Published: 09/29/67
(Supersedes: BK.3.03, 06/13/66)

Identification

The Fault Interceptor
Chester Jones

Purpose

The Fault Interceptor Module is the interface between the hardware fault mechanism and the procedures for handling the various fault conditions. The Fault Interceptor Module is responsible for saving the processor state when that processor faults, for calling the appropriate procedure for handling the fault, and for restoring the processor state after control returns from the fault handling procedure. The supervisor protection mechanism (Section BD.9) requires that crossing a wall in either direction be detected by a fault. Since it contains parts of the wall crossing mechanism, the Fault Interceptor Module is accessible in every ring in the sense that it can be entered without first crossing a wall, and it is able to switch rings when necessary. The Fault Interceptor Module must execute privileged hardware instructions and inhibit interrupts during certain crucial operations. Therefore, the Fault Interceptor Module is a hand-coded, master mode procedure that can be entered only as a result of a hardware fault condition.

Summary of the Fault Interceptor Actions.

When a Multics processor generates a fault, control passes (automatically, through the processor fault vector) to the Fault Interceptor Module which executes as part of the process that is running at the instant the fault occurs. While executing within the ring in which the fault occurs, the Fault Interceptor Module temporarily saves the processor state in the Process Concealed Stack (Section BJ.1.05) that belongs to the running process and makes space available for safe-storing the processor state should another fault occur. Then, the actions of the Fault Interceptor Module vary, depending on the probable cause of the fault.

For process faults, the Fault Interceptor Module performs the following steps:

1. Determines the number of the protection ring in which the fault is to be handled. If that ring number is not the same as the ring in which the fault occurred, then the FIM calls the Gatekeeper to switch to the appropriate ring.

2. Copies the safe-stored processor state into the paged stack for the ring in which the fault condition is to be signalled.

3. Uses the paged stack for that protection ring to call signal to indicate the occurrence of that fault condition.

4. Following a return from signal, copies the machine conditions from the paged stack into the Process Concealed Stack.

5. Checks the validity of the safe-stored processor state.

6. Switches back to the ring in which the fault occurred if necessary.

7. Restores the processor state to return control to the point at which the fault occurred.

For system faults (except for missing-page faults, connect faults, and timer runout faults), the Fault Interceptor Module performs the following steps:

1. Switches control to the hard core ring.

2. Switches from the Process Concealed Stack to a paged stack in the hard core ring.

3. Calls the procedure for handling that fault.

4. Switches back to the Process Concealed Stack from the paged stack in the hard core ring.

5. Switches control back to the ring in which the fault occurred.

6. Restores the processor state to return control to the point at which the fault occurred.

(For a description of the actions of the Fault Interceptor Module in response to missing-page faults, timer runout faults, and connect faults, see Sections BK.3.06, BK.3.07, and BK.3.08 respectively.)

Actions of the Fault Interceptor Module.

1. The processor state is stored in the current interrupt frame of the Process Concealed Stack. Pointers to the current interrupt frame are maintained at the base of the Process Concealed Stack. The processor state is stored in four steps:

a. The processor control unit is stored (store control unit instruction) using the scu pointer. (This instruction is executed in the processor fault vector.)

b. The arithmetic registers are stored (store registers instruction) using the sreg pointer.

c. The address base registers are stored (store bases instruction) using the stb pointer.

d. The simulated ring register contents are obtained from the Process Data Segment (Section BJ.1.03) and stored. (Actually, this step is not taken until after the linkage base registers are established. Eventually, if a hardware ring register is added to the GE-645, that register should be stored with the processor control unit using an scu instruction, making this step unnecessary.)

2. The values for the linkage pointer and linkage base registers are loaded into the lp-lb base register pair. (Since the Fault Interceptor Module is not called, it must establish its own linkage values.) The values to be loaded into lp-lb are determined at system initialization or reconfiguration time and stored "inside" the FIM procedure. This enables the FIM to establish its own linkage values using an "internal" address.

3. A new interrupt frame is allocated in the Process Concealed Stack (i.e. the Concealed Stack is pushed down) in preparation for the next fault (or process interrupt). Since a processor may fault (or accept a process interrupt) in the course of handling an earlier fault (or process interrupt), care is taken to prevent a possible overlapping of the new interrupt frame and the interim stack in use at the instant of the fault. A new interrupt frame is allocated as follows:

   a. The value of the stack base register, sb, (stored in Step 1) is examined. If the stack base register, sb, does <u>not</u> contain the segment number of the Process Data Segment, then the new interrupt frame is allocated immediately following the current interrupt frame. (It is important to emphasize that sb indicates whether the new interrupt frame should be allocated relative to the stb pointer or relative to sp|18.) In this case, the base location of the new interrupt frame is obtained by adding 32 (i.e. the length of an interrupt frame) to the stb pointer stored at the base of the Process Concealed Stack.

If the stack base register contains the segment number of the Process Data Segment, then the sp-sb base register pair points to the base location of the stack frame in use at the instant of the fault and sp|18 points to the base location of the next (intended) stack frame (i.e. next sp). In this case, a constant, k, is added to the value of next sp to obtain the base location of the new interrupt frame. (Currently, k is equal to 32.)

b. A back pointer to the previous interrupt frame is fabricated and stored into locations 31-32 of the new interrupt frame.

c. The stb, sreg, and scu pointers stored at the base of the Process Concealed Stack are adjusted to point to the appropriate areas within the new interrupt frame.

4. A new interim stack is created immediately following the just-allocated interrupt frame. The sp-sb address base register pair is set to point to the base location of the new interim stack. The value of the next sp is determined and stored at sp|18 in the first stack frame of the new interim stack. The amount of temporary storage required by the FIM is used in determining the value to be stored at sp|18. This step enables the FIM to use temporary storage.

5. The safe-stored machine conditions are examined to determine whether the fault is a system fault or a process fault. Usually, the fault code captured by the processor is enough to identify a particular fault; however, in some cases, other information captured by the processor must be used. For example, outward wall crossing attempts are detected by "attempt-to-execute-data" faults. In fact, no such fault exists on the GE-645. When an outward wall crossing attempt is made, an "illegal procedure" fault occurrs. The FIM examines the machine conditions to determine whether the illegal procedure fault was caused by an access violation. If so, the FIM determines whether the segment being accessed is a data segment. If so, the FIM determines which (even or odd) instruction was being executed and examines that instruction to determine whether it is a transfer-type instruction. For system faults only, the following steps are taken. (For process faults, skip to step 14.)

6. Control is switched to the hardcore protection ring of the running process with the following call;

```
call ring$load (0)
```

(See MSPM Section BG.3.05 for a description of ring$load.)

7. The safe-stored machine conditions are copied from the Process Concealed Stack into the paged fault-stack that belongs to the running process.

   a. The values of the stack base register pair, sp-sb, are obtained. The stack pointer register, sp, points to the base location of the stack frame in use and sp|18 points to the base location of the next (intended) stack frame.

   b. The interrupt frame containing the processor state (Step 1) is copied from the Process Concealed Stack into the paged stack. (Since the Process Concealed Stack has been "pushed down," a missing-page fault during the attempt to copy the processor state is acceptable.) The base location of the interrupt frame into which the processor state is copied is obtained by adding 32 to the value of next sp.

   c. An interim stack is created immediately following the interrupt frame. The first stack frame of the new interim stack is initialized.

   d. The stack base register pair, sp-sb, is set to point to the base location of the newly created interim stack.

8. The space required for temporarily saving the processor state is returned to the Process Concealed Stack (i.e. the Process Concealed Stack is "popped-up" one level.) The values for the stb, sreg, and scu pointers are derived from the back pointer stored in locations 31-32 of the current interrupt frame.

9. A standard CALL is issued to the appropriate module to handle the fault. This module may CALL other modules. Until a RETURN is made to the Fault Interceptor Module, all steps for handling the fault are taken by other procedures. When control returns to the Fault Interceptor Module, the sequence of steps presented here continues.

10. Control is returned to the ring in which the fault occurred with the following call;

    call ring$load (n)

11. The machine conditions are copied from the paged fault-stack into the current interrupt frame of the Process Concealed Stack.

12. The safe-stored processor state is checked to insure that the control unit information is valid.

13. The processor state is restored to return control to the point at which the fault occurred.

The following steps continue after step 5, if a process fault is detected.

14. A test is made to determine whether the condition associated with the process fault may be signalled in the protection ring in which it occurred or whether that condition must be signalled in a specific protection ring. If the condition must be signalled in a ring other than the one in which it occurred, the FIM must switch to the appropriate ring. Steps 6-8 (above) are executed in order to switch control (temporarily) to the hardcore ring. Then, the Gatekeeper is called to perform the housekeeping to switch to the protection ring in which the condition is to be signalled. Finally, control is switched to the appropriate protection ring by calling ring$load.

15. A "placeholder" frame is built in the stack for the protection ring in which the fault is to be signalled. (Note that this step is taken by the Gatekeeper in step 14 if the condition is to be signalled in a ring other than the one in which it occurred.) MSPM Section BD.9.01 contains a detailed description of how "placeholder" stack frames are constructed.

16. The machine conditions associated with the fault are deposited in the "placeholder" frame.

17. The condition associated with this fault is signalled as follows;

    call signal (conditionname, flag, meptr)

    Until a return is made to the FIM, all steps taken for handling the fault are taken by other procedures. When control returns to the FIM, the sequence of steps presented here continues.

18. If necessary, control is switched to the ring in which the fault occurred. (See step 14.)

19. The machine conditions are copied from the paged stack into the current interrupt frame of the Process Concealed Stack.

20. The copy of the machine conditions modified by the fault handling procedure is checked to insure that the control unit information is valid.

21. The processor state is restored to return control to the point at which the fault occurred.