MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Artificial Intelligence
Memo No. 98

Memorandum MAC-M-313
June, 1966

PDP-6 LISP

Peter Samson

SUBJECT: PDP-6 LISP
FROM: PETER SAMSON
DATE: JUNE 16, 1966

MAC MEMO MAC-M-313
ARTIFICIAL INTELLIGENCE MEMO 98

This is a mosaic description of PDP-6 LISP, intended for readers familiar with the LISP 1.5 Programmer's Manual or who have used LISP on some other computer. Some of the newer features (e.g. the display) are experimental and subject to change; in such respects this should not be regarded as a final document.

SOME DISTINCTIVE CHARACTERISTICS

Top-level typein is to EVAL. There is no EVALQUOTE.

EQUAL will not correctly compare fixed-point numbers to floating-point. Also (ZEROP Ø.Ø) is NIL.

T and NIL evaluate to T and NIL. There are not *T* and F.

Interpreted variables, and variables used free in compiled functions, are automatically SPECIAL and may be used without restriction to communicate values. Also any PROG and LAMBDA variables in a compiled function may be declared SPECIAL, and will be bound and restored correctly. COMMON does not exist.

Flags are not allowed; elements on a property list of an atom are expected to be paired.

MAP, MAPCAR, etc. assume the <u>first</u> argument is the function, and the <u>second</u> is the list.

Defining of functions is usually done with DEFPROP.

DETAILS

Typein is to EVAL: one types OBJECT (followd by a space) to get the value of an atom, or (FN (QUOTE ARG1) (QUOTE ARG2) ...) to apply a function to arguments. (Frequently users arrange their top-level functions to be FSUBR/FEXPRs to avoid the necessity of quoting the arguments.) Carriage-return and line-feed characters are ignored. Space and comma are identical, and separate elements of lists; spaces are ignored when adjacent to parentheses, commas, and other spaces; spaces also distinguish dot notation from floating-point numbers and decimal integers. (In output, a space is always put to each side of a dot-notation period.) Slash (/) may be used to quote any character as part of a print name.
The only i-o devices used by PDP-6 LISP are the on-line Teletype, the DECtapes, and the display. Certain characters when typed in set input and output switches for top-level input, READ, and PRINT. Certain functions mentioned later feed data to the display.

Input-output switch selection characters:

V Teletype output on
W Teletype output off
R DECtape output on
T DECtape output off

} at a given time, either, neither, or both output devices may be selected

⌊Q DECtape input on
⌊S DECtape input off
} Teletype input is initially on; reading a DECtape end-of-file turns DECtape input off and reselects Teletype input.

⌊G Quit: immediately returns LISP to the top level.

These characters are not seen by READ or READCH.

THE ALLOCATOR

There are five data storage areas in LISP:

a) Free Storage: holds s-expressions
b) Full Word Space: holds character strings of print names; floating-point numbers; large fixed-point numbers.
c) Binary Program Space: holds compiled functions and arrays.
d) Special Pushdown List: holds bindings of all special variables.
e) Regular Pushdown List: holds return addresses for subroutine calls; bindings of all local variables; also is used by various internal routines.

When LISP is loaded, it types out ALLOC? and waits for the user to type in Y or N (yes or no). If Y is typed, LISP types out MEMTOP=. The user then types in an octal number (ended by a carriage return) which LISP will take as the highest register of available memory. Then the program similarly requests typed-in parameters for the size of Full Word Space, Binary Program Space, Special Pushdown List, and Regular Pushdown List. Free Storage is given all remaining space. For any typed-in number, carriage return alone may be typed in, and a standard value will be taken:

Parameter	Standard Value
MEMTOP	3717Ø
FULL WDS	4ØØ
BIN.PROG.SP.	1ØØØ
SPEC.PDL	1ØØØ
REG. PDL	1ØØØ

If N is typed as the answer to ALLOC? all standard values are taken. In case of error, RUBOUT will type an X and cancel the current number.

OBJECTS

Object	Indicator	Effect/value (if different from LISP 1.5)
NIL	VALUE	False value of predicates, explicitly tested for by COND. (EVAL NIL) is NIL (Nihil ex nihilo). (MAKNUM NIL (QUOTE FIXNUM)) is Ø. NIL ends lists.
T	VALUE	True value of predicates. (EVAL T) is T (Veritas numquam perit).
numbers		There are two kinds of numbers: fixed-point and floating-point. Fixed-point numbers ≥ Ø and < about $4ØØØ_{1Ø}$ are represented by a "pointer" 1 greater than their value, and no additional list structure. All other numbers use a pointer to full-word space as part of an atom header with a FIXNUM or FLONUM indicator. Numeric Type-in: A number typed in with one or more digits to the right of a decimal point is taken as floating-point; a number without a decimal point is taken as a fixed-point integer in the current input radix, IBASE (initially 8); a typed in number ended by a decimal point is a decimal integer. Numeric Type-out: Output of a floating-point number will have one or more digits to the right of the decimal point; output of a fixed-point number will be in the current output radix, BASE (initially 8), and if that radix is 1Ø LISP will end the numb with a decimal point. Arithmetic functions use "contagious floating poi ---if any operand is floating-point the result will be.

CAR	SUBR	CAR of an atom is the "pointer" -1.
CDR	SUBR	CDR of an atom is its property list.
CAAR, CADR, etc. SUBRs through CDDDDR		any combination of 1 to 4 A's and D's
CONS	SUBR	
NCONS	SUBR	Takes one argument, CONSes it to NIL.
XCONS	SUBR	CONS with arguments reversed; used to get arguments to a CONS evaluated in other order.
QUOTE	FSUBR	
ATOM	SUBR	ATOM of any number is T.
EQ EQUAL	SUBR SUBR	Neither EQ nor EQUAL will correctly compare a fixed-point number to a floating-point number. EQ will work for fixed-point numbers less than about 4ØØØ$_{10}$; otherwise use EQUAL. Floating-point numbers are considered equal only if their values are exactly equal.
COND	FSUBR	"COND pairs" of predicate and value may have other than two elements: if the first element evaluates non-null, each of the others is evaluated (in CAR-to-CDR order), and the value of the last is returned for the COND. If there is only one element, if it is non-null its value is returned. If no predicate is true, the value of the COND is NIL.
LIST	FSUBR	
NOT NULL	SUBR SUBR	Identical in effect.
RPLACA	SUBR	
RPLACD	SUBR	
NCONC	SUBR	
APPEND	LSUBR	Appends together any number of lists; evaluates its arguments in CAR-to-CDR order and copies the top level of all but the last argument.
READ	SUBR	
READCH	SUBR	Reads one character from selected input device.
PRIN1	SUBR	Prints any s-expression, inserting slashes before characters which would not otherwise be syntactically correct as part of an atom's print name.
PRINC	SUBR	Prints any s-expression; does not insert slashes. Both PRIN1 and PRINC do not space either before or after the material they print.

PRINT	SUBR	Identical to (PROG2 (TERPRI) (PRIN1 xx) (PRINC (QUOTE /))).
TERPRI	SUBR	Prints carriage-return, line-feed. Value is NIL.
LINEL	VALUE	Used by LISP as the number of character spaces in an output line.
CHRCT	VALUE	Number of character spaces left in current output line; if CHRCT is Ø and LISP outputs a character, it first inserts carriage-return-line-feed and resets CHRCT to LINEL.
INTERN	SUBR	Argument is pointer to atom structure; puts said atom on OBLIST and returns (probably new) atom pointer.
MAKNAM	SUBR	Argument is list of atoms whose print names are single characters (actually it takes the first character of each print name). Value is pointer to s-expression (the atoms in which are not automatically put on the OBLIST) which if printed out would be the concatenation of the single characters taken as arguments.
READLIST	SUBR	Like MAKNAM, but automatically INTERNs any atoms appearing in the resulting s-expression.
EXPLODE	SUBR	Argument is s-expression; value is list of atoms whose print names are single characters, which concatenated would form the print of the argument. For example, (EXPLODE (QUOTE FOO)) has the value (F O O). EXPLODE, like PRIN1, inserts slashes, so (EXPLODE (QUOTE FOO/ BAR)) PRIN1's as (F O O // / B A R) or PRINC's as (F O O / B A R).
EXPLODEC	SUBR	EXPLODEC is to EXPLODE as PRINC is to PRIN1. Example: (EXPLODEC (QUOTE FOO/ BAR)) would PRIN1 as (F O O / B A R) or PRINC as (F O O B A R).
FLATSIZE	SUBR	Argument is s-expression; value is number of characters in the argument if the argument were printed out with PRIN1.
TYO	SUBR	Takes one argument, a fixed-point number; outputs (like PRINC) a character whose ASCII code is the integer. Its value is not useful.
REVERSE	SUBR	Reverses top level of a list.
EVAL	SUBR	Takes 1 or 2 arguments. Second should not be given unless it is desired to use other than the current a-list.
APPLY	SUBR	(APPLY fn (args) alist) or (APPLY fn (args)).
MEMBER	SUBR	uses EQUAL
MEMQ	SUBR	like MEMBER, but uses EQ.
SASSOC	SUBR	
ASSOC	SUBR	
SUBST	SUBR	
GENSYM	SUBR	GØØØØ, GØØØ1, etc.

PROG2	SUBR	Second of any number of arguments.
MAPLIST	SUBR	(MAPLIST fn list)
MAPCAR	SUBR	(MAPCAR fn list)
MAP	SUBR	Like MAPLIST, but returns NIL; does no CONSes.
MAPC	SUBR	Like MAPCAR, but returns NIL; does no CONSes.
LENGTH	SUBR	Returns fixed-point number. λ[[ℓ]; [atom [ℓ] → Ø; T → add1 [length [cdr [ℓ]]]]]
LAST	SUBR	λ[[ℓ]; [atom [cdr [ℓ]] → ℓ; T → last [cdr [ℓ]]]]
PLUS	LSUBR	A + B + C + ...
TIMES	LSUBR	A * B * C * ...
DIFFERENCE	LSUBR	A - B - C - ...
MINUS	SUBR	- A
QUOTIENT	LSUBR	A / (... * C * B)
REMAINDER	SUBR	Works only for fixed-point numbers.
ADD1	SUBR	Result is fixed or floating point, same as the argument
SUB1	SUBR	
NUMBERP	SUBR	
GREATERP	SUBR	
LESSP	SUBR	
ZEROP	SUBR	Works only for fixed-point Ø.
MINUSP	SUBR	
REMOB	FSUBR	Takes any number of atomic arguments; value is NIL.
OR	FSUBR	Returns first non-null argument or NIL; does not evaluate arguments past the one it returns.
AND	FSUBR	Returns last argument or NIL; does not evaluate arguments past the first NIL.
PROG	FSUBR	
SET	SUBR	Use SET, SETQ instead of CSET, CSETQ.
SETQ	FSUBR	
GO	FSUBR	
RETURN	SUBR	

GET	SUBR	λ[[a;b]; [null [cdr [a]] → NIL;
		eq [cadr [a]; b] → caddr [a];
		T → get [cddr [a]; b]]]
		A typical use: (GET (QUOTE atom) (QUOTE indicator)) returns the property (NIL if it is absent).

GETL	SUBR	Similar to GET, but second argument is a list of indicators. Value returned is CD...DR of first argument such that (CAR (GETL ...)) is the indicator and (CADR (GETL ...)) is the property. GETL, lile GET, stops at the first satisfactory pair on the property list.
MAKNUM	SUBR	Turns a pointer (machine address) into a number: the second argument should be (QUOTE FIXNUM) or (QUOTE FLONUM) to determine the type of the result.
BOOLE	LSUBR	Used in the form (BOOLE n a b c ...). A 36-bit bitwise Boolean operation is performed Ø and a, the result and b, etc. The number n selects the operation, according to the following chart: bit of a/bit of b $\begin{array}{ccc} & Ø & 1 \\ Ø & n_1 & n_2 \\ 1 & n_3 & n_4 \end{array}$ where n_i is the ith bit of n. Examples of n: 1 is LOGAND; 7 is LOGOR; 6 is LOGXOR.
TIME	SUBR	Returns value of 6Ø-cycle elapsed time counter.
SETTIME	SUBR	Sets (TIME) counter to value of argument.
FIX	SUBR	Argument is fixed- or floating-point number; value is (truncated) fixed point value of argument.
GC	SUBR	Takes no arguments, causes garbage collection, returns value NIL.
GCGAG	SUBR	(GCGAG T) turns on typeout of statistics for each garbage collection. (GCGAG NIL) turns it off.
SPEAK	SUBR	Has value of CONS counter.
EXAMINE	SUBR	Argument is number, which is taken as an absolute machine address; value is contents of said address as a fixed-point number.
DEPOSIT	SUBR	First argument is an address, as for EXAMINE; second argument is a fixed-point number to be deposited thereinto.
PUTPROP	SUBR	Adds a property to an atom. (PUTPROP (QUOTE atom) (QUOTE property) (QUOTE indicator)) Value is the property.
REMPROP	SUBR	Removes a property. (REMPROP (QUOTE atom) (QUOTE indicator)) Value is T if property was there, NIL otherwise.
DEFPROP	FSUBR	A common top-level defining function: like PUTPROP, except (a) arguments are not evaluated, (b) value is the atom.
BAKGAG	SUBR	(BAKGAG T) enables backtrace printout on any LISP error; (BAKGAG NIL) disables it. A backtrace is printed as a series of function calls, most recent (deepest) first:

fn1-fn2	fn1 calls fn2
fn1-EVALARGS	arguments being evaluated preparatory to calling fn1
fn1-ENTER	fn1 being evaluated
?-fn1	? represents an internal routine

ERRSET	FSUBR	(ERRSET (fn args...)) has the value NIL if an error occurs while evaluating (fn args...) and LIST of the value of (fn args...) otherwise.
ERR	SUBR	Causes a non-printing error.
BASE	VALUE	Radix of fixed-point number output. May be modified by SETQ.
IBASE	VALUE	Radix of fixed-point number input. May be modified by SETQ.
BPORG	VALUE	Current lowest unused location of Binary Program Space.
BPEND	VALUE	Highest location available for use as Binary Program Space.
ARRAY	FSUBR	(ARRAY name par dim1 dim2 ...) sets up <u>name</u> as an array (actually as a SUBR). <u>par</u> should be T to protect array elements from garbage collection; otherwise NIL. Then (name indx1 indx2 ...)
STORE	FSUBR	returns the value of an element, and (STORE (name indx1 indx2 ...) newval) sets the value of an element. An array may have no more than 5 dimensions; indices run from Ø to dim-1.
NSTORE	FSUBR	(NSTORE (name indx1 indx2 ...) number) deposits the low-order 18 bits of the <u>number</u> in the array. Both NSTORE and STORE evaluate their second argument before their first.
COMPILE	FSUBR	Arguments are names of EXPRs and FEXPRs. COMPILE uses PRINT to output each argument function in LAP-readable machine language.
LAP	FSUBR or FEXPR	(LAP name indicator), where the indicator is SUBR, FSUBR, or LSUBR, causes LAP to call READ repeatedly, each time reading one tag or storage word. An atom is taken as a tag, except for NIL which indicates the end of the function being read; a non-atomic s-expression is taken as a storage word in the following format: (Inst Acc Adr) or (Inst Acc Adr Indx). <u>Inst</u> should be a PDP-6 instruction mnemonic, optionally suffixed @ for the indirect bit. <u>Acc</u> should be a number Ø - 17 or P, the push-down pointer. <u>Adr</u> may be a numeric machine address, a tag in that function, a negative number, one of certain symbols for entry points to LISP internal routines, or a list in one of the following forms: (QUOTE atom) a pointer to <u>atom</u>; (SPECIAL atom) the special/value cell of <u>atom</u>; (E atom) like QUOTE, used when <u>atom</u> is the name of a function being called; (C w x y z) a constant, i.e. a location containing storage word (w x y z). <u>Indx</u> is an optional left-half quantity, such as an index register specification; <u>Indx</u> takes the same form as <u>Adr</u>.

The best way to get hand-coded functions into the system is with LAP. A SUBR may have no more than 5 arguments. The value of the first is expected in register 1, that of the second in 2, etc. The value of a function is returned in register 1. Locations 1 through 7 are the <u>ONLY</u> accumulators available for use within a subroutine. An FSUBR has one argument, in 1. (The current a-list may be gotten by calling *AMAKE.)

An LSUBR may have any number of arguments: their values are on the pushdown list, last argument nearest the free end. The first instruction an LSUBR performs must be (JSP 3 *LCALL).

Four UUO (trap) instructions are available for LAP: CALL, JCALL, CALLF, and JCALLF. These are to be used for function calls in the form (CALL n (E funct)) where the number n is the number of arguments being transferred (or a code for the type of function being called) as follows:

Ø - 5 calling SUBR or EXPR, Ø to 5 arguments.
16_8 calling LSUBR, arguments on pushdown list, -(number of arguments) in register 6.
17_8 calling FSUBR or FEXPR, argument list in 1.

When one of these UUOs is first executed, the UUO handler will:
call the interpreter if calling an EXPR or FEXPR;
otherwise in the case of CALL will change the CALL UUO to a PUSHJ to the function code and execute said PUSHJ; in the case of JCALL will change the JCALL to a JRST to the function code and execute said JRST: in the case of CALLF will PUSHJ to the function but not change the UUO; in the case of JCALLF will JRST but not change the UUO.

The F forms of UUO are necessary to call functions whose names are computed; the J forms save code in the case of (RETURN (fn ...)).

SPECIAL	FSUBR	Part of the compiler. Takes any number of arguments, which are variable names. This defines them as SPECIAL to the compiler. All variables are either SPECIAL, or local to a particular compiled function. Communication of variable values between functions, and within interpreted functions, must be done with SPECIAL variables. All interpreted variables are automatically SPECIAL. Except as countermanded by a (SPECIAL ...), the compiler assumes all LAMBDA- and PROG-variables in functions it compiles are local to said functions. Free variables used in a function being compiled are assumed SPECIAL, and the compiler prints out (var UNDECLARED). If a function not defined is called by a function being compiled, the compiler prints out (func UNDEFINED) and assumes it will be a SUBR or EXPR.
DISLIST	VALUE	Set by the user, and taken by LISP, as a list of arrays to be displayed on the scope. For the case of one such array, the form is (SETQ DISLIST (LIST (GET (QUOTE arrayname) (QUOTE SUBR)))). The successive array elements are the actual 18-bit data words sent to the display.
DISINI	SUBR	(DISINI (QUOTE arrayname)) initializes an array to be displayed from. An array should not be put on the DISLIST until it has been DISINIed. DISINI erases the previous contents of the array.
DISAD	SUBR	(DISAD (QUOTE arrayname) par arg) in effect performs a PRINT or PRINC (according as par evaluates to T or NIL) of arg into an array being displayed. In other words, the text of an s-expression is appended to the text currently in such an array.
DISCNT	VALUE	Number of character spaces remaining in current line on scope. (DISAD inserts carriage-return, line-feed when this reaches Ø and resets it to LINEL.)

DISINI puts data in the array to initialize the display at the upper left corner. Hence if more than one such array were on the DISLIST their contents would overwrite each other on the scope. However, by the function NSTORE it is possible to put data in an array for display other than the upper-left initialization of DISINI and the character strings of DISAD. In that case it might be useful to have more than one displayed array. For further details on the display feature, consult a PDP-6 systems programmer.

PNAME		Indicator for print name structure.
FIXNUM		Indicator for fixed-point number.
FLONUM		Indicator for floating-point number.
VALUE		Indicator for value property (similar to APVAL of LISP 1.5): the value is actually CDR of the property (CAR of it is NIL). So (SETQ FOO (QUOTE (A B C))) has the same effect as (DEFPROP FOO (NIL A B C) VALUE).
LAMBDA		
LABEL		
SUBR		Indicator for subroutine.
FSUBR		Indicator for special-form subroutine.
ARG	SUBR	The format (LAMBDA VAR (form ...)), where the "LAMBDA-list" is an atom, is interpreted as a function which takes any number of arguments. The bound variable VAR takes the value of the number of arguments actually supplied at any given call. The values of these arguments are available with the notation (ARG 1), (ARG 2), etc. The compiler will compile such an EXPR into an LSUBR, which will perform the same.
LSUBR		Indicator for LSUBR.
EXPR		Indicator for s-expression function.
FEXPR		Indicator for s-expression special form. A FEXPR's LAMBDA-list may have 1 or 2 elements.
SYM		Indicator for symbol value used by LAP.
FUNARG		
MACRO		Indicator of a MACRO property, expanded at compile time (and properly simulated by the interpreter). The property should be a function of one argument. When a call is encountered to a function with a MACRO property, the list which is the function call is fed, unevaluated, to the macro definition as the one argument. (E.g. CAR of the argument is the function name.) Then the MACRO property function is expanded, recursively if necessary, with CAR, CADR, etc. of its argument replaced by the elements of the function call as written. For example, CONSCONS might be defined as a macro in the following way:

```
 (DEFPROP CONSCONS (LAMBDA (L)
                                         (COND ((NULL (CDDR L)) (CADR L))
                                               (T (LIST (QUOTE CONS)
                                                        (CADR L)
                                                        (CONS (CAR L)
                                                              (CDDR L))))))
  MACRO)
```

in which case (CONSCONS A B C) would expand to
(CONS A (CONSCONS B C)), then to
(CONS A (CONS B (CONSCONS C))), and finally
(CONS A (CONS B C)) which is what would be compiled
or interpreted.

OBLIST	VALUE	The object list, a list of buckets of atoms.
SPECBIND	SYM	Called to cause new-level bindings of special variables. A use would resemble (JSP 6 SPECBIND) (Ø Ø VAR1) (Ø Ø VAR2) etc. (INST...)
SPECSTR	SYM	Called by PUSHJ to restore most recent batch of special bindings.
NUMVAL	SYM	(PUSHJ P NUMVAL) assumes a LISP number in 1; returns the 36-bit numeric value in 1 and the type (FIXNUM or FLONUM) in 2.
FIX1A	SYM	(PUSHJ P FIX1A) turns the actual fixed-point number in 1 into a LISP number.
*PLUS	SUBR	PLUS of 2 arguments.
*TIMES	SUBR	TIMES of 2 arguments.
*DIF	SUBR	DIFFERENCE of 2 arguments.
*QUO	SUBR	QUOTIENT of 2 arguments.
*APPEND	SUBR	APPEND of 2 arguments.
FUNCTION	FSUBR	Does not cause FUNARG binding when interpreted.
*FUNCTION	FSUBR	Causes FUNARG binding when interpreted; otherwise identical to FUNCTION.
IOC	FSUBR	In-out device selection. The argument is EXPLODEd; the resulting character string is modified as if it had been typed in with the CTRL key held; and then it is fed to the i-o switch processor. for example, (IOC VR) has the sme effect as typing in <u>VR</u>.
IOG	SUBR	(IOG chars (FUNCTION fn)) saves up the current state of all i-o switches; then feeds the <u>value</u> of <u>chars</u> at the i-o switch processor; then performs <u>fn</u>; then restores the old i-o switch settings.
NOUUO	SUBR	(NOUUO T) prohibits the UUO handler from changing UUO calls to PUSHJ or JRST. (NOUUO NIL) restores that ability.
*EVAL	SUBR	EVAL of 1 argument.
*RSET	SUBR	(*RSET T) prevents restoring bindings of special variables when returning to the top level due to an error. (*RSET NIL) resumes such restoring.

CURRENT PROBLEMS

If in the list of performed values of a COND argument of a PROG2 list there occurs a RETURN or GO, any following values will be performed if the expressi
is interpreted, but not if it is compiled.