Identification

The \texttt{substr} built-in function and pseudo-variable.
\texttt{substr\_sscs\_}, \texttt{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 \texttt{substr} function. In the implementation of \texttt{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. \texttt{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:

\begin{verbatim}
call substr\_ssbs\_(i,j,bl,spec);
call substr\_sscs\_(i,j,cl,spec);
\end{verbatim}

\texttt{Bl} is a bit-string, varying or non-varying. \texttt{Cl} is a character-string, varying or non-varying. \texttt{Bl} or \texttt{cl} corresponds to \texttt{s} in the PL/I manual's description of the \texttt{substr} function. \texttt{i} and \texttt{j} correspond to the \texttt{i} and \texttt{j} in that description. They are declared,

\begin{verbatim}
dcl (i,j) fixed bin (24);
\end{verbatim}

\texttt{Spec} is a dummy specifier: the argument pointer points to:

\begin{verbatim}
\begin{tabular}{|c|}
\hline
\multicolumn{1}{|c|}{"data pointer": an \texttt{its} pair to be filled in by \texttt{substr\_}.} \\
\hline
\multicolumn{1}{|c|}{"dope pointer"} \\
\hline
\end{tabular}
\end{verbatim}

\begin{verbatim}
\begin{tabular}{|c|c|}
\hline
offset & 240(8) length \\
\hline
\end{tabular}
\end{verbatim}

dope vector: entire contents to be filled in by \texttt{substr\_}.

See BP.2.01 for a discussion of specifiers and dope. \texttt{Substr\_} stores values into "data pointer" and the dope vector so that \texttt{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:
   \[
   \text{call substr$_{sscs_}$(i, j, b, \text{spec});}
   \]
   \[
   \text{call stgop$_{cscs_}$(spec, a)};
   \]
(See BP.6.01 for a description of stgop$_{cscs_}$.)

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