CA 23 − 2
(1976. 12. 15)

中型計算機における マイクロ・プログラム・サポート・ソフトウェア

平山正治・扇岡 璋・伊藤貴康
(三菱電機株式会社中央研究所)

1. まえがき

マイクロ・プログラム制御計算機の機能を生かし，各種のプログラムをファームウェア化して高速処理を行なえるようにする目的から，近年，ミニ・コンピュータの領域では，マイクロ・プログラムをユーザに開放するとともに，種々のマイクロ・プログラム作成用のサポート・ソフトウエアを提供している機種が多く，一般ユーザにも簡単にマイクロ・プログラムが利用できる状況にある。

しかし，中型以上の計算機に於いて，マイクロ・プログラムを開放する場合。

(1) マイクロ・プログラムの誤りによりソフトウェアの正常動作が保証されなくなり，マシンダウンを引き起こすなどメンテナンスが困難である。

(2) マルチ・プログラミングの条件下では，多数のジョブが同時に同じWCS(書き込み可能制御記憶)領域を使用する可能性があり，WCSの管理が必要となる。

等の問題点があり，ユーザによるマイクロ・プログラムの有効利用が難かしい状況にある。

上記問題を解決し，マイクロ・プログラムの開発，利用を効率良く行なうためのサポート・ソフトウエアについて検討し，トレーサとマイクロ・モニタを中心とする実験システムを汎用中型計算機MELCOM-COSMO 500 (以下M-500と略記する)を利用して開発した。 トレーサは従来のマイクロ・プログラム・シミュレータに対応するものであるが，シミュレーションの実行速度を上げ，かつ，システムをコンパクトに実現する目的から，マイクロ命令の実行は計算機自身のハードウェアを用いて行なわせ，シーケンスの制御，会話機能，および誤りチェック等の機能のみをソフトウェアで実現している事に特徴がある。 他方，マイクロ・モニタは上記の問題点(2)を解決するために実験的に開発したものである。マルチ・プログラミングの環境下におけるファームウエアやWCSの管理は，本来OSの機能の一部と考えられるが，特に高速性を要求されることから，ここではマイクロ・プログラムを用いてファームウエアのダイナミック・リンキングを行ない，WCSの管理に付随するオーバヘッドを可能なかぎり抑えることを目的としている。

以下ではM-500において実験的に開発したトレーサ，およびマイクロ・モニタの機能の詳細を報告する。

2. M-500におけるファームウェア機能

M-500は，システム・マイクロ・プログラム用に6K語(1語32ビット)の制御記憶(ROM)をもつが，これ以外に最大4K語(1語32ビット)のWCSを実装する事が可能であり，これをユーザがファームウェア領域として利用する事ができる。 1マイクロ命令の実行サイクルは300 nsecであり，ファームウェア

表1. ファームウェア化の例

プログラム名	条件	FORTRAN(A)	μ-プログラム(B)	A/B
スレッシュホールド	2048点	92.5 ms	4.5 ms	20.5
ヒストグラム	2048点	90.0 ms	3.0 ms	30.0
4語パック	2048語→512語	215 ms	2.5 ms	86

化により，かなりの高速性能が得られる．　この例を表１に示す．

M-500におけるファームウェアの開発は図１に示すように５個のサポート・ソフトウェアを利用して行なう事ができる．　ここで，マイクロ・モニタ以外の４個のプログラムは，M-500の汎用OS（UPS）の下で１個のユーザ・ジョブとして実行される．マイクロ・モニタはWCS中に常駐して，WCSの管理等，ユーザ・マイクロ・プログラムの実行をサポートする．

各サポート・ソフトウェアの機能の概略を表２に示す．

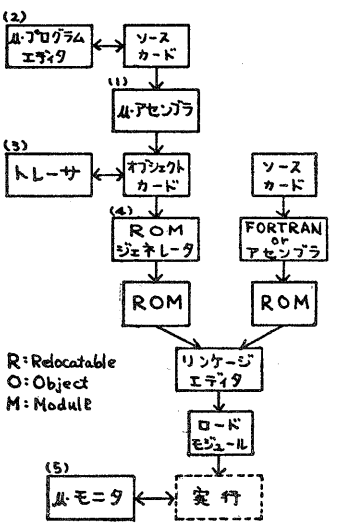


図１．　システム構成

表２．　プログラムの機能

プログラム名	機　能　の　概　要
μ-アセンブラ	μ-プログラム・ソースよりビット・パターンを生成し，制御記憶への割付けを行なう．
μ-プログラム・エディタ	ディスク中にあるμ-プログラム・ソースのメンテナンスを行なう．
トレーサ	チェック機能を含み，μ-プログラムの逐次実行を行なう．
ROM ジェネレータ	μ-アセンブラの出力から相対目的モジュール(ROM)を生成する．
μ-モニタ	WCSの管理等，μ-プログラムの実行時サポートを行なう．

３．　トレーサ

3.1　トレーサの特徴

本トレーサの特徴を以下に示す．

(i)　　シミュレーションのファームウェア化

通常，マイクロ・プログラムのデバッグはシミュレータを用いて行なわれるが，速度が遅い上，詳細な追跡が困難な場合が生ずる．　本トレーサでは命令の実行そのものは計算機自身で行ない，高速かつコンパクトな構成のシミュレータを実現している．

(ii)　　効率の良いデバッキング・エイド

マイクロ・プログラムのデバックは，シミュレータによる場合でも，直接メンテナンス・パネルを利用する場合でも困難なことが多いが，ここではシステム・ディスプレイを利用して会話的にデバッグを行なう機能をトレーサにもたせている．　またマルチ・プログラミング機能をもつ汎用OSの下で１個のユーザ・ジョブとして動作可能であり，トレーサの実行によって計算機の利用効率が低下する事はない．

(iii)　　リアル・タイム・チェック機能

トレーサではマイクロ命令の実行前にその命令のチェックを行なっている．従ってマイクロ・プログラムのリアル・タイム・チェックが可能となり，通常のアセンブラでは検出できない過去の履歴によって禁止されるマイクロ命令の実行を検出している．

3.2 トレーサの構成

トレーサの構成を図2に示す。トレーサは6個の処理モジュールより構成される。このうちのコマンド処理，マイクロ命令チェック，マイクロ命令実行の各モジュールは次節以降で説明し，ここではその他のモジュールの概要を述べる。

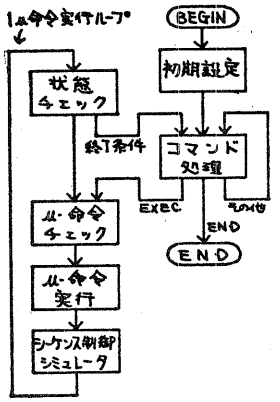


図2. トレーサの構成

(1) 初期設定モジュール

トレーサ自身の初期設定を行なう他，デバッグの対象とするマイクロ・プログラムを入力し，そのプログラムのデータの設定等，実行の準備を行なう。

(2) シーケンス制御シミュレータ

マイクロ命令は計算機のハードウェアを利用して1命令ずつ独立して行なわれるため，次に実行すべき命令は決定されない。このシーケンス制御をソフトウェアによってシミュレートしているのがこのモジュールである。

(3) 状態チェック・モジュール

マイクロ命令の実行を引き続いて行なうか，実行を終了してコマンド処理に移るかを決定する。この要因としては (i) 1マイクロ命令の実行毎にコマンド処理に移る動作モードである，(ii) あらかじめ設定されているブレーク・ポイントの条件を満足した，(iii) マイクロ命令の実行中にオペレータからの割込みが発生した，以上の3項目があり，これらのチェックを行なっている。

3.3 マイクロ命令の実行

前述したようにトレーサはマイクロ命令を1個ずつ取り出し実行する一種のシミュレータであるが，マイクロ命令の実行自身はM-500のハードウェアを使っておこなわれる。すなわちマイクロ命令実行モジュールではWCSに格納されているトレーサ・マイクロ部によって以下の手順でマイクロ命令が実行される。（図3参照）

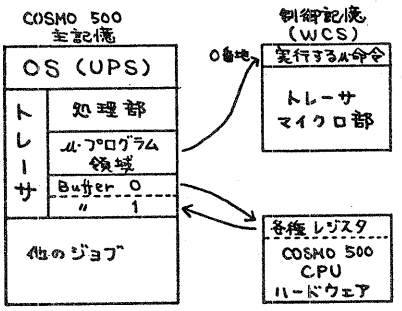


図3. マイクロ命令の実行

(i) 実行しようとするマイクロ命令をWCSの0番地に書き込む。

(ii) 主記憶中のバッファ（BF0）より前のマイクロ命令実行後の状態を取り出し，CPUの各レジスタにロードする。

(iii) 0番地のマイクロ命令を実行する。この後，M-500のM-LOOP機能(注)によりWCS 8番地の実行に移る。

(iv) CPUの各レジスタの内容を主記憶中のバッファ（BF1）にセイブする。（BF1からBF0への転送はシーケンス制御シミュレータで行なわれる。）

(注) マイクロ・プログラムの実行アドレスとメンテナンス・パネルのデータ・スイッチの内容が一致すると，強制的にWCSの8番地にブランチする機能で，マイクロ・プログラムのデバッグ等に有効な機能である。

3.4 トレーサ・コマンド

トレーサの動作の詳細を規定するために本トレーサでは27種のトレーサ・コマンドを設けている。このコマンドは入力モードの指定により，カード読取機とシステム・コンソールのいずれからでも入力でき，バッチ・ジョブと会話型ジョブの切り換えを自由に行なえる。27個のトレーサ・コマンドは，トレーサの各種実行モードを指定するモード指定コマンド，トレーサの実行を制御する実行制御コマンド，レジスタ，主記憶，制御記憶の値を設定，表示する設定・表示コマンドの3種に分類される。

各コマンドの機能概要を表3に示す。

	コマンド	機能
モード指定コマンド	CARD	トレーサ・コマンドの入力装置をカード読取機に指定する
	KEYIN	〃 システム・コンソール 〃
	COND	ブレーク・ポイントの条件を満足するまで連続実行する
	STEP	1μ命令の実行毎，コマンド処理に移る
	UNTRACE	トレース・レジスタの内容を出力しない
	TRACE	〃 する
	CHECK	μ命令の実行時チェックを行なう
	NOCHECK	〃 行なわない
	LPON	トレース・レジスタの内容をラインプリンタに出力する
	LPOFF	〃 しない
	OUTPUT	1μ命令実行毎のレジスタの内容をファイルに出力する
	NOOUT	〃 しない
実行制御コマンド	EXEC	μ命令の実行を開始する
	END	トレーサを終了する
	SAVE	CPUの全状態をファイルに退避する
	LOAD	〃 から復帰する
	INPUT	レジスタの内容をファイルから入力する
設定・表示コマンド	BPSET	ブレーク・ポイントの条件を設定する
	RGSET	トレース・レジスタを設定する
	RGRESET	トレース・レジスタに設定されているレジスタを削除する
	DUMP	全レジスタの内容を表示する
	DISPLAY	指定したレジスタ，主記憶，制御記憶の内容を16進数で表示する
	READ	〃 10進数 〃
	BINARY	〃 2進数 〃
	STORE	指定したレジスタ，主記憶，制御記憶に値を設定する
	BITSET	〃 の1ビットを"1"にする
	BITCLEAR	〃 "0" 〃

表3. トレーサ・コマンド

3.5 チェック機能

M-500のマイクロ・アセンブラでも種々の誤り検出を行なっているが，なかには検出できない誤りがあり，これらを実行すると計算機がハング・アップし，以後，動作不能におちいる事がある。トレーサではリアル・タイム・チェック機能により，OSの下でユーザ・ジョブとして正常に動作するようにチェックする事が可能である。従って，このチェック項目の中には，上記のようにハング・アップする命令ばかりでなく，主記憶データに対する読み，書きがユーザ領域に含まれるかどうか，ユーザ・マイクロ・プログラムでは通常使用しない命令を使用していないか，および2ステップの組み合わせで禁止されるものや前に実行した命令の履歴によって禁止されるマイクロ命令のチェック等を含んでいる。この結果，トレーサによって正常に動作する事が示されたマイクロ・プログラムは，OSの管理下でユーザ・ジョブの中からリンクされて，他のタスクやOSを破壊する事なく動作する事が保証される。

3.6 トレーサの使用例と評価実験

トレーサを使用してスレッシュホールドをとるマイクロ・プログラムのデバッグを行なった時のリスティングを図4に示す。

同じ関数でデータ数を 50 とした時のトレーサの実行速度を表4に示す．（この時の実行ステップ数は 260 ステップである．）

本トレーサの大部分はFORTRANで記述してあるため実行速度はそれほど高速になっていない．

表4．トレーサの実行速度

動　作	実行時間	実行速度/ステップ	比率
単純実行	6.7 s	25.8 ms	1.0
次命令チェックを含む	7.6 s	29.2 ms	1.1
OUTPUT動作を含む	23.0 s	88.5 ms	3.4
TRACE実行(LPON)	92.0 s	354. ms	13.7

プログラムの大きさは，FORTRANが約1700ステップ，アセンブラが約300ステップで，主記憶使用量18 K語，マイクロ部の大きさ131語である．

```
                    * C-500 MICRO PROGRAM TRACER *
#TRACE
                    *  FLAG SET.
#CONDITION
                    *  FLAG RESET.
#STORE  NA=X100
                    *  STORE COMMAND EXECUTION END.
#STORE  RF=X2494
                    *  STORE COMMAND EXECUTION END.
#RGSET  MD,TR,PC,ST,FG,FH,FI,W0,W1,W2
                    *  SET TO TRACE REGISTER.
                  MD TR PC ST FG FH FI W0 W1 W2
#BPSET  NA,EQ,X116
                    *  BREAK POINT SET.
#READ  MM:1,12
          MM(  1)  1000          1000          1000          1000
主記憶    MM(  5)  1000            10            10            10
          MM(  9)    10            10            10           100
                                                データ数      スレッシュホールド
#DISP  CM:X100,X11F
          CM(0100)  29E0 4001   29E0 B050   4880 D0B0   4881 D0B1
          CM(0104)  9070 4218   9050 4218   4880 D0B0   4881 D090
制御記憶  CM(0108)  9070 4C38   9050 4C38   9108 4051   29E0 B060
          CM(010C)  8850 5261   2110 4070   8850 B271   2110 4080
          CM(0110)  9109 4C81   910A 4090   2080 4091   2080 B0A0
          CM(0114)  810A B0A1   9069 DC29   F000 0000   811A B0C0
          CM(0118)  9069 D049   03FF F700   03FF F700   03FF F700
          CM(011C)  03FF F700   03FF F700   03FF F700   03FF F700
#SAVE  1
                    *  ALL REGISTER,CM,MM SAVED.
#EXEC
                    *  MICRO INSTRUCTION EXECUTE.
                    SC =    1      CA = 0100     CD = 29E0 4001    NA = 0
                          MD  0000    TR  0000    PC  0100    ST  1680
                          FH  401F    FI  0430    W0  0000    W1  0000
                    SC =    2      CA = 0101     CD = 29E0 B050    NA = 0
                          MD  6700    TR  0000    PC  0100    ST  1680
                          FH  401F    FI  0430    W0  0000    W1  0000
          （途中省略）
                    SC =   60      CA = 0106     CD = 4880 D0B0    NA = 0
                          MD  0000    TR  000A    PC  0100    ST  1680
                          FH  401F    FI  0430    W0  6709    W1  0000
                    *  BREAK POINT CONDITION SATISFIED.
#READ  MM:1,12
          MM(  1)     1             1             1             1
          MM(  5)     1             0             0             0
          MM(  9)     0             0            10           100
#LOAD  1
                    *  ALL REGISTER,CM,MM LOADED.
#READ  MM:1,12
          MM(  1)  1000          1000          1000          1000
          MM(  5)  1000            10            10            10
          MM(  9)    10            10            10           100
#END
                    *  TRACER EXECUTION END  *
```

図4．トレーサ使用例

4. マイクロ・モニタ

4.1 マイクロ・モニタの目的

OSが主記憶に常駐してユーザ・ジョブの管理とサポートを行うソフトウェアである事と同様に，WCS中に常駐してユーザ・マイクロ・プログラムの実行を管理・支援するマイクロ・プログラムをマイクロ・モニタと名付ける。 このようなマイクロ・モニタが必要な理由，およびこれを使用する事の効果は以下のとうりである。

(i) WCSの管理と効率的利用

WCSは計算機システム中に1個存在し，多くのジョブが同時に利用する資源のひとつであるから，本来OSの管理の対象となるべきものである。 しかし，通常のOSでは1回のモニタ・コールで長時間のオーバヘッドを要し，マイクロ・プログラムを利用する事の利点を失ないかねない。 従ってWCS中に常駐するとともに，ユーザ・ソフトウェアから直接リンクし，ユーザ・マイクロ・プログラムのダイナミック・リンキングを行なう管理マイクロ・プログラムが必要となる。

(ii) ソフトウェア・インターフェイスの標準化

ユーザ・ソフトウェアの中からマイクロ・プログラムにリンクするためには特殊なマシン命令を使用し，この命令のパラメータのセット，ソフトウェアとマイクロ・プログラム間でのパラメータの受け渡し等，繁雑な問題を多く含んでいる。 M-500のサポート・プログラムではこれらの操作をすべて吸収する事が配慮され，このためにマイクロ・モニタと次章で述べるROMジェネレータが利用できる。 (ROM: Relocatable Object Module) 従ってこれらのプログラムを使用すれば，一般ユーザは繁雑な処理の内容についてまったく関知する事なく，標準のインターフェイスでマイクロ・プログラムが利用できる。

(iii) マイクロ・プログラムの共通処理の吸収

計算機のハードウェアは通常，マシン命令の実行に使用されているが，ユーザ・マイクロ・プログラムでもこれと同じハードウェアを使用するため，正常なマシン命令の実行のために破壊する事が禁止されるレジスタがいくつかある。 従って，これらのレジスタを使用する時には実行前後にそれぞれ退避，復帰を行なわなければならないが，これらの処理を各マイクロ・プログラムで行なうのは，マイクロ・プログラムの開発者に負担となるばかりでなく，むだなWCSを必要とする。 この他によく使われるマイクロ・プログラム（たとえば乗除算）をマイクロ・モニタの一部として保持すればユーティリティとして利用できる。

4.2 マイクロ・モニタの構成

マイクロ・モニタの構成を図5に示す。 現在のマイクロ・モニタは4.1節の(iii)に言う所のユーティリティ・マイクロ・プログラムはもっておらず，1個のテーブルと4個のルーチンからなる構成をしている。 このテーブルをWCSロード・テーブルと呼び，マイクロ・プログラムをWCSにロードするとともにこのテーブルに登録する。 従ってこのテーブルはWCS中に存在するマイクロ・プログラムの管理テーブルである。 マイクロ・モニタは全体で47語である。

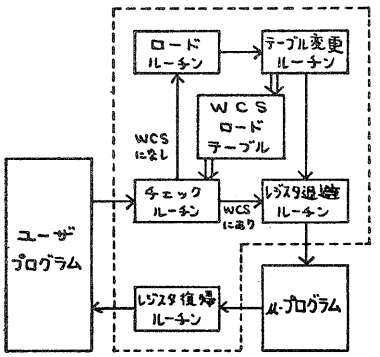


図5 マイクロ・モニタの構成

4.3 マイクロ・モニタの機能

マイクロ・モニタを使用する事の効果は4.1節で述べたが，マイクロ・モニタの最も重要な機能はマイクロ・プログラムのダイナミック・リンキングである．以下ではこの機能を中心にマイクロ・モニタの動作を説明する．（図5参照）

(i) ユーザ・ジョブから直接，チェック・ルーチンに入る．

(ii) チェック・ルーチンではWCSロード・テーブルを調べて，実行しようとするマイクロ・プログラムがWCSにロードされているかチェックする．
（ロードされている時はレジスタ退避ルーチンへ．
ロードされていない時はロード・ルーチンへ．

(iii) ロード・ルーチンでは主記憶よりマイクロ・プログラムを取り出し，WCSに格納する．その後，テーブル変更ルーチンへ

(iv) テーブル変更ルーチンでは，ロードしたマイクロ・プログラムをWCSロード・テーブルに登録するとともに，これをロードする事によってWCSから失なわれたマイクロ・プログラムをWCSロード・テーブルより削除する．

(v) レジスタ退避ルーチンでは正常なマシン命令の実行を保証するために必要ないくつかのレジスタをWCS中に退避する．その後，マイクロ・プログラムの実行に移る．（実行後はレジスタ復帰ルーチンへ）

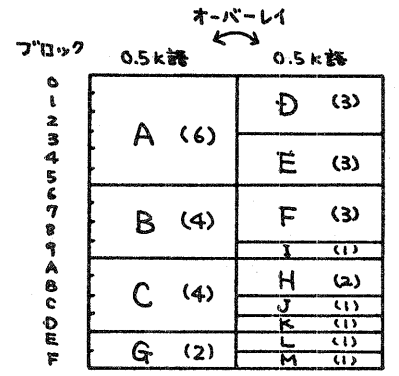
(vi) レジスタ復帰ルーチンでは前に退避してあったレジスタを復帰し，ユーザ・ジョブに戻る．

以上の機能によりマイクロ・プログラムの利用者はWCSへのロードにまったく関知する必要がなく，開発者は種々の制限を意識する事なく，自分のアルゴリズムだけに専念してプログラムの開発ができ，マイクロ・プログラムの高速化とWCSの効率的な利用が可能となる．

4.4 マイクロ・モニタの性能評価

マイクロ・モニタの効果は明らかであり，マイクロ・プログラムの実行は高速化されるが，これを経由するために若干のオーバヘッドも存在する．従ってどの位高速化されるかをある程度定量的に調べるため，以下の評価実験を行なった．

《状況設定》 0.5K語のWCS領域を13個（全体で1K語の大きさ）のマイクロ・プログラムで使用する．この時の配置とプログラムの大きさを図6に示す．(注) マイクロ・プログラムがコールされる確率は大きさや配置にかかわらず一様であるとする．



1ブロック＝32語　（）内ブロック数

図6. プログラムの大きさと配置

(注) この状況設定は，筆者の所属している研究室で開発中の画像処理ファームウェア・システムを，規模を縮少してモデル下したものである．

《マイクロ・モニタのオーバヘッド》 以下の式で示される．

WCSにあった時　6.8 (μsec.)

WCSになかった時　$16.5 + 1.9*32*B + 4.6*V$ (μsec.)

（B：ロードしたプログラムの大きさ（ブロック数）
V：テーブル変更の回数　　但し，1ブロックは32語である．

《マイクロ・モニタを使用しなかった時のオーバヘッド》　この場合は実行のたびごとにWCSへロードするため，以下の式で示される．

$$1.9 * 32 * B' \quad (\mu sec.) \qquad B' : ブロック数$$

以上の条件の下で1000回のマイクロ・プログラムへのコールが発生した場合をシミュレートした。　表よりマイクロ・モニタを使用した時としなかった時のオーバヘッドを計算してみると以下のようになる．　(msec.)

使用した時 (a) ……102.9
使用しなかった時 (b)……148.7

a/b　*100%　　69.2%

表5．シミュレーションの結果

《マイクロ・モニタを使用した時》	
WCSにあった回数	448回
WCSになかった回数	552回
ロードしたプログラムの総ブロック数(B)	1454ブロック
テーブル変更の総数(V)	822回
《マイクロ・モニタを使用しなかった時》	
ロードしたプログラムの総ブロック数(B')	2462ブロック

この結果よりマイクロ・モニタを使用すれば30％以上，オーバヘッドが短縮される事がわかる．　さらに本来は，マイクロ・プログラムをWCS上に配置する場合，関連の深いものがオーバレイしないように配慮する事が可能であり，また1回実行したマイクロ・プログラムがその後引き続いて実行される確率は非常に高い事から，上のシミュレーションの結果は最悪の場合と考えられ，実際の運用に際してはさらにオーバヘッドが減少するものと期待される．

5．高級言語とのインターフェイス

4.1節で述べたようにユーザ・ソフトウェアからマイクロ・プログラムを利用するには繁雑な処理を多く含んでいて，マイクロ・プログラムの開発者にも利用者にも負担となる．　一般の利用者にとってはソフトウェア・ライブラリと同様にプログラムの名称と引数の数，型だけでマイクロ・プログラムを利用できる事が望ましい．　この目的のために，ソフトウェアとのインターフェイスを標準化し，繁雑な処理を吸収するために開発されたのがROMジェネレータであり，この標準インターフェイスに従ってマイクロ・プログラムの実行をサポートするのがマイクロ・モニタである．

M-500マイクロ・アセンブラの出力は専用ローダの入力となるオブジェクト・カードであって，リンケージ・エディタによって他の高級言語とリンクできる相対オブジェクト・モジュール（ROM）とは異なった形式を採っている．　ROMジェネレータは，マイクロ・アセンブラのオブジェクト・カードと，いくつかのパラメータを入力してリンケージ・エディタの入力対象となるROMを生成するプログラムである．　但し，現在のROMジェネレータはマイクロプログラムのオブジェクト・カードから，直接ROMを

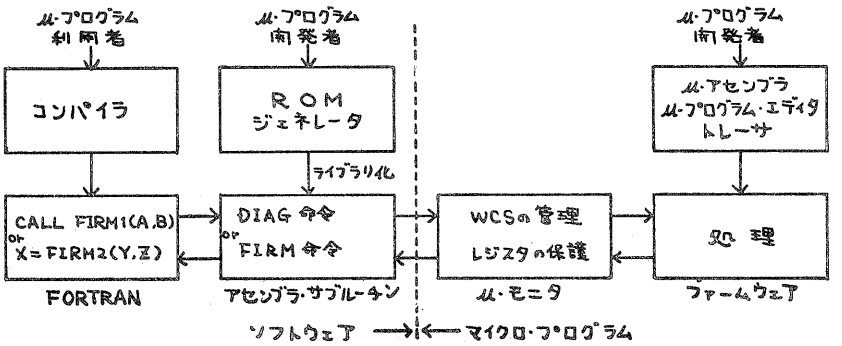


図7．高級言語とのインターフェイス

生成するのではなく，一旦アセンブラのソース・プログラムをジェネレートし，これをアセンブルする事によりROMを得ている．　このROMはFORTRANにおけるサブルーチン副プログラム，または関数副プログラムと同じ形式を採っているため，FORTRANユーザはそのプログラムにおいて

CALL　FIRM1(A,B)　　または　　X= FIRM2(Y,Z)

のように記述できる．　さらに，このROMをファームウェア・ライブラリとしてファイルに格納しておけば，一般ユーザはリンク時にこのライブラリを指定するだけでマイクロ・プログラムを利用する事ができる．

M-500においてマイクロ・プログラムを開発し，これを高級言語から利用する時の手順を図7に示す．　マイクロ・プログラムの開発者は，マイクロ・アセンブラ，マイクロ・プログラム・エディタ，およびトレーサを利用してマイクロ・プログラムを開発する．　この開発が終了した時点でROMジェネレータにより，マイクロ・プログラムを含むROMを作り，ライブラリに登録する．　以後，これを利用する時は上記のように記述し，マイクロ・モニタのサポートの下で実行される．

6.　あとがき

ファームウェアの利用をサポートするソフトウェアに関して，汎用中型計算機MELCOM-COSMO 500に於いて実験的に開発したソフトウェア・システムについて報告した．　このシステムは，筆者の所属する研究室に於ける大規模画像処理ファームウェアの開発や，エミュレータの開発に利用されている．

マルチ・プログラミングの環境の下でファームウェアを有効に利用するには，

(1)　マイクロ・プログラムの正常動作の保証．

(2)　実行時におけるWCSの管理

が特に必要と考えられ，(1)の観点からは

①　マイクロ・プログラム・チェック機能とその方式

②　プロテクション機能および計算機リソースの部分的開放

などが，(2)の観点からは

①　マイクロ・モニタのハードウェア化

などの問題があり，今後，マイクロ・モニタの評価等と併せて，これらの問題の検討を行ないたいと考えている．

謝辞

本システムの開発にあたって御協力いただいた三菱電機計算機製作所　計算機製造部　曽我正和，小野和夫，田渕謹也，清尾克彦の諸氏に感謝いたします．

文献

S.S.Husson ; Microprogramming , Principles and Practices. Prentics - Hall (1970)

R.F.Rosin : Comtemporary concept of micro-programming and emulation , Computing Surveys 1 , No. 4 (1969)

曽我他 : MELCOM - COSMO シリーズ モデル 500，三菱電機技報，Vol.49, No.5 (1975)