TABLE OF CONTENTS

Section		Page
	TABLE OF CONTENTS	ii
1.	INTRODUCTION	1
2.	DESCRIPTION OF THE LISP/360 SYSTEM TAPE	1
	2.1 File 1 - The Update Program	1
	2.2 File 2 - The Interpreter	1
	2.3 File 3 - LISP Test Cases	1
	2.3.1 PRETTYPRINT	1
	2.3.2 METEOR	2
	2.3.3 ERR	2
3.	PRINTING AND PUNCHING THE SYSTEM TAPE	2
	3.1 Printing the Tape	2
	3.2 Punching the Tape	3
4.	GENERATING A LISP/360 LOAD MODULE	3
	4.1 Compilation and Execution of the Update Program	3
	4.1.1 Format of Update Statements	4
	4.1.2 Memory Requirements of the Interpreter	5
	4.2 Generating and Testing the Load Module	6
	4.3 Prototype Job Control Cards for Generating the Load Module	7
5.	RUNNING LISP PROGRAMS	8
6.	REPORTING ERRORS	8
	REFERENCES	9

1. INTRODUCTION

This paper describes the LISP/360 system tape and procedures for printing and punching its files. Also described are the control cards necessary to generate a load module of the LISP/360 interpreter from the system tape. The generation of the load module includes an update phase where changes and corrections to the system are handled. The LISP/360 system can be generated and run on any IBM System/360 computer with OS/360, the Universal Instruction Set, and 128k bytes of memory.

2. DESCRIPTION OF THE LISP/360 SYSTEM TAPE

The system tape has a standard OS/360 tape label LISP. It contains three files with DSNAMEs LISP1, LISP2, and LISP3 respectively.

2.1 File 1 - The Update Program

The first file contains a PL/1 program for updating the LISP/360 system. This file has a record length of 80 and is unblocked.

2.2 File 2 - The Interpreter

The second file contains the LISP/360 interpreter in OS/360 assembler language. This file has a record length of 80 and a blocking factor of 10.

2.3 File 3 - LISP Test Cases

This file contains several LISP test cases. Included are the LISP programs PRETTYPRINT, METEOR, and ERR. This file has a record length of 80 and is unblocked.

2.3.1 PRETTYPRINT

This program will print the functions that are named as arguments for PRETTYPRINT. The functions must be EXPRs or FEXPRs which have been defined previously with DEFINE or DEFLIST. The functions are printed in such a way as to make them very readable.

2.3.2 METEOR

Particulars about this program are given in an article by Daniel G. Bobrow [1]. METEOR is a LISP-interpreter for a COMIT-like language. This language is very useful for string manipulation and transformation.

2.3.3 ERR

This is a LISP program for the Expression Recognition Routine, ERR, of Brooker and Morris as described in a paper by S. Rosen [2]. This program is described in a paper by L. E. Axsom [3].

3. PRINTING AND PUNCHING THE SYSTEM TAPE

The files may be printed or punched by the OS/360 utility IEBPTPCH as described by the following procedures.

3.1 Printing the Tape

The following prototype job control cards will print a file from the system tape.

```
//TAPELST Δ JØB
//    Δ      EXEC Δ PGM=IEBPTPCH
//SYSPRINT Δ DD Δ SYSØUT=A
//SYSUT1 Δ DD Δ VØLUME=SER=LISP,DSNAME=xxxxx,UNIT=2400,                C
//              LABEL=(y,SL),DISP=(ØLD,KEEP)
//SYSUT2 Δ DD Δ SYSØUT=A
//SYSIN Δ DD Δ *
        PRINT Δ TYPØRG=PS,MAXFLDS=1
        RECORD Δ FIELD=(80,1,,20)
/*
```

(Note: The symbol Δ indicates one or more blanks.)

- The first file will be printed when LISP1 is substituted for xxxxx and 1 for y.
- The second file will be printed when LISP2 is substituted for xxxxx and 2 for y.

- The third file will be printed when LISP3 is substituted for xxxxx and 3 for y.

3.2 Punching the Tape

The following prototype job control cards will punch a file from the system tape.

```
//TAPEUN Δ JØB
//   Δ      EXEC  Δ  PGM=IEBPTPCH
//SYSUT1 Δ DD Δ VØLUME=SER=LISP,DSNAME=xxxxx,UNIT=2400,                C
//                LABEL=(y,SL),DISP=(ØLD,KEEP)
//SYSUT2 Δ DD Δ UNIT=SYSCP
//SYSPRINT Δ DD Δ SYSØUT=A
//SYSIN Δ DD Δ *
        PUNCH Δ TYPØRG=PS
/*
```

- The first file will be punched when LISP1 is substituted for xxxxx and 1 for y.
- The second file will be punched when LISP2 is substituted for xxxxx and 2 for y.
- The third file will be punched when LISP3 is substituted for xxxxx and 3 for y.

4. GENERATING A LISP/360 LOAD MODULE

The generation of a LISP/360 load module requires a job with two job steps. The following sections describe these two steps.

4.1 Compilation and Execution of the Update Program

The PL/1 program in the first file on the system tape is compiled and executed. Any update cards included as data in the first job step are used together with the second file on the system tape to produce a data set, ASMINP, that is passed to the next job step. This data set may be on tape or disk.

The message SUBSCRIPT RANGE EXCEEDED SEE IMPLEMENTATION GUIDE may occur in this job step if there were too many update cards for the update program. More than 800 update cards are necessary to force this message. Update cards for corrections and extensions to the LISP/360 interpreter will be mailed periodically to those installations that requested the system under Plan 1. These cards will either replace previous update cards or be additions to previous update cards. A new system tape will be distributed to installations under Plan 1 if extensions and corrections to the system should require more than 800 update cards.

A renumbered file with corrections and extensions included is necessary before more updates can be processed. This renumbered file must be made by the installation if their own corrections and extensions should exceed the limit imposed by the update program. Update cards distributed by the Stanford Computation Center will, however, always refer to the system tape distributed by the Stanford Computation Center.

4.1.1 Format of Update Statements

Updates are made through the update statements INSERT, REPLACE, and DELETE. The numbers m and n indicated below are 8 digit numbers with leading zeros if necessary. m an n refer to the 8 digit number in columns 73 - 80 in the card images on the system tape. m must be less than or equal to n.

```
col. 10   col. 20
      ↓         ↓
     INSERT    m
```

The INSERT statement causes the cards which follow it to be inserted after card image m. Cards are inserted until the next update statement or end-of-file is encountered.

```
col. 10   col. 20   col. 30
      ↓         ↓         ↓
     REPLACE   m         n
```

REPLACE causes card images m through n to be replaced by the cards that follow REPLACE up to the next update statement or until an end-of-file is encountered. Replacement need not be one for one. If a single card image is to be replaced, only m need appear in the REPLACE card.

```
col. 10    col. 20    col. 30
  ↓          ↓          ↓
DELETE       m          n
```

DELETE causes deletion of card images m through n. If a single card is to be deleted, only m need appear in the DELETE card.

The update program prints out a log of all corrections made. All inserted cards are printed with the word INSERT to the right of the card image. All deleted cards are printed with the word DELET to the right of the card image. All inserted cards in the output data set are marked with asterisks in columns 73-80 by the update program.

4.1.2 Memory Requirements for the Interpreter

The storage set aside for stack, freecellstorage and binary program space is set by the following card images at assembly time:

```
                                                    cols. 73-80
STACKSIZ    EQU     4000        WORDS                00001130
STORESIZ    EQU     200000      DOUBLEWORDS          00001140
BPSSIZE     EQU     0           WORDS                00001150
```

This means that 176k bytes of memory are set aside for stack and freecellstorage by these card images on the system tape. No memory is set aside for binary program. The interpreter itself and the initial object list requires 42080 bytes. The memory required by the interpreter is therefore 218080 bytes.

Installations with less than 220k bytes available to run the LISP/360 interpreter must change the parameters STACKSIZ and STORESIZ during the update step.

The following update cards should be included in the update step if only 80k bytes are available to the LISP/360 interpreter:

```
    col. 10    col. 20    col. 30
      ↓          ↓          ↓
      REPLACE    00001130   00001140
STACKSIZ  EQU      500        WØRDS
STØRESIZ  EQU      4500       DØUBLEWØRDS
```

The following update cards should be included in the update step if only 200k bytes is available to the LISP/360 interpreter:

```
    col. 10   col. 20   col. 30
          ↓         ↓         ↓
          REPLACE   00001130  00001140
STACKSIZ  EQU       3000      WØRDS
STØRESIZ  EQU       18250     DØUBLEWØRDS
```

Installations with more than 220k bytes available to run the LISP/360 interpreter may increase STACKSIZ and STORESIZ so that all available memory is utilized by the interpreter. The parameter BPSSIZE must be increased when LAP and the compiler are released for use.

4.2 Generating and Testing the Load Module

The data set ASMINP produced by the previous job step is used as input to the assembler in this job step. The object module produced by the assembler is link edited and placed in the partitioned data set LISP as the member LISP. This load module is then used to run the LISP test cases in the third file on the system tape.

4.3 Prototype Job Control Cards For Generating the Load Module

NOTE: The prototype job control cards given below assume that the procedures PL1LFCLG and ASMFCLG are catalogued. It is also assumed that PL1LFCLG has the job steps PL1L, LKED, and GØ, and that ASMFCLG has the job steps ASM, LKED, and GØ.

```
//LISPUPD Δ JØB
//STEP1 Δ EXEC Δ PL1LFCLG
//PL1L.SYSIN Δ DD Δ DSNAME=LISP1,VØLUME=SER=LISP,UNIT=2400,          C
//              LABEL=(1,SL),DISP=(ØLD,PASS)
//GØ.ØUT Δ DD Δ DSNAME=ASMINP,VØLUME=SER=xxxxxxxx,UNIT=yyyy,        C
//              DISP=(NEW,PASS),SPACE=(TRK,(5,5)),                  C
//              DCB=(RECFM=FB,BLKSIZE=800,LRECL=80)
//GØ.IN Δ DD Δ DSNAME=LISP2,VØLUME=SER=LISP,UNIT=2400,              C
//              LABEL=(2,SL),DISP=(ØLD,PASS)
//GØ.SYSPRINT Δ DD Δ SYSØUT=A
//GØ.SYSIN Δ DD Δ *
  update cards
    (if any)
     go here
/*
//STEP2 Δ EXEC Δ ASMFCLG
//ASM.SYSIN Δ DD Δ DSNAME=ASMINP,VØLUME=SER=xxxxxxxx,UNIT=yyyy,     C
//              DISP=(ØLD,DELETE)
//LKED.SYSLMØD Δ DD Δ DSNAME=LISP(LISP),VØLUME=SER=zzzzzzzz,        C
//              UNIT=wwww,DISP=(NEW,KEEP),SPACE=(TRK,(5,5,1))
//GØ.LISPIN Δ DD Δ DSNAME=LISP3,VØLUME=SER=LISP,UNIT=2400,          C
//              LABEL=(3,SL),DISP=(ØLD,KEEP)
//GØ.LISPØUT Δ DD Δ SYSØUT=A
/*
```

The dummy symbols in the above cards are:

xxxxxxxx	the volume on which the data set ASMINP is allocated space. ASMINP is a temporary data set that is deleted in the second job step.
yyyy	is the unit on which the volume xxxxxxxx is mounted.
zzzzzzzz	the volume on which the LISP/360 load module will be generated.
wwww	is the unit on which the volume xxxxxxxx is mounted.

The DD cards pertaining to the data set ASMINP will have to be changed slightly before this data set can go on tape. An appropriately labeled tape is also required.

5. RUNNING LISP PROGRAMS

LISP programs punched on cards using the 029 keypunch can be run with the following job control cards:

```
//LISPTST Δ JØB
//JØBLIB Δ DD Δ DSNAME=LISP,VØLUME=SER=zzzzzzzz,UNIT=wwww,DISP=ØLD
//  Δ  EXEC Δ PGM=LISP
//LISPØUT Δ DD Δ SYSØUT=A
//LISPIN Δ DD Δ *
   [ LISP programs
     go here ]
/*
```

LISP/360 can read programs punched on an 026 keypunch when PARM=BCD is added to the EXEC card as follows:

```
// Δ EXEC  Δ  PGM=LISP,PARM=BCD
```

6. REPORTING ERRORS

Problems in the implementation of Stanford LISP/360 or errors in the use of it should be reported to

Systems Documentation Office
Room 185, Pine Hall
Stanford Computation Center
Stanford University
Stanford, California 94305
Area code 415, 321-2300, x.4877

Corrections and extensions to the interpreter will be distributed regularly to those installations that have specified Plan 1.

REFERENCES

[1] Bobrow, D. G. - "METEOR: A LISP Interpreter for String Transformations." A paper in [4].

[2] Rosen, Saul - "A Compiler-Building System Developed by Brooker and Morris." A paper in [5]

[3] Axsom, L. E. - "An Expression Recognition Routine in LISP 1.5." A paper in [5].

[4] Berkeley, E. C. and Bobrow, D. G. (Editors) - The Programming Language LISP: Its Operation and Applications. MIT Press (1966).

[5] Rosen, Saul (Editor) - Programming Systems and Languages. McGraw-Hill Book Company (1967).