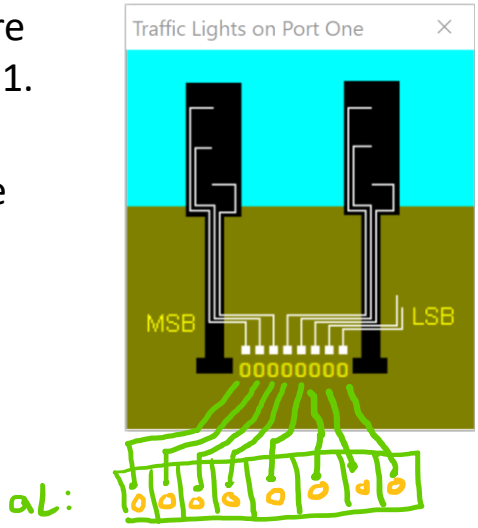
Writing to an Output Port

- The traffic lights are connected to Port 1.
- The graphic shows how the bits in the AL register are attached to the various lights:



140

Writing to an Output Port

- Note that the two least significant bits are unconnected in the device. Thus, the values that we associate with them are irrelevant – i.e., they can be any combination of 0 and 1.
- To write a value to a port
  - Place the value in the al register.
    - Determining the proper value require mapping the binary value in the register to a hex value in the program.
  - Issue the instruction OUT 01

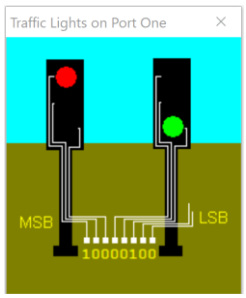
141

Binary and Hex

- Recall the bit sequence corresponding to each hex digit.

Binary				Hex
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

142



- To display red-green on the traffic lights

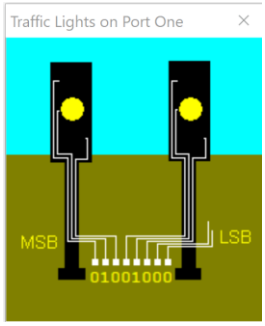
```
Source Code | L
mov al, 84
out 01
end
```

al: 10000100

8 4 Hex

Note: 85, 86 and 87 are also valid since the LSBs are unconnected

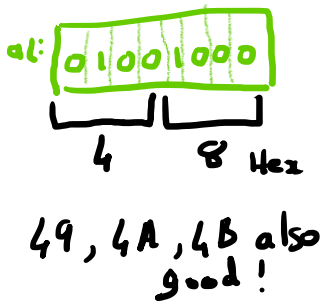
143



- To display orange-orange on the traffic lights

Source Code

```
mov al, 48
out 01
end
```



144

- Similarly,
  - green-red corresponds to 00110000 = 30
- To animate the traffic lights we can write:

Source Code | List File | Configuration | Tokens | R

```
;traffic light sequencing
start:
 mov al, 84 ; 10000100 - red-green
 out 01
 mov al, 48 ; 01001000 - orange-orange
 out 01
 mov al, 30 ; 00110000 - green-red
 out 01
 mov al, 48 ; 01001000 - orange-orange (again)
 out 01
jmp start
end
```

145

Introducing Delays

- The following code introduce a delay that causes the traffic lights to linger in specific configurations for a period of time.
- The delay is generated by placing a value in a register and decrementing this value in a loop until it reaches zero.
- A jump is used to transfer control to the delay code and another jump and an appropriate label is used to transfer control back.

146

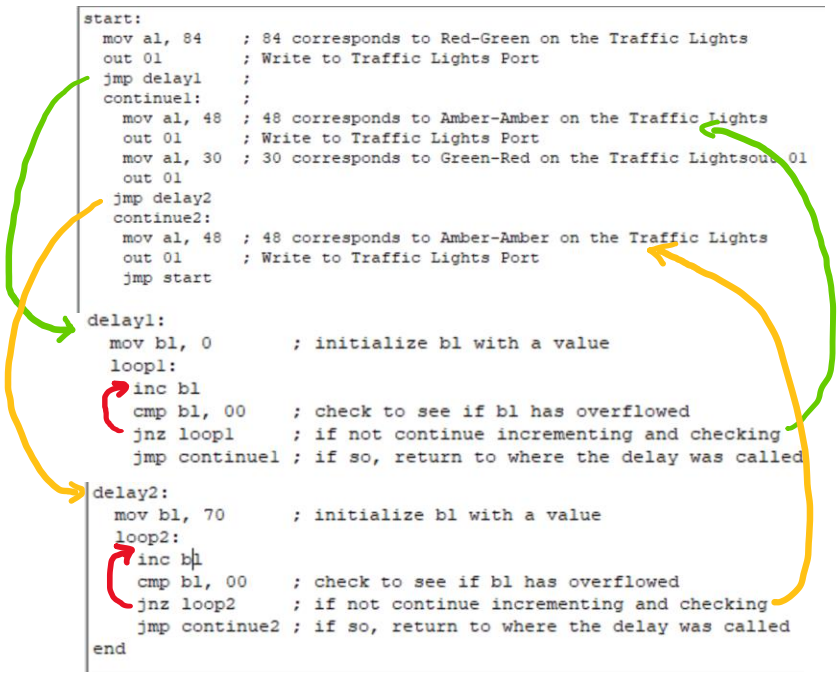
```
start:
  mov al, 84     ; 84 corresponds to Red-Green on the Traffic Lights
  out 01         ; Write to Traffic Lights Port
  jmp delay1     ;
  continue1:     ;
    mov al, 48   ; 48 corresponds to Amber-Amber on the Traffic Lights
    out 01       ; Write to Traffic Lights Port
    mov al, 30   ; 30 corresponds to Green-Red on the Traffic Lightsout 01
    mov al, 48   ; 48 corresponds to Amber-Amber on the Traffic Lights
    out 01       ; Write to Traffic Lights Port
    jmp start

delay1:
  mov bl, 0      ; initialize bl with a value
  loop1:
    inc bl
    cmp bl, 00   ; check to see if bl has overflowed
    jnz loop1    ; if not continue incrementing and checking
    jmp continue1 ; if so, return to where the delay was called
end
```

147

Introducing Delays

- The following code introduce two delays that cause the traffic lights to linger in specific configurations for different periods of time.

148



```
start:
  mov al, 84    ; 84 corresponds to Red-Green on the Traffic Lights
  out 01        ; Write to Traffic Lights Port
  jmp delay1    ;
  continue1:    ;
    mov al, 48  ; 48 corresponds to Amber-Amber on the Traffic Lights
    out 01      ; Write to Traffic Lights Port
    mov al, 30  ; 30 corresponds to Green-Red on the Traffic Lightsout 01
    out 01
   jmp delay2
   continue2:
    mov al, 48  ; 48 corresponds to Amber-Amber on the Traffic Lights
    out 01      ; Write to Traffic Lights Port
    jmp start

delay1:
   mov bl, 0        ; initialize bl with a value
   loop1:
     inc bl
     cmp bl, 00     ; check to see if bl has overflowed
     jnz loop1      ; if not continue incrementing and checking
     jmp continue1 ; if so, return to where the delay was called

delay2:
   mov bl, 70       ; initialize bl with a value
   loop2:
     inc bl
     cmp bl, 00     ; check to see if bl has overflowed
     jnz loop2      ; if not continue incrementing and checking
     jmp continue2 ; if so, return to where the delay was called
end
```

149

The Jump Structure

- Notice: we jump to some code, do something and jump back to the instruction following the original jump instruction.
- Notice also that apart from the duration of each delay that the code of each loop is exactly the same.
- Surely there must be a better way of doing this without duplicating code, peppering the code with multiple jump instructions and trying to remember all those labels.

150