A DESIGN AND A IMPLEMENTATIONS OF PERSONAL LISP SYSTEMS CONSIDERING IE

Masayuki Ida

Computer Science Research Group
Department of IE & SE
Aoyama Gakuin University
6-16-1 Chitosedai Setagaya, Tokyo, Japan 157

Abstract

The design philosophy and a two implementations of the experimental lisp named APCL (Aoyama Personal Common Lisp), and the features considering to be IE work stations with AI facility, are described. APCL is a common lisp subset with extended communication facilities named LLCP and RPS, "exec" feature, interrupt handling, general I/O handling and graphics. There are two implementations with a same function set. One is Lisp05, for 8086 based personal computer with CP/M-86 or CCP/M, which is 16 bits based but whose space is specially multi-segmented. The other is for ALPS/III, an 80286 based specialized engine attached to a host. ALPS/III-lisp has 32 bit capability.

Introduction

IE needs Lisp

In IE related fields, expert systems or application programs with intelligence such as ISA,IMACS [OCo83], become in success and more and more practical. To introduce AI-based technics to IE field, at first, Lisp system should be necessary.

Lisp for PC

PC(Personal computer) population becomes larger. Table 1 shows top five PCs among Japanese industry users of Jan 1985. It is important that almost all the japanese PCs have 8086 as CPU, but are not compatible to IBM PC. There are each native OS, kanji CP/M-86, kanji MSDOS, and so on. PC plays a role of workstation rather than terminal. Then it is necessary to implement Lisp for PC.

In 1974, when 8080 was just presented and there was no OS, the author and his group implemented Lisp on ALPS/I, which is 8080 based. In 1976, succeeded to implement all the functions of Reduce-II, a formula manipulation language, on ALPS/I Lisp [Ida79a][Ida79b]. Formula manipulator is necessary for Engineering workstation. System described in this paper is the successor of ALPS/I and is designed to be with stock OS.

Table 1 Top five personal computers for industry users in japan

name	manufacturer	percent
1. N5200	NEC	14.7%
2. PC9800	NEC	14.2%
3. 9450-II	Fujitsu	12.3%
4. 5550	IBM	10.9%
5. Multi16	Mitsubisi	6.0%

from Nikkei personal Computer journal p90, 1985.1.28 issue

Common Lisp as a standard of Lisp

In spite of "many parts of common lisp cannot be implemented very efficiently on stock hardware." as [Bro84] says, Common lisp[Guy84] becomes a defacto standard of lisp, along with the needs for standardization. This tendency is also welcomed in japan.

But common lisp has several weak points for my research area. One is lack of specifications for real time or instruments handling. Another is lack of specifications for communication. The other is lack of subsetting needed for small PCs. At least, in Japan, it is too expensive now to implement all the features of common Lisp on japanese PC. This paper tries to introduces industrial or factory common lisp for PC. This language system is called APCL (Aoyama Personal Common Lisp), which is a subset common lisp with some extensions considering IE.

Design

"exec" feature

It is irrational to implement lisp as one large complete software for PC workstation rather than expensive Lisp

machine.

"exec" feature is the same concept of "exec system call" in language C. "exec" enables program to internally invoke command interpreter with command string and execute any commands at any time. Implementing "exec" in Lisp bring the Lisp system the same effects as implementing large number of software in Lisp.

CCPM has "Interpret and Execute Command Line" system call. With this call, exec is easily implemented [Ida85]. Fig.1 shows the case for explicit call via "cpm" function of APCL.

On the other hand, CP/M-86 has no built-in "exec" facility. Then the codes to emulate it was developed by the author. Appendix shows the codes for it.

CCPM has multi window. Then user can easily interrupt the current job and execute different command in another window preserving the status of interrupted job. In CP/M-86, doing another commands while executing lisp, means aborting the current lisp data. Structures stored in memory is immediately lost. Exec save the situation. This is another advantage of exec in CP/M-86, comparing CCPM.

To explicitly use "exec" is open for users. Several lisp functions use "exec" internally.

Environment sharing -----LLCP + RPS -----

LLCP and RPS are new idea for Lisp communication with environment sharing concepts overcomming ordinary communication achieved by message passing, file transfer or login to remote TSS. Basic concepts are inspired by [Ida84d][Yag84].

Key point is, as for Lisp, communication of objects is necessary. Especially, dynamic property and structure should be trasferable.

1)LLCP (Lisp to Lisp Communication Program)
LLCP is a tool and a concept for transferring Lisp objects preserving their properties with textual representation suitable for serial transfer. Commonlisp definition of printed representation include representations for any type of object, such as, array, structure and so on with all informations. This specification brings LLCP the capability to transfer not only the simple value or symbolic name of object, but also the nature or environment relating the object. Fig.2 shows the concept of LLCP.

2) RPS (Remote Process Sharing)
RPS means communication as an environment sharing. RPS is not means of specific data transfer. In RPS, two or more terminal simultaneously interact with the same working Lisp system. Then programs and data are transparent to each other, if access control is not active. To implement RPS, terminal interface part is shared by two or more physical device drivers. Fig.3 shows the RPS concept. In CCPM multi window base system, one can monitor with one window and other terminals connecting to a lisp window can interact with the same Lisp. As for implementing RPS, access control is needed and it is achieved by using commonlisp's package feature, which enables to distinguish the domain of object among several category.

Process Control Extension

To utilize APCL in IE field, process control extension is designed. Following three facilities are included.

(i) Generalized I/O for port level I/O

(ii) Interrupt handling. Interrupt procedure may be coded with "defint" function.

(iii) Timer driven procedure. Timer driven procedure may be coded with "deftimer" function. Timer procedure is invoked via (timer n) function call. n is a fixnum representation of a interval time, whose unit is 100 milliseconds.

These facilities are based and introduced along with compiler facility.

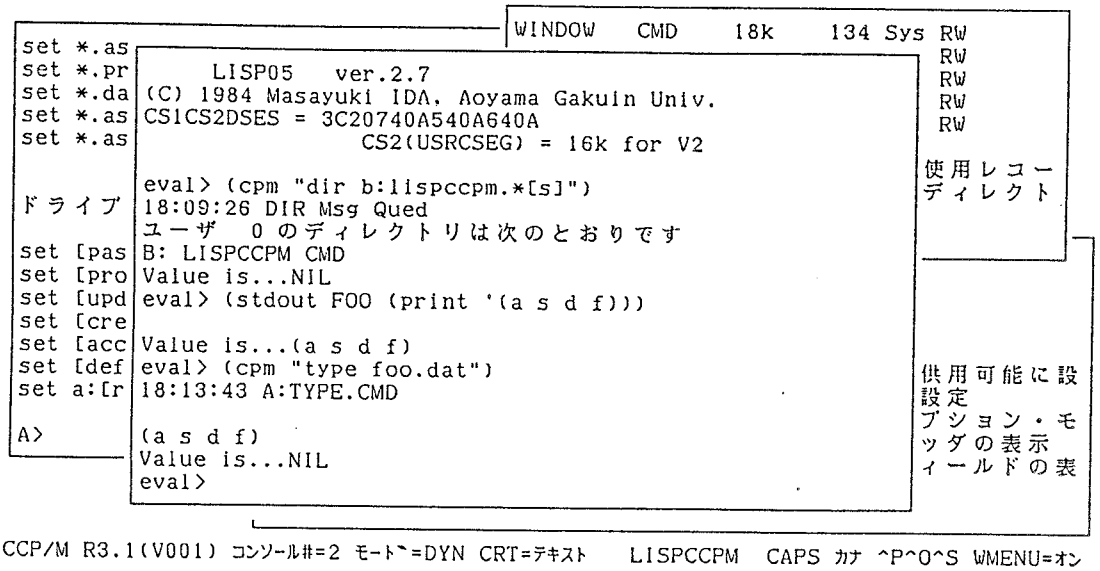

Fig.1 Lisp05, a version of APCL, on CCP/M

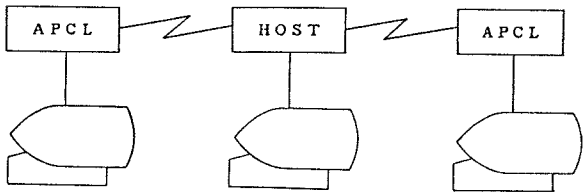

Fig.2 The LLCP Concept

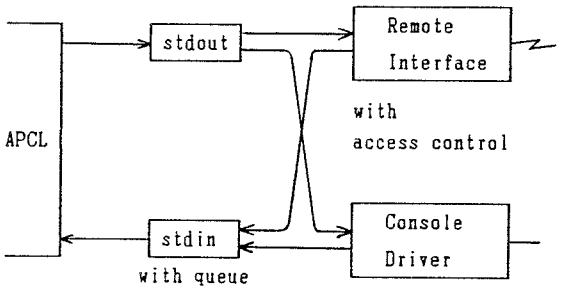

Fig. 3 RPS: Share the same environment

Functions and Facilities

Current implementation includes about 150 functions.

LLCP and RPS modules

Modules for LLCP and RPS are constructed with two level, physical and logical.

1) auxio and llxio: physical level communication handler package. There are several sub entrys. Any program can use this module and communicate to remote system. Auxio module is machine-dependent. Auxio for CP/M-86 of N5200/05 (almost APC-II in USA) consist of buffering with interrupt control, break sending, XON/XOFF control, SI/SO control (kana and kanji processing), application front-end. Llxio is the lisp functions for each auxio entry.

2) RPS: current implementation of RPS is for CCP/M. Package gating and driver are attached to auxio for CCP/M.

3) LLCP:

It has following functions

(i) llcp; object transfer

(ii) llx; execution. evaluate a form in a remote Lisp

(iii) cl; call lisp. operate as terminal for remote Lisp

(iv) lllog; display loging information

(v) llclean; initialization of spool directory

(vi) llstat; status inquiry and job control

(vii) llsub; network monitoring

(viii) llname; display system name

Graphics Module

Low-level graphics modules are implemented for mPD7220 graphic controller on N5200 and PC 9800. Functions included are line type selection, pen change, draw, move, draw-circle, and so on. They are numbered and called via "gbios" entry. General form is

(gbios function-no parameters)

Arbitrary graphic kernel can be implemented with this module. Drawing samples are in [Ida84d].

Interrupt and General I/O modules

Port-level I/O interfaces are as follows:

INPUT.

(gio_inb port_no) : byte port input

(gio_inw port_no) : word port input

OUTPUT.

(gio_outb port_no value) : byte port output

(gio_outw port_no value) : word port output

They directly execute specified I/O. They correspond to IN instruction and OUT instruction for each. With these port I/O, various kinds of I/O device may be controlled in Lisp.

To allow interrupt procedure, "defint" function is introduced.

```
(defint module_name
        interrupt-number
        body            )
```

Defint automatically invokes compiler and set compiled-object into appropriate area and set interrupt vector to indicate it.

Timer driven module has two functions; "deftimer" and "timer" as stated in earlier section.

OS exec module

Command execution facility of native OS is provided as stated earlier. For APCL on attached processor, it calls on-board ROM monitor. This invocation is a function call. Then not only top level, but also arbitrary points can invoke "exec". Examples is shown in Fig.1.

Screen editor

Screen editor with "show match" feature is implemented. When show_match is on, typing closing parenthesis automatically indicate corresponding parenthesis.

Compiler

Three pass compiler is implemented [Ida84c]. This compiler makes programs about 6 or 7 times faster than interpretive codes.

Debugging facility.

1) Standard DDT interface is equipped for APCL on CP/M-86. The form is;

(monitor)

On calling monitor, it execute "int 3" instruction, which is a break point execution. It will enter DDT command mode, when APCL is running under DDT86.

2) Trace/untrace for functions

3) Stepper for stepwise evaluation.

Redirection of standard input/output

Stdin and stdout functions are basic facility for stream management. A form,

(stdin file_name)

redirects the input stream from console device to the specified file. After EOF, input stream is reset to standard console. A form,

(stdout file_name forms)

redirects the output stream from console to the specified file on evaluating forms which is written as third parameter of stdout form. Fig.1 also shows the example of stdout.

Memory Model

Fig.4 shows the memory model. It has resident space, non-resident space for compiled codes, and non-resident space for independent modules called via "exec" facility. Screen editor, independent "cu" commands and any other software are loaded in this area. If under CP/M-86, OS size is 32kb. If under CCP/M, OS size is about 120kb. If attached engine, 8kb ROM of monitor is necessary.

Two Environments and Implementations of APCL

----- Lisp05 and ALPS/III-lisp -----

Machine environments are assumed to be in a personal computer network. As for cpu chips, I use 8080 to 80286 line since 1974 till now. 8086 line architecture is not always comfortable for implementors of Lisp, because of 64k segmentation. But it has

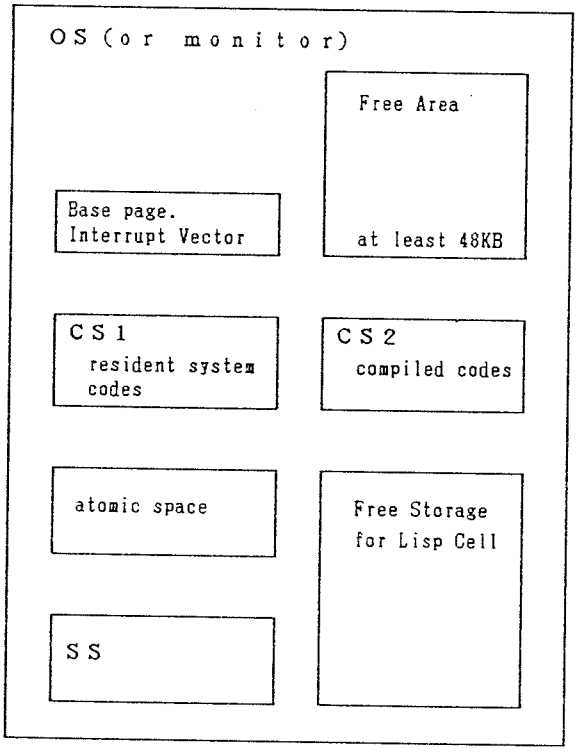

Fig. 4 Memory Allocations

large amount of softwares. Adaptive design for these line is necessary and done.

APCL is written in relocatable assembler, RASM86, for runtime efficiency sake. Base for development is DRC, lib86(library manager of DRI), link86(linker of DRI) on PC and connected UNIX minicomputer. Processors for APCL are sited to two categories.

1) Personal computer as a work station: APCL for personal computer is named Lisp05.

2) Attached processor for a TSS host: Generally speaking,Lisp is cpu bound job. Lisp based application needs a large memory and cpu load is so high. Then even in multi tasking OS of high speed computer, effective concurrency rate is not so high. It is not always wise to use Lisp directly under multi programming OS. Attaching Lisp engine to popular OS is a solution. Low cost attached processor with stock chips is designed. APCL for a specialized engine named ALPS/III [Ida84e], is called ALPS/III-lisp. ALPS/III is an 80286 based processor and is under development.

Lisp05

1) pointer:16 bit. But it is important that Lisp05 can identify 64k different objects with special multi-segmented design [Ida84a].

2) free storage: 32k cell.free storage is splitted into two 64kb banks. One is CAR bank. The other is CDR bank. DS and ES segment register of 8086 are fixed to point the base for each bank. To get car and to get cdr, only following code do.

```
car:  shl   bx,1     ; get addr.
      mov   ax,[bx]  ; get car
      ret
cdr:  shl   bx,1
      mov   ax,es:[bx]      ; get cdr
      ret
```

With this codes and pointer representation, a efficient and fast access of lisp object is attained.

3) memory assignment: There are five 64k segments: One for 32k resident space and 32k atomic objects, Two for CAR and CDR, One for non-resident space for compiled object (at least 16k), and stack segment (minimum 8k). Furthermore, at least 48k for "exec" area is necessary.

4) Machine: CP/M-86 or CCPM based machine with 384k bytes or more. At first it is designed for N5200/05, but is portable to many PCs. Communication line: One RS232C or more. Printer: Kanji & bit-based printing device. Auxiliary storage:two 8" floppy drive or more. Display: Kanji, color, and bit-based graphics capability.

ALPS/III lisp

1) pointer: 32 bits, which is compatible to dominated Lisps.

2) free storage: 1Mb or more. cdr coding representation.

```
CAR:  mov     es,cx
      mov     cx,es:2[bx]
```

```
        mov     bx,es:[bx]
        ret
CDR:    add     bx,4
        adc     cx,0
        ret
```

3) ALPS/III hardware.
CPU: 80286(80386 preferred). memory:2MB or more, 64KB ROM for resident. communication: high speed serial line (9600bps or faster). host to be attached: UNIX(or CCPM based) machine.

Conclusion and Discussions

This paper introduces some functions for factory needs trying to approach common lisp for industrial engineering. 8086/286 based PC can have lisp with many features.

Table 2 shows the measurement of the current version of Lisp05.

For discussion, Subsetting of common lisp may be necessary for PCs. Data sharing and parallel construction among interrupt procedure and interrupted one are future problems.

After prepared this paper, on 16th Feb.1985, I received the source codes of SPICE Lisp, developed by prof. Farhlman of CMU (I didn't look at its contents yet). SPICE lisp seems to conform commonlisp. Then, for future extension of APCL, I would like to refer it.

Acknowledgement

The author would like to express his gratitude to many persons and students who encourage and assist him. Especially thanks are due to Dr.Ohata, Dr.Mano, Mr.Misaki, Mr.Yagihasi, Mr.Yamada, and Mr.shigemitu.

References

1.[Bro84] R.A.Brooks and R.P.Gabriel, A Critique of Common Lisp, pp1-8 proc.of LISP conference, 1984
2.[Guy84] Guy L.Steele Jr. et.al., Common Lisp: The Language, Digital Press,1984
3.[Ida79a] M.Ida and K.Mano, ALPS/I: A Microcomputer based Lisp machine, J.IPSJ,1979
4.[Ida79b] M.Ida, An Adaptable Lisp Machine based on Microprocessors, proc.IMMCC IEEECS, pp210-215, Nov.1979
5.[Ida84a] -,Some features of an experimental Lisp for 8086, WGSYM 27-2 IPSJ,march 1984
6.[Ida84b] -,Some Considerations on the features of UNIX as a defacto standard OS, pp209-217, proc.IMAC'84
7.[Ida84c] -,A compiler for Lisp05, 29th annual conf. IPSJ, Sept.1984, pp365-366
8.[Ida84d] -,"5. Lisp",Nikkei science journal 73, pp46-55,Oct. 1984
9.[Ida84e] -,Scheme for designing ALPS/III,a portable back-end type Lisp processor, WGSYM 30-2, Dec.1984
10.[Ida85] -,"exec features on personal computer OS", Nikkei byte journal, to be appeared on May issue, 1985
11.[OCo83] D.E.O'Connor,"Using Expert Systems to Manage Change and Complexity in Manufacturing" in Artificial Intelligence Applications For Business, edited by W.Reitman, Ablex pub. corp. 1983 12.[Yag84] S.Yagihasi,Interface between a CP/M-86 and a host OS, 29th annual conf. IPSJ, sept.1984, pp349-350

Table 2 benchmark of Lisp05 ver.2.8

benchmark	interpreted code	compiled
tarai-4	12,800 msec.	1,900
tarai-5	309,000	51,000
tak-18-12-6	66,500	
tpu-1	8,300	
tpu-2	39,000	

Appendix. Codes for Exec Facility

```
; ----- exec facility for CP/M-86 -----
;
;       ONLY for CP/M-86 V1.1a
;       adopted from Lisp05
;       (c) 1985 Masayuki Ida
        public  xcpm
xcpm:   ; pointer in SI, length in CL
        push    ds
        mov     ax,cs
        mov     ds,ax   ;  set to 8080 model
        mov     di,offset  xcpm2
xcpm3:
        mov     al,0[si]
        cmp     al,61h  ; lower case letter
        jc      xcpm5   ;  to upper case
        xor     al,20h
xcpm5:  mov     0[di],al
        inc     si      ! inc di ! dec  cl
        jnz     xcpm3
        mov     byte ptr [di],0 ; set
                                ; tariling 0
        mov     cl,51
        mov     dx,cs   ; set DMA segment
        int     224
        mov     cl,26
        mov     dx,offset xcpm2
        int     224

        push    es
        mov     ax,0040h ; set base
        mov     es,ax    ; for CPM kernel
        mov     di,21f5h ; patch start here
        mov     es:word ptr [di],06c6h
        mov     es:word ptr 2[di],2498h
        mov     es:word ptr 4[di],0e900h
        mov     es:word ptr 6[di],0ed54h
        mov     di,0f4ch ; returning patch
        mov     es:byte ptr [di],0eah
        mov     es:1[di],offset xcpm4
        mov     es:3[di],cs
        mov     xcpm6,sp        ; save sp
        mov     cl,2fh
        int     224     ; program link
; executed and returned
;       reset patch
xcpm4:
        mov     ax,0040h
        mov     es,ax
        mov     di,0f4ch
        mov     es:word ptr [di],06c6h
        mov     es:word ptr 2[di],2498h
        mov     es:byte ptr 4[di],00
        mov     sp,xcpm6
        mov     ax,cs
        mov     ss,ax
        pop     es      !       pop     ds
        ret

xcpm2   rs      40
xcpm6   dw      0
```