

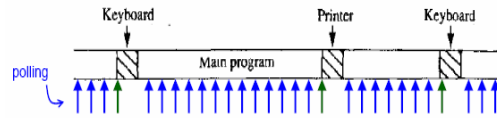
Interrupts

In this chapter, examples and detailed explanations of the interrupt structure of the entire Intel family of microprocessors will be provided.

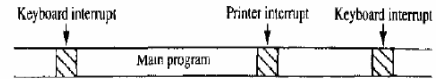
Slide 1

Programmed I/O versus Interrupt I/O

Programmed I/O:



Interrupted I/O:



A time line that indicates interrupt usage in a typical system

Slide 4

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.
- Develop interrupt service procedures that control lower speed external peripheral devices.
- Expand the interrupt structure of the microprocessor using the 8259A programmable interrupt controller and other techniques.

Slide 2

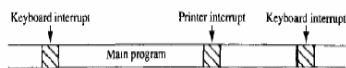
Interrupt Types

- The microprocessors also has **software interrupts** INT, INTO, INT 3 and BOUND.
Two flag bits, IF (interrupt flag) and TF (trap flag), are used with the interrupt structure and a **special return instruction IRET** (or IRETD in the 80386, 80486, or Pentium/Pentium Pro).
- The interrupts of the entire Intel family of microprocessors include **two hardware pins** that request interrupts (**INTR** and **NMI**) and one hardware pin (**INTA**) that acknowledges the interrupt requested through INTR. (**H/W Interrupt!**)

Slide 5

Basic Interrupt Processing

- Interrupts are particularly useful when interfacing I/O devices that provide or require data at relatively low data transfer rates.



A time line that indicates interrupt usage in a typical system

- Unlike the polling technique, interrupt processing allows the microprocessor to execute other software while there is no request for data transfer to or from the I/O devices.

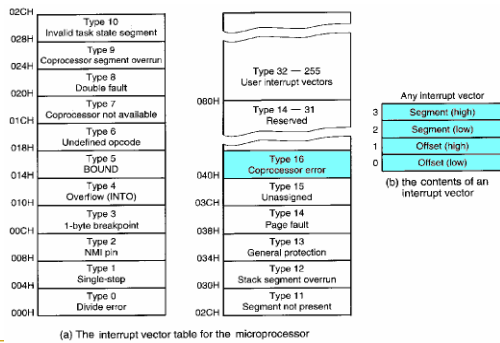
Slide 3

Interrupt Vectors

- The **interrupt table**
 - is located in the first 1024 bytes of memory at addresses 000000H-0003FFH.
 - contains **256** different **4-byte interrupt vectors**.
- Each **interrupt vector** contains the **starting address** (offset and segment) of the corresponding **interrupt service procedure**.

Slide 6

Interrupt Vectors



Slide 7

Interrupt Instructions

- The **IRET** instruction is a special return instruction used to return for both software and hardware interrupts. It is **different from the normal return** as it also **pops a copy of the flag register from the stack**.
- The **IRETD** instruction is a special return instruction used to return for both software and hardware interrupts in the protected mode in the **80386 and above microprocessors**. It **pops 32 bits EIP and EFLAG**.

Slide 10

Interrupt Instructions

- The **BOUND** instruction compares the content of a register with two words of memory data (lower and upper bounds).

A type 5 interrupt occurs if it is out of the bounds.

Example: **BOUND AX, DATA**

| | |
|--------|----------------------|
| DATA | Lower byte of B_L |
| DATA+1 | Higher byte of B_L |
| DATA+2 | Lower byte of B_U |
| DATA+3 | Higher byte of B_U |

If $[AX] > B_U$ or $[AX] < B_L$, a type 5 interrupt occurs.

Slide 8

Sequence of Events for a Real-time Interrupt

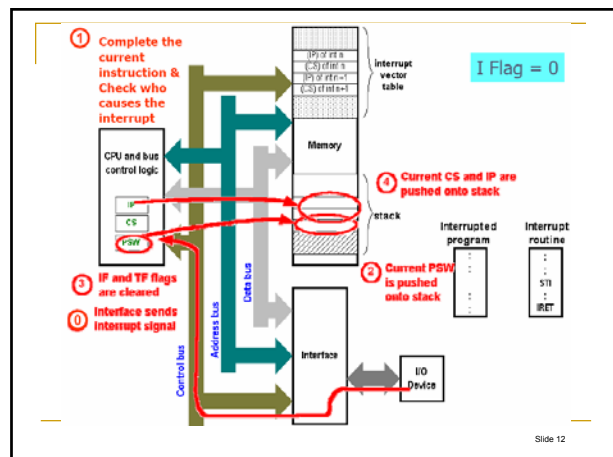
- Complete executing the **current instruction**.
- Check if the interrupt is activated by (1) instruction executions, (2) single-step, (3) NMI, (4) coprocessor segment overrun, (5) INTR, and (6) INT instruction in the order presented.
- Push the contents of the **flag register** onto the stack.
- Clear **IF** and **TF** to disable the INTR pin and the trap or single-step feature.
- Push the contents of **CS** onto the stack.
- Push the contents of **IP** onto the stack.
- Fetch and place the corresponding **interrupt vector** contents into both **IP** and **CS**.
- Run the **interrupt service procedure**.

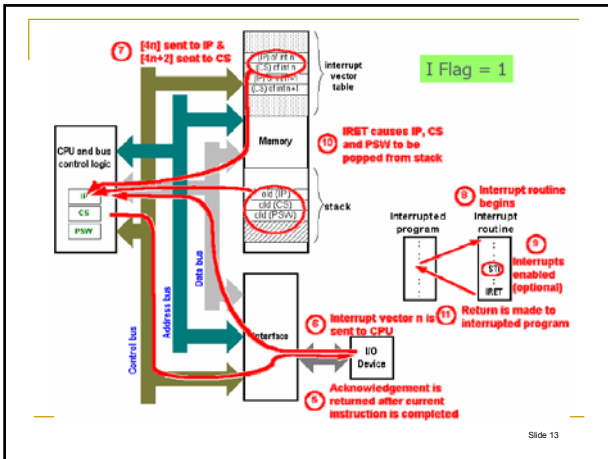
Slide 11

Interrupt Instructions

- The **INTO** instruction checks the overflow flag. A **type 4 interrupt** occurs if **OF=1**.
- The **INT n** instruction calls the interrupt service procedure that begins at the address represented in vector number n.
- The **INT 3** instruction is a 1-byte instruction and is often used as a **breakpoint interrupt**.

Slide 9





Interrupt Flag Bits – Set TF

You need working space, so temporarily borrow registers BP and AX. In order not to run the original data in BP & AX, we use the stack for temporal storage.

EXAMPLE 1

```

0000      ;A procedure that sets TF to enable trap.
          ;
0000      TRON PROC NEAR
0001      ;
0001      PUSH AX           ;save registers
0002      PUSH BP
0002      MOV BP,SP        ;get SP
0004      MOV AX,[BP+8]    ;get flags from stack
0007      OR AH,1         ;set TF
000A      MOV [BP+8],AX   ;save flags
000D      POP BP          ;restore registers
000E      POP AX
000F      IRET
          ;
0010      TRON ENDP      Resume BP & AX.
  
```

Slide 16

- ### Interrupt Flag Bits
- When the **interrupt flag (IF-bit)** is set, it allows the **INTR pin** to cause an interrupt.
 - The **IT-bit** is set and cleared by the **STI** and **CLI** instructions respectively.
 - When the **trap flag (TF-bit)** is set, it causes a **trap (single-step) interrupt** to occur after each instruction executes.
 - The TF-bit **cannot be directly set or cleared** with a single instruction.
- Slide 14

Interrupt Flag Bits – Clear TF

EXAMPLE 2

```

          ;A procedure that clears TF to disable trap.
          ;
0000      TROFF PROC NEAR
0001      ;
0001      PUSH AX           ;save registers
0002      PUSH BP
0002      MOV BP,SP        ;get SP
0004      MOV AX,[BP+8]    ;get TF
0007      AND AH,0FEH     ;clear TF
000A      MOV [BP+8],AX   ;save flags
000D      POP BP          ;restore registers
000E      POP AX
000F      IRET
          ;
0010      TROFF ENDP
  
```

Slide 17

- ### Interrupt Flag Bits
- Procedures for setting or clearing the trap flag:
 - Cause an interrupt.
 - Set or clear the trap flag via accessing the contents of the flag register pushed onto the stack.
 - resume the original program
- Slide 15