File created: 18-Oct-93 15:42:46 **{Pele:mv:envos}<LispCore>Sources>CLTL2>LLARITH.;2**

previous date: 3-Sep-91 17:57:33 {Pele:mv:envos}<LispCore>Sources>CLTL2>LLARITH.;1

Read Table: XCL

Package: INTERLISP

Format: XCCS


```
(RPAQQ LLARITHCOMS
       ((COMS                                                    ; OPCODES
              (FNS IDIFFERENCE IGREATERP IQUOTIENT)
              ;; \slowplus2 \slowdifference \slowtimes2 \slowquotient are redefined in cmlarith
              (FNS \\SLOWIPLUS2 \\SLOWPLUS2 \\SLOWIDIFFERENCE \\SLOWDIFFERENCE \\SLOWIGREATERP \\SLOWLLSH1
                   \\SLOWLLSH8 \\SLOWLOGAND2 \\SLOWLOGOR2 \\SLOWLOGXOR2 \\SLOWLRSH1 \\SLOWLRSH8 \\SLOWITIMES2
                   \\SLOWTIMES2 \\SLOWIQUOTIENT \\SLOWQUOTIENT))
        (COMS                                                    ; IPLUS and IDIFFERENCE that smash result into their first arg
              (FNS \\BOXIPLUS \\BOXIDIFFERENCE))
                                                                 ; subfunctions
        (FNS \\MAKENUMBER)
        (FNS OVERFLOW)
        (INITVARS (\\OVERFLOW T))
        (CONSTANTS (MAX.SMALLP 65535)
               (MIN.SMALLP -65536)
               (MAX.FIXP 2147483647)
               (MIN.FIXP -2147483648)
               (\\SIGNBIT 32768))
        (FNS \\GETBASEFIXP \\PUTBASEFIXP \\PUTBASEFIXP.UFN)
        (EXPORT (DECLARE\: DONTCOPY (RECORDS FIXP)
                       (CONSTANTS (MAX.SMALL.INTEGER 65535)
                              (MAX.POS.HINUM 32767))
                       ;; Unbox changed to handle ratios
                       (MACROS .UNBOX. .NEGATE. .LLSH1. .LRSH1. .BOXIPLUS.)))
        (DECLARE\: DONTCOPY (MACROS OLD.UNBOX.))
        ;; Eqp modified to be like =
        (FNS EQP FIX IQUOTIENT IREMAINDER LLSH LRSH LSH RSH \\RSH)
        (DECLARE\: EVAL@COMPILE DONTCOPY (MACROS NBITS.OR.LESS .SUBSMALL. \\IQUOTREM))
                                                                 ; Machine independent arithmetic functions
        ;; MINUSP redefined in cmlarith
        (FNS MINUSP ILESSP IMINUS IPLUS ITIMES LOGAND LOGOR LOGXOR SUB1 ZEROP ADD1 GCD IEQP INTEGERLENGTH)
        ;; abs, difference, greaterp, plus, lessp, and times  redefined in cmlarith.
        ;; quotient and minus modified to handle ratios
        ;; remainder remains as is
        (FNS ABS DIFFERENCE GREATERP PLUS QUOTIENT REMAINDER LESSP MINUS TIMES)
        (FNS FMINUS FREMAINDER)
        (FNS RANDSET RAND EXPT)
        (DECLARE\: DONTEVAL@LOAD DOCOPY (VARS (RANDSTATE)
                                              (\\TOL 9.9999925E-6)))
        (GLOBALVARS RANDSTATE \\TOL)
        (COMS (FNS |PutUnboxed| \\PUTFIXP \\PUTSWAPPEDFIXP \\HINUM \\LONUM)
              (EXPORT (DECLARE\: DONTCOPY (MACROS |PutUnboxed|))))
        (DECLARE\: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS
               (ADDVARS (NLAMA)
                      (NLAML)
                      (LAMA MIN MAX IMIN IMAX FMIN FMAX ODDP TIMES PLUS LOGXOR LOGOR LOGAND ITIMES IPLUS)))
        ;; ODDP redefined in cmlarith
        (COMS (FNS POWEROFTWOP IMOD ODDP)
              (DECLARE\: DONTCOPY (MACROS .2^NP.)))
        (COMS                                                    ; MIN and MAX
              (FNS FLESSP FMAX FMIN GEQ IGEQ ILEQ IMAX IMIN LEQ MAX MIN)
              (DECLARE\: EVAL@COMPILE (ADDVARS (GLOBALVARS MAX.INTEGER MIN.INTEGER MAX.FLOAT MIN.FLOAT))))
        (DECLARE\: DONTCOPY DOEVAL@COMPILE DONTEVAL@LOAD (LOCALVARS . T))))

;; OPCODES

(DEFINEQ

(IDIFFERENCE
  (LAMBDA (X Y)                                                  (* |lmm| "11-FEB-82 14:02")
    ((OPCODES IDIFFERENCE)
     X Y)))
```

```
(IGREATERP
  (LAMBDA (X Y)                                                    (* |lmm| "11-FEB-82 14:02")
    ((OPCODES IGREATERP)
     X Y)))

(IQUOTIENT
  (LAMBDA (X Y)                                                    (* |lmm| "11-FEB-82 14:02")
    ((OPCODES IQUOTIENT)
     X Y)))

)

;; \slowplus2 \slowdifference \slowtimes2 \slowquotient are redefined in cmlarith

(DEFINEQ

(\\SLOWIPLUS2
  (LAMBDA (X Y)                                                    ; Edited  8-Apr-87 11:23 by jop
    (\\CALLME 'IPLUS)
    (PROG (HX LX HY LY SIGNX)
          (.UNBOX. X HX LX (GO RETBIG))
          (.UNBOX. Y HY LY (GO RETBIG))
          (SETQ SIGNX (IGREATERP HX MAX.POS.HINUM))
          (SETQ HX (COND
                      ((IGREATERP HX (IDIFFERENCE MAX.SMALL.INTEGER HY))
                       (IDIFFERENCE HX (ADD1 (IDIFFERENCE MAX.SMALL.INTEGER HY))))
                      (T (IPLUS HX HY))))                          ; Add high parts
          (SETQ LX (COND
                      ((IGREATERP LX (IDIFFERENCE MAX.SMALL.INTEGER LY))
                                                                   ; Carry into high part.
                       (SETQ HX (COND
                                   ((EQ HX MAX.SMALL.INTEGER)
                                    0)
                                   (T (ADD1 HX))))
                       (IDIFFERENCE LX (ADD1 (IDIFFERENCE MAX.SMALL.INTEGER LY))))
                      (T (IPLUS LX LY))))
          (COND
             ((AND (EQ SIGNX (IGREATERP HY MAX.POS.HINUM))
                   (NEQ SIGNX (IGREATERP HX MAX.POS.HINUM)))       ; overflow occurs if X and Y are same sign, but result is opposite
                                                                   ; sign
              (GO RETBIG)))
          (RETURN (\\MAKENUMBER HX LX))
      RETBIG
          (RETURN (\\BIGNUM.PLUS X Y)))))

(\\SLOWPLUS2
  (LAMBDA (X Y)                                                    ; Edited  8-Apr-87 11:24 by jop
    ;; UFN for PLUS Microcode generally handles the case of two args both FIXPs
    (\\CALLME 'PLUS)
    (PROG NIL
      LP  (RETURN (COND
                     ((OR (FLOATP X)
                          (FLOATP Y))
                      (FPLUS X Y))
                     ((NOT (FIXP X))
                      (SETQ X (LISPERROR "NON-NUMERIC ARG" X T))
                      (GO LP))
                     ((NOT (FIXP Y))
                      (SETQ Y (LISPERROR "NON-NUMERIC ARG" Y T))
                      (GO LP))
                     (T (IPLUS X Y)))))))

(\\SLOWIDIFFERENCE
  (LAMBDA (X Y)                                                    ; Edited  8-Apr-87 11:24 by jop
    (\\CALLME 'IDIFFERENCE)
    (PROG (HX LX HY LY SIGNX)
          (.UNBOX. X HX LX (GO RETBIG))
          (.UNBOX. Y HY LY (GO RETBIG))
     ;; Allow this unboxing before the following test so that error checking will be performed on Y
          (COND
             ((EQ Y 0)
              (RETURN (\\MAKENUMBER HX LX))))
          (.NEGATE. HY LY)
          (SETQ SIGNX (IGREATERP HX MAX.POS.HINUM))
          (COND
             ((COND
                 ((AND (ZEROP LY)
                       (EQ HY \\SIGNBIT))                          ; Y = -Y = Min.integer.  Overflow occurs if X is positive
                  (SETQ HX (LOGXOR HX HY))
                  (NOT SIGNX))
                 (T (SETQ HX (COND
                                ((IGREATERP HX (IDIFFERENCE MAX.SMALL.INTEGER HY))
```

```
                                (IDIFFERENCE HX (ADD1 (IDIFFERENCE MAX.SMALL.INTEGER HY))))
                               (T (IPLUS HX HY))))              ; Add high parts
                    (SETQ LX (COND
                               ((IGREATERP LX (IDIFFERENCE MAX.SMALL.INTEGER LY))
                                                                ; Carry into high part.
                                (SETQ HX (COND
                                           ((EQ HX MAX.SMALL.INTEGER)
                                            0)
                                           (T (ADD1 HX))))
                                (IDIFFERENCE LX (ADD1 (IDIFFERENCE MAX.SMALL.INTEGER LY))))
                               (T (IPLUS LX LY))))              ; overflow occurs if X and Y are same sign, but result is opposite
                                                                ; sign
                    (AND (EQ SIGNX (IGREATERP HY MAX.POS.HINUM))
                         (NEQ SIGNX (IGREATERP HX MAX.POS.HINUM)))))
                (GO RETBIG)))
          (RETURN (\\MAKENUMBER HX LX))
      RETBIG
          (RETURN (\\BIGNUM.DIFFERENCE X Y)))))
```

(**SLOWDIFFERENCE**

```
(LAMBDA (X Y)                                                   ; Edited  8-Apr-87 11:24 by jop
  ;; UFN for DIFFERENCE Microcode generally handles the case of two args both FIXPs
  (\\CALLME 'DIFFERENCE)
  (PROG NIL
    LP  (RETURN (COND
                  ((OR (FLOATP X)
                       (FLOATP Y))
                   (FDIFFERENCE X Y))
                  ((NOT (FIXP X))
                   (SETQ X (LISPERROR "NON-NUMERIC ARG" X T))
                   (GO LP))
                  ((NOT (FIXP Y))
                   (SETQ Y (LISPERROR "NON-NUMERIC ARG" Y T))
                   (GO LP))
                  (T (IDIFFERENCE X Y)))))))
```

(**SLOWIGREATERP**

```
(LAMBDA (X Y)                                                   (* |lmm| "12-Apr-85 07:35")
  (\\CALLME 'IGREATERP)
  (PROG (HX LX HY LY)
        (.UNBOX. X HX LX (GO RETBIG))
        (.UNBOX. Y HY LY (GO RETBIG))
        (RETURN (COND
                  ((EQ HX HY)
                   (IGREATERP LX LY))
                  (T (IGREATERP (LOGXOR HX \\SIGNBIT)
                                (LOGXOR HY \\SIGNBIT)))))
    RETBIG
        (RETURN (EQ 1 (\\BIGNUM.COMPARE X Y))))))
```

(**SLOWLLSH1**

```
(LAMBDA (X)                                                     (* |lmm| "13-OCT-82 15:27")
  (PROG (LO HI)
        (.UNBOX. X HI LO)
        (RETURN (\\MAKENUMBER (IPLUS (LLSH (LOGAND HI 32767)
                                           1)
                                     (COND
                                        ((IGREATERP LO 32767)
                                         1)
                                        (T 0)))
                              (LLSH (LOGAND LO 32767)
                                    1))))))
```

(**SLOWLLSH8**

```
(LAMBDA (X)                                                     (* |lmm| "13-OCT-82 15:28")
  (PROG (HI LO)
        (.UNBOX. X HI LO)
        (RETURN (\\MAKENUMBER (IPLUS (LLSH (LOGAND HI 255)
                                           8)
                                     (LRSH LO 8))
                              (LLSH (LOGAND LO 255)
                                    8))))))
```

(**SLOWLOGAND2**

```
(LAMBDA (X Y)                                                   (* |lmm| "12-Apr-85 07:44")
  (\\CALLME 'LOGAND)
  (PROG (XH XL YH YL)
        (.UNBOX. X XH XL (GO RETBIG))
        (.UNBOX. Y YH YL (GO RETBIG))
        (RETURN (\\MAKENUMBER (LOGAND XH YH)
                              (LOGAND XL YL)))
```

```
      RETBIG
          (RETURN (\\BIGNUM.LOGAND X Y)))))
```

(\\SLOWLOGOR2

```
  (LAMBDA (X Y)                                          (* |lmm| "12-Apr-85 07:48")
    (\\CALLME 'LOGOR)
    (PROG (XH XL YH YL)
          (.UNBOX. X XH XL (GO RETBIG))
          (.UNBOX. Y YH YL (GO RETBIG))
          (RETURN (\\MAKENUMBER (LOGOR XH YH)
                          (LOGOR XL YL)))
      RETBIG
          (RETURN (\\BIGNUM.LOGOR X Y)))))
```

(\\SLOWLOGXOR2

```
  (LAMBDA (X Y)                                          (* |lmm| "12-Apr-85 07:51")
    (\\CALLME 'LOGXOR)
    (PROG (XH XL YH YL)
          (.UNBOX. X XH XL (GO RETBIG))
          (.UNBOX. Y YH YL (GO RETBIG))
          (RETURN (\\MAKENUMBER (LOGXOR XH YH)
                          (LOGXOR XL YL)))
      RETBIG
          (RETURN (\\BIGNUM.LOGXOR X Y)))))
```

(\\SLOWLRSH1

```
  (LAMBDA (X)                                            (* |JonL| "27-Sep-84 22:59")
    (PROG (HI LO)
          (.UNBOX. X HI LO)
          (RETURN (\\MAKENUMBER (LRSH HI 1)
                          (IPLUS (LRSH LO 1)
                                 (COND
                                    ((EQ 0 (LOGAND HI 1))
                                     0)
                                    (T 32768))))))))
```

(\\SLOWLRSH8

```
  (LAMBDA (X)                                            (* |lmm| "13-OCT-82 15:29")
    (PROG (HI LO)
          (.UNBOX. X HI LO)
          (RETURN (\\MAKENUMBER (LRSH HI 8)
                          (IPLUS (LLSH (LOGAND HI 255)
                                       8)
                                 (LRSH LO 8)))))))
```

(\\SLOWITIMES2

```
  (LAMBDA (X Y)                                          ; Edited  8-Apr-87 11:26 by jop
    (\\CALLME 'ITIMES)
    (COND
       ((OR (EQ X 0)
            (EQ Y 0))
        0)
       (T (PROG (HX HY LX LY SIGN (HR 0)
                    (LR 0)
                    CARRY)
                (.UNBOX. X HX LX (GO RETBIG))
                (.UNBOX. Y HY LY (GO RETBIG))
                (COND
                   ((IGREATERP HX MAX.POS.HINUM)
                    (|if| (EQUAL X MIN.FIXP)
                        |then| (GO RETBIG))
                    (.NEGATE. HX LX)
                    (SETQ SIGN T)))
                (COND
                   ((IGREATERP HY MAX.POS.HINUM)
                    (|if| (EQUAL Y MIN.FIXP)
                        |then| (GO RETBIG))
                    (.NEGATE. HY LY)
                    (SETQ SIGN (NOT SIGN))))
                (COND
                   ((NEQ HY 0)
                    (COND
                       ((NEQ HX 0)
                        (GO OVER)))
                    (|swap| LX LY)
                    (|swap| HX HY)))
          MLP (COND
                   ((ODDP (PROG1 LY
                                 (SETQ LY (LRSH LY 1))))
                    (COND
                       ((IGREATERP LR (IDIFFERENCE MAX.SMALL.INTEGER LX))
                                                         ; low parts overflow
```

```
                                                        ; make the low word be the less significant bits and return the
                                                        ; carry.
                    (SETQ LR (IDIFFERENCE LR (IDIFFERENCE MAX.SMALL.INTEGER (SUB1 LX))))
                    (SETQ CARRY 1))
                   (T                                   ; no carry just add the low halves.
                      (SETQ LR (IPLUS LR LX))
                      (SETQ CARRY 0)))
               ;; the low order part of the answer has been set and CARRY is the numeric value of the carry from the low part either 0 or 1
               (COND
                  ((IGREATERP (SETQ HR (IPLUS HR HX CARRY))
                              MAX.POS.HINUM)
                   (COND
                      ((AND (EQ LY 0)
                            SIGN
                            (EQ HR (ADD1 MAX.POS.HINUM))
                            (EQ LR 0))
                       (RETURN MIN.FIXP)))
                   (GO OVER)))))
          (COND
             ((ZEROP LY)
              (GO RET)))
          (COND
             ((IGEQ HX (LRSH (ADD1 MAX.POS.HINUM)
                             1))
              (GO OVERTEST)))
          (.LLSH1. HX LX)
          (GO MLP)
       OVERTEST
          (COND
             ((AND (EQ HX (LRSH (ADD1 MAX.POS.HINUM)
                                1))
                   (ZEROP LX)
                   SIGN
                   (EQ LY 1)
                   (EQ HR 0)
                   (EQ LR 0))                           ; odd special case
              (RETURN MIN.FIXP)))
       OVER
          (GO RETBIG)
       RET (COND
             (SIGN (.NEGATE. HR LR)))
          (RETURN (\\MAKENUMBER HR LR))
       RETBIG
          (RETURN (\\BIGNUM.TIMES X Y)))))))
```

(\\SLOWTIMES2

```
(LAMBDA (X Y)                                           (* |lmm| "21-Aug-84 16:22")
                                                        (* UFN |for| TIMES |Microcode| |generally| |handles| |the| |case|
  |of| |two| |args| |both| FIXPS)
  (\\CALLME 'TIMES)
  (PROG NIL
    LP  (RETURN (COND
                   ((OR (FLOATP X)
                        (FLOATP Y))
                    (FTIMES X Y))
                   ((NOT (FIXP X))
                    (SETQ X (LISPERROR "NON-NUMERIC ARG" X T))
                    (GO LP))
                   ((NOT (FIXP Y))
                    (SETQ Y (LISPERROR "NON-NUMERIC ARG" Y T))
                    (GO LP))
                   (T (ITIMES X Y)))))))
```

(\\SLOWIQUOTIENT

```
(LAMBDA (X Y)                                           (* |lmm| " 2-Jul-84 17:12")
  (\\CALLME 'IQUOTIENT)
  (\\IQUOTREM X Y X)
  X))
```

(\\SLOWQUOTIENT

```
(LAMBDA (X Y)                                           ; Edited  8-Apr-87 11:26 by jop
  ;; UFN for QUOTIENT Microcode generally handles the case of two args both FIXPs
  (\\CALLME 'QUOTIENT)
  (PROG NIL
    LP  (RETURN (COND
                   ((OR (FLOATP X)
                        (FLOATP Y))
                    (FQUOTIENT X Y))
                   ((NOT (FIXP X))
                    (SETQ X (LISPERROR "NON-NUMERIC ARG" X T))
                    (GO LP))
                   ((NOT (FIXP Y))
```

```
                    (SETQ Y (LISPERROR "NON-NUMERIC ARG" Y T))
                    (GO LP))
                   (T (IQUOTIENT X Y)))))))

)

;; IPLUS and IDIFFERENCE that smash result into their first arg

(DEFINEQ

(\\BOXIPLUS
  (LAMBDA (X Y)                                          ; Edited  8-Apr-87 11:27 by jop
    ;; UFN for BOXIPLUS ipcode
    (.BOXIPLUS. X Y)))

(\\BOXIDIFFERENCE
  (LAMBDA (X Y)                                          ; Edited  8-Apr-87 11:27 by jop
    (PROG ((HX (\\GETBASE X 0))
           (LX (\\GETBASE X 1))
           HY LY)
          (.UNBOX. Y HY LY)
          (.NEGATE. HY LY)
          (SETQ HX (COND
                     ((IGREATERP HX (IDIFFERENCE MAX.SMALL.INTEGER HY))
                      (IDIFFERENCE HX (ADD1 (IDIFFERENCE MAX.SMALL.INTEGER HY))))
                     (T (IPLUS HX HY))))                 ; Add high parts
          (\\PUTBASE X 1 (COND
                           ((IGREATERP LX (IDIFFERENCE MAX.SMALL.INTEGER LY))
                                                         ; Carry into high part.
                            (SETQ HX (COND
                                       ((EQ HX MAX.SMALL.INTEGER)
                                        0)
                                       (T (ADD1 HX))))
                            (IDIFFERENCE LX (ADD1 (IDIFFERENCE MAX.SMALL.INTEGER LY))))
                           (T (IPLUS LX LY))))
          (\\PUTBASE X 0 HX)
          (RETURN X))))

)

;; subfunctions

(DEFINEQ

(\\MAKENUMBER
  (LAMBDA (N0 N1)                                        ; Edited  8-Apr-87 11:28 by jop
    ;; used as punt case for arith opcodes which create large numbers
    (SETQ N1 (.COERCE.TO.SMALLPOSP. N1))
    (SELECTC (SETQ N0 (.COERCE.TO.SMALLPOSP. N0))
        (0 N1)
        (65535                                           ; This is a word's worth of 1 bits
               (\\VAG2 |\\SmallNegHi| N1))
        (|create| FIXP
               HINUM _ N0
               LONUM _ N1))))

)

(DEFINEQ

(OVERFLOW
  (LAMBDA (FLG)                                          (* |lmm:| 14-JAN-76 1 6)
    (PROG1 \\OVERFLOW
        (SETQ \\OVERFLOW (SELECTQ FLG
                           (NIL NIL)
                           (T T)
                           0)))))

)

(RPAQ? \\OVERFLOW T)

(DECLARE\: EVAL@COMPILE

(RPAQQ MAX.SMALLP 65535)

(RPAQQ MIN.SMALLP -65536)

(RPAQQ MAX.FIXP 2147483647)

(RPAQQ MIN.FIXP -2147483648)

(RPAQQ \\SIGNBIT 32768)
```

```
(CONSTANTS (MAX.SMALLP 65535)
       (MIN.SMALLP -65536)
       (MAX.FIXP 2147483647)
       (MIN.FIXP -2147483648)
       (\\SIGNBIT 32768))
)

(DEFINEQ

(\\GETBASEFIXP
  (LAMBDA (BASE OFFST)                                    (* |lmm| " 5-Jan-85 23:11")
    ((LAMBDA (|\\NewBaseAddr|)
       (\\MAKENUMBER (\\GETBASE |\\NewBaseAddr| 0)
              (\\GETBASE |\\NewBaseAddr| 1)))
     (\\ADDBASE BASE OFFST))))

(\\PUTBASEFIXP
  (LAMBDA (BASE OFFST VAL)                                (* |lmm| " 5-Jan-85 23:16")
    (PROG (HI LO)
          (.XUNBOX. VAL HI LO)
          (\\PUTBASE BASE OFFST HI)
          (\\PUTBASE BASE (ADD1 OFFST)
                 LO)
      VAL (RETURN VAL))))

(\\PUTBASEFIXP.UFN
  (LAMBDA (BASE VAL OFFST)                                (* |lmm| " 5-Jan-85 23:25")
    (PROG (HI LO)
          (.XUNBOX. VAL HI LO)
          (\\PUTBASE BASE OFFST HI)
          (\\PUTBASE BASE (ADD1 OFFST)
                 LO)
      VAL (RETURN VAL))))

)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE\: DONTCOPY

(DECLARE\: EVAL@COMPILE

(BLOCKRECORD FIXP ((HINUM WORD)
                   (LONUM WORD))
       (CREATE (CREATECELL \\FIXP))
       (TYPE? (EQ (NTYPX DATUM)
                  \\FIXP)))
)

(DECLARE\: EVAL@COMPILE

(RPAQQ MAX.SMALL.INTEGER 65535)

(RPAQQ MAX.POS.HINUM 32767)

(CONSTANTS (MAX.SMALL.INTEGER 65535)
       (MAX.POS.HINUM 32767))
)

(DECLARE\: EVAL@COMPILE

(PUTPROPS .UNBOX. MACRO (ARGS (LET ((ARG-FORM (CAR ARGS))
                                    (HIGH-VAR (CADR ARGS))
                                    (LOW-VAR (CADDR ARGS))
                                    (BIGNUM-FORM (CADDDR ARGS)))
                                   `(PROG NIL
                                      UBLP
                                          (SELECTC (NTYPX ,ARG-FORM)
                                              (\\FIXP (SETQ ,HIGH-VAR (|ffetch| (FIXP HINUM) |of| ,ARG-FORM))
                                                      (SETQ ,LOW-VAR (|ffetch| (FIXP LONUM) |of| ,ARG-FORM)))
                                              (\\SMALLP (COND
                                                           ((ILEQ 0 ,ARG-FORM)
                                                            (SETQ ,HIGH-VAR 0)
                                                            (SETQ ,LOW-VAR ,ARG-FORM))
                                                           (T (SETQ ,HIGH-VAR 65535)
                                                              (SETQ ,LOW-VAR (\\LOLOC ,ARG-FORM)))))
                                              (\\FLOATP (SETQ ,ARG-FORM (\\FIXP.FROM.FLOATP ,ARG-FORM))
                                                        (GO UBLP))
                                              (COND
                                                 ((TYPENAMEP ,ARG-FORM 'RATIO)
                                                  (SETQ ,ARG-FORM (IQUOTIENT (CL::RATIO-NUMERATOR ,ARG-FORM)
                                                                             (CL::RATIO-DENOMINATOR ,ARG-FORM)))
                                                  (GO UBLP))
                                                 ,@(COND
```

```
                                                    (BIGNUM-FORM `(((CL:INTEGERP ,ARG-FORM)
                                                                   ,BIGNUM-FORM)))
                                                    (T `(((CL:INTEGERP ,ARG-FORM)
                                                          (\\ILLEGAL.ARG ,ARG-FORM)))))
                                                (T (CL::%NOT-NONCOMPLEX-NUMBER-ERROR ,ARG-FORM))))))))

(PUTPROPS .NEGATE. MACRO ((HY LY)
                          (COND
                             ((EQ 0 LY)
                              (AND (NEQ HY 0)
                                   (SETQ HY (ADD1 (IDIFFERENCE MAX.SMALL.INTEGER HY)))))
                             (T (SETQ HY (IDIFFERENCE MAX.SMALL.INTEGER HY))
                                (SETQ LY (ADD1 (IDIFFERENCE MAX.SMALL.INTEGER LY)))))))

(PUTPROPS .LLSH1. MACRO ((HI LO)                                    ; shift the pair left one, assuming no overflow
                         (SETQ HI (LLSH HI 1))
                         (SETQ LO (LLSH (COND
                                           ((IGREATERP LO MAX.POS.HINUM)
                                            (|add| HI 1)
                                            (LOGAND LO MAX.POS.HINUM))
                                           (T LO))
                                        1))))

(PUTPROPS .LRSH1. MACRO ((HI LO)
                         (SETQ LO (LRSH LO 1))
                         (COND
                            ((NEQ (LOGAND HI 1)
                                  0)
                             (SETQ LO (IPLUS LO \\SIGNBIT))))
                         (SETQ HI (LRSH HI 1))))

(PUTPROPS .BOXIPLUS. MACRO (OPENLAMBDA (X Y)
                           (PROG ((HX (\\GETBASE X 0))
                                  (LX (\\GETBASE X 1))
                                  HY LY)
                                 (.UNBOX. Y HY LY)
                                 (SETQ HX (COND
                                             ((IGREATERP HX (IDIFFERENCE MAX.SMALL.INTEGER HY))
                                              (IDIFFERENCE HX (ADD1 (IDIFFERENCE MAX.SMALL.INTEGER HY))))
                                             (T (IPLUS HX HY))))
                                                          (* |Add| |high| |parts|)
                                 (\\PUTBASE X 1 (COND
                                                   ((IGREATERP LX (IDIFFERENCE MAX.SMALL.INTEGER LY))
                                                                  (* |Carry| |into| |high| |part.|)
                                                    (SETQ HX (COND
                                                                ((EQ HX MAX.SMALL.INTEGER)
                                                                 0)
                                                                (T (ADD1 HX))))
                                                    (IDIFFERENCE LX (ADD1 (IDIFFERENCE MAX.SMALL.INTEGER LY)
                                                                          )))
                                                   (T (IPLUS LX LY))))
                                 (\\PUTBASE X 0 HX)
                                 (RETURN X))))
)
)

;; END EXPORTED DEFINITIONS

(DECLARE\: DONTCOPY

(DECLARE\: EVAL@COMPILE

(PUTPROPS OLD.UNBOX. MACRO ((V HV LV BIGNUMFORM)
                            (PROG NIL
                              UBLP
                                   (SELECTC (NTYPX V)
                                        (\\FIXP (SETQ HV (|ffetch| (FIXP HINUM) |of| V))
                                                (SETQ LV (|ffetch| (FIXP LONUM) |of| V)))
                                        (\\SMALLP (COND
                                                     ((ILEQ 0 V)
                                                      (SETQ HV 0)
                                                      (SETQ LV V))
                                                     (T (SETQ HV 65535)
                                                        (SETQ LV (\\LOLOC V)))))
                                        (\\FLOATP (SETQ V (\\FIXP.FROM.FLOATP V))
                                                  (GO UBLP))
                                        (|if| (TYPENAMEP V 'BIGNUM)
                                             |then| (|if| 'BIGNUMFORM
                                                          |then| BIGNUMFORM
                                                        |else| (SETQ V (\\LISPERROR V "ARG NOT FIXP" T))
                                                               (GO UBLP))
                                          |else| (SETQ V (LISPERROR "NON-NUMERIC ARG" V T))
                                                 (GO UBLP))))))
)
)
```

;; Eqp modified to be like =

```
(DEFINEQ
```

(EQP

```
  (LAMBDA (X Y)                                          ; Edited  7-Dec-88 09:04 by jds
    (COND
       ((EQ X Y)
        T)
       ((AND (FIXP X)
             (FIXP Y))
        (IEQP X Y))
       ((AND (OR (FLOATP X)
                 (FIXP X))
             (OR (FLOATP Y)
                 (FIXP Y)))
        (FEQP X Y))
       ((AND (NUMBERP X)
             (NUMBERP Y))
        (= X Y))
       (T (\\EXTENDED.EQP X Y)))))
```

(FIX

```
  (LAMBDA (N)                                            ; Edited  8-Apr-87 11:30 by jop
    ;; FIX compiles open
    (IPLUS N 0)))
```

(IQUOTIENT

```
  (LAMBDA (X Y)                                          (* |lmm| "11-FEB-82 14:02")
    ((OPCODES IQUOTIENT)
     X Y)))
```

(IREMAINDER

```
  (LAMBDA (X Y)                                          (* |edited:| "29-APR-82 05:01")
    (\\IQUOTREM X Y NIL Y)
    Y))
```

(LLSH

```
  (LAMBDA (X N)                                          (* |lmm| "13-OCT-82 15:30")
    (COND
       ((IGREATERP 0 N)
        (LRSH X (IMINUS N)))
       (T (PROG (XHI XLO)
                (.UNBOX. X XHI XLO)
                (COND
                   ((IGREATERP N 31)
                    (RETURN 0)))
                (COND
                   ((IGREATERP N 15)
                    (SETQ XHI XLO)
                    (SETQ XLO 0)
                    (SETQ N (IDIFFERENCE N 16))))
                (COND
                   ((IGREATERP N 7)
                    (SETQ XHI (IPLUS (LLSH (LOGAND XHI 255)
                                           8)
                                     (LRSH XLO 8)))
                    (SETQ XLO (LLSH (LOGAND XLO 255)
                                    8))
                    (SETQ N (IDIFFERENCE N 8))))
                (FRPTQ N (SETQ XHI (LOGAND XHI MAX.POS.HINUM))
                       (.LLSH1. XHI XLO))
                (RETURN (\\MAKENUMBER XHI XLO)))))))
```

(LRSH

```
  (LAMBDA (X N)                                          ; Edited  8-Apr-87 11:30 by jop
    ;; assumes case where n is constant and 8 or 1 handled in microcode or by \SLOWLRSHn
    (COND
       ((IGREATERP 0 N)
        (LLSH X (IMINUS N)))
       (T (PROG (XHI XLO)
                (.UNBOX. X XHI XLO)
                (COND
                   ((IGREATERP N 31)
                    (RETURN 0)))
                (COND
                   ((IGREATERP N 15)
                    (SETQ XLO XHI)
                    (SETQ XHI 0)
                    (SETQ N (IDIFFERENCE N 16))))
```

```
                (COND
                   ((IGREATERP N 7)
                    (SETQ XLO (IPLUS (LRSH XLO 8)
                                     (LLSH (LOGAND XHI 255)
                                           8)))
                    (SETQ XHI (LRSH XHI 8))
                    (SETQ N (IDIFFERENCE N 8))))
                (FRPTQ N (.LRSH1. XHI XLO))
                (RETURN (\\MAKENUMBER XHI XLO)))))))
```

(LSH

```
(LAMBDA (X N)                                          ; Edited  7-Apr-89 16:19 by jds
```

;; Arithmetic left shift. Since this punts on the dorado, and RSH is optimized to be LSH for the Sun, we have to use \RSH here (the bottoming-out ;; version of RSH).

```
(COND
   ((ILEQ N 0)
    (COND
       ((EQ N 0)
        X)
       (T (\\RSH X (IMINUS N)))))
   ((EQ X 0)
    0)
   ((IGREATERP N (CONSTANT (INTEGERLENGTH MAX.FIXP)))
    (\\BIGNUM.LSH X N))
   (T (FRPTQ N (SETQ X (IPLUS X X)))
      X))))
```

(RSH

```
(LAMBDA (X N)                                          ; Edited  7-Apr-89 16:20 by jds
```

;; Arithmetic Right-shift. There's an optimizer on this function, so just call the underlying implementation.

```
(\\RSH X N)))
```

(\\RSH

```
(LAMBDA (X N)                                          ; Edited  7-Apr-89 16:21 by jds
```

;; This is the version of RSH where things bottom out if LSH doesn't handle all the possible cases.

```
(COND
   ((IGREATERP 0 N)
    (LSH X (IMINUS N)))
   ((EQ X 0)
    0)
   (T (PROG (XHI XLO)
            (.UNBOX. X XHI XLO (GO RETBIG))
            (COND
               ((IGREATERP N 31)
                (RETURN (COND
                           ((IGREATERP XHI 32767)          ; X WAS NEGATIVE
                            -1)
                           (T 0)))))
            (COND
               ((IGREATERP N 15)
                (SETQ XLO XHI)
                (SETQ XHI (COND
                             ((IGREATERP XHI 32767)
                              65535)
                             (T 0)))
                (SETQ N (IDIFFERENCE N 16))))
            (COND
               ((IGREATERP N 7)
                (SETQ XLO (IPLUS (LRSH XLO 8)
                                 (LLSH (LOGAND XHI 255)
                                       8)))
                (SETQ XHI (IPLUS (LRSH XHI 8)
                                 (COND
                                    ((IGREATERP XHI 32767)
                                     65280)
                                    (T 0))))
                (SETQ N (IDIFFERENCE N 8))))
            (FRPTQ N (SETQ XLO (IPLUS (LRSH XLO 1)
                                      (COND
                                         ((EQ 0 (LOGAND XHI 1))
                                          0)
                                         (T 32768))))
                   (SETQ XHI (IPLUS (LRSH XHI 1)
                                    (LOGAND XHI 32768))))
            (RETURN (\\MAKENUMBER XHI XLO))
         RETBIG
            (RETURN (\\BIGNUM.LSH X (IMINUS N)))))))

)

(DECLARE\: EVAL@COMPILE DONTCOPY
```

```
(DECLARE\: EVAL@COMPILE

(PUTPROPS NBITS.OR.LESS MACRO ((X N)
                               (ILESSP X (CONSTANT (LLSH 1 N)))))

(PUTPROPS .SUBSMALL. MACRO ((X Y)                          ; Subtract Y from X, returning the borrow out of the next word
                            (COND
                               ((ILEQ Y X)
                                (SETQ X (IDIFFERENCE X Y))
                                0)
                               (T (SETQ X (ADD1 (IDIFFERENCE MAX.SMALL.INTEGER (IDIFFERENCE Y X))))
                                  1))))

(PUTPROPS \\IQUOTREM MACRO ((X Y QUO REM)
                            (PROG (HX LX HY LY SIGNQUOTIENT SIGNREMAINDER (CNT 0)
                                      (HZ 0)
                                      (LZ 0))
                                  (.UNBOX. X HX LX (GO RETBIG))
                                  (.UNBOX. Y HY LY (GO RETBIG))
                                  (COND
                                     ((IGREATERP HX MAX.POS.HINUM)
                                      (.NEGATE. HX LX)
                                      (SETQ SIGNQUOTIENT (SETQ SIGNREMAINDER T))))
                                                           ; Remainder has sign of dividend
                                  (COND
                                     ((IGREATERP HY MAX.POS.HINUM)
                                      (.NEGATE. HY LY)
                                      (SETQ SIGNQUOTIENT (NOT SIGNQUOTIENT))))
                                  (COND
                                     ((NEQ HX 0)
                                      (GO BIGDIVIDEND))
                                     ((NEQ HY 0)           ; Y is big, X is small, so result is 0
                                      (GO DONE))
                                     ((EQ 0 LX)
                                      (GO RET0))
                                     ((EQ 0 LY)
                                      (GO DIVZERO))
                                     ((EQ LY 1)
                                      (SETQ LZ LX)
                                      (SETQ LX 0)
                                      (GO DONE)))          ; here we are dividing small X by small Y, and we know Y gt 1
                                  LP1                      ; shift Y left until it is as big as X, and count how many times
                                  (COND
                                     ((AND (ILESSP LY LX)
                                           (ILEQ LY MAX.POS.HINUM))
                                      (SETQ LY (LLSH LY 1))
                                      (SETQ CNT (ADD1 CNT))
                                      (GO LP1)))
                                  LP2
                                  ;; now start dividing Y into X by subtracting and shifting, ending up with Y shifted back where it started
                                  (COND
                                     ((ILEQ LY LX)
                                      (SETQ LX (IDIFFERENCE LX LY))
                                                           ; Y divides X once, so add bit into quotient
                                      (SETQ LZ (ADD1 LZ))))
                                  (SETQ LY (LRSH LY 1))
                                  (SETQ CNT (SUB1 CNT))
                                  (COND
                                     ((IGEQ CNT 0)
                                      (SETQ LZ (LLSH LZ 1))
                                      (GO LP2)))
                                  (GO DONE)
                                  BIGDIVIDEND
                                                           ; X is big, so result may be big.  Algorithm is same as above, but
                                                           ; everything is doubled in length
                                  (COND
                                     ((EQ 0 HY)
                                      (COND
                                         ((EQ 0 (SETQ HY LY))
                                          (GO DIVZERO))
                                         ((AND SIGNREMAINDER (NULL SIGNQUOTIENT)
                                               (EQ 1 LY)
                                               (EQ HX \\SIGNBIT)
                                               (EQ 0 LX))  ; Means that X is MIN.FIXP and Y is -1
                                          (GO RETBIG)))
                                      (SETQ LY 0)
                                      (SETQ CNT 16))
                                     ((AND SIGNREMAINDER (NULL SIGNQUOTIENT)
                                           (EQ 0 LX)
                                           (EQ HX \\SIGNBIT)
                                           (EQ 0 HY)
                                           (EQ 1 LY))      ; Means that X is MIN.FIXP and Y is -1
                                      (GO RETBIG)))
                                  BIGLP
                                  (COND
```

```
                ((AND (OR (AND (EQ HY HX)
                               (ILESSP LY LX))
                          (ILESSP HY HX))
                      (ILESSP HY MAX.POS.HINUM))
                 (.LLSH1. HY LY)
                 (SETQ CNT (ADD1 CNT))
                 (GO BIGLP)))
          BIGLP2
               (COND
                  ((OR (ILESSP HY HX)
                       (AND (EQ HY HX)
                            (ILEQ LY LX)))    ; Y divides X, so subtract Y from X and put a bit in quotient
                   (SETQ HX (IDIFFERENCE (IDIFFERENCE HX HY)
                                         (.SUBSMALL. LX LY)))
                   (SETQ LZ (ADD1 LZ))        ; note that this never overflows, because of the preceding left
                                              ; shift

                   ))
               (.LRSH1. HY LY)
               (SETQ CNT (SUB1 CNT))
               (COND
                  ((IGEQ CNT 0)
                   (.LLSH1. HZ LZ)
                   (GO BIGLP2)))
          DONE
               (COND
                  ('REM                       ; remainder is left in X
                        (COND
                           (SIGNREMAINDER (.NEGATE. HX LX)))
                        (SETQ REM (\\MAKENUMBER HX LX))))
               (COND
                  ('QUO (COND
                           (SIGNQUOTIENT (.NEGATE. HZ LZ)))
                        (SETQ QUO (\\MAKENUMBER HZ LZ))))
               (RETURN)
          DIVZERO
               (SELECTQ \\OVERFLOW
                    (T (ERROR "DIVIDE BY ZERO" Y))
                    (GO RET0))
          RET0
               (COND
                  ('REM (SETQ REM 0)))
               (COND
                  ('QUO (SETQ QUO 0)))
               (RETURN)
          RETBIG
               (|if| 'QUO
                   |then| (SETQ QUO (\\BIGNUM.QUOTIENT X Y)))
               (|if| 'REM
                   |then| (SETQ REM (\\BIGNUM.REMAINDER X Y)))
               (RETURN))))
)
)
```

;; Machine independent arithmetic functions

;; MINUSP redefined in cmlarith

```
(DEFINEQ
```

(MINUSP

```
  (LAMBDA (X)                                          (* FS "22-Nov-86 20:47")
                                                       (* "Replaced by Roach (via MOVD) in CMLARITH to handle
                                                       RATIOS")

    (COND
       ((FLOATP X)
        (FGREATERP 0.0 X))
       (T (IGREATERP 0 X)))))
```

(ILESSP

```
  (LAMBDA (X Y)
    (IGREATERP Y X)))
```

(IMINUS

```
  (LAMBDA (X)
    (IDIFFERENCE 0 X)))
```

(IPLUS

```
  (LAMBDA N                                            ; Edited  8-Apr-87 11:34 by jop
    ;; called only by interpreted code --- this defn relies on fact that compiler turns IPLUS calls into sequences of opcodes
    (SELECTQ N
         (2 (IPLUS (ARG N 1)
                   (ARG N 2)))
```

```
        (1 (IPLUS (ARG N 1)))
        (0 (IPLUS))
        (PROG ((R (IPLUS (ARG N 1)
                         (ARG N 2)
                         (ARG N 3)))
               (J 4))
          LP  (COND
                 ((ILEQ J N)
                  (SETQ R (IPLUS R (ARG N J)))
                  (SETQ J (ADD1 J))
                  (GO LP)))
              (RETURN R)))))
```

(ITIMES

```
(LAMBDA N                                                    ; Edited 8-Apr-87 11:34 by jop
```

;; called only by interpreted code --- this defn relies on fact that compiler turns ITIMES calls into sequences of opcodes

```
(SELECTQ N
     (2 (ITIMES (ARG N 1)
                (ARG N 2)))
     (1 (ITIMES (ARG N 1)))
     (0 (ITIMES))
     (PROG ((R (ITIMES (ARG N 1)
                       (ARG N 2)
                       (ARG N 3)))
            (J 4))
       LP  (COND
              ((ILEQ J N)
               (SETQ R (ITIMES R (ARG N J)))
               (SETQ J (ADD1 J))
               (GO LP)))
           (RETURN R)))))
```

(LOGAND

```
(LAMBDA N                                                    ; Edited 8-Apr-87 11:34 by jop
```

;; called only by interpreted code --- this defn relies on fact that compiler turns LOGAND calls into sequences of opcodes

```
(SELECTQ N
     (2 (LOGAND (ARG N 1)
                (ARG N 2)))
     (1 (LOGAND (ARG N 1)))
     (0 (LOGAND))
     (PROG ((R (LOGAND (ARG N 1)
                       (ARG N 2)
                       (ARG N 3)))
            (J 4))
       LP  (COND
              ((ILEQ J N)
               (SETQ R (LOGAND R (ARG N J)))
               (SETQ J (ADD1 J))
               (GO LP)))
           (RETURN R)))))
```

(LOGOR

```
(LAMBDA N                                                    ; Edited 8-Apr-87 11:34 by jop
```

;; called only by interpreted code --- this defn relies on fact that compiler turns LOGOR calls into sequences of opcodes

```
(SELECTQ N
     (2 (LOGOR (ARG N 1)
               (ARG N 2)))
     (1 (LOGOR (ARG N 1)))
     (0 (LOGOR))
     (PROG ((R (LOGOR (ARG N 1)
                      (ARG N 2)
                      (ARG N 3)))
            (J 4))
       LP  (COND
              ((ILEQ J N)
               (SETQ R (LOGOR R (ARG N J)))
               (SETQ J (ADD1 J))
               (GO LP)))
           (RETURN R)))))
```

(LOGXOR

```
(LAMBDA N                                                    ; Edited 8-Apr-87 11:35 by jop
```

;; called only by interpreted code --- this defn relies on fact that compiler turns LOGXOR calls into sequences of opcodes

```
(SELECTQ N
     (2 (LOGXOR (ARG N 1)
                (ARG N 2)))
     (1 (LOGXOR (ARG N 1)))
     (0 (LOGXOR))
     (PROG ((R (LOGXOR (ARG N 1)
```

```
                        (ARG N 2)
                        (ARG N 3)))
              (J 4))
         LP   (COND
                 ((ILEQ J N)
                  (SETQ R (LOGXOR R (ARG N J)))
                  (SETQ J (ADD1 J))
                  (GO LP)))
              (RETURN R)))))
```

(SUB1

```
  (LAMBDA (X)                                          ; Edited  8-Apr-87 11:35 by jop
    (IDIFFERENCE X 1)))
```

(ZEROP

```
  (LAMBDA (X)                                          (* |Pavel| " 6-Oct-86 22:13")
    (COND
       ((EQ X 0)
        T)
       ((FLOATP X)
        (\\FZEROP X)))))
```

(ADD1

```
  (LAMBDA (X)                                          ; Edited  8-Apr-87 11:35 by jop
    (IPLUS X 1)))
```

(GCD

```
  (LAMBDA (N1 N2)                                      ; Edited  8-Apr-87 11:35 by jop
    ;; Greatest common divisor, using Euler's Method
    (COND
       ((EQ 0 N2)
        N1)
       ((MINUSP N2)                                    ; GCD is always positive
        (GCD (MINUS N2)
             N1))
       (T (GCD N2 (IREMAINDER N1 N2))))))
```

(IEQP

```
  (LAMBDA (X Y)                                        (* |JonL| " 1-May-84 22:23")
    (EQ 0 (IDIFFERENCE X Y))))
```

(INTEGERLENGTH

```
  (LAMBDA (X)                                          ; Edited  8-Apr-87 11:37 by jop
    (SELECTC (NTYPX X)
        (\\SMALLP (COND
                     ((ILESSP X 0)
                      (SETQ X (IDIFFERENCE 0 X))))
                  (COND
                     ((NBITS.OR.LESS X 16)
                      (COND
                         ((NBITS.OR.LESS X 8)
                          (COND
                             ((NBITS.OR.LESS X 4)
                              (COND
                                 ((NBITS.OR.LESS X 2)
                                  (COND
                                     ((NBITS.OR.LESS X 1)
                                      (COND
                                         ((EQ X 0)
                                          0)
                                         (T 1)))
                                     (T 2)))
                                 ((NBITS.OR.LESS X 3)
                                  3)
                                 (T 4)))
                             ((NBITS.OR.LESS X 6)
                              (COND
                                 ((NBITS.OR.LESS X 5)
                                  5)
                                 (T 6)))
                             ((NBITS.OR.LESS X 7)
                              7)
                             (T 8)))
                         ((NBITS.OR.LESS X 12)
                          (COND
                             ((NBITS.OR.LESS X 10)
                              (COND
                                 ((NBITS.OR.LESS X 9)
                                  9)
                                 (T 10)))
```

```
                              ((NBITS.OR.LESS X 11)
                               11)
                              (T 12)))
                          ((NBITS.OR.LESS X 14)
                           (COND
                              ((NBITS.OR.LESS X 13)
                               13)
                              (T 14)))
                          ((NBITS.OR.LESS X 15)
                           15)
                          (T 16)))
                      (T (SHOULDNT))))
        (\\FIXP (PROG ((HX (|fetch| (FIXP HINUM) |of| X)))
                      (COND
                         ((IGREATERP HX MAX.POS.HINUM)          ; So X is negative
                          ((LAMBDA (LX)
                             (COND
                                ((AND (EQ HX \\SIGNBIT)
                                      (EQ LX 0))                ; So X is EQP to the minimum FIXP integer
                                 (RETURN (CONSTANT BITS.PER.FIXP)))
                                (T (.NEGATE. HX LX))))
                           (|fetch| (FIXP LONUM) |of| X))))
                      (RETURN (COND
                                 ((EQ HX 0)
                                  ;; This bizarre case shouldn't really happen, but I wouldn't like to rule it out -- a non-normalized FIXP that
                                  ;; realy should be a SMALLP
                                  (INTEGERLENGTH (|fetch| (FIXP LONUM) |of| X)))
                                 (T (IPLUS (INTEGERLENGTH HX)
                                           BITSPERWORD))))))
        (COND
           ((TYPENAMEP X 'BIGNUM)
            (\\BIGNUM.INTEGERLENGTH X))
           (T (CL::%NOT-INTEGER-ERROR X))))))

)

;; abs, difference, greaterp, plus, lessp, and times  redefined in cmlarith.
;; quotient and minus modified to handle ratios
;; remainder remains as is

(DEFINEQ

(ABS
  (LAMBDA (X)                                                   (* FS "22-Nov-86 20:58")
                                                                (* "Replaced in CMLARITH to handle RATIOS")
    (CL:CTYPECASE X ((OR INTEGER FLOAT)
                     (COND
                        ((< X 0)
                         (- 0 X))
                        (T X)))
          (RATIO (COND
                     ((< (CL:NUMERATOR X)
                         0)
                      (%MAKE-RATIO (- 0 (CL:NUMERATOR X))
                             (CL:DENOMINATOR X)))
                     (T X)))
          (COMPLEX (%COMPLEX-ABS X)))))

(DIFFERENCE
  (LAMBDA (X Y)                                                 ; Edited  8-Apr-87 11:39 by jop
    ((OPCODES DIFFERENCE)
     X Y)))

(GREATERP
  (LAMBDA (X Y)                                                 ; Edited  8-Apr-87 11:39 by jop
    (COND
       ((AND (FIXP X)
             (FIXP Y))
        (IGREATERP X Y))
       (T (FGREATERP X Y)))))

(PLUS
  (LAMBDA N                                                     ; Edited  8-Apr-87 11:39 by jop
    ;; Microcode generally handles the case of two args both FIXPs
    (PROG (R (J 0))
      LP  (COND
             ((NEQ J N)
              (SETQ J (ADD1 J))
              (SETQ R (COND
                         ((AND (FIXP (ARG N J))
                               (NOT (FLOATP R)))
```

```
                          (IPLUS (OR R 0)
                                 (ARG N J)))
                         (T (FPLUS (OR R 0.0)
                                   (ARG N J)))))
              (GO LP)))
          (RETURN R))))
```

(QUOTIENT

```
  (LAMBDA (X Y)                                          ; Edited 12-Feb-87 14:59 by jop
                                                         (* |lmm:| 17-DEC-75 25 36)
    (COND
       ((AND (FIXP X)
             (FIXP Y))
        (IQUOTIENT X Y))
       ((OR (FLOATP X)
            (FLOATP Y))
        (FQUOTIENT X Y))
       (T (/ X Y)))))
```

(REMAINDER

```
  (LAMBDA (X Y)                                          (* |lmm:| 17-DEC-75 21 30)
    (COND
       ((AND (FIXP X)
             (FIXP Y))
        (IREMAINDER X Y))
       (T (FREMAINDER X Y)))))
```

(LESSP

```
  (LAMBDA (X Y)                                          ; Edited  8-Apr-87 11:40 by jop
    (COND
       ((AND (FIXP Y)
             (FIXP X))
        (IGREATERP Y X))
       (T (FGREATERP Y X)))))
```

(MINUS

```
  (LAMBDA (X)                                            ; Edited  1-Mar-87 18:10 by jop
    (COND
       ((FLOATP X)
        (FDIFFERENCE 0.0 X))
       (T (DIFFERENCE 0 X)))))
```

(TIMES

```
  (LAMBDA N                                              ; Edited  8-Apr-87 11:40 by jop
    (PROG (R (J 0))
      LP  (COND
             ((NEQ J N)
              (SETQ J (ADD1 J))
              (SETQ R (COND
                         ((AND (FIXP (ARG N J))
                               (NOT (FLOATP R)))
                          (ITIMES (OR R 1)
                                  (ARG N J)))
                         (T (FTIMES (OR R 1.0)
                                    (ARG N J)))))
              (GO LP)))
          (RETURN R))))

)

(DEFINEQ
```

(FMINUS

```
  (LAMBDA (X)                                            (* |lmm| " 5-MAR-80 23:12")
    (FDIFFERENCE 0.0 X)))
```

(FREMAINDER

```
  (LAMBDA (X Y)                                          (* |rrb| "24-APR-80 10:37")
    (FDIFFERENCE X (FTIMES (FLOAT (FIX (FQUOTIENT X Y)))
                           Y))))

)

(DEFINEQ
```

(RANDSET

```
  (LAMBDA (X)                                            ; Edited  8-Apr-87 11:40 by jop
    (PROG (RS RS1 RS2)
          (COND
             ((NULL X)
              (GO OUT))
```

```
            ((EQ X T)                                                ; initialize with clock
             (SETQ RS1 (CLOCK))
             (SETQ RS2 (IDATE)))
            ((AND (FIXP (CDR (LISTP X)))
                  (FIXP (CAR X)))                                    ; user supplies initialization, old-style
             (SETQ RS1 (CAR X))
             (SETQ RS2 (CDR X)))
            ((AND (EQ (LENGTH X)
                      55)
                  (EVERY X (FUNCTION FIXP)))
             (SETQ RS (MAPCAR X (FUNCTION (LAMBDA (N)
                                          (IPLUS N)))))
             (GO XX))
            (T (ERROR '"ARG NOT PREVIOUS VALUE OF RANDSET" X)))
         (PROG ((\\OVERFLOW 0))
               (DECLARE (SPECVARS \\OVERFLOW))
               (SETQ RS
                (MAPCAR '(53375 47430 1274 55702 61592 27723 11236 16824 35838 62289 11525 37822 34676 105
                                58750 27759 9988 4217 56951 30292 24550 1397 54588 54264 43300 3862 39006 11386
                                52259 1055 955 16320 19910 58470 3263 64657 1704 17373 56820 17255 51637 47962
                                26272 4464 2884 51773 39422 64835 57733 34919 5315 12110 15116 10133 10816)
                        (FUNCTION (LAMBDA (Z)
                                  (SETQ RS1 (LOGAND RS1 65535))
                                  (LOGXOR Z (SETQ RS2 (PROG1 (LOGAND (IPLUS (ITIMES RS1 19869)
                                                                            RS1)
                                                                     65535)
                                                      (SETQ RS1 RS2)))))))))
       XX  (FRPLACD (LAST RS)
                    RS)
           (SETQ RANDSTATE (CONS RS (FNTH RS 31)))
       OUT (RETURN (|for| X |in| (CAR RANDSTATE) |as| I |from| 1 |to| 55 |collect| X)))))
```

(RAND

```
(LAMBDA (LOWER UPPER)                                                ; Edited 3-Sep-87 19:03 by jds
```

;; This function implements the XRAND subroutine described in Stanford memo STAN-CS-77-601, Analysis of Additive Random Number Generators, by John F. Reiser, on p 28.0 Rather than storing the X values in an array and computing indexes I and J, however, I have elected to retain state in a circular list of 51 elements. RANDSTATE is (CONS X (NTH X 31)); each time RAND is called, both CAR and CDR of RANDSTATE are CDR'ed to effectively increment the index. In addition, the numbers are stored as 16 bit binary fractions (i.e. the decimal point is on the left of the 16-bit quantity)

```
(PROG (I J)
      (OR (LISTP RANDSTATE)
          (PROGN (RANDSET T)
                 RANDSTATE))
      (SETQ I (CDAR RANDSTATE))
      (SETQ J (CDDR RANDSTATE))
      (RPLNODE RANDSTATE I J)
      (RPLACA I (LOGAND (IDIFFERENCE (CAR I)
                                     (CAR J))
                        MAX.SMALLP)))
(COND
   ((NOT UPPER)
    (COND
       ((NULL LOWER)                                                 ; both UPPER and LOWER nil.  Return number (0 (\,
                                                                     ; MAX.SMALLP)) --- not documented
        (CAAR RANDSTATE))
       ((ZEROP LOWER)                                                ; (RAND 0) = 0
        0)
       ((FIXP LOWER)                                                 ; (RAND n) = (RAND 0 n-1)
        (IREMAINDER (CAAR RANDSTATE)
                    LOWER))
       (T                                                            ; (RAND N) N floating.  Return (RAND 0 N)
          (FTIMES LOWER (FQUOTIENT (CAAR RANDSTATE)
                                   (CONSTANT (FLOAT (ADD1 MAX.SMALLP))))))))
   ((AND (FIXP LOWER)
         (FIXP UPPER))
    (OR (IGREATERP UPPER LOWER)
        (|swap| UPPER LOWER))
    (SETQ UPPER (IDIFFERENCE UPPER LOWER))
    (COND
       ((IGREATERP UPPER MAX.SMALLP)
        (IPLUS (IMOD (\\MAKENUMBER (CAAR RANDSTATE)
                                   (CADAR RANDSTATE))
                     (ADD1 UPPER))
               LOWER))
       (T (IPLUS (IREMAINDER (CAAR RANDSTATE)
                             (ADD1 UPPER))
                 LOWER))))
   (T (FPLUS (FTIMES (FDIFFERENCE UPPER LOWER)
                     (FQUOTIENT (CAAR RANDSTATE)
                                (CONSTANT (FLOAT (ADD1 MAX.SMALLP)))))
             LOWER))))
```

(EXPT

```
(LAMBDA (A N)                                                        ; Edited 8-Apr-87 11:41 by jop
```

```
    (COND
       ((FIXP N)
        (COND
           ((FIXP A)
            (COND
               ((NOT (IGREATERP N 0))
                (COND
                   ((EQ 0 N)
                    1)
                   (T (FEXPT A N))))
               ((EQ 0 A)
                0)
               (T ;; Integer EXPonentiation -- works by clever bit-dissection method
                  (PROG ((V 1))
                    LP  (COND
                           ((ODDP N)
                            (SETQ V (TIMES A V))))
                        (COND
                           ((EQ 0 (SETQ N (RSH N 1)))
                            (RETURN V)))
                        (SETQ A (TIMES A A))
                        (GO LP)))))
           ((FEQP 0.0 (SETQ A (FLOAT A)))
            (COND
               ((EQ 0 N)
                1.0)
               (T 0.0)))
           (T ;; Real EXPonentiation -- works by clever bit-dissection method
              (PROG ((V 1.0))
                    (COND
                       ((ILESSP N 0)
                        (SETQ A (FQUOTIENT 1.0 A))
                        (SETQ N (IMINUS N))))
                LP  (COND
                       ((ODDP N)
                        (SETQ V (TIMES A V))))
                    (COND
                       ((EQ 0 (SETQ N (LRSH N 1)))
                        (RETURN V)))
                    (SETQ A (TIMES A A))
                    (GO LP)))))
       (T (FEXPT A N)))))

)

(DECLARE\: DONTEVAL@LOAD DOCOPY

(RPAQQ RANDSTATE NIL)

(RPAQQ \\TOL 9.9999925E-6)
)

(DECLARE\: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS RANDSTATE \\TOL)
)

(DEFINEQ

(|PutUnboxed|
  (LAMBDA (PTR NUM)                                    (* |JonL| "25-JUL-83 02:29")
    (\\PUTFIXP PTR NUM)))

(\\PUTFIXP
  (LAMBDA (PTR NUM)                                    (* |lmm| "11-DEC-80 15:10")
    (PROG (HI LO)
          (.UNBOX. NUM HI LO)
          (|replace| (FIXP HINUM) |of| PTR |with| HI)
          (|replace| (FIXP LONUM) |of| PTR |with| LO)
          (RETURN NUM))))

(\\PUTSWAPPEDFIXP
  (LAMBDA (PTR NUM)                                    ; Edited  8-Apr-87 11:41 by jop
    ;; store in MESA order rather than LISP order
    (PROG (HI LO)
          (.UNBOX. NUM HI LO)
          (|replace| (FIXP LONUM) |of| PTR |with| HI)
          (|replace| (FIXP HINUM) |of| PTR |with| LO)
          (RETURN NUM))))

(\\HINUM
```

```
  (LAMBDA (NUM)                                                    (* |lmm| "12-APR-81 22:01")
    (PROG (HI LO)
          (.UNBOX. NUM HI LO)
          (RETURN HI))))
```

(**LONUM**

```
  (LAMBDA (NUM)                                                    (* |lmm| "12-APR-81 22:02")
    (PROG (HI LO)
          (.UNBOX. NUM HI LO)
          (RETURN LO))))

)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE\: DONTCOPY

(DECLARE\: EVAL@COMPILE

(PUTPROPS |PutUnboxed| DMACRO (= . \\PUTFIXP))
)
)

;; END EXPORTED DEFINITIONS

(DECLARE\: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA MIN MAX IMIN IMAX FMIN FMAX ODDP TIMES PLUS LOGXOR LOGOR LOGAND ITIMES IPLUS)
)

;; ODDP redefined in cmlarith

(DEFINEQ
```

(**POWEROFTWOP**

```
  (LAMBDA (X)
    (DECLARE (LOCALVARS . T))                                      ; Edited  8-Apr-87 11:42 by jop
    ;; Non-NIL iff arg is some power of 2
    (|if| (AND (EQ (SYSTEMTYPE)
                   'D)
               (NOT (SMALLP X)))
          |then| (AND (FIXP X)
                      (IGREATERP X 0)
                      (|if| (EQ (LOGAND X (CONSTANT (SUB1 (EXPT 2 16))))
                                0)
                          |then| (POWEROFTWOP (RSH X 16))
                        |else| (AND (EQ (RSH X 16)
                                        0)
                                    (.2^NP. (LOGAND X (SUB1 (EXPT 2 16)))))))
      |else| (|if| (IGREATERP X 0)
                 |then| (.2^NP. X)))))
```

(**IMOD**

```
  (LAMBDA (X N)                                                    (* |lmm| "20-OCT-82 15:07")
    (COND
       ((IGEQ (SETQ X (IREMAINDER X N))
              0)
        X)
       (T (IPLUS N X)))))
```

(**ODDP**

```
  (CL:LAMBDA (CL:NUMBER &OPTIONAL (MODULUS 2))                     (* |lmm| "22-May-86 17:26")
         (NOT (ZEROP (CL:MOD CL:NUMBER MODULUS)))))

)

(DECLARE\: DONTCOPY

(DECLARE\: EVAL@COMPILE

(PUTPROPS .2^NP. MACRO (OPENLAMBDA (X)
                         (EQ (LOGAND X (SUB1 X))
                             0)))
)
)

;; MIN and MAX
```

```
(DEFINEQ

(FLESSP
  (LAMBDA (X Y)
    (FGREATERP Y X)))

(FMAX
  (LAMBDA K                                                    (* |bvm:| "14-Feb-85 23:48")
    (COND
       ((EQ K 0)
        MIN.FLOAT)
       (T (PROG ((J 1)
                 (X (FLOAT (ARG K 1)))
                 Y)
                (OR (NUMBERP X)
                    (ERRORX (LIST 10 X)))
            LP  (COND
                   ((EQ J K)
                    (RETURN X)))
                (ADD1VAR J)
                (COND
                   ((FGREATERP (SETQ Y (FLOAT (ARG K J)))
                               X)
                    (SETQ X Y)))
                (GO LP))))))

(FMIN
  (LAMBDA K                                                    (* |bvm:| "14-Feb-85 23:49")
    (COND
       ((EQ K 0)
        MAX.FLOAT)
       (T (PROG ((J 1)
                 (X (FLOAT (ARG K 1)))
                 Y)
                (OR (NUMBERP X)
                    (ERRORX (LIST 10 X)))
            LP  (COND
                   ((EQ J K)
                    (RETURN X)))
                (ADD1VAR J)
                (COND
                   ((FGREATERP X (SETQ Y (FLOAT (ARG K J))))
                    (SETQ X Y)))
                (GO LP))))))

(GEQ
  (LAMBDA (X Y)
    (NOT (LESSP X Y))))

(IGEQ
  (LAMBDA (X Y)
    (NOT (ILESSP X Y))))

(ILEQ
  (LAMBDA (X Y)
    (NOT (IGREATERP X Y))))

(IMAX
  (LAMBDA K                                                    (* |bvm:| "14-Feb-85 23:49")
    (COND
       ((EQ K 0)
        MIN.INTEGER)
       (T (PROG ((J 1)
                 (X (ARG K 1)))
            LP  (COND
                   ((EQ J K)
                    (RETURN X)))
                (ADD1VAR J)
                (COND
                   ((ILESSP X (ARG K J))
                    (SETQ X (ARG K J))))
                (GO LP))))))

(IMIN
  (LAMBDA K                                                    (* |bvm:| "14-Feb-85 23:49")
    (COND
       ((EQ K 0)
        MAX.INTEGER)
       (T (PROG ((J 1)
```

```
                (X (ARG K 1)))
           LP   (COND
                   ((EQ J K)
                    (RETURN X)))
                (ADD1VAR J)
                (COND
                   ((IGREATERP X (ARG K J))
                    (SETQ X (ARG K J))))
                (GO LP))))))
```

(**LEQ**

```
  (LAMBDA (X Y)
    (NOT (GREATERP X Y))))
```

(**MAX**

```
  (LAMBDA K                                                    (* |lmm| "12-Apr-85 08:42")
    (COND
       ((EQ K 0)
        MIN.INTEGER)
       (T (PROG ((J 1)
                 (X (ARG K 1))
                 Y)
                (OR (NUMBERP X)
                    (ERRORX (LIST 10 X)))
           LP   (COND
                   ((EQ J K)
                    (RETURN X)))
                (ADD1VAR J)
                (COND
                   ((GREATERP (SETQ Y (ARG K J))
                              X)
                    (SETQ X Y)))
                (GO LP))))))
```

(**MIN**

```
  (LAMBDA K                                                    (* |lmm| "12-Apr-85 08:42")
    (COND
       ((EQ K 0)
        MAX.INTEGER)
       (T (PROG ((J 1)
                 (X (ARG K 1))
                 Y)
                (OR (NUMBERP X)
                    (ERRORX (LIST 10 X)))
           LP   (COND
                   ((EQ J K)
                    (RETURN X)))
                (ADD1VAR J)
                (COND
                   ((GREATERP X (SETQ Y (ARG K J)))
                    (SETQ X Y)))
                (GO LP))))))

)

(DECLARE\: EVAL@COMPILE

(ADDTOVAR GLOBALVARS MAX.INTEGER MIN.INTEGER MAX.FLOAT MIN.FLOAT)
)

(DECLARE\: DONTCOPY DOEVAL@COMPILE DONTEVAL@LOAD

(DECLARE\: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)
)

(PUTPROPS LLARITH COPYRIGHT ("Venue & Xerox Corporation" T 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991
                                    1993))
```

FUNCTION INDEX

MACRO INDEX

CONSTANT INDEX

VARIABLE INDEX

RECORD INDEX