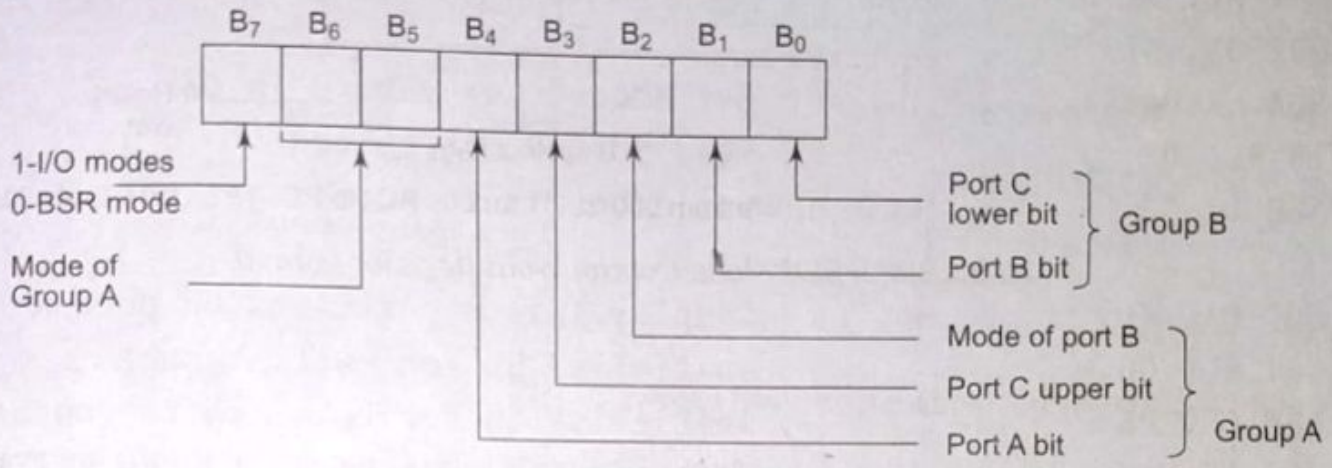
Thus 82H is the control word for the requirements in the problem. The port address decoding can be done as given below. The 8255 is to be interfaced with lower order data bus, i.e. D_0–D_7. The A_0 and A_1 pins of 8255 are connected to A_{01} and A_{02} pins of the microprocessor respectively. The A_{00} pin of the microprocessor is used for selecting the transfer on the lower byte of the data bus. Hence any change in the status of A_{00} does not affect the port to be selected, rather A_{01} and A_{02} of the microprocessor decide the port to be selected as they are connected to A_0 and A_1 of 8255. The 8255 port addresses are tabulated as shown below.



Group A modes

B_6	B_5	Mode
0	0	mode 0
0	1	mode 1
1	0	mode 2
1	1	x

(i) Port B mode is either 0 or 1 depending upon B2 bit.

(ii) A port is an output port if the port bit is 0 else it is input port

Fig. 5.18(b) *I/O Mode Control Word Register Format*

8255	*I/O Address lines*																*Hex. Port Addresses*
Ports	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}	
PortA	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0740H
Port B	0	0	0	0	0	1	1	1	0	1	0	0	0	0	1	0	0742H
Port C	0	0	0	0	0	1	1	1	0	1	0	0	0	1	0	0	0744H
CWR	0	0	0	0	0	1	1	1	0	1	0	0	0	1	1	0	0746H

Let us use absolute decoding scheme that uses all the 16 address lines for deriving the device address pulse. Out of $A_0 - A_{15}$ lines, two address lines A_{02} and A_{01} are directly required by 8255 for the three port and CWR address decoding. Hence only A_3 to A_{15} are used for decoding addresses. The complete hardware scheme is shown in Fig. 5.19. In the diagram, the 8086 is assumed to be in

the maximum mode so that $\overline{IORD}$ and $\overline{IOWR}$ are readily available. If the 8086 is in minimum mode, $\overline{RD}$ and $\overline{WR}$ of 8086 are to be connected accordingly to 8255 and M/ $\overline{IO}$ pin is combined with the chip select of above hardware suitably so as to select the 8255 when M/$\overline{IO}$ is low.

The ALP for the problem is developed as follows:

```
    MOV DX, 0746 H          : Initialise CWR with
    MOV AL, 82 H            : control word 82H
    OUT DX, AL              :
    SUB DX,04               : Get address of port B in DX
    IN AL, DX               : Read port B for switch
    SUB DX,02               : positions in to AL and get port A address
                            : in DX.
    OUT DX, AL              : Display switch positions on port A
    MOV BL, 00 H            : Initialise BL for switch count
    MOV CH, 08H             : Initialise CH for total switch number
YY: ROL AL                  : Rotate AL through carry to check,
    JNC XX                  : whether the switches are on or
    INC BL                  : off, i.e. either 1 or 0
XX :DEC CH                  : Check for next switch. If
    JNZ YY                  : all switch are checked, the
    MOV AL, BL              : number of on switches are
    ADD DX, 04              : in BL.Display it on port C
    OUT DX,AL               : lower.
    HLT                     : Stop
```

Program 5.5 *ALP for Problem 5.10*

Problem 5.11

Interface a 4*4 Keyboard with 8086 using 8255. and write an ALP for detecting a key closure and return the key code in AL. The debouncing period for a key is 10 ms. Use software key debouncing technique. DEBOUNCE is an available 10 ms delay routine.

Solution Port A is used as output port for selecting a row of keys while port B is used as an input port for sensing a closed key. Thus the keyboard lines are selected one by one through port A and the port B lines are polled continuously till a key closure is sensed. Then routine DEBOUNCE is called for key debouncing. The key code is decided depending upon the selected row and a low sensed column. The hardware circuit diagram is shown in Fig. 5.21.

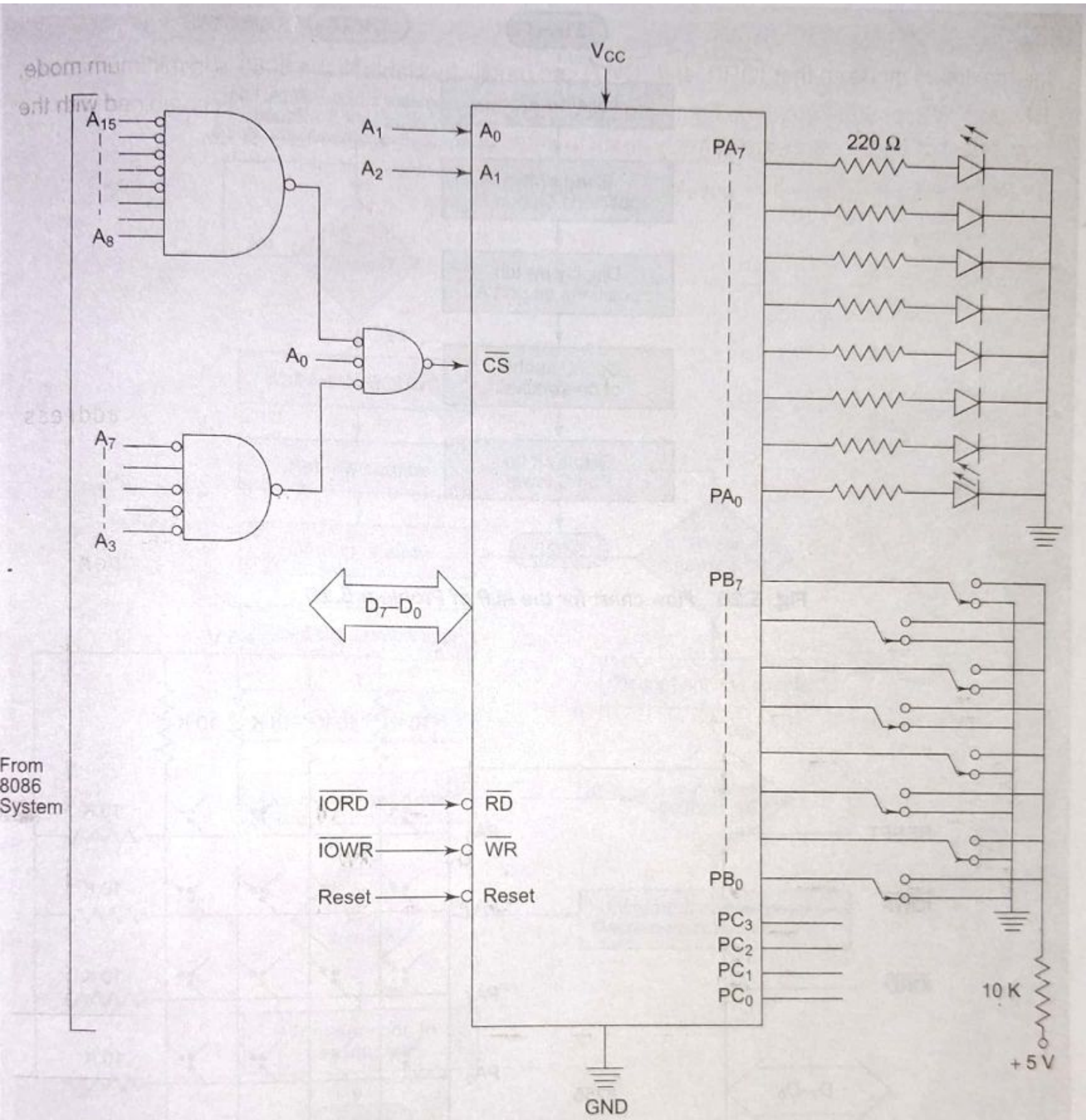


Fig. 5.19 *8255 Interfacing with 8086 for Problem 5.10*

The higher order lines of port A and port B are left unused. The addresses of port A and port B will be respectively 8000 H and 8002 H while the address of CWR will be 8006 H. The flow chart of the complete program is given in Fig. 5.22.

The ALP for the problem is given along with comments. The control word for this problem will be 82 H. Let us write this program using assembler directives. In this problem no major data is required hence only

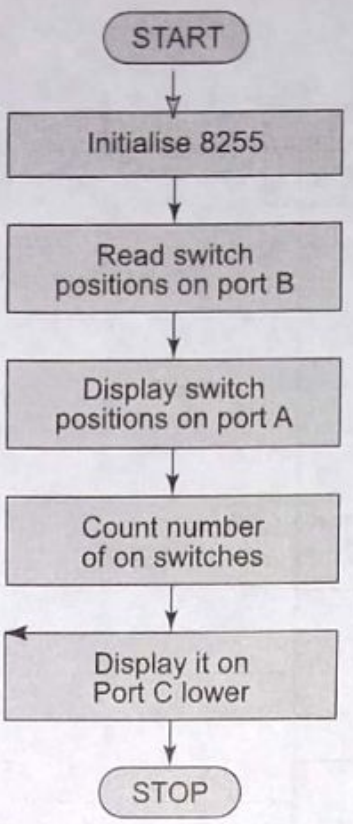


Fig. 5.20 *Flow chart for the ALP of Problem 5.10*

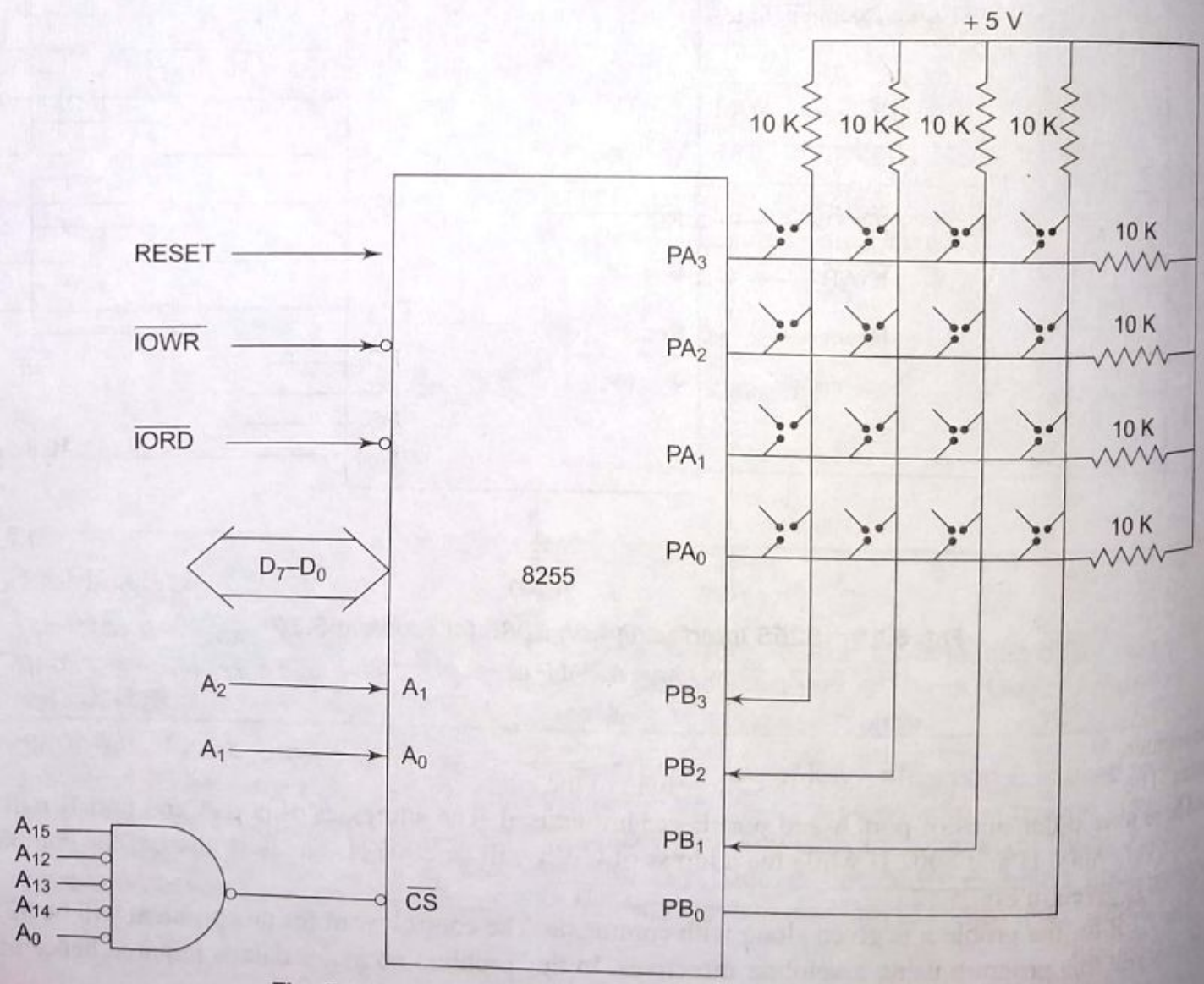


Fig. 5.21 *Interfacing 4 × 4 Keyboard for Problem 5.11*

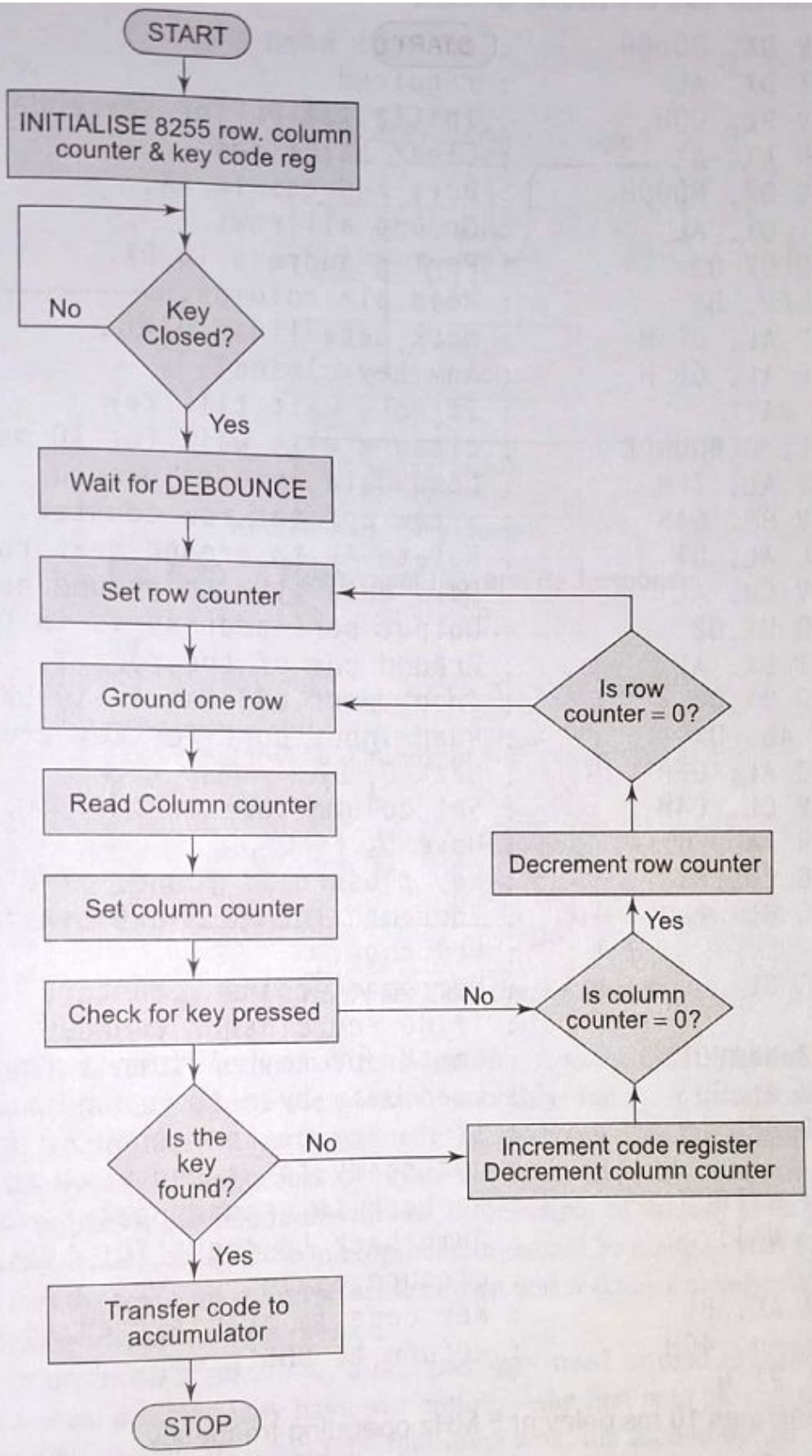


Fig. 5.22 *Flow chart for ALP of Problem 5.11*

one segment is used for storing the program code, i.e. code segment (CS). This program is written in MASM syntax. The 8255 is again interfaced to the lower byte of the 8086 data bus. Absolute decoding scheme is not used here to implement the circuit using minimum hardware.

```
CODE        SEGMENT
ASSUME      CS : CODE
START:      MOV AL, 82H          ; Load CWR with
```

```
            MOV DX, 8006H       ; control word
            OUT DX, AL          ; required
            MOV BL, 00H         ; Initialize BL for key code
            XOR AX, AX          ; Clear all flags
            MOV DX, 8000H       ; Port Address in AX.
            OUT DX, AL          ; Ground all rows.
            ADD DX,02           ; Port B address in DX.
WAIT :      IN AL, DX           ; Read all columns.
            AND AL, 0F H        ; Mask data lines D7-D4.
            CMP AL, 0F H        ; Any key closed?
            JZ WAIT             ; If not, wait till key
            CALL DEBOUNCE       ; closure else wait for 10 ms
            MOV AL, 7FH         ; Load data byte to ground
            MOV BH, 04H         ; a row and set row counter.
NXTROW :    ROL AL, 01          ; Rotate AL to ground next row.
            MOV CH, AL          ; Save data byte to ground next row.
            SUB DX,02           ; Output port address is in DX.
            OUT DX, AL          ; Ground one of the rows.
            ADD DX,02           ; Input port address is in DX.
            IN AL, DX           ; Read input port for key closure.
            AND AL, 0FH         ; Mask D4-D7.
            MOV CL, 04H         ; Set column counter.
NXTCOL :    ROR  AL, 01         ; Move D0 in CF.
            JNC CODEKY          ; Key closure is found, if CF=0.
            INC BL              ; Increment BL for next binary
                                ; key code.
            DEC CL              ; Decrement column counter,
                                ; if no key closure found.
            JNZ NXTCOL          ; Check for key closure in next column
            MOV AL,CH           ; Load data byte to ground next row.
            DEC BH              ; if no key closer found in column
                                ; get ready to ground next row.
            JNZ NXTROW          ; Go back to ground next row.
            JMP WAIT            ; Jump back to check for key.
                                ; closure again.
CODEKY :    MOV AL, BL          ; Key code is transferred to AL.
            MOV AH, 4CH         ; Return to DOS prompt.
            INT 21 H
```

This procedure generates 10 ms delay at 5 MHz operating frequency.

```
DEBOUNCE PROC NEAR
            MOV CL, 0E2H
BACK:       NOP
            DEC CL
            JNZ BACK
            RET
DEBOUNCE    ENDP
CODE        ENDS
            END START
```

Program 5.6 *ALP for Problem 5.11*

Key Debounce Whenever a mechanical push-button is pressed or released once, the mechanical components of the key do not change the position smoothly, rather, it generates a transient response as shown in Fig. 5.23. These transient variations may be interpreted as the multiple key pressures and responded accordingly by the microprocessor system. To avoid this problem, two schemes are suggested: the first one utilizes a bistable multivibrator at the output of the key to debounce it as shown in Fig. 5.24. The other scheme suggests that the microprocessor should be made to wait for the transient period (usually 10 msec), so that the transient response settles down and reaches a steady state. A logic '0' will be read by the microprocessor when the key is pressed.

In a number of high precision applications, a designer may need to read or write more than 8-bits of data. In these cases, a system designer may have two options—the first is to have more than one 8-bit port, read(write) the ports one by one and then form the multibyte data; the second option allows forming 16-bit ports using two 8-bit ports and use 16-bit read or write operations. The following example elaborates interfacing of a 16-bit port using two 8-bit ports.

Problem 5.12

Interface 16-bit 8255 ports with 8086. The address of port A is F0H.

Solution To implement a 16-bit port two 8255s are required. One will act as the lower 8-bit port, i.e. D_0–D_7, while the other will act as the upper 8-bit port D_8–D_{15}. The overall scheme is as shown in Fig. 5.25. While initialising AL and AH (AX) both should be loaded with a suitable (common) control word. In this system, port A, port B and port C all may work as 16-bit ports.

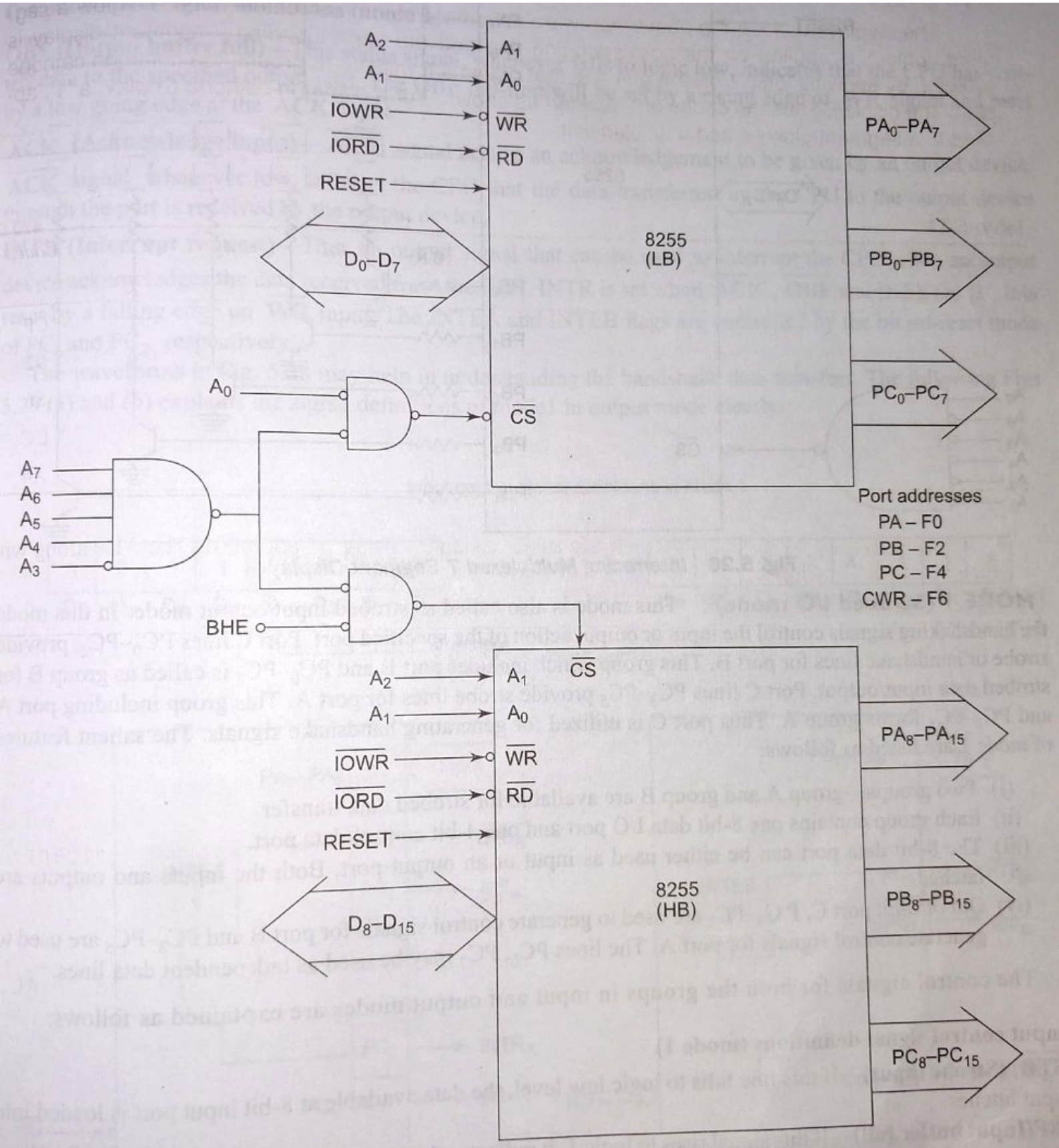


Fig. 5.25 *Interfacing 16-bit 8255 ports with 8086*

Problem 5.13

Interface an 8255 with 8086 at 80H as an I/O address of port A. Interface five 7 segment displays with the 8255. Write a sequence of instructions to display 1, 2, 3, 4 and 5 over the five displays continuously as per their positions starting with 1 at the least significant position.

Solution The hardware scheme for the above problem is shown in Fig. 5.26. In this scheme, I/O port A is multiplexed to carry data for all the 7-segment displays. The port B selects (grounds) one of the displays at a time.

The displays used in the above hardware scheme are common cathod type. To glow a segment, logic 1 is applied on the corresponding line and the corresponding 7-segment display is selected by applying logic 1 on the port line that drives a transistor to ground the common cathode pin of the display. Thus the codes are decided as shown. For a common cathode display, a '1', applied to a segment glows it and a '0' blanks it.

Table 5.11

Number to be displayed	PA_7 *dp*	PA_6 *a*	PA_5 *b*	PA_4 *c*	PA_3 *d*	PA_2 *e*	PA_1 *f*	PA_0 *g*	*Code*
1	1	1	0	0	1	1	1	1	CF
2	1	0	0	1	0	0	1	0	92
3	1	0	0	0	0	1	1	0	86
4	1	1	0	0	1	1	0	0	CC
5	1	0	1	0	0	1	0	0	A4

All these codes, decided as above, are stored in a look up table starting at 2000:0001. The ALP along with comments is given as follows:

```
AGAIN:   MOV CL, 05H          ; Count for displays
         MOV BX, 2000H        ; Initialise data segment
         MOV DS, BX           ; for look up table
         MOV CH, 01H          ; 1st number to be displayed
         MOV AL, 80H          ; Load control word in the
         OUT 86H,AL           ; CWR
         MOV DL,01H           ; Enable code for Least significant
                              ; 7-seg display
NXTDGT : MOV BX, 0000H        ; Set pointer to look up table
         MOV AL, CH           ; First no to display
                              ; Store number to be displayed in AL.
         XLAT                 ; Find code from look up table
         OUT 80H,AL           ; Display the code
         MOV AL, DL           ; Enable the display
         OUT 81H,AL           ;
         ROL DL               ; Go for selecting the next display
```

```
INC CH                    ; Next number to display
DEC CL                    ; Decrement count.
JNZ NXTDGT                ; Go for next digit display
JMP AGAIN                 ; Repeat the procedure
```

Program 5.7 *ALP for Problem 5.13*

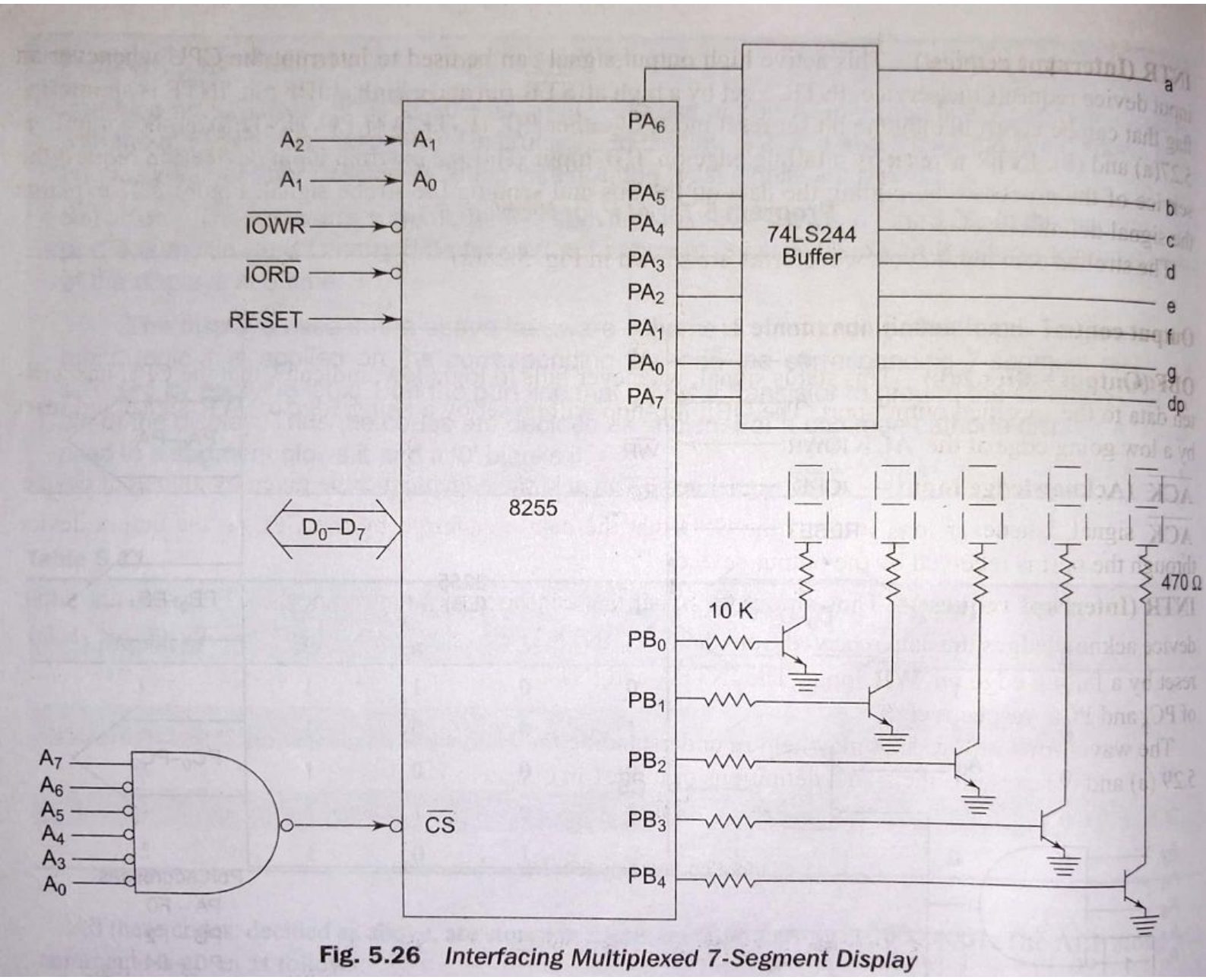


Fig. 5.26 *Interfacing Multiplexed 7-Segment Display*

(b) MODE - 1 (Strobed I/o mode) :-

* Handshaking signals control the input - output action of the specified port.
* Group A – Port A + Port C ($PC_3 - PC_5$ as strobe lines)
* Group B – Port B + Port C ($PC_0 - PC_2$ as strobe lines)
* PC_6 and PC_7 may be used as independent data lines.

Input control signal definitions (mode 1)

* $\overline{STB}$ (Strobe Input) : $\overline{STB} = 0$ ⇒ data available at 8-bit input port is loaded into the input latches.
* IBF (Input Buffer full) : IBF = 1 ⇒ acknowledgement of data loaded into the latches.
* INTR (Interrupt Request) : INTR = 1 ⇒ input device requests the CPU

Output control signal definition (mode 1)

* $\overline{OBF}$ (Output Buffer Full) : $\overline{OBF} = 0$ ⇒ CPU has written data to the specific o/p port
* $\overline{ACK}$ (Acknowledge input) : $\overline{ACK} = 0$ ⇒ data transferred by the CPU through o/p port has reached the device
* INTR (Interrupt Request) : INTR = 1 ⇒ Output device can interrupt the CPU when the data is received by it. (Further data can be written to output device).

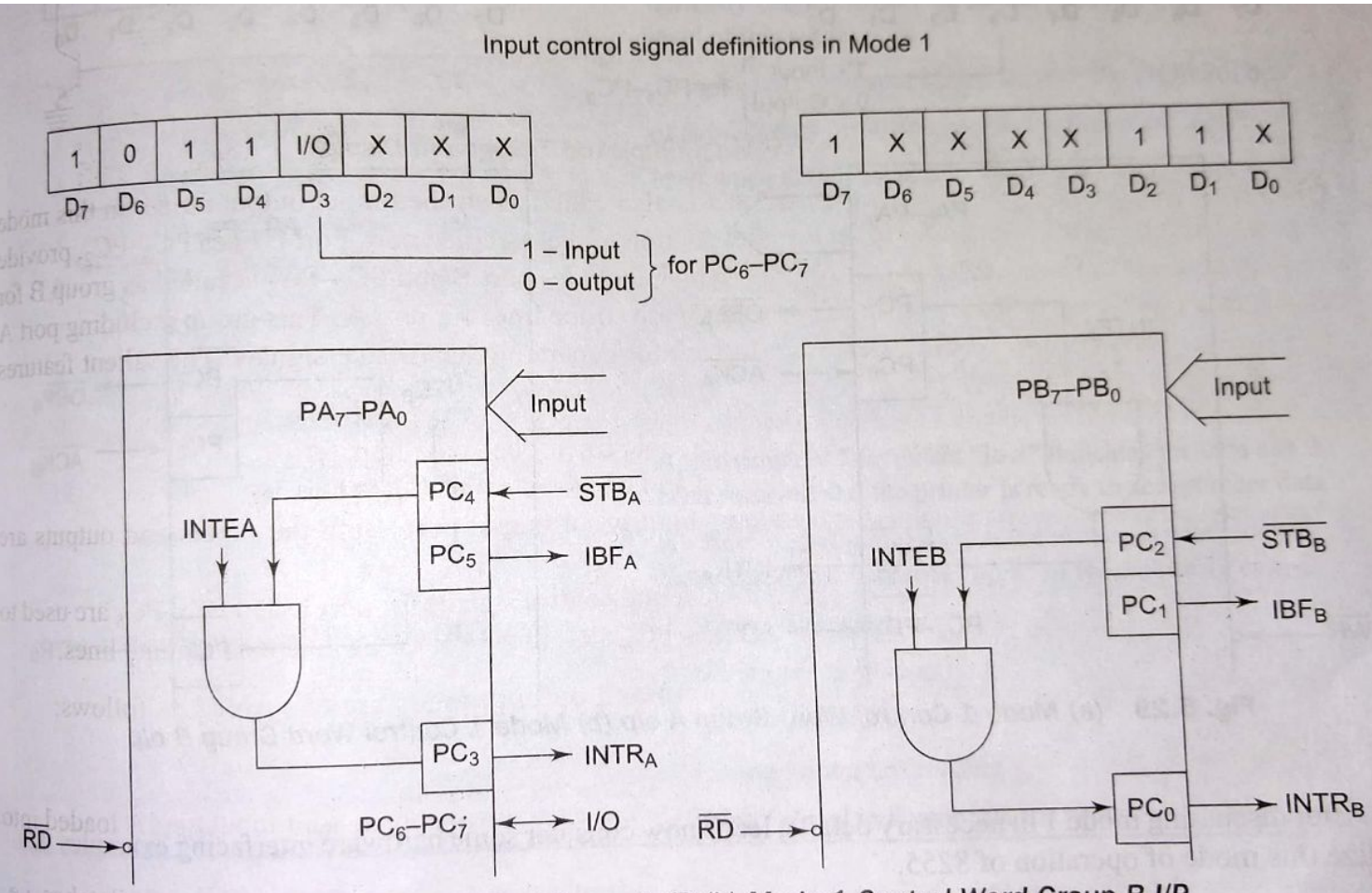


Fig. 5.27 *(a) Mode1 Control Word Group A I/P (b) Mode 1 Control Word Group B I/P*

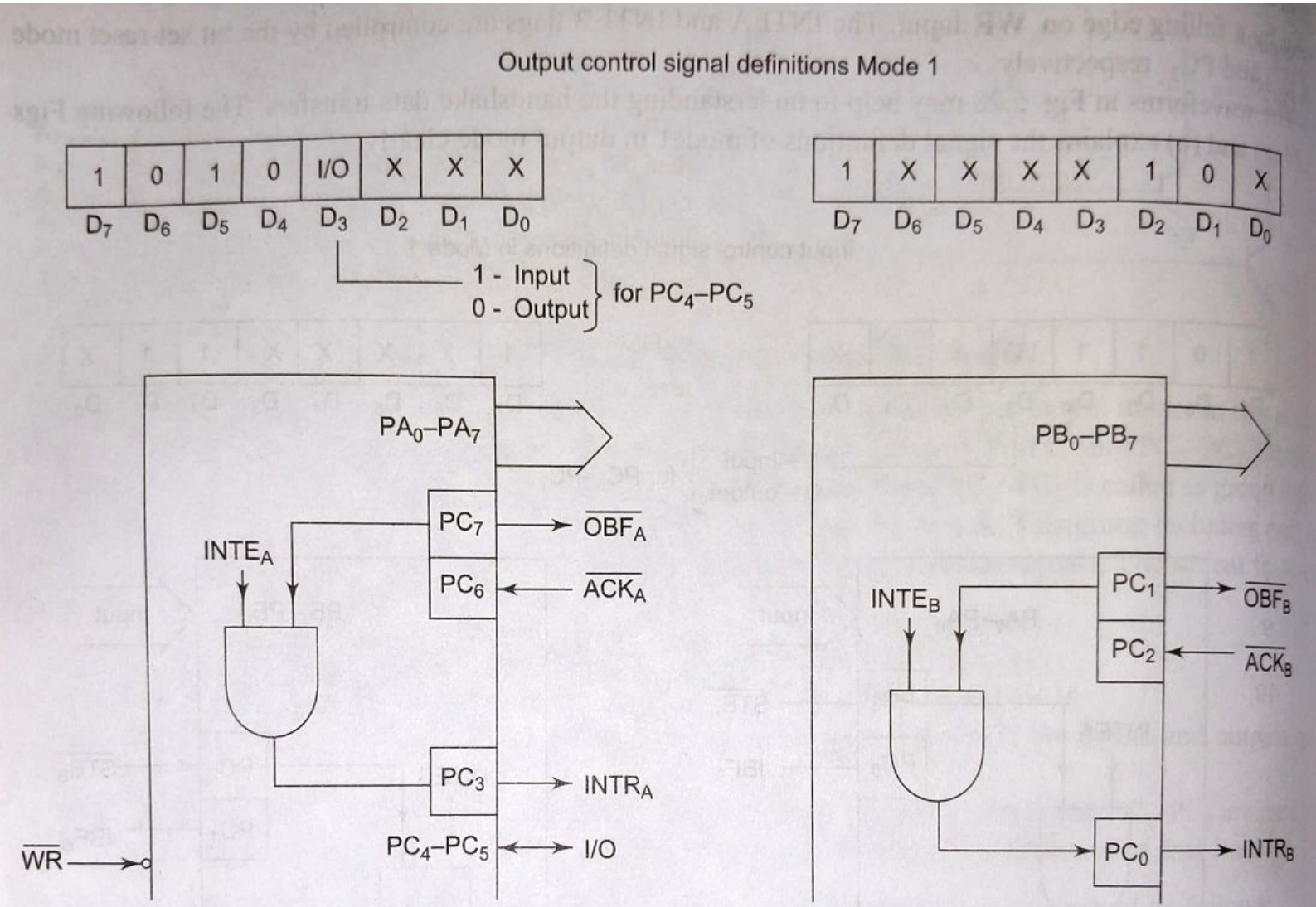


Fig. 5.29 *(a) Mode 1 Control Word Group A o/p (b) Mode 1 Control Word Group B o/p*

Problem 5.14

Interface a standard IEEE-488 parallel bus printer with 8086. Draw the necessary hardware scheme required for the same and write an ALP to print a character whose ASCII code is available in AL.

Solution Before going through this solution, one should refer to the standard Centronix, INB or EPSON printer pin configuration, given in Table 5.12. There are two types of parallel cables used to connect a microcomputer with a printer, viz. 25 pin cables and 36 pin cables. Basically the 25 pin and the 36 pin cables are similar except for the 11 extra pins for ground (GND) used as 'RETURN' lines for different signals.

The group A is used in mode 1 for handshake data transfer so that port A is used for data transfer and port C lines PC_3–PC_5 are used as handshake lines. Port B lines are used for checking the printer status, hence port B is used as input port in mode 0. Port C lower is used as output port for enabling the printer. The control words are shown in Fig. 5.30.

Table 5.12 *Pin Connections and Descriptions for Centronix-type Parallel Interface to IBM PC and EPSON FX-100 Printers*

Printer Controller				
Signal	*Return*	*Signal*	*Direction*	*Description*
Pin No.	*Pin No.*			
1	19	$\overline{\text{STROBE}}$	IN	STROBE pulse to read data in. Pulse width must be more than 0.5 µs at receiving terminal. The signal level is normally "high"; read-in of data is performed at the "low" level of this signal.
2	20	DATA 1	IN	These signals represent information of the 1st to 8th bits of parallel data respectively. Each signal is at "high" level when data is logical "1" and "low" when logical "0".
3	21	DATA 2	IN	
4	22	DATA 3	IN	
5	23	DATA 4	IN	
6	24	DATA 5	IN	
7	25	DATA 6	IN	
8	26	DATA 7	IN	
9	27	DATA 8	IN	
10	28	$\overline{\text{ACKNLG}}$	OUT	Approximately 5 µs pulse; "low" indicates the data has been received and the printer is ready to accept other data.
11	29	BUSY	OUT	A "high" signal indicates that the printer cannot receive data. The signal becomes "high" in the following cases. 1. During data entry. 2. During printing operation. 3. In "outline" state. 4. During printer error status.
12	30	PE	OUT	A "high" signal indicates that the printer is out of paper.