Chapter 4

1.

```
 1                                        ; 8086 PROGRAM    F4-01.ASM
 2                              ;ABSTRACT  : This program averages two temperatures
 3                                        ; named HI_TEMP and LO_TEMP and puts the
 4                                        ; result in the memory location AV_TEMP.
 5                              ;REGISTERS : Uses DS, CS, AX, BL
 6                              ;PORTS     : None used
 7
 8 0000                         DATA    SEGMENT
 9 0000  92                             HI_TEMP  DB  92H    ; Max temp storage
10 0001  52                             LO_TEMP  DB  52H    ; Low temp storage
11 0002  ??                             AV_TEMP  DB   ?     ; Store average here
12 0003                         DATA    ENDS
13
14 0000                         CODE    SEGMENT
15                                      ASSUME CS:CODE, DS:DATA
16 0000  B8 0000s               START:  MOV  AX, DATA       ; Initialize data segment
17 0003  8E D8                          MOV  DS, AX
18 0005  A0 0000r                       MOV  AL, HI_TEMP    ; Get first temperature
19 0008  02 06 0001r                    ADD  AL, LO_TEMP    ; Add second to it
20 000C  B4 00                          MOV  AH, 00H        ; Clear all of AH register
21 000E  80 D4 00                       ADC  AH, 00H        ; Put carry in LSB of AH
22 0011  B3 02                          MOV  BL, 02H        ; Load divisor in BL register
23 0013  F6 F3                          DIV  BL             ; Divide AX by BL. Quotient in AL,
24                                                          ; and remainder in AH
25 0015  A2 0002r                       MOV  AV_TEMP, AL    ; Copy result to memory
26 0018                         CODE    ENDS
27                                      END  START
```

Fig. 4.1 8086 program to average two temperatures.

2.

```
 1                                        ; 8086 PROGRAM F4-05.ASM
 2                              ;ABSTRACT  : Program produces a packed BCD byte from 2 ASCII-encoded digits
 3                                        ; The first ASCII digit (5) is loaded in BL.
 4                                        ; The second ASCII digit (9) is loaded in AL.
 5                                        ; The result (packed BCD) is left in AL
 6                              ;REGISTERS ; Uses CS, AL, BL, CL
 7                              ;PORTS     : None used
 8
 9 0000                         CODE    SEGMENT
10                                      ASSUME CS:CODE
11 0000  B3 35                  START:  MOV  BL, '5'   ; Load first ASCII digit into BL
12 0002  B0 39                          MOV  AL, '9'   ; Load second ASCII digit into AL
13 0004  80 E3 0F                       AND  BL, 0FH   ; Mask upper 4 bits of first digit
14 0007  24 0F                          AND  AL, 0FH   ; Mask upper 4 bits of second digit
15 0009  B1 04                          MOV  CL, 04H   ; Load CL for 4 rotates required
16 000B  D2 C3                          ROL  BL, CL    ; Rotate BL 4 bit positions
17 000D  0A C3                          OR   AL, BL    ; Combine nibbles, result in AL
18 000F                         CODE    ENDS
19                                      END START
```

Fig. 4.5 List file of 8086 assembly language program to produce packed BCD form two ASCII characters.

3.

```
 1                                        ; 8086 PROGRAM   F4-14A.ASM
 2                                ;ABSTRACT : Program section for PC board making machine.
 3                                        ; This program section reads the temperature of a cleaning bath
 4                                        ; solution and lights one of two lamps according to the
 5                                        ; temperature read. If the temp <30°C, a yellow lamp will be
 6                                        ; turned on. If the temp is ≥30°C, a green lamp will be turned on.
 7                                ;REGISTERS: Uses CS, AL, DX
 8                                ;PORTS    : Uses FFF8H - temperature input
 9                                        ; FFFAH - lamp control output (yellow=bit 0, green=bit 1)
10
11 0000                          CODE    SEGMENT
12                                       ASSUME CS:CODE
13                               ;initialize SDK-86 port FFFAH as output port, FFF8H as input port
14 0000  BA FFFE                         MOV DX, 0FFFEH      ; Point DX to port control register
15 0003  B0 99                           MOV AL, 99H         ; Load control word to initialize ports
16 0005  EE                              OUT DX, AL          ; Send control word to port control register
17
18 0006  BA FFF8                         MOV DX, 0FFF8H      ; Point DX at input port
19 0009  EC                              IN  AL, DX          ; Read temp from sensor on input port
20 000A  3C 1E                           CMP AL, 30          ; Compare temp with 30°C
21 000C  72 03                           JB  YELLOW          ; IF temp <30 THEN light yellow lamp
22 000E  EB 0A 90                        JMP GREEN           ; ELSE light green lamp
23 0011  B0 01                   YELLOW: MOV AL, 01H         ; Load code to light yellow lamp
24 0013  BA FFFA                         MOV DX, 0FFFAH      ; Point DX at output port
25 0016  EE                              OUT DX, AL          ; Send code to light yellow lamp
26 0017  EB 07 90                        JMP EXIT            ; Go to next mainline instruction
27 001A  B0 02                   GREEN:  MOV AL, 02H         ; Load code to light green lamp
28 001C  BA FFFA                         MOV DX, 0FFFAH      ; Point DX at output port
29 001F  EE                              OUT DX, AL          ; Send code to light green lamp
30 0020  BA FFFC                 EXIT:   MOV DX, 0FFFCH      ; Next mainline instruction
31 0023  EC                              IN  AL, DX          ; Read ph sensor
32 0024                          CODE    ENDS
33                                       END
```

(a)

```
20 000A  3C 1E                           CMP AL, 30          ; Compare temp with 30°C
21 000C  73 03                           JAE GREEN           ; IF temp ≥30 THEN light green lamp
22 000E  EB 0A 90                        JMP YELLOW          ; ELSE light yellow lamp
23 0011  B0 02                   GREEN:  MOV AL, 02H         ; Load code to light green lamp
24 0013  BA FFFA                         MOV DX, 0FFFAH      ; Point DX at output port
25 0016  EE                              OUT DX, AL          ; Send code to light green lamp
26 0017  EB 07 90                        JMP EXIT            ; Go to next mainline instruction
27 001A  B0 01                   YELLOW: MOV AL, 01H         ; Load code to light yellow lamp
28 001C  BA FFFA                         MOV DX, 0FFFAH      ; Point DX at output port
29 001F  EE                              OUT DX, AL          ; Send code to light yellow lamp
30 0020  BA FFFC                 EXIT:   MOV DX, 0FFFCH      ; Next mainline instruction
31 0023  EC                              IN  AL, DX          ; Read ph sensor
32 0024                          CODE    ENDS
33                                       END
```

(b)

Fig. 4.14 List file for printed-circuit-board-making machine program, (a) Below 30° version. (b) Program section for above 30° version

4.

```
 1                                      ; 8086 PROGRAM F4-16.ASM
 2                                ;ABSTRACT  : This program section reads the temperature of a cleaning bath
 3                                      ; solution and lights one of three lamps according to the
 4                                      ; temperature read. If the temp < 30°C, a yellow lamp will be
 5                                      ; turned on. If the temp ≥ 30° and < 40°, a green lamp will be
 6                                      ; turned on. Temperatures ≥ 40° will turn on a red lamp.
 7                                ;REGISTERS : Uses CS, AL, DX
 8                                ;PORTS     : Uses FFF8H - temperature input
 9                                      ;      FFFAH - lamp control output, yellow=bit 0, green=bit 1, red=bit 2
10 0000                           CODE   SEGMENT
11                                       ASSUME CS:CODE
12                                ;initialize port FFFAH for output and port FFF8H for input
13 0000  BA FFFE                         MOV DX, 0FFFEH      ; Point DX to port control register
14 0003  B0 99                           MOV AL, 99H         ; Load control word to set up output port
15 0005  EE                              OUT DX, AL          ; Send control word to control register
16
17 0006  BA FFF8                         MOV DX, 0FFF8H      ; Point DX at input port
18 0009  EC                              IN  AL, DX          ; Read temp from sensor on input port
19 000A  BA FFFA                         MOV DX, 0FFFAH      ; Point DX at output port
20 000D  3C 1E                           CMP AL, 30          ; Compare temp with 30°C
21 000F  72 0A                           JB  YELLOW          ; IF temp < 30 THEN light yellow lamp
22 0011  3C 28                           CMP AL, 40          ; ELSE compare with 40°
23 0013  72 0C                           JB  GREEN           ;  IF temp < 40 THEN light green lamp
24 0015  B0 04                   RED:    MOV AL, 04H         ;  ELSE temp ≥ 40 so light red lamp
25 0017  EE                              OUT DX, AL          ; Send code to light red lamp
26 0018  EB 0A 90                        JMP EXIT            ; Go to next mainline instruction
27 001B  B0 01                   YELLOW: MOV AL, 01H         ; Load code to light yellow lamp
28 001D  EE                              OUT DX, AL          ; Send code to light yellow lamp
29 001E  EB 04 90                        JMP EXIT            ; Go to next mainline instruction
30 0021  B0 02                   GREEN:  MOV AL, 02H         ; Load code to light green lamp
31 0023  EE                              OUT DX, AL          ; Send code to light green lamp
32 0024  BA FFFC                 EXIT:   MOV DX, 0FFFCH      ; Next mainline instruction
33 0027  EC                              IN  AL, DX          ; Read ph sensor
34 0028                          CODE    ENDS
35                                       END
```

5.

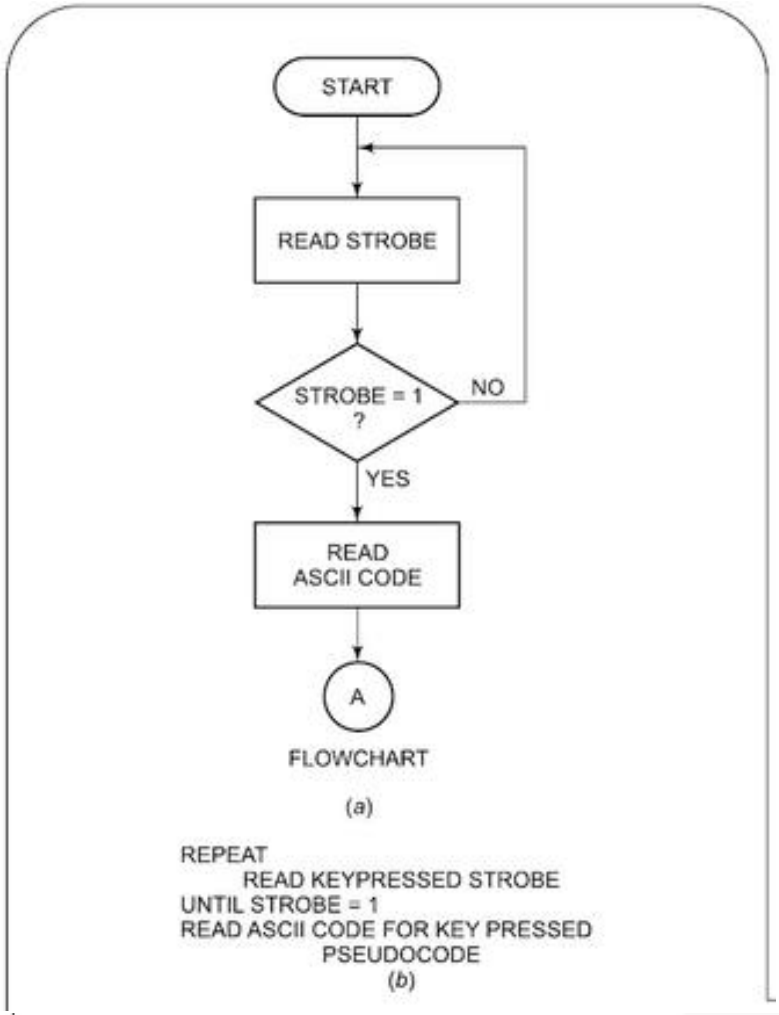
```
 1                                        ; 8086 PROGRAM   F4-18A.ASM
 2                                ;ABSTRACT  : Program turns heater off if temperature ≥ 100°C
 3                                        ; and turns heater on if temperature < 100°C.
 4                                ;REGISTERS : Uses CS, DX, AL
 5                                ;PORTS     : Uses FFF8H - temperature data input
 6                                        ;      FFFAH - MSB for heater control output, 0=off, 1=on
 7 0000                           CODE    SEGMENT
 8                                        ASSUME  CS:CODE
 9                                ; Initialize port FFFAH for output, and port FFF8H for input
10 0000  BA FFFE                          MOV  DX, 0FFFEH    ; Point DX to port control register
11 0003  B0 99                            MOV  AL, 99H       ; Control word to set up output port
12 0005  EE                               OUT  DX, AL        ; Send control word to port
13
14 0006  BA FFF8                 TEMP_IN:   MOV  DX, 0FFF8H    ; Point at input port
15 0009  EC                               IN   AL, DX        ; Input temperature data
16 000A  3C 64                            CMP  AL, 100       ; If temp ≥ 100 then
17 000C  73 08                            JAE  HEATER_OFF    ; turn heater off
18 000E  B0 80                            MOV  AL, 80H       ; else load code for heater on
19 0010  BA FFFA                          MOV  DX, 0FFFAH    ; Point DX to output port
20 0013  EE                               OUT  DX, AL        ; Turn heater on
21 0014  EB F0                            JMP  TEMP_IN       ; WHILE temp < 100 read temp again
22 0016  B0 00                   HEATER_OFF:MOV  AL, 00      ; Load code for heater off
23 0018  BA FFFA                          MOV  DX, 0FFFAH    ; Point DX to output port
24 001B  EE                               OUT  DX, AL        ; Turn heater off
25 001C                          CODE       ENDS
26                                        END
```

```
14 0006  BA FFF8                 TEMP_IN:   MOV  DX, 0FFF8H    ; Point DX at input port
15 0009  EC                               IN   AL, DX        ; Read in temperature data
16 000A  3C 64                            CMP  AL, 100       ; If temp < 100° then
17 000C  72 03                            JB   HEATER_ON     ; turn heater on
18 000E  EB 09 90                         JMP  HEATER_OFF    ; else temp ≥100 so turn heater off
19 0011  B0 80                   HEATER_ON: MOV  AL, 80H      ; Load code for heater on
20 0013  BA FFFA                          MOV  DX, 0FFFAH    ; Point DX at output port
21 0016  EE                               OUT  DX, AL        ; Turn heater on
22 0017  EB ED                            JMP  TEMP_IN       ; WHILE temp < 100° read temp again
23 0019  B0 00                   HEATER_OFF:MOV  AL, 00      ; Load code for heater off
24 001B  BA FFFA                          MOV  DX, 0FFFAH    ; Point DX at output port
25 001E  EE                               OUT  DX, AL        ; Turn heater off
26 001F                          CODE       ENDS
27                                        END
```

(b)

Fig. 4.18 List file for heater control program, (a) First approach, (b) Improved version of WHILE-DO section of program.

6.

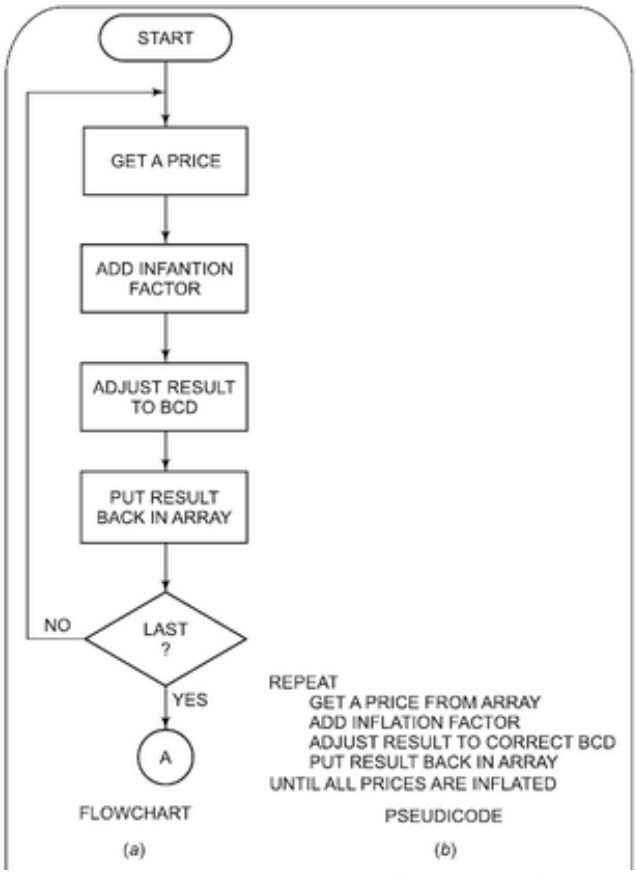



```
 1                                      ; 8086 PROGRAM F4-20C.ASM
 2                                ;ABSTRACT  : Program to read ASCII code after a strobe signal
 3                                      ; is sent from a keyboard
 4                                ;REGISTERS : Uses CS, DX, AL
 5                                ;PORTS     : Uses FFFAH - strobe signal input on LSB
 6                                      ;      FFF8H - ASCII data input port
 7
 8 0000                           CODE        SEGMENT
 9                                            ASSUME CS:CODE
10 0000  BA FFFA                             MOV DX, 0FFFAH    ; Point DX at strobe port
11 0003  EC                      LOOK_AGAIN: IN  AL, DX        ; Read keyboard strobe
12 0004  24 01                               AND AL, 01        ; Mask extra bits and set flags
13 0006  74 FB                               JZ  LOOK_AGAIN    ; If strobe is low then keep looking
14 0008  BA FFF8                             MOV DX, 0FFF8H    ; else point DX at data port
15 000B  EC                                  IN  AL, DX        ; Read in ASCII code
16 000C                          CODE        ENDS
17                                           END
```

(c)

Fig. 4.20 Flowchart, pseudocode, and assembly language for reading ASCII code when a strobe is present. (a) Flowchart, (b) Pseudocode, (c) List file program.

7.




```
 1                                      ; 8086 PROGRAM  : F4-21C.ASM
 2                                ;ABSTRACT  : Program adds an inflation factor to a series of prices
 3                                      ; in memory. It copies the new price over the old price.
 4                                ;REGISTERS : Uses DS, CS, AX, BX, CX
 5                                ;PORTS     : None used
 6
 7 0000                           ARRAYS   SEGMENT
 8 0000  20 28 15 26 19 27 16 +           COST     DB     20H, 28H, 15H, 26H, 19H, 27H, 16H, 29H
 9       29
10 0008  36 55 27 42 38 41 29 +           PRICES   DB     36H, 55H, 27H, 42H, 38H, 41H, 29H, 39H
11       39
12 0010                          ARRAYS   ENDS
13
14 0000                          CODE     SEGMENT
15                                        ASSUME CS:CODE, DS:ARRAYS
16 0000  B8 0000s                START:   MOV  AX, ARRAYS   ; Initialize data segment
17 0003  8E D8                            MOV  DS, AX        ; register
18 0005  8D 1E 0008r                      LEA  BX, PRICES    ; Initialize pointer
19 0009  B9 0008                          MOV  CX, 0008H     ; Initialize counter
20 000C  8A 07                   DO_NEXT: MOV  AL, [BX]      ; Copy a price to AL
21 000E  04 03 .                          ADD  AL, 03H       ; Add inflation factor
22 0010  27                               DAA                ; Make sure result is BCD
23 0011  88 07                            MOV  [BX], AL      ; Copy result back to memory
24 0013  43                               INC  BX            ; Point to next price
25 0014  49                               DEC  CX            ; Decrement counter
26 0015  75 F5                            JNZ  DO_NEXT       ; If not last, go get next
27 0017                          CODE     ENDS
28                                        END  START
```

(c)

Fig. 4.21 Adding a constant to a series of values in memory, (a) Flowchart, (b) Pseudocode, (c) List file of program.

8.

```
 1                                          ; 8086 PROGRAM F4-23.ASM
 2                              ;ABSTRACT  : Program adds a profit factor to each element in a
 3                                          ; COST array and puts the result in a PRICES array.
 4                              ;REGISTERS : Uses DS, CS, AX, BX, CX
 5                              ;PORTS     : None used
 6
 7       = 0015                 PROFIT     EQU      15H        ; profit = 15 cents
 8 0000                         ARRAYS     SEGMENT
 9 0000  20 28 15 26 19 27 16 + COST       DB  20H, 28H, 15H, 26H, 19H, 27H, 16H, 29H
10       29
11 0008  08*(00)               PRICES     DB  8 DUP(0)
12 0010                        ARRAYS     ENDS
13
14 0000                        CODE       SEGMENT
15                                        ASSUME  CS:CODE, DS:ARRAYS
16 0000  B8 0000s              START:     MOV  AX, ARRAYS      ; Initialize data segment
17 0003  8E D8                            MOV  DS, AX          ; register
18 0005  B9 0008                          MOV  CX, 0008H       ; Initialize counter
19 0008  BB 0000                          MOV  BX, 0000H       ; Initialize pointer
20 000B  8A 87 0000r           DO_NEXT:   MOV  AL, COST[BX]    ; Get element [BX] from COST
21 000F  04 15                            ADD  AL, PROFIT      ; Add the profit to value
22 0011  27                               DAA                  ; Decimal adjust result
23 0012  88 87 0008r                      MOV  PRICES[BX], AL  ; Store result in PRICES at [BX]
24 0016  43                               INC  BX              ; Point to next element in arrays
25 0017  49                               DEC  CX              ; Decrement the counter
26 0018  75 F1                            JNZ  DO_NEXT         ; If not last element, do again
27 001A                        CODE       ENDS
28                                        END  START
```

Fig. 4.23 List file of "price-calculating" program.

Chapter 5

1.

```
                                        ; 8086 PROGRAM F5-03.ASM
                              ;ABSTRACT : This program inputs a password and sounds an alarm
                                        ; if the password is incorrect
                              ;REGISTERS : Uses CS, DS, ES, AX, DX, CX, SI, DI
                              ;PORTS     : Uses FFFAH - Port 2B on SDK-86 for alarm output

0000                          DATA  SEGMENT
0000  46 41 49 4C 53 41 46 +       PASSWORD    DB   'FAILSAFE'     ; Password
      45
      = 0008                       STR_LENGTH  EQU  ($ - PASSWORD) ; Compute length of string
0008  08*(00)                     INPUT_WORD  DB   8 DUP(0)       ; Space for user password input
0010                          DATA  ENDS

0000                          CODE  SEGMENT
                                    ASSUME CS:CODE, DS:DATA, ES:DATA
0000  B8 0000s                       MOV  AX, DATA
0003  8E D8                          MOV  DS, AX           ; Initialize data segment register
0005  8E C0                          MOV  ES, AX           ; Initialize extra segment register
0007  BA FFFE                        MOV  DX, 0FFFEH       ; These next three instructions
000A  B0 99                          MOV  AL, 99H          ; set up an output port on
000C  EE                             OUT  DX, AL           ; the SDK-86 board
000D  8D 36 0000r                    LEA  SI, PASSWORD     ; Load source pointer
0011  8D 3E 0008r                    LEA  DI, INPUT_WORD   ; Load destination pointer
0015  B9 0008                        MOV  CX, STR_LENGTH   ; Load counter with password length
0018  FC                             CLD                   ; Increment DI & SI
0019  F3> A6                  REPE   CMPSB                 ; Compare the two string bytes
001B  75 03                          JNE  SOUND_ALARM      ; If not equal, sound alarm
001D  EB 08 90                       JMP  OK               ;  else continue
0020  B0 01                  SOUND_ALARM:MOV  AL, 01       ; To sound alarm, send a 1
0022  BA FFFA                        MOV  DX, 0FFFAH       ; to the output port whose
0025  EE                             OUT  DX, AL           ; address is in DX
0026  F4                             HLT                   ; and HALT.
0027  90                     OK:     NOP                   ; Program continues if password is OK
0028                          CODE    ENDS
                                      END
```

Fig. 5.3 Assembly language program for comparing strings.

2.

```
                                        ; 8086 PROGRAM F5-10.ASM
                                ;ABSTRACT  : This program takes in data samples from a port at 1 ms
                                        ; intervals, masks the upper 4 bits of each sample, and
                                        ; puts each masked sample in successive locations in an array.
                                ;REGISTERS : Uses CS, SS, DS, AX, BX, CX, DX, SI, SP
                                ;PORTS     : Uses 0FFF8H - data samples input from port P2A on SDK-86
                                ;PROCEDURES: Uses WAIT_1MS

= FFF8                          PRESSURE_PORT   EQU  0FFF8H

0000                            DATA      SEGMENT
0000  64*(0000)                       PRESSURES       DW   100  DUP(0)       ; Set up array of 100 words
= 0064                                 NBR_OF_SAMPLES  EQU  (($-PRESSURES)/2)
00C8                            DATA      ENDS

0000                            STACK_SEG  SEGMENT
0000  28*(0000)                           DW   40  DUP(0)     ; set stack length of 40 words
                                STACK_TOP  LABEL    WORD
0050                            STACK_SEG  ENDS

0000                            CODE       SEGMENT
                                           ASSUME  CS:CODE, DS:DATA, SS:STACK_SEG
0000  B8 0000s                  START:     MOV  AX, DATA             ; Initialize data segment register
0003  8E D8                               MOV  DS, AX
0005  B8 0000s                            MOV  AX, STACK_SEG        ; Initialize stack segment register
0008  8E D0                               MOV  SS, AX
000A  BC 0050r                            MOV  SP, OFFSET STACK_TOP ; Intialize stack pointer to top of stack

000D  8D 36 0000r                         LEA  SI, PRESSURES        ; Point SI to start of array
0011  BB 0064                             MOV  BX, NBR_OF_SAMPLES   ; Load BX with number of samples
0014  BA FFF8                             MOV  DX, PRESSURE_PORT    ; Point DX at input port
0017  ED                        NEXT_VALUE:IN   AX, DX               ; Read data from port
0018  25 0FFF                             AND  AX, 0FFFH            ; Mask upper 4 bits
001B  89 04                               MOV [SI],AX               ; Store data word in array
001D  E8 0006                             CALL WAIT_1MS             ; Delay 1 ms
0020  46                                  INC  SI                   ; Point SI at next location in array
0021  46                                  INC  SI
0022  4B                                 DEC  BX                   ; Decrement sample counter
0023  75 F2                               JNZ  NEXT_VALUE           ; Repeat until 100 samples done
0025  90                        STOP:      NOP

0026                            WAIT_1MS   PROC     NEAR
0026  B9 23F2                            MOV  CX, 23F2H            ; Load delay constant into CX
0029  E2 FE                     HERE:     LOOP HERE                 ; Loop until CX = 0
002B  C3                                  RET
002C                            WAIT_1MS   ENDP

002C                            CODE       ENDS
                                           END START
```

Fig. 5.10 Assembly language program to read in 100 samples of data at 1-ms intervals.