Module 6
Part - I

- INTERFACING DIP SWITCH WITH 8051
- INTERFACING LEDs WITH 8051
- INTERFACING SEVEN SEGMENT DISPLAY WITH 8051

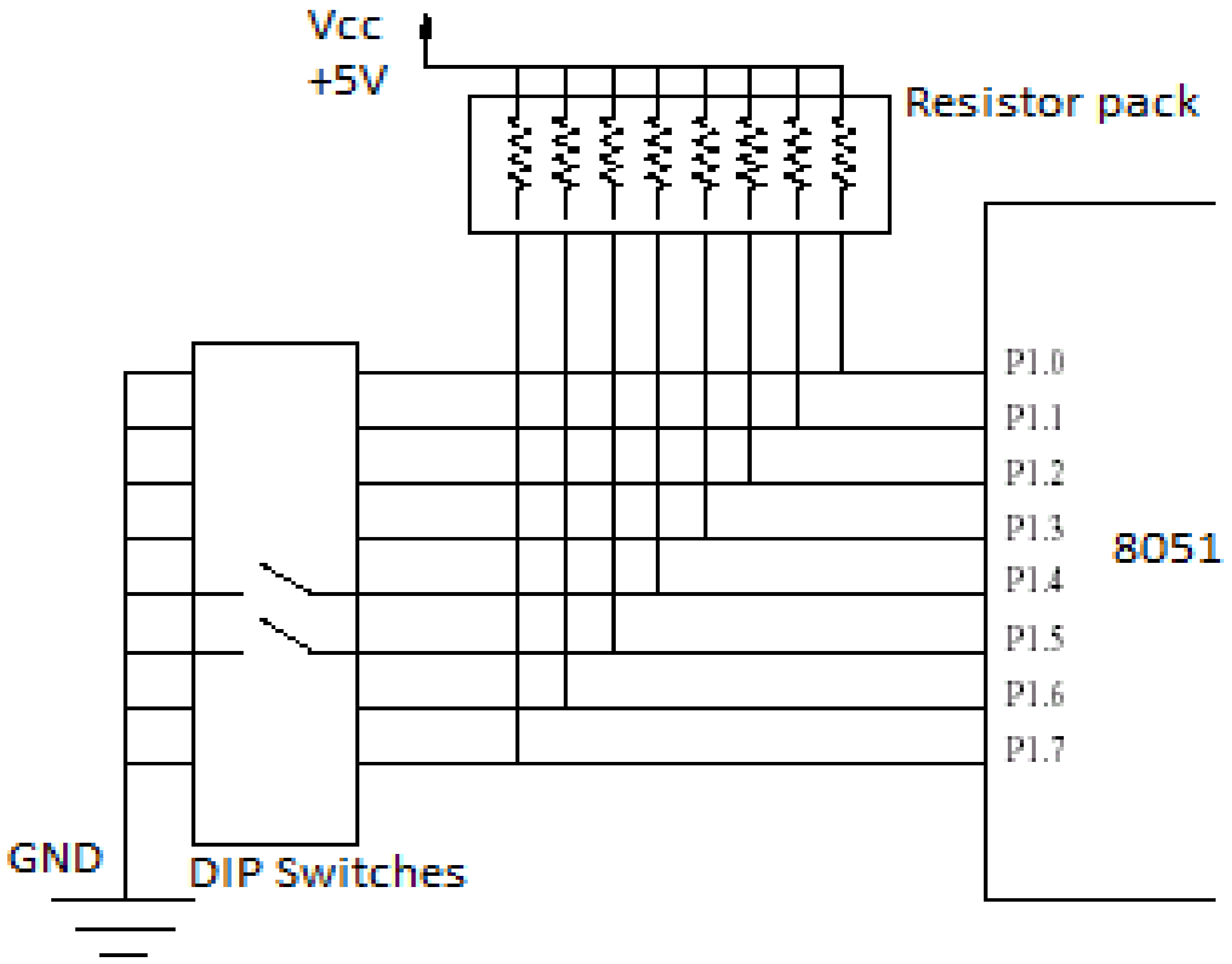
INTERFACING DIP SWITCH WITH 8051

- DIP Switches are manual electric switches that are packaged by group into a standard dual in- line package (DIP).

- DIP switches are also known as toggle switches, which mean they have two possible positions - on or off.

- DIP switches usually have 8 switches.

INTERFACING THE SWITCH TO MICROCONTROLLER



- A simple switch has an open state and closed state.
- However, a microcontroller needs to see a definite high or low voltage level at a digital input.
- A switch requires a pull-up resistor to produce a definite high or low voltage when it is open or closed.
- A resistor placed between a digital input and the supply voltage is called a "pull-up” resistor because it normally pulls the pin’s voltage upto the supply

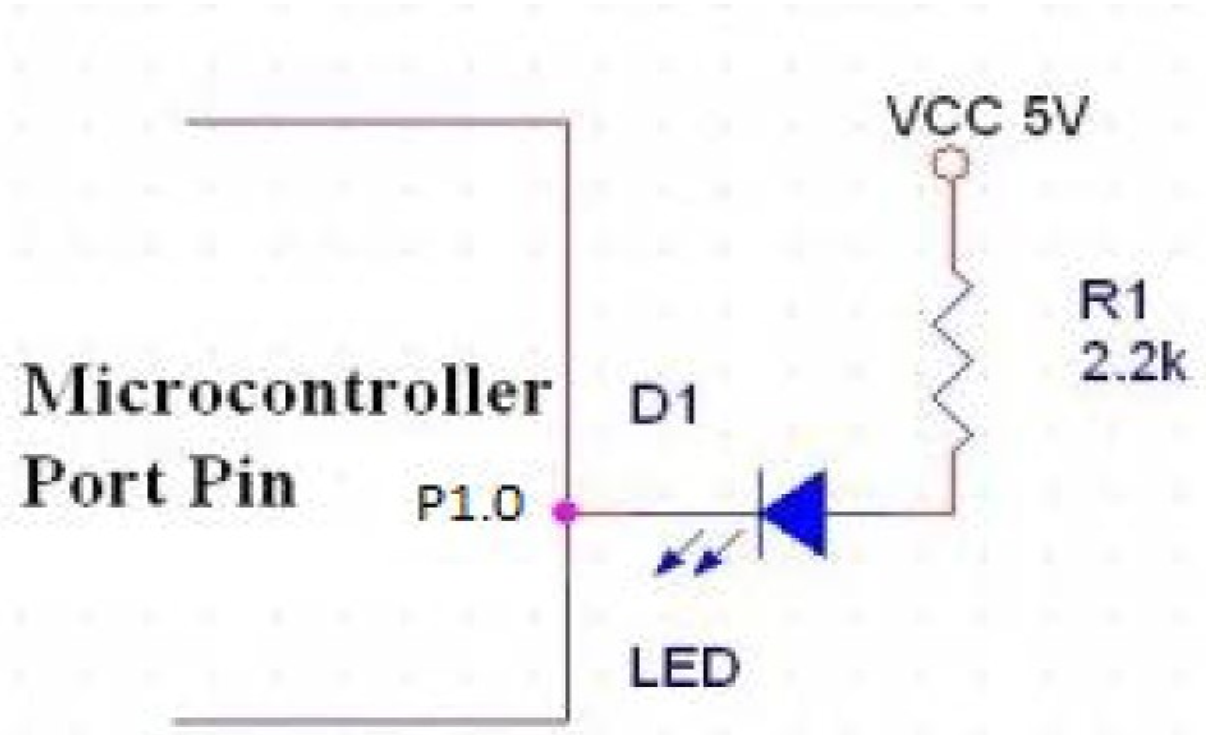
- Note: **To read an I/O port, initially a 1 is to be written** to each and every pin of that port. That is we have to write "FF" to the port if we use whole 8 bits of a particular port.
- E.g.:- **MOV P1,#FFH** ; then read the port using

  MOV A,P1
- Therefore, the program to read the port where the DIP switches are connected is as follows:

```
ORG 0000H
MOV P1,#FFH ; Initialise port
MOV A,P1 ; Move data from P1 to A.
END
```

INTERFACING LEDs WITH 8051

- LEDs can be connected to any I/O pin of 8051 as shown in figure.



- A current limiting resistor should be connected in series with **Vcc** or **GND**.

PROGRAMMING

- Flashing LED ALGORITHM
    1. Start.
    2. Turn ON LED.
    3. Wait for some time (delay).
    4. Turn OFF LED.
    5. Wait for some time (delay).

- Generating delay : Loop technique
    1. Start.
    2. Load a number in a Register. e.g. R0.
    3. Decrement Register.
    4. Is Register = 00? If NO, GO TO 3.
    5. STOP

PROGRAM:

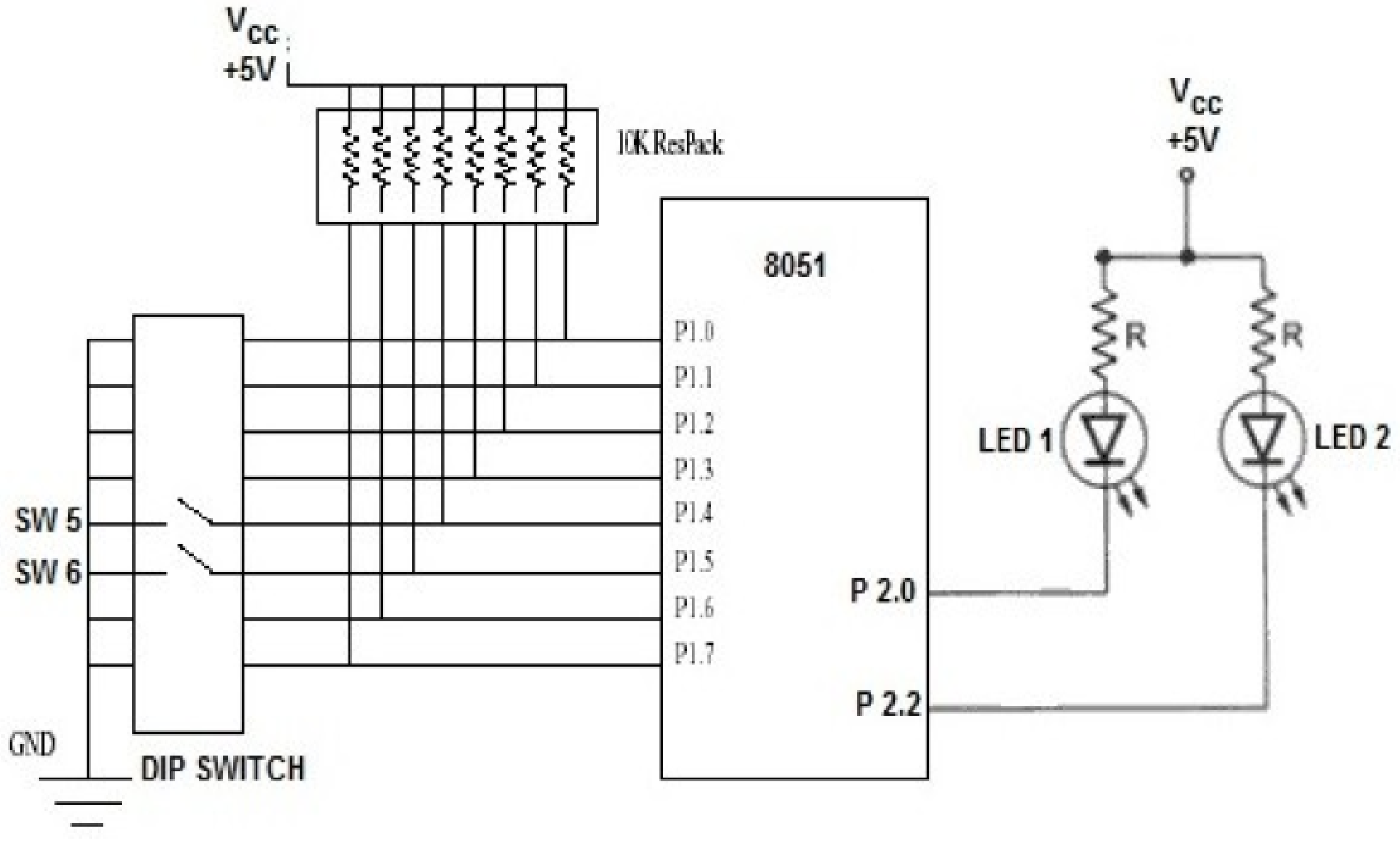
- In 8051 a single instruction "**DJNZ**" is specifically designed for this kind of programs. It stands for **Decrement & Jump if Not Zero**. This instruction takes care or STEP 3 & STEP 4 of the above algorithm

```
            ORG 0000H
LOOP:  CLR P1.0
            ACALL DELAY
            SETB P1.0.
            ACALL DELAY
            SJMP LOOP
```

- Delay Program

```
DELAY: MOV R7, #FF
AGAIN: DJNZ R7, AGAIN
              RET
```

- A DIP switch is connected to Port 1 and two LEDs are connected to P2.0 and P2.2 respectively. Write a program to blink the LED 1 if SW5 is ON and blink the LED2 if SW6 is ON. Note that at a time only one LED should be ON.



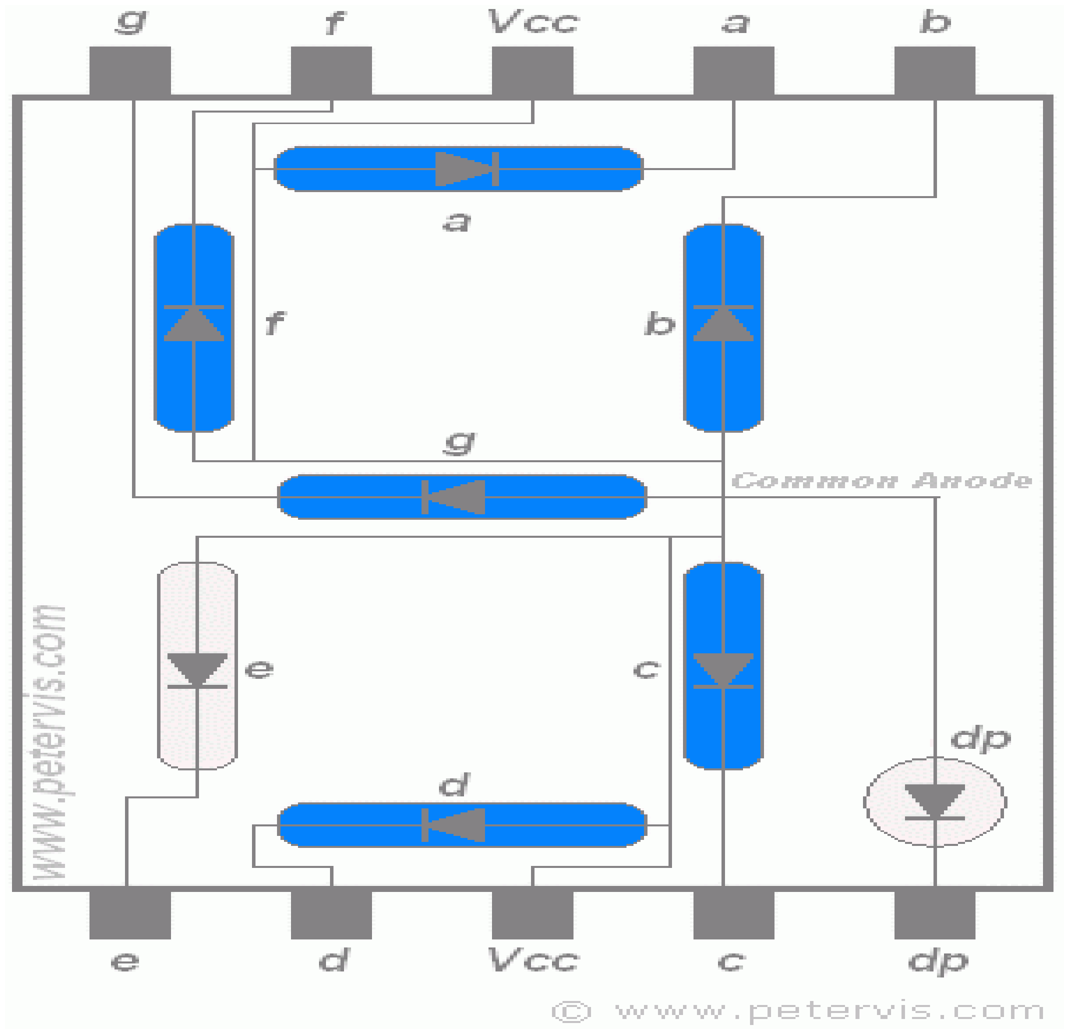
Program

```
          ORG 0000H
          MOV P1,#0FFH
AGAIN1:   JB P1.4, LED1
AGAIN2:   JB P1.5, LED2
LED1:     CLR P2.0
          SETB P2.0
          ACALL DELAY
          SJMP AGAIN1
LED2:     CLR P2.2
          ACALL DELAY
          SETB P2.2
          ACALL DELAY
          SJMP AGAIN2
```

INTERFACING SEVEN SEGMENT DISPLAY WITH 8051

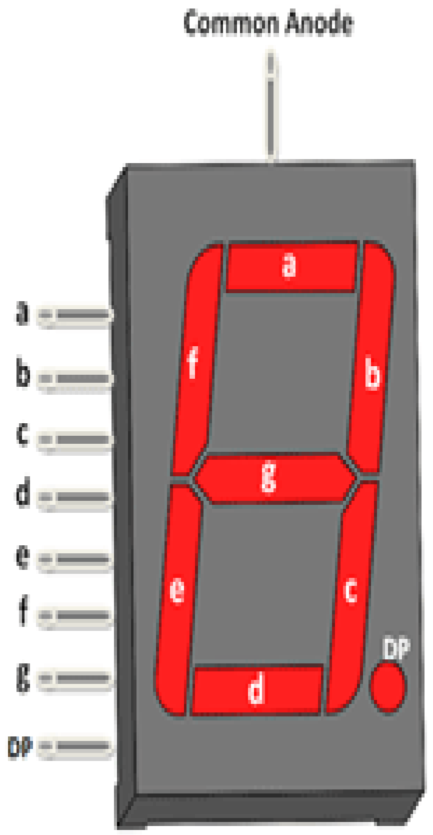
- A seven segment display consists of seven LEDs arranged in the form of a squarish '8'.
- Different characters can be displayed by selectively glowing the required LED segments.
- Seven segment displays are of two types,
  - ***common cathode***
  - ***common anode.***

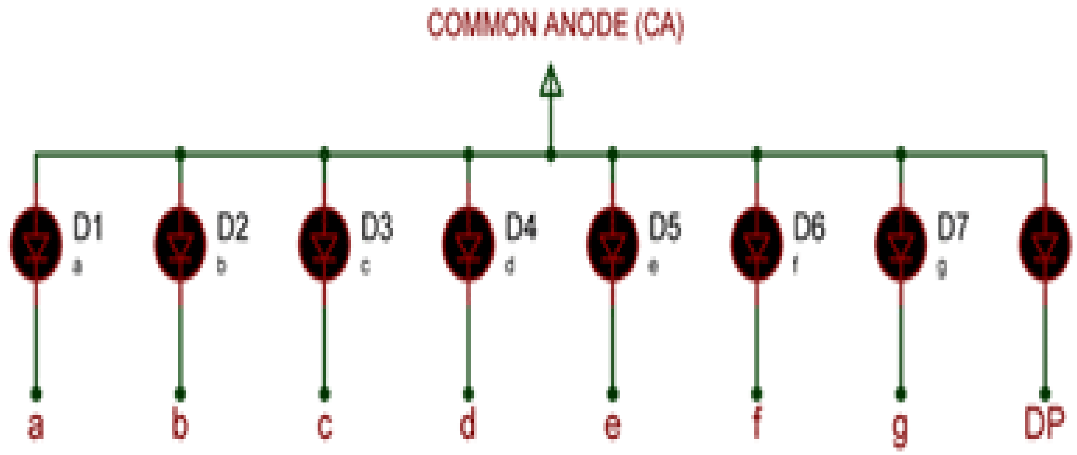
SEVEN SEGMENT DISPLAY



Common anode

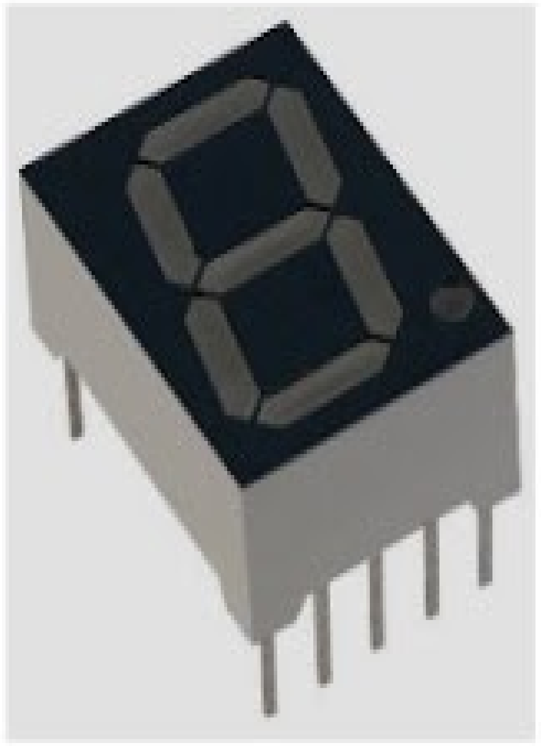
- In common anode type, **the anode of all LEDs are tied together as a single terminal** and cathodes are left alone as individual pins.

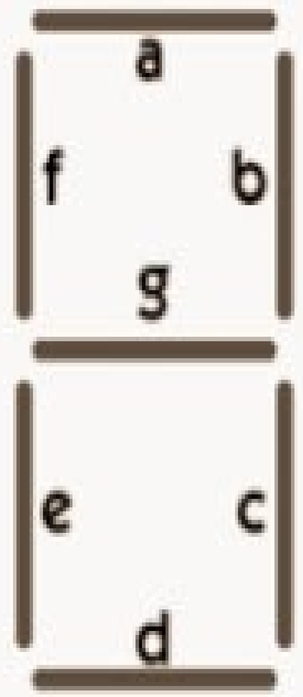


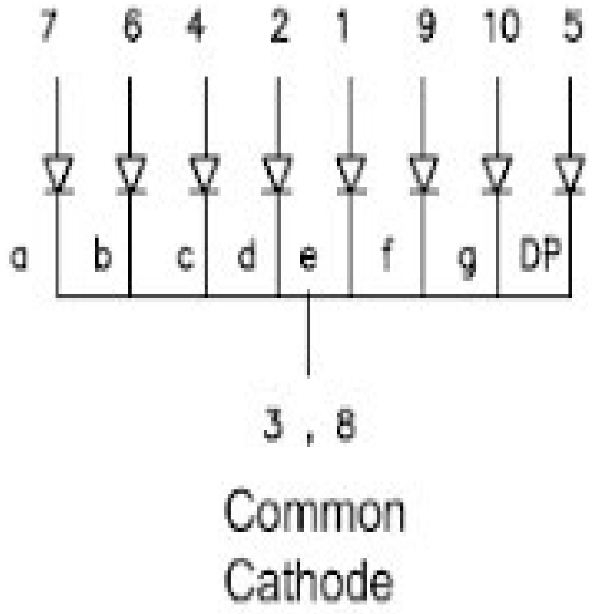


Common cathode

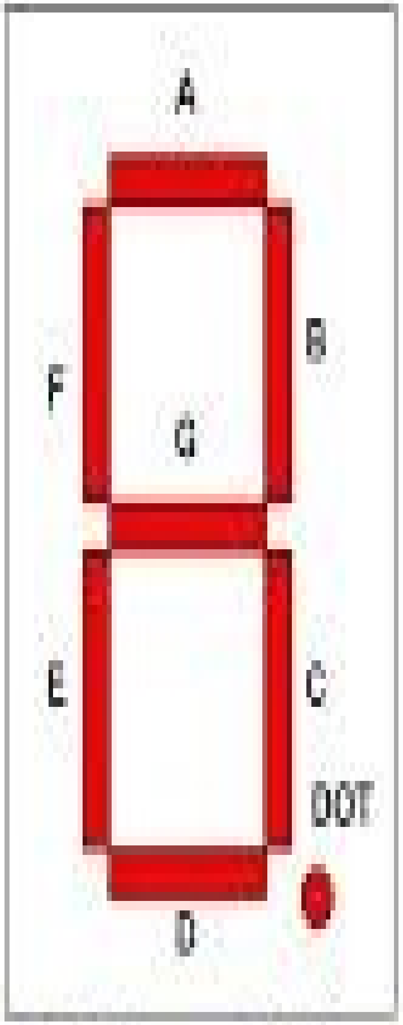
- In common cathode type , the **cathode of all LEDs are tied together to a single terminal** which is usually labeled as '**com**' and the anode of all LEDs are left alone as individual pins labeled as a, b, c, d, e, f, g & h (or dot) .



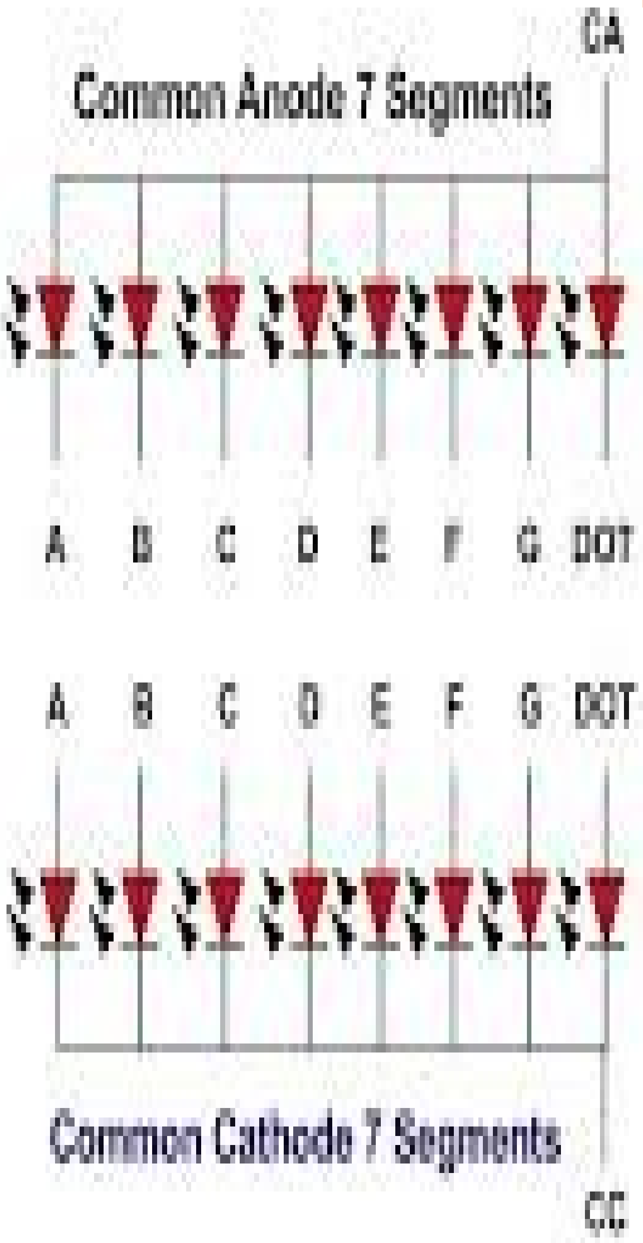


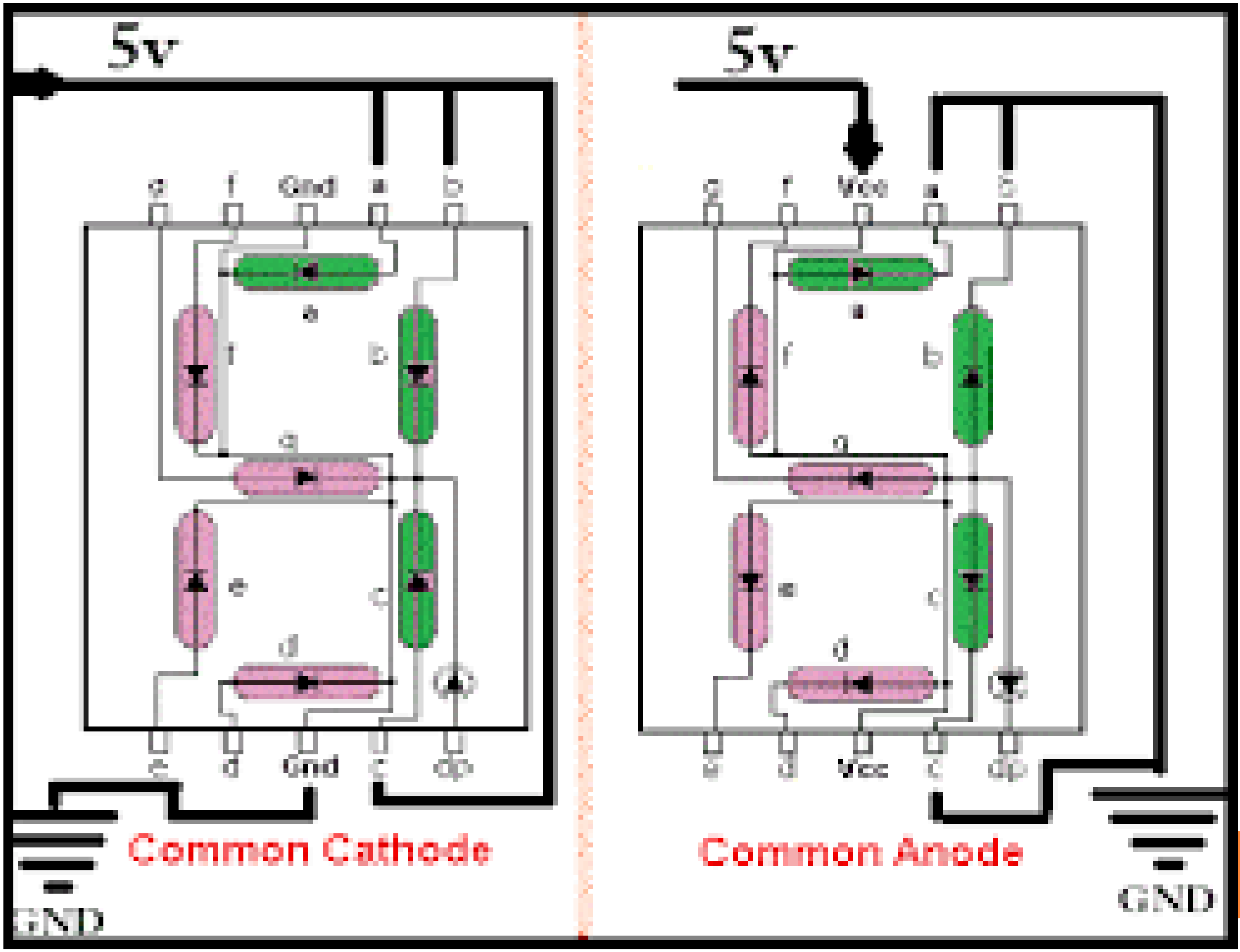


Typical 7 Segments Display



The 7 Segment's Name and the DOT




5v
g
f
Gnd
a
b
e
d
Gnd
c
dp
GND
Common Cathode
5v
g
f
Vcc
a
b
e
d
Vcc
c
dp
GND
Common Anode

Steps for interfacing 7 segment display with 8051

- Check if the seven segment is **common anode** or **common cathode.**
- If it is **common anode** then **connect a VCC** to the **common anode pin**.
  - To **switch on any of the respective segment/LED pass 0** to that pin through our 8051 microcontroller.
- If the Seven Segment is **common cathode connect Gnd** to the **common cathode pin**.
  - To **switch on any of the respective segments/LED give 1** to that pin through our 8051 microcontroller

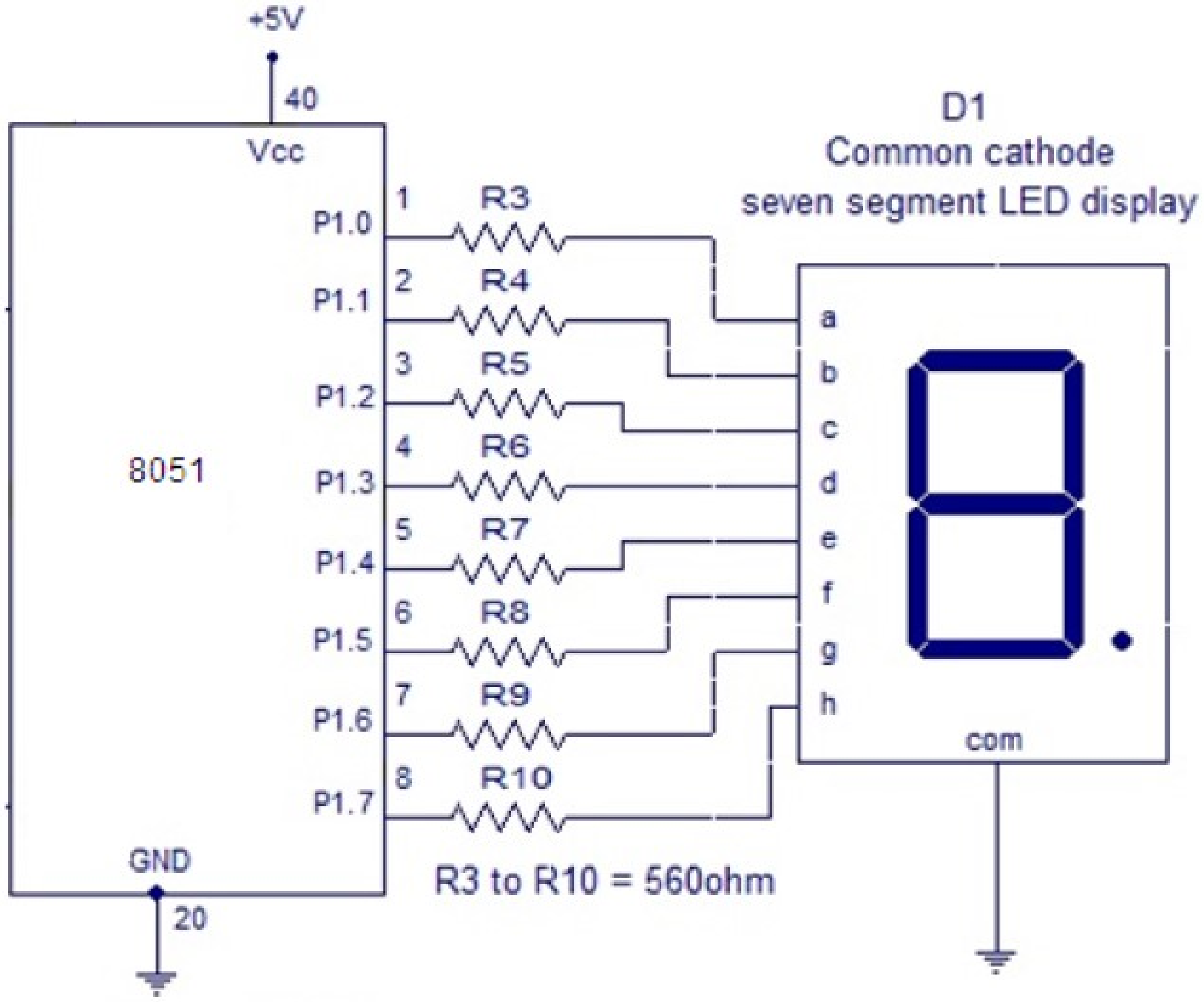
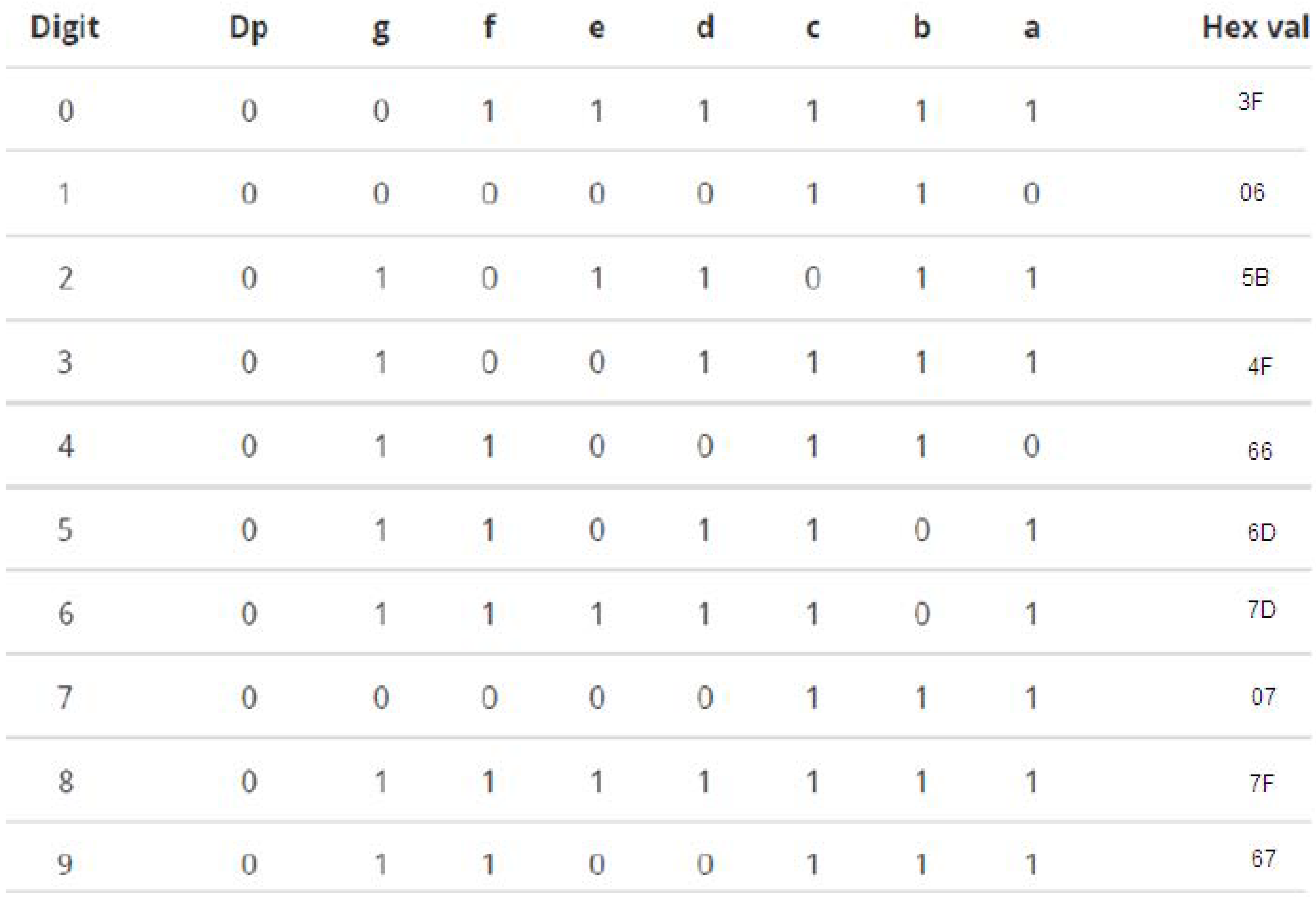


Figure: Interfacing seven segment display with 8051

- Following are the 8 bit values to display various digits for a common cathode type seven segment display.



Digit	Dp	g	f	e	d	c	b	a	Hex val
0	0	0	1	1	1	1	1	1	3F
1	0	0	0	0	0	1	1	0	06
2	0	1	0	1	1	0	1	1	5B
3	0	1	0	0	1	1	1	1	4F
4	0	1	1	0	0	1	1	0	66
5	0	1	1	0	1	1	0	1	6D
6	0	1	1	1	1	1	0	1	7D
7	0	0	0	0	0	1	1	1	07
8	0	1	1	1	1	1	1	1	7F
9	0	1	1	0	0	1	1	1	67

Program to display various digits from 0 to 9 with a delay

```
        ORG 0000H
AGAIN:  MOV A,#3F
        MOV P1, A
        ACALL DELAY
        MOV A,#06
        MOV P1, A
        ACALL DELAY
        MOV A,#5B
        MOV P1, A
        ACALL DELAY
        MOV A,#4F
        MOV P1, A
        ACALL DELAY
        MOV A,#66
        MOV P1, A
        ACALL DELAY
        MOV A,#6D
        MOV P1, A
        ACALL DELAY
        MOV A,#7D
        MOV P1, A
        ACALL DELAY
        MOV A,#07
        MOV P1, A
        ACALL DELAY
        MOV A,#7F
        MOV P1, A
        ACALL DELAY
        MOV A,#67
        MOV P1, A
        ACALL DELAY
        SJMP AGAIN

DELAY: MOV R1,#FF
BACK:   DJNZ R1, BACK
        RET
        END
```

Alternate program

```
        ORG 0000H// initial starting
                         address
        MOV A,#00H // set a count to
                        point the
                        address of each
                        digit pattern
NEXT:   MOV R0,A
        MOV DPTR,#ARRAY
        MOVC A,@A+DPTR // Read the
                    Digit drive pattern
        MOV P1,A // Move to the port for
                         display
        ACALL DELAY // Calls the delay
        MOV A,R0
        INC A
        SJMP NEXT
```

```
ARRAY: DB 3FH // digit drive pattern for 0
       DB 06H // digit drive pattern for 1
       DB 5BH // digit drive pattern for 2
       DB 4FH // digit drive pattern for 3
       DB 66H // digit drive pattern for 4
       DB 6DH // digit drive pattern for 5
       DB 7DH // digit drive pattern for 6
       DB 07H // digit drive pattern for 7
       DB 7FH // digit drive pattern for 8
       DB 6FH // digit drive pattern for 9

DELAY: MOV R2,#0FFH //subroutine for
                          delay
 WAIT:  DJNZ R2,WAIT
        RET
        END
```

Steps

- Instruction MOVC A,@A+DPTR is the instruction that produces the required digit drive pattern for the display.
- Execution of this instruction will add the value in the accumulator A with the content of the data pointer (starting address of the ARRAY) and will move the data present in the resultant address to A.
- In the program, initial value in A is 00H.
- Execution of MOVC A,@A+DPTR will add 00H to the content in DPTR .
- The result will be the address of label DB 3FH and the data present in this address i.e. 3FH (digit drive pattern for 0) gets moved into the accumulator. Moving this pattern in the accumulator to Port 1 will display 0 which is the first count.

Steps

- At the next count, value in A will advance to 01H and after the execution of MOVC A,@+DPTR ,the value in A will be 06H which is the digit drive pattern for 1 and this will display 1 which is the next count and this cycle gets repeated for subsequent counts.
- Label DB is known as Define Byte – which defines a byte.
- This table defines the digit drive patterns for 7 segment display as bytes (in hex format).
- MOVC operator fetches the byte from this table based on the result of adding DPTR and contents in the accumulator.
- Register R0 is used as a temporary storage of the initial value of the accumulator and the subsequent increments made to accumulator to fetch each digit drive pattern one by one from the ARRAY.