新しいプロセッサのためのクロス統合デバッガ

6N-5

前田 賢一　斎藤 光男
(株)東芝　総合研究所

木南 英志
東芝ソフトウェアエンジニアリング(株)

1．はじめに

近年，コンピュータ・アーキテクチャは，仮想記憶やキャッシュなどをアーキテクチャに取り込むことにより複雑化している．このため，新しいアーキテクチャをもつ計算機を開発するということは一般に困難な作業である．

従来，ハードウェアの開発のためにはＩＣＥが主として使用されてきた．これは，既にアーキテクチャが決定されたプロセッサの応用システムを開発するには有効な手段であるが，新しいアーキテクチャを開発する手段としては融通性がない

また，高速のＩＣＥを短期間に開発するのはプロセッサの開発以上に困難な作業で，プロセッサの高速化とともに不可能になりつつあるため，新しい方法を考える必要があった．

このため，筆者等はＡＩＰ（ＡＩプロセッサ）［1，2］開発のために，バックエンド・プロセッサという開発形態を採用し，制御デバッグ手段として本稿に述べる強力なクロス統合デバッガをホスト・プロセッサであるＡＳ３０００上に開発した．ハードウェアとしても，このようなデバッガの存在を設計時から想定し，必要なハードウェアの外部からのアクセスを可能としている．

このデバッガはハードウェア，マイクロコード，プログラムの全てを対象として，統一的にデバッグを行なうことができる．このため，ハードウェアの動作検証とともに，ＯＳの開発に対しても強力なツールとなっている．

また，ＯＳのソースコード・デバッグのためのクロスｄｂｘも本デバッガのハードウェア・アクセス機能を用いて実現されている．さらに，シミュレータとも組み合わされ，同じユーザ・インターフェースでハードウェアとシミュレータの両方を使うことができるようになっている．図１にデバッガ関連のソフトウェアの関係を示す．

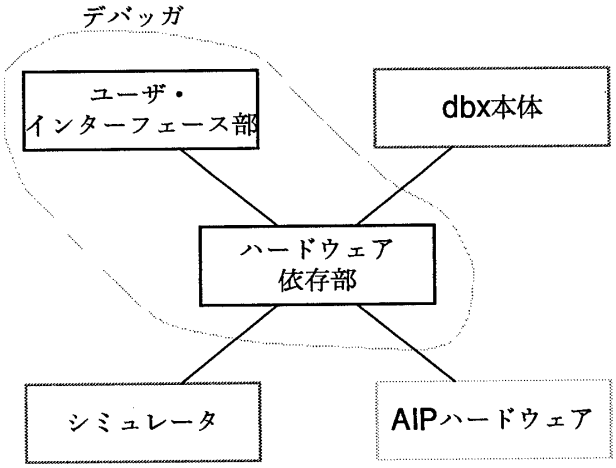

図1　デバッガ関連ソフトウェア

2．設計思想

本デバッガはＡＩＰをモニタリングするソフトウェア的な唯一の手段である．従って，必要なハードウェアには全てアクセスできなくてはならない．逆にハードウェアとしても，ソフトウェアからアクセスできる手段を用意しておく必要があり，ハードウェア設計時からこのようなデバッガを想定し，ハードウェアはＶＭＥバスを介して外部からアクセスできるようになっている．

ＶＭＥバスはホスト側ＯＳのｍｍａｐ関数によってアドレス空間にマッピングされ，通常のメモリと同様にアクセスすることができる．このようなアクセス形態を取ったことにより，ドライバーを作成する必要もなく，後述のようにＩ／Ｏのサポート等もスムーズに実現できた．

3．特徴のまとめ

（1）ユーザ・インターフェース

ユーザ・インターフェースは，その善し悪しでデバッグ効率に影響があるので，重要な問題である．

デバッグの際，同じ操作の繰返しや一部だけの

An Integrated Cross-Debugger for New Processor
Mitsuo SAITO*, Ken-ichi MAEDA*, Hideshi KIMINAMI**
*TOSHIBA Corporation, **TOSHIBA Software Engineering Co. Ltd.

変更操作が頻繁に発生する．これを効率良く実行するために，ｅｍａｃｓ風の行編集機能を持たせた．ｅｍａｃｓはエディタとして多くの開発者に愛用されていたため，これによって，作業効率が改善された．また，過去の入力のヒストリー機能もあり，長く複雑な入力の再利用も可能となっている．さらに，スクリーン・エディット機能により一部だけの修正が容易であり，入力ミスによる作業の中断も少ない．

（２）ターゲット・システム

ａ．ＣＰＵハードウェア・レベル

レジスタ，特殊レジスタ，パイプライン・レジスタ，ＷＣＳ，バスが表示／修正（できるもののみ）可能である．

ｂ．マイクロコード・レベル

マイクロコードはファイルからのロードとその場での入力が可能である．編集は前述のユーザ・インターフェースを用いてアセンブラ・ニモニックで表示／入力ができる．ファイルに格納することもできる．マイクロコード・レベルでのブレーク・ポイントやステップ実行も可能である．

ｃ．プログラム・レベル

プログラムの入力等はマイクロコードと同じサポートがされる．

Ｃで書かれたテスト・プログラム等を実行するためには，Ｉ／Ｏのサポート等が必要となる。本デバッガでは，ＡＩＰのメモリはｍｍａｐによりホストのメモリと同様に見えるため，ＡＩＰが停止した時ＡＩＰに代ってホストが扱う．ＡＩＰからの要求はＡＩＰのスタックに積まれ，そのアドレスをホストから参照することによってコミュニケートしている．この様子を図２に示す．

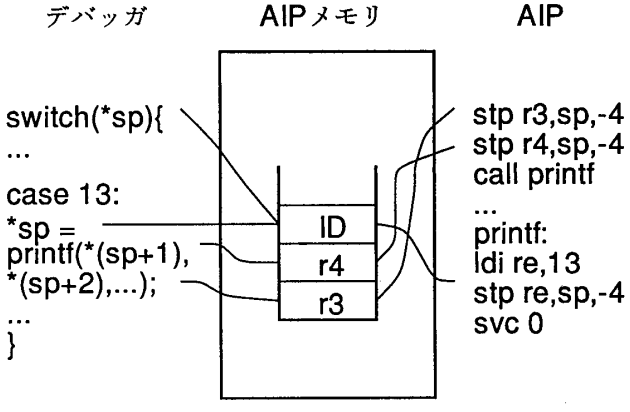

図2　I/Oのサポート

ブレークポイントの機能はハードウェアのデバッグにも極めて有効である．ブレークポイントの通過回数カウントも可能である。

ｄ．周辺ハードウェア・レベル

メモリ，MMU，キャッシュ，割込等の制御レジスタの表示／修正が可能である。

（３）シミュレータ機能

ハードウェアの動作は込み入っているため，直観的に理解するのが困難であることが多い．シミュレータはハードウェアとの動作比較のために不可欠のツールである．

このシミュレータはゲートやフリップフロップまでシミュレートするようなミクロなものではない．レジスタやバス等のレベルでシミュレートするものである．シミュレータを使っているということはスピードの点を除くとほとんど意識する必要がない．通常，シミュレータは単独で動くものであるが，デバッガと組み合わせることにより，一段と使い易い環境となった．

４．　おわりに

本稿では新しいアーキテクチャのプロセッサ開発手法として，また，ＯＳのカーネル・モードのデバッグ手段として，バス結合されたホストからのクロス統合デバッガを紹介した．

この種のツールは新しい開発に不可欠のものであり，場合によっては外注するなどの方法も取られるものであるが，今回は自分自身で作成する方法を取って成功したと考えている．すなわち，アーキテクチャの設計者でありＯＳのインプリメンタであった著者らが自ら作成したものであり，「自分達のツール」として使いやすいものであった．その証拠として，ボードの試作からＬＳＩの開発まで，マイナー・チェンジを経て，ハードウェア，ソフトウェア両方の開発者に数世代にわたって愛用されてきた．

今後はシンボリック・デバッガとの位置付けの検討（統合等）や，さらに新しいアーキテクチャの開発への活用を考えたい．

参考文献

［１］斎藤，他：ＡＩワークステーションの開発思想　人工知能学会第１回全国大会（１９８７）

［２］斎藤，他：ＡＩワークステーション（ＷＩＮＥ）の開発１－７　情報処理学会第３５回全国大会（１９８７）