File created: 4-Jun-90 14:31:10 **{PELE:MV:ENVOS}<LISPCORE>SOURCES>XCLC-OPTIMIZERS.;3**

changes to:
```
(IL:FUNCTIONS OPTIMIZE-LOGICAL-OP-1-ARG)
(OPTIMIZERS LOGIOR LOGXOR LOGAND LOGEQV)
(IL:VARS IL:XCLC-OPTIMIZERSCOMS)
```

previous date: 23-May-90 13:13:25 {PELE:MV:ENVOS}<LISPCORE>SOURCES>XCLC-OPTIMIZERS.;2

Read Table: XCL

Package: COMPILER

Format: XCCS

```
(IL:RPAQQ IL:XCLC-OPTIMIZERSCOMS
          (

;;; Compiler optimizers

            (IL:DEFINE-TYPES OPTIMIZERS)
            (IL:FUNCTIONS OPTIMIZER-LIST)
            (IL:PROP IL:PROPTYPE OPTIMIZER-LIST)
            (IL:FUNCTIONS DEFOPTIMIZER)

            ;; Random optimizers defined within the compiler.

            (OPTIMIZERS CAAAAR CAAADR CAAAR CAADAR CAADDR CAADR CAAR CADAAR CADADR CADAR CADDAR CADDDR CADDR CADR
                   CDAAAR CDAADR CDAAR CDADAR CDADDR CDADR CDAR CDDAAR CDDADR CDDAR CDDDAR CDDDDR CDDDR CDDR)
            (OPTIMIZERS (IL:ARG :OPTIMIZED-BY CONVERT-ARG-TO-\\ARG)
                   (IL:SETARG :OPTIMIZED-BY CONVERT-SETARG-TO-\\SETARG))
            (OPTIMIZERS VALUES VALUES-LIST)
            (OPTIMIZERS IL:LOADTIMECONSTANT IL:GETD IL:FGETD IL:EVQ)
            (OPTIMIZERS EQ EQL IL:EQP EQUAL IL:EQUAL EQUALP)
            (IL:FUNCTIONS OPTIMIZE-EQUALITY OPTIMIZE-EQL)
            (OPTIMIZERS (MULTIPLE-VALUE-CALL :OPTIMIZED-BY SCREEN-MV-CALL)
                   (NOT :OPTIMIZED-BY NOT-TO-IF)
                   (NULL :OPTIMIZED-BY NULL-TO-IF))
            (OPTIMIZERS IL:\\CALLME)

            ;; Optimizers for File Manager forms

            (IL:VARIABLES *INPUT-FILECOMS-VARIABLE*)
            (OPTIMIZERS IL:RPAQ IL:RPAQ? IL:RPAQQ IL:PRETTYCOMPRINT IL:FILECREATED)

            ;; Other Otimization

            (OPTIMIZERS IL:\\PILOTBITBLT)

            ;; Use the proper makefile-environment

            (IL:PROP IL:MAKEFILE-ENVIRONMENT IL:XCLC-OPTIMIZERS)

            ;; Use the proper compiler.

            (IL:PROP IL:FILETYPE IL:XCLC-OPTIMIZERS)))



;;; Compiler optimizers


(DEF-DEFINE-TYPE OPTIMIZERS "Compiler optimizers")


(DEFMACRO OPTIMIZER-LIST (FN)
   `(GET ,FN 'OPTIMIZER-LIST))

(IL:PUTPROPS OPTIMIZER-LIST IL:PROPTYPE IGNORE)

(DEFDEFINER (DEFOPTIMIZER (:PROTOTYPE (LAMBDA (XCL::NAME)
                                             (IF (SYMBOLP XCL::NAME)
                                                 `(DEFOPTIMIZER ,XCL::NAME ("Arg list")
                                                                           "Body")
                                                 (DESTRUCTURING-BIND (XCL::FORM-NAME XCL::OPTIMIZED-BY
                                                                            XCL::OPT-NAME)
                                                        XCL::NAME
                                                        (AND (EQ ':OPTIMIZED-BY XCL::OPTIMIZED-BY)
                                                             (NOT (NULL XCL::OPT-NAME))
                                                             `(DEFOPTIMIZER ,XCL::FORM-NAME ,XCL::OPT-NAME
                                                                                           ("Arg list")
                                                                                           "Body"))))))
                              (:NAME (LAMBDA (IL:WHOLE)
                                            (LET ((IL:NAME (SECOND IL:WHOLE))
                                                  (IL:OPT-NAME (THIRD IL:WHOLE)))
                                             (IF (LISTP IL:OPT-NAME)
                                                 IL:NAME      ; (defoptimizer form-name arglist . body)
                                                 `(,IL:NAME :OPTIMIZED-BY ,IL:OPT-NAME)
                                                              ; (defoptimizer form-name opt-name [arg-list . body])

                                                 )))))
    OPTIMIZERS (IL:NAME IL:OPT-NAME &REST IL:ARGLIST-BODY &ENVIRONMENT IL:ENV)
   (COND
      ((NOT IL:ARGLIST-BODY)                                           ; (defoptimizer name optfn)
       `(EVAL-WHEN (EVAL COMPILE LOAD)
               (PUSHNEW ',IL:OPT-NAME (OPTIMIZER-LIST ',IL:NAME))))
      (T (LET* ((IL:ARG-LIST IL:OPT-NAME)
                (IL:OPT-FN-NAME (IL:|if| (LISTP IL:OPT-NAME)
                                    IL:|then|                          ; (defoptimizer form-name arglist . body)
                                         (PACK (LIST "optimize-" IL:NAME)
                                               (SYMBOL-PACKAGE IL:NAME))
                                  IL:|else|                            ; (defoptimizer form-name opt-name arglist . body)
                                        (IL:SETQ IL:ARG-LIST (IL:POP IL:ARGLIST-BODY))
                                        IL:OPT-NAME)))
               (MULTIPLE-VALUE-BIND (IL:BODY IL:DECLS IL:DOC)
                   (IL:PARSE-DEFMACRO IL:ARG-LIST 'IL:$$WHOLE IL:ARGLIST-BODY IL:NAME IL:ENV :ENVIRONMENT
                          'IL:$$ENV :CONTEXT 'IL:$$CTX)
                 `(EVAL-WHEN (EVAL COMPILE LOAD)
                         (SETF (SYMBOL-FUNCTION ',IL:OPT-FN-NAME)
                               #'(LAMBDA (IL:$$WHOLE IL:$$ENV IL:$$CTX)
                                        ,@IL:DECLS
                                        (BLOCK ,IL:OPT-FN-NAME ,IL:BODY)))
                         (PUSHNEW ',IL:OPT-FN-NAME (OPTIMIZER-LIST ',IL:NAME))))))))
```

;; Random optimizers defined within the compiler.

```
(DEFOPTIMIZER CAAAAR (CL::X)
                        `(CAR (CAR (CAR (CAR ,CL::X)))))

(DEFOPTIMIZER CAAADR (CL::X)
                        `(CAR (CAR (CAR (CDR ,CL::X)))))

(DEFOPTIMIZER CAAAR (CL::X)
                      `(CAR (CAR (CAR ,CL::X))))

(DEFOPTIMIZER CAADAR (CL::X)
                        `(CAR (CAR (CDR (CAR ,CL::X)))))

(DEFOPTIMIZER CAADDR (CL::X)
                        `(CAR (CAR (CDR (CDR ,CL::X)))))

(DEFOPTIMIZER CAADR (CL::X)
                      `(CAR (CAR (CDR ,CL::X))))

(DEFOPTIMIZER CAAR (CL::X)
                    `(CAR (CAR ,CL::X)))

(DEFOPTIMIZER CADAAR (CL::X)
                        `(CAR (CDR (CAR (CAR ,CL::X)))))

(DEFOPTIMIZER CADADR (CL::X)
                        `(CAR (CDR (CAR (CDR ,CL::X)))))

(DEFOPTIMIZER CADAR (CL::X)
                      `(CAR (CDR (CAR ,CL::X))))

(DEFOPTIMIZER CADDAR (CL::X)
                        `(CAR (CDR (CDR (CAR ,CL::X)))))

(DEFOPTIMIZER CADDDR (CL::X)
                        `(CAR (CDR (CDR (CDR ,CL::X)))))

(DEFOPTIMIZER CADDR (CL::X)
                      `(CAR (CDR (CDR ,CL::X))))

(DEFOPTIMIZER CADR (CL::X)
                    `(CAR (CDR ,CL::X)))

(DEFOPTIMIZER CDAAAR (CL::X)
                        `(CDR (CAR (CAR (CAR ,CL::X)))))

(DEFOPTIMIZER CDAADR (CL::X)
                        `(CDR (CAR (CAR (CDR ,CL::X)))))
```

```
(DEFOPTIMIZER CDAAR (CL::X)
                    `(CDR (CAR (CAR ,CL::X))))

(DEFOPTIMIZER CDADAR (CL::X)
                     `(CDR (CAR (CDR (CAR ,CL::X)))))

(DEFOPTIMIZER CDADDR (CL::X)
                     `(CDR (CAR (CDR (CDR ,CL::X)))))

(DEFOPTIMIZER CDADR (CL::X)
                    `(CDR (CAR (CDR ,CL::X))))

(DEFOPTIMIZER CDAR (CL::X)
                   `(CDR (CAR ,CL::X)))

(DEFOPTIMIZER CDDAAR (CL::X)
                     `(CDR (CDR (CAR (CAR ,CL::X)))))

(DEFOPTIMIZER CDDADR (CL::X)
                     `(CDR (CDR (CAR (CDR ,CL::X)))))

(DEFOPTIMIZER CDDAR (CL::X)
                    `(CDR (CDR (CAR ,CL::X))))

(DEFOPTIMIZER CDDDAR (CL::X)
                     `(CDR (CDR (CDR (CAR ,CL::X)))))

(DEFOPTIMIZER CDDDDR (CL::X)
                     `(CDR (CDR (CDR (CDR ,CL::X)))))

(DEFOPTIMIZER CDDDR (CL::X)
                    `(CDR (CDR (CDR ,CL::X))))

(DEFOPTIMIZER CDDR (CL::X)
                   `(CDR (CDR ,CL::X)))

(DEFOPTIMIZER IL:ARG CONVERT-ARG-TO-\\ARG
   (NAME EXPR)
   (IF *NEW-COMPILER-IS-EXPANDING*
       `(IL:\\ARG ',NAME ,EXPR)
       'PASS))

(DEFOPTIMIZER IL:SETARG CONVERT-SETARG-TO-\\SETARG
   (NAME EXPR NEW-VALUE)
   (IF *NEW-COMPILER-IS-EXPANDING*
       `(IL:\\SETARG ',NAME ,EXPR ,NEW-VALUE)
       'PASS))

(DEFOPTIMIZER VALUES (&REST CL::ARGS &CONTEXT CL::CTXT)
                     (COND
                        ((AND CL::ARGS (NULL (CDR CL::ARGS)))  ; Throw away extra values.
                         `((IL:OPCODES IL:NOP)
                           ,(CAR CL::ARGS)))
                        (*NEW-COMPILER-IS-EXPANDING* (CASE (CONTEXT-VALUES-USED CL::CTXT)
                                                         ((0) `(PROGN ,@CL::ARGS))
                                                         ((1) `(PROG1 ,@CL::ARGS))
                                                         (OTHERWISE `(IL:MISCN VALUES ,@CL::ARGS))))
                        (T `(IL:MISCN VALUES ,@CL::ARGS))))

(DEFOPTIMIZER VALUES-LIST (CL::ARG &CONTEXT CL::CTXT)
                          (IF *NEW-COMPILER-IS-EXPANDING*
                              (CASE (CONTEXT-VALUES-USED CL::CTXT)
                                  ((0) CL::ARG)
                                  ((1) `(CAR ,CL::ARG))
                                  (OTHERWISE `(IL:MISCN VALUES-LIST ,CL::ARG)))
                              `(IL:MISCN VALUES-LIST ,CL::ARG)))

(DEFOPTIMIZER IL:LOADTIMECONSTANT (IL:FORM)

;;; The new compiler uses an unforgable data structure to mark load-time forms.  The old ByteCompiler used LOADTIMECONSTANTMARKER, a unique
;;; string.
```

```
                                              (IF *NEW-COMPILER-IS-EXPANDING*
                                                  (MAKE-EVAL-WHEN-LOAD :FORM IL:FORM)
                                                  (LIST 'QUOTE (CONS IL:LOADTIMECONSTANTMARKER IL:FORM))))
```

```
(DEFOPTIMIZER IL:GETD (IL:FN &CONTEXT IL:CTXT)
                       (IF (CONTEXT-PREDICATE-P IL:CTXT)
                           `(IL:\\DEFINEDP ,IL:FN)
                           'PASS))
```

```
(DEFOPTIMIZER IL:FGETD (IL:FN)
                        `(IL:GETD ,IL:FN))
```

```
(DEFOPTIMIZER IL:EVQ (IL:ARG)
                      IL:ARG)
```

```
(DEFOPTIMIZER EQ (CL::ONE CL::TWO)
                 (COND
                    ((AND (CONSTANTP CL::ONE)
                          (NULL (EVAL CL::ONE)))
                     `(NULL ,CL::TWO))
                    ((AND (CONSTANTP CL::TWO)
                          (NULL (EVAL CL::TWO)))
                     `(NULL ,CL::ONE))
                    (T 'PASS)))
```

```
(DEFOPTIMIZER EQL (&WHOLE CL::FORM)
                  (OPTIMIZE-EQL CL::FORM))
```

```
(DEFOPTIMIZER IL:EQP (&WHOLE IL:FORM)
                     (OPTIMIZE-EQUALITY IL:FORM))
```

```
(DEFOPTIMIZER EQUAL (&WHOLE CL::FORM)
                    (OPTIMIZE-EQUALITY CL::FORM))
```

```
(DEFOPTIMIZER IL:EQUAL (&WHOLE IL:FORM)
                       (OPTIMIZE-EQUALITY IL:FORM))
```

```
(DEFOPTIMIZER EQUALP (&WHOLE CL::FORM)
                     (OPTIMIZE-EQUALITY CL::FORM))
```

```
(DEFUN OPTIMIZE-EQUALITY (FORM)
```

;;; FORM is a call on one of the equality-testing predicates EQL, IL:EQP, EQUAL, IL:EQUAL, or EQUALP. If one of the arguments is a literal symbol,
;;; then we can use EQ.

```
   (DESTRUCTURING-BIND (FN ONE TWO)
          FORM
          (COND
             ((AND (CONSTANTP ONE)
                   (SYMBOLP (EVAL ONE)))
              `(EQ ,TWO ',(EVAL ONE)))
             ((AND (CONSTANTP TWO)
                   (SYMBOLP (EVAL TWO)))
              `(EQ ,ONE ',(EVAL TWO)))
             (T 'PASS))))
```

```
(DEFUN OPTIMIZE-EQL (FORM)
```

   ;; TRANSFORM to EQ if possible

```
   (DESTRUCTURING-BIND (FN ONE TWO)
          FORM
          (LET (E-ONE E-TWO)
               (COND
                  ((AND (CONSTANTP ONE)
                        (OR (SYMBOLP (SETQ E-ONE (EVAL ONE)))
                            (TYPEP E-ONE 'FIXNUM)))
                   `(EQ ',E-ONE ,TWO))
                  ((AND (CONSTANTP TWO)
                        (OR (SYMBOLP (SETQ E-TWO (EVAL TWO)))
                            (TYPEP E-TWO 'FIXNUM)))
                   `(EQ ,ONE ',E-TWO))
                  (T 'PASS)))))
```

```
(DEFOPTIMIZER MULTIPLE-VALUE-CALL SCREEN-MV-CALL
   (FN &BODY BODY)
```

;;; “Optimizer” for special form MULTIPLE-VALUE-CALL - handle special case of list and let the rest turn into an APPLY

```
   (COND
      ((AND (EQUAL FN '(IL:FUNCTION LIST))
            (NULL (CDR BODY)))
       (CONS 'IL:\\MVLIST BODY))
      (T `(IL:APPLY ,FN (NCONC ,@(IL:FOR F IL:IN BODY IL:COLLECT `(MULTIPLE-VALUE-LIST ,F)))))))


(DEFOPTIMIZER NOT NOT-TO-IF
   (X)
   (IF *NEW-COMPILER-IS-EXPANDING*
       `(IF ,X
            NIL
            T)
       'PASS))


(DEFOPTIMIZER NULL NULL-TO-IF
   (X)
   (IF *NEW-COMPILER-IS-EXPANDING*
       `(IF ,X
            NIL
            T)
       'PASS))


(DEFOPTIMIZER IL:\\CALLME (NAME &CONTEXT CTXT)
                          (COND
                             ((NOT (EQL (CONTEXT-VALUES-USED CTXT)
                                        0))
                              (WARN "The ~S special form appeared in non-effect context." 'IL:\\CALLME)
                              `(PROGN (IL:\\CALLME ,NAME)
                                      NIL))
                             ((AND (NOT (CONSTANTP NAME))
                                   (OR (ATOM NAME)
                                       (NOT (EQ (CAR NAME)
                                                'QUOTE))))
                              (WARN "The ~S special form was given an unquoted argument." 'IL:\\CALLME)
                              `(IL:\\CALLME ',NAME))
                             (T 'PASS)))
```

;; Optimizers for File Manager forms

```
(DEFVAR *INPUT-FILECOMS-VARIABLE*
```

;;; Used for communication between the optimizers on RPAQQ and PRETTYCOMPRINT so that the file coms can be eliminated from the file during
;;; compilation.

```
   )


(DEFOPTIMIZER IL:RPAQ (VAR EXPR &CONTEXT CTXT)
                      (IF (CONTEXT-TOP-LEVEL-P CTXT)
                          `(LOCALLY (DECLARE (GLOBAL ,VAR))
                                   (SETQ ,VAR ,EXPR))
                          'PASS))


(DEFOPTIMIZER IL:RPAQ? (VAR EXPR &CONTEXT CTXT)
                       (IF (CONTEXT-TOP-LEVEL-P CTXT)
                           `(LOCALLY (DECLARE (GLOBAL ,VAR))
                                   (AND (EQ ,VAR 'IL:NOBIND)
                                        (SETQ ,VAR ,EXPR)))
                           'PASS))


(DEFOPTIMIZER IL:RPAQQ (VAR EXPR &CONTEXT CTXT)
                       (IF (CONTEXT-TOP-LEVEL-P CTXT)
                           `(LOCALLY (DECLARE (GLOBAL ,VAR))
                                   (SETQ ,VAR ',EXPR))
                           'PASS))


(DEFOPTIMIZER IL:PRETTYCOMPRINT (COMS-NAME &CONTEXT CTXT)
                                (COND
                                   ((CONTEXT-TOP-LEVEL-P CTXT)
                                    NIL)
                                   (T 'PASS)))


(DEFOPTIMIZER IL:FILECREATED (FILEDATE FILENAME &REST JUNK &CONTEXT CTXT)
                             (DECLARE (IGNORE JUNK))
                             (IF (AND (CONTEXT-TOP-LEVEL-P CTXT)
                                              FILENAME
                                              (SYMBOLP FILENAME))
                                         `(IL:PUTPROP ',(IL:ROOTFILENAME FILENAME)
                                                 'IL:FILEDATES
                                                 '(,(CONS FILEDATE FILENAME)))
                                         'PASS))
```

;; Other Otimization

```
(DEFOPTIMIZER IL:\\PILOTBITBLT (&REST IL:ARGS)
                                     (IF (AND IL:ARGS (NULL (CDR IL:ARGS)))
                                         `(IL:\\PILOTBITBLT ,@IL:ARGS NIL)
                                         'PASS))
```

;; Use the proper makefile-environment

```
(IL:PUTPROPS IL:XCLC-OPTIMIZERS IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE (DEFPACKAGE "COMPILER"
                                                                                       (:USE "LISP" "XCL"))))
```

;; Use the proper compiler.

```
(IL:PUTPROPS IL:XCLC-OPTIMIZERS IL:FILETYPE :COMPILE-FILE)

(IL:PUTPROPS IL:XCLC-OPTIMIZERS IL:COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1988 1989 1990))
```

FUNCTION INDEX

OPTIMIZE-EQL 4
OPTIMIZE-EQUALITY 4

OPTIMIZER INDEX

CAAAAR 2
CAAADR 2
CAAAR 2
CAADAR 2
CAADDR 2
CAADR 2
CAAR 2
CADAAR 2
CADADR 2
CADAR 2
CADDAR 2
CADDDR 2
CADDR 2
CADR 2
CDAAAR 2
CDAADR 2
CDAAR 3
CDADAR 3
CDADDR 3
CDADR 3
CDAR 3
CDDAAR 3
CDDADR 3
CDDAR 3
CDDDAR 3
CDDDDR 3
CDDDR 3
CDDR 3
EQ 4
EQL 4
IL:EQP 4
EQUAL 4
IL:EQUAL 4
EQUALP 4
IL:EVQ 4
IL:FGETD 4
IL:FILECREATED 5
IL:GETD 4
IL:LOADTIMECONSTANT 3
IL:PRETTYCOMPRINT 5
IL:RPAQ 5
IL:RPAQ? 5
IL:RPAQQ 5
VALUES 3
VALUES-LIST 3
IL:\\CALLME 5
IL:\\PILOTBITBLT 6

PROPERTY INDEX

OPTIMIZER-LIST 1
IL:XCLC-OPTIMIZERS 6

NULL INDEX

NULL-TO-IF 5
:OPTIMIZED-BY 5

NOT INDEX

NOT-TO-IF 5
:OPTIMIZED-BY 5

MULTIPLE-VALUE-CALL INDEX

:OPTIMIZED-BY 4
SCREEN-MV-CALL 4

SETARG INDEX

CONVERT-SETARG-TO-\\SETARG 3
:OPTIMIZED-BY 3

ARG INDEX

CONVERT-ARG-TO-\\ARG 3
:OPTIMIZED-BY 3

VARIABLE INDEX

INPUT-FILECOMS-VARIABLE 5

DEFINER INDEX

DEFOPTIMIZER 1

MACRO INDEX

OPTIMIZER-LIST 1

DEFINE-TYPE INDEX

OPTIMIZERS 1