

File created: 18-Oct-93 17:19:21 {Pele:mv:envos}<LispCore>Sources>CLTL2>XCLC-OPTIMIZERS.;2

previous date: 4-Feb-92 10:31:17 {Pele:mv:envos}<LispCore>Sources>CLTL2>XCLC-OPTIMIZERS.;1

Read Table: XCL

Package: COMPILER

Format: XCCS

; Copyright (c) 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993 by Venue & Xerox Corporation. All rights reserved.

(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 CADDR CADR
             CDAAAR CDAADR CDAAR CDADAR CDADDR CDADR CDAR CDDAAR CDDADR CDDAR CDDDR CDDDR CDDR 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 LOAD-TIME-VALUE)
(IL:SPECIAL-FORMS LOAD-TIME-VALUE)
(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 Optimization
(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 CADDR (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 CDDAR (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 unforgeable 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 LOAD-TIME-VALUE (CL::FORM &OPTIONAL CL::READ-ONLY-P)
  ;; Copied from IL:LOADTIMECONSTANT; they're the same thing to the PavCompiler, I believe...
  (IF *NEW-COMPILER-IS-EXPANDING*
    (MAKE-EVAL-WHEN-LOAD :FORM CL::FORM)
    (LIST 'QUOTE (CONS IL:LOADTIMECONSTANTMARKER CL::FORM))))
```

```
(DEFINE-SPECIAL-FORM LOAD-TIME-VALUE (CL::FORM &OPTIONAL CL::READ-ONLY-P)
  (EVAL CL::FORM NIL))
```

```
(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 1991 1992 1993
))
```

FUNCTION INDEX

OPTIMIZE-EQL4 OPTIMIZE-EQUALITY4

OPTIMIZER INDEX

CAAAAR2	CADDR2	CDDAR3	IL:FILECREATED6
CAAADR2	CADR2	CDDDDR3	IL:GETD4
CAAAAR2	CDAAR2	CDDDR3	LOAD-TIME-VALUE4
CAADAR2	CDAADR2	CDDR3	IL:LOADTIMECONSTANT3
CAADDR2	CDAAR2	EQ4	IL:PRETTYCOMPRINT6
CAADR2	CDADAR3	EQL4	IL:RPAQ5
CAAR2	CDADDR3	IL:EQP4	IL:RPAQ?5
CADAAR2	CDADR3	EQUAL4	IL:RPAQQ5
CADADR2	CDAR3	IL:EQUAL4	VALUES3
CADAR2	CDDAAR3	EQUALP4	VALUES-LIST3
CADDR2	CDDADR3	IL:EVQ4	IL:\\CALLME5
CADDDR2	CDDAR3	IL:FGETD4	IL:\\PILOTBITBLT6

PROPERTY INDEX

OPTIMIZER-LIST1 IL:XCLC-OPTIMIZERS6

NULL INDEX

NULL-TO-IF5 :OPTIMIZED-BY5

NOT INDEX

NOT-TO-IF5 :OPTIMIZED-BY5

MULTIPLE-VALUE-CALL INDEX

:OPTIMIZED-BY5 SCREEN-MV-CALL5

SETARG INDEX

CONVERT-SETARG-TO-\\SETARG3 :OPTIMIZED-BY3

ARG INDEX

CONVERT-ARG-TO-\\ARG3 :OPTIMIZED-BY3

VARIABLE INDEX

INPUT-FILECOMS-VARIABLE5

MACRO INDEX

OPTIMIZER-LIST1

SPECIAL-FORM INDEX

LOAD-TIME-VALUE4

DEFINE-TYPE INDEX

OPTIMIZERS1

{MEDLEY}<CLTL2>XCLC-OPTIMIZERS.;1

DEFINER INDEX

DEFOPTIMIZER1
