



DECUS

PROGRAM LIBRARY

DECUS NO.

85

TITLE The LISP Implementation for the PDP-1 Computer

AUTHOR

L. Peter Deutsch and Edmund C. Berkeley

COMPANY

DATE

March 1964

FORMAT

LISP

Appendix 5

The LISP Implementation for the PDP-1 Computer

L. Peter Deutsch and Edmund C. Berkeley

TABLE OF CONTENTS

Part I

1. Introduction
2. Functions and Properties Included in Basic PDP-1 LISP
3. Use of these Functions and Suggested Test Sequences
4. Auxiliary Functions which May Be Defined with LISP Expressions
5. Some Additional Functions for Basic PDP-1 LISP
6. Input and Output
7. Operation of the System
8. Error Diagnostics
9. Some Remarks

Part II

1. Macro Symbolic Program for Basic PDP-1 LISP
2. Alphabetic Listing of Defined Macro Symbols
3. Numeric Listing of the Defined Macro Symbols
4. Mnemonic Key or Derivation of Symbols

Part I

1. Introduction

In October 1963 a system for implementing LISP on the PDP-1 computer was finished by L. Peter Deutsch. This system was further improved in March 1964 by adding:

- variable length of push-down list;
- variable quantity of combined storage;
- optional machine language subroutines;

and is here called Basic PDP-1 LISP. It uses a minimum of some 2000 (decimal) registers out of 4096 registers in a one-core PDP-1 computer; it may use 16,361 registers in a four-core PDP-1 computer.

Basic PDP-1 LISP is presented in considerable detail in this appendix for the following reasons:

- the structure of a system for programming LISP on any computer is thereby revealed;
- if changes are to be implemented, they can be easily linked with the existing system.

In a one-core PDP-1 computer with 4096 registers, as many as 4070 registers may be assigned to regular LISP, and only 23 reserved for the read-in routine (namely, from 7751 to 7777, octal).

With the system described here, additional LISP functions can be defined and included in the system and later used when desired. Or if desired, additional functions can be programmed in machine language and these can be inserted compatibly with the system.

Punched tapes for placing this LISP system on the PDP-1 computer are available through DECUS, the Digital Equipment Corporation Users Organization, Maynard, Mass.

In the following, it is assumed that the reader has a fairly good working knowledge of: (1) LISP (which may be obtained from the "LISP 1.5 Programmer's Manual," 1962); (2) the machine language codes for the PDP-1 computer (which may be obtained from the computer manual supplied by Digital Equipment Corporation); and (3) the program assembly language MACRO, in which the sym-

bolic tapes are written (a description may be obtained in two manuals published by Digital Equipment Corporation).

2. Functions and Properties included in Basic PDP-1 LISP

The functions and properties included in Basic PDP-1 LISP are shown in Table 1. These functions and properties together constitute a basic subset of the functions and properties of the LISP interpreter for the IBM 7090, as stated in the LISP 1.5 Programmer's Manual.

In order to obtain other LISP functions and properties as may be desired for any particular purpose, see Sections 4 and 5 below.

Table 1
FUNCTIONS AND PROPERTIES OF BASIC PDP-1 LISP

A. Functions Identical with the Corresponding IBM 7090 LISP Functions

ATOM	LIST	PROG
CAR	LOGAND	QUOTE
CDR	LOGOR	READ
COND	MINUS	RETURN
CONS	NULL	RPLACA
EVAL	NUMBERP	RPLACD
GENSYM	PLUS	SASSOC
GO	PRINT	SETQ
		TERPRI

B. Functions Somewhat Different from the Corresponding 7090 Functions

EQ	This works both on atoms and on numbers
GREATERP	This tests for X greater than Y, not for X greater than or equal to Y.
STOP	This is equivalent to PAUSE in 7090 LISP. It takes a numerical argument which appears in the accumulator when the computer halts.
PRIN1 X	This prints the <u>atom</u> X without the extra space at the end. Its value is NIL.

C. Functions Which Have No Analog in 7090 Functions

XEQ This provides for putting into storage a named machine language subroutine, which can be referred to and used by the PDP-1 LISP interpreter. It also provides for executing single specified machine language instructions.

The SUBR (XEQ C A I) executes the machine language instruction C, with A in the accumulator and I in the in-out register; and returns a value in the form of (a i P) where a is the new value of the accumulator after execution, i is the new value of the in-out register after execution, and P is T if the instruction skipped, and NIL if the instruction did not skip.

LOC X This gives the machine register in which the atom or list X begins; its value is the location.

Of the foregoing functions, COND, LIST, PROG, SETQ, PLUS, TIMES, LOGAND, LOGOR, and QUOTE are FSUBRs and the remainder are SUBRs.

D. The following special form is available and is identical with the corresponding form in 7090 LISP:

LAMBDA

E. The following permanent objects exist in the Basic PDP-1 LISP system:

OBLIST the current list of atomic symbols
NIL F has been replaced by NIL
T
EXPR
SUBR
FEXPR
FSUBR
APVAL

F. Miscellaneous

The print names of atomic symbols are not part of property lists. A quick examination of listings of the system will show exactly where the print names are.

Doing a CDR of an atom is permissible and will get the atom's property list. Doing a CAR of an atom may very easily wreck the system.

QUOTE should be used in place of 7090 FUNCTION. This may re-

quire a bit of extra care in defining functions with functional arguments.

It is advisable to use PROG to avoid recursion wherever possible, even though it may take more space.

3. Use of these Functions and Suggested Test Sequences

How to use these functions is briefly explained here.

As soon as the basic PDP-1 LISP system is read into the computer, control stops at register 4. Turn up sense switch 5 for typewriter input; press CONTINUE; and the system enters a waiting loop which causes lamps to light in the program counter, looking like 1335. At this point, the LISP system is ready for manual typewriter input. As soon as the operator types, for example:

(CAR (QUOTE (A B C D)))

together with a final space at the end of the last right parenthesis, the computer takes control of the typewriter, impulses a carriage return, and then types out:

A

which of course is the correct answer. Similarly, for the other suggested test sequences in Table 2 below.

Table 2
SUGGESTED TEST SEQUENCES

<u>Input</u>	<u>Response</u>
(CAR (QUOTE (A B C D)))	A
(CDR (QUOTE (A B C D)))	(B C D)
OBLIST	The interpreter will type out a complete list of the atomic symbols stored within it.
(LIST (QUOTE (A B C D)))	((A B C D))

NIL	NIL
(CDR NIL)	(APVAL NIL)
(CAR (QUOTE (T.NIL)))	T
(CONS (ATOM (CDR T)) (LIST (GENSYM) (GENSYM)))	(NIL GOOOO1 GOOOO2)
(COND (EQ T NIL) (STOP 1)) (T (EQ (PLUS 1 1) 2)))	T
(PROG (U) (PRINT NIL) (TERPRI) (PRINT T) (SETQ U T) (RETURN U))	NIL T T
(RPLACD (QUOTE CAAR) (QUOTE (EXPR (LAMBDA (X) (CAR (CAR X)))))) (CAAR (QUOTE ((A)))))	CAAR A
(STOP 2)	Computer stops and puts 2 in the accumulator.
(PRIN1 (QUOTE CAR))	CAR, with no punctua- tion before or after; the value of PRIN1 is NIL.
(PRINT X)	Prints out the value of X; the value of (PRINT X) is X.
(TERPRI)	Prints a carriage re- turn; the value of (TERPRI) is NIL.
(LOC NIL)	2651; this is the regis- ter where the NIL atom starts.
(LOC (QUOTE COND))	2725; this is the regis- ter where the COND atom starts.
(LOGAND 6 7 3)	2
(LOGOR 12 3 15)	17

(RPLACA (QUOTE (NIL X Y))
(QUOTE (A B)))

((A B) X Y)

Suppose the computer contains DDT — DDT is short for "Digital Equipment Corp. Debugging Tape"; its starting register is 6000, and in one of its customary forms it uses registers 5540 to 7750. Then, if the highest storage register of LISP is below 5540, the instruction:

(XEQ 606000 0 0)

transfers control to DDT, and puts zero in the accumulator and in the in-out register.

If there is the following subroutine stored in the computer:

5500	dzm 5507
5501	idx 5507
5502	lac 5507
5503	dpy'
5504	sma
5505	jmp 5501
5506	jmp 2241
5507	(being used for storage)

and LISP is below 5500, then:

(XEQ 605500 0 0)

Will cause a horizontal line to be drawn on the scope from the origin to the x-axis positive limit, and then control will be returned to LISP.
NIL will be typed out.
2241 is the register called "prx" in the macro symbolic.

4. Auxiliary Functions Which May Be Defined with LISP Expressions

Any of the functions listed below in Table 3 can be put into the system at will, as follows: Prepare a punched tape listing of it. Insert tape into the reader. Turn on the reader. Turn down Sense Switch 5. Thereupon the computer will read in the

tape. The typewriter, when the reading in is accomplished, will type back the name of the inserted function.

Many other functions besides those listed in Table 3 may be inserted.

Table 3

AUXILIARY LISP FUNCTIONS

<u>ABSOLUTE VALUE</u>	
(RPLACD (QUOTE ABSVAL) (QUOTE (EXPR (LAMBDA (X) (COND ((GREATERP 0 X) (MINUS X)) (T X)))))))	
<u>AND</u>	
(RPLACD (QUOTE AND) (QUOTE (FEPR (LAMEDA (X A) (PROG NIL N (COND ((NULL X) (RETURN T)) ((NULL (EVAL (CAR X) A)) (RETURN NIL)))) (SETQ X (CDR X)) (GO N)))))))	
<u>ASSOC</u>	
(RPLACD (QUOTE ASSOC) (QUOTE (EXPR (LAMBDA (X Y) (COND ((EQUAL (CAAR Y) X) (CAR Y)) (T (ASSOC X (CDR Y))))))))	
<u>CAAR</u>	
(RPLACD (QUOTE CAAR) (QUOTE (EXPR (LAMEDA (X) (CAR (CAR X)))))))	
<u>CADR</u>	
(RPLACD (QUOTE CADR) (QUOTE (EXPR (LAMEDA (X) (CAR (CDR X)))))))	
<u>CDDAR</u>	
(RPLACD (QUOTE CDAR) (QUOTE (EXPR (LAMEDA (X) (CDR (CAR X)))))))	
<u>CDDR</u>	
(RPLACD (QUOTE CDDR) (QUOTE (EXPR (LAMEDA (X) (CDR (CDR X)))))))	
<u>CSET</u>	
(RPLACD (QUOTE CSET) (QUOTE (EXPR (LAMEDA (X Y) (RPLACD X (LIST (QUOTE APVAL) Y)))))))	

CSETQ
(RPLACD (QUOTE CSETQ) (QUOTE (FEXPR (LAMEDA (X A) (CSET (CAR X)) (EVAL (CADR X) A))))))

DEX

(RPLACD (QUOTE DEX) (QUOTE (FEXPR (LAMEDA (X A) (RPLACD (CAR X) (CONS (QUOTE EXPR) (CDR X)))))))

DFX

(RPLACD (QUOTE DFX) (QUOTE (FEXPR (LAMEDA (X A) (RPLACD (CAR X) (CONS (QUOTE FEXPR) (CDR X)))))))

DIFFLIST

(RPLACD (QUOTE DIFFLIST) (QUOTE (EXPR (LAMEDA (A X) (COND ((NULL X) NIL) ((EQUAL A (CAR X)) (DIFFLIST A (CDR X)))) (T (CONS (CAR X) (DIFFLIST A (CDR X))))))))

DOUBLE

(RPLACD (QUOTE DOUBLE) (QUOTE (EXPR (LAMEDA (X) (PLUS X X))))))

EQUAL

(RPLACD (QUOTE EQUAL) (QUOTE (EXPR (LAMEDA (X Y) (COND ((ATOM X) (Eq X Y)) ((ATOM Y) NIL) ((EQUAL (CAR X) (CAR Y)) (EQUAL (CDR X) (CDR Y))) (T NIL))))))

GREATEST COMMON DIVISOR

(RPLACD (QUOTE GCD) (QUOTE (EXPR (LAMEDA (X Y) (COND ((GREATERP X Y) (GCD Y X)) ((ZEROP (REM Y X)) X) (T (GCD (REM Y X) X)))))))

LAST

(RPLACD (QUOTE LAST) (QUOTE (EXPR (LAMEDA (L) (COND ((NULL L) NIL) ((NULL (CDR L)) (CAR L)) (T (LAST (CDR L))))))))

LENGTH using Program Feature

(RPLACD (QUOTE LENGTH) (QUOTE (EXPR (LAMEDA (L) (PROG (U V) (SETQ V 0) (SETQ U L) A (COND ((NULL U) (RETURN V))) (SETQ U (CDR U)) (SETQ V (PLUS 1 V)) (GO A))))))

LENGTH using Recursion

(RPLACD (QUOTE LENGTHR) (QUOTE (EXPR (LAMEDA (L) (COND ((NULL L) 0) (T (PLUS 1 (LENGTHR (CDR L))))))))

MAPLIST using Recursion

(RPLACD (QUOTE MAPLIST) (QUOTE (EXPR (LAMEDA (X A) (COND ((NULL X) NIL) (T (CONS (A X) (MAPLIST (CDR X) A))))))))

MAPLIST using Program Feature

(RPLACD (QUOTE MAPLIST) (QUOTE (FEXPR (LAMEDA (X A) (PROG (V M R)

```

(SETQ R (SETQ M (LIST (EVAL (CADR X) A)))) (SETQ V (EVAL
(CAR X) A)) P (COND ((NULL V) (RETURN (CDR R))) (SETQ M
(CDR (RPLACD M (LIST (EVAL (LIST (CAR R) (LIST (QUOTE QUOTE)
V)) A)))) (SETQ V (CDR V)) (GO P)))))

MEMBER
(RPLACD (QUOTE MEMBER) (QUOTE (EXPR (LAMBDA (A X) (COND ((NULL X)
NIL) ((EQ A (CAR X)) T) (T (MEMBER A (CDR X)))))))

MINIMUM
(RPLACD (QUOTE MIN) (QUOTE (EXPR (LAMBDA (L) (COND ((NULL L) NIL)
((NULL (CDR L)) (CAR L)) (T (SMALLER (CAR L) (MIN (CDR L)
)))))))))

MOT
(RPLACD (QUOTE NOT) (QUOTE (EXPR NULL)))

OR
(RPLACD (QUOTE OR) (QUOTE (FEEXPR (LAMBDA (X A) (PROG NIL N (COND
((NULL X) (RETURN NIL)) ((EVAL (CAR X) A) (RETURN T)))
(SETQ X (CDR X)) (GO N))))))

PAIR
(RPLACD (QUOTE PAIR) (QUOTE (EXPR (LAMBDA (X Y) (PROG (U V M)
(SETQ U X) (SETQ V Y) (SETQ M NIL) K (COND ((NULL U) (COND
((NULL V) (RETURN M))) (SETQ M (CONS (CONS (CAR U) (CAR V))
M)) (SETQ U (CDR U)) (SETQ V (CDR V)) (GO K)))))))

PATRLIS
(RPLACD (QUOTE PATRLIS) (QUOTE (EXPR (LAMBDA (X Y A) (COND ((NULL
X) A) (T (CONS (CONS (CAR X) (CAR Y)) (PATRLIS (CDR X)
(CDR Y) A)))))))))

PDEF (Print and Punch Definition)
(RPLACD (QUOTE PDEF) (QUOTE (FEEXPR (LAMBDA (X A) (LIST (QUOTE
PFLACD) (LIST (QUOTE QUOTE (CAR X)) (LIST (QUOTE QUOTE) (CDR
(CAR X))))))))))

QUOTIENT using Program Feature
(RPLACD (QUOTE QUOTIENT) (QUOTE (EXPR (LAMBDA (Q D) (FROG (U V)
(SETQ V O) (SETQ U Q) A (COND ((GREATERP D U) (RETURN V))
(SETQ U (PLUS U (MINUS D))) (SETQ V (PLUS 1 V)) (GO A)))))))

QUOTIENT using Recursion
(RPLACD (QUOTE QUOTIENTR) (QUOTE (EXPR (LAMBDA (Y X) (COND ((  

GREATERP X Y) O) ((EQ X Y) 1) ((GREATERP D U) (RETURN V))
(QUOTIENTR (PLUS Y (MINUS X)) X)))))))

```

REMAINDER

(RPLACD (QUOTE REM) (QUOTE (EXPR (LAMBDA (Y X) (COND ((EQUAL Y X) 0) ((GREATERP X Y) Y) (T (REM (PLUS Y (MINUS X)) X)))))))

REVERSE (Defined Recursively with Auxiliary Function)

(RPLACD (QUOTE R1) (QUOTE (EXPR (LAMBDA (M L) (COND ((NULL L) M) (T (R1 (CONS (CAR L) M) (CDR L))))))))

(RPLACD (QUOTE REVERSE) (QUOTE (EXPR (LAMBDA (L) (R1 NIL L))))))

REVERSE using Program Feature

(RPLACD (QUOTE REVERSE) (QUOTE (EXPR (LAMBDA (M) (PROG (U V) (SETQ U M) K (COND ((NULL U) (RETURN V))) (SETQ V (CONS (CAR U) V)) (SETQ U (CDR U)) (GO K)))))))

SEQUENCE

(RPLACD (QUOTE SEQUENCE) (QUOTE (EXPR (LAMBDA (L) (PROG (U V W) (SETQ U L) (SETQ V (MIN L)) (SETQ W NIL) A (COND ((NULL U) (RETURN W))) (SETQ V (MIN U)) (SETQ U (DIFFLIST V U)) (SETQ W (APPEND W (LIST V))) (GO A)))))))

SMALLER

(RPLACD (QUOTE SMALLER) (QUOTE (EXPR (LAMBDA (X Y) (COND ((GREATERP X Y) Y) (T X)))))))

SUB2

(RPLACD (QUOTE SUB2) (QUOTE (EXPR (LAMBDA (A Z) (COND ((NULL A) Z) ((EQ (CAAR A) Z) (CDAR A)) (T (SUB2 (CDR A) Z)))))))

SUBLIS

(RPLACD (QUOTE SUBLIS) (QUOTE (EXPR (LAMBDA (A Y) (COND ((ATOM Y) (SUB2 A Y)) (T (CONS (SUBLIS A (CAR Y)) (SUBLIS A (CDR Y))))))))

SUBST

(RPLACD (QUOTE SUBST) (QUOTE (EXPR (LAMBDA (X Y Z) (COND ((EQUAL Y Z) X) ((ATOM Z) Z) (T (CONS (SUBST X Y (CAR Z)) (SUBST X Y (CDR Z))))))))

TIMES using Recursion

(RPLACD (QUOTE TIMES) (QUOTE (EXPR (LAMBDA (N M) (COND ((EQUAL N 1) M) (T (PLUS M (TIMES M (PLUS N (MINUS 1)))))))))))

TIMES using Program Feature
(RPLACD (QUOTE TIMES) (QUOTE (EXPR (LAMBDA (X N) (PROG (U V)
 (SETQ V 0) (SETQ U 0) A (COND ((EQ V N) (RETURN U))) (SETQ U
 (PLUS X U)) (SETQ V (PLUS V 1)) (GO A))))))

UNION
(RPLACD (QUOTE UNION) (QUOTE (EXPR (LAMBDA (X Y) (COND ((NULL X)
 Y) ((MEMBER (CAR X) Y) (UNION (CDR X) Y)) (T (CONS (CAR X)
 (UNION (CDR X) Y))))))))

ZEROP
(RPLACD (QUOTE ZEROP) (QUOTE (EXPR (LAMBDA (X) (COND ((EQUAL X
 0) T) (T NIL))))))

5. Some Additional Functions for Basic PDP-1 LISP

In order to remove symbols from the OBLIST, and reuse the storage capacity that they previously occupied, we use:

(RPLACD (QUOTE XSY) (QUOTE (EXPR (LAMBDA (X) (PROG (Y) (SETQ Y
 OBLIST) A (COND ((NULL (CDR Y)) (RETURN NIL)) ((EQ X (CAR
 (CDR Y))) (RETURN (RPLACD Y (CDR (CDR Y)))))) (SETQ Y (CDR Y))
 (GO A))))))

(RPLACD (QUOTE REMOVE) (QUOTE (EXPR (LAMBDA (X Y) (PROG NIL A
 (COND ((NULL X) (RETURN OBLIST)) (XSY (CAR X)) (SETQ X
 (CDR X)) (GO A))))))

XSY stands for "expunge symbol".

REMOVE is used as follows: Suppose we have a case where the OBLIST starts for example with G F OBLITT Y X ATOM CAR CDR COND CONS and we wish to delete F OBLITT Y. We put in: (REMOVE OBLITT F Y), and the computer response is:

G X ATOM CAR CDR COND

In this way, both accidentally mistyped expressions and symbols no longer needed in the LISP system can be removed from storage, and from any recollection within the LISP system. (Note: REMOVE will not operate on the first expression in the OBLIST, but only on the second and later expressions.)

In order to put in machine-language subroutines, outside of the storage used by LISP, name them, use them, and return from them to LISP, we use:

```
(RPLACD (QUOTE DEPOSIT) (QUOTE (EXPR (LAMBDA (X Y) (PROG NIL A  
      (COND ((NULL X) (RETURN Y))) (XEQ (PLUS 240000 Y) (CAR X) 0)  
      (SETQ X (CDR X)) (SETQ Y (PLUS 1 Y)) (GO A))))))  
  
(RPLACD (QUOTE PUTSUBR) (QUOTE (EXPR (LAMBDA (N X Y) (PROG NIL  
      (RPLACD N (LIST (QUOTE SUBR) (PLUS 160000 Y))) (RETURN  
      (DEPOSIT X Y)))))))  
  
(RPLACD (QUOTE DEFSUBR) (QUOTE (EXPR (LAMBDA (N X)  
      (RPLACD N (LIST (QUOTE SUBR) (PLUS 160000 X)))))))
```

The EXPR (DEPOSIT X A) deposits the list of numbers X starting at location A; its value is the first register beyond the list.

The EXPR (PUTSUBR N X A) performs (DEPOSIT X A), and then sets up N (name) as a SUBR starting at A.

An example (if LISP storage stops at 5477) is:

```
(PUTSUBR (QUOTE SHOWLINE) (LIST 345507 445507 205507 730007  
      640400 605501 602241) 5500)
```

This inserts the line-display program mentioned above into the computer starting at register 5500 and makes it accessible to LISP with the name SHOWLINE.

The EXPR (DEFSUBR N X) accepts an existing, inserted, machine-language subroutine starting at register X, gives it the name N, and makes it accessible to LISP with the name N. For example, the line-display program mentioned above, if already in the computer, could be named and called with:

```
(DEFSUBR (QUOTE SHOWLINE) 5500)
```

The last command in the subroutine, instead of 602241, should be either 600004, if LISP is to return to the starting address 4, or 600005, if LISP is to continue to the waiting loop.

If the A-LIST is wanted, establish GETALIST with:

```
(RPLACD (QUOTE GETALIST) (QUOTE (FEXPR (LAMBDA (X Y) Y))))
```

and then use:

```
(PRINT (GETALIST))
```

6. Input and Output

Input comes from the typewriter if sense switch 5 is up and from the tape reader otherwise. Output is normally on the typewriter; however, SS 3 up causes punching (with correct parity) and SS 6 up independently suppresses typeout.

Each S-expression typed in will be evaluated and its value printed out. Unlike 7090 LISP, arguments of functions are also evaluated on the top level; for example, to evaluate

cons [A;B]

it is necessary to write

(CONS (QUOTE A) (QUOTE B))

In preparing input:

Tab, space, and comma are equivalent;
Carriage return is ignored;

Backspace causes deletion of everything typed since the last control character (parenthesis, space/tab/comma, or period);

An extra space must be typed to terminate the entire expression;

Upper and lower case shifts will be noted but not necessarily inserted into the symbol at that point (for example, the sequence u.c., l.c., u.c., A, space, produces a symbol with print name u.c., A, l.c.);

Alphabetic characters should regularly and generally be in lower case; and basic functions, (such as CAR, CDR,) contrary to their representation throughout this report, are in PDP-1 LISP actually stored in lower-case symbols (such as car, cdr); and then taken in to the system and put out by the system as lower-case symbols;

It is very advisable to stick to "printout" format for all input since the READ routine is not guaranteed to work on any other form, although it may;

Hyphen, "-", is a letter and does not negate a following number;

All numbers are octal integers; to input the number -1 it is necessary to type 777776;

There is no limit on the length of a print name;

The character overbar "--" or vertical bar "|" will cause the next character to be inserted in the print name and considered a letter, regardless of what it actually is (the "--" or "|" itself does not appear

in the print name): thus atoms may be generated for output formatting purposes with names such as "tab" or "space".

In producing the output:

A carriage return is automatically generated after any 100(octal) characters not containing a carriage return;

Unlike the 7090 LISP output, no spaces are provided before and after the "." of concatenation (since there are no floating-point numbers to be concerned with).

7. Operation of the System

First, zero core, to avoid unnecessary difficulties.

Second, put the binary tape in the reader, and press READIN. Do nothing until the tape stops. Almost all of the tape will read in; and the machine will come to a halt. If you wish 7701 to be the highest register of free storage, and 300 to be the length of the push-down list, press READIN once more. The machine will stop at address 4. Turn up Sense Switch 5 (to control from the typewriter). Press CONTINUE.

If you wish to select the highest register of free storage, when the machine stops for the first time, with memory address at 0004, put the number of the highest register of free storage (recommended, 5000 to 7750; possible but not recommended, 4000 to 4777) in the Test Word switches and press CONTINUE. Then put the length of the push down list (suggested 200 to 400) in the Test Word switches, and press CONTINUE. The machine will go to address 4. Turn up Sense Switch 5, and press CONTINUE. The LISP system should be ready for use.

If the tape stops at an improper place, pull the tape back a block, check for missing holes, and CONTINUE. When the tape stops at 4, CONTINUEing begins the READ-EVAL-PRINT cycle. STARTing at 4 at any time and CONTINUEing is safe; indeed, it is the only way to annull most typing errors.

If the system "drops dead", the normal recourse is to start over.

Following is the assignment of the sense switches and the program flags:

SS	1	Idiot trace
	2	-
	3	Punch out
	4	-
	5	Type in
	6	No typeout
PF	1	Used for type-in
	2	Zero suppress in octal print
	3	-
	4	-
	5	Letter in symbol
	6	Off in error printout

8. Error Diagnostics

Error halts cause identification of the error and typing of the error code in red on the typewriter, regardless of the settings of Sense Switches 3 and 6; an error usually sends the system to address 4. The list of error indications follows:

icd	Illegal <u>COND</u> ; returns value NIL and continues.
uss	Unbound <u>symbol</u> in <u>SETQ</u> ; returns NIL and continues.
tma	Too <u>many</u> <u>arguments</u> for a SUBR (more than 3); ignores extra arguments and proceeds.
uas	Unbound <u>atomic symbol</u> (followed by the form currently being evaluated).
ilp	Illegal parity; halts with character in accumulator. CONTINUE ignores character, but SS 5 may be turned up, and typing used to provide a replacement if desired.
lts	LAMBDA variable list <u>too short</u> .
ats	Argument list (paired with LAMBDA list) <u>too short</u> .
sce	Storage <u>capacity exceeded</u> . CONTINUEing is not advisable, as it will probably call the same error again in short order, unless one promptly deletes several atoms having lengthy definitions from the OELIST.
pce	Pushdown <u>capacity exceeded</u> .
nna	Non- <u>numeric argument</u> for arithmetic, followed by the argument in question; returns value zero and proceeds.
ana	Argument <u>not atom</u> (for PRIN1); returns NIL as usual and proceeds.
ovf	Division <u>overflow</u> ; returns zero and proceeds.

9. Some Remarks

In general, each character in each LISP expression is recognized by the computer as 2 octal digits called concise code. The pairs of octal digits are packed 3 pairs at a time into the 6-octal-digit registers of the PDP-1. If a LISP atom has a number of characters which is not a multiple of three, there will be spaces left over, which are filled arbitrarily with a filler character, 76 (octal). For example, a LISP word with 7 characters such as SMALLER will be packed into three computer registers, S M A in one, L L E in a second, and R along with two filler characters in the third.

These three registers are linked by list structure. An example of a hypothetical list structure which might store SMALLER if introduced as a defined function into the LISP system would be as shown in Table 4:

Table 4

PDP-1 <u>Register</u>	<u>Contents</u>	<u>Meaning</u>	<u>Comments</u>
5763	405765	pointer to 5765	5765 is the start of the print name of the atom SMALLER
5764	005773	pointer to property list	5773 is the start of the property list
5765	224461	S M A	Concise code
5766	005767	pointer	5767 holds continuation of the list
5767	434365	L L E	Concise code
5770	005771	pointer	5771 holds continuation of the list
5771	767651	- - R	Concise code and 2 filler characters
5772	003011	nil	Terminator of list

If SMALLER were defined by the expression:

```
(RPLACD (QUOTE SMALLER) (QUOTE (EXPR (LAMBDA (X Y)
o                                4 4
      (COND ((GREATERP Y X) X) (T Y))))))  
4 56 6 6 543210
```

then the property list of SMALLER would be (hypothetically) as shown in Table 5:

Table 5

<u>Register</u>	<u>Contents</u>	<u>Meaning</u>
5773	003271	"EXPR"
5774	005775	pointer
5775	005777	pointer
5776	002651	"NIL"
5777	003255	"LAMBDA"
6000	006001	pointer to forking
6001	006003	pointer to (X Y)
6002	006007	pointer to (COND)
6003	007701	"X"
6004	006005	pointer
6005	007711	"Y"
6006	002651	"NIL"
6007	002725 etc.	"COND"

An accepted LISP expression L is identified within the machine by the address of the list structure in storage which represents L.

The computer evaluates expressions using either machine subroutines (SUBRs and FSUBRs) or LISP subroutines (EXPRs or FEXPRs).

The computer converts the resulting value into concise codes, and presents the value for output to the computer-associated typewriter or the punch.

Basic PDP-1 LISP is very flexible:

1. The number of registers on the push-down list can be reasonably varied between 200 and 400 octal. The number chosen can vary according to the amount of recursion it is desired to provide for.

2. The number of registers of storage (there is only one kind of storage) can be varied from under 1000 octal to over 4000 octal in a one-core machine. In the smallest extreme case, LISP system can occupy only the registers up to about 4000 octal; in the other extreme case LISP can occupy all the registers up to 7750 octal, leaving 7751 to 7777 for the read-in subroutine.

3. Machine subroutines may be located in core, and referred to and used. These machine subroutines should be located above the highest register in free storage.

4. DDT (the Digital Debugging Tape) may be loaded in registers 5500 up and LISP may be loaded below, so that the facilities of DDT are available for modifying LISP.

5. A core dump routine may be loaded into 400 (octal) registers above free storage and used upon LISP.

Part II

1. Macro Symbolic Program for Basic PDP-1 LISP

```
lisp 3-23-64 : 1 field
define termin          define move (B,A      load A,B
                  extend
define law B           define 1load A,B    count A,B
define dac A           define isp A
define termin          define jmp B
                        define termin
define law B           define init A,B    test K,P
define dap A           define sad (K
define termin          define jmp P
                        define termin
define idx A           define index A,B,C   undex A
define sas B
define jmp C
define termin
define index A,(B,C    define step A,B,C   swap
define termin          define rcl 9s
                        define rcl 9s
                        define termin
define law 1 B          define setup A,B    smi=spi 1
define dac A           define xy=0
define termin          define xx=hlt
                        define clo=spa
                        define sma
                        define szo
                        define 1-szf-szf
define jmp R           define mul=540000
define termin          define div=560000
define lac A           define move A,B
define dac B           start
define termin
```

Lisp interpreter 3-20-64, part 1

4/		buf, 77/	0 0
go,	hlt+cla+cli+7-opr-opr stf 6 extend dzm 77 law 77 dap avx	dap rx sub (1 dap .+1 lac xy dap ave+1 lac rx jda pwl ave, lac 100	
beg,	law pdo-1 dac pdl lac n dac ar2 cal rin cal evo cal pnt jmp beg	exit /create number	
t0, t1, g0, g1, h1, csi, cso, ffi, ga1, 0 gst, a0, a1, a2,	0 0 0 0 0 72 72 0 0 0 isi, repeat 5,20 isi-1	crn, lio (jmp rcl 2s rar 2s dac 100 jmp cpf	
		/print or punch character	
pc,	and (77 sad (76 jmp x ior (ral dac pcc sad (ral 77 jmp pcc-3 isp pch jmp pcc-1 law 277 cal out law i 100 dac pch law 252		
/append word to pdl			
pwl,	0 dap pwx idx pdl sad bfw jmp qg2 iac pwl dac 1 pdl	pcc, xx and (200 ior pcc dac 100 stf 2 jmp out	
pwx,	exit	pch, -100	
/retrieve word from pdl		/get numeric value	
uw, uwl,	0 dap uwx lio i pdl undex pdl	vag, lio i 100 cla rcl 2s sas (3 jmp q13 idx 100	
uwx,	exit		

lac i 100	ern,	0
rcl 8s		.+1
rcl 8s		347776
Jmp x	n,fro,	nil
/get two values		
vad, dio a1	define	error F
cal vag		jsp err
dac a0		F
lac a1		termin
cal vag		/garbage collector, non-compacting
dac a1	gc,	dap gcx
Jmp x		dio ga1
/pack character onto end of buffer		
oc, rar 6s		dio gfr
lio i isi		lac gfr
rcl 6s		sar 2s
sad (76		sza
Jmp oc1		jsp gfr+1
lac 100		lac ffi
ior (767600		sza i
cal cf		Jmp gco
lio t0		lac 100
idx t0		jda gfr
idx isi		lac isi
dac a1		sas (isi-1
dio isi		Jmp gci
lac i a1		law pdl+1
dac i t0		dac g1
dio i a1	gcp,	lac i g1
Jmp x		jda gfr
oci, dio i isi		idx g1
Jmp x		sub (1
/output routine		
out, lio 100		sad pdl
szs 36		Jmp g2e
ppa		Jmp gcp
szs i 66		/mark one list
tyo	gfr,	0
Jmp x		dap gfx
/error printout		
err, clf 6		lac gfr
dap erx		ral 1s
lac i erx		spq
dac ern		Jmp gfx
law erm		lac pdl
cal pra		jda pw1
stf 6	gfn,	lio i gfr
idx erx		idx gfr
erx, exit		lac i gfr
erm, 357776		spa
.+1		Jmp gfu
		ior (add
		dac i gfr
		spi
		Jmp gfd
		jda pw1

	dio gfr	/SASSOC
gfd,	jmp gfn	
	ril 1s	aso, cal asc
	spi i	
	jmp gfa	
gfu,	jsp uw1	
	dio gfr	
	sas gfr	
	jmp gfn	
gfx,	exit	ase, cal cns-1 jmp evo
gfa,	rir 1s	
	dio g0	
	dac gfr	
gfl,	idx g0	
	lac 1 g0	
	spa	
	jmp gfn	
	ior (add	
	dac 1 g0	
	dac g0	
	xor (add	
	sas n	
	jmp gfl	
	jmp gfn	
/garbage collector, linear sweep phase		as1, sad n jmp x lac 1 a1 dac t0 lac 1 to sad 100 jmp as2 idx a1 lac 1 a1 dac a1 jmp as1
g2e,	lac fro	as2, idx 1 pdl lac t0 jmp x
	dac g0	
g2n,	idx g0	/program feature
	lio 1 g0	
	smi	/PROG
	jmp g2f	
	ril 1s	
	sir 1s	
g2a,	dio 1 g0	
	idx g0	
	sas hi	
	jmp g2n	
g2x,	lio ga1	
gcx,	exit	
g2f,	lio fre	/append program variables
	sub (1	
	dac fre	
	jmp g2a	
gci,	sad n	lac ar1
	jmp gcp-2	
	dac gfr	
	dac g0	
	lac pdl	
	jda pwl	
	law gcp-2	
	dap gfx	
	jmp gfl	
		pg5, sad n jmp pg6 lac 1 ar1 cal cns-1 lio ar2 cal cns dac ar2 idx ar1 lac 1 ar1 dac ar1 jmp pg5

```

/expand go-list (on a-list)          goe,      lio 100
pg6,      lac pa3                  lac n
pg7,      dac ar1                cal cns
                     sad n           dac pa3
                     jmp pg0           jmp prx
                     lac i ar1
                     cal car
                     sma
                     jmp pg9
                     lac ar1
                     lio ar2
                     cal cns
                     dac ar2
pg9,      idx ar1                /SETQ
                     lac i ar1
                     jmp pg7
                     stq,      dac ar1
                     dio t1
                     lac i ar1
                     cal asc
                     jmp qa4
                     jda pwl
                     lac ar1
                     cal cdr
                     cal car
                     lio t1
                     cal evl
                     jda uw
                     dio t0
                     idx t0
                     lac uw
                     dac i t0
                     jmp x
/process program
pg0,      lac pa3
pg1,      sad n
                     jmp pg2
                     lac i pa3
                     cal car
                     spa
                     jmp pg3
                     lac ar2
                     jda pwl
                     lac 100
                     cal evo
                     jsp uw
                     dio ar2
                     cla
                     sas pa4
                     jmp pg4
                     /CDR
                     cdr,      idx 100
                     /CAR
                     car,      lac i 100
                     x,        jda uw
                     dio rx
                     lac uw
                     rx,        exit
                     /ATOM
pg3,      idx pa3
                     lac i pa3
                     dac pa3
                     jmp pg1
                     atm,      lac i 100
                     sma
                     jmp fal
                     tru,      lac tr
                     jmp x
                     /NULL
pg2,      lac pa4
                     jda uw
                     dio pa4
                     jsp uw
                     dio pa3
                     lac uw
                     jmp x
                     nul,      lio n
                     /EQ
                     eqq,      dio a1
                     sad a1
                     jmp tru
                     lac i a1
                     and i 100
                     and {jmp
                     sas {jmp
                     jmp fal
/RETURN
ret,      dac pa4
                     jmp x
/GO

```

lac 100	lio (-0	
cal vad	dio a0	
sad a0	lio (and a0	
jmp tru	jmp pl1	
jmp fal		
/RPLACD	lgo,	cal elc
rdc, idx 100	lio (ior a0	
sub (1	jmp plz	
 	tim,	cal elc
/RPLACA	lio (1	
rda, dio i 100	dio a0	
jmp x	lio (jmp tic	
	jmp pl1	
/create atom	tic,	mul a0
mka, ior (add	scr 1s	
dac 100	dio 100	
lio n	add 100	
	jmp pl0+1	
/CONS	gcs,	jsp gc
cns, idx ffi	lac fre	
cnc, lac fre	sas n	
sad n	jmp cna	
jmp gcs	jmp qg1	
cna, dac t0	/TERPRI	law 77
lac 100	tpr,	cal pc
dac i fre		jmp prx
idx fre		
lac i fre		
dio i fre	/PRIN1	
dac fre		
lac t0	pr1,	lac i 100
jmp x	sma	
 		jmp qp1
/PLUS		sub (lac
pls, cal elc		spa
lio (add a0		jmp prn
dzm a0		and (-jmp
plz, dio plo		
pl1, sad n		
pl2, jmp ple		
	pra,	sad n
		jmp x
		dac a0
		lac i a0
		ral 6s
		cal pc
plo, 0		lac i a0
		rar 6s
		cal pc
		lac i a0
		cal pc
		idx a0
		lac i a0
ple, lac a0		jmp pra
jmp crn		
/LOGAND, LOGOR, TIMES		
lga, cal elc		

prn,	lac 100 cal vag dac t1 clf 2 setup t0,6	lac t1 cal cns-1 lio ar2 cal efc lio ar1 dac 100
prv,	lio t1 sad (-1 stf 2 cla rcl 3s dio t1 sza 1 law 20 sad (20 szf 2 cal pc isp to jmp prv jmp prx	xen, /GENSYM gsm, law gst dac to
/NUMBERP		gsi, idx i to sad (12 jmp gsn sad (21 law 1 dac i to
nmp,	lac 1 100 and (jmp sad (jmp jmp tru jmp fal	gsp, lac gst+2 ral 6s ior gst+1 ral 6s ior gst cal cf law 6700 ior gst+4 ral 6s ior gst+3 lio to cal cpf cal mka jmp x
/do a CONS into full word space		
cf,	lio n	
cpf,	dzm ffi jmp cnc	
/MINUS		
min,	cal vag cma jmp crn	gsn, law 20 dac i to idx to sas (gst+5 jmp gsi jmp gsp
/XEQ		
xeq,	cal vad lac tr dac t1 lac a2 cal vag lio a0 dio xe1 lac a1 lio uw 0 jmp xen dio a2 cal crn dac ar1 lac a2 cal crn dac ar2	/QUOTIENT qot, cal vad lio a0 cla spi clc rcl 1s div a1 jmp q14 jmp crn
xer,		/COND cnd, dio ar2

cd1,	dac ar1 sad n jmp qa3 jda pwl lac ar2 jda pwl lac i ar1 cal car cal evo jda uw dio ar2 jsp uwL dio ar1 lac uw sas n jmp cdy idx ar1 lac i ar1 jmp cd1	avc, avt, avr, avn, avs,	xx sma jmp qc3 law 77 and avc sas {72 sad {74 dac csi sad csi jmp ava jmp x index avx,ave,avx init avx,buf dap avs rpa rcr 9s rpa rcl 9s dio xy step avs,dio 100,avn jmp ava
cdy,	lac i ar1 cal cdr cal car jmp evo		
/STOP		avi,	szf i 1 jmp ava ty1 clf 1 dio avc jmp avt
stp,	cal vag hlt+cli-opr jmp prx		
/GREATERP			/terminate print name
grp,	cal vad clo sub a0 szo lac a1 sma jmp fal jmp tru	mkn,	law 72 sas cso cal oc idx isi dac t0 lio n dio isi lac i t0 dio i t0 jmp x
/get a character			
ava,	szs 50 jmp avi cli		/pack character into print name
avx,	lac 77 sza 1 jmp avr rcl 9s dio i avx ral 2s spq jmp ava ral 7s ior (rar dac avc law 525	pak, pk1, start	dap pk1 lac csi sad cso jmp pk1 dac cso cal oc law dac 100 jmp oc

Lisp interpreter 3-20-64, part 2

/.

```

/PRINT

pnt,    dac a0          rid,    spi
        dac a1          jmp r12
        cal tpr

pn1,    lio i a0        riq,    idx ar1
        spi             lac i ar1
        jmp pn2         dio i ar1
        law 57          dac r19
                           jsp rhe
                           jmp rix

pn5,    cal pc          r13,    dac r19
        lac a0          jmp r13-2
        cal cdr
        jda pwl
        lio i a0
        dio a0
        jmp pn1
pn2,    lac a0          r12,    lac (jmp r13
        cal pr1         jda pwl
                           law ric

pn6,    jsp uw1          /read symbol and terminator
        cla
        dio a0
        spi
        jmp pn7
        lio i a0
        spi i
        jmp pn5
        lac a0
        sad n
        jmp pn3
        law 73
        cal pc
        lac a0
        cal pr1

pn3,    law 55          rhe,    dap rhx
        cal pc
        jmp pn6

pn7,    cal pc          rhn,    cal ava
        lac a1
        jmp a0

pn8,    0
pn9,    0
rin,    lac rx          rhb,    dac 100
        dac ar1
        dzm r19

ris,    jsp rhe          sza 1
        sza 1
        jmp ric
        sad (57
        jmp ria
        sad (55
        jmp rib

```

/READ

```

r18,    0
r19,    0
rin,    lac rx
        dac ar1
        dzm r19

ris,    jsp rhe
        sza 1
        jmp ric
        sad (57
        jmp ria
        sad (55
        jmp rib

```

/.

ryp,	stf 5 cal pak jmp rhn	ryn,	lio 100 lac t1 rir 3s rcl 3s dac t1 lac 100 jmp ryp+1
ryj,	lac t1 cal crn jmp rhr		
ryo,	cal ava jmp ryp	ryy,	clc lio (isi-1 dio isi
/symbol lookup		rhr,	dac t0 lac r19 lio r18 dio r19 lio t0
rye,	dac r18 cal mkn dac a0 sad n jmp ryy szf 1 5 jmp ryj lac i 1ob		rhx, exit
			/, space tab
rys,	dac t0 sad n jmp ryc lac i t0 dac t1 lac i t1 dac t1 lac a0	ric,	lac ar1 spi jmp ris spa jmp r14
		rio,	dio t0 cal cdr lio t0
ryw,	dac a1 sas n jmp ryt sad t1 jmp rhh	rie,	swap cal cns idx ar1 lac t0 dac i ar1 dac ar1 jmp ris
ryd,	idx t0 lac i t0 jmp rys		
ryt,	lac t1 sad n jmp ryd lac i a1 sas i t1 jmp ryd idx t1 lac i t1 dac t1 idx a1 lac i a1 jmp ryw	ri4,	lac t0 jmp ar1
			/
ryc,	lac a0 cal mka lio i 1ob cal cns dac i 1ob	ria,	dio t0 lac ar1 jda pw1 lac t0 spa jmp riz cal cns-1
rhh,	lac i t0 jmp rhr	riy,	dac ar1 lio ar1 cal rdc jmp ris
		riz,	dzm ar1 jmp ris

```

rib,    idx ar1          evc,    idx ar1
       lac i ar1          lio i ar1
       lio n              lac uw
       dio i ar1          dzm ar1
                               cal cns
                               jmp evo

rix,    jda uw          /x is atomic : search a-list,
       dio ar1          then p-list
       ril 1s
       lac uw
       spi
       jmp ar1
       lio uw
       lac ar1
       szs
       jmp rio
       lac uw
       jmp riy

/EVAL

evl,    dio ar2          ev5,    lac ar1
evo,    dac ar1          ev4,    cal cdr
                           sad n
                           jmp qa8
                           dac t0
                           lac i t0
                           sad 1ap
                           jmp ev6
                           idx t0
                           lac i t0
                           jmp ev4

ev2,    lac ar1          ev6,    idx t0
       szs 10
       cal pnt
       lac i ar1
       spa
       jmp e1
       dac t0
       lac i t0
       spa
       jmp e2

/car[x] not atomic      en1,    lac ar1
                           /exit from EVAL
                           sad 1la
                           jmp e3
                           lac ar2
                           jda pwl
                           lac ar1
                           jda pwl
                           lac i ar1
                           cal evo
                           jsp uw1
                           dio ar1
                           jsp uw1
                           dio ar2
                           jmp evc

/evaluate function name and try again
ev3,    lac i ar1          ex,    szs 10
       cal asr
       jmp qa8
       cal cdr
                           jmp pnt
                           jmp x

                           /car[x] is atomic : search
                           its p-list
                           e2,    lac t0
                           ev8,    cal cdr
                           sad n
                           jmp ev3
                           lac i uw
                           sad 1fs
                           jmp efs
                           sad 1sb
                           jmp esb
                           sad 1xp

```

```

        jmp exp           /function is SUBR
        sad 1fx
        jmp efx
        idx t1
        lac i t1
        jmp ev8

/function is FSUBR

efs,   idx uw
       lac i uw
       cal car
       cal vag
       dac exx
       idx ar1
       lac i ar1
       lio ar2

exy,   dac 100
       dzm ar1

exx,   0
       jmp ex

/function is FEXPR

efx,   idx uw
       lac i uw
       cal car
       jda pwl
       lac ar1
       cal cdr
       cal efq
       jda pwl
       lac ar2
       cal efq
       cal cns-1
       jsp uw1
       cal efc
       jsp uw1
       cal efc
       dac ar1
       jmp ev2

efq,   cal cns-1
       lio t0
       lac 1qu
       dac 100
       jmp cns

efc,   dio 100
       lio t0
       jmp cns

/function is EXPR

exp,   idx uw
       lac i uw
       dac a1
       idx ar1
       lio i ar1
       dzm ar1
       lac i a1
       cal cns
       jmp evo

esb,   idx uw
       lac i uw
       cal car
       jda pwl
       lac ar1
       cal cdr
       lio ar2
       cal elc
       jmp els

/evaluate argument list : also LIST

elc,   sad n
       jmp x
       dac ar1
       dio ar2
       lac ar2
       jda pwl
       lac ar1
       dzm ar1

ele,   lio i pd1
       dac t0
       jda pwl
       lac ar1
       jda pwl
       lac i t0
       cal evl
       cal cns-1
       jsp uw1
       dio ar1
       lio t0
       lac ar1
       sza i
       dio ar1
       idx ar1
       sub (1
       sas t0
       lio i ar1
       lac t0
       dac i ar1
       dac ar1
       idx t0
       dio i t0
       jsp uw1
       swap
       cal cdr
       sas n
       jmp ele
       jsp uw1
       dio ar2
       idx ar1
       lac i ar1
       lio n
       dio i ar1
       dac ar1
       szs 10
       cal pnt
       lac ar1
       jmp x

```

els,	dac ar1 jsp uw1 swap cal vag dac exx init esa,a0-1	cal cdr lio ar2 cal elc dac ar1 jsp uw1 dio a0 jsp uw1 dio ar2
<i>/store arguments for subroutine</i>		
eda,	lac ar1 sad n jmp exs idx esa sad (dac a2+1 jmp qa7 lac i ar1	ep1, lac a0 sad n jmp ep2 lac ar1 sad n jmp qf3 lac i a0 lio i ar1
esa,	dac xy idx ar1 lac i ar1 dac ar1 jmp eda	cal cns lio ar2 cal cns dac ar2 idx a0 lac i a0
exs,	lac a0 lio a1 jmp exy	dac a0 idx ar1 lac i ar1 dac ar1 jmp ep1
<i>/caar[x] = LAMBDA</i>		
e3,	lac ar1 jda pw1 lac ar2 jda pw1 lac i ar1 cal cdr cal car jda pw1 lac ar1	ep2, sas ar1 jmp qf2 jsp uw1 dio ar1 lac i ar1 cal cdr cal cdr cal car jmp evo

```

/error halt entries

qa3,    lac n
        sas pa3
        jmp x
        error flex icd      /illegal COND
        lac n
        jmp x

qa4,    error flex uss      /undefined atom in SETQ
        jmp prx

qa7,    error flex tma      /too many args
        jmp exs

qa8,    error flex uas      /unbound atomic symbol
        clf 6
        lac ar1
        cal pnt
        cal tpr
        jmp go

qc3,    error flex ilp      /illegal parity
        law 377
        and avc
        hlt+cli-opr+1
        jmp ava

qf2,    error flex lts      /LAMBDA list too short
        jmp go

qf3,    error flex ats      /arglist too short
        jmp go

qg2,    error flex pce      /pushdown cap. exc.
        jmp go

qg1,    error flex sce      /storage cap. exc.
        jmp go

q13,    lac 100
        dac a2
        error flex nna      /non-numeric arg for arith.
        clf 6
        lac a2
        cal pnt
        cal tpr
        jmp q1x

q14,    error flex ovf      /overfl^w

q1x,    cla 16
        jmp crn

qp1,    error flex ana      /arg non-atom for PRIN1

prx,fal,          lac n
        jmp x

start

```

lisp storage 3-23-64	rrc 1s rll 1s dio hi law end dac t0
constants	
/special symbols	
ssy,	/relocate storage
1qu, quo 1la, lam 1ap, apv 1ob, obl 1sb, sbr 1fs, fsb 1xp, xpr 1fx, fxp fre, nil bfw, frs-4 tr, t	rrs, law i 1 add t0 dac t0 law i 4 add i t0 sma jsp rrl jsp mvs law i 1 add t0 dac t0 sub frl spa jsp rrl jsp mvs lac t0 sas ofs jmp rrs law ssy dac t0
pdl, pdo-1	
ar1, nil ar2, nil pa3, nil pa4, 0	
pdo,	
/load storage parameters	/relocate special registers
lio mz clc+hl1-opr lat+cli-opr and ad dac hi1 hl1 lat and ad dac lp1	rss, jsp rrl idx t0 sas esy jmp rss lac i 1ob jda gfr law go dap gcx jmp g2e
law i end add hl1 spa jmp pdo law i frs-pdo add lp1 spa jmp pdo law i pdo+end-frs add hl1 sub lp1 spa jmp pdo	/relocate 1 word, move 1 word
rrl, dap rrx lac i t0 and ad sub ofr spa jmp rrx lac i t0 add fro sub ofs dac i t0 rrx, jmp .	
/set up registers	mvs, dap mvx lac t0 add fro sub ofs dac t1 lac i t0 dac i t1
stu, law pdo add lp1 dac fro lio hl1	

```

mvx,    jmp .
/constants etc.

ad,     177777
lp1,    0
hi1,    0
mz,     -0
ofs,    frs
frl,    fws
esy,    pdo
ofr,    pdo

define      item X
.2          .+3
add X      nil
termin

define      next A
A           .+1
termin

define      subr F
.2          .+7
add F+2   .+1
sbr       .+1
F          nil
termin

define      fsubr F
.2          .+7
add F+2   .+1
fsb       .+1
F          nil
termin

define      apval A
apv        .+1
A           nil
termin

terminin

frs,
nil, add f38 kz
t,   add f37 kt
kz, apval nil
kt, apval t
obj, add fb0 .+1
apv   .+1
obl, ols   nil

/objects list
ols, subr f2
subr f3
subr f4

fsubr f6
subr f7
subr f8
subr f12
subr f13
fsubr f14
subr f18
subr f21
subr f24
subr f26
subr f27
subr f32
subr f33
subr f34
fsubr f50
subr f51
subr f52
fsubr f53
subr f54
fsubr f60
fsubr f61
fsubr f62
fsubr f63
subr f00
subr f01
subr fa3

quo=. 2      fsubr fb5
lam=. 2      item f40
apv=. 2      item f42
sbr=. 2      item f43
xpr=. 2      item f44
fsb=. 2      item f45
fxp=. 2      item f46

next t
next obj
subr fb2
subr fb3
subr fb4
nil      nil

fws,
define      opr A
            termin
            loca A
            0

define      X
            termin
            nam1 X
            nil

define      X
            nam1 Y
            termin
            nam2 X,Y
            .+1

define      X
            nam2 Y,Z
            termin
            nam3 X,Y,Z
            .+1

```

/SUBRs and FSUBRs

f2,	loca atm	nam2 flex ato,767644
f3,	loca car	nam1 flex car
f4,	loca cdr	nam1 flex cdr
f6,	loca cnd	nam2 flex con,767664
f7,	loca cns	nam2 flex con,767622
f8,	loca eqq	nam1 766550
f12,	loca gsm	nam2 flex gen,flex sym
f13,	loca grp	nam3 flex gre,flex ate,765147
f14,	loca elc	nam2 flex lis,767623
f18,	loca min	nam2 flex min,762422
f21,	loca nmp	nam3 flex num,flex ber,767647
f24,	loca stp	nam2 flex sto,767647
f26,	loca pr1	nam2 flex pri,764501
f27,	loca qot	nam3 flex quo,flex tie,764523
f32,	loca rda	nam2 flex rpl,flex aca
f33,	loca rdc	nam2 flex rpl,flex acd
f00,	loca xeq	nam1 flex xeq
f01,	loca crn	nam1 flex loc
f34,	loca tpr	nam2 flex ter,flex pri
f50,	loca pgm	nam2 flex pro,767667
f51,	loca ret	nam2 flex ret,flex urn
f52,	loca goe	nam1 766746
f53,	loca stq	nam2 flex set,767650
f54,	loca aso	nam2 flex sas,flex soc
fb2,	loca rin	nam2 flex rea,767664
fb3,	loca evl	nam2 flex eva,767643
fb4,	loca pnt	nam2 flex pri,764523
fb5,	loca car	nam2 flex quo,762365
fa3,	loca nul	nam2 flex nul,767643
f60,	loca pls	nam2 flex plu,767622

f61, loca tim nam2 flex tim,766522
f62, loca lga nam2 flex log,flex and
f63, loca lgo nam2 flex log,764651
/miscellany
f38, nam1 flex nil
f40, nam2 flex lam,flex bda
f42, nam2 flex apv,766143
f43, nam2 flex sub,767651
f44, nam2 flex exp,767651
f45, nam2 flex fsu,766251
f46, nam2 flex fex,764751
fb0, nam2 flex obl,flex ist
f37, nam1 767623
end,
start pdo

2. Alphabetic Listing of Defined Macro Symbols

Following is an alphabetic listing of the defined symbols used in the macro symbolic program for Basic PDP-1 LISP. The listing shows either the numeric meaning of the instruction or the numeric register (octal) in which the subroutine commences. For the mnemonic derivation or significance of the symbols, see Section 4 below.

1ap	2333	clo	651600
1fs	2336	cna	620
1fx	2340	cnc	615
1la	2332	cnd	1070
1ob	2334	cns	614
1qu	2331	cpf	762
1sb	2335	crn	112
1xp	2337	csi	26
a1	42	cso	27
a2	43	dba	720061
ad	2470	dcc	720062
apv	3110	dia	720060
ar1	2345	div	560000
ar2	2346	dra	720063
a0	41	e1	1626
as1	405	e2	1661
as2	420	e3	2077
asc	403	eda	2061
ase	377	efc	1743
aso	374	efq	1736
asr	402	efs	1701
atm	562	efx	1715
ava	1133	elc	1770
avc	1152	ele	2000
ave	110	els	2050
avi	1204	en1	1655
avn	1173	end	3530
avr	1165	ep1	2120
avs	1177	ep2	2143
avt	1155	eqq	570
avx	1136	erm	227
beg	11	ern	231
bfw	2342	err	216
buf	63	erx	226
car	555	esa	2067
cd1	1071	esb	1757
cdr	554	esy	2476
cdy	1114	ev2	1565
cf	761	ev3	1614

ev4	1637	fb3	3410
ev5	1636	fb4	3416
ev6	1651	fb5	3424
ev8	1662	fb0	3522
evc	1620	ffi	30
evl	1563	fre	2341
evo	1564	frl	2475
ex	1656	fro	234
exp	1746	frs	2500
exs	2074	f01	3334
exx	1713	f00	3330
exy	1711	fsb	3124
f12	3226	fws	3170
f13	3234	fxp	3130
f14	3244	g1	24
f18	3252	g2a	351
f2	3170	g2e	341
f21	3260	g2f	357
f24	3270	g2n	343
f26	3276	g2x	355
f27	3304	ga1	31
f3	3176	gc	235
f32	3314	gci	363
f33	3322	gco	251
f34	3340	gcp	260
f37	3526	gcs	673
f38	3470	gcx	356
f4	3202	gfa	323
f42	3476	gfd	313
f43	3502	gfl	326
f44	3506	gfn	277
f45	3512	gfr	267
f46	3516	gfu	316
f40	3472	gfx	322
f51	3354	go	4
f52	3362	goe	526
f53	3366	grp	1123
f54	3374	g0	23
f50	3346	gsi	1025
f6	3206	gsm	1023
f61	3446	gsn	1051
f62	3454	gsp	1033
f63	3462	gst	34
f60	3440	hi	25
f7	3214	hi1	2472
f8	3222	ioh	730000
fa3	3432	isi	33
fal	2241	kt	2510
fb2	3402	kz	2504

lai	760040	plz	633
lam	3104	pn1	1240
lga	651	pn2	1253
lgo	656	pn3	1274
lia	760020	pn5	1244
lp1	2471	pn6	1255
min	764	pn7	1277
mka	611	pnt	1235
mkn	1212	pr1	703
mul	540000	pra	712
mvs	2460	prn	730
mvx	2467	prv	736
mz	2473	prx	2241
n	234	pw1	44
nil	2500	pxw	53
nmp	754	qa3	2154
nul	567	qa4	2163
obj	2514	qa7	2166
obl	2520	qa8	2171
oc	165	qc3	2200
oc1	206	qf2	2206
ofr	2477	qf3	2211
ofs	2474	qg1	2217
ols	2522	qg2	2214
out	210	qi3	2222
pa3	2347	qi4	2233
pa4	2350	qix	2235
pak	1224	qot	1057
pc	117	qp1	2237
pcc	135	quo	3074
pch	143	rda	607
pdl	2344	rdc	605
pdo	2351	ret	524
pg1	472	rhb	1360
pg2	516	rhe	1333
pg3	511	rhh	1462
pg4	515	rhn	1343
pg5	437	rhr	1476
pg6	452	rhx	1503
pg7	453	ri2	1330
pg9	466	ri3	1326
pgm	423	ri4	1524
pg0	471	ri8	1302
pk1	1232	ri9	1303
pl1	634	ria	1526
pl2	635	rib	1543
ple	647	ric	1504
plo	642	rid	1316
pls	631	rie	1514

rin	1304	spq	650500
rio	1511	ssy	2331
riq	1320	stp	1120
ris	1307	stq	533
rix	1547	stu	2377
riy	1534	swp	160060
riz	1541	szm	640500
rrl	2445	t	2502
rrs	2410	t1	22
rrx	2457	tic	666
rss	2434	tim	661
rx	561	tpr	700
ryc	1455	tr	2343
ryd	1436	tru	565
rye	1411	t0	21
ryj	1404	uw	54
ryn	1464	uwl	55
ryo	1407	uwx	62
ryp	1401	vad	156
rys	1421	vag	144
ryt	1441	x	556
ryw	1431	xei	1000
ryy	1473	xen	1017
sbr	3114	xeq	767
sft	660000	xer	1003
smi	652000	xpr	3120
sni	644000	xy	0

3. Numeric Listing of the Defined Macro Symbols

Following is a listing in numerical order by register number or other meaning of the defined symbols in the macro symbolic program for Basic PDP-1 LISP.

xy	0	gfr	267
go	4	gfn	277
beg	11	gfd	313
t0	21	gf <u>u</u>	316
t1	22	gfx	322
g0	23	gfa	323
g1	24	gfl	326
hi	25	g2e	341
csi	26	g2n	343
cso	27	g2a	351
ffi	30	g2x	355
ga1	31	gcx	356
isi	33	g2f	357
gst	34	gci	363
a0	41	aso	374
a1	42	ase	377
a2	43	asr	402
pwl	44	asc	403
pxw	53	as1	405
uw	54	as2	420
uw1	55	pgm	423
uxw	62	pg5	437
buf	63	pg6	452
ave	110	pg7	453
crn	112	pg9	466
pc	117	pg0	471
pcc	135	pg1	472
pch	143	pg3	511
vag	144	pg4	515
vad	156	pg2	516
oc	165	ret	524
oc1	206	goe	526
out	210	stq	533
err	216	cdr	554
erx	226	car	555
erm	227	x	556
ern	231	rx	561
fro	234	atm	562
n	234	tru	565
gc	235	nul	567
geo	251	eqq	570
gcp	260	rdc	605

rda	607	pk1	1232
mka	611	pnt	1235
cns	614	pn1	1240
cnc	615	pn5	1244
cna	620	pn2	1253
pls	631	pn6	1255
plz	633	pn3	1274
pl1	634	pn7	1277
pl2	635	r18	1302
plo	642	r19	1303
ple	647	rin	1304
lga	651	ris	1307
lgo	656	rid	1316
tim	661	riq	1320
tic	666	ri3	1326
gcs	673	ri2	1330
tpr	700	rhe	1333
pr1	703	rhn	1343
pra	712	rhb	1360
prn	730	ryp	1401
prv	736	ryj	1404
nmp	754	ryo	1407
cf	761	rye	1411
cpf	762	rys	1421
min	764	ryw	1431
xeq	767	ryd	1436
xei	1000	ryt	1441
xer	1003	ryc	1455
xen	1017	rhh	1462
gsm	1023	ryn	1464
gsi	1025	ryy	1473
gsp	1033	rhr	1476
gsn	1051	rhx	1503
qot	1057	ric	1504
cnd	1070	rio	1511
cd1	1071	rie	1514
cdy	1114	ri4	1524
stp	1120	ria	1526
grp	1123	riy	1534
ava	1133	riz	1541
avx	1136	rib	1543
avc	1152	rix	1547
avt	1155	evl	1563
avr	1165	evo	1564
avn	1173	ev2	1565
avs	1177	ev3	1614
avi	1204	evc	1620
mkn	1212	e1	1626
pak	1224	ev5	1636

ev4	1637	tr	2343
ev6	1651	pdl	2344
en1	1655	ar1	2345
ex	1656	ar2	2346
e2	1661	pa3	2347
ev8	1662	pa4	2350
efs	1701	pdo	2351
exy	1711	stu	2377
exx	1713	rrs	2410
efx	1715	rss	2434
efq	1736	rrl	2445
efc	1743	rrx	2457
exp	1746	mvs	2460
esb	1757	mvx	2467
elc	1770	ad	2470
ele	2000	lp1	2471
els	2050	hi1	2472
eda	2061	mz	2473
esa	2067	ofs	2474
exs	2074	frl	2475
e3	2077	esy	2476
ep1	2120	ofr	2477
ep2	2143	frs	2500
qa3	2154	nil	2500
qa4	2163	t	2502
qa7	2166	kz	2504
qa8	2171	kt	2510
qc3	2200	obj	2514
qf2	2206	obl	2520
qf3	2211	ols	2522
qg2	2214	quo	3074
qg1	2217	lam	3104
qi3	2222	apv	3110
qi4	2233	sbr	3114
qix	2235	xpr	3120
qp1	2237	fsb	3124
fal	2241	fxp	3130
prx	2241	f2	3170
1qu	2331	fws	3170
ssy	2331	f3	3176
1la	2332	f4	3202
1ap	2333	f6	3206
1ob	2334	f7	3214
1sb	2335	f8	3222
1fs	2336	f12	3226
1xp	2337	f13	3234
1fx	2340	f14	3244
fre	2341	f18	3252
bfw	2342	f21	3260

f24	3270	fb5	3424
f26	3276	fa3	3432
f27	3304	f60	3440
f32	3314	f61	3446
f33	3322	f62	3454
f00	3330	f63	3462
f01	3334	f38	3470
f34	3340	f40	3472
f50	3346	f42	3476
f51	3354	f43	3502
f52	3362	f44	3506
f53	3366	f45	3512
f54	3374	f46	3516
fb2	3402	fb0	3522
fb3	3410	f37	3526
fb4	3416	end	3530