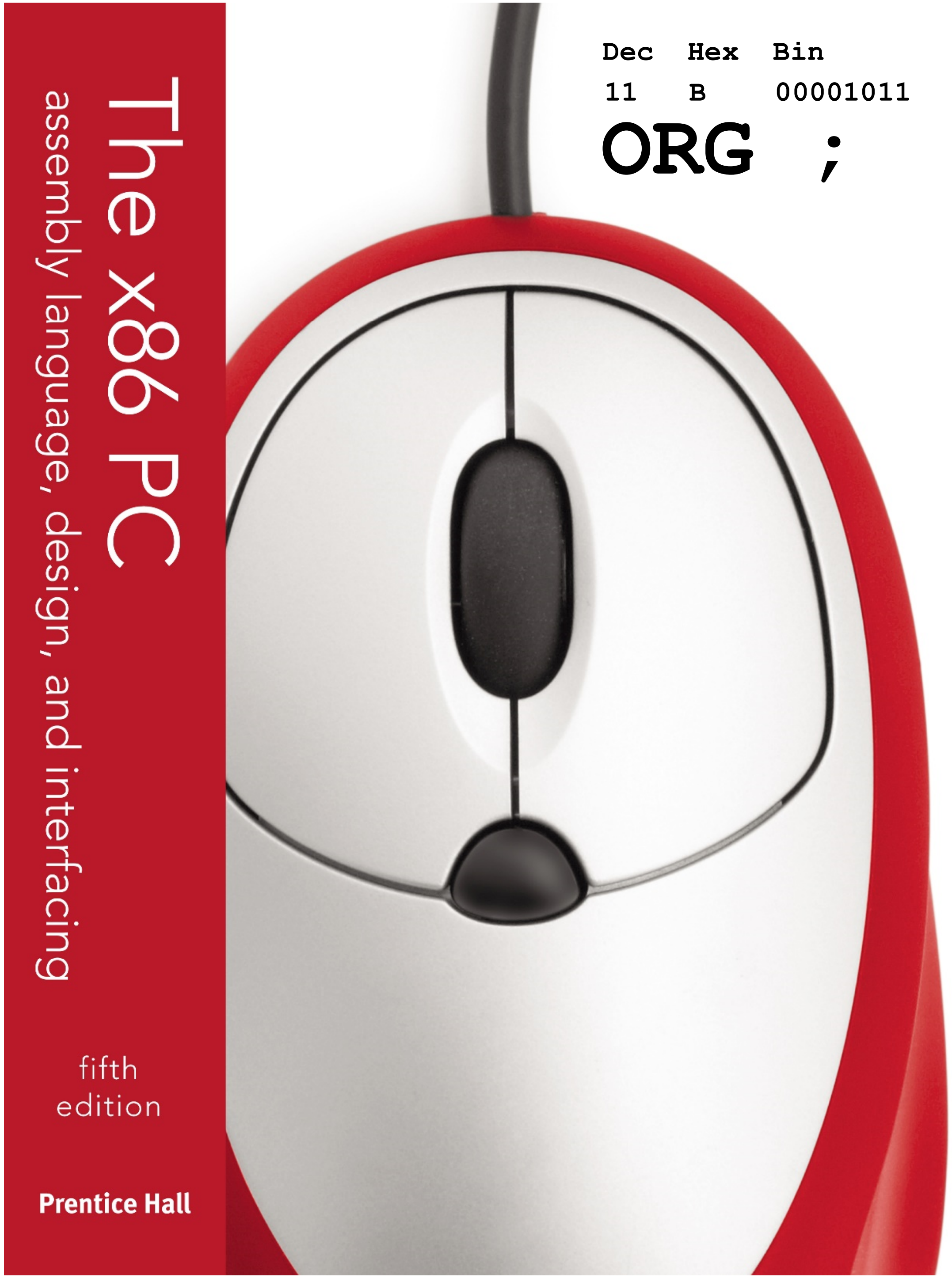


Dec	Hex	Bin
11	B	00001011

ORG ;

I/O PROGRAMMING

The x86 PC

assembly language,
design, and interfacing

fifth edition

MUHAMMAD ALI MAZIDI
JANICE GILLISPIE MAZIDI
DANNY CAUSEY

OBJECTIVES
this chapter enables the student to:

- Code Assembly language instructions to read and write data to and from I/O ports.
- Diagram the design of peripheral I/O using the 74LS373 output latch and the 74LS244 input buffer.
- Describe the I/O address map of x86 PCs.
- List the differences in memory-mapped I/O versus peripheral I/O.
- Describe the purpose of a simple programmable peripheral interface chip.

11.1: 8088 INPUT/OUTPUT INSTRUCTIONS

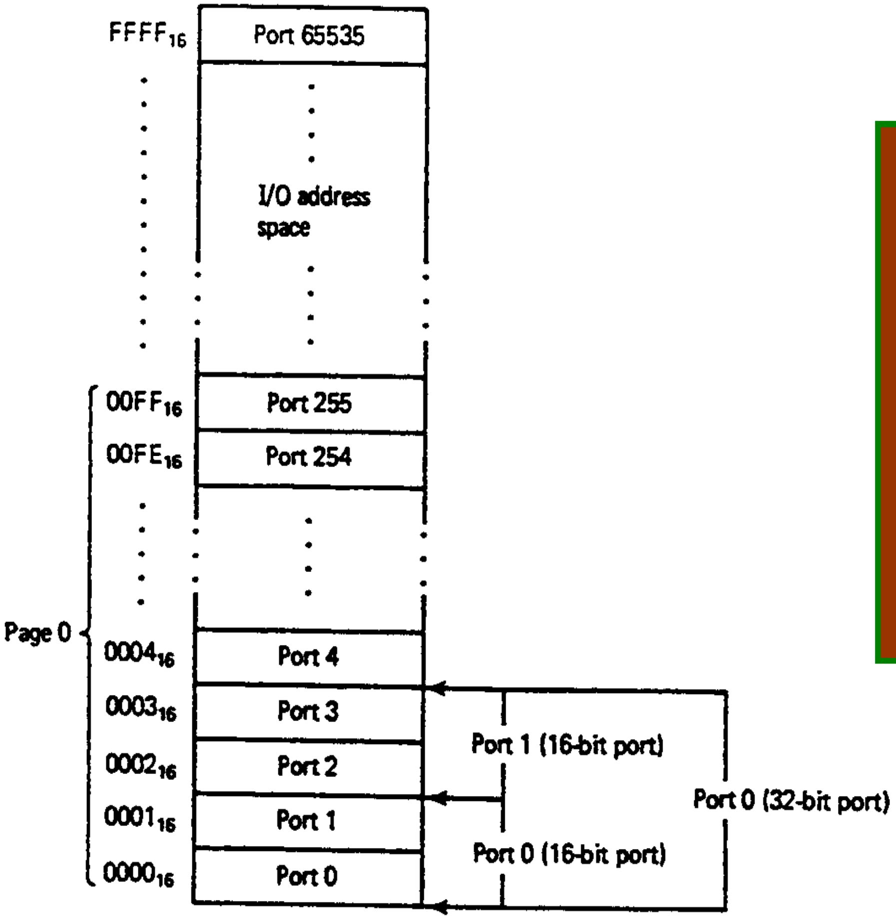
- All x86 processors, 8088 to Pentium®, can access external devices called *ports* using I/O instructions.
  - Memory can contain both opcodes and data.
  - I/O ports contain data only
    - Two instructions: “OUT” and “IN” send data from the accumulator (AL or AX) to ports or bring data from ports into the accumulator.

11.1: 8088 INPUT/OUTPUT INSTRUCTIONS
8-bit data ports

- 8088 I/O operation is applicable to all x86 CPUs.
  - The 8-bit port uses the D0–D7 data bus for I/O devices.
- Register **AL** is used as the source/destination for IN/OUT instructions.
  - To input or output data from any other registers, the data must first be moved to the AL register.
    - Instructions OUT and IN have the following formats:

	Inputting Data		Outputting Data	
Format:	IN	dest,source	OUT	dest,source
(1)	IN	AL,port#	OUT	port#,AL
(2)	MOV	DX,port#	MOV	DX,port#
	IN	AL,DX	OUT	DX,AL

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

Input Bus Cycle of the 8088

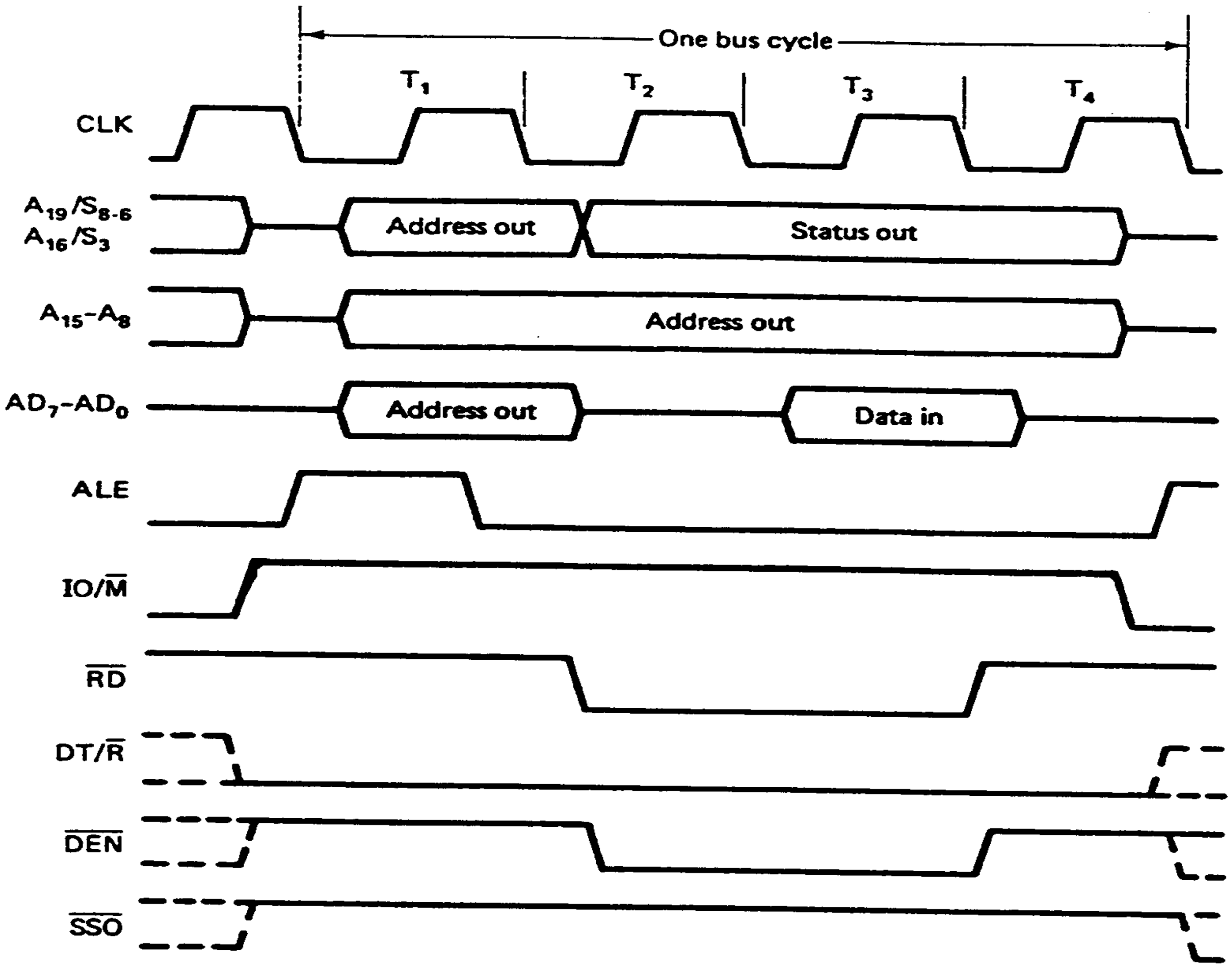


Figure 8–52 Input bus cycle of the 8088.

Output Bus Cycle of the 8088

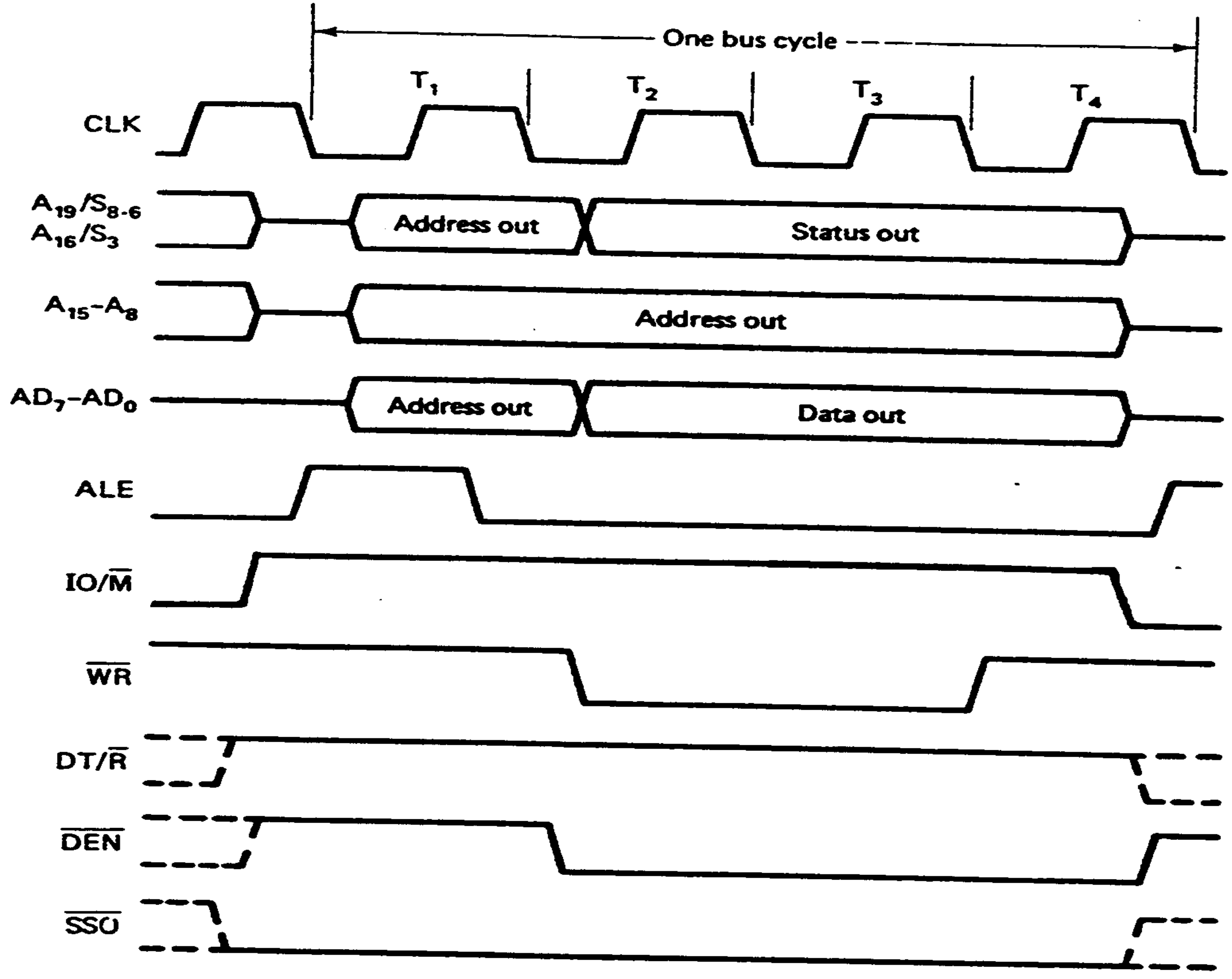


Figure 8–53 Output bus cycle of the 8088.

how to use I/O instructions

- I/O instructions are used in programming 8- and 16-bit peripheral devices.
  - Printers, hard disks, and keyboards.

- For an 8-bit port, use *immediate addressing*:

```
MOV    AL,36H        ;AL=36H
OUT    43H,AL        ;send value 36H to port address 43H
```

11.1: 8088 INPUT/OUTPUT INSTRUCTIONS
how to use I/O instructions

- 16-bit port address instruction using *register indirect addressing* mode with register **DX**.
  - This program toggles port address 300H continuously.

```
BACK: MOV   DX,300H      ;DX = port address 300H
      MOV   AL,55H
      OUT   DX,AL        ;toggle the bits
      MOV   AL,0AAH
      OUT   DX,AL        ;toggle the bits
      JMP   BACK
```

  - Only **DX** can be used for 16-bit I/O addresses.
  - Use register AL for 8-bit data.

I/O Instructions

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
```

how to use I/O instructions

Example shows decision making based on the data that was input.

Example 11-1

In a given 8088-based system, port address 22H is an input port for monitoring the temperature. Write Assembly language instructions to monitor that port continuously for the temperature of 100 degrees. If it reaches 100, then BH should contain 'Y'.

Solution:

```
BACK:       IN    AL,22H   ;get the temperature from port # 22H
            CMP   AL,100       ;is temp = 100?
            JNZ   BACK         ;if not, keep monitoring
            MOV   BH,'Y        ;temp = 100, load 'Y' into BH
```

Input Bus Cycle of the 8088

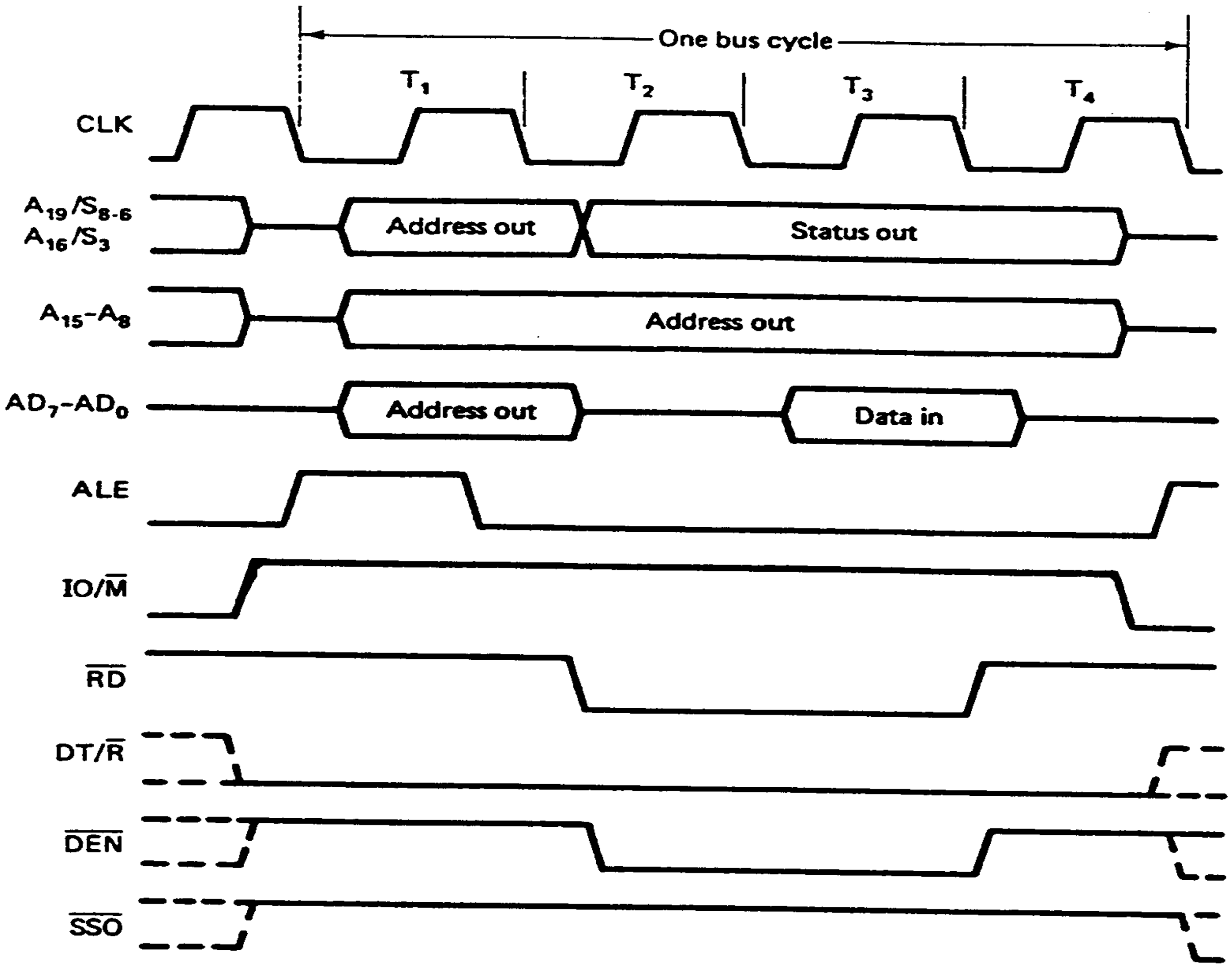


Figure 8–52 Input bus cycle of the 8088.

Output Bus Cycle of the 8088

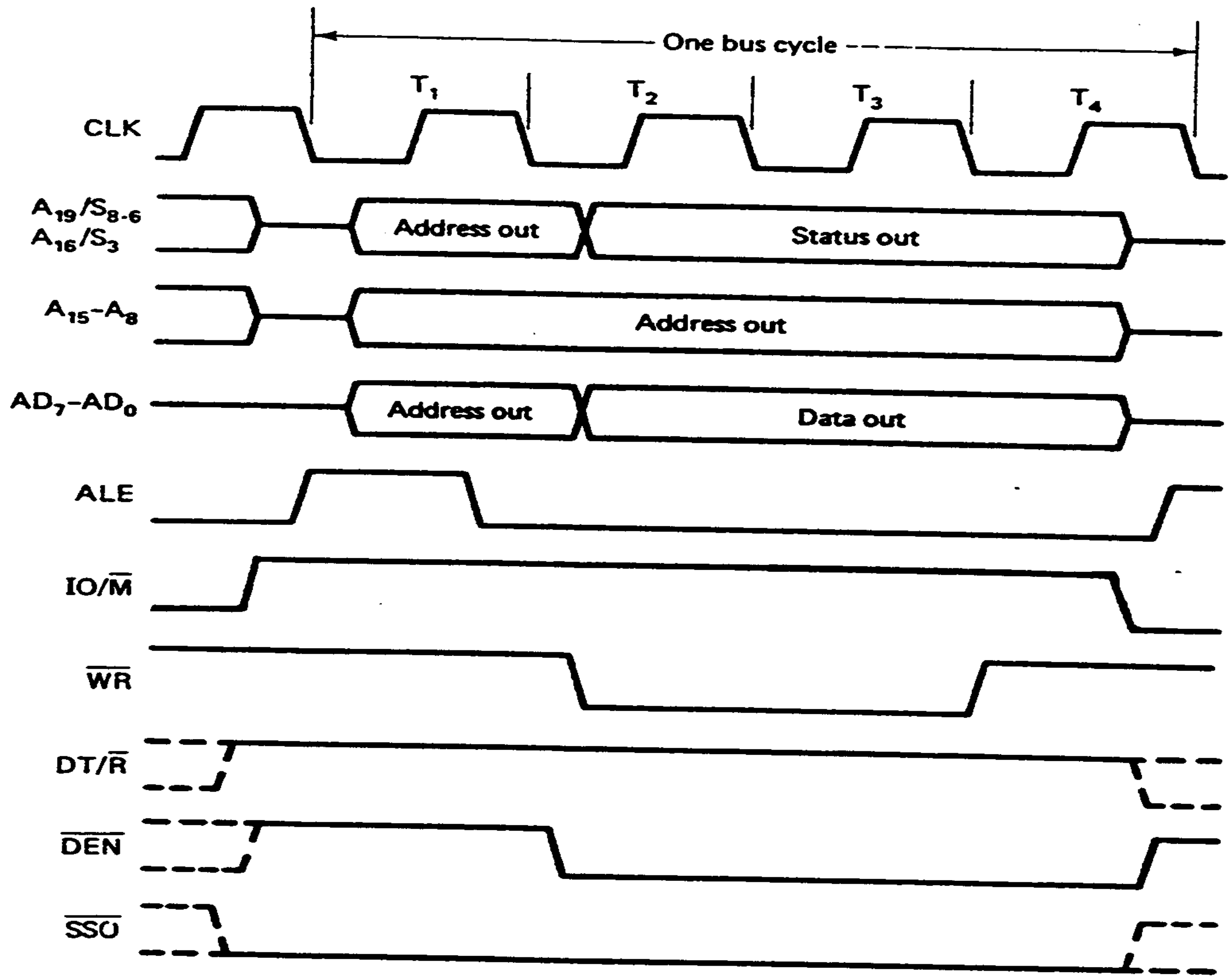


Figure 8–53 Output bus cycle of the 8088.

I/O Example

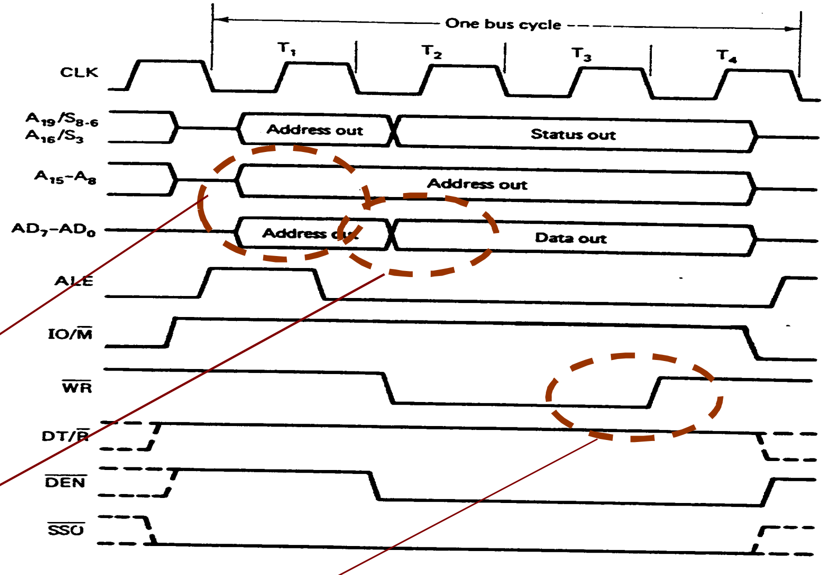
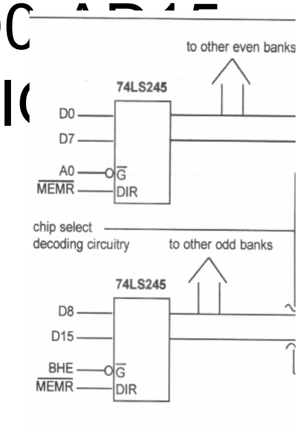


Figure 8–53 Output bus cycle of the 8088.

- 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 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

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 I[...] 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

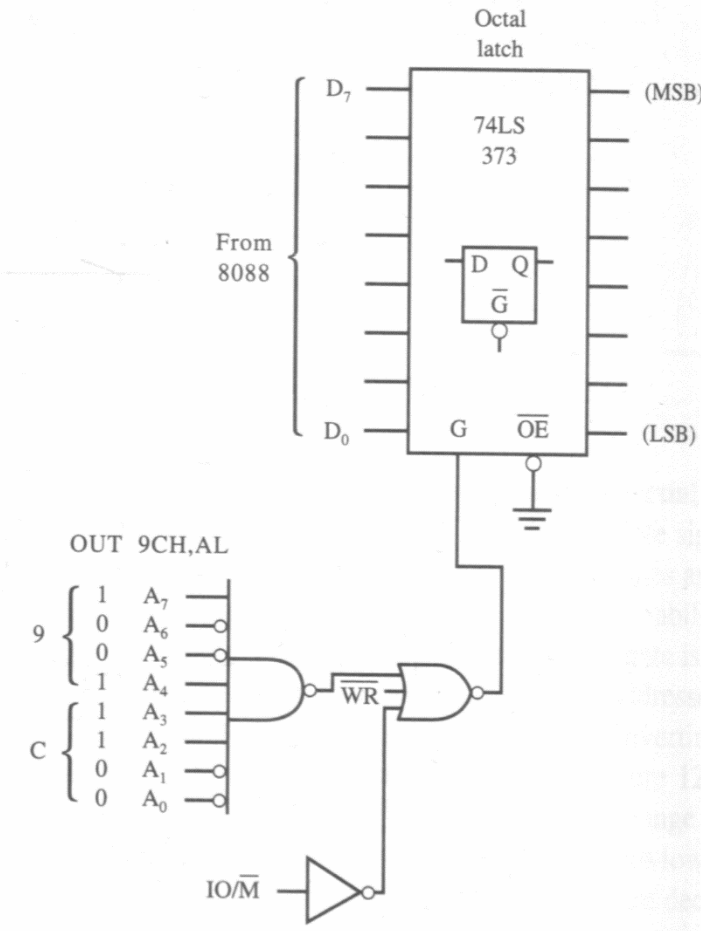


I/O Design in the 8088/86

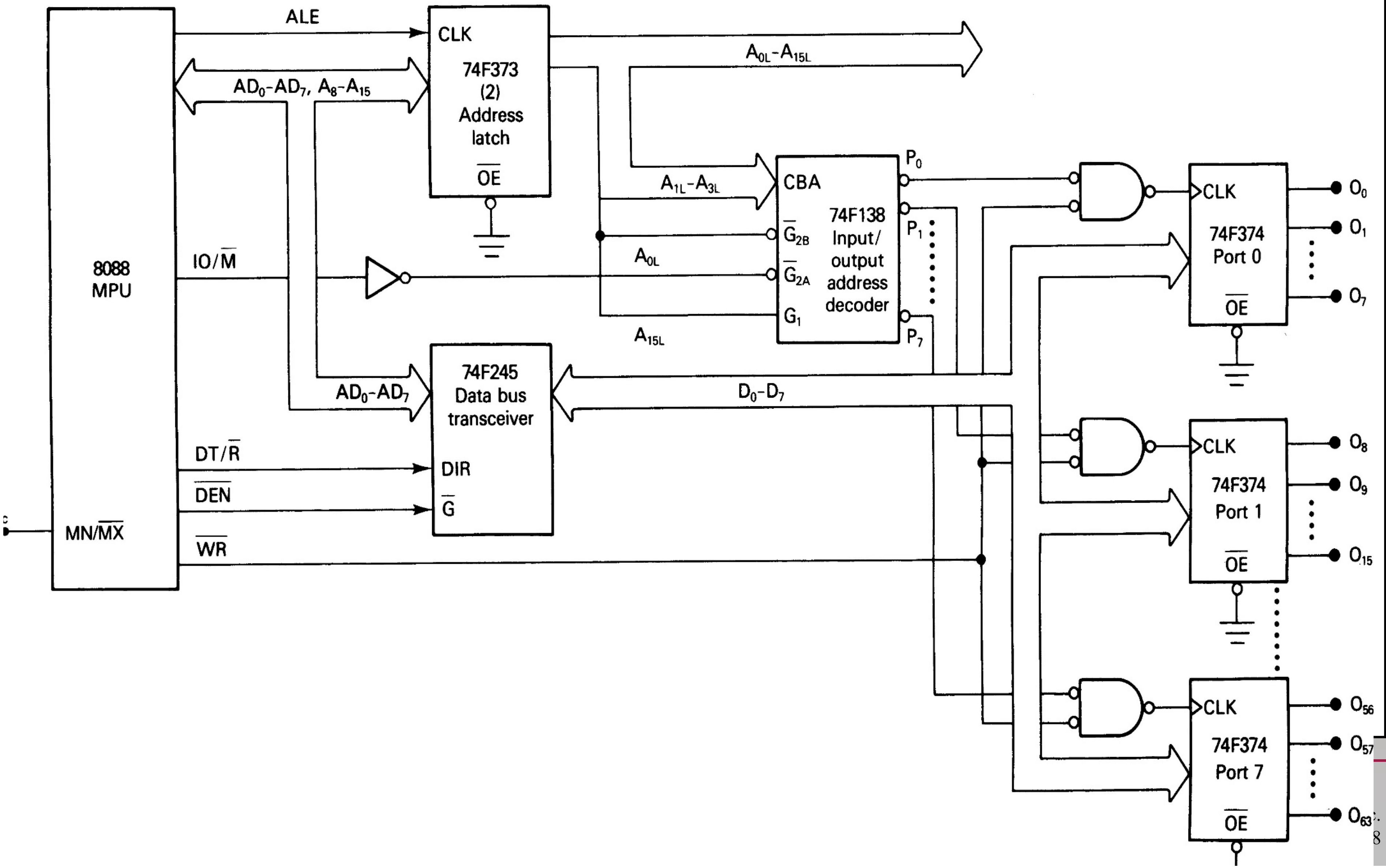
- In every computer, when data is sent out by the CPU, the data on the data bus must be latched by the receiving device
- While memories have an internal latch to grab the data on the data bus, a latching system must be designed for ports
- Since the data provided by the CPU to the port is on the system 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
  - 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

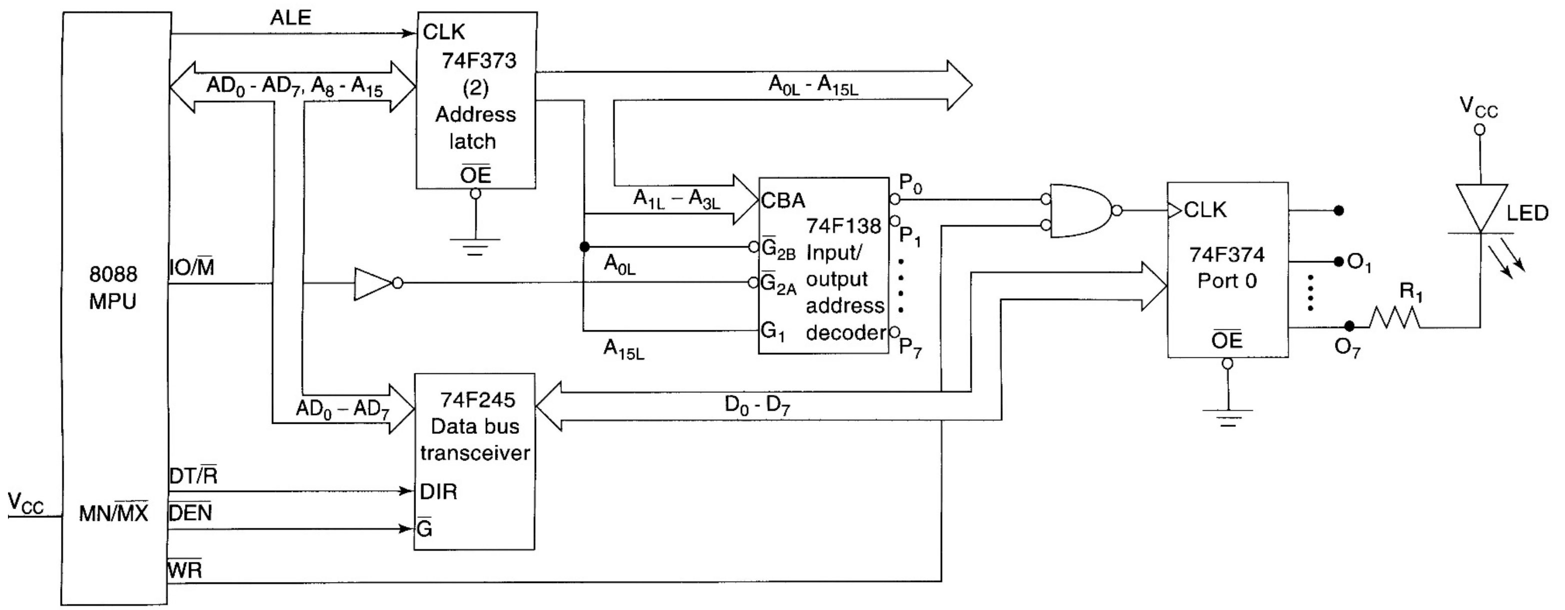


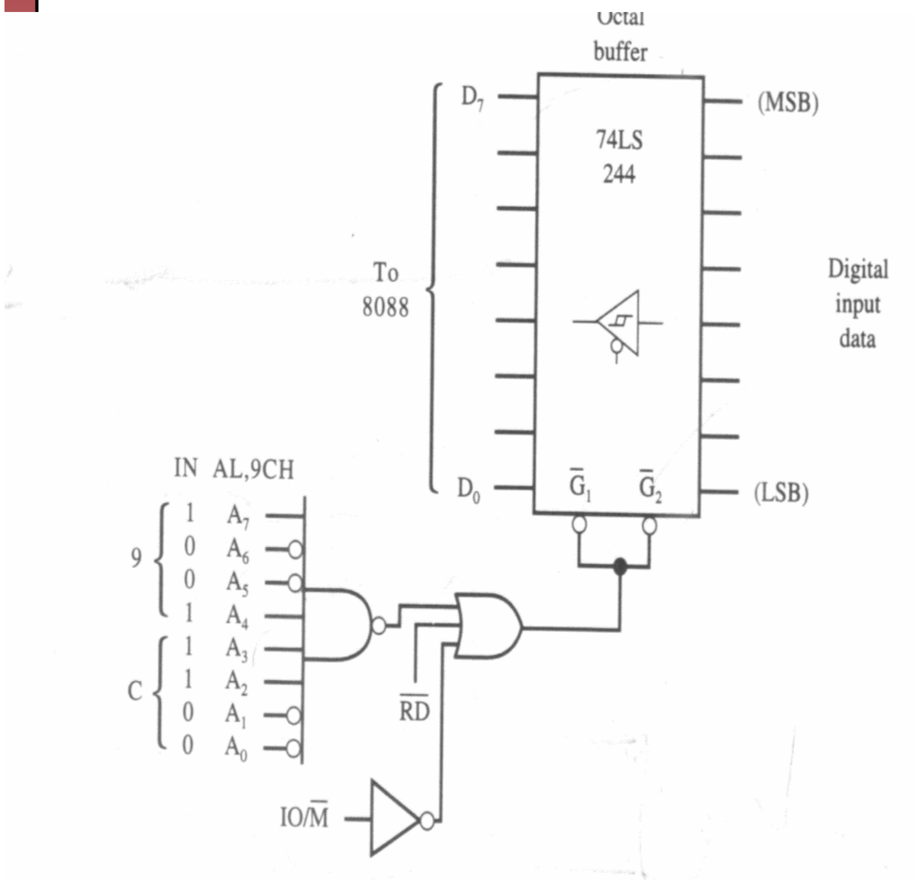
Figure 10–2 Driving an LED connected to an output port.

```
            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

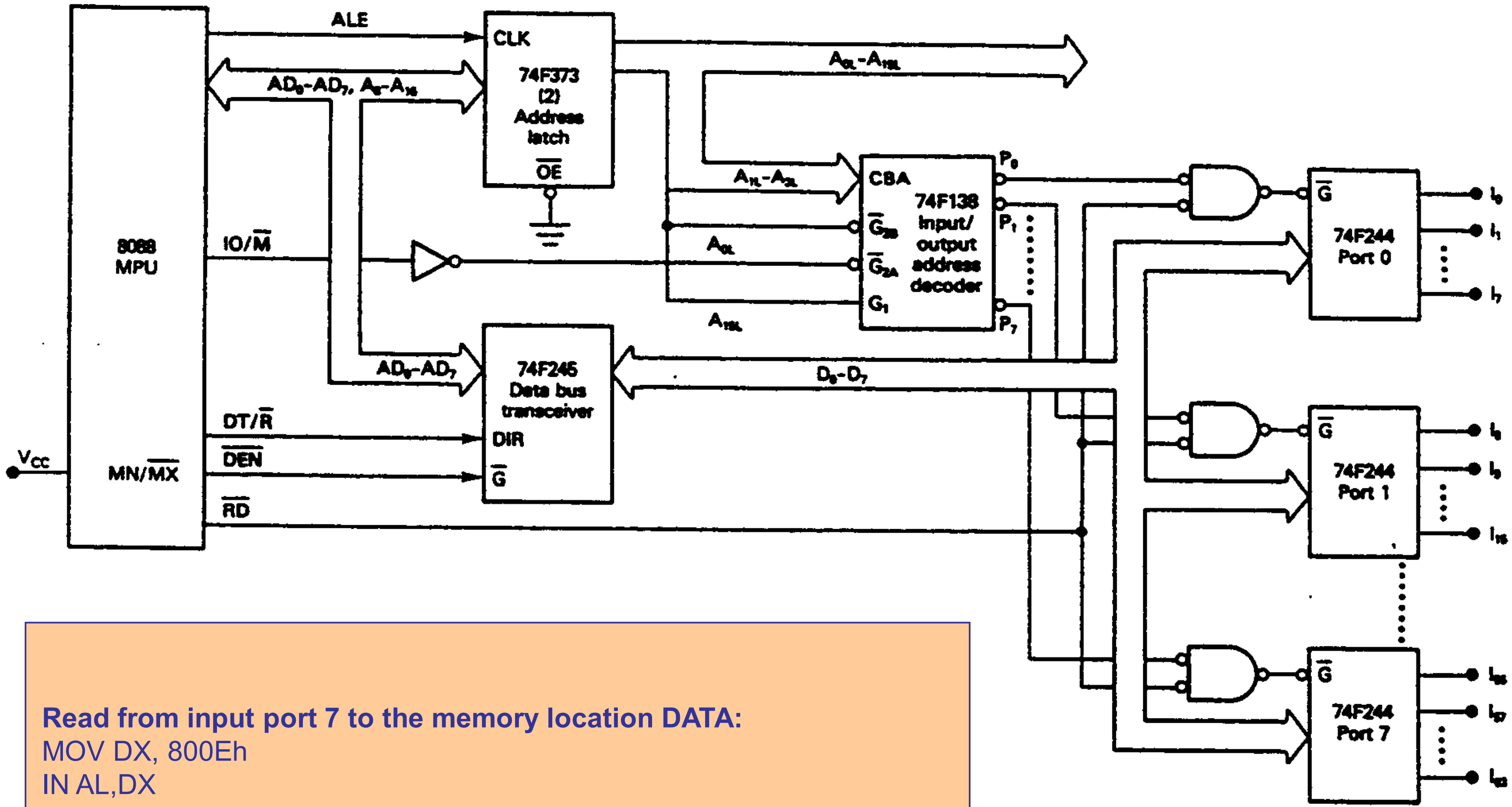
IN port design using the 74LS244

- 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

Interrupts vs Polling

CPU services devices in two ways:

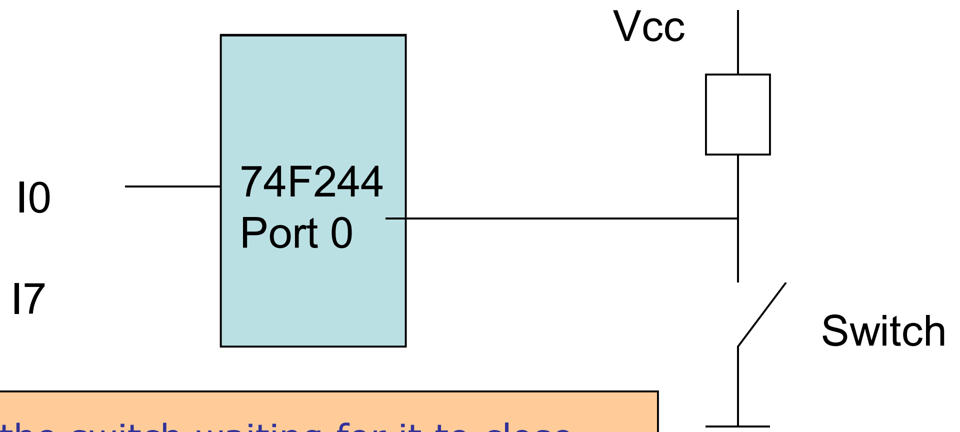
Interrupts and Polling

- 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

- 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

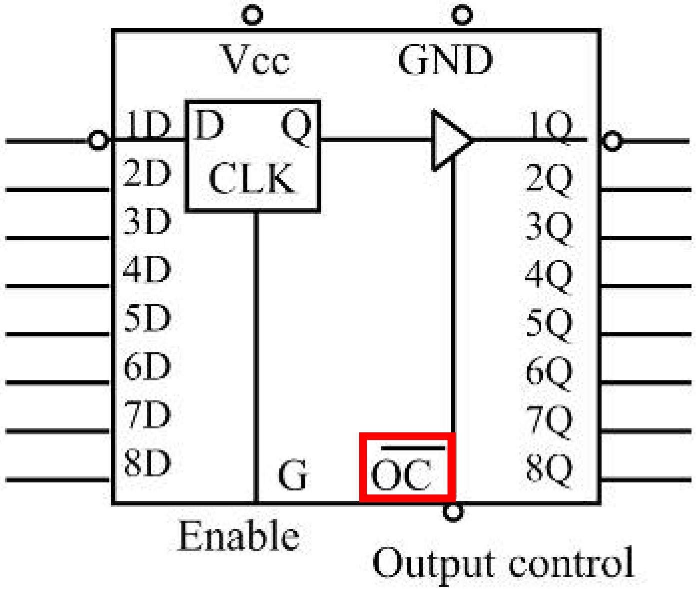
```
        MOV DX,8000h
POLL:   IN AL,DX
        SHL AL,1
        JC POLL
```

11.2: I/O ADDRESS DECODING AND DESIGN

- The concept of address bus decoding for I/O instructions is exactly the same as for memory.
  - 1. The control signals **IOR** and **IOW** are used along with the decoder.
  - 2. For an 8-bit port address, **A0–A7** is decoded.
  - 3. If the port address is 16-bit (using **DX**), **A0–A15** is decoded.

11.2: I/O ADDRESS DECODING AND DESIGN using 74LS373 in an output port design

- 74LS373 can be used as a latching system for simple I/O ports.
  - Pin **OC** must be grounded.



Function Table

Output Control	Enable G	D	Output
L	H	H	H
L	H	L	L
L	L	X	Q0
H	X	X	Z

Figure 11-1 74LS373 D Latch

11.2: I/O ADDRESS DECODING AND DESIGN using 74LS373 in an output port design

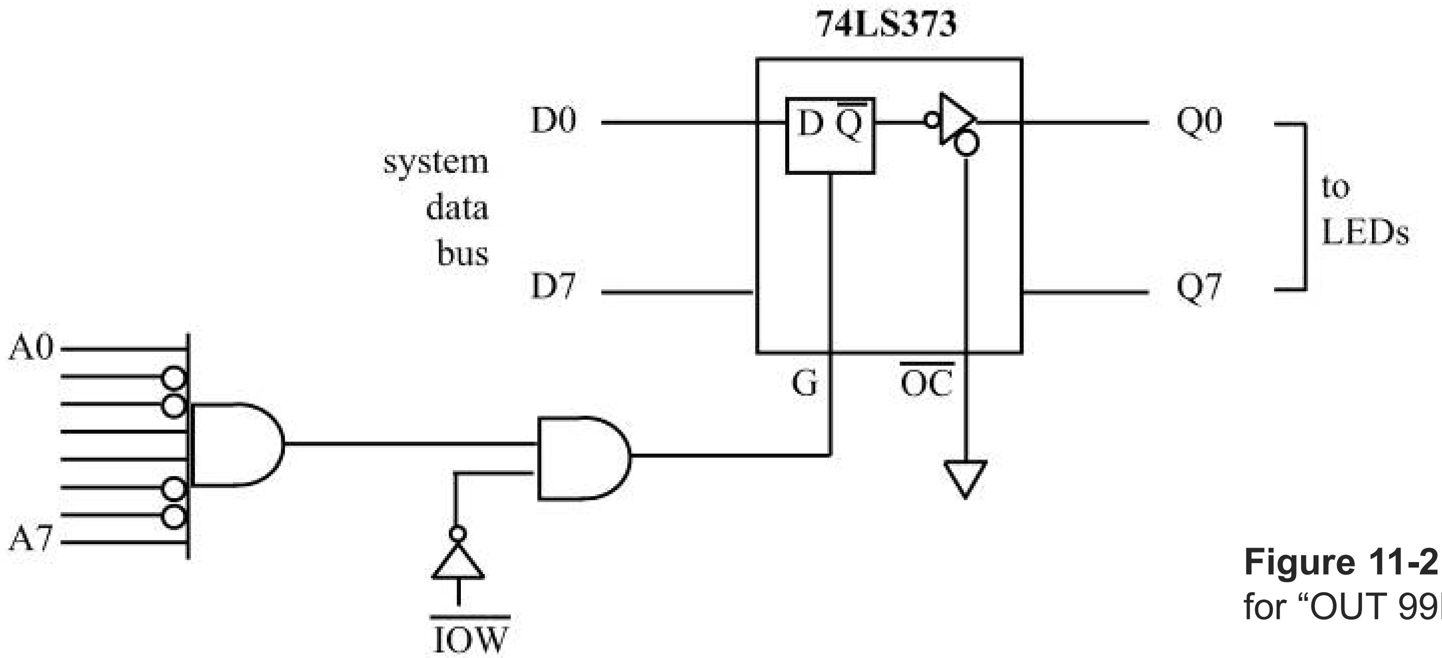


Figure 11-2 Design for “OUT 99H, AL”.

- For an output latch, it is common to AND the output of the address decoder with control signal **IOW**.
  - To provide the latching action.

11.2: I/O ADDRESS DECODING AND DESIGN using 74LS373 in an output port design

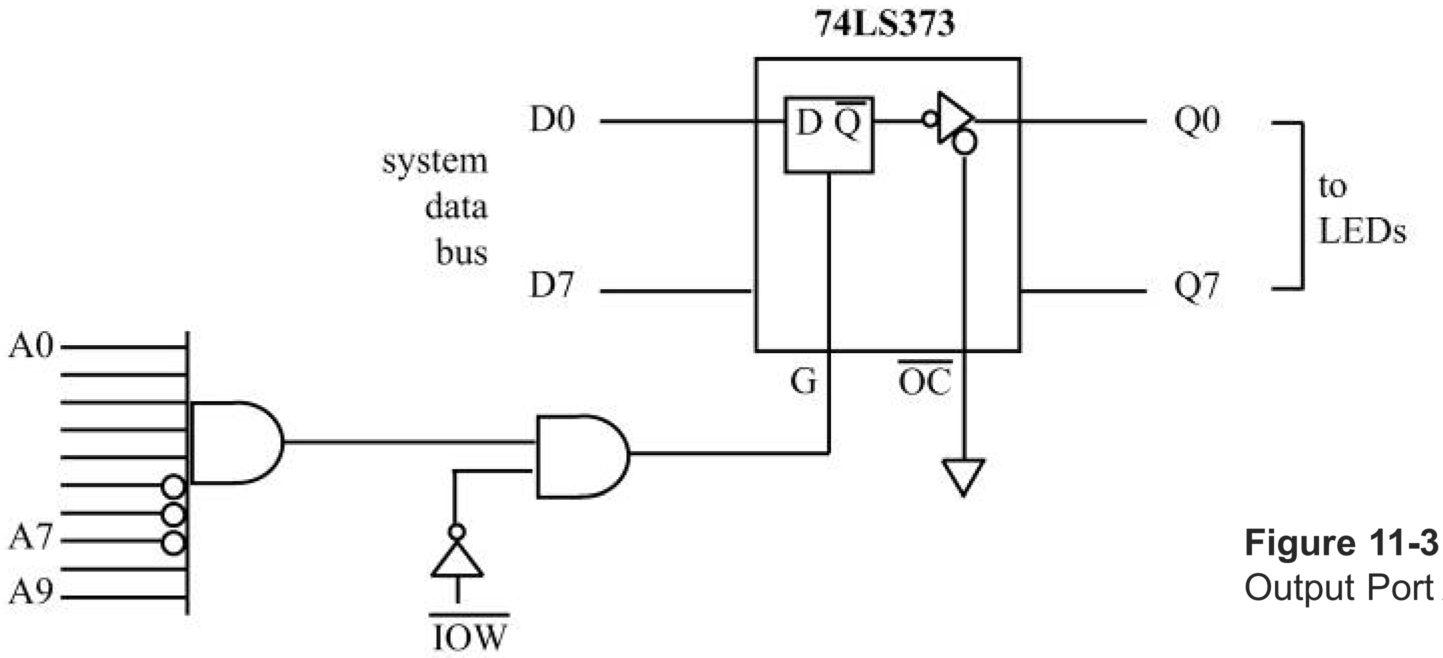


Figure 11-3 Design for Output Port Address 31H.

Example 11-2

Show the design of an output port with an I/O address of 31FH using the 74LS373.

Solution:

31F9H is decoded, then ANDed with IOW to activate the G pin of the 74LS373 latch. This is shown in Figure 11-3.

IN port design using 74LS244

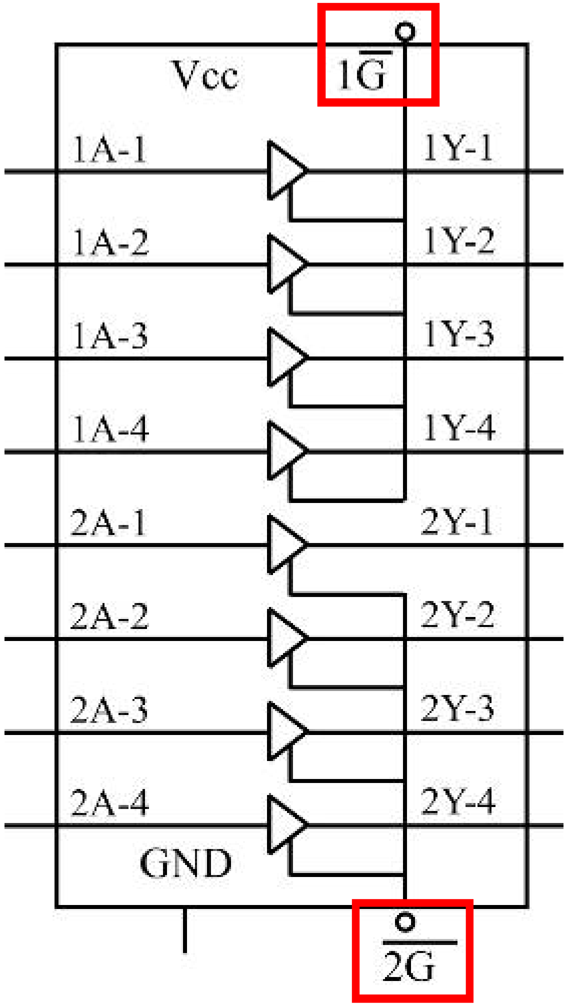


Figure 11-4 74LS244 Octal Buffer

- Data from a data bus, must come in through a three-state buffer—referred to as *tristated*.
  - Simple input ports we use the 74LS244 chip.
- Since **1G** & **2G** each control only 4 bits of 74LS244, they *both* must be activated for 8-bit input.

11.2: I/O ADDRESS DECODING AND DESIGN
IN port design using 74LS244

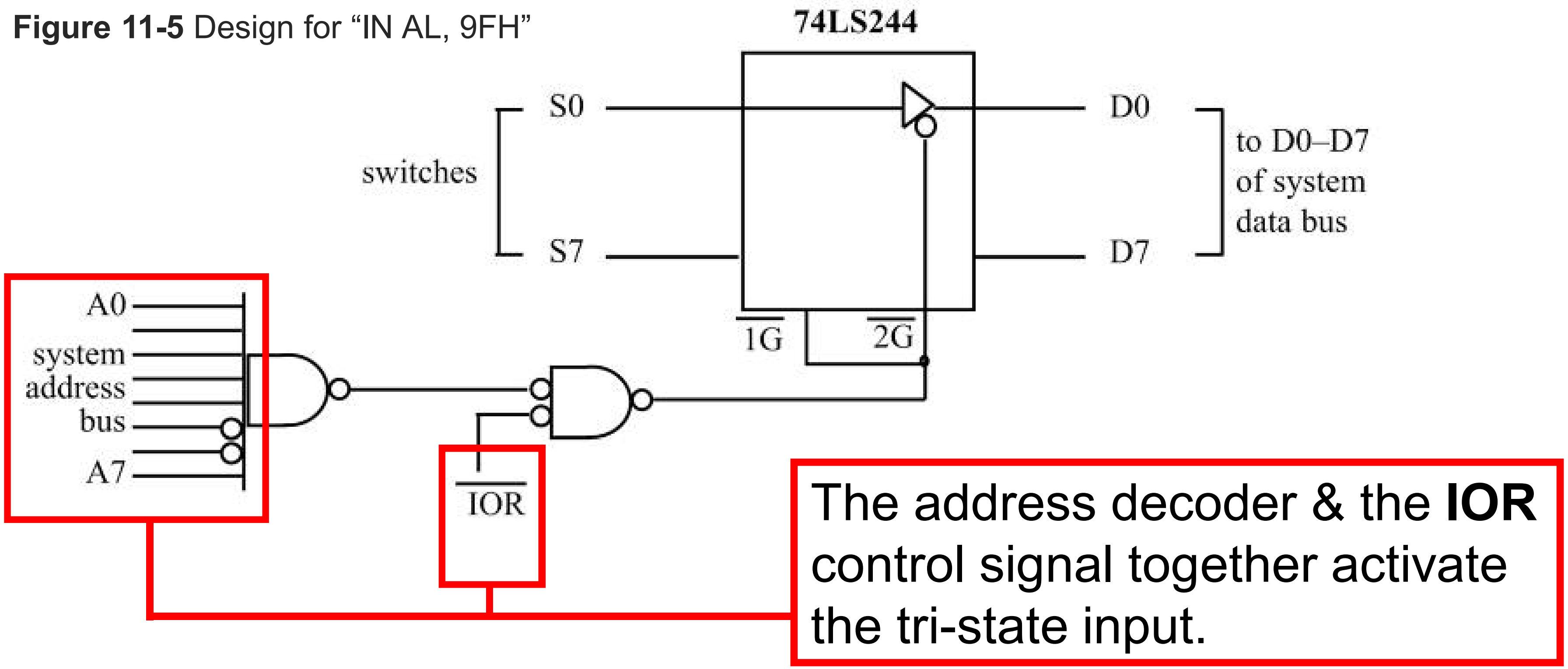
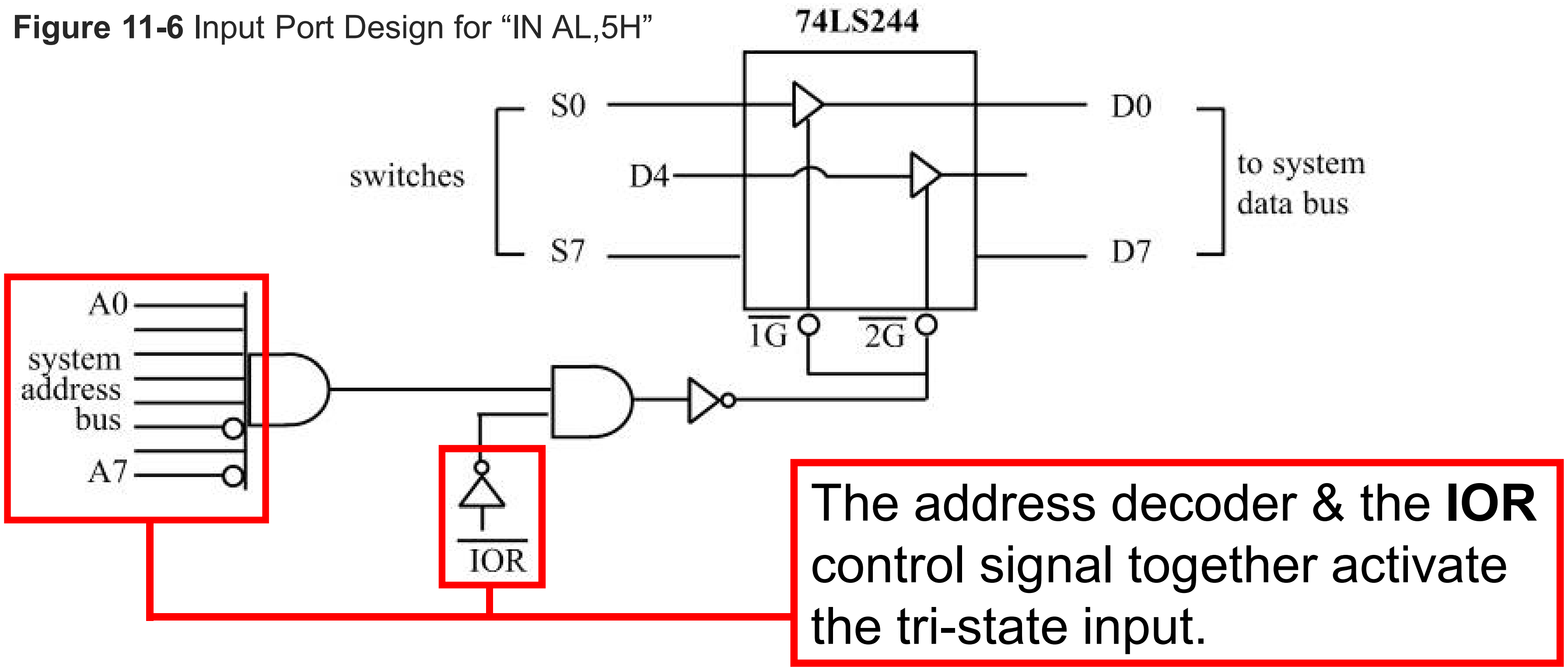


Figure 11-5 Design for "IN AL, 9FH"

- 74LS244 is widely used for buffering and providing high driving capability for unidirectional buses.

11.2: I/O ADDRESS DECODING AND DESIGN
IN port design using 74LS244

Figure 11-6 Input Port Design for "IN AL,5H"



- 74LS244 as an entry port to the system data bus.
  - Used for bidirectional buses, as seen in Chapter 9.

11.2: I/O ADDRESS DECODING AND DESIGN
memory-mapped I/O

- Communicating with I/O devices using IN and OUT instructions is referred to as *peripheral I/O*.
  - Some designers also refer to it as *isolated I/O*.

11.2: I/O ADDRESS DECODING AND DESIGN
memory-mapped I/O

- Some processors do not have IN & OUT instructions, but use *Memory-mapped I/O*.
  - A memory location is assigned as an input or output port.
  - Instructions access memory locations to access I/O ports.
    - Instead of **IN** and **OUT** instructions.
  - The entire 20-bit address, **A0–A19**, must be decoded.
    - The **DS** register must be loaded prior to accessing memory-mapped I/O.
  - In memory-mapped I/O interfacing, control signals **MEMR** and **MEMW** are used.
    - Memory I/O ports can number as high as 2^{20} (1,048,576).

11.2: I/O ADDRESS DECODING AND DESIGN
memory-mapped I/O

- Some processors do not have IN & OUT instructions, but use *Memory-mapped I/O*.
  - Memory-mapped I/O can perform arithmetic & logic operations on I/O data directly without first moving them into the accumulator.
  - Memory-mapped I/O uses memory address space, which could lead to memory space fragmentation.

Example 11-3

Show the design of "IN AL,9FH" using the 74LS244 as a tri-state buffer.

Solution:

9FH is decoded, then ANDed with IOR. To activate OC of the 74LS244, it must be inverted since OC is an active-low pin. This is shown in Figure 11-5.

11.3: I/O ADDRESS MAP OF x86 PCs

Designers of the original IBM PC decided to make full use of I/O instructions. To be compatible with the x86 IBM PC, follow the I/O map of Table 11-1, shown here.

The address range **300–31FH** is set aside for prototype cards to be plugged into the expansion slot.

Hex Range	Device	Hex Range	Device
000–01F	DMA controller 1, 8237A-5	378–37F	Parallel printer port 1
020–03F	Interrupt controller 1, 8259A, Master	380–38F	SDLC, bisynchronous 2
			Cluster
			Bisynchronous 1
			Monochrome display/printer adapter
			Enhanced graphics adapter
			Color graphics monitor adapter
			Disk controller
			Serial port 1
			Data acquisition (adapter 1)
			Cluster (adapter 1)
			Data acquisition (adapter 2)
			Cluster (adapter 2)
			Data acquisition (adapter 3)
			Cluster (adapter 3)
			GPIB (adapter 1)
			Cluster (adapter 4)
2E1	GPIB (adapter 0)	42E1	GPIB (adapter 2)
2E2 & 2E3	Data acquisition (adapter 0)	62E1	GPIB (adapter 3)
2F8–2FF	Serial port 2	82E1	GPIB (adapter 4)
300–31F	Prototype card	A2E1	GPIB (adapter 5)
360–363	PC network (low address)	C2E1	GPIB (adapter 6)
364–367	Reserved	E2E1	GPIB (adapter 7)
368–36B	PC network (high address)		
36C–36F	Reserved		

See the entire I/O map on page 296 of your textbook.

11.3: I/O ADDRESS MAP OF x86 PCs absolute vs. linear address decoding

- In decoding addresses, either all or a selected number of them are decoded.
  - In *absolute* decoding, all address lines are decoded.
  - If only selected address pins are decoded, it is called *linear select* decoding.
- Linear select is cheaper, but creates aliases, the same port with multiple addresses.
  - If you see a large gap in the I/O address map of the x86 PC, it is due to the address aliases of the original PC.

11.3: I/O ADDRESS MAP OF x86 PCs
prototype addresses 300–31FH in x86 PC

- Prototype cards at **300H–31FH** can be data acquisition boards used to monitor analog signals.
  - Temperature, pressure, etc., inputs use signals on the 62-pin section of the ISA expansion slot.

IOR and **IOW**. Both *active-low*.

When **AEN** = 0, the CPU is using the bus.

A0–A9 for address decoding.

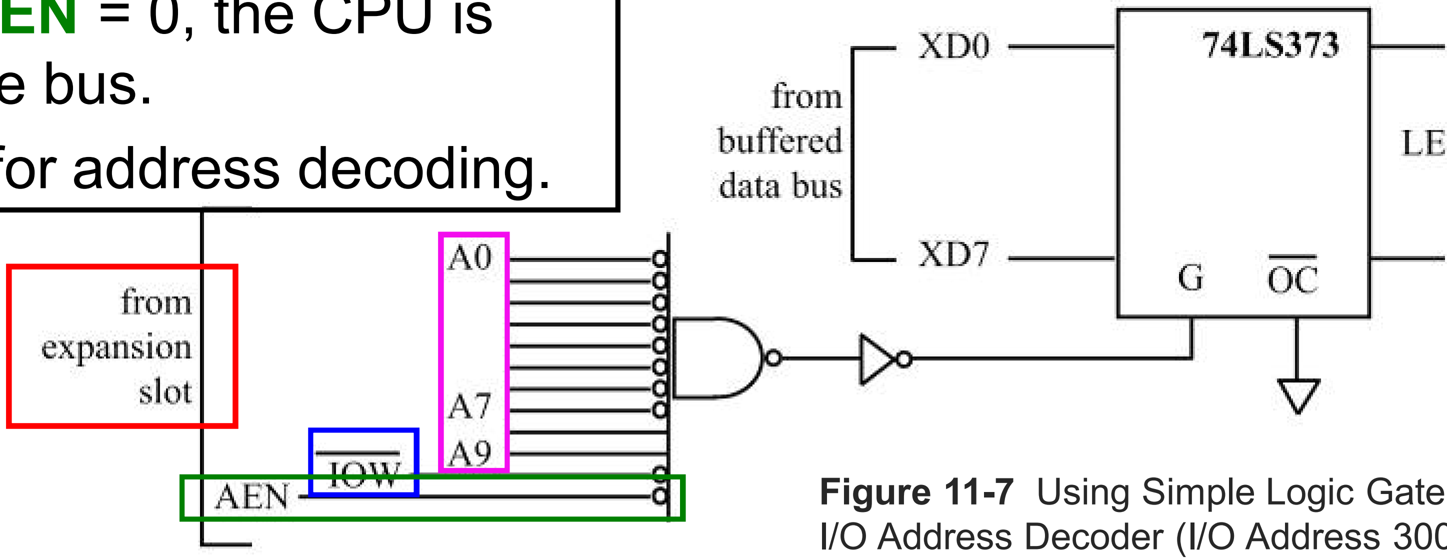
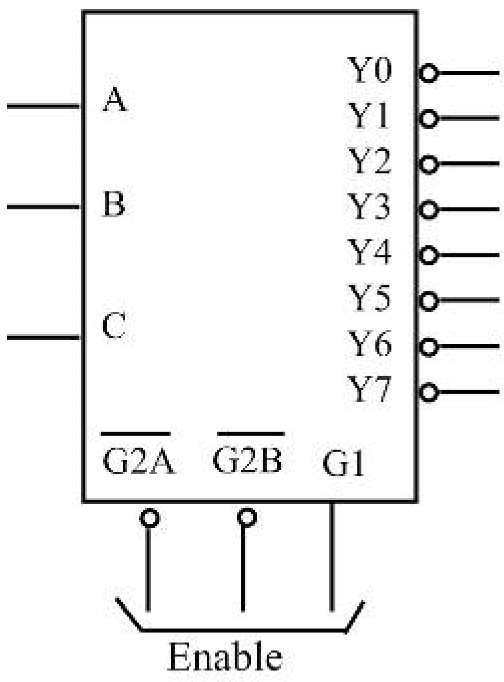


Figure 11-7 Using Simple Logic Gate for I/O Address Decoder (I/O Address 300H)

74LS138 as a decoder

- NANDs, inverters, and 74LS138 chips for decoders can be applied to I/O address decoding.

Block Diagram



Function Table

Inputs: Enable G1	Enable G2	Select C	Select B	Select A	Outputs Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

Figure 11-8 74S138 Decoder

11.3: I/O ADDRESS MAP OF x86 PCs
74LS138 as a decoder

- 74LS138 showing I/O address decoding for an input port located at address **304H**.
  - Each Y output controls a single I/O device.
  - **Y4** output, together with the signal at **IOR**, controls the 74LS244 input buffer.

Y0, with IOW can control a 74LS373 latch.

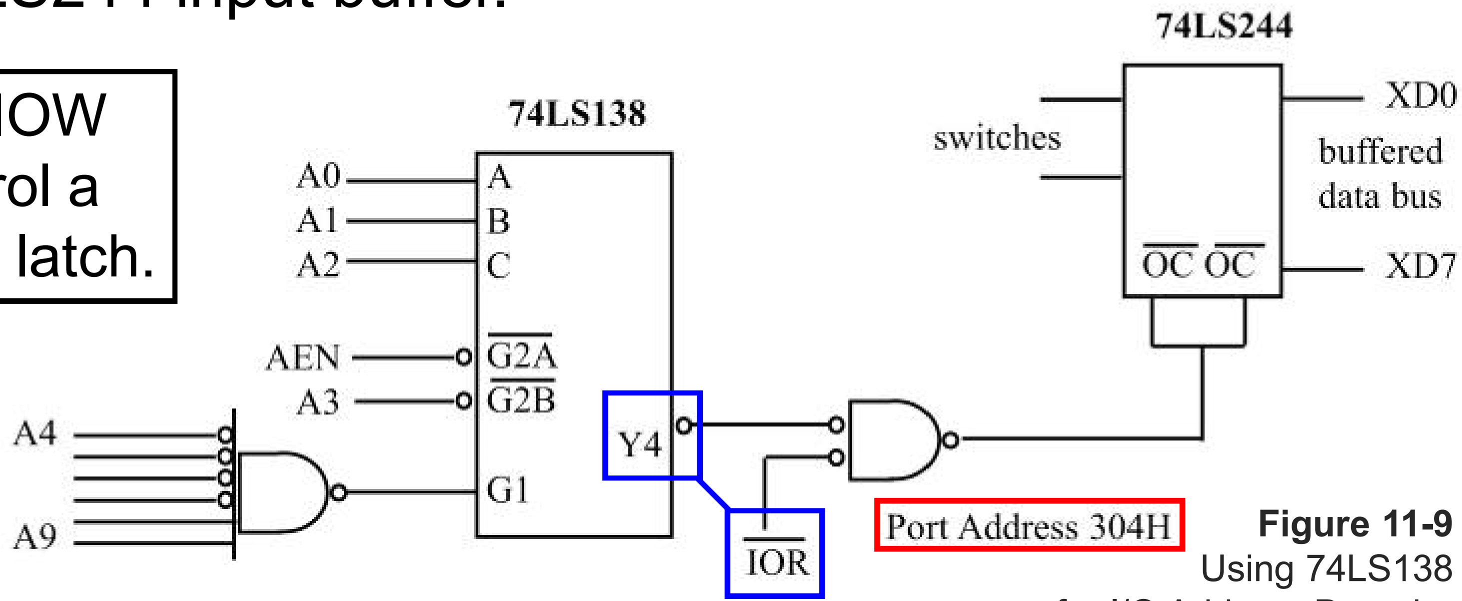
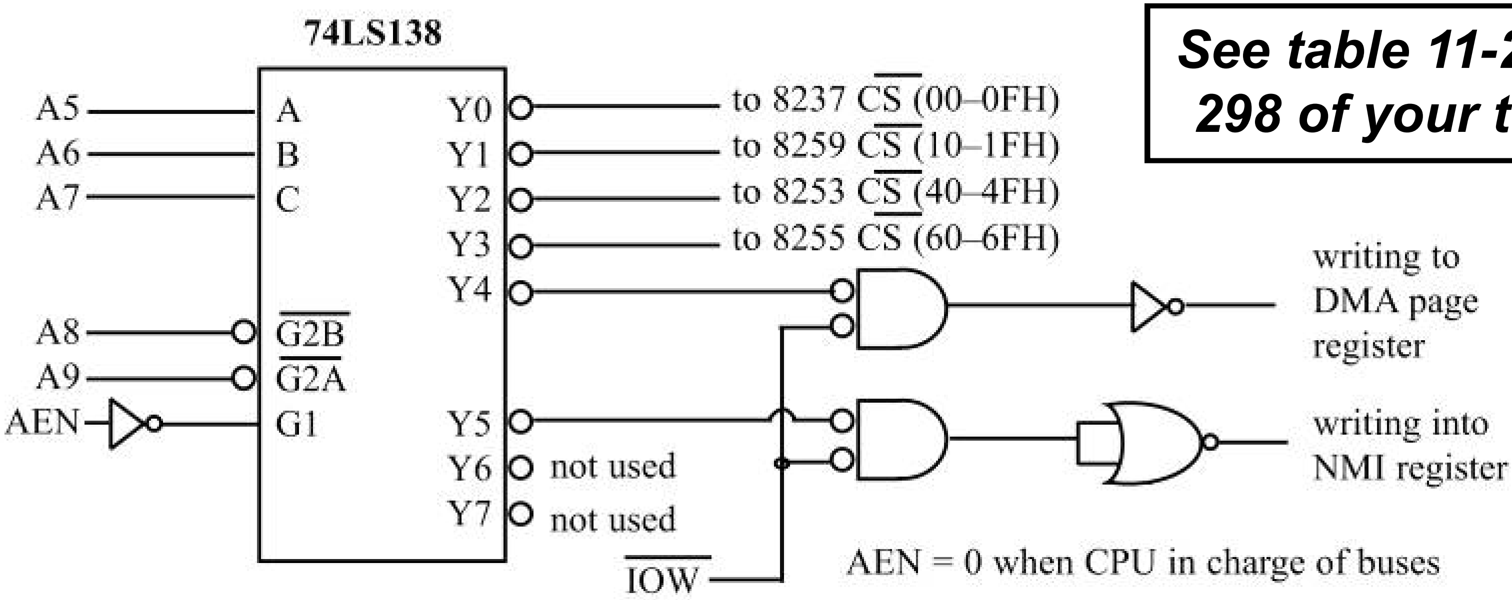


Figure 11-9
Using 74LS138
for I/O Address Decoder

11.3: I/O ADDRESS MAP OF x86 PCs
74LS138 as IBM PC I/O address decoder

- A0 to A4 go to individual peripheral input addresses.
- A5, A6, & A7 handle output selection of outputs Y0 to Y7.
- Pins A8, A9, & AEN all must be *low* to enable 74LS138.
  - AEN is low only when the x86 is in control of the system bus.



See table 11-2 on page 298 of your textbook.

Figure 11-10 PC/XT Port Address Coding

11.3: I/O ADDRESS MAP OF x86 PCs
port 61H and time delay generation

- Port 61H, a widely used port, can be used to generate a time delay.
  - In *any* PC from the 286 to the Pentium®.
- I/O port 61H has eight bits (D0–D7), of which D4 is of particular interest.
  - In all 286 & higher PCs, D4 of port 61H changes its state, indefinitely every 15.085 microseconds (ms).
    - Low for 15.085 ms.
    - High for the same amount of time.
    - Low again.

11.3: I/O ADDRESS MAP OF x86 PCs
port 61H and time delay generation

- The following program uses port 61H to generate a 1/2 second delay in all bits of port 310H.

```
;TOGGLING ALL BITS OF PORT 310H EVERY 0.5 SEC
            MOV   DX,310H
HERE:       MOV   AL,55H   ;toggle all bits
            OUT   DX,AL
            MOV   CX,33144 ;delay=33144x15.085 us=0.5 sec
            CALL  TDELAY
            MOV   AL,0AAH
            OUT   DX,AL
            MOV   CX,33144
            CALL  TDELAY
            JMP   HERE
```

See the entire program listing on page 299 of your textbook.

(MORE)

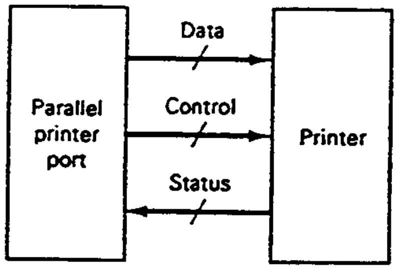
11.3: I/O ADDRESS MAP OF x86 PCs
port 61H and time delay generation

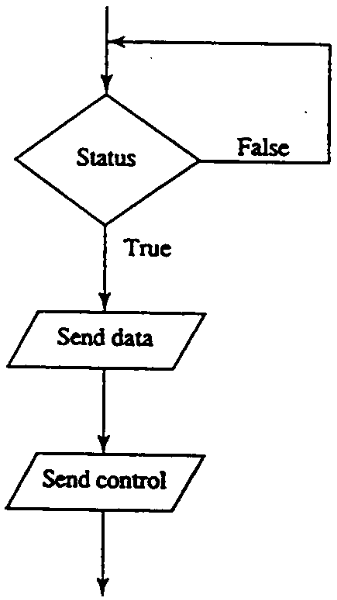
- When 61H is read, all bits are masked except D4.
  - The program waits for D4 to change, before it loops again.

```
;CX=COUNT OF 15.085 MICROSEC
TDELAY      PROC   NEAR
            PUSH   AX          ;save AX
W1:         IN     AL,61H
            AND    AL,00010000B
            CMP    AL,AH
            JE     W1          ;wait for 15.085 usec
            MOV    AH,AL
            LOOP   W1          ;another 15.085 usec
            POP    AX          ;restore AX
            RET
TDELAY      ENDP
```

See the entire program listing on page 299 of your textbook.

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

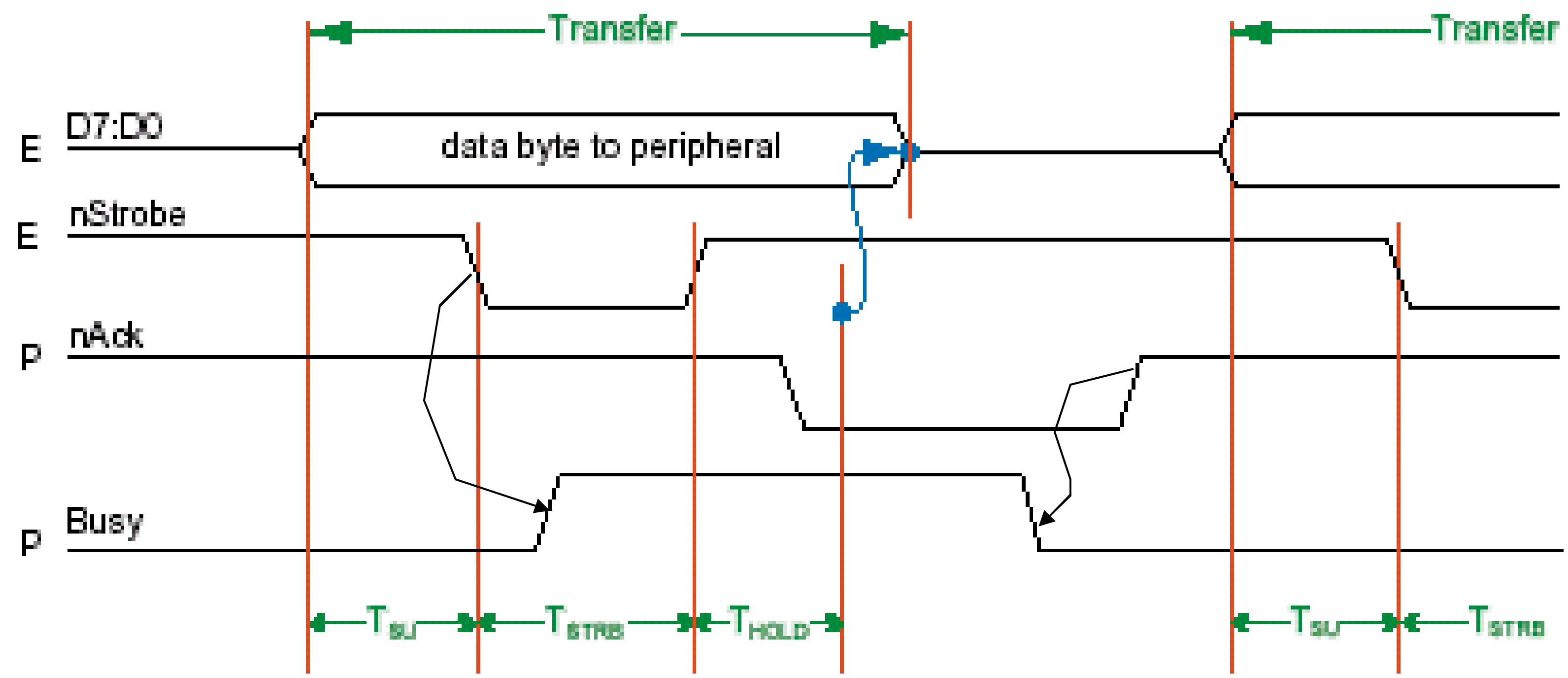
STROBE when PC presents a character

INITIALIZE Clear Printer Buffer and reset control

Operational Principle - Parallel Printer Port

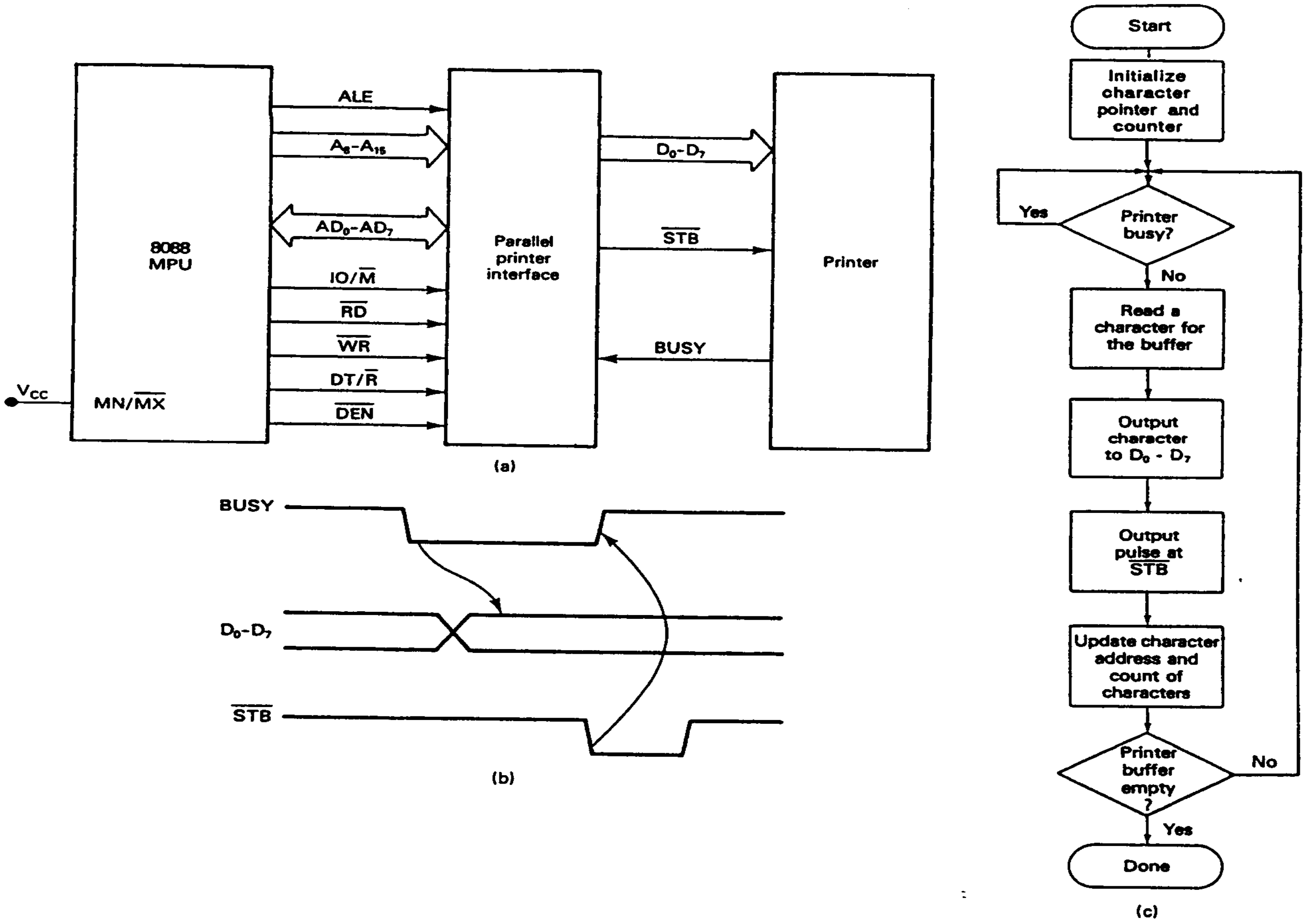
- The computer checks the BUSY signal from the printer, if not BUSY then
- When the PC presents a character to the data pins of the printer, it activates the STROBE pin, telling it that there is a byte sitting at the data pins. Prior to asserting STROBE pin, the data must be at at the printer's data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

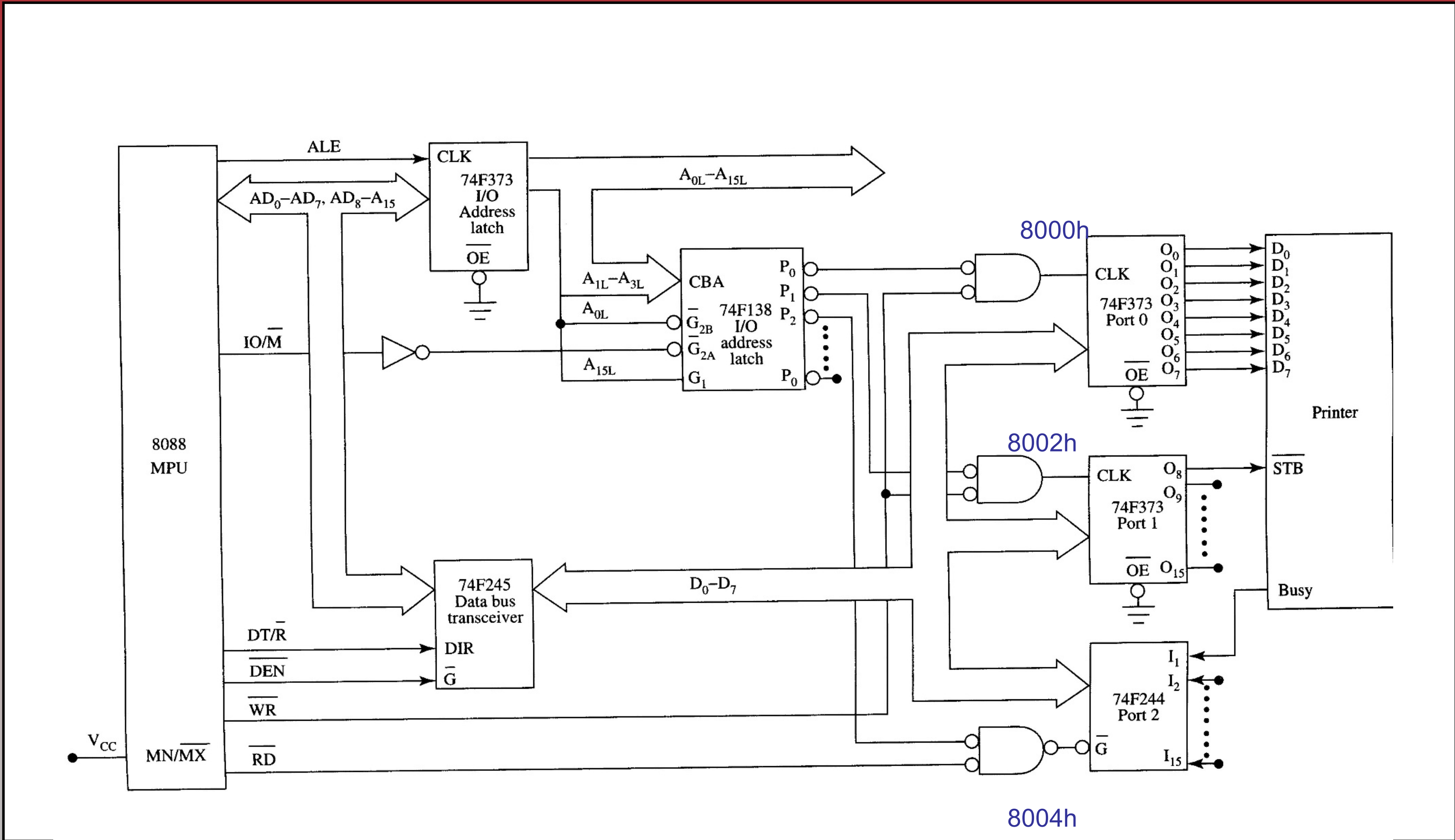


Transfer
Transfer
E
D7:D0
data byte to peripheral
E
nStrobe
P
nAck
P
Busy
T_{SU}
T_{STRB}
T_{HOLD}
T_{SU}
T_{STRB}

Handshaking



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
```

11.4: PROGRAMMING & INTERFACING THE 8255

- The 8255 is a widely used 40-pin, DIP I/O chip.
  - It has three separately accessible programmed ports, A, B & C.
  - Each port can be programmed to be input or output.
  - Ports can also be changed dynamically.

Port A (PA0–PA7)
Port B (PB0–PB7)
Port C (PC0–PC7)

← These 8-bit ports can be all input *or* all output.

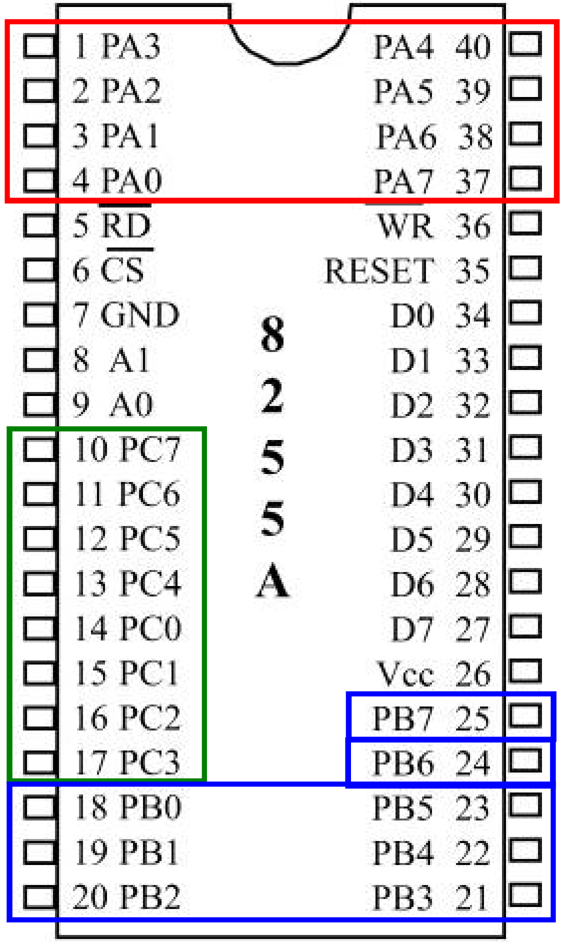


Figure 11-11 8255 PPI Chip

11.4: PROGRAMMING & INTERFACING THE 8255

- **RD** and **WR** - *active-low* 8255 control signal inputs.
  - If the 8255 is using peripheral I/O, **IOR** & **IOW** of the system bus are connected to these two pins.
  - If memory-mapped I/O, **MEMR** & **MEMW** activate them.
- **RESET** - an active-high signal input into the 8255, used to clear the control register.
  - All ports are initialized as input ports.

- **A0**, **A1**, and **CS**
  - **CS** (chip select) selects the entire chip.
  - Address pins **A0** and **A1** select the *specific port* within the 8255.

These three pins are used to access ports A, B, C, or the control register.

CS	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 is not selected

The control register must be programmed to select the operation mode of the three ports A, B, and C.

11.4: PROGRAMMING & INTERFACING THE 8255
mode selection of the 8255A

- 8255 ports can be programmed in various modes.
  - The *simple I/O mode*, Mode 0, is most widely used.

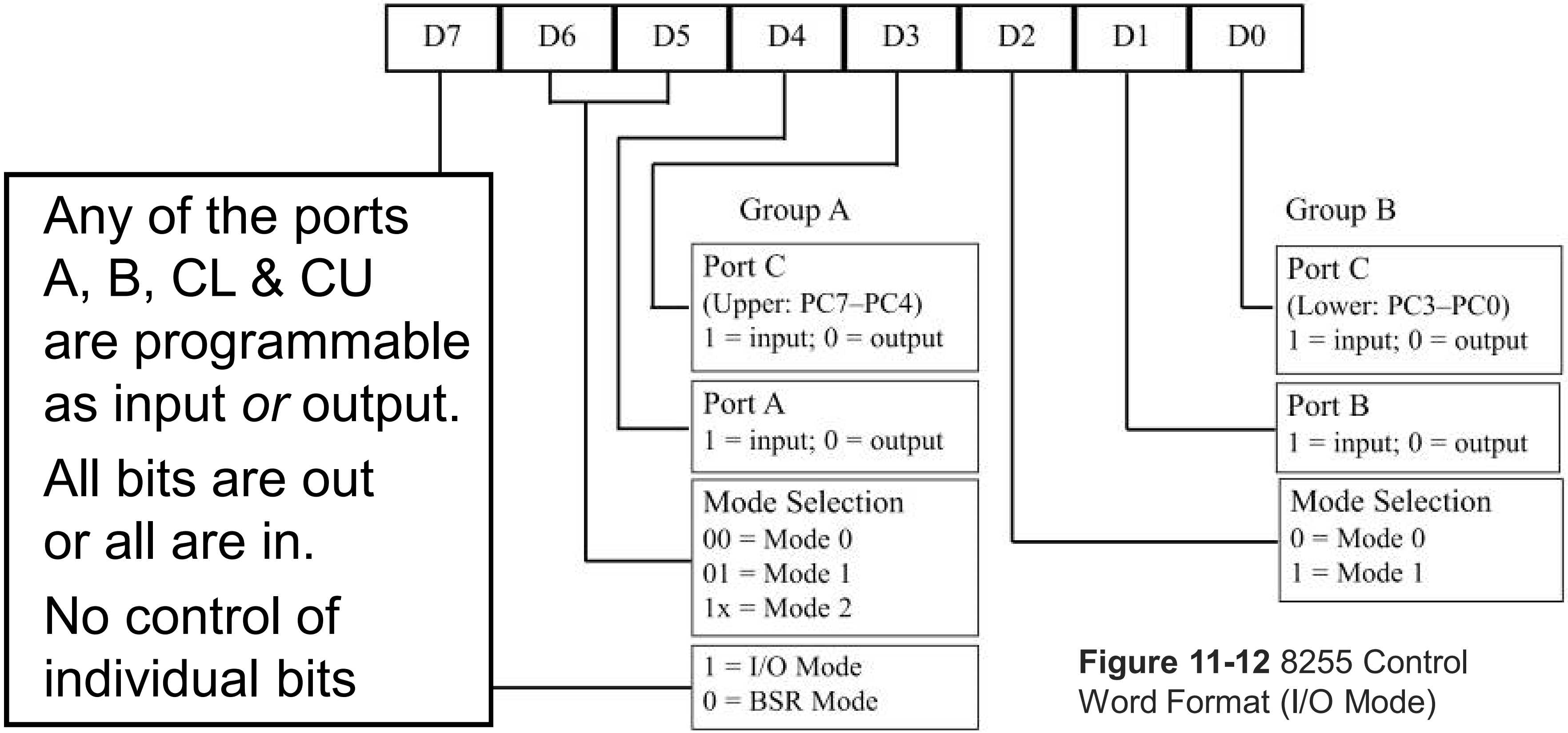


Figure 11-12 8255 Control Word Format (I/O Mode)

11.4: PROGRAMMING & INTERFACING THE 8255
mode selection of the 8255A

- In simple mode, any of the ports A, B, CL, and CU can be programmed as input or output.
  - All bits are out or all are in.
  - No control of individual bits

8255 Control Word Format

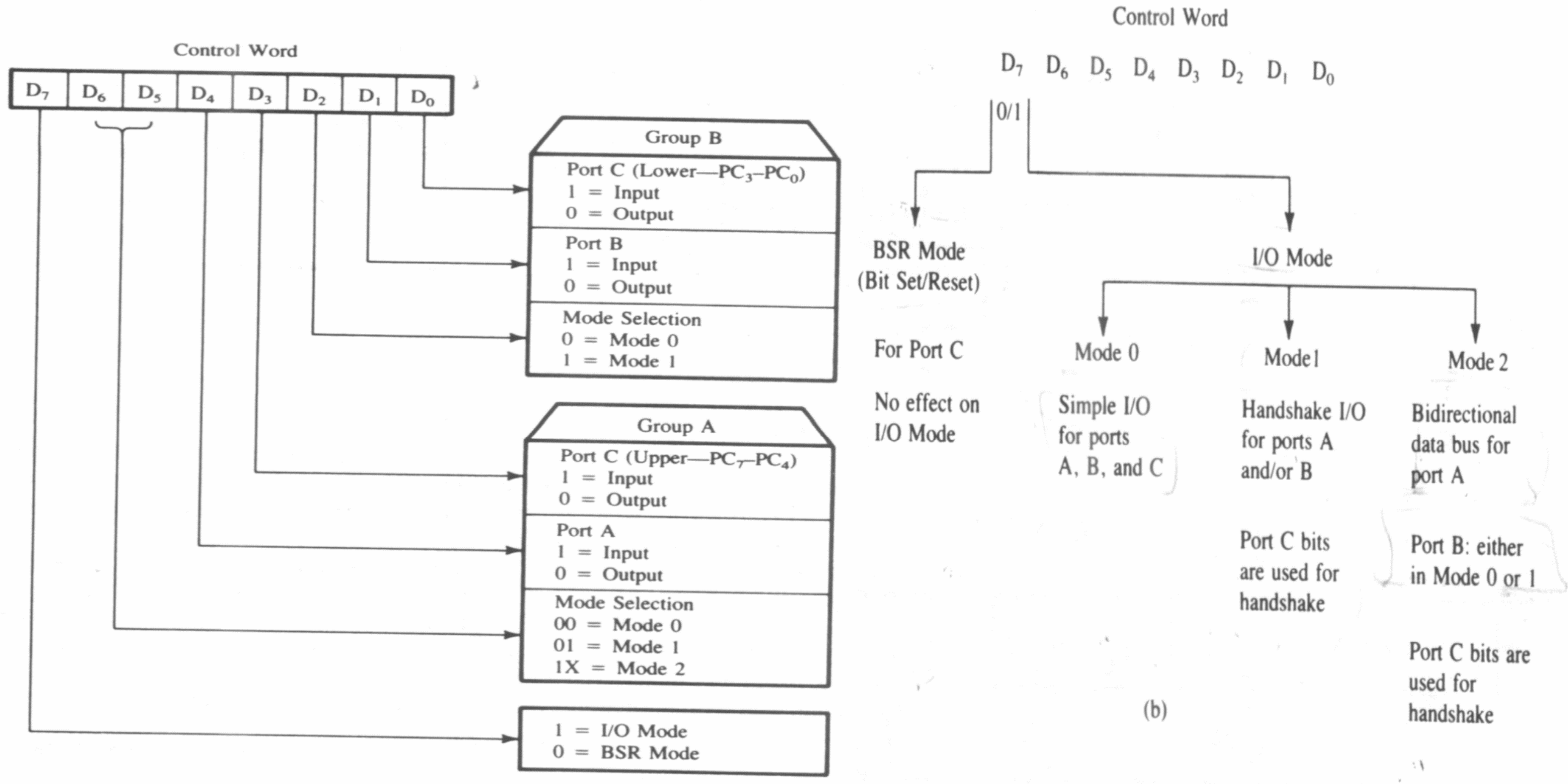
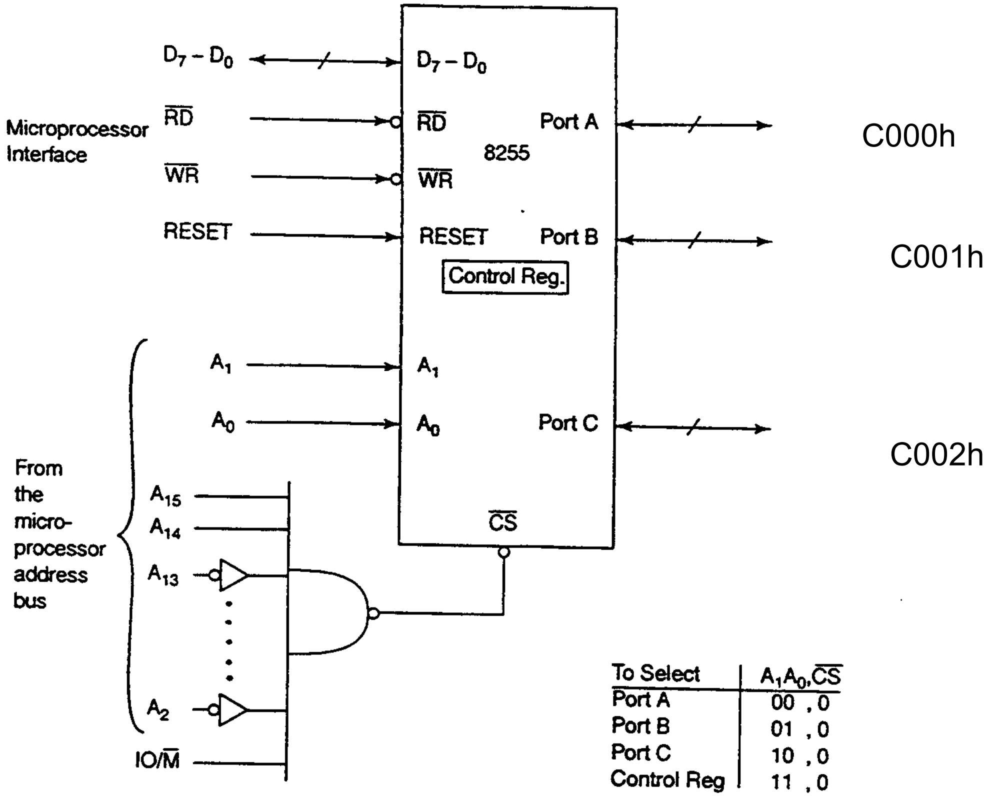


FIGURE 15.4
8255A Control Word Format for I/O Mode
SOURCE: Adapted from Intel Corporation, *Peripheral Components* (Santa Clara, Calif.: Author, 1993), p. 3–

Addressing an 8255



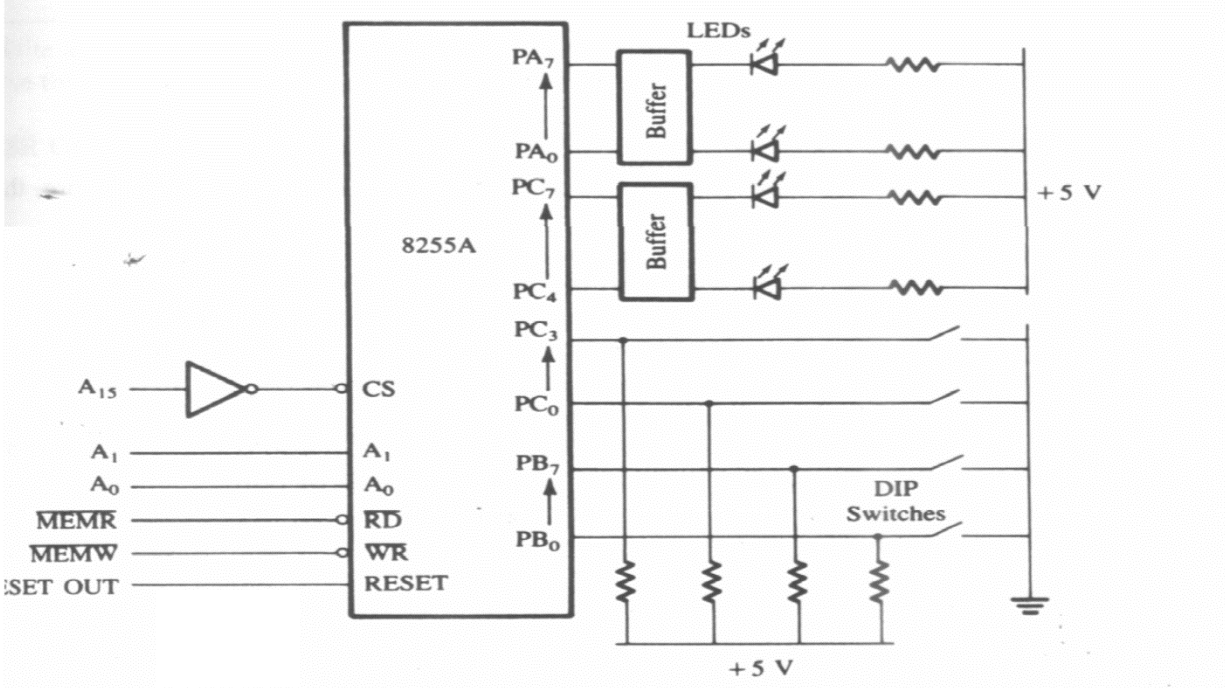
To Select	$A_1A_0,\overline{CS}$
Port A	00 , 0
Port B	01 , 0
Port C	10 , 0
Control Reg	11 , 0

Mode 0 - Simple input/output

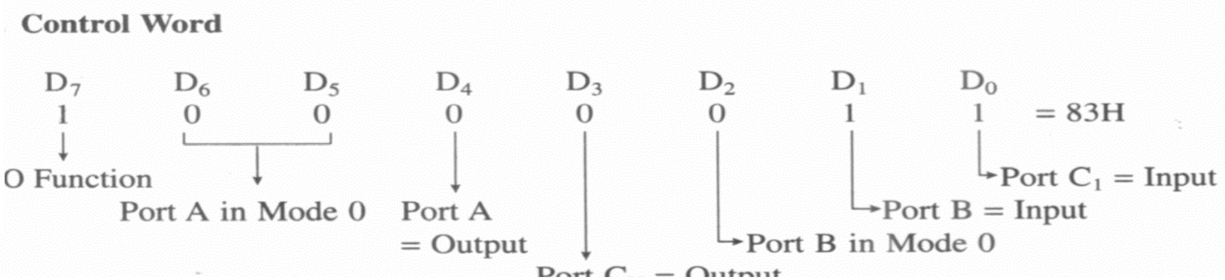
- Simple I/O mode: any of the ports A, B, CL, and CU can be programmed as input or output.
- Example: Configure port A as input, B as output, and all the bits of port C as output assuming a base address of 50h
- Control word should be 1001 0000b = 90h

```
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



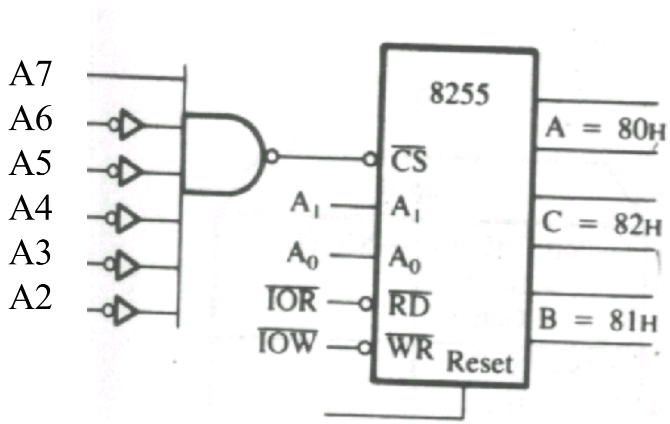
Initialize this device with the appropriate Control Word. Read from PORT C_L and display at PORT A.

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



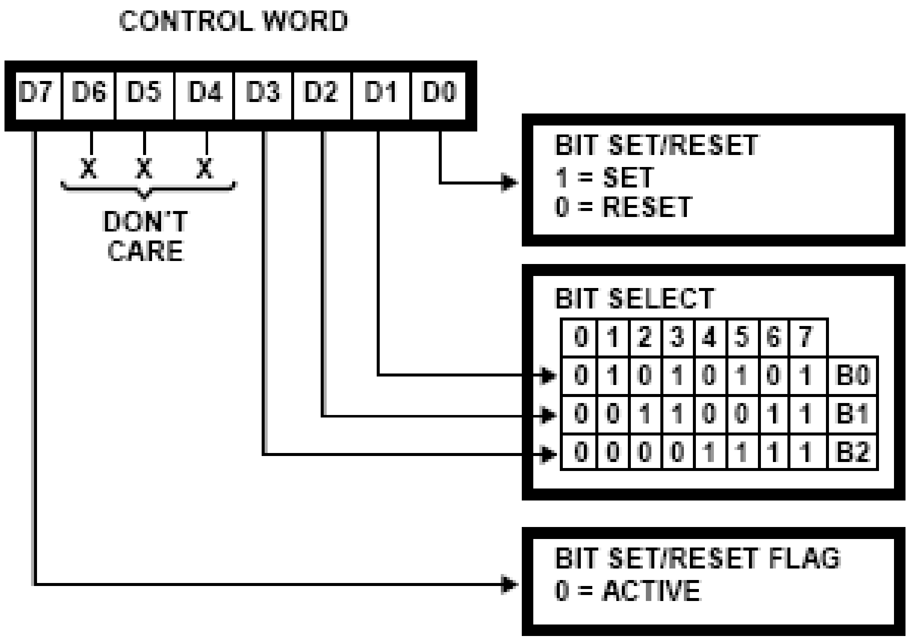


FIGURE 5. BIT SET/RESET FORMAT

- 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
```

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

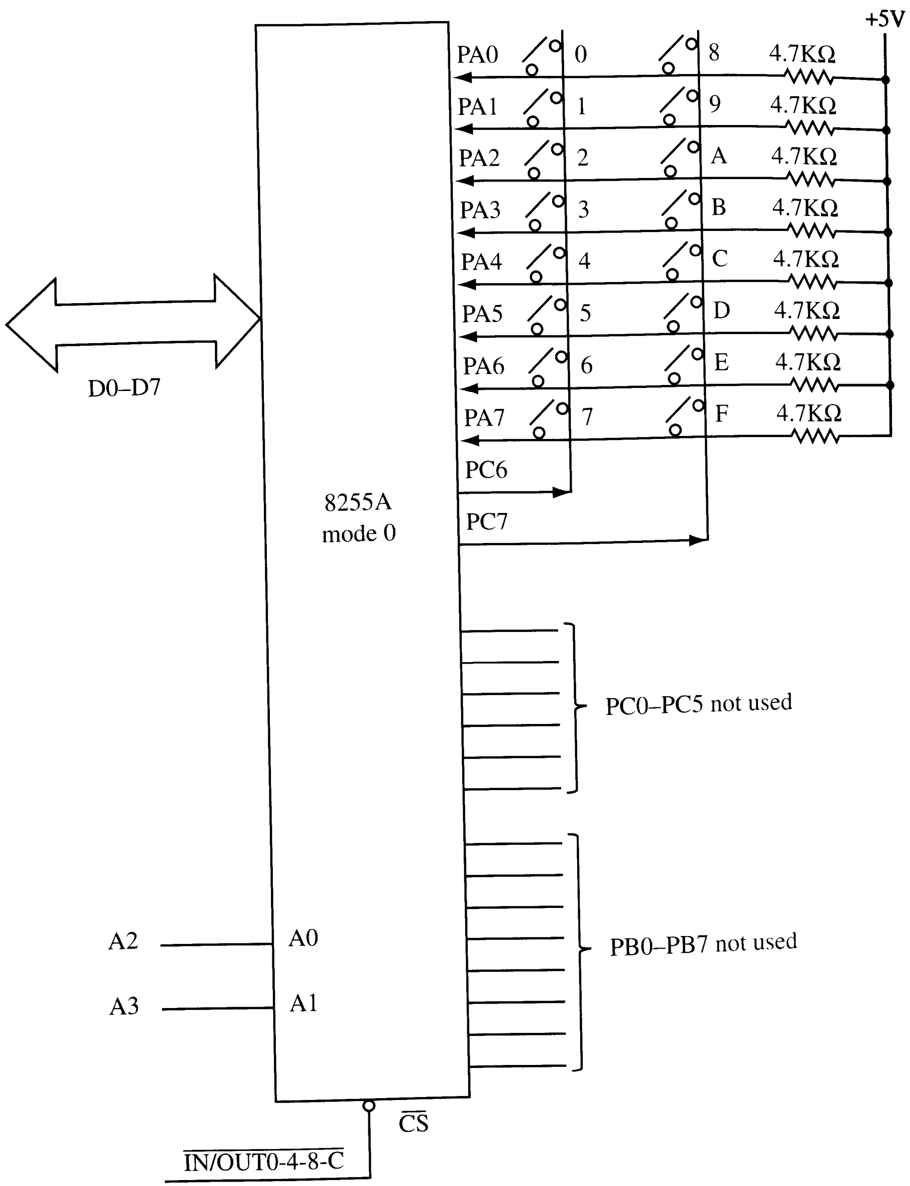
- Assume 8255 has a base address 300h
- Write a program to toggle all bits of port A continuously with a ¼ sec. Delay
- Use int 16h to exit if there is a key press

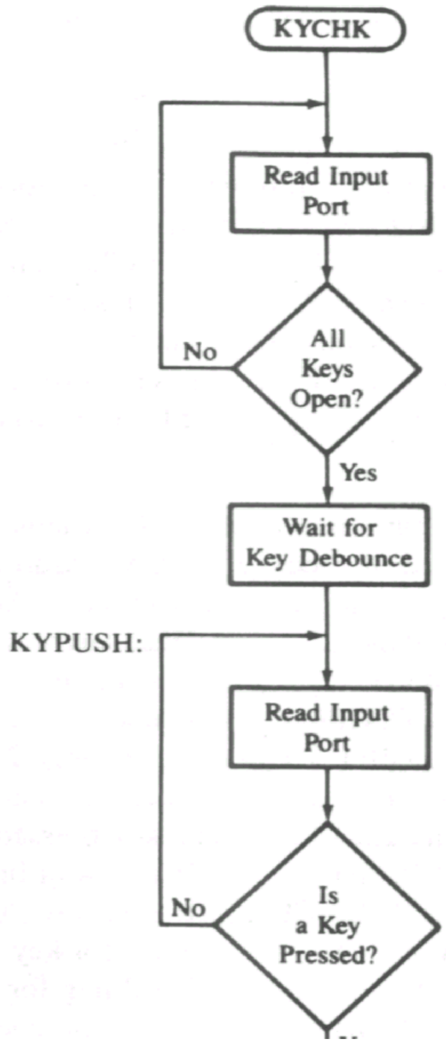
```
        MOV   DX,303h
        MOV AL,80h
        OUT DX,AL
AGAIN:  MOV DX,300h
        MOV AL,55h
        OUT DX,AL
        CALL QSDELAY
        MOV AL,0AAh
        OUT DX,AL
```

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

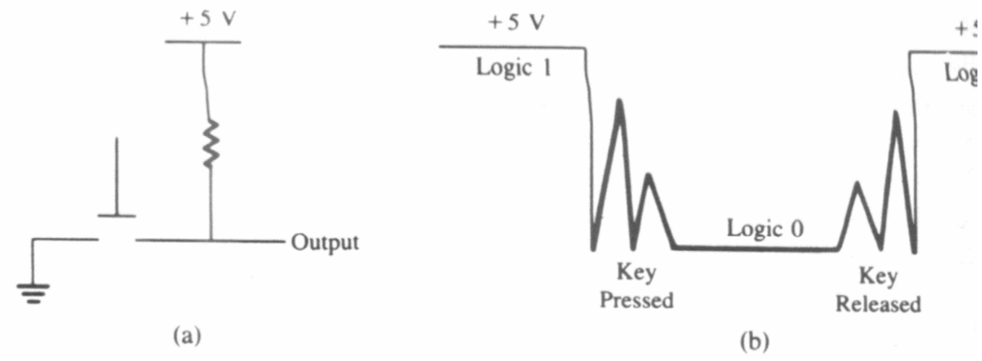


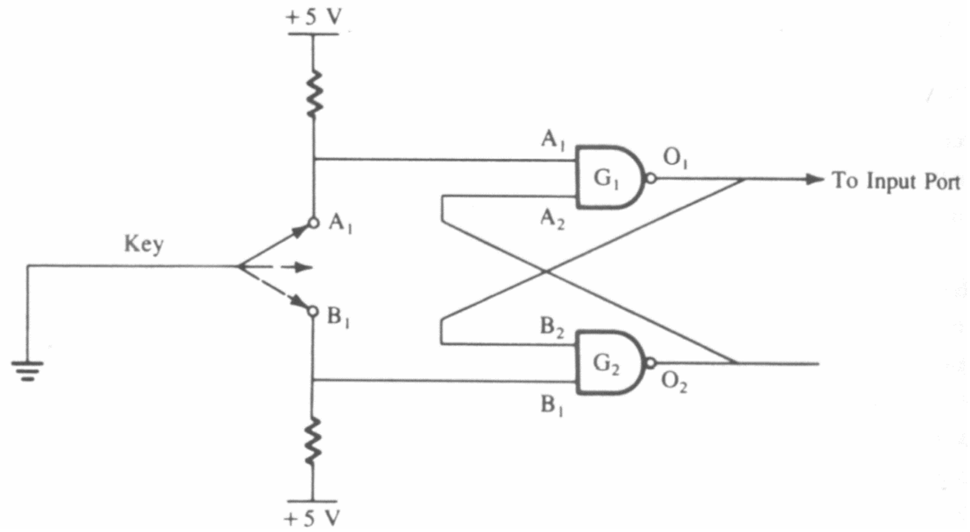


Key Debounce

✓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.





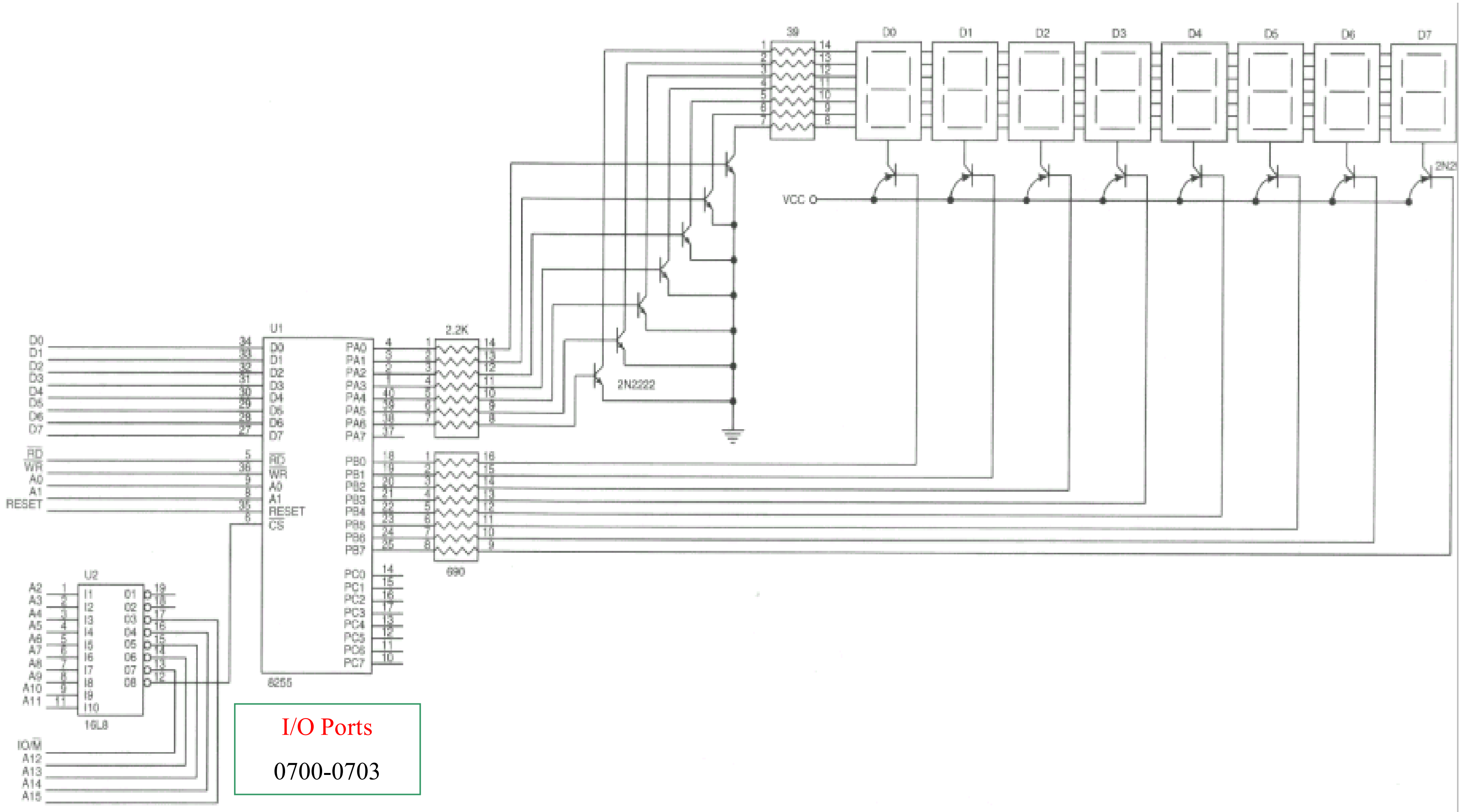
Key debounce technique in hardware

```
                          ;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



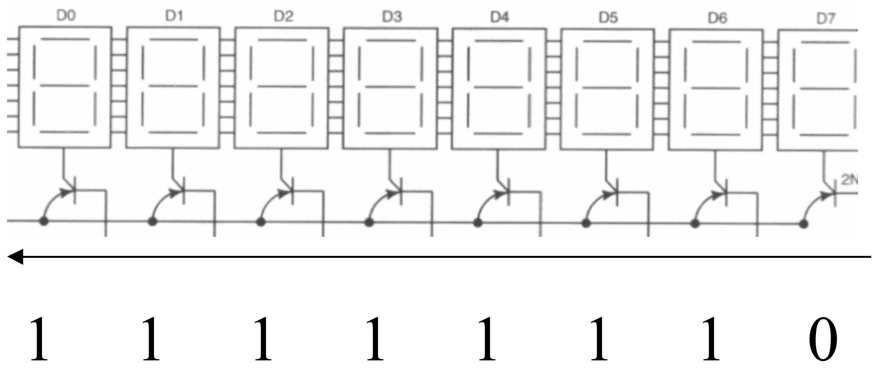
I/O Ports

0700-0703

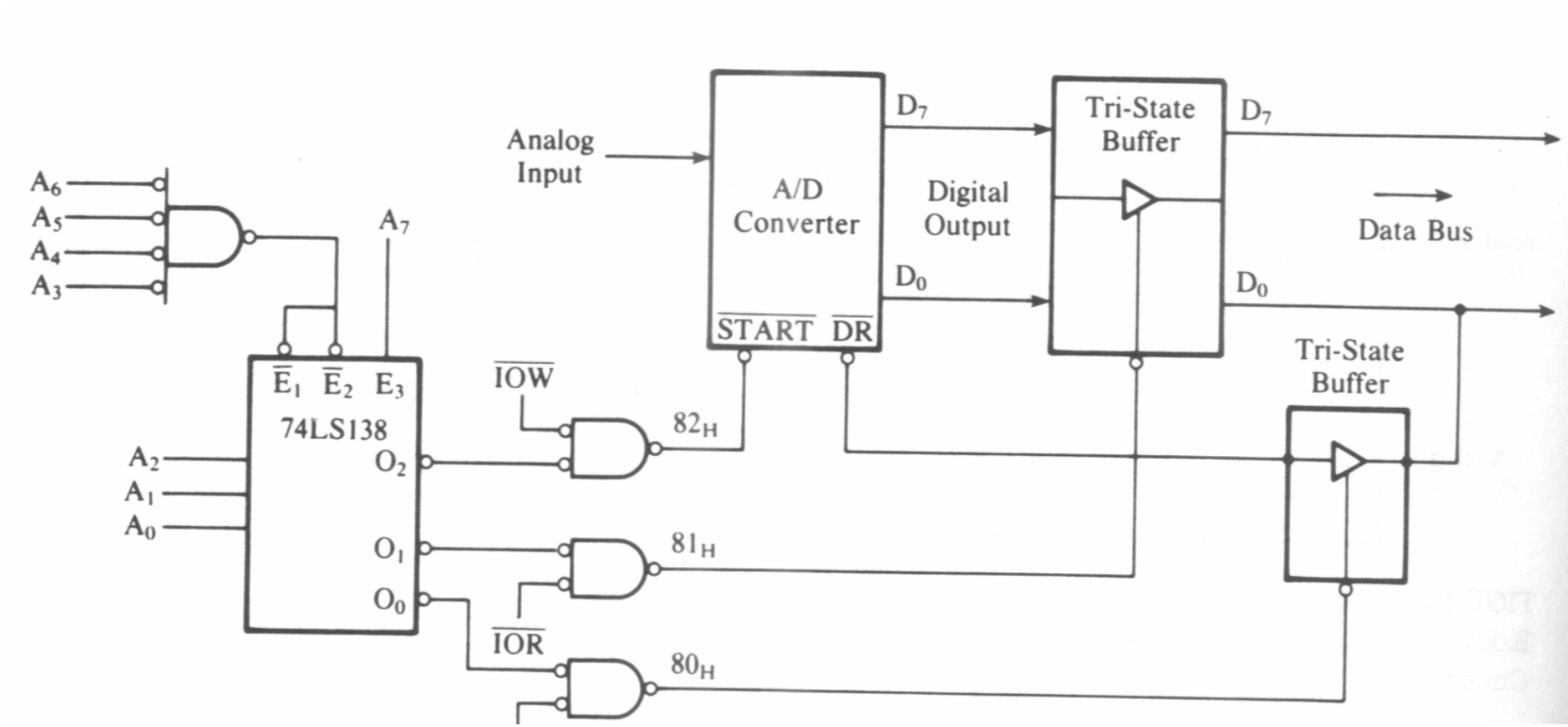
FIGURE 11–20 An 8-digit LED display interfaced to the 8088 microprocessor through an 82C55 PIA.

8 Digit LED

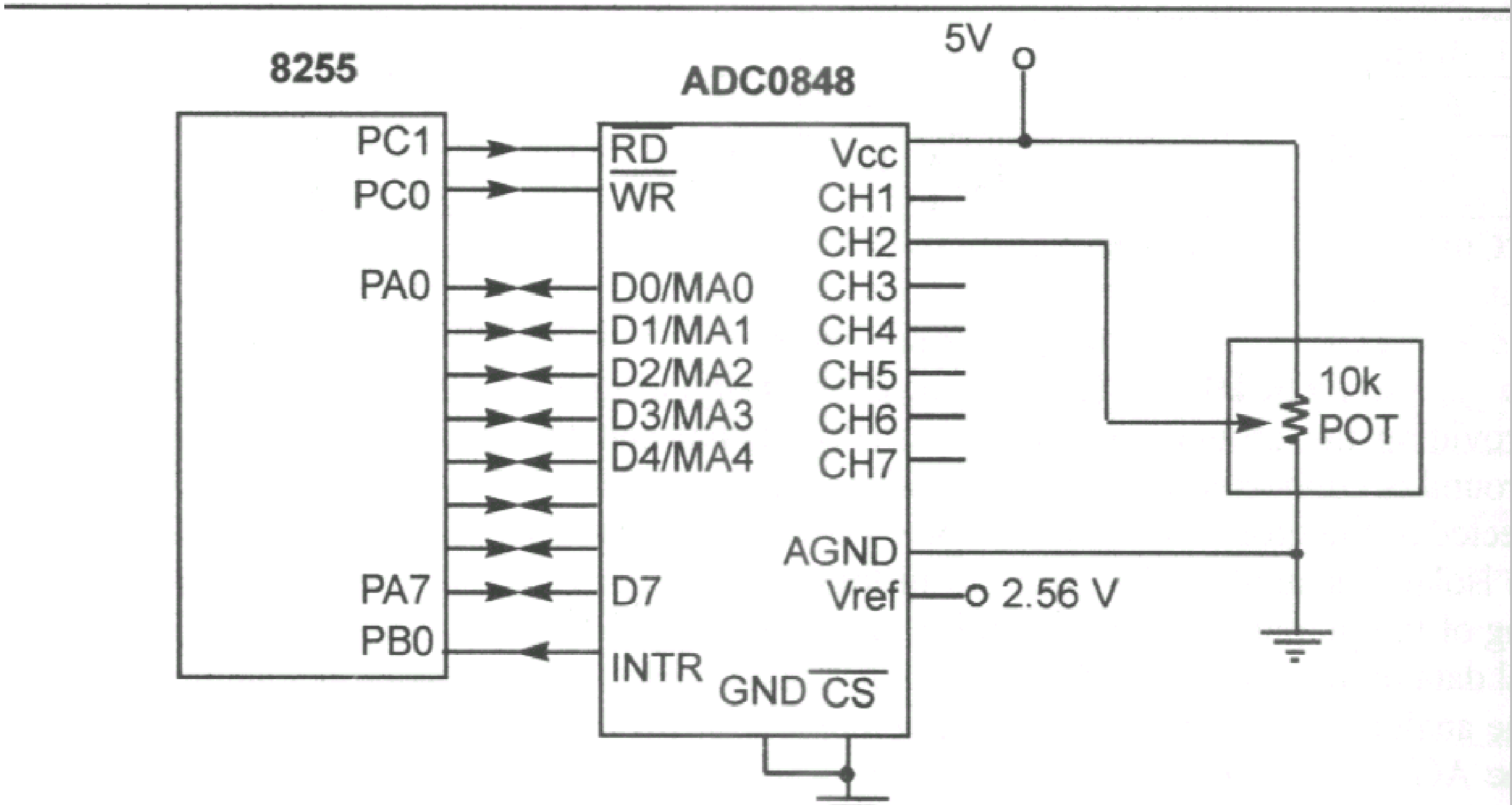
```
; INIT 8255
MOV AL,10000000B
MOV DX,703H
OUT DX,AL

;SETUP REGISTERS TO DISPLAY
MOV BX,7
MOV AH,7FH
MOV SI,OFFSET MEM
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
```



Using A/D with status check (polling)



Connecting the 804 A/D



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
```

Dec Hex Bin

11 B 00001011

ENDS ; I/O

The x86 PC

assembly language, design, and interfacing

fifth edition

MUHAMMAD ALI MAZIDI
JANICE GILLISPIE MAZIDI
DANNY CAUSEY

The x86 PC
assembly language, design, and interfacing
fifth edition

Prentice Hall