File created: 21-Sep-88 12:35:01 **{ERIS}<LISPCORE>INTERNAL>LIBRARY>XCLC-DEBUG.;2**

changes to: (PRINTERS RETURN-NODE THROW-NODE)
(IL:VARS IL:XCLC-DEBUGCOMS)

previous date: 11-Jan-88 19:40:48 {ERIS}<LISPCORE>INTERNAL>LIBRARY>XCLC-DEBUG.;1

Read Table: XCL

Package: COMPILER

Format: XCCS


```
(IL:RPAQQ IL:XCLC-DEBUGCOMS
          (
```

;;; Debugging support for the XCL Compiler

```
          ;; Printing nodes
          (IL:DEFINE-TYPES PRINTERS)
          (IL:FUNCTIONS DEFPRINTER SET-PF)
          (IL:COMMANDS "setpf")
                                                                    ; mv-prog1-node progv-node
          (PRINTERS BLOCK-NODE CALL-NODE CATCH-NODE IF-NODE GO-NODE LABELS-NODE LAMBDA-NODE LITERAL-NODE
                 MV-CALL-NODE MV-PROG1-NODE OPCODES-NODE PROGN-NODE RETURN-NODE SETQ-NODE TAGBODY-NODE
                 THROW-NODE UNWIND-PROTECT-NODE VAR-REF-NODE VARIABLE-STRUCT)
          (PRINTERS D-ASSEM::DVAR D-ASSEM::DTAG D-ASSEM::DJUMP D-ASSEM::DLAMBDA D-ASSEM:DCODE)
          ;; Mutator functions for SEdit
          (IL:FUNCTIONS OAM EOC)
          ;; Useful Exec commands
          (IL:COMMANDS "ic2")
          ;; Arrange to use the proper compiler.
          (IL:PROP (IL:FILETYPE IL:MAKEFILE-ENVIRONMENT)
                 IL:XCLC-DEBUG)))
```

;;; Debugging support for the XCL Compiler

;; Printing nodes

```
(DEF-DEFINE-TYPE PRINTERS "XCL Compiler node printing functions")

(DEFDEFINER (DEFPRINTER (:PROTOTYPE (LAMBDA (NAME)
                                            (AND (SYMBOLP NAME)
                                                 `(DEFPRINTER ,NAME ("Object" STREAM)
                                                              "Body")))))
    PRINTERS (TYPE ARGS &BODY (BODY DECLS))
   (LET ((PRINT-FN (INTERN (CONCATENATE 'STRING "\\print-" (STRING TYPE))
                           (SYMBOL-PACKAGE TYPE))))
        `(PROGN (DEFUN ,PRINT-FN (,@ARGS $$DEPTH)
                   (DECLARE (IGNORE $$DEPTH))
                   ,DECLS
                   (IF (OR (NULL *PRINT-LEVEL*)
                           (>= *PRINT-LEVEL* 0))
                       (LET ((*PRINT-LEVEL* (AND *PRINT-LEVEL* (1- *PRINT-LEVEL*))))
                            ,@BODY)
                       (PRINC "#" ,(SECOND ARGS))))
                (SET-PF ',TYPE ',PRINT-FN))))


(DEFUN SET-PF (TYPE FN)
   (LET ((PS (CL::PARSED-STRUCTURE TYPE)))
        (AND PS (SETF (CL::PS-PRINT-FUNCTION PS)
                      FN))))


(DEFCOMMAND "setpf" (TYPE FN)
   (LET ((PS (CL::PARSED-STRUCTURE TYPE)))
        (AND PS (PROG1 (CL::PS-PRINT-FUNCTION PS)
                       (SETF (CL::PS-PRINT-FUNCTION PS)
                             FN)))))
```

;; mv-prog1-node progv-node

```
(DEFPRINTER BLOCK-NODE (NODE STREAM)
                       (FORMAT STREAM "#<Block ~S ~S>" (BLOCK-NAME NODE)
                               (BLOCK-STMT NODE)))

(DEFPRINTER CALL-NODE (CALL STREAM)
                      (FORMAT STREAM "#<Call ~S ~{~S~^ ~}>" (CALL-FN CALL)
                              (CALL-ARGS CALL)))
```

```
(DEFPRINTER CATCH-NODE (NODE STREAM)
                       (FORMAT STREAM "#<Catch ~S ~S>" (CATCH-TAG NODE)
                              (CATCH-STMT NODE)))

(DEFPRINTER IF-NODE (NODE STREAM)
                    (FORMAT STREAM "#<If ~S ~S ~S>" (IF-PRED NODE)
                           (IF-THEN NODE)
                           (IF-ELSE NODE)))

(DEFPRINTER GO-NODE (NODE STREAM)
                    (FORMAT STREAM "#<Go ~S>" (GO-TAG NODE)))

(DEFPRINTER LABELS-NODE (LAB STREAM)
                        (FORMAT STREAM "#<Labels (~{(~S ~S)~}) ~S>" (MAPCAN
                                                                      #'(LAMBDA (FN-BINDING)
                                                                                (LIST (CAR FN-BINDING)
                                                                                      (CDR FN-BINDING)))
                                                                      (LABELS-FUNS LAB))
                              (LABELS-BODY LAB)))

(DEFPRINTER LAMBDA-NODE (LAM STREAM)
                        (FORMAT
                         STREAM "#<Lambda ~:S ~S>"
                         `(,@(LAMBDA-REQUIRED LAM)
                           ,@(AND (LAMBDA-OPTIONAL LAM)
                                  (CONS '&OPTIONAL (IL:FOR OPT-VAR IL:IN (LAMBDA-OPTIONAL LAM)
                                                       IL:COLLECT (IF (AND (LITERAL-P (SECOND OPT-VAR))
                                                                           (NULL (LITERAL-VALUE (SECOND
                                                                                                  OPT-VAR)))
                                                                           (NULL (THIRD OPT-VAR)))
                                                                      (FIRST OPT-VAR)
                                                                      OPT-VAR))))
                           ,@(AND (LAMBDA-REST LAM)
                                  (LIST '&REST (LAMBDA-REST LAM)))
                           ,@(AND (LAMBDA-KEYWORD LAM)
                                  (CONS '&KEY (IL:FOR KEY-VAR IL:IN (LAMBDA-KEYWORD LAM)
                                                      IL:COLLECT (COND
                                                                   ((AND (STRING= (FIRST KEY-VAR)
                                                                                  (VARIABLE-NAME (SECOND KEY-VAR)
                                                                                                 ))
                                                                         (LITERAL-P (THIRD KEY-VAR))
                                                                         (NULL (LITERAL-VALUE (THIRD KEY-VAR)))
                                                                         (NULL (FOURTH KEY-VAR)))
                                                                    (SECOND KEY-VAR))
                                                                   ((STRING= (FIRST KEY-VAR)
                                                                             (VARIABLE-NAME (SECOND KEY-VAR)))
                                                                    (CDR KEY-VAR))
                                                                   (T `((,(FIRST KEY-VAR)
                                                                         ,(SECOND KEY-VAR))
                                                                        ,@(CDDR KEY-VAR)))))))
                           ,@(AND (LAMBDA-ALLOW-OTHER-KEYS LAM)
                                  (LIST '&ALLOW-OTHER-KEYS)))
                          (LAMBDA-BODY LAM)))

(DEFPRINTER LITERAL-NODE (LIT STREAM)
                         (FORMAT STREAM "#<Lit: ~S>" (LITERAL-VALUE LIT)))

(DEFPRINTER MV-CALL-NODE (OBJ STREAM)
                         (FORMAT STREAM "#<MV-Call: ~S~{~^ ~S~}>" (MV-CALL-FN OBJ)
                                (MV-CALL-ARG-EXPRS OBJ)))

(DEFPRINTER MV-PROG1-NODE (NODE STREAM)
                          (FORMAT STREAM "#<MV-Prog1: ~{~S~^ ~}>" (MV-PROG1-STMTS NODE)))

(DEFPRINTER OPCODES-NODE (NODE STREAM)
                         (LET ((*PACKAGE* (IL:LOADTIMECONSTANT (FIND-PACKAGE "IL")))
                               (*PRINT-CASE* :DOWNCASE))
                              (FORMAT STREAM "#<Opcodes ~{~S~^ ~}>" (OPCODES-BYTES NODE))))

(DEFPRINTER PROGN-NODE (NODE STREAM)
                       (FORMAT STREAM "#<Progn ~{~S~^ ~}>" (PROGN-STMTS NODE)))

(DEFPRINTER RETURN-NODE (NODE STREAM)
                        (FORMAT STREAM "#<Return ~S ~S>" (RETURN-BLOCK NODE)
                               (RETURN-VALUE NODE)))

(DEFPRINTER SETQ-NODE (NODE STREAM)
                      (FORMAT STREAM "#<Setq ~S ~S>" (VARIABLE-NAME (SETQ-VAR NODE))
                            (SETQ-VALUE NODE)))

(DEFPRINTER TAGBODY-NODE (NODE STREAM)
                         (PRINC "#<Tagbody" STREAM)
                         (DOLIST (SEGMENT (TAGBODY-SEGMENTS NODE))
                             (FORMAT STREAM "~{ ~S~}~{ ~S~}" (SEGMENT-TAGS SEGMENT)
                                    (SEGMENT-STMTS SEGMENT)))
                         (PRINC ">" STREAM))
```

```
(DEFPRINTER THROW-NODE (NODE STREAM)
                       (FORMAT STREAM "#<Throw ~S ~S>" (THROW-TAG NODE)
                              (THROW-VALUE NODE)))

(DEFPRINTER UNWIND-PROTECT-NODE (UP STREAM)
                                (FORMAT STREAM "#<U-P ~S ~S>" (UNWIND-PROTECT-STMT UP)
                                       (UNWIND-PROTECT-CLEANUP UP)))

(DEFPRINTER VAR-REF-NODE (REF STREAM)
                         (FORMAT STREAM "#<Ref: ~S>" (VAR-REF-VARIABLE REF)))

(DEFPRINTER VARIABLE-STRUCT (VAR STREAM)
                            (FORMAT STREAM "#<~A: ~S>" (CASE (VARIABLE-KIND VAR)
                                                             (:FUNCTION "Fn")
                                                             (:VARIABLE "Var"))
                                   (VARIABLE-NAME VAR)))

(DEFPRINTER D-ASSEM::DVAR (D-ASSEM::VAR STREAM)
                          (FORMAT STREAM "#<DVar ~S>" (D-ASSEM::DVAR-NAME D-ASSEM::VAR)))

(DEFPRINTER D-ASSEM::DTAG (D-ASSEM::TAG STREAM)
                          (FORMAT STREAM "#<DTag @ ~O,~O>" (IL:\\HILOC D-ASSEM::TAG)
                                 (IL:\\LOLOC D-ASSEM::TAG)))

(DEFPRINTER D-ASSEM::DJUMP (D-ASSEM::JUMP STREAM)
                           (FORMAT STREAM "#<DJump @ ~O,~O>" (IL:\\HILOC D-ASSEM::JUMP)
                                  (IL:\\LOLOC D-ASSEM::JUMP)))

(DEFPRINTER D-ASSEM::DLAMBDA (D-ASSEM::DLAMBDA STREAM)
                             (FORMAT STREAM "#<DLambda ~S>" (D-ASSEM::DLAMBDA-NAME D-ASSEM::DLAMBDA)))

(DEFPRINTER D-ASSEM:DCODE (D-ASSEM:DCODE STREAM)
                          (FORMAT STREAM "#<DCode ~S>" (D-ASSEM::DCODE-FRAME-NAME D-ASSEM:DCODE)))
```

;; Mutator functions for SEdit

```
(DEFUN OAM (FORM)
   (COPY-TREE (OPTIMIZE-AND-MACROEXPAND-1 FORM (IL:LOADTIMECONSTANT (MAKE-ENV))
                      (IL:LOADTIMECONSTANT (MAKE-CONTEXT)))))


(DEFUN EOC (FORM)
   (LET ((*ENVIRONMENT* (IL:LOADTIMECONSTANT (MAKE-ENV)))
         (*CONTEXT* (IL:LOADTIMECONSTANT (MAKE-CONTEXT)))
         (FN (CAR FORM))
         (ARGS (CDR FORM)))
        (ASSERT (EQ (CAR FN)
                    'IL:OPENLAMBDA)
                NIL "EOC called on a non-OPENLAMBDA")
        (EXPAND-OPENLAMBDA-CALL FN ARGS)))
```

;; Useful Exec commands

```
(DEFCOMMAND "ic2" (XCL-USER::HI XCL-USER::LO)
   (FLET ((XCL-USER::OCTAL (XCL-USER::N)
                  (READ-FROM-STRING (FORMAT NIL "#o~D" XCL-USER::N))))
         (LET ((XCL-USER::CODE (IL:\\VAG2 (XCL-USER::OCTAL XCL-USER::HI)
                                          (XCL-USER::OCTAL XCL-USER::LO))))
              (IF (COMPILED-FUNCTION-P XCL-USER::CODE)
                  (IL:INSPECTCODE XCL-USER::CODE)
                  (INSPECT XCL-USER::CODE)))))
```

;; Arrange to use the proper compiler.

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

(IL:PUTPROPS IL:XCLC-DEBUG IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE "COMPILER"))

(IL:PUTPROPS IL:XCLC-DEBUG IL:COPYRIGHT ("Xerox Corporation" 1987 1988))
```

FUNCTION INDEX

PRINTER INDEX

COMMAND INDEX

PROPERTY INDEX

DEFINE-TYPE INDEX

DEFINER INDEX