HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

BÀI TẬP LỚN
KIẾN TRÚC MÁY TÍNH

Nhóm : LỚP 09 - NHÓM 20

Tên thành viên	Mã sinh viên	Nhiệm vụ
Trần Đức Trung	B23DCCN864	Thuyết trình và viết code
Trần Văn Hiếu	B23DCCN313	Làm nội dung và viết báo cáo
Nguyễn Hữu Hiếu	B23DCCN304	Phân tích và tổng hợp code
Nguyễn Thế Bình	B23DCCN085	Làm slide

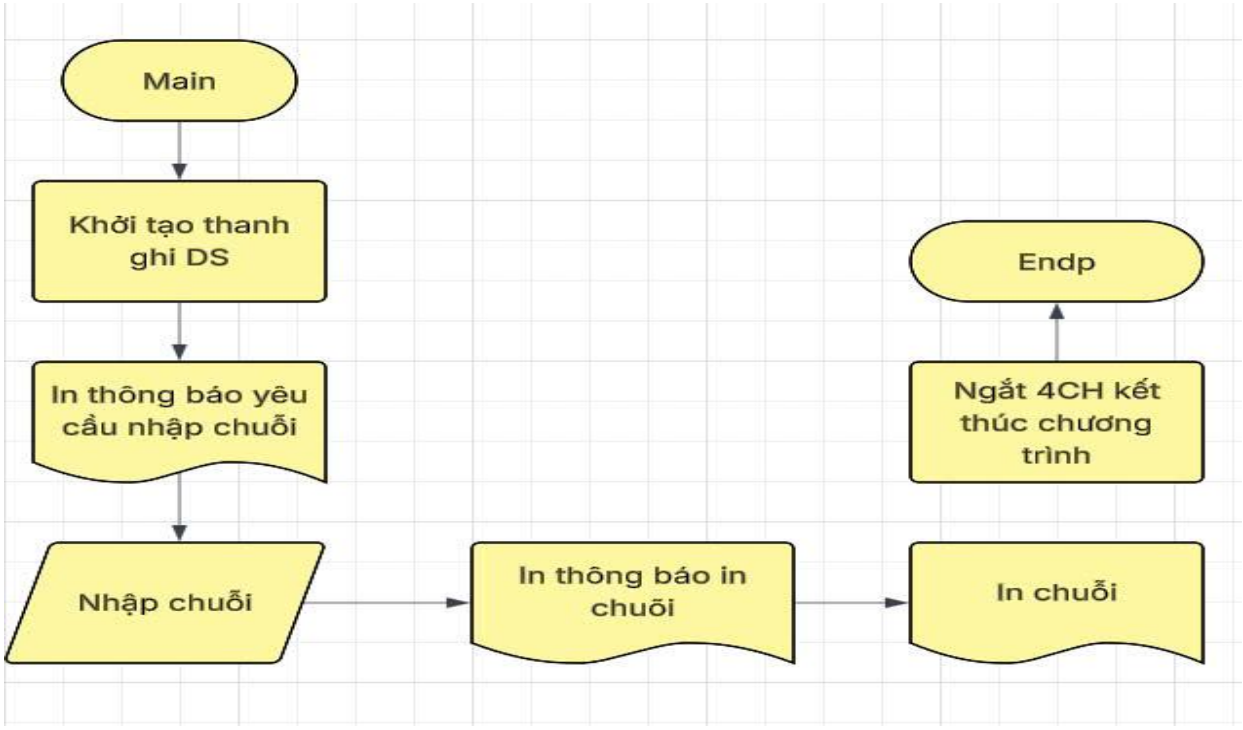
Giảng viên hướng dẫn : TS.Trần Tiến Công

Hà Nội - 2025

PHẦN 1 : BÀI LÀM CÁ NHÂN
(Sinh viên Trần Đức Trung - MSV B23DCCN864)

1.1. Lập trình hợp ngữ Assembly

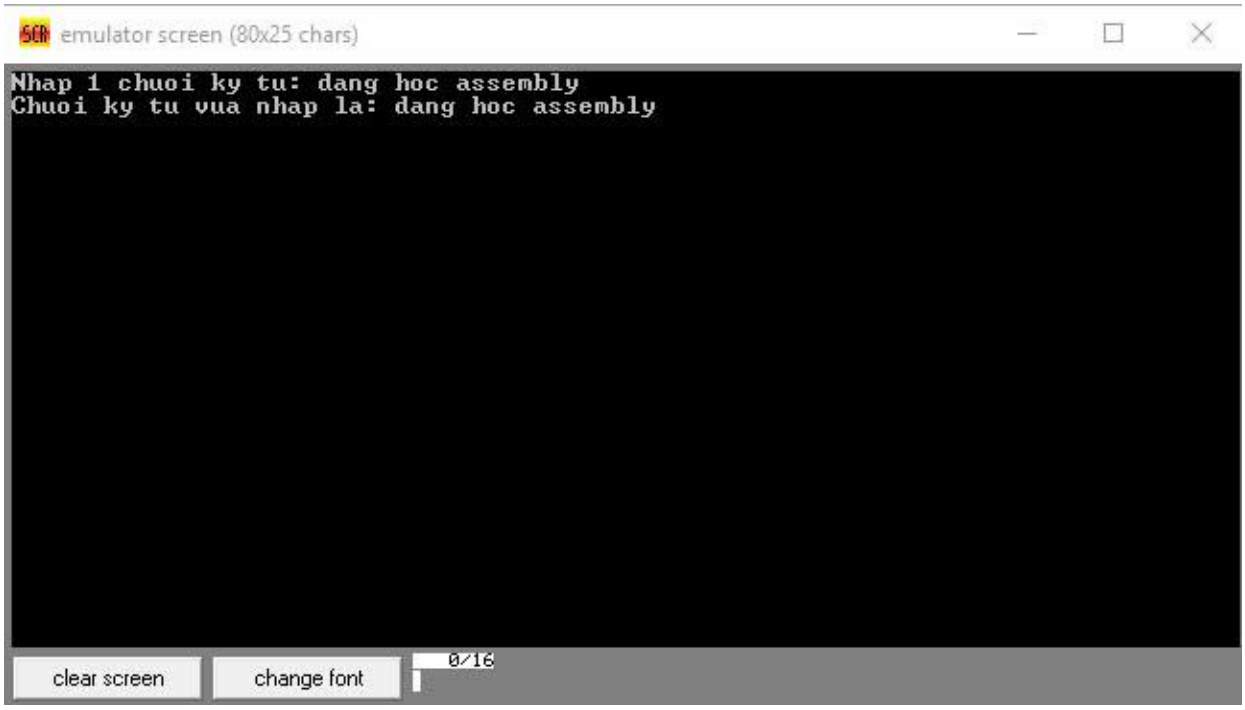
Câu 3 : *Viết chương trình hợp ngữ Assembly cho phép nhập 1 chuỗi ký tự và in ra màn hình chuỗi ký tự đó.*



Hình 3.1 *: Flow chart* ***Câu 3***

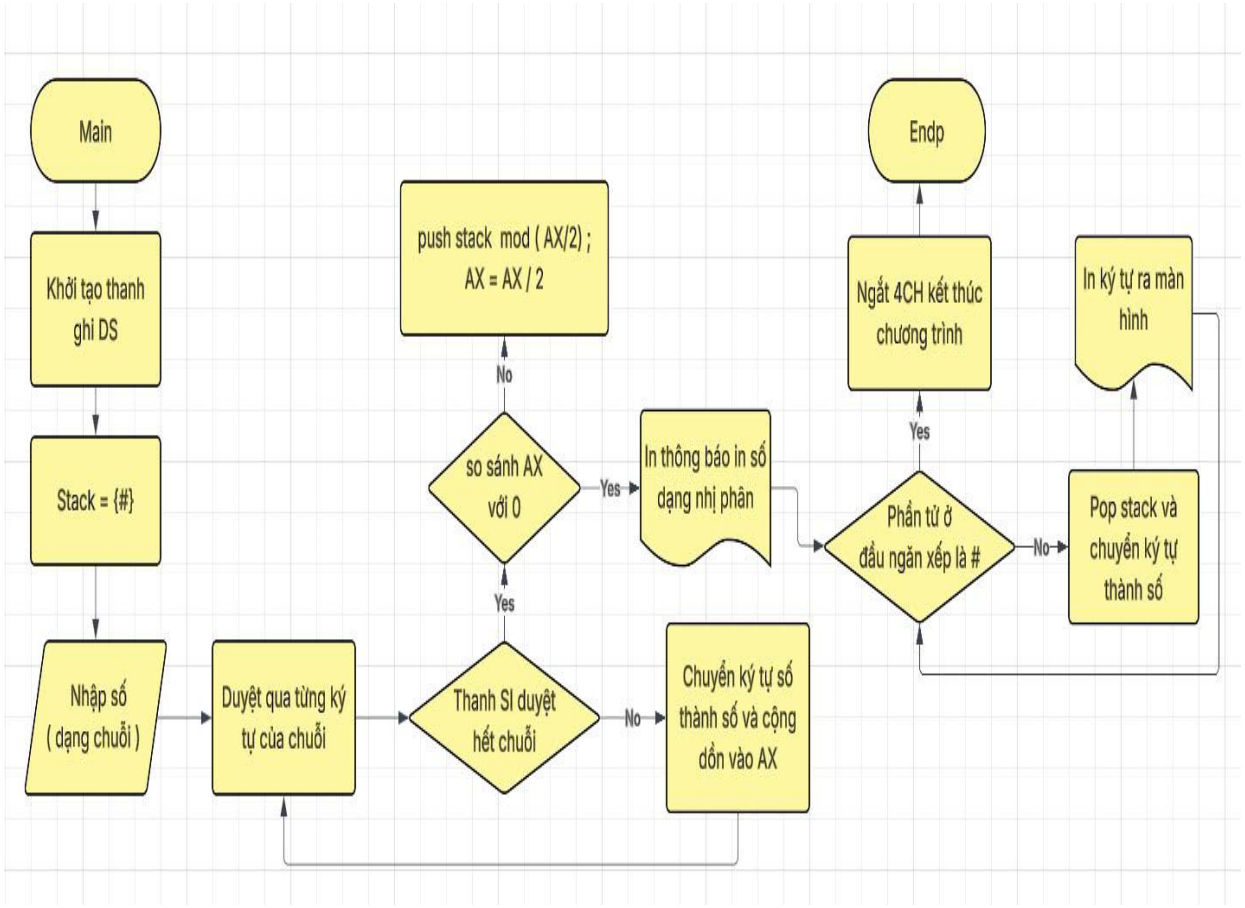
```
.Model Small
.Stack 100
.data
    op1 DB "Nhap 1 chuoi ky tu : $"              ; nhan op1
    op2 DB 13,10,"Chuoi ky tu vua nhap la : $"   ; nhan op2
    str DB 100 dup('$')                          ; chuoi str chua 100 ky tu
.code
MAIN PROC

    MOV AX , @data     ; khoi dau thanh ghi DS
    MOV DS, AX         ; tro thanh ghi DS ve dau doan data

    ; ham ngat AH loai 9 de in ra xau ky tu
    MOV AH , 9

    ; in ra nhan op1
    LEA DX , op1
    INT 21h

    MOV AH , 10        ; ham ngat AH loai 10 de nhap 1 chuoi ky tu
    LEA DX , str       ; tro den dia chi dau str
    INT 21h

    ; in nhan op2
    MOV AH , 9
    LEA DX , op2
    INT 21h

    LEA DX , str + 2  ; do str[0] luu kich thuoc toi da , str[1] luu kich thuoc thuc
    INT 21h           ; in chuoi vua nhap

    ; ham ket thuc chuong trinh
    MOV AH , 4CH
    INT 21h

MAIN ENDP
END
```

Hình 3.2 : *Mã nguồn* ***Câu 3***

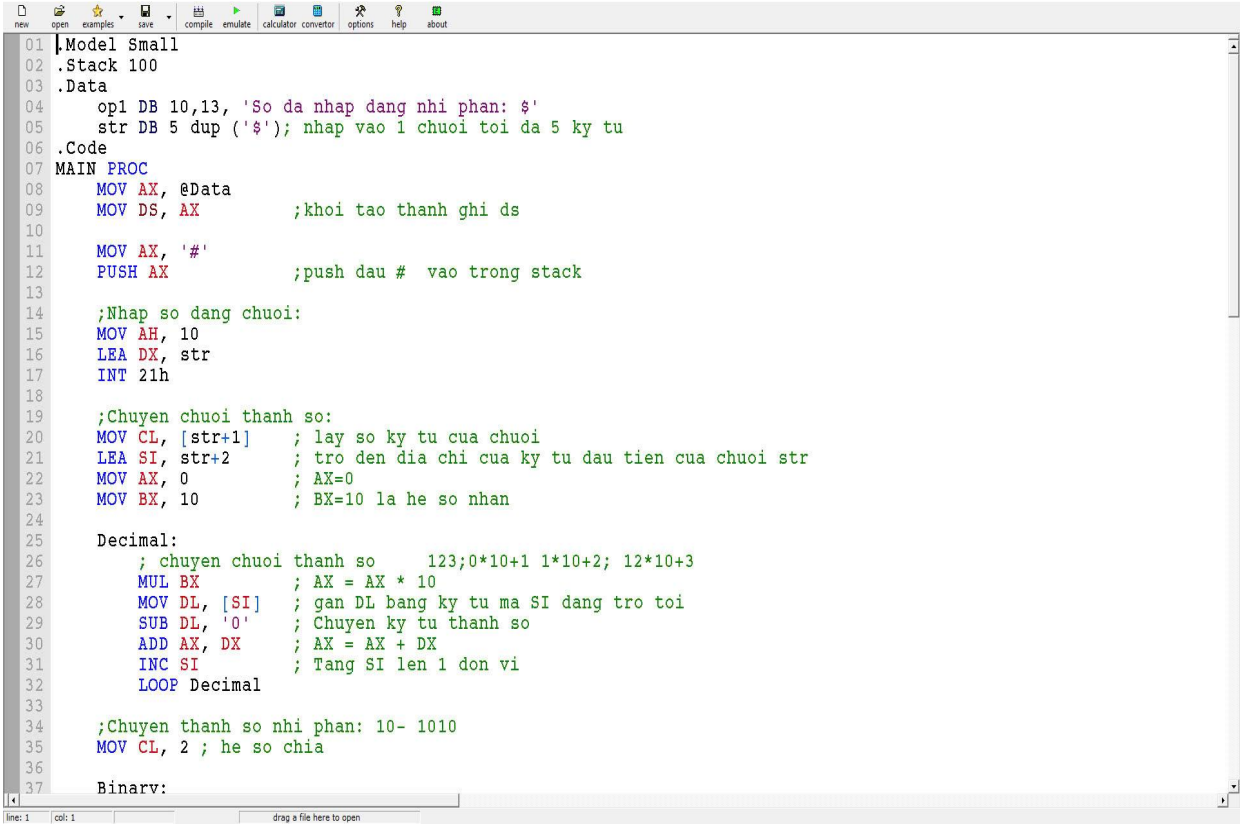


Hình 3.3 : *Giao diện hiển thị* ***Câu 3***

Câu 7 : *Viết chương trình Assembly chuyển đổi 1 số từ hệ 10 sang hệ nhị phân.*

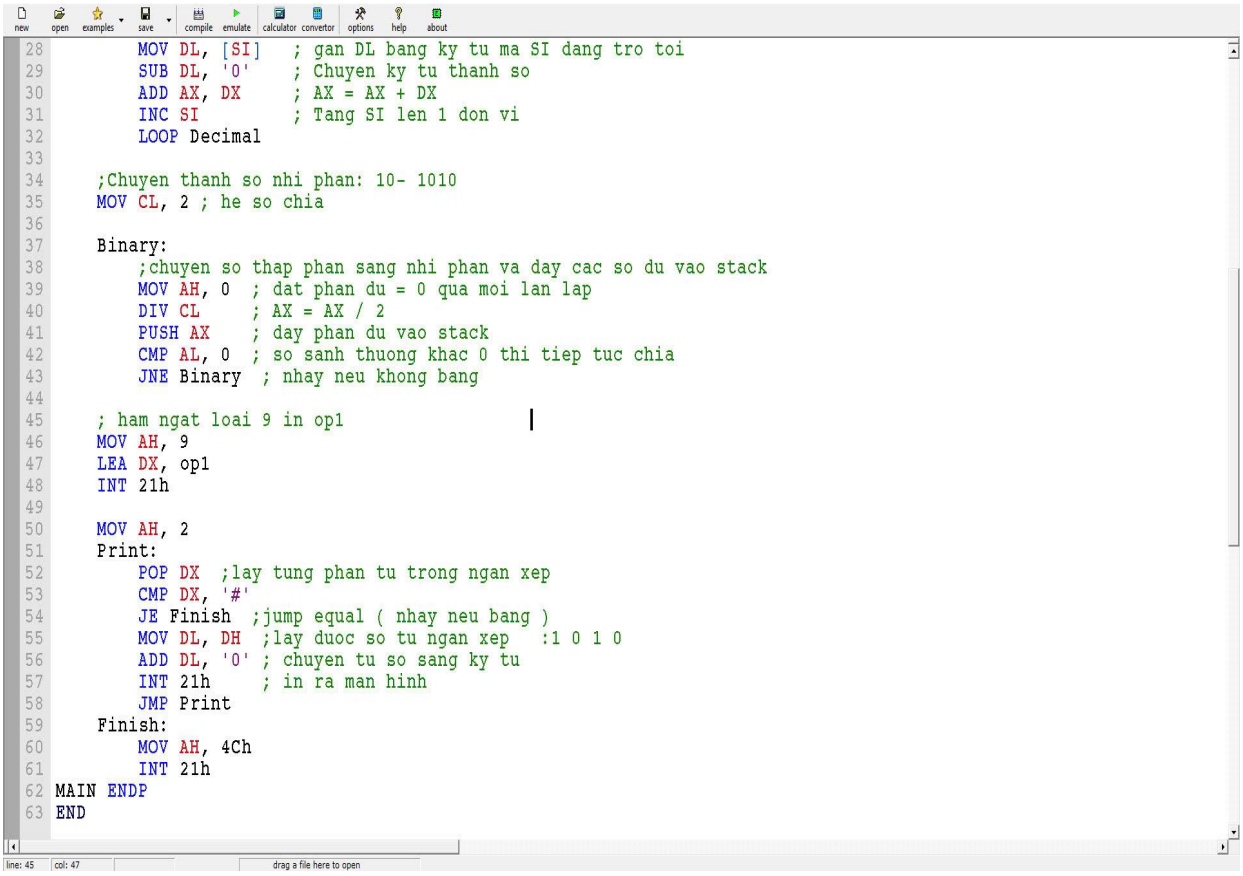


Hình 7.1 : *Flow chart* ***Câu 7***



```
.Model Small
.Stack 100
.Data
    op1 DB 10,13, 'So da nhap dang nhi phan: $'
    str DB 5 dup ('$'); nhap vao 1 chuoi toi da 5 ky tu
.Code
MAIN PROC
    MOV AX, @Data
    MOV DS, AX          ;khoi tao thanh ghi ds

    MOV AX, '#'
    PUSH AX             ;push dau #  vao trong stack

    ;Nhap so dang chuoi:
    MOV AH, 10
    LEA DX, str
    INT 21h

    ;Chuyen chuoi thanh so:
    MOV CL, [str+1]     ; lay so ky tu cua chuoi
    LEA SI, str+2       ; tro den dia chi cua ky tu dau tien cua chuoi str
    MOV AX, 0           ; AX=0
    MOV BX, 10          ; BX=10 la he so nhan

    Decimal:
        ; chuyen chuoi thanh so     123;0*10+1 1*10+2; 12*10+3
        MUL BX          ; AX = AX * 10
        MOV DL, [SI]    ; gan DL bang ky tu ma SI dang tro toi
        SUB DL, '0'     ; Chuyen ky tu thanh so
        ADD AX, DX      ; AX = AX + DX
        INC SI          ; Tang SI len 1 don vi
        LOOP Decimal

    ;Chuyen thanh so nhi phan: 10- 1010
    MOV CL, 2 ; he so chia

    Binary:
```

Hình 7.2 : *Mã nguồn* ***Câu 7***

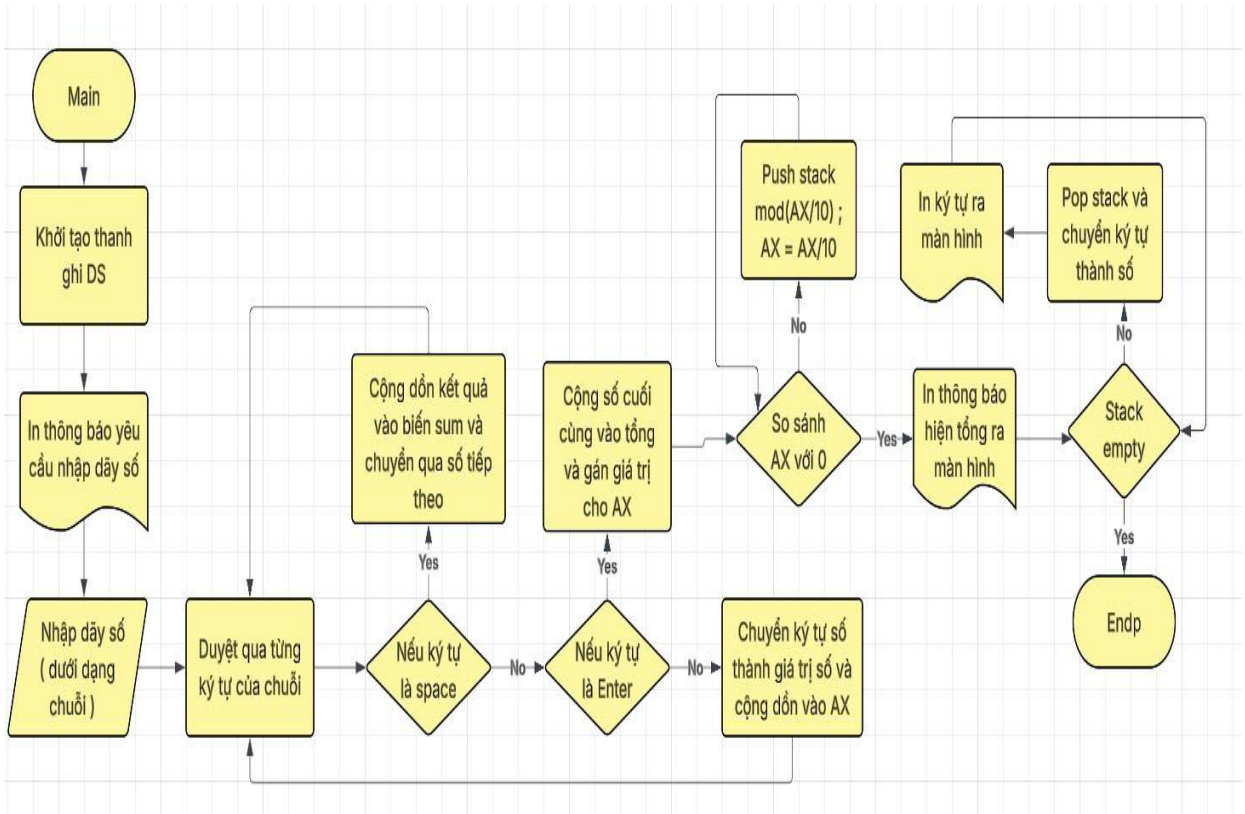


```
        MOV DL, [SI]    ; gan DL bang ky tu ma SI dang tro toi
        SUB DL, '0'     ; Chuyen ky tu thanh so
        ADD AX, DX      ; AX = AX + DX
        INC SI          ; Tang SI len 1 don vi
        LOOP Decimal

    ;Chuyen thanh so nhi phan: 10- 1010
    MOV CL, 2 ; he so chia

    Binary:
        ;chuyen so thap phan sang nhi phan va day cac so du vao stack
        MOV AH, 0  ; dat phan du = 0 qua moi lan lap
        DIV CL     ; AX = AX / 2
        PUSH AX    ; day phan du vao stack
        CMP AL, 0  ; so sanh thuong khac 0 thi tiep tuc chia
        JNE Binary  ; nhay neu khong bang

    ; ham ngat loai 9 in op1
    MOV AH, 9
    LEA DX, op1
    INT 21h

    MOV AH, 2
    Print:
        POP DX  ;lay tung phan tu trong ngan xep
        CMP DX, '#'
        JE Finish  ;jump equal ( nhay neu bang )
        MOV DL, DH  ;lay duoc so tu ngan xep   :1 0 1 0
        ADD DL, '0' ; chuyen tu so sang ky tu
        INT 21h     ; in ra man hinh
        JMP Print
    Finish:
        MOV AH, 4Ch
        INT 21h
MAIN ENDP
END
```

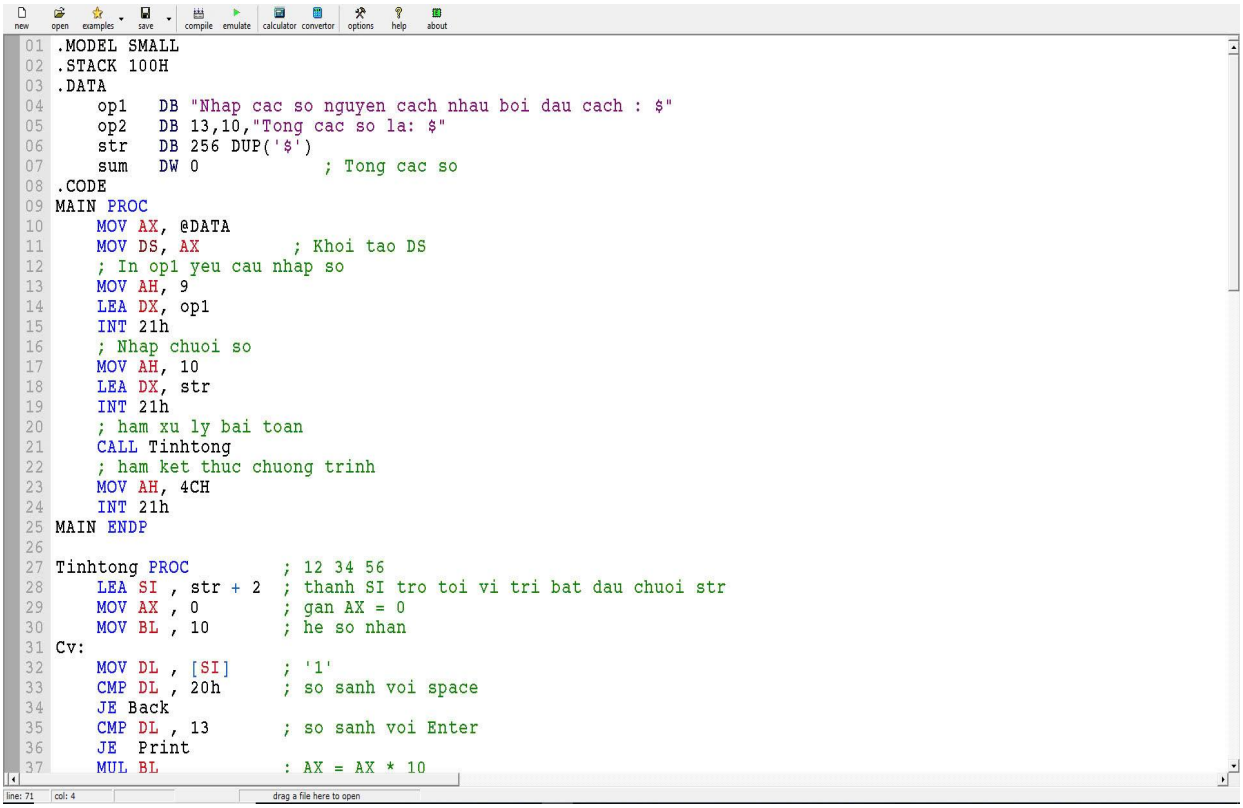
Hình 7.3 : *Mã nguồn* ***Câu 7***

Hình 7.4 : *Giao diện hiển thị* ***Câu 7***

Câu 13 : *Viết chương trình Assembly nhập vào 1 dãy số và in ra tổng của dãy số đó.*

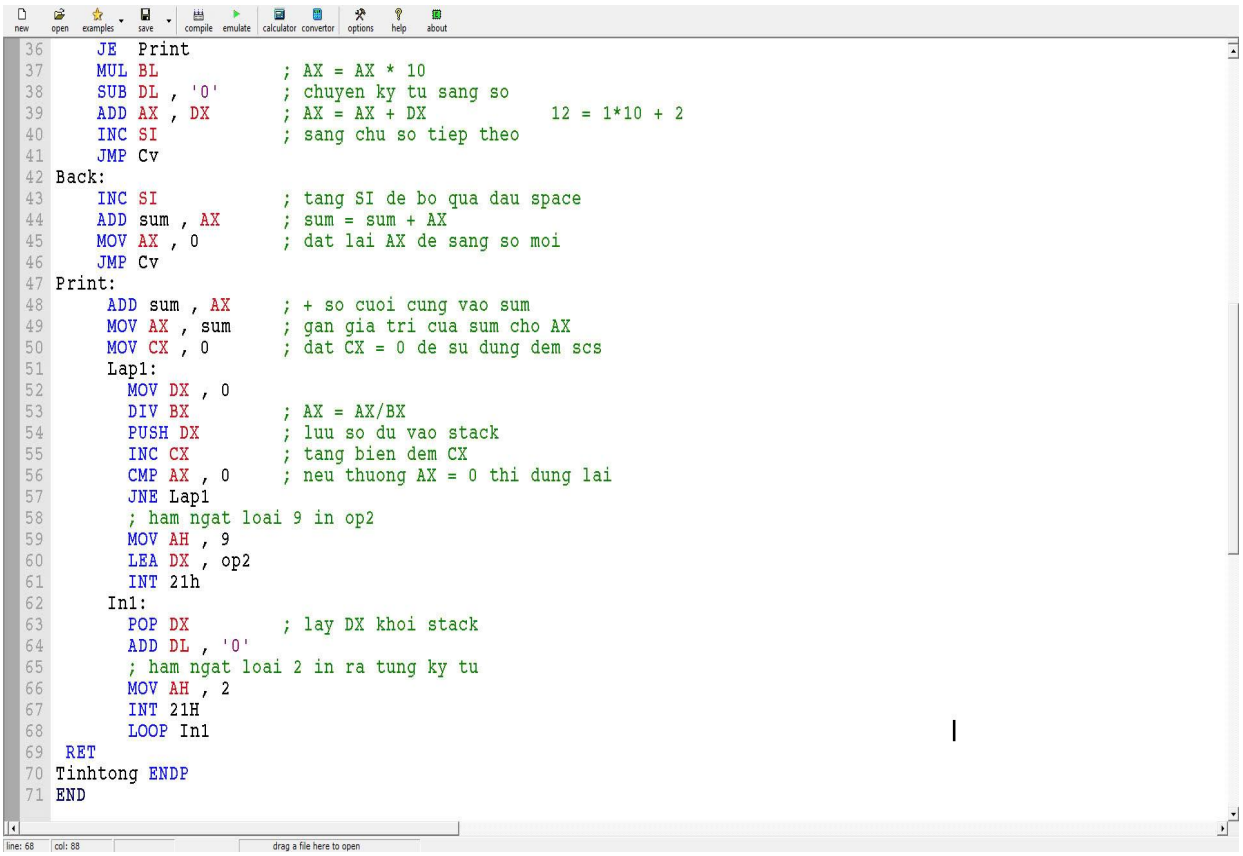


Hình 13.1 : *Flow chart* ***Câu 13***



```
.MODEL SMALL
.STACK 100H
.DATA
    op1   DB "Nhap cac so nguyen cach nhau boi dau cach : $"
    op2   DB 13,10,"Tong cac so la: $"
    str   DB 256 DUP('$')
    sum   DW 0            ; Tong cac so
.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX          ; Khoi tao DS
    ; In op1 yeu cau nhap so
    MOV AH, 9
    LEA DX, op1
    INT 21h
    ; Nhap chuoi so
    MOV AH, 10
    LEA DX, str
    INT 21h
    ; ham xu ly bai toan
    CALL Tinhtong
    ; ham ket thuc chuong trinh
    MOV AH, 4CH
    INT 21h
MAIN ENDP

Tinhtong PROC          ; 12 34 56
    LEA SI , str + 2   ; thanh SI tro toi vi tri bat dau chuoi str
    MOV AX , 0         ; gan AX = 0
    MOV BL , 10        ; he so nhan
Cv:
    MOV DL , [SI]      ; '1'
    CMP DL , 20h       ; so sanh voi space
    JE Back
    CMP DL , 13        ; so sanh voi Enter
    JE  Print
    MUL BL             : AX = AX * 10
```

Hình 13.2 : *Mã nguồn* ***Câu 13***



```
    JE  Print
    MUL BL             ; AX = AX * 10
    SUB DL , '0'       ; chuyen ky tu sang so
    ADD AX , DX        ; AX = AX + DX            12 = 1*10 + 2
    INC SI             ; sang chu so tiep theo
    JMP Cv
Back:
    INC SI             ; tang SI de bo qua dau space
    ADD sum , AX       ; sum = sum + AX
    MOV AX , 0         ; dat lai AX de sang so moi
    JMP Cv
Print:
     ADD sum , AX      ; + so cuoi cung vao sum
     MOV AX , sum      ; gan gia tri cua sum cho AX
     MOV CX , 0        ; dat CX = 0 de su dung dem scs
     Lap1:
       MOV DX , 0
       DIV BX          ; AX = AX/BX
       PUSH DX         ; luu so du vao stack
       INC CX          ; tang bien dem CX
       CMP AX , 0      ; neu thuong AX = 0 thi dung lai
       JNE Lap1
       ; ham ngat loai 9 in op2
       MOV AH , 9
       LEA DX , op2
       INT 21h
     In1:
       POP DX          ; lay DX khoi stack
       ADD DL , '0'
       ; ham ngat loai 2 in ra tung ky tu
       MOV AH , 2
       INT 21H
       LOOP In1
 RET
Tinhtong ENDP
END
```

Hình 13.3 : *Mã nguồn* ***Câu 13***

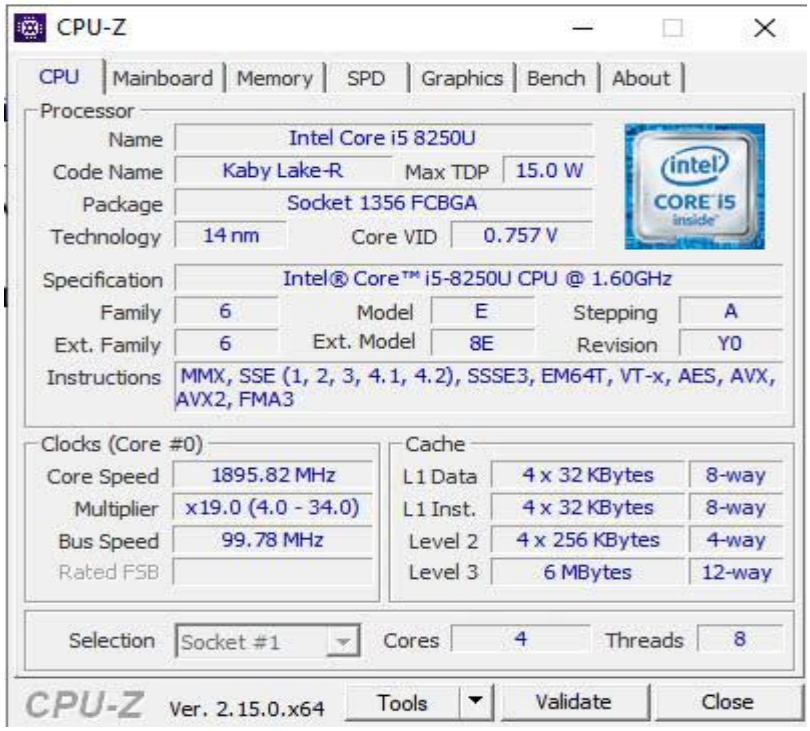
Hình 13.4 : *Giao hiện hiển thị* ***Câu 13***

1.2. Thực hành phân tích khảo sát hệ thống bộ nhớ

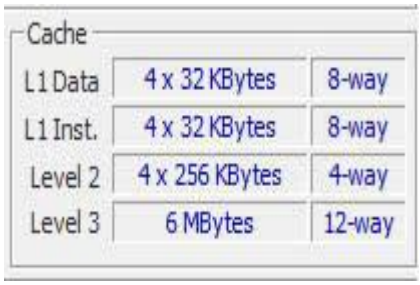
1.2.1 : *Khảo sát cấu hình của máy và hệ thống bộ nhớ của máy đang sử dụng (Bộ nhớ trong: ROM, RAM, Cache System, Bộ nhớ ngoài: ổ đĩa cứng, CD, Thiết bị vào ra.)*

- Sử dụng phần mềm CPU-Z 64-bit Ver 2.15.0 x64
- Sử dụng Task Manager

CPU Intel Core i5 8250U 4 nhân 8 luồng

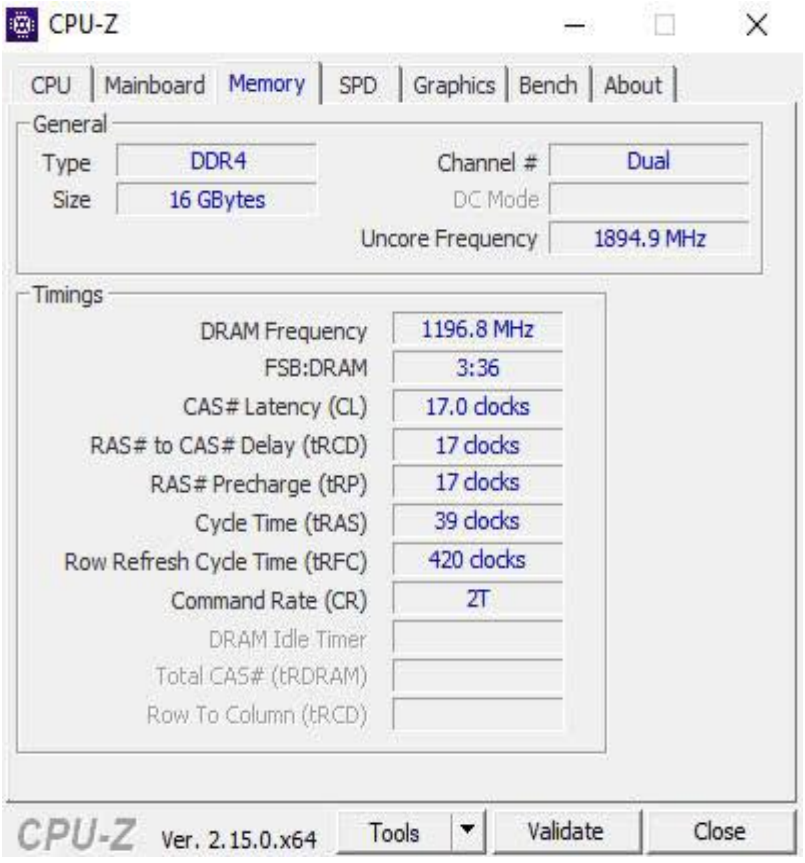


Hình 1 : *Giao diện phần mềm CPU-Z*

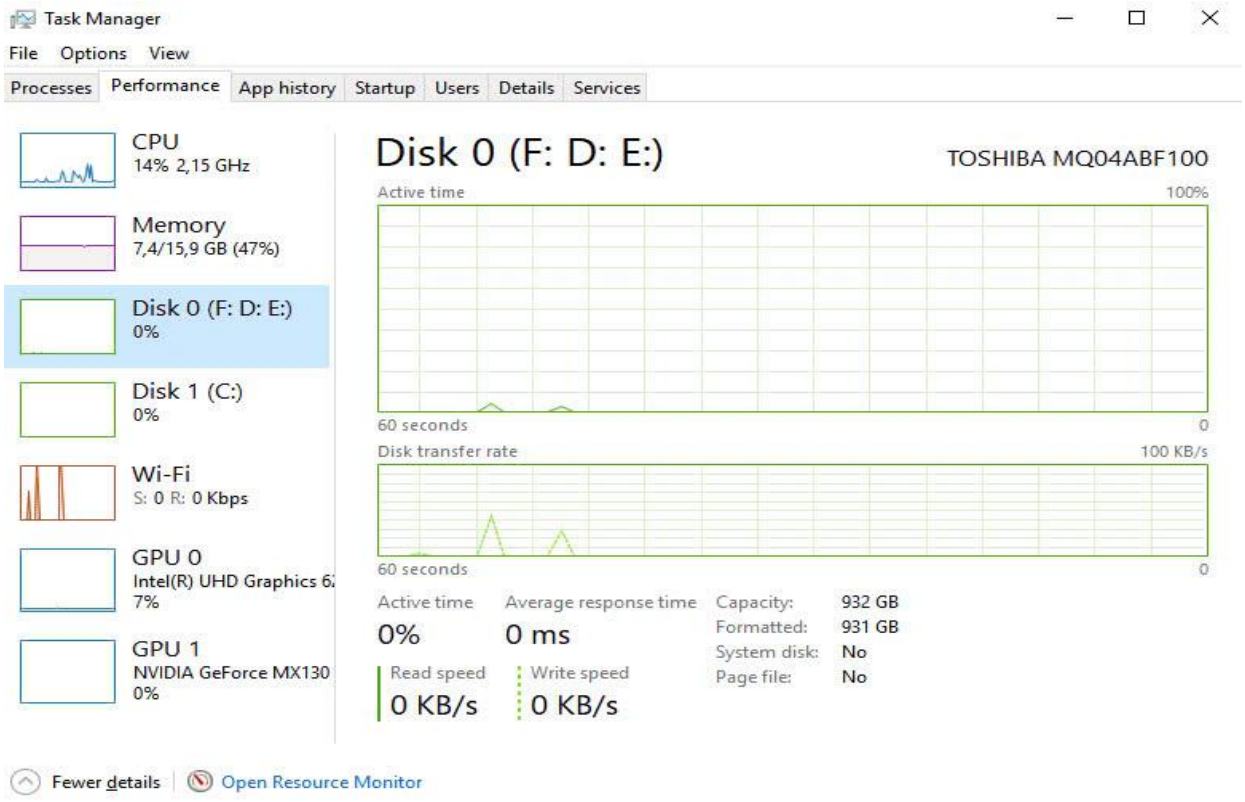


Hình 2 : *Cache*

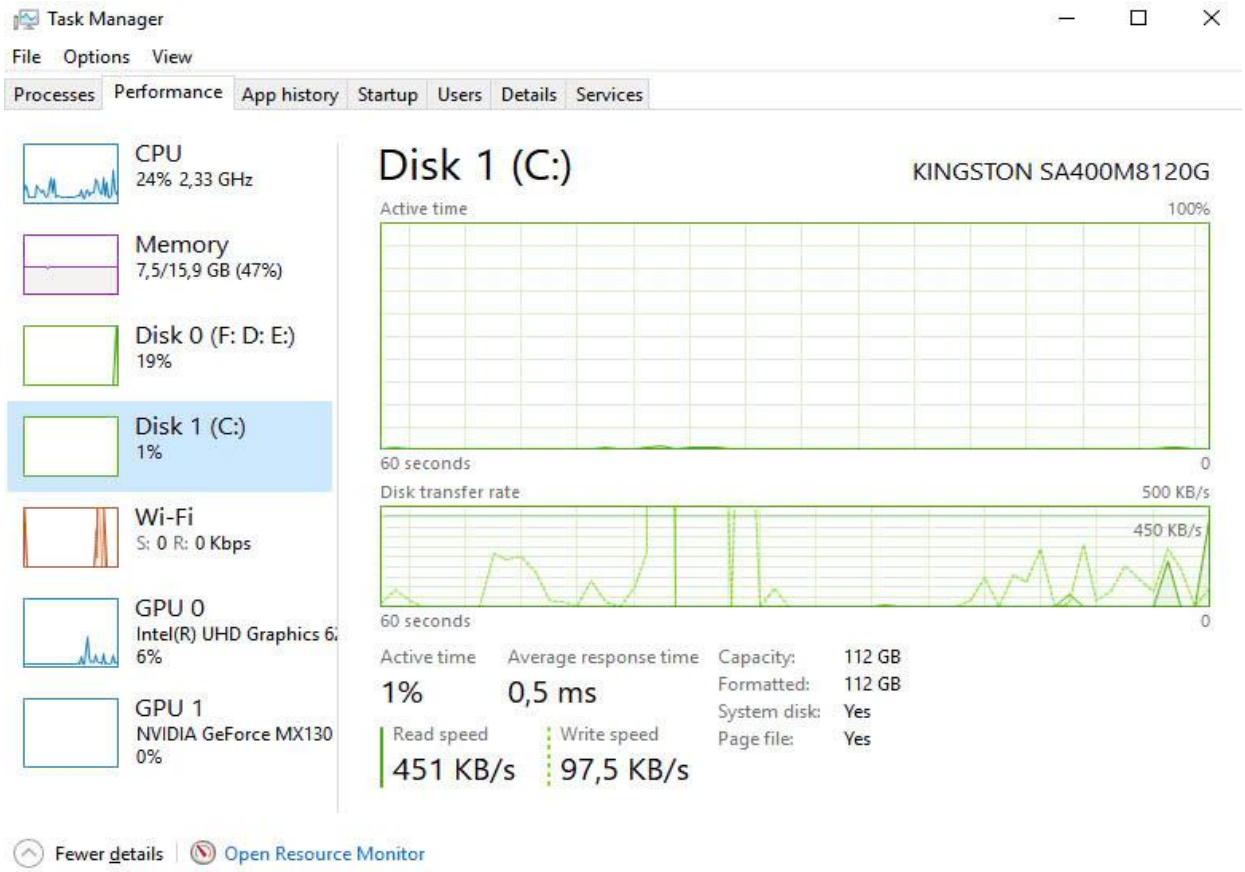
Hình 3 : *Thông tin về ROM*



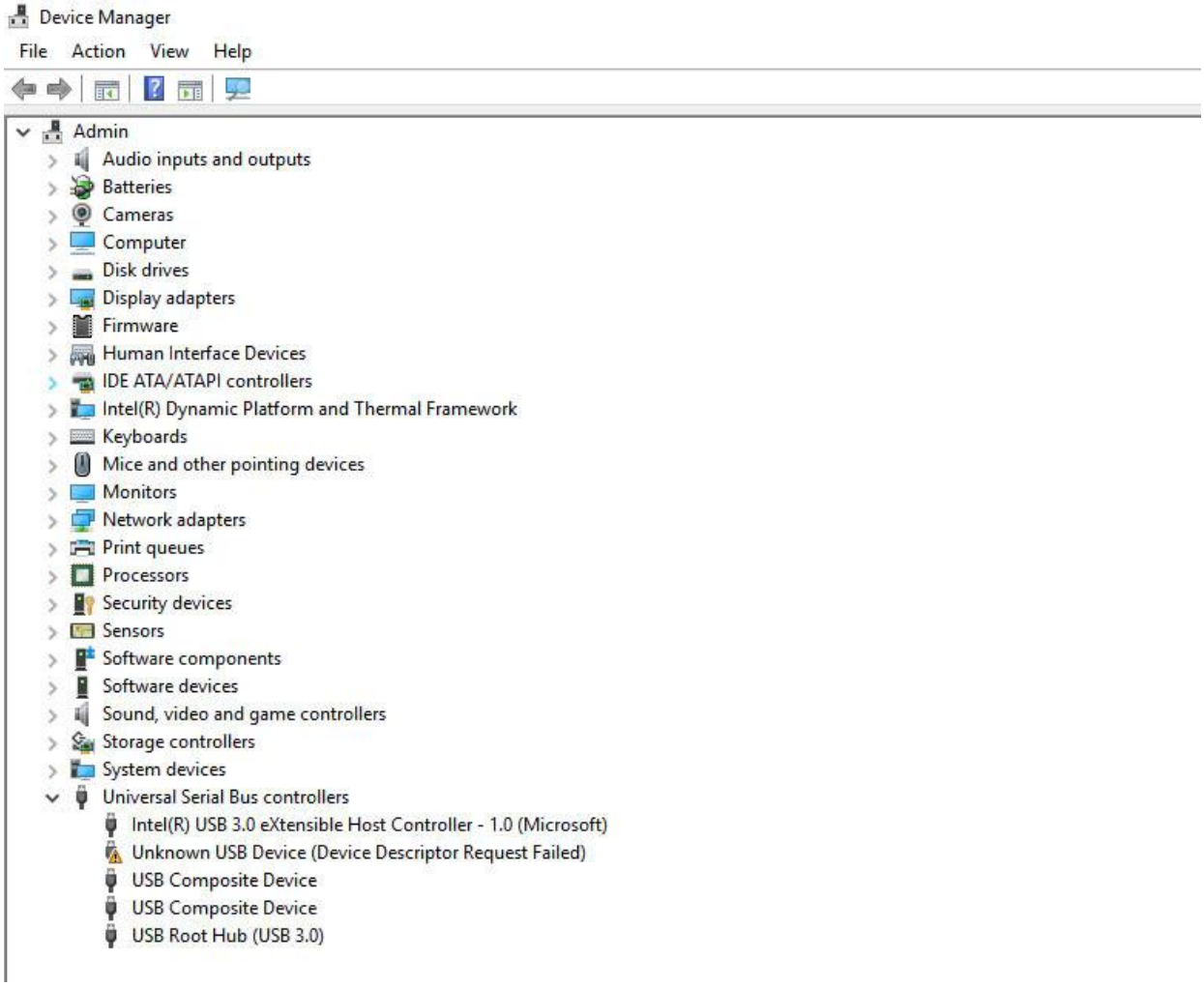
Hình 4 : *Thông tin về RAM 16GB*



Hình 5 : *Bộ nhớ ngoài (ổ đĩa F + D + E) HDD 1TB*



Hình 6 : *Bộ nhớ ngoài (ổ đĩa C) SSD 128GB*



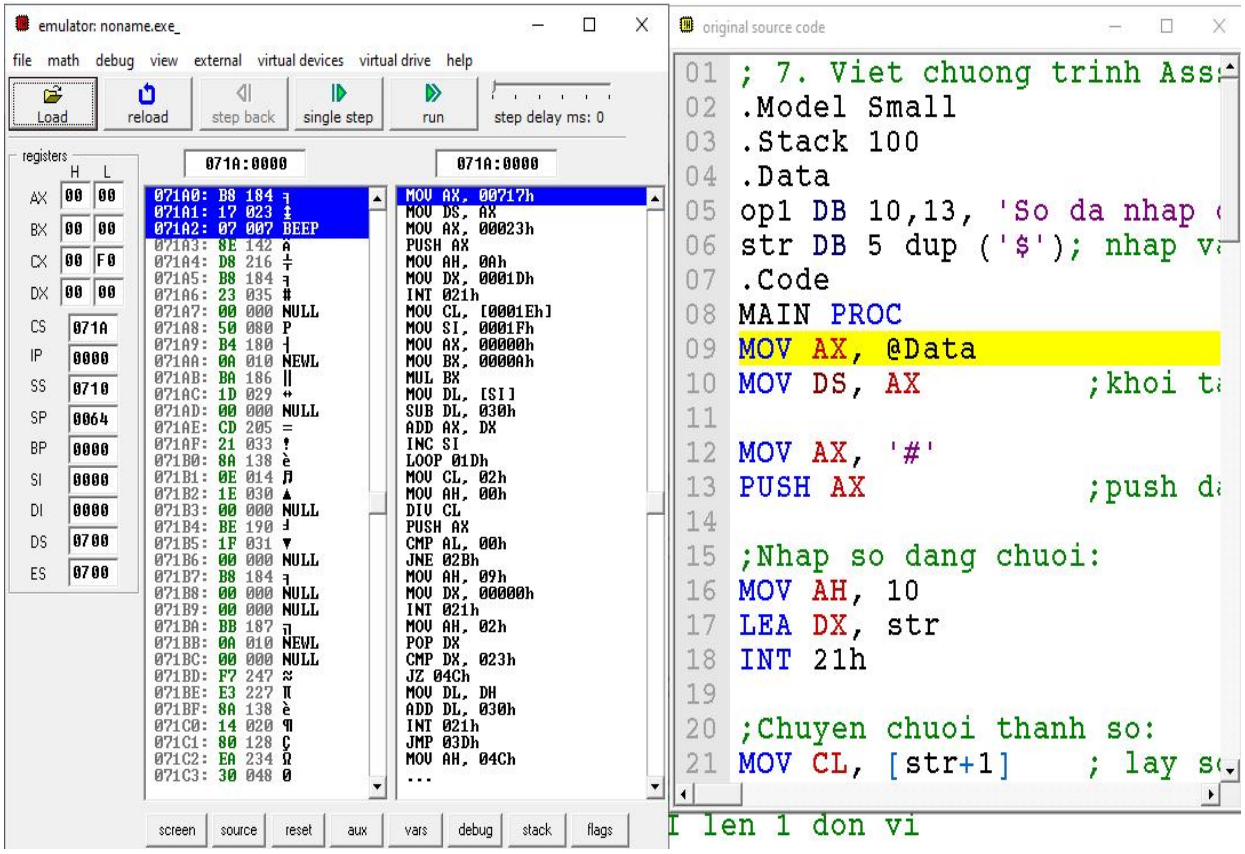
***Hình** 7 :* *Các thiết bị vào ra*

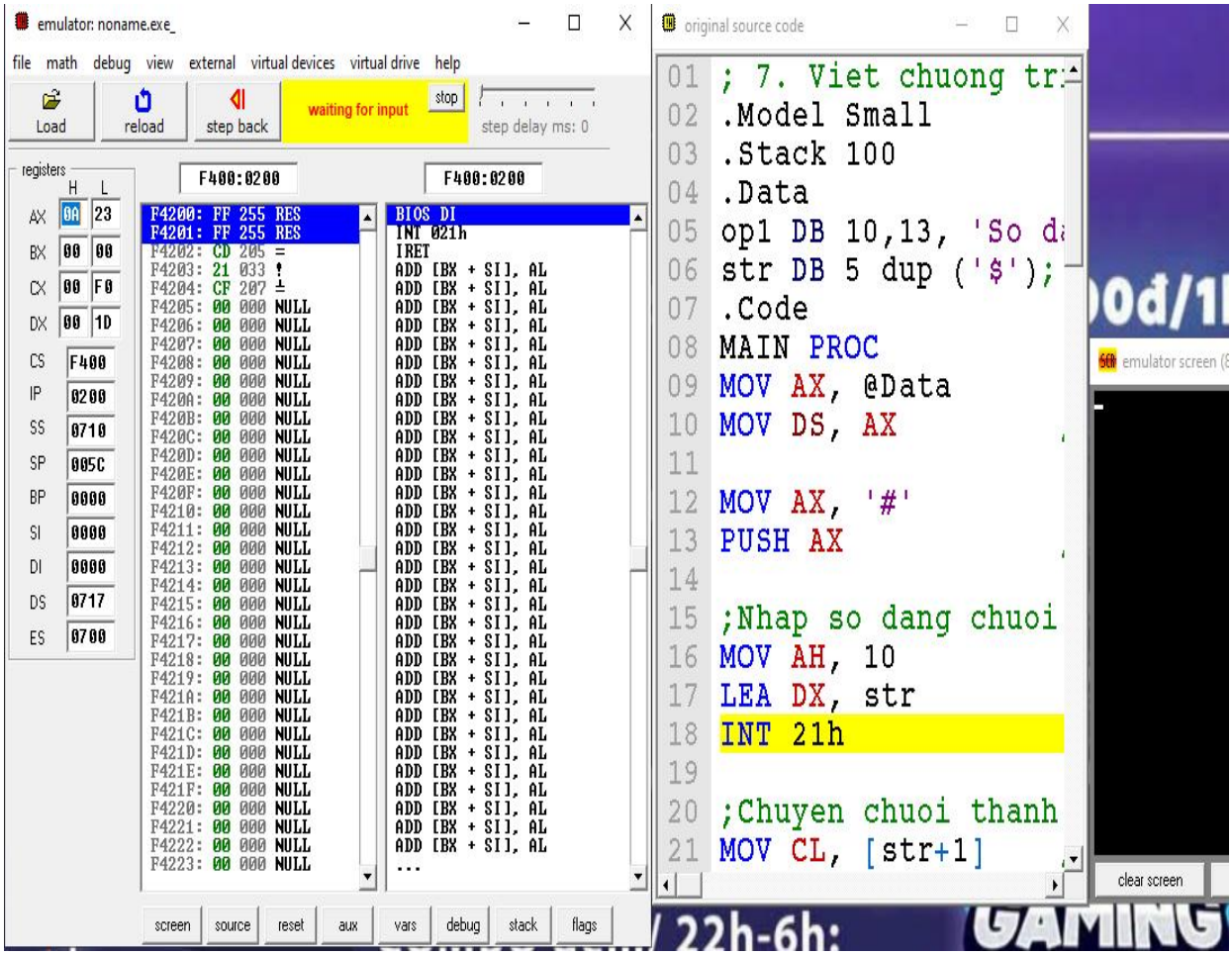
1.2.2 : *Dùng công cụ Debug khảo sát nội dung các thanh ghi IP, DS, ES, SS, CS, BP, SP*

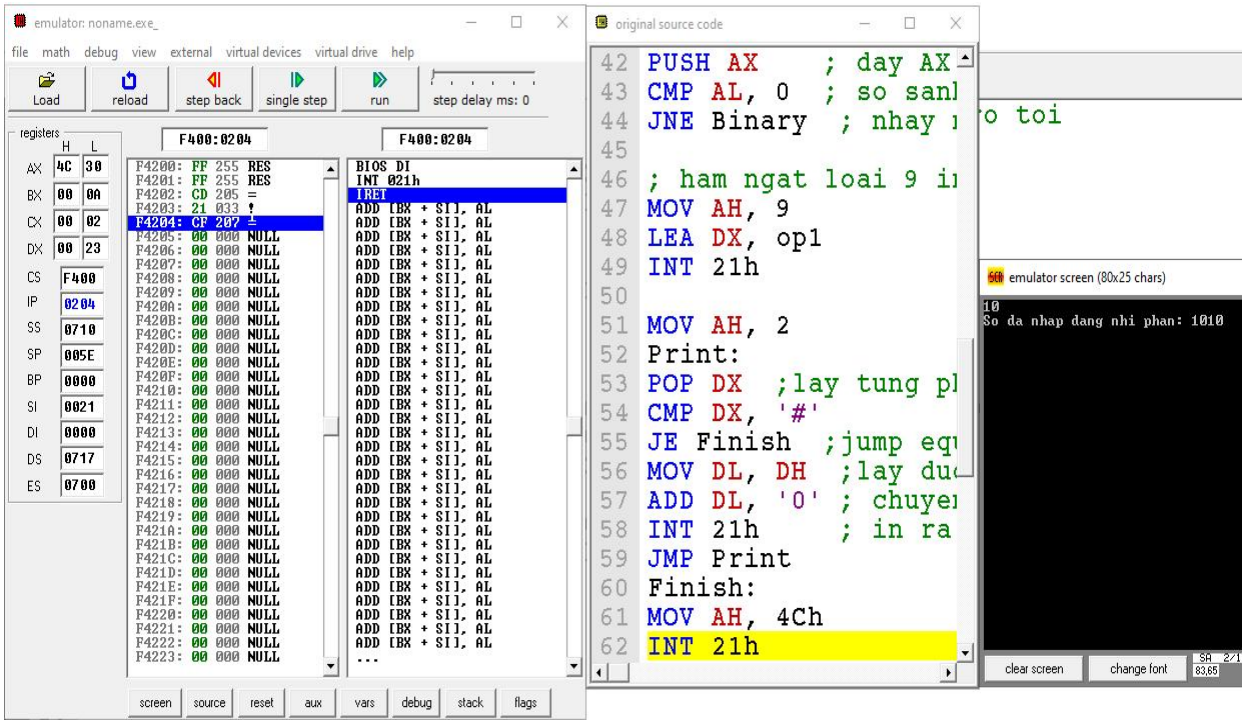
- Phần mềm sử dụng : Emu8086 microprocessor emulator
- Các bước thực hiện :
  + Mở file .asm bằng phần mềm trên
  + Trên thanh công cụ , chọn nút emulate
  + Sau khi cửa sổ .exe hiện ra , sử dụng nút single step (chạy từng lệnh) để theo dõi các thanh ghi

Dưới đây là hình ảnh các thanh ghi thay đổi khi nhập vào số thập phân và in nó ra màn hình dưới dạng nhị phân

- Hình 1 khi khởi tạo thanh ghi DS
- Hình 2 khi bắt đầu nhập input
- Hình 3 khi in ra output


emulator: noname.exe_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 00
BX 00 00
CX 00 F0
DX 00 00
CS 071A
IP 0000
SS 0710
SP 0064
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
071A:0000
071A:0000
071A0: B8 184 ╕
071A1: 17 023 ↨
071A2: 07 007 BEEP
071A3: 8E 142 Ä
071A4: D8 216 ╪
071A5: B8 184 ╕
071A6: 23 035 #
071A7: 00 000 NULL
071A8: 50 080 P
071A9: B4 180 ┤
071AA: 0A 010 NEWL
071AB: BA 186 ║
071AC: 1D 029 ↔
071AD: 00 000 NULL
071AE: CD 205 ═
071AF: 21 033 !
071B0: 8A 138 è
071B1: 0E 014 ♫
071B2: 1E 030 ▲
071B3: 00 000 NULL
071B4: BE 190 ╛
071B5: 1F 031 ▼
071B6: 00 000 NULL
071B7: B8 184 ╕
071B8: 00 000 NULL
071B9: 00 000 NULL
071BA: BB 187 ╗
071BB: 0A 010 NEWL
071BC: 00 000 NULL
071BD: F7 247 ≈
071BE: E3 227 π
071BF: 8A 138 è
071C0: 14 020 ¶
071C1: 80 128 Ç
071C2: EA 234 Ω
071C3: 30 048 0
MOV AX, 00717h
MOV DS, AX
MOV AX, 00023h
PUSH AX
MOV AH, 0Ah
MOV DX, 0001Dh
INT 021h
MOV CL, [0001Eh]
MOV SI, 0001Fh
MOV AX, 00000h
MOV BX, 0000Ah
MUL BX
MOV DL, [SI]
SUB DL, 030h
ADD AX, DX
INC SI
LOOP 01Dh
MOV CL, 02h
MOV AH, 00h
DIV CL
PUSH AX
CMP AL, 00h
JNE 02Bh
MOV AH, 09h
MOV DX, 00000h
INT 021h
MOV AH, 02h
POP DX
CMP DX, 023h
JZ 04Ch
MOV DL, DH
ADD DL, 030h
INT 021h
JMP 03Dh
MOV AH, 04Ch
...
screen source reset aux vars debug stack flags
original source code
01 ; 7. Viet chuong trinh Ass
02 .Model Small
03 .Stack 100
04 .Data
05 op1 DB 10,13, 'So da nhap
06 str DB 5 dup ('$'); nhap v
07 .Code
08 MAIN PROC
09 MOV AX, @Data
10 MOV DS, AX ;khoi t
11
12 MOV AX, '#'
13 PUSH AX ;push d
14
15 ;Nhap so dang chuoi:
16 MOV AH, 10
17 LEA DX, str
18 INT 21h
19
20 ;Chuyen chuoi thanh so:
21 MOV CL, [str+1] ; lay s
I len 1 don vi


emulator: noname.exe_
file math debug view external virtual devices virtual drive help
Load
reload
step back
waiting for input
stop
step delay ms: 0
registers
H L
AX 0A 23
BX 00 00
CX 00 F0
DX 00 1D
CS F400
IP 0200
SS 0710
SP 005C
BP 0000
SI 0000
DI 0000
DS 0717
ES 0700
F400:0200
F400:0200
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 ═
F4203: 21 033 !
F4204: CF 207 ╧
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
F4216: 00 000 NULL
F4217: 00 000 NULL
F4218: 00 000 NULL
F4219: 00 000 NULL
F421A: 00 000 NULL
F421B: 00 000 NULL
F421C: 00 000 NULL
F421D: 00 000 NULL
F421E: 00 000 NULL
F421F: 00 000 NULL
F4220: 00 000 NULL
F4221: 00 000 NULL
F4222: 00 000 NULL
F4223: 00 000 NULL
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
...
screen source reset aux vars debug stack flags
original source code
01 ; 7. Viet chuong tr
02 .Model Small
03 .Stack 100
04 .Data
05 op1 DB 10,13, 'So d
06 str DB 5 dup ('$');
07 .Code
08 MAIN PROC
09 MOV AX, @Data
10 MOV DS, AX
11
12 MOV AX, '#'
13 PUSH AX
14
15 ;Nhap so dang chuoi
16 MOV AH, 10
17 LEA DX, str
18 INT 21h
19
20 ;Chuyen chuoi thanh
21 MOV CL, [str+1]
emulator screen
clear screen
22h-6h:
GAMING



1.2.3 *: Giải thích nội dung các thanh ghi, trên cơ sở đó giải thích cơ chế quản lý bộ nhớ của hệ thống trong trường hợp cụ thể này.*

- Khi chương trình bắt đầu chạy, hệ điều hành tự động khởi tạo các thanh ghi , vùng nhớ và cấp phát không gian địa chỉ cho chương trình
- Tương ứng với các câu lệnh trong mã nguồn, nội dung các thanh ghi có thể thay đổi hoặc không

*** IP (Instruction Pointer) - Thanh ghi con trỏ lệnh :**

+ IP là thanh ghi trỏ đến địa chỉ của lệnh tiếp theo sẽ được thực thi trong mã máy

+ Khi một chương trình được thực thi, IP được cập nhật để trỏ đến lệnh tiếp theo trong mã nguồn. Điều này giúp CPU biết lệnh nào sẽ được thực thi tiếp theo.

*** CS (Code Segment) - Thanh ghi đoạn mã :**

+ CS chứa địa chỉ cơ sở của đoạn code

+ Khi CPU cần thực thi lệnh , CS kết hợp với IP để lấy lệnh từ bộ nhớ .

*** DS (Data Segment) - Thanh ghi đoạn dữ liệu :**

+ DS chứa địa chỉ cơ sở của đoạn dữ liệu

+ Dùng để truy xuất dữ liệu (biến , mảng ,chuỗi) với các thanh ghi khác như BX , SI , DI

*** ES (Extra Segment) - Thanh ghi đoạn mở rộng :**

+ Dùng bổ sung cho DS để truy cập dữ liệu ở đoạn khác

+ Thường dùng trong các thao tác liên quan tới chuỗi

*** SS (Stack Segment) - Thanh ghi đoạn ngăn xếp :**

+ Chứa địa chỉ cơ sở của stack

+ Kết hợp với SP/BP để quản lý ngăn xếp

*** SP (Stack Pointer) - Thanh ghi con trỏ ngăn xếp :**

+ Trỏ tới đỉnh stack

+ Tăng giảm tự động khi push/pop

*** BP (Base Pointer) - Thanh ghi gốc ngăn xếp**

+ Trỏ tới địa chỉ cơ sở của frame hàm

+ Thường dùng với thanh SS và giữ nguyên khi thực hiện lệnh trong hàm -> dễ truy xuất giá trị từ stack

***** Cụ thể . trong đoạn chương trình chuyển 1 số hệ thập phân sang hệ nhị phân , ta có cơ chế quản lý bộ nhớ như sau :**

1. **Data Segment (DS) - chứa dữ liệu tĩnh**

```
.Data
    op1 DB 10,13, 'So da nhap dang nhi phan: $'
    str DB 5 dup ('$'); nhap vao 1 chuoi toi da 5 ky tu
```

Các dữ liệu trong .data được trỏ tới bởi DS sau đoạn lệnh :

```
MOV AX, @Data
MOV DS, AX          ;khoi tao thanh ghi ds
```

2. **Code Segment (CS) - chứa mã lệnh chương trình**

- Tất cả mã lệnh (MOV, MUL, DIV, LOOP...) được nạp vào **đoạn mã (CS)** và thực thi tại đó.

.Code
MAIN PROC
...
MAIN ENDP
END

3. **Stack Segment (SS) - trỏ tới đoạn ngăn xếp**

- Khai báo vùng ngăn xếp có kích thước 100h : .Stack 100
- Dùng để lưu kết quả nhị phân tạm thời

Cơ chế sử dụng :

Lệnh	Tác động đến stack
PUSH AX	Giảm SP, ghi AX vào stack
POP DX	Lấy giá trị từ stack, tăng SP

4. **Các thanh ghi và bộ nhớ tạm thời**

Thanh ghi	Chức năng quản lý bộ nhớ
AX	Dùng lưu trữ giá trị số thập phân và kết quả chia
BX	Hệ số nhân (10) trong chuyển chuỗi → số
CX	Đếm vòng lặp (khi duyệt chuỗi và chia nhị phân)
DX	Lưu giá trị in ra màn hình
SI	Trỏ vào địa chỉ chuỗi nhập (str+2)
SP	Quản lý đỉnh ngăn xếp

PHẦN 2 : BÀI TẬP NHÓM

ĐỀ TÀI : LẬP TRÌNH GAME TIC TAC TOE TRÊN EMU8086

2.1 : Giới thiệu đề tài

Tic Tac Toe (còn được gọi là Cờ ca-rô) là một trò chơi cổ điển mà hầu như ai cũng biết đến.Với luật chơi đơn giản nhưng cũng không hề kém sự thách thức , trò chơi cũng là một bài toán trí tuệ với chiến thuật và tư duy logic

Trong Tic Tac Toe, hai người chơi sẽ thay phiên nhau đặt dấu (X hoặc O) vào các ô trên bảng 3x3, với mục tiêu tạo ra một chuỗi ba dấu liên tiếp theo hàng ngang, hàng dọc hoặc đường chéo.Người chiến thắng là người tạo ra được chuỗi ba dấu liên tiếp trước đối thủ , hoặc trò chơi sẽ kết thúc với kết quả hòa nếu bảng 3x3 không còn vị trí đánh .

Emu8086 là một trình mô phỏng máy tính sử dụng vi xử lý Intel 8086. Nó cho phép lập trình và chạy các đoạn mã hợp ngữ (Assembly) trong môi trường mô phỏng, giúp người học và nhà phát triển hiểu rõ hơn về cách hoạt động của vi xử lý và hệ thống máy tính thời kỳ đầu. Đề tài "Lập trình game Tic Tac Toe trên Emu8086" tập trung vào việc phát triển một phiên bản của trò chơi Tic Tac Toe trong môi trường Emu8086.

Dưới đây , chúng em (Trần Đức Trung , Trần Văn Hiếu , Nguyễn Hữu Hiếu , Nguyễn Thế Bình) xin phép trình bày chi tiết về *Đề tài : Lập trình game Tic Tac Toe trên Emu8086*

2.2 : Nội dung chính của đề tài

2.2.1 : Tổng quan về trò chơi

- Tic Tac Toe là trò chơi dành cho 2 người chơi , với bàn cờ 3x3 . Mỗi người chơi lần lượt đánh nước đi của mình lên trên bàn cờ , một người đánh “X” và người còn lại đánh “O”

- Mục tiêu của trò chơi là tạo ra một chuỗi ba dấu hiệu liên tiếp theo hàng ngang, hàng dọc hoặc đường chéo. Người chơi đầu tiên đạt được mục tiêu này sẽ là người chiến thắng. Nếu điền hết bàn cờ mà không tìm được ra người chiến thắng sẽ có kết quả là hòa.

- Cách chơi :

+ Người chơi thứ nhất sẽ bắt đầu bằng cách đặt dấu "X" vào một ô bất kỳ trên bàn cờ.Người chơi thứ hai sẽ tiếp tục với dấu “O” .

+ Người chơi thay phiên nhau đặt dấu hiệu của mình cho đến khi một người chiến thắng hoặc bàn cờ đầy mà không có ai thắng, dẫn đến kết quả hòa.

- Các chức năng có trong trò chơi :

+ Tạo ra bàn cờ 3x3 cho phép người chơi chọn số từ 1 đến 9 tương ứng với các ô trên bảng, điền ‘X’ hoặc ‘O’ vào ô đó.

+ Nếu người chơi nhập vào ô đã được điền trong bảng , hoặc nhập ký tự ngoài khoảng [1,9] , màn hình sẽ in ra thông báo nhập sai và yêu cầu người chơi nhập lại

+ Nếu có người chiến thắng (xuất hiện 1 chuỗi 3 dấu XXX hoặc OOO liên tiếp theo hàng ngang, hàng dọc hoặc đường chéo), sẽ in ra thông báo người chiến thắng.

+ Nếu bàn cờ đầy mà không có ai chiến thắng, thông báo cho biết kết quả là hòa.

2.2.2 : Giải thuật của chương trình

- ***Cấu trúc chính của chương trình :*** Chuỗi các thủ tục phối hợp với nhau để thao tác chơi game luân phiên (lần lượt đánh X và O) trên mảng 1 chiều 3x3 (9 phần tử)

- Thuật toán chương trình :

Sử dụng 1 vòng lặp (tối đa 9 lần) để điều khiển luồng chương trình . Trong vòng lặp này có các thủ tục sau :

+ **Xác định lượt chơi :** sử dụng biến turn để kiểm tra xem hiện tại là lượt của người chơi nào

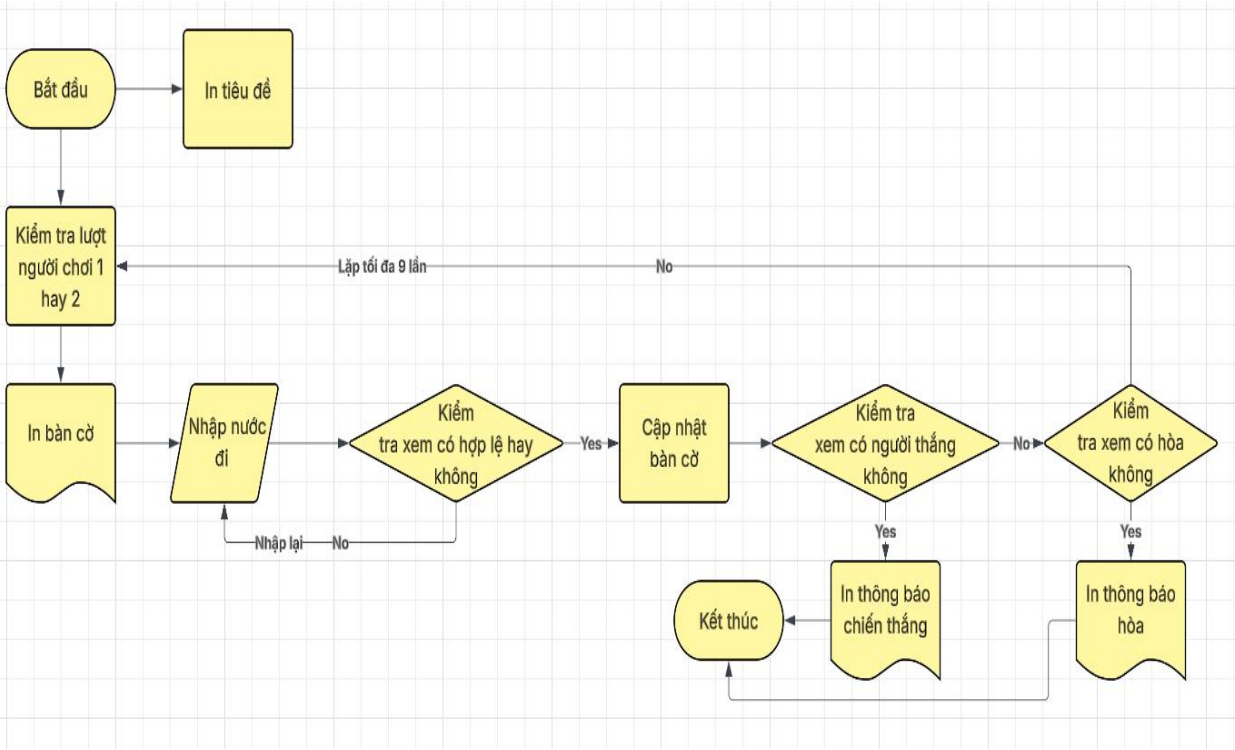
+ **Hiển thị bàn cờ :** gọi 1 hàm con (hàm map) để hiển thị bàn cờ chứa các nước đi từ trước

+ **Nhận dữ liệu từ người chơi :** nhận nước đi từ người chơi và cập nhật vào bàn cờ cho lượt sau

+ **Kiểm tra điều kiện thắng / hòa :** gọi hàm con kiểm tra điều kiện thắng / hòa .Nếu có người chiến thắng hoặc bàn cờ đầy thì in ra thông báo chiến thắng hoặc thông báo hòa , còn không thì quay lại cho lần lặp tiếp theo .

2.2.3 : Miêu tả chương trình

Lưu đồ thuật toán (Flow chart) của chương trình :



Phân tích chương trình : https://ideone.com/gx3R4p

```
.MODEL small           ; che do bo nho small
.STACK 100h           ; kich thuoc ngan xep 100
.DATA
      CRLF DB 13,10,"$"  ; dau xuong dong
      space DB 32,"$" ; dau cach
      pipe DB " | $"  ; dau ke doc
      line DB 13,10,"-----------",13,10,32,"$"   ;

      arr DB 32 , 32 , 32
          DB 32 , 32 , 32
          DB 32 , 32 , 32

     ; chuoi in ra ten game
     tictac DB "---------------------------",13,10
           DB "|       TIC TAC TOE       |",13,10
           DB "---------------------------",13,10,13,10
           DB "Player 1  : X   Player 2 : O",13,10,13,10,"$"

     ; chuoi in ra luot cua 2 nguoi choi
     play1 DB 13,10,13,10,"Player 1 turn : ",13,10,13,10,"$"
     play2 DB 13,10,13,10,"Player 2 turn : ",13,10,13,10,"$"

     ;chuoi in ra cau lenh nhap nuoc di
     pos DB "Enter your move (1-9) : $"

     ; chuoi in ra nguoi chien thang
     winner1 DB 13,10,13,10,"Player 1 is winner !$"
     winner2 DB 13,10,13,10,"Player 2 is winner !$"
     draw    DB 13,10,13,10,"The game is a draw !$"

     ; chuoi thong bao khi nhap input sai
     invalid DB 13,10,"Invalid input! Please enter 1-9.",13,10,"$"
     ocp DB 13,10,"This cell is already occupied! Please choose another one.",13,10,"$"

     ; cac bien de dieu khien tro choi
     turn DB 1 ; danh dau luot choi
     endg DB 0  ; bien danh dau trang thai ket thuc game
     char DB 88 ; ky tu cua luot choi ( bat dau = X co ma ASCII 88 )

```

- **.MODEL small:** Chế độ bộ nhớ nhỏ, chương trình có tối đa 64KB mã và 64KB dữ liệu.

- **.STACK 100h:** Khởi tạo ngăn xếp với kích thước 256 byte (100h = 256).
- **.DATA:** Bắt đầu khai báo vùng dữ liệu

CRLF : dấu xuống dòng

Pipe , line , space : các chuỗi để tạo bảng và giãn cách

Mảng **arr** 3x3 đại diện cho bảng Tic Tac Toe với khởi tạo ban đầu là bảng trắng.

Chuỗi **tictac** để in ra tiêu đề game .

Play1 , play2 : chuỗi in ra lượt của 2 người chơi

Pos : chuỗi in ra câu lệnh nhập nước đi

Winner1 , winner2 , draw : chuỗi in ra người thắng hoặc thông báo hòa

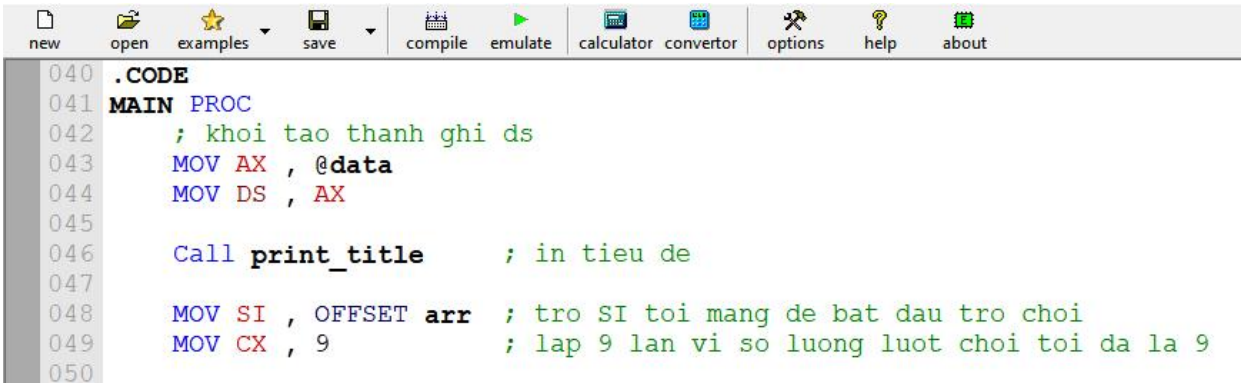
Invalid , ocp : chuỗi thông báo nhập input sai

Các biến điều khiển trò chơi :

Turn : turn = 1 là lượt của player 1 , turn = 0 là lượt của player 2

Endg : endg = 0 là trò chơi vẫn tiếp tục , endg = 1 là có người thắng , endg = 2 là kết quả hòa

Char : ký tự của lượt chơi , khởi tạo ban đầu là ký tự X cho player 1




```
040 .CODE
041 MAIN PROC
042     ; khoi tao thanh ghi ds
043     MOV AX , @data
044     MOV DS , AX
045
046     Call print_title     ; in tieu de
047
048     MOV SI , OFFSET arr  ; tro SI toi mang de bat dau tro choi
049     MOV CX , 9           ; lap 9 lan vi so luong luot choi toi da la 9
050
```

- **MAIN** : bắt đầu chương trình
- Khởi tạo thanh ghi DS và cho DS trỏ tới .data
- Gọi hàm con **print_title** để in tiêu đề bằng hàm ngắt loại 9 :

```
127
128 ; ham print_title de in tieu de game
129 print_title PROC
130     MOV AX , @data
131     MOV DS , AX
132     ;ham ngat loai 9 de in ra tieu de
133     MOV AH , 9
134     MOV DX , OFFSET tictac
135     INT 21h
136     ret
137 print_title ENDP
138
```

- Trỏ thanh ghi SI tới mảng arr để bắt đầu trò chơi
- Gán CX =9 để thực hiện vòng lặp game , tối đa 9 lần

```
; vong lap luan phien luot choi
game_loop:
   MOV AL , turn
   CMP AL , 1           ; neu turn = 1 thi la luot cua player 1
   JE call_player_1
   call player_2        ; neu turn != 1 thi goi ham con player_2
   JMP done             ; nhay toi label done de goi map de in ban co

call_player_1:
   call player_1

done:
   call map
```

- **game_loop** là vòng lặp điều khiển luồng chơi
- Đầu tiên , **MOV AL , turn** để gán giá trị của turn cho AL
- Nếu turn = 1 thì là lượt của player 1 , nhảy tới label **call_player_1** để gọi hàm con **player_1**

```
138
139 ;ham player_1 co tac dung chuan bi cho luot danh cua player 1
140 player_1 PROC
141     MOV AX , @data
142     MOV DS , AX
143
144     ; in chuoi play1
145     MOV AH , 9
146     LEA DX , play1
147     INT 21h
148
149     MOV char , 88    ; dat char = 'X'
150     MOV turn , 0     ; doi gia tri turn de toi luot player 2
151     ret
152 player_1 ENDP
153
```

Hàm **player_1** in chuỗi play1 để báo hiệu tới lượt của người chơi 1 , đặt biến **char** là ký tự X để cập nhật bàn cờ , sau đó chuyển biến **turn** = 0 để tới lượt người chơi 2 .

- Nếu **turn** = 0 thì là lượt của player 2 , gọi hàm con **player_2**

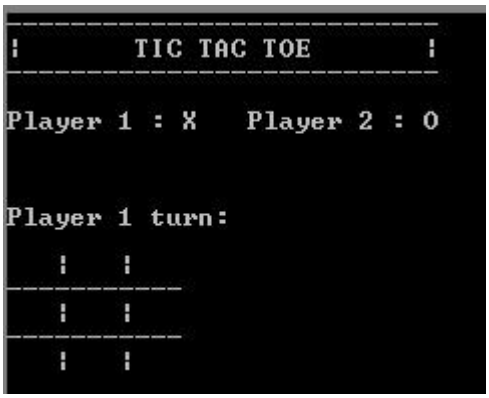
Hàm **player_2** in chuỗi play2 để báo hiệu tới lượt của người chơi 2 , đặt biến **char** là ký tự O để cập nhật bàn cờ , sau đó chuyển biến **turn** = 1 để tới lượt người chơi 1.

- Sau khi gọi hàm **player_1/player_2** , lệnh đều nhảy tới nhãn **done** gọi hàm con **map** để in trạng thái bàn cờ

```
167
168 ; ham map in ra trang thai ban co hien tai
169 map PROC
170     MOV AX , @data
171     MOV DS , AX
172
173     MOV SI , OFFSET arr  ; tro toi ban co
174
175     MOV AH , 9
176     LEA DX , space       ; in dau cach
177     INT 21h
178     MOV DL , [SI]        ; in gia tri o dau tien
179     MOV AH , 2
180     INT 21h
181     MOV AH , 9           ; in ke doc
182     LEA DX , pipe
183     INT 21h
184     MOV DL , [SI+1]      ; in gia tri o thu 2
185     MOV AH , 2
186     INT 21h
187     MOV AH , 9
188     LEA DX , pipe
189     INT 21h
190     MOV DL , [SI+2]      ; in gia tri o thu 3
191     MOV AH , 2
192     INT 21h
193     MOV AH , 9           ; in dau gach ngang
194     LEA DX , line
195     INT 21h
196     MOV DL , [SI+3]
197     MOV AH , 2
198     INT 21h
199     MOV AH , 9
200     LEA DX , pipe
201     INT 21h
202     MOV DL , [SI+4]
```

```
201     INT 21h
202     MOV DL , [SI+4]
203     MOV AH , 2
204     INT 21h
205     MOV AH , 9
206     LEA DX , pipe
207     INT 21h
208     MOV DL , [SI+5]
209     MOV AH , 2
210     INT 21h
211     MOV AH , 9
212     LEA DX , line
213     INT 21h
214     MOV DL , [SI+6]
215     MOV AH , 2
216     INT 21h
217     MOV AH , 9
218     LEA DX , pipe
219     INT 21h
220     MOV DL , [SI+7]
221     MOV AH , 2
222     INT 21h
223     MOV AH , 9
224     LEA DX , pipe
225     INT 21h
226     MOV DL , [SI+8]
227     MOV AH , 2
228     INT 21h
229     MOV AH , 9           ; in dau xuong dong
230     LEA DX , CRLF
231     INT 21h
232     ret
233 map ENDP
234
```

- Hình ảnh chương trình khi đã gọi xong hàm con **map** (ở lượt chơi đầu tiên) :




```
066
067 ; nhap nuoc di
068 input:
069     MOV AH , 9
070     LEA DX , CRLF          ; xuong dong
071     INT 21h
072     LEA DX , pos           ; in cau lenh yeu cau nhap nuoc di
073     INT 21h
074
075     MOV AH , 1             ; ham cho phep nhap 1 ky tu
076     INT 21h
077     SUB AL , 48            ; chuyen ky tu so thanh gia tri so
078     CMP AL , 1             ; neu input khong nam trong khoang [1,9]
079     JL ivl
080     CMP AL,9
081     JG ivl
082
083     MOV BL , AL
084     DEC BL                 ; chi so cua mang = input -1
085
086     MOV AH , 9             ; xuong dong
087     LEA DX , CRLF
088     INT 21h
089
090     MOV SI , OFFSET arr
091     CMP [SI+BX] , 32       ; kiem tra xem da co nuoc di trong o hay chua
092     JNE occupied           ; nhay toi label occupied neu nhu da co
```

- Nhãn **input** : cho phép người chơi nhập nước đi ở lượt hiện tại . Sau khi gọi hàm ngắt loại 1 cho phép nhật 1 ký tự , chương trình sẽ chuyển ký tự số thành giá trị số và kiểm tra xem có hợp lệ hay không .

Nếu ký tự nhập vào **không nằm trong khoảng [1,9]** , mã lệnh sẽ nhảy tới nhãn **ivl** để in câu lệnh : **"Invalid input! Please enter 1-9."** để yêu cầu người chơi nhập lại input

```
117
118 ivl:
119     MOV AH , 9             ; in thong bao nhap nuoc di khong hop le
120     LEA DX , invalid
121     INT 21h
122     jmp input              ; quay tro lai label input
123
```

Nếu ký tự nhập vào **nằm trong khoảng [1,9]** , trừ đi 1 để thành chỉ số của mảng (DEC BL) và kiểm tra xem đã có nước đi đó từ trước hay chưa bằng cách so sánh với ký tự space (32 trong bảng mã ASCII) .

```
345
346  occupied:
347     MOV AH , 9
348     LEA DX , ocp
349     INT 21h
350     JMP input
351
```

Nếu như đã có nước đi từ trước , nhảy tới nhãn occupied để yêu cầu người chơi nhập lại .

```
Player 2 turn:
 X |   |
-----------
   |   |
-----------
   |   |

Enter your move (1-9): 1

This cell is already occupied! Please choose another one.

Enter your move (1-9): 0
Invalid input! Please enter 1-9.
```

- Sau khi input hợp lệ , chương trình sẽ kiểm tra biến **turn** để xem đang là lượt của người chơi nào và đánh dấu vào mảng để cập nhật trạng thái bàn cờ .

```
093
094     MOV AL , turn
095     CMP AL , 0            ; neu nhu turn = 0 thi nhay toi label if
096     JE if
097     MOV [SI+BX] , 'O'     ; danh dau la O
098     JMP continue
099
100  if:
101     MOV [SI+BX] , 'X'     ; danh dau la X
102
```

- Sau khi cập nhật bàn cờ xong , kiểm tra điều kiện thắng / hòa để kết thúc game hoặc quay trở lại vòng lặp tiếp theo

```
102
103  continue:
104     call check            ; kiem tra xem co nguoi thang khong
105
106     MOV AL , endg         ; endg = 1 thi co nguoi thang
107     CMP AL , 1
108     JE break
109
110     call check_draw       ; kiem tra xem co hoa khong
111
112     MOV AL , endg         ; endg = 2 thi hoa nhau
113     CMP AL , 2
114     JE break
115
116     loop game_loop        ; neu khong co nguoi thang hoac hoa , lap lai
117
```

- Có người chơi chiến thắng khi 1 trong 3 hàng , 3 cột hoặc 2 đường chéo đều có cùng 1 ký tự (XXX hoặc OOO) , vậy nên chương trình sẽ lặp tối đa 3 lần để kiểm tra .

```
236
237 ; ham kiem tra xem co nguoi thang khong
238 check PROC
239     MOV AX , @data
240     MOV DS , AX
241     MOV CX , 3      ; lap toi da 3 lan
242     MOV BX , 0      ; de cong chi so
243
```

- Kiểm tra 3 hàng :

```
244  ; kiem tra 3 hang
245  three_rows:
246    MOV AL , char              ; AL = nuoc di hien tai
247    CMP [SI+BX] , 32           ; neu o dau tien trong , chuyen qua hang tiep theo luon
248    JE next_rows
249    CMP [SI+BX] , AL           ; neu 1 o khac nuoc di hien tai , chuyen qua hang tiep
250    JNE next_rows
251    CMP [SI+BX+1] , AL
252    JNE next_rows
253    CMP [SI+BX+2] , AL
254    JNE next_rows
255    call win
256  next_rows:
257    ADD BX , 3                 ; tang BX len 3 de chuyen xuong hang tiep
258    loop three_rows            ; lap
259    MOV CX , 3                 ; dat lai cho lan ktra tiep theo
260    MOV BX , 0
```

- Kiểm tra 3 cột :

```
261  three_columns:
262    MOV AL , char
263    CMP [SI+BX] , 32            ; neu o dau tien trong , chuyen qua cot tiep theo
264    JE next_columns
265    CMP [SI+BX] , AL            ; kiem tra cac o theo tung cot
266    JNE next_columns
267    CMP [SI+BX+3] , AL
268    JNE next_columns
269    CMP [SI+BX+6] , AL
270    JNE next_columns
271    call win
272  next_columns:
273    ADD BX , 1                  ; tang BX len 1 de chuyen sang cot tiep theo
274    loop three_columns
```

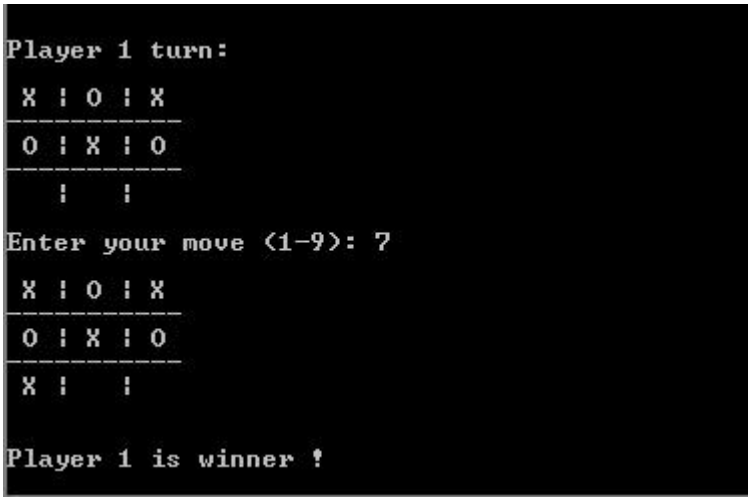
- Kiểm tra đường chéo chính và phụ :

```
275
276    CMP [SI] , 32               ; kiem tra duong cheo chinh
277    JE second_diag
278    CMP [SI] , AL
279    JNE second_diag
280    CMP [SI+4] , AL
281    JNE second_diag
282    CMP [SI+8] , AL
283    JNE second_diag
284    call win                    ; neu xuat hien duong cheo chinh , nhay toi label win
285  second_diag:
286    CMP [SI+2] , 32
287    JE exit
288    CMP [SI+2] , AL
289    JNE exit
290    CMP [SI+4] , AL
291    JNE exit
292
293    CMP [SI+6] , AL
294    JNE exit
295    call win                    ; neu xuat hien duong cheo phu , nhay qua label win
296  exit:
297    ret
298 check ENDP
299
```

- Chỉ cần kiểm tra thấy 3 ký tự cùng hàng , cùng cột hoặc cùng đường chéo liên tiếp , chương trình sẽ nhảy tới hàm con **win** để báo hiệu có người chiến thắng và kết thúc game .

```
; ham in ra trang thai bang cuoi cung va nguoi chien thang
win PROC
   MOV AX , @data
   MOV DS , AX

   MOV AH , 9            ; xuong dong
   LEA DX , CRLF
   INT 21h
   call map              ; in bang
   MOV AL , char
   CMP AL , 'O'           ; so sanh voi ky tu O , neu bang thi nhay qua label winnn
   JE winnn
   MOV AH , 9
   LEA DX , winner1      ; in ra nguoi cam quan X chien thang
   INT 21h
   MOV endg , 1          ; bien endg = 1 danh dau ket thuc game va co nguoi thang
  ret                    ; quay tro ve vi tri goi ham ( trong main)
 winnn:
   MOV AH , 9            ; in ra nguoi cam quan O chien thang
   LEA DX , winner2
   INT 21h
   MOV endg , 1          ; bien endg = 1 danh dau ket thuc game va co nguoi thang
  ret
win ENDP
```

- Hàm **win** sẽ xuống dòng và in ra trạng thái bảng Tic Tac Toe cuối cùng , sau đó kiểm tra xem ký tự vừa được đánh là ký tự nào để in ra người chiến thắng tương ứng và đánh dấu biến endg = 1 (tương đương với kết thúc game) .



Như trong hình , ký tự vừa được đánh là ký tự X nên chương trình sẽ in ra bảng trạng thái cuối cùng và dòng “Player 1 is winner ! ” .

- Nếu không có 3 ký tự nào thỏa mãn thì chương trình sẽ kết thúc hàm con **check** và quay trở lại kiểm tra hòa các lệnh nhảy tới nhãn **exit** ở trong hàm check.

```
continue:
   call check            ; kiem tra xem co nguoi thang khong

   MOV AL , endg         ; endg = 1 thi co nguoi thang
   CMP AL , 1
   JE break

   call check_draw       ; kiem tra xem co hoa khong
```

- Nếu có người chiến thắng , tức là lệnh nhảy tới hàm **win** thì biến endg sẽ được đặt bằng 1 (câu lệnh **MOV endg , 1** trong hàm **win**) và lệnh sẽ nhảy tới nhãn **break**

```
123
124  break:
125      ; ham ngat 4ch ket thuc chuong trinh
126      MOV AH , 4ch
127      INT 21h
128  MAIN ENDP
```

- Trong nhãn **break** có hàm ngắt 4CH kết thúc chương trình

Nếu không nhảy tới nhãn **break** tức là chưa có người chiến thắng , lệnh sẽ nhảy vào hàm **check_draw** để kiểm tra xem có hòa hay không

```
324
325  check_draw PROC
326      MOV CX , 9            ; bien lap CX = 9 vi phai lap 9 toi da 9 lan
327      MOV SI , OFFSET arr
328    full:
329      CMP [SI] , 32         ; neu o trong , thoat ham
330      JE not_full
331      INC SI                ; tang SI len 1 dv de chuyen qua o tiep theo
332      loop full             ; lap toi da 9 lan
333
334      MOV AH , 9            ; xuong dong
335      LEA DX , CRLF
336      INT 21h
337      call map              ; in ban co hoa
338
339      LEA DX , draw         ; in ra cau lenh thong bao hoa
340      INT 21h
341      MOV endg , 2          ; bien endg = 2 danh dau ket thuc game va hoa nhau
342    not_full:
343    ret
344  check_draw ENDP
```

- Chỉ cần có ô trống thì lệnh sẽ ngay lập tức thoát khỏi hàm check_draw và quay trở lại vòng lặp game_loop tiếp theo .

```
109
110      call check_draw       ; kiem tra xem co hoa khong
111
112      MOV AL , endg         ; endg = 2 thi hoa nhau
113      CMP AL , 2
114      JE break
115
116      loop game_loop        ; neu khong co nguoi thang hoac hoa , lap lai
117
```

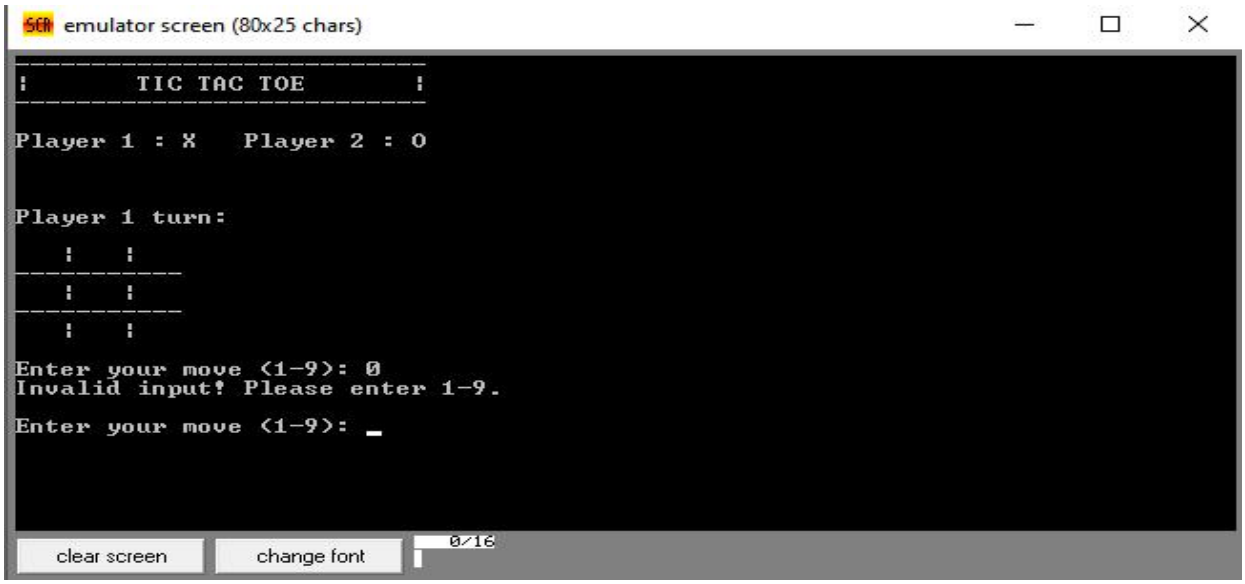
- Nếu endg = 2 (tức là hòa) thì sẽ in ra bàn cờ hòa và thông báo hòa , sau đó nhảy tới nhãn break kết thúc chương trình .

```
Player 1 turn:
 X | O | X
-----------
 O | O | X
-----------
 X |   | O

Enter your move (1-9): 8
 X | O | X
-----------
 O | O | X
-----------
 X | X | O

The game is a draw !
```

2.2.4 : Giao diện chương trình

- Giao diện bàn cờ khi bắt đầu chơi , lượt đầu tiên là của người chơi 1 đánh X

- Chương trình hiển thị dòng chữ “Enter your move <1-9> : ” để yêu cầu người chơi nhập ký tự trong khoảng [1,9] . Nếu người chơi nhập ký tự ngoài khoảng này , chương trình hiển thị như sau :



- Nếu người chơi nhập ký tự đã được đánh trước đó , chương trình sẽ thông báo :

- 2 người chơi thay phiên nhau chơi cho tới khi có người chiến thắng . Khi có người chiến thắng , chương trình sẽ in ra bàn cờ cuối cùng và cập nhật người chiến thắng .

- Nếu bàn cờ đã không còn nước đi , chương trình sẽ in ra thông báo hòa