車載組込みシステムフォーラム（ASIF）
2020年度第2回スキルアップセミナー
2020/10/27

仮想化技術の基礎と
車載システム向け仮想化技術の動向

本田　晋也
南山大学 理工学部
TOPPERSプロジェクトシニアテクニカルエキスパート
shonda@nanzan-u.ac.jp

最終更新：2020/10/21

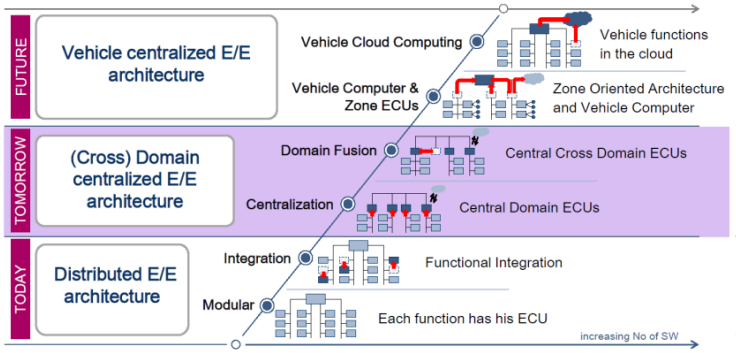
アジェンダ

仮想化技術の基礎について説明し，車載システム向けの仮想化のためのハードウェア及びソフトウェアについての現状について解説する.

- 仮想化技術
- 組込みシステムにおける仮想化
- ハードウェア仮想化支援機能
  - 車載システム向けプロセッサのハードウェア仮想化支援機能
- 仮想マシンモニタ(VMM)・ハイパーバイザー
  - インフォテイメント及びドメインコントローラ向け
  - 車載システム向け
  - RH850 HAV を用いたハイパーバイザ

車載制御システムにおけるECU統合

- 分散リアルタイムシステムの大規模・複雑化
  - 高度な機能，複数のコンピュータが連動した制御を実現するためにコンピュータ数が増加
    - 1機能 = 1ECU = 1サプライヤー
    - 搭載されるソフトウェアも大規模化
- 小型モビリティの実現
- ドメインコントローラ
  - 可能な限り各ECUの機能をまとめる
  - ガソリン，ハイブリッド，ディーゼル
- 自動車制御システムにおける課題
  - ECU設置スペースの不足
  - ネットワーク帯域の不足



Source: Bosch

ECUの数を減らすために，単一のECUで複数の車載制御アプリを動作させる**ECU統合**の検討が進んでいる

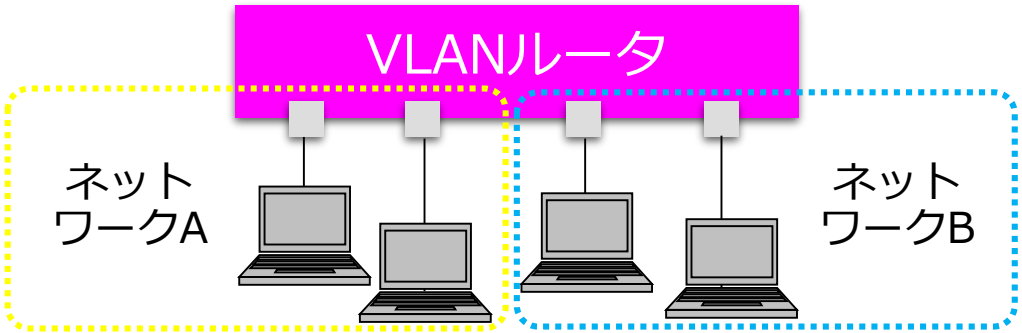
解決方法の一つとして**仮想化技術**の利用が挙げられる

仮想化技術

仮想化技術

- コンピュータシステムの仮想化
  - 一つのコンピュータ上で複数の仮想コンピュータやOSを動作させる
    - 命令セットが異なる場合も.
    - 実現方法は様々

- ネットワークの仮想化（Virtual LAN）
  - 1つのルータで機器複数のネットワークを混在させる



汎用コンピュータにおける仮想マシンの必要性

- ハードウェアとソフトウェアの分離
  - サーバースペースの有効利用
  - ハードウェア障害への耐性
  - 負荷の平準化等の柔軟性向上

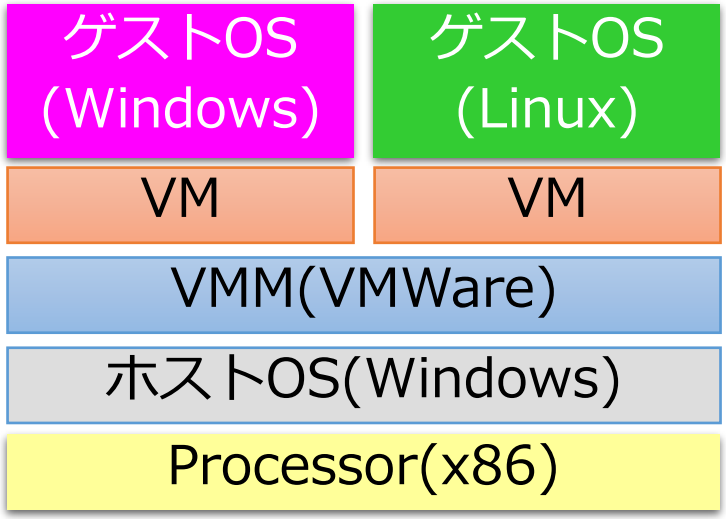
- 別のアーキテクチャ向けのプログラムを実行
  - Mac OS Rosetta2：X86_64のアプリをARM64で実行

- 別のOSやOS向けのプログラムを実行
  - Windows WSL/WSL2：LinuxやLinuxのアプリをWindowsで実行

- ユーザー・プログラム環境の分離
  - 必要なライブラリのバージョンが違う場合
  - セキュリティの観点からの分離

仮想マシン（VM）

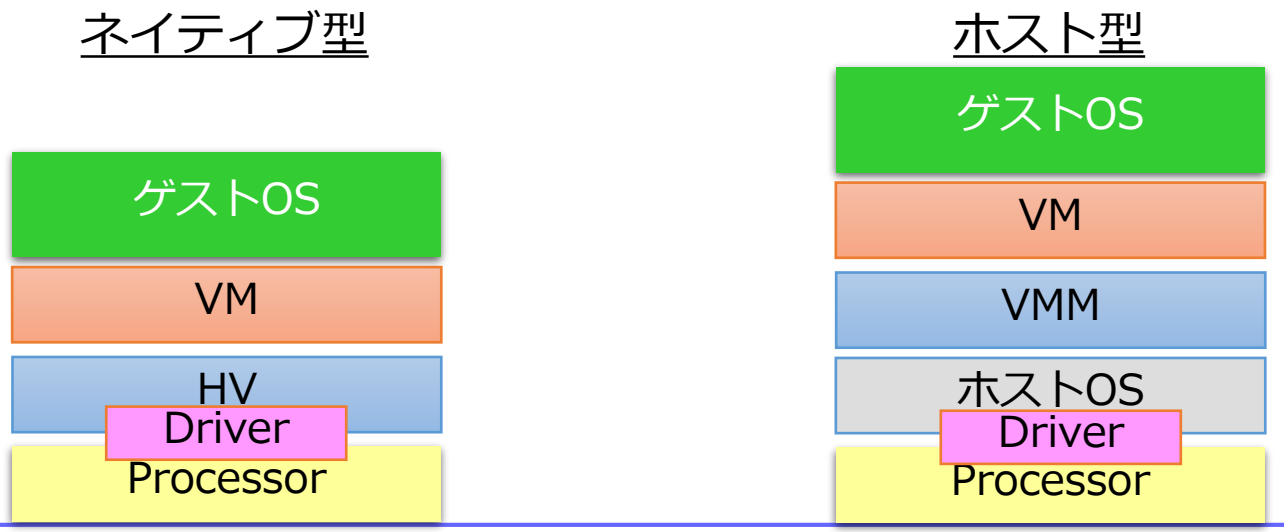
- 仮想マシン（VM）
  - ハードウェアを仮想化したもの．実態としてはVMMが管理する管理情報等
- ハイパーバイザー/仮想マシンモニタ(VMM)
  - 仮想マシンを制御するソフトウェア
  - 仮想マシンの起動・終了，切り替え，仮想デバイスの提供
  - 通常OSが動作するプロセッサのモードがスーパーバイザモード．
  - ハイパーバイザーはハイパーバイザモードで動作するためこの名が付いた．
- ホストOS：ハイパーバイザー・VMMを実行するOS
- ゲストOS：仮想マシン上で動作するOS



VMの実現方法による分類

- ベアメタル/ネイティブ型（Type 1）
  - VMWare ESX, ESXi, Xen, L4
  - ホストOSを使用しないため信頼性は高い
  - 使用可能なハードウェアに制限
  - ハイパーバイザー(HV)と呼ぶことが多い
- ホスト型（Type 2）
  - VMWare Workstation, VirtualBox, QEMU
  - 信頼性はホストOSに依存
  - 使用可能なハードウェアが多い（ホストOSがサポートしてれば動作）.
  - 仮想マシンモニタ(VMM)と呼ばれることが多い

VMWare ESXのサポートHW
ネットワークカード：2種類
SASコントローラ　：5種類
SATAコントローラ　：3種類



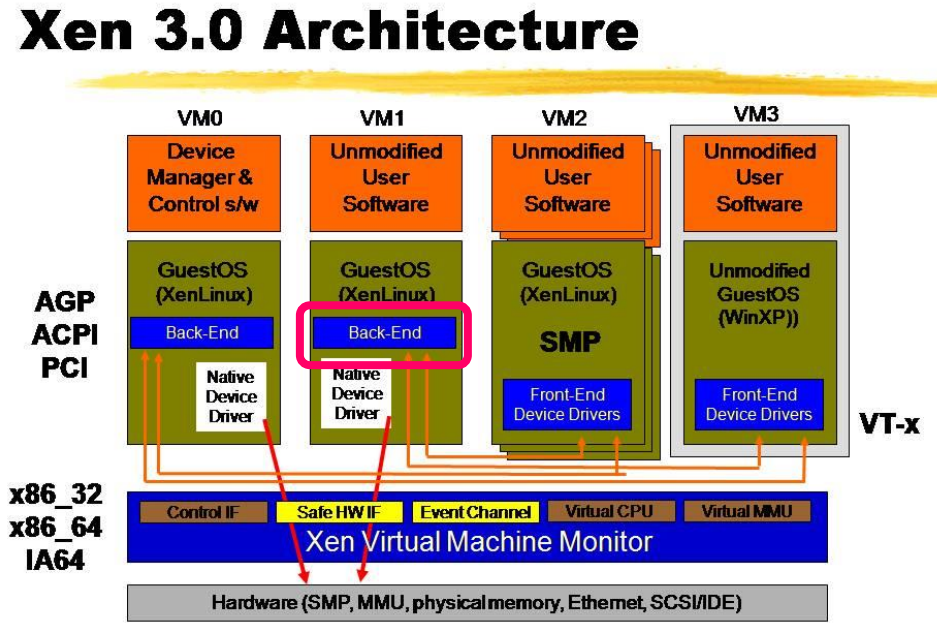
ゲストOSに関する分類

- 完全仮想化（full virtualization）
  - ゲストOSに手を入れる必要がない.
  - VMWare/VirtualBox/QEMU
  - ドライバのみ準仮想化することによりパフォーマンスや利便性を向上
    - VMWare tools/VirtualBox Guest Additions

- 準仮想化 （paravirtualisation）
  - ゲストOSを仮想化のために変更する
  - Xen

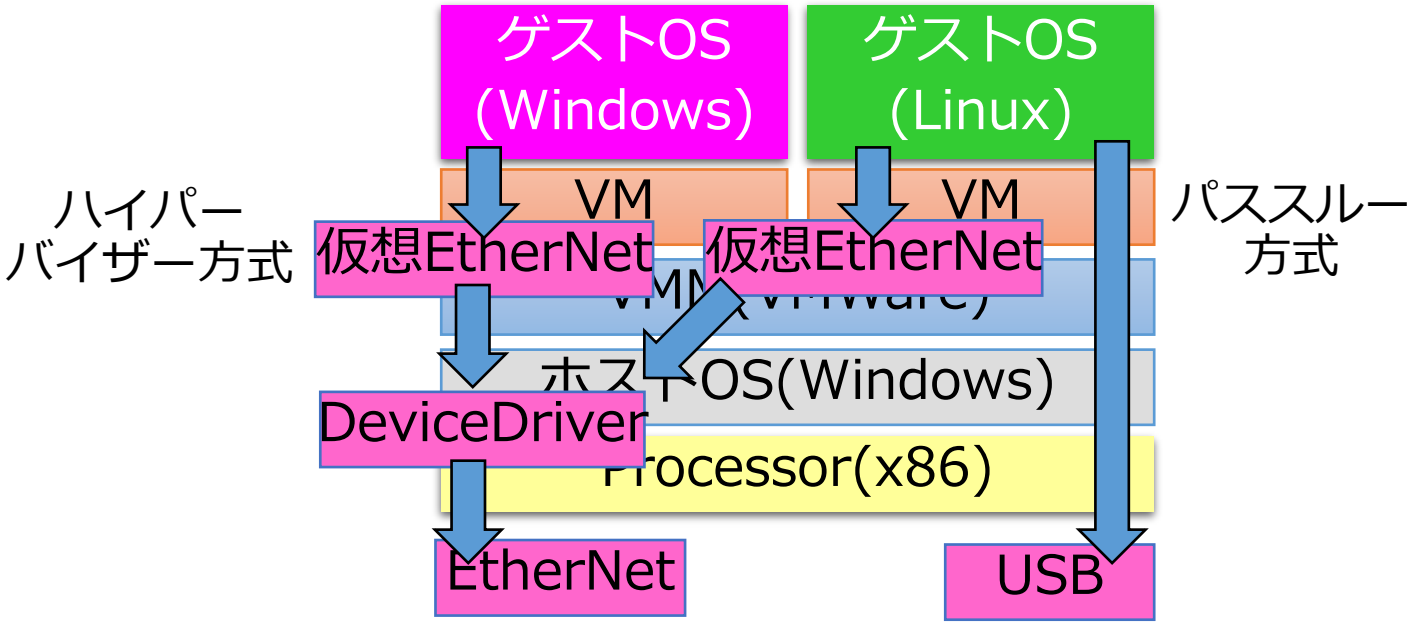
- OSレベル仮想化
  - Linux Containers(LXC)
  - Solaris Containers/FreeBSD jail

- アプリケーション仮想化
  - Java, .Net, 各種スクリプト言語



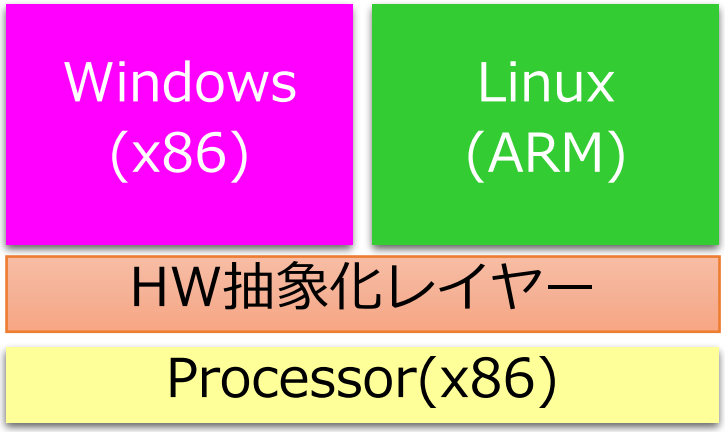
デバイスのアクセス方法

- ハイパーバイザー方式
  - 論理的なハードウエアを提供する方式
  - ゲストOS間でのデバイスの共有が可能
- パススルー方式
  - ゲストOSに直接ハードウェアを使用させる
  - ゲストOS間でのデバイスの共有は不可能



仮想マシンのデメリット

- 性能の低下
  - 複数のOSを実行するため，ハードウェアを抽象化するレイヤーが必要なため
  - ハードウェアによるサポートにより改善（後述）
- システムの複雑化
  - 複数のOS+HW抽象化レイヤー
  - デバッグが困難
- 方式によってはサポートするハードウェアが限定



組込みシステムにおける仮想化

仮想化の歴史（主に2000年以降，PC/組込み関連）

- 1960年台 初期のコンピュータから利用されている
- 1999年　VMWareの登場(PC向けの仮想化の登場)
- 2003年　Xenの登場
- 2007年　x86の仮想化支援機能Intel VTが搭載
  OKL4が東芝の携帯電話（W47T）に採用
  Linuxカーネルがサポートする仮想化機能KVM登場
- 2009年　Windows7に Windows XP Mode が搭載
- 2010年　組込み向けプロセッサARM(A)の仮想化支援機能が発表
- 2012年　パナソニックが仮想化技術で2個のOSを実行する
  スマートフォンを販売
- 2014年　組込み向けプロセッサARM(R)の車載システム向けの
  仮想化支援機能が発表される
- 2015年 組込み向けプロセッサARM(M)にIoT機器向けの仮想化機能が発表
  2018年頃から搭載したMCUが販売開始
- 2019年 ルネサスエレクトロニクスの車載システム向けプロセッサRH850に
  仮想化支援機能が搭載

組込みシステムにおける仮想マシンの必要性

汎用PCにおける要求に加えて以下の要求がある

- 異なる性質・性能の複数のプログラムでシステムを構成
  - 信頼性/リアルタイム性(RTOS)と非信頼/最新機能(GPOS)

- Secure/Non-Secureでシステムを分けたい
  - スマートフォンのアプリはオープンな環境上で実行
  - FeliCaやDRMのキーはクローズドな環境上で管理実行

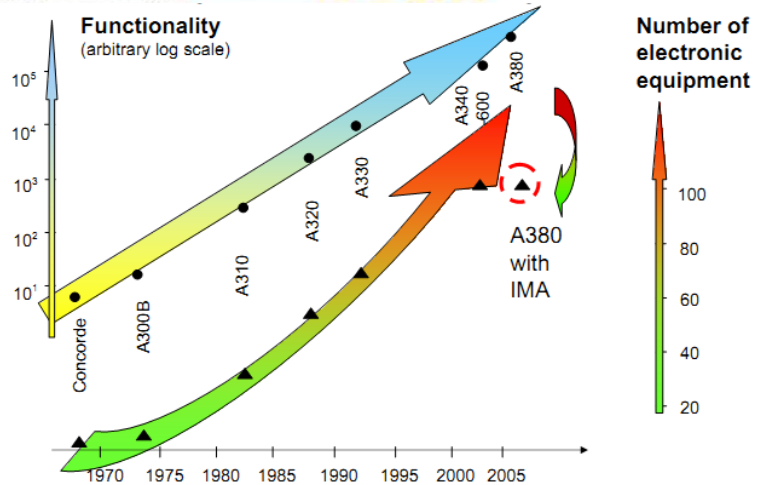
- 機能安全・機能(ECU)統合
  - 安全度水準が異なる複数のアプリケーションを実行
  - 別々のコンピュータ上で実行されていたアプリケーションの一つのコンピュータ上に統合して，コンピュータ数を減らす

組込みシステムにおける仮想マシンの必要性

- 機能統合例
  - 航空機のAvionicsの統合Integrated Modular Avionics(IMA)
  - A380や777から導入されつつある

IMA前

IMA後



出典: Jean-Bernard ITIER,
"A380 Integrated Modular Avionics",
ARTIST2 meeting on integrated modular avionics, 2007

組込みシステムにおける仮想化の要件

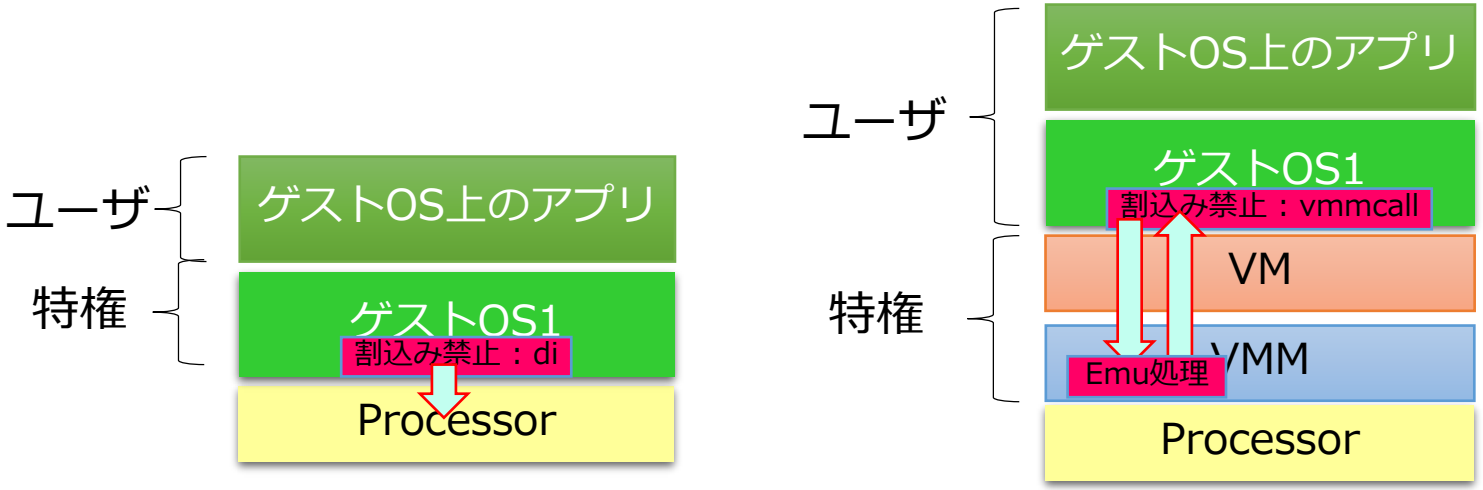
汎用システム向け仮想化の要件に加え以下のような要件がある

- リアルタイム性
  - 機器制御を行う場合
- リソース制約
  - 汎用PC程の性能やメモリ容量がない
- 多様なデバイスのサポート
  - 汎用PCではディスクやネットワークで十分
- 信頼性
  - 人命や高価な物を扱う機器を制御する場合がある
- 完全仮想化の必要性は低い
  - とはいえ，規模が大きいOSの変更は極力避けたい（検証が困難）
    - 変更が多いと時代進化について行けない.

ハードウェア仮想化支援機能

仮想化の実現

- 通常のプロセッサによる仮想化の実現
  - ゲストOSの変更が必要（準仮想化）
    - 例）割込み禁止処理
      - ✓ユーザーモードで実行すると例外が発生する
        - → VMMへの依頼に変更.
  - 実行性能が低下する
    - 例)割込み禁止処理
      - ✓オリジナル：1命令
      - ✓準仮想化　：数百命令(VMM呼び出しとエミュレーション処理)



PopekとGoldbergの仮想化要件

- コンピュータアーキテクチャが効率的にシステム仮想化を実現するための条件（仮想化要件）
  - Gerald J. Popek and Robert P. Goldberg, "Formal Requirements for Virtualizable Third Generation Architectures“,Communications of the ACM 17 (7): 412 –421. 1974.
- 命令セットを３種類に分類
  - 特権命令
    - ユーザーモードで実行するとトラップする命令
  - 制御センシティブ命令
    - システム資源の構成（再配置レジスタやプロセッサモード）を変える命令
  - 動作センシティブ命令
    - システム資源の構成に，動作や結果が依存する命令

- 仮想化要件
  - センシティブ命令が特権命令であれば仮想マシンを構築することができる
    - プログラムをユーザーモードで動作させることにより，仮想マシンの動作に影響を与える命令をトラップすることが可能.

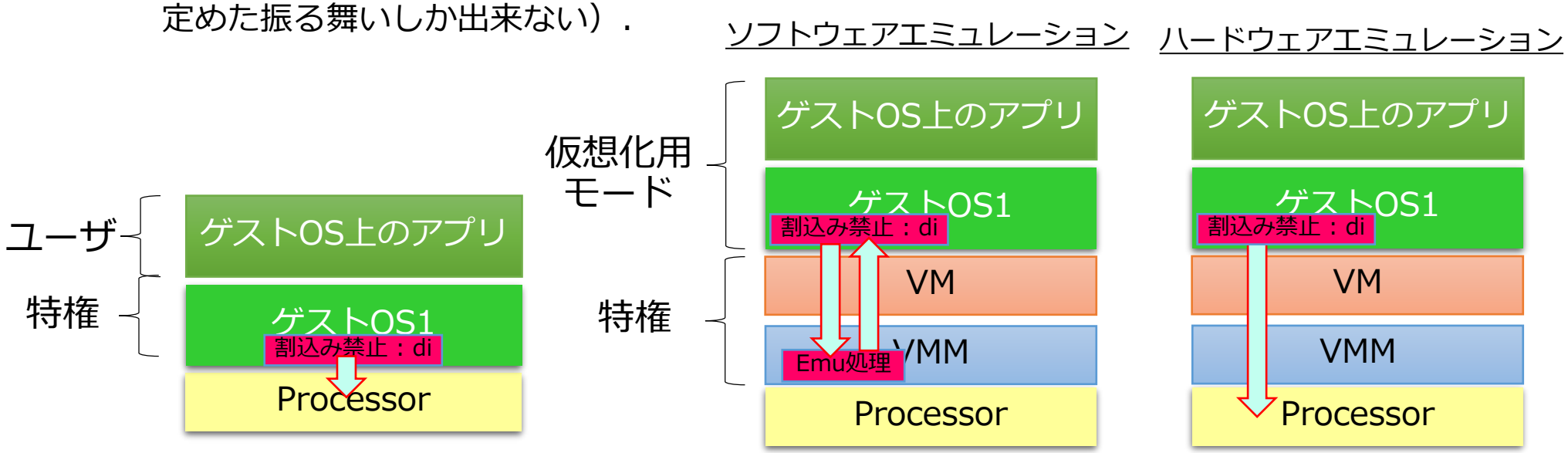
ハードウェア仮想化支援機能

- PopekとGoldbergの仮想化要件を満たし仮想マシンの実行効率を向上させるためのプロセッサの機能（既存のプロセッサを拡張することが一般的）
  - 完全仮想化が可能となる
    - ゲストOSのソースコードを変更できない場合に有効
  - 性能低下を抑制
    - センシティブ命令によるトラップの発生回数を削減
      - ✓例)VMM毎にレジスタを持つ

- ハードウェア仮想化支援機能の例
  - x86
    - Intel VT, AMD-V等
  - ARM
    - TrustZone, Virtualization Extension
  - RH850
    - RH850 Hardware-assisted Virtualization(HAV)

ハードウェア仮想化支援機能の例

- 例）割込み禁止処理
  - ソフトウェアエミュレーション方式
    - 割込み禁止命令を実行すると例外が発生してVMMが呼び出される.
    - VMMにより動作をエミュレーション
  - ハードウェアエミュレーション方式
    - 割込み禁止命令を実行するとハードウェアが直接エミュレーション
- 方法はプロセッサによって異なる
  - どちらも完全仮想化は実現
  - 性能はハードウェアエミュレーションの方が高速であるが，柔軟性が低い（ハードウェアで定めた振る舞いしか出来ない）.



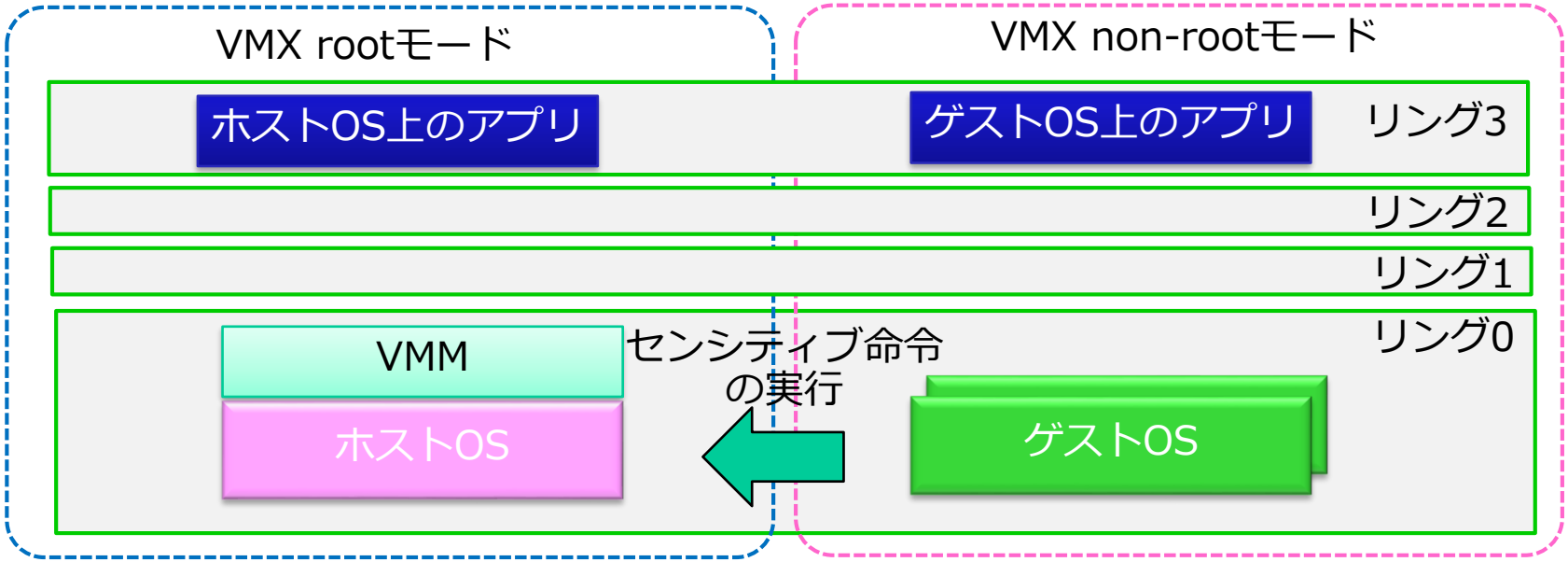
ハードウェア仮想化支援機能：x86

- PopekとGoldbergの仮想化要件を満たしていない.
- 非特権のセンシティブ命令が17個存在
  - センシティブなレジスタの操作
  - システム保護命令

- ハードウェア仮想化支援機能を持たないx86向けの仮想マシンは上記の命令を動的に変換している.
  - VMWareは変換方法の特許を取得
  - パフォーマンスが低下
    - 準仮想化により性能低下を防ぐ
      - ✓センシティブ命令を事前に置き換える.

ハードウェア仮想化支援機能：x86 Intel VT

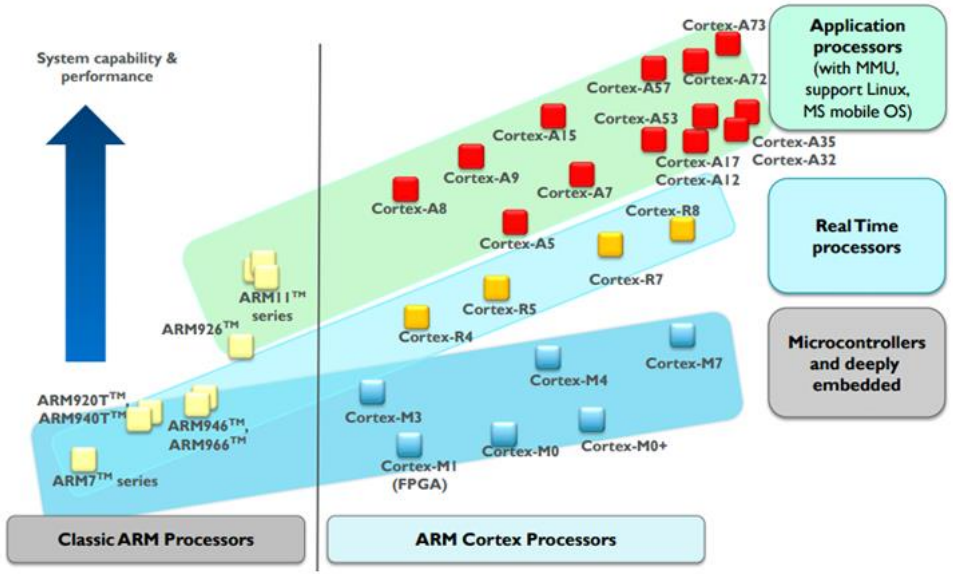
- ゲストOS用新しいモードを追加(VMX non-rootモード)
  - 従来のリングプロテクションとは直交している
  - non-rootモードで，センシティブ命令が実行されると，プロセッサ状態を保存してrootモードを呼び出す.
  - MMUは階層化されていない
    - VMMにより仮想化する必要がある



車載システム向けプロセッサの
ハードウェア仮想化支援機能

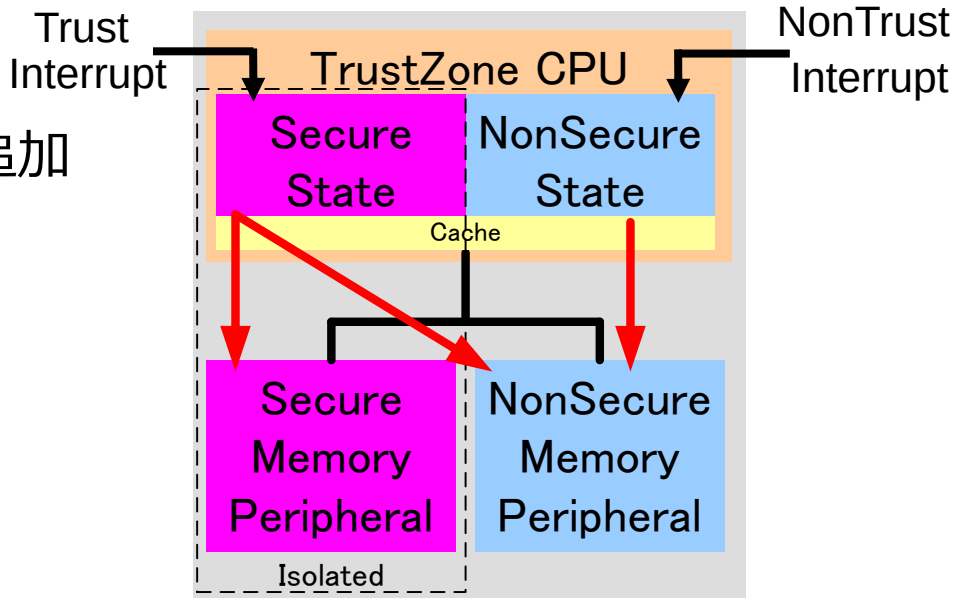
車載システム向けプロセッサのハードウェア仮想化支援機能

- インフォテイメント及びドメインコントローラ向け
  - ARM Cortex-A TrustZone
  - ARM Cortex-A Virtualization Extension(VE)
- 制御システム向け
  - ARM Cortex-R Virtualization Extension(VE)
  - ARM Cortex-M TrustZone
  - ルネサスエレクトロニクス RH850 Hardware-assisted Virtualization(HAV)



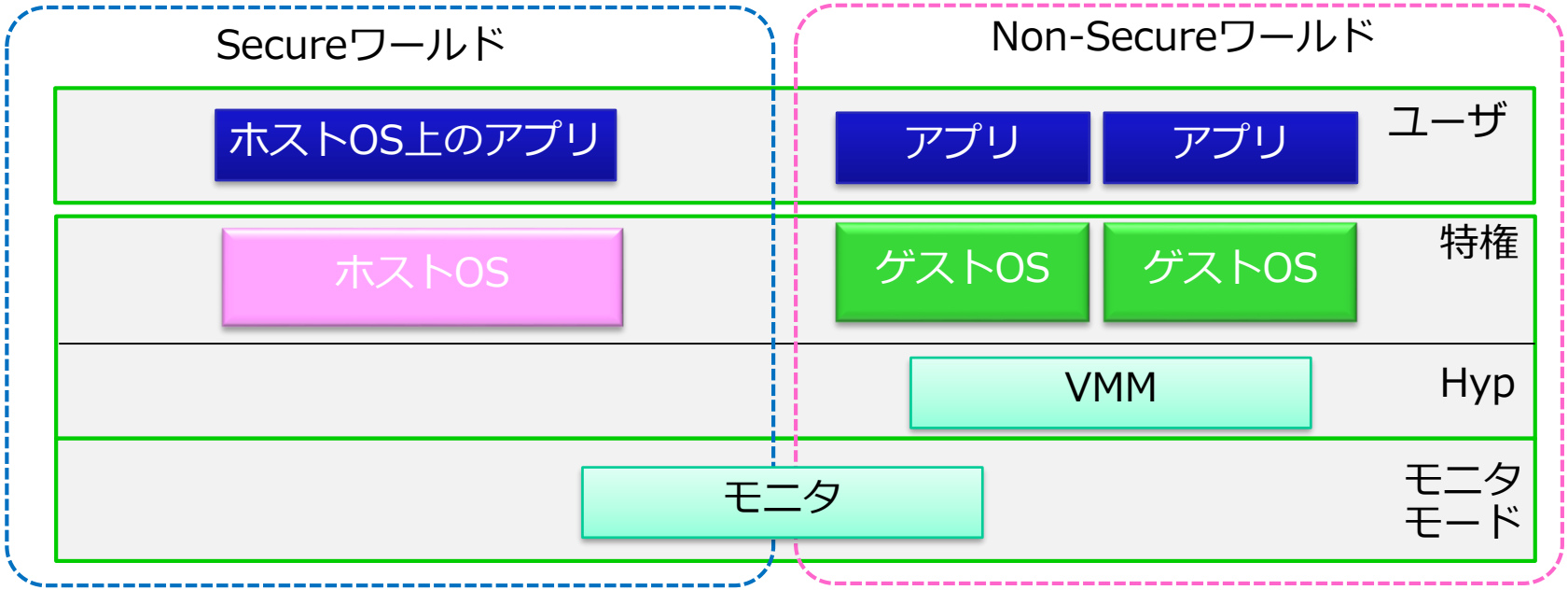
Cortex-A TrustZone

- Cortex-A
  - カーナビやドメインコントーラ等に使用される高性能プロセッサ
- 実行に制限があるNon-Secure状態を追加
  - これまでの状態はSecureと呼ぶ
- 仮想アドレス空間は，Non-Secure と Secure で独立
  - アドレス変換テーブルは独立に持つ
  - TLBやキャッシュは共有
    - どちらのエントリか示すビットが追加
- 周辺デバイス
  - デバイスは共有せず，それぞれのワールドにアサインする
  - NonSecureからSecureのデバイスにアクセスすると例外発生
  - 割込みもそれぞれのワールドで独立に持つ



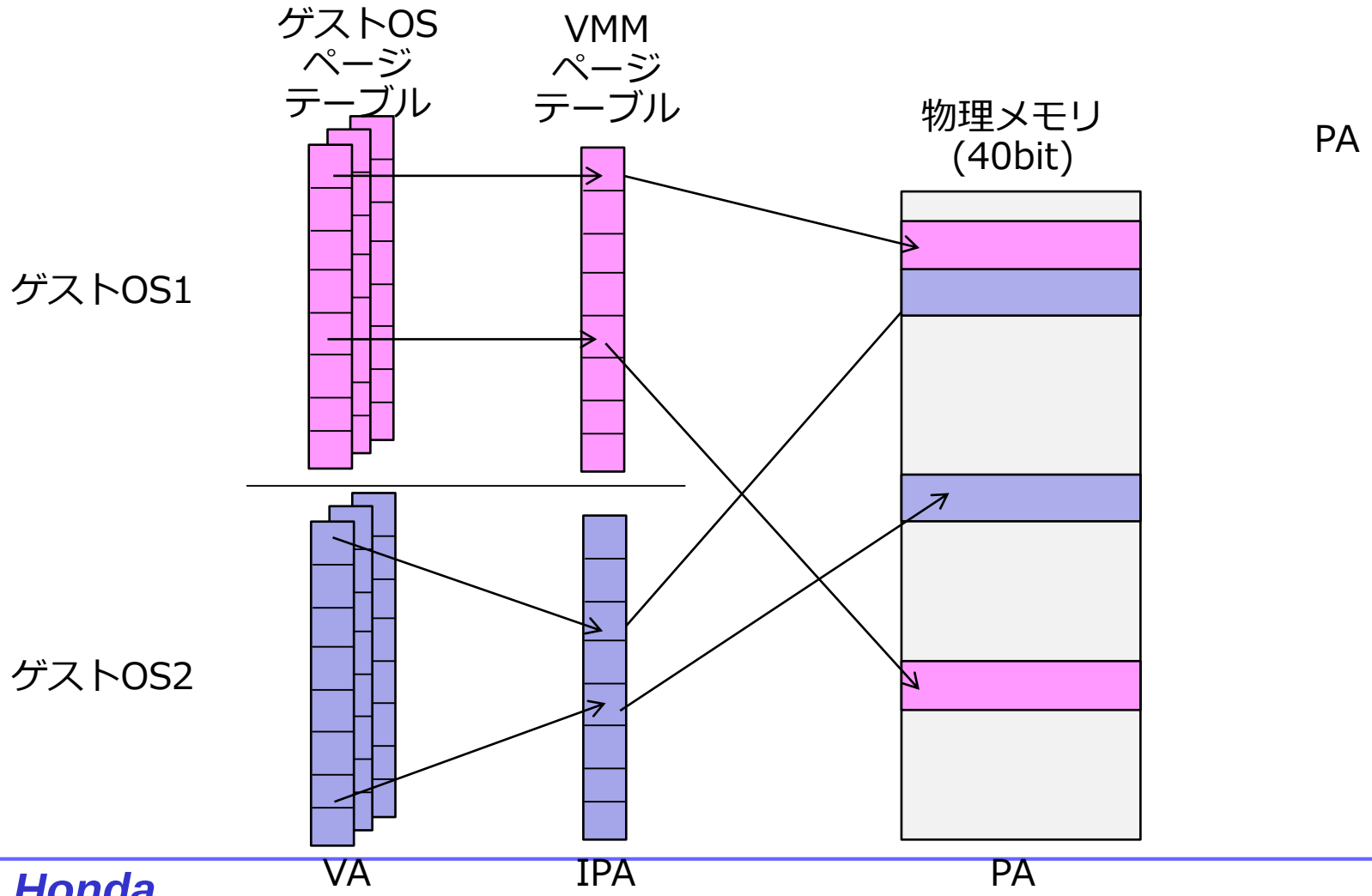
Cortex-A Virtualization Extension(VE)

- 完全仮想化を可能に
  - 一部ユーザーモードで実行してもトラップを発生しない特権命令があった
- ハイパーバイザー用の特権モードを追加（Hypモード）
  - TrustZoneとは直交した概念
- センシティブ命令のトラップを効率化
  - トラップの原因を詳細に取得可能
- ネイティブ型とホスト型(ARMv8.1以降)の両方をサポート



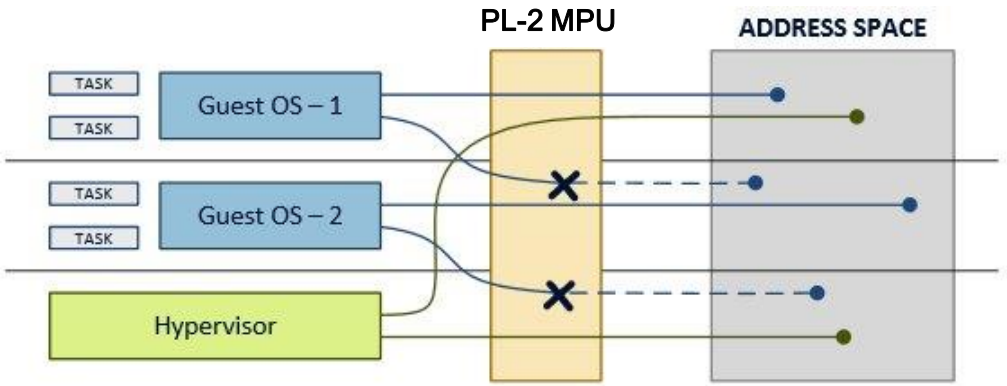
Cortex-A Virtualization Extension(VE)

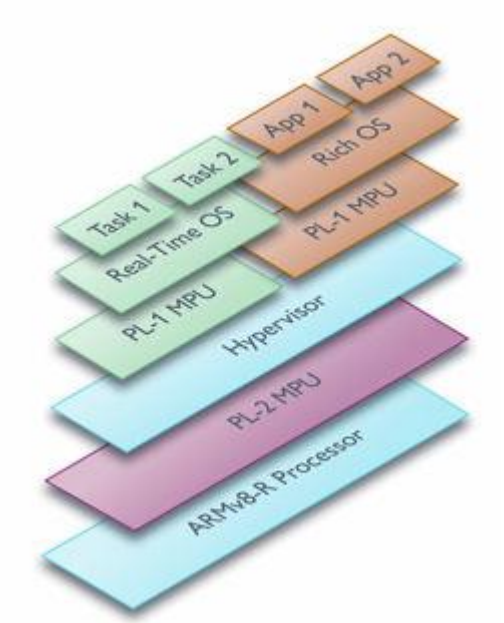
- 2レベルのアドレス変換をサポート
  - これまではVMMがページテーブルアクセス命令をトラップして実現



Cortex-R Virtualization Extension(VE)

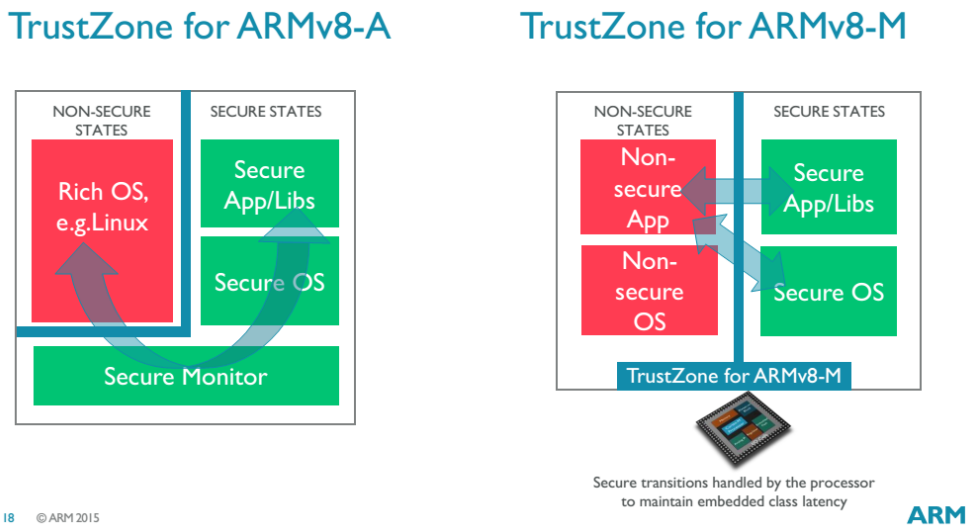
- 安全システム（車載制御や医療機器）のためのハードウェア仮想化
  - アプリケーション統合を目的にしている
- ARMv8-Rでサポート
  - 2013年に仕様の概要が公開，2016年にマイクロアーキテクチャが公開 (Cortex-R52).
  - Cortex-A VE と基本構成は同じだがMPUのみサポート
- 2段階のMPUを使用して仮想化を実現
  - Hypervisorが2段目のMPUを制御して各仮想マシンのアクセス可能なメモリを制限
  - ソースコードの再コンパイルは必要



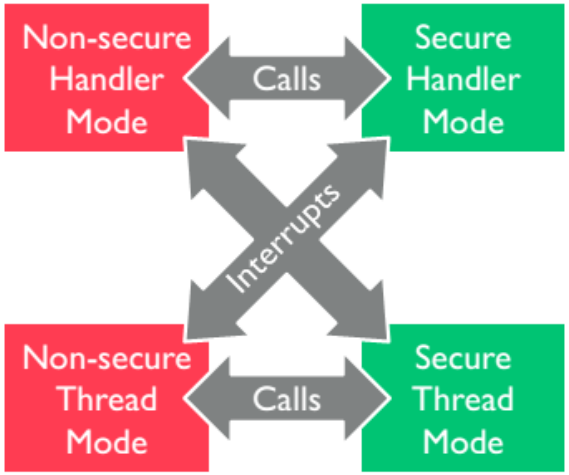


Cortex-M TrustZone

- 元々は小型家電向けであるが車載向けマイコンにも搭載されつつある
- Cortex-A TrustZone との違い
  - ハードウェアがコンテキストの保存・復帰はハードウェアで実施
    - この点ではVMMが必要ない
  - 割込み発生時に自動的に保存
- Secureの関数を直接Non-Secureから呼び出し可能
  - 呼び出しに制限を設けることで安全性を担保

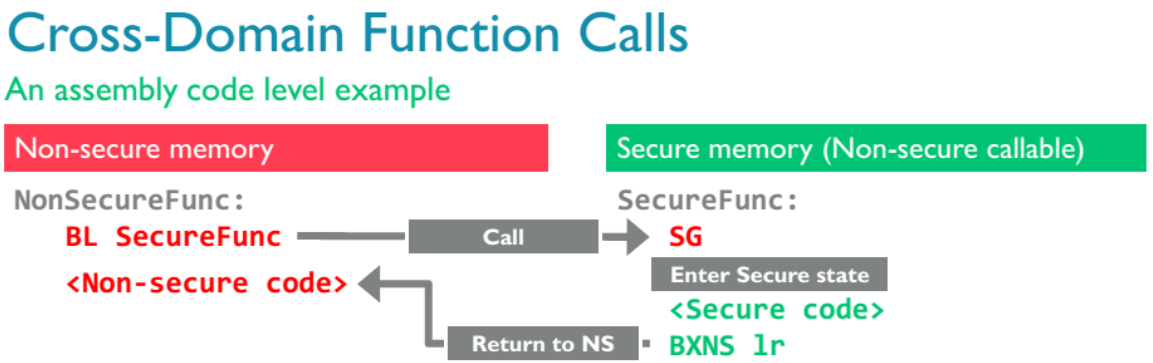


出展: ARM



Cortex-M TrustZone：関数呼び出し

- Non-Secure側から呼び出し可能なアドレスを制限する
  - Non-Secure Callable（NSC）領域と設定されたメモリのSecure Gateway(SG)命令が置かれたアドレスのみ実行可能
  - SecureからNon-SecureにリターンするBXNS命令を追加
- Non-Secure側から呼び出す関数はattributeを付けてコンパイル
  - SGとBXNSを自動的に付けてくれる
  - リテラルプール配置しないようにする



出展：ARM

```
int MySecureFunc(int v)
    __attribute(NSEntry)
{
    return v + 1;
}
```

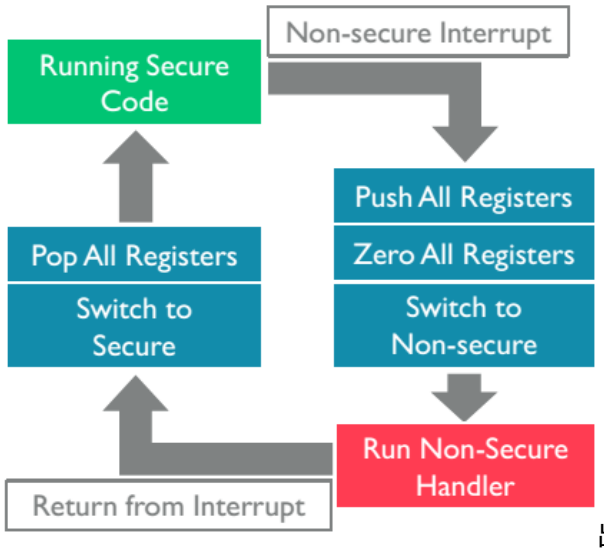
```
MySecureFunc:
    SG
    ADDS r0,r0,#1
    BXNS lr

extern MySecureFunc();
```

リテラルプール

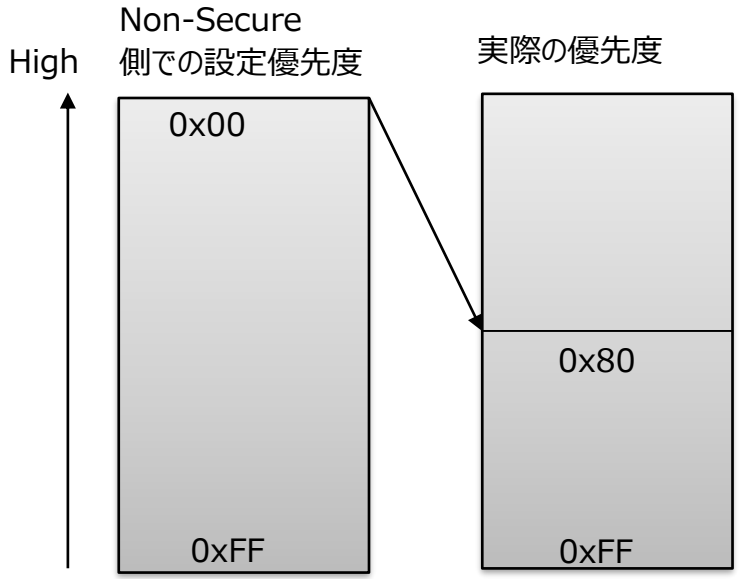
```
LDR r0, =magic_num
magic_num:
    .long 0x23292828
```

Cortex-M TrustZone ：割込み

- オリジナルのM-Profileの割込み機構
  - 割込みが発生するとハードウェアが呼び出し元保存レジスタ(callee saved register)をスタックに保存してからハンドラを実行する
- TrustZone拡張
  - secure 実行時にnon-secureの割込みを受け付けた場合は全てのレジスタをsecure側のスタックに保存して，レジスタの値をクリアしてハンドラを実行する
  - Non-Secure側の割込み優先度を下位半分の優先度にマップすることも可能
    - Secure側のハンドラの優先度が常に高い



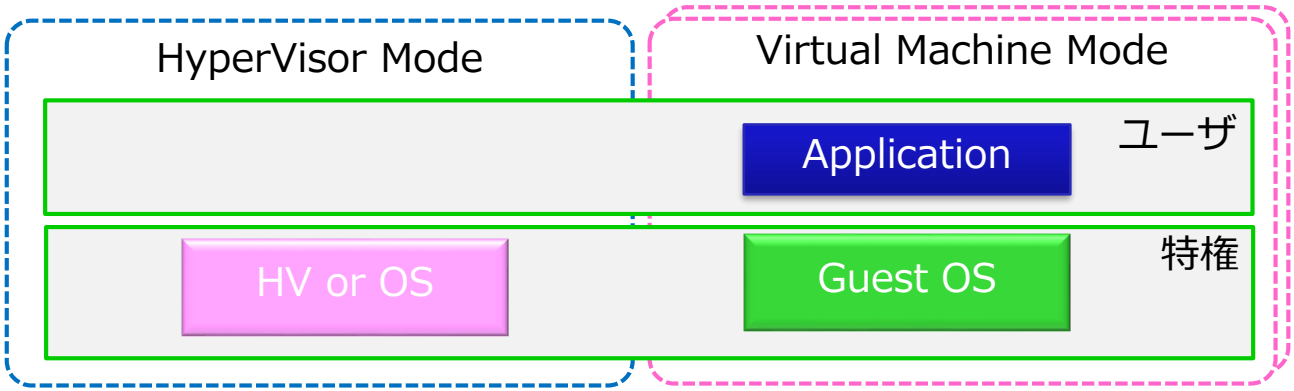
出展: ARM



RH850 Hardware-assisted Virtualization(HAV)

- RH850
  - パワートレイン等の高い性能とリアルタイム性が要求されるECUで使用されている32bitプロセッサ

- Virtual Machine 特権モード（VMモード）
  - 既存のモードと独立したモード（既存のモードをHV特権モードと呼ぶ）
  - プロセッサリソースへのアクセスに制限
  - VM用に用意されたプロセッサリソース（VMコンテキスト）
    - 1命令で退避・復帰が可能な命令を持つ



RH850 Hardware-assisted Virtualization(HAV)

- VM識別子（GPID）
  - VM毎に割り付けるIDとその値を保持するレジスタ(GPIDレジスタ)
  - GPIDレジスタの内容はVM切り換え時にHVにより内容を変更する
- 割込み
  - 特定のGPIDないしHVに割り付け可能
  - 特定のGPIDに割り付けた割込みは，GPIDレジスタの値が一致する場合のみ割込みが受け付けられる
  - HVに割り付けた割込みはVMの状態によらずに受け付けられる
- MPU
  - 標準のRH850より領域数が多い
  - 2段階の保護
    - 各領域をVM-MPUとHV-MPUに設定可能
    - VM-MPU：VM内のOSが自由に設定可能
    - HV-MPU：HVにより設定が可能
  - VM-MPUで許可された領域であってもHV-MPUで禁止されていれば，VM内からはアクセス出来ない
  - 複数の領域を1命令で退避・復帰が可能な命令を持つ

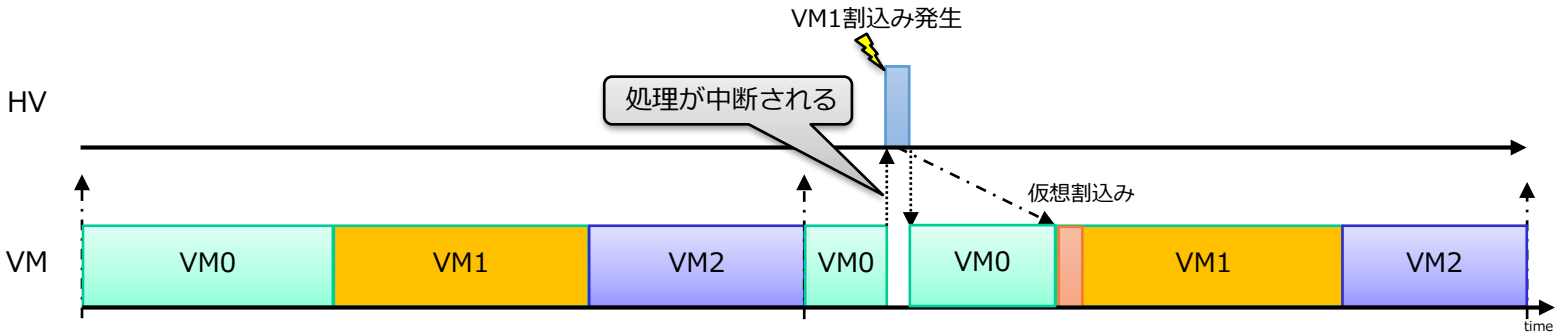
比較

	Cortex-A TrustZone	Cortex-A VE	Cortex-R VE	Cortex-M TrustZone	RH850 HAV
VM数*1	1	256	256	1	32
実現方法	ネイティブ型	ネイティブ型 ・ホスト型	ネイティブ型	HVなし	ネイティブ型 ・ホスト型
VMの実現*2	完全仮想化	完全仮想化	完全仮想化	完全仮想化	完全仮想化
VMの再リンク	必要	不要	必要	必要	必要
システムレジスタの多重化度	ほぼ多重化	少	少	ほぼ多重化	中
VM・HV切り替え速度	中	低	低	高 HWで保存・復帰	中 保存・復帰命令
他VMの割込みによる一時中断	-	あり	あり	-	なし
自割込みの受付け方式	ダイレクト	HV経由	HV経由	ダイレクト	ダイレクト
VM→HV呼び出し方法	Trap呼出し	Trap呼出し	Trap呼出し	関数呼出し	Trap呼出し
メモリ保護	バスガード機構*3	2階層MMU	2階層MPU	2階層MPU*4+バスガード機構*3	2階層MPU+バスガード機構*3

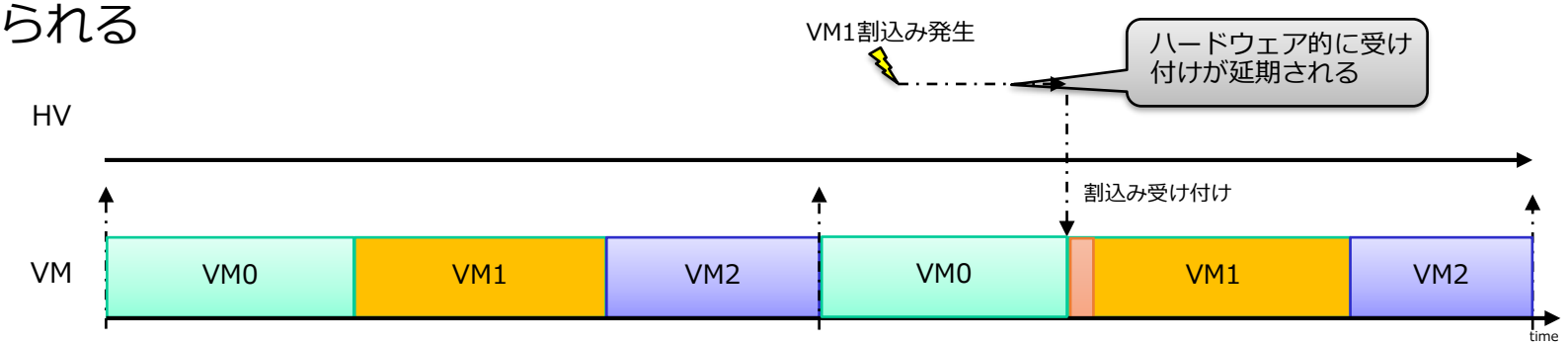
- *1 2以上の場合はVMを識別するIDの数
- *2 デバイスやクロック系の初期化や操作は除く
- *3 バス，メモリ，周辺回路側でアクセスするVMを判断して保護を実現
- *4 実際にはMPUの後段のSAU

他VMの割込みによる一時中断

- 実行中のVM以外のVMに割り付けた割込みが発生した場合の振る舞い
- 中断あり：Cortex-A/R VE
  - 実行中のVMを一旦停止してHVが動作して割込みの扱いを決定する

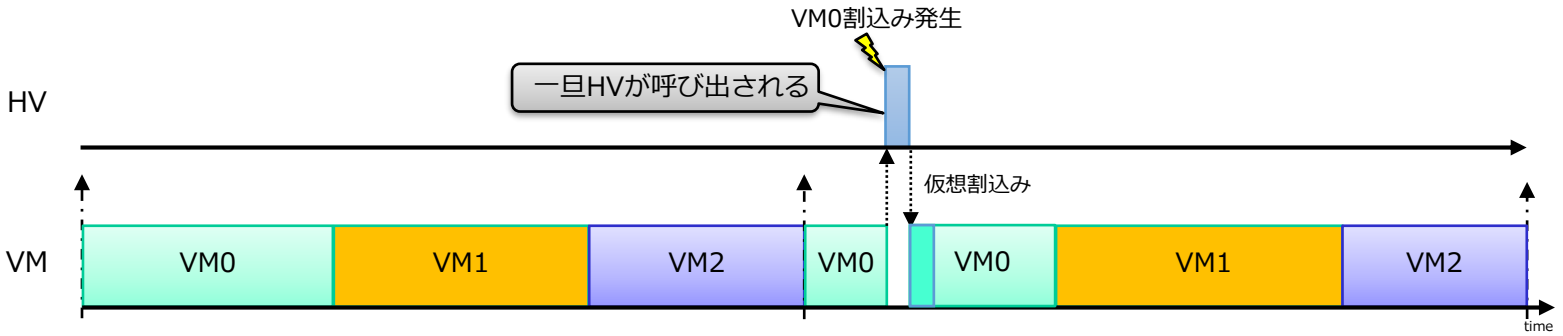


- 中断なし：RH850 HAV
  - ハードウェア的に受け付けず，割り付けられたVMが動作すると割込みが受け付けられる

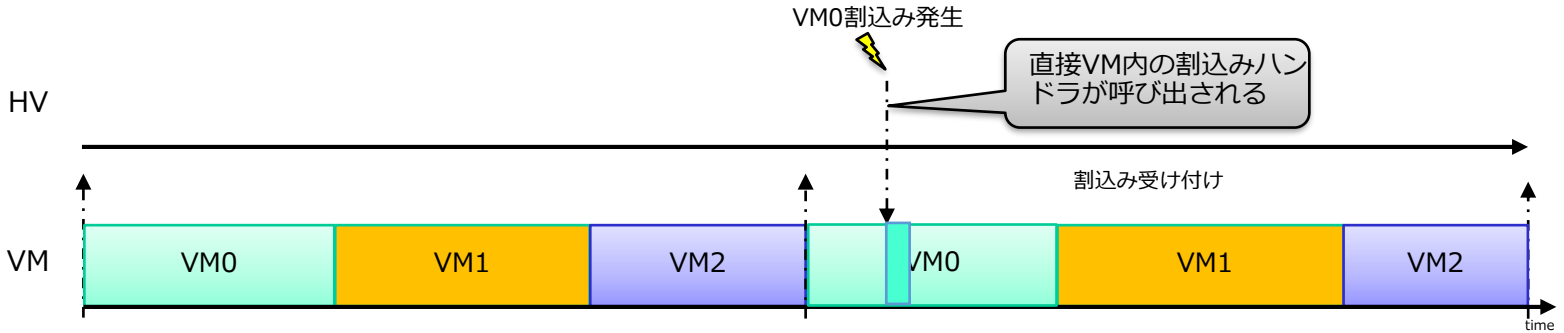


自割込みの受付け方式

- HV経由：Cortex-A/R VE
  - HVが一旦動作してからVMに割込みが通知される



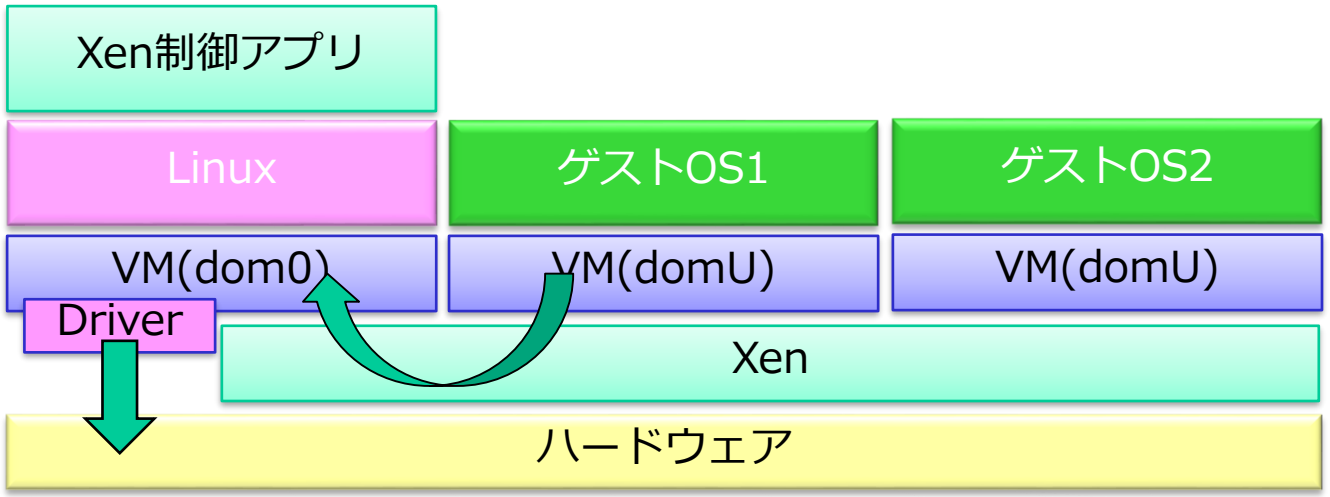
- ダイレクト：Cortex-A/M TrustZone, RH850 HAV
  - 実行中のVMが直接割込みを受け付ける



仮想マシンモニタ(VMM)・ハイパーバイザー
インフォテイメント及びドメインコントロー
ラ向け

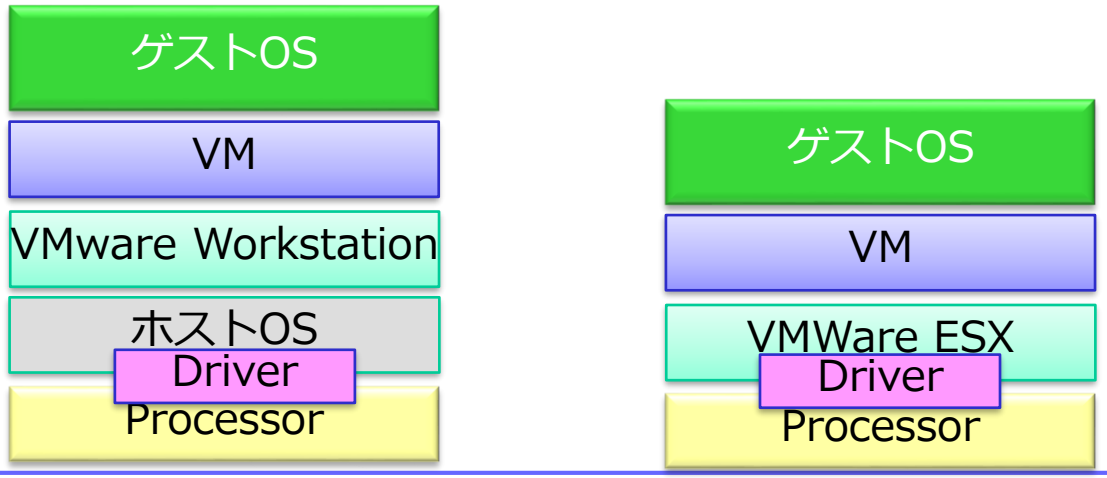
Xen

- ベアメタル/ネイティブ型でデバドラの問題を解決
- 準仮想化/完全仮想化(ハードウェアが仮想化支援を持つ場合)
- デバイス仮想化はdom0という特別なLinuxで実現
  - –ゲストOSからのハードウェアアクセス要求をXenがdom0のLinuxに伝え，dom0のLinuxがデバイスをアクセス
- 組込み向けプロセッサのARM向けの開発が進んでいる



VMWare

- 完全仮想化をサポート.
  - –準仮想化により性能向上
- VMware Workstation
  - –ホスト型（Type 2）
- VMware ESX
  - –ベアメタル型（Type 1）
  - –サービスコンソールとしてLinuxを実行
  - –ホストOSがないため信頼性が高い

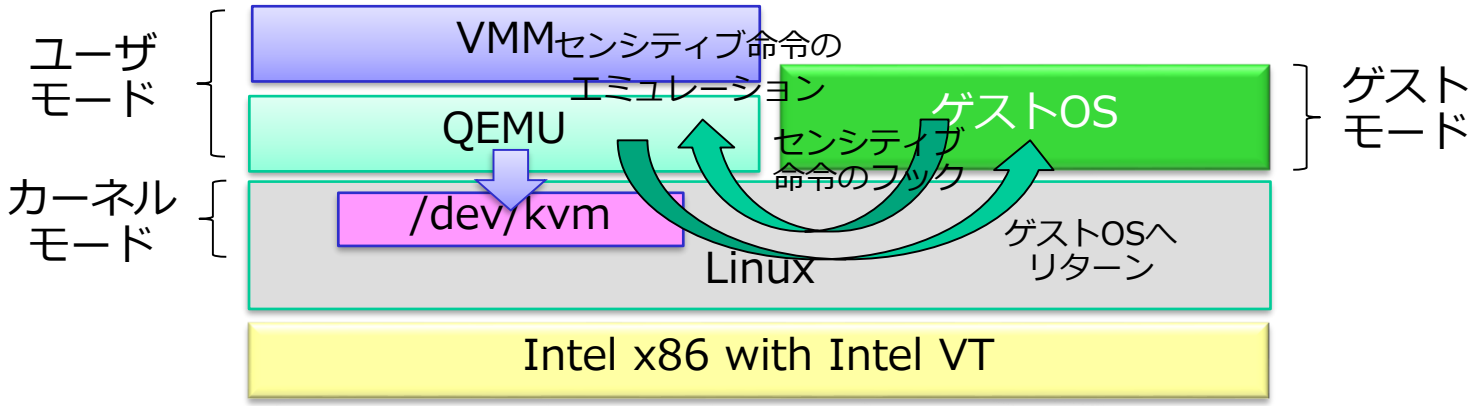


KVM

- Linuxの2.6.20以降に搭載
- Intel VT,AMD-Vを使用した完全仮想化
  - ゲストモード（VMX non-root）が追加される
  - Linux/Solaris/Windows が動作.
- ゲストOSはQEMUにより実行される.
  - 周辺装置のエミュレーションはQEMUを使用
- /dev/kvm
  - ハードウェアの仮想化を管理するためのデバドラ
- 必要に応じてパススルー方式でデバイスを直接アクセスさせる
- メモリ変換は"SPT"(Shadow Page Table)を使用している.
  - VMMがゲストOSのページテーブル（実際には使われない）を参照して本物のテーブルを作成する方法.
- 組込みシステムへの展開
  - ARM Virtualization Extension を用いたバージョンも開発.

KVM

- 仕組み
  - QEMUが/dev/kvm経由で仮想マシンの作成し実行
  - カーネルがゲストモードでゲストOSを実行
  - ゲストOSがセンシティブ命令を実行すると，例外が発生してカーネルに処理が移る.
  - カーネルは仮想化支援レジスタをチェックして何が起きたかチェックしてQEMUを呼び出す
  - QEMUは必要な処理をしてホストOSに戻る
  - ホストOSからゲストOSに戻る



仮想マシンモニタ(VMM)・ハイパーバイザー
組込みシステム向け

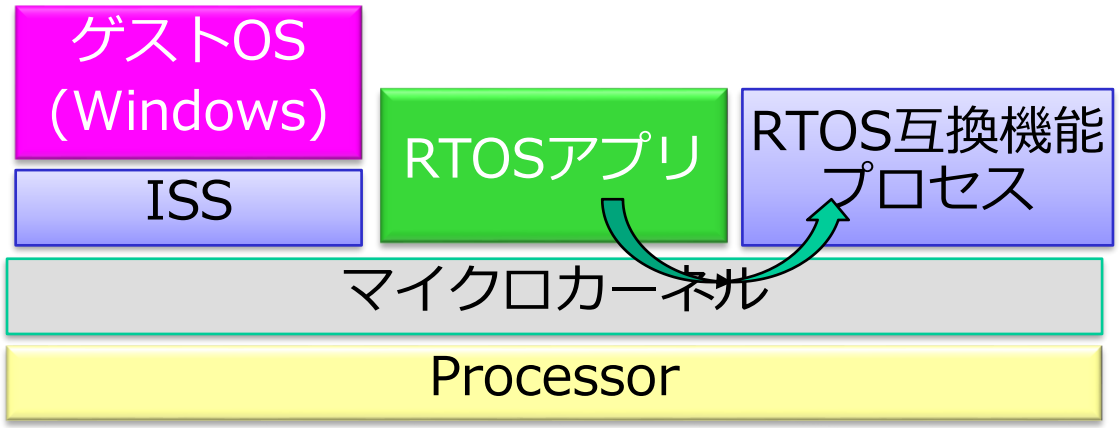
組込みシステム向け仮想マシン

- 基本的にベアメタル/ネイティブ型（Type 1）
  - リソース制約が厳しいため
- ARINC 653(後述)対応のものが多い
- マイクロカーネルベースのものが多い
- 商用ソフトウェアが多く技術の詳細が不明な場合が多い

- Xen-ARM
- PikeOS
- INTEGRITY Multivisor
- OKL4

仮想マシンの構成例：マイクロカーネル

- 最低源の機能のみを持つカーネル
- 他の機能はプロセス(サーバーとして)実現し，その間のIPCで接続することによりシステムを実現する.
- プロセスとして，ISSを実行したり（完全仮想化），他のOSを変更して実行することにより（準仮想化），仮想化が可能.
  - この場合，マイクロカーネルをハイパーバイザーと呼ぶことが多い（例：L4）.
- 特定のOSのAPIをサービスとして提供することで，仮想化とする場合もある.

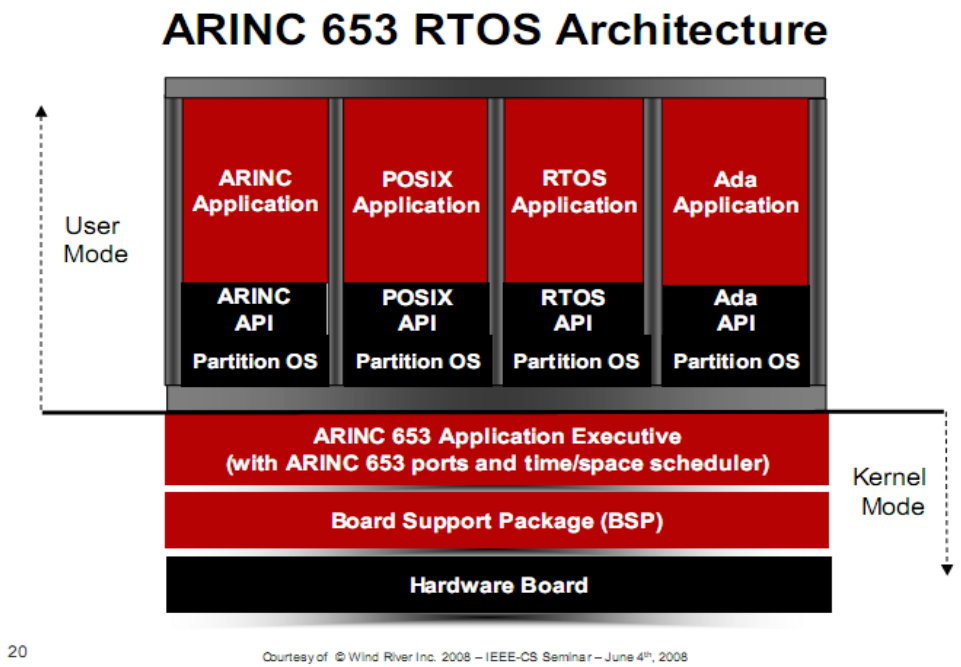


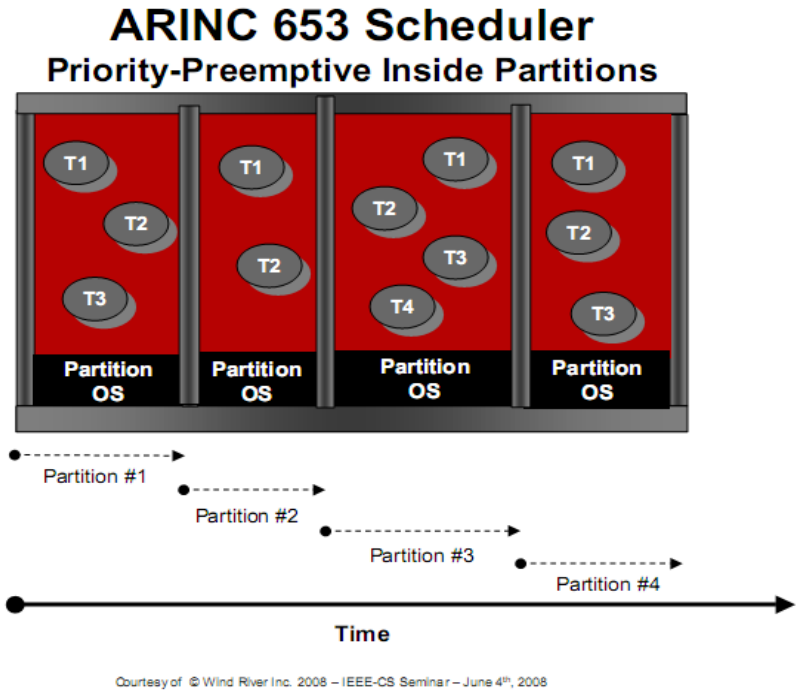
ARINC(Avionics Application Standard Software Interface) 653

- 航空電子機器用のソフトウェアプラットホーム仕様
  - Integrated modular avionics (IMA) 用 (ECU統合と同じ)
- 特徴
  - 個々のアプリケーションはパーティションと呼ばれる
    - アプリケーション内にはプロセスが存在
    - 複数のパーティションが集まってモジュールとなる.
    - 複数のモジュールが集まってシステムを構成する.
  - 51個のAPI APEX(APplication/EXecutive)により構成
    - メモリパーティション
    - 時間パーティション
    - ヘルスモニタリング（エラーディテクション，レポート）
    - ポートによるパーティション間のコミュニケーション
  - アプリ開発におけるメリット
    - ポータビリティ，再利用性，モジュール化
  - ARINC 653 準拠OS及びアプリはDO-178Bで認証される
    - 個々のアプリは異なるDO-178Bのレベル（A-E）で認証可能

ARINC 653 対応OSの構造とスケジューラ

- TDMAベースでVMをスケジューリング
  - Time Division Multiple Access
- VM内ではOSが動作してタスクをスケジューリング
  - 階層型スケジューラ構成





出典 : Wind River Systems / IEEE Seminar. August 2008.

Xen-ARM/PikeOS

- Xen-ARM
  - 08年にARMv4を対象にSamsungが開発をスタート
    - オープンソース
  - スマートフォンが主なターゲット
  - 構成はx86と同等
    - ドライバやMMUの操作はXenに依頼
  - Cortex-A VEをサポート

- PikeOS
  - マイクロカーネルベース
  - SYSGO社により開発
  - 準仮想化をサポート
  - 特権モードで動作するプログラムを少なくすることにシステムの信頼性を確保
  - 航空機のIMAでの利用実績有り（AirBus A350/A400M）
  - ARINC 653サポート

INTEGRITY Multivisor/OKL4

- INTEGRITY Multivisor
  - マイクロカーネルベース
  - Green Hills Software 社により開発
    - 元はコンパイラメーカ
  - 元は高信頼性システム向けOS
    - メモリ保護/時間保護
  - ARINC 653 ベースのパーティションスケジューラを装備
  - 仮想化支援ハードウェアを使う事により完全仮想化をサポート

- OKL4
  - マイクロカーネルベース
  - オープンソース
  - Open Kernel Labsにより開発
  - L4マイクロカーネルの亜種
    - 第二世代のマイクロカーネル（第一世代はMach）
  - 準仮想化をサポート

仮想マシンモニタ(VMM)・ハイパーバイザー
車載システム向け

車載システム向け

- COQOS Micro SDK（OPENSYNERGY）
  - 車載向けのハイパーバイザー
  - Cortex-R VE を使用
  - SYSGO PikeOS の車載版

- EB tresos Embedded Hypervisor
  - 車載向けのハイパーバイザー
  - Cortex-R VEを使用

- ETAS RTA Lightweight Hypervisor(ETAS)
  - 車載向けのハイパーバイザー
  - PowerPC （ハードウェア仮想化支援機能は使用しない）
  - アプリケーションコアでユーザーモードでOSを含むシステムを動作させる
  - 仕様書が公開されている

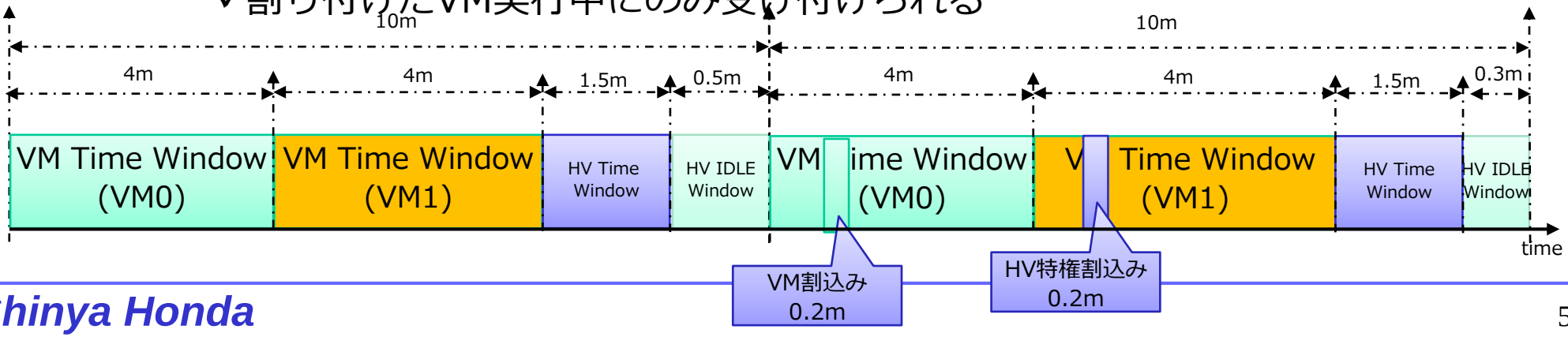
仮想マシンモニタ(VMM)・ハイパーバイザー
RH850 HAV を用いたハイパーバイザ

RH850 HAV を用いたハイパーバイザ

- ルネサスエレクトロニクスとの共同研究で開発しているハイパーバイザー
- 2018年度
  - 開発を開始.
  - シミュレータ上での基本機能の確認
- 2019年度
  - シングルコア対応.
  - 試作チップでの性能評価
  - AUTOSAR OSの実行
- 2020年度
  - マルチコア対応
  - VM間通信
    - ハードウェア支援機能の活用
  - TDMAスケジューリングの拡張
    - 割込みに連動した動的なVMスケジューリング
- 2021年度予定
  - デバイス共有

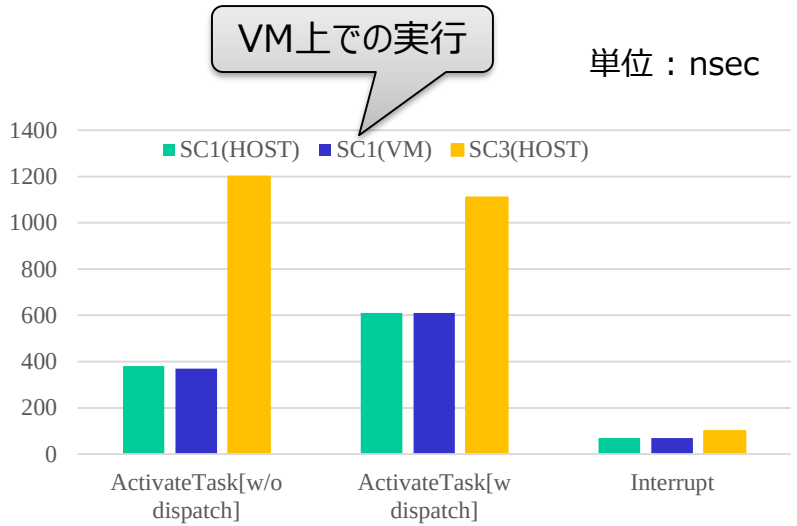
RH850 HAV を用いたハイパーバイザ：仕様

- 時間保護
  - TDMAスケジューリング
    - VMタイムウィンドウ(n個)：VMを実行
    - HVタイムウィンドウ(1個)：HV特権モードでハンドラを実行
  - HV IDLEウィンドウ：HV特権モードでアイドル処理を実行
- 割込み処理
  - VMないしHVに割り付ける
    - HV割込み
      - ✓どのタイムウィンドウでも受け付けられる
      - ✓受け付けた分タイムウィンドウの実行終了は後ろにずれる
    - VM割込み
      - ✓割り付けたVM実行中にのみ受け付けられる

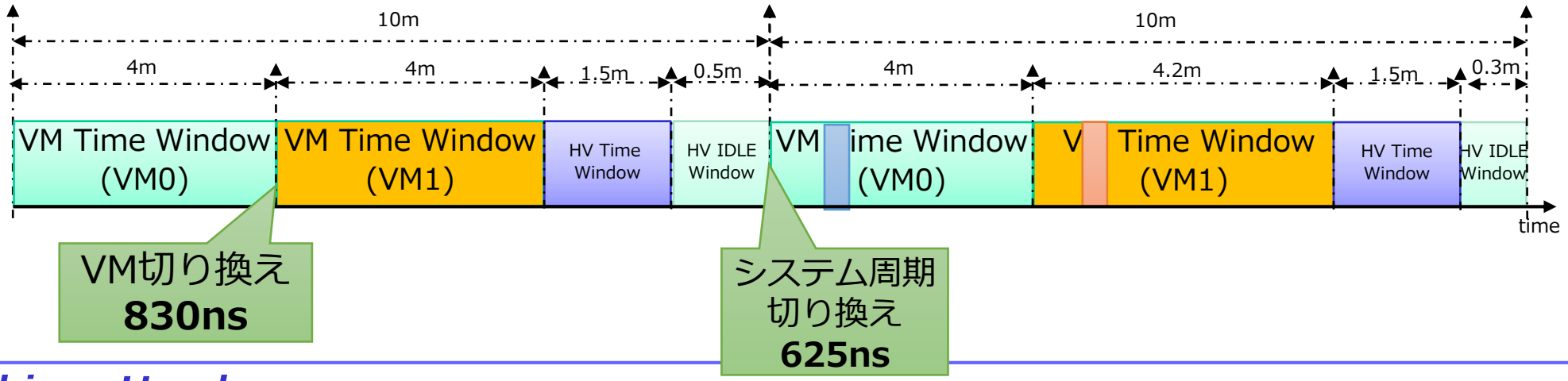


RH850 HAVを用いたハイパーバイザ：実行オーバヘッド

- 評価環境
  - HAVを搭載した研究用試作プロセッサ(400Mhz)
- RTOS(ATK2)のタスク切り替えとの比較
  - VM切り換えは機械的にVMコンテキストを切り換えるのに対して，RTOSは，エラーチェックや管理情報の更新等が必要となる
- VMでの実行によるRTOS(ATK2)の実行オーバヘッド
  - 実行オーバヘッドの発生はない



RTOSのタスク切り替え時間



RTOS の変更量評価/HVの規模評価

- VM上で動作させるRTOS の変更量評価
  - コンフィギュレーションレベルの変更
    - システム全体に影響するハードウェアの操作の削除.
      - ✓クロックや IO ポートの初期化処理
    - 使用するハードウェアリソースの変更
      ROM/RAM やタイマや UARTのチャネルを変更
- HV の規模評価
  - コンフィギュレーション
    - HV ：2 個のVM（VM毎280byteのコンテキスト保存領域が必要）
    - 保護無し A-OS ：11 個のタスク
    - 保護付き A-OS ：21 個のタスク
  - RTOSと比較して規模が小さい
    - 検証が容易

byte

対象ソフト	ROM	RAM	RAM(w/o stack)
HV	8,310	2,616	1,592
保護無し A-OS	48,116	9,342	2,830
保護付き A-OS	90,765	20,614	5,510

各ハイパーバイザーの比較

	RH850 HV	OPENSYNERGY	EB	ETAS
対応プロセッサ	RH850U2	Cortex-R52 (S32S)	Cortex-R52	PowerPC
仮想化ハードウェアの使用	Yes	Yes	Yes	No
スケジューリング	TDMA拡張*4	TDMA拡張*4	TDMA拡張*1	TDMA拡張*2
割込みのサポート	Yes	Yes	Yes	No
VMからデバイスのアクセス	Yes	Yes	Yes	No
マルチコアサポート	Yes*3	Yes	Yes	Yes
VM間通信	Yes*3	Yes	Yes	Yes

- *1 仕様を検討中と見られる
- *2 動的なスケジューリングをサポート
- *3 2020年度に開発
- *4 割込み拡張

まとめ

まとめ

仮想化技術の基礎について説明し，車載システム向けの仮想化のためのハードウェア及びソフトウェアについての現状について解説する.

- 仮想化技術
- 組込みシステムにおける仮想化
- ハードウェア仮想化支援機能
  - 車載システム向けプロセッサのハードウェア仮想化支援機能
- 仮想マシンモニタ(VMM)・ハイパーバイザー
  - インフォテイメント及びドメインコントローラ向け
  - 車載システム向け
  - RH850 HAV を用いたハイパーバイザ
    - 試用されたい方は声をかけて下さい.