TSS の下でのアセンブリ言語および計算機構造の教育援助システム†

阿草清滋†† 大川佳紀††
大野豊†† 田畑孝一†††

現在，高級言語による計算機利用が主流となってきているが，アセンブリ言語を用いたプログラミング教育の必要性も残されている．たとえば，システムの性能向上，特殊入出力装置の制御などのためにアセンブリ言語での記述が要求される．

また，計算機の構造の理解にはアセンブリ言語での計算機動作指令と，それに従った計算機の内部状態（レジスタやメモリの内容）の遷移をたどることが有効である．

これらの教育のために，TSS 環境下で学生があたかも1台ずつの計算機が与えられ，自由にそのアセンブラと計算機パネルを用いて実習できるシステムを設計・製作した．このシステムの特徴は

（1） 教師が教育したい計算機を定義できる．

（2） 学生は TSS の CRT 端末を計算機パネルとしてまた，アセンブリ・リスト表示端末として使え，両者を対応づけながら，計算機動作の過程をトレースできる．

（3） 仮想ディスク，仮想テープ，仮想テレタイプなどの入出力装置が付加されており，入出力制御の実習やファイル管理の実習などもできる．

などが挙げられる．

本システムの仮想ファイルは仮想記憶空間にとられるため，仮想計算機の入出力は OS のページングに頼り，TSS の性能に悪影響を与えない．

1. はじめに

計算機利用の拡大に伴い，高級言語によるソフトウェア開発が主流となった．これはソフトウェア生産性，ポータビィリィ，保守性など高級言語の良さが認められてきたことと，優れた言語，コンパイラの開発でオブジェクト効率がアセンブリ言語のそれに比して遜色のないものとなったことによる．しかし，ある範囲のプログラマにとってはアセンブリ言語を用いたソフトウェア開発が依然として要求されている．

また，計算機自体を研究の対象とする分野を専攻する学生の教育には，単に高級言語を用いたプログラミング手法の教育だけでは充分でなく，計算機の構造をも教育する必要がある．このためにはアセンブリ言語を用いてプログラムを記述し，そのプログラムの実行の様子を，計算機の内部状態の変化系列として見せることが役に立つと考えられる．

このようなアセンブリ言語の教育には次のような問題点がある．

（1） 教育すべき対象の計算機は一つとは限らず，必要とされる種類だけのハードウェアを用意することは難しい．

（2） 複数の学生を効率よく教育するためには複数個のハードウェアが準備されることが望まれる．

（3） 計算機の構造そのものを教育するにはオペレーティング・システムは不必要であり，オペレーティング・システムが存在すると入出力命令やその他の特殊な命令が使えないことすら多い．しかし，オペレーティング・システム無しにはアセンブリ言語によるプログラム開発も困難になる．

我々はこれらの問題を解決し，多人数の学生を対象に教師の希望する計算機のアセンブリ言語，構造を教育するための支援システムを設計・製作した．

本システムは大型計算機の TSS の下で動作し，教師の定義した計算機のアセンブリ・プログラミングとそのデバッグ，実行を管理するシステムである．本システムの特徴は以下に述べられる．

（1） 教師の希望する計算機を教えることができる．これは実存する計算機のアーキテクチャを持つ必要はなく，教育用に簡単化した計算機を定義できる．

（2） 学生は各個人用の計算機を与えられているかのように計算機実習ができる．すなわち，TSS の CRT

---

† A Supporting System for Teaching Assembly Language and Computer Architecture in TSS Environment by KIYOSHI AGUSA, YOSHINORI OHKAWA and YUTAKA OHNO (Department of Information Science, Kyoto University) and KOICHI TABATA (Educational Center for Information Processing, Kyoto University).

†† 京都大学工学部情報工学科
††† 京都大学情報処理教育センター

端末上に内部レジスタ表示用ランプや制御スイッチが並んで表示され，ミニコンピュータのパネルを操作するがごとく，各人のプログラムをデバッグできる．

（3） 入出力制御の演習が可能である．各仮想計算機にはプログラム実習に必要とされる最小限度の入出力装置が用意されている．これにはランダム・アクセスの2次記憶装置(磁気ディスク装置を想定)，シーケンシャル・アクセスの2次記憶装置（磁気テープ装置を想定)，キーボード入力，CRT 装置出力(システム・コンソール装置を想定）が含まれる．

（4） アセンブリ・プログラム・リストの管理・表示やメモリ・ダンプなどのデバッグ機能を持つ．

（5） 教師の計算機定義のためのデバッグ機能を有する．計算機定義に従って動作するシミュレータの動きをトレースでき，定義者の意図通りの定義がなされているか調べることができる．

本論文ではこの TSS 用計算機教育援助システム—CUTE (Supporting System for Computer Understanding in TSS Environment) の概要，ソフトウェア構成を示すとともに，簡単な計算機の定義例とその定義された計算機での実習例を示す．

2. システムの概要と機能

2.1 システム概要

TSS 用計算機教育援助システム CUTE は大型計算機の TSS 環境下で，教師が定義した仮想計算機（16ビット）を用い，学生の計算機理解，アセンブリ言語実習などに利用される．本論文では以下，仮想計算機の定義を行うユーザを“教師”,定義された仮想計算機を用いるユーザを“学生”と呼ぶ．

教師は学生に教えるべき計算機の構造を熟知している．このため，仮想計算機の動作をシステムの用意した記述言語で記述できる．この記述言語はハードウェアの接続関係レベルまで記述できる能力を持つ必要はなく，その機能記述ができればよい．CUTE ではマイクロ・プログラマブル仮想計算機を用意し，教師がこのマイクロ・プログラムを記述することで，かなり自由な仮想計算機を定義できるようにした．すなわちマイクロ・プログラム方式の計算機の特徴の1つである計算機の制御構造の変更の容易さ（1）を生かし，

1つのマイクロ・プログラマブル仮想計算機のハードウェア・シミュレータ・プログラムを用意することで，教師の定義した仮想計算機のシミュレーションを可能とした．このマイクロ・プログラマブル仮想計算機の概念を図1に示す．

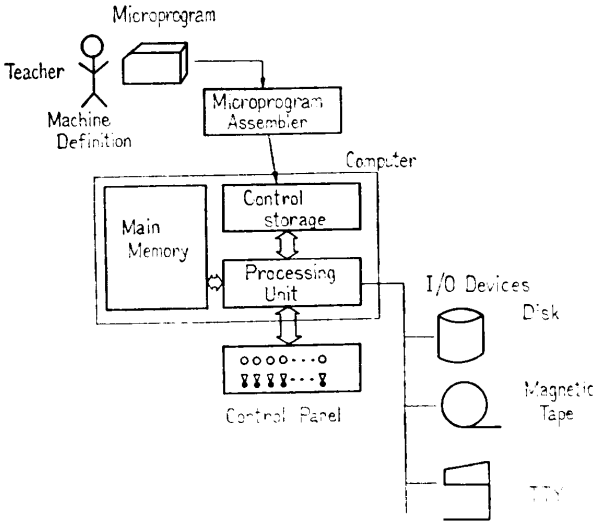

図 1 仮想計算機システムの概念図

Fig. 1 Microprogrammable Virtual Computer System.

教師は必要に応じて複数個の計算機を定義でき，学生にその1つを選択させて教育することもできる．図2にはシステム利用の概念図が示される．図2において教師 T_1 は仮想計算機 X，Y を定義しており，学生 S_2，S_3 は X を，学生 S_1 は Y を使用している．それと同時に，学生 S_4，S_5 は教師 T_2 により定義された仮想計算機 A を使用している．

2.2 教師用機能

CUTE を用いて教育を行う教師は，教育用計算機のアセンブラとハードウェアを定義しなければならない．CUTE には汎用アセンブラ，汎用シミュレータが用意されており，それらの定義を簡単に行えるようになっている．

アセンブラの作成のためには，アセンブリ言語定義言語 (ASDEL : Assembly Language Definition Language) を用いて，アセンブラの定義を行う．汎用アセンブリ・システムの中の定義記述解析モジュールはこの定義から，命令ニモニック，オブジェクト・コード，各種予約語などを処理し，アセンブラ定義テーブルを作成する．このテーブルを用いて学生の書いたソース・プログラムがアセンブルされる．

仮想計算機ハードウェアの定義のためには，マイクロ・プログラム・アセンブラが用意されている．教師はエディタを用い，マイクロ・プログラムのソースを作成し，マイクロ・プログラム・アセンブラにより，マイクロ・プログラム・オブジェクト・コードと付加

情報を含んだデータセットを得る．このデータセット名は仮想計算機名と対応づけられており，汎用ハードウェアシミュレータにより用いられる．

2.3 学生用機能

学生は TSS 用のエディタ，仮想計算機用アセンブラ，仮想計算機を用いて実習を行う．

学生はソース・プログラムと仮想計算機のデータセット名を入力し，システムを起動する．CUTE は教師の定義したアセンブラ定義テーブルを参考に，学生のソース・プログラムをアセンブルする．そのオブジェクト・コードとアセンブリ・リスト出力をパネル・ディスプレイ・コントローラに渡し，学生の指示待ちとなる．学生はメモリ内容参照・更新，レジスタ内容参照・更新，単一ステップ実行など一般のミニコンピュータと同様のパネル制御機能を用い，リストを参照しながらプログラムの実行，デバッグを進める．

入出力装置として仮想ディスク，仮想テープ，テレタイプが用意されている．これらは仮想記憶空間上にとられるが，システムの利用終了時に実際のディスク装置にセーブされる．次の利用時にはこの内容はリストアされ，あたかも実際の装置にテープ／ディスクを装填したかのようになる．これにより，学生プログラム開発結果の蓄積が可能で，またオーバレイなども行える．仮想ファイルを仮想記憶上にとることによりセンタの OS は最も適当と考えられるページングアルゴリズムで仮想ファイルを管理できる．もし各学生の仮想ファイルを実際のデータセットに割当てると，仮想ファイルの利用頻度が非常に高いとき，ディスクのヘッド競合，チャネル競合などでシステム性能を下げるかも知れない．これをページングに委ねることにより，OS にとって最適な物理的なレコード位置に仮想ファイルの領域を確保でき，システム性能への影響を最小化できる．

3. ソフトウェア構成

図3 に CUTE のソフトウェア構成を示す．CUTE は次の3つのサブシステムから構成されている．

1．汎用アセンブリ・システム

2．汎用ハードウェア・シミュレーション・システム

3．パネル・ディスプレイ・コントローラ

さらに，1はアセンブラ定義記述解析モジュールと汎用アセンブラに分かれ，2はマイクロプログラム・

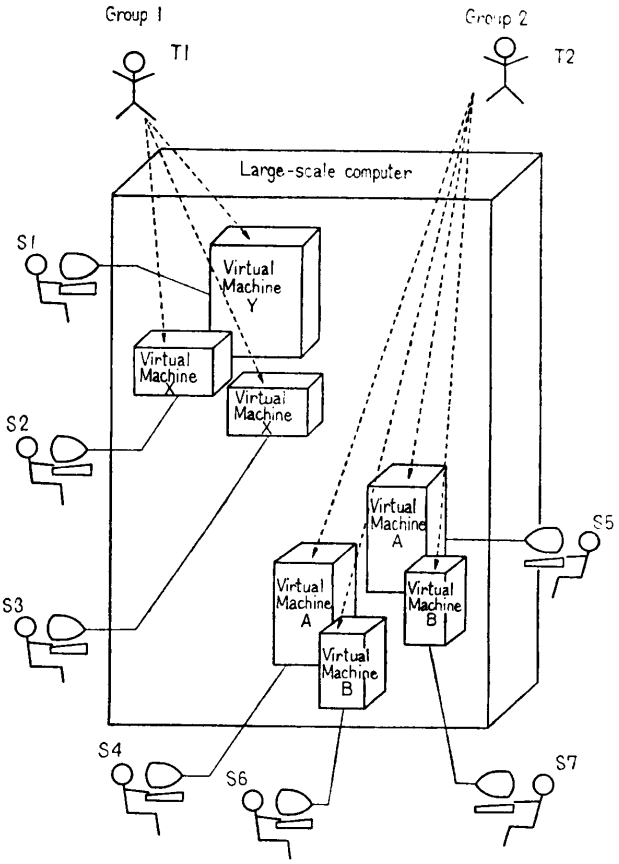

図 2 システム利用の概念図
Fig. 2 Conception of system.

アセンブラと汎用ハードウェア・シミュレータに分かれている．また，汎用ハードウェア・シミュレータはパネル・ディスプレイ・コントローラの副プログラムになっている．

モニタには，仮想計算機定義用モニタと仮想計算機動作用モニタの2つがある．

3.1 汎用アセンブリ・システム

3.1.1 アセンブラ定義記述解析モジュール

仮想計算機のアセンブラ定義は，アセンブリ言語定義言語 (ASDEL) により記述される．アセンブリ言語の仕様のうち，書式およびアセンブラ擬似命令は**図4**および**表1**に示すように固定されており，ASDEL により定義できる可変部分は，オブジェクト・コードの出力形式，ニモニック・コード，オペランド部の構成と特殊記号，および予約語である．ASDEL の言語仕様を**図5**に BNF 記述で示す．アセンブラ定義解析モジュールは，ASDEL によるアセンブラ定義記述を構文解析し，アセンブラ定義テーブルを出力する．

3.1.2 汎用アセンブラ

汎用アセンブラは，学生の指定した仮想計算機のアセンブラ定義テーブルを読み込み，それを参照しながら学生のプログラムをアセンブルする．その結果，汎用アセンブラは，仮想計算機の主記憶上にオブジェクト・コードを出力し，3.3 のパネル・ディスプレイ・コントローラにアセンブル・リストを引き渡す．

3.2 汎用ハードウェア・シミュレーションシステム

3.2.1 ハードウェア論理定義

仮想計算機のハードウェア論理は，機械語の実行内容をマイクロプログラムの形で記述することにより定義される．マイクロプログラムはマイクロプログラム・アセンブラによりアセンブルされ，オブジェクト形式になる．マイクロプログラムのステートメント書式を図 6 に示す．

① LL：ラベル(16進2桁またはbb*)

ラベルが限定されるとOP-TABLEの対応する欄に，このラベルの付けられているマイクロ命令のロケーションが登録される．

② mmm：ニモニック・コード

マイクロ命令のニモニック・コード(現在45種)，またはマイクロプログラムの終了を示す‘END’を指定する．

③ C および V：状態レジスタの C (carry) ビット，および V (overflow) ビットのセット／リセット指定

④ operands：オペランドの並び

オペランドの記述形式を図 7 に BNF 記述で示す．

⑤ ϕ：don't care

第1カラムが‘*’の場合，コメント行となる．

3.2.2 汎用ハードウェア・シミュレータ

汎用ハードウェア・シミュレータは，制御記憶にロードされたマイクロプログラム・オブジェクトを実行

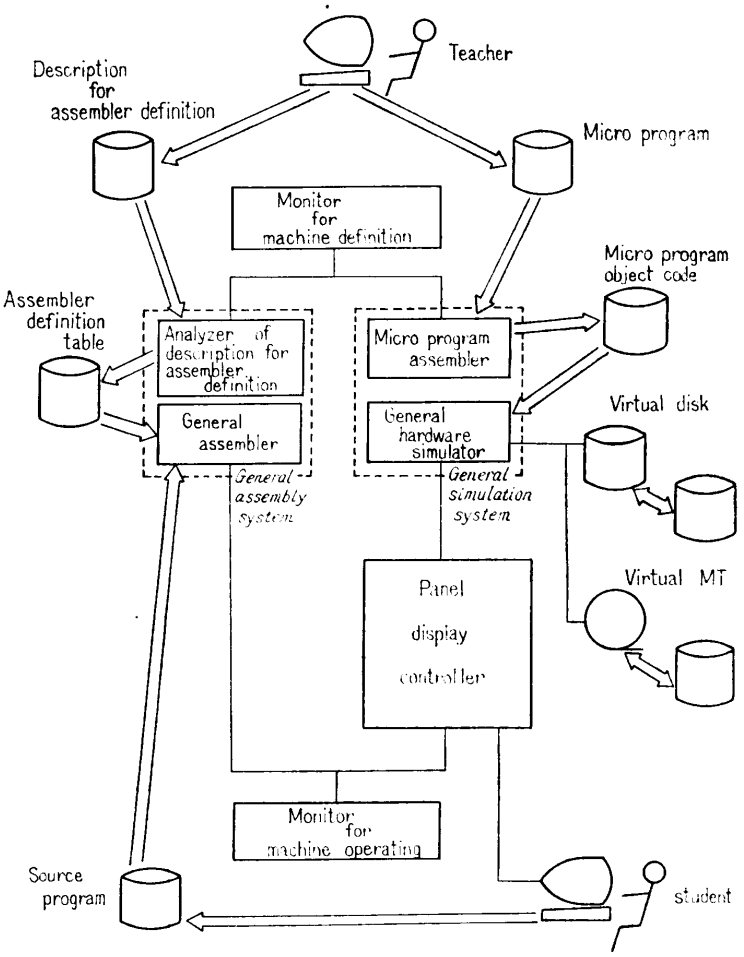

図 3 システムのソフトウェア構成

Fig. 3 System configuration.

表 1 アセンブラ擬似命令

Table 1 Instruction for assembler.

ORG statement (ロケーション・カウンタはオペランド欄で指定される値にセットされる)
mnemonic: ORG
operand: numeric value/code

EQU statement (ラベルは必須であり，式の値がラベルの値となる)
mnemonic: EQU
operand: expression

DC statement (16 ビットの領域（語境界）に式の値が格納される)
mnemonic: DC
operand: expression

DS statement (オペランド欄で指定された大きさの領域が確保される)
mnemonic: DS
operand: numeric value

END statement (このステートメント以後のステートメントは無視される)
mnemonic: END
operand: blank

START statement (このステートメント以前のすべてのステートメントは無視される)
mnemonic: START
operand: program name

Column	1	...	6	7	8	.....	13	14	15	...	32
Statement	label				mnemonic			operands			

図 4 アセンブリ言語の書式

Fig. 4 Format for assembly language.

* b は空白 (blank) を表わす．

```
〈アセンブラ定義記述〉=〈CODE ブロック〉〈FLD ブロック〉
  〈TYPE ブロック〉〈EQU ブロック〉
〈CODE ブロック〉=CODE 〈コード出力形式〉
〈コード出力形式〉=OCT|HEX
〈FLD ブロック〉=FLD 〈フィールド記述の並び〉; END
〈フィールド記述の並び〉=〈フィールド記述〉{; 〈フィールド記述〉}
〈フィールド記述〉=〈フィールド名〉〈フィールド位置〉:
  〈フィールド形式〉
〈フィールド名〉=〈名前〉
〈フィールド位置〉=(〈先頭〉-〈末尾〉)
〈先頭〉=〈10進値〉
〈末尾〉=〈10進値〉
〈フィールド形式〉=〈形式1〉|〈形式2〉|〈形式3〉
〈形式1〉=OPC〈ニモニック定義式の並び〉
〈ニモニック定義式の並び〉=〈ニモニック定義式〉
  {, 〈ニモニック定義式〉}
〈ニモニック定義式〉=〈ニモニック〉=〈コード〉
〈ニモニック〉=〈名前〉
〈コード〉=X'〈16進値〉'|O'〈8進値〉'|B'〈2進値〉'
〈形式2〉=SPC〈特殊文字定義式の並び〉
〈特殊文字定義式の並び〉=〈特殊文字定義式〉{, 〈特殊文字定義式〉}
〈特殊文字定義式〉='〈特殊文字〉'=〈コード〉
〈特殊文字〉=¥|#|@|=|"|. |%|; |:|&|1|>|<|-|?
〈形式3〉=NUM|NUM 〈SIGN〉
〈SIGN〉=SIGN
〈TYPE ブロック〉=TYPE〈タイプ記述の並び〉; END
〈タイプ記述の並び〉=〈タイプ記述〉{; 〈タイプ記述〉}
〈タイプ記述〉=〈フィールド名1〉*|〈フィールド名1〉
  :〈フィールド名2の並び〉
〈フィールド名2の並び〉=〈フィールド名2〉**{, 〈フィールド名2〉}
〈EQU ブロック〉=EQU〈予約語定義式の並び〉END
〈予約語定義式の並び〉=〈予約語定義式〉{, 〈予約語定義式〉}
〈予約語定義式〉=〈ラベル〉=〈コード〉; |〈ラベル〉=〈10進値〉;
〈ラベル〉=〈名前〉
〈名前〉=〈英字〉{〈英数字〉}⁸
〈英数字〉=〈英字〉|〈10進桁〉
〈英字〉=A|B|C|……|X|Y|Z
〈10進値〉=〈10進桁〉{〈10進桁〉}
〈16進値〉=〈16進桁〉{〈16進桁〉}
〈8進値〉=〈8進桁〉{〈8進桁〉}
〈2進値〉=〈2進桁〉{〈2進桁〉}
〈10進桁〉=0|1|2|…|9
〈16進桁〉=0|1|2|…|9|A|B|C|D|E|F
〈8進桁〉=0|1|2|…|7
〈2進桁〉=0|1
```

*) フィールド名1はOPC属性をもつフィールド名
**) フィールド名2は SPC 属性あるいは NUM 属性をもつフィールド名

図 5 ASDEL の BNF 表記

Fig. 5 BNF of ASDEL.

```
Column  1  2  3  4  5  6  7  8  9  10  ...  24  25  ...  72
Statement L  L     m  m  m  C  V      operands     comment
   L  L : label (hexadecimal)
m  m  m : mnemonic code
   C  V : CCRc, CCRv set/reset specification
```

図 6 マイクロプログラム・ステートメントの書式

Fig. 6 Format for micro program statement.

```
〈operands〉=〈register〉{, 〈register〉|, 〈immediate〉}²|
        〈relative〉{, 〈condition〉}¹
〈register〉=〈GR〉|〈CCR〉
〈GR〉=〈GR No.〉|〈GR No.〉@
〈CCR〉=CCR
〈GR No.〉=0|1|2|…|30|31
〈immediate〉=〈decimal value〉|〈hexadecimal value〉
〈decimal value〉=#〈digit〉{〈digit〉}
〈hexadecimal value〉=$〈hex digit〉{〈hex digit〉}³
〈relative〉=|*+〈relative value〉|*-〈relative value〉
〈relative value〉=〈digit〉{〈digit〉}
〈condition〉=〈test〉{, 〈test〉}
〈test〉=〈CCR bit〉|¬〈CCR bit〉
〈CCR bit〉=〈hex digit〉|〈flag〉
〈flag〉=I|V|N|Z
〈digit〉=0|1|2|…|9
〈hex digit〉=0|1|2|…|9|A|B|C|D|E|F
```

図 7 マイクロ・プログラムのオペランド記述形式

Fig. 7 BNF of microprogram operand.

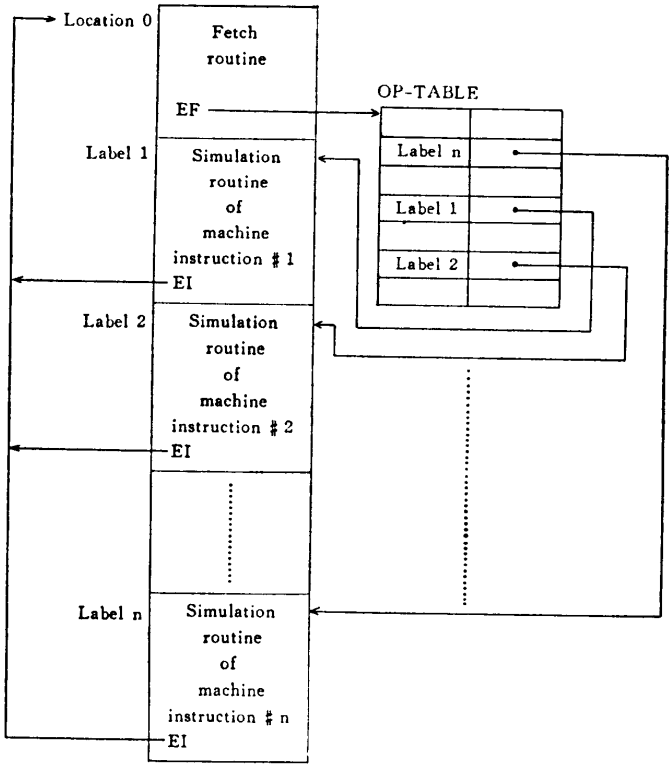

図 8 マイクロプログラムの構成と実行

Fig. 8 Execution of micro program.

することにより，仮想計算機の機械語の実行をシミュレートする．マイクロプログラムは通常，図8のような構成をとる．

fetch ルーチンは主記憶上にある仮想計算機の機械語を読み出し，デコードを行う部分である．EF 命令で，OP-TABLE の内容による間接分岐を行い，対応する各機械命令のシミュレート部に移る．各機械語シミュレート部では，仮想計算機の機械命令の機能に対応した処理を行い，最後に EI 命令により，fetch ルーチンに制御を戻す．この過程を繰り返し実行することにより，仮想計算機のハードウェア・シミュレーションを行う．

パネルランプ1	アセンブル・リスト表示画面
パネルランプ2	・ステートメント番号
実行状態表示部	・アセンブル・エラー・コード
TTY 用 入出力画面	・メモリ・ロケーション ・機械語コード ・ソース・ステートメント

図 9 表示画面構成
Fig. 9 Construction of Display panel.

3.3 パネル・ディスプレイ・コントローラ

パネル・ディスプレイ・コントローラは，TSS端末のビデオ・データ・システムを制御して，仮想計算機のレジスタや主記憶の内容，CPU 状態，およびプログラムのアセンブル・リストを表示する．表示画面の構成は図9のようなものである．

図9で，パネル・ランプに表示するものは，パネル・モードにより異なる．パネル・モードには，主記憶(MM)モード，汎用レジスタ(REG)モード，および状態レジスタ(CCR)モードがある．ランプは，2進，8進，および16進の3種類で表示する．また，実行状態表示部では，CPU 状態と実行時エラー・コードを表示する．画面左下は，テレタイプライタ入出力用領域であり，さらに，ここよりパネル・ランプへの入力データも送信する．

パネル・ディスプレイ・コントローラは，表2に示した種々のコマンドを用意しており，学生はこれらを使って，随時，仮想計算機のハードウェア状態を認識したり，プログラムの実行制御を行える．

表 2 パネル・ディスプレイ・コントローラのコマンド
Table 2 Command for Panel display controller.

START (PA 1 key)	アドレス・レジスタの指している番地からプログラムを開始させる
STOP (PA 2 key)	実行中のプログラムを停止させる
SI (PA 3 key)	アドレス・レジスタの指している番地にある命令を1つ実行させる
DATA (PF 1 key)	LAMP 1 に入れるべきデータを入力することを指定する
ADDR (PF 2 key)	LAMP 1 に表示されている値をアドレス・レジスタに入れる
REG (PF 3 key)	LAMP 1 に表示されている値 X のレジスタ・モードに変える
CCR (PF 4 key)	パネル・モードを状態レジスタ・モードに変える
READ (PF 5 key)	パネル・モードに従って，ハードウェアの内容をランプに表示する
WRITE (PF 6 key)	LAMP 1 に表示されている内容をパネル・モードで指定されたハードウェアに書きこむ
INIT (PF 7 key)	LAMP 1，LAMP 2 を0にし，パネル・モードをメモリ・モードに変える
UP (PF 8 key)	表示中のアセンブル・リストを1行ロール・アップする
DOWN (PF 9 key)	表示中のアセンブル・リストを1行ロール・ダウンする
NEXT (PF 10 key)	表示中のアセンブル・リストを 22 行（1ページ）ロール・ダウンする
PREV (PF 11 key)	表示中のアセンブル・リストを 22 行（1ページ）ロール・アップする
EXIT (PF 12 key)	パネル・ディスプレイを終了させる

4. システムの使用法

本システムの利用者，すなわち教師と学生は次の4つのコマンド・プロセジュアを用いて，システムを起動する．以下で使用する記号を定義する．

△：1つ以上の空白を意味する．
{,}：選択を意味する．
英文字：キー・ワードである．
—：次行継続．

a．VM△ 仮想計算機名 △{ASM, SIM}△—
データセット名 △{DEBUG, SAVE}
教師が仮想計算機を定義する際に用いる．
ASM はアセンブラ定義を指定する．
SIMはハードウェア論理定義を指定する．
データセット名は，アセンブラ定義記述あるいはマイクロプログラムの格納されたデータセット名を指定する．
DEBUGは入力データと診断メッセージをプリント出力し，定義結果は保存しないことを指定する．
SAVE はリスト出力を抑止し，定義結果を仮想計算機名と関連づけられた名をもつデータセットに出力保存することを指定する．

b．MT△ 仮想磁気テープ名 △{NEW, OLD}
c．DISK△ 仮想磁気ディスク名 △{NEW, OLD}
bとcは学生が仮想計算機の外部記憶装置にボリュームをセットする際に用いられる．
NEW は，セットするボリュームが新規セットするものであることを指定し，OLD は過去に読み書きを行ったボリュームをセットすることを指定する．

d．IS△ 仮想計算機名 △{MT, DISK, MTDISK, —
NONE}△ データセット名
学生が仮想計算機を動作させる際に用いる．
MT, DISK, MTDISK はそれぞれ，プログラム中で，仮想計算機の MT の入出力のみ，ディスクの入出力のみ，MT とディスク両方の入出力を扱うことを指定する．

```
CODE=HEX; /* OUTPUT OBJECT CODE WITH
      HEXADECIMAL */
FLD /* FIELD BLOCK */
  OPCODE (0-4) : OPC /* FIELD WITH OPERATION CODE
         ATTRIBUTE */
    L=X '01', /* MNEMONIC L */
    ST=X '07'; /* MNEMONIC ST */
  SPCH (5-6) : SPC /* FIELD WITH SPECIAL CHARACTER
           ATTRIBUTE */
      #=B '00';
  OPRND (7-15) NUM /* FIELD WITH NUMBER
        ATTRIBUTE */;
END /* OF FLD BLOCK */
TYPE /* TYPE BLOCK */
  OPCODE: SPCH, OPRND;
  /* ASSEMBLY LANGUAGE:     L     #, 100
                           ST     #, 200
       OBJECT CODE       : 0000100001100100
                           0011100011001000 */
END /* OF TYPE BLOCK */
EQU /* EQU BLOCK */
   KEYWORD=X '00';
END /* OF EQU BLOCK */
```

図 10 ASDEL によるアセンブラ定義記述例

Fig. 10 Example of description for assembler definition with 'ASDEL'.

NONE は外部記憶を使用しないことを指定する．

データセット名は，アセンブリ・プログラムの格納されたデータセット名を指定する．

4.1 仮想計算機の定義例

定義するマシンとして，ミニコン Hitac 10II を取り上げる．教師はエディタにより，次のような名前と内容のデータセットを作成したとする．(**図 10, 11** 参照)

H10. ASM : ASDEL によるアセンブラ定義記述

H10. SIM : マイクロプログラム

教師は，次の手順でマシン定義を行う．マシン名は 'TEN2' とする．

① VM TEN2 ASM H10. ASM DEBUG

を入力し，アセンブラ定数記述に誤りがないかを診断メッセージによりチェックする．誤りがあれば，訂正し ① を繰り返す．誤りがないことを確認すれば ② へ移る．

② VM TEN2 ASM H10. ASM SAVE

を入力し，アセンブラ定義テーブルを TEN2 と関連づけられた名をもつデータセットに出力保存する．TEN2 のアセンブラ定義完了である．

③ VM TEN2 SIM H10. SIM DEBUG

を入力し，マイクロプログラムに誤りがないかをマイクロプログラム・アセンブラの診断メッセージによりチェックする．誤りがあれば，訂正し ③ を繰り返す．

```
SOURCE STATEMENT
LL MMMCV.....OPERANDS.....
***** HITAC 10 SUBSET *****
    L       4, 1            IR←(MM)
    FEX     4, ¥ FE 00, 16      OP'
    FEX     4, ¥ 01 FF, 17      ADR
    LI      18, ¥ 0 E 00
    AND     1, 18, 3        CURR. PG
    EF      16
    FEX     4, ¥ FF 00, 16
    EF      16
    STB     CCR, #1
* #### FETCH END # # # # # # # # # #
*
*.....HALT.....
6F INC  1                       PC+1
   HLT
*
*.....ADD, I .....
OB OR   3, 17, 3        AD. MODFY
   L    3, 3             INDIRECT
   L    2, 3            MBR←(MM)
   INC  1                   PC+1
   A  C 0, 2, 0
   LCB  CCR, #0, # 12
   EI
*
*.....SUB, I .....
OF OR   3, 17, 3        AD. MODFY
   L    3, 3             INDIRECT
   L    2, 3            MBR←(MM)
   INC  1                   PC+1
   S  C 0, 2, 0
   BC   *+3, C.....THEN...
   RSB  CCR, #0        ←ELSE
   EI
   STB  CCR, #0.....
   EI
*
*.....LOAD, I .....
07 OR   3, 17, 3        AD. MODFY
   L    3, 3             INDIRECT
   L    2, 3            MBR←(MM)
   INC  1                   PC+1
   TR   2, 0            MBR→AC
   EI
*
*.....STORE.....
1D OR   3, 17, 3        AD. MODFY
   INC  1                   PC+1
   TR   0, 2            AC→MBR
   ST   2, 3           MBR→(MM)
   EI
*
*# # # # # # # # # # # # # # # # # # # # # #
*
## END
```

図 11 マイクロプログラム記述例

Fig. 11 Exampe of microprogram.

誤りがなければ ④ へ移る．

④ VM TEN2 SIM H10. SIM SAVE

を入力し，マイクロプログラム・オブジェクトを TEN2 と関連づけられた名をもつデータセットに出力保存

する．TEN 2のハードウェア論理定義完了である．

以上で，仮想計算機 TEN 2が定義された．

4.2 仮想計算機の使用例

4.1 で述べた手順で定義された仮想計算機 TEN 2を，学生が動作させる手順を述べる．学生は，システム起動前に次のデータセットを作成したとする．

EX Ø1：TEN 2のアセンブリ・プログラム

また，プログラム中で磁気テープを使用するものとする．

① MT VOL 1 NEW

を入力し，VOL 1 というボリュームを新規マウントする．

② IS TEN 2 MT EX Ø1

を入力し，TSS端末画面にパネルが表示されるのを待つ．

③ TSS端末画面にパネルが表示されれば，学生は**表3** の各種コマンドを用いて，演習・実験を行う．

たとえば，プログラムを実行したい場合，まずDATA コマンドとADDRコマンドによって，アドレス・レジスタにプログラムの先頭番地をセットし，次にSTART コマンドによって実行させればよい．また，主記憶のある番地の内容を参照したいなら，まずDATA コマンドと ADDR コマンドにより，アドレス・レジスタに参照番地をセットし，次に READ コマンドでその番地の内容をランプ表示させればよい．**図12** にその時の表示の模様を示す．左下に学生が入力したØ1ØØ の表示，LAMP 1にアドレス，LAMP 2 にメモリの内容が見える．

④ 演習・実験を終わりたいとき，EXIT コマンドを入力する．

磁気テープの内容の実際の保存は，システムが自動的に行うので，学生は関知しなくてもよい．

なお，学生が磁気ディスクを使用した場合も同様に，システムがその内容を保存する．

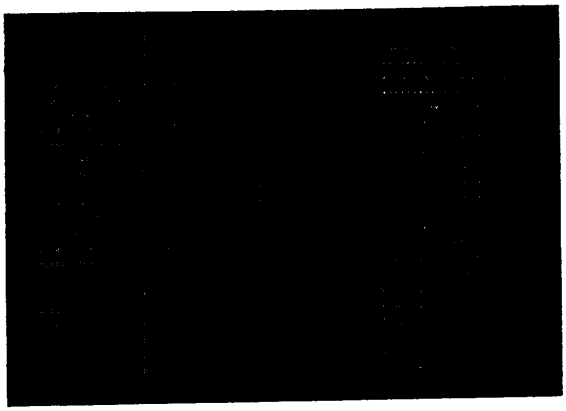

図 12 実習中の表示例

Fig. 12 Example of Display Screen in use.

5. おわりに

マイクロプロセッサの低価格化により，多くのパーソナル・コンピュータやワンボード・コンピュータを用いて計算機教育，アセンブラ教育が行われることもある．しかし，マイクロプロセッサのアーキテクチャは必ずしも標準的な計算機の構造を持っているとは限らず，計算機教育に適さないこともある．また，数多くのハードウェアの管理・保守の困難さ，ファイル管理機能の弱さなどの問題点があろう．

我々の設計・製作した計算機教育システム CUTE は，TSSの利用により多人数の学生が各個人用の仮想計算機を持つことを許し，また教師の希望する仮想計算機が用意できるので，計算機教育・アセンブリ言語プログラミング実習にきわめて有効であろう．特に，CRT 端末にミニコンピュータのパネル風のものを用意したため，学生は各個人専用の計算機が与えられたごとく実習ができ，レジスタやメモリ内容の参照を通して計算機の構造，動作を理解できる．また，基本的な入出力装置が用意されており，オペレーティング・システムの製作のような課題も与えることができる．

本システムは京都大学情報処理教育センターの Hitac M-180，VOS 3 上に主として PL/I を用いて作成され，そのプログラム・ステップ数は約6,000 ステートメントである．画面管理部はアセンブリ言語で書かれ約1,000 ステートメントである．ハードウェア・シミュレート部はマイクロ・プログラム方式のため多階層となり，シミュレート速度の低下が懸念されたが，画面表示の更新速度がそれ程高速でないため，シミュレーション速度は問題にならない．

現在はセンタ内端末の画面制御機能を用いているため，センタ外の一般の端末からは本システムを利用することができない．現在，センタ外端末による実習も可能とすべく，パネル・ディスプレイ・コントローラ部の変更を進めている．また，アセンブラ定義とハードウェア定義との間の矛盾のチェック，本システム利用のための案内機能などの追加について検討している．

謝 辞 本システムの設計・作製に当っては，京都大学大学院工学研究科の新実治男，中島 浩，浦野 郎の三氏の協力に負うところが大であり，心から感謝致

します．また，システム作製に色々と便宜を図って頂いた京都大学情報処理教育センターの方々に感謝致します．

参考文献

1) 萩原 宏：マイクロ・プログラミング，産業図書 (1977).
2) 阿草清滋，大川佳紀：計算機教育用システムについて，京都大学情報処理教育センター，広報 No. 2 (8月1980).
3) 大川佳紀，阿草清滋，田畑孝一，大野 豊：計算機教育用システム，情報処理学会，第 21 回全国大会論文，5L-7.

(昭和55年5月9日受付)
(昭和55年7月17日採録)