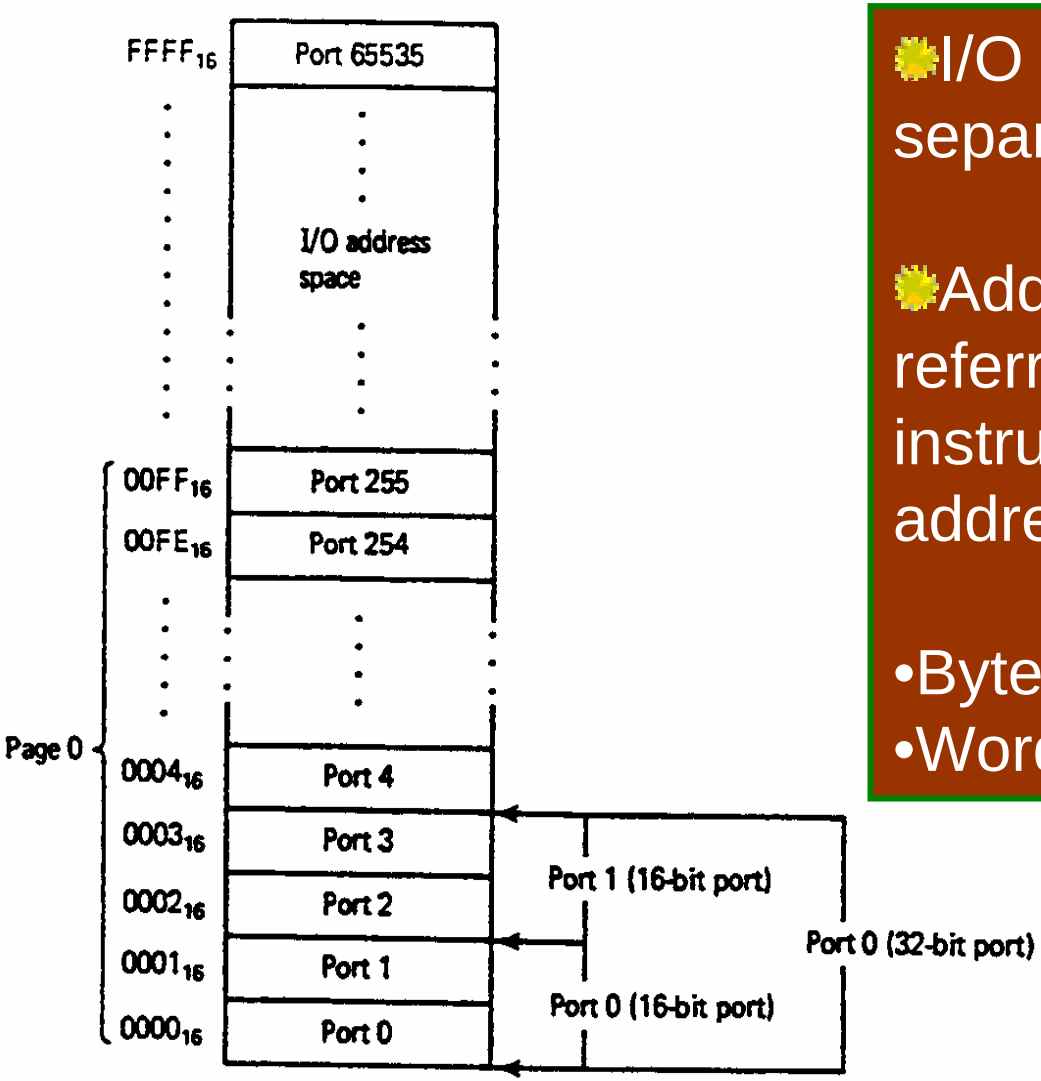
Weeks 9-10-11
Input/Output Interface Circuits and LSI Peripheral Devices

Core and Special Purpose I/O Interfaces

- Special purpose I/O interfaces are implemented as add-on cards on the PC
  - display
  - parallel printer interface
  - serial communication interface
  - local area network interface
  - not all microcomputer systems employ each of these types
- Core input/output interfaces are considered to be the part of the I/O subsystem such as:
  - parallel I/O to read the settings of the DIP switches on the processor board
  - interval timers used in DRAM refresh process
- We will study both

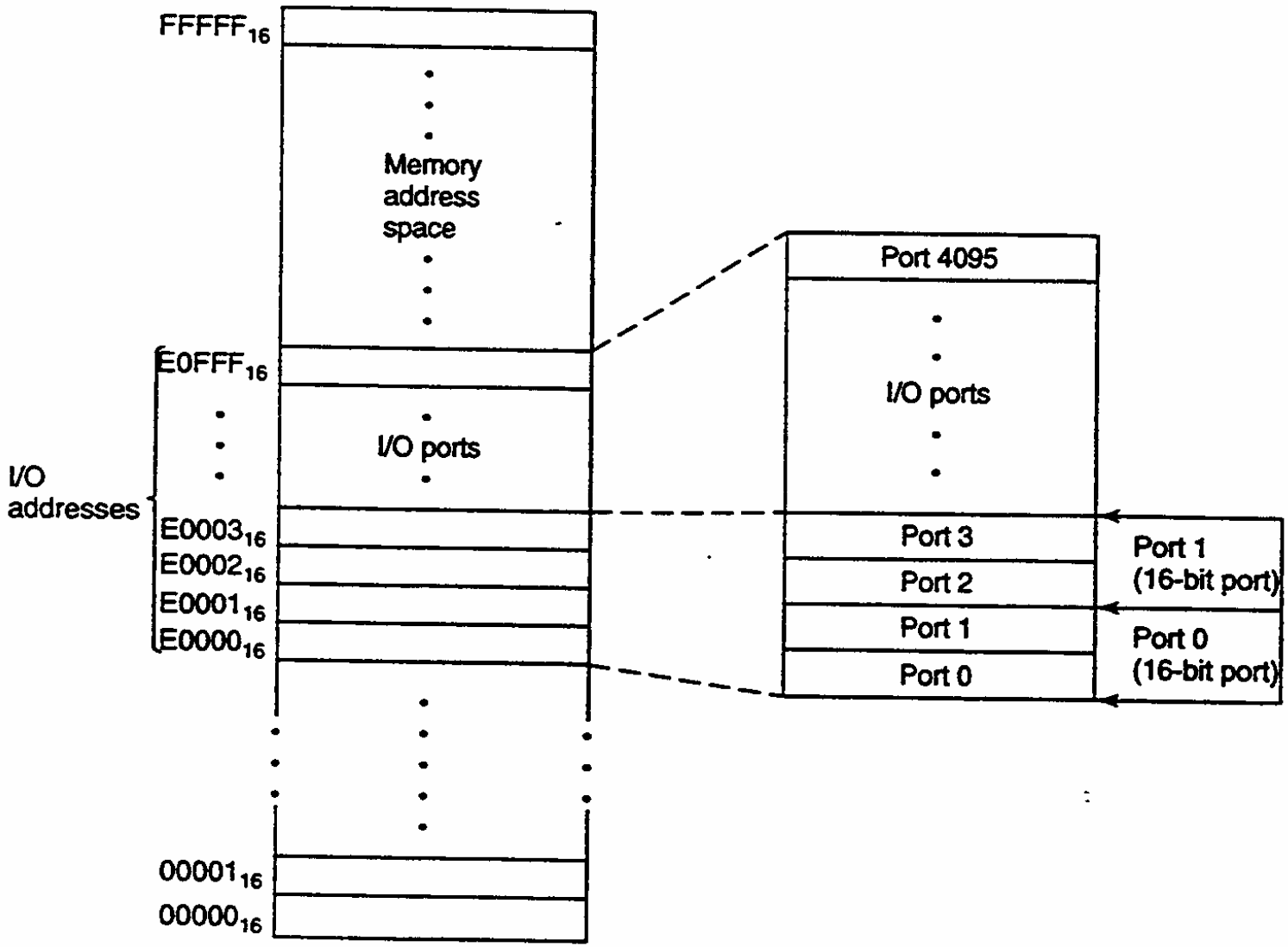
Isolated I/O



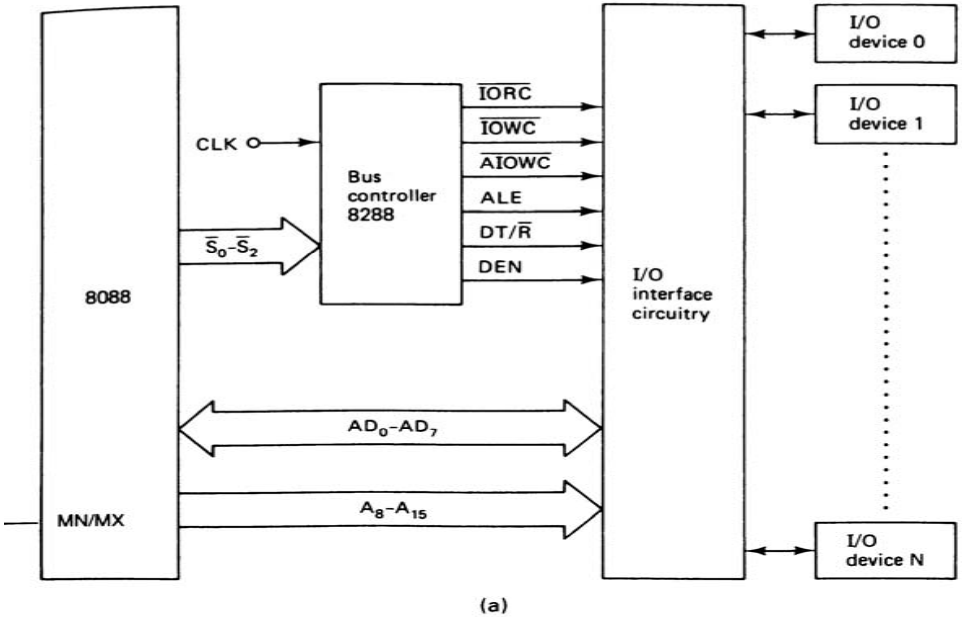
- I/O devices are treated separately from memory
- Address 0000 to 00FF: referred to page 0. Special instructions exist for this address range

•Byte wide ports
•Word wide ports

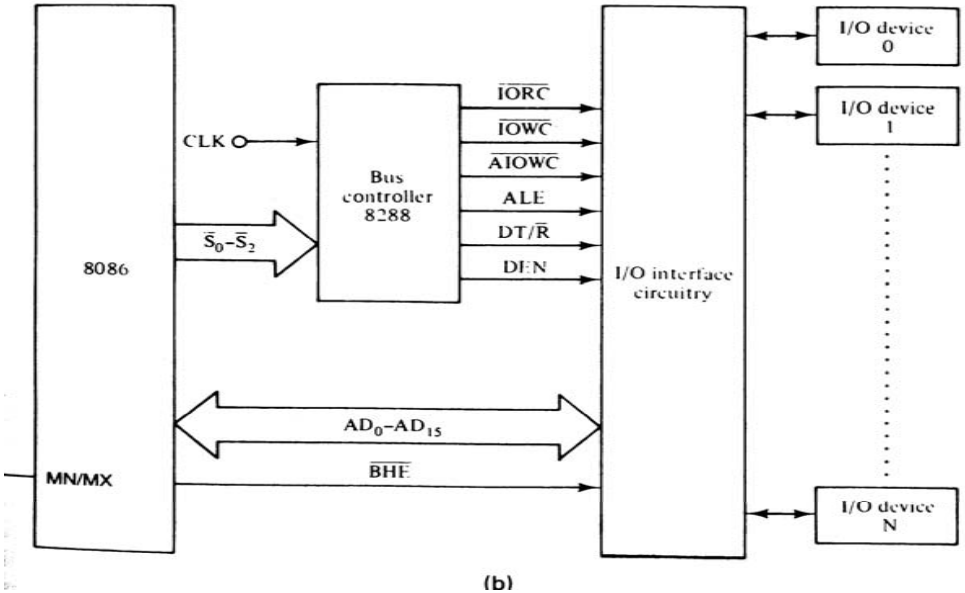
Memory Mapped I/O



Maximum Mode I/O interface - 8088/8086



(a)



(b)

I/O Bus Cycle Status Codes

Status inputs			CPU cycle	8288 command
$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$		
0	0	0	Interrupt acknowledge	$\overline{INTA}$
0	0	1	Read I/O port	$\overline{IORC}$
0	1	0	Write I/O port	$\overline{IOWC}$, $\overline{AIOWC}$
0	1	1	Halt	None
1	0	0	Instruction fetch	$\overline{MRDC}$
1	0	1	Read memory	$\overline{MRDC}$
1	1	0	Write memory	$\overline{MWTC}$, $\overline{AMWC}$
1	1	1	Passive	None

I/O Instructions

Mnemonic	Meaning	Format	Operation	
IN	Input direct	IN Acc,Port	(Acc) ← (Port)	Acc = AL or AX
	Input indirect (variable)	IN Acc,DX	(Acc) ← ((DX))	
OUT	Output direct	OUT Port,Acc	(Port) ← (Acc)	
	Output indirect (variable)	OUT DX,Acc	((DX)) ← (Acc)	

Example. Write a sequence of instructions that will output the data FFh to a byte wide output at address ABh of the I/O address space

```
MOV AL,0FFh
OUT 0ABh, AL
```

Example. Data is to be read from two byte wide input ports at addresses AAh and A9h and then this data will then be output to a word wide output port at address B000h

```
IN AL, 0AAh
MOV AH,AL
IN AL, 0A9h
MOV DX,0B00h
OUT DX,AX
```

Input Bus Cycle of the 8088

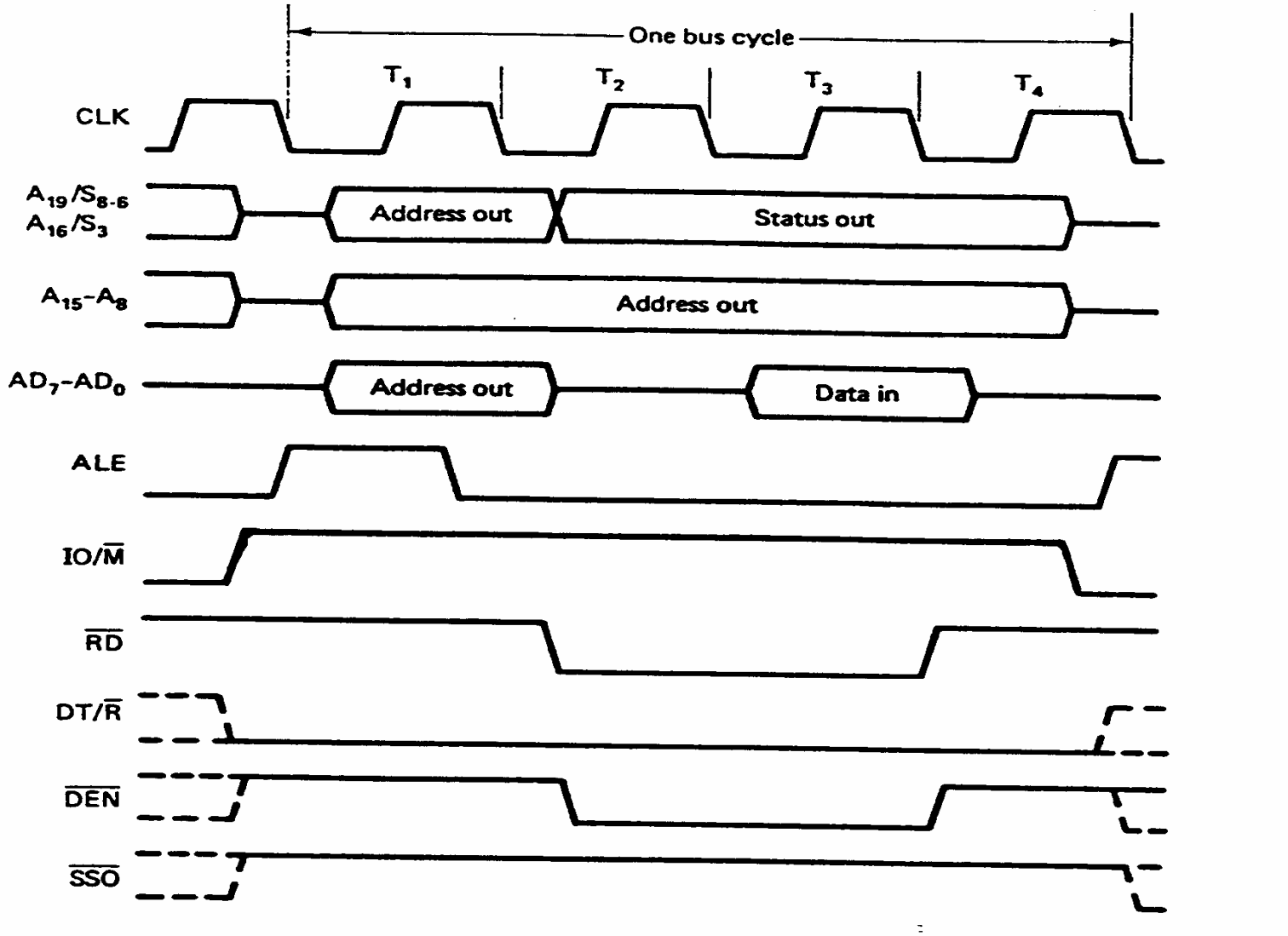


Figure 8–52 Input bus cycle of the 8088.

Output Bus Cycle of the 8088

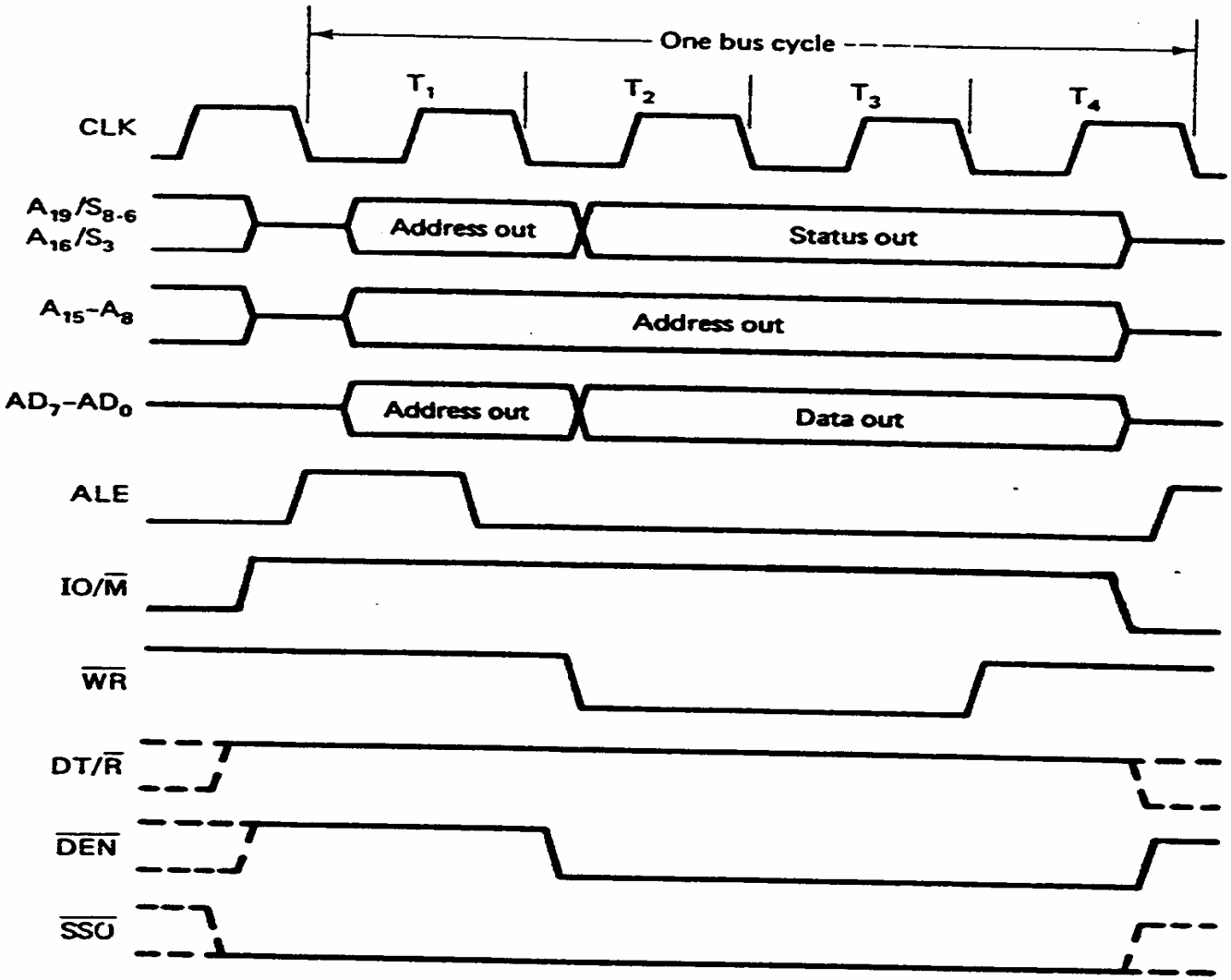


Figure 8–53 Output bus cycle of the 8088.

I/O Example

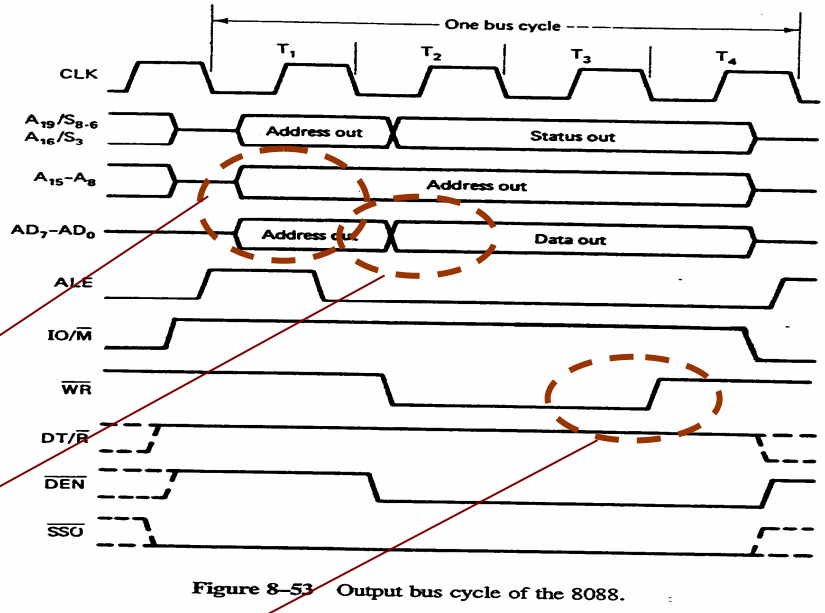
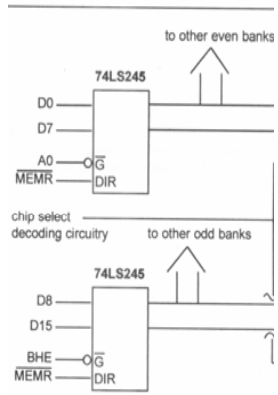


Figure 8–53 Output bus cycle of the 8088.

- Assume that AX = 76A9h.: Analyze the data transfer for a) 8088 b) 8086 when
  MOV DX, 648h
  OUT DX,AX
- 8088 case
  - 1st bus cycle
    - T1: Address 0648h is put on pins AD0-AD7, A8-15 and latched when ALE is activated
    - T2: The low byte A9h is put on the data bus pins AD0-AD7 and $\overline{\text{IOWC}}$ is activated
    - T3: Setup time
    - T4: Byte is written to the port assuming zero wait states
  - 2nd Bus Cycle (Similar to 1st Bus Cycle)
    - T1: Address 0649h is put on pins AD0-AD7, A8-15 and latched when ALE is activated
    - T2: The high byte 76h is put on the data bus pins and $\overline{\text{IOWC}}$ is activated
    - T3: Setup time
    - T4: Byte is written to the port assuming zero wait states

Example continued

- 8086 case
  - T1: Address 0648h is put on pins AD0-AD15 plus BHE=low is latched by the 74LS373 when ALE is activated
  - T2: 76A9h, the contents of AX, is put on AD0-AD15 (A9h on AD0-AD7, 76h on AD8-AD15) and IOWC is activated
  - T3: Setup time
  - T4: During this interval, with the help of the signals A0=0 and BHE=0, the low and high bytes are written to the appropriate ports.
    - ➢It must be noted that since the operand is a 16 bit word and the port address is an even address, the 8086 CPU does not generate address 0649h
    - ➢Port address 648h is connected to the D0-D7 data bus and port address 649h is connected to the D8-D15 data bus

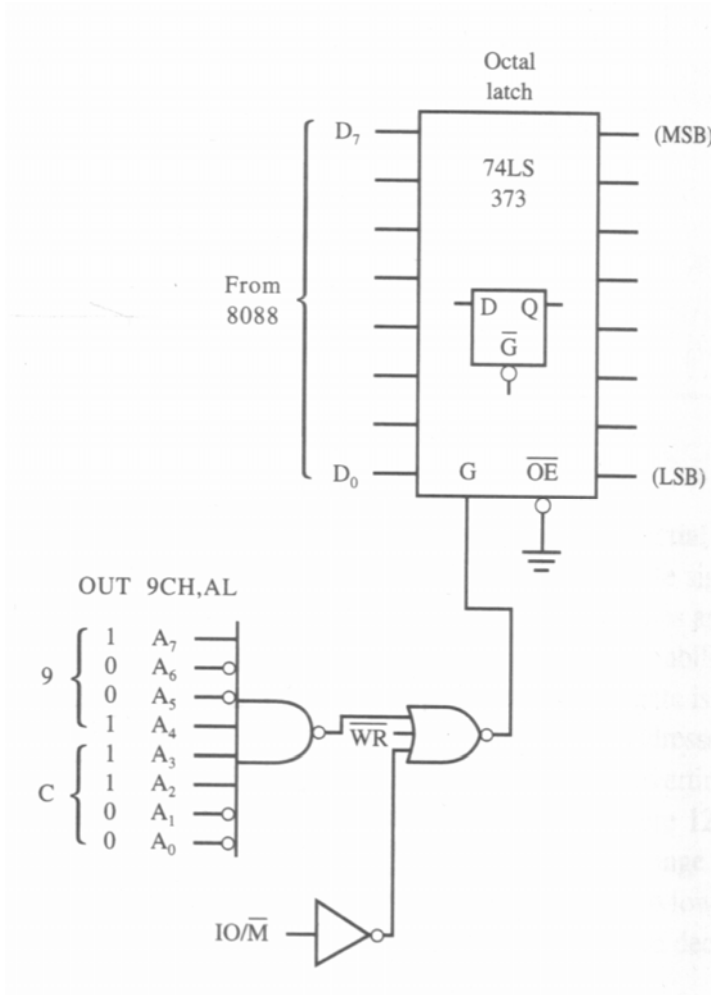


I/O Design in the 8088/86

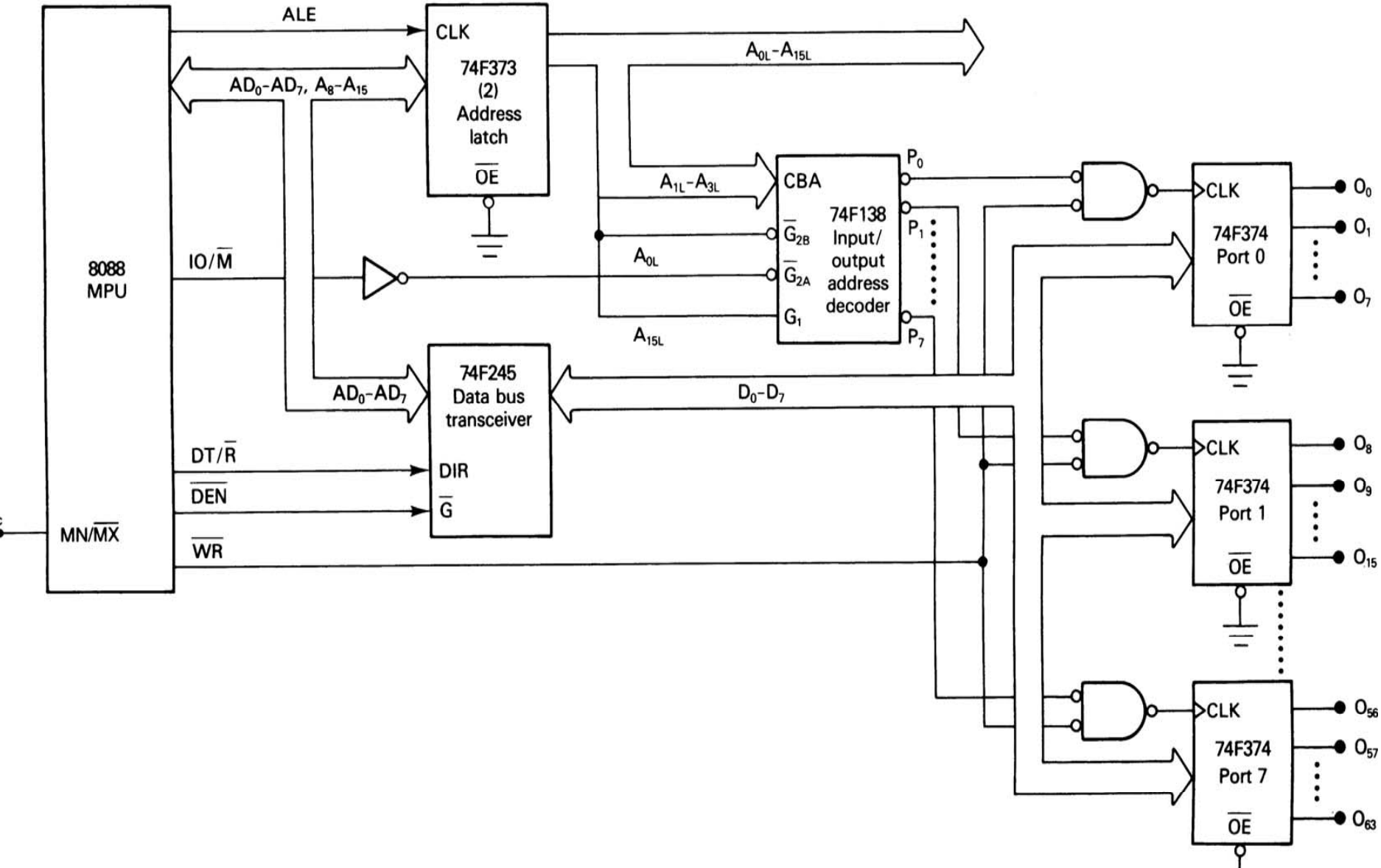
- In every computer, when data is sent out by the CPU, the data on the data bus must be latched by the receiving device
- While memories have an internal latch to grab the data on the data bus, a latching system must be designed for ports
- Since the data provided by the CPU to the port is on the system data bus for a limited amount of time (50 - 1000ns) it must be latched before it is lost
- Likewise, when data is coming in by way of a data bus (either from port or memory) it must come in through a three-state buffer

I/O Design

- Design for OUT 9CH,AL



Example - 64 line parallel output circuit - 8088



Examples

- To which output port in the previous figure are data written when the address put on the bus during an output bus cycle is 8002h?
  - A15 .. A0 = 1000 0000 0000 0010b
  - A15L = 1
  - A0L = 0
  - A3L A2L A1L = 001
  - $\overline{P1}$ = 0

- Write a sequence of instructions that output the byte contents of the memory address DATA to output port 0 in the previous figure

```
MOV DX, 8000h
MOV AL,DATA
OUT DX, AL
```

Time Delay Loop and Blinking a LED at an Output

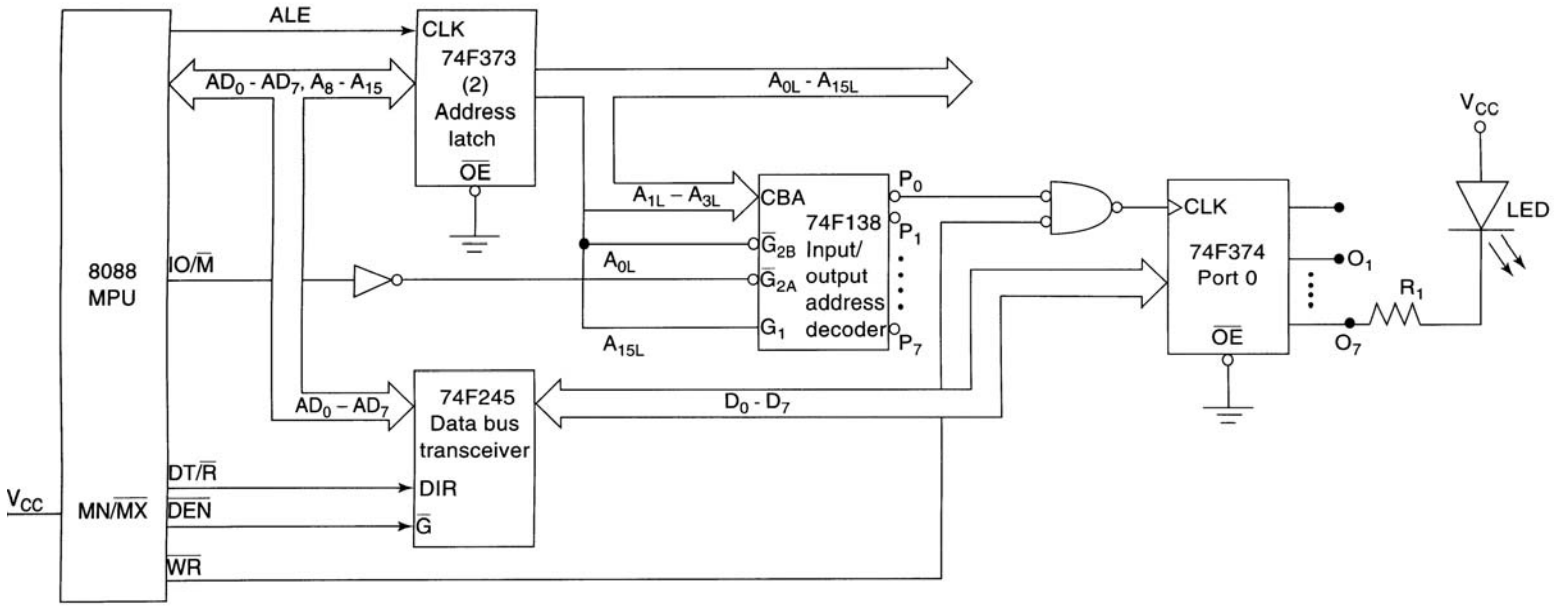


Figure 10–2 Driving an LED connected to an output port.

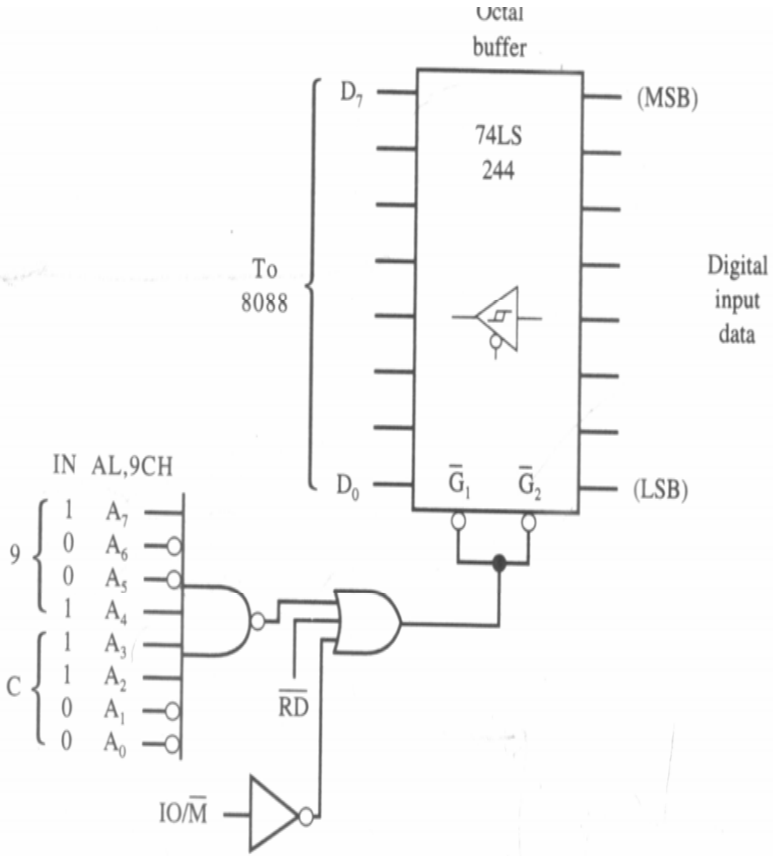
```
        MOV DX, 8000h     ;initialize address of port 0
        MOV AL, 00h       ; load data with bit 7 as logic 0
ON_OFF: OUT DX,AL         ; turned on
        MOV CX,0FFFFh     ; load delay count of FFFFh
HERE:   LOOP HERE
        XOR  AL,80h       ; complement bit 7
        JMP   ON_OFF
```

Aprox.

17 T states * 64K * Frequency

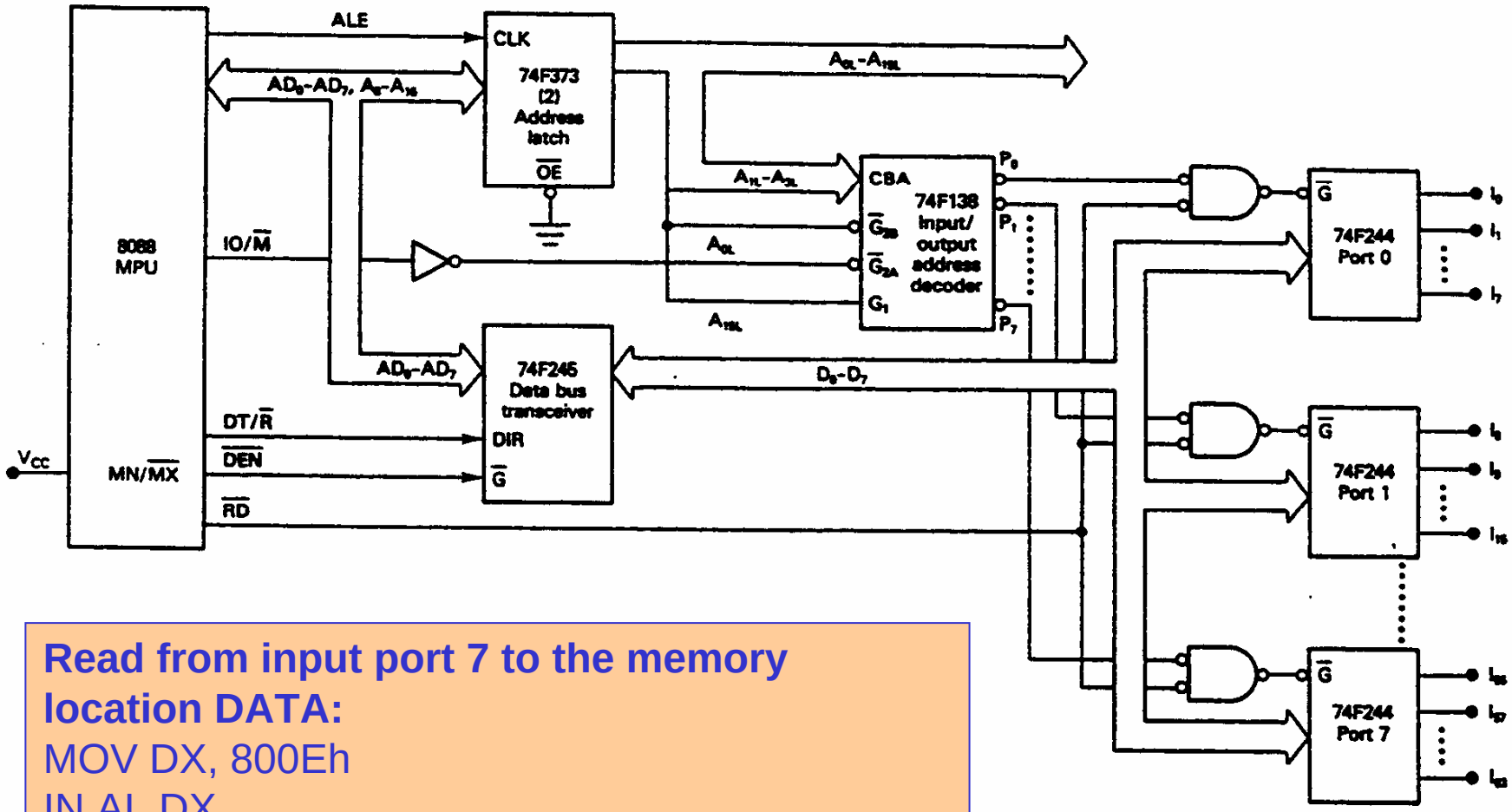
IN port design using the 74LS244

➢Design for IN AL,9CH



- In order to prevent any unwanted data (garbage) to come into the system (global) data bus, all input devices must be isolated through the tri-state buffer. The 74LS244 not only plays this role but also provides the incoming signals sufficient strength (driving capability) to travel all the way to the CPU
- It must be emphasized that every device (memory, peripheral) connected to the global data bus must have a latch or a tri-state buffer. In some devices such as memory, they are internal but must be present.

Example - 64 line parallel input circuit



Read from input port 7 to the memory location DATA:

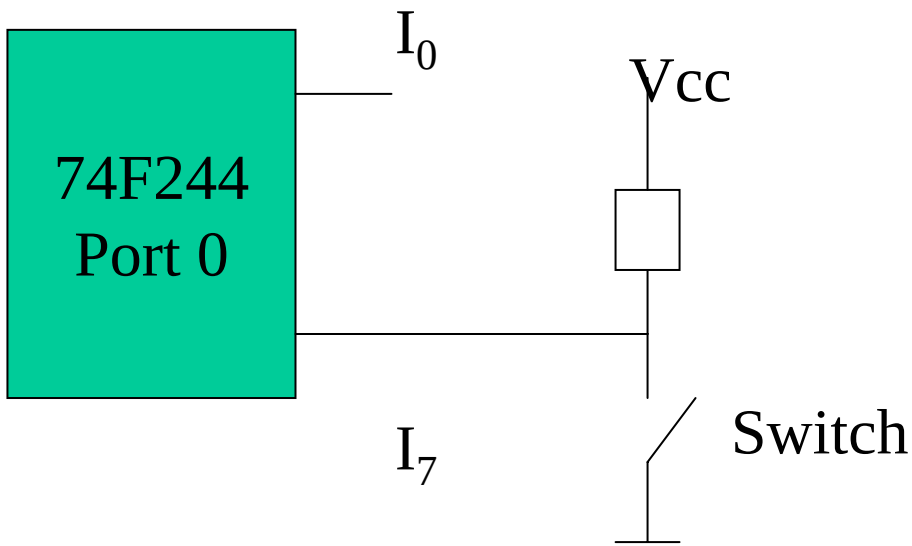
MOV DX, 800Eh

IN AL,DX

MOV DATA, AL

Example

- In practical applications, it is sometimes necessary within an I/O service routine to repeatedly read the value at an input line and test this value for a specific logic level.

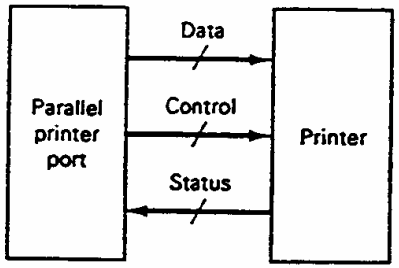


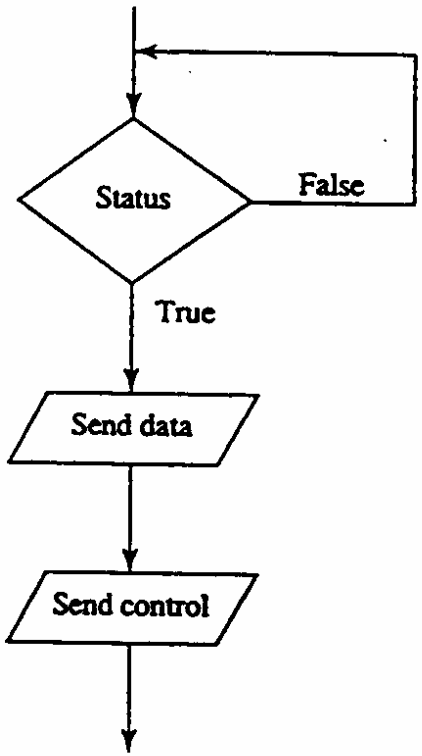

```
Poll the switch waiting for it to close
        MOV DX,8000h
POLL:   IN AL,DX
        SHL AL,1
        JC POLL
```

Input Output Handshaking

- The I/O ports of a computer typically operate at different data rates
- A hard disk drive, for example, might require the computer to input data at 10Mbps → 100Mbps
- CD-ROM drives operate at 300-600 Kbps
- However when inputting keystrokes from the operator, the data rate may fall to only one or two characters per sec.
- If the processor is to operate efficiently, one needs to develop a strategy to control or synchronize the flow of data between the processor and the widely varying rates of its I/O devices
- This type of synchronization is achieved by implementing what is known as handshaking as part of the input/output interface
- Printers typically have buffers that can be filled by the computer at high speed
- Once full the computer must wait while the data in the buffer is printed
- Most printer manufacturers have settled on a standard set of data and control signals Centronics Parallel Printer Interface

Parallel Printer Interface





Pin	Assignment
1	Strobe
2	Data 0
3	Data 1
4	Data 2
5	Data 3
6	Data 4
7	Data 5
8	Data 6
9	Data 7
10	Ack
11	Busy
12	Paper Empty
13	Select
14	Auto Foxt
15	Error
16	Initialize
17	Slctin
18	Ground
19	Ground
20	Ground
21	Ground
22	Ground
23	Ground
24	Ground
25	Ground

```
Data:       Data0, Data1, ........., Data7

Control:    Strobe
            Auto Foxt
            Initialize
            Slctin

Status:     Ack
            Busy
            Paper Empty
            Select
            Error
```

ACK is used by printer to acknowledge receipt of data and can accept a new character.

BUSY high if printer is not ready to accept a new character

SELECT when printer is turned on

ERROR goes low when there are conditions such as paper jam, out of paper, offline

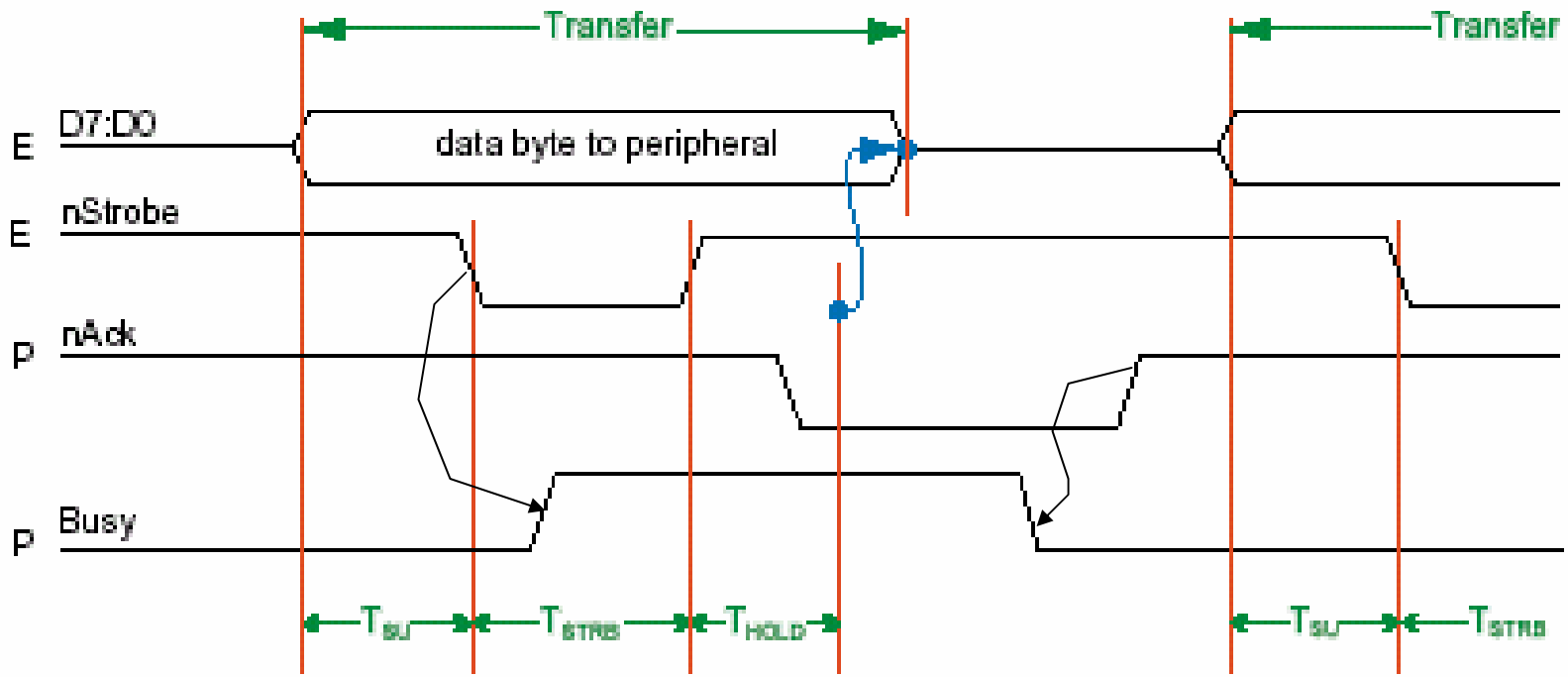
STROBE when PC presents a character

INITIALIZE Clear Printer Buffer and reset control

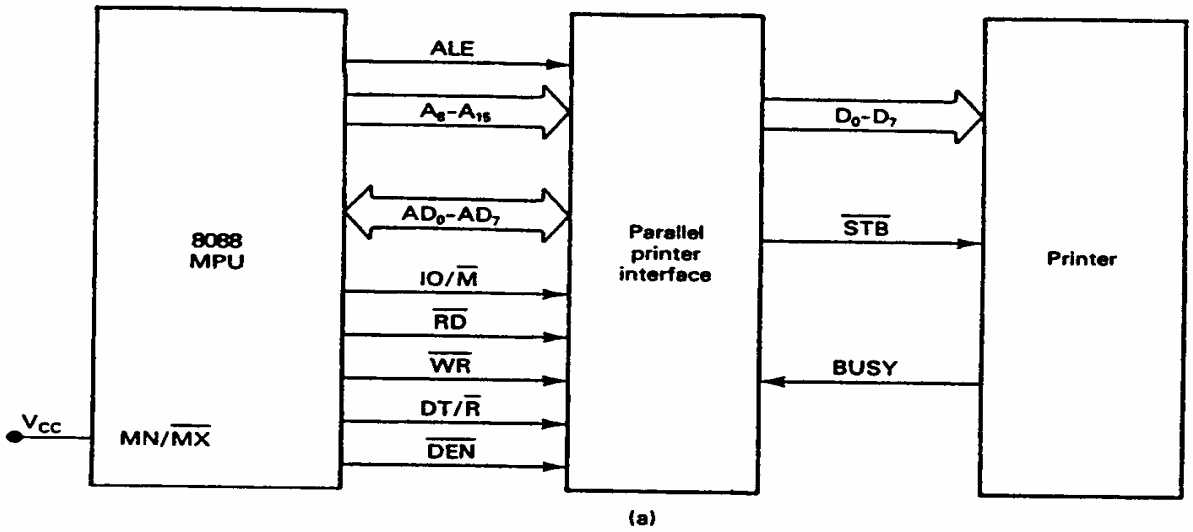
Operational Principle - Parallel Printer Port

- The computer checks the BUSY signal from the printer, if not BUSY then
- When the PC presents a character to the data pins of the printer, it activates the STROBE pin, telling it that there is a byte sitting at the data pins. Prior to asserting STROBE pin, the data must be at at the printer's data pins for at least 0.5 microsec. (data setup time)
- The STROBE must stay for 0.5 microsec

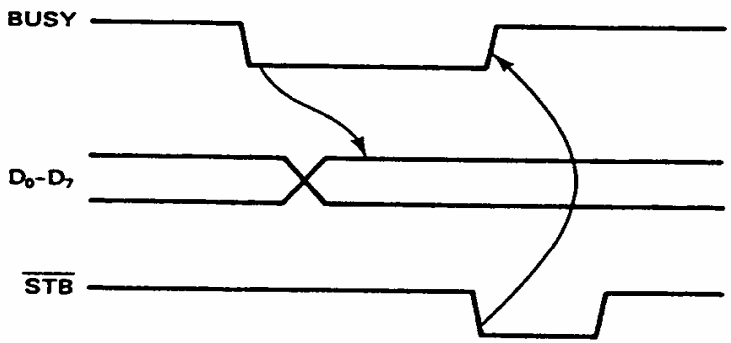
- The printer asserts BUSY pin indicating the computer to wait
- When the printer picks up the data, it sends back the ACK signal, keeps ACK low for 5 microsec.
- As the ACK signal is going high, the printer makes the BUSY pin low to indicate that it is ready to accept the next byte
- The CPU can use ACK or BUSY signals from the printer to initiate the process of sending another byte


Transfer
Transfer
E
D7:D0
data byte to peripheral
E
nStrobe
P
nAck
P
Busy
T_{SU}
T_{STRB}
T_{HOLD}
T_{SU}
T_{STRB}

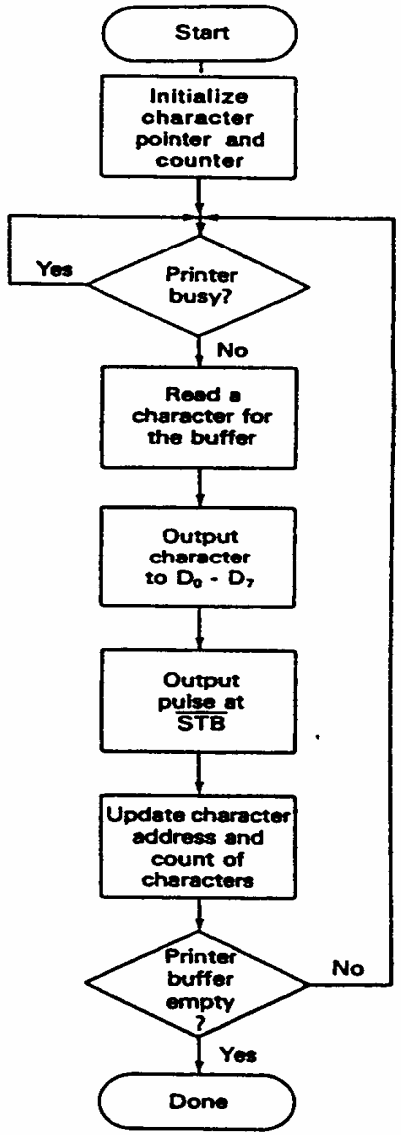
Handshaking



(a)

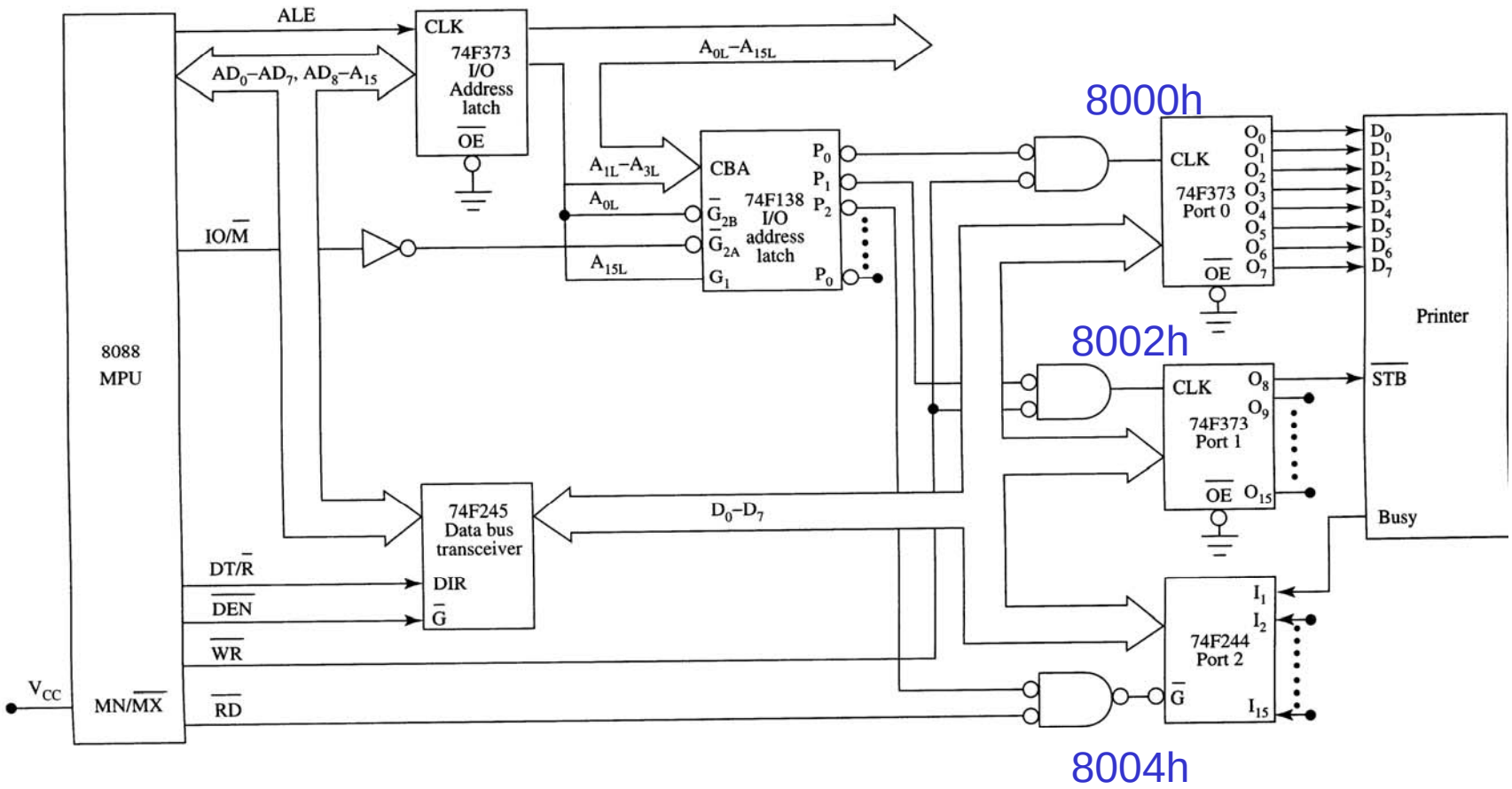


(b)



(c)

Printer Interface Circuit



Example

- Write a program that implements the flowchart. Character data is held in memory starting at address PRNT_BUFF, the number of characters held in the buffer is identified by the count address CHAR_COUNT.

```
                MOV CL, CHAR_COUNT
                MOV SI,  OFFSET PRNT_BUFF

POLL_BUSY:      MOV DX,8004h
                IN AL,DX
                AND AL,01h                      BUSY input checked
                JNZ POLL_BUSY

                MOV AL, [SI]
                MOV DX,8000h                    Character is output
                OUT DX,AL

                MOV AL, 00h             ;STB = 0
                MOV DX,8002h
                OUT DX,AL                       So as the strobe
                MOV BX,0Fh              ; delay for STB = 0
                STROBE: DEC BX
                JNZ STROBE
                MOV AL,01h
                OUT DX,AL               ; STB bar = 1

                 INC SI
                 DEC CL
                JNZ POLL_BUSY
```

IBM PC Printer Interfacing

- Base I/O port address of each printer is written into the BIOS data area 0040:0008 to 0040:000Fh
- 0040:0008 and 0040:0009 LPT1 (e.g., 0378h)
  - 0378h: data port
  - 0379h: status port read only
  - 037Ah: control port
- 0040:000A and 0040:000B LPT2
- BIOS INT 17h provides three services: printing a character, initializing the printer port and getting the printer status port
- AH = 0; print a character in AL and have the LPT number in DX (0 for LPT1, 1 for LPT2) also returns status
- AH = 01; initialize the printer port by setting the printer to the top of the page
- AH = 2; get the printer status after calling AH = status as:
  - bit 7: 1 ready 0 busy
  - bit 3: 1 I/O error
  - bit 5: 1 out of paper

Printer Time Out

Timeout occurs due to: printer offline / not able to print for any reason although it is properly connected

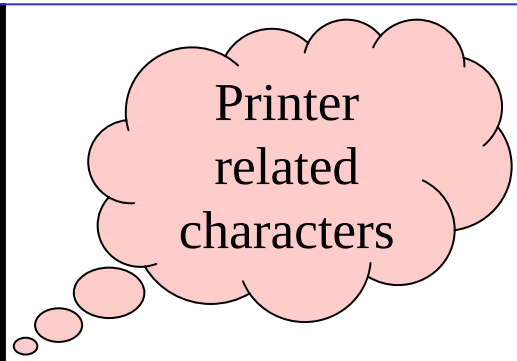
BIOS repeatedly tries for a period of 20 seconds to see if the printer is ready to accept data.

It is possible to alter the entered by BIOS at boot time.

0040:0078 → 0040:007B holds the wait times.

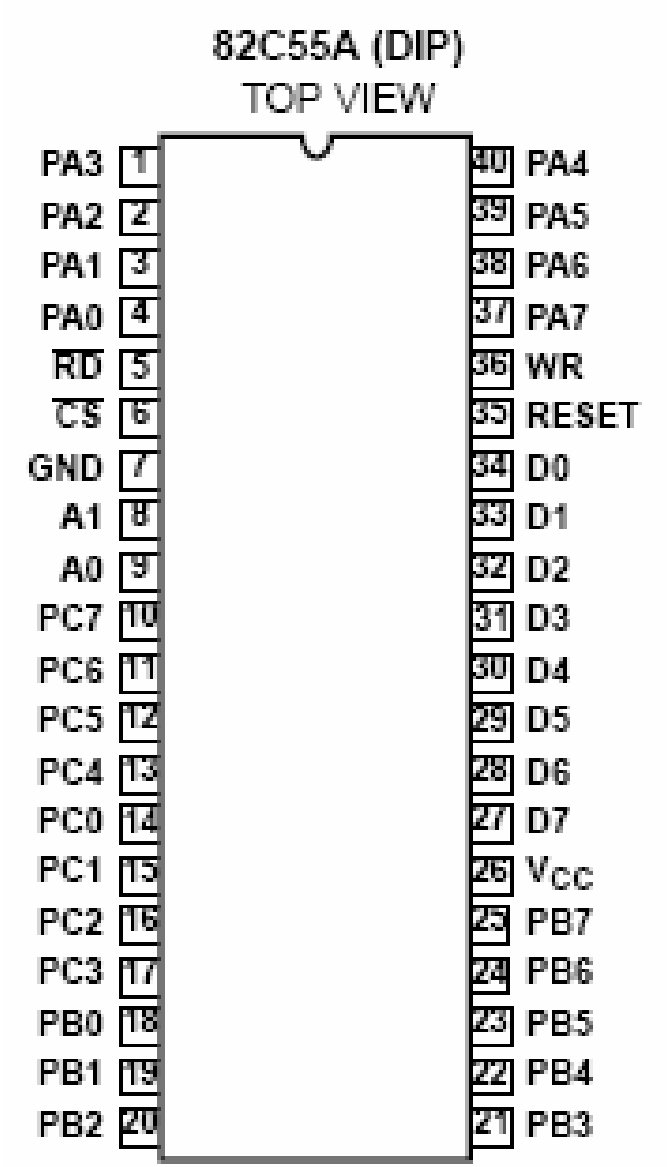
```
MOV AH,0 ; print character option
MOV DX,0 ; select LPT1
MOV AL,41h      ; ASCII code for letter A
INT 17h
```

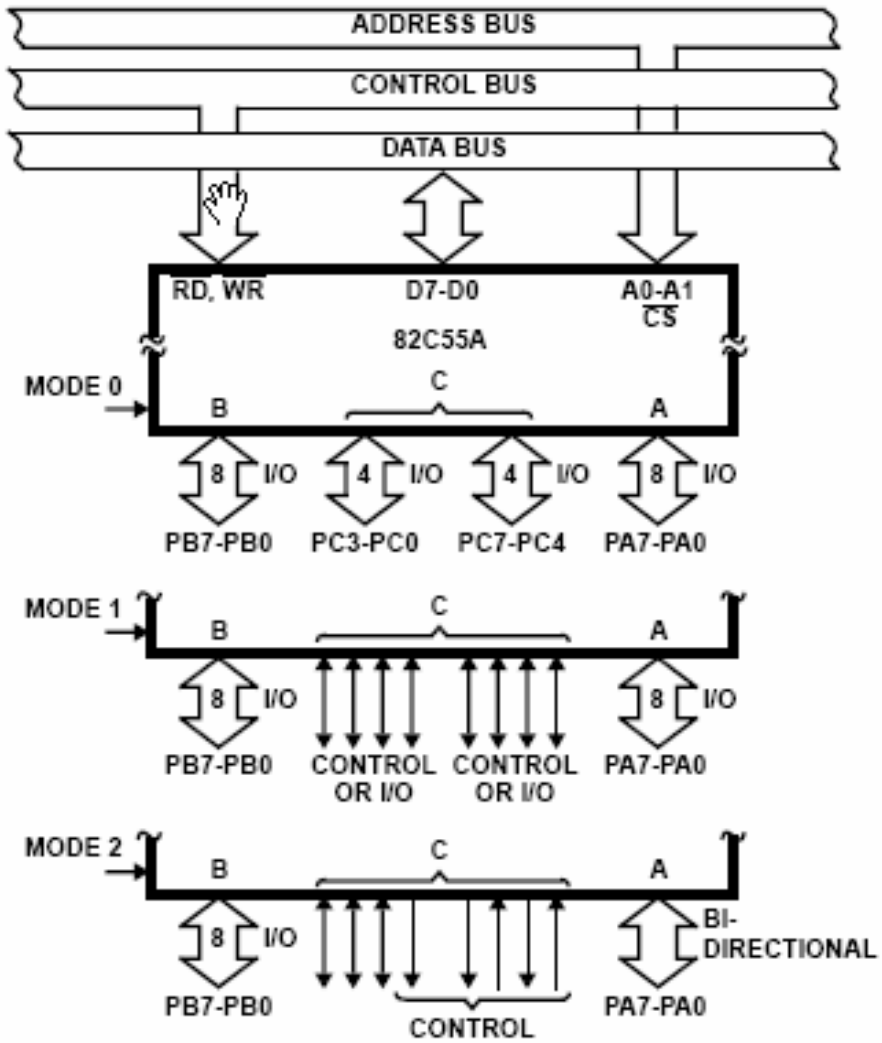
ASCII Table	Hex Code	Function
BS	08	Backspace
HT	09	Horizontal Tab
LF	0A	Line Feed
VT	0B	Vertical Tab
FF	0C	Form Feed
CR	0D	Carriage Return



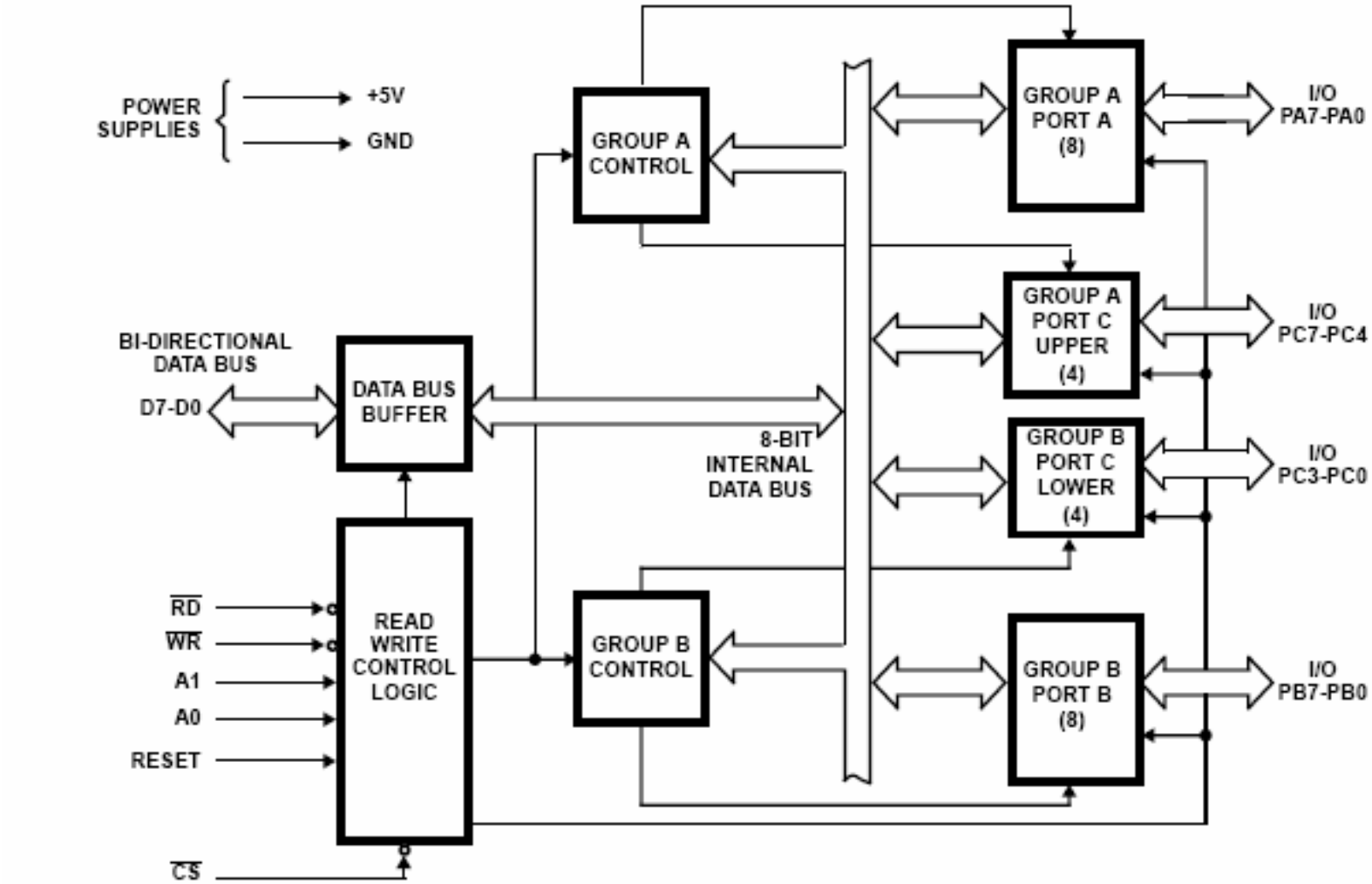
The 8255 Programmable Peripheral Interface

- Intel has developed several peripheral controller chips designed to support the 80x86 processor family. The intent is to provide a complete I/O interface in one chip.
- 8255 PPI provides three 8 bit input ports in one 40 pin package making it more economical than 74LS373 and 74LS244
- The chip interfaces directly to the data bus of the processor, allowing its functions to be programmed; that is in one application a port may appear as an output, but in another, by reprogramming it as an input. This is in contrast with the 74LS373 and 74LS244 which are hardwired and fixed
- Other peripheral controller chips include the 8259 Programmable Interrupt Controller (PIC), the 8253/54 Programmable Interval Timer (PIT) and the 8237 DMA controller

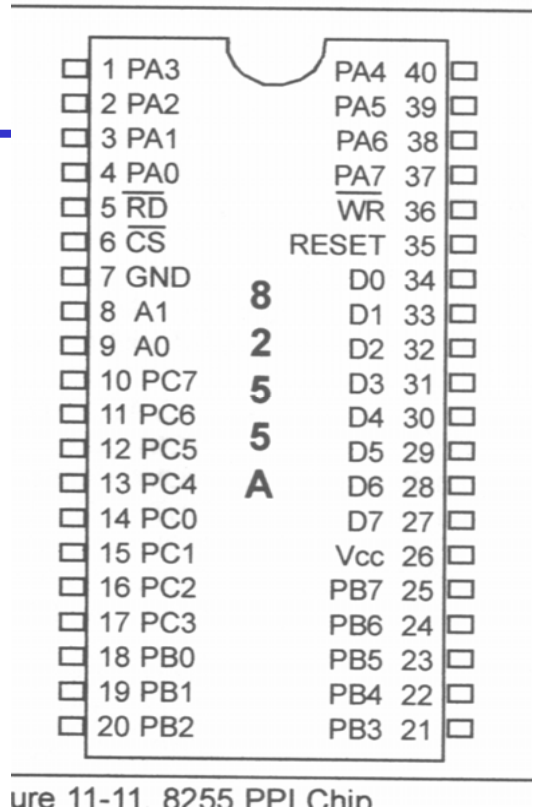

82C55A (DIP)
TOP VIEW
PA3 1
PA2 2
PA1 3
PA0 4
RD 5
CS 6
GND 7
A1 8
A0 9
PC7 10
PC6 11
PC5 12
PC4 13
PC0 14
PC1 15
PC2 16
PC3 17
PB0 18
PB1 19
PB2 20
40 PA4
39 PA5
38 PA6
37 PA7
36 WR
35 RESET
34 D0
33 D1
32 D2
31 D3
30 D4
29 D5
28 D6
27 D7
26 VCC
25 PB7
24 PB6
23 PB5
22 PB4
21 PB3


ADDRESS BUS
CONTROL BUS
DATA BUS
RD, WR
D7-D0
A0-A1
CS
82C55A
MODE 0
B
C
A
8 I/O
4 I/O
4 I/O
8 I/O
PB7-PB0
PC3-PC0
PC7-PC4
PA7-PA0
MODE 1
B
C
A
8 I/O
8 I/O
PB7-PB0
CONTROL OR I/O
CONTROL OR I/O
PA7-PA0
MODE 2
B
C
A
8 I/O
BI-DIRECTIONAL
PB7-PB0
CONTROL
PA7-PA0

8255A internal



8255 Pins

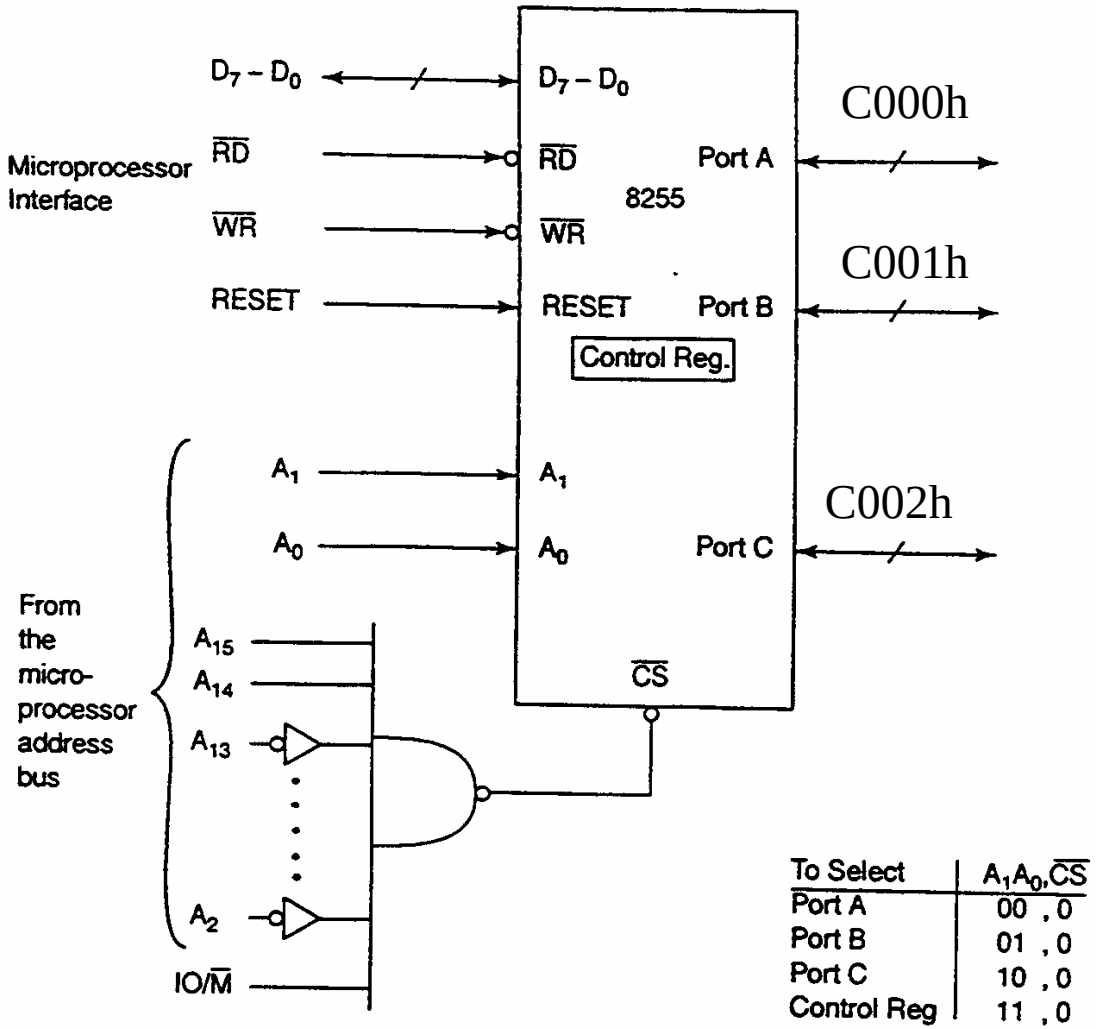


ure 11-11 8255 PPI Chip

- PA0 - PA7: input, output, or bi-directional port
- PB0 - PB7: input or output
- PC0 - PC7: This 8 bit port can be all input or output. It can also be split into two parts, CU (PC4 - PC7) and CL (PC0 - PC3). Each can be used for input and output.
- $\overline{RD}$ or $\overline{WR}$
  - $\overline{IOR}$ and $\overline{IOW}$ of the system are connected
- RESET
- A0, A1, and $\overline{CS}$
  - $\overline{CS}$ selects the entire chip whereas A0 and A1 select the specific port (A, B, or C)

CSBAR	A1	A0	SELECTS:
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control Register
1	x	x	8255 not selected

Addressing an 8255



To Select	$A_1A_0,\overline{CS}$
Port A	00 , 0
Port B	01 , 0
Port C	10 , 0
Control Reg	11 , 0

8255 Control Word Format

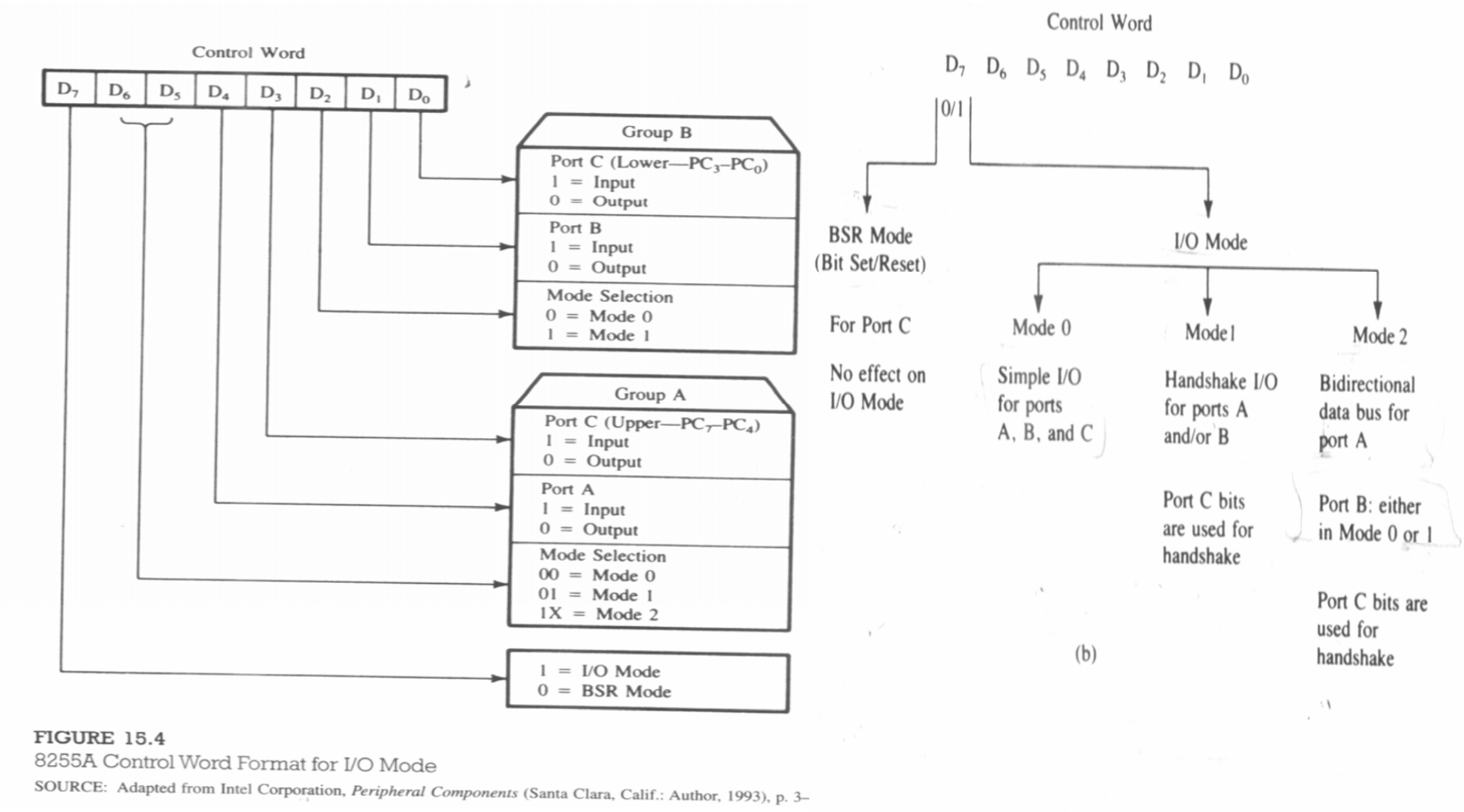


FIGURE 15.4
8255A Control Word Format for I/O Mode

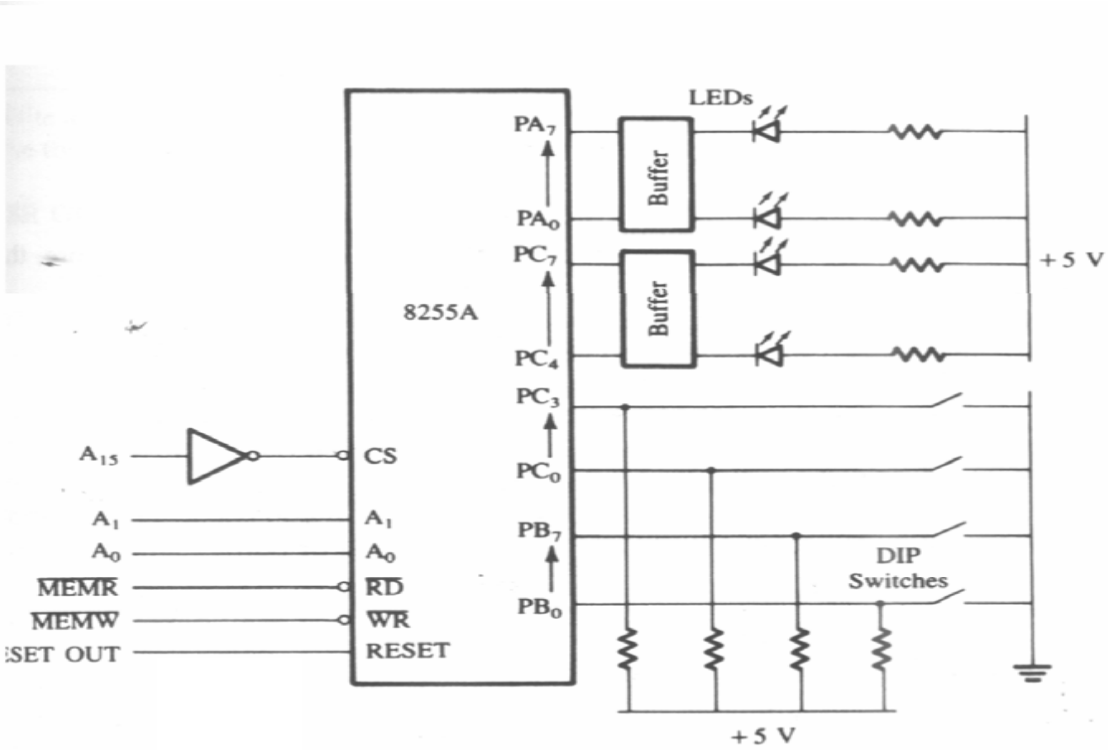
SOURCE: Adapted from Intel Corporation, *Peripheral Components* (Santa Clara, Calif.: Author, 1993), p. 3–

Mode 0 - Simple input/output

- Simple I/O mode: any of the ports A, B, CL, and CU can be programmed as input or output.
- Example: Configure port A as input, B as output, and all the bits of port C as output assuming a base address of 50h
- Control word should be 1001 0000b = 90h

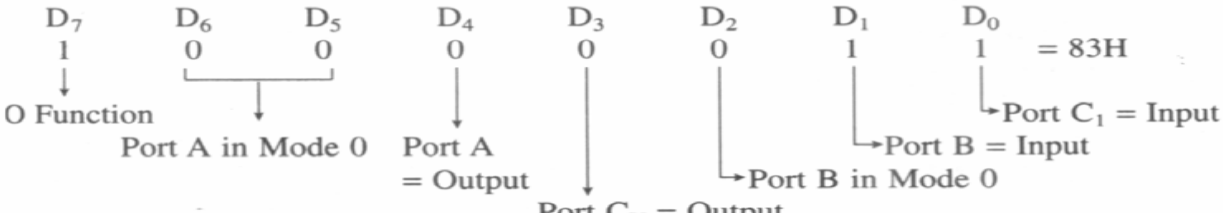
```
PORTA  EQU 50h
PORTB  EQU 51h
PORTC  EQU 52h
CNTREG EQU 53h
MOV AL, 90h
OUT CNTREG,AL
IN AL, PORTA
OUT PORTB, AL
OUT PORTC, AL
```

Mod 0 Simple I/O



IGURE 15.5
terfacing 8255A I/O Ports in Mode 0

Control Word



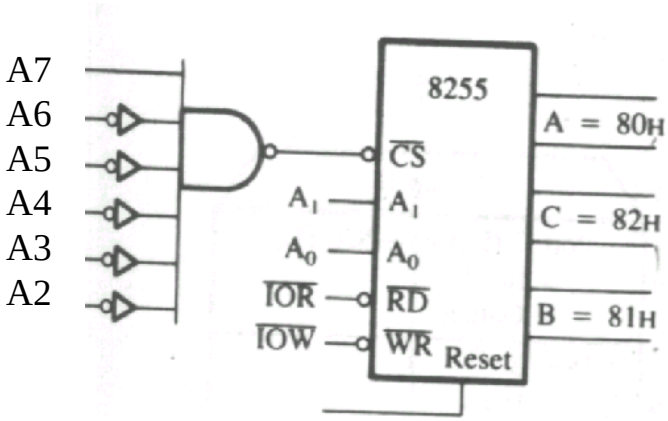
Initialize this device with the appropriate Control Word. Read from PORT C_L and display at PORT A.

PORT A 8000h
PORT B 8001H
PORT C 8002H
CONTROL 8003H

Be careful Memory I/O!

```
MOV AL,83H
MOV BX,8003H
MOV [BX],AL
MOV BX,8002H
MOV AL,[BX]
AND AL,0FH
DEC BX
DEC BX
MOV [BX],AL
```

BSR Mode



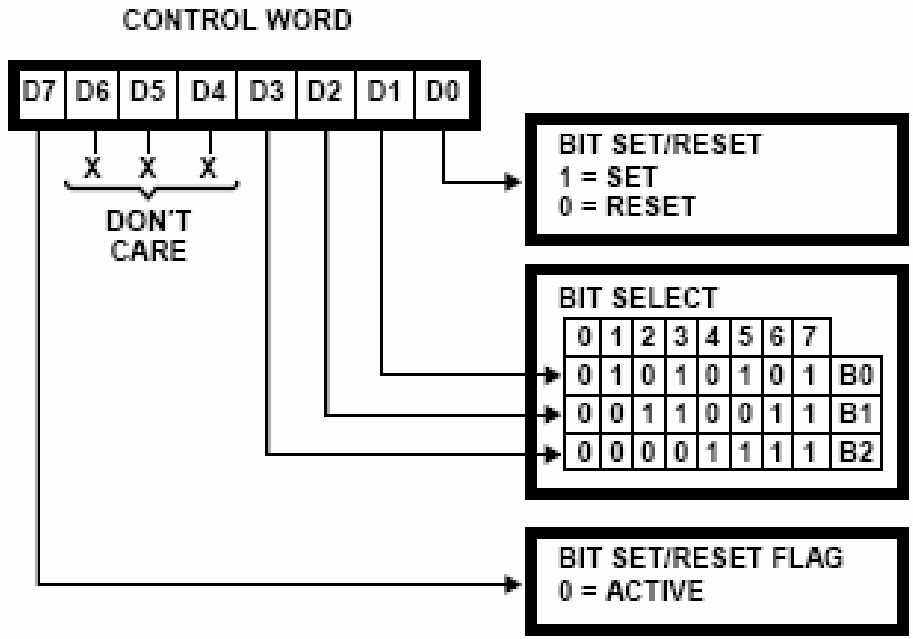


FIGURE 5. BIT SET/RESET FORMAT

- Concerned with the eight bits of port C only which can be set or reset by writing appropriate control word with D7=1
- It does not alter the previously transmitted control word with D7=0

- Ex: Write a BSR word subroutine to set PC7 and PC3

To Set PC7 → OFH ; To set PC3 → 07H

```
MOV AL,0FH
OUT 83H,AL
MOV AL,07H
OUT 83h,AL
```

PC Interface Trainer Decoding Circuitry

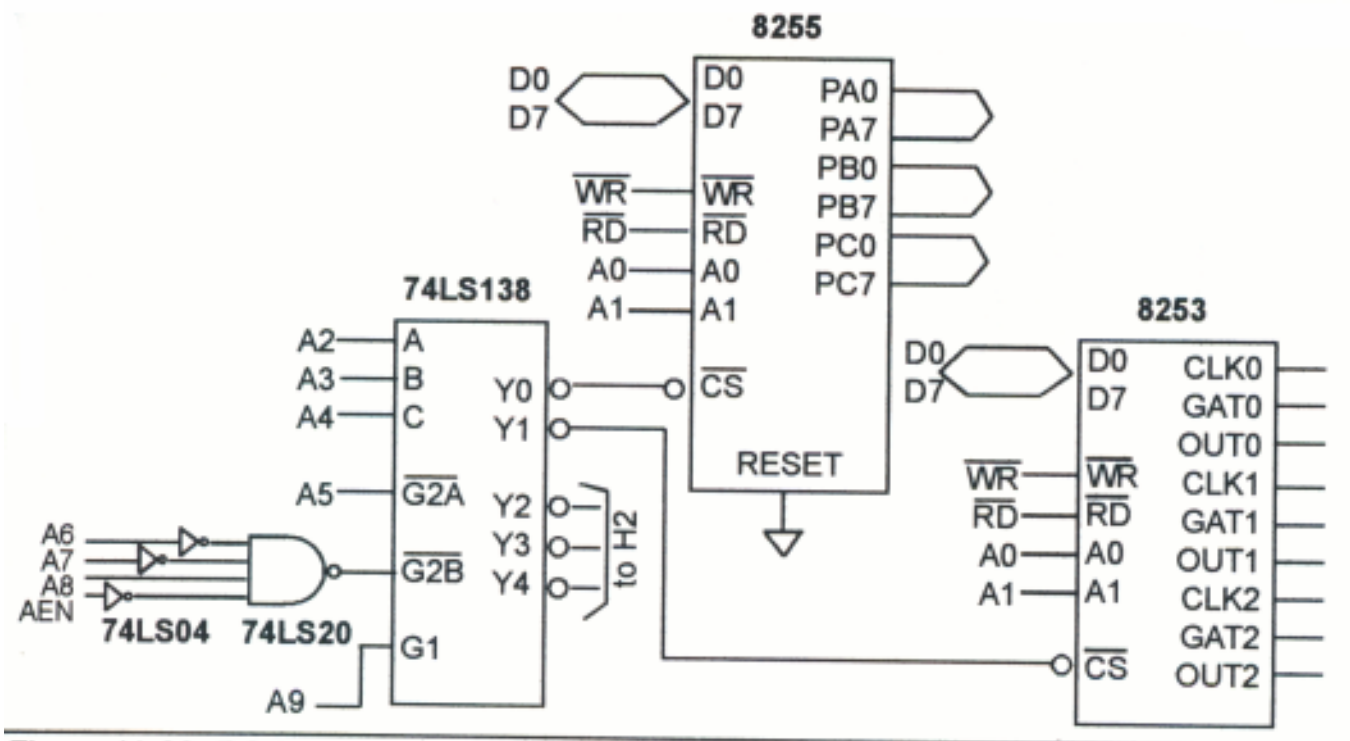
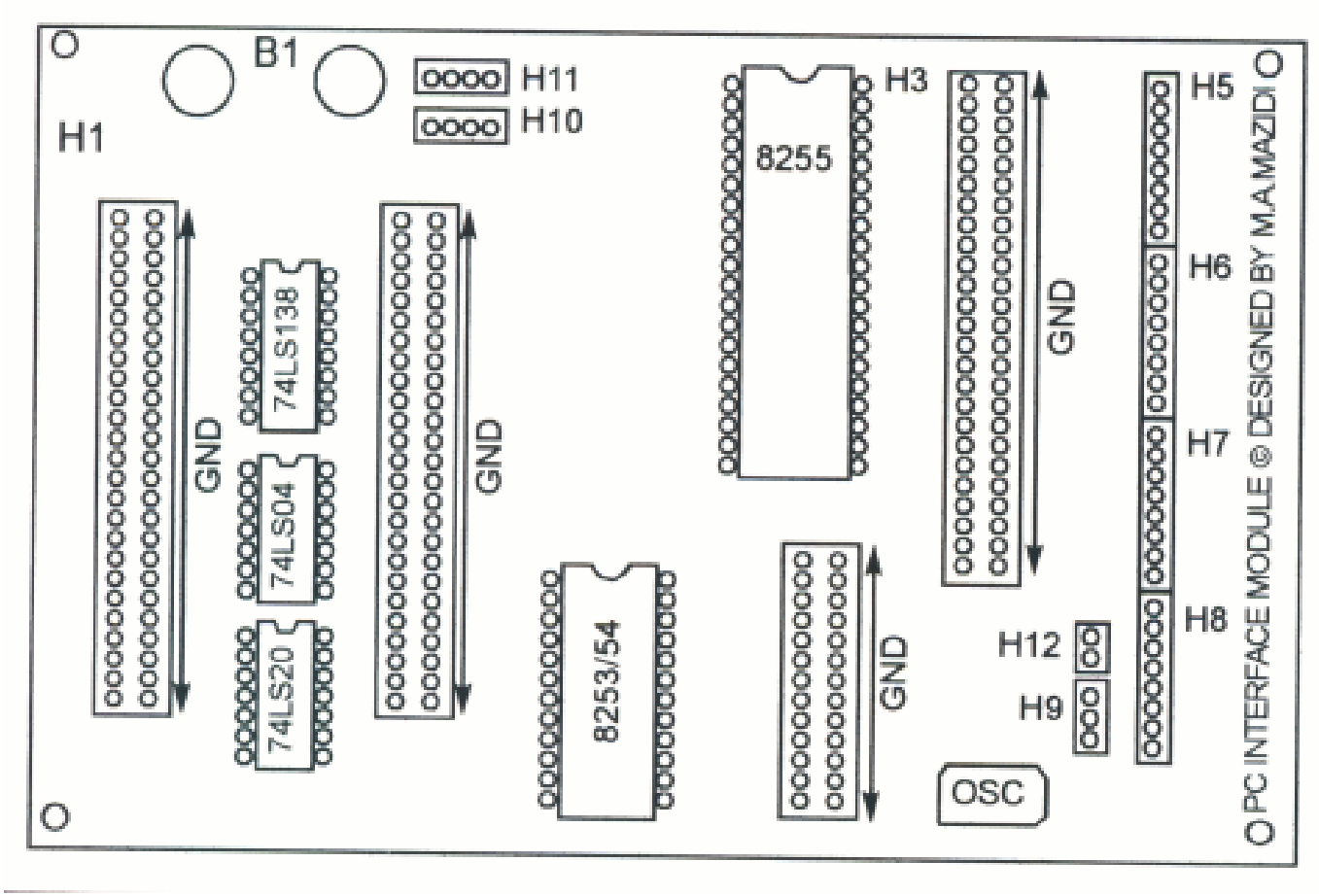


Figure 11-20. PC Interface Trainer Decoding Circuitry

PC Interface Trainer Module



Example 11.4 of Textbook

- Find the control word
  - PA = out
  - PB = in
  - PC0 – PC3 = in
  - PC4 – PC7 = out
- Program the 8255 to get data from port B and send it to port A; in addition data from PCL is sent out to the PCU
- Use port addresses 300h – 303h for the 8255 chip

Control Word:
The control word should be
1000 0011b = 83h

Program

```
B8255       EQU   300h
CNTL        EQU   83h

MOV DX,B8255+3
MOV AL,CNTL
OUT DX,AL
MOV DX,B8255+1
IN AL,DX
MOV DX,B8255
OUT DX,AL
MOV DX,B8255+2
IN AL,DX
AND AL,0Fh
MOV CL,4
ROL AL,CL
OUT DX,AL
```

Example 11-6 Textbook

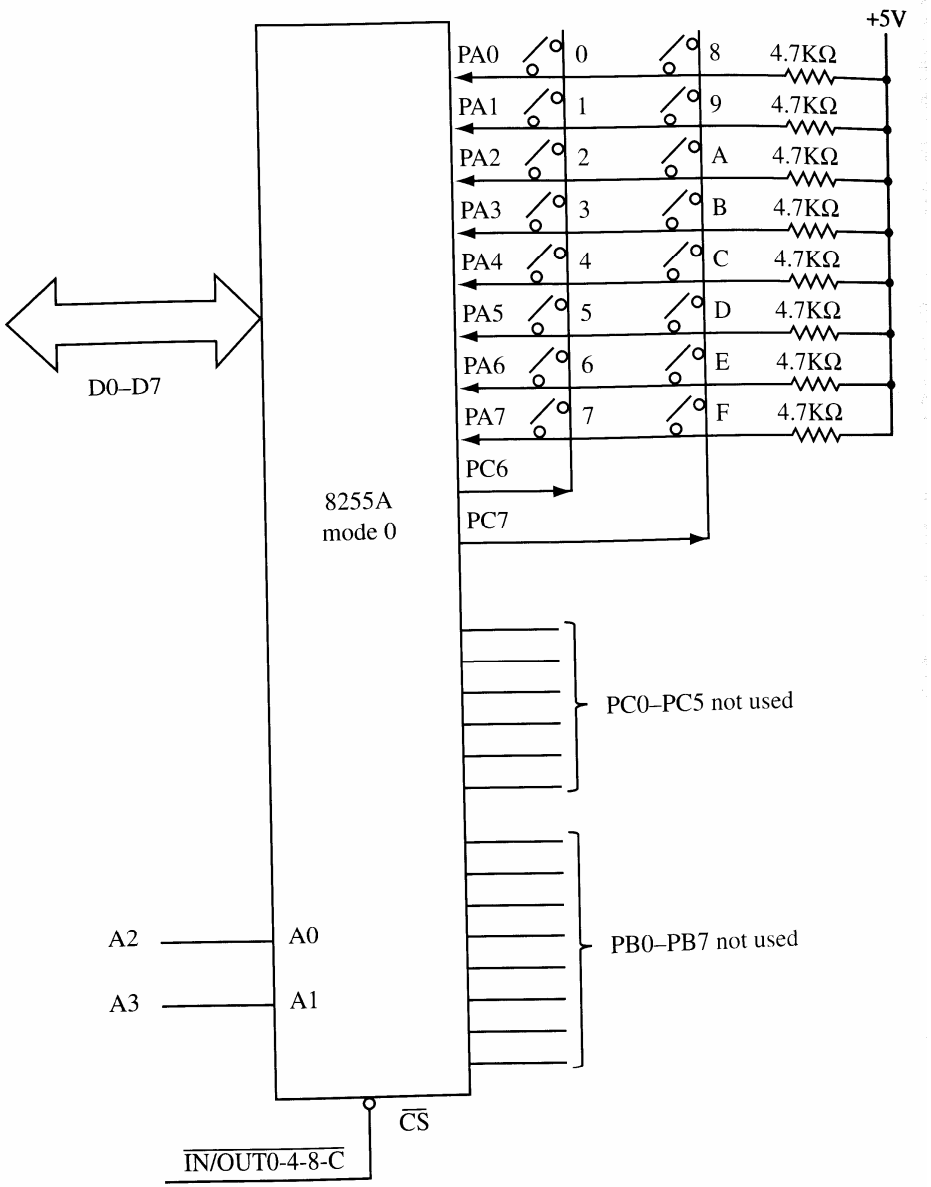
- Assume 8255 has a base address 300h
- Write a program to toggle all bits of port A continuously with a ¼ sec. Delay
- Use int 16h to exit if there is a key press

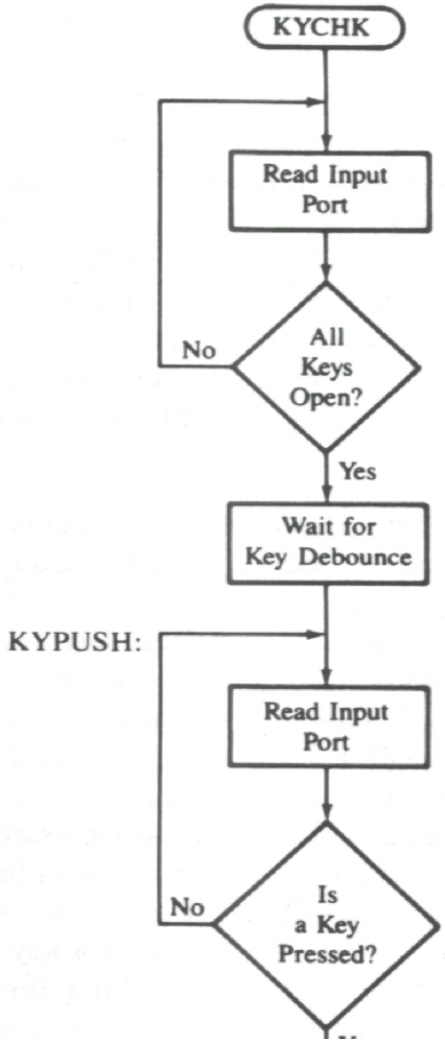
```
                MOV   DX,303h
                MOV AL,80h
                OUT DX,AL
AGAIN:          MOV DX,300h
                 MOV AL,55h
                 OUT DX,AL
                 CALL QSDELAY
                 MOV AL,0AAh
                 OUT DX,AL
```

Example Contd

```
                    CALL  QSDELAY
                    MOV AH,01
                    INT 16h
                    JZ AGAIN
                    MOV AH,4Ch
                    INT 21h
; to create  a processor independent delay IBM made PB4 of port 61h to toggle every
    15.085 microsec. (for 286 and higher processors)
QSDELAY             PROC  NEAR
                    MOV CX,16572  ;16572*15.085 microsec = ¼ s
                    PUSH  AX
            W1:     IN AL,61h
                    AND AL,00010000b
                    CMP AL,AH
                    JE   W1
                    MOV AH,AL
                     LOOP W1
                    POP AX
                    RET
 QSDELAY            ENDP
```

Mode 0 Design Example - Interfacing a matrix keyboard

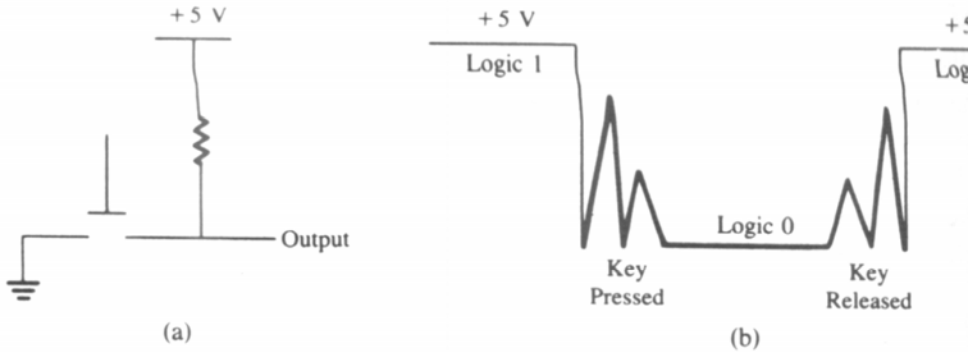


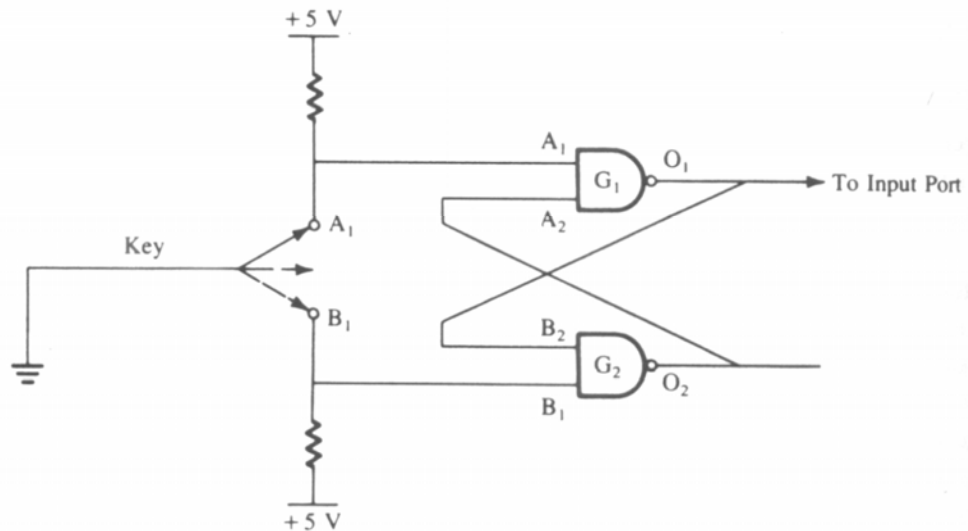


Key Debounce

✓When a mechanical push button key is pressed or released the metal contacts of the key momentarily bounce before giving a steady-state reading. Therefore it is necessary that the bouncing of the key not be read as an input.

✓Key debounce can be eliminated using either software or hardware.





Key debounce technique in hardware

```
                        ;Function:      Scan the keyboard shown in Fig. 8.14
                        ;               and return with the encoded key
                        ;               value in register AL.
                        ;Inputs:        none
                        ;Outputs:       hex key value in AL.
                        ;Calls:         10 ms delay procedure for debouncing
                        ;Destroys:      AX and flags

                        ;*******
                        ;  Set up segment to store key values
                        ;*******

0000                    KEY_CODE        SEGMENT BYTE
0000  00 01 02 03 04    COL1            DB      0,1,2,3,4
0005  05 06 07                         DB      5,6,7
0008  08 09 0A 0B 0C    COL2            DB      8,9,0AH,0BH,0CH
000D  0D 0E 0F                         DB      0DH,0EH,0FH
0010                    KEY_CODE        ENDS

0000                    CODE            SEGMENT BYTE
                                        ASSUME  CS:CODE,DS:KEY_CODE

                        ;*******
                        ;Program equates
                        ;*******

= 00F0                  PORT_A          EQU     00H             ;PPI port A address (see Fig. 8.12)
= 00F2                  PORT_C          EQU     08H             ;PPI port C address
= 00BF                  COL_1_LOW       EQU     10111111B       ;PC6 low
= 007F                  COL_2_LOW       EQU     01111111B       ;PC7 low
= 003F                  BOTH_COL_LOW    EQU     00111111B       ;PC6 and PC7 low
= 00FF                  KEY_UP          EQU     0FFH            ;Input 0FFH when no keys are down
= 16FA                  T1              EQU     8B82H           ;~ 10 ms time delay assuming 25 MHz 80
```

```
;*******
;   10 ms time delay for debouncing
;*******

DELAY           PROC    NEAR
                MOV     CX,T1
COUNT:          LOOP    COUNT
                RET
DELAY           ENDP


;*******
;  Main program begins here
;*******

KEYBOARD        PROC    NEAR
                PUSH    DS                  ;Save registers about to be used
                PUSH    CX
                PUSH    SI
                MOV     AX,KEY_CODE         ;Point DS to the key codes
                MOV     DS,AX

;Wait for previous key to be released

                MOV     AL,BOTH_COL_LOW     ;Scan both columns
                OUT     PORT_C,AL           ;Column lines on PC6 and PC7
POLL1:          IN      AL,PORT_A           ;Read keyboard
                CMP     AL,KEY_UP           ;All keys up?
                JNE     POLL1               ;No - so wait
                CALL    DELAY               ;Yes - wait for bounce on release

;Wait for a new key to be pressed

POLL2:          IN      AL,PORT_A           ;Read keyboard
                CMP     AL,KEY_UP           ;Any keys down?
                JE      POLL2               ;No - so wait
                CALL    DELAY               ;Yes - wait for bounce
```

```
                        ;See if the key is in column 1

0024  B0 BF                          MOV     AL,COL_1_LOW        ;Test for column 1
0026  E6 08                          OUT     PORT_C,AL           ;PC6 low
0028  E4 00                          IN      AL,PORT_A           ;Read column 1 keys
002A  3C FF                          CMP     AL,KEY_UP           ;Any key down?
002C  74 07                          JE      CHECK_COL_2         ;No - check for column 2
002E  8D 36 0000 R                   LEA     SI,COL1             ;Yes - point SI at the key values 0-7
0032  EB 0F 90                       JMP     LOOKUP              ;Now lookup code

                        If not column 1 then column 2

0035  B0 7F             CHECK_COL_2: MOV     AL,COL_2_LOW        ;Test for column 2
0037  E6 08                          OUT     PORT_C,AL           ;PC7 low
0039  E4 00                          IN      AL,PORT_A           ;Read column 2 keys
003B  3C FF                          CMP     AL,KEY_UP           ;Any key down?
003D  74 DC                          JE      POLL2               ;No - false input so repeat
003F  8D 36 0008 R                   LEA     SI,COL2             ;Yes - point SI at key values 8-F

                        ;Now lookup the key's value and store in AL

0043  D0 D8             LOOKUP:      RCR     AL,1                ;Rotate keyboard input code right
0045  73 03                          JNC     MATCH               ;If 0 key is found - so retrieve it
0047  46                             INC     SI                  ;No - advance pointer to next value
0048  EB F9                          JMP     LOOKUP              ;Repeat the loop

004A  8A 04             MATCH:       MOV     AL,[SI]             ;Get the key code
004C  5E                             POP     SI                  ;Restore all registers
004D  59                             POP     CX                  ;(except AX and flags)
004E  1F                             POP     DS
004F  C3                             RET
0050                    KEYBOARD     ENDP
0050                    CODE         ENDS
                                     END     KEYBOARD
```

8 Digit LED

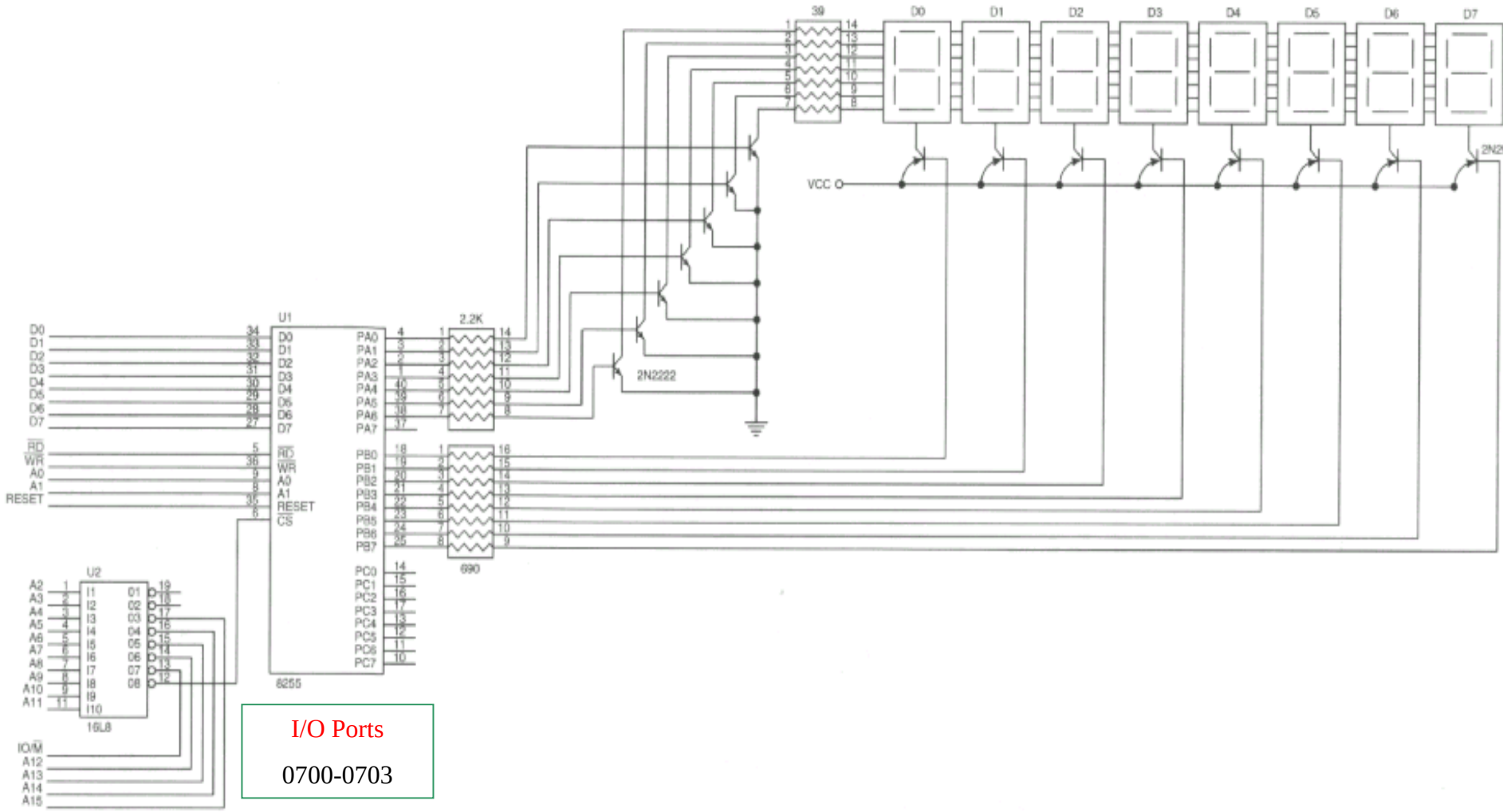
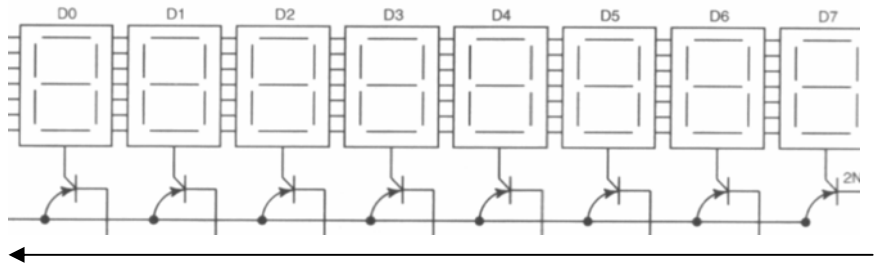


FIGURE 11–20 An 8-digit LED display interfaced to the 8088 microprocessor through an 82C55 PIA.

8 Digit LED

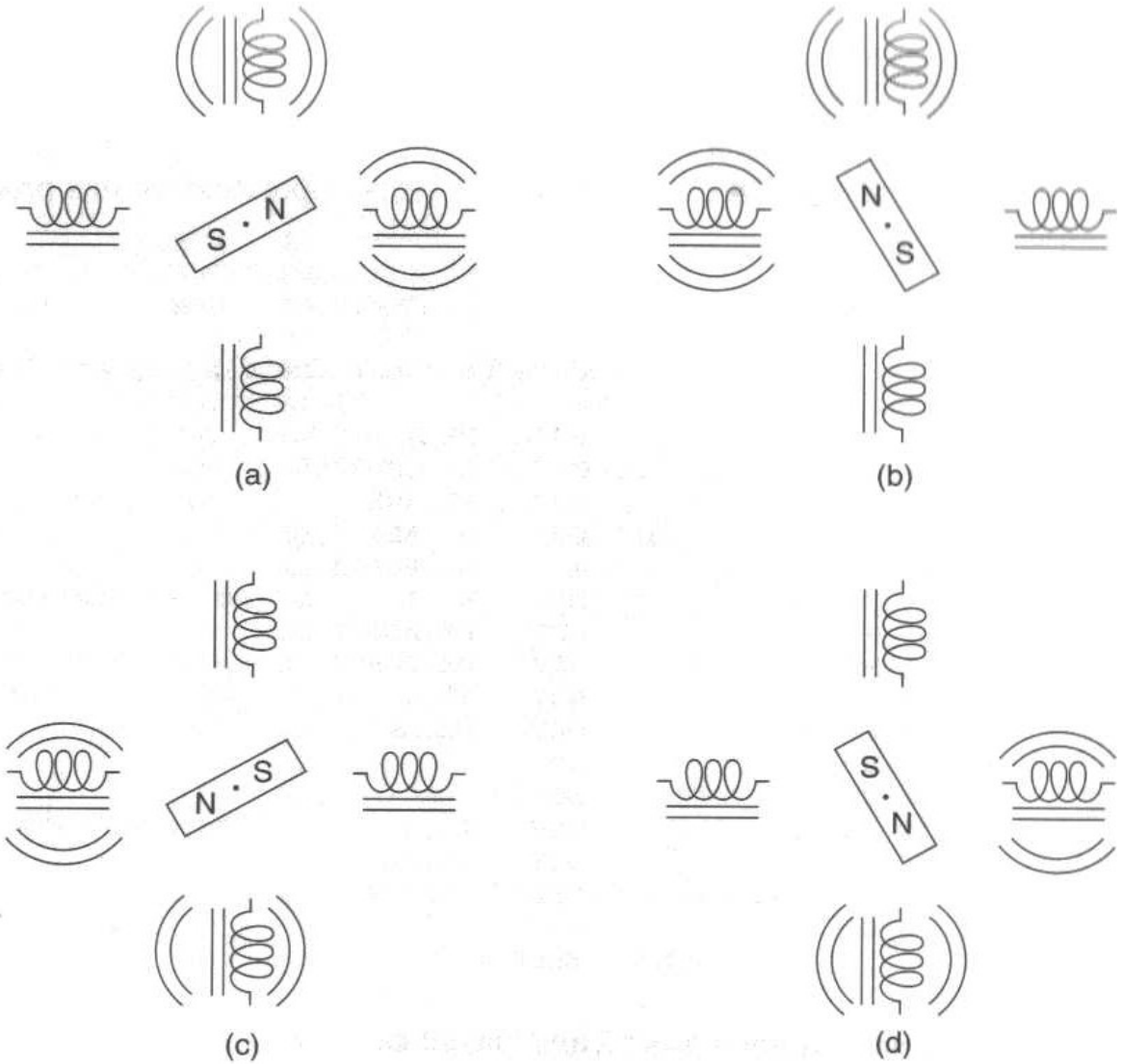
```
; INIT 8255
MOV AL,10000000B
MOV DX,703H
OUT DX,AL

;SETUP REGISTERS TO DISPLAY
MOV BX,8
MOV AH,7FH
MOV SI,OFFSET MEM-1
MOV DX.701H

; DISPLAY 8 DIGITS

DISP1: MOV AL,AH  ;select digit
OUT DX,AL
DEC DX            ;address PORT A
MOV AL,[BX+SI]
OUT DX,AL
CALL DELAY   ;wait 1 ms
ROR AH,1
INC DX            ;address PORT B
DEC BX            ;adjust count
JNZ DISP1

RET
```



1 1 1 1 1 1 1 0

Motor Application

FIGURE 11–22 The stepper motor showing full-step operation. (a) 45° (b) 135° (c) 225° (d) 315°.



Motor Application

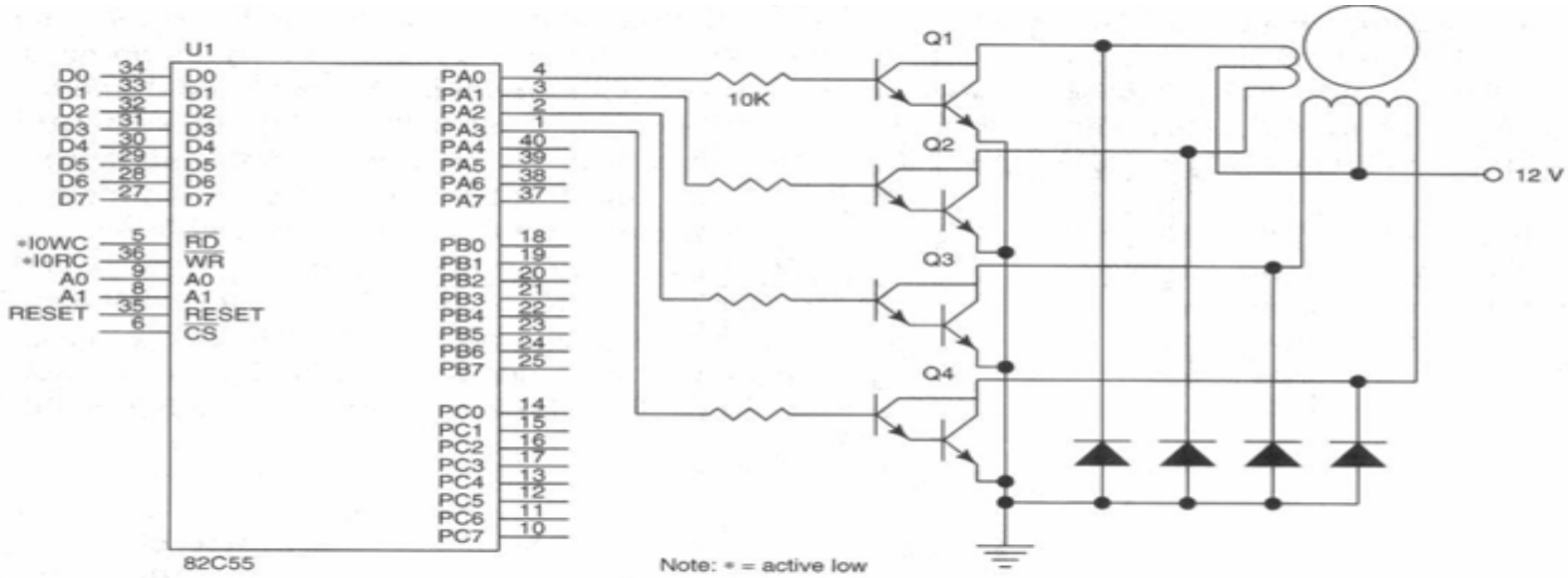


FIGURE 11–23 A stepper motor interfaced to the 82C55. This illustration does not show the decoder.

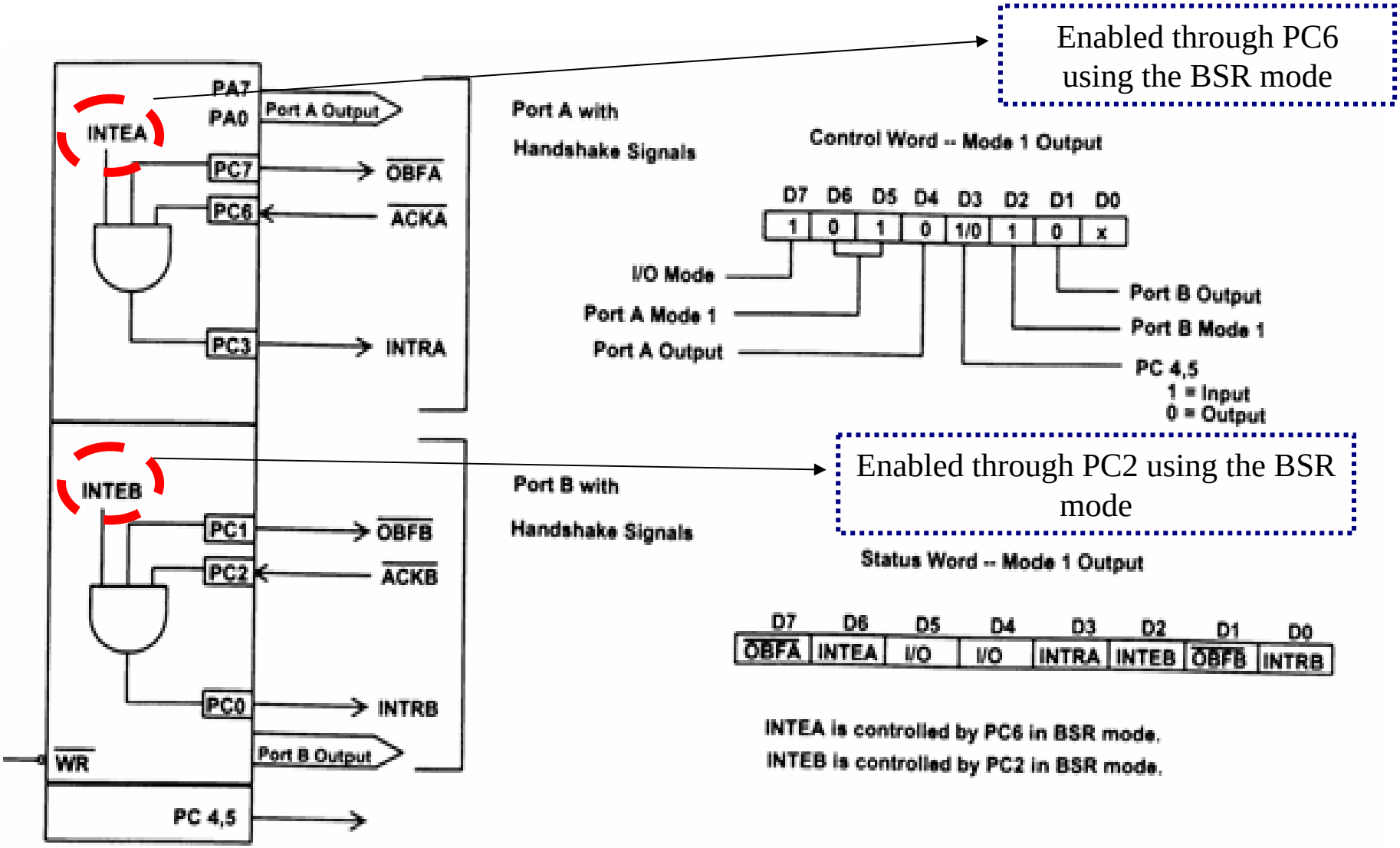
Rotate the motor clockwise if A15 is 1
else rotate counterclockwise

```
STEP:     PROC NEAR
          MOV AL, POS ;current position
          CMP CX, 8000H ;rotation amount
          JA RH
          CMP CX,0
          JE STEP_OUT
STEP1:    ROL AL,1
          OUT PORT, AL
          CALL DELAY
          LOOP STEP1
          JMP STEP_OUT
RH:       AND CX, 7FFFH; Clr 15
RH1:      ROR AL,1
          OUT PORT,AL
          CALL DELAY
          LOOP RH1
STEP_OUT: MOV POS,AL
          RET
```

Mode 1: I/O with Handshaking Capability

- Handshaking refers to the process of communicating back and forth between two intelligent devices
- Example: Process of communicating with a printer
  - a byte of data is presented to the data bus of the printer
  - the printer is informed of the presence of a byte of data to be printed by activating its strobe signal
  - whenever the printer receives the data it informs the sender by activating an output signal called ACK
  - the ACK signal initiates the process of providing another byte of data to the printer
- 8255 in mode 1 is equipped with resources to handle handshaking signals

Mode 1 Strobed Output



Mode 1 Strobed Output

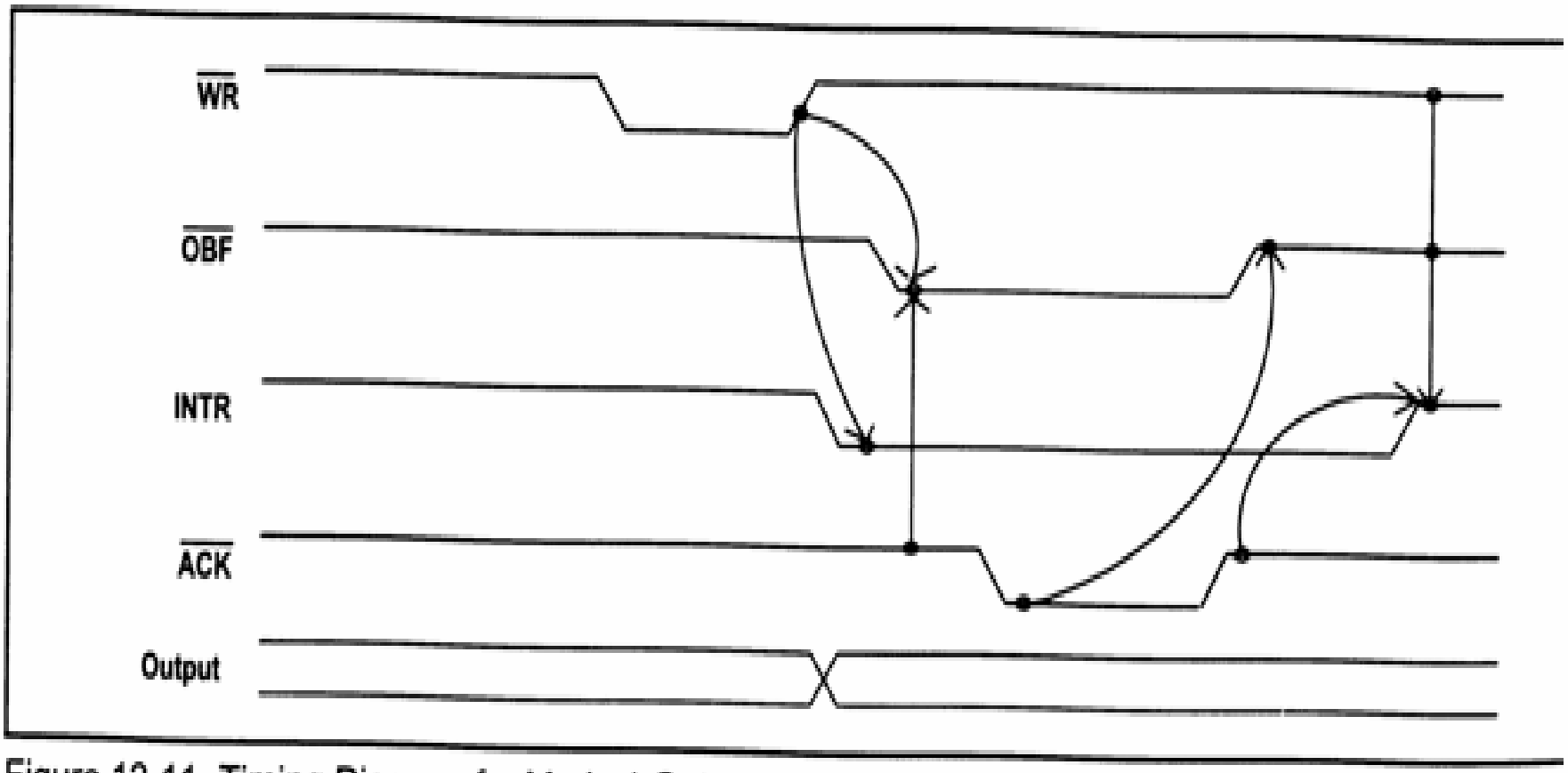


Figure 12-11. Timing Diagram for Mode 1 Output

Mode 1 Strobed Output Signals

- $\overline{\text{OBFa}}$ (output buffer full for port A)
  - indicates that the CPU has written a byte of data into port A
  - must be connected to the STROBE of the receiving equipment
  - Goes back high again after ACK'ed by the peripheral.
- ACKa (acknowledge for port A)
  - through ACK, 8255 knows that data at port A has been picked up by the receiving device
  - 8255 then makes $\overline{\text{OBFa}}$ high to indicate that the data is old now. $\overline{\text{OBFa}}$ will not go low until the CPU writes a new byte of data to port A.
- INTRa (interrupt request for port A)
  - it is the rising edge of ACK that activates INTRa by making it high. INTRa is used to get the attention of the microprocessor.
  - it is important that INTRa is high only if INTEa, OBFa, ACKa are all high
  - it is reset to zero when the CPU writes a byte to port A
- The 8255 enables the monitoring the status signals INTR, OBF, and INTE for both ports A and B. This is done by **reading port C** into the accumulator and testing the bits. This feature allows the implementation of polling

Interrupts vs Polling

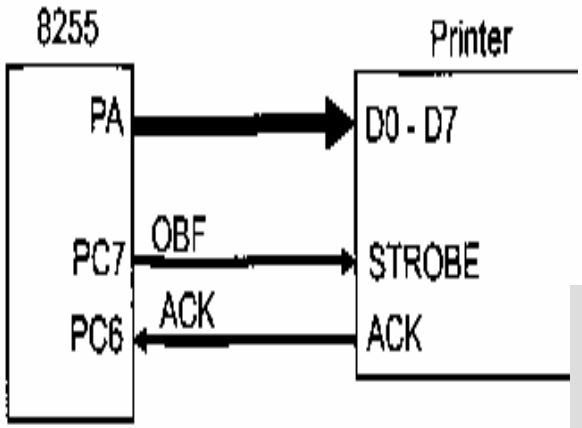
CPU services devices in two ways:

Interrupts and Polling

- In the interrupt method, whenever the device needs the service of the CPU, the device informs the CPU by sending an interrupt signal.
  - The CPU interrupts whatever it is doing and serves the request
  - The advantage of interrupts is that the CPU can serve many devices
  - Each receives a service based on its priority
  - Disadvantage of interrupts is that they require more hardware and software

- In polling, CPU monitors continuously a status condition and when the conditions are met, it will perform the service.
  - In contrast, polling is cheap and requires minimal software
  - But it ties down the CPU

Example



From the data segment:

```
MY_DATA     DB    "Hi. How are you?",CR,LF
            DB    "I am fine. How are you?",CR,LF,"$"
PA          EQU   300H          ;port A address
PB          EQU   301H          ;port B address
PC          EQU   302H          ;port C address
CWP         EQU   303H          ;control word port
LF          EQU   0AH           ;line feed
CR          EQU   0DH           ;carriage return
```

From the code segment:

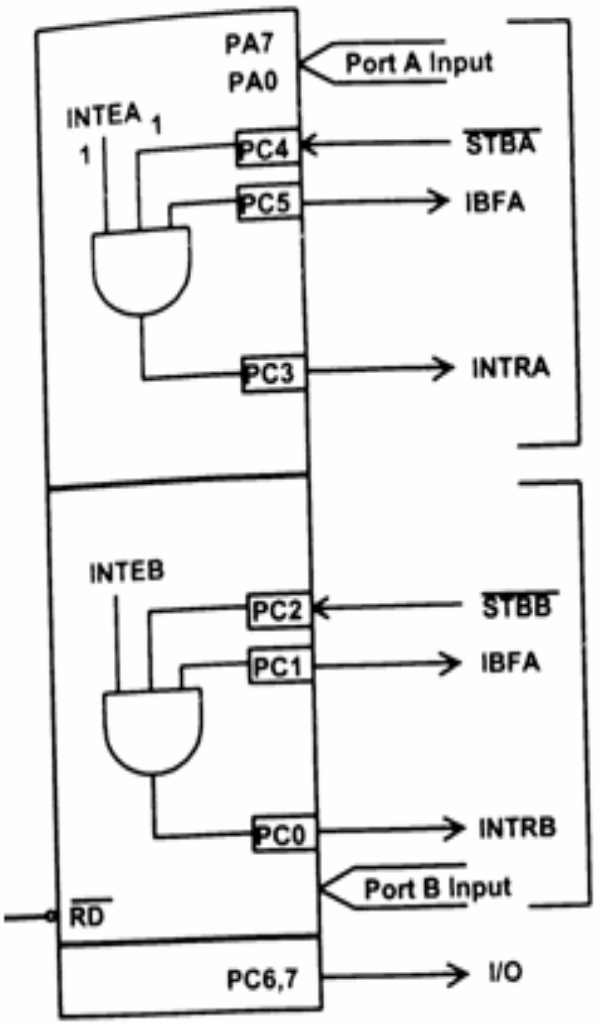
```
            MOV   AL,10100000B  ;control word PA=out mode 1
            MOV   DX,CWP        ;DX =  303 control word port
            OUT   DX,AL         ;issue control word
```

```
MOV SI,OFFSET MY_DATA
AGAIN: MOV AH,[SI]
CMP AH,'$'
JZ OVER
BACK: MOV DX,PC       ;check port C
IN AL,DX
AND AL, 80            ;check OBF
JZ BACK
MOV DX,PA
MOV AL,AH
OUT DX,AL
INC SI
JMP AGAIN
```

```
OVER:       ...                 ;go back to DOS
```

OBF signal can also be checked (easier)instead of enabling INTA

Mode 1 Strobed Input



Port A with

Handshake Signals

Control Word -- Mode 1 Input

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	1/0	1	1	x

I/O Mode

Port A Mode 1

Port A Input

Port B Input

Port B Mode 1

PC 6,7

1 = Input

0 = Output

Port B with

Handshake Signals

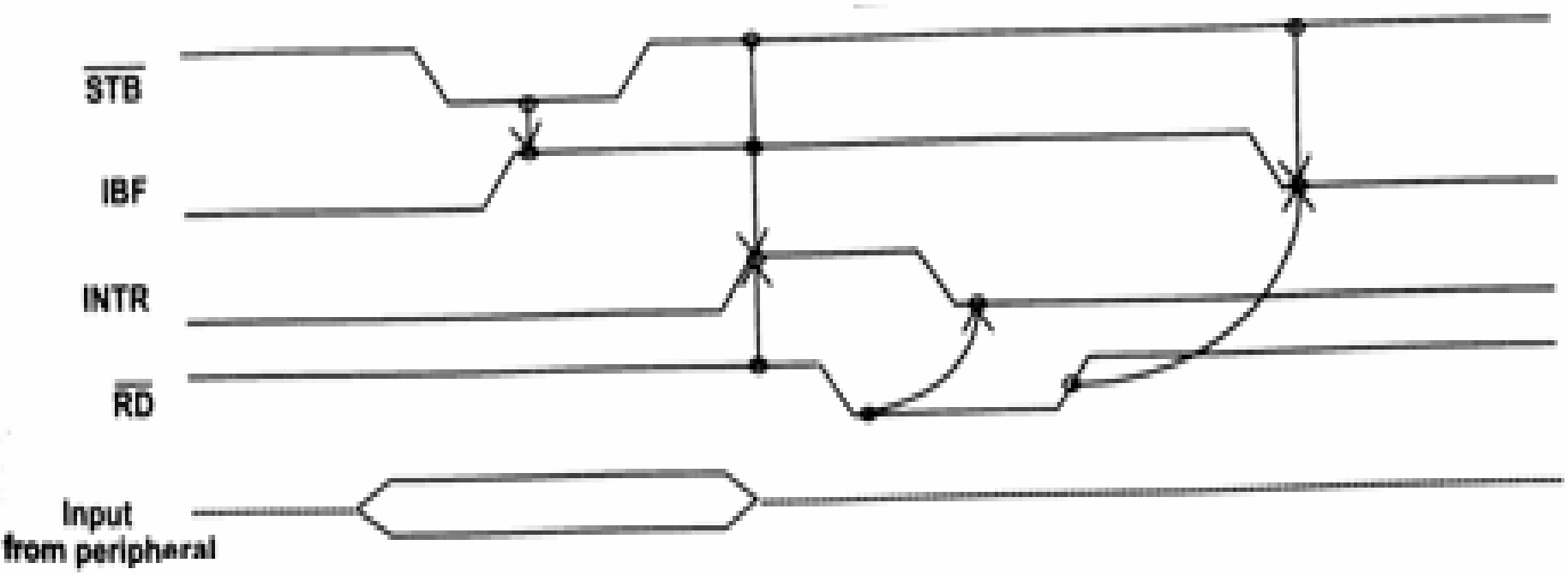
Status Word -- Mode 1 Input

D7	D6	D5	D4	D3	D2	D1	D0
I/O	I/O	IBFA	INTEA	INTRA	INTEB	IBFB	INTRB

INTEA is controlled by PC4 in BSR mode.

INTEB is controlled by PC2 in BSR mode.

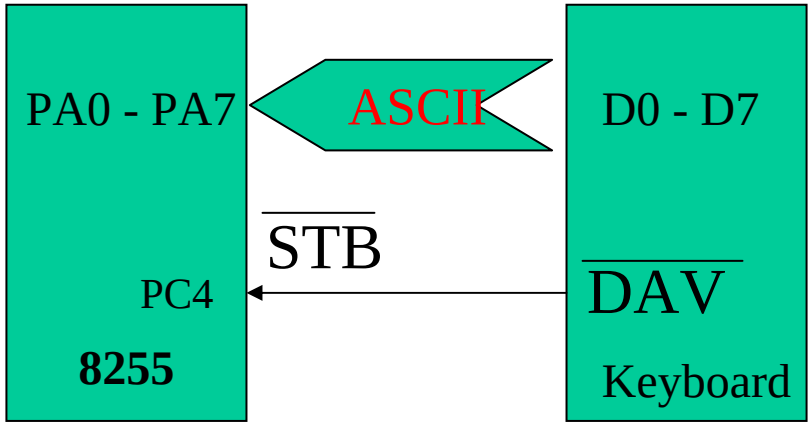
Mode 1 Strobed Input



Mode 1 Input Ports with Handshaking Signals

- $\overline{\text{STB}}$
  - When an external peripheral device provides a byte of data to an input port, it informs the 8255 through the STB pin. STB is of limited duration
- IBF (Input Buffer Full)
  - In response to STB, the 8255 latches into its internal register the data present at PA0-PA7 or PB0-PB7.
  - Through IBF it indicates that it has latched the data but it has not been read by the CPU yet
  - To get the attention of the CPU, IBF activates INTR
- INTR
  - Falling edge of RD makes INTR low
  - The RD signal from the CPU is of limited duration and when it goes high the 8255 in turn makes IBF inactive by setting it low
  - IBF in this way lets the peripheral know that the byte of data was latched by the 8255 and read into the CPU as well.
- The two flip flops INTEA and INTB are set/reset using the BSR mode. The INTEA is enabled or disabled through PC4 and INTEB is enabled or disabled through PC2.

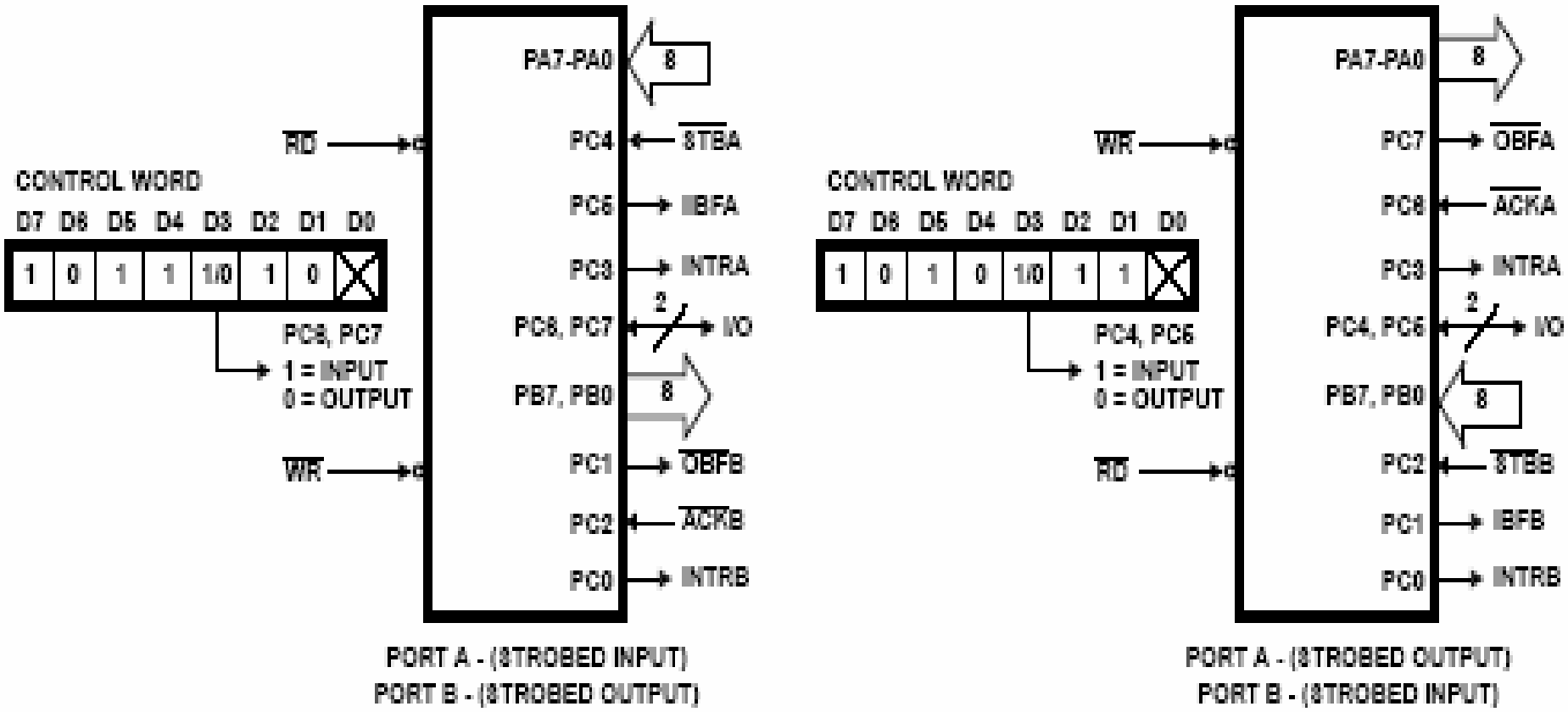
Mode 1 Strobed Input Example



Write a procedure that reads data from the keyboard each time a key is typed.

```
BIT5     EQU 20h
PORTC    EQU 22h
PORTA    EQU 20h
READ     PROC NEAR
         IN AL, PORTC
         TEST AL, BIT5 ;check on IBF?
         JZ  READ
          IN AL, PORTA
          RET
READ     ENDP
```

Mode 1 Other Configurations



Combinations of Mode 1: Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

FIGURE 10. COMBINATIONS OF MODE 1

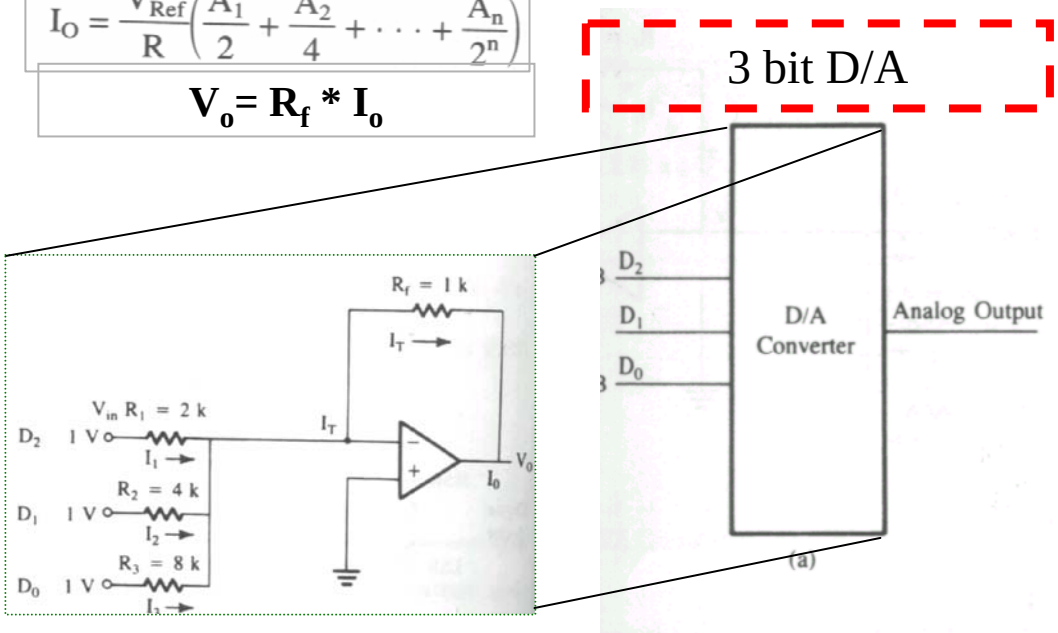
Interfacing DAC to a PC

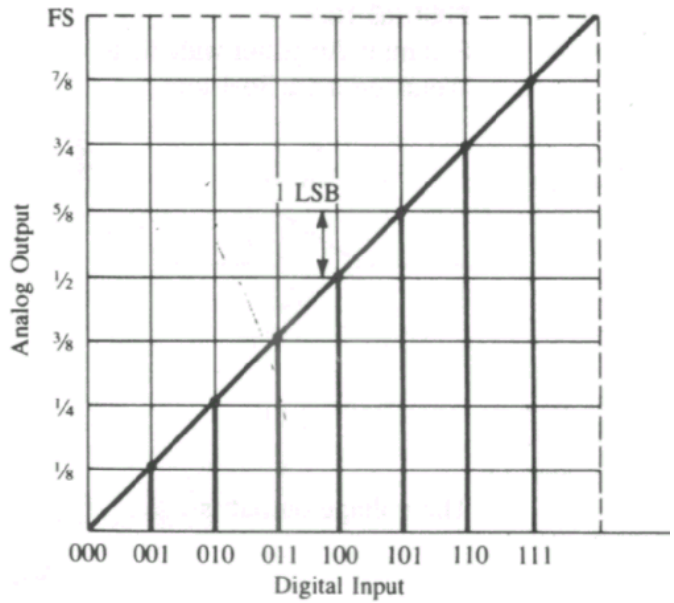
- The digital to analog converter (DAV) is a device widely used to convert digital pulses to analog signals.
- The first criterion to judge a DAC is its resolution which is the a function of the number of binary inputs
- The number of analog input levels is 2^n where n is the number of data bit inputs.
- Therefore the 8 input DAC such as the MC 1408 (DAC 808) provides 256 discrete voltage (or current) levels of output.
- $I_{out} = I_{ref}$ (D7/2 + D6/4 + D5/8 + D4/16 + D3/32 + D2/64 + D1/128 + D0 / 256)

DAC BASICS

$$I_O = \frac{V_{Ref}}{R}\left(\frac{A_1}{2} + \frac{A_2}{4} + \cdots + \frac{A_n}{2^n}\right)$$

$$V_o = R_f * I_o$$

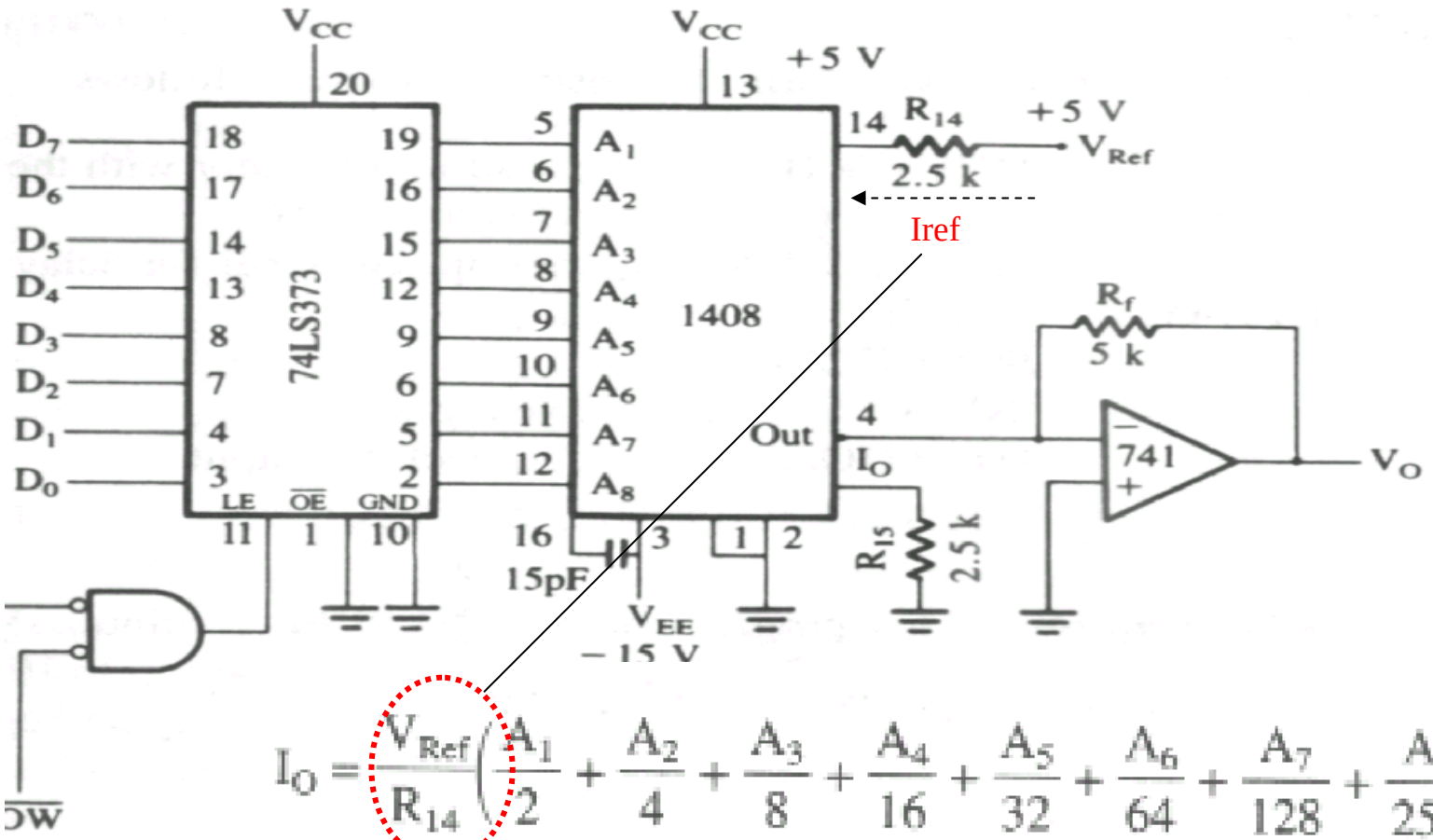




Calculate the values of the LSB, MSB and full scale output for an 8 bit DAC for the 0-10v range?

1. LSB = $1/2^8 = 1/256$
   for 10 v, LSB = 10/256
2. MSB = ½ full scale = 5v
3. Full Scale Output = (Full Scale Value – 1 LSB)
   = 10 v – 0.039v = 9.961v

Interfacing DAC via 8255



$$I_O = \frac{V_{Ref}}{R_{14}}\left(\frac{A_1}{2} + \frac{A_2}{4} + \frac{A_3}{8} + \frac{A_4}{16} + \frac{A_5}{32} + \frac{A_6}{64} + \frac{A_7}{128} + \frac{A_8}{256}\right)$$

$$I_O = \frac{5\text{ V}}{2.5\text{ k}}\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256}\right)$$

$$= 2\text{ mA}\ (255/256)$$

$$= 1.992\text{ mA}$$

For ex:

If $R_{14} = 2.5$

Example

- Assume that R=5K and Iref= 2ma calculate Vout for:

10011001

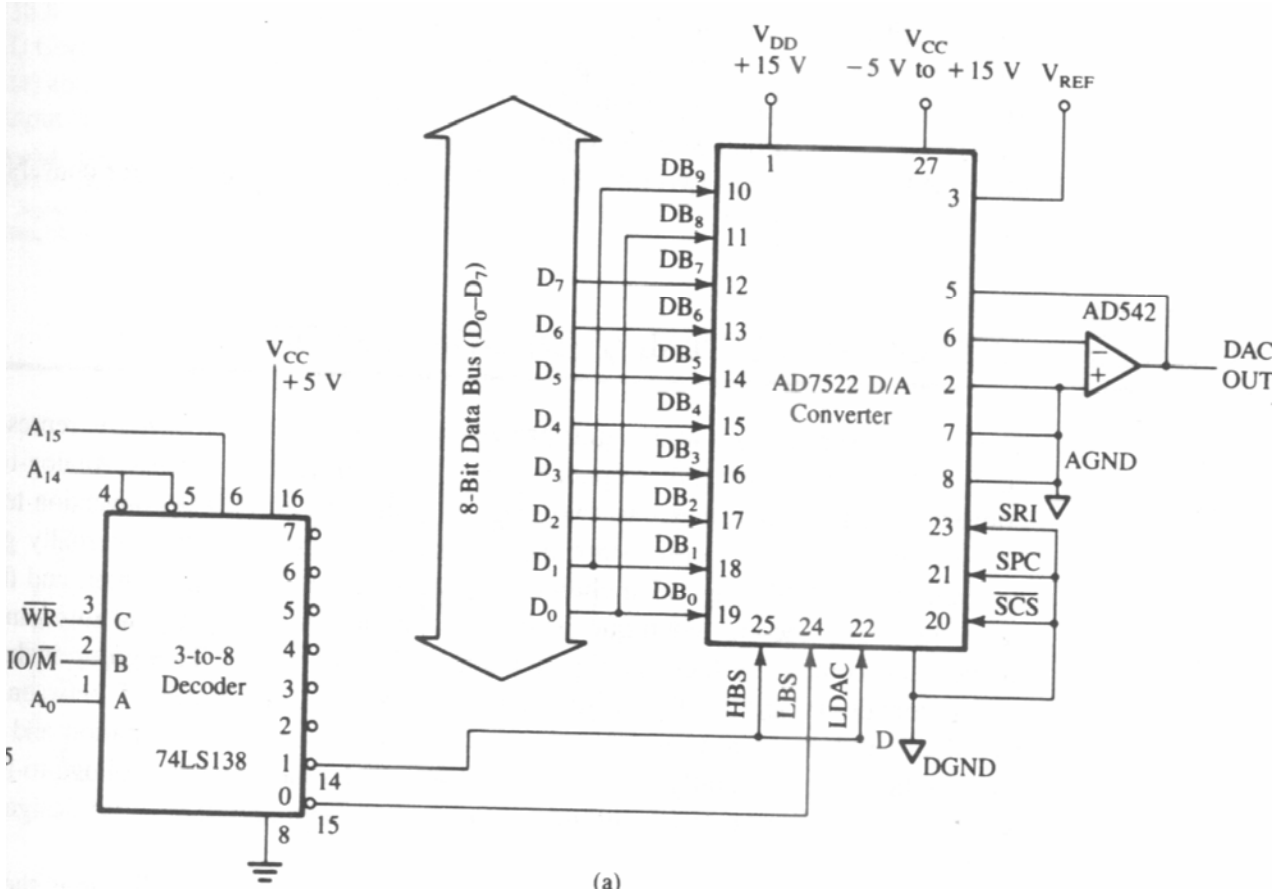
Iout = 2mA (153/255) = 1.195 mA

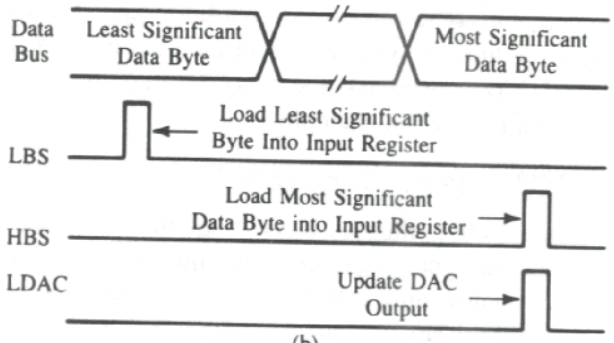
Vout = 1.195 x 5 K = 5.975V

Example - Generate a Stair-Step Ramp

```
        MOV   AL,80H
        MOV   DX,303H
        OUT   DX,AL
A1:     MOV   AH,01
        INT   16H
        JNZ   STOP
        SUB   AL,AL
        MOV   DX,300H
A2:     OUT   DX,AL
        INC   AL
        CMP   AL,0
        JZ    A1
        MOV   CX,02FFH
WT:     LOOP  WT
        JMP   A2
        MOV   AH,4CH
        INT   21H
```

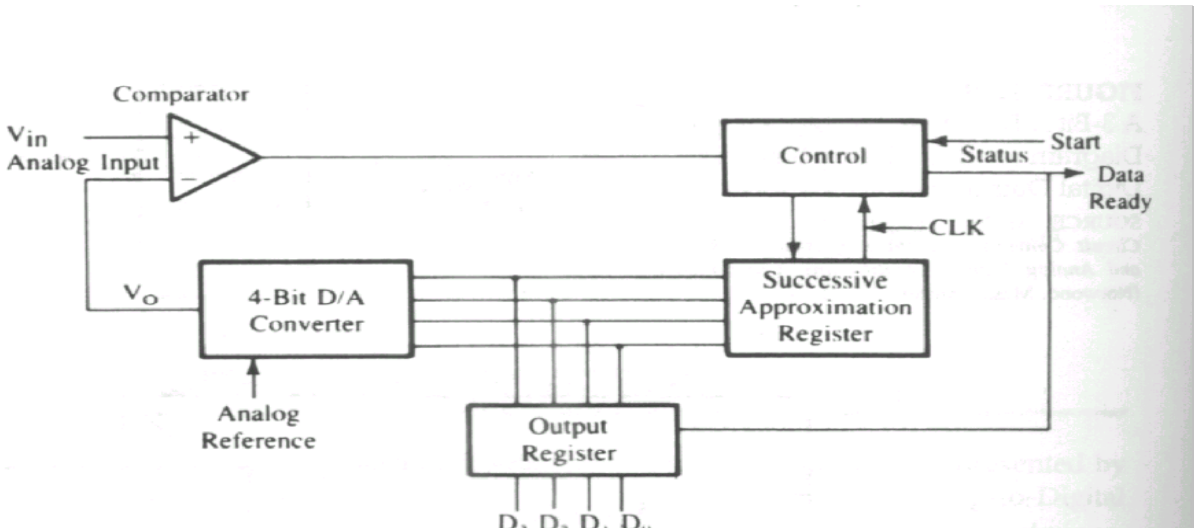
10 bit D/A operation

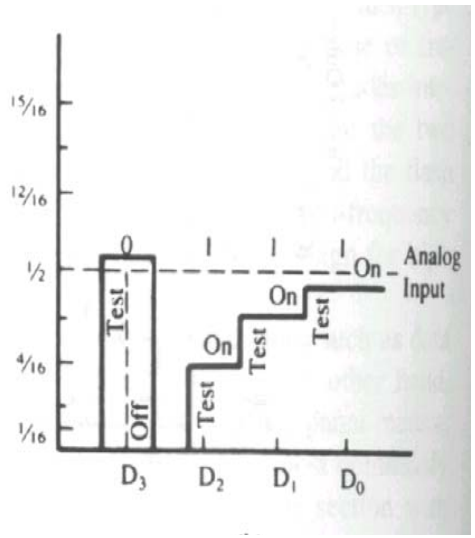




Interfacing ADC to a PC

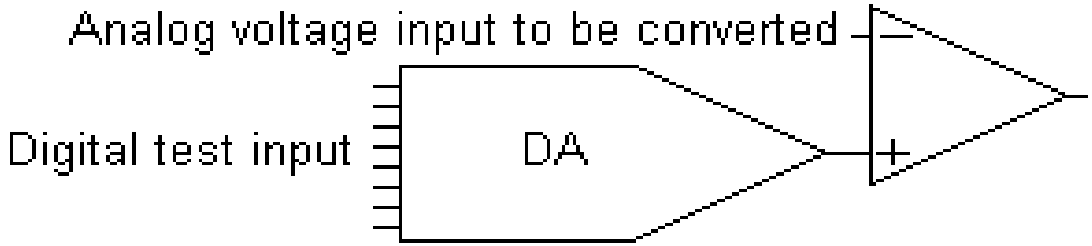
- A physical quantity (temperature, pressure, humidity, velocity) is converted to electrical (voltage, current) signals using a device called a transducer (sensor)
- We need an analog-to-digital converter (ADC) to translate the analog signals to digital numbers so that the PC can read them
- The techniques for ADC are
  - Successive Approximation Technique
    - Conversion Speed is important (Critical Conversion Time)



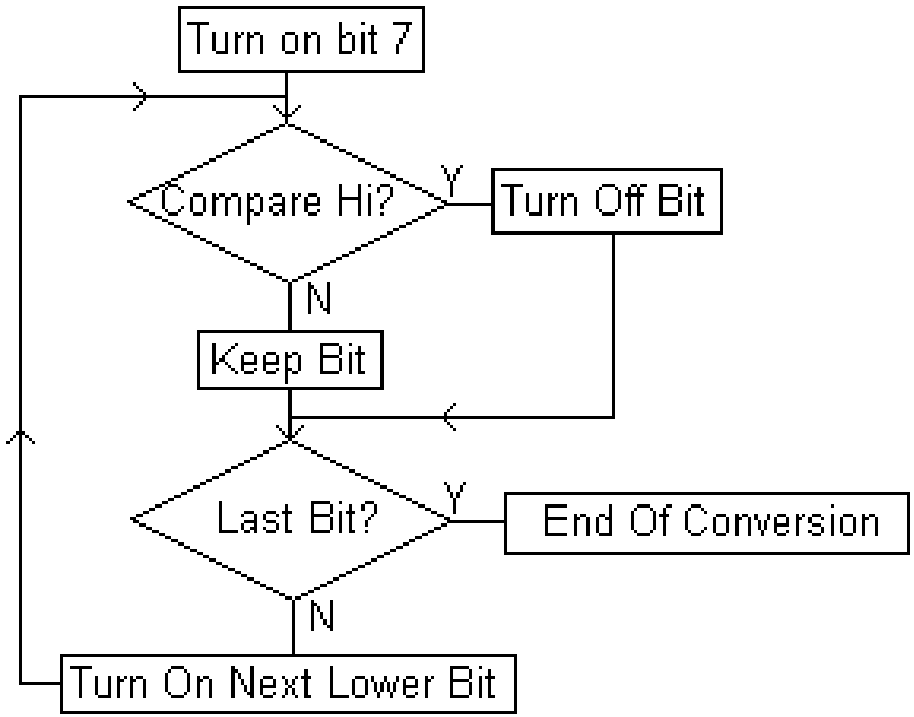


  - Integrating Type Converters
    - Conversion Accuracy is important

Successive Approximation Technique



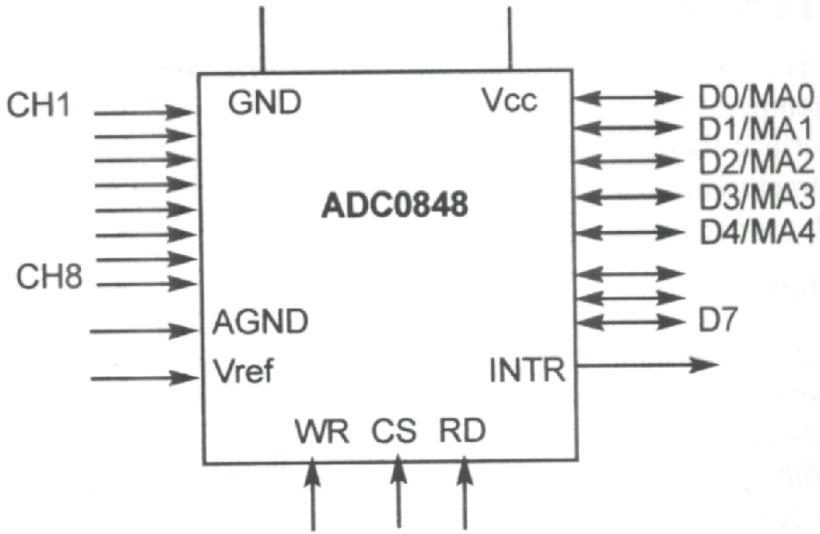
:



Contd

Test Bit	DA Binary Value	Decimal	(5 * DAvalue)/256	Comparison Result
10000000	10000000	128	2.5	low - keep bit
01000000	11000000	192	3.75	high - drop bit
00100000	10100000	160	3.125	low - keep bit
00010000	10110000	176	3.4375	high - drop bit
00001000	10101000	168	3.28125	high - drop bit
00000100	10100100	164	3.203125	low - keep bit
00000010	10100110	166	3.2421875	high - drop bit
00000001	10100101	165	3.22265625	high - drop bit

ADC0848



MA0 – MA4 (multiplexed address)

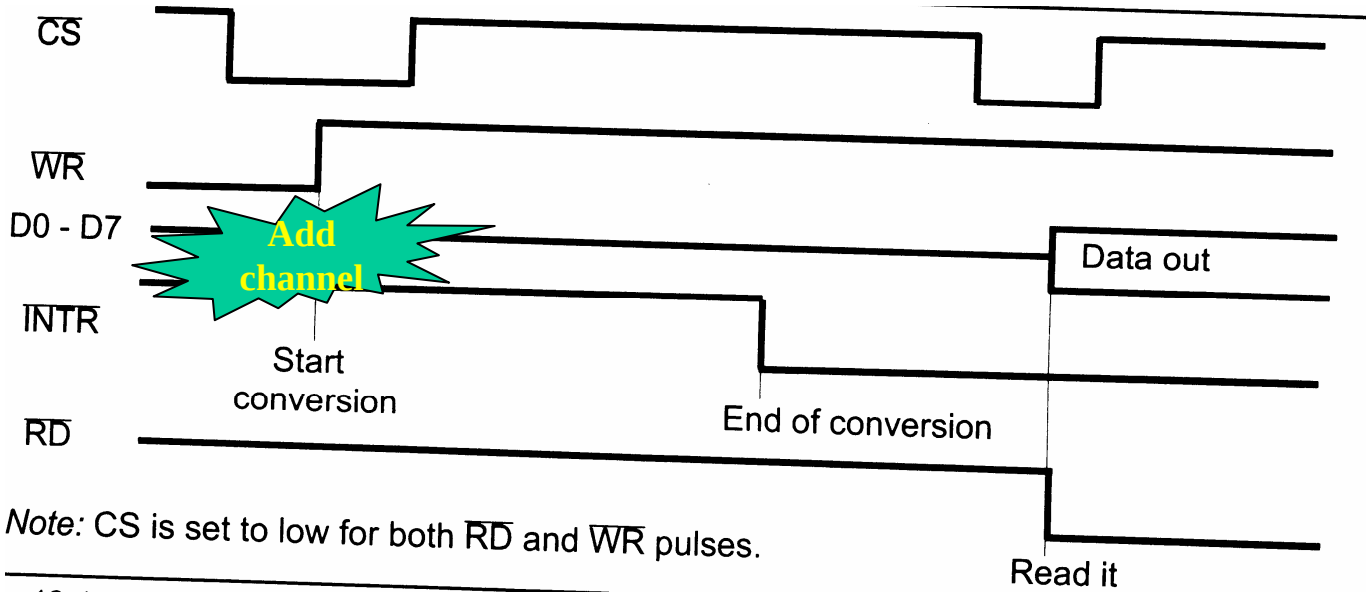
Table 12-11: ADC0848 Analog Channel Selection (Single-Ended Mode)

Selected Analog Channel	MA4	MA3	MA2	MA1	MA0
CH1	0	1	0	0	0
CH2	0	1	0	0	1
CH3	0	1	0	1	0
CH4	0	1	0	1	1
CH5	0	1	1	0	0
CH6	0	1	1	0	1
CH7	0	1	1	1	0
CH8	0	1	1	1	1

Note: Channel is selected when CS = 0, RD =1, and an L-to-H pulse is applied to WR.

- ADC 0848
  - –conversion time less than 110 μs
  - –RD active low: ADC converts the analog input to its binary equivalent and holds it in an internal register; with RD from high to low the data in the register shows at the D0-D7 pins
  - –WR start conversion: low to high input to inform the 804 to start the conversion process
  - –when the data conversion is complete, the INTR pin is forced low by the 804
  - –after INTR goes low, we make CS = 0 and send a high to low pulse to the RD pin to get the data out of the 804

ADC operation



1. Make CS = 0 and send a low to high pulse to the WR pin
2. Keep monitoring the INTR pin. If INTR is low, the conversion is finished and e can go to the next step
3. We then make CS = 0 again and send a high to low pulse to the RD pin to get the data pout of the 804 chip

ADC 804 Pins

- Vin (+) and Vin (-) are the differential analog inputs
- Vin = Vin (+) - Vin (-) ; to set this mode use MA4 and MA3 low.
- Vref is an input voltage used as the reference voltage

Vref (volts)	Vin (volts)	Step Size (mv)
5	0 to 5	5/256 = 19.53
4	0 to 4	4/256 = 15.62
3	0 to 3	3/256 = 11.71
2	0 to 2	2/256 = 7.81
1	0 to 1	1/256 = 3.90

Dout = Vin / (step size)

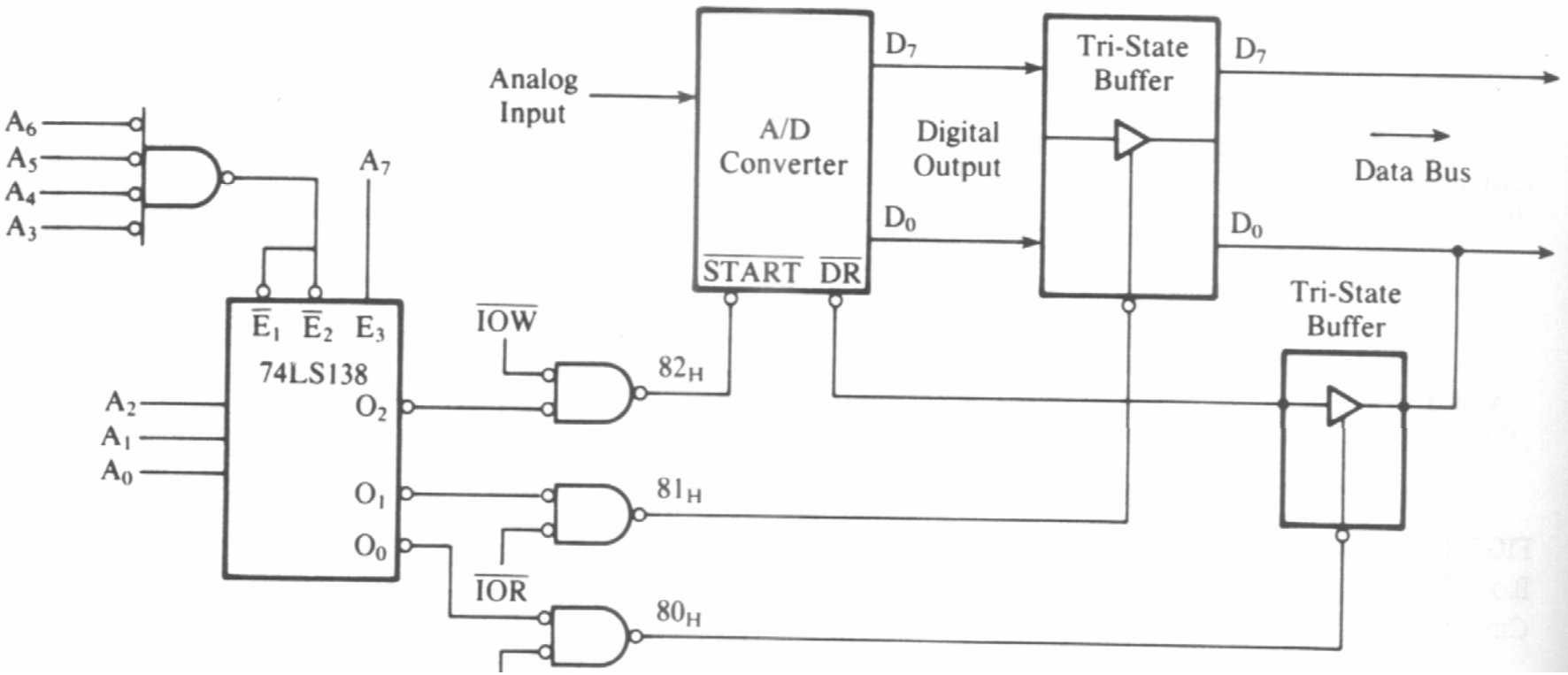
For given ADC, Vref = 2.56. Calculate the D0-D7 output if the analog input is

a) 1.7 v

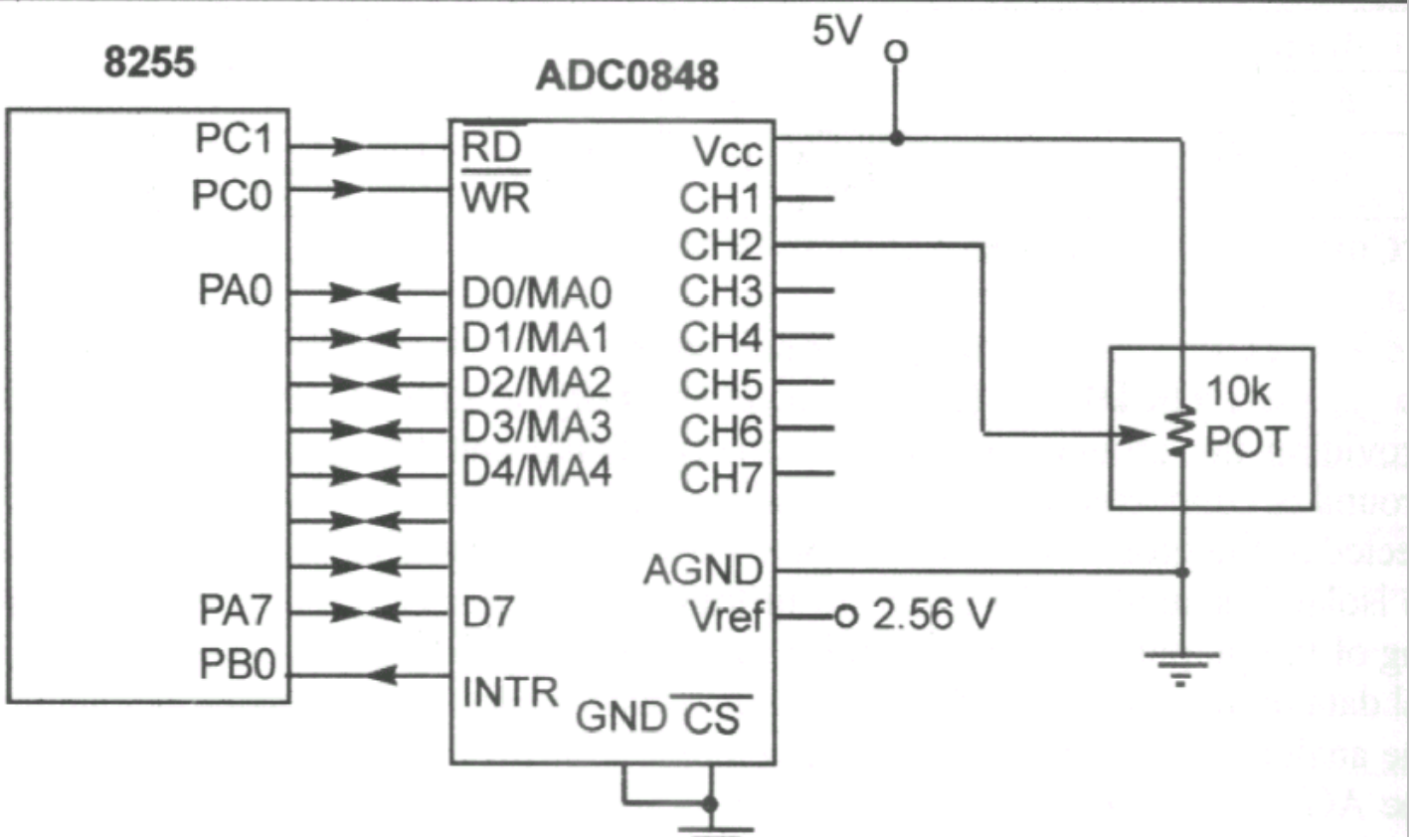
b) 2.1 v

Answer(a): Step size is 2.56/256=10mv
Dout = 1.7/10mv =170
170 is 10101011

Using A/D with status check



Connecting the 804



PA0-PA7 to DO-D7 of ADC	CHANNEL SELECTION(out), DATA READ (in)
PB0 TO INTR	PORT B AS INPUT
PC0 TO WR	PORT C AS OUTPUT
PC1 TO RD	PORT C AS OUTPUT

Required Steps

- CS=0 WR=0
  - Provide the address of the selected channel on DB0 – DB7
  - Apply a WR pulse
  - Channel 2 address is 09h, Channel 1 address is 08h , etc.
  - Not only we select the channel but conversion starts!
- While WR=1
  - Keep monitoring INTR asserted low
  - When INTR goes low, conversion finished
- After INTR becomes low
  - CS=0 and WR=1 and apply a low pulse to the RD pin to get the data from the 848 IC chip

Example

```
MOV AL,82h   ;PA=out PB=in PC=out
MOV DX,CNT_PORT
OUT DX,AL
MOV AL,09  ;channel 2 address
MOV DX,PORT_A
OUT DX,AL
MOV AL,02  ;WR=0 RD=1
MOV DX,PORT_C   ; not only selects channel but also
                ; starts conversion
OUT DX,AL
CALL DELAY          ;few microsecs
MOV AL,03           ; WR=1 RD=1
OUT DX,AL
CALL DELAY
MOV AL,92h          ; PA=in PB=in PC=out
MOV DX,CNT_PORT
OUT DX,AL
```

Example

```
MOV DX,PORT_B
B1: IN AL,DX
AND AL,01
CMP AL,01
JNE B1
MOV AL,01  ;RD=0
MOV DX,PORT_C
OUT DX,AL
MOV DX,PORT_A
IN AL,DX  ;get the converted data
```