HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN
KIẾN TRÚC MÁY TÍNH

Nhóm: LỚP N01 – NHÓM 04

Tên các thành viên

Họ tên	Mã sinh viên
Phạm Hoài Nam	**B21DCCN554**
Nguyễn Hải Yến	**B21DCCN129**
Hoàng Thu Hà	**B21DCCN309**
Ngô Tuấn Lộc	**B21DCCN491**
Trần Trọng Việt	**B21DCCN791**

Giảng viên hướng dẫn: ThS. Đinh Xuân Trường

Hà Nội 2023

Mục lục

I, Giới thiệu đề tài

Ping Pong là một trò chơi điện tử cổ điển và phổ biến, được lấy cảm hứng từ môn bóng bàn truyền thống. Ra đời những năm 1970, game Ping Pong trở thành một trong những game máy tính đầu tiên được phát triển, nó cung cấp một trải nghiệm chơi đơn giản nhưng vẫn đòi hỏi kỹ năng và phản xạ nhanh nhạy từ người chơi.

Trò chơi Ping Pong đã trải qua nhiều phiên bản và thể loại khác nhau trong suốt thời gian tồn tại của nó tuy nhiên có thể phân thành hai thể loại chính: một người chơi và hai người chơi.

- Chế độ một người chơi:

Trong chế độ này, người chơi sẽ đối đầu với máy tính và cố gắng đánh bại nó bằng cách điều khiển thanh trượt của mình để đỡ bóng và gửi nó trở lại máy tính.

Trong chế độ này, máy tính sẽ thường có một thuật toán điều khiển tự động để cố gắng ngăn chặn bóng và tạo khó khăn cho người chơi. Độ khó của máy tính có thể được điều chỉnh để tạo ra một trải nghiệm thử thách phù hợp với kỹ năng của người chơi.

- Chế độ hai người chơi:

Trong chế độ này, hai người chơi sẽ đấu trực tiếp với nhau. Mỗi người chơi sẽ điều khiển một thanh trượt riêng và cố gắng đỡ bóng và gửi nó trở lại phía đối thủ.

Mục tiêu của trò chơi này là đưa bóng qua thanh trượt của đối thủ để ghi điểm. Người chơi nào ghi được số điểm nhất định hoặc người chơi nào đạt điểm cao hơn đối thủ trong một khoảng thời gian nhất định sẽ chiến thắng.

Tóm lại, cả hai chế độ chơi đều mang lại trải nghiệm thú vị và thử thách. Chế độ một người chơi cho phép người chơi rèn kỹ năng và chiến đấu với máy tính, trong khi chế độ hai người chơi tạo ra một trận đấu đối kháng giữa hai người chơi thực.

Việc có cả hai chế độ chơi này làm cho Ping Pong trở thành một trò chơi linh hoạt và phổ biến, phù hợp cho cả việc chơi đơn và chơi với bạn bè và gia đình.

Trong những năm gần đây, Ping Pong đã được cải tiến và cập nhật với công nghệ hiện đại. Trò chơi có thể dễ dàng được tìm thấy trên các nền tảng di động, máy tính và các hệ thống chơi game hiện đại. Ngoài ra, có nhiều phiên bản trực tuyến và phức tạp hơn của trò chơi so với trước. Với thiết kế đơn giản, thú vị, có tính cạnh tranh, trò chơi Ping Pong vẫn là một trong những trò chơi phổ biến và được yêu thích cho đến ngày nay và được nhiều bạn trẻ đón nhận.

Lập trình trò chơi Ping Pong bằng ngôn ngữ Assembly là một đề tài rất thú vị và có nhiều tác dụng như:

1. Hiểu rõ hơn về ngôn ngữ lập trình Assembly: Lập trình game Ping Pong sẽ đòi hỏi bạn học cách viết mã Assembly, một ngôn ngữ gần gũi với ngôn ngữ máy tính. Việc làm quen với Assembly sẽ giúp bạn hiểu rõ hơn về cách máy tính hoạt động và cách tương tác với phần cứng.
2. Học cách xử lý và tương tác với sự kiện: Trò chơi Ping Pong yêu cầu xử lý sự kiện như di chuyển thanh trượt, va chạm với bóng và tính điểm. Việc lập trình game này bằng Assembly sẽ giúp bạn hiểu cách xử lý sự kiện và tương tác với người chơi.
3. Thách thức và rèn luyện kỹ năng lập trình: Lập trình game Ping Pong bằng Assembly đòi hỏi bạn có kiến thức về lập trình và khả năng giải quyết vấn đề. Nó sẽ đẩy bạn đến giới hạn của khả năng lập trình và rèn luyện kỹ năng logic và sáng tạo.
4. Tái hiện lại một trò chơi độc đáo: Điều này có thể gợi nhớ lại những kỷ niệm với các trò chơi điện tử cổ điển và tạo ra một trải nghiệm đặc biệt cho người chơi.

Tóm lại, có thể thấy rằng việc lập trình game Ping Pong bằng ngôn ngữ Assembly đem lại rất nhiều lợi ích cho người học.

II, Nội dung chính của đề tài

- Lập trình game Ping Pong bằng ngôn ngữ Assembly.
- Tổng quan trò chơi:
    - Trò chơi Ping Pong nhóm thiết kế là thể loại dành cho 2 người chơi, mỗi người sẽ điều khiển một thanh hình chữ nhật để đánh bóng qua màn hình và cố gắng ghi điểm bằng cách đánh bóng vượt qua đối thủ.
    - Người chơi sử dụng cố gắng đánh bóng qua phần sân đối thủ mà không để đối thủ đỡ được. Mỗi khi bóng vượt qua đối thủ và rơi xuống phía đối diện màn hình, người chơi sẽ nhận được 1 điểm, người chơi nào đạt được 5 điểm trước sẽ trở thành người chiến thắng.
- Cách chơi:
  Người chơi thứ nhất sẽ điều khiển thanh đập bên trái, sử dụng các phím: phím ‘W’ để đi lên và phím ‘S’ để đi xuống.
  Người chơi thứ hai điều khiển thanh đập bên phải, sử dụng các phím: phím ‘O’ để đi lên và phím ‘L’ để đi xuống.
- Các chức năng có trong trò chơi:
  1. Menu hiển thị

Menu hiển thị xuất hiện ở đầu khi người chơi mới vào game hoặc khi kết thúc bàn chơi, màn hình Menu sẽ bao gồm: chọn chế độ trò chơi, thoát khỏi game.

2. Màn hình thông báo

Màn hình thông báo sẽ xuất hiện kết quả của người chơi thứ nhất. Trong trò chơi sẽ chỉ có một người thắng duy nhất nên nếu người chơi thứ nhất là người thắng thì người chơi thứ

hai sẽ bị thua. Ngược lại người chơi thứ nhất bị thua thì người chơi thứ hai sẽ là người thắng.

Ngoài ra trên màn hình thông báo còn có 1 số tính năng khác như:

1. Nhấn phím ‘R’ để chơi lại
2. Nhấn phím ‘E’ để thoát khỏi màn chơi, quay về màn hình Main Menu ban đầu.

III, Cài đặt chương trình

a. Lý do sử dụng DOSBox

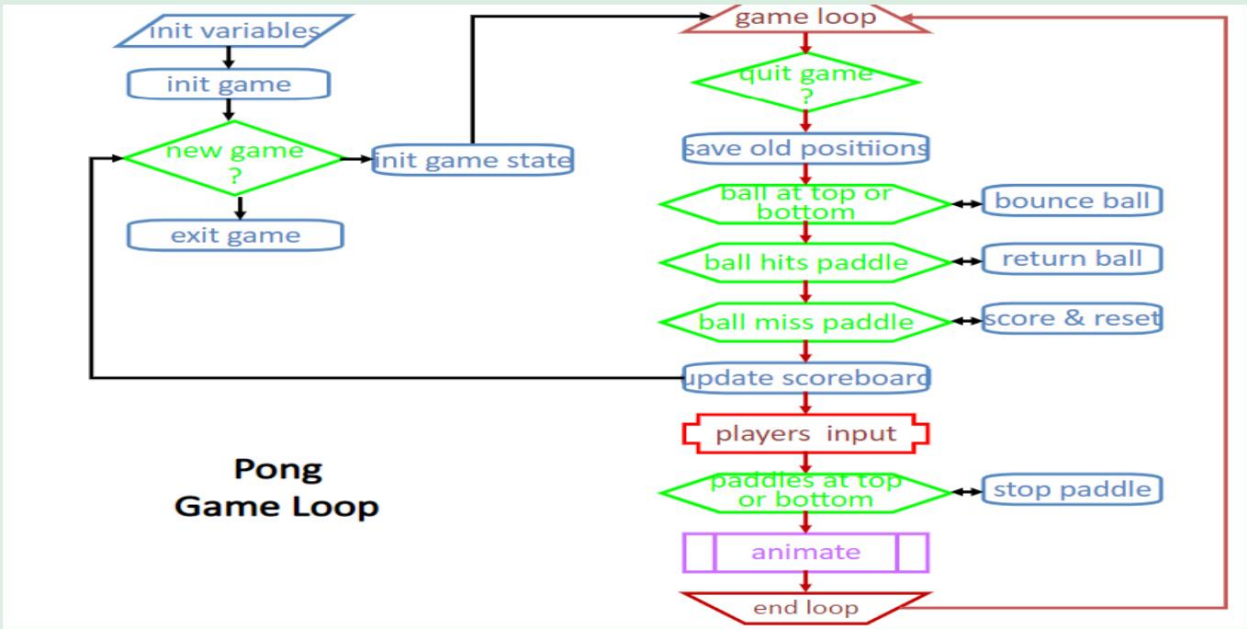
- Trong phần thực hành, nhóm có sử dụng hàm ngắt INT10H để tương tác với màn hình giúp có thể thực hiện các chức năng điều khiển và thao tác đối với giao diện màn hình.
- Hàm ngắt này không được hỗ trợ trên Emu8086.

b. Hướng dẫn cài đặt DosBox

- Bước 1: Tải folder code về trên máy tính, lưu vào ổ C của thiết bị, đặt tên file là KTMT.
  Link: https://bom.so/kvuVAR
- Bước 2: Tải và cài đặt phần mềm DosBox 0.74
  Link: https://bom.so/oMR7pO
- Bước 3: Mở DOSBox 0.74.exe và nhập lần lượt những dòng lệnh sau:
  + Dòng thứ 1: “mount c c:\”
  + Dòng thứ 2: “c:\”
  + Dòng thứ 3: “cd KTMT”
  + Dòng thứ 4: “Pong.exe”
  Sau đó nhấn Enter để bắt đầu tham gia trò chơi.

IV, Thuật toán Assembly

IV.1. Lưu đồ thuật toán



IV.2. Một số hàm ngắt quan trọng được sử dụng

1. Hàm ngắt INT10H

- Hàm ngắt INT10H được sử dụng để tương tác với màn hình trong lập trình Assembly. Nó cung cấp các chức năng điều khiển và thao tác đối với giao diện hiển thị.
- Trong chương trình, hàm ngắt INT10H được dùng để:
  + Cài đặt chế độ video: AH = 00H.
  + Tạo màn hình với AL = 13H 320x200 Đồ họa 256 màu (MCGA, VGA).
  + Vẽ các pixel: AH = 0CH để đặt cấu hình pixel, AL = 0FH để chọn màu mặc định cho pixel là màu trắng. dụng để tương tác

2. Hàm ngắt INT 16H

- Hàm ngắt INT16H được sử dụng để tương tác với bàn phím trong lập trình Assembly. Nó cho phép chương trình đọc các ký tự từ bàn phím hoặc thực hiện các thao tác liên quan đến đầu vào từ người dùng trong lập trình Assembly. Nó cung cấp các chức năng điều khiển và thao tác đối với giao diện hiển thị.
- Trong chương trình, hàm ngắt INT 16H dùng để:
  + Hàm 01h của ngắt 16h: Hỏi vùng đệm bàn phím (rỗng hay không rỗng). Đối với bàn phím từ 101 phím trở lên ta mở rộng hàm 01h thành hàm 11h.
  + Hàm 00H của ngắt 16H: Lấy một mã hai byte trong vùng đệm bàn phím BIOS. Nếu vùng đệm rỗng hàm sẽ đợi cho đến khi vùng đệm khác rỗng (có phím). Đối với bàn phím từ 101 phím trở lên ta mở rộng hàm 00H thành hàm 10H.

3. Hàm ngắt INT21H

- Hàm ngắt INT 21H là một hàm ngắt hệ thống rất mạnh mẽ trong lập trình Assembly. Nó cung cấp các chức năng liên quan đến I/O (input/output) và quản lý tệp tin trong

hệ điều hành DOS. Đó là các hàm thao tác vào/ra đối với kí tự, chuỗi ký tự, file, thư mục, kết thúc chương trình và trả lại quyền điều khiển cho Hệ điều hành DOS.

IV.3. Mã nguồn Assembly

1. Cài đặt màn hình, không gian, thông số trò chơi

a. Các biến lưu kích thước màn hình, trạng thái và tin nhắn trò chơi

```
     WINDOW_WIDTH DW 140h                ;chiều rộng của cửa sổ (320 pixels)

    WINDOW_HEIGHT DW 0C8h               ;chiều cao của cửa sổ (200 pixels)
    WINDOW_BOUNDS DW 6                  ;biến kiểm tra va chạm

    TIME_AUX DB 0                       ;biến kiểm tra thời gian
    GAME_ACTIVE DB 1                    ;biến kiểm tra trạng thái trò chơi (1 -> Yes, 0 -> No(game over))
    EXITING_GAME DB 0
    WINNER_INDEX DB 0                   ;chỉ số của người chiến thắng (1 -> player one, 2 -> player two)
    CURRENT_SCENE DB 0                  ;trạng thái màn hình hiện tại (0 -> main menu, 1 -> game)

    TEXT_PLAYER_ONE_POINTS DB '0','$'    ;điểm của người chơi thứ 1
    TEXT_PLAYER_TWO_POINTS DB '0','$'    ;điểm của người chơi thứ 2
    TEXT_GAME_OVER_TITLE DB 'CONGTATULATIONS!!!','$' ;title sau khi kết thúc trò chơi
    TEXT_GAME_OVER_WINNER DB 'Player 0 won','$' ;text lưu người chiến thắng
    TEXT_GAME_OVER_PLAY_AGAIN DB 'Press R to play again','$' ;tin nhắn tiếp tục trò chơi
    TEXT_GAME_OVER_MAIN_MENU DB 'Press E to exit to main menu','$' ;tin nhắn trở về menu
    TEXT_MAIN_MENU_TITLE DB 'MAIN MENU','$' ;title menu
    TEXT_MAIN_MENU_SINGLEPLAYER DB 'MULTIPLAYER - S KEY','$' ;chế độ chơi 2 người
    TEXT_MAIN_MENU_MULTIPLAYER DB 'SINGLEPLAYER - M KEY','$' ;chế độ chơi 1 người
    TEXT_MAIN_MENU_EXIT DB 'EXIT GAME - E KEY','$' ;tin nhắn thoát trò chơi
```

b. Các biến lưu thông số của bóng, thanh đỡ và điểm của người chơi

```
   BALL_ORIGINAL_X DW 0A0h              ;X vị trí của quả bóng khi bắt đầu trò chơi
    BALL_ORIGINAL_Y DW 64h              ;Y vị trí của quả bóng khi bắt đầu trò chơi
    BALL_X DW 0A0h                      ;vị trí X hiện tại (cột) của quả bóng
    BALL_Y DW 64h                       ;vị trí Y hiện tại (đường) của quả bóng
    BALL_SIZE DW 06h       ;kích thước của quả bóng (quả bóng có chiều rộng và chiều cao bao nhiêu pixel)
    BALL_VELOCITY_X DW 05h              ;X Vận tốc X (ngang) của quả bóng
    BALL_VELOCITY_Y DW 02h              ;Y Vận tốc Y (dọc) của quả bóng

    PADDLE_LEFT_X DW 0Ah                ;vị trí X hiện tại của thanh đánh bóng bên trái
    PADDLE_LEFT_Y DW 55h                ;vị trí Y hiện tại của thanh đánh bóng bên trái
    PLAYER_ONE_POINTS DB 0             ;điểm hiện tại của người chơi bên trái (người chơi 1)

    PADDLE_RIGHT_X DW 130h              ;vị trí X hiện tại của thanh đánh bóng bên phải
    PADDLE_RIGHT_Y DW 55h               ;vị trí Y hiện tại của thanh đánh bóng bên trái
    PLAYER_TWO_POINTS DB 0             ;điểm hiện tại của người chơi bên phải (người chơi 2)

    PADDLE_WIDTH DW 06h                 ;chiều rộng thanh đánh bóng
    PADDLE_HEIGHT DW 25h                ;chiều dài thanh đánh bóng
    PADDLE_VELOCITY DW 0Fh              ;vận tốc thanh đánh bóng
```

2. Vòng lặp của game

```
CODE SEGMENT PARA 'CODE'

    MAIN PROC FAR
    ASSUME CS:CODE,DS:DATA,SS:STACK      ;đưa code, dữ liệu và phân đoạn ngăn xếp các thanh ghi tương ứng
    PUSH DS                              ;đẩy DS vào ngăn xếp
    SUB AX,AX                            ;clear AX
    PUSH AX                              ;đẩy AX vào ngăn xếp
    MOV AX,DATA                          ;lưu DATA vào AX
    MOV DS,AX                            ;lưu AX vào DS
    POP AX                               ;đẩy top của ngăn xếp vào thanh ghi AX
```

```
    POP AX                              ;đẩy top của ngăn xếp vào thanh ghi AX

        CALL CLEAR_SCREEN               ;đặt cấu hình chế độ video ban đầu

        CHECK_TIME:                     ;vòng kiểm tra thời gian

            CMP EXITING_GAME,01h
            JE START_EXIT_PROCESS

            CMP CURRENT_SCENE,00h
            JE SHOW_MAIN_MENU

            CMP GAME_ACTIVE,00h
            JE SHOW_GAME_OVER

            MOV AH,2Ch                  ;lấy thời gian hệ thống
            INT 21h                     ;CH = giờ CL = phút DH = giây DL = 1/100 giây

            CMP DL,TIME_AUX            ;kiểm tra xem thời gian hiện tại có bằng thời gian trước đó không?
            JE CHECK_TIME                   ;nếu giống thì kiểm tra lại

            ;nếu đến đây, tức là đã kiểm tra xong

            MOV TIME_AUX,DL             ;cập nhật thời gian

            CALL CLEAR_SCREEN           ;xóa màn hình bằng cách khởi động lại chế độ video

            CALL MOVE_BALL              ;di chuyển quả bóng
            CALL DRAW_BALL              ;vẽ bóng

            CALL MOVE_PADDLES           ;di chuyển hai thanh bóng (kiểm tra cách nhấn phím)
            CALL DRAW_PADDLES           ;vẽ hai thanh đập với vị trí đã được cập nhật

            CALL DRAW_UI                ;vẽ giao diện người dùng trò chơi

            JMP CHECK_TIME              ;kiểm tra lại thời gian

            SHOW_GAME_OVER:
                CALL DRAW_GAME_OVER_MENU
                JMP CHECK_TIME

            SHOW_MAIN_MENU:
                CALL DRAW_MAIN_MENU
                JMP CHECK_TIME

            START_EXIT_PROCESS:
                CALL CONCLUDE_EXIT_GAME

        RET
    MAIN ENDP
```

Đoạn mã trên là một vòng lặp của chương trình. Các lệnh trong đoạn mã này có các tác dụng sau:

1. ASSUME CS:CODE, DS:DATA, SS:STACK: Đây là lệnh chỉ định các giá trị đặc tả segment (segment override) cho các thanh ghi CS (Code Segment), DS (Data Segment) và SS (Stack Segment).

2. PUSH DS: Đẩy giá trị của thanh ghi DS (Data Segment) vào ngăn xếp.

3. SUB AX, AX: Xóa giá trị của thanh ghi AX bằng cách trừ nó từ chính nó. Mục đích là xóa AX.

4. PUSH AX: Đẩy giá trị của thanh ghi AX vào ngăn xếp.

5. MOV AX, DATA: Di chuyển giá trị của hằng số DATA vào thanh ghi AX.

6. MOV DS, AX: Di chuyển giá trị của thanh ghi AX vào thanh ghi DS (Data Segment).

7. POP AX: Lấy giá trị trên cùng của ngăn xếp và lưu vào thanh ghi AX.

8. POP AX: Lấy giá trị trên cùng của ngăn xếp và lưu vào thanh ghi AX.

Sau đó, chương trình tiếp tục vào vòng lặp CHECK_TIME để kiểm tra thời gian. Trong vòng lặp này, chương trình kiểm tra các điều kiện để xác định trạng thái của trò chơi và thực hiện các hành động tương ứng.

Cụ thể, chương trình kiểm tra EXITING_GAME, CURRENT_SCENE, và GAME_ACTIVE để xác định trạng thái hiện tại của trò chơi và thực hiện các hành động tương ứng. Sau đó, nó lấy thời gian hệ thống và so sánh với giá trị TIME_AUX để kiểm tra xem thời gian đã thay đổi hay chưa. Nếu có sự thay đổi thì chương trình cập nhật thời gian, xóa màn hình, di chuyển và vẽ các thành phần trong trò chơi.

Cuối cùng, chương trình thực hiện các nhãn SHOW_GAME_OVER, SHOW_MAIN_MENU và START_EXIT_PROCESS tương ứng với các trạng thái của trò chơi.

3. Vẽ bóng và 2 thanh đỡ

a. Vẽ bóng

```
DRAW_BALL PROC NEAR

        MOV CX,BALL_X                   ;đặt cột ban đầu (X)
        MOV DX,BALL_Y                   ;đặt dòng ban đầu (Y)

        DRAW_BALL_HORIZONTAL:
            MOV AH,0Ch                              ;đặt cấu hình để viết một pixel
            MOV AL,0Fh                  ;chọn màu trắng làm màu
            MOV BH,00h                  ;đặt số trang
            INT 10h                     ;thực hiện cấu hình

            INC CX                      ;CX = CX + 1
            MOV AX,CX ;CX - BALL_X > BALL_SIZE (Y -> đi đến dòng tiếp theo,N -> tiếp tục đến cột tiếp theo
            SUB AX,BALL_X
            CMP AX,BALL_SIZE
            JNG DRAW_BALL_HORIZONTAL

            MOV CX,BALL_X               ;thanh ghi CX quay trở lại cột ban đầu
            INC DX                      ;tiến lên một dòng

            MOV AX,DX;DX - BALL_Y > BALL_SIZE (Y -> thoát khỏi thủ tục này,N -> tiếp tục đến dòng sau
            SUB AX,BALL_Y
            CMP AX,BALL_SIZE
            JNG DRAW_BALL_HORIZONTAL

        RET
    DRAW_BALL ENDP
```

Đoạn mã có tác dụng vẽ một quả bóng trên màn hình. Cụ thể, quá trình vẽ quả bóng được thực hiện theo hai bước:

1. Vẽ các điểm trên hàng ngang của quả bóng:

- Đầu tiên, thanh ghi CX được đặt bằng giá trị của biến BALL_X, đại diện cho cột ban đầu của quả bóng.

- Thủ tục sử dụng lệnh INT 10h với AX=0Ch để cấu hình viết một pixel trên màn hình với màu trắng.

- Sau đó, thanh ghi CX tăng lên một đơn vị (INC CX) để chuyển đến cột tiếp theo.

- Tiếp theo, tính toán AX bằng cách lấy giá trị hiện tại của CX trừ đi BALL_X và so sánh với BALL_SIZE để xác định liệu đã vẽ đủ chiều dài của quả bóng hay chưa.

- Nếu AX (CX - BALL_X) nhỏ hơn hoặc bằng BALL_SIZE (kích thước của quả bóng), quá trình vẽ trên hàng ngang vẫn chưa hoàn thành, do đó quay lại bước vẽ tiếp theo (JNG DRAW_BALL_HORIZONTAL).

2. Vẽ các điểm trên dòng dọc của quả bóng:

- Thủ tục tăng giá trị của thanh ghi DX lên một đơn vị (INC DX), để chuyển đến dòng tiếp theo của quả bóng.

- Tiếp theo, tính toán AX bằng cách lấy giá trị hiện tại của DX trừ đi BALL_Y và so sánh với BALL_SIZE để xác định liệu đã vẽ đủ chiều cao của quả bóng hay chưa.

- Nếu AX (DX - BALL_Y) nhỏ hơn hoặc bằng BALL_SIZE, quá trình vẽ trên dòng dọc vẫn chưa hoàn thành, do đó quay lại bước vẽ tiếp theo (JNG DRAW_BALL_HORIZONTAL).

b. Vẽ thanh đỡ trái

```
DRAW_PADDLES PROC NEAR

        MOV CX,PADDLE_LEFT_X            ;đặt cột ban đầu (X)
        MOV DX,PADDLE_LEFT_Y            ;đặt dòng ban đầu (Y)

        DRAW_PADDLE_LEFT_HORIZONTAL:
            MOV AH,0Ch                  ;đặt cấu hình để viết một pixel
            MOV AL,0Fh                  ;chọn màu trắng làm màu
            MOV BH,00h                  ;đặt số trang
            INT 10h                     ;thực hiện cấu hình

            INC CX                      ;CX = CX + 1
            MOV AX,CX                   ;CX - PADDLE_LEFT_X > PADDLE_WIDTH (Y -> đi đến dòng tiếp theo,N -
> tiếp tục đến cột tiếp theo
            SUB AX,PADDLE_LEFT_X
            CMP AX,PADDLE_WIDTH
            JNG DRAW_PADDLE_LEFT_HORIZONTAL

            MOV CX,PADDLE_LEFT_X        ;thanh ghi CX quay trở lại cột ban đầu
            INC DX                      ;tiến lên một dòng

             MOV AX,DX                   ;DX - PADDLE_LEFT_Y > PADDLE_HEIGHT (Y -> tiến lên một dòng,N ->
tiếp tục đến dòng tiếp theo
            SUB AX,PADDLE_LEFT_Y
            CMP AX,PADDLE_HEIGHT
            JNG DRAW_PADDLE_LEFT_HORIZONTAL
```

Đoạn mã trên là một thủ tục (procedure) có tên là DRAW_PADDLES, và nó có tác dụng vẽ thanh đập (paddle) trên màn hình. Cụ thể, đoạn mã này vẽ thanh đập bên trái (PADDLE_LEFT).

Quá trình vẽ thanh đập được thực hiện theo hai bước tương tự như quá trình vẽ quả bóng trong đoạn mã trước:

1. Vẽ các điểm trên hàng ngang của thanh đập:

- Đầu tiên, thanh ghi CX được đặt bằng giá trị của biến PADDLE_LEFT_X, đại diện cho cột ban đầu của thanh đập bên trái.

- Thủ tục sử dụng lệnh INT 10h với AX=0Ch để cấu hình viết một pixel trên màn hình với màu trắng.

- Sau đó, thanh ghi CX tăng lên một đơn vị (INC CX) để chuyển đến cột tiếp theo.

- Tiếp theo, tính toán AX bằng cách lấy giá trị hiện tại của CX trừ đi PADDLE_LEFT_X và so sánh với PADDLE_WIDTH để xác định liệu đã vẽ đủ chiều dài của thanh đập hay chưa.

- Nếu AX (CX - PADDLE_LEFT_X) nhỏ hơn hoặc bằng PADDLE_WIDTH (độ rộng của thanh đập), quá trình vẽ trên hàng ngang vẫn chưa hoàn thành, do đó quay lại bước vẽ tiếp theo (JNG DRAW_PADDLE_LEFT_HORIZONTAL).

2. Vẽ các điểm trên dòng dọc của thanh đập:

- Thủ tục tăng giá trị của thanh ghi DX lên một đơn vị (INC DX), để chuyển đến dòng tiếp theo của thanh đập.

- Tiếp theo, tính toán AX bằng cách lấy giá trị hiện tại của DX trừ đi PADDLE_LEFT_Y và so sánh với PADDLE_HEIGHT để xác định liệu đã vẽ đủ chiều cao của thanh đập hay chưa.

- Nếu AX (DX - PADDLE_LEFT_Y) nhỏ hơn hoặc bằng PADDLE_HEIGHT, quá trình vẽ trên dòng dọc vẫn chưa hoàn thành, do đó quay lại bước vẽ tiếp theo (JNG DRAW_PADDLE_LEFT_HORIZONTAL).

Sau khi hoàn thành cả hai bước, thủ tục DRAW_PADDLES kết thúc và trở về vị trí gọi thủ tục ban đầu.

c. Vẽ thanh đỡ phải

```
        MOV CX,PADDLE_RIGHT_X           ;đặt cột ban đầu (X)
        MOV DX,PADDLE_RIGHT_Y           ;đặt dòng ban đầu (Y)

        DRAW_PADDLE_RIGHT_HORIZONTAL:
            MOV AH,0Ch                  ;đặt cấu hình để viết một pixel
            MOV AL,0Fh                  ;chọn màu trắng làm màu
```

```
            MOV BH,00h                  ;đặt số trang
            INT 10h                     ;thực hiện cấu hình

            INC CX                      ;CX = CX + 1
            MOV AX,CX                   ;CX - PADDLE_RIGHT_X > PADDLE_WIDTH (Y -> Wđi đến dòng tiếp theo,N
-> tiếp tục đến cột tiếp theo
            SUB AX,PADDLE_RIGHT_X
            CMP AX,PADDLE_WIDTH
            JNG DRAW_PADDLE_RIGHT_HORIZONTAL

            MOV CX,PADDLE_RIGHT_X       ;thanh ghi CX quay trở lại cột ban đầu
            INC DX                      ;tiến lên một dòng

            MOV AX,DX                   ;DX - PADDLE_RIGHT_Y > PADDLE_HEIGHT (Y -> thoát khỏi thủ tục này,N
-> tiếp tục đến dòng tiếp theo
            SUB AX,PADDLE_RIGHT_Y
            CMP AX,PADDLE_HEIGHT
            JNG DRAW_PADDLE_RIGHT_HORIZONTAL
```

Đoạn mã bạn đã cung cấp là một thủ tục (procedure) có tên là DRAW_PADDLES, và nó có tác dụng vẽ thanh đập (paddle) trên màn hình. Cụ thể, đoạn mã này vẽ thanh đập bên phải (PADDLE_RIGHT).

Quá trình vẽ thanh đập bên phải được thực hiện theo hai bước tương tự như quá trình vẽ thanh đập bên trái:

1. Vẽ các điểm trên hàng ngang của thanh đập:

- Đầu tiên, thanh ghi CX được đặt bằng giá trị của biến PADDLE_RIGHT_X, đại diện cho cột ban đầu của thanh đập bên phải.

- Thủ tục sử dụng lệnh INT 10h với AX=0Ch để cấu hình viết một pixel trên màn hình với màu trắng.

- Sau đó, thanh ghi CX tăng lên một đơn vị (INC CX) để chuyển đến cột tiếp theo.

- Tiếp theo, tính toán AX bằng cách lấy giá trị hiện tại của CX trừ đi PADDLE_RIGHT_X và so sánh với PADDLE_WIDTH để xác định liệu đã vẽ đủ chiều dài của thanh đập hay chưa.

- Nếu AX (CX - PADDLE_RIGHT_X) nhỏ hơn hoặc bằng PADDLE_WIDTH (độ rộng của thanh đập), quá trình vẽ trên hàng ngang vẫn chưa hoàn thành, do đó quay lại bước vẽ tiếp theo (JNG DRAW_PADDLE_RIGHT_HORIZONTAL).

2. Vẽ các điểm trên dòng dọc của thanh đập:

- Thủ tục tăng giá trị của thanh ghi DX lên một đơn vị (INC DX), để chuyển đến dòng tiếp theo của thanh đập.

- Tiếp theo, tính toán AX bằng cách lấy giá trị hiện tại của DX trừ đi PADDLE_RIGHT_Y và so sánh với PADDLE_HEIGHT để xác định liệu đã vẽ đủ chiều cao của thanh đập hay chưa.

- Nếu AX (DX - PADDLE_RIGHT_Y) nhỏ hơn hoặc bằng PADDLE_HEIGHT, quá trình vẽ trên dòng dọc vẫn chưa hoàn thành, do đó quay lại bước vẽ tiếp theo (JNG DRAW_PADDLE_RIGHT_HORIZONTAL).

Sau khi hoàn thành cả hai bước, thủ tục DRAW_PADDLES kết thúc và trở về vị trí gọi thủ tục ban đầu.

4. Di chuyển và kiểm tra va chạm giữa các đối tượng

a. Di chuyển quả bóng

```
MOVE_BALL PROC NEAR                 ;xử lý chuyển động của quả bóng

;   Di chuyển quả bóng theo chiều ngang
        MOV AX,BALL_VELOCITY_X
        ADD BALL_X,AX

;       Kiểm tra xem bóng đã qua đường biên trái chưa (BALL_X < 0 + WINDOW_BOUNDS)
;       If is colliding, restart its position
        MOV AX,WINDOW_BOUNDS
          CMP  BALL_X,AX                        ;BALL_X được so sánh với biên trái của màn hình (0 +
WINDOW_BOUNDS)
        JL GIVE_POINT_TO_PLAYER_TWO      ;nếu nhỏ hơn, cho người chơi hai 1 điểm và đặt lại vị trí bóng

;       Kiểm tra xem bóng đã qua đường biên phải chưa (BALL_X > WINDOW_WIDTH - BALL_SIZE  - WINDOW_BOUNDS)
;       If is colliding, restart its position
        MOV AX,WINDOW_WIDTH
        SUB AX,BALL_SIZE
        SUB AX,WINDOW_BOUNDS
       CMP BALL_X,AX                ;BALL_X được so sánh với biên phải của màn hình (BALL_X > WINDOW_WIDTH
- BALL_SIZE  - WINDOW_BOUNDS)
        JG GIVE_POINT_TO_PLAYER_ONE     ;nếu lớn hơn, cho người chơi một 1 điểm và đặt lại vị trí bóng
        JMP MOVE_BALL_VERTICALLY

        GIVE_POINT_TO_PLAYER_ONE:        ;cho người chơi một 1 điểm và đặt lại vị trí bóng
            INC PLAYER_ONE_POINTS       ;tăng điểm của người chơi 1
            CALL RESET_BALL_POSITION     ;đặt lại vị trí bóng vào giữa màn hình

            CALL UPDATE_TEXT_PLAYER_ONE_POINTS ;cập nhật điểm cho người chơi 1

            CMP PLAYER_ONE_POINTS,05h   ;kiểm tra xem người chơi đã đạt 5 điểm chưa
            JGE GAME_OVER                ;nếu điểm người chơi này là 5 hoặc nhiều hơn, trò chơi kết thúc
            RET

        GIVE_POINT_TO_PLAYER_TWO:        ;cho người chơi hai 1 điểm và đặt lại vị trí bóng
            INC PLAYER_TWO_POINTS      ;tăng điểm của người chơi 2
            CALL RESET_BALL_POSITION     ;đặt lại vị trí bóng vào giữa màn hình

            CALL UPDATE_TEXT_PLAYER_TWO_POINTS ;cập nhật điểm cho người chơi 2

            CMP PLAYER_TWO_POINTS,05h  ;kiểm tra xem người chơi đã đạt 5 điểm chưa
            JGE GAME_OVER                ;nếu điểm người chơi này là 5 hoặc nhiều hơn, trò chơi kết thúc
            RET

        GAME_OVER:                       ;có người chơi đạt 5 điểm
            CMP PLAYER_ONE_POINTS,05h    ;kiểm tra xem người chơi nào có 5 điểm trở lên
            JNL WINNER_IS_PLAYER_ONE     ;nếu người chơi 1 có ít nhất 5 điểm là người chiến thắng
            JMP WINNER_IS_PLAYER_TWO     ;nếu không thì người chơi thứ hai là người chiến thắng

            WINNER_IS_PLAYER_ONE:
                MOV WINNER_INDEX,01h     ;cập nhật chỉ số người chiến thắng với chỉ số người chơi một
                JMP CONTINUE_GAME_OVER
            WINNER_IS_PLAYER_TWO:
                MOV WINNER_INDEX,02h     ;cập nhật chỉ số người chiến thắng với chỉ số người chơi hai
```

```
                JMP CONTINUE_GAME_OVER

            CONTINUE_GAME_OVER:
                MOV PLAYER_ONE_POINTS,00h   ;reset điểm người chơi 1
                MOV PLAYER_TWO_POINTS,00h   ;reset điểm người chơi 2
                CALL UPDATE_TEXT_PLAYER_ONE_POINTS
                CALL UPDATE_TEXT_PLAYER_TWO_POINTS
                MOV GAME_ACTIVE,00h            ;dừng trò chơi
                RET

;   Di chuyển quả bóng theo chiều dọc
        MOVE_BALL_VERTICALLY:
            MOV AX,BALL_VELOCITY_Y
            ADD BALL_Y,AX

;       Kiểm tra xem bóng đã vượt qua biên trên chưa (BALL_Y < 0 + WINDOW_BOUNDS)
;       If is colliding, reverse the velocity in Y
        MOV AX,WINDOW_BOUNDS
        CMP BALL_Y,AX                   ;BALL_Y được so sánh với biên trên của màn hình (0 + WINDOW_BOUNDS)
        JL NEG_VELOCITY_Y                ;nếu nhỏ hơn thì hoàn nguyên vận tốc trong Y

;       Kiểm tra xem bóng đã vượt qua biên dưới chưa (BALL_Y > WINDOW_HEIGHT - BALL_SIZE - WINDOW_BOUNDS)
;       If is colliding, reverse the velocity in Y
        MOV AX,WINDOW_HEIGHT
        SUB AX,BALL_SIZE
        SUB AX,WINDOW_BOUNDS
         CMP BALL_Y,AX                      ;BALL_Y được so sánh với biên dưới của màn hình (BALL_Y >
WINDOW_HEIGHT - BALL_SIZE - WINDOW_BOUNDS)
        JG NEG_VELOCITY_Y               ;nếu lớn hơn thì hoàn nguyên vận tốc trong Y

;       Kiểm tra xem quả bóng có va chạm với thanh bên phải không
        ; maxx1 > minx2 && minx1 < maxx2 && maxy1 > miny2 && miny1 < maxy2
        ; BALL_X + BALL_SIZE > PADDLE_RIGHT_X && BALL_X < PADDLE_RIGHT_X + PADDLE_WIDTH
        ; && BALL_Y + BALL_SIZE > PADDLE_RIGHT_Y && BALL_Y < PADDLE_RIGHT_Y + PADDLE_HEIGHT

        MOV AX,BALL_X
        ADD AX,BALL_SIZE
        CMP AX,PADDLE_RIGHT_X
        JNG CHECK_COLLISION_WITH_LEFT_PADDLE  ;nếu không có va chạm, hãy kiểm tra va chạm thanh bên trái

        MOV AX,PADDLE_RIGHT_X
        ADD AX,PADDLE_WIDTH
        CMP BALL_X,AX
        JNL CHECK_COLLISION_WITH_LEFT_PADDLE  ;nếu không có va chạm, hãy kiểm tra va chạm thanh bên trái

        MOV AX,BALL_Y
        ADD AX,BALL_SIZE
        CMP AX,PADDLE_RIGHT_Y
        JNG CHECK_COLLISION_WITH_LEFT_PADDLE  ;nếu không có va chạm, hãy kiểm tra va chạm thanh bên trái

        MOV AX,PADDLE_RIGHT_Y
        ADD AX,PADDLE_HEIGHT
        CMP BALL_Y,AX
        JNL CHECK_COLLISION_WITH_LEFT_PADDLE  ;nếu không có va chạm, hãy kiểm tra va chạm thanh bên trái

;       Nếu đến đây, quả bóng đang va chạm với thanh bên phải

        JMP NEG_VELOCITY_X

;       Kiểm tra xem quả bóng có va chạm với thanh bên trái không
        CHECK_COLLISION_WITH_LEFT_PADDLE:
        ; maxx1 > minx2 && minx1 < maxx2 && maxy1 > miny2 && miny1 < maxy2
        ; BALL_X + BALL_SIZE > PADDLE_LEFT_X && BALL_X < PADDLE_LEFT_X + PADDLE_WIDTH
        ; && BALL_Y + BALL_SIZE > PADDLE_LEFT_Y && BALL_Y < PADDLE_LEFT_Y + PADDLE_HEIGHT

        MOV AX,BALL_X
        ADD AX,BALL_SIZE
```

```
        CMP AX,PADDLE_LEFT_X
        JNG EXIT_COLLISION_CHECK  ;nếu không có va chạm, dừng lại việc kiểm tra

        MOV AX,PADDLE_LEFT_X
        ADD AX,PADDLE_WIDTH
        CMP BALL_X,AX
        JNL EXIT_COLLISION_CHECK  ;nếu không có va chạm, dừng lại việc kiểm tra

        MOV AX,BALL_Y
        ADD AX,BALL_SIZE
        CMP AX,PADDLE_LEFT_Y
        JNG EXIT_COLLISION_CHECK  ;nếu không có va chạm, dừng lại việc kiểm tra
        MOV AX,PADDLE_LEFT_Y
        ADD AX,PADDLE_HEIGHT
        CMP BALL_Y,AX
        JNL EXIT_COLLISION_CHECK  ;nếu không có va chạm, dừng lại việc kiểm tra

;       Nếu đến đây, quả bóng đang va chạm với thanh bên trái

        JMP NEG_VELOCITY_X

        NEG_VELOCITY_Y:
           NEG BALL_VELOCITY_Y   ;đảo ngược vận tốc theo Y của quả bóng(BALL_VELOCITY_Y = - BALL_VELOCITY_Y)
            RET
        NEG_VELOCITY_X:
            NEG BALL_VELOCITY_X              ;đảo ngược vận tốc ngang của quả bóng
            RET

        EXIT_COLLISION_CHECK:
            RET
    MOVE_BALL ENDP
```

Đoạn code được sử dụng để xử lý chuyển động của quả bóng trong trò chơi.

Quả bóng được di chuyển theo chiều ngang và chiều dọc. Đầu tiên, nó tính toán và cập nhật vị trí của quả bóng dựa trên vận tốc ngang và dọc của nó.

Sau đó, kiểm tra xem quả bóng đã vượt qua các biên của màn hình hay chưa. Nếu quả bóng vượt qua biên trái, nó gọi thủ tục "GIVE_POINT_TO_PLAYER_TWO" để tăng điểm của người chơi hai và đặt lại vị trí của quả bóng vào giữa màn hình. Tương tự, nếu quả bóng vượt qua biên phải, nó gọi thủ tục "GIVE_POINT_TO_PLAYER_ONE" để tăng điểm của người chơi một và đặt lại vị trí của quả bóng.

Sau đó, kiểm tra xem quả bóng có va chạm với thanh bên phải hay thanh bên trái không. Nếu có va chạm với thanh bên phải, ngược vận tốc ngang của quả bóng bằng cách gọi "NEG_VELOCITY_X". Tương tự, nếu có va chạm với thanh bên trái, thủ tục cũng đảo ngược vận tốc ngang của quả bóng.

Cuối cùng, kiểm tra xem quả bóng có vượt qua biên trên hoặc biên dưới của màn hình hay không. Nếu có, nó đảo ngược vận tốc dọc của quả bóng bằng cách gọi "NEG_VELOCITY_Y".

b. Di chuyển của thanh đỡ trái

```
;       chuyển động của thanh đập trái

        ;kiểm tra xem có phím nào đang được nhấn không (nếu không kiểm tra thanh còn lại)
        MOV AH,01h
        INT 16h
```

```
        JZ CHECK_RIGHT_PADDLE_MOVEMENT ;ZF = 1, JZ -> Jump If Zero

        ;kiểm tra phím nào đang được nhấn (AL = ASCII character)
        MOV AH,00h
        INT 16h

        ;nếu nó là 'w' hoặc 'W' di chuyển lên
        CMP AL,77h ;'w'
        JE MOVE_LEFT_PADDLE_UP
        CMP AL,57h ;'W'
        JE MOVE_LEFT_PADDLE_UP

        ;nếu nó là 's' hoặc 'S' di chuyển xuống
        CMP AL,73h ;'s'
        JE MOVE_LEFT_PADDLE_DOWN
        CMP AL,53h ;'S'
        JE MOVE_LEFT_PADDLE_DOWN
        JMP CHECK_RIGHT_PADDLE_MOVEMENT

        MOVE_LEFT_PADDLE_UP:
            MOV AX,PADDLE_VELOCITY
            SUB PADDLE_LEFT_Y,AX

            MOV AX,WINDOW_BOUNDS
            CMP PADDLE_LEFT_Y,AX
            JL FIX_PADDLE_LEFT_TOP_POSITION
            JMP CHECK_RIGHT_PADDLE_MOVEMENT

            FIX_PADDLE_LEFT_TOP_POSITION:
                MOV PADDLE_LEFT_Y,AX
                JMP CHECK_RIGHT_PADDLE_MOVEMENT

        MOVE_LEFT_PADDLE_DOWN:
            MOV AX,PADDLE_VELOCITY
            ADD PADDLE_LEFT_Y,AX
            MOV AX,WINDOW_HEIGHT
            SUB AX,WINDOW_BOUNDS
            SUB AX,PADDLE_HEIGHT
            CMP PADDLE_LEFT_Y,AX
            JG FIX_PADDLE_LEFT_BOTTOM_POSITION
            JMP CHECK_RIGHT_PADDLE_MOVEMENT

            FIX_PADDLE_LEFT_BOTTOM_POSITION:
                MOV PADDLE_LEFT_Y,AX
```

Đoạn mã trên xử lý chuyển động của thanh đập bên trái.

1. Đoạn mã bắt đầu bằng việc kiểm tra xem có phím nào đang được nhấn không bằng cách gọi hàm ngắt INT 16h với AH=01h. Kết quả của việc gọi này sẽ được lưu trữ trong cờ ZF (Zero Flag).

2. Nếu không có phím nào được nhấn (ZF = 1), đoạn mã sẽ nhảy tới nhãn CHECK_RIGHT_PADDLE_MOVEMENT để kiểm tra chuyển động của thanh đập bên phải.

3. Nếu có phím được nhấn, đoạn mã tiếp tục bằng việc kiểm tra phím nào được nhấn bằng cách gọi hàm ngắt INT 16h với AH=00h. Ký tự ASCII của phím được nhấn sẽ được lưu trữ trong thanh ghi AL.

4. Đoạn mã tiếp tục kiểm tra ký tự đã nhận được. Nếu ký tự là 'w' hoặc 'W', thanh đập sẽ di chuyển lên (MOVE_LEFT_PADDLE_UP). Nếu ký tự là 's' hoặc 'S', thanh đập sẽ di chuyển xuống (MOVE_LEFT_PADDLE_DOWN).

5. Sau khi xử lý di chuyển của thanh đập bên trái, đoạn mã nhảy tới nhãn CHECK_RIGHT_PADDLE_MOVEMENT để kiểm tra chuyển động của thanh đập bên phải.

6. MOVE_LEFT_PADDLE_UP: Đoạn mã xử lý di chuyển thanh đập lên bằng cách giảm giá trị của PADDLE_LEFT_Y (vị trí thanh đập bên trái theo trục y) bởi giá trị PADDLE_VELOCITY (tốc độ di chuyển). Sau đó, nó kiểm tra xem vị trí thanh đập bên trái có nằm trong giới hạn cửa sổ không. Nếu không, nó sẽ điều chỉnh lại vị trí cho phù hợp (FIX_PADDLE_LEFT_TOP_POSITION).

7. FIX_PADDLE_LEFT_TOP_POSITION: Đoạn mã này đặt lại vị trí của thanh đập bên trái nếu nó vượt quá giới hạn trên cùng của cửa sổ.

8. MOVE_LEFT_PADDLE_DOWN: Đoạn mã xử lý di chuyển thanh đập xuống bằng cách tăng giá trị của PADDLE_LEFT_Y. Sau đó, nó kiểm tra xem vị trí thanh đập bên trái có vượt quá giới hạn dưới cùng của cửa sổ. Nếu vượt quá, nó sẽ điều chỉnh lại vị trí (FIX_PADDLE_LEFT_BOTTOM_POSITION).

9. FIX_PADDLE_LEFT_BOTTOM_POSITION: Đoạn mã này đặt lại vị trí của thanh đập bên trái nếu nó vượt quá giới hạn dưới cùng của màn chơi.

10. Sau khi xử lý di chuyển của thanh đập bên trái, đoạn mã tiếp tục tới CHECK_RIGHT_PADDLE_MOVEMENT để kiểm tra chuyển động của thanh đập bên phải.

c. Di chuyển của thanh đỡ phải

```
;       chuyển động của thanh đập phải
        CHECK_RIGHT_PADDLE_MOVEMENT:

            ;nếu nó là 'o' hoặc 'O' di chuyển lên
            CMP AL,6Fh ;'o'
            JE MOVE_RIGHT_PADDLE_UP
            CMP AL,4Fh ;'O'
            JE MOVE_RIGHT_PADDLE_UP

            ;nếu nó là 'l' hoặc 'L' di chuyển xuống
            CMP AL,6Ch ;'l'
            JE MOVE_RIGHT_PADDLE_DOWN
            CMP AL,4Ch ;'L'
            JE MOVE_RIGHT_PADDLE_DOWN
            JMP EXIT_PADDLE_MOVEMENT


            MOVE_RIGHT_PADDLE_UP:
                MOV AX,PADDLE_VELOCITY
                SUB PADDLE_RIGHT_Y,AX

                MOV AX,WINDOW_BOUNDS
                CMP PADDLE_RIGHT_Y,AX
```

```
            JL FIX_PADDLE_RIGHT_TOP_POSITION
            JMP EXIT_PADDLE_MOVEMENT

            FIX_PADDLE_RIGHT_TOP_POSITION:
                MOV PADDLE_RIGHT_Y,AX
                JMP EXIT_PADDLE_MOVEMENT

        MOVE_RIGHT_PADDLE_DOWN:
            MOV AX,PADDLE_VELOCITY
            ADD PADDLE_RIGHT_Y,AX
            MOV AX,WINDOW_HEIGHT
            SUB AX,WINDOW_BOUNDS
            SUB AX,PADDLE_HEIGHT
            CMP PADDLE_RIGHT_Y,AX
            JG FIX_PADDLE_RIGHT_BOTTOM_POSITION
            JMP EXIT_PADDLE_MOVEMENT

            FIX_PADDLE_RIGHT_BOTTOM_POSITION:
                MOV PADDLE_RIGHT_Y,AX
                JMP EXIT_PADDLE_MOVEMENT
```

Đoạn mã trên xử lý chuyển động của thanh đập bên phải.

1. Đoạn mã bắt đầu bằng nhãn CHECK_RIGHT_PADDLE_MOVEMENT để kiểm tra chuyển động của thanh đập bên phải sau khi xử lý chuyển động của thanh đập bên trái.

2. Đoạn mã tiếp tục kiểm tra ký tự đã nhận được từ bàn phím. Nếu ký tự là 'o' hoặc 'O', thanh đập bên phải sẽ di chuyển lên (MOVE_RIGHT_PADDLE_UP). Nếu ký tự là 'l' hoặc 'L', thanh đập bên phải sẽ di chuyển xuống (MOVE_RIGHT_PADDLE_DOWN). Nếu không phù hợp với các ký tự này, đoạn mã sẽ nhảy tới nhãn EXIT_PADDLE_MOVEMENT để kết thúc xử lý chuyển động của thanh đập.

3. MOVE_RIGHT_PADDLE_UP: Đoạn mã này xử lý di chuyển thanh đập bên phải lên bằng cách giảm giá trị của PADDLE_RIGHT_Y (vị trí thanh đập bên phải theo trục y) bởi giá trị PADDLE_VELOCITY (tốc độ di chuyển). Sau đó, nó kiểm tra xem vị trí thanh đập bên phải có nằm trong giới hạn màn chơi không. Nếu không, nó sẽ điều chỉnh lại vị trí cho phù hợp (FIX_PADDLE_RIGHT_TOP_POSITION).

4. FIX_PADDLE_RIGHT_TOP_POSITION: Đoạn mã này đặt lại vị trí của thanh đập bên phải nếu vị trí vượt quá giới hạn trên cùng của màn chơi.

5. MOVE_RIGHT_PADDLE_DOWN: Đoạn mã này xử lý di chuyển thanh đập bên phải xuống bằng cách tăng giá trị của PADDLE_RIGHT_Y. Sau đó, nó kiểm tra xem vị trí thanh đập bên phải có vượt quá giới hạn dưới cùng của màn chơi không. Nếu có, nó sẽ điều chỉnh lại vị trí (FIX_PADDLE_RIGHT_BOTTOM_POSITION).

6. FIX_PADDLE_RIGHT_BOTTOM_POSITION: Đoạn mã này đặt lại vị trí của thanh đập bên phải nếu vị trí vượt quá giới hạn dưới cùng của màn chơi.

7. Sau khi xử lý di chuyển của thanh đập bên phải, đoạn mã nhảy tới nhãn EXIT_PADDLE_MOVEMENT để kết thúc xử lý chuyển động của thanh đập.

5. Tính năng in Menu và thông báo

a. In Menu

```
;       Shows the menu title
        MOV AH,02h                      ;set cursor position
        MOV BH,00h                      ;set page number
        MOV DH,04h                      ;set row
        MOV DL,04h                      ;set column
        INT 10h
        MOV AH,09h                      ;WRITE STRING TO STANDARD OUTPUT
        LEA DX,TEXT_MAIN_MENU_TITLE     ;give DX a pointer
        INT 21h                         ;print the string
;       Shows the singleplayer message
        MOV AH,02h                      ;set cursor position
        MOV BH,00h                      ;set page number
        MOV DH,06h                      ;set row
        MOV DL,04h                      ;set column
        INT 10h
        MOV AH,09h                      ;WRITE STRING TO STANDARD OUTPUT
        LEA DX,TEXT_MAIN_MENU_SINGLEPLAYER      ;give DX a pointer
        INT 21h                         ;print the string

;       Shows the multiplayer message
        MOV AH,02h                      ;set cursor position
        MOV BH,00h                      ;set page number
        MOV DH,08h                      ;set row
        MOV DL,04h                      ;set column
        INT 10h

        MOV AH,09h                      ;WRITE STRING TO STANDARD OUTPUT
        LEA DX,TEXT_MAIN_MENU_MULTIPLAYER      ;give DX a pointer
        INT 21h                         ;print the string

;       Shows the exit message
        MOV AH,02h                      ;set cursor position
        MOV BH,00h                      ;set page number
        MOV DH,0Ah                      ;set row
        MOV DL,04h                      ;set column
        INT 10h
        MOV AH,09h                      ;WRITE STRING TO STANDARD OUTPUT
        LEA DX,TEXT_MAIN_MENU_EXIT      ;give DX a pointer
        INT 21h                         ;print the string

        MAIN_MENU_WAIT_FOR_KEY:
;       Waits for a key press
            MOV AH,00h
            INT 16h

;       Check which key was pressed
            CMP AL,'S'
            JE START_SINGLEPLAYER
            CMP AL,'s'
            JE START_SINGLEPLAYER
            CMP AL,'M'
            JE START_MULTIPLAYER
            CMP AL,'m'
            JE START_MULTIPLAYER
            CMP AL,'E'
            JE EXIT_GAME
            CMP AL,'e'
            JE EXIT_GAME
            JMP MAIN_MENU_WAIT_FOR_KEY

        START_SINGLEPLAYER:
            MOV CURRENT_SCENE,01h
            MOV GAME_ACTIVE,01h
            RET

        START_MULTIPLAYER:
            JMP MAIN_MENU_WAIT_FOR_KEY ;TODO
```

```
        EXIT_GAME:
            MOV EXITING_GAME,01h
            RET

    DRAW_MAIN_MENU ENDP
```

Đoạn mã trên có chức năng hiển thị tiêu đề và các lựa chọn của menu chính.

1. Đoạn mã bắt đầu bằng việc thiết lập vị trí con trỏ (cursor position) trên màn hình bằng cách sử dụng hàm ngắt INT 10H với AH=02h. Đoạn mã này được lặp lại cho mỗi lựa chọn trong menu để đặt con trỏ ở vị trí thích hợp trên màn hình.

2. Sau đó, đoạn mã sử dụng hàm ngắt INT 21h với AH=09h để viết chuỗi ra đầu ra tiêu chuẩn (standard output). Để làm điều này, đoạn mã sử dụng địa chỉ của chuỗi tiêu đề hoặc thông điệp được lưu trữ trong thanh ghi DX (LEA DX, TEXT_MAIN_MENU_TITLE). INT 21h sẽ in chuỗi này ra màn hình.

3. Sau khi hiển thị các tiêu đề và thông điệp, đoạn mã chờ người dùng nhấn một phím. Nó sử dụng hàm ngắt INT 16h với AH=00h để chờ và lấy phím được nhấn từ bàn phím.

4. Sau khi lấy phím, đoạn mã kiểm tra xem phím nào đã được nhấn bằng cách so sánh giá trị trong thanh ghi AL với các ký tự tương ứng với lựa chọn. Nếu phím tương ứng với lựa chọn đã được nhấn, nó sẽ thực hiện một hành động tương ứng. Ví dụ: nếu nhấn phím 'S' hoặc 's', nó sẽ nhảy tới nhãn START_SINGLEPLAYER.

5. Nhãn START_SINGLEPLAYER xử lý việc chuyển sang chế độ chơi đơn người (singleplayer) bằng cách đặt giá trị vào các biến tương ứng và sử dụng hướng dẫn RET để quay lại mã gọi.

6. Nhãn START_MULTIPLAYER và nhãn EXIT_GAME cũng có chức năng tương tự, nhưng cần được cài đặt thêm chức năng tương ứng cho chế độ chơi đa người (multiplayer) và thoát trò chơi (exit game).

Qua đoạn mã này, menu chính được hiển thị trên màn hình với tiêu đề và các lựa chọn. Khi người dùng nhấn một phím tương ứng với lựa chọn, mã sẽ xử lý theo hướng dẫn tương ứng và chuyển đổi chế độ chơi hoặc thoát trò chơi tùy thuộc vào lựa chọn của người dùng.

b. In thông báo

```
;       Shows the menu title
        MOV AH,02h                      ;set cursor position
        MOV BH,00h                      ;set page number
        MOV DH,04h                      ;set row
        MOV DL,04h                      ;set column
        INT 10h

        MOV AH,09h                      ;WRITE STRING TO STANDARD OUTPUT
        LEA DX,TEXT_MAIN_MENU_TITLE     ;give DX a pointer
        INT 21h                         ;print the string

;       Shows the singleplayer message
        MOV AH,02h                      ;set cursor position
        MOV BH,00h                      ;set page number
        MOV DH,06h                      ;set row
```

```
        MOV DL,04h                      ;set column
        INT 10h

        MOV AH,09h                      ;WRITE STRING TO STANDARD OUTPUT
        LEA DX,TEXT_MAIN_MENU_SINGLEPLAYER      ;give DX a pointer
        INT 21h                         ;print the string

;       Shows the multiplayer message
        MOV AH,02h                      ;set cursor position
        MOV BH,00h                      ;set page number
        MOV DH,08h                      ;set row
        MOV DL,04h                      ;set column
        INT 10h

        MOV AH,09h                      ;WRITE STRING TO STANDARD OUTPUT
        LEA DX,TEXT_MAIN_MENU_MULTIPLAYER      ;give DX a pointer
        INT 21h                         ;print the string

;       Shows the exit message
        MOV AH,02h                      ;set cursor position
        MOV BH,00h                      ;set page number
        MOV DH,0Ah                      ;set row
        MOV DL,04h                      ;set column
        INT 10h

        MOV AH,09h                      ;WRITE STRING TO STANDARD OUTPUT
        LEA DX,TEXT_MAIN_MENU_EXIT      ;give DX a pointer
        INT 21h                         ;print the string

        MAIN_MENU_WAIT_FOR_KEY:
;       Waits for a key press
            MOV AH,00h
            INT 16h
;       Check whick key was pressed
            CMP AL,'S'
            JE START_SINGLEPLAYER
            CMP AL,'s'
            JE START_SINGLEPLAYER
            CMP AL,'M'
            JE START_MULTIPLAYER
            CMP AL,'m'
            JE START_MULTIPLAYER
            CMP AL,'E'
            JE EXIT_GAME
            CMP AL,'e'
            JE EXIT_GAME
            JMP MAIN_MENU_WAIT_FOR_KEY
        START_SINGLEPLAYER:
            MOV CURRENT_SCENE,01h
            MOV GAME_ACTIVE,01h
            RET
        START_MULTIPLAYER:
            JMP MAIN_MENU_WAIT_FOR_KEY ;TODO

        EXIT_GAME:
            MOV EXITING_GAME,01h
            RET
    DRAW_MAIN_MENU ENDP
```

Đoạn mã trên có chức năng hiển thị menu kết thúc trò chơi và xử lý lựa chọn của người chơi.

1. Đầu tiên, đoạn mã thiết lập vị trí con trỏ (cursor position) bằng cách sử dụng hàm ngắt INT 10h với AH=02h. Nó đặt con trỏ ở vị trí cần thiết trên màn hình.

2. Sau đó, đoạn mã gọi một hàm có tên UPDATE_WINNER_TEXT để cập nhật nội dung chuỗi hiển thị người chiến thắng.

3. Tiếp theo, đoạn mã sử dụng hàm ngắt INT 21h với AH=09h để viết chuỗi ra đầu ra tiêu chuẩn (standard output). Đoạn mã này hiển thị chuỗi thông báo người chiến thắng lên màn hình.

4. Sau đó, đoạn mã tiếp tục hiển thị các thông điệp như "Play Again" và "Main Menu" tương tự như trên. Địa chỉ của các chuỗi được lưu trữ trong thanh ghi DX và sau đó được in ra màn hình.

5. Sau khi hiển thị các thông điệp, đoạn mã chờ người dùng nhấn một phím. Nó sử dụng hàm ngắt INT 16h với AH=00h để chờ và lấy phím được nhấn từ bàn phím.

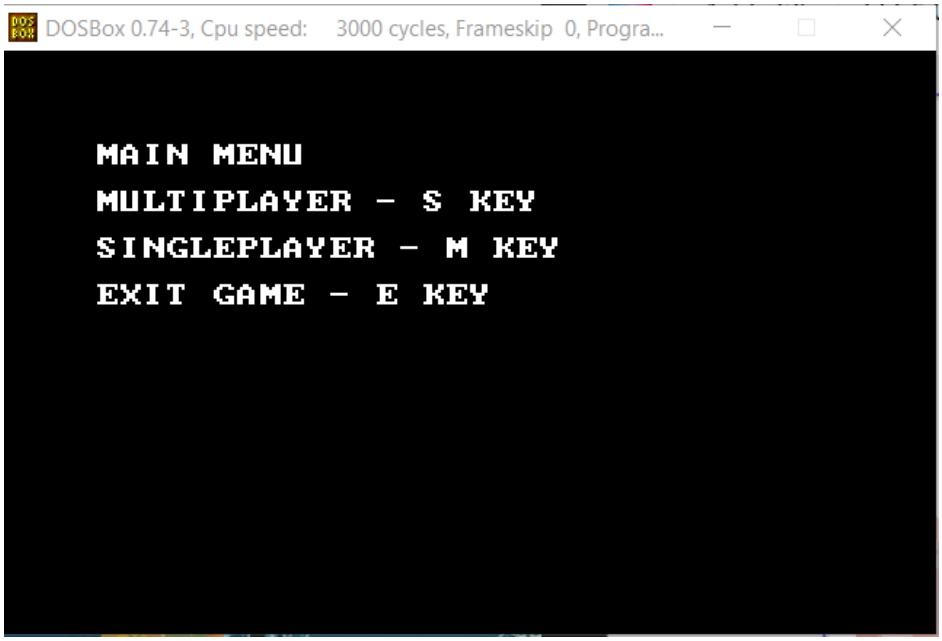
6. Sau khi lấy phím, đoạn mã kiểm tra xem phím nào đã được nhấn bằng cách so sánh giá trị trong thanh ghi AL với các ký tự tương ứng với lựa chọn. Nếu phím tương ứng với lựa chọn đã được nhấn, nó sẽ thực hiện một hành động tương ứng. Ví dụ: nếu nhấn phím 'R' hoặc 'r', nó sẽ nhảy tới nhãn RESTART_GAME.

7. Nhãn RESTART_GAME xử lý việc khởi động lại trò chơi bằng cách đặt giá trị vào biến GAME_ACTIVE và sử dụng hướng dẫn RET để quay lại mã gọi.

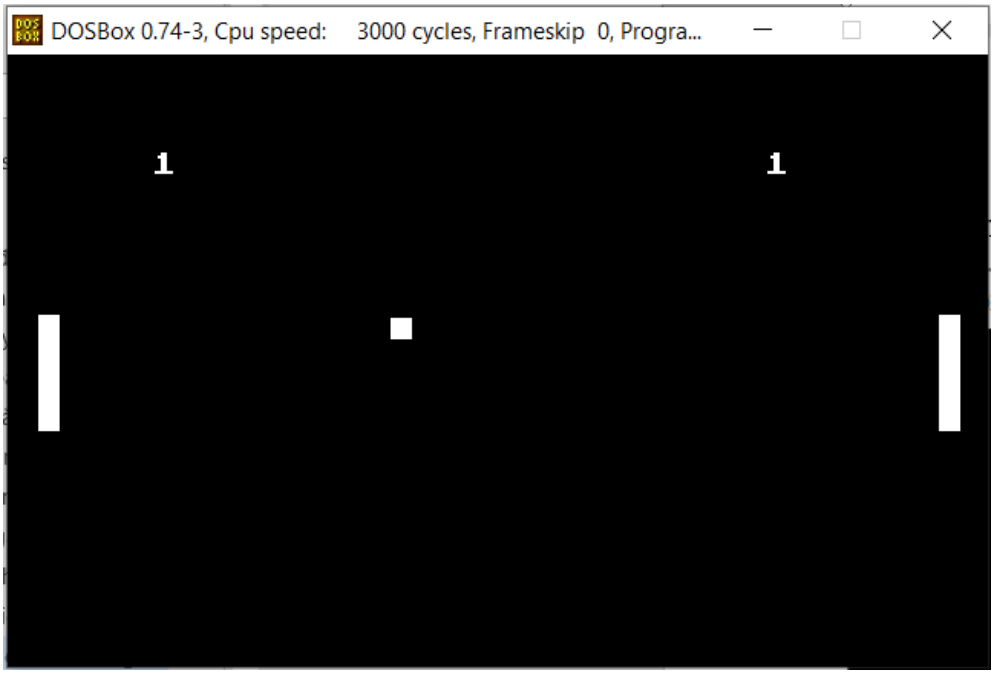
8. Nhãn EXIT_TO_MAIN_MENU xử lý việc thoát về menu chính bằng cách đặt giá trị vào biến GAME_ACTIVE và CURRENT_SCENE và sử dụng hướng dẫn RET để quay lại mã gọi.

V, Giao diện trò chơi

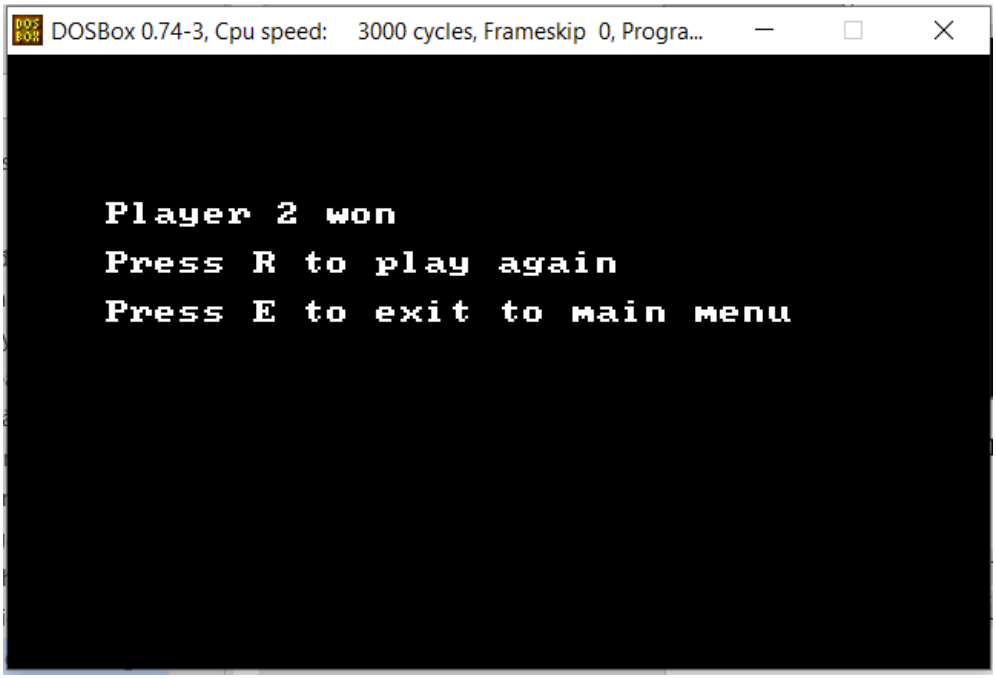
- Màn hình Main Menu của trò chơi



- Giao diện chơi của game



- Màn hình thông báo kết quả trò chơi



VI, Tài liệu tham khảo

Youtube: https://www.youtube.com/playlist?list=PLvpbDCl_H7mfgmEJPl1bTHlH5g-f0kWDM

Github: https://github.com/programmingdimension/8086-Assembly-Pong/blob/master/pong.asm

Google Search

VII, Thành viên và sự đóng góp

STT	Tên	Mã sinh viên	Phần trăm đóng góp (%)
1	Phạm Hoài Nam	B21DCCN554	19
2	Nguyễn Hải Yến	B21DCCN129	24
3	Hoàng Thu Hà	B21DCCN309	19
4	Trần Trọng Việt	B21DCCN791	19
5	Ngô Tuấn Lộc	B21DCCN491	19