Published: 11/10/66

Identification

Overview of Interrupt Handling
J. H. Saltzer

Discussion

An Interrupt, by Multics definition, is caused by a signal from some source other than a condition within the hardware of the processor. An interrupt may be triggered, for example, by an I/O device, or another processor. (A condition detected within the processor such as accumulator overflow or addressing failure, causes a Fault. Cf. BK.3, below.). An interrupt signal may be temporarily ignored, although remembered (inhibited), in one of two ways:

1. by operating the processor in inhibited mode. This mode will inhibit all interrupts.

2. by placing a mask on selected interrupt sources. Only interrupts arising from those sources will be inhibited.

An important difference between inhibited mode and masking is that a processor must be executing in a master mode segment in order to remain in inhibited mode. On the other hand, once a mask is set, by a master mode instruction, the mask remains effective even when control passes to a slave procedure.

Interrupts occuring in the Multics system are divided into two mutually exclusive categories, named system and process. The system interrupts arise from I/O devices and the Calendar Clock. Process interrupts are triggered directly or indirectly by a processor operating in the Traffic Controller module of the supervisor. System interrupts may be viewed as signals from outside the system directed to some process in the system, generally not the one running on the processor being interrupted. System interrupts are interpreted to have the meaning "start doing something" to some process in the system.

Process interrupts, on the other hand, are directed by the Traffic Controller to the process currently executing on some processor. Process interrupts are interpreted to mean: "change your execution state" to the process which is interrupted. As might be expected, system and process interrupts are handled in quite different patterns.

All interrupts are passed directly to the Interrupt Interceptor module by a transfer instruction in the processor interrupt vector. This transfer instruction is set up at system initialization or reconfiguration time. The Interrupt Interceptor is a standard Multics procedure segment, except for its unorthodox method of entry; it operates in master mode, but not absolute mode.

The Interrupt Interceptor, after safe-storing machine conditions (as described in BK.1), calls one of several interrupt handlers to service the particular interrupt which occurred. The handler, a slave mode procedure, performs the appropriate action, and returns to the Interrupt Interceptor, which restores the processor state and returns to the interrupted program.

Interrupts are assigned to priority classes; in general when an interrupt of priority class "X" occurs, the Interrupt Interceptor masks all interrupts of priority equal to or below "X". The class assignment is accomplished by a table which is read into core at system initialization time. While handling a system interrupt, the Interrupt Interceptor and the handler it calls use the processor stack (BK.1.03), a wired-down storage area in the processor data segment, for calls and temporary storage; correspondingly, while handling process interrupts, the process concealed stack in the process data segment is used. "Cascaded" interrupts (a high-priority interrupt interrupting handling of a low-priority interrupt) are thereby allowed. The maximum number of cascaded external interrupts, determined by the number of interrupt classes a processor is responsible for, helps establish the required size of the processor data segment for that processor.

System Interrupts

The general philosophy of handling system interrupts in Multics is the following: System interrupts occur at unpredictable and unscheduled times. To minimize the effect of an interrupt upon orderly scheduling, the procedure executed following the instant of the interrupt is restricted to identifying the cause of the interrupt sufficiently to wake up (schedule) an appropriate process to complete its handling. The time required to handle the interrupt is carefully accounted for and charged to the process which is responsible for the interrupt. The execution meter is adjusted so that the process that was interrupted will neither be charged for nor affected by the time spent handling the interrupt.

The following system interrupts may occur:

1. MSU-302 (3 interrupts per Drum Controller)
2. GIOC (8 interrupts per controller)
3. Calendar Clock (2 interrupts per clock)

Process Interrupts

The process interrupts have a specific meaning for the process which was interrupted. For these interrupts, the Interrupt Interceptor safestores machine conditions in the process concealed stack rather than the processor stack. After masking any future Process Interrupts, it calls the Process Interrupt Handler. The Process Interrupt Handler calls various entries of the Process Exchange, depending on which process interrupt happened; control of the processor may vanish from the process while in the Process Exchange. When control returns to the Process Interrupt Handler, it returns to the Interrupt Interceptor, which restores the processor state and restarts the interrupted procedure. The following process interrupts may occur:

1. Pre-emption interrupt (Handler should call restart).
2. Time-out interrupt (Handler should call restart or possibly Wakeup and Block. Note that the single interval timer in each processor is shared by the Traffic Controller, for pre-emption, and by the user, for loop limiting and program timing).
3. Quit Interrupt (Handler should call Block).