Published: 09/05/68

Identification

Expression Parsing
G. D. Chang

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

EXPRESSION

Function of Entry:

Checks to see if the list of tokens is an expression:

if it is: returns with "1"b and parses the expression and creates a computation tree representing the expression

if it is not: returns with "0"b

Calling Sequence for Entry:

```
bit = expression (k, arg, cur_block, backptr, contx);
```

Declaration of Arguments:

```
dcl  (arg, cur_block, backptr) ptr,
     (k, contx)  fixed bin(15);
```

Description of Arguments:

k - number in the token_list

cur_block - ptr to the current block node

backptr - ptr to the parent node

contx - =1 if called from procedure "reference"
        =0 if called from somewhere else

arg - ptr to the node representing the expression, if expression.

copy_exp

Function of Entry:

Copies an expression.

Calling Sequence for Entry:

```
call  copy_exp (inptr, backptr, outptr);
```

Declaration of Arguments:

```
dcl  (inptr, backptr, outptr) ptr;
```

Description of Arguments:

inptr - ptr to the node representing the expression to be copied

backptr - ptr to the parent node

outptr - ptr to the created node representing the same expression

CREATE_IDENTIFIER

Function of Entry:

Returns a ptr to the token_table node, whose string is created by the concatenation of "cp." and a decimal integer.

Calling Sequence for Entry:

```
ptr = create_identifier;
```

Declaration of Arguments:

Description of Arguments:

DISPLAY_POINTER

Function of Entry:

Returns a ptr to the token table node, whose string is created by the concatenation of "dp." and an octal integer.

Calling Sequence for Entry:

```
ptr = display_pointer (cur_block);
```

Declaration of Arguments:

```
dcl  cur_block ptr;
```

Description of Arguments:

cur_block - ptr to the current block node

DECLARE

Function of Entry:

Declares a name, creates its token_table_node, creates symbol_node, and a data_attribute_block.

Calling Sequence for Entry:

```
call  declare (cur_block, arg, type, class);
```

Declaration of Arguments:

```
dcl  (cur_block, arg) ptr,
     (type, class) fixed bin(15);
```

Description of Arguments:

cur_block - ptr to the current block_node.

arg - ptr to the created data_attribute_block

type - type of data in data_attribute_block

class - storage class in data_attribute_block

MAKE_DATT

Function of Entry:

Creates and initializes a data_attribute_block.

Calling Sequence for Entry:

```
call  make_datt(p);
```

Declaration of Arguments:

```
dcl  p  ptr;
```

Description of Arguments:

p - ptr to the created data_attribute_block

MAKE_LATT

Function of Entry:

Creates and initializes a label_attribute_block.

Calling Sequence for Entry:

```
call  make_latt(p);
```

Declaration of Arguments:

```
dcl  p  ptr;
```

Description of Arguments:

p - ptr to the created block

MAKE_TEMP

Function of Entry:

Creates and initializes a temporary node.
It also chains all the temporary nodes
together whose root is in the b --> block.context

Calling Sequence for Entry:

```
call  proc (b, p);
```

Declaration of Arguments:

```
dcl  (b, p) ptr;
```

Description of Arguments:

b - ptr to the current block node

p - ptr to the created temporary node