HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN I

—————o0o—————

BÀI TẬP LỚN
MÔN KIẾN TRÚC MÁY TÍNH

Đề tài: **“ PROGRAMMING TIC TAC TOE GAME ON EMU8086”**

Giảng viên hướng dẫn:	TS.Trần Tiến Công
Nhóm thực hiện:	Nhóm 5
Niên khóa:	2024–2025
Hệ đào tạo:	Đại học chính quy

Hà Nội, 05/2025

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN I

—————o0o—————

BÀI TẬP LỚN
MÔN KIẾN TRÚC MÁY TÍNH

Đề tài: **“ PROGRAMMING TIC TAC TOE GAME ON EMU8086”**

Giảng viên hướng dẫn: TS.Trần Tiến Công
Sinh viên thực hiện: name
Mã sinh viên: ?
Lớp ??
Hệ đào tạo: Đại học chính quy

Hà Nội, 05/2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Điểm: . . . . . (*Bằng chữ:* . . . . .)

Hà Nội, ngày . . . . . tháng . . . . . năm 20. . . . . .

Giảng viên hướng dẫn

LỜI NÓI ĐẦU

Trong quá trình học tập và nghiên cứu môn Kiến trúc máy tính, việc hiểu và vận dụng được các kiến thức về cấu trúc vi xử lý, thanh ghi, lệnh hợp ngữ, cùng cách chương trình hoạt động ở cấp thấp là rất quan trọng. Việc lập trình trên bộ giả lập Emu8086 giúp sinh viên tiếp cận trực tiếp với kiến trúc của bộ vi xử lý 8086 – một trong những nền tảng căn bản của kiến trúc máy tính hiện đại.

Đề tài "Lập trình trò chơi Tic Tac Toe trên Emu8086"được lựa chọn nhằm giúp sinh viên củng cố và vận dụng kiến thức lý thuyết vào một ứng dụng thực tế. Thông qua việc xây dựng trò chơi cờ ca-rô 3x3 bằng ngôn ngữ Assembly, sinh viên sẽ rèn luyện được khả năng tư duy logic, tổ chức bộ nhớ, điều khiển luồng chương trình, cũng như kỹ năng thao tác với ngắt và hiển thị trên màn hình console.

Báo cáo này trình bày chi tiết quá trình phân tích, thiết kế, cài đặt và kiểm thử chương trình. Mặc dù còn nhiều hạn chế, nhưng bài tập lớn này là một bước tiến quan trọng trong việc tiếp cận thực tế với lập trình hệ thống.

Em xin chân thành cảm ơn thầy đã tận tình giảng dạy và hướng dẫn để em hoàn thành đề tài này.

Mục lục

PHẦN II: LÀM VIỆC NHÓM

Đề tài: "Programming Tic Tac Toe game on Emu8086"

1.Giới thiệu đề tài

1.1. Giới thiệu chung về trò chơi Tic Tac Toe

Tic Tac Toe (hay còn gọi là trò chơi cờ ca-rô 3x3) là một trò chơi chiến lược đơn giản dành cho hai người chơi. Trò chơi thường được chơi trên một lưới gồm 3 hàng và 3 cột (3x3). Hai người chơi lần lượt điền ký hiệu của mình vào các ô trống trên bảng: một người sử dụng ký hiệu "X", người còn lại sử dụng "O".
Mục tiêu của mỗi người chơi là cố gắng tạo được một dãy ba ký hiệu của mình theo hàng ngang, hàng dọc hoặc đường chéo trước đối phương. Nếu bảng được điền đầy mà không có người chơi nào chiến thắng, ván chơi được xem là hòa.
Dù đơn giản, Tic Tac Toe lại là một trò chơi giúp rèn luyện khả năng tư duy logic và chiến lược. Trò chơi này thường được lựa chọn làm đề tài lập trình vì có cấu trúc rõ ràng, dễ cài đặt nhưng vẫn chứa đựng các yếu tố xử lý luồng điều khiển, tương tác với người dùng và kiểm tra điều kiện thắng – những yếu tố cốt lõi trong phát triển phần mềm.

1.2. Mục tiêu của đề tài

Mục tiêu chính của đề tài "Programming Tic Tac Toe game on Emu8086" là:

- Vận dụng kiến thức về hợp ngữ và kiến trúc máy tính để xây dựng một ứng dụng hoàn chỉnh – trò chơi Tic Tac Toe – trên nền tảng bộ vi xử lý 8086.
- Làm quen với môi trường giả lập Emu8086, sử dụng thành thạo các lệnh cơ bản như: hiển thị ký tự (INT 10h), nhập xuất bàn phím (INT 21h), thao tác với thanh ghi và bộ nhớ.
- Thiết kế và triển khai một chương trình điều khiển logic trò chơi, bao gồm:
  - Hiển thị bảng cờ 3x3
  - Nhận lượt chơi từ người dùng
  - Kiểm tra điều kiện thắng, thua, hoặc hòa
  - Luân phiên người chơi
- Củng cố tư duy lập trình cấp thấp thông qua việc xử lý địa chỉ, vòng lặp, rẽ nhánh và thao tác trực tiếp trên phần cứng ảo.
- Rèn luyện kỹ năng phân tích và giải quyết bài toán thực tế bằng ngôn ngữ bậc thấp, tạo nền tảng cho các môn học chuyên sâu sau này như hệ điều hành, lập trình hệ thống, nhúng, và vi điều khiển.

2. Cơ sở lý thuyết

Để xây dựng được trò chơi Tic Tac Toe trên môi trường Emu8086, cần nắm vững một số kiến thức lý thuyết liên quan đến:

2.1.Kiến trúc vi xử lý 8086

- Bộ vi xử lý 8086 là vi xử lý 16-bit với kiến trúc CISC (Complex Instruction Set Computer).
- Có các thanh ghi cơ bản như: AX, BX, CX, DX (dữ liệu), SI, DI (chỉ số), BP, SP (ngăn xếp), CS, DS, ES, SS (đoạn).
- Thanh ghi cờ (FLAGS) phản ánh trạng thái hoạt động sau mỗi lệnh.

Bộ nhớ được chia thành các đoạn, truy cập thông qua cặp thanh ghi segment:offset.

2.2.Ngôn ngữ Assembly

- Là ngôn ngữ lập trình bậc thấp gần với mã máy, sử dụng mnemonic (từ gợi nhớ) để đại diện cho các lệnh của CPU.
- Dựa vào biểu diễn thủ công các thao tác điều khiển, xử lý logic và thao tác với dữ liệu.

2.3.Lập trình trên Emu8086

- Emu8086 là công cụ mô phỏng hoạt động của vi xử lý 8086, hỗ trợ:
  - Viết, dịch, chạy và gỡ lỗi chương trình Assembly.
  - Tương tác với bàn phím, màn hình, bộ nhớ thông qua các ngắt DOS (INT 21h) và ngắt BIOS (INT 10h).
- Một số ngắt thường dùng:
  - INT 10h: thao tác hiển thị ký tự, con trỏ, màu sắc.
  - INT 21h: nhập ký tự từ bàn phím, xử lý I/O cơ bản.

2.4.Nguyên lý hoạt động của trò chơi Tic Tac Toe

- Bảng 3x3 được lưu trữ dưới dạng mảng một chiều hoặc hai chiều.
- Mỗi lượt người chơi cần:
  - Nhập vị trí đánh (1–9 hoặc tọa độ hàng/cột).
  - Kiểm tra tính hợp lệ của ô đó.
  - Cập nhật ký hiệu và kiểm tra điều kiện thắng.
- Chương trình cần luân phiên người chơi và kiểm tra trạng thái kết thúc sau mỗi lượt đi.

3. Phân tích và thiết kế chương trình

3.1. Cấu trúc chương trình

- Khởi tạo: Thiết lập màn hình, vẽ bảng chơi.
- Vòng lặp trò chơi:
  - Nhận vị trí người chơi nhập (qua bàn phím).
  - Kiểm tra hợp lệ.
  - Cập nhật ô
  - Kiểm tra thắng/thua/hòa.
  - Chuyển lượt.
- Kết thúc: Thông báo kết quả, yêu cầu chơi lại hoặc thoát.

3.2. Biểu diễn dữ liệu

- Ma trận 3x3 được biểu diễn bằng mảng 9 phần tử trong bộ nhớ.
- Mỗi ô chứa mã ASCII tương ứng với ký hiệu 'X', 'O', hoặc ' ' (rỗng).
- Các biến điều khiển dùng thanh ghi và bộ nhớ.

4. Cài đặt và giải thích mã nguồn

4.1. Code của chương trình

Link code: **https://github.com/nvbangg/Assembly_TicTacToe/blob/main/TicTacToe.asm**

```
.model small
.stack 100h
.data
    ki_tu db '123456789'           ; mảng kí tự trong các ô (ban đầu là 1-9)
    bang db 13,10, '   |   |   ',13,10, '-----------',13,10, '   |   |   ',13,10, '-----------',13,10, '   |   |   ',13,10,'$' ; giao diện bảng
    pos dw 3,7,11,29,33,37,55,59,63 ; vị trí các 'ki_tu' trong 'bang' để cập nhật
    msg_turn_x db 13,10, 'Nhap vi tri (1-9). Luot X: $'
    msg_turn_o db 13,10, 'Nhap vi tri (1-9). Luot O: $'
    msg_invalid db '    Khong hop le! Thu lai.$'
    msg_x_win db 13,10, 'NGUOI THANG: X$'
    msg_o_win db 13,10, 'NGUOI THANG: O$'
    msg_hoa db 13,10, 'HOA VAN$'
    msg_replay db 13,10, 'Choi lai? (Y/N): $'
    turn db 'X'             ; lưu lượt chơi hiện tại: X hoặc O
    turn_begin db 'O'       ; lưu lượt chơi đầu mỗi vòng
    res db ' ', '$'         ; lưu kết quả: X, O hoặc H (hòa)
    mang_win dw 1,2,3, 4,5,6, 7,8,9, 1,4,7, 2,5,8, 3,6,9, 1,5,9, 3,5,7  ; mảng lưu các bộ 3 vị trí thắng

.code
main proc
    mov ax, @data
    mov ds, ax
    call init
    game_loop:
        call ve_bang
        call nhap_nuoc
        call check_end
        cmp al, 1           ; kiểm tra al=1 (có người thắng/hòa)
        je game_over
        call doi_luot
        jmp game_loop
    game_over:
        call ve_bang
        mov ah, 9           ; dùng hàm 9 để in chuỗi
        mov al, res         ; lấy kết quả (X, O hoặc H)
        lea dx, msg_x_win
        cmp al, 'X'         ; al=X thì sẽ in msg_x_win
        je hien_kq
        lea dx, msg_o_win
        cmp al, 'O'         ; al=O thì sẽ in msg_o_win
```

```
        je hien_kq
        lea dx, msg_hoa     ; sẽ in msg_hoa
    hien_kq:
        int 21h             ; in kết quả
        lea dx, msg_replay  ; in thông báo hỏi chơi lại
        int 21h
        mov ah, 1           ; hàm 1 để nhập ký tự (lưu trong al)
        int 21h
        and al, 0dfh        ; chuyển ký tự thành chữ hoa
        cmp al, 'Y'
        jne exit            ; al=Y thì exit
        call init           ; al!=Y thì reset game
        jmp game_loop
    exit:
        mov ah, 4ch         ; hàm 4Ch thoát chương trình
        int 21h
main endp

init proc
    lea si, ki_tu           ; si trỏ đến mảng kí tự
    mov bl, '1'             ; bl bắt đầu từ ký tự '1'
    mov cx, 9               ; cx=9 để lặp 9 lần
    reset:
        mov [si], bl        ; gán giá trị bl vào ô hiện tại
        inc bl              ; tăng bl lên 1
        inc si              ; trỏ đến ô tiếp theo
        loop reset          ; lặp reset đến khi cx=0
    mov al, 'X'+'O'         ; đổi lượt chơi đầu bằng 'X'+'O' - turn_begin
    sub al, turn_begin
    mov turn_begin, al      ; cập nhật turn_begin và turn hiện tại
    mov turn, al
    ret                     ; trở về hàm gọi
init endp

ve_bang proc
    mov ax, 3               ; hàm 3 của ngắt 10h để xóa màn hình
    int 10h
    mov cx, 9               ; cx=9 để lặp 9 lần
```

```
    mov si, 0               ; đặt si=0 (bắt đầu từ ô đầu tiên)
    update_cell:            ; vòng lặp cập nhật từng ô
        mov bx, si          ; bx=si*2 (vì pos lưu dạng dw cần 2B)
        add bx, bx
        mov di, pos[bx]     ; lấy vị trí cần cập nhật trong chuỗi bang và lưu vào di
        mov al, ki_tu[si]   ; lấy ký tự từ mảng bảng
        mov bang[di], al    ; cập nhật ký tự vào chuỗi hiển thị
        inc si              ; tăng si (chuyển đến ô tiếp theo)
        loop update_cell    ; lặp update_cell đến khi cx=0
    mov ah, 9
    lea dx, bang            ; in bảng
    int 21h
    ret
ve_bang endp

nhap_nuoc proc
    mov ah, 9               ; hàm 9 để in chuỗi
    lea dx, msg_turn_x      ; giả sử là lượt X
    cmp turn, 'X'           ; turn='X' thì in msg_turn_x
    je hien_nhap
    lea dx, msg_turn_o      ; không thì in msg_turn_o
    hien_nhap:
        int 21h             ; in thông báo lượt chơi
        mov ah, 1           ; hàm 1 để nhập ký tự (lưu trong al)
        int 21h
        cmp al, '1'
        jl nhap_sai         ; nếu nhỏ hơn '1' thì nhập sai
        cmp al, '9'
        jg nhap_sai         ; nếu lớn hơn '9' thì nhập sai
        sub al, '1'         ; chuyển '1'-'9' thành 0-8 (chỉ số mảng)
        mov bl, al          ; lưu vào bl
        mov bh, 0           ; bh=0 để bx=bl
        lea si, ki_tu       ; si trỏ đến mảng kí tự
        add si, bx          ; si= địa chỉ ô được chọn
        mov al, [si]        ; lấy giá trị ô và kiểm tra ô đánh chưa (có x hoặc o)
        cmp al, 'X'
        je nhap_sai
        cmp al, 'O'
        je nhap_sai
```

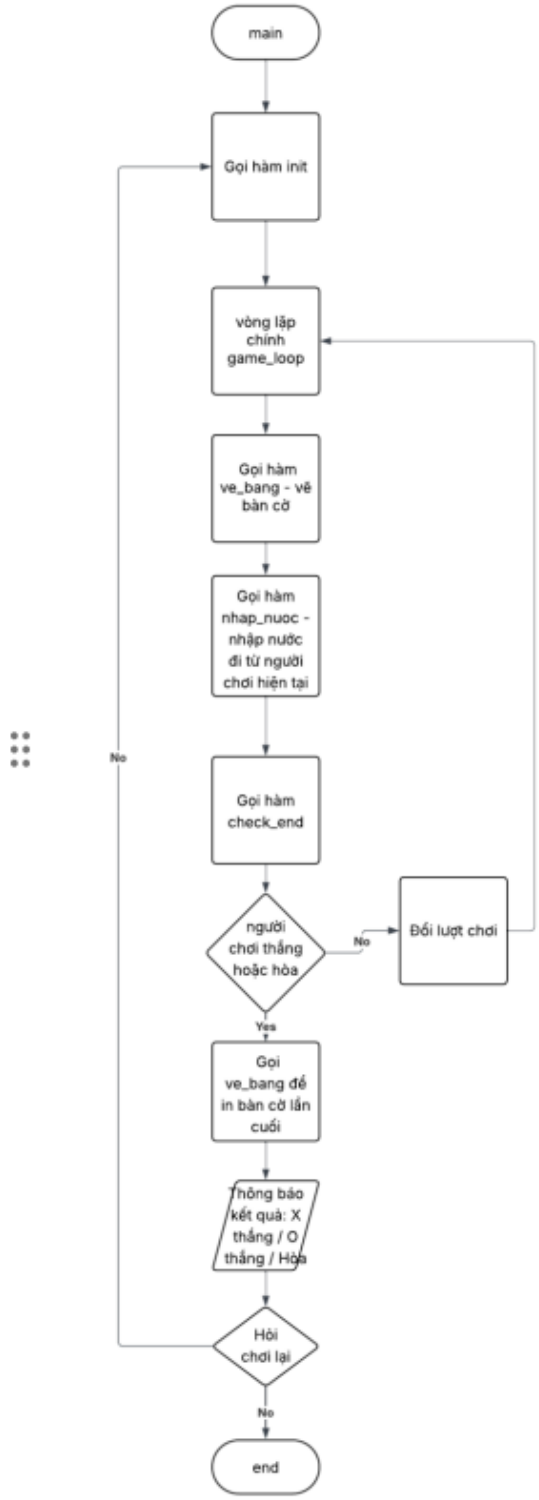
```
        mov al, turn        ; lấy lượt chơi hiện tại (X/O)
        mov [si], al        ; đánh dấu X hoặc O vào ô
        ret
    nhap_sai:
        mov ah, 9           ; hàm 9 để in chuỗi
        lea dx, msg_invalid ; in thông báo nước đi không hợp lệ
        int 21h
        jmp nhap_nuoc       ; quay lại nhập lại
nhap_nuoc endp

doi_luot proc
    mov al, 'X'+'O'         ; đổi lượt chơi bằng 'X'+'O' - turn
    sub al, turn
    mov turn, al
    ret
doi_luot endp

check_end proc
    mov si, 0                   ; si=0 để bắt đầu từ ô đầu tiên
    mov cx, 8                   ; cx=8 để lặp 8 lần
    check_win:
        mov bx, mang_win[si]    ; lấy vị trí thứ 1 trong bộ 3 mang_win
        mov ah, ki_tu[bx-1]     ; lấy ki_tu tại vị trí đó (trừ 1 vì mảng từ 0)
        mov bx, mang_win[si+2]  ; lấy vị trí thứ 2 trong bộ 3 mang_win
        cmp ah, ki_tu[bx-1]     ; so sánh ki_tu tại vị trí 1 và 2
        jnz chua_win            ; nếu khác nhau thì chưa thắng
        mov bx, mang_win[si+4]  ; lấy vị trí thứ 3 trong bộ 3 mang_win
        cmp ah, ki_tu[bx-1]     ; so sánh ki_tu tại vị trí 1 và 3
        jnz chua_win            ; nếu khác nhau thì chưa thắng
        mov res, ah             ; nếu giống nhau thì lưu người thắng (X/O)
        mov al, 1               ; al=1 báo hiệu game kết thúc
        ret
    chua_win:                   ; xử lý khi chưa thắng ở bộ 3 hiện tại
        add si, 6               ; tăng si lên 6 (mỗi bộ 3 chiếm 6 byte trong mang_win)
        loop check_win          ; lặp check_win đến khi cx=0
    lea si, ki_tu               ; si trỏ đến mảng kí tự
    mov cx, 9                   ; cx=9 để lặp 9 lần
    check_hoa:
```

```
        mov al, [si]            ; lấy giá trị ô hiện tại
        cmp al, '9'
        jbe chua_hoa            ; nếu <= '9' thì còn ô chưa đánh -> chưa hòa
        inc si                  ; tăng si (kiểm tra ô tiếp)
        loop check_hoa          ; lặp check_hoa đến khi cx=0
        mov res, 'H'            ; nếu tất cả ô đều đã đánh thì đánh dấu hòa
        mov al, 1               ; al=1 báo hiệu game kết thúc
        ret
    chua_hoa:                   ; xử lý khi chưa hòa
        mov al, 0               ; al=0 (game chưa kết thúc)
        ret
check_end endp
end main
```

Lưu đồ thuật toán:



4.2. Giải thích mã nguồn

Cấu trúc dữ liệu

- ki_tu: Mảng chứa 9 ô, khởi tạo với các ký tự ’1’ đến ’9’ tương ứng các vị trí trên bàn cờ.

- bang: Giao diện bàn cờ được in ra màn hình với ký tự trống và đường gạch ngang.
- pos: Mảng chứa vị trí các ô ki_tu trong chuỗi bang để hiển thị.
- mang_win: Mảng chứa tất cả 8 bộ ba vị trí tạo thành điều kiện thắng trong trò chơi.

Các thông báo

- Các chuỗi thông báo như: yêu cầu nhập nước đi, báo thắng, hòa, thông báo sai vị trí và hỏi người chơi có muốn chơi lại.

Biến điều khiển

- turn: Lượt chơi hiện tại, 'X' hoặc 'O'.
- turn_begin: Lượt bắt đầu game, dùng để đảo lượt khi chơi lại.
- res: Kết quả trò chơi: 'X', 'O', hoặc 'H'.

Khai báo dữ liệu (data segment)

```
.model small
.stack 100h
.data
    ki_tu db '123456789'            ; mảng kí tự trong các ô (ban đầu là 1-9)
    bang db 13,10, '   |   |   ',13,10, '-----------',13,10, '   |   |   ',13,10, '-----------',13,10, '   |   |   ',13,10,'$' ; giao diện bảng tr
    pos dw 3,7,11,29,33,37,55,59,63 ; vị trí các 'ki_tu' trong 'bang' để cập nhật
    msg_turn_x db 13,10, 'Nhap vi tri (1-9). Luot X: $'
    msg_turn_o db 13,10, 'Nhap vi tri (1-9). Luot O: $'
    msg_invalid db '    Khong hop le! Thu lai.$'
    msg_x_win db 13,10, 'NGUOI THANG: X$'
    msg_o_win db 13,10, 'NGUOI THANG: O$'
    msg_hoa db 13,10, 'HOA VAN$'
    msg_replay db 13,10, 'Choi lai? (Y/N): $'
    turn db 'X'             ; lưu lượt chơi hiện tại: X hoặc O
    turn_begin db 'O'       ; lưu lượt chơi đầu mỗi vòng
    res db ' ', '$'         ; lưu kết quả: X, O hoặc H (hòa)
    mang_win dw 1,2,3, 4,5,6, 7,8,9, 1,4,7, 2,5,8, 3,6,9, 1,5,9, 3,5,7  ; mảng lưu các bộ 3 vị trí thắng
```

- Mảng ki_tu đại diện cho 9 ô trên bàn cờ, ban đầu là các số từ '1' đến '9'.
- Khi người chơi đánh dấu, các vị trí này được thay bằng 'X' hoặc 'O'.
- bang db: Đây là giao diện của bàn cờ khi hiển thị trên màn hình.
- Các %c là placeholder (giả định thôi, vì Assembly không hỗ trợ %c), thực tế là bạn gán thủ công tại các vị trí trong chuỗi.
- pos chứa vị trí offset trong chuỗi bang, nơi các ô ki_tu[i] sẽ được gán vào để in ra.
- mang_win chứa tất cả các bộ ba vị trí tạo thành một chiến thắng. Tổng cộng có 8 bộ ba = 24 byte. Ví dụ: 1,2,3 là hàng ngang đầu tiên.
- turn: người chơi hiện tại (X hoặc O)
- turn_begin: dùng để lưu người chơi bắt đầu game
- res: lưu kết quả của game (X/O/H)

Hàm init – Khởi tạo lại trò chơi

```
59  init proc
60      lea si, ki_tu           ; si trỏ đến mảng kí tự
61      mov bl, '1'             ; bl bắt đầu từ ký tự '1'
62      mov cx, 9               ; cx=9 để lặp 9 lần
63      reset:
64          mov [si], bl        ; gán giá trị bl vào ô hiện tại
65          inc bl              ; tăng bl lên 1
66          inc si              ; trỏ đến ô tiếp theo
67          loop reset          ; lặp reset đến khi cx=0
68      mov al, 'X'+'O'         ; đổi lượt chơi đầu bằng 'X'+'O' - turn_begin
69      sub al, turn_begin
70      mov turn_begin, al      ; cập nhật turn_begin và turn hiện tại
71      mov turn, al
72      ret                     ; trở về hàm gọi
73  init endp
```

- Đặt lại ki_tu từ ’1’ đến ’9’ mỗi khi bắt đầu game mới.
- Dùng vòng lặp để gán giá trị al = si + ’1’.
- Đảo người bắt đầu giữa ’X’ và ’O’ qua mỗi ván chơi mới.

Hàm ve_bang – In bàn cờ

```
75  ve_bang proc
76      mov ax, 3               ; hàm 3 của ngắt 10h để xóa màn hình
77      int 10h
78      mov cx, 9               ; cx=9 để lặp 9 lần
79      mov si, 0               ; đặt si=0 (bắt đầu từ ô đầu tiên)
80      update_cell:            ; vòng lặp cập nhật từng ô
81          mov bx, si          ; bx=si*2 (vì pos lưu dạng dw cần 2B)
82          add bx, bx
83          mov di, pos[bx]     ; lấy vị trí cần cập nhật trong chuỗi bang và lưu vào di
84          mov al, ki_tu[si]   ; lấy ký tự từ mảng bảng
85          mov bang[di], al    ; cập nhật ký tự vào chuỗi hiển thị
86          inc si              ; tăng si (chuyển đến ô tiếp theo)
87          loop update_cell    ; lặp update_cell đến khi cx=0
88      mov ah, 9
89      lea dx, bang            ; in bảng
90      int 21h
91      ret
92  ve_bang endp
```

- Duyệt từng ô bàn cờ và gán ký tự ’X’, ’O’ hoặc ’1’–’9’ tương ứng vào vị trí trong chuỗi bang.
- Sau khi gán xong, gọi int 21h với ah = 09h để in chuỗi bang.

Hàm nhap_nuoc – Nhập nước đi từ người chơi

```
nhap_nuoc proc
    mov ah, 9               ; hàm 9 để in chuỗi
    lea dx, msg_turn_x      ; giả sử là lượt X
    cmp turn, 'X'           ; turn='X' thì in msg_turn_x
    je hien_nhap
    lea dx, msg_turn_o      ; không thì in msg_turn_o
    hien_nhap:
        int 21h             ; in thông báo lượt chơi
        mov ah, 1           ; hàm 1 để nhập ký tự (lưu trong al)
        int 21h
        cmp al, '1'
        jl nhap_sai         ; nếu nhỏ hơn '1' thì nhập sai
        cmp al, '9'
        jg nhap_sai         ; nếu lớn hơn '9' thì nhập sai
        sub al, '1'         ; chuyển '1'-'9' thành 0-8 (chỉ số mảng)
        mov bl, al          ; lưu vào bl
        mov bh, 0           ; bh=0 để bx=bl
        lea si, ki_tu       ; si trỏ đến mảng kí tự
        add si, bx          ; si= địa chỉ ô được chọn
        mov al, [si]        ; lấy giá trị ô và kiểm tra ô đánh chưa (có x hoặc o)
        cmp al, 'X'
        je nhap_sai
        cmp al, 'O'
        je nhap_sai
        mov al, turn        ; lấy lượt chơi hiện tại (X/O)
        mov [si], al        ; đánh dấu X hoặc O vào ô
        ret
    nhap_sai:
        mov ah, 9           ; hàm 9 để in chuỗi
        lea dx, msg_invalid ; in thông báo nước đi không hợp lệ
        int 21h
        jmp nhap_nuoc       ; quay lại nhập lại
nhap_nuoc endp
```

- Hiển thị hướng dẫn, nhập phím từ người chơi.
- Kiểm tra phím có trong khoảng ’1’ đến ’9’.
- Kiểm tra ô đã đánh hay chưa.
- Nếu hợp lệ, ghi ký hiệu của người chơi (X/O) vào ô.

```
DOI_LUOT PROC
    MOV  AL, 'X' + 'O'     ; Tổng = 167
    SUB  AL, LUOT_CHOI     ; Hoán đổi X<->O
    MOV  LUOT_CHOI, AL
    RET
DOI_LUOT ENDP
```

- Đổi giữa ’X’ và ’O’ bằng cách al = 155 - turn

Hàm check_end – Kiểm tra thắng hoặc hòa

```
KT_KET_THUC PROC
    XOR  SI, SI          ; SI = 0
    MOV  CX, 8           ; 8 cách thắng cần kiểm tra

KT_THANG:
    MOV  BX, BANG_WIN[SI] ; Lấy vị trí thứ 1
    MOV  AH, BANG[BX-1]
    MOV  BX, BANG_WIN[SI+2] ; Lấy vị trí thứ 2
    CMP  AH, BANG[BX-1]   ; So sánh 1 và 2
    JNZ  KHONG_THANG
    MOV  BX, BANG_WIN[SI+4] ; Lấy vị trí thứ 3
    CMP  AH, BANG[BX-1]   ; So sánh 1 và 3
    JNZ  KHONG_THANG
    MOV  KET_QUA, AH      ; Lưu người thắng
    MOV  AL, 1            ; Báo game kết thúc
    RET

KHONG_THANG:
    ADD  SI, 6            ; Chuyển đến bộ 3 ô tiếp theo
    LOOP KT_THANG

    LEA  SI, BANG         ; Kiểm tra hòa
    MOV  CX, 9
KT_HOA:
    MOV  AL, [SI]         ; Lấy ký tự ô hiện tại
    CMP  AL, '9'          ; Nếu còn số (chưa đánh)
    JBE  CHUA_HOA         ; Thì chưa hòa
    INC  SI
    LOOP KT_HOA
    MOV  KET_QUA, 'D'     ; Đánh dấu hòa (Draw)
    MOV  AL, 1            ; Báo game kết thúc
    RET

CHUA_HOA:
    XOR  AL, AL           ; AL = 0 (Game chưa kết thúc)
    RET
KT_KET_THUC ENDP
```

- Với mỗi tổ hợp 3 vị trí từ mang_win, kiểm tra xem cả 3 đều là 'X' hoặc 'O'.
- Nếu có 3 ký tự giống nhau → thắng → res = 'X' hoặc 'O'.
- Nếu không còn ô nào chưa đánh (tức là không có số '1' đến '9') → hòa.

Vòng lặp chính (game_loop)

```
    game_loop:
        call ve_bang
        call nhap_nuoc
        call check_end
        cmp al, 1           ; kiểm tra al=1 (có người thắng/hòa)
        je game_over
        call doi_luot
        jmp game_loop
    game_over:
        call ve_bang
        mov ah, 9           ; dùng hàm 9 để in chuỗi
        mov al, res         ; lấy kết quả (X, O hoặc H)
        lea dx, msg_x_win
        cmp al, 'X'         ; al=X thì sẽ in msg_x_win
        je hien_kq
        lea dx, msg_o_win
        cmp al, 'O'         ; al=O thì sẽ in msg_o_win
        je hien_kq
        lea dx, msg_hoa     ; sẽ in msg_hoa
```

- Gọi lần lượt vẽ bàn cờ, nhập nước đi, và kiểm tra kết thúc.
- Nếu có người thắng hoặc hòa, nhảy đến game_over.

- Đổi lượt chơi nếu chưa kết thúc.

```
32      game_over:
33          call ve_bang
34          mov ah, 9           ; dùng hàm 9 để in chuỗi
35          mov al, res         ; lấy kết quả (X, O hoặc H)
36          lea dx, msg_x_win
37          cmp al, 'X'         ; al=X thì sẽ in msg_x_win
38          je hien_kq
39          lea dx, msg_o_win
40          cmp al, 'O'         ; al=O thì sẽ in msg_o_win
41          je hien_kq
42          lea dx, msg_hoa     ; sẽ in msg_hoa
43      hien_kq:
44          int 21h             ; in kết quả
45          lea dx, msg_replay  ; in thông báo hỏi chơi lại
46          int 21h
47          mov ah, 1           ; hàm 1 để nhập ký tự (lưu trong al)
48          int 21h
49          and al, 0dfh        ; chuyển ký tự thành chữ hoa
50          cmp al, 'Y'
51          jne exit            ; al=Y thì exit
52          call init           ; al!=Y thì reset game
53          jmp game_loop
```

- Nếu kết thúc, hiển thị bảng + kết quả + hỏi chơi lại.
- Nếu chọn Y, call init và quay lại game_loop.

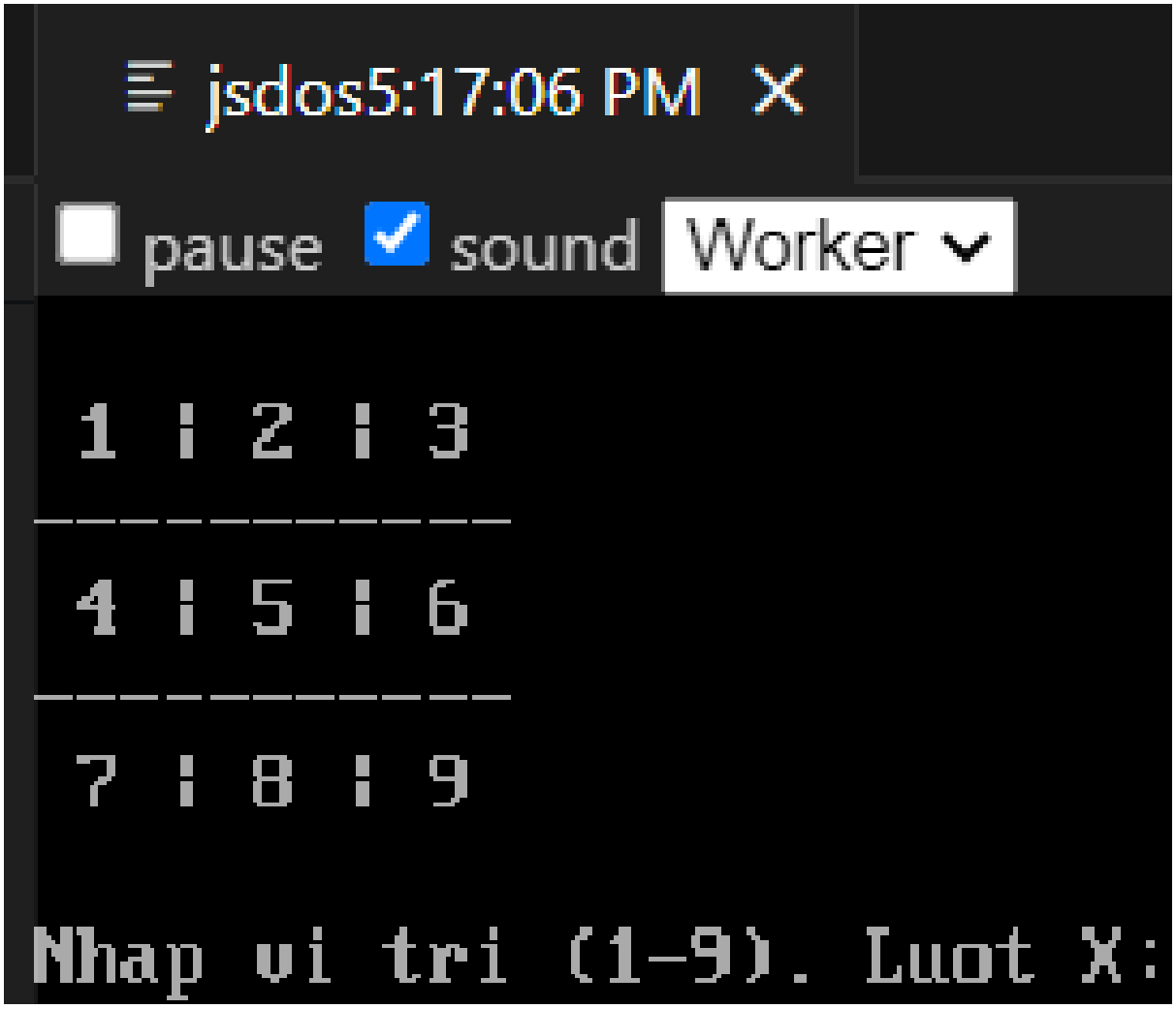
Kết thúc chương trình

```
54      exit:
55          mov ah, 4ch         ; hàm 4Ch thoát chương trình
56          int 21h
57  main endp
```

- Kết thúc chương trình với DOS.

4.3. Demo chạy chương trình

Giao diện của chương trình khi bắt đầu:

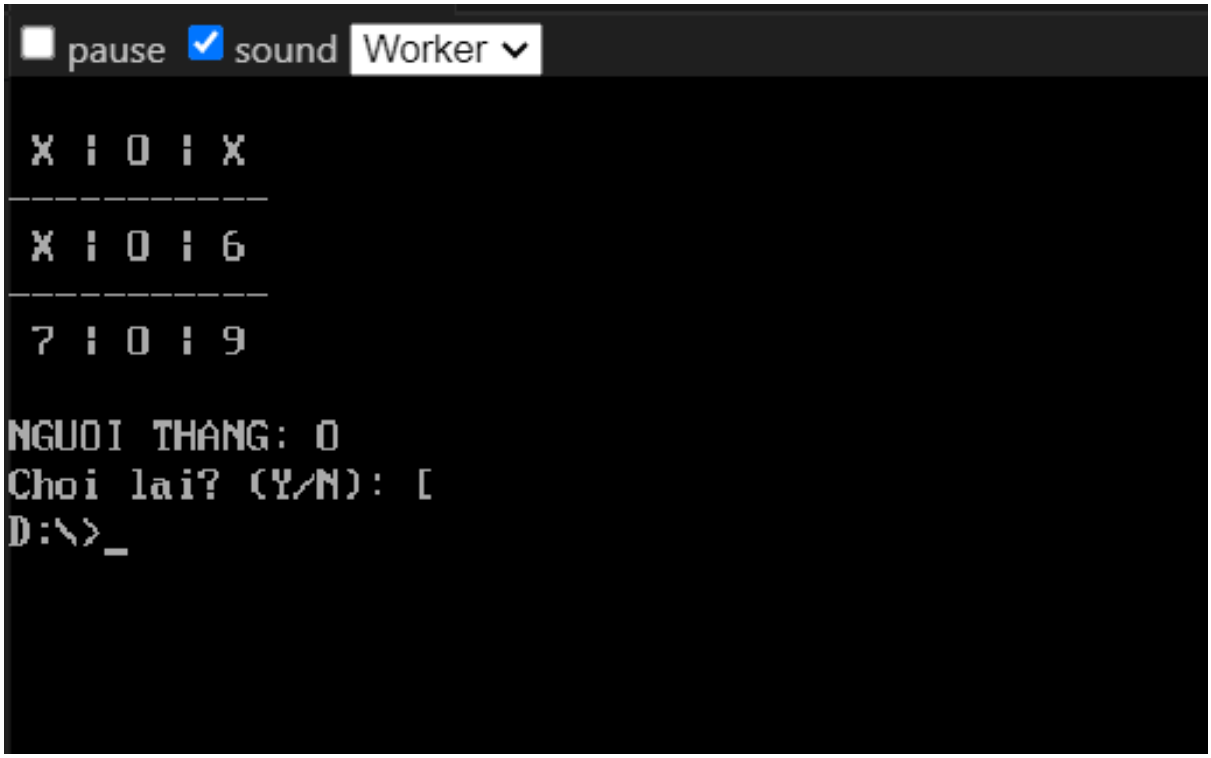


Giao diện khi bắt đầu nhập X nhưng không hợp lệ:

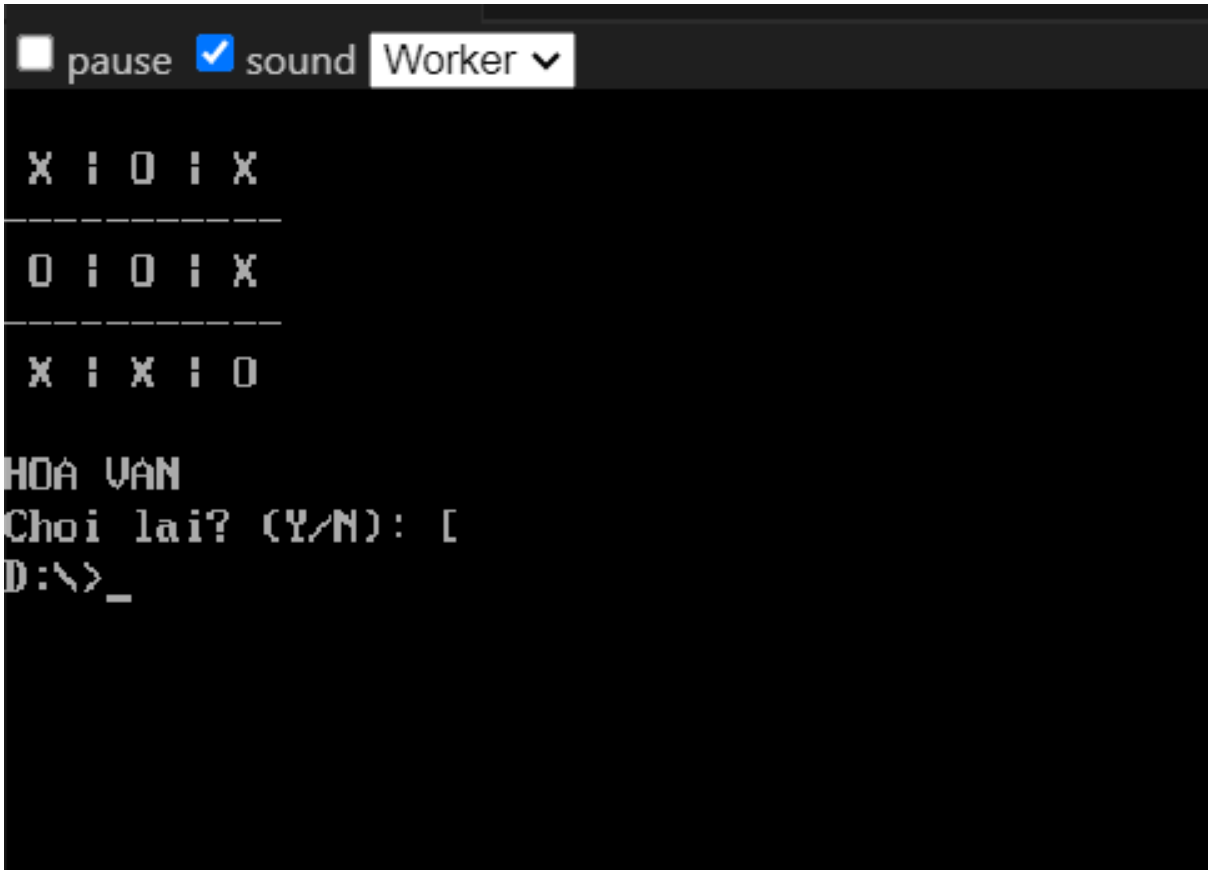
Giao diện khi nhập O không hợp lệ:

Giao diện người chơi thắng là X:

Giao diện người chơi thắng là O:



Giao diện trò chơi hòa nhau:



5. Đánh giá hiệu quả

Chương trình vận hành đúng yêu cầu:

- Hiển thị được bảng trò chơi rõ ràng
- Nhận đầu vào từ người chơi và không cho phép đánh trùng ô
- Cập nhật luân phiên chơi và kiểm tra thắng/hòa
- Hiển thị thông báo thắng hoặc hòa sau khi trò chơi kết thúc

. Tài liệu tham khảo

- Tài liệu Emu8086 User Manual : **https://fr.slideshare.net/slideshow/kin-thc-c-bn-v-lp-trnh-hp-ng-assembly/266393305**
- Emu8086 Tutorial **https://fr.scribd.com/doc/87705752/Emu8086-Tutorial**
- Bài giảng môn Kiến trúc máy tính PTIT **http://docx.com.vn/tai-lieu/giao-trinh-kien-truc-may-tinh-hoc-vien-cong-nghe-buu-chinh-vien-thong-50**
- Code **https://www.ee.nthu.edu.tw/jcliao/mic97/chap08/TICTAC.ASM**