記 号 処 理 51－3
（1989. 6. 2）

AI電卓における記号処理の可能性について

田添英一　前田敦司　中西正和
慶応義塾大学理工学部数理科学科

近年現れたLisp処理系を搭載したポケットコンピュータは初学者のためのLisp言語の学習方法や記号処理システムの普及に大きな影響を与える可能性がある。この観点からポケットコンピュータ上のLisp処理系について、処理系の性能・仕様の妥当性を検証し、その現実的な用途について考察した。その結果、現在のポケットコンピュータ上のLisp処理系では、性能上の制約により入門用以外の用途に使用することは困難であることが結論づけられた。

Symbolic Computation on Pocket Computer

TAZOE Eiichi, MAEDA Atusi and NAKANISHI Masakazu
Department of Mathematics
Faculty of Science and Technology
Keio University
3-14-1, Hiyosi, Kouhoku, Yokohama 223, Japan

Recently pocket computer equipped with Lisp language processor has appeared. This pocket computer can have an impact on the way of Lisp education for beginners and can spread the availability of symbolic computation system. From these points of view, we measured the performance of the Lisp processor and reviewed its language specification. Practical application of the pocket computer is argued.

We conclude that the limitation of current Lisp interpreter performance on the pocket computer effectively restricts its practial usage only to educational purposes.

1 はじめに

Lisp言語を用いて実用的なプログラミングを行なうためのハードウエア環境は、かつては大型計算機やLisp専用機に限られていた。

近年、ミニコンピュータやワークステーションの性能の向上とともに、それらの上の高性能なLisp処理系が広く普及した。またパーソナルコンピュータにおいても、ハードウエア・ソフトウエアの両面でLispプログラミング環境は急速にその実用性を増しつつある。

最近、Lisp処理系を標準装備した電卓サイズのコンピュータ（ポケットコンピュータ）が発表された。ハードウエアの性能は極めて貧弱なものであるが、初期のLisp処理系においてもわずか数Kセルの容量で定理証明や数式処理などの記号処理プログラムが動作していたことを考えれば[4]、「記号処理のできるポケットコンピュータ」というアイディアは非常に魅力的なものと思える。[1]

本報告では、このポケットコンピュータ（AI-1000）の概要を紹介し、実際に使用した経験をもとに、AI-1000が上にあげたような期待に応え得るシステムかどうかについて考察する。

2 ポケットコンピュータの用途

ポケットコンピュータの特長として、価格が安いこと、小型で携帯できることが挙げられる。これらの特長を活かす用途としては次のものが考えられる。

1. プログラミングの入門用に用いる。
2. より高価な計算機の代用として一般の情報処理に用いる。
3. 高機能の電卓として即興的な計算に用いる。
4. 電話回線などを通して端末機として用いる。
5. 他のコンピュータが使えないような場所でもプログラミング作業ができる入力装置として用いる。

BASIC処理系を装備したポケットコンピュータはすでに約10年前から製品化されている。パーソナルコンピュータよりもかなり安価でありながら汎用のプログラミング言語であるBASICが使用できたため、当初はより大型のコンピュータの代用として小規模な数値計算などの用途にも用いられた。しかしながら、次第に情報処理機器としての利用は少なくなり、入門用か、あるいはプログラミングのできる電卓としての即興的な計算以外には用いられなくなってきている。その理由としては以下のものが挙げられる。

1. ソフトウエアの市場が発展せず、ユーザがプログラムを書く必要があった。
2. パーソナルコンピュータのローエンド機種が低価格化し、価格の利点が低下した。
3. コンピュータの台数が増え、携帯性の利点が低下した。（移動先にもコンピュータがあることが多くなった。）

Lisp処理系を備えたポケットコンピュータについても、パーソナルコンピュータの発展によって価格と携帯性の利点が相対的に低下する点は同様である。

以上の点を踏まえて、AI-1000がどのような用途に使用可能か、性能の面から検討する。

3 AI-1000の性能

3.1 ソフトウエアおよびハードウエアの仕様

表1にAI-1000Lispの標準シンボルの一覧を示す。データ型も少なく機能を省略した関数も多いが、約200個もの関数を備えている。関数名や変数名はCommon Lisp[1]に準じている。言語仕様だけから考えるとAI-1000はかなり本格的なプログラミングの可能なシステムに見える。

しかしながら、AI-1000のRAMの大きさは標準で32KB、最大64KBと極めて限られたものであ

[1] REDUCEやMathematicaのような強力な数式処理システムが走るポケットコンピュータが普及すれば、理系の学生の学習スタイルは大きく変わるのではないだろうか。

表 1: AI1000-Lisp の標準シンボル

&key	atom	format	minusp	set
&optional	beep	fourth	mod	set-cursor
&rest	boundp	funcall	nconc	setf
*	break	function	nil	setq
init	butlast	functionp	ninth	seventh
print-base	car~cddddr	gc	not	signum
print-case	ceiling	gensym	nreverse	sin
print-escape	char-code	get	nth	sinh
print-radix	clear-screen	go	nthcdr	sixth
read-base	close	if	null	special-form-p
standard-input	code-char	implode	numberp	sqrt
standard-output	cond	intern	oddp	step
terminal-io	cons	lambda	open	streamp
+	consp	last	or	string/=
−	cos	length	pairlis	string<
/	cosh	let	pi	string<=
/=	defmacro	let*	plusp	string=
1+	defun	list	prin1	string>
1−	delete	listp	princ	string>=
<	do	load	print	stringp
<=	do*	log	prog	subst
=	draw	logand	prog1	symbol-name
>	drawc	logior	prog2	symbol-value
>=	ed	lognot	progn	symbolp
abs	eighth	logtest	psetq	t
acons	endp	logxor	quote	tan
acos	eq	loop	rassoc	tanh
acosh	eql	macrop	read	tenth
and	equal	make-array	read-char	terpri
angle	eval	makunbound	rem	third
append	evenp	mapc	remove	trace
apply	exp	mapcan	remprop	truncate
aref	explode	mapcar	rest	unless
ash	expt	mapcon	return	untrace
asin	fboundp	mapl	reverse	vectorp
asinh	fifth	maplist	round	write-char
assoc	first	max	rplaca	zerop
atan	floor	member	rplacd	
atanh	fmakunbound	min	second	

表2: データオブジェクトのサイズ（[2]より転載）

データ型	サイズ（バイト）
cons セル	5
シンボル	9
文字列	長さ+1
数値	8

表4: ベンチマークテストの結果

テスト	実行時間
tak	1時間22分35.0秒
tarai-2	3.7秒
tarai-3	51.7秒
tarai-4	16分22.0秒
6-queen	3分58.7秒

る。外部記憶装置[2]を用いない場合にはソースプログラムテキストもRAM上に保持することになるため、Lispデータ領域はさらに制限を受ける。（表2にLispデータオブジェクトのサイズを、表3にデータ領域の初期設定値を示す。）

3.2 容量および速度について

AI-1000Lispインタプリタの実行速度を計測するため、いくつかのベンチマークテストを行なった。ベンチマークの結果を表4に示す。ベンチマークテストはすべてRAM32KB、領域の割り当ては表3の設定で実行した。計時はストップウォッチによって行なった。[3]

ベンチマークテストの例題として8-queenでなく6-queenを用いたのは、メモリ領域の不足で8-queenが実行できなかったためである。その際、メモリを節約する実験の意味もかねて図1に示す関数mloadを用いてプログラムの占有するセルを節約しようと試みた。しかしながら、組み込みのload関数に比べてはるかに速度が遅くなる上、（図1のファイルをload関数で読み込んだ場合3.8秒。mloadで読み込んだ場合5分55秒）小さなプログラムではそれほど節約の効果がなかった[4]ため、以後はこの関数を使用しなかった。

```
(setq monocopy nil)
(defun monocopy (x)
  (if (atom x)
      x
      (prog ((car (car x))
             (cdr (cdr x)))
        (setq x nil
              car (monocopy car)
              cdr (monocopy cdr))
        (mapc
         #'(lambda (cons)
             (and (eq (car cons) car)
                  (eq (cdr cons) cdr)
                  (return cons)))
         monocopy)
        (setq monocopy
              (cons (cons car cdr)
                    monocopy))
        (return (car monocopy)))))
(defun mload (name)
  (let ((file (open name
                    :direction :input)
        (eof '#:eof))
    (setq monocopy nil)
    (do ((form (read file nil eof)
               (read file nil eof)))
        ((eq form eof)
         (setq monocopy nil)
         t)
      (print (eval (monocopy form))))))
```

図1: セル領域節約のための単写プログラム

[2]オプションでカセットテープ、3.5インチFDDが接続可能。またRS-232C（オプション）からもプログラムをロードできる。

[3]AI-1000Lispには計時機能がない。

[4]n-queenのプログラムの場合、プログラムの占有するセル93個を71個に節約できる。

表 3: データ領域の初期設定値 ([2]より転載)

	RAM32K バイト	RAM64K バイト
ユーザエリア(固定)	22527	55295
work	1024	1024
file	4095	8191
lisp	17408	46080
cell	10880	28800
symbol	2176	5760
data	4352	11520

4 評価

4.1 マニュアルについて

操作マニュアル (約110ページ) [3]とLisp言語入門書 (パート1 Lisp 言語入門 約130ページ、パート2 リファレンスマニュアル約230ページ) [2]の2冊が附属している。Lisp言語入門書からは、メーカーがLisp言語の入門をAI-1000の主な用途として想定していることが汲み取れる。記述も正確さよりは分かり易さを主眼としているようである。しかしながら、パート1の部分はエディタやインタプリタの操作説明も兼ねているため独習書として十分な内容とは言い難い。バッククォートやラムダリストキーワードなど、単なる機能の紹介に終っているものもある。

4.2 言語仕様、性能について

関数の数など機能の面ではパーソナルコンピュータ上のLisp処理系と比べても見劣りしないが、速度面では1けた以上の差がある。用途を入門用に限ったとしても、例えばオプショナル引数のsupplied-p変数やキーワード引数などは、意義を理解させるだけの十分な解説も入門書になく、いたずらに処理系の速度を低下させているに過ぎないように見受けられる。

実用的な用途に使うには、block/return-fromやcatch/throwなどの制御構造の欠如、特にunwind-protectの欠如は大きな制約となろう。

また、システムプログラミングに関しては、symbol-functionやmacro-function (関数、setf-methodとも)、macroexpandの欠如が致命的である。一方でspecial-form-pやmacrop[5]といった関数が用意されているが、symbol-function等が無い以上その有用性は疑問である。

AI-1000と同程度の容量の小型コンピュータ上で記号処理システムの稼働を可能にしていた過去のLisp処理系と比較して、AI-1000Lispのインタプリタは特にメモリ効率の面で大きく劣っている。言語仕様に違いがあるので、当時使われていた効率向上のための技法[5][6]がそのまま適用できるかどうかは分からないが、変数のスコープがレキシカルスコープのみでtail recursionの効率化が行ないやすいという利点もあることから、効率向上の余地は十分にあるものと思われる。

4.3 AI-1000の用途

速度と容量を考慮すると残念ながらAI-1000はLispプログラムの実行環境として実用的というにはほど遠い。従ってAI-1000の用途としては主としてLisp言語の入門用および電卓としての用途に限られるであろう。その他の用途、例えばプログラムの入力機器としての能力は、キーボードやエディタの制約からメモ用紙と大差ない。

Lispに対して知識があり、Lispでプログラムを書くのが好きなユーザにとっては、BASICでなくLispでプログラムを書けることだけで他のポケットコンピュータと比べて利点があるといえるかも知れない。しかし、ポケットコンピュータ自体を必要とする用途が特にあるならばともかく、単に

[5] Common Lisp の macro-function の代用の述語

娯楽としてのプログラミングの用途にはAI-1000は適さないと思われる。速度・容量・エディタの制約に加えて、先に述べた通り言語仕様にも不満があるからである。

5 結論

現時点ではAI-1000の用途は事実上、入門用のみに限られる。これを念頭に置くならばAI-1000Lispの仕様のうちいくつかは不必要であり、効率の向上のためにも取り除くべきものと考えられる。また入門書のさらなる充実が強く望まれる。

入門用のみに用いるにせよ、あるいはそれ以外の用途に用いるにせよ、処理系の速度・メモリ効率の向上は不可欠である。特に現在は実行時間のかなりの割合がガベージコレクションのために消費されており、処理系のメモリ効率の向上はガベージコレクションの回数の削減により速度の向上にも寄与すると思われる。

References

[1] Steele, Guy L. Jr., et al., 「Common Lisp: the Language」, Digital Press (1984)

[2] 「AI-1000 Lisp言語入門書」, カシオ計算機株式会社 (1988)

[3] 「AI-1000 操作マニュアル」, カシオ計算機株式会社 (1988)

[4] 後藤英一他監修, 「記号処理シンポジウム報告集」, 情報処理学会プログラミング・シンポジウム委員会 (1974)

[5] 菱沼千明,山下堅治,中西正和, 「LISPインタープリタにおけるスタック技法とaリストの抑制法」, 情報処理 Vol. 17, No. 11 (1976), pp.1002–1008

[6] 酒井俊夫, 「ミニコン用LISPに関する考察及び提案」, 修士論文、慶應義塾大学工学研究科 (1976)