**BÁO CÁO KẾT QUẢ THẢO LUẬN NHÓM 2  
ĐỀ TÀI “TRAFFIC LIGHTS”**

1. **Tìm hiểu cách giao tiếp**

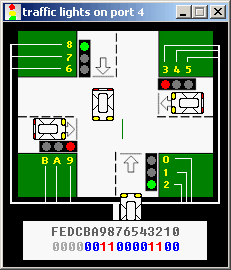
Ứng dụng đèn giao thông này được điều khiển bằng cách gửi dữ liệu vào cổng 4. Dữ liệu điều khiển là một mảng các giá trị. Mỗi giai đoạn nhất định, mảng dữ liệu đó sẽ được đọc và gửi vào cổng.

Có 12 bóng đèn: 4 đèn xanh, 4 đèn vàng và 4 đèn đỏ. Mỗi bóng được đặt tên theo thứ tự từ 0 đến B theo hệ thập lục phân.

Có thể thiết lập trạng thái của mỗi đèn bằng cách cách cài đặt các bit:

**1 – Đèn mở  
0 – Đèn tắt**

Chỉ có 12 bit thấp được dùng (từ 0 đến B), những bit cuối (từ C đến F) không được sử dụng.

Ví dụ:

MOV AX, 0000001100001100b

OUT 4, AX

Chuyển 0000001100001100b vào thanh ghi AX, từ thanh ghi AX xuất ra cổng số 4. Chỉ lấy 12 bit đầu (từ 0 đến B) tương ứng 12 bóng đèn. Nếu số 1 ở bóng đèn nào thì bóng đèn đó sáng.

**Hình 1**

1. **Đọc hiểu mã nguồn**

#start=Traffic\_Lights.exe#  ;Khởi động chương trình Traffic\_Lights.exe

name "traffic"

;Nếu dữ liệu đầu vào là hằng số (kiểu EQU) thì chuyển trực tiếp

mov ax, all\_red ;Di chuyển tình huống all\_red vào thanh ghi AX

out 4, ax ;Xuất dữ liệu trong thanh ghi AX vào cổng 4

;Nếu dữ liệu đầu vào là biến số (kiểu DW) thì chuyển qua thanh ghi SI

mov si, offset situation    ;Di chuyển offset situation vào SI (Source Index)

next:

mov ax, [si] ;Di chuyển vùng nhớ có địa chỉ chứa trong SI vào AX

out 4, ax ;Xuất AX vào cổng 4

;Tạm dừng 5 giây (5.000.000 micro giây => 4C4B40h)

mov cx, 4Ch ;Chuyển 2 bit cao vào thanh ghi CX

mov dx, 4B40h ;Chuyển 4 bit thấp vào thanh ghi DX

;Gọi hàm chờ BIOS với đầu vào là CX:DX (đơn vị micro giây)

mov ah, 86h

int 15h

;Chuyển sang tình huống tiếp theo

;Bằng cách cộng địa chỉ bộ nhớ (chứa trong SI) cho 2

add si, 2

cmp si, sit\_end ;Kiểm tra đã tới tình huống cuối cùng chưa

jb next ;Nếu chưa tới thì chạy tiếp

mov si, offset situation    ;Ngược lại thì quay về tình huống ban đầu

jmp next

;Danh sách các tình huống sử dụng trong chương trình

situation dw 0000\_0011\_0000\_1100b

s1 dw 0000\_0110\_1001\_1010b

s2 dw 0000\_1000\_0110\_0001b

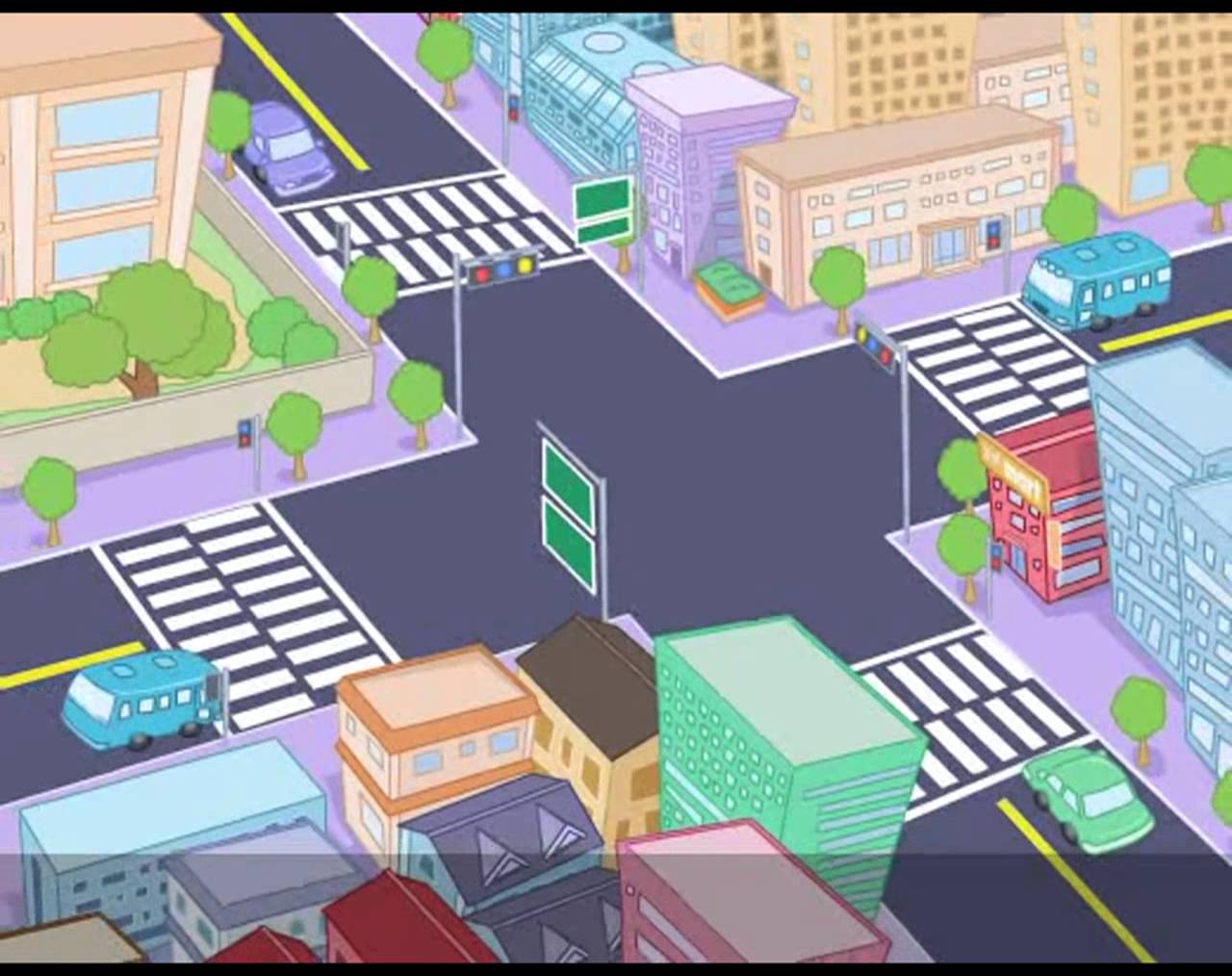
s3 dw 0000\_1000\_0110\_0001b

s4 dw 0000\_0100\_1101\_0011b

sit\_end = $

all\_red equ 0000\_0010\_0100\_1001b

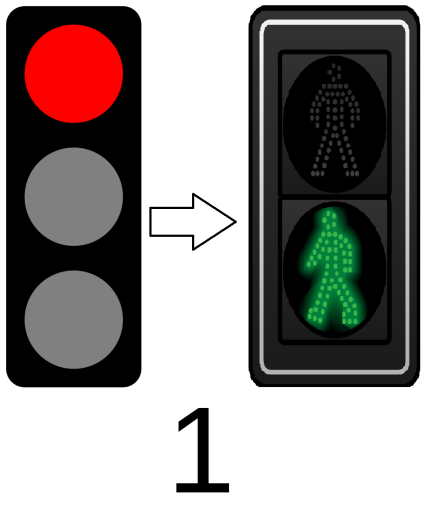
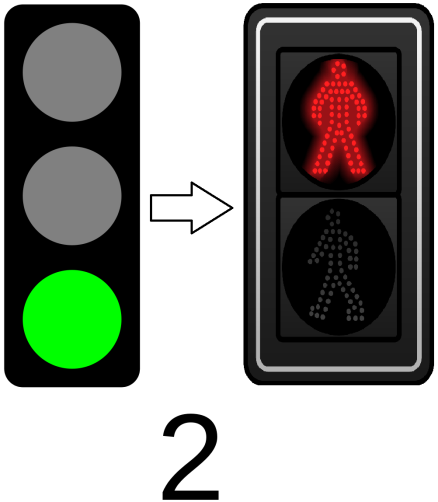
1. **Xây dựng kịch bản**
2. Kịch bản 1: Kết hợp với đèn dành cho người đi bộ ở ngã tư (Traffic Lights for Pedestrians)



Hình 2

1. Mô tả hoạt động

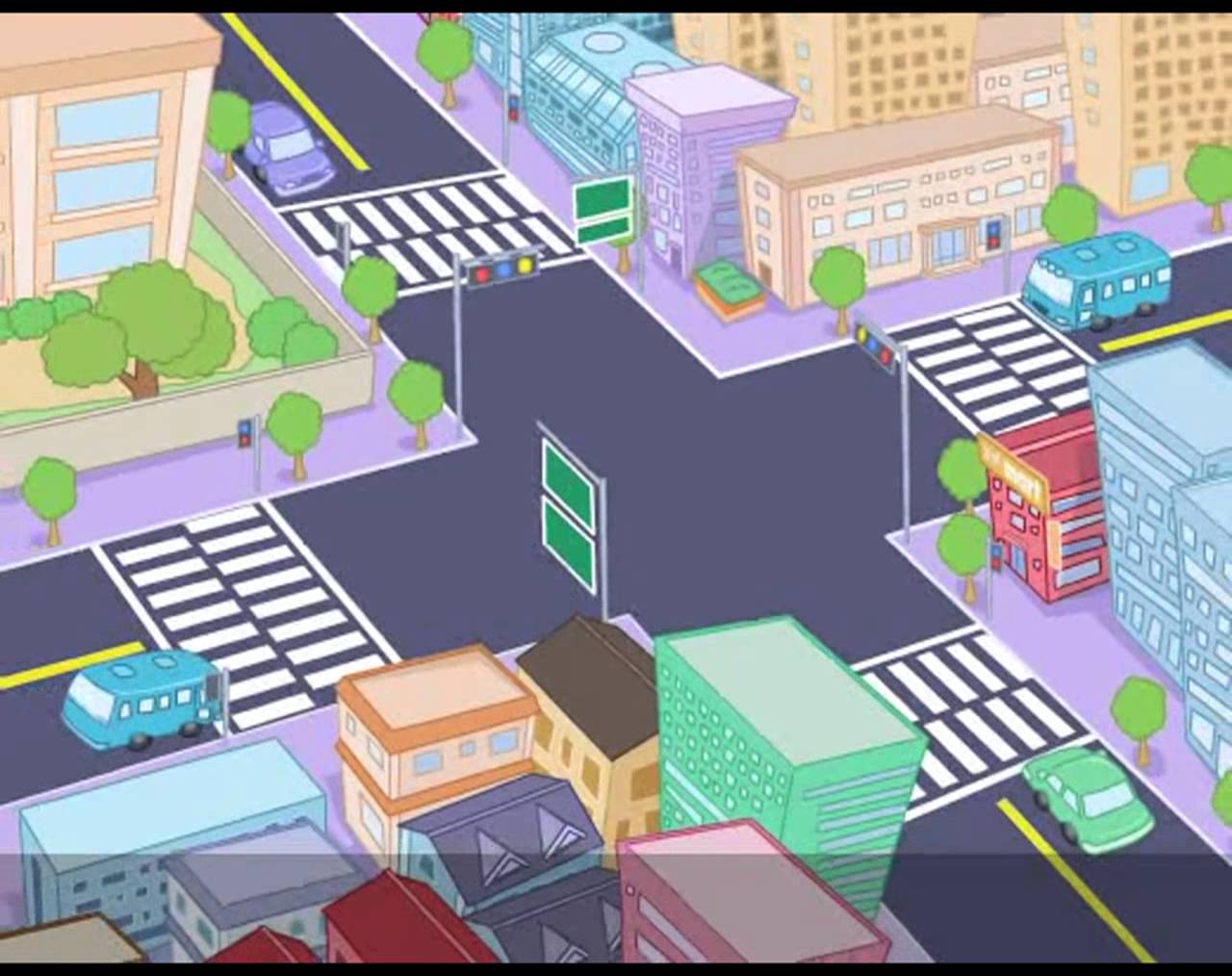
Nếu đèn giao thông ở một con đường chuyển sang màu **xanh** thì đèn dành cho người đi bộ ở đường đó chuyển sang màu **đỏ**, người đi bộ không được phép qua đường (hình 3.2). Ngược lại nếu đèn giao thông chuyển sang màu **đỏ** thì đèn dành cho người đi bộ chuyển sang màu **xanh**, người đi bộ sẽ được phép qua đường (hình 3.1).



**Hình 3.2**

**Hình 3.1**

Như vậy sẽ cần thêm 8 bóng đèn (4 đèn xanh, 4 đèn đỏ) từng cặp khác màu phối hợp với nhau ở 4 con đường (vị trí được khoanh tròn màu đỏ).



**D**

**C**

**B**

**A**

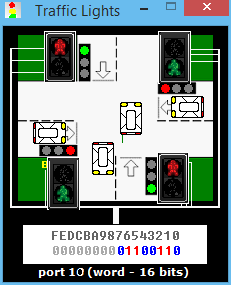
Hình 4

Ta đánh dấu các bóng đèn này lại:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| … | D | | C | | B | | A | | Vị trí |
| … | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Bit |
| ... | **Đỏ** | **Xanh** | **Đỏ** | **Xanh** | **Đỏ** | **Xanh** | **Đỏ** | **Xanh** | Màu |

Ta sẽ chỉ cần 8 bit đầu (từ 0 đến 7) và 8 bit còn lại (từ 8 đến F) sẽ không cần dùng đến.

Giả sử để điều khiển những bóng đèn này cần cổng 10.

Minh họa khi đèn giao thông đang ở trạng thái như hình 1, dữ liệu cần gửi cho đèn dành cho người đi bộ sẽ là thế này:

MOV AX, 0000000001100110b

OUT 10, AX

**B**

**C**

Quan sát hình bên, ta thấy vị trí A và C đèn giao thông đang màu **xanh**, đèn dành cho người đi bộ màu **đỏ**, người đi bộ không được phép băng qua 2 con đường này. Nhưng ngược lại, người đi bộ ở vị trí B và D thì được phép băng qua đường bởi vì các phương tiện đang chờ đèn **đỏ**, đèn dành cho người đi bộ đang màu **xanh**.

**A**

**D**

**Hình 5**

1. Thuật toán

#start=Traffic\_Lights.exe#

name "Traffic\_Lights\_for\_Pedestrians"

start:

nop

;Đèn giao thông đang ở trạng thái Xanh - Đỏ - Xanh - Đỏ

mov ax, 030Ch ;030Ch <=> 0000\_0011\_0000\_1100b

out 4, ax

;Đèn dành cho người đi bộ sẽ ở trạng thái Đỏ - Xanh - Đỏ - Xanh

xor ax, ax

mov ax, 66h ;66h <=> 0000\_0000\_0110\_0110b

out 10, ax

;Chờ chuyển tín hiệu (5s)

mov cx, 4Ch

mov dx, 4B40h

mov ah, 86h

int 15h

;Đèn giao thông đang ở trạng thái Đỏ - Xanh - Đỏ - Xanh

mov ax, 0861h ;0861h <=> 0000\_1000\_0110\_0001b

out 4, ax

;Đèn dành cho người đi bộ sẽ ở trạng thái Xanh - Đỏ - Xanh - Đỏ

xor ax, ax

mov ax, 99h ;99h <=> 0000\_0000\_1001\_1001b

out 10, ax

mov cx, 4Ch

mov dx, 4B40h

mov ah, 86h

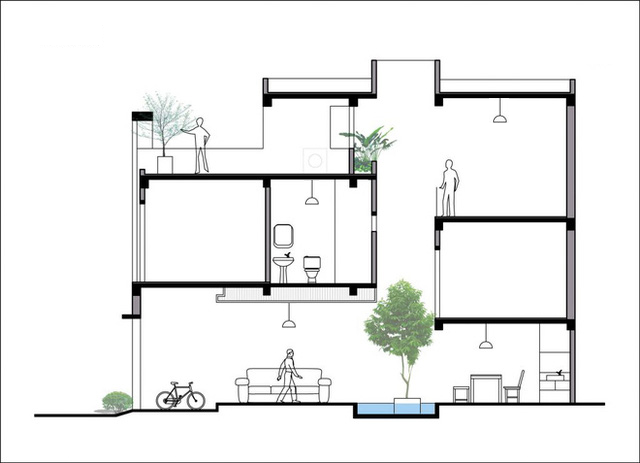
int 15h

;Quay lại từ đầu

jmp start

1. Kịch bản 2: Điều khiển việc bật/ tắt các đèn trong nhà
2. Mô tả hoạt động

Giả sử một căn nhà có tổng cộng là 4 bóng đèn lớn – dùng để thắp sáng cả căn phòng (không tính các bóng đèn nhỏ), 4 bóng đèn này phân bố ở 4 nơi trong căn nhà. Với mỗi bóng đèn đều có một công tắc điều khiển riêng. Bạn gặp một vấn đề, đó là mỗi lần muốn tắt một bóng đèn ở đâu thì phải lên tận nơi đó để tắt, rất phiền phức. Lúc này bạn cần có một bảng điều khiển có khả năng bật/ tắt các bóng đèn, như vậy không cần phải lên từng nơi để tắt.



**Hình 6**

**4**

**3**

**2**

**1**

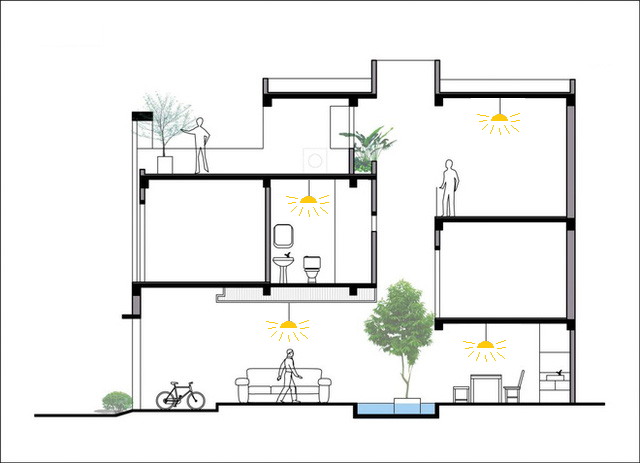
Để điều khiển 4 bóng đèn cần bảng điều khiển và cổng giao tiếp là 100.

Như vậy, chỉ có 4 bit đầu (từ 0 đến 3) được dùng để điều khiển đèn, còn 12 bit cuối (từ 4 đến F) không dùng đến.

Ta thiết lập các chức năng cho bảng điều khiển như sau:

**0 – Tất cả đều TẮT  
 1 – Bật/ Tắt đèn ở vị trí 1  
 2 – Bật/ Tắt đèn ở vị trí 2  
 3 – Bật/ Tắt đèn ở vị trí 3  
 4 – Bật/ Tắt đèn ở vị trí 4  
 5 – Tất cả đều BẬT**

Như vậy, nếu lần đầu chọn chức năng 1 thì đèn 1 sẽ **bật**, nhưng chọn lần thứ hai thì đèn 1 sẽ **tắt**. Nếu tất cả các đèn đều **bật** mà chọn chức năng 0 thì tất cả sẽ **tắt** (hình 6). Ngược lại, nếu chọn chức năng số 5 thì tất cả đều **bật** (hình 7).



**Hình 7**

1. Thuật toán

.MODEL SMALL

.STACK

.DATA

MESSAGE dw 13, 10, "Nhap so thu tu chuc nang : $"

;Cài đặt trạng thái BẬT của mỗi đèn

;Vị trí các bit 7654\_3210

all\_on equ 0000\_1111b

first\_light equ 0000\_0001b

second\_light equ 0000\_0010b

third\_light equ 0000\_0100b

fourth\_light equ 0000\_1000b

OUTPUT\_MESSAGE MACRO INPUT

MOV AH, 9

LEA DX, INPUT

INT 21H

ENDM

.CODE

MOV AX, @DATA

MOV DS, AX

MAIN:

OUTPUT\_MESSAGE MESSAGE

MOV AH, 1

INT 21H

;Nếu ký tự nhập vào nhỏ hơn 0 hoặc lớn hơn 5 thì đó là nhập sai

CMP AL, '0'

JB ERROR

CMP AL, '5'

JA ERROR

;Ngược lại thì thực hiện chức năng đó

CMP AL,'0'

JE TURN\_OFF\_ALL

CMP AL, '1'

JE FIRST

CMP AL, '2'

JE SECOND

CMP AL, '3'

JE THIRD

CMP AL, '4'

JE FOURTH

JMP TURN\_ON\_ALL

;Phải dùng tạm thanh ghi BX rồi mới chuyển vào AX là vì

;Khi lặp lại, giá trị thanh ghi AX bị thay đổi do dùng hàm 1 ngắt 21h

;Không còn mang giá trị cũ (Đèn nào Bật, Đèn nào Tắt) nữa

;Như vậy, BX sẽ giữ chức năng lưu trạng thái của các đèn

TURN\_OFF\_ALL:

XOR BX, BX

JMP \_LOOP

TURN\_ON\_ALL:

MOV BX, all\_on

JMP \_LOOP

FIRST:

XOR BX, first\_light

JMP \_LOOP

SECOND:

XOR BX, second\_light

JMP \_LOOP

THIRD:

XOR BX, third\_light

JMP \_LOOP

FOURTH:

XOR BX, fourth\_light

\_LOOP:

MOV AX, BX

OUT 100, AX

ERROR:

JMP MAIN

END