The research reported in this paper was sponsored by the Advanced Research Projects Agency Information Processing Techniques Office and was monitored by the Electronic Systems Division, Air Force Systems Command under contract F1962867C0004, Information Processing Techniques, with the System Development Corporation.

TECH MEMO *a working paper*

System Development Corporation / 2500 Colorado Avenue / Santa Monica, California 90406
Information International Inc. / 11161 Pico Boulevard / Los Angeles, California 90064

TM-3417/525/00

AUTHOR R. E. Long
[illegible]

TECHNICAL Jeff. Barnett
J. Barnett

RELEASE [illegible]
C. Weissman, S.D.C.
[illegible]
D. Anschultz, I.I.I.
for J. I. Schwartz

DATE 4/26/67 PAGE 1 OF 5 PAGES

(page 2 blank)

LISP 2 Storage Management: Paging of Binary Programs

ABSTRACT

This document describes the paging of binary program space by the storage management functions of the LISP 2 system proposed for the IBM S/360 computer. Included are descriptions of the functions required to create, delete, load and unload library files.

1. INTRODUCTION

The LISP 2 system controls the use of binary program space with programs that automatically perform the following functions: (1) load pages when needed, (2) inventory the relative ages of programs, and (3) unload the oldest programs when more space is needed. The age of a program is determined by the number of LISP system interrupts since the program was last executed. The execution of a program automatically restores its age to the initial state. The system attempts to maintain a reasonable balance between the size of binary program space and the size of list-array space.

2. LIBRARY FILES

Programs that are to be paged in LISP 2 are written onto library files. A library file may contain any number of functions and macros. A particular program may be written on several library files. When it is necessary to load any of the members of a library file into core, the entire file must be loaded. Any number of library files may be included in one disc file.

The following is a description of the functions required to create, delete, load, and unload library files.

3. PAGING FUNCTIONS

3.1 DUMPL$LISP

The binary program images (abbreviated BPI's) for functions and macros may be grouped into library files and copied onto the disc by using the function DUMPL$LISP. This action does not cause the BPI's to be deleted from binary program space. A function may be placed in several library files, and one or more library files may be placed onto one binary program disc file. (When grouping functions into library files, consider that loading requires transfer of one entire library file.)

```
DUMPL (lname filename flist)

     lname      =  library file name
     filename   =  disc file name
     flist      =  ( (name . section)*)
     name       =  name of function to be dumped
     section    =  section of function to be dumped
```

DUMPL adds the functions in flist to library file lname, which is placed on binary program disc file filename. The system will automatically open the disc file. If lname is already an established library file within filename, DUMPL loads lname (by calling LOADL), removes any dead BPI's and then dumps all of the library files onto the disc. The value of DUMPL is a list of the names of the functions and macros that were dumped.

3.2 DUMPSEC$LISP

The BPI's for all of the functions and macros in a section may be placed on disc by using DUMPSEC$LISP

DUMPSEC (lname filename section)

This will place all of the functions and macros in a particular section into library file lname on disc file filename. The value of DUMPSEC is a list of the functions and macros that were dumped.

3.3 REMOVEL$LISP

Library files may be removed from the system by calling REMOVEL$LISP (lname), which also loads lname if necessary, by calling LOADL, and returns as its value a list of the alive files in lname.

3.4 UNLOADL$LISP

All of the inactive functions and macros in a library file may be unloaded from binary program space by calling the function UNLOADL$LISP (lname). The value of UNLOADL is NIL.

3.5 UNLOADFN$LISP

UNLOADFN$LISP (name section) will unload the function name$section.

3.6 LOADL$LISP

All of the functions and macros in a library file may be loaded into binary program space from disc by calling the function LOADL$LISP (lname). The value of LOADL is lname.

4. COMPUTATIONAL GERIATRICS OF LISP 2 PROGRAMS

If a function which has been unloaded is called, the LISP 2 system will automatically reload a library file containing it. If this reloading requires more binary program space than is presently available, some programs which are not in use may be unloaded, and if necessary, the garbage collector may be called to increase the size of binary program space.

In order to decide which programs to unload when space is needed, the following interrupt-aging scheme is employed. A function or macro which is in core has one of nine different states or ages. These are, in order from new to old: the untrapped state, and the trapped state with age 0, 1, 2, 3, 4, 5, 6, and 7. Immediately after a function has been loaded, it is in the untrapped state. When an interrupt occurs, all functions and macros that exist in the library files and are presently in core are aged to the next older state (those of age 7 stay in that state).

The interrupt occurs automatically after every nth function return. This number may be varied by changing INTCNT$SYS. The interrupt count is not incremented during garbage collection and an interrupt cannot occur then.

If a function or macro is called, the trap function ITRAP1$SYS automatically returns it to the untrapped state and then the processing continues. Therefore, at the time a function is called, its age is automatically restored to the initial state. (The mechanism of ITRAP1$SYS is similar to that of a trace function. The age of a function is stored in three bits of the function descriptor.)

Automatic unloading of programs may occur during execution of GETHBPS$SYS or during garbage collection, since both of these programs may call UNLBPS$SYS. It unloads all programs in the oldest state; then, if necessary, all of the programs in the next oldest state and so on until enough space is released.

The system for paging of binary programs described above is used in the Q-32 LISP 2 system.