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

Identification

The Post_binder
R. H. Thomas

Purpose

The binder 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_name>, that are made indirectly through the linkage segment, <bound_segment_name. link>, are changed into direct references by use of the command:

post_bind bound_segment_name -delete-

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

Restrictions

1) No provision is made to handle linkage segments with more than one linkage block. (Temporary)

2) Only type 1 and type 2 binding can be handled. (Temporary)

3) No attempt is made to delete information from <bound_segment_name. symbol>. (Note, however, that the relocation code bits found in <bound_segment_name. symbol> must be altered.)

Implementation

In the following discussion, it is assumed that the segments <a> and <b> have been bound to form the segment <ab>.

Indirect intra-segment references are changed into direct references in two steps. In the first step the intra-segment references made through the linkage segment are found and part of the information required by the delete option is generated. In the second step, the references found in step 1 are made direct and the remaining information needed by the delete option is generated.

Step 1: Step 1 creates three tables: the instruction table (IT), the text deletion table (TD), and the link deletion table (LD). The IT contains pointers to the intra-segment references in <ab> that are changed in step 2; in addition, it contains information used in step 2 to re-evaluate the address portions of the intra-segment references. The TD table and the LD table are used by the delete option. The TD table contains an entry for each word in <ab>; when step 1 is finished each entry contains the relocation code bits for the corresponding word in <ab>. The entries in the LD table correspond to the words in <ab.link> and are identical in format to the entries in the TD table.

Step 2: The pointers in the IT are used to find the references to be altered. The address field (bits 0-17) is re-evaluated and bit 29 is set to 0. The relocation code for the left half word of the reference is changed from 1100 to 1000. (see BD.2.01). As the address field is being re-evaluated, entries in the TD and LD tables are updated. At the completion of step 2 each entry contains the following information:

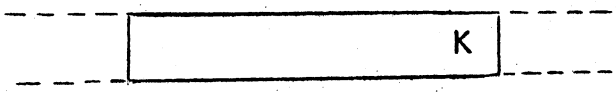
either 1. The word referred to in <ab> or <ab.link> will be deleted by the delete option

or 2.
(a) The word referred to will not be deleted.
(b) The relocation code for the word.
(c) The number of words above the word referred to that will be deleted by the delete option.

Words that are candidates for deletion are (in <ab>) expression words, type pairs, segment names, external symbol names appearing in link definitions and (in <ab. link>) links. (see BD.7.01).

Deletion of information no longer necessary after the intra-segment references are changed is accomplished in three steps.

Step 1: The relocation code bits in the TD and LD tables are used to examine and adjust when necessary each half word in <ab> and <ab. link>. The chart below illustrates how the TD and LD tables are used to adjust each half word. (See BD.2.01):



Half word to be adjusted; part of Ith word in segment

The half word to be adjusted is assumed to have value K and to be part of the Ith word in the segment. The relocation code is found in TD(I) or LD(I). LENA is the length of the "a-part" of <ab> and is calculated by the binder (see BX.14.01).

Relocation Code		Action
0	(Absolute)	if segment is <ab. link> and half word is left half of a tra instruction: K←K - LD (K + I) + LD (K); otherwise leave half word unchanged.
1000	(Text)	if K ≤ LENA leave half word unchanged; if K > LENA: K←K - TD(K)
1001	(Negative Text)	if K ≥ -LENA leave half word unchanged; if K < -LENA: K←K+TD(-K)
1010	(Link)	K←K- LD(K)
1011	(Negative Link)	K←K+ LD(K)
1100	(Link Pointer)	The left-most three bits are left unchanged and LD(K) is subtracted from the remaining 15 bits.

Relocation Code		Action
1101	(Definition Pointer)	K←K+ TD(K+def_ptr)
1110	(Symbol)	leave half word unchanged
1111	(Escape)	should never occur

Step 2: <ab> and <ab.link> are rewritten omitting the words that are marked for deletion in the TD and LD tables. The relocation information for the undeleted words is entered into <ab.symbol> from the TD and LD tables.

Step 3: The values of the symbols appearing in the external definition section are adjusted so that the values are correct after the unnecessary information is deleted.