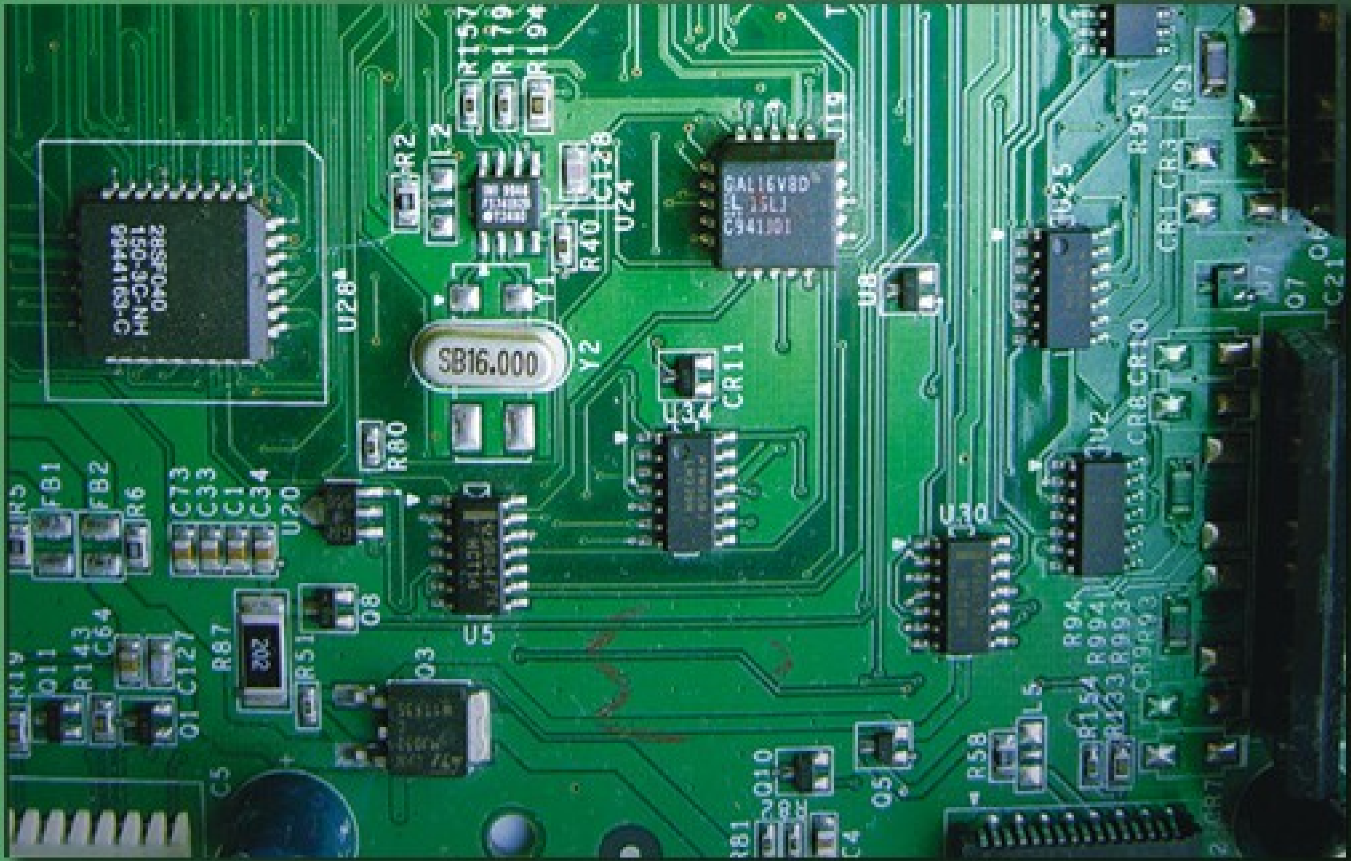
The Intel Microprocessors

8086/8088, 80186/80188, 80286, 80386, 80486 Pentium, Pentium Pro Processor, Pentium II, Pentium 4, and Core2 with 64-bit Extensions

Architecture, Programming, and Interfacing



EIGHTH EDITION

Barry B. Brey

PEARSON

Chapter 12: Interrupts

Introduction

- In this chapter, the coverage of basic I/O and programmable peripheral interfaces is expanded by examining a technique called interrupt-processed I/O.
- An interrupt is a hardware-initiated procedure that interrupts whatever program is currently executing.
- This chapter provides examples and a detailed explanation of the interrupt structure of the entire Intel family of microprocessors.

Chapter Objectives

Upon completion of this chapter, you will be able to:

- Explain the interrupt structure of the Intel family of microprocessors.
- Explain the operation of software interrupt instructions INT, INTO, INT 3, and BOUND.
- Explain how the interrupt enable flag bit (IF) modifies the interrupt structure.
- Describe the function of the trap interrupt flag bit (TF) and the operation of trap-generated tracing.

Chapter Objectives *(cont.)*

Upon completion of this chapter, you will be able to:

- Develop interrupt-service procedures that control lower-speed, external peripheral devices.
- Expand the interrupt structure of the microprocessor by using the 8259A programmable interrupt controller and other techniques.
- Explain the purpose and operation of a real-time clock.

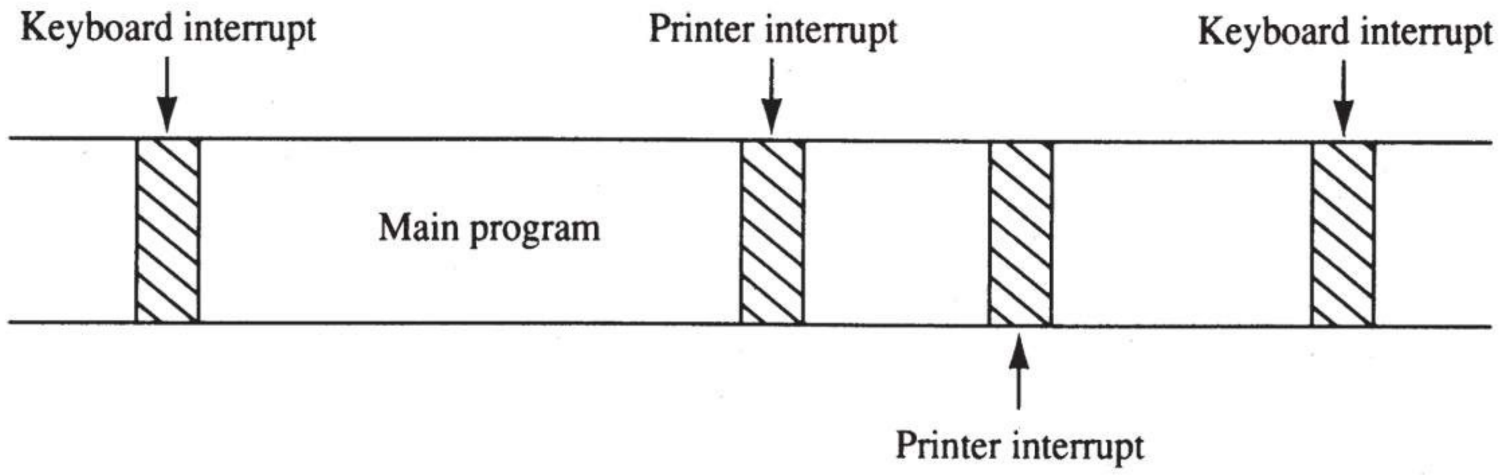
12–1 BASIC INTERRUPT PROCESSING

- This section discusses the function of an interrupt in a microprocessor-based system.
- Structure and features of interrupts available to Intel microprocessors.

The Purpose of Interrupts

- Interrupts are useful when interfacing I/O devices at relatively low data transfer rates, such as keyboard inputs, as discussed in Chapter 11.
- Interrupt processing allows the processor to execute other software while the keyboard operator is thinking about what to type next.
- When a key is pressed, the keyboard encoder debounces the switch and puts out one pulse that interrupts the microprocessor.

Figure 12–1 A time line that indicates interrupt usage in a typical system.



- a time line shows typing on a keyboard, a printer removing data from memory, and a program executing
- the keyboard interrupt service procedure, called by the keyboard interrupt, and the printer interrupt service procedure each take little time to execute

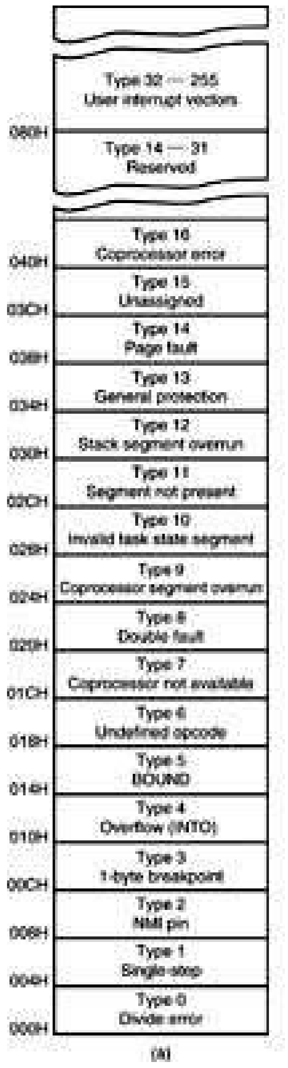
Interrupts

- Intel processors include two hardware pins (INTR and NMI) that request interrupts…
- And one hardware pin (INTA) to acknowledge the interrupt requested through INTR.
- The processor also has software interrupts INT, INTO, INT 3, and BOUND.
- Flag bits IF (interrupt flag) and TF (trap flag), are also used with the interrupt structure and special return instruction IRET
  - IRETD in the 80386, 80486, or Pentium

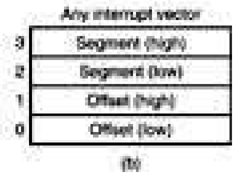
Interrupt Vectors

- Interrupt vectors and the vector table are crucial to an understanding of hardware and software interrupts.
- The **interrupt vector table** is located in the first 1024 bytes of memory at addresses 000000H–0003FFH.
  - contains 256 different four-byte interrupt vectors
- An interrupt vector contains the address (segment and offset) of the interrupt service procedure.

Figure 12–2 (a) The interrupt vector table for the microprocessor and (b) the contents of an interrupt vector.



- the first five interrupt vectors are identical in all Intel processors
- Intel reserves the first 32 interrupt vectors
- the last 224 vectors are user-available
- each is four bytes long in real mode and contains the starting address of the interrupt service procedure.
- the first two bytes contain the offset address
- the last two contain the segment address



Intel Dedicated Interrupts

- **Type 0**
  The **divide error** whenever the result from a division overflows or an attempt is made to divide by zero.
- **Type 1**
  Single-step or trap occurs after execution of each instruction if the trap (TF) flag bit is set.
  - upon accepting this interrupt, TF bit is cleared so the interrupt service procedure executes at full speed

- **Type 2**
  The **non-maskable interrupt** occurs when a logic 1 is placed on the NMI input pin to the microprocessor.
  - non-maskable—it cannot be disabled
- **Type 3**
  A special one-byte instruction (INT 3) that uses this vector to access its interrupt-service procedure.
  - often used to store a breakpoint in a program for debugging

- **Type 4**

  Overflow is a special vector used with the INTO instruction. The INTO instruction interrupts the program if an overflow condition exists.

  - as reflected by the overflow flag (OF)

- **Type 5**
  The **BOUND** instruction compares a register with boundaries stored in the memory.

  If the contents of the register are greater than or equal to the first word in memory and less than or equal to the second word, no interrupt occurs because the contents of the register are within bounds.
  - if the contents of the register are out of bounds, a type 5 interrupt ensues

- **Type 6**
  An **invalid opcode** interrupt occurs when an undefined opcode is encountered in a program.
- **Type 7**
  The **coprocessor not available** interrupt occurs when a coprocessor is not found, as dictated by the machine status word (MSW or CR0) coprocessor control bits.
  - if an ESC or WAIT instruction executes and no coprocessor is found, a type 7 exception or interrupt occurs

- **Type 8**
  A **double fault** interrupt is activated when two separate interrupts occur during the same instruction.
- **Type 9**
  The **coprocessor segment overrun** occurs if the ESC instruction (coprocessor opcode) memory operand extends beyond offset address FFFFH in real mode.

- **Type 10**
  An **invalid task state segment** interrupt occurs in the protected mode if the TSS is invalid because the segment limit field is not 002BH or higher.
  - usually because the TSS is not initialized
- **Type 11**
  The **segment not present** interrupt occurs when the protected mode P bit (P = 0) in a descriptor indicates that the segment is not present or not valid.

- **Type 12**
  A **stack segment overrun** occurs if the stack segment is not present (P = 0) in the protected mode or if the limit of the stack segment is exceeded.

- **Type 13**
  The **general protection fault** occurs for most protection violations in 80286–Core2 in protected mode system.

  These errors occur in Windows as general protection faults.

  A list of these protection violations follows.

- **Type 13** protection violations *(cont.)*
  - (a) Descriptor table limit exceeded
  - (b) Privilege rules violated
  - (c) Invalid descriptor segment type loaded
  - (d) Write to code segment that is protected
  - (e) Read from execute-only code segment
  - (f) Write to read-only data segment
  - (g) Segment limit exceeded

- (h) CPL = IOPL when executing CLTS (Page 814, Clear task switched flag), HLT (Halt), LGDT (Load global descriptor table, pp.834), LIDT (Load interrupt descriptor table), LLDT (Load local descriptor), LMSW (Load machine status word), or LTR (Load task register)
- (i) CPL > IOPL when executing CLI (Clear interrupt flag), IN (Input data from port), INS (Input string form port), LOCK (Lock the bus), OUT (Output data to port), OUTS (Output string to port), and STI (Set interrupt flag)

CPL (Current privilege level); IOPL (Input/output privilege level)

- **Type 14**
  Page fault interrupts occur for any page fault memory or code access in 80386, 80486, and Pentium–Core2 processors.
- **Type 16**
  Coprocessor error takes effect when a coprocessor error (ERROR = 0) occurs for ESCape or WAIT instructions for 80386, 80486, and Pentium–Core2 only.

- **Type 17**
  Alignment checks indicate word and doubleword data are addressed at an odd memory location (or incorrect location, in the case of a doubleword).
  – interrupt is active in 80486 and Pentium–Core2
- **Type 18**
  A **machine check** activates a system memory management mode interrupt in Pentium–Core2.

Interrupt Instructions: BOUND, INTO, INT, INT 3, and IRET

- Five software interrupt instructions are available to the microprocessor:
- INT and INT 3 are very similar.
- BOUND and INTO are conditional.
- IRET is a special interrupt return instruction.

- **BOUND** : Check array against boundary
- **INTO: Interrupt on overflow**
- **INT: Interrupt**
- **INT 3: Interrupt 3**
- **IRET: Return from interrupt**

- BOUND has two operands, and compares a register with two words of memory data.
- INTO checks or tests the overflow flag (O).
  - If O = 1, INTO calls the procedure whose address is stored in interrupt vector type 4
  - If O = 0, INTO performs no operation and the next sequential program instruction executes
- The INT *n* instruction calls the interrupt service procedure at the address represented in vector number *n*.

- INT 3 instruction is often used as a breakpoint-interrupt because it is easy to insert a one-byte instruction into a program.
  - breakpoints are often used to debug software
- The IRET instruction is a special return instruction used to return for both software and hardware interrupts.
  - much like a far RET, it retrieves the return address from the stack

Operation of a Real Mode Interrupt

- When the processor completes executing the current instruction, it determines whether an interrupt is active by checking:
  - (1) instruction executions
  - (2) single-step
  - (3) NMI
  - (4) coprocessor segment overrun
  - (5) INTR
  - (6) INT instructions in the order presented

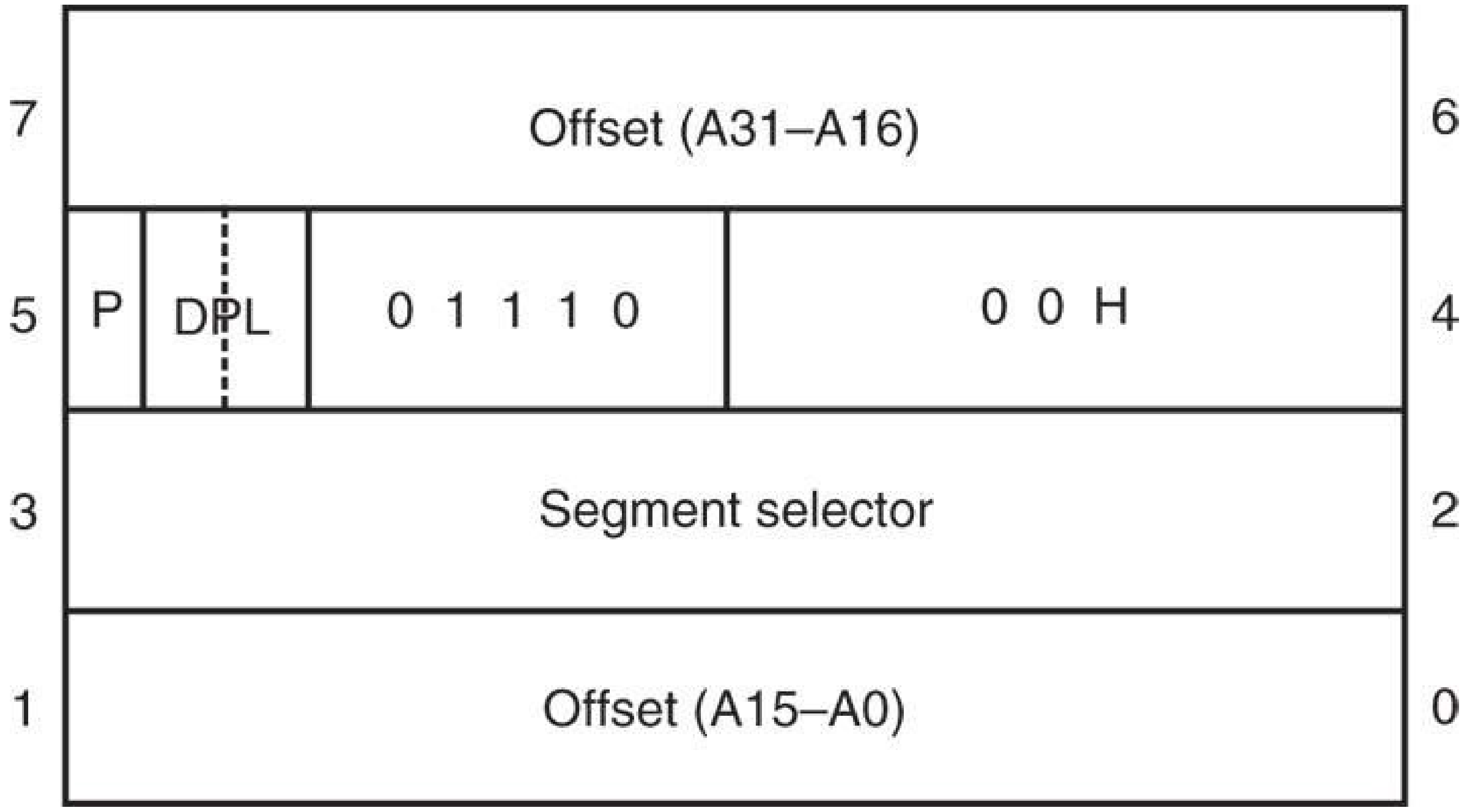
- If one or more are present:
  - 1. Flag register contents are pushed on the stack
  - 2. Interrupt (IF) & trap (TF) flags clear, disabling the INTR pin and trap or single-step feature
  - 3. Contents of the code segment register (CS) are pushed onto the stack
  - 4. Contents of the instruction pointer (IP) are pushed onto the stack
  - 5. Interrupt vector contents are fetched and placed into IP and CS so the next instruction executes at the interrupt service procedure addressed by the vector

Operation of a Protected Mode Interrupt

- In protected mode, interrupts have the same assignments as real mode.
  - the interrupt vector table is different
- In place of interrupt vectors, protected mode uses a set of 256 interrupt descriptors stored in an interrupt descriptor table (IDT).
  - the table is 256×8 (2K) bytes long
  - each descriptor contains eight bytes

- The interrupt descriptor table is located at any memory location in the system by the interrupt descriptor table address register (IDTR).
- Each IDT entry contains the address of the interrupt service procedure
  - in the form of a segment selector and a 32-bit offset address
  - also contains the P bit (present) and DPL bits to describe the privilege level of the interrupt
- Fig 12–3 shows interrupt descriptor contents.

Figure 12–3 The protected mode interrupt descriptor.



Interrupt Flag Bits

- The interrupt flag (IF) and the trap flag (TF) are both cleared after the contents of the flag register are stacked during an interrupt.
- the contents of the flag register and the location of IF and TF are shown here
  - when IF is set, it *allows* the INTR pin to cause an interrupt
  - when IF is cleared, it *prevents* the INTR pin from causing an interrupt

- when TF = 1, it causes a trap interrupt (type 1) to occur after each instruction executes
- Trap is often called a *single-step*
- when TF = 0, normal program execution occurs
- the interrupt flag is set and cleared by the STI and CLI instructions, respectively
- the contents of the flag register and the location of IF and TF are shown here

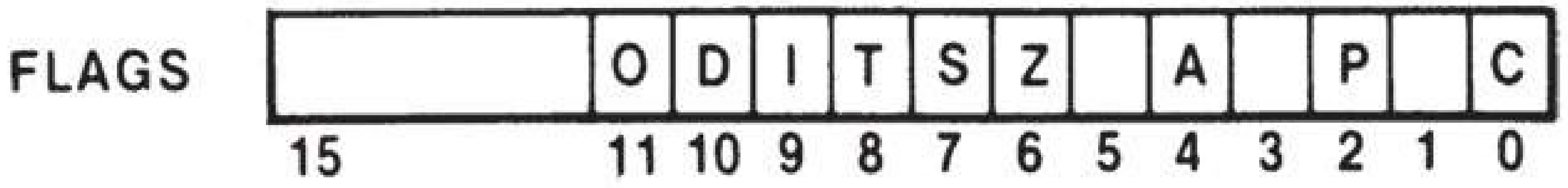


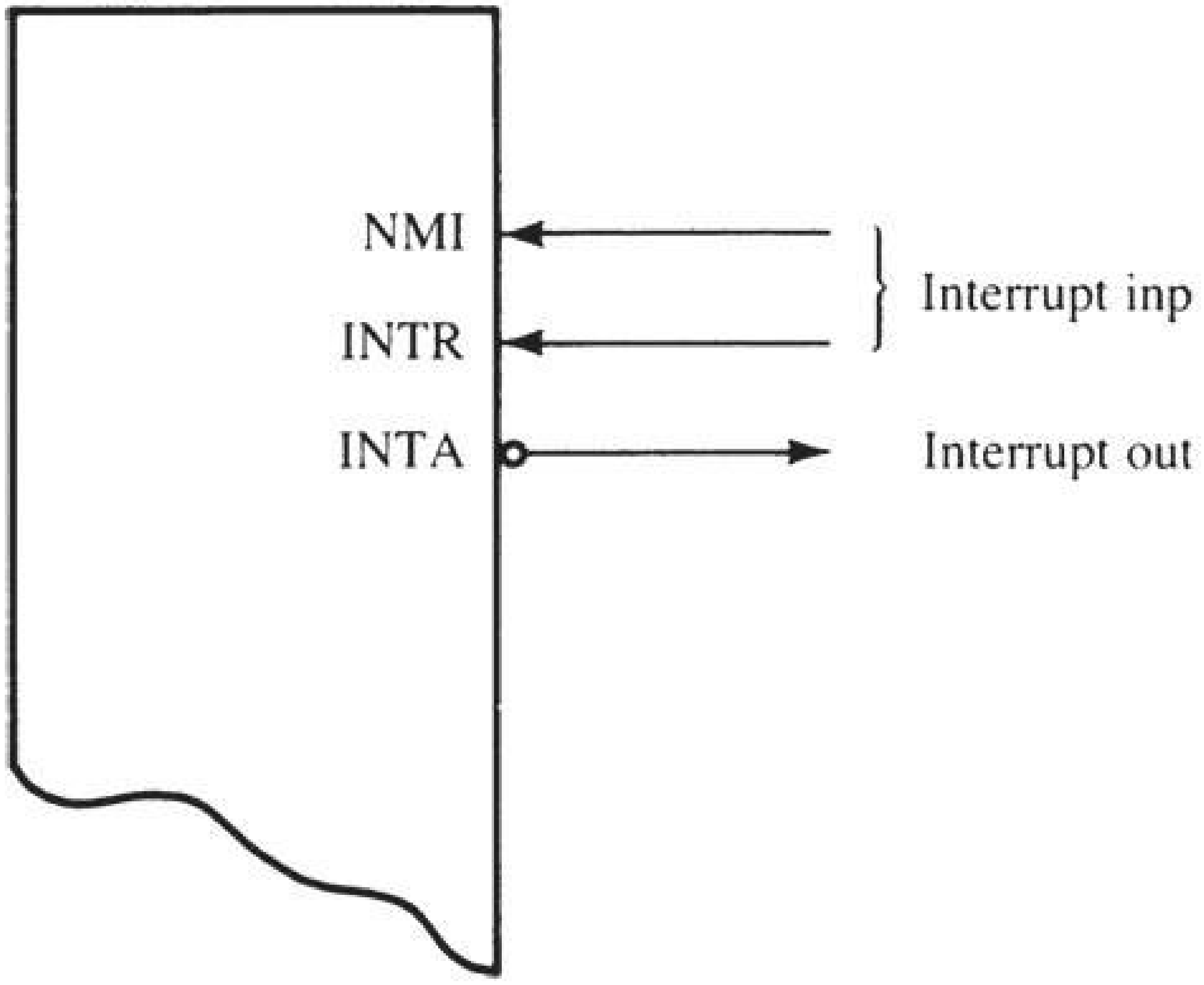
Figure 12–4 The flag register. (Courtesy of Intel Corporation.)

12–2 HARDWARE INTERRUPTS

- The two processor hardware interrupt inputs:
  - non-maskable interrupt (NMI)
  - interrupt request (INTR)
- When NMI input is activated, a type 2 interrupt occurs
  - because NMI is internally decoded
- The INTR input must be externally decoded to select a vector.

- Any interrupt vector can be chosen for the INTR pin, but we usually use an interrupt type number between 20H and FFH.
- Intel has reserved interrupts 00H - 1FH for internal and future expansion.
- $\overline{\text{INTA}}$ is also an interrupt pin on the processor.
  - it is an output used in response to INTR input to apply a vector type number to the data bus connections D_7–D_0
- Figure 12–5 shows the three user interrupt connections on the microprocessor.

Figure 12–5 The interrupt pins on all versions of the Intel microprocessor.



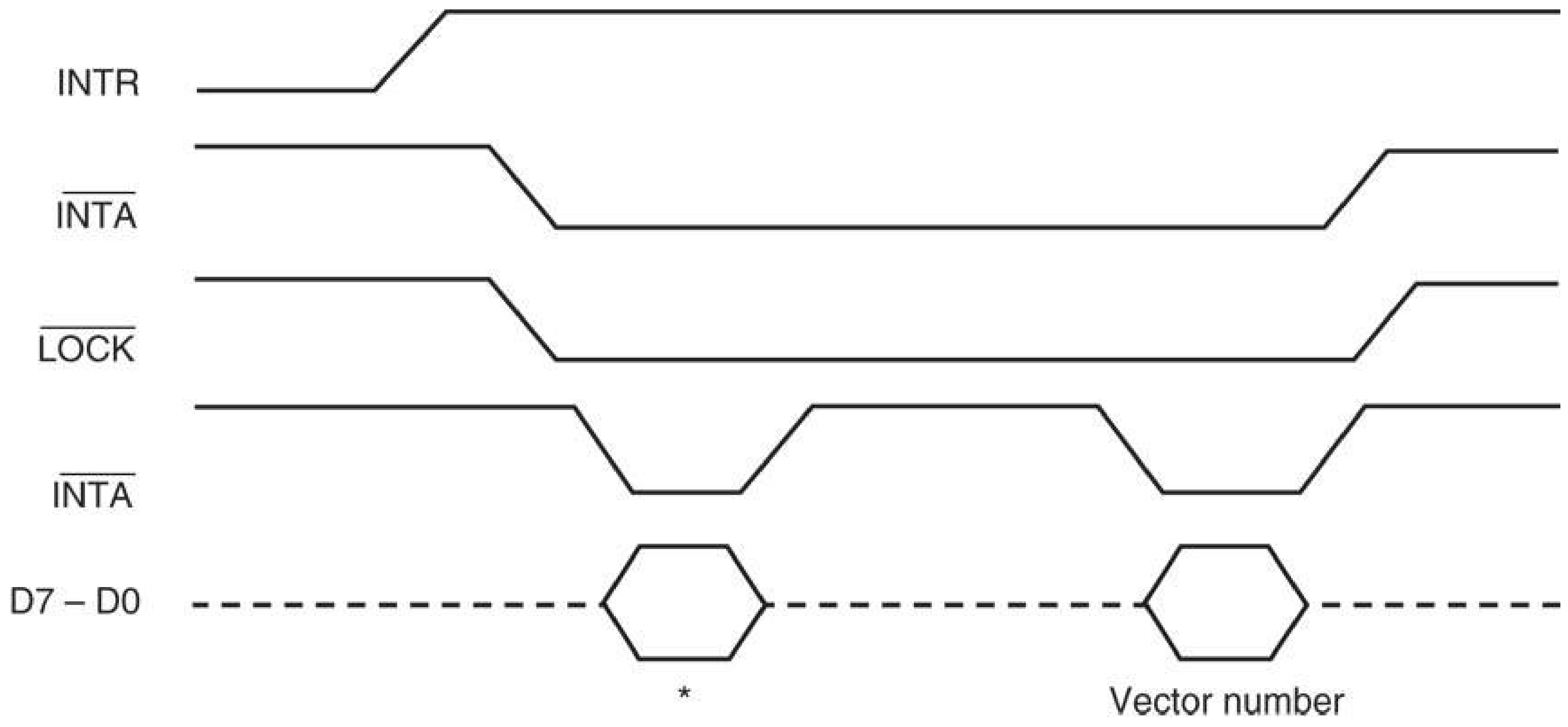
- The **non-maskable interrupt** (NMI) is an edge-triggered input that requests an interrupt on the positive edge (0-to-1 transition).
  - after a positive edge, the NMI pin must remain logic 1 until recognized by the microprocessor
  - before the positive edge is recognized, NMI pin must be logic 0 for at least two clocking periods
- The NMI input is often used for parity errors and other major faults, such as power failures.
  - power failures are easily detected by monitoring the AC power line and causing an NMI interrupt whenever AC power drops out

INTR and $\overline{\text{INTA}}$

- The interrupt request input (INTR) is level-sensitive, which means that it must be held at a logic 1 level until it is recognized.
  - INTR is set by an external event and cleared inside the interrupt service procedure
- INTR is automatically disabled once accepted.
  - re-enabled by IRET at the end of the interrupt service procedure
- 80386–Core2 use IRETD in protected mode.
  - in 64-bit protected mode, IRETQ is used

- The processor responds to INTR by pulsing INTA output in anticipation of receiving an interrupt vector type number on data bus connections D_7–D_0.
- Fig 12–8 shows the timing diagram for the INTR and pins of the microprocessor.
- Two INTA pulses generated by the system insert the vector type number on the data bus.

Figure 12–8 The timing of the INTR input and $\overline{\text{INTA}}$ output. *This portion of the data bus is ignored and usually contains the vector number.



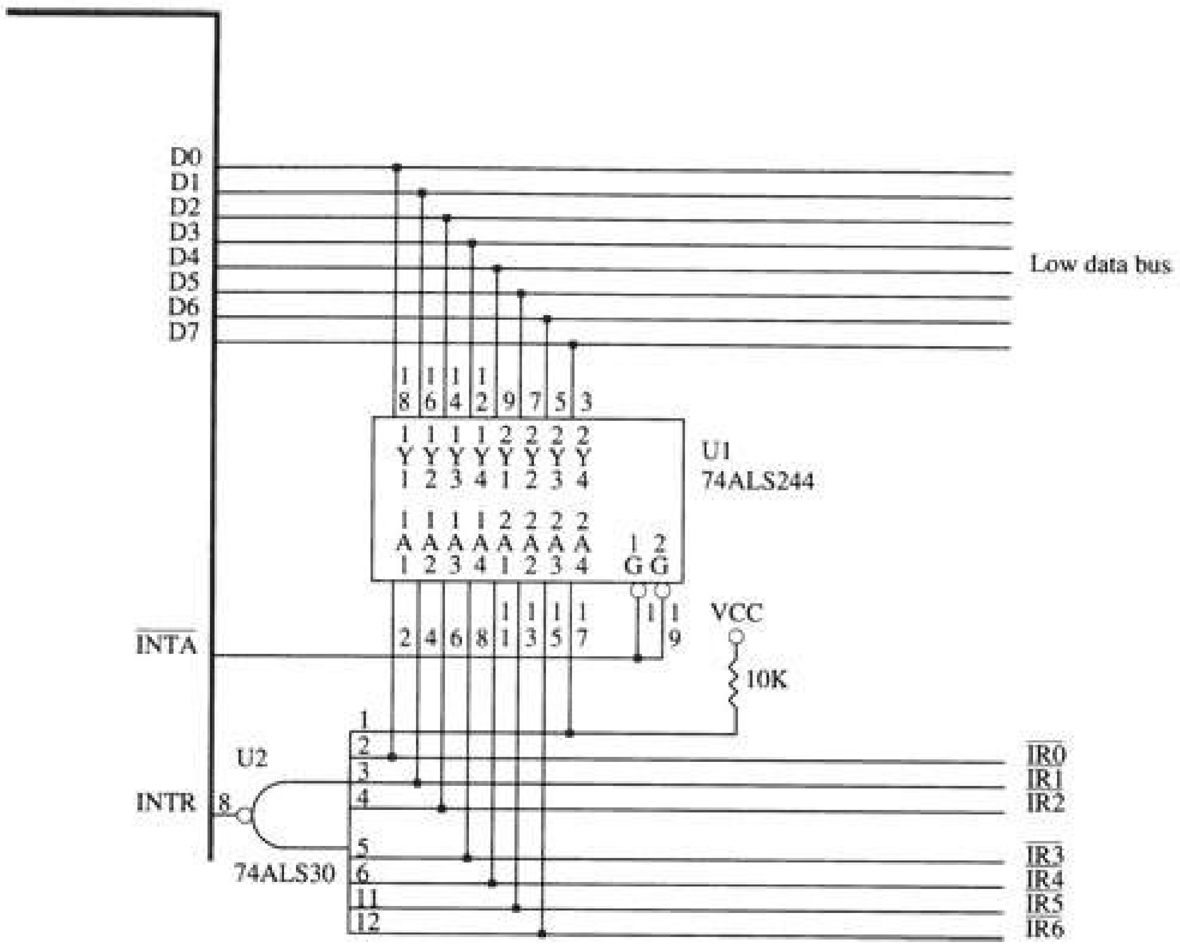
12–3 EXPANDING THE INTERRUPT STRUCTURE

- This covers three common methods of expanding the interrupt structure of the processor.
- It is possible to expand the INTR input so it accepts seven interrupt inputs.
- Also explained is how to "daisy-chain" interrupts by software polling.

Using the 74ALS244 to Expand Interrupts

- The modification shown in Fig 12–13 allows the circuit of Fig 12–10 to accommodate up to seven additional interrupt inputs.
- The only hardware change is the addition of an eight-input NAND gate, which provides the INTR signal to the microprocessor when any of the IR inputs becomes active.

Figure 12–13 Expanding the INTR input from one to seven interrupt request lines.



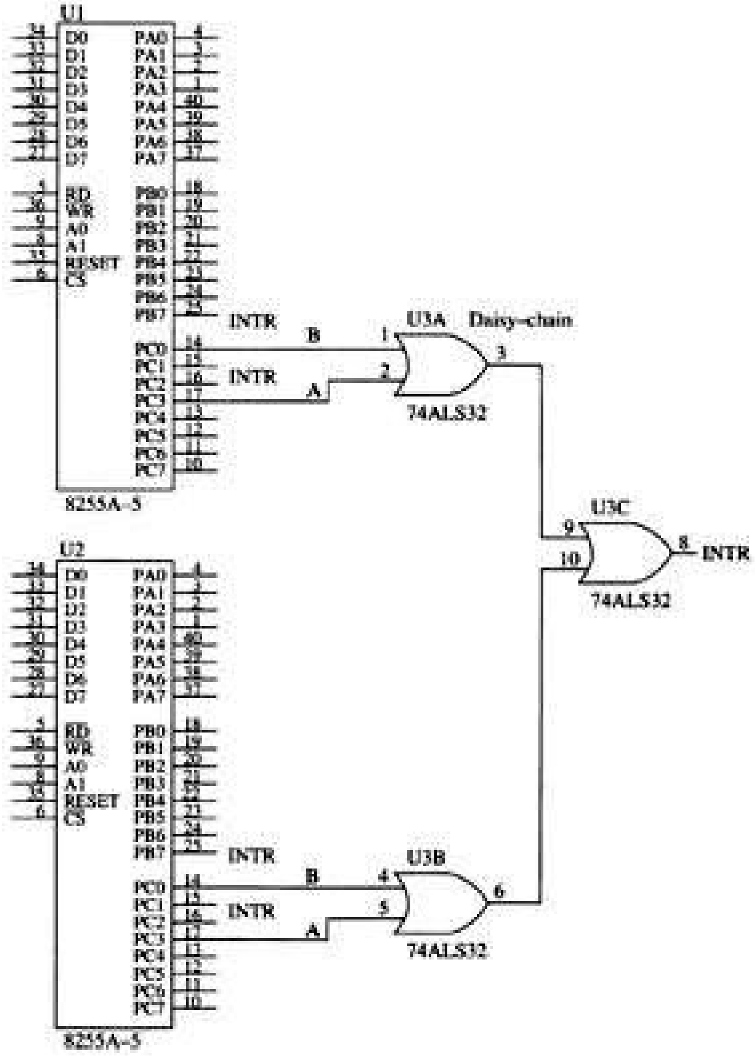
Operation

- If any of the $\overline{IR}$ inputs becomes logic 0, the output of the NAND gate goes to logic 1 and requests an interrupt through the INTR input.
- The interrupt vector that is fetched during the pulse depends on which interrupt request line becomes active.
  - Table 12–1 shows the interrupt vectors used by a single interrupt request input
- If two or more interrupt requests are active active, a new interrupt vector is generated.

Daisy-Chained Interrupt

- Expansion by a daisy-chained interrupt is in many ways better than using the 74ALS244.
  - because it requires only one interrupt vector
- Fig 12–14 shows a two 82C55 peripheral interfaces with their four INTR outputs daisy-chained and connected to the single INTR input of the processor.
- If any interrupt output becomes logic 1, so does INTR input, causing an interrupt.

Figure 12–14 Two 82C55 PIAs connected to the INTR outputs are daisy-chained to produce an INTR signal.

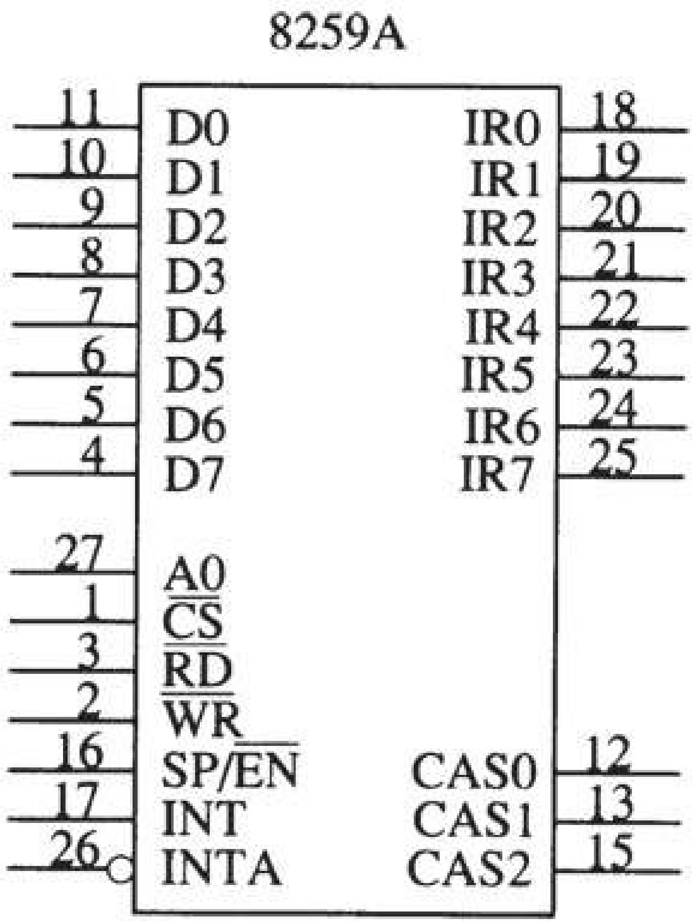


– any INTR output from the two 82C55s will cause the INTR pin on the processor to go high, requesting an interrupt
– The task of locating which INTR output became active is up to the interrupt service procedure, which must poll the 82C55s to determine which output caused the interrupt

12–4 8259A PROGRAMMABLE INTERRUPT CONTROLLER

- 8259A (PIC) adds eight vectored priority encoded interrupts to the microprocessor.
- Expandable, without additional hardware, to accept up to 64 interrupt requests.
  - requires a master 8259A & eight 8259A slaves
- A pair of these controllers still resides and is programmed as explained here in the latest chip sets from Intel and other manufacturers.

General Description of the 8259A



- 8259A is easy to connect to the microprocessor
- all of its pins are direct connections except the $\overline{CS}$ pin, which must be decoded, and the $\overline{WR}$ pin, which must have an I/O bank write pulse

Figure 12–15 The pin-out of the 8259A programmable interrupt controller (PIC).

8259A Pin-Outs

D_0–D_7

- The bidirectional **data connections** are normally connected to the data bus on the microprocessor.

IR_0–IR_7

- **Interrupt request inputs** are used to request an interrupt and to connect to a slave in a system with multiple 8259As.

$\overline{\text{WR}}$

- The **write input** connects to write strobe signal ($\overline{\text{IOWC}}$) on the microprocessor.

$\overline{\text{RD}}$

- The **read input** connects to the $\overline{\text{IORC}}$ signal.

INT

- The **interrupt output** connects to the INTR pin on the processor from the master and is connected to a master IR pin on a slave.

$\overline{\text{INTA}}$

- **Interrupt acknowledge** is an input that connects to the $\overline{\text{INTA}}$ signal on the system. In a system with a master and slaves, only the master $\overline{\text{INTA}}$ signal is connected.

A_0

- The **A_0 address input** selects different command words within the 8259A.

$\overline{\text{CS}}$

- **Chip select** enables the 8259A for programming and control.

CAS_0–CAS_2

- The **cascade lines** are used as outputs from the master to the slaves for cascading multiple 8259As in a system.

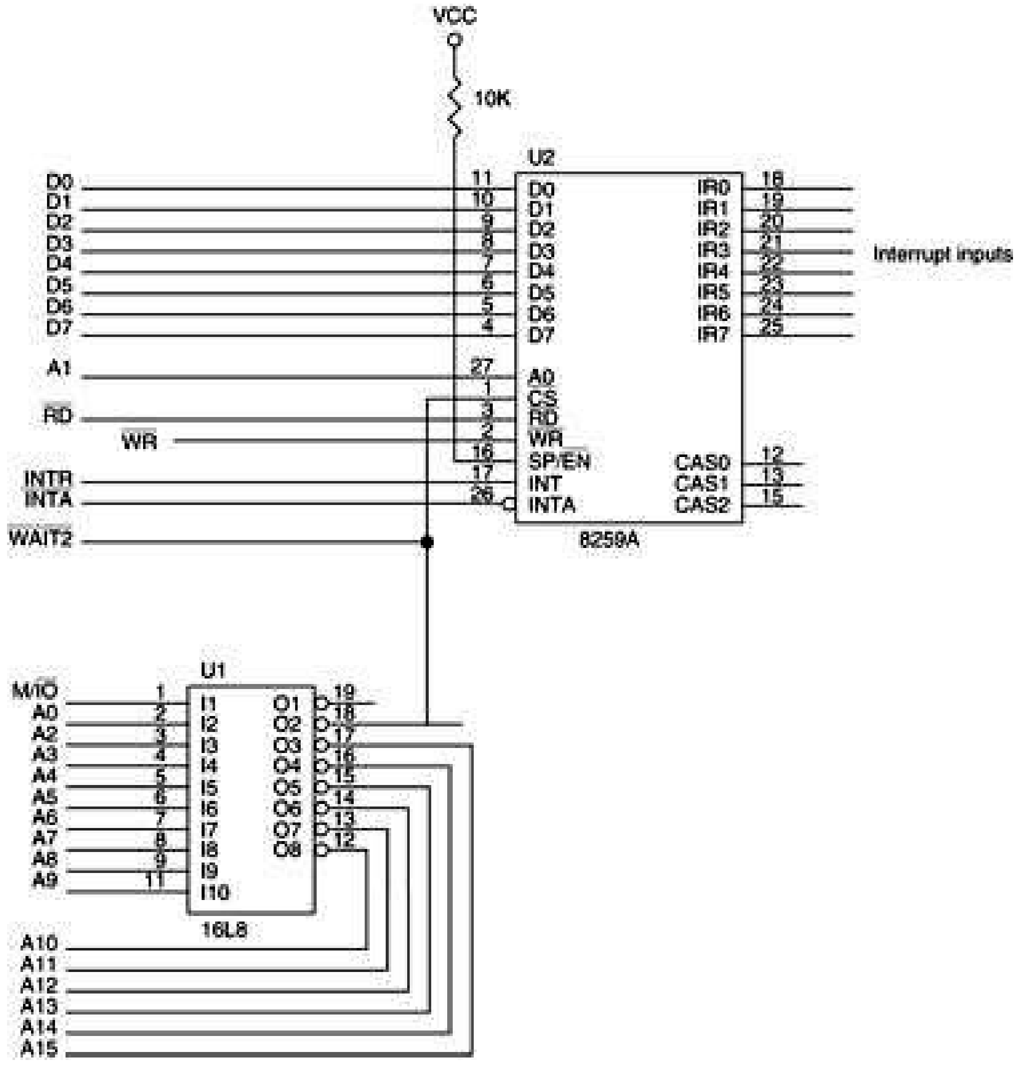
SP/$\overline{\text{EN}}$

- **Slave program/enable buffer** is a dual-function pin.
  - when the 8259A is in buffered mode, this output controls the data bus transceivers in a large microprocessor-based system
  - when the 8259A is not in the buffered mode, this pin programs the device as a master (1) or a slave (0)

Connecting a Single 8259A

- Fig 12–16 shows a single 8259A connected to the microprocessor.
- The 8259A is decoded at I/O ports 0400H and 0401H by the PLD.
- The 8259A requires four wait states for it to function properly with a 16 MHz 80386SX
  - more for some other versions of the Intel microprocessor family

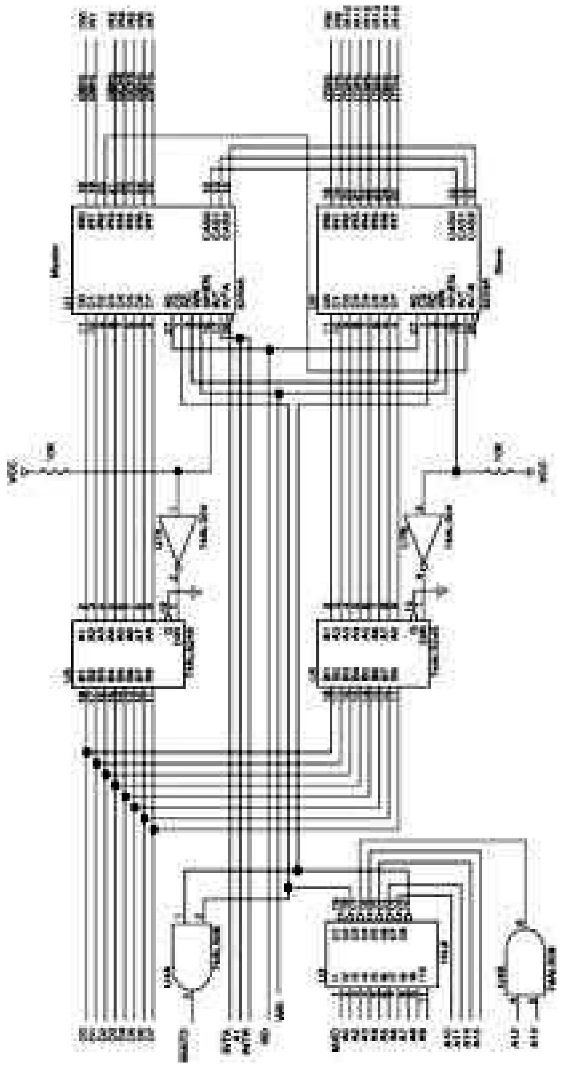
Figure 12–16 An 8259A interfaced to the 8086 microprocessor.



Cascading Multiple 8259As

- Fig 12–17 shows two 8259As connected in a way often found in the ATX-style system.
  - XT- or PC-style computers use a single 8259A controller at interrupt vectors 08H–0FH
- The ATX-style computer uses interrupt vector 0AH as a cascade input from a second 8259A located at vectors 70H through 77H.
- Appendix A contains a table that lists the functions of all the interrupt vectors used.

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.



- this circuit uses vectors 08H–0FH & I/O ports 0300H & 0302H for U1, the master
- vectors 70H–77H & I/O ports 0304H & 0306H for U2, the slave
- data bus buffers to illustrate the use of the SP/$\overline{\text{EN}}$ pin on 8259A
- these buffers are used only in very large systems with many devices on their data bus connections
- seldom found in actual practice

8259A Block Diagram

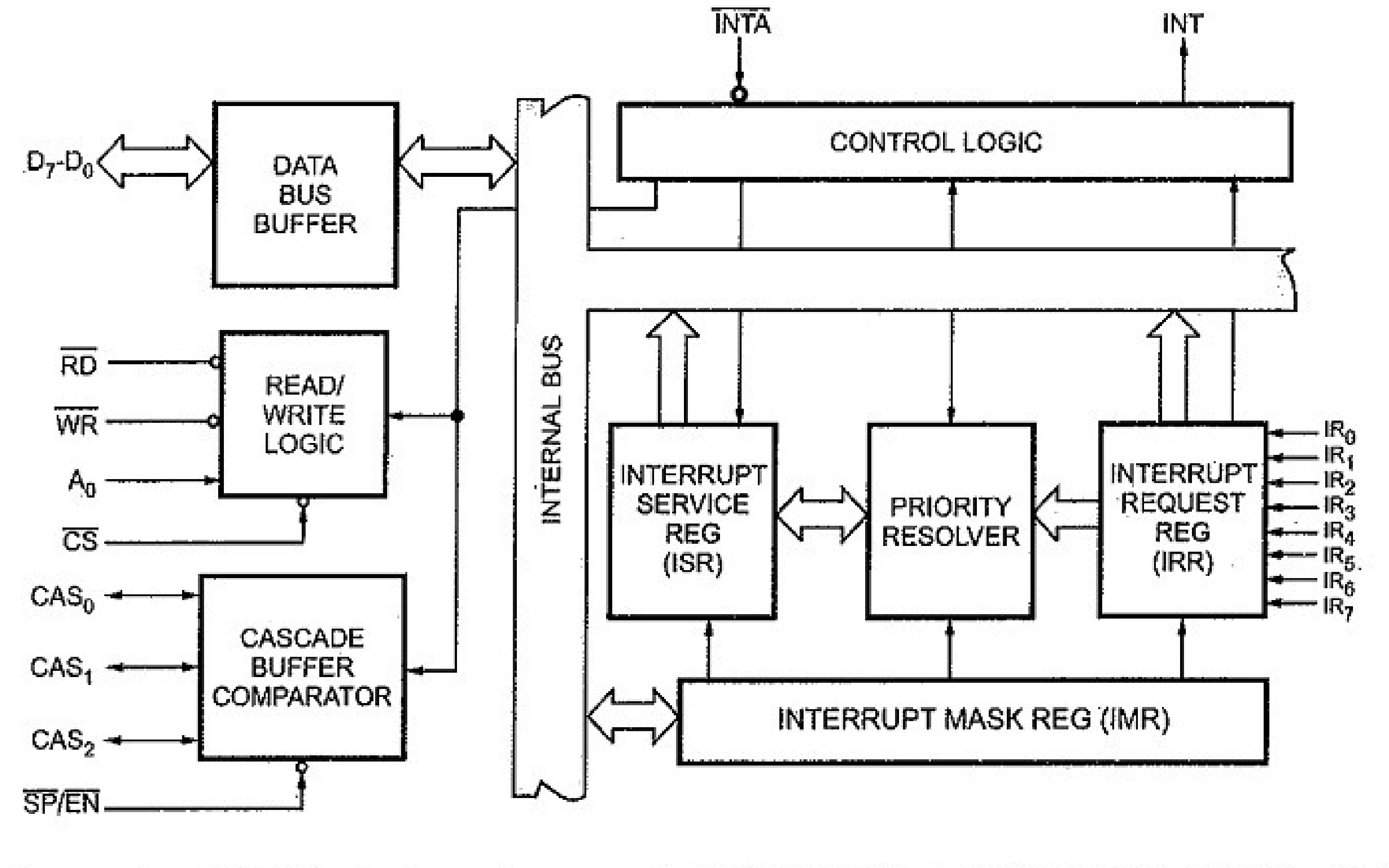


Fig. 14.71 Block diagram of 8259A

Programming the 8259A

- 8259A is programmed by initialization and operation command words.
- **Initialization command words** (ICWs) are programmed before the 8259A is able to function in the system.
  - dictate the basic operation of the 8259A
- **Operation command words** (OCWs) are programmed during the normal course of operation.
  - OCWs control the operation of the 8259A

Initialization Command Words

- the four initialization command words (ICWs) are selected when the A_0 pin is logic 1
- if a single 8259A is used in a system, ICW_1, ICW_2, and ICW_4 must be programmed when powered up
- if programmed in cascade mode by ICW_1, then ICW_3 must be programmed

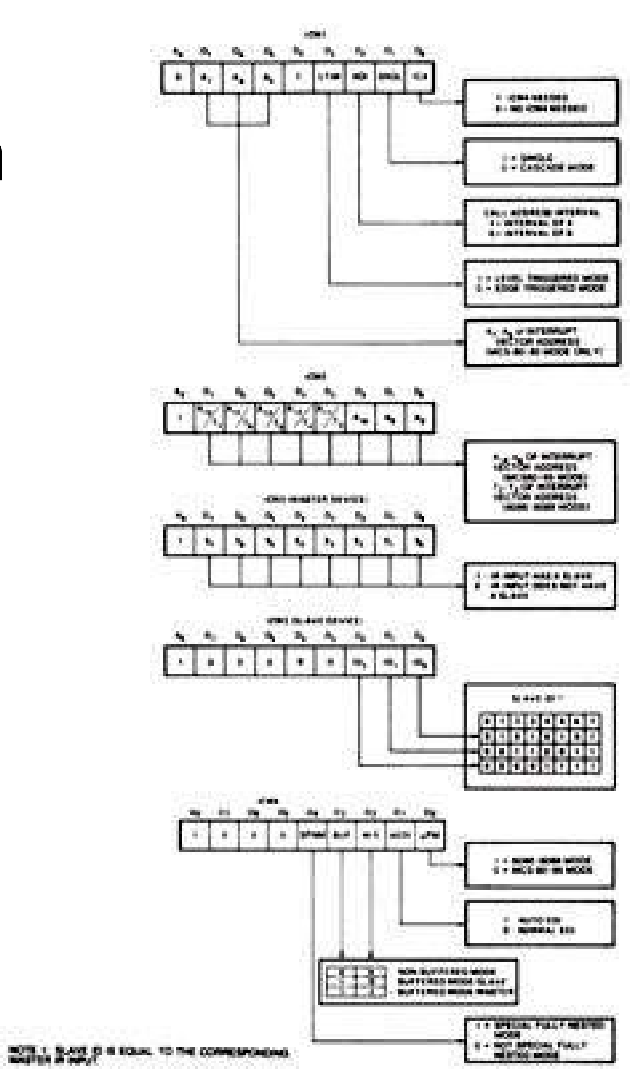


Figure 12–18 The 8259A initialization command words (ICWs). (Courtesy of Intel Corporation.)

ICW Descriptions - ICW_1

- **ICW_1** programs basic operation of the 8259A.
  - selects single or cascade operation by programming the SNGL bit
  - if cascade operation is selected, ICW_3 must also be programmed
  - the LTIM bit determines whether interrupt request inputs are positive edge-triggered or level-triggered

ICW Descriptions - ICW$_2$

- **ICW$_2$** selects the vector number used with the interrupt request inputs.
  - if programming 8259A so it functions at vector locations 08H–0FH, place 08H into this command word
  - if programming for vectors 70H–77H, place 70H in this ICW

ICW Descriptions - ICW_3

- **ICW_3** is used only when ICW_1 indicates the system is operated in cascade mode.
  - this ICW indicates where the slave is connected to the master. (In Figure 12–18 a slave was connected to IR_2)
  - to program ICW_3 for this connection, in both master and slave, place 04H in ICW_3
  - if two slaves connected using IR_0 and IR_1, the master is programmed with an ICW_3 of 03H
  - one slave is programmed with an ICW_3 of 01H and the other with an ICW_3 of 02H

ICW Descriptions - ICW$_4$

- **ICW$_4$** is programmed for use with the 8086–Pentium 4 processors, and the rightmost bit must be logic 1 to select operation with them.
- Remaining bits are programmed as follows:
  - **SFNM**—Selects the special fully nested mode of operation if logic 1 is placed in this bit
  - **BUF** and **M/S**—Buffered and master slave are used together to select buffered operation or nonbuffered operation for the 8259A as a master or a slave

ICW Descriptions - ICW$_4$ (*cont.*)

- Bit programming:
  - **AEOI**—Selects automatic or normal end of interrupt. The EOI commands of OCW2 are used only if AEOI mode is not selected by ICW$_4$
  - if AEOI is selected, the interrupt automatically resets the interrupt request bit and does not modify priority
  - this is the preferred mode of operation for the 8259A and reduces the length of the interrupt service procedure

Operation Command Words

- used to direct 8259A operation once iprogrammed with the ICW
- OCWs are selected when the A_0 pin is at logic 0 level
- except OCW_1, which is selected when A_0 is logic 1
- shown here are the binary bit patterns for all three operation command words of the 8259A

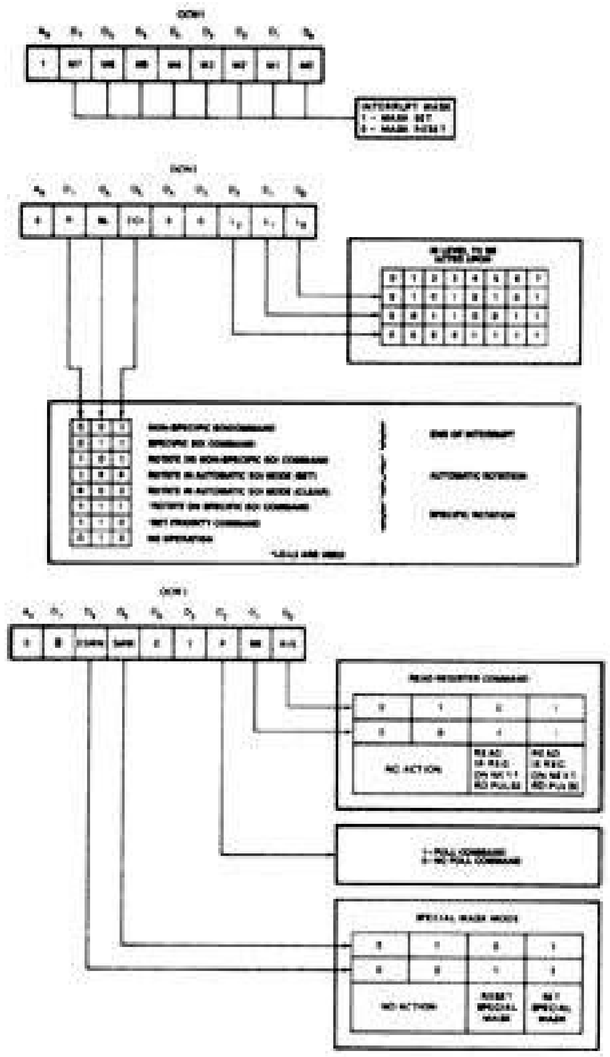


Figure 12–19 The 8259A operation command words (OCWs). (Courtesy of Intel Corporation.)

OCW Descriptions - OCW_1

- **OCW_1** is used to set and read the interrupt mask register.
  - when a mask bit is set, it will turn off (mask) the corresponding interrupt input
  - the mask register is read when OCW_1 is read
  - because the state of the mask bits is unknown when 8259A is first initialized, OCW_1 must be programmed after programming the ICW upon initialization

OCW Descriptions - OCW_2

- **OCW_2** is programmed only when the AEOI mode is not selected for the 8259A.
- Selects the way 8259A responds to an interrupt. Modes are listed as follows:
  - **Nonspecific End-of-Interrupt**—Command sent by the interrupt service procedure to signal the end of the interrupt
  - 8259A determines which interrupt level was active and resets the correct interrupt status register bit
  - resetting the bit allows the interrupt to take action again or a lower priority interrupt to take effect

OCW Descriptions - OCW$_2$ (*cont.*)

- **OCW$_2$** interrupt modes:
  - **Specific End-of-Interrupt**—A command that allows a specific interrupt request to be reset
  - exact position determined with bits L_2–L_0 of OCW$_2$

OCW Descriptions - OCW_2 (*cont.*)

- **OCW_2** interrupt modes:
  - **Rotate-on-Nonspecific EOI**—functions exactly like the Nonspecific End-of-Interrupt command, except it rotates interrupt priorities after resetting the interrupt status register bit
  - the level reset by this command becomes the lowest priority interrupt
  - if IR_4 was just serviced by this command, it becomes the lowest priority interrupt input and IR_5 becomes the highest priority

OCW Descriptions - OCW_2 (*cont.*)

- **OCW_2** interrupt modes:
  - **Rotate-on-Automatic EOI**—A command that selects automatic EOI with rotating priority
  - must only be sent to the 8259A once if this mode is desired.
  - if this mode must be turned off, use the clear command

OCW Descriptions - OCW$_2$ (*cont.*)

- **OCW$_2$** interrupt modes:
  - **Rotate-on-Specific EOI**—Functions as the specific EOI, except that it selects rotating priority.
  - **Set priority**—Allows the programmer to set the lowest priority interrupt input using the L_2–L_0 bits.

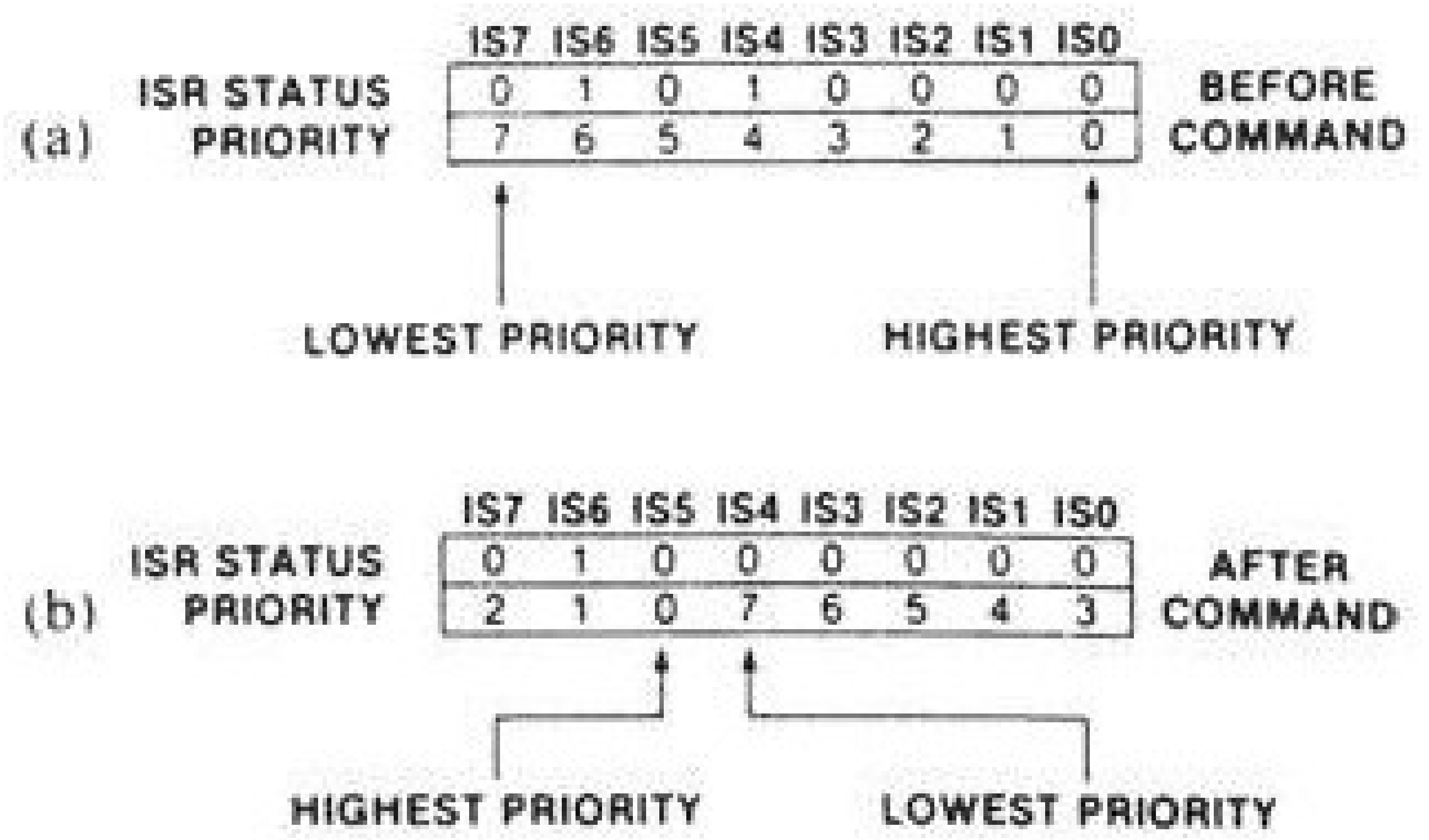
OCW Descriptions - OCW_3

- **OCW_3** selects register to be read, operation of the special mask register & poll command.
  - if polling is selected, the P bit must be set and then output to the 8259A; the next read operation will read the poll word
  - the rightmost three bits of the word indicate the active interrupt request with the highest priority
  - the leftmost bit indicates if there is an interrupt and must be checked to determine whether the rightmost three bits contain valid information

Status Register

- Three status registers are readable in 8259A:
- Interrupt request register (IRR).
  - an 8-bit register that indicates which interrupt request inputs are active
- In-service register (ISR).
  - an 8-bit register that contains the level of the interrupt being serviced
- interrupt mask register (IMR).
  - an 8-bit register that holds the interrupt mask bits and indicates which interrupts are masked off

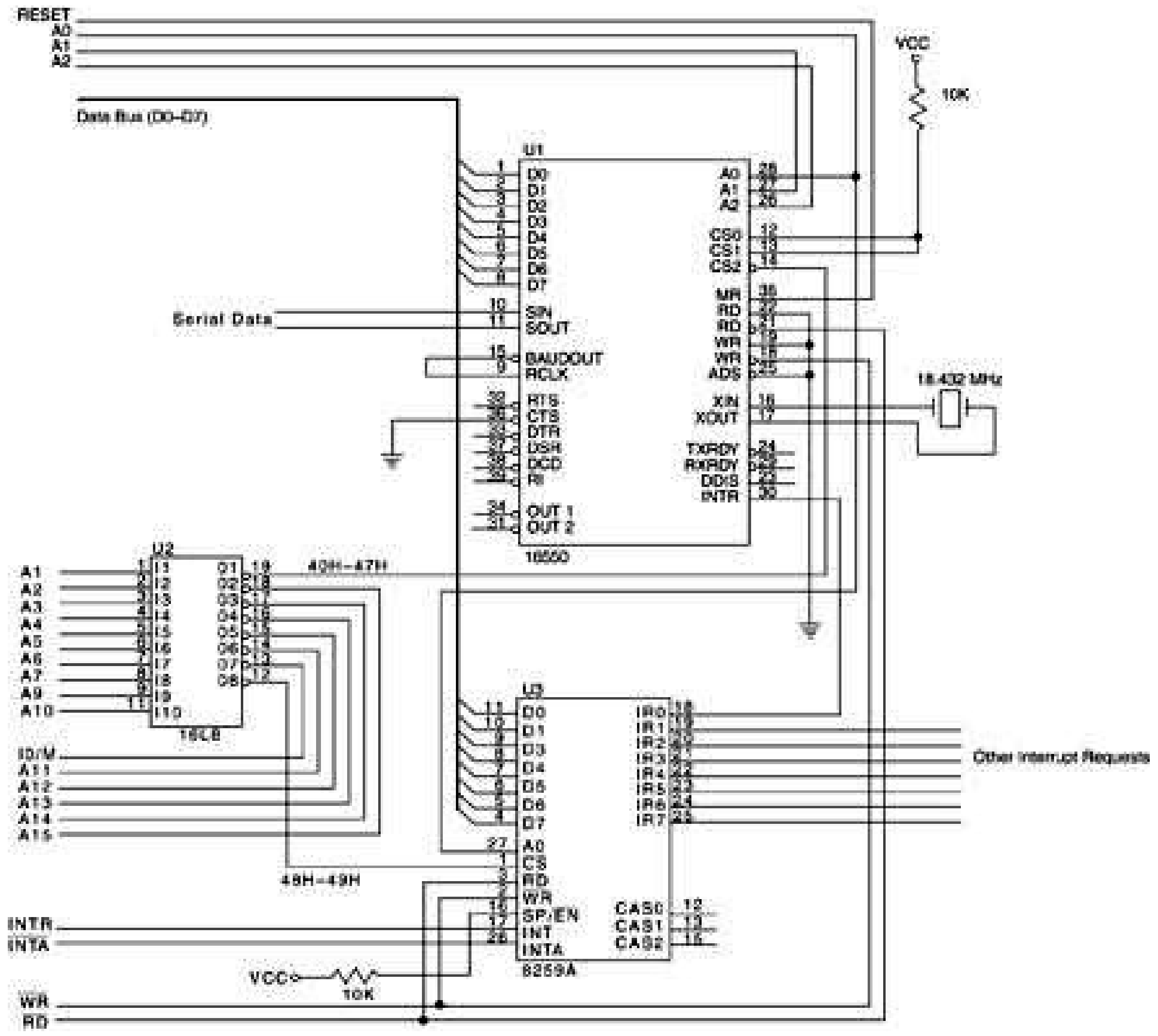
Figure 12–20 The 8259A in-service register (ISR). (a) Before IR_4 is accepted and (b) after IR_4 is accepted. (Courtesy of Intel Corporation.)



8259A Programming Example

- Fig 12–21 shows 8259A connected to a 16550 programmable communications controller.
  - the INTR pin from the 16550 is connected to the programmable interrupt controller’s interrupt request input IR_0
- The 16550 is decoded at I/O ports 40H and 47H, and the 8259A is decoded at 8-bit I/O ports 48H and 49H.
- Both are interfaced to the data bus of an 8088.

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

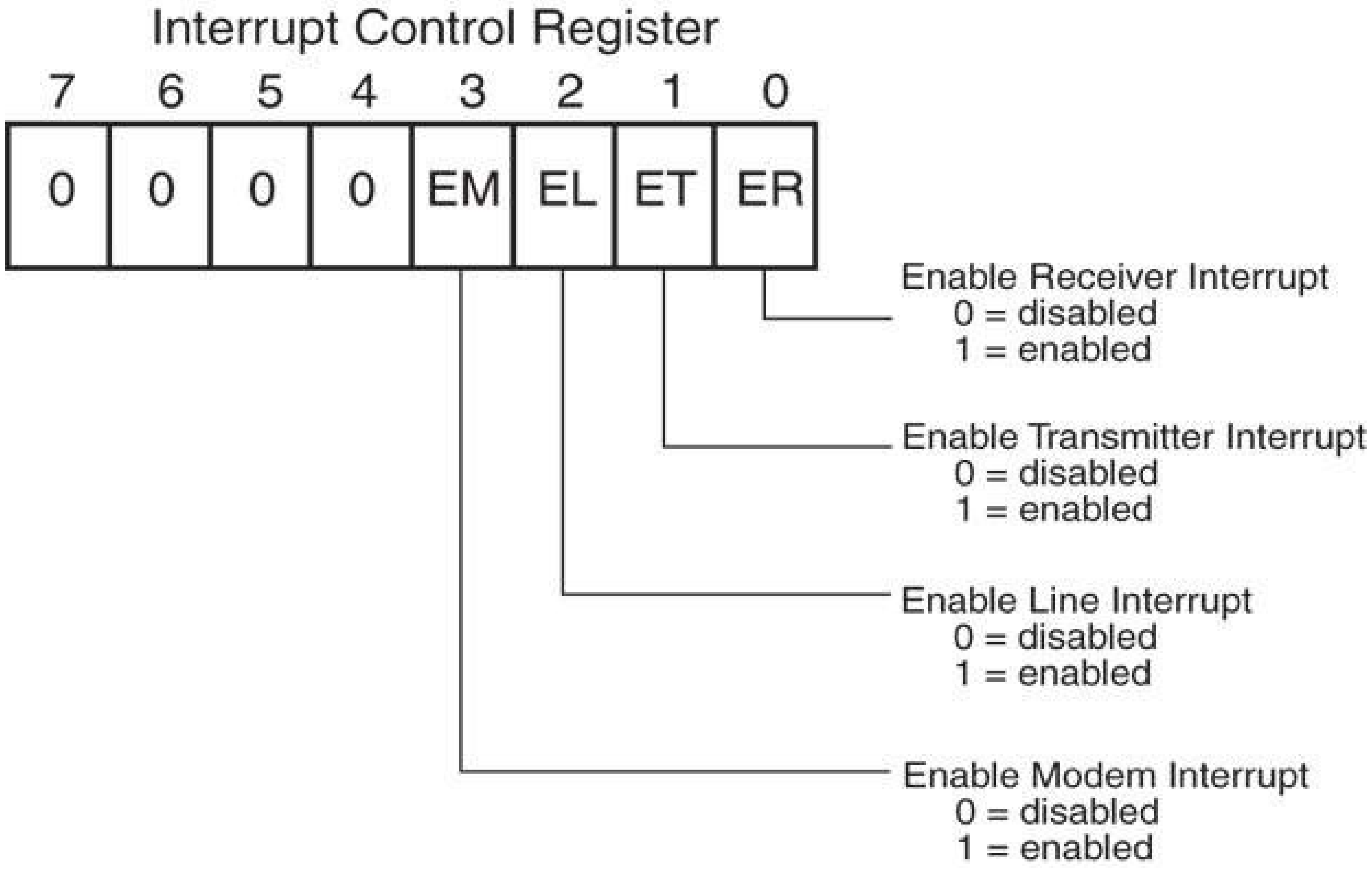


Initialization Software

- The first portion of the software for this system must program both the 16550 and the 8259A.
  - then enable the 8088 INTR pin so interrupts can take effect
- Example 12–8 lists the software required to program both devices and enable INTR.
- The software uses two memory FIFOs to hold data for the transmitter and receiver.
  - each FIFO is 16K bytes long and addressed by a pair of pointers (input and output)

- INIT programs the 16550 UART for operation with seven data bits, odd parity, one stop bit, and a baud rate clock of 9600.
- The second part of the procedure programs the 8259A, with its three ICWs and one OCW.
- The final part enables the receiver and error interrupts of the 16550.
  - transmitter interrupt not enabled until data are available for transmission
- See Figure 12–22 for the contents of the interrupt control register of the 16550 UART.

Figure 12–22 The 16550 interrupt control register.



SUMMARY

- An interrupt is a hardware- or software-initiated call that interrupts the currently executing program at any point and calls a procedure.
- The procedure is called by the interrupt handler or an interrupt service procedure.
- Interrupts are useful when an I/O device needs to be serviced only occasionally at low data transfer rates.

SUMMARY *(cont.)*

- The microprocessor has five instructions that apply to interrupts: BOUND, INT, INT 3, INTO, and IRET.
- The INT and INT 3 instructions call procedures with addresses stored in the interrupt vector whose type is indicated by the instruction.
- The BOUND instruction is a conditional interrupt that uses interrupt vector type number 5.

SUMMARY *(cont.)*

- The INTO instruction is a conditional interrupt that interrupts a program only if the overflow flag is set.
- Finally, the IRET, IRETD, or IRETQ instruction is used to return from interrupt service procedures.
- The microprocessor has three pins that apply to its hardware interrupt structure.

SUMMARY (*cont.*)

- Real mode interrupts are referenced through a vector table that occupies memory locations 0000H-03FFH.
- Each interrupt vector is four bytes long and contains the offset and segment addresses of the interrupt service procedure.
- In protected mode, the interrupts reference the interrupt descriptor table (IDT) that contains 256 interrupt descriptors

SUMMARY (*cont.*)

- Two flag bits are used with the interrupt structure of the microprocessor: trap (TF) and interrupt enable (IF).
- The IF flag bit enables the INTR interrupt input.
- TF flag bit causes interrupts to occur after the execution of each instruction, as long as TF is active.

SUMMARY (*cont.*)

- The first 32 interrupt vector locations are reserved for Intel use, with many predefined in the microprocessor.
- The last 224 interrupt vectors are for the user's use and can perform any function desired.

SUMMARY (*cont.*)

- Whenever an interrupt is detected, the following events occur:
  (1) the flags are pushed onto the stack
  (2) the IF and TF flag bits are both cleared
  (3) the IP and CS registers are both pushed onto the stack
  (4) the interrupt vector is fetched from the interrupt vector table and the interrupt service subroutine is accessed through the vector address

SUMMARY (*cont.*)

- Tracing or single-stepping is accomplished by setting the TF flag bit.
- This causes an interrupt to occur after the execution of each instruction for debugging.
- The non-maskable interrupt input (NMI) calls the procedure whose address is stored at interrupt vector type number 2.
- This input is positive edge-triggered.

SUMMARY (*cont.*)

- The INTR pin is not internally decoded, as is the NMI pin.
- Methods of applying the interrupt vector type number to the data bus vary widely.
- One method uses resisters to apply interrupt type number FFH to the data bus, while another uses a three-state buffer to apply any vector type number.

SUMMARY (*cont.*)

- The 8259A programmable interrupt controller (PIC) adds at least eight interrupt in-puts to the microprocessor.
- If more interrupts are needed, this device can be cascaded to provide up to 64 interrupt inputs.

SUMMARY (*cont.*)

- Programming the 8259A is a two-step process.
- First, a series of initialization command words (ICWs) are sent to the 8259A,
- Second, a series of operation command words (OCWs) are sent.

SUMMARY

- The 8259A contains three status registers:
  IMR (interrupt mask register)
  ISR (in-service register)
  IRR (interrupt request register).
- A real-time clock is used to keep time in real time. In most cases, time is stored in either binary or BCD form in several memory locations.