Published: 08/29/68

Identification

Lexical Analyzer
James D. Mills

(Note that the following are Abstracts, which should be replaced by a full description at a later time.)

lex

Function of Entry:

Lex does lexical analysis of the source program. It is called once per statement. Upon each call it reads the input stream and separates the characters into "tokens" such as identifiers, constants, operators, etc. It pools these tokens and returns with an array of pointers to token nodes.

Calling Sequence for Entry:

```
call lex;
```

Declaration of Arguments:

```
dcl  token_list(1000) ptr external static,
     statement_id fixed bin(31) external static;
```

Description of Arguments:

token_list is an array of pointers. The first n elements of the array are filled with pointers to the token nodes created by the lexical analyzer.

statement_id is a fixed binary number. The rightmost 15 bits contain the line number of the source program. The 15 bits starting with bit 2 contain the number of the statement within the line.

data

Function of Entry:

This is a data segment containing three tables which drive the lexical analyzer.

Calling Sequence for Entry:

```
action_matrix(i) = data$action(i);
new_state_matrix(i) = data$new_state(i);
token_type_matrix(i) = data$token_type(i);
```

Declaration of Arguments:

```
dcl  (data$action,
      data$new_state,
      data$token_type) (0:898) fixed bin (15) external;
```

Description of Arguments:

data$action	gives the action to be performed for each character read.
data$new_state	gives the next state for the lexical analyzer based on the previous state and input character.
data$token_type	gives the type of a token when it is recognized - e.g., identifier, bit_string, etc.

insert_token

Function of Entry:

This procedure forms part of the lexical analyzer. Its purpose is to maintain all token table nodes in a pooled, hash-coded table. It performs insertion and look-up in the table.

Calling Sequence for Entry:

```
call insert_token (p, string, type);
```

Declaration of Arguments:

```
dcl p ptr,
    string char(256) varying,
    type fixed bin(15);
```

Description of Arguments:

p is the value returned from insert_token. It points to the token node returned.

string is the character string representation of the token being inserted into the table.

type is the type of the token being inserted.