File created: 24-Jun-88 16:37:57 **{POGO:AISNORTH:XEROX}<DSUNUNIX>KRIVACIC>TRANSLATOR>NATIVE-TRANSLATOR.;82**

changes to: (FNS MAKE-OPCODE-LIST MAKE-ORDERING-LIST ADD-FN-HEADER-INFO FN-CALL-PARSER LINK-OBJECT-CODE PVAR_PARSER MAKE-VAR-OFFSETS IL-ADD-FN-HEADER-INFO)
(VARS NATIVE-TRANSLATORCOMS)
(FUNCTIONS CODEBASELT2 CODEBASELT)

previous date: 22-Jun-88 19:17:38 {POGO:AISNORTH:XEROX}<DSUNUNIX>KRIVACIC>TRANSLATOR>NATIVE-TRANSLATOR.;65

Read Table: XCL

Package: NATIVE-TRANSLATOR

Format: XCCS

; Copyright (c) 1988 by Xerox Corporation. All rights reserved.

```
(RPAQQ NATIVE-TRANSLATORCOMS
       ((RECORDS BYTE-INFO-REC TRANSLATION-REC INFO-REC NATIVE-LINKER-INFO LINE-INFO-REC LINE-RECORD-INFO)
```

;;; Interface Functions

```
        (FNS BYTE-TO-NATIVE-TRANSLATE NATIVE-TO-BYTE-UNTRANSLATE LINK-OBJECT-CODE)
        (FNS NBT NATIVE-TRANS NATIVE-TRANSLATE)
        (FNS NUT FETCH-GCONST)
        (FNS LINK-C-CODE LINK-FN-CODE-BLOCK UNPACK-NUMBER)
```

;; Pass 1 Functions

```
        (FNS CODEWALK1 SETJUMPTARGET)
        (FUNCTIONS GETBYTE)
```

;; Pass 2 Functions

```
        (FNS CODEWALK2 CONDITIONAL-PARSER INLINE-EXPAND)
```

;; Parsing Functions

```
        (FNS BCE-PARSER STR-PARSER COND-PARSER CONST-PARSER COPY-PARSER JUMP-PARSER FN-CALL-PARSER
             FN-CALL-PARSERX ENVCALL-PARSER RETURN-PARSER SWAP-PARSER PVAR_PARSER)
```

;; Pattern Matching Routines

```
        (FNS PARM-SUBSTITUTE TOS-CHECK)
```

;; Output of Code lines

```
        (FNS ADD-CASE ADD-PUSH-OPERAND-LINE ADD-FN-HEADER-INFO IL-ADD-FN-HEADER-INFO CL-ADD-FN-HEADER-INFO)
```

;; Low Level Output of code lines

```
        (FNS ADD-LINE ADD-OPERAND-LINE ADD-LF ADD-ASM-LINE ADD-INLINE-LINE BCE-LINE)
```

;; Error Line Functions

```
        (FNS ADD-ERROR-LINE ADD-ERROR-ENTRY ADD-ERROR-SELECT)
```

;; Low Level Routines

```
        (FNS FIX-FILENAME PC-XFORM BCE-PC ENVCALL-FN-OBJECT FIND-FN0-OBJECTS CONST-POINTERP MAKE-VAR-OFFSETS)
```

;; Deferred Stack Funcions

```
        (FNS NEXT-OPERAND PUSH-ALL-OPERANDS OPERAND-PUSH OPERAND-GET OPERAND-POP GET-VAL GET-SHIFTED-VAL
             GET-INFO ADD-INFO SET-INFO)
```

;; Writeout Files

```
        (FNS MAKE-PROGRAM-FILE WRITE-PROGRAM-FILE WRITE-INLINE-FILE WRITE-INCLUDE-FILE PRINT-CODE-LINE
             PRINT-LINE-INFO)
```

;; Initialization

```
        (FNS TRANSLATION-INIT STRIP-ENDING-SLASH SETUP-TRANSLATION-FNS MAKE-TRANSLATION-ENTRY
             MAKE-TRANSLATION-ENTRIES MAKE-TRANSLATION-PATTERN-LIST MAKE-INLINE-LISTS MAKE-OPCODE-LIST
             MAKE-ORDERING-LIST)
```

;; Opcode Verification Fns

```
        (FNS VERIFY-OPCODES VERIFY-OPCODE)
```

;; New Code Block Fns

```
        (FNS LOADNATIVE GET-NATIVE-LOAD-SIZE LOAD-NATIVE-FILE SET-CODE-BASE MAKE-NEW-CODE-BLOCK
             SET-NEW-FUNCTION-DEF GET-FUNCTION-DEF SET-NATIVE-ADDR GET-NATIVE-ADDR LISP-ADDR-TO-NATIVE-ADDR
             NATIVE-ADDR-WORD-OFFSET WALK-CODE CODE-BLOCK-COPY ADD-GCONST MAKE-PC-OFFSET)
```

;; UNIX Exec Functions

```
        (FNS DO-EXEC-COMMAND TRAN-END-OF-UNIX-STREAM)
```

;; Macros

```
        (FUNCTIONS SWAPPED-FN-OBJ FN-OBJ CODEBASELT CODEBASELT2)
```

;; Variables

```
        (INITVARS (*NATIVE-TEMP-FILE-DIRECTORY* "/tmp")
               (*NATIVE-INCLUDE-FILE-DIRECTORY* NIL)
               (*NATIVE-LISP-RUN-FILENAME* NIL)
               (*NATIVE-BIN-DIRECTORY* "/bin")
               (*REMOVE-TEMP-NATIVE-FILES* T)
```

```
               (*UNIX-STREAMS* NIL)
               (*NATIVE-GCONST-OFFSET* 12)
               (*KEEP-NATIVE-SOURCES* NIL))
```

;; Makefile Environment

```
         (PROP (FILETYPE MAKEFILE-ENVIRONMENT)
               NATIVE-TRANSLATOR)))

(DECLARE\: EVAL@COMPILE

(DATATYPE BYTE-INFO-REC
       (PC OPCODE OPCODE-REC OP-NAME NEXT-BYTE-REC OPLENGTH JUMP-TARGET NEGATIVE-JUMP-TARGET JUMP-TO-ADDRESS
           ARG1 ARG2 ARG3 LEVEL-ADJUST STACK-EFFECT ENTRY-STACK-DEPTH CURRENT-STACK-MAX CURRENT-START-LEVEL
           ENTRY-ADDRESS OPCODE-PROPS))

(DATATYPE TRANSLATION-REC (MAY-UFN STACK-ADJUST STACK-ARGS PUSHING-RESULT DEFER-PUSH PATTERN TRANS-PATTERN
                                   TRANS-PARAMATERS PARSE-FN INLINE-EXIT-FN INLINE-EXPANSIONS POPPING-TOS))

(RECORD INFO-REC (POP-COUNT INFO-TYPE))

(BLOCKRECORD NATIVE-LINKER-INFO ((MACHINE-TYPE BITS 16)
                                 (MAGIC BITS 16)
                                 (TEXT-SIZE BITS 32)
                                 (DATA-SIZE BITS 32)
                                 (BSS-SIZE BITS 32)
                                 (SYMBOL-SIZE BITS 32)
                                 (ENTRY-POINT BITS 32)
                                 (TEXT-RELOCATION-SIZE BITS 32)
                                 (DATA-RELOCATION-SIZE BITS 32))
       (ACCESSFNS (RECORD-SIZE (PROGN 32)))
       (CREATE (PROGN (\\ALLOCBLOCK 8 UNBOXEDBLOCK.GCT 8 CELLSPERQUAD))))

(DATATYPE LINE-INFO-REC (PATTERN-LIST PARAMETER-LIST ACTUAL-PARAMETERS))

(DATATYPE LINE-RECORD-INFO (PREFIX-STRING LINE-INFO-LIST POSTFIX-STRING))
)

(/DECLAREDATATYPE 'BYTE-INFO-REC
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
                 POINTER POINTER POINTER POINTER POINTER POINTER)
```

;; ---field descriptor list elided by lister---

```
       '38)

(/DECLAREDATATYPE 'TRANSLATION-REC '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
                                     POINTER POINTER POINTER)
```

;; ---field descriptor list elided by lister---

```
       '24)

(/DECLAREDATATYPE 'LINE-INFO-REC '(POINTER POINTER POINTER)
```

;; ---field descriptor list elided by lister---

```
       '6)

(/DECLAREDATATYPE 'LINE-RECORD-INFO '(POINTER POINTER POINTER)
```

;; ---field descriptor list elided by lister---

```
       '6)
```

;;; Interface Functions

```
(DEFINEQ
```

(IL:BYTE-TO-NATIVE-TRANSLATE

```
  (LAMBDA (FNS)                                          ; Edited 10-Jun-88 14:33 by rtk
    (NBT FNS)))
```

(IL:NATIVE-TO-BYTE-UNTRANSLATE

```
  (LAMBDA (FNS)                                          ; Edited 10-Jun-88 14:39 by rtk
    (NUT FNS)))
```

(LINK-OBJECT-CODE

```
  (LAMBDA (FUNCTION-NAME OBJECT-FILE-NAME)
    (LINK-C-CODE FUNCTION-NAME OBJECT-FILE-NAME)))

)

(DEFINEQ
```

(NBT

```
  (LAMBDA (FUNCTION)                                     ; Edited 15-Jun-88 15:09 by rtk
    (COND
       ((NULL (AND IL:*NATIVE-INCLUDE-FILE-DIRECTORY* IL:*NATIVE-LISP-RUN-FILENAME*))
```

```
        (PRINTOUT T "You must setup: " T)
        (|if| (NULL IL:*NATIVE-INCLUDE-FILE-DIRECTORY*)
            |then| (PRINTOUT T "  IL:*NATIVE-INCLUDE-FILE-DIRECTORY*" T))
        (|if| (NULL IL:*NATIVE-LISP-RUN-FILENAME*)
            |then| (PRINTOUT T "  IL:*NATIVE-LISP-RUN-FILENAME*" T))
        (PRINTOUT T "before running the translator." T))
       ((LISTP FUNCTION)
        (LET ((RESULT-VALUE T))
             (|for| FN |in| FUNCTION |while| (SETQ RESULT-VALUE (NBT FN)))))
       (T (NATIVE-TRANS FUNCTION)))))
```

(NATIVE-TRANS

```
  (LAMBDA (|fn|)                                          ; Edited 21-Jun-88 18:14 by rtk
    (LET ((*FN-OBJECT-REMAP-LIST* NIL)
          (RE-TRANSLATE-LIST NIL)
          (TRANSLATE-RESULT NIL)
          (*NATIVE-STREAM* T)
          (|fn-to-translate| |fn|)
          |is-remapped-fn|)
         (DECLARE (SPECVARS *FN-OBJECT-REMAP-LIST* *NATIVE-STREAM*))
         (|repeatwhile| |fn-to-translate| |do| ;; Translate Next Fn
                        (SETQ TRANSLATE-RESULT (NATIVE-TRANSLATE |fn-to-translate| T T
                                                    (NEQ (MACHINETYPE)
                                                         'MAIKO)))
                        ;; Check the Translation Results
                        (|if| (LISTP TRANSLATE-RESULT)
                            |then| (|for| |fn-object-info| |in| TRANSLATE-RESULT
                                      |do| (|push| *FN-OBJECT-REMAP-LIST* (CONS |fn-object-info|
                                                                               NIL)))
                                   (|push| RE-TRANSLATE-LIST |fn-to-translate|)
                                   (|for| |item| |in| TRANSLATE-RESULT
                                      |do| (|push| RE-TRANSLATE-LIST |item|))
                              |elseif| (AND TRANSLATE-RESULT (SETQ |is-remapped-fn| (ASSOC
                                                                                    |fn-to-translate|
                                                                                    *FN-OBJECT-REMAP-LIST*
                                                                                    )))
                            |then| (RPLACD |is-remapped-fn| TRANSLATE-RESULT))
                        (SETQ |fn-to-translate| (AND TRANSLATE-RESULT RE-TRANSLATE-LIST
                                                     (|pop| RE-TRANSLATE-LIST))))
         TRANSLATE-RESULT)))
```

(NATIVE-TRANSLATE

```
  (LAMBDA (|fn| |make-the-file| |set-native| |keep-cr-eol|)      ; Edited 21-Jun-88 20:51 by rtk
    (DECLARE (SPECIAL *NATIVE-STREAM*))
    (LET (*ENTRY-POINTS* *CODE-BASE* *START-PC* *FN-NAME* *CODE-SIZE* *NUMBER-OF-ARGS* *PVAR-QUAD-SIZE*
          *STACK-MIN-SIZE* *INLINES* *EVAL-STACK* *ERROR-STACK* *ERROR-CASES* *MATCHFOUND*
          *TRANSLATION-TABLE* *BYTE-INFO-TABLE* *ENTRY-ADDRS* (*ENTRY-POINT-MAX* 0)
          (*SHOW-INLINE* NIL)
          (*SHOW-BYTECODES* T)
          (*TARGET-MACHINE* 'SUN3)
          (*TARGET-MACHINE.N* 'SUN3.N)
          (|file-name| (FIX-FILENAME |fn|))
          (*GCONST-PTRS* NIL)
          PC-OFFSET OLD-CODE-BASE (*DEBUG-TRANSLATOR* (OR (NEQ 'MAIKO (MACHINETYPE))
                                                          (BOUNDP '*NATIVE-TRANSLATOR-DEBUG*)))
          (*SAW-ENVCALL* NIL)
          (*REPLACEMENT-STRING* NIL)
          (NEW-STKMIN NIL)
          (*VAR-OFFSETS* NIL)
          (GCONST-OFFSET *NATIVE-GCONST-OFFSET*))
         (DECLARE (SPECVARS *TARGET-MACHINE* *TARGET-MACHINE.N* *ENTRY-POINTS* *CODE-BASE* *START-PC* *FN-NAME*
                            *CODE-SIZE* *NUMBER-OF-ARGS* *PVAR-QUAD-SIZE* *STACK-MIN-SIZE* *INLINES* *EVAL-STACK*
                            *ERROR-STACK* *ERROR-CASES* *MATCHFOUND* *TRANSLATION-TABLE* *BYTE-INFO-TABLE*
                            *SHOW-INLINE* *SHOW-BYTECODES* *ENTRY-ADDRS* *DEBUG-TRANSLATOR* *GCONST-PTRS*
                            *ENTRY-POINT-MAX* *SAW-ENVCALL* *REPLACEMENT-STRING* *VAR-OFFSETS*))
         (|if| (NULL (FMEMB (MACHINETYPE)
                            '(MAIKO KATANA)))
             |then| (SETQ *NATIVE-TEMP-FILE-DIRECTORY* ""))
         (TRANSLATION-INIT)
         (SETQ *CODE-BASE* (SET-CODE-BASE |fn|))
         (AND (NEQ 0 (LOGAND (|fetch| (FNHEADER STARTPC) |of| *CODE-BASE*)
                             1))
              (ERROR "Illegal Start Pc, Cannot Translate " |fn|))
         (SETQ PC-OFFSET (MAKE-PC-OFFSET *CODE-BASE*))
         (SETQ *VAR-OFFSETS* (MAKE-VAR-OFFSETS *CODE-BASE*))
         (SETQ NEW-STKMIN (CODEWALK1 (+ PC-OFFSET GCONST-OFFSET)))
         (|if| (|fetch| (FNHEADER NATIVE) |of| *CODE-BASE*)
             |then| (PRINTOUT *NATIVE-STREAM* "Already Native Code for " |fn| T)
           |else| (LET ((|re-map-list| (FIND-FN0-OBJECTS *SAW-ENVCALL*)))
                       (|if| |re-map-list|
                           |then| |re-map-list|
                         |else| (SETQ OLD-CODE-BASE *CODE-BASE*)
```

```
                              (MAKE-NEW-CODE-BLOCK |fn| PC-OFFSET NEW-STKMIN GCONST-OFFSET)
                              (SETUP-TRANSLATION-FNS)
                              (|if| *DEBUG-TRANSLATOR*
                                  |then| (SETQ CODE-BASE *CODE-BASE*)
                                         (SETQ EP *ENTRY-POINTS*))
                              (CODEWALK2 |fn| |file-name| GCONST-OFFSET (+ PC-OFFSET GCONST-OFFSET))
                              (|if| |make-the-file|
                                  |then| (MAKE-PROGRAM-FILE |file-name| |keep-cr-eol|))
                              (|if| |set-native|
                                  |then| (LOADNATIVE |fn| |file-name| |file-name| *CODE-BASE* OLD-CODE-BASE)
                                |else| *CODE-BASE*)))))))

)

(DEFINEQ
```

(NUT

```
  (LAMBDA (FN)                                                    ; Edited 13-Jun-88 11:49 by rtk
    (IF (LISTP FN)
        THEN (MAPCAR FN 'NUT)
      ELSE (LET* ((CODE-BASE (SET-CODE-BASE FN))
                  (OLD-CODE-BLOCK (AND CODE-BASE (OR (|fetch| (FNHEADER NATIVE) |of| CODE-BASE)
                                                     (ERROR FN "not native  code"))
                                       (FETCH-GCONST CODE-BASE 6))))
                 (AND OLD-CODE-BLOCK (SET-NEW-FUNCTION-DEF FN OLD-CODE-BLOCK T))
                 T))))
```

(FETCH-GCONST

```
  (LAMBDA (FN-OBJ OFFSET)                                         ; Edited 31-May-88 17:46 by rtk
    (LET* ((BYTE-OFFSET (+ (|fetch| (FNHEADER STARTPC) |of| FN-OBJ)
                           OFFSET))
           (HI-BLOCK (\\GETBASEBYTE FN-OBJ (+ BYTE-OFFSET 1)))
           (LO-BLOCK1 (\\GETBASEBYTE FN-OBJ (+ BYTE-OFFSET 2)))
           (LO-BLOCK2 (\\GETBASEBYTE FN-OBJ (+ BYTE-OFFSET 3)))
           (LO-BLOCK (LOGOR (LLSH LO-BLOCK1 8)
                            LO-BLOCK2)))
          (VAG2 HI-BLOCK LO-BLOCK))))

)

(DEFINEQ
```

(LINK-C-CODE

```
  (LAMBDA (LISP-FN-NAME C-CODE-FILE-NAME ENTRY-PT-NAME)           ; Edited 17-Jun-88 18:14 by rtk
    (LET* ((*NATIVE-STREAM* T)
           (SOURCE-FN-OBJ (SET-CODE-BASE 'LINK-FN-CODE-BLOCK))
           (STARTPC (|fetch| (FNHEADER STARTPC) |of| SOURCE-FN-OBJ))
           (DEST-FN-OBJ (CODE-BLOCK-COPY SOURCE-FN-OBJ STARTPC (+ 8 STARTPC)
                                         0
                                         (+ STARTPC (MAKE-PC-OFFSET SOURCE-FN-OBJ))
                                         *NATIVE-GCONST-OFFSET* NIL LISP-FN-NAME)))
          (LET* ((FULL-FILE-NAME-NO-BRACKETS (CONCAT *NATIVE-TEMP-FILE-DIRECTORY* "/" (FIX-FILENAME LISP-FN-NAME
                                                                                                    )))
                 (FILE-SIZE 0)
                 (NATIVE-CODE-BLOCK-PTR 0)
                 (NATIVE-CODE-ADDR 0)
                 (HEX-LOAD-ADDR 0)
                 (LOAD-FILE-SIZE 0)
                 (RET-TO-DISPATCH (OR (AND (EQ (MACHINETYPE)
                                               'MAIKO)
                                           (SUBRCALL NATIVE-MEMORY-REFERENCE 2 0))
                                      0))
                 (NATIVE-CODE-INSERT-BYTES (CONS 72 (CONS 121 (APPEND (UNPACK-NUMBER RET-TO-DISPATCH 4)
                                                                      (UNPACK-NUMBER 20081 2)
                                                                      (LIST 4 14 15 9)))))
                 (NATIVE-CODE-OFFSET (+ (LENGTH NATIVE-CODE-INSERT-BYTES)
                                        4)))
                (AND (OR (AND (EQ (MACHINETYPE)
                                  'MAIKO)
                              ;; Execute the Unix C Compiler
                              ;; Get the Object File SIze
                              (SETQ FILE-SIZE (GET-NATIVE-LOAD-SIZE (CONCAT "{UNIX}" C-CODE-FILE-NAME)))
                              ;; Allocate a block big enough to hold the object
                              (SETQ NATIVE-CODE-BLOCK-PTR (\\ALLOCBLOCK (FOLDHI (+ NATIVE-CODE-OFFSET FILE-SIZE)
                                                                                  BYTESPERCELL)
                                                                        UNBOXEDBLOCK.GCT CELLSPERQUAD CELLSPERQUAD))
                              (SETQ NATIVE-CODE-ADDR (LISP-ADDR-TO-NATIVE-ADDR NATIVE-CODE-BLOCK-PTR))
                              ;; Execute the Unix Linker
                                                                  (* DO-EXEC-COMMAND (CL:FORMAT NIL "~a/ld -N -s  -e _~a
                                                                  -Ttext ~x -A ~a ~a -o ~a -lc" *NATIVE-BIN-DIRECTORY*
                                                                  (OR ENTRY-PT-NAME LISP-FN-NAME)
                                                                  (+ NATIVE-CODE-OFFSET NATIVE-CODE-ADDR)
                                                                  *NATIVE-LISP-RUN-FILENAME* C-CODE-FILE-NAME
                                                                  FULL-FILE-NAME-NO-BRACKETS))
```

```
                    (DO-EXEC-COMMAND (CL:FORMAT NIL "~a/ld -N -s   -Ttext ~x -A ~a ~a -o ~a -lc"
                                                *NATIVE-BIN-DIRECTORY* (+ NATIVE-CODE-OFFSET
                                                                          NATIVE-CODE-ADDR)
                                                *NATIVE-LISP-RUN-FILENAME* C-CODE-FILE-NAME
                                                FULL-FILE-NAME-NO-BRACKETS))
                    (PROGN (PRINTOUT *NATIVE-STREAM* "Load " C-CODE-FILE-NAME " At " NATIVE-CODE-ADDR
                                   " for " FILE-SIZE " bytes." T)
                           T)
                    ;; Load the code into lisp space
                    (SETQ NATIVE-ENTRY-ADDR (LOAD-NATIVE-FILE FULL-FILE-NAME-NO-BRACKETS
                                                              NATIVE-CODE-BLOCK-PTR NATIVE-CODE-INSERT-BYTES)))
                ;; Allocate a dummy block if not maiko
                (SETQ NATIVE-CODE-BLOCK-PTR (\\ALLOCBLOCK (FOLDHI *CODE-SIZE* BYTESPERCELL)
                                                          UNBOXEDBLOCK.GCT CELLSPERQUAD CELLSPERQUAD)))
            ;; Set Native Adder in Fn Object
            (PROGN (PRINTOUT T "set native addr" T)
                   T)
            (SET-NATIVE-ADDR LISP-FN-NAME DEST-FN-OBJ NATIVE-CODE-BLOCK-PTR NATIVE-CODE-ADDR
                   NATIVE-ENTRY-ADDR)
            ;; Add the New GCONST  xx POP opcodes
            (PROGN (PRINTOUT T "ADD-GCONST" T)
                   T)
            (ADD-GCONST DEST-FN-OBJ 0 NATIVE-CODE-BLOCK-PTR)
            ;; Set the New Function Definition
            (PROGN (PRINTOUT T "SET-NEW-FUNCTION-DEF" T)
                   T)
            (SET-NEW-FUNCTION-DEF LISP-FN-NAME DEST-FN-OBJ T)
           DEST-FN-OBJ)))))
```

(LINK-FN-CODE-BLOCK

```
  (LAMBDA (DUMMY)
    NIL))
```

(UNPACK-NUMBER

```
  (LAMBDA (NUMBER SIZE)                                          ; Edited  1-Jun-88 11:50 by rtk
    (|for| I |from| (SUB1 SIZE) |to| 0 |by| -1 |collect| (LOGAND 255 (LRSH NUMBER (TIMES 8 I))))))

)
```

;; Pass 1 Functions

```
(DEFINEQ
```

(CODEWALK1

```
  (LAMBDA (PC-OFFSET)                                            ; Edited 21-Jun-88 20:26 by rtk
```

;;; This Pass identifies jump targets, sets jump addresses, identifies following opcodes, and other information used in the 2nd pass.

```
    (DECLARE (SPECIAL *ENTRY-POINTS* *CODE-SIZE* *START-PC* *FN-NAME* *CODE-SIZE* *NUMBER-OF-ARGS*
                      *PVAR-QUAD-SIZE* *STACK-MIN-SIZE* *CODE-BASE* *GCONST-PTRS* *SAW-ENVCALL*
                      *FN-OBJECT-REMAP-LIST* *DEBUG-TRANSLATOR*))
    (LET (TAG OP# (STACK-DEPTH 0)
              (MAX-PUSH-COUNT 0)
              (START-LEVEL 0)
              (STACK-HEADER-OVERHEAD 6)
              (SAFTEY-SLOTS 16))
         (SETQ *START-PC* (|fetch| (FNHEADER STARTPC) |of| *CODE-BASE*))
         (SETQ *FN-NAME* (|fetch| (FNHEADER FRAMENAME) |of| *CODE-BASE*))
         (SETQ *NUMBER-OF-ARGS* (|fetch| (FNHEADER NA) |of| *CODE-BASE*))
         (SETQ *PVAR-QUAD-SIZE* (|fetch| (FNHEADER PV) |of| *CODE-BASE*))
         (SETQ *STACK-MIN-SIZE* (|fetch| (FNHEADER STKMIN) |of| *CODE-BASE*))
         (SETQ STACK-DEPTH (IPLUS (ITIMES (ADD1 *PVAR-QUAD-SIZE*)
                                          CELLSPERQUAD)
                                  (MAX *NUMBER-OF-ARGS* 0)
                                  STACK-HEADER-OVERHEAD SAFTEY-SLOTS))
         (SETQ START-LEVEL STACK-DEPTH)
         (PRINTOUT *NATIVE-STREAM* "Translation Pass 1: " *FN-NAME* T)
         (PROG ((CODELOC *START-PC*)
                B B1 B2 B3 LEN PC LEVADJ STACK-EFFECT STK NEW-REC LAST-REC)
           LP   (SETQ PC CODELOC)
                (SETQ LEN (LOCAL (|fetch| OPNARGS |of| (SETQ TAG (\\FINDOP (SETQ B (GETBYTE *CODE-BASE*)))))))
                (SETQ OP# (|fetch| OP# |of| TAG))
                (SETQ LEVADJ (|fetch| LEVADJ |of| TAG))
                (LET ((ELT-CACHE (ELT *ENTRY-POINTS* (IPLUS PC-OFFSET PC))))
                     ;; Stack Depth is Unknown (there must only be a backward jump to here
                     (|if| (NULL STACK-DEPTH)
                         |then| (SETQ STACK-DEPTH 0)
                                (SETQ START-LEVEL 0))
```

```
      ;; There was a forward jump to this location, compare stack depth
      (|if| (AND ELT-CACHE)
          |then|                                          (* |if| (AND (NEQ (CAR ELT-CACHE) STACK-DEPTH)
                                                          (NEQ 0 STACK-DEPTH) *DEBUG-TRANSLATOR*) |then|
                                                          (PRINTOUT T "UnEqual Levels at "
                                                          (IPLUS PC-OFFSET PC) T) (PRINTOUT T "Old: " ELT-CACHE
                  ", New: " STACK-DEPTH T))
                  (SETQ STACK-DEPTH (MAX (CAR ELT-CACHE)
                                         STACK-DEPTH))
                  (SETQ START-LEVEL (MIN (CADDR ELT-CACHE)
                                         START-LEVEL))
                  (SETQ MAX-PUSH-COUNT (MAX (CADR ELT-CACHE)
                                            MAX-PUSH-COUNT)))
      (SETQ MAX-PUSH-COUNT (MAX (- STACK-DEPTH START-LEVEL)
                                MAX-PUSH-COUNT))
      ;; Make the new record
      (SETQ NEW-REC
       (|create| BYTE-INFO-REC
              PC _ (+ PC PC-OFFSET)
              OPCODE _ B
              OPCODE-REC _ TAG
              LEVEL-ADJUST _ LEVADJ
              OPLENGTH _ LEN
              JUMP-TARGET _ ELT-CACHE
              ENTRY-STACK-DEPTH _ STACK-DEPTH
              CURRENT-STACK-MAX _ MAX-PUSH-COUNT
              CURRENT-START-LEVEL _ START-LEVEL
              ARG1 _ 0
              ARG2 _ 0
              ARG3 _ 0)))
(AND LAST-REC (|replace| (BYTE-INFO-REC NEXT-BYTE-REC) |of| LAST-REC |with| NEW-REC))
(SETQ LAST-REC NEW-REC)
(COND
   ((IGREATERP LEN 0)
    (SETQ B1 (GETBYTE *CODE-BASE*))
    (|replace| (BYTE-INFO-REC ARG1) |of| NEW-REC |with| B1)))
(COND
   ((IGREATERP LEN 1)
    (SETQ B2 (GETBYTE *CODE-BASE*))
    (|replace| (BYTE-INFO-REC ARG2) |of| NEW-REC |with| B2)))
(COND
   ((IGREATERP LEN 2)
    (SETQ B3 (GETBYTE *CODE-BASE*))
    (|replace| (BYTE-INFO-REC ARG3) |of| NEW-REC |with| B3)))
(|replace| (BYTE-INFO-REC OP-NAME) |of| NEW-REC |with| (OR (|fetch| OPCODENAME |of| TAG)
                                                           (|fetch| OPPRINT |of| TAG)))
(SETQ STACK-EFFECT (COND
                       ((NUMBERP LEVADJ)
                        LEVADJ)
                       ((FMEMB LEVADJ '(NCJUMP CJUMP))
                        -1)
                       ((EQ 'JUMP LEVADJ)
                        0)
                       ((EQ 'FNX LEVADJ)
                        (MINUS (SUB1 B1)))
                       ((LISTP LEVADJ)
                        (COND
                           ((EQ 'POP.N LEVADJ)
                            (MINUS B1))
                           ((EQ 'UNWIND LEVADJ)
                            0)
                           ((EQ 'JUMP LEVADJ)
                            0)
                           ((FMEMB (|fetch| OPCODENAME |of| TAG)
                                   '(UBFLOAT1 UBFLOAT2 UBFLOAT3))
                            (CAR LEVADJ))
                           ((FMEMB (|fetch| OPCODENAME |of| TAG)
                                   '(DOVEMISC))
                            (NTH LEVADJ B1))
                           (T 0)))
                       (T 0)))
(|replace| (BYTE-INFO-REC STACK-EFFECT) |of| NEW-REC |with| STACK-EFFECT)
(SETA *ENTRY-POINTS* (+ PC PC-OFFSET)
      NEW-REC)
(COND
   ((LISTP OP#)
    (SETQ OP# (CAR OP#))))
(SELECTQ (OR (AND (NEQ T (|fetch| OPPRINT |of| TAG))
                  (|fetch| OPPRINT |of| TAG))
             (|fetch| OPCODENAME |of| TAG))
    (-X- (SETQ *CODE-SIZE* (IPLUS CODELOC 5))     (* |if| *DEBUG-TRANSLATOR* |then|
                                                  (PRINTOUT T "My stack min: " (TIMES 2 MAX-PUSH-COUNT)
         " Given: " (|fetch| (FNHEADER STKMIN) |of| *CODE-BASE*) T))
         (RETURN (MAX (TIMES 2 MAX-PUSH-COUNT)
                      (|fetch| (FNHEADER STKMIN) |of| *CODE-BASE*))))
    (GCONST (LET* ((|const-ptr| (VAG2 B1 (LOGOR (LLSH B2 8)
```

```
                                                    B3)))
                                  (|remap-info| (FASSOC |const-ptr| *FN-OBJECT-REMAP-LIST*))
                                  (|remap-datum| (AND |remap-info| (CDR |remap-info|)))))
                                 (|if| |remap-datum|
                                     |then| (|push| *GCONST-PTRS* (LIST |remap-datum| (+ PC PC-OFFSET)
                                                                       |const-ptr|))
                                   |else| (|push| *GCONST-PTRS* (LIST |const-ptr| (+ PC PC-OFFSET)
                                                                     NIL)))))
                (JUMP (|if| (SETJUMPTARGET (IPLUS (IDIFFERENCE B OP#)
                                                   2)
                                           CODELOC NEW-REC STACK-EFFECT PC-OFFSET)
                           |then| (SETQ STACK-DEPTH NIL)))
                (JUMPX (|if| (SETJUMPTARGET (COND
                                              ((IGEQ B1 128)
                                               (IDIFFERENCE B1 256))
                                              (T B1))
                                            CODELOC NEW-REC (|if| (EQ (|fetch| LEVADJ |of| TAG)
                                                                      'NCJUMP)
                                                                |then| 0
                                                               |else| STACK-EFFECT)
                                            PC-OFFSET)
                           |then| (SETQ STACK-DEPTH NIL)))
                (JUMPXX (|if| (SETJUMPTARGET (IPLUS (LLSH B1 8)
                                                    B2
                                                    (COND
                                                       ((IGREATERP B1 127)
                                                        -65536)
                                                       (T 0)))
                                             CODELOC NEW-REC STACK-EFFECT PC-OFFSET)
                           |then| (SETQ STACK-DEPTH NIL)))
                (RETURN (SETQ STACK-DEPTH NIL))
                (\\RETURN (SETQ STACK-DEPTH NIL))
                (ENVCALL (|push| *SAW-ENVCALL* NEW-REC))
                NIL)
          (SETQ STACK-DEPTH (+ STACK-DEPTH (COND
                                             ((OR (EQ LEVADJ 'JUMP)
                                                  (AND (LISTP LEVADJ)
                                                       (EQ (CAR LEVADJ)
                                                           'JUMP)))
                                              (SETQ STACK-DEPTH NIL)
                                              (GO LP))
                                             (T STACK-EFFECT))))
          (|if| (EQ 0 (LOGAND 15 PC))
              |then| (BLOCK))
          (GO LP)))))
```

(SETJUMPTARGET

```
(LAMBDA (N CODELOC BYTE-REC THIS-STACK-EFFECT PC-OFFSET)       ; Edited 6-Jun-88 15:18 by rtk
  (LET* ((TARGET (+ N (IDIFFERENCE CODELOC (ADD1 (|fetch| (BYTE-INFO-REC OPLENGTH) |of| BYTE-REC)))
                    PC-OFFSET))
         (JUMP-TARGET-INFO (ELT *ENTRY-POINTS* TARGET))
         (JUMP-EXIT-STACK-DEPTH (IPLUS (|fetch| (BYTE-INFO-REC ENTRY-STACK-DEPTH) |of| BYTE-REC)
                                       THIS-STACK-EFFECT))
         (FORWARD-JUMP (GREATERP TARGET (+ CODELOC PC-OFFSET)))
         (TARGET-ENTRY-STACK-DEPTH (AND JUMP-TARGET-INFO (OR (AND (NOT FORWARD-JUMP)
                                                                  (|fetch| (BYTE-INFO-REC ENTRY-STACK-DEPTH)
                                                                     |of| JUMP-TARGET-INFO))
                                                             (AND FORWARD-JUMP (CAR JUMP-TARGET-INFO))))))
        (|replace| (BYTE-INFO-REC JUMP-TO-ADDRESS) |of| BYTE-REC |with| TARGET)
        (COND
           (FORWARD-JUMP
                  ;; Forward JUMP (which has already been referenced)
                  (DESTRUCTURING-BIND (TARGET-ENTRY-DEPTH TARGET-MAX-DEPTH TARGET-START-LEVEL)
                         (OR JUMP-TARGET-INFO (LIST JUMP-EXIT-STACK-DEPTH (|fetch| (BYTE-INFO-REC
                                                                                     CURRENT-STACK-MAX)
                                                                                |of| BYTE-REC)
                                                    (|fetch| (BYTE-INFO-REC CURRENT-START-LEVEL) |of| BYTE-REC)))
                             (* |if| (NEQ TARGET-ENTRY-DEPTH
                             JUMP-EXIT-STACK-DEPTH) |then|
                             (PRINTOUT T "UnEqual Stack Depth Jump from: "
                             (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC) T)
                             (PRINTOUT T "JUMP: " (|fetch| (BYTE-INFO-REC
                             OPCODE-REC) |of| BYTE-REC) T)
                             (PRINTOUT T "Target JUMP level: " JUMP-TARGET-INFO T))
                         (SETA *ENTRY-POINTS* TARGET (LIST (MAX TARGET-ENTRY-DEPTH JUMP-EXIT-STACK-DEPTH)
                                                           (MAX TARGET-MAX-DEPTH (|fetch| (BYTE-INFO-REC
                                                                                            CURRENT-STACK-MAX
                                                                                             )
                                                                                       |of| BYTE-REC))
                                                           (MIN TARGET-START-LEVEL (|fetch| (BYTE-INFO-REC
                                                                                              CURRENT-START-LEVEL
                                                                                               )
                                                                                      |of| BYTE-REC))))))
           (T ;; A backwards JUMP
```

```
                (|replace| (BYTE-INFO-REC JUMP-TARGET) |of| JUMP-TARGET-INFO |with| T)
                (|replace| (BYTE-INFO-REC NEGATIVE-JUMP-TARGET) |of| JUMP-TARGET-INFO |with| T)
                                                                (* |if| (NEQ JUMP-EXIT-STACK-DEPTH
                                                                TARGET-ENTRY-STACK-DEPTH) |then|
                                                                (PRINTOUT T "UnEqual Stack Depth, jump from: "
                                                                (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC) T)
                                                                (PRINTOUT T "JUMP[" JUMP-EXIT-STACK-DEPTH "]: "
                                                                (|fetch| (BYTE-INFO-REC OPCODE-REC) |of| BYTE-REC) T)
                                                                (PRINTOUT T "TO[" TARGET-ENTRY-STACK-DEPTH "]: "
                                                                (|fetch| (BYTE-INFO-REC OPCODE-REC) |of|
                                                                JUMP-TARGET-INFO) T))
                ))
           (SELECTQ (|fetch| (BYTE-INFO-REC LEVEL-ADJUST) |of| BYTE-REC)
                (NCJUMP NIL)
                (CJUMP NIL)
                T))))

)


(DEFMACRO GETBYTE (BASE)
   `(\\GETBASEBYTE ,BASE (PROG1 CODELOC (|add| CODELOC 1))))

;; Pass 2 Functions

(DEFINEQ

(CODEWALK2
  (LAMBDA (FN ENTRY-NAME GCONST-OFFSET PC-ADJUST-SIZE)          ; Edited 21-Jun-88 20:26 by rtk

;;; 2nd pass which generates the code from the ENTRY-POINTS array generated in pass1

    (DECLARE (SPECIAL *ENTRY-POINTS* *CODE-SIZE* *START-PC* *CPROGRAM* *INLINES* *EVAL-STACK* *ERROR-STACK*
                    *ERROR-CASES* *TRANSLATION-TABLE* *TARGET-MACHINE.N* *TARGET-MACHINE*))
    (PRINTOUT *NATIVE-STREAM* "Translation Pass 2: " FN T)
    (SETQ *CPROGRAM* NIL)
    (SETQ *INLINES* NIL)
    (SETQ *EVAL-STACK* NIL)
    (SETQ *ERROR-CASES* NIL)
    (SETQ *ERROR-STACK* NIL)
    (ADD-FN-HEADER-INFO FN ENTRY-NAME)
    (LET
     ((*PREV-UFN* T)
      *IGNORE-JUMP* *IGNORE-THIS-JUMP*)
     (DECLARE (SPECVAR *PREV-UFN* *IGNORE-JUMP* *IGNORE-THIS-JUMP*))
     (|bind| (\i _ (+ *START-PC* GCONST-OFFSET)) |while| (GEQ *CODE-SIZE* \i)
        |do| (SETQ *IGNORE-THIS-JUMP* *IGNORE-JUMP*)
             (SETQ *IGNORE-JUMP* NIL)
             (SETQ *ERROR-STACK* NIL)
             (LET* ((*ARG-COUNT* 0)
                    (*INFO-REC* NIL)
                    (*ERROR-PC* NIL)
                    (*INLINE-ERROR-STACK* NIL)
                    (*PC-BUMP-SIZE* 1)
                    (BYTE-REC (ELT *ENTRY-POINTS* \i))
                    (OPCODE (|fetch| (BYTE-INFO-REC OPCODE) |of| BYTE-REC))
                    (TRANS-REC (ELT *TRANSLATION-TABLE* OPCODE)))
                   (DECLARE (SPECVARS *PC-BUMP-SIZE* *ARG-COUNT* *ERROR-PC* *INLINE-ERROR-STACK* *INFO-REC*))
                   (|if| (ZEROP OPCODE)
                       |then| (ADD-LINE (CONCAT "/* exit " ENTRY-NAME " */ } "))
                              (RETURN))
                   (ADD-CASE (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC)
                          *PREV-UFN*
                          (|fetch| (BYTE-INFO-REC JUMP-TARGET) |of| BYTE-REC)
                          (|fetch| (BYTE-INFO-REC NEGATIVE-JUMP-TARGET) |of| BYTE-REC)
                          (|fetch| (TRANSLATION-REC MAY-UFN) |of| TRANS-REC))
                   (|if| TRANS-REC
                       |then| (LET* ((|inline-expansions| (|fetch| (TRANSLATION-REC INLINE-EXPANSIONS) |of| TRANS-REC)
                                                         )
                                     |expansion|
                                     (|parsed-line| (COND
                                                        ((AND |inline-expansions|
                                                              (SETQ |expansion|
                                                                    (OR (AND (CL:MULTIPLE-VALUE-BIND (MDATA MINFO)
                                                                                 (OPERAND-GET)
                                                                                 (EQ 'SMALL-CONST (|fetch| (INFO-REC
                                                                                                                INFO-TYPE)
                                                                                                           |of| MINFO)))
                                                                             (LISTGET |inline-expansions|
                                                                                    *TARGET-MACHINE.N*))
                                                                        (LISTGET |inline-expansions| *TARGET-MACHINE*))))
                                                         (INLINE-EXPAND TRANS-REC BYTE-REC |expansion|))
                                                        (T (APPLY* (|fetch| (TRANSLATION-REC PARSE-FN) |of| TRANS-REC)
                                                                   TRANS-REC BYTE-REC FN NIL PC-ADJUST-SIZE NIL)))))
                                    (|if| |parsed-line|
                                        |then| (CONDITIONAL-PARSER TRANS-REC BYTE-REC FN |parsed-line|)
                                               (ADD-ERROR-ENTRY TRANS-REC BYTE-REC)))
```

```
                    (SETQ *PREV-UFN* (|fetch| (TRANSLATION-REC MAY-UFN) |of| TRANS-REC))
                |else| (CONDITIONAL-PARSER TRANS-REC BYTE-REC FN (BCE-PARSER \i BYTE-REC))
                    (SETQ *PREV-UFN* T))
            (SETQ \i (IPLUS \i (|fetch| (BYTE-INFO-REC OPLENGTH) |of| BYTE-REC)
                            *PC-BUMP-SIZE*))
            (|if| (EQ 0 (LOGAND 15 \i))
                |then| (BLOCK)))))))
```

(CONDITIONAL-PARSER

```
(LAMBDA (TRANS-REC BYTE-REC FN-NAME |operand-string|)          ; Edited 21-Jun-88 22:42 by rtk
  (DECLARE (SPECIAL *IGNORE-JUMP* *INFO-REC*))
```

;; See if Next OPCODE is a Jump.

;; IF so we can combine the condition & the Jump to avoid the Push & Pop

```
  (LET* ((|next-rec| (|fetch| (BYTE-INFO-REC NEXT-BYTE-REC) |of| BYTE-REC))
         (|next-opcode-rec| (|fetch| (BYTE-INFO-REC OPCODE-REC) |of| |next-rec|))
         (|next-next-rec| (|fetch| (BYTE-INFO-REC NEXT-BYTE-REC) |of| |next-rec|)))
        (COND
           ((AND (EQ (|fetch| (BYTE-INFO-REC LEVEL-ADJUST) |of| |next-rec|)
                     'CJUMP)
                 (|fetch| (TRANSLATION-REC PUSHING-RESULT) |of| TRANS-REC)
                 (NOT (|fetch| (BYTE-INFO-REC JUMP-TARGET) |of| |next-rec|)))
            (PUSH-ALL-OPERANDS)
            (IF (TYPEP |operand-string| 'LINE-INFO-REC)
                THEN (ADD-OPERAND-LINE (|if| (FMEMB (|fetch| OPCODENAME |of| |next-opcode-rec|)
                                                    '(TJUMP TJUMPX))
                                               |then| "    if ((" 
                                               |else| "       if (!(")
                                           |operand-string|
                                           (CONCAT ")) {goto pc" (|fetch| (BYTE-INFO-REC JUMP-TO-ADDRESS) |of| |next-rec|)
                                                   "; }"))
                  ELSE (ADD-OPERAND-LINE (|if| (FMEMB (|fetch| OPCODENAME |of| |next-opcode-rec|)
                                                      '(TJUMP TJUMPX))
                                                 |then| "       if ((" 
                                                 |else| "          if (!(")
                                             NIL
                                             (CONCAT |operand-string| ")) {goto pc" (|fetch| (BYTE-INFO-REC JUMP-TO-ADDRESS)
                                                                                            |of| |next-rec|)
                                                     "; }")))
            (|if| (|fetch| (TRANSLATION-REC MAY-UFN) |of| TRANS-REC)
                |then| (ADD-LINE (CONCAT "goto pc" (|fetch| (BYTE-INFO-REC PC) |of| |next-next-rec|)
                                         ";")
                                 T)
                |else| (SETQ *IGNORE-JUMP* T))
            (|replace| (BYTE-INFO-REC JUMP-TARGET) |of| |next-next-rec| |with| T))
           (T (|if| (|fetch| (TRANSLATION-REC DEFER-PUSH) |of| TRANS-REC)
                    |then| (OPERAND-PUSH |operand-string| *INFO-REC*)
                    |else| (|if| (OR (|fetch| (TRANSLATION-REC MAY-UFN) |of| TRANS-REC)
                                     (|fetch| (TRANSLATION-REC PUSHING-RESULT) |of| TRANS-REC))
                               |then| (PUSH-ALL-OPERANDS)
                               (|if| (|fetch| (TRANSLATION-REC PUSHING-RESULT) |of| TRANS-REC)
                                   |then| (SETQ *ERROR-STACK* (ADD-PUSH-OPERAND-LINE |operand-string| *ERROR-STACK*
                                                                                     *INFO-REC*))
                                    |else| (ADD-OPERAND-LINE "      " |operand-string| ";")))))))))
```

(INLINE-EXPAND

```
(LAMBDA (TRANS-REC BYTE-REC EXPANSION-LIST)                    ; Edited 21-Jun-88 23:44 by rtk
  (DECLARE (SPECIAL *ERROR-STACK* *ERROR-PC* *INLINE-ERROR-STACK* *REPLACEMENT-STRING*))
  (LET* ((*TOS-INFO* NIL)
         (*TOS-VAL* NIL)
         (|pc| (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC))
         (|may-ufn| (|fetch| (TRANSLATION-REC MAY-UFN) |of| TRANS-REC))
         (|inline-name| (CONCAT "_" (CAAAR EXPANSION-LIST)
                                |pc|))
         (|error-pc| (CONCAT "errorpc" |pc|))
         (|argcount| 0)
         (\n (|fetch| (BYTE-INFO-REC ARG1) |of| BYTE-REC))
         (|new-lines| NIL)
         (TOS-CALL-VALUE NIL)
         (TOS-1-CALL-VALUE NIL)
         (ERROR-RETRY-STRING (|fetch| (TRANSLATION-REC TRANS-PATTERN) |of| TRANS-REC))
         (ERROR-RETRY-PARMS (|fetch| (TRANSLATION-REC TRANS-PARAMATERS) |of| TRANS-REC))
         (NEW-LINE-INFO-REC (|create| LINE-INFO-REC))
         (INLINE-LINES NIL)
         (PATTERN-LIST NIL)
         (CALL-ACTUALS NIL))
        (DECLARE (SPECVARS *TOS-INFO* *TOS-VAL*))
```

;; substitute the inline info first

```
        (SETQ INLINE-LINES (|for| INLINE-PARMS |in| (CDR EXPANSION-LIST)
                              |collect| (LET ((INLINE-LINE-INFO (|create| LINE-INFO-REC)))
                                          (|replace| (LINE-INFO-REC PATTERN-LIST) |of| INLINE-LINE-INFO
                                             |with| (CAR INLINE-PARMS))
                                          (|replace| (LINE-INFO-REC PARAMETER-LIST) |of| INLINE-LINE-INFO
                                             |with| (CADR INLINE-PARMS))
```

```
                                        (|replace| (LINE-INFO-REC ACTUAL-PARAMETERS) |of| INLINE-LINE-INFO
                                           |with| (|for| PARM |in| (CADR INLINE-PARMS)
                                                     |collect| (PARM-SUBSTITUTE PARM TRANS-REC BYTE-REC NIL NIL
                                                                  )))
                                        INLINE-LINE-INFO)))
```

;; substitute the call line parameters

```
(SETQ PATTERN-LIST (CAAR EXPANSION-LIST))
(SETQ CALL-ACTUALS (|for| PARM |in| (CADAR EXPANSION-LIST) |collect| (PARM-SUBSTITUTE PARM TRANS-REC
                                                                          BYTE-REC NIL NIL)))
(|replace| (LINE-INFO-REC ACTUAL-PARAMETERS) |of| NEW-LINE-INFO-REC |with| CALL-ACTUALS)
(|replace| (LINE-INFO-REC PATTERN-LIST) |of| NEW-LINE-INFO-REC |with| PATTERN-LIST)
(|replace| (LINE-INFO-REC PARAMETER-LIST) |of| NEW-LINE-INFO-REC |with| (CADAR EXPANSION-LIST))
(|for| PAT |in| (CADAR EXPANSION-LIST) |as| ACTUAL |in| CALL-ACTUALS
   |do| (OR (AND (EQ 'POP (CAR PAT))
                 (SETQ |argcount| (PLUS |argcount| 4))
                 (SETQ TOS-CALL-VALUE ACTUAL))
            (AND (EQ 'POP-1 (CAR PAT))
                 (SETQ |argcount| (PLUS |argcount| 4))
                 (SETQ TOS-1-CALL-VALUE ACTUAL))))
```

;; Generate the INLINES, replacing other data

```
(ADD-INLINE-LINE (CONCAT ".inline " |inline-name| ", " |argcount|)
       NIL "")
(|for| LINE |in| INLINE-LINES |do| (ADD-INLINE-LINE "       " LINE ""))
(ADD-INLINE-LINE ".end " NIL "")
(SETQ *ARG-COUNT* 0)
(|if| (NEQ (STRPOS "BCE" (OR (AND (STRINGP ERROR-RETRY-STRING)
                                  ERROR-RETRY-STRING)
                             (CAR ERROR-RETRY-STRING)))
          1)
     |then| ;; Make the Error Entry to try the Out of Line Routine.
            (LET* ((SAVE-ERROR-STACK *ERROR-STACK*)
                   (TOS-IS-DEFERRED (NEQ TOS-CALL-VALUE 'POP))
                   (TOS-1-IS-DEFERRED (NEQ TOS-1-CALL-VALUE 'POP))
                   (ERROR-LINE-INFO-REC (|create| LINE-INFO-REC)))
                  (SETQ PATTERN-LIST ERROR-RETRY-STRING)
                  (SETQ *ERROR-STACK* NIL)
                  (|replace| (LINE-INFO-REC ACTUAL-PARAMETERS) |of| ERROR-LINE-INFO-REC
                     |with| (|for| PARM |in| ERROR-RETRY-PARMS
                               |collect| (PARM-SUBSTITUTE
                                          PARM TRANS-REC BYTE-REC NIL NIL
                                          (SELECTQ (CAR PARM)
                                               (POP (OR (AND TOS-IS-DEFERRED (OR TOS-CALL-VALUE
                                                                                 (GET-VAL *TOS-VAL*
                                                                                          *TOS-INFO*)))
                                                         (OR (AND (EQ TOS-1-CALL-VALUE 'POP)
                                                                  'GET_POPPED_2)
                                                             'GET_POPPED)))
                                               (POP-1 (OR (AND TOS-IS-DEFERRED (OR (AND TOS-1-IS-DEFERRED
                                                                                        TOS-1-CALL-VALUE)
                                                                                   'GET_POPPED))
                                                          'GET_POPPED))
                                               (ERRORPC (CONCAT |error-pc| "_b"))
                                               NIL))))
                  (|replace| (LINE-INFO-REC PARAMETER-LIST) |of| ERROR-LINE-INFO-REC |with| ERROR-RETRY-PARMS)
                  (|replace| (LINE-INFO-REC PATTERN-LIST) |of| ERROR-LINE-INFO-REC |with| PATTERN-LIST)
                  (SETQ *ERROR-STACK* SAVE-ERROR-STACK)
                  (SETQ *INLINE-ERROR-STACK* (LIST (CONS 'PUSH ERROR-LINE-INFO-REC)
                                                   (CONCAT "goto case" (PLUS |pc| 1 (|fetch| (BYTE-INFO-REC
                                                                                              OPLENGTH)
                                                                                     |of| BYTE-REC))
                                                           "_label;")))))
NEW-LINE-INFO-REC)))

)

;; Parsing Functions

(DEFINEQ

(BCE-PARSER
  (LAMBDA (TRANS-REC BYTE-REC FN-NAME |optional-string|)          ; Edited 17-Jun-88 13:04 by rtk
    (DECLARE (SPECIAL *CODE-BASE*))
    (PUSH-ALL-OPERANDS)
    (CONCAT "BCE(" (BCE-PC (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC)
                           *CODE-BASE*)
            ", "
            (|fetch| (BYTE-INFO-REC OPCODE) |of| BYTE-REC)
            ")")))

(STR-PARSER
  (LAMBDA (TRANS-REC BYTE-REC FN-NAME WAS-OPTIONAL-STRING)        ; Edited 22-Jun-88 01:08 by rtk
    (DECLARE (SPECIAL *EVAL-STACK* *CODE-BASE*))
    (LET ((PATTERN-LIST (|fetch| (TRANSLATION-REC TRANS-PATTERN) |of| TRANS-REC))
```

```
          (PARAMATERS (|fetch| (TRANSLATION-REC TRANS-PARAMATERS) |of| TRANS-REC))
          (NEW-LINE-INFO-REC (|create| LINE-INFO-REC))
          (*ADD-HEAD* NIL)
          (*ADD-TAIL* NIL))
         (DECLARE (SPECVARS *ADD-HEAD* *ADD-TAIL*))
         (|replace| (LINE-INFO-REC PARAMETER-LIST) |of| NEW-LINE-INFO-REC |with| PARAMATERS)
         (|replace| (LINE-INFO-REC ACTUAL-PARAMETERS) |of| NEW-LINE-INFO-REC
            |with| (|for| PARM |in| PARAMATERS |collect| (PARM-SUBSTITUTE PARM TRANS-REC BYTE-REC NIL NIL)))
         (IF *ADD-HEAD*
             THEN (SETQ PATTERN-LIST (CONS *ADD-HEAD* PATTERN-LIST)))
         (IF *ADD-TAIL*
             THEN (SETQ PATTERN-LIST (APPEND PATTERN-LIST (LIST *ADD-TAIL*))))
         (|replace| (LINE-INFO-REC PATTERN-LIST) |of| NEW-LINE-INFO-REC |with| PATTERN-LIST)
        NEW-LINE-INFO-REC)))
```

(COND-PARSER

```
(LAMBDA (TRANS-REC BYTE-REC FN-NAME WAS-OPTIONAL-STRING)        ; Edited 22-Jun-88 17:41 by rtk
  (DECLARE (SPECIAL *EVAL-STACK* *CODE-BASE*))
  (LET ((PATTERN-LIST (|fetch| (TRANSLATION-REC TRANS-PATTERN) |of| TRANS-REC))
        (PARAMATERS (|fetch| (TRANSLATION-REC TRANS-PARAMATERS) |of| TRANS-REC))
        (NEW-LINE-INFO-REC (|create| LINE-INFO-REC))
        (*ADD-HEAD* NIL)
        (*ADD-TAIL* NIL))
       (DECLARE (SPECVARS *ADD-HEAD* *ADD-TAIL*))
       (|replace| (LINE-INFO-REC PARAMETER-LIST) |of| NEW-LINE-INFO-REC |with| PARAMATERS)
       (|replace| (LINE-INFO-REC ACTUAL-PARAMETERS) |of| NEW-LINE-INFO-REC
          |with| (|for| PARM |in| PARAMATERS |collect| (PARM-SUBSTITUTE PARM TRANS-REC BYTE-REC NIL NIL)))
       (|replace| (LINE-INFO-REC PATTERN-LIST) |of| NEW-LINE-INFO-REC |with| PATTERN-LIST)
       (|if| *ADD-HEAD*
            |then| ;; It is NOT followed by a jump, so add the IF... stuff appropriate thing
                   (ADD-OPERAND-LINE *ADD-HEAD* NEW-LINE-INFO-REC *ADD-TAIL*)
                   NIL
         |else| ;; Followed by a JUMP so return the deffered value
                NEW-LINE-INFO-REC))))
```

(CONST-PARSER

```
(LAMBDA (TRANS-REC BYTE-REC)                                    ; Edited 21-Jun-88 18:59 by rtk
  (PROG1 (STR-PARSER TRANS-REC BYTE-REC)
      (SET-INFO 'INFO-TYPE 'SMALL-CONST))))
```

(COPY-PARSER

```
(LAMBDA (TRANS-REC BYTE-REC FN-NAME |optional-string|)          ; Edited 21-Jun-88 11:53 by rtk
  (DECLARE (SPECIAL *EVAL-STACK* *CODE-BASE* *INFO-REC*))
  (LET ((|c-string| (OR |optional-string| (|fetch| (TRANSLATION-REC PATTERN) |of| TRANS-REC))))
       ;; Check if the operand should be pushed
       (|if| *EVAL-STACK*
            |then| (TOS-CHECK TRANS-REC BYTE-REC ""))
       ;; Is the Operand still Delayed?
       (|if| *EVAL-STACK*
            |then| (CL:MULTIPLE-VALUE-BIND (|eval-string| |info-rec|)
                       (OPERAND-GET T)
                     (SETQ *INFO-REC* |info-rec|)
                     (SETQ |c-string| |eval-string|))
         ELSE (STR-PARSER TRANS-REC BYTE-REC FN-NAME |optional-string|)))))
```

(JUMP-PARSER

```
(LAMBDA (TRANS-REC BYTE-REC)                                    ; Edited 21-Jun-88 12:03 by rtk
  (DECLARE (SPECIALS *IGNORE-THIS-JUMP*))
  (|if| *IGNORE-THIS-JUMP*
       |then| NIL
    |else| (STR-PARSER TRANS-REC BYTE-REC))))
```

(FN-CALL-PARSER

```
(LAMBDA (TRANS-REC BYTE-REC FN-NAME OPTIONAL-STRING PC-OFFSET NUM-ARGS)
                                                                ; Edited 23-Jun-88 19:35 by rtk
  ;; NUM-ARGS is set by FNX call
  (DECLARE (SPECIAL *CODE-BASE* *OLD-CODE-BASE*))
  (LET* ((REAL-ARGS (OR NUM-ARGS (|fetch| (TRANSLATION-REC STACK-ARGS) |of| TRANS-REC)))
         (ALLOWED-ARGS (MIN 5 REAL-ARGS))
         (WHO-CALLED (OR (AND NUM-ARGS (|fetch| (BYTE-INFO-REC ARG2) |of| BYTE-REC)
                              (|fetch| (BYTE-INFO-REC ARG3) |of| BYTE-REC)
                              (LOGOR (LLSH (|fetch| (BYTE-INFO-REC ARG2) |of| BYTE-REC)
                                           8)
                                     (|fetch| (BYTE-INFO-REC ARG3) |of| BYTE-REC)))
                         (AND (|fetch| (BYTE-INFO-REC ARG1) |of| BYTE-REC)
                              (|fetch| (BYTE-INFO-REC ARG2) |of| BYTE-REC)
                              (LOGOR (LLSH (|fetch| (BYTE-INFO-REC ARG1) |of| BYTE-REC)
```

```
                                        8)
                                    (|fetch| (BYTE-INFO-REC ARG2) |of| BYTE-REC)))))
        (FN-DEF-CELL (\\DEFCELL (\\VAG2 0 WHO-CALLED)))
        (FN-DEF-CELL-68K (LISP-ADDR-TO-NATIVE-ADDR FN-DEF-CELL))
        (*CALL-SELF* (AND (LITATOM FN-NAME)
                          (EQ (\\LOLOC FN-NAME)
                              WHO-CALLED)))
        (NEW-PC (PLUS (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC)
                      (OR (AND NUM-ARGS 4)
                          3)))
        (*FN-CALL-STR* (OR (AND *CALL-SELF* (CONCAT NEW-PC ", pc_" ALLOWED-ARGS))
                           (CONCAT (BCE-PC (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC)
                                           *CODE-BASE*)
                                   ", " NEW-PC ", " WHO-CALLED ", " FN-DEF-CELL-68K ", ret_to_fn"
                                   (|if| (GEQ 4 REAL-ARGS)
                                       |then| REAL-ARGS
                                     |else| "x")))))
       (DECLARE (SPECVARS *CALL-SELF* *FN-CALL-STR*))
       ;; Must push all for function call & return
       (PUSH-ALL-OPERANDS)
       (ADD-OPERAND-LINE "   " (STR-PARSER TRANS-REC BYTE-REC FN-NAME OPTIONAL-STRING PC-OFFSET NUM-ARGS)
             ";")
       NIL)))
```

(FN-CALL-PARSERX

```
(LAMBDA (TRANS-REC BYTE-REC FN-NAME OPTIONAL-STRING)          ; Edited 14-Jun-88 11:24 by rtk
  (FN-CALL-PARSER TRANS-REC BYTE-REC FN-NAME OPTIONAL-STRING NIL (|fetch| (BYTE-INFO-REC ARG1) |of| BYTE-REC))))
```

(ENVCALL-PARSER

```
(LAMBDA (TRANS-REC BYTE-REC FN-NAME OPTIONAL-STRING NUM-ARGS)  ; Edited 22-Jun-88 16:05 by rtk
  (DECLARE (SPECVARS *EVAL-STACK* *CODE-BASE*))
  (LET* ((FUNCTION-PTR (LISTGET (|fetch| (BYTE-INFO-REC OPCODE-PROPS) |of| BYTE-REC)
                                'CODE-CONST))
         (NUM-OF-ARGS (LISTGET (|fetch| (BYTE-INFO-REC OPCODE-PROPS) |of| BYTE-REC)
                               'ARG-CONST))
         (BCE-PC-VALUE (BCE-PC (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC)
                               *CODE-BASE*))
         (RETURN-PC (ADD1 (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC))))
        (|if| FUNCTION-PTR
            |then| (ADD-OPERAND-LINE " " NIL (CONCAT "envcall_native(" RETURN-PC ", " (OR NUM-OF-ARGS "POP")
                                                     ", "
                                                     (LISP-ADDR-TO-NATIVE-ADDR FUNCTION-PTR)
                                                     ", "
                                                     (ITIMES (NATIVE-ADDR-WORD-OFFSET FUNCTION-PTR)
                                                             2)
                                                     ", "
                                                     (OR (OPERAND-POP)
                                                         "POP")
                                                     ")"))
                     (PUSH-ALL-OPERANDS)
              |else| (PUSH-ALL-OPERANDS)
                     (ADD-OPERAND-LINE "    " (STR-PARSER TRANS-REC BYTE-REC)
                           ";")
                     (STR-PARSER TRANS-REC BYTE-REC)))
  NIL))
```

(RETURN-PARSER

```
(LAMBDA (TRANS-REC BYTE-REC)                                  ; Edited 20-Jun-88 20:16 by rtk
  (DECLARE (SPECVARS *CODE-BASE* *EVAL-STACK*))
  ;; Must push all for function call & return
                                                              (* IGNORE FOR NOW |if| |tos-operand| |then|
                                                              (* |;;| "If TOS is not IVAR[0], then Set IVAR[0]  to result")
                                                              (|if| (OR (NOT (STRINGP |tos-operand|))
                                                              (NOT (STRING-EQUAL |tos-operand| "IVAR[0]"))) |then|
                                                              (SETQ |c-string| (CONCAT "IVAR[0] = "
                                                              (OPERAND-POP) "; " |c-string|))) |else|
                                                              (* |;;| "Result is whatever is at TOS")
                                                              (SETQ PREFIX-STR "IVAR[0] = POP; "))
  (LET ((PREFIX-STR "IVAR[0] = POP; ")
        (PATTERN-LIST (|fetch| (TRANSLATION-REC TRANS-PATTERN) |of| TRANS-REC))
        (PARAMATERS (|fetch| (TRANSLATION-REC TRANS-PARAMATERS) |of| TRANS-REC))
        (NEW-LINE-INFO-REC (|create| LINE-INFO-REC)))
       (PUSH-ALL-OPERANDS)
       (|replace| (LINE-INFO-REC PATTERN-LIST) |of| NEW-LINE-INFO-REC |with| PATTERN-LIST)
       (|replace| (LINE-INFO-REC PARAMETER-LIST) |of| NEW-LINE-INFO-REC |with| PARAMATERS)
       (|replace| (LINE-INFO-REC ACTUAL-PARAMETERS) |of| NEW-LINE-INFO-REC
          |with| (|for| PARM |in| PARAMATERS |collect| (PARM-SUBSTITUTE PARM TRANS-REC BYTE-REC NIL NIL)))
       (ADD-OPERAND-LINE PREFIX-STR NEW-LINE-INFO-REC))
  NIL))
```

(SWAP-PARSER

```
(LAMBDA (TRANS-REC BYTE-REC)                                  ; Edited 21-Jun-88 14:16 by rtk
```

```
    (DECLARE (SPECIALS *EVAL-STACK* *PREV-UFN*))
    (|if| (AND (GREATERP (LENGTH *EVAL-STACK*)
                         1)
               (NOT (OR (|fetch| (BYTE-INFO-REC JUMP-TARGET) |of| BYTE-REC)
                        *PREV-UFN*)))
          |then| (LET ((\a (CAR *EVAL-STACK*))
                       (\b (CADR *EVAL-STACK*))
                       (\c (CDDR *EVAL-STACK*)))
                   (SETQ *EVAL-STACK* (CONS \b (CONS \a \c)))
                   NIL)
      |else| (STR-PARSER TRANS-REC BYTE-REC))))
```

(PVAR_PARSER

```
  (LAMBDA (TRANS-REC BYTE-REC FN-NAME |optional-string| PC-ADJUST-SIZE)
                                                        ; Edited 23-Jun-88 18:28 by rtk
    (DECLARE (SPECIAL *EVAL-STACK* *CODE-BASE* *ENTRY-POINTS*))
    ;; Must Look for PVAR_ of SI::*CATCH-RETURN-PC*
    (LET ((CATCH-PC_ (FASSOC 'SI::*CATCH-RETURN-PC* *VAR-OFFSETS*))
          (PVAR-SLOT (OR (AND (EQ 'PVARX_ (|fetch| (BYTE-INFO-REC OP-NAME) |of| BYTE-REC))
                              (LRSH (|fetch| (BYTE-INFO-REC ARG1) |of| BYTE-REC)
                                    1))
                         (LOGAND (|fetch| (BYTE-INFO-REC OPCODE) |of| BYTE-REC)
                                 7))))
      (|if| (AND CATCH-PC_ (EQ PVAR-SLOT (CADR CATCH-PC_)))
            |then| ;; Adjust the Catch PC
                   (CL:MULTIPLE-VALUE-BIND (CATCH-PC CATCH-INFO)
                       (OPERAND-POP T)
                     (|if| (AND CATCH-PC (EQ 'SMALL-CONST (GET-INFO 'INFO-TYPE CATCH-INFO)))
                          |then| (LET ((NEW-CATCH-PC (+ CATCH-PC PC-ADJUST-SIZE)))
                                   ;; Look to the SIC load & fix it with the correct PC
                                   (LET* ((PREVIOUS-PC (|for| PC |from| (- (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC)
                                                                           1)
                                                          |by| -1 |thereis| (ELT *ENTRY-POINTS* PC)))
                                          (PREVIOUS-BYTE-REC (ELT *ENTRY-POINTS* PREVIOUS-PC))
                                          (PREVIOUS-OPCODE (|fetch| (BYTE-INFO-REC OP-NAME) |of| PREVIOUS-BYTE-REC))
                                          )
                                     ;; fixup the BYTECODES too
                                     (COND
                                        ((AND (EQ 'SIC PREVIOUS-OPCODE)
                                              (GREATERP 128 NEW-CATCH-PC))
                                         (\\PUTBASEBYTE *CODE-BASE* (+ PREVIOUS-PC 1)
                                                 NEW-CATCH-PC))
                                        ((AND (EQ 'SICX PREVIOUS-OPCODE)
                                              (GREATERP 32768 NEW-CATCH-PC))
                                         (\\PUTBASEBYTE *CODE-BASE* (+ PREVIOUS-PC 1)
                                                 (LRSH NEW-CATCH-PC 8))
                                         (\\PUTBASEBYTE *CODE-BASE* (+ PREVIOUS-PC 2)
                                                 (LOGAND NEW-CATCH-PC 255)))
                                        ((ERROR "Cannot Translate This FN due to CATCH return PC"))))
                                   ;; Mark the Catch PC as an entry point & jump target
                                   (|replace| (BYTE-INFO-REC JUMP-TARGET) |of| (ELT *ENTRY-POINTS* NEW-CATCH-PC)
                                      |with| T)
                                   (OPERAND-PUSH NEW-CATCH-PC CATCH-INFO))
                        |else| (ERROR "Non-Constant Use of PC value"))))
      (STR-PARSER TRANS-REC BYTE-REC FN-NAME |optional-string|))))

)

;; Pattern Matching  Routines

(DEFINEQ

(PARM-SUBSTITUTE
  (LAMBDA (PARAMETER TRANS-REC BYTE-REC INFO-TYPE NEW-INFO-TYPE OPTIONAL-REPLACEMENT-VAL)
    (DECLARE (SPECIAL *EVAL-STACK* *ARG-COUNT*))        ; Edited 21-Jun-88 21:02 by rtk
    (AND INFO-TYPE (SET-INFO 'INFO-TYPE INFO-TYPE))
    (LET* ((PRE-FN (CADR PARAMETER))
           (POST-FN (CADDR PARAMETER))
           (REPLACEMENT-VALUE (OR OPTIONAL-REPLACEMENT-VAL (AND PRE-FN (CL:APPLY PRE-FN (LIST TRANS-REC BYTE-REC
                                                                                              )))))
           (|str-info| NEW-INFO-TYPE))
          (SETQ *ARG-COUNT* (+ *ARG-COUNT* 1))
          (SETQ REPLACEMENT-VALUE (SELECTQ REPLACEMENT-VALUE
                                           (POP (OR (AND *EVAL-STACK* (CL:MULTIPLE-VALUE-BIND (|value| |info|)
                                                                          (OPERAND-POP)
                                                                        (SETQ |str-info| |info|)
                                                                        |value|))
                                                    (AND (ADD-INFO 'POP-COUNT 1)
                                                         REPLACEMENT-VALUE)
                                                    REPLACEMENT-VALUE))
                                           (TOS (OR (CL:MULTIPLE-VALUE-BIND (|value| |info|)
```

```
                                                    (OPERAND-GET)
                                                   (SETQ |str-info| |info|)
                                                   |value|)
                                                  REPLACEMENT-VALUE))
                                            REPLACEMENT-VALUE))
          (OR (AND POST-FN (CL:APPLY POST-FN (LIST TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE |str-info|)))
              REPLACEMENT-VALUE))))
```

(TOS-CHECK

```
(LAMBDA (TRANS-REC BYTE-REC)
  (DECLARE (SPECIALS *PC-BUMP-SIZE*))                          ; Edited 22-Jun-88 03:10 by rtk
  (LET ((|next-opcode| (|fetch| (BYTE-INFO-REC NEXT-BYTE-REC) |of| BYTE-REC))
        (IS-NCJUMP (FMEMB (FETCH (BYTE-INFO-REC OP-NAME) OF BYTE-REC)
                          '(NTJUMPX NFJUMPX))))
       (COND
          ((AND |next-opcode| (NOT (|fetch| (BYTE-INFO-REC JUMP-TARGET) |of| |next-opcode|))
                (NOT (|fetch| (TRANSLATION-REC MAY-UFN) |of| TRANS-REC))
                (NOT IS-NCJUMP)
                (EQ (|fetch| (OPCODENAME) |of| (|fetch| (BYTE-INFO-REC OPCODE-REC) |of| |next-opcode|))
                    'POP)
                (|add| *PC-BUMP-SIZE* 1))
           ;; Determined if a POP could be used instead of a TOS
           'POP)
          (T ;; determine if must push out eval stack before using TOS
             ;; this is a hack by putting a space as 1st char in eval stack when it is complicated & shouldn't be repeated
             (AND (OR (GET-INFO 'POP-COUNT)
                      (LET ((OPERAND (OPERAND-GET)))
                           (IF (TYPEP OPERAND 'LINE-INFO-REC)
                               THEN (STREQUAL (SUBSTRING (CAR (FETCH (LINE-INFO-REC PATTERN-LIST)
                                                                     OF OPERAND))
                                                         1 1)
                                              " ")
                             ELSE (AND OPERAND (STREQUAL (SUBSTRING OPERAND 1 1)
                                                         " "))))
                  IS-NCJUMP)
              (PUSH-ALL-OPERANDS))
             'TOS)))))

)

;; Output of Code lines

(DEFINEQ

(ADD-CASE
  (LAMBDA (|pc| |prev-ufn| |jump-target| |negative-jump-target| |can-ufn|)
                                                               ; Edited 21-Jun-88 23:43 by rtk
    (DECLARE (SPECIALS *ENTRY-POINTS* *START-PC*))
    (|if| (OR |jump-target| |prev-ufn| (AND (GET-INFO 'POP-COUNT)
                                            |can-ufn|))
        |then| (PUSH-ALL-OPERANDS))
    (|if| |jump-target|
        |then| (ADD-LINE (CONCAT "pc" (|if| (MINUSP |pc|)
                                           |then| (CONCAT "_" (ABS |pc|))
                                          |else| |pc|)
                                 ": ")))
    (|if| (OR |prev-ufn| |jump-target| |negative-jump-target|)
        |then| (LET* ((|entry-case| (CONCAT "case" (PC-XFORM |pc|)))
                      (THIS-ENTRY-POINT (AND (GEQ |pc| *START-PC*)
                                             (ELT *ENTRY-POINTS* |pc|))))
                     (|if| THIS-ENTRY-POINT
                         |then| (|replace| (BYTE-INFO-REC ENTRY-ADDRESS) |of| THIS-ENTRY-POINT |with| T))
                     (ADD-LINE (CONCAT |entry-case| "_label: " |entry-case| "();"))
                     (ADD-INLINE-LINE (CONCAT ".inline _" |entry-case| ", 0"))
                     (ADD-INLINE-LINE "    " |entry-case| ": ")
                     (ADD-INLINE-LINE ".end ")))
    (|if| |negative-jump-target|
        |then| (ADD-LINE (CONCAT "TIMER_STACK_CHECK(" (BCE-PC |pc| *CODE-BASE*)
                                 ");")
                       T))))
```

(ADD-PUSH-OPERAND-LINE

```
(LAMBDA (LINE-INFO |error-cases| |info-rec|)                 ; Edited 21-Jun-88 19:45 by rtk
  ;; PUSH the given operand on the Lisp Stack
  ;; IF it uses a POP, then do TOS = TOS instead of PUSH(POP)
  ;; RETURNS: the new error-stack
  (LET ((|error-return| |error-cases|))
       (COND
          ((AND (TYPEP LINE-INFO 'LINE-INFO-REC)
                (|fetch| (INFO-REC POP-COUNT) |of| |info-rec|)
```

```
            (FMEMB 'POP (|fetch| (LINE-INFO-REC ACTUAL-PARAMETERS) |of| LINE-INFO)))
        (SETQ |error-return| (|bind| (|found-pop| _ NIL) |for| |case| |in| |error-cases|
                                |collect| (OR (AND (NOT |found-pop|)
                                                  (SETQ |found-pop| (EQ 'POP |case|))
                                                  'SAVE_PUSH_TOS)
                                              |case|)))
        (|replace| (LINE-INFO-REC ACTUAL-PARAMETERS) |of| LINE-INFO
           |with| (REVERSE (|bind| (POP-FOUND _ NIL) |for| PARM |in| (REVERSE (|fetch| (LINE-INFO-REC
                                                                                        ACTUAL-PARAMETERS)
                                                                                  |of| LINE-INFO))
                                |collect| (COND
                                             ((OR POP-FOUND (NEQ PARM 'POP))
                                              PARM)
                                             (T (SETQ POP-FOUND T)
                                                'TOS)))))
        (ADD-OPERAND-LINE "       TOS = " LINE-INFO ";")
        ;; Fixup the INLINE-EXPAND re-evaluate call also
        (|if| *INLINE-ERROR-STACK*
            |then| (RPLACA *INLINE-ERROR-STACK* (CONS 'TOS (CDAR *INLINE-ERROR-STACK*)))))
       ((TYPEP LINE-INFO 'LINE-INFO-REC)
        (ADD-OPERAND-LINE "       PUSH(" LINE-INFO ");"))
       (T (ADD-OPERAND-LINE "     PUSH(" NIL (CONCAT LINE-INFO ");"))))
    |error-return|)))
```

(ADD-FN-HEADER-INFO

```
(LAMBDA (FN-NAME ENTRY-NAME)                                    ; Edited 24-Jun-88 15:30 by rtk
  (DECLARE (SPECIAL *NUMBER-OF-ARGS* *ENTRY-POINT-MAX* *PVAR-QUAD-SIZE* *STACK-MIN-SIZE* *CODE-BASE*))
  (SETQ *ENTRY-POINT-MAX* (MAX 5 (ADD1 *NUMBER-OF-ARGS*)))
  (ADD-LINE "#include \"nativeincludes.h\"")
  (ADD-LF)
  (ADD-LINE "#define entry_pc ((int) PC)")
  (ADD-LF)
  (ADD-LINE "LispPTR T_NIL_VALUES[2] = {NIL_PTR, ATOM_T};")
  (ADD-LF)
  (ADD-LINE (CONCAT "int " ENTRY-NAME "()"))
  (ADD-LINE "{")
  (ADD-LF)
  (ADD-LINE "extern int entry_table[255];")
  (ADD-LINE "register LispPTR *CSTKPTR;")
  (ADD-LINE "register LispPTR *IVAR;")
  (ADD-LINE "register LispPTR *PVAR;")
  (ADD-LINE "register LispPTR TOS_CACHE;")
  (ADD-LINE "register LispPTR *DATUM68K;")
  (ADD-LINE "register LispPTR TEMPREG;")
  (ADD-LF)
  (ADD-LINE "goto entrylabel;")
  (ADD-LF)
  (ADD-LINE "entry_table_setup_label:")
  (ADD-LINE "entry_table_setup();")
  (ADD-LF)
  (ADD-LINE "unknown_entry_point: asm(\"unknown_entry_point: \");")
  (ADD-LINE "QUIT_NATIVE(entry_pc + (int)FuncObj);" T)
  (ADD-LF)
  (ADD-LINE "illegal_pc: asm(\"illegal_pc: \");")
  (ADD-LINE "NATIVE_EXT(entry_pc + (int)FuncObj);" T)
  (ADD-LINE "error(\"Illegal PC in native code\");" T)
  (ADD-LINE "asmgoto(&ret_to_dispatch);" T)
  (ADD-LF)
  (ADD-LINE '|include-errors|)
  (ADD-LF)
  (ADD-LINE "entrylabel: ")
  (ADD-LF)
  ;; This entry point is only executed once, it replaces the native entry address in the code block with the entry code following.
  (ADD-LINE "entry_point_setup();")
  (ADD-LINE "CSTKPTR = (LispPTR *) CurrentStackPTR;")
  (ADD-LINE "if ((int) entry_pc <= 0) {IVAR = CSTKPTR + (int) entry_pc;}")
  (ADD-LINE "else {IVAR = (LispPTR *) IVar;}")
  (ADD-LINE "PVAR = (LispPTR *) PVar;")
  (ADD-LF)
  (ADD-LINE "switchlabel: ")
  (ADD-LINE (CONCAT "asmgoto(entry_table[entry_pc+" *ENTRY-POINT-MAX* "]);"))
  (ADD-LF)
  ;; Add Arg Count Dispatch Entry Code
  (COND
     ((GREATERP 0 *NUMBER-OF-ARGS*)
      (CL-ADD-FN-HEADER-INFO FN-NAME ENTRY-NAME))
     (T (IL-ADD-FN-HEADER-INFO FN-NAME ENTRY-NAME)))
  ;; Push the PVAR info
  (ADD-CASE 1 T NIL T)
  (ADD-LINE "{register tempreg = 0x0ffffffff;" T)
  (|if| (|fetch| (FNHEADER CLOSUREP) |of| *CODE-BASE*)
      |then| (ADD-LINE "PUSH(native_closure_env);" T)
```

```
            (ADD-LINE "native_closure_env = tempreg;" T)
      |else| (ADD-LINE "PUSH(tempreg);" T))
    (ADD-LINE "PUSH(tempreg);" T)
    (|for| |npvars| |from| 0 |to| *PVAR-QUAD-SIZE* |do| (ADD-LINE "PUSH(tempreg);" T)
                                                       (ADD-LINE "PUSH(tempreg);" T))
    (ADD-LINE "}" T)
    ;; Add Entry Point for the StartPC
    (ADD-CASE (|fetch| (FNHEADER STARTPC) |of| *CODE-BASE*)
           T NIL NIL)))
```

(IL-ADD-FN-HEADER-INFO

```
(LAMBDA (FN-NAME ENTRY-NAME)                                    ; Edited 23-Jun-88 16:58 by rtk
  (DECLARE (SPECIAL *NUMBER-OF-ARGS* *ENTRY-POINT-MAX* *PVAR-QUAD-SIZE* *STACK-MIN-SIZE* *CODE-BASE*))
  ;; Entry points when we have Too Many/Few arguments
  (ADD-CASE (MINUS *ENTRY-POINT-MAX*)
           T T)
  (ADD-LINE "{register int i;" T)
  (ADD-LINE (CONCAT "for (i = (-entry_pc) & 0x7f; i > " *NUMBER-OF-ARGS* "; i--) POP; ")
           T)
  (ADD-LINE (CONCAT "STACK_ONLY_CHECK(" (|fetch| (FNHEADER STKMIN) |of| *CODE-BASE*)
                    ");")
           T)
  (ADD-LINE (CONCAT "for (i = (-entry_pc) & 0x7f; i<" *NUMBER-OF-ARGS* "; i++) {")
           T)
  (ADD-LINE "PUSH(NIL_PTR);" T)
  (ADD-LINE "}" T)
  (ADD-LINE (CONCAT (OR (AND (EQ *NUMBER-OF-ARGS* 0)
                             "goto pc")
                        "goto pc_")
                    *NUMBER-OF-ARGS* ";")
           T)
  (ADD-LINE "}" T)
  (|for| \i |from| (MAX 4 *NUMBER-OF-ARGS*) |to| 0 BY -1
     |do| (COND
             ((GREATERP \i *NUMBER-OF-ARGS*)
              (ADD-CASE (MINUS \i)
                     T T)
              (ADD-LINE "POP;" T)
              (|if| (EQ (ADD1 *NUMBER-OF-ARGS*)
                        \i)
                  |then| (ADD-LINE (CONCAT (OR (AND (EQ *NUMBER-OF-ARGS* 0)
                                                    "goto pc")
                                               "goto pc_")
                                           *NUMBER-OF-ARGS* ";")
                                   T)))))
  ;; Entry points when we have Too Few or Correct arguments
  (|for| \i |from| 0 |to| (MAX 4 *NUMBER-OF-ARGS*) |do| (COND
                                                          ((GREATERP *NUMBER-OF-ARGS* \i)
                                                           (ADD-CASE (MINUS \i)
                                                                  T T)
                                                           (ADD-LINE "PUSH(NIL_PTR);" T))
                                                          ((EQ *NUMBER-OF-ARGS* \i)
                                                           (ADD-CASE (MINUS \i)
                                                                  T T)
                                                           ;; IVAR IS ALREADY SET IN ENTRY CODE
                                                           (ADD-LINE (CONCAT "IVAR = CSTKPTR - " \i ";") T)
                                                           (ADD-LINE (CONCAT "framesetup(" \i ", " *STACK-MIN-SIZE*
                                                                             ", " (SWAPPED-FN-OBJ *CODE-BASE*)
                                                                             ");")
                                                                  T))))))
```

(CL-ADD-FN-HEADER-INFO

```
(LAMBDA (FN-NAME ENTRY-NAME)                                    ; Edited 22-Jun-88 18:57 by rtk
  (DECLARE (SPECIAL *NUMBER-OF-ARGS* *ENTRY-POINT-MAX* *PVAR-QUAD-SIZE* *STACK-MIN-SIZE* *CODE-BASE*))
  ;; Entry points when we have Too Many/Few arguments
  (FOR I FROM (MINUS *ENTRY-POINT-MAX*) TO 0 DO (ADD-CASE I T T))
  (ADD-LINE (CONCAT "framesetup(0, " *STACK-MIN-SIZE* ", " (SWAPPED-FN-OBJ *CODE-BASE*)
                    ");")
           T)))

)

;; Low Level Output of  code lines

(DEFINEQ

(ADD-LINE
  (LAMBDA (|line| |indent|)                                     ; Edited 20-Jun-88 17:50 by rtk
    (ADD-OPERAND-LINE (OR (AND |indent| "        ")
                          "")
           NIL |line|)))
```

```
(ADD-OPERAND-LINE
  (LAMBDA (PREFIX-STRING OPERAND-LIST POSTFIX-STRING)
    (DECLARE (SPECIALS *CPROGRAM*))                          ; Edited 20-Jun-88 17:50 by rtk
    (SETQ *CPROGRAM* (TCONC *CPROGRAM* (|create| LINE-RECORD-INFO
                                          PREFIX-STRING _ PREFIX-STRING
                                          LINE-INFO-LIST _ OPERAND-LIST
                                          POSTFIX-STRING _ POSTFIX-STRING)))))

(ADD-LF
  (LAMBDA NIL                                                ; Edited 17-Feb-88 10:19 by rtk
    (ADD-LINE " ")))

(ADD-ASM-LINE
  (LAMBDA (|line|)                                           ; Edited 17-Feb-88 11:07 by rtk
    (ADD-LINE (CONCAT "asm(\"" |line| "\");")
           T)))

(ADD-INLINE-LINE
  (LAMBDA (HEAD-STR LINE-INFO TAIL-STR)
    (DECLARE (SPECIALS *CPROGRAM* *INLINES* *SHOW-INLINE*))  ; Edited 21-Jun-88 23:30 by rtk
    (SETQ *INLINES* (TCONC *INLINES* (|create| LINE-RECORD-INFO
                                        PREFIX-STRING _ HEAD-STR
                                        LINE-INFO-LIST _ LINE-INFO
                                        POSTFIX-STRING _ TAIL-STR)))))

(BCE-LINE
  (LAMBDA (PC)                                               ; Edited  7-Mar-88 12:20 by rtk
    (DECLARE (SPECVARS *CODE-BASE*))
    (CONCAT "BCE(" (BCE-PC PC *CODE-BASE*)
           ");")))

)

;; Error Line Functions

(DEFINEQ

(ADD-ERROR-LINE
  (LAMBDA (HEAD-STR LINE-INFO TAIL-STR)
    (DECLARE (SPECIALS *ERROR-CASES*))                       ; Edited 21-Jun-88 11:13 by rtk
    (SETQ *ERROR-CASES* (TCONC *ERROR-CASES* (|create| LINE-RECORD-INFO
                                                PREFIX-STRING _ HEAD-STR
                                                LINE-INFO-LIST _ LINE-INFO
                                                POSTFIX-STRING _ TAIL-STR)))))

(ADD-ERROR-ENTRY
  (LAMBDA (TRANS-REC BYTE-REC)                               ; Edited 22-Jun-88 00:15 by rtk
    (DECLARE (SPECIALS *ERROR-STACK* *CODE-BASE* *ARG-COUNT* *ERROR-PC* *INLINE-ERROR-STACK*))
    ;; *INLINE-ERROR-STACK* format:
    ;; ( "string" | ( [TOS | POP]  LINE-INFO-REC) )
    ;; *ERROR-STACK* format:
    ;; ( )
    (LET
     ((|pc| (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC)))
     (|if| (|fetch| (TRANSLATION-REC MAY-UFN) |of| TRANS-REC)
          |then|
          (|if| *INLINE-ERROR-STACK*
              |then| (ADD-ERROR-LINE "errorpc" |pc| "_b:")
            |else| (ADD-ERROR-LINE "errorpc" |pc| ":")
                   (ADD-ERROR-LINE "asm(\"errorpc" |pc| ":\");"))
          (LET ((|stack-fix| (AND *ERROR-PC* (SELECTQ *ARG-COUNT*
                                                (0 NIL)
                                                (1 "fixsp1();")
                                                (2 "fixsp2();")
                                                (3 "fixsp3();")
                                                (CONCAT "fixspn(" *ARG-COUNT* ");")))))
               (|if| |stack-fix|
                   |then| (ADD-ERROR-LINE "        " NIL |stack-fix|)))
          (|for| |cases| |in| *ERROR-STACK* |do| (COND
                                                   ((EQ |cases| 'POP)
                                                    (ADD-ERROR-LINE "        " NIL "CSTKPTR++;"))
                                                   ((EQ |cases| 'TOS)
                                                    NIL)
                                                   ((EQ |cases| 'SAVE_PUSH_TOS)
                                                    NIL)
                                                   ((EQ |cases| 'COPY_TOP)
                                                    (ADD-ERROR-LINE "        " NIL "PUSH(COPY_TOP);"))
```

```
                                        (T (ADD-ERROR-LINE "     PUSH(" |cases| ");"))))
        (ADD-ERROR-LINE "      " NIL (BCE-LINE |pc|))
        (|if| *INLINE-ERROR-STACK*
            |then| (ADD-ERROR-LINE "errorpc" |pc| ":")
                   (ADD-ERROR-LINE "asm(\"errorpc" |pc| ":\");")
                   (|for| |cases| |in| *INLINE-ERROR-STACK*
                      |do| (COND
                              ((LISTP |cases|)
                               (LET ((THIS-CASE (CDR |cases|)))
                                    (SELECTQ (CAR |cases|)
                                         (PUSH (ADD-ERROR-LINE "  PUSH(" THIS-CASE ");"))
                                         (TOS (LET* ((LINE-INFO (CDR |cases|))
                                                     (PATTERN-LIST (FETCH (LINE-INFO-REC PATTERN-LIST)
                                                                          OF LINE-INFO))
                                                     (PARMS (FETCH (LINE-INFO-REC PARAMETER-LIST) OF LINE-INFO))
                                                     (ACTUALS (FETCH (LINE-INFO-REC ACTUAL-PARAMETERS)
                                                                     OF LINE-INFO)))
                                                    (REPLACE (LINE-INFO-REC ACTUAL-PARAMETERS) OF LINE-INFO
                                                       WITH (|bind| (POP-FOUND _ NIL) |for| PARM |in| ACTUALS
                                                                  |collect| (COND
                                                                               ((EQ PARM 'GET_POPPED)
                                                                                'TOS)
                                                                               ((EQ PARM 'GET_POPPED_2)
                                                                                'GET_POPPED)
                                                                               (T PARM))))
                                                    (ADD-ERROR-LINE "     TOS = " LINE-INFO ";")))
                                         (ADD-ERROR-LINE "       " NIL THIS-CASE))))
                              (T (ADD-ERROR-LINE "       " NIL |cases|)))))))))
```

(ADD-ERROR-SELECT

```
  (LAMBDA (ENTRY-NAME)
    (DECLARE (SPECIALS *ERROR-CASES*))                          ; Edited  4-Apr-88 19:44 by rtk
    (LET ((ERROR-PC-STREAM (OPENSTREAM (CONCAT ENTRY-NAME ".h")
                                       'OUTPUT)))
         (PRINTOUT ERROR-PC-STREAM T)
         (PRINTOUT ERROR-PC-STREAM "/* Error Exits for " ENTRY-NAME " */" T)
         (TERPRI ERROR-PC-STREAM)
         (TERPRI ERROR-PC-STREAM)
         (|for| |cases| |in| (CAR *ERROR-CASES*) |do| (PRINTOUT ERROR-PC-STREAM |cases| T))
         (TERPRI ERROR-PC-STREAM)
         (CLOSEF ERROR-PC-STREAM))))

)
```

;; Low Level Routines

```
(DEFINEQ
```

(FIX-FILENAME

```
  (LAMBDA (|fn-name|)                                           ; Edited  6-Jun-88 18:28 by rtk
    (PACK (|for| |letter| |in| (UNPACK (CONCAT 'LISP_TO_C_ (CL:MACHINE-INSTANCE)
                                              "_" |fn-name|))
            |when| (STRPOS |letter| "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_-\\.,")
            |collect| (COND
                         ((EQ |letter| '\\)
                          '|bs_|)
                         ((EQ |letter| '-)
                          '_)
                         ((EQ |letter| '\.)
                          '_)
                         ((EQ |letter| '\,)
                          '_)
                         (T |letter|))))))
```

(PC-XFORM

```
  (LAMBDA (|pc|)                                                ; Edited  7-Apr-88 12:07 by rtk
    (|if| (MINUSP |pc|)
        |then| (CONCAT "_" (ABS |pc|))
      |else| |pc|)))
```

(BCE-PC

```
  (LAMBDA (|pc| |fn-base| |numeric|)                            ; Edited 20-May-88 15:08 by rtk
    (OR (AND |numeric| (+ |pc| (FN-OBJ |fn-base|)))
        (CONCAT "(" |pc| " + " (FN-OBJ |fn-base|)
                ")"))))
```

(ENVCALL-FN-OBJECT

```
  (LAMBDA (BYTE-REC)                                            ; Edited 19-May-88 17:02 by rtk
```

;; Determine if the Function Pointer to this call is a GCONST. If it is then Also determine if the GCONST has already been translated, or must be
;; added to the re-translate list.

;; RETURNS: the GCONST Function Pointer if it must be translated, else NIL.

```
(DECLARE (SPECIAL *ENTRY-POINTS* *START-PC* *FN-OBJECT-REMAP-LIST*))
                                                          ; Edited 8-May-88 11:03 by rtk
(LET*
 ((MAX-DESCEND 20)
  (ENVCALL-PC (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC))
  (CURR-PC (SUB1 ENVCALL-PC))
  (FN-OBJECT-ENTRY-LEVEL (IDIFFERENCE (|fetch| (BYTE-INFO-REC ENTRY-STACK-DEPTH) |of| BYTE-REC)
                                      2))
  (ARG-COUNT-ENTRY-LEVEL (SUB1 FN-OBJECT-ENTRY-LEVEL))
  (LOOP-DONE NIL)
  (RETURN-VALUE NIL)
  (SAW-FN-GCONST NIL))
 (|while| (NOT LOOP-DONE)
    |do| (LET* ((THIS-ENTRY (ELT *ENTRY-POINTS* CURR-PC))
                (THIS-ENTRY-STACK-DEPTH (AND THIS-ENTRY (|fetch| (BYTE-INFO-REC ENTRY-STACK-DEPTH) |of|
                                                                                        THIS-ENTRY
                                                                                        ))))
                (SETQ LOOP-DONE
                 (OR (AND THIS-ENTRY (OR (AND (EQ FN-OBJECT-ENTRY-LEVEL THIS-ENTRY-STACK-DEPTH)
                                              (EQ 'GCONST (|fetch| (BYTE-INFO-REC OP-NAME) |of| THIS-ENTRY))
                                              (LET* ((GCONST-PTR (VAG2 (|fetch| (BYTE-INFO-REC ARG1)
                                                                          |of| THIS-ENTRY)
                                                                       (LOGOR (LLSH (|fetch| (BYTE-INFO-REC
                                                                                                 ARG2)
                                                                                             |of| THIS-ENTRY)
                                                                                    8)
                                                                              (|fetch| (BYTE-INFO-REC ARG3)
                                                                                 |of| THIS-ENTRY))))
                                                     (GCONST-INFO (FASSOC GCONST-PTR *FN-OBJECT-REMAP-LIST*))
                                                     (GCONST-BEEN-REMAPPED (AND GCONST-INFO (CDR GCONST-INFO))
                                                          ))
                                              (|if| (AND GCONST-BEEN-REMAPPED (|fetch| (FNHEADER NATIVE)
                                                                                   |of| GCONST-BEEN-REMAPPED))
                                                  |then| ;; no need to push the address if native code
                                                       (|replace| (BYTE-INFO-REC OPCODE) |of| THIS-ENTRY
                                                          |with| (CAR (\\FINDOP 'NOP)))
                                                       (|replace| (BYTE-INFO-REC OPCODE-PROPS)
                                                          |of| BYTE-REC |with| (LIST 'CODE-CONST
                                                                                    GCONST-BEEN-REMAPPED))
                                                       (SETQ SAW-FN-GCONST T))
                                              (AND (NULL GCONST-BEEN-REMAPPED)
                                                   (SETQ RETURN-VALUE GCONST-PTR))))
                                         (|fetch| (BYTE-INFO-REC JUMP-TARGET) |of| THIS-ENTRY)
                                         (GEQ FN-OBJECT-ENTRY-LEVEL THIS-ENTRY-STACK-DEPTH)))
                     (GREATERP *START-PC* CURR-PC)
                     (GREATERP ENVCALL-PC (IPLUS CURR-PC MAX-DESCEND))))
                (SETQ CURR-PC (SUB1 CURR-PC))))
 (SETQ LOOP-DONE NIL)
 ;; Look for the number of args being pushed as a constant (if we have a constant code object
 (|while| (AND SAW-FN-GCONST (NOT RETURN-VALUE)
               (NOT LOOP-DONE))
    |do| (LET* ((THIS-ENTRY (ELT *ENTRY-POINTS* CURR-PC))
                (THIS-ENTRY-STACK-DEPTH (AND THIS-ENTRY (|fetch| (BYTE-INFO-REC ENTRY-STACK-DEPTH) |of|
                                                                                        THIS-ENTRY
                                                                                        ))))
                (SETQ LOOP-DONE (OR (AND THIS-ENTRY
                                         (OR (AND (EQ ARG-COUNT-ENTRY-LEVEL THIS-ENTRY-STACK-DEPTH)
                                                  (FMEMB (|fetch| (BYTE-INFO-REC OP-NAME) |of| THIS-ENTRY)
                                                         '(SIC \'1 \'0))
                                                  (LET* ((OPCODE-NAME (|fetch| (BYTE-INFO-REC OP-NAME)
                                                                         |of| THIS-ENTRY))
                                                         (NUM-OF-ARGS (COND
                                                                         ((EQ '\'1 OPCODE-NAME)
                                                                          1)
                                                                         ((EQ '\'0 OPCODE-NAME)
                                                                          0)
                                                                         (T (|fetch| (BYTE-INFO-REC ARG1)
                                                                               |of| THIS-ENTRY)))))
                                                       ;; Don't have to push a num of args
                                                       (|replace| (BYTE-INFO-REC OPCODE) |of| THIS-ENTRY
                                                          |with| (CAR (\\FINDOP 'NOP)))
                                                       (|replace| (BYTE-INFO-REC OPCODE-PROPS) |of| BYTE-REC
                                                          |with| (CONS 'ARG-CONST (CONS NUM-OF-ARGS
                                                                                        (|fetch| (BYTE-INFO-REC
                                                                                                    OPCODE-PROPS)
                                                                                           |of| BYTE-REC))))
                                                       T))
                                             (|fetch| (BYTE-INFO-REC JUMP-TARGET) |of| THIS-ENTRY)
                                             (GEQ ARG-COUNT-ENTRY-LEVEL THIS-ENTRY-STACK-DEPTH)))
                                    (GREATERP *START-PC* CURR-PC)
                                    (GREATERP ENVCALL-PC (IPLUS CURR-PC MAX-DESCEND))))
                (SETQ CURR-PC (SUB1 CURR-PC))))
 RETURN-VALUE)))
```

(FIND-FN0-OBJECTS

```
(LAMBDA (ENVCALL-LIST)                                  ; Edited 19-May-88 10:36 by rtk
```

;; look throuth the list of envcalls. Return a list of those which have constant function objects which have not been translated yet.

```
    (LET ((FN-OBJECT-LIST NIL))
         (|for| BYTE-REC |in| ENVCALL-LIST |do| (LET ((FN-TO-TRANSLATE (ENVCALL-FN-OBJECT BYTE-REC)))
                                                  (|if| FN-TO-TRANSLATE
                                                      |then| (|push| FN-OBJECT-LIST FN-TO-TRANSLATE))))
         FN-OBJECT-LIST)))
```

(CONST-POINTERP

```
(LAMBDA (\x)
  NIL))
```

(MAKE-VAR-OFFSETS

```
(LAMBDA (CODE-BASE)
  (DECLARE (USEDFREE CODEBASE IVARS PVARS FVARS I4 I6 OUTF))  ; Edited 23-Jun-88 18:09 by rtk
```

;; Return List of (Symbol Name, Slot Offset , Access Type)

```
    (LET ((START1 (UNFOLD (|fetch| (FNHEADER OVERHEADWORDS) |of| T)
                          BYTESPERWORD))
          (START2 (UNFOLD (|fetch| (FNHEADER NTSIZE) |of| CODE-BASE)
                          BYTESPERWORD))
          VAR-LIST NAME TAG THEELT)
         (COND
            ((ILESSP START1 (SETQ START2 (IPLUS START2 START1)))
             (|for| NT1 |from| START1 |by| BYTESPERWORD |while| (ILESSP NT1 START2) |as| NT2 |from| START2 |by|
                                                                                                  BYTESPERWORD
                |do| (COND
                       ((SETQ NAME (\\INDEXATOMVAL (CODEBASELT2 CODE-BASE NT1)))
                        (SETQ TAG (CODEBASELT CODE-BASE (ADD1 NT2)))
                        (SETQ THEELT (CODEBASELT CODE-BASE NT2))
                        (|push| VAR-LIST (LIST NAME TAG (SELECTC THEELT
                                                              ((LRSH IVARCODE 8)
                                                                  'IVAR)
                                                              ((LRSH PVARCODE 8)
                                                                  'PVAR)
                                                              'FVAR))))))))
         VAR-LIST)))

)

;; Deferred Stack Funcions

(DEFINEQ
```

(NEXT-OPERAND

```
(LAMBDA NIL                                             ; Edited 27-Apr-88 21:25 by rtk
```

;; Get the Next operand for a binary operation, keeping numeric constants

```
    (DECLARE (SPECIALS *EVAL-STACK*))
    (OR (OPERAND-POP T)
        'POP)))
```

(PUSH-ALL-OPERANDS

```
(LAMBDA NIL                                             ; Edited 28-Apr-88 15:02 by rtk
  (DECLARE (SPECIAL *EVAL-STACK*))
  (SETQ *EVAL-STACK* (REVERSE *EVAL-STACK*))
  (|while| *EVAL-STACK* |do| (CL:MULTIPLE-VALUE-BIND (|operand| |info-rec|)
                                 (OPERAND-POP)
                               (ADD-PUSH-OPERAND-LINE |operand| NIL |info-rec|)))))
```

(OPERAND-PUSH

```
(LAMBDA (LINE-INFO |info|)                              ; Edited 21-Jun-88 19:42 by rtk
  (DECLARE (SPECIALS *EVAL-STACK*))
  (AND (GET-INFO 'POP-COUNT)
       (PUSH-ALL-OPERANDS))
  (COND
     ((AND (TYPEP LINE-INFO 'LINE-INFO-REC)
           (EQ (LENGTH (|fetch| (LINE-INFO-REC PATTERN-LIST) |of| LINE-INFO))
               1)
           (EQ (LENGTH (|fetch| (LINE-INFO-REC PARAMETER-LIST) |of| LINE-INFO))
               1))
```

;; turn LINE-INFO-RECS back into constants when you can

```
      (|push| *EVAL-STACK* (CONS (CAR (|fetch| (LINE-INFO-REC ACTUAL-PARAMETERS) |of| LINE-INFO))
                                 |info|)))
     (T (|push| *EVAL-STACK* (CONS LINE-INFO |info|))))))
```

(OPERAND-GET

```
(LAMBDA (|keep-constants|)                              ; Edited  7-May-88 10:53 by rtk
```

```
    (DECLARE (SPECIALS *EVAL-STACK*))
    (LET* ((|operand| (AND *EVAL-STACK* (CAR *EVAL-STACK*)))
           (|operand-string| (AND |operand| (CAR |operand|)))
           (|operand-info| (AND |operand| (CDR |operand|)))
           (|operand-val| (AND |operand| (OR (AND |keep-constants| |operand-string|)
                                             (GET-VAL |operand-string| |operand-info|)))))
          (CL:VALUES |operand-val| |operand-info|)))
```

(OPERAND-POP

```
  (LAMBDA (|keep-constants|)                                    ; Edited 24-May-88 10:25 by rtk
    (DECLARE (SPECIAL *EVAL-STACK*))
    (CL:MULTIPLE-VALUE-PROG1 (OPERAND-GET |keep-constants|)
           (AND *EVAL-STACK* (|pop| *EVAL-STACK*)))))
```

(GET-VAL

```
  (LAMBDA (\x |operand-info|)                                   ; Edited 24-May-88 10:48 by rtk
    (|if| (AND (NUMBERP \x)
               (EQ (|fetch| (INFO-REC INFO-TYPE) |of| |operand-info|)
                   'SMALL-CONST))
          |then| (|if| (GEQ \x 0)
                       |then| (CONCAT "(S_POSITIVE | " \x ")")
                      |else| (CONCAT "(S_NEGATIVE | " (LOGAND \x 65535)
                                     ")"))
      |else| \x)))
```

(GET-SHIFTED-VAL

```
  (LAMBDA (\x |shiftcount|)                                     ; Edited 18-Feb-88 17:05 by rtk
    (|if| (NUMBERP \x)
        |then| (LOGAND (LLSH \x |shiftcount|)
                       4294934528)
      |else| \x)))
```

(GET-INFO

```
  (LAMBDA (|prop| |info|)                                       ; Edited  7-May-88 10:46 by rtk
    (DECLARE (SPECIAL *EVAL-STACK*))
    (LET ((|stack-pos| (OR |info| (AND *EVAL-STACK* (CDAR *EVAL-STACK*)))))
         (AND |stack-pos| (SELECTQ |prop|
                                   (POP-COUNT (|fetch| (INFO-REC POP-COUNT) |of| |stack-pos|))
                                   (INFO-TYPE (|fetch| (INFO-REC INFO-TYPE) |of| |stack-pos|))
                                   NIL)))))
```

(ADD-INFO

```
  (LAMBDA (|prop| |val|)
    (DECLARE (SPECIAL *INFO-REC*))                              ; Edited 29-Apr-88 11:05 by rtk
    (LET ((|oldvalue| (OR (AND *INFO-REC* (GET-INFO |prop| *INFO-REC*))
                          (AND (SETQ *INFO-REC* (|create| INFO-REC
                                                          POP-COUNT _ 0))
                               0))))
         (SET-INFO |prop| (IPLUS |val| |oldvalue|)))))
```

(SET-INFO

```
  (LAMBDA (|prop| |val|)
    (DECLARE (SPECIAL *INFO-REC*))                              ; Edited  7-May-88 10:47 by rtk
    (OR *INFO-REC* (SETQ *INFO-REC* (|create| INFO-REC)))
    (SELECTQ |prop|
        (POP-COUNT (|replace| (INFO-REC POP-COUNT) |of| *INFO-REC* |with| |val|))
        (INFO-TYPE (|replace| (INFO-REC INFO-TYPE) |of| *INFO-REC* |with| |val|))
        NIL)))

)

;; Writeout Files

(DEFINEQ

## (MAKE-PROGRAM-FILE

```
  (LAMBDA (|file-name| |keep-cr-eol|)                           ; Edited 19-Apr-88 18:56 by Krivacic
    (LET ((|namec| (CONCAT |file-name| ".c"))
          (|nameil| (CONCAT |file-name| ".il")))
         (WRITE-PROGRAM-FILE |file-name| |namec| |keep-cr-eol|)
         (WRITE-INLINE-FILE |file-name| |nameil| |keep-cr-eol|))))
```

(WRITE-PROGRAM-FILE

```
  (LAMBDA (|file-name| |namec| |keep-cr-eol|)                    ; Edited 21-Jun-88 20:31 by rtk
    ;; Printout the inline assembly file
    (DECLARE (SPECIAL *CPROGRAM*))
    (LET* ((|full-file-name| (OR (AND (EQ 'MAIKO (MACHINETYPE))
                                      (CONCAT "{UNIX}" *NATIVE-TEMP-FILE-DIRECTORY* "/" |namec|))
                                 |namec|))
```
```

```
          (|stream| (OR (AND |file-name| (OPENSTREAM |full-file-name| 'OUTPUT 'NEW '((EOL LF))))
                        T)))
        (PRINTOUT *NATIVE-STREAM* "Writing Native File: " |full-file-name| T)
        (|if| (AND (NOT |keep-cr-eol|)
                   (NEQ |stream| T))
            |then| (SETFILEINFO |stream| 'EOL 'LF))
        (|for| |line| |in| (CAR *CPROGRAM*) |as| I |from| 0 |do| (PRINT-CODE-LINE |stream| |line|)
                                                                  (|if| (EQ 0 (LOGAND I 15))
                                                                      |then| (BLOCK)))
        (|if| (NEQ |stream| T)
            |then| (CLOSEF |stream|)))))
```

(WRITE-INLINE-FILE

```
(LAMBDA (|file-name| |nameil| |keep-cr-eol|)                    ; Edited 21-Jun-88 20:30 by rtk
  ;; Printout the inline assembly file
  (DECLARE (SPECIAL *INLINES* *CODE-SIZE* *ENTRY-POINT-MAX* *CODE-BASE* *START-PC*))
  (LET* ((|full-file-name| (OR (AND (EQ 'MAIKO (MACHINETYPE))
                                    (CONCAT "{UNIX}" *NATIVE-TEMP-FILE-DIRECTORY* "/" |nameil|))
                               |nameil|))
         (|stream| (OR (AND |file-name| (OPENSTREAM |full-file-name| 'OUTPUT 'NEW '((EOL LF))))
                       T)))
        (PRINTOUT *NATIVE-STREAM* "Writing Inline File: " |full-file-name| T)
        (|if| (AND (NOT |keep-cr-eol|)
                   (NEQ |stream| T))
            |then| (SETFILEINFO |stream| 'EOL 'LF))
        ;; Write the InLine Defs
        (|for| |line| |in| (CAR *INLINES*) |as| I |from| 0 |do| (PRINT-CODE-LINE |stream| |line|)
                                                                 (|if| (EQ 0 (LOGAND I 15))
                                                                     |then| (BLOCK)))
        ;; Write the Entry Point Table
        (PRINTOUT |stream| ".inline _entry_table_setup,0" T)
        (PRINTOUT |stream| "_entry_table:" T)
        (|for| \i |from| (MINUS *ENTRY-POINT-MAX*) |to| 0 |do| (PRINTOUT |stream| " .long case" (PC-XFORM \i)
                                                                          T))
        (PRINTOUT |stream| "  .long case" (PC-XFORM 1)
              T)
        (|for| \i |from| 2 |to| (SUB1 *CODE-SIZE*) |do| (LET ((ENTRY-POINT (ELT *ENTRY-POINTS* \i)))
                                                           (COND
                                                              ((OR (EQ \i (|fetch| (FNHEADER STARTPC) |of|
                                                                                                  *CODE-BASE*
                                                                                   ))
                                                                   (AND (TYPEP ENTRY-POINT 'BYTE-INFO-REC)
                                                                        (|fetch| (BYTE-INFO-REC ENTRY-ADDRESS)
                                                                          |of| ENTRY-POINT)))
                                                               (PRINTOUT |stream| "       .long case" (PC-XFORM
                                                                                                        \i)
                                                                      T))
                                                              (ENTRY-POINT (PRINTOUT |stream| "   .long
                                                                                           unknown_entry_point" T))
                                                              (T (PRINTOUT |stream| "     .long illegal_pc" T))))
                                                        (BLOCK))
        (PRINTOUT |stream| ".end" T)
        (PRINTOUT |stream| T)
        (PRINTOUT |stream| ".inline _entry_point_setup,0" T)
        (PRINTOUT |stream| "  unlk a6" T)
        (PRINTOUT |stream| "  lea entry_point,a0" T)
        (PRINTOUT |stream| "  movl a0," (OR (AND (EQ 'MAIKO (MACHINETYPE))
                                                 (SUBRCALL GET-NATIVE-ADDR-FROM-LISP-PTR *CODE-BASE*))
                                            (FN-OBJ *CODE-BASE*))
              " + " *START-PC* " - 4" T)
        (PRINTOUT |stream| "entry_point: " T)
        (PRINTOUT |stream| ".end" T)
        (PRINTOUT |stream| T)
        (|if| (NEQ |stream| T)
            |then| (CLOSEF |stream|)))))
```

(WRITE-INCLUDE-FILE

```
(LAMBDA (ERROR-PC-STREAM)                                       ; Edited 21-Jun-88 20:31 by rtk
  ;; Printout the include file of error exits
  (DECLARE (SPECIALS *INLINES* *CODE-SIZE*))
  (PRINTOUT ERROR-PC-STREAM T)
  (PRINTOUT ERROR-PC-STREAM "/* Error Exits */" T)
  (TERPRI ERROR-PC-STREAM)
  (TERPRI ERROR-PC-STREAM)
  (|for| |cases| |in| (CAR *ERROR-CASES*) |as| I |from| 0 |do| (PRINT-CODE-LINE ERROR-PC-STREAM |cases|)
                                                               (|if| (EQ 0 (LOGAND I 15))
                                                                   |then| (BLOCK)))
  (TERPRI ERROR-PC-STREAM)))
```

(PRINT-CODE-LINE

```
  (LAMBDA (STREAM LINE)                                    ; Edited 21-Jun-88 23:44 by rtk
    (COND
       ((TYPEP LINE 'LINE-RECORD-INFO)
        (COND
           ((EQ '|include-errors| (|fetch| (LINE-RECORD-INFO POSTFIX-STRING) |of| LINE))
            (WRITE-INCLUDE-FILE STREAM))
           (T (LET ((PREFIX (|fetch| (LINE-RECORD-INFO PREFIX-STRING) |of| LINE))
                    (LINE-INFO (|fetch| (LINE-RECORD-INFO LINE-INFO-LIST) |of| LINE))
                    (POSTFIX (|fetch| (LINE-RECORD-INFO POSTFIX-STRING) |of| LINE)))
                   (AND PREFIX (PRINTOUT STREAM PREFIX))
                   (AND LINE-INFO (PRINT-LINE-INFO STREAM LINE-INFO))
                   (AND POSTFIX (PRINTOUT STREAM POSTFIX))
                   (TERPRI STREAM)))))
       (T (PRINTOUT STREAM LINE T)))))
```

(PRINT-LINE-INFO

```
  (LAMBDA (STREAM LINE-INFO)                               ; Edited 21-Jun-88 19:29 by rtk
    (AND LINE-INFO
         (COND
            ((TYPEP LINE-INFO 'LINE-INFO-REC)
             (LET ((LINE-PATTERN (|fetch| (LINE-INFO-REC PATTERN-LIST) |of| LINE-INFO))
                   (FORMAL-PARMS (|fetch| (LINE-INFO-REC PARAMETER-LIST) |of| LINE-INFO))
                   (ACTUAL-PARMS (|fetch| (LINE-INFO-REC ACTUAL-PARAMETERS) |of| LINE-INFO)))
                  (|for| PAT |in| LINE-PATTERN
                     |do| (COND
                             ((LITATOM PAT)
                              ;; Must replace formal parm with actual parm
                              (LET ((ACTUAL-LINE (OR (|for| ACTUAL-PARM |in| ACTUAL-PARMS |as| FORMAL-PARM
                                                        |in| FORMAL-PARMS |thereis| (AND (EQ (CAR FORMAL-PARM)
                                                                                             PAT)))
                                                     PAT)))
                                   (|if| (TYPEP ACTUAL-LINE 'LINE-INFO-REC)
                                       |then| (PRINT-LINE-INFO STREAM ACTUAL-LINE)
                                     |else| (PRINTOUT STREAM ACTUAL-LINE))))
                             (T ;; Output the Current String
                                (PRINTOUT STREAM PAT))))))
            (T (PRINTOUT STREAM LINE-INFO))))))

)

;; Initialization

(DEFINEQ
```

(TRANSLATION-INIT

```
  (LAMBDA (|force-init|)                                   ; Edited 21-Jun-88 19:07 by rtk
    (DECLARE (SPECIAL *CPROGRAM* *ENTRY-POINTS* *TRANSLATION-TABLE* *NATIVE-TEMP-FILE-DIRECTORY*
                      IL:*NATIVE-INCLUDE-FILE-DIRECTORY* IL:*NATIVE-LISP-RUN-FILENAME* *NATIVE-BIN-DIRECTORY*))
    (SETQ *NATIVE-TEMP-FILE-DIRECTORY* (STRIP-ENDING-SLASH *NATIVE-TEMP-FILE-DIRECTORY*))
    (SETQ IL:*NATIVE-INCLUDE-FILE-DIRECTORY* (STRIP-ENDING-SLASH IL:*NATIVE-INCLUDE-FILE-DIRECTORY*))
    (SETQ *NATIVE-BIN-DIRECTORY* (STRIP-ENDING-SLASH *NATIVE-BIN-DIRECTORY*))
    (SETQ *CPROGRAM* NIL)
    (SETQ *ENTRY-POINTS* (ARRAY 4000 'POINTER NIL 0))
    (|if| (OR |force-init| (NOT (BOUNDP '*NATIVE-TRANSLATION-TABLE*)))
        |then| (SETQ *TRANSLATION-TABLE* (ARRAY 256 'POINTER NIL 0))
               (SETQ *NATIVE-TRANSLATION-TABLE* *TRANSLATION-TABLE*)
               ;; ARGS:  pattern may-ufn stack-args pushing-result defer-push parse-fn inline-exit-fn inline-expansion
               (LET ((ORDERING-LIST (MAKE-ORDERING-LIST))
                     (OPCODE-LIST (MAKE-OPCODE-LIST)))
                    (MAKE-TRANSLATION-ENTRIES OPCODE-LIST ORDERING-LIST))
      |else| (SETQ *TRANSLATION-TABLE* *NATIVE-TRANSLATION-TABLE*))))
```

(STRIP-ENDING-SLASH

```
  (LAMBDA (FILE-NAME)                                      ; Edited 17-Jun-88 18:45 by rtk
    (LET ((POS (STRPOS "/" FILE-NAME NIL NIL NIL NIL NIL T)))
         (COND
            ((EQ POS (NCHARS FILE-NAME))
             (SUBSTRING FILE-NAME 1 (- POS 1)))
            (T FILE-NAME)))))
```

(SETUP-TRANSLATION-FNS

```
  (LAMBDA NIL                                              ; Edited  2-May-88 18:01 by rtk
    (DECLARE (SPECIAL *BYTE-INFO-TABLE* *CODE-SIZE*))
    (SETQ *BYTE-INFO-TABLE* (ARRAY *CODE-SIZE* 'POINTER NIL 0))))
```

(MAKE-TRANSLATION-ENTRY

```
  (LAMBDA (|entry| ORDERING-LIST)                          ; Edited 22-Jun-88 00:30 by rtk
    (DECLARE (SPECIALS *TRANSLATION-TABLE*))
    (|if| (NEQ '* (CAR |entry|))
        |then| (DESTRUCTURING-BIND (|opcode| |pattern| |may-ufn| |stack-args| |pushing-result| |defer-push|
```

```
                                    |parse-fn| |inline-exit-fn| |inline-expansion|)
                        |entry|
                        (LET ((|opcode-info| (\\FINDOP |opcode|)))
                             (|if| |opcode-info|
                                 |then| (LET* ((|opcode-range| (|fetch| OP# |of| |opcode-info|))
                                              (|opcodes| (OR (AND (LISTP |opcode-range|)
                                                                  |opcode-range|)
                                                             (LIST |opcode-range| |opcode-range|)))
                                              (|level-adjust| (OR (|fetch| LEVADJ |of| |opcode-info|)
                                                                  0))
                                              (*ARG-COUNT* 0))
                                             (DECLARE (SPECVAR *ARG-COUNT*))
                                             (|bind| |tran-rec| |for| |opcode| |from| (CAR |opcodes|)
                                                |to| (CADR |opcodes|) |do| (SETQ |tran-rec| (|create| TRANSLATION-REC))
                                                                            (|replace| (TRANSLATION-REC MAY-UFN)
                                                                               |of| |tran-rec| |with| |may-ufn|)
                                                                            (|replace| (TRANSLATION-REC STACK-ARGS)
                                                                               |of| |tran-rec| |with| (OR |stack-args| 0))
                                                                            (|replace| (TRANSLATION-REC PUSHING-RESULT)
                                                                               |of| |tran-rec| |with| |pushing-result|)
                                                                            (|replace| (TRANSLATION-REC DEFER-PUSH)
                                                                               |of| |tran-rec| |with| |defer-push|)
                                                                            (|replace| (TRANSLATION-REC PATTERN)
                                                                               |of| |tran-rec| |with| |pattern|)
                                                                            (|replace| (TRANSLATION-REC STACK-ADJUST)
                                                                               |of| |tran-rec| |with| |level-adjust|)
                                                                            (|replace| (TRANSLATION-REC PARSE-FN)
                                                                               |of| |tran-rec| |with| (OR |parse-fn|
                                                                                                        'STR-PARSER))
                                                                            (|replace| (TRANSLATION-REC INLINE-EXIT-FN)
                                                                               |of| |tran-rec| |with| |inline-exit-fn|)
                                                                            (|replace| (TRANSLATION-REC INLINE-EXPANSIONS)
                                                                               |of| |tran-rec| |with| (MAKE-INLINE-LISTS
                                                                                                       |inline-expansion|
                                                                                                       ORDERING-LIST))
                                                                            (|replace| (TRANSLATION-REC POPPING-TOS)
                                                                               |of| |tran-rec| |with| (STRPOS "$(Tos)"
                                                                                                           |pattern|))
                                                                            (CL:MULTIPLE-VALUE-BIND (PATTERN-LIST
                                                                                                     PARAMETER-LIST
                                                                                                     )
                                                                               (MAKE-TRANSLATION-PATTERN-LIST
                                                                                  |pattern| ORDERING-LIST)
                                                                              (|replace| (TRANSLATION-REC TRANS-PATTERN)
                                                                                 |of| |tran-rec| |with| PATTERN-LIST)
                                                                              (|replace| (TRANSLATION-REC TRANS-PARAMATERS
                                                                                                          )
                                                                                 |of| |tran-rec| |with| PARAMETER-LIST))
                                                                            (SETA *TRANSLATION-TABLE* |opcode|
                                                                                  |tran-rec|)))
                                    |else| (PRINTOUT T "Opcode " |opcode| " not found." T)))))
    (BLOCK)))
```

(MAKE-TRANSLATION-ENTRIES

```
  (LAMBDA (|entry-list| ORDERING-LIST)                    ; Edited 21-Jun-88 19:07 by rtk
    (|for| |entry| |in| |entry-list| |do| (MAKE-TRANSLATION-ENTRY |entry| ORDERING-LIST))))
```

(MAKE-TRANSLATION-PATTERN-LIST

```
  (LAMBDA (STRING-PATTERN ORDERING-LIST)                  ; Edited 21-Jun-88 19:37 by rtk
```

;;; Turn the String into a list format used in the translator of

;;; (("str" | replacement-symbol) (ordering of replacement symbols))

```
    (LET ((PRINT-LIST NIL)
          (PARAMETER-LIST NIL)
          (DOLLAR-POS NIL)
          (ORDERED-PARAMETER-LIST NIL))
```

;; Break up the string into sections

```
         (SETQ PRINT-LIST (|while| (SETQ DOLLAR-POS (STRPOS "$" STRING-PATTERN))
                             |join| (LET* ((HEAD (SUBSTRING STRING-PATTERN 1 (SUB1 DOLLAR-POS)))
                                           (TEMP-TAIL (SUBSTRING STRING-PATTERN DOLLAR-POS))
                                           (FOUND-PARM (|for| PARM |in| ORDERING-LIST
                                                              |thereis| (EQ 1 (STRPOS (CAR PARM)
                                                                                      TEMP-TAIL))))
                                           (TAIL (SUBSTRING TEMP-TAIL (+ (NCHARS (CAR FOUND-PARM))
                                                                         1))))
                                          (SETQ PARAMETER-LIST (CONS FOUND-PARM PARAMETER-LIST))
                                          (SETQ STRING-PATTERN TAIL)
                                          (OR (AND HEAD (LIST HEAD (CADR FOUND-PARM)))
                                              (LIST (CADR FOUND-PARM))))))
         (|if| STRING-PATTERN
             |then| (SETQ PRINT-LIST (APPEND PRINT-LIST (LIST STRING-PATTERN))))
```

;; Order the Parameter List

```
        (|for| PARAMETER |in| (REVERSE ORDERING-LIST) |do| (|if| (FMEMB PARAMETER PARAMETER-LIST)
                                                                |then| (SETQ ORDERED-PARAMETER-LIST
                                                                       (CONS (CDR PARAMETER)
                                                                             ORDERED-PARAMETER-LIST))))
        ;; return ( ( pattern list) (parameter list))
        (CL:VALUES PRINT-LIST ORDERED-PARAMETER-LIST))))
```

(MAKE-INLINE-LISTS

```
(LAMBDA (EXPANSION-LIST ORDERING-LIST)                    ; Edited 21-Jun-88 19:07 by rtk
  (|for| MACHINE-TYPE-LIST |on| EXPANSION-LIST |by| (CDDR EXPANSION-LIST)
     |join| (LIST (CAR MACHINE-TYPE-LIST)
                  (|for| EXPANSION-STRING |in| (CADR MACHINE-TYPE-LIST) |collect| (CL:MULTIPLE-VALUE-BIND (V-HEAD
                                                                                                          V-TAIL)
                                                                                     (MAKE-TRANSLATION-PATTERN-LIST
                                                                                      EXPANSION-STRING ORDERING-LIST)
                                                                                     (LIST V-HEAD V-TAIL)))))))
```

(MAKE-OPCODE-LIST

```
(LAMBDA NIL                                               ; Edited 24-Jun-88 16:33 by rtk
    ;; ARGS:  pattern may-ufn stack-args pushing-result defer-push parse-fn inline-exit-fn inline-expansion
    '(;; Variable Reference
      (IVAR "IVAR[$op<3>]" NIL 0 T T STR-PARSER)
      (IVARX "IVAR[$x/2]" NIL 0 T T STR-PARSER)
      (IVARX_ "IVAR[$x/2] = $Tos " NIL 1 NIL NIL STR-PARSER)
      (PVAR "PVAR[$op<3>]" NIL 0 T T STR-PARSER)
      (PVAR_ "PVAR[$op<3>] = $Tos " NIL 1 NIL NIL PVAR_PARSER)
      (PVAR_^
       "PVAR[$op<3>] = $(Tos) " NIL 1 NIL NIL PVAR_PARSER)
      (PVARX "PVAR[$x/2]" NIL 0 T T STR-PARSER)
      (PVARX_ "PVAR[$x/2] = $Tos " NIL 1 NIL NIL PVAR_PARSER)
      (GVAR_ "N_OP_gvar_($Tos, $x16)" NIL 1 NIL NIL STR-PARSER)
      (GVAR " (GetLongWord(Valspace + $x16<<1))" NIL 0 T T STR-PARSER)
      (FVAR " N_OP_fvarn($op<3><<1)" NIL 0 T NIL STR-PARSER)
      (FVARX " N_OP_fvarn($x)" NIL 0 T NIL STR-PARSER)
      (FVARX_ "N_OP_fvar_($(Tos), $x)" NIL 1 T NIL STR-PARSER)
      ;; Stack Operations
      (COPY "COPY_TOP" NIL 1 T T COPY-PARSER)
      (SWAP "{LispPTR temp = TOS; TOS = PREV_TOS; PREV_TOS = temp;}" NIL 2 NIL NIL SWAP-PARSER)
      (POP "$(Tos)" NIL 1 NIL NIL STR-PARSER)
      (POP.N "CSTKPTR = CSTKPTR - ($x)" T 0 NIL NIL STR-PARSER)
      (COPY.N " *(CSTKPTR - ($x/2 + 1))" T 0 T NIL STR-PARSER)
      (STORE.N " *(CSTKPTR - ($x/2 + 1)) = TOS" T 1 T NIL STR-PARSER)
      (FINDKEY "N_OP_findkey($(Tos), $x)" NIL 1 T T STR-PARSER)
      (BIND "CSTKPTR = (LispPTR *) N_OP_bind(CSTKPTR, $(Tos), $x, $x2) + 1" T 1 NIL NIL STR-PARSER)
      (UNBIND "{register LispPTR SAVE_TOS = $(Tos); CSTKPTR = (LispPTR *) N_OP_unbind(CSTKPTR);
             PUSH(SAVE_TOS);}" T 0 NIL NIL STR-PARSER)
      (DUNBIND "CSTKPTR = (LispPTR *) N_OP_dunbind(CSTKPTR, $Tos)" T 1 NIL NIL STR-PARSER)
                                                                  (* UNWIND "CALL_OP_FN($bce-pc, $next-bce-pc,
                                                                  OP_unwind)" T 0 NIL NIL STR-PARSER)
      (UNWIND "CSTKPTR = (LispPTR *) N_OP_unwind($CSTKPTR, $(Tos), $x, $x2, $errorpc) + 1" T 1 NIL NIL
             STR-PARSER)
      (MYALINK " ((((NATIVE_CURRENTFX->alink) & 0xfffe) - FRAMESIZE) | S_POSITIVE)" NIL 0 T T STR-PARSER)
      (ARG0 "N_OP_arg0($(Tos), $errorpc)" T 0 T NIL STR-PARSER)
      (MYARGCOUNT "MYARGCOUNT" T 0 NIL NIL STR-PARSER NIL NIL)
      (STKSCAN "N_OP_stkscan($(Tos), $errorpc)" T 1 T NIL STR-PARSER)
      ;; Arithmetic Operations
      (DIFFERENCE "N_OP_difference($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER NIL
             (SUN3.N ("DIFFERENCE_N_$pc($(Tos-1))" "movl a7@+,d0" "moveq #15,d2" "roll d2,d0" "subqb #7,d0"
                           "bne $errorpc " "subl #$#(Tos<<15),d0 " "bvs $errorpc " "lsrl d2,d0" "orl
                           #0x000E0000,d0")
                     SUN3
                     ("DIFFERENCE_$pc($(Tos-1), $(Tos))" "movl a7@+,d0" "movl a7@+,d1" "moveq #15,d2" "roll
                            d2,d0" "subqb #7,d0" "bne $errorpc " "roll d2,d1" "subqb #7,d1" "bne $errorpc "
                            "subl d1,d0 " "bvs $errorpc " "lsrl d2,d0" "orl #0x000E0000,d0")))
      (IDIFFERENCE "N_OP_idifference($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER NIL
             (SUN3.N ("IDIFFERENCE_N_$pc($(Tos-1))" "movl a7@+,d0" "moveq #15,d2" "roll d2,d0" "subqb #7,d0"
                           "bne $errorpc " "subl #$#(Tos<<15),d0 " "bvs $errorpc " "lsrl d2,d0" "orl
                           #0x000E0000,d0")
                     SUN3
                     ("IDIFFERENCE_$pc($(Tos-1), $(Tos))" "movl a7@+,d0" "movl a7@+,d1" "moveq #15,d2" "roll
                            d2,d0" "subqb #7,d0" "bne $errorpc " "roll d2,d1" "subqb #7,d1" "bne $errorpc "
                            "subl d1,d0 " "bvs $errorpc " "lsrl d2,d0" "orl #0x000E0000,d0")))
      (IDIFFERENCE.N "N_OP_idifferencen($(Tos), $x, $errorpc)" T 1 T NIL STR-PARSER NIL
             (SUN3 ("IDIFFERENCE_N_$pc($(Tos-1))" "movl a7@+,d0" "moveq #15,d2" "roll d2,d0" "subqb #7,d0"
                          "bne $errorpc " "subl ##(n<<15),d0 " "bvs $errorpc " "lsrl d2,d0" "orl
                          #0x000E0000,d0")))
      (PLUS2 "N_OP_plus2($(Tos-1), $(Tos), $errorpc) " T 2 T NIL STR-PARSER NIL
             (SUN3.N ("PLUS_N_$pc($(Tos-1))" "movl a7@+,d0" "moveq #15,d2" "roll d2,d0" "subqb #7,d0" "bne
                           $errorpc " "addl #$#(Tos<<15),d0 " "bvs $errorpc " "lsrl d2,d0" "orl
```

```
                        #0x000E0000,d0")
                SUN3
                ("PLUS_$pc($(Tos-1), $(Tos))" "movl a7@+,d0" "movl a7@+,d1" "moveq #15,d2" "roll d2,d0"
                    "subqb #7,d0" "bne $errorpc " "roll d2,d1" "subqb #7,d1" "bne $errorpc " "addl d1,d0
" "bvs $errorpc " "lsrl d2,d0" "orl #0x000E0000,d0")))
    (IPLUS2 "N_OP_iplus2($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER NIL
        (SUN3.N ("PLUS_N_$pc($(Tos-1))" "movl a7@+,d0" "moveq #15,d2" "roll d2,d0" "subqb #7,d0" "bne
                    $errorpc " "addl #$#(Tos<<15),d0 " "bvs $errorpc " "lsrl d2,d0" "orl
                    #0x000E0000,d0")
                SUN3
                ("PLUS_$pc($(Tos-1), $(Tos))" "movl a7@+,d0" "movl a7@+,d1" "moveq #15,d2" "roll d2,d0"
                    "subqb #7,d0" "bne $errorpc " "roll d2,d1" "subqb #7,d1" "bne $errorpc " "addl d1,d0
" "bvs $errorpc " "lsrl d2,d0" "orl #0x000E0000,d0")))
    (IPLUS.N "N_OP_iplusn($(Tos), $x, $errorpc)" T 1 T NIL STR-PARSER NIL
        (SUN3 ("IÕ—_N_$pc($(Tos-1))" "movl a7@+,d0" "moveq #15,d2" "roll d2,d0" "subqb #7,d0" "bne $errorpc
" "addl #$#(n<<15),d0 " "bvs $errorpc " "lsrl d2,d0" "orl #0x000E0000,d0")))
    (QUOTIENT "N_OP_quot($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
    (IQUOTIENT "N_OP_iquot($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
    (TIMES2 "N_OP_times2($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
    (ITIMES2 "N_OP_itimes2($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
    (IREMAINDER "N_OP_iremainder($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
    (BOXIPLUS "N_OP_boxiplus($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
    (BOXIDIFFERENCE "N_OP_boxidifference($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
    (LOGAND2 "N_OP_logand($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER NIL
        (SUN3.N ("LOGAND_N_$pc($(Tos-1))" "movl a7@+,d0" "moveq #15,d2" "roll d2,d0" "cmpb #7,d0" "bne
                    $errorpc " "andl #$#(Tos<<15) + 7,d0 " "rorl d2,d0")
                SUN3
                ("LOGAND_$pc($(Tos-1), $(Tos))" "movl a7@+,d0" "movl a7@+,d1" "moveq #15,d2" "roll d2,d0"
                    "cmpb #7,d0" "bne $errorpc " "roll d2,d1" "cmpb #7,d1" "bne $errorpc " "andl d1,d0 "
                    "rorl d2,d0")))
    (LOGOR2 "N_OP_logor($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER NIL
        (SUN3.N ("LOGOR_N_$pc($(Tos-1))" "movl a7@+,d0" "moveq #15,d2" "roll d2,d0" "cmpb #7,d0" "bne
                    $errorpc " "orl #$#(Tos<<15),d0 " "rorl d2,d0")
                SUN3
                ("LOGOR_$pc($(Tos-1), $(Tos))" "movl a7@+,d0" "movl a7@+,d1" "moveq #15,d2" "roll d2,d0"
                    "cmpb #7,d0" "bne $errorpc " "roll d2,d1" "cmpb #7,d1" "bne $errorpc " "orl d1,d0 "
                    "rorl d2,d0")))
    (LOGXOR2 "N_OP_logxor($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER NIL
        (SUN3.N ("LOGXOR_N_$pc($(Tos-1))" "movl a7@+,d0" "moveq #15,d2" "roll d2,d0" "cmpb #7,d0" "bne
                    $errorpc " "eorl #$#(Tos<<15),d0 " "rorl d2,d0")
                SUN3
                ("LOGXOR_$pc($(Tos-1), $(Tos))" "movl a7@+,d0" "movl a7@+,d1" "moveq #15,d2" "roll d2,d0"
                    "cmpb #7,d0" "bne $errorpc " "roll d2,d1" "subqb #7,d1" "bne $errorpc " "eorl d1,d0 "
                    "rorl d2,d0")))

    ;; Shifts

    (LRSH8 "N_OP_lrsh8($(Tos), $errorpc)" T 1 T NIL STR-PARSER NIL (SUN3 ("LRSH8_$pc($(Tos))" "movl a7@+,d0"
                                                                        "movl d0,d1" "swap d1"
                                                                        "cmpw #0xe,d1" "bne $errorpc "
                                                                        "lsrw #8,d0")))
    (LRSH1 "N_OP_lrsh1($(Tos), $errorpc)" T 1 T NIL STR-PARSER NIL (SUN3 ("LRSH1_$pc($(Tos))" "movl a7@+,d0"
                                                                        "movl d0,d1" "swap d1" "bne
                                                                        $errorpc " "lsrw #1,d0")))
    (LLSH8 "N_OP_llsh8($(Tos), $errorpc)" T 1 T NIL STR-PARSER NIL
        (SUN3 ("LLSH8_$pc($(Tos))" "movl a7@+,d0" "movl d0,d1" "swap d1" "cmpw #0xE,d1" "bne $errorpc "
                    "cmpw #0x0ff,d0" "bhi $errorpc " "lslw #8,d0")))
    (LLSH1 "N_OP_llsh1($(Tos), $errorpc)" T 1 T NIL STR-PARSER NIL
        (SUN3 ("LLSH1_$pc($(Tos))" "movl a7@+,d0" "movl d0,d1" "swap d1" "cmpw #0xE,d1" "bne $errorpc "
                    "lslw #1,d0" "bcs $errorpc ")))
    (LSH "N_OP_lsh($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)

    ;; Constants

    (\'0 "$0" NIL 0 T T CONST-PARSER)
    (\'1 "$1" NIL 0 T T CONST-PARSER)
    (\'NIL "NIL_PTR" NIL 0 T T STR-PARSER)
    (\'T "ATOM_T" NIL 0 T T STR-PARSER)
    (SIC "$x" NIL 0 T T CONST-PARSER)
    (SNIC "$-x" NIL 0 T T CONST-PARSER)
    (SICX "$x16" NIL 0 T T CONST-PARSER)
    (ACONST "$a16" NIL 0 T T STR-PARSER)
    (GCONST "$g24" NIL 0 T T STR-PARSER)

    ;; Conditionals

    (GREATERP "(N_OP_greaterp($(Tos-1), $(Tos), $errorpc))" T 2 T NIL STR-PARSER NIL
        (SUN3.N ("GREATERP_N_$pc($(Tos-1))" "movl a7@+,d1" "moveq #15,d2" "roll d2,d1" "subqb #7,d1"
                    "bne $errorpc " "clrl d0" "cmpl #$#(Tos<<15),d1 " "ble gt_lab$pc" "moveq #76,d0"
                    "gt_lab$pc: ")
                SUN3
                ("GREATERP_$pc($(Tos-1), $(Tos))" "movl a7@+,d3" "movl a7@+,d1" "moveq #15,d2" "roll d2,d1"
                    "subqb #7,d1" "bne $errorpc " "roll d2,d3" "subqb #7,d3" "bne $errorpc " "clrl d0"
                    "cmpl d1,d3 " "ble gt_lab$pc" "moveq #76,d0" "gt_lab$pc: ")))
    (IGREATERP "(N_OP_igreaterp($(Tos-1), $(Tos), $errorpc))" T 2 T T STR-PARSER NIL
        (SUN3.N ("IGREATERP_N_$pc($(Tos-1))" "movl a7@+,d1" "moveq #15,d2" "roll d2,d1" "subqb #7,d1"
                    "bne $errorpc " "clrl d0" "cmpl #$#(Tos<<15),d1 " "ble igt_lab$pc" "moveq #76,d0"
                    "igt_lab$pc: ")
                SUN3
                ("IGREATERP_$pc($(Tos-1), $(Tos))" "movl a7@+,d3" "movl a7@+,d1" "moveq #15,d2" "roll
```

```
                        d2,d1" "subqb #7,d1" "bne $errorpc " "roll d2,d3" "subqb #7,d3" "bne $errorpc "
                        "clrl d0" "cmpl d1,d3 " "ble igt_lab$pc" "moveq #76,d0" "igt_lab$pc: ")))
(EQ "($(C2Tos) == $(Tos-1))" NIL 2 NIL T STR-PARSER)
(EQL "(N_OP_eqlop($(Tos-1), $(Tos), $errorpc))" T 2 T NIL STR-PARSER)
```

;; Type opcodes

```
(INSTANCEP "(N_OP_instancep($(Tos), $x16))" NIL 1 T NIL STR-PARSER)
(TYPEMASK.N "N_OP_TYPEMASK($x<<8)" T 1 NIL NIL STR-PARSER)
(DTEST "(N_OP_dtest($(Tos), $x16, $errorpc))" T 1 T NIL STR-PARSER)
(TYPECHECK "(N_OP_dtest($(Tos), $x16, $errorpc))" T 1 T NIL STR-PARSER)
(TYPEP "((DLword)GetTypeNumber($(C3Tos)) == $x)" NIL 1 T NIL COND-PARSER)
(NTYPX " (S_POSITIVE | (unsigned int)GetTypeNumber($(Tos)))" T 1 T T STR-PARSER)
(LISTP "((DLword)GetTypeNumber($(C3Tos)) == TYPE_LISTP)" NIL 1 T NIL COND-PARSER)
```

;; Jumps

```
(TJUMP "if ($(Tos)) { goto pc$jt; }" NIL 1 NIL NIL JUMP-PARSER)
(FJUMP "if (!($(Tos))) { goto pc$jt; }" NIL 1 NIL NIL JUMP-PARSER)
(TJUMPX "if ($(Tos)) { goto pc$jt; }" NIL 1 NIL NIL JUMP-PARSER)
(FJUMPX "if (!($(Tos))) { goto pc$jt; }" NIL 1 NIL NIL JUMP-PARSER)
(NFJUMPX "if (!($Tos)) { goto pc$jt; } else POP" NIL 1 NIL NIL JUMP-PARSER)
(NTJUMPX "if ($Tos) { goto pc$jt; } else POP" NIL 1 NIL NIL JUMP-PARSER)
(JUMP "goto pc$jt " T 0 NIL NIL JUMP-PARSER)
(JUMPX "goto pc$jt " T 0 NIL NIL JUMP-PARSER)
(JUMPXX "goto pc$jt " T 0 NIL NIL JUMP-PARSER)
(-X- "{}" NIL NIL NIL NIL STR-PARSER)
(NOP "{}" NIL NIL NIL NIL STR-PARSER)
```

;; Function call & return

```
(FN0 "fncall_$who(0, $fn-call-args)" T 0 T NIL FN-CALL-PARSER)
(FN1 "fncall_$who(1, $fn-call-args)" T 1 T NIL FN-CALL-PARSER)
(FN2 "fncall_$who(2, $fn-call-args)" T 2 T NIL FN-CALL-PARSER)
(FN3 "fncall_$who(3, $fn-call-args)" T 3 T NIL FN-CALL-PARSER)
(FN4 "fncall_$who(4, $fn-call-args)" T 4 T NIL FN-CALL-PARSER)
(FNX "fncall_$who($x, $fn-call-args)" T 3 T NIL FN-CALL-PARSERX)
(RETURN "IVAR[0] = $(Tos); return_op($bce-pc, $swapped-fn-obj)" T 1 NIL NIL STR-PARSER)
(APPLYFN "RETURN_TO_FN_CALL($bce-pc, ret_to_apply)" T 2 NIL NIL STR-PARSER)
(ENVCALL "RETURN_TO_FN_CALL($bce-pc, ret_to_envcall)" T 3 T NIL ENVCALL-PARSER)
(CHECKAPPLY* "N_OP_CHECKAPPLY($Tos, $bce-pc)" T 1 NIL NIL STR-PARSER)
```

;; Pointer Operations

```
(GETBASEPTR.N " GETBASEPTR_N($(Tos),$x)" NIL 1 T T STR-PARSER NIL NIL)
(ADDBASE "BCE($bce-pc, $op)" T 2 T NIL STR-PARSER NIL
       (SUN3.N ("ADDBASE_N_$pc($(Tos-1))" "movl a7@+,d0" "andl #0xFFFFFF,d0" "addl #$#(Tos),d0 ")
              SUN3
              ("ADDBASE_$pc($(Tos-1), $(Tos))" "movl a7@+,d0" "movl a7@+,d1" "moveq #15,d2" "roll d2,d1"
                     "subqb #7,d1" "bne $errorpc " "asrl d2,d1" "andl #0xFFFFFF,d0" "addl d1,d0 ")))
(GETBASE.N " GETBASE_N($(Tos),$x)" NIL 1 T T STR-PARSER NIL NIL)
(PUTBASE.N "N_OP_putbasen($(Tos-1), $(Tos), $x, $errorpc)" T 2 T NIL STR-PARSER)
(PUTBASEPTR.N "N_OP_putbaseptrn($(Tos-1), $(Tos), $x, $errorpc)" T 2 T NIL STR-PARSER NIL NIL)
(PUTBITS.N.FD "N_OP_putbitsnfd($(Tos-1), $(Tos), $x, $x2, $errorpc)" T 2 T NIL STR-PARSER NIL NIL)
(GETBITS.N.FD "N_OP_getbitsnfd($(Tos), $x, $x2)" T 2 T NIL STR-PARSER)
(GETBASEBYTE "N_OP_getbasebyte($(Tos-1), $(Tos),$errorpc)" T 2 T NIL STR-PARSER)
(PUTBASEBYTE "N_OP_putbasebyte($(Tos-2), $(Tos-1), $(Tos),$errorpc)" T 3 T NIL STR-PARSER)
(RPLPTR.N "N_OP_rplptr($(Tos-1), $(Tos), $x)" NIL 2 T NIL STR-PARSER)
(ATOMCELL.N "N_OP_atomcellN($(Tos), $x, $errorpc)" T 1 T NIL STR-PARSER)
(VAG2 "*** need asm code error****" NIL 2 T NIL STR-PARSER NIL (SUN3 ("VAG2_$pc($(Tos-1), $(Tos))"
                                                                    "movl a7@+,d0" "movl a7@+,d1"
                                                                    "swap d0" "clrw d0" "movw
                                                                    d1,d0")))
(HILOC "( S_POSITIVE | (((unsigned int) $(Tos)) >> 16) )" NIL 1 T T STR-PARSER)
(LOLOC "( S_POSITIVE | (((unsigned int) $(Tos)) & 0xffff) )" NIL 1 T T STR-PARSER)
```

;; List Operations

```
(CAR "N_OP_CAR($(Tos), $error-label)" T 2 T NIL STR-PARSER)
(CDR "N_OP_cdr($(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(CONS " N_OP_cons($(Tos-1), $(Tos))" NIL 2 T NIL STR-PARSER)
(RPLCONS "N_OP_rplcons($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(RPLACA "N_OP_rplaca($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(RPLACD "N_OP_rplacd($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(FMEMB "N_OP_fmemb($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(LISTGET "N_OP_listget($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(ASSOC "N_OP_assoc($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(CMLASSOC "N_OP_classoc($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(CMLMEMBER "N_OP_clfmemb($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
```

;; Array Opcodes

```
(AREF1 "N_OP_aref1($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(AREF2 "N_OP_aref2($(Tos-2), $(Tos-1), $(Tos), $errorpc)" T 3 T NIL STR-PARSER)
(ASET1 "N_OP_aset1($(Tos-2), $(Tos-1), $(Tos), $errorpc)" T 3 T NIL STR-PARSER)
(ASET2 "N_OP_aset2($(Tos-3), $(Tos-2), $(Tos-1), $(Tos), $errorpc)" T 4 T NIL STR-PARSER)
```

;; Other Opcodes

```
(DRAWLINE "N_OP_drawline($(Tos-8), $(Tos-7), $(Tos-6), $(Tos-5), $(Tos-4), $(Tos-3), $(Tos-2), $(Tos-1),
       $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(BLT "N_OP_blt($(Tos-2), $(Tos-1), $(Tos), $errorpc)" T 3 T NIL STR-PARSER)
(MAKENUMBER "N_OP_makenumber($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(BIN "N_OP_bin($(Tos), $errorpc)" T 2 T NIL STR-PARSER)
```

```
(RCLK "N_OP_rclk($(Tos))" T 1 T NIL STR-PARSER)
(CREATECELL "N_OP_createcell($(Tos), $errorpc)" T 1 T NIL STR-PARSER)
(PILOTBITBLT "N_OP_pilotbitblt($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
```

;; Misc Opcodes

```
(MISC1 NIL T 1 NIL NIL BCE-PARSER)
(MISC2 NIL T 2 NIL NIL BCE-PARSER)
(MISC3 "PUSH(N_OP_misc3($(Tos-2), $(Tos-1), $(Tos), $x,  $errorpc))" T 3 T NIL STR-PARSER)
(MISC4 "PUSH(N_OP_misc4($(Tos-3), $(Tos-2), $(Tos-1), $(Tos), $x,  $errorpc))" T 4 T NIL STR-PARSER)
(MISC7 NIL T 7 NIL NIL BCE-PARSER)
(MISC8 NIL T 8 NIL NIL BCE-PARSER)
(MISC10 NIL T 10 NIL NIL BCE-PARSER)
(UBFLOAT3 "CALL_OP_FN($bce-pc, $next-bce-pc, OP_ubfloat3)" T 3 NIL NIL STR-PARSER)
(RECLAIMCELL "CALL_OP_FN($bce-pc, $next-bce-pc, OP_reclaimcell)" T 2 T NIL STR-PARSER)
(GCSCAN1 "CALL_OP_FN($bce-pc, $next-bce-pc, OP_gcscan1)" T 2 T NIL STR-PARSER)
(GCSCAN2 "CALL_OP_FN($bce-pc, $next-bce-pc, OP_gcscan2)" T 1 T NIL STR-PARSER)
(GCREF "CALL_OP_FN($bce-pc, $next-bce-pc, OP_gcref)" T 1 T NIL STR-PARSER)
(FQUOTIENT "N_OP_fquotient($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(FTIMES2 "N_OP_ftimes2($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(FDIFFERENCE "N_OP_fdifference($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(FPLUS2 "N_OP_fplus2($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(FGREATERP "N_OP_fgreaterp($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(EQUAL "N_OP_equal($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(CMLEQUAL "N_OP_clequal($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(= "N_OP_eqq($(Tos-1), $(Tos), $errorpc)" T 2 T NIL STR-PARSER)
(SUBRCALL "CALL_OP_FN($bce-pc, $next-bce-pc, OP_subrcall)" T 0 NIL NIL STR-PARSER)
```

;; Unimplemented Opcodes

```
(BUSBLT NIL T 0 NIL NIL BCE-PARSER)
(RDPROLOGPTR NIL T 0 NIL NIL BCE-PARSER)
(RDPROLOGTAG NIL T 0 NIL NIL BCE-PARSER)
(WRTPTR&TAG NIL T 0 NIL NIL BCE-PARSER)
(WRTPTR&0TAG NIL T 0 NIL NIL BCE-PARSER)
(DOVEMISC NIL T 0 NIL NIL BCE-PARSER)
(RAID NIL T 0 NIL NIL BCE-PARSER)
(\\RETURN NIL T 0 NIL NIL BCE-PARSER)
(READFLAGS NIL T 0 NIL NIL BCE-PARSER)
(READRP NIL T 0 NIL NIL BCE-PARSER)
(WRITEMAP NIL T 0 NIL NIL BCE-PARSER)
(READPRINTERPORT NIL T 0 NIL NIL BCE-PARSER)
(WRITEPRINTERPORT NIL T 0 NIL NIL BCE-PARSER)
(RETCALL NIL T 0 NIL NIL BCE-PARSER)
(FLOATBLT NIL T 0 NIL NIL BCE-PARSER)
(FFTSTEP NIL T 0 NIL NIL BCE-PARSER)
(UPCTRACE NIL T 0 NIL NIL BCE-PARSER)
(UBFLOAT1 NIL T 1 NIL NIL BCE-PARSER)
(UBFLOAT2 NIL T 2 NIL NIL BCE-PARSER)
(POPDISP NIL T 2 T NIL BCE-PARSER)
(RESTLIST NIL T 2 T NIL BCE-PARSER)
(CONTEXTSWITCH NIL T 2 T NIL BCE-PARSER)
(EVAL NIL T 2 T NIL BCE-PARSER)
(P-MISC2 NIL T 2 T NIL BCE-PARSER)
(BOUT NIL T 2 T NIL BCE-PARSER)
(BASE-< NIL T 2 T NIL BCE-PARSER))))
```

(**MAKE-ORDERING-LIST**

```
(LAMBDA NIL                                                    ; Edited 24-Jun-88 16:37 by rtk
  `(("$CSTKPTR" CSTKPTR ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                             'CSTKPTR))
    ("$Tos" TOS ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                     (TOS-CHECK TRANS-REC BYTE-REC PATTERN-LIST)))
    ("$(Tos)" POP ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                       'POP)
          ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
               (DECLARE (SPECIAL *ERROR-STACK*))
               (|push| *ERROR-STACK* (GET-VAL REPLACEMENT-VALUE STR-INFO))
               REPLACEMENT-VALUE))
    ("$(C2Tos)" POP ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                         'POP)
          ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
               (DECLARE (SPECIAL *ERROR-STACK* *ADD-HEAD* *ADD-TAIL*))
               (|push| *ERROR-STACK* (GET-VAL REPLACEMENT-VALUE STR-INFO))
               (COND
                  ((NEQ (|fetch| (BYTE-INFO-REC LEVEL-ADJUST) |of| (|fetch| (BYTE-INFO-REC NEXT-BYTE-REC)
                                                                     |of| BYTE-REC))
                        'CJUMP)
                   (SETQ *ADD-HEAD* "(")
                   (SETQ *ADD-TAIL* " ? ATOM_T : NIL_PTR)"))
                  (T NIL))
               REPLACEMENT-VALUE))
    ("$(C3Tos)" POP ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                         (COND
                            ((EQ (|fetch| (BYTE-INFO-REC LEVEL-ADJUST) |of| (|fetch| (BYTE-INFO-REC NEXT-BYTE-REC)
                                                                               |of| BYTE-REC))
                                 'CJUMP)
                             'POP)
```

```
                          (T (PUSH-ALL-OPERANDS)
                             'TOS)))
        ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
             (DECLARE (SPECIAL *ERROR-STACK* *ADD-HEAD* *ADD-TAIL*))
             (COND
                ((EQ (|fetch| (BYTE-INFO-REC LEVEL-ADJUST) |of| (|fetch| (BYTE-INFO-REC NEXT-BYTE-REC)
                                                                    |of| BYTE-REC))
                     'CJUMP)
                 (|push| *ERROR-STACK* (GET-VAL REPLACEMENT-VALUE STR-INFO)))
                (T (SETQ *ADD-HEAD* "      if (!")
                   (SETQ *ADD-TAIL* ") TOS = NIL_PTR;")))
             REPLACEMENT-VALUE))
("$#(Tos)" TOS-IMM ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                          (CL:MULTIPLE-VALUE-BIND (TOS-VAL TOS-INFO)
                              (OPERAND-POP T)
                            (SETQ *TOS-INFO* TOS-INFO)
                            (SETQ *TOS-VAL* TOS-VAL)
                            TOS-VAL)))
("$#(Tos<<15)" TOS<<15 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                              (CL:MULTIPLE-VALUE-BIND (TOS-VAL TOS-INFO)
                                  (OPERAND-POP T)
                                (SETQ *TOS-INFO* TOS-INFO)
                                (SETQ *TOS-VAL* TOS-VAL)
                                (|push| *ERROR-STACK* (GET-VAL TOS-VAL TOS-INFO))
                                TOS-VAL))
        ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
             (DECLARE (SPECIAL *ERROR-STACK*))
             (GET-SHIFTED-VAL REPLACEMENT-VALUE 15)))
("$(Tos-1)" POP-1 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                        'POP)
        ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
             (DECLARE (SPECIAL *ERROR-STACK*))
             (|push| *ERROR-STACK* (GET-VAL REPLACEMENT-VALUE STR-INFO))
             REPLACEMENT-VALUE))
("$(Tos-2)" POP-2 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                        'POP)
        ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
             (DECLARE (SPECIAL *ERROR-STACK*))
             (|push| *ERROR-STACK* (GET-VAL REPLACEMENT-VALUE STR-INFO))
             REPLACEMENT-VALUE))
("$(Tos-3)" POP-3 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                        'POP)
        ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
             (DECLARE (SPECIAL *ERROR-STACK*))
             (|push| *ERROR-STACK* (GET-VAL REPLACEMENT-VALUE STR-INFO))
             REPLACEMENT-VALUE))
("$(Tos-4)" POP-4 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                        'POP)
        ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
             (DECLARE (SPECIAL *ERROR-STACK*))
             (|push| *ERROR-STACK* (GET-VAL REPLACEMENT-VALUE STR-INFO))
             REPLACEMENT-VALUE))
("$(Tos-5)" POP-5 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                        'POP)
        ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
             (DECLARE (SPECIAL *ERROR-STACK*))
             (|push| *ERROR-STACK* (GET-VAL REPLACEMENT-VALUE STR-INFO))
             REPLACEMENT-VALUE))
("$(Tos-6)" POP-6 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                        'POP)
        ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
             (DECLARE (SPECIAL *ERROR-STACK*))
             (|push| *ERROR-STACK* (GET-VAL REPLACEMENT-VALUE STR-INFO))
             REPLACEMENT-VALUE))
("$(Tos-7)" POP-7 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                        'POP)
        ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
             (DECLARE (SPECIAL *ERROR-STACK*))
             (|push| *ERROR-STACK* (GET-VAL REPLACEMENT-VALUE STR-INFO))
             REPLACEMENT-VALUE))
("$(Tos-8)" POP-8 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                        'POP)
        ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
             (DECLARE (SPECIAL *ERROR-STACK*))
             (|push| *ERROR-STACK* (GET-VAL REPLACEMENT-VALUE STR-INFO))
             REPLACEMENT-VALUE))
("$op<3><<1" OP<3><<1 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                            (LLSH (LOGAND (|fetch| (BYTE-INFO-REC OPCODE) |of| BYTE-REC)
                                          7)
                                  1)))
("$op<3>" OP<3> ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                      (LOGAND (|fetch| (BYTE-INFO-REC OPCODE) |of| BYTE-REC)
                              7)))
("$op" OP ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                (|fetch| (BYTE-INFO-REC OPCODE) |of| BYTE-REC)))
("$x/2" X/2 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                  (LRSH (|fetch| (BYTE-INFO-REC ARG1) |of| BYTE-REC)
```

```
                              1)))
      ("$x2" X2 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                     (|fetch| (BYTE-INFO-REC ARG2) |of| BYTE-REC)))
      ("$x16<<1" X16<<1 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                            (LLSH (LOGOR (LLSH (|fetch| (BYTE-INFO-REC ARG1) |of| BYTE-REC)
                                               8)
                                         (|fetch| (BYTE-INFO-REC ARG2) |of| BYTE-REC))
                                  1)))
      ("$x<<8" X<<8 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                         (LLSH (|fetch| (BYTE-INFO-REC ARG1) |of| BYTE-REC)
                               8)))
      ("$x16" X16 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                       (LOGOR (LLSH (|fetch| (BYTE-INFO-REC ARG1) |of| BYTE-REC)
                                    8)
                              (|fetch| (BYTE-INFO-REC ARG2) |of| BYTE-REC))))
      ("$-x" -X ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                     (MINUS (ADD1 (LOGXOR (|fetch| (BYTE-INFO-REC ARG1) |of| BYTE-REC)
                                          255)))))
      ("$x" X ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                   (|fetch| (BYTE-INFO-REC ARG1) |of| BYTE-REC)))
      ("$a16" SYMBOL-INDEX ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                                (SET-INFO 'INFO-TYPE 'ACONST)
                                (LOGOR (LLSH (|fetch| (BYTE-INFO-REC ARG1) |of| BYTE-REC)
                                             8)
                                       (|fetch| (BYTE-INFO-REC ARG2) |of| BYTE-REC))))
      ("$g24" GCONST-PTR ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                              (SET-INFO 'INFO-TYPE 'GCONST)
                              (LOGOR (LLSH (|fetch| (BYTE-INFO-REC ARG1) |of| BYTE-REC)
                                           16)
                                     (LLSH (|fetch| (BYTE-INFO-REC ARG2) |of| BYTE-REC)
                                           8)
                                     (|fetch| (BYTE-INFO-REC ARG3) |of| BYTE-REC))))
      ("$jt" JUMP-TARGET ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                              (|fetch| (BYTE-INFO-REC JUMP-TO-ADDRESS) |of| BYTE-REC)))
      ("$swapped-fn-obj" SWAPPED-FN-OBJECT ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                                                (DECLARE (SPECIAL *CODE-BASE*))
                                                (SWAPPED-FN-OBJ *CODE-BASE*)))
      ("$pc" PC ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                     (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC)))
      ("$bce-pc" BCE-PC ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                             (DECLARE (CL:SPECIAL *CODE-BASE*))
                             (BCE-PC (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC)
                                     *CODE-BASE*)))
      ("$next-bce-pc" NEXT-BCE-PC ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                                       (DECLARE (CL:SPECIAL *CODE-BASE*))
                                       (+ (|fetch| (BYTE-INFO-REC OPLENGTH) |of| BYTE-REC)
                                          (BCE-PC (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC)
                                                  *CODE-BASE* T)
                                          1)))
      ("$errorpc" ERRORPC ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                               (CONCAT "errorpc" (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC)))
            ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
                 (DECLARE (CL:SPECIAL *ERROR-PC*))
                 (SETQ *ERROR-PC* T)
                 REPLACEMENT-VALUE))
      ("$error-label" ERRORPC ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                                   (CONCAT "errorpc" (|fetch| (BYTE-INFO-REC PC) |of| BYTE-REC)))
            ,#'(LAMBDA (TRANS-REC BYTE-REC PARAMETER REPLACEMENT-VALUE STR-INFO)
                 (DECLARE (CL:SPECIAL *ERROR-PC* *ARG-COUNT*))
                 (SETQ *ERROR-PC* T)
                 (SETQ *ARG-COUNT* 0)
                 REPLACEMENT-VALUE))
      ("$who" WHO-CALLED ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                              (DECLARE (CL:SPECIAL *CALL-SELF*))
                              (|if| *CALL-SELF*
                                  |then| "self"
                                |else| "other")))
      ("$fn-call-args" FN-CALL-ARGS ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                                         (DECLARE (CL:SPECIAL *FN-CALL-STR*))
                                         *FN-CALL-STR*))
      ("$0" \'0 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                     0))
      ("$1" \'1 ,#'(LAMBDA (TRANS-REC BYTE-REC PATTERN-LIST)
                     1)))))

)

;; Opcode Verification Fns

(DEFINEQ

(VERIFY-OPCODES
  (LAMBDA NIL                                                    ; Edited 31-May-88 18:01 by rtk
    (LET (*TRANSLATION-TABLE* *BYTE-INFO-TABLE* (*CODE-SIZE* 0))
         (DECLARE (SPECVARS *TRANSLATION-TABLE* *CODE-SIZE* *BYTE-INFO-TABLE*))
         (TRANSLATION-INIT T)
         (|for| |opcode| |in| \\OPCODES |do| (VERIFY-OPCODE (|fetch| (OP#) |of| |opcode|)
```

```
                                        |opcode|)))))
```

(VERIFY-OPCODE

```
  (LAMBDA (|opcode| |opcode-rec|)                          ; Edited 13-Apr-88 15:54 by rtk
    (COND
       ((LISTP |opcode|)
        (|for| |ops| |from| (CAR |opcode|) |to| (CADR |opcode|) |do| (VERIFY-OPCODE |ops| |opcode-rec|)))
       (T (|if| (NOT (ELT *TRANSLATION-TABLE* |opcode|))
              |then| (|if| (AND (NULL (STRPOS "was" (|fetch| (OPCODENAME) |of| |opcode-rec|)))
                                (NULL (STRPOS "unused" (|fetch| (OPCODENAME) |of| |opcode-rec|))))
                           |then| (PRINTOUT T "Opcode: " (|fetch| (OPCODENAME) |of| |opcode-rec|)
                                           " Missing: "
                                           (|fetch| (OP#) |of| |opcode-rec|)
                                           T)))))))

)

;; New Code Block Fns

(DEFINEQ

(LOADNATIVE
  (LAMBDA (|f-name| |file-name| ENTRY-PT-NAME |fn-obj| OLD-FN-OBJECT)
                                                           ; Edited 21-Jun-88 18:07 by rtk
    (PRINTOUT *NATIVE-STREAM* "Compile to Native: " |fn| " File: " |file-name| T)
    (LET* ((|full-file-name-no-brackets| (CONCAT *NATIVE-TEMP-FILE-DIRECTORY* "/" |file-name|))
           (|lisp-full-file-name| (CONCAT "{UNIX}" |full-file-name-no-brackets|))
           (|lisp-full-file-name.o| (CONCAT |lisp-full-file-name| ".o"))
           |relocatable-stream| |file-size| (|native-code-block-ptr| 0)
           (|native-code-addr| 0)
           |hex-load-addr| |load-request-result| (|load-file-size| 0)
           (NATIVE-ENTRY-PT-ADDR 0))
          (AND (|if| (EQ (MACHINETYPE)
                         'MAIKO)
                     |then| (AND ;; Execute the Unix C Compiler
                                 (DO-EXEC-COMMAND (CL:FORMAT NIL "~a/cc -c -pipe -O ~a.c -o ~a.o ~a.il
                                                                 ~a/disp68K.il -I~a" *NATIVE-BIN-DIRECTORY*
                                                                 |full-file-name-no-brackets|
                                                                 |full-file-name-no-brackets|
                                                                 |full-file-name-no-brackets|
                                                                 IL:*NATIVE-INCLUDE-FILE-DIRECTORY*
                                                                 IL:*NATIVE-INCLUDE-FILE-DIRECTORY*))
                                 ;; Remove the Temp Files
                                 (OR *KEEP-NATIVE-SOURCES* (AND *REMOVE-TEMP-NATIVE-FILES*
                                                                (DO-EXEC-COMMAND (CL:FORMAT NIL "rm ~a.c ~a.il"
                                                                                            |full-file-name-no-brackets|

                                                                                            |full-file-name-no-brackets|
                                                                                              )))
                                     T)
                                 ;; Get the Object File SIze
                                 (SETQ |file-size| (GET-NATIVE-LOAD-SIZE |lisp-full-file-name.o|))
                                 ;; Allocate a block big enough to hold the object
                                 (SETQ |native-code-block-ptr| (\\ALLOCBLOCK (FOLDHI |file-size| BYTESPERCELL)
                                                                             UNBOXEDBLOCK.GCT CELLSPERQUAD CELLSPERQUAD))
                                 (SETQ |native-code-addr| (LISP-ADDR-TO-NATIVE-ADDR |native-code-block-ptr|))
                                 ;; Execute the Unix Linker
                                                                  (* DO-EXEC-COMMAND (CL:FORMAT NIL "~a/ld -N -s -e _~a
                                                                  -Ttext ~x -A ~a ~a.o -o ~a -lc" *NATIVE-BIN-DIRECTORY*
                                                                  ENTRY-PT-NAME |native-code-addr|
                                                                  IL:*NATIVE-LISP-RUN-FILENAME* |full-file-name-no-brackets|
                                 |full-file-name-no-brackets|))
                                 (DO-EXEC-COMMAND (CL:FORMAT NIL "~a/ld -N -s -Ttext ~x -A ~a ~a.o -o ~a -lc"
                                                             *NATIVE-BIN-DIRECTORY* |native-code-addr|
                                                             IL:*NATIVE-LISP-RUN-FILENAME*
                                                             |full-file-name-no-brackets|
                                                             |full-file-name-no-brackets|))
                                 (PROGN (PRINTOUT *NATIVE-STREAM* "Load " |file-name| " At " |native-code-addr| "
                                                for " |file-size| " bytes." T)
                                        T)
                                 ;; Remove the Temp .o File
                                 (OR (AND *REMOVE-TEMP-NATIVE-FILES* (DO-EXEC-COMMAND (CL:FORMAT NIL "rm ~a.o"
                                                                                      |full-file-name-no-brackets|
                                                                                             )))
                                     T)
                                 ;; Load the code into lisp space
                                 (SETQ NATIVE-ENTRY-PT-ADDR (LOAD-NATIVE-FILE |lisp-full-file-name|
                                                                              |native-code-block-ptr| NIL))
                                 ;; Remove the Temp File
```

```
                        (OR (AND *REMOVE-TEMP-NATIVE-FILES* (DO-EXEC-COMMAND (CL:FORMAT NIL "rm ~a"
                                                                                  |full-file-name-no-brackets|
                                                                                  )))
                            T))
                |else| ;; Allocate a dummy block if not maiko
                       (SETQ |native-code-block-ptr| (\\ALLOCBLOCK (FOLDHI *CODE-SIZE* BYTESPERCELL)
                                                                   UNBOXEDBLOCK.GCT CELLSPERQUAD CELLSPERQUAD)))
          ;; Set Native Adder in Fn Object
          (SET-NATIVE-ADDR |f-name| |fn-obj| |native-code-block-ptr| NATIVE-ENTRY-PT-ADDR)
          ;; Add the New GCONST xx POP opcodes
          (ADD-GCONST |fn-obj| 0 |native-code-block-ptr|)
          (ADD-GCONST |fn-obj| 6 OLD-FN-OBJECT)
          ;; Set the New Function Definition
          (SET-NEW-FUNCTION-DEF |f-name| |fn-obj| *NATIVE-STREAM*)
          |fn-obj|)))))
```

(GET-NATIVE-LOAD-SIZE

```
(LAMBDA (FILE-NAME)                                    ; Edited 17-Jun-88 17:00 by rtk
  ;; Return the Size of Block needed for the Native Code Object
  (LET ((FN-HEADER-INFO (CREATE NATIVE-LINKER-INFO))
        FILE-STREAM)
       (AND (SETQ FILE-STREAM (OPENSTREAM FILE-NAME 'INPUT))
            (\\BINS FILE-STREAM FN-HEADER-INFO 0 (FETCH (NATIVE-LINKER-INFO RECORD-SIZE) OF FN-HEADER-INFO))
            (CLOSEF FILE-STREAM)
            (IPLUS (FETCH (NATIVE-LINKER-INFO DATA-SIZE) OF FN-HEADER-INFO)
                   (FETCH (NATIVE-LINKER-INFO TEXT-SIZE) OF FN-HEADER-INFO)
                   (FETCH (NATIVE-LINKER-INFO BSS-SIZE) OF FN-HEADER-INFO))))))
```

(LOAD-NATIVE-FILE

```
(LAMBDA (FILE-NAME LOAD-PTR INITIAL-BYTES)             ; Edited 17-Jun-88 17:24 by rtk
  (LET ((FN-HEADER-INFO (CREATE NATIVE-LINKER-INFO))
        FILE-STREAM LOAD-SIZE)
       (AND (SETQ FILE-STREAM (OPENSTREAM FILE-NAME 'INPUT))
            (\\BINS FILE-STREAM FN-HEADER-INFO 0 (FETCH (NATIVE-LINKER-INFO RECORD-SIZE) OF FN-HEADER-INFO))
            (SETQ LOAD-SIZE (IPLUS (FETCH (NATIVE-LINKER-INFO DATA-SIZE) OF FN-HEADER-INFO)
                                   (FETCH (NATIVE-LINKER-INFO TEXT-SIZE) OF FN-HEADER-INFO)))
            (PROGN (PRINTOUT *NATIVE-STREAM* "Load: " FILE-NAME " at " LOAD-PTR " for " LOAD-SIZE " Entry PT:
" (FETCH (NATIVE-LINKER-INFO ENTRY-POINT) OF FN-HEADER-INFO)
                              T)
                   T)
            (\\BINS FILE-STREAM LOAD-PTR (LENGTH INITIAL-BYTES)
                   LOAD-SIZE)
            (CLOSEF FILE-STREAM)
            ;; Add in intiial code
            (PROGN (IF INITIAL-BYTES
                       THEN ;; Add the ENTRY - PT to end of INTIIAL BYTES
                            (SETQ INITIAL-BYTES (APPEND INITIAL-BYTES (UNPACK-NUMBER (FETCH (
                                                                                            NATIVE-LINKER-INFO
                                                                                              ENTRY-POINT)
                                                                                          OF FN-HEADER-INFO)
                                                                                    4)))
                            (FOR OFFSET FROM 0 TO (SUB1 (LENGTH INITIAL-BYTES)) AS VALUE IN INITIAL-BYTES
                               DO (\\PUTBASEBYTE LOAD-PTR OFFSET VALUE)))
                   T)
            (FETCH (NATIVE-LINKER-INFO ENTRY-POINT) OF FN-HEADER-INFO)))))
```

(SET-CODE-BASE

```
(LAMBDA (FN)                                           ; Edited 19-Apr-88 15:51 by Krivacic
  ;; Set the Code Base
  (|if| (\\CODEBLOCKP FN)
      |then| FN
    |else| (OR (CCODEP FN)
               (ERROR FN "not compiled code"))
           (\\GET-COMPILED-CODE-BASE FN))))
```

(MAKE-NEW-CODE-BLOCK

```
(LAMBDA (FN PC-OFFSET NEW-STKMIN GCONST-OFFSET)        ; Edited  9-Jun-88 12:16 by rtk
```

;;; Mske a new code block for the function, moving the code 32 bits down to leave a slot for the address fo the Native Code. Return the pointer to the
;;; new code object.

```
  (DECLARE (SPECIAL *CODE-SIZE* *START-PC* *FN-NAME* *CODE-BASE* *GCONST-PTRS*))
  ;; Save Code Base
  (SETQ *CODE-BASE* (CODE-BLOCK-COPY *CODE-BASE* *START-PC* *CODE-SIZE* 0 (+ *START-PC* PC-OFFSET)
```

```
                              GCONST-OFFSET *GCONST-PTRS* NIL))
    (|replace| (FNHEADER STKMIN) |of| *CODE-BASE* |with| NEW-STKMIN)
    ;; Fixup the Global Values
    (SETQ *START-PC* (+ *START-PC* PC-OFFSET))
    (SETQ *CODE-SIZE* (+ *CODE-SIZE* PC-OFFSET GCONST-OFFSET))))
```

(SET-NEW-FUNCTION-DEF

```
(LAMBDA (FN FN-OBJ STREAM)                                  ; Edited  1-Jun-88 12:22 by rtk
  (PRINTOUT STREAM "Redefining " FN T)
  (COND
     ((\\CODEBLOCKP FN)
      NIL)
     ((LITATOM FN)
      (PUTD FN (|create| COMPILED-CLOSURE
                         FNHEADER _ FN-OBJ)))
     ((AND (EQ (NTYPX FN)
               \\COMPILED-CLOSURE))
      (|replace| (COMPILED-CLOSURE FNHEADER) |of| FN |with| FN-OBJ)))
  FN-OBJ))
```

(GET-FUNCTION-DEF

```
(LAMBDA (FN)                                                ; Edited 20-Apr-88 15:52 by rtk
  (COND
     ((\\CODEBLOCKP FN)
      FN)
     ((LITATOM FN)
      (|fetch| (LITATOM DEFPOINTER) |of| FN))
     ((AND (EQ (NTYPX FN)
               \\COMPILED-CLOSURE))
      (|fetch| (COMPILED-CLOSURE FNHEADER) |of| FN)))))
```

(SET-NATIVE-ADDR

```
(LAMBDA (FN-NAME FN-OBJ NATIVE-CODE-BLOCK NATIVE-ENTRY-POINT)  ; Edited 17-Jun-88 17:11 by rtk
  (LET ((OFFSET (NATIVE-ADDR-WORD-OFFSET FN-OBJ)))
       (PUTBASE FN-OBJ OFFSET (LRSH NATIVE-ENTRY-POINT 16))
       (PUTBASE FN-OBJ (ADD1 OFFSET)
                (LOGAND NATIVE-ENTRY-POINT 65535)))
  (|replace| (FNHEADER NATIVE) |of| FN-OBJ |with| T)))
```

(GET-NATIVE-ADDR

```
(LAMBDA (FN-OBJ)                                            ; Edited 19-May-88 16:09 by rtk
  (|if| (FN-OBJ)
      |then| (LET* ((OFFSET (NATIVE-ADDR-WORD-OFFSET FN-OBJ))
                    (RESULT (LOGOR (LLSH (GETBASE FN-OBJ OFFSET)
                                         16)
                                   (GETBASE FN-OBJ (ADD1 OFFSET)))))
                   RESULT)
    |else| (ERROR "Illegal FN-OBJ in GET-NATIVE-ADDR"))))
```

(LISP-ADDR-TO-NATIVE-ADDR

```
(LAMBDA (ADDR)                                              ; Edited 19-May-88 12:41 by rtk
  (|if| (EQ (MACHINETYPE)
            'MAIKO)
      |then| (SUBRCALL GET-NATIVE-ADDR-FROM-LISP-PTR ADDR)
    |else| (LOGOR (LLSH (\\HILOC ADDR)
                        16)
                  (\\LOLOC ADDR)))))
```

(NATIVE-ADDR-WORD-OFFSET

```
(LAMBDA (FN-OBJ)                                            ; Edited 20-May-88 14:34 by rtk
  (CL:FLOOR (- (|fetch| (FNHEADER STARTPC) |of| FN-OBJ)
               4)
         2)))
```

(WALK-CODE

```
(LAMBDA (CODE-BASE)                                         ; Edited 20-May-88 16:18 by rtk
```

;;; This Pass identifies jump targets, sets jump addresses, identifies following opcodes, and other information used in the 2nd pass.

```
    (DECLARE (SPECIAL *CODE-SIZE*))
    (LET (TAG OP# (GCONST-PTRS NIL)
              (START-PC (|fetch| (FNHEADER STARTPC) |of| CODE-BASE))
              (FN-NAME (|fetch| (FNHEADER FRAMENAME) |of| CODE-BASE)))
         (PRINTOUT T "Code Walk: " FN-NAME T)
         (PROG ((CODELOC START-PC)
                B B1 B2 B3 LEN PC LEVADJ STACK-EFFECT STK NEW-REC LAST-REC)
           LP   (SETQ PC CODELOC)
                (SETQ LEN (LOCAL (|fetch| OPNARGS |of| (SETQ TAG (\\FINDOP (SETQ B (GETBYTE CODE-BASE)))))))
                (COND
```

```
                ((IGREATERP LEN 0)
                 (SETQ B1 (GETBYTE CODE-BASE))))
               (COND
                  ((IGREATERP LEN 1)
                   (SETQ B2 (GETBYTE CODE-BASE))))
               (COND
                  ((IGREATERP LEN 2)
                   (SETQ B3 (GETBYTE CODE-BASE))))
               (SELECTQ (OR (AND (NEQ T (|fetch| OPPRINT |of| TAG))
                                 (|fetch| OPPRINT |of| TAG))
                            (|fetch| OPCODENAME |of| TAG))
                    (-X- (SETQ *CODE-SIZE* (IPLUS CODELOC 5))
                         (PRINTOUT T "Size of " CODE-BASE " is " *CODE-SIZE*)
                         (RETURN GCONST-PTRS))
                    (GCONST (LET* ((|const-ptr| (VAG2 B1 (LOGOR (LLSH B2 8)
                                                                B3))))
                                  (|push| GCONST-PTRS (LIST |const-ptr| CODELOC NIL))))
                    NIL)
               (GO LP)))))
```

(CODE-BLOCK-COPY

```
(LAMBDA (SRC-FN-OBJECT SRC-START-PC SRC-CODE-SIZE SRC-ALT-CODE-OFFSET DEST-START-PC DEST-ALT-CODE-OFFSET
         GCONST-PTRS NEW-FN-NAME)                          ; Edited 9-Jun-88 12:24 by rtk
  (LET* ((CELLS-TO-PC (CL:CEILING DEST-START-PC BYTESPERCELL))
         (INIT-ON-PAGE (CEIL (CL:1+ CELLS-TO-PC)
                             CELLSPERQUAD))
         (ALLOCATE-CODE-SIZE (CEIL (+ SRC-CODE-SIZE (- (+ DEST-START-PC DEST-ALT-CODE-OFFSET)
                                                       (+ SRC-START-PC SRC-ALT-CODE-OFFSET))
                                      BYTESPERQUAD)
                                   BYTESPERQUAD))
         (NEW-CODE-BASE (\\ALLOC.CODE.BLOCK ALLOCATE-CODE-SIZE INIT-ON-PAGE))
         (FN-HEADER-WORD-SIZE (CL:CEILING SRC-START-PC BYTESPERWORD))
         (CODE-COPY-SIZE (CL:CEILING (- SRC-CODE-SIZE (+ SRC-START-PC SRC-ALT-CODE-OFFSET))
                                     BYTESPERWORD))
         (CODE-BLT-DEST-BASE (\\ADDBASE NEW-CODE-BASE (CL:CEILING (+ DEST-START-PC DEST-ALT-CODE-OFFSET)
                                                                  BYTESPERWORD)))
         (CODE-BLT-SOURCE-BASE (\\ADDBASE SRC-FN-OBJECT (CL:CEILING (+ SRC-START-PC SRC-ALT-CODE-OFFSET)
                                                                    BYTESPERWORD)))
         (NOP-OPCODE (CAR (\\FINDOP 'NOP))))
        ;; Copy Fn Header
        (UNINTERRUPTABLY
            (\\BLT NEW-CODE-BASE SRC-FN-OBJECT FN-HEADER-WORD-SIZE)
            ;; Fixup References in Frame Name
            (\\ADDREF (OR (AND NEW-FN-NAME (PROGN (|replace| (FNHEADER \#FRAMENAME) |of| NEW-CODE-BASE
                                                          |with| NEW-FN-NAME)
                                                  NEW-FN-NAME))
                          (|fetch| (FNHEADER \#FRAMENAME) |of| SRC-FN-OBJECT)))
            ;; Fixup references to debugging info
            (LET* ((NTSIZE (|fetch| (FNHEADER NTSIZE) |of| SRC-FN-OBJECT))
                   (TEMP (+ (UNFOLD (|fetch| (FNHEADER OVERHEADWORDS) |of| T)
                                    BYTESPERWORD)
                            (COND
                               ((EQ NTSIZE 0)                ; No nametable, but there's a quad of zeros there anyway
                                BYTESPERQUAD)
                               (T (UNFOLD NTSIZE (ITIMES 2 BYTESPERWORD))))))
                   (NEW-NTSIZE (IDIFFERENCE SRC-START-PC TEMP))
                   (DEBUG-INFO-PTR (AND (EQ NEW-NTSIZE BYTESPERCELL)
                                        (GETBASEPTR SRC-FN-OBJECT (FOLDLO TEMP BYTESPERWORD)))))
                  (|if| DEBUG-INFO-PTR
                      |then| (PRINTOUT T "Debugging Info: " DEBUG-INFO-PTR T)
                             (\\ADDREF DEBUG-INFO-PTR))))
        ;; Insert the NOPs where the GCONST to reference the native code bock will go
        (|for| I |from| 0 |to| (SUB1 DEST-ALT-CODE-OFFSET) |do| (\\PUTBASEBYTE NEW-CODE-BASE (+ DEST-START-PC I)
                                                                               NOP-OPCODE))
        ;; Copy Code & add GCONST refs
        (UNINTERRUPTABLY
            (\\BLT CODE-BLT-DEST-BASE CODE-BLT-SOURCE-BASE CODE-COPY-SIZE)
            (|for| |ptr| |in| GCONST-PTRS |do| (\\ADDREF (CAR |ptr|))
                                               ;; Check for a re-mapped code object
                                               (|if| (CADDR |ptr|)
                                                   |then| (LET* ((|new-ptr| (CAR |ptr|))
                                                                 (|hi-val| (\\HILOC |new-ptr|))
                                                                 (|mid-val| (LRSH (\\LOLOC |new-ptr|)
                                                                                  8))
                                                                 (|low-val| (LOGAND 255 (\\LOLOC |new-ptr|)))
                                                                 (|pc-offset| (CADR |ptr|)))
                                                                (\\PUTBASEBYTE NEW-CODE-BASE (+ 1 |pc-offset|)
                                                                               |hi-val|)
                                                                (\\PUTBASEBYTE NEW-CODE-BASE (+ 2 |pc-offset|)
                                                                               |mid-val|)
```

```
                                                    (\\PUTBASEBYTE NEW-CODE-BASE (+ 3 |pc-offset|)
                                                          |low-val|)))))

        ;; Fix the Start PC

        (|replace| (FNHEADER STARTPC) |of| NEW-CODE-BASE |with| DEST-START-PC)

        ;; return the result code block

        NEW-CODE-BASE)))
```

(ADD-GCONST

```
  (LAMBDA (FN-OBJ OFFSET PTR)                                   ; Edited  1-Jun-88 12:15 by rtk
    (LET* ((BYTE-OFFSET (+ (|fetch| (FNHEADER STARTPC) |of| FN-OBJ)
                           OFFSET))
           (HI-BLOCK (\\HILOC PTR))
           (LO-BLOCK (\\LOLOC PTR))
           (LO-BLOCK1 (LRSH LO-BLOCK 8))
           (LO-BLOCK2 (LOGAND LO-BLOCK 255))
           (POP-OP (CAR (\\FINDOP 'POP))))
          (UNINTERRUPTABLY
              (\\PUTBASEBYTE FN-OBJ BYTE-OFFSET (CAR (\\FINDOP 'GCONST)))
              (\\PUTBASEBYTE FN-OBJ (+ BYTE-OFFSET 1)
                     HI-BLOCK)
              (\\PUTBASEBYTE FN-OBJ (+ BYTE-OFFSET 2)
                     LO-BLOCK1)
              (\\PUTBASEBYTE FN-OBJ (+ BYTE-OFFSET 3)
                     LO-BLOCK2)
              (\\PUTBASEBYTE FN-OBJ (+ BYTE-OFFSET 4)
                     POP-OP)

              ;; Keep Pointer Around

              (\\ADDREF PTR)
              T))))
```

(MAKE-PC-OFFSET

```
  (LAMBDA (CODEBASE)                                            ; Edited 31-May-88 18:21 by rtk
    (OR (AND (|fetch| (FNHEADER NATIVE) |of| CODEBASE)
             0)
        (LOGAND (+ (LOGAND (|fetch| (FNHEADER STARTPC) |of| CODEBASE)
                           3)
                   7)
                2147483644))))

)

;; UNIX Exec Functions

(DEFINEQ

(DO-EXEC-COMMAND
  (LAMBDA (EXEC-CMD)                                            ; Edited 10-Jun-88 14:11 by rtk
    (PRINTOUT *NATIVE-STREAM* EXEC-CMD T)
    (LET ((RETURN-CODE (IF *UNIX-STREAMS*
                           THEN (LET ((UNIX-STREAM (CREATE-PROCESS-STREAM EXEC-CMD)))
                                     (SETFILEINFO UNIX-STREAM 'ENDOFSTREAMOP 'TRAN-END-OF-UNIX-STREAM)
                                     (CL:CATCH 'NATIVE-STREAM-EOF
                                         (CL:DO ((CH (READC UNIX-STREAM)
                                                     (READC UNIX-STREAM)))
                                                (NIL)
                                                (IF (EQ CH (CHARACTER 10))
                                                    THEN (TERPRI *NATIVE-STREAM*)
                                                  ELSE (PRIN1 CH *NATIVE-STREAM*))))
                                     (UNIX-STREAM-CLOSE UNIX-STREAM))
                         ELSE (SUBRCALL OLD-COMPILE-LOAD-NATIVE EXEC-CMD))))
          (IF (NEQ 0 RETURN-CODE)
              THEN (PRINTOUT *NATIVE-STREAM* "Error in Compile: " RETURN-CODE T)
                   NIL
            ELSE T))))

(TRAN-END-OF-UNIX-STREAM
  (LAMBDA (STREAM)                                              ; Edited 17-May-88 11:53 by rtk
    (CL:THROW 'NATIVE-STREAM-EOF NIL)))

)

;; Macros

(DEFMACRO SWAPPED-FN-OBJ (|base|)
   `(LOGOR (LLSH (\\LOLOC ,|base|)
                 16)
           (\\HILOC ,|base|)))
```

```
(DEFMACRO FN-OBJ (|base|)
   `(LET* ((|base-addr| (LOGOR (LLSH (\\HILOC ,|base|)
                                     16)
                               (\\LOLOC ,|base|)))
           (|machine-addr| (|if| (OR (EQ (MACHINETYPE)
                                         'KATANA)
                                     (EQ (MACHINETYPE)
                                         'MAIKO))
                               THEN ((OPCODES 125 118 1)
                                     |base-addr|)
                               |else| |base-addr|)))
          |machine-addr|))

(DEFMACRO CODEBASELT (BASE BYTE-OFFSET)
   `(\\GETBASEBYTE ,BASE ,BYTE-OFFSET))

(DEFMACRO CODEBASELT2 (BASE BYTE-OFFSET)
   `(LOGOR (LLSH (CODEBASELT ,BASE ,BYTE-OFFSET)
                 BITSPERBYTE)
           (CODEBASELT ,BASE (ADD1 ,BYTE-OFFSET))))
```

;; Variables

```
(RPAQ? *NATIVE-TEMP-FILE-DIRECTORY* "/tmp")

(RPAQ? *NATIVE-INCLUDE-FILE-DIRECTORY* NIL)

(RPAQ? *NATIVE-LISP-RUN-FILENAME* NIL)

(RPAQ? *NATIVE-BIN-DIRECTORY* "/bin")

(RPAQ? *REMOVE-TEMP-NATIVE-FILES* T)

(RPAQ? *UNIX-STREAMS* NIL)

(RPAQ? *NATIVE-GCONST-OFFSET* 12)

(RPAQ? *KEEP-NATIVE-SOURCES* NIL)
```

;; Makefile Environment

```
(PUTPROPS NATIVE-TRANSLATOR FILETYPE TCOMPL)

(PUTPROPS NATIVE-TRANSLATOR MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE "NATIVE-TRANSLATOR"))

(PUTPROPS NATIVE-TRANSLATOR COPYRIGHT ("Xerox Corporation" 1988))
```

FUNCTION INDEX

ADD-ASM-LINE 17
ADD-CASE 14
ADD-ERROR-ENTRY 17
ADD-ERROR-LINE 17
ADD-ERROR-SELECT 18
ADD-FN-HEADER-INFO 15
ADD-GCONST 35
ADD-INFO 21
ADD-INLINE-LINE 17
ADD-LF 17
ADD-LINE 16
ADD-OPERAND-LINE 17
ADD-PUSH-OPERAND-LINE 14
BCE-LINE 17
BCE-PARSER 10
BCE-PC 18
IL:BYTE-TO-NATIVE-TRANSLATE 2
CL-ADD-FN-HEADER-INFO 16
CODE-BLOCK-COPY 34
CODEWALK1 5
CODEWALK2 8
COND-PARSER 11
CONDITIONAL-PARSER 9
CONST-PARSER 11
CONST-POINTERP 20
COPY-PARSER 11
DO-EXEC-COMMAND 35
ENVCALL-FN-OBJECT 18
ENVCALL-PARSER 12
FETCH-GCONST 4
FIND-FN0-OBJECTS 20
FIX-FILENAME 18
FN-CALL-PARSER 11
FN-CALL-PARSERX 12
GET-FUNCTION-DEF 33
GET-INFO 21
GET-NATIVE-ADDR 33
GET-NATIVE-LOAD-SIZE 32
GET-SHIFTED-VAL 21
GET-VAL 21
IL-ADD-FN-HEADER-INFO 16
INLINE-EXPAND 9
JUMP-PARSER 11
LINK-C-CODE 4
LINK-FN-CODE-BLOCK 5
LINK-OBJECT-CODE 2
LISP-ADDR-TO-NATIVE-ADDR 33
LOAD-NATIVE-FILE 32
LOADNATIVE 31
MAKE-INLINE-LISTS 25
MAKE-NEW-CODE-BLOCK 32
MAKE-OPCODE-LIST 25
MAKE-ORDERING-LIST 28
MAKE-PC-OFFSET 35
MAKE-PROGRAM-FILE 21
MAKE-TRANSLATION-ENTRIES 24
MAKE-TRANSLATION-ENTRY 23
MAKE-TRANSLATION-PATTERN-LIST ...24
MAKE-VAR-OFFSETS 20
NATIVE-ADDR-WORD-OFFSET 33
IL:NATIVE-TO-BYTE-UNTRANSLATE 2
NATIVE-TRANS 3
NATIVE-TRANSLATE 3
NBT 2
NEXT-OPERAND 20
NUT 4
OPERAND-GET 20
OPERAND-POP 21
OPERAND-PUSH 20
PARM-SUBSTITUTE 13
PC-XFORM 18
PRINT-CODE-LINE 22
PRINT-LINE-INFO 23
PUSH-ALL-OPERANDS 20
PVAR_PARSER 13
RETURN-PARSER 12
SET-CODE-BASE 32
SET-INFO 21
SET-NATIVE-ADDR 33
SET-NEW-FUNCTION-DEF 33
SETJUMPTARGET 7
SETUP-TRANSLATION-FNS 23
STR-PARSER 10
STRIP-ENDING-SLASH 23
SWAP-PARSER 12
TOS-CHECK 14
TRAN-END-OF-UNIX-STREAM 35
TRANSLATION-INIT 23
UNPACK-NUMBER 5
VERIFY-OPCODE 31
VERIFY-OPCODES 30
WALK-CODE 33
WRITE-INCLUDE-FILE 22
WRITE-INLINE-FILE 22
WRITE-PROGRAM-FILE 21

VARIABLE INDEX

KEEP-NATIVE-SOURCES 36
NATIVE-BIN-DIRECTORY 36
NATIVE-GCONST-OFFSET 36
NATIVE-INCLUDE-FILE-DIRECTORY .36
NATIVE-LISP-RUN-FILENAME 36
NATIVE-TEMP-FILE-DIRECTORY 36
REMOVE-TEMP-NATIVE-FILES 36
UNIX-STREAMS 36

RECORD INDEX

BYTE-INFO-REC 2
INFO-REC 2
LINE-INFO-REC 2
LINE-RECORD-INFO 2
NATIVE-LINKER-INFO 2
TRANSLATION-REC 2

MACRO INDEX

CODEBASELT .....36 CODEBASELT2 .....36 FN-OBJ .....36 GETBYTE .....8 SWAPPED-FN-OBJ ...35

PROPERTY INDEX

NATIVE-TRANSLATOR 36