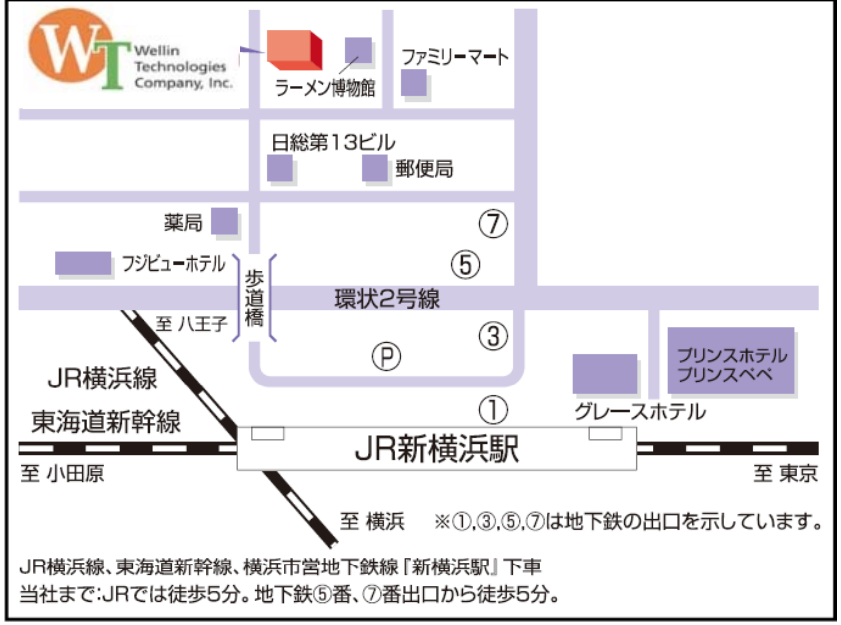
Intel VT
～仮想化を支える基礎技術～

株式会社ウェルインテクノロジー
加藤　秀一
hidekazu@wellintec.co.jp

会社概要

名称	株式会社ウェルインテクノロジー
所在地	〒222-0033 神奈川県横浜市港北区新横浜2-14-2 新横浜214　8階
設立年	2009年
役員	代表取締役社長　　橋本　竜也
従業員数	38名(2009年8月現在)
事業内容	1. ソフトウェア・ハードウェアの製品開発及び輸入販売、技術サポート、コンサルティング
	2. 組み込み用ミドルウェアの製品開発
	3. 組み込み用OS及びミドルウェアを含む移植サービス
	4. ソフトウェアの開発環境の開発
	5. 開発用プラットフォーム、製品用CPUボードシステムの設計供給(デジタル情報家電、テレコム機器、産業機器向け)

アクセス



JR横浜線、東海道新幹線、横浜市営地下鉄線『新横浜駅』下車
当社まで:JRでは徒歩5分。地下鉄⑤番、⑦番出口から徒歩5分。

自己紹介

担当分野

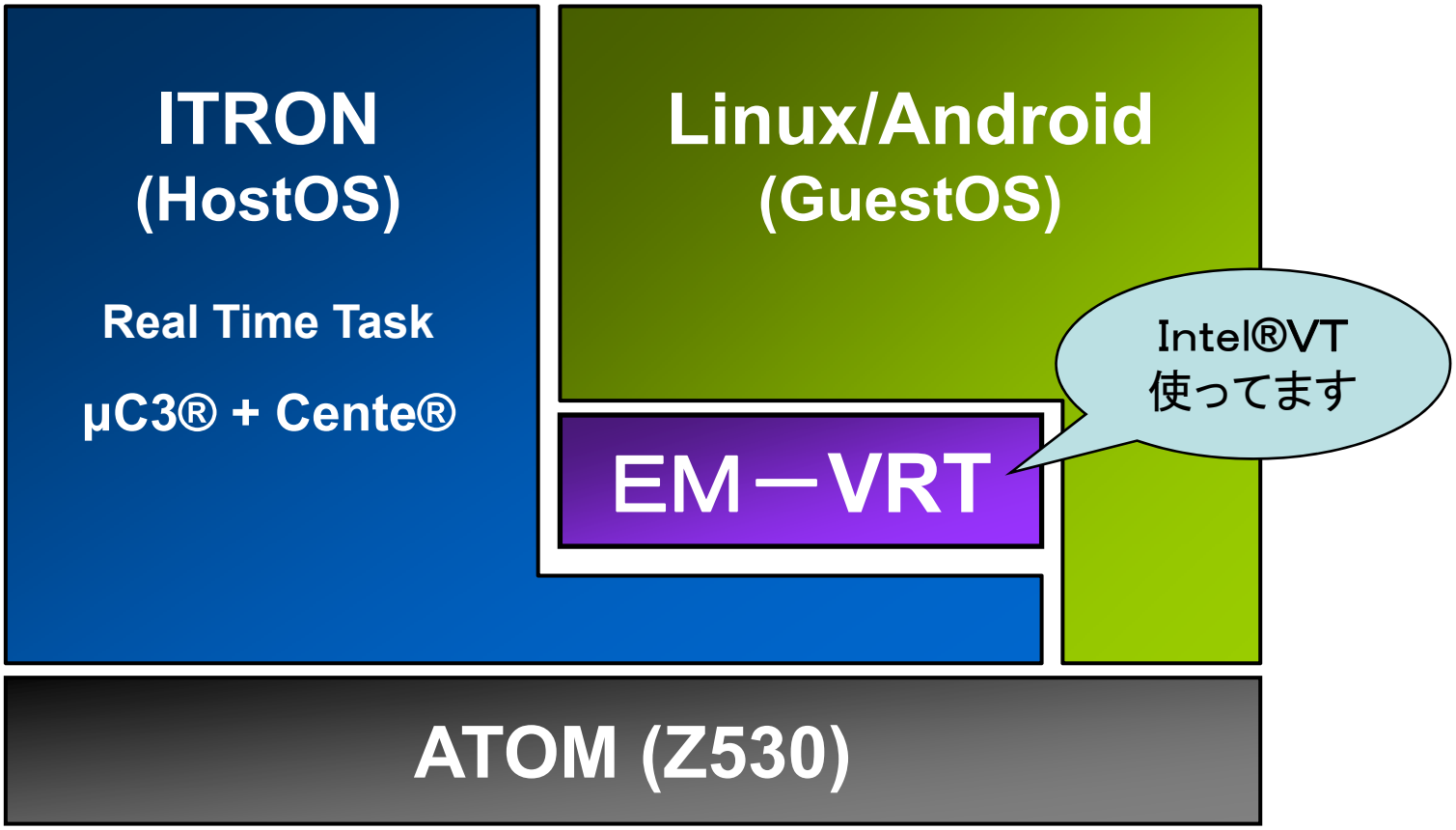
RTOSをメインとした組み込み機器の開発

現在のお仕事

組み込み向け HostOS 型仮想マシン
EM-VRT開発担当

- HostOS ：μC3（ITRON）
- GuestOS: Linux
- 対応プロセッサ：ATOM/ARM/SH

EM-VRT



Agenda

- Intel VT
- 仮想マシン処理概要
- 仮想マシン実装例

参考資料

Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3B

この資料に IntelVT の説明から仮想マシンの作成方法まで、ほぼ全てが記載されています

本資料中でアルファベット表記されている用語は参考資料の記述に基づいてます

Intel VT

- プロセッサの仮想化を支援する機能
- VT-x
  - Core2Duo/Xeon/ATOM 向け仮想化支援機能
- VT-i
  - Itanium2 向け仮想化支援機能
- VT-d/VT-c
  - I/O 仮想化支援機能

主な追加機能

- 動作モード
  - VMX root/VMX none-root
- VMCS
  - 仮想マシン用の情報領域
  - 動作条件、レジスタ情報 ...
- 命令(VMX Instruction)
  - vmxon/vmxoff/vmread/vmwrite
  - vmlaunch/vmresume/vmcall ...
- 追加レジスタ／フラグ
  - VMX 用 MSR
  - CR4.VMXE

動作モード（１／５）

- VMX root/VMX none-root が追加
  - 通常動作時と同様に RING0 - RING3 の特権モードを持つ
  - 基本的には通常動作時と変わらないが、VMX root モードに遷移しないと VT に関する追加命令の実行が不可能
  - VMX root モード以外で追加命令を実行すると例外となる

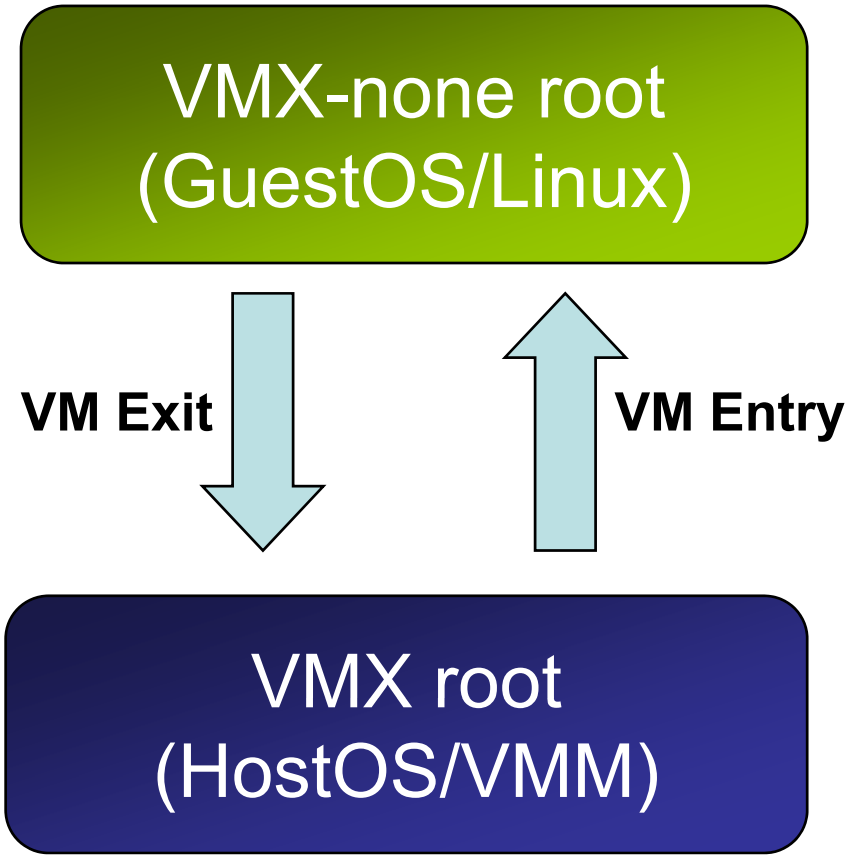
動作モード（２／５）

- VMX root
  - 仮想マシンモニタ(VMM) が動作するモード
  - vmxon 命令により通常動作モードから VMX root へ遷移
- VMX none-root
  - GuestOS(Linux) を動作させるモード
  - VMX root から vmlaunch/vmresume により VMX none-root へ遷移

動作モード（3／5）

VM Exit 発生契機

- 割り込み
- メモリアクセス違反
- I/Oアクセス違反
- その他例外
- 命令トラップ



VM Entry 発生契機

- VMM による指示
  vmlaunch
  vmresume

動作モード（４／５）

- 状態遷移
  - VM Entry
    - VMX root から VMX none-root へ状態遷移する
    - 追加命令により状態遷移が実行される (vmlaunch/vmresume)
    - VM Entry が成功した後は、GuestOS のコンテキストでプログラムが実行される。

動作モード（５／５）

- 状態遷移
  - VM Exit
    - VMX none-root から VMX root へ状態遷移する
    - GuestOS 実行時に、何らかの異常が発生した時や、システムで設定した条件が成立した時に状態遷移が行われる
    - VM Exit が実行されると、VMCS にその理由が記録される
    - VMM は VMCS から VM Exit の発生要因をチェックしてエミュレート処理を実行する
    - GuestOS の実行が継続可能であれば vmresume 命令で VMX none-rootの動作を再開する

VMCS（1／3）

- VMX none-root の動作用に用意する情報領域
- vmwrite/vmread 命令で情報の読み書きを行う
- 最大4096bytesの領域を必要とする
  - 必要なサイズは IA32_VMX_BASIC MSR の Bits44:32 に設定されているが、面倒なので最大値を常に採用

VMCS（2／3）

- 仮想マシン毎に1つ VMCS を用意する必要が有る
- 複数の VMCS を切り替えて使用する事により、複数の仮想マシンを実行する事が可能

VMCS（3／3）

- VMCS 領域を使用して HostOS/GuestOS 間のプロセッサ状態（レジスタ情報等）が切り替えられる

表. VM Entry/VM Exit 発生時のレジスタ操作（一部）

切り替え対象	切り替え対象外
ES/CS/SS/DS/FS/GS RSP/RIP/RFLAGS TR/GDTR/LDTR/IDTR CR0/CR3/CR4	RAX/RDX/RCX/RBX/RSI/RDI FPU 関連レジスタ

追加命令（1／2）

- vmxon
  - VMX root モードへの状態遷移
- vmxoff
  - VMX root モードから通常動作モードへの遷移
- vmptrld
  - 操作対象となる Current VMCS の設定

追加命令（2／2）

- vmread/vmwrite
  - Current VMCS に対する READ/WRITE 命令
- vmclear
  - VMCS領域の初期化等で実行
- vmlaunch
  - VMX none-root へ最初の状態遷移を行う
    Current VMCS で最初に仮想マシンを起動する時に実行
- vmresume
  - 再度 VMX none-root へ状態遷移を行う
    VM Exit 発生後、仮想マシンの処理を再開する時に実行

仮想マシン処理概要(起動編)

① VMX root モードへ状態遷移
- 4096bytes のメモリ領域を初期化した後、vmxon 命令を実行

② VMCS の初期化

③ GuestOS のプログラムをメモリ上にロード

④ VMX none-root へ遷移して GuestOS を実行
- ②で設定された EIP のアドレスから実行開始

仮想マシン処理概要（GuestOS 実行編）

⑤ VM Exit 発生により VMX root へ状態遷移

⑥ VMCS から VM Exit の発生要因を判定して各種エミュレーション処理を実行

- Hypervisor Call や仮想デバイスの処理等もこのタイミングで実行される

⑦ エミュレーションの実行結果をメモリや VMCS に設定

⑧ vmresume により仮想マシンの処理を再開

以降は (5)-(8) の繰り返し

仮想マシン実装例(起動編)

① VMX root モードへ状態遷移

– 4096bytes のメモリ領域を初期化した後、vmxon 命令を実行

```
mov     $IA32_VMX_BASIC, %ecx    ; 0480h
rdmsr
mov     %eax, vmx_resion         ; VMCS revision
pushl   $0
pushl   $vmx_resion
vmxon   (%esp)
lea     8(%esp), %esp

;   EFLAGS.CF = 0 なら成功
```

仮想マシン実装例（起動編）

② VMCS の初期化

```
mov     $IA32_VMX_BASIC, %ecx     ; 0480h
rdmsr
mov     %eax, vmcs_region         ; VMCS revision
pushl    $0
pushl    $vmcs_region
vmclear   (%esp)
vmptrld   (%esp)                  ; Current VMCS
add       $8, %esp
```

仮想マシン実装例(起動編)

② VMCS の初期化(パラメータ設定)

```
; GuestOS の RIP 設定
mov       $VMCS_GUEST_RIP, %eax ; 681Eh
pushl     $0
push      $GUEST_START                ; Start Addr.
vmwrite   (%esp), %eax
add       $8, %esp

; 以降も同様に各種パラメータを設定する
```

仮想マシン実装例（起動編）

③ GuestOS のプログラムをメモリ上にロード

省　略

仮想マシン実装例(起動編)

④ VMX none-root へ遷移して GuestOS を実行

```
; HostOSのレジスタ退避
pushf
pusha
fxsave    host_fpu_regs
; HostOSへの復帰アドレス設定
mov       $VMCS_HOST_RIP, %eax    ; 6C16h
push      $0
push      $vm_exit
vmwrite    %(esp), %eax
; 同様に HostOS の RSP も設定
```

仮想マシン実装例(起動編)

```
; GuestOS のレジスタ設定
mov     guest_regs_eax, %eax
mov     guest_regs_edx, %edx
....
mov     guest_regs_edi, %edi
finit

; GuestOS 起動
vmlaunch

; vmlaunch以降の命令は vmlaunch が失敗した時に実行される
```

仮想マシン実装例（GuestOS 実行編）

⑤ VM-Exit 発生により VMX root へ状態遷移

```
vm_exit:
  ; GuestOS のレジスタ状態を退避
  mov     %eax, guest_regs_eax
  mov     %edx, guest_regs_edx
  ....
  mov     %edi, guest_regs_edi
  fxsave  guest_fpu_regs
  ; HostOS のレジスタ状態を復元
  fxrstor   host_fpu_regs
  popa
  popf
```

仮想マシン実装例（GuestOS 実行編）

⑥ VMCS から VM-Exit の発生要因を判定して各種エミュレーション処理を実行

```
; VM Exit 要因を読み出す
mov       $VMCS_EXIT_REASON, %edx          ; 4402h
vmread   %edx, %eax
```

主に読み出すパラメータ
Exit Reason
Exit Qualification
VM-exit interruption information
VM-exit interruption error code
IDT-vectoring information field
IDT-vectoring error code
VM-exit instruction length
VM-exit instruction information

仮想マシン実装例（GuestOS 実行編）

⑦ エミュレーションの実行結果をメモリや VMCS に設定

例）特定の命令のエミュレーションを行った場合などは、guest_regs_xxx に適切な値を設定した後、VM-exit instruction length から読み取った値を GuestOS の RIP に加算

仮想マシン実装例（GuestOS 実行編）

⑧ vmresume により仮想マシンの処理を再開

```
; HostOS のレジスタ状態を退避
pushf
pusha
fxsave    host_fpu_regs
; VM Exit 後の実行アドレスとスタックポインタを設定
mov       $VMCS_HOST_RIP, %eax          ; 6C16h
push      $0
push      $vm_exit
vmwrite   %(esp), %eax
; 同様に RSP についても設定する
```

仮想マシン実装例（GuestOS 実行編）

```
; GuestOS のレジスタ状態を復元
fxrstor  guest_fpu_regs
mov      guest_regs_eax, %eax
mov      guest_regs_edx, %edx
....
mov      guest_regs_edi, %edi

; GuestOS を再開
vmresume

; 以降の命令は vmresume が失敗した時に実行される
; vmresume が成功すると GuestOS が中断点から再開する
```

作った感想

- 汎用レジスタが切り替え対象外なのが意外だった
- VMCSの設定が大変
  - 最初は起動可能な設定が解らない
  - 設定方法が理解できると、思い通りの仮想化を実現可能
- Intelプロセッサに対する深い知識が必要
  - Intelプロセッサへの理解が深まると同時に、実装の自由度が増した

ありがとうございました