MEMO TO: F.J. Corbató
R.C. Daley

FROM: R.M. Graham RMG

DATE: January 22, 1969

SUBJ: Transforming EPLBSA into a Multics Maintainable Program

Outlined herein is a plan for transforming the assembler (EPLBSA), currently a collection of FORTRAN and GMAP procedures, into a collection of EPL and EPLBSA procedures which can be maintained within the Multics system. The reason this is a problem is that the FORTRAN and GMAP procedures are bound (via gecos_seg) into a single segment and they know that both common storage and all other procedures are in the same segment. The following plan permits gradual replacement, a procedure or two at a time, of the individual procedures in the assmebler.

1. The general idea is to treat all common storage as based storage.

2. A small FORTRAN procedure is written which is called before anything else is executed inside the assembler. This procedure will call an EPLBSA procedure once for each common variable. The call has one argument which is the location of the common variable. The EPLBSA procedure builds up a list of pointers to the common variables in the assembler's common area.

3. This list of pointers is stored in a single segment (say with name C). Each pointer in the segment C is given an external symbolic name identical to the name of the common variable to which it points. Each routine converted to EPL or EPLBSA then refers to common by using these pointers and based storage. These based storage references can be turned into direct external segment references after the entire assembler is converted. When a module is converted a reference to a common variable, say X, will be written as C$X->Y, where Y is a based variable with the appropriate attributes for the common variable named X in the assembler. When the assembler is totally converted the editor can be used to sweep through and change all occurences of C$X->Y to C$X. The declaration of C$X is then changed from pointer to the attributes of Y and the declaration for Y is deleted.

4. The remaining problem seems to center around the fact that call-outs from a gecos_seg produced procedure segment may have only one argument. Most of the routines in the assembler have more than one argument; hence, some minor revisions will have to be made in the FORTRAN and GMAP routines to make calls to any converted procedure have one argument. This single argument should be a location in common where the real arguments are found in sequence. The converted procedure which is being called should have a short section of code in the beginning which generates pointers to these arguments. The arguments are

referred to in the body of the procedure using these pointers and based storage. Later, when the assembler is all converted, these based references to the procedure's arguments can be easily converted to dummy variables by use of the editor.