多重OS「江戸」の設計と実現†

岡野裕之†† 堀 素史††
中川正樹†† 高橋延匡††

パーソナルコンピュータ上のアプリケーションプログラムで，ビットマップディスプレイやマウスなど，新しいデバイスを使用するものには，異機種との互換性のない場合が多い．これは，OSがハードウェアの規格化を十分に行っていないからである．また，パーソナルコンピュータの性質上，個人のニーズに合せたOSがあってしかるべきだが，異なるOSへの移行が容易でないために，一度採用されたOSが新しいニーズに十分対応できずにいるという現状がある．我々は，これらの問題に対処するために，ハードウェアとOSの間のハイパOS層に当たるソフトウェア，多重OS「江戸」を設計・実現した．「江戸」は，前者の問題を解決するために，ハードウェアを抽象化し，仮想マシン・インタフェースを設定した．また，後者の問題を解決するために，複数のOSを同時に実行する多重OS環境を実現した．「江戸」の設計は，我々が研究・開発している，日本語OS OS/omicronの実行を前提として行った．「江戸」の実現によって，OS，アプリケーションプログラムのマシン独立性を高め，OSの移行を容易とした．さらに，OSのデバッグ環境を提供した．本論文では，多重OS「江戸」の設計と実現について報告する．そして，パーソナルな環境に固有の問題を議論する．

1. はじめに

近年のパーソナルコンピュータの発展は目覚ましく，そのハードウェアには多種多様なものがあり，移り変りも激しい．また，パーソナルコンピュータにおいては，同一のOSによる環境であっても，異機種間の互換性の少ない場合がある．このため，アプリケーションプログラムの開発には，異機種への移植という非本質的な部分の負担がかかることになり，これを嫌って単一機種のみを対象にするものも多い．このような状況は，アプリケーションプログラムの発展や普及を妨げることになり，好ましいことではない．

ところで，パーソナルコンピュータに限らず，OSの仕様変更，あるいは異種OSの採用は，多くのアプリケーションプログラムを動作不能としてしまう．これらすべてを新OS上に移植する労力はかなりのものとなり，このため，容易にOSを変更できないという状況が生じている．これに対し，汎用大型計算機においては，仮想マシンによって旧OSと新OSを同時に運用し，新OS上で動作不能なアプリケーションを救済する一方，新OSの開発および環境整備を同時進行させる場合が多い．

我々は，パーソナルコンピュータにおける上記の問題を認識し，次のような方針でこれを解決しようと考えた．

（1） ハードウェアを抽象化し，仮想的なインタフェースを設定する．その上でOSを運用する．

特に，最近のパーソナルコンピュータでは一般的となった，ビットマップディスプレイやマウスの仮想化を行う．また，フォントやキーコード入力のインタフェースも規格化する．

（2） 仮想マシンによって，複数のOSを同時に実行する．

従来汎用大型計算機で使われてきた技術をパーソナルコンピュータに適用する．これによってOSの開発を容易とする．

また，別の問題として，パーソナルコンピュータの性能向上の必要性がある．新しいプロセッサを開発することなしに性能向上を実現する方法として，マルチプロセッサシステムにすることが考えられる．ハードウェアのコストが急激に下がっている現在，プロセッサを5つ付け，2倍の性能を得るといったハードウェアが出現しても不思議はない．その点，プロセッサの個数に対して効率よく性能の得られる方法ではないが，現在の技術で十分実現可能な，共有メモリ方式を採用するのが得策である．そこで，上記に加え，次の方針も採り入れた．

（3） 共有メモリ方式のハードウェアで実現可能とする．

† Design and Implementation of a Hyper OS Edo by HIROYUKI OKANO, MOTOFUMI HORI, MASAKI NAKAGAWA and NOBUMASA TAKAHASHI (Department of Information Science, Faculty of Technology, Tokyo University of Agriculture and Technology).

†† 東京農工大学工学部数理情報工学科

これらの方針によって今回作成したソフトウェアを，我々は多重 OS「江戸」と呼んでいる．「江戸」は，高橋研究室で自作している日本語 OS OS/o (omicron)[1)~3)] の実行環境を提供できるよう設計され，市販のシングルプロセッサのパーソナルコンピュータ上に実現された．「江戸」の開発により OS/o の移植性を高めることができ，「江戸」を市販のパーソナルコンピュータ上で動作させたことにより OS/o の公開を容易とした．本論文では，「江戸」の設計および実現について報告する．

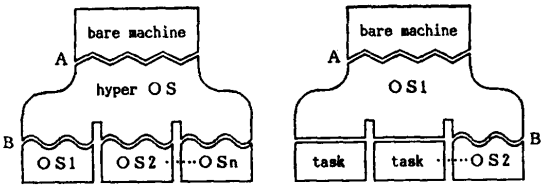
2. 多重 OS「江戸」の設計

多重 OS「江戸」の設計は，ハードウェアを抽象化する仮想マシン・インタフェースの設計と，多重 OS 機能（複数の OS を実行する機能）を実現するカーネルの設計に分かれる．本章では，「江戸」の基盤となる多重 OS 機能の方式設計について述べたあと，仮想マシン・インタフェースの設計，カーネルの設計について述べる．

2.1 多重 OS 機能の方式

多重 OS 機能を実現している既存の仮想マシンには，大きく分けて２つの方式がある．その１つは，図 1（a）のように，各 OS に同一の仮想マシン・インタフェースを与える方式である．この方式は汎用大型計算機の仮想マシン[4)]によく用いられるもので，各 OS のパフォーマンスは同程度によい．また，各 OS が独立していて依存関係がないため，OS 間の絶縁性が高い．この方式では，図中のインタフェース A をハイパ OS が仮想化し，B とすることによって，各 OS のマシン独立性を高められる．

もう１つの方式として，図１（b）のように，ある OS の１つのタスク（あるいはプロセス）としての別の OS を実行するものがある．この方式による実現例として，NEC PC-98XL^2 上の UNIX で MS-DOS をエミュレートしたもの[5)]がある．これは，MS-DOS の実行中に発生する例外（IN/OUT 命令によるものを含む）をエミュレートすることによって，MS-DOS に裸のマシンとほぼ等価なインタフェースを提供し，市販の PC-98 用アプリケーションを UNIX 上で実行する．図１（b）の方式は，このように，ユーザやアプリケーションを新 OS 上に移行する手段として用いられている．この方式では，ハイパ OS として機能する OS１が裸のマシンに直接載るため，この OS１のマシン独立性は高められない．



(a) Virtual machines are provided at the same level.

(b) Virtual machine(s) are provided at the task level of OS1.

A : bare machine interface.
B : virtual machine interface.

図 1 多重 OS 機能を実現する方式
「江戸」の採用した方式は(a)である．
Fig. 1 Methods to execute a plural set of OS's. Edo has employed the method (a).

「江戸」は，各 OS のマシン独立性を高めることを目的の１つとしたため，図１（a）の方式を採用した．

2.2 仮想マシン・インタフェースの設計

多重 OS 機能を実現するためには，デバイス（入出力装置），メモリ，プロセッサなどのハードウェア資源を，各 OS に分配する必要がある．そこで，デバイスを管理するために，「江戸」にデバイスドライバ群を持たせることにした．これによって，一連の不可分な入出力ポート操作を「江戸」の中に閉じ込められるだけでなく，フロッピディスクのように複数 OS で共有できないデバイスの使用権を管理したり，１つのデバイスを仮想的に複数個に見せたりできる．このデバイスドライバ群と OS とのインタフェースを，仮想マシン・インタフェースと呼ぶことにする．

第１章で述べたように，パーソナルコンピュータでは，異機種間におけるアプリケーションプログラムの可搬性が乏しい．これは，デバイスの仮想化が行われていないことによる．これは本来 OS の仕事であるが，ビットマップディスプレイやマウスといった，新しいデバイスへの対応がなされていない場合が多く，上記の問題を引き起こしている．

パーソナルコンピュータには，その性質上，個人のニーズに合せた様々な機種が存在し，各々多様なデバイスを持っている．したがって，これらのハードウェアをすべて包含する仮想的なインタフェースなどはあり得ないはずである．逆に言えば，ある一定の枠に固定できないこと，つまり，フレキシビリティこそがパーソナルコンピュータの特徴である．しかし，限定した範囲のデバイスに関しては仮想化を行うべきであり，この範囲は時と共に変更する必要がある．OS はこのことを考慮に入れて，拡張・変更可能な構成とする必要がある．

OS がデバイスの仮想化を行う場合，機種間の差異を吸収する層と，さらに高度な仮想化を加える層に分けることができる（図 2）．「江戸」はハードウェアと OS の間のハイパ OS と呼ばれる部分に位置し，前者の層に属する仮想化を行う．よって，「江戸」が提供する仮想マシン・インタフェースは，後者の層における仮想化の自由度を高く保つように，抽象化のレベルの低いものとした．また，その仕様は，OS/o, あるいは OS/o がターゲットとするアプリケーションプログラムに適するようにした．

「江戸」の仮想マシン・インタフェースは，スーパバイザコール（SVC）の形とした．たとえば，OS 側からあるデバイスをアクセスする場合，SVC によって「江戸」のデバイスドライバを利用することになる．このほかの方法としては，M 68000 系の入出力がメモリマップドであることを利用し，MMU (Memory Management Unit) のインバリッドフラグによって参照不可としたメモリに仮想的な入出力ポートを対応付け，そこへのアクセスをメモリフォルトによって検出してエミュレートする方法を考えた．このような，入出力ポート単位のエミュレートは，裸のマシン自体が広く流通していて，暗黙の規格として通用するような場合に有効であり，実現例として上記の文献 5) がある（ただし，文献 5) は 80386 CPU の IN/OUT 命令をエミュレートしている）．しかし，この方法では余りにも低レベルなインタフェースとなること，また性能が前者の方法に比べて上がらないことから，SVC による方法を採用した．「江戸」と同様な，SVC による仮想マシン・インタフェースを採用した例として文献 6)，文献 7) がある．しかしこれらは，ハイパ OS にタスク管理を含めた形態をしており，仮想マシン・インタフェースの抽象度を低く抑え，OS に高い自由度を与える「江戸」の設計とはかなり異なっている．次章で述べるように，OS/o は日本語文字コード体系を採用し，アプリケーションに開かれたタスク制御を目指している[8]．したがって上記の機能に関しては，完全に OS に任せるべきであると判断した．

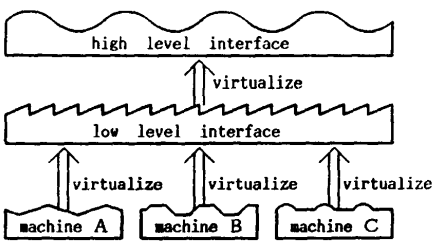


図 2 ハードウェアの仮想化の層
Fig. 2 Layers of hardware virtualization.

2.3 カーネルの設計

図 1 (a) のようなハイパ OS を設計する場合，各 OS を時分割で動作させる方法と，割込みが起こるたびにその行く先の OS にスイッチする，事象駆動による方式が考えられる．「江戸」はパーソナルユースが前提であるため，プロセッサを無駄なく使用できる，事象駆動方式を採用した．

我々が開発中の OS/o は，次章で述べるとおり，モトローラ社の M 68000 ファミリ[9]をターゲットプロセッサとしている．よって「江戸」のターゲットプロセッサも M 68000 ファミリとした．M 68000 ファミリの各プロセッサは，例外スタックフレーム（例外発生時に積まれるスタック）の仕様がそれぞれ異なっており，その部分で OS がプロセッサ依存になる場合がある．そこで，「江戸」が提供する CPU 環境ではターゲットプロセッサによらず，割込みの仕様を MC 68000 のものとした．

「江戸」は，共有メモリ型のマルチプロセッサシステムで実行可能なように設計した．マルチプロセッサシステム上で OS を構築する際，OS がプロセッサ識別子（ID）を知る必要があると考えられたので，これを提供することにした．また，ハードウェアのモデルとして，入出力機能は各プロセッサに平等にあるのではなく，プロセッサごとに違ったデバイスがつながっていると仮定した．

以下の章では，OS/o の概要を述べたあと，多重 OS「江戸」の実現とその結果について述べる．

3. OS/o について

当研究室では，OS/o と呼ばれる OS の研究・開発を行っている．OS/o の特徴としては次のものが挙げられる．

（1） 研究者を対象としたパーソナルコンピュータ用の OS である．

このため，シングルユーザ，マルチタスクで，ソースコードの公開された OS となっている．また，実時間処理に対応するため，実記憶方式を採用している．

（2） 日本語情報処理に適している．

OS/o は，標準文字コードとして JIS 2 バイトコード（フル 2 バイト）を採用しており，ファイル名やディレクトリ名に日本語が使用できる．これは，システム記述言語として開発された言語 C コンパイラ CAT

(C Compiler developed at Tokyo University of Agriculture and Technology) のフル2バイト化によって実現している[8),10),11)].

(3) アプリケーション指向の OS である.

OS/o はアプリケーションとして，やはり当研究室で研究を行っている，日本語文書処理システム「浄書」(JOSHO)[12),13)]，オンライン日本語手書き文字認識 JOLIS，日本語入力エディタ，CAI などを対象にしている.

(4) ターゲットプロセッサとしてモトローラ社の M 68000 ファミリを採用している.

M 68000 は広くリニアなアドレス空間を持つため，実記憶系に適し，長期の研究に耐えうると判断して採用した.

OS/o は今まで，MC 68000 のシングルボードコンピュータ上で，開発環境用の OS として CP/M-68 K を使って開発してきた. このハードウェアはビットマップディスプレイを持たないため，ユーザインタフェースの研究を行う際に不都合な面があった. このため，OS/o を，高解像度のビットマップディスプレイと高い CPU パワーを持ったマシンに載せることが望まれた. また，知的所有権が問題となっている現在，市販のパーソナルコンピュータ上に OS/o を実現し，教育用に公開したいという考えもあった. 多重 OS「江戸」は，OS/o をパーソナルコンピュータ上で実現し，しかも，今までの開発環境である CP/M-68 K から，OS/o の研究開発および先に述べた OS/o のアプリケーションのすべてを，OS/o 上へ移行することを容易にする.

4. 多重 OS「江戸」の実現

「江戸」は，**表 1** のような仕様を持つ，市販のパーソナルコンピュータ上に実現した. これらのマシンは，OS/o のターゲットプロセッサである M 68000 ファミリを採用したシングルプロセッサシステムである.「江戸」は共有メモリ型のマルチプロセッサシステムを仮定して設計したのだが，シングルプロセッサでの実現となった.

「江戸」は，多重 OS 機能を実現するカーネルと，ハードディスクや RS-232 C，ビットマップディスプレイなどのデバイスを制御し，その機能を OS に提供するデバイスドライバによって構成されている (**図 3**).「江戸」のすべてのモジュールの記述は，アセンブリ言語によって行った (**表 2**). 以下，カーネルの動作原理，メモリ管理などについて述べたあと，デバイスドライバについて述べる.

表 1 多重 OS「江戸」のターゲットマシンの仕様
Table 1 Specification of the target machine of Edo.

「江戸」初版	
プロセッサ	MC 68010 (10 MHz)
コプロセッサ	なし
主記憶	4 Mbytes
内蔵ハードディスク	40 Mbytes
フロッピディスク	3.5 インチ (HD) 1 台，8 インチ (2D) 1 台
表示能力	1120×780 (16 色)
その他	マウス，漢字フォント ROM

「江戸」第二版	
プロセッサ	MC 68020 (20 MHz)
コプロセッサ	MC 68881 (20 MHz)，URR プロセッサ
主記憶	8 Mbytes (最大 16 Mbytes)
内蔵ハードディスク	88 Mbytes
フロッピディスク	3.5 インチ (HD) 1 台
表示能力	1120×780 (16 色)
その他	マウス，漢字フォント ROM

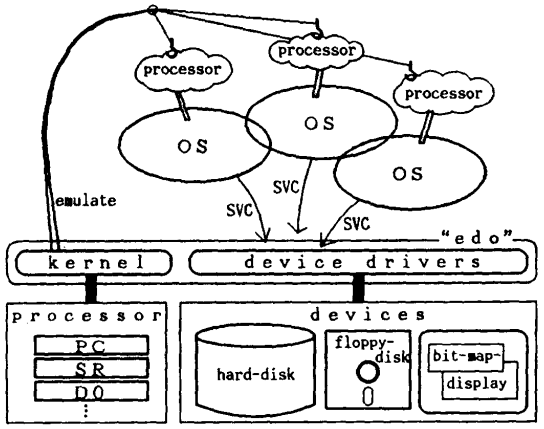


図 3 多重 OS「江戸」の構成
Fig. 3 Structure of Edo.

4.1 「江戸」の動作原理

「江戸」がターゲットとするプロセッサは，OS/o と同じく，M 68000 ファミリである (ただし，MC 68010 より上位のもの). M 68000 は，ユーザ状態とスーパバイザ状態という 2 つの特権状態を持っており，いくつかの特権化された命令 (特権命令) は，ユーザ状態では実行不可となっている. 特権命令をユーザ状態で実行しようとすると，特権違反例外が起こり，スーパバイザに制御が移るようになっている.「江戸」はこの機能を利用して，多重 OS 機能を実現する. つま

表 2 多重 OS「江戸」第二版の記述行数
(アセンブリ言語)
Table 2 Static steps of Edo (in the assembly language).

分 類	名 称	行 数
カーネル	特権命令エミュレータ	518
	割込みエミュレータ	1488
	その他(コマンドインタプリタなど)	5447
デバイスドライバ	ハードディスク	926
	フロッピディスク	1455
	RS-232 C	700
	ウィンドウシステム	7522
	その他 (キーボード, タイマなど)	1721
		計 19777

り, OS をユーザ状態で走らせ, OS の実行する特権命令をエミュレートすることによって, CPU 資源を各 OS に分配する (図 4)[4].

OS に対して行われる例外処理は, すべて「江戸」がエミュレートする.「江戸」における例外処理の流れを図 5 に示す. 図中の COSCB, OSCB は,「江戸」の持つ OS 管理表であり, 表 3 のような構造を持つものである. 図 5 に示すように, COSCB はスーパバイザスタック中の一定位置に書き込まれており, そこからのポインタによって, その時点でアクティブとなっている OS の OSCB が示される. これらの OS 管理表の構造は,「江戸」をマルチプロセッサで実現することを考慮して決定した. マルチプロセッサで実現する場合, OSCB はグローバルな管理表として 1 か所にまとめ, 各プロセッサの持つ COSCB がこれをポイントし, 参照するようにする.

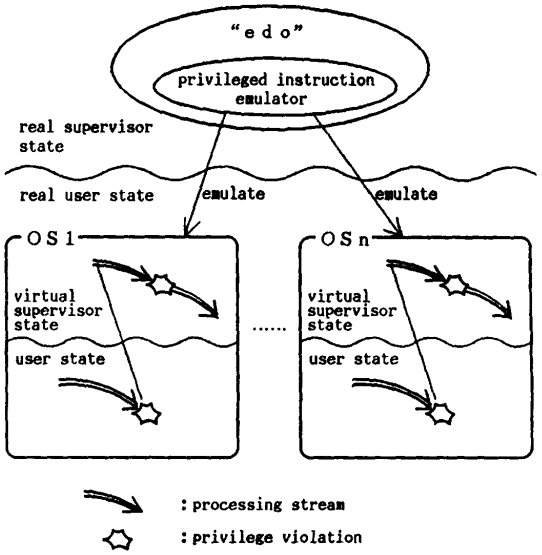


図 4 多重 OS 機能の概念図
特権命令のエミュレートによって, 仮想的なスーパバイザ状態が作られる.
Fig. 4 Conceptual structure of the virtual machines.
Each virtual supervisor state is realized by emulation of privileged instructions.

例外が発生すると, 図中左側にある実際のベクタテーブルがプロセッサによって参照され,「江戸」内にある各例外処理ルーチンへ制御が移る.「江戸」が OS に対して例外を発生させる場合は, 各 OS の持つベクタテーブルを「江戸」が参照し, 例外をエミュレートする. このとき, 例外を起こすべき OS がその

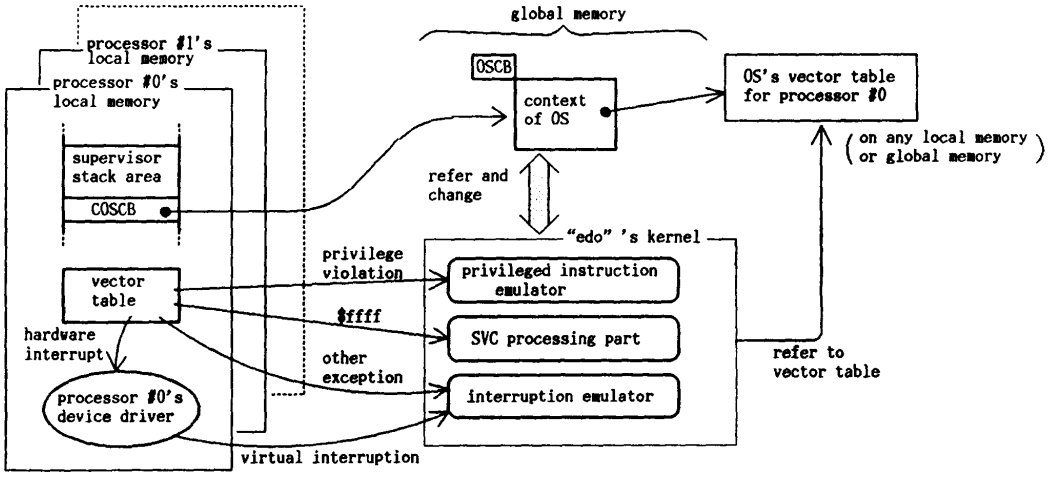


図 5 多重 OS「江戸」の例外処理
Fig. 5 Exception handling by Edo.

表 3 OS 管理表の内容
Table 3 Contents of the OS control tables.

COSCB (Current OSCB)	
processor_ID	プロセッサ ID
active_OSnum	現在アクティブな OS の OS 番号
active_OSCB	現在アクティブな OS の OSCB のアドレス
top_of_OSCB	OSCB の先頭アドレス
max_OS	起動可能な OS の最大数
—	初期コンテキスト (A 0-A 7)

OSCB (Operating System Control Block)	
OS_name	OS の名前 (JIS X 0208 で 16 文字+null)
OS_number	起動される順に, 0, 1, 2, …と振られる番号
processor_state	プロセッサ状態 (通常, 停止, 例外, ホールト)
system_byte	SR の上位バイト
swapped_USP	スワップされている USP
swapped_SSP	スワップされている SSP
vector_table	ベクタテーブルのアドレス
SVC_pointer	SVC に用いるアドレスレジスタ (0〜7)
active_COSCB	COSCB へのポインタ
—	非実行中の D0-D7/A0-A7

時点でアクティブでない場合, COSCB を書き換え, アクティブ OS を変更 (OS スイッチ) してから例外をエミュレートする. このように,「江戸」では OS への例外をすべてエミュレートによって行うため, 割込みデバイスの仮想化が可能となっている. (これについては, あらためて述べる.) 反面, すべての例外をエミュレートしなければならず, オーバヘッドが大きいという欠点がある.

4.2 「江戸」の提供する CPU 環境

多重 OS「江戸」初版は MC 68010 上に, 第二版は MC 68020 上に実現されているが,「江戸」上で動作する OS に提供される CPU 環境は, これらターゲットプロセッサと同一ではない. ここで言う CPU 環境とは, 次のようなことを指している.

- レジスタセット
- 使用可能な特権命令
- 例外スタックフレームの内容
- 「江戸」への SVC 用の命令

レジスタセットは, ユーザ状態でアクセスできるものに関してはすべて同一であるが, 特権命令によってアクセスする一部のレジスタが省略されている. これは一部の特権命令 (movec, moves 命令) のエミュレートを行っていないからで, これによって処理を簡単にしている.

例外スタックフレームが実際のプロセッサと異なるのは,「江戸」の大きな特徴と言える. 具体的には, MC 68000 が生成する例外スタックフレームをそのままエミュレートしている. M 68000 シリーズのスタックフレームは, MC 68000 と MC 68010 以降のもので仕様が変更されており, 普通, MC 68000 上で動作する OS を MC 68010 以降のプロセッサに持ってきてもうまく動作しない. OS/*o* の開発機, および最初のターゲットマシンが MC 68000 のシングルボードコンピュータであったために, 我々は OS/*o* を含めて MC 68000 用のソフトウェアを多く所有している. そのため「江戸」は, これらのソフトウェアを生かすべく, プロセッサによらず例外スタックフレームを MC 68000 仕様としている.

「江戸」では, ビットパターン $ffff (アセンブリ言語の "dc −1" により生成される) を,「江戸」への SVC 用の命令として使用している. この命令は, M 68000 の命令セットのうち, コプロセッサ用の命令として使用されているが, ユーザ定義のコプロセッサ用として用意されているものであり,「江戸」が使用しても差し支えないものである[14].「江戸」への SVC は, 機能番号と引数を特定のアドレスレジスタによって指したのち, dc −1 で設定した $ffff を実行することによって行われる. このときに使用されるアドレスレジスタは, OS の起動時に指定できる. この仕様は, 言語 C からの利用を考慮して決定した. A 7 レジスタを使った SVC の例を次に示す.

```
move    引数2, −(A7)
move    引数1, −(A7)
move    機能番号, −(A7)
dc −1
move    (A7)+, 戻り値
addq    #4, A7
```

4.3 「江戸」のメモリ管理

現段階の「江戸」では, メモリ管理を全く行っていない.「江戸」の下で動作する OS は, あらかじめ, 互いに競合しないようにアロケートしておかなければならない.「江戸」がメモリ管理を行わない理由として, 次のことが挙げられる.

(1) OS/*o* が実記憶方式を採用しているため,「江戸」も実記憶方式を用いる.

(2) OS どうしが互いにメモリを参照し合ったり, メモリを介した通信を行ったりしたい.

上記(1)の理由から, 実記憶方式を採用しているのであるが, 実記憶の枠組みの中でも, MMU を用いた

空間の組替えの機能をOSに提供するような選択もあった．しかし，上記(2)のような理由から，単一空間上にOSを配置することにした．これによって，OS/o第二版を，OS/o第一版がトレースしたりしてデバッグすることが可能となる．ただしこの場合でも，安全上の理由から，ページごとのアクセス属性をOSに設定させたり，バウンダリレジスタの機能を提供したりすべきであると考える．これは，公開する時点までには当然実現する予定である．

4.4 デバイスドライバ

「江戸」には，ハードディスク，RS-232C，ビットマップディスプレイ，マウスなどのデバイスドライバが組み込まれており，ユーザ(OS)はSVCによってこれらを利用できる．デバイスドライバをアクセスするときのSVCの仕様が，すなわち「江戸」の仮想マシン・インタフェースである(**表4**)．デバイスドライバのうち，ビットマップディスプレイとマウスを管理している部分は特に，ウィンドウシステムと呼ばれ，大きな位置を占めている[15]．ウィンドウシステムの詳細は次節に回し，ここではそれ以外のデバイスドライバについて記述する．

ウィンドウシステム以外のデバイスとしては，現在次のものが実現されている．

- キーボード
- ハードディスク
- フロッピディスク
- カレンダ
- RS-232C

これらのデバイスの，仮想マシン・インタフェースの決定は，なるべくハードウェアイメージを残すという方針で決定された．これは，「江戸」によるハードウェアの抽象化を低レベルなものにし，OSによる抽象化の余地を多く残した方が，OSにとって自由度が高いだろうという考えに基づいている．次にこの特徴を列挙する．

(1) キーボード入力では，キーコードが入力できる．

(2) キー割込みが受けられる．

(3) ハードディスク，フロッピディスクは，処理の終了報告を割込みによって行う．

(4) ハードディスクのOSごとのパーティションは「江戸」で管理せず，各OSはハードディスク全体を自由にアクセスできる．

(5) フロッピディスクはユニットのロック，アンロックの機能を提供し，OSごとの排他制御を行う．

上記(1)で言うキーコードとは，ハードウェアが固有に持つものである．キーコードが入力できることによって，“変換キー”など，JISコードに割り当てら

表4 多重OS「江戸」のSVC一覧
Table 4 List of SVC's to Edo.

システム関係	edo_processor	プロセッサIDを得る．
	edo_getVBR	ベクタテーブルのアドレスを得る．
	edo_other_os	CPU割当てを放棄する．
	edo_os_int	指定したOSに割込みをかける．
	edo_my_name	自分のOSの名前を得る．
	edo_memory_size	メモリサイズを得る．
ウィンドウシステム	edo_set_kb_int	キー割込みを設定する．
	edo_keycode	キー入力を一文字得る．
	edo_beep	ブザーを鳴らす．
	edo_set_mu_int	マウス割込みを設定する．
	edo_mouse_mode	マウス割込み要因を設定する．
	edo_getVFM	仮想FMのアドレスと大きさを得る．
	edo_disp_position	仮想FM上の表示領域を設定する．
	edo_area_put	仮想FM上の更新領域を報告する．
	edo_char_put	文字を仮想FMに展開する．
	edo_font_address	フォント・アドレスを計算するサブルーチンのアドレスを得る．
	edo_disp_size	表示領域の大きさを得る．
	edo_mouse_status	マウスの状態を得る．
	edo_get_mu_int	マウス割込みの結果を得る．
	edo_mouse_cursor	マウスカーソルパターンを登録する．
ディスク	edo_set_fd_int	フロッピディスク(FD)の割込みを設定する．
	edo_set_hd_int	ハードディスク(HD)の割込みを設定する．
	edo_fd_command	FDに対してコマンドを発行する．
	edo_hd_command	HDに対してコマンドを発行する．
	edo_get_fd_int	FD割込みの結果を得る．
	edo_get_hd_int	HD割込みの結果を得る．
	edo_fd_lock	FDをユニット単位でロックする．
	edo_fd_unlock	FDをユニット単位でアンロックする．
日時	edo_gettime	日付時間情報を得る．
	edo_settime	日付時間情報を設定する．
RS-232C	edo_set_rs_int	RS-232C割込みを設定する．
	edo_Rx_buffer	RS-232C受信バッファを設定する．
	edo_Tx_buffer	RS-232C送信バッファを設定する．
	edo_setBPS	RS-232CのBPSを設定する．
	edo_Tx_trigger	RS-232Cの文字送出のタイミングを報告する．
	edo_rs_lock	RS-232Cをチャンネル単位でロックする．
	edo_rs_unlock	RS-232Cをチャンネル単位でアンロックする．

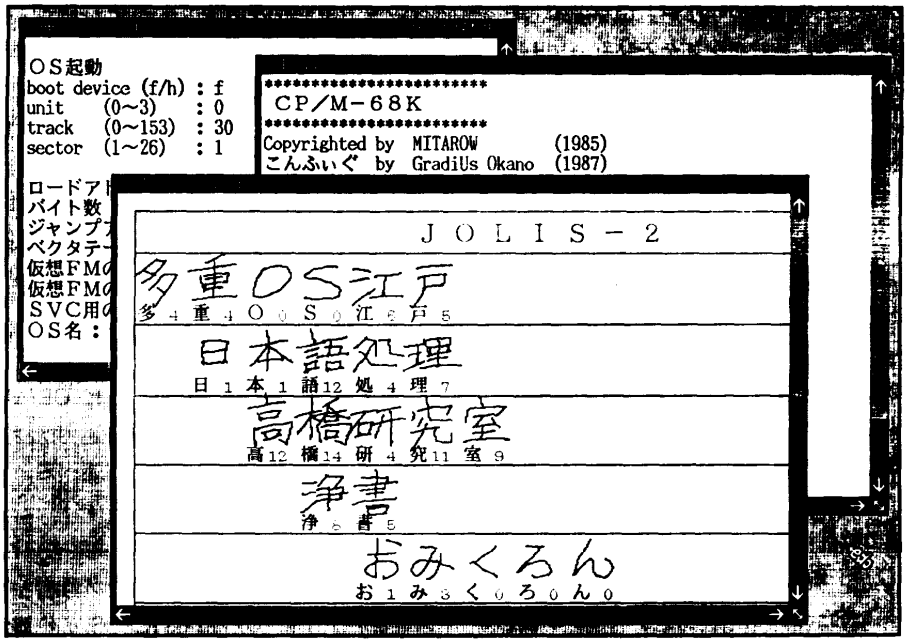


図 6 「江戸」の画面のハードコピー
Fig. 6 A hard-copy of the screen of Edo.

れていないキーの入力ができるようになるほか，ノートオン，ノートオフで別のキーコードを用意することによって，キーの上げ下げの検出も可能となる．(キーの上げ下げに関しては，ハードウェア上の制約から，現在は実現されていない．)「江戸」が提供するキーコードはハードウェア固有のものであり，標準キーコードのような規格は用意しなかった．というのも，キーボードは多種多様であり，標準化が難しいからである．無理をして標準化を行い，ハードウェア（キーボード）の選択の幅を狭めるよりも，ソフトウェアでキーコードテーブルを持つなどして対処する方が得策と考えた．

上記(2)，(3)にあるように，「江戸」の仮想マシン・インタフェースは，ハードウェアイメージに合せて割込みを用いている．つまり「江戸」は，OS 上の割込み（例外）をすべてエミュレートによって行っているため，実際には発生していない仮想的な割込みを OS に対して起こすことができ，これをデバイスドライバが利用している．これによって実際のハードウェアとのギャップを少なくすることができ，慣れやすいインタフェースとなっている．

4.5 ウィンドウシステム

「江戸」が提供する仮想マシン・インタフェースのうちで，ビットマップディスプレイとマウスのインタフェースを提供するのがウィンドウシステムである．

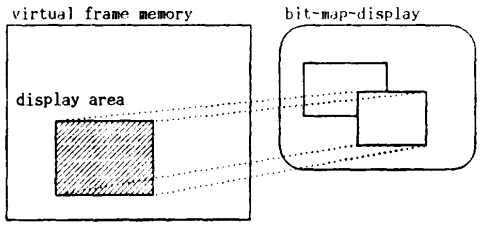


図 7 仮想フレームメモリ上の表示領域
Fig. 7 Display area on virtual frame memory.

「江戸」のウィンドウシステムは，オーバラッピング方式を用いたマルチウィンドウを採用している（図 6）．各ウィンドウには，OS が1つずつ持っている，仮想フレームメモリの一部が表示される（図 7）．つまり，「江戸」のウィンドウシステムは，各 OS に1つずつウィンドウを開くもので，OS のタスクごとにウィンドウを開くものではない．すなわち，「江戸」が提供する仮想フレームメモリは，あくまでも個々のハードウェアの差異を吸収しただけの低レベルなものにとどめ，そのインタフェースはビットマップを直接操作するという形式をとる．そして，仮想フレームメモリの上で OS が別のウィンドウシステムを走らせ，さらに高度な仮想化を加える．

「江戸」が OS に対して複数のウィンドウを提供するという選択もあった．この場合 OS 上のウィンドウシステムは，マルチウィンドウ処理を行わなくて済

む．しかし，こうしたのでは，「江戸」の環境でしか動作しない特殊なウィンドウシステムになってしまう．「江戸」の，OS のデバッグ環境という立場からも，仮想マシン・インタフェースはハードウェアイメージに近いものにし，「江戸」以外の環境への移行までも考慮すべきであると考えた．この方針をもとに，仮想フレームメモリは OS に対して1つずつ提供することにし，その仕様は，白黒で任意の大きさとした．

OS がウィンドウに何かを表示する方法として，「江戸」は2種類の SVC を提供している（表4参照）．1つは edo_char_put で，これにより仮想フレームメモリ上の任意の位置に，引数の JIS コードで示された文字を展開し，表示する．このとき，ウィンドウの重なり，表示領域の位置関係によってクリッピングが必要な場合は，「江戸」が自動的にこれを行う．もう1つの方法は edo_area_put で，あらかじめ OS が仮想フレームメモリ上に展開した文字，図形などを表示するときに使用する．引数は仮想フレームメモリ上の矩形領域の左上と右下の点の座標で，この領域が表示される．

「江戸」のウィンドウシステムの機能は，ウィンドウの表示のほかに次のものがある．

（1） キーボード入力をどの OS に向けるかを決定する．

（2） ウィンドウ上で行われるマウスのクリック，マウスの移動を OS に伝える．

（3） 文字フォントを提供する．

上記(1)は，重なりの一番上になるウィンドウにキー入力を向けることにしている．また(2)は，割込みによってそのタイミングを伝えることによって実現している．マウスのクリックにはウィンドウ上のそれのほかに，ウィンドウの枠に書かれた矢印のクリックも含まれる．計4つある矢印へのクリックは，別々のイベントとして入力される．(3)の文字フォントについては，我々が研究・開発している日本語フォーマッタ「浄書」の実行に必要なものを提供することにした．「浄書」は，JIS 第1水準，第2水準の漢字フォントセットを2種類（24×24 dot, 32×32 dot）必要とする．このうち，ターゲットマシンには 32×32 dot のフォント ROM が装備されていないため，システム立上げ時に，ハードディスクからメモリ上にロードすることにした．

仮想フレームメモリの仕様も同様に，「浄書」とのインタフェースを考慮して決定した．「浄書」は，LBP を出力装置に持つオフライン型の日本語フォーマッタであるが，この出力を「江戸」を通じてディスプレイに表示できれば，プレビュとして利用できるほか，WYSIWYG への発展なども望めることになる．「浄書」のフレームメモリは白黒で 4096×4096 dot なので，「江戸」の仮想フレームメモリは，白黒で任意の大きさの矩形領域ということにした．

5. 多重 OS「江戸」の評価

本章では，実際に「江戸」上に CP/M-68 K，OS/*o* を移植した際に得られた成果を述べたあと，「江戸」の問題点と今後の課題について述べる．

5.1 「江戸」の成果

「江戸」によって次のような成果を得た．

（1） OS 移植の際のデバッグ環境としての効果

「江戸」上に CP/M-68 K，OS/*o* を移植してみて，「江戸」の多重 OS 機能が非常に有用であることを確認した．OS/*o* のデバッグには，CP/M-68 K 上のデバッガを利用でき，メモリダンプやディスアセンブルが行えた．さらに，「江戸」に簡単なデバッグ機能を付加し，OS に割込みが起こったかどうかをモニタするなどを行った．「江戸」がターゲットとしたマシンを含め，市販のパーソナルコンピュータには ROM デバッガが付いていないことが多く，これに OS を載せるのは大変である．このような場合，ハードウェアと OS の間に「江戸」のような層を置くことが，有効な手段になる．

（2） 開発環境の移行の容易化

現在，「江戸」の多重 OS 機能を利用して，メモリを介した CP/M-68 K，OS/*o* 間の通信が暫定的に行われており，ファイル転送が可能となっている．これは，開発環境の移行を行う上で重要な機能であった．これがない場合，どちらかの OS 上で他方の OS のファイルシステムをシミュレートする必要があった．

（3） OS とアプリケーションプログラムの可搬性の向上

「江戸」には初版と第二版があるが，これらの上では全く同じ OS とアプリケーションプログラムの動作が可能である．初版と第二版の「江戸」のターゲットマシンは，プロセッサが異なるし，入出力機能も異なる．しかし「江戸」が，CPU 環境を含めた仮想マシン・インタフェースを提供するために，完全な互換性が保たれた．

（4） OS/o 新バージョンの開発環境の提供

OS/o の次のバージョンは，「江戸」上で動作している現バージョンの OS/o を開発環境とし，「江戸」上にも実現する予定である．OS の開発には，OS のデバッグ環境のほかに，現行 OS からのすべてのソフトウェア資産の移行を容易とする手段が必要であり，「江戸」が有効である．

（5） OS 間通信の実験環境の提供

今後の「江戸」の活用法として，1つのマシン上で OS/o を2つ走らせ，OS 間通信の実験を行うことなどがある．現バージョンの OS/o は OS 間通信の機能を持っておらず，ハードウェアも持ち合せていないが，導入するハードウェアのモデルさえ決まれば，「江戸」によってそれをエミュレートできる．

5.2 「江戸」のオーバヘッド

「江戸」の主なオーバヘッドは，特権命令のエミュレート，割込みのエミュレート，ウィンドウの表示であった．特権命令のエミュレートのオーバヘッドは，ori to SR (SR レジスタと即値の論理和をとる) 命令を「江戸」第二版上で計測した結果，実計算機上で 0.8 μs のところを，「江戸」上では 106 μs であった．これは，OS のタスクスイッチ時間などに直接影響する要因となる．割込みのエミュレートの例として trap 命令を計測した結果，実計算機で 1.9 μs のところを，「江戸」上では 103 μs であった．trap 命令は OS への SVC に使われる命令なので，SVC を多く使用するアプリケーションほど，オーバヘッドが効いてくる．言語Cコンパイラの例では，約 10% のオーバヘッドとなった．

ウィンドウの表示のオーバヘッドは，仮想フレームメモリと実フレームメモリの二段階の転送を経て表示するために，かなり大きい．しかし，文字列を表示する際，一文字ごとに画面更新の SVC (edo_area_put) を行わず，文字列を仮想フレームメモリに展開してから一回だけ画面更新を行う，というようにしてオーバヘッドを抑えることができた．また，テキストをスクロールする際には広い領域に対して画面更新しなくてはならないが，一度に3～4行スクロールすることによってオーバヘッドを抑えた．

5.3 「江戸」の仕様の有効性

「江戸」の持つ次のような仕様について，その有効性を考えてみる．

（1） 多重 OS 機能を図1(a)の方式で実現した．

実際に「江戸」を運用した結果，CP/M-68 K，または OS/o だけを起動して使用する場合も多く，OS の起動順序が自由なこの方式が有効であると感じられた．また，この方式では OS 間の絶縁性が高く，1つの OS の状態を完全に独立して他の OS から参照できるため，OS のデバッグに有効であることが判明した．

（2） SVC による抽象度の低い仮想マシン・インタフェースを提供した．

抽象度を低く抑えたことにより，別の計算機で裸のマシン上に載っていた CP/M-68 K と，別の仮想マシン上に載っていた OS/o を，「江戸」上に簡単に移植することができた．特に，ディスクの抽象度が低いため，「江戸」上で構築するファイルシステムに，高い自由度を与えることができた．

（3） 事象駆動による OS スイッチ方式を採用した．

汎用大型計算機では，各 OS を別々のユーザが利用しているために，時分割で OS スイッチをする．これに対し，「江戸」は全 OS を一人のユーザが使用することを前提としており，ある時点で使用していない OS はむしろ止まっていた方がよい．実際の運用を見ても，OS 間ファイル転送の時以外は，どれか1つの OS を選択的に使用する傾向がある．事象駆動方式はこれに適していると言える．

（4） ウィンドウシステムによるユーザインタフェースを提供した．

汎用大型計算機では，端末ごとに OS が限定される場合が多い．これに対し「江戸」のユーザは，1つの画面で複数の OS を同時に扱わなくてはならない．この場合，マウス・クリックでウィンドウを切り換えるだけで，OS 間を渡り歩くことができるウィンドウシステムが有効である．また，進行する複数の OS の状況を同時に確認することができ，効率よく作業できることが判明した．

5.4 「江戸」の問題点と今後の課題

「江戸」の問題点として次のようなことが挙げられる．

（1） 特権命令，割込みをエミュレートによって行うため，オーバヘッドが大きい．

（2） 仮想フレームメモリが白黒だけを対象としている．

（3） 例外スタックフレームが MC 68000 仕様である．

（4） アセンブリ言語のみで記述されており，保守

性が悪い.

(5) 実記憶系のOSを複数個実行するため，メモリ不足となる.

上記(1)の問題は，「江戸」が本質的に抱える欠点である．(2)の問題は，仮想フレームメモリのインタフェースが簡潔であるという利点ともなっている．また(3)の問題は，MC 68020 用の OS を「江戸」上に移植する場合などに表面化するであろうが，現時点では MC 68000 のソフトウェアをそのまま継承できるという利点になっている．(4)については，現在言語Cによる再記述を進めている．アセンブリ言語で記述した理由というのは，言語Cコンパイラ CAT の生成するオブジェクトのコンテキストが，OS のカーネル部分を記述するには重いことと，その部分の記述を対象にしていないからである．また，上位 OS として高い効率を実現する必要があったからである．現在，CAT のコンテキストを「江戸」用に変換し，効率に余り関係を持たない部分から，言語Cによる再記述を行っている.

上記(5)の問題は，今回の実現では最も問題となった．実用的には，CP/M-68K と OS/*o*，あるいは OS/*o* 2つの同時実行が辛うじて可能な程度である．特に「江戸」初版では，ターゲットマシンが最大で 4 M-bytes のメモリしか実装できないため，状況が悪い．メモリが不足する原因は，OS/*o* が実記憶系であるというだけでなく，日本語入力をサポートするために大きな辞書を抱えていることによる．これまでの OS/*o* は，辞書は ROM に焼けばよいという考え方で設計されていたため，ユーザが自由に ROM を使えない市販のパーソナルコンピュータでは，辞書の一部をメモリに常駐させなければならなかった．思い切ったメモリの増設や，ユーザ用 ROM の提供が，これからのパーソナルコンピュータには必要であると思う.

6. おわりに

本研究によって次の成果が得られた.

(1) パーソナルユースに適合した多重 OS 環境を設計・実現した.

(2) 仮想マシン・インタフェースを設定することにより，OS，アプリケーションプログラムのマシン独立性が高められた.

(3) 多重 OS 機能の実現により，異なる OS への開発環境の移行が容易となった.

(4) 旧 OS から新 OS をデバッグすることが可能となった.

(5) 市販のパーソナルコンピュータ上に OS/*o* の実行環境を実現することにより，OS/*o* を研究室外へ公開することを容易とした.

今後の課題としては，仮想マシン・インタフェースの拡張，マルチプロセッサシステムでの実現などがある．「江戸」の仮想マシン・インタフェースは，現時点でパーソナルコンピュータのハードウェアを完全に仮想化しているわけではなく．また，そのようなことを目標とはしていない．しかし，実際にユーザに使用してもらうと，ここが足りないというような不満が出て来る．インタフェースの拡張は，こういうニーズが出て来て初めて行うものだと考える.

謝辞 本研究の実現において御協力頂いた，(株)日立製作所神奈川工場 篠崎雅継主任技師ほかの方々に深謝する.

参考文献

1) 高橋，並木，武山，中川：OS/*o* のアーキテクチャと第一版の実現，情報処理学会 OS 研究会資料，24-11 (1984).
2) 鈴木，中川，高橋：OS/*o* 第二版におけるタスク管理，第 36 回情報処理学会全国大会論文集，5D-2 (1988).
3) 田中，鈴木，中川，高橋：OS/*o* 第二版のファイルシステム，第 36 回情報処理学会全国大会論文集，5D-3 (1988).
4) Buzen, J. P. and Gagliadi, U. O.: The Evolution of Virtual Machine Architecture, *National Computer Conference*, Vol. 42, pp. 291-299 (1973).
5) 八木橋，真鍋：流通ソフトウェアを考慮したマルチ OS，情報処理学会 OS 研究会資料，39-1 (1988).
6) Loucks, L. K. and Sauer, C. H.: Advanced Interactive Executive (AIX) Operating System Overview, *IBM Syst. J.*, Vol. 26, No. 4, pp. 326-345 (1987).
7) 鈴木：Syte 3000 の仮想マシン GEM，情報処理学会 OS 研究会資料，24-7 (1984).
8) 鈴木，小林，田中，中川，高橋：OS/omicron における日本語プログラミング環境，情報処理学会論文誌，Vol. 30, No. 1, pp. 2-11 (1989).
9) M 68000 マイクロプロセッサユーザーズ・マニュアル，CQ 出版社，Motorola Inc. (1984).
10) 並木，屋代，田中，篠田，藤森，中川，高橋：OS/omicron 用システム記述用 C 処理系 Cat のソフトウェア工学的見地からの方式設計，電子情報通信学会論文誌 (D)，Vol. J71-D, No. 4, pp. 652-660 (1988).

11) 田中，中川，高橋：OS/o 用言語 C コンパイラ cat の日本語化の方式とその実現，第 34 回情報処理学会全国大会論文集，3 V-5 (1987).
12) 里山，中川，高橋：日本語文書出力システム「浄書」の基本設計と開発システムの実現，情報処理学会ヒューマンフレンドリなシステムシンポジウム報告集，pp. 181-193 (1986).
13) 里山，中川，高橋：OS/omicron における文書出力システム浄書 (JOSHO)，第 33 回情報処理学会全国大会論文集，4 V-12 (1986).
14) MC 68020 ユーザーズ・マニュアル，CQ 出版社，Motorola Inc. (1986).
15) 岡野，堀，中川，高橋：多重 OS「江戸」とそのウィンドウ・システムの開発，情報処理学会 OS 研究会資料，39-4 (1988).

(昭和 63 年 9 月 5 日受付)
(平成 元 年 6 月 13 日採録)

岡野　裕之（学生会員）

昭和 63 年東京農工大学工学部数理情報卒業．同年 4 月，同大大学院修士課程入学，現在に至る．オペレーティングシステムなどのシステムソフトウェアの研究に従事．

堀　　素史（学生会員）

昭和 63 年東京農工大学工学部数理情報卒業．同年 4 月，同大大学院修士課程入学，現在に至る．プログラミング環境やユーザインタフェースなどの研究に従事．

中川　正樹（正会員）

昭和 52 年東京大学理学部物理卒業．昭和54年同大学院修士課程修了．同大学院在学中英国 Essex 大学留学 (M. Sc. in Computer Studies)．昭和 54 年東京農工大学工学部数理情報助手．平成 1 年同助教授．同年 4 月より電子情報助教授．オンライン手書き日本語入力，日本語計算機システム，文書処理の研究に従事．理学博士．

高橋　延匡（正会員）

昭和 32 年早稲田大学第一理工学部数学卒業．同年 4 月(株)日立製作所中央研究所入社，HITAC 5020 モニタ，TSS の開発などに従事．昭和 52 年東京農工大学工学部数理情報教授．平成 1 年同大工学部電子情報教授．オペレーティング・システム (OS/omicron)，日本語情報処理 (JOLIS, JOSHO) などの研究，教育に従事．理学博士．ACM，電子情報通信学会，ソフトウェア科学会，計量国語学会各会員．