Segmented Program Execution [7]

Segmented programs are called into execution in the same manner as are nonsegmented programs. Additional control statement parameters can be provided.

$SEGRES

7.1

On execution, the operating system transfers control to the $SEGRES routine. $SEGRES is a system routine that resides in the loader and is loaded with the object module. It reads segments into memory for execution and writes segments to mass storage to save current segment states.

$SEGRES accepts control from the operating system when execution begins, and it is responsible for some initialization functions. $SEGRES first determines whether the executable binary code can be read from the executable file every time a segment needs to be loaded. If you specified SAVE=ON, all segments are copied to a scratch file in which all reading and saving are done. You can control the scratch file location by using the TMPDIR environment variable. Control transfers to the main entry point in your program after the copy operation to the scratch file.

$SEGCALL intercepts subroutine calls that might require the loading of called segments into memory. $SEGCALL also saves memory-resident segments if SAVE=ON for those segments; this ensures that they are not overwritten.

At execution time, common block /$SEGRES/ conveys information collected by the loader during the load process to $SEGRES. The information that is passed includes segment sizes and addresses, and addresses of intercepted calls between segments.

Subroutine call overhead
7.2

In a segmented load, there are five types of calls to subroutines. Table 5 describes the overhead needed for each type of subroutine call.

Table 5. Subroutine call overhead

Segment containing called routine	Action taken by `$SEGRES`
Same segment as calling routine	The call is not intercepted.
Predecessor segment of calling routine	The call is not intercepted.
Successor segment in memory	After determining that the segment is resident, control transfers to the called routine.
Successor segment not in memory	One or more successor segments are read into memory; then control is transferred to the called routine.
Successor segment not in memory and `SAVE=ON`	One or more currently-resident segments are written to a scratch file so that they are not overwritten when the called segment is read into memory. After the needed segments are read into memory, control transfers to the called routine.