裸金属二进制翻译器的设计和实现

Martins3

2022 年 5 月 30 日

The Open Source Community

目录

研究背景

二进制翻译器的应用场景

- 安全，逆向，性能和调试
- **跨架构**

跨架构二进制翻译器

任何指令集架构初期都需要一个二进制翻译器

- 没有硬件的情况下，软件工程师可以在二进制翻译器上开发操作系统。
- 软件适配需要时间。Apple 使用 Rosetta 2 让软件厂商有时间适配 ARM 架构的 Mac 。

龙芯推出了基于 LoongArch 架构的 3A5000，相较于上一代的基于 MIPS 架构的 3A4000，有 50% 的性能提升。

跨架构二进制翻译

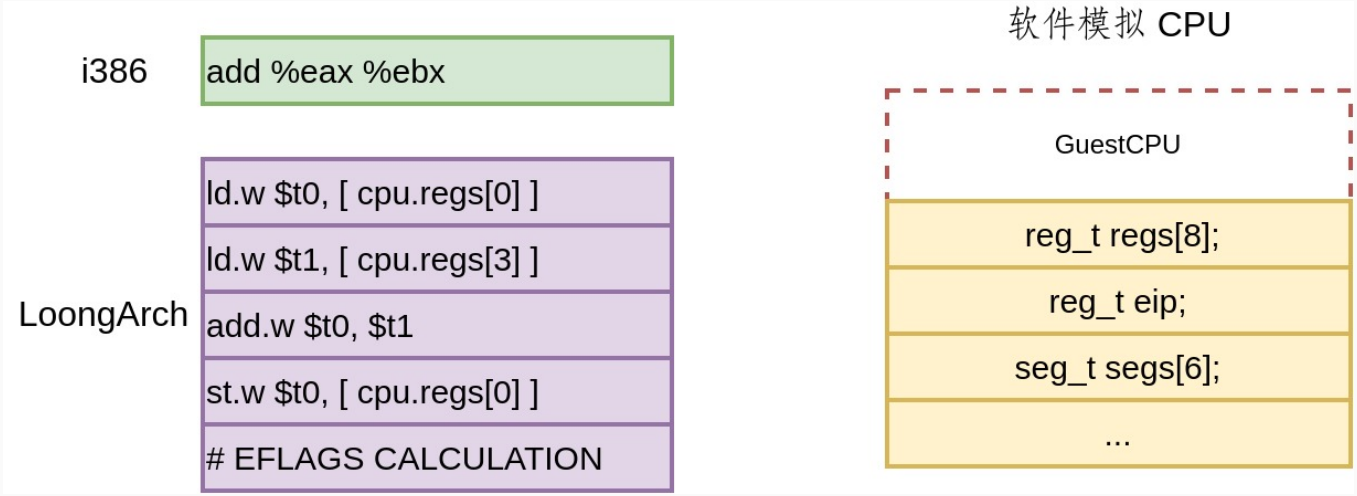


图 1: 将一条 x86 指令翻译成为 LoongArch 指令

- 使用软件模拟硬件。
- 根据指令手册使用宿主机（Host）指令模拟客户机（Guest）指令。

静态二进制翻译

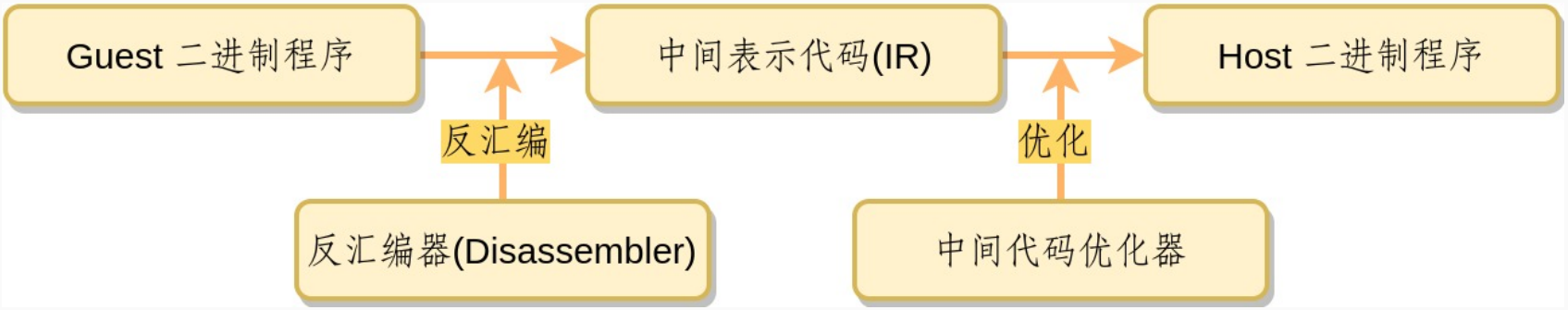


图 2: 静态二进制翻译器执行流程

静态二进制翻译在 Guest 程序执行之前完成所有的翻译工作，因为缺乏运行时信息，能够翻译的 Guest 程序类型有限。

动态二进制翻译

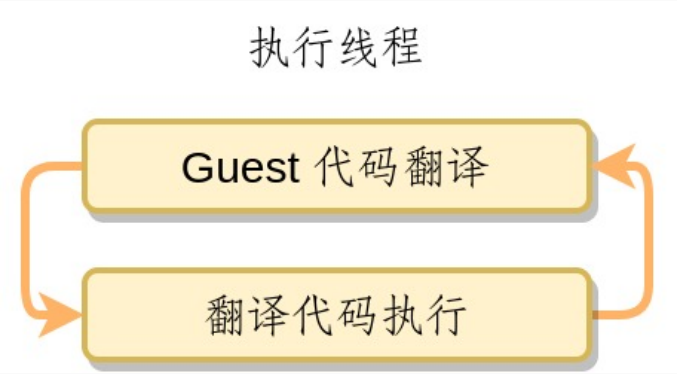


图 3: 动态二进制翻译器执行流程

动态二进制翻译器采取边翻译边执行的模式，利用运行时信息来指导下一步的翻译。

动态跨架构二进制翻译的优化：翻译代码缓存

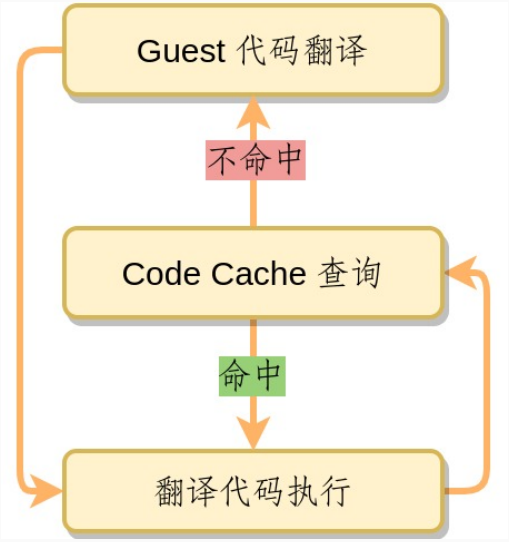


图 4：含有 Code Cache 的执行流程

翻译过的 Guest 代码无需重复翻译，当下次执行的时候可以直接使用缓存的翻译代码（Code Cache）。

动态跨架构二进制翻译的优化：翻译代码链接

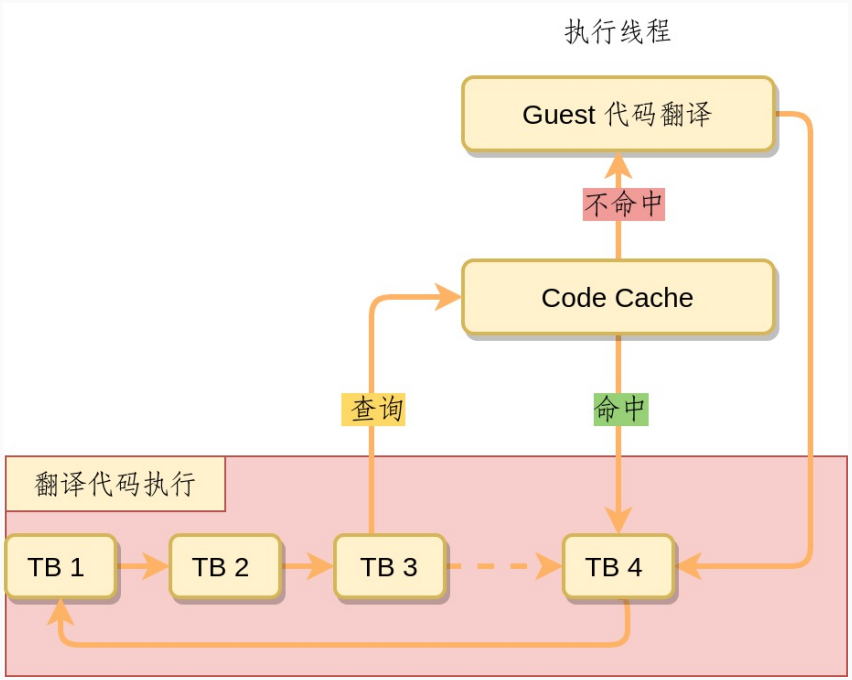


图 5: 含有 TB Chain 的执行流程

最长的连续的不含有分支的指令构成一个翻译基本块 (Translation Block, 简称 TB)。将 TB 链接起来从而消除 Guest 和 Host 的上下文切换。

系统级和进程级二进制翻译对比

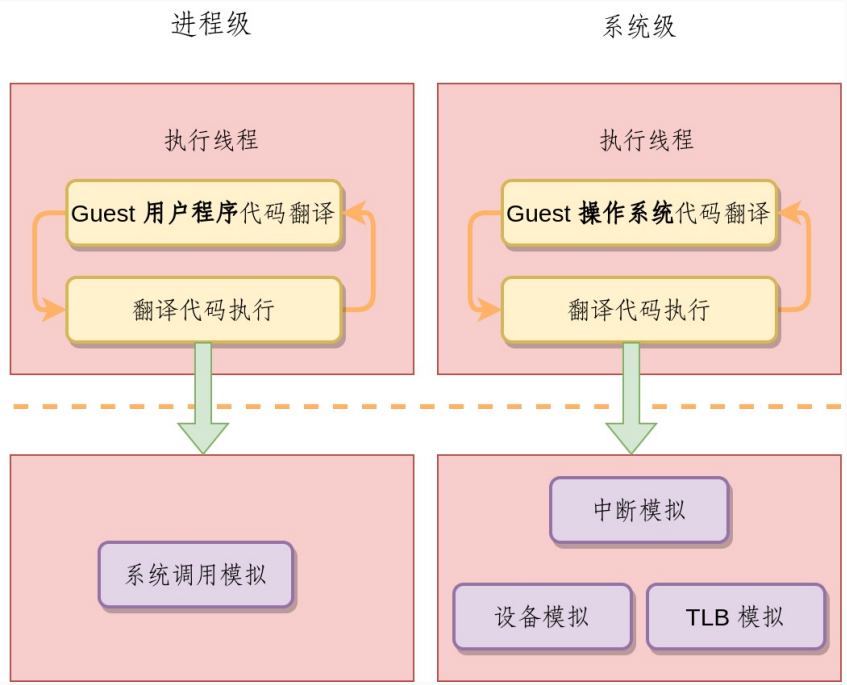


图 6: 系统级和进程级二进制翻译的执行流程

基本流程相同，但是系统级二进制翻译器无需模拟系统调用，但是需要模拟设备，TLB 和中断。

进程级二进制翻译器的问题

- 如果想要 x86 Window 程序运行在 LoongArch Linux 上需要引入 Wine 来模拟操作系统的差异。
- 4K 页 Guest 运行在 16K 页的 Host 上，需要特殊的处理。

系统态二进制翻译的挑战

CPU

指令增多，部分指令在系统态有额外的语义

设备

- 模拟设备，然后将设备的请求转发给 Host 操作系统

中断

- Host 接收中断，模拟中断控制器，然后注入到 Guest 中

访存

- 使用软件 TLB 实现虚实地址转换

这些挑战导致系统级二进制翻译器的性能低于进程级二进制翻译器，裸金属二进制翻译器（简称 BMBT）尝试解决这些挑战。

相关工作

Transmeta Code Morphing

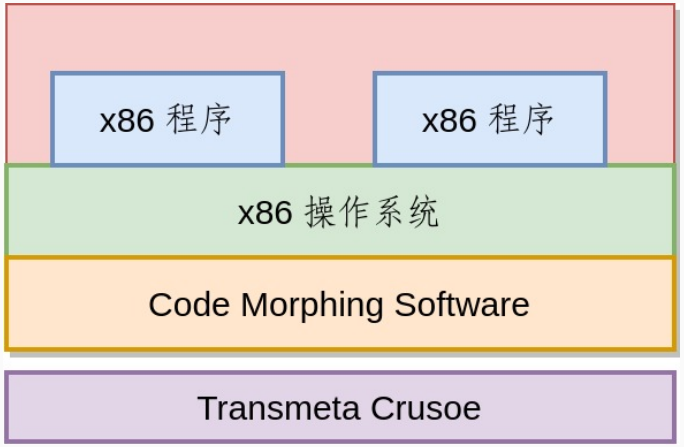


图 7: Transmeta Cursoe & CMS

1. x86 **指令翻译为** VLIW **架构指令**。
2. **额外的硬件支持加速二进制翻译**。

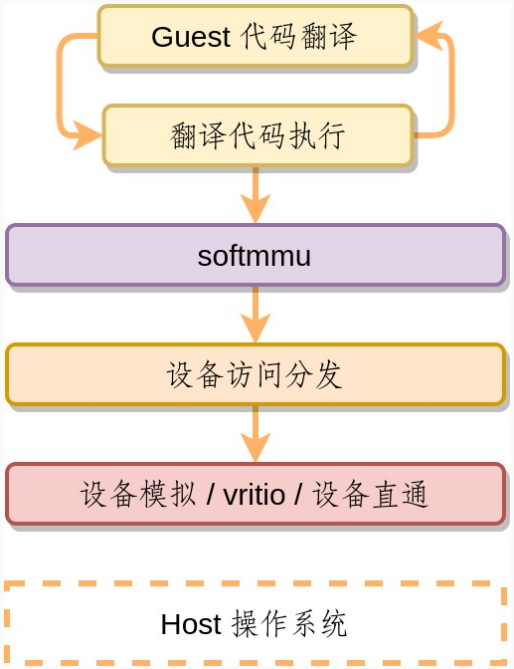


图 8: QEMU 软件架构

1. 支持 20 种架构之间的互相翻译。
2. 运行在用户态，利用操作系统来访问硬件来模拟 Guest 需要的资源。

Captive

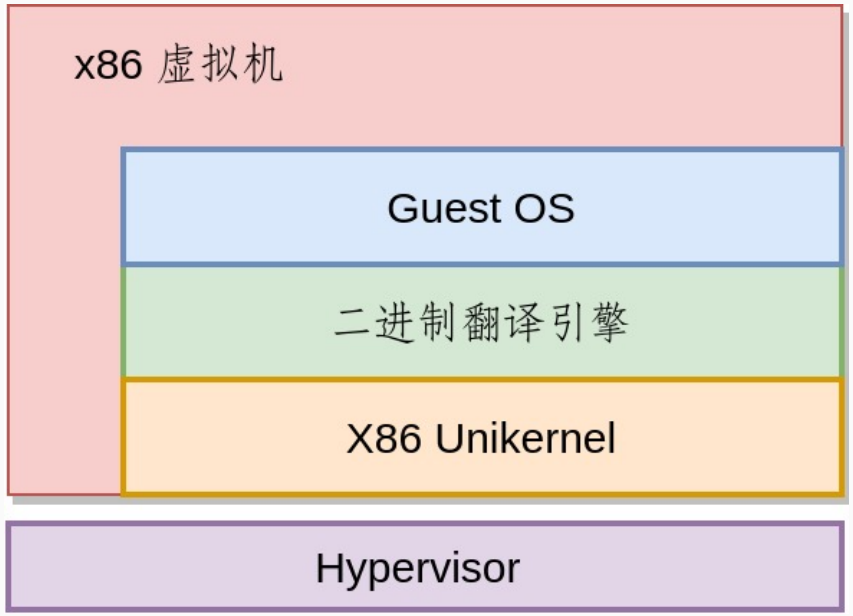


图 9: Captive 软件架构

1. 将 arm 翻译为 x86 。
2. 通过硬件辅助虚拟化的 Hypervisor 将二进制翻译器放到 non-root 态中的系统态中，从而直接访问硬件资源。

现有方案的问题

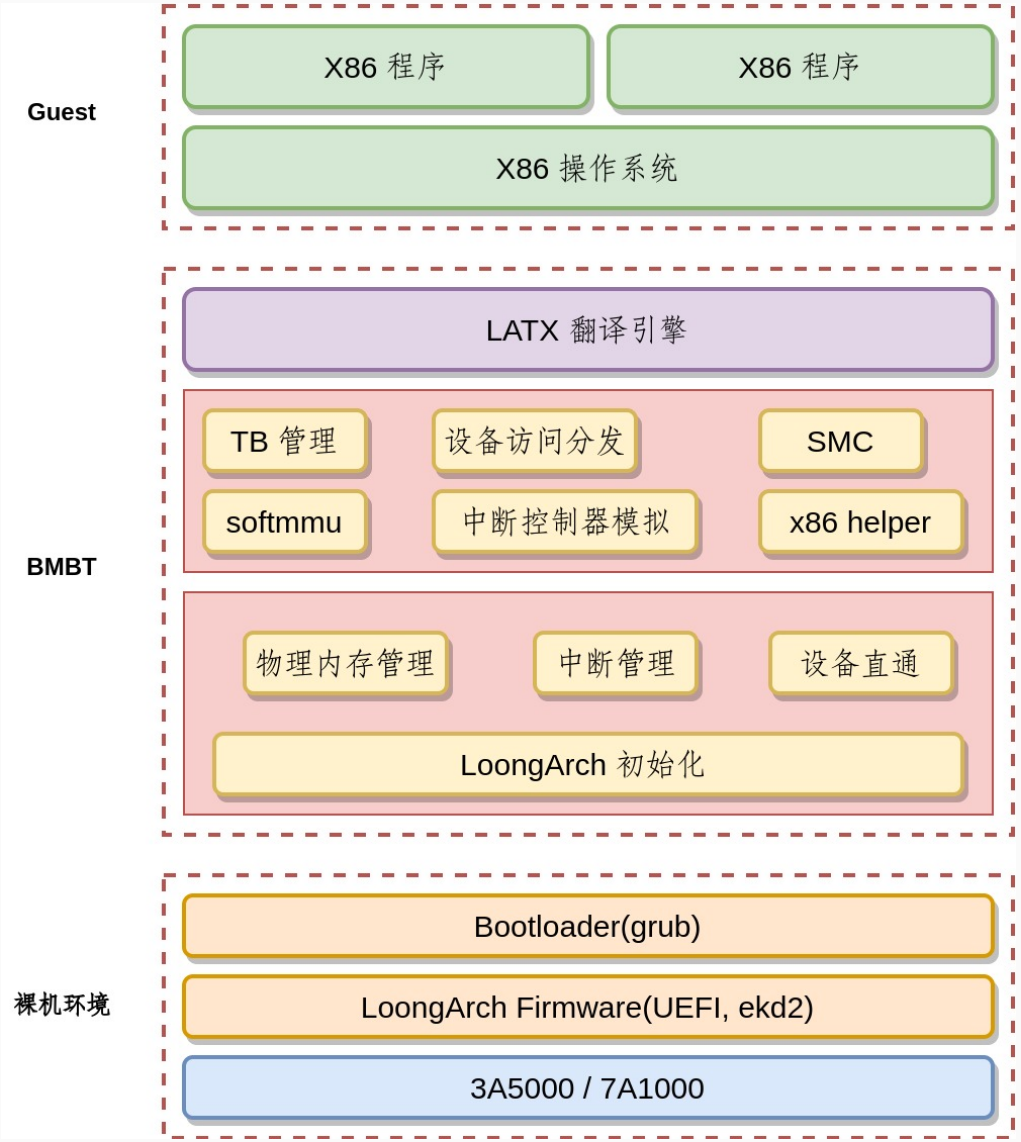
- Transmeta CMS 需要额外的硬件支持，此外 Cursoe 是 VLIW 架构，对于乱序多发射架构 CPU 下的二进制翻译器参考意义不大。
- QEMU 运行在用户态，难以消除操作系统的软件栈和用户态系统态上下文切换。
- Captive 等利用硬件辅助虚拟化直接访问硬件资源，但是虚拟机退出的开销不可忽视。

裸金属二进制翻译器

为何将二进制翻译器直接运行在系统态中

- 只有一个地址空间，没有系统态和用户态之间的上下文切换，没有 TLB miss 。
- 消除操作系统的软件栈。
- 直接访问硬件，可以实现设备直通和硬件 TLB 访存加速。

BMBT 软件架构


Guest
X86 程序
X86 程序
X86 操作系统
BMBT
LATX 翻译引擎
TB 管理
设备访问分发
SMC
softmmu
中断控制器模拟
x86 helper
物理内存管理
中断管理
设备直通
LoongArch 初始化
裸机环境
Bootloader(grub)
LoongArch Firmware(UEFI, ekd2)
3A5000 / 7A1000

CPU 虚拟化

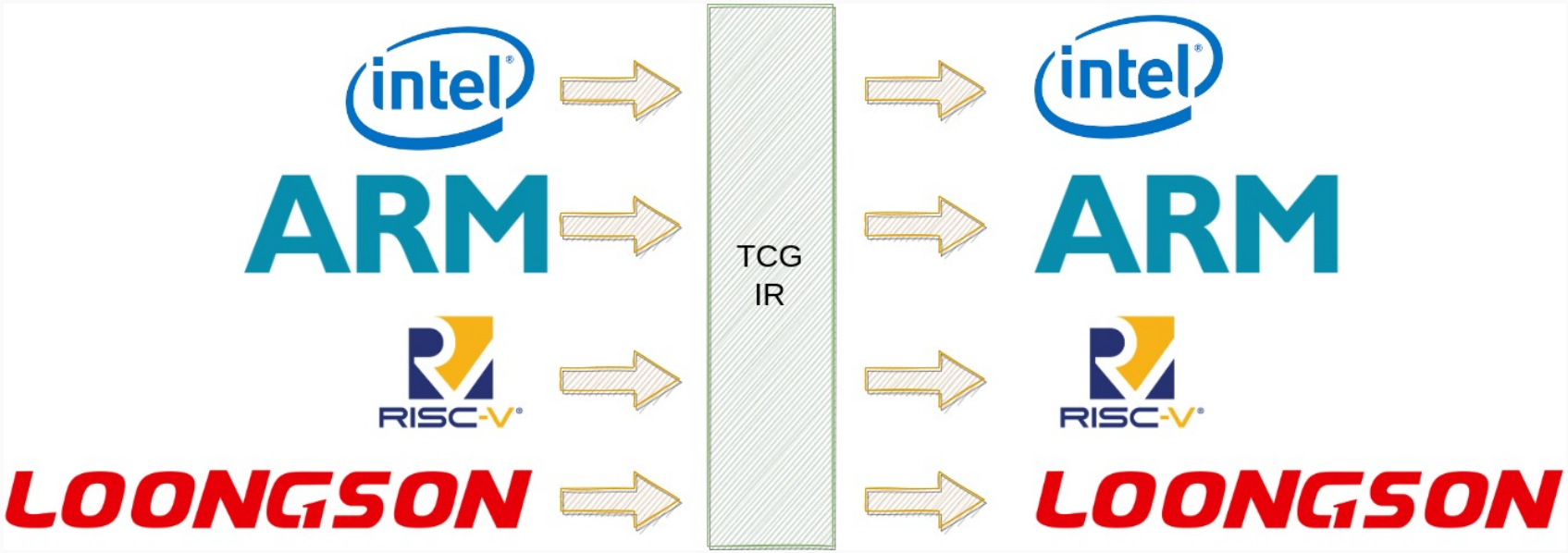


图 10: QEMU TCG

QEMU 为了实现多个架构之间的互相翻译，采用了 Tiny Code Generator（简称 TCG ）作为中间指令。

CPU 虚拟化

图 11: LATX

LATX 是直接从 x86 到 LoongArch 的指令翻译引擎。

设备虚拟化

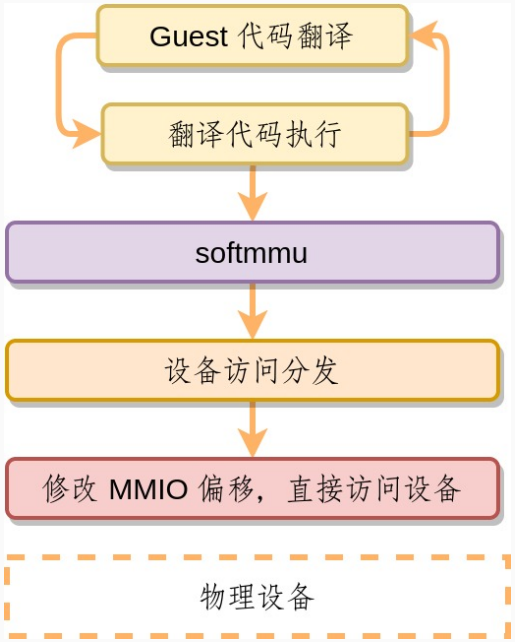


图 12: BMBT 设备虚拟化

包括 PCIe 在内的大多数设备都是架构无关的，BMBT 无需实现设备复用，因此可以让 Guest 直接操控设备，实现设备直通。

设备虚拟化: PCIe 设备虚拟化

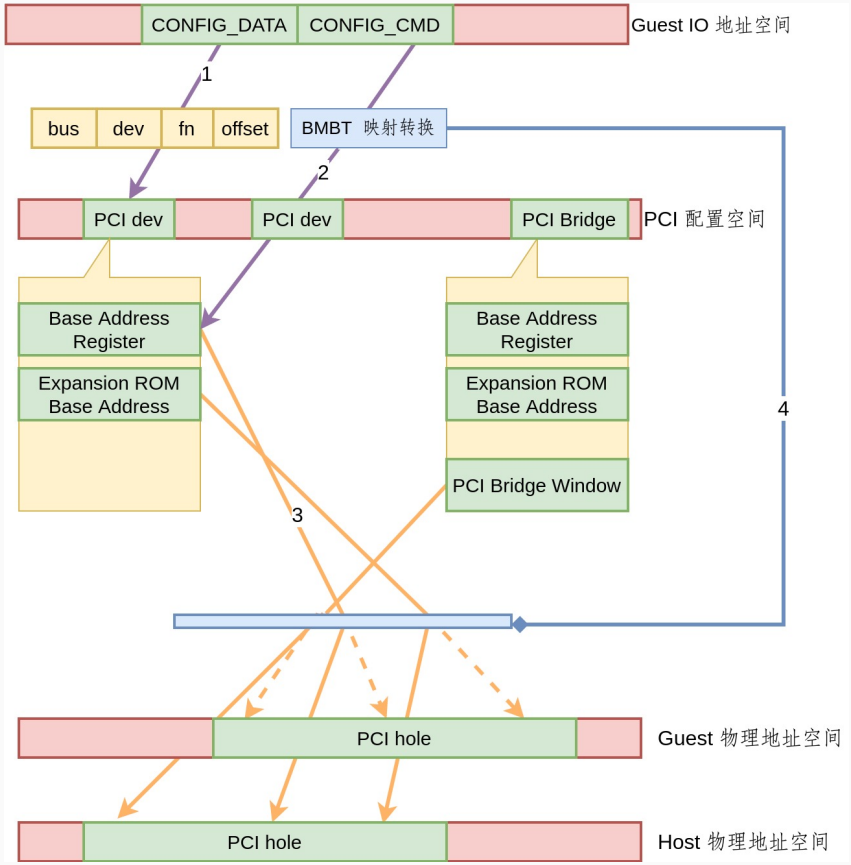
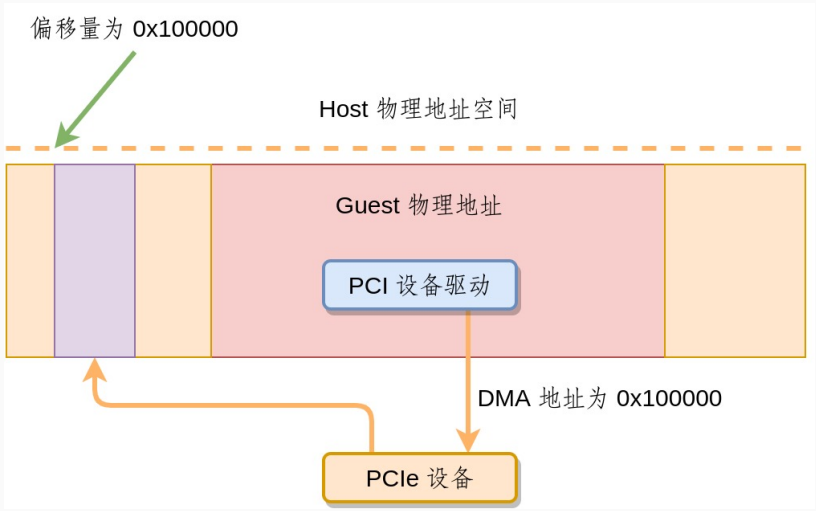
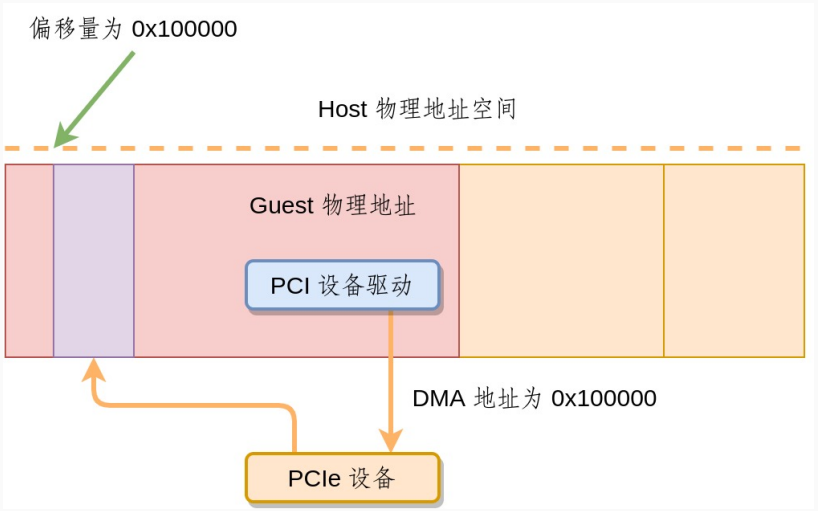


图 13: PCIe 设备直通

LoongArch 和 x86 对于配置空间和 mmio 空间的范围有不同的规定，因此需要进行偏移修正。

设备虚拟化：设备直通中的 DMA





左图中，没有保证 Guest Physical Address （简称 GPA）和 Host Physical Address （简称 HPA）相等，导致设备将数据拷贝到错误的位置。

中断虚拟化

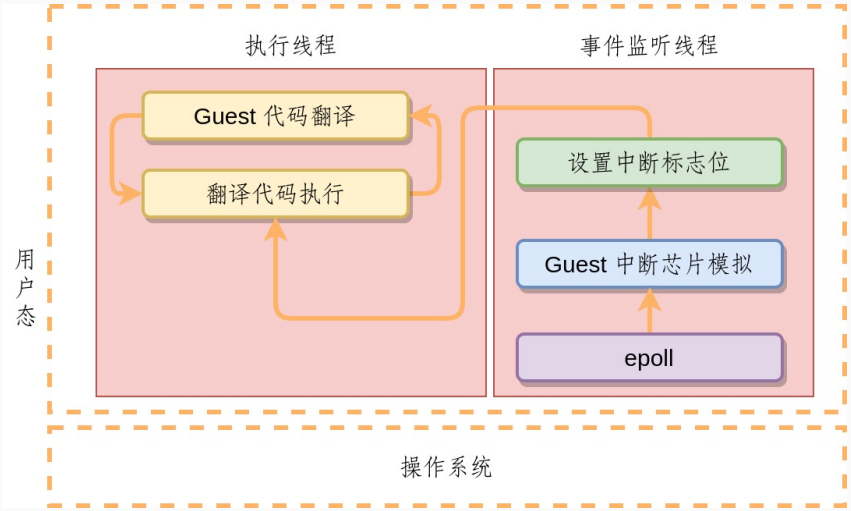


图 14: QEMU 中断软件架构

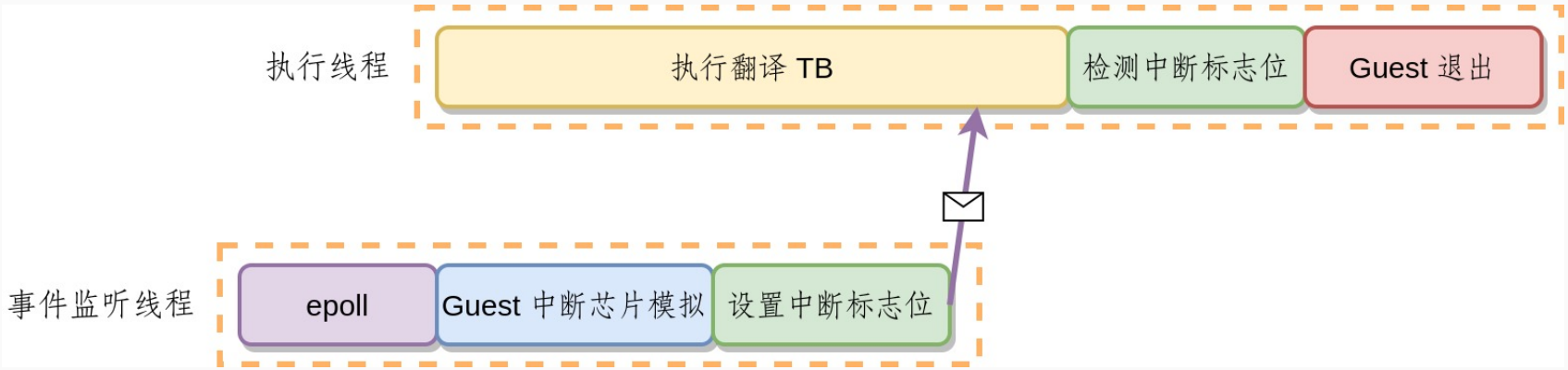
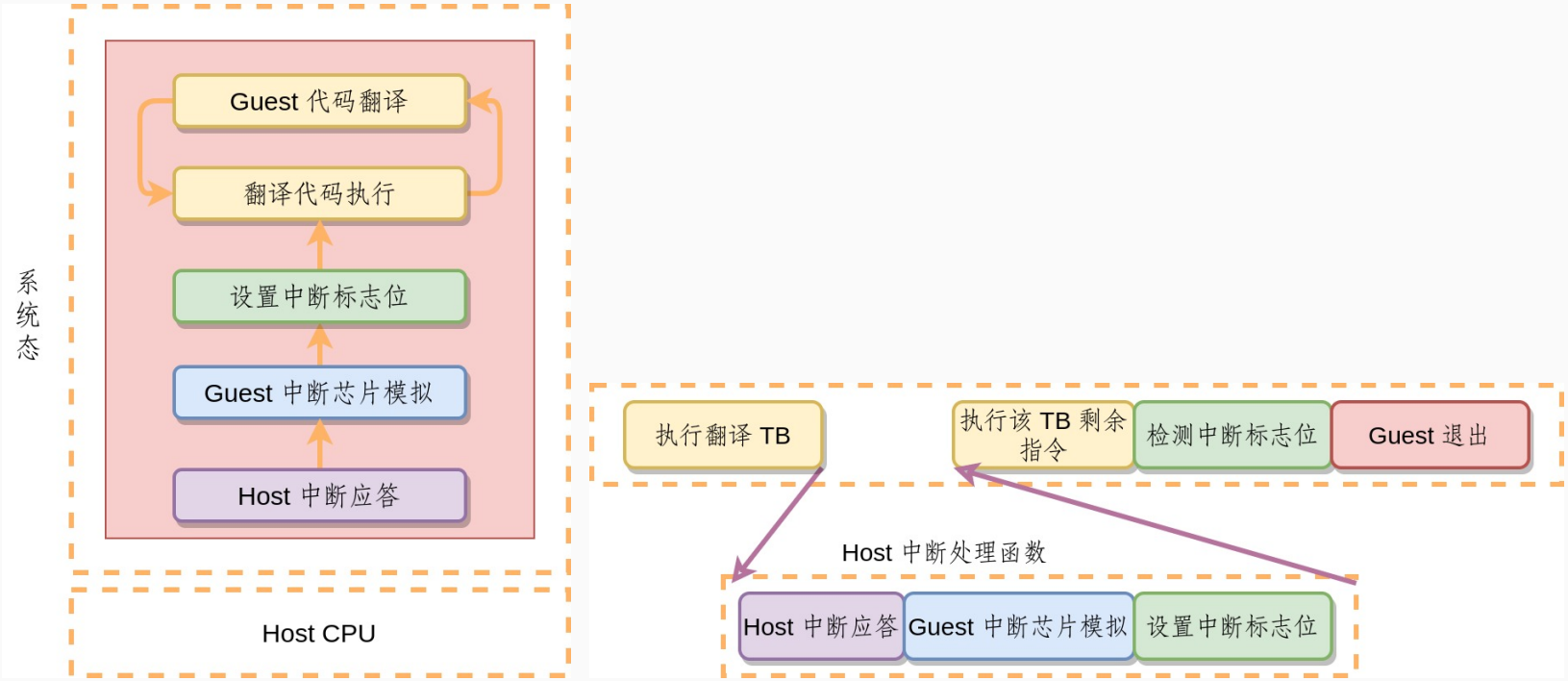


图 15: QEMU 中断执行流程

中断虚拟化



在裸金属上，CPU 接收中断后可以直接注入，无需额外的线程来进行事件监听。

中断虚拟化

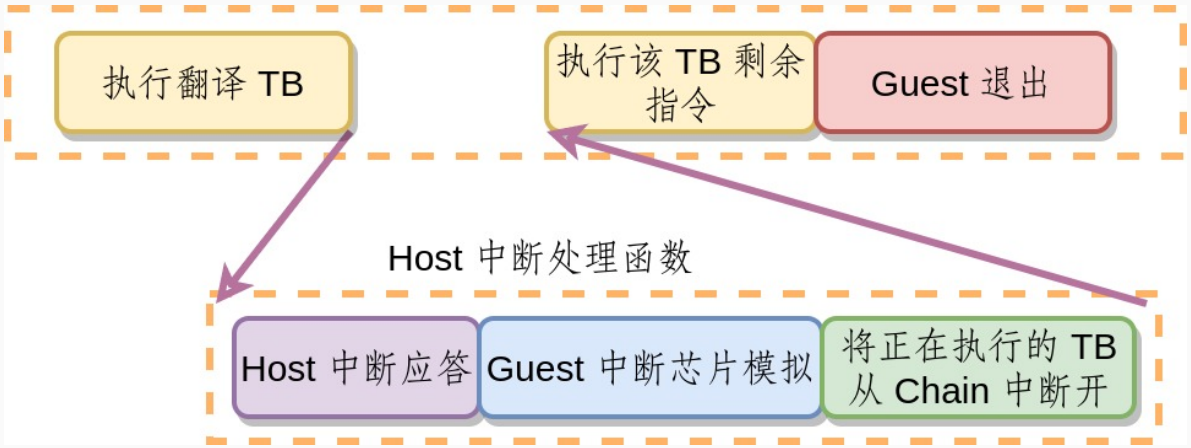


图 16: 消除中断标志位检测的执行流程

在中断处理函数中将正在执行的 TB 从 TB Chain 中断开，可以消除额外的中断标志位检测指令。

内存虚拟化: TLB miss 的消除

图 17: 进程级二进制翻译地址空间

将二进制翻译器和 Guest 的数据放到一个地址空间中，此时 GVA 等于 Host Virtual Address （简称 HVA），Guest 的访存进行的装换为: HVA -> HPA 。

内存虚拟化: TLB miss 的消除

图 18: 运行在 Linux 上的系统级二进制翻译器的地址空间

通过 mmap 映射出来一块 Host 虚拟地址空间给 Guest 当物理内存使用。

内存虚拟化: TLB miss 的消除

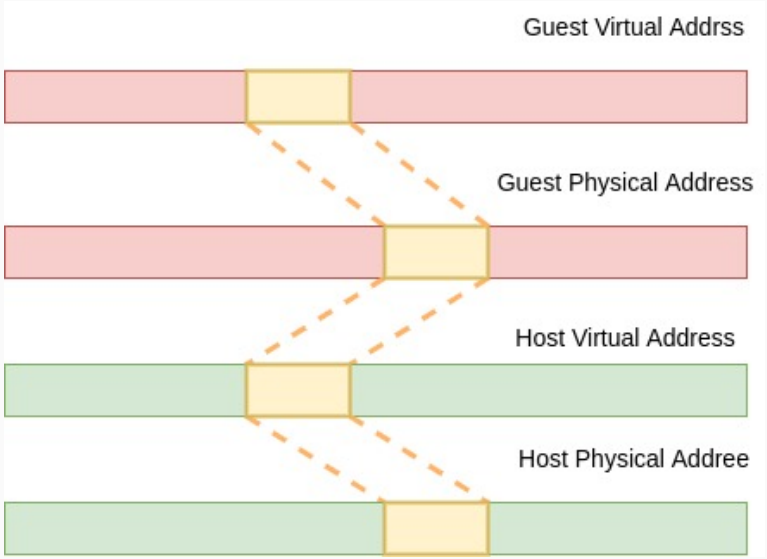


图 19: QEMU 中 Guest 的地址翻译过程

一条 Guest 访存指令需要进行的地址装换为: GVA -> GPA -> HVA -> HPA 。

内存虚拟化: TLB miss 的消除

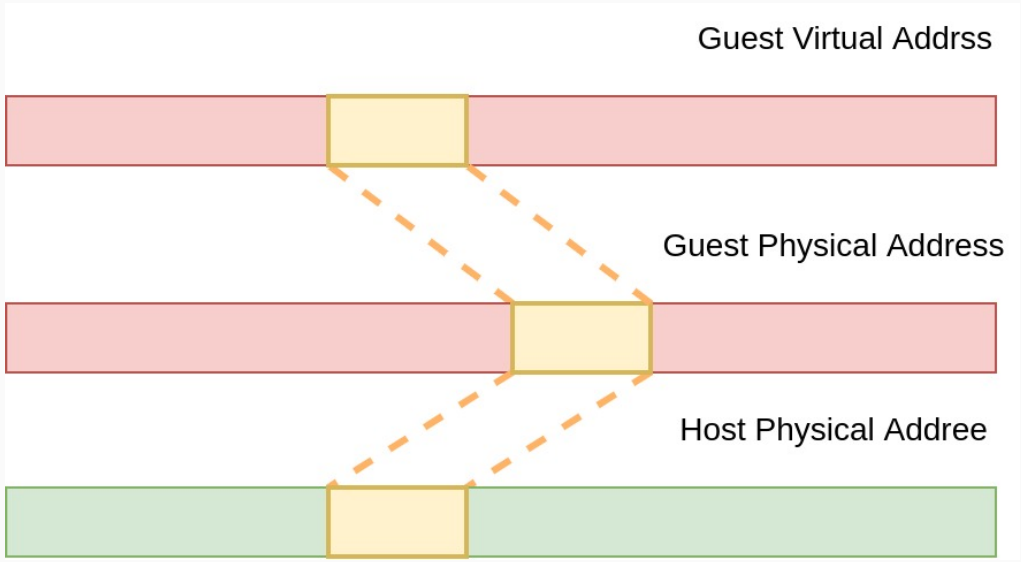


图 20: BMBT 中 Guest 的地址翻译过程

BMBT 直接运行在裸金属的环境中，可以消除掉从 HVA 到 HPA 的转换 [1] ，也就是说可以消除所有的 TLB miss 。

[1] LoongArch 支持直接映射窗口, x86 可以关闭页表映射

内存虚拟化：物理内存分配器

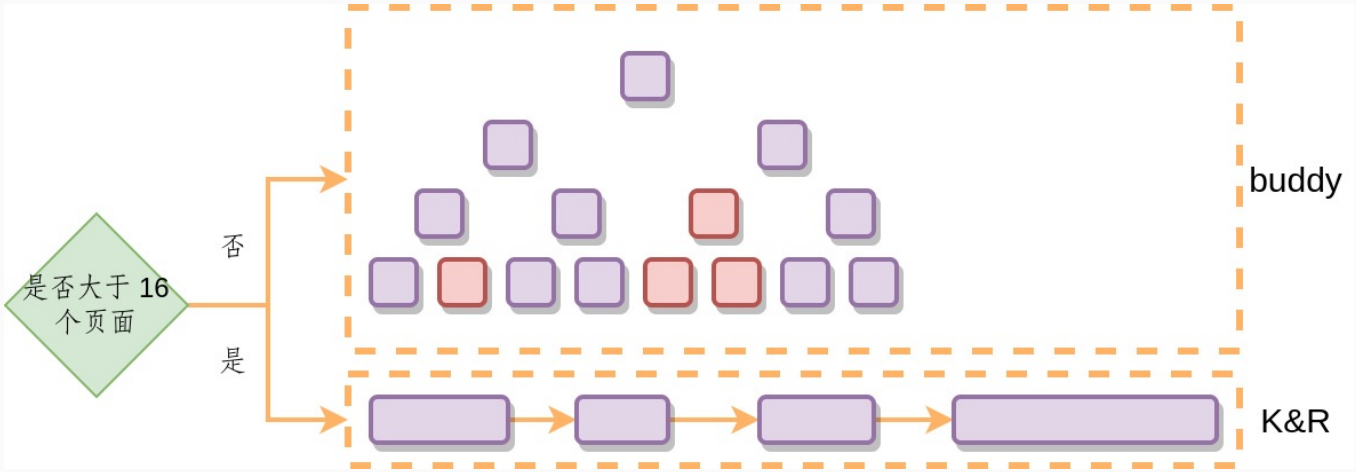


图 21：混合物理内存分配器

现代操作系统为了满足各种用户程序的需求，其物理内存分配器被设计的非常复杂。在 BMBT 中物理内存分配的模式单一，因此 BMBT 混合 K&R malloc 算法和伙伴算法，在防止碎片化的前提下保证了分配效率。

内存虚拟化: 硬件 TLB 加速

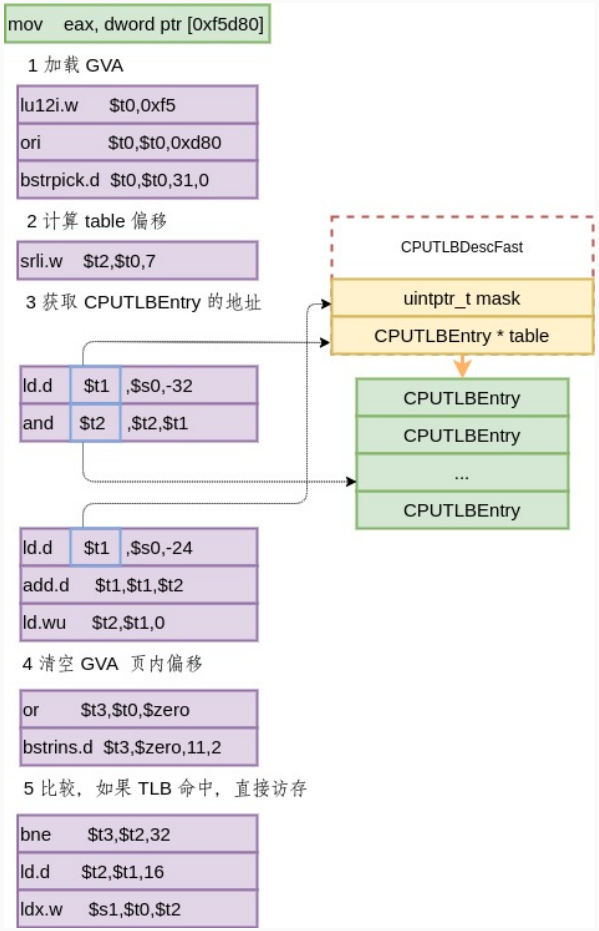


图 22: QEMU 中访存指令的翻译过程

一条 x86 访存指令需要 14 条 LoongArch 指令模拟。

内存虚拟化: 硬件 TLB 加速

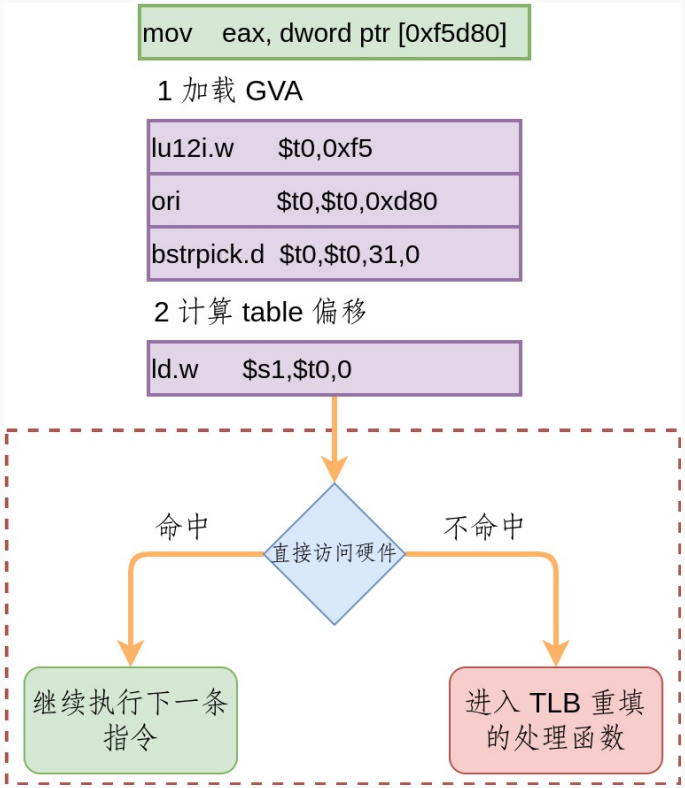


图 23: 硬件 TLB 加速访存后的翻译过程

一条 x86 访存指令被翻译为 4 条 LoongArch 指令。

内存虚拟化: 硬件 TLB 加速

访问硬件 TLB 有三种方法:

1. 在虚拟机中执行，例如 Captive 。
2. 在 Dune[2] 中执行。
3. 直接在裸机中执行。

前两种方法都引入了硬件辅助虚拟化的开销。

[2] Adam Belay et al. "Dune: Safe user-level access to privileged {CPU} features". In: *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*. 2012, pp. 335–348.

BMBT：裸金属二进制翻译器

CPU

- 使用 LATX 作为指令翻译引擎

设备

- 大多数设备可以直通，无需模拟

中断

- 无需另一个线程事件监听，直接注入中断
- 可以消除掉每一个 TB 中额外的中断标志位检测指令

访存

- 消除了所有的 TLB miss
- 可以直接访问硬件 TLB 来加速 Guest 访存

性能测试

BMBT 性能测试

CPU

- SPEC 2000 定点提升 21%，浮点提升 35.31%

IO 性能

- dd 拷贝测试中，bs 等于 1G 和 4k 的情况下分别快 83% 和 22%
- fio 写操作快 189.23%
- ping 延迟显著降低

访存

- memcpy 快 28.67%

展望

进一步的优化

CPU

- 支持多核

设备

- 压缩设备访问的代码路径

中断

- 消除额外的中断检测指令

访存

- 集成硬件 TLB 加速 Guest 访存

现有工作的完善

- 集成性能分析工具。
- 支持 amd64 。
- 支持 Windows 操作系统。
- 支持 ACPI 。
- 支持 UEFI 固件。
- 支持 IOMMU 。