Published: 08/26/68
(Supersedes; BX.14.02, 11/08/67;
BX.14.02, 06/19/67)

Identification

The Post_binder
R. H. Thomas

Purpose

The binder (BX.14.01) combines two segments into one segment. Prior to binding a reference to one segment from the other must be made through a linkage segment. Although such references after binding are no longer inter-segment but rather are intra-segment, they continue to be made indirectly through the bound linkage segment. This section describes two additions to the basic binder. The first addition enables such intra-segment references to be made directly and the second addition deletes from the bound segment and bound linkage segment information that is no longer necessary after the intra-segment references are made direct.

Usage

Intra-segment references in the output of the basic binder, <bound_segment>, that are made indirectly through the linkage segment, <bound_segment. link>, are changed into direct references by use of the command:

```
post_bind bound_segment name1 name2 ... namen - delete -
```

Name1, name2, ..., namen are the names of the component segments that make up bound_segment. If the user fails to specify all component names only intra-segment references to the component segments specified are made direct.

If the optional argument "delete" is included the post_binder will delete from <bound_segment> and <bound_segment.link> the information no longer necessary to make intra-segment references.

Words that are candidates for deletion are link pairs, expression words, type pairs, segment names, and external symbol names (see BD.7.01). In addition duplicate definitions of the symbols rel_text, rel_link, rel_symbol and symbol_table are deleted (see BX.14.01).

No attempt is made to delete information from <bound_segment.symbol>. Note, however, that the relocation code bits found in <bound_segment.symbol> must be altered because intra-segment references are no longer link pointer 15 relocatable (see BD.2.01) and because the code for deleted information must also be deleted.

Implementation

Post binding is accomplished in two steps. In the first step intra-segment references made indirectly through the linkage segment are found and made direct and information required by the delete option is generated. In the second step, which is performed only if the delete option is requested, unnecessary linking information is deleted.

Step 1: The binding information for the bound text segment is scanned for link pointer 15 relocatable half words (code = "10100"b; see BD.2.01). Whenever such a reference to the linkage section is found, the corresponding link is examined to see if it is an intra-segment reference. If the link is intra-segment, it is evaluated, the address field of the word in the text segment is changed to the location referenced by the link, bit 29 of the word is set to 0, the modifier bits (30-35) of the word are set to 0, and the relocation code for the left half of the word is changed from 10100 to 10000.

In addition, if the delete option is requested two tables are created and maintained: the text deletion table (TDT) and the link deletion table (LDT). The TDT contains an entry for each word in bound_segment; At the completion of step 1 each entry contains the following information:

either 1. The word referred to in bound_segment will be deleted by the delete option

or 2. (a) The word referred to will not be deleted.
(b) The number of words that will be deleted by the delete option between the origin of the text segment and the word referred to.

The LDT contains the same information for the bound linkage section.

Step 2: Prior to deletion every halfword in the text, linkage and symbol segments is relocated. Relocation is accomplished in the following manner:

(a) Each halfword in the text segment with the exception of definition pointer relocatable (code = "10101"b) halfwords is relocated.

(b) The linkage section with the exception of definition-relocatable halfwords is relocated one block at a time.

(c) Each halfword in the symbol segment is relocated.

(d) In order to relocate definition pointer relocatable halfwords the linkage section is scanned a block at a time. The definition pointer is followed to the definitions section and all definition pointer halfwords are relocated. Next, all definition pointer relocatable halfwords in the current linkage block are relocated.

The chart below illustrates how the TDT and LDT table are used to adjust each halfword. To read this chart assume that the halfword to be adjusted has value K and that TDT(I) (LDT(I)) represents the number of words to be deleted between the origin of the text (link) segment and the Ith word in the segment.

Relocation Code		Action
0	Absolute	no relocation
10000	Text	K = K - TDT(K)
10001	Neg Text	K = K + TDT(K)
10010	Link Pointer 18	K = K - LDT(K)
10011	Neg Link Pointer 18	K = K + LDT(K)
10100	Link Pointer 15	The left-most three bits are unchanged; LDT(K) is subtracted from the remaining 15 bits
10101	Definition Pointer	K = K - deletion_table(K + def_ptr)+ deletion_table(def_ptr)
10110	Symbol	no relocation
10111	Neg Symbol	no relocation
11000	Link Block	K = K - LDT(K + current_blk_ptr) + LDT(current_blk_ptr)
11001	Neg Link Block	K = K + LDT(K + current_blk_ptr) - LDT(current_blk_ptr)
11010	Self_Relative	K = K - deletion_table(K + I) + deletion_table(I) (Halfword relocated is part of Ith word of segment)
11011	Not Used	should never occur
11100	Not Used	should never occur
11101	Not Used	should never occur
11110	Not Used	should never occur
11111	Escape	should never occur

* deletion_table refers to the proper deletion table. For definitions pointer relocatable items deletion_table is TDT if the definitions are in the text segment and LDT if they are in the linkage section.

After relocation is completed the bound segment and bound linkage section are rewritten omitting the words marked for deletion in the TD and LD tables.

The relocation information for the undeleted words is packed into the bound symbol segment.