Traffic light

```
asm
org 100h
#start=Traffic_Lights.exe#
mov ax, all_red
out 4, ax
mov si, offset start
next:
   mov ax, [si]
   out 4, ax
   ; Wait for approximately 8 seconds delay
   mov cx, 70h       ; Updated delay for ~8 seconds
   mov dx, 0000h
   mov ah, 86h
   int 15h
   add si, 2         ; Move to next pattern
   cmp si, s_end
   jb  next
   mov si, offset start
   jmp next
;    FEDC_BA98_7654_3210
start dw 0000_0010_0100_1001b
s1    dw 0000_0010_0100_1010b
s2    dw 0000_0010_0100_1100b
s3    dw 0000_0010_0101_0001b
s4    dw 0000_0010_0110_0001b
s5    dw 0000_0010_1000_1001b
s6    dw 0000_0011_0000_1001b
s7    dw 0000_0100_0100_1001b
s8    dw 0000_1000_0100_1001b
s9    dw 0000_0011_0000_1100b
s10   dw 0000_0010_0100_1001b
s11   dw 0000_0110_1001_1010b
s12   dw 0000_1000_0110_0001b
```

```
s13   dw 0000_1000_0110_0001b
s14   dw 0000_0100_1101_0011b
s_end = $
all_red equ 0000_0010_0100_1001b
ret
```

stepper motor

```
org 100h
mov dx, 07h         ; motor connected to port 0
mov cx, 4           ; 4 step sequence
lea si, steps       ; point to step sequence table
rotate:
   mov al, [si]     ; get step pattern
   out dx, al       ; send to port
   call delay       ; wait
   inc si           ; next step
   loop rotate      ; repeat 4 steps
jmp start           ; keep rotating
start:
   mov cx, 4
   lea si, steps ;lea si, stepsanti for anticlockwise
   jmp rotate
; Step sequence for clockwise rotation
stepsanti db 09h,0Ch,06h,03h
steps db 03h,06h,0Ch,09h
delay proc
   mov bx, 000fh
d1: dec bx
   jnz d1
   ret
delay end
```

EMU8086 experiment — BCD Addition

ASSEMBLY LANGUAGE PROGRAM

```
org 100h
; Program for BCD Addition
.data
num1 db 25h       ; First BCD number = 25
num2 db 37h       ; Second BCD number = 37
result db ?       ; To store result
.code
start:
   mov ax, @data
   mov ds, ax
   mov al, num1    ; Load first BCD number
   add al, num2    ; Add second BCD number (BCD addition)
   daa             ; Decimal Adjust for Addition (corrects result to BCD format)
   mov result, al  ; Store result in memory
 ; Display Result
   mov ah, 09h
   lea dx, msg
   int 21h
   mov al, result
   call display_bcd ; Show BCD value on screen
   mov ah, 4Ch
   int 21h
display_bcd proc
   push ax
   mov ah, 0
   mov bl, al
   mov al, bl
   ; Display upper nibble
   mov ah, 02h
   mov dl, al
   and dl, 0F0h
   shr dl, 4
```

```
    add dl, 30h
    int 21h
    ; Display lower nibble
    mov dl, al
    and dl, 0Fh
    add dl, 30h
    int 21h
    pop ax
    ret
display_bcd endp
msg db 'Result (BCD Addition) = $'
end start
```

palindrome

```
ASSEMBLY LANGUAGE PROGRAM
org 100h
.data
msg1 db 'Enter a string: $'
msg2 db 0Dh,0Ah,'It is a Palindrome.$'
msg3 db 0Dh,0Ah,'It is NOT a Palindrome.$'
string db 20,0,20 dup('$')    ; Buffer for input string

.code
start:
   mov ax, @data
   mov ds, ax
   ; --- Prompt user to enter string ---
   mov ah, 09h
   lea dx, msg1
   int 21h
   mov ah, 0Ah
   lea dx, string
   int 21h
   ; SI points to first character
   lea si, string+2
   ; CX = string length
   mov cl, [string+1]
   mov ch, 0
   ; DI points to last character
   lea di, string+2
   add di, cx
   dec di
check_loop:
   cmp si, di
```

```
    jge palindrome      ; crossed or equal → all matched

    mov al, [si]
    mov bl, [di]
    cmp al, bl
    jne not_palindrome  ; mismatch → not palindrome
    inc si
    dec di
    jmp check_loop
palindrome:
    mov ah, 09h
    lea dx, msg2
    int 21h
    jmp end_prog
not_palindrome:
    mov ah, 09h
    lea dx, msg3
    int 21h
end_prog:
    mov ah, 4Ch
    int 21h
end start
```