TO: C. T. Clingen, F. J. Corbató, R. Freiburghouse, J. W. Gintell, N. I. Morris, J. H. Saltzer, M. D. Schroeder, S. H. Webber, V. L. Voydock

FROM: R. J. Feiertag

DATE: June 22, 1971

SUBJECT: Problems of pointer comparisons and the null pointer on the follow-on processor

In the current PL/I implementation a PL/I pointer is considered to be equivalent to a GE-645 ITS pointer. Strictly speaking this is not true on the current hardware and will be even less "true" on the follow-on processor.

A PL/I pointer describes the location of some piece of data. In order to do this in the Multics virtual memory one need only specify a segment number, a word offset, and a bit offset. The GE-645 ITS pointer contains the ITS modifier and a tag field in addition to these three items. Since all PL/I pointers have the same modifier and tag fields, comparison operations on PL/I pointers can be accomplished by comparing the ITS pointers used to represent them.

In the follow-on processor an ITS pointer has an additional field, the ring number. This field will not be the same for all PL/I pointers and, therefore, PL/I pointer equality can no longer be determined by an ITS pointer comparison. For this reason PL/I pointer comparisons will have to be made by comparing ITS pointers excluding the ring field.

Currently, a PL/I null pointer is represented by a segment number of 777777 octal, a word offset of 1, and a bit offset of zero. On the follow-on processor the store pointer in packed format instruction will fault trying to store this pointer because the segment number is too large. Not being able to store null pointers in packed format would render the packed format essentially useless. Ruling out the possibility of changing

the hardware to handle this case, the only remaining solution is to change the representation of the null pointer.

Clearly, in order to avoid later recurrence of this problem, the segment number chosen to represent the null pointer should be less than the smallest maximum segment number Multics will ever have. The ideal choice is, therefore, segment number 0. A possible representation for the null pointer would then be segment number 0, word offset 1, bit offset 0. Whatever the new representation there is a considerable upward compatability problem involved. Assuming the two representations would coexist for some period of time, i.e. no flag day, every pointer comparison would have to make a special check for the two representations and do some extra work if both operands were null pointers in different forms. For the new representation suggested above, the comparison operation would have to check for segment numbers of 777777 and 0 and if the both forms were found, one in each operand, then treat the segment numbers as identical. Note that in the new representation I have chosen the word offset and bit offset to be identical to the current representation making that part of the comparison easy.

For convenience of adding and removing this compatability step the pointer comparison operation should probably be placed in the PL/I operator segment. It is clear that the transition period will be long, therefore, the necessary modifications should be made as soon as possible.

Please send me comments on any of the ideas I have discussed here.