

File created: 18-Oct-93 15:40:08 {Pele:mv:envos}<LispCore>Sources>CLTL2>FASDUMP.;2

previous date: 3-Sep-91 17:55:43 {Pele:mv:envos}<LispCore>Sources>CLTL2>FASDUMP.;1

Read Table: XCL

Package: FASL

Format: XCCS

; Copyright (c) 1986, 1987, 1988, 1990, 1991, 1993 by Venue & Xerox Corporation. All rights reserved.

(IL:RPAQQ IL:FASDUMPCOMS

(

;;; FASL Dumper.

```
(IL:DECLARE\ IL:EVAL@COMPILE IL:EVAL@LOAD IL:DONTCOPY (IL:FILES (IL:LOADCOMP)
                                                    IL:FASLOAD))
(IL:STRUCTURES HANDLE)
(IL:VARIABLES DUMMY-HANDLE)
(IL:VARIABLES +SMALLEST-FOUR-BYTE-INTEGERS+ +LARGEST-FOUR-BYTE-INTEGERS+)
(IL:VARIABLES *GATHER-DUMPER-STATS* *TABLE-ATTEMPTS* *TABLE-HITS*)
(IL:FUNCTIONS RESET-DUMPER-STATS)
(IL:FUNCTIONS DOTTED-LIST-LENGTH STATE-CASE FAT-STRING-P REMEMBER ELEMENTS-IDENTICAL-P END-BLOCK
              END-TEXT WRITE-OP LOOKUP-VALUE SAVE-VALUE)
(IL:FUNCTIONS DUMP-VALUE-FETCH DUMP-CHARACTER DUMP-SYMBOL DUMP-LIST DUMP-SIMPLE-VECTOR
              DUMP-ARRAY-DESCRIPTOR DUMP-BIT-ARRAY DUMP-GENERAL-ARRAY DUMP-ARRAY WRITE-INTEGERS-BYTES
              INTEGER-BYTE-LIST DUMP-RATIONAL DUMP-COMPLEX DUMP-INTEGERS DUMP-PACKAGE DUMP-DCODE DUMP-STRING
              DUMP-FLOAT32 DUMP-STRUCTURE DUMP-BITMAP)
(IL:FUNCTIONS OPEN-FASL-HANDLE WITH-OPEN-HANDLE BEGIN-TEXT BEGIN-BLOCK VALUE-DUMPABLE-P DUMP-VALUE
              DUMP-FUNCTION-DEF DUMP-FUNCALL DUMP-EVAL CLOSE-FASL-HANDLE)
;; Arrange for the correct compiler and makefile environment
(IL:PROP (IL:FILETYPE IL:MAKEFILE-ENVIRONMENT)
         IL:FASDUMP))
```

;;; FASL Dumper.

```
(IL:DECLARE\ IL:EVAL@COMPILE IL:EVAL@LOAD IL:DONTCOPY
(IL:FILESLOAD (IL:LOADCOMP)
              IL:FASLOAD)
)
(DEFSTRUCT (HANDLE (:CONSTRUCTOR MAKE-HANDLE))
  STREAM
  (STATE :BLOCK-END)
  (LAST-INDEX 0)
  (HASH (MAKE-HASH-TABLE :TEST #'EQ)))
(DEFCONSTANT DUMMY-HANDLE (MAKE-HANDLE :STREAM (OPEN "{null}" :DIRECTION :OUTPUT)
                                       :STATE :BLOCK :HASH NIL))
(DEFCONSTANT +SMALLEST-FOUR-BYTE-INTEGERS+ (- (EXPT 2 31)))
(DEFCONSTANT +LARGEST-FOUR-BYTE-INTEGERS+ (1- (EXPT 2 31)))
(DEFVAR *GATHER-DUMPER-STATS* NIL)
(DEFVAR *TABLE-ATTEMPTS* 0
  "Number of table lookups by the FASL dumper.")
(DEFVAR *TABLE-HITS* 0
  "Number of successful table lookups by the FASL dumper.")
(DEFUN RESET-DUMPER-STATS ()
  (SETQ *TABLE-ATTEMPTS* 0 *TABLE-HITS* 0))
(DEFUN DOTTED-LIST-LENGTH (X)
  (DO ((N 0 (+ N 2))
      (FAST X (CDDR FAST))
      (SLOW X (CDR SLOW)))
      (NIL)
      (COND
        ((NULL FAST)
         (RETURN N))
        ((ATOM FAST)
```

```

    (RETURN (VALUES N T)))
  ((NULL (CDR FAST))
   (RETURN (1+ N)))
  ((ATOM (CDR FAST))
   (RETURN (VALUES (1+ N)
                    T))))
  ((AND (EQ FAST SLOW)
        (> N 0))
   (RETURN NIL))))))

```

```

(DEFMACRO STATE-CASE (&REST CLAUSES)
  `(ECASE (HANDLE-STATE HANDLE)
    (IL:\\, @ CLAUSES)))

```

```

(DEFUN FAT-STRING-P (STRING)
  (COND
    ((IL:STRINGP STRING)
     (EQ (IL:FETCH (IL:STRINGP IL:TYP) IL:OF STRING)
          IL:\\ST.POS16))
    (T (IL:%FAT-STRING-ARRAY-P STRING))))

```

```

(DEFMACRO REMEMBER (VALUE &BODY BODY)
  `(LET (($REMEMBER-VAL$ ,VALUE))
    (WHEN REMEMBER
      (WRITE-OP HANDLE 'FASL-TABLE-STORE))
    ,@BODY
    (WHEN REMEMBER (SAVE-VALUE HANDLE $REMEMBER-VAL$))))

```

```

(DEFUN ELEMENTS-IDENTICAL-P (ARRAY)
  (LET* ((SEQ (IL:%FLATTEN-ARRAY ARRAY))
         (TESTELT (AREF SEQ 0)))
    (EVERY #'(LAMBDA (X)
              (EQL X TESTELT))
           SEQ)))

```

```

(DEFUN END-BLOCK (HANDLE)
  (STATE-CASE (:BLOCK (WHEN CHECK-TABLE-SIZE
                        (WRITE-OP HANDLE 'FASL-VERIFY-TABLE-SIZE)
                        (DUMP-VALUE HANDLE (HANDLE-LAST-INDEX HANDLE)
                                           NIL))
              (IL:BOUT (HANDLE-STREAM HANDLE)
                        END-MARK)
              (SETF (HANDLE-LAST-INDEX HANDLE)
                    0)
              (SETF (HANDLE-HASH HANDLE)
                    (MAKE-HASH-TABLE :TEST #'EQ))
              (SETF (HANDLE-STATE HANDLE)
                    :BLOCK-END))))

```

```

(DEFUN END-TEXT (HANDLE)
  (STATE-CASE (:TEXT (IL:BOUT (HANDLE-STREAM HANDLE)
                              END-MARK)
              (SETF (HANDLE-STATE HANDLE)
                    :BLOCK))))

```

```

(DEFUN WRITE-OP (HANDLE OPNAME)
  (STATE-CASE (:BLOCK (LET ((STREAM (HANDLE-STREAM HANDLE))
                           (OPSEQ (OPCODE-SEQUENCE OPNAME)))
                        (IF (NULL OPSEQ)
                            (ERROR 'UNIMPLEMENTED-OPCODE :OPNAME OPNAME)
                            (DOLIST (OP OPSEQ)
                                      (IL:BOUT STREAM OP)))))))

```

```

(DEFUN LOOKUP-VALUE (HANDLE VALUE)
  (LET ((HASH-TABLE (HANDLE-HASH HANDLE))
        (AND HASH-TABLE (IL:GETHASH VALUE HASH-TABLE))))

```

```

(DEFUN SAVE-VALUE (HANDLE VALUE)
  (LET ((HASH-TABLE (HANDLE-HASH HANDLE))
        (UNLESS (NULL HASH-TABLE)
                 (SETF (IL:GETHASH VALUE HASH-TABLE)
                       (HANDLE-LAST-INDEX HANDLE))
                 (INCF (HANDLE-LAST-INDEX HANDLE))))))

```

```

(DEFUN DUMP-VALUE-FETCH (HANDLE INDEX)
  (WRITE-OP HANDLE 'FASL-TABLE-FETCH)
  (DUMP-VALUE HANDLE INDEX NIL))

```

```

(DEFUN DUMP-CHARACTER (HANDLE CHAR REMEMBER)
  (DECLARE (IGNORE REMEMBER))
  ;; Characters don't get remembered.
  (LET ((CODE (CHAR-CODE CHAR))
        (STREAM (HANDLE-STREAM HANDLE)))
    (WRITE-OP HANDLE 'FASL-CHARACTER)
    (IF (< CODE 256)
        (IL:BOUT STREAM CODE)
        (PROGN (IL:BOUT STREAM 255)
                (IL:BOUT16 STREAM CODE))))))

```

```

(DEFUN DUMP-SYMBOL (HANDLE SYMBOL REMEMBER)
  ;; No point in remembering the pname because SYMBOL-NAME always gives you a new one.
  (LET* ((PNAME (SYMBOL-NAME SYMBOL))
         (PACKAGE (SYMBOL-PACKAGE SYMBOL))
         (PKG-NAME (AND PACKAGE (PACKAGE-NAME PACKAGE))))
    (REMEMBER SYMBOL (COND
                      ((KEYWORDP SYMBOL)
                       (WRITE-OP HANDLE 'FASL-KEYWORD-SYMBOL)
                       (DUMP-VALUE HANDLE PNAME NIL))
                      (EQUAL PKG-NAME "LISP")
                       (WRITE-OP HANDLE 'FASL-LISP-SYMBOL)
                       (DUMP-VALUE HANDLE PNAME NIL))
                      (EQUAL PKG-NAME "INTERLISP")
                       (WRITE-OP HANDLE 'FASL-INTERLISP-SYMBOL)
                       (DUMP-VALUE HANDLE PNAME NIL))
                      (T (WRITE-OP HANDLE 'FASL-SYMBOL-IN-PACKAGE)
                         (DUMP-VALUE HANDLE PNAME NIL)
                         (DUMP-VALUE HANDLE PACKAGE REMEMBER))))))

```

```

(DEFUN DUMP-LIST (HANDLE LIST REMEMBER)
  (MULTIPLE-VALUE-BIND (LENGTH DOTTED)
    (DOTTED-LIST-LENGTH LIST)
    (UNLESS LENGTH
      (ERROR 'OBJECT-NOT-DUMPABLE :OBJECT LIST))
    (REMEMBER LIST (WRITE-OP HANDLE (IF DOTTED
                                         'FASL-LIST*
                                         'FASL-LIST))
                  (DUMP-VALUE HANDLE (IF DOTTED
                                         (1+ LENGTH)
                                         LENGTH)
                                     NIL)
                  (DOTIMES (I LENGTH)
                    (DUMP-VALUE HANDLE (CAR LIST))
                    (POP LIST))
                  (WHEN DOTTED (DUMP-VALUE HANDLE LIST NIL))))))

```

```

(DEFUN DUMP-SIMPLE-VECTOR (HANDLE VECTOR REMEMBER)
  (LET ((LENGTH (LENGTH VECTOR)))
    (REMEMBER VECTOR (WRITE-OP HANDLE 'FASL-VECTOR)
                (DUMP-VALUE HANDLE LENGTH REMEMBER)
                (DOTIMES (I LENGTH)
                  (DUMP-VALUE HANDLE (SVREF VECTOR I)
                                      REMEMBER))))))

```

```

(DEFUN DUMP-ARRAY-DESCRIPTOR (HANDLE ARRAY REMEMBER &KEY (INITIAL-ELEMENT NIL USE-SINGLE-ELT))
  (REMEMBER ARRAY (WRITE-OP HANDLE 'FASL-CREATE-ARRAY)
                (DUMP-VALUE HANDLE (IF (EQL (ARRAY-RANK ARRAY)
                                             1)
                                         (CAR (ARRAY-DIMENSIONS ARRAY))
                                         (ARRAY-DIMENSIONS ARRAY))
                                      REMEMBER)
                (DUMP-VALUE HANDLE `(:ELEMENT-TYPE , (ARRAY-ELEMENT-TYPE ARRAY)
                                       :ADJUSTABLE
                                       , (ADJUSTABLE-ARRAY-P ARRAY)
                                       , @ (WHEN (ARRAY-HAS-FILL-POINTER-P ARRAY)
                                               `(:FILL-POINTER , (FILL-POINTER ARRAY)))
                                       , @ (WHEN USE-SINGLE-ELT
                                               `(:INITIAL-ELEMENT , INITIAL-ELEMENT)))
                                      REMEMBER)))

```

```

(DEFUN DUMP-BIT-ARRAY (HANDLE ARRAY REMEMBER)
  (LET ((NBITS (ARRAY-TOTAL-SIZE ARRAY))
        (UNLESS (ZEROP (IL:%ARRAY-OFFSET ARRAY))
          (ERROR 'OBJECT-NOT-DUMPABLE :OBJECT ARRAY))
        (REMEMBER ARRAY (WRITE-OP HANDLE 'FASL-INITIALIZE-BIT-ARRAY)
                          (DUMP-ARRAY-DESCRIPTOR HANDLE ARRAY REMEMBER)
                          (DUMP-VALUE HANDLE NBITS REMEMBER)

```

```
(IL:\BOUTS (HANDLE-STREAM HANDLE)
  (IL:%ARRAY-BASE ARRAY)
  0
  (CEILING NBITS 8))))
```

(DEFUN **DUMP-GENERAL-ARRAY** (HANDLE ARRAY REMEMBER)

;; Arrays don't get remembered. Displacement information is lost.

```
(LET* ((NELTS (ARRAY-TOTAL-SIZE ARRAY))
      (ELT-TYPE (ARRAY-ELEMENT-TYPE ARRAY)))
  (WRITE-OP HANDLE 'FASL-INITIALIZE-ARRAY)
  (DUMP-ARRAY-DESCRIPTOR HANDLE ARRAY NIL)
  (DUMP-VALUE HANDLE NELTS NIL)
  (LET ((INDIRECT (MAKE-ARRAY NELTS :DISPLACED-TO ARRAY :ELEMENT-TYPE ELT-TYPE)))
    (DOTIMES (I NELTS)
      (DUMP-VALUE HANDLE (AREF INDIRECT I)
        NIL))))))
```

(DEFUN **DUMP-ARRAY** (HANDLE ARRAY REMEMBER)

```
(COND
  ((XCL:DISPLACED-ARRAY-P ARRAY)
   (ERROR 'OBJECT-NOT-DUMPABLE :OBJECT ARRAY))
  ((ADJUSTABLE-ARRAY-P ARRAY)
   (DUMP-GENERAL-ARRAY HANDLE ARRAY REMEMBER))
  ((TYPEP ARRAY '(ARRAY BIT))
   (DUMP-BIT-ARRAY HANDLE ARRAY REMEMBER))
  ((TYPEP ARRAY 'VECTOR)
   (DUMP-SIMPLE-VECTOR HANDLE ARRAY REMEMBER))
  (T (DUMP-GENERAL-ARRAY HANDLE ARRAY REMEMBER))))
```

(DEFUN **WRITE-INTEGERS-BYTES** (HANDLE NBYTES VALUE)

```
(LET ((STREAM (HANDLE-STREAM HANDLE)))
  (DOLIST (BYTE (INTEGER-BYTE-LIST VALUE NBYTES))
    (IL:BOU STREAM BYTE))))
```

(DEFUN **INTEGER-BYTE-LIST** (VALUE NBYTES)

```
(DO ((COUNT 0 (1+ COUNT))
     (RESULT NIL)
     (N VALUE)
     (BYTE)
     ((>= COUNT NBYTES)
      RESULT)
     (MULTIPLE-VALUE-SETQ (N BYTE)
       (FLOOR N 256))
     (PUSH BYTE RESULT)))
```

(DEFUN **DUMP-RATIONAL** (HANDLE VALUE REMEMBER)

```
(DECLARE (IGNORE REMEMBER))
(WRITE-OP HANDLE 'FASL-RATIO)
(DUMP-VALUE HANDLE (NUMERATOR VALUE)
  NIL)
(DUMP-VALUE HANDLE (DENOMINATOR VALUE)
  NIL))
```

(DEFUN **DUMP-COMPLEX** (HANDLE VALUE REMEMBER)

```
(DECLARE (IGNORE REMEMBER))
(WRITE-OP HANDLE 'FASL-COMPLEX)
(DUMP-VALUE HANDLE (REALPART VALUE)
  NIL)
(DUMP-VALUE HANDLE (IMAGPART VALUE)
  NIL))
```

(DEFUN **DUMP-INTEGERS** (HANDLE VALUE REMEMBER)

```
(DECLARE (IGNORE REMEMBER))
(COND
  ((AND (<= 0 VALUE)
        (< VALUE 128))
   (IL:BOU (HANDLE-STREAM HANDLE)
     VALUE))
  ((AND (<= +SMALLEST-FOUR-BYTE-INTEGERS+ VALUE +LARGEST-FOUR-BYTE-INTEGERS+)
        (WRITE-OP HANDLE 'FASL-INTEGERS)
        (WRITE-INTEGERS-BYTES HANDLE 4 VALUE))
   (T (WRITE-OP HANDLE 'FASL-LARGE-INTEGERS)
      (LET* ((MINBITS (1+ (INTEGER-LENGTH VALUE)))
            (NBYTES (CEILING MINBITS 8)))
        ;; According to the book, MINBITS gives the minimum field width for this number in 2's complement representation.
        (DUMP-VALUE HANDLE NBYTES NIL)
        (WRITE-INTEGERS-BYTES HANDLE NBYTES VALUE))))))
```

```
(DEFUN DUMP-PACKAGE (HANDLE PACKAGE REMEMBER)
  (REMEMBER PACKAGE (WRITE-OP HANDLE 'FASL-FIND-PACKAGE)
    (DUMP-VALUE HANDLE (PACKAGE-NAME PACKAGE)
      REMEMBER)))
```

```
(DEFUN DUMP-DCODE (HANDLE DCODE REMEMBER)
  (LET ((STREAM (HANDLE-STREAM HANDLE))
        (MACROLET ((DUMP-SEQ (SEQ DUMP-LENGTH &REST STUFF)
                          `(LET ((SEQ ,SEQ)
                                ,@(AND DUMP-LENGTH '((DUMP-VALUE HANDLE (LENGTH SEQ)
                                                                REMEMBER))))
                          (IF (LISTP SEQ)
                              (DOLIST (ELT SEQ)
                                ,@STUFF)
                              (DOTIMES (INDEX (LENGTH SEQ))
                                (LET ((ELT (AREF SEQ INDEX))
                                      ,@STUFF)))))))
```

;; If group fixups are necessary, wrap the whole thing in a FASL-LOCAL-FN-FIXUPS.

```
(UNLESS (NULL (D-ASSEM::DCODE-LOCAL-FN-FIXUPS DCODE))
  (WRITE-OP HANDLE 'FASL-LOCAL-FN-FIXUPS))
(REMEMBER DCODE ; So that it turns up as a value fetch in the local function fixups
; below.
```

```
(WRITE-OP HANDLE 'FASL-DCODE)
(DUMP-VALUE HANDLE (LENGTH (D-ASSEM::DCODE-NAME-TABLE DCODE))
  REMEMBER)
(LET* ((CODE-ARRAY (D-ASSEM::DCODE-CODE-ARRAY DCODE))
       (NBYTES (LENGTH CODE-ARRAY)))
  (DUMP-VALUE HANDLE NBYTES REMEMBER)
  (DOTIMES (I NBYTES)
    (IL:BOUT STREAM (AREF CODE-ARRAY I))))
(DUMP-SEQ (D-ASSEM::DCODE-NAME-TABLE DCODE)
  NIL
  (IL:BOUT STREAM (FIRST ELT))
  (DUMP-VALUE HANDLE (SECOND ELT)
    REMEMBER)
  (DUMP-VALUE HANDLE (THIRD ELT)
    REMEMBER))
(DUMP-VALUE HANDLE (D-ASSEM::DCODE-FRAME-NAME DCODE)
  REMEMBER)
(IL:BOUT STREAM (D-ASSEM::DCODE-NLOCALS DCODE))
(IL:BOUT STREAM (D-ASSEM::DCODE-NFREEVARS DCODE))
(IL:BOUT STREAM (D-ASSEM::DCODE-ARG-TYPE DCODE))
(DUMP-VALUE HANDLE (D-ASSEM::DCODE-NUM-ARGS DCODE)
  REMEMBER)
(DUMP-VALUE HANDLE (D-ASSEM::DCODE-CLOSURE-P DCODE)
  REMEMBER)
(DUMP-VALUE HANDLE (D-ASSEM::DCODE-DEBUGGING-INFO DCODE)
  REMEMBER)
(MACROLET ((DUMP-FIXUPS (LIST)
              `(DUMP-SEQ ,LIST T (DUMP-VALUE HANDLE (FIRST ELT))
                (DUMP-VALUE HANDLE (SECOND ELT))))
  (DUMP-FIXUPS (D-ASSEM::DCODE-FN-FIXUPS DCODE))
  (DUMP-FIXUPS (D-ASSEM::DCODE-SYM-FIXUPS DCODE))
  (DUMP-FIXUPS (D-ASSEM::DCODE-LIT-FIXUPS DCODE))
  (DUMP-FIXUPS (D-ASSEM::DCODE-TYPE-FIXUPS DCODE))))
```

;; Now do the actual group fixups if needed.

```
(UNLESS (NULL (D-ASSEM::DCODE-LOCAL-FN-FIXUPS DCODE))
  (DUMP-SEQ (D-ASSEM::DCODE-LOCAL-FN-FIXUPS DCODE D-ASSEM:DCODE)
    T
    (DUMP-VALUE HANDLE (FIRST ELT))
    (DUMP-VALUE HANDLE (SECOND ELT))
    (DUMP-VALUE HANDLE (THIRD ELT))))
NIL))
```

```
(DEFUN DUMP-STRING (HANDLE STRING REMEMBER)
  (REMEMBER STRING (LET ((STREAM (HANDLE-STREAM HANDLE))
                        (NCHARS (LENGTH STRING)))
    (COND
      ((FAT-STRING-P STRING)
        (WRITE-OP HANDLE 'FASL-FAT-STRING)
        (DUMP-VALUE HANDLE NCHARS REMEMBER)
        (DO ((I 0 (1+ I))
            (CSET 0))
          ((>= I NCHARS)
            ; Always run-encode
            (LET* ((CHAR (CHAR-CODE (CHAR STRING I)))
                  (NEW-CSET (IL:LRSH CHAR 8)))
              (UNLESS (EQL NEW-CSET CSET)
                (SETQ CSET NEW-CSET)
                (IL:BOUT STREAM 255)
                (IL:BOUT STREAM CSET))
              (IL:BOUT STREAM (LOGAND CHAR 255))))))
      (T (WRITE-OP HANDLE 'FASL-THIN-STRING)
```

```

      (DUMP-VALUE HANDLE NCHARS REMEMBER)
;; should use \bouts
      (DOTIMES (I NCHARS)
        (IL:BOUT STREAM (CHAR-CODE (CHAR STRING I))))))

```

```

(DEFUN DUMP-FLOAT32 (HANDLE NUMBER REMEMBER) ; Floats don't get remembered
  (WRITE-OP HANDLE 'FASL-FLOAT32)
  (IL:\BOOTS (HANDLE-STREAM HANDLE)
    NUMBER 0 4))

```

```

(DEFUN DUMP-STRUCTURE (HANDLE VALUE REMEMBER)
  (LET ((TYPE (IL:TYPE NAME VALUE)))
    (REMEMBER VALUE (WRITE-OP HANDLE 'FASL-STRUCTURE)
      (DUMP-VALUE HANDLE TYPE T)
      (DUMP-VALUE HANDLE (IL:FOR FIELD IL:IN (CL::STRUCTURE-SLOT-NAMES TYPE T) IL:AS DESCRIPTOR
        IL:IN (IL:GETDESCRIPTORS TYPE) IL:JOIN (LIST FIELD (IL:FETCHFIELD DESCRIPTOR
          VALUE)))
        T))))

```

```

(DEFUN DUMP-BITMAP (HANDLE VALUE REMEMBER)
  (LET ((WIDTH (IL:BITMAPWIDTH VALUE))
    (HEIGHT (IL:BITMAPHEIGHT VALUE))
    (BITS-PER-PIXEL (IL:BITSPERPIXEL VALUE))
    (BASE (IL:FETCH (IL:BITMAP IL:BITMAPBASE) IL:OF VALUE))
    (STREAM (HANDLE-STREAM HANDLE)))
    (REMEMBER VALUE ; Remember the bitmap itself.
      (WRITE-OP HANDLE 'FASL-BITMAP16)
      (DUMP-VALUE HANDLE WIDTH)
      (DUMP-VALUE HANDLE HEIGHT)
      (DUMP-VALUE HANDLE BITS-PER-PIXEL)
      (IL:\BOOTS STREAM BASE 0 (* 2 HEIGHT (CEILING (* WIDTH BITS-PER-PIXEL)
        16))))))

```

```

(DEFUN OPEN-FASL-HANDLE (NAME &REST OPEN-OPTIONS)
  (LET ((STREAM (APPLY #'OPEN NAME :DIRECTION :OUTPUT :ELEMENT-TYPE '(UNSIGNED-BYTE 8)
    :IF-EXISTS :NEW-VERSION OPEN-OPTIONS)))
    ;; A newly opened stream has fileptr = 0..
    (IL:BOUT STREAM SIGNATURE)
    (IL:BOUT16 STREAM CURRENT-VERSION)
    (MAKE-HANDLE :STREAM STREAM)))

```

```

(DEFMACRO WITH-OPEN-HANDLE ((HANDLE FILENAME &REST OPEN-OPTIONS)
  &BODY
  (BODY DECLS))
  (LET ((ABORT (IL:GENSYM "FASL:WITH-OPEN-FASL-HANDLE")))
    `(LET ((HANDLE (OPEN-FASL-HANDLE ,FILENAME ,@OPEN-OPTIONS))
      (ABORT T))
      ,@DECLS
      (UNWIND-PROTECT
        (MULTIPLE-VALUE-PROG1 (PROGN ,@BODY)
          (SETQ ,ABORT NIL))
        (WHEN ,HANDLE
          (CLOSE-FASL-HANDLE ,HANDLE :ABORT ,ABORT))))))

```

```

(DEFUN BEGIN-TEXT (HANDLE)
  (STATE-CASE (:TEXT :BLOCK-END)
    (:BLOCK (END-BLOCK HANDLE)))
  (SETF (HANDLE-STATE HANDLE)
    :TEXT)
  (HANDLE-STREAM HANDLE))

```

```

(DEFUN BEGIN-BLOCK (HANDLE)
  (STATE-CASE (:BLOCK-END (BEGIN-TEXT HANDLE)
    (END-TEXT HANDLE))
    (:TEXT (END-TEXT HANDLE))
    (:BLOCK))

```

```

(DEFUN VALUE-DUMPABLE-P (OBJ)
  (XCL:CONDITION-CASE (PROGN (DUMP-VALUE DUMMY-HANDLE OBJ NIL)
    T)
    (OBJECT-NOT-DUMPABLE NIL NIL)))

```

```

(DEFUN DUMP-VALUE (HANDLE VALUE &OPTIONAL (REMEMBER T)
  &AUX INDEX)
  (STATE-CASE (:BLOCK (COND
    ((EQ VALUE NIL)

```

```

( WRITE-OP HANDLE 'FASL-NIL))
(EQ VALUE T)
( WRITE-OP HANDLE 'FASL-T))
(PROG1 (SETQ INDEX (LOOKUP-VALUE HANDLE VALUE))
(WHEN *GATHER-DUMPER-STATS* (INCF *TABLE-ATTEMPTS*))
(WHEN *GATHER-DUMPER-STATS* (INCF *TABLE-HITS*))
(DUMP-VALUE-FETCH HANDLE INDEX))
(T (TYPECASE VALUE
(INTEGER (DUMP-INTEGGER HANDLE VALUE REMEMBER))
(RATIONAL (DUMP-RATIONAL HANDLE VALUE REMEMBER))
(SINGLE-FLOAT (DUMP-FLOAT32 HANDLE VALUE REMEMBER))
(COMPLEX (DUMP-COMPLEX HANDLE VALUE REMEMBER))
(CCHARACTER (DUMP-CHARACTER HANDLE VALUE REMEMBER))
(SYMBOL (DUMP-SYMBOL HANDLE VALUE REMEMBER))
(PACKAGE (DUMP-PACKAGE HANDLE VALUE REMEMBER))
(CONS (DUMP-LIST HANDLE VALUE REMEMBER))
(D-ASSEM:DCODE (DUMP-DCODE HANDLE VALUE REMEMBER))
(STRING (DUMP-STRING HANDLE VALUE REMEMBER))
(ARRAY (DUMP-ARRAY HANDLE VALUE REMEMBER))
(COMPILER::EVAL-WHEN-LOAD (LET ((REMEMBER T))
; always remember these.
(REMEMBER VALUE (DUMP-EVAL HANDLE
(
COMPILER::EVAL-WHEN-LOAD-FORM
VALUE))))))
(CL::STRUCTURE-OBJECT (DUMP-STRUCTURE HANDLE VALUE REMEMBER))
(IL:BITMAP (DUMP-BITMAP HANDLE VALUE REMEMBER))
(OTHERWISE (ERROR 'OBJECT-NOT-DUMPABLE :OBJECT VALUE))))))

```

```

(DEFUN DUMP-FUNCTION-DEF (HANDLE DCODE NAME)
(STATE-CASE (:BLOCK (WRITE-OP HANDLE 'FASL-SETF-SYMBOL-FUNCTION)
(DUMP-VALUE HANDLE NAME)
(DUMP-VALUE HANDLE DCODE))))

```

```

(DEFUN DUMP-FUNCALL (HANDLE FUNCTION)
(STATE-CASE (:BLOCK (WRITE-OP HANDLE 'FASL-FUNCALL)
(DUMP-VALUE HANDLE FUNCTION))))

```

```

(DEFUN DUMP-EVAL (HANDLE FORM)
(STATE-CASE (:BLOCK (WRITE-OP HANDLE 'FASL-EVAL)
(DUMP-VALUE HANDLE FORM))))

```

```

(DEFUN CLOSE-FASL-HANDLE (HANDLE &REST CLOSE-OPTIONS &KEY ABORT &ALLOW-OTHER-KEYS)
(STATE-CASE (:TEXT (END-TEXT HANDLE)
(END-BLOCK HANDLE))
(:BLOCK (END-BLOCK HANDLE))
(:BLOCK-END))
(IL:BOUT (HANDLE-STREAM HANDLE)
END-OF-DATA-MARK)
(SETF (HANDLE-STATE HANDLE)
:CLOSED)
(APPLY #'CLOSE (HANDLE-STREAM HANDLE)
CLOSE-OPTIONS))

```

:: Arrange for the correct compiler and makefile environment

```

(IL:PUTPROPS IL:FASDUMP IL:FILETYPE :COMPILE-FILE)
(IL:PUTPROPS IL:FASDUMP IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE "FASL"))
(IL:PUTPROPS IL:FASDUMP IL:COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1988 1990 1991 1993))

```

FUNCTION INDEX

BEGIN-BLOCK	6	DUMP-DCODE	5	DUMP-SIMPLE-VECTOR	3	INTEGER-BYTE-LIST	4
BEGIN-TEXT	6	DUMP-EVAL	7	DUMP-STRING	5	LOOKUP-VALUE	2
CLOSE-FASL-HANDLE	7	DUMP-FLOAT32	6	DUMP-STRUCTURE	6	OPEN-FASL-HANDLE	6
DOTTED-LIST-LENGTH	1	DUMP-FUNCALL	7	DUMP-SYMBOL	3	RESET-DUMPER-STATS	1
DUMP-ARRAY	4	DUMP-FUNCTION-DEF	7	DUMP-VALUE	6	SAVE-VALUE	2
DUMP-ARRAY-DESCRIPTOR	3	DUMP-GENERAL-ARRAY	4	DUMP-VALUE-FETCH	2	VALUE-DUMPABLE-P	6
DUMP-BIT-ARRAY	3	DUMP-INTEGERS	4	ELEMENTS-IDENTICAL-P	2	WRITE-INTEGERS-BYTES	4
DUMP-BITMAP	6	DUMP-LIST	3	END-BLOCK	2	WRITE-OP	2
DUMP-CHARACTER	3	DUMP-PACKAGE	5	END-TEXT	2		
DUMP-COMPLEX	4	DUMP-RATIONAL	4	FAT-STRING-P	2		

MACRO INDEX

REMEMBER	2	STATE-CASE	2	WITH-OPEN-HANDLE	6
----------------	---	------------------	---	------------------------	---

VARIABLE INDEX

GATHER-DUMPER-STATS	1	*TABLE-ATTEMPTS*	1	*TABLE-HITS*	1
-----------------------------	---	------------------------	---	--------------------	---

CONSTANT INDEX

+LARGEST-FOUR-BYTE-INTEGERS+	1	+SMALLEST-FOUR-BYTE-INTEGERS+	1	DUMMY-HANDLE	1
------------------------------------	---	-------------------------------------	---	--------------------	---

PROPERTY INDEX

IL:FASDUMP	7
------------------	---

STRUCTURE INDEX

HANDLE	1
--------------	---
