File created: 13-Jun-2021 09:51:42 **{DSK}<Users>kaplan>Local>medley3.5>git-medley>sources>BYTECOMPILER.;2**

changes to: (FNS COMP.MLLIST)

previous date: 26-Apr-91 17:25:53 {DSK}<Users>kaplan>Local>medley3.5>git-medley>sources>BYTECOMPILER.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;

(RPAQQ **BYTECOMPILERCOMS**
[

;;; THE BYTE LISP COMPILER

```
        (COMS (INITVARS (*BYTECOMPILER-IS-EXPANDING* NIL))
              (FNS BYTEBLOCKCOMPILE2 BYTECOMPILE2 COMP.ATTEMPT.COMPILE COMP.RETFROM.POINT COMP.TRANSFORM
                   COMPERROR COMPPRINT COMPERRM)
              (FNS COMP.TOPLEVEL.COMPILE COMP.BINDLIST COMP.CHECK.VAR COMP.BIND.VARS COMP.UNBIND.VARS)
              (FNS COMP.VALN COMP.PROGN COMP.PROGLST COMP.EXP1 COMP.EXPR COMP.TRYUSERFN COMP.USERFN COMP.CONST
                   COMP.CALL COMP.VAR COMP.VAL1 COMP.PROG1 COMP.EFFECT COMP.VAL COMP.MACRO)
              (FNS COMP.VARTYPE COMP.LOOKUPVAR COMP.LOOKUPCONST)
              (FNS COMP.ST COMP.STFN COMP.STCONST COMP.STVAR COMP.STPOP COMP.DELFN COMP.STRETURN COMP.STTAG
                   COMP.STJUMP COMP.STSETQ COMP.STCOPY COMP.DELPUSH COMP.DELPOP COMP.STBIND COMP.STUNBIND)
              (VARS *NO-SIDE-EFFECT-FNS*)
              (GLOBALVARS *NO-SIDE-EFFECT-FNS*)
              (FNS COMP.ARGTYPE COMP.CLEANEXPP COMP.CLEANFNP COMP.CLEANFNOP COMP.GLOBALVARP COMP.LINKCALLP
                   COMP.ANONP COMP.NOSIDEEFFECTP)
              (FNS COMP.CPI COMP.CPI1 COMP.PICOUNT)
              (PROP BYTEMACRO EVQ)
              (FNS COMP.EVQ)
              (PROP BYTEMACRO AND OR)
              (FNS COMP.BOOL)
              (FNS COMP.APPLYFNP)
              (PROP BYTEMACRO AC)
              (FNS COMP.AC COMP.PUNT)
              (PROP BYTEMACRO FUNCTION)
              (FNS COMP.FUNCTION COMP.LAM1 COMP.GENFN)
              (INITVARS (COMP.GENFN.NUM 0))
              (GLOBALVARS COMP.GENFN.NUM COMP.UNBOXED.TAG)
              (PROP BYTEMACRO COND SELECTQ)
              (FNS COMP.COND COMP.IF COMP.SELECTQ)
              (PROP BYTEMACRO PROGN PROG1)
              (PROP BYTEMACRO QUOTE *)
              (FNS COMP.QUOTE COMP.COMMENT)
              (PROP BYTEMACRO DECLARE)
              (FNS COMP.DECLARE COMP.DECLARE1)
              (PROP (BYTEMACRO CROPS)
                    * MCROPS)
              (FNS COMP.CARCDR COMP.STCROP)
              (PROP BYTEMACRO NOT NULL)
              (FNS COMP.NOT)
              (PROP BYTEMACRO SETQ SETN)
              (FNS COMP.SETQ COMP.SETN)
              (FNS COMP.LAMBDA)
              (PROP DMACRO CL:TAGBODY)
              (PROP BYTEMACRO PROG GO RETURN CL:RETURN-FROM)
              (FNS COMP.PROG COMP.GO COMP.RETURN COMP.BLOCK COMP.RETURN-FROM COMP.TAGBODY)
              (PROP BYTEMACRO CL:LABELS)
              (FNS COMP.LABELS)
              (VARS COMP.UNBOXED.TAG NUMBERFNS (GLOBALVARFLG T)
                    (NEWOPTFLG)
                    (COMPVERSION (DATE)))
              (OPTIMIZERS IMINUS)
              (MACROS IPLUS ITIMES LOGOR LOGXOR LOGAND IDIFFERENCE IQUOTIENT IREMAINDER LSH LLSH RSH LRSH FIX
                      PLUS DIFFERENCE TIMES QUOTIENT FPLUS FDIFFERENCE FTIMES FQUOTIENT FABS FGREATERP FLESSP
                      FREMAINDER)
              (FNS COMP.NUMERIC COMP.NUMBERCALL COMP.FIX COMP.STFIX COMP.DELFIX)
              (PROP BYTEMACRO EQ EQUAL EQP)
              (FNS COMP.EQ)
              (PROP BYTEMACRO .TEST.)
              (FNS COMP.NUMBERTEST)
              (PROP BYTEMACRO * MAPFNS)
              (PROP BYTEMACRO .DOCOLLECT. .DOJOIN.)
              (FNS COMP.MAP)
              (PROP BYTEMACRO LISPXWATCH)
              (OPTIMIZERS BLKAPPLY BLKAPPLY*)
              (OPTIMIZERS ADD1VAR KWOTE FRPLNODE RPLNODE LISTGET1 FRPLNODE2)
              (PROP BYTEMACRO SUB1VAR)
              (OPTIMIZERS EQMEMB MKLIST)
```

```
        (COMS ;; Pass 1 listing
              (FNS COMP.MLLIST COMP.MLL COMP.MLLVAR COMP.MLLFN)
              (VARS COPS)
              (IFPROP MLSYM * (PROGN COPS)))
        (COMS ;; ARJ --- JUMP LENGTH RESOLVER
              (FNS OPT.RESOLVEJUMPS OPT.JLENPASS OPT.JFIXPASS OPT.JSIZE))
        (COMS ;; Utilities used by all files
              (FNS OPT.CALLP OPT.JUMPCHECK OPT.DREV OPT.CHLEV OPT.CHECKTAG OPT.NOTJUMP OPT.INITHASH
                   OPT.COMPINIT))
        (P (MOVD? 'NILL 'REFRAME)
           (AND (GETD 'OPT.COMPINIT)
                (OPT.COMPINIT)))
        (PROP BYTEMACRO LOADTIMECONSTANT)
        (PROP BYTEMACRO FRPTQ)
        (FNS OPT.CFRPTQ)
        (DECLARE%: EVAL@COMPILE DONTCOPY
               (SPECVARS AC ALAMS1 ALLVARS ARGS ARGVARS BLKDEFS BLKFLG CODE COMFN COMFNS COMTYPE CONSTS
                      EMFLAG EXP FRAME FREELST FREEVARS LAPFLG LBCNT LEVEL LOCALVARS LOCALVARS LSTFIL
                      MACEXP NLAMS1 PIFN COMPILE.CONTEXT PROGCONTEXT RETURNLABEL SPECVARS SPECVARS
                      SUBFNFREEVARS TAGS TOPFN TOPFRAME TOPLAB VARS INTERNALBLKFNS)
               (SPECVARS PLVLFILEFLG))
        (PROP BYTEMACRO IMAX2 IMIN2)
        (PROP BOX FLOAT)
        (FNS COMP.AREF COMP.ASET COMP.BOX COMP.LOOKFORDECLARE COMP.DECLARETYPE COMP.FLOATBOX
             COMP.FLOATUNBOX COMP.PREDP COMP.UBFLOAT2 COMP.UNBOX))
  (ADDVARS (COMPILETYPELST))
  (COMS                                                    ; POST OPTIMIZATION
        (FNS OPT.POSTOPT OPT.SETUPOPT OPT.SCANOPT OPT.XVARSCAN OPT.XVARSCAN1 OPT.JUMPOPT OPT.JUMPTHRU
             OPT.LBMERGE OPT.PRDEL OPT.UBDEL OPT.LBDEL OPT.LABELNTHPR OPT.JUMPREV OPT.COMMONBACK
             OPT.DELTAGREF OPT.FINDEND OPT.RETOPT OPT.RETFIND OPT.RETPOP OPT.RETOPT1 OPT.RETTEST
             OPT.RETMERGE OPT.CODELEV OPT.CODEFRAME OPT.DEFREFS OPT.SETDEFREFS)
        (FNS OPT.FRAMEOPT OPT.FRAMEMERGE OPT.NONILVAR OPT.MERGEFRAMEP OPT.FRAMELOCAL OPT.CLEANFRAME
             OPT.FRAMEDEL OPT.FRAMEVAR OPT.DELETEFRAMECHECK OPT.ONLYMEMB)
        (VARS MERGEFRAMETYPES (OPTIMIZATIONSOFF))
        (FNS OPT.SKIPPUSH OPT.DELCODE OPT.PRATTACH OPT.JUMPCOPYTEST OPT.EQOP OPT.EQVALUE OPT.DELCOPYFN)
        (FNS OPT.DEADSETQP OPT.DS1)
        (INITVARS (*BC-MACRO-ENVIRONMENT* (COMPILER::MAKE-ENV))
               (*BYTECOMPILER-OPTIMIZE-MACROLET* T))
        (FUNCTIONS CL:MACROLET)
        (DECLARE%: EVAL@COMPILE DONTCOPY (SPECVARS *BYTECOMPILER-IS-EXPANDING* *BC-MACRO-ENVIRONMENT*)
               (SPECVARS CODE LEVEL)
               (SPECVARS CL:LABELS PASS ANY CODE FRAME FRAMES)
               (GLOBALVARS MERGEFRAMEMAX MERGEFRAMEFLG MERGEFRAMETYPES *BYTECOMPILER-OPTIMIZE-MACROLET*)
               (SPECVARS VARS ANY FRAME)
               (SPECVARS ICNT TAG)
               (SPECVARS FRAME LEVEL ANY)
               (SPECVARS FRAME LEVEL ANY)
               (SPECVARS TAGS ANY)))
  (COMS                                                    ; CONSISTENCY CHECKS
        (DECLARE%: EVAL@COMPILE DONTCOPY (MACROS OPT.CCHECK)
               (VARS (COMPILECOMPILERCHECKS NIL)))
        (FNS OPT.COMPILERERROR OPT.OPTCHECK OPT.CCHECK))
  (GLOBALVARS ALAMS BYTE.EXT BYTEASSEMFN BYTECOMPFLG COMPILERMACROPROPS CIA CLEANFNLIST COMP.SCRATCH
         COMPILETYPELST COMPILEUSERFN COMPSTATLST COMPSTATS CONDITIONALS CONST.FNS CONSTOPS DONOTHING
         FILERDTBL FNA FORSHALLOW FRA HEADERBYTES HOKEYDEFPROP LAMBDANOBIND LAMS LBA LEVELARRAY LINKEDFNS
         LOADTIMECONSTANT MAXBNILS MAXBVALS MCONSTOPS MERGEFRAMEFLG MERGEFRAMEMAX MERGEFRAMETYPES MOPARRAY
         MOPCODES NODARR NOSTATSFLG NUMBERFNS OPCOPY OPNIL OPPOP OPRETURN PRA SELECTQFMEMB SELECTVARTYPES
         STATAR STATMAX STATN SYSSPECVARS UNIQUE#ARRAY VCA VCONDITIONALS VREFFRA COUTFILE XVARFLG
         MERGEFRAMEFLG OPTIMIZATIONSOFF NOFREEVARSFNS EQCONSTFN NEWOPTFLG)
  [P (CL:PROCLAIM '(CL:SPECIAL COMPVARMACROHASH]
  (DECLARE%: DONTCOPY                                      ; for compiling compiler
         EVAL@COMPILE
         (RECORDS CODELST)
         (PROP MACRO OASSOC)
         (RECORDS OP JUMP TAG VAR)
         (RECORDS FRAME COMINFO COMP JD BLOCKSTATUS))
  (MACROS THETYPE)
  (PROP FILETYPE BYTECOMPILER)
  (PROP MAKEFILE-ENVIRONMENT BYTECOMPILER)
  (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                       (NLAML OPT.INITHASH)
                                                                       (LAMA])
```

;;; THE BYTE LISP COMPILER

```
(RPAQ? *BYTECOMPILER-IS-EXPANDING* NIL)

(DEFINEQ
```

(BYTEBLOCKCOMPILE2

```
  [LAMBDA (BLKNAME BLKDEFS ENTRIES)                        (* Pavel "15-Nov-86 16:11")
    (COND
       [(EQ BYTECOMPFLG 'NOBLOCK)                          ; use PDP-10 compiler for blocks
```

```
        (RESETVARS (BYTECOMPFLG)
                   (RETURN (BLOCKCOMPILE2 BLKNAME BLKDEFS ENTRIES]
       (T (PROG [(BLKFLG T)
                 (INTERNALBLKFNS (AND (NEQ BYTECOMPFLG 'RETRY)
                                      (for X in BLKDEFS when (NOT (OR (FMEMB (CAR X)
                                                                             ENTRIES)
                                                                      (EQ (CAR X)
                                                                          BLKNAME)
                                                                      (FMEMB (CAR X)
                                                                             RETFNS)
                                                                      (AND (LISTP NOLINKFNS)
                                                                           (FMEMB (CAR X)
                                                                                  NOLINKFNS))
                                                                      (FMEMB (CAR X)
                                                                             BLKAPPLYFNS)))
                                          collect (CONS (CAR X)
                                                        (PACK* '\ BLKNAME '/ (CAR X]
                                                                                  ; this is a dummy block compiler
                 (SETQ COMP.GENFN.NUM 0)
                 (RETURN (MAPCONC BLKDEFS (FUNCTION (LAMBDA (X)
                                                      (PROG1 (COMP.ATTEMPT.COMPILE
                                                               (OR (CDR (FASSOC (CAR X)
                                                                                INTERNALBLKFNS))
                                                                   (CAR X))
                                                               (CADDR X)
                                                               (CAR X))

                                      ;; The FRPLACA allows the function definitions to be reclaimed.  This is written to parallel
                                      ;; BLOCKCOMPILE2 which needs the list of BLKDEFS for something.  --- rrb

                                                    [FRPLACA (CDDR X)
                                                           (LIST (CAR (CADDR X))
                                                                 (CADR (CADDR X])])
```

(BYTECOMPILE2

```
  [LAMBDA (FN DEF)                                          (* JonL "17-Dec-83 03:41")
    (PROG ((BLKFLG NIL))
          (SETQ COMP.GENFN.NUM 0)
          (COMP.ATTEMPT.COMPILE FN DEF)
          (RETURN FN])
```

(COMP.ATTEMPT.COMPILE

```
  [LAMBDA (TOPFN DEF RECNAME)                               (* Pavel "15-Nov-86 16:09")
    (PROG ((EMFLAG TOPFN)
           COMFNS FLG SUBFNFREEVARS)                        ; compile attempt
          (SETQ FLG (COMP.RETFROM.POINT TOPFN DEF RECNAME))
          [COND
             ((NULL EMFLAG)
              (LISPXPRIN1 '"-----
                          " T)
              (COND
                 ((NEQ COUTFILE T)
                  (LISPXPRIN1 '"-----
                              " COUTFILE]
          (COND
             (FLG                                           ; compile succeed
                  (RETURN COMFNS))
             ((AND (GETD 'COMPILE2)
                   (NEQ BYTECOMPFLG T))                     ; retry with COMPILE2
              (LISPXPRINT (CONS TOPFN '(-- retrying with COMPILE2))
                     T T)
              [COND
                 (BLKFLG (OR (EQ SPECVARS T)
                             (EVAL (CONS 'SPECVARS LOCALFREEVARS]
              (RETURN (COMPILE2 TOPFN DEF)))
             (T (LISPXPRINT [LIST (CONS TOPFN '(not compiled]
                      T T)
                (RETURN])
```

(COMP.RETFROM.POINT

```
  [LAMBDA (COMFN DEF RECNAME)                               (* Pavel "15-Nov-86 16:06")
    (PROG ((LBCNT 0))
```

;; This is the RETFROM point in case of an error while compiling COMFN or any of its generated subfunctions.

```
          (FETCH (COMP CLEAR) OF T)
```

;; CLEAR is an accessfn which clears all of the hash tables used by any HASHLINK field in the compiler; done this way so that the program need
;; not know which hash tables are used

```
          (RETURN (PROG1 (COMP.TOPLEVEL.COMPILE COMFN DEF RECNAME)
                         (FETCH (COMP CLEAR) OF T))])
```

(COMP.TRANSFORM

```
  [LAMBDA (FORM)                                            ; Edited 22-Jun-88 18:18 by TAL
```

;;; FORM is a form whose CAR is guaranteed to have a macro definition or optimizer. Transform it as much as possible and then compile it appropriately.

```
    ;; I'd like to be able to provide an environment, but I don't know how.
    (PROG ([CONTEXT (COND
                        ((EQ COMPILE.CONTEXT 'EFFECT)
                         (COMPILER:MAKE-CONTEXT :VALUES-USED 0))
                        ((COMP.PREDP COMPILE.CONTEXT)
                         (SELECTQ (fetch (JUMP OPNAME) of COMPILE.CONTEXT)
                              ((TJUMP FJUMP)
                                   (COMPILER:MAKE-CONTEXT :VALUES-USED 1 :PREDICATE-P T))
                              ((NTJUMP NFJUMP)                    ; We need the value, so make it argument context instead of
                                                                  ; predicate.
                                   (COMPILER:MAKE-CONTEXT :VALUES-USED 1 :PREDICATE-P NIL))
                              (OPT.COMPILERERROR)))
                        (T (COMPILER:MAKE-CONTEXT]
           VAL
           (*BC-MACRO-ENVIRONMENT* *BC-MACRO-ENVIRONMENT*)
           (*BYTECOMPILER-IS-EXPANDING* T))                       ; First, try to use an optimizer.
          (DECLARE (SPECVARS *BYTECOMPILER-IS-EXPANDING* *BC-MACRO-ENVIRONMENT*))
          (CL:MULTIPLE-VALUE-BIND (KIND EXPANDER)
              (COMPILER:ENV-FBOUNDP *BC-MACRO-ENVIRONMENT* (CAR FORM)
                    :LEXICAL-ONLY T)
            [for OPT-FN in (AND (NOT KIND)
                                (COMPILER:OPTIMIZER-LIST (CAR FORM)))
                do (LET ((RESULT (CL:FUNCALL OPT-FN FORM *BC-MACRO-ENVIRONMENT* CONTEXT)))
                        (if (AND (NEQ RESULT 'IGNOREMACRO)
                                 (NEQ RESULT 'COMPILER:PASS)
                                 (NEQ RESULT FORM))
                            then                                  ; An optimization has taken place.  Start over.
                                 (SETQ VAL (COMP.EXP1 RESULT))
                                 (GO OUT]
            [if (EQ KIND :MACRO)
                then                                              ; We've got a locally-defined macro...
                     (RETURN (COMP.EXP1 (CL:FUNCALL EXPANDER FORM *BC-MACRO-ENVIRONMENT*])
                                                                  ; now try interlisp macro
          [LET ((MACROPROP (GETMACROPROP (CAR FORM)
                                         COMPILERMACROPROPS)))
               (AND MACROPROP (RETURN (COMP.MACRO FORM MACROPROP]
                                                                  ; Next, look for a DEFMACRO-produced expansion function.
          [LET [(EXPN-FN (GET (CAR FORM)
                              'MACRO-FN]
               (COND
                  (EXPN-FN (RETURN (COMP.EXP1 (CL:FUNCALL EXPN-FN FORM *BC-MACRO-ENVIRONMENT*]
          [RETURN (COMP.CALL (CAR FORM)
                             (CDR FORM)
                             (COMP.ARGTYPE (CAR FORM]
      OUT (RETURN VAL])
```

(COMPERROR

```
  [LAMBDA (X)                                                     (* Pavel "15-Nov-86 16:10")
```

;;; Terminal-error handler: Aborts the compilation of this function after issuing the proper message.

```
    (AND X (COMPERRM X))
    (RETFROM 'COMP.RETFROM.POINT NIL])
```

(COMPPRINT

```
  [LAMBDA (X)                                                     ; Edited 20-Jan-88 10:54 by jds
                                                                  (* A separate function so it can be broken or advised)
    (PRIN1 X COUTFILE T])
```

(COMPERRM

```
  [LAMBDA (X FL)                                                  (* jds " 1-Feb-84 15:34")
                                                                  (* Emit an error message for the compiler)
    (AND (NULL FL)
         (SETQ FL COUTFILE))                                      (* If he specified no file, use the compiler-message file.)
    (COND
       (EMFLAG (LISPXTAB 0 0 FL)
              (LISPXPRIN1 '"-----In " FL)
              (LISPXPRIN2 EMFLAG FL T)
              (LISPXPRINT '%: FL)))
    [COND
       (X (LISPXPRIN1 '***** FL T)
          (PROG ((PLVLFILEFLG T))
                (RESETFORM (PRINTLEVEL 2 20)
                       (LISPXPRINT X FL T]
    (COND
       ((NEQ FL T)                                                (* so message gets printed in both places)
                                                                  (* i.e., force the message to go to the terminal as well.)
        (COMPERRM X T)))
    (SETQ EMFLAG NIL])

)
```

```
(DEFINEQ

(COMP.TOPLEVEL.COMPILE
  [LAMBDA (COMFN DEF RECNAME OUTER-ALLVARS)                    ; Edited 17-Jul-90 10:28 by jds
    ;; This function controls the compilation of a single function.
    (PROG (ALAMS1 NLAMS1 CONSTS ALLVARS ALLDECLS ARGVARS ARGS COMTYPE CODE FREEVARS CI (LEVEL 0)
                FRAME PIFN TOPLAB (LOCALVARS LOCALVARS)
                (SPECVARS SPECVARS)
                (*BC-MACRO-ENVIRONMENT* (COMPILER::COPY-ENV *BC-MACRO-ENVIRONMENT*))
                (COMPILER::*ENVIRONMENT* (COMPILER::MAKE-ENV :PARENT T))
                TOPFRAME MACEXP AC FRELST (COMPILE.DUNBIND.POP.MERGE.FLG T)
                TEMP)
      RETRY
          [OR [AND (LISTP DEF)
                   (LISTP (CDR DEF))
                   (SETQ COMTYPE (COND
                                      [(OR (LISTP (SETQ ARGS (CADR DEF)))
                                           (NULL ARGS))
                                       (SELECTQ (CAR DEF)
                                            (NLAMBDA 1)
                                            ([LAMBDA OPENLAMBDA]
                                                0)
                                            (CL:LAMBDA (SETQ DEF (\TRANSLATE-CL%:LAMBDA DEF))
                                                       (GO RETRY))
                                            (COND
                                               ((AND COMPILEUSERFN (SETQ DEF (APPLY* COMPILEUSERFN NIL DEF)))
                                                (GO RETRY]
                                      (T (COND
                                            ((AND LAMBDANOBIND (EQ ARGS 'NOBIND))
                                             (SETQ ARGS NIL)
                                             2)
                                            (T (SETQ ARGS (LIST ARGS))
                                               (SELECTQ (CAR DEF)
                                                    (LAMBDA 2)
                                                    (NLAMBDA 3)
                                                    (COND
                                                       ((AND COMPILEUSERFN (SETQ DEF (APPLY* COMPILEUSERFN NIL DEF
                                                                                                            )))
                                                        (GO RETRY]
              (COMPERROR (CONS COMFN '(not compilable]
          (SETQ PIFN (COND
                        ((EQ PIFN T)                           ; compile as call to self
                         0)
                        ((GETPROP COMFN OPCODEPROP)
                         0)
                        ((EQ 0 COMTYPE)
                         (OR RECNAME COMFN))
                        (T 0)))
          (SETQ FRAME (SETQ TOPFRAME (create FRAME
                                             VARS _ (SETQ ARGVARS (SETQ ALLVARS (COMP.BINDLIST ARGS)))
                                             NNILS _ 0)))
          (COMP.STTAG (SETQ TOPLAB (create TAG)))
          (COMP.VALN (CDDR DEF)
                 'RETURN)
          (COMP.UNBIND.VARS TOPFRAME T)
          (SETQ CI (create COMINFO
                           COMTYPE _ COMTYPE
                           CODE _ (OPT.POSTOPT CODE)
                           TOPFRAME _ TOPFRAME
                           ARGS _ ARGVARS))
          (SETQ FREELST (FOR X IN FREEVARS WHEN (EQ (FETCH OPNAME OF X)
                                                    'FVAR)
                           COLLECT (FETCH OPARG OF X)))
          [SETQ ALAMS1 (SUBSET ALAMS1 (FUNCTION (LAMBDA (X)
                                                (NOT (GETPROP X OPCODEPROP]
    ;; Print out the status message for this function, noting the free variable references and calls to unknown functions.  We don't report free variables
    ;; that are either proclaimed special or bound in a super function of this one.
          [LET* ((OUTER-VARS (FOR X IN OUTER-ALLVARS COLLECT (FETCH OPARG OF X)))
                 (USES-LIST (FOR X IN FREELST UNLESS (OR (VARIABLE-GLOBALLY-SPECIAL-P X)
                                                         (FMEMB X OUTER-VARS))
                               COLLECT X)))
                (COMPPRINT (CL:FORMAT NIL "(~S ~A~@[ (uses~{ ~S~})~]~@[ (calls~{ ~S~})~]~@[ (nlams~{
                                  ~S~})~])~%%" COMFN (CADR DEF)
                                  USES-LIST ALAMS1 NLAMS1))     (* (COMPPRINT (BQUOTE ((\, COMFN)
                                                                (\, (CADR DEF)) (\,@ (AND USES-LIST
                                                                (BQUOTE ((:USES (\,@ USES-LIST))))))
                                                                (\,@ (AND ALAMS1 (BQUOTE ((:CALLS
                                                                (\,@ ALAMS1)))))) (\,@ (AND NLAMS1
                                                                (BQUOTE ((:NLAMS (\,@ NLAMS1)))))))))
           ]
          (SELECTQ LAPFLG
              ((1 T)
                   (RESETFORM (OUTPUT LSTFIL)
                          (COMP.MLLIST COMFN CI)))
```

```
            NIL)
         (APPLY* BYTEASSEMFN COMFN CI)
         [COND
            ((NEQ COMFN TOPFN)                                    ; generated subfunction
             (SETQ SUBFNFREEVARS (APPEND SUBFNFREEVARS FREELST]
         (SETQ COMFNS (CONS COMFN COMFNS))
         (RETURN COMFN])
```

(COMP.BINDLIST

```
  [LAMBDA (VARS)                                                  (* lmm " 1-Jul-84 17:00")
    (for VAR in VARS collect (create VAR
                                     VARNAME _ (COMP.CHECK.VAR VAR T)
                                     COMP.VARTYPE _ (COMP.VARTYPE VAR])
```

(COMP.CHECK.VAR

```
  [LAMBDA (X BIND)                                                (* lmm " 6-Apr-84 17:49")
    [COND
       (BIND [COND
                ((NEQ X (COMP.USERFN X))
                 (COMPERRM (APPEND '(Attempt to bind CONSTANT)
                                   X]
             (COND
                ((COMP.GLOBALVARP X)
                 (COMPERRM (CONS X '(- is global]
    (OR (AND (LITATOM X)
             (NEQ X T)
             X)
        (COMPERROR (CONS X '(is not a legal variable name])
```

(COMP.BIND.VARS

```
  [LAMBDA (ARGS VALS TYPE DECLARATIONS)                           (* Pavel "15-Nov-86 16:39")
    (PROG (VLV VLN NVALS NNILS DECL X VAR DECLS VAL)
          (for VARNAME in ARGS do (SETQ VAR (create VAR
                                                    VARNAME _ (COMP.CHECK.VAR VARNAME T)
                                                    COMP.VARTYPE _ (COMP.VARTYPE VARNAME)))
                                  (if (SETQ X (CDR (FASSOC VARNAME DECLARATIONS)))
                                      then                        ; variable declared to be of a given type
                                           (COMP.EXPR (SETQ VAL (pop VALS))
                                                      (AND VAL X))
                                           (replace (VAR COMP.VARTYPE) of VAR with 'HVAR)
                                           (push DECLS (CONS VAR X))
                                           (push VLV VAR)
                                    elseif [OR (NULL (SETQ X (pop VALS)))
                                               (PROGN (COMP.VAL X)
                                                      (COND
                                                         ((EQ (CAR CODE)
                                                              OPNIL)
                                                          (COMP.DELPUSH)
                                                         T]
                                      then (push VLN VAR)
                                     else (push VLV VAR)))
          (for X in VALS do (COMP.EFFECT X))
          (SETQ NNILS (LENGTH VLN))
          [COND
             ((IGREATERP (SETQ NVALS (LENGTH VLV))
                         MAXBVALS)
              (COMPERROR (CONS EXP '(-- too many variables with values]
          (RETURN (create FRAME
                          PARENT _ FRAME
                          NVALS _ (LENGTH VLV)
                          VARS _ (OPT.DREV VLV (OPT.DREV VLN))
                          FRAMETYPE _ TYPE
                          NNILS _ NNILS
                          DECLS _ DECLS])
```

(COMP.UNBIND.VARS

```
  [LAMBDA (F TOPFLG)                                              (* lmm "29-Jun-84 09:34")
    (COND
       ((NOT (OR TOPFLG (EQ COMPILE.CONTEXT 'RETURN)
                 (OPT.JUMPCHECK CODE)))
        (OPT.CCHECK (EQ F FRAME))
        (COMP.STUNBIND (EQ COMPILE.CONTEXT 'EFFECT))
        (replace (FRAME PRIMARYRETURN) of (CAR CODE) with T)))
    'NOVALUE])

)

(DEFINEQ
```

(COMP.VALN

```
  [LAMBDA (L COMPILE.CONTEXT)                                     (* lmm "29-Jun-84 08:25")
    (COMP.PROGN L])
```

(COMP.PROGN

```
[LAMBDA (A)                                                    (* lmm "13-Jul-84 21:18")
  (COND
     ((NULL (CDR A))
      (COMP.EXP1 (CAR A)))
     (T (PROG [(FLG (AND (NOT OPTIMIZATIONSOFF)
                         (EQ COMPILE.CONTEXT 'RETURN]
          LP  (COMP.EFFECT (CAR A))
              (AND FLG (while (EQ (CAR CODE)
                                  OPPOP)
                          do                                   (* delete POP in PROGN)
                             (COMP.DELPOP)))
              (COND
                 ((OPT.JUMPCHECK CODE))
                 ((CDR (SETQ A (CDR A)))
                  (GO LP))
                 (T (RETURN (COMP.EXP1 (CAR A])
```

(COMP.PROGLST

```
[LAMBDA (LST N CONTEXT)                                        (* lmm "18-Sep-84 16:28")
  (PROG (VAL)
        (while (IGREATERP N 0) do (SETQ VAL (COMP.EXPR (pop LST)
                                                       (AND (EQ N 1)
                                                            CONTEXT)))
                                  (add N -1))
        (while (EQ (CAR (LISTP (CAR LST)))
                   '*)
           do (pop LST))
        [if LST
            then (COMPERRM '(extraneous arguments to %, (CAR EXP)
                                %: ., LST))
                 (SELECTQ CONTEXT
                      ((NIL EFFECT)                            (* ok NIL)
                           (MAPC LST (FUNCTION COMP.EFFECT)))
                      (COMPERRM '(not compiled]
        (RETURN VAL])
```

(COMP.EXP1

```
[LAMBDA (E)                                                    (* lmm "29-Jun-84 08:25")
  (COMP.EXPR E COMPILE.CONTEXT])
```

(COMP.EXPR

```
[LAMBDA (EXP COMPILE.CONTEXT)                                  ; Edited 26-Apr-91 13:08 by jds
  (DECLARE (SPECVARS *BC-MACRO-ENVIRONMENT*))
  (PROG (M V)
        [COND
           ((NULL FRAME)
            (COND
               [(OPT.JUMPCHECK CODE)
                (RETURN (COND
                           ((COMP.PREDP COMPILE.CONTEXT)
                            'PREDVALUE)
                           (T 'NOVALUE]
               (T (OPT.COMPILERERROR]
        (AND (EQ COMPILE.CONTEXT 'EFFECT)
             (COMP.NOSIDEEFFECTP EXP)
             (RETURN 'NOVALUE))
    TOP [SETQ V (COND
                   [(NLISTP EXP)
                    (COND
                       ((LITATOM EXP)
                        (SELECTQ EXP
                             ((T NIL)
                                  (COMP.CONST EXP))
                             (COMP.VAR EXP)))
                       ([OR (NUMBERP EXP)
                            (PROGN                             ; non-quoted string
                                   (OR [NULL (SETQ M (CDR (FASSOC (TYPENAME EXP)
                                                                  COMPILETYPELST]
                                       (EQ EXP (SETQ EXP (APPLY* M EXP]
                        (COMP.CONST EXP))
                       (T (GO TOP]
                   [[NOT (LITATOM (SETQ M (CAR EXP]
                    (SELECTQ (CAR (LISTP M))
                         ([LAMBDA NLAMBDA OPENLAMBDA]
                              (COMP.LAMBDA M (CDR EXP)))
                         (CL:LAMBDA                            ; Edited by TT(13-June-90) support convertion of CL:LAMBDA
                                    (SETQ EXP (CONS (\TRANSLATE-CL%:LAMBDA M)
                                                    (CDR EXP)))
                                    (GO TOP))
                         (OPCODES (OR (fetch EXTCALL of FRAME)
                                      (COMP.CLEANFNOP M 'FREEVARS)
                                      (replace EXTCALL of FRAME with F))
```

```
                                    (COMP.STFN (CAR EXP)
                                           (for X in (CDR EXP) sum (COMP.VAL X)
                                                                   1)))
                              (COND
                                 ((SETQ M (COMP.TRYUSERFN EXP))
                                  (SETQ EXP M)
                                  (GO TOP))
                                 (T (COMPERROR (CONS M '(- non-atomic CAR of form]
                           ((OR (AND (SETQ V (GETMACROPROP M COMPILERMACROPROPS))
                                     (NEQ V T))
                                (GET M 'MACRO-FN)
                                (COMPILER:OPTIMIZER-LIST M)
                                (EQ (COMPILER:ENV-FBOUNDP *BC-MACRO-ENVIRONMENT* M :LEXICAL-ONLY T)
                                    :MACRO))
                            (COMP.TRANSFORM EXP))
                           ((AND (EQ COMPILE.CONTEXT 'RETURN)
                                 (EQ M PIFN))
                            (COMP.CPI M (CDR EXP)))
                           ((SETQ V (COMP.ARGTYPE M))
                            (COMP.CALL M (CDR EXP)
                                  V))
                           ((SETQ V (COMP.TRYUSERFN EXP))
                            (SETQ EXP V)
                            (GO TOP))
                           (T (COMP.CALL M (CDR EXP]
              (RETURN (SELECTQ COMPILE.CONTEXT
                           (NIL NIL)
                           (EFFECT (OR (EQ V 'NOVALUE)
                                       (COMP.STPOP))
                                   'NOVALUE)
                           (RETURN (OR (OPT.JUMPCHECK CODE)
                                       (COMP.STRETURN))
                                   'NOVALUE)
                           (COND
                              ((COMP.PREDP COMPILE.CONTEXT)
                               (COND
                                  ((NEQ V 'PREDVALUE)                ; in this case, COMPILE.CONTEXT is a jump instruction
                                   (COMP.STJUMP COMPILE.CONTEXT)))
                               'PREDVALUE)
                              ((EQ (CAR (LISTP COMPILE.CONTEXT))
                                   'TYPE)
                               NIL)
                              ((EQ (CAR (LISTP COMPILE.CONTEXT))
                                   'UNBOXED)
                               (OR (EQ V 'UNBOXED)
                                   (COMP.UNBOX (CDR COMPILE.CONTEXT)))
                               'UNBOXED])
```

(COMP.TRYUSERFN

```
[LAMBDA (EXP M)
  (AND COMPILEUSERFN (COND
                        ((EQ (SETQ M (COMP.USERFN EXP))
                             'INSTRUCTIONS)
                         [COMPERRM (CONS EXP '(COMPILEUSERFN returned INSTRUCTIONS]
                         NIL)
                        (T M])
```

(COMP.USERFN

```
[LAMBDA (X)                                                  ; Edited  7-Apr-87 13:12 by Pavel
  (COND
     ((CL:KEYWORDP X)
      (LIST 'QUOTE X))
     [(AND (EQ [CAR (LISTP (CAR (LISTP X]
               'CL:LAMBDA)
           (COND
              ((INTERSECTION (CADR (CAR X))
                             CL:LAMBDA-LIST-KEYWORDS)
               (ERROR "Can't cope with lambda keywords in internal LAMBDA lists"))
              (T `([LAMBDA ,@(CDAR X]
                   ,@(CDR X]
     ((LITATOM X)
      (OR (AND COMPVARMACROHASH (GETHASH X COMPVARMACROHASH))
          X))
     (T (LET [(FN TOPFN)
              (OTHERVARS (FOR X IN ALLVARS COLLECT (FETCH OPARG OF X]
             (DECLARE (SPECVARS FN OTHERVARS))               ; uses FN DEF ARGS OTHERVARS
             (APPLY* COMPILEUSERFN (CDR X)
                    X])
```

(COMP.CONST

```
[LAMBDA (X)                                                  (* lmm "13-Jul-84 21:18")
  (COND
     ((AND (NOT OPTIMIZATIONSOFF)
           (EQ COMPILE.CONTEXT 'EFFECT))                     (* CONST in (EQ COMPILE.CONTEXT
                                                             (QUOTE EFFECT)))
```

```
        'NOVALUE)
       ((AND (NOT OPTIMIZATIONSOFF)
             (COMP.PREDP COMPILE.CONTEXT))
        [AND (SELECTQ (fetch OPNAME of COMPILE.CONTEXT)
                  (TJUMP X)
                  (NTJUMP (COND
                              (X (COMP.STCONST X)
                                 T)))
                  (FJUMP (NOT X))
                  (NFJUMP (COND
                              ((NOT X)
                               (COMP.STCONST X)
                              T)))
                  (SHOULDNT))
             (COMP.STJUMP 'JUMP (CAR (fetch OPARG of COMPILE.CONTEXT))
                    (CDR (fetch OPARG of COMPILE.CONTEXT]
        'PREDVALUE)
       (T (COMP.STCONST X])
```

(COMP.CALL

```
  [LAMBDA (F A TYP)                                    ; Edited  9-Feb-87 18:29 by Pavel
    (PROG ((N 0))
          (OR (fetch EXTCALL of FRAME)
              (COMP.CLEANFNOP F 'FREEVARS)
              (replace EXTCALL of FRAME with F))
          (SELECTQ TYP
               (3                                      ; call nlambda by applying with entire arglist as first arg
                  (pushnew NLAMS1 F)
                  (COMP.STCONST A)
                  (RETURN (COMP.STFN F 1)))
               (1                                      ; call NLAMBDA spread merely by not compiling arguments
                  (pushnew NLAMS1 F))
               (NIL                                    ; unknown argtype, assume lambda, but warn user
                    (pushnew ALAMS1 F))
               NIL)
      LP  [COND
             ((LISTP A)
              (SELECTQ TYP
                   (1 (COMP.STCONST (CAR A)))
                   (COMP.VAL (CAR A)))
              (SETQ N (ADD1 N))
              (SETQ A (CDR A))
              (GO LP))
             (A (COMPERROR (CONS A '(- unusual tail for argument list]
          (RETURN (COMP.STFN F N])
```

(COMP.VAR

```
  [LAMBDA (VAR)                                        (* lmm "24-Jan-85 18:40")
    (COND
       ((EQ COMPILE.CONTEXT 'EFFECT)                   (* VAR in EFFECT)
        'NOVALUE)
       (T (SETQ VAR (COMP.LOOKUPVAR VAR T))
          (COMP.STVAR VAR)
          (LET [(DECL (CDR (ASSOC VAR ALLDECLS]
               (if (EQ (CAR (LISTP DECL))
                       'UNBOXED)
                   then (COMP.BOX (CDR DECL])
```

(COMP.VAL1

```
  [LAMBDA (L COMPILE.CONTEXT)                          (* lmm "29-Jun-84 08:25")
    (COMP.PROG1 L])
```

(COMP.PROG1

```
  [LAMBDA (A)                                          (* lmm "29-Jun-84 08:25")
    (COND
       ((NULL (CDR A))
        (COMP.EXP1 (CAR A)))
       (T (PROG1 (COMP.EXPR (CAR A)
                        (COND
                           ((EQ COMPILE.CONTEXT 'EFFECT)
                            COMPILE.CONTEXT)))
                 (MAPC (CDR A)
                       (FUNCTION COMP.EFFECT)))])
```

(COMP.EFFECT

```
  [LAMBDA (E)                                          (* lmm "13-Jul-84 21:18")
    (PROG ((LV LEVEL))
          (COND
             ((OPT.JUMPCHECK CODE)                     (* code for effect eliminated after JUMP or RETURN)
              (RETURN))
             (T (OPT.CCHECK LV)))
          (RETURN (PROG1 (COMP.EXPR E 'EFFECT)
```

```
                    (OPT.CCHECK (OR AC (EQ LEVEL LV)
                                       (OPT.JUMPCHECK CODE)))])
```

(COMP.VAL

```
  [LAMBDA (X)                                                    (* lmm "13-Jul-84 21:18")
    (PROG ((LV LEVEL))
          (COND
             ((OPT.JUMPCHECK CODE)                               (* code for value eliminated after JUMP or RETURN)
              (RETURN)))
          (RETURN (PROG1 (COMP.EXPR X)
                         (OPT.CCHECK (OR (EQ (ADD1 LV)
                                             LEVEL)
                                         AC
                                         (OPT.JUMPCHECK CODE)))))])
```

(COMP.MACRO

```
  [LAMBDA (EXP MAC)                                              ; Edited 11-May-87 16:25 by amd
    (COND
       [(NLISTP MAC)
        (SELECTQ MAC
             (T                                                  ; The macro is 'T'.  Compile this as a function-call.
                (COMP.CALL (CAR EXP)
                       (CDR EXP)
                       (COMP.ARGTYPE (CAR EXP))))
             (COMP.PUNT (COMP.PUNT))
             (BLKAPPLY* MAC (CDR EXP]
       (T (SELECTQ (CAR MAC)
             (APPLY (APPLY (CADR MAC)
                           (CDR EXP)))
             (APPLY* (APPLY (CADR MAC)
                            (CONS (CDR EXP)
                                  (CDDR MAC))))
             (OPENLAMBDA (COMP.LAMBDA MAC (CDR EXP)))
             (LET* ((*BYTECOMPILER-IS-EXPANDING* T)
                    (EXPANSION (MACROEXPANSION EXP MAC T COMPILE.CONTEXT)))
                   (DECLARE (SPECVARS *BYTECOMPILER-IS-EXPANDING*))
                   (if (EQ EXPANSION EXP)
                       then                                      ; can't expand, e.g.  returns IGNOREMACRO
                            (COMP.CALL (CAR EXP)
                                   (CDR EXP)
                                   (COMP.ARGTYPE (CAR EXP)))
                     else (COMP.EXP1 EXPANSION])

)

(DEFINEQ
```

(COMP.VARTYPE

```
  [LAMBDA (VAR)                                                  (* lmm "13-MAR-81 09:36")
    (OPT.CCHECK (AND VAR (LITATOM VAR)))
    (COND
       ((COMP.ANONP VAR)
        'HVAR)
       (T 'AVAR])
```

(COMP.LOOKUPVAR

```
  [LAMBDA (V FORVALUE)                                           (* jds " 1-Feb-84 15:08")
    (PROG (X)
          (COND
             ((SETQ X (find VAR in ALLVARS suchthat (EQ (fetch VARNAME of VAR)
                                                        V)))
              (RETURN X)))
          (COND
             ((SETQ X (find VAR in FREEVARS suchthat (EQ (fetch VARNAME of VAR)
                                                         V)))
              (RETURN X)))
          [COND
             ((NEQ V (SETQ X (COMP.USERFN V)))
              (COND
                 (FORVALUE (RETAPPLY 'COMP.VAR (FUNCTION COMP.VAL)
                                     (LIST X)
                                     T))
                 (T (COMPERRM (CONS V " - is compile time constant, yet is bound or set."]
          (SETQ FREEVARS (CONS (SETQ X (create VAR
                                               COMP.VARTYPE _ (COND
                                                                 ((AND GLOBALVARFLG (COMP.GLOBALVARP V))
                                                                  'GVAR)
                                                                 (T 'FVAR))
                                               VARNAME _ (COMP.CHECK.VAR V)))
                               FREEVARS))
          (RETURN X])
```

(COMP.LOOKUPCONST

```
  [LAMBDA (X)                                                    (* lmm "24-JUN-78 22:56")
    (COND
       ((NULL X)
        OPNIL)
       (T (OR [CAR (SOME CONSTS (FUNCTION (LAMBDA (Y)
                                          (EQ X (fetch OPARG of Y]
              (PROG1 (SETQ X (create OP
                                     OPNAME _ 'CONST
                                     OPARG _ X))
                  (SETQ CONSTS (NCONC1 CONSTS X)))])

)

(DEFINEQ
```

(COMP.ST

```
  [LAMBDA (X DL)                                                 (* lmm "13-Jul-84 21:18")
    (OPT.CCHECK DL)
    (COND
       [(OR LEVEL (EQ DL T))
        (SETQ CODE (CONS X CODE))
        (SETQ LEVEL (COND
                       ((FIXP DL)
                        (IPLUS LEVEL DL]
       (T (OPT.CCHECK (OPT.JUMPCHECK CODE))                      (* didn't store code after JUMP or RETURN)
          NIL])
```

(COMP.STFN

```
  [LAMBDA (FN N)                                                 (* lmm "16-APR-82 00:14")
    (COMP.ST (create OP
                     OPNAME _ 'FN
                     OPARG _ (CONS N (OR (AND BLKFLG (LITATOM FN)
                                                (CDR (FASSOC FN INTERNALBLKFNS)))
                                         FN)))
          (IDIFFERENCE 1 N])
```

(COMP.STCONST

```
  [LAMBDA (X)                                                    (* lmm "16-APR-82 00:14")
    (COMP.ST (COMP.LOOKUPCONST X)
          1])
```

(COMP.STVAR

```
  [LAMBDA (VREF)                                                 (* lmm "16-APR-82 00:14")
    (COMP.ST VREF 1])
```

(COMP.STPOP

```
  [LAMBDA (N)                                                    (* lmm "16-APR-82 00:14")
    (RPTQ (OR N 1)
          (COMP.ST OPPOP -1])
```

(COMP.DELFN

```
  [LAMBDA NIL                                                    (* lmm%: "22-JUL-77 02:40")
    [SETQ LEVEL (IPLUS (SUB1 LEVEL)
                       (CAR (fetch OPARG of (CAR CODE]
    (SETQ CODE (CDR CODE])
```

(COMP.STRETURN

```
  [LAMBDA NIL                                                    (* lmm "16-APR-82 00:13")
    (COMP.ST OPRETURN T)
    (SETQ LEVEL (SETQ FRAME])
```

(COMP.STTAG

```
  [LAMBDA (TAG)                                                  (* lmm "13-Jul-84 21:18")
    (PROG ((NLV (fetch (TAG LEVEL) of TAG))
           (NF (fetch (TAG FRAME) of TAG)))
          (OR (COND
                 [(OR NLV NF)
                  (AND (EQ NLV (OR LEVEL (SETQ LEVEL NLV)))
                       (EQ NF (OR FRAME (SETQ FRAME NF]
                 ((OR LEVEL FRAME)
                  (AND (replace (TAG LEVEL) of TAG with LEVEL)
                       (replace (TAG FRAME) of TAG with FRAME)))
                 (T T))
              (OPT.COMPILERERROR))
          [COND
             ((AND (EQ (fetch OPNAME of (CAR CODE))
                       'JUMP)
                   (EQ (fetch (JUMP TAG) of (CAR CODE))
                       TAG))                                     (* delete JUMP to next in COMP.STTAG)
              (SETQ CODE (CDR CODE]
```

```
          (COMP.ST TAG 0])
```

(COMP.STJUMP

```
  [LAMBDA (OP TAG JT)                                    (* lmm "13-Jul-84 21:18")
    (COND
       ((OPT.JUMPCHECK CODE)                             (* JUMP not stored after JUMP or RETURN)
        NIL)
       (T [COND
             ((NULL TAG)

          (* even if OP is given and in correct format, re-cons it up since OPT.POSTOPT might smash it)

              (SETQ TAG (CAR (fetch OPARG of OP)))
              (SETQ JT (CDR (fetch OPARG of OP)))
              (SETQ OP (fetch OPNAME of OP]
          (COMP.ST (create JUMP
                           OPNAME _ OP
                           TAG _ TAG
                           JT _ JT)
           0)
          (PROG ((F (fetch FRAME of TAG))
                 (V (fetch (TAG LEVEL) of TAG))
                 NV)
                (COND
                   (F (OPT.CCHECK (EQ F FRAME)))
                   (T (replace (TAG FRAME) of TAG with FRAME)))
                (SETQ NV (SELECTQ OP
                             (JUMP (PROG1 LEVEL
                                          (SETQ FRAME (SETQ LEVEL))))
                             ((FJUMP TJUMP)
                                 (SETQ LEVEL (SUB1 LEVEL)))
                             ((NFJUMP NTJUMP)
                                 (PROG1 LEVEL
                                        (SETQ LEVEL (SUB1 LEVEL))))
                             (ERRORSET (PROG1 (SUB1 LEVEL)
                                              (SETQ FRAME JT)
                                              (SETQ LEVEL 0)))
                             (OPT.COMPILERERROR)))
                (OPT.CCHECK (OR (NULL NV)
                                (IGEQ NV 0)))
                (OPT.CCHECK (OR (NULL LEVEL)
                                (IGEQ LEVEL 0)))
                (COND
                   (V (OPT.CCHECK (EQ V NV)))
                   (T (replace (TAG LEVEL) of TAG with NV])
```

(COMP.STSETQ

```
  [LAMBDA (VREF)                                         (* lmm "16-APR-82 00:14")
    (OPT.CCHECK (IGREATERP LEVEL 0))
    (COMP.ST (create OP
                     OPNAME _ 'SETQ
                     OPARG _ VREF)
     0])
```

(COMP.STCOPY

```
  [LAMBDA NIL                                            (* lmm "16-APR-82 00:14")
    (OPT.CCHECK (IGREATERP LEVEL 0))
    (COMP.ST OPCOPY 1])
```

(COMP.DELPUSH

```
  [LAMBDA NIL                                            (* lmm%: " 9-AUG-76 21:50:49")
    (SUB1VAR LEVEL)
    (SETQ CODE (CDR CODE])
```

(COMP.DELPOP

```
  [LAMBDA NIL                                            (* lmm "28-OCT-77 15:23")
    (SETQ LEVEL (ADD1 LEVEL))
    (SETQ CODE (CDR CODE])
```

(COMP.STBIND

```
  [LAMBDA (F)                                            (* lmm " 1-Jul-84 14:48")
    [COND
       ((NULL (fetch PARENT of F))
        (replace PARENT of F with FRAME))
       (T (OPT.CCHECK (EQ (fetch PARENT of F)
                          FRAME]
    [COND
       [(NULL (fetch (FRAME LEVEL) of F))
        (replace (FRAME LEVEL) of F with (IDIFFERENCE LEVEL (fetch NVALS of F]
       (T (OPT.CCHECK (EQ (fetch (FRAME LEVEL) of F)
                          (IDIFFERENCE LEVEL (fetch NVALS of F]
```

```
    (COND
       ([EVERY CODE (FUNCTION (LAMBDA (X)
                                (SELECTQ (fetch OPNAME of X)
                                     ((TAG HVAR AVAR GVAR CONST)
                                         T)
                                     (FN (OR (NULL (fetch (FRAME VARS) of F))
                                             (COMP.CLEANFNOP (CDR (fetch OPARG of X))
                                                    'FREEVARS)))
                                     NIL]                          (* PROG is first thing in function)
        (replace CPIOK of F with T)))
    (COMP.ST (create OP
                     OPNAME _ 'BIND
                     OPARG _ (CONS NIL F))
          0)
    (SETQ FRAME F)
    (SETQ LEVEL 0])
```

(COMP.STUNBIND

```
  [LAMBDA (D)                                                      (* lmm "16-APR-82 00:14")
    (COMP.ST (create OP
                     OPNAME _ (COND
                                 (D 'DUNBIND)
                                 (T 'UNBIND))
                     OPARG _ (CONS LEVEL FRAME))
          0)
    [SETQ LEVEL (IPLUS (fetch (FRAME LEVEL) of FRAME)
                       (COND
                          (D 0)
                          (T 1]
    (SETQ FRAME (fetch PARENT of FRAME])

)

(RPAQQ *NO-SIDE-EFFECT-FNS*
       (CL::%%* CL::%%+ CL::%%- CL::%%/ CL::%%< CL::%%= CL::%%> CL::%%LLSH1 CL::%%LLSH8 CL::%%LOGIOR CL::%%LRSH1
              CL::%%LRSH8 CL:* + - / CL:/= /= CL:1+ CL:1- < <= = > >= ABS CL:ACOS CL:ACOSH ADD1
              CL:ADJUSTABLE-ARRAY-P CL:ALPHA-CHAR-P CL:ALPHANUMERICP AND ANTILOG APPEND ARCCOS ARCSIN ARCTAN
              ARCTAN2 CL:AREF CL:ARRAY-ELEMENT-TYPE CL:ARRAY-HAS-FILL-POINTER-P CL:ARRAY-RANK ARRAYORIG
              CL:ARRAYP ARRAYP ARRAYSIZE ARRAYTYP CL:ASH CL:ASIN CL:ASINH ASSOC CL:ATAN CL:ATANH CL:ATOM ATOM
              CL:BIT-VECTOR-P BITCLEAR BITSET BITTEST CL:BOOLE CL:BOTH-CASE-P BYTE CL:BYTE-POSITION BYTE-SIZE
              BYTEPOSITION BYTESIZE CAAAAR CAAADR CAAAR CAADAR CAADDR CAADR CAAR CADAAR CADADR CADAR CADDAR
              CADDDR CADDR CADR CAR CDAAAR CDAADR CDAAR CDADAR CDADDR CDADR CDAR CDDAAR CDDADR CDDAR CDDDAR
              CDDDDR CDDDR CDDR CDR CL:CEILING CL:CHAR-BIT CL:CHAR-BITS CL:CHAR-CODE CL:CHAR-DOWNCASE
              CL:CHAR-EQUAL CL:CHAR-FONT CL:CHAR-GREATERP CL:CHAR-INT CL:CHAR-LESSP CL:CHAR-NAME
              CL:CHAR-NOT-EQUAL CL:CHAR-NOT-GREATERP CL:CHAR-NOT-LESSP CL:CHAR-UPCASE CL:CHAR/= CL:CHAR<
              CL:CHAR<= CL:CHAR= CL:CHAR> CL:CHAR>= CL:CHARACTER CHARACTER CL:CHARACTERP CL:CIS CL:CODE-CHAR
              CL:COMMONP CL:COMPILED-FUNCTION-P COMPLEX CL:COMPLEXP CL:CONJUGATE CONS CL:CONSP COPY COPYALL
              CL:COS COS CL:COSH DATEFORMAT CL:DECODE-FLOAT CL:DECODE-UNIVERSAL-TIME CL:DENOMINATOR
              CL:DEPOSIT-FIELD DEPOSITBYTE DIFFERENCE CL:DIGIT-CHAR CL:DIGIT-CHAR-P DPB CL:EIGHTH ELT ELTD
              CL:ENCODE-UNIVERSAL-TIME CL:ENDP EQ EQL EQP CL:EQUAL EQUAL CL:EQUALP EVENP CL:EXP CL:EXPT EXPT
              FASSOC CL:FCEILING FCHARACTER FDIFFERENCE FEQP CL:FFLOOR FGREATERP CL:FIFTH CL:FIRST FIX FIXP FIXR
              FLESSP FLOAT CL:FLOAT-DIGITS CL:FLOAT-PRECISION CL:FLOAT-RADIX CL:FLOAT-SIGN CL:FLOATP FLOATP
              CL:FLOOR FMAX FMEMB FMIN FMINUS CL:FOURTH FPLUS FPLUS2 FQUOTIENT FREMAINDER CL:FROUND FTIMES
              CL:FTRUNCATE CL:FUNCTIONP CL:GCD GCD GEQ GETHASH GETP GETPROP CL:GRAPHIC-CHAR-P GREATERP HARRAYP
              HARRAYSIZE CL:HASH-TABLE-P CL:IDENTITY IDIFFERENCE IEQP IGEQ IGREATERP ILEQ ILESSP CL:IMAGPART
              IMAX IMIN IMINUS IMOD CL:INPUT-STREAM-P CL:INT-CHAR CL:INTEGER-DECODE-FLOAT CL:INTEGER-LENGTH
              INTEGERLENGTH CL:INTEGERP INTERSECTION IPLUS IQUOTIENT IREMAINDER CL:ISQRT ITIMES CL:KEYWORDP
              KWOTE LAST CL:LCM LDB CL:LDB-TEST LEQ LESSP LIST CL:LIST-LENGTH CL:LISTP LISTP LITATOM LLSH
              LOADBYTE LOG CL:LOG LOGAND CL:LOGANDC1 CL:LOGANDC2 CL:LOGBITP CL:LOGCOUNT CL:LOGEQV CL:LOGIOR
              CL:LOGNAND CL:LOGNOR LOGNOT LOGOR CL:LOGORC1 CL:LOGORC2 CL:LOGTEST LOGXOR CL:LOWER-CASE-P LRSH LSH
              CL:MAKE-CHAR CL:MASK-FIELD MASK.0'S MASK.1'S MAX MEMB MEMBER MIN MINUS MINUSP CL:MOD CL:NAME-CHAR
              NEQ NILL CL:NINTH NLISTP NOT CL:NTH CL:NTHCDR NTYPX NULL CL:NUMBERP NUMBERP CL:NUMERATOR ODDP OR
              CL:OUTPUT-STREAM-P CL:PACKAGEP CL:PATHNAMEP CL:PHASE PLUS CL:PLUSP POWEROFTWOP PROG1 PROGN
              QUOTIENT CL:RANDOM-STATE-P CL:RATIONAL CL:RATIONALIZE CL:RATIONALP READTABLEP CL:REALPART RELSTKP
              CL:REM REMAINDER CL:REST ROT ROUND RSH SASSOC CL:SCALE-FLOAT XCL::SCEILING CL:SECOND
              CL:SET-CHAR-BIT CL:SEVENTH XCL::SFLOOR CL:SIGNUM CL:SIMPLE-BIT-VECTOR-P CL:SIMPLE-STRING-P
              CL:SIMPLE-VECTOR-P CL:SIN SIN CL:SINH CL:SIXTH SMALLP CL:SQRT SQRT XCL::SROUND STACKP
              CL:STANDARD-CHAR-P STKNARGS CL:STREAM-ELEMENT-TYPE STREAMP CL:STRING-CHAR-P STRING-EQUAL
              CL:STRING-GREATERP CL:STRING-LESSP CL:STRING-NOT-EQUAL CL:STRING-NOT-GREATERP CL:STRING-NOT-LESSP
              STRING.EQUAL CL:STRING/= CL:STRING< CL:STRING<= CL:STRING= CL:STRING> CL:STRING>= CL:STRINGP
              STRINGP XCL::STRUNCATE SUB1 CL:SUBTYPEP CL:SXHASH CL:SYMBOLP TAILP TAN CL:TAN CL:TANH CL:TENTH
              CL:THIRD TIMEREXPIRED? TIMES TRUE CL:TRUNCATE CL:TYPE-OF TYPEP UNION CL:UPPER-CASE-P CL:VECTORP
              ZERO CL:ZEROP ZEROP \ADDBASE \ARG0 \CALLME \GETBASE \GETBASEBYTE \GETBASEFIXP \GETBASEPTR
              \GETBASESTRING \VAG2 create fetch))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS *NO-SIDE-EFFECT-FNS*)
)

(DEFINEQ
```

(COMP.ARGTYPE

```
  [LAMBDA (FN)                                                     (* lmm "25-FEB-82 16:29")
    (PROG NIL
          (RETURN (COND
```

```
                    ((NOT (LITATOM FN))
                     (ARGTYPE FN))
                    ((FMEMB FN LAMA)
                     2)
                    ((FMEMB FN LAMS)
                     0)
                    ((FMEMB FN NLAML)
                     1)
                    ((FMEMB FN NLAMA)
                     3)
                    (T (ARGTYPE (OR [AND BLKFLG (OR (CADDR (FASSOC FN BLKDEFS))
                                                    (AND (FMEMB FN BLKLIBRARY)
                                                         (GETP FN 'BLKLIBRARYDEF]
                                    (GETPROP FN 'BROKEN)
                                    (AND (GETD FN)
                                         FN)
                                    (GETPROP FN 'EXPR)
                                    (RETURN (COND
                                               ((FMEMB FN NOFIXFNSLST)
                                                2)
                                               (T NIL])
```

(COMP.CLEANEXPP

```
[LAMBDA (X TYPE)                                          (* lmm "15-APR-82 23:01")
  (COND
     ((NLISTP X))
     ((COMP.CLEANFNP (CAR X)
             TYPE)
      (EVERY (CDR X)
             (FUNCTION (LAMBDA (X)
                       (COMP.CLEANEXPP X TYPE])
```

(COMP.CLEANFNP

```
[LAMBDA (X TYPE)                                          (* lmm "15-APR-82 23:02")
  (COND
     ((LITATOM X)
      (APPLY* CLEANFNTEST X))
     ((LISTP X)
      (SELECTQ (CAR X)
           ([LAMBDA OPENLAMBDA]
               [EVERY (CDDR X)
                      (FUNCTION (LAMBDA (X)
                                (COMP.CLEANEXPP X TYPE])
           NIL])
```

(COMP.CLEANFNOP

```
[LAMBDA (FN TYPE)                                         (* lmm "15-APR-82 23:07")
  (APPLY* CLEANFNTEST FN TYPE])
```

(COMP.GLOBALVARP

```
[LAMBDA (X)                                               (* lmm%: " 9-AUG-76 20:34:14")
  (OR (GETP X 'GLOBALVAR)
      (FMEMB X GLOBALVARS])
```

(COMP.LINKCALLP

```
[LAMBDA (FN)                                              (* edited (18-NOV-75 . 2341))
  (COND
     ((AND (LISTP NOLINKFNS)
           (FMEMB FN NOLINKFNS))
      NIL)
     ((AND BLKFLG (OR (FASSOC FN BLKDEFS)
                      (FMEMB FN BLKLIBRARY)))
      T)
     ((AND (LISTP LINKFNS)
           (FMEMB FN LINKFNS))
      T)
     ((EQ NOLINKFNS T)
      NIL)
     ((OR BLKFLG (EQ LINKFNS T))
      T])
```

(COMP.ANONP

```
[LAMBDA (E)                                               (* lmm "12-May-86 13:23")
  (COND
     ((NEQ LOCALVARS T)
      (FMEMB E LOCALVARS))
     (T (NOT (OR (EQ SPECVARS T)
                 (FMEMB E SPECVARS)
                 (VARIABLE-GLOBALLY-SPECIAL-P E)
                 (AND BLKFLG (FMEMB E LOCALFREEVARS])
```

```
(COMP.NOSIDEEFFECTP
  [LAMBDA (EXP)                                          ; Edited 17-May-90 16:47 by nm
    (COMP.CLEANEXPP EXP *NO-SIDE-EFFECT-FNS*])

)

(DEFINEQ

(COMP.CPI
  [LAMBDA (FN ARGS)                                      (* Pavel "15-Nov-86 16:22")
    (PROG ((F FRAME))
      LP  (COND
             ((EQ F TOPFRAME)
              (COMP.CPI1 ARGS ARGVARS (COMP.PICOUNT ARGS))
              (while (NEQ FRAME TOPFRAME) do                 ; unbind localvar FRAME before recursion
                                          (COMP.STUNBIND T))
              (COND
                 ((NEQ LEVEL 0)                          ; pop stack before recursion
                  (COMP.STPOP LEVEL)))
              (COMP.STJUMP 'JUMP TOPLAB)                 ; COMP.CPI succeeds
              (RETURN 'NOVALUE))
             ((SELECTQ (fetch FRAMETYPE of F)
                  ((PROG LAMBDA)
                       [COND
                          ((OASSOC 'AVAR (fetch VARS of F))
                           (COND
                              ((NOT (fetch CPIOK of F))  ; can't remove recursion inside frame with SPECVARS
                               T)
                              (T                         ; COMP.CPI can succeed because SPECVARS bound first thing
                                                         ; in function
                                 NIL])
                  (PROGN                                 ; can't remove recursion inside ERRORSET
                         T))
              (COMP.CALL FN ARGS 0))
             ((SETQ F (fetch PARENT of F))
              (GO LP))
             (T (OPT.COMPILERERROR])

(COMP.CPI1
  [LAMBDA (ARGS VARS N)                                  (* lmm "16-APR-82 00:28")
    (COND
       [(NULL VARS)
        (COND
           ((LISTP ARGS)
            (COMP.EFFECT (CAR ARGS))
            (COMP.CPI1 (CDR ARGS)
                   VARS
                   (SUB1 N]
       ([OR (IGREATERP N 0)
            (NOT (LITATOM (CAR ARGS)))
            (NEQ (CAR ARGS)
                 (fetch OPARG of (CAR VARS]
        (COMP.VAL (CAR ARGS))
        (COMP.CPI1 (CDR ARGS)
               (CDR VARS)
               (SUB1 N))
        (COMP.STSETQ (CAR VARS))
        (COMP.STPOP))
       (T (COMP.CPI1 (CDR ARGS)
                 (CDR VARS)
                 (SUB1 N])

(COMP.PICOUNT
  [LAMBDA (ARGS)                                         (* lmm "27-OCT-81 20:57")
    (PROG ((N 0)
           (ND 0)
           (VARS ARGVARS))
      LP  (COND
             (VARS (SETQ N (ADD1 N))
                   (COND
                      [(AND (LITATOM (CAR ARGS))
                            (EQ (CAR ARGS)
                                (fetch OPARG of (CAR VARS]
                      ((NOT (COMP.CLEANEXPP (CAR ARGS)
                                    'COMP.PICOUNT))
                       (SETQ ND N)))
                   (SETQ VARS (CDR VARS))
                   (SETQ ARGS (CDR ARGS))
                   (GO LP)))
          (RETURN ND])

)

(PUTPROPS EVQ BYTEMACRO COMP.EVQ)
```

```
(DEFINEQ

(COMP.EVQ
  [LAMBDA (X)                                                    (* lmm "18-Sep-84 16:06")
    (RESETVARS (COMPVARMACROHASH)
               (RETURN (COMP.PROGLST X 1])

)

(PUTPROPS AND BYTEMACRO (APPLY* COMP.BOOL T))

(PUTPROPS OR BYTEMACRO (APPLY* COMP.BOOL NIL))

(DEFINEQ

(COMP.BOOL
  [LAMBDA (A FLAG)                                               (* lmm "29-Apr-85 13:33")
    (COND
       ((NULL A)                                                 (* (AND/OR))
        (COMP.CONST FLAG))
       ((NULL (CDR A))                                           (* (AND/OR expr))
        (COMP.EXP1 (CAR A)))
       (T (PROG ((END (create TAG))
                 P)
                (SETQ P (create JUMP
                                OPNAME _ [COND
                                           ((COMP.PREDP COMPILE.CONTEXT)
                                                                 (* AND/OR in PREDF)
                                            (SELECTQ (fetch OPNAME of (SETQ P COMPILE.CONTEXT))
                                                 ((TJUMP NTJUMP)
                                                     (COND
                                                        (FLAG 'FJUMP)
                                                        (T (GO LP))))
                                                 ((FJUMP NFJUMP)
                                                     (COND
                                                        (FLAG (GO LP))
                                                        (T 'TJUMP)))
                                                 (COMP.COMPILERERROR)))
                                           [(EQ COMPILE.CONTEXT 'EFFECT)
                                                                 (* AND/OR in EFFECT)
                                            (COND
                                               (FLAG 'FJUMP)
                                               (T 'TJUMP]
                                           (T                    (* other AND/OR)
                                              (COND
                                                 (FLAG 'NFJUMP)
                                                 (T 'NTJUMP]
                                TAG _ END))
            LP  (COND
                   ((CDR A)
                    (COMP.EXPR (CAR A)
                               P)
                    (SETQ A (CDR A))
                    (GO LP)))
                (RETURN (PROG1 [COMP.EXPR (CAR A)
                                          (SELECTQ COMPILE.CONTEXT
                                               ((EFFECT RETURN NIL)
                                                  COMPILE.CONTEXT)
                                               (COND
                                                  ((COMP.PREDP COMPILE.CONTEXT)
                                                   COMPILE.CONTEXT)
                                                  (T NIL]
                               (COMP.STTAG END])

)

(DEFINEQ

(COMP.APPLYFNP
  [LAMBDA (X)                                                    (* edited%: "21-MAY-80 09:38")
    (AND (LISTP X)
         (SELECTQ (CAR X)
              ((FUNCTION QUOTE)
                   (AND (NULL (CDDR X))
                        (SELECTQ (COMP.ARGTYPE (CADR X))
                             (NIL (pushnew ALAMS1 (CADR X))
                                  T)
                             ((0 1 2)
                                  T)
                             NIL)))
              NIL])

)

(PUTPROPS AC BYTEMACRO COMP.AC)
```

```
(DEFINEQ

(COMP.AC
  [LAMBDA NIL                                                    (* lmm%: " 1-OCT-76 12:41:01")
    (OR (EQ (SETQ AC EXP)
            DONOTHING)
        (COMP.PUNT))
    NIL])

(COMP.PUNT
  [LAMBDA NIL                                                    (* lmm "22-OCT-79 12:44")
    (PROG [(EM (CONS (CAR EXP)
                     '(-- can't compile]
          (COMPERROR (COND
                       [MACEXP (CONS 'Under (CONS (CAR MACEXP)
                                                  (CONS '- EM]
                       (T EM])

)

(PUTPROPS FUNCTION BYTEMACRO COMP.FUNCTION)

(DEFINEQ

(COMP.FUNCTION
  [LAMBDA (A)                                                    (* lmm "16-APR-82 00:18")
    (PROG ((FN (CAR A)))
          [COND
             ((LISTP FN)
              (SETQ FN (COMP.LAM1 FN]
          (RETURN (COND
                     ((CDR A)
                      (COMP.CALL 'FUNCTION (CONS FN (CDR A))
                             1))
                     (T (COMP.STCONST FN])

(COMP.LAM1
  [LAMBDA (DEF)                                                  (* Pavel "15-Nov-86 16:12")
    (PROG ((FN (COMP.GENFN)))
          (COMP.TOPLEVEL.COMPILE FN DEF NIL ALLVARS)
          (for X in ALLVARS when (AND (NEQ (fetch OPNAME of X)
                                           'AVAR)
                                      (FMEMB (fetch OPARG of X)
                                             SUBFNFREEVARS))
             do                                                  ; change LOCALVAR to SPECVAR because subfn uses it free
                (replace OPNAME of X with 'AVAR))
          (RETURN FN])

(COMP.GENFN
  [LAMBDA NIL                                                    (* Pavel "28-Oct-86 20:16")
    (COND
       ((IGEQ COMP.GENFN.NUM 9999)
        (SETQ COMP.GENFN.NUM 0)))
    (CL:INTERN (CL:FORMAT NIL "~AA~4,'0D" (STRING COMFN)
                      (add COMP.GENFN.NUM 1))
           (CL:SYMBOL-PACKAGE COMFN])

)

(RPAQ? COMP.GENFN.NUM 0)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS COMP.GENFN.NUM COMP.UNBOXED.TAG)
)

(PUTPROPS COND BYTEMACRO COMP.COND)

(PUTPROPS SELECTQ BYTEMACRO COMP.SELECTQ)

(DEFINEQ

(COMP.COND
  [LAMBDA (A)                                                    (* lmm "12-Mar-85 07:04")
    (PROG (TEST CLAUSE (END (create TAG))
                ENDF NEXT [CONTEXT (SELECTQ COMPILE.CONTEXT
                                            ((EFFECT RETURN NIL)
                                                COMPILE.CONTEXT)
                                            (COND
                                               ((COMP.PREDP COMPILE.CONTEXT)
                                                NIL)
                                               ((EVERY A (FUNCTION CDR))
                                                COMPILE.CONTEXT)
                                               (T NIL]
```

```
              COMPVAL)
      LP  [SETQ TEST (CAR (SETQ CLAUSE (CAR A]
          (COND
             [(CDR CLAUSE)                                    (* is there anything after the test?)

          (* * compile the test in a context where, if false, it will jump to NEXT)

              (LET [(MORE (LET ((HERE CODE))
                               [COMP.EXPR TEST (create JUMP
                                                       OPNAME _ 'FJUMP
                                                       TAG _ (SETQ NEXT (create TAG]
                               (OR OPTIMIZATIONSOFF (for (X _ CODE) by (CDR X)
                                                       while (AND X (NEQ X HERE))
                                                       do (AND (EQ [CAR (LISTP (fetch OPARG of (CAR X]
                                                                  NEXT)
                                                               (RETURN T]
                     [COND
                        ((NOT (OPT.JUMPCHECK CODE))              (* it doesn't ALWAYS jump to next)
                         (SETQ COMPVAL (COMP.VALN (CDR CLAUSE)
                                                  CONTEXT))
                         (OR (OPT.JUMPCHECK CODE)
                             (COMP.STJUMP 'JUMP (SETQ ENDF END]
                     (COND
                        (MORE (COMP.STTAG NEXT))
                        (T (GO OUT]
              [(CDR A)                                           (* this is a form (COND (TEST) --) where there is more to come)
               (COMP.EXPR TEST (create JUMP
                                       OPNAME _ (COND
                                                   ((EQ CONTEXT 'EFFECT)
                                                    'TJUMP)
                                                   (T 'NTJUMP))
                                       TAG _ (SETQ ENDF END]
              (T                                                 (* (COND -- (A)) is equivalent to (COND --
                                                                 (T A)))
                 (SETQ COMPVAL (COMP.EXPR TEST CONTEXT))
                 (GO OUT)))
          (COND
             ((SETQ A (CDR A))
              (GO LP)))
          (AND (NEQ CONTEXT 'EFFECT)
               (COMP.EXPR NIL))
      OUT (AND ENDF (COMP.STTAG END))
          (RETURN (COND
                     ((EQ CONTEXT 'EFFECT)
                      'NOVALUE)
                     (T COMPVAL])
```

(COMP.IF

```
  [LAMBDA (A)                                                    (* lmm "24-May-86 16:32")
                                                                 (* used by common lisp IF)
    (DESTRUCTURING-BIND (TEST THEN ELSE)
           A
           (PROG (CONDTEST (END (create TAG))
                        ENDF NEXT (CONTEXT (SELECTQ COMPILE.CONTEXT
                                                    ((EFFECT RETURN NIL)
                                                        COMPILE.CONTEXT)
                                                    NIL))
                        COMPVAL)
                 [COMP.EXPR TEST (create JUMP
                                         OPNAME _ 'FJUMP
                                         TAG _ (SETQ NEXT (create TAG]
                 [COND
                    ((NOT (OPT.JUMPCHECK CODE))
                     (SETQ COMPVAL (COMP.EXPR THEN CONTEXT))
                     (OR (OPT.JUMPCHECK CODE)
                         (COMP.STJUMP 'JUMP (SETQ ENDF END]
                 (COMP.STTAG NEXT)
                 [COND
                    ((NOT (OPT.JUMPCHECK CODE))                  (* it doesn't ALWAYS jump to next)
                     (SETQ COMPVAL (COMP.EXPR ELSE CONTEXT]
                 (AND ENDF (COMP.STTAG END))
                 (RETURN (COND
                            ((EQ CONTEXT 'EFFECT)
                             'NOVALUE)
                            (T COMPVAL])
```

(COMP.SELECTQ

```
  [LAMBDA (A)                                                    (* lmm "13-Jul-84 21:18")
    (PROG ((END (create TAG))
           VAR THISLABEL NEXT TEST CLAUSE)                       (* compile SELECTQ)
          (COMP.VAL (CAR A))
          (SETQ A (CDR A))
          (COND
             ((FMEMB (fetch OPNAME of (CAR CODE))
                     SELECTVARTYPES)                             (* SELECTQVARTYPES is (AVAR HVAR) for Alto and NIL for
                                                                 maxc)
```

```
                                                                    (* SELECTQ var)
          (SETQ VAR (CAR CODE))
          (COMP.DELPUSH))
         ((AND (EQ (fetch OPNAME of (CAR CODE))
                   'SETQ)
               (FMEMB (fetch OPNAME of (fetch OPARG of (CAR CODE)))
                      SELECTVARTYPES))                              (* SELECTQ SETQ)
          (SETQ VAR (fetch OPARG of (CAR CODE)))
          (COMP.STPOP))
         [(EQ (fetch OPNAME of (CAR CODE))
              'CONST)                                               (* SELECTQ of constant)
          (RETURN (COMP.PROGN (PROG [(C (fetch OPARG of (CAR CODE]
                                        (COMP.DELPUSH)
                                    ALP (COND
                                           ((NULL (CDR A))
                                            (RETURN A)))
                                        [COND
                                           ((COND
                                               ((LISTP (CAAR A))
                                                (FMEMB C (CAAR A)))
                                               (T (EQ (CAAR A)
                                                      C)))
                                            (RETURN (CDAR A]
                                        (SETQ A (CDR A))
                                        (GO ALP]
          (T (SETQ THISLABEL T)))
   LP   [COND
          ((NULL (CDR A))
           (AND THISLABEL (NULL VAR)
                (COMP.STPOP))
           (RETURN (PROG1 (COMP.EXPR (CAR A)
                                     (COND
                                        ((COMP.PREDP COMPILE.CONTEXT)
                                         NIL)
                                        (T COMPILE.CONTEXT)))
                          (OR (EQ COMPILE.CONTEXT 'RETURN)
                              (COMP.STTAG END)))]
       (SETQ THISLABEL)
       [COND
          ([LISTP (SETQ TEST (CAR (SETQ CLAUSE (PROG1 (CAR A)
                                                      (SETQ A (CDR A)))]
           (COND
              ((NLISTP (CDR TEST))
               (SETQ TEST (CAR TEST)))
              (SELECTQFMEMB                                         (* FMEMB in SELECTQ)
                     (COND
                        (VAR (COMP.STVAR VAR))
                        ((CDR A)
                         (COMP.STCOPY)))
                     (COMP.STCONST (APPEND TEST))
                     (COMP.STFN 'FMEMB 2)
                     (GO DUN))
              (T (SETQ THISLABEL (create TAG))
                 (MAP TEST (FUNCTION (LAMBDA (Y)
                                        (COND
                                           ((CDR Y)
                                            (COND
                                               (VAR (COMP.STVAR VAR))
                                               (T (COMP.STCOPY)))
                                            (COMP.STCONST (CAR Y))
                                            (COMP.STFN 'EQ 2)
                                            (COMP.STJUMP 'TJUMP THISLABEL))
                                           (T (SETQ TEST (CAR Y]
       (COND
          (VAR (COMP.STVAR VAR))
          ((OR THISLABEL (CDR A))
           (COMP.STCOPY)))
       (COMP.STCONST TEST)
       (COMP.STFN 'EQ 2)
   DUN [COND
          ((AND (NULL THISLABEL)
                (NULL (CDR A))
                (NULL (CAR A)))                                     (* SELECTQ ends in NIL)
           (COMP.STJUMP (COND
                           ((EQ COMPILE.CONTEXT 'EFFECT)
                            'FJUMP)
                           (T 'NFJUMP))
                 END)
           (RETURN (PROG1 (COMP.VALN (CDR CLAUSE)
                                     (COND
                                        ((EQ COMPILE.CONTEXT 'EFFECT)
                                         COMPILE.CONTEXT)))
                          (COMP.STTAG END]
       (COMP.STJUMP 'FJUMP (SETQ NEXT (create TAG)))
       (COND
          (THISLABEL (COMP.STTAG THISLABEL)))
       (COND
```

```
             ((AND (OR THISLABEL (CDR A))
                   (NULL VAR))
              (COMP.STPOP)))
          (COMP.VALN (CDR CLAUSE)
                 (SELECTQ COMPILE.CONTEXT
                      ((EFFECT RETURN)
                          COMPILE.CONTEXT)
                     NIL))
          (OR (EQ COMPILE.CONTEXT 'RETURN)
              (COMP.STJUMP 'JUMP END))
          (COMP.STTAG NEXT)
          (GO LP])

)

(PUTPROPS PROGN BYTEMACRO COMP.PROGN)

(PUTPROPS PROG1 BYTEMACRO COMP.PROG1)

(PUTPROPS QUOTE BYTEMACRO COMP.QUOTE)

(PUTPROPS * BYTEMACRO COMP.COMMENT)

(DEFINEQ
```

(COMP.QUOTE

```
  [LAMBDA (A)                                                    (* lmm%: " 9-AUG-76 22:04:49")
    [COND
       ((CDR A)
        (COMPERRM (CONS EXP '(- probable parenthesis error]
    (COMP.CONST (CAR A])
```

(COMP.COMMENT

```
  [LAMBDA (A)                                                    (* lmm "29-Jun-84 08:25")
    (COND
       ((NOT (EQ COMPILE.CONTEXT 'EFFECT))
        [COMPERRM (CONS EXP '(- value of comment used?]
        (COMP.STCONST (CAR A)))
       (T 'NOVALUE])

)

(PUTPROPS DECLARE BYTEMACRO COMP.DECLARE)

(DEFINEQ
```

(COMP.DECLARE

```
  [LAMBDA (A)                                                    (* lmm "24-May-86 20:36")
                                                                 (* compile DECLARE)
    [MAPC A (FUNCTION (LAMBDA (B)
                      (SELECTQ (CAR B)
                           (LOCALVARS (COMP.DECLARE1 (CDR B)
                                            'LOCALVARS
                                            'SPECVARS SYSSPECVARS))
                           (SPECVARS (COMP.DECLARE1 (CDR B)
                                           'SPECVARS
                                           'LOCALVARS SYSLOCALVARS))
                           (CL:SPECIAL [MAPC (fetch VARS of FRAME)
                                            (FUNCTION (LAMBDA (V VTAG)
                                                        (COND
                                                           ((AND (EQ (fetch OPNAME of V)
                                                                     'HVAR)
                                                                 (FMEMB (fetch OPARG of V)
                                                                        (CDR B)))
                                                            (replace OPNAME of V with 'AVAR])
                           (IGNORE)
                           (USEDFREE NIL)
                           ((ADDTOVAR DEFLIST PUTPROPS CONSTANTS SETQQ USEDFREE GLOBALVARS)
                                (EVAL B))
                           (UNBOXED (push ALLDECLS COMP.UNBOXED.TAG))
                           (TYPE                                 (* handled elsewhere))
                           (COMPERRM (CONS B '(- used in DECLARE]
    (COMP.CONST (CAR A])
```

(COMP.DECLARE1

```
  [LAMBDA (VAL VAR OTHERVAR SYSOTHERVAR)                         (* lmm "31-MAR-78 02:47")
    (SET VAR (COND
                ((LISTP VAL)
                 (COND
                    ((LISTP (SETQ VAR (EVALV VAR)))
                     (APPEND VAL VAR))
                    ((EQ VAR T))
                    (T VAL)))
                ((EQ VAL T)
                 (SET OTHERVAR SYSOTHERVAR)
```

```
                T)
               (T VAL)))
    (MAPC (fetch VARS of FRAME)
          (FUNCTION (LAMBDA (V VTAG)
                       (COND
                          ((NEQ (SETQ VTAG (COMP.VARTYPE (fetch OPARG of V)))
                                (fetch OPNAME of V))                    (* Already made some decision based on localvars
                                                                        (COMPERRM (CONS EXP (QUOTE
                                                                        (-
                                                                        illegal DECLARE)))))
                           (replace OPNAME of V with VTAG])

)

(RPAQQ MCROPS
       (CAR CDR CAAR CDAR CADR CDDR CAAAR CDAAR CADAR CDDAR CAADR CDADR CADDR CDDDR CAAADR CDAADR CADADR CDDADR CAADDR CDADDR CADDDR CDDDDR))

(PUTPROPS CAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDR BYTEMACRO COMP.CARCDR)

(PUTPROPS CAAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CADR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDDR BYTEMACRO COMP.CARCDR)

(PUTPROPS CAAAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDAAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CADAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDDAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CAADR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDADR BYTEMACRO COMP.CARCDR)

(PUTPROPS CADDR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDDDR BYTEMACRO COMP.CARCDR)

(PUTPROPS CAAAAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDAAAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CADAAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDDAAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CAADAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDADAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CADDAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDDDAR BYTEMACRO COMP.CARCDR)

(PUTPROPS CAAADR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDAADR BYTEMACRO COMP.CARCDR)

(PUTPROPS CADADR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDDADR BYTEMACRO COMP.CARCDR)

(PUTPROPS CAADDR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDADDR BYTEMACRO COMP.CARCDR)

(PUTPROPS CADDDR BYTEMACRO COMP.CARCDR)

(PUTPROPS CDDDDR BYTEMACRO COMP.CARCDR)

(PUTPROPS CAR CROPS (A))

(PUTPROPS CDR CROPS (D))

(PUTPROPS CAAR CROPS (A A))

(PUTPROPS CDAR CROPS (A D))

(PUTPROPS CADR CROPS (D A))
```

```
(PUTPROPS CDDR CROPS (D D))

(PUTPROPS CAAAR CROPS (A A A))

(PUTPROPS CDAAR CROPS (A A D))

(PUTPROPS CADAR CROPS (A D A))

(PUTPROPS CDDAR CROPS (A D D))

(PUTPROPS CAADR CROPS (D A A))

(PUTPROPS CDADR CROPS (D A D))

(PUTPROPS CADDR CROPS (D D A))

(PUTPROPS CDDDR CROPS (D D D))

(PUTPROPS CAAAAR CROPS (A A A A))

(PUTPROPS CDAAAR CROPS (A A A D))

(PUTPROPS CADAAR CROPS (A A D A))

(PUTPROPS CDDAAR CROPS (A A D D))

(PUTPROPS CAADAR CROPS (A D A A))

(PUTPROPS CDADAR CROPS (A D A D))

(PUTPROPS CADDAR CROPS (A D D A))

(PUTPROPS CDDDAR CROPS (A D D D))

(PUTPROPS CAAADR CROPS (D A A A))

(PUTPROPS CDAADR CROPS (D A A D))

(PUTPROPS CADADR CROPS (D A D A))

(PUTPROPS CDDADR CROPS (D A D D))

(PUTPROPS CAADDR CROPS (D D A A))

(PUTPROPS CDADDR CROPS (D D A D))

(PUTPROPS CADDDR CROPS (D D D A))

(PUTPROPS CDDDDR CROPS (D D D D))

(DEFINEQ

(COMP.CARCDR
  [LAMBDA (A)                                          (* lmm "18-Sep-84 16:13")
                                                       (* Used for compiling CAR/CDR etc)
    (COND
       ((EQ COMPILE.CONTEXT 'EFFECT)                   (* CAR/CDR in EFF)
        (COMP.PROGLST A 1 COMPILE.CONTEXT))
       (T (COMP.PROGLST A 1)
          (MAPC (GETPROP (CAR EXP)
                         'CROPS)
                (FUNCTION (LAMBDA (X)
                            (COMP.STFN (SELECTQ X
                                               (A 'CAR)
                                               'CDR)
                                       1])


(COMP.STCROP
  [LAMBDA (X)                                          (* lmm "16-APR-82 00:16")
    (COMP.STFN (SELECTQ X
                       (A 'CAR)
                       'CDR)
           1])

)

(PUTPROPS NOT BYTEMACRO COMP.NOT)

(PUTPROPS NULL BYTEMACRO COMP.NOT)

(DEFINEQ

(COMP.NOT
  [LAMBDA (A TMP)                                      (* lmm "18-Sep-84 16:30")
    (COND
```

```
       ((AND (COMP.PREDP COMPILE.CONTEXT)
             (SETQ TMP (OPT.NOTJUMP COMPILE.CONTEXT)))
        (COMP.PROGLST A 1 TMP))
       (T (COMP.PROGLST A 1)
          (COMP.STFN 'NULL 1])

)

(PUTPROPS SETQ BYTEMACRO COMP.SETQ)

(PUTPROPS SETN BYTEMACRO COMP.SETN)

(DEFINEQ

(COMP.SETQ
  [LAMBDA (A)                                          (* lmm "29-Oct-84 15:23")
    (PROG (VAR DECL)
          (SETQ VAR (COMP.LOOKUPVAR (CAR A)))
          [SETQ DECL (LISTP (CDR (ASSOC VAR ALLDECLS]
          (COMP.PROGLST (CDR A)
                1 DECL)
          (COMP.STSETQ VAR)
          (IF (AND (NEQ COMPILE.CONTEXT 'EFFECT)
                   (EQ (CAR DECL)
                       'UNBOXED))
              THEN (COMP.BOX (CDR DECL])


(COMP.SETN
  [LAMBDA (A)                                          (* lmm%: "20-OCT-76 01:33:55")
    [COMPERRM (CONS (CAR A)
                    '(- warning%: SETN compiled as SETQ]
    (COMP.SETQ A])

)

(DEFINEQ

(COMP.LAMBDA
  [LAMBDA (FN VALS)                                    (* Pavel "15-Nov-86 16:23")
    (PROG ((VARS (CADR FN))
           F
           (EXPS (CDDR FN))
           V E (I 0)
           SUBOLD SUBNEW VAR)
          [if (EQ (CAR FN)
                  'OPENLAMBDA)
              then                                     ; compile OPENLAMBDA expression
                   [while VARS do (COMP.VAL (pop VALS))
                                  (COND
                                     ((EQ (fetch OPNAME of (CAR CODE))
                                          'CONST)
                                      (push SUBOLD (pop VARS))
                                      [push SUBNEW (KWOTE (fetch OPARG of (CAR CODE]
                                      (COMP.DELPUSH))
                                     (T (push V (pop VARS]
                   (for X in VALS do (COMP.EFFECT X))
                   (while (AND V (SETQ VAR (SELECTQ (fetch OPNAME of (CAR CODE))
                                                    ((AVAR HVAR FVAR GVAR)
                                                         (PROG1 (fetch OPARG of (CAR CODE))
                                                                (COMP.DELPUSH)))
                                                    (SETQ (PROG1 (fetch OPARG of (fetch OPARG of (CAR CODE)))
                                                                 (COMP.STPOP)))
                                                    NIL)))
                     do                                ; substitute for variable in OPENLAMBDA
                        (push SUBNEW VAR)
                        (push SUBOLD (pop V)))
                   [if (NULL V)
                       then                             ; OPENLAMBDA with all variables substituted for
                            (RETURN (COMP.PROGN (SUBPAIR SUBOLD SUBNEW EXPS]
                   (while V do [push SUBNEW (CAR (push VARS (COMP.GENFN]
                               (push SUBOLD (pop V))
                               (push VALS DONOTHING))
                   (SETQ EXPS (CONS '(DECLARE (LOCALVARS . T))
                                    (SUBPAIR SUBOLD SUBNEW EXPS)))
            else (SELECTQ (ARGTYPE FN)
                          (0)
                          (1                           ; open NLAMBDA nospread
                             (SETQ VALS (MAPCAR VALS (FUNCTION KWOTE))))
                          (2                           ; open LAMBDA nospread
                             (RETURN (COMP.CALL (COMP.LAM1 FN)
                                                VALS 2)))
                          (3                           ; open NLAMBDA spread
                             (SETQ VARS (LIST VARS))
                             (SETQ VALS (LIST (KWOTE VALS))))
                          (COMPERROR (CONS FN '(- illegal open function]
          (SETQ F (COMP.BIND.VARS VARS VALS 'LAMBDA (COMP.LOOKFORDECLARE EXPS)))
```

```
          (PROG ((ALLVARS (APPEND (fetch VARS of F)
                                  ALLVARS))
                 (ALLDECLS (APPEND (fetch DECLS of F)
                                   ALLDECLS))
                 (LOCALVARS LOCALVARS)
                 (SPECVARS SPECVARS))
                (COMP.STBIND F)
                (COMP.VALN EXPS (SELECTQ COMPILE.CONTEXT
                                         ((EFFECT RETURN)
                                              COMPILE.CONTEXT)
                                         NIL)))
          (RETURN (COMP.UNBIND.VARS F])

)

(PUTPROPS CL:TAGBODY DMACRO COMP.TAGBODY)

(PUTPROPS PROG BYTEMACRO COMP.PROG)

(PUTPROPS GO BYTEMACRO COMP.GO)

(PUTPROPS RETURN BYTEMACRO COMP.RETURN)

(PUTPROPS CL:RETURN-FROM BYTEMACRO COMP.RETURN-FROM)

(DEFINEQ

(COMP.PROG
  [LAMBDA (A)                                                    (* lmm "13-Jul-84 21:18")
    (PROG ([VARS (for X in (CAR A) collect (COND
                                           ((LITATOM X)
                                            X)
                                           [(NLISTP X)
                                            (COMPERROR (CONS X '(- bad PROG variable]
                                           (T (CAR X]
           [VALS (for X in (CAR A) collect (AND (LISTP X)
                                                (COND
                                                   ((CDDR X)
                                                    (CONS 'PROG1 (CDR X)))
                                                   (T (CADR X]
           F)
          [SETQ F (COMP.BIND.VARS VARS VALS 'PROG (COMP.LOOKFORDECLARE (SETQ A (CDR A]
          (PROG ((ALLVARS (APPEND (fetch VARS of F)
                                  ALLVARS))
                 (ALLDECLS (APPEND (fetch DECLS of F)
                                   ALLDECLS))
                 (LOCALVARS LOCALVARS)
                 (SPECVARS SPECVARS)
                 TAGS
                 (RETURNLABEL (create TAG
                                      LEVEL _ (COND
                                                 ((EQ COMPILE.CONTEXT 'EFFECT)
                                                  0)
                                                 (T 1))
                                      FRAME _ F))
                 PROGLEVEL
                 (PROGCONTEXT (SELECTQ COMPILE.CONTEXT
                                       ((EFFECT RETURN)
                                          COMPILE.CONTEXT)
                                       NIL))
                 FLG)
                (COMP.STBIND F)
                [for X in A do (COND
                                  ((LISTP X))
                                  [(NOT (LITATOM X))
                                   (COMPERROR (CONS X '(- illegal tag]
                                  [(FASSOC X TAGS)
                                   (COMPERROR (CONS X '(- multiply defined tag]
                                  (T (SETQ TAGS (CONS (CONS X (SETQ X (create TAG
                                                                      LBNO _ X)))
                                                      TAGS))
                                     (replace (TAG FRAME) of X with FRAME)
                                     (replace (TAG LEVEL) of X with 0]
                (replace PROGLABELS of F with TAGS)
                [SETQ FLG (AND (NOT OPTIMIZATIONSOFF)
                               (NULL TAGS)
                               (EQ PROGCONTEXT 'RETURN]          (* Check if can delete extra POP's)
                [for X in A do (COND
                                  [(LITATOM X)
                                   (COMP.STTAG (CDR (FASSOC X TAGS]
                                  (T (COMP.EFFECT X)
                                     (AND FLG (while (EQ (CAR CODE)
                                                         OPPOP)
                                                 do                (* delete POP in PROG)
                                                    (COMP.DELPOP]
                (COND
                   ((NOT (OR (EQ COMPILE.CONTEXT 'EFFECT)
```

```
                        (OPT.JUMPCHECK CODE)))          (* PROG dropped off)
                  (COMP.EXPR NIL)))
             (OR (EQ COMPILE.CONTEXT 'RETURN)
                 (COMP.STTAG RETURNLABEL)))
        (RETURN (COMP.UNBIND.VARS F])
```

(COMP.GO

```
[LAMBDA (A)                                             (* lmm " 2-Jun-86 23:03")
  (PROG (D ANYPROG)
        [COND
           ((OPT.JUMPCHECK CODE)                        (* UNREACHABLE GO -- DON'T COMPILE)
            (RETURN 'NOVALUE]
    LP  [SELECTQ (fetch FRAMETYPE of FRAME)
             ([LAMBDA PROG]
                  [COND
                     ((SETQ D (FASSOC (CAR A)
                                      (fetch PROGLABELS of FRAME)))
                      (COND
                         ((NOT (ZEROP LEVEL))           (* GO needs to POP)
                          (COMP.STPOP LEVEL)))
                      (COMP.STJUMP 'JUMP (CDR D))
                      (RETURN 'NOVALUE])
                 (COMPERROR (CONS (CAR A)
                                  '(- illegal GO]       (* non local GO)
        (COMP.STUNBIND T)
        (GO LP])
```

(COMP.RETURN

```
[LAMBDA (A)                                             (* lmm "18-Sep-84 16:31")
  (PROG ((PROGFRAME FRAME))
        [COND
           ((NEQ PROGCONTEXT 'RETURN)
            (COND
               ([NOT (OR (EQ PROGCONTEXT 'EFFECT)
                         (EQ LEVEL 0)
                         (NEQ (fetch FRAMETYPE of FRAME)
                              'PROG]                    (* RETURN POPs beforehand)
                (COMP.STPOP LEVEL]
    CHKLP
        [SELECTQ (fetch FRAMETYPE of PROGFRAME)
             (PROG)
             (LAMBDA (SETQ PROGFRAME (fetch PARENT of PROGFRAME))
                     (GO CHKLP))
             (COMPERROR (CONS COMFN '(- illegal RETURN]
        (COMP.PROGLST A 1 PROGCONTEXT)
        [COND
           ((OPT.JUMPCHECK CODE)
            (RETURN 'NOVALUE]
        (COND
           ((NEQ PROGCONTEXT 'RETURN)
            [PROG NIL
              LP  (SELECTQ (fetch FRAMETYPE of FRAME)
                           (PROG (OPT.CCHECK (EQ FRAME PROGFRAME)))
                           (LAMBDA                      (* RETURN inside LAMBDA)
                                   (COMP.STUNBIND (EQ PROGCONTEXT 'EFFECT))
                                   (GO LP))
                           (COMPERROR (CONS COMFN '(- illegal RETURN]
            [COND
               ((EQ PROGCONTEXT 'EFFECT)
                (COMP.STPOP LEVEL))
               ((NEQ LEVEL 1)
                (OPT.COMPILERERROR '(unimplemented RETURN]
            (COMP.STJUMP 'JUMP RETURNLABEL)))
        (RETURN 'NOVALUE])
```

(COMP.BLOCK

```
[LAMBDA (A)                                             (* lmm " 2-Jun-86 23:05")
  (if (NULL (CAR A))
      then (COMP.PROG (CONS NIL A))
    else (PROG (F)
               (SETQ F (COMP.BIND.VARS NIL NIL 'LAMBDA))
               (PROG ((BLOCKEND (create TAG
                                        LEVEL _ (COND
                                                   ((EQ COMPILE.CONTEXT 'EFFECT)
                                                    0)
                                                   (T 1))
                                        FRAME _ F))
                      (CTX (SELECTQ COMPILE.CONTEXT
                                    ((EFFECT RETURN)
                                        COMPILE.CONTEXT)
                                    NIL))
                      FLG)
                     (COMP.STBIND F)
                     [replace PROGLABELS of F with (LIST (CONS 'COMPILER-BLOCK-DATA
```

```
                                                          (create BLOCKSTATUS
                                                                  BLOCKCONTEXT _ CTX
                                                                  BLOCKTAG _ (CAR A)
                                                                  BLOCKEND _ BLOCKEND]
                    [COMP.RETURN-FROM (LIST (CAR A)
                                            (CONS 'PROGN (CDR A]
                    (OR (EQ COMPILE.CONTEXT 'RETURN)
                        (COMP.STTAG BLOCKEND)))
                (RETURN (COMP.UNBIND.VARS F])
```

(COMP.RETURN-FROM

```
  [LAMBDA (A)                                                    (* lmm " 2-Jun-86 23:10")
    (if (NULL (CAR A))
        then (COMP.RETURN (CDR A))
      else (PROG ((BLOCKFRAME FRAME)
                  DATA CTX)
           CHKLP
                [SELECTQ (fetch FRAMETYPE of BLOCKFRAME)
                     (LAMBDA (if (OR [NOT (SETQ DATA (CDR (FASSOC 'COMPILER-BLOCK-DATA (fetch PROGLABELS
                                                                                          of BLOCKFRAME]
                                     (NEQ (CAR A)
                                          (fetch BLOCKTAG of DATA)))
                                then (SETQ BLOCKFRAME (fetch PARENT of BLOCKFRAME))
                                     (GO CHKLP)))
                     (PROG (SETQ BLOCKFRAME (fetch PARENT of BLOCKFRAME))
                           (GO CHKLP))
                     (COMPERROR (CONS COMFN '(- illegal RETURN]
                (SETQ CTX (fetch BLOCKCONTEXT of DATA))
                [COND
                   ((NEQ CTX 'RETURN)
                    (COND
                       ([NOT (OR (EQ CTX 'EFFECT)
                                 (EQ LEVEL 0)
                                 (NEQ (fetch FRAMETYPE of FRAME)
                                      'PROG]                     (* RETURN POPs beforehand)
                        (COMP.STPOP LEVEL]
                (COMP.PROGLST (CDR A)
                      1 CTX)
                [COND
                   ((OPT.JUMPCHECK CODE)
                    (RETURN 'NOVALUE]
                [COND
                   ((NEQ CTX 'RETURN)
                    [until (EQ FRAME BLOCKFRAME) do (COMP.STUNBIND (EQ CTX 'EFFECT]
                    [COND
                       ((EQ CTX 'EFFECT)
                        (COMP.STPOP LEVEL))
                       ((NEQ LEVEL 1)
                        (OPT.COMPILERERROR '(unimplemented RETURN]
                    (COMP.STJUMP 'JUMP (fetch BLOCKEND of DATA]
                (RETURN 'NOVALUE])
```

(COMP.TAGBODY

```
  [LAMBDA (A)                                                    (* lmm " 2-Jun-86 23:05")
    (PROG ((VARS NIL)
           (VALS NIL)
           F)
          (SETQ F (COMP.BIND.VARS NIL NIL 'LAMBDA))
          [PROG (TAGS)
                (COMP.STBIND F)
                [for X in A do (COND
                                   ((LISTP X))
                                   [(NOT (LITATOM X))
                                    (COMPERROR (CONS X '(- illegal tag]
                                   [(FASSOC X TAGS)
                                    (COMPERROR (CONS X '(- multiply defined tag]
                                   (T (SETQ TAGS (CONS (CONS X (SETQ X (create TAG
                                                                              LBNO _ X)))
                                                       TAGS))
                                      (replace (TAG FRAME) of X with FRAME)
                                      (replace (TAG LEVEL) of X with 0]
                (replace PROGLABELS of F with TAGS)              (* Check if can delete extra POP's)
                [for X in A do (COND
                                   [(LITATOM X)
                                    (COMP.STTAG (CDR (FASSOC X TAGS]
                                   (T (COMP.EFFECT X]
                (COND
                   ((NOT (OR (EQ COMPILE.CONTEXT 'EFFECT)
                             (OPT.JUMPCHECK CODE)))               (* PROG dropped off)
                    (COMP.EXPR NIL]
          (RETURN (COMP.UNBIND.VARS F])

)

(PUTPROPS CL:LABELS BYTEMACRO COMP.LABELS)
```

```
(DEFINEQ

(COMP.LABELS
  [LAMBDA (DEF)                                                    ; Edited  2-Dec-87 12:32 by amd

;;; the byte compiler does a better job with LABELS because compiling UNDO needed it (!)

    (LET [(FUNCTIONS (MAPCAR (CAR DEF)
                             (FUNCTION (LAMBDA (X)
                                       (CONS (COMP.GENFN)
                                             X]                    ; list of functions to be substituted
         (CL:FLET [(TRANSFORM (FORM CONTEXT)
                              (CL:IF (NLISTP FORM)
                                  FORM
                                  [COND
                                     ((FMEMB (CAR FORM)
                                             '(FUNCTION CL:FUNCTION))
                                      (for Z in FUNCTIONS when (EQ (CADR FORM)
                                                                   (CADR Z))
                                         do [RETURN `',(CAR Z] finally (RETURN FORM)))
                                     (T (for Z in FUNCTIONS when (EQ (CAR FORM)
                                                                     (CADR Z))
                                           do [RETURN `(,(CAR Z)
                                                      ,@(CDR FORM]
                                           finally (RETURN FORM])]
                  (FOR Z IN FUNCTIONS
                     DO (COMP.TOPLEVEL.COMPILE
                          (CAR Z)
                          [DESTRUCTURING-BIND (FN-NAME FN-ARGLIST &REST FN-BODY)
                                 (CDR Z)
                                 (CL:MULTIPLE-VALUE-BIND (BODY DECLS)
                                     (PARSE-BODY FN-BODY NIL T)
                                   `[LAMBDA ,FN-ARGLIST
                                      ,(WALK-FORM `(CL:LOCALLY ,@DECLS (CL:BLOCK ,FN-NAME ,@BODY))
                                               :WALK-FUNCTION
                                               (FUNCTION TRANSFORM])]
                          NIL ALLVARS))
                  (for X in ALLVARS when (AND (NEQ (fetch OPNAME of X)
                                                   'AVAR)
                                              (FMEMB (fetch OPARG of X)
                                                     SUBFNFREEVARS))
                     do                                            ; change LOCALVAR to SPECVAR because subfn uses it free
                        (replace OPNAME of X with 'AVAR))
                  (COMP.EXPR (WALK-FORM `(PROGN ,@(CDR DEF))
                                        :WALK-FUNCTION
                                        (FUNCTION TRANSFORM))
                        COMPILE.CONTEXT])

)

(RPAQQ COMP.UNBOXED.TAG ("I'm on ALLDECLS if FPLUS compiles with unboxed arithmetic"))

(RPAQQ NUMBERFNS (ITIMES2 LOGOR2 LOGXOR2 LOGAND2 LLSH1 LRSH1 LLSH8 LRSH8 IPLUS ITIMES LOGOR LOGXOR LOGAND
                  IDIFFERENCE IQUOTIENT IREMAINDER IMINUS LSH LLSH RSH LRSH FIX))

(RPAQQ GLOBALVARFLG T)

(RPAQQ NEWOPTFLG NIL)

(RPAQ COMPVERSION (DATE))


(DEFOPTIMIZER IMINUS (X)
                     `(IDIFFERENCE 0 ,X))

(DECLARE%: EVAL@COMPILE

(PUTPROPS IPLUS BYTEMACRO (APPLY* COMP.NUMERIC IPLUS))

(PUTPROPS ITIMES BYTEMACRO (APPLY* COMP.NUMERIC ITIMES FIX 0))

(PUTPROPS LOGOR BYTEMACRO (APPLY* COMP.NUMERIC LOGOR FIX -1))

(PUTPROPS LOGXOR BYTEMACRO (APPLY* COMP.NUMERIC LOGXOR))

(PUTPROPS LOGAND BYTEMACRO (APPLY* COMP.NUMERIC LOGAND FIX 0))

(PUTPROPS IDIFFERENCE BYTEMACRO COMP.NUMBERCALL)

(PUTPROPS IQUOTIENT BYTEMACRO COMP.NUMBERCALL)

(PUTPROPS IREMAINDER BYTEMACRO COMP.NUMBERCALL)

(PUTPROPS LSH BYTEMACRO COMP.NUMBERCALL)

(PUTPROPS LLSH DMACRO COMP.SHIFT)
```

```
(PUTPROPS RSH BYTEMACRO COMP.NUMBERCALL)

(PUTPROPS LRSH DMACRO COMP.SHIFT)

(PUTPROPS FIX BYTEMACRO COMP.FIX)

(PUTPROPS PLUS DMACRO [APPLY* COMP.NUMERIC PLUS PLUS NIL ((FLOAT FPLUS (OPCODES UBFLOAT2 0])

(PUTPROPS DIFFERENCE DMACRO [APPLY* COMP.NUMBERCALL PLUS ((FLOAT FDIFFERENCE (OPCODES UBFLOAT2 1])

(PUTPROPS TIMES DMACRO [APPLY* COMP.NUMERIC TIMES PLUS 0 ((FLOAT FTIMES (OPCODES UBFLOAT2 3])

(PUTPROPS QUOTIENT DMACRO (APPLY* COMP.NUMBERCALL PLUS))

(PUTPROPS FPLUS DMACRO [APPLY* COMP.NUMERIC FPLUS FLOAT NIL ((FLOAT FPLUS (OPCODES UBFLOAT2 0])

(PUTPROPS FDIFFERENCE DMACRO [APPLY* COMP.NUMBERCALL FLOAT ((FLOAT FDIFFERENCE (OPCODES UBFLOAT2 1])

(PUTPROPS FTIMES DMACRO [APPLY* COMP.NUMERIC FTIMES FLOAT 0 ((FLOAT FTIMES (OPCODES UBFLOAT2 3])

(PUTPROPS FQUOTIENT DMACRO [APPLY* COMP.NUMBERCALL FLOAT ((FLOAT FQUOTIENT (OPCODES UBFLOAT2 4])

(PUTPROPS FABS DMACRO [(X)
                        (\FLOATBOX ((OPCODES UBFLOAT1 2)
                                    (\FLOATUNBOX X])

(PUTPROPS FGREATERP DMACRO (APPLY* COMP.COMPARENUM FLOAT FGREATERP NIL (OPCODES UBFLOAT2 5)))

[PROGN (PUTPROPS FLESSP MACRO [LAMBDA (X Y)
                               (FGREATERP Y X])
       (PUTPROPS FLESSP DMACRO (APPLY* COMP.COMPARENUM FLOAT FLESSP FGREATERP (OPCODES SWAP UBFLOAT2 5)))]

(PUTPROPS FREMAINDER DMACRO [APPLY* COMP.NUMBERCALL FLOAT ((FLOAT FREMAINDER (OPCODES UBFLOAT2 8])
)

(DEFINEQ

(COMP.NUMERIC
  [LAMBDA (A 2FN TYPE ZERO COERSIONS)                    ; Edited 12-Apr-88 17:03 by amd
    ;; compile call to number function of arbitrary args.  2FN is holder of opcode.  TYPE is FIX, FLOAT, PLUS (NIL->FIX)
    ;; ZERO IF GIVEN IS ZERO OF FUNCTION, E.G.  0 FOR TIMES, -1 FOR LOGOR
    ;; coercions say what to do if compile context is other numeric type
    (PROG ((N 0)
           V
           (FN (CAR EXP))
           TMP)
          [COND
             ((AND (EQ COMPILE.CONTEXT 'EFFECT)
                   (NOT OPTIMIZATIONSOFF))
              (RETURN (COMP.PROGN A]
          (OR 2FN (SETQ 2FN FN))
          (SELECTQ (CAR (LISTP COMPILE.CONTEXT))
              (TYPE [COND
                        ((AND (NEQ (CDR COMPILE.CONTEXT)
                                   TYPE)
                              (SETQ TMP (FASSOC (CDR COMPILE.CONTEXT)
                                                COERSIONS)))
                         (SETQ TYPE (CAR TMP))
                         (SETQ 2FN (CADR TMP])
              (UNBOXED (if (SETQ TMP (CADDR (FASSOC (CDR COMPILE.CONTEXT)
                                                    COERSIONS)))
                           then (while A do (COMP.EXPR (pop A)
                                                       COMPILE.CONTEXT)
                                            (SETQ N (ADD1 N)))
                                (FRPTQ (SUB1 N)
                                       (COMP.STFN TMP 2))
                                (RETURN 'UNBOXED)))
              NIL)
          (if (AND (SETQ TMP (CADDR (FASSOC TYPE COERSIONS)))
                   (FMEMB COMP.UNBOXED.TAG ALLDECLS))
              then (while A do (COMP.EXPR (pop A)
                                          (CONS 'UNBOXED TYPE))
                               (SETQ N (ADD1 N)))
                   (FRPTQ (SUB1 N)
                          (COMP.STFN TMP 2))
                   (RETURN (COMP.FLOATBOX)))
          [while A do [COMP.EXPR (pop A)
                                 (CONS 'TYPE (OR TYPE (SETQ TYPE 'FIX]
                      (SETQ N (ADD1 N))
                      (COND
                         ((NOT OPTIMIZATIONSOFF)
                          (COMP.DELFIX TYPE)
                          (while (OPT.CALLP (CAR CODE)
                                            2FN)
```

```
                        do (SETQ N (IPLUS N (CAR (fetch OPARG of (CAR CODE)))
                                           -1))
                           ;; merge nested arithmetic calls
                           (COMP.DELFN))
                     (COND
                        ((AND (EQ (fetch OPNAME of (CAR CODE))
                                  'CONST)
                              (NUMBERP (fetch OPARG of (CAR CODE)))
                              (IGREATERP N 0))
                         [SETQ V (COND
                                    [V ;; combine number args
                                       (APPLY* FN V (fetch OPARG of (CAR CODE]
                                    (T ;; move number constants to end
                                       (APPLY* (OR TYPE (FUNCTION FIX))
                                               (fetch OPARG of (CAR CODE]
                         (COMP.DELPUSH)
                         (SETQ N (SUB1 N]
          [COND
             (V (COND
                   ((EQL (APPLY* FN V)
                         (APPLY* FN))
                    ;; I.E., IS UNIT OF FUNCTION: 1 FOR TIMES, ETC
                    )
                   ((EQL V ZERO)
                    (FRPTQ N (COMP.STPOP))
                    (RETURN (COMP.STCONST V)))
                   ((AND (IGREATERP N 0)
                         (MINUSP V)
                         (EQ 2FN 'IPLUS))
                    ;; turn IPLUS of negative to IDIFFERENCE
                    (COMP.STCONST (IMINUS V))
                    (COMP.STFN 'IDIFFERENCE 2))
                   (T (COMP.STCONST V)
                      (add N 1]
          (COND
             ((EQ N 0)
              ;; number function, 0 args
              (COMP.STCONST (APPLY* FN)))
             ((EQ N 1)
              ;; number fn, 1 arg
              (COMP.STFIX TYPE))
             (T (FRPTQ (SUB1 N)
                       (COMP.STFN 2FN 2])
```

(COMP.NUMBERCALL

```
[LAMBDA (A TYPE COERSIONS)                                  (* lmm " 9-Mar-85 14:55")
  (PROG ((N 0)
         TMP
         (2FN (CAR EXP)))
        [COND
           ((AND (EQ COMPILE.CONTEXT 'EFFECT)
                 (NOT OPTIMIZATIONSOFF))
            (RETURN (COMP.PROGN A]
        (SELECTQ (CAR (LISTP COMPILE.CONTEXT))
            (TYPE [COND
                     ((AND (NEQ (CDR COMPILE.CONTEXT)
                                TYPE)
                           (SETQ TMP (FASSOC (CDR COMPILE.CONTEXT)
                                             COERSIONS)))
                      (SETQ TYPE (CAR TMP))
                      (SETQ 2FN (CADR TMP])
            (UNBOXED (if (SETQ TMP (CADDR (FASSOC (CDR COMPILE.CONTEXT)
                                                  COERSIONS)))
                         then (while A do (COMP.EXPR (pop A)
                                                     COMPILE.CONTEXT)
                                          (SETQ N (ADD1 N)))
                              (FRPTQ (SUB1 N)
                                     (COMP.STFN TMP 2))
                              (RETURN 'UNBOXED)))
            NIL)
        (if (AND (SETQ TMP (CADDR (FASSOC TYPE COERSIONS)))
                 (FMEMB COMP.UNBOXED.TAG ALLDECLS))
            then (while A do (COMP.EXPR (pop A)
                                        (CONS 'UNBOXED TYPE))
                             (SETQ N (ADD1 N)))
                 (FRPTQ (SUB1 N)
                        (COMP.STFN TMP 2))
                 (RETURN (COMP.FLOATBOX)))
        (while A do (COMP.VAL (pop A))
```

```
                    [COND
                       ((NOT OPTIMIZATIONSOFF)
                        (COMP.DELFIX TYPE)                    (* remove extraneous FIX, FLOAT calls)
                        (COND
                           ((AND (NEQ TYPE 'PLUS)
                                 (EQ (fetch OPNAME of (CAR CODE))
                                     'CONST))                 (* if FIX or FLOAT type and arg is constant, then coerce.)
                            (COMP.STCONST (APPLY* (OR TYPE 'FIX)
                                                  (PROG1 (fetch OPARG of (CAR CODE))
                                                         (COMP.DELPUSH]
                    (SETQ N (ADD1 N)))
          [COND
             ((AND (NOT OPTIMIZATIONSOFF)
                   (EQ (fetch OPNAME of (CAR CODE))
                       'CONST)
                   (EQ N 2))
              (COND
                 ((EQ (fetch OPNAME of (CAR (fetch PREV of CODE)))
                      'CONST)
                  (COMP.STCONST (PROG1 (APPLY* (CAR EXP)
                                               (fetch OPARG of (CAR (fetch PREV of CODE)))
                                               (fetch OPARG of (CAR CODE)))
                                       (COMP.DELPUSH)
                                       (COMP.DELPUSH)))
                  (RETURN (COMP.STFIX TYPE)))
                 ((FMEMB 2FN (SELECTQ (fetch OPARG of (CAR CODE))
                                      (0 '(IDIFFERENCE LSH RSH LLSH LRSH))
                                      (1 '(IQUOTIENT))
                                      NIL))
                  (COMP.DELPUSH)
                  (RETURN (COMP.STFIX TYPE]
          (RETURN (COMP.STFN 2FN N])
```

(COMP.FIX

```
  [LAMBDA (A)                                                 (* lmm "18-APR-80 18:28")
    (COMP.VAL1 A)
    (COMP.STFIX])
```

(COMP.STFIX

```
  [LAMBDA (TYPE)                                              (* lmm "13-Jul-84 21:18")
    (OR TYPE (SETQ TYPE 'FIX))
    (COND
       [[AND (EQ (fetch OPNAME of (CAR CODE))
                 'CONST)
             (NUMBERP (fetch OPARG of (CAR CODE]              (* COMPILE TIME FIX)
        (COMP.STCONST (PROG1 (APPLY* TYPE (fetch OPARG of (CAR CODE)))
                             (COMP.DELPUSH]
       ((AND (EQ TYPE 'FIX)
             (OPT.CALLP (CAR CODE)
                    NUMBERFNS)))
       (T (COMP.STFN TYPE 1])
```

(COMP.DELFIX

```
  [LAMBDA (TYPE)                                              (* lmm "16-APR-82 00:19")
                                                              (* have compiled call to number function;
                                                              delete any coersions-to-TYPE)
    (while (OPT.CALLP (CAR CODE)
                      (SELECTQ TYPE
                           ((FIX NIL)
                                '(IPLUS FIX))
                           (FLOAT 'FLOAT)
                           'PLUS)
                      1)
       do (COMP.DELFN])
)

(PUTPROPS EQ BYTEMACRO COMP.EQ)

(PUTPROPS EQUAL BYTEMACRO COMP.EQ)

(PUTPROPS EQP BYTEMACRO COMP.EQ)

(DEFINEQ
```

(COMP.EQ

```
  [LAMBDA (A)                                                 (* lmm " 2-Jan-85 00:23")
    (COND
       ((EQ COMPILE.CONTEXT 'EFFECT)
        (COMP.PROGN A))
       (T (PROG (C)
                (COMP.VAL (pop A))
                [COND
                   ((OR OPTIMIZATIONSOFF (NEQ (fetch OPNAME of (CAR CODE))
```

```
                                          'CONST))
                    (COMP.PROGLST A 1))
                   ([NULL (SETQ C (fetch OPARG of (CAR CODE]       (* (EQ NIL --))
                    (COMP.DELPUSH)
                    (RETURN (COMP.NOT A)))
                   (T (COMP.DELPUSH)
                      (COMP.PROGLST A 1)
                      (COND
                         [(EQ (fetch OPNAME of (CAR CODE))
                              'CONST)                              (* (EQ CONST CONST))
                          (RETURN (COMP.STCONST (PROG1 (APPLY* (CAR EXP)
                                                               C
                                                               (fetch OPARG of (CAR CODE)))
                                                       (COMP.DELPUSH]
                         (T                                        (* (EQ CONST EXPRESSION))
                            (COMP.STCONST C]
               (RETURN (COMP.STFN (COND
                                     ([AND (EQ (fetch OPNAME of (CAR CODE))
                                               'CONST)
                                           (LITATOM (fetch OPARG of (CAR CODE]
                                                                   (* EQ IFF EQUAL)
                                      'EQ)
                                     (T (CAR EXP)))
                                  2])

)

(PUTPROPS .TEST. BYTEMACRO (APPLY COMP.NUMBERTEST))

(DEFINEQ

(COMP.NUMBERTEST
  [LAMBDA (X FORM FLG)                                             (* lmm "13-Jul-84 21:18")
    (PROG (EXIT (TEST (SUBPAIR '                                   (*)
                               (LIST DONOTHING)
                               FORM))
                A)
          (COMP.EXPR X)
          (RETURN (SELECTQ (AND (COMP.PREDP COMPILE.CONTEXT)
                                (fetch OPNAME of COMPILE.CONTEXT))
                           ((FJUMP TJUMP NFJUMP)                   (* .TEST. in PREDF)
                              (COMP.EXPR TEST COMPILE.CONTEXT))
                           (NTJUMP [COND
                                      ((OR (FMEMB (fetch OPNAME of (SETQ A (CAR CODE)))
                                                  '(AVAR HVAR GVAR FVAR))
                                           (AND (EQ (fetch OPNAME of A)
                                                    'SETQ)
                                                (PROGN (SETQ A (fetch OPARG of A))
                                                       T)))        (* .TEST. VAR in NTJUMP)
                                       [COMP.EXPR TEST (create JUMP
                                                               OPNAME _ 'FJUMP
                                                               TAG _ (SETQ EXIT (create TAG]
                                       (COMP.STVAR A)
                                       (COMP.STJUMP 'JUMP (fetch (JUMP TAG) of COMPILE.CONTEXT))
                                       (COMP.STTAG EXIT)
                                       (RETURN 'PREDVALUE))
                                      (T                           (* .TEST. in NTJUMP PREDF)
                                         (COMP.STCOPY)
                                         [COMP.EXPR TEST (create JUMP
                                                                 OPNAME _ 'FJUMP
                                                                 TAG _ (SETQ EXIT (create TAG]
                                         (COMP.STJUMP 'JUMP (fetch (JUMP TAG) of COMPILE.CONTEXT))
                                         (COMP.STTAG EXIT)
                                         (COMP.STPOP)
                                         (RETURN 'PREDVALUE])
                           (COND
                              ((OR (FMEMB (fetch OPNAME of (SETQ A (CAR CODE)))
                                          '(AVAR HVAR GVAR FVAR))
                                   (AND (EQ (fetch OPNAME of A)
                                            'SETQ)
                                        (PROGN (SETQ A (fetch OPARG of A))
                                               T)))                (* .TEST. VAR not in PREDF)
                               [COMP.EXPR TEST (create JUMP
                                                       OPNAME _ 'NFJUMP
                                                       TAG _ (SETQ EXIT (create TAG]
                               (COMP.STVAR A)
                               (COMP.STTAG EXIT))
                              (T                                   (* .TEST. not in PREDF)
                                 (COMP.STCOPY)
                                 [COMP.EXPR TEST (create JUMP
                                                         OPNAME _ 'TJUMP
                                                         TAG _ (SETQ EXIT (create TAG]
                                 (COMP.STPOP)
                                 (COMP.STCONST)
                                 (COMP.STTAG EXIT])

)
```

```
(RPAQQ MAPFNS (MAP MAPC MAPLIST MAPCAR MAPCON MAPCONC SUBSET SOME EVERY NOTANY NOTEVERY))

(PUTPROPS MAP BYTEMACRO (APPLY* COMP.MAP))

(PUTPROPS MAPC BYTEMACRO (APPLY* COMP.MAP T))

(PUTPROPS MAPLIST BYTEMACRO (APPLY* COMP.MAP NIL T))

(PUTPROPS MAPCAR BYTEMACRO (APPLY* COMP.MAP T T))

(PUTPROPS MAPCON BYTEMACRO (APPLY* COMP.MAP NIL J))

(PUTPROPS MAPCONC BYTEMACRO (APPLY* COMP.MAP T J))

(PUTPROPS SUBSET BYTEMACRO (APPLY* COMP.MAP T S))

(PUTPROPS SOME BYTEMACRO (APPLY* COMP.MAP BOTH NIL TJUMP))

(PUTPROPS EVERY BYTEMACRO (APPLY* COMP.MAP BOTH NIL FJUMP T))

(PUTPROPS NOTANY BYTEMACRO (APPLY* COMP.MAP BOTH NIL TJUMP T))

(PUTPROPS NOTEVERY BYTEMACRO (APPLY* COMP.MAP BOTH NIL FJUMP NIL))

(PUTPROPS .DOCOLLECT. BYTEMACRO [(VAL TAIL ITEM)
                                 (COND
                                    [(NOT TAIL)
                                     (SETQ TAIL (SETQ VAL (LIST ITEM]
                                    (T (FRPLACD TAIL (SETQ TAIL (LIST ITEM])

(PUTPROPS .DOJOIN. BYTEMACRO [(VAL TAIL ITEM)
                              (AND (LISTP ITEM)
                                   (COND
                                      (TAIL (FRPLACD (SETQ TAIL (LAST TAIL))
                                                     ITEM))
                                      (T (SETQ TAIL (SETQ VAL ITEM])

(DEFINEQ

(COMP.MAP
  [LAMBDA (L CARFLG COLLECT PRED NEG WHILEF)                    (* lmm "18-Sep-84 17:05")
                                                                (* compile call to mapping function)
    (PROG [(FROMFORM (CAR L))
           (DOF (CADR L))
           (BYF (CADDR L))
           BOUNDVARS BINDVALS F VAL (XARG '($X]
          (COMP.PROGLST (CDDDR L)
                 0)
          [COND
             [(COMP.APPLYFNP DOF)
              (SETQ DOF (CADR DOF))
              (COND
                 ((AND (NOT CARFLG)
                       (EQ (CAR (LISTP DOF))
                           'LAMBDA))                            (* leave DOF alone)
                  NIL)
                 (T (SETQ DOF (LIST 'LAMBDA XARG (CONS DOF (COND
                                                             ([AND (EQ CARFLG 'BOTH)
                                                                   (NOT (AND (COMP.CLEANFNP DOF 'NARGS)
                                                                             (EQ (NARGS DOF)
                                                                                 1]
                                                              '((CAR $X)
                                                                $X))
                                                             [CARFLG '((CAR $X]
                                                             (T '($X]
             (T                                                 (* map function with computed functional arg)
                (SETQ BINDVALS (LIST DOF FROMFORM))
                [SETQ BOUNDVARS (LIST '$F1 (SETQ FROMFORM '$L]
                (SETQ DOF (LIST 'LAMBDA XARG (SELECTQ CARFLG
                                                 (BOTH '(APPLY* $F1 (CAR $X)
                                                               $X))
                                                 (NIL '(APPLY* $F1 $X))
                                                 '(APPLY* $F1 (CAR $X]
          [COND
             ((NULL BYF)
              (SETQ BYF 'CDR))
             [(COMP.APPLYFNP BYF)                               (* mapping function with BY argument)
              (OR (EQ [CAR (LISTP (SETQ BYF (CADR BYF]
                      'LAMBDA)
                  (SETQ BYF (LIST 'LAMBDA XARG (LIST BYF '$X]
             (T                                                 (* mapping function with computed BY argument)
                (SETQ BINDVALS (CONS BYF BINDVALS))
                (SETQ BOUNDVARS (CONS '$F2 BOUNDVARS))
                (SETQ BYF '(LAMBDA ($X)
                             (COND
                                ((NULL $F2)
```

```
                          (CDR $X))
                         (T (APPLY* $F2 $X]
      [COND
         ((NULL WHILEF)
          (SETQ WHILEF 'LISTP))
         [(COMP.APPLYFNP WHILEF)
          (OR (EQ [CAR (LISTP (SETQ WHILEF (CADR WHILEF]
                  'LAMBDA)
              (SETQ WHILEF (LIST 'LAMBDA XARG (LIST WHILEF '$X]
         (T (SETQ BINDVALS (CONS (LIST 'OR WHILEF ''LISTP)
                                 BINDVALS))
            (SETQ BOUNDVARS (CONS '$F3 BOUNDVARS))
            (SETQ WHILEF '(LAMBDA ($X)
                          (APPLY* $F3 $X]
      [COND
         (COLLECT (push BINDVALS NIL NIL NIL NIL)
                  (push BOUNDVARS (SETQ VAL '$V)
                        '$Z
                        '$W
                        '$X]                                    (* bind extra vars)
      (SETQ F (COMP.BIND.VARS (OPT.DREV BOUNDVARS)
                              (OPT.DREV BINDVALS)
                              'MAP))
      [PROG ((ALLVARS (APPEND (fetch VARS of F)
                              ALLVARS))
             (SPECVARS SPECVARS)
             (LOCALVARS LOCALVARS)
             (LP (create TAG))
             (ENDLP (create TAG))
             (OUT (create TAG))
             NXT)
            (COMP.STBIND F)
            [COMP.EFFECT '(DECLARE (LOCALVARS $F1 $F2 $X $V $Z $W $F3]
            (COMP.VAL FROMFORM)
            (OPT.CCHECK (AND (EQ LEVEL 1)
                             (EQ FRAME F)))
            (COMP.STJUMP 'JUMP ENDLP)
            (SETQ LEVEL 1)
            (SETQ FRAME F)
            (COMP.STTAG LP)
            (COMP.STCOPY)
            [COND
               (COLLECT (OPT.CCHECK (NOT PRED))
                        (SELECTQ COLLECT
                             ((T J)                             (* collect or join)
                              (COMP.EFFECT (LIST 'SETQ '$X DONOTHING))
                              [COMP.EFFECT (LIST 'SETQ '$W (COND
                                                             ((EQ (CADR DOF)
                                                                  XARG)
                                                              (CADDR DOF))
                                                             (T (LIST DOF '$X]
                              [COMP.EFFECT (SELECTQ COLLECT
                                                    (J '(.DOJOIN. $V $Z $W))
                                                    '(.DOCOLLECT. $V $Z $W])
                             (S                                 (* SUBSET)
                                [COMP.EXPR (LIST DOF DONOTHING)
                                           (create JUMP
                                                   OPNAME _ 'FJUMP
                                                   TAG _ (SETQ NXT (create TAG]
                                (COMP.STCOPY)
                                (COMP.EFFECT (LIST 'SETQ '$W (LIST 'CAR DONOTHING)))
                                (COMP.EFFECT '(.DOCOLLECT. $V $Z $W))
                                (COMP.STTAG NXT))
                             (SHOULDNT)))
               (PRED (COMP.EXPR (LIST DOF DONOTHING)
                                (create JUMP
                                        OPNAME _ PRED
                                        TAG _ OUT)))
               (T (COMP.EFFECT (LIST DOF DONOTHING]
            (OPT.CCHECK (EQ LEVEL 1))
            (COMP.EXPR (LIST BYF DONOTHING))                    (* get next element)
            (COMP.STTAG ENDLP)
            (COMP.EXPR (LIST WHILEF DONOTHING))
            (COMP.STJUMP 'NTJUMP LP)
            (COND
               [PRED (COND
                        ((AND (EQ PRED 'TJUMP)
                              (NULL NEG))
                         (COMP.VAL NIL)
                         (COMP.STTAG OUT))
                        (T (COMP.VAL NEG)
                           (COMP.STJUMP 'JUMP (SETQ NXT (create TAG)))
                           (COMP.STTAG OUT)
                           (COMP.STPOP)
                           (COMP.VAL (NULL NEG))
                           (COMP.STTAG NXT]
               (T (COMP.VAL VAL]
```

```
          (RETURN (COMP.UNBIND.VARS F])

)

(PUTPROPS LISPXWATCH BYTEMACRO T)

(DEFOPTIMIZER BLKAPPLY (&REST ARGS)
                       (CONS 'APPLY ARGS))

(DEFOPTIMIZER BLKAPPLY* (&REST ARGS)
                        (CONS 'APPLY* ARGS))

(DEFOPTIMIZER ADD1VAR (X)
                      `(SETQ ,X (ADD1 ,X)))

(DEFOPTIMIZER KWOTE (&REST ARGS)
                    (CONS '(OPENLAMBDA (Q)
                              (COND
                                 ((AND Q (NEQ Q T)
                                       (NOT (NUMBERP Q)))
                                  (LIST 'QUOTE Q))
                                 (T Q)))
                          ARGS))

(DEFOPTIMIZER FRPLNODE (&REST ARGS)
                       (CONS '(OPENLAMBDA (X A D)
                                 (FRPLACD (FRPLACA X A)
                                          D))
                             ARGS))

(DEFOPTIMIZER RPLNODE (&REST ARGS)
                      (CONS '(OPENLAMBDA (X A D)
                                (RPLACD (RPLACA X A)
                                        D))
                            ARGS))

(DEFOPTIMIZER LISTGET1 (&REST ARGS)
                       (CONS '(OPENLAMBDA (X Y)
                                 (CADR (MEMB Y X)))
                             ARGS))

(DEFOPTIMIZER FRPLNODE2 (&REST ARGS)
                        (CONS '(OPENLAMBDA (X Y)
                                  (FRPLACD (FRPLACA X (CAR Y))
                                           (CDR Y)))
                              ARGS))

(PUTPROPS SUB1VAR BYTEMACRO ((X)
                             (SETQ X (SUB1 X))))

(DEFOPTIMIZER EQMEMB (&REST ARGS)
                     (CONS '(OPENLAMBDA (X Y)
                               (OR (EQ X Y)
                                   (AND (LISTP Y)
                                        (FMEMB X Y)
                                        T)))
                           ARGS))

(DEFOPTIMIZER MKLIST (&REST ARGS)
                     (CONS '[OPENLAMBDA (X)
                               (OR (LISTP X)
                                   (AND X (LIST X]
                           ARGS))
```

;; Pass 1 listing

```
(DEFINEQ

(COMP.MLLIST
  [LAMBDA (FN CC)                                    ; Edited 13-Jun-2021 09:50 by rmk:
    (RESETLST
        (RESETSAVE (RADIX 10))
        (RESETSAVE (LINELENGTH 72))
        (PRIN2 FN)
        (MAPRINT (fetch (COMINFO ARGS) of CC)
               NIL "(" ")" " " (FUNCTION COMP.MLLVAR))
        (SPACES 5)
```

```
        [PRINT (CDR (FASSOC (fetch (COMINFO COMTYPE) of CC)
                            '((0 . LAMBDA)
                              (2 . LAMBDA*)
                              (1 . NLAMBDA)
                              (2 . NLAMBDA*)
                              (NIL . ???]
        (COMP.MLL (fetch (COMINFO CODE) of CC)))])
```

(COMP.MLL

```
  [LAMBDA (LL)                                                    (* Pavel "15-Nov-86 16:02")
    [for X in LL
       do (if (type? TAG X)
              then (if (NOT (ZEROP (POSITION)))
                       then (TERPRI))
                   (PRIN2 (fetch (TAG LBNO) of X))
                   (PRIN1 '%:)
            else (PROG ((S (GETPROP (fetch OPNAME of X)
                                    'MLSYM))
                        (P (POSITION)))
                   (if (ILESSP P 5)
                       then (SPACES (IDIFFERENCE 6 P))
                     elseif (IGREATERP P 60)
                       then (TERPRI)
                            (SPACES 6)
                     else (SPACES 1))
                   (AND (CAR S)
                        (PRIN1 (CAR S)))
                   [SELECTQ (CDDR S)
                        (CONST (PRIN2 (FETCH OPARG OF X)))
                        (VAR (COMP.MLLVAR X))
                        (FN                                       ; FN and LINKEDFN
                            (COMP.MLLFN X))
                        (VREF                                     ; SETQ ARG
                              (COMP.MLLVAR (fetch OPARG of X)))
                        (JUMP (PRIN2 (fetch (TAG LBNO) of (fetch (JUMP TAG) of X))))
                        (BIND (PROG [NN N (F (CDR (FETCH OPARG OF X]
                                    (SETQ N (SETQ NN (FETCH NVALS OF F)))
                                    (FOR V IN (FETCH VARS OF F)
                                       DO (PRIN1 (IF (EQ N NN)
                                                     THEN    ; 1ST one
                                                          ""
                                                   ELSEIF (ZEROP N)
                                                     THEN ';
                                                   ELSE '%,))
                                          (SETQ N (IPLUS N -1))
                                          (COMP.MLLVAR V))
                                    (if (ZEROP N)
                                        then                      ; All val-bound
                                             (PRIN1 ";"))))
                        (UNBIND (PRIN1 (CAR (fetch OPARG of X))))
                        (PROGN (PRIN1 (fetch OPNAME of X))
                               (AND (fetch OPARG of X)
                                    (PRIN1 (LIST (fetch OPARG of X]
                   (AND (CADR S)
                        (PRIN1 (CADR S]
    (TERPRI)
    (TERPRI])
```

(COMP.MLLVAR

```
  [LAMBDA (X N)                                                   (* Pavel "15-Nov-86 16:02")
    (SETQ N (FETCH (VAR VARNAME) OF X))
    (PRIN2 (SELECTQ (FETCH OPNAME OF X)
                (HVAR (PRIN1 "@")
                      N)
                (XVAR 'XVAR)
                N])
```

(COMP.MLLFN

```
  [LAMBDA (X FN)                                                  (* Pavel "15-Nov-86 16:03")
    [PRIN2 (SETQ FN (CDR (FETCH OPARG OF X]
    (SETQ X (CAR (FETCH OPARG OF X)))
    (AND (LITATOM FN)
         (OR (AND (ZEROP (ARGTYPE FN))
                  (EQ (NARGS FN)
                      X))
             (PROGN (SPACES 1)
                    (PRIN2 X])

)

(RPAQQ COPS (BIND UNBIND DUNBIND ERRORSET JUMP TJUMP FJUMP NTJUMP NFJUMP POP COPY RETURN TAG FN CONST SETQ AVAR
                 HVAR GVAR FVAR STORE))

(PUTPROPS BIND MLSYM ("BIND[" %] . BIND))
```

```
(PUTPROPS UNBIND MLSYM ("UNBIND(" %) . UNBIND))

(PUTPROPS DUNBIND MLSYM ("DUNBIND(" %) . UNBIND))

(PUTPROPS ERRORSET MLSYM ("ERRORSET " %
 . JUMP))

(PUTPROPS JUMP MLSYM ("JUMP " %
 . JUMP))

(PUTPROPS TJUMP MLSYM ("TJUMP " %
 . JUMP))

(PUTPROPS FJUMP MLSYM ("FJUMP " %
 . JUMP))

(PUTPROPS NTJUMP MLSYM ("NTJUMP " %
 . JUMP))

(PUTPROPS NFJUMP MLSYM ("NFJUMP " %
 . JUMP))

(PUTPROPS FN MLSYM (%[ %] . FN))

(PUTPROPS CONST MLSYM ("'" NIL . CONST))

(PUTPROPS SETQ MLSYM ("SETQ<" > . VREF))

(PUTPROPS AVAR MLSYM (< > . VAR))

(PUTPROPS HVAR MLSYM (< > . VAR))

(PUTPROPS GVAR MLSYM (< > . VAR))

(PUTPROPS FVAR MLSYM (< > . VAR))
```

;; ARJ --- JUMP LENGTH RESOLVER

```
(DEFINEQ
```

(OPT.RESOLVEJUMPS

```
  [LAMBDA (JL PROP FN)                                        (* lmm "19-JUL-80 10:00")
    (PROG ((CU 0)
           Z NEW)
          [for X in JL do (replace JSN of X with (fetch JMIN of X))
                          (COND
                             [(fetch JPT of X)                (* Jump)
                              (SETQ Z (CAR (GETPROP (fetch OPNAME of (CAR (fetch JPT of X)))
                                                    PROP)))
                              (replace JML of X with (CAR Z))
                              (add CU (replace JU of X with (IDIFFERENCE (CDR Z)
                                                                         (CAR Z]
                             (T                               (* Tag)
                                (replace JU of X with CU]
          (while (LISTP (SETQ NEW (OPT.JLENPASS JL PROP))) do (SETQ JL NEW))
          (COND
             (NEW (OPT.JFIXPASS JL FN])
```

(OPT.JLENPASS

```
  [LAMBDA (JL PROP)                                           (* lmm "19-JUL-80 10:08")
    (PROG ((INC 0)
           (DEC 0)
           (CU 0)
           X U U1 DEF MIN ML SMIN SMAX)
```

(* JPT is NIL (for tags) or a pointer into ACODE (for jumps)%. JMIN is the lowest possible location for the instruction or tag. JU is the cumulative uncertainty (for tags) or the length uncertainty (for jumps)%. JML is the minimum length (for jumps)%. JSN is a serial number (the original JMIN) used to decide whether a jump goes forward or backward.)

(* In the loop, CU is the cumulative uncertainty, DEC is the cumulative decrease in uncertainty, and INC is the cumulative increase in minimum location.)

```
          [for J in JL do (SETQ X (CAR (fetch JPT of J)))
                          (add (fetch JMIN of J)
                               INC)
                          (COND
                             ((NULL X)
                              (SETQ DEC (IDIFFERENCE CU (fetch JU of J)))
                              (replace JU of J with CU))
                             ((NEQ (SETQ U (fetch JU of J))
                                   0)
                              [SETQ DEF (fetch (TAG JD) of (CAR (fetch OPARG of X]
                              (SETQ MIN (IDIFFERENCE (fetch JMIN of DEF)
                                                     (fetch JMIN of J)))
```

```
                              (SETQ SMAX (OPT.JSIZE X (IPLUS (IDIFFERENCE (fetch JU of DEF)
                                                                  CU)
                                                              (COND
                                                                 ((IGREATERP (fetch JSN of DEF)
                                                                         (fetch JSN of J))
                                                                  (IPLUS (SETQ MIN (IPLUS MIN INC))
                                                                         DEC))
                                                                 (T MIN)))
                                                        PROP))
                              (SETQ SMIN (OPT.JSIZE X MIN PROP))
                              [COND
                                 ((NEQ SMIN (SETQ ML (fetch JML of J)))
                                  (replace JML of J with SMIN)
                                  (add INC (IDIFFERENCE SMIN ML]
                              (COND
                                 ((NEQ (SETQ U1 (IDIFFERENCE SMAX SMIN))
                                       U)
                                  [COND
                                     ((ILESSP U1 0)
                                      (OPT.COMPILERERROR '(U1 negative]
                                  (add DEC (IDIFFERENCE U1 U))
                                  (replace JU of J with U1)))
                              (add CU U1]
          (RETURN (COND
                     ((AND (NEQ DEC 0)
                           (NEQ CU 0))
                      JL)
                     (T T])
```

(OPT.JFIXPASS

```
  [LAMBDA (JL FN)                                                    (* lmm "19-JUL-80 10:23")
    (PROG (X)
          (for J in JL do (COND
                             ([NULL (SETQ X (CAR (fetch JPT of J]
                              (replace JU of J with 0))
                             (T (APPLY* FN (fetch JPT of J)
                                           (IDIFFERENCE [fetch JMIN of (fetch (TAG JD)
                                                                              of (CAR (fetch OPARG of X]
                                                        (fetch JMIN of J])
```

(OPT.JSIZE

```
  [LAMBDA (OP D FN)                                                  (* lmm "27-OCT-81 20:28")
    (PROG [(Z (CDR (GETPROP (fetch OPNAME of OP)
                            FN]
      LP  (COND
             ((NLISTP Z)
              (RETURN Z))
             (T [SETQ Z (COND
                           ((ILESSP D (CAR Z))
                            (CADR Z))
                           (T (CDDR Z]
                (GO LP])

)

;; Utilities used by all files

(DEFINEQ

(OPT.CALLP
  [LAMBDA (OP FN N)                                                  (* lmm%: "22-JUL-77 02:40")
    (AND (EQ (fetch OPNAME of OP)
             'FN)
         (OR (NULL N)
             (EQ (CAR (fetch OPARG of OP))
                 N))
         (OR (NULL FN)
             (EQ (CDR (fetch OPARG of OP))
                 FN)
             (AND (LISTP FN)
                  (FMEMB (CDR (fetch OPARG of OP))
                         FN])

(OPT.JUMPCHECK
  [LAMBDA (C)                                                        (* lmm%: "22-JUL-77 02:39")
    (SELECTQ (fetch OPNAME of (CAR C))
         ((JUMP RETURN)
             T)
         NIL])

(OPT.DREV
  [LAMBDA (L Z)
    (PROG (Y)
```

```
      R1  (COND
             ((NLISTP (SETQ Y L))
              (RETURN Z)))
          (SETQ L (CDR L))
          (SETQ Z (FRPLACD Y Z))
          (GO R1])
```

(OPT.CHLEV

```
  [LAMBDA (N)                                                    (* lmm "14-MAR-81 09:54")
    (COND
       (LEVEL (PROG1 (add LEVEL N)
                     (OPT.CCHECK (IGEQ LEVEL 0)))])
```

(OPT.CHECKTAG

```
  [LAMBDA (TAG TAGFLAG)                                          (* lmm "14-MAR-81 09:15")
    (COND
       ((NULL LEVEL)
        (replace (TAG LEVEL) of TAG with NIL))
       ((NULL (fetch (TAG LEVEL) of TAG))
        (AND TAGFLAG (SETQ LEVEL NIL)))
       (T (OPT.CCHECK (EQ LEVEL (fetch (TAG LEVEL) of TAG)))
          T])
```

(OPT.NOTJUMP

```
  [LAMBDA (X)                                                    (* lmm%: "22-JUL-77 03:39")
    (PROG NIL
          (RETURN (create OP
                          OPNAME _ (OR (SELECTQ (fetch OPNAME of X)
                                                (FJUMP 'TJUMP)
                                                (TJUMP 'FJUMP)
                                                NIL)
                                       (RETURN))
                          OPARG _ (fetch OPARG of X])
```

(OPT.INITHASH

```
  [NLAMBDA (X)                                                   ; Edited  3-Oct-88 16:42 by tal
    (DECLARE (LOCALVARS . T))
    (LET ((H (EVALV X)))
         (COND
            [(HARRAYP H)
             (COND
                ((NEQ (HARRAYPROP H 'NUMKEYS)
                      0)
                 (CLRHASH H]
            (T (SET X (HASHARRAY 100])
```

(OPT.COMPINIT

```
  [LAMBDA NIL                                                    (* lmm%: "22-JUL-77 16:51")
    [MAPC '((OPRETURN . RETURN)
            (OPPOP . POP)
            (OPCOPY . COPY)
            (OPNIL . CONST))
          (FUNCTION (LAMBDA (X)
                      (SET (CAR X)
                           (create OP
                                   OPNAME _ (CDR X]
    (SETQ DONOTHING (LIST 'AC])

)

(MOVD? 'NILL 'REFRAME)

(AND (GETD 'OPT.COMPINIT)
     (OPT.COMPINIT))

(PUTPROPS LOADTIMECONSTANT BYTEMACRO (= . DEFERREDCONSTANT))

(PUTPROPS FRPTQ BYTEMACRO OPT.CFRPTQ)

(DEFINEQ
```

(OPT.CFRPTQ

```
  [LAMBDA (L)                                                    (* lmm "29-Jun-84 08:25")
    (COND
       ((EQ COMPILE.CONTEXT 'EFFECT)
        (PROG ((END (create TAG))
               (ST (create TAG)))
              (COMP.VAL (CAR L))                                 (* counter)
              (COMP.STTAG ST)
              (COMP.STCOPY)
              (COMP.VAL 0)
              (COMP.STFN 'IGREATERP 2)
```

```
              (COMP.STJUMP 'FJUMP END)
              (COMP.VALN (CDR L)
                    'EFFECT)
              (COMP.VAL 1)
              (COMP.STFN 'IDIFFERENCE 2)
              (COMP.STJUMP 'JUMP ST)
              (COMP.STTAG END)))
       (T (COMP.EXP1 (CONS 'RPTQ L])

)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(SPECVARS AC ALAMS1 ALLVARS ARGS ARGVARS BLKDEFS BLKFLG CODE COMFN COMFNS COMTYPE CONSTS EMFLAG EXP FRAME
       FREELST FREEVARS LAPFLG LBCNT LEVEL LOCALVARS LOCALVARS LSTFIL MACEXP NLAMS1 PIFN COMPILE.CONTEXT
       PROGCONTEXT RETURNLABEL SPECVARS SPECVARS SUBFNFREEVARS TAGS TOPFN TOPFRAME TOPLAB VARS INTERNALBLKFNS)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(SPECVARS PLVLFILEFLG)
)
)

(PUTPROPS IMAX2 BYTEMACRO (OPENLAMBDA (X Y)
                            (COND
                               ((NOT (IGREATERP X Y))
                                Y)
                               (T X))))

(PUTPROPS IMIN2 BYTEMACRO (OPENLAMBDA (X Y)
                           (COND
                              ((IGREATERP X Y)
                               Y)
                              (T X))))

(PUTPROPS FLOAT BOX (\FLOATBOX . \FLOATUNBOX))

(DEFINEQ
```

(COMP.AREF

```
  [LAMBDA (A)                                                    (* raf "18-Jun-85 17:52")
    (PROG (DECL)
          [COND
             ([AND (LITATOM (CAR A))
                   (EQ [CAR (SETQ DECL (CDR (FASSOC (COMP.LOOKUPVAR (CAR A))
                                                    ALLDECLS]
                       'ARRAY)
                   (EQ (LENGTH (MKLIST (CADDR DECL)))
                       (LENGTH (CDR A]
              (COND
                 ((EQUAL (CADR DECL)
                         '(BYTE 16))
                  (RETURN (COMP.EXPR (CONS '\16AREF A)
                                     COMPILE.CONTEXT)))
                 ((FMEMB (CADR DECL)
                         '(FLOATP FLONUM))
                  (RETURN (COMP.EXPR (CONS '\LAREF A)
                                     COMPILE.CONTEXT)))
                 (T (HELP]
          (MAPC A (FUNCTION COMP.VAL))
          (COMP.STFN (SELECTQ (LENGTH A)
                              (2 '\AREF.1)
                              (3 '\AREF.2)
                              'CL:AREF)
                     (LENGTH A])
```

(COMP.ASET

```
  [LAMBDA (A)                                                    (* kbr%: "12-Mar-85 17:08")
    (PROG (DECL)
          [COND
             ([AND (LITATOM (CADR A))
                   (EQ [CAR (SETQ DECL (CDR (FASSOC (COMP.LOOKUPVAR (CADR A))
                                                    ALLDECLS]
                       'ARRAY)
                   (EQ (LENGTH (MKLIST (CADDR DECL)))
                       (LENGTH (CDDR A]
              (COND
                 ((EQUAL (CADR DECL)
                         '(BYTE 16))
                  (RETURN (COMP.EXPR (CONS '\16ASET A)
                                     COMPILE.CONTEXT)))
                 ((FMEMB (CADR DECL)
                         '(FLOATP FLONUM))
```

```
                (RETURN (COMP.EXPR (CONS '\LASET A)
                                   COMPILE.CONTEXT)))
              (T (HELP]
        (MAPC A (FUNCTION COMP.VAL))
        (COMP.STFN (SELECTQ (LENGTH A)
                           (3 '\ASET.1)
                           (4 '\ASET.2)
                           'ASET)
              (LENGTH A])
```

(COMP.BOX

```
[LAMBDA (TYPE)                                          (* lmm " 1-Jul-84 17:45")
  (PROG [(BOXER (AND (LITATOM TYPE)
                     (GETPROP TYPE 'BOX]
        (if BOXER
            then (if (OPT.CALLP (CAR CODE)
                                (CDR BOXER)
                                1)
                     then                               (* top of stack was (unbox value)%, just get rid of BOX)
                          (COMP.DELFN)
                          (COMP.STFIX TYPE)
                     else (COMP.STFN (CAR BOXER)
                                     1])
```

(COMP.LOOKFORDECLARE

```
[LAMBDA (EXPS)                                          (* lmm " 1-Jul-84 16:54")
  (while (EQ (CAR (LISTP (CAR EXPS)))
             COMMENTFLG)
     do (pop EXPS))
  (if (EQ (CAR (LISTP (CAR EXPS)))
          'DECLARE)
      then (for Y in (CDAR EXPS) bind DECLS
              do (SELECTQ (CAR Y)
                     (TYPE [for Z in (CDDR Y) do (push DECLS (CONS Z (COMP.DECLARETYPE (CADR Y])
                     NIL)
              finally (RETURN DECLS])
```

(COMP.DECLARETYPE

```
[LAMBDA (X)                                             (* lmm "13-Jul-84 22:19")
                                                        (* returns a valid compile context, too)
  (SELECTQ X
       ((FLOATING FLOATP FLOAT)

         (* if you declare a variable to be FLOAT, you are really saying to hold it "unboxed")

          '(UNBOXED . FLOAT))
       (if (LISTP X)
           then (SELECTQ (CAR X)
                     (ARRAY X)
                     NIL])
```

(COMP.FLOATBOX

```
[LAMBDA NIL                                             (* lmm "28-Jun-84 15:09")
  (COND
     ((OPT.CALLP (CAR CODE)
                 '\FLOATUNBOX 1)
      (COMP.DELFN))
     (T (COMP.STFN '\FLOATBOX 1])
```

(COMP.FLOATUNBOX

```
[LAMBDA NIL                                             (* lmm "28-Jun-84 15:08")
  (PROGN (COMP.DELFIX 'FLOAT)
         (COND
            ((OPT.CALLP (CAR CODE)
                        '\FLOATBOX 1)
             (COMP.DELFN))
            [[AND (EQ (fetch OPNAME of (CAR CODE))
                      'CONST)
                  (NUMBERP (fetch OPARG of (CAR CODE]
             (PROG [(NUM (fetch OPARG of (CAR CODE]
                   (COMP.DELPUSH)
                   (if (EQUAL (SETQ NUM (FLOAT NUM))
                              0)
                       then (COMP.STCONST NIL)
                       else (COMP.EXPR `(\VAG2 %, (fetch (FLOATP HIWORD) of NUM)
                                               %,
                                               (fetch (FLOATP LOWORD) of NUM]
            (T (COMP.STFN '\FLOATUNBOX 1])
```

(COMP.PREDP

```
[LAMBDA (CTX)                                           (* lmm "29-Jun-84 08:30")
```

```
    (AND (LISTP CTX)
         (FMEMB (CAR CTX)
                '(TJUMP FJUMP NTJUMP NFJUMP])
```

(COMP.UBFLOAT2

```
  [LAMBDA (A OP)                                          (* lmm "29-Jun-84 09:07")
    (PROG ((N 0))
          [COND
             ((AND (EQ COMPILE.CONTEXT 'EFFECT)
                   (NOT OPTIMIZATIONSOFF))
              (RETURN (COMP.PROGN A]
          (while A do (COMP.VAL (pop A))
                      (COMP.FLOATUNBOX)
                      (SETQ N (ADD1 N)))
          (FRPTQ (SUB1 N)
                 (COMP.STFN (LIST 'OPCODES 'UBFLOAT2 OP)
                            1))
          (COMP.FLOATBOX])
```

(COMP.UNBOX

```
  [LAMBDA (TYPE)                                          (* lmm "29-Dec-84 11:46")
    (PROG [(BOXER (AND (LITATOM TYPE)
                       (GETPROP TYPE 'BOX]
          (if BOXER
              then (COND
                      ((OPT.CALLP (CAR CODE)
                                  (CAR BOXER)
                                  1)                      (* top of stack was (box value)%, just get rid of BOX)
                       (COMP.DELFN))
                      ((EQ TYPE 'FLOAT)
                       (COMP.FLOATUNBOX))
                      (T (HELP)

          (* if top of stack is (convert-type value) then get rid of convert-type before putting in unbox)

                         (COMP.DELFIX TYPE)
                         (COMP.STFN (CDR BOXER)
                                    1)))
            else (HELP "CAN'T UNBOX" TYPE])

)

(ADDTOVAR COMPILETYPELST )

;; POST OPTIMIZATION

(DEFINEQ

## (OPT.POSTOPT

  [LAMBDA (CODE)                                          (* lmm "29-Dec-84 20:48")
    (COND
       [OPTIMIZATIONSOFF (while CODE bind C VAL do (SETQ TAGS NIL)
                                                   (while (EQ (fetch OPNAME of (SETQ C (pop CODE)))
                                                              'TAG)
                                                      do (push TAGS C))
                                                   (while (AND (EQ (fetch OPNAME of C)
                                                                   'JUMP)
                                                               (FMEMB (fetch OPARG of C)
                                                                      TAGS))
                                                      do (SETQ C (pop CODE)))
                                                   (for TAG in TAGS do (push VAL TAG))
                                                   (push VAL C)
                                 finally (RETURN (CDR VAL]
       (T (PROG ((FRAME TOPFRAME)
                 CL:LABELS ANY (FRAMES (LIST (LIST TOPFRAME)))
                 (PASS 1)
                 DELETEDBINDS)
                (SETQ CODE (CONS NIL (NCONC1 CODE NIL)))
                (OPT.SETUPOPT)
          OPTLP
                (SETQ ANY)                                (* optimization pass)
                (AND (OPT.FRAMEOPT (EQ PASS 1))
                     (SETQ ANY T))
                (OPT.SCANOPT)
                (OPT.JUMPOPT)
                (OPT.RETOPT)
                (OPT.CCHECK (OPT.OPTCHECK))
                [COND
                   ((NOT ANY)
                    (AND [NOT (OR (AND XVARFLG (PROGN (OPT.XVARSCAN)
                                                      (OPT.FRAMEOPT T NIL T)))
                                  (AND MERGEFRAMEFLG (OPT.FRAMEOPT T T XVARFLG]
                         (RETURN (CDR (OPT.DREV (CDR CODE]
                (SETQ PASS (ADD1 PASS))
                (BLOCK)
```

```
                (GO OPTLP])
```

(OPT.SETUPOPT

```
  [LAMBDA NIL                                                    (* lmm%: "22-JUL-77 02:59")
                                                                 (* set up code list as doubly linked list, scan for tags)
    (PROG ((C CODE)
           P B)
      LPC (COND
             ((NULL C)
              (RETURN)))
          (SELECTQ (fetch OPNAME of (CAR C))
              (TAG [COND
                      ((SETQ B (FASSOC (CAR C)
                                       CL:LABELS))
                       (FRPLACA (CDR B)
                              C))
                      (T (SETQ CL:LABELS (CONS (LIST (CAR C)
                                                     C)
                                               CL:LABELS])
              ((JUMP TJUMP FJUMP NTJUMP NFJUMP ERRORSET)
                  [COND
                     ((SETQ B (FASSOC (fetch (JUMP TAG) of (CAR C))
                                      CL:LABELS))
                      (NCONC1 B C))
                     (T (SETQ CL:LABELS (CONS (LIST (fetch (JUMP TAG) of (CAR C))
                                                    NIL C)
                                              CL:LABELS])
              NIL)
          (SELECTQ (fetch OPNAME of (CAR C))
              ((ERRORSET BIND)
                  [COND
                     ((SETQ B (FASSOC (CDR (fetch OPARG of (CAR C)))
                                      FRAMES))
                      (RPLACA (CDR B)
                             C))
                     (T (SETQ FRAMES (CONS (LIST (CDR (fetch OPARG of (CAR C)))
                                                 C)
                                           FRAMES])
              ((UNBIND DUNBIND)
                  [COND
                     ((SETQ B (FASSOC (CDR (fetch OPARG of (CAR C)))
                                      FRAMES))
                      (NCONC1 B C))
                     (T (SETQ FRAMES (CONS (LIST (CDR (fetch OPARG of (CAR C)))
                                                 NIL C)
                                           FRAMES])
              NIL)
          (SETQ B (CDR C))
          (replace PREV of C with B)
          (replace NXT of C with P)
          (SETQ P C)
          (SETQ C B)
          (GO LPC])
```

(OPT.SCANOPT

```
  [LAMBDA NIL                                                    (* lmm "29-Apr-85 19:26")
    (PROG ((CD CODE)
           A B P X Y)
      LP  (SETQ B (fetch PREV of CD))
          [AND P (OPT.CCHECK (EQ CD (fetch PREV of P]
          (SELECTQ (fetch OPNAME of (SETQ A (CAR CD)))
              (CONST (COND
                        ((AND (OPT.CALLP (CAR P)
                                         NIL 1)
                              (OR (FMEMB [SETQ X (CDR (fetch OPARG of (CAR P]
                                         CONSTFNS)
                                  (FMEMB X VCONDITIONALS)
                                  (FMEMB X CONDITIONALS)))       (* CONST FN.1 -> (FN CONST))
                         [RPLACA CD (create OP
                                            OPNAME _ 'CONST
                                            OPARG _ (APPLY* X (fetch OPARG of A]
                         (OPT.PRDEL P)
                         (GO BLP))
                        ([AND (SETQ A (FASSOC (fetch OPARG of A)
                                              CONST.FNS))
                              (SOME (CDR A)
                                    (FUNCTION (LAMBDA (X)
                                                (OPT.CALLP (CAR P)
                                                           (CAR (SETQ A (CDR X)))
                                                           (CAR X]   (* constant + fn -> otherfn)
                         (OPT.PRDEL CD)
                         (OPT.PRDEL P)
                         [MAPC (CDR A)
                               (FUNCTION (LAMBDA (X)
                                           (SETQ B (OPT.PRATTACH (create OP
```

```
                                                                OPNAME _ (CAR X)
                                                                OPARG _ (CDR X))
                                                        B]
                    (GO BLP)))
                (GO CHECKPUSH))
        (HVAR (GO CHECKPUSH))
        ((AVAR GVAR FVAR)
            (GO CHECKPUSH))
        (SETQ (COND
                  ((EQ (fetch OPARG of A)
                       (CAR B))                          (* want OPT.EQVALUE B CD execept OPT.EQVALUE takes
                                                         the wrong kind of arg)
                                                         (* var (setq var) => var)
                   (OPT.PRDEL CD)
                   (GO BLP))
                  ((OPT.DEADSETQP (fetch OPARG of A)
                          P)                             (* delete dead SETQ)
                   (OPT.PRDEL CD)
                   (GO BLP))))
        (POP (SELECTQ (fetch OPNAME of (CAR B))
                  ((AVAR HVAR FVAR GVAR COPY CONST)      (* push POP deleted)
                       (OPT.PRDEL B)
                       (OPT.PRDEL CD)
                       (SETQ B P)
                       (GO BLP))
                  (FN (COND
                         ((COMP.CLEANFNOP (CDR (fetch OPARG of (CAR B)))
                                 'NOSIDE)                (* cleanfn POP deleted)
                          (RPTQ (PROG1 (CAR (fetch OPARG of (CAR B)))
                                       (OPT.PRDEL B)
                                       (OPT.PRDEL CD)
                                       (SETQ B (fetch PREV of P)))
                                (SETQ B (OPT.PRATTACH OPPOP B)))
                          (GO BLP))))
                  (SETQ (COND
                           ([EQUAL (CAR (fetch PREV of B))
                                   (CONSTANT (create OP
                                                     OPNAME _ 'COPY]
                                                         (* COPY SETQ POP -> SETQ)
                            (OPT.PRDEL (fetch PREV of B))
                            (OPT.PRDEL CD)
                            (SETQ B P)
                            (GO BLP))))
                  NIL))
        (DUNBIND (COND
                    ((AND COMPILE.DUNBIND.POP.MERGE.FLG (EQ (CAR B)
                                                            OPPOP))
                                                         (* merge pop with DUNBIND)
                     (OPT.PRDEL B)                       (* (DUNBIND level . frame))
                     [RPLACA (fetch OPARG of (CAR CD))
                             (ADD1 (CAR (fetch OPARG of (CAR CD]
                     (GO ALP))))
        (UNBIND (COND
                   ((SELECTQ (fetch OPNAME of (CAR B))
                        (CONST                           (* CONST UNBIND)
                               (replace OPNAME of A with 'DUNBIND)
                                                         (* change to DUNBIND)
                                                         (* level is 1 less)
                               [RPLACA (fetch OPARG of A)
                                       (SUB1 (CAR (fetch OPARG of A])
                               (FN (COND
                                      ((AND (EQ (CAR (fetch OPARG of (CAR B)))
                                                1)
                                            (COMP.CLEANFNOP (CDR (fetch OPARG of (CAR B)))
                                                   'FREEVARS))   (* clean FN UNBIND)
                                       T)))
                               NIL)
                    (RPLACA CD (CAR B))
                    (RPLACA B A)                         (* switch CONST and DUNBIND)
                    (RPLACA (MEMB CD (CDDR (FASSOC (CDR (fetch OPARG of A))
                                                   FRAMES)))
                            B)
                    (GO BLP))))
        NIL)
TAG2
    (COND
       ((NULL B)
        (RETURN)))
    (SETQ P CD)
    (SETQ CD B)
    (GO LP)
BLP (SETQ CD B)
CLP (SETQ P (fetch NXT of CD))
ALP (SETQ ANY T)
    (GO LP)
CHECKPUSH
    (AND NEWOPTFLG (SELECTQ (fetch OPNAME of (CAR B))
```

```
                            (POP (COND
                                   ((OPT.EQVALUE (fetch PREV of B)
                                          CD)                    (* X POP X)
                                    (OPT.PRDEL CD)
                                    (OPT.PRDEL B)
                                    (SETQ CD (fetch PREV of P))
                                    (GO ALP))))
                            NIL))
          [COND
             (NEWOPTFLG (COND
                           ((SETQ X (OPT.JUMPCOPYTEST CD B))    (* can insert COPY at X and then delete CD)
                            (SETQ X (OPT.DELCOPYFN P X))
                            (SETQ P (fetch NXT of CD))
                            [COND
                               ((EQ X (fetch PREV of CD))
                                (OPT.PRDEL CD))
                               (T (FRPLACA CD '(SWAP]
                            (OPT.PRATTACH OPCOPY X)
                            (SETQ CD (fetch PREV of P))
                            (GO ALP)))
                    (COND
                       ((AND (SETQ X (OPT.SKIPPUSH B 1 CD T))
                             (SETQ X (OPT.JUMPCOPYTEST CD X)))
                        (SETQ X (OPT.DELCOPYFN P X))
                        (OPT.PRATTACH OPCOPY X)
                        (FRPLACA CD '(SWAP))
                        (GO ALP)))
                    (GO TAG2))
             (T (COND
                   ((OPT.EQVALUE B CD)                          (* val val -> val COPY)
                    (FRPLACA CD OPCOPY))
                   ((EQ (CAR B)
                        OPPOP)
                    (COND
                       ((OPT.EQVALUE (fetch PREV of B)
                             CD)                                 (* SETQ POP PUSH)
                        (OPT.PRDEL CD)
                        [OPT.PRDEL (PROG1 B
                                          (SETQ CD (fetch PREV of B)))]
                        (GO ALP]
          (GO TAG2])
```

(OPT.XVARSCAN

```
  [LAMBDA NIL                                                   (* rmk%: " 2-Apr-85 12:44")
    (PROG ((CD CODE)
           A)
          [for X in FRAMES do (replace NOXVAR of (CAR X) with (NEQ NIL (OASSOC 'AVAR (fetch VARS of (CAR X]
      LP  (SELECTQ (fetch OPNAME of (SETQ A (CAR CD)))
               (HVAR (AND (NOT (FMEMB A (FETCH VARS OF TOPFRAME)))
                          (OPT.XVARSCAN1 A CD)))
               (SETQ (SETQ A (fetch OPARG of A))
                     (COND
                        ((EQ (fetch OPNAME of A)
                             'HVAR)
                         (OPT.XVARSCAN1 A CD))))
               ((UNBIND DUNBIND)
                    (OR (OPT.CODELEV CD 0)
                        (replace NOXVAR of (CDR (fetch OPARG of A)) with T)))
               NIL)
          (COND
             ((NULL (SETQ CD (fetch PREV of CD)))
              (RETURN)))
          (GO LP])
```

(OPT.XVARSCAN1

```
  [LAMBDA (A CD)                                                (* rmk%: " 2-Apr-85 12:03")
    (PROG ((FR (OPT.CODEFRAME CD)))
          (OR FR (OPT.COMPILERERROR))
          (COND
             ((FMEMB A (fetch VARS of FR))
              (RETURN)))
      LP  (SETQ FR (fetch PARENT of FR))
          (COND
             ((FMEMB A (fetch VARS of FR))
              (replace NOXVAR of FR with T)
              (RETURN)))
          (COND
             ((EQ FR TOPFRAME)                                  (* can't find A)
              (OPT.COMPILERERROR)))
          (GO LP])
```

(OPT.JUMPOPT

```
  [LAMBDA NIL                                                   (* lmm "11-NOV-81 21:17")
    (MAPC CL:LABELS (FUNCTION (LAMBDA (X)
```

```
                              (COND
                                 ((CADR X)                          (* Label defined)
                                  (COND
                                     ((OR (OPT.JUMPTHRU (CAR X)
                                                (CDR X))
                                          (OPT.JUMPREV (CAR X)
                                                (CDR X)))
                                      (SETQ ANY T])
```

(OPT.JUMPTHRU

```
  [LAMBDA (TAG OPT.DEFREFS)                                         (* lmm "13-Jul-84 21:18")
    (PROG ((DR OPT.DEFREFS)
           P APD ALST ANY INFO Y REF BR END (DEF (CAR OPT.DEFREFS))
           PD B (FRAME (fetch (TAG FRAME) of TAG))
           (LEVEL (fetch (TAG LEVEL) of TAG)))
      LQ   (while [OR [type? TAG (SETQ APD (CAR (fetch PREV of DEF]
                      (type? TAG (SETQ APD (CAR (SETQ PD (fetch NXT of DEF]
             do                                                     (* two adjacent tags -
                                                                    merge them)
                 (OPT.LBMERGE TAG APD))
           [COND
              ((NULL (CDR DR))                                      (* tag which is not reference; delete it)
               (RETURN (OPT.LBDEL TAG]
           [COND
              [(EQ APD OPNIL)                                       (* instruction after the tag is NIL)
               (SETQQ ALST ((FJUMP NFJUMP . OPNIL]
              (T (SETQ ALST (SELECTQ (fetch OPNAME of APD)
                                 (JUMP '((JUMP)
                                         (TJUMP)
                                         (FJUMP)
                                         (NTJUMP)
                                         (NFJUMP)))
                                 (TJUMP '((NTJUMP TJUMP)
                                          (NFJUMP FJUMP . 1)))
                                 (FJUMP '((NTJUMP TJUMP . 1)
                                          (NFJUMP FJUMP)))
                                 (NTJUMP '((NTJUMP)
                                           (NFJUMP FJUMP . 1)))
                                 (NFJUMP '((NTJUMP TJUMP . 1)
                                           (NFJUMP)))
                                 (POP '((NTJUMP TJUMP . 1)
                                        (NFJUMP FJUMP . 1)
                                        (JUMP NIL . JP)))
                                 (RETURN '((JUMP NIL . R)))
                                 ((AVAR GVAR FVAR HVAR)
                                      '((FJUMP NFJUMP . L)
                                        (TJUMP NTJUMP . L)
                                        (JUMP NIL . LL)))
                                 (RETURN]
      LP   (COND
              ((NOT (SETQ INFO (FASSOC [fetch OPNAME of (CAR (SETQ REF (CADR DR]
                               ALST)))
               (GO NX)))
           (COND
              ((EQ REF PD)
               [COMPERRM (CONS COMFN '(-- infinite loop]
               (GO NX)))
           (SETQ BR (fetch PREV of REF))
           (SETQ Y (SELECTQ (CDDR INFO)
                        (NIL                                        (* JUMP to JUMP)
                             (fetch (JUMP TAG) of APD))
                        (R                                          (* JUMP to RETURN)
                           (FRPLACA REF OPRETURN)
                           NIL)
                        (L                                          (* VARIABLE REFERENCE)
                           (COND
                              ((OR (OPT.EQVALUE BR PD)
                                   (AND (EQ (fetch OPNAME of (CAR REF))
                                            'TJUMP)
                                        (OPT.CALLP (CAR BR)
                                               VCONDITIONALS 1)
                                        (OPT.EQVALUE (fetch PREV of BR)
                                               PD)))                (* VAR CJUMP to VAR)
                               (OPT.LABELNTHPR DEF 1 LEVEL 1))
                              [(SETQ Y (OPT.JUMPCOPYTEST PD BR))    (* VAR CJUMP |..| VAR -> VAR COPY CJUMP POP |..|
                                                                    VAR)
                               (PROG ((N 1)
                                      PDN)
                                     [COND
                                        (NEWOPTFLG (SETQ PDN (fetch NXT of PD))
                                               (while (AND (OPT.CALLP (CAR (SETQ INFO (fetch NXT of Y)))
                                                                NIL 1)
                                                           (COMP.CLEANFNOP (CDR (fetch OPARG of (CAR INFO)))
                                                                'NOSIDE)
                                                           (OPT.EQOP (CAR INFO)
                                                                (CAR PDN)))
```

```
                                        do (SETQ Y INFO)
                                           (SETQ PDN (fetch NXT of PDN))
                                           (add N 1]
                                    (OPT.PRATTACH OPCOPY Y)
                                    (OPT.PRATTACH OPPOP REF)
                                    (SETQ INFO)
                                    (RETURN (OPT.LABELNTHPR DEF N LEVEL 1]
                              (T (GO NX))))
                        (LL (COND
                               ((AND (EQ (CAR BR)
                                         OPPOP)
                                     (OPT.EQVALUE (fetch PREV of BR)
                                           PD))                    (* SETQ var POP JUMP to var)
                                (OPT.PRDEL BR)
                                (OPT.LABELNTHPR DEF 1 LEVEL 1))
                               (T (GO NX))))
                        (1                                         (* NTJUMP to POP)
                           (OPT.LABELNTHPR DEF 1 LEVEL -1))
                        (OPNIL                                     (* FJUMP to NIL)
                               (OPT.LABELNTHPR DEF 1 LEVEL 1))
                        (JP (COND
                               ((SETQ B (OPT.SKIPPUSH BR 1 NIL T))
                                                                   (* JUMP to POP)
                                [PROG NIL
                                  LPB (SETQ BR (PROG1 (fetch PREV of BR)
                                                      (OPT.PRDEL BR)))
                                      (COND
                                         ((NEQ BR B)
                                          (GO LPB]
                                (OPT.LABELNTHPR DEF 1 LEVEL -1))
                               (T (GO NX))))
                        (OPT.COMPILERERROR)))
                (COND
                   (Y (replace (JUMP TAG) of (CAR REF) with Y)
                      (NCONC1 (OPT.DEFREFS Y)
                              REF)))
                (SETQ ANY T)
```

(* Since the jump to this tag was redirected, delete the jump from the REFS for this tag)

```
                (FRPLACD DR (CDDR DR))
                [COND
                   ((CADR INFO)
                    (replace OPNAME of (CAR REF) with (CADR INFO]
                (GO LX)
             NX (SETQ DR (CDR DR))
             LX (COND
                   ((CDR DR)
                    (GO LP)))
                [COND
                   ((NULL (CDR OPT.DEFREFS))
                    (RETURN (OPT.LBDEL TAG]
                (RETURN ANY])
```

(OPT.LBMERGE

```
  [LAMBDA (TO FROM)                                    (* lmm%: "22-JUL-77 16:03")
    (PROG [(REFS (CDR (OPT.DEFREFS FROM]
          [MAPC REFS (FUNCTION (LAMBDA (X)
                                 (replace (JUMP TAG) of (CAR X) with TO]
          (NCONC (OPT.DEFREFS TO)
                 REFS)
          [OR (fetch (TAG LEVEL) of FROM)
              (PROGN (replace (TAG LEVEL) of TO with NIL)
                     (OR (fetch FRAME of FROM)
                         (replace FRAME of TO with NIL]
          (RETURN (OPT.LBDEL FROM])
```

(OPT.PRDEL

```
  [LAMBDA (X)                                          ; Edited 10-Jul-90 23:18 by jds
```

;; Remove X from the code stream by splicing it out of the doubly-linked list of code elements.

```
    (PROG ((B (fetch PREV of X))
           (P (fetch NXT of X)))
          (AND B (replace NXT of B with P))
          (AND P (replace PREV of P with B))
          (replace NXT of X with NIL])
```

(OPT.UBDEL

```
  [LAMBDA (CD)                                         (* lmm "14-MAR-81 09:16")
    (DREMOVE CD (OR (FASSOC (CDR (fetch OPARG of (CAR CD)))
                            FRAMES)
                    (OPT.COMPILERERROR])
```

(OPT.LBDEL

```
[LAMBDA (TAG)                                          ; Edited 10-Jul-90 23:19 by jds

    ;; Deleting a tag from the code stream.  Remove references to the tag.

    (PROG ((DEF (CAR (OPT.DEFREFS TAG)))
           B)
          (SETQ B (fetch PREV of DEF))
          (OPT.PRDEL DEF)
          (OPT.SETDEFREFS TAG NIL)
          (SETQ CL:LABELS (DREMOVE (FASSOC TAG CL:LABELS)
                                   CL:LABELS))
          [COND
             ((OPT.JUMPCHECK B)                        ; If there's a jump between this tag and any previous tags, delete
                                                       ; code before deleted tag
              (OPT.DELCODE (fetch NXT of B]
          (RETURN T])
```

(OPT.LABELNTHPR

```
[LAMBDA (CODE CNT LEVEL DL)                            (* lmm%: "22-JUL-77 16:12")
  (PROG ((CD CODE)
         G)
        (OPT.CHLEV DL)
    LP  (SETQ CD (fetch NXT of CD))
        (COND
           ((IGREATERP CNT 0)
            (OR (type? TAG (CAR CD))
                (SUB1VAR CNT))
            (GO LP))
           (T (RETURN (COND
                         ((type? TAG (CAR CD))
                          (OPT.CHECKTAG (CAR CD)
                                 T)
                          (CAR CD))
                         (T (PROG1 (SETQ G (create TAG))
                                   (replace (TAG FRAME) of G with FRAME)
                                   (SETQ CD (OPT.PRATTACH G (fetch PREV of CD)))
                                   (OPT.SETDEFREFS G (LIST CD))
                                   (replace (TAG LEVEL) of G with LEVEL))])
```

(OPT.JUMPREV

```
[LAMBDA (TAG OPT.DEFREFS)                              (* lmm "13-Jul-84 21:18")
                                                       (* OPT.JUMPREV checks the things that PRECEDE particular
                                                       kinds of jumps)
  (PROG ((DR OPT.DEFREFS)
         R
         (D (CAR OPT.DEFREFS))
         END ANY LB CD (LEVEL (fetch (TAG LEVEL) of TAG))
         (FRAME (fetch (TAG FRAME) of TAG))
         BD ABD FLG BR ABR OABR PR APD OAR TMP)
    LP  (SETQ R (CADR DR))
        (SETQ PR (fetch NXT of R))
        (SETQ BD (fetch PREV of D))
        (SETQ ABD (CAR BD))
        (SETQ BR (fetch PREV of R))
        (SETQ ABR (CAR BR))
        (SETQ OABR (fetch OPNAME of ABR))
        (SETQ OAR (fetch OPNAME of (CAR R)))

        (* variable code%: last letter is R for reference {i.e. place of jump}, D for definition {i.e.
        place where TAG is} -
        preceding letters%: -
        A for CAR -
        O for COP {op code} -
        P for CPR {next byte} -
        B for CBR {previous byte})

        (SELECTQ OAR
            (JUMP [COND
                      ((EQ R BD)                       (* JUMP to next location deleted)
                       (OPT.PRDEL R))
                      [(AND (OPT.EQOP ABD ABR)
                            (SETQ TMP (OPT.COMMONBACK BD R LEVEL)))

        (* OPT.COMMONBACK returns NIL if does nothing; T if deleted safe code or SAME if it deleted some code that contained a
        reference to the label that is now being worked on.)
                                                       (* merge similar code before JUMP and TAG)
                                                       (* IF SAME don't continue with this label! could have deleted
                                                       other references to it)
                       (COND
                          ((EQ TMP T)
                           (SETQ ANY T)
                           (GO LX))
                          (T (RETURN T]
                      [[AND (CAR PR)
                            (NOT (type? TAG (CAR PR]   (* delete code after JUMP)
```

```
                    (COND
                       ((OPT.DELCODE PR)                    (* returns T if it deleted any jumps
                                                            (may have deleted a jump for this tag))
                        (RETURN))
                       (T (GO NX]
                  ([AND (SELECTQ (fetch OPNAME of ABD)
                                 (RETURN T)
                                 (JUMP (NOT (FMEMB BD DR)))
                                NIL)
                        (SETQ END (fetch NXT of (OPT.FINDEND D R]
                                                            (* move jumped-to code in line)
                   (PROGN (replace NXT of BD with END)
                          (replace PREV of (PROG1 END
                                                  (SETQ END (fetch PREV of END)))
                             with BD))
                   (PROGN (replace NXT of BR with D)
                          (replace PREV of D with BR)
                          (replace PREV of PR with END)
                          (replace NXT of END with PR)))
                  (T (SELECTQ OABR
                              (CONST                        (* CONST JUMP)
                                     (SELECTQ (fetch OPNAME of APD)
                                              ((TJUMP NTJUMP)
                                                   (SETQ FLG (fetch OPARG of ABR)))
                                              ((FJUMP NFJUMP)
                                                   (SETQ FLG (NULL (fetch OPARG of ABR))))
                                              (GO NX))
                                     (NCONC1 [OPT.DEFREFS (replace (JUMP TAG) of (CAR R)
                                                                   with (COND
                                                                          (FLG (SELECTQ (fetch OPNAME of APD)
                                                                                        ((TJUMP FJUMP)
                                                                   (* T JUMP to TJUMP)
                                                                                            (OPT.PRDEL BR))

                                                                   (* T JUMP to NTJUMP))
                                                                                        (fetch (JUMP TAG) of APD))
                                                                          (T
                                                                   (* T JUMP to NF/FJUMP)
                                                                             (OPT.PRDEL BR)
                                                                             (OPT.LABELNTHPR D 1 LEVEL -1]
                                                      R))
                                              ((TJUMP FJUMP)
                                                   (COND
                                                      ((EQ (fetch (JUMP TAG) of (CAR R))
                                                           (fetch (JUMP TAG) of ABR))
                                                                   (* TJUMP->TAG JUMP->TAG => POP JUMP->TAG)
                                                       (OPT.PRDEL R)
                                                       (OPT.PRATTACH OPPOP (fetch PREV of BR))
                                                       (replace OPNAME of ABR with 'JUMP))
                                                      (T (GO NX))))
                                              (GO NX])
              ((FJUMP TJUMP)
                   (COND
                      ((EQ R BD)                            (* TJUMP to next location)
                       (FRPLACA R OPPOP))
                      [(EQ OABR 'CONST)
                       (COND
                          ((SELECTQ OAR
                                (TJUMP (fetch OPARG of ABR))
                                (NULL (fetch OPARG of ABR)))
                                                            (* T TJUMP -> JUMP)
                           (replace OPNAME of (CAR R) with 'JUMP)
                           (OPT.PRDEL BR)
                           (SETQ ANY T)                     (* try again)
                           (GO LP))
                          (T                                (* T FJUMP -> NOOP)
                             (OPT.PRDEL R)
                             (OPT.PRDEL BR]
                      ((OPT.CALLP ABR '(NOT NULL)
                                  1)                        (* NULL TJUMP)
                       (FRPLACA R (OPT.NOTJUMP (CAR R)))
                       (OPT.PRDEL BR)
                       (GO REDO))
                      ((AND (EQ ABR OPCOPY)
                            (EQ (CAR PR)
                                OPPOP))                     (* COPY TJUMP POP -> NTJUMP)
                       (OPT.PRDEL BR)
                       (OPT.PRDEL PR)
                       (replace OPNAME of (CAR R) with (SELECTQ OAR
                                                                (TJUMP 'NTJUMP)
                                                                'NFJUMP))
                       (GO REDO))
                      ((AND (EQ (fetch OPNAME of ABD)
                                'JUMP)
                            (EQ (fetch PREV of BD)
                                R))                         (* FJUMP.1 JUMP.2 1%: => TJUMP.2)
                       (replace OPNAME of ABD with (SELECTQ OAR
                                                            (TJUMP 'FJUMP)
```

```
                                                    'TJUMP))
                    (OPT.PRDEL R))
                   ((SETQ CD (OPT.JUMPCOPYTEST PR BR))      (* What is before the jump is also after -
                                                               e.g. X TJUMP X)
                    (COND
                       ((EQ (CAR PR)
                            (CAR (fetch NXT of D)))         (* X TJUMP.1 X |...| 1%:X |...| -> X COPY TJUMP.2 |...|
                                                               1%:X 2%: |...|)
                        (OPT.PRATTACH OPCOPY CD)
                        (SETQ LB (OPT.LABELNTHPR D 1 LEVEL 1)))
                       ((AND (OPT.JUMPCHECK (fetch PREV of D))
                             (OR (OPT.EQVALUE BR PR)
                                 (AND (EQ OAR 'FJUMP)
                                      (OPT.CALLP ABR VCONDITIONALS 1)
                                      (OPT.EQVALUE (fetch PREV of BR)
                                                   PR)))
                             (SETQ END (OPT.FINDEND D R)))  (* X FJUMP.1 X .a. 1%: .b. -> X NTJUMP.2 1%: .b.
                                                               |...| 2%: .a.)
                        (PROGN (replace NXT of (fetch PREV of D) with (fetch NXT of END))
                               (replace PREV of (fetch NXT of END) with (fetch PREV of D)))
                        (PROGN (replace NXT of R with D)
                               (replace PREV of D with R)
                               (replace PREV of PR with END)
                               (replace NXT of END with PR))
                        (replace OPNAME of (CAR R) with (SELECTQ OAR
                                                             (FJUMP 'NTJUMP)
                                                             'NFJUMP))
                        (SETQ LB (OPT.LABELNTHPR PR 0 LEVEL 1)))
                       (T (GO NX)))
                    (OPT.PRDEL PR)
                    (replace (JUMP TAG) of (CAR R) with LB)
                    (NCONC1 (OPT.DEFREFS LB)
                            R))
                   (T (GO NX))))
              ((NFJUMP NTJUMP)
                   (COND
                      [(EQ OABR 'CONST)
                       (COND
                          ((SELECTQ OAR
                                (NTJUMP (fetch OPARG of ABR))
                                (NULL (fetch OPARG of ABR)))  (* T NTJUMP -> JUMP)
                           (replace OPNAME of (CAR R) with 'JUMP)
                           (GO REDO))
                          (T                                  (* T NFJUMP -> NOOP)
                             (OPT.PRDEL BR)
                             (OPT.PRDEL R]
                      ((OPT.EQVALUE BR PR)                    (* X NTJUMP X -> X COPY TJUMP)
                       (OPT.PRATTACH OPCOPY (fetch PREV of R))
                       (OPT.PRDEL PR)
                       (replace OPNAME of (CAR R) with (SELECTQ OAR
                                                            (NTJUMP 'TJUMP)
                                                            'FJUMP))
                       (GO REDO))
                      [(EQ OAR 'NTJUMP)
                       (COND
                          [(NOT (OR (OPT.CALLP ABR CONDITIONALS)
                                    (OPT.CALLP ABR VCONDITIONALS)))
                           (COND
                              ((EQ (CAR (fetch NXT of R))
                                   OPNIL)                     (* NTJUMP NIL -> COPY TJUMP)
                               (OPT.PRDEL (fetch NXT of R))
                               (OPT.PRATTACH OPCOPY BR)
                               (replace OPNAME of (CAR R) with 'TJUMP)
                               (GO REDO))
                              (T (GO NX]
                          [(OPT.CALLP ABR VCONDITIONALS 1)
                           (COND
                              ((OPT.EQVALUE (fetch PREV of BR)
                                            PR)               (* X LISTP NTJUMP X -> X COPY LISTP TJUMP)
                               (OPT.PRATTACH OPCOPY (fetch PREV of BR))
                               (OPT.PRDEL PR)
                               (replace OPNAME of (CAR R) with 'TJUMP)
                               (GO REDO))
                              (T (GO NX]
                          (T (GO NX]
                       (T (GO NX))))
                   (GO NX))
               (SETQ ANY T)
               (FRPLACD DR (CDDR DR))
               (GO LX)
          NX   (SETQ DR (CDR DR))
          LX   (COND
                  ((CDR DR)
                   (GO LP)))
               (RETURN ANY)
          REDO
               (SETQ ANY T)
```

```
          (GO LP])
```

(OPT.COMMONBACK

```
[LAMBDA (BDEF REF LEVEL)                                    ; Edited 10-Jul-90 13:59 by jds
```

;; When the code preceding a jump is the same as the code preceding the label, can delete the code preceding the jump and move the label back
;; --- BDEF is the code preceding the label and REF is the jump and the code that precedes it

```
(PROG ((BREF (fetch PREV of REF))
       G FLG TMP (FRAME FRAME))
  M    (COND
          ((EQ (fetch OPNAME of (CAR BDEF))
               'TAG)
           (OPT.CHECKTAG (CAR BDEF)
                  LEVEL)
           (SETQ BDEF (fetch PREV of BDEF))
           (GO M)))
       (COND
          ((OPT.EQOP (CAR BDEF)
                  (CAR BREF))
           [SELECTQ (fetch OPNAME of (CAR BREF))
                ((AVAR HVAR GVAR FVAR CONST COPY)
                     (OPT.CHLEV -1))
                ((SETQ STORE SWAP RETURN))
                (POP (COND
                        ((AND [NOT (OPT.EQOP (CAR (fetch PREV of BREF))
                                             (CAR (fetch PREV of BDEF]
                              (EQ (fetch OPNAME of (CAR (fetch PREV of BREF)))
                                  'SETQ)
                              (EQ (fetch OPNAME of (CAR (fetch PREV of BDEF)))
                                  'SETQ))                   ; no OPT.COMMONBACK for different SETQ pop.
                         (GO EXIT)))
                     (OPT.CHLEV 1))
                ((TJUMP FJUMP NTJUMP NFJUMP)
                     (OPT.CHLEV 1)
                     [COND
                        ((EQ (fetch (JUMP TAG) of (CAR BREF))
                             (fetch (JUMP TAG) of (CAR REF)))
                         (SETQ FLG 'SAME]
                     (OPT.DELTAGREF BREF))
                (FN [OPT.CHLEV (SUB1 (CAR (fetch OPARG of (CAR BDEF])
                ((UNBIND DUNBIND)
                     (OPT.UBDEL BREF)
                     [SETQ LEVEL (CAR (fetch OPARG of (CAR BREF]
                     [SETQ FRAME (CDR (fetch OPARG of (CAR BREF])
                (OPT.COMPILERERROR '(OPT.COMMONBACK shouldn't get here]
           (OR FLG (SETQ FLG T))
           (SETQ BDEF (fetch PREV of BDEF))
           (SETQ BREF (PROG1 (fetch PREV of BREF)
                             (OPT.PRDEL BREF)))
           (GO M)))
  EXIT
       (COND
          (FLG (SETQ G (OPT.LABELNTHPR BDEF 0 LEVEL 0))
               (OPT.DELTAGREF REF)
               (replace (JUMP TAG) of (CAR REF) with G)
               (NCONC1 (OPT.DEFREFS G)
                       REF)
               (RETURN FLG])
```

(OPT.DELTAGREF

```
[LAMBDA (REF)                                               ; Edited 10-Jul-90 23:01 by jds
```

;; Delete a reference to a jumnp-target tag. If the tag has no references, remove it from the list LABELS, so we don't try to optimize the code
;; around it.

```
(LET [(TAG (fetch (JUMP TAG) of (CAR REF]
     (for X on (OPT.DEFREFS TAG) when (EQ (CADR X)
                                          REF)
        do (RETURN (RPLACD X (CDDR X))) finally (OPT.COMPILERERROR))
     (COND
        ((NOT (OPT.DEFREFS TAG))
```

;; No remaining refs to this tag. Remove it from LABELS, so we don't try to do jump optimization with respect to it.

```
         (SETQ CL:LABELS (DREMOVE (FASSOC TAG CL:LABELS)
                                  CL:LABELS])
```

(OPT.FINDEND

```
[LAMBDA (C STOP)                                            (* lmm%: "22-JUL-77 03:38")
  (PROG NIL
    LP  (COND
           ((EQ C STOP)
            (RETURN)))
        (COND
           ((OPT.JUMPCHECK C)
            (RETURN C)))
```

```
          (COND
             ((SETQ C (fetch NXT of C))
              (GO LP])
```

(OPT.RETOPT

```
  [LAMBDA NIL                                                    (* DD%: "21-FEB-83 17:17")
                                                                 (* optimizations involving RETURN)
    (PROG ((RL (OPT.RETFIND CODE))
           TESTL TARGL)
          [MAPC RL (FUNCTION (LAMBDA (C)
                             (COND
                                ((OPT.RETPOP C)
                                 (SETQ ANY T)))
                             (COND
                                ((OPT.RETTEST C C)               (* Test if C is a possible test.)

          (* Looking for the case where two identical sequences ending with RETURN one of which is preceded by a conditional
          jump; -
          TJUMP->x stuff RETURN x%: |...| stuff RETURN |...| becomes -
          FJUMP->y x%: |...| y%: stuff RETURN)

                                 (SETQ TESTL (CONS C TESTL)))
                                (T (SETQ TARGL (CONS C TARGL]
          (OR TESTL (RETURN ANY))
          [SETQ TESTL (SUBSET TESTL (FUNCTION (LAMBDA (X)
                                                (NOT (OPT.RETOPT1 X TARGL]
          [MAP TESTL (FUNCTION (LAMBDA (Z)
                               (AND (LISTP Z)
                                    (OPT.RETOPT1 (CAR Z)
                                           (CDR Z]
          (RETURN ANY])
```

(OPT.RETFIND

```
  [LAMBDA (C)                                                    (* lmm%: "18-AUG-76 02:12:31")
                                                                 (* returns the list of all RETURN's in the code)
    (PROG ((L1 C)
           R)
      LP   (COND
              ((SETQ L1 (FMEMB OPRETURN (CDR L1)))
               (SETQ R (CONS L1 R))
               (GO LP)))
          (RETURN R])
```

(OPT.RETPOP

```
  [LAMBDA (RET)                                                  (* rmk%: " 2-Apr-85 12:46")
                                                                 (* can delete any UNBIND's preceding a RETURN -
                                                                 the RETURN does it automatically)
    (PROG (ANY TAGS VAL)
      LP   (SELECTQ [fetch OPNAME of (CAR (SETQ RET (fetch PREV of RET]
                (UNBIND (SELECTQ (fetch OPNAME of VAL)
                                 ((AVAR HVAR)                    (* don't delete UNBIND when followed by VAR RETURN)
                                     )
                                 (PROGN                          (* delete UNBIND before RETURN)
                                        (OPT.UBDEL RET)
                                        (GO DEL))))
                (POP (COND
                        (VAL                                     (* delete POP before VAR RETURN)
                             (GO DEL))))
                (DUNBIND (COND
                            (VAL                                 (* delete DUNBIND before VAR RETURN)
                                 (OPT.UBDEL RET)
                                 (GO DEL))))
                (COPY (COND
                         ((NOT (fetch OPARG of (CAR RET)))       (* delete COPY before RETURN)
                          (GO DEL))))
                ((AVAR HVAR FVAR GVAR CONST)
                     (COND
                        ((NULL VAL)
                         (SETQ VAL (CAR RET))
                         (GO LP))
                        (T                                       (* VAR VAR RETURN)
                           (GO DEL))))
                (TAG (if [AND XVARFLG (SELECTQ (fetch OPNAME of VAL)
                                               (CONST NIL)
                                               (NOT (FMEMB VAL (fetch VARS of TOPFRAME]
                         then                                    (* if have XVARs then TAGs can't be ambiguous)
                       else (SETQ TAGS (CONS (CAR RET)
                                             TAGS))
                            (GO LP)))
                NIL)
          (RETURN ANY)
        DEL (OPT.PRDEL RET)
        DOIT
          (SETQ ANY T)
```

```
          [MAPC TAGS (FUNCTION (LAMBDA (X)
                                 (replace (TAG LEVEL) of X with NIL]
          (SETQ TAGS)
          (GO LP])
```

(OPT.RETOPT1

```
  [LAMBDA (X L)                                                    (* lmm%: "13-OCT-76 18:45:46")
    (PROG (END Y1)
          (RETURN (COND
                     ([SETQ Y1 (SOME L (FUNCTION (LAMBDA (Y)
                                                   (SETQ END (OPT.RETTEST X Y]
                      (OPT.RETMERGE X END (CAR Y1))
                      (SETQ ANY T])
```

(OPT.RETTEST

```
  [LAMBDA (TEST TARGET)                                            (* jds "ANOTHER FAKE DATE")
    (PROG ((L1 TEST)
           (L2 TARGET)
           F1 F2 ONLYIFSAMEFRAME)
          [COND
             ((EQ L1 L2)
              (SETQ F1 (SETQ F2 T]
      LP  (SETQ L1 (fetch PREV of L1))
          (SETQ L2 (fetch PREV of L2))
      L1  (COND
             ((type? TAG (CAR L1))
              [OR F1 (SETQ F1 (fetch (TAG FRAME) of (CAR L1]
              (SETQ L1 (fetch PREV of L1))
              (GO L1)))
      L2  (COND
             ((type? TAG (CAR L2))
              [OR F2 (SETQ F2 (fetch (TAG FRAME) of (CAR L2]
              (SETQ L2 (fetch PREV of L2))
              (GO L2)))
          (SELECTQ (fetch OPNAME of (CAR L1))
               (RETURN (GO RET))
               (JUMP (GO RETJ))
               ((FJUMP TJUMP)
                    (COND
                       ((EQ (fetch (JUMP TAG) of (CAR L1))
                            (CAR (fetch NXT of TEST)))
                        (GO RETJ))))
               (AVAR (COND
                        ((EQ (CAR L1)
                             (CAR L2))
                         (SETQ ONLYIFSAMEFRAME T)
                         (GO LP))))
               (HVAR [COND
                        ((EQ (CAR L1)
                             (CAR L2))
                         (COND
                            ((EQ (OPT.CODEFRAME L1)
                                 (OPT.CODEFRAME L2))
                             (COND
                                ((EQ (OPT.CODELEV L1 0)
                                     (OPT.CODELEV L2 0))
                                 (GO LP)
```

(* if NOXVAR would work, we could do this. Unfortunately, NOXVAR is ignored at this point
(replace (FRAME NOXVAR) of (OPT.CODEFRAME L1) with T))

```
                                ])
               ((UNBIND DUNBIND)
                    (COND
                       ([AND [EQ [CAR (LISTP (fetch OPARG of (LISTP (CAR L1]
                                 (CAR (LISTP (fetch OPARG of (LISTP (CAR L2]
                             (EQ [CDR (fetch OPARG of (LISTP (CAR L1]
                                 (CDR (fetch OPARG of (LISTP (CAR L2]
                        (SETQ F1 (SETQ F2 T))                      (* same frame)
                        (GO LP))))
               (FN (COND
                      ((OPT.EQOP (CAR L1)
                                 (CAR L2))
                       (GO LP))))
               (BIND                                               (* don't merge binds)
                     NIL)
               ((POP CONST FVAR GVAR SWAP)
                    (COND
                       ((EQ (CAR L1)
                            (CAR L2))
                        (GO LP))))
               ((STORE COPY)
                    (COND
                       ((EQUAL (CAR L1)
                               (CAR L2))
```

```
                        (GO LP))))
              NIL)
         (RETURN)
     RETJ
         [OR F1 (SETQ F1 (fetch (TAG FRAME) of (fetch (JUMP TAG) of (CAR L1]
     RET [COND
            (ONLYIFSAMEFRAME (COND
                                ((NEQ (OR F1 (OPT.CODEFRAME L1))
                                      (OR F2 (OPT.CODEFRAME L2)))
                                                                  (* OPT.RETTEST fail because not same frame)
                                 (RETURN]
         (RETURN L1])
```

(OPT.RETMERGE

```
  [LAMBDA (TEST END TARGET)                                       (* lmm "13-OCT-78 21:25")
    (PROG ((L1 TEST)
           (L2 TARGET)
           G VEQ FEQ LEV)
          [COND
             ([AND (SETQ LEV (OPT.CODEFRAME (fetch PREV of TEST)))
                   (EQ LEV (OPT.CODEFRAME (fetch PREV of TARGET]
              (SETQ FEQ T)
              (COND
                 ((AND (SETQ LEV (OPT.CODELEV (fetch PREV of TEST)
                                              0))
                       (EQ LEV (OPT.CODELEV (fetch PREV of TARGET)
                                            0)))
                  (SETQ VEQ T]
     LP   (COND
             ((EQ L1 END)
              (SELECTQ (fetch OPNAME of (CAR L1))
                   ((TJUMP FJUMP)
                        [COND
                           [[NOT (type? TAG (SETQ G (CAR L2]
                            (SETQ G (create TAG))
                            [COND
                               (FEQ [replace (TAG FRAME) of G with (fetch (TAG FRAME) of (fetch (JUMP TAG)
                                                                                                of (CAR L1]
                                    (COND
                                       (VEQ (replace (TAG LEVEL) of G with (fetch (TAG LEVEL)
                                                                                  of (fetch (JUMP TAG)
                                                                                            of (CAR L1]
                           (OPT.SETDEFREFS G (LIST (OPT.PRATTACH G L2]
                          (T (OR VEQ (replace (TAG LEVEL) of G with NIL))
                             (OR FEQ (replace (TAG FRAME) of G with NIL]
                        (FRPLACA L1 (OPT.NOTJUMP (CAR L1)))
                        [DREMOVE L1 (OPT.DEFREFS (fetch (JUMP TAG) of (CAR L1]
                        (replace (JUMP TAG) of (CAR L1) with G)
                        (NCONC1 (OPT.DEFREFS G)
                                L1))
                   ((JUMP RETURN))
                   (OPT.COMPILERERROR))
              (RETURN)))
          (COND
             ((type? TAG (CAR L1))
              (OR VEQ (replace (TAG LEVEL) of (CAR L1) with NIL))
              (OR FEQ (replace (TAG FRAME) of (CAR L1) with NIL))
              (RPLACA (OPT.DEFREFS (CAR L1))
                      (OPT.PRATTACH (CAR L1)
                                    L2))
              (SETQ L1 (PROG1 (fetch PREV of L1)
                              (OPT.PRDEL L1)))
              (GO LP)))
     L2   (COND
             ((type? TAG (CAR L2))
              (OR VEQ (replace (TAG LEVEL) of (CAR L2) with NIL))
              (OR FEQ (replace (TAG FRAME) of (CAR L2) with NIL))
              (SETQ L2 (fetch PREV of L2))
              (GO L2)))
          (SELECTQ (fetch OPNAME of (CAR L1))
              ((UNBIND DUNBIND)
                   (OPT.UBDEL L1))
              ((TJUMP NTJUMP FJUMP NFJUMP JUMP BIND ERRORSET)
                   (OPT.COMPILERERROR))
              NIL)
          (SETQ L1 (PROG1 (fetch PREV of L1)
                          (OPT.PRDEL L1)))
          (SETQ L2 (fetch PREV of L2))
          (GO LP])
```

(OPT.CODELEV

```
  [LAMBDA (CD LEV)                                                (* jds "THIS IS A FAKE DATE")
    (PROG NIL
          (RETURN (IPLUS (SELECTQ (fetch OPNAME of (CAR CD))
                                  (TAG (OR (fetch (TAG LEVEL) of (CAR CD))
```

```
                                    (RETURN)))
                               ((NTJUMP NFJUMP TJUMP FJUMP)
                                    (RETURN (OPT.CODELEV (fetch PREV of CD)
                                                   (SUB1 LEV))))
                               ((AVAR HVAR COPY CONST FVAR GVAR)
                                    (RETURN (OPT.CODELEV (fetch PREV of CD)
                                                   (ADD1 LEV))))
                               (FN [RETURN (OPT.CODELEV (fetch PREV of CD)
                                                  (ADD1 (IDIFFERENCE LEV (CAR (fetch OPARG of (CAR CD])
                               (POP (RETURN (OPT.CODELEV (fetch PREV of CD)
                                                   (SUB1 LEV))))
                               ((BIND ERRORSET)
                                   0)
                               (DUNBIND [fetch (FRAME LEVEL) of (CDR (fetch OPARG of (CAR CD])
                               (UNBIND (ADD1 (OR [fetch (FRAME LEVEL) of (CDR (fetch OPARG of (CAR CD]
                                                 (RETURN))))
                               ((SETQ STORE SWAP)
                                    (RETURN (OPT.CODELEV (fetch PREV of CD)
                                                   LEV)))
                               (NIL (OPT.CCHECK (NOT (CDR CD)))
                                    0)
                               (OPT.COMPILERERROR (CAR CD)))
                          LEV])
```

(OPT.CODEFRAME

```
  [LAMBDA (CD)                                                    (* rmk%: " 2-Apr-85 12:47")
    (SELECTQ (fetch OPNAME of (CAR CD))
         (TAG (OR (fetch (TAG FRAME) of (CAR CD))
                  (OPT.CODEFRAME (fetch PREV of CD))))
         ((NTJUMP NFJUMP TJUMP FJUMP)

           (* can't assume that code of jumped-to is same, because return-merging might have messed it up)

             (OPT.CODEFRAME (fetch PREV of CD)))
         ((BIND ERRORSET)
             (CDR (fetch OPARG of (CAR CD))))
         ((UNBIND DUNBIND)
             [fetch PARENT of (CDR (fetch OPARG of (CAR CD])
         (NIL TOPFRAME)
         ((JUMP RETURN)
            NIL)
         (OPT.CODEFRAME (fetch PREV of CD])
```

(OPT.DEFREFS

```
  [LAMBDA (D)                                                     ; Edited 10-Jul-90 23:02 by jds
    ;; Given a jump-target tag, return a list of the references to that tag.
    (CDR (FASSOC D CL:LABELS])
```

(OPT.SETDEFREFS

```
  [LAMBDA (D V)                                                   (* lmm%: "22-JUL-77 15:58")
    (FRPLACD [OR (FASSOC D CL:LABELS)
                 (CAR (SETQ CL:LABELS (CONS (CONS D)
                                            CL:LABELS]
           V])

)

(DEFINEQ
```

(OPT.FRAMEOPT

```
  [LAMBDA (TRYLOCAL TRYMERGE TRYXVAR)                             (* lmm "16-DEC-81 17:05")
    (PROG (ANY)
          [COND
             (TRYLOCAL (MAPC FRAMES (FUNCTION (LAMBDA (X)
                                          (AND (OPT.FRAMELOCAL (CAR X))
                                               (SETQ ANY T]
          [MAPC FRAMES (FUNCTION (LAMBDA (F)
                                   (AND (CADR F)
                                        (OPT.FRAMEVAR F)
                                        (SETQ ANY T]
          [COND
             (TRYMERGE (MAPC FRAMES (FUNCTION (LAMBDA (F)
                                          (AND (CADR F)
                                               (OPT.FRAMEMERGE F)
                                               (SETQ ANY T]
          [SETQ FRAMES (SUBSET FRAMES (FUNCTION (LAMBDA (F)
                                          (NOT (AND (CADR F)
                                                    (OPT.FRAMEDEL F TRYXVAR)
                                                    (SETQ ANY T]
          (RETURN ANY])
```

(OPT.FRAMEMERGE

```
[LAMBDA (F)                                                    (* lmm "29-Dec-84 10:35")
  (AND MERGEFRAMEFLG (PROG ((FR (CAR F))
                            VAR VARS P)
                           (COND
                              ((AND (SETQ VARS (fetch VARS of FR))
                                    (NULL (CDR (FNTH VARS MERGEFRAMEMAX)))
                                    (SETQ P (fetch PARENT of FR))
                                    (OPT.MERGEFRAMEP FR P VARS))
                               [PROG ((N (fetch NVALS of FR))
                                      (V VARS)
                                      (CD (fetch PREV of (CADR F)))
                                      P2)
                                 PLP (COND
                                        ((AND (SETQ P2 (fetch PARENT of P))
                                              (OPT.MERGEFRAMEP FR P2 VARS))
                                         (SETQ P P2)
                                         (GO PLP)))
                                     (replace VARS of P with (NCONC (fetch VARS of P)
                                                                    VARS))
                                     (replace VARS of FR with NIL)
                                     (replace NNILS of P with (IPLUS (fetch NNILS of P)
                                                                     (fetch NNILS of FR)
                                                                     (fetch NVALS of FR)))
                                     (replace NNILS of FR with (replace NVALS of FR with 0))
                                  LP  (COND
                                         (V (SETQ VAR (create OP
                                                              OPNAME _ 'SETQ
                                                              OPARG _ (CAR V)))
                                            [COND
                                               ((IGREATERP N 0)
                                                (OPT.PRATTACH OPPOP (OPT.PRATTACH VAR CD)))
                                               (T [COND
                                                     ((ZEROP N)
                                                      (SETQ CD (OPT.PRATTACH OPNIL CD]
                                                   (OR (OPT.NONILVAR (CAR V)
                                                                     CD P)
                                                       (SETQ CD (OPT.PRATTACH VAR CD]
                                            (SETQ N (SUB1 N))
                                            (SETQ V (CDR V))
                                            (GO LP)))
                                     (COND
                                        ((MINUSP N)
                                         (OPT.PRATTACH OPPOP CD]
                               (RETURN T])
```

(OPT.NONILVAR

```
[LAMBDA (V CD FR)                                              (* lmm " 8-JAN-82 09:06")
                                                               (* used by OPT.FRAMEMERGE)
  (PROG NIL
        (RETURN (AND (SELECTQ (fetch OPNAME of (CAR CD))
                              ((CONST POP COPY AVAR HVAR FVAR GVAR TJUMP FJUMP NTJUMP NFJUMP SETQ STORE SWAP)
                                  T)
                              (NIL NIL)
                              (FN (COMP.CLEANFNOP (CDR (fetch OPARG of (CAR CD)))
                                                  'FREEVARS))
                              (BIND (COND
                                       ([EQ FR (CDR (fetch OPARG of (CAR CD]
                                        (RETURN T))
                                       (T T)))
                              ((TAG RETURN)
                                  NIL)
                              ((UNBIND DUNBIND ERRORSET)
                                  T)
                             NIL)
                     (OPT.NONILVAR V (CDR CD)
                            FR])
```

(OPT.MERGEFRAMEP

```
[LAMBDA (FR PARENT VARS)                                       (* lmm "29-Dec-84 10:31")
  (AND (FMEMB (fetch FRAMETYPE of PARENT)
              MERGEFRAMETYPES)
       (COND
          [(OASSOC 'AVAR VARS)
           (AND (OPT.CLEANFRAME PARENT FR)
                (PROG NIL
                      [for V in VARS do (if (FMEMB (fetch OPARG of V)
                                                   SYSSPECVARS)
                                            then (GO BAD))
                                        [for F in FRAMES when (NEQ (CAR F)
                                                                   FR)
                                           do (for V2 in (fetch VARS of (CAR F))
                                                 do (COND
                                                       ((EQ (fetch OPARG of V2)
                                                            (fetch OPARG of V))
                                                        (GO BAD]
```

```
                                        (for V2 in FREEVARS do (COND
                                                                  ((EQ (fetch OPARG of V2)
                                                                       (fetch OPARG of V))
                                                                   (GO BAD]
                        (RETURN T)
                    BAD (RETURN]
              (T (EQ MERGEFRAMEFLG T])
```

(OPT.FRAMELOCAL

```
[LAMBDA (F)                                                    (* lmm "29-Dec-84 20:45")
  (PROG (VARS ANY)
        (COND
           ((AND (OASSOC 'AVAR (SETQ VARS (fetch (FRAME VARS) of F)))
                 (OPT.CLEANFRAME F))                           (* make vars local when no external calls)
            (for X in VARS when (AND (EQ (fetch OPNAME of X)
                                         'AVAR)
                                     (NOT (FMEMB (fetch OPARG of X)
                                                 SYSSPECVARS)))
               do (replace OPNAME of X with 'HVAR)
                  (SETQ ANY T))
            (RETURN ANY])
```

(OPT.CLEANFRAME

```
[LAMBDA (FRAME AVOIDING)                                       (* lmm%: " 9-NOV-76 16:20:20")
  (AND (NOT (fetch EXTCALL of FRAME))
       (for F in FRAMES when (AND (EQ (fetch PARENT of (CAR F))
                                      FRAME)
                                  (NEQ (CAR F)
                                       AVOIDING))
          always (OPT.CLEANFRAME (CAR F)
                                 AVOIDING])
```

(OPT.FRAMEDEL

```
[LAMBDA (F TRYXVAR)                                            (* lmm "13-Jul-84 21:18")
  (PROG (VARS (FRM (CAR F))
              PARENT OP FLV TMP DOXVAR)
        (SELECTQ (fetch FRAMETYPE of FRM)
            ((NIL ERRORSET)
                (RETURN))
            NIL)
        (SETQ VARS (fetch VARS of FRM))
        (SETQ FLV (fetch (FRAME LEVEL) of FRM))
        (SETQ DOXVAR NIL)
        (COND
           ([AND [NOT (SOME (CDDR F)
                            (FUNCTION (LAMBDA (X)
                                        (AND (EQ (fetch OPNAME of (CAR X))
                                                 'UNBIND)
                                             (IGREATERP (CAR (fetch OPARG of (CAR X)))
                                                        1]
                 (OR (NULL VARS)
                     (AND (NOT (OASSOC 'AVAR VARS))
                          (OR (OPT.DELETEFRAMECHECK VARS F)
                              (AND TRYXVAR (NOT (fetch NOXVAR of FRM))
                                   (SETQ DOXVAR T]            (* frame with no specvars, no UNBIND's with LEVEL gt 1)
            (OR (SETQ PARENT (fetch PARENT of FRM))
                (OPT.COMPILERERROR))
            [COND
               (DOXVAR (add FLV (fetch NNILS of FRM)
                            (fetch NVALS of FRM]
            [for VR on VARS
               do (for CD on CODE do (COND
                                       [(AND (EQ (fetch OPARG of (CAR CD))
                                                 (CAR VR))
                                             (EQ (fetch OPNAME of (CAR CD))
                                                 'SETQ))
                                        (COND
                                           [DOXVAR (OPT.CCHECK (EQ FRM (OPT.CODEFRAME CD)))
                                                   (RPLACA CD (create OP
                                                                      OPNAME _ 'STORE
                                                                      OPARG _ (OR (OPT.CODELEV
                                                                                    CD
                                                                                    (LENGTH (CDR VR)))
                                                                                  (OPT.COMPILERERROR]
                                           (T (OPT.PRDEL CD)  (* delete SETQ in OPT.FRAMEDEL)
                                              ]
                                        ((AND DOXVAR (EQ (CAR CD)
                                                         (CAR VR)))
                                         (OPT.CCHECK (EQ (OPT.CODEFRAME CD)
                                                         FRM))
                                         (RPLACA CD (COND
                                                      ([ZEROP (SETQ TMP (OPT.CODELEV (fetch PREV of CD)
                                                                                     (LENGTH (CDR VR]
                                                       OPCOPY)
```

```
                                                    (T (create OP
                                                               OPNAME _ 'COPY
                                                               OPARG _ TMP]
            [MAPC CL:LABELS (FUNCTION (LAMBDA (X)
                                      (COND
                                         ((EQ (fetch (TAG FRAME) of (CAR X))
                                              FRM)
                                          (replace (TAG FRAME) of (CAR X) with PARENT)
                                          (AND (fetch (TAG LEVEL) of (CAR X))
                                               FLV
                                               (replace (TAG LEVEL) of (CAR X)
                                                  with (IPLUS (fetch (TAG LEVEL) of (CAR X))
                                                              FLV]
            [PROG ((CD (CADR F)))                                  (* delete the bind and all of the var references after)
                  [MAPC (CONS NIL (AND (NOT DOXVAR)
                                       VARS))
                        (FUNCTION (LAMBDA NIL
                                    (SETQ CD (PROG1 (fetch NXT of CD)
                                                    (OPT.PRDEL CD]
                  (FRPTQ (fetch NNILS of FRM)
                         (OPT.PRATTACH OPNIL (fetch PREV of CD]
            (COND
               ((fetch EXTCALL of FRM)
                (replace EXTCALL of PARENT with T)))
            [MAPC (CDDR F)
                  (FUNCTION (LAMBDA (CD)                           (* change DUNBIND to POP of LEVEL)
                              (SELECTQ (PROG1 (fetch OPNAME of (SETQ OP (CAR CD)))
                                              (SETQ CD (PROG1 (fetch PREV of CD)
                                                              (OPT.PRDEL CD))))
                                       (UNBIND [COND
                                                  [DOXVAR (COND
                                                             ([NOT (ZEROP (SETQ TMP
                                                                                (IPLUS (CAR (fetch OPARG of OP))
                                                                                       (LENGTH VARS)
                                                                                       -1]
                                                              (SETQ CD (OPT.PRATTACH (create OP
                                                                                             OPNAME _ 'STORE
                                                                                             OPARG _ TMP)
                                                                                     CD))
                                                              (FRPTQ TMP (OPT.PRATTACH OPPOP CD]
                                                  (T (OPT.CCHECK (EQ (CAR (fetch OPARG of OP))
                                                                     1])
                                       (DUNBIND (FRPTQ [COND
                                                          (DOXVAR (IPLUS (CAR (fetch OPARG of OP))
                                                                         (fetch NVALS of FRM)
                                                                         (fetch NNILS of FRM)))
                                                          (T (CAR (fetch OPARG of OP]
                                                       (OPT.PRATTACH OPPOP CD)))
                                       (OPT.COMPILERERROR]
            [MAPC FRAMES (FUNCTION (LAMBDA (F2)
                                     (COND
                                        ((EQ (fetch PARENT of (CAR F2))
                                             FRM)
                                         (replace PARENT of (CAR F2) with PARENT)
                                         (replace (FRAME LEVEL) of (CAR F2) with (AND FLV (SETQ TMP
                                                                                                  (fetch (FRAME LEVEL)
                                                                                                     of (CAR F2)))
                                                                                          (IPLUS TMP FLV]
            (RETURN T])
```

(OPT.FRAMEVAR

```
  [LAMBDA (F)                                                      (* lmm "13-Jul-84 21:18")
    (PROG (VARS CD (FR (CAR F))
                VAL ANY NNILS NVALS)
          [SETQ VARS (REVERSE (OR (fetch VARS of FR)
                                  (RETURN]
          (SETQ NNILS (fetch NNILS of FR))
          (SETQ NVALS (fetch NVALS of FR))
          [for V on VARS as I from NNILS to 0 by -1 when (NEQ (fetch OPNAME of (CAR V))
                                                              'AVAR)
             do (COND
                   ((NOT (SETQ CD (FMEMB (CAR V)
                                         CODE)))
                    [COND
                       ((ZEROP I)
                        (SETQ I 1)
                        (OPT.PRATTACH OPPOP (fetch PREV of (CADR F)))
                        (SETQ NVALS (SUB1 NVALS)))
                       (T (SETQ NNILS (SUB1 NNILS]                 (* local var bound but not used)
                    (PROG ((CD CODE))
                      LP   (COND
                              ((NOT CD)
                               (RETURN)))                          (* delete all SETQ's)
                           (COND
                              ((AND (EQ (fetch OPARG of (CAR CD))
                                        (CAR V))
```

```
                                (EQ (fetch OPNAME of (CAR CD))
                                    'SETQ))                    (* local var set but never used)
                            (OPT.PRDEL CD)))
                       (SETQ CD (fetch PREV of CD))
                       (GO LP))
                 (RPLACA V NIL)
                 (SETQ ANY T))
                ([NOTANY CODE (FUNCTION (LAMBDA (X)
                                          (AND (EQ (fetch OPNAME of X)
                                                   'SETQ)
                                               (EQ (fetch OPARG of X)
                                                   (CAR V]
                 (COND
                    ([SETQ VAL (COND
                                  ((NEQ I 0)                   (* NIL var never set)
                                   (SETQ NNILS (SUB1 NNILS))
                                   OPNIL)
                                  ((AND (EQ [fetch OPNAME of (SETQ VAL (CAR (fetch PREV of (CADR F]
                                            'CONST)
                                        (APPLY* EQCONSTFN (fetch OPARG of VAL)))
                                   (SETQ I 1)
                                   (SETQ NVALS (SUB1 NVALS))  (* delete this var, can try next)
                                                              (* var bound to CONST and never set)
                                   (PROG1 (CAR (fetch PREV of (CADR F)))
                                          (OPT.PRDEL (fetch PREV of (CADR F))))]
                       (do (FRPLACA CD VAL) repeatwhile (SETQ CD (FMEMB (CAR V)
                                                                        CD)))
                       (FRPLACA V NIL)
                       (SETQ ANY T]
          (COND
             (ANY [replace VARS of FR with (OPT.DREV (SUBSET VARS (FUNCTION (LAMBDA (X)
                                                                              X]
                  (replace NNILS of FR with NNILS)
                  (replace NVALS of FR with NVALS)))
          (RETURN ANY])
```

(OPT.DELETEFRAMECHECK

```
  [LAMBDA (VARS F)                                             (* lmm%: "22-JUL-77 02:58")
    (PROG ((CD (OPT.ONLYMEMB (CAR VARS)
                     CODE)))
          (OR (AND CD (EQ (fetch PREV of CD)
                          (CADR F)))
              (RETURN))
      LP  (SETQ VARS (CDR VARS))
          (SETQ CD (fetch NXT of CD))
          (COND
             ((NULL VARS)
              (RETURN T)))
          (COND
             ((EQ (OPT.ONLYMEMB (CAR VARS)
                                CODE)
                  CD)
              (GO LP])
```

(OPT.ONLYMEMB

```
  [LAMBDA (X Y)                                                (* lmm%: " 6-OCT-76 15:06:48")
    (AND (SETQ Y (FMEMB X Y))
         (NOT (FMEMB X (CDR Y)))
         Y])
```

)

```
(RPAQQ MERGEFRAMETYPES (PROG LAMBDA MAP))

(RPAQQ OPTIMIZATIONSOFF NIL)

(DEFINEQ
```

(OPT.SKIPPUSH

```
  [LAMBDA (CD N VL LEVOPFLG)                                   (* lmm "19-JAN-82 22:16")
    (OR N (SETQ N 1))
    (COND
       ((ILESSP N 0)
        NIL)
       ((ZEROP N)
        CD)
       (T (SELECTQ (fetch OPNAME of (CAR CD))
                ((AVAR HVAR FVAR GVAR CONST)
                     (OPT.SKIPPUSH (fetch PREV of CD)
                            (SUB1 N)
                            VL LEVOPFLG))
                (COPY (AND (NOT (fetch OPARG of (CAR CD)))
                           (OPT.SKIPPUSH (fetch PREV of CD)
                                  (SUB1 N)
                                  VL LEVOPFLG)))
```

```
            (SWAP (AND (IGEQ N 2)
                       (OPT.SKIPPUSH (fetch PREV of CD)
                              N VL LEVOPFLG)))
            (POP (OPT.SKIPPUSH (fetch PREV of CD)
                       (ADD1 N)
                       VL LEVOPFLG))
            ((FJUMP TJUMP NFJUMP NTJUMP)
                 (AND NEWOPTFLG (NOT LEVOPFLG)
                      (OPT.SKIPPUSH (fetch PREV of CD)
                             (ADD1 N)
                             VL LEVOPFLG)))
            (FN (COND
                   ((OR (COMP.CLEANFNOP (CDR (fetch OPARG of (CAR CD)))
                              'NOSIDE)
                        (AND NEWOPTFLG (SELECTQ (fetch OPNAME of (CAR VL))
                                                ((CONST HVAR)
                                                   T)
                                                ((FVAR AVAR GVAR)
                                                     (COMP.CLEANFNOP (CDR (fetch OPARG of (CAR CD)))
                                                            'FREEVARS))
                                                NIL)))
                    (OPT.SKIPPUSH (fetch PREV of CD)
                           [SUB1 (IPLUS N (CAR (fetch OPARG of (CAR CD]
                           VL LEVOPFLG))))
            (SETQ (COND
                     ([AND NEWOPTFLG VL (NEQ (CAR VL)
                                             (fetch OPARG of (CAR CD]
                      (OPT.SKIPPUSH (fetch PREV of CD)
                             N VL LEVOPFLG))))
           NIL])
```

(OPT.DELCODE

```
  [LAMBDA (CD)                                                    ; Edited 10-Jul-90 23:45 by jds
```

;; Remove (unreachable) code from the code stream.

```
    (PROG (X FLG)
      LP  (SELECTQ (fetch OPNAME of (SETQ X (CAR CD)))
                   (NIL (RETURN FLG))
                   (TAG (RETURN FLG))
                   ((BIND ERRORSET)
                        (RPLACA (CDR (FASSOC (CDR (fetch OPARG of X))
                                             FRAMES))
                               NIL)
                        (for LB in CL:LABELS when (EQ (fetch (TAG FRAME) of (CAR LB))
                                                      (CDR (fetch OPARG of X)))
                           do (MAPC (CDR LB)
                                    (FUNCTION OPT.PRDEL))))
                   ((UNBIND DUNBIND)
                        (DREMOVE CD (FASSOC (CDR (fetch OPARG of X))
                                            FRAMES)))
                   ((JUMP FJUMP TJUMP NFJUMP NTJUMP ERRORSET)        ; delete unreachable jump
                        (OPT.DELTAGREF CD)
                        (SETQ FLG T))
                                                                     ; delete unreachable code
                   )
            (SETQ ANY T)
            (SETQ CD (PROG1 (fetch NXT of CD)
                            (OPT.PRDEL CD)))
            (GO LP])
```

(OPT.PRATTACH

```
  [LAMBDA (ITEM BEFORE)                                           (* lmm%: "22-JUL-77 02:58")
    (PROG ((AFTER (fetch NXT of BEFORE))
           (NEW (CONS)))
          (replace NXT of NEW with AFTER)
          (replace PREV of NEW with BEFORE)
          (FRPLACA NEW ITEM)
          (replace NXT of BEFORE with NEW)
          (AND AFTER (replace PREV of AFTER with NEW))
          (RETURN NEW])
```

(OPT.JUMPCOPYTEST

```
  [LAMBDA (VL CDFROM)                                             (* lmm "15-JAN-82 18:08")
```

(* Where can a COPY be inserted such that VL would be on the stack -
either returns the code list or NIL -
used by transformation -
var TJUMP->l var |...| l%: var -
=> var COPY TJUMP->l2 |...| l%: var l2%:)

```
    (COND
       ((OPT.EQVALUE CDFROM VL)
        CDFROM)
       ((AND (OPT.CALLP (CAR CDFROM))
```

```
            (OR (EQ (fetch OPNAME of (CAR VL))
                    'HVAR)
                (COMP.CLEANFNP (CDR (fetch OPARG of (CAR CDFROM)))
                       'FREEVARS))
            (SETQ CDFROM (OPT.SKIPPUSH (fetch PREV of CDFROM)
                                  [SUB1 (CAR (fetch OPARG of (CAR CDFROM]
                                  VL T)))
        (OPT.JUMPCOPYTEST VL CDFROM])
```

(OPT.EQOP

```
  [LAMBDA (OP1 OP2)                                          (* lmm " 8-JAN-82 09:04")
    (OR (EQ OP1 OP2)
        (AND (EQ (fetch OPNAME of OP1)
                 (fetch OPNAME of OP2))
             (SELECTQ (fetch OPNAME of OP1)
                  ((FVAR GVAR CONST COPY STORE)
                       (EQ (fetch OPARG of OP1)
                           (fetch OPARG of OP2)))
                  ((POP RETURN SWAP)
                       [OPT.CCHECK (AND (NOT (fetch OPARG of OP1))
                                        (NOT (fetch OPARG of OP2]
                       T)
                  (FN (EQUAL OP1 OP2))
                  ((JUMP TJUMP NTJUMP FJUMP NFJUMP BIND ERRORSET UNBIND DUNBIND)
                       [AND (EQ (CAR (fetch OPARG of OP1))
                                (CAR (fetch OPARG of OP2)))
                            (EQ (CDR (fetch OPARG of OP1))
                                (CDR (fetch OPARG of OP2])
                  (SETQ (OPT.EQOP (fetch OPARG of OP1)
                                  (fetch OPARG of OP2)))
                  NIL])
```

(OPT.EQVALUE

```
  [LAMBDA (CD V)                                             (* lmm "19-JAN-82 22:25")
    (PROG NIL
      LP  (RETURN (SELECTQ (fetch OPNAME of (CAR CD))
                       (COPY (COND
                                 ((NULL (fetch OPARG of (CAR CD)))
                                  (SETQ CD (fetch PREV of CD))
                                  (GO LP))))
                       (SETQ (COND
                                 ((EQ (fetch OPARG of (CAR CD))
                                      (CAR V)))
                                 (T (SETQ CD (fetch PREV of CD))
                                    (GO LP))))
                       ((HVAR AVAR FVAR GVAR CONST)
                            (EQ (CAR CD)
                                (CAR V)))
                       ((POP FJUMP TJUMP NFJUMP NTJUMP SWAP)
                            (COND
                               ((SETQ CD (OPT.SKIPPUSH (fetch PREV of CD)
                                                       1 V))
                                (GO LP))))
                       NIL])
```

(OPT.DELCOPYFN

```
  [LAMBDA (P X)                                              (* lmm "18-JAN-82 13:17")
    (while (AND (OPT.CALLP (CAR P)
                           NIL 1)
                (OPT.EQOP (CAR P)
                          (CAR (fetch NXT of X)))
                (COMP.CLEANFNOP (CDR (fetch OPARG of (CAR P)))
                       'NOSIDE)
                (for Z_P by (fetch PREV of Z) while (AND Z (NEQ Z X))
                   always (SELECTQ (fetch OPNAME of (CAR Z))
                                (FN (COMP.CLEANFNOP (CDR (fetch OPARG of (CAR Z)))
                                                    'NOSIDE))
                                ((FVAR AVAR HVAR GVAR SETQ)
```

(* SETQ is OK since we have already guaranteed that the value skipped is not modified by intervening setqs)

```
                                    T)
                                NIL)))
      do [SETQ P (fetch NXT of (PROG1 (fetch PREV of P)
                                      (OPT.PRDEL P]
          (SETQ X (fetch NXT of X)))
    X])

)

(DEFINEQ
```

(OPT.DEADSETQP

```
  [LAMBDA (VAR CD)                                           (* lmm "13-Jul-84 21:18")
```

```
    (DECLARE (SPECVARS ICNT))
    (SELECTQ (fetch OPNAME of VAR)
        ((AVAR HVAR)
            (PROG (TAGS (ICNT 50))                          (* ICNT is used to limit the nmber of instructions looked at past
                                                               the setq.)
                                                            (* look for dead SETQ)
                  (RETURN (OPT.DS1 VAR CD))))
        NIL])
```

(OPT.DS1

```
  [LAMBDA (VAR CD)                                          (* lmm "13-Jul-84 21:18")

          (* test if VAR is used in CD -- TAGS is a list of tags already visited)

    (PROG (A)
      LP  [SELECTQ (fetch OPNAME of (SETQ A (CAR CD)))
            (SETQ (AND (EQ (fetch OPARG of A)
                           VAR)
                       (RETURN T)))
            (FN (AND (EQ (fetch OPNAME of VAR)
                         'AVAR)
                     (NOT (COMP.CLEANFNOP (CDR (fetch OPARG of A))
                                          'FREEVARS))
                     (RETURN)))
            ((UNBIND DUNBIND)
                (COND
                   ([FMEMB VAR (fetch (FRAME VARS) of (CDR (fetch OPARG of A]
                    (RETURN T))))
            (RETURN [RETURN (AND (SETQ A (OPT.CODEFRAME (fetch PREV of CD)))
                                 (never (EQ (fetch FRAMETYPE of A)
                                            'ERRORSET)
                                        repeatwhile (SETQ A (fetch PARENT of A])
            (JUMP (OR [SETQ CD (CAR (OPT.DEFREFS (fetch (JUMP TAG) of A]
                      (RETURN))
                  (GO LP))
            ((TJUMP FJUMP NTJUMP NFJUMP ERRORSET)
                (OR [OPT.DS1 VAR (CAR (OPT.DEFREFS (fetch (JUMP TAG) of A]
                    (RETURN)))
            (TAG [COND
                    ((FMEMB A TAGS)
                     (RETURN T))
                    (T (SETQ TAGS (CONS A TAGS])
            (COND
               ((EQ A VAR)
                (RETURN]
          (OR (SETQ CD (fetch NXT of CD))
              (OPT.COMPILERERROR))
      NX  [COND
             ((ZEROP ICNT)                                  (* DEADSETP gives up)
              (RETURN))
             (T (SETQ ICNT (SUB1 ICNT]
          (GO LP])

)

(RPAQ? *BC-MACRO-ENVIRONMENT* (COMPILER::MAKE-ENV))

(RPAQ? *BYTECOMPILER-OPTIMIZE-MACROLET* T)


(DEFMACRO CL:MACROLET (CL::MACRODEFS &BODY CL::BODY &ENVIRONMENT CL::ENV)
   (DECLARE (SPECVARS *BYTECOMPILER-IS-EXPANDING*))
   ;; This macro for the old interpreter and compiler only.  The new interpreter has a special-form definition.  When the new compiler is expanding, we
   ;; simply return a disguised version of the form.
   (IF (AND *BYTECOMPILER-IS-EXPANDING* *BYTECOMPILER-OPTIMIZE-MACROLET*)
       THEN (LET ((CL::NEW-ENV (COMPILER::MAKE-CHILD-ENV CL::ENV)))
                 (DECLARE (CL:SPECIAL *BC-MACRO-ENVIRONMENT*))
                 [FOR CL::FN IN CL::MACRODEFS DO (COMPILER::ENV-BIND-FUNCTION CL::NEW-ENV (CAR CL::FN)
                                                                              :MACRO
                                                                              (COMPILER::CRACK-DEFMACRO (CONS 'DEFMACRO CL::FN]
                 (CL:SETQ *BC-MACRO-ENVIRONMENT* CL::NEW-ENV)
                 (CONS 'CL:LOCALLY CL::BODY))
     ELSEIF (TYPEP CL::ENV 'COMPILER:ENV)
       THEN `(SI::%%MACROLET ,CL::MACRODEFS ,@CL::BODY)
     ELSE (LET* ((CL::NEW-ENV (\MAKE-CHILD-ENVIRONMENT CL::ENV))
                 (CL::FUNCTIONS (ENVIRONMENT-FUNCTIONS CL::NEW-ENV)))
                (FOR CL::FN IN CL::MACRODEFS
                   DO (CL:SETQ CL::FUNCTIONS (LIST* (CAR CL::FN)
                                                    [CONS :MACRO
                                                          `(CL:LAMBDA (SI::$$MACRO-FORM SI::$$MACRO-ENVIRONMENT
                                                                       )
                                                                  (CL:BLOCK ,(CAR CL::FN)
                                                                      ,(PARSE-DEFMACRO (CADR CL::FN)
                                                                              'SI::$$MACRO-FORM
                                                                              (CDDR CL::FN)
```

```
                                                                    (CAR CL::FN)
                                                                    NIL :ENVIRONMENT
                                                                    'SI::$$MACRO-ENVIRONMENT))]
                                                  CL::FUNCTIONS)))
                (CL:SETF (ENVIRONMENT-FUNCTIONS CL::NEW-ENV)
                         CL::FUNCTIONS)
                (WALK-FORM (CONS 'CL:LOCALLY CL::BODY)
                       :ENVIRONMENT CL::NEW-ENV))))

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(SPECVARS *BYTECOMPILER-IS-EXPANDING* *BC-MACRO-ENVIRONMENT*)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(SPECVARS CODE LEVEL)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(SPECVARS CL:LABELS PASS ANY CODE FRAME FRAMES)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS MERGEFRAMEMAX MERGEFRAMEFLG MERGEFRAMETYPES *BYTECOMPILER-OPTIMIZE-MACROLET*)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(SPECVARS VARS ANY FRAME)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(SPECVARS ICNT TAG)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(SPECVARS FRAME LEVEL ANY)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(SPECVARS FRAME LEVEL ANY)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(SPECVARS TAGS ANY)
)
)
```

;; CONSISTENCY CHECKS

```
(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS OPT.CCHECK MACRO [ARGS (COND
                                    (COMPILECOMPILERCHECKS (LIST 'AND (LIST 'NOT (CAR ARGS))
                                                                 (LIST 'OPT.COMPILERERROR (CADR ARGS])
)

(RPAQQ COMPILECOMPILERCHECKS NIL)
)

(DEFINEQ
```

(OPT.COMPILERERROR

```
  [LAMBDA (MESS1 MESS2)                                    (* lmm " 1-MAR-78 02:55")
    (LISPXPRIN1 "Compiler error
          " T)
    (HELP MESS1 MESS2])
```

(OPT.OPTCHECK

```
  [LAMBDA NIL                                              (* lmm "14-MAR-81 11:03")
                                                           (* set up code list as doubly linked list, scan for tags)
    (PROG ((CD CODE)
           P B)
      LPC (COND
             ((NULL CD)
```

```
            [for X in CL:LABELS do (COND
                                       ((CDR X)
                                        [OR (FMEMB (CAR X)
                                                   CODE)
                                            (OPT.COMPILERERROR (CAR X)
                                                   '(not in code]
                                        [MAPC (CDR X)
                                              (FUNCTION (LAMBDA (Y)
                                                        (OR (TAILP Y CODE)
                                                            (OPT.COMPILERERROR Y '(NOT CODE TAIL]
                                        [OR (EQ (CAR (CADR X))
                                                (CAR X))
                                            (OPT.COMPILERERROR X '(TAG wrong]
                                        (EVERY (CDDR X)
                                               (FUNCTION (LAMBDA (Y)
                                                         (OR (EQ (fetch (JUMP TAG) of (CAR Y))
                                                                 (CAR X))
                                                             (OPT.COMPILERERROR X '(TAG wrong]
            [for X in FRAMES do (COND
                                    [(EQ (CAR X)
                                         TOPFRAME)
                                     (AND (CDR X)
                                          (OPT.COMPILERERROR (CONS 'TOPFRAME X]
                                    (T [for Y in (CDR X) do (OR (TAILP Y CODE)
                                                                (OPT.COMPILERERROR (LIST '(NOT IN CODE)
                                                                                         Y X)))
                                                               (OR (EQ (CDR (fetch OPARG of (CAR Y)))
                                                                       (CAR X))
                                                                   (OPT.COMPILERERROR (LIST '(WRONG FRAME)
                                                                                            Y X]
                                       (OR (FASSOC (fetch PARENT of (CAR X))
                                                   FRAMES)
                                           (OPT.COMPILERERROR '(PARENT NOT FRAME)
                                                  X]
                (RETURN T)))
           (SELECTQ (fetch OPNAME of (CAR CD))
                (TAG (OR (SETQ B (FASSOC (CAR CD)
                                         CL:LABELS))
                         (OPT.COMPILERERROR))
                     (OR (EQ (CAR (CDR B))
                             CD)
                         (OPT.COMPILERERROR))
                     (OR (OR (NULL (fetch (TAG FRAME) of (CAR CD)))
                             (FASSOC (fetch (TAG FRAME) of (CAR CD))
                                     FRAMES))
                         (OPT.COMPILERERROR)))
                ((BIND ERRORSET)
                     (OR (EQ (CADR (FASSOC (CDR (fetch OPARG of (CAR CD)))
                                           FRAMES))
                             CD)
                         (OPT.COMPILERERROR)))
                ((UNBIND DUNBIND)
                     (OR (FMEMB CD (CDDR (FASSOC (CDR (fetch OPARG of (CAR CD)))
                                                 FRAMES)))
                         (OPT.COMPILERERROR)))
                ((JUMP TJUMP FJUMP NTJUMP NFJUMP)
                     (OR (SETQ B (FASSOC (fetch (JUMP TAG) of (CAR CD))
                                         CL:LABELS))
                         (OPT.COMPILERERROR))
                     [OR (MEMB CD B)
                         (OPT.COMPILERERROR CD '(NOT IN JUMP LIST])
                NIL)
           (SETQ B (CDR CD))
           (OR (AND (EQ (fetch PREV of CD)
                        B)
                    (EQ (fetch NXT of CD)
                        P))
               (OPT.COMPILERERROR))
           (SETQ P CD)
           (SETQ CD B)
           (GO LPC])
```

(OPT.CCHECK

```
  [LAMBDA (X)                                          (* lmm "14-MAR-81 09:18")
    (OR X (OPT.COMPILERERROR])
```

)

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS ALAMS BYTE.EXT BYTEASSEMFN BYTECOMPFLG COMPILERMACROPROPS CIA CLEANFNLIST COMP.SCRATCH
       COMPILETYPELST COMPILEUSERFN COMPSTATLST COMPSTATS CONDITIONALS CONST.FNS CONSTOPS DONOTHING FILERDTBL
       FNA FORSHALLOW FRA HEADERBYTES HOKEYDEFPROP LAMBDANOBIND LAMS LBA LEVELARRAY LINKEDFNS LOADTIMECONSTANT
       MAXBNILS MAXBVALS MCONSTOPS MERGEFRAMEFLG MERGEFRAMEMAX MERGEFRAMETYPES MOPARRAY MOPCODES NODARR
       NOSTATSFLG NUMBERFNS OPCOPY OPNIL OPPOP OPRETURN PRA SELECTQFMEMB SELECTVARTYPES STATAR STATMAX STATN
       SYSSPECVARS UNIQUE#ARRAY VCA VCONDITIONALS VREFFRA COUTFILE XVARFLG MERGEFRAMEFLG OPTIMIZATIONSOFF
```

```
       NOFREEVARSFNS EQCONSTFN NEWOPTFLG)
)

(CL:PROCLAIM '(CL:SPECIAL COMPVARMACROHASH))

(DECLARE%: DONTCOPY EVAL@COMPILE

(DECLARE%: EVAL@COMPILE

[RECORD CODELST (OP . PREV)
       (ACCESSFNS CODELST ((NXT (GETHASH DATUM PRA)
                                (PUTHASH DATUM NEWVALUE PRA]
)

(PUTPROPS OASSOC MACRO ((X Y)
                        (FASSOC X Y)))

(DECLARE%: EVAL@COMPILE

(RECORD OP (OPNAME . OPARG))

(RECORD JUMP (OPNAME TAG . JT)                                  (* kind of OP)
       )

[TYPERECORD TAG (LBNO . LEVEL)                                  (* kind of OP)
       LBNO _ (SETQ LBCNT (ADD1 LBCNT))
       (ACCESSFNS TAG ((FRAME (GETHASH DATUM FRA)
                              (PUTHASH DATUM NEWVALUE FRA))
                       (JD (GETHASH DATUM LBA)
                           (PUTHASH DATUM NEWVALUE LBA]

(RECORD VAR (COMP.VARTYPE . VARNAME)                            (* A particular kind of OP)
       )
)

(DECLARE%: EVAL@COMPILE

(RECORD FRAME (FRAMETYPE (NNILS VARS . DECLS)
                     LEVEL
                     (BINDLST NVALS EXTCALL . CPIOK) . PROGLABELS)

          (* FRAMETYPE is one of PROG LAMBDA ERRORSET MAP NIL -
          VARS are variables bound, NNILS are %# which are bound to NIL -
          LEVEL is %# of things on stack between this and next higher frame)

       (ACCESSFNS FRAME ((PARENT (GETHASH DATUM FRA)
                                 (PUTHASH DATUM NEWVALUE FRA))
                         (VREFFROM (GETHASH DATUM VREFFRA)
                                 (PUTHASH DATUM NEWVALUE VREFFRA))
                         (NODBIND (GETHASH DATUM NODARR)
                                 (PUTHASH DATUM NEWVALUE NODARR))
                         (PRIMARYRETURN (GETHASH DATUM BCINFO)
                                 (PUTHASH DATUM NEWVALUE BCINFO)))
                                                                (* PARENT is next higher enclosing frame -
                                                                shares hash table with TAG.FRAME)
              )
       (RECORD CPIOK NOXVAR

          (* Share the CPIOK field used by the compiler pass 1 and the NOXVAR field used by the maxc assembler)
)
       NNILS _ 0)

(RECORD COMINFO (COMTYPE TOPFRAME CODE ARGS))

[ACCESSFNS COMP (CLEAR (PROGN (OPT.INITHASH FRA)
                              (OPT.INITHASH LBA)
                              (OPT.INITHASH PRA)
                              (OPT.INITHASH VREFFRA)
                              (OPT.INITHASH NODARR)
                              (OPT.INITHASH BCINFO]

(RECORD JD (JPT (JMIN . JSN)
                JU . JML)

          (* JPT is NIL (for tags) or a pointer into ACODE (for jumps)%. JMIN is the lowest possible location for the instruction or tag.
          JU is the cumulative uncertainty (for tags) or the length uncertainty
          (for jumps)%. JML is the minimum length (for jumps)%. JSN is a serial number
          (the original JMIN) used to decide whether a jump goes forward or backward.)

       )

(RECORD BLOCKSTATUS (BLOCKCONTEXT BLOCKTAG BLOCKEND))
)
)

(DECLARE%: EVAL@COMPILE
```

```
(PUTPROPS THETYPE MACRO [(THETYPE . FORMS)
                         ([LAMBDA (THEVALUE)
                            (DECLARE (LOCALVARS THEVALUE)
                                     (TYPE THETYPE THEVALUE))
                           THEVALUE] . FORMS])
)

(PUTPROPS BYTECOMPILER FILETYPE CL:COMPILE-FILE)

(PUTPROPS BYTECOMPILER MAKEFILE-ENVIRONMENT (:READTABLE "INTERLISP" :PACKAGE "INTERLISP"))

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

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML OPT.INITHASH)

(ADDTOVAR LAMA )
)

(PUTPROPS BYTECOMPILER COPYRIGHT ("Venue & Xerox Corporation" 1981 1982 1983 1984 1985 1986 1987 1900 1988
                                  1989 1990 1991 2021))
```

FUNCTION INDEX

MACRO INDEX

PROPERTY INDEX

VARIABLE INDEX

OPTIMIZER INDEX

RECORD INDEX