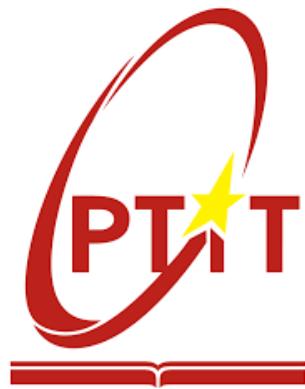


HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN



**BÁO CÁO BÀI TẬP LỚN
HỌC PHẦN KIẾN TRÚC MÁY TÍNH**

ĐỀ TÀI 1: Lập trình game TICTACTOE trên emu8086

ĐỀ TÀI 2: Lập trình máy tính BMI trên emu8086

Nhóm thực hiện:

Nhóm 04 – Lớp N02

Giảng viên hướng dẫn:

TS. Đặng Hoàng Long

Hà Nội – 2025

MỤC LỤC

MỤC LỤC.....	2
BẢNG PHÂN CHIA CÔNG VIỆC	3
TÓM TẮT QUÁ TRÌNH THỰC HIỆN.....	5
CHƯƠNG 1: LẬP TRÌNH GAME TIC-TAC-TOE TRÊN EMU8086	6
1. Giới thiệu	6
2. Nội dung chính của đề tài	6
2.1. Tổng quan về trò chơi.....	6
2.2. Giải thuật chính của chương trình	7
2.3. Miêu tả chương trình.....	8
2.3.1. Lưu đồ thuật toán	8
2.3.2. Phân tích chương trình	9
3. Kết quả thực hiện và kết luận.....	31
3.1. Kết quả thực hiện	31
3.2. Kết luận.....	36
CHƯƠNG 2: LẬP TRÌNH MÁY TÍNH BMI TRÊN EMU8086	37
1. Giới thiệu	37
2. Nội dung chính của đề tài	37
2.1. Tổng quan về Máy tính BMI.....	37
2.2. Giải thuật chính của chương trình	38
2.3. Miêu tả chương trình.....	39
2.3.1. Lưu đồ thuật toán	39
2.3.2. Phân tích chương trình	40
3. Kết quả thực hiện và kết luận.....	61
3.1. Kết quả thực hiện	61
3.2. Kết luận.....	65
CHƯƠNG 3: TỔNG KẾT VÀ ĐÁNH GIÁ.....	66
TÀI LIỆU THAM KHẢO	68

BẢNG PHÂN CHIA CÔNG VIỆC

Đề tài 1: Lập trình game Tic-Tac-Toe bằng Assembly trên EMU8086

Số thứ tự	Họ và tên	Nội dung công việc	Ghi chú
1	Trần Văn Hậu B23DCCN287	<p>Đề xuất và xây dựng ý tưởng tổng thể của project.</p> <p>Thiết kế luồng chính và cấu trúc chương trình.</p> <p>Lập trình phần main và xử lý vòng lặp trò chơi.</p> <p>Viết code nhân và kiểm tra đầu vào của người chơi.</p>	<p>Năng nổ</p> <p>Theo dõi tiến độ làm việc của cả nhóm.</p> <p>Nhanh chóng sửa chữa khi có thành viên báo lỗi sai</p> <p>Đưa ra cách xử lý giao diện khi nhập để tăng tốc độ chương trình</p>
2	Đặng Thảo Nguyên B23DCCN609	<p>Chịu trách nhiệm chính viết báo cáo của nhóm.</p> <p>Thiết kế giao diện lời chào và hướng dẫn chơi.</p> <p>Lập trình hiển thị bảng chơi dưới dạng ma trận.</p> <p>Viết phần reset trạng thái game và các biến điều khiển.</p>	<p>Năng nổ.</p> <p>Đốc thúc thành viên làm nhanh công việc trước deadline.</p> <p>Tỉ mỉ xem xét các lỗi nhỏ có thể mắc</p>
3	Hoàng Xuân Hưng B23DCCN366	<p>Lập trình hiển thị kết quả thắng/thua/hòa.</p> <p>Thêm chức năng chơi lại (restart game).</p> <p>Tìm tài liệu về macro trong emu8086.inc.</p> <p>Góp phần cải thiện giao diện người dùng.</p>	<p>Nhiệt tình hỗ trợ các bạn.</p> <p>Giúp đỡ các bạn hiểu hơn về thư viện</p>
4	Nguyễn Thanh Phong B23DCCN646	<p>Lập trình kiểm tra thắng cuộc và đổi lượt chơi.</p> <p>Thêm thông báo người chơi tiếp theo.</p> <p>Thực hiện kiểm thử toàn bộ chương trình.</p> <p>Ghi nhận, sửa lỗi và hoàn thiện chức năng game.</p>	<p>Nhiệt tình.</p> <p>Tìm ra lỗi vặt khi hiển thị sai.</p>

Đề tài 2: Lập trình máy tính BMI bằng Assembly trên EMU8086

Số thứ tự	Họ và tên	Nội dung công việc	Ghi chú
1	Dương Trí Dũng B23DCCN198	Đề xuất xây dựng ý tưởng project Thiết kế cấu trúc chương trình Viết code phần nhập số, in số và tính toán BMI.	Đề xuất ra các chức năng mới để chương trình đa năng hơn Tiếp thu ý kiến khi có đóng góp.
2	Nguyễn Minh Hiển B23DCCN289	Lập trình chức năng hiển thị bảng phân loại BMI. Viết code phần phân tích chỉ số BMI và in ra đánh giá + lời khuyên sức khỏe.	Thử nghiệm nhiều cách phân loại BMI trước khi lựa chọn phương án phù hợp.
3	Dương Đăng Khoa B23DCCN441	Lập trình phần cân nặng lý tưởng. Tim hiểu và xây dựng các macro GOTO+printf+CLEAR_SCREEN Viết báo cáo	Nhiệt tình góp ý khi tìm ra lỗi sai trong chương trình Linh hoạt, tiếp thu ý kiến để sửa lỗi chương trình
4	Dương Thé Tùng B23DCCN899	Thiết kế giao diện màn hình chính. Lập trình lời chào, khung hiển thị và menu lựa chọn. Hỗ trợ viết báo cáo	Giao diện đẹp, gọn gàng. Tinh thần trách nhiệm, hỗ trợ nhóm nhiệt tình.

TÓM TẮT QUÁ TRÌNH THỰC HIỆN

Thời gian	Công việc	Chi tiết thực hiện
Tháng 2	Khởi động và phân chia dự án	<p>Thống nhất thực hiện 2 project: Tic Tac Toe và BMI</p> <p>Chia nhóm nhỏ phụ trách từng project</p> <p>Xác định định hướng, mục tiêu và phân công công việc cụ thể</p>
Tháng 3 (tuần 1-2)	Xây dựng giao diện và cấu trúc chương trình	<p>Tic Tac Toe: Viết thủ tục WELCOME, BOARD; khởi tạo trạng thái; tìm hiểu macro GOTOXY, CLEAR_SCREEN</p> <p>BMI: Viết DRAW_BORDER, MENU, INTRO; xây dựng hoặc sử dụng macro in/goto</p>
Tháng 3 (tuần 3-4)	Xử lý chức năng chính, tối ưu chương trình	<p>Tic Tac Toe: thủ tục INPUT, kiểm tra ô, CHECK thắng/thua, VICTORY, DRAW, TRYAGAIN</p> <p>BMI: Tính BMI, phân loại, gợi ý cân nặng, kiểm tra lỗi nhập, tối ưu read/print num</p>
Tháng 4	Hoàn thiện chương trình, kiểm thử, rà soát lỗi	<p>Rà soát lỗi nhập liệu, giao diện</p> <p>Kiểm thử toàn bộ chương trình, đảm bảo hoạt động ổn định trước khi nộp</p>
Tháng 5	Viết báo cáo	Viết và hoàn thiện báo cáo

CHƯƠNG 1: LẬP TRÌNH GAME TIC-TAC-TOE TRÊN EMU8086

1. Giới thiệu

Tic Tac Toe (còn được gọi là Cờ ca-rô) là một trò chơi cỗ điện mà hầu như ai cũng biết đến. Với luật chơi đơn giản nhưng đầy thách thức, nó đã trở thành trò chơi phổ biến trong nhiều năm qua.

Trong bài tập lớn này, chúng em lựa chọn hiện thực trò chơi Tic Tac Toe bằng ngôn ngữ Assembly x86 và chạy trên phần mềm mô phỏng emu8086. Đây là một thách thức thú vị vì trò chơi vốn rất dễ để hiện thực trong các ngôn ngữ bậc cao như C/C++ hoặc Python, nhưng lại đòi hỏi tư duy chặt chẽ và thao tác chi tiết khi hiện thực bằng Assembly. Mục tiêu của đề tài là xây dựng một chương trình hoàn chỉnh có thể cho phép hai người chơi lần lượt nhập nước đi, hiển thị bảng cờ, kiểm tra điều kiện thắng hoặc hòa, và thông báo kết quả.

Toàn bộ chương trình sẽ được viết bằng Assembly và chạy ổn định trong môi trường emu8086. Việc chọn đề tài này không chỉ giúp chúng em rèn luyện khả năng lập trình Assembly mà còn hiểu sâu hơn về cách hoạt động của máy tính ở mức thấp. Thông qua quá trình thiết kế và hiện thực, chúng em học được cách xử lý nhập và xuất, vẽ giao diện đơn giản bằng ký tự, kiểm soát luồng chương trình, và tối ưu hóa mã lệnh trong giới hạn tài nguyên của bộ vi xử lý 8086.

2. Nội dung chính của đề tài

2.1. Tổng quan về trò chơi

Với Tic Tac Toe, người chơi cần phải suy nghĩ chiến lược, xác định các nước đi quan trọng, và cố gắng ngăn chặn đối thủ của mình tạo ra chuỗi thắng. Trò chơi mang lại cảm giác thú vị và có thể trở thành một bài toán trí tuệ với chiến thuật và tư duy logic.

- Cách chơi:

- Người chơi thứ nhất sẽ bắt đầu trò chơi 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.
- Người chơi cần suy nghĩ chiến lược và tìm cách ngăn chặn đối thủ trong khi cố gắng tạo ra chuỗi ba dấu hiệu liên tiếp.

- Các chức năng có trong trò chơi:

- Tạo ra bàn cờ 3x3, cho phép người chơi chọn số từ 1 đến 9 còn trống tương ứng với các ô trên bàn cờ, điền X/O vào ô đó
- Nếu có người chiến thắng (xuất hiện chuỗi 3 dấu XXX hoặc OOO liên tiếp theo hàng ngang, dọc hoặc đường chéo), sẽ in ra thông báo người chơi chiến thắng
- Nếu bàn cờ đã đầy mà không ai chiến thắng, in ra thông báo kết quả hòa

- Sau khi kết thúc trò chơi, in ra thông báo người chơi có muốn tiếp tục chơi hay không. Nếu chọn có, trò chơi sẽ được bắt đầu lại, nếu không thì chương trình sẽ kết thúc.
- Nếu trong bất kỳ bước nào, người chơi nhập sai đầu vào, sẽ in ra thông báo yêu cầu người chơi nhập lại nhằm đảm bảo chương trình không bị lỗi.
- **Đặc điểm của trò chơi:**
 - Trò chơi được thiết kế dành cho 2 người
 - Giao diện đơn giản, dễ nhìn
 - Trò chơi kinh điển, mang tính cạnh tranh cao

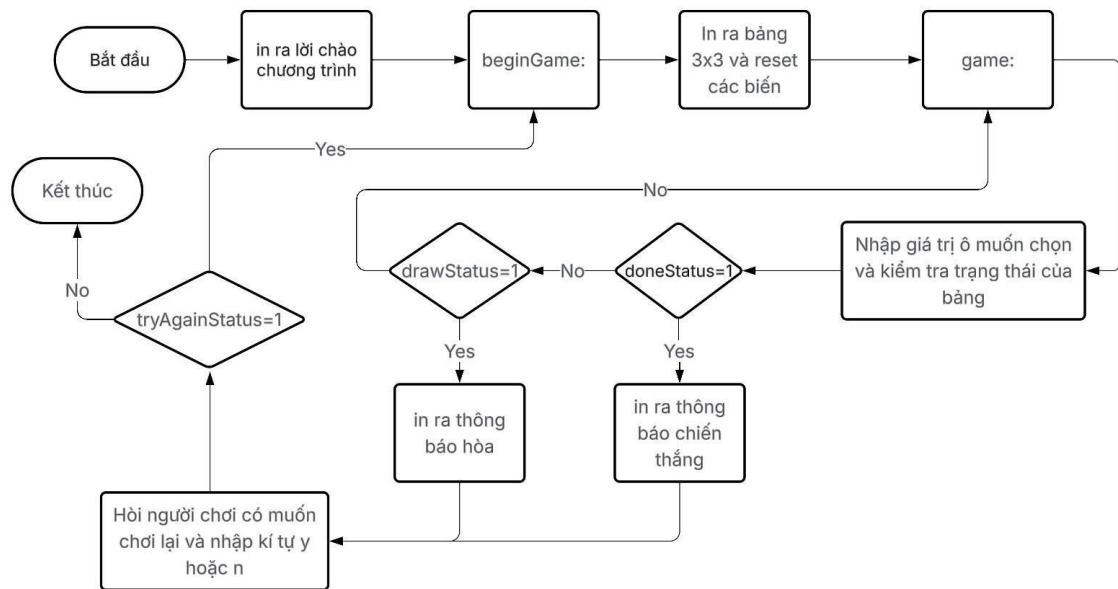
2.2. Giải thuật chính của chương trình

Thuật toán sử dụng:

- **Quản lý lượt chơi:**
 - Biến đếm **moves** được khởi tạo từ 0 nhằm theo dõi tổng số lượt đã thực hiện, tối đa là 9. Biến **player** dùng để xác định người chơi hiện tại, luân phiên giữa hai giá trị đại diện là “X” và “O”.
- **Tiếp nhận và xác thực dữ liệu nhập:**
 - Người chơi nhập vào vị trí mong muốn thông qua bàn phím. Hệ thống kiểm tra tính hợp lệ của đầu vào (bao gồm giới hạn phạm vi và trạng thái ô đã đánh dấu). Nếu dữ liệu không hợp lệ, yêu cầu người chơi nhập lại cho đến khi đạt yêu cầu.
- **Cập nhật trạng thái bảng:**
 - Sau mỗi lượt hợp lệ, hệ thống cập nhật bảng trò chơi với ký hiệu tương ứng của người chơi tại vị trí được chọn.
- **Kiểm tra điều kiện chiến thắng:**
 - Sau mỗi lượt đi, thuật toán đánh giá trạng thái bảng để xác định kết quả. Có 8 tổ hợp chiến thắng được xét: 3 hàng ngang, 3 cột dọc và 2 đường chéo. Nếu một người chơi thỏa mãn bất kỳ tổ hợp nào, hệ thống xác định người thắng cuộc.
- **Xử lý kết thúc ván chơi:**
 - Khi có người thắng hoặc khi bảng đã đầy (sau 9 lượt) mà không có người thắng, trò chơi sẽ dừng và thông báo kết quả tương ứng (thắng hoặc hòa).
- **Xử lý lựa chọn sau ván chơi:**
 - Khi một ván chơi kết thúc, hệ thống đưa ra tùy chọn cho người dùng: tiếp tục chơi ván mới hoặc kết thúc trò chơi. Dữ liệu đầu vào sẽ được kiểm tra để đảm bảo hợp lệ trước khi chuyển sang trạng thái mới tương ứng.

2.3. Miêu tả chương trình

2.3.1. Lưu đồ thuật toán



2.3.2. Phân tích chương trình

```
1     include 'emu8086.inc'
2 ;khai báo thư viện để sử dụng macro gotoxy và clear_screen
3
4 ;GOTOXY col, row - macro có 2 tham số cột và dòng, thiết lập vị trí con trỏ
5 ;CLEAR_SCREEN - xóa màn hình bằng cách cuộn màn hình và đặt con trỏ len vị trí trên cùng
6
7 .model small
8 .stack 100h
9 .data
10    ;chào mừng đến với tic tac toe
11    ;in ra dòng chữ tic tac toe
12    wc0 db 'WELCOME TO$'
13    wc1 db 2,2,2,2,2,32,2,2,2,32,32,2,2,2,32,32,2,2,32,32,2,2,32,32,2,2,32,32,2,2,2,32,32,2,2,2,'$'
14    wc2 db 32,32,2,32,32,32,32,32,2,32,32,32,32,32,32,2,32,32,32,32,32,32,32,32,32,32,32,2,32,32,2,32,2,'$'
15    wc3 db 32,32,2,32,32,32,2,32,32,32,32,32,32,32,32,2,32,32,32,32,32,32,32,32,32,32,2,32,32,2,32,2,32,2,'$'
16    wc4 db 32,32,2,32,32,32,2,32,32,32,32,32,32,2,32,32,32,2,32,32,32,32,32,32,32,32,2,32,32,2,32,32,2,'$'
17    wc5 db 32,32,2,32,32,32,2,32,32,2,2,32,32,32,32,2,32,32,2,32,32,32,32,32,32,2,32,32,2,32,32,2,32,32,2,'$'
18    ;in ra thông tin nhóm cùng giang viên
19    wc6 db 'Nhóm 4$'
20    wc7 db 'Giang viên: Đặng Hoàng Long$'
21    wc8 db 'Bấm nút bất kỳ để tiếp tục...$'
22
23    ;trang thái của các ô trên bảng
24    cell db '0123456789','$'
25
26    ;kiểu tu người chơi
27    inforPlayer1 db 'Người chơi 1 : (X) $'
28    inforPlayer2 db 'Người chơi 2 : (O) $'
```

include 'emu8086.inc' : khai báo thư viện để sử dụng macro như GOTOXY (di chuyển con trỏ) và CLEAR_SCREEN (xóa màn hình).

.model small: khai báo chương trình với kích cỡ nhỏ (dưới 64KB).

.stack 100h: khai báo đoạn ngắn xếp với 256 byte.

.data: Đánh dấu bắt đầu vùng dữ liệu của chương trình.

- **wc0 db 'WELCOME TO\$'** : Đây là cách khai báo một chuỗi trong hộp ngõ. Dòng chữ "WELCOME TO" sẽ được lưu trữ dưới label (nhãn/biến) wc0
 - db (define byte): dùng để khai báo một chuỗi hoặc giá trị byte.
 - 'WELCOME TO\$': Chuỗi sẽ hiển thị là "WELCOME TO", kết thúc bằng ký tự \$ để dùng trong thủ tục hiển thị (như int 21h).
- **wc1 db 2,2,2,2,2,32,2,32,...**: Đây là dòng dữ liệu được mã hóa để tạo hiệu ứng hình ảnh cho dòng chữ "TIC TAC TOE" theo kiểu nghệ thuật (tạo hình bằng ký tự).
 - 32 là mã ASCII của khoảng trắng.
 - 2 là mã ASCII của ký tự đặc biệt hoặc dấu "block" (█, █, █.. tùy font), dùng để tạo hình.
- Các biến **wc2 - wc8** có chức năng tương tự như **wc1**, kết hợp lại sẽ hiển thị đầy đủ dòng chữ "TIC TAC TOE" bằng ký tự đồ họa trên màn hình

- Các biến **wc6** đến **wc8**: chứa các chuỗi văn bản khác như thông tin nhóm, giảng viên, và hướng dẫn tiếp tục chương trình.
- **cell db '0123456789','\$'** : Đây là mảng chứa các ký tự tương ứng với các ô trên bàn cờ, dùng để lưu trạng thái ô đã đánh hay chưa.
- **inforPlayer1 db 'Nguoi choi 1 : (X) \$'**
- **inforPlayer2 db 'Nguoi choi 2 : (O) \$'**
 - Hai biến này dùng để hiển thị thông tin cho biết người chơi 1 dùng ký hiệu X, người chơi 2 dùng ký hiệu O.

```

30      ;tao bang
31
32      line1 db  '-----|$'
33      line2 db  '| | | | |$'
34      line3 db  '| 1 | 2 | 3 |$'
35      line4 db  '| | | | |$'
36      line5 db  '|-----|$'
37      line6 db  '| | | | |$'
38      line7 db  '| 4 | 5 | 6 |$'
39      line8 db  '| | | | |$'
40      line9 db  '|-----|$'
41      line10 db '| | | | |$'
42      line11 db '| 7 | 8 | 9 |$'
43      line12 db '| | | | |$'
44      line13 db '|-----|$'

```

- Các biến **line1**- **line13**: dùng để khai báo từng dòng (dạng chuỗi) hiển thị lên màn hình, nhằm tạo ra bàn cờ Tic Tac Toe

```

46      player db '1$'          ;ki tu nguoi choi thu 1,2
47      currentMark db 'X$'    ;ki tu o danh dau cua nguoi choi
48      moves db 0            ; so lan di chuyen
49      doneStatus db 0       ;da hoan thanh tran dau
50      drawStatus db 0       ;trang thai hoa
51      tryAgainStatus db 0   ;trang thai ban co muon choi lai khong
52
53      ;hien ra ket qua tran dau
54      win1 db 'Nguoi choi $'
55      win2 db ' chien thang $'
56      drw db 'Tran dau hoa!           $'
57
58      ;nhap o muon chon
59      inp db 'nhap o muon chon. $'
60      wasTaken db 'O da duoc chon. Moi ban chon lai.     $'
61
62      ;nhap khi choi lai
63      try db 'ban co muon choi lai khong?(y/n): $'
64      wronginput db 'khong dung ki tu, moi ban nhap lai. $'

```

- **player**: biến này lưu ký tự '1' ban đầu, dùng để xác định lượt chơi (người chơi 1 hay 2).
- **currentMark**: Dùng để lưu ký hiệu của người chơi (nếu người chơi là 1 thì đánh dấu là X, 2 thì là O)
- **moves**: biến đếm tổng số lượt đi, bắt đầu từ 0
- **doneStatus**: biến đánh dấu trạng thái trận đấu đã kết thúc hay chưa (1 là trận đấu đã kết thúc, 0 là chưa)
- **drawStatus**: biến kiểm tra trạng thái trận đấu đã hòa không (1 là trận đấu hòa)
- **tryAgainStatus**: biến lưu lựa chọn của người chơi khi được hỏi “có muốn chơi lại không”.
- Các biến **win1**, **win2**, **drw**: chứa các chuỗi văn bản hiển thị kết quả trận đấu.
 - win1: "Người chơi [số] chiến thắng"
 - win2: Phần tiếp theo ghép nối với win1
 - drw: "Trận đấu hòa!"
- **inp**, **wasTaken**, **try**, **wronginput**: Các biến trên đều là các chuỗi thông báo, được dùng để hiển thị cho người chơi trong quá trình tương tác với chương trình. Chúng giúp người chơi biết mình cần làm gì hoặc nhận thông báo khi có lỗi nhập liệu.
 - inp: Nhắc người chơi “nhập ô muốn chọn.”
 - wasTaken: Thông báo “Ô đã được chọn. Mời bạn chọn lại.”
 - try: Hỏi người chơi “Bạn có muốn chơi lại không?(y/n)”
 - wronginput: Thông báo khi nhập sai ký tự (không phải y/n) “không đúng ký tự, mời bạn nhập lại.”

```

66    .code
67    main proc
68        mov ax, @data
69        mov ds, ax
70
71        call WELCOME           ;goi phan loi chao
72        beginGame:
73        call BOARD             ;tao bang
74        call INIT              ;khoi tao lai tat ca cac gia tri
75        game:
76        call INPUT              ;nhap o muon chon trong luot choi
77        call CHECK              ;kiem tra chien thang hoac hoa
78
79        cmp doneStatus,1       ;kiem tra trang thai chien thang
80        je callVictory         ;nhay den cho goi ham in ra chien thang
81
82        cmp drawStatus,1       ;kiem tra trang thai hoa
83        je callDraw             ;nhay den cho goi ham in ra hoa
84
85        jmp game

```

```

87      callVictory:           ;
88      call VICTORY          ;goi ham in ra chien thang
89      call TRYAGAIN         ;goi ham hoi xem nguoi choi co muon choi lai
90      cmp tryAgainStatus,1  ;kiem tra nguoi choi co muon choi lai
91      je callTryAgain      ;nhay den goi ham choi lai
92
93      jmp EXIT             ;nguoi choi khong muon choi lai nen thoat chuong trinh
94
95      callDraw:             ;
96      call DRAW              ;goi ham in ra kq hoa
97      call TRYAGAIN         ;kiem tra nguoi choi co muon choi lai
98      cmp tryAgainStatus,1  ;nhay den goi ham choi lai
99      je callTryAgain
100
101     jmp EXIT             ;nguoi choi khong muon choi lai nen thoat chuong trinh
102
103     callTryAgain:          ;choi lai
104     jmp beginGame:        ;tro lai luc ban dau
105
106     EXIT:                 ;ket thuc chuong trinh
107     mov ah,4ch
108     int 21h
109
110     main endp

```

- **.code:** khai báo bắt đầu vùng chứa mã lệnh của chương trình.
- **main proc, main endp:** khai báo thủ tục bắt đầu và kết thúc chương trình chính.
- **mov ax, @data ; mov ds, ax :** đưa địa chỉ phần dữ liệu vào thanh ghi DS, cho phép truy xuất dữ liệu được khai báo trong .data.
- **call WELCOME:** gọi Thủ tục WELCOME để hiển thị lời chào và thông tin nhóm.
- **beginGame:** nhãn đánh dấu điểm bắt đầu chương trình xử lý trò chơi.
- **call BOARD, call INIT:** gọi Thủ tục để tạo bảng và khởi tạo các biến để bắt đầu ván mới.
- **game:** nhãn đánh dấu chương trình nơi ván chơi diễn ra.
- **call INPUT, call CHECK:** gọi Thủ tục nhập và kiểm tra trạng thái trận đấu
- **cmp doneStatus,1 ; je callVictory:** So sánh biến doneStatus với 1
 - Nếu doneStatus = 1 tức là ván chơi kết thúc, chương trình nhảy đến nhãn callVictory để xử lý kết quả trận đấu.
- **cmp drawStatus,1 ; je callDraw:** So sánh biến drawStatus với 1
 - Nếu drawStatus = 1 tức là ván chơi hòa, chương trình nhảy đến nhãn callDraw để xử lý kết quả trận đấu.
- **jmp Game:** nếu không thỏa mãn điều kiện thắng/hòa thì nhảy đến nhãn Game để tiếp tục trận đấu

- **call VICTORY, call TRYAGAIN**: gọi thủ tục để đưa ra kết quả và xử lý sau trận đấu
- **cmp tryAgainStatus, 1 ; je callTryAgain** : nếu tryAgainStatus = 1 (tức người chơi chọn chơi lại) thì nhảy đến nhãn callTryAgain
- **jmp EXIT**: nếu không chơi lại thì nhảy đến nhãn EXIT
- **callDraw**: Nhãn xử lý kết quả khi trận đấu hòa
- **EXIT** : nhãn với chức năng kết thúc chương trình

```

119      WELCOME proc
120          call CLEAR_SCREEN    137      GOTOXY 15,9
121          mov ah,9             138      lea dx, wc5
122          GOTOXY 32,3         139      int 21h
123          lea dx,wc0           140      GOTOXY 35,12
124          int 21h             141      lea dx,wc6
125          GOTOXY 15,5         142      int 21h
126          lea dx, wc1           143      GOTOXY 28,13
127          int 21h             144      lea dx,wc7
128          GOTOXY 15,6         145      int 21h
129          lea dx, wc2           146      GOTOXY 23,14
130          int 21h             147      lea dx,wc8
131          GOTOXY 15,7         148      int 21h
132          lea dx, wc3           149
133          int 21h             150      mov ah,7
134          GOTOXY 15,8         151      int 21h
135          lea dx, wc4           152      ret
136          int 21h             153      WELCOME endp

```

Đây là thủ tục **WELCOME** dùng để hiển thị lời chào khi bắt đầu chương trình

- **mov ah, 9; int 21h** : in ra chuỗi có địa chỉ tại thanh ghi DX. Đây là Thủ tục dịch vụ 09h của ngắt 21h, in chuỗi kết thúc bằng ký tự \$
- **GOTOXY a,b** : Macro dùng để đưa con trỏ đến vị trí cột thứ a, dòng thứ b trên màn hình.
- **lea dx, wc[i]** (với i chạy từ 0 đến 8) : đưa địa chỉ của chuỗi được lưu trong biến wc[i] vào thanh ghi DX để chuẩn bị in ra màn hình
- **mov ah, 7; int 21h** : giúp nhập vào 1 ký tự từ bàn phím nhưng không hiển thị ra màn hình
- **ret**: kết thúc các thủ tục tại thủ tục WELCOM.

```
169      BOARD proc
170          call CLEAR_SCREEN
171
172          GOTOXY 2,3
173          mov ah,9
174          lea dx,inforPlayer1
175          int 21h
176
177          GOTOXY 2,4
178          mov ah,9
179          lea dx,inforPlayer2
180          int 21h
181
182          GOTOXY 30,2      ;dua con tro den cot 30 dong 2
183          mov ah,9
184          lea dx,line1
185          int 21h
186
187          GOTOXY 30,3      ;dua con tro den cot 30 dong 3
188          mov ah,9
189          lea dx,line2
190          int 21h
191
192          GOTOXY 30,4      ;dua con tro den cot 30 dong 4
193          mov ah,9
194          lea dx,line3
195          int 21h
```

```
197      GOTOXY 30,5      ;dua con tro den cot 30 dong 5
198      mov ah,9
199      lea dx,line4
200      int 21h
201
202      GOTOXY 30,6      ;dua con tro den cot 30 dong 6
203      mov ah,9
204      lea dx,line5
205      int 21h
206
207      GOTOXY 30,7      ;dua con tro den cot 30 dong 7
208      mov ah,9
209      lea dx,line6
210      int 21h
211
212      GOTOXY 30,8      ;dua con tro den cot 30 dong 8
213      mov ah,9
214      lea dx,line7
215      int 21h
216
217      GOTOXY 30,9      ;dua con tro den cot 30 dong 9
218      mov ah,9
219      lea dx,line8
220      int 21h
221
222      GOTOXY 30,10     ;dua con tro den cot 30 dong 10
223      mov ah,9
224      lea dx,line9
225      int 21h
```

```

227      GOTOXY 30,11      ;dua con tro den cot 30 dong 11
228      mov ah,9
229      lea dx,line10
230      int 21h
231
232      GOTOXY 30,12      ;dua con tro den cot 30 dong 12
233      mov ah,9
234      lea dx,line11
235      int 21h
236
237      GOTOXY 30,13      ;dua con tro den cot 30 dong 13
238      mov ah,9
239      lea dx,line12
240      int 21h
241
242      GOTOXY 30,14      ;dua con tro den cot 30 dong 14
243      mov ah,9
244      lea dx,line13
245      int 21h
246
247      ret
248      BOARD endp

```

Đây là thủ tục **BOARD**, dùng để tạo bảng 3x3 và hiển thị thông tin người chơi

Các lệnh sử dụng trong thủ tục này tương tự như trong thủ tục WELCOME

- Gọi **CLEAR_SCREEN** để xóa màn hình.
- Sử dụng **GOTOXY** để định vị con trỏ đến từng vị trí cụ thể.
- Dùng **mov ah, 9** và **int 21h** để in các chuỗi: thông tin người chơi và các dòng kẻ tạo bảng.

```

250      ;khoi tao tat ca cac bien ve ban dau
251      INIT proc
252          mov cell[1],'1'
253          mov cell[2],'2'
254          mov cell[3],'3'
255          mov cell[4],'4'
256          mov cell[5],'5'
257          mov cell[6],'6'
258          mov cell[7],'7'
259          mov cell[8],'8'
260          mov cell[9],'9'
261          ;nguo Choi 1 la nguoi Choi di truoc
262          mov player,'1'
263          mov currentMark,'X'
264          mov moves,0
265          mov doneStatus,0
266          mov drawStatus,0
267          mov tryAgainStatus,0
268      ret
269      INIT endp

```

Thủ tục **INIT**, được dùng để khởi tạo lại các biến về giá trị ban đầu, chuẩn bị cho việc bắt đầu một ván chơi mới.

- Các biến được khởi tạo lại bao gồm các ô cờ, lượt chơi, ký hiệu đánh, số lượt đi và các trạng thái kiểm tra kết thúc ván.
- Cấu trúc lệnh tương tự như các thủ tục đã trình bày ở trên.

```

271      ; ham thay doi luot cua nguoi choi
272      PLAYERCHANGE proc
273          cmp player,'1'      ;kiem tra nguoi choi la 1 de doi sang 2
274          je changeToPlayer2
275          jmp changeToPlayer1
276
277          changeToPlayer1:    ;doi sang nguoi choi 1
278              mov player,'1'
279              mov currentMark,'X'
280              ret
281
282          changeToPlayer2:    ;doi sang nguoi choi 2
283              mov player,'2'
284              mov currentMark,'0'
285          ret
286      PLAYERCHANGE endp

```

Thủ tục **PLAYERCHANGE** dùng để thay đổi trạng thái người chơi

- **cmp player, '1'**: So sánh giá trị biến player với ký tự '1' để kiểm tra xem hiện tại có phải là lượt của người chơi 1 không.
- **jmp changeToPlayer1**: Nếu không đúng, nhảy đến nhãn **changeToPlayer1** để chuyển lượt về người chơi 1.

Nhãn **changeToPlayer1**:

- **mov player, '1'**: Gán giá trị '1' cho biến player, xác định lượt chơi hiện tại là người chơi 1.
- **mov currentMark, 'X'**: Gán ký tự 'X' cho biến currentMark, đánh dấu lượt đi của người chơi 1 bằng ký hiệu X.
- **ret**: Kết thúc Thủ tục, trả về chương trình gọi.

Nhãn **changeToPlayer2** có chức năng tương tự

```
289     CHECK proc
290         check1:           ;kiem tra hang 1 2 3
291             mov al,cell[1]
292             cmp al,cell[2]
293             jne check2
294             cmp al,cell[3]
295             jne check2
296
297             mov doneStatus,1
298             ret
299
300         check2:           ;kiem tra hang 4 5 6
301             mov al,cell[4]
302             cmp al,cell[5]
303             jne check3
304             cmp al,cell[6]
305             jne check3
306
307             mov doneStatus,1
308             ret
309
310         check3:           ;kiem tra hang 7 8 9
311             mov al,cell[7]
312             cmp al,cell[8]
313             jne check4
314             cmp al,cell[9]
315             jne check4
316
317             mov doneStatus,1
318             ret
```

```
320     check4:          ;kiem tra cot 1 4 7
321     mov al,cell[1]
322     cmp al,cell[4]
323     jne check5
324     cmp al,cell[7]
325     jne check5
326
327     mov doneStatus,1
328     ret
329
330     check5:          ;kiem tra cot 2 5 8
331     mov al,cell[2]
332     cmp al,cell[5]
333     jne check6
334     cmp al,cell[8]
335     jne check6
336
337     mov doneStatus,1
338     ret
339
340     check6:          ;kiem tra cot 3 6 9
341     mov al,cell[3]
342     cmp al,cell[6]
343     jne check7
344     cmp al,cell[9]
345     jne check7
346
347     mov doneStatus,1
348     ret
```

```

350      check7:           ;kiem tra duong cheo chinh 1 5 9
351      mov al,cell[1]
352      cmp al,cell[5]
353      jne check8
354      cmp al,cell[9]
355      jne check8
356
357      mov doneStatus,1
358      ret
359
360      check8:           ;kiem tra duong cheo phu 3 5 7
361      mov al,cell[3]
362      cmp al,cell[5]
363      jne checkdraw
364      cmp al,cell[7]
365      jne checkdraw
366
367      mov doneStatus,1
368      ret
369      checkdraw:         ;kiem tra ket qua hoa
370      mov al,moves
371      cmp al,9           ;so sanh so luong buoc di voi 9
372      jl callPlayerChange ;so luong buoc di nho hon 9 nen doi nguoi choi
373
374      mov drawStatus,1    ;trang thai hoa la true
375      ret
376
377
378      callPlayerChange:
379      call PLAYERCHANGE   ;goi ham doi luot choi
380      ret
381      ret
382      CHECK endp

```

Thủ tục **CHECK** dùng để kiểm tra kết quả trận đấu sau mỗi nước đi

Nhân **check1**: kiểm tra hàng ngang đầu tiên (ô 1,2,3)

- **mov al, cell[1]**: gán giá trị trong ô 1 vào thanh ghi AL
 - AL là thanh ghi 8-bit, nên được sử dụng để thao tác với các giá trị 8-bit như các phần tử cell[i]. Do mỗi ô cờ chỉ chứa một ký tự (ví dụ: '1', 'X', 'O'), tức là một byte (8 bit), việc sử dụng AL là phù hợp.
 - Nếu dùng thanh ghi có kích thước lớn hơn như AX (16-bit), có thể dẫn đến sai lệch dữ liệu hoặc lỗi chương trình do không khớp kích thước.
- **cmp al, cell[2]**: so sánh giá trị trong ô 2 với giá trị trong thanh ghi AL (Tức so sánh ô 1 với ô 2)
 - nếu không bằng thì nhảy sang nhãn check2 (**jne check2**)
- **cmp al, cell[3]**: Nếu ô 2 bằng ô 1, tiếp tục so sánh với ô số 3.
- **jne check2**: Nếu ô 3 không bằng ô 1 thì nhảy sang nhãn check 2.
- **mov doneStatus,1** : Khi cả 3 ô đều có giá trị bằng nhau, cập nhật biến doneStatus thành 1 (đánh dấu rằng trận đấu đã có người chiến thắng)

Với các nhãn **check2** đến **check8**: chức năng tương tự check1

Nhãn **checkdraw**: Kiểm tra trạng thái hòa khi chưa có chiến thắng từ 8 nhãn check

- **mov al, moves**: gán số bước đi (moves) vào thanh ghi AL
- **cmp al, 9** : so sánh số bước đi với 9 (số bước đi tối đa)
 - **jl callPlayerChange**: Nếu số bước đi nhỏ hơn 9 thì chuyển đến nhãn callPlayerChange (đổi lượt chơi để tiếp tục ván đấu)
 - Nếu lệnh trên không được thực hiện thì số bước đi đã đạt tối đa, chuyển trạng thái trận đấu về hòa (**mov drawStatus,1**) và kết thúc thủ tục.

```

384      ;ham xuat ket qua chien thang
385      VICTORY proc
386          GOTOXY 26,19      ;dua con tro den cot 26 dong 19
387          mov ah,9
388          lea dx,win1      ;in ra "nguo Choi"
389          int 21h
390
391          lea dx,player    ;in ra nguoi Choi chien thang
392          int 21h
393
394          lea dx,win2      ;in ra "chien thang"
395          int 21h
396
397          GOTOXY 22,20      ;dua con tro den cot 22 dong 20
398          mov ah,9
399          lea dx,wc8
400          int 21h
401
402          mov ah,7      ;bam nut bat ki
403          int 21h
404          ret
405      VICTORY endp

407      ; ham xuat ket qua hoa
408      DRAW proc
409          GOTOXY 26,19      ;dua con tro den cot 26 dong 19
410          mov ah,9
411          lea dx,drw      ;in ra ket qua hoa
412          int 21h
413
414          GOTOXY 22,20      ;dua con tro den cot 22 dong 20
415          mov ah,9
416          lea dx,wc8      ;in ra dong chu bam nut bat ki de tiep tuc
417          int 21h
418
419          mov ah,7      ;bam nut bat ki
420          int 21h
421
422          ret
423      DRAW endp

```

Thủ tục **VICTORY** và **DRAW** sử dụng các biến thông điệp đã được khai báo trước đó để hiển thị kết quả trận đấu tương ứng.

```

426      ; kiem tra nguoi choi co muon choi lai khong
427      TRYAGAIN proc
428          call CLEAR_SCREEN
429          ag:                 ; dung de khi nguoi choi nhap sai thi se quay lai
430          GOTOXY 24,10       ;in ra ban co muon choi lai
431          mov ah,9
432          lea dx,try
433          int 21h
434
435          mov ah,1
436          int 21h
437
438          cmp al,'y'        ;so sanh y la yes
439          je callYES         ;nhay den callYes
440          cmp al,'Y'
441          je callYES
442
443          cmp al,'n'        ;so sanh n la no
444          je callNO          ;nhay den callno
445          cmp al,'N'
446          je callNO
447
448          ;nhap sai ki tu y hoac n
449          GOTOXY 24,9
450          mov ah,9
451          lea dx,wronginput   ;in ra viec nguoi choi nhap sai
452          int 21h
453          jmp ag             ;nhay den again

455          callYES:
456          mov tryAgainStatus,1; thay doi trang thai choi lai la true
457          ret
458
459          callNO:              ; nguoi choi khong muon choi lai
460          ret
461          ret
462      TRYAGAIN endp

```

Thủ tục TRYAGAIN dùng để in ra thông điệp:

“Bạn có muốn chơi lại hay không?(y/n)”

- mov ah, 1; int 21h: Nhập ký tự từ bàn phím có hiển thị ra màn hình

Nếu người chơi nhập y/Y thì nhảy đến nhãn **callYES** để bắt đầu trận đấu mới

- **mov tryAgainStatus, 1**: thay đổi trạng thái của **tryAgainStatus** về 1

Nếu người chơi nhập n/N thì nhảy đến nhãn **callNO** để thoát khỏi chương trình

Nếu nhập sai ký tự, chương trình nhảy đến nhãn **WRONGINPUT** để hiển thị thông điệp “Không đúng ký tự, mời bạn nhập lại” và nhảy về nhãn **ag** để nhập lại.

```
464      ;ham nhap o nguoi choi muon dien vao
465      INPUT proc
466          startInput:      ;
467          GOTOXY 20,16      ;di chuyen con tro den vi tri de in ra thong bao moi nguoi choi lua chon o
468
469          cmp player,'1'    ;so sanh nguoi choi thu 1
470          je take1        ;nhay den phan xu ly nguoi choi thu 1 neu dung
471
472          ;xu ly nguoi choi thu 2
473          mov ah,9
474          lea dx,inforPlayer2 ; in ra nguoi choi thu 2
475          int 21h
476          jmp takeinput
477
478          take1:
479          mov ah,9
480          lea dx,inforPlayer1 ;in ra nguoi choi thu 1
481          int 21h
482
483          takeinput:
484          mov ah,9
485          lea dx,inp          ;in ra nguoi choi chon o muon chon
486          int 21h
487
488          mov ah,1            ;nhap 1 ki tu
489          int 21h
490
491          inc moves         ;tang so lan choi len 1
492          mov bl,al
493          sub bl,'0'
494
495          mov cl,currentMark ;chuyen ki tu nguoi choi thu 1 hoac 2 vao thanh ghi cl
```

```

497      cmp bl,1
498      je checkCell1
499      cmp bl,2
500      je checkCell2
501      cmp bl,3
502      je checkCell3
503      cmp bl,4
504      je checkCell4
505      cmp bl,5
506      je checkCell5
507      cmp bl,6
508      je checkCell6
509      cmp bl,7
510      je checkCell7
511      cmp bl,8
512      je checkCell8
513      cmp bl,9
514      je checkCell9
515
516      ;nhap sai ki tu
517      dec moves           ;giam so lan choi xuong 1
518      GOTOXY 20,16
519      mov ah,9
520      lea dx,wronginput   ;in ra nhap sai ki tu
521      int 21h
522
523      mov ah,7
524      int 21h
525      jmp startInput     ;nhap lai

```

Thủ tục **INPUT** dùng để xử lý việc nhập nước đi từ người chơi.

Tại nhãn **startInput**, chương trình kiểm tra biến **player** để xác định lượt chơi.

- Nếu là người chơi 1 (**player = '1'**), in thông điệp "Người chơi 1: (X)"
- Ngược lại, in thông điệp "Người chơi 2: (O)"

Nhãn **takeinput** chịu trách nhiệm:

- In thông điệp "Nhập ô muốn chọn:"
- Nhận ký tự nhập từ bàn phím bằng int 21h với ah = 1
- Tăng biến đếm số bước đi thêm 1 (**inc moves**)
- Chuyển đổi ký tự vừa nhập sang số (ví dụ: ký tự '3' chuyển thành số 3 bằng **mov al, bl và sub bl, '0'**).
- Đưa ký hiệu người chơi (X hoặc O) vào thanh ghi CL để chuẩn bị ghi vào ô đã chọn

Sau đó, chương trình so sánh giá trị người chơi vừa nhập (BL) với các ô cờ (1 đến 9) và nhảy đến nhãn tương ứng (checkCell1, checkCell2, v.v.) để xử lý.

Nếu nhập sai (không phải số từ 1 đến 9 hoặc ô đã chọn bị chiếm), chương trình:

- Giảm biến moves đi 1 (**dec moves**)
- In thông điệp lỗi: "Không đúng ký tự, mời bạn nhập lại."
- Nhảy về **startInput** để nhập lại

```
527      ;kiem tra o duoc chon
528      checkcell1:
529      cmp cell[1], '0'
530      je taken
531      cmp cell[1], 'X'
532      je taken
533      mov cell[1], cl
534      GOTOXY 32,4          ;chuyen con tro vao vi tri o thu 1
535      mov ah,9
536      lea dx,currentMark ;ghi de ki tu nguoi choi vao o 1
537      int 21h
538      ret
539
540      checkcell2:
541      cmp cell[2], '0'
542      je taken
543      cmp cell[2], 'X'
544      je taken
545      mov cell[2], cl
546      GOTOXY 36,4          ;chuyen con tro vao vi tri o thu 2
547      mov ah,9
548      lea dx,currentMark ;ghi de ki tu nguoi choi vao so 2
549      int 21h
550      ret
```

```

552         checkcell3:
553             cmp cell[3], '0'
554             je taken
555             cmp cell[3], 'X'
556             je taken
557             mov cell[3], cl
558             GOTOXY 40,4           ;chuyen con tro vao vi tri o thu 3
559             mov ah,9
560             lea dx, currentMark ;ghi de ki tu nguoi choi vao o 3
561             int 21h
562             ret
563
564         checkcell4:
565             cmp cell[4], '0'
566             je taken
567             cmp cell[4], 'X'
568             je taken
569             mov cell[4], cl
570             GOTOXY 32,8           ;chuyen con tro vao vi tri o thu 4
571             mov ah,9
572             lea dx, currentMark ;ghi de ki tu nguoi choi vao o 4
573             int 21h
574             ret
575         checkcell5:
576             cmp cell[5], '0'
577             je taken
578             cmp cell[5], 'X'
579             je taken
580             mov cell[5], cl
581             GOTOXY 36,8           ;chuyen ki tu con tro vao o thu 5
582             mov ah,9
583             lea dx, currentMark ;chuyen ki tu nguoi choi vao o 5
584             int 21h
585             ret
586
587         checkcell6:
588             cmp cell[6], '0'
589             je taken
590             cmp cell[6], 'X'
591             je taken
592             mov cell[6], cl
593             GOTOXY 40,8           ;chuyen ki tu con tro vao o thu 6
594             mov ah,9
595             lea dx, currentMark ;ghi de ki tu nguoi choi vao o 6
596             int 21h
597             ret
598

```

```
600     checkcell7:  
601         cmp cell[7],'0'  
602         je taken  
603         cmp cell[7],'X'  
604         je taken  
605         mov cell[7],cl  
606         GOTOXY 32,12      ;chuyen ki tu con tro vao o thu 7  
607         mov ah,9  
608         lea dx,currentMark ;ghi de ki tu nguoi choi vao o 7  
609         int 21h  
610         ret  
611  
612     checkcell8:  
613         cmp cell[8],'0'  
614         je taken  
615         cmp cell[8],'X'  
616         je taken  
617         mov cell[8],cl  
618         GOTOXY 36,12      ;chuyen ki tu con tro vao o thu 8  
619         mov ah,9  
620         lea dx,currentMark ;ghi de ki tu nguoi choi vao o 8  
621         int 21h  
622         ret  
624     checkcell9:  
625         cmp cell[9],'0'  
626         je taken  
627         cmp cell[9],'X'  
628         je taken  
629         mov cell[9],cl  
630         GOTOXY 40,12      ;chuyen ki tu con tro vao o thu 9  
631         mov ah,9  
632         lea dx,currentMark ;ghi de ki tu nguoi choi vao o 9  
633         int 21h  
634         ret
```

Các nhãn **checkcell1**, **checkcell2**,...: Kiểm tra tình trạng ô cờ tương ứng

- Nếu ô đã chứa 'X' hoặc 'O' (tức là đã có người chọn), thì nhảy đến nhãn taken để xử lý tình huống chọn trùng.

- Nếu ô còn trống, thì ghi ký hiệu của người chơi hiện tại vào ô và hiển thị nó lên màn hình.

Với các nhãn **checkcell[i]** còn lại có chức năng tương tự.

```
636      ;o da duoc chon
637      taken:
638      dec moves
639      GOTOXY 20,16
640      mov ah,9
641      lea dx,wasTaken      ;in ra o da duoc chon
642      int 21h
643
644      mov ah,7
645      int 21h
646      jmp startInput      ;nhap lai
647      ret
648      INPUT endp
649
650      DEFINE_CLEAR_SCREEN
651
652      end main
```

Nhãn **taken**: xử lý trường hợp ô đã được lựa chọn

- **dec moves**: Giảm biến moves đi 1 đơn vị vì nước đi hiện tại không hợp lệ
- Đưa ra thông điệp “Ô đã được chọn. Mời bạn chọn lại.” bằng lệnh mov ah, 9 int 21h
- Dùng lệnh mov ah, 7 int 21h để chờ người chơi bấm phím bất kỳ (không hiển thị lên màn hình).
- **jmp startInput**: nhảy đến nhãn startInput để nhập lại

DEFINE_CLEAR_SCREEN: định nghĩa thủ tục để xóa màn hình.

3. Kết quả thực hiện và kết luận

3.1. Kết quả thực hiện

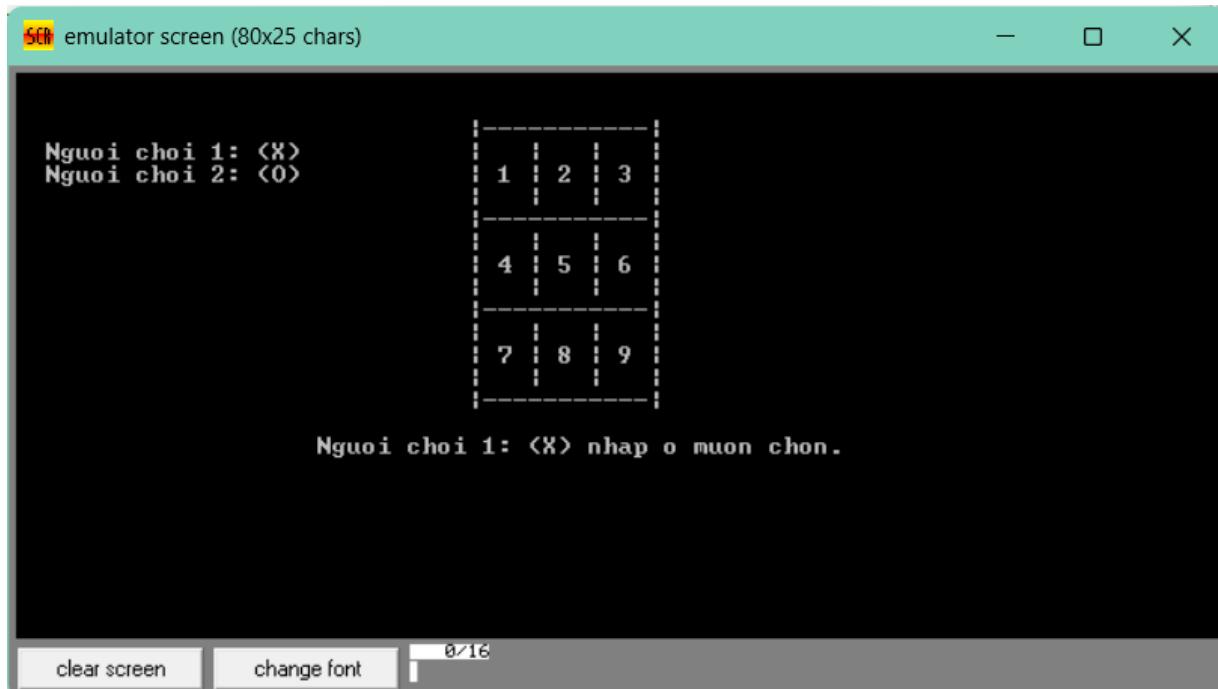
Khi chương trình chạy, giao diện sẽ hiển thị lời chào, tên chương trình và thông tin của nhóm

Chương trình yêu cầu nhập một ký tự bất kỳ để tiếp tục

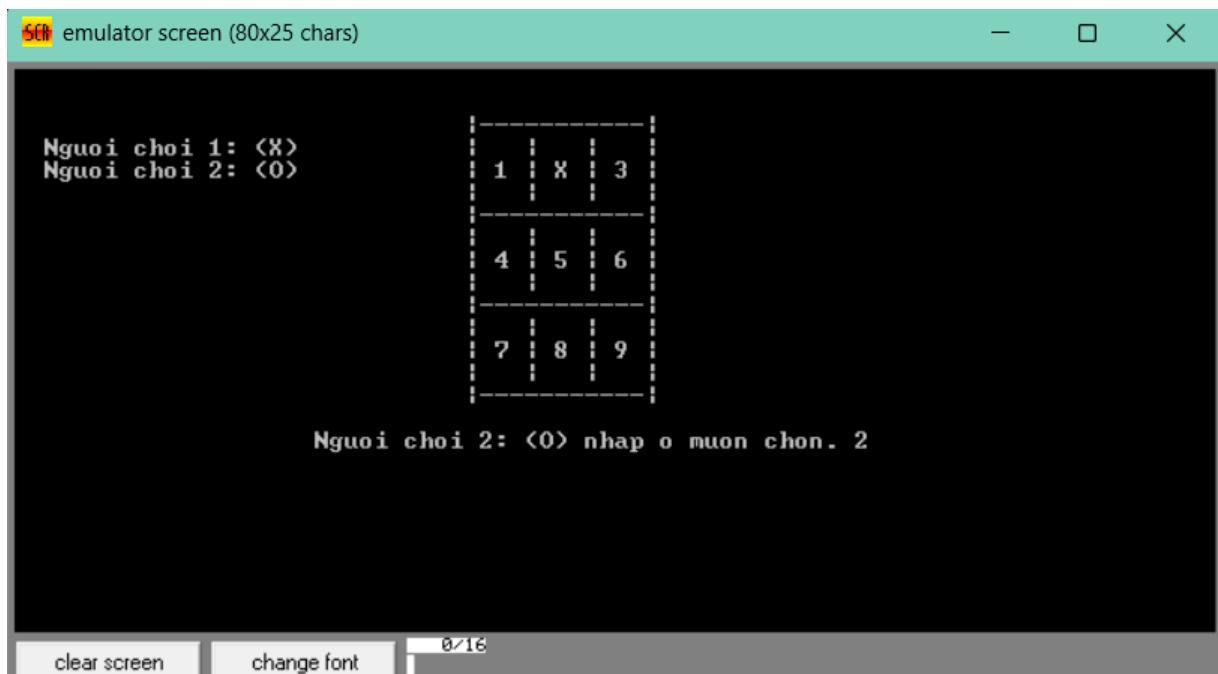


Chương trình tiếp tục và chuyển sang hiển thị bảng chơi và thông tin ký tự người chơi.

Yêu cầu người chơi 1 bắt đầu lượt đi trước bằng cách nhập ô muốn chọn tương ứng với nước đi mong muốn.



Sau khi người chơi 1 chọn nước đi, chuyển đến người chơi thứ 2 lựa chọn



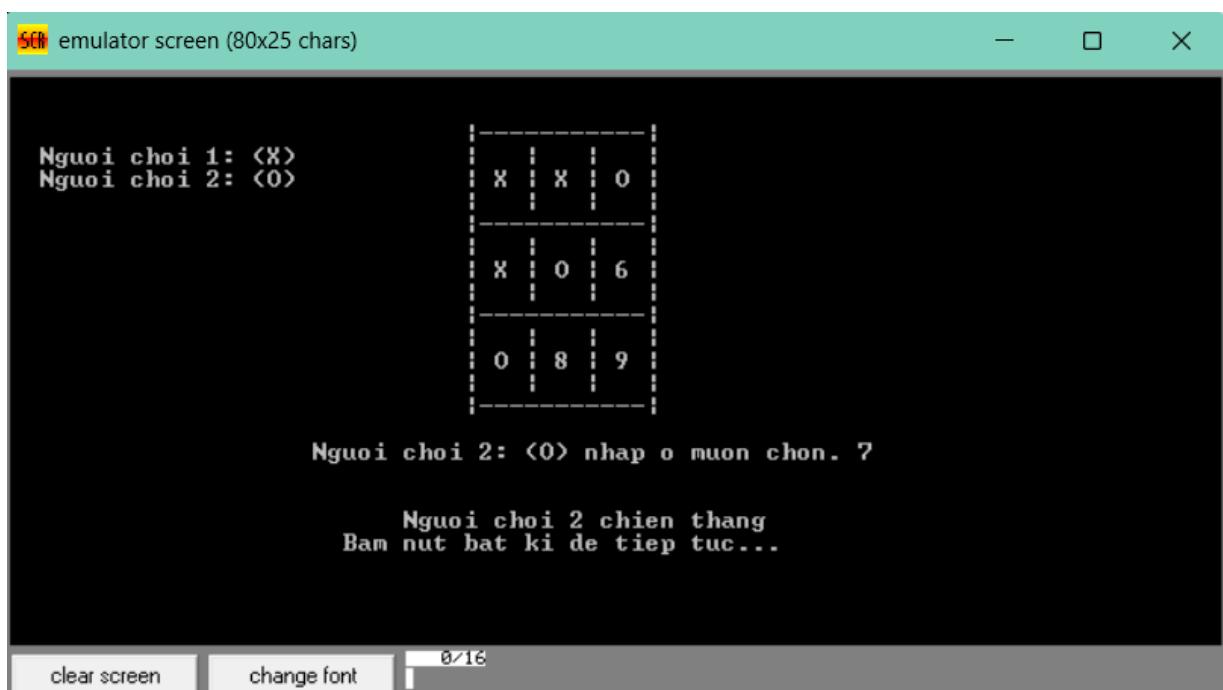
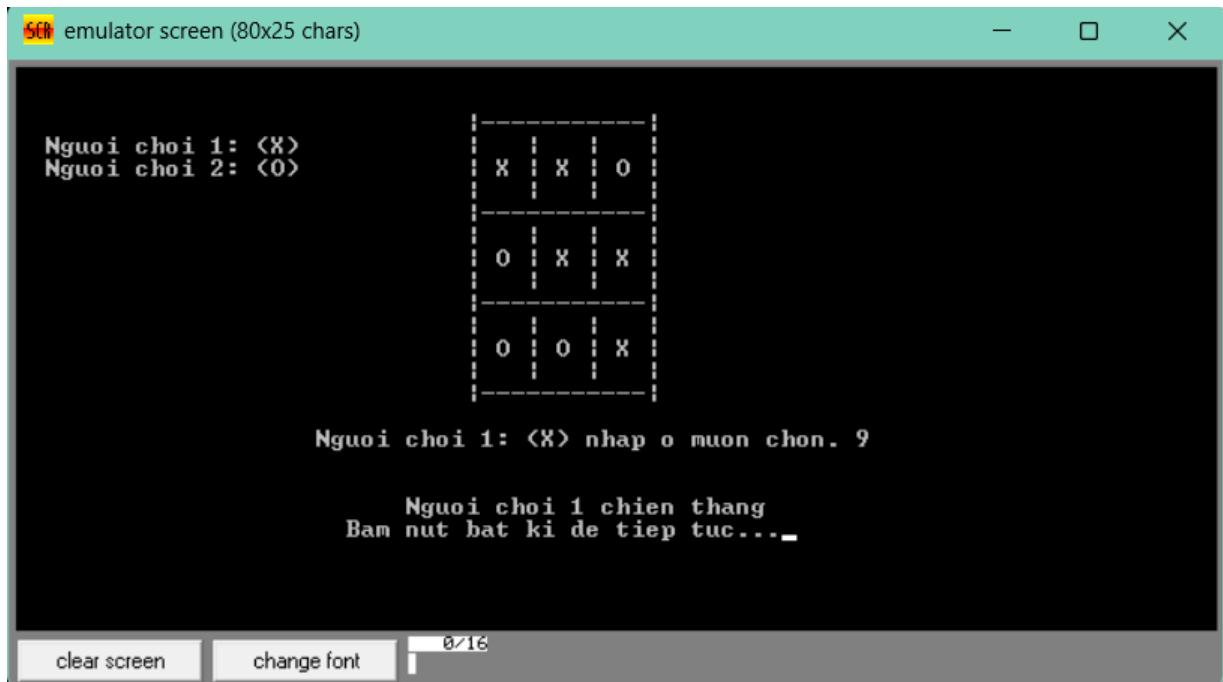
Nếu người chơi nhập sai ký tự, chương trình sẽ hiển thị thông báo và yêu cầu nhập lại

```
SCR emulator screen (80x25 chars)
-
X
Nguoi choi 1: <X>
Nguoi choi 2: <O>
1 | 8 | 3
-----
0 | 8 | 6
-----
7 | 0 | 9
-----
khong dung ki tu, moi ban nhap lai. y
clear screen change font 8/16
```

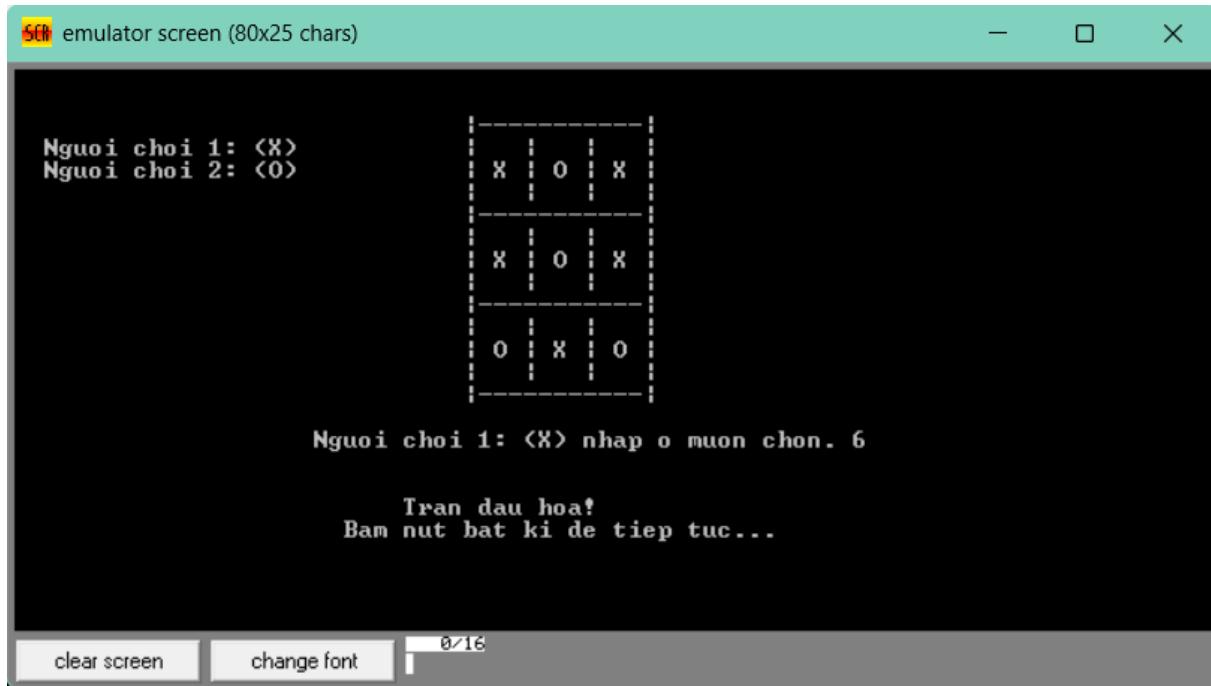
Nếu người chơi lựa chọn ô trùng, chương trình sẽ hiển thị thông báo và yêu cầu chọn lại

```
SCR emulator screen (80x25 chars)
-
X
Nguoi choi 1: <X>
Nguoi choi 2: <O>
1 | 8 | 3
-----
0 | X | X
-----
7 | 0 | 9
-----
O da duoc chon. Moi ban chon lai. . 2
clear screen change font 8/16
```

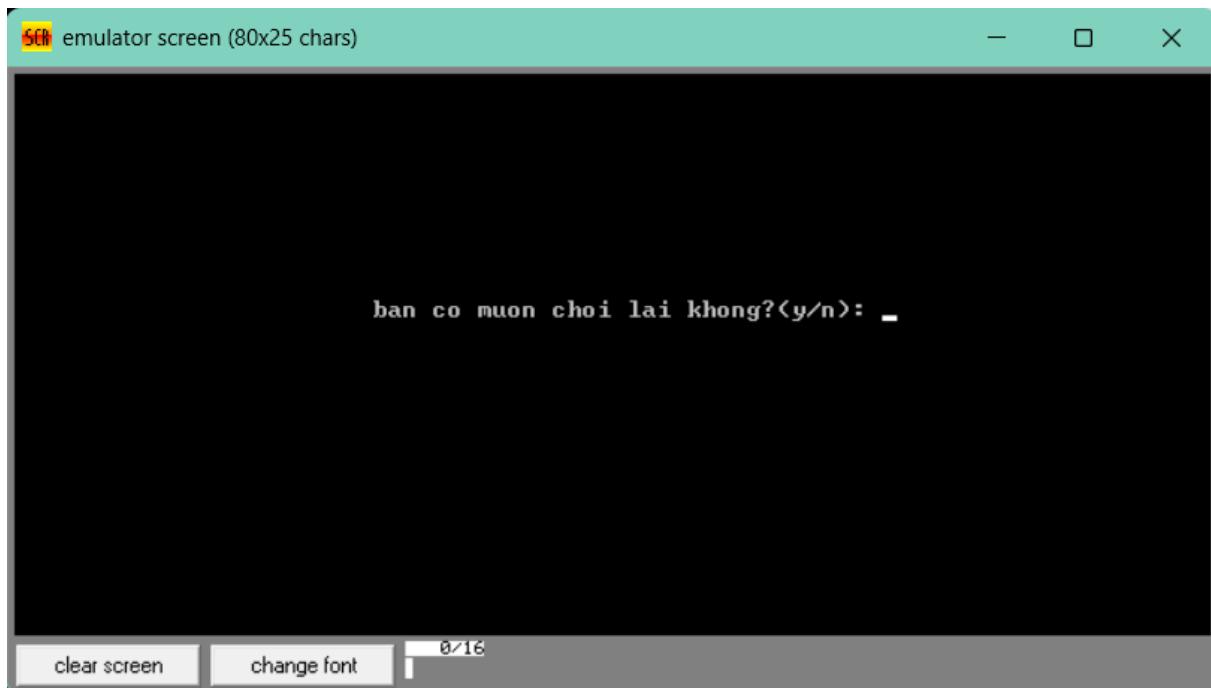
Khi có chiến thắng, chương trình hiển thị ra thông tin người chơi chiến thắng



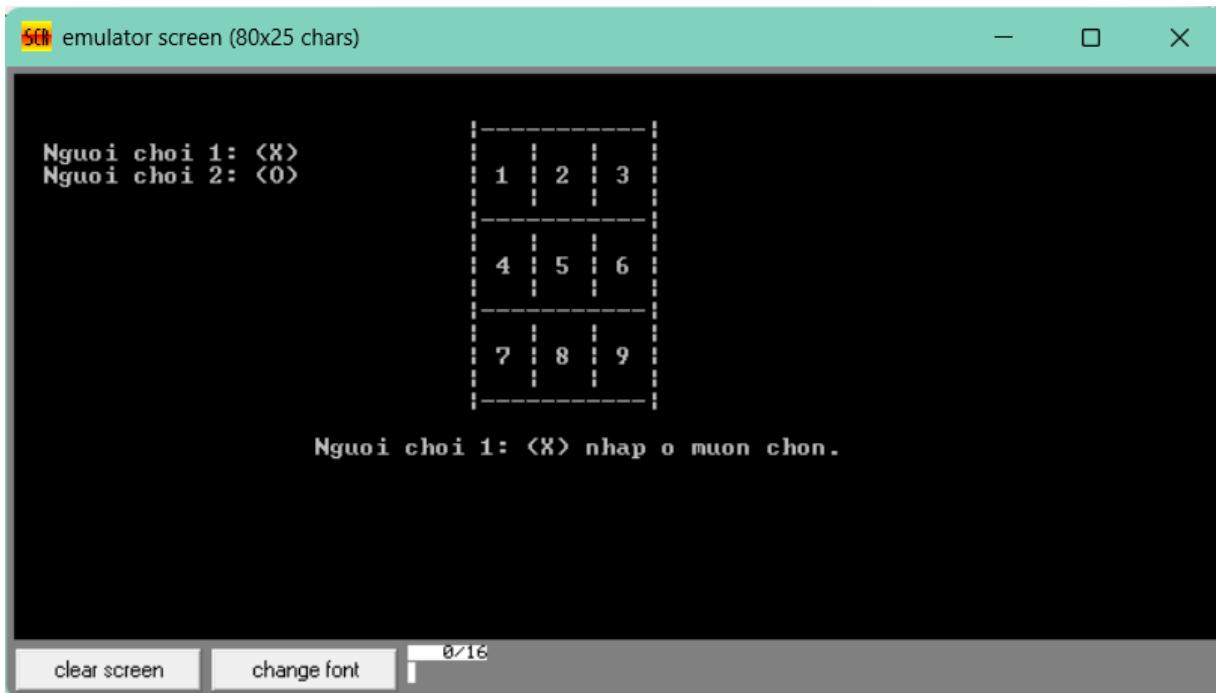
Khi 2 người chơi đã lập đầy bảng chơi mà chưa có chiến thắng, chương trình sẽ hiển thị kết quả hòa



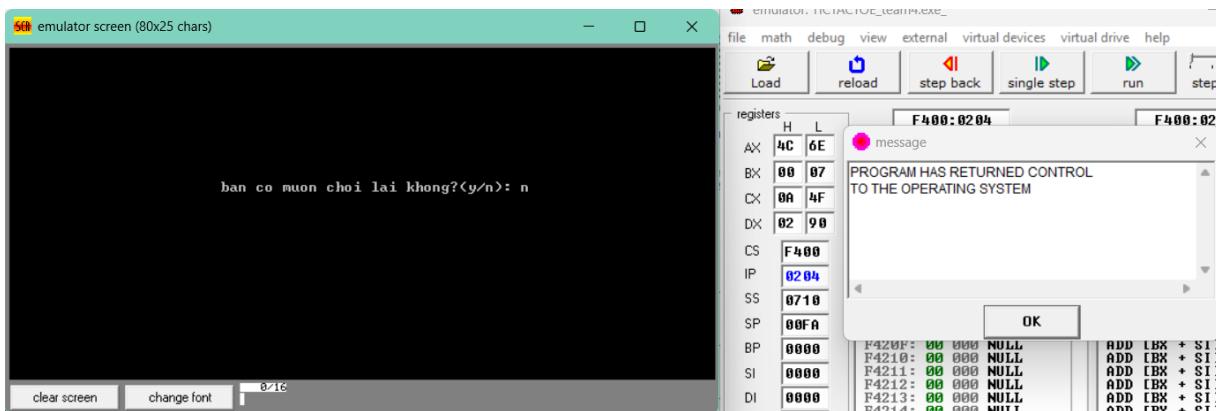
Chương trình tiếp tục bằng cách hỏi người chơi có muốn chơi lại hay không



Nếu nhập y/Y, chương trình sẽ trở về giao diện bảng chơi cho ván mới



Nếu nhập n/N, chương trình sẽ kết thúc



3.2. Kết luận

Kết quả kiểm thử cho thấy:

- Chương trình chạy ổn định
- Không phát sinh lỗi
- Các chức năng hoạt động đúng yêu cầu.

CHƯƠNG 2: LẬP TRÌNH MÁY TÍNH BMI TRÊN EMU8086

1. Giới thiệu

Trong bối cảnh hiện nay, khi sức khỏe ngày càng được quan tâm, việc theo dõi và đánh giá tình trạng cơ thể là vô cùng quan trọng. Một trong những chỉ số phổ biến để đánh giá nhanh tình trạng sức khoẻ của một người là BMI (Body Mass Index) – chỉ số khối cơ thể. Dựa trên chiều cao và cân nặng, chỉ số BMI giúp người dùng nhận biết được tình trạng thiếu cân, bình thường, thừa cân hoặc béo phì.

Đề tài "Máy tính BMI" trong bài tập lớn này được thực hiện nhằm xây dựng một chương trình tính chỉ số BMI bằng ngôn ngữ Assembly – một ngôn ngữ lập trình bậc thấp, gần với ngôn ngữ máy và chạy trên phần mềm mô phỏng emu8086.

Việc thực hiện đề tài này không chỉ giúp sinh viên củng cố kiến thức về lập trình hợp ngữ như hiểu rõ hơn về cách thức hoạt động của bộ vi xử lý, thanh ghi, và luồng điều khiển mà còn giúp sinh viên tiếp cận các ứng dụng thực tiễn, từ đó tăng khả năng kết hợp giữa lý thuyết và thực hành trong lĩnh vực kỹ thuật máy tính.

2. Nội dung chính của đề tài

2.1. Tổng quan về Máy tính BMI

Máy tính BMI cho phép người dùng nhập vào chiều cao (đơn vị mét) và cân nặng (đơn vị kg), sau đó tính toán chỉ số BMI dựa trên công thức chuẩn:

$$BMI = \frac{\text{Cân nặng}(kg)}{(\text{Chiều cao}(m))^2}$$

- Cách sử dụng
 - Lựa chọn chức năng cần thực hiện

Khi chương trình bắt đầu, người dùng được yêu cầu nhập các phím 1 hoặc 2 hoặc 3 hoặc 4 để máy tính thực hiện chức năng tương ứng.

- Nhập dữ liệu đầu vào

Khi chương trình yêu cầu, người dùng nhập vào cân nặng (đơn vị: kg) → nhấn Enter.

Tiếp theo, nhập chiều cao (đơn vị: cm) → nhấn Enter.

- Xử lý và hiển thị kết quả

Chương trình sẽ tự động tính toán chỉ số BMI theo công thức chuẩn.

Sau đó, in ra kết quả chỉ số BMI và thông báo tình trạng cơ thể tương ứng

- Các chức năng của chương trình:

- Tính toán chỉ số BMI: Dựa vào cân nặng (kg) và chiều cao (cm) do người dùng nhập vào, chương trình tính toán chỉ số BMI theo công thức.
- Xem bảng phân loại BMI Sau khi tính BMI, chương trình so sánh kết quả với bảng phân loại tiêu chuẩn (theo WHO dành cho người châu Á) để xác định tình trạng cơ thể.
- Tính cân nặng lý tưởng Giúp người dùng xác định mức cân nặng lý tưởng dựa trên chiều cao đã nhập.

- Đặc điểm của chương trình

- Chương trình được thiết kế đơn giản, người dùng dễ sử dụng
- Giao diện đơn giản, dễ nhìn

2.2. Giải thuật chính của chương trình

Thuật toán sử dụng:

- Khởi tạo chương trình và hiển thị giao diện:
 - Khi bắt đầu, hệ thống khởi tạo các thanh ghi và vùng nhớ cần thiết.
 - Hiển thị lời chào và hướng dẫn người dùng nhập cân nặng và chiều cao.
- Tiếp nhận và xác thực dữ liệu nhập:
 - Người dùng nhập:
 - Cân nặng (kg)
 - Chiều cao(cm)
 - Hệ thống:
 - Kiểm tra tính hợp lệ của đầu vào (đảm bảo là số hợp lệ), nếu không hợp lệ thì sẽ báo lỗi và nhập đến khi nào hợp lệ.
 - Xử lý dữ liệu nhập vào của người dùng để thuận tiện cho việc tính toán.
- Tính toán chỉ số BMI:
 - Áp dụng công thức tính BMI (quy đổi):
$$BMI = \frac{Cân\ nặng(kg) * 10000}{(Chiều\ cao(cm))^2}$$
 - Thực hiện nhân, chia và bình phương sử dụng các phép toán số học cơ bản trong Assembly để thực hiện tính toán.
 - Hiển thị chỉ số bmi tương ứng ra màn hình

- Tính cân nặng lý tưởng:

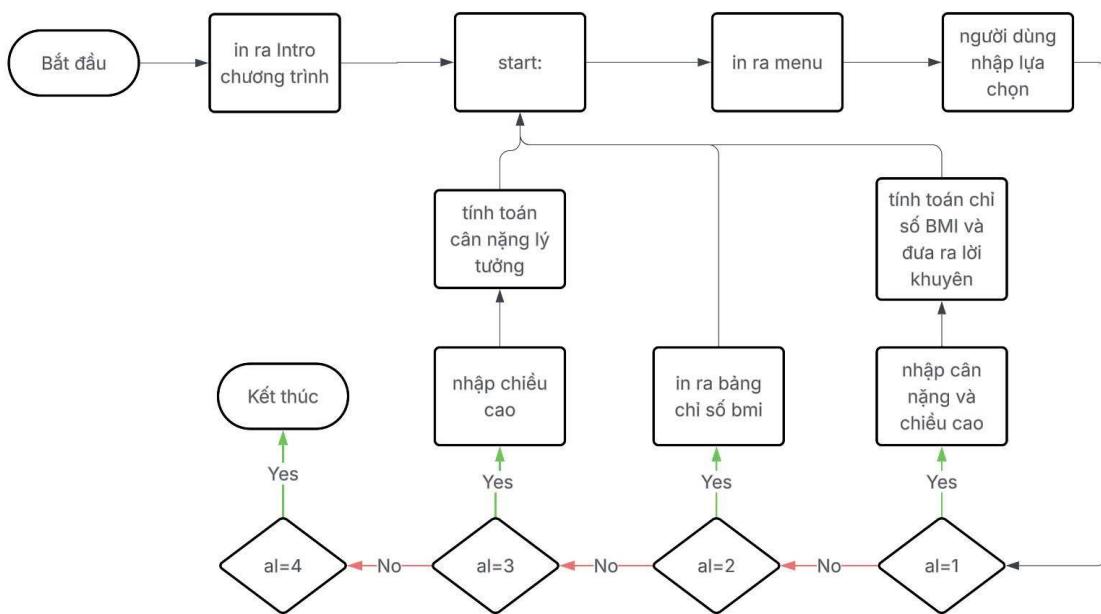
- Sử dụng công thức:

$$\text{Cân nặng lý tưởng}(kg) = BMI * \frac{(\text{Chiều cao}(cm))^2}{10000}$$

- Thực hiện các phép nhân và chia để ra cân nặng lý tưởng.
- Hiển thị kết quả ra màn hình cho người dùng.

2.3. Miêu tả chương trình

2.3.1. Lưu đồ thuật toán



2.3.2. Phân tích chương trình

```

include 'emu8086.inc'
.model small
.stack 100h
.data
    line1 db '|', 77 dup('-', '|', '$'
    line2 db '|', 77 dup(' ', '|', '$'

    1e1 dw 10          ; Hang so 10
    1e4 dw 10000       ; Hang so 10000
    chuso db ?         ; Bien tam luu tung chu so vua nhap
    weight dw ?        ; Can nang nguoi dung nhap (don vi: kg)
    height dw ?        ; Chieu cao nguoi dung nhap (don vi: cm)
    tittle db '== BMI CALCULATOR FOR HUMAN ==$'           ; Chuoi loi chao
    someword db 'MADE BY G4$'
    proceed db 'ENTER TO START...$'
    return db 'ENTER TO RETURN...$'
    xuong_dong db 10, 13, '$'; Ky tu xuong dong

    Hi dw 0            ; Phan cao cua tich 32 bit
    Lo dw 0            ; Phan thap cua tich 32 bit
    cham_phay db '.$.'
    ans_nguyen dw 0    ; Ky tu dau cham
    ans_du dw 0         ; Phan nguyen cua BMI
    ans_champhay dw 0   ; Phan du sau khi chia
                        ; Phan thap phan cua BMI

    loading dw 'LOADING....$'

    thank db 'THANK YOU SO MUCH$'

yeu_cau_nhap_can_nang dw 'Please enter your weight(kg):      $' ; Cau nhap can nang
yeu_cau_nhap_chieu_cao dw 'Please enter your height(cm):     $' ; Cau nhap chieu cao
ketqua dw 'This is your BMI : $'           ; Chuoi hien thi ket qua BMI

Error dw 'It seem that you made some mistakes ?$' ; Thong bao loi khi nhap sai
estimate dw 'Your state is $'                   ; 
blankstring db '                                $'

; BMI (Body Mass Index) classification table
; Cac muc BMI
tittle1 dw 'BMI (kg/m)$'
tittle2 dw 'Category$'

```

include ‘emu8086.inc’ : khai báo thư viện để sử dụng thủ tục clear screen

.model small: khai báo chương trình với kích cỡ nhỏ

.stack 100h: khai báo đoạn ngăn xếp với 256 byte

.data: Đánh dấu bắt đầu của phân đoạn dữ liệu, nơi lưu trữ dữ liệu tĩnh.

- Kiểu dữ liệu
 - db (defined byte) kích thước 1 byte dùng để lưu các giá trị nhỏ như số nguyên 0–255 (unsigned) hoặc -128 đến 127 (signed), hoặc 1 ký tự ASCII.
 - dw (defined word) kích thước 2 byte dùng để lưu các giá trị nhỏ như số nguyên 0–65535 (unsigned) hoặc -32768 đến 32767 (signed), hoặc 1 ký tự ASCII.
- Hàng số, biến và các dòng thông báo:
 - **1e1, 1e4:** các hàng số 10 và 10000
 - **chuso:** biến lưu từng chữ số vừa nhập

- **weight,height:** biến lưu cân nặng và chiều cao
- **Hi, Lo:** biến lưu phần thấp và phần cao của phép nhân 32 bit
- **ans_nguyen, ans_du, ans_champhay** để lưu phần nguyên, phần dư và phần thập phân của giá trị BMI
- **line1, line2, loading:** vẽ bảng màn hình loading
- **title, someword, proceed:** vẽ màn hình intro
- **option1, option2, option3, option4:** in ra các mục của menu
- **choice:** in ra thông báo lựa chọn
- **return:** in thông báo quay trở lại menu
- **xuong_dong:** xuống dòng
- **thank:** in thông báo cảm ơn
- **yeu_cau_nhap_can_nang, yeu_cau_nhap_chieu_cao:** thông báo để người dùng nhập cân nặng và chiều cao
- **ketqua:** thông báo để in ra chỉ số BMI
- **Error:** dòng thông báo khi nhập sai
- **estimate:** thông báo tình trạng sức khoẻ người dùng
- **blankstring:** xâu rỗng để ghi đè lên xâu ký tự khác
- **title1, title2:** thông báo các mức trong bảng phân loại BMI

```

muc1 dw 160
tb1.1 dw 'Below 16.0$'
tb1 dw 'Severely underweight$'

muc2 dw 170
tb2.1 dw '16.0 - 16.9$'
tb2 dw 'Moderately underweight$'

muc3 dw 185
tb3.1 dw '17.0 - 18.4$'
tb3 dw 'Mildly underweight$'

advice1 dw 'Eat more nutrient-rich meals and add strength training to build muscle.$' ;Advice for BMI < 18.5
muc4 dw 250
tb4.1 dw '18.5 - 24.9$'
tb4 dw 'Normal weight$'

advice2 dw 'Just Maintain a balanced diet and stay active to keep your weight stable.$' ;Advice : 18.5 - 24.9
muc5 dw 300
tb5.1 dw '25.0 - 29.9$'
tb5 dw 'Overweight$'

advice3 dw 'You should Focus on portion control and increase daily physical activity.$' ;Advice : 25.0 - 29.9

muc6 dw 350
tb6.1 dw '30.0 - 34.9$'
tb6 dw 'Obesity Class I (Moderate)$'

muc7 dw 400
tb7.1 dw '35.0 - 39.9$'
tb7 dw 'Obesity Class II (Severe)$'

tb8.1 dw '40.0 and above$'
tb8 dw 'Obesity Class III (Very severe)$'

advice4x1 dw 'You need to consult a healthcare provider for a personalized plan$' ;Advice for BMI > 40.0
advice4x2 dw 'And prioritize gradual lifestyle changes.$'
;Ideal Weight

IBMI dw 22
tb_Ideal_weight dw 'Your ideal weight is: $'

```

- **muc1** đến **muc7**: các hằng số để đánh giá chỉ số BMI (sau khi x10)
- **tb2,tb2.1** đến **tb8,tb8.1**: dòng thông báo chỉ số BMI với phân loại tương ứng
- **advice**: in ra lời khuyên với từng tình trạng sức khoẻ
- **IBMI**: biến lưu chỉ số BMI lý tưởng
- **tb_Ideal_weight**: dòng thông báo chỉ số BMI lý tưởng

.code: khai báo đoạn mã lệnh

```

main proc
    mov ax, @data
    mov ds, ax           ; Khoi tao thanh ghi DS de truy xuat bien
    call Intro
Start:
    call CLEAR_SCREEN   ;Xoa man hinh
    call DRAW_BORDER    ;In vien
    call MENU            ;In menu

    cmp al, '1'          ;Chay chuong trinh tinhh bmi
    jne skip1
    call CACULATE
    jmp start

skip1:
    cmp al, '2'          ;Chay chuong trinh in BMI TABLE
    jne skip2
    call BMI_TABLE
    jmp start

skip2:
    cmp al, '3'          ;Chay chuong trinh tinh can nang ly tuong
    jne skip3
    call IDEAL_WEIGHT
    jmp start

skip3:
    cmp al, '4'
    jne start

    call CLEAR_SCREEN   ;Xoa man hinh
    goto 30,12
    printf thank|
    mov ah, 4Ch           ; Ket thuc chuong trinh
    int 21h
main endp

```

- **main proc, main endp**: bắt đầu và kết thúc chương trình chính
- **mov ax, @data ; mov ds, ax** : đưa địa chỉ phần dữ liệu vào thanh ghi DS
- **call Intro**: gọi thủ tục Intro để hiển thị lời chào
- **call CLEAR_SCREEN**: xoá màn hình
- **call DRAW_BORDER**: in viền
- **call MENU**: in menu
- **cmp al, 1**: nếu bằng 1 thì **call CALCULATE**, ngược lại nhảy xuống skip1
- **cmp al, 2**: nếu bằng 2 thì **call BMI_TABLE**, ngược lại nhảy xuống skip2
- **cmp al, 3**: nếu bằng 3 thì **call IDEAL_WEIGHT**, ngược lại nhảy xuống skip3
- **cmp al, 4**: nếu bằng 4 thì thoát chương trình và in thông báo cảm ơn, ngược lại nhảy lại Start
- **mov ah, 4ch + int21h**: lệnh 4ch của ngắt 21h để kết thúc chương trình

```

;Dat vi tri con tro o vi tri x, y
goto macro x, y
    push ax
    push bx
    push dx
    mov ah, 2
    mov bh, 0
    mov dl, x
    mov dh, y
    int 10h
    pop dx
    pop bx
    pop ax

endm
;In 1 chuoi
printf macro str
    mov ah, 9
    lea dx, str
    int 21h

endm

```

Mã lệnh thực hiện như sau:

- **goto macro x,y** nhận 2 tham số x và y để di chuyển con trỏ đến vị trí cột x hàng y trong bảng:
 - **push ax, bx, dx** để đẩy dữ liệu của thanh ghi ax, bx, cx vào ngăn xếp tránh sai lệch dữ liệu của các thanh ghi này trong quá trình tính toán
 - **pop dx, bx, ax** lấy dữ liệu từ đỉnh stack vào lại các thanh ghi ban đầu để trả lại kết quả cũ
 - mã lệnh được lưu trong 2 thanh ghi ah, bh và toạ độ lưu trong thanh ghi dl và dh
 - **endm** để kết thúc thủ tục macro
- **printf macro str** dùng để in 1 chuỗi ra màn hình:
 - **lea dx, str**: nạp địa chỉ của chuỗi str vào thanh ghi dx
 - **mov ah, 9** của int 21h để in 1 chuỗi ra màn hình

Thủ tục **Menu**: in ra các chức năng chính cho người dùng lựa chọn

```
;Chuong trinh in Menu
MENU proc
    ;Dua con tro ve vi tri (x, y) va in option
    ;option 1
    goto 2,2
    printf option1

    ;option 2
    goto 2,4
    printf option2

    ;option 3
    goto 2,6
    printf option3

    ;option 4
    goto 2,8
    printf option4

    ;Nhap select option
Make_choice:
    goto 2,10

    mov ah, 9
    lea dx, choice
    int 21h

    mov ah, 1      ;nhap ki tu muon chon
    int 21h
    cmp al, '4'   ;Neu option > 4 thi nhap lai
    jg Make_choice

    cmp al, '1'   ;Neu option < 1 thi nhap lai
    jl Make_choice
ret
MENU endp
```

Mã lệnh thực hiện như sau:

- Các câu lệnh **goto 2, 2 ; goto 2,4** để đưa con trỏ đến các vị trí thích hợp trên bảng để in thông báo
- **printf option1 đến option 4:** in các thông báo lựa chọn
- Vòng lặp **Make_choice:**
 - In ra chuỗi choice ('Select an option: \$')
 - Sau đó dùng **mov ah, 1 + int 21h** để nhập 1 ký tự thể hiện lựa chọn, ký tự nhập lưu trong **al**
 - So sánh ký tự vừa nhập với 1 và 4 nếu nhỏ hơn 1 hoặc lớn hơn 4 thì nhập lại
- **ret :** kết thúc thủ tục

Thủ tục **DRAW_BORDER** để in viền

```
;chuong trinh in vien
DRAW_BORDER proc
    ;In chuoi loading
    goto 35,12
    printf loading

    ;Dat vi tri con tro bat dau la (0, 0)
    goto 0,0
    printf line1
    printf xuong_dong
    mov cx,22
    in_canh:
    printf line2
    printf xuong_dong
    loop in_canh
    printf line1

ret
DRAW_BORDER endp
```

Mã lệnh thực hiện như sau:

- **goto 35, 12:** cho con trỏ đến vị trí dòng 35, cột 12 và in chuỗi loading
- **goto 0,0:** Đặt vị trí con trỏ tại vị trí 0,0 để in line1
- **mov cx, 22:** tạo biến đếm để vòng lặp in_canh in ra line2 22 lần
- **printf line1:** In thêm line1 ở cuối
- **ret:** kết thúc thủ tục

Thủ tục **Intro**

```
;In Intro
Intro proc
    call DRAW_BORDER

    ;Di chuyen vi tri con tro va in
    goto 24, 2
    printf tittle

    goto 33,4
    printf someword

    goto 31,12
    printf proceed

    mov ah, 1 ;ENTER TO START
    int 21h
ret
Intro endp
```

Mã lệnh thực hiện như sau:

- Gọi thủ tục **DRAW_BORDER** để vẽ viền
- Di chuyển con trỏ (**goto x,y**) đến các vị trí để in ra các thông báo (tittle, someword, proceed) với printf macro đã được định nghĩa ở trên
- Mov ah,1 + int 21h: nhập phím bất kỳ, ở đây nhấn Enter để tiếp tục
- ret: kết thúc thủ tục

Thủ tục CALCULATE để tính chỉ số BMI

```
175 ;Chuong trinh tinh toan bmi
176 CACULATE proc
177     call CLEAR_SCREEN
178     call DRAW_BORDER
179
180 Input_Weight: ;Nhập cân nặng
181     ; Hiện thông báo nhập cân nặng
182     goto 2,2
183     printf 'yêu cầu_nhap_can_nang'
184
185     ; Gọi hàm nhập số nguyên
186     call read_num
187     mov weight, ax      ; Lưu giá trị nhập vào biến weight
188
189     ; Nếu weight = 0 thì nhập lại
190     cmp weight, 0
191     je Input_Weight
192
193     ; Tính từ số BMI = cân nặng * 10000
194     mov ax, weight
195     mul 1e4            ; AX * 10000 => kết quả 32 bit: DX:AX
196     mov Hi, dx
197     mov Lo, ax
198
199 Input_Height: ;Nhập chiều cao
200     ; Hiện thông báo nhập chiều cao
201     goto 2,4
202     printf 'yêu cầu_nhap_chieu_cao'
203
204     call read_num
205     mov height, ax      ; Lưu vào biến height
206
207     cmp height, 0
208     je Input_Height
209
210     ; Tính height^2
211     mov ax, height
212     mul height
213     mov cx, ax          ; cx = height^2
214
215     mov height, cx      ; cập nhật height thành height^2 để sử dụng sau
216
217     ; BMI = (cân nặng * 10000) / (height^2)
218     mov dx, Hi           ; phần cao của số chia
219     mov ax, Lo           ; phần thấp
220     div cx              ; kết quả chia 32-bit / 16-bit
221
222     mov ans_nguyen, ax  ; phần nguyên của BMI
223     mov ans_du, dx       ; phần dư
224
225
226
```

Mã lệnh thực hiện như sau:

- Bắt đầu chương trình:

```
175 ;Chuong trinh tinh toan bmi
176 CACULATE proc
177     call CLEAR_SCREEN
178     call DRAW_BORDER
```

- **call CLEAR_SCREEN:** Gọi thủ tục CLEAR_SCREEN để xóa toàn bộ nội dung trên màn hình.
- **call DRAW_BORDER:** Gọi thủ tục DRAW_BORDER để vẽ khung viền cho giao diện.
- Nhập cân nặng:

```
180 Input_Weight: ;Nhập cân nặng
181     ; Hiện thông báo nhập cân nặng
182     goto 2,2
183     printf 'yêu cầu_nhap_can_nang'
184
185     ; Gọi hàm nhập số nguyên
186     call read_num
187     mov weight, ax      ; Lưu giá trị nhập vào biến weight
188
189     ; Nếu weight = 0 thì nhập lại
190     cmp weight, 0
191     je Input_Weight
192
193     ; Tính từ số BMI = cân nặng * 10000
194     mov ax, weight
195     mul 1e4            ; AX * 10000 => kết quả 32 bit: DX:AX
196     mov Hi, dx
197     mov Lo, ax
```

- Hiển thị thông báo nhập cân nặng:
 - **goto 2,2:** Di chuyển con trỏ đến vị trí cột 2, dòng 2.
 - **printf yeu_cau_nhap_can_nang:** Hiển thị chuỗi "Please enter your weight(kg): "
- Nhập cân nặng từ bàn phím:
 - **call read_num:** Gọi thủ tục read_num để đọc một số nguyên từ bàn phím. Kết quả trả về trong thanh ghi AX.
 - **mov weight, ax:** Lưu giá trị cân nặng vừa nhập (từ AX) vào biến 'weight'.
 - **cmp weight, 0:** So sánh cân nặng với 0.
 - **je Input_Weight:** Nếu cân nặng bằng 0 (Jump if Equal), nhảy về nhãn Input_Weight để yêu cầu nhập lại.
- Nhập chiều cao:
 - **call read_num:** Gọi thủ tục read_num để đọc một số nguyên từ bàn phím. Kết quả trả về trong thanh ghi AX.
 - **mov height, ax:** Lưu giá trị chiều cao vừa nhập (từ AX) vào biến 'height'.
 - **cmp height, 0:** So sánh chiều cao với 0.
 - **je Input_Height:** Nếu chiều cao bằng 0 (Jump if Equal), nhảy về nhãn Input_Height để yêu cầu nhập lại.

```

200 Input_Height: ;Nhap chieu cao
201 ; Hien thi thong bao nhap chieu cao
202 goto 2,4
203 printf yeu_cau_nhap_chieu_cao
204
205 call read_num
206 mov height, ax      ; Luu vao bien height
207
208 cmp height, 0
209 je Input_Height
210
211 ; Tinh height^2
212 mov ax, height
213 mul height
214 mov cx, ax          ; cx = height^2
215
216 mov height, cx      ; cap nhat height thanh height^2 de su dung sau
217

```

- Hiển thị thông báo nhập chiều cao:
 - **goto 2,4:** Di chuyển con trỏ đến vị trí cột 2, dòng 4.
 - **printf yeu_cau_nhap_chieu_cao:** Hiển thị chuỗi "Please enter your height(cm): ".
- Nhập cân nặng từ bàn phím:
 - **call read_num:** Gọi thủ tục read_num để đọc một số nguyên từ bàn phím. Kết quả trả về trong thanh ghi AX.
 - **mov height, ax:** Lưu giá trị chiều cao vào biến 'height'.
 - **cmp height, 0:** So sánh chiều cao với 0.
 - **je Input_Height:** Nếu chiều cao bằng 0, nhảy về Input_Height để nhập lại.

- Tính mẫu số của công thức **BMI = chiều cao*chiều cao**:
 - **mov ax, height**: Nạp chiều cao vào AX.
 - **mul height**: Nhân AX với chính nó (height * height).
 - **mov cx, ax**: Lưu kết quả height^2 vào thanh ghi CX. CX sẽ là số chia.
 - **mov height, cx**: Cập nhật biến 'height' bằng giá trị height^2. (Biến 'height' bây giờ chứa giá trị height^2, không phải chiều cao ban đầu. Điều này được sử dụng lại ở phần tính số thập phân.)

- Tính và in ra BMI:

```

219 ; BMI = <cân nặng * 10000> / <height^2>
220 mov dx, Hi           ; phan cao cua so chia
221 mov ax, Lo           ; phan thap
222 div cx               ; ket qua chia 32-bit / 16-bit
223
224 mov ans_nguyen, ax   ; phan nguyen cua BMI
225 mov ans_du, dx        ; phan du
226
227 ; In thong bao ket qua
228 goto 2,6
229 printf ketqua
230
231 ; In phan nguyen cua BMI
232 mov ax, ans_nguyen
233 call print_num
234
235 ; In dau cham
236 lea dx, cham_phay
237 mov ah, 9
238 int 21h
239
240 ; Tinh va in phan thap phan: <du * 10> / height^2
241 mov ax, ans_du
242 mul ie1
243 mov cx, height
244 div cx
245 mov ans_champhay, ax
246 call print_num

```

- Thực hiện phép tính BMI:
 - **mov dx, Hi**: Nạp phần cao của (cân nặng * 10000) vào DX.
 - **mov ax, Lo**: Nạp phần thấp của (cân nặng * 10000) vào AX.
 - **div cx**: Chia DX:AX cho CX (chứa height^2). Kết quả: Thương số (phần nguyên của BMI lưu trong AX. Số dư lưu trong DX.)
 - **mov ans_nguyen, ax**: Lưu phần nguyên của BMI vào biến 'ans_nguyen'.
 - **mov ans_du, dx**: Lưu phần dư của phép chia vào biến 'ans_du'.
- Hiển thị thông báo kết quả:
 - **goto 2,6**: Di chuyển con trỏ đến vị trí cột 2, dòng 6.
 - **printf ketqua**: Hiển thị chuỗi "This is your BMI: ".
- In ra phần nguyên của BMI:
 - **mov ax, ans_nguyen**: Nạp phần nguyên của BMI vào AX.
 - **call print_num**: Gọi thủ tục print_num để hiển thị số nguyên này ra màn hình.
 - **lea dx, cham_phay**: Nạp địa chỉ của chuỗi ".\$" vào DX.
 - **mov ah, 9**: Chức năng 09h của ngắt 21h: hiển thị chuỗi.

- int 21h: Gọi ngắt DOS.
- Tính và in ra phần thập phân của BMI = (dư * 10) / height^2 (lấy sau dấu thập phân một chữ số):
 - **mov ax, ans_du:** Nạp số dư (ans_du) vào AX.
 - **mul 1e1:** Nhân AX với 1e1 (hằng số 10). Kết quả (ans_du * 10) trong DX:AX.
 - **mov cx, height:** Nạp giá trị của height^2 vào CX (đã được lưu trong biến 'height' trước đó).
 - **div cx:** Chia (ans_du * 10) cho height^2. Thương số (chữ số thập phân đầu tiên) lưu trong AX.
 - **mov ans_champhay, ax:** Lưu chữ số thập phân này vào biến 'ans_champhay'.
 - **call print_num:** Gọi thủ tục print_num để hiển thị chữ số thập phân.
- Phân loại BMI và trở lại menu:

```

248 ; Goi ham phan loai BMI
249 goto 2,10
250 call phan_loai
251
252 ; in ra return
253 goto 2,15
254 printf return
255 ; Nhan phim lua chon
256 mov ah, 1
257 int 21h
258
259
260
261 ret
262 CACULATE endp
263

```

Phân loại BMI:

- **goto 2,10:** Di chuyển con trỏ đến vị trí cột 2, dòng 10.
- **call phan_loai:** Gọi thủ tục **phan_loai** để đánh giá BMI và đưa ra lời khuyên.
- Hiển thị thông báo return:
 - **goto 2,15:** Di chuyển con trỏ đến vị trí cột 2, dòng 15.
 - **printf return:** Hiển thị chuỗi "ENTER TO RETURN...".

Trở lại menu:

- **mov ah, 1:** Chức năng 01h của ngắt 21h: đọc một ký tự từ bàn phím (có hiển thị).
- **int 21h:** Gọi ngắt DOS. Chương trình sẽ dừng ở đây cho đến khi người dùng nhấn một phím.
- **ret:** kết thúc thủ tục.

Bảng phân loại BMI

```
257 ;Chuong trinh hien bang bmi
258 BMI_TABLE proc
259     call CLEAR_SCREEN      ;Xoa man hinh
260     call DRAW_BORDER        ;In vien
261
262     ;Set vi tri con tro va in thong tin
263     goto 0,3
264     printf line1
265
266     ;In title
267     goto 2,2
268     printf tittle1
269
270     goto 40,2
271     printf tittle2
272
273     ;In muc 1
274     goto 2,4
275     printf tb1.1
276     goto 40,4
277     printf tb1
278
279     ;In muc 2
280     goto 2,6
281     printf tb2.1
282     goto 40,6
283     printf tb2
284
285     ;In muc 3
286     goto 2,8
287     printf tb3.1
288     goto 40,8
289     printf tb3
290
291     ;In muc 4
292     goto 2,10
293     printf tb4.1
294     goto 40,10
295     printf tb4
296
297     ;In muc 5
298     goto 2,12
299     printf tb5.1
300     goto 40,12
301     printf tb5
302
303     ;In muc 6
304     goto 2,14
305     printf tb6.1
306     goto 40,14
307     printf tb6
308
```

Mã lệnh thực hiện như sau:

- Bắt đầu chương trình:
- **call CLEAR_SCREEN:** Gọi thủ tục CLEAR_SCREEN để xóa toàn bộ nội dung trên màn hình.
- **call DRAW_BORDER:** Gọi thủ tục DRAW_BORDER để vẽ khung viền cho giao diện.

In tiêu đề bảng phân loại BMI:

```
262      ;Set vi tri con tro va in thong tin
263      goto 0,3
264      printf line1
265
266      ;In title
267      goto 2,2
268      printf tittle1
269
270      goto 40,2
271      printf tittle2
```

In các thông tin phân loại:

```
273      ;In muc 1
274      goto 2,4
275      printf tb1.1
276      goto 40,4
277      printf tb1
278
279      ;In muc 2
280      goto 2,6
281      printf tb2.1
282      goto 40,6
283      printf tb2
284
285      ;In muc 3
286      goto 2,8
287      printf tb3.1
288      goto 40,8
289      printf tb3
290
291      ;In muc 4
292      goto 2,10
293      printf tb4.1
294      goto 40,10
295      printf tb4
296
297      ;In muc 5
298      goto 2,12
299      printf tb5.1
300      goto 40,12
301      printf tb5
302
303      ;In muc 6
304      goto 2,14
305      printf tb6.1
306      goto 40,14
307      printf tb6
308
309      ;In muc 7
310      goto 2,16
311      printf tb7.1
312      goto 40,16
313      printf tb7
314
315      ;In muc 8
316      goto 2,18
317      printf tb8.1
318      goto 40,18
319      printf tb8
320
321      ;Quay tro lai menu
322      goto 2,22
323      printf return
324
325      mov ah, 1 ;Enter to return
326      int 21h
327
328      ret
329      BMI_TABLE endp
```

- In mục 1 (In dòng đầu tiên của bảng dữ liệu)
 - **goto 2,4:** Di chuyển con trỏ đến cột 2, dòng 4.
 - **printf tb1.1:** Hiển thị chuỗi "Below 16.0".
 - **goto 40,4:** Di chuyển con trỏ đến cột 40, vẫn ở dòng 4.
 - **printf tb1:** Hiển thị chuỗi " Severely underweight".
- In mục 2 (In dòng thứ hai của bảng dữ liệu)
 - **goto 2,6:** Di chuyển con trỏ đến cột 2, dòng 6 .
 - **printf tb2.1:** Hiển thị chuỗi "16.0 - 16.9".
 - **goto 40,6:** Di chuyển con trỏ đến cột 40, dòng 6.
 - **printf tb2:** Hiển thị chuỗi " Moderately underweight".
- In mục 3 (In dòng thứ ba của bảng dữ liệu)
 - **goto 2,8:** Di chuyển con trỏ đến cột 2, dòng 8.
 - **printf tb3.1:** Hiển thị chuỗi "17.0 - 18.4".
 - **goto 40,8:** Di chuyển con trỏ đến cột 40, dòng 8.
 - **printf tb3:** Hiển thị chuỗi "Mildly underweight".
- In mục 4 (In dòng thứ tư của bảng dữ liệu)
 - **goto 2,10:** Di chuyển con trỏ đến cột 2, dòng 10.
 - **printf tb4.1:** Hiển thị chuỗi "18.5 - 24.9".
 - **goto 40,10:** Di chuyển con trỏ đến cột 40, dòng 10.
 - **printf tb4:** Hiển thị chuỗi " Normal weight ".
- In mục 5 (In dòng thứ năm của bảng dữ liệu)
 - **goto 2,12:** Di chuyển con trỏ đến cột 2, dòng 12.
 - **printf tb5.1:** Hiển thị chuỗi "25.0 - 29.9".
 - **goto 40,12:** Di chuyển con trỏ đến cột 40, dòng 12.
 - **printf tb5:** Hiển thị chuỗi " Overweight ".
- In mục 6 (In dòng thứ sáu của bảng dữ liệu)
 - **goto 2,14:** Di chuyển con trỏ đến cột 2, dòng 14.
 - **printf tb6.1:** Hiển thị chuỗi "30.0 - 34.9".
 - **goto 40,14:** Di chuyển con trỏ đến cột 40, dòng 14.
 - **printf tb6:** Hiển thị chuỗi "Obesity Class I (Moderate)".
- In mục 7 (In dòng thứ bảy của bảng dữ liệu)
 - **goto 2,16:** Di chuyển con trỏ đến cột 2, dòng 16.
 - **printf tb7.1:** Hiển thị chuỗi "35.0 - 39.9".
 - **goto 40,16:** Di chuyển con trỏ đến cột 40, dòng 16.

- **printf tb7:** Hiển thị chuỗi " Obesity Class II (Severe)".
- In mục 8 (In dòng thứ tám của bảng dữ liệu - "Obesity Class III (Very severe)")
- **goto 2,18:** Di chuyển con trỏ đến cột 2, dòng 18.
- **printf tb8.1:** Hiển thị chuỗi "40.0 and above".
- **goto 40,18:** Di chuyển con trỏ đến cột 40, dòng 18.
- **printf tb8:** Hiển thị chuỗi " Obesity Class III (Very severe)".
- Quay trở lại menu (Thông báo cho người dùng cách quay lại)
- **goto 2,22:** Di chuyển con trỏ đến cột 2, dòng 22.
- **printf return:** Hiển thị chuỗi "ENTER TO RETURN...".
- **mov ah, 1:** Chuẩn bị đọc một ký tự từ bàn phím.
- **int 21h:** Gọi ngắt DOS. Chương trình sẽ dừng ở đây cho đến khi người dùng nhấn một phím.
- **ret:** kết thúc thủ tục.

Tính cân nặng lý tưởng

```

332 IDEAL_WEIGHT proc
333     call CLEAR_SCREEN ;Xoa man hinh
334     call DRAW_BORDER ;In vien
335
336     startideal: |
337     goto 2, 2 ;Dua con tro ve vi tri <2,2>
338 ; Hien thi thong bao nhap chieu cao
339     printf yeu_cau_nhap_chieu_cao
340     goto 32,2
341
342     call read_num
343     cmp ax,0
344     je startideal
345     mov height, ax      ; Luu vao bien height
346
347     goto 2, 4 ;Dua con tro ve vi tri <2,4>
348
349     printf tb_Ideal_weight
350
351     mov ax, height
352     mul height
353 ;Ideal BMI = 22
354     mul IBMI
355     div ie4
356
357     call print_num
358
359     goto 2, 6 ;Dua con tro ve vi tri <2,6>
360     printf return
361
362     mov ah, 1    ;Enter to return
363     int 21h
364
365     ret
366 IDEAL_WEIGHT endp

```

Mã lệnh thực hiện như sau:

- Bắt đầu chương trình:

```

332 IDEAL_WEIGHT proc
333     call CLEAR_SCREEN ;Xoa man hinh
334     call DRAW_BORDER ;In vien

```

- **call CLEAR_SCREEN:** Gọi thủ tục CLEAR_SCREEN để xóa toàn bộ nội dung trên màn hình.
- **call DRAW_BORDER:** Gọi thủ tục DRAW_BORDER để vẽ khung viền cho giao diện.
- Nhập chiều cao:

```

336      startideal:
337      goto 2, 2 ;Dua con tro ve vi tri (2,2)
338      ; Hien thi thong bao nhap chieu cao
339      printf yeu_cau_nhap_chieu_cao
340      goto 32,2
341
342      call read_num
343      cmp ax,0
344      je startideal
345      mov height, ax      ; Luu vao bien height
346

```

- **startideal:** Nhãn để có thể quay lại đây nếu người dùng nhập chiều cao không hợp lệ.
- **goto 2, 2:** Di chuyển con trỏ đến cột 2, dòng 2.
- **printf yeu_cau_nhap_chieu_cao:** 5. Hiển thị chuỗi "Please enter your height(cm):"
- **goto 32,2:** Di chuyển con trỏ đến cột 32, dòng 2.
- **call read_num:** Gọi thủ tục 'read_num' để đọc một số nguyên (chiều cao) từ bàn phím. Kết quả (chiều cao) được trả về trong thanh ghi AX.
- **cmp ax,0:** So sánh chiều cao vừa nhập (trong AX) với 0.
- **je startideal:** Nếu chiều cao bằng 0 (jump if Equal), quay lại nhãn 'startideal' để yêu cầu nhập lại.
- **mov height, ax:** Nếu chiều cao hợp lệ (khác 0), lưu giá trị từ AX vào biến 'height'.

Tính toán và in ra cân nặng lý tưởng:

```

347      goto 2, 4 ;Dua con tro ve vi tri (2,4)
348
349      printf tb_Ideal_weight
350
351      mov ax, height
352      mul height
353      ;Ideal_BMI = 22
354      mul IBMI
355      div 1e4
356
357      call print_num
358
359      goto 2, 6 ;Dua con tro ve vi tri (2,6)
360      printf return
361
362      mov ah, 1    ;Enter to return
363      int 21h
364
365      ret
366 IDEAL_WEIGHT endp

```

- **goto 2, 4:** Di chuyển con trỏ đến cột 2, dòng 4.
- **printf tb_Ideal_weight:** Hiển thị chuỗi "Your ideal weight is: "
- **mov ax, height:** Nạp giá trị chiều cao (đã lưu ở bước 10) vào thanh ghi AX.
- **mul height:** Nhân AX với chính nó ($height * height = height^2$). Kết quả 32-bit của phép nhân nằm trong DX:AX.
- **mul IBMI:** Nhân AX (chứa $height_cm^2$) với IBMI (hằng số 22). Kết quả 32-bit của ($height_cm^2 * 22$) nằm trong DX:AX.
- **div 1e4:** Chia DX:AX (chứa $height_cm^2 * 22$) cho 1e4 (hằng số 10000). Thương số (phần nguyên của cân nặng lý tưởng) được lưu trong AX. Phần dư được lưu trong DX (không được sử dụng ở đây).
- **call print_num:** Gọi thủ tục 'print_num' để hiển thị giá trị cân nặng lý tưởng (từ AX) ra màn hình.
- **goto 2, 6:** Di chuyển con trỏ đến cột 2, dòng 6.
- **printf return:** Hiển thị chuỗi "ENTER TO RETURN...\$".
- **mov ah, 1:** Chuẩn bị đọc một ký tự từ bàn phím.
- **int 21h:** Gọi ngắt DOS. Chương trình sẽ dừng ở đây cho đến khi người dùng nhấn một phím.
- **ret:** kết thúc thủ tục.

Đánh giá BMI dựa trên kết quả tính được

```

444 ; === Ham danh gia BMI dua tren ket qua tinh duoc ===
445 phan_loai proc
446     printf estimate
447     ; Tinh BMI * 10 de so sanh so nguyen
448     mov ax, ans_nguyen
449     mul 1e1
450     add ax, ans_champhay
451
452     ; So sanh theo nguong BMI va in ra advice
453     cmp ax, muc1
454     jl Muc_1
455     cmp ax, muc2
456     jl Muc_2
457     cmp ax, muc3
458     jl Muc_3
459     cmp ax, muc4
460     jl Muc_4
461     cmp ax, muc5
462     jl Muc_5
463     cmp ax, muc6
464     jl Muc_6
465     cmp ax, muc7
466     jl Muc_7
467     jmp Muc_8

```

```

469 Muc_1:
470     printf tb1
471     goto 2,12
472     printf advice1
473     ret
474 Muc_2:
475     printf tb2
476     goto 2,12
477     printf advice1
478     ret
479 Muc_3:
480     printf tb3
481     goto 2,12
482     printf advice1
483     ret
484 Muc_4:
485     printf tb4
486     goto 2,12
487     printf advice2
488     ret
489 Muc_5:
490     printf tb5
491     goto 2,12
492     printf advice3
493     ret
494 Muc_6:
495     printf tb6
496     goto 2,12
497     printf advice4x1
498     goto 2,13
499     printf advice4x2
500     ret
501 Muc_7:
502     printf tb7
503     goto 2,12
504     printf advice4x1
505     goto 2,13
506     printf advice4x2
507     ret
508 Muc_8:
509     printf tb8
510     goto 2,12
511     printf advice4x1
512     goto 2,13
513     printf advice4x2
514     ret
515 phan_loai endp

```

Mã lệnh thực hiện như sau:

- Bắt đầu chương trình và so sánh các mức BMI:

```

444 ; === Ham danh gia BMI dua tren ket qua tinh duoc ===
445 phan_loai proc
446     printf estimate
447     ; Tinh BMI * 10 de so sanh so nguyen
448     mov ax, ans_nguyen
449     mul iei
450     add ax, ans_champhay
451
452     ; So sanh theo nguong BMI va in ra advice
453     cmp ax, muc1
454     jl Muc_1
455     cmp ax, muc2
456     jl Muc_2
457     cmp ax, muc3
458     jl Muc_3
459     cmp ax, muc4
460     jl Muc_4
461     cmp ax, muc5
462     jl Muc_5
463     cmp ax, muc6
464     jl Muc_6
465     cmp ax, muc7
466     jl Muc_7
467     jmp Muc_8

```

- **printf estimate:** In ra chuỗi "Your state is "
- **mov ax, ans_nguyen:** Nạp phần nguyên của BMI vào thanh ghi AX.
- **mul 1e1:** Nhân AX với 1e1 (hằng số 10).
- **add ax, ans_champhay:** Cộng chữ số thập phân đầu tiên của BMI vào AX.
- Sử dụng chuỗi lệnh **cmp** kết hợp **jl** để phân loại chỉ số BMI theo các ngưỡng đã định nghĩa trước (muc1, muc2, ..., muc7). Cụ thể:
 - + **cmp ax, muc[i]:** So sánh giá trị AX (đã quy đổi thành BMI × 10) lần lượt với các ngưỡng muc1 đến muc7.
 - + Khi tìm được khoảng phù hợp, chương trình nhảy (jl) đến nhãn **Muc_[i]** tương ứng để in thông báo và lời khuyên.
- Nếu không thỏa mãn bất kỳ điều kiện nào phía trên (tức $BMI \geq 40.0$), chương trình nhảy đến nhãn **Muc_8**.

In ra đánh giá sau khi so sánh:

```

469 Muc_1:
470     printf tb1
471     goto 2,12
472     printf advice1
473     ret
474 Muc_2:
475     printf tb2
476     goto 2,12
477     printf advice1
478     ret
479 Muc_3:
480     printf tb3
481     goto 2,12
482     printf advice1
483     ret
484 Muc_4:
485     printf tb4
486     goto 2,12
487     printf advice2
488     ret
489 Muc_5:
490     printf tb5
491     goto 2,12
492     printf advice3
493     ret
494 Muc_6:
495     printf tb6
496     goto 2,12
497     printf advice4x1
498     goto 2,13
499     printf advice4x2
500     ret
501 Muc_7:
502     printf tb7
503     goto 2,12
504     printf advice4x1
505     goto 2,13
506     printf advice4x2
507     ret
508 Muc_8:
509     printf tb8
510     goto 2,12
511     printf advice4x1
512     goto 2,13
513     printf advice4x2
514     ret
515 Muc_9:
516     printf tb9
517     goto 2,12
518     printf advice4x1
519     goto 2,13
520     printf advice4x2
521     ret
522 phan_loai endp

```

Dựa trên giá trị BMI đã tính được, chương trình sử dụng nhãn **Muc_1** đến **Muc_8** để phân loại tình trạng cơ thể và hiển thị kết quả tương ứng.

Cụ thể:

- Tại mỗi nhän, chương trình dùng **printf** để in nội dung từ biến **tb[i]** (mô tả mức BMI) và **advice[i]** (lời khuyên sức khỏe). Đồng thời sử dụng lệnh **goto** để đưa con trỏ tới vị trí mong muốn trên màn hình
- Kết thúc mỗi nhän là lệnh ret để trở về chương trình chính.

Đọc số từ bàn phím

```
525 ; === Ham nhap so nguyen tu ban phim ===
526 read_num proc
527     mov ax, 0
528     mov bx, 0
529     read_loop:
530         mov ah, 1
531         int 21h           ; Nhan mot ky tu
532         cmp al, 13        ; Neu enter thi ket thuc
533         je read_done
534
535         cmp al, '9'       ; Neu khong phai chu so thi bao loi
536         jg read_again
537
538         cmp al, '0'       ; Neu khong phai chu so thi bao loi
539         jl read_again
540
541         sub al, '0'        ; Chuyen ky tu ve dang so
542         mov chuso, al
543         mov ax, bx
544         mul 1e1             ; Nhan 10 de dich trai so
545         mov bx, ax
546         add bl, chuso
547         mov ax, bx
548         jmp read_loop
549
550     read_done:
551         goto 2-10
552         printf blankstring
553         mov ax, bx
554         mov weight, ax
555         ret
556
557     read_again:
558         goto 2-10
559         printf Error
560         mov ax, 0
561
562     ret
563 read_num endp
```

Mã lệnh thực hiện như sau:

- **read_num:** Bắt đầu nhän đọc số từ bàn phím
 - **mov ax, 0:** Khởi tạo thanh ghi AX (sẽ dùng để tính toán và chứa kết quả cuối cùng) bằng 0.
 - **mov bx, 0:** Khởi tạo thanh ghi BX bằng 0. BX sẽ được sử dụng để tích lũy giá trị số trong quá trình đọc từng chữ số.
- **read_loop:** Bắt đầu vòng lặp đọc từng ký tự.
 - **mov ah, 1:** Chuẩn bị đọc một ký tự từ bàn phím
 - **int 21h:** Gọi ngắt DOS. Ký tự người dùng nhập được lưu vào thanh ghi AL.
 - **cmp al, 13:** So sánh ký tự vừa nhập (trong AL) với mã ASCII của phím Enter
 - **je read_done:** Nếu người dùng nhấn Enter (Jump if Equal), nhảy đến 'read_done' để kết thúc việc đọc.
 - **cmp al, '9':** So sánh ký tự trong AL với ký tự '9'.

- **jg read_again:** Nếu AL > '9' (Jump if Greater), ký tự không phải là chữ số. Nhảy đến 'read_again' để xử lý lỗi.
 - **cmp al, '0':** So sánh ký tự trong AL với ký tự '0'.
 - **jl read_again:** Nếu AL < '0' (Jump if Less), ký tự cũng không phải là chữ số. Nhảy đến 'read_again'.
 - **sub al, '0':** Chuyển đổi ký tự chữ số ASCII thành giá trị số thực. Kết quả số này nằm trong AL.
 - **mov chuso, al:** Lưu giá trị số vừa chuyển đổi vào biến tạm 'chuso'.
 - **mov ax, bx:** Nạp giá trị số đã tích lũy được từ các chữ số trước đó (từ BX) vào AX.
 - **mul 1e1:** Nhân AX với 1e1 (hằng số 10).
 - **mov bx, ax:** Cập nhật BX với giá trị vừa nhân.
 - **add bl, chuso:** Cộng chữ số mới.
 - **mov ax, bx:** Cập nhật lại AX bằng giá trị mới của BX.
 - **jmp read_loop:** Nhảy trở lại đầu vòng lặp 'read_loop' để đọc ký tự tiếp theo.
- **read_done:** Nhãn này được nhảy đến khi người dùng nhấn Enter.
- **goto 2,10:** Di chuyển con trỏ đến cột 2, dòng 10.
 - **printf blankstring:** In một chuỗi toàn khoảng trắng (từ biến 'blankstring').
 - **mov ax, bx:** Giá trị số nguyên cuối cùng được tích lũy trong BX. Gán giá trị này vào AX để trả về.
 - **mov weight, ax:** gán giá trị trong AX vào 'weight'.
 - **ret:** kết thúc thủ tục.
- **read_again:** Nhãn này được nhảy đến nếu người dùng nhập ký tự không phải số.
- **goto 2,10:** Di chuyển con trỏ đến cột 2, dòng 10.
 - **printf Error:** In ra thông báo lỗi "It seem that you made some mistakes?"
 - **mov ax, 0:** Đặt AX = 0. Đây là giá trị trả về mặc định nếu có lỗi nhập liệu
 - **ret:** kết thúc thủ tục.

Thủ tục in số từ bàn phím

```
566 print_num proc
567     mov cx, 0
568 Lappush:
569     mov dx, 0
570     div 1e1           ; Tach lay tung chu so
571     add dx, '0'       ; Chuyen thanh ky tu
572     push dx          ; Day vao ngan xep
573     inc cx
574     cmp ax, 0
575     jne Lappush
576 Hienthi:
577     pop dx           ; Lay chu so ra
578     mov ah, 2
579     int 21h
580     loop Hienthi
581
582 ret
print_num endp
```

Mã lệnh thực hiện như sau:

- **mov cx, 0:** Khởi tạo thanh ghi CX (bộ đếm số lượng chữ số) bằng 0.
- **Lappush:** Nhấn bắt đầu vòng lặp tách và đẩy chữ số vào stack.
 - **mov dx, 0:** Xóa thanh ghi DX.
 - **div 1e1:** Chia DX:AX cho 1e1 (hàng số 10). Thương số (số còn lại sau khi tách chữ số hàng đơn vị) được lưu vào AX. Số dư (chữ số hàng đơn vị vừa tách được) được lưu vào DX.
 - **add dx, '0':** Chuyển đổi giá trị số của chữ số dư (trong DX) thành ký tự ASCII.
 - **push dx:** Đẩy giá trị ký tự chữ số (trong DX) vào ngăn xếp. Ngăn xếp hoạt động theo cơ chế LIFO (Last In, First Out). Chữ số hàng đơn vị được đẩy vào trước, rồi đến hàng chục,...
 - **inc cx:** Tăng bộ đếm CX lên 1, đếm số lượng chữ số đã được đẩy vào stack.
 - **cmp ax, 0:** So sánh AX (thương số còn lại) với 0.
 - **jne Lappush:** Nếu AX khác 0 (Jump if Not Equal), có nghĩa là vẫn còn chữ số để tách. Nhảy trở lại 'Lappush'. Vòng lặp này tiếp tục cho đến khi AX bằng 0 (tất cả các chữ số đã được tách).
- **Hienthi:** Bắt đầu vòng lặp lấy chữ số từ stack và hiển thị.
 - **pop dx:** Lấy (pop) giá trị từ đỉnh ngăn xếp vào DX. Do LIFO, chữ số được lấy ra đầu tiên sẽ là chữ số cuối cùng được đẩy vào.
 - **mov ah, 2:** Chuẩn bị hiển thị một ký tự (chức năng 02h của ngắt 21h).
 - **int 21h:** Gọi ngắt DOS để hiển thị ký tự.
 - **loop Hienthi:** Vòng lặp này tiếp tục cho đến khi tất cả các chữ số đã được pop từ stack và hiển thị (khi CX = 0).
- **ret:** kết thúc thủ tục.

3. Kết quả thực hiện và kết luận

3.1. Kết quả thực hiện

Sau khi ấn run chương trình sẽ có giao diện như sau

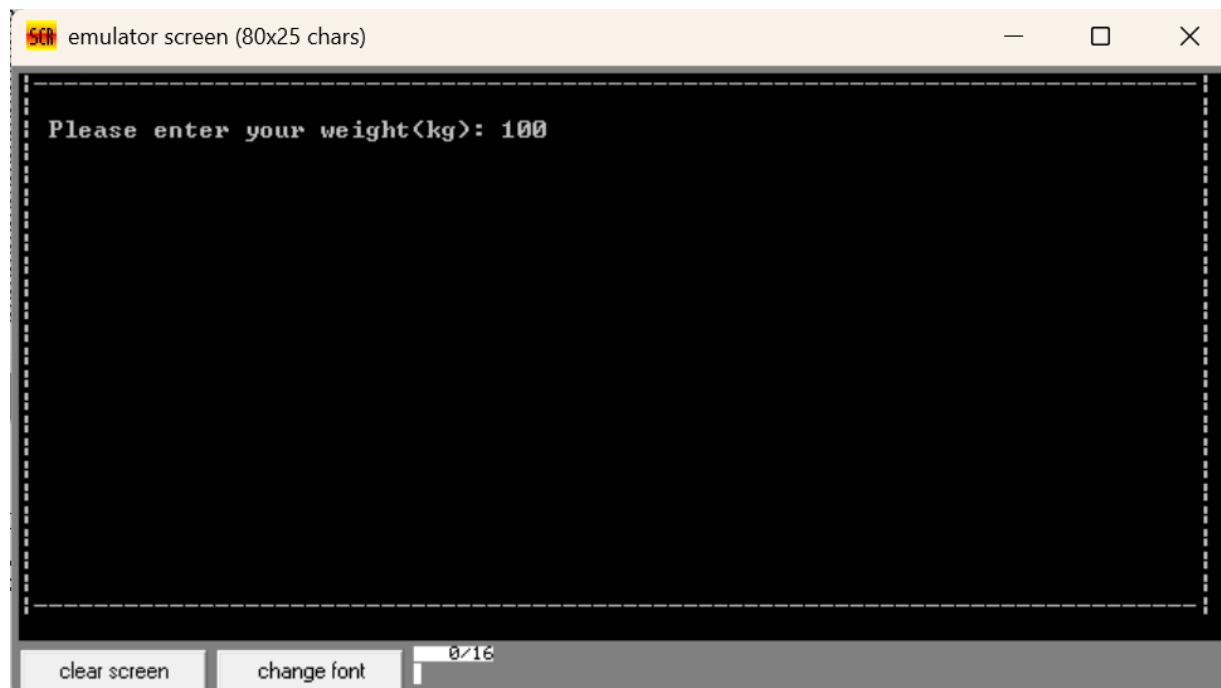


Sau khi ấn Enter chương trình sẽ đến phần menu

Người dùng có thể nhấn 1, 2, 3, 4 với các chức năng tương ứng.

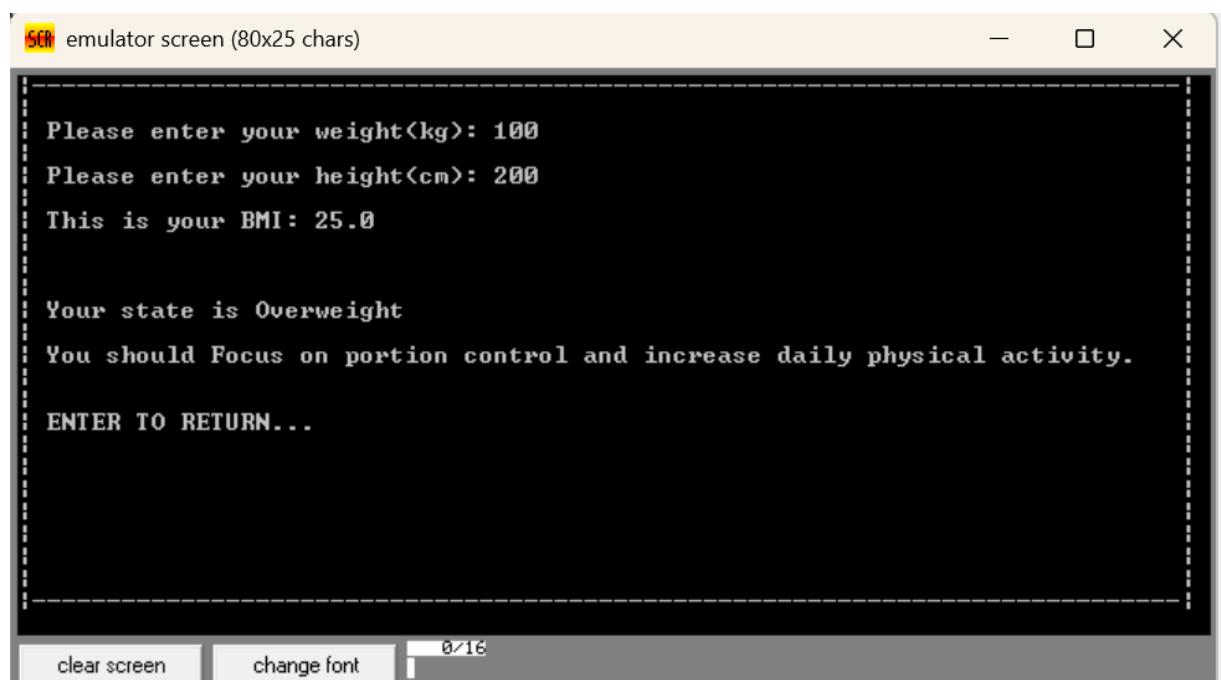


Nếu nhấn 1 chương trình sẽ yêu cầu người dùng nhập cân nặng và chiều cao



```
Please enter your weight(kg): 100
```

clear screen change font 0/16



```
Please enter your weight(kg): 100
Please enter your height(cm): 200
This is your BMI: 25.0

Your state is Overweight
You should Focus on portion control and increase daily physical activity.

ENTER TO RETURN...
```

clear screen change font 0/16

Sau khi nhập xong chỉ số BMI, tình trạng cân nặng và lời khuyên tương ứng sẽ được hiển thị trên màn hình

Người dùng nhấn Enter để quay trở lại menu

Trong trường hợp người dùng nhập sai (nhập ký tự khác 0-9) thì chương trình sẽ thông báo nhập sai, xoá bỏ phần đã nhập trước đó và yêu cầu người dùng nhập lại.

The screenshot shows a terminal window titled "SCR emulator screen (80x25 chars)". The window contains the following text:
Please enter your weight(kg):

It seem that you made some mistakes ?

At the bottom of the window, there is a toolbar with buttons for "clear screen", "change font", and a status bar showing "0/16".

Khi người dùng nhấn 2 ở giao diện menu thì bảng phân loại chỉ số BMI sẽ được hiển thị giúp người dùng có thể biết được tình trạng cân nặng của bản thân

The screenshot shows a terminal window titled "SCR emulator screen (80x25 chars)". The window displays a table of BMI categories:

BMI <kg/m>	Category
Below 16.0	Severely underweight
16.0 – 16.9	Moderately underweight
17.0 – 18.4	Mildly underweight
18.5 – 24.9	Normal weight
25.0 – 29.9	Overweight
30.0 – 34.9	Obesity Class I (Moderate)
35.0 – 39.9	Obesity Class II (Severe)
40.0 and above	Obesity Class III (Very severe)

ENTER TO RETURN...

At the bottom of the window, there is a toolbar with buttons for "clear screen", "change font", and a status bar showing "0/16".

Sau đó nhấn Enter để quay trở lại menu

Khi người dùng nhấn 3 ở giao diện Menu thì chương trình sẽ tiếp tục yêu cầu người dùng nhập vào chiều cao để tính cân nặng dựa trên chỉ số BMI lý tưởng.

```
Please enter your height(cm):  
Please enter your height(cm): 180  
Your ideal weight is: 71  
ENTER TO RETURN....
```

Sau khi nhập xong chiều cao, chương trình sẽ hiển thị cân nặng lý tưởng

Người dùng nhấn Enter để quay trở lại menu

Cuối cùng nếu nhấn 4 ở Menu để thoát chương trình thì sẽ có lời cảm ơn.



3.2. Kết luận

Kết quả kiểm thử cho thấy:

- Chương trình chạy ổn định
- Không phát sinh lỗi
- Các chức năng hoạt động đúng yêu cầu.

CHƯƠNG 3: TỔNG KẾT VÀ ĐÁNH GIÁ

Sau gần ba tháng làm việc nhóm, cả nhóm đã hoàn thành hai project trên nền tảng emu8086 với tinh thần nghiêm túc, chủ động và hợp tác tốt. Các thành viên đều có trách nhiệm với phần việc được giao, đồng thời sẵn sàng hỗ trợ nhau khi cần thiết.

Về chương trình:

Tic-Tac-Toe:

- **Ưu điểm:**
 - Giao diện rõ ràng, sử dụng bảng ma trận dễ nhìn, thuận tiện theo dõi lượt chơi.
 - Luồng chơi hợp lý, có xử lý thay đổi lượt, kiểm tra thắng/hòa, thông báo kết quả.
 - Có chức năng chơi lại và xử lý nhập sai giúp cải thiện trải nghiệm người dùng.
 - Ứng dụng hiệu quả macro như GOTOXY, CLEAR_SCREEN để tối ưu hiển thị.
- **Nhược điểm:**
 - Không có chế độ chơi với máy, chỉ hỗ trợ 2 người chơi thủ công.
 - Giao diện còn đơn giản, chưa có hiệu ứng hay âm thanh, màu sắc để tăng tính hấp dẫn.

BMI:

- **Ưu điểm:**
 - Tính toán chính xác chỉ số BMI từ chiều cao và cân nặng người dùng.
 - Phân loại rõ ràng; phân chia thành các mức.
 - Cung cấp lời khuyên và khuyến nghị sức khỏe, có gợi ý cân nặng lý tưởng.
 - Giao diện sạch, dễ sử dụng, hỗ trợ người dùng nhập và xuất thuận tiện.
- **Nhược điểm:**
 - Thiếu xác nhận đầu vào chật chẽ: dễ bị lỗi nếu nhập sai định dạng hoặc số âm.
 - Chỉ hỗ trợ đơn vị chiều cao theo cm và cân nặng theo kg, không linh hoạt với đơn vị khác.
 - Giao diện vẫn còn đơn giản, chưa có tùy chọn nâng cao như lưu kết quả hay chuyển đổi đơn vị.

Phương hướng phát triển của đề tài

Về cơ bản cả 2 đề tài đã đáp ứng được mục tiêu ban đầu mà nhóm chúng em đặt ra và hoàn thành được các chức năng cơ bản.

Nếu có thể phát triển, mở rộng thêm, chúng em mong muốn:

Đối với Game Tic Tac Toe:

- Thêm chế độ chơi với máy (AI) để người dùng có thể chơi một mình.
- Cải tiến giao diện: Bổ sung màu sắc, hiệu ứng rõ ràng phân biệt cho từng nước đi, hiệu ứng chiến thắng.
- Tăng mức độ thử thách: Cho phép người dùng chọn cấp độ khó.
- Lưu lịch sử chơi: Ghi lại số trận thắng/thua/hòa của người chơi qua các lượt.

Đối với Máy tính BMI:

- Cải thiện xử lý đầu vào: Tăng khả năng kiểm tra lỗi nhập, không cho nhập ký tự không hợp lệ, v.v.
- Tùy chọn đơn vị: Cho phép người dùng nhập chiều cao và cân nặng theo đơn vị mong muốn, sau đó tự động chuyển đổi.
- Bổ sung thông tin y tế: Giải thích thêm về ý nghĩa của BMI, nguy cơ sức khỏe kèm theo từng mức.

TÀI LIỆU THAM KHẢO

Các kiến thức nền tảng được tham khảo từ:

- [1] Giáp Văn Quân. (2016, November 24). *Bài 1. Các lệnh cơ bản trong EMU8086* [Video]. YouTube. <https://www.youtube.com/watch?v=joGVfLfGRk8>
- [2] Huy Init. (2021, October 27). *EXPIRED - Cách học + Tài liệu assembly* [Video]. YouTube. <https://www.youtube.com/watch?v=OLXTbVrE6A0>
- [3] Huy Init. (2021a, June 1). *EXPIRED - Game Tic Tac Toe bằng assembly (Nhóm 15 KMTT)* [Video]. YouTube. <https://www.youtube.com/watch?v=mgZ5goJkDWI>
- [4] World Health Organization. (1995). *Physical status : the use of and interpretation of anthropometry, report of a WHO expert committee.*
<https://iris.who.int/handle/10665/37003>