仮想マシンを用いた組込み OS のデバッグ環境の開発

吉田幸二† 山本茂樹‡ 早川栄一‡
拓殖大学大学院 工学研究科電子情報工学専攻†
拓殖大学 工学部情報工学科‡

1. はじめに

近年、組込みシステムの分野において、ウェアラブルコンピュータやユビキタスシステムの研究、開発が行われている。これらのシステムで使われているオペレーティングシステム(以下、OS)も開発されており、OS の開発環境が重要になってくる。

OS のデバッグを行う場合、通常のアプリケーションのデバッグとは異なり、不正なメモリアクセスによるデータの破壊、割込み処理や特権状態時のデバッグ対象のトレースができないなどの問題がある。

そのため、このようなエラーで破壊されない、安全なデバッグ環境[1]が求められる。また、組込みシステムを対象としているため性能面を考慮した環境でなければならない。

本報告では、組込みシステムを対象とした OS のデバッグ環境の開発について述べる。

2. 設計方針

(1) 安全なデバッグ環境を提供する

不正なメモリアクセスによるデータの破壊から、デバッグ対象,デバッガなどを保護することで、安全なデバッグ環境を提供する。

(2) デバッグ時のオーバヘッドを軽減する

組込み OS を対象としているので、システムにある程度のリアルタイム性が求められる。
そのため、デバッグ時にかかるオーバヘッドをできるだけ抑える。

(3) 割込み処理のトレースをできるようにする

既存のデバッガではトレースが難しい割込み処理のトレースをできるようにする。そうすることで、デバッグが困難な割込み処理のデバッグを支援する。

Development of a Debugging Environment for Embedded Operating System using Virtual Machine
Koji Yoshida†,Sigeki Yamamoto and Eiichi Hayakawa‡
†Graduate school of Engineering,Takushoku University
‡Takushoku University

3. 設計

3.1. システムの全体構成

本システムは、実機上で動作する仮想マシン上にデバッグ対象の OS を走らせ､GDB を用いてリモートデバッグを行う。

仮想マシンを実機上に提供することで、メモリの保護や割込み処理時のトレースなどの機能を実現する。

3.2. 仮想マシンの設計

図 2 に仮想マシンの構成を示す。

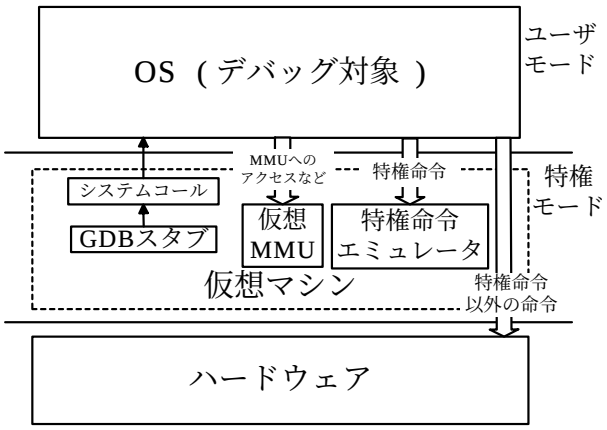

図 1：仮想マシンの構成

本システムで述べる仮想マシンは、従来のようなハードウェアのすべての機能をエミュレーションするものではなく、保護に必要な機能だけを仮想マシンで処理することで、実行時のオーバヘッドを軽減する。次に仮想マシンの機能を上げる。

(1) 特権命令

デバッグ対象はユーザモードで実行されるので、特権命令は実行できない。そこで、特権命令の処理は仮想マシンで行う。

(2) MMU へのアクセス

特権命令と同様にユーザモードからはアクセスできないため、これらのアクセスを仮想マシンで処理する。

(3) GDB スタブ

デバッガとして､GDB を使用するので GDB ス

タブを仮想マシン内に実装する。

(4) デバッグ対象の管理

デバッグ対象を仮想マシンのタスクとして扱う。そのため、OS のようなタスク管理,メモリ管理,割込み,システムコールなどの機能も仮想マシンの機能として実装する。

特権以外の命令や I/O などは仮想マシンでは処理せず、直接ハードウェアで実行することで、オーバヘッドを軽減する。

3.2.1. 特権命令エミュレータ

デバッグ対象は、ユーザモードで実行しているため、特権命令を実行しようとすると例外が発生する。特権命令エミュレータは、この例外ハンドラとして実装する。例外原因から、実行され様とした命令を判別し、その命令を実行する。

3.2.2. 仮想 MMU

仮想 MMU は、MMU の保護機能を使ってメモリの保護を行う。

デバッグ対象は、ユーザモードで実行しているので、ユーザモードでは許可されていないメモリ領域へのアクセスや MMU へのアクセスを行うと、例外が発生する。仮想 MMU は、これらの例外ハンドラとして実装する。例外が発生した場合、仮想 MMU は例外原因からデバッグ対象の要求を判別し、その処理を実行する。

3.2.3. システムコール

デバッグ対象は、仮想マシンのタスクとして実行させる。そのため、GDB スタブに対してシステムコールを提供する。仮想マシンが提供するシステムコールを表 1 に示す。

表 1：システムコール

システムコール	説明
createTask	タスクの生成
deleteTask	タスクの削除
allocateMemory	メモリの確保
freeMemory	メモリの解放
setExceptionHandler	例外ハンドラの登録
getDebugChar	シリアルポートから 1 文字取得する
putDebugChar	シリアルポートへ 1 文字出力する

3.3. GDB スタブ

デバッガには、GDB を使用するので、仮想マシン内に GDB スタブを実装する。これは、すでに提供されている GDB スタブをもとに、デバッグ対象をロードする部分を、仮想マシンのタスクとして登録できるように手を加え、シリアルポートを使用して GDB と通信できるインタフェースを実装する。

4. 実装

今回は、実機としてりぬくす工房から出ている CAT68701 評価ボードを使用しプロトタイプの実装を行った。

プロトタイプの実装に使用した開発環境を表 2 に示す。

表 2：開発環境

PC	CPU	AMD Athlon 1Ghz
	メモリ	256MB
CAT68701 評価ボード	CPU	Super-H7708R
	メモリ	32MB
		シリアルポート 10Base-T LAN ポート

5. 評価

特権命令エミュレータを実装し、エミュレーションにかかるオーバヘッドを測定した。サンプルとして、LDTLB と STC の二つの命令について測定を行った。測定結果を表 3 に示す。

表 3：測定結果

測定した命令	実機	エミュレータ
LDTLB 命令	11ns	1720ns
STC 命令	11ns	1760ns

測定結果からエミュレーションした場合は、実機で実行した場合の約 1/160 倍の速度で実行できることがわかった。実行される命令全体に対して特権命令が占める割合は少ないので、全体としてそれほど遅くはならない。また、本システムはハードリアルタイムを対象とは考えていないので、このくらいのオーバヘッドでも、問題にはならない。

6. おわりに

本報告では、仮想マシンを用いた組込み OS のデバッグ環境の開発について述べた。これにより、実機の 1/160 程度の速度だが、ユーザモードからでも特権命令を処理できるようになった。

今後は、他の機能とシステム全体の評価を行い、システムの改善を行う。

参考文献

[1]清水正明･早川栄一･並木美太郎･高橋延匡：OS デバッグ環境の設計と実現，情報処理学会 「オペレーティングシステム」研究会 057-001,1992 年