Virtual Machine Monitor on ARM processor

2013년 6월 7일

IAMROOT.ORG

김강호, 전승협, 고광원
khk@etri.re.kr, shjeon00@etri.re.kr, kwangwon.koh@etri.re.kr,
ETRI

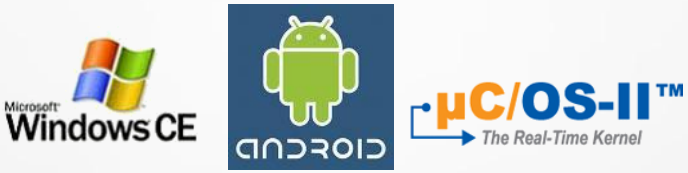
CONTENTS

1. ViMo Introduction
2. ARM Architecture Overview
3. ARM Virtualization
4. ViMo Demo

ViMo Introduction

ViMo Project

- Light weight virtual machine monitor
  - Multiple OSes on ARM-based platform
- Mar. 1st, 2009 ~ Feb. 29th, 2012



Where to start

- From scratch
  - ViMo

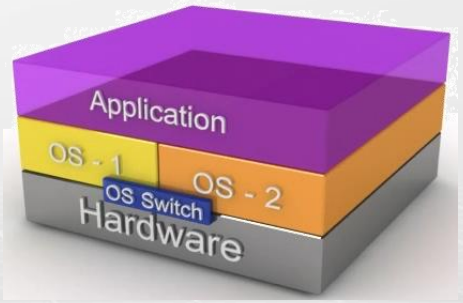
- Adaption
  - Xen-arm (para-virtualization)
  - MVP (para-virtualization, source code not available)
  - VLX (beyond our budget, PPC, para-virtualization)
  - QEMU virtualizer (hosted type)

Output

- Type-I hypervisor
  - ViMo Express
  - ViMo

- Type-II hypervisor
  - ViMo-t2

- OS switch
  - ViMo switch



ARM Architecture Overview

ARM System

- CPU
- VMSA(Virtual Memory System Architecture)
- I/O Device Controller

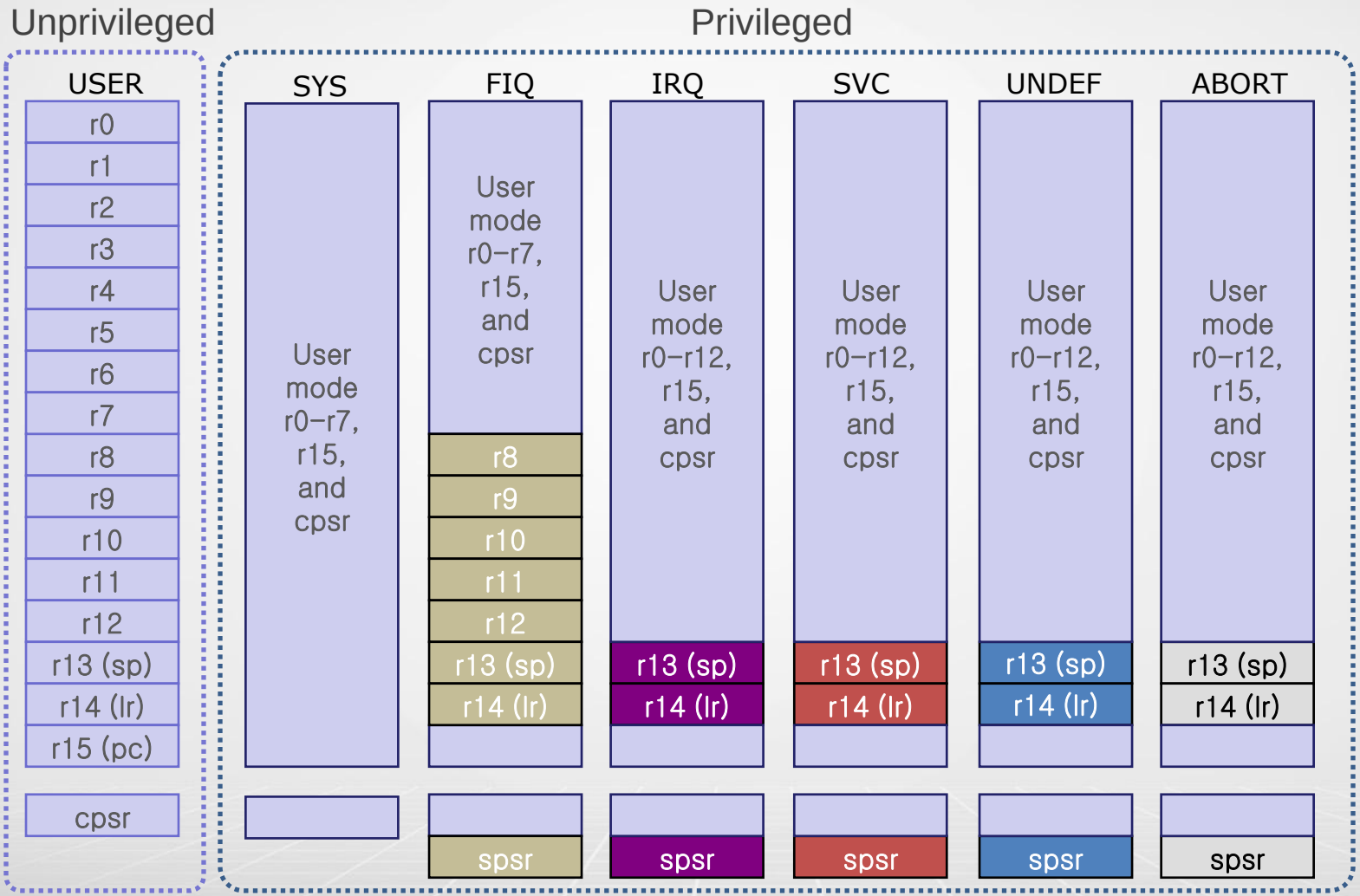
CPU (1)

- 148 instructions

Type	Instruction
Branch, Branch with link, Branch Exchange	B, BL, BX
Data processing	ADD, ADC, SUB, SBC, RSB, RSC, AND, ORR, BIC, MOV, MVN, CMP, CMN, TST, TEQ
Multiply	MUL, MLA, SMULL, SMLAL, SMULL, UMLAL
Load/Store	LDR, LDRB, LDRBT, LDRH, LDRSB, LDRSH, LDRT, STR, STRB, STRBT, STRH, STRT
Load/Store Multiple	LDM, STM
Swap	SWP, SWPB
Software Interrupt	SWI
PSR transfer	MRS, MSR
Coprocessor	MRC, MCR, LDC, STC
Undefined	

CPU (2)

- CPU mode



VMSA (1)

- Page descriptor format
  - 1st level
    - Super-section: 16MB page frame
    - Section: 1MB page frame
    - Coarse: 1KB page table
  - 2nd level
    - Large: 64KB page frame
    - Extended small: 4KB page frame
- Access Permission
  - No access, Read, write
- Domain

VMSA (2)

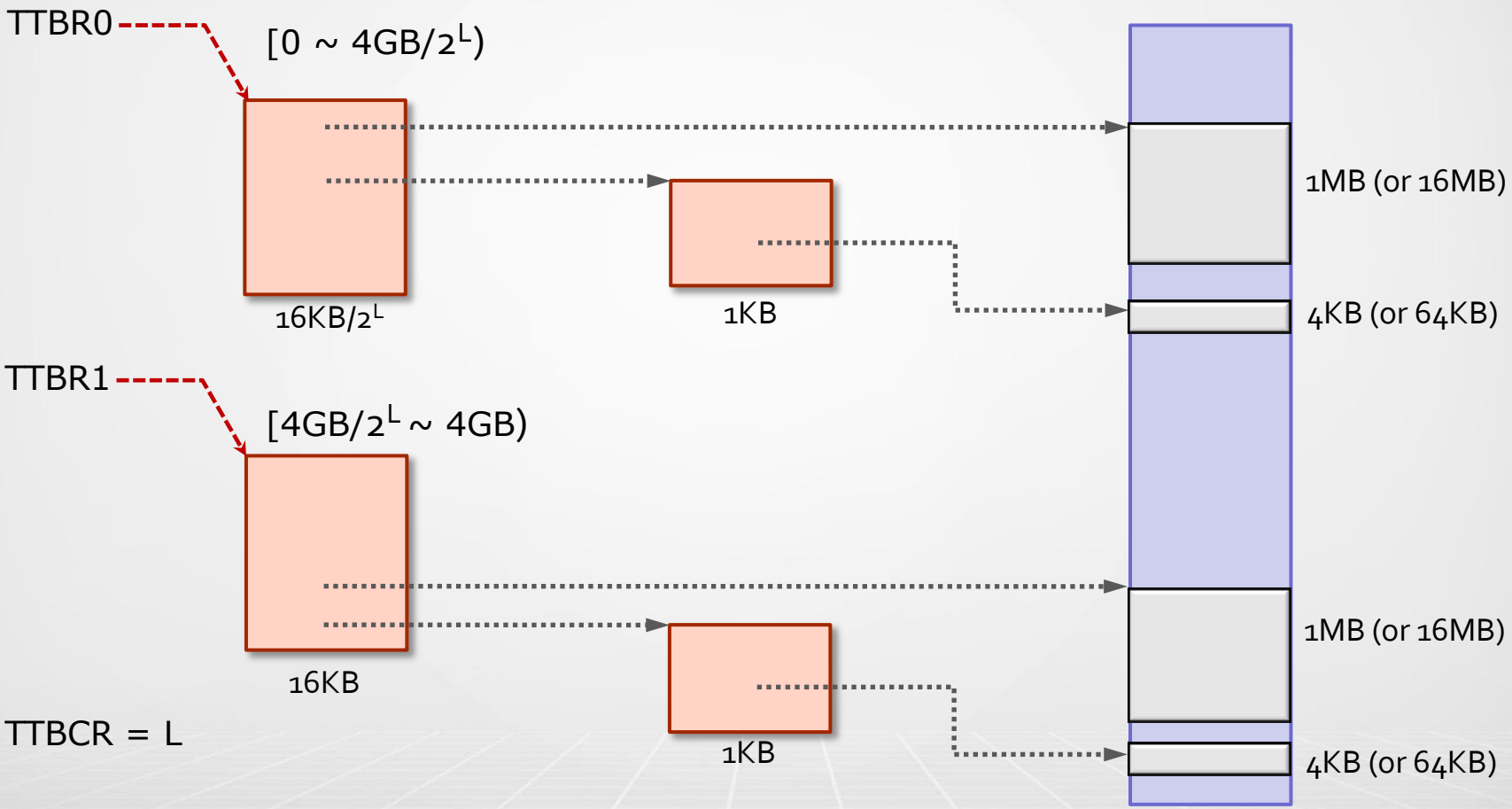
- Domain
  - Client, Manager, No Access types
  - 1MB unit
  - 16 domains

Table B4-2 Domain Access Values

Value	Access types	Description
0b00	No access	Any access generates a domain fault
0b01	Client	Accesses are checked against the access permission bits in the TLB entry
0b10	Reserved	Using this value has UNPREDICTABLE results
0b11	Manager	Accesses are not checked against the access permission bits in the TLB entry, so a permission fault cannot be generated

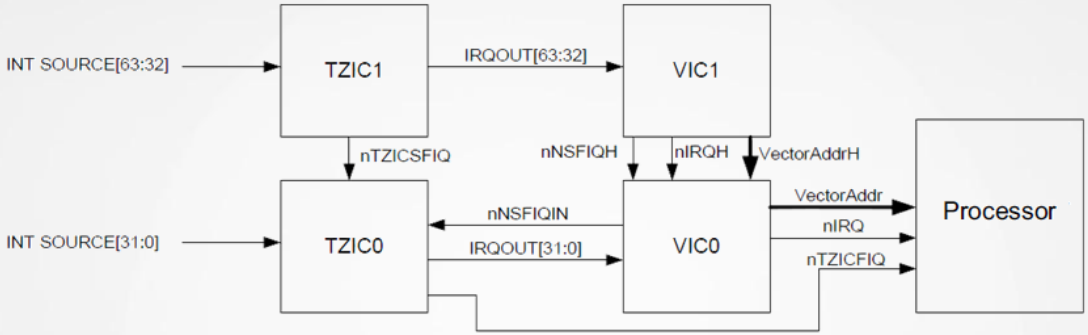
VMSA (3)

- ARM page table example



Interrupt Controller

- Vectored Interrupt Controller(VIC)



4 REGISTER DESCRIPTION

Register	Address	R/W	Description	Reset Value
VIC0IRQSTATUS	0xE400_0000	R	IRQ Status Register	0x00000000
VIC0FIQSTATUS	0xE400_0004	R	FIQ Status Register	0x00000000
VIC0RAWINTR	0xE400_0008	R	Raw Interrupt Status Register	-
VIC0INTSELECT	0xE400_000C	R/W	Interrupt Select Register	0x00000000
VIC0INTENABLE	0xE400_0010	R/W	Interrupt Enable Register	0x00000000
VIC0INTENCLEAR	0xE400_0014	W	Interrupt Enable Clear Register	-
VIC0SOFTINT	0xE400_0018	R/W	Software Interrupt Register	0x00000000
VIC0SOFTINTCLEAR	0xE400_001C	W	Software Interrupt Clear Register	-
VIC0PROTECTION	0xE400_0020	R/W	Protection Enable Register	0x0
VIC0SWPRIORITYMASK	0xE400_0024	R/W	Software Priority Mask Register	0xFFFF
VIC0PRIORITYDAISY	0xE400_0028	R/W	Vector Priority Register for Daisy Chain	0xF
VIC0VECTADDR0	0xE400_0100	R/W	Vector Address 0 Register	0x00000000
VIC0VECTADDR1	0xE400_0104	R/W	Vector Address 1 Register	0x00000000
VIC0VECTADDR2	0xE400_0108	R/W	Vector Address 2 Register	0x00000000
VIC0VECTADDR3	0xE400_010C	R/W	Vector Address 3 Register	0x00000000
VIC0VECTADDR4	0xE400_0110	R/W	Vector Address 4 Register	0x00000000
VIC0VECTADDR5	0xE400_0114	R/W	Vector Address 5 Register	0x00000000
VIC0VECTADDR6	0xE400_0118	R/W	Vector Address 6 Register	0x00000000
VIC0VECTADDR7	0xE400_011C	R/W	Vector Address 7 Register	0x00000000
VIC0VECTADDR8	0xE400_0120	R/W	Vector Address 8 Register	0x00000000

ARM Virtualization

Virtualization category

- Guest OS modification
  - **Full virtualization**
  - **Pre-virtualization**
  - Para-virtualization
- H/W assisted
  - **Pure software**
  - H/W assisted
- Virtual Machine Type
  - **Type-I**
    - **Bare-metal, native**
  - **Type-II**
    - **hosted**
- Virtualization level
  - **System virtualization**
  - OS virtualization
  - Application virtualization
- Application area
  - Server/desktop
  - mobile/embedded
  - Network
  - Storage

System Virtualization

- CPU virtualization
- Memory virtualization
- I/O device virtualization

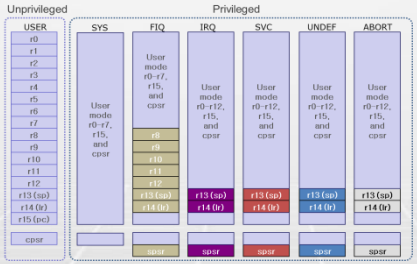
- Underlying issues
  - Resolving address space compression problem
  - Intercepting exceptions

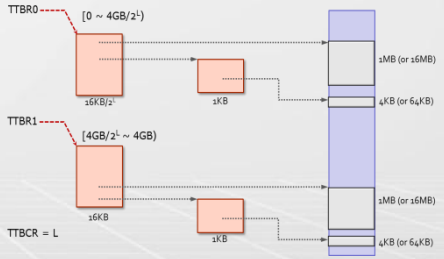
ARM System Virtualization

- How to virtualize CPU in ARM system
- How to virtualize memory in ARM system
- How to virtualize IO controllers in ARM system

- How to resolve the address space problem in ARM
- How to intercept interrupts & exceptions in ARM

Type	Instruction
Branch, Branch with link, Branch Exchange	B, BL, BX
Data processing	ADD, ADC, SUB, SBC, RSB, RSC, AND, ORR, BIC, MOV, MVN, CMP, CMN, TST, TEQ
Multiply	MUL, MLA, SMULL, SMLAL, SMULL, UMLAL
Load/Store	LDR, LDRB, LDRBT, LDRH, LDRSB, LDRSH, LDRT, STR, STRB, STRBT, STRH, STRT
Load/Store Multiple	LDM, STM
Swap	SWP, SWPB
Software Interrupt	SWI
PSR transfer	MRS, MSR
Coprocessor	MRC, MCR, LDC, STC
Undefined	



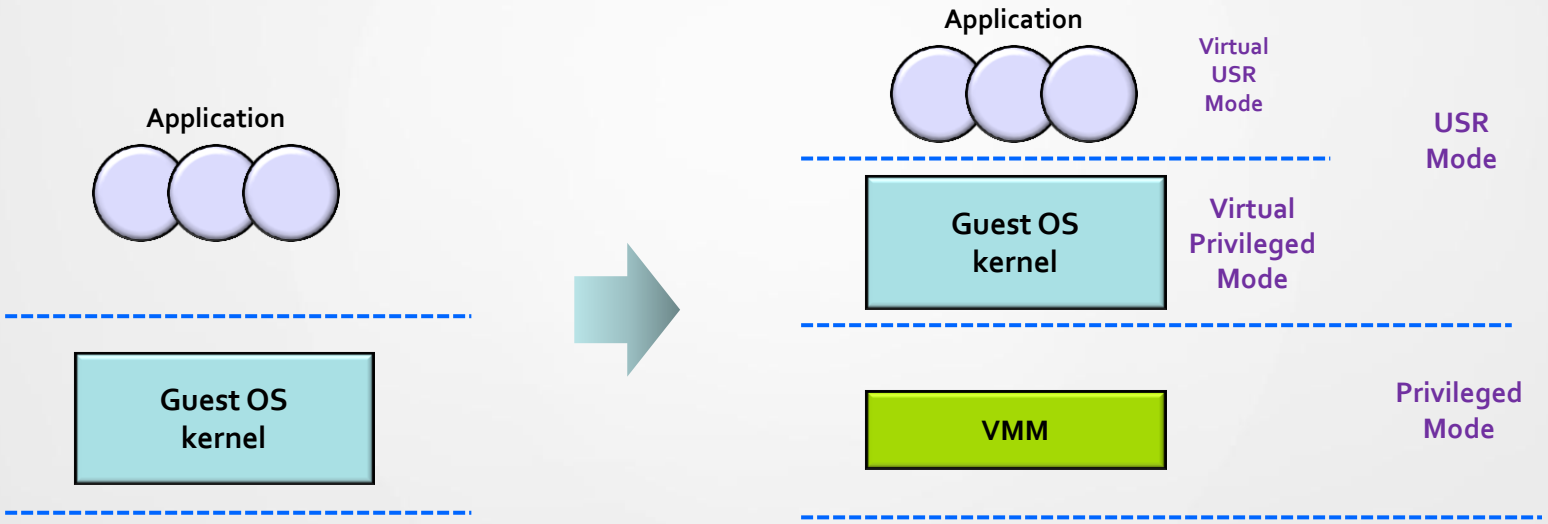


Classification of instructions

- Privileged instructions
  - Instructions that traps if the processor is in user mode and do not trap if it is in system mode
- Control sensitive instructions
  - Instructions that attempt to change the configuration of resources in the system
- Behavior sensitive instructions
  - Instructions whose behavior or result depends on the configuration of resources
- Critical instructions
  - Sensitive but **not privileged** instructions

CPU Virtualization (1)

- Deprivileging
  - Put the guest OS kernel in the unprivileged mode



CPU Virtualization (2)

- Privileged instructions
  - UNDEF exception
  - Eg) MCR, MRC
- **Critical instructions**
  - No exception
  - No or partial effect in User mode
  - Eg) CPS, MSR, MRS, STM, LDM, MOVS

CPS

Exceptions

None.

Operation

```
if InAPrivilegedMode() then
    if imod[1] == 1 then
        if A == 1 then CPSR[8] = imod[0]
        if I == 1 then CPSR[7] = imod[0]
        if F == 1 then CPSR[6] = imod[0]
    /* else no change to the mask */
    if mmod == 1 then
        CPSR[4:0] = mode
```

Notes

User mode CPS has no effect in User mode.

CPU Virtualization (3)

- Critical instruction
  - E.g) MOVS

Exceptions

None.

Operation

```
if ConditionPassed(cond) then
    Rd = shifter_operand
    if S == 1 and Rd == R15 then
        if CurrentModeHasSPSR() then
            CPSR = SPSR
        else UNPREDICTABLE
    else if S == 1 then
        N Flag = Rd[31]
        Z Flag = if Rd == 0 then 1 else 0
        C Flag = shifter_carry_out
        V Flag = unaffected
```

CPU Virtualization (4)

- Critical instruction
  - E.g) MSR

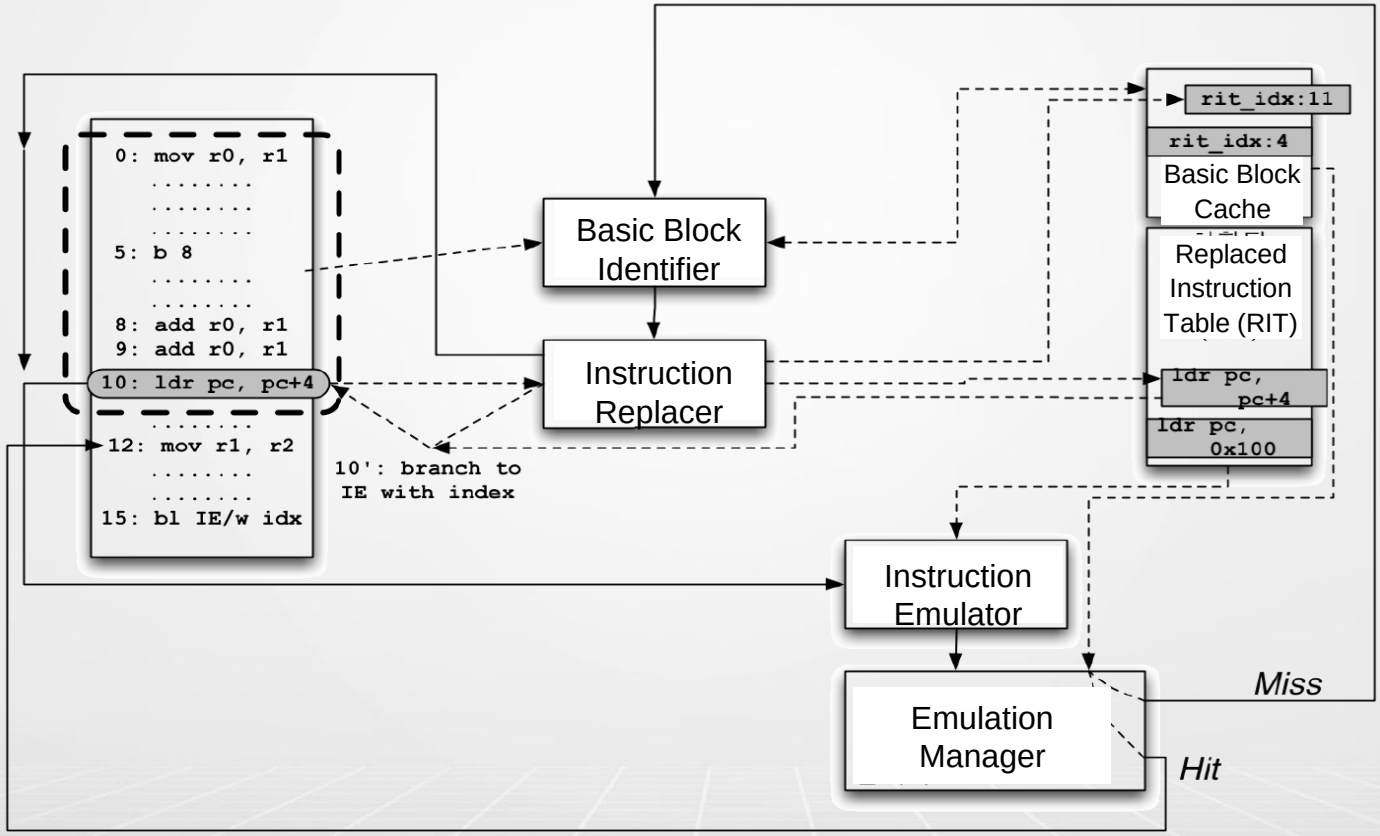
Exceptions

None.

```
if ConditionPassed(cond) then
    if opcode[25] == 1 then
        operand = 8_bit_immediate Rotate_Right (rotate_imm * 2)
    else
        operand = Rm
    if (operand AND UnallocMask) !=0 then
        UNPREDICTABLE              /* Attempt to set reserved bits */
    byte_mask = (if field_mask[0] == 1 then 0x000000FF else 0x00000000) OR
                (if field_mask[1] == 1 then 0x0000FF00 else 0x00000000) OR
                (if field_mask[2] == 1 then 0x00FF0000 else 0x00000000) OR
                (if field_mask[3] == 1 then 0xFF000000 else 0x00000000)
    if R == 0 then
        if InAPrivilegedMode() then
            if (operand AND StateMask) != 0 then
                UNPREDICTABLE      /* Attempt to set non-ARM execution state */
            else
                mask = byte_mask AND (UserMask OR PrivMask)
        else
            mask = byte_mask AND UserMask
        CPSR = (CPSR AND NOT mask) OR (operand AND mask)
    else  /* R == 1 */
        if CurrentModeHasSPSR() then
            mask = byte_mask AND (UserMask OR PrivMask OR StateMask)
            SPSR = (SPSR AND NOT mask) OR (operand AND mask)
        else
            UNPREDICTABLE
```

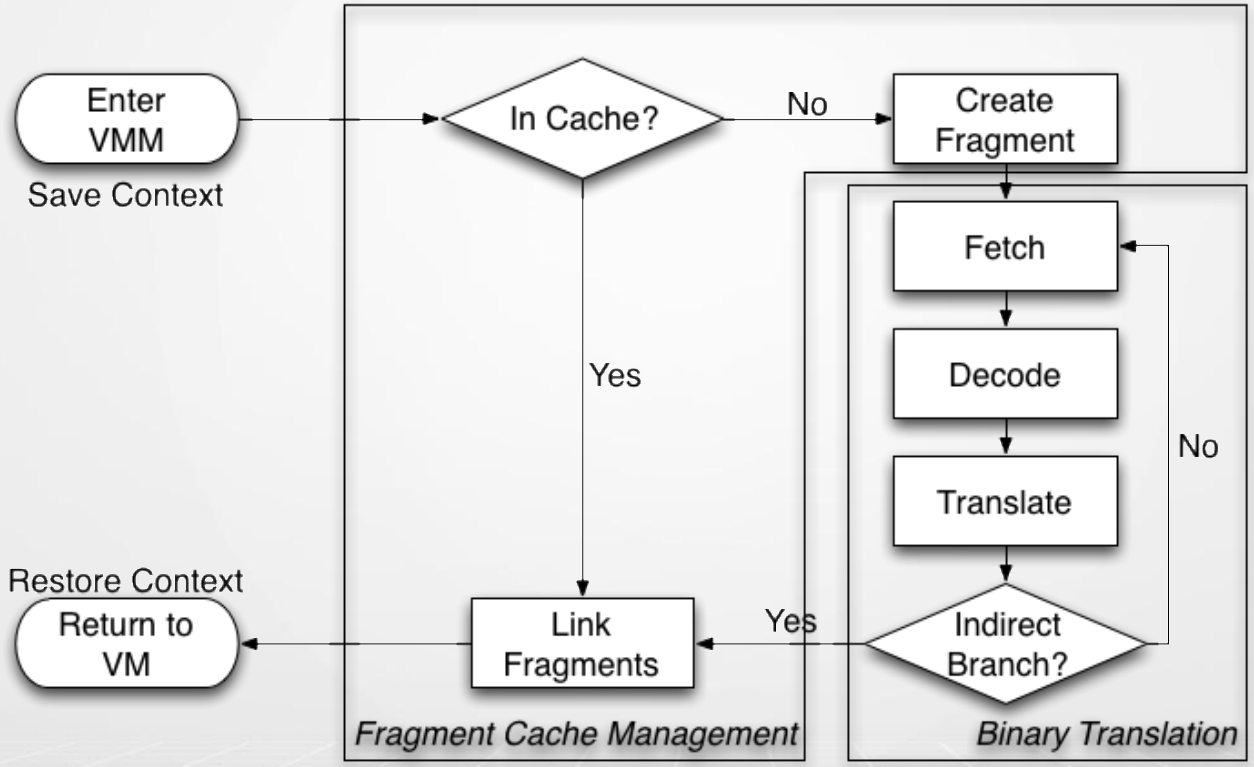
CPU Virtualization (5)

- Scanning & Patching at runtime
  - Critical instructions



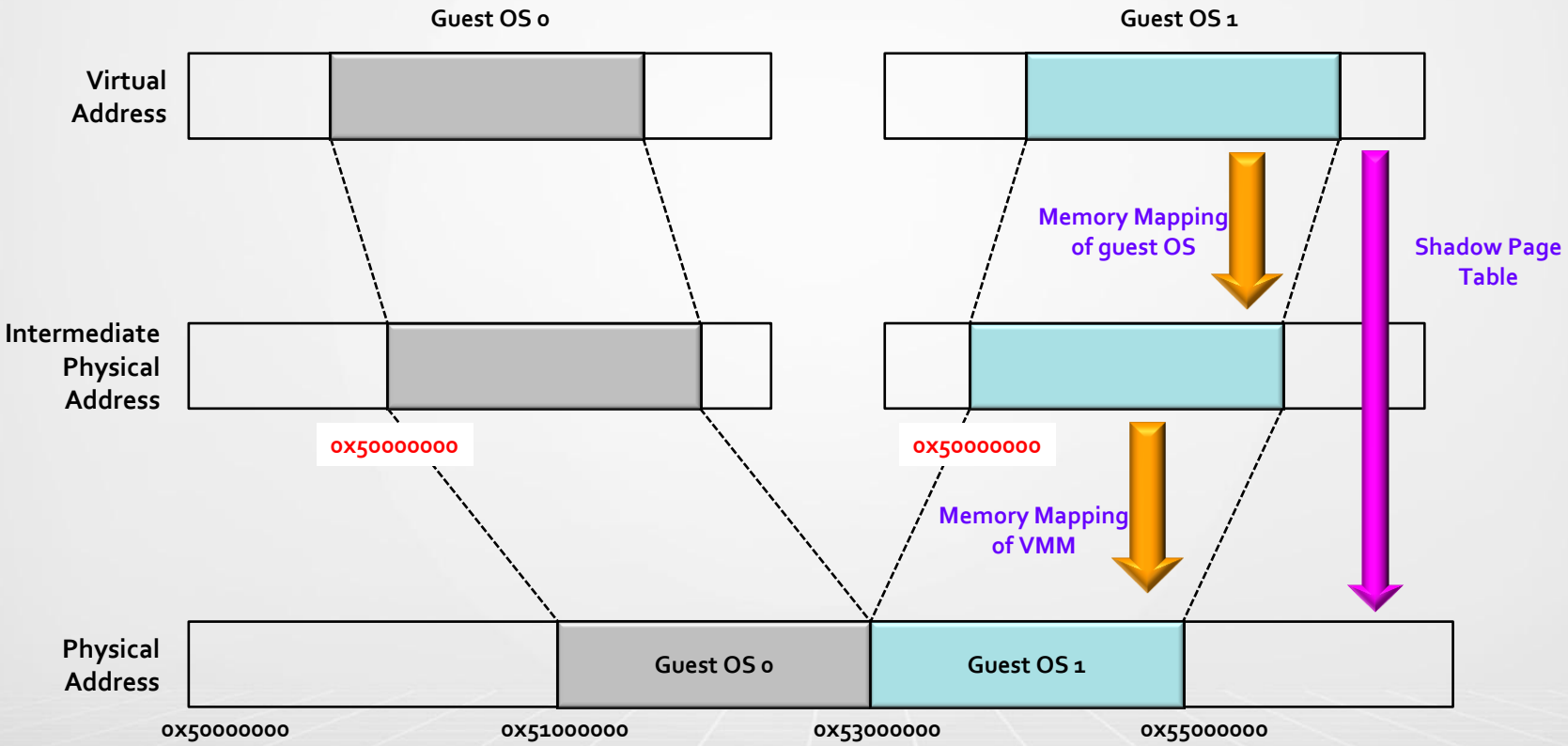
CPU Virtualization (6)

- Dynamic Binary Translation
  - Code cache management is critical



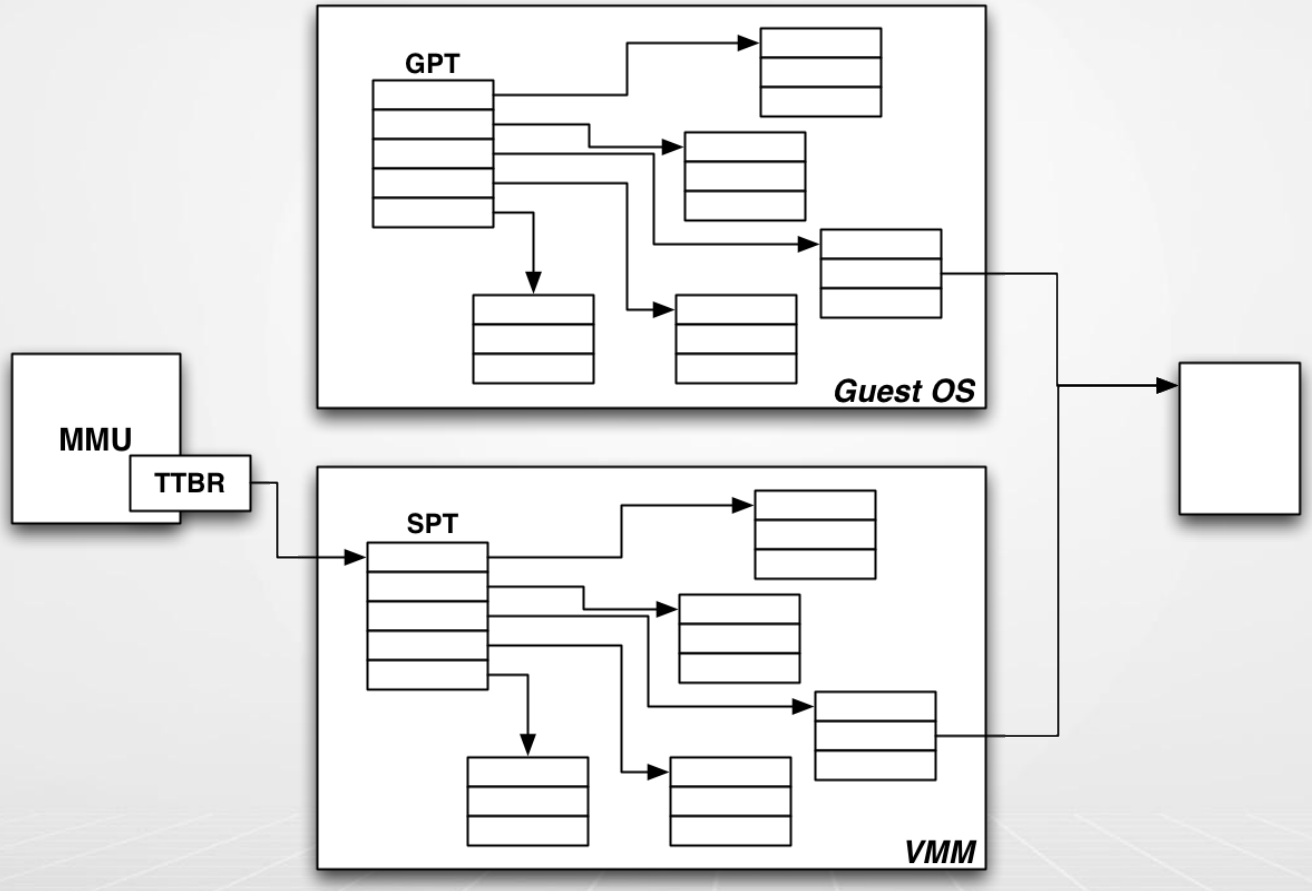
Memory Virtualization (1)

- Restricting memory access to the area where VMM allows to use



Memory Virtualization (2)

- Shadow page table



Memory Virtualization (3)

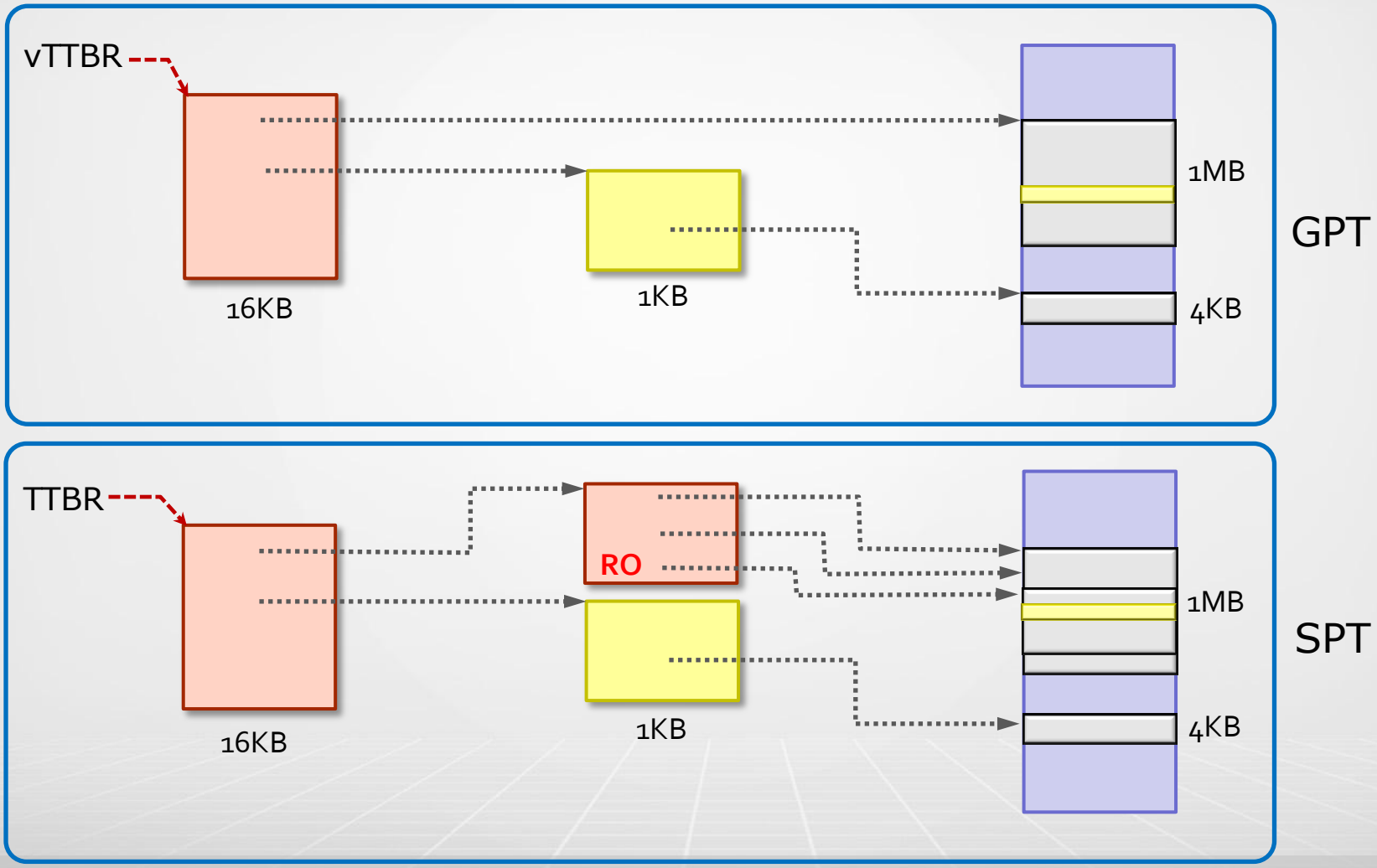
- Intercepting MMU control instructions
  - MCR instructions
    - MCR p15, …
  - User mode execution → UNDEF exception
- Intercepting page table entry update operations
  - STR, STM instructions
  - Read-only page marking → Data Abort exception
- Intercepting TLB operations
  - MCR instructions
    - MCR p15, 0, <rd>, c8, c5, 0 ;
  - User mode execution → UNDEF exception

Memory Virtualization (4)

- Intercept "Registering the page table to MMU" instruction
  - Constructing a shadow page table
    - Guest Page Table Copy
    - Page table read-only marking
    - Access Permission change
    - Exception vector table insertion
    - Data cache clean & TLB flush
- Intercept PTE/PDE update
  - Updating corresponding entry in the shadow page table
  - Page table read-only marking & AP update
  - Data cache clean & TLB flush

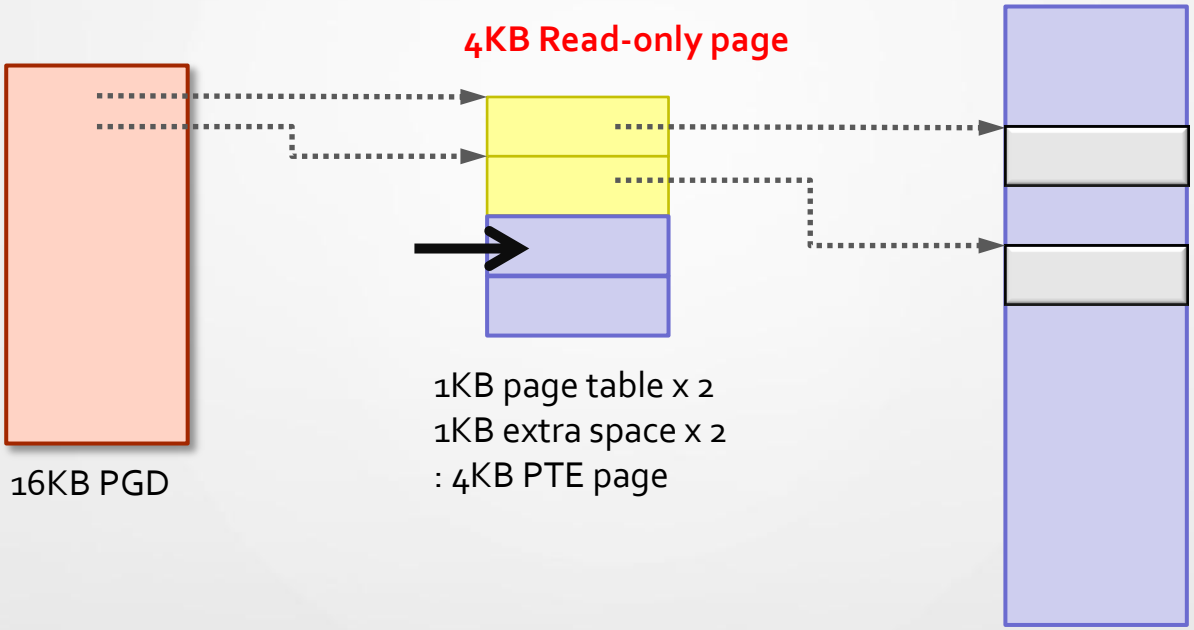
Memory Virtualization (5)

- Shadow page table: read-only marking



Memory Virtualization (6)

- False page table update detection
  - Lower 2KB in 4KB page table
  - Instruction emulation



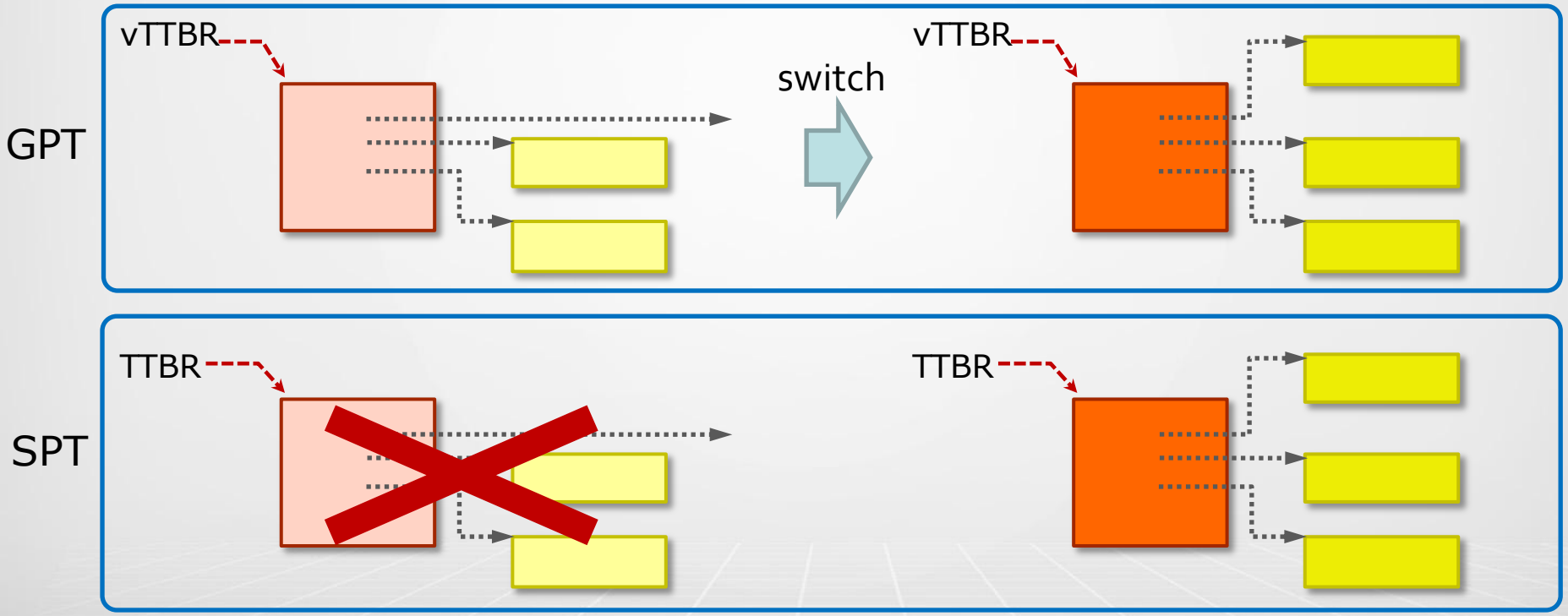
Memory Virtualization (7)

IT R&D Global Leader ETRI

- AP change
  - Kernel domain
    - SRW_URO if the page frame is for page table
    - S(P_AP)_U(U_AP) → SRW_U(P_AP), otherwise
  - User domain
    - No change
  - I/O domain
    - SRW_URW, unconditionally
- Domain Access Type update
  - Virtual Kernel: CCC----------C
  - Virtual User: NCN----------C
  - VMM: MMM----------M

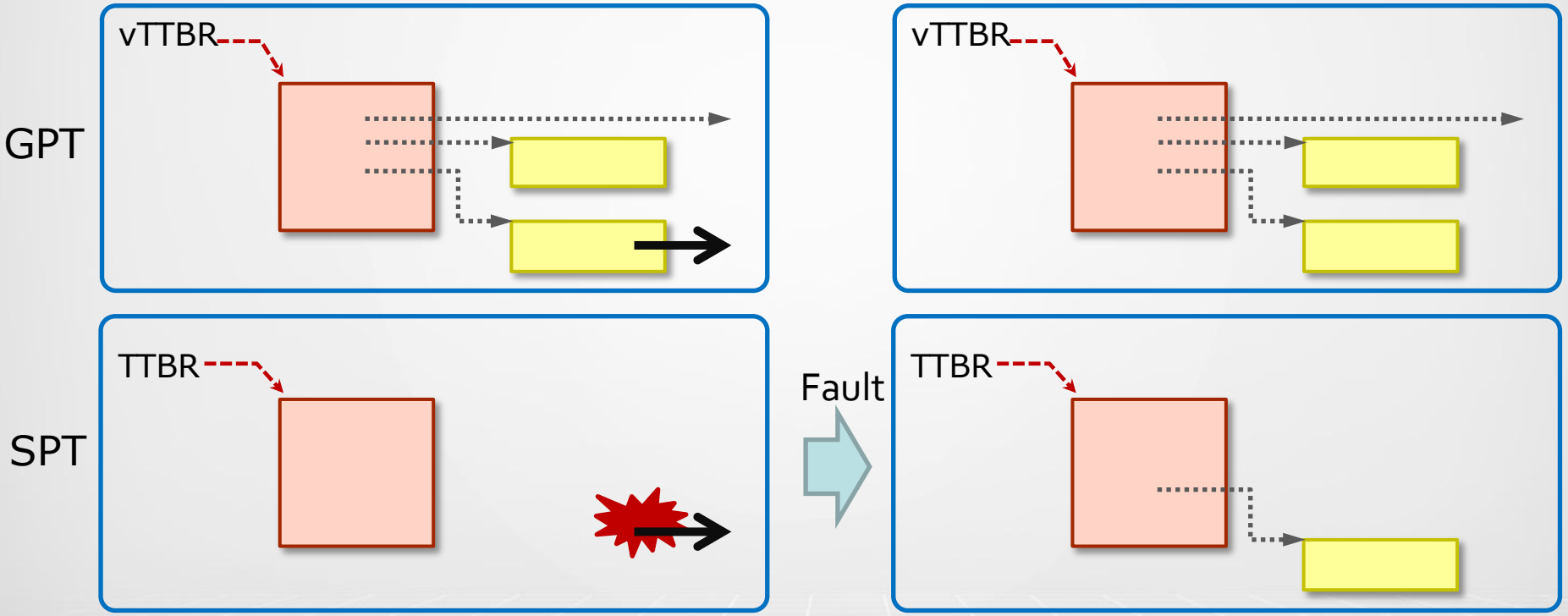
SPT Cache (1)

- Process context switch
  - Brute force
    - Destroy the previous SPT before constructing a new one



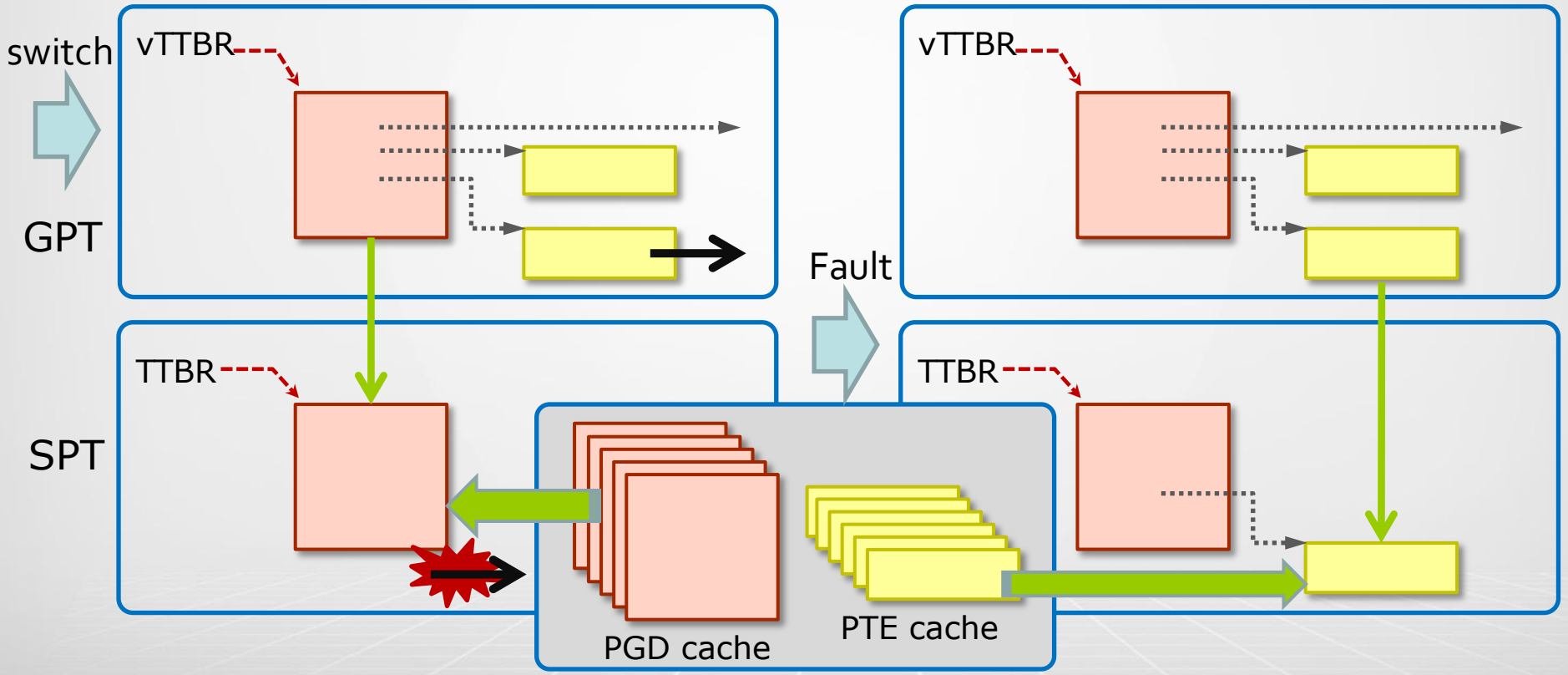
SPT Cache (2)

- Process context switch
  - Fault driven SPT construction



SPT Cache (3)

- SPT caching
  - Reuse SPT pages
  - PGD cache (16KB), PTE cache (1KB)

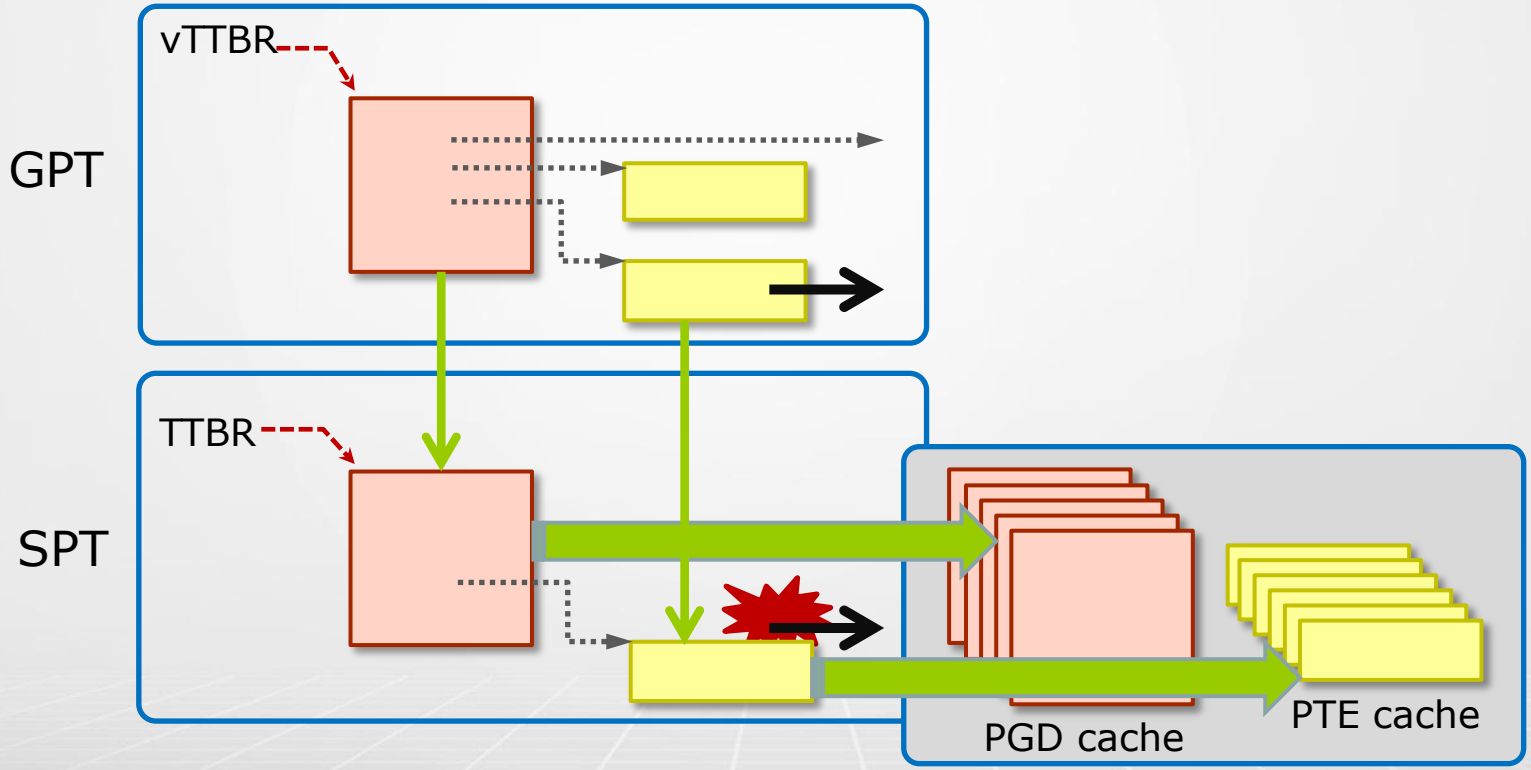


SPT Cache Issues (1)

- Caching time
  - PGD: TTBR change
  - PTE: Page fault (Abort exception)
- Cache consistency
  - PGD: PGD update
  - PTE: N/A
- Invalidating cache
  - PGD: N/A
  - PTE: PTE update
- Cache replacement policy
  - LRU
- Read-only marking pages
  - Only for the cached page tables

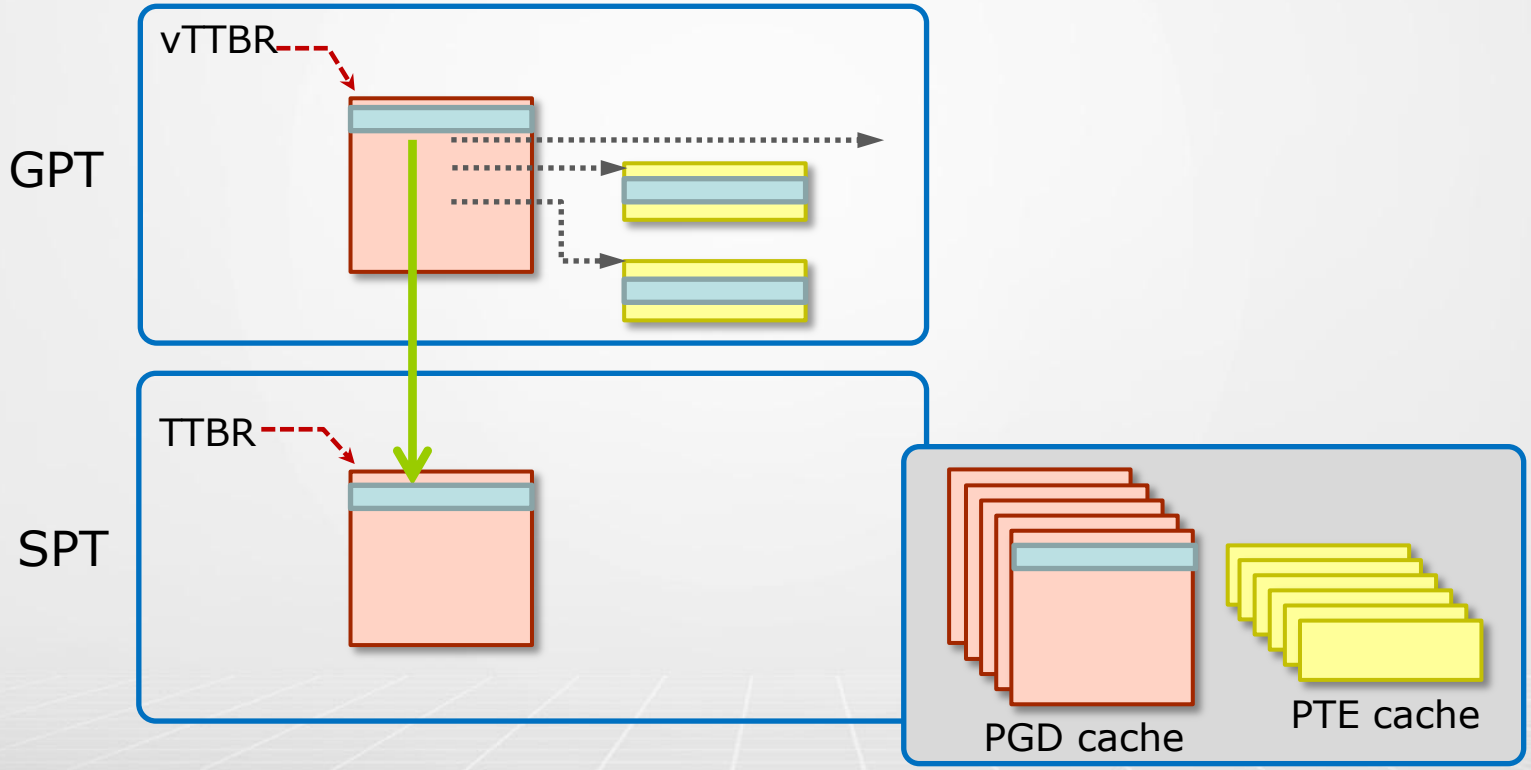
SPT Cache Issues (2)

- Caching time
  - PGD: TTBR change
  - PTE: Page fault (Abort exception)



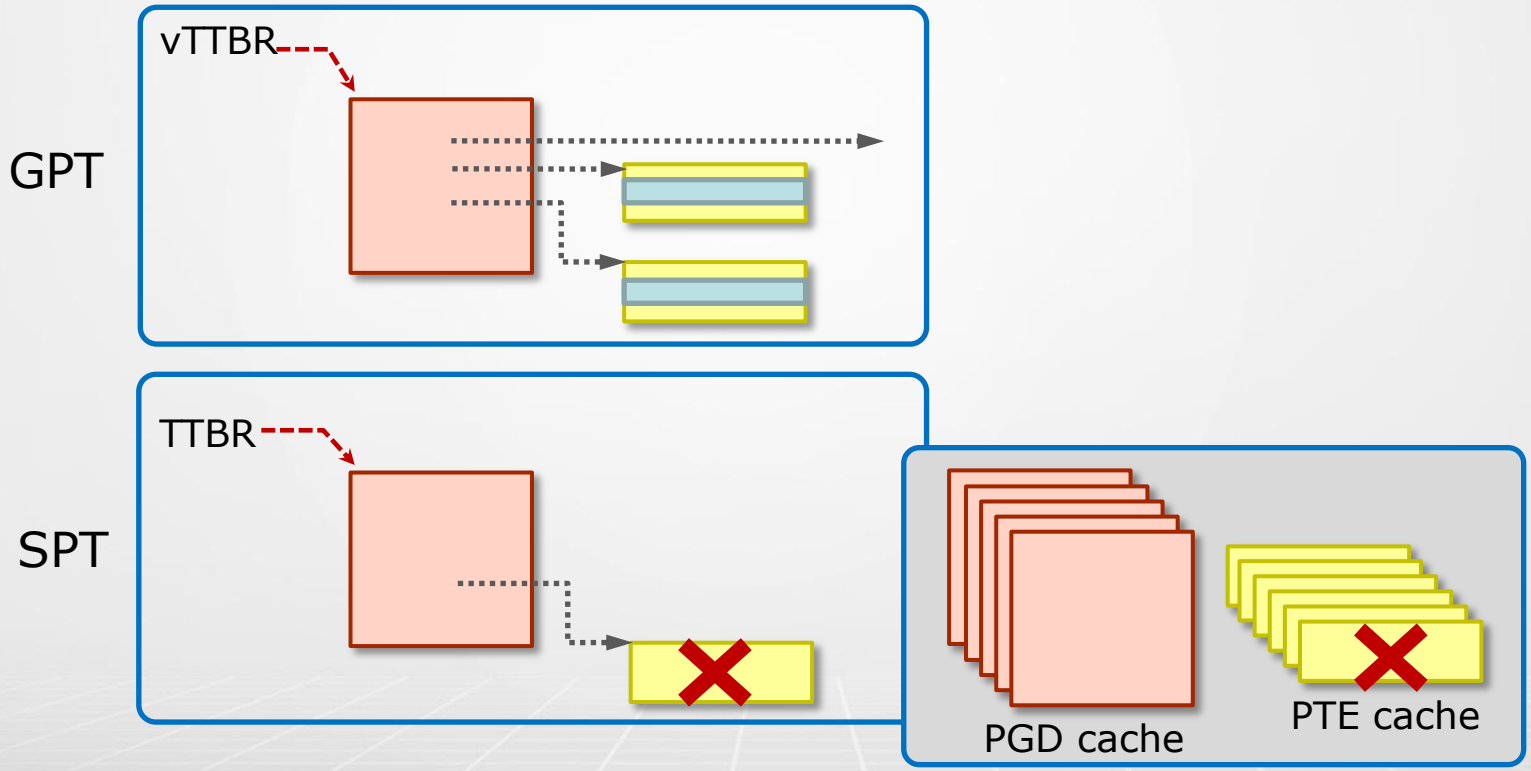
SPT Cache Issues (3)

- Cache consistency
  - PGD: PGD update
  - PTE: N/A



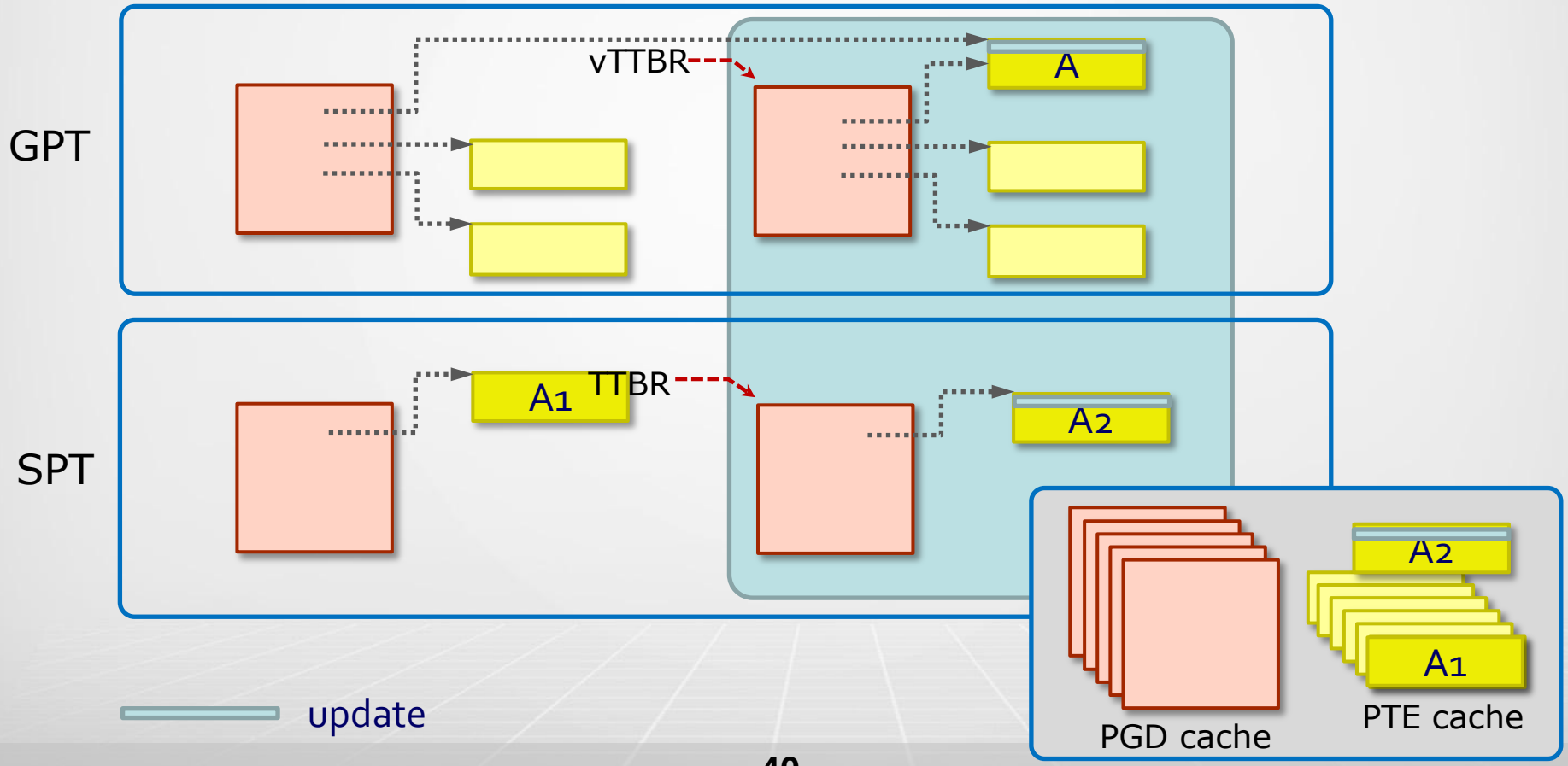
SPT Cache Issues (4)

- Invalidating cache
  - PGD: N/A
  - PTE: PTE update



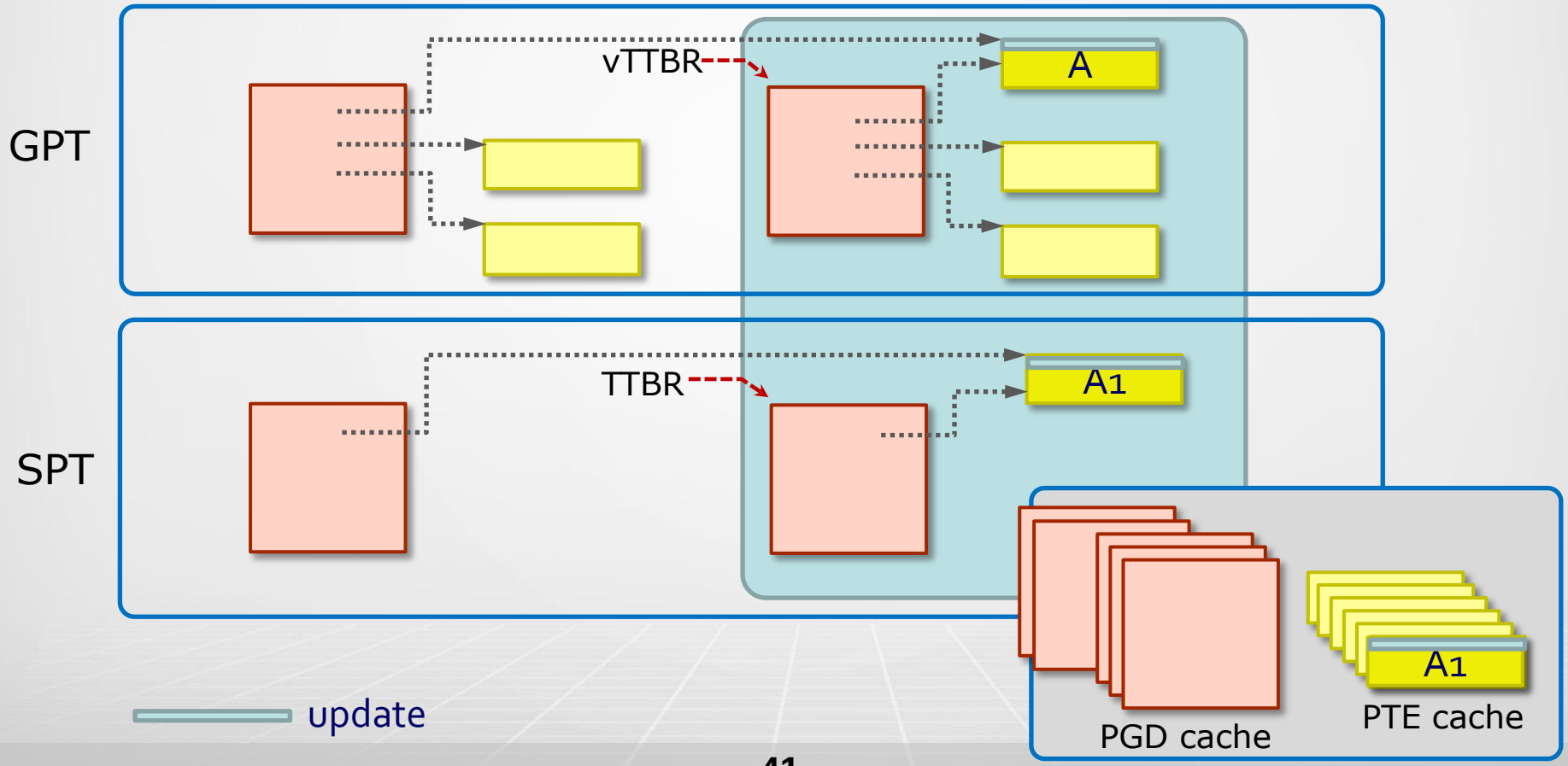
SPT Cache Issues (5)

- Shared page table
  - Cache coherency problem



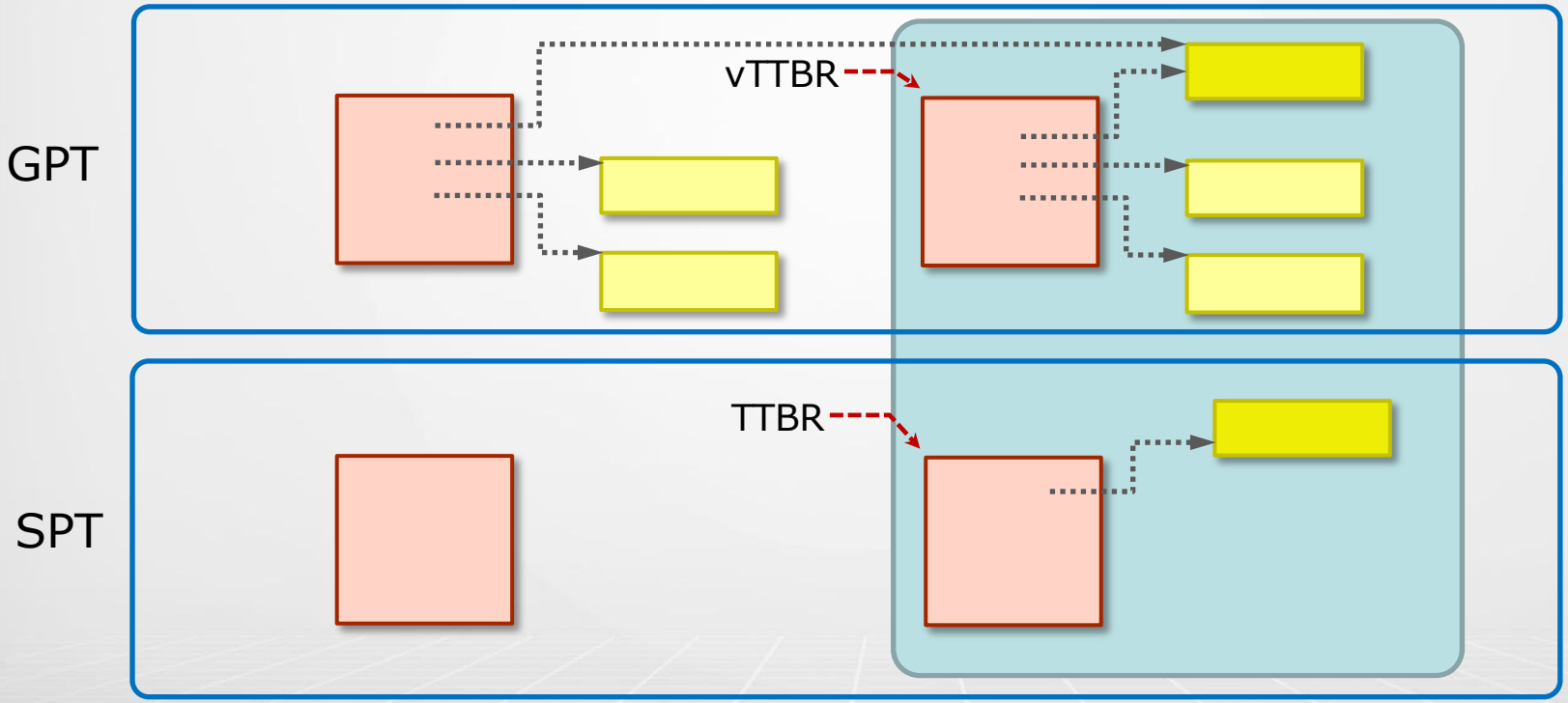
SPT Cache Issues (6)

- Shared page table
  - Cache coherence problem ➔ identifying GPT pages



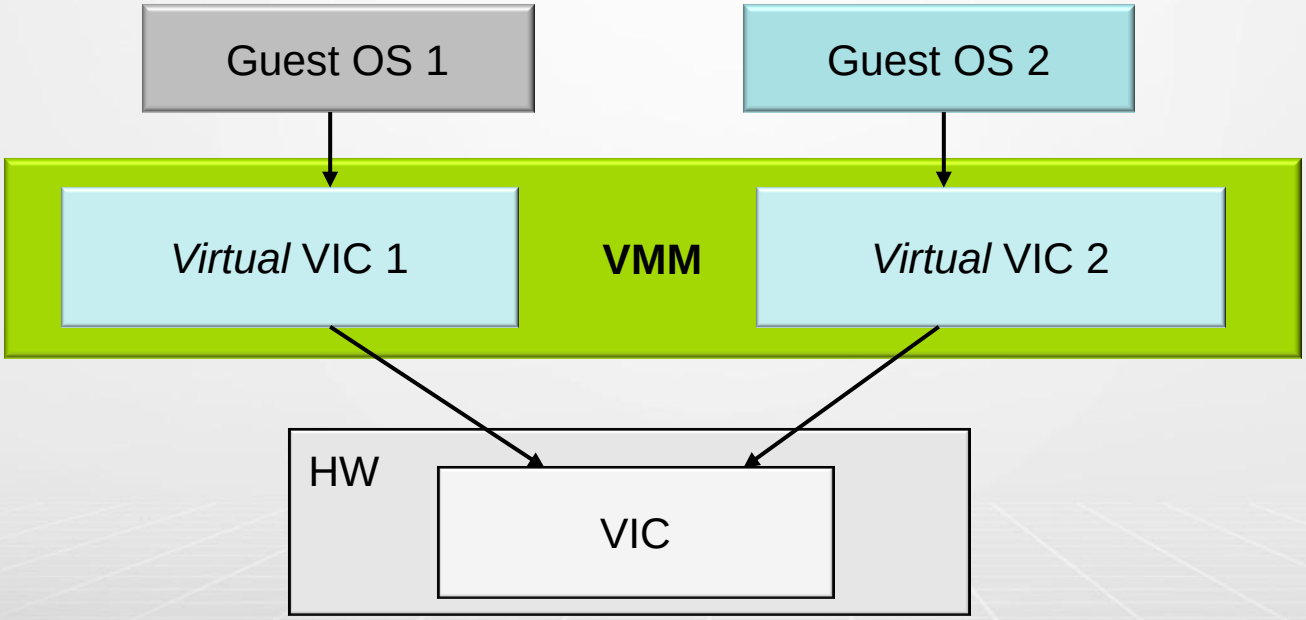
SPT Cache Issues (7)

- Shared page table
  - Complex back link management
    - ➔ Keeping one back link for each PTE SPT page



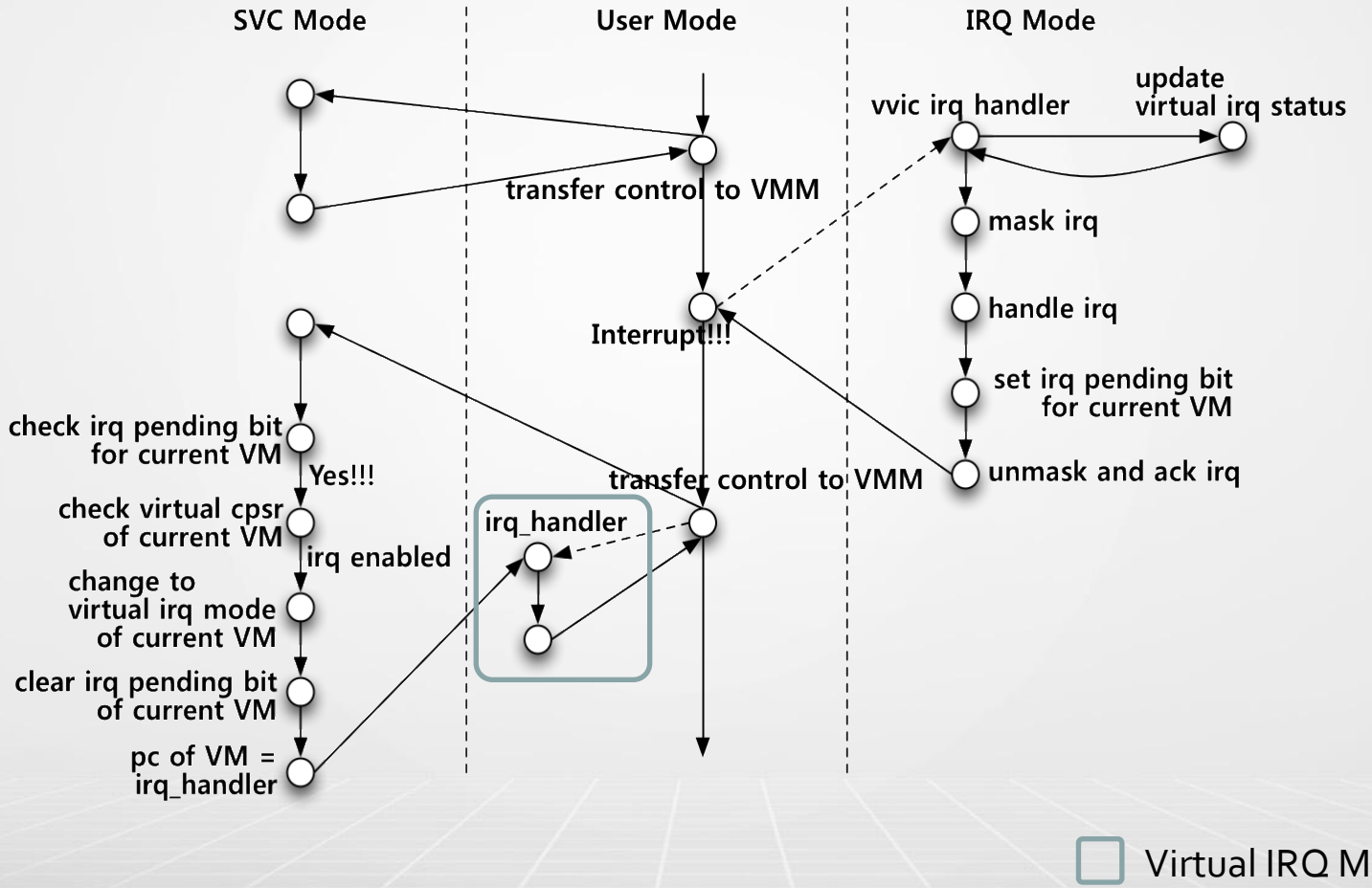
Interrupt Virtualization (1)

- VIC virtualization
  - No mapping for the register memory
  - Read only mapping for the register memory
  - Shared memory b/w ViMo and guest OS



Interrupt Virtualization (2)

- Interrupt delivery to guest OS



Address Space Compression (1)

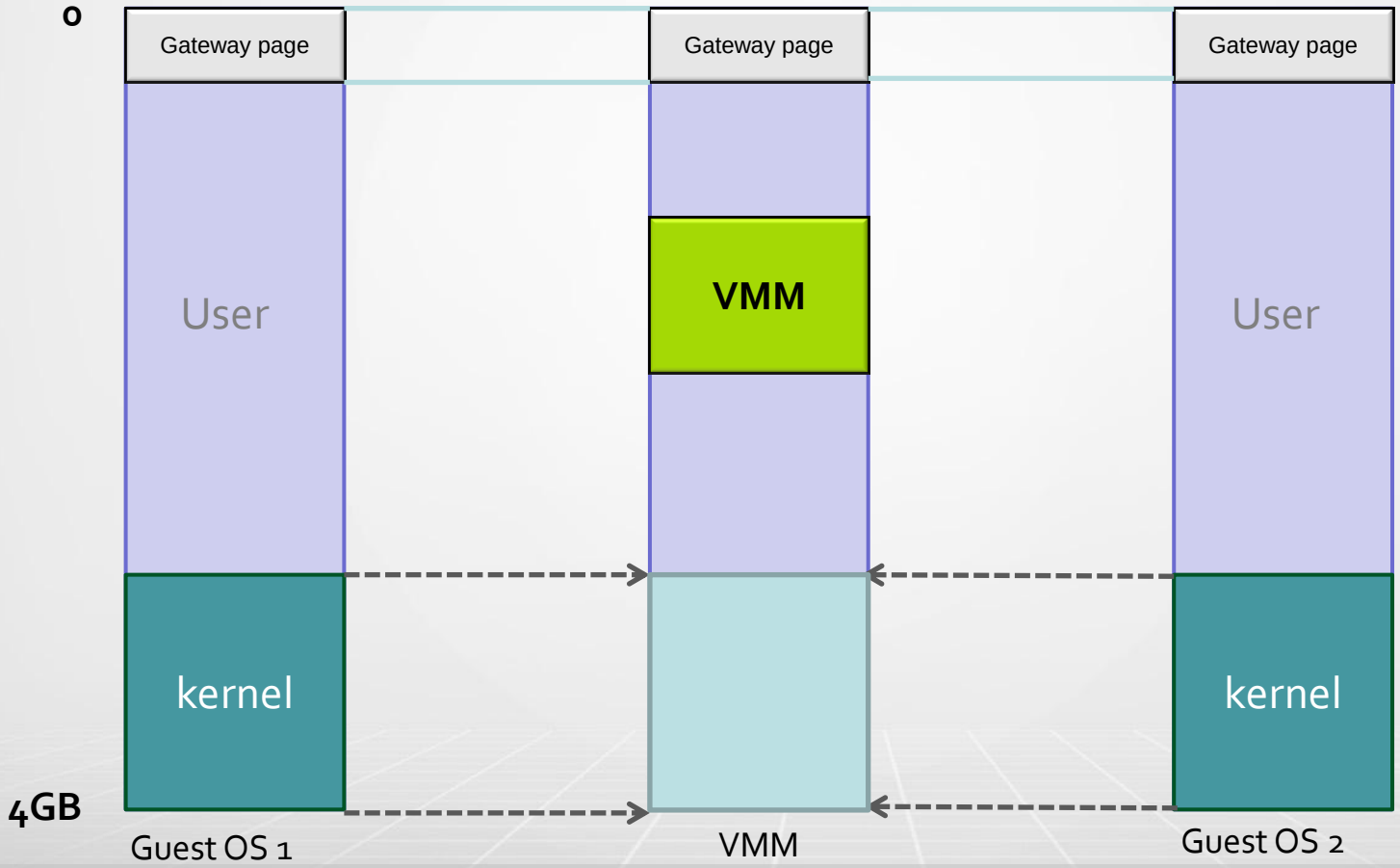
IT R&D Global Leader ETRI

- Address space compression problem
  - VMM needs to occupy part of the guest's virtual address space
  - How to protect this part of virtual address space from the guests?

- Linux does not use the first 4KB virtual address
- 4KB is too small to place VMM
- Independent 4GB virtual address space to VMM

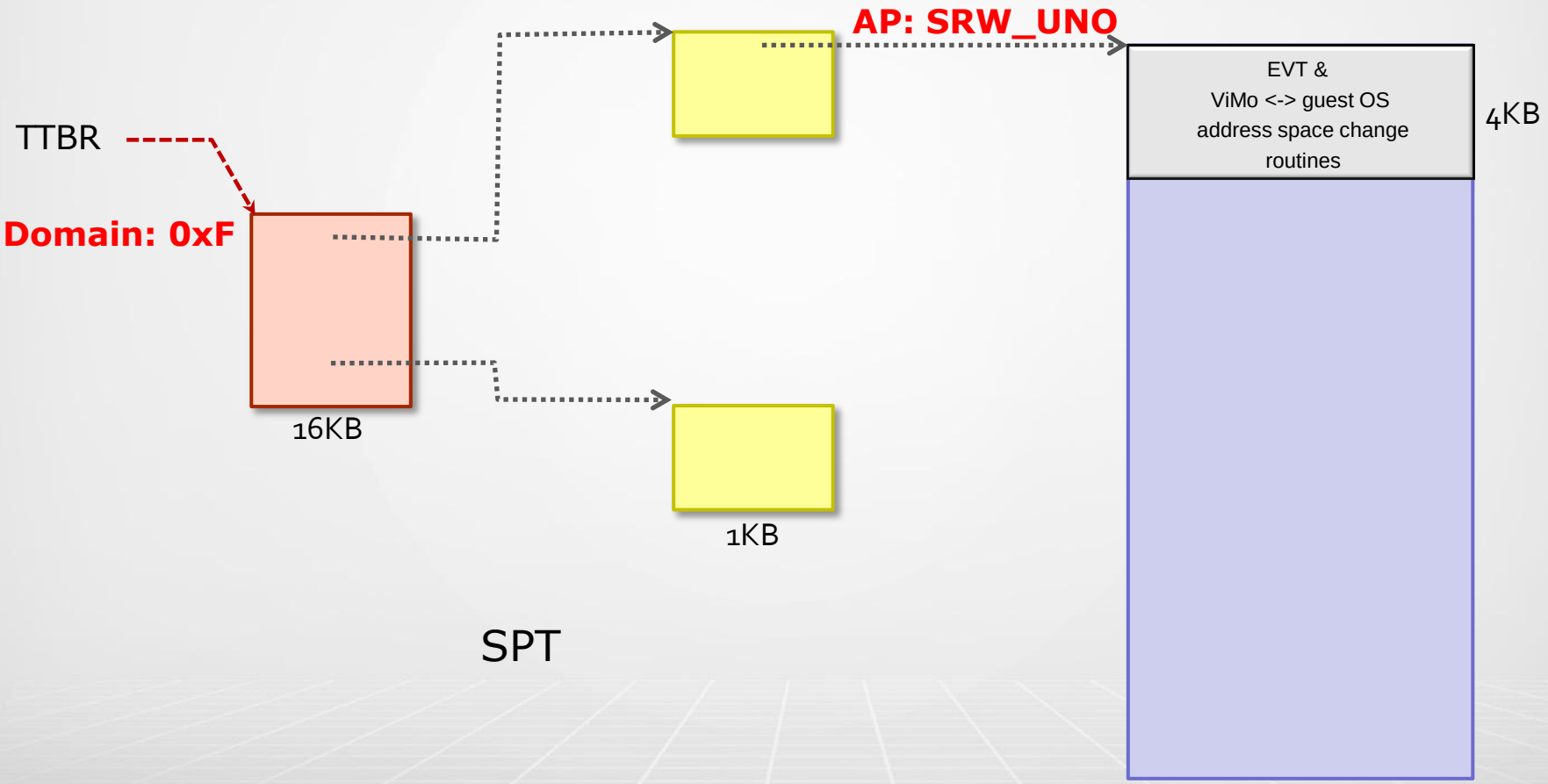
Address Space Compression (2)

- Address space change b/w guest and VMM
- VMM accessing to guest OSes



Intercepting Exceptions

- ARM exceptions
  - Undefined, SWI, Prefetch abort, Data abort, IRQ, FIQ



ARM Type-II VMM (1)

IT R&D Global Leader ETRI

- Host OS
  - Android host OS
- Guest OS
  - Linux
- CPU virtualization
  - Deprivileging
  - Pre-virtualization
- Memory virtualization
  - SPT
  - Intercepting TLB operations
- IO virtualization
  - Para-virtualization
  - Split device driver

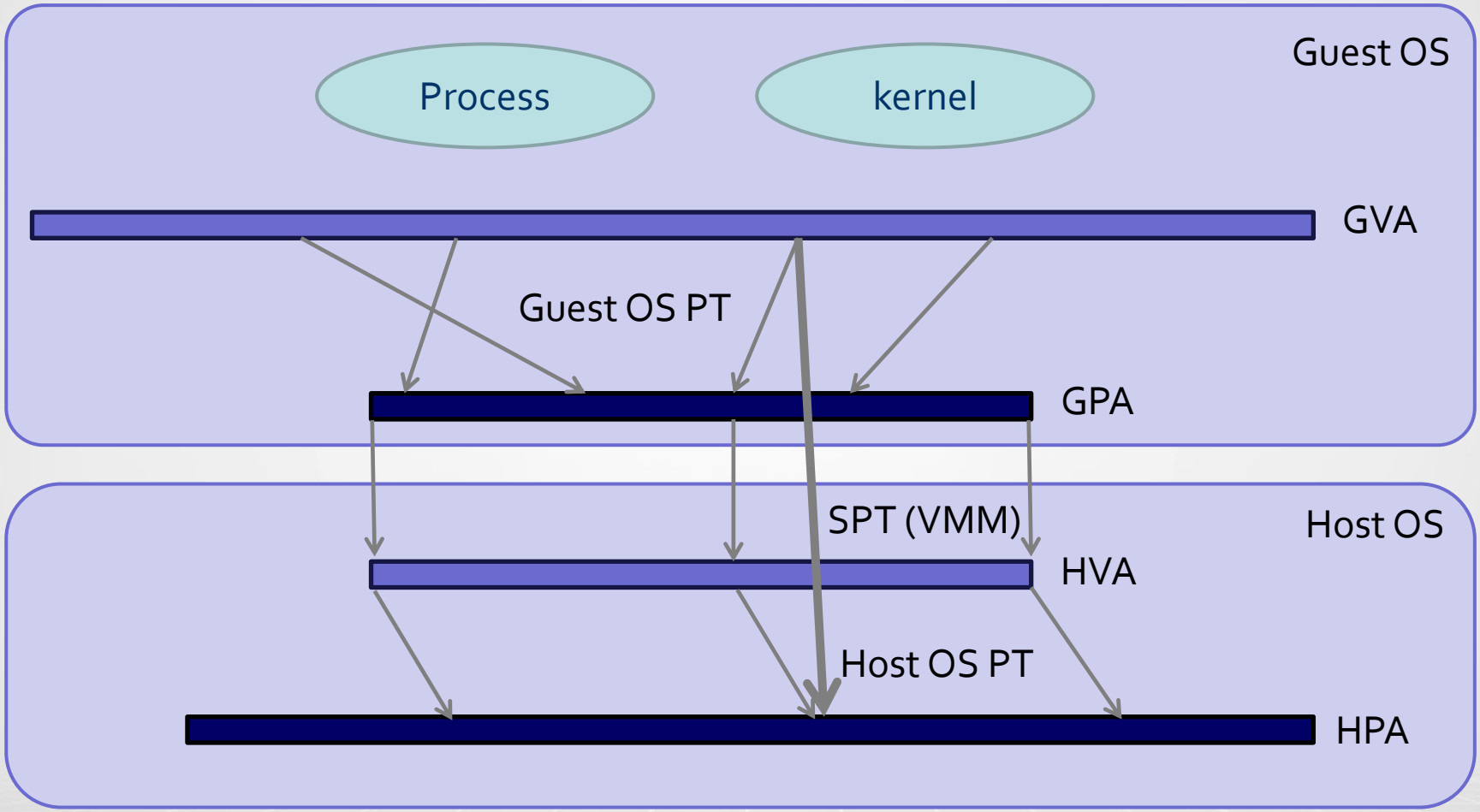
ARM Type-II VMM (2)

IT R&D Global Leader ETRI

- Deprivileging
  - One user process controlling one VM
- Pre-virtualization
  - gcc ETRI patch
  - Critical instruction location table
  - Encoding critical instructions
  - Kernel patch at loading time
  - Decode and emulate at run time

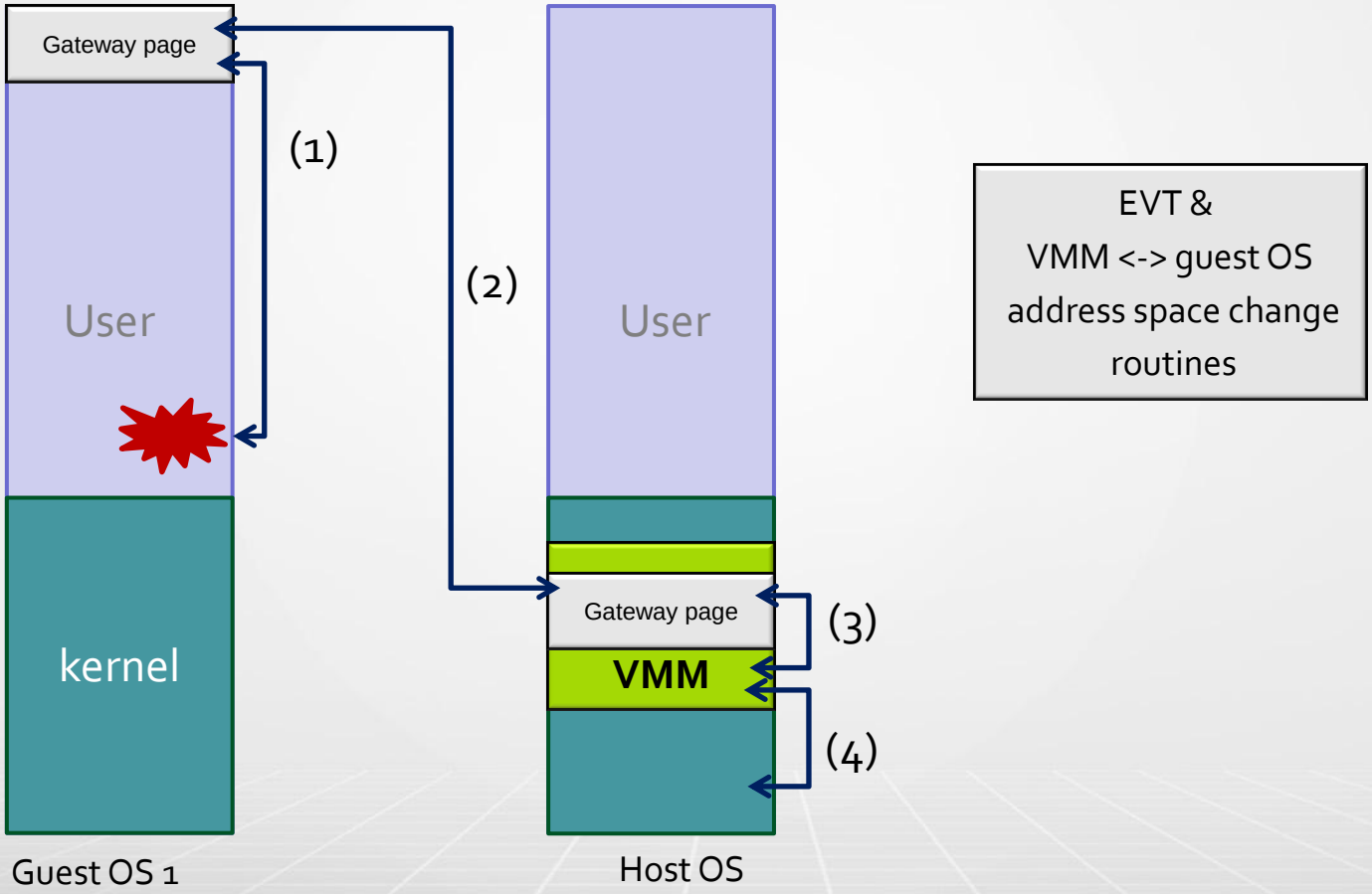
ARM Type-II VMM (3)

- Memory virtualization



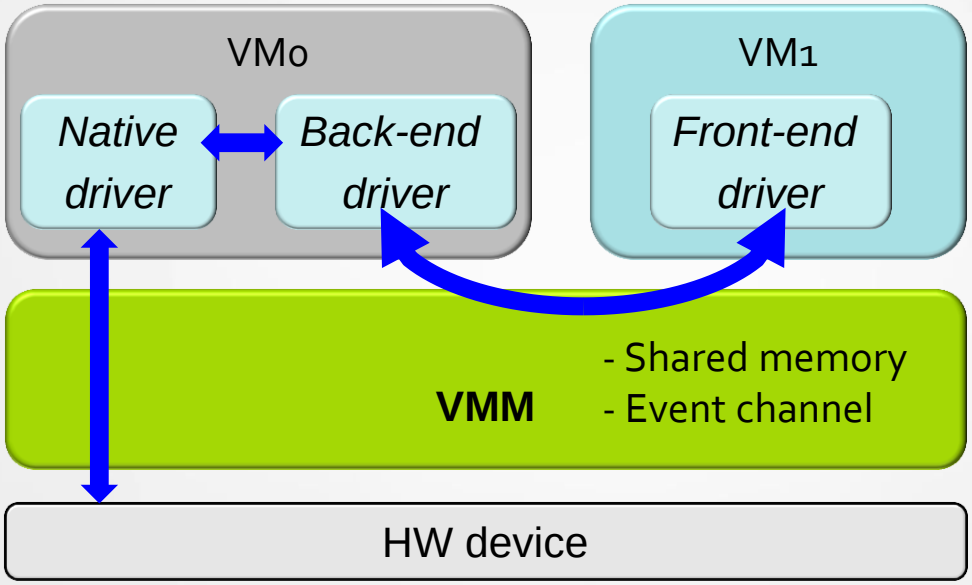
ARM Type-II VMM (4)

- Exception & Interrupt handling
  - Relocatable gateway page



I/O Virtualization

- NAND, Keypad, Audio
  - Split device driver



- Touch Screen
  - Virtual device driver at VMM

Q&A

ViMo-TypeII

- 호스트 기반 가상머신 모니터
  - 호스트: 안드로이드 2.2, 2.3, 3.x, 4.0
  - 가상머신: 리눅스, uC/OS-II
- 보드
  - LG전자 옵티머스 블랙
  - 삼성 갤럭시탭 (7, 10.1, 7.7), 갤럭시S/S2/Note
  - Asus Transformer Prime (Quad Core)

ATOM OS Switch

- Board: ASUS EEE PC (넷북)
  - CPU : ATOM N570 (1.6 GHz)
  - Main memory : 1GB
- OS : Windows 7 + Android
  - OS suspend and resume

Windows 7

Android

ViMo Express

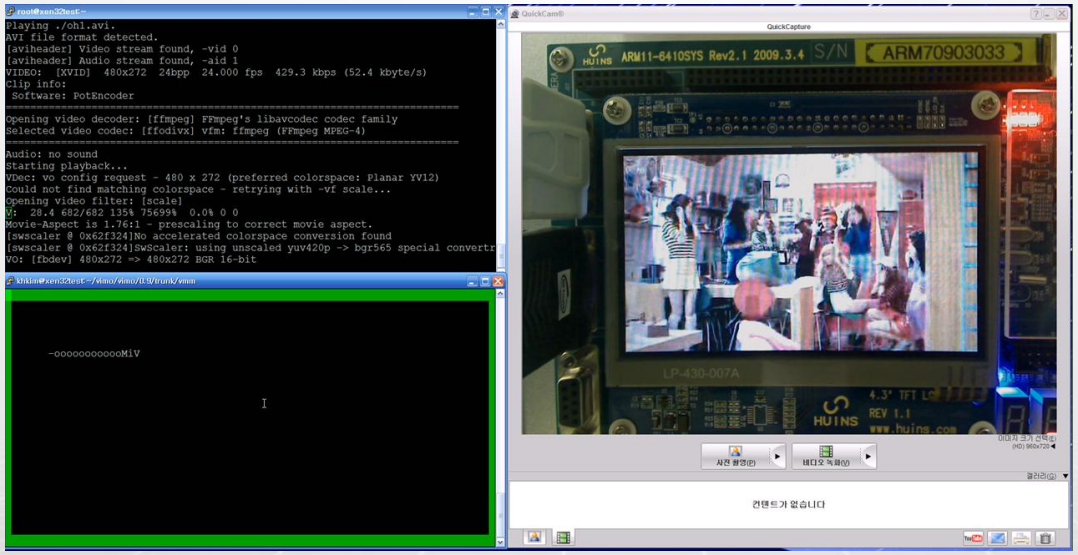
- Board : MVC100
  - CPU : S5PC100 (Cortex A8)
  - Main memory : 256MB
- OS : Android + WinCE (navigator)

ViMo Express

- Board : HUINS-AchroHD
  - CPU : S5PC100 (Cortex A8)
  - Main memory : 512MB
- OS : Android + uC/OS-II
  - Android (movie play), uC/OS-II (motor control)

ViMo v2.0

- Board : HUINS-6410sys
  - CPU : S3C6410 (ARM11)
  - Main memory : 256MB
- OS: Linux + Linux
  - Linux A (Text program), Linux B (Play movie)
  - Linux A (Kernel Panic), Linux B (play movie)



ViMo v2.0

- Board : HUINS-AchroHD
  - CPU : S5PC100
  - Main memory : 512MB
- OS: Android + X-enabled Linux
  - Android (movie play), Linux (movie play)
  - VM switching

ViMo switch

- Board: ODROID
  - CPU : S5PC100 (Cortex A8)
  - Main memory : 512MB
- OS : Android + Android
  - OS suspend and resume