban phim
133     li s3,1
134     li s4,2
135     beq t2,s3, invalid_key
136     beq t2,s4, invalid_key
137
138     # Kiem tra luot hien tai
139     la t3, turn
140     lw t4, 0(t3)
141     beqz t4, draw_0         # Neu turn = 0, ve O
142     j draw_X                # Neu turn = 1, ve X
143
```

- Kiểm tra ô trống
  - Khôi phục giá trị row và col từ t5 và t3 bước tính offset bitmap.
  - Truy cập mảng board: Lấy trạng thái ô tại board[row][col].
  - Kiểm tra trạng thái:

+ Nếu trạng thái là 1 (X) hoặc 2 (O), nhảy đến invalid_key.

+ Nếu trạng thái là 0 (trống), tiếp tục xử lý lượt chơi.

- Xử lý lượt chơi
  - Lấy giá trị của biến turn từ bộ nhớ vào t4.

    + Nếu turn = 0, nhảy đến draw_O.

    + Nếu turn = 1, nhảy đến draw_X.

2.3.2 Vẽ kí hiệu X O

- Vẽ X: Ghi màu trắng vào các pixel tạo hình chữ X (5x5 pixel) theo mẫu:
  - Các vị trí: (0,0), (0,4), (1,1), (1,3), (2,2), (3,1), (3,3), (4,0), (4,4).
  - Offset tương ứng: 0, 16, 132, 140, 264, 388, 396, 512, 528 (tính bằng byte).
- Vẽ O: Ghi màu trắng vào các pixel tạo hình chữ O (5x5 pixel) theo mẫu:Các vị trí: hàng 0 và 4, hàng 1, 2, 3 (pixel đầu và cuối).
- Cập nhật mảng board với giá trị 1 (X) hoặc 2 (O).
- Chuyển lượt (turn) và tăng move_count.

```
144 draw_X:
145     li t0, WHITE
146     sw t0, 0(a0)
147     sw t0, 16(a0)
148     addi a0,a0,132
149     sw t0, 0(a0)
150     sw t0, 8(a0)
151     addi a0,a0,132
152     sw t0, 0(a0)
153     addi a0,a0,124
154     sw t0, 0(a0)
155     sw t0, 8(a0)
156     addi a0,a0,124
157     sw t0, 0(a0)
158     sw t0, 16(a0)
159     # Ghi trang thai vao board = 1 (X)
160     li t5, 1
161     sw t5, 0(t1)
162     # Chuyen turn = 1 -> 0 (O)
163     la t3, turn
164     li t4, 0
165     sw t4, 0(t3)
166     # Tang move_count
167     la t0, move_count
168     lw t1, 0(t0)
169     addi t1, t1, 1
170     sw t1, 0(t0)
171     # Kiem tra thang thua
172     jal check_winner
173     li a2, 1
174     li a3, 2
175     beq a1, a2, end_game          # Neu X thang
176     beq a1, a3, end_game          # Neu O thang
177     beqz a1, check_draw       # Neu hoa
178     j next_row
```

```
180 draw_O:
181     li t0, WHITE
182     sw t0, 0(a0)
183     sw t0, 4(a0)
184     sw t0, 8(a0)
185     sw t0, 12(a0)
186     sw t0, 16(a0)
187     addi a0, a0,128
188     sw t0, 0(a0)
189     sw t0, 16(a0)
190     addi a0, a0,128
191     sw t0, 0(a0)
192     sw t0, 16(a0)
193     addi a0, a0,128
194     sw t0, 0(a0)
195     sw t0, 16(a0)
196     addi a0, a0,128
197     sw t0, 0(a0)
198     sw t0, 4(a0)
199     sw t0, 8(a0)
200     sw t0, 12(a0)
201     sw t0, 16(a0)
202     # Ghi trang thai vao board = 2 (O)
203     li t5, 2
204     sw t5, 0(t1)
205     # Chuyen turn = 0 -> 1 (X)
206     la t3, turn
207     li t4, 1
208     sw t4, 0(t3)
209     # Tang move_count
210     la t0, move_count
211     lw t1, 0(t0)
212     addi t1, t1, 1
213     sw t1, 0(t0)
214     # Kiem tra thang thua
215     jal check_winner
216     li a2, 1
217     li a3, 2
218     beq a1, a2, end_game          # Neu X thang
219     beq a1, a3, end_game          # Neu O thang
220     beqz a1, check_draw       # Neu hoa
```

2.4. Kiểm tra thắng

- Mục đích: Kiểm tra xem có 4 ký hiệu giống nhau trên hàng, cột, hoặc đường chéo.

```
243 check_winner:
244     addi sp, sp, -16
245     sw ra, 12(sp)
246     sw s0, 8(sp)
247     sw s1, 4(sp)
248     sw s2, 0(sp)
249     # Kiem tra 4 hang
250     jal winRow0
251     bne a1, zero, end_check
252     jal winRow1
253     bne a1, zero, end_check
254     jal winRow2
255     bne a1, zero, end_check
256     jal winRow3
257     bne a1, zero, end_check
258     # Kiem tra 4 cot
259     jal winCol0
260     bne a1, zero, end_check
261     jal winCol1
262     bne a1, zero, end_check
263     jal winCol2
264     bne a1, zero, end_check
265     jal winCol3
266     bne a1, zero, end_check
267     # Kiem tra 2 duong cheo
268     jal winDiag0
269     bne a1, zero, end_check
270     jal winDiag1
271     bne a1, zero, end_check
272     # Khong co nguoi thang
273     li a1, 0
274     j end_check
275
276 end_check:
277     lw s2, 0(sp)
278     lw s1, 4(sp)
279     lw s0, 8(sp)
280     lw ra, 12(sp)
281     addi sp, sp, 16
282     ret
```

- Xác định xem có người chơi nào đạt được 4 ký hiệu giống nhau (1 cho X hoặc 2 cho O) trên một hàng, cột, hoặc đường chéo của bàn cờ 4x4.
- Gọi lần lượt các hàm để kiểm tra 4 hàng, 4 cột, 2 đường chéo. Nếu bất kỳ hàm nào trả về a1 ≠ 0 (1 hoặc 2), nhảy ngay đến end_check để kết thúc kiểm tra.
- Nếu không tìm thấy người thắng, gán a1 = 0 và nhảy đến end_check.
- Tải lại các thanh ghi từ ngăn xếp và trả về (ret) với giá trị a1.

  + a1 = 1: Người chơi X thắng.

  + a1 = 2: Người chơi O thắng.

+ a1 = 0: Không có người thắng (trò chơi tiếp tục hoặc kiểm tra hòa).

2.4.1 Kiểm tra thắng

- Kiểm tra hàng winRow0 đến winRow3 (tương tự nhau):
  - Lấy giá trị ô đầu tiên trong hàng (board[row][0]).
  - Nếu ô không trống, so sánh với 3 ô còn lại trong hàng.
  - Nếu tất cả giống nhau, trả về giá trị (1 hoặc 2); ngược lại trả về 0.

```
284 winRow0:
285     la t0, board
286     lw t1, 0(t0)            # t1 = dia chi cua line1
287     lw t2, 0(t1)            # t2 = board[0][0]
288     beqz t2, no_win_row0    # Neu o trong, khong thang
289     lw t3, 4(t1)            # t3 = board[0][1]
290     bne t2, t3, no_win_row0
291     lw t4, 8(t1)            # t4 = board[0][2]
292     bne t2, t4, no_win_row0
293     lw t5, 12(t1)           # t5 = board[0][3]
294     bne t2, t5, no_win_row0
295     mv a1, t2               # a1 = 1 (X) hoac 2 (O)
296     ret
297 no_win_row0:
298     li a1, 0
299     ret
```

- Kiểm tra cột winCol0 đến winCol3 (tương tự nhau):
  - Lấy giá trị ô đầu tiên trong cột (board[0][col]).
  - Nếu ô không trống, so sánh với 3 ô còn lại trong hàng.
  - Nếu tất cả giống nhau, trả về giá trị (1 hoặc 2); ngược lại trả về 0.

```
352  winCol0:
353      la t0, board
354      lw t1, 0(t0)              # t1 = line1
355      lw t2, 0(t1)              # t2 = board[0][0]
356      beqz t2, no_win_col0
357      lw t3, 4(t0)              # t3 = line2
358      lw t3, 0(t3)              # t3 = board[1][0]
359      bne t2, t3, no_win_col0
360      lw t4, 8(t0)              # t4 = line3
361      lw t4, 0(t4)              # t4 = board[2][0]
362      bne t2, t4, no_win_col0
363      lw t5, 12(t0)             # t5 = line4
364      lw t5, 0(t5)              # t5 = board[3][0]
365      bne t2, t5, no_win_col0
366      mv a1, t2
367      ret
368  no_win_col0:
369      li a1, 0
370      ret
```

- Kiểm tra đường chéo (winDiag0, winDiag1):
  - Đường chéo chính: So sánh board[0][0], board[1][1], board[2][2], board[3][3].
  - Đường chéo phụ: So sánh board[0][3], board[1][2], board[2][1], board[3][0].
  - Nếu không có người thắng, trả về a1= 0.

```
winDiag0:
    la t0, board
    lw t1, 0(t0)              # t1 = line1
    lw t2, 0(t1)              # t2 = board[0][0]
    beqz t2, no_win_diag0
    lw t3, 4(t0)              # t3 = line2
    lw t3, 4(t3)              # t3 = board[1][1]
    bne t2, t3, no_win_diag0
    lw t4, 8(t0)              # t4 = line3
    lw t4, 8(t4)              # t4 = board[2][2]
    bne t2, t4, no_win_diag0
    lw t5, 12(t0)             # t5 = line4
    lw t5, 12(t5)             # t5 = board[3][3]
    bne t2, t5, no_win_diag0
    mv a1, t2
    ret
no_win_diag0:
    li a1, 0
    ret
```

2.4.2. Kiểm tra hòa

```
check_draw:
    la t0, move_count
    lw t1, 0(t0)
    li t2, 16                 # Ban co 4x4 co 16 o
    beq t1, t2, is_draw       # Neu da di 16 nuoc, hoa
    j polling                 # Tiep tuc quet ban phim

is_draw:
    la a0, draw_msg
    li a7, 4
    ecall
    la a0, newline
    ecall
    j reset_game
```

- Mục đích: Xác định trò chơi hòa khi bàn cờ đầy.
  - So sánh move_count với 16.
  - Nếu bằng 16, hiển thị thông báo "Draw!" và kết thúc.

2.5. Hiển thị kết quả

- Nếu check_winner trả về 1, in "X wins!".
- Nếu trả về 2, in "O wins!".
- Nếu hòa, in "Draw!".
- Dùng ecall (a7 = 4) để in chuỗi, sau đó thoát (a7 = 10).

```
474  end_game:
475      # In thong bao thang
476      li t0, 1
477      beq a1, t0, x_wins
478      la a0, win_msg_0
479      j print_winner
480  x_wins:
481      la a0, win_msg_X
482  print_winner:
483      li a7, 4
484      ecall
485      la a0, newline
486      ecall
```

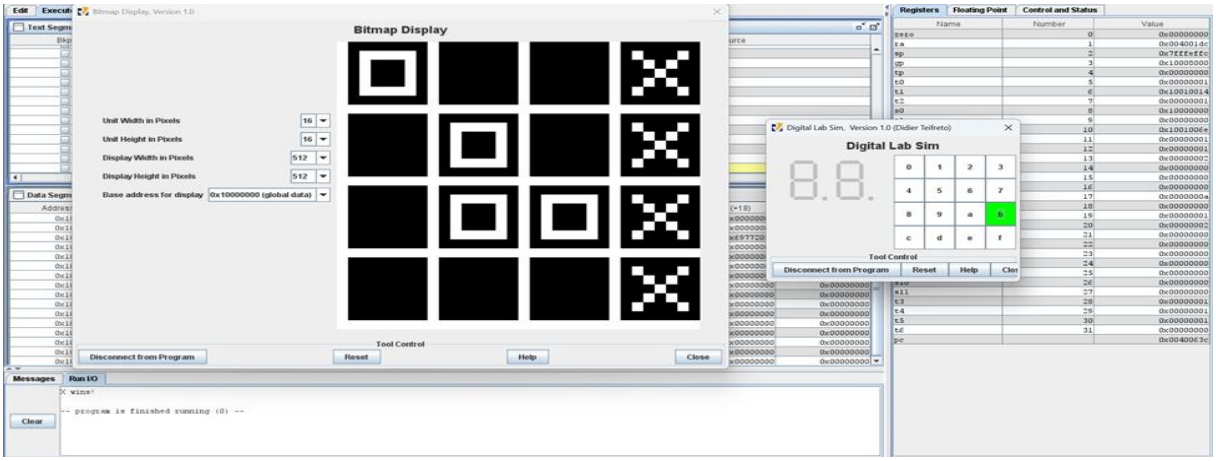
2.6. Reset game

- Khởi tạo lại trò chơi: Đặt lại tất cả các biến và trạng thái cần thiết để bắt đầu một ván mới, bao gồm:
- Xóa bàn cờ (đặt tất cả ô về trạng thái trống).
- Đặt lại số đếm nước đi (move_count) về 0.
- Mỗi khi reset, đảo giá trị của start_turn (từ 0 thành 1 hoặc từ 1 thành 0) và gán giá trị này cho turn để luân phiên thứ tự đi đầu.
- Xóa màn hình bitmap để vẽ lại bàn cờ từ đầu.
- Sau khi reset, nhảy đến nhãn new_game để bắt đầu ván mới.

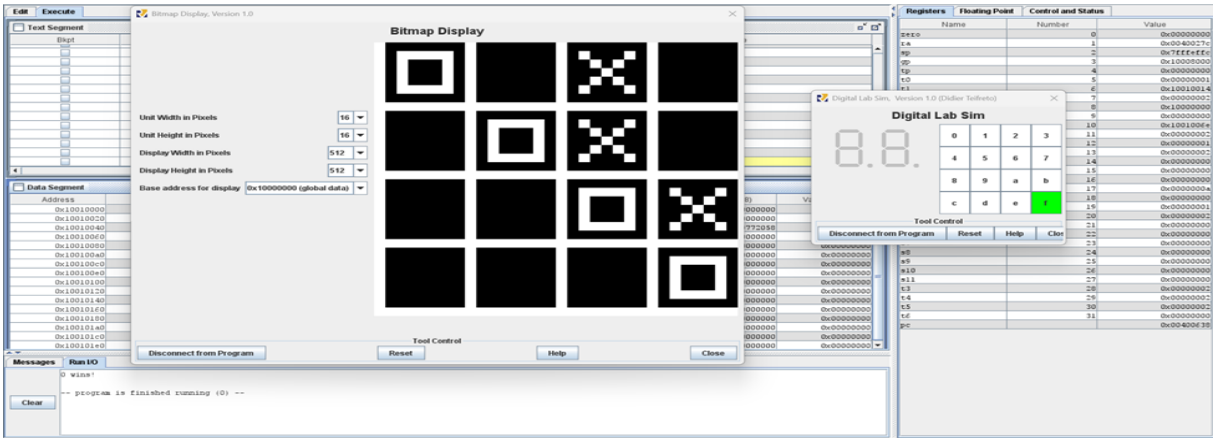
```
reset_game:
    # Reset board (dat tat ca ve 0)
    la t0, board
    li t2, 4          # 4 hang
    mv t4, t0         # t4 = dia chi board
reset_board_loop:
    lw t1, 0(t4)      # t1 = dia chi lineX
    sw zero, 0(t1)
    sw zero, 4(t1)
    sw zero, 8(t1)
    sw zero, 12(t1)
    addi t4, t4, 4    # Chuyen den con tro hang tiep theo
    addi t2, t2, -1   # Giam so hang
    bnez t2, reset_board_loop

    # Reset move_count
    la t0, move_count
    sw zero, 0(t0)

    # Reset turn (luan phien bat dau)
    la t0, start_turn
    lw t1, 0(t0)      # t1 = start_turn (0 hoac 1)
    xori t1, t1, 1    # Dao gia tri: 0 -> 1, 1 -> 0
    sw t1, 0(t0)      # Luu start_turn moi
    la t0, turn
    sw t1, 0(t0)      # Gan turn = start_turn moi

    # Xoa man hinh (boi toan bo man hinh ve mau den)
    li a0, MONITOR_SCREEN
    li t1, SCREEN_END
clear_screen:
    sw zero, 0(a0)
    addi a0, a0, 4
    blt a0, t1, clear_screen
    j new_game
```

3. Kết quả

3.1. X wins



3.2. O wins



3.3. Hòa