6. Interrupt system. Interrupt table. Interrupt Acknowledge sequence.

An interrupt is a hardware-initiated procedure that interrupts whatever program is currently executed. Interrupts are particularly useful when interfacing I/O devices that provide or require data at relatively low data transfer rate.

An interrupt is used to cause a temporary halt in the execution of program.

- The meaning of 'interrupts' is to break the sequence of operations.
- While the Microprocessor is executing a program, an 'interrupt' breaks the normal sequence of execution of instructions, diverts its execution to some other program called Interrupt Service Routine (ISR).
- After executing ISR, IRET returns the control back again to the main program.

Interrupt processing is an alternative to polling. For example, getting the code, generated by the keyboard, when the user pushes a button, can be made by 2 ways:

-Polling :- The CPU executes a program that checks for the available of data, If a key is pressed, it reads the data, otherwise keep waiting or looping;

-Interrupt:-

The processor executes another program (main), when a key is pressed. The Keyboard generates an interrupt. The CPU will response to the interrupt and will read the data from keyboard. After that it returns to the original program. So by proper use of interrupt, the CPU can serve many devices "simultaneously".

Unlike the polling technique, interrupt processing allows the microprocessor to execute other software while the keyboard operator is thinking about what key to type next. As soon as a key is pressed, the keyboard encoder de-bounces the switch and puts out one pulse that interrupts the microprocessor. In this way, the microprocessor executes other software until the key is actually pressed when it reads a key and returns to the program that was interrupted. As a result, the microprocessor can print reports or complete any other task while the operator is typing a document and thinking about what to type next.

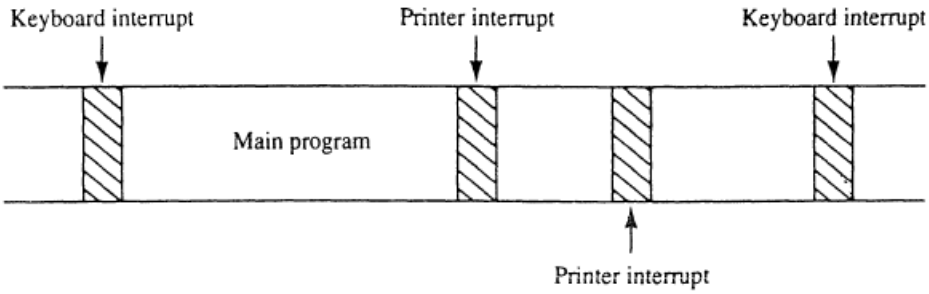

Fig. 11-1 A time line that indicates interrupt usage in a typical system

Figure 11–1 shows a time line that indicates a typist typing data on a keyboard, a printer removing data from the memory, and a program executing. The program is the main program that is interrupted for each keystroke and each character that is to print on the printer. Note that the keyboard interrupt service procedure, called by the keyboard interrupt, and the printer interrupt service procedure each take little time to execute.

If an interrupt has been requested, the 8086 Microprocessor processes it by performing the following series of steps (fig. 2):

1. Pushes the content of the flag register onto the stack to preserve the status of the interrupt (IF) and trap flags (TF)
2. Disables the INTR interrupt by clearing IF in the flag register
3. Resets TF in the flag register, to disable the single step or trap interrupt
4. Pushes the content of the code segment (CS) register onto the stack

5. Pushes the content of the instruction pointer (IP) onto the stack

6. Performs an indirect far jump to the start of the interrupt service routine (ISR), corresponding to the received interrupt.

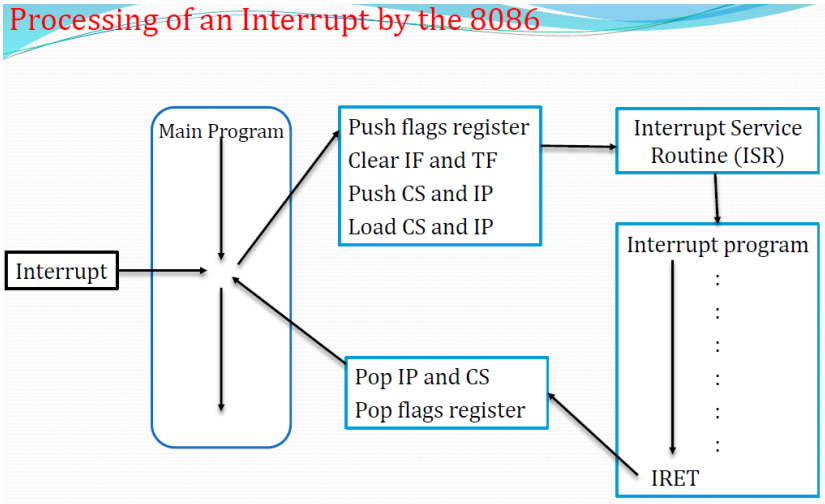

Fig. 2 8086 microprocessor interrupt processing

Interrupt types

Three types of interrupts sources are there (fig. 3):

1. An external signal applied to NMI or INTR input pin(hardware interrupt)
2. Execution of Interrupt instruction(software interrupt)
3. Interrupt raised due to some error condition produced in 8086 instruction execution process. (Divide by zero, overflow errors, etc)

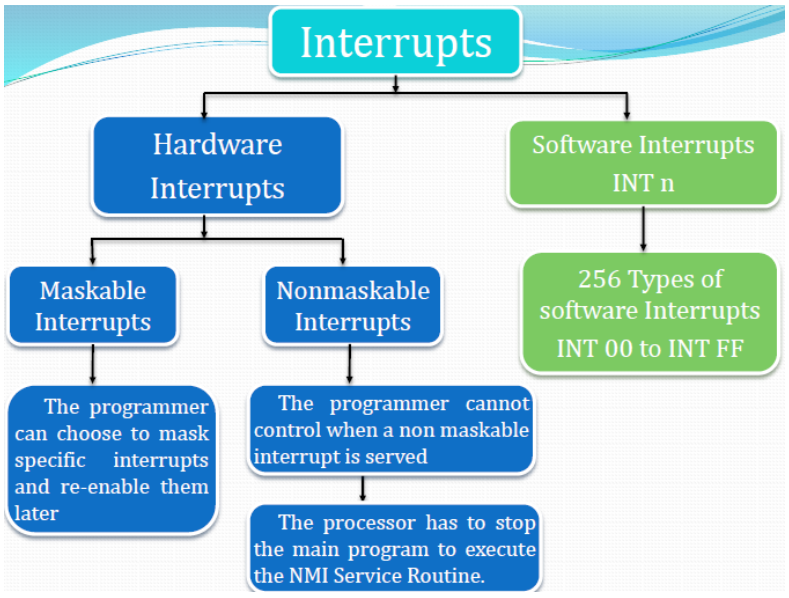

Fig. 3 Types of Interrupts, depending on their source

Vector Table occupies location 00000H to 0003FFh of the system memory. It contains the code segment (CS) and Instruction Pointer (IP) for each kind of interrupt.

Figure 11–2 illustrates the interrupt vector table for the microprocessor. The first five interrupt vectors are identical in all Intel microprocessor family members from the 8086 to the Pentium. Other interrupt vectors exist for the 80286 that are upward compatible to the 80386, 80486 and Pentium/Pentium Pro, but not downward compatible to the 8086 or 8088. Intel reserves the first 32 interrupt vectors for their use in various microprocessor family members. The last 224 vectors are available as user interrupt vectors. Each vector is four bytes long and contains the **starting address** of the interrupt service procedure. The first two bytes of the vector contain the offset address, and the last two bytes contain the segment address.

The following list describes the function of each dedicated interrupt in the microprocessor:

Type 0 Divide Error—Occurs whenever the result of a division overflows or whenever an attempt is made to divide by zero.

Type 1 Single-Step or Trap—Occurs after the execution of each instruction if the trap (TF) flag bit is set. Upon accepting this interrupt, the TF-bit is cleared so that the

Type 2 Non-maskable Hardware Interrupt—A result of placing a logic 1 on the NMI input pin to the microprocessor. This input is non-maskable, which means that it cannot be disabled.

Type 3 One-Byte Interrupt—A special 1-byte instruction (INT 3) that uses this vector to access its interrupt service procedure. The INT 3 instruction is often used to store a breakpoint in a program for debugging.

Type 4 Overflow—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).

The INTO instruction checks the overflow flag (OF). If OF = 1, the INTO instruction calls the procedure whose address is stored in interrupt vector type number 4. If OF = 0, then the INTO instruction performs no operation and the next sequential instruction in the program executes.

The INT n instruction calls the interrupt service procedure that begins at the address represented in vector number n. For example, an INT 80H or INT 128 call the interrupt service procedure whose address is stored in vector type number 80H (000200H–00203H). To determine the vector address, just multiply the vector type number (n) by 4. This gives the beginning address of the 4-byte long interrupt vector. For example, an INT 5 = 4 × 5 or 20 (14H). The vector for INT 5 begins at address 000014H and continues to 000017H. Each INT instruction is stored in two bytes of memory with the first byte containing the opcode and the second the interrupt type number. The only exception to this is the INT 3 instruction, a 1-byte instruction. The INT 3 instruction is often used as a breakpoint interrupt because it is easy to insert a 1-byte instruction into a program. Breakpoints are often used to debug faulty software.

The IRET instruction is a special return instruction used to return for both software and hardware interrupts. The IRET instruction is much like a normal far RET, because it retrieves the return address from the stack. It is unlike the normal return because it also retrieves a copy of the flag register from the stack. An IRET instruction removes six bytes from the stack: two for the IP, two for the CS, and two for the flags.

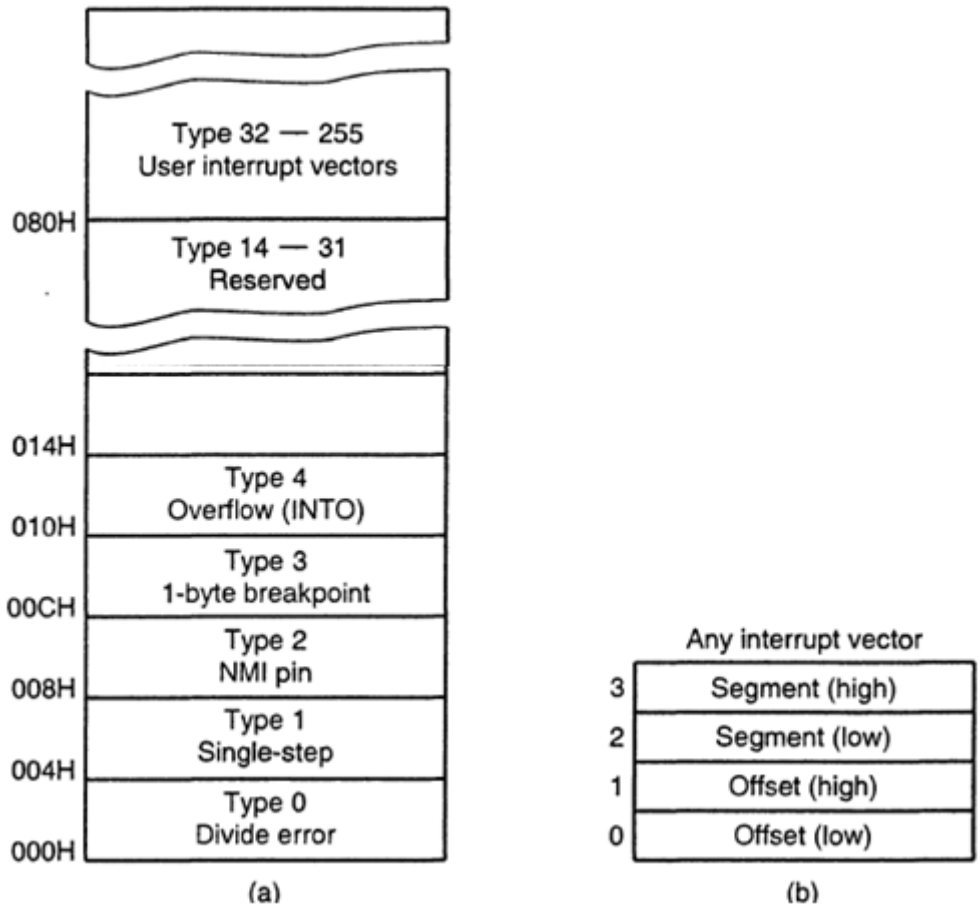

Fig. 11-2 (a) The interrupt vector table, and (b) the contents of interrupt vector

When the microprocessor completes executing the current instruction, it determines whether an interrupt is active by checking (1) instruction executions, (2) single-step, (3) NMI, (4) co-processor segment overrun, (5) INTR, and (6) INT instruction in the order presented. If one or more of these interrupt conditions are present, the following sequence of events occurs:

1. The contents of the flag register are pushed onto the stack.
2. Both the interrupt (IF) and trap (TF) flags are cleared. This disables the INTR pin and also the trap or single-step feature.
3. The contents of the code segment register (CS) are pushed onto the stack.
4. The contents of the instruction pointer (IP) are pushed onto the stack.
5. The interrupt vector contents are fetched and placed into both IP and CS so that the next instruction executes at the interrupt service procedure addressed by the vector.

Whenever an interrupt is accepted, the microprocessor stacks the contents of the flag register, CS and IP; clears both IF and TF; and jumps to the procedure addressed by the interrupt vector. After the flags are pushed onto the stack, IF and TF are cleared. These flags are returned to the state prior to the interrupt when the IRET instruction is encountered at the end of the interrupt service procedure. Therefore, if interrupts were enabled prior to the interrupt service procedure, they are automatically re-enabled by the IRET instruction at the end of the procedure.

The return address (in CS and IP) is pushed onto the stack during the interrupt. Sometimes the return address points to the next instruction in the program, and sometimes it points to the instruction or point in the program where the interrupt occurred.

Interrupt flags 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. When the IF bit is set, it allows the INTR pin to cause an interrupt, and

when the IF-bit is cleared, it prevents the INTR pin from causing an interrupt. When TF = 1, it causes a trap interrupt (type number 1) to occur after each instruction executes. This is why we often call trap a *single-step*. When TF = 0, normal program execution occurs. This flag bit allows debugging, as explained in later chapters that detail the 808386–Pentium Pro.

The interrupt flag is set and cleared by the STI and CLI instructions respectively. There are no special instructions that set or clear the trap flag.

HARDWARE INTERRUPTS

The microprocessor has two hardware interrupt inputs: non-maskable interrupt (NMI) and interrupt request (INTR). Whenever the 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. The $\overline{\text{INTA}}$ signal is also an interrupt pin on the microprocessor, but it is an output that is used in response to the INTR input to apply a vector type number to the data bus connections D7–D0. Figure 11–5 shows the three user interrupt connections on the 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 a logic 1 until it is recognized by the microprocessor. Note that before the positive edge is recognized, the NMI pin must be a logic 0 for at least two clocking periods.

The NMI input is often used for parity errors and other major system faults such as power failures. Power failures are easily detected by monitoring the AC power line and causing an NMI

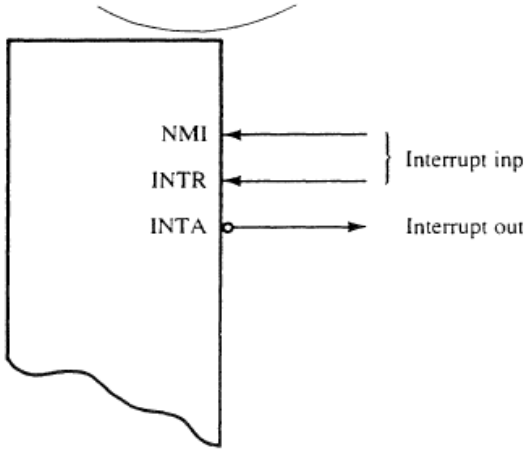

FIGURE 11–5 The interrupt pins on all versions of the Intel microprocessor

interrupt whenever AC power drops out, for example.

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. The INTR pin is set by an external event and cleared inside the interrupt service procedure. This input is automatically disabled once it is accepted by the microprocessor and re-enabled by the IRET instruction at the end of the interrupt service procedure. The 80386–Pentium Pro use the IRETD instruction in the protected mode of operation.

The microprocessor responds to the INTR input by pulsing the $\overline{\text{INTA}}$ output in anticipation of receiving an interrupt vector type number on data bus connection D_7–D_0. Figure 11–8 shows the timing diagram for the INTR and $\overline{\text{INTA}}$ pins of the microprocessor. There are two $\overline{\text{INTA}}$ pulses generated by the system that are used to insert the vector type number on the data bus.

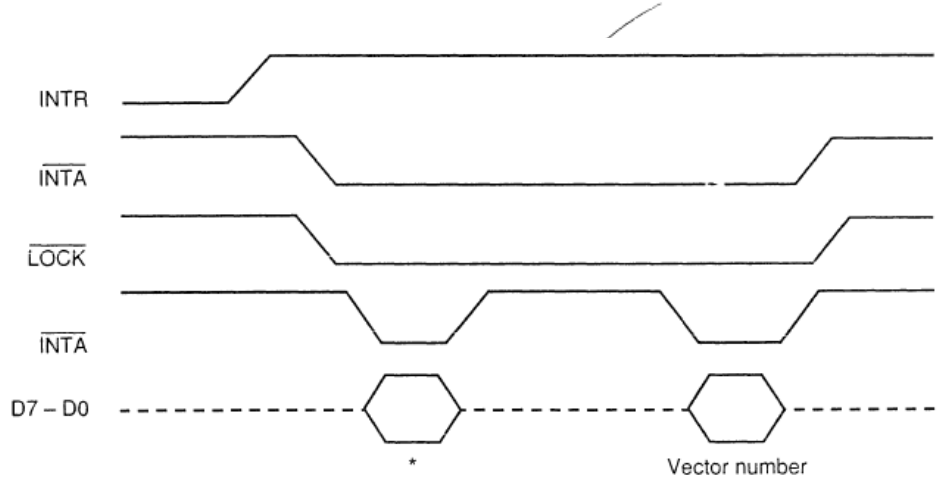

FIGURE 11–8 The timing of the INTR input and INTA output. *Note: This portion of the data bus is ignored and usually contains the vector number.

SUMMARY

1. 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.
2. Interrupts are useful when an I/O device needs to be serviced only occasionally at low data transfer rates.
3. 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 interrupt vector whose type is indicated by the instruction. The BOUND instruction is a conditional interrupt that uses interrupt vector type number 5. The INTO instruction is a conditional interrupt that interrupts a program only if the overflow flag is set. Finally, the IRET instruction is used to return from interrupt service procedures.
4. The microprocessor has three pins that apply to its hardware interrupt structure: INTR, NMI, and $\overline{\text{INTA}}$. The interrupt inputs are INTR and NMI, which are used to request interrupts. $\overline{\text{INTA}}$ is an output used to acknowledge the INTR interrupt request.
5. Real mode interrupts are referenced through a vector table that occupies memory locations 00000H–003FFH. 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. Each interrupt descriptor contains a segment selector and a 32-bit offset address.

6. 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, and the TF flag bit causes interrupts to occur after the execution of each instruction as long as TF is active.
7. The first 32 interrupt vector locations are reserved for Intel use, with many predefined in the microprocessor. The last 224 interrupt vectors are for user use and can perform any function desired.
8. 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, and (4) the interrupt vector is fetched from the interrupt vector table and the interrupt service subroutine is accessed through the vector address.
9. 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.
10. 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.
11. The INTR pin is not internally decoded as is the NMI pin. Instead, $\overline{\text{INTA}}$ is used to apply the interrupt vector type number to data bus connections D_0–D_7 during the $\overline{\text{INTA}}$ pulse.
12. Methods of applying the interrupt vector type number to the data bus during $\overline{\text{INTA}}$ vary widely. One method uses resistors to apply interrupt type number FFH to the data bus, while another uses a three-state buffer to apply any vector type number.