Identification

The EPL run-time routine, movstr_
movstr_$movb_
movstr_$movc_
movstr_$not_
movstr_$and_
movstr_$or_
movstr_$xor_
movstr_$exclor_
movstr_$notand_
movstr_$notor_
movstr_$nnot_

Ruth A. Weiss

Purpose

EPL uses movstr_ for all string copying operations that are not compiled in-line. Movstr_ is not called directly by EPL but anytime a string is moved at least one of the entries to movstr_ is invoked. EPL compiles a call to stgop (See BN.7.09 for stgop_) which in turn may call movstr_. The many entries to movstr_ were written for the PL/I function routine, bool_ (See BN.7.04 for bool_ ) but they may be called directly in any EPL program. The EPL run-time routines catstr_ and andstr_ also call movstr_.

Usage

Movstr_ accepts either varying or non-varying strings as arguments. If the second argument is a non-varying string and has a longer length than the first, the first string is extended on the right with a padding byte to the lengths of the second string. Padding = '0' b for all entries except movc_, notor_ and nnot_. Movc_ has ASCII blank and notor_ and nnot_ have '1' b for padding.

The possible calls are listed below. If a particular call to stgop_ always invokes a particular call to movstr_, the call to stgop_ is listed first. B1 and b2 are bit strings and c1 and c2 are character strings.

```plaintext
call stgop_\_cscs_\_(c1,c2);
call movstr_\_movc_\_(c1,c2);
c2=c1;
call stgop_\_bsbs_\_(b1,b2);
call movstr_\_movb_\_(b1,b2);
b2=b1;
```
call stgop$_ntbs_$(b1,b2);
call movstr$_not_$(b1,b2);

\[ b2 = \overline{b1}; \]

call movstr$_and_$(b1,b2);
\[ b2 = b2 \& b1; \]

call movstr$_or_$(b1,b2);
\[ b2 = b2 | b1; \]

call movstr$_exclor_$(b1,b2);
\[ b2 = b2 \text{ exclusive or } b1; \]

call movstr$_notand_$(b1,b2);
\[ b2 = b2 \& \overline{b1}; \]

call movstr$_notor_$(b1,b2);
\[ b2 = b2 | \overline{b1}; \]

call movstr$_nnot_$(b1,b2);
\[ b2 = \overline{b1}; \]

**Error**

If either argument is not a string, will stop on oct 0.