低速小規模マイクロプロセッサを内蔵したサービスプロセッサ†

岩根雅彦†† 佐藤文孝††† 村山正樹††††

本論文では，まず低速小規模マイクロプロセッサを内蔵したSVPの構成において，SVPに複数のCPUインタフェースを設け，システム操作卓機能を実行するチャンネルサービスプロセッサと保守修復機能を実行する保守プロセッサを時分割で動作させることによって，一つのSVPでマルチCPUシステムがサポートできることを示す．次に，保守プロセッサにおいて，簡単な低級言語を使用することによって，保守コマンド実行時間が実用的な範囲に収まり，CPUのレジスタの読み出しおよび書き込み手順がわかりやすく，CPUハードウェアの変更に容易に対処できる方法について提案する．次に，マイクロ診断において，保守プロセッサと同様に簡単な低級言語を使用することによって，実用的な範囲にマイクロ診断プログラムの実行時間とプログラム容量が収まり，CPUハードウェアの変更に容易に対処でき，かつ検査手順がコンパクトになるような方法について提案する．これらの提案した方法により保守プロセッサおよびマイクロ診断プログラムを含んだSVPを開発した結果，4k語のメモリをもった低速マイクロプロセッサを内蔵したSVPでも，大規模高速マイクロプロセッサを内蔵したSVPと機能的にほぼ同等であり，かつ保守コマンドの応答時間は約1秒の性能があり，十分実用になることがわかった．

1. はじめに

サービスプロセッサSVP (SerVice Processor) は，CPU (Central Processing Unit) とは独立したコンピュータシステムであり，何らかの形でCPUと接続され，プロセッサ，記憶装置，フロッピディスク，ディスプレ装置，CPUインタフェース等で構成されている[1]．SVPの内蔵プロセッサには，LSI-11[3]，0.2 MIPSのマイクロプロセッサ[2]のような高性能のマイクロプロセッサまたはミニコンピュータ[1]が採用され，記憶装置は32kB～128kBの容量をもっていた．

SVPは，システム操作卓機能のほか，システム異常状態監視機能，再試行機能，構成制御機能，保守診断機能，およびシステム立ち上げ機能等をもち，システム運用および保全を行うための重要な役割を果たす処理装置であるので，できるだけ簡潔な信頼性の高い構成にする必要がある．そのために部品数が少ない構成にすることが重要であり，このことはコスト低減につながる．これらの機能の能力は，SVPの内蔵プロセッサのソフトウェアによって決まるが，従来のSVPは内蔵プロセッサの能力を十分に生かしきっておらず，次の欠点をもっていた．

マイクロ診断，FLTおよびLSSDで使用される検査データは，一般に単一固定故障を仮定して作成される[6]ので間欠故障や多重故障などの仮定外の故障に対して正しく故障個所を指摘できないことがある．このようなとき少し手間どっても保守作業者が故障個所を局所化できる補助機能が必要である．おもな補助機能の一つはレジスタの読み出しおよび書き込み機能である．大規模なマイクロプロセッサを内蔵したSVPではレジスタをニーモニックで指示している[5]が，小規模なマイクロプロセッサを内蔵したSVPでは番号で指示していた[3]ので操作性がよくなかった．また，SVPソフトウェアとCPUの修復に必要な手順が分離されておらず，CPUハードウェアの変更に伴ってSVPソフトウェアを変更する必要があったのでソフトウェアの保守性がよいとはいえなかった．さらに，CPUとSVPは1対1に対応しておりマルチCPUシステムのとき，CPUと同数のSVPが必要であった．

このような問題点を解決し，さらに小容量のメモリをもった低速プロセッサで，その操作性および応答性において実用に耐えるようにSVPを設計するためには工夫が必要である．本論文では，まず低速小規模マイクロプロセッサを内蔵したSVPの構成について述べる．SVPに複数のCPUインタフェースを設け，システム操作卓機能を実行するチャンネルサービスプロセッサと保守修復機能を実行する保守プロセッサを

† A Service Processor Using a Low Speed Small Scale Microprocessor by MASAHIKO IWANE (Department of Information and Control Engineering, Toyota Technological Institute), FUMITAKA SATO (Ome Works, Toshiba Corporation) and MASAKI MURAYAMA (Toshiba Computer Engineering Corporation).

†† 豊田工業大学制御情報工学科
††† (株)東芝青梅工場
†††† 東芝コンピュータエンジニアリング(株)

時分割で動作させることによって，一つの SVP でマルチ CPU システムがサポートできることを示す．次に，保守プロセッサについて述べる．マンマシンインタフェースをよくするための操作員にわかりやすい保守コマンドおよびコマンド形式について述べ，この保守コマンドの実行時間について議論する．そして，簡単な低級言語を使用することによって，保守コマンド実行時間が実用的な範囲に収まり，CPU のレジスタの読み出しおよび書き込み手順がわかりやすく，かつ CPU ハードウェアの変更に容易に対処できる方法について提案する．次に，マイクロ診断プログラムの設計思想および設計方法について述べる．保守プロセッサと同様に，マイクロ診断プログラムの実行時間を予測する．そして，簡単な低級言語を使用することによって，実用的な範囲にマイクロ診断プログラムの実行時間とプログラム容量が収まり，CPU ハードウェアの変更に容易に対処でき，かつ検査手順がコンパクトになるような方法について提案し，最後に，これらの提案した方法により保守プロセッサおよびマイクロ診断プログラムを開発した結果について検討する．

2. システムの構成

計算機システム全体の構成は前論文[9]で述べられているので，ここでは，以後の議論に必要なことのみをできるだけ簡単に述べる．

SVP は，フロッピディスク FD, CRI ディスプレイ CRT, キーボード KB, CI ポート CCI, 2個の DI ポート DI#1, #2 およびこれらを制御するマイクロプロセッサ μCPU とメモリ μM 等から構成されたプログラム内蔵の独立した処理装置であり，ハードウェア構成を**図1**に示す．

μCPU は，12ビット並列プロセッサで，基本的に4bits の命令コードをもち，1.2MHz のクロックで動作し，レジスタタイプの通常命令を約 7 μs, インダイレクトインデックスタイプの命令を約 20 μs で実行する．4k 語（1語＝12bits）のメモリ領域に対して直接アクセス可能であり，また8レベルの割り込み受付とそれらの優先度を決定する機能をもっている[4]．

プログラム用のメモリとして4k 語をもち，これらのメモリにはパリティビットを付加している．メモリレイアウトを**図2**に示す．フロッピディスクはメモリと DMA (Direct Memory Access) で接続され，巡回符号による誤り検出を行っている．

本 SVP が対象とする CPU[7)~9)] は，演算処理部 (EPU) と診断部 (DU) とから構成され，その規模は約 120,000 ゲートである．EPU は仮想記憶をもっており，かつ命令語の先回り制御をパイプライン処理している．1命令語を実行するために必要な仕事は，命

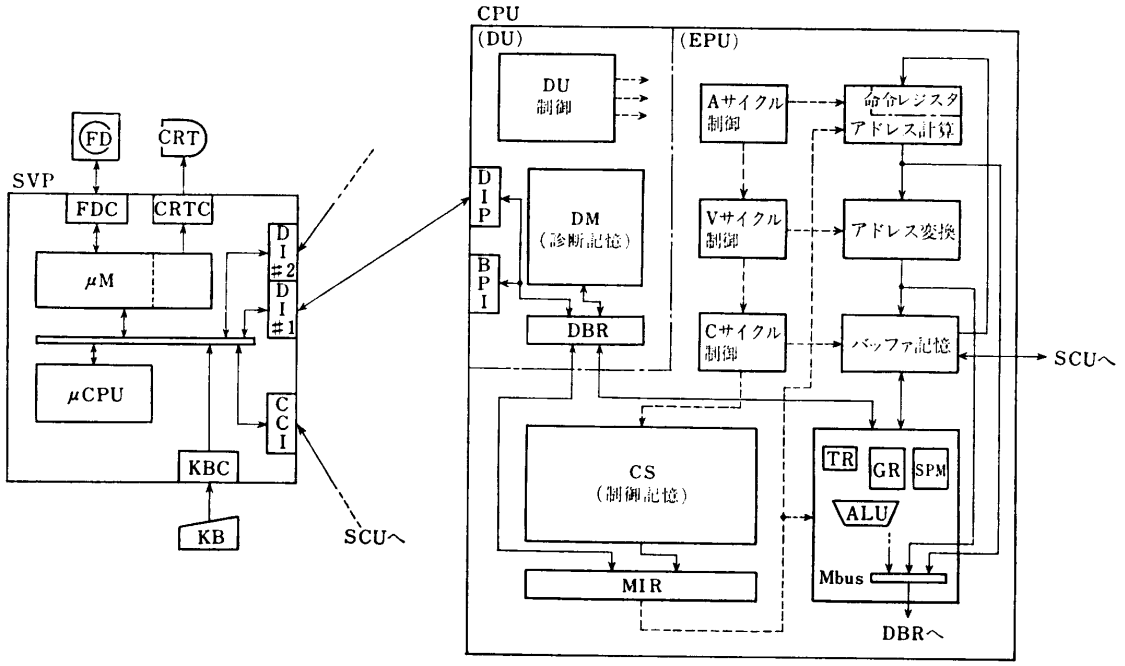


図1 ハードウェア構成
Fig. 1 Hardware configuration.

令の読み出し，オペランドアドレスの計算，仮想アドレスから実アドレスへの変換，バッファ記憶からオペランドの読み出し，命令実行という5ステップに分解でき，これらのステップをおのおのIサイクル，Aサイクル，Vサイクル，Cサイクル，Eサイクルと呼ぶ．IサイクルはA, V, Cサイクル用ハードウェアで処理できる．A, V, Cサイクルの処理は画一的なものが多いのでその大部分の制御を結線論理とし，Eサイクルの処理は多種多様の演算操作を行うのでマイクロプログラム制御としている．

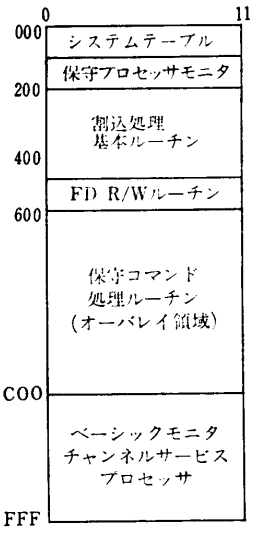


図 2 SVP メモリレイアウト
Fig. 2 Memory layout of SVP.

DU は，SVP とのインタフェースと SVP が使用不可能のときに使用する基本保守パネルインタフェース BPI および診断記憶 DM 等で構成され，EPU とは独立に動作し，**表1**に示した診断コマンドを SVP 等から受け取って解釈実行して EPU を制御することができる．

SVP のソフトウェアは，システム操作卓機能を実行するチャンネルサービスプロセッサ，保守修復機能を実行する保守プロセッサおよびシステム立ち上げや CPU の異常を処理し回復する割り込み処理プロセッサ，さらに SVP 全体を制御するベーシックモニタで構成されている．

SVP 処理モードには，単一処理モードと共用処理モードがあり，単一処理モードのときはチャンネルサービスプロセッサのみが動作し，共用処理モードのときはチャンネルサービスプロセッサと保守プロセッサが交互に動作する．チャンネルサービスプロセッサと保守プロセッサ間の制御の移行は一つのチャンネルメッセージまたは一つの保守コマンドの終了時点で起こりうる．マルチ CPU システムにおいて，一つの CPU に故障が生じ，残った CPU でシステムを再立ち上げしてジョブを継続するとき，同時に故障した CPU を修復する必要がある．このような場合に共用処理モードを

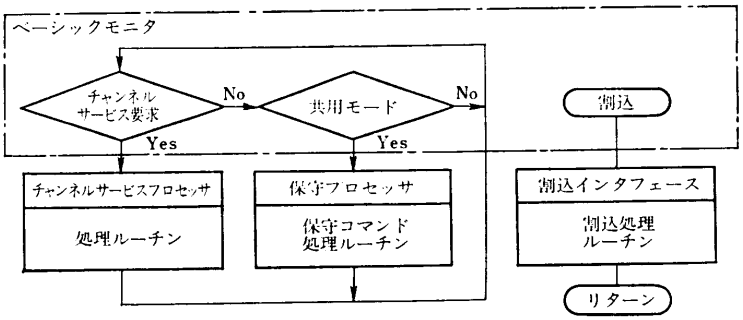


図 3 SVP 基本フローチャート
Fig. 3 Basic flowchart of SVP.

表 1 おもな診断コマンド一覧
Table 1 Diagnostic commands.

コマンド種類	コード部 0……7	番地部 8……15	データ 16……47	説明
Start EPU	000 (制御記憶番地)		—	EPU クロックの起動
Step EPU	001 (制御記憶番地)		—	EPU クロックの1ステップのみ起動
Stop TEST	01000 (診断記憶番地)		—	診断記憶中の診断コマンドの実行停止
Branch TEST	01010 (診断記憶番地)		—	指定された診断記憶へジャンプ
Start TEST	01011 (診断記憶番地)		—	診断記憶中の診断コマンドの実行
Read CS	011 (制御記憶番地)		—	制御記憶を MIR へ読み出し
Read DATA	10000000	Reg/Bus	有	指定されたレジスタ/バスの読み出し
Write DATA	10100000	Reg/Bus	有	指定されたレジスタ/バスへ書き込み

表 2 おもな保守コマンド一覧
Table 2 Maintenance commands.

保守コマンド	説明
Macro	マクロコマンドの定義および実行
Read	レジスタまたはメモリ内容の読み出し
Write	レジスタまたはメモリへデータの書き込み
Execute	EPU の起動
Step	1マイクロ命令または1命令実行モードの設定
Config.	システム構成の変更
Initialize	EPU の初期化
Adr Stop	制御記憶または主記憶アドレス一致停止モード設定
Flt Stop	例外事態発生停止モード設定
Load	制御記憶または診断記憶へローディング
Transfer	診断コマンド(immediate form)の転送
Diagnose	マイクロ診断プログラムの実行
Bye	保守プロセッサ動作の終了

使用すれば，コスト面で有利である．**図3**に SVP の基本動作のフローチャートを示す．

3. 保守プロセッサ

診断部 DU のハードウェア量を少なくするために診断コマンドは低水準であり，人間とのインタフェースである保守コマンドは高水準であることが必要である．保守コマンドには，**表2**に示すように EPU の起動，EPU の初期化，構成変更，マイクロ診断等がある．とくに Macro 保守コマンドは，いくつかの保守コマンド列を定義して登場したり，登場した保守コマンド列を実行する機能である．この保守コマンドはマイクロプログラムのパッチにも使用できる．

サービスプロセッサ SVP はキーボード KB の「保守」キーが押されると共用処理モードとなり画面上に〈M〉を表示して保守コマンド待ちとなる．保守コマンドが入力されると，保守プロセッサはフロッピディスクからオーバレイ領域に該当する保守コマンド処理ルーチンを読み込み，保守コマンド処理ルーチンに制御を渡す．保守コマンド処理ルーチンは診断コマンド列からなる手順をフロッピディスクから読み込んで診断部 DU に送り，結果を画面上に表示する．

このときの保守コマンド実行時間 T は次式で近似することができる．

$$T \sim n * t(a) + t(b) + t(c) \qquad (1)$$

ここで，$t(a)$ は保守コマンド処理ルーチンまたは診断コマンド列よりなる手順をフロッピディスクから読み込む平均時間，n はフロッピディスクから読み込む回数，$t(b)$ はマイクロプロセッサが診断コマンド処理ルーチンを実行している時間，$t(c)$ は診断部 DU が診断コマンドを実行する時間である．一般に $t(c) \ll t(b) \ll t(a)$ であるので(1)は次式となる．

$$T \sim n * t(a) \qquad (2)$$

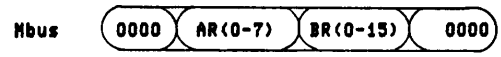
また，フロッピディスクのヘッドの移動時間を $t(d)$，ヘッドロードの時間を $t(e)$，回転待ち時間を $t(f)$ とすると，フロッピディスクからの平均読み込み時間 T は次式で近似できる．

$$T \sim n * (t(d) + t(e) + t(f)) \qquad (3)$$

マンマシンインタフェースを考えると，保守コマンドを入力してその応答が画面上に表示されるまでの時間は短いほうがよいが，実用面からは 1.0 秒弱程度の待ち時間は許容限度内である．平均して 10 トラック移動すると，これに要する時間が約 100 ms，平均回転待ち時間が約 80 ms およびヘッドロード時間が約 20 ms であるので，応答時間が 1.0 秒以内となるように計算すると，n は5以内である．

EPU 内レジスタの書き込みおよび読み出しのための診断コマンドによる手順は目的レジスタごとに異なるので，指定されたレジスタに対する診断コマンドの手順のみをフロッピディスクから読み込むようにしないとメモリに入りきらない．

Mbus を介して読み出された目的レジスタの内容をそのままの形で画面上に表示したり，目的レジスタへ書き込むための経路を考慮してキーボードから書き込みデータを入力することは人間にとって非常にわかりにくい．たとえばマイクロ命令で AR レジスタの内容を**図4**に示した形で Mbus に出力できるようになっている．この AR レジスタの内容を画面上に表示




```
0010    Name
0020    AR
0030    Read
0040    Write Data to MIR(AR →Mbus)
0050    Read Data in Mbus
0060    Rov04
0070    Shf104
0080    Mask
0090    FF000000
0100    Out02
0110    End
```

図 4 読み出し手順例
Fig. 4 A procedure to read out the register.

表 3 保守プロセッサ用制御語
Table 3 Control words for maintenance processor.

制御語	コード (16進)	説明
Ident. (Name)	800	レジスタ，バス名を示し手順の先頭に置く．
Read (Read)	801	レジスタ，バス内容を読み出す診断コマンド列を示す．
Write (Wrt)	802	レジスタ，バスに書き込む診断コマンド列を示し，Dpnt XX とともに使用する．
Data Pointer (Dpnt XX)	9XX	KB から入力されるデータの診断コマンド列内の位置を XX によって示す．
Receive (Rcv XX)	AXX	DI から受けとるデータのバイト数を示す．
Mask (Mask)	803	書き込みデータ，読み出しデータにマスクをかける．
Shift Right (Shfr XX)	BXX	書き込みデータ，読み出しデータを XX ビット右へシフトする．
Shift Left (Shfl XX)	CXX	書き込みデータ，読み出しデータを XX ビット左へシフトする．
Output (Outxx)	DXX	データを XX 桁 CRT へ表示する．
Input (Inp XX)	EXX	KB より XX 桁のデータを受けとる．
Step	804	EPU を1サイクル進める．
End (End)	80F	手順終了

する例を考える．この場合，次の手順が必要である．まず，Write Data 診断コマンドで MIR に上記マイクロ命令を書き込み，Read Data 診断コマンドで Mbus を4バイト読み出す．次に AR レジスタ部分を抽出するために4ビット左シフトし，それを FF000000 (16) でマスクする．そして，マスクしたものを左から2バイト表示する．このような編集を行うことによってマンマシンインタフェースが改善される．この編集はレジスタごとに固有であり，個々のレジスタごとの編集プログラムを組み込むことは，プログラムが複雑になるうえに CPU のハードウェアの変更があるたびに編集プログラムも変更しなければならない．そこで，レジスタごとに異なる編集作業を基本的ないくつかの作業に分解し，制御語でその作業を指示することによって編集プログラムを簡潔にでき，さらに診断コマンド列と保守コマンド処理ルーチンが分離できCPU のハードウェアの変更にも診断コマンド列の変更だけで対処できる．

表3 に保守コマンド処理ルーチン用制御語を示す．AR レジスタを Mbus から読み込み，画面上に表示するための制御語を含んだ診断コマンド列を図4に示し，制御語の役割を次に説明する．Name 制御語は手順の先頭におかれ，他の手順との区別のために使用される．Name に続く AR はレジスタ名である．Read 制御語は直後に続く Write Data to MIR (AR→Mbus) および Read Mbus が CPU からデータを受けとる診断コマンド列であることを表し，Rcv 04 制御語は4バイトのデータを CPU から受けとることを示す．Shfl 04 制御語は受けとった4バイトのデータを4ビット左シフトすることを指示し，Mask 制御語は直後に続く FF000000 でこのデータをマスクすることを示す．次の Out 02 制御語はマスクしたデータを左から2桁を CRT に表示することを指示し，End 制御語は手順の終りを示す．

このように，制御語を含んだ診断コマンド列を簡単な低級言語で記述し，診断アセンブラでビットパターンに変換して，あらかじめフロッピディスクに格納しておく．

4. マイクロ診断プログラム

故障検出機構で検出された故障は，修復単位に局所化されて交換修理される必要がある．この修復作業は容易でかつ短いことが望ましい．修復作業の容易さには，マイクロ診断プログラムが正確なこと，取扱いがやさしいこと，どんな故障に対しても何らかの修復処理ができること，CPU ハードウェアの変更に柔軟に対処できることが含まれる．

マイクロ診断プログラムの正確さは診断コマンド列で作成されたテストプログラムに依存するので，テストプログラム自身を十分チェックしておけばよい．マイクロ診断プログラムの実行に際して，装置の繋ぎ換えのような特別な作業は不要とし，かつ SVP の簡単なキーボード入力とニーモニックによる診断結果を画面上へ表示することによって取扱いを容易にする．また，テストプログラムを診断記憶にロードして CPU の遊休時に動作する PATROL (自己動作確認機構)[9] として実行させたり，たんに診断記憶上でループテストとして実行させたりすることによって間欠故障に対処する．SVP マイクロプロセッサソフトウェアとテストプログラムを分離することによって CPU ハードウェアの変更に対してテストプログラム部分のみを変更するだけですむように設計する．

一般にマイクロ診断の診断順序は Start Small により実行される[6]．この CPU の A,V,C サイクル処理ハードウェアは結線論理により制御され，Eサイクル処理ハードウェアはマイクロプログラムによって制御される．また診断部から直接書き込めるレジスタは制御記憶回路周辺レジスタとEサイクル処理ハードウェア内 TR レジスタで，診断部から直接読み出せるレジスタおよびバスは制御記憶回路周辺レジスタと Mbus および診断バスである．このことから，SVP をハードコアとして診断部 DU，制御記憶周辺回路，Eサイクル処理回路，Aサイクル処理回路ならびに制御回路，Vサイクル処理回路ならびに制御回路，Cサイクル処理回路ならびに制御回路，SCU インタフェース回路の順序で診断していくのが妥当である．

マルチ CPU システムでは一つの CPU が修復中であり，他の CPU は稼動中であるとき，修復中の CPU の SCU インタフェース回路を診断したいことがある．このとき診断部 DU 内モードレジスタ TMR の CPU Stand-alone ビットをセットすることにより SCU インタフェース折り返し機構が動作し，故障 CPU を物理的に切り離さないでも診断可能となる．

CPU ハードウェアの変更に対処するために，マイクロ診断プログラムに保守コマンド処理ルーチンと同様に制御語を導入して，マイクロプロセッサアセンブリ言語で書かれた診断エクゼクティブとおもに診断コマンド列からなるテストプログラムに分離した．テストプログラムは多数の小さなテスト単位（テストページ）から構成されている．診断エクゼクティブは診断条件の設定，テストページのフロッピディスクからの読み出しおよび実行，診断結果の画面上への表示等の各テストページに共通した処理を行うように設計する．

保守コマンド Diag の入力によってマイクロ診断エクゼクティブをフをフロッピディスクからオーバレイ領域に読み込んで制御をマイクロ診断エクゼクティブに渡し，診断エクゼクティブは順次テストページをフロッピディスクから読み出し診断部 DU に診断コマンドを送ることによってマイクロ診断を進めていく．1テストページをフロッピディスクから1回で読み込めるならばマイクロ診断プログラム実行時間は次式で近似できる．

$$T\sim t(g)+n*(t(h)+t(i)+t(j)) \qquad (4)$$

ここで，$t(g)$ はマイクロ診断プログラムの条件設定に必要な SVP のマイクロプロセッサと診断部 DU の実行時間の和，n はテストページの数，$t(h)$ は1テストページを実行するのに必要なマイクロプロセッサの実行時間，$t(i)$ は1テストページをフロッピディスクから読み込む時間，$t(j)$ は1テストページの診断コマンドを実行する時間である．一般に，$t(g)\ll n*(t(h)+t(i)+t(j))$ であるので，マイクロ診断プログラムの実行時間 T は次式となる．

$$T\sim n*(t(h)+t(i)+t(j)) \qquad (5)$$

大多数のテストページでは，$t(i)\gg t(h)+t(j)$ であるので，実行時間 T は次式で近似できる．

$$T\sim n*t(i) \qquad (6)$$

テストページを実行順序に従ってフロッピディスクに格納したとき，トラック上に連続するテストページの読み込みであるので，トラックに格納されているテストページ数を m，1回転待ち時間を約 160 ms，ヘッドロード時間を 20 ms とすると，診断プログラムの実行時間 T の予測値は次式となる．

$$T\sim 0.16\left(1+\frac{1}{m}\right)n+0.02 \qquad (7)$$

また，任意のテストページを選択して実行するとき，すぐにそのテストページが読み込まれなければならないので，ディレクトリが必要となる．しかし，すべてのテストページのフロッピディスク上の格納場所を示すディレクトリをもつことは，SVP メモリのオーバレイ領域が少ないので，容量の点からも，また実行時間の点からも得策ではない．そこで，テストページをテスト番号の小さい順にフロッピディスクに格納しておき，各トラックの先頭に格納されているテストページのテスト番号をディレクトリとしてもつことにする．このときのテストページをオーバレイ領域へロードする時間 $t(k)$ は次式で近似できる．

$$t(k)\sim t(l)+m*t(p)/2 \qquad (8)$$

ここで，$t(l)$ はディレクトリをフロッピディスクから読み込み該当トラックを捜す時間，m は1トラックに格納されているテストページ数，$t(p)$ は1テストページをフロッピディスクから読み込む平均時間である．

1トラックに平均して20テストページ格納するように設計すると，$t(l)\ll 10t(p)$ であるので，該当テストページをオーバレイ領域へロードする時間 $t(k)$ は次式となる．

$$t(k)\sim 10t(p) \qquad (9)$$

フロッピディスクからの平均読み込み時間 $t(p)$ は，トラック移動を伴わないので次式で近似できる．

表 4 マイクロ診断用制御語
Table 4 Control words for microdiagnostics.

制　御　語	コード (16進)	説　　明
Ident. (No.)	800	テストページ番号
PCB Name (Board)	801	被疑基板名
Logic (logic)	802	被疑回路名
Test Data (Data)	803	テストデータ
Receive (Rcv)	804	DI からデータを受けとる診断コマンド列
Send (Send)	805	DI からデータを受けとらない診断コマンド列
Interrupt (Int)	806	正解値 (CPU からの割り込みがある)
No Intr (Noint)	807	正解値 (CPU からの割り込みがない)
End (End)	80F	1テストページ終了

```
      ** MICRO DIAGNOSTICS **
0100*EJECT
0110C TEST PAGE NO 001430
0120C REGISTER NAME SPM(0)
0130      IDENT
0140       001430
0150      BOARD
0160       EA,EB
0170      LOGIC
0180       TR→SPM(0)
0190      SEND
0200       Write Data to TMR (All Clear)
0210       Read CS
0220       Write Data to TMR (No Clear)
0230      DATA
0240       00000000
0250       FFFFFFFF
0260      SEND
0270       Write Data to TR (DATA→TR)
0280       Write Data to MIR (TR →ALU →SPM(0))
0290       Step EPU
0300       Write Data to MIR (SPM(0) →Mbus)
0310      RCV
0320       Read Data in Mbus
0330      NOINT
0340       FFFFFFFF
0350       00000000
0360       FFFFFFFF
0370      END
```

図 5 マイクロ診断テストページ例
Fig. 5 A test page of microdiagnostics.

$$t(p) \sim 1.05t(q) \qquad (10)$$

ここで，$t(q)$ は1回転待ち時間であり，トラック上に連続するテストページの読み込みであるので全テストページの実行の場合と同様に約 160 ms とすると，$t(k)$ は約 1.7 秒となる．

テスト番号とテストページの実行順序を一致させることによって，全テストページの実行と選択されたテストページの実行とを統一的に取り扱うことができる．このように設計することによって操作性のよいマイクロ診断プログラムを開発することができる．

CPU のハードウェア変更への対応ならびに診断エクゼクティブおよびテストページの簡潔化を考慮して表 4 で示す制御語を導入した．そこで，テストページは制御語と診断コマンドで構成される．テストページの一例を図 5 に示す．

図 5 において，0100 行から 0120 行目は診断アセンブラへの指令または注釈である．制御語 IDENT に続く 001430 はテスト番号で，テストページを捜すときにも使用される．制御語 BOARD の次の EA, EB は被疑プリント基板名を表し，制御語 LOGIC の後の TR→SPM(0) は被検査論理回路名を示す．ここまでは，診断辞書に相当するもので，次の診断手順によって故障が検出されたならば，マイクロ診断エクゼクティブはこれらの情報を編集して画面上に表示する．制御語 SEND に続く 0200 行から 0220 行目までの診断コマンド列は，次の EPU の駆動に先がけての初期化するためのものである．制御語 DATA の後の 00000000 と FFFFFFFF は直後の診断コマンドのデータである．この制御語以降でかつ制御以降でかつ制御語 NOINT までにある診断コマンド列は制御語 DATA で与えられたデータ分繰り返し実行される．すなわち，0260 行目の 00000000 → TR，0280, 0290, 0300, 0320 行の診断コマンドが最初に実行される．次にFFFFFFFF → TR，0280～0320 行の診断コマンドが実行される．これらの診断コマンド列は TR レジスタに与えたデータをスクラッチパッドメモリ SPM (0) に転送し Mbus で読み取る動作を行う．また，制御語 RCV に続く診断コマンド列は読み込みデータを伴う．制御語 NOINT の直後の FFFFFFFF はマスク語で続く正解値と観測値とをマスクして比較す

る．すなわち，00000000 を TR レジスタに与えて検査したときの観測値を00000000とし，またFFFFFFFFを TR レジスタに与えて検査したときの観測値をFFF7FFFF とすると，まず (FFFFFFFF). AND. (00000000) と (FFFFFFFF). AND. (00000000) が比較され，次に (FFFFFFFF). AND. (FFFFFFFF) と (FFFFFFFF). AND. (FFF7FFFF) とが比較される．この場合では当然故障が検出されたことになる．

このようにテストページを設計することによって，同じ検査手順に対して異なったデータを与えることができテストページも簡潔になる．なお低級言語で書かれたテストページを診断アセンブラによってビットパターンに変換し，フロッピディスクにあらかじめ格納しておく．

5. 開発結果および検討

ほとんどの保守コマンドの実行時間が1秒以内になるように設計すると，一つの保守コマンドの実行ではフロッピディスクのアクセスを5回以下にしなければならない．保守コマンド処理ルーチンのなかで処理内容が多様なものは READ 保守コマンドと WRITE 保守コマンドである．READ 保守コマンドには，レジスタの内容を読み出すもの，主記憶の内容を読み出すもの，制御記憶の内容を読み出すものおよび診断記憶の内容を読み出すものが必要である．これらを指定するためにサブフィールドを設けて次のようなフォーマットとした．

READ/REG/Reg. name : レジスタ
READ/MEM/memory address (from-to) : 主記憶
READ/CS/memory address (from-to) : 制御記憶
READ/DM/memory address (from-to) : 診断記憶

WRITE 保守コマンドもサブフィールドが必要であり，このようにサブフィールドが必要な保守コマンドは READ 保守コマンドと同様な次のフォーマットにした．

field 0/field 1/…/field m

保守コマンドがキーボードから入力されたとき，保守プロセッサモニタは，モニタ内にもっているディレクトリから field 0 に対応する保守コマンド処理制御ルーチンをオーバレイ領域に読み込んで制御をこのルーチンに渡す．制御ルーチンは内蔵ディレクトリから field 1 に対応する処理ルーチンをオーバレイ空領域に読み込み制御をこのルーチンに渡す．以下同様の処理を行う．

このように保守コマンドおよび保守プロセッサを設計していった結果，m は2までで十分であり，処理の多様な READ/REG 保守コマンドおよび WRITE/REG 保守コマンドではフロッピディスクのアクセス回数は4回となり，これらの保守コマンドの実行時間は約1秒であった．共用処理モードのとき，チャンネルからのメッセージは保守コマンド入力待ちから処理終了までの間チャンネル側で待つようにチャンネルインタフェースを設計したが，これらのメッセージにはシステムの運用に致命的な障害を与えるものはないので実用上問題はなかった．また，保守プロセッサが使用するオーバレイ領域は約 1.5k 語で十分であった．

なお，READ/REG および WRITE/REG 保守コマンドでアクセスできるレジスタはそれぞれ520個と160個であり，対応する手順を格納しておくフロッピディスクの容量はそれぞれ 30k バイトおよび 16k バイトであった．

マイクロ診断プログラムは，保守コマンド diag の入力によってマイクロ診断エクゼクティブの制御ルーチンがオーバレイ領域にフロッピディスクから読み込まれて始まるが，使用するオーバレイ領域は約 1.5k 語で十分であった．また，1テストページを256バイトとすることによって全体のテストページ数は約2100となった．故意に故障を起こさせて実験したところ，仮定した故障の約 90% を検出し，このうち約 85% が適中した．CPU の A, V, C サイクルハードウェアの大部分が結線論理で動作するために，テストページの作成が困難であった．また，シングル CPU で故障のないときのマイクロ診断プログラムの実行時間は約8分であった．予測式では約5.9分となるが，この差はマイクロ診断プログラムの条件設定，トラック移動時間等を無視したために生じたものと思われる．

図6 にチャンネルサービスプロセッサ，保守プロセッサ，割り込みプロセッサおよびマイクロ診断プログラムが動作したときの CRT 画面例を示す．

なお，SVP メモリはベーシックモニタ，チャンネルサービスプロセッサ，保守プロセッサおよび割り込み処理プロセッサに約 3.6k 語使用され，残りの 0.4k 語は機能拡張（たとえばリモート保守のための回線処理等）のために確保してある．また DU を含めた CPU の保守用ハードウェア量は約7%であり，SVP

```
      MEDIA-S#12345-06 MNT 0-12-03 MASTER-TAPE
#### CPU INTERRUPT ####
    # REASON --- FAULT ON FAULT #
    # MULTI CPU ? yes
    # AUTO RECONFIGURATION.  CPU#1 RELEASED #
    # PLEASE FIX CPU#1#
     OUT OF PAPER PR 012
(M)#read/reg/tr
(M)   TR=0000FFFF
    MEDIA-S#12345-06 MNT 0-12-03 MASTER-TAPE
(M)#diag
(D) CPU STAND-ALONE ? yes
(D) SELECT TESTS ? no
(D) START MICRO DIAGNOSTICS.
(D) TEST NO. 001000      OK.
(D) TEST NO. 001100      OK.
(D) TEST NO. 001430
(D) FAULT DETECTED!
(D) PCB NAME. EA,EB
(D) LOGIC NAME  TR→SPM(0)
(D) S/B DATA FFFFFFFF
(D) WAS DATA FFF7FFFF
(M)#bye
```

図 6 画 面 例
Fig. 6 Display of alternative operations.

は従来の操作卓に約10%のハードウェアを追加するだけで実現できた.

6. おわりに

本論文では,48kビットのプログラム用メモリとフロッピディスクを使用した操作性のよい保守プロセッサおよびマイクロ診断プログラムの設計思想と,これらを開発して使用した結果について議論した.

保守プロセッサにおいて,保守コマンドのサブフィールドごとにコマンド処理ルーチンの1モジュールを対応させ,制御ルーチンがサブフィールドを解釈する都度,1.5k語のオーバレイ領域の一部に読み込み実行させてコマンド処理を行った.フロッピディスクからオーバレイ領域に,プログラム,ディレクトリおよびレジスタ読み出し等の手順の読み込み回数を4回以内に設計することによって,保守コマンドを入力したときの応答時間を1秒以内にできた.また,レジスタ読み出し等の手順に簡単な言語(12種類の制御語)を導入することによって,この手順とSVPソフトウェアを分離でき,かつ,レジスタはTR, SPM等のニーモニックで指定できるようになった.保守コマンドでアクセスできるレジスタは約520個であり,保守プロセッサの全体の容量は約100kバイトであった.

マイクロ診断プログラムにおいて,1テストページを256バイトに固定し,テストページの実行中は制御ルーチンおよびテストページ処理ルーチンをオーバレイ領域に常駐させることによって,約2,100テストページの故障のないときの実行時間を約8分にできた.また,テストページに簡単な言語(9種類の制御語)を導入することによってCPUの検査手順とSVPソフトウェアを分離でき,さらに同一の検査手順で検査データ部分を変える制御語によってテストページをコンパクトにでき,マイクロ診断プログラムの全体容量は約550kバイトとなった.

このような設計思想のもとで,実用的な保守プロセッサおよびマイクロ診断等の機能をもったSVPを開発することができた.また間欠故障に対する検出および修復手法の確立等を含めたSVP機能の拡張が今後の課題である.

謝辞 本研究は第一筆者が株式会社東芝に在職中に行ったものであり,ご協力いただいた東芝青梅工場の方々に感謝します.また,本論文をまとめるにあたりご助言を賜った京都大学萩原宏教授に深謝します.

参 考 文 献

1) 山田:コンピュータアーキテクチャ,産業図書,東京 (1976).
2) 中西,西田:下位機種コンピュータにおけるサービスプロセッサの実現手法,日経エレクトロニクス,1978年5月29日号 (1978).
3) DEC: VAX 11/780 CPU Technical Description, EK-KA-780-TD (1978).
4) 岩根,飯田,西中:オペレーションプロセッサ,昭和52年信学全大, p. 1356 (1977).
5) Reilly, J., Sutton, A., Nasser, R. and Griscom, R.: Processor Controller for the IBM 3081, *IBM J. Res. Dev.*, Vol. 26, No. 1, pp. 22-29 (Jan. 1982).
6) 萩原:マイクロプログラミング,産業図書,東京 (1977).
7) Sato, F. and Uchida, Y.: The ACOS Series 77 TOSBAC System 600 Computer, *Toshiba Review*, No. 114, pp. 23-27 (1978).
8) 岩根,佐藤,鈴木,高橋:ACOS 77/600 CPUのマイクロ診断方式,信学技法, Vol. 78, No. 26, pp. 25-34 (1978).
9) 岩根,佐藤:大型計算機における保全性設計,信学論, Vol. 63-D, No. 3, pp. 240-247 (1980).

(昭和59年8月8日受付)
(昭和59年12月20日採録)