AFL言語の概要

平成23年9月18日

菅野　淳

コンピュータでのイノベーション

- 日本でコンピュータ分野のイノベーションが見られない
- 日本初のアイテムは皆無
  メソドロジー・ツールは日本は遅れている
- 70年代から80年代は先端技術者が集まっていた
  第5世代コンピュータ、LISP、Prologなど
- 90年代からインターネットに着目した米国中心の発展
- 現在、車・家電を中心に多くの組み込みシステム応用が生まれている

- 20年前の技術で実装している
- 相変わらずC言語を使用し、従来のCPUコアをベースにしたマルチコア技術で実装
- 安い労働力コストで海外展開
- 研究者・開発者のモチベーションが低い

新しいマシン語の提案

- **機械語のレベルを100倍にする**
  - 機械語の単位を1ビットから1バイトへ128倍の情報量
  - これからは機械語でプログラムする
  - 開発ツールも新機械語のためのツールとなる

- **プログラムの開発ツール・手法も大変革**
  - 上流から下流へのステップ、開発からデバックのステップが混在する

- **これは新機械語（インタープリタ）が可能とする**
  - CPU環境・ネットワーク環境を言語レベルで吸収する
  - AFLによる新言語の提案

新しいマシン語の開発提案

- **リアルタイムOSと統合したAFLシステムを開発する**
  OSの利用では、特殊なソフト構造とタイミングを理解した上で、複雑な手順で利用しなければならないが、プログラム言語機能とリンクをさせて単純明快にする。
- **マルチコアCPUに対応したAFLシステムを開発する**
  解釈、ソース再構成、実行、メモリ管理、ガーベージコレクションの各ステップを内部コアで並列に実行する。
- **外部機能、分散CPUの組み合わせを言語レベルで利用できる**
  各種ツールの利用をAFL演算子に取り込む。
  既存資産の再利用のために、この資産の呼び出しをAFL演算子に取り込む
- **その後は自然言語プログラミング**
  現在プログラミング言語が英語主体の特殊構文となっているのを、日本語や中国語の自然言語で記述する。

AFL言語とは

・AFLの開発経過
1976年「ノイマン型高級言語」情報処理学会
1977年「高級言語AFLとその処理システム」日経エレクトロニクス
1983年「日本語AFL」日経コンピュータ

・AFLはインタプリタ言語であり、プログラム容量は百キロバイト、処理時間は1ステップ当たり1μ秒の速度であり、携帯端末等の組み込みソフトに適する。

・応用プログラムの記述ステップ数は2桁程度は減少する。

AFL言語の基本実行構造

個別応用ソフト
AFL システム ノイマン型高級言語:プログラムとデータの区別をしなく、動的なプログラムの構築が容易
組み込み OS （Linux, iTRON）、汎用OS（Windows）
ハードウェア（CPU,メモリ、I/O）

記述プログラム言語　　C++
実行モジュール容量　　120KB（2011/9現在）
実行時間
n=1,s0=[n=n+1,<s<n.eq.2000000>>],s1=[],<s0>, の計算　1.2μ秒／サイクル
実行環境　　Visual studio Ver6.0

AFL言語に向いた応用分野

- 文字列処理・文書処理
  電子政府向け（XML解釈実行機能）
  XMLデータベース、XML検索

- 組み込みシステム
  画像処理：ビット処理（I/O処理）、画像処理機能
  画像改質、デジタルコード認識など携帯電話応用

- ユビキタス環境：自立的通信機能
  インタープリタ機能・プログラムの自動生成機能を生かした
  モバイルエージェント

- 日本語、中国語、英語の自然言語
  自然言語のコンパイラをAFLで記述
  （日本語プラグラム言語で1000～2000ステップ）

AFL処理系概要

AFLはソースプログラムを逐次解釈実行するインタープリタ言語である。
演算子は2項演算子から構成され、以下の順に実行される。

1　直接定数演算子
2　間接定数演算子
3　かっこ演算子
4　掛け算、割り算演算子
5　論理演算子、記号演算子、加算、減算演算子
6　代入演算子

AFLの高速化

高速処理の理由

- プログラムの構文が2項演算子を基本にした単純構造である
- データ型を文字型のみに限定している
- 演算子、演算データの参照をハッシュ演算を利用して高速に行っている
- 処理系が百KBと小さく、ハード化（DSP等）が容易である
- 基本演算子の実行時間は1μ秒程度

πの計算時間（実行パソコン　コアi5　2．3GHz ）

- 10000桁　16秒
- 30000桁　4分42秒

AFL言語によるプログラム例1

a=1,b=2,c=3,d=4,e=5,
y=(a+b)*c+d.gt.e.cpls.6,
を実行すると、以下の順序で演算が実行される。

- y=(a+b)*c+d.gt.e.cpls.6,
- y=3*c+d.gt.e.cpls.6,
- y=9+d.gt.e.cpls.6,
- y=13.gt.e.cpls.6,
- y=1.cpls.6,
- y=16,

結果として、yという名前のところに16というデータが格納される。

AFL言語によるプログラム例2

a=10,　d=[a],
b=[c=].cpls.[<d>],
<b>,
を実行すると、

c=<d>,

c=a,

というステートメントが実行され、最終的には
Cという名前のところに10というデータが格納される。

AFL言語によるプログラム例3

数値計算例（100の階乗計算）
AFLでは演算制度は自由に設定でき、下記の例は100*99*・・・*1を計算している

```
[a].sys.200,
s0=[d=d*n,n=n-1,<s<n.ceq.1>>],
s1=[d,],
d=1,n=100,
<s0>,
```

実行結果

```
93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000
```

AFL言語によるプログラム例4

1000桁のパイの計算：マチンの公式を用いて、πを計算する。

```
[A].sys.1000,
pai=0,n=1,data1=1/5,data2=1/239,
s0=[oldpai=pai, pai=pai+(16*data1-4*data2)/n, data1=-
data1/(5*5),
  data2=-data2/(239*239), n=n+2,
  <s<pai.eq.oldpai>>],
 s1=[[pai=]'pai,],
<s0>,
```

AFL言語による日本語プログラム例5

演算精度　は　1000　桁、
π=0、n=1、データ1=1/5、データ2=1/239、

［旧π　は　π、
　π　は　π＋(16*データ1－4*データ2)/n，　データ1　は　－データ1/(5*5)、
　データ2　は　－データ2/(239*239)，n　は　n＋2、］

を　π　が　旧π　と　等しい　まで　繰り返す、
π　を　印字する、

実行結果

pai=3.14159265358979323846264338327950288419716939937510582097494459230781640628
62089986280348253421170679821480865132823066470938446095505822317253594081284811
17450284102701938521105559644622948954930381964428810975665933446128475648233786
78316527120190914564856692346034861045432664821339360726024914127372458700660631
55881748815209209628292540917153643678925903600113305305488204665213841469519415
11609433057270365759591953092186117381932611793105118548074462379962749567351885
75272489122793818301194912983367336244065664308602139494639522473719070217986094
37027705392171762931767523846748184676694051320005681271452635608277857713427577
89609173637178721468440901224953430146549585371050792279689258923542019956112129
02196086403441815981362977477130996051870721134999999837297804995105973173281609
63185950244594553469083026425223082533446850352619311881710100031378387528865875
33208381420617177669147303598253490428755468731159562863882353787593751957781857
78053217122680661300192787661119590921642016 3

AFLの目指す将来イメージ

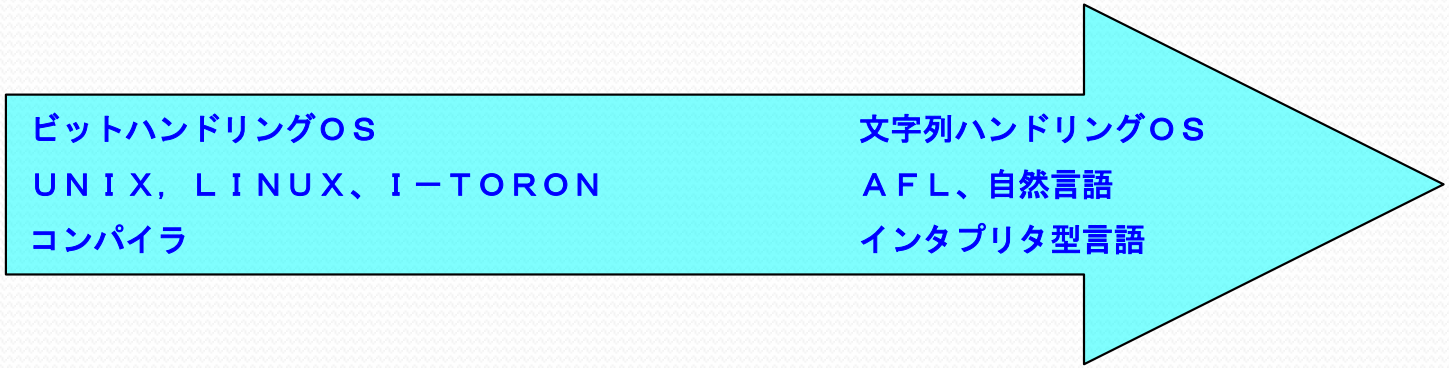
- 現在、コンピュータがビットを単位としたマシン語で稼動しているのを、バイト単位で稼動するという新しいアーキテクチャを提示する。ここではプログラミング言語が英語主体の特殊構文となっているのを、日本語や中国語の自然言語で記述できるようにする。

- さらにOSの利用に関しても、特殊なソフト構造とタイミングを理解した上で、複雑な手順で利用しなければならないのを、プログラム言語機能とリンクをさせて単純明快にする。

- この結果、ユビキタス社会のシステム構築における効率が飛躍的に向上することが可能となる。

ユビキタス社会の実現に向けて

現在	壁	将来
携帯電話 カーナビ 情報家電 いずれも数メガバイト単位の巨大なプログラム	プログラム量の爆発 ソフト開発空洞化 心臓部ブラックボックス化 トラブル増加 社会不安	情報家電 高度ＩＴＳ ＩＣカード マイクロＩＤチップ 個人認証 安全な電子化社会 （通信インフラ） ＦＴＴＨ 高速無線ＬＡＮ 赤外線



寄稿

パーソナル・コンピュータで稼働する日本語プログラミング言語の開発

自然言語の利用でマン・マシン・インタフェースを改善する「日本語 AFL」

上田 謙一　松下技研研究開発部主任技師　　菅野 淳　松下技研研究開発部技師　　本田 邦夫　松下技研研究開発部

●データやコマンドを日本語化したのではなく，自然な日本文で記述できる本格的日本語プログラミング言語「日本語 AFL」を開発し，試験稼働させている。

●一般の人へのなじみやすさという特徴を生かし，パーソナル・コンピュータ上で動かしているが，実用性は高い。日本語 AFL のコンパイラも日本語 AFL で記述した。

●算術演算，論理演算，ファイル処理などだけでなく記号処理も強力で人工知能に適している。

半導体技術の進歩に伴い，パーソナル・コンピュータの性能は大幅に向上し，漢字かなまじりの日本語情報処理もパーソナル・コンピュータで可能となってきている。しかし，これまでの日本語情報処理は在庫管理や給与計算などにおける品名や氏名などのデータを漢字かなまじりの日本語で表現することにとどまっていた。最近になって，この日本語による情報処理をプログラミングの分野にまで応用しようとする動きがでてきた。

これまでに開発されたプログラミング言語は BASIC, FORTRAN, COBOL などだが，これらは英語の構文が基礎となっている。欧米人にとっては親しみやすいものでも，日本人にとってはなじみにくく，これらのプログラミング言語を扱うことからコンピュータは難しいものと感じさせてしまうことが多かった。

このため，パーソナル・コンピュータが安価で入手しやすくなり，各家庭で利用できるにもかかわらず，普及が，いま一歩であった。そこで，パーソナル・コンピュータをだれでも使いこなせるようにするため，日本語でプログラムする言語の開発が期待されていた。

これまでに，日本語プログラミング言語として，いくつか開発されているが，これらは COBOL 言語を基礎とし，COBOL の文法に縛られた言語となっており，自然な日本語とほど遠い欠点があった。このような理由から筆者らは「日本語 AFL（A Fundamental Language）」と呼ぶ新しい日本語プログラミング言語を開発した。現在，当社でこの試験システムが順調に動いている。

自然な記述

図 1 は日本語 AFL による簡単なプログラム例である。最初のステートメントで，面積の算出式が「縦＊横」と定義される。②のステートメントで縦と横の値が決まり，③のステートメントで面積が計算される。計算結果の 5000 は「結果」という名前がついて保持される。

図 1 ●日本語 AFL プログラム例①
␣は空白を示す。

```
ステップ番号
① 面積␣は␣「縦＊横」.
② 縦＝100，横＝50.
③ 面積␣を␣計算する。
```

図 2 の例についても読者は容易にこのプログラムが何をしようとしているかがわかるであろう。①と②のステートメントはそれぞれ，直方体の体積と円柱の体積の算出式を定義している。③～⑤のステートメントで直方体の体積を計算し，⑥～⑧のステートメントで円柱の体積を計算している。

このように日本語 AFL では算出式の定義およびデータがプログラミングの実行段階で設定できるので，自由度の大きいプログラム表現が可能となっている。

日本語でできるプログラミング

パソコン普及に拍車

松下技研 初開発、近く市販

朝日 58.1.9(夕)

日本語でパソコンのプログラムが書けます―松下グループの松下技研（本社・川崎市、城阪俊吉社長）は六日、カナ漢字交じり文でプログラミングができる言語「日本語 AFL」を開発した、と発表した。日本語のプログラミング言語が開発され、市販されるのは初めて。

現在、パソコンに広く使われているプログラミング言語の「ベーシック」は英語の構文を基本にしたもので、日本人にはなじみにくい、といわれてきた。このため、せっかくパソコンを買っても使いこなせないという不満が消費者の間にくすぶり始めている。日本語でプログラムが書ければ、こうした障害を取り除くことができ、パソコンの普及に拍車がかかりそうだ。

松下技研が開発した新言語は、同社が五年前から独自に開発してきた AFL という言語を基本に、日本語化したもの。つまり日本語で書かれたプログラムを AFL に翻訳しながら、パソコンに命令を出す仕組みになっている。

日本語 AFL を構成している基本構文は「である、計算する」などの述語が約三十種類、「行目、回、大きい」といった修飾語が約三十種類、「は、を、に、から」などの助詞が約十五種類、その他に、名詞が自由に設定できるようになっている。これらの基本構文を使えば、通常のプログラムはほとんど日本語で書ける、という。また、日本語 AFL にはカナ漢字変換機能もあり、キーボードからカナで入力すると、自動的に二十八＊バイト以上必要。パソコンが理解できる0と1の信号に翻訳するのに、二段階の翻訳を経ることになっており、処理速度が速く、メモリーも大きなものが必要になるためだ。つまり、この言語は日本語から英語（AFL）へ、さらに0と1の機械語に翻訳しながら命令を実行させる仕組みになっている。

同社はこの新言語をパソコンメーカー、ソフトウエア販売会社向けに販売するが、価格は契約によって異なる。契約には一括契約と印税方式があり、一括契約では一億円前後になるほか、消費者はフロッピーディスクを買えるが、価格は未定。

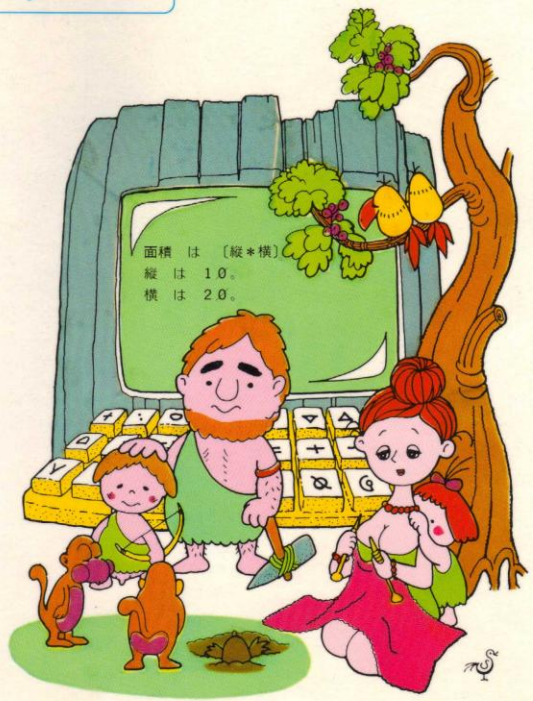
（1）日刊

新開発！日本語プログラミング言語

要説 日本語ＡＦＬ

那野比古 著

監修 松下技研(株)「日本語ＡＦＬ」開発グループ



東京 ブック

Getting personal computers to understand Japanese

TOKYO

Imagine the hassle of trying to programme your personal computer in a foreign language. Until late last month, all Japanese personal computer buffs had to do so. Not only did they have to master the intricacies of a computer language like Basic or Pascal—a problem in itself—but they also had to wrestle with the English used by such languages. They should welcome the development of a new personal computer programming language called AFL (short for a fundamental language). Developed by Matsushita's R&D arm in Tokyo, AFL's structure allows a programmer to write programmes in Japanese, without fussing about the complications of Basic or any other computer language.

AFL is designed to function rather like a set of grammatical rules, from which expressions, sentences, and thus entire computer programmes can be written. It lays down the rules for programming by defining what statements are permitted. The grammatical rules are, say Matsushita's boffins, independent of any particular human language: using AFL, it is theoretically possible to write programmes in tongues as diverse as Thai or Arabic. But so far AFL has been developed only to cope with Japanese.

The job of combining AFL's computer grammar with the Japanese language was carried out by a small Tokyo computer company, Kokusai Data Machine Systems. First, Kokusai devised a two-step method of putting Japanese programming instructions on to the screen of a personal computer. This is complicated by the fact that literate Japanese sentences are a mixture of two alphabets, katakana and kanji. (While katakana is a phonetic script, kanji characters are complex pictograms.) Instructions are first keyed into the personal computer in katakana, a script that can be handled by a normal keyboard. These katakana statements are then converted into literate Japanese sentences with the aid of an electronic dictionary which hunts out kanji synonyms for katakana words—where they exist. It is tricky: the dictionary holds over 2,000 kanji characters.

The next step is to translate the programme written in Japanese into the grammatical rules of AFL. This is accomplished by a compiler, which puts the programme into a form that the computer's hardware can digest.

Though it takes a personal computer user as many steps to write his own programme in a Japanese version of AFL as it does to write it in Basic, the ease of using AFL is immense. Mr Kohei Yaguda, Kakusai's general manager, claims that a complete beginner can master AFL in a few minutes. Some Japanese software specialists claim that a mere 1% of neophyte Japanese personal computer users can master languages like Basic.

Not that AFL has been developed just to cater for do-it-yourself Japanese beginners at programming. The hope is that, by being able to programme in Japanese, the Japanese will start to close the gap between them and the Americans in creating good, easy-to-use software packages. AFL is another good reason for IBM Japan's pursuit of a joint-venture with Matsushita.

Embryo sexing

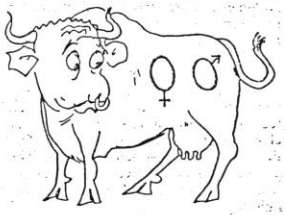
Vive la difference

An American company, Genetic Engineering Incorporated, claims to have developed the first reliable method of discovering the sex of cattle embryos. Maybe. Other scientists in the field are sceptical. And Genetic Engineering itself is being understandably secretive about how it has managed it; it is in the process of filing a patent.

Predicting accurately whether an embryo would produce a bull calf or a heifer would be a money-spinner. Embryo transfer, used to maximise the number of offspring a breeder can get from prize stock, is already big business. A top cow is given fertility drugs to induce her to produce a lot of eggs (up to 40), artificially inseminated with sperm from a high-class bull, the fertilised eggs flushed from her uterus and either introduced immediately into less valuable cows or frozen for use later.

Such embryos command fancy prices: several hundreds of pounds each in Britain. They would be roughly twice as valuable if the provider could guarantee whether any particular embryo was male or female. A customer with a beef herd does not want to be saddled with a cow—nor a dairy breeder with a bull. At present, each has only a 50% chance of getting what he wants.

In theory, sexing embryos should not



be too difficult. Scientists have known for years that a male embryo carries on its surface an antigen, or molecular identity tag, that female embryos lack. And an antibody that recognises and binds to this tell-tale antigen has been isolated. Tag the antibody with, eg, a fluorescent marker, expose the embryo to it and you should be able to tell what you have. If the antibody binds to the embryo, it is male. If it does not, it is female.

In mice, this approach works with an 80% reliability. But cattle embryos present a problem. The difficulty is that the number of antigen targets on the surface of an embryo varies over time. When the embryo comprises only eight cells—about three days after conception—a large number of antigens are expressed on its surface. Unfortunately, by the time that it can be removed from the cow—about six days after conception—this is no longer so. The upshot, say sceptics, is that an antibody sexing test is not reliable at this crucial stage.

Genetic Engineering insists that its test, using a pure antibody known as a monoclonal antibody, is reliable and the company is planning to build up a bank of frozen, sexed embryos. In support of its claim, it points to two calves born in