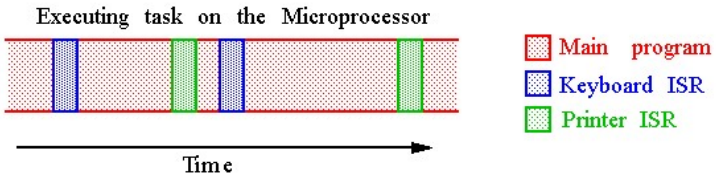
CompE 475 Microprocessors

Chapter 12: Basic Interrupt Processing

Dr. Khurram Waheed
San Diego State University

Interrupts

- Interrupt processing is an alternative to polling.
  - Executing task on the Microprocessor

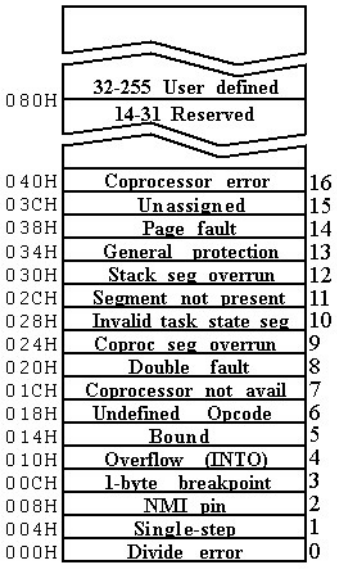


- The Intel microprocessors support hardware interrupts through:
  - Two pins that allow interrupt requests, INTR and NMI
  - One pin that acknowledges, INTA, the interrupt requested on INTR.
- And software interrupts through instructions:
  - INT, INTO, INT 3, BOUND
- Control is provided through
  - IF and TF flag bits
  - IRET and IRETD

Interrupt Vector Table

- INT and INT3 behave in a similar way.
  - INT n:
    - Calls ISR located at vector n (n*4).
- The INT instruction requires two bytes of memory, opcode plus n.
  - BOUND and INTO are both conditional.
  - ***BOUND:***
    - **BOUND** *AX*, DATA ;Compares AX with DATA
    - AX is compared with DATA and DATA+1, if less than an interrupt occurs.
    - AX is compared with DATA+2 and DATA+3, if greater than an interrupt occurs.
  - ***INTO:***
    - Checks the overflow flag (OF). If OF=1, the ISR is called.
- IRET removes 6 bytes from the stack, 2 for IP, 2 for CS and 2 for FLAGS.

Interrupt Vector Table



The interrupt vector table is located in the first 1024 bytes of memory at addresses 000000H through 0003FFH.

There are 256 4-byte entries (segment and offset in real mode).

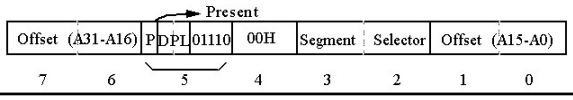
Seg high	Seg low	Offset high	Offset low
Byte 3	Byte 2	Byte 1	Byte 0

Real Mode Interrupts

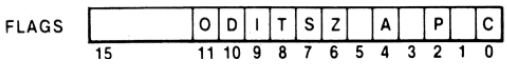
- After the execution of each instruction, the microprocessor determines whether an interrupt is active by checking, in order:
  - Other instruction executions
  - Single-step
  - NMI
  - Coprocessor segment overrun
  - INTR
  - INT
- If one or more of these conditions are present, then:
  - FLAGS is pushed onto the stack
  - Both the interrupt (IF) and trap (TF) flags are cleared, which disables the INTR pin and the trap or single-step feature.
  - The CS and IP are pushed onto the stack.
  - The interrupt vector contents are fetched and loaded into CS and IP and execution resumes in the ISR.
  - On IRET, CS, IP and FLAGS are popped.
  - IF and TF are set to the state prior to the interrupt.

Real and Protected Mode Interrupts

- The return address (CS/IP) is pushed onto the stack during the interrupt.
- The return address can point to:
  - The next instruction.
  - The offending (current) instruction.
  - The latter case occurs for interrupts 0, 5, 6, 7, 8, 10, 11, 12 and 13.
    - This makes it possible to try the instruction again.
- Protected Mode:
  - The same interrupt assignments are made and the same sequence of operations occurs in protected mode but the interrupt table is different.
  - Instead, 256 interrupt descriptors are used in the interrupt descriptor table (IDT).



Interrupt Flag Bits



Interrupt Flag – Allows INTRE input to cause an interrupt, Set and cleared by STI and CLI

Trap Flag – A *single step* interrupt; causes interrupt after every instruction

```
;A procedure that sets TF to enable trap.
;
TRON   PROC  NEAR

       PUSH  AX            ;save registers
       PUSH  BP
       MOV   BP,SP         ;get SP
       MOV   AX,[BP+8]     ;get flags from stack
       OR    AH,1          ;set TF
       MOV   [BP+8],AX     ;save flags
       POP   BP            ;restore registers
       POP   AX
       IRET

TRON   ENDP
```

```
;A procedure that clears TF to disable trap.
;
TROFF PROC  NEAR

       PUSH  AX            ;save registers
       PUSH  BP
       MOV   BP,SP         ;get SP
       MOV   AX,[BP+8]     ;get TF
       AND   AH,0FEH       ;clear TF
       MOV   [BP+8],AX     ;save flags
       POP   BP            ;restore registers
       POP   AX
       IRET

TROFF ENDP
```


A Debug (Trace) Procedure

EXAMPLE 12–3 (1)

```
.MODEL TINY
.CODE
RNAME DB 'AX = ','BX = ','CX = ','DX = '


      DB 'SP = ','BP = ','SI = ','DI = '


      DB 'IP = ','FL = ','CS = ','DS = '


      DB 'ES = ','SS = '


DISP  MACRO PAR1
      PUSH  AX
      PUSH  DX
      MOV   DL,PAR1
      MOV   AH,6
      INT   21H
      POP   DX
      POP   AX
      ENDM
```

(2)

```
CRLF  MACRO
      DISP  13
      DISP  10
      ENDM

TRACE PROC  FAR USES AX BP BX

      MOV   BX,OFFSET RNAME    ;address names
      CRLF
      CALL  DREG               ;display AX
      POP   AX                 ;get BX
      PUSH  AX
      CALL  DREG               ;display BX
      MOV   AX,CX
      CALL  DREG               ;display CX
      MOV   AX,DX
      CALL  DREG               ;display DX
      MOV   AX,SP
      ADD   AX,12
      CALL  DREG               ;display SP
      MOV   AX,BP
      CALL  DREG               ;display BP
      MOV   AX,SI
      CALL  DREG               ;display SI
      MOV   AX,DI
      CALL  DREG               ;display DI
      MOV   BP,SP
      MOV   AX,[BP+6]
```

(3)

```
      CALL  DREG               ;display IP
      MOV   AX,[BP+10]
      CALL  DREG               ;display Flags
      MOV   AX,[BP+8]
      CALL  DREG               ;display CX
      MOV   AX,DS
      CALL  DREG               ;display DS
      MOV   AX,ES
      CALL  DREG               ;display ES
      MOV   AX,SS
      CALL  DREG               ;display SS
      IRET

TRACE ENDP

DREG  PROC  NEAR USES CX
      MOV   CX,5               ;load count
DREG1:
      DISP  CS:[BX]            ;display character
      INC   BX                 ;address next
      LOOP  DREG1              ;repeat 5 times
      MOV   CX,4               ;load count
```

(4)

```
DREG2:
      ROL   AX,1               ;position digit
      ROL   AX,1
      ROL   AX,1
      ROL   AX,1
      PUSH  AX
      AND   AL,0FH             ;convert to ASCII
      .IF AL > 9
        ADD AL,7
      .ENDIF
      ADD   AL,30H
      DISP   AL
      POP   AX
      LOOP  DREG2              ;repeat 4 times
      DISP  ' '
      RET

DREG  ENDP
      END
```

Storing an Interrupt Vector in Interrupt Table

EXAMPLE 12–4

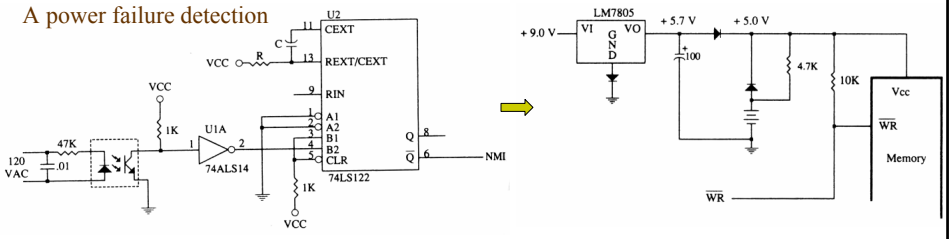
```
.MODEL TINY
.CODE
;A program that installs NEW40 at INT 40H.
;
.STARTUP
        JMP   START
OLD     DD    ?
;
;new interrupt procedure
;
NEW40 PROC  FAR

      IRET

NEW40 ENDP
```

```
START:
                MOV   AX,CS     ;get data segment
                MOV   DS,AX
                MOV   AH,35H    ;get old interrupt vector
                MOV   AL,40H
                INT   21H
                MOV   WORD PTR OLD,BX
                MOV   WORD PTR OLD+2,ES
        ;
        ;install new interrupt vector 40H
        ;
                MOV   DX,OFFSET NEW40
                MOV   AH,25H
                MOV   AL,40H
                INT   21H
        ;
        ;leave NEW40 in memory
        ;
                MOV   DX,OFFSET START
                SHR   DX,1
                SHR   DX,1
                SHR   DX,1
                SHR   DX,1
                INC   DX
                MOV   AX,3100H
                INT   21H
                END
```


Hardware Interrupts

- Interrupt Inputs
  - Non masking Interrupt - NMI (Type 2; internal)
  - General INTR (Type 20H – FFH; External)
- Interrupt Output
  - /INTA (used in response to INTR to apply a vector number to the data bus)

A power failure detection



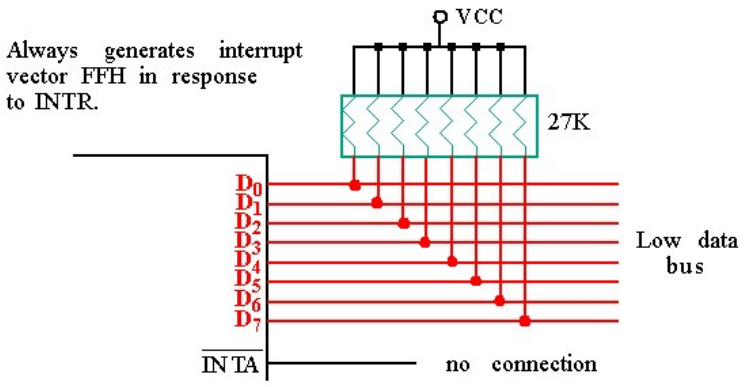
NMI is typically used for major system faults

Hardware Interrupts

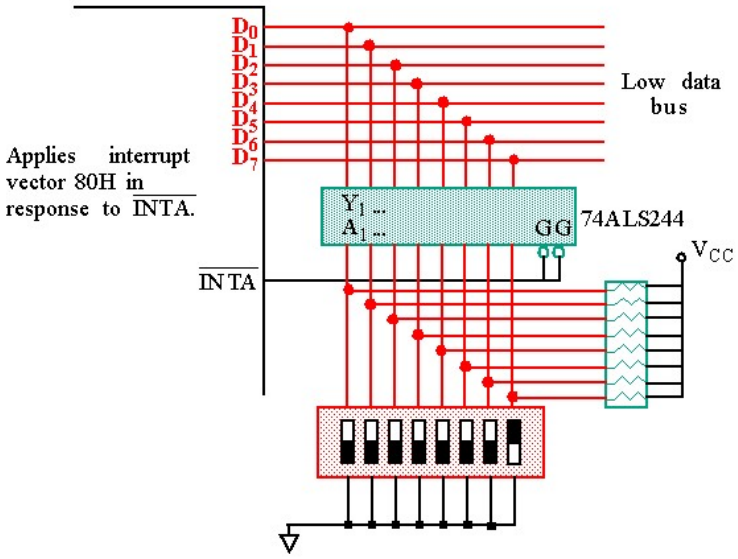
- The INTR pin must be externally decoded to select a vector.
  - Any vector is possible, but the interrupt vectors between 20H and FFH are usually used (Intel reserves vectors between 00H and 1FH).
  - INTA is an output of the microprocessor to signal the external decoder to place the interrupt number on data bus connections D7-D0.
- The INTR pin is set by an external device (8259A) and cleared in the ISR.
  - The input is automatically disabled by the microprocessor once it is recognized and re-enabled by IRET or IRETD instruction.
- Timing diagram of the handshake.

Hardware Interrupts

- Simplest method of generating an interrupt vector:



Tri-state buffer for generating interrupt vector



Converting INTR into an edge-triggered request input

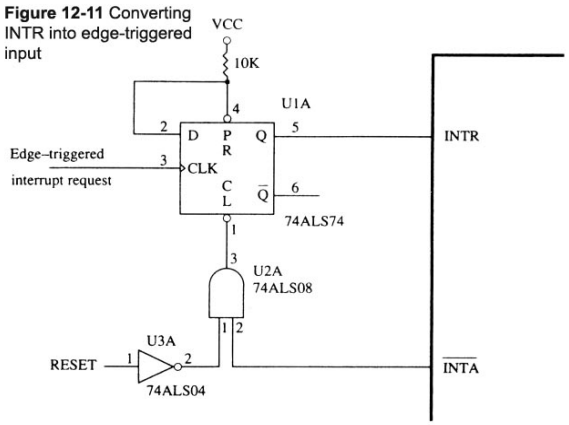
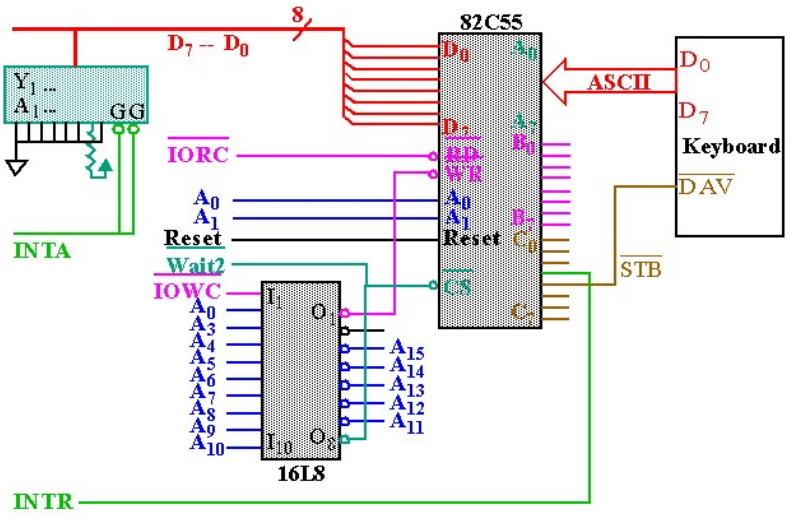


Figure 12-11 Converting INTR into edge-triggered input

Example: 8255 Interrupt Configuration

Keyboard Interface



Example: 8255 Interrupt Configuration

Keyboard Interface

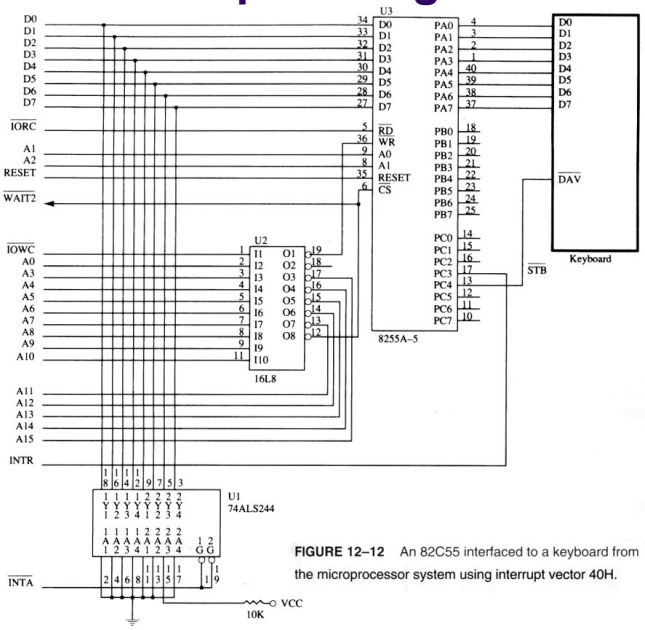


FIGURE 12–12 An 82C55 interfaced to a keyboard from the microprocessor system using interrupt vector 40H.

Example: 8255 Interrupt Configuration

Keyboard Interface

EXAMPLE 12–5

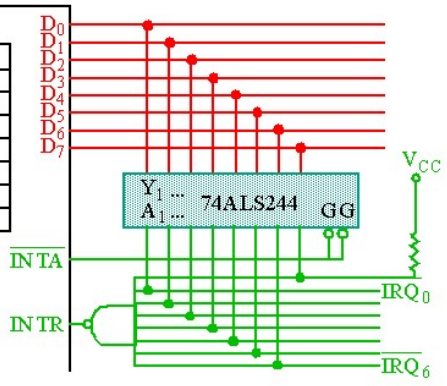
```
;An interrupt service procedure that reads a key
;from the keyboard in Figure 12-12.
;
PORTA EQU   500H
CNTR  EQU   506H

FIFO  DB 256 DUP (?)      ;queue


INP   DW    ?             ;input pointer
OUTP  DW    ?             ;output pointer

KEY   PROC  FAR USES AX BX DI DX

      MOV   BX,CS:INP     ;load input pointer
      MOV   DI,CS:OUTP    ;load output pointer

      INC   BL            ;test for queue = full
      CMP   BX,DI
      JE    FULL          ;if queue is full

      DEC   BL
      MOV   DX,PORTA
      IN    AL,DX         ;get data from 82C55
      MOV   CS:[BX],AL    ;save data in queue
      INC   BYTE PTR INP
      JMP   DONE
FULL:
      MOV   AL,8          ;disable 82C55 interrupt
      MOV   DX,CNTR
      OUT   DX,AL
DONE:
      IRET

KEY   ENDP
```

EXAMPLE 12–6

```
;A procedure that reads data from the queue of
;Example 12-5 and returns with it in AH.
;
READ  PROC  FAR USES BX DI DX

EMPTY:
      MOV   BX,CS:INP       ;load input pointer
      MOV   DI,CS:OUTP      ;load output pointer
      CMP   BX,DI
      JE    EMPTY           ;if queue is empty

      MOV   AH,CS:[DI]      ;get data
      MOV   AL,9            ;enable 82C55 interrupt

      MOV   DX,CNTR
      OUT   DX,AL
      INC   BYTE PTR CS:OUTP
      RET

READ  ENDP
```

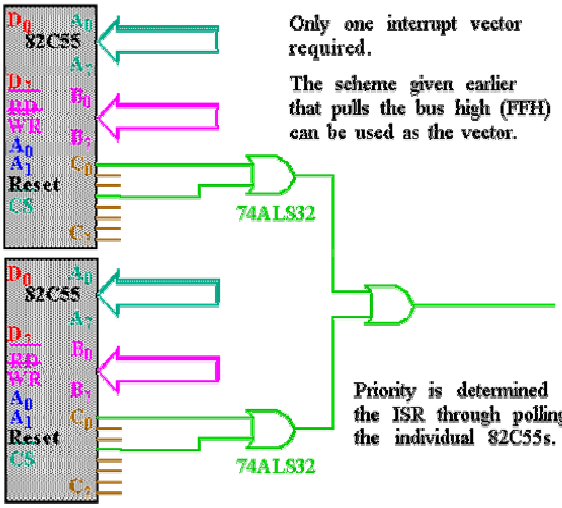

Handling more than 1 IRQ

IRQs							
6	5	4	3	2	1	0	Vect
1	1	1	1	1	1	0	FEH
1	1	1	1	1	0	1	FDH
1	1	1	1	0	1	1	FBH
1	1	1	0	1	1	1	F7H
1	1	0	1	1	1	1	EFH
1	0	1	1	1	1	1	DFH
0	1	1	1	1	1	1	BFH



- If any of /IRQ$_x$ goes low, the NAND goes low requesting an interrupt.
  - Note that if more than one IRQ goes low, a unique interrupt vector is generated and an interrupt priority needs to be defined.
  - The Interrupt Vector table must be expanded to accommodate this.

Daisy Chained Mechanism – Multiple IRQs



Only one interrupt vector required.

The scheme given earlier that pulls the bus high (FFH) can be used as the vector.

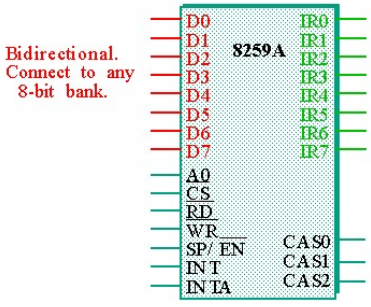
Priority is determined in the ISR through polling the individual 82C55s.

EXAMPLE 12–7

```
;A procedure that services the daisy-chain
;of Figure 12-14.
;
C1     EQU   504H      ;first 82C55
C2     EQU   604H      ;second 82C55
MASK1  EQU   1         ;INTRB
MASK2  EQU   8         ;INTRA

POLL   PROC  FAR USES AX DX

       MOV   DX,C1     ;address first 82C55
       IN    AL,DX     ;get port C
       TEST  AL,MASK1
       JNZ   LEVEL_0   ;if INTRB is set
       TEST  AL,MASK2
       JNZ   LEVEL_1   ;if INTRA is set

       MOV   DX,C2     ;address second 82C55
       IN    AL,DX     ;get port C
       TEST  AL,MASK1
       JNZ   LEVEL_2   ;if INTRB is set
       JMP   LEVEL_3   ;for INTRA

POLL   ENDP
```


8259A – Programmable Interrupt Controller

- The 8259A adds 8 vectored priority encoded interrupts to the microprocessor.
- It can be expanded to 64 interrupt requests by using one master 8259A and 8 slave units.



- /CS and /WR must be decoded. Other connections are direct to micro.

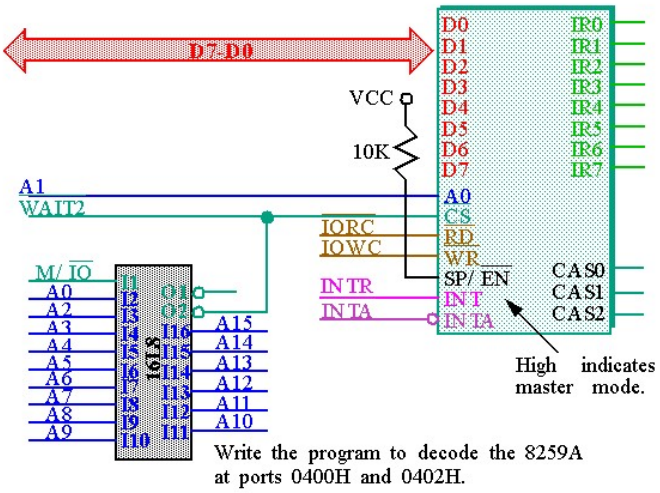
8259A – Programmable Interrupt Controller

The meaning of the other connections:

- /WR
  - Connects to a write strobe signal (one of 8 for the Pentium).
- /RD
  - Connects to the /IORC signal.
- INT
  - Connects to the INTR pin on the microprocessor.
- /INTA
  - Connects to the /INTA pin on the microprocessor.
- A0
  - Selects different command words in the 8259A.
- /CS
  - Chip select - enables the 8259A for programming and control.
- SP/~EN
  - Slave Program (1 for master, 0 for slave)/Enable Buffer (controls the data bus transievers when in buffered mode).
- CAS2-CAS0
  - Used as outputs from the master to the slaves in cascaded systems.

8259A – Programmable Interrupt Controller

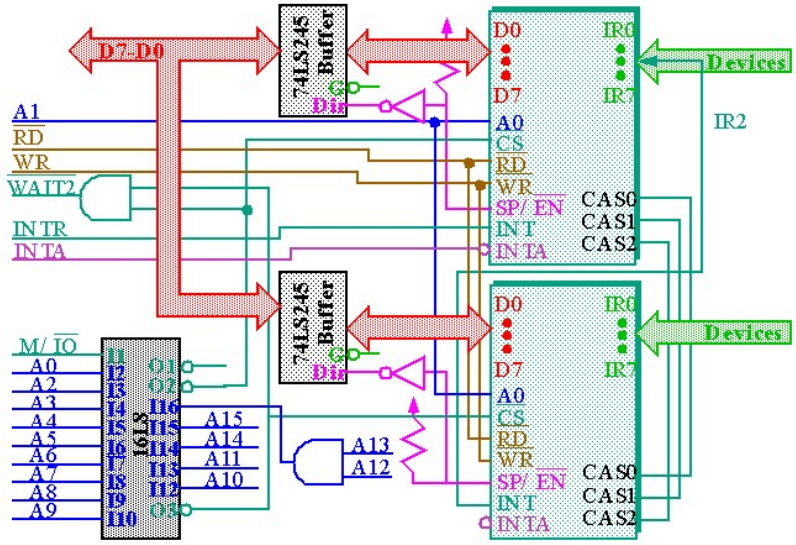
- A single 8259A connected in the 8086.



Write the program to decode the 8259A at ports 0400H and 0402H.

Connecting two 8259A – PICs

- Connecting Multiple PICs



Connecting two 8259A – PICs

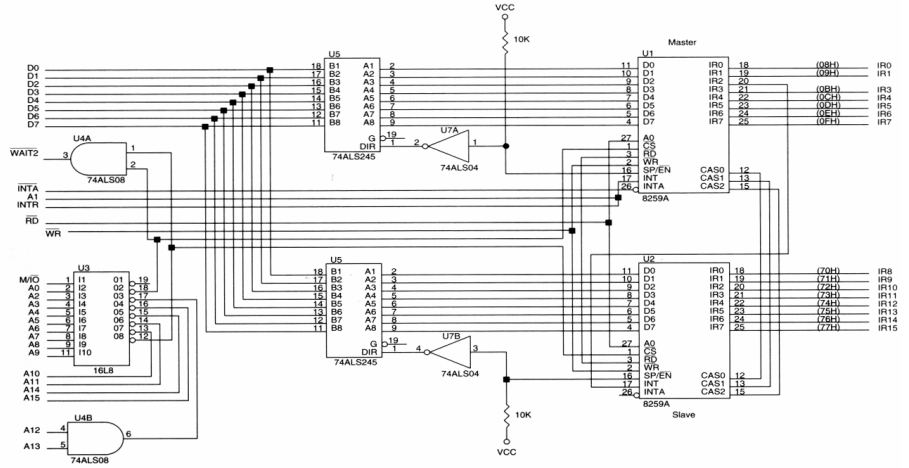
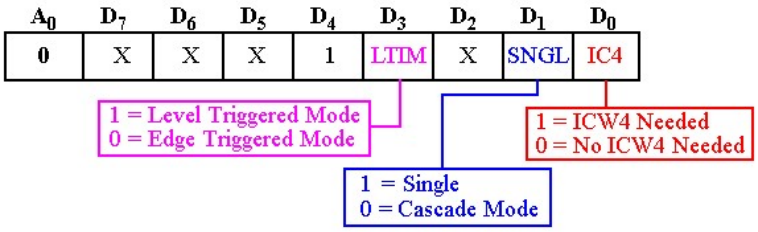


FIGURE 12–17 Two 8259As interfaced to the 8259A at I/O ports 0300H and 0302H for the master and 0304H and 0306H for the slave.

8259A - Programming

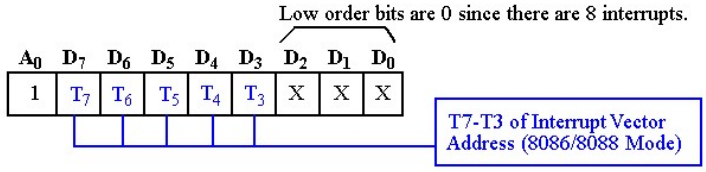
- Programmed by Initialization (ICWs) and Operation (OCWs) Command Words.
- There are 4 ICWs.
  - At power-up, ICW1, ICW2 and ICW4 must be sent.
  - If ICW1 indicates cascade mode, then ICW3 must also be sent.
- **ICW1**



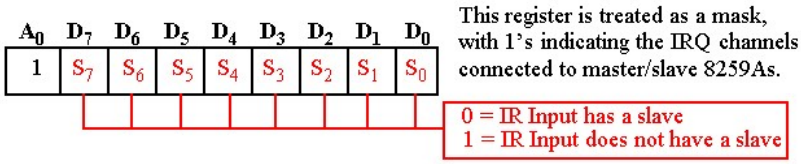
  - LTIM indicates if IRQ lines are positive edge-triggered or level-triggered.

8259A - Programming

- **ICW2**

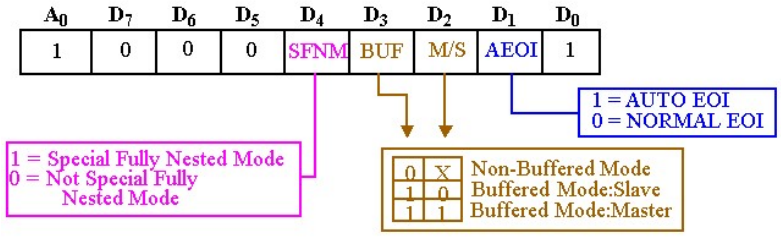


  - These bits determine the vector numbers used with the IRQ inputs.
  - For example, if programmed to generate vectors 08H-0FH, 08H is placed into these bit positions.
- **ICW3**



8259A - Programming

- **ICW4**

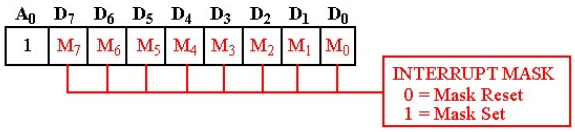


  - Fully nested mode allows the highest-priority interrupt request from a slave to be recognized by the master while it is processing another interrupt from a slave.
  - AEOI, if 1, indicates that an interrupt automatically resets the interrupt request bit, otherwise OCW2 is consulted for EOI processing.

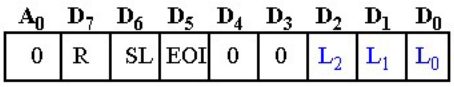
8259A - Programming

The Operation Command Words (OCWs) are used to direct the operation of the 8259A.

- **OCW1**



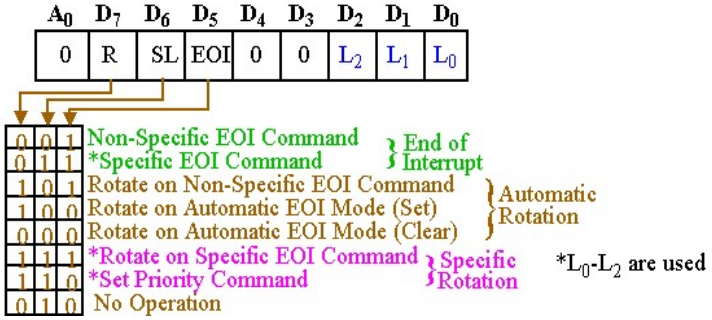
  - OCW1 is used to read or set the interrupt mask register.
    - If a bit is set, it will turn off (mask) the corresponding interrupt input.
- **OCW2**



  - Only programmed when the AEOI mode in ICW4 is 0.
  - Allows you to control priorities after each interrupt is processed.

8259A - Programming

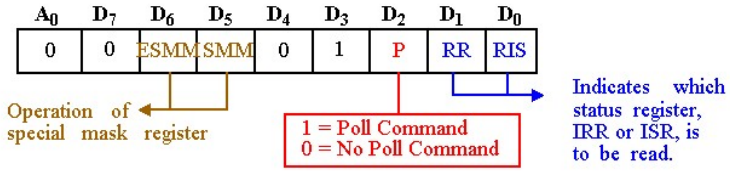
- **OCW2** (contd..)



- ***Non-specific EOI:*** Here, the ISR sets this bit to indicate EOI. The 8259A automatically determines which interrupt was active and re-enables it and lower priority interrupts.
- ***Specific EOI:*** ISR resets a specific interrupt request given by L2-L0.
- Rotate commands cause priority to be rotated w.r.t. the current one being processed.
- ***Set priority:*** allows the setting of the lowest priority interrupt (L2-L0).

8259A - Programming

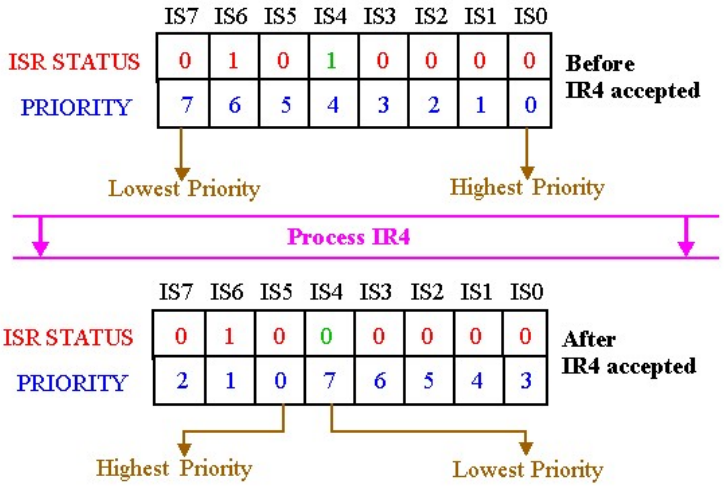
- **OCW3**



- If polling is set, the next read operation will read the poll word.
- If the leftmost bit is set in the poll word, the rightmost 3 bits indicate the active interrupt request with highest priority.
- Allows ISR to service highest priority interrupt.
- There are three status registers, Interrupt Request Register (IRR), In-Service Register (ISR) and Interrupt Mask Register (IMR).
- ***IRR:*** Indicates which interrupt request lines are active.
- ***ISR:*** Level of the interrupt being serviced.
- ***IMR:*** A mask that indicates which interrupts are on/off.

8259A - Programming

- **ISR** update procedure with rotating priority configured.



8259A - Programming

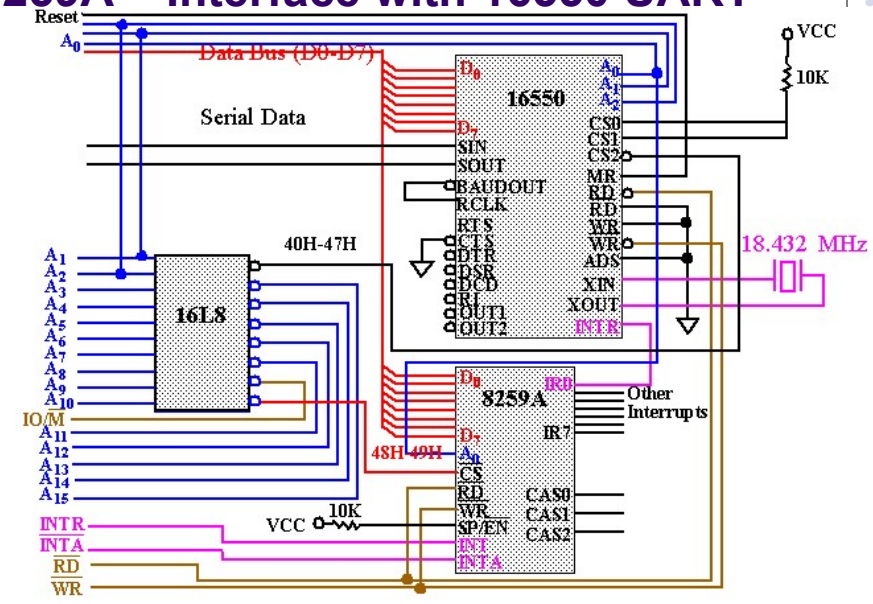
EXAMPLE 12–8

```
;Initialization software for the 16550 and 8259A
;of the circuit in Figure 12-21.
;
PIC1   EQU   48H        ;8259A control A0 = 0
PIC1   EQU   49H        ;8259A control A0 = 1
ICW1   EQU   1bH        ;8259A ICW1
ICW2   EQU   80H        ;8259A ICW2
ICW4   EQU   3          ;8259A ICW4
OCW1   EQU   0FEH       ;8259A OCW1
LINE   EQU   43H        ;16550 line register
LSB    EQU   40H        ;16550 Baud divisor LSB
MSB    EQU   41H        ;16550 Baud divisor MSB
FIFO   EQU   42H        ;16550 FIFO register
ITR    EQU   41H        ;16550 interrupt register

START PROC  NEAR
;
;Program 16550, but do not enable interrupts yet
;
       MOV   AL,10001010B    ;enable Baud divisor
       OUT   LINE,AL

       MOV   AL,120          ;program Baud rate
       OUT   LSB,AL          ;9600 Baud rate
       MOV   AL,0
       OUT   MSB,AL

       MOV   AL,00001010B    ;program 7-data, odd
       OUT   LINE,AL         ;parity, one stop
       MOV   AL,00000111B    ;enable transmitter and
       OUT   FIFO,AL         ;and receiver
;
;Program 8259A
;
       MOV   AL,ICW1         ;program ICW1
       OUT   PIC1,AL

       MOV   AL,ICW2         ;program ICW2
       OUT   PIC2,AL

       MOV   AL,ICW4         ;program ICW4
       OUT   PIC2,AL

       MOV   AL,OCW1         ;program OCW1
       OUT   PIC2,AL
       STI                   ;enable system INTR pin
;
;enable 16550 interrupts
;
       MOV   AL,5            ;enable receiver and
       OUT   ITR,AL          ;error interrupts
       RET

START ENDP
```

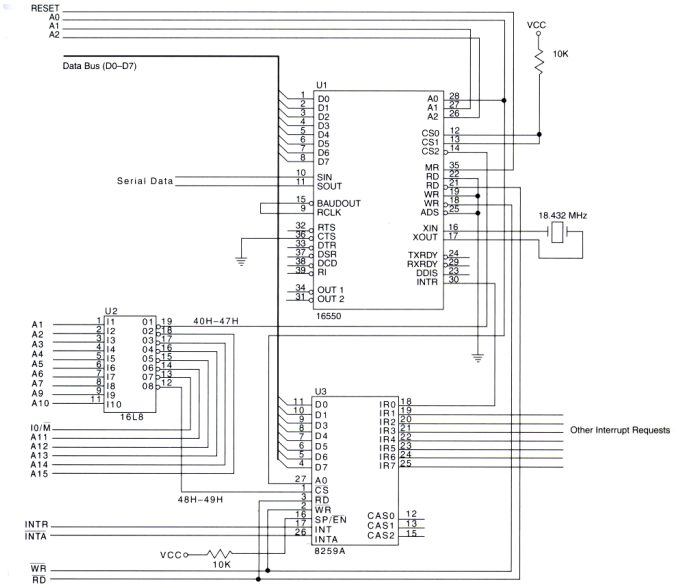
8259A – Interface with 16550 UART

- In the following configuration the 16550 is connected to the 8259A through IR0.
- An interrupt is generated, if enabled through the interrupt control register, when either:
  - The transmitter is ready to send another character.
  - The receiver has received a character.
  - An error is detected while receiving data.
  - A modem interrupt occurs.
- The 16550 is decoded at 40H and 47H.
  - The 8259A is decoded at 48H and 49H.
- Program shows the steps involved in programming both devices.
  - Since the 16550 generates only one interrupt request for each of the above interrupts, the 16550 must be polled.
  - Remember the interrupt identification register of the 16550?

8259A – Interface with 16550 UART



8259A – Interface with 16550 UART



CompE 475 - SDSU

FIGURE 12–21 The 16550 UART interfaced to the 8088 microprocessor through the 8259A.

35

8259A – Interface with 16550 UART

Interrupt Control Register

7	6	5	4	3	2	1	0
0	0	0	0	EM	EL	ET	ER

Enable Reciever Interrupt
0 = disabled
1 = enabled

Enable Transmitter Interrupt
0 = disabled
1 = enabled

Enable Line Interrupt
0 = disabled
1 = enabled

Enable Modem Interrupt
0 = disabled
1 = enabled

Interrupt Identification Register

7	6	5	4	3	2	1	0
0	0	0	0	ID	ID	ID	PN

Interrupt Pending
0 = interrupt pending
1 = no interrupt

Interrupt Identification bits (defined in text)

CompE 475 - SDSU

36

8259A – Interface with 16550 UART

TABLE 12–2 The interrupt control bits of the 16550.

Bit3	*Bit2*	*Bit1*	*Bit0*	*Priority*	*Type*	*Reset Control*
0	0	0	1	—	No interrupt	—
0	1	1	0	1	Receiver error (parity, framing, overrun, or break)	Reset by reading the line register
0	1	0	0	2	Receiver data available	Reset by reading the data
1	1	0	0	2	Character time-out, nothing has been removed from the receiver FIFO for at least four character times	Reset by reading the data
0	0	1	0	3	Transmitter empty	Reset by writing to the transmitter
0	0	0	0	4	Modem status	Reset by reading the modem status

Note: 1 is the highest priority and 4 the lowest.

See textbook for programming examples and other details !