```
;This program adds value 3 to the ACC ten times
MOV A,#0 ;A=0, clear ACC
MOV R2,#10 ;load counter R2=10
AGAIN: ADD A,#03 ;add 03 to ACC
DJNZ R2,AGAIN ;repeat until R2=0,10 times
MOV R5,A ;save A in R5

;Write a program to (a) load the accumulator with the value 55H, and
;(b) complement the ACC 700 times
MOV A,#55H ;A=55H
MOV R3,#10 ;R3=10, outer loop count
NEXT: MOV R2,#70 ;R2=70, inner loop count
AGAIN: CPL A ;complement A register
DJNZ R2,AGAIN ;repeat it 70 times
DJNZ R3,NEXT

;Find the sum of the values 79H, F5H, E2H. Put the sum in registers
R0 (low byte) and R5 (high byte).
MOV A,#0 ;A=0
MOV R5,A ;clear R5
ADD A,#79H ;A=0+79H=79H
JNC N_1 ;if CY=0, add next number
INC R5 ;if CY=1, increment R5
N_1: ADD A,#0F5H ;A=79+F5=6E and CY=1
JNC N_2 ;jump if CY=0
INC R5 ;if CY=1,increment R5 (R5=1)
N_2: ADD A,#0E2H ;A=6E+E2=50 and CY=1
JNC OVER ;jump if CY=0
INC R5 ;if CY=1, increment 5
OVER: MOV R0,A ;now R0=50H, and R5=02

;LCALL Example
ORG 0
BACK: MOV A,#55H ;load A with 55H
MOV P1,A ;send 55H to port 1
LCALL DELAY ;time delay
MOV A,#0AAH ;load A with AA (in hex)
MOV P1,A ;send AAH to port 1
LCALL DELAY
SJMP BACK ;keep doing this indefinitely

;---------- this is delay subroutine ------------
ORG 300H ;put DELAY at address 300H
DELAY: MOV R5,#0FFH ;R5=255 (FF in hex), counter
AGAIN: DJNZ R5,AGAIN ;stay here until R5 become 0
RET ;return to caller (when R5 =0)
END

;Use PUSH/POP in Subroutine
ORG 0
BACK: MOV A,#55H ;load A with 55H
MOV P1,A ;send 55H to p1
MOV R4,#99H
MOV R5,#67H
LCALL DELAY ;time delay
MOV A,#0AAH ;load A with AA
MOV P1,A ;send AAH to p1
LCALL DELAY
SJMP BACK ;keeping doing this

;-------this is the delay subroutine------
ORG 300H
DELAY: PUSH 4 ;push R4
PUSH 5 ;push R5
MOV R4,#0FFH;R4=FFH
NEXT: MOV R5,#0FFH;R5=FFH
AGAIN: DJNZ R5,AGAIN
DJNZ R4,NEXT
POP 5 ;POP into R5
POP 4 ;POP into R4
RET ;return to caller
END
```

```
73
74
75  ;The following code will continuously send out to port 0 the
76  ;alternating value 55H and AAH
77  BACK: MOV A,#55H
78  MOV P0,A
79  ACALL DELAY
80  MOV A,#0AAH
81  MOV P0,A
82  ACALL DELAY
83  SJMP BACK
84
85  ;-------this is the delay subroutine------
86  DELAY: MOV R5,#0FFH ;R5=255 (FF in hex), counter
87  AGAIN: DJNZ R5,AGAIN ;stay here until R5 become 0
88  RET ;return to caller (when R5 =0)
89
90
91  ;Port 0 is configured first as an input port by writing 1s to it, and then
92  ;data is received from that port and sent to P1
93  MOV A,#0FFH ;A=FF hex
94  MOV P0,A ;make P0 an i/p port
95  ;by writing it all 1s
96  BACK: MOV A,P0 ;get data from P0
97  MOV P1,A ;send it to port 1
98  SJMP BACK ;keep doing it
99
100
101 ;The following code will continuously send out to port 0 the
102 ;alternating value 55H and AAH
103 MOV A,#55H
104 BACK: MOV P1,A
105 ACALL DELAY
106 CPL A
107 SJMP BACK
108
109 ;-------this is the delay subroutine------
110 DELAY: MOV R5,#0FFH ;R5=255 (FF in hex), counter
111 AGAIN: DJNZ R5,AGAIN ;stay here until R5 become 0
112 RET ;return to caller (when R5 =0)
113
114 ;Port 1 is configured first as an input port by writing 1s to it, then data
115 ;is received from that port and saved in R7 and R5
116 MOV A,#0FFH ;A=FF hex
117 MOV P1,A ;make P1 an input port
118 ;by writing it all 1s
119 MOV A,P1 ;get data from P1
120 MOV R7,A ;save it to in reg R7
121 ACALL DELAY ;wait
122 MOV A,P1 ;another data from P1
123 MOV R5,A ;save it to in reg R5
124
125 ;-------this is the delay subroutine------
126 DELAY: MOV R5,#0FFH ;R5=255 (FF in hex), counter
127 AGAIN: DJNZ R5,AGAIN ;stay here until R5 become 0
128 RET ;return to caller (when R5 =0)
129
130 ;Write the following programs.
131 ;Create a square wave of 50% duty cycle on bit 0 of port 1.
132 HERE: SETB P1.0 ;set to high bit 0 of port 1
133 LCALL DELAY ;call the delay subroutine
134 CLR P1.0 ;P1.0=0
135 LCALL DELAY
136 SJMP HERE ;keep doing it
137
138 ;;Another way to write the above program is:
139 ;HERE: CPL P1.0 ;set to high bit 0 of port 1
140 ;LCALL DELAY ;call the delay subroutine
141 ;SJMP HERE ;keep doing it
142
143 ;-------this is the delay subroutine------
144 DELAY: MOV R5,#0FFH ;R5=255 (FF in hex), counter
```

```
145   AGAIN: DJNZ R5,AGAIN ;stay here until R5 become 0
146   RET ;return to caller (when R5 =0)
147
148   ;Write a program to perform the following:
149   ;(a) Keep monitoring the P1.2 bit until it becomes high
150   ;(b) When P1.2 becomes high, write value 45H to port 0
151   ;(c) Send a high-to-low (H-to-L) pulse to P2.3
152
153   SETB P1.2 ;make P1.2 an input
154   MOV A,#45H ;A=45H
155   AGAIN: JNB P1.2,AGAIN ; get out when P1.2=1
156   MOV P0,A ;issue A to P0
157   SETB P2.3 ;make P2.3 high
158   CLR P2.3 ;make P2.3 low for H-to-L
159
160
161
162   ;Assume that bit P2.3 is an input and represents the condition of an
163   ;oven. If it goes high, it means that the oven is hot. Monitor the bit
164   ;continuously. Whenever it goes high, send a high-to-low pulse to port
165   ;P1.5 to turn on a buzzer.
166
167   HERE: JNB P2.3,HERE ;keep monitoring for high
168   SETB P1.5 ;set bit P1.5=1
169   CLR P1.5 ;make high-to-low
170   SJMP HERE ;keep repeating
171
172
173   ;A switch is connected to pin P1.7. Write a program to check the status
174   ;of SW and perform the following:
175   ;(a) If SW=0, send letter 'N' to P2
176   ;(b) If SW=1, send letter 'Y' to P2
177
178   SETB P1.7 ;make P1.7 an input
179   AGAIN: JB P1.2,OVER ;jump if P1.7=1
180   MOV P2,'N' ;SW=0, issue 'N' to P2
181   SJMP AGAIN ;keep monitoring
182   OVER: MOV P2,#'Y' ;SW=1, issue 'Y' to P2
183   SJMP AGAIN ;keep monitoring
184
185   ;A switch is connected to pin P1.7. Write a program to check the status
186   ;of SW and perform the following:
187   ;(a) If SW=0, send letter 'N' to P2
188   ;(b) If SW=1, send letter 'Y' to P2
189   ;Use the carry flag to check the switch status.
190
191   SETB P1.7 ;make P1.7 an input
192   AGAIN: MOV C,P1.2 ;read SW status into CF
193   JC OVER ;jump if SW=1
194   MOV P2,#'N' ;SW=0, issue 'N' to P2
195   SJMP AGAIN ;keep monitoring
196   OVER: MOV P2,#'Y' ;SW=1, issue 'Y' to P2
197   SJMP AGAIN ;keep monitoring
198
199   ;Example 4-7
200   ;A switch is connected to pin P1.0 and an LED to pin P2.7. Write a
201   ;program to get the status of the switch and send it to the LED
202
203   SETB P1.7 ;make P1.7 an input
204   AGAIN: MOV C,P1.0 ;read SW status into CF
205   MOV P2.7,C ;send SW status to LED
206   SJMP AGAIN ;keep repeating
207
208
209   ;Example 5-1
210   ;Write code to send 55H to ports P1 and P2, using
211   ;(a) their names (b) their addresses
212   ;Solution :
213
214   ;(a)
215   MOV A,#55H ;A=55H
216   MOV P1,A ;P1=55H
```

```
217   MOV P2,A ;P2=55H
218
219   ;(b) From Table 5-1, P1 address=80H; P2 address=A0H
220   MOV A,#55H ;A=55H
221   MOV 80H,A ;P1=55H
222   MOV 0A0H,A ;P2=55H
223
224   ;Example 5-2
225   ;Show the code to push R5 and A onto the stack and then pop them
226   ;back them into R2 and B, where B = A and R2 = R5
227   ;Solution:
228   PUSH 05 ;push R5 onto stack
229   PUSH 0E0H ;push register A onto stack
230   POP 0F0H ;pop top of stack into B
231   ;now register B = register A
232   POP 02 ;pop top of stack into R2
233   ;now R2=R6
234
235
236   ;Example 5-3
237   ;Write a program to copy the value 55H into RAM memory locations
238   ;40H to 41H using
239   ;(a) direct addressing mode, (b) register indirect addressing mode
240   ;without a loop, and (c) with a loop
241   ;Solution:
242   ;(a)
243   MOV A,#55H ;load A with value 55H
244   MOV 40H,A ;copy A to RAM location 40H
245   MOV 41H,A ;copy A to RAM location 41H
246   ;(b)
247   MOV A,#55H ;load A with value 55H
248   MOV R0,#40H ;load the pointer. R0=40H
249   MOV @R0,A ;copy A to RAM R0 points to
250   INC R0 ;increment pointer. Now R0=41h
251   MOV @R0,A ;copy A to RAM R0 points to
252   ;(c)
253   MOV A,#55H ;A=55H
254   MOV R0,#40H ;load pointer.R0=40H,
255   MOV R2,#02 ;load counter, R2=3
256   AGAIN: MOV @R0,A ;copy 55 to RAM R0 points to
257   INC R0 ;increment R0 pointer
258   DJNZ R2,AGAIN ;loop until counter = zero
259
260
261   ;Example 5-4
262   ;Write a program to clear 16 RAM locations starting at RAM address 60H
263   ;Solution:
264   CLR A ;A=0
265   MOV R1,#60H ;load pointer. R1=60H
266   MOV R7,#16 ;load counter, R7=16
267   AGAIN: MOV @R1,A ;clear RAM R1 points to
268   INC R1 ;increment R1 pointer
269   DJNZ R7,AGAIN ;loop until counter=zero
270
271
272   ;Example 5-5
273   ;Write a program to copy a block of 10 bytes of data from 35H to 60H
274   ;Solution:
275   MOV R0,#35H ;source pointer
276   MOV R1,#60H ;destination pointer
277   MOV R3,#10 ;counter
278   BACK: MOV A,@R0 ;get a byte from source
279   MOV @R1,A ;copy it to destination
280   INC R0 ;increment source pointer
281   INC R1 ;increment destination pointer
282   DJNZ R3,BACK ;keep doing for ten bytes
283
284
285   ;Example 5-6
286   ;In this program, assume that the word "USA" is burned into ROM
287   ;locations starting at 200H. And that the program is burned into ROM
288   ;locations starting at 0. Analyze how the program works and state
```

```
289   ;where "USA" is stored after this program is run.
290   ;Solution:
291   ORG 0000H ;burn into ROM starting at 0
292   MOV DPTR,#200H ;DPTR=200H look-up table addr
293   CLR A ;clear A(A=0)
294   MOVC A,@A+DPTR ;get the char from code space
295   MOV R0,A ;save it in R0
296   INC DPTR ;DPTR=201 point to next char
297   CLR A ;clear A(A=0)
298   MOVC A,@A+DPTR ;get the next char
299   MOV R1,A ;save it in R1
300   INC DPTR ;DPTR=202 point to next char
301   CLR A ;clear A(A=0)
302   MOVC A,@A+DPTR ;get the next char
303   MOV R2,A ;save it in R2
304   Here: SJMP HERE ;stay here
305
306   ;Data is burned into code space starting at 200H
307   ORG 200H
308   MYDATA:DB "USA"
309   END ;end of program
310
311
312
313   ;Example 5-8
314   ;Write a program to get the x value from P1 and send x2 to P2,
315   ;continuously
316   ;Solution:
317   ORG 0
318   MOV DPTR,#300H ;LOAD TABLE ADDRESS
319   MOV A,#0FFH ;A=FF
320   MOV P1,A ;CONFIGURE P1 INPUT PORT
321   BACK: MOV A,P1 ;GET X
322   MOVC A,@A+DPTR ;GET X SQAURE FROM TABLE
323   MOV P2,A ;ISSUE IT TO P2
324   SJMP BACK ;KEEP DOING IT
325
326   ORG 300H
327   XSQR_TABLE: DB 0,1,4,9,16,25,36,49,64,81
328   END
329
330   ;Example 5-10
331   ;Write a program to toggle P1 a total of 200 times. Use RAM
332   ;location 32H to hold your counter value instead of registers R0 - R7
333   ;Solution:
334
335   MOV P1,#55H ;P1=55H
336   MOV A, P1
337   MOV 32H,#200 ;load counter value into RAM loc 32H
338   LOP1: CPL A ;toggle P1
339   MOV P1, A
340   ACALL DELAY
341   DJNZ 32H,LOP1 ;repeat 200 times
342
343   ;-------this is the delay subroutine------
344   DELAY: MOV R5,#0FFH ;R5=255 (FF in hex), counter
345   AGAIN: DJNZ R5,AGAIN ;stay here until R5 become 0
346   RET ;return to caller (when R5 =0)
347
348   ;Example 5-24
349   ;A switch is connected to pin P1.7. Write a program to check the status
350   ;of the switch and make the following decision.
351   ;(a) If SW = 0, send "0" to P2
352   ;(b) If SW = 1, send "1" to P2
353   ;Solution:
354   SW EQU P1.7
355   MYDATA EQU P2
356   HERE: MOV C,SW
357   JC OVER
358   MOV MYDATA,#'0'
359   SJMP HERE
360   OVER: MOV MYDATA,#'1'
```

```
361   SJMP HERE
362   END
363
364   ;Example 5-27
365   ;Assume that the on-chip ROM has a message. Write a program to
366   ;copy it from code space into the upper memory space starting at
367   ;address 80H. Also, as you place a byte in upper RAM, give a copy to
368   ;P0.
369   ;Solution:
370   ORG 0
371   MOV DPTR,#MYDATA
372   MOV R1,#80H ;access the upper memory
373   B1: CLR A
374   MOVC A,@A+DPTR ;copy from code ROM
375   MOV @R1,A ;store in upper memory
376   MOV P0,A ;give a copy to P0
377   JZ EXIT ;exit if last byte
378   INC DPTR ;increment DPTR
379   INC R1 ;increment R1
380   SJMP B1 ;repeat until last byte
381   EXIT: SJMP $ ;stay here when finished
382
383
384   ;---------------
385   ORG 300H
386   MYDATA: DB "The Promise of World Peace",0
387   END
388
389
390   ;Assume that RAM locations 40 - 44H have the following values.
391   ;Write a program to find the sum of the values. At the end of the
392   ;program, register A should contain the low byte and R7 the high byte.
393   ;40 = (7D)
394   ;41 = (EB)
395   ;42 = (C5)
396   ;43 = (5B)
397   ;44 = (30)
398   ;Solution:
399
400   MOV R0,#40H ;load pointer
401   MOV R2,#5 ;load counter
402   CLR A ;A=0
403   MOV R7,A ;clear R7
404   AGAIN: ADD A,@R0 ;add the byte ptr to by R0
405   JNC NEXT ;if CY=0 don't add carry
406   INC R7 ;keep track of carry
407   NEXT: INC R0 ;increment pointer
408   DJNZ R2,AGAIN ;repeat until R2 is zero
409
410
411   ;Write a program to add two 16-bit numbers. Place the sum in R7 and
412   ;R6; R6 should have the lower byte.
413   ;Solution:
414   CLR C ;make CY=0
415   MOV A, #0E7H ;load the low byte now A=E7H
416   ADD A, #8DH ;add the low byte
417   MOV R6, A ;save the low byte sum in R6
418   MOV A, #3CH ;load the high byte
419   ADDC A, #3BH ;add with the carry
420   MOV R7, A ;save the high byte sum
421
422
423   ;Assume that 5 BCD data items are stored in RAM locations starting
424   ;at 40H, as shown below. Write a program to find the sum of all the
425   ;numbers. The result must be in BCD.
426   ;40=(71)
427   ;41=(11)
428   ;42=(65)
429   ;43=(59)
430   ;44=(37)
431   ;Solution:
432
```

```
433   MOV R0,#40H ;Load pointer
434   MOV R2,#5 ;Load counter
435   CLR A ;A=0
436   MOV R7,A ;Clear R7
437   AGAIN: ADD A,@R0 ;add the byte pointer to by R0
438   DA A ;adjust for BCD
439   JNC NEXT ;if CY=0 don't accumulate carry
440   INC R7 ;keep track of carries
441   NEXT: INC R0 ;increment pointer
442   DJNZ R2,AGAIN ;repeat until R2 is 0
443
444
445   ;Assume that register A has packed BCD, write a program to convert
446   ;packed BCD to two ASCII numbers and place them in R2 and R6.
447   MOV A,#29H ;A=29H, packed BCD
448   MOV R2,A ;keep a copy of BCD data
449   ANL A,#0FH ;mask the upper nibble (A=09)
450   ORL A,#30H ;make it an ASCII, A=39H('9')
451   MOV R6,A ;save it
452   MOV A,R2 ;A=29H, get the original data
453   ANL A,#0F0H ;mask the lower nibble
454   RR A ;rotate right
455   RR A ;rotate right
456   RR A ;rotate right
457   RR A ;rotate right
458   ORL A,#30H ;A=32H, ASCII char. '2'
459   MOV R2,A ;save ASCII char in R2
460
461
462
463   ;Example 9-7
464   ;Find the delay generated by timer 0 in the following code, using both
465   ;of the Methods of Figure 9-4. Do not include the overhead due to
466   ;instruction.
467   CLR P2.3 ;Clear P2.3
468   MOV TMOD,#01 ;Timer 0, 16-bitmode
469   HERE: MOV TL0,#3EH ;TL0=3Eh, the low byte
470   MOV TH0,#0B8H ;TH0=B8H, the high byte
471   SETB P2.3 ;SET high timer 0
472   SETB TR0 ;Start the timer 0
473   AGAIN: JNB TF0,AGAIN ;Monitor timer flag 0
474   CLR TR0 ;Stop the timer 0
475   CLR TF0 ;Clear TF0 for next round
476   CLR P2.3
477   ;Solution:
478   ;(a) (FFFFH - B83E + 1) = 47C2H = 18370 in decimal and 18370 ×
479   ;1.085 us = 19.93145 ms
480   ;(b) Since TH - TL = B83EH = 47166 (in decimal) we have 65536 -
481   ;47166 = 18370. This means that the timer counts from B38EH to
482   ;FFFF. This plus Rolling over to 0 goes through a total of 18370 clock
483   ;cycles, where each clock is 1.085 us in duration. Therefore, we have
484   ;18370 × 1.085 us = 19.93145 ms as the width of the pulse.
485
486
487   ;Example 9-8
488   ;Modify TL and TH in Example 9-7 to get the largest time delay
489   ;possible. Find the delay in ms. In your calculation, exclude the
490   ;overhead due to the instructions in the loop.
491   ;Solution:
492   ;To get the largest delay we make TL and TH both 0. This will count
493   ;up from 0000 to FFFFH and then roll over to zero.
494   CLR P2.3 ;Clear P2.3
495   MOV TMOD,#01 ;Timer 0, 16-bitmode
496   HERE: MOV TL0,#0 ;TL0=0, the low byte
497   MOV TH0,#0 ;TH0=0, the high byte
498   SETB P2.3 ;SET high P2.3
499   SETB TR0 ;Start timer 0
500   AGAIN: JNB TF0,AGAIN ;Monitor timer flag 0
501   CLR TR0 ;Stop the timer 0
502   CLR TF0 ;Clear timer 0 flag
503   CLR P2.3
504   ;Making TH and TL both zero means that the timer will count from
```

```
505   ;0000 to FFFF, and then roll over to raise the TF flag. As a result, it
506   ;goes through a total Of 65536 states. Therefore, we have delay =
507   ;(65536 - 0) × 1.085 us = 71.1065ms.
508
509
510   ;Example 9-9
511   ;The following program generates a square wave on P1.5 continuously
512   ;using timer 1 for a time delay. Find the frequency of the square
513   ;wave if XTAL = 11.0592 MHz. In your calculation do not
514   ;include the overhead due to Instructions in the loop.
515   MOV TMOD,#10;Timer 1, mod 1 (16-bitmode)
516   AGAIN: MOV TL1,#34H ;TL1=34H, low byte of timer
517   MOV TH1,#76H ;TH1=76H, high byte timer
518   SETB TR1 ;start the timer 1
519   BACK: JNB TF1,BACK ;till timer rolls over
520   CLR TR1 ;stop the timer 1
521   CPL P1.5 ;comp. p1. to get hi, lo
522   CLR TF1 ;clear timer flag 1
523   SJMP AGAIN ;is not auto-reload
524   ;Solution:
525   ;Since FFFFH - 7634H = 89CBH + 1 = 89CCH and 89CCH = 35276
526   ;clock count and 35276 × 1.085 us = 38.274 ms for half of the
527   ;square wave. The frequency = 13.064Hz.
528   ;Also notice that the high portion and low portion of the square wave
529   ;pulse are equal. In the above calculation, the overhead due to all
530   ;the instruction in the loop is not included.
531
532
533   ;Example 9-10
534   ;Assume that XTAL = 11.0592 MHz. What value do we need to load
535   ;the timer's register if we want to have a time delay of 5 ms
536   ;(milliseconds)? Show the program for timer 0 to create a pulse width
537   ;of 5 ms on P2.3.
538   ;Solution:
539   ;Since XTAL = 11.0592 MHz, the counter counts up every 1.085 us.
540   ;This means that out of many 1.085 us intervals we must make a 5 ms
541   ;pulse. To get that, we divide one by the other. We need 5 ms / 1.085
542   ;us = 4608 clocks. To Achieve that we need to load into TL and TH
543   ;the value 65536 - 4608 = EE00H. Therefore, we have TH = EE and TL = 00.
544   CLR P2.3 ;Clear P2.3
545   MOV TMOD,#01 ;Timer 0, 16-bitmode
546   HERE: MOV TL0,#0 ;TL0=0, the low byte
547   MOV TH0,#0EEH ;TH0=EE, the high byte
548   SETB P2.3 ;SET high P2.3
549   SETB TR0 ;Start timer 0
550   AGAIN: JNB TF0,AGAIN ;Monitor timer flag 0
551   CLR TR0 ;Stop the timer 0
552   CLR TF0 ;Clear timer 0 flag
553
554
555   ;Example 9-11
556   ;Assume that XTAL = 11.0592 MHz, write a program to generate a
557   ;square wave of 2 kHz frequency on pin P1.5.
558   ;Solution:
559   ;This is similar to Example 9-10, except that we must toggle the bit to
560   ;generate the square wave. Look at the following steps.
561   ;(a) T = 1 / f = 1 / 2 kHz = 500 us the period of square wave.
562   ;(b) 1 / 2 of it for the high and low portion of the pulse is 250 us.
563   ;(c) 250 us / 1.085 us = 230 and 65536 - 230 = 65306 which in hex
564   ;is FF1AH.
565   ;(d) TL = 1A and TH = FF, all in hex. The program is as follow.
566   MOV TMOD,#01 ;Timer 0, 16-bitmode
567   AGAIN: MOV TL1,#1AH ;TL1=1A, low byte of timer
568   MOV TH1,#0FFH ;TH1=FF, the high byte
569   SETB TR1 ;Start timer 1
570   BACK: JNB TF1,BACK ;until timer rolls over
571   CLR TR1 ;Stop the timer 1
572   CLR P1.5 ;Clear timer flag 1
573   CLR TF1 ;Clear timer 1 flag
574   SJMP AGAIN ;Reload timer
575
576
```

```
577   ;Example 9-12
578   ;Assume XTAL = 11.0592 MHz, write a program to generate a square
579   ;wave of 50 kHz frequency on pin P2.3.
580   ;Solution:
581   ;Look at the following steps.
582   ;(a) T = 1 / 50 = 20 ms, the period of square wave.
583   ;(b) 1 / 2 of it for the high and low portion of the pulse is 10 ms.
584   ;(c) 10 ms / 1.085 us = 9216 and 65536 - 9216 = 56320 in decimal,
585   ;and in hex it is DC00H.
586   ;(d) TL = 00 and TH = DC (hex).
587   MOV TMOD,#10H ;Timer 1, mod 1
588   AGAIN: MOV TL1,#00 ;TL1=00,low byte of timer
589   MOV TH1,#0DCH ;TH1=DC, the high byte
590   SETB TR1 ;Start timer 1
591   BACK: JNB TF1,BACK ;until timer rolls over
592   CLR TR1 ;Stop the timer 1
593   CLR P2.3 ;Comp. p2.3 to get hi, lo
594   SJMP AGAIN ;Reload timer, mode 1 isn't auto-reload
595
596   ;Example 9-14
597   ;Assume XTAL = 11.0592 MHz, find the frequency of the square
598   ;wave generated on pin P1.0 in the following program
599   MOV TMOD,#20H ;T1/8-bit/auto reload
600   MOV TH1,#5 ;TH1 = 5
601   SETB TR1 ;start the timer 1
602   BACK: JNB TF1,BACK ;till timer rolls over
603   CPL P1.0 ;P1.0 to hi, lo
604   CLR TF1 ;clear Timer 1 flag
605   SJMP BACK ;mode 2 is auto-reload
606   ;Solution:
607   ;First notice the target address of SJMP. In mode 2 we do not need to
608   ;reload TH since it is auto-reload. Now (256 - 05) × 1.085 us =
609   ;251 × 1.085 us = 272.33 us is the high portion of the pulse. Since
610   ;it is a 50% duty cycle square wave, the period T is twice that; as
611   ;a result T = 2 × 272.33 us = 544.67 us and the frequency =
612   ;1.83597 kHz
613
614
615
616   ;Example 9-15
617   ;Find the frequency of a square wave generated on pin P1.0.
618   ;Solution:
619   MOV TMOD,#2H ;Timer 0, mod 2 (8-bit, auto reload)
620   MOV TH0,#0
621   AGAIN: MOV R5,#250 ;multiple delay count
622   ACALL DELAY
623   CPL P1.0
624   SJMP AGAIN
625   DELAY: SETB TR0 ;start the timer 0
626   BACK: JNB TF0,BACK ;stay timer rolls over
627   CLR TR0 ;stop timer
628   CLR TF0 ;clear TF for next round
629   DJNZ R5,DELAY
630   RET
631   ;T = 2 ( 250 × 256 × 1.085 us ) = 138.88ms, and frequency = 72 Hz
632
633
634   ;Example 9-18
635   ;Assuming that clock pulses are fed into pin T1, write a program
636   ;for counter 1 in mode 2 to count the pulses and display the state
637   ;of the TL1 count on P2, which connects to 8 LEDs.
638   ;Solution:
639   MOV TMOD,#01100000B ;counter 1, mode 2, C/T=1 external pulses
640   MOV TH1,#0 ;clear TH1
641   SETB P3.5 ;make T1 input
642   AGAIN: SETB TR1 ;start the counter
643   BACK: MOV A,TL1 ;get copy of TL
644   MOV P2,A ;display it on port 2
645   JNB TF1,Back ;keep doing, if TF = 0
646   CLR TR1 ;stop the counter 1
647   CLR TF1 ;make TF=0
648   SJMP AGAIN ;keep doing it
```

```
649
650
651   ;Write a program for the 8051 to transfer letter "A" serially at 4800
652   ;baud, continuously.
653   ;Solution:
654   MOV TMOD,#20H ;timer 1,mode 2(auto reload)
655   MOV TH1,#-6 ;4800 baud rate
656   MOV SCON,#50H ;8-bit, 1 stop, REN enabled
657   SETB TR1 ;start timer 1
658   AGAIN: MOV SBUF,#"A" ;letter "A" to transfer
659   HERE: JNB TI,HERE ;wait for the last bit
660   CLR TI ;clear TI for next char
661   SJMP AGAIN ;keep sending A
662
663   ;Write a program for the 8051 to transfer "YES" serially at 9600
664   ;baud, 8-bit data, 1 stop bit, do this continuously
665   ;Solution:
666   MOV TMOD,#20H ;timer 1,mode 2(auto reload)
667   MOV TH1,#-3 ;9600 baud rate
668   MOV SCON,#50H ;8-bit, 1 stop, REN enabled
669   SETB TR1 ;start timer 1
670   AGAIN: MOV A,#"Y" ;transfer "Y"
671   ACALL TRANS
672   MOV A,#"E" ;transfer "E"
673   ACALL TRANS
674   MOV A,#"S" ;transfer "S"
675   ACALL TRANS
676   SJMP AGAIN ;keep doing it
677   ;serial data transfer subroutine
678   TRANS: MOV SBUF,A ;load SBUF
679   HERE: JNB TI,HERE ;wait for the last bit
680   CLR TI ;get ready for next byte
681   RET
682
683
684   ;Write a program for the 8051 to receive bytes of data serially, and
685   ;put them in P1, set the baud rate at 4800, 8-bit data, and 1 stop bit
686   ;Solution:
687   MOV TMOD,#20H ;timer 1,mode 2(auto reload)
688   MOV TH1,#-6 ;4800 baud rate
689   MOV SCON,#50H ;8-bit, 1 stop, REN enabled
690   SETB TR1 ;start timer 1
691   HERE: JNB RI,HERE ;wait for char to come in
692   MOV A,SBUF ;saving incoming byte in A
693   MOV P1,A ;send to port 1
694   CLR RI ;get ready to receive next byte
695   SJMP HERE ;keep getting data
696
697   ;Example 10-5
698   ;Assume that the 8051 serial port is connected to the COM port of
699   ;IBM PC, and on the PC, we are using the terminal.exe program to
700   ;send and receive data serially. P1 and P2 of the 8051 are connected
701   ;to LEDs and switches, respectively. Write an 8051 program to (a)
702   ;send to PC the message "We Are Ready", (b) receive any data send
703   ;by PC and put it on LEDs connected to P1, and (c) get data on
704   ;switches connected to P2 and send it to PC serially. The program
705   ;should perform part (a) once, but parts (b) and (c) continuously, use
706   ;4800 baud rate.
707
708   ;Solution:
709   ORG 0
710   MOV P2,#0FFH ;make P2 an input port
711   MOV TMOD,#20H ;timer 1, mode 2
712   MOV TH1,#0FAH ;4800 baud rate
713   MOV SCON,#50H ;8-bit, 1 stop, REN enabled
714   SETB TR1 ;start timer 1
715   MOV DPTR,#MYDATA ;load pointer for message
716   H_1: CLR A
717   MOVC A,@A+DPTR ;get the character
718   JZ B_1 ;if last character get out
719   ACALL SEND ;otherwise call transfer
720   INC DPTR ;next one
```

```
721   SJMP H_1 ;stay in loop
722   B_1: MOV a,P2 ;read data on P2
723   ACALL SEND ;transfer it serially
724   ACALL RECV ;get the serial data
725   MOV P1,A ;display it on LEDs
726   SJMP B_1 ;stay in loop indefinitely
727
728   ;----serial data transfer. ACC has the data------
729   SEND: MOV SBUF,A ;load the data
730   H_2: JNB TI,H_2 ;stay here until last bit gone
731   CLR TI ;get ready for next char
732   RET ;return to caller
733
734   ;----Receive data serially in ACC---------------
735   RECV: JNB RI,RECV ;wait here for char
736   MOV A,SBUF ;save it in ACC
737   CLR RI ;get ready for next char
738   RET ;return to caller
739
740   ;-----The message--------------
741   MYDATA: DB "We Are Ready",0
742   END
743
744   ;Example 10-6
745   ;Assume that XTAL = 11.0592 MHz for the following program,
746   ;state (a) what this program does, (b) compute the frequency used
747   ;by timer 1 to set the baud rate, and (c) find the baud rate of the data transfer.
748   ;Solution:
749   ;(a) This program transfers ASCII letter B (01000010
750   ;binary) continuously
751   ;(b) With XTAL = 11.0592 MHz and SMOD = 1 in the
752   ;above program, we have:
753   ;11.0592 / 12 = 921.6 kHz machine cycle frequency.
754   ;921.6 / 16 = 57,600 Hz frequency used by timer 1
755   ;to set the baud rate.
756   ;57600 / 3 = 19,200, the baud rate.
757
758   MOV A,PCON ;A=PCON
759   MOV ACC.7 ;make D7=1
760   MOV PCON,A ;SMOD=1, double baud rate with same XTAL freq.
761   MOV TMOD,#20H ;timer 1, mode 2
762   MOV TH1,-3 ;19200 (57600/3 =19200)
763   MOV SCON,#50H ;8-bit data, 1 stop bit, RI enabled
764   SETB TR1 ;start timer 1
765   MOV A,#"B" ;transfer letter B
766   A_1: CLR TI ;make sure TI=0
767   MOV SBUF,A ;transfer it
768   H_1: JNB TI,H_1 ;stay here until the last bit is gone
769   SJMP A_1 ;keep sending "B" again
770
771
772
773
774   ;Example 10-10
775   ;Write a program to send the message "The Earth is but One
776   ;Country" to serial port. Assume a SW is connected to pin P1.2.
777   ;Monitor its status and set the baud rate as follows:
778   ;SW = 0, 4800 baud rate
779   ;SW = 1, 9600 baud rate
780   ;Assume XTAL = 11.0592 MHz, 8-bit data, and 1 stop bit.
781   ;Solution:
782   SW BIT P1.2
783   ORG 0H ;starting position
784   MAIN:
785   MOV TMOD,#20H
786   MOV TH1,#-6 ;4800 baud rate (default)
787   MOV SCON,#50H
788   SETB TR1
789   SETB SW ;make SW an input
790   S1: JNB SW,SLOWSP ;check SW status
791   MOV A,PCON ;read PCON
792   SETB ACC.7 ;set SMOD high for 9600
```

```
793   MOV PCON,A ;write PCON
794   SJMP OVER ;send message
795
796   SLOWSP:
797   MOV A,PCON ;read PCON
798   SETB ACC.7 ;set SMOD low for 4800
799   MOV PCON,A ;write PCON
800   OVER: MOV DPTR,#MESS1 ;load address to message
801   FN: CLR A
802   MOVC A,@A+DPTR ;read value
803   JZ S1 ;check for end of line
804   ACALL SENDCOM ;send value to serial port
805   INC DPTR ;move to next value
806   SJMP FN ;repeat
807   ;------------
808   SENDCOM:
809   MOV SBUF,A ;place value in buffer
810   HERE: JNB TI,HERE ;wait until transmitted
811   CLR TI ;clear
812   RET ;return
813   ;------------
814   MESS1: DB "The Earth is but One Country",0
815   END
816
817
818   ;Example 11-2
819   ;Write a program that continuously get 8-bit data from P0 and sends it
820   ;to P1 while simultaneously creating a square wave of 200 µs period
821   ;on pin P2.1. Use timer 0 to create the square wave. Assume that
822   ;XTAL = 11.0592 MHz.
823   ;Solution:
824   ;We will use timer 0 in mode 2 (auto reload). TH0 = 100/1.085 us = 92
825
826   ;--upon wake-up go to main, avoid using
827   ;memory allocated to Interrupt Vector Table
828   ORG 0000H
829   LJMP MAIN ;by-pass interrupt vector table
830   ;
831   ;--ISR for timer 0 to generate square wave
832   ORG 000BH ;Timer 0 interrupt vector table
833   CPL P2.1 ;toggle P2.1 pin
834   RETI ;return from ISR
835
836   ;--The main program for initialization
837   ORG 0030H ;after vector table space
838   MAIN: MOV TMOD,#02H ;Timer 0, mode 2
839   MOV P0,#0FFH ;make P0 an input port
840   MOV TH0,#-92 ;TH0=A4H for -92
841   MOV IE,#82H ;IE=10000010 (bin) enable Timer 0
842   SETB TR0 ;Start Timer 0
843   BACK: MOV A,P0 ;get data from P0
844   MOV P1,A ;issue it to P1
845   SJMP BACK ;keep doing it loop unless interrupted by TF0
846   END
847
848
849   ;Example 11-3
850   ;Rewrite Example 11-2 to create a square wave that has a high portion
851   ;of 1085 us and a low portion of 15 us. Assume XTAL=11.0592MHz.
852   ;Use timer 1.
853   ;Solution:
854   ;Since 1085 us is 1000 × 1.085 we need to use mode 1 of timer 1.
855   ;--upon wake-up go to main, avoid using
856   ;memory allocated to Interrupt Vector Table
857   ORG 0000H
858   LJMP MAIN ;by-pass int. vector table
859   ;--ISR for timer 1 to generate square wave
860   ORG 001BH ;Timer 1 int. vector table
861   LJMP ISR_T1 ;jump to ISR
862
863   ;--The main program for initialization
864   ORG 0030H ;after vector table space
```

```
865   MAIN: MOV TMOD,#10H ;Timer 1, mode 1
866   MOV P0,#0FFH ;make P0 an input port
867   MOV TL1,#018H ;TL1=18 low byte of -1000
868   MOV TH1,#0FCH ;TH1=FC high byte of -1000
869   MOV IE,#88H ;10001000 enable Timer 1 int
870   SETB TR1 ;Start Timer 1
871   BACK: MOV A,P0 ;get data from P0
872   MOV P1,A ;issue it to P1
873   SJMP BACK ;keep doing it
874   ;Timer 1 ISR. Must be reloaded, not auto-reload
875   ISR_T1: CLR TR1 ;stop Timer 1
876   MOV R2,#4 ; 2MC
877   CLR P2.1 ;P2.1=0, start of low portion
878   HERE: DJNZ R2,HERE ;4x2 machine cycle 8MC
879   MOV TL1,#18H ;load T1 low byte value 2MC
880   MOV TH1,#0FCH;load T1 high byte value 2MC
881   SETB TR1 ;starts timer1 1MC
882   SETB P2.1 ;P2.1=1,back to high 1MC
883   RETI ;return to main
884   END
885
886   ;Example 11-5
887   ;Assume that the INT1 pin is connected to a switch that is normally
888   ;high. Whenever it goes low, it should turn on an LED. The LED is
889   ;connected to P1.3 and is normally off. When it is turned on it should
890   ;stay on for a fraction of a second. As long as the switch is pressed low,
891   ;the LED should stay on.
892   ;Solution:
893   ORG 0000H
894   LJMP MAIN ;by-pass interrupt vector table
895   ;--ISR for INT1 to turn on LED
896   ORG 0013H ;INT1 ISR
897   SETB P1.3 ;turn on LED
898   MOV R3,#255
899   BACK: DJNZ R3,BACK ;keep LED on for a while
900   CLR P1.3 ;turn off the LED
901   RETI ;return from ISR
902   ;--MAIN program for initialization
903   ORG 30H
904   MAIN: MOV IE,#10000100B ;enable external INT 1
905   HERE: SJMP HERE ;stay here until get interrupted
906   END
907
908   ;Assume that pin 3.3 (INT1) is connected to a pulse generator, write a
909   ;program in which the falling edge of the pulse will send a high to
910   ;P1.3, which is connected to an LED (or buzzer). In other words, the
911   ;LED is turned on and off at the same rate as the pulses are applied to
912   ;the INT1 pin.
913   ;Solution:
914   ORG 0000H
915   LJMP MAIN
916   ;--ISR for hardware interrupt INT1 to turn on LED
917   ORG 0013H ;INT1 ISR
918   SETB P1.3 ;turn on LED
919   MOV R3,#255
920   BACK: DJNZ R3,BACK ;keep the buzzer on for a while
921   CLR P1.3 ;turn off the buzzer
922   RETI ;return from ISR
923   ;------MAIN program for initialization
924   ORG 30H
925   MAIN: SETB TCON.2 ;make INT1 edge-triggered int.
926   MOV IE,#10000100B ;enable External INT 1
927   HERE: SJMP HERE ;stay here until get interrupted
928   END
929
930
931   ;Example 11-8
932   ;Write a program in which the 8051 reads data from P1 and writes it to
933   ;P2 continuously while giving a copy of it to the serial COM port to be
934   ;transferred serially. Assume that XTAL=11.0592. Set the baud rate at 9600.
935   ;Solution:
936   ORG 0000H
```

```
937   LJMP MAIN
938   ORG 23H
939   LJMP SERIAL ;jump to serial int ISR
940   ORG 30H
941   MAIN: MOV P1,#0FFH ;make P1 an input port
942   MOV TMOD,#20H ;timer 1, auto reload
943   MOV TH1,#0FDH ;9600 baud rate
944   MOV SCON,#50H ;8-bit,1 stop, ren enabled
945   MOV IE,10010000B ;enable serial int.
946   SETB TR1 ;start timer 1
947   BACK: MOV A,P1 ;read data from port 1
948   MOV SBUF,A ;give a copy to SBUF
949   MOV P2,A ;send it to P2
950   SJMP BACK ;stay in loop indefinitely
951
952   ;-----------------SERIAL PORT ISR
953   ORG 100H
954   SERIAL: JB TI,TRANS;jump if TI is high
955   MOV A,SBUF ;otherwise due to receive
956   CLR RI ;clear RI since CPU doesn't
957   RETI ;return from ISR
958   TRANS: CLR TI ;clear TI since CPU doesn't
959   RETI ;return from ISR
960   END
961
962   ;Example 11-9
963   ;Write a program in which the 8051 gets data from P1 and sends it to
964   ;P2 continuously while incoming data from the serial port is sent to P0.
965   ;Assume that XTAL=11.0592. Set the baud rata at 9600.
966   ;Solution:
967   ORG 0000H
968   LJMP MAIN
969   ORG 23H
970   LJMP SERIAL ;jump to serial int ISR
971   ORG 30H
972   MAIN: MOV P1,#0FFH ;make P1 an input port
973   MOV TMOD,#20H ;timer 1, auto reload
974   MOV TH1,#0FDH ;9600 baud rate
975   MOV SCON,#50H ;8-bit,1 stop, ren enabled
976   MOV IE,10010000B ;enable serial int.
977   SETB TR1 ;start timer 1
978   BACK: MOV A,P1 ;read data from port 1
979   MOV P2,A ;send it to P2
980   SJMP BACK ;stay in loop indefinitely
981
982   ;-----------------SERIAL PORT ISR
983   ORG 100H
984   SERIAL: JB TI,TRANS;jump if TI is high
985   MOV A,SBUF ;otherwise due to receive
986   MOV P0,A ;send incoming data to P0
987   CLR RI ;clear RI since CPU doesn't
988   RETI ;return from ISR
989   TRANS: CLR TI ;clear TI since CPU doesn't
990   RETI ;return from ISR
991   END
992
993   ;Example 11-10
994   ;Write a program using interrupts to do the following:
995   ;(a) Receive data serially and sent it to P0,
996   ;(b) Have P1 port read and transmitted serially, and a copy given to
997   ;P2,
998   ;(c) Make timer 0 generate a square wave of 5kHz frequency on P0.1.
999   ;Assume that XTAL-11,0592. Set the baud rate at 4800.
1000  ;Solution:
1001  ORG 0
1002  LJMP MAIN
1003  ORG 000BH ;ISR for timer 0
1004  CPL P0.1 ;toggle P0.1
1005  RETI ;return from ISR
1006  ORG 23H ;
1007  LJMP SERIAL ;jump to serial interrupt ISR
1008  ORG 30H
```

```
1009   MAIN: MOV P1,#0FFH ;make P1 an input port
1010   MOV TMOD,#22H;timer 1,mode 2(auto reload)
1011   MOV TH1,#0F6H;4800 baud rate
1012   MOV SCON,#50H;8-bit, 1 stop, ren enabled
1013   MOV TH0,#-92 ;for 5kHZ wave
1014   MOV IE,10010010B ;enable serial int.
1015   SETB TR1 ;start timer 1
1016   SETB TR0 ;start timer 0
1017   BACK: MOV A,P1 ;read data from port 1
1018   MOV SBUF,A ;give a copy to SBUF
1019   MOV P2,A ;send it to P2
1020   SJMP BACK ;stay in loop indefinitely
1021   ;-----------------SERIAL PORT ISR
1022   ORG 100H
1023   SERIAL:JB TI,TRANS;jump if TI is high
1024   MOV A,SBUF ;otherwise due to receive
1025   MOV P0,A ;send serial data to P0
1026   CLR RI ;clear RI since CPU doesn't
1027   RETI ;return from ISR
1028   TRANS: CLR TI ;clear TI since CPU doesn't
1029   RETI ;return from ISR
1030   END
1031
1032
1033   ;To send any of the commands to the LCD, make pin RS=0. For data,
1034   ;make RS=1. Then send a high-to-low pulse to the E pin to enable the
1035   ;internal latch of the LCD. This is shown in the code below.
1036   ;calls a time delay before sending next data/command
1037   ;P1.0-P1.7 are connected to LCD data pins D0-D7
1038   ;P2.0 is connected to RS pin of LCD
1039   ;P2.1 is connected to R/W pin of LCD
1040   ;P2.2 is connected to E pin of LCD
1041
1042   ORG 0
1043
1044   MOV A,#38H ;INIT. LCD 2 LINES, 5X7 MATRIX
1045   ACALL COMNWRT ;call command subroutine
1046   ACALL DELAY ;give LCD some time
1047
1048   MOV A,#0EH ;display on, cursor on
1049   ACALL COMNWRT ;call command subroutine
1050   ACALL DELAY ;give LCD some time
1051
1052   MOV A,#01 ;clear LCD
1053   ACALL COMNWRT ;call command subroutine
1054   ACALL DELAY ;give LCD some time
1055
1056   MOV A,#06H ;shift cursor right
1057   ACALL COMNWRT ;call command subroutine
1058   ACALL DELAY ;give LCD some time
1059
1060   MOV A,#84H ;cursor at line 1, pos. 4
1061   ACALL COMNWRT ;call command subroutine
1062   ACALL DELAY ;give LCD some time
1063
1064   MOV A,#'N' ;display letter N
1065   ACALL DATAWRT ;call display subroutine
1066   ACALL DELAY ;give LCD some time
1067
1068   MOV A,#'O' ;display letter O
1069   ACALL DATAWRT ;call display subroutine
1070   AGAIN: SJMP AGAIN ;stay here
1071
1072   COMNWRT: ;send command to LCD
1073       MOV P1,A ;copy reg A to port 1
1074       CLR P2.0 ;RS=0 for command
1075       CLR P2.1 ;R/W=0 for write
1076       SETB P2.2 ;E=1 for high pulse
1077       CLR P2.2 ;E=0 for H-to-L pulse
1078       RET
1079   DATAWRT: ;write data to LCD
1080       MOV P1,A ;copy reg A to port 1
```

```
1081        CLR P2.0 ;RS=0 for command
1082        CLR P2.1 ;R/W=0 for write
1083        SETB P2.2 ;E=1 for high pulse
1084        CLR P2.2 ;E=0 for H-to-L pulse
1085        RET
1086    DELAY: MOV R3,#50 ;50 or higher for fast CPUs
1087    HERE2: MOV R4,#255 ;R4 = 255
1088    HERE: DJNZ R4,HERE ;stay until R4 becomes 0
1089        DJNZ R3,HERE2
1090        RET
1091        END
1092
1093
1094    ;Check busy flag before sending data, command to LCD
1095    ;p1=data pin
1096    ;P2.0 connected to RS pin
1097    ;P2.1 connected to R/W pin
1098    ;P2.2 connected to E pin
1099    ORG 0
1100    MOV A,#38H ;init. LCD 2 lines ,5x7 matrix
1101    ACALL COMMAND ;issue command
1102    MOV A,#0EH ;LCD on, cursor on
1103    ACALL COMMAND ;issue command
1104    MOV A,#01H ;clear LCD command
1105    ACALL COMMAND ;issue command
1106    MOV A,#06H ;shift cursor right
1107    ACALL COMMAND ;issue command
1108    MOV A,#86H ;cursor: line 1, pos. 6
1109    ACALL COMMAND ;command subroutine
1110    MOV A,#'N' ;display letter N
1111    ACALL DATA_DISPLAY
1112    MOV A,#'O' ;display letter O
1113    ACALL DATA_DISPLAY
1114    HERE:SJMP HERE ;STAY HERE
1115
1116    COMMAND:
1117        ACALL READY ;is LCD ready?
1118        MOV P1,A ;issue command code
1119        CLR P2.0 ;RS=0 for command
1120        CLR P2.1 ;R/W=0 to write to LCD
1121        SETB P2.2 ;E=1 for H-to-L pulse
1122        CLR P2.2 ;E=0,latch in
1123        RET
1124    DATA_DISPLAY:
1125        ACALL READY ;is LCD ready?
1126        MOV P1,A ;issue data
1127        SETB P2.0 ;RS=1 for data
1128        CLR P2.1 ;R/W =0 to write to LCD
1129        SETB P2.2 ;E=1 for H-to-L pulse
1130        CLR P2.2 ;E=0,latch in
1131        RET
1132    READY:
1133        SETB P1.7 ;make P1.7 input port
1134        CLR P2.0 ;RS=0 access command reg
1135        SETB P2.1 ;R/W=1 read command reg
1136
1137    ;read command reg and check busy flag
1138    BACK:SETB P2.2 ;E=1 for H-to-L pulse
1139        CLR P2.2 ;E=0 H-to-L pulse
1140        JB P1.7,BACK ;stay until busy flag=0
1141        RET
1142        END
1143
1144    ;To send any of the commands to the LCD, make pin RS=0. For data,
1145    ;make RS=1. Then send a high-to-low pulse to the E pin to enable the
1146    ;internal latch of the LCD. This is shown in the code below.
1147    ;calls a time delay before sending next data/command
1148    ;P1.0-P1.7 are connected to LCD data pins D0-D7
1149    ;P2.0 is connected to RS pin of LCD
1150    ;P2.1 is connected to R/W pin of LCD
1151    ;P2.2 is connected to E pin of LCD
1152    ORG 0H
```

```
1153
1154    MOV A,#38H ;INIT. LCD 2 LINES, 5X7 MATRIX
1155    ACALL COMNWRT ;call command subroutine
1156    ACALL DELAY ;give LCD some time
1157
1158    MOV A,#0EH ;display on, cursor on
1159    ACALL COMNWRT ;call command subroutine
1160    ACALL DELAY ;give LCD some time
1161
1162    MOV A,#01 ;clear LCD
1163    ACALL COMNWRT ;call command subroutine
1164    ACALL DELAY ;give LCD some time
1165
1166    MOV A,#06H ;shift cursor right
1167    ACALL COMNWRT ;call command subroutine
1168    ACALL DELAY ;give LCD some time
1169
1170    MOV A,#84H ;cursor at line 1, pos. 4
1171    ACALL COMNWRT ;call command subroutine
1172    ACALL DELAY ;give LCD some time
1173
1174    MOV A,#'N' ;display letter N
1175    ACALL DATAWRT ;call display subroutine
1176    ACALL DELAY ;give LCD some time
1177
1178    MOV A,#'O' ;display letter O
1179    ACALL DATAWRT ;call display subroutine
1180    AGAIN: SJMP AGAIN ;stay here
1181
1182    COMNWRT: ;send command to LCD
1183        MOV P1,A ;copy reg A to port 1
1184        CLR P2.0 ;RS=0 for command
1185        CLR P2.1 ;R/W=0 for write
1186        SETB P2.2 ;E=1 for high pulse
1187        ACALL DELAY ;give LCD some time
1188        CLR P2.2 ;E=0 for H-to-L pulse
1189        RET
1190    DATAWRT: ;write data to LCD
1191        MOV P1,A ;copy reg A to port 1
1192        SETB P2.0 ;RS=1 for data
1193        CLR P2.1 ;R/W=0 for write
1194        SETB P2.2 ;E=1 for high pulse
1195        ACALL DELAY ;give LCD some time
1196        CLR P2.2 ;E=0 for H-to-L pulse
1197        RET
1198    DELAY: MOV R3,#50 ;50 or higher for fast CPUs
1199    HERE2: MOV R4,#255 ;R4 = 255
1200    HERE: DJNZ R4,HERE ;stay until R4 becomes 0
1201    DJNZ R3,HERE2
1202    RET
1203    END
1204
1205
1206    ;Check busy flag before sending data, command to LCD
1207    ;p1=data pin
1208    ;P2.0 connected to RS pin
1209    ;P2.1 connected to R/W pin
1210    ;P2.2 connected to E pin
1211    ORG 0H
1212
1213    MOV A,#38H ;init. LCD 2 lines ,5x7 matrix
1214    ACALL COMMAND ;issue command
1215
1216    MOV A,#0EH ;LCD on, cursor on
1217    ACALL COMMAND ;issue command
1218
1219    MOV A,#01H ;clear LCD command
1220    ACALL COMMAND ;issue command
1221
1222    MOV A,#06H ;shift cursor right
1223    ACALL COMMAND ;issue command
1224
```

```
1225    MOV A,#86H ;cursor: line 1, pos. 6
1226    ACALL COMMAND ;command subroutine
1227
1228    MOV A,#'N' ;display letter N
1229    ACALL DATA_DISPLAY
1230
1231    MOV A,#'O' ;display letter O
1232    ACALL DATA_DISPLAY
1233    HERE:SJMP HERE ;STAY HERE
1234
1235    COMMAND:
1236        ACALL READY ;is LCD ready?
1237        MOV P1,A ;issue command code
1238        CLR P2.0 ;RS=0 for command
1239        CLR P2.1 ;R/W=0 to write to LCD
1240        SETB P2.2 ;E=1 for H-to-L pulse
1241        CLR P2.2 ;E=0,latch in
1242        RET
1243    DATA_DISPLAY:
1244        ACALL READY ;is LCD ready?
1245        MOV P1,A ;issue data
1246        SETB P2.0 ;RS=1 for data
1247        CLR P2.1 ;R/W =0 to write to LCD
1248        SETB P2.2 ;E=1 for H-to-L pulse
1249        CLR P2.2 ;E=0,latch in
1250        RET
1251    READY:
1252        SETB P1.7 ;make P1.7 input port
1253        CLR P2.0 ;RS=0 access command reg
1254        SETB P2.1 ;R/W=1 read command reg
1255        ;read command reg and check busy flag
1256    BACK:SETB P2.2 ;E=1 for H-to-L pulse
1257    CLR P2.2 ;E=0 H-to-L pulse
1258    JB P1.7,BACK ;stay until busy flag=0
1259    RET
1260    END
1261
1262
1263    ;Call a time delay before sending next data/command
1264    ; P1.0-P1.7=D0-D7, P2.0=RS, P2.1=R/W, P2.2=E
1265    ORG 0
1266    MOV DPTR,#MYCOM
1267    C1: CLR A
1268        MOVC A,@A+DPTR
1269        ACALL COMNWRT ;call command subroutine
1270        ACALL DELAY ;give LCD some time
1271        INC DPTR
1272        JZ SEND_DAT
1273        SJMP C1
1274    SEND_DAT:
1275    MOV DPTR,#MYDATA
1276    D1: CLR A
1277        MOVC A,@A+DPTR
1278        ACALL DATAWRT ;call command subroutine
1279        ACALL DELAY ;give LCD some time
1280        INC DPTR
1281        JZ AGAIN
1282        SJMP D1
1283    AGAIN: SJMP AGAIN ;stay here
1284
1285    COMNWRT: ;send command to LCD
1286        MOV P1,A ;copy reg A to P1
1287        CLR P2.0 ;RS=0 for command
1288        CLR P2.1 ;R/W=0 for write
1289        SETB P2.2 ;E=1 for high pulse
1290        ACALL DELAY ;give LCD some time
1291        CLR P2.2 ;E=0 for H-to-L pulse
1292        RET
1293    DATAWRT: ;write data to LCD
1294        MOV P1,A ;copy reg A to port 1
1295        SETB P2.0 ;RS=1 for data
1296        CLR P2.1 ;R/W=0 for write
```

```
1297        SETB P2.2 ;E=1 for high pulse
1298        ACALL DELAY ;give LCD some time
1299        CLR P2.2 ;E=0 for H-to-L pulse
1300        RET
1301    DELAY: MOV R3,#250 ;50 or higher for fast CPUs
1302    HERE2: MOV R4,#255 ;R4 = 255
1303    HERE: DJNZ R4,HERE ;stay until R4 becomes 0
1304    DJNZ R3,HERE2
1305    RET
1306
1307    ;ORG 300H
1308    ;MYCOM: DB 38H,0EH,01,06,84H,0 ; commands and null
1309    ;MYDATA: DB "HELLO",0
1310    ;END
1311
1312    ;Program 12-4: Keyboard Program
1313    ;keyboard subroutine. This program sends the ASCII
1314    ;code for pressed key to P0.1
1315    ;P1.0-P1.3 connected to rows, P2.0-P2.3 to column
1316    MOV P2,#0FFH ;make P2 an input port
1317    K1: MOV P1,#0 ;ground all rows at once
1318    MOV A,P2 ;read all col
1319    ;(ensure keys open)
1320    ANL A,00001111B ;masked unused bits
1321    CJNE A,#00001111B,K1 ;till all keys release
1322    K2: ACALL DELAY ;call 20 msec delay
1323    MOV A,P2 ;see if any key is pressed
1324    ANL A,00001111B ;mask unused bits
1325    CJNE A,#00001111B,OVER;key pressed, find row
1326    SJMP K2 ;check till key pressed
1327    OVER: ACALL DELAY ;wait 20 msec debounce time
1328    MOV A,P2 ;check key closure
1329    ANL A,00001111B ;mask unused bits
1330    CJNE A,#00001111B,OVER1;key pressed, find row
1331    SJMP K2 ;if none, keep polling
1332    OVER1: MOV P1, #11111110B ;ground row 0
1333    MOV A,P2 ;read all columns
1334    ANL A,#00001111B ;mask unused bits
1335    CJNE A,#00001111B,ROW_0 ;key row 0, find col.
1336    MOV P1,#11111101B ;ground row 1
1337    MOV A,P2 ;read all columns
1338    ANL A,#00001111B ;mask unused bits
1339    CJNE A,#00001111B,ROW_1 ;key row 1, find col.
1340    MOV P1,#11111011B ;ground row 2
1341    MOV A,P2 ;read all columns
1342    ANL A,#00001111B ;mask unused bits
1343    CJNE A,#00001111B,ROW_2 ;key row 2, find col.
1344    MOV P1,#11110111B ;ground row 3
1345    MOV A,P2 ;read all columns
1346    ANL A,#00001111B ;mask unused bits
1347    CJNE A,#00001111B,ROW_3 ;key row 3, find col.
1348    LJMP K2 ;if none, false input, repeat
1349
1350    ROW_0: MOV DPTR,#KCODE0 ;set DPTR=start of row 0
1351    SJMP FIND ;find col. Key belongs to
1352    ROW_1: MOV DPTR,#KCODE1 ;set DPTR=start of row
1353    SJMP FIND ;find col. Key belongs to
1354    ROW_2: MOV DPTR,#KCODE2 ;set DPTR=start of row 2
1355    SJMP FIND ;find col. Key belongs to
1356    ROW_3: MOV DPTR,#KCODE3 ;set DPTR=start of row 3
1357    FIND: RRC A ;see if any CY bit low
1358    JNC MATCH ;if zero, get ASCII code
1359    INC DPTR ;point to next col. addr
1360    SJMP FIND ;keep searching
1361    MATCH: CLR A ;set A=0 (match is found)
1362    MOVC A,@A+DPTR ;get ASCII from table
1363    MOV P0,A ;display pressed key
1364    LJMP K1
1365
1366    ;ASCII LOOK-UP TABLE FOR EACH ROW
1367    ORG 300H
1368    KCODE0: DB '0','1','2','3' ;ROW 0
```

```
1369    KCODE1: DB '4','5','6','7' ;ROW 1
1370    KCODE2: DB '8','9','A','B' ;ROW 2
1371    KCODE3: DB 'C','D','E','F' ;ROW 3
1372
1373    ;---------- this is delay subroutine ------------
1374    ORG 400H ;put DELAY at address 300H
1375    DELAY: MOV R5,#0FFH ;R5=255 (FF in hex), counter
1376    AGAIN: DJNZ R5,AGAIN ;stay here until R5 become 0
1377    RET ;return to caller (when R5 =0)
1378
1379    END
1380
```