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:

<table>
<thead>
<tr>
<th>Format</th>
<th>Inputting Data</th>
<th>Outputting Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>IN AL, port#</td>
<td>OUT port#, AL</td>
</tr>
<tr>
<td>(2)</td>
<td>MOV DX, port#</td>
<td>MOV DX, port#</td>
</tr>
<tr>
<td></td>
<td>IN AL, DX</td>
<td>OUT DX, AL</td>
</tr>
</tbody>
</table>
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
Figure 8–52 Input bus cycle of the 8088.
Output Bus Cycle of the 8088

Figure 8–53  Output bus cycle of the 8088.
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*:

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

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

<table>
<thead>
<tr>
<th>Example 11-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>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'.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution:</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACK: IN AL,22H ;get the temperature from port # 22H</td>
</tr>
<tr>
<td>CMP AL,100 ;is temp = 100?</td>
</tr>
<tr>
<td>JNZ BACK ;if not, keep monitoring</td>
</tr>
<tr>
<td>MOV BH,'Y' ;temp = 100, load 'Y' into BH</td>
</tr>
</tbody>
</table>
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

- Assume that AX = 76A9h.: Analyze the data transfer for a) 8088 b) 8086 when MOV DX, 648h OUT DX, AX

- 8088 case
  - 1\textsuperscript{st} 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
  - 2\textsuperscript{nd} Bus Cycle (Similar to 1\textsuperscript{st} 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 IOWC is activated
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

• 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

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

[Diagram showing 74F244 Port 0 connected to a switch.]

Poll the switch waiting for it to close
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.

![Figure 11-1 74LS373 D Latch](image-url)
11.2: I/O ADDRESS DECODING AND DESIGN using 74LS373 in an output port design

For an output latch, it is common to AND the output of the address decoder with control signal \textbf{IOW}.

- To provide the latching action.

Figure 11-2 Design for “OUT 99H, AL”.
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.
11.2: I/O ADDRESS DECODING AND DESIGN

IN port design using 74LS244

- 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.

**Figure 11-4 74LS244 Octal Buffer**
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.

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

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

### Table 11-1: I/O Address Map of x86 PCs

<table>
<thead>
<tr>
<th>Hex Range</th>
<th>Device</th>
<th>Hex Range</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>000–01F</td>
<td>DMA controller 1, 8237A-5</td>
<td>378–37F</td>
<td>Parallel printer port 1</td>
</tr>
<tr>
<td>020–03F</td>
<td>Interrupt controller 1, 8259A, Master</td>
<td>380–38F</td>
<td>SDLC, bisynchronous 2 Cluster</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bisynchronous 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Monochrome display/printer adapter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Enhanced graphics adapter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Color graphics monitor adapter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Disk controller</td>
</tr>
<tr>
<td>300–31F</td>
<td>Prototype card</td>
<td>A2E1</td>
<td>GPIB (adapter 5)</td>
</tr>
<tr>
<td>360–363</td>
<td>FC network (low address)</td>
<td>C2E1</td>
<td>GPIB (adapter 6)</td>
</tr>
<tr>
<td>364–367</td>
<td>Reserved</td>
<td>E2E1</td>
<td>GPIB (adapter 7)</td>
</tr>
<tr>
<td>368–36B</td>
<td>FC network (high address)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36C–36F</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

**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.
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.

```assembly
; TOGGING 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.
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**

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 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
The x86 PC
Assembly Language, Design, and Interfacing
By Muhammad Ali Mazidi, Janice Gillespie Mazidi and Danny Causey

Pearson Prentice Hall - Upper Saddle River, NJ 07458
Handshaking
Printer Interface Circuit

The x86 PC
Assembly Language, Design, and Interfacing
By Muhammad Ali Mazidi, Janice Gillespie Mazidi and Danny Causey

Pearson Prentice Hall - Upper Saddle River, NJ 07458
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
  JNZ POLL_BUSY

  BUSY input checked

  MOV AL, [SI]
  MOV DX, 8000h
  OUT DX, AL

  Character is output

  MOV AL, 00h ; STB = 0
  MOV DX, 8002h
  OUT DX, AL

  ; delay for STB = 0
  MOV BX, 0Fh
  STROBE: DEC BX
  JNZ STROBE

  STB bar = 1

  INC SI
  DEC CL
  JNZ POLL_BUSY

  So as the strobe
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.
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.

<table>
<thead>
<tr>
<th>CS</th>
<th>A1</th>
<th>A0</th>
<th>Selects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Port A</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Port B</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Port C</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Control register</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>8255 is not selected</td>
</tr>
</tbody>
</table>

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

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)](image-url)
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

- **Addressing**
  - Port A: C000h
  - Port B: C001h
  - Control Register: C002h

- **Addressing Scheme**
  - From the microprocessor address bus
  - To Select
    - Port A: 00h, 0
    - Port B: 01h, 0
    - Port C: 10h, 0
    - Control Reg: 11h, 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

```assembly
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

Initialize this device with the appropriate Control Word. Read from PORT C 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
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

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

```assembly
MOV DX, 303h
MOV AL, 80h
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

+5V

PA0 → 0 8 4.7KΩ
PA1 → 1 9 4.7KΩ
PA2 → 2 A 4.7KΩ
PA3 → 3 B 4.7KΩ
PA4 → 4 C 4.7KΩ
PA5 → 5 D 4.7KΩ
PA6 → 6 E 4.7KΩ
PA7 → 7 F 4.7KΩ

8255A mode 0

PC6
PC7

PC0–PC5 not used

A2
A0
A1

PB0–PB7 not used

D0–D7

KYCHK

Read Input Port

All Keys Open?

Wait for Key Debounce

KYPUSH:

Read Input Port

Is a Key Pressed?

No

Yes
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 00 01 02 03 04 0005 05 06 07 0008 08 09 0A 0B 0C 000D 0D 0E 0F
0010 KEY_CODE      SEGMENT BYTE
0000 KEY_CODE DB 0,1,2,3,4
                DB 5,6,7
                DB 8,9,0AH,0BH,0CH
                DB 0DH,0EH,0FH
                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 1011111B ; PC6 low
= 007F COL_2_LOW EQU 0111111B ; PC7 low
= 003F BOTH_COL_LOW EQU 0011111B ; PC6 and PC7 low
= 00FF KEY_UP EQU 0FFH ; Input OFFH 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
ENDP

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

KEYBOARD PROC NEAR
PUSH DS
PUSH CX
PUSH SI
MOV AX, KEY_CODE
MOV DS, AX

;Save registers about to be used
;Point DS to the key codes

;Wait for previous key to be released

MOV AL, BOTH_COL_LOW
OUT PORT_C, AL
IN AL, PORT_A
CMP AL, KEY_UP
JNE POLL1
CALL DELAY

;Scan both columns
;Column lines on PC6 and PC7
;Read keyboard
;All keys up?
;No - so wait
;Yes - wait for bounce on release

;Wait for a new key to be pressed

POLL2:
IN AL, PORT_A
CMP AL, KEY_UP
JE POLL2
CALL DELAY

;Read keyboard
;Any keys down?
;No - so wait
;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          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:  RCK     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
0050  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,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

<table>
<thead>
<tr>
<th>Connection</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA0-PA7 to DO-D7 of ADC</td>
<td>CHANNEL SELECTION(out), DATA READ (in)</td>
</tr>
<tr>
<td>PB0 TO INTR</td>
<td>PORT B AS INPUT</td>
</tr>
<tr>
<td>PC0 TO WR</td>
<td>PORT C AS OUTPUT</td>
</tr>
<tr>
<td>PC1 TO RD</td>
<td>PORT C AS OUTPUT</td>
</tr>
</tbody>
</table>
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 microseconds
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
ENDS ; I/O

11  B   00001011