1. Write an assembly program that displays the binary pattern from 0 to 255 (and back to 0) on the LEDs interfaced with port 1.

**Solution:**

**; This program displays the binary pattern**

**; from 0 to 255 (and back to 0) on the LEDs**

**; interfaced with port 1.**

**; A 1 in the pattern is represented by the LED on,**

**; while a 0 in the pattern is represented by the LED off.**

**; However, logic 0 on a port 1 pin turns on the LED,**

**; therefore it is necessary to write the inverse of the**

**; pattern to the LEDs. The easiest way to do this is**

**; to send the data FFH to 0 (and back to FFH) to the LEDs.**

**; Since port 1 is initially at FFH all we need to do is**

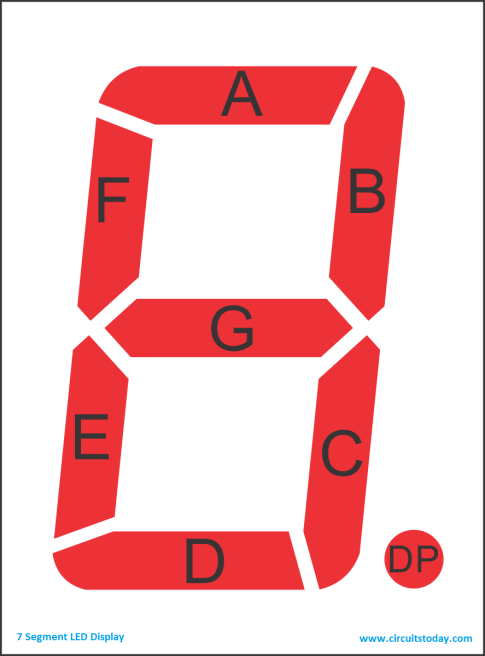
**; continuously decrement port 1.**

**start:**

**DEC P1**

**JMP start**

**Seven Segment:**

P1.7 → Seg. **dp**

P1.6 → Seg. **g**

P1.5 → Seg. **f**

P1.4 → Seg. **e**

P1.3 → Seg. **d**

P1.2 → Seg. **c**

P1.1 → Seg. **b**

P1.0 → Seg. **a**

Note: Any bit of the Port set to 0, would lit up the corresponding segment on the display. (Ex-- *MOV P1, #249*)

dp g f e d c b a

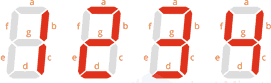
1 1 1 1 1 0 0 1 (249) -- Show-1

1 0 1 0 0 1 0 0 (164) -- Show-2

1 0 1 1 0 0 0 0 (176) -- Show-3

1 0 0 1 1 0 0 1 (153) -- Show-4

1. Write an assembly language program that multiplexes the number 1234 on the four 7-segment displays.

Display-1 = show 1

Display-2 = show 2

Display-3 = show 3

Display-4 = show 4

**Select Display-1** => P3.3=1 and P3.4=1

**Select Display-2** => P3.3=0 and P3.4=1

**Select Display-3** => P3.3=1 and P3.4=0

**Select Display-4** => P3.3=0 and P3.4=0

**Solution:**

; This program multiplexes the number 1234

; on the four 7-segment displays.

; Note: a logic 0 lights a display segment.

start:

SETB P3.3 ; |

SETB P3.4 ; | enable display 3

MOV P1, #11111001B ; put pattern for 1 on display

CALL delay

CLR P3.3 ; enable display 2

MOV P1, #10100100B ; put pattern for 2 on display

CALL delay

CLR P3.4 ; |

SETB P3.3 ; | enable display 1

MOV P1, #10110000B ; put pattern for 3 on display

CALL delay

CLR P3.3 ; enable display 0

MOV P1, #10011001B ; put pattern for 4 on display

CALL delay

JMP start ; jump back to start

; a crude delay

delay:

MOV R0, #200

DJNZ R0, $

RET

Ref:

<https://www.youtube.com/watch?v=Rn_97h4N6mM>

<https://technobyte.org/seven-segment-8051-interfacing-tutorial-quad-single-module/>

<https://factorization.info/prime-factors/0/prime-factors-of-8051.html>