```
Dec   Hex   Bin
11    B     00001011
```

ORG ; ELEVEN

8255 I/O PROGRAMMING

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

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 the 8255 programmable peripheral interface chip.

OBJECTIVES (cont)

this chapter enables the student to:

- Code Assembly language instructions to perform I/O through the 8255.
- Code I/O programming for Microsoft Visual C/C++.
- Code I/O programming for Linux C/C++.

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

11.1: 8088 INPUT/OUTPUT INSTRUCTIONS
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*:
  - For more ports, use 16-bit port address instruction.

```
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.

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

Example 11-1 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:

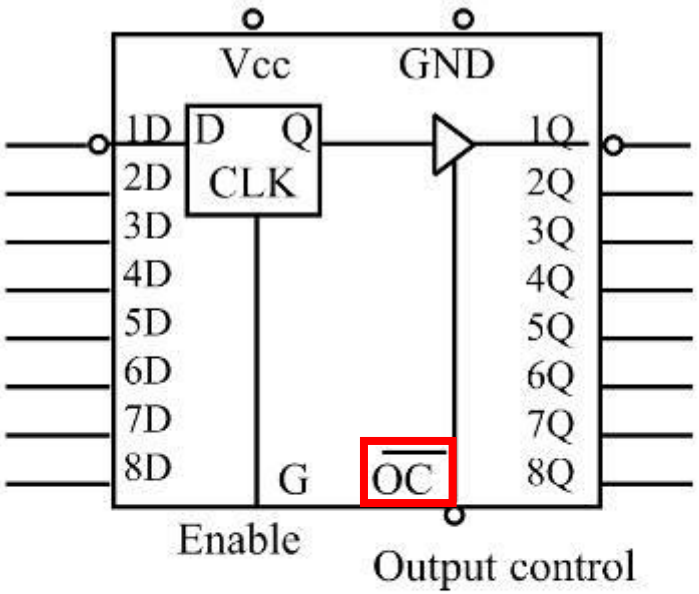
```
BACK:   IN    AL,22H   ;get the temperature from port # 22H
        CMP   AL,100       ;is temp = 100?
        JNZ   BACK         ;if not, keep monitoring
        MOV   BH,'Y'       ;temp = 100, load 'Y' into BH
```

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

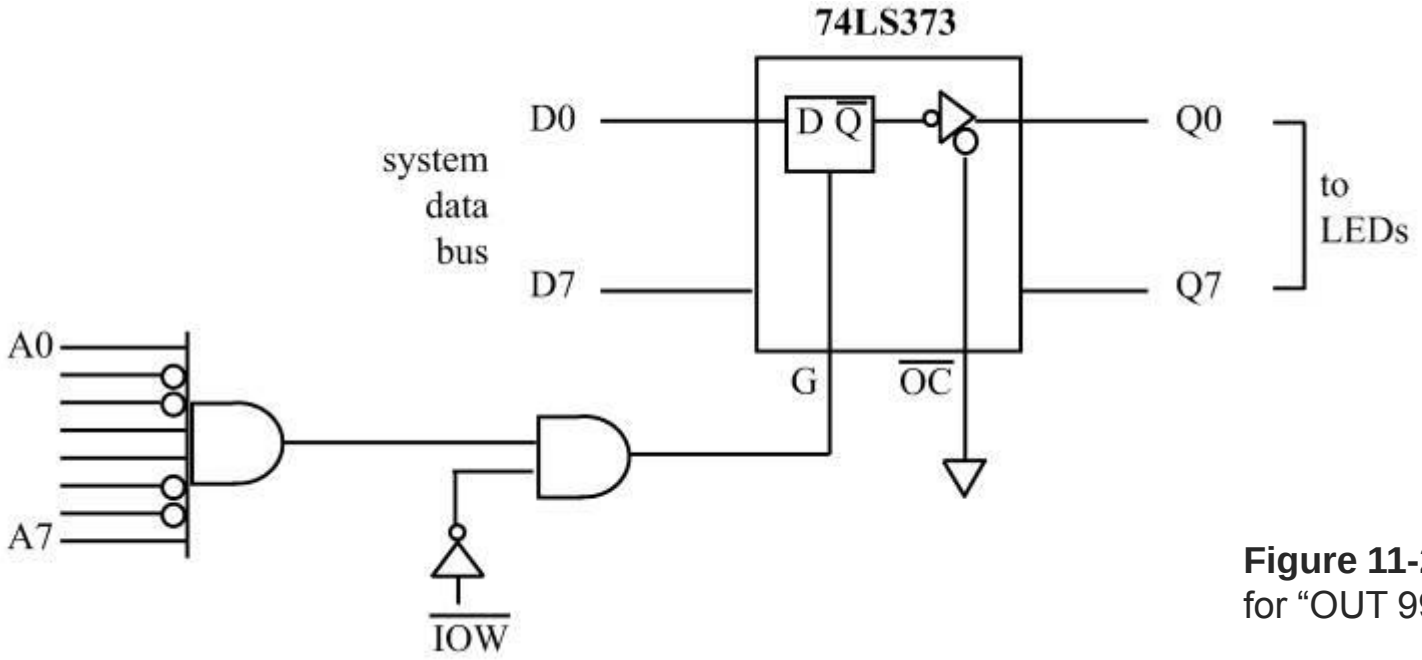


Figure 11-2 Design for "OUT 99H, AL".

- 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

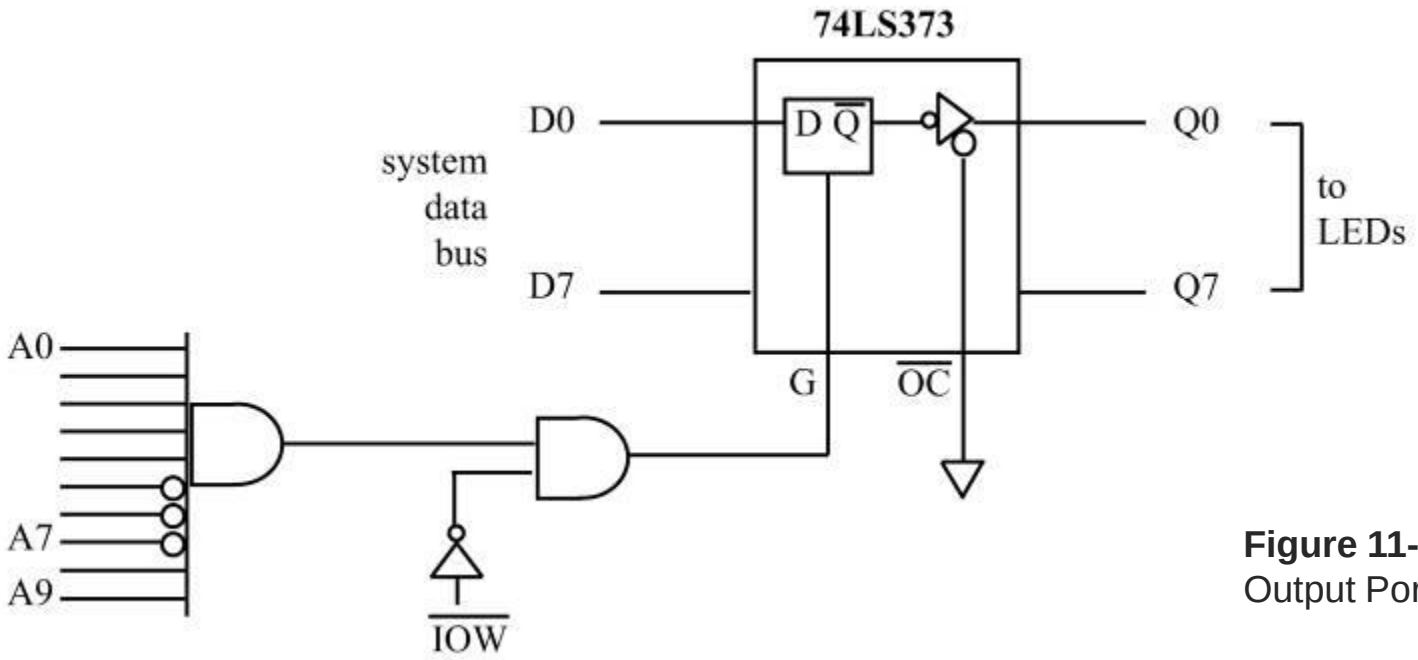


Figure 11-3 Design for Output Port Address 31FH.

Example 11-2

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

Solution:

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

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

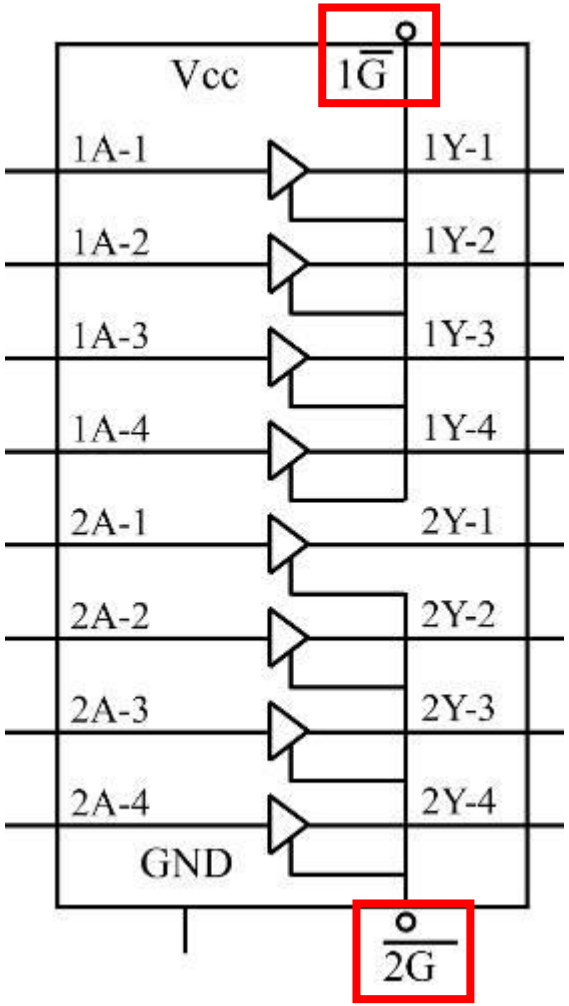
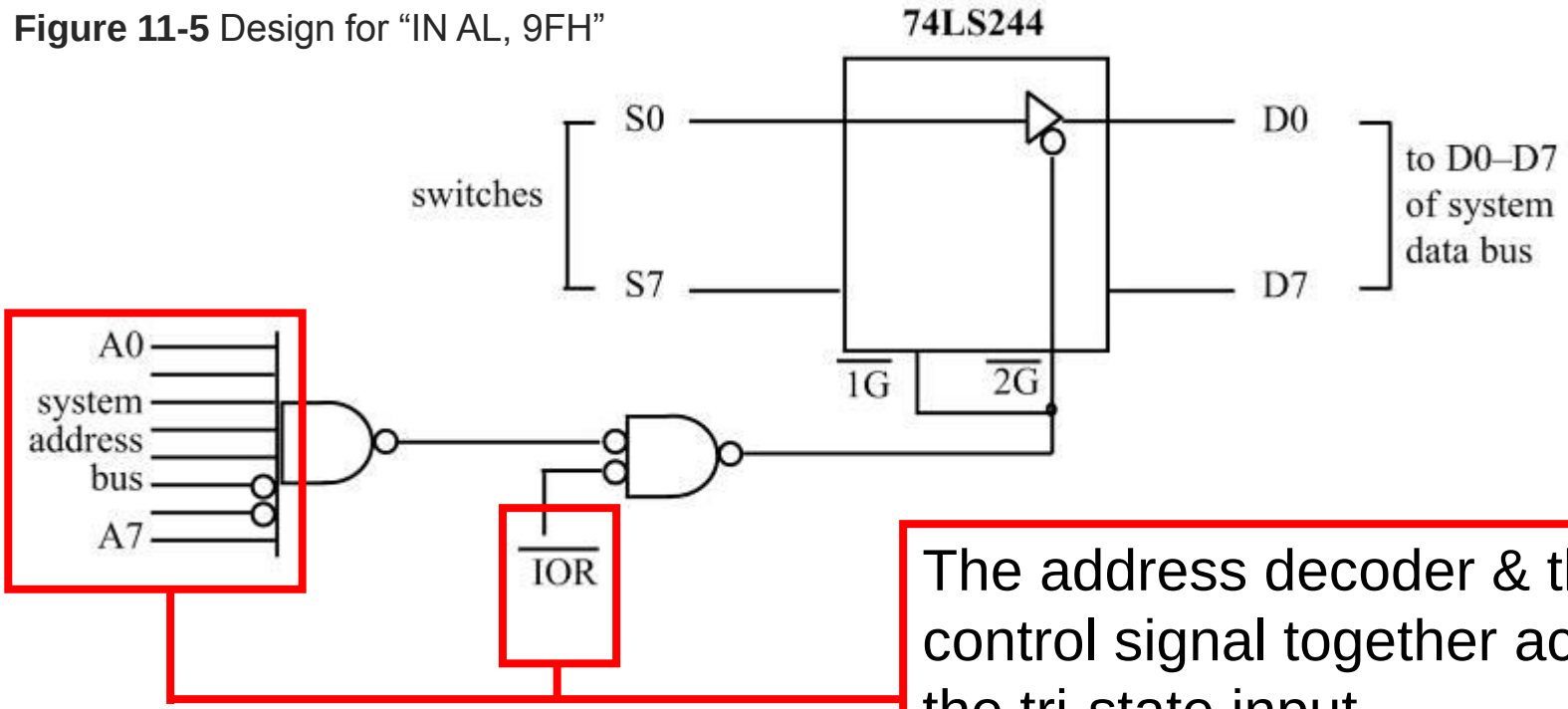


Figure 11-4 74LS244 Octal Buffer

- 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"

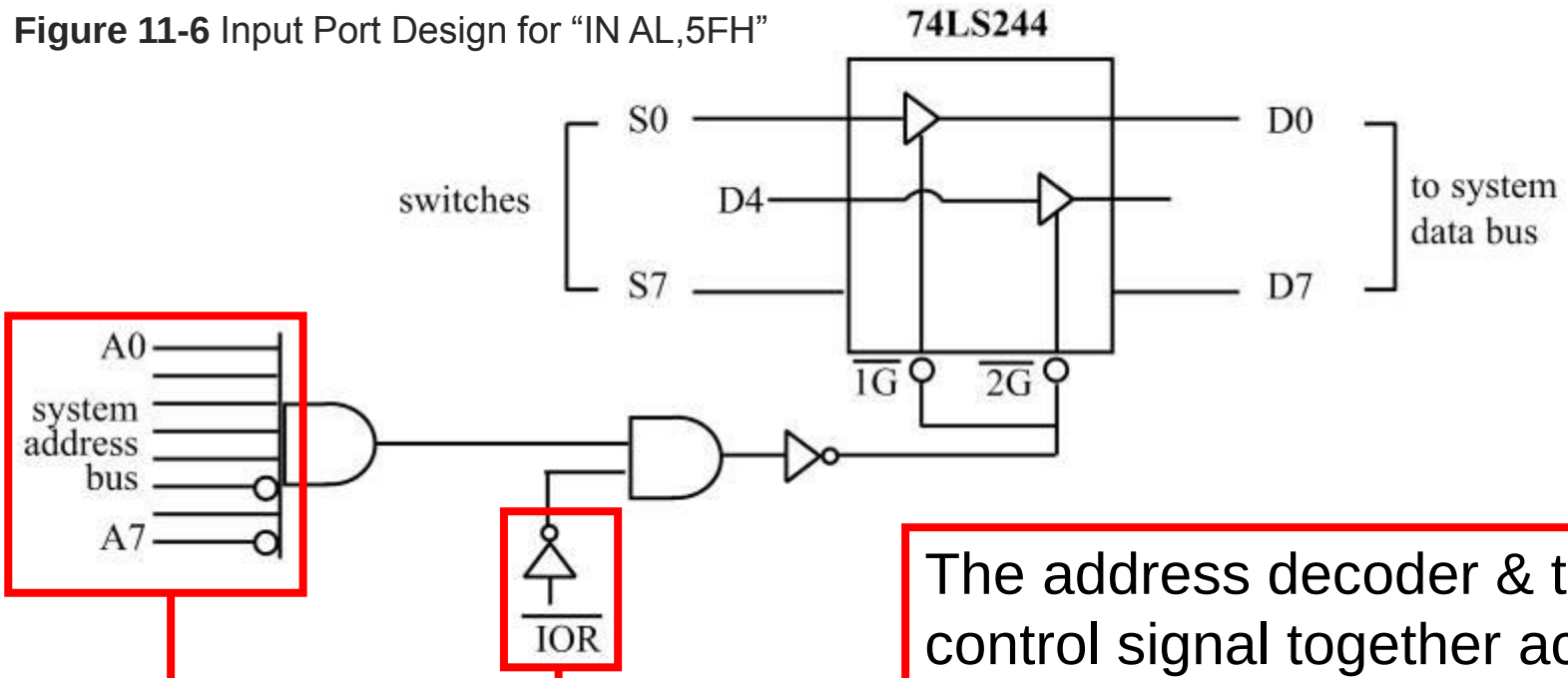


The address decoder & the **IOR** control signal together activate the tri-state input.

- 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,5FH"



The address decoder & the **IOR** control signal together activate the tri-state input.

- 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).

11.3: I/O ADDRESS MAP OF x86 PCs

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		

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.

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.

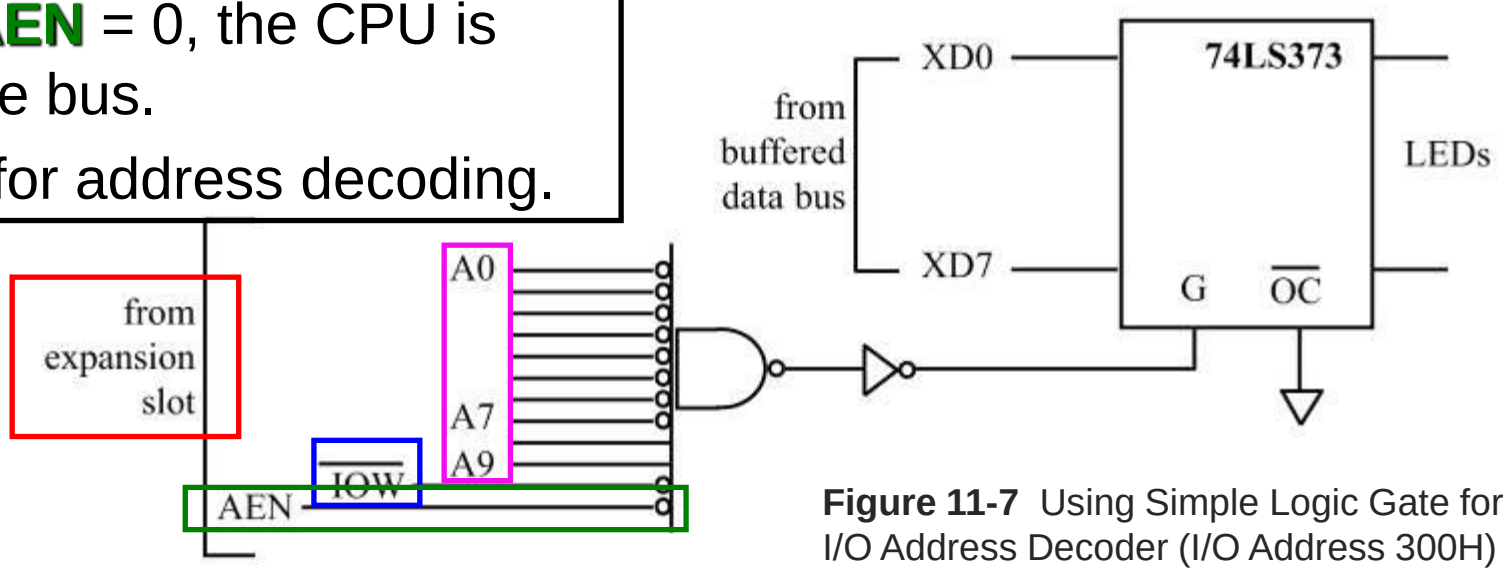
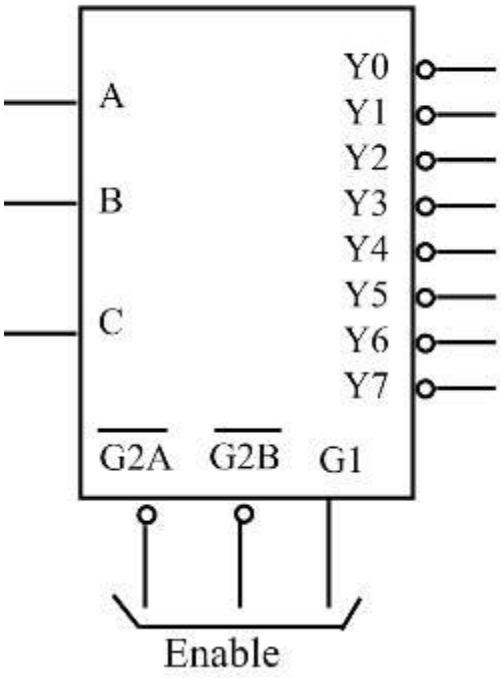


Figure 11-7 Using Simple Logic Gate for I/O Address Decoder (I/O Address 300H)

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

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

Block Diagram



Function Table

Inputs			Outputs
Enable		Select	
G1	G2	C B A	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.

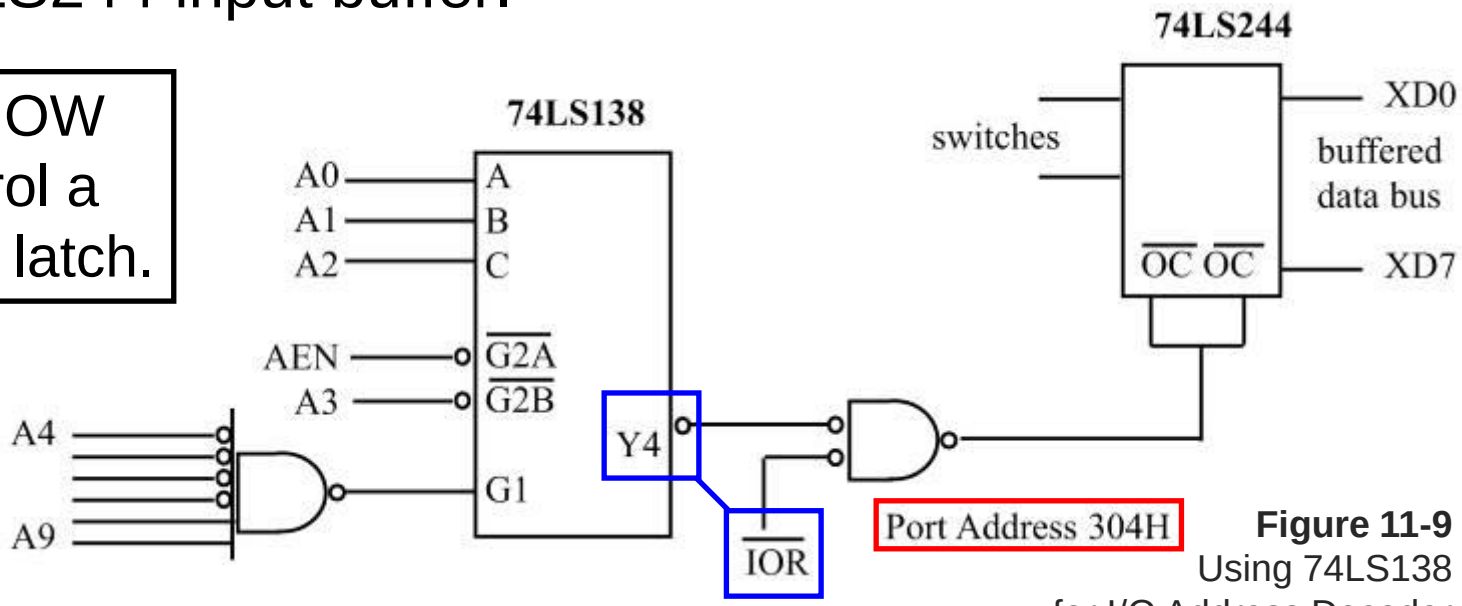
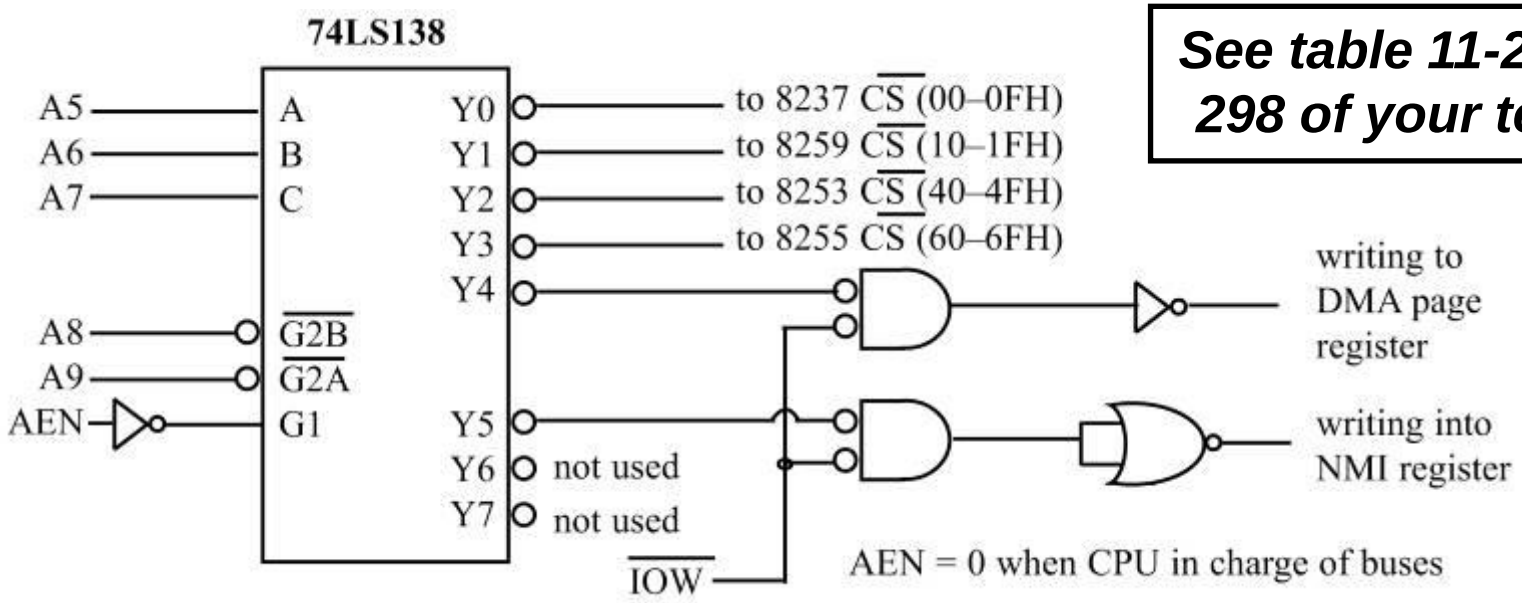


Figure 11-9
Using 74LS138 for I/O Address Decoder

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

11.4: 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.

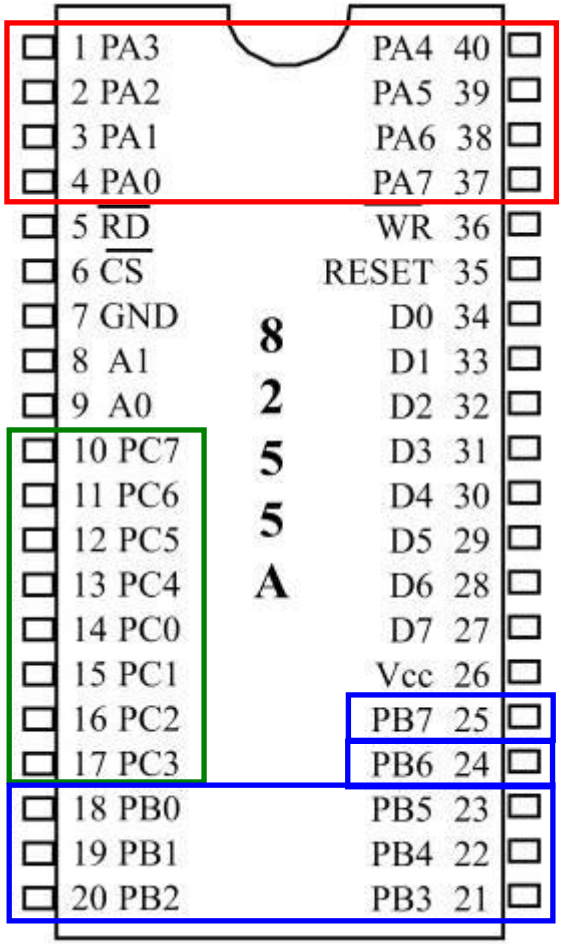


Figure 11-11 8255 PPI Chip

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.

11.4: PROGRAMMING & INTERFACING THE 8255

- **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.

Any of the ports A, B, CL & CU are programmable as input *or* output.

All bits are out or all are in.

No control of individual bits

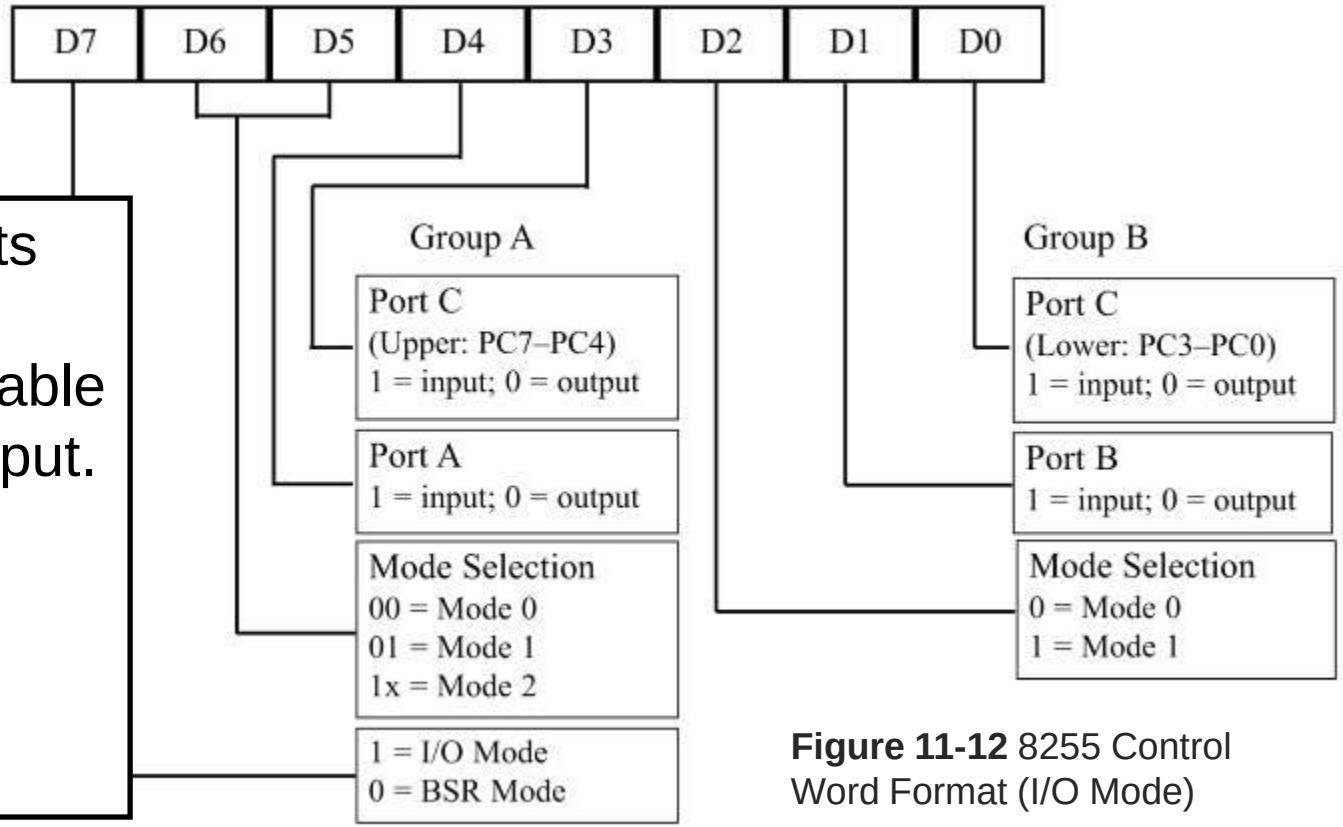


Figure 11-12 8255 Control Word Format (I/O Mode)

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

Programming of 8255 ports in simple I/O mode is illustrated in Examples 11-4, 11-5, and 11-6 on pages 301 -303 of your textbook.

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

- Example 11-4

```
B8255C EQU 300H
  CNTL EQU 83H
   MOV DX,B8255C+3
   MOV AL,CNTL
   OUT DX,AL
   MOV DX,B8255C+1
   IN AL,DX
   MOV DX,B8255C
   OUT DX,AL
   MOV DX,B8255C+2
   IN AL,DX
   AND AL,0FH
   ROL AL,1
   ROL AL,1
   ROL AL,1
   ROL AL,1
   OUT DX,AL
```

- Example 11-5

```
MOV AL,90H
MOV DX,313H
OUT DX,AL
MOV DX,310H
IN AL,DX
MOV DX,311H
OUT DX,AL
MOV DX,312H
OUT DX,AL
```

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

- Example 11-6

```
    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
    CALL QSDELAY
    MOV AH,01
    INT 16H
    JZ AGAIN
    MOV AH,4CH
    INT 21H

    QSDELAY PROC NEAR
    MOV CX,16572
    PUSH AX
W1: IN AL,61H
    AND AL,00010000B
    CMP AL,AH
    JE W1
    MOV AH,AL
    LOOP W1
    POP AX
    RET
    QSDELAY ENDP
```

11.4: PROGRAMMING & INTERFACING THE 8255
Visual C/C++ I/O programming

- There is no object or class for directly accessing I/O ports in the full Windows version of Visual C++.
  - This precludes any hacking into the system hardware.
    - This applies to Windows NT, 2000, XP, and higher.
- To access I/O and other hardware features of the x86 PC in the XP environment requires use of the Windows Platform SDK provided by Microsoft.
  - Direct I/O addressing is available Windows 9x using Visual C++ in console mode.

11.4: PROGRAMMING & INTERFACING THE 8255
Visual C/C++ I/O programming

- The instruction syntax for I/O operations is shown:

Table 11-5: I/O Operations in Microsoft Visual C++ (for Windows 98)

x86 Assembly	Visual C++
OUT port#,AL	_outp(port#,byte)
OUT DX,AL	_outp(port#,byte)
IN AL,port#	_inp(port#)
IN AL,DX	_inp(port#)

- C programming makes no distinction between the 8-bit and 16-bit I/O addresses.
  - In the instruction "**`outp (port#, byte)`**" the port # can take any address value between 0000 and FFFFH.
    - See Examples 11-8 and 11-9 on pages 307 & 308.

11.4: PROGRAMMING & INTERFACING THE 8255
Visual C/C++ I/O programming

- Example 11-8

```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
#include<iomanip.h>
#include<windows.h>
void main()
{
   cout<<setiosflags(ios::unitbuf);
   cout<<"This program toggles the bits for Port A and Port B.";
   _outp(0x303,0x80);
   do
   {
      _outp(0x300,0x55);
      _outp(0x301,0x55);
      _sleep(500);
      _outp(0x300,0xAA);
      _outp(0x301,0xAA);
      _sleep(500);
      }
   while(!kbhit());
   }
```

11.4: PROGRAMMING & INTERFACING THE 8255
Visual C/C++ I/O programming

- Example 11-9

```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
#include<iomanip.h>
#include<windows.h>
#include<process.h>
void main()
{
   unsigned char mybyte;
   cout<<setiosflags(ios::unitbuf);
   system("CLS");
   _outp(0x303,0x90);
   _sleep(5);
   mybyte=_inp(0x300);
   _outp(0x301,mybyte);
   _sleep(5);
   _outp(0x302,mybyte);
   _sleep(5);
   cout<<mybyte;
   cout<<"\n\n";
   }
```

11.4: PROGRAMMING & INTERFACING THE 8255
I/O programming in Linux C/C++

- Linux is a popular operating system for the x86 PC.
  - Find the C/C++ compiler at: *http://gcc.gnu.org*
  - More information can be found at: *www.microdigitaled.com* .

See Examples 11-10 & 11-11 on pages 309 & 310

Table 11-6: Input/Output Operations in Linux

x86 Assembly	Linux C/C++
OUT port#,AL	outb(byte,port#)
OUT DX,AL	outb(byte,port#)
IN AL,port#	inb(port#)
IN AL,DX	inb(port#)

The x86 PC

assembly language, design, and interfacing

fifth edition

Prentice Hall

The x86 PC

assembly language, design, and interfacing

fifth edition

MUHAMMAD ALI MAZIDI
JANICE GILLISPIE MAZIDI
DANNY CAUSEY