*File created:* 22-Jun-88 20:23:31 **{POGO:AISNORTH:XEROX}<LOOPSCORE>SAVOIR>SYSTEM>MPATCH.;4**

*changes to:* (VARS MPATCHCOMS)
(FNS COMP.EXPR)
(FUNCTIONS CL:MACROLET)
(OPTIMIZERS CL:MACROLET)

*previous date:* 22-Jun-88 19:02:49 {POGO:AISNORTH:XEROX}<LOOPSCORE>SAVOIR>SYSTEM>MPATCH.;3

*Read Table:* INTERLISP

*Package:* INTERLISP

*Format:* XCCS

```
;;
;; Copyright (c) 1988 by Savoir and Xerox.  All rights reserved.


(RPAQQ MPATCHCOMS ((FUNCTIONS CL:MACROLET)
                   (FNS COMP.TRANSFORM COMP.EXPR)
                   (INITVARS (*BYTECOMPILER-OPTIMIZE-MACROLET* T)
                         (*BC-MACRO-ENVIRONMENT* (COMPILER::MAKE-ENV)))
                   (GLOBALVARS *BYTECOMPILER-OPTIMIZE-MACROLET*)
                   (PROP FILETYPE MPATCH)))


(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))))

(DEFINEQ

(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])
```

## (COMP.EXPR

```
  [LAMBDA (EXP COMPILE.CONTEXT)                                  ; Edited 22-Jun-88 19:59 by TAL
    (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]
      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)))
                           (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])

)

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

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

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS *BYTECOMPILER-OPTIMIZE-MACROLET*)
)

(PUTPROPS MPATCH FILETYPE :COMPILE-FILE)

(PUTPROPS MPATCH COPYRIGHT ("Savoir and Xerox" 1988))
```

## FUNCTION INDEX



## VARIABLE INDEX



## PROPERTY INDEX



## MACRO INDEX