KVM for Embedded PowerPC

Hollis Blanchard, IBM Linux Technology Center

KVM Forum

29 Aug 2007

Overview

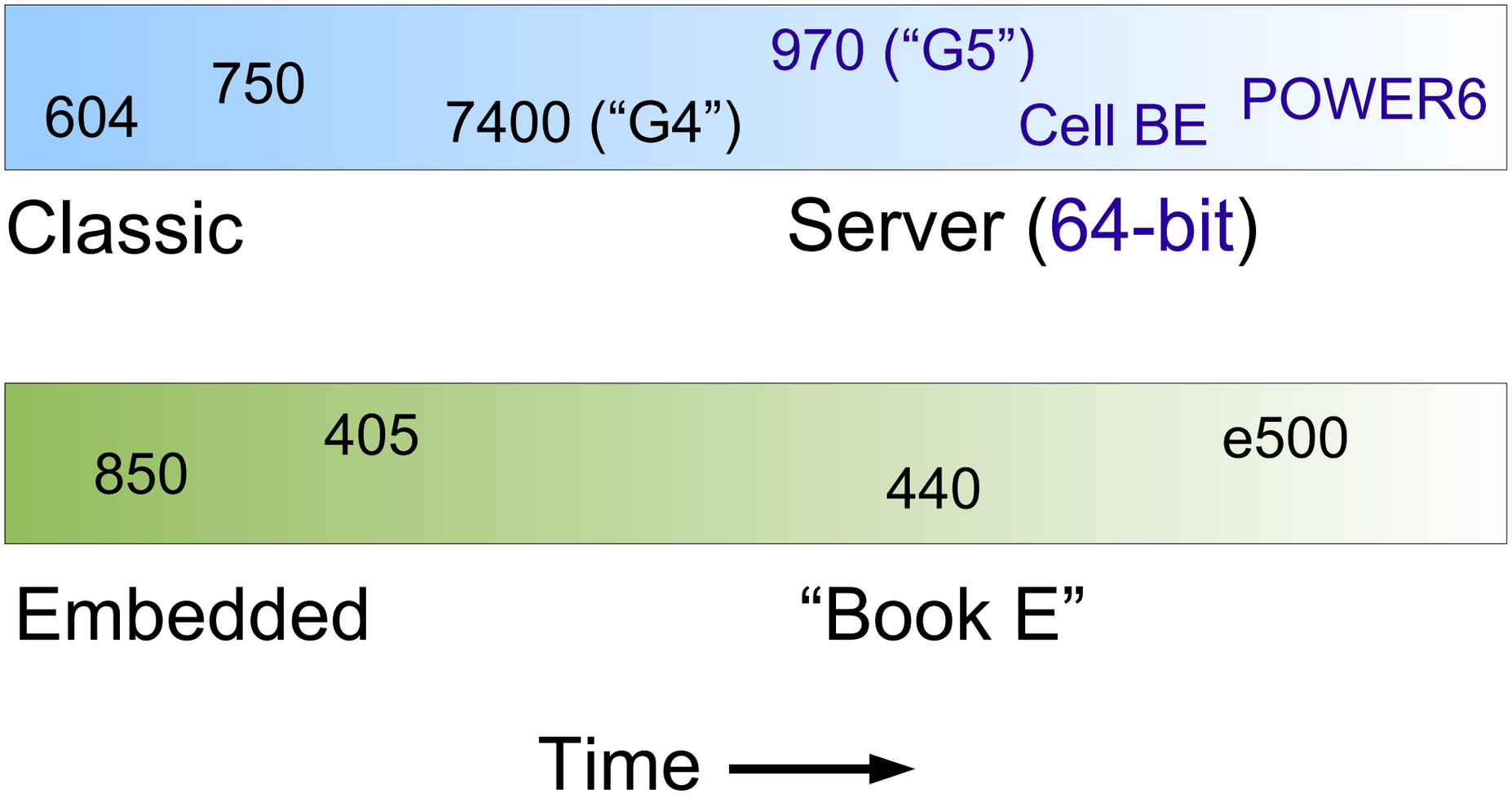
- PowerPC Architecture
- Embedded Systems
- Virtualizing PowerPC Book E

PowerPC Architecture

PowerPC in General

- 64-bit architecture with 32-bit implementations
- RISCish
- Load/store architecture
- 32 general-purpose registers
- PowerPC cores developed by IBM, Freescale, AMCC, PA Semiconductor, Sony/Toshiba/IBM
- A single core may be packaged into many systems-on-a-chip (SoC)
  - E.g. 440 core in 440GP, 440EP, 440SP, 440SPE, 440GX, ...
  - Each has different on-chip IO devices, often at different physical addresses
  - Some have an FPU, some don't.

PowerPC Architecture Evolution



Server PowerPC Architecture

- MMU
  - Hardware-walked hash table
    - Large contiguous block of memory
  - Real mode (MMU disabled)
  - Segmentation differentiates address spaces
- Interrupt handlers at fixed addresses near 0 (e.g. 0x500)
- Hypervisor support in hardware
  - Additional privilege level, hypercall instruction, etc
- Almost all systems using server PowerPC already ship with a hypervisor

Book E PowerPC Architecture

- MMU
  - Software-controlled TLB
    - On-chip array; no hardware pagetable walker
    - Small number of simultaneous translations, e.g. 64 entries
  - No real mode
    - Software must ensure interrupt handlers are always mapped
  - PID register differentiates address spaces
- Programmable interrupt handler addresses
- Sophisticated cache control and locking
- No hypervisor support

Embedded Systems

- "Embedded system" means dedicated-purpose
  - Not necessarily slow or low-power
- PowerPC is popular in relatively high-performance embedded applications
  - Routers, RAID controllers, game consoles, ...
- SoC means a core plus integrated IO
  - IO includes SDRAM controller, UART, Ethernet, I^2C, USB, ...
- Embedded PowerPC can be found as bare cores (add your own northbridge) and as SoCs
  - Freescale's e600 core in both 8641 SoC and 7448 processor
  - AMCC 440 core is found in 440GP, 440GX, 440SP, 440EP, ...
  - IBM sells custom SoCs: mix-and-match cores and IO blocks

Why Virtualization for Embedded Systems?

- Lots of reasons – more than possible to list here
- Legacy supervisor-level software
  - Proprietary 3rd-party operating systems, homebrew kernels
  - Exploit multi-core chips with single-core software stacks
- Workload consolidation (e.g. device control + UI)
- Protect intrusion detection services
- Uniprocessor software on multicore processors
- Sandbox untrusted code (e.g. game consoles)
- Reliable remote kernel upgrades
- RTOS + Linux
- Improve reliability through isolation of privileged code

Why KVM for Embedded PowerPC?

- Linux's pre-existing support for the diversity in embedded PowerPC
  - Linux core support: 405, 440, 850, 604, 750, 970, e500, e300, e600, Cell, 1682M, ...
  - Architecture families: Classic, Book E, Server
- What Linux code can we reuse?
  - Bootstrap support (firmware handoff)
  - MMU support
  - PIC drivers
  - scheduler (with real-time support)
  - memory allocator

Virtualizing PowerPC Book E

- Instruction virtualization
- Memory virtualization
- Performance implications
- KVM interface

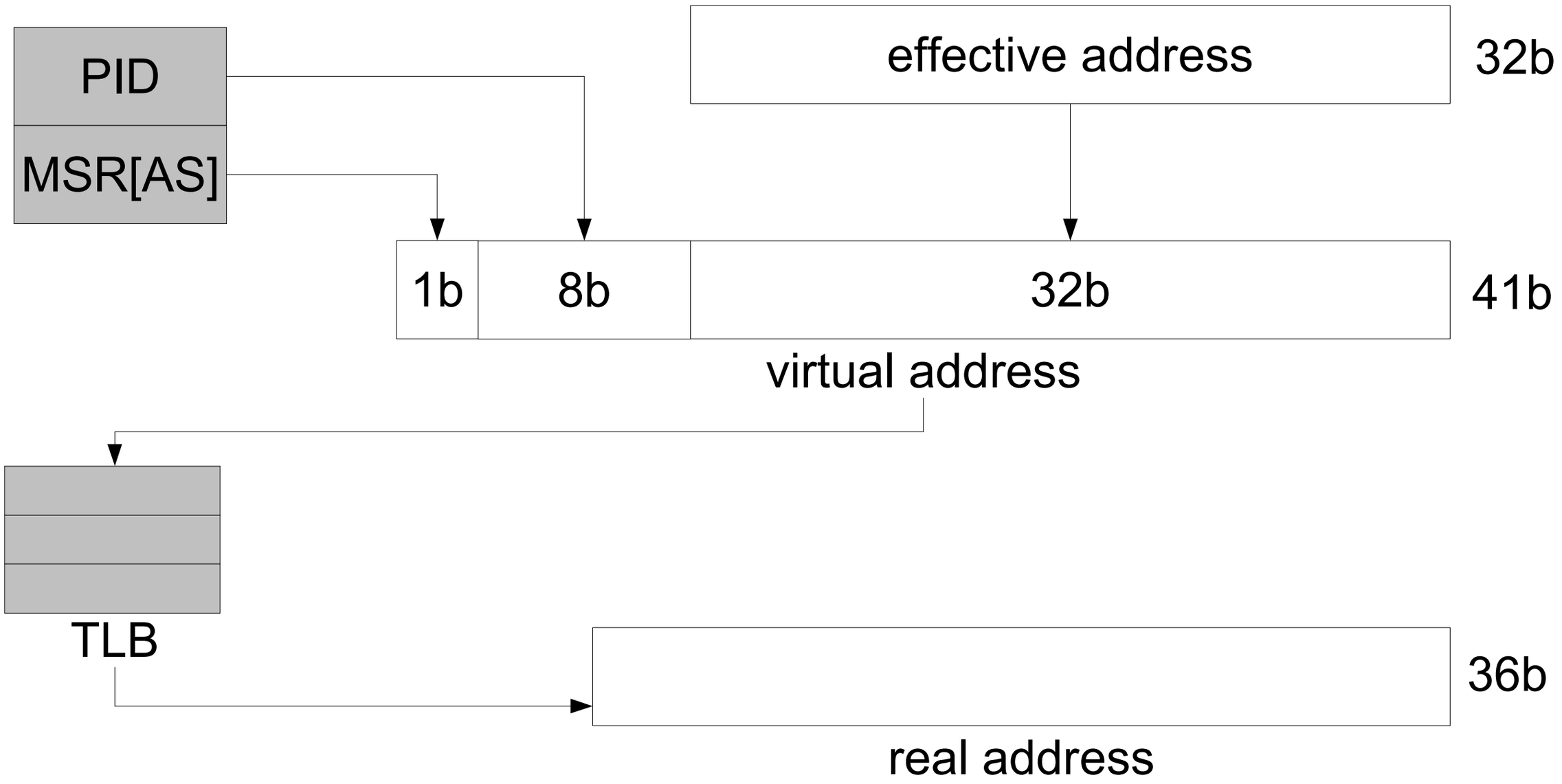
Instruction Virtualization

- No hypervisor mode in hardware; only user and supervisor privilege levels
  - To enforce isolation, guest kernels run in user mode
- All supervisor instructions executed in user mode trap to the host
- Host can decode and emulate the instructions
  - Full virtualization
- Unfortunately, there are a lot of privileged instructions executed by every interrupt handler
  - Performance of interrupt-heavy workloads will suffer
- However, user mode instructions execute natively
  - Performance of compute-bound workloads should be decent

Data TLB Miss Fast Path

```
        mtspr   SPRN_SPRG0, r10
        mtspr   SPRN_SPRG1, r11
        mtspr   SPRN_SPRG4W, r12
        mtspr   SPRN_SPRG5W, r13
        mfcr    r11
        mtspr   SPRN_SPRG7W, r11
        mfspr   r10, SPRN_DEAR
        lis     r11, TASK_SIZE@h
        cmplw   r10, r11
        blt+    3f
        lis     r11, swapper_pg_dir@h
        ori     r11, r11, swapper_pg_dir@l
        mfspr   r12,SPRN_MMUCR
        rlwinm  r12,r12,0,0,23
        b       4f
3:      mfspr   r11,SPRN_SPRG3
        lwz     r11,PGDIR(r11)
        mfspr   r12,SPRN_MMUCR
        mfspr   r13,SPRN_PID
        rlwimi  r12,r13,0,24,31
4:      mtspr   SPRN_MMUCR,r12
        rlwinm  r12, r10, 13, 19, 29
        lwzx    r11, r12, r11
        rlwinm. r12, r11, 0, 0, 20
        beq     2f
        rlwimi  r12, r10, 23, 20, 28
        lwz     r11, 4(r12)
        andi.   r13, r11, _PAGE_PRESENT
        beq     2f
        ori     r11, r11, _PAGE_ACCESSED
        stw     r11, 4(r12)
        lis     r13, tlb_44x_index@ha
        lwz     r13, tlb_44x_index@l(r13)
        lis     r11, tlb_44x_hwater@ha
        lwz     r11, tlb_44x_hwater@l(r11)
        addi    r13, r13, 1
        cmpw    0, r13, r11
        ble     7f
        li      r13, 0
7:      lis     r11, tlb_44x_index@ha
        stw     r13, tlb_44x_index@l(r11)
        lwz     r11, 0(r12)
        lwz     r12, 4(r12)
        rlwimi  r11, r12, 0, 0 , 19
        tlbwe   r11, r13, PPC44x_TLB_XLAT
        li      r11, PPC44x_TLB_VALID | PPC44x_TLB_4K
        rlwimi  r10, r11, 0, 20, 31
        tlbwe   r10, r13, PPC44x_TLB_PAGEID
        li      r10, PPC44x_TLB_SR@l
        rlwimi  r10, r12, 0, 30, 30
        rlwimi  r10, r12, 29, 29, 29
        rlwimi  r10, r12, 29, 28, 28
        rlwimi  r11, r12, 31, 26, 26
        and     r11, r12, r11
        rlwimi  r10, r11, 0, 26, 26
        rlwimi  r12, r10, 0, 26, 31
        rlwinm  r12, r12, 0, 20, 15
        tlbwe   r12, r13, PPC44x_TLB_ATTRIB
        mfspr   r11, SPRN_SPRG7R
        mtcr    r11
        mfspr   r13, SPRN_SPRG5R
        mfspr   r12, SPRN_SPRG4R
        mfspr   r11, SPRN_SPRG1
        mfspr   r10, SPRN_SPRG0
        rfi
```

Book E Address Spaces



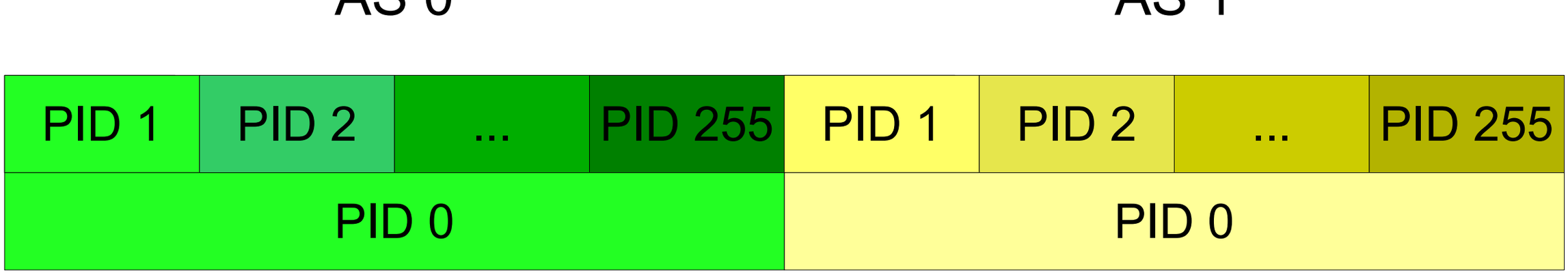
PowerPC 440 MMU

- Fully associative software-managed 64-entry TLB
- Variable page sizes (1KB – 1GB)
- 36-bit physical addresses

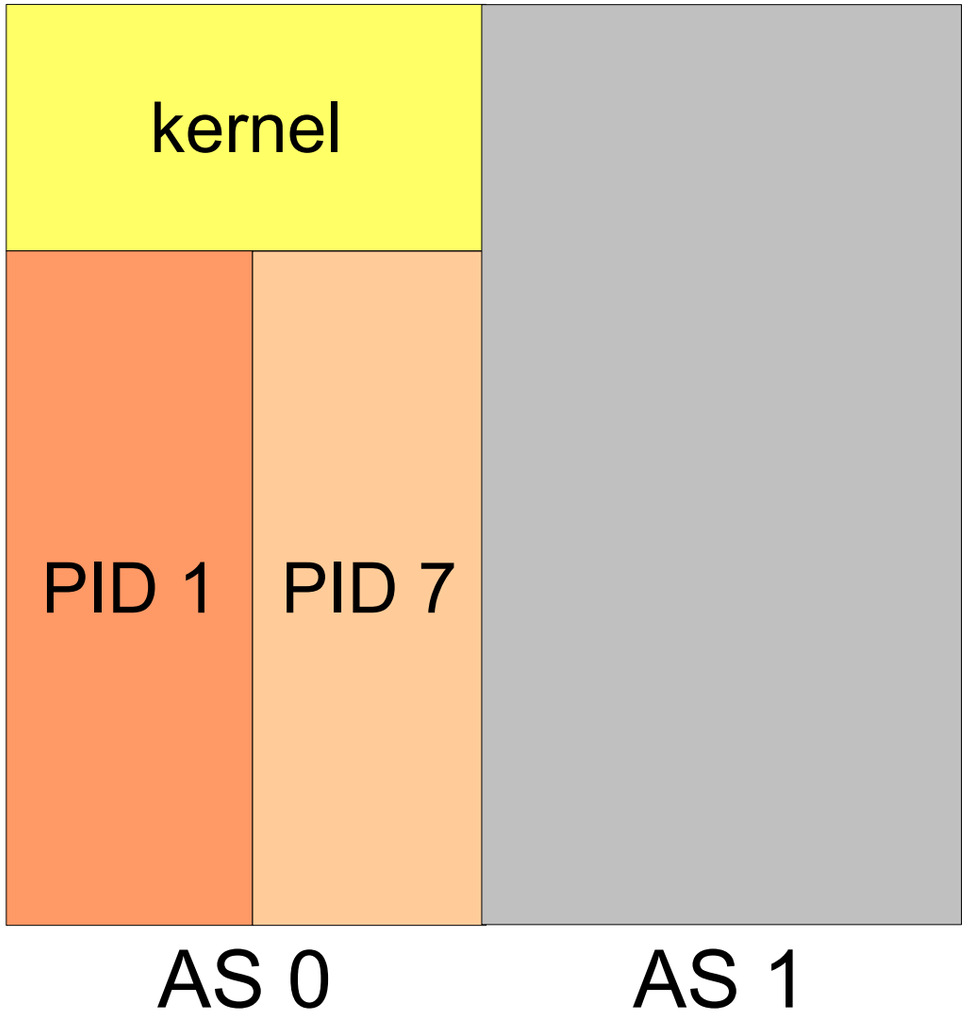
63
:
1
0

Book E Address Spaces

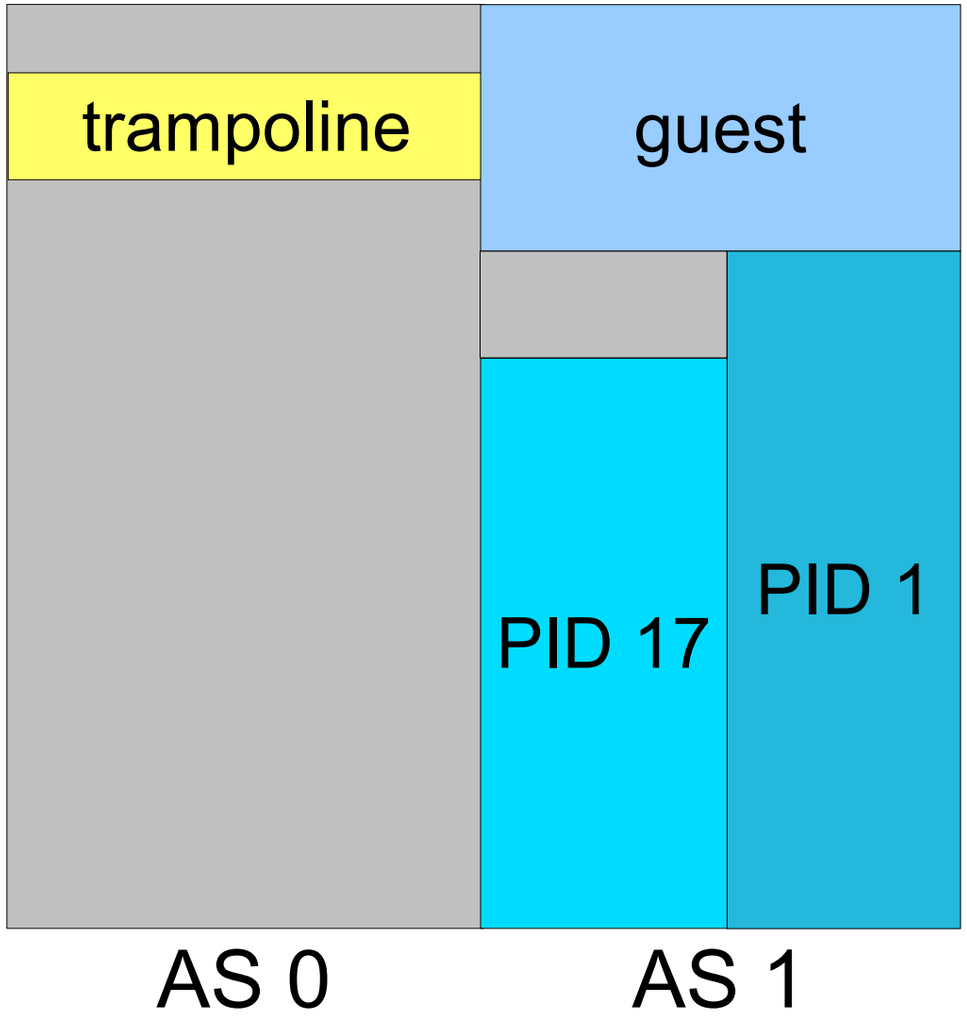
- PID 0 matches entire AS
- 255 areas (4GB each) per AS



Address Space Virtualization

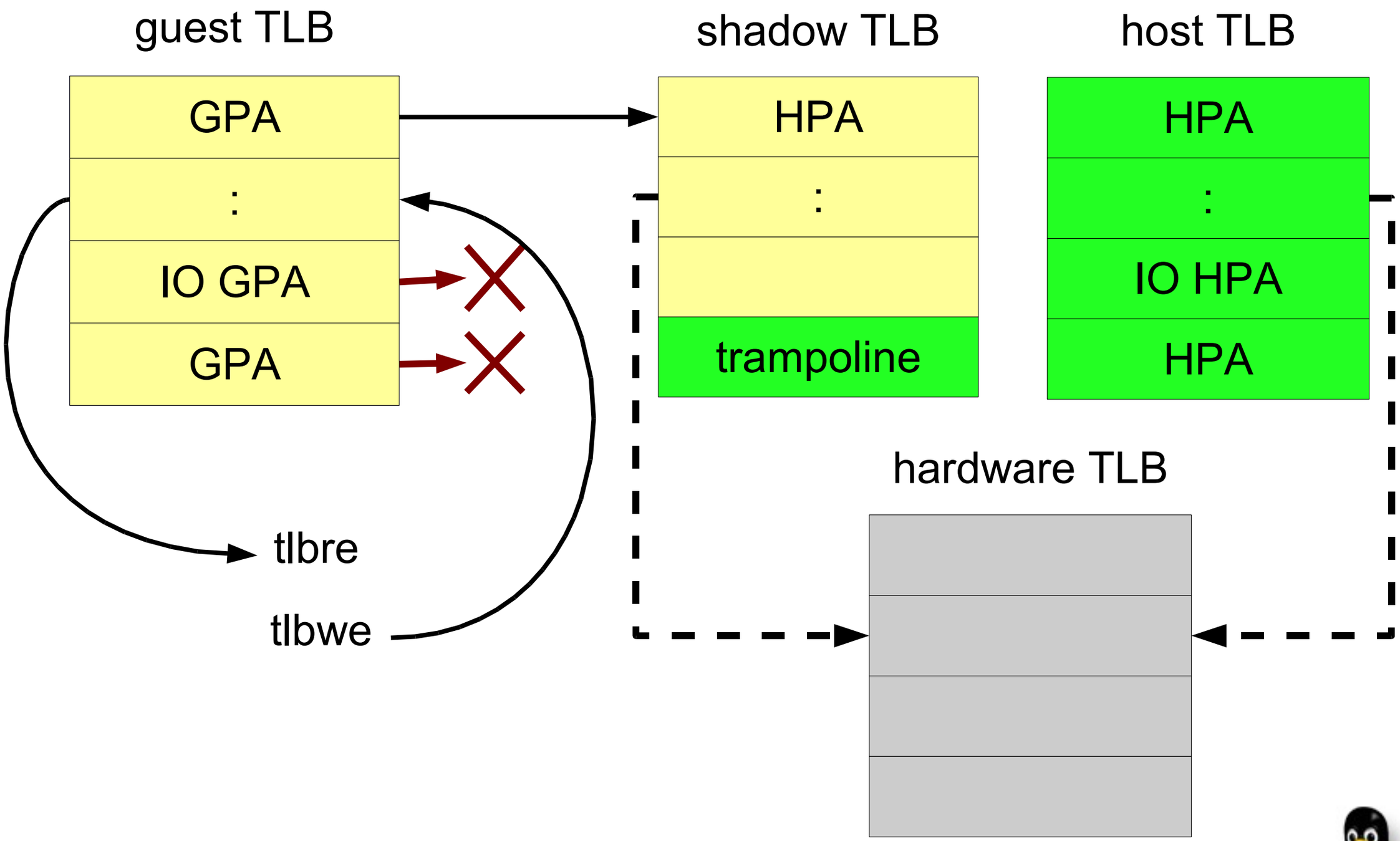


plain Linux



KVM

TLB Virtualization



Performance Loss Mitigations

- Unmodified guests: rewrite instruction stream
  - Guest access to many SPRs do not require host processing
  - Could rewrite “mtspr” instructions to store to a per-vcpu memory mapping
  - All PowerPC instructions are 32 bits wide
  - Branch instructions are relative, and have +/- 8MB range
    - If we need to insert more than one instruction, things start to get hairy
- Modified guests
  - Don't abstract TLB entries; abstract the whole MMU
  - Virtual IO device drivers

KVM Kernel/Userspace Interface

- struct kvm_vcpu defined per architecture
  - Shared KVM code treats as an opaque pointer
- Userspace still needs some PowerPC knowledge
  - qemu, libkvm, etc
- What is the completely common layer?
  - virt-manager? Higher?
- Explitiсly sized types
  - “long” changes size between 32- and 64-bit ABIs
- Endianness
  - Casting to char* should be a warning flag
- Bit fields
  - Not portable! No.

KVM Guest/Host Interface

- Explicitly sized types, endianness, bitfields, etc
- Do not pass virtual addresses!
  - Infeasible or impossible to translate virtual -> physical on some architectures
- Atomic operations only on architecture-defined types
  - 32-bit PowerPC atomic operations have 4-byte granularity
  - E.g. shared “pending interrupts” bitfield

Current Status

- Prototyping only
- Can execute ~50 Linux early boot instructions
- Low-level code only; not sharing any KVM code today
  - x86 code mixed with generic code
  - KVM ioctl interface is x86-only
- No qemu interaction yet
  - Just a simple loader

Summary

- Virtualization is becoming very important to embedded applications.
  - Embedded applications bring unique requirements.
- Embedded PowerPC is a clean architecture but lacks hardware virtualization support.
- Virtualizing embedded PowerPC presents some interesting technical challenges.