Published: 10/02/67
(Supersedes: BK.3.06, 09/13/66)

Identification

Page/Segment Fault Handler
Chester Jones

Purpose

This section describes the actions of the Fault Interceptor Module in response to missing-page and missing-segment faults.

Page/Segment Fault Handler Actions

When a Multics processor generates a missing-page or missing-segment fault (directed fault 0), control automatically enters the page/segment fault handler, which executes on behalf of the process that is running at the instant the fault occurs. The following steps are taken by the page/segment fault handler:

1. The processor state is stored in the current interrupt frame of the Process Concealed Stack (Section BJ.5.06). 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 cannot be taken until after the linkage base registers are set. Eventually, if a hardware ring register is added to the GE-645, that register should be stored with the processor control unit, making this step unnecessary.)

2. The values for the linkage pointer and linkage base registers are determined and loaded into the lp-lb address base register pair. (Since the Fault Interceptor Module is not called, it must establish its own linkage values.)

3. A new interrupt frame is allocated in preparation for the next fault or internal interrupt. A processor may detect a missing page or segment and generate the corresponding fault while handling an earlier fault or internal interrupt. Therefore, 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 Process Concealed Stack pointer, sp, is obtained. If the Process Concealed Stack is not "empty" ("empty" implies not in use), the sp-sb base register pair (stored in Step 1) points to the base location of the stack frame (in the Process Concealed Stack) 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 the _next_ sp to obtain the base location of the new interrupt frame. (Currently, k is equal to 32.) If the Process Concealed Stack is "empty" (i.e. not in use) at the instant a page or segment fault occurs, then the new interrupt frame is allocated immediately following the current interrupt frame. 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 at the base of the Process Concealed Stack.

    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. Control is switched to the hard core ring. (How this is done is described in Section BK.3.03.)

5. The safe-stored processor control unit is examined to distinguish missing-page faults from missing-segment faults. For missing-segment faults only, the machine conditions are examined to determine whether reference to the fault_stack caused the missing-segment fault to occur. If not, the interrupt frame containing the safe-stored processor state is copied from the Process Concealed Stack into the fault_stack. Since the Process Concealed Stack has been "pushed down" one level, a missing-page fault is acceptable during the attempt to copy the interrupt frame. If reference to the fault_stack did cause the missing-segment fault to occur, then the Process Concealed Stack is used to handle the fault.

6. A new interim stack to be used in handling the missing-page/segment fault is created immediately following the new interrupt frame. The sp-sb address base register pair is set to point to the base location of the new interim stack. (Note that for missing-page faults and for missing-segment faults caused by reference to the fault_stack, the stack in use is the Process Concealed Stack; for other missing-segment faults, the stack in use is the paged fault_stack.)

7. 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 Fault Interceptor Module for handling page and segment faults is used in determining the value to be stored at sp|18. (Since the Fault Interceptor Module is not CALLed, it must determine its own temporary storage requirements in order to initialize the first frame of the interim stack.)

8. The appropriate argument list containing a pointer to the safe-stored processor state (scuptr), a pointer to the value of the descriptor base register at the instant of the fault (dbr), and a pointer to the ring number at the instant of the fault (ringnumber), is fabricated and stored in the temporary storage for the Fault Interceptor Module in the interim stack.

9. A standard CALL is issued to the appropriate Basic File System module to handle the fault. For missing-page faults, Page Control is called as follows:

```
call pagefault(scuptr,dbr,status)
```

For missing-segment faults, Segment Control is called as follows:

```
call segfault(scuptr,dbr,ringnumber,status)
```

Pagefault and segfault may CALL other modules; pagefault may not, however, allow control to pass outside the hard core ring of the running process. The Fault Interceptor Module will experience a RETURN from pagefault and segfault, although control may not return until much later (i.e. until after Page Control or Segment Control has received a Wake-up signal). When control returns to the Fault Interceptor Module, the sequence of steps presented here continues.

10. For missing-segment faults only, the interrupt frame containing the safe-stored processor state is copied from the paged hard core ring stack into the Process Concealed Stack. (This step is taken only if missing-segment is not fault_stack.)

11. Control is switched to the ring in which the fault occurred.

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

13. The scu, sreg, and stb pointers stored at the base of the Process Concealed Stack are restored from locations 31-32 of the current interrupt frame to point to the base location of the previous interrupt frame.

14. The processor state is restored to its state at the instant the fault occurred. This is done as follows:

    a. The arithmetic registers are restored (load registers instruction) using the sreg pointer.

    b. The address base registers are restored (load bases instruction) using the stb pointer.

    c. The processor control unit is restored (restore control unit instruction) using the scu pointer. When the processor control unit is restored, control returns automatically to the point at which the fault occurred.

Error Conditions

In the above list of actions taken to handle missing-page and missing-segment faults, error conditions have been ignored completely in an attempt to remain legible. The occurrence of what appears to be a normal missing-page or missing-segment fault may, in fact, indicate one of the following conditions:

1. Boundary violation of the descriptor segment (i.e. segment number is out of range).

2. Boundary violation on the length of the segment.

In addition, certain error conditions may arise in the course of retrieving segments and pages. For example, Page Control may discover a parity error which it cannot correct in its attempts to retrieve a page.

The Fault Interceptor Module examines the return argument *status* after calls to segfault and pagefault. If an error status is detected, the FIM proceeds or if a process fault had occurred, as described in BK.3.03. In particular, an appropriate condition is signalled.