バッカスのFP方式に関する研究（続報）
—アセンブリ言語によるバッカスFPの処理プログラム—

金高光平*・渡辺哲**・原田智***
堀井健一****・竹島直樹*****・伊藤惇

Study on Buckus' FP System—continued
(Assembly Program for Backus' FP System)

Kohei KANETAKA, Satoshi WATANABE, Satoshi HARATA,
Ken'ichi HORII, Naoki TAKESHIMA, Jun ITO

(昭和62年10月31日受理)

In principle von Neumann computer has three parts: a central processing unut (CPU), a store, and a connecting tube that transmit "a word at a time" between the CPU and the store. J. Backus, IBM Research Laboratory, San Jose, proposed to call this tube the von Neumann bottleneck, because programing by von Neumann computer is basically planning and detailing the "enormous" traffic of words through this tube and almost of that trafic is not significant data but only concerns where to find it. In his paper, he suggested that basic defects of von Neumann languages make their expressive weakness and their "cancerous" growth inevitable and further he proposed new functional programing system (Backus' FP) alternating with our old conventional languages, von Neumann ones. Recently many studies of the "non"-von Neumann computer and the languages have so actively been made for the reason just described. Under such circumstances, in this report, an assembly language program processing the Backus' FP system is presented.

1. はじめに

関数の組合せによってプログラムを作成するFP方式は，質，生産性のみならず，数学的であることが我々の考え方と対応するので考えやすいなど，優れた点が多く注目を集めている。

FORTRANの生みの親であるJ. Backusは従来のプログラミング言語に致命的欠陥があると指摘している[1]。イノマン型計算機の「1時に1語」的な考えを引きずっている従来の言語は，我々を1語ごとにものを考えさせ，もっと大きな概念単位で考えることのさまたげになっていると言うのである。

このようにして出現したFP[1,2]についての期待は大きく[3]，またすでに井田らによる事例研究[4]，解説[5]，C言語による処理系[6]など[7,8,9]が発表されている。本研究は，Backusの提案したFPを用いてプログラムを作成し，基本関数と関数形成について検討を加え，さらにアセンブリ言語を用いてパソコン上で動く処理系を作成したので公表する。

* アルプス電気　** 日本電気ソフトウェア
*** 日立デバイスエンジニアリング
**** 日立通信システム　***** TDK

2. FPについての提案

FPにおける基本関数には15種，24個あるが，そのうちatom, null, apndl, apndr, sr, tlr, rotl, rotrは次のように他の基本関数で定義することができる。

atom ≡ or ∘ [eq ∘ [length, $\bar{1}$], null]
null ≡ eq ∘ [id, 0]
apndl ≡ merge　　apndr ≡ merge
sr ≡ s ∘ reverse
tlr ≡ reverse ∘ reverse
rotl ≡ merge ∘ [tl, 1]
rotr ≡ merge ∘ [lr, tlr]

ここでmergeはapndlとapndrを単にまとめたものであるが，以下に示す基本関数int, lessを追加することを提案する。

int : x is a numerical atom → integer part of x ;⊥
less : x ≡ x = ⟨y, z⟩ & y<z → T
; x = ⟨y, z⟩ & y ≧ z → F ; ⊥
merge : ⟨⟨y_1, y_2,……, y_m⟩, ⟨z_1, z_2,……, z_n⟩⟩
→⟨y_1, y_2,……, y_m, z_1, z_2,……, z_n⟩
; ⊥ m≧1, n≧1
Example
int :⟨5.93⟩＝5
less :⟨5, 8⟩＝ T
less :⟨16, 9⟩＝ F
merge :⟨ϕ, 4⟩＝⟨4⟩
merge :⟨⟨4, 1⟩, ⟨8, 5⟩⟩＝⟨4, 1, 8, 5⟩
これらの新たに提案した3つの関数を用いることによって，より広い範囲のプログラムを作成することが可能になるが，特にここではデータ処理プログラムについて示す。
先頭の要素より大きい要素からなる列を求める。
Def RGT ≡ RGT1 ○ [1, id]
Def RGT1 ≡ atom → id ; RGT2
Def RGT2 ≡ eq ○ [length ○ merge, $\bar{1}$]→ 0 ; RGT3
Def RGT3 ≡ gt ○ [1 ○ 2, 1]→ KEEP ; CUT
Def KEEP ≡ merge ○ [1 ○ 2, RGT1 ○ [1, tl ○ 2]]
Def CUT ≡ RGT1 ○ [1, t ○ 2]
先頭の要素と同じ大きさの要素からなる列を求める。
Def REQ ≡ REQ1 ○ [1, id]
Def REQ1 ≡ atom → id ; REQ2
Def REQ2 ≡ eq ○ [length ○ merge, $\bar{1}$]→ 0 ; REQ3
Def REQ3 ≡ eq ○ [1 ○ 2, 1]→ KEEP ; CUT
最大値を選択する。
Def MAX ≡ atom → id ; MAX ○ COMP
Def COMP ≡ less ○ [lr, 2r]→ tlr ○ rotr
大きい順に並べかえる
Def SORTK or ○ [atom, null] ○ id ; SORT'
Def SORT' ≡ (/merge) ○ [SORTK ○ RGT, REQ, SORTK ○ RLT]
これらの他に SQRT(平方根)，GCM(最大公約数)，LSM（最小公倍数)，IM（逆行列)，FIBO（フィボナッチ数列)，QUAD（2次方程式の根）などのプログラムが作成可能となる。

また，関数形成についても，8つのうち Binary to unary, While のふたつは，次のように他の基本関数や関数形成により，定義することができる。

(bu fx) ≡ f ○ [x, id]
while ≡ p → while f ; id
Example (last element of sequence)
Def LAST ≡ while (ne atom) tl
Def LAST ≡ atom → id ; LAST ○ tl

3. 処理系について

Z80アセンブリ言語で書かれた本処理系は cp/m80 上で動作するインタープリタである。図1にメモリマップを示す。

処理は，中間言語作成，オブジェクト格納，実行に分けて行われる。全体のフローチャートを図2に示し，以下にその概略を述べる。

3・1 中間言語作成　中間言語は1ワードで構成されており，実行順に並べられていて，終了の場合は1バイト目が00Hになっている。

3・1・1 基本関数　1バイト目には基本関数を表わす01H, 3, 4バイトめには関数の処理先アドレスが格納されている。2バイト目は，普通使わないがセレクター関数の場合のみ，その数が格納される（図3 (a))。

3・1・2 ユーザー関数　ユーザー関数が定義され

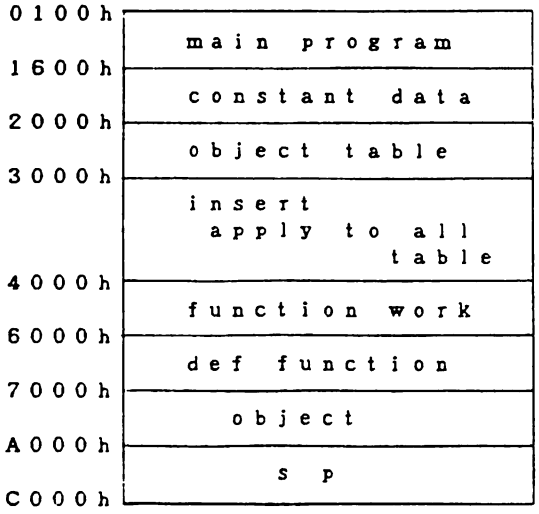

図1　メモリマップ

た場合には，その中間言語はライブラリとして6000H 番地から格納されるが，そのユーザー関数が別の関数の中で定義または実行された場合，中間言語は図3（b）のようになる。

3・1・3　関数形成

a．constant　constant が表わす数値は，中間言語の4バイト中には無く，1600H 番地からまとめて格納されている。中間言語にはこの格納アドレスが入れられている（図3（c））。

b．condition　中間言語は4つの部分に分れられる。最初が condition であることを表わすもので，1ワードである（図3（d））。2番目には条件の関数の中間言語が入り，3番目，4番目にはそれぞれ true, false の場合の関数の中間言語が格納される。

c．construction　最初の1ワードで construction であることと，construction 内の関数の数を示しているその後は，construction 内の，関数の中間言語が，後から順に格納されている（図3（e））。

construction 内の関数の区切りとして次の中間言語の1バイト目の第8ビットを1として格納している。

d．insert　insert はオブジェクトが作用して初めてその処理が決る動的なものである。よって中間言

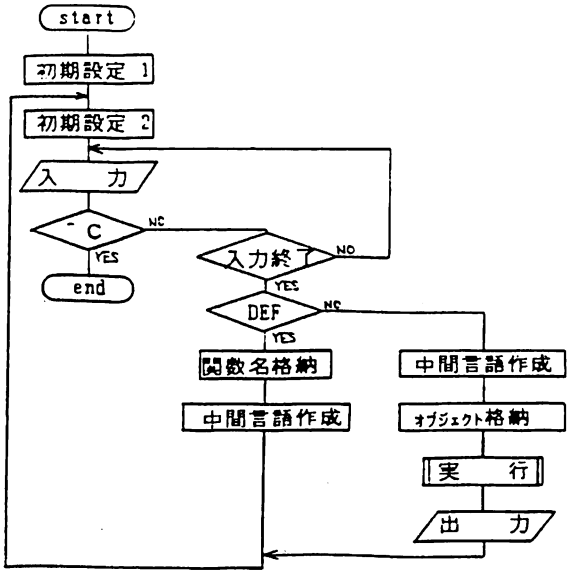

図2　フローチャート　1

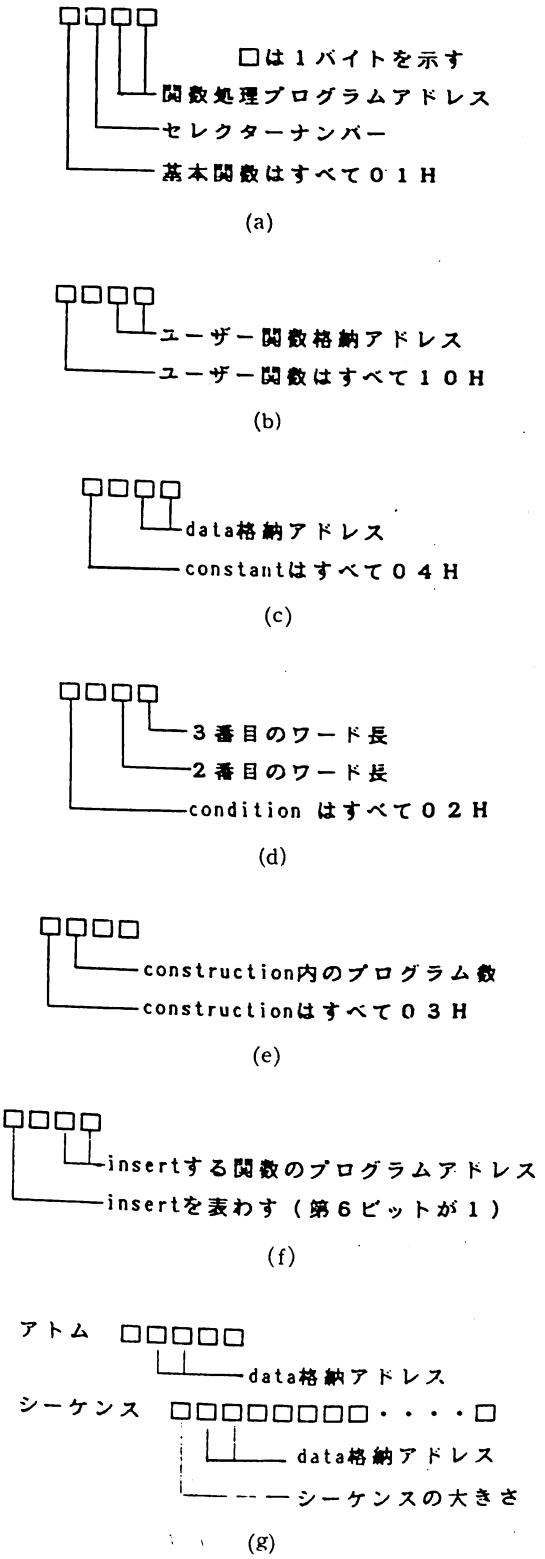

図3　格納状態

語は insert 処理される関数の中間言語の1バイト目，第6ビットが1となっている（図3（f））。

e．**apply to all**　insert と同様であるが，1バイト目は第7ビットが1となる。

3・2　オブジェクト　オブジェクトは固定少数点方式で，7000H 番地より格納されていて，2000H 番地から格納されている図3（g）のように管理されている。

3・3　実　行　中間言語の格納順に沿って実行していく。ただしユーザー定義関数の場合はプログラムカウンタを 6000H 番地からの，それぞれの中間言語格納先に置換えて実行し，実行後にもどすことになる。

3・3・1　construction　construction 内の関数の実行順に，プログラムカウンタの表と，それぞれに対応するオブジェクトを作成し，実行する。図4にフローチャートを示す。

3・3・2　insert, apply to all　作用されたオブジェクトに対応する数の中間言語を作成する。ただし，apply to all については再帰の実行が可能である。図5にフローチャートを示す。

ここで言う中間言語とは，insert, apply to all をそれぞれ次のように分解したものである。

$/f : \langle x_1, x_2, \cdots\cdots, x_n \rangle$
$\rightarrow f \circ [1, f)) \circ [2, \cdots, f \circ [n-1, n]]] :$
$\langle x_1, x_2, \cdots\cdots, x_n \rangle$
$n\rangle$
$\rightarrow f : \langle x_1, f : \langle x_2, \cdots\cdots, f : \langle x_{n-1}, x_n \rangle\rangle\rangle$
$\alpha\ f : \langle x_1, x_2, \cdots\cdots, x_n \rangle$
$\rightarrow [f \circ 1, f \circ 2, \cdots\cdots, f \circ n] : \langle x_1, x_2, \cdots\cdots, x_n \rangle$
$\rightarrow \langle f : x_1, f : x_2, \cdots\cdots, f : x_n \rangle \quad x_i = \langle x_i, x_{i+1} \rangle$

4．む　す　び

Backus によって提案された FP により約50種のプログラムを作成し，基本関数，関数形成を検討し，これらにコメントを加えると共に，cp/m80 上で動く処理プログラムをアセンブリ言語を用いて作成した。まだ多くの問題点はあるが，今後さらに検討したい。

最後に原稿作成にご協力頂いた本校技官の斎藤輝雄氏に感謝いたします。

参　考　文　献

1) Backus, J., CACM, 21-8 (1978), pp. 613-641.

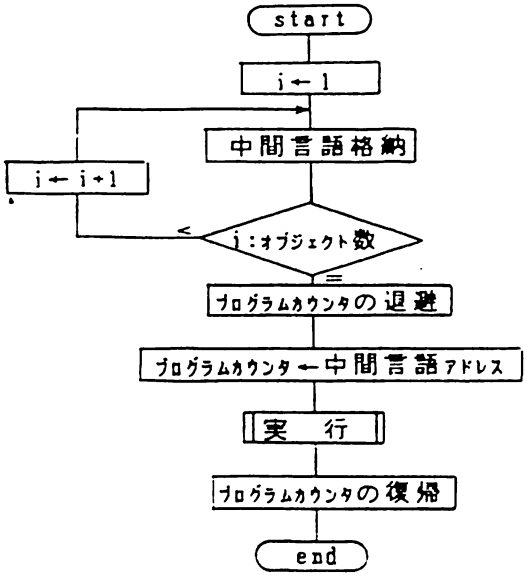

図5　フローチャート　3

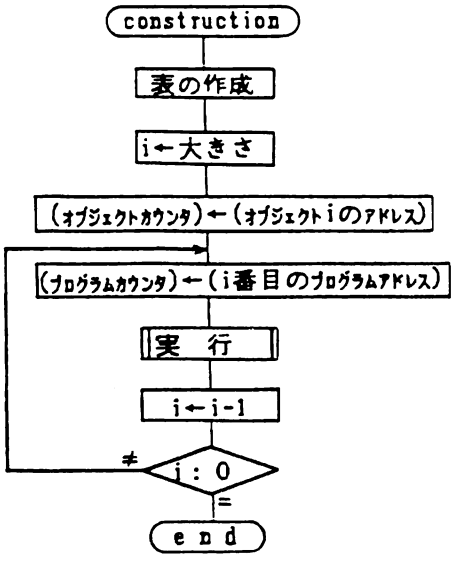

図4　フローチャート　2

2) Backus, j., Proc. Functional programming languages and computer architecture, Oct. 1980, pp. 1-10.
3) 例えば，片山他，情報処理，24-2 (1983), p. 112.
4) 井田他，日本ソフトウエア科学会第一回大会論文集，2F-2, pp. 107-116.
5) 井田，bit, 16-2 (1984), pp. 60-66. 16-3, pp. 52-57. 16-4, pp. 70-77. 16-5, pp. 112-119. 16-7, pp. 80-87. 16-8, pp. 84-90. 16-9, pp. 74-82. 16-10, pp. 99-104.
6) 伊藤（民），I/O, 1986年2月，pp. 254-270. 3月，pp. 271-278.
7) 伊藤，情報処理学会全国大会，31回(1985), pp. 9-10.
8) 竹島他，情報処理学会全国大会，32回(1986), pp. 23-24.
9) 伊藤他，秋田高専研究紀要，21（昭61年，2月），pp. 21-23.