INTERRUPTS AND THE 8259 CHIP

1

Objectives

- Explain how the IBM PC executes interrupts by using the interrupt vector table and interrupt service routines
- List the differences between interrupts and CALL instructions
- Describe the difference between hardware and software interrupts
- Examine the ISR for any interrupt, given its interrup number

2

Objectives (Con't)

- Describe the function of each pin of the 8259 PIC (Programmable Interrupt Controller) chip
- Explain the purpose of each of the 4 control words of the 8259 and demonstrate how they are programmed
- Diagram how the 8259 is interfaced in IBM PC AT machines

3

- What is the difference between Interrupts and Exceptions?

4

INTERRUPTS AND EXCEPTIONS

- Interrupts and exceptions both alter the program flow. The difference between the two is that interrupts are used to handle external events (serial ports, keyboard) and exceptions are used to handle instruction faults, (division by zero, undefined opcode).

5

INTERRUPTS

- Interrupts are handled by the processor after finishing the current instruction. If it finds a signal on its interrupt pin, it will look up the address of the interrupt handler in the interrupt table and pass that routine control.
- After returning from the interrupt handler routine, it will resume program execution at the instruction after the interrupted instruction.

6

EXCEPTIONS

- Exceptions on the other hand are divided into three kinds: Faults, Traps and Aborts.
- Faults are detected and serviced by the processor before the faulting instructions.
- Traps are serviced after the instruction causing the trap. User defined interrupts go into this category and can be said to be traps; this includes the MS-DOS INT 21h software interrupt, for example.
- Aborts are used only to signal severe system problems, when operation is no longer possible.

7

- How does the CPU respond to an interrupt request?

8

CPU Interrupt Response

- When an interrupt is executed:
  a. Complete execution of current instruction
  b. Push Flag
  c. Clear IF & TF (Interrupt Flag and Trap Flag)
  d. Push CS & IP (instruction pointer) on the top of the stack
  e. Fetch ISP address or ISR (Interrupt Service Procedure/Routine)

9

- See next table for information on interrupt assignments in the Intel 386, 486 SX/DX processors, and the Pentium processor.

10

INTERRUPT VECTORS

Number	Address	Microprocessor	Function
0	00-03	All	Divide error
1	04-07	All	Single step
2	08-0B	All	NMI pin
3	0C-0F	All	Breakpoint
4	10-13	All	Interrupt on overflow
5	14-17	80286-Pentium	Bound instruction (print screen)

11

INTERRUPT VECTORS

Number	Address	Micro-processor	Function
6	18-1B	80286-Pentium	Invalid opcode
7	1C-1F	80286-Pentium	Coprocessor emulation
8	20-23	80386-Pentium	Double fault (clock tick)
9	24-27	80386	Coprocessor segment overrun (keyboard)

12

INTERRUPT VECTORS

Number	Address	Micro-processor	Function
A	28-2B	80386-Pentium	Invalid task state segment (IRQ2)
B	2C-2F	80386-Pentium	Segment not present (IRQ3)
C	30-33	80386-Pentium	Stack fault (IRQ4)
D	34-37	80386-Pentium	General protection fault (IRQ5)

13

INTERRUPT VECTORS

Number	Address	Micro-processor	Function
E	38-3B	80386-Pentium	Page fault (IRQ6)
F	3C-3F	-----	Reserved (IRQ7)
10	40-43	80286-Pentium	Floating-point error (Video)
11	44-47	80486SX	Alignment check interrupt

14

INTERRUPT VECTORS

Number	Address	Micro-processor	Function
12	48-4F	Pentium	Machine check exception
13-1F	50-7F	---	Reserved

15

INT

- There are 256 software interrupt instructions.
  - The INT instruction takes a numeric operand: 0-255
- Real mode: It's multiplied by 4 to get address of vector.

16

Interrupt Service Procedure (ISP)

- Also called the Interrupt Handler
- When CPU is interrupted:
  - Go through the procedure in CPU Interrupt Response
  - Depending on the interrupt type #, from interrupt vector table, CPU fetches:
    - The first 2 bytes → IP value
    - The next 2 bytes → CS value
  - Execute the ISP

17

- What is the difference between INT and CALL instructions?

18

Comparisons between INT and CALL instructions

INT nn → hex # (00 to FF)
INT nn (a 2-byte instruction). First byte: opcode; second byte: int #

- INT nn
  - Goes to interrupt vector table for ISP address
  - Hardware interrupt can be activated any time
  - Can be masked
  - Push CS, IP & Flag on top of stack
  - Use IRET (Interrupt Return)
- CALL
  - Can jump to any location within the range (depending on CPU) below 1 MB. Pentium – 4 GB
  - Software used by programmer
  - Cannot be masked
  - Push CS and IP only
  - Use RET

- Indicate the types of Interrupts

Types of Interrupts

- Software interrupts
- Hardware interrupts
- Exception (or Conditional) interrupts

Software Interrupts

- INT nn is referred to as a software interrupt since it is invoked from software and not from external hardware

Real-Mode: Software Interrupts

- If you've ever programmed for DOS, you probably know that the INT 21h interface has a salient importance in linking applications with operating system services.
- For instance, calling an operating system function for opening a file, simply requires issuing an INT 21h interrupt with the correct parameters set in the processor registers.

Real-Mode: Software Interrupts

- The BIOS (Basic Input Output System) uses a similar software communication mechanism.
- The INT 10h, 13h and 16h are all interfaces to internal BIOS functions, which control the screen, the disk controller and the keyboard.
- Instead of accessing hardware devices directly, DOS uses the BIOS services to control the operation of the system.
- Modern operating systems such as Windows NT drop their reliance upon the BIOS in favor of faster mechanisms (device drivers) for accessing the PC's hardware.

Real-Mode: Software Interrupts

- When the processor bumps into an INT instruction, it pushes the address of the next instruction (CS and IP registers) and the contents of its flags register to the stack and then jumps to execute the interrupt handler.
- This ensures that when the handler code ends, the processor returns automatically to the original code stream (the code which was executing before the interrupt occurred).

25

Real-Mode: Software Interrupts

- Here are the precise steps taken by the processor when it encounters an INT instruction:
  - Push Flags, CS and IP to the stack (in this order).
  - Multiply interrupt number by 4 and use the resulting number as an offset into the interrupt table (located at the beginning of the physical address space).
  - Get CS and IP of the interrupt handler from the table entry.
  - Disable interrupts
  - Jump to execute the handler.
  - When the handler code ends (an IRET instruction is executed), POP CS, IP and the flags from the stack so that control returns to the currently active application.

26

Hardware Interrupts

- There are 3 pins in the 80x86 that are associated with hardware interrupts. They are:
  - INTR (Interrupt Request
  - NMI (Non Maskable Interrupt)
  - INTA (Interrupt Acknowledge)

27

Real-Mode: Hardware Interrupts

- Hardware interrupts are not very different in behavior. But still, when considering the path traveled by a hardware interrupt from the instant it leaves the hardware device until it reaches the processor, you must take into account the actions taken by the interrupt controller (or the 8259A if you prefer).

28

Real-Mode: Hardware Interrupts

- When the processor executes an INT instruction, it retrieves the interrupt code from the opcode itself. This code is later used by the processor to index the **IVT** (interrupt vector table) and find the address of the interrupt handler.
- On the other hand, when a hardware interrupt is detected, the interrupt controller sends the interrupt code to the processor via the data bus.
- When the code is finally acquired by the processor (either from the INT opcode or from the interrupt controller) the steps taken are equivalent to those discussed earlier.

29

Real-Mode: Hardware Interrupts. Table 1 presents the interrupt vectors occupied by the interrupt controller chips under different operating systems

OS	Vectors occupied by the master 8259A	Vectors occupied by the slave 8259A
DOS	8H - FH	70H – 77H
Windows 95 / 98	50H – 57H	58H - 5FH
Windows NT	30H – 37H	38H - 3FH

30

Exception (or Conditional) interrupts

- Also software interrupts
- Example:
  - Divide-by-zero
  - Single step
  - Breakpoint
  - Signed number overflow

Dedicated Interrupts

- 5 types. From type 0 to type 4.

Type 0 (INT 00H) – divide-by-zero

- Exception or conditional interrupt
- Invoked if divide by zero
  - Example

```
MOV     AL,64H
SUB     CL,CL
DIV     CL
```

Type 0 (INT 00H) – divide-by-zero

- Invoked if quotient > assigned reg. size
- 5 (dividend)
  3 (divider)
- Example:

```
MOV     AX,0FFFFH      ;AX = 65,535
MOV     CL,5           ;CL = 5
DIV     CL             ;13107 >> AL (255)
```

→ Interrupt type 0 will invoke

Type 1 (INT 01H) – Single Step

- In single step or trace-mode:
  CPU executes one instruction at a time
- If TF is set: CPU will automatically do type 1 interrupt after each instruction is executed
- To implement single step:
  - Set TP
  - Write an ISP to save all registers on the stack
  - Load the starting address of type 1 ISP into 00004H – 00005H (always 2 bytes)

Type 1 (INT 01H) – Single Step

- To set the TF

```
PUSHF                   ;save the flag on the stack
MOV  BP,SP              ;use BP as index
OR   [BP] + 0,0100H     ;set TF to 1
POPF
```

- To reset the TF

```
PUSHF                   ;save the flag on the stack
MOV  BP,SP              ;use BP as index
AND  [BP] + 0,FFFFH     ;set TF to 0
POPF
```

→ CPU is no longer in type 1

Type 2 (INT 02H) – NMI (Non Maskable Interrupt)

- Triggered on PGT at NMI pin of the CPU
- Cannot be masked by any program instruction
- One application: AC power failure

Type 3 (INT 03H) – Breakpoint

- Use INT 03H as a breakpoint in any part of a long program

37

Type 4 (INT 04H)

- It is called Signed number overflow (INTO) and is used after a signed number arithmetic or logic operation
- If OF = 1: CPU will activate (do type 4)
- If OF = 0: The INTO instruction is not executed but is bypassed and acts as a NOP operation
- Example:

```
MOV     AL,+64
MOV     BL,+64
ADD     AL,BL    →AL =128
INTO
```

38

Logical Address & Physical Address - Review

- CS:348AH, IP:6C43H
  → CS:IP 348AH:6C43H (logical address)
- Physical address:
  - Write 348A in binary: 0011 0100 1000 1010
  - Shift left 4 times → has four # 0's
    0011 0100 1000 1010 0000 = 348A0H
  - Add IP: 348A0H + 6C43H = 3B4E3H
  - Thus, physical address = 3B4E3H

39

Problem 1

- For a given ISR, the logical address is F000:FF53. Find the physical address.

40

Problem 1

- For a given ISR, the logical address is F000:FF53. Find the physical address.

Solution:

CS:F000, IP:FF53

F000H = 1111 0000 0000 0000B

Shift 4 times: 1111 0000 0000 0000 0000B
F0000H

F0000H + FF53H = FFF53H

41

8259 PIC (Programmable Interrupt Controller)

- This IC makes the job of expanding the number of hardware interrupts much easier

42

How does the PIC work?

- The PIC receives an interrupt request from an I/O device and tells the microprocessor.
- The CPU completes whatever instruction it is currently executing and then fetches a new routine that will service the requesting device.
- Once this peripheral service is completed, the CPU resumes doing exactly what it was doing when the interrupt request occurred.
- The PIC functions as an overall manager of hardware interrupt requests in an interrupt driven system environment.

43

How do the 8259A/82C59A-2 accomplish the interrupt activity?

- The device requiring service signals the PIC via one of the seven PIC interrupt request (IR) input lines. The corresponding bit in the PIC Interrupt Request register (IIR) is set.
- The PIC activates the INT line which is connected to the CPU INTR line.
- If the interrupts are not masked at the CPU, it finishes the currently executing instruction and sends one interrupt acknowledge (INTA) pulse to the PIC.

44

How do the 8259A/82C59A-2 accomplish the interrupt activity?

- In an X86 environment, the PIC responds by setting the highest priority In Service Register (ISR) bit and the corresponding IIR bit is reset. There is no PIC activity on the data bus in this cycle.
- The CPU will initiate a second INTA pulse. During this pulse, the PIC releases an 8-bit pointer on to the data bus where it is read by the CPU.
- The CPU reads the interrupt-type number, determines the associated address of the interrupt service routine (ISR), then fetches and executes the ISR.
- In the Automatic End Of Interrupt (AEOI) Mode the ISR bit is reset at the end of the second INTA pulse. Otherwise the ISR bit remains set until an appropriate EOI command is issued at the end of the ISR.

45

Is it really this simple?

- Basically, yes.
- The scenario can be complicated by the requirement to program the PIC to fit the environment.
- For example, there are various Modes to which the PIC can be programmed.
- Another complication is interconnecting or cascading one master and up to eight slave PIC's in an application.

46

How is the PIC programmed?

- Programming the PIC can be broken into two parts.
- These are initialization and operation.
- The PIC therefore accepts two types of command words generated by the CPU:
  - Initialization Command Words (ICWs)
  - Operation Command Words (OCWs)

47

Initialization Command Words (ICWs)

- Determine the basic operating mode of the PIC.
- Two to four ICW bytes must be used.
- The ICW inputs are timed by WR# pulses.
- ICW1 and ICW2 must always be used.
- ICW3 and ICW4 are used only if they are required by specifying so in ICW1.
- The initialization sequence must start with ICW1, then ICW2, ICW3 and ICW4.
- ICWs must be completed before continuing on to the second type of command word.

48

Operation Command Words (OCWs)

- Can be written into the PIC anytime after the ICWs are written.
- There are three OCWs.
- The OCWs customize the priority features and give the user the ability to write/read various registers.
- The OCWs also tell the PIC to operate in various interrupt modes. There are 4 modes.

49

Fully nested mode

- Default mode after initialization.
- The IRs are ordered in priority 0 - 7, with 0 as the highest.
- When an interrupt is acknowledged the highest priority request is determined and its vector is placed on the bus.
- Additionally, bit IS0 - IS7 is set and remains set until the microprocessor issues an EOI or until the trailing edge of the last INTA if AEOI is set.
- While the ISx bit is set, all further interrupts of the same or lower priority are inhibited, while higher level priority interrupts will be acknowledged only if the microprocessor internal interrupt flip-flop has been re-enabled through software.

  - Rotating priority mode.
  - Special mask mode.
  - Polled mode.

50

What is the clock speed and what does the 82C59"-2" mean?

- The device is not clocked, but it usually connects to an 8MHz bus.
- Marketing shows it as an 8MHz device.
- The "-2" means an improved timing.
- In this case it is improved AC characteristics when compared to the older (no longer available) 82C59A

51

Method for handling interrupts with a standalone 8259A/82C59A-2

The best method for handling interrupts with a stand-alone 8259A/82C59A-2 interrupt service routine is as follows:

1. Save all registers.
2. Execute uninterruptable code.
3. Enable interrupts.
4. Execute interruptible code.
5. Disable interrupts.
6. Issue nonspecific EOI to master IRQ.
7. Restore registers.
8. Interrupt return.

52

Method for handling interrupts with a cascaded 8259A/82C59A-2

B=Slave and A=Master:

1. Save all registers.
2. Execute uninterruptable code.
3. Enable interrupts.
4. Execute interruptible code.
5. Disable interrupts.
6. Issue nonspecific EOI to Bank B PIC.
7. Read Bank B ISR.
8. Is there a lower priority interrupt in service on Bank B PIC?
   - NO: Issue nonspecific EOI to Bank A PIC.
     - Restore registers.
     - Interrupt return.
   - YES: Restore registers.
     - Interrupt return.

53

Does INTA acknowledge all of the interrupt requests?

- An INTA is generated by the CPU as a result of the INT from the 8259A.
- As long as any interrupt request line to the 8259A is active, the 8259A determines the priority of the interrupt request and places the appropriate 8-bit pointer to the correct interrupt service routine onto the data bus at the second INTA pulse from the CPU.
- All interrupt requests are acknowledged.

54

8259 PIC (Internal block diagram)

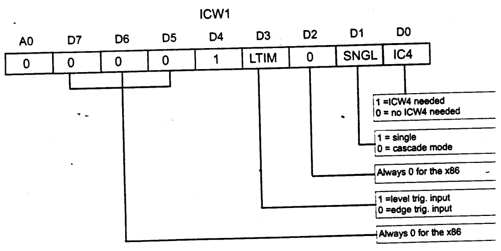
- Interrupt Request Register (IRR)
  - Keeps track of which interrupt inputs are requesting the service
- Interrupt Mask Register (IMR)
  - Masks/unmasks individual interrupt inputs
- Interrupt Service Register (ISR)
  - Keeps track of which interrupt inputs are currently being serviced by the CPU
- Priority Resolver (PR)
  - Determines whether to activate a new interrupt while the CPU is currently serving one

55

8259 PIC Control Words and Ports

- The four control words associated with the 8259 are ICW1 (initialization command word), ICW2, ICW3 and ICW4
- ICW1- To initialize
  - A single or cascaded 8259s
  - Level trigger or edge triggered input
  - ICW4 is needed or not

56

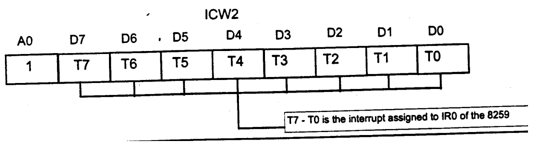


57

8259 PIC Control Words and Ports

- ICW2
  - Assigns interrupt numbers to IR0 – IR7
  - D0, D1, D2 vary from 000 to 111 along with D3-D7.
  - D3-D7 can only be programmed according to the assignment of the INT type, with the lower bits being provided by the 8259 –depending on which of IR0 to IR7 is activated.

58



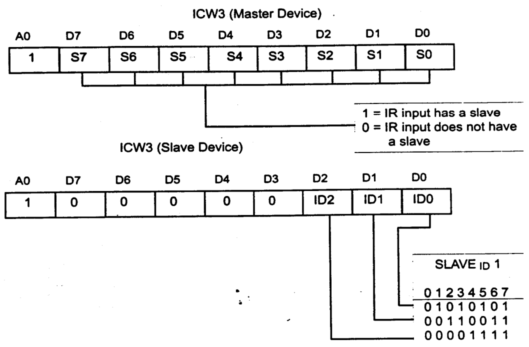
D2=D1=D0 = 000 on 80x86 systems

59

8259 PIC Control Words and Ports

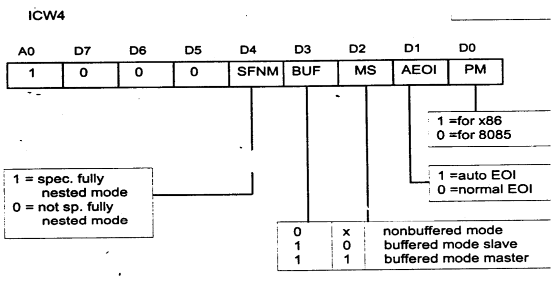
- ICW3
  - Initialized only when cascaded 8259s are used
  - A single 8259 can connect to 8 slaves 8259s providing up to 64 hardware interrupts
  - There are separate ICW3 words for the master and the slave

60



8259 PIC Control Words and Ports

- ICW4
  - Initialized for
    - 80x86 or 8085
    - Auto or Normal EOI (End of Interrupt)
    - Buffered or non buffered
    - Special or not special fully nested mode
  - If
    - Auto EOI → no need for an EOI instruction used before IRET in ISP
    - Normal EOI → ICW2 must be initialized
    - 8259 in masked mode (cascaded) D4 must be 1



Note: Buffered mode is used in systems which require buffering on the data bus. The 8259's $\overline{SP}/\overline{EN}$ pin goes low when its data bus output is enabled; therefore, $\overline{SP}/\overline{EN}$ can be used to enable a transceiver/buffer for the data bus output.

8259 Initialization

Analyze Table 14-3

CS	A0	Initialization
0	0	ICW1
0	1	ICW2, ICW3, ICW4
1	X	8259 is not addressed

How can the 8259 make a distinction between ICW2, ICW3, and ICW4 when they are sent to the same address?

- This is one of the functions of ICW1
- D0, the LSB of ICW1, will tell the 8259 if it should look for ICW4 or not.
- Likewise, if D1 is high (1) it knows that the system is configured in slave mode and it should not expect any ICW3 in the initialization sequence.

Problem 2

- Find the address for ICW1 – ICW4 if CS is activated by A7 – A1 = 0101010

Problem 2

- Find the addresses for ICW1 – ICW4 if CS is activated by A7 – A1 = 0101010

Solution – From Table 14-3

A7	A6	A5	A4	A3	A2	A1	A0	
0	1	0	1	0	1	0	0	54H
0	1	0	1	0	1	0	1	55H

Thus, 54H is the port for ICW1 and 55H is the port for ICW2, ICW3 and ICW4

Review Example 14-8

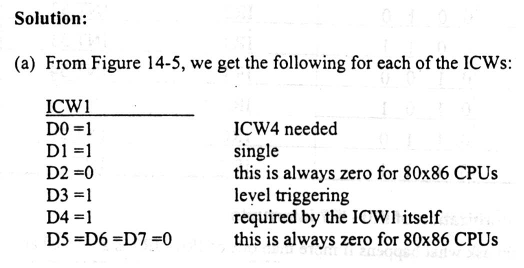
(a) Find the ICWs of the 8259 if it is used with an 8088/86 CPU, single, level triggering IRs, and IR0 is assigned “INT 50H.” The 8259 is in slave buffered mode with normal EOI.

(b) Show the program to initialize the 8259 using the port addresses in Example 14-7

(c) Find the addresses associated with IR0, IR1, and IR2 in the interrupt vector table.

Note: This example is not PC-compatible and is given only for an exercise

Review Example 14-8

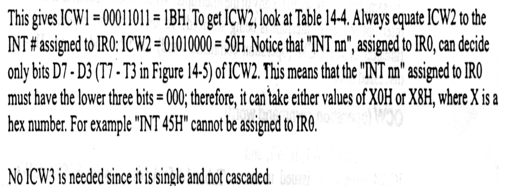


Solution:

(a) From Figure 14-5, we get the following for each of the ICWs:

ICW1	
D0 =1	ICW4 needed
D1 =1	single
D2 =0	this is always zero for 80x86 CPUs
D3 =1	level triggering
D4 =1	required by the ICW1 itself
D5 =D6 =D7 =0	this is always zero for 80x86 CPUs

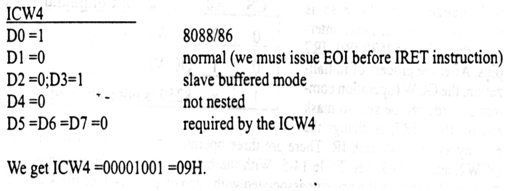
Review Example 14-8



This gives ICW1 = 00011011 = 1BH. To get ICW2, look at Table 14-4. Always equate ICW2 to the INT # assigned to IR0: ICW2 = 01010000 = 50H. Notice that "INT nn", assigned to IR0, can decide only bits D7 - D3 (T7 - T3 in Figure 14-5) of ICW2. This means that the "INT nn" assigned to IR0 must have the lower three bits = 000; therefore, it can take either values of X0H or X8H, where X is a hex number. For example "INT 45H" cannot be assigned to IR0.

No ICW3 is needed since it is single and not cascaded.

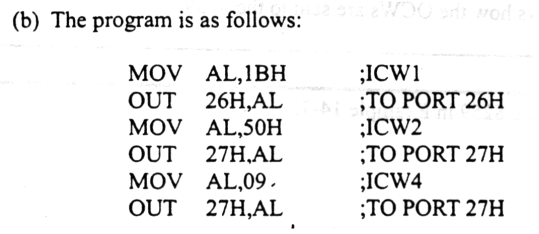
Review Example 14-8



ICW4	
D0 =1	8088/86
D1 =0	normal (we must issue EOI before IRET instruction)
D2 =0;D3=1	slave buffered mode
D4 =0	not nested
D5 =D6 =D7 =0	required by the ICW4

We get ICW4 =00001001 =09H.

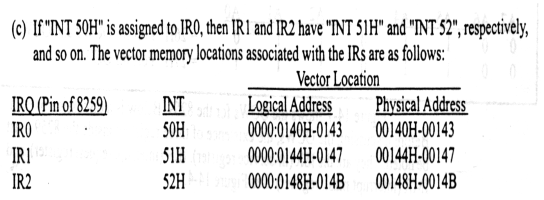
Review Example 14-8



(b) The program is as follows:

```
MOV   AL,1BH      ;ICW1
OUT   26H,AL      ;TO PORT 26H
MOV   AL,50H      ;ICW2
OUT   27H,AL      ;TO PORT 27H
MOV   AL,09       ;ICW4
OUT   27H,AL      ;TO PORT 27H
```

Review Example 14-8



(c) If "INT 50H" is assigned to IR0, then IR1 and IR2 have "INT 51H" and "INT 52", respectively, and so on. The vector memory locations associated with the IRs are as follows:

		Vector Location	
IRQ (Pin of 8259)	INT	Logical Address	Physical Address
IR0	50H	0000:0140H-0143	00140H-00143
IR1	51H	0000:0144H-0147	00144H-00147
IR2	52H	0000:0148H-014B	00148H-0014B

Masking and Prioritization of IR0 – IR7 Interrupts

- OCW (operation command word)
- OCW can be sent to mask any of IR0-IR7 or change the priority assigned to each IR
- Analyze table 14-5

Table 14-5: Addresses for 8259 OCWs

CS	A0	Operation Command Word
0	0	OCW2, OCW3
0	1	OCW1
1	x	8259 is not addressed

Problem 3

- Find the port addresses for the OCWs of the 8259. A7 – A1: 0011101

Problem 3

- Find the port addresses for the OCWs of the 8259. A7 – A1: 0011101

Solution

- From Table 14-5

A7	A6	A5	A4	A3	A2	A1	A0	
0	0	1	1	1	0	1	0	3AH
0	0	1	1	1	0	1	1	3BH

Thus, 3AH is the port for OCW2, OCW3 and 3BH is the port for OCW1.

OCW1

- It is used to mask any of IR0 – IR7.
  - Logic 1 is for masking (disabling – reset)
  - Logic 0 is for unmasking (enabling – set)
- When OCW1 is written to 8259 (by making A0 = 1 & CS = 0) it goes to IMR (Interrupt Mask Register)
- OCW1 can be read to find out which IR lines are masked/unmasked

Operation Command Word 1

D7	D6	D5	D4	D3	D2	D1	D0

Interrupt Mask

1 = Mask Set

0 = Mask Reset

Ex. If D1 = 0 → Service IRQ1

If D1 = 1 → Do not service (mask) IRQ1

Review Example 14-10

- Write the code to unmask (enable) IR0 –IR7. Port for OCW1 is 27H.

Solution –

- To enable IR0 – IR7, use 0 for M0-M7 in OCW1
- OCW1 = 0000 0000

```
– MOV    AL,00    ;OCW1 to enable IR0 – IR7
– OUT    27,AL    ;issue OCW1 to IMR
```

79

Problem 4

- Write the code to unmask (enable) IR0 – IR3 and mask (disable) IR4 – IR7.
  A7 – A1 = 0101110

80

Problem 4

- Write the code to unmask (enable) IR0 – IR3 and mask (disable) IR4 – IR7.
  A7 – A1 = 0101110

Solution – From Table 14-5

A7	A6	A5	A4	A3	A2	A1	A0		
0	1	0	1	1	1	0	0	5CH	OCW2, OCW3
0	1	0	1	1	1	0	1	5DH	OCW1

To enable IR0 – IR3 and disable IR4 – IR7 → OCW1 = 11110000B = F0H
IR7 IR0

```
MOV   AL,11110000B     ;OCW1 to enable IR0 – IR3 and disable IR4 – IR7
OUT   5DH,AL           ;issue OCW1 to IMR
```

81

OCW2

- Used to assign a specific priority to IRs
- 3 different methods for assigning priority to IR0-IR7
  - Fully nested mode
  - Automatic rotation mode
  - Specific rotation mode

82

Operation Command Word 2

D7	D6	D5	D4	D3	D2	D1	D0
R	SL	EOI	0	0			

D2, D1, D0: Interrupt Request Level to act upon

D7-D6-D5

001	Non-specific EOI command
010	NOP
011	Specific EOI command
100	Rotate in automatic EOI mode
101	Rotate on non-specific EOI command
110	Set priority command
111	Rotate on specific EOI command

83

OCW2

- Fully nested mode
  - This is the default mode when 8259 is initialized: assigns the highest priority to IR0 and the lowest to IR7
  - OCW2 can be programmed to assign the highest priority to any IR
  - Ex.: the following shows OCW2 if IR6 has been assigned the highest priority, then IR7 and so on

IR0	IR1	IR2	IR3	IR4	IR5	IR6	IR7
2	3	4	5	6	7	0	1

84

OCW2

- **Automatic rotation mode**
  - Automatically rotates from IR0 to IR7, then IR0 again
  - When an IR has been served it will take the lowest priority and will not be served until every other request has had a chance.
  - This prevents interrupt starvation, where one device monopolizes the interrupt service.

85

OCW2

- **Specific rotation mode**
  - The 8259 can be programmed to make the rotation follow a specific sequence.
  - The IR being served will become the lowest priority until all the IRs are served
  - The only difference between this mode and automatic rotation is the sequence of rotation

86

Assign the highest priority to IR6

- Use D2 – D0 = 110
- D4 - D3 = 0 (always for OCW2)
- D5 (EOI – End of Interrupt), D6 (SL – Select) and D7 (R – Rotation) are used to program the 8259 for the various priority schemes discussed earlier

87

Importance of the EOI Command (End of Interrupt)

- EOI & IRET should be the last instruction of ISR (any interrupt service routine for IR0-IR7)
- If programmer fails to issue the EOI for any IRs, this particular IR will be out of circulation

88

Assume that IR3 is being serviced

- IR2, IR1 and IR0 are allowed to come in and interrupt it since they have higher priority.
- However, no lower-priority interrupts of IR4 – IR7 are responded to.

89

Example 14-11

Show the last 3 instructions of the interrupt service routine of the 8259

Solution

Before the interrupt service routine returns control to the main program, it issues OCW2 to the 8259

```
INT_SERV   PROC   FAR       ;routine for IR1
           …..    …..
           MOV    AL,20H    ;the EOI byte for OCW2
           OUT    26H,AL    ;to port designated for OCW2
           IRET             ;return from ISR to the main
                            ;program
```

90

OCW3

- Reads the 8259's internal registers IRR (inter request register) and ISR (in-service register)
- D0 & D1 are used to read IRR and ISR to find out which of IR0-IR7 is pending for service and which one is being served
- D2, D5 & D6 are for changing the masking mode and other advanced functions of 8259

91

Operation Command Word 3

D7	D6	D5	D4	D3	D2	D1	D0
0	ESMM	SMM	0	1	P	RR	RIS

RR = Read Register

SMM = Special Mask Mode

D0: 1 = read IRR on next read, 0 = read ISR on next read

D1: 1 = act on value of bit 0, 0 = no action if bit 0 set

D2: 1 = poll command issued, 0 = no poll command issued

D5: 1 = set special mask, 0 = reset special mask

D6: 1 = act on value of bit 5, 0 = no action if bit 5 set

92

Interrupts in 80286 and higher 80x86 PCs

- In PC/XT
  - IRQ0 & IRQ1 – used by system board
  - IRQ2 – IRQ7 –on expansion slot
- 80286 and all newer microprocessors
  - IBM used second 8259
  - Therefore, there is a total of 16 interrupts

93

IBM PC hardware interrupts

- Cascaded 8259s connections
  - To be downward compatible with PC/XT
    - IRQ0 → system timer
    - IRQ1 → keyboard
  - IRQ2 (master) → INTR (slave)
  - SP'/EN (master) → V_{CC}
    SP'/EN (slave) → Ground
  - INTR (master) → INTR (slave)
  - INTA (CPU) to both INTA of master and slave
- See Figure 14-11

94

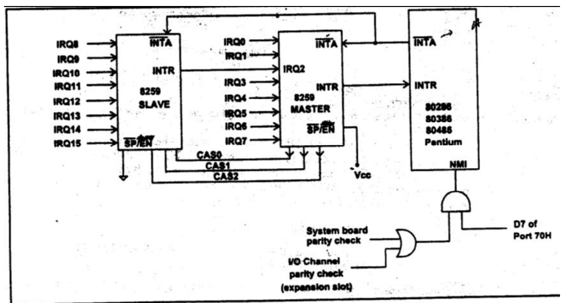


Figure 14-11. 8259 Chips in Master/Slave Relation for 286 and Higher PCs

95

IBM PC hardware interrupts

- Hardware interrupt assignment
  - IR0 → INT 08H
  - IR1 → INT 09H
  - IR2 → INT 0AH
  - IR3 → INT 0BH
  - IR4 → INT 0CH
  - IR5 → INT 0DH
  - IR6 → INT 0EH
  - IR7 → INT 0FH

96

IBM PC hardware interrupts

- Missing IRQs on AT expansion slot
  - Two ways to activate the interrupt input IRQ.
  - Depending on how it is programmed
    - Edge and level triggered interrupts
      - Level triggered mode
        - » The 8259 will recognize a high-on the IRQ input as an interrupt request
      - Edge-triggered mode
        - » The 8259 will recognize an interrupt request only when a low-to-high pulse is applied to an IRQ input

97

Missing IRQs on AT expansion slot

- IRQ8 is missing on the expansion slot. Therefore, it is used on the system board by Motorola MC146818, real-time clock. CMOS RAM chip
- IRQ13 is used to capture match coprocessor interrupts; however, this is handled by NMI (Non Maskable Interrupt) for upward compatibility

98

Missing IRQs on AT expansion slot

This code from BIOS of the IBM PC AT documentation shows this process

```
INT_MATH    PROC    NEAR
            PUSH    AX          ;save (ax)
            SUB     AL,AL
            OUT     0F0H,AL     ;remove the interrupt request
            MOV     AL,20H      ;issue EOI
            OUT     0A0H,AL     ;to slave
            OUT     20H,AL      ;to master
            POP     AX          ;restore (AX)
            INT     02          ;give control to NMI (Int type 2)
            IRET                ;return
INT_MATH    ENDP
```

99

80286 Interrupt Priority

Order	Interrupt
1	INT instruction or exception
2	Single step
3	NMI
4	Processor extension segment overrun
5	INTR

100

80x86 IRQ Priority

IRQ0	HIGHEST PRIORITY
IRQ1	
IRQ8	IRQ15
IRQ9	IRQ3
IRQ10	IRQ4
IRQ11	IRQ5
IRQ12	IRQ6
IRQ13	IRQ7 LOWEST PRIORITY
IRQ14	

101