Bangladesh Army University of Engineering & Technology (BAUET)
Department of Computer Science and Engineering

Microprocessor and Micro-controller

Interrupts

Prof. Dr. Md. Rabiul Islam
Dept. of Computer Science & Engineering
Rajshahi University of Engineering & Technology
Rajshahi-6204, Bangladesh.
rabiul_cse@yahoo.com

Outline

- Overview of Interrupt
- Interrupt Services
- Types of Interrupt
  - Hardware Interrupt
  - Software Interrupt
  - Internal Interrupt
- Interrupt Vector
  - BIOS and DOS Interrupt
  - Interrupt Vector Table

References

Chapter 15 BIOS and DOS Interrupts, Yutha Yu and Charles Marut, “Assembly Language Programming and Organization of the IBM PC", McGraw-Hill International Edition, 1992.

Overview of Interrupts

- **Interrupt** is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor.

- The microprocessor responds to that interrupt with an **ISR** (Interrupt Service Routine), which is a short program to instruct the microprocessor on how to handle the interrupt.

Interrupt Services

Some questions to be answered are how does the 8086 find out a device is signaling? How does it know which interrupt routine to execute? How does it resume the previous task?

Because an interrupt signal may come at any time, the 8086 checks for the signal after executing each instruction. On detecting the interrupt signal, the 8086 acknowledges it by sending an **interrupt acknowledge signal**. The interrupting device responds by sending an eight-bit number on the data bus, called an **interrupt number**. Each device uses a different interrupt number to identify its own service routine. The process of sending control signals back and forth is called **hand-shaking**; it is needed to identify the interrupt device. We say that a type *N* interrupt occurs when a device uses an interrupt number *N* to interrupt the 8086.

The transfer to an interrupt routine is similar to a procedure call. Before transferring control to the interrupt routine, the 8086 first saves the address of the next instruction on the stack; this is the return address. The 8086 also saves the FLAGS register on the stack; this ensures that the status of the suspended task will be restored. It is the responsibility of the interrupt routine to restore any registers it uses.

Types of Interrupt

There are three types of interrupts:

1. Hardware Interrupt
   - NMI (Non-maskable Interrupt)
   - INTR (Interrupt Request)
2. Software Interrupt
3. Internal Interrupt

Hardware Interrupt

Hardware Interrupt

The notion of interrupt originally was conceived to allow hardware devices to interrupt the operation of the CPU. For example, whenever a key is pressed, the 8086 must be notified to read a key code into the keyboard buffer. The general **hardware interrupt** goes like this: (1) a hardware that needs service sends an **interrupt request signal** to the processor; (2) the 8086 suspends the current task it is executing and transfers control to an **interrupt routine**; (3) the interrupt routine services the hardware device by performing some I/O operation; and (4) control is transferred back to the original executing task at the point where it was suspended.

Hardware Interrupt

❑ NMI

It is a single non-maskable interrupt pin (NMI) having higher priority than the maskable interrupt request pin (INTR)and it is of type 2 interrupt.

When this interrupt is activated, these actions take place −

- Completes the current instruction that is in progress.
- Pushes the Flag register values on to the stack.
- Pushes the CS (code segment) value and IP (instruction pointer) value of the return address on to the stack.
- IP is loaded from the contents of the word location 00008H.
- CS is loaded from the contents of the next word location 0000AH.
- Interrupt flag and trap flag are reset to 0.

Hardware Interrupt

❑ INTR

The INTR is a maskable interrupt because the microprocessor will be interrupted only if interrupts are enabled using set interrupt flag instruction. It should not be enabled using clear interrupt Flag instruction.

The INTR interrupt is activated by an I/O port. If the interrupt is enabled and NMI is disabled, then the microprocessor first completes the current execution and sends '0' on INTA pin twice. The first '0' means INTA informs the external device to get ready and during the second '0' the microprocessor receives the 8 bit, say X, from the programmable interrupt controller.

These actions are taken by the microprocessor −

- First completes the current instruction.
- Activates INTA output and receives the interrupt type, say X.
- Flag register value, CS value of the return address and IP value of the return address are pushed on to the stack.
- IP value is loaded from the contents of word location X × 4
- CS is loaded from the contents of the next word location.
- Interrupt flag and trap flag is reset to 0

Software Interrupt

Software interrupts are used by programs to request system services. A **software interrupt** occurs when a program calls an interrupt routine using the INT instruction. The format of the INT instruction is

```
INT  interrupt-number
```

The 8086 treats this interrupt number in the same way as the interrupt number generated by a hardware device. We have already given a number of examples of doing I/O with INT 21h.

Internal Interrupt or Process Exception

There is a third kind of interrupt, called a **processor exception**. A processor exception occurs when a condition arises inside the processor, such as divide overflow, that requires special handling. Each condition corresponds to a unique interrupt type. For example, divide overflow is type 0, so when overflow occurs in a divide instruction the 8086 automatically executes interrupt 0 to handle the overflow condition.

Interrupt Vector

BIOS and DOS Interrupt:

- The computer manufacturer provides some hardware device service routines in ROM. These are the BIOS interrupt routine.

- The high level system interrupt routines likes INT 21h are part of DOS and are loaded into memory when the machine is started.

- Some additional interrupt numbers are reserved by IBM for further use and the remaining numbers are available for the user.

Interrupt Types 0–1Fh:	BIOS Interrupts
Interrupt Types 20h–3Fh:	DOS Interrupts
Interrupt Types 40h–7Fh:	reserved
Interrupt Types 80h–F0h:	ROM BASIC
Interrupt Types F1h–FFh:	not used

Interrupt Vector

Interrupt Vector Table:

The 8086 does not generate the interrupt routine's address directly from the interrupt number. Doing so would mean that a particular interrupt routine must be placed in exactly the same location in every computer—an impossible task, given the number of computer models and updated versions of the routines. Instead, the 8086 uses the interrupt number to calculate the address of a memory location that contains the actual address of the interrupt routine. This means that the routine may appear anywhere, so long as its address, called an **interrupt vector**, is stored in a predefined memory location.

All interrupt vectors are placed in an **interrupt vector table**, which occupies the first 1 KB of memory. Each interrupt vector is given as segment:offset and occupies four bytes; the first four bytes of memory contain interrupt vector 0.

To find the vector for an interrupt routine, we simply multiply the interrupt number by 4. This gives the memory location containing the offset of the routine; the segment address of the routine is in the next word. For example, take interrupt 9, the keyboard interrupt routine: the offset address is stored in location $9 \times 4 = 36 = 00024h$, and the segment address is found in location $24h + 2 = 00026h$. BIOS initializes its interrupt vectors when the computer is turned on, and the DOS interrupt vectors are initialized when DOS is loaded.

Interrupt Vector

Interrupt Vector Table:

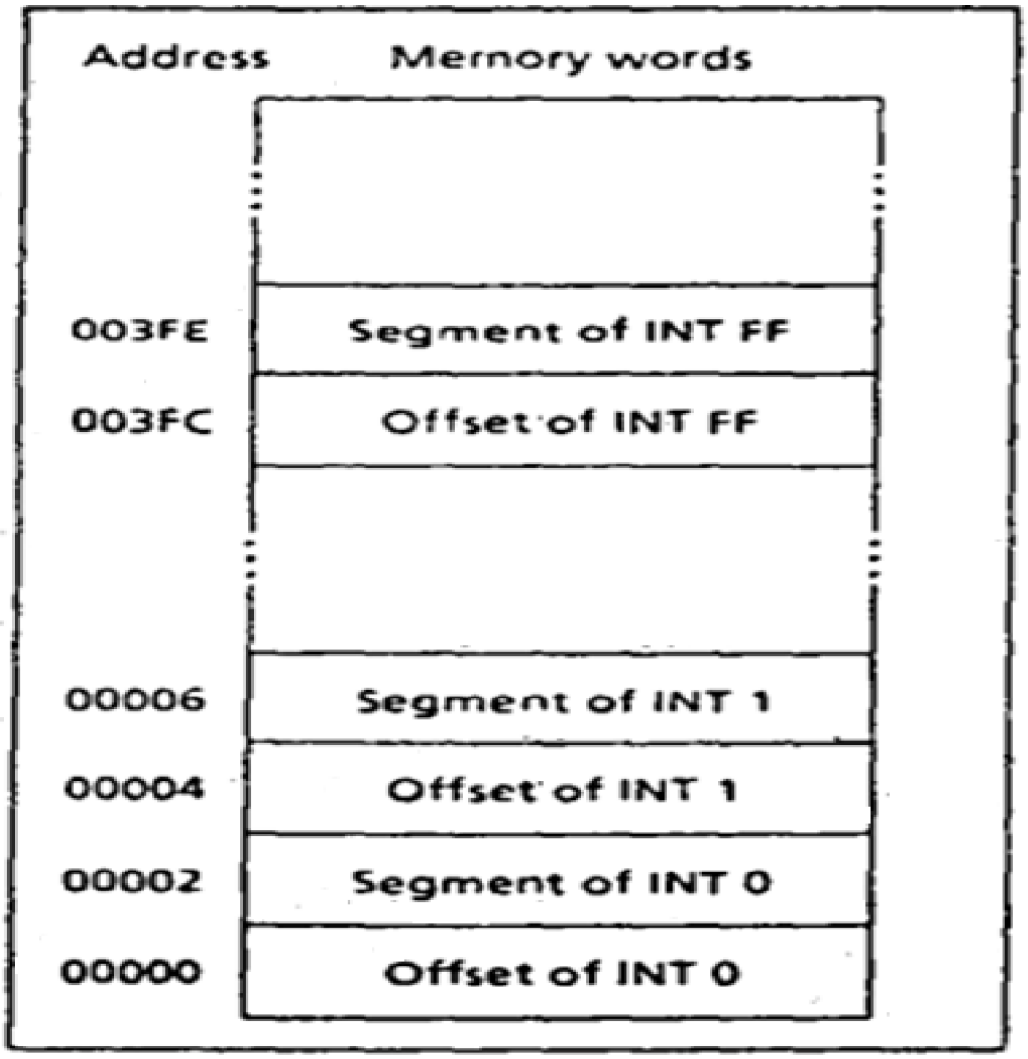

Figure: Interrupt Vector Table

Thanks

For any additional query please contact with me through

rabiul_cse@yahoo.com