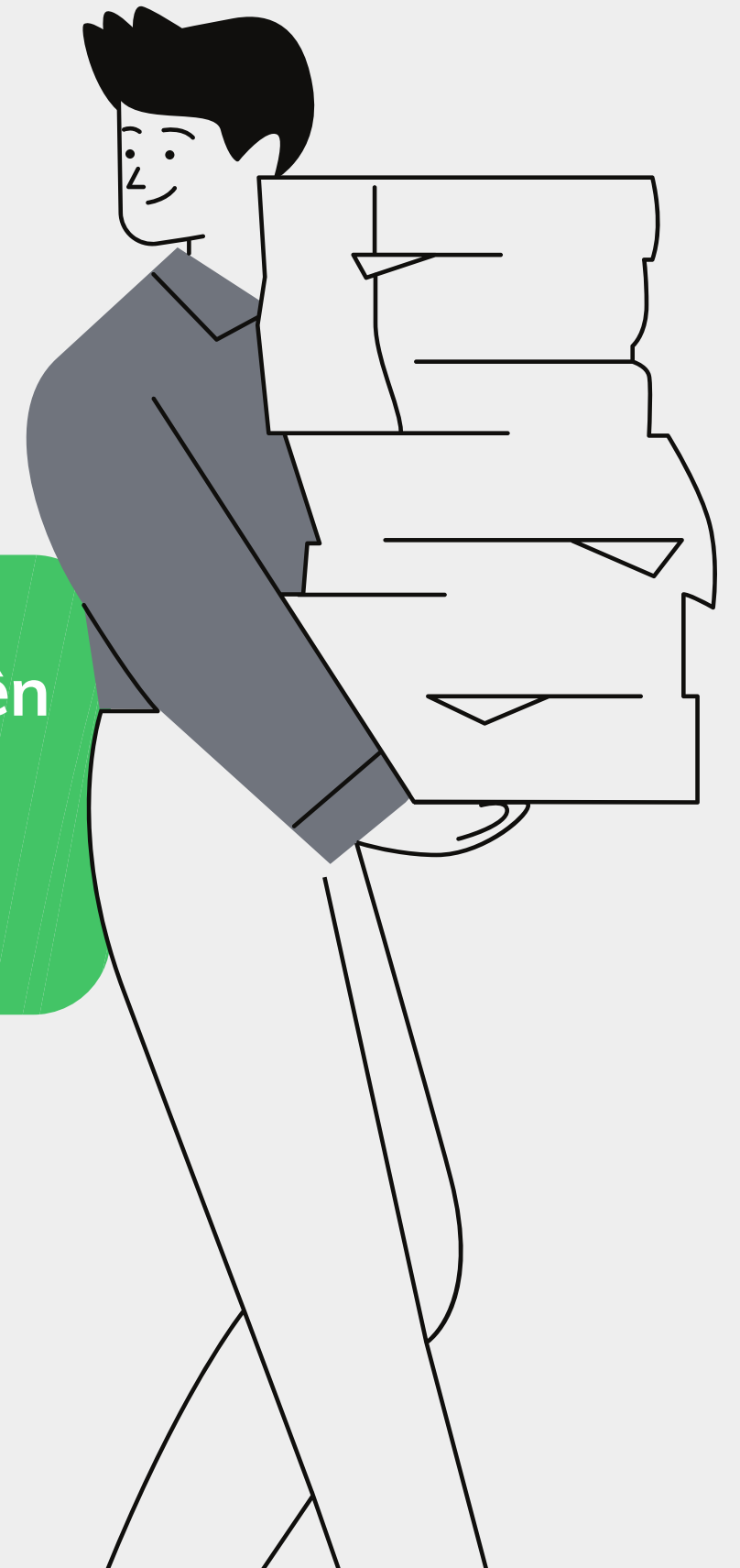


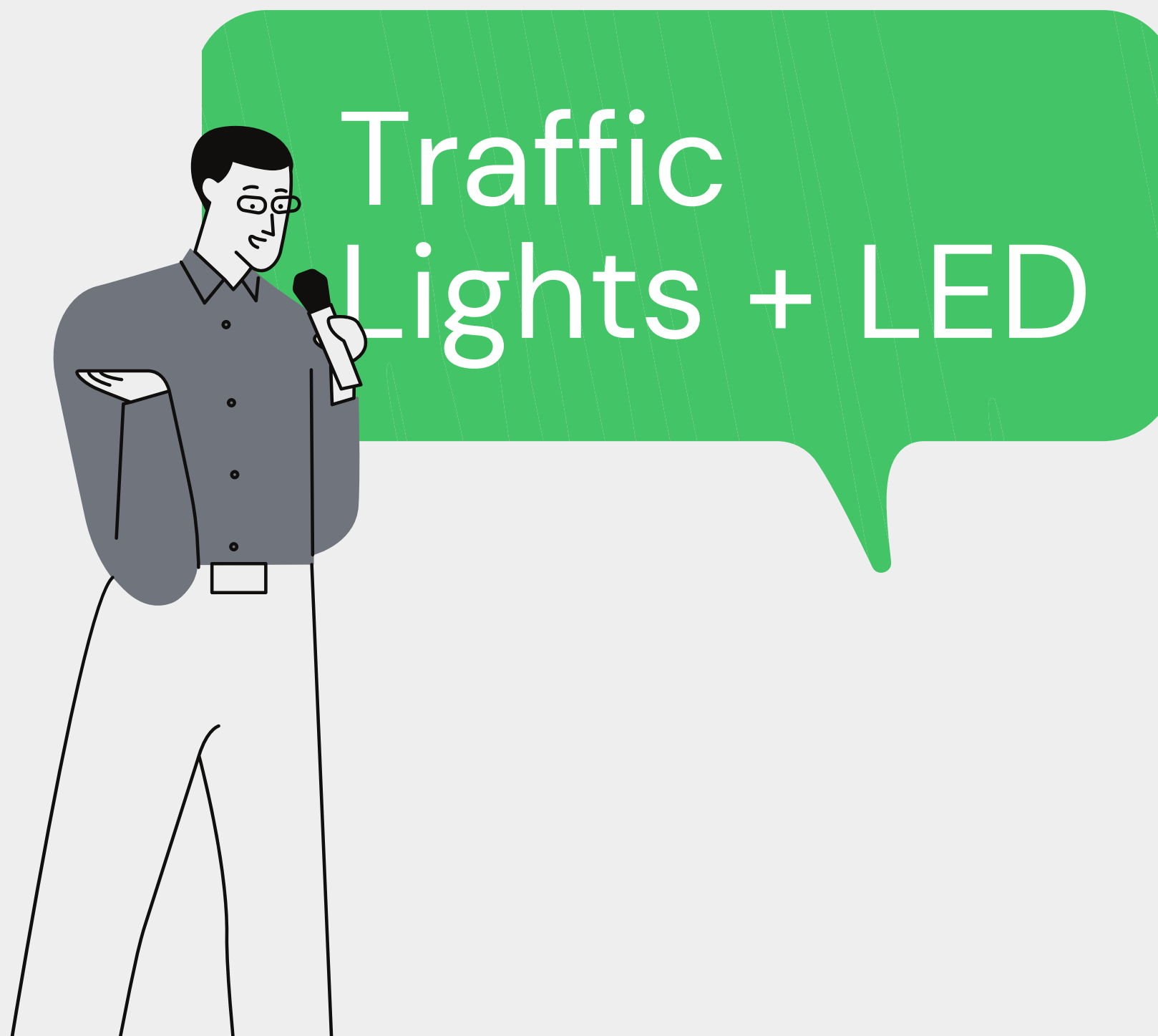
Kiến trúc máy tính

Xây dựng một mô phỏng trên
phần mềm emu8086 có kết
nối đến một số ngoại vi.

Team 11

1. Vương Huy Long
2. Lê Trường Giang
3. Nguyễn Hoàng Dương
4. Vũ Minh Trí





1 Nguyên lý hoạt động

2 Tình huống thực tế

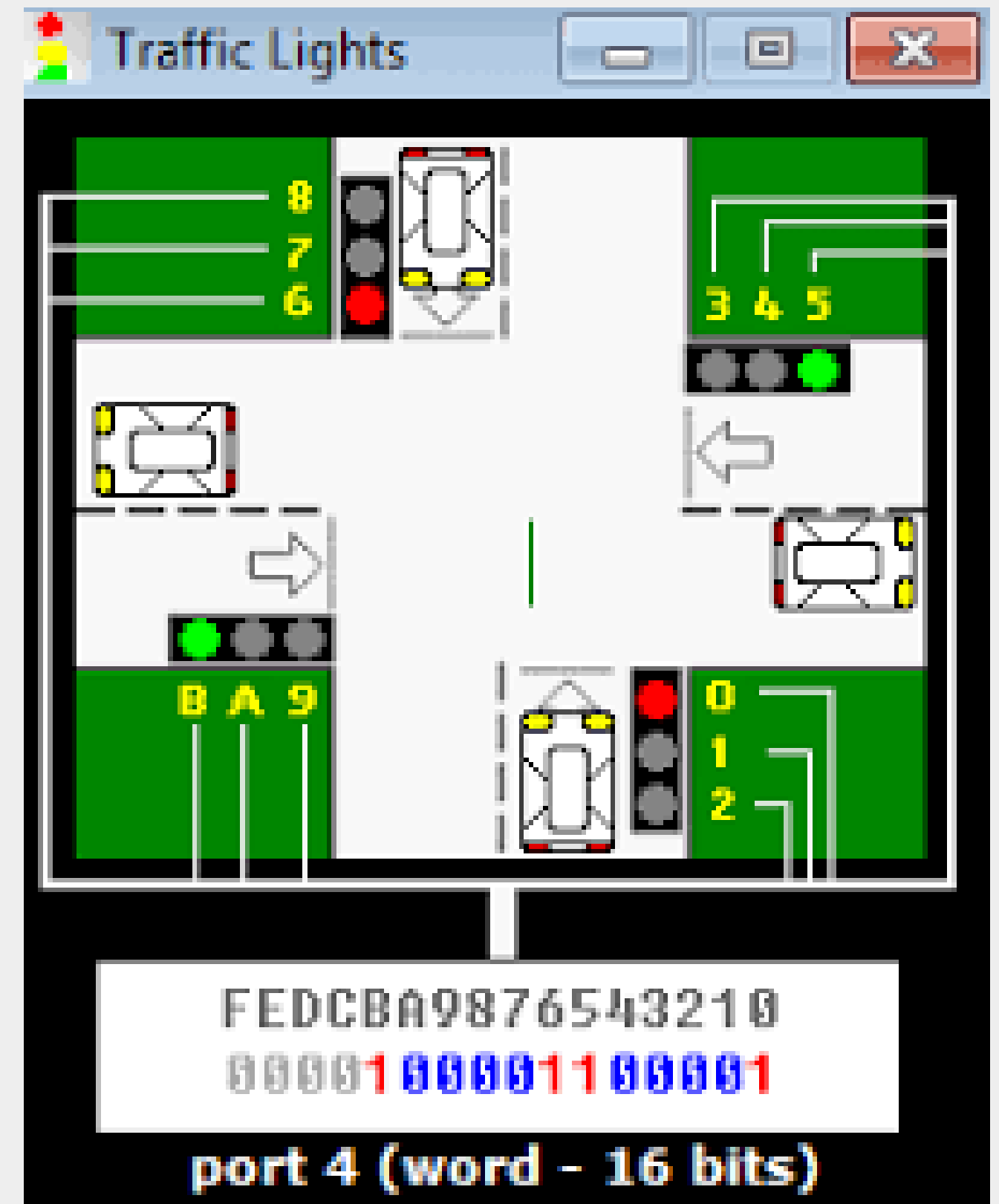
3 Ý tưởng bài toán

4 Code emu 8086

1

Nguyên lí hoạt động

- Hoạt động thế nào?
- Input/Output?



Traffic Lights

Hoạt động thế nào?



Đèn đỏ
0, 3, 6, 9

Dừng

(30s)



Đèn vàng
1, 5, 7, A

Dừng

(3s)



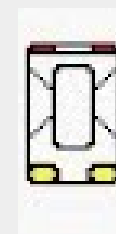
Đèn xanh
2, 5, 8, B

Di chuyển

(30s)

Note:

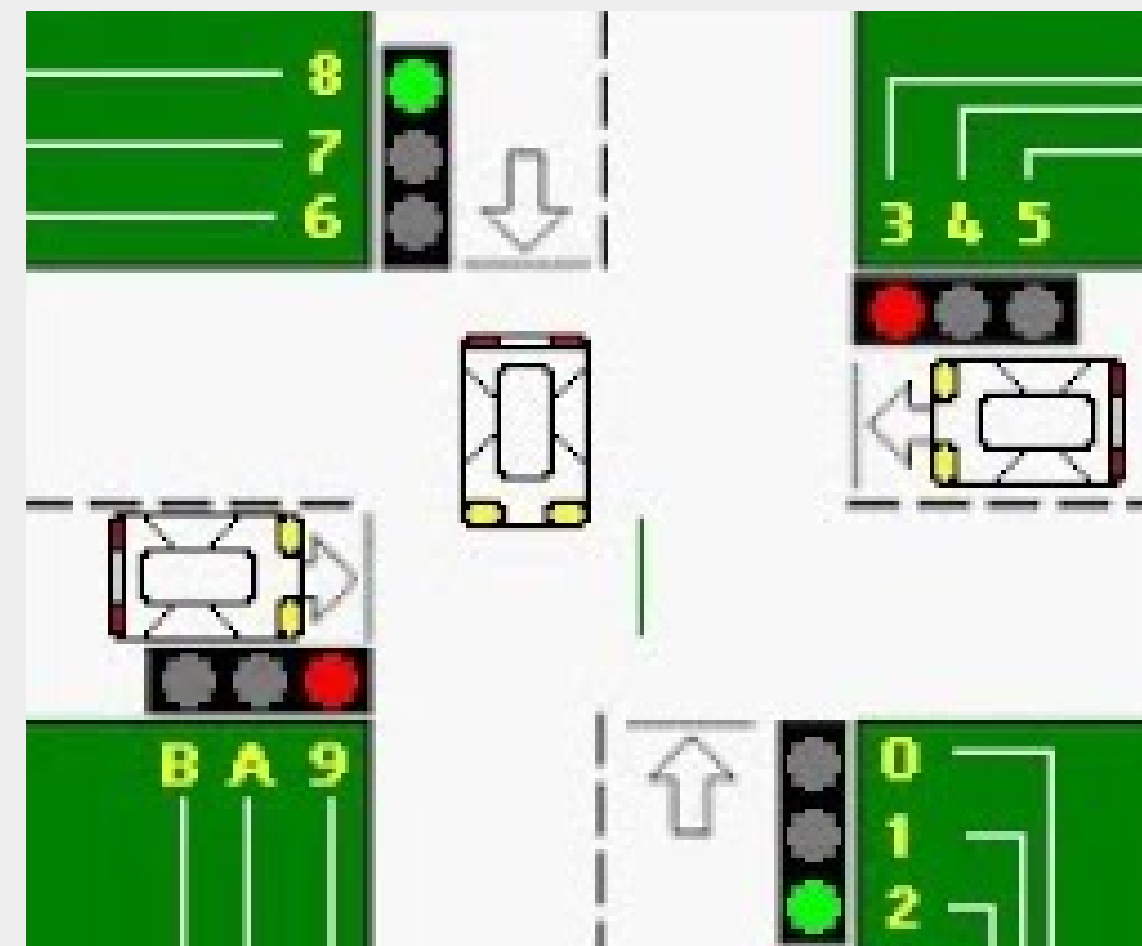
- Ứng dụng chỉ hỗ trợ xe di chuyển trong làn của mình, không được rẽ



Thiết bị
di chuyển



Đèn giao
thông



Traffic Lights

Hoạt động thế nào?

1 là đèn bật, 0 là đèn tắt

số đèn sẽ tương ứng với vị trí của nó trong chuỗi bit tín hiệu

Các chuỗi này được biểu diễn bằng kiểu dữ liệu WORD (16 bit), được chuyển vào cổng 4 của ứng dụng, có dạng như sau :

FEDC_BA98_7654_3210

- Trong đó : 4 bit FEDC ko được sử dụng.
- 4 bit B,8,5,2 tương ứng đèn **XANH**
- 4 bit 9,6,3,0 tương ứng đèn **ĐỎ**
- 4 bit A,7,4,1 tương ứng đèn **VÀNG**



Đèn đỏ
0, 3, 6, 9



Đèn vàng
1, 5, 7, A



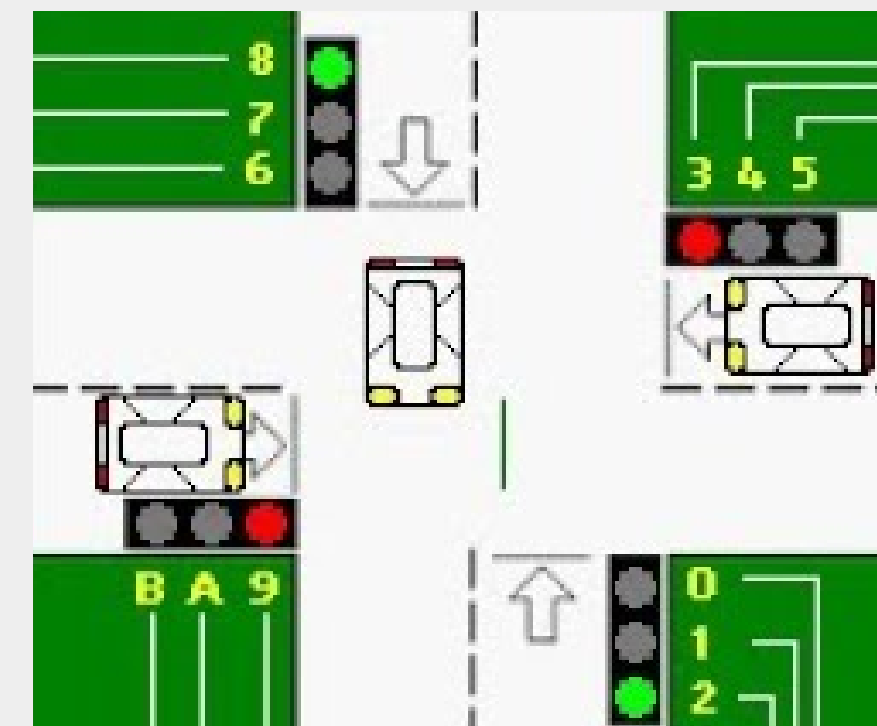
Đèn xanh
2, 5, 8, B



Thiết bị
di chuyển



Đèn giao
thông



Traffic Lights

Hoạt động thế nào?

1 là đèn bật, 0 là đèn tắt

số đèn sẽ tương ứng với vị trí của nó trong chuỗi bit tín hiệu

Các chuỗi này được biểu diễn bằng kiểu dữ liệu WORD (16 bit), được chuyển vào cổng 4 của ứng dụng, có dạng như sau :

FEDC_BA98_7654_3210

- Trong đó : 4 bit FEDC ko được sử dụng.
- 4 bit B,8,5,2 tương ứng đèn xanh
- 4 bit 9,6,3,0 tương ứng đèn đỏ
- 4 bit A,7,4,1 tương ứng đèn vàng

Ví dụ : 0000 0011 0000 1100 là chuỗi bit thể hiện 4 đèn như hình.



Đèn đỏ
0, 3, 6, 9



Đèn vàng
1, 5, 7, A



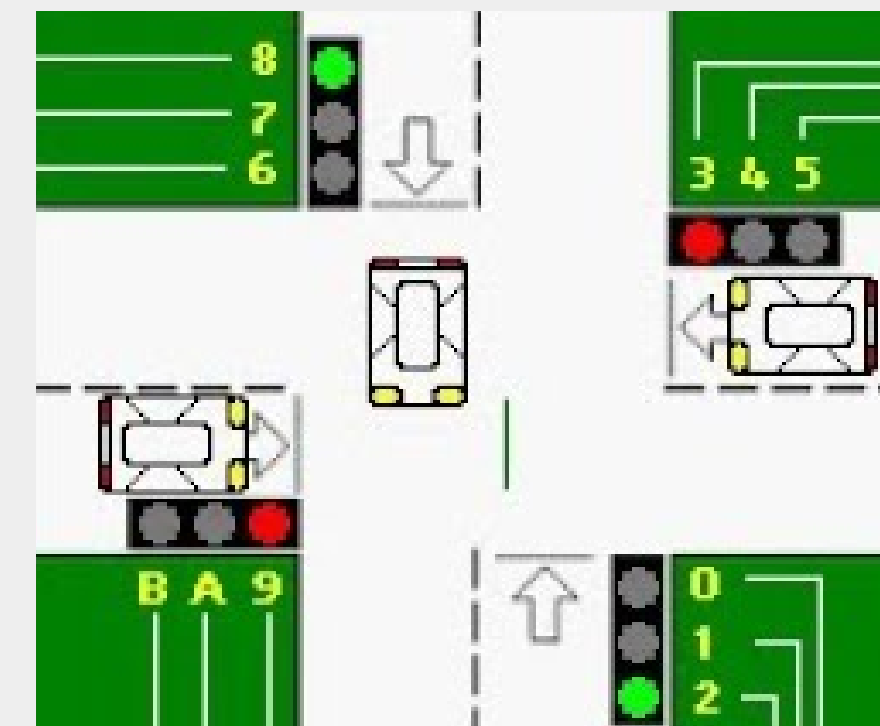
Đèn xanh
2, 5, 8, B



Thiết bị
di chuyển



Đèn giao
thông



tình huống



- Trong quá trình giao thông có thể sẽ phát sinh sự cố ngoài ý muốn khiến giao thông bị cản trở.

- Ta cần xử lý các sự cố sao cho không xảy ra mâu thuẫn giữa các tuyến đường.



2

tình huống



- Trong quá trình giao thông có thể sẽ phát sinh sự cố ngoài ý muốn khiến giao thông bị cản trở.

- Ta cần xử lý các sự cố sao cho không xảy ra mâu thuẫn giữa các tuyến đường.

3

ý tưởng bài toán

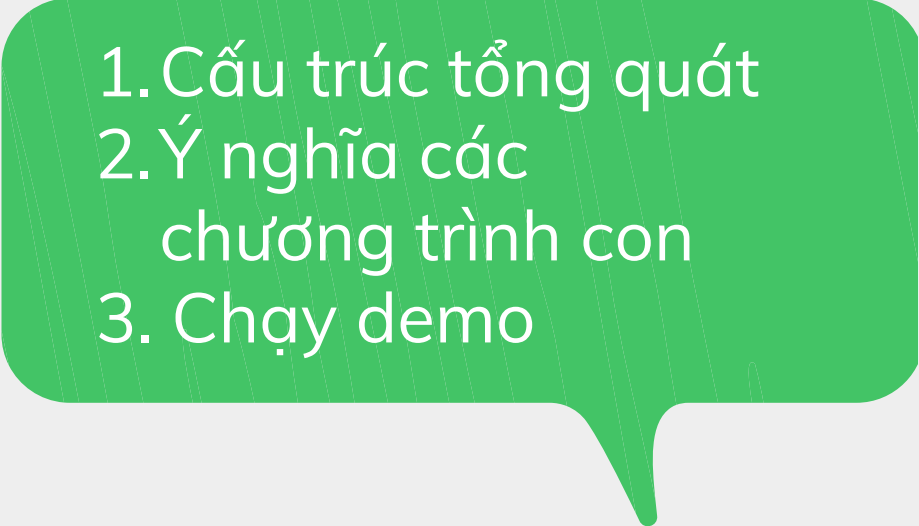
- Các tuyến đường có sự cố sẽ bật đèn đỏ để tạm dừng, đến khi sự cố được giải quyết

- Các tuyến đường còn lại chuyển sang trạng thái đèn kế tiếp (xanh - vàng - đỏ)

- Cần chú ý không để xảy ra mâu thuẫn 2 trục đường (không cùng xanh/đỏ)



4. Code

- 
1. Cấu trúc tổng quát
 2. Ý nghĩa các chương trình con
 3. Chạy demo

4. Code

4.1 Main

```
Ondinh:
    Call SINH_SUCO    ; tao suco.
    ;Kiem tra co su co khong?
    cmp SUCO,0
    je LED:           |

    ;Thong bao len CONSOLE neu co
    Call THONGBAO

LED:
    CALL _PORT4
    CALL WAITT

    CALL SUCO_KETTHUC

    CALL WAITT

Vonglap:
    jmp Ondinh
```

- Sinh sự cố → Thức tạo giá trị cho biến sự cố.
- Thông báo nếu có sự cố.
- Đưa trạng thái hiện tại sang trạng thái mới.
- Hiển thị đèn đếm ngược.
- Kiểm tra sự cố đã kết thúc chưa?

4. Code

4.2 Các chương trình con



1

SINH_SUCO

sinh một sự cố ngẫu
nhiên với xác suất x%
(ví dụ 10%)

2

THONGBAO

In thông báo nếu có sự cố/sự cố
kết thúc

3

_PORT4

Chuyển trạng thái hiện tại sang
trạng thái mới bằng chương
trình con BATDEN

4

BATDEN

- Kiểm tra và giải quyết nếu trạng thái hiện tại có xung đột.
- Ví dụ nếu 2 tuyến vuông góc đều có đèn đỏ, vậy ta phải chuyển 1 tuyến thành xanh.

5

WAITT

Hiển thị thời gian lên đèn LED:

- Đèn xanh/đỏ: đếm 30s
- Đèn vàng: 3s

6

SUCO_KETTHUC

- Nếu sự cố kết thúc thì thông báo
- Chuyển trạng thái kế tiếp

4. Code

4.2 Các chương trình con

1

SINH_SUCO

- Bước 1: Sinh giá trị ngẫu nhiên x (x thuộc tập D(0,1))
- Bước 2: Nếu x == 1. Sinh giá trị cho SUCO bằng cách sinh giá trị y cho địa chỉ SI (khởi tạo SI = offset SUCO0) (y thuộc tập T(0 → 9)).

$$SI = SI + y * 2$$

(Do SUCO là kiểu dữ liệu DW do vậy khoảng cách giữa 2 ô nhớ là 2)

- Bước 3: Chuyển giá trị thuộc ô nhớ SI vào biến SUCO.

```
SINH_SUCO proc

    mov ax,0
    mov ah,0
    int 1ah

    mov al,dl
    mov dl,0
    mov bl,10
    div bl
    cmp ah,1
    je sinhSuco
    jmp nothing
sinhSuco:
    mov ah,0
    int 1ah

    mov al,dl
    mov dl,0
    mov bl,10
    div bl
    mov al,ah
    mov bl,2
    mul bl

    mov si, offset SUCO0
    add si,ax
    mov ax,[si]
    mov SUCO,ax
nothing:
ret
SINH_SUCO ENDP
```



4. Code

4.2 Các chương trình con

2 THONGBAO

In thông báo nếu có sự cố/sự cố kết thúc

THONGBAO **proc**

```
mov ax,SUCO
cmp ax,0
je KOSUCO
mov dx,offset nhap
mov ah,9
int 21h
```

```
mov ah,2h      ;xuong dong
mov dl,0ah
int 21h
```

```
mov ah,2h      ;xuong dong
mov dl,0dh
int 21h
```

```
jmp EndTB
KOSUCO:
mov dx,offset nhap1
mov ah,9
int 21h
```

```
mov ah,2h      ;xuong dong
mov dl,0ah
int 21h
```

```
mov ah,2h      ;xuong dong
mov dl,0dh
int 21h
```

```
EndTB:
ret
THONGBAO ENDP
```



4. Code

4.2 Các chương trình con

3 _PORT4

Chuyển trạng thái hiện tại sang trạng thái mới bằng hàm BATDEN

```
_PORT4 proc  
    begin:  
  
        CALL BATDEN  
  
        mov ax,II  
        out 4,ax  
ret  
_PORT4 ENDP
```



4. Code

4.2 Các chương trình con

4

BATDEN

- Kiểm tra và giải quyết nếu trạng thái hiện tại có xung đột.
- Ví dụ nếu 2 tuyến vuông góc đều có đèn đỏ, vậy ta phải chuyển 1 tuyến thành xanh (ví dụ đèn B5 như hình bên)

BATDEN proc

```
mov ax, IT
and ax, Do_39
cmp ax, 0
je tuyen1

mov ax, IT
and ax, Do_06
cmp ax, 0
je tuyen1

mov ax, IT
or ax, Xanh_B5
and ax, turnGreen5
and ax, turnGreenB
mov IT, ax
```



4. Code

4.2 Các chương trình con

4 BATDEN

Xét tuyến đường 1 (có đèn 0, 1, 2):
(các tuyến khác tương tự)

```
tuyen1:
    ; Vang_1, Xang_2, Do_0, turnGREEN2, turnRED0, turnY1

    ; Kiem tra co su co hay k
    mov ax, SUCO
    and ax, Do_0
    cmp ax, 0
    jne cosuco1
DENXANH1:
    mov ax, IT
    and ax, Xanh_2
    cmp ax, 0
    je DENVANG1
    mov ax, Vang_1 ; neu co bat den vang tuong ung
    or IT, ax
    mov ax, turnY1
    and IT, ax
    jmp ketthuc_batden_tuyen1
DENVANG1:
    mov ax, IT
    and ax, Vang_1
    cmp ax, 0
    je DENDOX1

cosuco1:
    mov ax, Do_0 ; neu co bat den vang tuong ung
    or IT, ax
    mov ax, turnRED0
    and IT, ax
    jmp ketthuc_batden_tuyen1
DENDOX1:
    mov ax, Xanh_2 ; neu co bat den vang tuong ung
    or IT, ax
    mov ax, turnGREEN2
    and IT, ax
ketthuc_batden_tuyen1:
```



4. Code

4.2 Các chương trình con

5

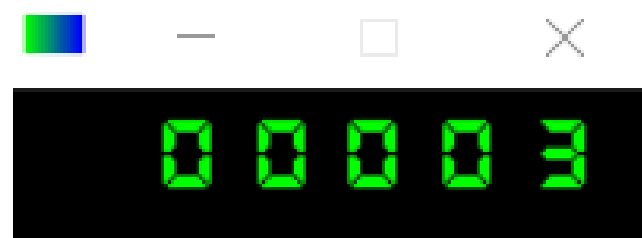
WAITT

Hiển thị thời gian lên đèn LED:

- Đèn xanh/đỏ: đếm 30s
- Đèn vàng: 3s

- Ta sử dụng một chuỗi CHECK để kiểm tra trạng thái đèn.
CHECK DW 0000 0100 1001 0001 (full đèn vàng).

- CHECK AND TT = 0000 0i00 i00i 000i
Kết quả != 0 thì LED đếm ngược 3 giây.
Kết quả == 0 thì LED đếm ngược 30 giây



```
CHECK DW 0000_0100_1001_0001b
WAITT PROC
    mov ax, CHECK
    AND ax, IT
    cmp ax, 0
    je 30s
```

```
3s:
    CALL WAIT_3s
    jmp Ketthuc
```

```
30s:
    CALL WAIT_30s
```

```
Ketthuc:
ret
WAITT ENDP
```

```
WAIT_3s PROC
mov ax, 3
push ax
x2:
    pop ax
    out 199, ax
    sub ax, 1
    cmp ax, 0
    je Ket2
    push ax
    mov cx, 0Ch
    mov dx, 4B40h
    mov ah, 86h
    int 15h
    jmp x2
Ket2:
ret
WAIT_3s ENDP
```

4. Code

4.2 Các chương trình con

6

SUCO_KETTHUC

- Biến ENSUCO nhận giá trị 0 (hết SUCO) hoặc 1(còn) ngẫu nhiên.
- Nếu hết Sự cố thì ta chuyển SUCO = 0, nếu ko giữ yên sự cố. Nếu sự cố kết thúc thông báo lên màn hình.
- Sau đó sử dụng hàm _PORT 4 để chuyển trạng thái.

```
SUCO_KETTHUC PROC
    mov ah,0
    int 1ah

    mov al,dl
    mov dl,0
    mov bl,2
    div bl
    cmp ah,1
    je tieptuc
    mov SUCO,0000_0000_0000b
    CALL THONGBAO
    tieptuc:

    CALL BAIDEN

Kettthuc_SUCO:
    mov ax,11
    out 4,ax
    ret
SUCO_KETTHUC ENDP
```



Thank you!

Cảm ơn thầy và các
bạn đã lắng nghe,
chúc Việt Nam mau
chiến thắng đại dịch!!

Let's Start!

Demo EMU 8086

