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

APPENDICES

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
```

MPMC - Module 5 Exam Preparation

IO Interfacing with Microcontroller 8051 | 20 Marks

LCD | LED | Keypad | ADC | DAC

1. LCD INTERFACING

Q: Write an ALP to display 'NO' on LCD at Line 1, Position 4. (Standard LCD program)

Key Points to Remember:

- RS=0 for command, RS=1 for data
- E pin: High-to-Low pulse latches data
- 38H = Init 2 lines 5x7 | 0EH = Display ON cursor ON | 01H = Clear | 06H = Shift right
- 80H = Line 1 start | C0H = Line 2 start | 84H = Line 1 position 4

Assembly Code:

```
; P1.0-P1.7 → D0-D7 of LCD
; P2.0 → RS  |  P2.1 → R/W  |  P2.2 → E

        ORG 0H
        MOV A,#38H      ;Init LCD: 2 lines, 5x7 matrix
        ACALL COMNWRT   ;send command
        ACALL DELAY
        MOV A,#0EH      ;display ON, cursor ON
        ACALL COMNWRT
        ACALL DELAY
        MOV A,#01H      ;clear LCD
        ACALL COMNWRT
        ACALL DELAY
        MOV A,#06H      ;shift cursor right
        ACALL COMNWRT
        ACALL DELAY
        MOV A,#84H      ;cursor at Line1, Position 4
        ACALL COMNWRT
        ACALL DELAY
        MOV A,#'N'      ;display 'N'
        ACALL DATAWRT
        ACALL DELAY
        MOV A,#'O'      ;display 'O'
        ACALL DATAWRT
AGAIN:  SJMP AGAIN       ;stay here

COMNWRT:                 ;send COMMAND to LCD
        MOV P1,A
        CLR P2.0         ;RS=0 for command
        CLR P2.1         ;R/W=0 for write
        SETB P2.2        ;E=1 (high pulse)
        ACALL DELAY
        CLR P2.2         ;E=0 (H-to-L pulse)
        RET
```

```
DATAWRT:                ;send DATA to LCD
        MOV P1,A
        SETB P2.0       ;RS=1 for data
        CLR P2.1        ;R/W=0 for write
        SETB P2.2       ;E=1 (high pulse)
        ACALL DELAY
        CLR P2.2        ;E=0 (H-to-L pulse)
        RET

DELAY:  MOV R3,#50
HERE2:  MOV R4,#255
HERE:   DJNZ R4,HERE
        DJNZ R3,HERE2
        RET         END
```

 COMNWRT and DATAWRT subroutines are the SAME — only RS pin differs. Memorize this!

Q: Write ALP to display 'VIT' on Line 1 if switch at P0.4=1, else display 'Vellore' on Line 2.

```
SW    EQU P0.4
        ORG 0000H
; --- LCD INIT (5 commands) ---
        MOV DPTR,#CommandPtr
        MOV R7,#05H
        MOV R6,#00H
C1:     MOV A,R6
        MOVC A,@A+DPTR
        ACALL Command
        ACALL Delay
        INC R6
        DJNZ R7,C1

MAIN:   JB SW,SHOW_VIT     ;If switch=1 show VIT

; --- SWITCH=0: Show VELLORE on Line 2 ---
        MOV A,#0C0H        ;Line 2 address
        ACALL Command
        MOV DPTR,#DataPtr2
        MOV R7,#07H
        MOV R6,#00H
D2:     MOV A,R6
        MOVC A,@A+DPTR
        ACALL Dataa
        ACALL Delay
        INC R6
        DJNZ R7,D2
        SJMP MAIN

; --- SWITCH=1: Show VIT on Line 1 ---
SHOW_VIT: MOV A,#80H       ;Line 1 address
        ACALL Command
        MOV DPTR,#DataPtr1
        MOV R7,#03H
        MOV R6,#00H
D1:     MOV A,R6
        MOVC A,@A+DPTR
        ACALL Dataa
        ACALL Delay
        INC R6
        DJNZ R7,D1
        SJMP MAIN
```

```
Command: MOV P1,A
        CLR P2.0  CLR P2.1
        SETB P2.2  CLR P2.2
        ACALL Delay  RET

Dataa:  MOV P1,A
        SETB P2.0  CLR P2.1
        SETB P2.2  CLR P2.2
        ACALL Delay  RET

Delay:  MOV R3,#50
HERE2:  MOV R4,#255
HERE:   DJNZ R4,HERE
        DJNZ R3,HERE2  RET

        ORG 0200H
CommandPtr: DB 38H,0EH,01H,06H,80H
DataPtr1:   DB 'VIT'
        ORG 0300H
DataPtr2:   DB 'VELLORE'          END
```

Q: Write ALP to display 'HELLO' on LCD using MOVC instruction (lookup table method).

```
        ORG 0
        MOV DPTR,#MYCOM
        CLR A
        MOVC A,@A+DPTR
        ACALL COMNWRT
        ACALL DELAY
        INC DPTR
        JZ SEND_DAT
        SJMP $-8              ;loop for commands

SEND_DAT: MOV DPTR,#MYDATA
D1:     CLR A
        MOVC A,@A+DPTR
        JZ AGAIN
        ACALL DATAWRT
        ACALL DELAY
        INC DPTR
        SJMP D1
AGAIN:  SJMP AGAIN

; ... (same COMNWRT, DATAWRT, DELAY subroutines) ...

        ORG 300H
MYCOM:  DB 38H,0EH,01H,06H,86H,0  ;commands + null
MYDATA: DB 'HELLO',0               ;data + null         END
```

The null byte 0 acts as a stop condition — JZ exits the loop when 0 is found.

2. LED INTERFACING

Q: Write an ALP to blink 8 LEDs connected to Port 0 of 8051.

Circuit: P0.0–P0.7 → 1KΩ resistors → LEDs → GND

```
        ORG 00H
START:  MOV P0,#0FFH      ;Turn ON all 8 LEDs (11111111)
        ACALL DELAY
        MOV A,P0
        CPL A             ;Complement: all LEDs OFF (00000000)
        MOV P0,A
        ACALL DELAY
        SJMP START        ;repeat forever

DELAY:  MOV R2,#255
WAIT1:  DJNZ R2,WAIT1
        RET          END
```

CPL A complements accumulator — 0xFF becomes 0x00 and vice versa = blink!

Q: Write ALP to display digits 0-9 on Seven Segment Display connected to Port 1.

Seven Segment Lookup Table (Common Cathode):

0	1	2	3	4	5	6	7	8	9
3FH	06H	5BH	4FH	66H	6DH	7DH	07H	7FH	6FH

```
        ORG 0000H
MAIN:   MOV DPTR,#400H
REPEAT: CLR A
        MOVC A,@A+DPTR    ;load segment code from table
        MOV P1,A          ;send to Port 1
        ACALL DELAY
        INC DPTR          ;next digit
        CJNE A,0,REPEAT   ;stop when 0 reached
        SJMP MAIN

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

        ORG 400H
        DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH  ;0-9         END
```

Q: Write ALP to display 'HELLO' using Seven Segment Display connected to Port 0.

Hex codes: H=76H, E=79H, L=38H, L=38H, O=3FH

```
        ORG 0000H
MAIN:   MOV DPTR,#200H
        MOV R0,#05H       ;5 characters
REPEAT: CLR A
        MOVC A,@A+DPTR
        MOV P0,A          ;send to Port 0
        ACALL DELAY
        INC DPTR
        DJNZ R0,REPEAT
        SJMP MAIN

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

        ORG 200H
        DB 76H,79H,38H,38H,3FH   ;H,E,L,L,O         END
```

3. KEYBOARD (KEYPAD) INTERFACING

Q: Explain keyboard scanning and write ALP to identify the pressed key and send ASCII to P0.

4x4 Matrix Key Layout:

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

Key Scanning Steps:

- Ground all rows → read columns. If all 1s → no key pressed
- If any column = 0 → key pressed, wait 20ms debounce
- Ground one row at a time → read columns to find which row
- Rotate column bits into carry to find exact column
- Use lookup table to get ASCII code

Example: Row=1110 (D0), Column=1011 (D2) → Key 2 pressed

```
; P1.0-P1.3=Rows(Out)  P2.0-P2.3=Columns(In)
K1:     MOV P2,#0FFH      ;make P2 input port
        MOV P1,#0          ;ground ALL rows
        MOV A,P2
        ANL A,#00001111B   ;mask unused bits
        CJNE A,#00001111B,K1  ;wait till all keys released

K2:     ACALL DELAY
        MOV A,P2
        ANL A,#00001111B
        CJNE A,#00001111B,OVER ;any key pressed?
        SJMP K2

OVER:   ACALL DELAY        ;debounce 20ms
        MOV A,P2
        ANL A,#00001111B
        CJNE A,#00001111B,OVER1
        SJMP K2

OVER1:  MOV P1,#11111110B  ;ground Row 0
        MOV A,P2  ANL A,#0FH
        CJNE A,#00001111B,ROW_0
        MOV P1,#11111101B  ;ground Row 1
        MOV A,P2  ANL A,#0FH
        CJNE A,#00001111B,ROW_1
        MOV P1,#11111011B  ;ground Row 2
        MOV A,P2  ANL A,#0FH
        CJNE A,#00001111B,ROW_2
        MOV P1,#11110111B  ;ground Row 3
        MOV A,P2  ANL A,#0FH
```

```
        CJNE A,#00001111B,ROW_3
        LJMP K2

ROW_0:  MOV DPTR,#KCODE0  SJMP FIND
ROW_1:  MOV DPTR,#KCODE1  SJMP FIND
ROW_2:  MOV DPTR,#KCODE2  SJMP FIND
ROW_3:  MOV DPTR,#KCODE3

FIND:   RRC A              ;rotate column bits into carry
        JNC MATCH          ;if carry=0, key found
        INC DPTR           ;else next column
        SJMP FIND

MATCH:  CLR A
        MOVC A,@A+DPTR     ;get ASCII from table
        MOV P0,A           ;send to P0
        LJMP K1

        ORG 300H
KCODE0: DB '0','1','2','3'
KCODE1: DB '4','5','6','7'
KCODE2: DB '8','9','A','B'
KCODE3: DB 'C','D','E','F'         END
```

4. ADC (ANALOG-TO-DIGITAL CONVERTER) INTERFACING

ADC0804 Key Facts:

- 8-bit resolution → 256 steps
- Works at +5V, conversion time ≥ 110µs
- Step size = Vref / 256 (if Vref=2.56V → step = 10mV)
- INTR=1 → conversion not done | INTR=0 → conversion done

ADC0804 Pin Connections:

ADC Pin	8051 Pin	Direction	Purpose
RD	P2.5	Output	Read data from ADC
WR	P2.6	Output	Start conversion
INTR	P2.7	Input	Conversion done flag
D0-D7	P1.0-P1.7	Input	Digital output data
CS	GND	—	Chip Select (always 0)

Q: Write ALP to continuously read analog input from ADC0804 and store in register A.

Steps: 1) WR low→high (start conversion) → 2) Monitor INTR → 3) RD low (read data)

```
        MOV P1,#0FFH        ;make P1 input port
        SETB P2.7           ;make P2.7 input (INTR monitor)
        SETB P2.5           ;RD=1 initially

BACK:   CLR P2.6            ;WR=0
        SETB P2.6           ;WR=1 → Low-to-High pulse starts conversion

HERE:   JB P2.7,HERE        ;wait: INTR=1 means NOT done, wait...
                            ;INTR=0 means conversion DONE → proceed

        CLR P2.5            ;RD=0 → enable output of ADC
        MOV A,P1            ;read 8-bit digital data into A
        SETB P2.5           ;RD=1 → deselect for next round         SJMP BACK
;repeat continuously
```

WR needs L-to-H transition to START conversion. INTR goes LOW when done.

Q: For ADC0848 with Vref=2.56V, find D0-D7 output for (a) Vin=1.7V (b) Vin=2.1V

```
Step size = Vref / 256 = 2.56 / 256 = 10 mV
```

```
(a) Dout = 1.7V / 10mV = 170 decimal = 10101010 binary (b) Dout = 2.1V / 10mV = 210
decimal = 11010010 binary
```

Q: LM35 connected to ADC804. If Vref/2=1.28V, explain how temperature maps to ADC output.

```
LM35 outputs 10mV per degree Celsius
ADC804: 8-bit, 256 steps, Vref=2.56V (Vref/2=1.28V)
Step size = 2.56/256 = 10mV

Temperature 0°C  → Vin=0mV   → Dout=0000 0000
Temperature 1°C  → Vin=10mV  → Dout=0000 0001
Temperature 10°C → Vin=100mV → Dout=0000 1010
Temperature 30°C → Vin=300mV → Dout=0001 1110
 => Digital output DIRECTLY equals temperature in Celsius!
```

5. DAC (DIGITAL-TO-ANALOG CONVERTER) INTERFACING

DAC0808 Key Facts:

- 8-bit current output DAC
- Connected to 8051 Port 1 (D0-D7)
- Output current formula: Iout = Iref × (D7/2 + D6/4 + D5/8 + ... + D0/256)
- Op-Amp converts current to voltage: Vout = -Iout × Rf

Q: Write ALP to generate a STAIR-STEP RAMP using DAC0808 connected to Port 1.

Concept: Send 00H, 01H, 02H ... FFH repeatedly → staircase waveform

```
        CLR A               ;start from 0
AGAIN:  MOV P1,A            ;send data to DAC
        INC A               ;increment: 0→1→2→...→255→0
        ACALL DELAY         ;let DAC settle
        SJMP AGAIN          ;repeat forever

DELAY:  MOV R5,#255
D1:     DJNZ R5,D1
        RET          END
```

When A=FFH and INC A is executed → A becomes 00H automatically (overflow). Perfect loop!

Q: Write ALP to generate a SINE WAVE using DAC0808 and lookup table.

Sine Wave Values (Vout = 5 + 5×sin θ, scaled by 25.6):

Angle	sin θ	Vout	DAC Value	Angle	DAC Value
0°	0	5V	128	210°	64
30°	0.5	7.5V	192	240°	17
60°	0.866	9.33V	238	270°	0
90°	1.0	10V	255	300°	17
120°	0.866	9.33V	238	330°	64
150°	0.5	7.5V	192	360°	128
180°	0	5V	128	—	—

```
AGAIN:  MOV DPTR,#TABLE     ;point to sine table
        MOV R2,#COUNT       ;number of values (13)
BACK:   CLR A
        MOVC A,@A+DPTR      ;get value from table
        MOV P1,A            ;send to DAC
        INC DPTR
        DJNZ R2,BACK        ;loop through table
        SJMP AGAIN          ;repeat sine wave

        ORG 300H
```

```
TABLE:  DB 128,192,238,255,238,192  ;0° to 150°
        DB 128,64,17,0,17,64,128    ;180° to 360°          END
```

 DAC input value = Vout × 25.6 because 10V full scale / 256 steps × 25.6 = 1V per step

Q: Verify sine wave DAC values for (a) 30° and (b) 60°.

```
(a) 30°: Vout = 5 + (5 × sin 30°) = 5 + (5 × 0.5) = 5 + 2.5 = 7.5V
        DAC input = 7.5 × 25.6 = 192 (decimal)

(b) 60°: Vout = 5 + (5 × sin 60°) = 5 + (5 × 0.866) = 5 + 4.33 = 9.33V          DAC input
= 9.33 × 25.6 = 238 (decimal)
```

6. QUICK REFERENCE — LAST MINUTE REVISION

LCD Command Codes:

Code	Command	Code	Command
38H	2 lines, 5x7 matrix init	01H	Clear display
0EH	Display ON, cursor ON	06H	Shift cursor right
0FH	Display ON, cursor blink	80H	Line 1 start
0CH	Display ON, cursor OFF	C0H	Line 2 start
84H	Line1, Position 4	88H	Line1, Position 8

7-Segment Alphabet Codes:

```
H=76H  E=79H  L=38H  O=3FH  S=6DH  C=39H P=73H  A=77H  B=7CH  d=5EH  F=71H  G=3DH
```

ADC0804 — Quick Steps:

```
1. MOV P1,#0FFH    → Make P1 input
2. SETB P2.7       → Monitor INTR
3. CLR P2.6        → WR=0
4. SETB P2.6       → WR=1  (L-to-H starts conversion)
5. JB P2.7,HERE    → Wait for INTR=0 (conversion done)
6. CLR P2.5        → RD=0  (enable ADC output)
7. MOV A,P1        → Read digital data 8. SETB P2.5       → RD=1  (done)
```

DAC0808 Formula:

```
Iout = Iref × (D7/2 + D6/4 + D5/8 + D4/16 + D3/32 + D2/64 + D1/128 + D0/256)
Vout = -Iout × Rf
 For Sine Wave: Vout = 5 + (5 × sin θ)  →  DAC value = Vout × 25.6
```

Most Expected Question Types (20 marks exam):

#	Question Type	Marks	Priority
1	**LCD: Display string at specific position / switch-based display**	**10**	★★★★★
2	**ADC0804: Continuous analog read ALP + calculation**	**8**	★★★★★
3	DAC: Stair-step ramp / Sine wave generation	8	★★★★
4	Seven Segment: Display digits/string via lookup table	6	★★★★
5	Keypad: Scanning logic / identify pressed key	6	★★★

6	LED: Blinking program	4	⋆⋆

ALL THE BEST FOR YOUR EXAM! □