AOSCC 2025 | 欢迎

路虽远，行则将至；事虽难，做则必成。

跨架构UEFI启动新范式: MultiArchUefiPkg – LoongArch64 移植实践

分享者: edk2-reviewer MarsDoge

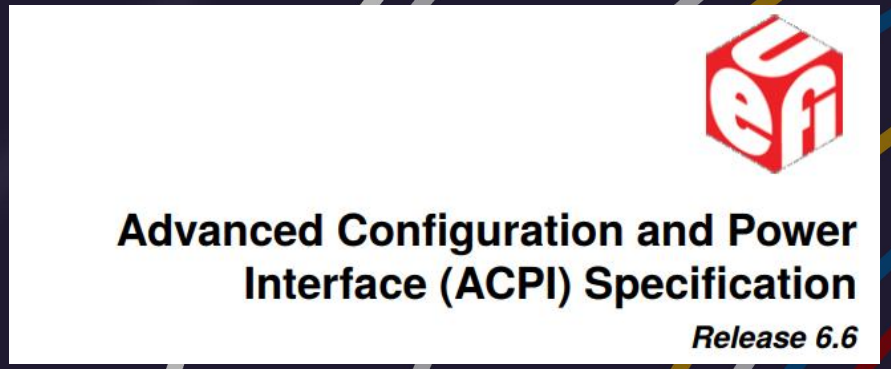
Email: mars@dogexorg.com

What? UEFI?

https://uefi.org/

UEFI（Unified Extensible Firmware Interface）

- 是一种开放的、跨平台的固件接口标准
- 替代传统的BIOS，负责从开机到操作系统加载前的初始化过程
- 由 UEFI Forum（一个国际标准组织）维护
- 定义了多个规范，主要有：
  > UEFI Specification（接口规范）
  > PI Specification（Platform Initialization，平台初始化规范）
  > ACPI Specification（高级配置和电源接口规范）



UEFI 规范管与不管?

管什么？（规范层面要定义 / 约束的项）；不管什么？（实现层面可任意选型 / 自由度大的项）；

- 约束输出格式与ABI // PE: Portable Executable COFF: Common Object File Format

输出 PE/COFF，可重定位； 必须符合 EFI ABI（参数传递、结构对齐、宽字符_wchar 2B）；

- “接口”而非”实现”

规范把固件在某一阶段之后暴露给OS加载器的一切接口全都标准化，但不规定固件内部怎样初始化硬件。

- 前置阶段交给PI/ACPI等规范

诸如开机自检、锁Cache、建立堆栈、如何进入C环境、内存以及IO总线训练、初始化处理器/芯片组/板卡定制化外设，再到USB/PCIE等通用总线驱动再到设备驱动、 ACPI 表格内容，都由各自的独立规范定义；

- 轻量级插拔框架 - 由 Handle、GUID、Protocol 三元组构成

某个Handle(实体)上，安装了某种类型(GUID)的协议，其具体实现是某个结构体(Protocol)；

- Boot Manager 仅给“骨架”

规范列出必须支持的 NVRAM 变量和启动条目格式，但把 UI、策略等留给OEM/IBV自由发挥，以便做品牌差异化。

- 退出 Boot Services 之后就“放手”

当 OS 调用 ExitBootServices()，UEFI 所有 Boot Services 立即失效，除保留Runtime Services常驻内存；此后的电源管理、设备驱动等均由 OS 自己掌控，规范不再干预。

UEFI 的分阶段执行:

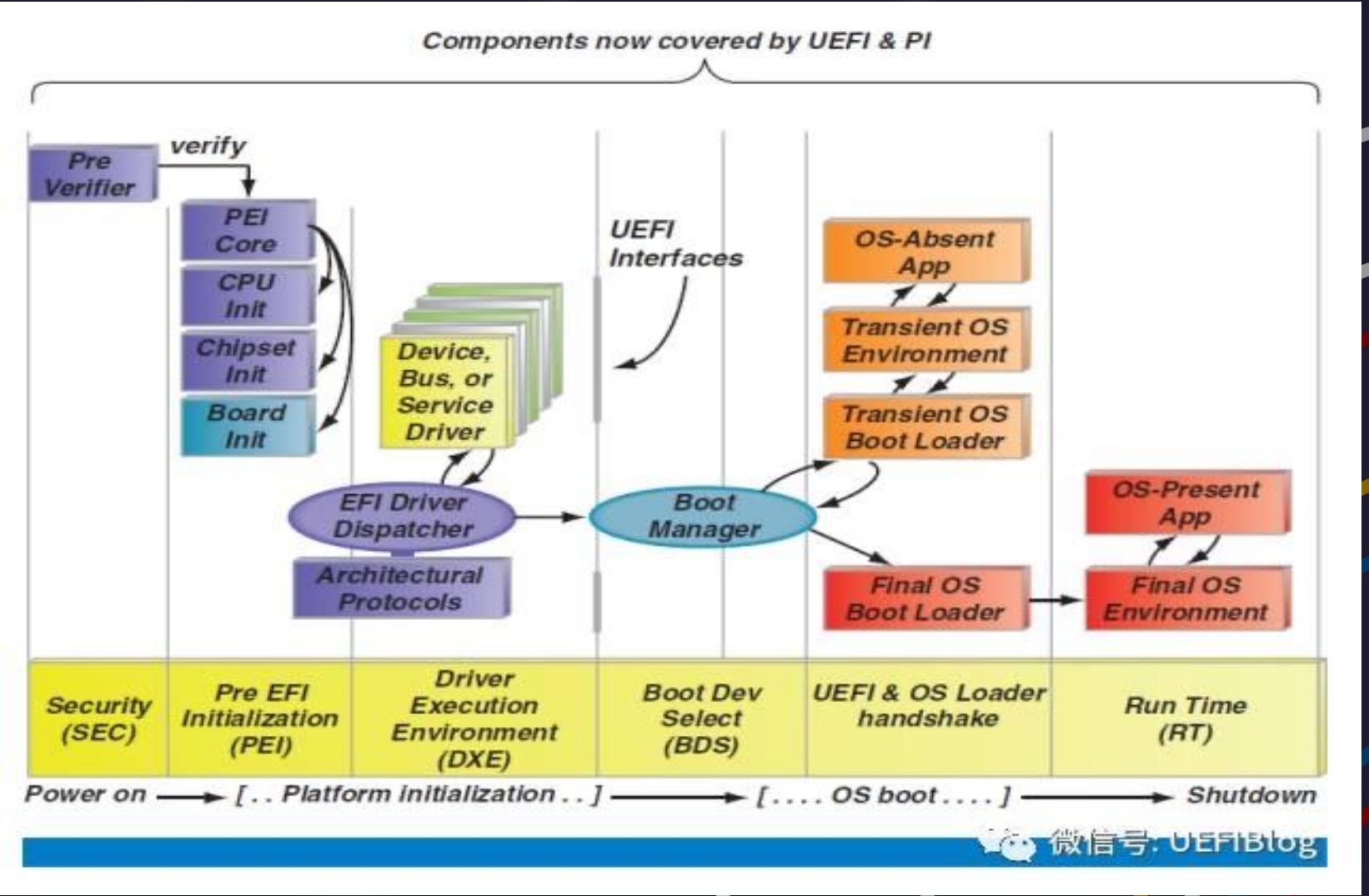
> SEC（Security Phase）
上电后最早执行的阶段，CPU的复位向量即是SEC的第一条指令，一般存储一条汇编跳转；
职责：初步初始化CPU，建立临时堆栈（一般在Cache-as-RAM上）；

> PEI（Pre-EFI Initialization）
职责：POST自检的核心阶段，主要负责初始化内存(DRAM)，初始化南北桥，构造 HOB（Hand-Off Blocks），为后续阶段传递信息；

> DXE（Driver Execution Environment）
职责： 驱动加载和协议安装，安装各种UEFI协议（如 PCI I/O、USB、SATA、Nvme等），建立 Handle-Protocol 体系；

> BDS（Boot Device Selection）
职责： 扫描引导设备（硬盘、网络、USB等），执行启动策略，选择操作系统引导项；

> TSL（Transient System Load）/ RT
职责：将控制权交给操作系统，只保留运行时服务常驻内存；



What? TianoCore?

https://www.tianocore.org

社区组织：托管在 GitHub（github.com/tianocore），原由 Intel 发起并由多家厂商和开发者共同维护；

将 Unified Extensible Firmware Interface (UEFI) 相关规范代码实例化；

What? EDKII?

https://github.com/tianocore/edk2/

EDK2（EFI Development Kit II）

- 是 UEFI 规范开源代码实现主框架
- 由 TianoCore项目维护（UEFI Forum 社区支持）

> 为什么叫 EDKII?

第一代 EDK 早期仅支持 Intel 平台且构建系统陈旧，
EDK2 从 2009 年起重构为跨架构、跨工具链的新框架；

> edk2-platforms 与 edk2 的区别?

edk2 提供“通用模块”；edk2-platforms 收集针对具体主板/SoC的开源PlatformPkg（硬件初始化代码），
分仓以便精简主线，早期虚拟机固件来自于此，目前已转移到edk2/OvmfPkg主仓库;

三者之间的关系是什么?

UEFI 是标准 → TianoCore 是社区 → EDK2 是社区里的核心代码库，用来实现并演示 UEFI 规范。

UEFI 规范（UEFI Spec & PI Spec）

└──► 任何厂商 / 项目都可实现

├──► 商业闭源实现（AMI/Insyde/ASUS/Phoenix/国产百敖/…）IBV: Independent BIOS Vendor（独立 BIOS 提供商）

└──► 开源实现 = TianoCore 项目  参考Edk2基本框架实现私有协议以及UI框架

├──► 核心代码 = **EDK2** 主仓库，包含所有规范实现框架和基础库 （许可证License：采用 BSD 2-Clause开源许可证;

├──► 示例平台 = 虚拟机固件OVMF (x86/RISC-V/LoongArch64)、ArmVirtPkg …

├──► 扩展仓库 = edk2-platforms(开源板卡级代码);

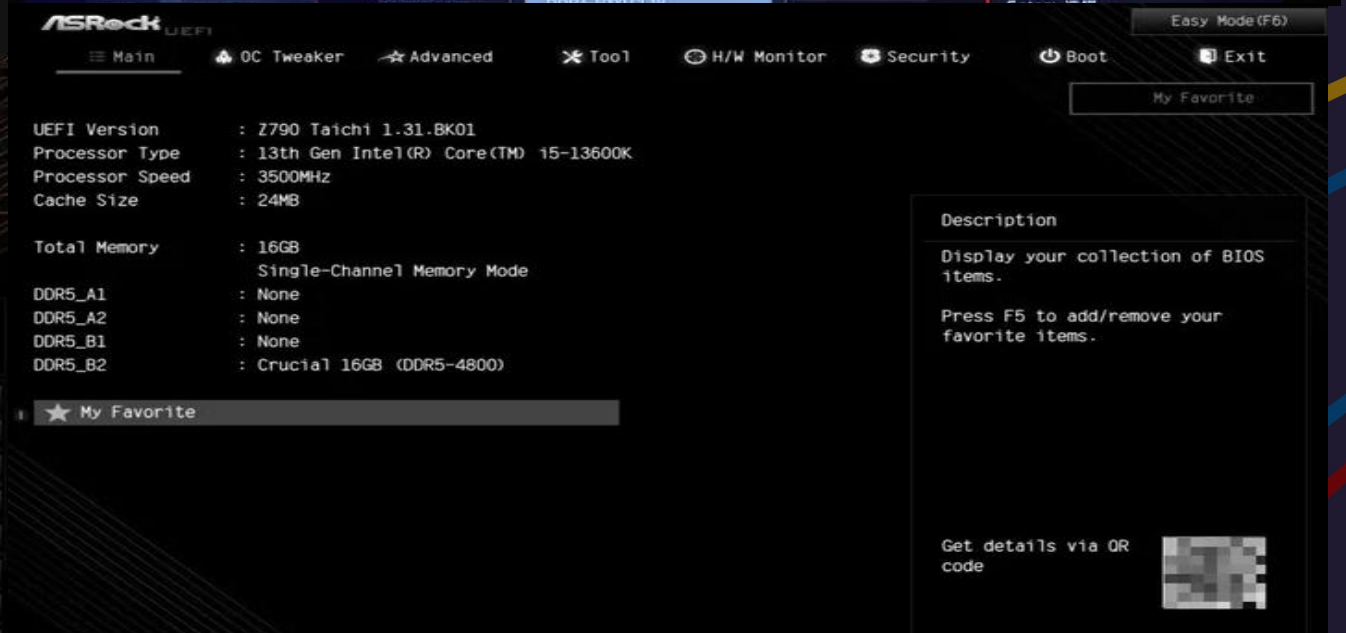
└──► 辅助工具链 = edk2-libc、edk2-test ...

典型IBV的UEFI

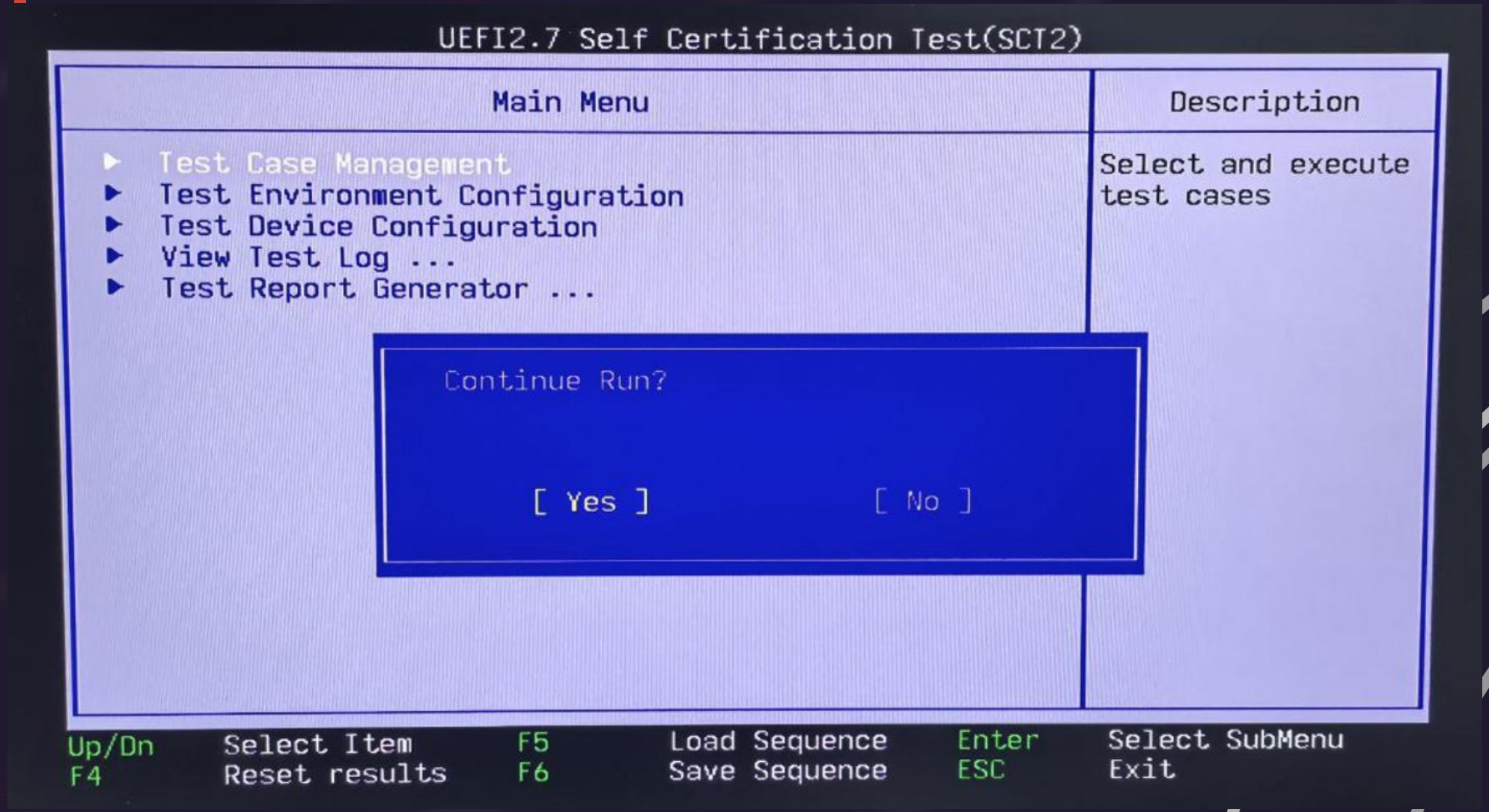

ASUS UEFI BIOS Utility – EZ Mode
01/13/2023 Friday 12:24
主机配置信息
PRIME Z790M-PLUS D4 BIOS Ver. 0809
13th Gen Intel(R) Core(TM) i5-13600KF
CPU频率: 5100 MHz
内存频率: 32768 MB (DDR4 3200MHz)
CPU温度
34°C
CPU电压
1.261 V
主板温度
28°C
内存信息
DIMM_A1: N/A
DIMM_A2: KingBank 16384MB 3200MHz
DIMM_B1: N/A
DIMM_B2: KingBank 16384MB 3200MHz
硬盘及外置存储信息
AHCI:
SATA6G_4: ST3000DM008-2DM166 (3000.5GB)
NVME:
M.2_1: Netac NVMe SSD 2TB (2000.3GB)
性能模式切换
平衡
启动设备顺序
切换全部
Windows Boot Manager (M.2_1: Netac NVMe SSD 2TB) (2000.3GB)
X.M.P.
Disabled
Intel快速存储技术(Intel®RST)
On Off
风扇信息
CPU 风扇 761 RPM
CPU额外风扇 2280 RPM
机箱风扇1 N/A
机箱风扇2 N/A
机箱风扇3 N/A
水泵 3199 RPM
风扇
风扇转速设置
选择启动设备(F8)


MSI CLICK BIOS 5
20:26 星期日 18 4月, 2021
CPU Speed 3.70 GHz
DDR Speed 2133 MHz
GAME BOOST
CPU
XMP Profile 1
XMP Profile 2
Motherboard settings
SETTINGS
OC
Overclocking
Memory Try It!
MB: MAG B560M MORTAR WIFI (MS-7D17)
CPU: Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz
内存容量: 16384MB
BIOS版本: E7D17IMS.107
BIOS构建日期: 01/11/2021
HELP
INFO
开启Memory Try It!功能可以通过选择最佳化的内存预设值,改善内存的相容性及效能。


EASY MODE
03/30/2021 Tuesday 18:02
AORUS
MB: Z590 AORUS XTREME
BIOS Ver. F6
CPU: 11th Gen Intel(R) Core(TM) i9-11900K @ 3.50GHz
RAM: 16GB
CPU 频率 5301.21 MHz
处理器温度 52.0 °C
CPU Voltage 1.464 V
内存频率 3200.00 MHz
内存电压 1.347 V
VRM MOS
DDR4_A1: G.SKILL 8GB 2133MHz
DDR4_A2: N/A
DDR4_B1: G.SKILL 8GB 2133MHz
DDR4_B2: N/A
X.M.P. - DDR4-3600 18-22-22-42-64-1.35
X.M.P. Profile1
SATA PCIE M.2
P3: ST31000524AS (1000.2GB)
Intel® 快速储存技术
ON OFF
开机顺序
UEFI: KingstonDataTraveler 3.0PMAP, Partition 1 (KingstonDataTraveler 3.0PMAP)
SAMSUNG MZVLB512HAJQ-00000
CT2000P5SSD8
Smart Fan 6
CPU_FAN 1134 RPM
CPU_OPT N/A
SYS_FAN1 N/A
SYS_FAN2 N/A
SYS_FAN3 N/A
SYS_FAN4 N/A
SYS_FAN5_PUMP N/A
SYS_FAN6_PUMP N/A
SYS_FAN7_PUMP N/A
SYS_FAN8_PUMP N/A
中文 (简体)
说明
进阶模式 (F2)
Smart Fan 6 (F6)
载入出厂预设值 (F7)
Q-Flash (F8)
储存并离开 (F10)


ASRock UEFI
Easy Mode(F6)
Main
OC Tweaker
Advanced
Tool
H/W Monitor
Security
Boot
Exit
My Favorite
UEFI Version : Z790 Taichi 1.31.BK01
Processor Type : 13th Gen Intel(R) Core(TM) i5-13600K
Processor Speed : 3500MHz
Cache Size : 24MB
Total Memory : 16GB
Single-Channel Memory Mode
DDR5_A1 : None
DDR5_A2 : None
DDR5_B1 : None
DDR5_B2 : Crucial 16GB (DDR5-4800)
My Favorite
Description
Display your collection of BIOS items.
Press F5 to add/remove your favorite items.
Get details via QR code

edk2 原生 – SCT(Self-Certification Test)



炫酷UI编译背后Loong目前所做的尝试

背景：部分图形引擎基于C++实现，裸机UEFI环境可以执行C++就很重要了；

UEFI 要求所有 EFI 模块必须是 PE/COFF（PE32 或 PE32+）格式

> Windows/MSVC 下：cl+link 能直接生成带有 CRT stub、全局构造表的 PE/COFF;

> GNU 环境下：

- 输出ELF（Executable and Linkable Format）格式:

  .init_array/.ctors //存放构造函数地址

  Linux系统 CRT startup 代码在程序启动时遍历调用：

```
for (func = __init_array_start; func < __init_array_end; func++)
    (*func)();
```

- 裸UEFI 仅用objcopy将ELF转换成PE/COFF格式(只是封装转换,
- 不做真正的链接)，换壳的PE即不带任何可执行的.CRT初始化代码，无法构造/析构；
- loong正在尝试的方案：在链接ELF时，通过lds将对应构造表重命名合并到 PE/COFF 要求的CRT段再去objcopy；

Why ? Nothing ?

Background

Typical non-x86 SBC

Off-the-shelf adapter (NIC/IPU, GPU, RAID)

Nothing

www.uefi.org

我：请问贵司的显卡在别的平台上试过吗?

客户：我们的显卡在Intel和AMD的主板上跑都没问题的。

我：那在ARM的机器上试过吗?

客户：需要吗?

我：不需要吗?

客户（一脸懵逼）：需要吗?

为了避免尴尬，我赶紧换了一个问题：请问贵司的显卡带ARM原生的Option ROM吗?

客户：我们的显卡在Intel和AMD的主板上跑都没问题的。

我：那就只有x86的，没有ARM的了?

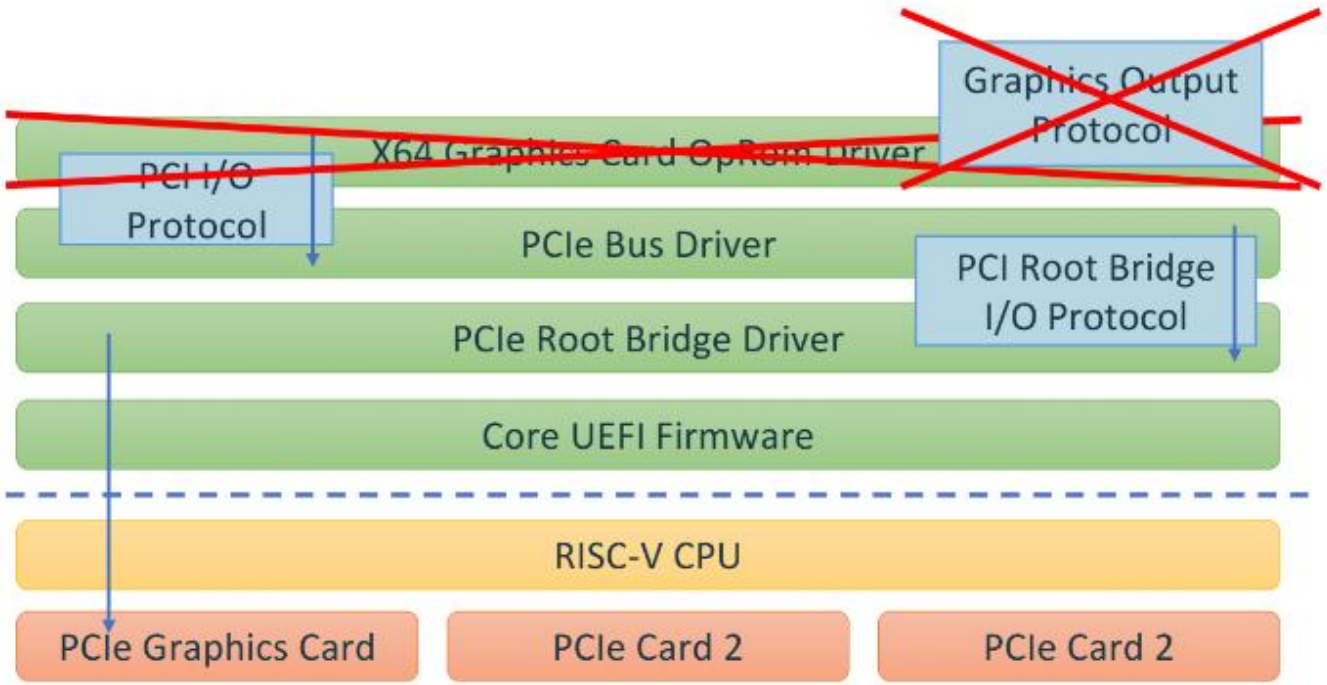
客户：需要吗?

我：不需要吗?

客户（一脸懵逼）：需要吗?

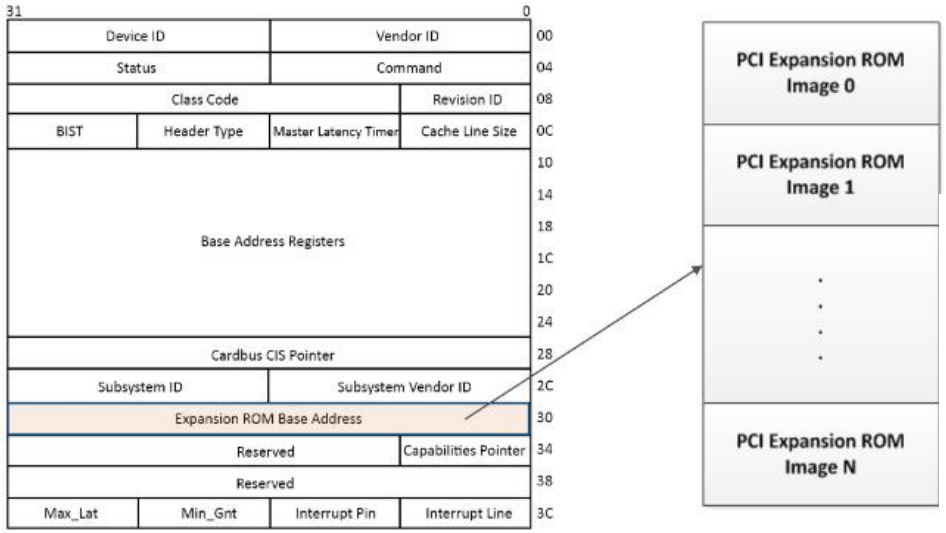
- 暂且不聊纯CPU模拟的EFI App形式的跨架构执行（这已然支持 **x64Shell.efi**
- 本次主要重点分享只存在x86二进制驱动的外设如何在非-x86 UEFI跑起来

How? 是怎样无事发生的呢？



How？PCI总线怎样加载外设驱动？

PCIe Firmware Drivers



Legacy PC-AT BIOS ROM

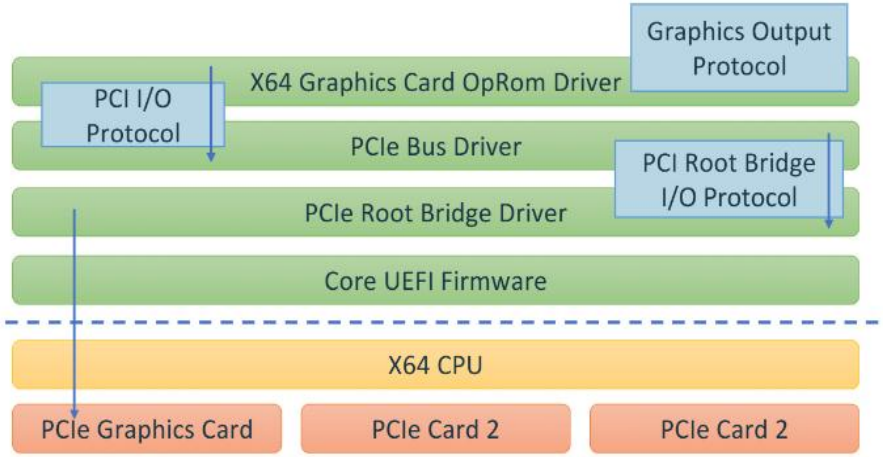
X64 UEFI Driver

符合PCIE规范: 配置空间的一扇门(Expansion ROM BAR)来获取存于外设ROM上的BIN程序(可同时包含多个镜像块, 但通常只存在x86驱动二进制);

OptionROM单块: 0x55AA起始，512B对齐;

Life of PCIe in UEFI

硬件 → 根桥驱动 → 总线驱动 → OpROM 驱动 → 图形协议



Why？为什么驱动存于卡上且闭源？

- 主因: 商业与知识产权: 硬件内部寄存器、DSP/FPGA 微码、授权 IP等;
- 代码体积 & 环境限制: 主Flash容量宝贵，不同类型不同厂商卡专有硬件细节驱动五花八门, 更新路径独立;
- 安全签名链：开源安全启动谁来签名?

UEFI下需求的典型外设功能驱动:

- 独立显卡: Nvidia / AMD GPU EDID获取送显点亮;
- RAID阵列卡: Broadcom JBOD/RAID0/1/5等模式;
- 1000M/10G/25G/40G/100G网卡：Intel、Marvell 网络PXE引导;
- 一些国内新IP外设们们.....

MultiArchUefiPkg的前世今生？

前世：https://github.com/ardbiesheuvel/X86EmulatorPkg

目的: 为了Aarch64平台EFI环境可以广泛兼容x64生态；

Ard：为 Tianocore/EDK2 实现了一个 DXE 驱动程序，允许为x86_64构建的UEFI驱动程序在 64 位 ARM 系统（又名 AArch64）上执行。

技术：Not UEFI in Qemu, Qemu in UEFI;

今生：https://github.com/intel/MultiArchUefiPkg

Intel借鉴 unicorn 模拟器引擎移植efi基础库进行重构X86EmulatorPkg -> MultiArchUefiPkg;

引入双向”Thunk/Wrapper”机制，实现Native->Emulator、Emulator->Native;

实现 RISC-V EFI环境运行X86/AArch64 EFI驱动及应用程序（猜测Intel异构RISC-V尝试；

前世: X86EmulatorPkg解决了什么？

How did the Arm ecosystem solve this?

X86EmulatorPkg

- ☑ Supports x64 OpRoms and UEFI applications on AArch64 systems.
  - Open Source UEFI Boot Service Driver
  - Targets 64-bit AArch64 systems (servers, workstations)
  - Developed by Linaro engineers 6 years ago.
  - Uses Qemu Tiny Code Generator for efficient translation of x64 to AArch64 code.
  - https://github.com/ardbiesheuvel/X86EmulatorPkg
- ✗ Not trivially portable to RISC-V!
  - Old TCG code of unknown provenance.
  - Backporting RISC-V support sounds hard (and time consuming) unless you're a Qemu guru.

优点：

- 支持了AArch64运行x64 OptionRom(通常只提供x86机器码)和 UEFI应用程序;
- 开源了该UEFI.BootServive驱动;
- 裁剪QEMU-TCG(Tiny Code Generator),高效把x64指令动态翻译成AArch64指令;

缺点：

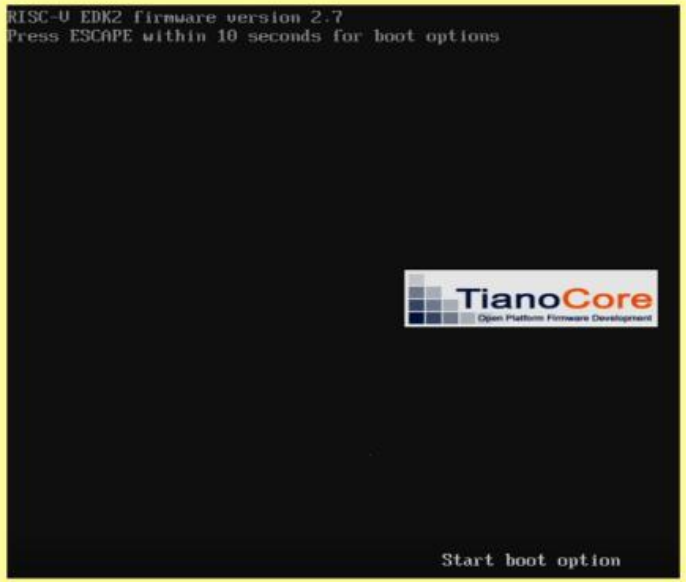
- 只关注AArch64服务器以及工作站;
- 依赖QEMU-TCG代码仓库陈旧，不容易移植到RISC-V，意味着接口变化多，需要大量重写;

今生: MultiArchUefiPkg?

MultiArchUefiPkg

Rewrite of X86EmulatorPkg

- Portable: Supports AArch64 and 64-bit RISC-V UEFI hosts.
- 64-bit x64 and AArch64 UEFI Boot Service emulation.
- Clean: Abstracts Qemu/TCG with Unicorn Engine API.
- https://github.com/intel/MultiArchUefiPkg
- RISE Project in the Firmware WG
- Correctness, perf, size.

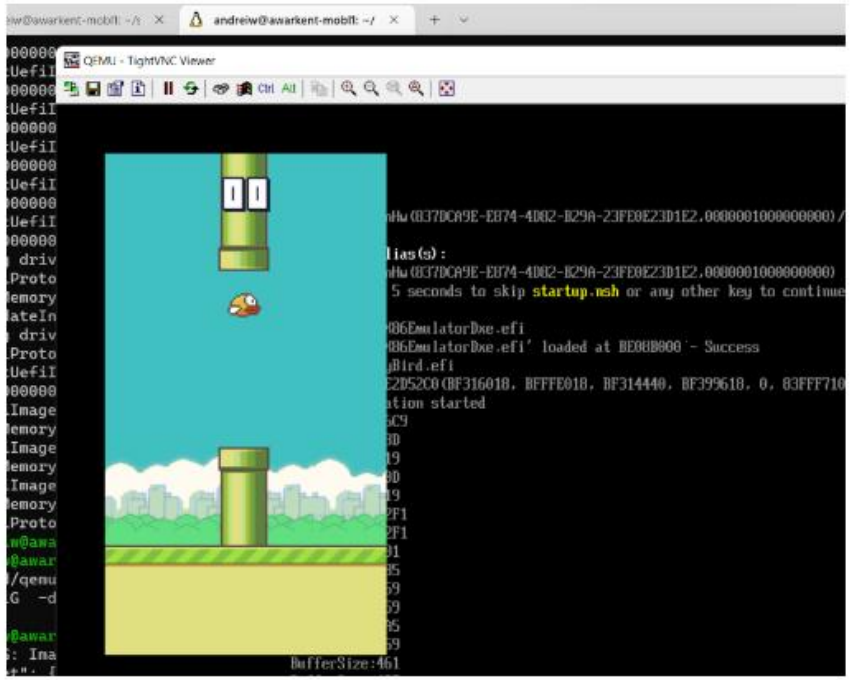



```
7D 0000000A B - -   1   6 Usb Bus Driver                          UsbBusDxe
7E 0000000A D - -   1   - Usb Keyboard Driver                     UsbKbDxe
7F 00000011 D - -   1   - Usb Mass Storage Driver                 UsbMassStorageDxe
82 00014300 B - -   1   1 AMD GOP X64 Release Driver Rev.1.67 Offset(0x10000,0x1E5FF)
Shell> _
```

- 架构可继续扩展: 支持 X86/AArch64/RISC-V/LoongArch64 互EFI环境模拟执行;
- 更干净: 借助中间层Unicorn Engine API 抽象 QEMU-TCG, 不再依赖老旧QEMU;
- 双向”Thunk/Wrapper”机制，实现Native->Emulator、Emulator->Native的切换;
- 结构层次化，代码开源，文档完善，自带测试套件及CI等配套基础设施;

MultiArchUefiPkg 演示:

MultiArchUefiPkg



HOST ISA: AArch64 UEFI　Guest ISA: X86 EFI App（Game

How？UEFI是怎样装载非原生驱动？

UEFI_DXE阶段所接触的驱动类型:

UEFI_DRIVER: 可插拔式设备驱动，类似内核模块;

DXE_DRIVER: 系统级驱动，类似内置驱动（built-in driver）;

Application: 非常驻内存，执行完退出，不提供Protocol, 可被Shell/BootManager启动;

PCI根驱动深度优先扫描分配各自总线号写入各自下游配置空间，并为每个设备安装PCI IO;

再为其分配BAR地址时，会一并扫描是否有扩展BAR, 即:

OptionRom驱动从外设内按照PCIe协议访问并识别非原生PE再加载到内存，记录ImageRecord并为其安装影子BS表，并将页表内该段内存设置为NX (No Exec);

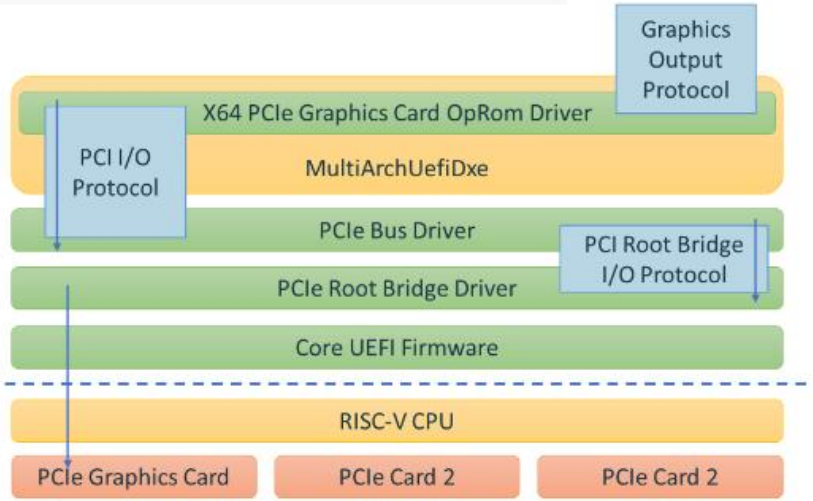
DXE_CORE提供 EDKII_PECOFF_IMAGE_EMULATOR_PROTOCOL 标准GUID接口, 模拟器驱动作为DXE_DRIVER加载通过HOOK形式安装协议接口的具体实例对应的OPS再等待执行回调;

How？UEFI是怎样执行非原生驱动？

How it works

- Possible entirely due to narrowly-defined EFI ABI
- Models Boot Services environment, with certain services filtered or disabled.
- Tiano support for foreign binaries - **EDKII_PECOFF_IMAGE_EMULATOR_PROTOCOL**
- Emulation is only interesting if thunking goes both ways!
  - RISC-V No-Execute handler traps for native → emulated.
  - Unicorn No-Execute handler traps for emulated → native.

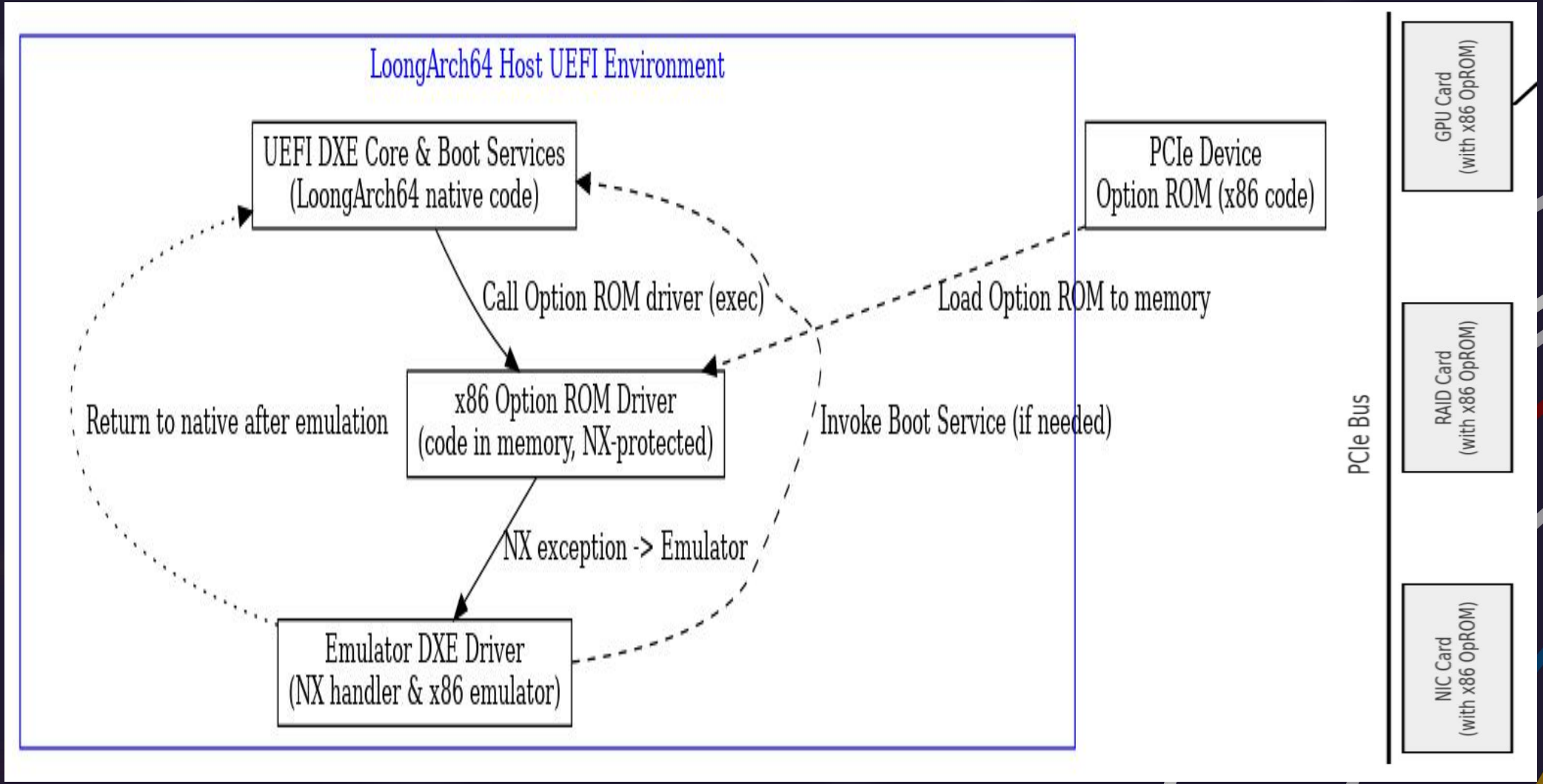
```
UINT64
EFIAPI
Fn(UINT64, UINT64, UINT64, UINT64,
   UINT64, UINT64, UINT64, UINT64,
   UINT64, UINT64, UINT64, UINT64,
   UINT64, UINT64, UINT64, UINT64);
```



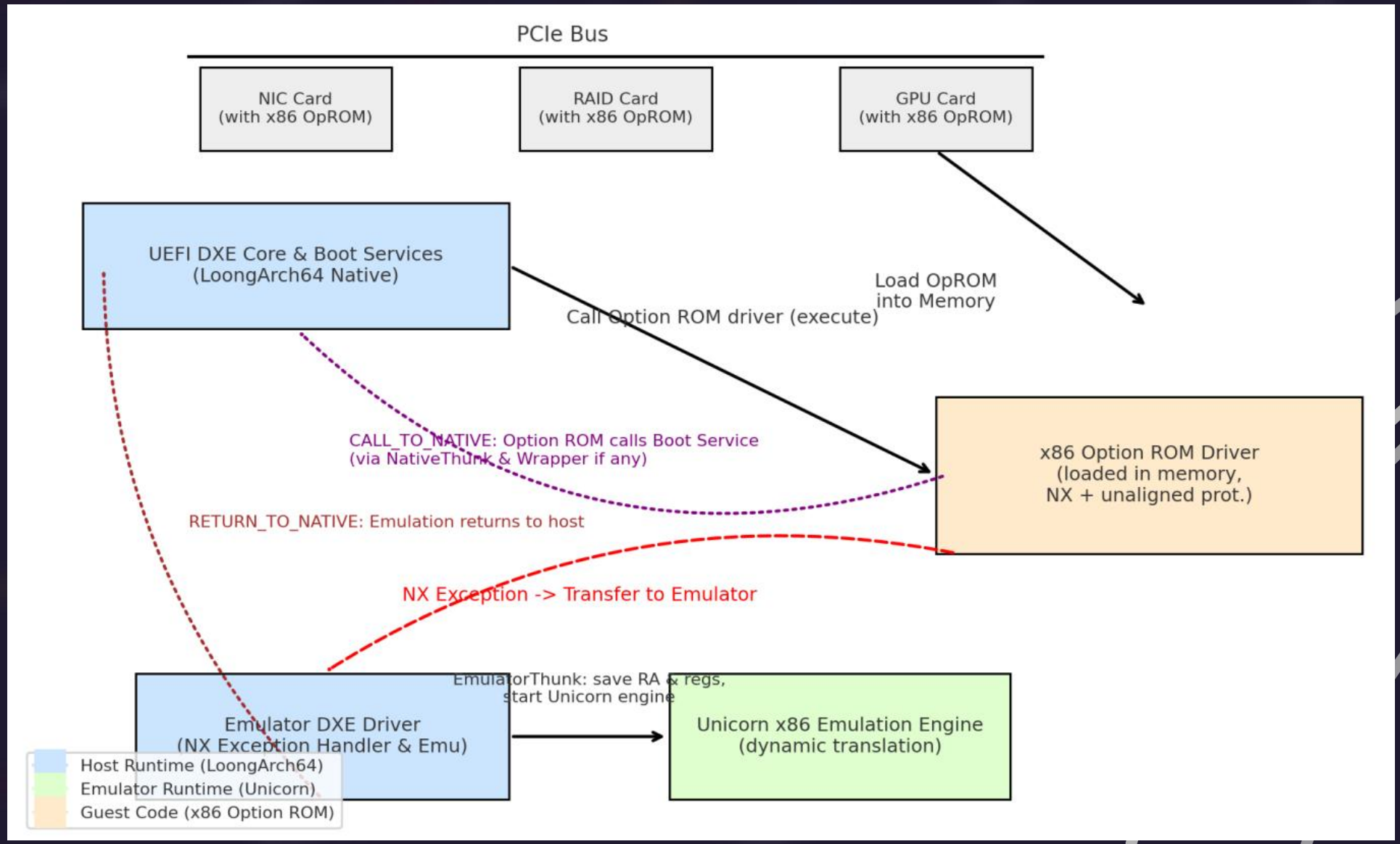
1. 完全依赖明确、严格定义的EFI ABI（Application Binary Interface）
2. 模拟了UEFI BS（Boot Service） 环境
3. Native -> emulated
   Emulated -> Native

简概：通过“双向thunk机制”：在PC去Guest代码时触发事先注册的TLB异常，异常处理内保存Context 并让EPC 指向 EMU的CpuRunFunc，当调用BS服务时会通过事先仿真的Wrapper/Thunk让模拟器抛出控制权回Native的真BS.

EFI模拟执行框架：


LoongArch64 Host UEFI Environment
UEFI DXE Core & Boot Services
(LoongArch64 native code)
PCIe Device
Option ROM (x86 code)
Call Option ROM driver (exec)
Load Option ROM to memory
x86 Option ROM Driver
(code in memory, NX-protected)
Return to native after emulation
Invoke Boot Service (if needed)
NX exception -> Emulator
Emulator DXE Driver
(NX handler & x86 emulator)
PCIe Bus
GPU Card
(with x86 OpROM)
RAID Card
(with x86 OpROM)
NIC Card
(with x86 OpROM)

EFI模拟三部分协同：


PCIe Bus
NIC Card
(with x86 OpROM)
RAID Card
(with x86 OpROM)
GPU Card
(with x86 OpROM)
UEFI DXE Core & Boot Services
(LoongArch64 Native)
Load OpROM
into Memory
Call Option ROM driver (execute)
x86 Option ROM Driver
(loaded in memory,
NX + unaligned prot.)
CALL_TO_NATIVE: Option ROM calls Boot Service
(via NativeThunk & Wrapper if any)
RETURN_TO_NATIVE: Emulation returns to host
NX Exception -> Transfer to Emulator
EmulatorThunk: save RA & regs,
start Unicorn engine
Emulator DXE Driver
(NX Exception Handler & Emu)
Unicorn x86 Emulation Engine
(dynamic translation)
Host Runtime (LoongArch64)
Emulator Runtime (Unicorn)
Guest Code (x86 Option ROM)

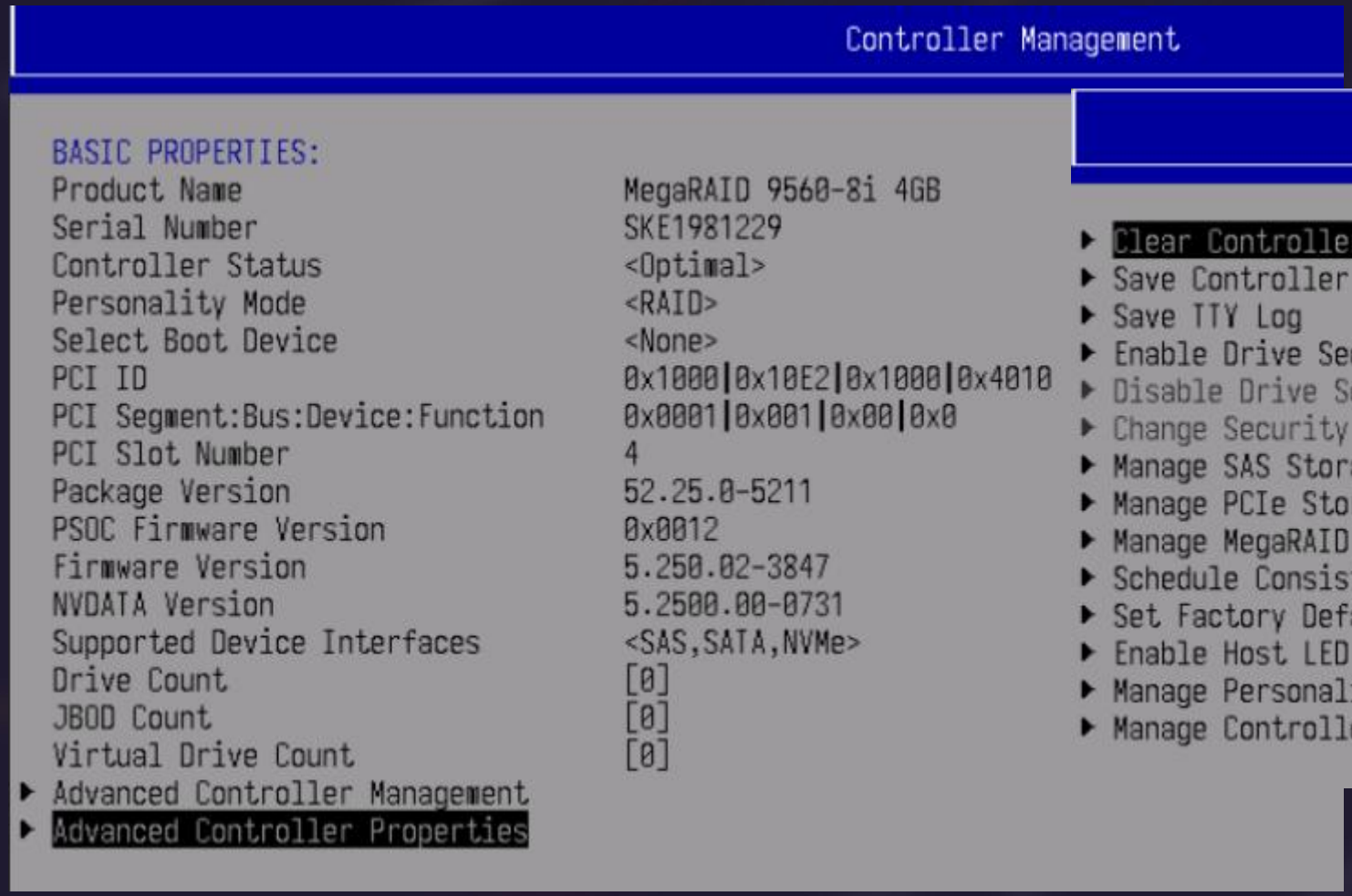
MultiArchUefiPkg - LoongArch64

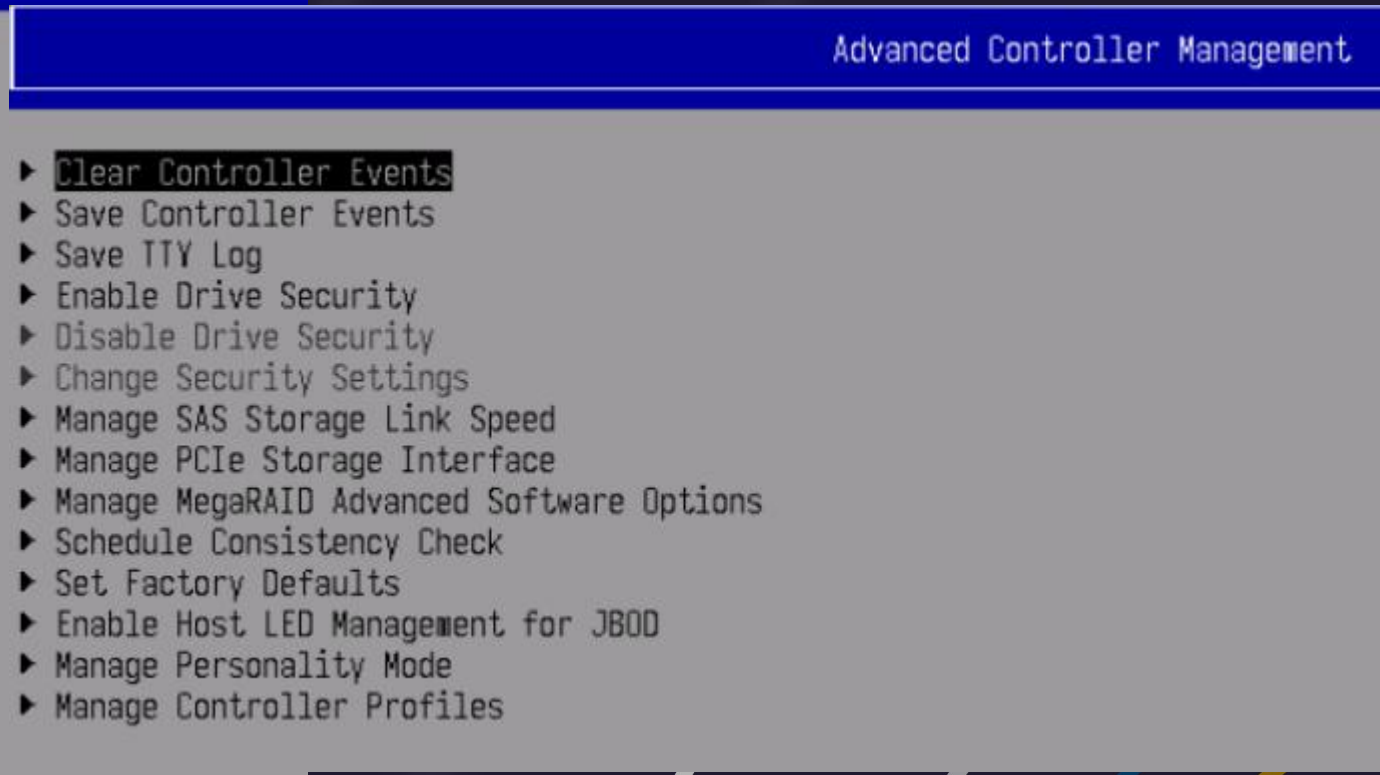
- https://github.com/intel/MultiArchUefiPkg/pull/66
- https://github.com/tianocore/edk2/pull/6524
- https://github.com/intel/unicorn-for-efi/pull/14

> 为了移植MultiArchUefiPkg，LoongArch在上游都做了哪些事？

- 1. edk2针对PCI系统扫描OptionRom时，loongarch64支持加载非原生PE/COFF的EFI驱动;
- 2. edk2针对类似Shell程序，LoongArch64支持加载非原生PE/COFF的EFI.App;
- 3. edk2支持LoongArch64的MMU，允许TLB_REFILL重填例外以及NX/Unaligned异常注册;
- 4. edk2支持LoongArch64的4级4KiB基础页表/3级2MiB大页的虚实1:1映射48位地址空间;
- 5. unicorn-for-efi：加入loongarch64模拟库unicorn衔接;
- 6. MultiArchUefiPkg: 添加LoongArch64的Wrapper/Thunk的具体实现;

运行x64 RAID驱动 - LoongArch64





先前RAID驱动从内核megaraid_sas移植，只支持RAID引导，组模式功能缺失，费时费力不好用，兼容性差，不同型号需要再移植;
现阶段: 功能与x86一致完整，完整执行RAID内FW，兼容性好;

运行x64 NIC驱动 - LoongArch64

先前驱动从Intel官网或第三方开源获取，良莠不齐;
驱动严格受限主BIOS芯片容量，无法做到全类型外设驱动集成;

运行x64 GOP驱动 - LoongArch64


```
Shell> drivers
                T   D
D               Y C I
R               P F A
V  VERSION      E G G #D #C DRIVER NAME                          IMAGE NAME
== ======== = = = == == ================================== ==========
4C 0000000A D - -  1  - MultiArchUefiPkg Emulator Driver    FvFile(E6727A5E-CBCD-44C8-B37F-78BC3A0C
16C8)
59 0000000A B - -  3  3 Console Splitter Driver             ConSplitterDxe
5A 0000000A B - -  2  2 Console Splitter Driver             ConSplitterDxe
5B 0000000A ? - -  -  - Console Splitter Driver             ConSplitterDxe
5C 0000000A B - -  2  2 Console Splitter Driver             ConSplitterDxe
5D 0000000A B - -  1  1 Console Splitter Driver             ConSplitterDxe
73 0000000A D - -  6  - Simple Network Protocol Driver      SnpDxe
74 0000000A B - -  6 12 MNP Network Service Driver          MnpDxe
75 0000000A B - -  6  6 VLAN Configuration Driver           VlanConfigDxe
76 0000000A B - -  6  6 ARP Network Service Driver          ArpDxe
77 0000000A B - -  6  6 DHCP Protocol Driver                Dhcp4Dxe
78 0000000A B - - 12 54 IP4 Network Service Driver          Ip4Dxe
79 0000000A B - - 12 12 MTFTP4 Network Service             Mtftp4Dxe
7A 0000000A B - - 42 72 UDP Network Service Driver         Udp4Dxe
7B 0000000A ? - -  -  - iSCSI Driver                       IScsiDxe
7C 0000000A ? - -  -  - iSCSI Driver                       IScsiDxe
7E 0000000A B - - 48  1 UEFI PXE Base Code Driver          UefiPxeBcDxe
7F 0000000A ? - -  -  - UEFI PXE Base Code Driver          UefiPxeBcDxe
85 00000000 D - -  6  - DNS Network Service Driver         DnsDxe
86 00000000 ? - -  -  - DNS Network Service Driver         DnsDxe
87 0000000A ? - -  -  - HttpDxe                            HttpDxe
88 0000000A ? - -  -  - HttpDxe                            HttpDxe
89 00000010 ? - -  -  - EfiFs ext2/3/4 driver v1.9 (GRUB 2. ext2
8B 09040600 ? X X  -  - Intel(R) PRO/1000 Open Source 9.4.0 GigUndiDxe
8C 02000000 ? - X  -  - WangXun(R) 40/25/10GbE Open Source  FvFile(2D239B6C-AA93-4BC4-8584-1B6C3B01
```

```
02000000 ? - X  -  - WangXun(R) 40/25/10GbE Open Source  FvFile(2D239B6C-AA93-4BC4-8584-1B6C3B01
3)
00011600 ? - -  -  - mucse(R) Tsrn10 Driver 0.1.16       FvFile(AAC9296D-388E-DA83-F247-44B4BE7D
B)
00000010 B - -  1  1 Aspeed Gop Driver                   ASpeedAst2x00GopDxe
00000034 ? - -  -  - JMGPU Video Driver                  JmGopDxe
0000000A B - -  4 52 PCI Bus Driver                      PciBusDxe
0000000A D - -  3  - Platform Console Management Driver  ConPlatformDxe
0000000A D - -  2  - Platform Console Management Driver  ConPlatformDxe
0000000A B - -  1  1 Serial Terminal Driver              TerminalDxe
00000030 ? - -  -  - ls gop driver                       LsGraphicsOutput
0000000A D - -  1  - Graphics Console Driver             GraphicsConsoleDxe
0000000A D - -  4  - Generic Disk I/O Driver             DiskIoDxe
0000000B B - -  1  3 Partition Driver(MBR/GPT/El Torito) PartitionDxe
0000000A D - -  1  - Sata Controller Init Driver         SataController
00000010 D - -  1  - AtaAtapiPassThru Driver             AtaAtapiPassThruDxe
00000010 B - -  1  1 ATA Bus Driver                      AtaBusDxe
0000000A D - -  1  - SCSI Bus Driver                     ScsiBus
0000000A ? - -  -  - Scsi Disk Driver                    ScsiDisk
00000010 ? - -  -  - NVM Express Driver                  NvmExpressDxe
00000010 D - -  2  - Usb Ohci Driver                     OhciDxe
00000030 D - -  2  - Usb Ehci Driver                     EhciDxe
00000030 D - -  1  - Usb Xhci Driver                     XhciDxe
0000000A D - -  5  - Usb Bus Driver                      UsbBusDxe
0000000A D - -  2  - Usb Keyboard Driver                 UsbKbDxe
0000000A D - -  2  - Usb Mouse Driver                    UsbMouseDxe
00000011 ? - -  -  - Usb Mass Storage Driver             UsbMassStorageDxe
00000001 ? - -  -  - LSI Raid Controller Init Driver     LSIRaidController
0000000A D - -  1  - FAT File System Driver              Fat
00000010 ? - -  -  - EfiFs ISO9660 driver v1.9 (GRUB 2.0 iso9660
002055B1 B - X  2  2 Realtek UEFI UNDI Driver(Beta)      FvFile(12283619-ADAE-4F9B-83F9-CBEC7B06
```

先前BMC芯片AST图形驱动从内核移植通过PCIIO协议来驱动，费时费力，只兼容单一Ast2500型号，还可能出现由于本需BIOS来做的GPU_POST逻辑缺失，可能引起上游内核外设驱动底层因底层特性未开启从而出现性能瓶颈;

Loong运行x64.EFI驱动及应用演示

```
FS0:\> pci
   Seg  Bus  Dev  Func
   ---  ---  ---  ----
   ...
    01   02   00    00 ==> Display Controller - VGA/8514 controller
           Vendor 102B Device 2527 Prog Interface 0


FS0:\> LoongArch64LoadOpRom.efi 01 02 00 00
   ROM 0x00016A00 bytes
   -------------------
   +0x00000000: UNSUPPORTED BIOS image (0x9000 bytes)
   +0x00009000: UNSUPPORTED 0x8664 UEFI image (0xDA00
bytes)

FS0:\> X64LoadOpRom.efi
   Image type X64 can't be loaded on LoongArch64 UEFI system.
```

```
   FS0:\> load EmulatorDxe.efi
   ...
   FS0:\> X64LoadOpRom.efi 01 02 00 00
ROM 0x00016A00 bytes
   -------------------
   +0x00000000: UNSUPPORTED BIOS image (0x9000
bytes)
   +0x00009000: SUPPORTED 0x8664 UEFI image
(0xDA00 bytes)
   +0x00009000:    Subsystem: 0xB
   +0x00009000:    InitializationSize: 0xDA00 (bytes)
   +0x00009000:    EfiImageHeaderOffset: 0x38
   +0x00009000:    Compressed: yes
   ...
   Loading driver at 0x000F8840000
EntryPoint=0x000F88421A0
   Recursive connect...
```

愿景 - LoongArch64 UEFI原生+模拟

- 成为打造一个互操作的 LoongArch 生态系统的一环，支持像 PC/服务器一样灵活扩展与启动;
- 插入标准 PCIe设备（如显卡、SSD、NIC）即能识别/启动，无需定制固件;
- 和x86/ARM一样拥有成熟的固件抽象层，支持多操作系统;

目前已验证的x86.EFI:

类别	驱动型号（厂商）
Application	x64Shell.efi (EDK)
BMC.Driver.Gop	AST2500/AST2600 (ASPEED)
RAID 控制器.Driver	MegaRAID 9361/9560 (LSI)
存储控制器.Driver	SAS 3008 HBA (Broadcom)
网卡.Driver.NIC	82579/82599/X710 (Intel); 10G (网迅); 25G (Marvell)
GPU (AMD).GOP	RX 9070/7900 XT/6900 XT/6750 XT (AMD)
GPU (NVIDIA).GOP	RTX 4090/4060; GTX 730 K (NVIDIA)

收集外设验证情况： https://github.com/loongson/Firmware/issues/128

欢迎大家贡献交流

MultiArchUefiPkg - LoongArch64 移植贡献者

- Chao Li： https://github.com/kilaterlee
- Dongyan Qian： https://github.com/MarsDoge

参考资料：

https://uefi.org/

https://tianocore.org/

https://github.com/ardbiesheuvel/X86EmulatorPkg

https://www.unicorn-engine.org/

https://github.com/intel/MultiArchUefiPkg/blob/main/Docs/RviSummitMarch2023/2023-06-08-Andrei-WARKENTIN-abstract.pdf

https://github.com/intel/MultiArchUefiPkg/blob/main/Docs/RviSummitMarch2023/multi_isa_uefi_compat.pdf

https://github.com/intel/MultiArchUefiPkg/blob/main/Docs/RviSummitMarch2023/multi_isa_uefi_compat_poster.pdf

https://github.com/intel/MultiArchUefiPkg/blob/main/Docs/Uefi2023/multi_isa_fw_compat.pdf

“

谢谢！AOSC 加油!

”