5G－6

OS/omicronにおける新しいデバイスの追加に対するOSの機能拡張の一方式

早川栄一，中原雅彦，岡野裕之，並木美太郎，高橋延匡
（東京農工大学　工学部　情報工学講座）

1.はじめに

近年のワークステーションは，ハードウェアとしては，非常にコストパフォーマンスが高いものが登場している．しかし，ソフトウェア，特にシステムソフトウェアはメーカから与えられたものしか動作しないものが多い．我々は，並列処理や日本語情報処理，新しい浮動小数点演算の研究といったニーズに対して，既存のシステムは有効ではないと考えている．そのために，当研究室ではオペレーティング・システムOS/omicronの研究，開発を行っている．現在のシステムに対して，新しいデバイス，例えば浮動小数点プロセッサ（FPU）やDSP（デジタル・シグナル・プロセッサ）を自由に接続し，システムでサポートしたいという要求がある．ところが，タスクごとに与えられた資源として提供する場合にはシステムの大幅な変更が必要である．本稿では，これを改善するために，デバイスごとに必要に応じてコンテキスト退避領域を生成し，コンテキストスイッチを行う方式を提案し，OS/omicron第2版上での実現について述べる．

2.デバイスとコンテキストスイッチ

システムに周辺デバイスを増設するときに，そのデバイスをタスクごとに割り当てられた資源として活用したいことがある．これらは，CPUにおけるレジスタのように使用頻度が高くはない．しかし，コンテキストとして提供するためにはTCB（タスク・コントロール・ブロック）と呼ばれるOS内の領域に，退避領域を確保しなければならない．そのため，デバイスの増加はTCBのサイズの増大を招く．また，周辺デバイスの速度はプロセッサに比べて遅い場合が多く，コンテキストセーブ／リストアに多くの時間がかかる．OS/omicron第2版は割り込み処理やOSの処理をタスクとして実現しているために，割り込みやSVC（スーパーバイザコール）が発生するごとにコンテキストスイッチがおこる．このときの，コンテキストスイッチにかかる時間の増加は，OSの処理のオーバーヘッドを招く．

一例をあげれば，当研究室ではURR浮動小数点プロセッサを使用した浮動小数点方式の研究を行っている[1]．このプロセッサは，ペリフェラルプロセッサとしてシステムに接続されている．このプロセッサをマルチタスク環境で提供するためには，URRプロセッサのコンテキストを生成しなければならない．URRプロセッサのコンテキストは最低で46バイト必要であり，TCBのサイズは約20%増加する．また，URRプロセッサはメモリマップトI/Oの形で接続されているために，CPUレジスタに比べて，レジスタの転送には時間がかかる．

3.拡張コンテキスト

上記の問題に対して，我々は次の2点を実現する必要がある．

・TCB自体の増大を避けること．
・デバイスの無駄なコンテキストセーブ／リストアを減らすこと．

このため，我々は拡張コンテキスト方式を考案した．拡張コンテキスト方式は，次の特徴を持つ．

（1） TCB領域から，デバイスに依存したコンテキストと，依存しないコンテキストを分離したこと．(図1)
（2） コンテキストセーブ／リストアが必要な区間を指定できるようにしたこと．
（3） 任意のデバイスに対して拡張コンテキストを実現できるようにしたこと．

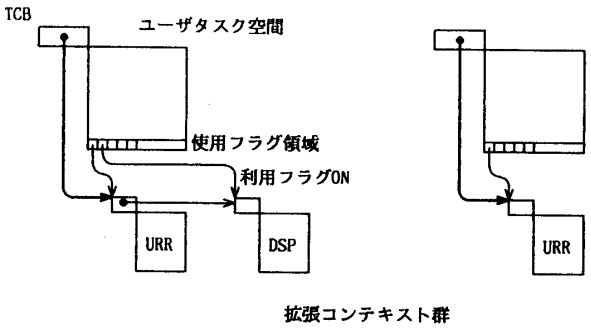

図1 拡張コンテキストとTCB

A Method of Operating System Enhancement for New Devices on OS/omicron
Eiichi HAYAKAWA, Masahiko NAKAHARA, Hiroyuki OKANO, Mitarou NAMIKI and Nobumasa TAKAHASHI
Department of Computer Science, Faculty of Technology,
Tokyo University of Agriculture and Technology

（1）によって，そのデバイスを使用しない通常のタスクは，デバイス依存のコンテキストをセーブする必要がなくなる．また，TCBの仕様を変更せずに，コンテキストセーブ／リストアが必要なデバイスを追加することが容易になる．コンテキストスイッチに時間がかかるデバイスであっても，（2）によってセーブ／リストアの回数を減らすことが可能になる．（3）は（1）から可能である．我々は，URRプロセッサに対してOS/omicron第2版の上でこの処理を実現した．

4．拡張コンテキストの実現

4．1　拡張コンテキストの構成

1つのデバイスに対するコンテキストの処理は次の（1）初期化，（2）拡張コンテキスト領域生成，（3）コンテキストセーブ，（4）コンテキストリストア，（5）拡張コンテキスト領域解放，（6）デバイス終了待ちの6つの手続きからなる．

拡張コンテキストは，コンテキストセーブ／リストアが必要になった時にのみ割り当てられる．ただし，一度割り当てられたコンテキストは，未使用のマークがつくが，そのタスクの終了まで解放されない．コンテキスト領域が不足した場合，未使用の領域の解放が行われて新しい領域が生成される．コンテキスト領域の生成，解放の操作はOSによって，自動的に行われるのでユーザが記述する必要はない．

4．2　拡張コンテキストの使用指定

拡張コンテキストを使用する場合は、各デバイスに対応した使用フラグというフラグをONにする．このフラグがONになっている間は，OSはそのデバイスのコンテキストのセーブ／リストア処理を行う．使用フラグの領域はユーザ領域にある．そのため，フラグのON－OFFではコンテキストスイッチは起こらない．

4．3　拡張コンテキストの動作

URRの場合を例にとって拡張コンテキストの動作を示す．（図2）

ユーザタスクでは，URRへのアクセスを行う前に使用宣言を行う．このとき，ユーザ領域にある，デバイスに対応した（この場合URRの）使用フラグをONにする．そして，URRのプロセッサを使用している間にコンテキストスイッチがおこった場合，OSへ制御が移る．この時点で使用フラグの領域はTCBへ退避される．OS内では，使用フラグがONになっていた場合，URRの演算が行われていれば，その終了を待つ．演算が終わった時点でコンテキスト領域を確保し，URRのコンテキストをセーブする．そして，次のタスクにコンテキストを切り替える．次のタスクがURRを利用していた場合は，URRのコンテキストをURRプロセッサにリストアし，コンテキスト領域に未使用のマークをつける．そして，次のタスクへのディスパッチを行う．

4．4　URRプロセッサのプログラミング

URRプロセッサを用いた浮動小数点のプログラミングでは，利用フラグのON－OFFの手続きは，当研究室で開発された言語CコンパイラCATのコードジェネレータによって自動的に生成，挿入される[1]．これによって，プログラマは拡張コンテキストを利用していることを意識する必要がない．

5．おわりに

システムに付加されるデバイスは多種多様であるが，マルチタスク環境を提供するシステムでは，タスクに割り当てられた資源としてデバイスを提供する場合がある．本方式は汎用的なデバイスを，マルチタスク環境へ取り込むための有効な手段であると考える．

参考文献

[1]　中原他：1つの実行形式で複数の浮動小数点表現を扱えるコードを生成する言語CコンパイラCAT／N，第40回情処全国大会，1990．3．

[2]　鈴木：OS/omicron第2版におけるタスク管理の実現，第38回情処全国大会，1989．3．

[3]　SUN Microsystems：*System Services Overview*，1989．3．

[4]　下原他：REALOS／F32：ITRON for GMICRO－（2）　技術的特徴　－，情処第39回全国大会，1989．9．

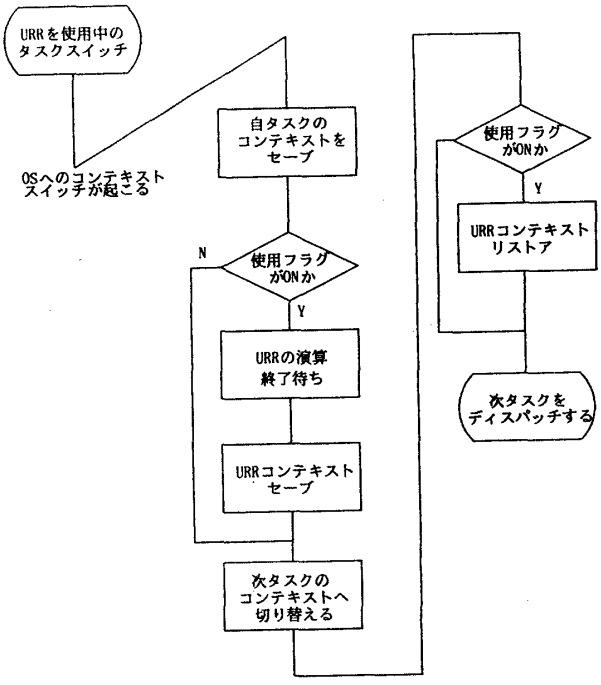

図2 URR使用時の拡張コンテキストの動作