Published: 09/28/66

Identification

"Entry variables" in PL/I.
Fake_entry$create, fake_entry$call.
D. B. Wagner

Purpose

A problem with the PL/I language is that there is no such thing as an "entry variable," that is, for example, there is no equivalent to the MAD sequence.

```
FUNCTION NAME  A

A = SIN.

...

B = A. (C)
```

This causes trouble in a number of places: in particular in the Request Dispatcher (described in BY.6.01). The most reasonable design for the Request Dispatcher involves a calling argument which is an array of entries.

The library routines fake_entry$create and fake_entry$call provide a way of getting around this difficulty.

Usage

The calling forms are:

```
a = fake_entry$create (q);

call fake_entry$call (a,...);
```

The ellipsis represents an arbitrary sequence of arguments. A and q are declared as follows:

```
dcl q entry,

    a bit (216);  /* 216=6*36 */
```

fake_entry$create bears a family resemblance to unspec: it takes an entry and returns a bit-string representation of it. Fake_entry$call is used to call the entry represented by such a bit-string, giving any arguments the user desires.

Implementation

The bit-string a above is 6 words long, and these words are used as follows. The first and second contain an ITS

pair pointing to the entry, the third and fourth contain an ITS pair representing a stack level (always present, but not used unless q is an internal procedure), and the fifth and sixth contain validating information (not yet specified). This is precisely the same form as a PL/I label variable.

It should be noted that since a fake entry contains a stack pointer, one should be chary of putting it into static or controlled storage.