More extensions to Interlisp-D arithmetic

The following "logical" arithmetic functions are derived from Common Lisp, and have both macro and function definitions (the macros are for speed in running of compiled code). The following code equivalences are primarily for definitional purposes, and should not be considered an implementation (especially since the ireal implementation tends to be faster and less "consy" than apparent here).

```
(LOGNOT x)             ==  (LOGXOR x -1)
(BITTEST x mask)       ==  (NOT (ZEROP (LOGAND x mask)))
(BITCLEAR x mask)      ==  (LOGAND x (LOGNOT mask))
(BITSET x mask)        ==  (LOGOR x mask)
(MASK.1'S pos size)    ==  (LLSH (SUB1 (EXPT 2 size) pos)
(MASK.0's pos size)    ==  (LOGNOT (MASK.1'S pos size))
(LOADBYTE x pos size)  ==  (LOGAND (LRSH x pos) (MASK.1'S 0 size))
(DEPOSITBYTE x pos size byte)
                       ==  (LOGOR (BITCLEAR x (MASK.1'S pos size))
                                  (LLSH (LOGAND byte (MASK.1'S 0 size)) pos))
```

The notion of a "byte specifier" has been added; from a pair of positon and size arguments, a byte-spec is constructed by the macro BYTE [note reversal of arguments as compare with above functions]

```
    (BYTE size pos)
```

Similarly, the macros BYTESIZE and BYTEPOSITION will select out the two fields. [currently, byte-specs are implemented as a typerecord; Common Lisp leaves unspecified whether there are any range limitations on "size" or "pos"]

Two more "byte" functions are provided, with compiler macro support also:

```
(LDB bspec val)       == (LOADBYTE val (BYTEPOSITION bspec) (BYTESIZE bspec))
(DPB n bspec val)     == (DEPOSITBYTE val
                                      (BYTEPOSITION bspec)
                                      (BYTESIZE bspec)
                                      n)
```

The final function in this series is not quite so easy to describe -- ROT for "Rotate bits in field".

```
       (ROT x n fieldsize)
```

is a slight extension of the CommonLisp ROT function. It performs a bitwise left-rotation of the integer x, by n places, within a field of fieldsize bits wide; bits being shifted out of the position selected by

```
                (EXPT 2 (SUB1 fieldsize))
```

will flow into the "units" position. The optional argument fieldsize defaults to "cell" size (the integerlength of the current maximum fixp), and must either be a positive integer, or else be one of the litatoms CELL or WORD. In the latter two cases the appropriate numerical values are respectively substituted. A macro optimizes the case where fieldsize is WORD and n is 1.