THE 8051 MICROCONTROLLER AND EMBEDDED SYSTEMS

Using Assembly and C

SECOND EDITION

Muhammad Ali Mazidi
Janice Gillispie Mazidi
Rolin D. McKinlay

PEARSON

Authorized adaptation from the United States edition, entitled *The 8051 Microcontroller and Embedded Systems: Using Assembly and C, 2nd Edition,* ISBN: 9780131194021 by Mazidi, Muhammad Ali; Mazidi, Janice Gillispie; McKinlay, Rolin D., published by Pearson Education, Inc., publishing as Prentice Hall
© 2006

ISBN 978-81-317-1026-5

First Impression, 2008
Tenth Impression, 2011
Eleventh Impression, 2012

This edition is manufactured in India and is authorized for sale only in India, Bangladesh, Bhutan, Pakistan, Nepal, Sri Lanka and the Maldives.

Published by Dorling Kindersley (India) Pvt. Ltd., licensees of Pearson Education in South Asia.

Head Office: 7th Floor, Knowledge Boulevard, A-8(A), Sector-62, Noida 201309, UP, India.
Registered Office: 11 Community Centre, Panchsheel Park, New Delhi 110 017, India.

Printed in India by Manipal Technologies Ltd.

CONTENTS AT A GLANCE

CHAPTERS

0: Introduction to Computing
1: The 8051 Microcontrollers
2: 8051 Assembly Language Programming
3: Jump, Loop, and Call Instructions
4: I/O Port Programming
5: 8051 Addressing Modes
6: Arithmetic, Logic Instructions, and Programs
7: 8051 Programming in C
8: 8051 Hardware Connection and Intel Hex File
9: 8051 Timer Programming in Assembly and C
10: 8051 Serial Port Programming in Assembly and C
11: Interrupts Programming in Assembly and C
12: LCD and Keyboard Interfacing
13: ADC, DAC, and Sensor Interfacing
14: 8051 Interfacing to External Memory
15: 8051 Interfacing with the 8255
16: DS12887 RTC Interfacing and Programming
17: Motor Control: Relay, PWM, DC, and Stepper Motors

APPENDICES

A: 8051 Instructions, Timing, and Registers
B: Basics of Wire Wrapping
C: IC Technology and System Design Issues
D: Flowcharts and Pseudocode
E: 8051 Primer for X86 Programmers
F: ASCII Codes
G: Assemblers, Development Resources, and Suppliers
H: Data Sheets

8051 Programming
INTEERUPTS, LCD & LED

Interrupts

1. Develop an Assembly Language Program (ALP) using interrupts to do the following:

(a) Receive data serially and send to P1

(b) Read port 2, transmit data serially, and give a copy to P0

(c) Make timer 0 generate a square wave of 2 KHz frequency on P3.6

Assume that XTAL = 11.0592 MHz. Set the baud rate at 9600.

Solution:

Serial Communication:

Serial baud rate = 9600

8051 UART uses Timer-1 auto-reload.

Baud count to be loaded into serial buffer 28800 / 9600 = -3

Timer:

a. **Desired time delay: ½ KHZ= 0.5 ms; Half time is or 250 µs**

b. **Required Count=** $\frac{\text{desired time delay}}{\text{Machine cycle time}} = \frac{250\ \mu s}{1.085\ \mu s} = 230$

c. **Count to be loaded = 256– desired Count =256-230 = 26**

d. **Convert 26 decimal to hexadecimal =1AH**

Set TH1=1AH ; to generate a time delay

Program:

```
ORG 0000H
LJMP MAIN
ORG 000BH
T0_ISR:  CPL P3.6          ; toggle square wave
         RETI
;---------------- Serial ISR ----------------;
ORG 0023H
SER_ISR:     JNB RI,CHK_TI
             MOV A,SBUF        ; received data
             MOV P1,A          ; copy to P1
             CLR RI
```

```
        CHK_TI:     JNB TI,EXIT_SER
                    CLR TI
                    EXIT_SER: RETI
;---------- Main Program ----------;
ORG 0030H
MAIN:
;---------- Timer0 → square wave--------;
MOV TMOD,#22H          ; T1 mode2, T0 mode2
MOV TH0,#1AH
MOV TL0,#1AH
; Serial setup
MOV SCON,#50H          ; mode1, REN enabled
MOV TH1,#0FDH          ; baud 9600
MOV TL1,#0FDH
; Enable interrupts
MOV IE,#92H            ; EA + ES + ET0
SETB TR0
SETB TR1
MOV A,P2               ; read port 2
MOV P0,A               ; copy to P0
MOV SBUF,A             ; transmit serially
HERE: SJMP HERE
END
```

---

2. Develop an Assembly Language Program (ALP) using interrupts to do the following:

(a) Generate a 10 KHz frequency on P2.1 using T0 8-bit auto reload

(b) Use timer 1 as an event counter to count up 1-Hz pulse and display it on P0. The pulse is connected to EX1. Assume that XTAL = 11.0592 MHz.

Timer:

e. **Desired time delay:** $\frac{1}{10\ KHZ}$ **= 0.1 ms; Half time is or 0.05 ms or 50 µs**

f. **Required Count=** $\frac{\text{desired time delay}}{\text{Machine cycle time}} = \frac{50\ \mu s}{1.085\ \mu s} = 46$

g. **Count to be loaded = 256– desired Count =256-46 = 210**

h. **Convert 210 decimal to hexadecimal =D2H**

Set TH1=D2H ; to generate a time delay

Program:

```
ORG 0000H
LJMP MAIN
ORG 000BH        ; Timer0 Interrupt Vector
LJMP T0_ISR
ORG 0013H        ; External Interrupt1 Vector
LJMP EX1_ISR

ORG 0030H
MAIN: MOV TMOD,#26H          ; T0 mode2 timer, T1 mode2 counter
      MOV TH0,#0D2H          ; Reload value for 10kHz
      MOV TL0,#0D2H
      MOV TH1,#00H           ; Clear counter
      MOV TL1,#00H

      SETB IT1               ; Edge triggered EX1
      MOV IE,#10001110B      ; EA, EX1, ET0 enable
      SETB TR0               ; Start Timer0
      SETB TR1               ; Start Counter1
HERE:  SJMP HERE        ; Infinite loop

;-------------- --; Timer0 ISR : Generate 10kHz;-------
T0_ISR:
  CPL P2.1         ; Toggle P2.1
  RETI
;--------; External Interrupt1 ISR; Count 1Hz pulses;----------
EX1_ISR:
  MOV A,TL1        ; Read counter value
  MOV P0,A         ; Display on Port0
  RETI
END
```

3. With the help of interrupt programming, develop an ALP of 8051 microcontrollers to perform the following tasks; continuously get numeric data from P0 and send it to P1. If the data is even, serially send it through port P2 with a baud rate of 19200. Otherwise, the data will be sent through P2 with a baud rate of 9600.

Solution:

- **Serial baud rate = 9600**
- **8051 UART uses Timer-1 auto-reload.**
- **Baud count to be loaded into serial buffer 28800 / 9600 = -3**

Program:

```
ORG 0000H
LJMP MAIN
;------------ Serial Interrupt -------------
ORG 0023H
SER_ISR:
CLR TI                    ; transmission complete
RETI
;------------ Main Program -----------------
ORG 0030H
MAIN:
MOV TMOD,#20H             ; Timer1 mode-2
MOV TH1,#0FDH
MOV TL1,#0FDH
MOV SCON,#50H             ; serial mode-1
MOV IE,#90H               ; enable serial interrupt
SETB TR1                  ; start timer

LOOP:
MOV A,P0                  ; read data
MOV P1,A                  ; copy to P1
ANL A,#01H          ; check even/odd
JNZ ODD
; EVEN → 19200 baud
SETB PCON.7
```

```
SJMP SEND
ODD:
CLR PCON.7                    ; 9600 baud
SEND:
MOV A,P0
MOV SBUF,A          ; transmit
WAIT: JNB TI,WAIT
SJMP LOOP
END
```

---

4. Write 8051 (AT89C51) ALP for the following logic. The existing values of TCON=01H and IE = 0FFH.

- **Eventhough All interrupts are enabled, Since IT0 = 1, a common program is toggling an LED when INT0 occurs. (External Interrupt 0)**

Program:

```
ORG 0000H
LJMP MAIN

ORG 0003H       ; INT0 interrupt vector
LJMP INT0_ISR

ORG 0030H
MAIN:
   MOV TCON,#01H          ; INT0 edge triggered
   MOV IE,#0FFH           ; Enable all interrupts
   SETB P2.0              ; Initial LED state
HERE: SJMP HERE        ; Wait for interrupt
;----------------------; INT0 Interrupt ISR;----------------------
INT0_ISR:
   CPL P2.0         ; Toggle LED
   RETI
END
```

LCD and LED

1. Develop an 8051 ALP to interface 16x2 LCD display with suitable diagram. Assume the string "VIT" is stored in ROM location from 200H onwards and the string "Vellore" is stored in ROM location from 300H onwards. A switch is connected to port pin P0.4. If the switch input is 1, display the string "VIT" in line 1 of the LCD display otherwise display the string "Vellore" in line 2 of the LCD display.

Program:

```
SW EQU P0.4
ORG 0000H;
----------; LCD INITIALIZATION;----------
MOV DPTR, #CommandPtr        ; Initialize DPTR as a pointer to starting of the  look
                               up table = 0200H
MOV R7, #05H                 ; Initialize R5 with Loop Count of 5 (5 Commands)
MOV R6, #00H                 ; Initialize R6 with Index of 0
C1:
MOV A, R6                    ; A ← Index from R6
MOVC A, @A+DPTR              ; A ← ASCII code from Look Up Table
ACALL Command                ; Call "Command" function withASCII value in A
ACALL Delay
INC R6                       ; Increment the Index
DJNZ R7, C1                  ; Loop till all 5 numbers are displayed

MAIN:
CHECK_SW: JB   SW,SHOW_VIT      ; If switch =1;

----------; SWITCH = 0 → DISPLAY "VELLORE" LINE2;--------

SHOW_VELLORE:    MOV A,#0C0H ;   Line2 address
                 ACALL Command

;----------; SEND DATA FROM LOOKUP TABLE;----------
MOV DPTR, #DataPtr2          ; Initialize DPTR as a pointer to starting of  0300H
MOV R7, #07H                 ; Initialize R5 with Loop Count of 5 (5 Commands)
MOV R6, #00H                 ; Initialize R6 with Index of 0
D2:
MOV A, R6                    ; A ← Index from R6
MOVC A, @A+DPTR              ; A ← ASCII code from Look Up Table
ACALL Dataa                  ; Call "Data" function
```

```
ACALL Delay
INC R6                                  ; Increment the Index
DJNZ R7, D2                             ; Loop till all 14 characters are displayed
SJMP CHECK_SW

;--------------------; SWITCH =1 → DISPLAY "VIT" LINE1;-------------

SHOW_VIT:       MOV A,#80H          ; Line1 address
                ACALL Command

;-------------; SEND DATA FROM LOOKUP TABLE;-------------

MOV DPTR, #DataPtr1             ; Initialize DPTR as a pointer to table = 0200H
MOV R7, #03H                    ; Initialize R5 with Loop Count of 5 (5 Commands)
MOV R6, #00H                    ; Initialize R6 with Index of 0
D1:
MOV A, R6                       ; A ← Index from R6
MOVC A, @A+DPTR                 ; A ← ASCII code from Look Up Table
ACALL Dataa                     ; Call the "Data" function
ACALL Delay
INC R6                          ;Increment the Index
DJNZ R7, D1                     ; Loop till all 14 characters are displayed
SJMP CHECK_SW

Command:                        ;send command to LCD
MOV   P1,A                      ; Put Command value in P1 from A Register
CLR   P2.0                      ; Make Register Select = 0 for Command
CLR   P2.1                      ; Make Read/Write Select = 0 for Write
SETB  P2.2                      ; Make Latch Enable = 1 (High Pulse)
CLR   P2.2                      ;Make Latch Enable = 0 (Falling Edge)
ACALL DELAY                     ;Call a delay of 10 milliseconds
RET

Dataa:                          ;write data to LCD
MOV   P1,A                      ; Put Command value in P1 from A Register
SETB  P2.0                      ;Make Register Select = 0 for Command
CLR   P2.1                      ; Make Read/Write Select = 0 for Write
SETB  P2.2                      ; Make Latch Enable = 1 (High Pulse)
CLR   P2.2                      ;Make Latch Enable = 0 (Falling Edge)
ACALL DELAY                     ;Call a delay of 10 milliseconds
RET

DELAY:  MOV   R3,#50      ;50 or higher for fast CPUs
HERE2:  MOV   R4,#255     ;R4 = 255
HERE:   DJNZ  R4,HERE     ;stay until R4 becomes 0
```

```
DJNZ  R3,HERE2
RET

;-------------------------; LOOKUP TABLES;------------
ORG   0200H
CommandPtr:  DB  38H,0EH,01H,06H,80H        ;Commands Code
DataPtr1:        DB    "VIT"                          ; Data1
ORG   0300H
DataPtr2:        DB    "VELLORE"                  ; Data2
END
```

---

2. Develop an ALP of 8051 for displaying reversely the nine characters string (Your Reg. No) on one line LCD display with Cursor blinking. The HEX value of command words are given below.

S. No	HEX Value	Command to LCD
1	38H	8-bit 2-line 5x7 Dots LCD
2	30H	8-bit 1-line 5x7 Dots LCD
3	06H	Increment cursor
4	04H	Decrement cursor
5	80H	Beginning of First Row
6	8FH	End of First Row
7	0EH	Display on Cursor on
8	01H	Clear cursor

Assume Port2 is used for Data/Command word transferring, Port1.0 is connected to RS of LCD, Port1.1 is connected to WR/ of LCD and Port1.2 is connected to Enable of LCD.

Program:

```
ORG 0000H
----------; LCD INITIALIZATION;----------
MOV DPTR, #CommandPtr        ; Initialize DPTR as a pointer to look  up table =  0400H
MOV R7, #05H                 ; Initialize R5 with Loop Count of 5 (5 Commands)
MOV R6, #00H                 ; Initialize R6 with Index of 0
C1:
MOV A, R6                    ;A←          Index          from          R6
```

```
MOVC A, @A+DPTR          ;A←ASCII code from Look Up Table
ACALL Command            ; Call "Command" function with ASCII value in A
ACALL    Delay                                                    ;
INC R6                   ;Increment the Index
DJNZ R7, C1              ; Loop till all 5 numbers are displayed
;--------------; SET CURSOR TO END;--------------
MOV    A,#8FH
ACALL Command
;----------------; DISPLAY STRING REVERSE;--------------------
MOV DPTR, #RegNo      ; Initialize DPTR as a pointer to 0300H
MOV R7, #09H          ; 9 characters
MOV R6, #08H          ; Start from last character
Reverse:
MOV A, R6             ; A ← Index from R6
MOVC A, @A+DPTR       ; A ← ASCII code from Look Up Table
ACALL Dataa           ; Call the "Data" function register
ACALL Delay
DEC R6                ; Increment the Index
DJNZ R7, Reverse      ; Loop till all 14 characters are displayed
Here: SJMP Here

Command:              ;send command to LCD
MOV   P1,A            ; Put Command value in P1 from A Register
CLR   P2.0            ; Make Register Select = 0 for Command
CLR   P2.1            ; Make Read/Write Select = 0 for Write
SETB  P2.2            ; Make Latch Enable = 1 (High Pulse)
CLR   P2.2            ;Make Latch Enable = 0 (Falling Edge)
ACALL DELAY           ;Call a delay of 10 milliseconds
RET

Dataa:                ;write data to LCD
MOV   P1,A            ; Put Command value in P1 from A Register
SETB  P2.0            ;Make Register Select = 0 for Command
CLR   P2.1            ; Make Read/Write Select = 0 for Write
SETB  P2.2            ; Make Latch Enable = 1 (High Pulse)
CLR   P2.2            ;Make Latch Enable = 0 (Falling Edge)
ACALL DELAY           ;Call a delay of 10 milliseconds
RET

DELAY:  MOV   R3,#50     ;50 or higher for fast CPUs
HERE2:  MOV   R4,#255    ;R4 = 255
```

```
HERE:   DJNZ  R4,HERE   ;stay until R4 becomes 0
DJNZ  R3,HERE2
RET

;-------------------------; LOOKUP TABLES;------------
ORG   0300H
CommandPtr:  DB  38H,0EH,01H,06H,80H    ;Commands Code
RegN0:         DB     "24BCE0450"              ; Data
END
```

---

3. Design an assembly language program in 8051 that alters the displayed message on the LCD based on environmental conditions (e.g., displaying "Hot" if the temperature exceeds a threshold) and displays "Cold" if the temperature is below the threshold. Assume threshold temperature to be 30 degrees celsius.

Program:

```
TEMP EQU P0
ORG 0000H;
------------; LCD INITIALIZATION;------------
MOV DPTR, #CommandPtr        ; Initialize DPTR as a pointer to starting of the  look
                             up table = 0200H
MOV R7, #05H                 ; Initialize R5 with Loop Count of 5 (5 Commands)
MOV R6, #00H                 ; Initialize R6 with Index of 0
C1:
MOV A, R6                    ; A ← Index from R6
MOVC A, @A+DPTR              ; A ← ASCII code from Look Up Table
ACALL Command                ; Call "Command" function withASCII value in A
ACALL Delay
INC R6                       ; Increment the Index
DJNZ R7, C1                  ; Loop till all 5 numbers are displayed

MAIN:
MOV A,TEMP                   ; Read temperature value
MOV B,#1EH                   ; 30°C threshold
CLR C
SUBB A,B                     ; Compare A with 30
JC SHOW_COLD                 ; If Temp <30 → Cold----
```

```
-------; ;------------ DISPLAY HOT ----------------------;--------

SHOW_HOT:       MOV A,#80H ;   Line1 address
                ACALL Command

;-------------; SEND DATA FROM LOOKUP TABLE;-----------
MOV DPTR, #DataPtr2          ; Initialize DPTR as a pointer to starting of  0400H
MOV R7, #03H                 ; Initialize R5 with Loop Count of 3 (HOT)
MOV R6, #00H                 ; Initialize R6 with Index of 0
D1:
MOV A, R6                    ; A ← Index from R6
MOVC A, @A+DPTR              ; A ← ASCII code from Look Up Table
ACALL Dataa                  ; Call "Command" function withASCII value in A
ACALL Delay
INC R6                       ; Increment the Index
DJNZ R7, D1                  ; Loop till all 4 characters are displayed
SJMP MAIN

;--------------------; ;------------ DISPLAY COLD ----------------------;-

SHOW_COLD:      MOV A,#80H          ; Line1 address
                ACALL Command

;-------------; SEND DATA FROM LOOKUP TABLE;-------------

MOV DPTR, #DataPtr1          ; Initialize DPTR as a pointer to table = 0200H
MOV R7, #04H                 ; Initialize R5 with Loop Count of 4 (COLD)
MOV R6, #00H                 ; Initialize R6 with Index of 0
D1:
MOV A, R6                    ; A ← Index from R6
MOVC A, @A+DPTR              ; A ← ASCII code from Look Up Table
ACALL Dataa                  ; Call the "Data" function with the ASCII value in A
ACALL Delay
INC R6                       ;Increment the Index
DJNZ R7, D1                  ; Loop till all 4 characters are displayed
SJMP CHECK_SW

Command:                     ;send command to LCD
MOV   P1,A                   ; Put Command value in P1 from A Register
CLR   P2.0                   ; Make Register Select = 0 for Command
CLR   P2.1                   ; Make Read/Write Select = 0 for Write
SETB  P2.2                   ; Make Latch Enable = 1 (High Pulse)
CLR   P2.2                   ;Make Latch Enable = 0 (Falling Edge)
ACALL DELAY                  ;Call a delay of 10 milliseconds
```

```
RET

Dataa:                          ;write data to LCD
MOV   P1,A                      ; Put Command value in P1 from A Register
SETB  P2.0                      ;Make Register Select = 0 for Command
CLR   P2.1                      ; Make Read/Write Select = 0 for Write
SETB  P2.2                      ; Make Latch Enable = 1 (High Pulse)
CLR   P2.2                      ;Make Latch Enable = 0 (Falling Edge)
ACALL DELAY                     ;Call a delay of 10 milliseconds
RET

DELAY:  MOV   R3,#50      ;50 or higher for fast CPUs
HERE2:  MOV   R4,#255     ;R4 = 255
HERE:   DJNZ  R4,HERE     ;stay until R4 becomes 0
DJNZ  R3,HERE2
RET

;-------------------------; LOOKUP TABLES;------------
ORG   0200H
CommandPtr:  DB  38H,0EH,01H,06H,80H        ;Commands Code
DataPtr1:        DB    "HOT"                 ; Data1

ORG   0300H
DataPtr2:        DB    "COLD"                ; Data2
END
```

Q-3: Write an 8051 ALP to find (compute) the numbers which are divisible by both 3 and 7 in the range 1 to 99 and store them in memory. Interface two seven segment displays in P0 and P1 and display the stored numbers one after other.

- Numbers divisible by **3 and 7 → multiples of 21**
- Numbers within **1–99**:
- `21, 42, 63, 84`
- These numbers are **stored in RAM** and displayed sequentially.

Program:

```
ORG 0000H

MOV R0,#30H          ; RAM location
```

```
MOV A,#21            ; First number
MOV R2,#04H          ; Total numbers

STORE:
MOV @R0,A            ; Store number
ADD A,#21            ; Next multiple
INC R0
DJNZ R2,STORE
;------------------------; Display Numbers;------------------------
MOV R0,#30H
MOV R3,#04H

DISPLAY:  MOV A,@R0
          MOV B,#10
          DIV AB          ; A=tens , B=units

MOV DPTR,#TABLE
MOVC A,@A+DPTR
MOV P1,A                  ; Tens digit

MOV A,B
MOVC A,@A+DPTR
MOV P0,A                  ; Units digit
ACALL DELAY

INC R0
DJNZ R3, DISPLAY
SJMP DISPLAY

;------------------------; Delay;------------------------
DELAY:   MOV R5,#200
D1:  MOV R6,#255
D2: DJNZ R6,D2
DJNZ R5,D1
RET

;------------------------; Seven Segment Table;------------------------
TABLE:
DB 3FH,06H,5BH,4FH,66H
DB 6DH,7DH,07H,7FH,6FH

END
```

8051 Programming
Timers and Counters

1. Write an 8051 based program assuming the crystal frequency is 12 MHz, find the timer register values if we want to have a time delay of 8 ms? Also generate the 8 ms pulse width with equal of ON and OFF period timer 0 and through 16-bit mode of operation.

Solution:

1. Desired time delay: 8 ms
2. Divide the desired time delay by Machine cycle time.

(a) **Machine cycle time=** $\frac{12}{f}=\frac{12}{12\ MHZ}$ **=1 µs**

(b) **Required Count=** $\frac{\text{desired time delay}}{\text{Machine cycle time}}=\frac{8\ ms}{1\ \mu s}=\frac{8000\ \mu s}{1\ \mu s}$**=8000**

3. Count to be loaded = 65535 – desired Count+1
= 65535 – 8000+1 = 57536
4. Convert 57536 decimal to hexadecimal =E0C0H
5. Set TH0=E0H ; TL0=C0H to generate a time delay.

Program:

```
        MOV  TMOD,#10H      ;Timer 1, mode 1(16-bit)
AGAIN:  MOV  TL1,#0C0H      ;TL1=C0H, Low byte
        MOV  TH1,#0E0H      ;TH1=E0H, High byte
        SETB TR1            ;start Timer 1
BACK:   JNB  TF1,BACK       ;stay until timer rolls over
        CLR  TR1            ;stop Timer 1
        CPL  P1.5           ;complement P1.5 to get hi, lo
        CLR  TF1            ;clear Timer 1 flag
        SJMP AGAIN          ;reload timer since mode 1 is not auto-reload
```

---

2. Write an 8051 assembly program to generate a signal at p1.5 with 4ms on time and 8ms off time. Assume clock frequency of 8051 is 12 MHz and use the mode1 timer T0.

Solution:

- **Here, two periods, T_{ON} = 4 ms and T_{OFF} = 8 ms**

  Case (i) For T_{ON}: (Desired time delay 4 ms)
  1. Divide the desired time delay by Machine cycle time.

(a) **Machine cycle time=** $\frac{12}{f}=\frac{12}{12\,MHZ}$ **= 1 µs**

(b) **Required Count=** $\frac{\text{desired time delay}}{\text{Machine cycle time}}=\frac{4000\,\mu s}{1\,\mu s}=4000$

2. Count to be loaded = 65535 – desired Count+1

=65535 – 4000+1 = 61536

3. Convert 61536 decimal to hexadecimal =F060H

4. Set TH0=F0H; TL0=60H to generate a time delay.

Case (ii) For T_{OFF}: (Desired time delay 8 ms)

1. Divide the desired time delay by Machine cycle time.

(c) **Machine cycle time=** $\frac{12}{f}=\frac{12}{12\,MHZ}$ **= 1 µs**

(d) **Required Count=** $\frac{\text{desired time delay}}{\text{Machine cycle time}}=\frac{8000\,\mu s}{1\,\mu s}=8000$

2. Count to be loaded = 65535 – desired Count+1

=65535 – 4000+1 = 57536

3. Convert 57536 decimal to hexadecimal =E0C0H

4. Set TH0=E0H; TL0=C0H to generate a time delay.

Program:

```
ORG 0000H
START:  MOV P1, #00H        ; Clear Port1 initially
MAIN:   SETB P1.5           ; Turn ON signal at P1.5
        ACALL DELAY_ON      ; Call 4 ms delay
        CLR P1.5            ; Turn OFF signal at P1.5
        ACALL DELAY_OFF     ; Call 8 ms delay
        SJMP MAIN           ; Repeat forever

---------; 4 ms Delay using Timer0;-----------
DELAY_ON: MOV TMOD, #01H    ; Timer0 Mode1 (16-bit)
          MOV TH0, #0F0H    ; Load high byte
          MOV TL0, #0B0H    ; Load low byte
          SETB TR0          ; Start Timer0
WAIT1:  JNB TF0, WAIT1      ; Wait until overflow
        CLR TR0             ; Stop Timer0
        CLR TF0             ; Clear overflow flag
        RET
----------------; 8 ms Delay using Timer0;-------------
DELAY_OFF: MOV TMOD, #01H   ; Timer0 Mode1 (16-bit)
           MOV TH0, #0E0H   ; Load high byte
           MOV TL0, #0C0H   ; Load low byte
           SETB TR0         ; Start Timer0
WAIT2:  JNB TF0, WAIT2      ; Wait until overflow
        CLR TR0             ; Stop Timer0
        CLR TF0             ; Clear overflow flag
        RET
END
```

3. Write an 8051 (AT89C51) ALP to generate a periodic waveform simultaneously as follows: i) 0.1ms ON & 0.1ms OFF ii) 0.2ms ON & 0.2ms OFF using timer. Don't access TL0/1 register in the ALP. Provide necessary timer calculations.

Solution:

Waveform 1

- **ON = 0.1 ms = 100 µs**
  OFF = 0.1 ms = 100 µs

Waveform 2

- **ON = 0.2 ms = 200 µs**
  OFF = 0.2 ms = 200 µs

We use Mode-2 (8-bit auto reload) so TL registers are never touched.

Assume standard 8051 crystal: fosc=12 MHz

(a) Machine cycle time= $\frac{12}{f}=\frac{12}{12\,MHZ}$ **=1 µs**

For Waveform 1: Use Timer 0

a. **Desired time delay: 0.1 ms or 100 µs**

b. **Required Count=** $\frac{\text{desired time delay}}{\text{Machine cycle time}}=\frac{100\ \mu s}{1\ \mu s}=100$

c. **Count to be loaded = 256– desired Count =256-100 = 156**

d. **Convert 156 decimal to hexadecimal =9CH**

e. **Set TH0=9CH ; to generate a time delay.**

For Waveform 2: Use Timer 1

a. **Desired time delay: 0.2 ms or 200 µs**

b. **Required Count=** $\frac{\text{desired time delay}}{\text{Machine cycle time}}=\frac{200\ \mu s}{1\ \mu s}=200$

c. **Count to be loaded = 256– desired Count =256-200 = 56**

d. **Convert 156 decimal to hexadecimal =38H**

e. **Set TH1=38H ; to generate a time delay.**

Program:

```
ORG 0000H

START:  MOV P1, #00H            ; Clear Port1 initially
        MOV TMOD, #22H          ; Timer0 & Timer1 in Mode2 (8-bit auto-reload)

MAIN:
  ------   ; Signal 1: P1.0 (0.1ms ON/OFF) ;----------
    SETB P1.0                   ; ON
```

```
        ACALL DELAY_T0          ; 0.1 ms delay
        CLR P1.0                ; OFF
        ACALL DELAY_T0          ; 0.1 ms delay

        ;--------; Signal 2: P1.1 (0.2ms ON/OFF) ;------------
        SETB P1.1               ; ON
        ACALL DELAY_T1          ; 0.2 ms delay
        CLR P1.1                ; OFF
        ACALL DELAY_T1          ; 0.2 ms delay
        SJMP MAIN               ; Repeat forever

;----------; Timer0 → 0.1 ms delay;-------------
DELAY_T0:
        MOV TH0, #9CH           ; High byte
        SETB TR0                ; Start Timer0
WAIT0:  JNB TF0, WAIT0          ; Wait for overflow
        CLR TR0                 ; Stop Timer0
        CLR TF0                 ; Clear flag
        RET

;--------; Timer1 → 0.2 ms delay;-----------------
DELAY_T1:
        MOV TH1, #038H          ; High byte
          SETB TR1              ; Start Timer1
WAIT1:  JNB TF1, WAIT1           ; Wait for overflow
        CLR TR1                  ; Stop Timer1
        CLR TF1                 ; Clear flag
        RET
END
```

---

4. Write an 8051 (AT89C51) ALP to generate a pulse waveform with frequency 2KHz using timer. Don't access TL0/1 register in the ALP. Provide the necessary timer calculations.

Frquency= 2KHZ.

- **T = 1 / f = 1 / 2 kHz = 0.5 ms the period of the square wave.**
- **1/2 of it for the high and low portions of the pulse is 0.25 ms or 250 µs.**

Assume standard 8051 crystal: fosc=11.0592 MHz

(b) Machine cycle time= $\frac{12}{f}=\frac{12}{11.0592\ MHZ}$ **=1.085 µs**

For Waveform 1: Use Timer 0

a. **Desired time delay: 0.25 ms or 250 µs**

b. **Required Count= $\frac{\text{desired time delay}}{\text{Machine cycle time}}=\frac{250\ \mu s}{1.085\ \mu s}$=230**

c. **Count to be loaded = 256– desired Count =256-230 = 26**

d. **Convert 156 decimal to hexadecimal =1AH**

e. **Set TH0=1AH ; to generate a time delay.**

Program:

```
ORG 0000H

START:  MOV P1, #00H            ; Clear Port1 initially
        MOV TMOD, #02H          ; Timer0 in Mode2 (8-bit auto-reload)

MAIN:
   ------   ; Signal 1: P1.0 (0.25 ms ON/OFF) ;----------
      SETB P1.0                 ; ON
      ACALL DELAY_T0            ; 0.1 ms delay
      CLR P1.0                  ; OFF
      ACALL DELAY_T0            ; 0.1 ms delay

;----------; Timer0 → 0.25 ms delay;-------------
DELAY_T0:
      MOV TH0, #1AH             ; High byte
      SETB TR0                  ; Start Timer0
WAIT0:  JNB TF0, WAIT0          ; Wait for overflow
      CLR TR0                   ; Stop Timer0
      CLR TF0                   ; Clear flag
      RET
END
```

---

5. The car parking area can accommodate a maximum of 100 cars. Assume that a sensor is connected to P3.4 of 8051 microcontroller to sense the car entering the parking area. If the count reaches 100, the microcontroller should turn "ON" a buzzer connected to P1.7 for 1 second. Assume the crystal frequency as 22 MHz. Write a suitable 8051 assembly code to implement.

Solution:

A sensor at **P3.4** detects a car entering.

- Each detection → increment car count.

- When count = **100 cars**:
- Turn **ON buzzer (P1.7)** for **1 second**
- Turn buzzer OFF
- Reset counter (or stop counting — here we reset)

Timing Design :

Crystal frequency:

fosc = 22 MHz

$$Machine\ Cyle\ time = \frac{12}{f} = \frac{12}{22 \times 10^6} = 0.545\ \mu s$$

Assume ~10 ms timer overflow (desired Time delay)

Timer counts needed= $\frac{\textbf{\textit{desired Time delay}}}{Machine\ Cyle\ time} = \frac{10\ ms}{0.545\ \mu s} = \frac{10000\ \mu s}{0.545\ \mu s} = 18349$

Timer count preload:

65536−18349= 47187= **B8 53 H**

So:

TH1 = B8 H
TL1 = 53 H

Repeat this overflow $\frac{1\ s}{10\ ms}$ = **100 times ≈ 1 second**.

8051 Assembly Program

```
ORG 0000H                    Initialization
MOV TMOD,#10H                Timer1 mode 1 (16-bit)
MOV R0,#00                   Car counter = 0
MAIN:
WAIT_LOW:  JNBP3.4,WAIT_LOW  wait until sensor HIGH
WAIT_HIGH: JB P3.4,WAIT_HIGH ; wait until sensor LOW
INC R0                       ; car count++
MOV A,R0                     ; Check if count = 100
CJNE A,#100,MAIN

                             ; BUZZER ON FOR 1 SECOND
```

```
SETB P1.7                          ; buzzer ON
MOV R2,#100                        ; loop counter
DELAY_LOOP: MOV TH1,#B8H
            MOV TL1,#53H
            SETB TR1
WAIT_T1: JNB TF1,WAIT_T1
         CLR TR1
         CLR TF1
         DJNZ R2,DELAY_LOOP
CLR P1.7                           ; buzzer OFF
MOV R0,#00                         ; reset car count
SJMP MAIN
END
```

Try:

6. A football stadium is designed with a seating capacity of precisely 1000 persons. It is assumed that there is just one entry and exit gate accessible within the stadium. The entry gate is configured so that whenever a person enters a stadium, a high-to-low pulse will be generated and that will be given to the port3.4 of the 8051 microcontroller. The stepper motor regulates the opening and shutting of the entry gate. To open/close the entry gate, send a logic 1/logic 0 to 10 port0.5 of the 8051 microcontroller, which connects to the stepper motor. Develop an ALP of 8051 that counts and allows maximum up to 1000 persons and closes the gate after that irrespective of persons leaving from the exit gate. Use the mode 2 timer programming.

---

7. An 8051 microcontroller based system is designed to count the students entering into a class room of capacity 60, where an enable switch is connected to the microcontroller port which helps the faculty to start the counting process. After 3 minutes of the counting process, a motor which is connected to the P2.1 should be turned 'ON' to close the door by sending 'HIGH' value to P2.1. Also the final count value after 3 minutes should be sent to P1. Develop an ALP for the above scenario.

Solution:

Assumption

- Timer (for 3-minute timing
- Counter (to count students)
  - Timer-0 → time measurement
  - Timer-1 → external counter (student entries)
  - Enable switch → P3.0
  - Motor (door close) → P2.1
  - Count output → Port-1

Timing design:

- Assume standard crystal = **11.0592 MHz**
- Crystal frequency:
- fosc = 11.0592 MHz
- $Machine\ Cyle\ time = \frac{12}{f} = \frac{12}{11.0592 \times 10^6} = 1.0850\ \mu s$

Assume , 50 ms timer overflow (desired Time delay)

- Timer counts needed= $\frac{\boldsymbol{desired\ Time\ delay}}{Machine\ Cyle\ time} = \frac{50\ ms}{0.545\ \mu s} = \frac{50000\ \mu s}{1.085\ \mu s} = 46080$

Timer count preload:

- 65536−46080 = 19456 = **4C 00 H**

So: **TH1 = 4C H; TL1 = 00 H**

For 3 minutes,

3 min = 180 s

180 s / 0.05 s = 3600 loops

8051 Assembly Program

```
ORG 0000H                          Initialization
MOV TMOD,#51H                       ; T1 → mode1 counter
                                   T0 → mode1 timer
CLR P2.1                           ; motor OFF
WAIT_EN: JB P3.0,WAIT_EN           WAIT_EN: JB P3.0,WAIT_EN
SETB TR1                           ; start counter
MOV R5,#14      ; outer loop       ; 3-minute loop counter
MOV R6,#240     ; inner loop
TIME_LOOP:MOV TH0,#4CH             ;  50 ms timer delay
          MOV TL0,#00H

SETB TR0
WAIT_T0: JNB TF0,WAIT_T0
CLR TR0
CLR TF0
DJNZ R6,TIME_LOOP                  ;--------------------------------
DJNZ R5,TIME_LOOP                  ; Stop counting after 3 minutes
                                   ;--------------------------------
CLR TR1
MOV A,TL1                          ; read student count
MOV P1,A
SETB P2.1                          ; close door motor
HERE: SJMP HERE
END
```

Try:

8. Assume that two sensors are connected to P3.4 and P3.5 of the 8051 Microcontroller. Sensor connected to P3.4 is placed near entry door of the auditorium and another sensor is connected to P3.5 is placed near the exit door. The maximum capacity of the auditorium is 200. Write an ALP to find the number of people inside the auditorium and send the count to port P0. An LED connected to P1.7 will glow if the auditorium is full

---

9. Develop an 8051 ALP to generate 10 KHz frequency using timer 0 in order to blink the serial lamp (connected to P2.1) used to decorate the hall. Imagine a sensor is connected with the pin T1 to count the number of entries inside

the hall. For this one can feed the input by connecting a switch at pin T1 in which each turning off of switch indicates the number of incoming persons entering into the hall. Count the number of persons & display the value on port 0. When the count becomes FFH then turn on 8 LEDs connected to P0.

Solution: Assumption

Part-1 → Lamp blinking (Timer-0)

- Generate **10 kHz square wave** on **P2.1**
- Timer-0 used for delay

Part-2 → Entry counter (Timer-1)

- T1 pin counts incoming persons
- When counter reaches **FFH**:
  - Turn ON all LEDs on **Port-0**

Timing design:

Assume standard crystal = **11.0592 MHz**

Crystal frequency:

fosc = 11.0592 MHz

$$Machine\ Cyle\ time = \frac{12}{f} = \frac{12}{11.0592\ \times 10^6} = 1.0850\ \mu s$$

Given 10 kHz square wave, thereore time = $\frac{\mathbf{1}}{\mathbf{10\ KHz}} = \mathbf{0.1}\ \boldsymbol{ms}$

But for 50 % duty cycle, ~0.05 ms only timer overflow (desired Time delay)

Timer counts needed= $\frac{\boldsymbol{desired\ Time\ delay}}{Machine\ Cyle\ time} = \frac{0.05\ ms}{0.545\ \mu s} = \frac{50\ \mu s}{1.085\ \mu s} = 46$

Timer count preload:

65536−46= 65490= **FF D2 H**

So:

TH1 = FF H

TL1 = D2 H

We run both timers continuously:

- Timer-0 → blink lamp
- Timer-1 → hardware counter

Eg: Toggle lamp → check counter → update display

```
ORG 0000H                    Initialization
MOV TMOD,#51H                ; T0 → Mode1 Timer (10kHz generation)
                             ; T1 → Mode1 Counter (Entry counting)
SETB TR1                     ; start counter (T1 pin)
MAIN:                        ; Generate 10 kHz blink
MOV TH0,#0FFH
MOV TL0,#0D2H
SETB TRO
WAIT0: JNB TF0,WAIT0
CLR TR0
CLR TF0
CPL P2.1                     ; toggle lamp
                             ; Read counter value
MOV A,TL1                    ; read low byte
MOV P0,A                     display count
CJNE A,#0FFH,MAIN            Check for FFH
MOV P0,#0FFH                 ; count reached FF → LEDs ON
SJMP MAIN
END
```

---

10. Let an input device is connected to P1 of 8051 Microcontroller. Develop an ALP that continuously monitors the MSB and LSB pins of P1 and based on the binary combinations generate square waveforms as given in table 1.

S.No	MSB	LSB	Frequency	Baud rate
1	0	0	500 Hz	1200
2	0	1	1.6 KHz	2400
3	1	0	750 Hz	4800
4	1	1	2 KHz	9600

```
ORG 0000H

MOV TMOD,#01H       ; Timer0 Mode1 (16-bit)
MAIN: MOV A,P1
      MOV C,ACC.7       ; Get MSB
      MOV R0,A
      ANL R0,#01H       ; Get LSB
```

```
        JNC MSB0            ; If MSB=0
;-----------------------; MSB = 1;-----------------------------
        CJNE R0,#00H,FREQ_2K
-------------; MSB=1, LSB=0 → 750 Hz----------------------
FREQ_750:
        MOV TH0,#0FDH
        MOV TL0,#09AH
        ACALL TIMER_DELAY
        CPL P2.0
        SJMP MAIN

FREQ_2K:
        MOV TH0,#0FFH
        MOV TL0,#01AH
        ACALL TIMER_DELAY
        CPL P2.0
        SJMP MAIN

;----------------; MSB = 0;-----------------------------
MSB0:CJNE R0,#00H,FREQ_16K

------; MSB=0, LSB=0 → 500 Hz-------
FREQ_500:
        MOV TH0,#0FCH
        MOV TL0,#066H
        ACALL TIMER_DELAY
        CPL P2.0
        SJMP MAIN

FREQ_16K:
        MOV TH0,#0FEH
        MOV TL0,#0E0H
        ACALL TIMER_DELAY
        CPL P2.0
        SJMP MAIN
;------------------; TIMER0 DELAY;----------------------------
TIMER_DELAY:
        SETB TR0
WAIT: JNB TF0,WAIT
        CLR TR0
        CLR TF0
        RET

END
```

Serial Communication

1. Develop a program to receive data until the received data is "N", means if 8051 receives "N" then it should stop receiving the character. Baud rate of 9600.

Code:

```
ORG 0000H
;-------------; Initialize Serial Communication;----------------
MOV TMOD,#20H       ; Timer1 Mode2 (8-bit auto reload)
MOV TH1,#0FDH       ; 9600 baud rate
MOV SCON,#50H       ; Mode1, REN=1 (Receive Enable)
SETB TR1            ; Start Timer1
;-----------------; Receive Loop;----------
RECEIVE:    JNB RI,RECEIVE      ; Wait until character received
            MOV A,SBUF          ; Move received data to A
CLR RI              ; Clear receive interrupt flag
CJNE A,#'N',RECEIVE  ; If received data ≠ 'N', continue receiving

---------; If 'N' received → stop program------
STOP: SJMP STOP          ; Stay here forever
END
```

---

2. An 8051 microcontroller operating with crystal frequency of 11.0592 MHz is connected to a switch at P1.0. Write an ALP to monitor the status of switch for every 1ms after successful transmission of the message. If the switch is open, then transmit a string "PLACE TO LEARN" stored at ROM location of 200H at 9600 baud rate. If the switch is closed, then transmit a string "CHANCE TO GROW" stored at ROM location 400H at 4800 baud rate. Use timer for the delay generation. Assume the overhead delay caused due to transmission of string is negligible.

Program:

```
SW1     EQU P1.0

ORG 0000H
```

```
MAIN: MOV TMOD,#21H          ; T1 Mode2 (baud), T0 Mode1 (delay)

CHECK_SW:  JB SW1,SW_OPEN        ; If SW=1 (open)

;------------; SWITCH CLOSED → 4800 BAUD → "CHANCE TO GROW";--------------------
SW_CLOSED:
      MOV TH1,#0FAH         ; 4800 baud rate
      MOV SCON,#50H
      SETB TR1
      MOV DPTR,#MSG2
      ACALL SEND_STRING
      ACALL DELAY_1MS
      SJMP CHECK_SW
;-----------; SWITCH OPEN → 9600 BAUD → "PLACE TO LEARN";-------------------------
SW_OPEN:
      MOV TH1,#0FDH         ; 9600 baud rate
      MOV SCON,#50H
      SETB TR1
      MOV DPTR,#MSG1
      ACALL SEND_STRING
      ACALL DELAY_1MS
      SJMP CHECK_SW

;----------; SEND STRING SUBROUTINE;-----------------------------------------------
SEND_STRING:
NEXT_CHAR:
      CLR A
      MOVC A,@A+DPTR
      JZ EXIT_SEND
      ACALL SEND_CHAR
      INC DPTR
      SJMP NEXT_CHAR
      RET
;--------------------------- --; SEND SINGLE CHARACTER;---------------------------------------
SEND_CHAR:  MOV SBUF,A
WAIT_TI:   JNB TI,WAIT_TI
      CLR TI
      RET
;---------- ----------------; 1ms DELAY USING TIMER0;-----------------------------------------
DELAY_1MS:
      MOV TH0,#0FCH         ; Load for 1ms delay
      MOV TL0,#066H
      SETB TR0
WAIT_TF0:  JNB TF0,WAIT_TF0
```

```
      CLR TR0
      CLR TF0
      RET
;---------------; STRING STORAGE;-----------------------------------------------
ORG 0200H
MSG1:   DB "PLACE TO LEARN"

ORG 0400H
MSG2:   DB "CHANCE TO GROW"

END
```

---

3. Develop an ALP for the following scenario: Get an 8 bit data" X" from port P0 and another 8 bit data "Y" from port P1. Perform "X-Y" and send the result through serial port at 19200 baud rate for 150 times.

Solution:

```
ORG 0000H
MAIN: MOV TMOD,#20H          ; Timer1 Mode2 (8-bit auto reload)
        MOV TH1,#0FDH        ; Load for 19200 baud
        MOV SCON,#50H        ; Mode1, REN=1
        SETB PCON.7          ; SMOD = 1 (Double baud rate)
        SETB TR1              ; Start Timer1
        MOV R7,#96H          ; 150 decimal = 96H
AGAIN: MOV A,P0              ; Read X from Port 0
         MOV B,P1            ; Read Y from Port 1
        CLR C
        SUBB A,B             ; A = X - Y
        ACALL  SEND_SERIAL   ; Send result
        DJNZ R7,AGAIN        ; Repeat 150 times
STOP:   SJMP STOP
;---------------; SERIAL TRANSMIT SUBROUTINE;-----------------------------------
SEND_SERIAL:  MOV SBUF,A      ; Load result to serial buffer
WAIT_TI:   JNB TI,WAIT_TI     ; Wait until transmission complete
             CLR TI           ; Clear transmit flag
             RET
END
```

4. Two switches (SW1 and SW2) are connected to P1.2 and P1.3 of 8051 microcontroller. Develop an ALP to monitor the status of switches and transmit the following messages with the given baud rate.

S.No	SW1	SW2	Message to be transmitted serially	Baud rate
1	OFF	OFF	OFF	1200
2	OFF	ON	SWITCH2	2400
3	ON	OFF	SWITCH1	4800
4	ON	ON	ON	9600

```
SW1 EQU P1.2
SW2 EQU P1.3
ORG 0000H
MAIN:  MOV TMOD,#20H      ; timer1 auto reload
MOV SCON,#50H      ; serial mode
SETB TR1
LOOP: JB SW1,SW1_ON   ;---- Check switches ----
JB SW2,CASE01                    ; 0 0
MOV TH1,#-24
MOV DPTR,# MESS1
SJMP SEND
SW1_ON: JB SW2,CASE11       ; 1 0
MOV TH1,#-12
MOV DPTR,# MESS2
SJMP SEND
CASE01: MOV TH1,#0F4H        ; 0 1
MOV DPTR,# MESS3
SJMP SEND
CASE11:  MOV TH1,#0FDH   ; 1 1
MOV DPTR,# MESS4
;---- Send message ----
SEND:  CLR A
NEXT: MOVC A,@A+DPTR
```

```
JZ LOOP
MOV SBUF,A
WAIT: JNB TI,WAIT
CLR TI
INC DPTR
CLR A
SJMP NEXT
;---- Messages ----
MESS1: DB "OFF",0
MESS2: DB "SWITCH2",0
MESS 3: DB "SWITCH1",0
MESS 4: DB "ON",0
END
```

Chapter 1 : LCD Interfacing (16×2 LCD)

1.1 Introduction to LCD

LCD (Liquid Crystal Display) is widely used in embedded systems as it replaces LEDs for displaying alphanumeric data. The reasons for its widespread use are:

- Declining prices of LCD
- Ability to display numbers, characters, and graphics
- Built-in refreshing controller — relieves the CPU of the task of refreshing the LCD
- Ease of programming for characters and graphics
- 16×2 LCD can display 16 characters per row across 2 rows
- Each character is displayed in a 5×7 dot matrix space

1.2 Pin Description of 16×2 LCD

Pin	Symbol	I/O	Function / Description
1	VSS	—	Ground (0V)
2	VCC	—	+5V Power Supply
3	VEE	—	Contrast control voltage (0V = max contrast)
4	RS	I	Register Select: RS=0 → Command register; RS=1 → Data register
5	R/W	I	Read/Write: R/W=0 → Write to LCD; R/W=1 → Read from LCD
6	E	I/O	Enable pin — LCD latches data/command on H→L edge
7–14	DB0–DB7	I/O	8-bit bidirectional data bus (DB7 = MSB, DB0 = LSB)

1.3 LCD Instruction / Command Code Table

To send a command: set RS=0, R/W=0, put command byte on data bus, then give a High-to-Low pulse on the Enable (E) pin. To send data (a character): RS=1, R/W=0, then pulse E High→Low.

Code (Hex)	Command Sent to LCD Instruction Register
01H	Clear display screen
02H	Return home (cursor to position 0,0; display unchanged)
04H	Decrement cursor — shift cursor to left after each write
06H	Increment cursor — shift cursor to right after each write
05H	Shift entire display to the right
07H	Shift entire display to the left
08H	Display OFF, cursor OFF
0AH	Display OFF, cursor ON
0CH	Display ON, cursor OFF

0EH	Display ON, cursor ON (visible, not blinking)
0FH	Display ON, cursor ON and blinking
10H	Shift cursor position to the left
14H	Shift cursor position to the right
18H	Shift the entire display to the left
1CH	Shift the entire display to the right
38H	Initialize: 2 lines, 5×7 dot matrix font
80H	Force cursor to beginning of 1st line (DDRAM address 00H)
C0H	Force cursor to beginning of 2nd line (DDRAM address 40H)
81H–8FH	Cursor positions 1–15 on Line 1
C1H–CFH	Cursor positions 1–15 on Line 2

DDRAM Address Map for 16×2 LCD:
Line 1 positions: 80H 81H 82H 83H 84H 85H 86H 87H 88H 89H 8AH 8BH 8CH 8DH 8EH 8FH
Line 2 positions: C0H C1H C2H C3H C4H C5H C6H C7H C8H C9H CAH CBH CCH CDH CEH CFH

1.4 Hardware Connection (8051 ↔ LCD)

8051 Pin	LCD Pin	Signal	Purpose
P1.0 – P1.7	DB0 – DB7	Data Bus	Send 8-bit command or character code
P2.0	RS (Pin 4)	Register Select	0 = command mode, 1 = data mode
P2.1	R/W (Pin 5)	Read/Write	Always 0 (write) during normal operation
P2.2	E (Pin 6)	Enable	High-to-Low pulse latches data into LCD
VCC	VCC + 10kΩ Pot → VEE	Power / Contrast	Adjust pot for optimum contrast

1.5 Key Subroutines Used in All LCD Programs

Every LCD program uses these two subroutines and a time-delay routine. Understanding them is essential before reading the examples.

```
;============================================================
; COMNWRT – Send a COMMAND byte (in register A) to LCD
;============================================================
COMNWRT:
MOV P1, A ; put command byte on data bus (P1)
CLR P2.0 ; RS = 0 → command register selected
CLR P2.1 ; R/W = 0 → write operation
SETB P2.2 ; E = 1 → rising edge (≥ 450 ns pulse width)
ACALL DELAY ; short delay ≥ 450 ns
CLR P2.2 ; E = 0 → falling edge latches the command
RET
```

```
;============================================================
; DATAWRT – Send a DATA byte (ASCII character in A) to LCD
;============================================================
DATAWRT:
MOV P1, A ; put character code on data bus
SETB P2.0 ; RS = 1 → data register selected
CLR P2.1 ; R/W = 0 → write operation
SETB P2.2 ; E = 1 → rising edge
ACALL DELAY
CLR P2.2 ; E = 0 → falling edge latches the character
RET

;============================================================
; DELAY – Software time delay (approx. several ms)
;============================================================
DELAY:
MOV R3, #50 ; outer loop count (increase for slower CPUs)
HERE2:
MOV R4, #255 ; inner loop count
HERE:
DJNZ R4, HERE ; decrement R4 until zero
DJNZ R3, HERE2 ; decrement R3 until zero
RET
```

■ Example 1 : Display 'NO' on Line 1, Position 4 (Time-Delay Method)

Write a complete 8051 assembly program to display the string **"NO"** on a 16×2 LCD starting at **Line 1, Position 4** using the time-delay method.
Hardware: P1.0–P1.7 → DB0–DB7, P2.0 → RS, P2.1 → R/W, P2.2 → E

✔ Solution :

Initialization sequence: 38H (2-line 5×7) → 0EH (display ON, cursor ON) → 01H (clear) → 06H (auto-increment cursor) → 84H (Line 1, position 4) → then send characters.

```
;============================================================
; Display 'NO' on Line 1, Position 4 of 16x2 LCD
; P1 = data bus | P2.0=RS | P2.1=R/W | P2.2=E
;============================================================
ORG 0H

; --- Step 1: Initialize LCD ---
MOV A, #38H ; Function set: 2 lines, 5x7 dot matrix
ACALL COMNWRT ; send command
ACALL DELAY

MOV A, #0EH ; Display ON, cursor ON (not blinking)
ACALL COMNWRT
ACALL DELAY

MOV A, #01H ; Clear display screen
ACALL COMNWRT
ACALL DELAY

MOV A, #06H ; Entry mode: increment cursor, no display shift
```

```
ACALL COMNWRT
ACALL DELAY

; --- Step 2: Set cursor position = Line 1, Position 4 ---
; DDRAM address for Line 1, pos 4 = 00H + 4 = 04H
; Set DDRAM address command = 80H | address = 80H | 04H = 84H
MOV A, #84H ; cursor → Line 1, position 4
ACALL COMNWRT
ACALL DELAY

; --- Step 3: Send characters ---
MOV A, #'N' ; ASCII code of 'N' = 4EH
ACALL DATAWRT
ACALL DELAY

MOV A, #'O' ; ASCII code of 'O' = 4FH
ACALL DATAWRT
ACALL DELAY

AGAIN: SJMP AGAIN ; infinite loop – stay here

;=============================================================
; COMNWRT – Send command byte (A) to LCD
;=============================================================
COMNWRT:
MOV P1, A ; command on data bus
CLR P2.0 ; RS=0 (command)
CLR P2.1 ; R/W=0 (write)
SETB P2.2 ; E=1
ACALL DELAY
CLR P2.2 ; E=0 → latch
RET

;=============================================================
; DATAWRT – Send data byte (A) to LCD
;=============================================================
DATAWRT:
MOV P1, A ; character on data bus
SETB P2.0 ; RS=1 (data)
CLR P2.1 ; R/W=0 (write)
SETB P2.2 ; E=1
ACALL DELAY
CLR P2.2 ; E=0 → latch
RET

;=============================================================
; DELAY – Software time delay
;=============================================================
DELAY:
MOV R3, #50
HERE2: MOV R4, #255
HERE: DJNZ R4, HERE
DJNZ R3, HERE2
RET
END
```

Command 84H explained: The 'Set DDRAM Address' command = 80H | (cursor address).
Line 1 starts at address 00H. Position 4 means address = 04H.
So command = 80H | 04H = **84H** (cursor placed at column 4 of line 1).

■ Example 2 : Display 'HELLO' using MOVC Look-up Table Method

Write a complete 8051 assembly program to display **'HELLO'** on a 16×2 LCD using the **MOVC look-up table method**. Store LCD commands in MYCOM and the message string in MYDATA at code memory address 300H. Cursor starts at position 6 of line 1.

✔ Solution :

In this method, commands and data are stored as byte tables in code memory. MOVC fetches each byte using DPTR. A null byte (00H) marks the end of each table. The JZ instruction detects this null and jumps to the next phase.

```
;============================================================
; Display 'HELLO' on LCD using MOVC look-up table
; P1=data bus | P2.0=RS | P2.1=R/W | P2.2=E
;============================================================
ORG 0

; --- Phase 1: Send all initialization commands ---
MOV DPTR, #MYCOM ; DPTR points to command table
C1: CLR A ; A = 0 (offset = 0)
MOVC A, @A+DPTR ; fetch byte from code memory
JZ SEND_DAT ; if byte = 0 (null), go to data phase
ACALL COMNWRT ; send command to LCD
ACALL DELAY
INC DPTR ; advance to next command
SJMP C1 ; loop

; --- Phase 2: Send character data ---
SEND_DAT:
MOV DPTR, #MYDATA ; DPTR points to data table
D1: CLR A
MOVC A, @A+DPTR ; fetch character
JZ AGAIN ; null = end of string
ACALL DATAWRT ; send character to LCD
ACALL DELAY
INC DPTR
SJMP D1

AGAIN: SJMP AGAIN ; stay here forever

;============================================================
; COMNWRT – Command write
;============================================================
COMNWRT:
MOV P1, A
CLR P2.0 ; RS=0
CLR P2.1 ; R/W=0
SETB P2.2 ; E=1
ACALL DELAY
CLR P2.2 ; E=0 latch
RET

;============================================================
; DATAWRT – Data write
;============================================================
DATAWRT:
```

```
MOV P1, A
SETB P2.0 ; RS=1
CLR P2.1 ; R/W=0
SETB P2.2 ; E=1
ACALL DELAY
CLR P2.2 ; E=0 latch
RET

;============================================================
; DELAY – Time delay
;============================================================
DELAY:
MOV R3, #250
HERE2: MOV R4, #255
HERE: DJNZ R4, HERE
DJNZ R3, HERE2
RET

;============================================================
; Look-up Tables at code memory address 300H
;============================================================
ORG 300H

; Command table:
; 38H = 2-line 5x7 init
; 0EH = display ON, cursor ON
; 01H = clear display
; 06H = increment cursor (auto-right)
; 86H = cursor to Line 1, Position 6 (80H + 06H)
; 00H = null terminator
MYCOM: DB 38H, 0EH, 01H, 06H, 86H, 0

; Data table: ASCII codes + null terminator
MYDATA: DB 'H', 'E', 'L', 'L', 'O', 0
END
```

How MOVC works: MOVC A, @A+DPTR reads from *code memory* at address (A + DPTR). Since A is cleared (CLR A) before each fetch, it reads exactly from the address in DPTR. After reading, DPTR is incremented to point to the next byte. The null byte (00H) at the end acts as a string terminator — JZ detects A=0.

■ Practice Problem : Display 'SCOPE' on Line 2 (MOVC Method)

Write a complete 8051 assembly program to display **'SCOPE'** on the **second line** of a 16×2 LCD using the MOVC look-up table method.
Hardware: P1 = data bus, P2.0 = RS, P2.1 = R/W, P2.2 = E

✔ Solution :

To position the cursor at the start of Line 2, use command **C0H** (DDRAM address 40H → command = 80H | 40H = C0H). The initialization sequence is the same as before; only the cursor-position command and the data string differ.

```
;============================================================
; Display 'SCOPE' on Line 2 of 16x2 LCD – MOVC method
; P1=data bus | P2.0=RS | P2.1=R/W | P2.2=E
;============================================================
ORG 0H

; --- Phase 1: Send commands ---
MOV DPTR, #MYCOM
C1: CLR A
MOVC A, @A+DPTR ; fetch command byte
JZ SEND_DAT ; 00H = end of commands
ACALL COMNWRT
ACALL DELAY
INC DPTR
SJMP C1

; --- Phase 2: Send data ---
SEND_DAT:
MOV DPTR, #MYDATA
D1: CLR A
MOVC A, @A+DPTR ; fetch character
JZ AGAIN ; 00H = end of string
ACALL DATAWRT
ACALL DELAY
INC DPTR
SJMP D1

AGAIN: SJMP AGAIN

;============================================================
; COMNWRT
;============================================================
COMNWRT:
MOV P1, A
CLR P2.0 ; RS=0 (command)
CLR P2.1 ; R/W=0 (write)
SETB P2.2 ; E=1
ACALL DELAY
CLR P2.2 ; E=0
RET

;============================================================
; DATAWRT
;============================================================
DATAWRT:
MOV P1, A
SETB P2.0 ; RS=1 (data)
CLR P2.1 ; R/W=0 (write)
SETB P2.2 ; E=1
ACALL DELAY
CLR P2.2 ; E=0
RET

;============================================================
; DELAY
;============================================================
DELAY:
MOV R3, #250
HERE2: MOV R4, #255
```

```
HERE: DJNZ R4, HERE
DJNZ R3, HERE2
RET

;=============================================================
; Look-up Tables
;=============================================================
ORG 300H

; Command table:
; 38H = init 2-line 5x7
; 0EH = display ON, cursor ON
; 01H = clear screen
; 06H = auto-increment cursor
; C0H = force cursor to Line 2, position 0 (80H | 40H = C0H)
; 00H = null terminator
MYCOM: DB 38H, 0EH, 01H, 06H, 0C0H, 0

; Data: 'S' 'C' 'O' 'P' 'E' + null
MYDATA: DB 'S', 'C', 'O', 'P', 'E', 0
END
```

Key command — C0H explained:
DDRAM address for start of Line 2 = 40H.
'Set DDRAM Address' command = 80H OR address = 80H | 40H = **C0H**
Similarly, any position on Line 2 can be set: C1H = pos 1, C2H = pos 2 … CFH = pos 15.

Chapter 2 : LED Interfacing with 8051

2.1 Introduction to LED Interfacing

An LED (Light Emitting Diode) is the most commonly used output component in microcontroller-based systems. To interface an LED:

Point	Description
Current limiting	A resistor (330Ω – 1kΩ) must be placed in series with every LED to prevent excess current from dama
Interface 1 (Source current)	Port pin → Resistor → LED anode → Cathode → GND. Port pin HIGH = LED ON. Port pin LOW = LE
Interface 2 (Sink current)	VCC → Resistor → LED anode → Cathode → Port pin. Port pin LOW = LED ON (forward biased). Po
Port output	8051 port pins can source/sink ~1.6 mA directly. For brighter LEDs, use a transistor driver (BC547/UL

2.2 Hardware Connection: 8 LEDs on Port 0

Connect 8 LEDs to Port 0 (P0.0–P0.7) through 8 × 1kΩ resistors. The other end of each LED (cathode) connects to GND. Writing FFH to Port 0 turns all LEDs ON; 00H turns all OFF.

■ Example 3 : Continuously Blink All 8 LEDs on Port 0

Write an 8051 assembly program to continuously blink all 8 LEDs connected to **Port 0** with a visible delay between ON and OFF states.

✔ Solution :

The CPL (complement) instruction toggles the accumulator bits, alternating Port 0 between FFH (all ON) and 00H (all OFF).

```
;============================================================
; Blink all 8 LEDs connected to Port 0
; LED ON = port pin HIGH (1) → Interface 1 type connection
;============================================================
ORG 00H

START:
MOV P0, #0FFH ; write 1111 1111 → all 8 LEDs ON
ACALL DELAY ; wait (visible ON time)

MOV A, P0 ; read current port value into A
CPL A ; complement A: FFH → 00H
MOV P0, A ; write 0000 0000 → all 8 LEDs OFF
ACALL DELAY ; wait (visible OFF time)

SJMP START ; repeat forever

;============================================================
; DELAY subroutine – approx. several hundred milliseconds
;============================================================
```

```
DELAY:
MOV R2, #255 ; outer counter
WAIT1:
DJNZ R2, WAIT1 ; decrement until zero
RET
END
```

Trace of execution:
1st iteration: P0 = FFH → LEDs ON → delay → A = FFH → CPL A = 00H → P0 = 00H → LEDs OFF → delay
2nd iteration: P0 = FFH again → LEDs ON → … (repeats continuously)

Chapter 3 : Seven Segment Display Interfacing

3.1 Introduction to 7-Segment Display

A 7-segment LED display consists of **7 LED segments (a–g)** arranged in the shape of a figure-8, plus an optional decimal point (dp). By selectively lighting the appropriate segments, digits 0–9 and several characters can be displayed.

Feature	Description
Segments	7 segments labelled a (top), b (upper-right), c (lower-right), d (bottom), e (lower-left), f (upper-left),
Common Cathode (CC)	All cathodes tied to GND. Send logic HIGH (1) to light a segment. Widely used with microcontroller
Common Anode (CA)	All anodes tied to VCC. Send logic LOW (0) to light a segment. Codes are the bitwise complement
Current limit	Each segment needs a series resistor (~330Ω) to limit current.

3.2 Segment-to-Port Pin Mapping

Standard connection: segment a → port bit 0 (D0/LSB), b → D1, c → D2, d → D3, e → D4, f → D5, g → D6, dp → D7 (MSB).

3.3 Common-Cathode Encoding Table (Digits 0–9 and Letters)

Bit positions: D7=dp, D6=g, D5=f, D4=e, D3=d, D2=c, D1=b, D0=a

Char	g	f	e	d	c	b	a	Binary	Hex Code
0	0	1	1	1	1	1	1	0111111	3FH
1	0	0	0	0	1	1	0	0000110	06H
2	1	0	1	1	0	1	1	1011011	5BH
3	1	0	0	1	1	1	1	1001111	4FH
4	1	1	0	0	1	1	0	1100110	66H
5	1	1	0	1	1	0	1	1101101	6DH
6	1	1	1	1	1	0	1	1111101	7DH
7	0	0	0	0	1	1	1	0000111	07H
8	1	1	1	1	1	1	1	1111111	7FH
9	1	1	0	1	1	1	1	1101111	6FH
H	1	1	1	0	1	1	0	1110110	76H
E	1	1	1	1	0	0	1	1111001	79H
L	0	1	1	1	0	0	0	0111000	38H
O	0	1	1	1	1	1	1	0111111	3FH
S	1	1	0	1	1	0	1	1101101	6DH
C	0	1	1	1	0	0	1	0111001	39H

P	1	1	1	0	0	1	1	1110011	73H

Note: 1 = segment ON (for Common Cathode). For Common Anode, complement each code (e.g., 3FH → C0H for digit '0').

■ Example 4 : Display Digits 0–9 Sequentially on 7-Segment (Port 1)

Write an 8051 assembly program to sequentially display digits **0 through 9** on a common-cathode 7-segment display connected to **Port 1**. Use a look-up table stored at code memory address **400H**. Include a visible time delay between each digit.

✔ Solution :

MOVC fetches each segment pattern from the table. The table ends with a stop value of 00H. When 00H is detected (CJNE A, #0, REPEAT fails), the program loops back to MAIN and restarts from digit 0.

```
;============================================================
; Display digits 0–9 on Common-Cathode 7-segment on Port 1
; Segment encoding: a=D0, b=D1, c=D2, d=D3, e=D4, f=D5, g=D6
;============================================================
ORG 0000H

MAIN:
MOV DPTR, #400H ; point DPTR to look-up table

REPEAT:
CLR A ; offset = 0
MOVC A, @A+DPTR ; fetch segment pattern from table
MOV P1, A ; output pattern to Port 1 (display digit)
ACALL DELAY ; wait for digit to be visible
INC DPTR ; advance to next entry
CJNE A, #0, REPEAT ; if A ≠ 0 (not stop byte), display next
SJMP MAIN ; restart from digit 0

;============================================================
; DELAY – multi-level loop for visible delay
;============================================================
DELAY:
MOV R0, #08H
LP2: MOV R1, #0FFH
LP1: MOV R2, #0FFH
LP3: DJNZ R2, LP3 ; innermost loop
DJNZ R1, LP1
DJNZ R0, LP2
RET

;============================================================
; Look-up table at 400H
; Segment codes for digits 0 to 9 (Common Cathode)
; Followed by 00H as stop/sentinel byte
;============================================================
ORG 400H
DB 3FH ; '0' → segments a,b,c,d,e,f ON (g OFF)
DB 06H ; '1' → segments b,c ON
DB 5BH ; '2' → segments a,b,d,e,g ON
```

```
DB 4FH ; '3' → segments a,b,c,d,g ON
DB 66H ; '4' → segments b,c,f,g ON
DB 6DH ; '5' → segments a,c,d,f,g ON
DB 7DH ; '6' → segments a,c,d,e,f,g ON
DB 07H ; '7' → segments a,b,c ON
DB 7FH ; '8' → all segments ON
DB 6FH ; '9' → segments a,b,c,d,f,g ON
DB 00H ; sentinel – end of table
END
```

■ Example 5 : Display 'HELLO' on 7-Segment Display (Port 0)

Write an 8051 assembly program to display **'HELLO'** on a common-cathode 7-segment display connected to **Port 0**. Use a look-up table at code memory address **200H** and display each character with a visible delay.

✔ Solution :

Encoding: H=76H, E=79H, L=38H, O=3FH (from the encoding table above). R0 is pre-loaded with 5 (number of characters). DJNZ decrements it and stops after 5 characters have been displayed.

```
;============================================================
; Display 'HELLO' on Common-Cathode 7-segment on Port 0
; Look-up table at code memory 200H
;============================================================
ORG 0000H

MAIN:
MOV DPTR, #200H ; point to HELLO table
MOV R0, #05H ; 5 characters to display

REPEAT:
CLR A
MOVC A, @A+DPTR ; fetch segment code
MOV P0, A ; output to Port 0
ACALL DELAY ; visible delay
INC DPTR
DJNZ R0, REPEAT ; loop 5 times
SJMP MAIN ; restart

DELAY:
MOV R0, #08H
LP2: MOV R1, #0FFH
LP1: MOV R2, #0FFH
LP3: DJNZ R2, LP3
DJNZ R1, LP1
DJNZ R0, LP2
RET

;============================================================
; Look-up table at 200H
; Segment codes for H, E, L, L, O
;============================================================
ORG 200H
DB 76H ; 'H' → segments b,c,e,f,g ON
DB 79H ; 'E' → segments a,d,e,f,g ON
DB 38H ; 'L' → segments d,e,f ON
DB 38H ; 'L' → same as above
DB 3FH ; 'O' → same as digit '0' (a,b,c,d,e,f ON)
END
```

Segment code derivation for 'H' = 76H:
Segments needed for H: b, c, e, f, g (middle bar + right columns + left columns)
Bit map: D6=g=1, D5=f=1, D4=e=1, D3=d=0, D2=c=1, D1=b=1, D0=a=0
Binary: 0111 0110 = **76H**

■ Practice Problem 1 : Display 'SCOPE' on Port 1 (Look-up Table at 250H)

Write an 8051 assembly program to display **'SCOPE'** on a common-cathode 7-segment display connected to **Port 1**. Store the look-up table at code memory address **250H**.

✔ Solution :

Encoding: S=6DH, C=39H, O=3FH, P=73H, E=79H (from the encoding table).

```
;============================================================
; Display 'SCOPE' on Common-Cathode 7-segment on Port 1
; Look-up table at code memory 250H
;============================================================
ORG 0000H

MAIN:
MOV DPTR, #250H ; point DPTR to SCOPE table
MOV R0, #05H ; 5 characters

REPEAT:
CLR A
MOVC A, @A+DPTR ; fetch segment pattern
MOV P1, A ; output to Port 1
ACALL DELAY ; visible delay
INC DPTR
DJNZ R0, REPEAT ; repeat 5 times
SJMP MAIN ; restart

DELAY:
MOV R1, #08H
LP2: MOV R2, #0FFH
LP1: MOV R3, #0FFH
LP3: DJNZ R3, LP3
DJNZ R2, LP1
DJNZ R1, LP2
RET

;============================================================
; Look-up table at 250H
; Segment codes for S, C, O, P, E
;============================================================
ORG 250H
DB 6DH ; 'S' → segments a,c,d,f,g ON (same as digit 5)
DB 39H ; 'C' → segments a,d,e,f ON
DB 3FH ; 'O' → segments a,b,c,d,e,f ON (same as '0')
DB 73H ; 'P' → segments a,b,e,f,g ON
DB 79H ; 'E' → segments a,d,e,f,g ON
END
```

Segment Code Verification for 'SCOPE':

Letter	a	b	c	d	e	f	g	Binary	Hex
S	1	0	1	1	0	1	1	1101101	6DH
C	1	0	0	1	1	1	0	0111001	39H
O	1	1	1	1	1	1	0	0111111	3FH

P	1	1	0	0	1	1	1	1110011	73H
E	1	0	0	1	1	1	1	1111001	79H

■ Practice Problem 2 : Display 'SCOPE' on Port 1 with 50 µs Delay per Character

Write an 8051 assembly program to display **'SCOPE'** on a common-cathode 7-segment display connected to **Port 1**. The look-up table is at code memory address **250H**. Include a delay of exactly **50 µs** between displaying each character.

Assume: System clock = 12 MHz → 1 machine cycle = 1 µs

✔ Solution :

Timing analysis for the 50 µs delay subroutine:

Instruction	Machine Cycles	Notes
`ACALL DELAY50`	2	call instruction
`MOV R5, #N`	1	load counter
`DJNZ R5, LP  (×N)`	2 × N	loop body
`RET`	2	return
Total	**5 + 2N µs**	**at 12 MHz**

Solving for N: 5 + 2N = 50 → 2N = 45 → N = 22.5 ≈ **22**
Actual delay with N=22: 5 + 2×22 = 49 µs ≈ 50 µs ✓

```
;============================================================
; Display 'SCOPE' on Port 1 with exactly ~50 µs per character
; System clock = 12 MHz (1 machine cycle = 1 µs)
; Look-up table at code memory 250H
;============================================================
ORG 0000H

MAIN:
MOV DPTR, #250H ; point to SCOPE look-up table
MOV R0, #05H ; 5 characters to display

REPEAT:
CLR A ; clear accumulator
MOVC A, @A+DPTR ; fetch 7-seg pattern (2 cycles)
MOV P1, A ; send to display (2 cycles)
ACALL DELAY50 ; ~50 µs delay (see below)
INC DPTR ; next pattern (2 cycles)
DJNZ R0, REPEAT ; loop 5 times (2 cycles)
SJMP MAIN ; repeat the word forever

;============================================================
; DELAY50 – Software delay of approximately 50 µs
; At 12 MHz: ACALL=2, MOV=1, DJNZ×22=44, RET=2 → total=49 µs
;============================================================
```

```
DELAY50:
MOV R5, #22 ; N = 22 (1 cycle)
LP: DJNZ R5, LP ; 2 µs per iteration × 22 = 44 µs
RET ; 2 cycles
; Total: 2(ACALL) + 1(MOV) + 44(loop) + 2(RET) = 49 µs ≈ 50 µs

;============================================================
; Look-up table at 250H
;============================================================
ORG 250H
DB 6DH ; 'S' = 6DH
DB 39H ; 'C' = 39H
DB 3FH ; 'O' = 3FH
DB 73H ; 'P' = 73H
DB 79H ; 'E' = 79H
END
```

Timing Verification:
ACALL DELAY50 = 2 cycles (2 µs)
MOV R5, #22 = 1 cycle (1 µs)
DJNZ × 22 = 44 cycles (44 µs)
RET = 2 cycles (2 µs)
Total = 49 µs ≈ 50 µs ✓ (error < 2%)

Topic	Key Commands / Codes	Programs Covered
LCD Interfacing	38H=init, 0EH=disp on, 01H=clear 06H=incr cursor 80H+n=Line1 pos n, C0H+n=Line2 pos n RS=0→command, RS=1→data E: H→L pulse latches data	• Display 'NO' at Line1, Pos4 • Display 'HELLO' (MOVC) • Display 'SCOPE' on Line2
LED Interfacing	FFH=all ON, 00H=all OFF CPL A → toggle port Interface1: pin HIGH=ON Interface2: pin LOW=ON	• Blink 8 LEDs on Port 0
7-Segment Display	CC codes: 0=3FH,1=06H,2=5BH 3=4FH,4=66H,5=6DH,6=7DH 7=07H,8=7FH,9=6FH H=76H,E=79H,L=38H,S=6DH C=39H,O=3FH,P=73H	• Digits 0–9 on Port1 (table@400H) • 'HELLO' on Port0 (table@200H) • 'SCOPE' on Port1 (table@250H) • 'SCOPE' with 50µs delay

All examples and practice problems from the Module 5 slides are fully solved above.