



BTL (Draft)

▼ Giới thiệu đề tài

TIC-TAC-TOE

- Tic-Tac-Toe, hay còn được gọi là X-O, là trò chơi dân gian phổ biến trên toàn thế giới. Trò chơi này đơn giản nhưng không kém phần thú vị, được chơi bởi hai người hoặc hai đội chơi, mỗi người hoặc đội chơi lần lượt đánh dấu ký tự X hoặc O trên một bảng 3x3 ô vuông. Người chơi nào có được ba ô liên tiếp theo hàng ngang, cột dọc hoặc đường chéo trước sẽ chiến thắng.
- Trong lần này, dưới những kiến thức được học trên trường và tìm hiểu, tham khảo bên ngoài, nhóm chúng em sẽ xây dựng trò chơi Tic-Tac-Toe bằng ngôn ngữ lập trình Assembly sử dụng emu8086.



Fact

- Trò chơi được thiết kế dựa trên màn hình kích cỡ **80x25** ký tự (chars)

- Báo cáo nhóm 16 có 5 phần chính
 - Xây dựng một số Macros sử dụng xuyên suốt**
 - Thiết kế giao diện trò chơi**
 - Thiết kế logic trò chơi**
 - Mã nguồn**
 - Tài liệu tham khảo**

▼ Khai báo một số Macros sử dụng xuyên suốt đề tài (Predefined Macros)

▼ Set vị trí con trỏ trên màn hình (Set cursor position) : `go_to_xy row,col`

- Sử dụng hàm ngắt `INT10H/AH=2` để set vị trí con trỏ trên màn hình

◦ `INT 10h / AH = 2`

▪ Input :

- `DH` = row (hàng)
- `DL` = column (cột)
- `BH` = page number (0..7).

```
go_to_xy macro row,col ; set cursor position
    mov dh,row ; set row position
    mov dl,col ; set col position
    mov bh,0 ; page = 0
    mov ah,2
    int 10h
endm
```

▼ Kéo toàn bộ ký tự lên phía trên trên màn hình (scroll up window) : `clear_screen`

- Sử dụng hàm ngắt `INT 10H / AH = 06h & AL = 00h` (Tương đương với `AX = 0600h`) để kéo toàn bộ ký tự lên phía trên màn hình

◦ `INT 10H/ AX = 0600h`

▪ `BH = 00000111b = 07h` (Tham khảo bảng mã màu ở phần tài liệu tham khảo)

- `0000b = 0h` → màu của **background** : **đen**
- `0111b = 7h` → màu phía **trước background** (ví dụ : màu chữ) : **xám**

- **CH, CL** = vị trí hàng và cột của góc trên bên trái của màn hình
 - Ví dụ : **CH** = 00h, **CL** = 00h (**CX** = 0000h)
 - → Hàng có vị trí **00h (0d)** và cột có vị trí **00h (0d)** của góc trên bên trái của màn hình
- **DH, DL** = vị trí hàng và cột của góc dưới bên phải của màn hình
 - Ví dụ : **DH** = 18h, **DL** = 4Fh (**DX** = 184Fh)
 - → Hàng có vị trí **18h (24d)** và cột có vị trí **4Fh (80d)** của góc dưới bên phải của màn hình

```
; clear the screen
clear_screen macro
    mov ax,0600h ; here ah = 06h and al = 00h (scroll entire window)
    mov bh,07h ; here background color = black and fore color = gray
    mov cx,0000h ; window: from row = 00h (0d), col = 00h (0d)
    mov dx,184Fh ; to row = 18h (24d), col = 4Fh (80d)
    int 10h
endm
```

▼ In ra chuỗi ký tự với màu sắc xác định ở vị trí bởi con trỏ trên màn hình (Print colored string) :

print_colored_string string,row,col,color

- Một số hàm ngắt sử dụng
 - **INT 10h/AH=2h** : Set vị trí con trỏ trên màn hình
 - **INT 10h/AH=9h** : In ký tự với màu sắc xác định ở vị trí bởi con trỏ trên màn hình
 - **AL** = Ký tự muốn hiển thị.
 - **BH** = Số trang.
 - **BL** = Thuộc tính (Màu sắc - Tham khảo bảng mã màu ở mục tài liệu tham khảo)
 - **CX** = Số lần in ký tự
 - Xây dựng Macro
 - Thanh ghi **SI** trỏ tới ký tự đầu tiên của của **string**
 - Thanh ghi **DI** dùng để lưu trữ vị trí cột
 - Chú ý rằng thanh ghi **DI** là thanh ghi 16 bit nên **không thể dùng macro go_to_xy trong trường hợp này**
 - Set vị trí con trỏ trên màn hình : (**INT10h/AH=2h**)

```
;set cursor position ;Because of di is 16-bit reg,so we can NOT pass it to go_to_xy macro
mov ah,02h
mov dx,di ; dl = low byte of di
mov dh,row
mov bh,0h
int 10h
```

- Sử dụng lệnh **LODSB** để lưu ký tự trỏ bởi thanh ghi **SI** vào thanh ghi **AL** , sau đó thực hiện thủ tục sau (*)
 - Nếu **AL** ≠ '\$'
 - Thì in ký tự ở **AL**
 - Tăng giá trị **SI** để trỏ tới ký tự tiếp theo
 - Tăng vị trí cột ở thanh ghi **DI**
 - Tiếp tục lặp lại chu trình trên (chu trình *)
 - Nếu **AL** = '\$'
 - Kết thúc thủ tục



Nhược điểm

- Mất khá nhiều thời gian nếu chuỗi ký tự lớn

◦ Mã nguồn

```
;Print string with specific color at cursor position
print_colored_string macro string,row,col,color

    local lop,fin            ; unique label purpose

    mov si, offset string    ; si point to first char of string
    mov di, col              ; di = starting column position note that di is 16-bit reg
    lop:
        ;set cursor position ;Because of di is 16-bit reg,so we can NOT pass it to go_to_xy macro
        mov ah,02h
        mov dx,di            ; dl = low byte of di
        mov dh,row
        mov bh,0h
        int 10h

        lodsb                ; load current character from ds:[si] and increase si (if df = 0)
        cmp al,'$'           ; Check whether we reach terminate character or not
        JE fin              ; If yes ( AL = '$' ) -> jump to label fin
                            ; If no -> print character
        ;print current character stored at al by using int10h/ah=9
        mov ah,09h
        mov bh,0h
        mov bl,color         ; color must be get from emu8086 bit color table
        mov cx,1             ; no.of times to write characters = 1
        int 10h
        inc di               ; increase column position
        jmp lop
    fin:
endm
```

▼ Thiết kế giao diện trò chơi

▼ Thiết kế Logo **Tic-Tac-Toe**

- Một số lưu ý
 - Trò chơi được thiết kế dựa trên màn hình kích cỡ **80x25** ký tự (chars)
 - Vì vậy để hiển thị một chuỗi ký tự (**string**) ra giữa màn hình, nhóm chúng em dùng công thức sau để tính vị trí cột của con trỏ :

$$column = \frac{80 - Number\ of\ chars}{2}$$

- Ở phần này nhóm chúng em sử dụng hàm ngắt **INT21H/AH=7** để nhận ký tự từ bàn phím người chơi, tuy nhiên **ký tự sẽ không hiển thị trên màn hình** (Input without echo)

```
;INT 21h / AH=7 - character input without echo to AL.
;if there is no character in the keyboard buffer
;the function waits until any key is pressed.
mov ah, 7
int 21h
```

• Các bước thực hiện

- Xây dựng Logo
 - Sử dụng ký tự in được ASCII để xây dựng logo **TIC-TAC-TOE**, 2 mã ký được sử dụng:
 - **04h(4d)** → ♦ (EOT)
 - **20h(32d)** → Khoảng trống (space)

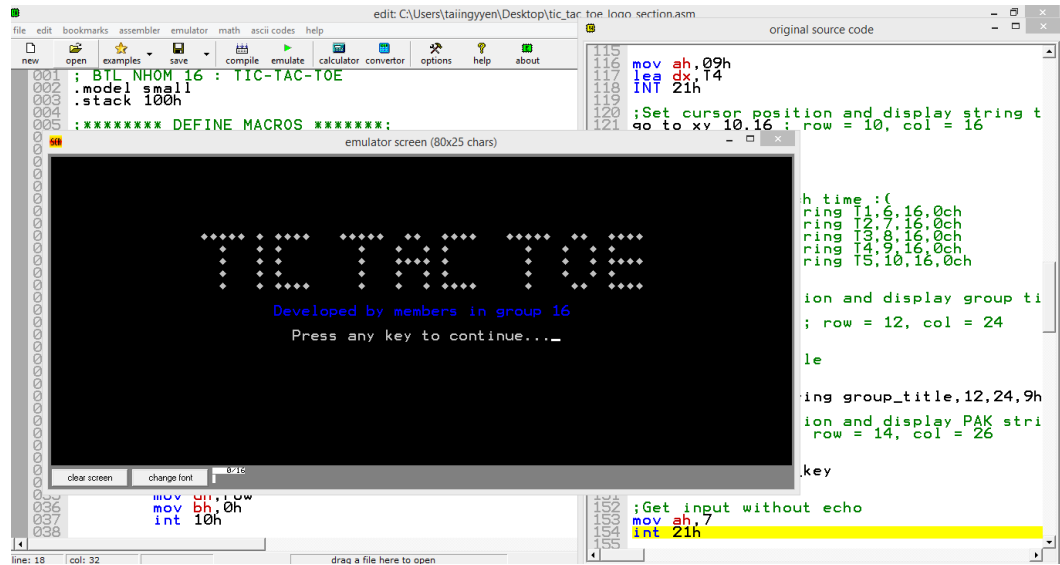
```
; STRING FOR LOGO SECTION
;LOGO TIC-TAC-TOE
T1 DB 4, 4, 4, 4, 4, 32, 4, 32, 4, 4, 4, 4, 32, 32, 32, 4, 4, 4, 4, 32, 32, 4, 4, 32, 32,
T2 DB 32, 32, 4, 32, 32, 32, 4, 32, 4, 32, 32, 32, 32, 32, 32, 32, 4, 32, 32, 32, 4, 32, 32, 4, 32
```

```

T3 DB 32, 32, 4, 32, 32, 32, 4, 32, 4, 32, 32, 32, 32, 32, 32, 32, 4, 32, 32, 32, 4, 4, 4, 4, 32
T4 DB 32, 32, 4, 32, 32, 32, 4, 32, 4, 32, 32, 32, 32, 32, 32, 32, 4, 32, 32, 32, 4, 32, 32, 4, 32
T5 DB 32, 32, 4, 32, 32, 32, 4, 32, 4, 4, 4, 32, 32, 32, 32, 32, 4, 32, 32, 32, 4, 32, 32, 4, 32
;String for group title
group_title db "Developed by members in group 16$" ; 32 chars
;"Press any key to continue" (PAK) string
press_any_key db "Press any key to continue...$" ; 28 chars

```

- Lần lượt hiện thị các xâu ký tự (string), thu được kết quả



- Mã Nguồn

```

;***** DEFINE MACROS *****;
; set cursor position
go_to_xy macro row,col
    mov dh,row ; set row-position cursor
    mov dl,col ; set col-position cursor
    mov bh,0 ; set page number default value is 0
    mov ah,2
    int 10h
endm

; clear the screen
clear_screen macro
    mov ax,0600h ; here ah = 06h and al = 00h (scroll entire window)
    mov bh,07h ; here background color = black (0) and fore color = gray(7)
    mov cx,0000h ; scroll from row = 00h (0d), col = 00h (0d)
    mov dx,184Fh ; to row = 18h (24d), col = 4Fh (80d)
    int 10h
endm

;Print string with specific color at cursor position
print_colored_string macro string,row,col,color

    local lop,fin ; unique label purpose

    mov si, offset string ; si point to first char of string
    mov di, col ; di = starting column position note that di is 16-bit reg
    lop:
        ;set cursor position ;Because of di is 16-bit reg,so we can NOT pass it to go_to_xy macro
        mov ah,02h
        mov dx,di ; dl = low byte of di
        mov dh,row
        mov bh,0h
        int 10h

        lodsb ; load current character from ds:[si] and increse si (if df = 0)
        cmp al,'$' ; Check whether we reach terminate character or not
        JE fin ; If yes ( AL = '$' ) -> jump to label fin
                ; If no print character

        ;print current character stored at al by using int10h/ah=9
        mov ah,09h
        mov bh,0h
        mov bl,color ; color must be get from emu0086 bit color table

```

```

        mov cx,1          ; no.of times to write characters = 1
        int 10h
        inc di            ; increase column position
        jmp lop
fin:
endm

;***** DEFINE DATA SEGMENT *****;
.data ;
;Global string
; "Press any key to continue" string
press_any_key db "Press any key to continue...$" ; 28 chars

;STRING FOR LOGO SECTION
;LOGO TIC-TAC-TOE
T1 DB 4, 4, 4, 4, 4, 32, 4, 32, 4, 4, 4, 4, 32, 32, 32, 4, 4, 4, 4, 32, 32, 4, 4, 32, 3
T2 DB 32, 32, 4, 32, 32, 32, 4, 32, 4, 32, 32, 32, 32, 32, 32, 32, 4, 32, 32, 32, 4, 32, 32, 4,
T3 DB 32, 32, 4, 32, 32, 32, 4, 32, 4, 32, 32, 32, 32, 32, 32, 32, 4, 32, 32, 32, 4, 4, 4, 4,
T4 DB 32, 32, 4, 32, 32, 32, 4, 32, 4, 32, 32, 32, 32, 32, 32, 32, 4, 32, 32, 32, 4, 32, 32, 4,
T5 DB 32, 32, 4, 32, 32, 32, 4, 32, 4, 4, 4, 4, 32, 32, 32, 32, 32, 4, 32, 32, 32, 4, 32, 32, 4,
;String for group title
group_title db "Developed by members in group 16$" ; 32 chars
;***** DEFINE CODE SEGMENT *****;
.code
MAIN proc
; assign base address of data segment to ds register
mov ax,@data
mov ds,ax

;**** Display logo TIC-TAC-TOE (Hien thi logo TIC-TAC-TOE) ****;
LOGO:
;Set cursor position and display string t1
go_to_xy 6,16 ; row = 6, col = 16

mov ah,09h
lea dx,T1
INT 21h

;Set cursor position and display string t2
go_to_xy 7,16 ; row = 7, col = 16

mov ah,09h
lea dx,T2
INT 21h

;Set cursor position and display string t3
go_to_xy 8,16 ; row = 8, col = 16

mov ah,09h
lea dx,T3
INT 21h

;Set cursor position and display string t4
go_to_xy 9,16 ; row = 9, col = 16

mov ah,09h
lea dx,T4
INT 21h

;Set cursor position and display string t5
go_to_xy 10,16 ; row = 10, col = 16

mov ah,09h
lea dx,T5
INT 21h

;TEST -> Too much time :(
;print_colored_string T1,6,16,0ch
;print_colored_string T2,7,16,0ch
;print_colored_string T3,8,16,0ch
;print_colored_string T4,9,16,0ch
;print_colored_string T5,10,16,0ch

;Set cursor position and display group title ; col = (80 - no.of chars)/2 = (80-32)/2 = 24
;
;go_to_xy 12,24 ; row = 12, col = 24

;mov ah,09h
;lea dx,group_title
;INT 21h

print_colored_string group_title,12,24,9h

;Set cursor position and display PAK string ; col = (80 - no.of chars)/2 = (80-28)/2 = 26
go_to_xy 14,26 ; row = 14, col = 26

```

```

mov ah,09h
lea dx,press_any_key
INT 21h

;Get input without echo
mov ah,7
int 21h

;Clear the screen
clear_screen

main endp
END

```

▼ Luật chơi & Thiết kế giao diện luật chơi



Luật chơi

- Tic-Tac-Toe là một trò chơi 2 người
- Người chơi 1 sẽ chơi trước, người chơi 2 chơi sau
- Người chơi 1 đánh dấu **X**, người chơi 2 đánh dấu **O**
- Bảng sẽ được đánh dấu bằng các ô ký hiệu từ 1 → 9
- Nhập số 1 → 9 để đánh dấu vào ô trên bảng
- Trò chơi kết thúc nếu 3 hàng, hoặc 3 cột hoặc 3 đường chéo được đánh dấu giống nhau

- Tương tự như thiết kế Logo, lần lượt set vị trí con trỏ và hiện thị các chuỗi ký tự

```

;STRING FOR GAME-RULES SECTION
;Rules
R DB 'Game Rules:$'
R0 DB '0. This is 2 players game.$'
R1 DB '1. Players will take turns.$'
R2 DB '2. Player 1 will start the game.$'
R3 DB '3. Player 1 will set "X" and Player 2 will set "O".$'
R4 DB '4. The board is marked with cell numbers.$'
R5 DB '5. Enter CELL NUMBER to place your mark.$'
R6 DB '6. Set 3 of your marks horizontally, vertically or diagonally to win.$' ; ~70 chars

R7 DB 'Good Luck!',32,':')$'

; Code section
;***** Display RULES (Hien thi luat choi) *****;
RULE:
; Print Rules title with row = 4, col = 4, color = light red (1100b) (0Ch)
print_colored_string R,3,4,0ch

; Can also use
;Set cursor position and display Rules title
;go_to_xy 4,4
;;mov ah,09h
;lea dx,R
;int 21h

;Set cursor position and display R0 ; col = (80-70)/2 = 5
go_to_xy 5,5
mov ah,09h
lea dx,R0
int 21h

;Set cursor position and display R1
go_to_xy 7,5
mov ah,09h
lea dx,R1
int 21h

;Set cursor position and display R2
go_to_xy 9,5
mov ah,09h
lea dx,R2
int 21h

;Set cursor position and display R3
go_to_xy 11,5
mov ah,09h

```

```

lea dx,R3
int 21h

;Set cursor position and display R4
go_to_xy 13,5
mov ah,09h
lea dx,R4
int 21h

;Set cursor position and display R5
go_to_xy 15,5
mov ah,09h
lea dx,R5
int 21h

;Set cursor position and display R6
go_to_xy 17,5
mov ah,09h
lea dx,R6
int 21h

; Print R7 with row = 18, col = 4, color = light red (1110b) (0Eh)
print_colored_string R7,19,4,0Eh;

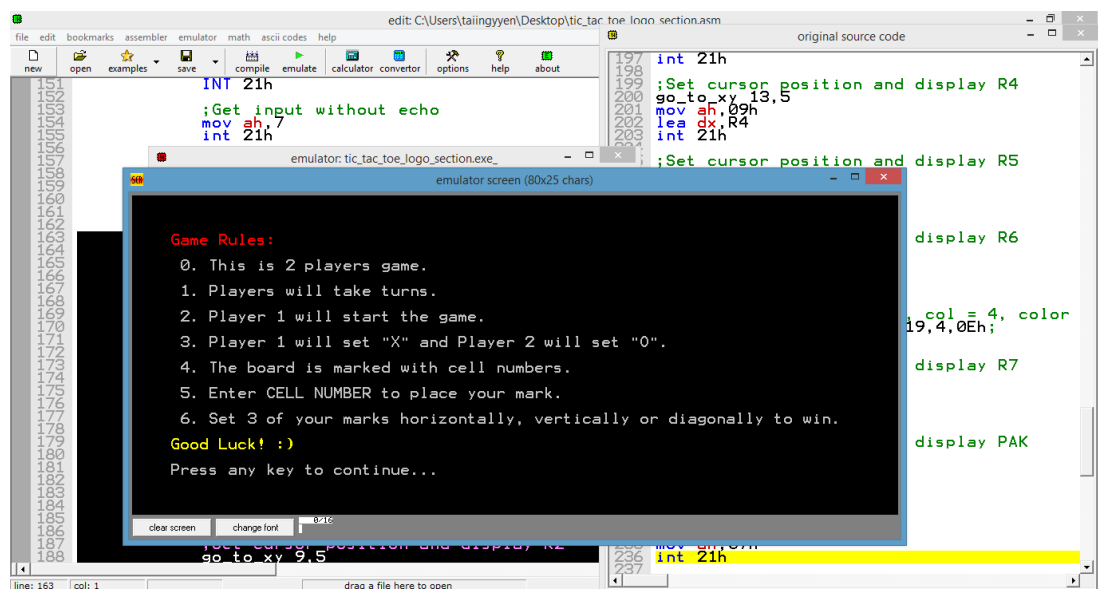
;Can also use
;Set cursor position and display R7
;go_to_xy 19,4
;mov ah,09h
;lea dx,R7
;int 21h

;Set cursor position and display PAK
go_to_xy 21,4
mov ah,09h
lea dx,press_any_key
int 21h

;get input without echo
mov ah,07h
int 21h

```

- Kết quả



▼ Thiết kế giao diện bảng

- Phác họa giao diện bảng


☆ Phác họa giao diện bảng trong Tic Tac Toe

Col Row	34	35	36	37	38	39	40	41	42	43	44
6											
7		1				2				3	
8											
9	—	—	—	—	—	—	—	—	—	—	—
10											
11		4				5				6	
12											
13	—	—	—	—	—	—	—	—	—	—	—
14											
15		7				8				9	
16											
17											
18	<INPUT>										

- Số ký tự trên 1 hàng : **11** ký tự
- Số ký tự trên 1 cột : **13** ký tự
- Xâu ký tự sử dụng để thiết kế bảng Tic Tac Toe

```
;STRING FOR BOARD SECTION
;BOARD LINES
L1 DB ' | | $' ; chars = 11
L2 DB '-----$' ; chars = 11
;Cell numbers
C1 db '1$'
C2 db '2$'
C3 db '3$'
C4 db '4$'
C5 db '5$'
C6 db '6$'
C7 db '7$'
C8 db '8$'
C9 db '9$'
```

- Dựa vào phác họa
 - Lần lượt in các xâu:
 - L1 : ở vị trí con trỏ trên màn hình có row = **6,7,8,10,11,12,14,15,16** col = **34**
 - L2 : ở vị trí con trỏ trên màn hình có row = **9,13** col = **34**
 - Các ô số
 - 1 ở vị trí row = **7**, col = **35**
 - 2 ở vị trí row = **7**, col = **39**
 - 3 ở vị trí row = **7**, col = **43**
 - 4 ở vị trí row = **11**, col = **35**
 - 5 ở vị trí row = **11**, col = **39**
 - 6 ở vị trí row = **11**, col = **43**
 - 7 ở vị trí row = **15**, col = **35**
 - 8 ở vị trí row = **15**, col = **39**

-  ở vị trí row = 15, col = 43

- Mã nguồn

```
.data
;STRING FOR BOARD SECTION
;BOARD LINES
L1 DB ' | | $' ; chars = 11
L2 DB '-----$' ; chars = 11
;Cell numbers
C1 db '1$'
C2 db '2$'
C3 db '3$'
C4 db '4$'
C5 db '5$'
C6 db '6$'
C7 db '7$'
C8 db '8$'
C9 db '9$'

.code
;***** Display BOARD (Hien thi bang) *****;
BOARD:
;Clear the screen
clear_screen

;set cursor position row = 6,7,8 col = 34 & display string : L1 : ' | | $'
go_to_xy 6,34

mov ah,09h
lea dx,L1
int 21h

go_to_xy 7,34

mov ah,09h
lea dx,L1
int 21h

go_to_xy 8,34

mov ah,09h
lea dx,L1
int 21h

;Display cell 1,2,3
;display cell 1 at row = 7, col = 35
go_to_xy 7,35
mov ah,09h
lea dx,c1
int 21h

;display cell 2 at row = 7, col = 39
go_to_xy 7,39
mov ah,09h
lea dx,c2
int 21h

;display cell 3 at row = 7, col = 43
go_to_xy 7,43
mov ah,09h
lea dx,c3
int 21h

;set cursor position row = 9 col = 34 & display string L2 : '-----$'
go_to_xy 9,34

mov ah,09h
lea dx,L2
int 21h

;set cursor position row = 10,11,12 col = 34 & display string : L1 : ' | | $'
go_to_xy 10,34

mov ah,09h
lea dx,L1
int 21h

go_to_xy 11,34

mov ah,09h
lea dx,L1
int 21h

go_to_xy 12,34
```

```

mov ah,09h
lea dx,L1
int 21h

;Display cell 4,5,6
;display cell 4 at row = 11, col = 35
go_to_xy 11,35

mov ah,09h
lea dx,c4
int 21h

;display cell 5 at row = 11, col = 39
go_to_xy 11,39

mov ah,09h
lea dx,c5
int 21h

;display cell 6 at row = 11, col = 43
go_to_xy 11,43

mov ah,09h
lea dx,c6
int 21h

;set cursor position row = 13 col = 34 & display string L2 : '-----$'
go_to_xy 13,34

mov ah,09h
lea dx,l2
int 21h

;set cursor position row = 14,15,16 col = 34 & display string : L1 : ' | | $'
go_to_xy 14,34

mov ah,09h
lea dx,L1
int 21h

go_to_xy 15,34

mov ah,09h
lea dx,L1
int 21h

go_to_xy 16,34

mov ah,09h
lea dx,L1
int 21h
;Display cell 7,8,9
;display cell 7 at row = 15, col = 35
go_to_xy 15,35

mov ah,09h
lea dx,c7
int 21h

;display cell 8 at row = 15, col = 39
go_to_xy 15,39

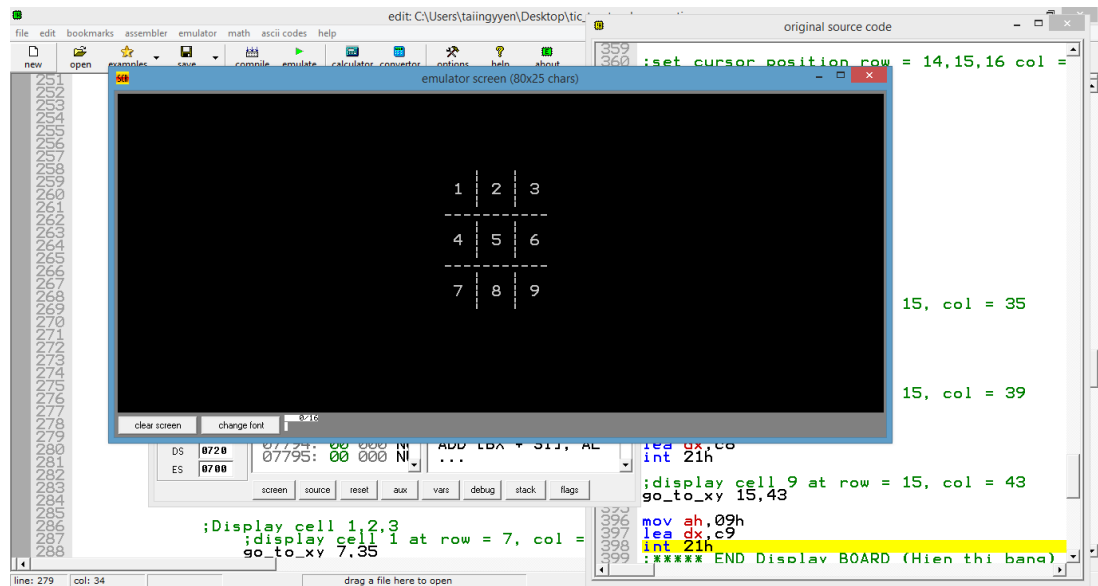
mov ah,09h
lea dx,c8
int 21h

;display cell 9 at row = 15, col = 43
go_to_xy 15,43

mov ah,09h
lea dx,c9
int 21h
;***** END Display BOARD (Hien thi bang) *****;

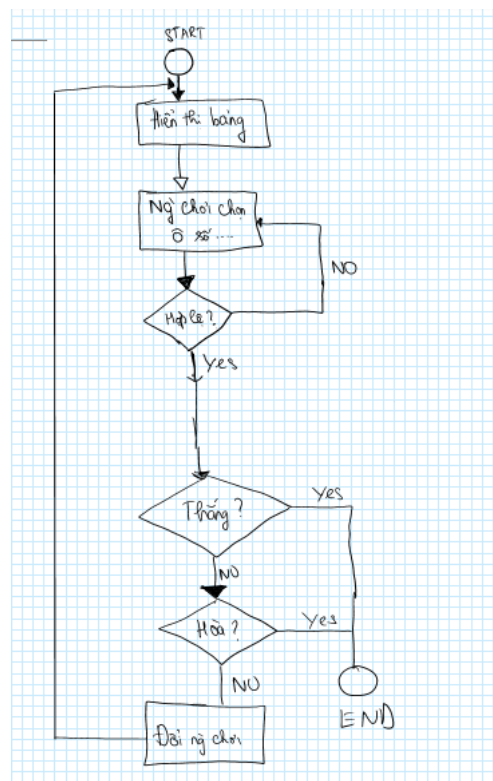
```

- Kết quả



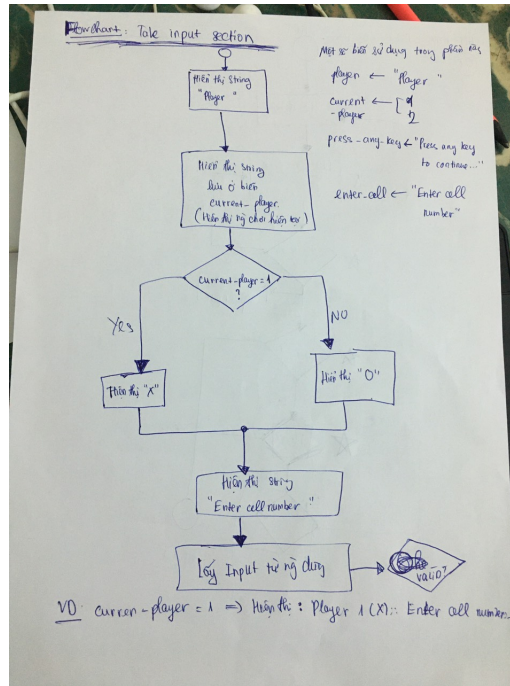
▼ Thiết kết Logic của trò chơi

▼ Lưu đồ của trò chơi (Tic Tac Toe Game Flowchart)



▼ Lấy dữ liệu từ người chơi (Take Input Section)

- Lưu đồ (Take Input Section Flowchart)



Một số biến sử dụng trong phần này

- `player` ← "player"
- `current_player` ← 1 hoặc 2
- `press_any_key` ← "Press any key to continue..."
- `enter_cell` ← "Enter cell number"

• Mã nguồn

```

;***** TAKE INPUT SECTION *****;
INPUT:
;Display string player
mov ah,09h
lea dx,player
int 21h

;Display current player
mov ah,02h
mov dl,current_player
int 21h

;Display cell mark of current player
cmp current_player,50 ; ASCII 50->2
je player_2_label

player_1_label:
;Display cell mark X
mov ah,09h
lea dx,player_cell_1
int 21h
jmp take_input:

player_2_label:
;display cell mark 0
mov ah,09h
lea dx,player_cell_2
int 21h

take_input:
;Display 'enter cell number' string
mov ah,09h
lea dx,enter_cell
int 21h

;take input from player & store at BL reg

```

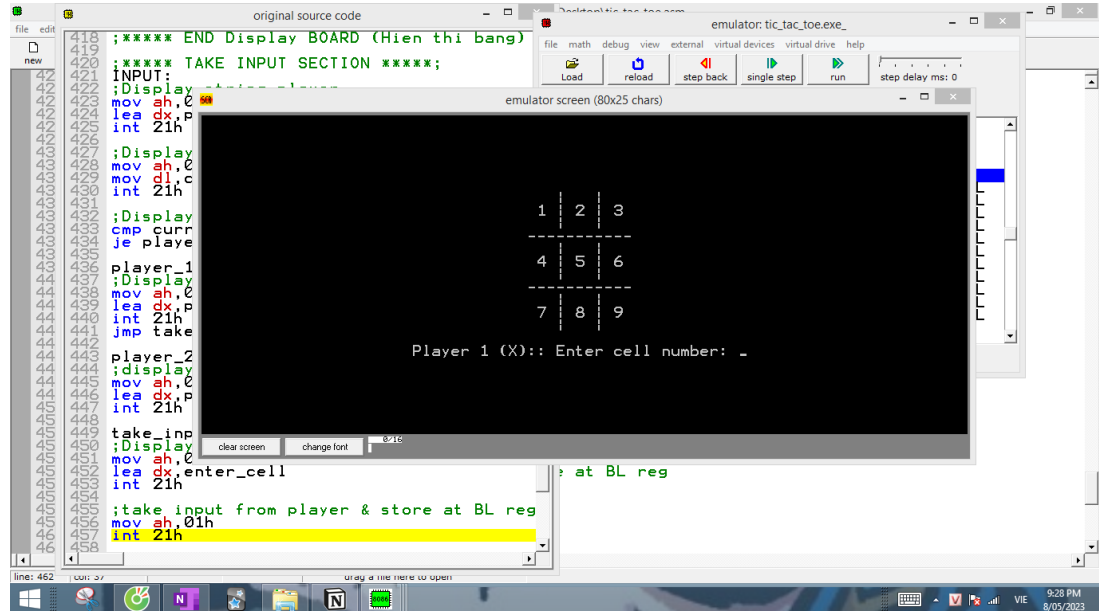
```

mov ah,01h
int 21h

;***** END TAKE INPUT SECTION *****;

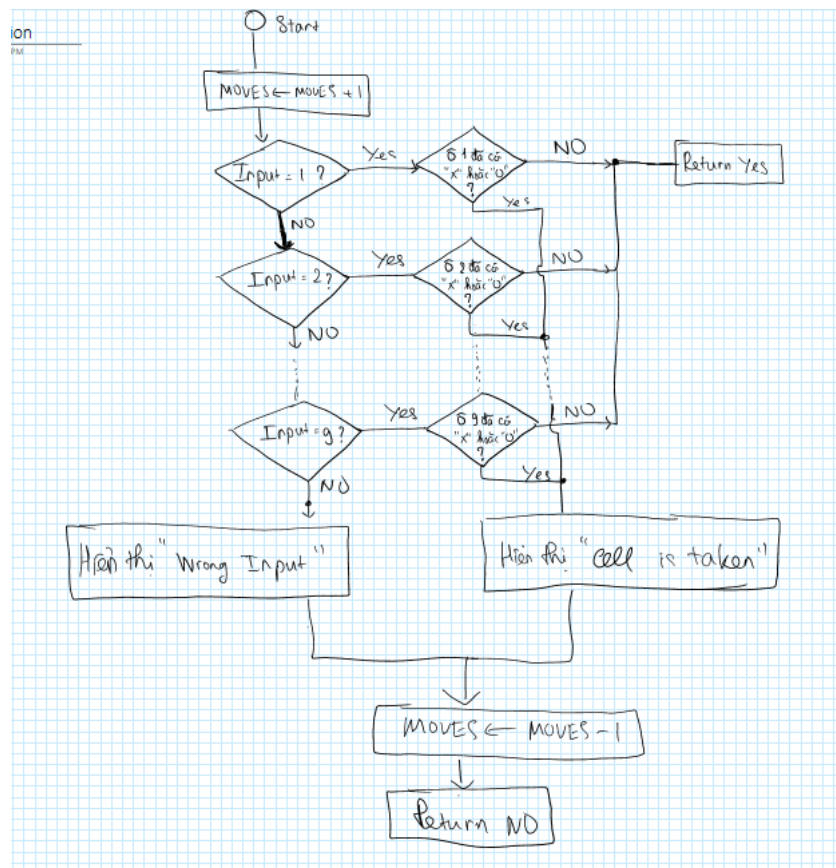
```

- Giao diện hiển thị



▼ Kiểm tra dữ liệu thu được có hợp lệ hay không (Check Valid Section)

- Lưu đồ





Một số biến sử dụng ở trong này

- **WI** ← “Wrong input! Press any key...”
- **taken_str** ← “This cell is taken! Press any key...”
- **MOVES** ← biến đếm dùng để kiểm tra trạng thái Hòa
- **AI** ← input
- **CL** ← X hoặc O tùy thuộc vào người chơi hiện tại

- Mã nguồn

```
;***** CHECK VALID SECTION *****;
CHECK_VALID:

    inc moves ; INCREMENTING MOVES COUNTER BY 1

    ; CHECKING IF INPUT IS BETWEEN 1-9 ; Now Bl <- Input
    CMP al, 1
    je C1_check

    CMP al, 2
    je C2_check

    CMP al, 3
    je C3_check

    CMP al, 4
    je C4_check

    CMP al, 5
    je C5_check

    CMP al, 6
    je C6_check

    CMP al, 7
    je C7_check

    CMP al, 8
    je C8_check

    CMP al, 9
    je C9_check

INVALID:    ; IF INPUT IS INVALID

    dec moves ; DECREMENTING MOVES BY 1, SINCE IT WAS INVALID

    ;set cursor position at row = 18, col = 23
    go_to_xy 18,23

    lea dx, wi ; display WRONG INPUT MESSAGE
    mov ah,09h
    int 21h

    mov ah, 07h ; INPUT WITHOUT ECHO
    int 21h

    ;set cursor position at row = 18, col = 23
    go_to_xy 18,23

    lea dx, emp ; CLEARING THAT LINE
    mov ah, 09h
    int 21h

    ;set cursor position at row = 18, col = 23
    go_to_xy 18,23

    jmp input ; jmp to input section
;-----

TAKEN: ;Display "This cell is taken" string

    dec moves

    ;set cursor position at row = 18, col = 23
    go_to_xy 18,23
```

```

    lea dx, taken_str ; display WRONG INPUT MESSAGE
    mov ah, 09h
    int 21h

    mov ah, 07h ; INPUT WITHOUT ECHO
    int 21h

    ; SET CURSOR
    go_to_xy 18,23

    lea dx, emp ; CLEARING THAT LINE
    mov ah, 09h
    int 21h

    ; SET CURSOR
    go_to_xy 18,23

    jmp input
; SETTING BOARD POSITION AS INPUT MARK
C1_check:
    CMP C1, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN
    CMP C1, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
    je TAKEN

    MOV C1, CL
    JMP board ; For temporary, change latter

C2_check:
    CMP C2, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN
    CMP C2, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
    je TAKEN

    MOV C2, CL
    JMP board ; For temporary, change latter
C3_check:
    CMP C3, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN
    CMP C3, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
    je TAKEN

    MOV C3, CL
    JMP board ; For temporary, change latter
C4_check:
    CMP C4, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN
    CMP C4, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
    je TAKEN

    MOV C4, CL
    JMP board ; For temporary, change latter
C5_check:
    CMP C5, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN
    CMP C5, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
    je TAKEN

    MOV C5, CL
    JMP board ; For temporary, change latter
C6_check:
    CMP C6, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN
    CMP C6, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
    je TAKEN

    MOV C6, CL
    JMP board ; For temporary, change latter
C7_check:
    CMP C7, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN
    CMP C7, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
    je TAKEN

    MOV C7, CL
    JMP board ; For temporary, change latter
C8_check:
    CMP C8, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN
    CMP C8, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
    je TAKEN

    MOV C8, CL
    JMP board ; For temporary, change latter
C9_check:
    CMP C9, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN

```

```

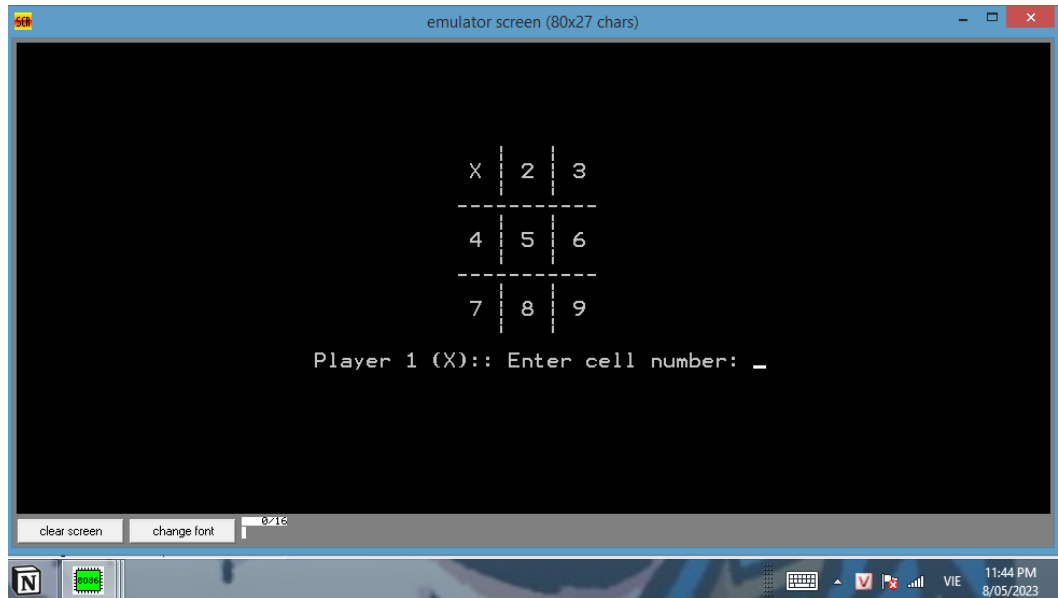
CMP C9, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
je TAKEN

MOV C9, CL
JMP board ; For temporary, change latter
; -----
;***** END VALID SECTION *****;

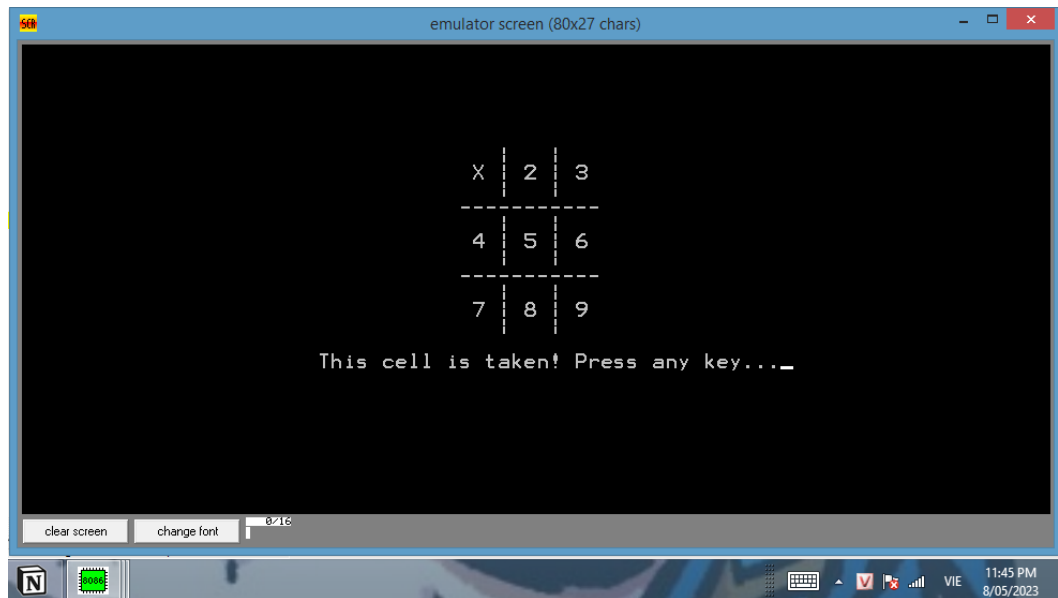
```

- Giao diện hiển thị

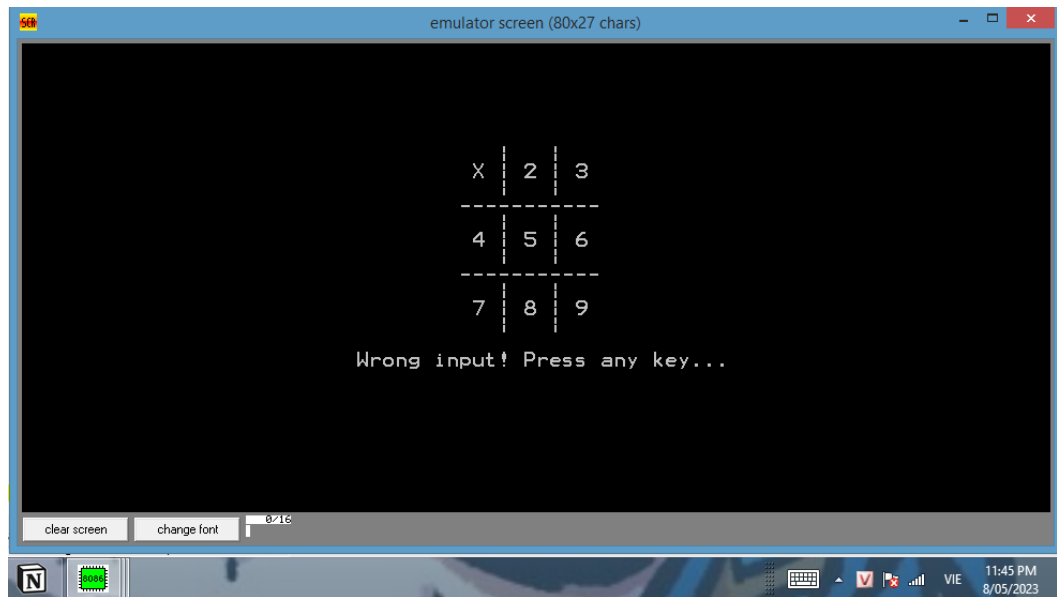
- Input = 1



- Input = 1 sau đó input = 1

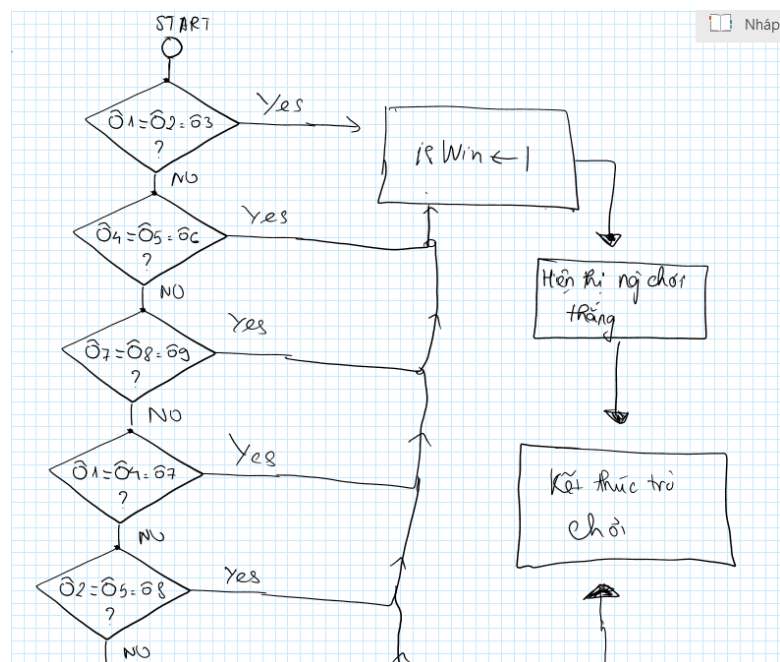


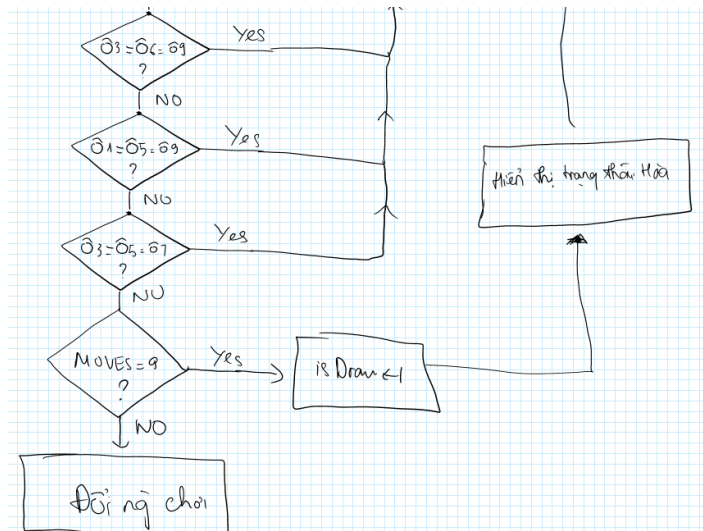
- Input = a



▼ Kiểm tra điều kiện thắng và điều kiện hòa (Check Win & Draw Condition)

- Lưu đồ





Một số biến sử dụng trong phần này

- `isWin` = 1 nếu trò chơi đã tìm được người thắng, `isWin` = 0 nếu ngược lại
- `isDraw` = 1 nếu 2 người chơi hòa, `isWin` = 0 nếu ngược lại

- Mã nguồn kiểm tra điều kiện thắng và hòa

```

;***** CHECK IF WINNING CONDITION IS MET *****;

CHECK:
; THERE ARE 8 POSSIBLE WINNING COMBINATIONS

CHECK1: ; CHECKING 1, 2, 3
MOV AL, C1
MOV BL, C2
MOV CL, C3

CMP AL, BL
JNZ CHECK2

CMP BL, CL
JNZ CHECK2

MOV isWin, 1
JMP BOARD

CHECK2: ; CHECKING 4, 5, 6
MOV AL, C4
MOV BL, C5
MOV CL, C6

CMP AL, BL
JNZ CHECK3

CMP BL, CL
JNZ CHECK3

MOV isWin, 1
JMP BOARD

CHECK3: ; CHECKING 7, 8, 9
MOV AL, C4
MOV BL, C5
MOV CL, C6

CMP AL, BL
JNZ CHECK4

CMP BL, CL
JNZ CHECK4

MOV isWin, 1

```

```

        JMP BOARD

CHECK4:    ; CHECKING 1, 4, 7
        MOV AL, C1
        MOV BL, C4
        MOV CL, C7

        CMP AL, BL
        JNZ CHECK5

        CMP BL, CL
        JNZ CHECK5

        MOV isWin, 1
        JMP BOARD

CHECK5:    ; CHECKING 2, 5, 8
        MOV AL, C2
        MOV BL, C5
        MOV CL, C8

        CMP AL, BL
        JNZ CHECK6

        CMP BL, CL
        JNZ CHECK6

        MOV isWin, 1
        JMP BOARD

CHECK6:    ; CHECKING 3, 6, 9
        MOV AL, C3
        MOV BL, C6
        MOV CL, C9

        CMP AL, BL
        JNZ CHECK7

        CMP BL, CL
        JNZ CHECK7

        MOV isWin, 1
        JMP BOARD

CHECK7:    ; CHECKING 1, 5, 9
        MOV AL, C1
        MOV BL, C5
        MOV CL, C9

        CMP AL, BL
        JNZ CHECK8

        CMP BL, CL
        JNZ CHECK8

        MOV isWin, 1
        JMP BOARD

CHECK8:    ; CHECKING 3, 5, 7
        MOV AL, C3
        MOV BL, C5
        MOV CL, C7

        CMP AL, BL
        JNZ DRAWCHECK

        CMP BL, CL
        JNZ DRAWCHECK

        MOV isWin, 1
        JMP BOARD

DRAWCHECK:
        MOV AL, MOVES
        CMP AL, 9
        JL PLRCHANGE

        MOV isDraw, 1
        JMP BOARD

; JMP EXIT

```

```

; ----- PLAYER -----
PLRCHANGE:

    CMP current_player, 49 ; Compare current_play vs '1'
    JZ P2                 ; jump to label p2 if equal
    CMP current_player, 50 ; Compare current_play vs '2'
    JZ P1                 ; jump to label p1 if equal;

P1:
    MOV current_player, 49 ; current_player <- '1'
    MOV current_char, 88   ; current_char <- 'X'

    JMP BOARD

P2:
    MOV current_player, 50 ; current_player <- '2'
    MOV current_char, 79   ; ; current_char <- 'O'
    JMP BOARD

```

- Mã nguồn hiển thị người chơi thắng hoặc trạng thái hòa

```

;***** DISPLAY WINNER *****
VICTORY:

    LEA DX, player
    MOV AH, 09h
    INT 21H

    LEA DX, current_player
    MOV AH, 09h
    INT 21H

    LEA DX, winner_str
    MOV AH, 09h
    INT 21H

    go_to_xy 18,23

    LEA DX, press_any_key ; PRESS ANY KEY
    MOV AH, 9
    INT 21H

    MOV AH, 7 ; INPUT WITHOUT ECHO
    INT 21H

    ;JMP TRYAGAIN
    jmp terminate

;***** DISPLAY WINNER END *****

;***** DISPLAY DRAWING STATE *****
DRAW:
    LEA DX, draw_str
    MOV AH, 09h
    INT 21H

    go_to_xy 18,23

    LEA DX, press_any_key ; PRESS ANY KEY
    MOV AH, 09h
    INT 21H

    MOV AH, 07h ; INPUT WITHOUT ECHO
    INT 21H

    ;JMP TRYAGAIN
    jmp terminate

;.....
;.....
;.....

;ADD THESE CODES BELOW AT THE END OF BOARD SECTION
BOARD:
    ;....
    ;....
    ;....
    CMP isWin, 1
    je VICTORY

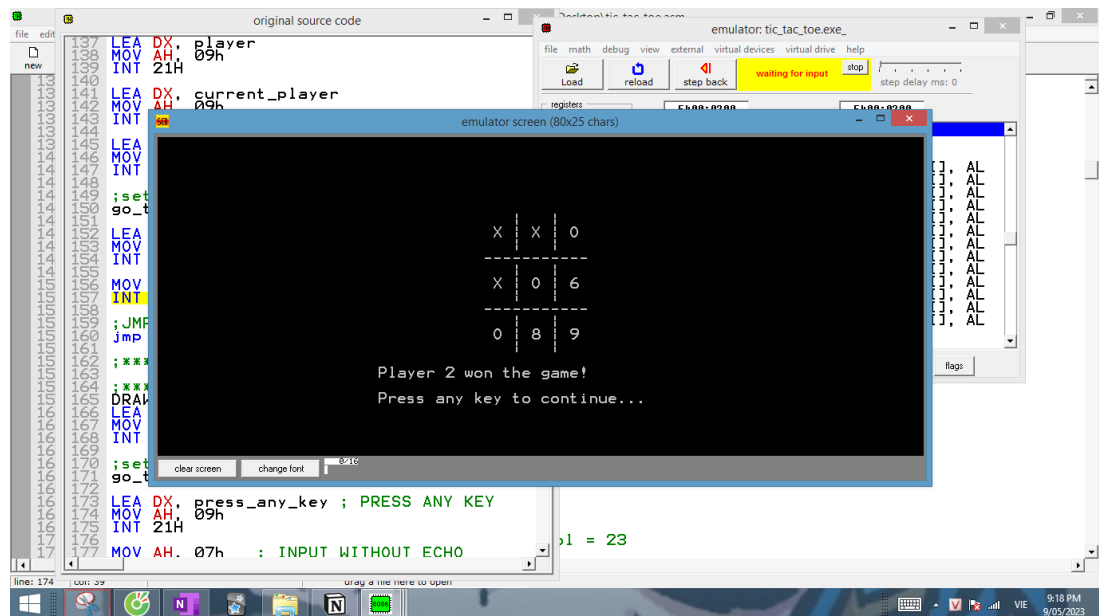
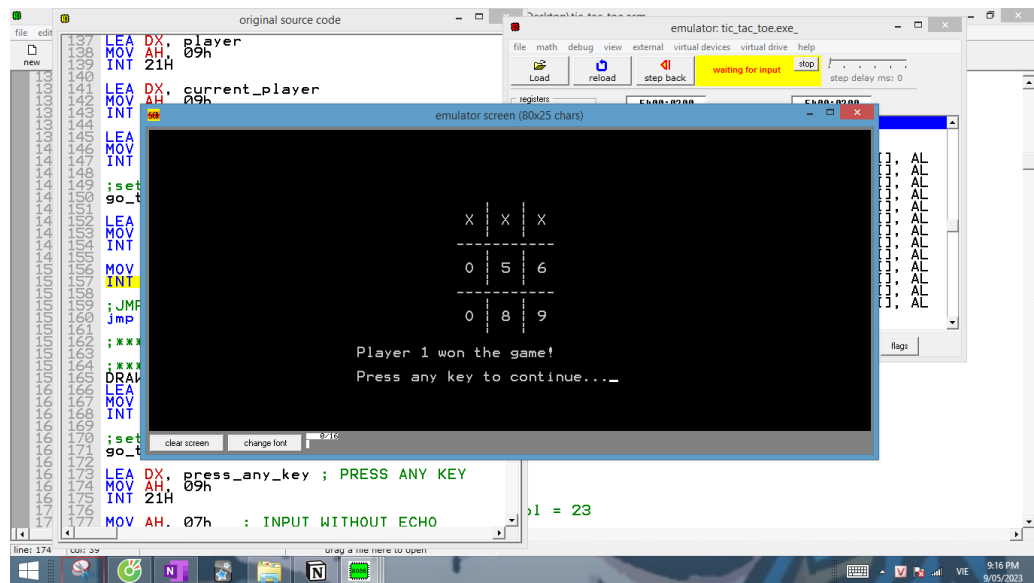
```

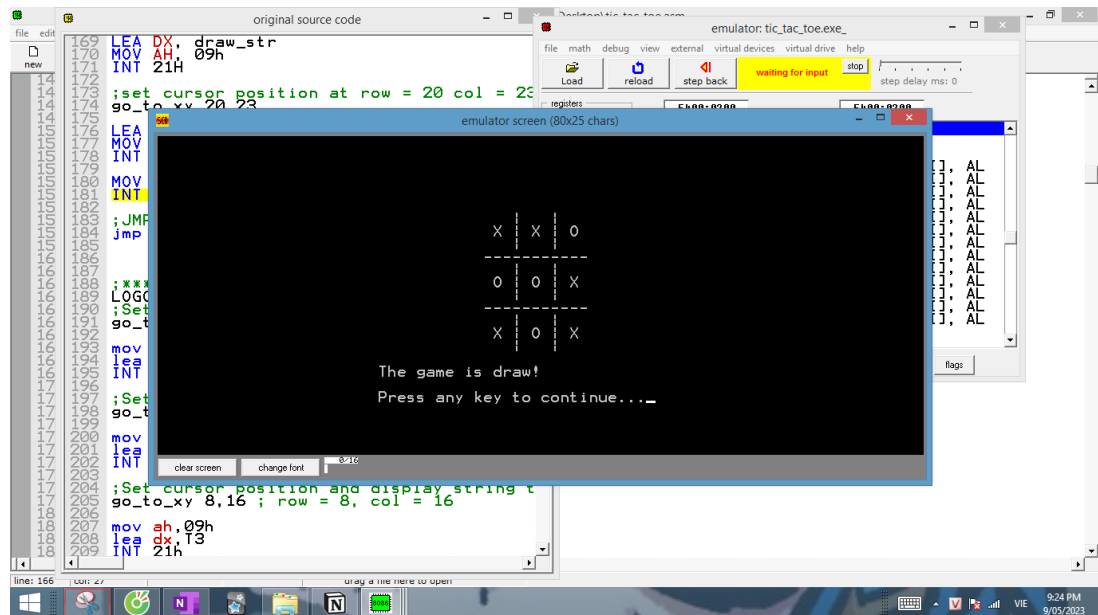
```

CMP isDraw, 1
je DRAW

```

- Giao diện hiển thị





▼ Mã nguồn (Source Code)

```
; BTL NHOM 16 : TIC-TAC-TOE
.model small
.stack 100h

;***** DEFINE MACROS *****;
; set cursor position
go_to_xy macro row,col
    mov dh,row ; set row-position cursor
    mov dl,col ; set col-position cursor
    mov bh,0 ; set page number default value is 0
    mov ah,2
    int 10h
endm

; clear the screen
clear_screen macro
    mov ax,0600h ; here ah = 06h and al = 00h (scroll entire window)
    mov bh,07h ; here background color = black (0) and fore color = gray(7)
    mov cx,0000h ; scroll from row = 00h (0d), col = 00h (0d)
    mov dx,184Fh ; to row = 18h (24d), col = 4Fh (80d)
    int 10h
endm

;Print string with specific color at cursor position
print_colored_string macro string,row,col,color

    local lop,fin ; unique label purpose

    mov si, offset string ; si point to first char of string
    mov di, col ; di = starting column position note that di is 16-bit reg
    lop:
        ;set cursor position ;Because of di is 16-bit reg,so we can NOT pass it to go_to_xy macro
        mov ah,02h
        mov dx,di ; dl = low byte of di
        mov dh,row
        mov bh,0h
        int 10h

        lodsb ; load current character from ds:[si] and increse si (if df = 0)
        cmp al,'$' ; Check whether we reach terminate character or not
        JE fin ; If yes ( AL = '$' ) -> jump to label fin
        ; If no print character

        ;print current character stored at al by using int10h/ah=9
        mov ah,09h
        mov bh,0h
        mov bl,color ; color must be get from emu8086 bit color table
        mov cx,1 ; no.of times to write characters = 1
        int 10h
        inc di ; increase column position
        jmp lop

    fin:
endm
```



```

    LEA DX, current_player
    MOV AH, 09h
    INT 21H

    LEA DX, winner_str
    MOV AH, 09h
    INT 21H

;set cursor position at row = 20 col = 23
go_to_xy 20,23

    LEA DX, press_any_key ; PRESS ANY KEY
    MOV AH, 9
    INT 21H

    MOV AH, 7 ; INPUT WITHOUT ECHO
    INT 21H

;JMP TRYAGAIN
jmp terminate

;***** DISPLAY WINNER END *****

;***** DISPLAY DRAWING STATE *****

    go_to_xy 20,39

DRAW:
    LEA DX, draw_str
    MOV AH, 09h
    INT 21H

;set cursor position at row = 20 col = 23
go_to_xy 20,23

    LEA DX, press_any_key ; PRESS ANY KEY
    MOV AH, 09h
    INT 21H

    MOV AH, 07h ; INPUT WITHOUT ECHO
    INT 21H

;JMP TRYAGAIN
jmp terminate

;***** Display logo TIC-TAC-TOE (Hien thi logo TIC-TAC-TOE) *****;
LOGO:
;Set cursor position and display string t1
go_to_xy 6,16 ; row = 6, col = 16

    mov ah,09h
    lea dx,T1
    INT 21h

;Set cursor position and display string t2
go_to_xy 7,16 ; row = 7, col = 16

    mov ah,09h
    lea dx,T2
    INT 21h

;Set cursor position and display string t3
go_to_xy 8,16 ; row = 8, col = 16

    mov ah,09h
    lea dx,T3
    INT 21h

;Set cursor position and display string t4
go_to_xy 9,16 ; row = 9, col = 16

    mov ah,09h
    lea dx,T4
    INT 21h

;Set cursor position and display string t5
go_to_xy 10,16 ; row = 10, col = 16

    mov ah,09h
    lea dx,T5
    INT 21h

;TEST -> Too much time :(

```



```

;print_colored_string T1,6,16,0ch
;print_colored_string T2,7,16,0ch
;print_colored_string T3,8,16,0ch
;print_colored_string T4,9,16,0ch
;print_colored_string T5,10,16,0ch

;Set cursor position and display group title ; col = (80 - no.of chars)/2 = (80-32)/2 = 24
;
;go_to_xy 12,24 ; row = 12, col = 24

;mov ah,09h
;lea dx,group_title
;INT 21h

; Print Group title with row = 12, col = 24, color = light blue (1001b) (9h)
print_colored_string group_title,12,24,9h

;Set cursor position and display PAK string ; col = (80 - no.of chars)/2 = (80-28)/2 = 26
go_to_xy 14,26 ; row = 14, col = 26

mov ah,09h
lea dx,press_any_key
INT 21h

;Get input without echo
mov ah,7
int 21h

;Clear the screen
clear_screen

;jump to rule section
jmp RULE

;***** END Display logo TIC-TAC-TOE (Hien thi logo TIC-TAC-TOE) *****;

;***** Display RULES (Hien thi luat choi) *****;
RULE:
; Print Rules title with row = 4, col = 4, color = light red (1100b) (0Ch)
print_colored_string R,3,4,0ch

; Can also use
;Set cursor position and display Rules title
;go_to_xy 4,4
;;mov ah,09h
;lea dx,R
;int 21h

;Set cursor position and display R0 ; col = (80-70)/2 = 5
go_to_xy 5,5
mov ah,09h
lea dx,R0
int 21h

;Set cursor position and display R1
go_to_xy 7,5
mov ah,09h
lea dx,R1
int 21h

;Set cursor position and display R2
go_to_xy 9,5
mov ah,09h
lea dx,R2
int 21h

;Set cursor position and display R3
go_to_xy 11,5
mov ah,09h
lea dx,R3
int 21h

;Set cursor position and display R4
go_to_xy 13,5
mov ah,09h
lea dx,R4
int 21h

;Set cursor position and display R5
go_to_xy 15,5
mov ah,09h
lea dx,R5
int 21h

```

```

;Set cursor position and display R6
go_to_xy 17,5
mov ah,09h
lea dx,R6
int 21h

; Print R7 with row = 18, col = 4, color = light red (1110b) (0Eh)
print_colored_string R7,19,4,0Eh;

;Can also use
;Set cursor position and display R7
;go_to_xy 19,4
;mov ah,09h
;lea dx,R7
;int 21h

;Set cursor position and display PAK
go_to_xy 21,4
mov ah,09h
lea dx,press_any_key
int 21h

;get input without echo
mov ah,07h
int 21h

;***** END Display RULES (Hien thi luat choi) *****;

;***** Display BOARD (Hien thi bang) *****;
BOARD:
;Clear the screen
clear_screen

;set cursor position row = 6,7,8 col = 34 & display string : L1 : ' | | $'
go_to_xy 6,34

mov ah,09h
lea dx,L1
int 21h

go_to_xy 7,34

mov ah,09h
lea dx,L1
int 21h

go_to_xy 8,34

mov ah,09h
lea dx,L1
int 21h

;Display cell 1,2,3
;display cell 1 at row = 7, col = 35
go_to_xy 7,35
mov ah,09h
lea dx,c1
int 21h

;display cell 2 at row = 7, col = 39
go_to_xy 7,39
mov ah,09h
lea dx,c2
int 21h

;display cell 3 at row = 7, col = 43
go_to_xy 7,43
mov ah,09h
lea dx,c3
int 21h

;set cursor position row = 9 col = 34 & display string L2 : '-----$'
go_to_xy 9,34

mov ah,09h
lea dx,L2
int 21h

;set cursor position row = 10,11,12 col = 34 & display string : L1 : ' | | $'
go_to_xy 10,34

mov ah,09h
lea dx,L1

```

```

int 21h

go_to_xy 11,34

mov ah,09h
lea dx,L1
int 21h

go_to_xy 12,34

mov ah,09h
lea dx,L1
int 21h

;Display cell 4,5,6
;display cell 4 at row = 11, col = 35
go_to_xy 11,35

mov ah,09h
lea dx,c4
int 21h

;display cell 5 at row = 11, col = 39
go_to_xy 11,39

mov ah,09h
lea dx,c5
int 21h

;display cell 6 at row = 11, col = 43
go_to_xy 11,43

mov ah,09h
lea dx,c6
int 21h

;set cursor position row = 13 col = 34 & display string L2 : '-----$'
go_to_xy 13,34

mov ah,09h
lea dx,L2
int 21h

;set cursor position row = 14,15,16 col = 34 & display string : L1 : ' | | $'
go_to_xy 14,34

mov ah,09h
lea dx,L1
int 21h

go_to_xy 15,34

mov ah,09h
lea dx,L1
int 21h

go_to_xy 16,34

mov ah,09h
lea dx,L1
int 21h
;Display cell 7,8,9
;display cell 7 at row = 15, col = 35
go_to_xy 15,35

mov ah,09h
lea dx,c7
int 21h

;display cell 8 at row = 15, col = 39
go_to_xy 15,39

mov ah,09h
lea dx,c8
int 21h

;display cell 9 at row = 15, col = 43
go_to_xy 15,43

mov ah,09h
lea dx,c9
int 21h

;Set cursor position at row = 18, col = 23 for message
go_to_xy 18,23

```

```

        CMP isWin, 1
        je VICTORY

        CMP isDraw, 1
        je DRAW
;***** END Display BOARD (Hien thi bang) *****;

;***** TAKE INPUT SECTION *****;
INPUT:
;Display string player
        mov ah,09h
        lea dx,player
        int 21h

;Display current player
        mov ah,02h
        mov dl,current_player
        int 21h

;Display cell mark of current player
        cmp current_player,50 ; ASCII 50->2
        je player_2_label

        player_1_label:
        ;Display cell mark X
        mov ah,09h
        lea dx,player_cell_1
        int 21h
        jmp take_input:

        player_2_label:
        ;display cell mark 0
        mov ah,09h
        lea dx,player_cell_2
        int 21h

take_input:
;Display 'enter cell number' string
        mov ah,09h
        lea dx,enter_cell
        int 21h

;take input from player & store at AL reg
        mov ah,01h
        int 21h

        sub al,48 ;Get 'actual' value of input
        mov cl, current_char

;***** END TAKE INPUT SECTION *****;

;***** CHECK VALID SECTION *****;
CHECK_VALID:

        inc moves ; INCREMENTING MOVES COUNTER BY 1

; CHECKING IF INPUT IS BETWEEN 1-9 ; Now Bl <- Input
        CMP al, 1
        je C1_check

        CMP al, 2
        je C2_check

        CMP al, 3
        je C3_check

        CMP al, 4
        je C4_check

        CMP al, 5
        je C5_check

        CMP al, 6
        je C6_check

        CMP al, 7
        je C7_check

        CMP al, 8
        je C8_check

        CMP al, 9
        je C9_check

INVALID: ; IF INPUT IS INVALID

        dec moves ; DECREMENTING MOVES BY 1, SINCE IT WAS INVALID

```

```

;set cursor position at row = 18, col = 23
go_to_xy 18,23

lea dx, w1 ; display WRONG INPUT MESSAGE
mov ah,09h
int 21h

mov ah, 07h ; INPUT WITHOUT ECHO
int 21h

;set cursor position at row = 18, col = 23
go_to_xy 18,23

lea dx, emp ; CLEARING THAT LINE
mov ah, 09h
int 21h

;set cursor position at row = 18, col = 23
go_to_xy 18,23

jmp input ; jmp to input section
;-----

TAKEN: ;Display "This cell is taken" string

dec moves

;set cursor position at row = 18, col = 23
go_to_xy 18,23

lea dx, taken_str ; display WRONG INPUT MESSAGE
mov ah,09h
int 21h

mov ah, 07h ; INPUT WITHOUT ECHO
int 21h

; SET CURSOR
go_to_xy 18,23

lea dx, emp ; CLEARING THAT LINE
mov ah, 09h
int 21h

; SET CURSOR
go_to_xy 18,23

jmp input
; SETTING BOARD POSITION AS INPUT MARK
C1_check:
    CMP C1, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN
    CMP C1, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
    je TAKEN

    MOV C1, CL
    JMP check

C2_check:
    CMP C2, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN
    CMP C2, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
    je TAKEN

    MOV C2, CL
    JMP check

C3_check:
    CMP C3, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN
    CMP C3, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
    je TAKEN

    MOV C3, CL
    JMP check

C4_check:
    CMP C4, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN
    CMP C4, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
    je TAKEN

    MOV C4, CL
    JMP check

C5_check:
    CMP C5, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
    je TAKEN
    CMP C5, 79 ; CHECKING IF THE CELL IS ALREADY 'O'

```

```

        je TAKEN

        MOV C5, CL
        JMP check
C6_check:
        CMP C6, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
        je TAKEN
        CMP C6, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
        je TAKEN

        MOV C6, CL
        JMP check
C7_check:
        CMP C7, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
        je TAKEN
        CMP C7, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
        je TAKEN

        MOV C7, CL
        JMP check
C8_check:
        CMP C8, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
        je TAKEN
        CMP C8, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
        je TAKEN

        MOV C8, CL
        JMP check
C9_check:
        CMP C9, 88 ; CHECKING IF THE CELL IS ALREADY 'X'
        je TAKEN
        CMP C9, 79 ; CHECKING IF THE CELL IS ALREADY 'O'
        je TAKEN

        MOV C9, CL
        JMP check
; -----
;***** END VALID SECTION *****;

;***** CHECK IF WINNING CONDITION IS MET *****;

CHECK:
; THERE ARE 8 POSSIBLE WINNING COMBINATIONS

CHECK1: ; CHECKING 1, 2, 3
        MOV AL, C1
        MOV BL, C2
        MOV CL, C3

        CMP AL, BL
        JNZ CHECK2

        CMP BL, CL
        JNZ CHECK2

        MOV isWin, 1
        JMP BOARD

CHECK2: ; CHECKING 4, 5, 6
        MOV AL, C4
        MOV BL, C5
        MOV CL, C6

        CMP AL, BL
        JNZ CHECK3

        CMP BL, CL
        JNZ CHECK3

        MOV isWin, 1
        JMP BOARD

CHECK3: ; CHECKING 7, 8, 9
        MOV AL, C4
        MOV BL, C5
        MOV CL, C6

        CMP AL, BL
        JNZ CHECK4

        CMP BL, CL
        JNZ CHECK4

        MOV isWin, 1
        JMP BOARD

```

```

CHECK4:    ; CHECKING 1, 4, 7
            MOV AL, C1
            MOV BL, C4
            MOV CL, C7

            CMP AL, BL
            JNZ CHECK5

            CMP BL, CL
            JNZ CHECK5

            MOV isWin, 1
            JMP BOARD

CHECK5:    ; CHECKING 2, 5, 8
            MOV AL, C2
            MOV BL, C5
            MOV CL, C8

            CMP AL, BL
            JNZ CHECK6

            CMP BL, CL
            JNZ CHECK6

            MOV isWin, 1
            JMP BOARD

CHECK6:    ; CHECKING 3, 6, 9
            MOV AL, C3
            MOV BL, C6
            MOV CL, C9

            CMP AL, BL
            JNZ CHECK7

            CMP BL, CL
            JNZ CHECK7

            MOV isWin, 1
            JMP BOARD

CHECK7:    ; CHECKING 1, 5, 9
            MOV AL, C1
            MOV BL, C5
            MOV CL, C9

            CMP AL, BL
            JNZ CHECK8

            CMP BL, CL
            JNZ CHECK8

            MOV isWin, 1
            JMP BOARD

CHECK8:    ; CHECKING 3, 5, 7
            MOV AL, C3
            MOV BL, C5
            MOV CL, C7

            CMP AL, BL
            JNZ DRAWCHECK

            CMP BL, CL
            JNZ DRAWCHECK

            MOV isWin, 1
            JMP BOARD

DRAWCHECK:
            MOV AL, MOVES
            CMP AL, 9
            JL PLRCHANGE

            MOV isDraw, 1
            JMP BOARD

; ----- PLAYER -----
PLRCHANGE:

```

```

        CMP current_player, 49 ; Compare current_play vs '1'
        JZ P2                  ; jump to label p2 if equal
        CMP current_player, 50 ; Compare current_play vs '2'
        JZ P1                  ; jump to label p1 if equal;

P1:
        MOV current_player, 49 ; current_player <- '1'
        MOV current_char, 88   ; current_char <- 'X'

        JMP BOARD

P2:
        MOV current_player, 50 ; current_player <- '2'
        MOV current_char, 79   ; ; current_char <- '0'
        JMP BOARD

main endp

terminate:
        mov ah,4ch
        int 21h
END

```

▼ Tài liệu tham khảo (References)

- Ý tưởng đề tài được tham khảo từ [Tic-Tac-Toe](https://github.com/TanvirSojal/Tic-Tac-Toe) - [TanvirSoja](#)

<https://github.com/TanvirSojal/Tic-Tac-Toe>

- [Assembly Language for x86 Processors 7th Edition](#) by [Kip Irvine](#) (Author)
- [Documentation for emu8086 - assembler and microprocessor emulator](#)
- [Microprocessors - Bharat D. Acharya.](#)
- [How to print colored string in emu8086](#)

▼ Bảng mã màu emu8086

HEX	BIN	COLOR
0	0000	black
1	0001	blue
2	0010	green
3	0011	cyan
4	0100	red
5	0101	magenta
6	0110	brown
7	0111	light gray
8	1000	dark gray
9	1001	light blue
A	1010	light green
B	1011	light cyan
C	1100	light red
D	1101	light magenta
E	1110	yellow
F	1111	white