プログラミングに関する認知科学的研究(2)

松　原　康　夫

Cognitive Scientific Study of Programming (2)

by Yasuo MATSUBARA

Programming as a human activity can be investigated from various points of view. In this series of papers, the process of programming is considered from the cognitive scientific point of view.

In this paper, we consider several topics related to the teaching of programming. Firstly, we consider how to teach concepts of control structure and data structure to students. Secondly, the merits and the drawbacks of the language LOGO are considered, in the context of teaching programming to beginners. We propose several conditions required for a language alternative to LOGO. Finally, we consider the teaching of assembler and machine language, which is very important if students are to understand the mechanism of computers.

1．はじめに

この一連の論文は，人間のプログラミングの過程を認知科学的な立場から考察するものである。(1)で述べたように，プログラミング教育を通して認知科学的な知見を得ることだけでなく，認知科学を教育に応用することも本研究の目的である。

プログラミングを多くの人間が学ぶようになったのは比較的最近のことである。そのため，一般に通用する合理的な教え方は未だ確立されていない。しかし，パーソナルコンピュータの普及や，低学年においてコンピュータ教育を進めようとする国策により，プログラミングを学ぼうとする人口は今後急速に増加することが予想される。従って，プログラミングに関する教育方法論の確立されることが必要となるが，その際単なる経験に基づくものでなく，認知科学的な考察[2]に耐える合理的な方法論であることが最も理想的である。

とくに本論文では，プログラミング教育に関わるいくつかの問題について考察を行う。

一つは初心者教育に関して，日頃の教育経験から，変数や配列の教え方について述べる。もう一つは，プログラミングの初心者教育のためにどのような言語が望ましいかについて考察する。最後に，機械語やアセンブラ言語について何をどう教えるべきかについて筆者の考えを述べる。

2．初心者教育におけるデータ構造

プログラムを捉える際に，制御構造とデータ構造に分けて考えることはよく行われることである。ここでもそれに従おう。最初にデータ構造について考察する。

2．1．変数の概念

初心者教育に於いて最初にぶつかるのが，変数をどうやって教えるかという問題である。

(1)で述べたように筆者は箱の比喩を使うことにしているが，このことは初心者教育に携わるかなり多くの人が行っているようである。

ここでは箱による比喩を使う場合に教えなければいけない変数の性質を，いわば公理として整理してみよう。

公理(1)　変数は丁度一つの値が入る箱である。

そして，変数の値とは，箱の中身のことである。

公理(2)　変数に値を入れると，それまでに入っていた値は消える。

公理(3)　変数の値を取り出しても，値は消えたり変わったりしない。

以上の公理から次のような定理が導かれる。

定理(1)　一度変数に入れた値は，次に別の値を入れない限り，何度でも同じ値を取り出すことができる。

これに加えて，代入文の性質を公理として記述すると次のようになる。

公理(4)　代入文を実行すると，左辺の変数に，そのときの右辺の式の値が入る。

公理(5)　代入文を実行すると，右辺の式の値は，そこに現れる変数のそのときまでの値を（つまり箱の中身）を取り出して作られる。

特別の場合の注意事項として，次の定理が導かれる。

定理(2)　代入文の左辺の変数が右辺にも現れている場合，右辺の式の値は，この代入文を実行する直前の値を使って作られる。

但し，以上の公理は変数や代入文の意味論的な性質を規定するものである。構文論は前もって規定されているものとする。

認知科学的には，物事を理解することは結局その中に内在する論理的な構造を把握することであると考えられている。従って以上のような公理で述べられている事項を，表面上どのような形で表すにせよ，十分に把握させることが，変数及び代入文を学ばせる上で必要であることになる。

2．2．配列の概念

基本的な単純変数の概念を把握したあとで，配列を教えることになる。プログラミングの初心者教育を担当した教師のほとんどが，配列を教えることの難しさを訴えるようである。とくに理科系の教師はベクトルやマトリクスとの類推で教えることが多いようである。

ところが，数学的な思考法に慣れていない文科系の学生にとっては，この教え方は適切ではない。この点については，前述したように，現実の世界に存在して誰でもがよく把握している概念との類推で教えるべきである。

その最もよい例は，アパートやマンションの構造である。これらは次のようないくつかの点で，配列と共通している。

1）各部屋に通常自然数の番号がついていること。

2）各部屋は何かを入れる入れ物であること。

3）1次元のものから始まって，実際上何次元のものでも考えられること。

人間は，まったく同一の論理構造を持つ問題でも，日常慣れている形式であれば柔軟に対応できるが，慣れていない形式であると問題を解決できない場合が多いことが，認知科学的に知られている。従って，以上のような類推を使うことは，単に論理的な構造を把握させるのに役立つだけでなく，日常的に慣れている形式で問題を考えさせることにより，問題を解き易くすることにもなる。

筆者は以上のような観点から，実際のBASICを使った初心者教育の場に於て，単にアパートやマンションとの類推で，配列とはこんなものである，と説明するだけでなく，アパートを題材とした以下のような一連の問題を学生に解かせてみた。

a．1～7号室のあるアパート（コーポ文教）に入居者を入れる。

プログラムとしては，文字列型の配列BUNKYOU$を宣言して DATA 文に書いた名前を格納し，その内容を印刷させる。

b．コーポ文教が2階建てのアパートであり，各階に1～8号室があるとする。a．と同様にして各部屋の住人の名前を入れた後，階数と番号をキーボードから入力してその部屋の住人の名前を印刷させる。これを，0が入力されるまで繰り返す。

c．コーポ文教はb．と同様とし，名前を同様に入れておく。そしてキーボードから名前を入力して，その名前の住人を捜す。見つかればその部屋の階数と番号を印刷する。見つからなければそのことを報告する。

d．同じコーポ文教の各部屋に，手紙が届くことを考える。DATA 文またはキーボードからの入力で，何階の何号室に手紙が届いたかを，数十回繰り返し指定する。0が入力されると，それまでに各部屋の住人にそれぞれ何通の手紙が届いたかを，印刷する。

プログラムでは，別の配列 TEGAMI を用意して，手紙が届く度に対応する配列要素に1を加えていく。

e．今度は，コーポ文教の各部屋の住人の貯金額を考える。配列 CHOKIN を宣言して，DATA 文で与えた貯金額を格納しておく。そして，最も大きい貯金額の住人とその部屋の位置を印刷させる。

f．e．と同様の状況で，このアパートの大家が貯金を奨励しているので，貯金額の大きい順に部屋を入れ換えることにした。入れ換える前と後の状況をそれぞれ印刷する。

ソーティングの方法としては，e．との関連で最大値法を用いる。

以上のような一連の問題は，配列の初歩的な使い方をほぼ尽くしている（高度な使い方はいくらでもある）。実際にやってみた印象としては，問題そのものを理解するという点では，よく理解しているようであった。そして，このように最初から2次元配列を使用することの困難は生じなかったようである。

このような教育方法の効果を定量的に把握することは，なかなか困難なことである。最終的には試験の点数が出るが，とくに配列の教え方による効果だけを他の要素から分離するためには，他の教え方をした対照群との比較が必要となるが，現実に比較を行うのは難しい。

3．初心者教育における制御構造

2．に続いてここでは，プログラムがどの様な順序で実行されるかを教える方法について考察する。

3．1．フローチャートについて

フローチャートがいつ頃から使われていたかについては知らないが，昭和45年頃筆者が FORTRAN を教えられたときは，プログラミングの一環として教えられた記憶がある。その後，ソフトウェア危機が叫ばれ，それを救う手段としてソフトウェア工学が台頭してくるとフローチャートの評判は落ちる一方であった。現在でもその状況は変わっていないであろう。

しかしながら，他方では現実の世の中で，フローチャートの果たしている役割は決して小さくはない。この矛盾は何処からくるものであろうか。筆者なりにこの問題を整理してみたい。

筆者の経験からいうと，始めてプログラミングを教えられたときには，フローチャートを描いてからコーディングをしなさい，と言われ，コーディングシートが実際に用意されていたのである。

しかしながら，少し慣れると FORTRAN のプログラムとフローチャートが余りにも一対一に対応しているために，わざわざフローチャートを書いてからコーディングすることが面倒くさくなってくる。デバッグをするのにも直接プログラムを修正してしまう。従っ

てレポートなどにプログラムを載せるときは，プログラムから逆にフローチャートを描いていた。

また，その頃のコンピュータ関係の本には，プログラムの説明をするために必ずフローチャートが載っていたものである。実際，よく描かれたフローチャートは確かにプログラムを直接読むより理解がし易かったのであるが，そのように描かれたフローチャートは大変少なかったようである。平面図形として描ききれずに，矢印に番号をつけて他の部分に飛んでいるため，全体の構造がとうてい把握し切れないものとなっていた。

最近の本で見かけるフローチャートはその頃に比べて大変見やすいものになってきている。大きい処理は一つのフローチャートで表さずに，階層的な表現をしている。つまりある程度まとまった高度な処理を一つの箱で表わすことによって一つのフローチャートをコンパクトなものとし，その部分は別のフローチャートで詳しく表現するのである。

こういった変化は，プログラムそのものの構造を見易くするという構造化プログラミングの考え方に影響されているのであろう。

また一方でプログラムコードそのものの構造が大変読み易い，いわゆる構造的な言語が普及してくると，そういった言語を使う場面では殆どフローチャートを使う必要が無くなってきたのである。逆に言えば，フローチャートを別に必要とするようなプログラミング言語は，少なくとも計算機科学者の間では無用のものとなってきたのである。

構造的な言語の大部分はALGOL系の言語であり，中でも教育用の言語としてはPASCALが普及してきた。少なくとも小規模のプログラムを記述するにはPASCALは大変よい言語である。アルゴリズムを記述する場合にPASCALを用いることが多くなってきている。実際には他の言語でインプリメントする場合でも，その仕様を記述するためにPASCALを用いることもある。

以上で述べたように，最近のプログラミング言語を使う場合にはフローチャートを使う必要性が減少してきていることは事実である。

ところが反面，現実の社会に於て，フローチャートの果たす役割は必ずしも減少していないようである。その理由として次のような事が考えられる。

1）現実の社会に於いては，構造的でない昔からの言語の使われる率が多いこと。実際，COBOLやFORTRANの使われる割合が非常に大きいであろう。また，初心者教育の場に於いては，BASICの使われることが未だ比較的多い。

これらの言語に於いては，制御の流れをプログラムから読み取ることが容易ではないので，フローチャートを使って制御の流れを視覚的に表現する必要がある。

2）フローチャートそのものが視覚的に理解し易い性質を持っていること。他に何の知識が無くても，フローをたどることによってプログラムが実行されることが直観的に理解できる。従って，プログラムの働きを非専門家に説明する場合や，特定の言語に依存しないアルゴリズムを記述する場合等に使われることが多い。また，初心者教育においては，PASCAL等のプログラミング言語の働きを教える場合にも使うことがある。

3．2．今後のフローチャートの位置づけ

以上で述べたように，現実の世界でフローチャートの占める位置は複雑である。では，教育の場では，どのように位置付けられるだろうか。

教育の場といってもいろいろあるが，大学に於ける理工系の情報専門の学科等では，フローチャートをわざわざ教えることはあまりないようである。しかし教えなくても自然に解ってくる，というところであろう。またプログラムの仕様を記述するための手段としては，自然言語を使ってアルゴリズム的な書き

方をさせる場合もあるだろうし，フローチャートの変形や，他の図形的な記述法を教える場合もあるだろう。

専門学校等では，教育の目的を即戦力として使える人員の養成に置く場合が多いと考えられる。プログラミング言語としてもCOBOLやFORTRAN等が多いことから，どうしても普通のフローチャートをきちんと教える必要がある。

実は，プログラムの性質を理論的に追求する場合でもオーソドックスなフローチャートが用いられるのである。これは，フローチャートが，特定の言語に依存しない手順を視覚的に解りやすく表現できるためと考えられる。

また，まったくプログラミング言語を知らない初心者にとっては，プログラムにおける制御の流れというものを直観的に把握させるために大変便利なものであると言える。

ここで，プログラムを自分で作る訓練をするためにフローチャートがふさわしいかどうかは別問題である。しかし，その前の段階として，プログラムが具体的にどのように動作するものであるかを正確に理解させる必要がある。そのための一つの道具としてフローチャートをもっと利用すべきかもしれない。

4．LOGOについて

初心者教育用の言語として，また認知科学的な考察から生まれた言語としてLOGOが有名である[3]。必ずしもコンピュータの入門教育用として設計されたわけではないが，今後コンピュータ教育が低学年にまで普及することを考えると，プログラミングの入門教育用言語として候補の一つに挙げられることはまちがいない。また，現在の日本の大学教育においても，高校までにほとんどコンピュータを知らずに来た学生に教えるための言語として使われる可能性が，全く無いとは言いきれない。

本章では，以上のような場面において，LOGOが真に適切な言語であるか考察してみたい。

4．1．LOGOの特性

LOGOは本来，子供（幼児）が，思考能力を発達させるのに役立つ，手がかりを与える目的でパパート（S. Papert）によって考え出されたものである。パパートは，ジャン・ピアジェと一緒に研究をしたこともあり，パパートは，MITの人口知能研究所でLOGOを考えたのである。

LOGOはこうした背景によるのか，プログラミング言語としてよりも，むしろ幼児のお絵描きツールといったイメージが強いようである。実際そのタートル・グラフィックの機能は，一部他の言語にも付加機能として取り入れられるなど，教育用の道具としてはある程度の市民権を得ているようでもある。しかしながら，LOGOはLISPと殆ど同じ言語であることはあまり知られていないようである。

パパートによれば，LOGOは子供が自由に遊びの道具として使うことによって，自然に思考を発達させることができるので，教師は不要ということになっている。この点については，とかく批判が多いようである。

実際にLOGOを使ってみればすぐに解ることだが，タートル・グラフィックを使って繰り返しや再帰を含む図形を描いている分には，さほどの困難を感じないが，これが変数を使った本格的なプログラミングに入ろうとすると極端に難しくなるのである。これが教師無しで，しかも忍耐できる妥当な時間内で子供に習得できるなどとはとても思えない。

また，タートル・グラフィックについても，ディスプレイ画面上の抽象的なタートルに子供が感情移入できるとも思えないのである。少なくとも，機械で作ってある，3次元世界のタートルでない限り子供が興味を示すとは思われないし，プログラムをキーボードから入力することも幼児には不自然と思われる。この点についてはいろいろと工夫がなされて

いるようである。

以下においては，プログラミングの入門教育用として，LOGOが適切であるかどうかについて考察したい。

LOGOが初心者用言語としてLISPより優れている点は，少なくとも表面上はプログラムが読み易いということである。つまりカッコのお化けと言う印象を与えないのである。そして制御構造が逐次実行の他は，場合分けと繰り返し，そして手続きの再帰呼び出しがあるだけである。繰り返しは，最も単純な繰り返しであり，何回繰り返すかを定数または変数で与えるだけである。

以上のように大変簡単な制御機能を持ち，タートル・グラフィックは再帰の概念をつかませるには面白い道具であると言える。ただ，プログラミング教育の入門用としては，次の二つの点が問題となるだろう。

一つは，基本となるデータ構造がリストであること。もう一つは，手続きを呼び出すときの引き数として変数を書くとき，変数そのものとその中身を区別して書かなければならないことである。これらは，言ってみれば母体がLISPであることからくる宿命のようなものかも知れない。現実に使われている言語の大部分において，基本となるデータ構造が配列であり，代入文の左辺以外に変数を書けばその中身を意味するという意味論を採用していることからすれば，プログラミング教育の入門用としてLOGOを用いることは適切ではないことになるのである。

しかしながら，LOGOの発想の原点には，自分がタートルに感情移入することによって，絵を描く手続きを考えることを容易にしようという考えがある。実際に大学生に入門教育を行っていて，教えられたことをやるだけでなく自分でプログラムを考えさせることの困難さを知っている筆者にとっては，大変重要なヒントであるように思われる。

4．2．LOGOを超えて

それでは，LOGOの発想を活かして，現代のプログラミング教育の入門用として使える言語のイメージを構成してみよう。

1）データを操作する基本的な文は代入文であること。

2）基本となるデータ構造は配列であり，代入文の左辺以外に変数を書いたときは，その中身を意味すること。

3）現代のグラフィックの技術を活用して，もっと感情移入しやすい環境を用意すること。

例えば，配列をアパートやマンションの形で表す事等が考えられる。

4）制御の流れそのものを視覚に訴えて理解し易いものとすること。LOGOにおいては子供がタートルに感情移入することを期待しているが，それならばタートルは制御の流れを追うべきである。テレビ・ゲームに子供(大学生を含む）が熱中するのを見ているとタートルをオートバイや自動車に置き換えるのもひとつの方法であろうと思う。

5）制御構造の世界とデータ構造の世界とがどのようにして相互作用するかが直観的に理解できるようにすること。現実に使われているプログラミング言語の大部分では，このように制御構造とデータ構造という，半ば独立した二つの世界が存在し，それらがインタラクションすることによって処理が行われているわけである。初心者教育に於てはこの構造を体得させることが大事である。

そのために，制御を追うタートルとは別にデータを移動させるタートルを用意する事などが考えられる。

いずれにしても，LOGOが最初に考えられて（1967）から20年の歳月が流れているのである。その間にコンピュータは急速な発展を遂げ，個人の所有するパーソナル・コンピュータは20年前の大型計算機をはかるに凌ぐ性能と容量を持つに到っている。このことを考えれば，LOGOの考え方を継承しつつも

LOGOを超える，現代にふさわしい入門教育用の言語が生まれることが期待されるのである。

5．アセンブラ教育

現代に於ては，システム記述に使える言語が種々開発され，機械語をプログラムする機会は，比較的減少している。しかし，今後如何なる言語が開発されようとも，フォン・ノイマン方式の計算機アーキテクチャが支配的であり続ける限り，機械語やアセンブラによるプログラミングの機会は無くなることは有り得ないと言ってよい。本章では，機械語や，アセンブラに関する教育をどのようにしていくべきかについて筆者の考えを述べる。

5．1．アセンブラ教育の位置づけ

学校教育の中でアセンブラまたは機械語教育を考えるならば，少なくともここ当分の間は大学等の高等教育機関に限定してよいであろう。では，大学のしかも情報教育に重点を置く学部または学科においてどの程度まで教える必要があるのだろうか。

結論から言うと，全く触れないで済ますことはできないということである。その必要性は二つの事項からくる。一つは，現実の社会において，プログラミングに携わる以上は，どうしてもこれらの言語に接触する機会があるだけでなく，実際にプログラミングする場合があることである。もう一つは，コンピュータの動作原理を理解するためには，どうしても計算機が機械語のプログラムをどのように実行するかを知っておく必要があることである。

前者については，最近いろいろなシステム記述言語が発達してきたため，ほとんどのことがその言語の範囲内で記述できるようになってはきたが，それから外れる機能を要求される場合には機械語ないしはアセンブラでサブルーチンを書いてリンクする必要が生じるのである。また，別の高級言語を使ってサブルーチンを書く場合でも，リンクのためには主記憶やレジスタ等の概念を理解しておく必要がある。

また，後者については，現代の大部分の言語は，ソース・プログラムをコンパイルしてオブジェクト・プログラムを生成し，それをリンカまたはリンケージ・ローダが実行形式プログラムに変換する方式をとっているが，何故このような方式が採られるのかと言う理由は，アセンブラや機械語の知識が無ければ理解できない事柄である。

この他にもこれらに関する知識がなければ理解できない事柄はたくさんあるのである。特にオペレーティング・システムに関する事柄を理解するためには，どうしてもこれらの知識が必要である。例えば，割り込みを理解しなければI/Oの処理ができず，タイム・シェアリングやマルチ・プログラミングの原理が全く理解できないのである。

それでは次に，アセンブラや機械語に関する教育をどのようにして行うべきかについて考察する。

まず，第一に考える必要のあることはどのような計算機のアセンブラを教えるかである。いわゆるホストマシンから，ミニコンピュータそしてパーソナル・コンピュータまで考えられるし，1ワードの語長が32ビットのものから16ビット，8ビットそして4ビットのものまである。

筆者の経験では，語長は少なくとも16ビット以上のものでなければならない。8ビットでは直接にアクセスできるアドレス空間が狭いために，プログラミング上の本質的でない事柄のために，多くの注意を払う必要がある。また，ホスト・マシンのアセンブラは，余りにも機能が多すぎて，基本的な事柄だけを教えるのには不向きである。とくに割り込み等を教えることは，オペレーティング・システムの管理下でTSSを使っている限り不可能となる。

筆者は，10年ほど前に富士通のFACOM U-200というミニコンを使ってアセンブラの教育を行った経験がある。この機械は16ビット・マシンであり，命令の数が余り多くなくて教え易い機械であった。そして高沢助教授の開発した常駐型のアセンブラを使うことができたが，このシステムはオペレーティング・システムを一切使わず，紙テープから直接IPLでコアにロードされるものであった。OSを使っていないため，とくに割り込みの実習教育を行うのに適していたのである。筆者は，タイマの割り込みを利用してタイムシェアリング・システムの実験を学生に行わせたのである。

ところで，最近急速に普及しているパーソナル・コンピュータは，基本的に16ビットのものが多い。しかし，アドレス空間が拡張してあり，命令の数も多いので，少なくともアセンブラの入門に向いているとは言えないだろう。また，MS-DOS等のオペレーティング・システムのもとで使うことになるので，割り込みの本格的な実験を行うことは難しい。

やはり，初心者教育には命令数を限定して，基本的な事項だけを学べるような機械を設定するべきである。その点で，通産省の実施する情報処理技術者試験で使用される仮想コンピュータが，従来の8ビット・マシンであるCOMP-XとCAP-Xから，昭和62年度より16ビットのCOMETとCASLに変更された[4]-[6]のは歓迎するべきである。

この機械は，2ワード命令を基本としており，2ワード目でアドレスを指定するために全てのアドレス空間が直接アクセスできる。そしてレジスタまわりの命令と，オーソドックスなロード命令及びストア命令が用意されている。汎用レジスタはGR0からGR4までがあり，GR4はスタック・ポインタを兼ねている。このスタックのおかげでサブルーチンが容易に組めるようになっている。アセンブラ教育も初歩的な教育はこの程度から始めるのが適当と考えられる。

しかし教育上の立場から言えば，割り込みの概念の無いことが惜しまれる。

5．2．CASLアセンブラとCOMETシミュレータ

筆者は，アセンブラ及び機械語の教育に使い得るだけでなく，そのソース・プログラムをシステム・プログラム論の教材としても使うことを意図して，MS-DOS上で動作するCASLアセンブラとCOMETシミュレータを作成したので，これについて述べる。

作成に当たってとくに留意した点は次の二点である。

1）オブジェクト・プログラムつまり相対形式プログラムの構造が理解し易いように，バイナリー・ファイルではなくアスキー・ファイルとする。

2）割り込みの実習ができるように，CASLとCOMETを拡張する。

割り込みの実習を行うには，このようなシミュレータを用いることが最もよい方法であると考えられる。

COMETの拡張した仕様は次のとおりである。

主記憶は，0番地から3FFF番地までの16Kワードだけ実装されているものとする。従って，COMETシミュレータがオブジェクト・プログラムを実行形式プログラムに変換しながらロードしたあと，スタック・ポインタを兼ねるGR4の内容はシステムによって十六進の4000に設定されている。

CPUの実行状態を表すレジスタとしてステータス・レジスタSTRを用意した。これは16ビットの長さをもち，その14，15ビットがフラグ・レジスタFRの役割を果たす。このSTRとプログラム・カウンタを合わせてプログラム・ステータス・ワード，PSWと呼ぶ。STRのビット0はWAITビットであり，これが1のときCPUは待ち状態となる。STRのビット7が1のときは割り込み可能であり，

0のときは禁止状態である。

CPUはプログラム実行中にエラーを発見すると,STRのビット1を立てて待状態に入る。またエラーの種類を表すために，メモリの実装されていないアドレスにアクセスしたときはビット2を，レジスタ番号が4を超えたときはビット3を，インデクス・レジスタの番号が4を超えたときはビット4をそれぞれ立てる。

STRはFFF0というアドレスでアクセスできる。

また，タイマのために，TIME 0とTIME 1という二つのポートを用意する。TIME 0のビット0が1のときはタイマが割り込み可の状態であり，0のときは割り込み禁止の状態である。TIME 1はカウンタであり，プログラムによって数を書き込むと，1命令実行する毎に1ずつ減少していく。そして1から0になったときに割り込み要求を出すのである。TIME 0とTIME 1はそれぞれF000, F001番地に割り付けられている。拡張したCOMETのメモリ・マップを表1に示す。

CASLとCOMETでは，具体的な命令コードを定めていないので，これらのシステムの作成者は自分で適当にコードを定めなければならない。本システムで使用した命令コードを表2に示す。このコードを決定するに当たって，文献(5)を参考とした。

VECT以下の命令は，割り込みの実習ができるようにするために，CASLに拡張命令として付加したものである。

VECTは，オペランドで示したアドレスを，割り込み時に使用する新しいPSWの内容を用意しておくアドレスとしてCPUに対して宣言する命令である。

SVECTは割り込み時に，旧いPSWの内容を退避するアドレスをCPUに対して宣言する命令である。

INT 0は，CPUを割り込み禁止の状態にするものであり，INT 1は割り込みの可の状態にするものである。

表1 拡張COMETのメモリ・マップ

0000～3FFF	メイン・メモリ	
FFF0	STR	
F000	TIME 0	タイマ
F001	TIME 1	

表2 命令コード

MNEMONIC	コード	長さ	MNEMONIC	コード	長さ
EXIT	FF00	1	JPZ	60	2
IN	FF01	3	JMI	61	2
OUT	FF02	3	JNZ	62	2
			JZE	63	2
LD	10	2	JMP	64	2
ST	11	2	PUSH	70	2
LEA	12	2	POP	71	2
ADD	20	2	CALL	80	2
SUB	21	2	RET	81	2
AND	30	2			
OR	31	2	VECT	FC	2
EOR	32	2	SVECT	FD	2
CPA	40	2	LPSW	FE	2
CPL	41	2	INT 0	FB00	1
SLA	50	2	INT 1	FB01	1
SRA	51	2			
SLL	52	2			
SRL	53	2			

LPSWは，オペランドで示された番地からPSWの内容をロードする命令である。

例として,図5.1にDCで定義した16進数を，サブルーチンで文字列に変換して出力するプログラムを示す。また，それをCASLアセンブラにかけて出てくるソース・リストとオブジェクト・プログラムをそれぞれ図5.2，図5.3に示す。

このオブジェクト・プログラムをCOMETシミュレータでロードして，そのメモリ・ダンプをとり，実行させた様子を図5.4に示す。

これらのシステムは，MS-Cコンパイラを用いて開発した。そのソースリストを付録として掲載しておく。

```
 1:   ; main routine
 2: MAIN START
 3:   LD GR0,C
 4:   LEA GR2,BUF
 5:   CALL HEXAP
 6:   OUT BUF,NUM
 7:   EXIT
 8: BUF DS 4
 9: NUM DC 4
10: C DC #FE02
11:  END
12:  ;GR0 out by hexadecimal
13: HEXAP START
14:  PUSH 0,GR1
15:  PUSH 0,GR3
16:  LD GR3,C4
17: LOOP ST GR0,W
18:  LD GR1,MASK
19:  AND GR1,W
20:  SRL GR1,12
21:  CPL GR1,C10
22:  JPZ AL1
23:  ADD GR1,AD1
24:  JMP N1
25: AL1 ADD GR1,AD2
26: N1 ST GR1,0,GR2
27:  ADD GR2,C1
28:  SLL GR0,4
29:  SUB GR3,C1
30:  JNZ LOOP
31:  POP GR3
32:  POP GR1
33:  RET
34: MASK DC #F000
35: W DS 1
36: C1 DC 1
37: C10 DC 10
38: C4 DC 4
39: AD1 DC #0030
40: AD2 DC #0037
41:   END
```

図　5．1

```
CASL-E  Assembler  System written by Y.Matsubara (27 Sep. 1987).
          1:   ; main routine

0000      1: MAIN   START
0000      2:        LD      GR0,C
0002      3:        LEA     GR2,BUF
0004      4:        CALL    HEXAP
0006      5:        OUT     BUF,NUM
0009      6:        EXIT
000A      7: BUF    DS      4
000E      8: NUM    DC      4
000F      9: C      DC      #FE02
0010     10:        END
         11:  ;GR0 out by hexadecimal

0000      1: HEXAP  START
0000      2:        PUSH    0,GR1
0002      3:        PUSH    0,GR3
0004      4:        LD      GR3,C4
0006      5: LOOP   ST      GR0,W
0008      6:        LD      GR1,MASK
000A      7:        AND     GR1,W
000C      8:        SRL     GR1,12
000E      9:        CPL     GR1,C10
0010     10:        JPZ     AL1
0012     11:        ADD     GR1,AD1
0014     12:        JMP     N1
0016     13: AL1    ADD     GR1,AD2
0018     14: N1     ST      GR1,0,GR2
001A     15:        ADD     GR2,C1
001C     16:        SLL     GR0,4
001E     17:        SUB     GR3,C1
0020     18:        JNZ     LOOP
0022     19:        POP     GR3
0024     20:        POP     GR1
0026     21:        RET
0028     22: MASK   DC      #F000
0029     23: W      DS      1
002A     24: C1     DC      1
002B     25: C10    DC      10
002C     26: C4     DC      4
002D     27: AD1    DC      #0030
002E     28: AD2    DC      #0037
002F     29:        END
```

図　5．2

```
SMAIN  :V0000:A1000:R000F:A1220:R000A:A8000:EHEXAP :AFF02:R000A
R000E:AFF00:T0004:A0004:AFE02:SHEXAP :V0000:A7001:A0000:A7003
A0000:A1030:R002C:A1100:R0029:A1010:R0028:A3010:R0029:A5310
A000C:A4110:R002B:A6000:R0016:A2010:R002D:A6400:R0018:A2010
R002E:A1112:A0000:A2020:R002A:A5200:A0004:A2130:R002A:A6200
R0006:A7130:A0000:A7110:A0000:A8100:A0000:AF000:T0001:A0001
A000A:A0004:A0030:A0037:
```

図　5．3

```
Comet Simulator written by Y.Matsubara( 25 Nov. 1987).

 Loading Object Programs.
ac=0000  svalue=0000
ac=0010  svalue=0000
--ofset=0000
--ofset=0010
Object program loading finished.  ac==003F
PC==0000
COM>M:0-3F
------- 0/8  1/9  2/A  3/B  4/C  5/D  6/E  7/F
M[0000] 1000 000F 1220 000A 8000 0010 FF02 000A
M[0008] 000E FF00 0000 0000 0000 0000 0004 FE02
M[0010] 7001 0000 7003 0000 1030 003C 1100 0039
M[0018] 1010 0038 3010 0039 5310 000C 4110 003B
M[0020] 6000 0026 2010 003D 6400 0028 2010 003E
M[0028] 1112 0000 2020 003A 5200 0004 2130 003A
M[0030] 6200 0016 7130 0000 7110 0000 8100 0000
M[0038] F000 0000 0001 000A 0004 0030 0037
COM>G0
:FE02
The waite bit was set. STR==8001  PC==000A
COM>
```

図 5．4

6．むすび

実際にプログラミングを教える立場にあって，如何なる方法を使えば学生が理解できるかは，常に脳裏にある問題である。そのための方法論を考えるに当たって，認知科学は，重要なヒントを与える源泉である。

プログラミングの入門教育において，配列を如何にして教えるかは，これに携わる教師に共通の悩みである。この際，学生がすでに頭の中に構築して日常生活の中で何気なく使っている思考の枠組みを利用することが，認知科学的に言って最も容易な教え方である。

筆者は実際の入門教育の場で，この考え方に立って一つの方法を実施してみて，その有効性を確認した。このことは逆に言えば，人間が新しい概念を修得するときには，それまでに修得した概念を利用しているのだということを裏書きするものでもある。

プログラミングを教えるに当たって経験する，多分これも共通の悩みは，いかにして学生に自発的に，ものを考えさせるかと言うことである。そのためには，学生の興味を引くような問題を用意することが重要と考えられる。そこで簡単なグラフィックの機能を使って絵を描かせたりすることになるのである。しかし，これを単なる表面的な興味に終わらせずに，プログラムを自分で考えることに繋いでゆく事は困難な問題である。

絵を描く機能を利用して，感情移入させることによってプログラミングを自然に修得させようとしたのがLOGOという言語である。これは，認知科学的な発想をもとにして考察されたものではあるが，現在のところ額面どうりの成功を納めているとはいい難い。その失敗の原因を探ることによって，本論文では現代のコンピュータ技術に見合った，今後の入門教育用のプログラミング言語のあるべき姿を提案した。

コンピュータ教育の中で，アセンブラ及び機械語の教育をどのように位置づけ，どのように行うかは避けて通ることのできない問題である。本論文では，この問題についての筆者の考えを述べるとともに，その具体的な答えの一つとして，筆者が開発したCASLアセンブラとCOMETシミュレータについて報

告した。これらは，オブジェクト・プログラムの役割が理解し易いことに留意し，またとくに割り込みの実験ができるように拡張したものである。割り込みを含むCPUの動作を初心者に理解させるために，視覚的なモデルを構成することは今後に残された課題とする。

今回は実際の教育の場に於て，如何にして理解させるかという方法論を中心として議論を行った。今後，課題を与えられた人間が如何なる過程でプログラミングを行っているのかについて考察を深めていきたい。その際，課題を記述する自然言語の果たす役割は非常に重要であるので，自然言語に関する研究も併せて行う予定である。

末筆ながら，本研究が昭和61年度文教大学情報学部共同研究費の補助を受けたことに謝意を表する。そして，ご支援いただいた広内哲夫先生に感謝いたします。

参考文献

(1)松原康夫：“プログラミングに関する認知科学的研究(1)”，情報研究第7号，PP. 96-104，(1986)．
(2)佐伯　胖編：“理解とは何か”，認知科学選書4，東京大学出版会（Nov. 1985)．
(3)祐安重夫：“LOGO人工知能へのアプローチ”，ラジオ技術社（1984)．
(4)昭和62年度情報処理技術試験案内書，情報処理技術者試験センター
(5) M. M. L. “はじめてのCASL”，工学社（1986)．
(6)原野秀永監修：“わかるアセンブラCOMET & CASL”，学習研究社（1987）

```
 1: #include <stdio.h>
 2: #define MAXT 1000
 3: FILE  *fp1,*fp2;
 4: struct lab{char label[7]; unsigned value,def;} ltable[MAXT];
 5: struct mt{char mcode[6];unsigned ocode,type;} mtable[21] =
 6:      {{"ADD  ",0X2000,1},{"AND. ",0X3000,1},{"CALL ",0X8000,2},
 7:       {"CPA  ",0X4000,1},{"CPL  ",0X4100,1},{"EOR  ",0X3200,1},
 8:       {"JMI  ",0X6100,2},{"JMP  ",0X6400,2},{"JNZ  ",0X6200,2},
 9:       {"JPZ  ",0X6000,2},{"JZE  ",0X6300,2},{"LD   ",0X1000,1},
10:       {"LEA  ",0X1200,1},{"OR   ",0X3100,1},{"PUSH ",0X7000,2},
11:       {"SLA  ",0X5000,1},{"SLL  ",0X5200,1},{"SRA  ",0X5100,1},
12:       {"SRL  ",0X5300,1},{"ST   ",0X1100,1},{"SUB  ",0X2100,1}
13:      };
14: int freearea=0,lstart=0,lend=0,pass,startf,ocnt=0;
15:     /*      freearea  :  indicates the top of the free area.       */
16:     /*      lstart:   indicates the beginning of local area.  */
17:     /*      lend   :   indicates the tail of local area.       */
18:     /*      ocnt : count the numnber of character in object file  */
19: /*****************MAIN*******MAIN*******MAIN********************/
20:
21: main(argc,argv)
22: int argc;
23: char *argv[];
24: { char ccc[13];
25:   FILE *fopen();
26: /*---------------------- file  name  ノ ショリ--------------------------*/
27: {int i,j,k;
28:   if (argc != 2) {printf("ヒキスウ ノ カズ ガ チガウ"); goto owari;} ;
29:   for(i=0;argv[1][i] != 0 && argv[1][i] !='.';i++);
30:   if(i==0){printf("file name error");goto owari;}
31:   if(argv[1][i]==0) {printf("bad extension"); goto owari; }
32:   i++; k=i;for(j=0;argv[1][i]!=0;j++)ccc[j]=argv[1][i++];
33:   ccc[j]=0;
34:   if(!issame(ccc,"cas") && !issame(ccc,"CAS"))
35:                                 {printf("bad extension"); goto owari;}
36:   leteq(ccc,argv[1]); leteq(&ccc[k],"OBJ");
37: }
38: /*-------------------------------------------------------------------*/
39:   {char gyo[73];
40:
41:   printf(" CASL-E  Assembler  System written by");
42:   printf(" Y.Matsubara (27 Sep. 1987).¥n");
43:
44:
45:   for(pass=1;pass<=2;pass++)
46:      { int gline=1,lline=1,level=0; unsigned ac=0;
47:        int locsw=0; unsigned locsave;
48:        fp1=fopen(argv[1],"r");
49:        if(pass==2){int i;fp2=fopen(ccc,"w");lend=0;
50:                /*   for(i=0;i<freearea;i++)
51:                     printf("#%s#%0x#¥n",ltable[i].label,ltable[i].value);
52:                */ }
53:       while (getgyo(gyo)!=0)
54: /*-------- 1  ｷﾞｮｳ  ノ  ショリ  ----------------------------------------*/
55:       {int i=0,j,inc=0,err=0,dflag=0,cflag=0,iop,loc;
56:          /* cflag==1 on coment line   */
57:          /* dflag==1 on double definition  */
58:        char label[7],olab[7],mnemonic[6],b[5];
59: /*  LABEL ﾗﾝ ノ ショリ */
60:       startf=0;
61:       leteq(label,"      ");
62:            if (isupper(gyo[0]))
63:               { for(i=0;i<=5 && (isupper(gyo[i]) || isdigit(gyo[i]));i++)
64:                                                    label[i]=gyo[i];
65:                 if(gyo[i] !=' ')
66:                   {if(gyo[i] != ';')err=1; else err=2;  goto linend;}
67:                 while (gyo[i]==' ')i++;
```

```
68:              }
69:            else
70:              { while (gyo[i]==' ')i++;
71:                if(gyo[i]==';'){cflag=1;goto linend;}
72:              }
73: /*  mnemonic - operand ｼｮﾘ  */
74:       {
75:        leteq(mnemonic,"      ");
76:        for (j=0;j<=4 && isupper(gyo[i]);j++) mnemonic[j]=gyo[i++];
77:        iop=i;  /* mnemonic code ﾉ ｵﾜﾘ ﾉｲﾁ ｦ ｵﾎﾞｴﾃ ｵｸ */
78:        if(gyo[i]!=' ' && gyo[i]!=';' && gyo[i]!='¥0'){err=3;goto linend;}
79:        while(gyo[i]==' ')i++;
80:        if ( issame(mnemonic,"DC    "))
81:          {int n;
82:            if(isdigit(gyo[i]) || gyo[i] == '-' || gyo[i] == '+')
83:              { if(!isdigit(gyo[i]) && !isdigit(gyo[i+1]))
84:                                        {err=4;printf("loc2"); goto linend;}
85:                n=todeci(gyo,&i);
86:                if(pass==2)  object(phexa(b,n),'A');  inc=1;
87:              }
88:            else if(gyo[i]=='#')
89:              {inc = 1 ;   i++;
90:               if(hexop(b,gyo,&i,pass)!=4){err=4 ; goto linend;}
91:               if(pass==2)object(b,'A');
92:              }
93:            else if(gyo[i]=='¥'')
94:               {char c[73]; unsigned n; i++;
95:                 for(j=0;gyo[i]!='¥'' && gyo[i]!='¥0';j++)c[j]=gyo[i++];
96:                 c[j]='¥0';/* printf("#j==%d#i==%d#%s#¥n",j,i,c); */
97:                 if(gyo[i]=='¥0'){err=5;goto linend;}
98:                 if(j==0){err=6; goto linend;}
99:                 i++;                                    inc=j;
100:                if(pass==2)
101:                  for(j=0;c[j]!='¥0';j++)
102:                    {n=c[j] & 0X00FF; object(phexa(b,n),'A'); }
103:              }
104:           else if(isupper(gyo[i]))   /* ｵﾍﾟﾗﾝﾄﾞ LABEL ﾉ ｼｮﾘ */
105:              {                                       inc=1;
106:               loc=labop(olab,gyo,&i);
107:               if(pass==2)
108:                   {if(ltable[loc].def)
109:                       object(phexa(b,ltable[loc].value),'R');
110:                     else
111:                       object(olab,'E');
112:                   }
113:              }
114:           else
115:              {   err=4;  goto linend; }
116:
117:           } /* DC ﾉ ｵﾜﾘ */
118:         else if(issame(mnemonic,"DS    "))
119:             {if(!(gyo[i]=='+' && isdigit(gyo[i+1])) && !isdigit(gyo[i]))
120:                                                    {err=4; goto linend;}
121:               inc=todeci(gyo,&i);
122:               if(pass==2)object(phexa(b,inc),'T');
123:              }
124:         else if(issame(mnemonic,"START"))
125:             {/* START   */ unsigned svalue=0;
126:              startf=1;
127:              if(level==1){err=8; goto linend;}
128:              level=1;
129:              if(pass==1)
130:                {leteq(ltable[freearea].label,label);
131:                 ltable[freearea].value=0;ltable[freearea++].def=10;
132:                 lstart=freearea; lend=freearea;
133:                }
134:              if(issame(label,"      ")){err=7; goto linend;}
```

```
135:              if(pass==2)
136:                {lstart=++lend;
137:                 while(ltable[lend].def!=10 && lend!=freearea) lend++;
138:                }
139:              leteq(olab,"      ");
140:              if(isupper(gyo[i]))
141:                 {loc=labop(olab,gyo,&i);
142:                  if(pass==2 && ltable[loc].def!=1){err=9; goto linend;}
143:                  svalue=ltable[loc].value;
144:                 }
145:               if(pass==2)
146:                {/*  start label & it's value  output  */
147:                 object(label,'S');
148:                 object(phexa(b,svalue),'V');
149:                 printf("¥n");
150:                }
151:               lline=1; ac=0;
152:             }/*  START ノ ｵﾜﾘ */
153:          else if(issame(mnemonic,"END  "))
154:             {if(level!=1){err=8;goto linend;}
155:              level=0;
156:              if(!issame(label,"      ")){err=1;goto linend;}
157:             }
158:          else if(issame(mnemonic,"INT  "))
159:             {char c;                                          inc=1;
160:              c=gyo[i++];
161:              if(c!='0' && c!='1'){err=4; goto linend;}
162:              if(pass==2)
163:                 {if(c=='0')object("FB00",'A');
164:                   else     object("FB01",'A');
165:                 }
166:             }
167:          else if(   issame(mnemonic,"LPSW ")
168:                  || issame(mnemonic,"VECT ")
169:                  || issame(mnemonic,"SVECT")
170:                 )
171:             {                                                 inc=2;
172:              if(pass==2)
173:                {     if(issame(mnemonic,"LPSW "))object("FE00",'A');
174:                 else if(issame(mnemonic,"VECT "))object("FC00",'A');
175:                 else object("FD00",'A');
176:                }
177:              if(gyo[i]=='#')
178:                {i++;
179:                 if(hexop(b,gyo,&i,pass)!=4){err=4; goto linend;}
180:                 if(pass==2)object(b,'A');
181:                }
182:              else if(isupper(gyo[i]))
183:                {loc=labop(olab,gyo,&i);
184:                 if(pass==2)
185:                   {if(ltable[loc].def)
186:                        object(phexa(b,ltable[loc].value),'R');
187:                    else
188:                      object(olab,'E');
189:                   }
190:                }
191:              else
192:                { err=4; goto linend;}
193:             } /*  LPSW , VECT , SVECT ノ  ｵﾜﾘ */
194:          else if(issame(mnemonic,"IN   ") || issame(mnemonic,"OUT  "))
195:             {                                                 inc=3;
196:              if(pass==2)
197:                {if(issame(mnemonic,"IN   "))object("FF01",'A');
198:                         else               object("FF02",'A');
199:                }
200:              if(!isupper(gyo[i])){err=4; goto linend;}
201:              loc=labop(olab,gyo,&i);
```

```
202:              if(pass==2)
203:                {if(ltable[loc].def) object(phexa(b,ltable[loc].value),'R');
204:                  else               object(olab,'E');
205:                }
206:              if(gyo[i++]!=',' || !isupper(gyo[i])) {err=4; goto linend;}
207:              loc=labop(olab,gyo,&i);
208:              if(pass==2)
209:                {if(ltable[loc].def) object(phexa(b,ltable[loc].value),'R');
210:                  else               object(olab,'E');
211:                }
212:
213:             }
214:          else if(issame(mnemonic,"EXIT "))
215:             {                                                    inc=1;
216:              if(pass==2) object("FF00",'A');
217:             }
218:          else if(issame(mnemonic,"POP  "))
219:             {int grnum=0;                                        inc=2;
220:              if(gyo[i++]!='G' || gyo[i++]!='R' || !isdigit(gyo[i]))
221:                                                   {err=4; goto linend;}
222:              grnum=gyo[i++]-'0';  if(grnum > 4){err=4; goto linend;}
223:              if(pass==2)
224:                { unsigned opc;       opc=0X7100 | (grnum << 4);
225:                 object(phexa(b,opc),'A'); object(phexa(b,0),'A');
226:                }
227:             }
228:          else if(issame(mnemonic,"RET  "))
229:             {                                                    inc=2;
230:              if(pass==2)
231:                {object("8100",'A');
232:                 object(phexa(b,0),'A');
233:                }
234:             }
235:          else  /*   Executive  Operations other than
236:            IN , OUT , EXIT ,  POP , RET,  LPSW ,VECT , SVECT ,INT */
237:             {int type,opc,grnum,xrnum,loc,ladr=1,n;              inc=2;
238:              type=mtsearch(&opc,mnemonic,mtable);
239:              if(type==-1){err=3;goto linend;}
240:             /*  operand ｶｲﾄｷ  */
241:              if(type==1)   /*     GR,adr[,GR]   */
242:                 {if(gyo[i++]!='G' || gyo[i++]!='R' || !isdigit(gyo[i]))
243:                                                   {err=94;goto linend;}
244:                  grnum=gyo[i++]-'0'; if(grnum>4){err=94 ; goto linend;}
245:                  if(gyo[i++]!=','){err=94; goto linend;}
246:                 }
247:                else /* type==2  */  grnum=0;
248:               /*  type == 1 or 2   */
249:              if(isupper(gyo[i]))loc=labop(olab,gyo,&i);
250:                else{ ladr=0; n=todeci(gyo,&i);   }
251:              if(gyo[i]==',')
252:                 {i++;
253:                  if(gyo[i++]!='G' || gyo[i++]!='R' || !isdigit(gyo[i]))
254:                                                   {err=14;goto linend;}
255:                  xrnum=gyo[i++]-'0';
256:                  if(xrnum >4 || xrnum==0){err=24; goto linend;}
257:                 }
258:               else /*  adr  only */     xrnum=0;
259:              if(pass==2)
260:                  { opc=opc | (grnum << 4) | xrnum ;
261:                    object(phexa(b,opc),'A');
262:                    if(ladr==1)
263:                      {if(ltable[loc].def)
264:                          object(phexa(b,ltable[loc].value),'R');
265:                       else
266:                          object(olab,'E');
267:                      }
268:                    else  object(phexa(b,n),'A');
```

```
269:                    }
270:                } /* Executive operations other than POP , RET  ノ  オワリ    */
271:        }  /* mnemonic - operand  オワリ  */
272: /*  キ゛ョウ  マヅ ノ ショリ  */
273:       if(level==0 && !issame(mnemonic,"END  "))
274:        {err=9;goto linend;}
275:         while(gyo[i]==' ')i++;
276:            if(gyo[i]!='¥0' && gyo[i]!=';'){ err=34;goto linend;}
277: /*  label ノ ショリ           */
278:          if(!issame(label,"      ") && startf==0)
279:           { int loc;
280:            loc=ltsearch(label);
281:            if(pass==1)
282:               {if(loc==-1)
283:                 {leteq(ltable[freearea].label,label);
284:                  ltable[freearea].value=ac;   ltable[freearea++].def=1;
285:                  lend=freearea;
286:                 }
287:                else if(ltable[loc].def==0)
288:                       { ltable[loc].value=ac;ltable[loc].def=1;}
289:               }
290:            else /* pass==2 */
291:              { if(ltable[loc].value!=ac) dflag=1;
292:              }   /* double definition */
293:         }
294: /*  source printing */
295:   linend:  if(pass==2)
296:           {
297:            if(err==0){if(cflag==1)printf("       ");
298:                       else if(dflag==1)printf(" d.def.");
299:                            else printf(" %s  ",phexa(b,ac));}
300:              else     printf(" err%2d ",err);
301:            printf("%5d: ",lline);
302:            if(err==0 && cflag==0)
303:                       printf("%s %s %s",label,mnemonic,&gyo[iop]);
304:              else     printf("%s",gyo);
305:            printf("¥n"); /* カイキ゛ョウ */
306:           }
307: /* --------      address counter increment  -----------------*/
308:             ac+=inc;
309: /*-----------  line number  increment   --------------------*/
310:           gline++;    lline++;
311:         }/*--- while (getgyo(gyo)!=0) ノ オワリ ----*/
312:       fclose(fp1);
313:       if(pass==2){fclose(fp2); if(level==1) printf(" ¥"END¥" expected¥n");}
314:
315:       }/*---- for(pass=1;pass<=2;pass++) ノ オワリ  ------*/
316:    }/*----  ltable   ト  mtable  ノ  オワリ ---------------------------*/
317: owari:;
318: }/*----------------main ノ オワリ-----------------------------------------*/
319: getgyo(gyo)
320: char gyo[];
321: {int i;char c;
322:    for (i=0;i<72 && (c=getc(fp1))!=EOF && c!='¥n';++i) gyo[i]=c;
323:    while(c!=EOF && c!='¥n')c=getc(fp1);
324:  gyo[i]='¥0' ;return(i);
325: }/*-------------getgyo ノ オワリ-------------------------------------*/
326: ltsearch(label)
327: char label[];
328: {int i;  /***ミツカラナイ  トキ      -1      ヲ  カエス***/
329: /*printf("#%s#ヲ サカ゛ス#lstart=%d#lend=%d#¥n",label,lstart,lend);*/
330:      for(i=lstart;i < lend && (!issame(ltable[i].label,label));i++)
331:     /* printf("#%d#%s#",i,ltable[i].label);printf("i=%d",i)*/;
332:     if(i == lend)  return(-1); else return(i);
333: }
334: isdigit(c)
335: char c;
```

```
336: {if(c>='0' && c<='9')return(1);
337:    else return(0);
338: }
339: isupper(c)
340: char c;
341: {if (c >= 'A' && c <= 'Z')return(1);
342:     else return(0);
343: }
344: issame(s1,s2)
345: char s1[],s2[];
346: {int i;
347:    for(i=0;s1[i]!=0 && s1[i]==s2[i];i++);
348: if(s1[i]==s2[i]) return 1 ;
349:         else     return 0 ;
350: }
351: leteq(s1,s2)
352: char s1[],s2[];
353: {int i;
354:   for(i=0;(s1[i]=s2[i])!=0;i++);
355:   return(i);
356: }
357: phexa(b,n)
358: char b[];unsigned n;
359: { int i,j; unsigned m;
360:    for(i=3;i>=0;i--)
361:      { m=n/16;j=n-m*16;
362:        b[i]=((j<10)? j+'0': j-10+'A');
363:        n=m;
364:      }
365:    b[4]='¥0';
366:    return &b[0];
367: }
368: todeci(gyo,pi)
369: char gyo[]; int *pi;
370: {unsigned n=0; int i,j;
371:  i=*pi; j=1;
372:  if(gyo[i]=='+')i++; else if(gyo[i]=='-'){j=-1;i++;}
373:  while(isdigit(gyo[i]))n=n*10+gyo[i++]-'0';
374:  *pi=i; if(j==-1)n=-n;
375:  return  n;
376: }
377: labop(olab,gyo,pi)
378: char gyo[],olab[]; int *pi;
379: {int i,j,loc;
380:  i=*pi;
381:  leteq(olab,"      ");
382:  for(j=0;
383:      j<6 && i<=72 && (isdigit(gyo[i]) || isupper(gyo[i]));
384:      j++) olab[j]=gyo[i++];
385:  *pi=i;
386:  loc=ltsearch(olab);
387:  if(pass==1 && loc==-1)
388:                {leteq(ltable[freearea].label,olab);
389:                 ltable[freearea++].def=0;
390:                 lend=freearea;
391:                }
392:  return loc;
393: }
394: object(b,c)
395: char b[],c;
396: {int j;
397:  putc(c,fp2);
398:  for(j=0;b[j]!='¥0';j++)
399:     putc(b[j],fp2);
400:  ocnt++; if(ocnt>=10){ocnt=0; putc('¥n',fp2);}
401:          else putc(':',fp2);
402: }
```

```
hexop(b,gyo,pi)
char b[],gyo[];int *pi;
{int i,j;
 for(j=0;j<4;j++)b[j]=' ';
 i=*pi;
 for(j=0;isdigit(gyo[i]) || (gyo[i]>='A' && gyo[i]<='F');j++)
                                                  if(j<4) b[j]=gyo[i++];
 b[4]='¥0';
 *pi=i;
 return j;
}
mtsearch(popc,mnemonic,mtable)
struct mt mtable[];
unsigned *popc;
char mnemonic[];
{int low,high,mid;char c;
    for(high=20,low=0 ; high>=low;   )
      { mid=(high+low)/2;
        c=scomp(mnemonic,mtable[mid].mcode);
        if(c=='<')      high=mid-1;
        else if(c=='>') low=mid+1;
        else {high=mid-1; low=mid;}
      }
    if(c=='=') {*popc=mtable[mid].ocode;return mtable[mid].type ;}
         else  return -1;
}
scomp(m1,m2)
char m1[],m2[];
{int i;
 for(i=0;m1[i]!=0 && m1[i]==m2[i];i++)
            ;
if(m1[i]>m2[i])       return '>';
else if(m1[i]==m2[i]) return '=';
else                  return '<';
}
btohexa(b)
char b[];
{unsigned i,n; char c;
  n=0;
  for(i=0;i<=3;i++)
       {c=b[i];n=n*16+((c<='9')? c-'0': c-'A'+10);
       }
  return n;
}
```

```
 1: #include <stdio.h>
 2: #define  LEND 200    /* the end of label table */
 3: #define  MEND 16384  /* the end of main memory */
 4: FILE *fp1;
 5: /*---- for loading------*/
 6: unsigned  m[MEND],ac;
 7: struct lab{char label[7]; unsigned value;} ltable[LEND];
 8: int freearea=0;   /*--freearea of ltanle--*/
 9: /*-----for chat---------*/
10: unsigned vect=0,svect=0,intreq=0,time0=0,time1=0;
11: unsigned pc=0,gr[5],h[5],str=0,listsw=0;
12:     /* h[0]--h[4] indicate halting points. */
13: char s[81];
14:
15: /*------- main function ---------*/
16:
17: main(argc,argv)
18: int argc; char *argv[];
19: { char term1[10],term2[10],slabel[7];
20:   FILE *fopen();
21:  int ofile,j,k,pass;
22: /*------title printing ------------*/
23:  shome();
24:  printf("Comet Simulator written by Y.Matsubara( 27 Sep. 1987).¥n");
25:  printf("¥n Loading Object Programs.¥n");
26: /*------title  printing end----------*/
27:  if(argc<2){printf("ﾋｷｽｳ ｶﾞ ﾅｲ.¥n"); goto owari;}
28:  for(pass=1;pass<=2;pass++)
29:   {ac=0;
30:    for(ofile=1;ofile<argc;ofile++)
31:    {char c; int i; unsigned ofset;
32:     fp1=fopen(argv[ofile],"r");
33:     while(getterm(term1)!=0)
34:      {
35:        switch(term1[0])
36:        {case 'A':m[ac++]=hexa(&term1[1]); break;
37:         case 'R':m[ac++]=hexa(&term1[1])+ofset; break;
38:         case 'S':{if(getterm(term2)==0 || term2[0]!='V')
39:                     {printf("S ﾉ ﾂｷﾞﾆ V ｶﾞ ﾅｲ.¥n"); goto owari;}
40:                  leteq(slabel,&term1[1]);
41:                  if(pass==1)
42:                   {storoku(&term1[1],ac+hexa(&term2[1]));
43:                    printf("ac=%04X  svalue=%04X ¥n",ac,hexa(&term2[1]));
44:                   }
45:               /*  Whether the interpretation is correct or not
46:                   should be made sure.
47:               */
48:                   ofset=ac;
49:                   if(pass==2)printf("--ofset=%04X¥n",ofset);
50:                   break;
51:                  }
52:         case 'T':{int i,t;
53:                   t=hexa(&term1[1]);
54:                   for(i=1;i<=t;i++)m[ac++]=0;
55:                   break;
56:                  }
57:         case 'E':{int l;
58:                   if(pass==2)
59:                    {l=ltsearch(&term1[1]);
60:                     if(l==-1)
61:                      {printf("external label undefined ¥"%s¥" in %s¥n"
62:                              ,&term1[1],slabel
63:                             );
64:                       m[ac]=0XFFFF;
65:                      }
66:                     else m[ac]=ltable[l].value;
67:                    }
```

```
 68:                     ac++;
 69:                     break;
 70:                    }
 71:         case 'L':{ac=hexa(&term1[1]); break;
 72:                    }
 73:         default: printf("Error in object file ¥"%s¥" .¥n",argv[ofile]);
 74:                  break;
 75:        }
 76:      }
 77:     fclose(fp1);
 78:    }
 79:   }
 80: /*---------------------------------------*/
 81: printf("Object program loading finished.  ac==%04X¥n",ac);
 82: /*---------------------------------------*/
 83: /* -----pc should be set to the value of first label----*/
 84:     pc=ltable[0].value;
 85:     printf("PC==%04X¥n",pc);
 86: /*------SP should  be give an initial value ----------*/
 87:     gr[4]=MEND;
 88: /*-------------------------------------------------*/
 89: chat();
 90: owari:;
 91: }
 92: getterm(term)
 93: char term[];
 94: {int i;char c;
 95:   for(i=0;i<10 && (c=getc(fp1))!=EOF && c!='¥n' && c!=':';i++)
 96:         term[i]=c          ;
 97:   term[i]='¥0'; return(i);
 98: }
 99: hexa(term)
100: char term[];
101: {int i; char c; unsigned n=0;
102:  for(i=0;term[i]!='¥0';i++)
103:   {c=term[i];n=n*16+((c<='9')? c-'0':c-'A'+10);
104:   }
105:  return n;
106: }
107: storoku(term1,value)
108: char term1[]; int value;
109: { int loc;
110:      if((loc=ltsearch(term1))!=-1)
111:        { printf("label conflict %s .¥n",term1); return -1;}
112:      else
113:        { if(freearea==LEND)
114:            {printf("ltable overflow.¥n");return -1;}
115:          leteq(ltable[freearea].label,term1);
116:          ltable[freearea].value=value;
117:          freearea++;
118:          return 1;
119:        }
120: }
121: leteq(s1,s2)
122: char s1[],s2[];
123: {int i;
124:   for(i=0;(s1[i]=s2[i])!=0;i++);
125:   return i;
126: }
127: ltsearch(label)
128: char label[];
129: {int i;
130:  for(i=0; i<freearea && (!issame(ltable[i].label,label));i++)
131:                          ;
132:  if(i==freearea) return -1 ; else return i;
133: }
134: issame(s1,s2)
```

```
135: char s1[],s2[];
136: {int i;
137:    for(i=0;s1[i]!=0 && s1[i]==s2[i];i++);
138:    if(s1[i]==s2[i]) return 1;
139:          else       return 0;
140: }
141: shome()
142: {printf("%c[2J",0x1b);
143: }
144: chat()
145: {int i,j; char b[11]; unsigned n,n1;
146:  for(j=0;j<=4;j++)h[j]=0;
147: /*---------------------------------------*/
148: PROMPT:
149:  printf("COM>"); simin(80,s);
150:  for(i=0;s[i]!='¥0';i++)
151:      if(s[i]>='a' && s[i]<='z')
152:          s[i]=s[i]-0x20;
153:  for(i=0;isupper(s[i]) && i<10;i++) b[i]=s[i];
154:  b[i]='¥0';
155:  if(issame(b,"PC"))
156:    {if(s[i]=='¥0')printf("PC==%04X¥n",pc);
157:     else if(s[i++]=='=')
158:      {if(rhexa(&n,&i)>4 || s[i]!='¥0') goto COMERR;
159:       pc=n;
160:      }
161:     else goto COMERR;
162:    }
163:  else if(issame(b,"GR"))
164:    {char c;    c=s[i++];
165:     if(c=='¥0')
166:       {int j;
167:        for(j=0;j<=4;j++)printf("GR%d==%04X  ",j,gr[j]);
168:        printf("¥n");
169:       }
170:     else if(c>='0' && c<='4')
171:       {
172:        if(s[i]=='¥0')
173:          {printf("  GR%c==%04X¥n",c,gr[c-'0']);
174:          }
175:        else if(s[i]=='=')
176:          {i++;
177:           if(rhexa(&n,&i)>4 || s[i]!='¥0') goto COMERR;
178:           gr[c-'0']=n;
179:          }
180:        else goto COMERR;
181:       }
182:     else goto COMERR;
183:    }
184:  else if(issame(b,"STR"))
185:    {if(s[i]=='¥0')printf("STR==%04X¥n",str);
186:     else if(s[i]=='=')
187:       {i++;
188:        if(rhexa(&n,&i)>4 || s[i]!='¥0')goto COMERR;
189:        str=n;
190:       }
191:     else goto COMERR;
192:    }
193:  else if(issame(b,"PSW"))
194:    {if(s[i]!='¥0')goto COMERR;
195:     printf("STR==%04X; PC==%04X¥n",str,pc);
196:    }
197:  else if(issame(b,"M"))
198:    {if(s[i++]!=':')goto COMERR;
199:     if(rhexa(&n,&i)>4)goto COMERR;
200:     if(s[i]=='¥0')printf("M[%04X]==%04X¥n",n,m[n]);
201:     else if(s[i]=='-')
```

```
202:       {unsigned j,k,l,nn;
203:        i++;
204:        if(rhexa(&n1,&i)>4 || s[i]!='¥0')goto COMERR;
205:        if(n>=n1)goto COMERR;
206:        nn=n & 0XFFF8;
207:        printf("------- 0/8  1/9  2/A  3/B  4/C  5/D  6/E  7/F¥n");
208:        l=nn;
209:        for(j=nn;l<n1;j++)
210:          {printf("M[%04X]",l);
211:           for(k=0;k<=7;k++)
212:             {if(l>=n && l<n1)printf(" %04X",m[l]);
213:              else printf("     ");
214:              l++;
215:             }
216:           printf("¥n");
217:          }
218:       }
219:     else if(s[i]=='=')
220:       {i++;
221:        if(rhexa(&n1,&i)>4 || s[i]!='¥0')goto COMERR;
222:        m[n]=n1;
223:       }
224:     else goto COMERR;
225:    }
226:  else if(issame(b,"LON"))
227:    {if(s[i]!='¥0')goto COMERR;
228:     listsw=1;
229:    }
230:  else if(issame(b,"LOFF"))
231:    {if(s[i]!='¥0')goto COMERR;
232:     listsw=0;
233:    }
234:  else if(issame(b,"H"))
235:    {if(s[i]=='¥0')
236:       {int j;
237:        for(j=0;j<=4;j++)printf("H%d==%04X  ",j,h[j]);
238:        printf("¥n");
239:       }
240:     else if(s[i]>='0' && s[i]<='4')
241:       {char c;   c=s[i++];
242:        if(s[i]=='¥0') printf("H%c==%04X¥n",c,h[c-'0']);
243:        else if(s[i++]=='=')
244:          {
245:           if(rhexa(&n,&i)>4 || s[i]!='¥0')goto COMERR;
246:           h[c-'0']=n;
247:          }
248:        else goto COMERR;
249:       }
250:     else goto COMERR;
251:    }
252:  else if(issame(b,"E"))
253:    {if(s[i]!='¥0')goto COMERR;
254:     goto OWARI;
255:    }
256:  else if(issame(b,"GO"))
257:    {if(s[i]!='¥0')goto COMERR;
258: ITERATE:
259:     if(str>>15)
260:       {printf("The waite bit inhibits execution.¥n"); goto PROMPT;}
261:     execute();
262:     if(kbhit())
263:       {simin(80,s);
264:        if(issame(s,"H") || issame(s,"h"))
265:          {printf("Stop at %04X¥n",pc); goto PROMPT;}
266:       }
267:     for(j=0;j<=4 && pc != h[j];j++)
268:                 ;
```

```
269:     if(j<=4){printf("Halting point h%d==%04X¥n",j,pc); goto PROMPT;}
270:     if(str>>15)
271:       {printf("The waite bit was set. STR==%04X  PC==%04X¥n",str,pc);
272:        goto PROMPT;
273:       }
274:     goto ITERATE;
275:    }
276:  else if(issame(b,""))
277:    {if(s[i++]!='1' || s[i]!='¥0')goto COMERR;
278:     if(str>>15)
279:       {printf("The waite bit inhibits execution.¥n"); goto PROMPT;}
280:     execute();
281:    }
282:  else goto COMERR;
283:  goto PROMPT;
284: COMERR:printf("Command Error¥n");
285:  goto PROMPT;
286: OWARI:;
287: }
288: rhexa(n,i)
289: unsigned *n,*i;
290: {int j,k; unsigned m,l;char c;
291:  k=0;
292:  for(j=*i,m=0; isdigit(c=s[j]) || (c>='A' && c<='F');j++)
293:    {c=s[j];
294:     if(isdigit(c))l=c-'0'; else l=c+10-'A';
295:     m=m << 4  | l;
296:     k++;
297:    }
298:  *n=m; *i=j;
299:  return k;
300: }
301: isupper(c)
302: char c;
303: {if (c >= 'A' && c <= 'Z')return(1);
304:     else return(0);
305: }
306: isdigit(c)
307: char c;
308: {if(c>='0' && c<='9')return(1);
309:    else              return(0);
310: }
311: execute()
312: {unsigned i,j,k,l;
313:  if((str & 0X0100) && intreq==1) /*-------  Interruption  */
314:    {intreq=0;
315:     /*  save the current state */
316:     i=svect;
317:     m[i++]=str;m[i++]=pc;
318:
319:        /*  load new state */
320:     i=vect;
321:     str=m[i++];pc=m[i++];
322:
323:    }
324:  else    /*-----------------------  Execution of Instruction */
325:    {char s[81]; unsigned opc,w1,w2,w3;
326:     w1=mr(pc++);  opc=w1 >> 8;
327:     if(w1==0XFF00)str=str | 0X8000;  /*EXIT  */
328:     else if(w1==0XFF01)              /*IN    */
329:       {w2=mr(pc++);w3=mr(pc++);
330:        printf("?");
331:        if(simin(80,s)!=EOF)
332:          {for(i=0;s[i]!='¥0';i++)mw(w2++,s[i]);
333:           mw(w3,i);
334:          }
335:        else mw(w3,-1);
```

```
336:       }
337:     else if(w1==0XFF02)                /*OUT  */
338:       {w2=mr(pc++);
339:        w3=mr(pc++);k=mr(w3);
340:        printf(":");
341:        for(;k>0;k--)printf("%c",mr(w2++));
342:        printf("¥n");
343:       }
344:     else if(opc==0X10)    /*LD   */
345:       {w2=mr(pc++);
346:        gr[g(w1)]=mr(w2+x(w1));
347:       }
348:     else if(opc==0X11)    /*ST   */
349:       {w2=mr(pc++);
350:        mw(w2+x(w1),gr[g(w1)]);
351:       }
352:     else if(opc==0X12)    /*LEA  */
353:       {w2=mr(pc++);
354:        i=w2+x(w1);
355:        gr[g(w1)]=i;
356:        str=(str & 0XFFFC) | flag(i);
357:       }
358:     else if(opc==0X20)    /*ADD  */
359:       {w2=mr(pc++);
360:        i=(  gr[g(w1)]=gr[g(w1)] + mr(w2+x(w1))  );
361:        str=(str & 0XFFFC) | flag(i);
362:       }
363:     else if(opc==0X21)    /*SUB  */
364:       {w2=mr(pc++);
365:        i=(  gr[g(w1)]=gr[g(w1)] - mr(w2+x(w1))  );
366:        str=(str & 0XFFFC) | flag(i);
367:       }
368:     else if(opc==0X30)    /*AND  */
369:       {w2=mr(pc++);
370:        i=(  gr[g(w1)]=gr[g(w1)] & mr(w2+x(w1))  );
371:        str=(str & 0XFFFC) | flag(i);
372:       }
373:     else if(opc==0X31)    /*OR   */
374:       {w2=mr(pc++);
375:        i=(  gr[g(w1)]=gr[g(w1)] | mr(w2+x(w1))  );
376:        str=(str & 0XFFFC) | flag(i);
377:       }
378:     else if(opc==0X32)    /*EOR  */
379:       {w2=mr(pc++);
380:        i=(  gr[g(w1)]=gr[g(w1)] ^ mr(w2+x(w1))  );
381:        str=(str & 0XFFFC) | flag(i);
382:       }
383:     else if(opc==0X40)    /*CPA  */
384:       {w2=mr(pc++);
385:        i=gr[g(w1)] - mr(w2+x(w1));
386:        str=(str & 0XFFFC) | flag(i);
387:       }
388:     else if(opc==0X41)    /*CPL  */
389:       {int i1;
390:        w2=mr(pc++);
391:        i1=gr[g(w1)] - mr(w2+x(w1));
392:        if(i1>0)i=0; else if(i1<0) i=2; else i=1;
393:        str=(str & 0XFFFC) | i;
394:       }
395:     else if(opc==0X50)    /*SLA  */
396:       {w2=mr(pc++);
397:        i=gr[g(w1)] << (w2+x(w1));
398:        gr[g(w1)]=i;
399:        str=(str & 0XFFFC) | flag(i);
400:       }
401:     else if(opc==0X51)    /*SRA  */
402:       {int i1;
```

```
403:        w2=mr(pc++);
404:        i1=gr[g(w1)];  i1=i1>> (w2+x(w1));
405:        gr[g(w1)]=i1;
406:        str=(str & 0XFFFC) | flag(i);
407:       }
408:     else if(opc==0X52)      /*SLL  */
409:     {w2=mr(pc++);
410:        i=gr[g(w1)] << (w2+x(w1));
411:        gr[g(w1)]=i;
412:        str=(str & 0XFFFC) | flag(i);
413:       }
414:     else if(opc==0X53)      /*SRL  */
415:       {w2=mr(pc++);
416:        i=gr[g(w1)] >> (w2+x(w1));
417:        gr[g(w1)]=i;
418:        str=(str & 0XFFFC) | flag(i);
419:       }
420:     else if(opc==0X60)    /*JPZ  */
421:       {w2=mr(pc++);
422:        i=str & 0X3;
423:        if(i==0 || i==1)pc=w2+x(w1);
424:       }
425:     else if(opc==0X61)    /*JMI  */
426:       {w2=mr(pc++);
427:        i=str & 0X3;
428:        if(i==2)pc=w2+x(w1);
429:       }
430:     else if(opc==0X62)    /*JNZ  */
431:       {w2=mr(pc++);
432:        i=str & 0X3;
433:        if(i==0 || i==2)pc=w2+x(w1);
434:       }
435:     else if(opc==0X63)    /*JZE  */
436:       {w2=mr(pc++);
437:        i=str & 0X3;
438:        if(i==1)pc=w2+x(w1);
439:       }
440:     else if(opc==0X64)    /*JMP  */
441:       {w2=mr(pc);
442:        pc=w2+x(w1);
443:       }
444:     else if(opc==0X70)    /*PUSH */
445:       {w2=mr(pc++);
446:        gr[4]--;
447:        mw(gr[4],w2+x(w1));
448:       }
449:     else if(opc==0X71)    /*POP  */
450:       {pc++;
451:        gr[g(w1)]=mr(gr[4]);
452:        gr[4]++;
453:       }
454:     else if(opc==0X80)    /*CALL */
455:       {w2=mr(pc++);
456:        gr[4]--;
457:        mw(gr[4],pc);
458:        pc=w2+x(w1);
459:       }
460:     else if(opc==0X81)    /*RET  */
461:       {pc=mr(gr[4]++);
462:       }
463:     else if(w1==0XFB00)   /*INT 0    */
464:       {str=str & 0XFEFF;
465:       }
466:     else if(w1==0XFB01)   /*INT 1    */
467:       {str= str | 0X0100;
468:       }
469:     else if(opc==0XFE)    /*LPSW */
```

```
470:        {w2=mr(pc);
471:         str=mr(w2++);pc=mr(w2++);
472:
473:        }
474:      else if(opc==0XFC)     /*VECT */
475:        {vect=mr(pc++);
476:        }
477:      else if(opc==0XFD)      /*SVECT*/
478:        {svect=mr(pc++);
479:        }
480:      else      /* operation code error */
481:        {str=str | 0XC000;
482:        }
483:     }  /*------------------the end of Execution of instruction.--*/
484: /*----  Here the timer should be renewed. -----*/
485:  if(time0>>15==1 && time1==1)intreq=1;
486:  if(time1!=0)time1--;
487: }
488: mr(a)
489: unsigned a;
490: {if(a<=MEND)return m[a];
491:  else if(a==0XFFF0)return str;
492:  else if(a==0XF000)return time0;
493:  else if(a==0XF001)return time1;
494:  else { str=str | 0XA000; return 0;}
495: }
496: mw(a,v)
497: unsigned a,v;
498: {if(a<=MEND){m[a]=v; return 1;}
499:  else if(a==0XFFF0){str=v; return 1;}
500:  else if(a==0XF000){time0=v; intreq=0; return 1;}
501:  else if(a==0XF001){time1=v; intreq=0; return 1;}
502:  else {str=str | 0XA000; return 0;}
503: }
504: g(w1)
505: unsigned w1;
506: {unsigned i;
507:  i= (w1 & 0X00F0)>>4;
508:  if(i>4){str=str | 0X9000; i=0;}
509:  return i;
510: }
511: x(w1)
512: unsigned w1;
513: {unsigned i;
514:  i=w1 & 0X000F;
515:  if(i>4){str=str | 0X8800; i=0;}
516:  if(i==0)return 0;
517:  else return gr[i];
518: }
519: flag(i)
520: int i;
521: {if(i>0) return 0;
522:  else if(i<0) return 2;
523:  else return 1;
524: }
525: simin(n,s)
526: char s[];int n;
527: {char c;int j;
528:  for(j=0;j<n && (c=getchar())!=EOF && c!='¥n';j++)s[j]=c;
529:  s[j]='¥0';
530:  return c;
531: }
```