Artificial Intelligence Project--RLE and MIT Computation Center
Memo 39--The New Compiler

by T. Hart -- L. Levin

This memo introduces the brand new LISP 1.5 Compiler designed and programmed by Tim Hart and Mike Levin. It is written entirely in LISP and is the first compiler that has ever compiled itself by being executed interpretively.

The purpose of the LISP Compiler is to replace S-expression definitions of functions with efficient machine language subroutines. A subroutine can be expected to run about 40 times as fast as the interpreter can execute the same function from its S-expression definition. Subroutines typically take 70-80 per cent of the storage required by their corresponding S-expressions.

The compiler as it exists on the standard compiler tape is a machine language program that was obtained by having the S-expression definition of the compiler work on itself through the interpreter.

The compiler is designed so that compiled functions and interpretal functions can be intermixed freely. Suitable declarations allow free variables to be transmitted between the compiled functions and the interpreter Cset constants may also be picked up by compiled functions.

It is possible to compile special forms and to compile functions that call special forms. No special provision is needed for doing this.

Using the Compiler

In order that a function be compiled, it is first necessary that it be defined, that is it must have an EXPR or FEXPR on its property list. This is usually done by a DEFINE or a DEFLIST.

If there are any variable declarations, they must be declared before compiling. This is described under the heading "Free Variables."

To compile any number of functions, one simply punches:

COMPILE ((FN1 FN2...))

The order of compilation is of no significance. It is not necessary to compile inside functions first, or even to have them defined until they are to be used at run time.

It is not necessary to compile all functions being used in a particular run, however running interpreted functions may slow down the run very much.

When the compiler is not to be used any more, it can be removed and converted into free storage by punching:

EXCISE NIL

Free Variables

A variable is bound in a particular function when it occurs in a list of bound variables following the word LAMBDA or PROG. Any variable that is not bound is free.

Example:

```
(LAMBDA (A) (PROG (B)
   S (SETQ B A)
     (COND ((NULL B) (RETURN C)))
     (SETQ C (CONS (CAR A) C))
     (GO S) ))
```

A and B are bound variables, C is a free variable.

When a variable is used free, it must have been bound by a higher level function. If a program is being run interpretively, and a free variable is used without having been bound on a higher level, error diagnostic *A 8* will occur.

If the program is being run compiled, the diagnostic may not occur, and the variable may have value NIL.

All free variables in compiled programs must be declared SPECIAL or COMMON.

If a variable is used free in compiled functions only, and is always bound in other compiled functions then it must be declared SPECIAL. Declaring special variables will not increase program size or decrease speed very much.

Special variables are declared by punching:

SPECIAL ((VAR1 VAR2...))

This declaration must be made before compiling any of the functions that bind or use the free variable.

When a variable is to be used free in a compiled function and bound in an interpreted function or visa-versa it must be declared COMMON. When a variable is declared COMMON, its treatment by compiled functions is identical to its treatment by the interpreter. It may be passed between compiled functions and the interpreter.

Functional Constants

Consider the following S-expression definition of a function

```
(YDOT (LAMBDA (X Y) (MAPLIST X (FUNCTION
    (LAMBDA (J) (CONS (CAR J) Y)) ))))
  ydot[(A B C D);X]=[((A.X) (B.X) (C.X) (D.X))]
```

Following the word FUNCTION is a functional constant. If we consider it as a separate function, it is evident that it contains a bound variable "J", and a free variable "Y". This free variable must be declared SPECIAL or COMMON, even though it is bound in YDOT.

Functional Arguments

MAPLIST can be defined in S-expressions as follows:

```
(MAPLIST (LAMBDA (L FN) (COND
     ((NULL L) NIL)
     (T (CONS (FNL) (MAPLIST (CDR L) FN))) )))
```

The variable FN is used to bind a functional argument. That is, the value of FN is a function definition. This type of variable must be declared COMMON.

Compiler Design

Compilation proceeds in three steps. The output of each one is the input for the next.

PASSONE performs several modifications on the input. Its output is still in the form of S-expression definitions of functions. Among these are the following.

1. All SPECIAL and COMMON variables are flagged for special attention.

2. Function definitions are rewritten so as to use FEXPR and FSUBR functions in the most effective way.

3. Recursive functions that would be more effective if written with iterative loops are rewritten using the PROG feature.

PHASE2 generates assembly language from the S-expression input. The main problems it solves are:

1. Determining the order of computation for nested composition of functions.

2. Locating arguments for calling other functions, and assigning storage to the result that is returned.

3. Determining an efficient sequence of conditional transfers to execute conditional expressions and Boolean predicates.

LAP, the LISP Assembly Program, has been described in another memo.

Compiler Machine Conventions

LISP functions are closed subroutines. They are called by:

TSX FN, 4

with the arguments located in the AC, the MQ, $ARG3, $ARG4,....

They return TRA 1,4 with the value in the AC.

The compliment of IR1 is always the first free cell on the push down list. Every compiled functions calls a subroutine called *MOVE that moves the arguments of the function from AC,MQ etc. to locations 1,1 2,1 etc. Then further space is left on the push down list for temporary storage, and IR1 is decremented to point to a higher location.

Ordinary variables are cells on the push down list. They are not available to other functions.

SPECIAL variables are set up as fixed storage locations. They can be referenced directly by any compiled function. When something is to be stored in a SPECIAL cell, the old value is saved on the push down list and later restored.

COMMON variables are located on an A-list that is passed between interpreter and compiled functions.

LINK

Although it is normal to TSX to functions the first time that they are executed, there will be an STR in this location in the program. This trap is picked up by LINK If LINK finds that there is a subroutine to go to, it replaces the STR with a TSX. If there is no SUBR, then LINK connects with the interpreter, so that if there is an EXPR, it can be interpreted.

CS-TR Scanning Project
Document Control Form

Date : 11/30/95

Report # AIM-39

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☐ Technical Report (TR) ☒ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 4 (8-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☐ Single-sided or
☒ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☒ Other: POOR COPY OF MIMEOGRAPH

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number): _____

Photographs/Tonal Material (by page number): _____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP: (1-4) TITLE PAGE, 2-4	
(5-8) SCANCONTROL, TRGT'S (3)	

Scanning Agent Signoff:
Date Received: 11/30/95 Date Scanned: 12/11/95 Date Returned: 12/14/95

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**