~~J. McCarthy~~
LISP

LISP 1.5 IMPLEMENTATION

on the

CD 3600

and the

IBM SYSTEM /360 SERIES

by

J. G. Kent

CONTENTS

1 INTRODUCTION

This paper describes certain aspects of the implementation of a LISP 1.5 interpreter on the CD 3600. The CD 3600 is a 1's complement binary computer with a 48-bit wordlength and 1 to 8 banks of 32768_{10} words of storage. Core speed is 1.5 microseconds. SCOPE, the monitor for CD 3600, occupies 6000_{10} words of storage.

The machine has an accumulator A, an accumulator extension Q, and a flag register D. A, Q and D are all 48 bits long. In addition there are six 15 bits index registers B1-B6 and various other registers.

This paper also indicates some features of the IBM System /360 Series which will make it possible to write a more efficient interpreter for these computers.

The implementing of LISP 1.5 on CD 3600 was performed at the Kjeller Computer Installation, Kjeller, Norway, as the main part of the author's thesis work for his M.A. degree. As several installations have asked me to make a LISP 1.5 interpreter for the IBM System 360, I have started to do this at the University of Waterloo, Waterloo, Ontario, Canada.

It is assumed in the following that the reader has a good working knowledge of the LISP 1.5 Programmer's Manual, (5).

The readers who want to look more closely at the implementation of LISP 1.5 on CD 3600 should obtain my M.A. Thesis (1). The thesis contains among other things a complete description (with some flowcharts) of my implementation of LISP 1.5 on CD 3600 (hereafter called LISP3600).

2 NEW FEATURES IN THE LISP3600 SYSTEM

The LISP3600 system is an interpretative LISP system modelled after the original LISP 1.5 system for the IBM 7090. Care has been taken to ensure compatibility between these two versions.

The actual implementation of this interpreter differs in some important respects from the original version to increase the efficiency.

The most marked differences are in the organization of storage, where the idea of a separate block for "fullwordstorage" has been abandoned, and in the organization of the internal representation of LISP-atoms. Several of the indicators needed on the property lists in LISP 1.5 has been rendered unnecessary. Note that the interpreter and the initial object list are assembled relocatable. LISP3600 operates under control of the standard operating system SCOPE for the CD 3600.

2.1 Organization of the system

2.1.1 Organization of storage on a one bank CD 3600.

Core store is distributed according to this figure.

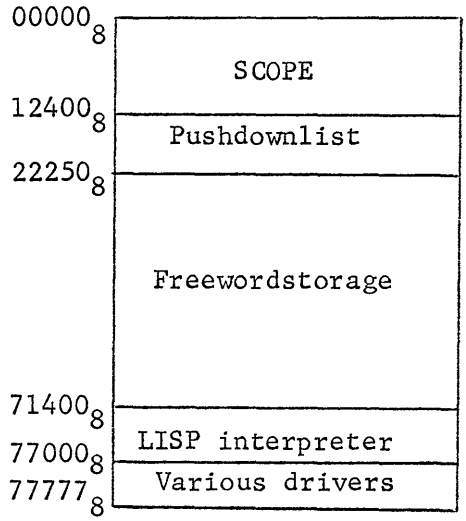


Figure 2.1 Organization of core store on a onebank CD 3600

The boundary between freewordstorage and the pushdownlist is fixed though easy to reset when reassembling the LISP-system.

One a onebank CD 3600 the length of the pushdownlist has been set to 4000_{10} words. The interpreter occupies about 2850_{10} words and freewordstorage the rest of core store, about 20000_{10} words. This compares favourably with LISP 1.5 which by excising LAP and the compiler has 16300_{10} words of free- and fullwordstorage and 2560_{10} words of pushdownlist.

2.1.2 Organization of storage on a twobank CD 3600

Core store is distributed as follows.

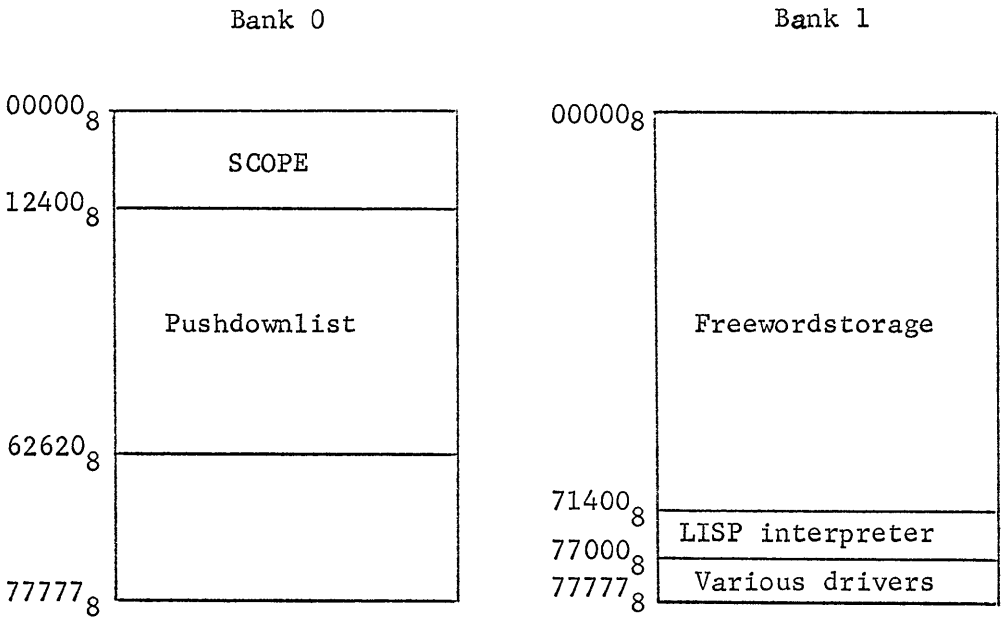


Figure 2.2 Organization of core store on a twobank CD 3600

On a twobank CD 3600 the pushdownlist is in another bank than the interpreter and freewordstorage. This means that the length of freewordstorage is increased to 24000_{10} words. The length of the pushdownlist has been set to 20000_{10} words.

2.1.3 Arguments and registers

Between LISP-functions arguments are transmitted through the A-register, Q-register and the locations ARG3,...

The A-register is used for transmitting information to and from the pushdownlist. Only the contents of one word is saved at a time.

The value of a function is always held in the A-register when returning from the function. The D-register contains several indicators (binary switches) needed in the interpreter. Information about the status of the interpreter can then be read out of the bit for bit displayed D-register on the console.

2.1.4 Object list

That part of the object list which contains the standard atoms has been generated in assembly language. The object list is sequential in LISP3600, and not bucket sorted as in LISP 1.5. This means it is very easy to generate the object list by using the ECHO feature of COMPASS.

2.1.5 Atoms

The atoms and their property lists have been reorganized in LISP3600. All LISP cells having bit 47 set are so called atomheads. An atomhead contains in its upper

address a pointer to the atom's fullwordlist and in the lower address a pointer to the atom's property list.

The atom EXAMPLE with an empty property list:

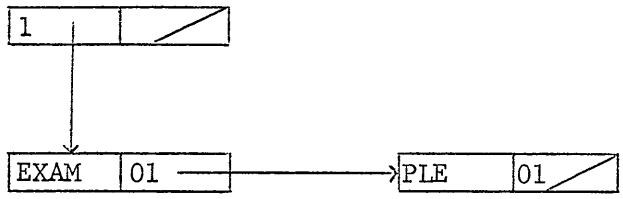


Figure 2.3 The atom EXAMPLE

2.1.6 Property lists

A typical property list might look like this:

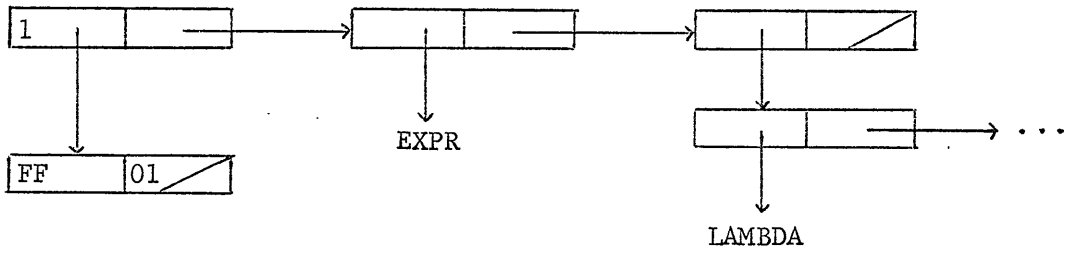


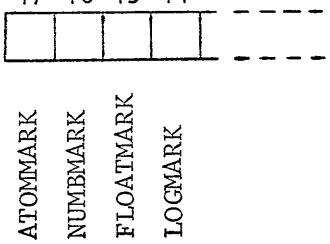
Figure 2.4 Property list of the atom FF

FF is as we see a function namely an EXPR which starts this way (LAMBDA (X) ...).

2.1.7 Binary markers

A LISP-cell is one word on the CD 3600. Since the word length is 48 bits, and only 15 bits are needed to express an address, 9 bits in the upper halfword and 9 bits in the lower halfword are released for other uses. As mentioned above

bit 47 set indicates that this word is an atomhead.



47	46	45	44	
1	0	0	0	BCD ALFAMER
1	1	0	0	FIXPOINT
1	1	1	0	FLOATPOINT
1	1	0	1	LOGICAL

Figure 2.5 Markers and their meaning in the atomhead

The bits 46, 45 and 44 refer to the fullwordlist associated with the atom.

A function that is to be traced has bit 39 set. This means that the indicator TRACE is unneeded.

When bit 22 is set this indicates that the word in question is a fullword. Bit 23 is used by the garbage collector to mark active words.

2.1.8 Fullwords

The fullwords in freewordstorage replace the "fullwordstorage" in LISP 1.5.

A fullword is a word with bit 22 set and the upper 24 bits occupied by either:

a) Four BCD characters from a printname. (If need be filled in from the right with blanks.)

or

b) 24 bits from a 48 bit number.

or

c) The address of a binary LISP-routine (SUBR or FSUBR).

2.1.9 Printnames

All nonumeric atoms have in their upper address of their atomhead the address of a linear list of their BCD printnames. For instance the atom DIFFERENCE has this fullwordlist:

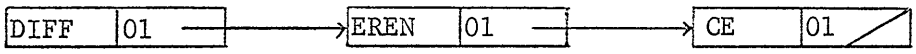


Figure 2.5 The fullwordlist of the atom DIFFERENCE

2.1.10 Numbers

There are three kinds of numbers:

a) Fixed point (integers).

b) Floating point.

c) Logical (octal).

All are stored as 48 bit binary numbers with the help of two fullwords, and must be converted to BCD in input and output. (The BCD representation of a number is not stored).

3 EXTENT OF IMPLEMENTATION

3.1 Extensions

a) Alphameric atoms may in LISP3600 have up to 82 characters.

b) Fixed point numbers may have absolute values up to 2^{47}.

c) Floating point significance on input is 10 digits.

d) Floating point numbers may have values between 10^{307} and 10^{-307}.

e) Numbers are considered equal if the absolute values of

their difference is less than 10^{-8}.

f) A completely new function called APPEND1 is included as a SUBR. See reference 1 for details.

g) CAR of an atom is not junk as in LISP 1.5 but the address of the fullwordlist of that atom.

h) Three peculiar new functions called SETBIT, CLEARBIT and TESTBIT exist as SUBRs. They can be used to set, clear and test bits in the D-register.

i) Tracing will only occur if SETBIT(7) has been evaluated before tracing is to start. This means that evaluation of functions are faster when SETBIT(7) have not been executed because all the tests for tracing in EVAL are skipped.

j) Whenever an error occurs in LISP3600 the lists bound on the pushdownlist is printed out. This holds true for all runtime errors except the STACK EXCEEDED error. The most recently used list in the stack (the list on top) is printed last. The last printed lists will therefore give a good indication of what caused the error.

3.2 Omissions

a) The following functions are not implemented: LAP, COMPILE, EXCISE, ARRAY, ERRORSET, RECLAIM, COUNT, UNCOUNT, TRACECOUNT and SPEAK.

b) No control cards of any type exists in LISP3600. A LISP3600 run consists of a single packet ending with a card containing a 4-8 punch in column 1. This last card acts as an end-of-file to LISP and prevents LISP from reading into the next job.

3.3 Differences

a) The scale factor in a logical number is an exponent to the base 2.

b) A minus sign preceding a logical number will cause the logical number to be complemented after an eventual shifting.

c) Blanks are used as fill-in in the fullwords. This makes it impossible to print more than one blank at a time. But this means that the constant $$ B$ will print as a single space.

d) The function CLEARBUFF has not been implemented because it is never needed.

e) The functions INTERN and MKNAM have been combined into a single function namely MKATOM.

MKATOM = INTERN(MKNAM)

f) Because of the reorganization of all property lists, the printname is CAR of the atom.

g) UNPACK takes an atom as its argument.

h) PRINT should not be used directly after PRIN1 without executive TERPRI in between, because PRINT sets the output buffer to blanks before printing thereby destroying what was put in by PRIN1.

i) GO must only be given atomic labels.

j) + and - should never be used as characters in an atom.

4 LISP3600 VERSUS OTHER LISP SYSTEMS

4.1 LISP 1.5 (IBM 7090)

Identical LISP programs have been run on the IBM 7090 using the LISP 1.5 system, and on the CD 3600 using the LISP3600 system. The execution times thereby obtained showed that programs are executed from 15 to 20 times faster under the LISP3600 system. Even if we allow for the difference in speed between IBM 7090 and CD 3600 the LISP3600 system should be about three times as fast as the LISP 1.5 system.

A few runs that could not be run under LISP 1.5 were run successfully under LISP3600 on a onebank CD 3600, because of the slight increase in the length of freeword-storage and the pushdownlist in the LISP3600 system. Several runs with extremely heavy recursion that could not be run under the LISP 1.5 system were run successfully under LISP3600 on a twobank CD 3600, the reason being the very long pushdownlist available on a twobank CD 3600.

4.1.1 Suggested reasons for the increased speed of the LISP3600 system

a) Better instruction repertoire on the CD 3600. This is discussed in the next section.

b) CD 3600 has six index registers as opposed to IBM 7090's three.

c) The reorganization of property lists that eliminates the search for an atom's printname. This is especially important in the handling of numbers. The address of the fullwordlist containing the number is always CAR of its atom. The type of

the number in question is indicated by binary markers in that number's atomhead.

d) The slightly larger freewordstorage in the LISP3600 system, which means that garbage collections do not occur as often as in the LISP 1.5 system.

It must however be remembered that in the LISP 1.5 system there exists a compiler. By having the most important functions compiled, the execution time of big programs can be significantly reduced. The LISP3600 system does not have a compiler.

One of the reasons why the interpreter in the LISP3600 system could be made shorter than its LISP 1.5 counterpart, is the introduction of so-called "combined arithmetic routines". All functions which are equal in all respects save the actual operation involved are combined into single routines with different entry points. For instance the functions PLUS and TIMES both use the same routine PLUSTIME, with the two entry points PLUS and TIMES. PLUSTIME (as the other combined routines) performs the correct operation by executing it indirectly through the location ADR., which is loaded with the address of the correct instruction at the entry point. In other words the address of an addinstruction is placed in ADR. at the entry point PLUS prior to transferring to PLUSTIME.

4.2 The AN/FSQ-32/V LISP system (LISPQ32) (2)

An algebraic simplification program (3) written in LISPQ32, has been modified (4) so that it may be run on the LISP3600 system. Comparing almost identical examples run on

both systems has shown that the program is between ten and thirty times slower when run on the LISP3600 system. Even though the computers are of approximately the same speed the result is very favourable for the LISP3600 system. This may seem a strange conclusion until one considers the fact that the LISPQ32 system is a compiler oriented system while the LISP3600 system is completely interpretative.

5 SOME USEFUL FEATURES OF THE CD 3600

5.1 The D-register

The D-register cannot load words from or store words in memory directly. It was however very useful as a flag register. The D-register keeps track of the status of the interpreter by various bit combinations, which are set, cleared or tested by the interpreter. An added advantage of the D-register is the fact that it is displayed bit for bit on the console of the CD 3600. Information about the status of the LISP3600 system is therefore readily available.

This made debugging easier and was also used to time certain routines in the interpreter such as the garbage collector.

5.2 Registers

The increased number of index registers (6) and the very good inter-register instructions in many cases made the storing and restoring of registers unnecessary.

5.3 Addressing

Double indexing and indirect addressing is always useful when one is doing list processing.

5.4 Instructions for operations on bits

Instructions for setting, clearing and testing (ZBJP and NBJP) any bit in any register made it easy to test for the various markers used in some LISP-cells.

5.5 Instructions for operations on bytes

The instructions SBYT and LBYT, which can store and load a byte from the A-register or the Q-register, and the instruction SCAN which can compare any byte in storage with a byte in the A-register or in the Q-register, were used frequently throughout the interpreter. A byte may be specified to be of any length between 1 bit and 48 bits in these instructions.

5.6 The return jump instructions

Almost all linkages in the interpreter utilize the return jump instructions. These instructions store the return address in the first instruction in the routine they are jumping to. Control is then transferred to the second instruction in the routine in question. By executing a jump back to the first instruction in the subroutine a correct jump will be made from there back to the calling program with the aid of the address stored there by the return jump instruction.

5.7 Discussion of the locate list element instruction

The locate list element instruction (LSTU/L) has not been used in the interpreter at all. LSTU/L scans a liststructure containing two 18 bit addresses in each word in the same way as the two 15 bit addresses carried in all LISP-cells. LSTU/L scans down a liststructure for the n'th element. This is done by either using all upper addresses (LSTU) or all lower addresses (LSTL) in the n-1 preceding elements in the liststructure. It will in other words simulate a CAR chain or a CDR chain. The instruction requires however the setting up of two index registers and is fairly slow. If going down only one or two elements in a liststructure it is much faster to use the index register load instructions with indexing. Another disadvantage of LSTU/L is the fact that it considers a word containing an address of zero to be the last word in a list. This would have been very well indeed if the atom NIL could have its atomhead in address zero. This is however impossible since the word with address zero in storage is used by the interrupt system on the CD 3600. If LSTU/L reaches a word containing an address of zero before the n'th element has been found it will terminate scanning and give as a result address zero instead of the address of the word which contained the address zero, which would have been more useful.

LSTU/L would however have been very useful if the LISP3600 system had been designed for a multibank CD 3600. As the system is now, freewordstorage must be wholly contained in

a single bank, because the LISP-cells only contain 15 bit addresses. A 15 bit address can only address 32768_{10} words of storage which is one memory bank on the CD 3600. To address any word in any of the eight possible memory banks an address of 18 bits is required. LSTU/L would come in very handy in this case because this instruction scans liststructures containing 18 bit addresses. A liststructure containing 18 bit pointers could weave in and out of banks with no difficulty.

5.8 Arithmetic instructions

The arithmetic instruction set is very good. It is for example possible to convert a number from integer to floating point or vice versa using only three instructions.

5.9 The ECHO facility

As already mentioned above, the ECHO facility in CD 3600's assembly language made the generation of the initial object list easy. The ECHO feature is a macro-like feature whereby a specified number of instructions can be repeated a specified number of times with parameter substitution.

5.10 SCOPE loading procedures

Because of the loading procedures in SCOPE, the monitor for CD 3600, it is very easy to find out at run time how much storage is left over. The interpreter will therefore always utilize all available space for freewordstorage. This is only partly true on a multibank CD 3600, where only the highest numbered bank will be fully utilized.

6 TWO FEATURES THAT WOULD HAVE MADE CD 3600 MORE SUITED TO LIST PROCESSING

6.1 Addressing

It would have been easier to make a LISP system utilizing all available banks if the index registers had been 18 bits long, and addressing had been performed via an indexregister to get the required 18 bit address. The bankswitching that has to be performed now with the aid of special 3 bit bank registers is cumbersome.

6.2 Indexing with the A-register

Some of the functions in LISP such as CAR and CDR could have been made shorter and faster if it had been possible to use the lower 15 bits of the A-register as an index register. As it is now the transmission of the lower address of the A-register to and from index registers is very frequent.

7 PRELIMINARY REMARKS ABOUT LISP/360

The implementation of LISP 1.5 on IBM System /360 (LISP/360) has already started. The system will be modelled after the LISP3600 system. It is however our hope that the system will eventually contain all the unimplemented functions of LISP3600 including LAP and COMPILE. We will also try to make the garbage collector compacting and the object list bucket sorted. The LISP/360 system will be made in such a way that it can utilize the so-called Large Capacity Storage that is available for the IBM System /360 Series computers.

7.1 Some conventions

7.1.1 LISP-cells

A LISP-cell will in the LISP/360 system be one doubleword. This has several advantages:

a) Each LISP-cell can then contain two full 24 bit addresses, which means that freewordstorage may utilize all available store on any IBM/360 computer.

b) Single precision numbers can be stored in a single fullword. This will increase the speed of arithmetic in LISP/360 considerably.

c) Space is left over in the LISP-cell for binary markers as in the LISP3600 system. Since the space left over is one byte in the upper word and one byte in the lower word, the test under mask instruction (TM) makes it easy to test these markers.

7.1.2 Register use

Some of the 16 registers available on the IBM/360 has been assigned special tasks.

Register	Task
2	Internal linkage register
6	Contains the address of NIL
7	Stack pointer
8	A-register
9	Q-register
	Used for transmitting arguments
12	Base register for the interpreter
13	Contains a pointer to a save area used by systemprograms for storing the registerblock

7.1.3 Storage allocation

A system for storage allocation for freewordstorage and the pushdownlist that would suit everybody's needs has not been found. Three proposals have been made:

a) Let the amount of space set aside for the LISP/360 system be an assembly parameter.

b) Issue the GETMAIN macroinstruction continuously immediately after loading, until all available space is under control of the LISP/360 system. Since total amount of core needed for a job must be specified on the job card, this proposal is just another way of doing scheme a) with the added advantage that the space set aside for freewordstorage and the pushdownlist is a job parameter. Operating System /360 will not allow a job to use more space than what is specified on the job card.

c) Issue the GETMAIN macroinstruction only when more space is needed. A certain amount of storage will be made available initially as an assembly parameter.

The GETMAIN macroinstruction codes in a call to an Operating System /360 routine, which will try to assign the specified amount of storage available to the program issuing the GETMAIN.

The very first version of the LISP/360 system will use proposal a) for its storage allocation.

Other ways of allocating storage may be necessary when one wants to utilize the Large Capacity Storage.

7.1.4 Relocation

Several relocation schemes have been considered. More information is needed about Operating System /360 Option 4 and the Roll in - Roll out feature. In the first version of the LISP/360 system every LISP-cell will carry full 24 bit physical addresses. This means that no relocation of freewordstorage or the pushdownlist will be possible once it has been loaded into storage.

8 SOME USEFUL FEATURES OF THE IBM/360

8.1 Registers

The 16 general registers in which both indexing, arithmetic and logical operations may occur. However register 0 cannot be used for indexing. In the routines so far written the increase in the number of registers and the fact that they can be used for indexing has reduced the number of instructions considerably.

8.2 Addressing

The addressing scheme of the IBM System /360 which makes bytes and words just as easily addressable, have made programming of the interpreter simpler.

The 24 bit address of the IBM System /360 which means that about 16 million bytes are immediately addressable seems to be just what is needed in list processing. This fact coupled with the availability of Large Capacity Storage whereby present IBM/360 computers can get up to 8 million bytes of continuously addressable core store, may prove to be of great importance in LISP processing.

References

1. Kent, J. G. - An Interpretative System for the Programming of Recursive Functions on a Digital Computer, Intern rapport E-88, Norwegian Defence Research Establishment, Kjeller, Norway. (1966)

2. Saunders, S. A. - The LISP System for the Q-32 Computer, in the book, "The Programming Language LISP: Its Operation and Applications", Information International Inc., Cambridge, Massachusetts. (1964)

3. Korsvold, K. - An On-line Algebraic Simplify Program, Stanford Artificial Intelligence Project Memo 37, California. (1965)

4. Ødmansson, E. - Applications of the Programming Language LISP, Intern rapport E- , Norwegian Defence Research Establishment.

5. McCarthy, J. et. al. - The LISP 1.5 Programmer's Manual, MIT Press, Cambridge, Massachusetts. (1962)

6. - Control Data 3600 Computer System Reference Manual, Pub. No. 60021300. (July 1964)

7. - IBM System /360 Principles of Operation, Form A22-6821-4. (1966)