6–4 INTRODUCTION TO INTERRUPTS

An interrupt is either a **hardware-generated CALL** (externally derived from a hardware signal) or a **software-generated CALL** (internally derived from the execution of an instruction or by some other internal event). At times, an internal interrupt is called an *exception.* Either type interrupts the program by calling an **interrupt service procedure** (ISP) or interrupt handler.

This section explains software interrupts, which are special types of CALL instructions. This section describes the three types of software interrupt instructions (INT, INTO, and INT 3), provides a map of the interrupt vectors, and explains the purpose of the special interrupt return instruction (IRET).

Interrupt Vectors

An **interrupt vector** is a 4-byte number stored in the first 1024 bytes of the memory (00000H–003FFH) when the microprocessor operates in the real mode. In the protected mode, the vector table is replaced by an interrupt descriptor table that uses 8-byte descriptors to describe each of the interrupts. There are 256 different interrupt vectors, and each vector contains the address of an interrupt service procedure. Table 6–4 lists the interrupt vectors, with a brief description and the memory location of each vector for the real mode. Each vector contains a value for IP and CS that forms the address of the interrupt service procedure. The first 2 bytes contain the IP, and the last 2 bytes contain the CS.

TABLE 6–4 Interrupt vectors defined by Intel.

Number	*Address*	*Microprocessor*	*Function*
0	0H–3H	All	Divide error
1	4H–7H	All	Single-step
2	8–BH	All	NMI pin
3	CH–FH	All	Breakpoint
4	10H–13H	All	Interrupt on overflow
5	14H–17H	80186–Core2	Bound instruction
6	18H–1BH	80186–Core2	Invalid opcode
7	1CH–1FH	80186–Core2	Coprocessor emulation
8	20H–23H	80386–Core2	Double fault
9	24H–27H	80386	Coprocessor segment overrun
A	28H–2BH	80386–Core2	Invalid task state segment
B	2CH–2FH	80386–Core2	Segment not present
C	30H–33H	80386–Core2	Stack fault
D	34H–37H	80386–Core2	General protection fault (GPF)
E	38H–3BH	80386–Core2	Page fault
F	3CH–3FH	—	Reserved
10	40H–43H	80286–Core2	Floating-point error
11	44H–47H	80486SX	Alignment check interrupt
12	48H–4BH	Pentium–Core2	Machine check exception
13–1F	4CH–7FH	—	Reserved
20–FF	80H–3FFH	—	User interrupts

Intel reserves the first 32 interrupt vectors for the present and future microprocessor products. The remaining interrupt vectors (32–255) are available for the user. Some of the reserved vectors are for errors that occur during the execution of software, such as the divide error interrupt. Some vectors are reserved for the coprocessor. Still others occur for normal events in the system. In a personal computer, the reserved vectors are used for system functions, as detailed later in this section. Vectors 1–6, 7, 9, 16, and 17 function in the real mode and protected mode; the remaining vectors function only in the protected mode.

Interrupt Instructions

The microprocessor has three different interrupt instructions that are available to the programmer: INT, INTO, and INT 3. In the real mode, each of these instructions fetches a vector from the vector table, and then calls the procedure stored at the location addressed by the vector. In the protected mode, each of these instructions fetches an interrupt descriptor from the interrupt descriptor table. The descriptor specifies the address of the interrupt service procedure. The interrupt call is similar to a far CALL instruction because it places the return address (IP/EIP and CS) on the stack.

INTs. There are 256 different software interrupt instructions (INTs) available to the programmer. Each INT instruction has a numeric operand whose range is 0 to 255 (00H–FFH). For example, the INT 100 uses interrupt vector 100, which appears at memory address 190H–193H. The address of the interrupt vector is determined by multiplying the interrupt type number by 4. For example, the INT 10H instruction calls the interrupt service procedure whose address is stored beginning at memory location 40H (10H × 4) in the real mode. In the protected mode, the interrupt descriptor is located by multiplying the type number by 8 instead of 4 because each descriptor is 8 bytes long.

Each INT instruction is 2 bytes long. The first byte contains the opcode, and the second byte contains the vector type number. The only exception to this is INT 3, a 1-byte special software interrupt used for breakpoints.

Whenever a software interrupt instruction executes, it (1) pushes the flags onto the stack, (2) clears the T and I flag bits, (3) pushes CS onto the stack, (4) fetches the new value for CS from the interrupt vector, (5) pushes IP/EIP onto the stack, (6) fetches the new value for IP/EIP from the vector, and (7) jumps to the new location addressed by CS and IP/EIP.

The INT instruction performs as a far CALL except that it not only pushes CS and IP onto the stack, but it also pushes the flags onto the stack. The INT instruction performs the operation of a PUSHF, followed by a far CALL instruction.

Notice that when the INT instruction executes, it clears the interrupt flag (I), which controls the external hardware interrupt input pin INTR (interrupt request). When I = 0, the microprocessor disables the INTR pin; when I = 1, the microprocessor enables the INTR pin.

Software interrupts are most commonly used to call system procedures because the address of the system function need not be known. The system procedures are common to all system and application software. The interrupts often control printers, video displays, and disk drives. Besides relieving the program from remembering the address of the system call, the INT instruction replaces a far CALL that would otherwise be used to call a system function. The INT instruction is 2 bytes long, whereas the far CALL is 5 bytes long. Each time that the INT instruction replaces a far CALL, it saves 3 bytes of memory in a program. This can amount to a sizable saving if the INT instruction often appears in a program, as it does for system calls.

IRET/IRETD. The interrupt return instruction (IRET) is used only with software or hardware interrupt service procedures. Unlike a simple return instruction (RET), the IRET instruction will (1) pop stack data back into the IP, (2) pop stack data back into CS, and (3) pop stack data back into the flag register. The IRET instruction accomplishes the same tasks as the POPF, followed by a far RET instruction.