TO: MSPM Distribution
FROM: M. A. Padlipsky
DATE: 11/09/67
SUBJ: Sections BY.13.00 - BY.13.03

Numerous minor changes have been made to the existing BY.13.02 - BY.13.03 sections, necessitating their reissue. Further, an Overview to the area (linkage maintenance) has been added, as BY.13.00. BY.13.01 is also being issued at this time.

Published: 11/09/67

Identification

Overview of linkage maintenance
D. L. Boyd

Purpose

This section provides an overview of the linkage maintenance routines. These routines make it possible for the user to force a link to be set (completed), generate a pointer to a defined segment and symbol, and add links and definitions to linkage sections.

Introduction

The link forcing procedure is described in BY.13.01. This routine forces a designated link to be completed in a linkage section.

The generate_ptr procedure, BY.13.02, generates a pointer to a given segment name and external symbol if they exist or to the first location of a segment. It also returns to the calling procedure the class (see BD.7.01) of the symbol.

The procedure link_change generates linkage section information during the execution of a process. The information generated may be a link, a definition, a link with a trap, or a definition with a trap. This procedure is described in BY.13.03.

Discussion

In order to use the linkage maintenance routines, it is necessary to have some understanding of the composition of a linkage section. Linkage sections are described in detail in BD.7.01.

Briefly, the linkage section is made up of one or more linkage blocks that are threaded together. Each block contains information (in words three and four) pointing to the start of the next block. If there is no next block, that information is zero. Word seven of each block contains the current length of the block. The linkage maintenance routines assume that the storage immediately following the last linkage block is unused. This area will be used by these routines for additions to the linkage section and is called the linkage section's free storage.

Whenever a definition is added (i.e., link_change$make_definition and link_change$make_trap_definition are called), a new block is created at the beginning of free storage, if there is only one block, or the last block is extended if the definition pointer in the last block points into the block itself. If a new block is created it is threaded to the immediately preceeding block by changing words three and four in that block so that they point to the start of the new block. The new block contains in word seven the length of the new block and word eight in the initial block is changed to give a new total length for the linkage segment. The remaining contents of a block are described in BD.7.01.

Whenever new linkage information is added (i.e., link_change$make_link and link_change$make_trap_link are called), it is inserted at the start of the linkage section's free storage. The last block is extended to include this information by adding the length in words, to word seven of the last block and word eight of the initial block which increases the total length. The same criteria is used for creating a new block when adding a link as is used when adding a definition.

Calls to add definitions and links to the same linkage section may be done randomly. The definitions are all threaded together bypassing the links. In either case, link_change cannot work with an empty linkage section. It requires at least the eight word header described in BD.7.01.

It is possible to generate a pointer to any definitions that have been added as well as the original ones. The generate_ptr routine uses the threaded blocks to search through each linkage block looking for the appropriate definition. Pointers may be generated to the actual location in a segment or a segment's symbol segment and to the link pair in the linkage segment if the definition is an entry point.

The link forcing routine will set any link that has not been previously set. It is actually an entry into the Multics linker (BD.7.04).