Identification

The substr built-in function and pseudo-variable.
substr$_sscs_ , substr$_ssbs_ .
D. B. Wagner and M. D. McIlroy

Purpose

See the PL/I manual (IBM form C28-6571-3, pp. 103 and 153) for a discussion of the substr function. In the implementation of substr the PL/I compiler may use the procedure described here to make up a dummy dope vector for a substring of a character-or bit-string. Substr cannot be used directly in a PL/I program because its calling sequence is (and must be) peculiar.

Usage

The two possible calls are:

call substr$_ssbs_(i,j,b1,spec);
call substr$_sscs_(i,j,c1,spec);

b1 is a bit-string, varying or non-varying. c1 is a character-string, varying or non-varying. b1 or cl corresponds to $ in the PL/I manual's description of the substr function. i and j correspond to the i and j in that description. They are declared,
dcl (i,j) fixed bin (24);

Spec is a dummy specifier: the argument pointer points to:

"data pointer": an its pair to be filled in by substr_.
"dope pointer"

dope vector: entire contents to be filled in by substr_.

See BP.2.01 for a discussion of specifiers and dope. Substr_ stores values into "data pointer" and the dope vector so that spec becomes a specifier for the appropriate substring of the given string.
The statement
   \( a = \text{substr}(b, i, j); \)
might be implemented as the following calls:
   call substr$_{\text{sscs}}$(i, j, b, spec);
   call stgop$_{\text{cscs}}$(spec, a);
(See BP.6.01 for a description of stgop$_{\text{cscs}}$.)

The statement
   \( \text{substr}(b, i, j) = a; \)
might be implemented as the following calls:
   call substr$_{\text{sscs}}$(i, j, b, spec);
   call stgop(a, spec);

The above implementation, however, is not satisfactory
for the following statement, if \( a \) is a non-varying string.
   \( \text{substr}(a, i, j) = a; \)

Here the danger is that the move from \( a \) to the substring
may "clobber" parts of \( a \). See BP.6.01 for a deeper discussion
of this problem.