Bölüm 9: Aygıtların Kontrolü

Mikroişlemciler

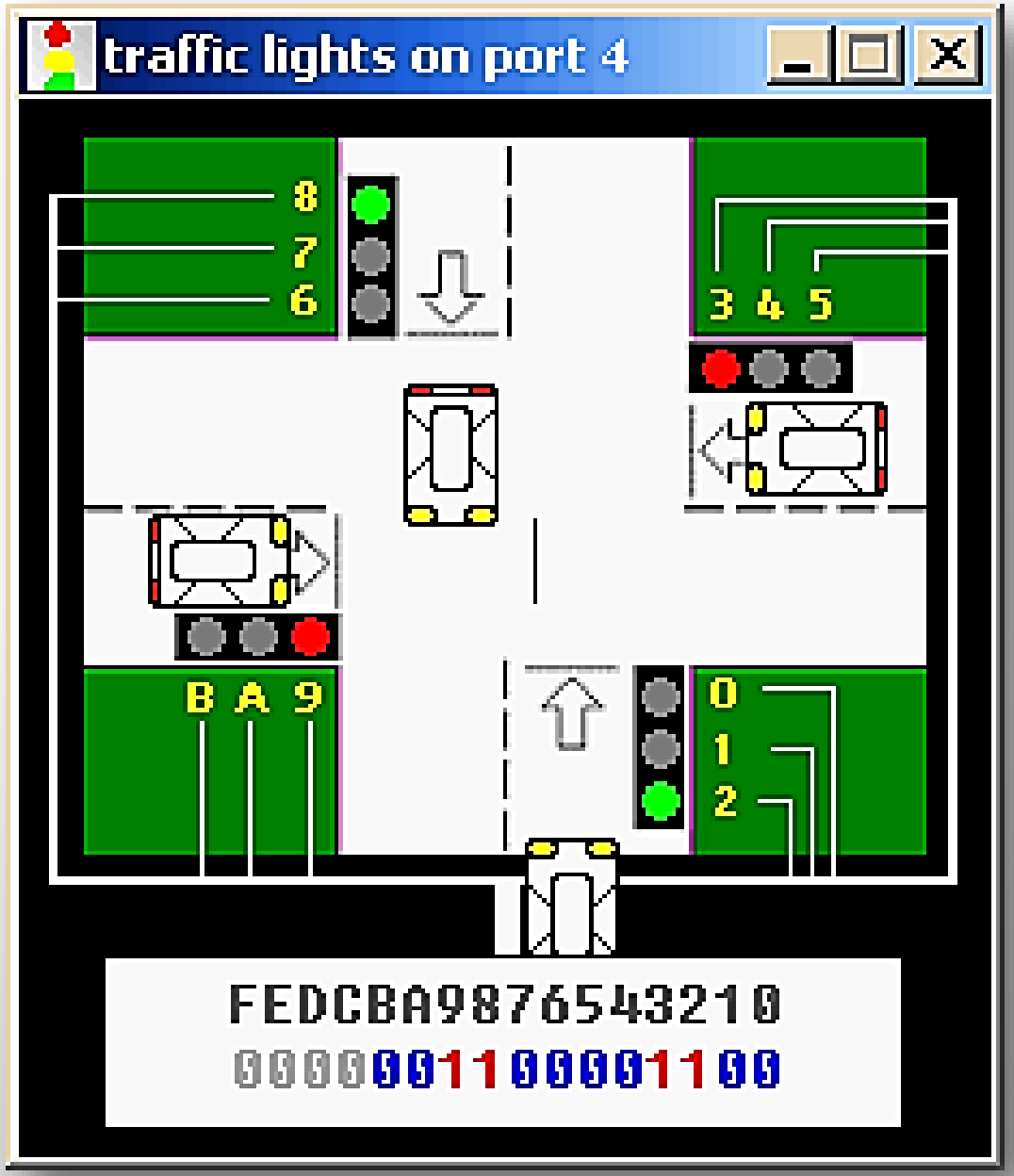
8086 Emülatörü

- Emülatöre bağlı 7 farklı aygıt bulunmakta:
  - trafik lambaları (*traffic lights*),
  - step motor (*stepper motor*),
  - LED ekran (*LED display*),
  - termometre (*thermometer*),
  - yazıcı (*printer*),
  - robot (*robot*),
  - basit bir test cihazı (*test device*).

Aygıt Kontrolü

- X86 ailesi CPU'lar, geniş bir aygıt yelpazesini kontrol edebilir.
  - Temel G/Ç port numaralarında farklılıklar olabilir.
  - Hileli (*tricky*) elektronik bir ekipman kullanılarak değiştirilebilir.
- *.bin* uzantılı dosya, Salt Okunur Bellek (ROM) yongasına yazılır.
- Sistem, programı
  - ROM yongasından okur,
  - RAM modülüne yükler,
  - ve programı yürütür.
- Bu prensip, birçok modern cihazda kullanılır.

Traffic Lights



Trafik Işıkları Kontrol

```
mov ax, all_red
out 4, ax

mov si, offset situation

next:
mov ax, [si]
out 4, ax
```

Trafik Işıkları Kontrol

```
; wait 5 seconds (5 million microseconds)
mov cx, 4Ch     ; 004C4B40h = 5,000,000
mov dx, 4B40h
mov ah, 86h
int 15h
```

Trafik Işıkları Kontrol

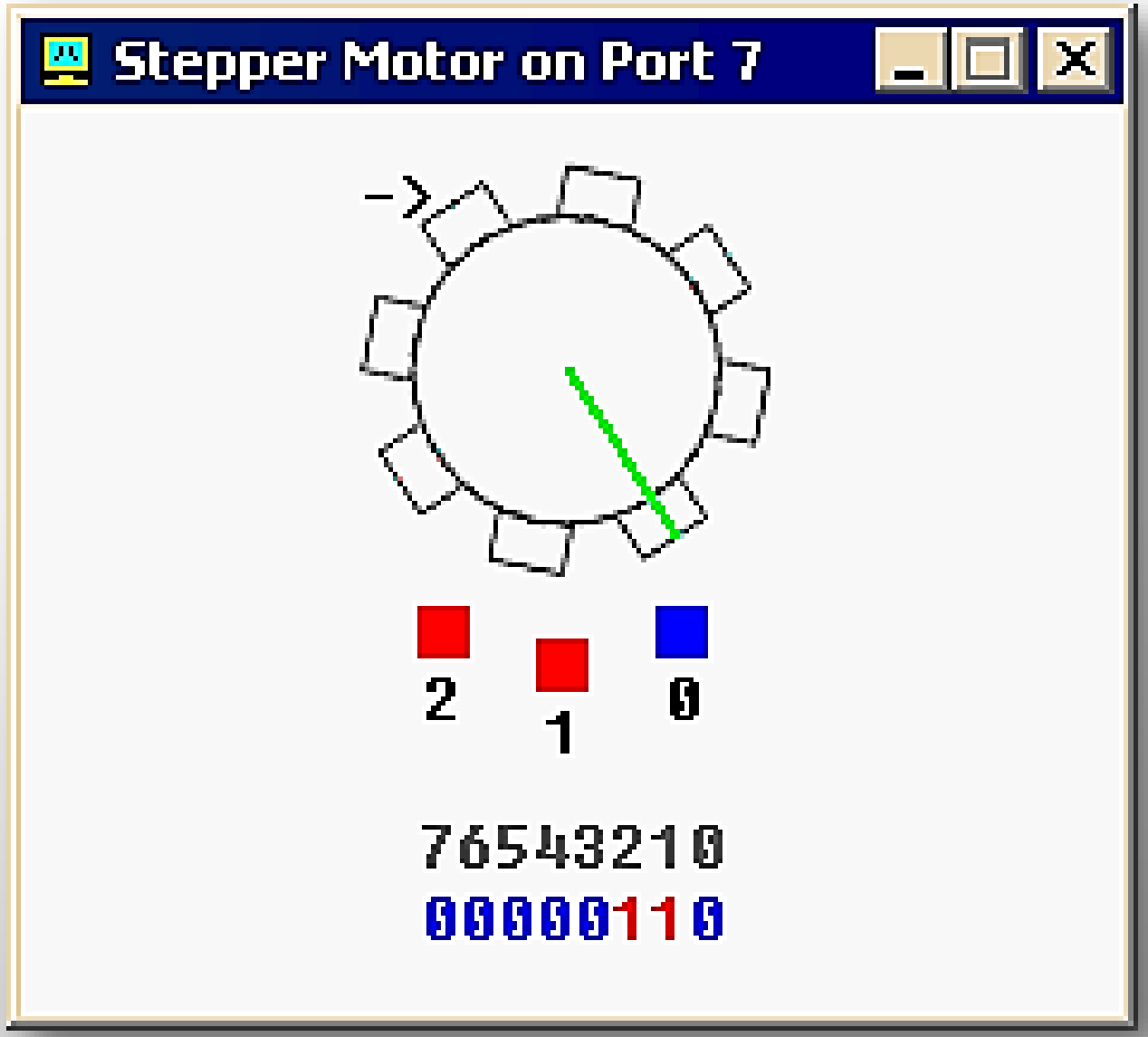
```
add si, 2 ; next situation
cmp si, sit_end
jb  next
mov si, offset situation
jmp next
```

Trafik Işıkları Kontrol

```
;                       FEDC_BA98_7654_3210
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
```

Stepper-Motor



Step Motoru

- Motor, hem saat yönünde hem de tersine döndürülebilir.
- Yarı adım, 11.25 dereceye eşittir.
- Tam adım, 22.5 dereceye eşittir.
- Motor, çift mıknatıs, ardından tek mıknatıs açarak yarı adım döndürülebilir.
- Tam adım için, çift mıknatıs, çift mıknatıs ve tek mıknatıs ile devam eder.

Step Motoru Kontrol

```
steps_before_direction_change = 20h ; 32 (decimal)

jmp start
; ========== data ================
; bin data for clock-wise
; half-step rotation:
datcw    db 0000_0110b
         db 0000_0100b
         db 0000_0011b
         db 0000_0010b
```

Step Motoru Kontrol

```
; bin data for counter-clock-wise
; half-step rotation:
datccw   db 0000_0011b
         db 0000_0001b
         db 0000_0110b
         db 0000_0010b
; bin data for clock-wise
; full-step rotation:
datcw_fs db 0000_0001b
         db 0000_0011b
         db 0000_0110b
         db 0000_0000b
```

Step Motoru Kontrol

```
; bin data for counter-clock-wise
; full-step rotation:
datccw_fs db 0000_0100b
          db 0000_0110b
          db 0000_0011b
          db 0000_0000b
start:
mov bx, offset datcw ; start from clock-wise half-step.
mov si, 0
mov cx, 0 ; step counter
```

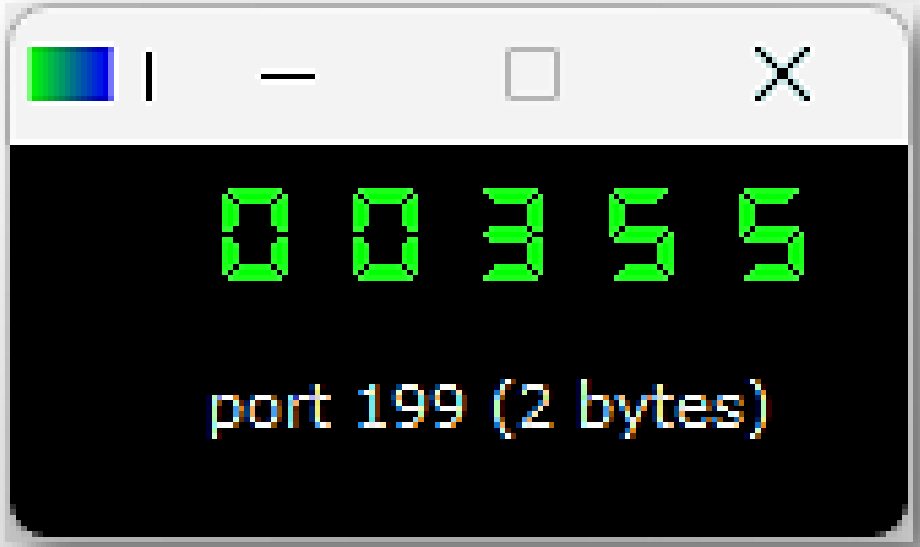
Step Motoru Kontrol

```
next_step:
; motor sets top bit when it's ready to accept new command
wait:   in al, 7
        test al, 10000000b
        jz wait
mov al, [bx][si]
out 7, al
inc si
cmp si, 4
jb next_step
mov si, 0
```

Step Motoru Kontrol

```
inc cx
cmp cx, steps_before_direction_change
jb  next_step
mov cx, 0
add bx, 4 ; next bin data
cmp bx, offset datccw_fs
jbe next_step
mov bx, offset datcw ; return to clock-wise half-step.
jmp next_step
```

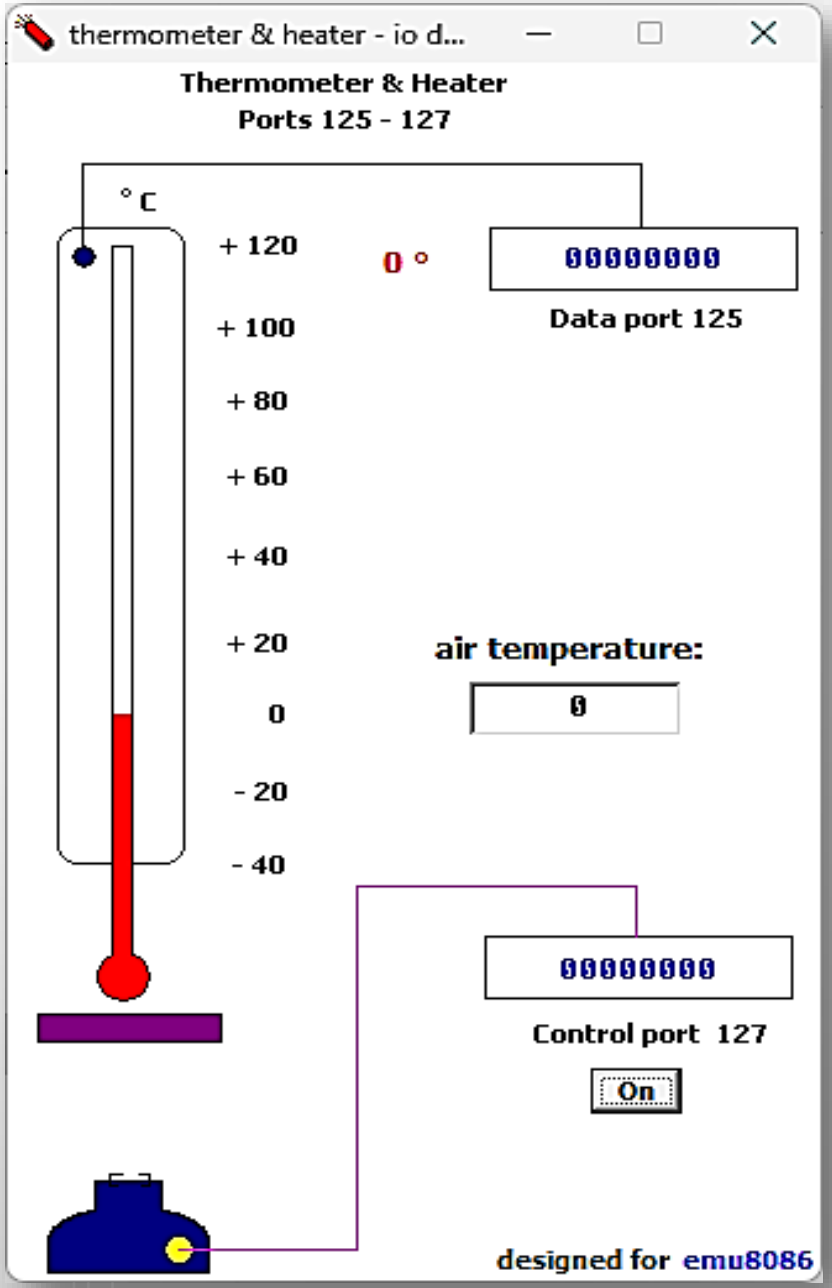
LED Display



LED Ekran Kontrolü

```
mov ax, 1234
out 199, ax
mov ax, -5678
out 199, ax
; Eternal loop to write values to port:
mov ax, 0
x1:
out 199, ax
inc ax
jmp x1
hlt
```

Thermometer



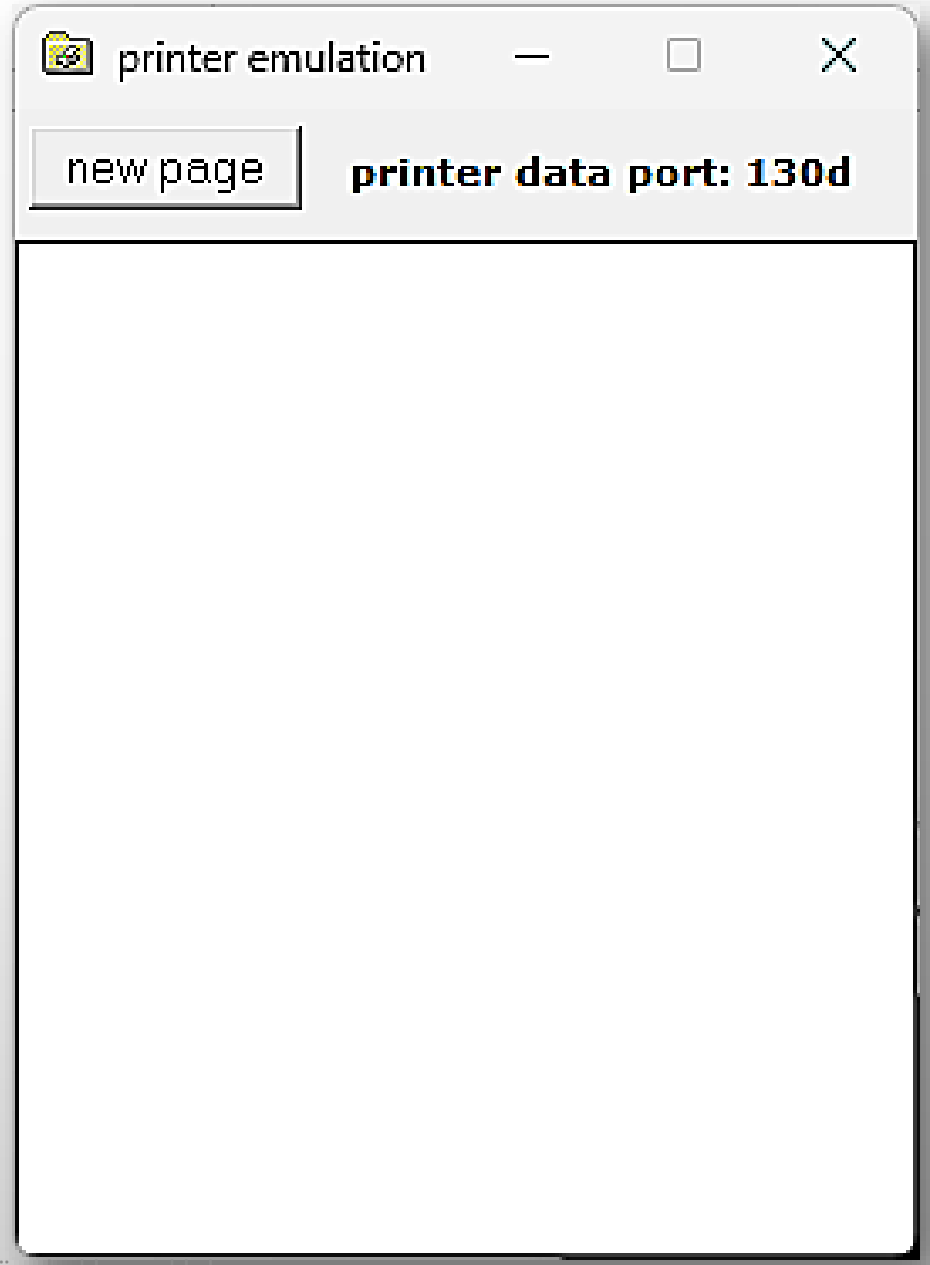
Termometre Kontrolü

```
; set data segment to code segment:
mov ax, cs
mov ds, ax
start:
in al, 125
cmp al, 60
jl  low
cmp al, 80
jle  ok
jg   high
```

Termometre Kontrolü

```
low:
mov al, 1
out 127, al   ; turn heater "on".
jmp ok
high:
mov al, 0
out 127, al   ; turn heater "off".
ok:
jmp start   ; endless loop.
```

Printer



Printer Kontrolü

```
org 100h

jmp start
msg db "Hello, Printer!", 0Ah, 0Dh
    db "***************"
    db 13, 9    ; carriage return and vertical tab
    db "Have a nice printing day!"
msg_end db 0
msg2 db "press any key to eject the page.$"
```

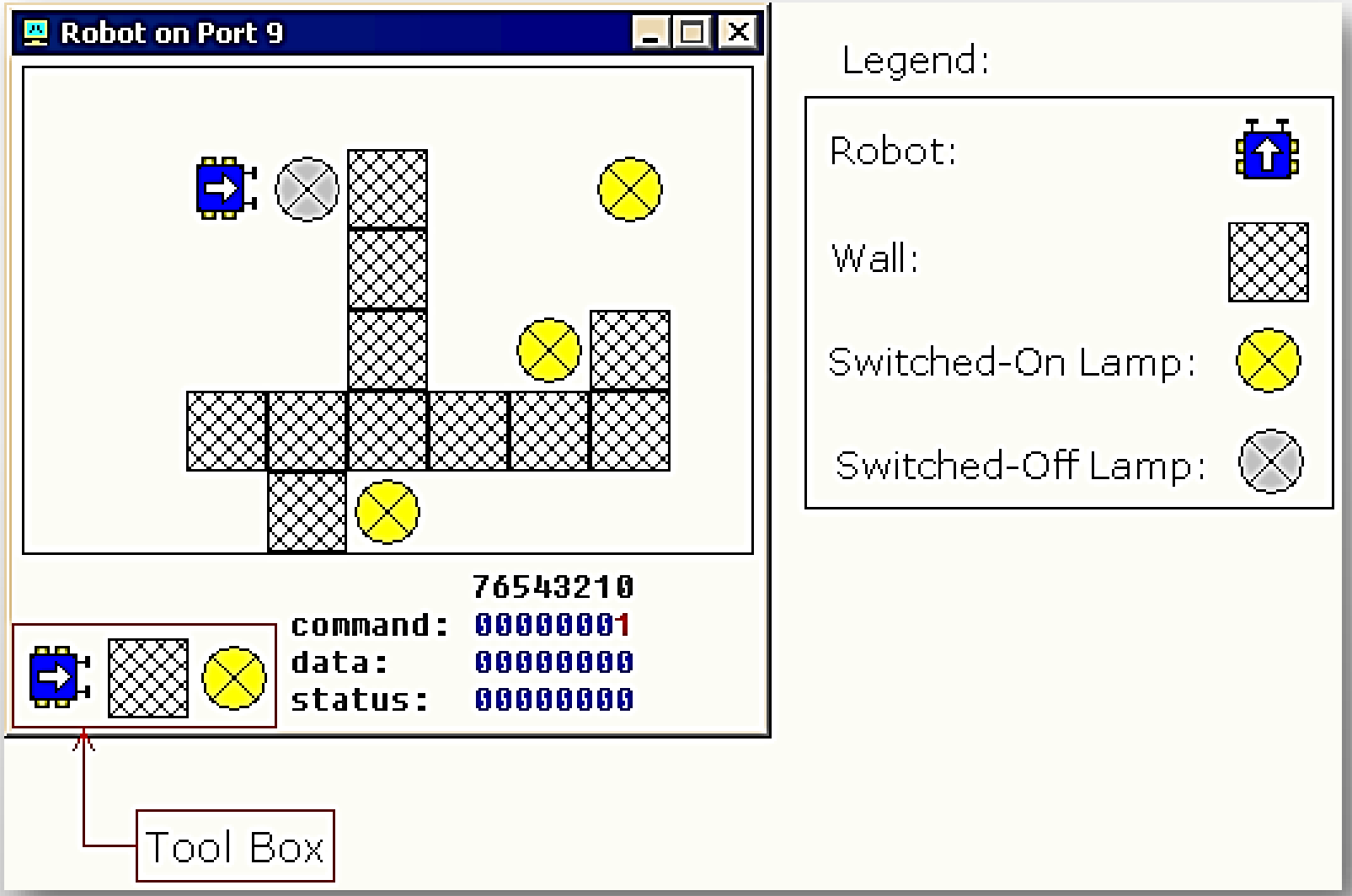
Printer Kontrolü

```
start:
    mov dl, 12        ; form feed code. new page.
    mov ah, 5
    int 21h
    mov si, offset msg
    mov cx, offset msg_end - offset msg
print:
    mov dl, [si]
    mov ah, 5         ; MS-DOS print function.
    int 21h
    inc si            ; next char.
    loop print
```

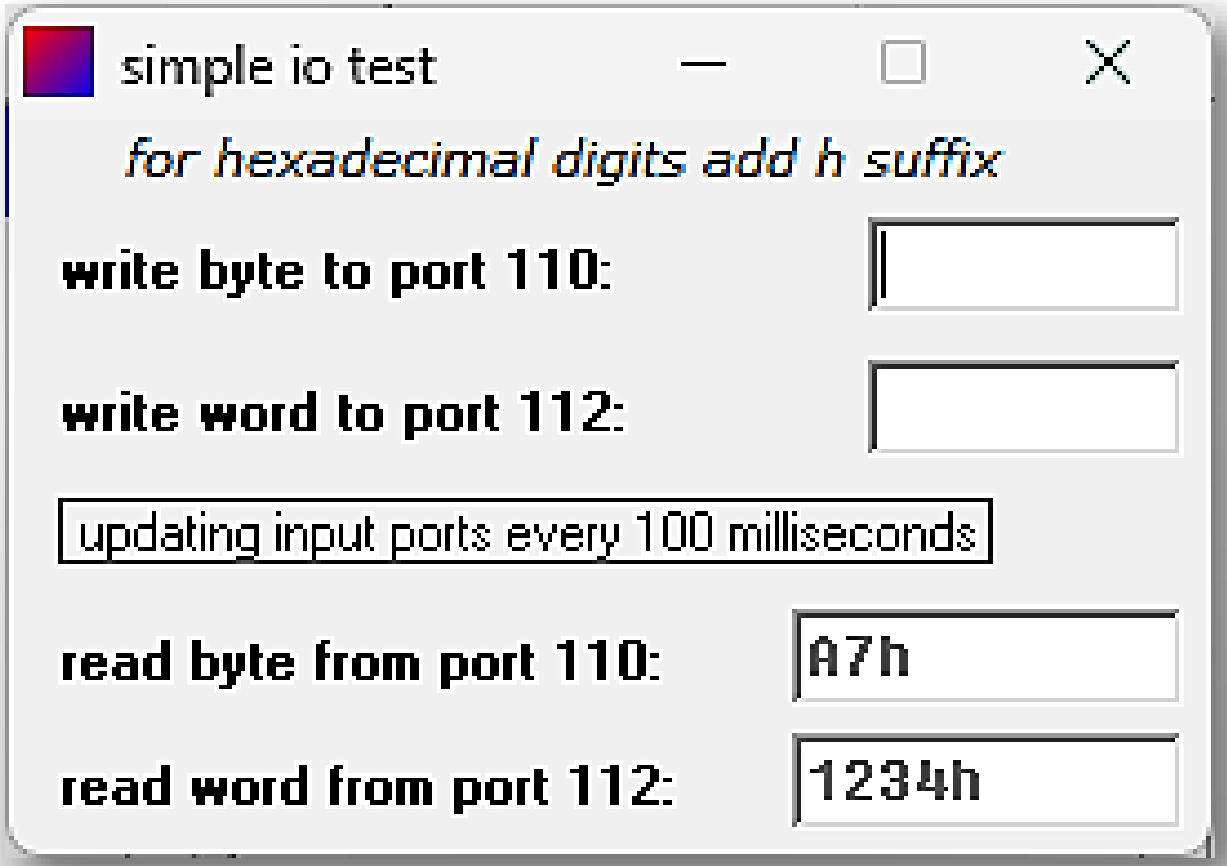
Printer Kontrolü

```
    mov dx, offset msg2
    mov ah, 9
    int 21h
    mov ax, 0          ; wait for any key...
    int 16h
    mov dl, 12         ; form feed code. page out!
    mov ah, 5
    int 21h
ret
```

Robot



Test Device



SON