Interrupts

When the CPU detects an interrupt signal, it stops the current activity and jumps to a special routine, called an interrupt handler. This handler then detects why the interrupt occurred and takes the appropriate action. When the handler is finished executing this action, it jumps back to the interrupted process.

Several levels or "types" of interrupts are supported, ranging from 0 to 255. Each type has a reserved memory location, called an interrupt vector. The interrupt vector points to the appropriate interrupt handler. When two or more interrupts occur at the same time, the CPU uses a priority system The 256 priority levels supported by the Intel 8086-processors can be split into three categories:

- Internal Hardware-Interrupts
- External Hardware-Interrupts
- Software Interrupts

Internal Hardware-Interrupts

Internal hardware-interrupts are the result of certain situations that occur during the execution of a program, e.g. divide by zero. The interrupt levels attached to each situation are stored in hardware and cannot be changed.

Divide by zero	00h	Single Step	04h
NMI	08h	Breakpoint	0ch
Overflow			

External Hardware-Interrupts

External hardware-interrupts are produced by controllers of external devices or coprocessors and are linked to the processor pin for Non Maskable Interrupts (NMI) or to the pin for Maskable Interrupts (INTR). The NMI line is usually reserved for interrupts that occur because of fatal errors like a parity error or a power distortion.

Interrupts from external devices can also be linked to the processor via the Intel 8259A Programmable Interrupt Controller (PIC). The CPU uses a group of I/O ports to control the PIC and the PIC puts its signals on the INTR pin. The PIC makes it possible to enable or disable interrupts and to change the priority levels under supervision of a program.

The instructions STI and CLI can be used to enable/disable interrupts on the INTR pin, this has no effect on NMI interrupts.

Software Interrupts

Software interrupts are the result of an INT instruction in an executed program. This can be seen as a programmer triggered event that immediately stops execution of the program and passes execution over to the INT handler. The INT handler is usually part of the operating system and will determine the action that should be taken (e.g. output to screen, execute file etc.) An example is INT 21h, which is the DOS service interrupt. When the handler is called it will read the value stored in AH (sometimes even AL) and jumps to the right routine.

Interrupt Table

Each interrupt level has a reserved memory location, called an interrupt vector. All these vectors (or pointers) are stored in a table, the interrupt table. This table lies at linear address 0, or with 64KB segments, at 0000:0000. Each vector is 2 words long (4 bytes). The high word contains the offset and the low word the segment of the INT handler.

Since there are 256 levels and each vector is 4 bytes long, the table contains 1024 bytes (256*4=1024). The INT number is multiplied by 4 to fetch the address from the table.

How INT's are Processed

When the CPU registeres an INT it will push the FLAGS register to the stack and it will also push the CS and IP registers. After that the CPU disables the interrupt system. Then it gets the 8-bit value the interrupting device sends and multiplies this by 4 to get the offset in the interrupt table. From this offset it gets the address of the INT handler and carries over execution to this handler.

The handler usually enables the interrupt system immediately, to allow interrupts with higher priority. Some devices also need a signal that the interrupt has been acknowledged. When the handler is finished it must signal the 8259A PIC with an EOI (End Of Interrupt). Then the handler executes an IRET instruction