OBJECTIVES: AT THE END OF THIS SECTION, THE STUDENT WILL BE ABLE TO DISCUSS THE DETAILS OF THE INTERRUPT STRUCTURE OF THE PC/AT

- Name the various sources of interrupts.
- Describe basic interrupt operations.
- Discuss the Interrupt Vector Table.
- Explain the CPU & PIC operation during an interrupt in detail.
- Look at an example Interrupt Service Routine
- Discuss the Hardware Interrupts associated with the PICs.
- Discuss the NMI Hardware Interrupt Logic
- Describe Exceptions & Software Interrupts
VOCABULARY

- IRET - Interrupt Return
- ISR - Interrupt Service Routine
- IVT - Interrupt Vector Table
- NMI - Non Maskable Interrupt
- PIC - 8259A Programmable Interrupt Controller
3 TYPES OF INTERRUPTS
3 TYPES OF INTERRUPTS

- HARDWARE INTERRUPTS
  - INTR (coming from the PICs)--generated by external events like keyboard interrupts, disk interrupts, com port interrupts
  - NMI--taking care of critical problems (parity error)

- INTERNALLY GENERATED EXCEPTIONS
  - DIVIDE BY ZERO
  - DEBUG INTERRUPT
  - INTERRUPT ON OVERFLOW
  - INVALID OPCODE

- SOFTWARE INTERRUPTS
  - The INT N instruction, which allows programs to transfer to BIOS and DOS (e.g. INT 10, INT 21)
3 TYPES OF INTERRUPTS

3 TYPES OF INTERRUPTS

Parity Error
Edge Triggered
COMM1, Keyboard, etc
Level Sensitive
NMI
INTR

HARDWARE

MOV BL, 0
DIV BL

CPU

EXCEPTION

INT 10H

SOFTWARE

TYPE 0

INT 10H

MOV BL, 0
DIV BL

TYPE 0

INT 10H
Basic Interrupt Operations
Basic Interrupt Operations

Mainline Program

Push Flags
Clear IF (Disable Interrupts)
Clear TF (NOT Single Step)
Push CS
Push IP
Fetch ISR Address

Interrupt Service Routine

Push Registers

POP REGISTERS
IRET

Pop IP
Pop CS
Pop Flags
Basic Interrupt Operations

Interrupt detected by CPU.

1. Push Flag, CS & IP registers onto stack and disable interrupt recognition
   - The flag word is saved on the stack.
   - The IF and TF flags are cleared.
   - The return address of the interrupted program is saved (pushed) on the stack.

2. For H/W interrupts, the CPU generates two Interrupt Acknowledge Bus cycles to request the Interrupt Table entry number from the PIC (8259A Programmable Interrupt Controller)
   - The CPU reads the IVT entry number from the lower data path (D7:0) on 2nd INTA# Bus cycle.
Basic Interrupt Operations

3. The new CS:IP is loaded from the IVT.
   - Index into the IVT (Interrupt Vector Table) in memory;
   - Read table entry into the CPU (used as the new CS:IP value).

4. The ISR (Interrupt Service Routine) is executed using the CS:IP register pair loaded from the IVT.
   - Service the Interrupt
     - (Read keyboard, change video mode, read a disk sector, etc)
   - Execute IRET (Interrupt Return) instruction (at end of ISR)
     - Pop original CS:IP & Flag contents off Stack to return control to the original program.
     - Restore the flag word
   - Continue interrupted program
THE INTERRUPT VECTOR TABLE
PARTIAL INTERRUPT VECTOR TABLE
Expanded on following pages

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Vector</th>
<th>CS</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRQ7 - LPT1</td>
<td>000C</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ6 - FLOPPY</td>
<td>0038</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ5 - LPT2 / HD</td>
<td>0034</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ4 - COM1</td>
<td>0030</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ3 - COM2</td>
<td>002C</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ2 - CASCADE</td>
<td>0028</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ1 - KBD</td>
<td>0024</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ0 - TOD</td>
<td>0020</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>NPX UNAVAILABLE</td>
<td>001C</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>INVALID OPCODE</td>
<td>0018</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>PRT SCRN</td>
<td>0014</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>OVERFLOW</td>
<td>0010</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>BREAKPOINT</td>
<td>000C</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>NMI</td>
<td>0008</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>DEBUG</td>
<td>0004</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>DIV / 0</td>
<td>0000</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ15 - ISA BUS</td>
<td>01DC</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ14 - AT HD</td>
<td>01D8</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ13 - FERR</td>
<td>01D4</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ12 - ISA BUS</td>
<td>01D0</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ11 - ISA BUS</td>
<td>01CC</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ10 - ISA BUS</td>
<td>01C8</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ9 - REDIR IRQ2</td>
<td>01C4</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>IRQ8 - RTC</td>
<td>01C0</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>BIOS - PRINTER</td>
<td>000C</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>BIOS - KEYBOARD</td>
<td>0058</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>BIOS - FCN</td>
<td>0054</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>BIOS - SERIAL</td>
<td>0050</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>BIOS - DISK</td>
<td>004C</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>BIOS - MEM SIZE</td>
<td>0048</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>BIOS - EQPT CHK</td>
<td>0044</td>
<td>CS</td>
<td>IP</td>
</tr>
<tr>
<td>BIOS - VIDEO</td>
<td>0040</td>
<td>CS</td>
<td>IP</td>
</tr>
</tbody>
</table>

* Interrupts marked with an asterisk are Intel reserved interrupts which share PC assigned interrupt functions.
THE INTERRUPT VECTOR TABLE

All interrupts and exceptions use the IVT.

- The interrupt vector table is simply a table containing the "SEGMENT:OFFSET" addresses of the interrupt routines.
- Each entry in the table occupies four bytes (DWord):
  - OFFSET (IP value)
    - 1 word or 2 bytes
    - Low Word of DWord
  - SEGMENT (CS value)
    - 1 word or 2 bytes
    - High Word of DWord
THE INTERRUPT VECTOR TABLE

- The table (and the CPU) supports 256 table entries, referred to as interrupt TYPES.
  - This number is commonly referred to by the following names:
    - Interrupt Type Code
    - Interrupt ID
    - Interrupt Vector
    - Interrupt Table Entry Number

- Each entry is referred to by the interrupt TYPE NUMBER.
  - The valid type numbers run from 0 to 255
  - The table is 1024 bytes long (1024/4 = 256)
# PARTIAL INTERRUPT VECTOR TABLE

<table>
<thead>
<tr>
<th>INT TYPE</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INT TYPE</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td></td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td></td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td></td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0014</td>
<td></td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0018</td>
<td></td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001C</td>
<td></td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INT TYPE</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
<th>CS</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0020</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0030</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0034</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0038</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Intermits marked with an asterisk are Intel reserved interrupts which share PC assigned interrupt functions.
### PARTIAL INTERRUPT VECTOR TABLE

<table>
<thead>
<tr>
<th>INT TYPE</th>
<th>CS</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>BIOS - PRINTER</td>
<td></td>
</tr>
<tr>
<td>005C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>BIOS - KEYBOARD</td>
<td></td>
</tr>
<tr>
<td>0058</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>BIOS - FCN</td>
<td></td>
</tr>
<tr>
<td>0054</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>BIOS - SERIAL</td>
<td></td>
</tr>
<tr>
<td>0050</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>BIOS - DISK</td>
<td></td>
</tr>
<tr>
<td>004C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>BIOS - MEM SIZE</td>
<td></td>
</tr>
<tr>
<td>0048</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>BIOS - EQPT CHK</td>
<td></td>
</tr>
<tr>
<td>0044</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>BIOS - VIDEO</td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INT TYPE</th>
<th>CS</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>77</td>
<td>IRQ15 - ISA BUS</td>
<td></td>
</tr>
<tr>
<td>01DC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>IRQ14 - AT HD</td>
<td></td>
</tr>
<tr>
<td>01D8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>IRQ13 - FERR</td>
<td></td>
</tr>
<tr>
<td>01D4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>IRQ12 - ISA BUS</td>
<td></td>
</tr>
<tr>
<td>01D0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>IRQ11 - ISA BUS</td>
<td></td>
</tr>
<tr>
<td>01CC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>IRQ10 - ISA BUS</td>
<td></td>
</tr>
<tr>
<td>01C8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>IRQ9 - REDIR IRQ2</td>
<td></td>
</tr>
<tr>
<td>01C4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>IRQ8 - RTC</td>
<td></td>
</tr>
<tr>
<td>01C0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
THE INTERRUPT VECTOR TABLE

Notice that the address of a particular vector is four times the type number.

EXAMPLE:

- TYPE 09H AT ADDRESS 24H
  - \(4 \times 9t = 36t = 24H\)
  - IP @ 24H & 25H
  - CS @ 26H & 27H

- TYPE 10H AT ADDRESS 40H
  - \(4 \times 16t = 64t = 40H\)
  - IP @ 40H & 41H
  - CS @ 42H & 43H
LOOKING AT THE VECTOR TABLE WITH ITP

DISPLAY PART OF THE IVT WITH ITP

• DISPLAY 100 WORDS STARTING AT ADDRESS 0:0 (0p)
  • ORD2 0000:0000 LENGTH 100

LOOKING AT A PARTICULAR VECTOR WITH ITP

• LOOK AT THE KEYBOARD VECTOR (IRQ1 = TYPE 9)
  • TYPE 9 ==> 9x4 = 36t = 24H
• ORD2 0000:0024 LENGTH 2
• WORD 24p LENGTH 2
• WORD (4*9) p LENGTH 2
The Interrupt Subsystem
The second INTA pulse causes the 8259A to put an 8-bit pointer onto the data bus.
The Interrupt Subsystem

The 8259 PICs provide the following:

- **INDIVIDUAL MASKING**
  - Each IRQ input can be separately masked and unmasked.

- **PRIORITIZATION**—Interrupt sources are prioritized,
  - level 0 being highest and level 7 lowest.

- **INTERRUPT TYPE NUMBERS**
  - Used by the CPU to obtain the address of the interrupt service routine from the vector table.

- Both PICs are initialized by BIOS to provide the correct interrupt TYPE NUMBERS and mode of operation.
  - PIC #1 IR0 - TYPE 8
  - PIC #2 IR0 - TYPE 70
The Interrupt Subsystem

The PIC will do the following when an interrupt request line has been raised.

- Check the mask register.
- Compare priorities. The request must have the highest priority of the current requests and higher than those in service.
- If unmasked and possessing a high enough priority, the 8259 issues an interrupt to the CPU INTR input.
- During the CPU initiated interrupt acknowledge cycles, the 8259 puts the interrupt type number on the data bus for the processor to read.
  - The processor uses the type number to transfer control to the interrupt routine.
The Interrupt Subsystem

- The CPU INTR input is level sensitive (1 = asserted).
- The CPU INTR input is Maskable. The CPU interrupt enable flag (IF) controls the enabling and disabling of INTR.
  - IF = 1 (true) enables INTR interrupts; use STI instruction to enable.
  - IF = 0 (false) disables INTR interrupts; use CLI instruction to disable.
- Two Intel 8259A Programmable Interrupt Controllers (PIC) field all interrupt sources in the PC/AT (excluding NMI sources).
  - 8259 #1 is the MASTER PIC.
  - 8259 #2 is the SLAVE PIC.
- The MASTER passes interrupt requests directly to the INTR pin, including those passed to the MASTER from the SLAVE.
DETAILED INTERRUPT OPERATION
DETAILED INTERRUPT OPERATION

- The following example will trace the steps of an Interrupt Subsystem.
- The Keyboard Interface uses the IRQ1 input to the master interrupt controller to generate an interrupt request when a key is pressed.
- The interrupt table entry number assigned to this input is 09h.
DETAILED INTERRUPT OPERATION OVERVIEW

1. One of the interrupt request lines IRO-IR7 is raised.
2. The 8259A responds with an INTR signal to the CPU.
3. Interrupt detected by CPU.
4. The CPU outputs the Interrupt Acknowledge bus cycles if the IF flag is set.
5. The PIC places a Type number on D0:D7.
6. The microprocessor calculates the actual start memory address in the Interrupt Vector Table.
7. The new CS:IP is loaded from the IVT.
8. The **ISR** (Interrupt Service Routine) is executed.

Continue interrupted program
DETAILED INTERRUPT OPERATION

The following actions are carried out for an interrupt request by a peripheral:

1. One of the interrupt request lines IRO-IR7 is raised to a high level and the corresponding bit in the IRR is set.
   - e.g. - Keyboard Controller generates IRQ1 (which is wired to IR1) when a key is pressed which sets Bit 1 in the IRR.

2. The 8259A detects this signal and responds with an INTR signal to the CPU.

3. Interrupt detected by CPU.

4. The CPU outputs the Interrupt Acknowledge bus cycles if the IF flag is set.
DETAILED INTERRUPT OPERATION

5. The second interrupt acknowledge causes the PIC to place an 8-bit number on the lower data path, D0:D7.
   • The CPU reads this pointer as the number of the interrupt handler to call.
   • It is formed by adding the three-bit binary value of the interrupt request number to the base address stored in a register.
     ∙ Both PICs are initialized by BIOS to provide the correct interrupt TYPE NUMBERS and mode of operation.
       » PIC #1- IR0 = TYPE 8; PIC #2 - IR0 = TYPE 70
     ∙ e.g. - IRQ1 causes a TYPE 9
       ∙ 1t + 8t = 9t
       ∙ 001y + 1000y = 1001y
DETAILED INTERRUPT OPERATION

6. The microprocessor calculates the actual start memory address of entry 9 in the Interrupt Table by multiplying the interrupt table entry number, 09h in this case, by a factor of 4.

- **THE KEYBOARD VECTOR (IRQ1 = TYPE 9)**
  - TYPE 9 ==> 9x4 = 36t = 24h
  - 9 times 4 equals 36 in decimal, or 24h (HEX).
- The start memory address of entry 9 in the Interrupt Table is 00024h.
DETAILED INTERRUPT OPERATION

7. The new CS:IP is loaded from the IVT.

- The microprocessor automatically performs memory read cycles to read the four bytes of information from the interrupt table entry.
  - The interrupt table entry number assigned to this input is 09h. This means that entry 9 in the Interrupt Table contains the 4 byte start address of the keyboard interrupt service routine in memory.
  - The start memory address of entry 9 in the Interrupt Table is 00024h.
  - The 4 bytes of data will be at memory addresses 24h, 25h, 26h & 27h.
DETAILED INTERRUPT OPERATION

7. (Cont.)

- The microprocessor reads the contents of location **00024h and 00025h** and places them into temporary register inside the microprocessor.
  - This will be the ISR **IP Register** value.

- The microprocessor reads the contents of location **00026h and 00027h** and places them into another temporary register inside the processor.
  - This will be the ISR **CS Register** value.

- The microprocessor will use these stored values to fetch the first instruction of the **Interrupt Service Routine** for the keyboard.
DETAILED INTERRUPT OPERATION

8. The **ISR** (Interrupt **Service Routine**) is executed using the **CS:IP** register pair loaded from the IVT.

   - The ISR handles the Interrupt Request.
     - e.g. IRQ1 - Read keyboard controller data port to get key pressed.
   - Execute IRET (Interrupt Return) instruction (at end of ISR)
     - Pop original CS:IP & Flag contents off Stack to return control to the original program.
     - Restore the flag word

Continue interrupted program
DETAILED INTERRUPT OPERATION REVIEW

1. One of the interrupt request lines IRO-IR7 is raised.
2. The 8259A responds with an INTR signal to the CPU.
3. Interrupt detected by CPU.
4. The CPU outputs the Interrupt Acknowledge bus cycles if the IF flag is set.
5. The PIC places a Type number on D0:D7.
6. The microprocessor calculates the actual start memory address in the Interrupt Vector Table.
7. The new CS:IP is loaded from the IVT.
8. The **ISR** (Interrupt **Service** **Routine**) is executed.

Continue interrupted program
Example--Simple Interrupt Routine

```
READ_UART: ;ISR (INTERRUPT SERVICE ROUTINE)
STI              ;RE-ENABLE INTERRUPTS
                ;ALLOW THIS ISR TO BE INTERRUPTED
PUSH AX          ;SAVE REGISTERS USED IN THIS ISR
PUSH DX
MOV DX,3F8H     ;ADDRESS OF COM1 DATA PORT
IN AL,DX        ;READ THE BYTE. THIS CLEARS THE
                ; INTERRUPT REQUEST TO THE PIC
MOV AH,0EH      ;SET-UP FOR INT 10 FUNCTION 0EH
INT 10H         ; 0EH IS “ECHO TO SCREEN”
CLI              ;DISABLE INTERRUPTS
MOV AL,20H       ;ADDRESS OF PIC #1
OUT 20H,AL      ;EOI (END OF INTERRUPT) COMMAND
                ;  CLEARS IN SERVICE REG. BIT IN PIC
POP DX
POP AX          ;RESTORE REGISTERS
IRET            ;POPS IP, CS, & FLAGS FROM
                ; INTERRUPTED PROGRAM.
```
HARDWARE

INTERRUPT

GENERATION
## HARDWARE INTERRUPTS

### TYPE | ISA BUS | FUNCTION
--- | --- | ---
IRQ0 | 08H | TOD
IRQ1 | 09H | KEYBOARD
(IRQ2) | (0AH) | (X)
IRQ3 | 0BH | X
IRQ4 | 0CH | X
IRQ5 | 0DH | X
IRQ6 | 0EH | X
IRQ7 | 0FH | X

### PIC #1 PORTS
- PORTS: 20H, 21H
- 8259 #1 MASTERS
- CAS0-2

### PIC #2 PORTS
- PORTS: 0A0H, 0AIH
- 8259 #2 SLAVES

### IRQ2 redirect note:
- IRQ9 ISR (Type 71) invokes S/W Interrupt 0AH (old IRQ2 was Type 0AH)
# Interrupt Levels

<table>
<thead>
<tr>
<th>PRIORITY</th>
<th>PIC NO.</th>
<th>INT. NO</th>
<th>INTERRUPT SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>NMI</td>
<td>PARITY ERROR DETECTED</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>IRQ0</td>
<td>INTERVAL TIMER (PIT), COUNTER 0 OUTPUT</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>IRQ1</td>
<td>FULL KEYBOARD OUTPUT BUFFER</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>IRQ2</td>
<td>INTERRUPT FROM CONTROLLER 2 (CASCADE)</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>IRQ8</td>
<td>REAL-TIME CLOCK INT</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>IRQ9</td>
<td>SOFTWARE REDIRECTED TO INT OAH (IRQ2)</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>IRQ10</td>
<td>RESERVED</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>IRQ11</td>
<td>RESERVED</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>IRQ12</td>
<td>AUXILIARY DEVICE</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>IRQ13</td>
<td>INT FROM COPROCESSOR</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>IRQ14</td>
<td>FIXED DISK CONTROLLER</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>IRQ15</td>
<td>RESERVED</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>IRQ3</td>
<td>COM2</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>IRQ4</td>
<td>COM1 (PRIMARY)</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>IRQ5</td>
<td>LPT2</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>IRQ6</td>
<td>FLOPPY DISK CONTROLLER</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>IRQ7</td>
<td>LPT1 (PRIMARY)</td>
</tr>
</tbody>
</table>

* - NMI is not an input to either PIC. Priority set by CPU.
ISA INTERRUPTS

- In ISA environments, IRQ lines are typically not shareable because only one transition is registered if more than one card generates a transition.

  - More than one ISA device may share an IRQ line as long as it is guaranteed they never generate requests simultaneously.

  - The ISR must determine which device caused the IRQ.
    - Possibly by reading an Interrupt Pending Bit Status bit implemented in the device.

- Since only one device may use each IRQ line, a fully loaded machine may easily use up all available interrupt lines.
NON-MASKABLE-INTERRUPT (NMI)
NON-MASKABLE-INTERRUPT (NMI)

- The NMI input to the CPU is provided to handle critical situations.
- NMI is edge-triggered interrupt TYPE 2
- IN THE PC/AT
  - NMI is used to handle memory parity errors in the PC/AT.
  - The inputs to the NMI pin can be enabled and disabled by Port 70 Bit 7.
  - The parity error signals to NMI are disabled by default when the system is reset.
  - The inputs to NMI are enabled after memory (and thus the parity bits) is initialized during POST.
NMI - HARDWARE INTERRUPT LOGIC

IOCHCK# (signal from ISA bus adapter cards)
- PORT 61H BIT3 = 0 ENABLE
- PORT 61H BIT3 = 1 DISABLE

SYSTEM BOARD DRAM PARITY CHECK
- PORT 61H BIT2 = 0 ENABLE
- PORT 61H BIT2 = 1 DISABLE

PORT 70H - BIT7 = 0 ENABLE NMI
PORT 70H - BIT7 = 1 DISABLE NMI

NOTE: PORT (70) also used for RTC
ENABLING AND DISABLING PARITY ERRORS

- IOCHCK# (signal from ISA bus, adapter cards)
  - PORT 61H BIT3 = 0 ENABLE
  - PORT 61H BIT3 = 1 DISABLE

- SYSTEM BOARD DRAM PARITY CHECK
  - PORT 61H BIT2 = 0 ENABLE
  - PORT 61H BIT2 = 1 DISABLE

- NMI
  - PORT 70H BIT7 = 0 ENABLE
  - PORT 70H BIT7 = 1 DISABLE
## I/O Address 61H Bit Assignment

<table>
<thead>
<tr>
<th>BIT/VALUE</th>
<th>FUNCTION</th>
<th>ACCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT 7</td>
<td>Onboard parity error</td>
<td>Read only</td>
</tr>
<tr>
<td></td>
<td>Parity error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>No parity error</td>
<td></td>
</tr>
<tr>
<td>BIT 6</td>
<td>ISA parity error</td>
<td>Read only</td>
</tr>
<tr>
<td></td>
<td>Parity error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>No parity error</td>
<td></td>
</tr>
<tr>
<td>BIT 5</td>
<td>Speaker signal</td>
<td>Read only</td>
</tr>
<tr>
<td></td>
<td>Speaker signal on</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Speaker signal off</td>
<td></td>
</tr>
<tr>
<td>BIT 4</td>
<td>Refresh signal</td>
<td>Read only</td>
</tr>
<tr>
<td></td>
<td>The refresh signal toggles</td>
<td></td>
</tr>
<tr>
<td></td>
<td>with each refresh.</td>
<td></td>
</tr>
<tr>
<td>BIT 3</td>
<td>Enable ISA parity error</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td>Parity error disabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Parity error enabled</td>
<td></td>
</tr>
<tr>
<td>BIT 2</td>
<td>Enable onboard parity error</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td>Memory parity error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Parity error disabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Parity error enabled</td>
<td></td>
</tr>
<tr>
<td>BIT 1</td>
<td>Speaker data</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td>Speaker data on</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Speaker data off</td>
<td></td>
</tr>
<tr>
<td>BIT 0</td>
<td>Enable speaker</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td>Speaker enabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Speaker disabled</td>
<td></td>
</tr>
</tbody>
</table>
Software
Interrupts & Exceptions
Software Exceptions

There are two types of software conditions that can cause an interrupt to an Intel x86 microprocessor.

- **Exception interrupts.**
  - The Interrupt Type Number is set by the Processor itself.

- **Software interrupts**
  - Software interrupts generated by the INT (interrupt) instruction.
  - This instruction takes the following format: **INT 10**
Software Exceptions

- Exception interrupts.
- The Interrupt Type Number is set by the Processor itself.
  - Exception interrupt results when the microprocessor attempts to execute an instruction and incurs an error while doing so.
- A classic example would be an attempt to divide a number by zero.
  - Other sources of Exception interrupts are:
    - Invalid OP CODE (Type 6)
    - Segment Not Present (Type 0Bh)
    - General Protection Error (Type 0Dh)
Software Exceptions

Exception interrupts (Cont.)

- When the CPU attempts execution of an instruction that would cause a divide-by-zero condition, the CPU generates a divide-by-zero exception interrupt (Type 0).
  - Entry 0 in the interrupt table is dedicated to this condition and points to the start address of the divide-by-zero interrupt service routine supplied by the programmer.

  - NOTE: Exception interrupts also use entry numbers 8 through 10h which coincide with numbers used by hardware devices.
  - To eliminate possible conflicts, the exception handlers (ISRs) have the responsibility of determining if this was caused by an exception or hardware interrupt.
Software Interrupt Instruction

Software interrupts

- This instruction takes the following format:  \textbf{INT 10}

- The interrupt instruction allows the program to simulate a hardware interrupt request.

- When the microprocessor executes the example instruction shown above, it reacts very much as it would to a hardware interrupt request received on the maskable interrupt request line (INTR)
Software Interrupt Instruction

Software interrupts (Cont.)

• It first saves the contents of the Flag, CS and IP registers on the stack.
  • At this time, CS and IP point to the instruction immediately after the INT instruction.

• Since the interrupt was not actually caused by a hardware interrupt request on the INTR line, the microprocessor doesn’t perform two interrupt acknowledge bus cycles to request the interrupt table entry number from the 8259 interrupt controller.

• Instead interrupt table entry number is supplied by the hexadecimal number to the right of the INT instruction
  » INT 10
Review - The Interrupt Process

Hardware Event e.g. Key-Press

Interrupt Ctrl-1

IRQ-X

Ctrl-2

INT

Interrupt

Apply Running when Interrupt Occurs

Interrupt Service Routine

IRET

Return to Application after Interrupt

SS:SP

Old FLAGS

Old IP

Old CS

Old FLAGS

Application

Running

when

Interrupt

Occurs

VECTOR-X

Vect-X CS

Vect-X IP

VECTOR-Y

Vect-Y CS

Vect-Y IP

PUSH onto Stack

INT-Y

INT

INT-Y

Ctrl-2

Interrupt

SOFTWARE Interrupt Instruction

SOFTWARE Interrupt Instruction

INT

INT-Y

INT

INT-Y

Interrupt Ctrl-2

IRQ...

IRQ-X

Hardware Event e.g. Key-Press

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT

INT-Y

INT
SUMMARY

WE HAVE DISCUSSED THE FOLLOWING:

- Various Sources of Interrupts.
- Basic interrupt operations.
- The Interrupt Vector Table.
- PIC & CPU operation during an interrupt.
- Hardware interrupts associated with the PICs.
- An example Interrupt Subroutine.
- How NMI handles the critical error of memory failure.
- Software Interrupts & Exceptions.