Published: 03/24/67

Identification

Subroutine Overlay
R. L. Rappaport

Purpose

Subroutine Overlay is a master mode procedure which restores a loading process' descriptor segment to the state it was in before the process was unloaded.

Introduction

Various hardware and software considerations require that certain hardcore supervisor modules be known by the same segment number in each process. Even though distinct processes may have distinct versions of one of these modules, the copy in each process is known by the same segment number. For example, machine conditions at the time of a fault are saved in the Process Data Segment of the process which experienced the fault. These conditions are stored using ITS pointers located at known absolute locations in core storage. The ITS pointers are initialized at system initialization time (see Section BL.0). Each process has its own distinct Process Data Segment and since these ITS pointers must point to the Process Data Segment of a process whenever it is executing, this segment must be known by the same number in each process. The segments which must be known by the same number in each process are called the special segments throughout this document.

Discussion

In order for a process to function in the system the process at all times must have a consistent set of these special segments each known by the correct segment number. When a process is unloaded its descriptor segment is destroyed. In initiating the reloading of a process in swap_dbr (see Section BJ.5.01), the calling process in swap_dbr creates a standard descriptor segment for the unloaded target process. This standard descriptor segment is created, in the main, by copying from the template descriptor segment (see Section BJ.5.06). This descriptor segment contains segment descriptor words for a standard consistent set of special segments. In particular, in the location reserved for the Process Data Segment, this descriptor segment

contains a segment descriptor word for an Interim Process Data Segment. This Interim segment is explicitly created for the unloaded target process at the same time the standard descriptor segment was created. Once the unloaded target process gains control of the processor it goes about retrieving its lost special segments. It does this in two steps.

The first step in retrieving the lost special segments consists of obtaining valid segment descriptor words for the individual segments. That is, the segments are activated, if not already active, and the segment descriptor words are placed in the newly created standard descriptor segment. However, these segment descriptor words are not placed in the locations in which they will ultimately reside. For example, suppose the Process Data Segment is known in each pr cess as segment number K. When reactivated by the loading process, it will be reactivated as segment number $J \neq K$. This is done so that the process never has an inconsistent set of segment descriptor words in the reserved locations in its descriptor segment. Some of the special segments reactivated in this way are segments that must be wired down whenever the process uses them. All such special segments are wired down at the time of their reactivation. An example of a segment which must be wired down is the Process Data Segment of the process. This reactivation and partial wiring of special segments is accomplished in the Process Bootstrap Module (see Section BJ.5.03) which is called from swap_dbr immediately after the unloaded target process gains control.

Upon return from the Process Bootstrap Module, all the special segments of a process have segment descriptor words in the process' descriptor segment. However, they are located in the wrong locations. It is therefore time to overlay these segment descriptor words into the appropriate locations. This overlaying is done at one time while the processor is inhibited so that, from the outside, the process always appears to have a consistent set of special segments. This overlaying is accomplished in subroutine overlay which can now be specified completely.

Specification of Overlay

Overlay is called from swap_dbr after swap_dbr receives a return from the Process Bootstrap Module. Overlay is called on the Process Concealed Stack (which is contained in the Process Data Segment). However, at this time,

the Process Data Segment is being referenced by the segment number by which the segment was retrieved rather than by the one that is normally used by loaded processes. This is because the process entering Overlay still has a segment descriptor word for the Interim Process Data Segment in this location. The calling sequence for overlay is simply:

```
call overlay;
```

In the process definitions segment (see Section BJ.1.00) of the process there is a table which relates the segment numbers by which special segments are retrieved and the segment numbers by which they are known. Overlay begins by making a copy of this table in the Process Concealed Stack. This is done to place this data into wired-down storage. The actual segment descriptor words are overlayed while the processor is inhibited and no page faults can be tolerated while this is going on. Once this data is in wired-down storage overlay inhibits the processor and then ripples through the table overlaying the indicated segment descriptor words. Having changed the segment descriptor words, the associative memory of the processor must be cleared in order to destroy any invalid associations. Finally, while the processor is still inhibited, base register sb must be redirected from the segment number by which the Process Data Segment was retrieved to the segment number by which this segment is known. This step is taken so that if a fault occurs, the Fault Interceptor Module (see BK.3.01) can determine whether the Process Concealed Stack is empty or not by testing the value stored in register sb. At this point the processor can be uninhibited. Before returning, overlay modifies the value of sb stored on the call to overlay. This is to enable the return to work correctly.

Overlay is described step by step below and illustrated in figure 1.

1. Copy contents of special segment table into wired-down data base.

2. (Inhibit on) Load the A-register with the next segment descriptor word that is to be overlayed.

3. (Inhibit on) Store the A-register into the appropriate location in the descriptor segment.

4. (Inhibit on) If all the segment descriptor words have not been overwritten go to step 2.

5. (Inhibit on) Clear the associative memory.

6. (Inhibit on) Reload base register sb.

7. Modify the value of sb stored in the stack.

8. Return.

Wrapup

Throughout this document there has been considerable discussion of the set of special segments of a process of which the Process Data Segment is an example. In most cases, this is the only special segment that a process has. In an attempt to remain general the discussion was carried through assuming several special segments.

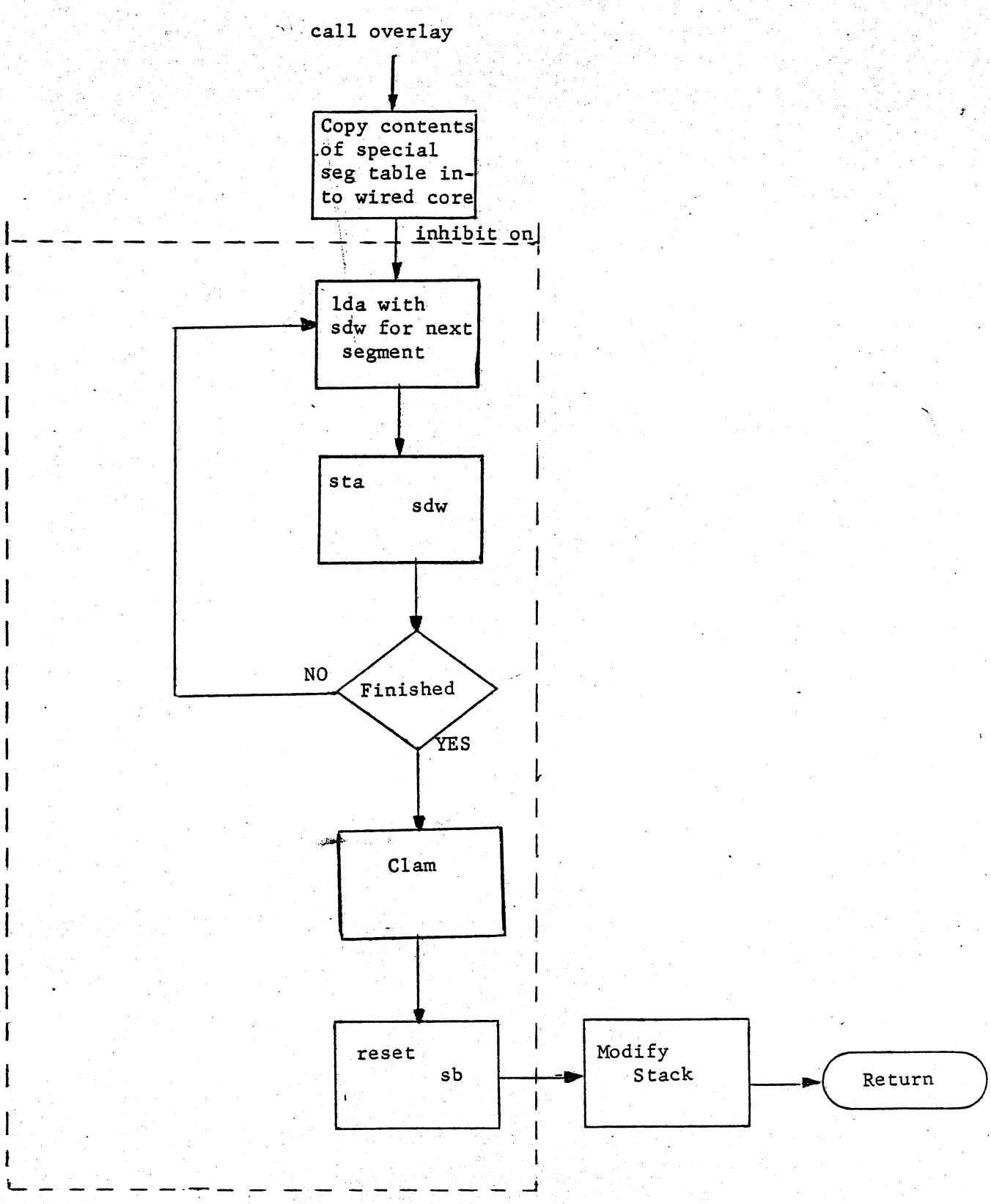

Figure 1 Flow Diagram of Overlay