Vol. 37 No. 9　　情　報　処　理　　Sep. 1996

解　説

組み込みシステム用ソフトウェア開発環境†

河井　淳††　西山由高††

1. はじめに

近年，音声/画像データなどの多様化したメディアを扱う，いわゆるマルチメディア関連の技術/機器の開発が盛んになっている．マルチメディア機器としてはゲーム機や携帯型情報端末などが身近なものとして連想されるが，このような機器の構成要素として，組み込み用RISCプロセッサ（マイコン）が重要な役割を果たしている．

従来，組み込み用マイコン上で動作するソフトウェアは，機能を小さく限定され，しかも，リアルタイム性を要求されることから，プロセッサ固有のアセンブリ言語で開発されていた．しかし，組み込み用マイコンの世界に押し寄せたRISC化，さらには，32ビット/64ビット化の波により，C言語やC++言語といった高級言語を用いた開発が可能となっている．また，組み込みソフトウェアの大規模化/複雑化により，組み込みソフトウェアを制御するためのリアルタイムOSをベースとした開発環境も登場するようになった．本稿では，最近の組み込み用RISCプロセッサの技術動向と，組み込み用ソフトウェアの開発を支援する環境について説明する．

2. 組み込み用RISCプロセッサの動向

PCやワークステーションのCPUとして使用されるRISCは，IBM 801の出現以来毎年急速に高性能化が進んできた．その勢いは現在においても衰えを知らない．

●性能重視の従来型Hot RISC

RISCの性能向上を達成するために，これまで多くの技術改良がなされてきた．CPUアーキテクチャの観点では，まずCPI (Clocks Per Instruction；平均命令実行クロック数) を限りなく1に近づける目的で，多段パイプライン構成，大容量レジスタファイル，ハーバードバス方式と命令/データ分離キャッシュメモリなどが導入された．これに加えて，命令のパイプライン実行により生じるインタロックを低減するために，フォワーディングパス，遅延分岐および分岐予測機構が設置された．次の段階では，CPIを1以下，すなわち1クロックサイクルに複数命令実行を可能にすることを目的として，スーパスカラ，あるいはVLIW (Very Long Instruction Word；超長形式機械命令)が導入された．これにともない，命令の並列実行を可能とするための複数語の命令/データアクセスパスの設置や，演算リソース利用効率を最適化するための命令再スケジュール機構によるOut of order issue（アウト・オブ・オーダ発行）や，レジスタリネーミング機構によるOut of order completion（アウト・オブ・オーダ完了）の設置，そして，分岐による命令フェッチのオーバヘッドを隠蔽するための分岐予測，および分岐ターゲットバッファ（分岐先命令アドレスキャッシュ）の設置などがなされている．これらの詳細については本稿のテーマにそれるので，詳細な解説は他の文献などを参照されたい(文献1)，2))．また，CPUの動作周波数の向上は，半導体デバイス技術，回路遅延や配線遅延の低減，クロックスキュー最小化のための回路技術，およびレイアウト技術によるところが大きい．このように，汎用RISC，すなわちPC，ワークステーションのCPUとして特定のアプリケーションに依存しないRISCの進化は，一重に性能向上の歴史であるといえよう．最近では，このような性能重視のRISCはHot RISCと呼ばれ，性能向上の一方で回路規模や消費電力が膨大になってきている[3]．

† Programming Environments for Embeded Systems by Atsushi KAWAI and Yoshitaka NISHIYAMA (Media Laboratories, OKI Electric Industry Co., Ltd.).

†† 沖電気工業(株)研究開発本部マルチメディア研究所MMコントローラPJ2G

表-1 組み込み用RISC基本仕様[4]

CPU名 (開発メーカ)	V 830 (NEC)	SH-3 (日立)	R 3900 (東芝)	mmX：開発コード (沖電気)	ARM 7 (シャープ)
命令長	16/32 bit	16 bit	32 bit	32 bit	32 bit
汎用レジスタ	32 bit×32 本	32 bit×16 本	32 bit×32 本	32 bit×32 本	32 bit×31 本
乗算/積和命令	あり	あり	あり	あり	あり
メモリオペランド	なし	あり	なし	なし	なし
性 能	118 MIPS*1 (100 MHz)	60 MIPS*1 (60 MHz)	52.5 MIPS*2 (50 MHz)	50.5 MIPS*1 (40 MHz)	30 MIPS*2 (33 MHz)
CPU規模	95 kトランジスタ	60 kトランジスタ	110 kトランジスタ	79 kトランジスタ	36 kトランジスタ

性能値順に記載．*1：Dhrystone 1.1 実行性能，*2：Dhrystone 2.1 実行性能．

●低消費電力/省チップ面積を考慮したトータルコスト重視のCold RISC

一方，特定のアプリケーションシステムのみを実行するCPUは，組み込み用CPUと呼ばれ，情報機器，家電製品，自動車，制御機器など多岐にわたる機器に内蔵され，これまでは，主として機器制御をソフトウェアで行う目的で用いられている．処理の規模/性能要求が比較的低いことから，圧倒的に4ビット～8ビットCPUが用いられていて，その多くはCISCである．ところが最近になり，ゲーム機器やPDA（Personal Digital Assistants；携帯情報端末）などのデータ処理をともなう装置，とりわけマルチメディア処理を対象とする装置への組み込みCPUの需要が急速に高まっている．それらは機器制御のみならず，音声処理，画像処理，あるいはグラフィクスなどの大規模なデータ処理，さらに，並行してネットワーク制御などのリアルタイム性の高い処理を実行するために，高性能なCPUを要求する．このような性能要求を満足するCPUとして，組み込み用RISCが最近盛んに開発されている．組み込み用RISCは，汎用RISCのHot RISCに対してCold RISCと呼ばれている[3]．Cold RISCの意味するところは，性能追求を第一義とせずに，対象システムを特定することにより，コストパフォーマンスの最適化を図ること，すなわち部品コストの削減，および消費電力を最小にして必要性能を満足させるトータルパフォーマンスの追求にある．組み込み用RISC，言い替えればCold RISCの特徴としては，単位消費電力あたりの処理性能（MIPS/W；1ワットあたりのMIPS値で表わされる）が高いこと，そして，LSI内でのCPUの占有面積が小さいことにある．

●アーキテクチャ技術動向

組み込み用RISC特有の特徴として，汎用RISCとは異なったアーキテクチャ採用の動きが見られる．**表-1**にいくつかの32ビット組み込み用RISCの基本仕様を示す．表に示されるように，汎用RISCでは常識となっている32ビット固定長命令，および32本のレジスタファイルではなく，V 830のように16ビット長命令と32ビット長命令とを混在させているものや，SH-3のように16ビット固定長命令のみとし，かつ，レジスタ本数を16に削減しているものもある．また，通信処理や，音声処理，およびイメージ処理などに多用される乗算や積和演算性能を向上させるために，機械語レベルでこれをサポートするものも多い．以下これらの特徴について考察する．

まず第1点は，16ビット命令と16本のレジスタファイルの採用である[5]．組み込み用RISCとして命令長とレジスタ数を半減させることで，CPU面積，およびプログラムのオブジェクトコードサイズを削減することを狙う．この場合懸念されるのは，16ビット命令を採用した場合，機械命令で指定できる機能の範囲が制限され，命令の強力さが一部損なわれることである．たとえば，32ビット命令で一般的な3オペランド命令ではなく2オペランド命令の採用や，即値，メモリディスプレースメント，あるいは分岐先PCオフセット値の削減による命令ステップ数の増加があげられる．また，レジスタ数についても，32ビットRISCで一般的な32本から16本となることで，CPU内部に保持できるデータ量が半減し，このことがメモリアクセス頻度の増加をもたらす

要因となる．つまり，命令長とレジスタ数を削減することは，汎用RISCの持つ大柄で平坦なアーキテクチャに対して，機能，および性能面で制限を与えるといえよう．組み込み用RISCの場合，このようなコスト削減にともなう性能低下が改善になるのか否かは，対象とするアプリケーションレンジの要求性能に大きく依存するため，一概に評価を下すことができないのは当然である．また，性能のみを評価することよりも，性能/消費電力，あるいは性能/CPU面積のようにコストパフォーマンスの視点での評価がより重要な意味を持つと考えるべきである．命令長（32ビットvs 16ビット）とレジスタ数（32本vs 16本）の評価については，すでに公開されている例があるが[6]，筆者らも独自に開発した32ビット組み込み用RISC（開発コード：mmX：multimedia eXecutor）を他のRISCと比較した[7]．この中で，対象アプリケーションをマルチメディア処理に想定し，いくつかのサンプルプログラムについてオブジェクトコードサイズの評価を行っている．

もう1点は，専用積和演算ユニットによる積和命令（まるめ，飽和演算機能を含む），および，メモリオペランド命令の採用である．これらは，CPUの機能，性能の向上につながるものである．積和命令は，マルチメディア処理で特に高い性能を要求する信号処理能力を強化するために導入されている．また，メモリオペランド命令は，一過性のデータを内部レジスタ経由することなく演算することで，内部レジスタの使用効率を向上させ，変数，あるいはポインタなどのレジスタ内でのライフサイクルを延ばすことを狙うものである．

この他，組み込み用RISCとして，様々な特徴を持ったものが開発されている．オブジェクトコードサイズを小さくするために，新たに16ビット命令を定義し，これをプロセッサ内部で32ビット命令に拡張して実行することで，オブジェクトコードの互換性を維持するものや（ARM社Thumb），CISC命令をRISC風に単純化し，シンプルなプロセッサアーキテクチャでこれを実行するものがある（Motorola社ColdFire）．また，メモリアクセス能力を向上させるために，4ワード（16バイト）を1命令で転送するためのブロックアクセス命令を備えるものがある（V 830）．さらに，リアルタイム処理性能を向上させるために，プロセッサ内部に複数のレジスタファイルを備え，割り込み受付け時などにこれらを切り替えることで，コンテキストの退避，および復帰時間を削減するものがある（ARM，SH-3）．その他，低消費電力化のための，クロック停止機構，クロック部分供給機構，バス駆動停止機構などが用意され，ソフトウェアにより動作モードを切り替えることにより平均消費電力を低減する機能や，消費電力を抑えた回路構成をとるものも多い．また，組み込み用RISCをコアプロセッサとし，固有ロジックを含むASICを開発するためのメガセル・ライブラリ化や，トップダウン設計によるASIC開発環境の構築が盛んに進められている[8]．

以上組み込み用RISCに採用されているアーキテクチャ上の特徴について触れたが，その採用はまだまだ機種によりまちまちである．このことは，設計思想に統一は見られず過渡的な状況にあるとも解釈できる．しかし，むしろ今後は個々の対象アプリケーションレンジに合った特徴付けが進むことから，組み込み用RISCのアーキテクチャは多様化する方向にあるともいえよう．また，組み込みシステムといえども32ビットRISCを想定するアプリケーションソフトウェアは相当程度の規模と複雑さを必要とすることから，システムの大部分はC言語，あるいはC++などの高級言語で記述されるであろう．このためコンパイラの能力がシステムパフォーマンスを決定する大きなファクタとなる．アーキテクチャ上のスペックダウンを補い，かつ性能とオブジェクトコードサイズが最適となるコード生成を行えるコンパイラ，言い替えればアーキテクチャを熟知したコンパイラの開発はCPUそのものの開発と同等に重要となる．また，組み込み用RISCの対象アプリケーションはますます多様化することを鑑みれば，対象アプリケーションレンジを絞ったアーキテクチャの合成，あるいはカスタマイズ，そしてそのアーキテクチャを最大限に活かすコンパイラを自動生成するような，いわゆるハードウェア/ソフトウェア・コデザインシステムの発展も待たれる[9]．

3. 組み込み用ソフトウェア開発環境

本章では，組み込み用ソフトウェアの開発言語とデバッグ環境の現状について解説する．また，ソフトウェア開発環境の新しい流れの1つとして，mmXのソフトウェアシミュレータを用いたプログラムの評価，デバッグ環境についても簡単に紹介する．

3.1 ソフトウェア開発用クロス環境

組み込み用のソフトウェア開発の流れの概略を図-1に示す．設計やプログラミングはホストマシン(PCやワークステーション)上で，テストやデバッグはプロトタイプ機で，最終的な運用テストや評価は実機で行うのが一般的である．

組み込み機器の最終製品（実機）は，低価格化を図るために最小のハードウェア構成になっている．また，メモリ量の削減や性能の向上のため，プログラムには必要最低限の機能しか実装されておらず，PCやワークステーションでは一般的なプログラム開発支援用ツールは含まれていない．そのため，組み込みプログラムの設計，コーディング，コンパイル，リンクといった段階は，各種ツールが整備されているホストマシン上で行う．しかし，実機のプロセッサやOSは一般にホストマシンと異なるため，組み込みプログラムのテストやデバッグには，実機やデバッグ用のプロトタイプ機を用いる．組み込みプロセッサ用のコードを生成するツールは，ホストマシン用のコードを生成するツールと区別して，クロスツール（クロスコンパイラ，クロスアセンブラ…）と呼ばれる．

テストやデバッグに用いるプロトタイプ機では，実機に搭載する量産用のプロセッサ（実チップ）を，デバッグ用回路を内蔵した評価チップやICE（in-circuit emulator）と差し替えたり，ROMにリモートデバッガを搭載している．ICEやリモートデバッガを用いた場合の接続例を図-2に示す．

ICEは，実行された命令のアドレスや割込み信号などのトレース情報を，プロセッサと同じ速度でメモリに保存していくリアルタイムトレース機能や，アドレスや各種信号に対する停止条件を指定するハードウェアブレーク機能を備えており，実機とほぼ同じ外部環境下でのデバッグが可能である．また，最近の組み込み用RISCは動作周波数が高くなり，ICEを作成することが難しくなっているため，実チップにあらかじめデバッグ回路を含めたり，ICE機能に必要な回路を組み込んだ評価チップを実チップと同時に提供する傾向にある．

3.2 コンパイラ

4ビットや8ビットのマイコンを組み込んだ機器では，プログラムの規模自体が小さいこと，内蔵メモリ容量やコストの制約からプログラムサイズを小さくする必要があること，プロセッサの性能を最大限に引き出す必要があることなどの理由により，アセンブリ言語を用いたプログラム開発が主流であった．しかし，機器が高機能化するに

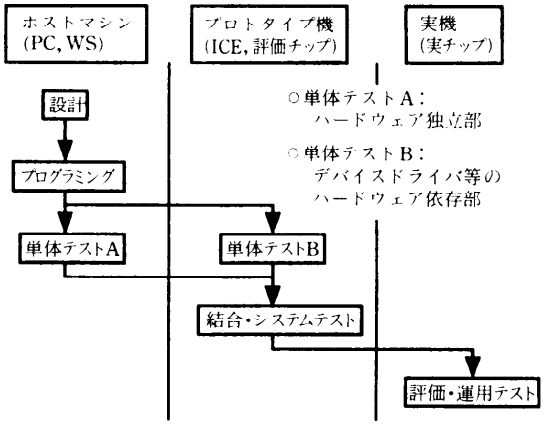

図-1 組み込み用ソフトウェア開発の流れ

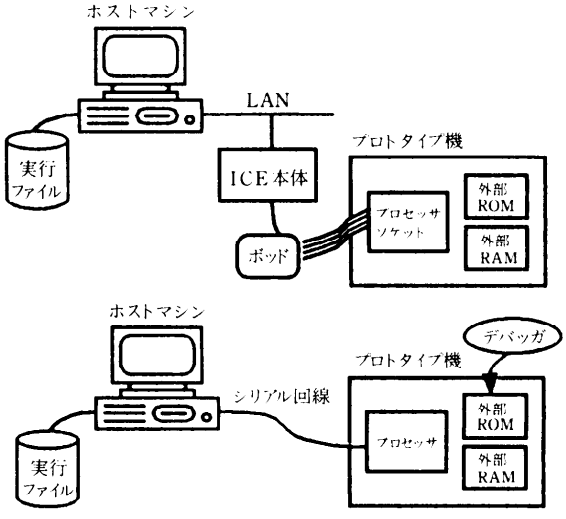

図-2 ICEやリモートデバッガを用いた接続例

つれてプログラムの規模や複雑さも増大し，開発や保守の効率向上のためにC言語やC++言語のような高級言語も併用されるようになっている．16ビット化や32ビット化によるプロセッサ性能の向上やメモリコストの低下が，高級言語(コンパイラ)を用いた場合のコード量増加や性能低下といった欠点を補ってくれるようになったことや，コンパイラの最適化技術が進歩して，アセンブリ言語を用いた場合に比べて遜色ない程度のコードが生成できるようになったことも一因である．

特に，組み込み用RISCでは，プロセッサ性能を引き出すために，16本あるいは32本あるレジスタの大域割り付け，遅延分岐，命令スケジューリングなどの最適化を施し，メモリへのアクセス回数やパイプラインストールを減らす必要がある．アセンブリ言語を用いる場合，これらのRISC型プロセッサ特有の最適化を人手で施す必要があり，従来のCISC型プロセッサに比べてプログラムの可読性が低下するとともに，プログラムを修正する場合の波及範囲が広くなり，変更に時間を要する．そのため，組み込み用RISCでは，アセンブリ言語で記述する部分をOSの一部や高速処理が必要な最低限の範囲にとどめ，それ以外は高級言語で記述する傾向がより顕著である．

一方，組み込み用RISCでは，一般に内蔵の命令キャッシュやROMの容量が小さいことやCISC型に比べてコード効率が悪くなることを考慮して，最適化コンパイラといえども，性能最優先のコードだけではなく，コード効率（サイズ）優先の最適化を行う機能も設けられている．サイズ優先のコード生成の例を**表-2**に示す．

表-2 サイズ優先のコード生成例

① 関数の入口/出口でスタックフレームの割り付け/開放やレジスタの退避/復帰を行うためのプロローグ/エピローグコードをサブルーチン化する．
② 8ビット，16ビット，32ビットなどのサイズの変数(データ)をそのサイズごとに集めて，連続する領域に割り付けることにより，境界整合のためのパディングの量を減らす．
③ C言語のswitch文などをコンパイルする際に，比較＋分岐の命令列を生成するか，ジャンプテーブルを用いるかの決定をコードサイズ優先で行う．また，ジャンプテーブルのエントリとして，ジャンプ命令や32ビットアドレスではなく，16ビットあるいは8ビットのオフセットを用いる．
④ 分岐命令の遅延スロットを埋める際に，コードの複写が発生しないように，遅延スロットに入れる命令を選択する．
⑤ インライン展開，ループの終了判定用コードの複写，ループ展開などをコードサイズを考慮して抑制したり，禁止する．

高級言語を用いてプログラムを記述した場合，組み込み機器固有の周辺デバイスの制御や正確なタイミング調整を要する以外の機能(モジュール)は，デバッグ環境が充実しているホストマシン上で単体テストできるため，プログラムテストの多くの部分を，実機やプロトタイプ機のハードウェア開発と並行して行える．また，組み込み用OSの中には，同一のシステムコールをホストマシンのOSでエミュレーションするためのライブラリを提供しているものもあり，これらのOSを用いたプログラム開発では，タスク間の排他制御や通信，スケジューリングなどのテストやデバッグまでもホストマシンで行える．

3.3 デバッグシミュレーション環境

デバイスドライバを含めたシステムテスト，外部割り込みやクリティカルパスの時間計測やタイミング調整などには，ICEと接続した実機やプロトタイプ機を使用する．機能やピンの配置が決まっている(1チップ)マイコンを用いる場合，実チップと同時にICEや評価用チップが提供されていることもあり，このようなテストや調整を迅速に行える．しかし，実チップのパッケージや実装方法によってはICEのプローブの形状が合わなかったり，ICEの動作が実チップと異なるといった問題点もある．また，最近では，高性能の組み込み用RISCを採用して，専用LSIで行っていた処理をソフトウェア処理に置き換え，LSIの部品点数を減らしてコスト削減を図ろうとする流れもある．特に，CPUコアとユーザ回路を1チップ化するASICでは，チップの面積，消費電力，パッケージなどを考慮してトータルコストが最小になるような設計を行う．そのためには，動作周波数，内蔵メモリの容量，外部メモリの種類，バス幅などのパラメータの組合せに対して事前評価を行い，CPUコアによるソフトウェア処理とユーザ回路によるハードウェア処理を適切に切り分ける必要がある．しかし，用途ごとに構成が異なるASICでは，チップの機能やピン配置があらかじめ決まっていないため，ICE開発の期間やコストが大きくなる．そのため，シミュレーション環境[10]のニーズも高まっている．

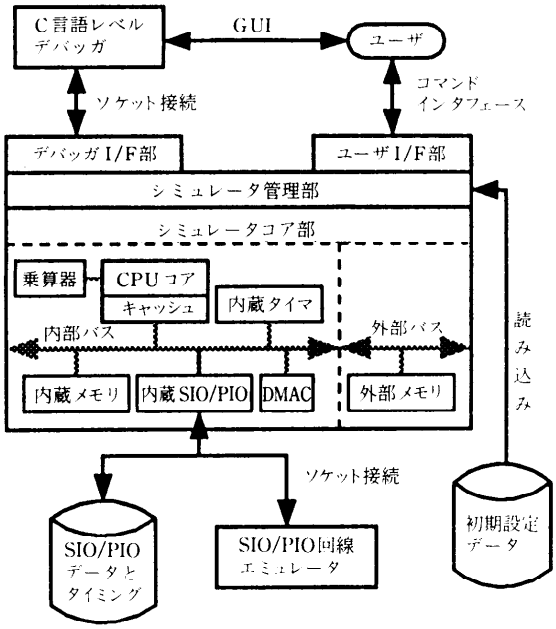

図-3 mmXシミュレータの構成（概略）

たとえば，筆者らは，mmXを用いたASIC開発支援ツールの1つとして，図-3に示すようなソフトウェアシミュレータを開発した．このシミュレータは，ASICの動作周波数，内蔵キャッシュやメモリの容量，乗算器の有無，外部バス幅，外部メモリの種類などに関するパラメータを初期設定ファイルから読み込み，ユーザからの入力コマンドにしたがってクロック単位のシミュレーションを行う．また，CPUコアと同時に提供されるASIC内蔵用の標準周辺モジュール（タイマ，シリアルIO，パラレルIO，DMACなど）の有無や動作を含めた正確なシミュレーションが可能である．

このシミュレータの利用により，従来のICEだけでは不十分であった，各種組合せに対する事前評価が可能になり，より効率的なASIC構成を決定できる．また，このシミュレータは，クロック単位の実行トレース機能やアセンブリ言語レベルのデバッグ機能を持っており，さらに，C言語レベルデバッガと連動することにより，マルチウィンドウ（GUI）ベースのデバッグも可能である．そのため，プロトタイプ機や実機の開発と並行して(あるいは先行して)，プログラムのテストやデバッグを行える期間（範囲）がより広くなり，全体的な開発期間の短縮につながる．

4. 組み込み用リアルタイムOS

本章では，組み込み用リアルタイムOSに要求される機能や性能，音声や動画像のような連続メディア向けの機能について述べる．

4.1 組み込み用OSとして要求される機能/性能

組み込み機器が高機能化するにつれて，プロセッサに接続される周辺デバイスやプログラムで実現すべき機能（モジュール）の数が増え，各機能も複雑化している．そのため，割り込み処理と単一タスクのループ処理による各機能の逐次的な実行だけでは，複雑なプログラム開発には対応しきれなくなり，リアルタイムOSを用いたマルチタスクプログラムの開発が一般化している．OSの採用により，プログラムを機能に応じたタスクに分割できるため，より構造化したプログラム開発や新規機能の追加が容易になる．また，割り込み処理やタスク間の排他制御や通信の機能はOSが提供してくれるため，開発工数を削減できる．また，個々のプロセッサに依存しない共通的な機能は，高級言語とOSを併用してタスク化することにより，プログラムの再利用も容易になる．

一方，組み込み用リアルタイムOSには，プロセッサ性能やメモリサイズの制約から，小規模でかつ高速であることが要求される．特に，割込み禁止時間やタスクのディスパッチ時間は，リアルタイムOSを用いる場合のオーバヘッドになるため，これらの最悪時間が極力小さくなるような仕様あるいは実装方法が採用されている．また，リアルタイム処理のための正確なタイミング設計を行うには，OSのオーバヘッドが小さいだけでなく，タスク数や資源数に依存しない（一定時間内に納まる）ことも重要になる．

商用の組み込み用リアルタイムOSは，ファイルシステムやネットワーク機能を除くカーネルのサイズが数K～10数Kバイトと小さく，ROM化が可能になっている．また，実機に必要な機能(システムコール)に応じて，メモリ使用量を抑制できるようにモジュール化されている．OSのオーバヘッドも，20 MIPS程度のプロセッサでは，割込み禁止時間が5～10 μ秒程度，タスクのディスパッチ時間が10～20 μ秒程度のものがある．

リアルタイムOSを用いたマルチタスクプログ

ラミングでは，複雑な処理を記述しやすくなる半面，デバッグやタイミングの調整の際には，プログラム中の変数だけでなく，各タスクの内部状態を参照したり，タスクが発行したシステムコール，外部割込み，タスクの状態遷移などの履歴をトレースする必要がある．そのため，リアルタイムOS対応のソフトウェア開発環境では，ICEやシミュレータを用いて収集したトレース結果を，タスク遷移，割込み発生，システムコール発行と関連付けてタイムチャート表示するモニタリング機能を持つ．また，タスクごとにブレークポイントを設定したり，タスクの内部状態を参照するためのデバッガも用意されている．特に，モニタリング機能は，タスクのデットロック状態や優先度逆転状態の検出，割り込みに対するタスク応答の遅れ，タスクのスケジューリングの誤りといったバグの検出には必須の機能である．

4.2 動　　向

日本国内では，TRONプロジェクトの一貫として，組み込み向けに設計されたITRON仕様に準拠したリアルタイムOSが，各種のプロセッサ上で実装されている．その中でも，当初，8ビットマイコンへの適用を想定して設計されたμITRON仕様(1989年公開)に基づくOSは，その後，16ビットマイコンや32ビット組み込み用RISC上にも実装されるようになり，現在，製品化されているものだけでも30程度ある[11]．

μITRON仕様は，組み込み用OSに必要な高速性と機能の豊富さの両面を考慮した仕様になっている．また，弱い標準化の方針が採用されており，個々のプロセッサアーキテクチャに適した実装方法や，アプリケーションが必要な機能のみを実装するといった選択が可能になっている．最新のμITRON 3.0仕様（1993年公開）では，粗結合ネットワークによって接続された他ノード（プロセッサ）上のオブジェクト（タスクやセマフォなど）を直接操作するための仕様が追加された．また，現在，共有メモリ型のマルチプロセッサシステム向けのITRON-MP仕様の検討が行われている．

一方，従来のリアルタイムOSでは，タスク生成時に，タスクの開始時刻，デッドライン，最悪実行時間，周期などの時間制約に関するパラメータを明示的に記述しておらず，これらの時間制約が達成可能か否かの解析や，一時的な過負荷状態の発生によってデッドラインに間に合わなかったタスクの検出や回復，中断といった処理はプログラマにまかされている．しかし，Real-Time Mach[12]のような新しいリアルタイムOSでは，タスクごとに時間制約の設定が可能であり，デッドライン条件を守れなかったタスクの検出や回復処理タスクへのメッセージ送信もOSが行ってくれる．また，タスクのスケジューリングポリシー（ラウンドロビン，固定優先度，レートモノトニック…）や排他制御のためのプロトコル（固定優先度，優先度継承，カーネライズドモニタ…）も指定できる．これにより，アプリケーションプログラムは，その性質に適したOSの振舞いを選択できる．

5. おわりに

組み込み機器においても，画像や音声といったマルチメディア処理や機器の高機能化により，プログラム規模が大きくなり，開発の効率化からC言語やC++言語が用いられるようになってきた．また，32ビットRISC型プロセッサに代表されるように，処理能力の大幅な向上により，従来は専用LSIで行っていた処理を，ソフトウェアで処理することが可能になってきた．それにつれ，プログラムを機能ごとに複数のタスクに分割し，タスク間の同期や通信処理を実時間で行う必要から，リアルタイムOSも用いられるようになっている．

一方，機器の開発期間の短縮やASIC化の際のチップの構成を事前に決定する必要から，従来のICEと実機（またはプロトタイプ機）を用いた性能評価だけでなく，シミュレータを用いた事前評価も重要になってきた．また，シミュレータを用いることにより，ハードウェア開発とソフトウェア開発を並行して行える期間がより長くなり，全体的な開発期間が短縮できるようになる．

参 考 文 献

1) Hennessy, J. L. and Patterson, D. A.: Computer Architecture: A Quantitative Approarch, Morgan Kaufmann Publishers, Inc. (1990)；富田, 村上, 新實（訳），ヘネシー & パターソンコンピュータ・アーキテクチャ—設計・実現・評価の定量的アプローチ—，日経BP社 (1992).
2) Johnson, M.: SUPERSCALAR MICRO-

PROCESSOR DESIGN, Prentice-Hall Inc. (1991)；村上 監訳, スーパースカラ・プロセッサ—マイクロプロセッサ設計における定量的アプローチ—, 日経BP社 (1994).
3) Special Report: Embedded 32-bit RISC, COMPUTER DESIGN, pp. 77-96 (Jan. 1994).
4) 特集 RISC安価で高速なRISCプロセサが組み込み市場の主役に, 日経エレクトロニクス No. 636, 1995 5-22, pp. 75-90 (1995).
5) 日立シングルチップRISCマイコンSH 7032, SH 7034 ハードウエアマニュアル, 日立製作所 (1993).
6) Bunda, J. et al.: 16-Bit vs. 32-Bit Instructions for Pipelined Microprocessors, Symp. on COMPUTER ARCHITECTURE Proceedings, pp. 237-246, IEEE Computer Society (1993).
7) 河井, 他：マルチメディア用RISCコントローラとその応用, 情報処理学会, 計算機アーキテクチャ (110 21)・設計自動化 (73 21) 合同研究会資料 (1995).
8) 河井, 他：組み込み用32 bit RISCとASIC開発環境, 情報処理学会, 計算機アーキテクチャ研究会 (112 5) 資料 (1995).
9) 安浦：ハードウエア/ソフトウエア・コデザイン—ソフトウエアコアプロセッサによるシステム設計—, 電子情報通信学会技術研究報告, ICD 94-130, DSP 94-86 (1994).
10) ICE中心のソフト開発から脱皮, シミュレータ利用で検証期間が1/3に, 日経エレクトロニクス No. 573, 1993 2-1, pp 139-147 (1993).
11) 高田, 他：ITRONサブプロジェクトの現状と展望, 情報処理, Vol. 35, No. 10 (Oct. 1994).
12) Nakajima, T.: Current Status of Real Time Mach (In Japanese), JAIST Research Reports IS-RR 94 18 S, Japan Advanced Institute of Science and Technology (1994).

(平成7年11月14日受付)

河井 淳

1952年生. 1975年慶應義塾大学電気工学科卒業. 同年沖電気工業(株)入社. 入社以来コンピュータシステム, マイクロプロセッサの開発を担当. 現在同社研究開発本部マルチメディア研究所に勤務. IEEE CS会員.

西山 由高

1960年生. 1983年神戸大学工学部電子工学科卒業. 1985年同大学院修士課程修了. 同年沖電気工業(株)入社. 入社以来, 画像処理システム, コンパイラ, モニタの開発を担当. 現在同社研究開発本部マルチメディア研究所に勤務.