TSS 環境におけるプログラム実行演示システムの作成とその検討†

宇津宮 孝一†† 樽 美和幸†††
荒牧重登†† 牛島和夫†††

TSS がもつ応答性の良さと豊かな情報提示機能を利用して，計算機の動作やプログラムの実行過程を実際と同じように模倣し，その様子を通常の CRT 端末や大きな画面からなるスクリーン・ディスプレイ装置に映し出すプログラム実行演示システムを開発した．このシステムを情報処理教育等に用いれば，集団教育時には教師がスクリーン・ディスプレイを通じて皆の前で実演して見せることにより，個別学習時には各人がプログラムの実行の様子を擬似体験することにより，学習者は計算機の概念，アルゴリズム及びプログラミング言語の習得が容易になり，教育効果の向上が期待できる．

本論文では，このような目的で実現された3つの言語に対する実行演示システム，すなわち（1）アセンブリ言語用 SIM960，（2）PASCAL 用 PASVID，及び（3）FORTRAN 用 FORVID に関し，使用するハードウェアの構成，各システムのソフトウェアの構成と機能について述べ，これらの実現方法とその問題点等について考察している．

1. はじめに

最近各教育機関においても，情報処理教育等を実施するために TSS を採用する事例が多くなってきており，かなりの成果を上げているようである[1]．この要因は，会話型処理により学習者に対する計算機からの応答が即座に得られるので学習意欲をそそること及び十分な実習時間が与えられること等に基づくものと考えられる．しかしながら我々の教育実践を通じていくつかの懸念すべき問題点も浮かび上がっている．たとえば，

（1） 意図した結果が得られない場合，とにかくやってみるという気持が先立ち，静かに考えたり，良質のプログラムを書かねばという態度が薄れがちとなること

（2） 各人の学習進度に関してばらつきの度合が著しくなる傾向が強いこと

（3） 端末中心のため端末数に不足をきたし，また計算機そのものに対する体験学習の機会が少なくなること

上記の問題に対処し，教育効果をも高めるためには，計算機の命令の動きやプログラムの実行の様子を実演して表示（演示）することにより，集団教育時には演示内容を全員で議論する過程を通じて，個別学習時には擬似体験を通じて，計算機及びプログラミングに対する理解を深めるシステムが是非とも必要である[2),3)]．

しかしながら，動かして見せる一つ一つのプログラムごとに演示を行うための部分をその都度作成するのは大変な労力を強いられる．演示対象のプログラムを用意するだけで実行の様子を表示できれば，教師にも学習者にも負担を掛けることがなく大変便利である．

我々はこのような主旨に基づいて，演示用のスクリーン・ディスプレイ装置を設置し，皆の前で実際にプログラムを動かして個々の文または命令の実行の様子を見せるというプログラム実行演示システム（以下演示システムと呼ぶ）を次の言語に対して開発した．

（1） アセンブリ言語用演示システム SIM960
（2） PASCAL 用演示システム PASVID
（3） FORTRAN 用演示システム FORVID

もちろん，これらは教師が多人数の学習者に対して用いるばかりでなく，学習者自身が TSS 用 CRT 端末で利用できるので，両者の併用により学習者の計算機に対する概念やアルゴリズムの習得が容易になる．

本論文では，演示システムのハードウェア構成，ソフトウェア構成とその機能，及び各システムの概要と

† Implementation of a Visual Demonstration System of Program Execution in TSS Environment by KOUICHI UTSUMIYA (Educational Center for Information Processing, Kyushu University), KAZUYUKI TARUMI (Department of Computer Science and Communication Engineering, Kyushu University), SHIGETO ARAMAKI (Educational Center for Information Processing, Kyushu University) and KAZUO USHIJIMA (Department of Computer Science and Communication Engineering, Kyushu University).

†† 九州大学情報処理教育センター
††† 九州大学工学部情報工学科

機能を述べ，さらにその実現方法等について検討する．

2. 演示システムの構成と機能

2.1 ハードウェアの構成

従来の TSS 端末は，個人のプログラム開発やプログラミング等の個別学習には十分効果を発揮しているが，集団教育で多人数の学習者を対象とするプログラムの実行演示には適さない．すなわち，1台で演示するには画面が小さ過ぎて判読しにくく，複数台で使用する場合にはそれらの制御が容易でない．

そこで，講義室に従来のTV映像のほか計算機出力もカラー表示できる高解像度72インチ（横 150cm×縦 110cm）スクリーン・ディスプレイ装置（SONY VPP 721 S）を設置し，これにプログラムの実行の様子を表示させることにした．

スクリーン・ディスプレイ装置は表示能力しかないので，これを駆動するためTV映像の場合はTVカメラ，VTR 等の機器が，計算機出力の場合はディスプレイ端末が必要である．端末は，次の理由から従来のTSS 端末を用いずにマイクロ・コンピュータ（日立 Basic Master MB 6890 Level 3）を採用した．

（1） ホスト計算機の端末としてばかりでなく，ホスト計算機と独立に動作させることもできる．

（2） ホスト計算機からのメッセージをマイコンのプログラムで見やすい形に編集することができる．

（3） 平仮名及び図形の表示も可能である．

（4） カラー表示ができるので，図形，文字及び画面の背景色を見やすいように調整できる．

演示システムのハードウェア構成を**図1**に示す．図に示すように，モデム・インタフェースをもつ計算機であれば自由に接続できるばかりでなく，従来の視聴覚教材の提示装置としても利用できるので，集団教育時に非常に有効である．

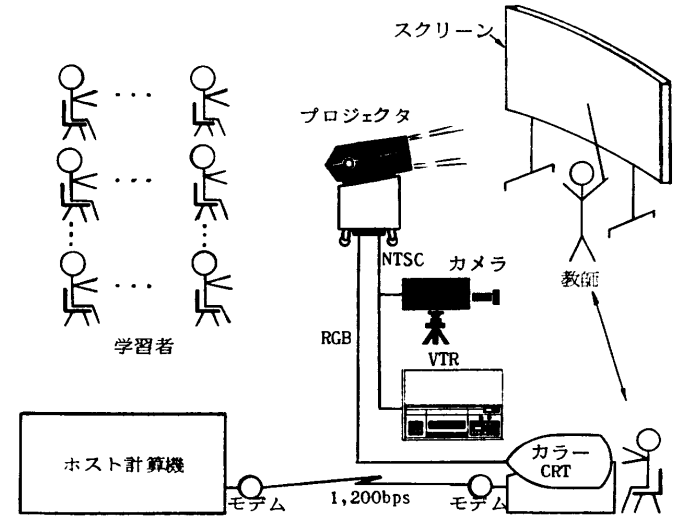


図 1 演示システムのハードウェア構成
Fig. 1 Configuration of visual demonstration system.

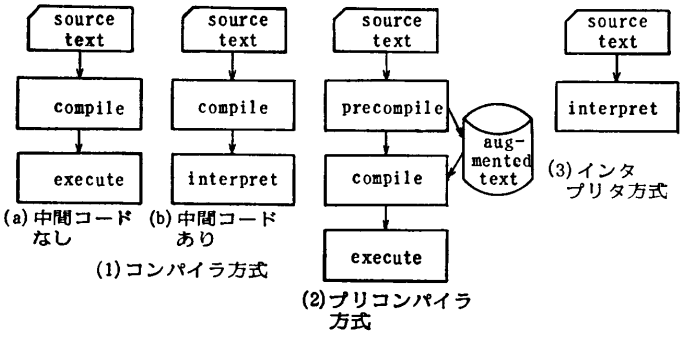


図 2 システムの実現方式
Fig. 2 Typical forms of system implementation.

プログラムの実行演示は，教師がマイコンにコマンド（指令）を入力し説明を加えながら行ってゆくことになる．この演示システムはホスト計算機 ACOS-700 上で動作し，マイコンを通常の TSS 端末として取り扱っているので，従来どおり各人がTSS での使用もできる．

2.2 ソフトウェアの構成

演示システムのソフトウェアの構成方法としては，**図2**に示すように(1)コンパイラ方式，(2)プリコンパイラ方式，及び(3)インタプリタ方式の3つがある．演示システムの実現方式を評価する際の基準として次の項目が考えられる．

①起動時間　演示システムが実行・表示を開始するまでの時間を表す．演示システムは授業中に使用することが多く，この起動の間，多人数に対して待たせることになるので特に注意を払わなければならない．

②実行速度　表示を伴う実行の場合は，人による確認動作が必要となるので実行時間はそれほど問題とはならない．しかしながら，表示しないで実行する場合は実行時間はできる限り短い方がよい．

③機能　演示システムにおいては，変数

の表示方法，表示の条件等機能を拡大すると①②が増大することになるので，機能を考える際にはその配慮が必要である．また，このようなシステムは特に現場での使用経験を生かすことが重要となるので，どのような機能を追加すべきかという拡張性に対しての配慮も必要となる．

以下これらの点を考慮しながら，演示システムとして実現する場合について各方式を考察してみる．

（1） コンパイラ方式は，演示するプログラムを解析し，必要があれば表示用のコードをそう入したオブジェクトを出力する．この中にはオブジェクトである中間コードをインタプリタが解釈・実行し表示を行う方式も含める．中間コードをもたない場合には，演示システムの全機能をオブジェクトに含ませなければならず，コンパイラの解析負担は大きくなり，実行までの起動（待ち）時間は長くなる．中間コードをもつ場合には，インタプリタ側に機能を分担させることにより起動時間も短くできる．機能を充実するには中間コードをもたせて起動時間を短くした方がよい．

（2） プリコンパイラ方式は既存のコンパイラを利用する方式である．これは演示用のコードをソース・プログラムにそう入するので，プリコンパイラ部分での解析負担は大きい．さらに，プリコンパイルとコンパイルの2段階の処理が必要となり起動時間は三者の中で一番長い．しかし，実行速度は機械語を直接実行するので一番速くなる．

（3） インタプリタ方式はソース・プログラムを直接解釈・実行し表示を行うので実行速度は遅くなる．起動時間はインタプリタが起動されるまでの時間なので非常に短いと考えられる．インタプリタ型の言語や制御構造，データ構造が単純な言語に適している．制御構造やデータ構造が豊富で複雑な高級言語の場合には，それだけ解析が多くなり実行速度も遅く，システム自体も大きくなり実用的でなくなる．

2.3 機　能

プログラムの実行の様子を見せる場合，（1）言語がもつ制御構造に基づいた制御の移行（control flow）の表示と（2）プログラムがもつデータ構造に対するデータの流れ（data flow）の表示との二とおりの方法が考えられる．（1）は言語により定められた制御構造があるので，演示システムで言語ごとに，制御に対する表示方法を規定できる．（2）は言語のデータ構造のほかに各プログラムごとにデータの流れが異なるので，演示システムで解析し適当な表示を行うにはシステムが大きくなり実現は容易でない．本稿のシステムでは，（1）の制御位置の表示を各言語に対して定め，（2）のデータ表示は使用者がプログラムの実行に応じて端末指令として与えることにより行うことにした．

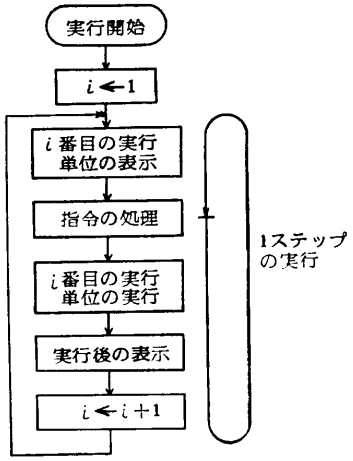


図 3 演示システムの実行過程
Fig. 3 Execution process of demonstrator.

演示システムの実行は，演示するプログラミング言語によりシステム側で定めた実行単位を基準にして進められ，これに対する表示を制御位置の表示としている（**図3**）．すなわち，演示システムは，i 番目の実行単位の実行；その実行後の表示；$i+1$ 番目の実行単位の表示の3過程を1ステップの実行とする．指令入力や表示を省略することも可能であり，これらの組合せにより，（1）manual モード，（2）automatic モード，及び（3）blind モードの3つの実行モードがある．

（1） manual モードは，指令入力時に復帰（return）キーを直接押下するごとに1ステップずつ実行・表示を行う．つまり次に復帰キーを押すまでの間，プログラムの実行は一時中断された状態になる．その間に，教師がアルゴリズムや言語の構文といった特徴を説明したり，質疑応答を繰返すことにより，言語やプログラミングに対する学習者の理解が深められる．

（2） automatic モードは，動画を見るように一定の時間間隔で自動的に実行・表示を繰返し，中断（break）キーにより manual モードに移行する．このモードは，変数の値が実行のたびにどのように変化してゆくかを追跡したり，論理的な誤りにより発生する予期せぬ動作を発見する場合等に有効である．特に従来暗箱の中に隠されていたプログラムの振舞を視覚的に確認できるという効果は測り知れない．

これらの2つのモードでは表示を行うが，

（3） blind モードは，表示をしないで指定されたステップ数または中断キーが押されるまで実行を続ける．したがって，表示しない分だけ実行速度が速いので，表示を必要としない部分から早く抜け出してプログラムの他の部分を見たい場合等に用いると効果的である．

各実行単位における表示は，manual 及び automatic モードで行うが，表示内容は言語により異なる．

3. アセンブリ言語用演示システム SIM960

3.1 概　要

アセンブリ言語用演示システム SIM960 (Simulator 960) は，計算機の構造・命令・割込み等の概念を教育するために開発したシミュレータである[4]．これは M. I. T. で作成された SIM 360[2] の考え方を参考にし，TSS 環境で動かすために計算機の操作パネル等の概念を導入して新たに開発したものである．モデル計算機は次の理由から IBM/360 計算機を選んだ．

（1） この計算機は近年の計算機の雛形で，現在も実働しており，後継機種も多い．

（2） このアセンブリ言語によって得られた知識は他の計算機にも応用できる．

（3） この計算機をモデルにした教科書[5]も多く出版されているので教育的である．

SIM 960 は**図4**に示す計算機をシミュレートする．各入出力装置には TSS 端末が割付けられており，装置が起動された時点でデータを入力したり，実行結果を出力したりすることができる．また，磁気ディスク・ファイルの使用も可能である．したがって，使用者は TSS 環境で仮想的な裸の計算機を各人が占有することになる．

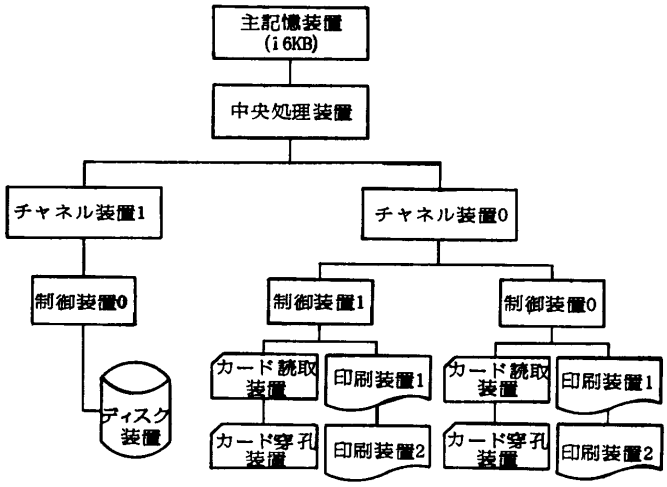


図 4 シミュレートする計算機の構成

Fig. 4 System configuration of simulated computer.

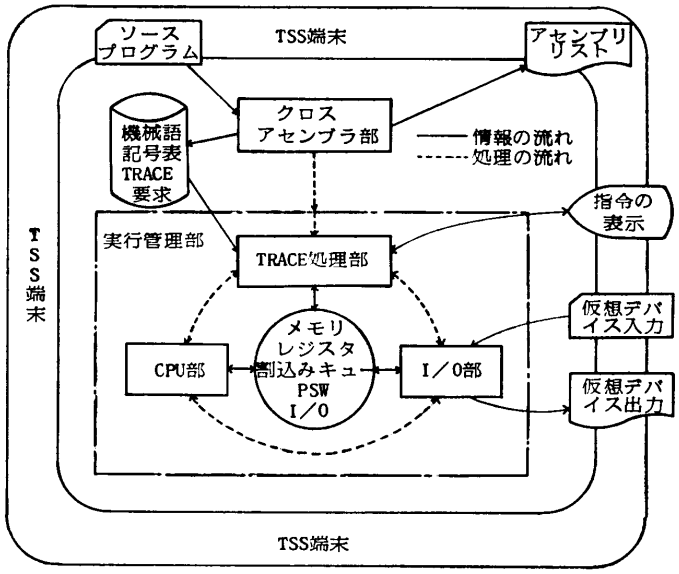


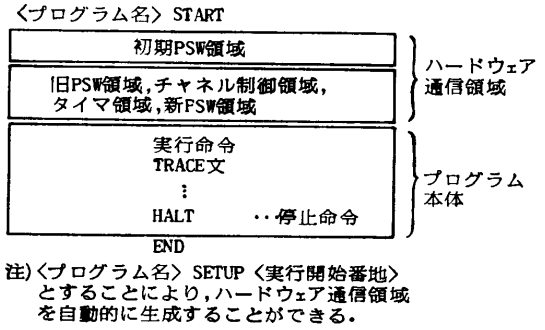
図 5 SIM960 の構成

Fig. 5 SIM960 organization.

3.2 SIM960 の構成

SIM960 自身は，**図5**に示すようにクロス・アセンブラ部と実行管理部とから成り，IBM/360 計算機の機械語を中間コードとして用いるコンパイラ方式を採用した．この方式を採ったのは次の理由による．

（1） モデル計算機のアセンブラがなく，SIM960 自体でエラー・チェックを行う必要がある．



注）〈プログラム名〉 SETUP 〈実行開始番地〉 とすることにより，ハードウェア通信領域を自動的に生成することができる．

図 6 SIM960 におけるプログラムの構造

Fig. 6 Program structure in SIM960.

表 1 SIM960 の指令一覧
Table 1 SIM960 commands.

種別	入力形式 指令名	入力形式 オペランド	機能・意味
実行制御指令	[S]		命令を1ステップ実行する.
	A[V]	ステップ数 [Δ秒数]	指定ステップ数命令を実行する. 秒数が指定された時は,その値の間隔で各命令を実行する.
	PS	番地	指定番地まで実行後一時停止する.
	PO		PS指令を解除する.
	JP	番地	命令カウンタの値を指定された番地に変更する.
内容変更指令	CM	番地Δ内容	指定された番地の主記憶の内容を変更する.
	CR	レジスタ番号Δ内容	指定されたレジスタの内容を変更する.
表示指令	DD	装置番地	装置の状態を表示する.
	DI		割込みキューの内容を表示する.
	DM	番地Δ語数	指定された番地から指定された語数だけ主記憶の内容を表示する.
	DR		すべてのレジスタの内容を表示する.
	DS		標準のTRACE情報を出力する.
	DT		TRACE要求を表示する.
TRACE指令	KL	TRACE条件	プログラム中のTRACE条件を解除する.
	TR	TRACE条件	TRACE条件を設定する.
	TO	TRACE条件	TRACE条件を解除する.
その他の指令	EN		シミュレーションを終了させる.
	EX	TSSコマンド	TSSコマンドを呼出す.

注) Δは1個以上の空白を,[ ] は省略できることを表す.

表 2 SIM960 の TRACE 機能
Table 2 TRACE functions in SIM960.

形式 命令	形式 オペランド	TRACE情報表示時点
TRACE	BRANCH	分岐が生じた時
	ADDRS(番地)	指定番地が命令のオペランドとなった時
	EXEC(番地)	指定番地の命令が実行される直前
	INSTR(簡略命令コード)	指定の簡略命令コードの命令が実行された時
	INTRPT(割込み)	指定の割込みが発生した時
	CHANL(チャネル)	指定チャネルが何らかの動作をした時
	DUMP	この次の命令が実行された時
TROFF	上記のオペランド又はALL	TRACE機能を解除する.

注1) オペランドはコンマで区切って複数個並べることができる.
2) オペランドの後にコロンに続けて制限回数を指定することができる.

(2) ソース・プログラムを直接解釈・実行する場合にも，定数や変数領域の解釈及び番地の決定などのために，実行前に少なくとも1回はプログラム全体の解析が必要となる.

(3) (1),(2) によりクロス・アセンブラを分離した場合，プログラムの解析は簡単なものであり，起動時間の増加は少ないと考えられる.

(4) さらに実行管理部を分離することにより，演示システムの実行時の機能を充実できる.

実行管理部は，TRACE 処理部, CPU 部，及び I/O 部の3つの機能単位から構成されている. クロス・アセンブラ部はアセンブリ・プログラムを機械語に変換し，TRACE 処理部で必要な記号表等の情報を生成する. TRACE 処理部は，機械語を主記憶に格納し，プログラムあるいは TSS 端末からの要求に応じて実行・表示を行う. CPU 部は，I/O 関係命令以外の 75 種類の命令（論理演算, 固定小数点演算, 分岐・比較, 特権命令等）を逐次解釈・実行し，スーパバイザ・コール等5種類の割込みを処理する. I/O 部は，SIO (Start I/O) 命令等3種類の I/O 命令及び I/O 割込みを処理する.

3.3 SIM960 の機能

SIM960 に対するプログラムの構造を**図 6** に示す. アセンブルを終了したプログラムは初期 PSW (Program Status Word) の指定番地から実行を開始し，プログラム中の TRACE 要求または TSS 端末からの指令に基づいて，実行過程を表示しながら実行が進められる. 実行の最小単位は1つの機械語命令が1ステップである. SIM960 では，**表 1** に示す計算機の操作パネルに対応する実行制御指令や内容変更・表示指令のほか，**表 2** に示す7つの条件 (TRACE 要求) に対して入出力命令を使うことなく簡単に表示を行わせることもできる.

SIM960 は各命令の実行前に**図 7** に示すような表示を行い，その表示には標準及びオプション出力がある.

(1) 標準出力には次のものがある.
(a) 実行モード，プログラム名 (図7の①,②)
(b) TRACE 要求の種類 (図7の④)
(c) プログラム・カウンタの値 (図7の③)
(d) 実行命令数と実行時間 (図7の⑤, ⑥)

```
*S960AS SOURCE OBJECT RESULT ・・・ファイルSOURCEのソース・プログラムを
                                   アセンブルする.
*LIST RESULT ・・・ファイルRESULTのアセンブリ・リストを表示する.
SIM960 CROSS ASSEMBLER V02.L01
 LOC  OBJECT CODE       SOURCE STATEMENT

00000                   *-------------------------PROGRAM GCD
00000 00000000 00000180 GCD     SETUP   BEGIN                  プログラム名
00180                   M       EQU     4
00180                   N       EQU     5
00180 1945              BEGIN   CR      M,N     ・・・Compare M with N
00182 47800196                  BZ      STOP    ・・・If M=N then go to STOP
00186 47200190                  BP      SUBN    ・・・If M>N then go to SUBN
0018A 1B54                      SR      N,M     ・・・N:=N−M
0018C 47F00180                  B       BEGIN   ・・・Go to BEGIN
00190 1B45              SUBN    SR      M,N     ・・・M:=M−N
00192 47F00180                  B       BEGIN   ・・・Go to BEGIN
00196 0000              STOP    HALT            ・・・Halt
00198                           END
*S960GO OBJECT ・・・ファイルOBJECTの目的プログラムをロードして実行する.
   〈開始メッセージ・初期状態の表示〉 ・・・省略
==> TR TSS;R=(M,N)    ・・・TRACE要求を設定し,レジスタM,Nの値を出力させる.
==> CR M 40  CR N 16  ・・・レジスタM,Nの値をそれぞれ40,16に変更する.
==> PS STOP           ・・・番地"STOP"に達した場合,一時停止するよう指示する.
==> AV 10 1           ・・・10ステップの命令を1秒間隔で自動的に実行するよう指示
                           する.
    SIM960 SIMULATOR  VER.02.01      ①MODE :  AUTO    ②PROGRAM:GCD
-------------------------------------------------------------------------
LOC:000182 ③      ④TRACE:TSS       ⑤COUNT:     1    ⑥TIME  :        0
MNE:BC⑦           ⑧CODE :4780      ⑨ADDR1:000196     ADDR2  :
PSW: IDENT :CURRENT  CMASK:'00000000'  KEY  : '0000'  AMWP  :    '0000'
 ⑩  I_CODE:0000      ILC  :'10'       CC   :  '10'    P_MASK:    '0000'
R04:(        40:00000028:....)  R05:(        16:00000010:....)⑪
              Mの値(40)                        Nの値(16)
   〈途中の表示〉 ・・・省略
R04:(         8:00000008:....)  R05:(        16:00000010:....)
              途中のMの値                      途中のNの値
==> A 1000・・・自動的に1000ステップ命令を実行する.

    SIM960 SIMULATOR  VER.02.01      MODE :MANUAL     PROGRAM:GCD
-------------------------------------------------------------------------
LOC:000196          TRACE:TSS        COUNT:    17     TIME  :       11
MNE:HALT            CODE :0000       ADDR1:          ADDR2  :
PSW: IDENT :CURRENT CMASK:'00000000' KEY  : '0000'  AMWP  :    '0000'
     I_CODE:0000    ILC  :'01'       CC   :  '00'    P_MASK:    '0000'
R04:(         8:00000008:....)  R05:(         8:00000008:....)
              Mの値(求める最大公約数)          Nの値
==> EN ・・・終了指示
```

図 7 SIM960 の使用例

Fig. 7 Example of SIM960.

(e) 命令コードとオペランド (図7の⑦, ⑧, ⑨)

(f) PSW の値 (図7の⑩)

(2) オプション出力には次のものがある.

(a) レジスタの内容 (図7の⑪), 主記憶の内容

(b) 各種ステータス

SIM960 での実行位置は, ソース・プログラムではなくプログラム・カウンタや命令コードなどで表しているので, 実際の使用に当ってはアセンブリ・リストを併用する必要がある.

3.4 使用例

図7に最大公約数を求める使用例を示す.

4. PASCAL 用演示システム PASVID

4.1 概 要

PASVID (PASCAL Visual Demonstration System) は言語 PASCAL[6] に対する演示システムである[7]. PASVID は主として教育を目的とすることから, 演示するプログラムは 200 行程度を目安とし, 文法上の誤りはないものとしている. したがって, システム自体は文法のエラー・チェックは行わない.

4.2 PASVID の構成

PASVID は, すでにいくつかの処理系でも実績のあるプリコンパイラ方式を採用した[8]. ソフトウェア

の構成はこれらとほぼ同じなので省略する．プリコンパイラ部では，演示するプログラムに次の情報をそう入しコンパイラに渡す．

①初期設定手続きとその呼出し部分

②実行制御位置表示手続きとその呼出し部分

③変数の表示・変更手続きとその呼出し部分

コンパイルが終了すると演示するプログラムをファイルから入力し，TSS 端末からの指令に基づいて実行・表示を行う．次の理由でプリコンパイラ方式とした．

（1） 実行制御の表示を主体とすること及び PASCAL 自体が在来の言語に比べて言語仕様が簡潔であることから，プログラムを解析して上記の情報をそう入するのにそれほど時間はかからないと考えられる．

（2） コンパイラやインタプリタ方式に比べて開発期間が短くて済み，高級言語に対する演示システムに求められる機能等の研究に即座に取り組める．

（3） 実行速度が速い．

4.3 PASVID の機能

PASCAL プログラムの実行演示においては，プログラムの実行単位をどのようにとるかが問題となる．というのは，アセンブリ言語の場合には1命令を1実行単位とすることができたが，PASCAL にはそのような区切りがないためである．PASVID では，それぞれの文や手続きの先頭などで表示の効果があると思われる点（これを表示点と呼ぶ）を決め，ある表示点から次の表示点までを実行の単位とした．while 文における表示点と表示部分のテキストの展開例を**図8**に示す．

PASVID では実行制御の表示は，図8に示すようにこれから実行しようとする部分をソース・プログラム上で指示記号（"="）によって指示することにより行う．また，構文の提示や実行によって参照された変数の値の表示機能ももたせている．構文を示し，教師がそれに説明を加えることにより，PASCAL の構文とその制御構造を学習者に理解させやすくする．これは学習者自身が個別学習をする場合にも有効である．

PASVID は**表3**にまとめた端末指令を使用しながら実行演示を行うことになる．

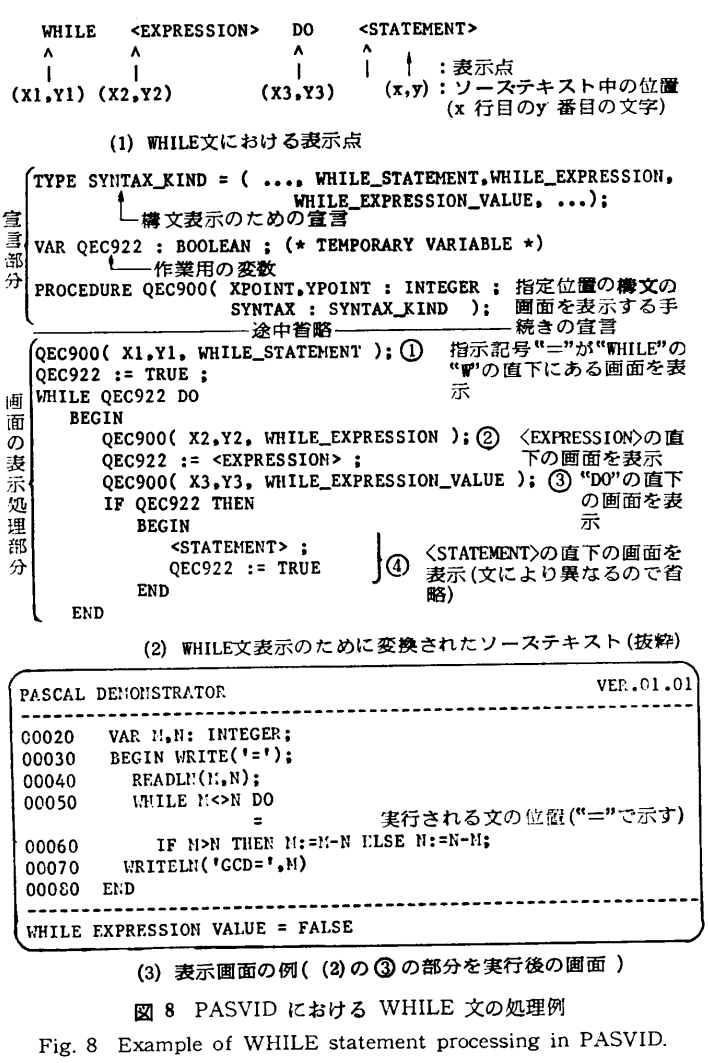


図 8 PASVID における WHILE 文の処理例

Fig. 8 Example of WHILE statement processing in PASVID.

5. FORTRAN 用演示システム FORVID

5.1 概　要

FORVID (FORTRAN Visual Demonstration System) は言語 FORTRAN に対する演示システムであり[9]，言語仕様としては FORTRAN 77 のサブセットを用いた[10]．

5.2 FORVID の構成

FORVID は次の理由からインタプリタ方式を採用することにした．

（1） 演示開始までの起動時間を短くできる．

（2） FORTRAN のデータ構造は PASCAL ほど複雑でなく，表示のための解析を実行中に行っても実行速度の低下は大きくないと考えられる．

表 3 PASVID 及び FORVID の指令一覧
Table 3 Commands of PASVID and FORVID.

種別	指令名	入力形式	機能・意味
実行制御指令	S(Single step)	[S]	1ステップ実行し,表示を行う.
	A(Automatic)	A△{#行番号 / ステップ数}	指定の行番号又はステップ数に達するまで実行・表示を行う.
	B(Blind)	B△{#行番号 / ステップ数}	A 指令との違いは表示をしない点である.
	E(End)	E	実行を終了させる.
表示・変数制御指令	D(Display)	D△変数名	指定された変数名の値を表示する.
	R(Refer)	R△{+ / −}	+を指定するとこれ以降参照が起った変数の値を表示する. −を指定すると表示しない.
	C(Change)	C△変数名	変数の値を変更する. 変更する値は,次行"="の出力に続いて入力する.
	F(Format)	F△{+ / −}	+を指定すると,これ以降構文の形式も併せて表示する. −を指定すると表示しない.

注) △は1個以上の空白を,[ ]は省略できることを,{ }は選択パラメータを表す.

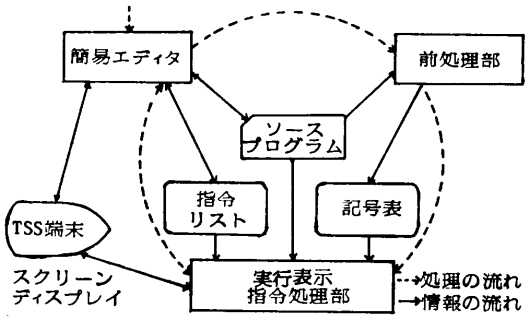


図 9 FORVID の構成
Fig. 9 FORVID organization.

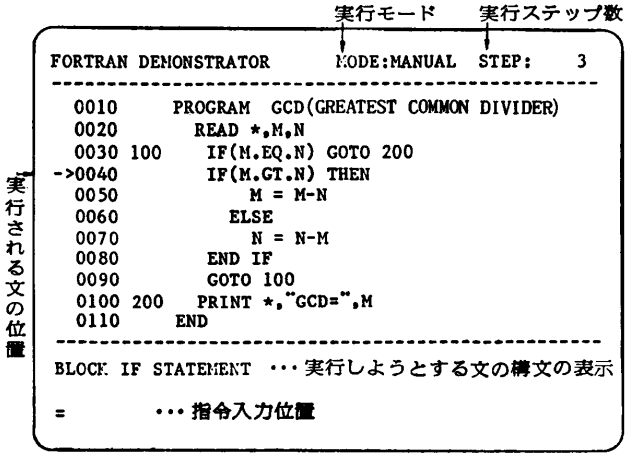


図 10 FORVID の使用例
Fig. 10 Example of FORVID.

(3) FORTRAN プログラムに対する解析部分が実行中に存在するので,FORTRAN に似た指令体系を用いて表示を行うことが可能となる.

FORVID は図 9 に示すように(1)簡易エディタ部,(2)前処理部,及び(3)実行表示・指令処理部の3つの機能単位から構成される.

(1) 簡易エディタ部は,ファイルからのソース・テキストの入力,行のそう入・削除及びテキストの表示と,さらにエディタとしての基本機能だけをもたせている.

(2) 前処理部は,(1)で編集されたソース・プログラムを解析し,記号表の作成,文の分類,記憶領域の割付け及び DATA 文に対する初期値の設定を行う.前処理部の入力(ソース・プログラム,200 行程度を目安)と出力(記号表など)はすべて主記憶上にあり,ファイルとの入出力動作がないので起動時間を短くできる.また,前処理部を設けることにより実行中のソース・プログラムの解釈の時間も短縮することができる.

(3) 実行表示・指令処理部は,ソース・プログラムを逐次解釈・実行し,指令リストまたは端末からの指令に基づいて実行位置や変数の値の表示等を行う.

5.3 FORVID の機能

FORTRAN プログラムに対する実行単位は,代入文,call 文等の各文とした.したがって,manual モードでの実行においては,復帰キーを押すごとに1文ずつ実行・表示がなされる.

manual または automatic モードでの表示例を図 10 に示す.これから実行をしようとする文の先頭には記号"−〉"を付け,さらにその前後数行を表示することによりプログラム上の位置付けを明確にしている.これは PASCAL と違って FORTRAN の場合は1行1文を原則とするからである.

FORVID と PASVID の共通の端末指令を表3に示す.FORVID では,インタプリタという特徴を生かして FORTRAN と同様な文を指令として使用することもできるがここでは省略する.

6. 検 討

ここでは演示システムに関して，実現技法と教育システムの2つの観点から検討する．

SIM 960 は，仮想的な裸の計算機を提供するということのほか，学習者にシミュレータそのものも理解させたいという配慮から初版は ACOS-6 PASCAL (約 8,000 行) を用いて開発した．その後主たる目的を実行演示に移し，さらに blind モードでの実行速度の向上 (4～5倍) と機能の拡張を図るためシステム記述言語Bで書き直した版 (約 4,500 行) が実働中である．課題は，実行制御の位置をソース・プログラム上で行うことである．

PASVID は SIM960 の経験を踏まえ，プリコンパイラの効率を主として考慮し言語Bを用いて実現した．大きさは言語Bで約2,500行，PASCAL プログラム・テキスト中にそう入される手続き等は，20行程度のプログラムに対して約250行である．PASCAL には中断キーの取り扱い等システム記述性にいくつかの問題点があり，さらに TSS の負荷が増大するとコンパイル終了までに予想以上の時間を要するという結果になっている．したがって，高負荷時の授業での使用には多少工夫が必要である．

FORVID は前二者の経験を生かし，インタプリタ方式で主として FORTRAN (約5,000行) と一部アセンブラを用いて作成している．教育以外の使用と他システムへの移植についても一応考慮を払っている．

我々は，TSS 環境において使用するプログラム実行演示システムを，3つの代表的な言語に対して，くしくも言語処理系の伝統的作成技法であるコンパイラ，プリコンパイラ及びインタプリタ方式で実現を試みる結果となった．どの方式にも一長一短があるので，使用環境，言語の特徴，開発工期，拡張性及び移植性といった点を考慮して採用しなければならない．これまでの経験によると，TSS 環境における実行演示という目的に限っていえば，そのオブジェクトが速い高級言語を用いインタプリタ方式で作成したものが移植も可能でかつ実用的であると考えている．これは，大規模でない TSS においては特に，多人数使用時のファイルの多用とコンパイラの速度が応答時間に大きな影響を及ぼすからである．

次に，教育システムとしての観点から考察してみる．情報処理教育等を実施する場合，TSS による形態は学習者主導形の個別学習に適しており，バッチによる形態は教師主導形の集団教育が主体となる．演示システムは，教師が演示を通じて学習者の反応を確かめ，全体の理解度を高めながら授業の展開を行うという伝統的一斉授業の良さを継承することによって，個別学習を補完するものとして位置付けられる．

教師は TSS 端末を使うことができさえすれば，従来の黒板や OHP の延長として演示装置上で，学習者に問題を解かせたり，学習結果や教師が前もって計算機内に用意した教材を動的に提示したりすることが容易にできる．さらに，教師に多少の準備期間とスタッフが与えられれば，VTR 等視聴覚機器の採用も可能である．その結果，TSS による教育は，「できる者とそうでない者の差が著しくなった」，「個人指導的になり教師の負担が重くなった」，「端末数が不足して困る」等現場教師から寄せられる問題点はある程度解消できるのではないかと思われる．

一方学習者にとっては，プログラムの実行過程が動画を見るように分かるという経験は，教育の初期の段階では従来と違って非常に印象的である．さらにプログラム等自分の作品が人前で演示され，教師や仲間から批評されるという体験を通じて，他人へ情報を伝達するために表現の仕方を考え，良い作品を作ろうという意識を高揚する契機とすることができる．ただし個別学習においては，プログラムの実行過程等を体験と推察力に基づいて考えることが学習者の情報処理に対する理解を深めているという観点からは，この種のシステムの使用に際しては十分な教育的配慮が必要であろう．

演示システムが定着し，その評価を広く得るためには，計算機に蓄積された日本語や図形を含む教材の演示も容易となり，情報処理機能と十分に大きな画面を有する低価格の提示装置の普及が不可欠である．

7. おわりに

以上，TSS 環境において使用するプログラム実行演示システムの構成と機能及び問題点等について述べた．

このシステムは教育用としてばかりでなく，使い方によってはプログラマの開発支援の有力な道具としてその効果を発揮することができる．また，プログラムの動的振舞を視覚という観点から解析することにより，プログラムの構造や良否に関して興味ある結果が得られるのではないかと考えている．

我々はこのシステムを通常のモノクロ CRT 端末で

開発してきた．今後，演示システムをカラー表示機能を有するスクリーン・ディスプレイ装置を用いて豊富な教育実践を積み重ねることによって，TSS 端末が行形式から画面形式へ発展してゆく過程で画面エディタ等使いやすい機能が実現されてきたように，小画面から大画面，モノクロからカラーへの過程で，視覚情報の提示に関して新たな知見が得られることを期待している．

最後に，本研究に関して貴重な示唆を頂いた九州大学工学部吉田将教授に謝意を表します．

参考文献

1) たとえば，鈴木：情報処理能力を育てる（9），教育と情報，第270号，pp. 54-60 (1980).
2) Donovan, J. J. and Madnick, S. E.: Software Project, p. 509, McGRAW-HILL, New York (1977).
3) Kline, R. B., Hamor, G. D., Krause, K. L. and Druffel, L. E.: Visual Demonstration of Program Execution, SIGCSE Bulletin, Vol. 10, No. 1, pp. 16-18 (1978).
4) 樽美，宇津宮，荒牧：計算機理解援助用シミュレータ SIM960，信学技報，ET 79-4, pp. 27-32 (1979).
5) たとえば，金山：アセンブリ プログラム入門，p. 300，近代科学社 (1977).
6) Jensen, K. and Wirth, N.: Pascal User Manual and Report 2nd Edition, p. 167, Springer-Verlag, New York (1978).
7) 樽美，宇津宮，荒牧：PASCAL プログラム実行演示システム，九大情報処理教育センター広報，Vol. 3, No. 2, pp. 2-16 (1980).
8) 牛島，江嶋：PASCAL プログラム輪郭作成プログラムの実現とその移し換えについて，情報処理学会論文誌，Vol. 22, No. 1, pp. 29-35 (1981).
9) 樽美，宇津宮，荒牧：FORTRAN プログラム実行演示システムの設計，電気四学会九支連大講演論文集，p. 34 (1980).
10) Meissner, L. P. and Organick, E. I.: FORTRAN 77, p. 500, Addison-Wesley. Reading, Massachusetts (1980).

(昭和56年2月16日受付)
(昭和56年7月13日採録)