LIME

'83 クリスマス
'84 お正月
特 度 号

Limited Expression
of
KTTU COMPUTER CLUB

No. 6

Box用

目　次

CONS 83

— After Festival —

MASA

先日行なわれた'84松ヶ崎祭では、CONS (Compact Network System) が展示され、不完全ながらもその基本動作は確認され、数台のPC-8801による実演が行なわれました。ここではそのCONSのアーキテクチャの概容をお伝えしたいと思います。

基本的な構造は、Line VOL.2に記載されているCONS'82と同一であり、今回は、その変更点を重点的に説明を行ないます。

○ネットワーク形態

CONSでは、各ノード(節)が輪のように信号線で接続されるリングネットワークの形態をとりますが、今回は新たに、通信専用のサブ・プロセッサを設け、ネットワークのほとんどの処理はサブ・プロセッサ内部で行なわれます。

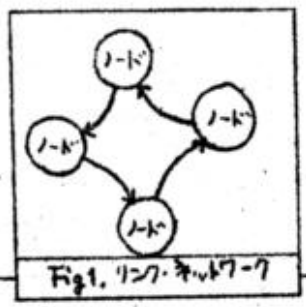


Fig1. リング・ネットワーク

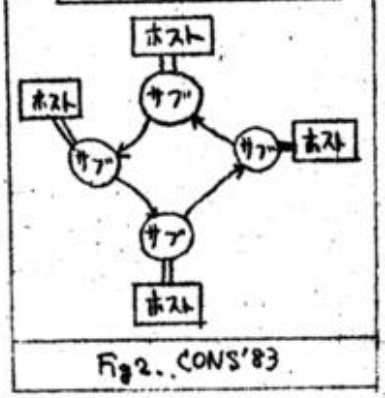


Fig2. CONS'83

○サブ・プロセッサ

仕様は次の通りです。

CPU … Z-80A
ROM … 2KB (2716)
RAM … 2KB (6116)
SIO … Z-80SIO/0

これにRS-232Cのバッファを2ch装備し、1方はネットワーク用として、他のサブプロセッサと接続され、ボーレートは9600bpsです。他方はホスト・コンピュータと接続され、ボーレートは4800bpsとなり、より多くのコンピュータと接続可能です。

さらに、電源はホスト・コンピュータとは独立して供給され、ホスト・コンピュータが停止している間もネットワークは駆動可能です。

○ プロトコル

CONS上のデータはすべて一定のフォーマットで区切られており、これをパケットと呼びます。このフォーマットは次のようになっています。

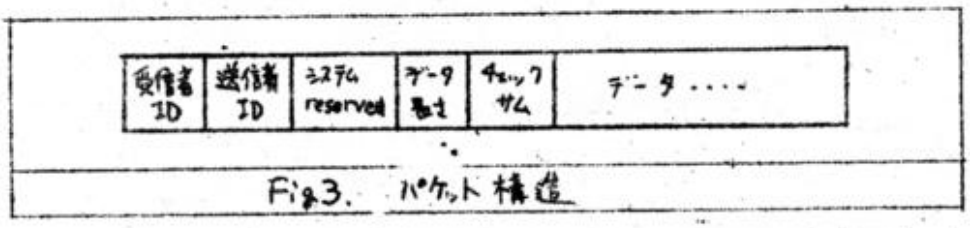


Fig3. パケット構造

このパケットは、実際にはサブ・プロセッサ間でやりとりされ、ホスト・コンピュータはIDの管理やエラーチェック等はほとんど関与しないため、ホスト・コンピュータで実行されるプログラムでの処理は簡単になります。

ホストからホストまでの通信手順をFig4に示します。

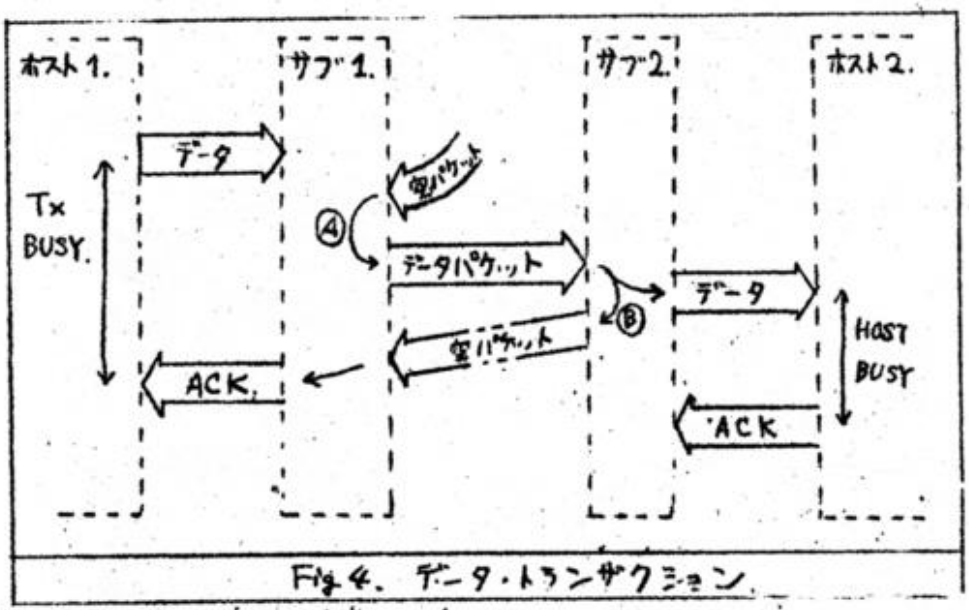


Fig4. データ・トランザクション

CONS'83ではパケットの数は1個に制限されています。このためネットワーク上ではパケットが常に回っており、パケットに何もデータが含まれない状態を空パケットといいます。Fig4のⒶでは空のパケットにデータを載せており、Ⓑでデータを取り出して再度空パケットに戻しています。

○問題点。

今回の展示では、サブ・プロセッサのハードウェアが完動したにもかかわらず高速でデータ通信を行なうと、時として誤動作を起こし、ネットワークが停止することがありました。これは、ホストコンピュータ内部の処理がすべてBASICで記述するため、処理速度が追いつかず、正常な運行ができなかったのと、それに対するエラー対策やより上位のプロトコルと下位のプロトコルとの連絡が完全でなかったことが災いしたようです。

○ CONS'84から完成へ。

現段階ではCONSは実験の域を超えることができませんが、これが実用レベルに達するためには上記の問題を克服することが必要です。そのためにはプロトコル細部の見直しや、パケットのフォーマットの強化、が必要となります。そのためにもJISの通信手順の採用が有効と思われます。

さらに通信能力を高めるために、パケットの個数を複数個許し、空パケットをなくすことも採用されるべきでしょう。

そしてCONSシステムとしてはこれ以上の改良を施すとともに、そのネットワークの実験としてのプロジェクトは終うしたいと思います。

最後に、ネットワークソフト開発班の「オヤブン」として働いてくれた並河氏に感謝の意を表すると共にCONSをベースとして、より熟成されたシステムを完成できることをひとつ期待してみたいと思います。

END.

1982年9月. ネットワークを利用したゲームの開発決意。
1982年11月. CONS Ver1.0 開発、学園祭で展示を試みるも失敗に終る。
1983年4月. CONS'83としてニューバージョンの開発決定。
1983年10月. サブ・プロセッサ基板プロトタイプ完成。
1983年11月. CONS'83システム完成、3台のPC-8801を用いて試運転、一部のアプリケーションの実演。
1983年12月. ------

P.4

半川　決

'84 役員一覧

部長	並河　一比古
副部長	岸田　剛
会計	渡利　一夫
主務	中川　靖
文連	中坊　昌也
OB世話役	加芝　直志
コンパ委員	田付　敏一
庶務	大橋　伸一郎

執行日　'83 おいコン終了時
'84年1月13日より

ネットワーク・サブ・コンピュータ(ナビコン)の製作とメンテナンス

83 学園祭

先年は、CONSにおいてネットワークの処理とネットワークを使うアプリケーションプログラムは、一台のコンピュータで、全て実行されていました。

そこで本年はネットワークの処理を全て実行するサブ・コンピュータを製作しました。

このコンピュータの構成はCPU部、メモリー部、I/O部よりできています。CPUにはZ80-CPUをメモリー部には、プロトコル用にROM 2716, ワーキング並びにバッファリング用にRAM 6116を、そしてI/O部はZ80-SIO(0)を使いました。ネットワーク, ホストコンピュータとの通信はすべてRS-232Cに従っていますので、そのバッファに1488, 1489を使っています。シリアルI/Oには8251(USART)を使用することも考えましたが、今回はネットワーク

通信とホストコンピュータとの通信の2チャンネル必要なため、Z80-SIOを使いました

製作ですが、6~7台のコンピュータを作ろうと考えていましたので、プリントパターンも作ってやればよかったのですが、回路のミスやプロトコルの製作途中でもあって、多々回路変更を生じることが予想されたので全コンピュータとも手配線で作りました。やはり手配線では配線もれや誤配線、ハンダ付け不良等により信頼性は低くかったので、なかなか完成には困難でした。それからコネクターですが、今回はCPU、SIO、ROM、RAMと通信用、電源に使いましたが、テストを何度も行っているうちに、コネクター部の接触不良による故障がありました。これらの故障の修理には、CPUテスターが大変役立ちました。これはプログラムを1ステップずつ進めることができまたアドレスを任意に決定してデーターの読み書きができるなど、CPUのすることなら、大体マニュアルでも実行できます。

今後のメンテナンスは、手配線でケースに入っていないので、できるだけさわらぬように大切にするように心がける必要があります。（ケースに入れてしまえば問題はない）

次回から、このような製作においての注意点。

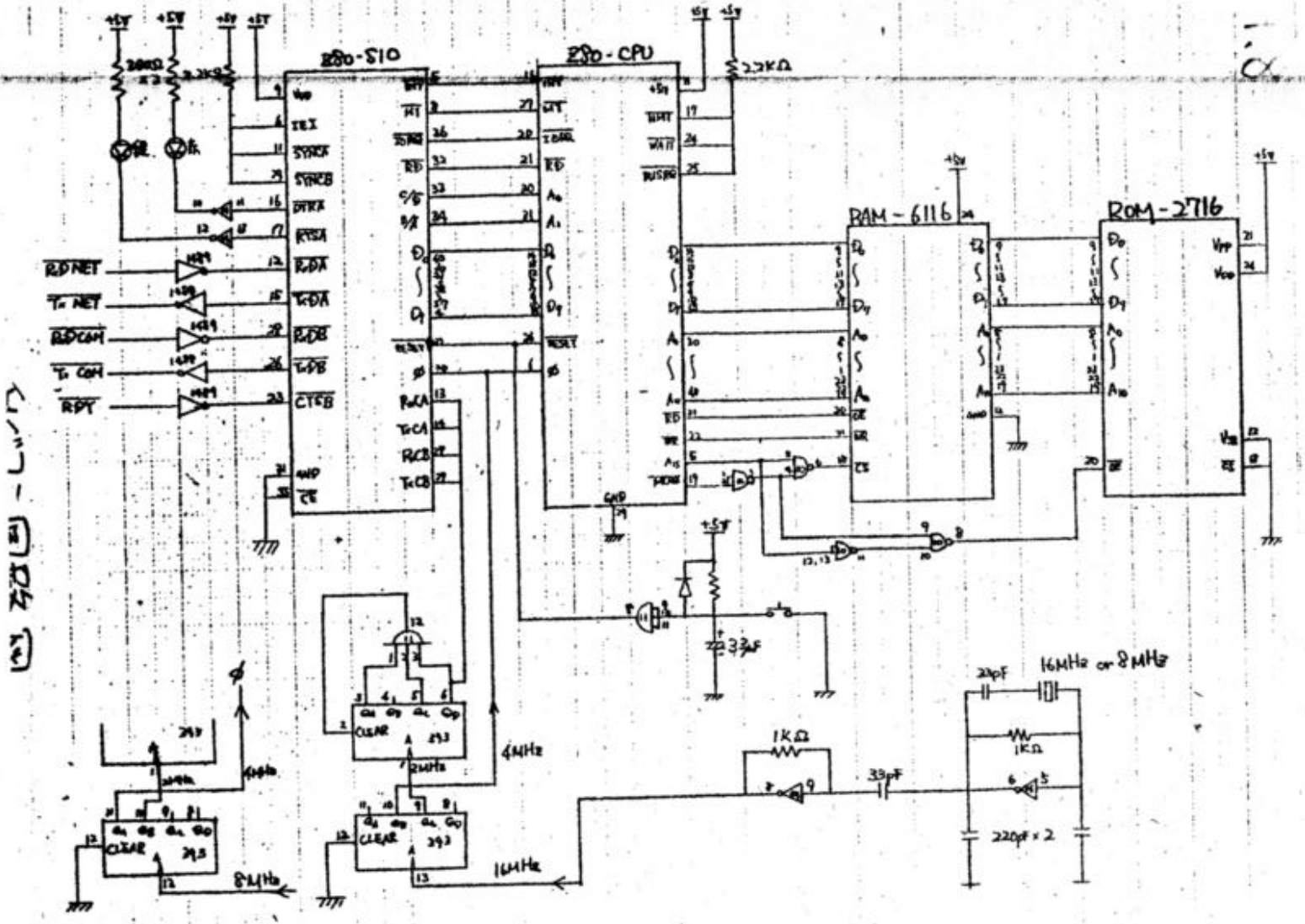
- プロトタイプをできるだけ早期に完成させ変更点のないようになってからプリントパターンを作るようにする。
- コネクターは使わないことを前提として使う場合には、できるだけぬきさしをしないようにする。

最後に、製作にたずさわってくれた

藤嵐君、田中(M)君、高津君、足立君、中川君
大橋君、慶山君

ごくろうさんでした。

（なかじま　かずき）


Z80-SIO
Z80-CPU
RAM-6116
ROM-2716
2.2KΩ
RDNET
TxNET
RDCOM
TxCOM
RDY
16MHz or 8MHz
33pF
1KΩ
220pf x 2
4MHz
2MHz
8MHz
1MHz

Context Free...

今どういうわけかPC-8801を借りている。しかもCP/Mのオマケが沢山ついてきている。ぼくは毎日CP/Mを起動し、おもむろにmulispと打つ。するとアホな、いや、かわいらしいLispがお出ましになる。これがまた何もないLispで、かわいそうなので手足をLispで書いてつけてやる。最近は学祭が終った暇でもってpretty printerをつけてあげた。前につくった構造editorと仲良くさせてやる。そんなこんなが今年の7月、PCがウチに来てから続いている。Lispはカッコだらけで、読み書きしにくいという人がいる。しかし、pretty printerを使うとずっと読み易いし、構造editorだとほとんどカッコの数を数えなくてすむ。要はtoolの問題であり、Lispでチョイ、チョイと書いてしまえばよい。(ただpretty printerはむずかしいが、それはLispの問題ではない。それどころかLispのそれはやさしい方だろう)

それでもLispのカッコには欠点がある。典型的なのが算術式。そこでLispも演算子中心に書式をかえてしまえばどうなるか。

まず1引数関数は単項演算子になる。4!の"!"などがそれである。これと同様に、

(car '(a b))　は　'(a b) car.
(sin 3.14)　は　3.14 sin.
(car (cdr x))　は　X cdr car.

'(a b) は (a b) を評価しないという印でQUOTE関数ではない。

2引数関数は二項演算子になる。

(plus 3 4)　は　3 + 4.

二項演算子は、＋－＊／＾などが多い。特殊文字に統一しておこう。それで、

(cons x y)　は　X @ Y.
(append x y)　は　X , Y.

などとする。すると,

(append (append x y) z)　は　X, Y, Z.

①特殊文字でできている演算子は二項演算子である。(ただし使ってはいけない文字もある) ②英数字でできている演算子は単項演算子である。③優先順位は②の方が①より高い。大体こんなものだろう。

(cons (car x)(cdr y))　は

X car @ Y cdr.

さて3つ以上引数のある関数は？これは

(subst x y z) は X subst: Y in: Z.

というスタイルの演算子を導入する。

④ subst: ～ in: ～ で1つの演算子であり、(X subst: y) in: z. ではない。⑤英数字列の後にコロンをつけた演算子は二項以上の演算子（キーワード演算子とよぶ）を担当する。⑥キーワード演算子の優先順位は他の2つより低い⑦同類の演算子に属するものは同じ優先順位とする。

3 + 4 * 5. の値は 35

となる。）

以上は SUBR, EXPR タイプの演算ばかりだった。FSUBR はどうする？ 簡単なところで setq。

特別な演算子 ← として、setq を使う。

(setq x (car y)) は X ← y car.

Xは評価せず、優先順位は自分自身も含めて、他のどれよりも低い二項演算子である。したがって

X ← Y ← Z ← '(a b).

とできる。

progn などはどうだろう。

(eval '(car x)) は '(x car) eval.

(progn (setq x (read)) (print x)) は、

'((X ← keyboard get) (X print)) progn.

そこで楽なように特殊カッコを用意する。角カッコで

```
[ X ← Keyboard get.
  X print. ]    は
    '((X ← Keyboard get)(X print))
```

と同じことである。

これだけではなく,

```
[ X  Y | X ← 'a. y ←  .] prog. "1"
```

ということもできる。また,

```
[ X Y |  ····] let: '(a b).
```

とすれば、XとYにはそれぞれaとbがバインドされて評価する。そういう[と]を仮に用意できたとする。

これにより制御構造がつくれる。まず分岐から

```
X ifTrue: [ ····· ].
X ifFalse: [      ].
X ifTrue: [       ] ifFalse: [      ].
```

X = nil なら ifFalse:の[ ]を,それ以外なら ifTrue:の方を評価する。ループは、

```
X whileTrue: [      ].
X whileFalse: [     ].
```

X = nil の間回るのが下の方。

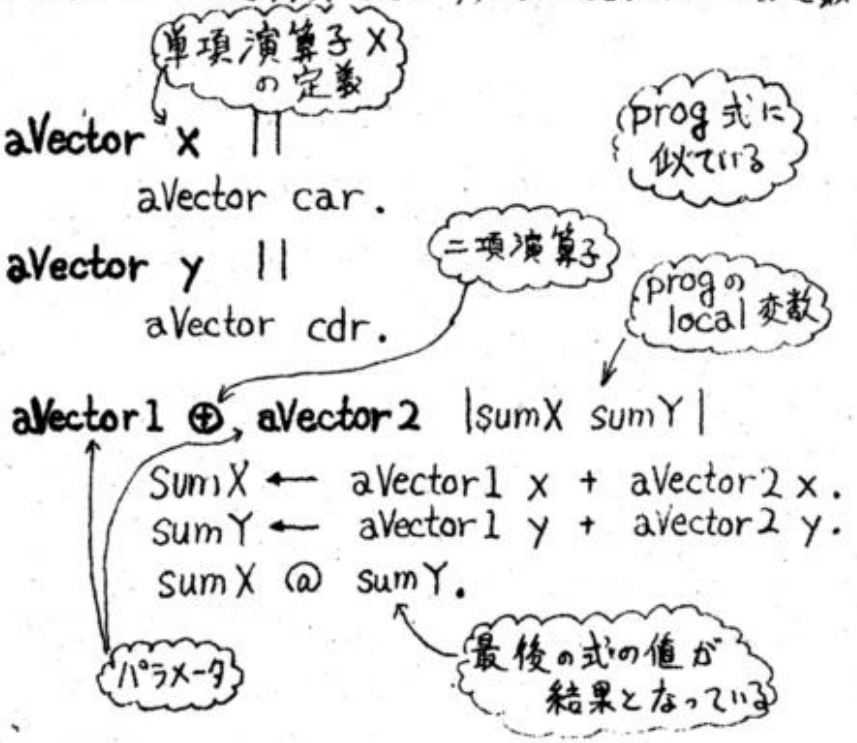
goto はない? しかし, Lispですので...

つくれないこともない。

あとは関数の定義ならぬ演算子の定義である。
2次元のベクトルの演算の例がやさしいだろう。
まずベクトルは、

X ← 2@3.　{Xは(2.3)となる}

という形である。以下ではベクトルの加法⊕の定義



これで、

X ← 2@3.　Y ← -1@4.

Z ← X⊕Y.　{すなわち Z=(1.7)}

システムがグラフィックをつかえるなら、

'(783.508) plot.

は(783,508)という座標に点をうつ。

window plotLine:'(100,150) len:100 angle:40 deg.
は(100,150)から40°で長さ100の線を引く。
windowは((10.20)(500.500))といったリスト。
あとは想像にまかせる。
append もやさしい

```
X , Y ||
    X null ifTrue:[Y] ifFalse:
        [X car cons: X cdr , Y .].
```

cond もつくるとどうなるか？
書式は

```
[ X atom  → [X = Y].
  Y atom  → nil.
  X car == Y car → [X cdr == Y cdr].
  T → nil ] cond.
```

という感じで。"→"は@と同じ。それでcondは

```
aBlock cond |signal|
aBlock null ifTrue:[nil] ifFalse:
[(signal ← aBlock car eval) car
    ifTrue:[signal cdr progn.]
    ifFalse:[aBlock cdr cond.]].
```

となる。

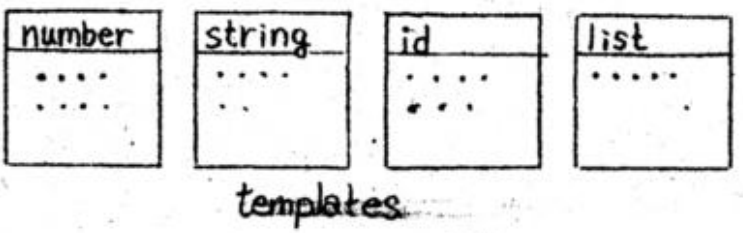
ところで、

4 + 2　"A" + "BC"　'(A B) + '(C D)

というふうに + という演算子を使いたい。このように名前を同じにして異なる演算ということは多い。そこで今までどおりに

```
X + y. ||
  [ X numberp → [ X plus: y.].
    X stringp → [ X concat: y.].
    X idp     → [(X mkstring concat:
                  y mkstring ) mknam.].
    T         → [ X append: y ].] cond.
```

とするか？しかし，これはメンドウだ。他の多くの言語はデータ型の宣言によって上のような名前の重複を人間に許していない。わずかに孫手続など階層構造で逃げている。でも，Pascal の write 文などはどうだろう？(これは駒嶽せんせいのしてき)とにかく，上のようなことは避けたい。そこで template というものがあって リスト，アトム，数値アトム，文字列アトム などの名前をつけてやる。

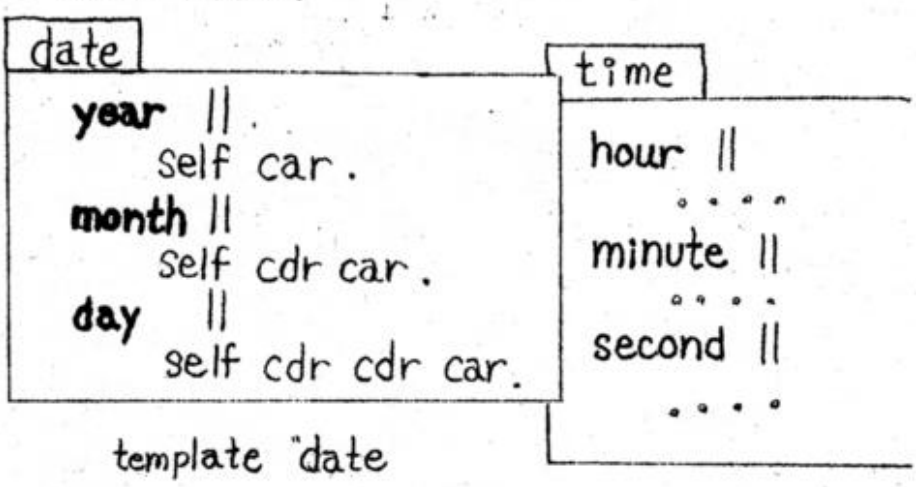


そして、そのtemplateに演算子の定義をかくと、template と同じ型のデータのとき、その定義された演算を実行する。

```
2 + ....
```

とすると、ここで 2 があるので、+ は数値演算の加算に変身する。"A" + ... '(a b) + ... も同じ。

'(1983 11 26) と '(12 24 13) は、前者が date で後者が time である。



date型のリストにはyearとかmonthという演算を行なえるが、hourやminuteは未定義である。(ここで引数が消えているが、第1引数はselfという変数になっている。したがって第2引数以後は書く必要がある。) これは、ちょうどレコード型に似ている。また、Lisp風には、名前(id)のみならずS-式には必ずプロパティがあると思ってもらえばよい。同じようなリストでも、プロパティがdateとtimeでは扱いが異なる。同様なフレームでも、入るデータが異なる。(月は1~12

の整数だが，分は0～59の整数）たとえば、

— date —

```
newMonth: X ||
        1 <= X <= 12 ifFalse: ["date-month" error.].
        self cdr replaca: X .
        self.
```

として，~newMonth: 13 を実行して，エラー処理に入る。
(ここでのerrorはシステムレベルでも，ユーザーレベルでも
errorの定義によるし，バックトレースして再度やってもよい。)
ところが，「13月を入れると次の年の1月にする」と変更が
おこる。 newMonth: ~ の定義を変更すればよい。
こんなことより、yearとhour，monthとminute
のように，同じ操作が定義されていてメンドウではない
か。そこで （。。。。…） というリストをSequential
list型として、

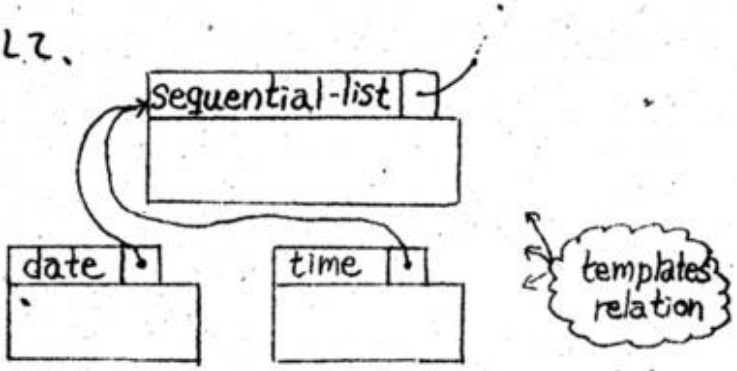
Sequential-list

```
at: N ||
    (self nth: N ) car .

put: X at: N ||
      (self nth: N ) replaca: X .
       self.
nth: N ||
    N = 1 ifTrue: [ self ]
          ifFalse: [ self cdr nth: N -1 ].
```

P.18.

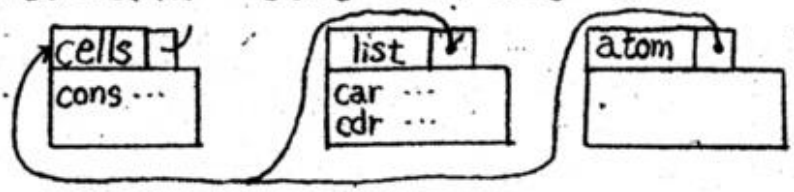
を定義して、



という連絡を設ける。 date型データをさす変数Xにおいて

X year. X at: 1.

は同じ結果を返すことになる。つまり, date型で、そのtemplateにない演算子は、sequential-listにあるかどうか見るようになる。timeも同じである。またyearもhourもtemplate内では self at: 1. と扱える。2つのtemplateは、sequential-listというtemplateの操作を相続することができる。逆に言うと, sequential-listにdateとtimeというインターフェースをつけたことになる。さらに, sequential-listの中にもcar, cdrなどの記述はないがsequential-listでcar cdrが使えるのは、上位にlistというtemplateに連絡しているからである。そしてlist中にcarやcdrの定義がある。consはもっと上位のcellsというところにある。



上位へと連絡をたどると，システムレベルまでが見えてしまう。その template には もろ list-processor の記述がある。のぞいても良いし、のぞかなくても良い。 また、どの Layer から がシステムレベル などということはユーザーによって ちがってくる。(ふつうの Lisp では あたりまえのこと。ただし ふつう というのは Lisp machine Lisp などのこと。)こうなると，ある特定の Ework*(仮名)とよばれる人たちには たまらない オモチャ となる。(少々 値は張るが...)

話をもとにもどす。ちょっと 待てよ? もし、いきなり

```
'(1983 11 26) month.
```

としたら、'(1983 11 26) を date 型とわかってもらえるか? (1983 11 26) と書くと Lisp では自動的にリストを生成してくれる。そして、それは list 型である。

```
aDay ← '(1983 11 26) prop: 'date.
aDay month.
```

とでも しなくては いけない? でも

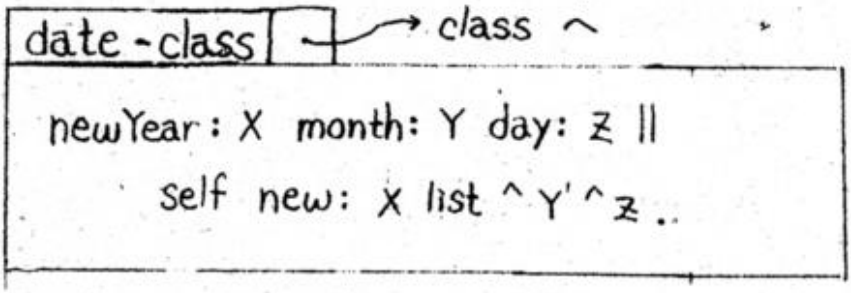
```
aDay ← 'aho prop: 'date.
```

ということも できてしまう。つまり データの発生にも立ち

---

*Ework : Star Wars 皿 に登場する森林惑星の原住民。つまり，おのとかくわでたたき切ったりする人々。ただ帝国軍との戦斗では、武器の種類の豊富さで、帝国軍を圧倒していたのは、Lisp に通ずる かわいらしさと スゴさがある。

会わなければならない。そこで，template で



をつくる。そしてグローバルな変数 Date に date-class 型のデータ 'date をバインドさせておく。この template は class という template につなぐ。そして

aDay ← Date newYear: 1983 month: 11 day: 26.

すると上の template の演算がおこなわれる。X new: y は、y というデータに X というプロパティをつける。そしてそれは、class という template に定義されている。こういう定義は date の template と放さずに定義しておく方がよい。そして変数 Date にバインドしておく。

そろそろ紙面も少なくなってきたし、疲れてもきた。こんな物を誰が読むのだろう？と思えてくるのはいつものこと。いいかげんな文面と内容に目をとおす。プロパティ？ property-list と少しちがうなあ。誤解を招きそう... しかし，この内容はフィクションであり、現実のシステム，言語とは何らかかわりあいは、

あります。pretty-printerを使ったとき，閉じカッコが、画面の右端からこぼれしまい，やれやれとしたとき，カッコが多いのも困ったもんだと思った。それでカッコが減らせないかと思ったのが前半。ところがどう間違ったか、データ型の話になってしまったのが後半。行きあたりばったりで、好きなよう述べさせてもらっているわけだ。何よりも短く話すためにLispを知っているという前提のもとに書いている。(近刊のASCIIにもLispの紹介が連載されていたようだが..?) 手を抜いた説明も多い。欲をいうとSmalltalkを知っていると十分パロディになる。しかし、全部がSmalltalkをPalo(?)ているわけではない。あえてオブジェクトという概念をさけて、作用素(演算子)中心に説明した。(素直にオブジェクト指向ふうに説明をすれば楽だったか?)

今まで(といっても十年以上前)の言語では、制御構造のmodularityという点が強調された。ひとつには、扱うデータの範囲が狭かったからだと思うし，すべてそのわくでしか考えなかったのだろう。(それでは人工知能屋さんが困る。)ところがデータ構造の方が複雑になってくると、データ構造の変更にともなって、あちらこちらの手続きの変更が多くなってくる。そこでデータに一種のインターフェースをつけてやることにする。そうするとデータの"implementation"にかかわらず、observalであり、modularityが増す。これは何も目新しいことでないことはわかると思う。そして要はデータ処理なのだ

から、手続きなどはデータの内側にしまえばよい。こうしたdatactな言語としてCLU, Mesaなどがある。さらに、洗練されたsemanticsをもったのがSmalltalkである。

PARCでは十年もの間Smalltalkのimplementをやっていた。ところが実は、IBMも同じようなシステムを研究していてシステム/38という。(今年のbit, 10月, 11月号に紹介がある)違うのは、パーコンと大型という点である。

あっさりした仕様もないことを書いてしまった。たしか、このLimeは「学祭まとめ号」のはずだ。とすると、この原稿はいったい何なのか?)))))

参考にしたご本。

- "The Smalltalk-80 System" Byte. 1981. 8.
- "22. Frames" Lisp (P.H. Winston)
- "抽象データ型言語" 情報処理 1981. 6.

——この原稿はすべてパロディです。

(@)

APPLE の SOFT について

大橋

今までAppleは、日本の雑誌から疎外され、Appleのプログラムはまず載らなかったが、この頃たまに見かけるようになってきたのは実に喜ばしいことです。ソフトもLOG-INやMICOM GAMESなどによく載っています。I/Oやマイコンはダメです。特にうれしいのは電波新聞社からAPPLE DOS入門という本が発売されたことです。

この本は一般的なDOS使用法から、実用プログラム、プロテクトのしかた、などAppleを使うのに便利なことがたくさん書いてあります。

さて、本題に入りますと、Apple のSOFTは実にすごいと感心するところがあります。例えばファンクションキーなどのハード面で日本のコンピューターに遅れても、すぐにそれを克服できるソフトやカードが発売されることです。以下その例を示しますと、...

(Appleになかったもの)	(それに対応するソフトなど)
ファンクションキー	double-take (ソフト)
16ビットのスピード	6502 3倍速カード
	68000 カード (12MHz) (CPU)
CP/M 86, MS-DOS	8088 カード (CPU)
CP/M	Z-80 カード (CPU)

特に double-take は リストの上下スクロール、リストの清書、RENUMBER AUTO-NUMBER などができる便利なソフトです。

下のプログラムは Lab. Letters に載っていた清書プログラムです。

```
0300 - A5 67 85 D2 A5 68 85 D3
0308 - A5 D2 85 D0 A5 D3 85 D1
0310 - A0 00 B1 D0 D0 05 C8 B1
0318 - D0 F0 2C A0 00 B1 D0 85
0320 - D2 C8 B1 D0 85 D3 18 A5
0328 - 04 65 D0 85 D0 A5 D1 69
0330 - 00 85 D1 A0 00 B1 D0 C9
0338 - 3A D0 CD A9 20 91 D0 E6
0340 - D0 D0 02 E6 D1 D0 EC 60
```

これによって左のプログラムは右のようにリストされます ：はコロンです。

```
10 FOR I=1 TO 100
20 ::: A(I)=0 : B(I)=1
30 :: FOR J=1 TO 50
40 ::::: C(J)=1:D(J)=20
50 :: NEXT J
60 NEXT I
```

CALL 768
LIST

```
10  FOR I = 1 TO 100
20    A(I) = 0 : B(I) = 1
30    FOR J = 1 TO 50
40      C(J) = 1 : D(J) = 20
50    NEXT J
60  NEXT I
```

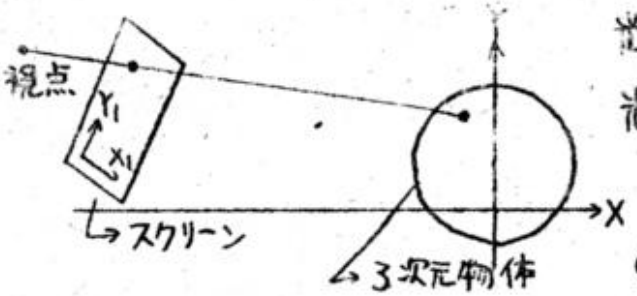
ところでこれからの目標ですが、話をきくとグラフィックのドットが細かくなり カラー数が増える、ということなので Apple の 280×192 をもっと細かくしたいので、コンピューターグラフィックスをやろうと思います。

なんか、Apple にもグラフィックボードとかいうお高い、9220 を使い、1024×1024×2、512×512×4 などの分解能をもつ、スーパーリゾリューションボードがあるそうなんですが、詳細は不明です。

パソコンによる安上がりな COMPUTER GRAPHICS

中川靖

最近コンピューターグラフィックスというものをテレビや雑誌等でよく見かけるようになったが、その時のCGを、パソコンでやろうとバカな事を考えている。CGにも種類はいくつかあるが、その方法も今はやりのRay Tracing法でやろうというものだ。Ray Tracingはごぞんじの方も多いと思われるが、かいつまんでいうと、図のように、



光というものを1本の光線として表現して、スクリーン上のこれから求める画素と視点を一本の直線で結んで、もしその先に物体があればその画素は物体表面の色や照明によって決定される色、輝度であり、もし先に物体がなければ

P.26

その画素は背景色であるというものだそうだ。
(あまり理解してない)。これをパソコンで行うのは
絶対に不可能であると書かれている雑誌がみうけ
られるが、頭をつかえばできないことはない。パソコン
で行うのにまず問題となるのはスピード、そして表示色数
である。スピードの点は、一度できた画面を次回以降は
DISK上にでもおとしておけば次使うときにすぐ見れ
し、我慢して何日もかけてパソコンに計算してもら
うことにすればなんとかなる。一方の表示色数だが、
これだけは大量のメモリーと高速D/Aコンバーターが
ないといけないようだが、しかし、ふつうのデジタル型の
CRTとその単なるBGR各々1枚ずつのディスプレイ
でも、多重露出をつかい、1枚ずつCRTに表示
して、多重露出を何回かくり返すと、例えば薄い
青なら何回も多重露出することによって表現できるから
金なし工機大生にもCGができる。他にも
安上がりでやる方法がいくつかあるそうだがよく
わからんので勉強しときます。

何でもあるプログラム

by A.F.

PC-8800を使用している人、N88モードでDISK BASICが立ち上がったらな — と思っている人はいませんか。

今回 DISK BASIC(N)をN88モードで立ち上がるように。IPLも改造しましたのでお知せします。

まず ドライブ1に N88 DISK BASICの DISKETをいれ、N88 BASICを立ち上げます。それから ドライブ2に DISK BASICのディスケットをいれます。

MONでモニターに入り次のようにします。

```
h1fd000,d2ff,ff

h1^r2,0,0,1,d00c,d10b
h1^r2,0,0,2,d10c,d20b
h1^r2,0,0,3,d20c,d30b
```

それから リスト1を打ち込みます。

```
D000:3A 09 00 FE 6A 28 05 3E
D008:04 D3 31 C7
```

リスト 1

次に リスト2に したがって データーを打ち込み直します。

```
D039:C2->CE D03C:38->44
D042:BC->C8 D08E:87->93
D09B:97->A3 D0E4:68->74
D0F4:49->55 D0FD:49->55
D106:49->55 D10B:A5->B1
D111:49->55 D116:A6->B2
D13C:A7->B3 D143:68->74
D250:A8->B4
```

リスト 2

打ちまちがいがないことを確認したのち

```
hJ^w2,0,0,1,d000,d0ff
hJ^w2,0,0,2,d100,d1ff
hJ^w2,0,0,3,d200,d2ff
```

とします。これでドライブ2のDISKETに新しいIPLがSAVEされました。こののち、このDISKETをドライブ1にいれてRESET SWを押して正常にN BASICが立ちあがることを確認します。正常に立ちあがらない場合は打ちまちがいですから もう一度やりなおして下さい。

以上で、改造は終りました。本体にN88モードからDISK BASICを立ちあげて下さい。

但し、失敗してもだいじょうぶなように、必ず改造の前にはBACK UPを取っておいて下さい。

K.Z4

Software Industry
v.s.
Software Science

Take

㊐の中で 虫の無いProgramは良い Programと思われている。

㊑かし、それでは、虫の有るProgramは悪いのか?

▷物には、耐用年数がある。投下した資本に対して、ある期間使用に耐えれば、もとがとれた、という奴よ。Hardには、耐用年数を認める人が多い。

▷ところが、Soft.には、完全主義者があふれかえっている。しかし、5年もたったProgramは、使用環境が変化して、陳腐化してるに決ってる!! そんな物のメンテナンスに金をかけてると、新規開発してる方が安くつく。

▷そこでもしその耐用年数内に見つからないBugはBugにはならない。使用期間中だけちゃんと動けば、完璧というもんだ。

▷Reasonableな値段で、物を生産することは満足できる値段で満足できる機能を与えることである。

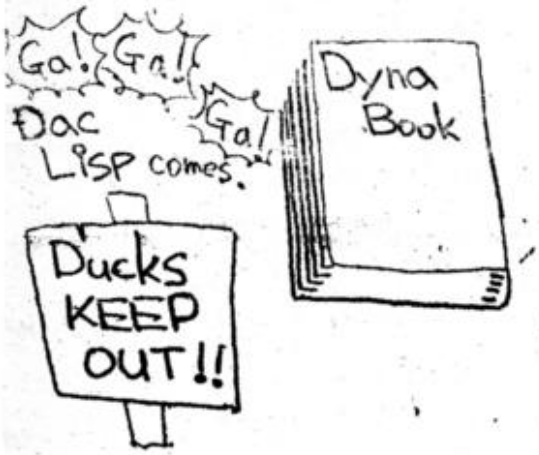
□もし、Man Powerがただなら無限に近い人間と投入して巨大かつ強力な(この2つが共存するかどうか?)Systemを構築できる。その上に、開発時間が無限なら、非常に強大なSystemを作ることも可能だ(?)
□しかし、世の中では、有限な時間内に、有限な資源で、有限な(幸いなことに)Systemを作らねばならない。その時、でき出たSystemがそこそこ使えれば(?)それは、Payするといわれて、なかなか使えれば、それは、よくできてるといわれ、ぶいぶい使えれば、然也……それは、それは、えらいとしか言いようがない。
□また、物は複雑度がませば、その管理費(生産時においても、運用時においても)にも金がかかる。ということは、安かろう悪かろうと言われても、必要以上に十分な機能、あっても、めったに活用されない機能というものは切り捨てたとしても、十分な利点がある。
□もし石井家ならば、Pay等ということを要求されることは、めったにない。学校ならば、Man Powerは無料なので、Debugに死ぬほど手間をかけられる。
□しかしながら、銭もうけでは、そうはいかない。世間のちっぽけな(いかがわしい)Soft wear House等では、2~3人の小人数で、せっせこと手作り風の職人仕事をしている所もあるが、そんなのは、前近代的だ。

□Softwear でも、Muss Product をやらねばだめだ。
化学に工業的方法というのがあるように、Programも
工業的方法で、大量生産するのだ。
□つまり、あるToolを使い、ある方法で、一定水準以上の
人間を投入すれば、常に一定水準以上のProgramが
でき上がるということだ。しかも、一定水準の人間と
いうのは、現在いうところのProgrammerでは、なく、
もっと大量雇用の可能な普通の人でないとだめだ。
□ここで、問題になるのが、製品(Program)の品質で、
それは、一定の水準しか保障されていないので、
おおむね、複雑な物を生産しようとすると、Bugが出る。
これが、耐用年数内で見つかると、Bugで、そうでないとき
Payする線に、あった、Resonableな欠陥であった。
□もちろん、ここで、もう少し、高い金を払えるなら、耐用
年数中に Bugに出会わなくなるようにできるだろう。
つまり、その時点、もう少し、高い水準の製品を手に
入れられるということだ。
□けれども、Softwareは、車や、電気製品とは、違って
一つだけを、職人的に作り上げても、それから大量の
Copyを生みだせるので、Softwareそれ自身の
作り方は、車ほど、マスプロでなくても良いと思われるだろう。

1.>
□しかし、今や、ミシンや電子レンジにまで、Programmが必要なのだから、量のみだけではなく、種類もたくさんいるのだ。つまり、多品種を短期に大量生産できるような体制を必要とされているのだ。
□だから、Payする線を保って大量生産は、行なわれるべきものである。
□だから、耐用年数内に見つからないBugは、Bugでない。
□さぁ、それは、どうでもよいのだが、一定水準の人間で一定水準の製品を生み出せる、Toolと管理法は、Software Industryとしては、是非とも、必要なのである。
□ぼくたちは、Industryを やるのさ。

(Jun/19/1983 Tak)

私、素人の使える、(プログラムのいらない)マシン(Work Station)ができたら、(普及したら)プログラムの需要は、激減するだろーか?



LANGUAGE(Prolog)←
SMALLTALK talk:forever
(L(I(S(P))))
BCPL→B→C→P→L
Microsoft's 4MByte BASIC

アルバイト論 云々

by Akira

Lime前号の記事を読んで感想を……

コンピュータ部々員が、コンピュータ関係以外のアルバイトをすることに、賛成です。私自身、大学入学以来家庭教師のアルバイトをやってます。ただ、コンピュータ関係のアルバイトをしてはいけないということはないでしょう。何故ならコンピュータ関係のアルバイトといってもそれ以外の知識を吸収できうるからです。アルバイトをするということで他人と接することができます。学生生活において人と「おつきあい」することはとても大切なことです。将来、職につくと専門的知識にかたよることが予想されます。大学生としては、勉強の他に、社会生活において一個人としての人格を形成するという目的をもっています。多様な知識に少しでも首をつっこんでおけば、いずれ役にたちます。つまり、ある専門家としての視点からの他に、別の視点からながめなければならないことが、あります。この時、一方向にこり固っていては、すばらしいものはできません。

人間は、すぐに対応し、しばらくすると慣れてきます。そしてそこに安住する傾向をもちます。したがって違った刺激が必要なのです。全く別の刺激によって、目をさまされて

P.54

さらに進んだ知識を得ることがあります。

コンピュータ関係のアルバイトは ある程度、専門知識を有する部員にとって とても魅力あるものです。したがって専門的アルバイト、それ以外のアルバイトについて「やってはいけない」「やらねばならない」ということは、ないわけです。

要は 学生時代におけるアルバイトで、実社会で役立つ知識などを吸収すべきなのです。

コンピュータを趣味とする人は 俗に性格が暗く陰気で一人にこもりがちだといわれています。何故なら、他の人と接する機会を自ら制限する傾向にあるからです。これでは人としての機能の欠落をまねきます。こんなことが、自分の身の上に起こっては 困らないのです。

そこで、こんなことが起らないように、アルバイトを役立てて、いろいろな知識を吸収すればいいのです。

Random Access Talk

この題目 どーゆー意味やろ?

書いたやつが書いたやつだけ なんや よーわからん!

by ふる

← Random Access Talkの略やで PARTとは ちゃう!

RAT-I

とーとつに『学祭を終えて』の感想から…

例によって学祭は内輪だけの楽しみ(苦痛が快感になったのかもしれんが)に終ってもたみたいや。小生作天そばチビコンも、うまいこと動かんかったし…。

あんなもん動かいで当然じゃ!

結局,展示も部屋すらよくわからんよーなもんになってもた。

0天そばチビコン…テンプラはんだ付盛そば型チビコン

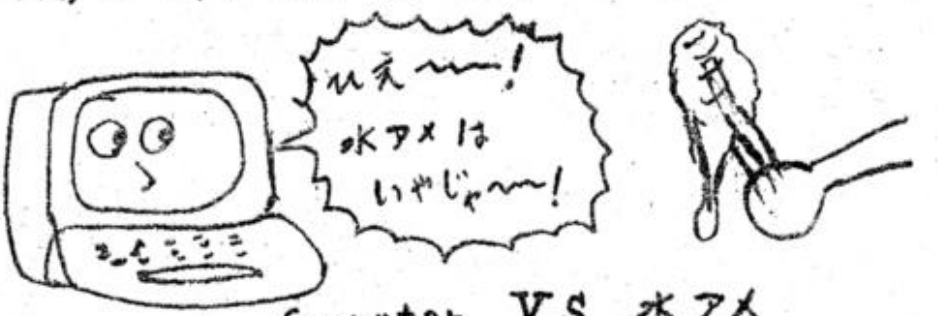
なんで毎年、学祭はこうパッとしいひんのやろ。やっぱりふだんの部活からしてだらだらとやとるからやろか。他大学のコンピュータ部なんかの学祭での活動を見たことが無いのでどうとも言えんが、人目を引くよーなもんがでけんのやろか。まあどんな部でも人目を引くよーなもんをやっとるとは言えへんけど。

ゲーム屋をやれば人、(正確には美少女)御子様がよーけ

チビコン達を、責めないで

私はチビコンが嫌いです。
チビコンは高度で礼儀知らずで気分屋です。
リセットをかけると暴走し
放っておくと発熱する。
学祭までは悩みの種
学祭終わればジャンク板。
そんな御荷物みたいな
そんな宅急便みたいな
そんなチビコンが嫌いだ!
私は思うのです。
この世の中からチビコンがいなくなってくれたらと。
チビコンが私達の世の中のために何かしてくれたでしょうか。
いいえ、チビコンは常に部員の足を引っぱるだけです。
多勝ってで"足が臭い"
パケット、エラーパケット
"円"入りパケット
パケットを食べてばかり
エラーパケットを回して遊ぶ。
あの世間体を気にする発光ダイオードが嫌いだ。
チカチカしない発光ダイオードが嫌いだ。
私は本当にチビコンが嫌いだ!

集まってにぎやかで人目を集めるみたいだが，結局は何もやとらんのと同じなんよね。



Computer VS 水アメ

今まで書いたようなことは毎年，言われてきたことで，そう知りつつ，今年もヌ，愚かな事を繰り返すのである。まったく，人間とは愚な生き物のである。

RAT-Ⅱ

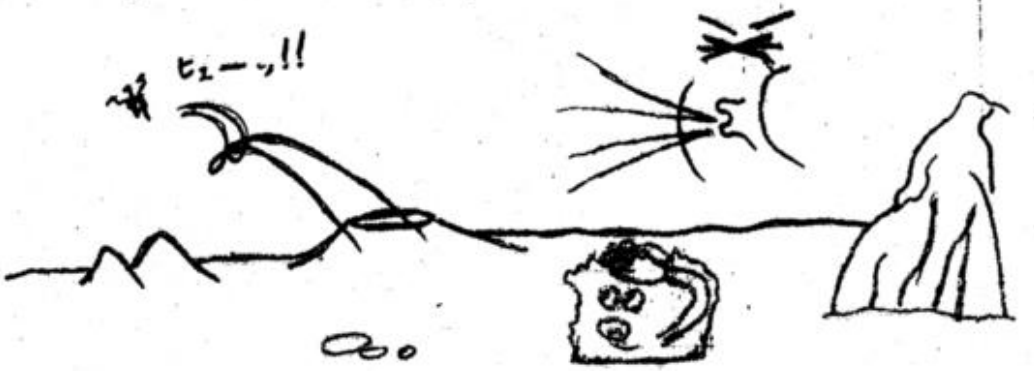
先日，『新・第3の選択』(水島保男著 たま出版)を読みました。内容は，米ソが月や金星・木星など人かに探査機を送り込んでるが，その探査データが隠され，ほんの1部の操作された情報しか公開されていないとゆーものだ。中には，ほんまかいな，ちょっとこじつけすぎとちゃうかと思うところもあ

たが，なかなか読みごたえのあるものだった。

特に，金星について，探査機が送り込まれたにもかかわらず，そのデータが皆無と言って良いほど公開されとらんことに興味がわいた。

ソ連が4回ほど，アメリカは1度に5機も送り込んで失敗しとる。ソ連の方の失敗は，3回ほどは着陸したけど金星の硫酸性の雲と，400度という気温，90気圧という大気によってつぶれたというもので，つぶれるまでに送られて来たというデータのほんの1部が公開され，写真も1枚だけ公開された。で，その公開された写真には，探査機の足の1部が写っていたが，硫酸性の雲の中を通過したにもかかわらず，腐食せず，ピカピカに輝いていた。又，金星上では風速数十m/sの風が吹いていたとなっとるが，90気圧中での数十m/sは，1気圧では数百m/sというとてつもない風圧になり，そんな風圧を受けても，じっと地面に着地しているがマンの

でけるよーな探査機など考えられん。



そんなところへ，1度に5機も探査機を送り込むというアメリカの考えもよーわからん。

で，その本によると，これらの温度や気圧等のデータは，それ以前の天文学の値に合わせただけであって，実際のデータではないと述べている。

そして，ボソは何らかの目的で，事実を隠し，何かわからんが，宇宙で何かを行なおうとしているのだそーだ。

これを読んで，ウンウン，そーかと思ってしまった。この本にも書いてあったが，探査機を送り込むという，画期的な事を行なったにもかかわらず，目新しい発見は何もなかったと言ってもよいほどである。

P.40 新しい技術が応用されると格段に進歩するのが最近の学問である。半導体が電子機器に応用されて、今日のコンピュータばやりになったし、遺伝子のDNAの2重らせん構造が発見されるやいなや分子生物学という学問ができ、バイオテクノロジーばやりになっている。

ところが、探査機を送り込んでも、公開されたデータには目新しいものがなく、ごくまれに、公開された写真等に写っている不可怪な物体(以、UFO等)についてはいっさい答えていない。

やっぱり何か隠しているとしか考えようがない。

もっとも、不思議なのは、アポロ計画等で月へ行った宇宙飛行士達がその職を引退すると、どーゆーわけか、宣教師になったり、哲学を研究しはじめたり、想像でけんよーな、とゆーより、人生とか生命とかについて倫理的哲学的研究をはじめる人が多いのである。

このことについてNASAは“地球を宇宙から見る、月に立つ、といったような通常では考えられない体験がその人の人生観を変えるからだろう。”と説明している。

これは，あまりにも有名であるが，アポロの宇宙飛行士の一人（誰やったか忘れた）が，地球に帰って来てからというもの アルコールを手ばなさない日がないほど酒のみになってしまい，いつも『NASAの偽装（カムフラージュ）につかれた』，『我々はNASAに利用された』等とのたまわっているのだ。

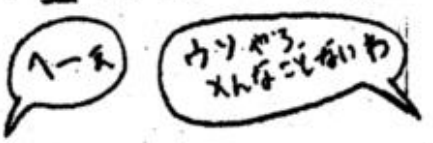
本にも書いてあったが，宇宙飛行士達は，事実を口外するのを禁じられているだけでなく，その事実が世の中に与える影響が計り知れないものなので，口外する勇気がないのである。

もし，あんたさんが，宇宙飛行士で，世の中をひっくりかえすよーな，大混乱を招くよーな事実を知ったとして，それを誰に話せるだろうか。

たった一口で世の中がムチャクチャになってしまうというPresserに耐えられるであろうか。

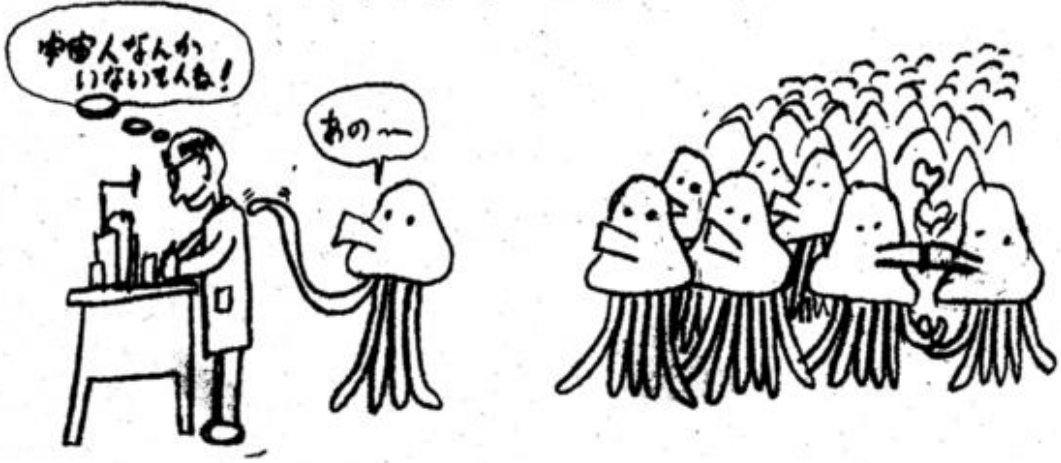
RAT-Ⅲ

高位のヨギ（ヨガの修業者のことです）になると，テレポーテションができ，他の星の住人と交信できるとかいうことである。



RAT-Ⅳ

科学者の中には，地球外の知的生命体（この言語の意味もよーわからんが…）なんか，いるわけないもんねー！と思っている人がいるのです。この地球人類だけだと思っとるそーです。



RAT-Ⅴ

「先生，重力場における空間歪の解析のプログラムができました。」

「では，すぐにＴ大へ送ってくれ。」

「はい」

「しかし，コンピュータは便利だ。やりたいと思ったことを言うだけで，自動的にプログラムしてくれる。昔は，プログラマーとか言う人がいたらしい。又，コンピュータの仕組を知っている人がいたらしい。今の我々にとってはコンピュータはブラックボックスだが」

こんぷゅーたで10倍くるしむ法

TakeB

おつき君に、『先輩は、Tomさんみたいに、どんなふうに生きてきたか書きはれへんのですか』と尋かれた。『そんなもん読んでどーすんねん』と答えると、『どーやったら、Computerが、わかるようになるか参考に…』ということなので、だいそれたこととは思いますが、少々の参考に。

だい1歩は、高校1年のとき、はじめること。コンピューター部というのがあったので、先生の処へ行く。『先輩は、みな卒業したし、わしは、コンピュータ解からん。』と言われたが、『やりたかったら、しっかりやりなさい』とも、言われたので、その気となった。時に1976年、マイクロプロセッサ且、あるにはあったが、それを知ったのもずっとあと、TK-80さえ、影もない春のうららかな日であった。

じつは、その前に、廿楽無線(銭)なるものをやっていて、CQ誌連載のLogic ICの使い方で、修業した覚えもある。しかし、でんせんも無いのに、話がとどくのは、不条理である。1975年には、CQ出版が、作るコンピュータというのが出ていて欲しかったが、手に入ったのは、ずっと後になった。

㊂て、高校にあった、コンピュータだが、その名もSX-103.
一昔前の普通高校のコンピュータ部員なら、その名も高い。
High Grade Machineなのだね。こいつは、プログラムメモリ
500 step, Data Memory 30ヶ(ただし、Data memoryは、
精度を落せば2分割して、倍に、使えた)、プリンタは、サーマルで
プログラム中にコンスタントな文字か、Acc上の数字しか
出力できないが、コンスタント・キャラクタは、英字大小に、カタカナ、
数字があり、Acc上の数字出力は、プログラムでFormatできた。
しかも、メモリによるメモリの[illegible]デバッシングもあったので、
本格的にあそべた。
㊃っきりきって、上の仕様書、プログラム要点だが、[illegible]
まだ、なかったはずだから、画期的という奴よ。しかも、
デスクトップサイズなので、ものすごく本格的に見えたし、外部
記憶に、磁気カードを使って、それなりにカッコよかった。(Canonだからね)
㊄ishは、サーマルプリンタに、ニーモニックで出力されたが、
打ち込む時は、プログラムのステップ(アドレス)しか、LEDに
ディスプレイされないので、頭の中で、プログラムのバッファを
作って作業した。

つまり、いちいちプログラムリストを取ると、紙がどんどんへるし、また、サーマルプリンタ用紙は、1巻800円もしたので、プログラムは、紙の上で、ちみつなコーディングし、デバッグする時は、どこをどう修整したか、頭で、憶えておいてやった。インストラクションのインサートやデリートは、1発だったので、しっかり憶えてないと、むちゃくちゃになるのであった。

教訓1。プログラムは、頭の中のバッファにためる。

私しゃ、(と、その一味)は、独学で計算機と闘かわねばならなかったので、Canonの Manual だけが、たよりだった。幸い、SX-103は、高等学校数学教育用という奴(文部省基準I-A型該当だと)だったので、マニュアルはこんせつ、てーねーだった。しかし、条件Jumpなどの概念は、うわさでは聞いていたものの、(1970年代のコンピュータ神話の論理判断という奴)実際に知った時は、これだけでコンピュータが、他のMachineと隔絶されるのかという感慨ヒトシオと、それを使えば、色々できそうだと、いう期待で、大感動だった。

そして、とりあえず、数あてゲームをつくり、(だれでも、やることはいっしょ)次に、文字を、入力したくなった。つぎ、ジャンケンゲームでもしたいが、SX-103には、文字を刻印したKeyさえなかったのだ。

それから1週間程小悩んで、グー、チョキ、パーを数字で表わせば良いと気付いた。つまり、計算機内部は数しか操作できないが、数で、グー、チョキ、パーを表わせば、計算機で、それを扱えることが解った。まるで、ENIAC以来の道を1人で歩んでる。

その頃は、まだ、Programは100step程度で、問題はなかった。その年の秋には、I/Oが創刊しているが、まじめな竹岡君には、知るよしも無かった。

次の年つまり、高校2年には春に、日本橋の上新で、I/Oを見つけ、コンピュータを趣味としている人間がたくさんいるのに、おどろいた。その頃、コンピュータ雑誌といえば、学習コンピュータ、コンピュートピアとbitしかなかった。日本橋の三宮無線は、前の年の夏ごろから、TLCS-12A EXPのKITを、置いていた、岡本には、ミニコンタイプのコンピュータ(中味は不明)が、お目見えしていた。そのI/Oには、TK-80広告があり、テンキーがあるのに、おどろいた。(TLCS-12のEXPは、ディップスイッチでBinary入力をさせていた。)

思い返せば高一の冬に、TLCS-12でChipから自作しようと思っていたのだった(あれらEX-Pは12万円もした)しかし、それからCQ出版の「つくるコンピュータ」にやっと出会い、それにそって作れば非常にTinyなTTLマシンがケース無しで1万円ぐらいあればできそうだと思っていたのだった。(これも高2の春ごろ)

教訓2. 貧乏人は、Hardを自作する。

高2の夏は、先生の計いで、工業高校へFortranを習いに行った。HITAC-1のとかくとーしちまった。日立のASR-33コンパクトのTTYは、穴落ちがはげしかった。紙テープを編集することがプログラムのEDITだったので、日に頁使ってる。SX-103の方がよっぽど便利だと思った。それ以来、ON-LINE Debugのとりこじゃ。

その時、ラインプリンタで、Sinカーブ書くことも習った。数字しか扱かえないものが、2次元図形を描くというのは、画期的だった。すぐに、SX-103でやったが、Back-Feedしないので、かなり頭を使った。(バイオリズムは、なかなか難しい)

教訓3。プリンタは、Back-Feedしない物を、使う。

その頃は、もうSX-103のプログラミングに自精通して、不自由なくなんでもできた。そして、『計算機は、しょせん、一から十まで教えておいたことしかできない。全然人工知能なんか、とは、違うもんだ。つまらんよ。』と思い始めていた。しかし、逆に心のどこかで、『いや、もっとちがう何かがあるはずだ』と、いつも思っていた。そんなある日、紀伊国屋の計算機の処で、『人工知能』なる本を立ち読みし、これが、追い求めていたものだと思い、大いに感動した。

『人工知能』産報出版、スレイグル著。

まあ、本の中の末日、名人に読んでもらうとして、このHeuristic Programming と、Mini-Max は すごいと思い、どうしても、プログラムしてみたくなった。そこで、SX-10の能力を考え、「3目並べ(Tic Tac Toe)」と、いうのに、ターゲットをしぼったのだった。

3目並べの、初版は、確か、1977年の学祭前には動いていたので、9月か10月には、動かしていた。そのプログラムたるや、データーメモリ目ぎっしりで、プログラムは、497stepであった。この497stepは血のにじむおもいで、約縮あたもので、少しでも、兼用できそうなものは、全て、兼用して、死ぬ程苦労したものだった。一番そーぜつなのは、心臓部の思考ルーチンを、思考以外に、勝負の勝ち負け判定と、打つ所があるかの判定の3回も、CALLしていた事で、1手指すのに、40秒も、かかったほどだ。(後に もっと全体の流れが最適化できて、1手は1回で打て、プログラムは430step前後になった)

教訓4。プログラムは小さなメモリ上でつくる。

1977年の夏には、ASCIIが、創刊した。日本橋には(秋ごろ)ByteShopができ、APPLE IIを毎週のように見に行った。もちろん、部品買い出しのついでだが。PETやTRSはまだなかった。IMSAI、ALTAIR、POLY-88が、えらそぶってた。(ぼくは、TK-80を、かっちまったのでした。6月ごろだったかな?)

[写]のころ、自作のマシンを、世界中のどのマシンよりも使い勝手よくしてやろうと思い、また 標準Busに合わせて作るということを目標に、日夜思いをめぐらしていた。(MONITOR, O.S.もつくるのだ。)
新しいマシンが出れば、カタログをすぐ取り寄せ、どんなもんか、考えた。BASICのインタプリタ等も、色々と、言語仕様を考え、Very Tiny Languageと言われる物の、Syntaxも色々考えた。(もちろん、ALGOL, Pascalもやった)

教訓5。なぜその機能があるか、を考え、なければ、どうするかを考える。

[画]とは、高3になったのであまり、遊ばず、浪人したので遊べず、大学に入ったので、真面目一方、で、現在。
その間、昔に MB-6880, M2-80K, PC-8001…等パーコンだらけ。しかし、やっぱり、修業をしたい人は、わざと、不便な状態に身を置いて、教訓1ヶ月をやってみると、コンピュータが1ヶ倍楽しく、いや1ヶ倍苦しくなるかもね。

(Jun/19/1983 Takeo)

親愛なるHacker予備軍の君へ。

どくだん＋へんけん
＝Takeoちゃんのすいせん本（青春のおもいで…）

1. Palo Alto Tiny BASIC Source List
（まいこんやる奴は、読まなあかん）
2.『人工知能』スレイグル
（天然痴脳のかはどーぞ）
3.『つくるコンピューター』（CQ出版、つくるシリーズNo.2）
（どう先、絶版らしいが、びんぼー人のバイブル）
4.『林檎物語』（田淵由美子）
（永遠の名作。。。。）
5.『コンパイラのうちとそと』（共立出版）
（コンパイラなんか簡単と思わせてくれる）

◎まいこん少年が毎号目を通す本（買わんでええ）

DDJ, bit ←Softの石頭派
インターフェイス ←Hardの石頭派

ASCII, I/O, Byte, Interface Age ←いかがわし－。

◎まいこん少年が知らなあかん言語

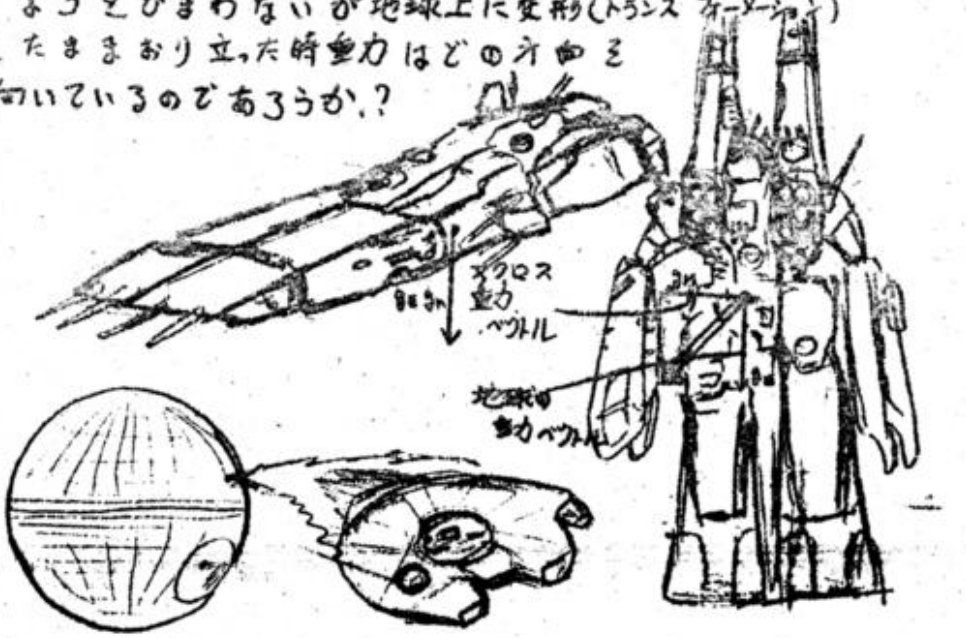
アルゴリズム記述によく使われる
- 8080 Machine Code & Assembler（絶対必要）
- ALGOL 60（これぐらいは常識）
- LISP（LISTも処理できなければ…）
- PROLOG（知ったかぶりするkは…）

※P.S. 2001年 岸田剛

ある雑誌を読んでいて思った事だけど、スターウォーズに出てくるデススターの重力はどの方向に向いているのであろうか？

あの丸い形から考えると中心方向にはたらいていると考えられるけれど、『新たなる希望』や『ジェダイの復讐』でファルコン号は、星に垂直にはいっているのである。あれはいったいどうなっているのであろうか。

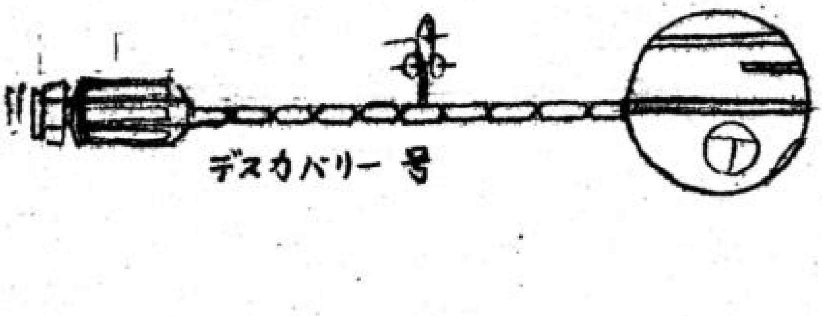
重力の問題はどSFの世界（特にアニメーション）の世界で無視されているものはないのではなかろうか超時空要塞マクロスでは宇宙空間でいくら変形しようとかまわないが地球上に変形（トランスフォーメーション）したままおり立った時重力はどの方向を向いているのであろうか？



（※ パーフェクト ソルジャー の事ではありません 近伸の事です）

回転していたり　無重力の小さな宇宙船ではゆかに接着テープがついていたりなかなかよくきえられていた。けれどあの作品中で一番現実とかけはなれていたのは食事である。あの作品中に出てくる色とりどりのねり菓子のようなわけのわからない食べ物である。昔なつかしいアポロ時代の歯みがきこチューブの食事が影響しているのだろうけれど現在でも電子レンジに入れて食べる地上とほとんど変わらない食事をしているというのにどうして未来になってあんなきみょうなものを食べなければいけないのであろうか理解に苦しむ。

　ところであのデスカバリー号の中で超大型のコンピューターのHAL9000というのが出てくるが（ちなみにHALの由来はH→I・A→B・L→M・つまりIBM（International Business Machines）である）古来の話ではあのコンピューターは狂ったと言われているがぼくはそう思わない。事の始まりは故しょうしてもいない通信システムを故しょうしているとHALが言った事から始まる。そこからHALは船内の人たちを殺していくのだ。そこでぼくが考えるのは、HALは進化したのではないかという事だ。いくらコンピューターといえども何兆回かに1回や2回はミスをおかす事があるであろうがそこでHALは自分のミスを隠そうとして自分が破壊されようとしているのを知ると人間を殺したかったのである。しかしそこにはHALの誇りと死の恐怖というものがあったの



デスカバリー号

だ、いくら超大型コンピューターといえども人間を殺してまでも自己を守ることがプログラムされているはずがないそこで考えられるのはHAL自身が自分という意志を持ったという事である。自己の意志を持ったHALはある意味では生物と呼べるのではないであろうか。これこそ第6世代のコンピューターではないであろうか、そしてこのコンピューターが出る時には人類はコンピューターにとって変わられるかもしれない。ナンチャッテ!

[なお P.S.(追伸)と本編があるような書き方をしましたがスペースがないのでカットします]

by Takeshi kishida

趣味の欄(先輩ゴメンナサイ)

おわりに

Limaをまたしても出せたのは、大変に喜こばしいことであります。たくさんの原稿ありがとうございました。

昨年のこのシーズンには投稿者も少なくどうなることかと思っていましたが...

来年のみなさんのカツヤクに期待します。

よいお年をおむかえ下さい。

83年12月 Editor

いつになったら京都工芸繊維大学はMIT(Matsugasaki Institute of Technology)になるのでしょう?

KTTU
Computer
Club