

File created: 23-Nov-2021 12:29:28 {DSK}<Users>kaplan>Local>medley3.5>my-medley>sources>FASLOAD.;5

changes to: (IL:FNS CONVERT-FASL-DATE)

previous date: 23-Nov-2021 09:44:12 {DSK}<Users>kaplan>Local>medley3.5>my-medley>sources>FASLOAD.;2

Read Table: XCL

Package: FASL

Format: XCCS

; Copyright (c) 1986-1992, 2018, 2021 by Venue & Xerox Corporation.

(IL:RPAQQ **IL:FASLOADCOMS**

;; FASL file loader.

;; THIS FILE IS DUPLICATED as ...<Lispcore>Sources> for the large-symbol version, and <Lispcore>Sources>2-byte> for the older  
;; 2-byte atom version. IF YOU CHANGE THIS COPY, CHANGE THE OTHER, AS WELL!

(IL:COMS

;; Common definitions.

(IL:DECLARE\ : IL:EVAL@COMPILE IL:EVAL@LOAD IL:DONTCOPY (IL:FILES (NIL IL:SOURCE)  
IL:FASL-SUPPORT))

(IL:STRUCTURES FASL-ERROR UNIMPLEMENTED-OPCODE OBJECT-NOT-DUMPABLE UNEXPECTED-END-OF-BLOCK  
INCONSISTENT-TABLE)

(IL:VARIABLES SIGNATURE)

(IL:VARIABLES CHECK-TABLE-SIZE FASL-EXTENDED END-MARK END-OF-DATA-MARK VERSION-RANGE  
CURRENT-VERSION)

(IL:FUNCTIONS TABLE-STATS))

(IL:COMS

;; Reader.

(IL:COMS

; Setting up the table

(IL:STRUCTURES OPTABLE)

(IL:FUNCTIONS MAKE-OPTABLE DEFINE-OPCODE-RANGE DEFINE-SINGLE-OPCODE ADD-OP-TRANSLATION  
OPCODE-SEQUENCE)

; Opcode definers

(IL:FUNCTIONS DEFOP DEFRANGE))

(IL:FUNCTIONS FASL-END-OF-BLOCK FASL-EXTENDED SETESCAPE UNIMPLEMENTED-OPCODE)

(IL:VARIABLES \*DEFAULT-OPTABLE\* \*CURRENT-OPTABLE\* INITIAL-VALUE-TABLE-SIZE  
VALUE-TABLE-INCREMENT \*VALUE-TABLE\* \*BLOCK-LEVEL\* DEBUG-READER DEBUG-STREAM)

;; The main reader functions:

(IL:FUNCTIONS PROCESS-FILE PROCESS-SEGMENT)

(IL:FUNCTIONS WITH-OPTABLE CHECK-VERSION READ-TEXT PROCESS-BLOCK SKIP-TEXT NEXT-VALUE DO-OP  
NEW-VALUE-TABLE CLEAR-TABLE STORE-VALUE FETCH-VALUE COLLECT-LIST)

;; FASL Opcode processors:

(FASL-OPS FASL-SHORT-INTEGERS FASL-NIL FASL-T FASL-INTEGERS FASL-LARGE-INTEGERS FASL-RATIO

FASL-COMPLEX FASL-VECTOR FASL-CREATE-ARRAY FASL-INITIALIZE-ARRAY

FASL-INITIALIZE-BIT-ARRAY FASL-THIN-STRING FASL-FAT-STRING FASL-CHARACTER

FASL-LISP-SYMBOL FASL-KEYWORD-SYMBOL FASL-FIND-PACKAGE FASL-SYMBOL-IN-PACKAGE FASL-LIST

FASL-LIST\* FASL-INTERLISP-SYMBOL FASL-DCODE FASL-LOCAL-FN-FIXUPS FASL-TABLE-STORE

FASL-TABLE-FETCH FASL-VERIFY-TABLE-SIZE FASL-EVAL FASL-FLOAT32

FASL-SETF-SYMBOL-FUNCTION FASL-FUNCALL FASL-BITMAP16 FASL-STRUCTURE))

(XCL:OPTIMIZERS FIXUP-NTOFFSET)

;; make sure there's some print function around so that you can load early.

(IL:P (IL:MOVD? 'IL:PRIN1 'PRINC)

(IL:MOVD? 'IL:TERPRI 'TERPRI))

(IL:COMS

;; ADDITION TO FILEDATE so it will handle FASL files as well as LCOMs and source files.

(IL:FNS IL:FASL-FILEDATE CONVERT-FASL-DATE))

;; Arrange for the correct compiler and makefile environment

(IL:PROP (IL:FILETYPE IL:MAKEFILE-ENVIRONMENT)

IL:FASLOAD))

;; FASL file loader.

;; THIS FILE IS DUPLICATED as ...<Lispcore>Sources> for the large-symbol version, and <Lispcore>Sources>2-byte> for the older 2-byte atom  
;; version. IF YOU CHANGE THIS COPY, CHANGE THE OTHER, AS WELL!

;; Common definitions.

(IL:DECLARE\ : IL:EVAL@COMPILE IL:EVAL@LOAD IL:DONTCOPY

(IL:FILESLOAD (NIL IL:SOURCE)

IL:FASL-SUPPORT)

)

(XCL:DEFINE-CONDITION **FASL-ERROR** (ERROR)

NIL)

```
(XCL:DEFINE-CONDITION UNIMPLEMENTED-OPCODE (FASL-ERROR)
  (OPNAME)
  (:REPORT (LAMBDA (CONDITION *STANDARD-OUTPUT*)
    (FORMAT T "Unimplemented FASL op: ~S" (UNIMPLEMENTED-OPCODE-OPNAME CONDITION))))))
```

```
(XCL:DEFINE-CONDITION OBJECT-NOT-DUMPABLE (FASL-ERROR)
  (OBJECT)
  (:REPORT (LAMBDA (CONDITION *STANDARD-OUTPUT*)
    (FORMAT T "Object not dumpable:~&~S" (OBJECT-NOT-DUMPABLE-OBJECT CONDITION))))))
```

```
(XCL:DEFINE-CONDITION UNEXPECTED-END-OF-BLOCK (FASL-ERROR)
  (STREAM)
  (:REPORT (LAMBDA (CONDITION *STANDARD-OUTPUT*)
    (FORMAT T "Unexpected FASL-END-OF-BLOCK at ~D." (IL:GETFILEPTR (
      UNEXPECTED-END-OF-BLOCK-STREAM
      CONDITION))))))
```

```
(XCL:DEFINE-CONDITION INCONSISTENT-TABLE (FASL-ERROR)
  (TABLE EXPECTED)
  (:REPORT (LAMBDA (CONDITION *STANDARD-OUTPUT*)
    (FORMAT T "Inconsistent FASL table size.~&Expected ~D but found ~D." (
      INCONSISTENT-TABLE-EXPECTED
      CONDITION)
      (LENGTH (OPTABLE-VECTOR (INCONSISTENT-TABLE-TABLE CONDITION))))))
```

```
(DEFCONSTANT SIGNATURE 145
  "First byte of a FASL file.")
```

```
(DEFVAR CHECK-TABLE-SIZE T)
```

```
(DEFCONSTANT FASL-EXTENDED 254)
```

```
(DEFCONSTANT END-MARK 255)
```

```
(DEFCONSTANT END-OF-DATA-MARK 255
  "End-of-data marker: if first byte of a segment, terminate processing")
```

```
(DEFCONSTANT VERSION-RANGE '(8 . 8)
  "Handles (car version-range) <= version <= (cdr version-range)")
```

```
(DEFCONSTANT CURRENT-VERSION 8)
```

```
(DEFUN TABLE-STATS (TABLE)
  (LET ((ITEMS (LIST (CONS '--TOTAL-- (LENGTH TABLE))))
        (DOTIMES (I (LENGTH TABLE)
          ITEMS)
          (LET* ((TYPE (TYPE-OF (AREF TABLE I)))
                 (PAIR (OR (FIND TYPE ITEMS :TEST 'EQUAL :KEY 'CAR)
                            (CAR (PUSH (CONS TYPE 0)
                                         ITEMS))))
                (INCF (CDR PAIR))))))
```

```
:: Reader.
:: Setting up the table
```

```
(DEFSTRUCT (OPTABLE (:CONSTRUCTOR NEW-OPTABLE))
  VECTOR
  OPNAMES
  NEXT)
```

```
(DEFUN MAKE-OPTABLE ()
  (LET ((TABLE (NEW-OPTABLE))
        (VECTOR (MAKE-ARRAY 256 :INITIAL-ELEMENT 'UNIMPLEMENTED-OPCODE)))
    (SETF (OPTABLE-VECTOR TABLE)
          VECTOR)
    (SETF (SVREF VECTOR END-MARK)
          'FASL-END-OF-BLOCK)
    TABLE))
```

```
(DEFUN DEFINE-OPCODE-RANGE (NAME FIRST-OPCODE RANGE OFFSET TABLE)
  ;; For implementation of DEFRANGE definer--define a range of opcodes having the same implementation.
```

```
(LET ((PACKAGE (SYMBOL-PACKAGE NAME))
      (PNAME (SYMBOL-NAME NAME))
      (DOTIMES (I RANGE)
        (DEFINE-SINGLE-OPCODE NAME (+ I FIRST-OPCODE)
          TABLE
          (INTERN (IL:CONCAT PNAME (+ I OFFSET))
            PACKAGE))))))
```

; Using IL:CONCAT here to minimize bootstrap woes

```
(DEFUN DEFINE-SINGLE-OPCODE (NAME OPCODE TABLE TRANS-NAME)
  ;; For implementation of DEFOP definer -- define NAME to be a fasl op numbered OPCODE in TABLE. NAME is the name of both the opcode as a
  ;; FASL::FASL-OPS and the function implementing the opcode. TRANS-NAME is a name to associate with the opcode in the OPNAMES slot of the
  ;; table (it is a generated name when we are called from DEFRANGE).
```

```
(SETF (ELT (OPTABLE-VECTOR TABLE)
          OPCODE)
      NAME)
(ADD-OP-TRANSLATION TRANS-NAME OPCODE TABLE))
```

```
(DEFUN ADD-OP-TRANSLATION (NAME OPCODE TABLE)
  (LET ((PAIR (ASSOC NAME (OPTABLE-OPNAMES TABLE))))
    (IF PAIR
      (SETF (CDR PAIR)
            OPCODE)
      (PUSH (CONS NAME OPCODE)
            (OPTABLE-OPNAMES TABLE)))))
```

```
(DEFUN OPCODE-SEQUENCE (OPNAME &OPTIONAL (TABLE *DEFAULT-OPTABLE*)
                        &AUX ENTRY)
```

```
(COND
  ((NULL TABLE)
   NIL)
  ((SETQ ENTRY (ASSOC OPNAME (OPTABLE-OPNAMES TABLE)))
   (LIST (CDR ENTRY)))
  ((SETQ ENTRY (OPCODE-SEQUENCE OPNAME (OPTABLE-NEXT TABLE)))
   (CONS FASL-EXTENDED ENTRY))
  (T NIL)))
```

;; Opcode definers

```
(XCL:DEFDEFINER DEFOP FASL-OPS (IL:NAME (OPCODE &KEY (INDIRECT 0)
                                                    (TABLE '*DEFAULT-OPTABLE*'))
                                &BODY BODY)
```

```
(IF (ZEROP INDIRECT)
  `(PROGN (DEFUN ,IL:NAME (STREAM OPCODE)
                        ,@BODY)
          (DEFINE-SINGLE-OPCODE ',IL:NAME ,OPCODE ,TABLE ',IL:NAME))
  `(PROGN (UNLESS (OPTABLE-NEXT ,TABLE)
                  (SETF (OPTABLE-NEXT ,TABLE)
                        (MAKE-OPTABLE))
                  (SETESCAPE ,TABLE))
          (DEFOP ,IL:NAME (,OPCODE :INDIRECT ,(1- INDIRECT)
                            :TABLE
                            (OPTABLE-NEXT ,TABLE))
                    ,@BODY))))
```

```
(XCL:DEFDEFINER DEFRANGE FASL-OPS (IL:NAME (FIRST-OPCODE &KEY (INDIRECT 0)
                                                    (TABLE '*DEFAULT-OPTABLE*'))
                                RANGE OFFSET &BODY BODY)
```

```
(IF (ZEROP INDIRECT)
  `(PROGN (DEFUN ,IL:NAME (STREAM OPCODE)
                        ,@BODY)
          (DEFINE-OPCODE-RANGE ',IL:NAME ,FIRST-OPCODE ,RANGE ,OFFSET ,TABLE))
  `(PROGN (UNLESS (OPTABLE-NEXT ,TABLE)
                  (SETF (OPTABLE-NEXT ,TABLE)
                        (MAKE-OPTABLE))
                  (SETESCAPE ,TABLE))
          (DEFRANGE ,IL:NAME (,FIRST-OPCODE :INDIRECT ,(1- INDIRECT)
                            :TABLE
                            (OPTABLE-NEXT ,TABLE)) ,@BODY))))
```

```
(DEFUN FASL-END-OF-BLOCK (STREAM OP)
  (IF (ZEROP *BLOCK-LEVEL*)
    (THROW 'FASL-BLOCK-FINISHED NIL)
    (ERROR 'UNEXPECTED-END-OF-BLOCK :STREAM STREAM)))
```

```
(DEFUN FASL-EXTENDED (STREAM OP)
  (WITH-OPTABLE (OPTABLE-NEXT *CURRENT-OPTABLE*)
    (DO-OP STREAM)))
```

```
(DEFUN SETESCAPE (TABLE)
  (SETF (SVREF (OPTABLE-VECTOR TABLE)
```

```
FASL-EXTENDED)
#'FASL-EXTENDED))
```

```
(DEFUN UNIMPLEMENTED-OPCODE (STREAM OPCODE)
 (ERROR 'UNIMPLEMENTED-OPCODE :OPNAME OPCODE))
```

```
(DEFVAR *DEFAULT-OPTABLE* (MAKE-OPTABLE))
```

```
(DEFVAR *CURRENT-OPTABLE* NIL)
```

```
(DEFPARAMETER INITIAL-VALUE-TABLE-SIZE 2048)
```

```
(DEFCONSTANT VALUE-TABLE-INCREMENT 1024)
```

```
(DEFVAR *VALUE-TABLE* NIL)
```

```
(DEFVAR *BLOCK-LEVEL* 0)
```

```
(DEFVAR DEBUG-READER NIL)
```

```
(DEFVAR DEBUG-STREAM NIL)
```

:: The main reader functions:

```
(DEFUN PROCESS-FILE (STREAM &KEY (TEXT-FN (AND *LOAD-VERBOSE* #'(LAMBDA (TEXT)
 (PRINC TEXT)
 (TERPRI))))
 (ITEM-FN NIL))
```

::: Calls FASL:PROCESS-SEGMENT with the appropriate arguments for each segment in the file. The stream should be positioned at the beginning.

```
(UNLESS (EQL (IL:BIN STREAM)
 SIGNATURE)
 (ERROR "Not a FASL file."))
(LET ((IL:FILEPKGFLG NIL)
 (IL:DFNFLG T)
 (IL:LISPXHIST NIL)
 (IL:ADDSPELLFLG NIL))
```

; Bind these so that LOADING a FASL file is like LOADING  
; SYSLOAD.

```
(DECLARE (SPECIAL IL:FILEPKGFLG IL:DFNFLG IL:LISPXHIST IL:ADDSPELLFLG))
(IF (< (CHECK-VERSION STREAM)
 5)
 (DO NIL
 ((IL:EOFP STREAM)
 (VALUES)
 (PROCESS-SEGMENT STREAM TEXT-FN ITEM-FN))
 (DO NIL
 ((OR (IL:EOFP STREAM)
 (EQL (IL:\\PEEKBIN STREAM)
 END-OF-DATA-MARK))
 (VALUES)
 (PROCESS-SEGMENT STREAM TEXT-FN ITEM-FN))))))
```

```
(DEFUN PROCESS-SEGMENT (STREAM &OPTIONAL TEXT-FN ITEM-FN (OPTABLE *DEFAULT-OPTABLE*))
 (IF TEXT-FN
 (FUNCALL TEXT-FN (READ-TEXT STREAM))
 (SKIP-TEXT STREAM))
 (PROCESS-BLOCK STREAM ITEM-FN OPTABLE))
```

```
(DEFMACRO WITH-OPTABLE (TABLE &BODY BODY)
 `(LET ((*CURRENT-OPTABLE* ,TABLE)
 ,@BODY))
```

```
(DEFUN CHECK-VERSION (STREAM)
 (LET ((VERSION (IL:BIN16 STREAM)))
 (UNLESS (AND (<= (CAR VERSION-RANGE)
 VERSION)
 (<= VERSION (CDR VERSION-RANGE)))
 (ERROR "Version not supported: ~D." VERSION))
 (RETURN-FROM CHECK-VERSION VERSION)))
```

```
(DEFUN READ-TEXT (STREAM)
```

:: RMK: This really should be doing READCCODE to read the bytes, but that fails because this string is not delimited by quotes, rather it has 255 as the end marker. 255 is the XCCS character set shift, will presumably do something else in Unicode.

:: Any reason not to print the string as a string?

```
(DO ((RESULT (MAKE-ARRAY 512 :ELEMENT-TYPE 'CHARACTER :ADJUSTABLE T :FILL-POINTER 0))
      (BYTE (IL:BIN STREAM)
            (IL:BIN STREAM)))
    ((EQL BYTE END-MARK)
     RESULT)
 (VECTOR-PUSH-EXTEND (CODE-CHAR (IL:\\CHECKEOLC BYTE NIL STREAM)
                               RESULT)))
```

```
(DEFUN PROCESS-BLOCK (STREAM &OPTIONAL ITEM-FN (OPTABLE *DEFAULT-OPTABLE*))
  (IL:WITH-READER-ENVIRONMENT IL:*COMMON-LISP-READ-ENVIRONMENT*
    (CATCH 'FASL-BLOCK-FINISHED
      (WITH-OPTABLE OPTABLE (DO ((*VALUE-TABLE* (NEW-VALUE-TABLE))
                                VAL)
                                ()
                                (SETF VAL (DO-OP STREAM 0))
                                (WHEN ITEM-FN (FUNCALL ITEM-FN VAL)))))))
```

```
(DEFUN SKIP-TEXT (STREAM)
  (DO ((BYTE (IL:BIN STREAM)
            (IL:BIN STREAM)))
      ((EQL BYTE END-MARK)
       (VALUES))))
```

```
(DEFMACRO NEXT-VALUE ()
  ' (DO-OP STREAM))
```

```
(DEFUN DO-OP (STREAM &OPTIONAL (*BLOCK-LEVEL* (1+ *BLOCK-LEVEL*)))
  (LET ((OP (IL:BIN STREAM)
        VAL)
        (WHEN DEBUG-READER
          (FORMAT DEBUG-STREAM "~VT-A (~30)~%" (* *BLOCK-LEVEL* 4)
                  (CAR (ASSOC OP (OPTABLE-OPNAMES *CURRENT-OPTABLE*)))
                  OP)))
        (SETQ VAL (FUNCALL (SVREF (OPTABLE-VECTOR *CURRENT-OPTABLE*)
                                OP)
                          STREAM OP)))
        (WHEN DEBUG-READER
          (FORMAT DEBUG-STREAM "~VTValue: ~S~%" (* *BLOCK-LEVEL* 4)
                  VAL)))
    (RETURN-FROM DO-OP VAL)))
```

```
(DEFUN NEW-VALUE-TABLE ()
  (MAKE-ARRAY INITIAL-VALUE-TABLE-SIZE :FILL-POINTER 0 :EXTENDABLE T))
```

```
(DEFUN CLEAR-TABLE (&OPTIONAL (TABLE *VALUE-TABLE*))
  (SETF (FILL-POINTER TABLE)
        0))
```

```
(DEFUN STORE-VALUE (OBJ &OPTIONAL (TABLE *VALUE-TABLE*))
  :: This may want to change to another representation if we can't make VECTOR-PUSH-EXTEND fast enough.
  (VECTOR-PUSH-EXTEND OBJ TABLE VALUE-TABLE-INCREMENT)
  OBJ)
```

```
(DEFUN FETCH-VALUE (INDEX &OPTIONAL (TABLE *VALUE-TABLE*))
  (AREF TABLE INDEX))
```

```
(DEFUN COLLECT-LIST (STREAM NELTS DOTTED)
  (IF (AND DOTTED (EQL NELTS 2))
      (RETURN-FROM COLLECT-LIST (CONS (DO-OP STREAM)
                                      (DO-OP STREAM))))
      (WHEN DOTTED (DECF NELTS))
      (LET ((RESULT (IL:|to| NELTS IL:|collect| (DO-OP STREAM))))
          :: Assume dotted and other than a simple cons is rare.
          (WHEN DOTTED
            (SETF (CDR (LAST RESULT))
                  (DO-OP STREAM)))
            (RETURN-FROM COLLECT-LIST RESULT))))
```

:: FASL Opcode processors:

```
(DEFRANGE FASL-SHORT-INTEGERS (0) 128 0
```

"An entire set of FASL opcodes representing small integers"  
OPCODE)

(DEFOP **FASL-NIL** (128)  
NIL)

(DEFOP **FASL-T** (129)  
T)

(DEFOP **FASL-INTEGER** (130)  
(+ (IL:LLSH (IL:BIN STREAM)  
24)  
(IL:LLSH (IL:BIN STREAM)  
16)  
(IL:LLSH (IL:BIN STREAM)  
8)  
(IL:BIN STREAM)))

(DEFOP **FASL-LARGE-INTEGER** (131)  
(LET ((NBYTES (NEXT-VALUE))  
(FIRST-TIME T)  
(MASK 0))  
(DO ((OFFSET (\* (1- NBYTES)  
8)  
(- OFFSET 8))  
(RESULT 0)  
BYTE)  
((< OFFSET 0)  
(IF (ZEROP MASK)  
RESULT  
(- (1+ RESULT))))  
(SETF BYTE (IL:BIN STREAM))  
(WHEN FIRST-TIME  
(SETF FIRST-TIME NIL)  
(WHEN (> BYTE 127)  
(SETQ MASK 255)))  
(SETF (LDB (BYTE 8 OFFSET)  
RESULT)  
(LOGXOR BYTE MASK))))))

(DEFOP **FASL-RATIO** (134)  
(/ (NEXT-VALUE)  
(NEXT-VALUE)))

(DEFOP **FASL-COMPLEX** (135)  
(COMPLEX (NEXT-VALUE)  
(NEXT-VALUE)))

(DEFOP **FASL-VECTOR** (136)  
(LET\* ((NELTS (NEXT-VALUE))  
(VECTOR (MAKE-ARRAY NELTS :INITIAL-ELEMENT NIL)))  
(DOTIMES (I NELTS VECTOR)  
(SETF (AREF VECTOR I)  
(NEXT-VALUE))))))

(DEFOP **FASL-CREATE-ARRAY** (137)  
(APPLY #'MAKE-ARRAY (NEXT-VALUE)  
(NEXT-VALUE)))

(DEFOP **FASL-INITIALIZE-ARRAY** (138)  
(LET\* ((ARRAY (NEXT-VALUE))  
(INDIRECT (IL:%FLATTEN-ARRAY ARRAY))  
(NELTS (NEXT-VALUE)))  
(DOTIMES (I NELTS ARRAY)  
(SETF (AREF INDIRECT I)  
(NEXT-VALUE))))))

(DEFOP **FASL-INITIALIZE-BIT-ARRAY** (139)  
(LET\* ((ARRAY (DO-OP STREAM))  
(BASE (IL:%ARRAY-BASE ARRAY))  
(NBITS (DO-OP STREAM)))  
(MULTIPLE-VALUE-BIND (NBYTES LEFTOVER)  
(FLOOR NBITS 8)  
(IL:\\BINS STREAM BASE 0 NBYTES)  
(UNLESS (ZEROP LEFTOVER)  
(LET ((BD (BYTE LEFTOVER (- 8 LEFTOVER))))  
(SETF (LDB BD (IL:\\GETBASEBYTE BASE NBYTES))

```
(LDB BD (IL:BIN STREAM))))))
ARRAY)))
```

```
(DEFOP FASL-THIN-STRING (140)
 (LET* ((NCHARS (NEXT-VALUE))
 (STRING (IL:ALLOCSTRING NCHARS)))
 (IL:\\BINS STREAM (IL:FETCH (IL:STRINGP IL:BASE) IL:OF STRING)
 0 NCHARS)
 STRING))
```

```
(DEFOP FASL-FAT-STRING (141)
 ;; Read a string of specified length that has been encoded in standard NS format.
 (LET* ((NCHARS (NEXT-VALUE))
 (STRING (IL:ALLOCSTRING NCHARS)))
 (IL:ACCESS-CHARSET STREAM 0) ; Make sure we're in charset zero
 (UNWIND-PROTECT
 (DOTIMES (I NCHARS STRING)
 (SETF (SVREF STRING I)
 (CODE-CHAR (IL:READCCODE STREAM)))) ; Restore charset zero, in case anyone cares
 (IL:ACCESS-CHARSET STREAM 0))))
```

```
(DEFOP FASL-CHARACTER (142)
 (LET ((CODE (IL:BIN STREAM))
 (CODE-CHAR (IF (EQL CODE 255)
 (IL:BIN16 STREAM)
 CODE))))
```

```
(DEFOP FASL-LISP-SYMBOL (143)
 (INTERN (NEXT-VALUE)
 (FIND-PACKAGE "LISP")))
```

```
(DEFOP FASL-KEYWORD-SYMBOL (144)
 (INTERN (NEXT-VALUE)
 (FIND-PACKAGE "KEYWORD")))
```

```
(DEFOP FASL-FIND-PACKAGE (145)
 (LET ((NAME (NEXT-VALUE))
 (OR (FIND-PACKAGE NAME)
 (ERROR "FASL reader error: package ~S not found." NAME))))
```

```
(DEFOP FASL-SYMBOL-IN-PACKAGE (146)
 (LET* ((PNAME (NEXT-VALUE))
 (PACKAGE (NEXT-VALUE))
 (IF (NULL PACKAGE)
 (MAKE-SYMBOL PNAME)
 (INTERN PNAME PACKAGE))))
```

```
(DEFOP FASL-LIST (147)
 (COLLECT-LIST STREAM (NEXT-VALUE)
 NIL))
```

```
(DEFOP FASL-LIST* (148)
 (COLLECT-LIST STREAM (NEXT-VALUE)
 T))
```

```
(DEFOP FASL-INTERLISP-SYMBOL (149)
 (INTERN (NEXT-VALUE)
 (FIND-PACKAGE "INTERLISP")))
```

```
(DEFOP FASL-DCODE (150)
```

;;; DIRE WARNING!!! Be sure you have your pointy hat with lots of stars on if you're going to muck around with this code. Due to unfortunately  
;;; unavoidable performance requirements, this code duplicates D-ASSEM:INTERN-DCODE. If you make a change here, you should probably change  
;;; the corresponding code there.

```
(LET ((OVERHEADBYTES (* (IL:FETCH (IL:FNHEADER IL:OVERHEADWORDS) IL:OF T)
 IL:BYTESPERWORD))
 NT-COUNT RAW-CODE START-PC CLOSURE-INFO)
 (SETF NT-COUNT (NEXT-VALUE))
 (LET ((CODE-LEN (NEXT-VALUE))
 (MULTIPLE-VALUE-SETQ (RAW-CODE START-PC)
 (D-ASSEM:ALLOCATE-CODE-BLOCK NT-COUNT CODE-LEN))
 (IL:\\BINS STREAM RAW-CODE START-PC CODE-LEN)
 (IL:REPLACE (IL:FNHEADER IL:STARTPC) IL:OF RAW-CODE IL:WITH START-PC))
```

;; Set up the free variable lookup name table.

```
(DO* ((I 0 (1+ I))
      (INDEX OVERHEADBYTES (+ INDEX (IL:CONSTANT (IL:BYTESPERNAMEENTRY))))
      ;; NTSIZE and NTBYTESIZE the sizes of half the table in words and bytes resp.
      (NTSIZE (IL:CEIL (1+ (IL:UNFOLD NT-COUNT (IL:CONSTANT (IL:WORDSPERNAMEENTRY))))
              (IL:WORDSPERQUAD))
      (NTBYTESIZE (* NTSIZE IL:BYTESPERWORD))
      PFI OFFSET NAME FVAROFFSET)
      ((>= I NT-COUNT)
       (IL:REPLACE (IL:FNHEADER IL:FVAROFFSET) IL:OF RAW-CODE IL:WITH (OR FVAROFFSET 0))
       (IL:REPLACE (IL:FNHEADER IL:NTSIZE) IL:OF RAW-CODE IL:WITH (IF (ZEROP NT-COUNT)
                               0
                               NTSIZE))))

      (SETF PFI (IL:BIN STREAM))
      (SETF OFFSET (NEXT-VALUE))
      (SETF NAME (NEXT-VALUE))
      (D-ASSEM:FIXUP-NTENTRY RAW-CODE INDEX (IL:\\ATOMVALINDEX NAME))
      (FIXUP-NTOFFSET RAW-CODE (+ INDEX NTBYTESIZE)
      (IL:LSSH PFI 14)
      OFFSET)
      (WHEN (AND (NULL FVAROFFSET)
                (= PFI D-ASSEM:+FVAR-CODE+))
            (SETF FVAROFFSET (FLOOR INDEX IL:BYTESPERWORD))))

;; Fill in the fixed-size fields at the front of the block.
(LET ((FRAME-NAME (NEXT-VALUE))
      (IL:UNINTERRUPTABLY
      (IL:\\ADDRF FRAME-NAME)
      (IL:REPLACE (IL:FNHEADER IL:FRAME-NAME) IL:OF RAW-CODE IL:WITH FRAME-NAME)))
      (LET ((NLOCALS (IL:BIN STREAM))
            (NFREEVARS (IL:BIN STREAM))
            (IL:REPLACE (IL:FNHEADER IL:NLOCALS) IL:OF RAW-CODE IL:WITH NLOCALS)
            (IL:REPLACE (IL:FNHEADER IL:PV) IL:OF RAW-CODE IL:WITH (1- (CEILING (+ NLOCALS NFREEVARS)
                                         IL:CELLSPERQUAD))))
            (IL:REPLACE (IL:FNHEADER IL:ARGTYPE) IL:OF RAW-CODE IL:WITH (IL:BIN STREAM))
            (IL:REPLACE (IL:FNHEADER IL:NA) IL:OF RAW-CODE IL:WITH (NEXT-VALUE))
            (SETF CLOSURE-INFO (NEXT-VALUE))
            (IL:REPLACE (IL:FNHEADER IL:CLOSUREP) IL:OF RAW-CODE IL:WITH (EQ CLOSURE-INFO :CLOSURE))
            (IL:REPLACE (IL:FNHEADER IL:FIXED) IL:OF RAW-CODE IL:WITH T)

            ;; Fill in debugging info. It goes into the spare cell just before the code: it's -3 instead of -bytespercell to right-justify the pointer in the cell.
            ;; Aren't you glad I told you this?

            (D-ASSEM:FIXUP-PTR RAW-CODE (- START-PC (IL:BIG-VMEM-CODE 4 3))
            (NEXT-VALUE))

            ;; Do fixups
            (DO ((FN-FIXUP-COUNT (NEXT-VALUE))
                (I 0 (1+ I))
                (OFFSET VALUE)
                ((>= I FN-FIXUP-COUNT))
                (SETF OFFSET (NEXT-VALUE))
                (SETF VALUE (NEXT-VALUE))
                (D-ASSEM:FIXUP-SYMBOL RAW-CODE (+ START-PC OFFSET)
                VALUE))
                (DO ((SYM-FIXUP-COUNT (NEXT-VALUE))
                    (I 0 (1+ I))
                    (OFFSET VALUE)
                    ((>= I SYM-FIXUP-COUNT))
                    (SETF OFFSET (NEXT-VALUE))
                    (SETF VALUE (NEXT-VALUE))
                    (D-ASSEM:FIXUP-SYMBOL RAW-CODE (+ START-PC OFFSET)
                    VALUE))
                    (DO ((LIT-FIXUP-COUNT (NEXT-VALUE))
                        (I 0 (1+ I))
                        (OFFSET VALUE)
                        ((>= I LIT-FIXUP-COUNT))
                        (SETF OFFSET (NEXT-VALUE))
                        (SETF VALUE (NEXT-VALUE))
                        (D-ASSEM:FIXUP-PTR RAW-CODE (+ START-PC OFFSET)
                        VALUE))
                        (DO ((TYPE-FIXUP-COUNT (NEXT-VALUE))
                            (I 0 (1+ I))
                            (OFFSET VALUE)
                            ((>= I TYPE-FIXUP-COUNT))
                            (SETF OFFSET (NEXT-VALUE))
                            (SETF VALUE (NEXT-VALUE))
                            (D-ASSEM:FIXUP-WORD RAW-CODE (+ START-PC OFFSET)
                            (IL:\\RESOLVE.TYPENUMBER VALUE))))

                            ;; Finally, wrap this up in a closure-object if requested.
                            (IF (EQ CLOSURE-INFO :FUNCTION)
                                (IL:MAKE-COMPILED-CLOSURE RAW-CODE NIL)
                                RAW-CODE)))
```



```
(LET ((PASS-THROUGH (NEXT-VALUE))) ; This will typically correspond to the DCODE that had the fixups,
; but can be anything.
(DO ((FIXUP-COUNT (NEXT-VALUE))
(I 0 (IL:ADD1 I))
CODE-TO-FIX OFFSET VALUE)
((IL:IGEQ I FIXUP-COUNT)
PASS-THROUGH)
(SETF CODE-TO-FIX (NEXT-VALUE)
OFFSET
(NEXT-VALUE)
VALUE
(NEXT-VALUE))
(MACROLET ((GET-CODE (THING)
(XCL:ONCE-ONLY (THING)
` (IF (TYPEP ,THING 'IL:COMPILED-CLOSURE)
(IL:FETCH (IL:COMPILED-CLOSURE IL:FNHEADER) IL:OF ,THING)
,THING))))
(IF (EQ CODE-TO-FIX VALUE)
(LET ((CODE (GET-CODE CODE-TO-FIX)))
(D-ASSEM:FIXUP-PTR-NO-REF CODE (IL:IPLUS (IL:FETCH (IL:FNHEADER IL:STARTPC)
IL:OF CODE)
OFFSET)
VALUE))
(LET ((CODE (GET-CODE CODE-TO-FIX)))
(D-ASSEM:FIXUP-PTR CODE (IL:IPLUS (IL:FETCH (IL:FNHEADER IL:STARTPC) IL:OF CODE)
OFFSET)
VALUE))))))
```

```
(DEFOP FASL-TABLE-STORE (152)
(STORE-VALUE (NEXT-VALUE)))
```

```
(DEFOP FASL-TABLE-FETCH (153)
(FETCH-VALUE (NEXT-VALUE)))
```

```
(DEFOP FASL-VERIFY-TABLE-SIZE (154)
(LET ((EXPECTED (NEXT-VALUE))
(OR (EQL EXPECTED (XCL:VECTOR-LENGTH *VALUE-TABLE*))
(ERROR 'INCONSISTENT-TABLE :TABLE *VALUE-TABLE* :EXPECTED EXPECTED))))
```

```
(DEFOP FASL-EVAL (155)
(EVAL (NEXT-VALUE)))
```

```
(DEFOP FASL-FLOAT32 (132)
(LET ((RESULT (IL:NCREATE 'IL:FLOATP))
(IL:\\BINS STREAM RESULT 0 4)
RESULT))
```

```
(DEFOP FASL-SETF-SYMBOL-FUNCTION (156)
(SETF (SYMBOL-FUNCTION (NEXT-VALUE))
(NEXT-VALUE)))
```

```
(DEFOP FASL-FUNCALL (157)
(FUNCALL (NEXT-VALUE)))
```

```
(DEFOP FASL-BITMAP16 (158)
```

;;; Load an Interlisp BITMAP.

```
(LET* ((WIDTH (NEXT-VALUE))
(HEIGHT (NEXT-VALUE))
(BITS-PER-PIXEL (NEXT-VALUE))
(BITMAP (IL:BITMAPCREATE WIDTH HEIGHT BITS-PER-PIXEL))
(BASE (IL:FETCH (IL:BITMAP IL:BITMAPBASE) IL:OF BITMAP)))
(IL:\\BINS STREAM BASE 0 (* 2 HEIGHT (CEILING (* WIDTH BITS-PER-PIXEL)
16)))
BITMAP))
```

```
(DEFOP FASL-STRUCTURE (159)
```

;;; Load a DEFSTRUCT-defined structure instance.

```
(IL:CREATE-STRUCTURE (CONS (NEXT-VALUE)
(NEXT-VALUE)))
```

```
(XCL:DEFOPTIMIZER FIXUP-NTOFFSET (RAW-CODE OFFSET TYPE VALUE &ENVIRONMENT IL:ENV)
```

;; Do the fixups for a name-table offset entry, given a code block, the NTOffset's offset within the  
;; codeblock, and the variable type and FVAR offset.

```

(COND
  ((IL:FMEMB :3-BYTE (COMPILER::ENV-TARGET-ARCHITECTURE IL:ENV))
   ;; 3-byte case; the nametable entry is a full cell.
   \ (PROGN (D-ASSEM:FIXUP-WORD ,RAW-CODE ,OFFSET ,TYPE)
         (D-ASSEM:FIXUP-WORD ,RAW-CODE (+ ,OFFSET IL:BYTESPERWORD)
          ,VALUE)))
  (T ;; Old nametable case, it's just a word.
   \ (D-ASSEM:FIXUP-WORD ,RAW-CODE ,OFFSET (IL:IPLUS ,TYPE ,VALUE))))))

```

;; make sure there's some print function around so that you can load early.

```

(IL:MOVD? 'IL:PRIN1 'PRINC)
(IL:MOVD? 'IL:TERPRI 'TERPRI)

```

;; ADDITION TO FILEDATE so it will handle FASL files as well as LCOMs and source files.

```
(IL:DEFINEQ
```

**(IL:FASL-FILEDATE**

```

(IL:LAMBDA (STREAM IL:CFLG)
           ; Edited 23-Nov-2021 08:26 by rmk:
           ; Edited 17-Feb-89 11:25 by jds
           ; CFLG IS T FOR COMPILED FILES

```

;; If STREAM is open on a FASL file, returns the FILEDATE for that file. Otherwise, returns NIL.  
 ;; Used in FILEDATE; kept a separate function because FILEDATE is defined before the FASL package is loaded.

```

(COND
  ((EQL (IL:BIN STREAM)
        SIGNATURE)
   ; "Aha, a Dfasl file"
   (IL:SETFILEPTR STREAM 0)
   (CONVERT-FASL-DATE (PROCESS-FILE STREAM :TEXT-FN #'(IL:LAMBDA (IL:X)
                                                             (IL:RETFROM 'PROCESS-FILE IL:X))
                      :ITEM-FN
                      'IL:NILL)
    IL:CFLG))))))

```

**(CONVERT-FASL-DATE**

```

(IL:LAMBDA (IL:DATESTRING IL:CFLG)
           ; Edited 23-Nov-2021 12:29 by rmk:
           ; Edited 17-Apr-2018 07:55 by rmk:
           ; Edited 23-Jan-89 13:55 by gadener

```

;; CONVERT-FASL-DATE takes the file text info from a DFASL file describing creation dates for source and compiled code and returns either one  
 ;; of these dates, depending on the value of CLFG, in da-mon-yr hr:mn:sc format.  
 ;;  
 ;; RMK: 23-Nov-2021. Some DFASL files have a different date format, without the day before a comma and without a period at the end of the lines.  
 ;; It seems that the easiest thing is just to isolate the full date strings, stripping off the period at the end and then canonicalize the return date with  
 ;; (GDATE (IDATE)). IDATE in particular seems to recognize all the formats.  
 ;;

;; END-POS is the end of the line that contains the key substring, last char could be period

```

(LET* ((IL:DATE-SUFFIX (IL:SUBSTRING IL:DATESTRING (IL:STRPOS (IF IL:CFLG
                                                                "FASL file created "
                                                                "Source file created ")
                                                                IL:DATESTRING 1 NIL NIL T)))
       (IL:END-POS (OR (IL:STRPOS (IL:CHARACTER (IL:CHARCODE EOL))
                                  IL:DATE-SUFFIX)
                      (IL:SUB1 (IL:NCHARS IL:DATE-SUFFIX))))
       (IL:GDATE (IL:IDATE (IL:SUBSTRING IL:DATE-SUFFIX 1 (IF (EQ (IL:CHARCODE \.)
                                                                    (IL:NTHCHARCODE IL:END-POS -1))
                                                                (IL:SUB1 IL:END-POS 1)
                                                                IL:END-POS))))))

```

)

;; Arrange for the correct compiler and makefile environment

```

(IL:PUTPROPS IL:FASLOAD IL:FILETYPE COMPILE-FILE)
(IL:PUTPROPS IL:FASLOAD IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE "FASL"))
(IL:PUTPROPS IL:FASLOAD IL:COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1988 1989 1990 1991 1992 2018 2021))

```

---

**FUNCTION INDEX**

ADD-OP-TRANSLATION	.....3	DEFINE-SINGLE-OPCODE	....3	MAKE-OPTABLE	.....2	READ-TEXT	.....4
CHECK-VERSION	.....4	DO-OP	.....5	NEW-VALUE-TABLE	.....5	SETESCAPE	.....3
CLEAR-TABLE	.....5	FASL-END-OF-BLOCK	.....3	OPCODE-SEQUENCE	.....3	SKIP-TEXT	.....5
COLLECT-LIST	.....5	FASL-EXTENDED	.....3	PROCESS-BLOCK	.....5	STORE-VALUE	.....5
CONVERT-FASL-DATE	.....10	IL:FASL-FILEDATE	.....10	PROCESS-FILE	.....4	TABLE-STATS	.....2
DEFINE-OPCODE-RANGE	.....2	FETCH-VALUE	.....5	PROCESS-SEGMENT	.....4	UNIMPLEMENTED-OPCODE	....4

---

**FASL-OP INDEX**

FASL-BITMAP16	.....9	FASL-INITIALIZE-BIT-ARRAY	.....6	FASL-SETF-SYMBOL-FUNCTION	.....9
FASL-CHARACTER	.....7	FASL-INTEGERS	.....6	FASL-SHORT-INTEGERS	.....5
FASL-COMPLEX	.....6	FASL-INTERLISP-SYMBOL	.....7	FASL-STRUCTURE	.....9
FASL-CREATE-ARRAY	.....6	FASL-KEYWORD-SYMBOL	.....7	FASL-SYMBOL-IN-PACKAGE	.....7
FASL-DCODE	.....7	FASL-LARGE-INTEGERS	.....6	FASL-T	.....6
FASL-EVAL	.....9	FASL-LISP-SYMBOL	.....7	FASL-TABLE-FETCH	.....9
FASL-FAT-STRING	.....7	FASL-LIST	.....7	FASL-TABLE-STORE	.....9
FASL-FIND-PACKAGE	.....7	FASL-LIST*	.....7	FASL-THIN-STRING	.....7
FASL-FLOAT32	.....9	FASL-LOCAL-FN-FIXUPS	.....8	FASL-VECTOR	.....6
FASL-FUNCALL	.....9	FASL-NIL	.....6	FASL-VERIFY-TABLE-SIZE	.....9
FASL-INITIALIZE-ARRAY	.....6	FASL-RATIO	.....6		

---

**VARIABLE INDEX**

*BLOCK-LEVEL*	.....4	*VALUE-TABLE*	.....4	DEBUG-STREAM	.....4
*CURRENT-OPTABLE*	.....4	CHECK-TABLE-SIZE	.....2	INITIAL-VALUE-TABLE-SIZE	.....4
*DEFAULT-OPTABLE*	.....4	DEBUG-READER	.....4		

---

**CONSTANT INDEX**

CURRENT-VERSION	.....2	END-OF-DATA-MARK	.....2	SIGNATURE	.....2	VERSION-RANGE	.....2
END-MARK	.....2	FASL-EXTENDED	.....2	VALUE-TABLE-INCREMENT	....4		

---

**STRUCTURE INDEX**

FASL-ERROR	.....1	OBJECT-NOT-DUMPABLE	.....2	UNEXPECTED-END-OF-BLOCK	.....2
INCONSISTENT-TABLE	.....2	OPTABLE	.....2	UNIMPLEMENTED-OPCODE	.....2

---

**DEFINER INDEX**

DEFOP	.....3	DEFRANGE	.....3
-------	--------	----------	--------

---

**MACRO INDEX**

NEXT-VALUE	.....5	WITH-OPTABLE	.....4
------------	--------	--------------	--------

---

**PROPERTY INDEX**

IL:FASLOAD	.....10
------------	---------

---

**OPTIMIZER INDEX**

FIXUP-NTOFFSET	.....9
----------------	--------

---