MEMO TO: F.J. Corbató
R.C. Daley

FROM: R.M. Graham

SUBJ: Revision of EPLBSA

DATE: January 22, 1969

Following is a list of suggested revisions to EPLBSA which will improve its performance. The gain in each of these is not uniform. My notebook on EPLBSA contains an analysis of the time saved for each of these revisions.

1. The only place in the assembler that storage is ever returned to the free storage pool is in the output listing section. I do not totally understand what is going on here, but I am convinced that some other way can be found to handle the output listing. If the output listing does not return any storage to free storage, then the free storage management algorithm can be greatly simplified. This will enable the initialization of free storage, which currently takes half a second, to be eliminated altogether. Secondly, it allows the algorithm for finding a block of free storage to be greatly simplified. At present, all free storage is threaded together on a list so that returned storage may be reused. If no storage is ever returned to the free pool, then the available storage is just the remaining words in a single segment. Searching for a new block is then merely taking the next N words in sequence and updating a counter which counts the number of used words in the segment, rather than searching through a threaded list looking for a block of N contiguous words.

2. If references to free storage were changed from absolute addresses to an index in an array, then all of the machine language subroutines CLH, CRH, etc., could be replaced. Once this change is made all references to free storage can be made using subscripted variable rather than calling a subroutine.

3. In pass 1 only a table of the pseudo-ops needs to be searched and not the entire operation code table. Searching a table of pseudo-ops would be considerably faster than searching the entire operation code table.

4. The length of POSTP1 could be reduced nearly 400 words by combining common code in three different subroutines.

5. At present, two table look-ups are made for the first symbol in the variable field even for instructions such as shift instructions where no symbol appears. This can obviously be eliminated.

6. The codes used to identify character classes can be recoded as a numerical index which can be used for indexed transfers in the routines that get characters from the input stream. This would allow new line, space, horizontal tab, and semicolon to be identified in a single test, rather than four separate tests. In addition, the subroutines NEXT and NEXTNB could probably both be combined with the subroutine GETID thereby increasing the performance.

7. The routine EXPEVL could be modified to get rid of subroutine calls to PREC by directly using the precedence of the operators in EPLBSA expressions.

8. The subroutines PUTXT, PULNK, and PUSMB could be eliminated and words in the text, link, and symbol segments would be stored directly in the segments. However, the problem of what to do with relocation bits is still unsolved.

9. All the error flags could be put in one 36 bit word instead of stored in 36 separate words. This would eliminate the packing and unpacking of these error flags. In addition, error flags are now carried over from pass 1 to pass 2. I am sure the need for carrying the error flags over can be eliminated.

10. Eliminate all unnecessary character conversions, e.g., BCD to ASCII and ASCII to BCD.

11. Duplicates of routines such as NEXT and GETID, which are used in both pass 1 and pass 2 could be used. This would reduce the working set and each of the routines might be specialized to the pass in which it occurred with some increase in efficiency.

12. At least the loading order of the procedures should be checked to be certain that the working set is reduced to as small a size as is possible.