UEFI and RISC-V

Abner Chang, Dong Wei
The UEFI Forum Organization

Board of Directors (12 Promoters)

Officers:
President: Mark Doran (Intel); VP (CEO): Dong Wei (HPE)
Secretary: Jeff Bobzin (Insyde); Treasurer: Bill Keown (Lenovo)

Industry & Communications WG
UEFI Specification WG
Platform Initialization WG
Test WG
ACPI WG
Security Response Team

Security Subteam
Configuration Subteam
Network Subteam
Shell Subteam
ARM Binding Subteam

12 Promoters
42 Contributors
213 Adopters
28 Individual Adopters
UEFI Technology

Platform Initialization (PI)
- Interfaces produced & consumed by firmware only; promote interoperability between firmware components

UEFI
- Pre-OS (and limited runtime program interfaces) between UEFI Applications (incl. OSes)/UEFI Drivers and system firmware
ACPI Technology

– Static tables and primary runtime interpreted control methods provided by system firmware to the OS for system configuration, power management and error handling

– Processor architecture agnostic
UEFI & ACPI History

**UEFI History**

1995  HP/Intel needed a boot architecture for Itanium servers that overcame BIOS PC-AT limitations

1997 - 2000  Intel created EFI with HP and others in the industry, made it processor agnostic (x86, ia64)

2004  tianocore.org, open source EFI community launched

2004  Unified EFI (UEFI)

2005  The UEFI Forum, with 11 promoters, was formed to standardize EFI, extended to x64

2009  UEFI extended to ARM AArch32

2012  Windows 8 and ubiquitous native UEFI adoption for client PCs (Boot Performance, Secure Boot focused)

2013  Linux Distro extended support for UEFI Secure Boot. First Linux Foundation hosted UEFI Plugfest. UEFI v2.4 extended to ARM AArch64.

2014  ACPI v5.1 for ARM AArch64 support (e.g., ARM SBSA/SBBR servers)

2015  UEFI v2.5, PI v1.4, ACPI v6.0 for NVDIMM support

2016  Ready for RISC-V?

**ACPI History**

1996  Intel/Microsoft/Toshiba created ACPI 1.0 for 16 and 32 bit PC client devices

2000  Compaq/Intel/Microsoft/Phoenix/Toshiba publishes ACPI 2.0 for 64-bit support as well as support for multiprocessor workstations and servers

2004  HP/Intel/Microsoft/Phoenix/Toshiba published ACPI 3.0 further enhancing the spec to support both client and server systems

2009  ACPI 4.0 is published providing additional support for both client and server systems

2011  Hardware-reduced ACPI model was introduced into the published ACPI 5.0 spec to include the support for SoC devices. ARM specific descriptions are also introduced

2013  ACPI Asset transferred to the UEFI Forum.

**UEFI as the converged firmware infrastructure**
UEFI/PI Execution Phases

Power on ➔ [....Platform initialization..] ➔ [....OS boot...] ➔ Shutdown
RISC-V UEFI Port on EDKII (EFI Development Kit II)

OVMF (Open Virtual Machine Firmware) RISC-V Package on QEMU
FD, Flash Device (ROM)

Power on — [....Platform initialization..] — [....OS boot...] — Shutdown

UEFI/PI Execution Phases

Reset Vector (VTF)

Security SecMain.efi (SEC)

Pre EFI Initialization (PEI)

Driver Execution Environment (DXE)

Boot Dev Select (BDS)

Transient System Load (TSL)

Runtime (RT)

After Life (AL)

Generate Reset Vector VTF for RISC-V

RISC-V Reset Vector (0xF...FF00)

Generate Fw

Two standard values 0xF...FFE00 or 0x0...0200

CSR MTVEC

EB7FF0037 C5 FC FF 67 00 85 32 04 00 00 00 00 00 00 00 7...0...2......
**UEFI/PI Execution Phases**

Power on → [...Platform initialization..] → [...OS boot...] → Shutdown
**FD, Flash Device (ROM)**

- **Volume Top File**
  - FD
  - FV
  - FV
  - FV
  - FFS
  - FFS
  - FSV
  - FFS
  - FV
  - FFS
  - FFS
  - FFS
  - FV
- **Generate Reset Vector VTF for RISC-V**
- **RISC-V Reset Vector (0xF…FF00)**
- **Power on**
  - [....Platform initialization..]
  - [....OS boot...]
  - Shutdown

### UEFI/PI Execution Phases

- **Reset Vector**
  - Security SecMain.efi
- **Pre EFI Initialization**
  - Pre EFI Initialization (PEI)
- **Boot Dev Select**
  - Boot Dev Select (BDS)
- **Transient System Load**
  - Transient System Load (TSL)
- **Runtime**
  - Runtime (RT)
- **After Life**
  - After Life (AL)

- **FD, Flash Device (ROM)**
  - Select (BDS)
  - Transient System Load (TSL)
  - Runtime (RT)
  - After Life (AL)

- **UEFI/PI Execution Phases**

- **Memory address**: 32-bit, I/O address: 32-bit
- **SET_JUMP/LONG_JUMP**
- **SET_JUMP requires return value on EDKII**
- **Memory and I/O**
  - Memory map Read/Write and Memory map I/O read/write
- **BaseLib**
  - Processor intrinsic function, Ena/Dis interrupt (CSR), Stack switch, Breakpoint (EBREAK), CPU pause (NOP)

- **CPU HOB**
- **PECOFF RISC-V relocation type**
- **Prepare Temporary memory**
- **Platform memory initialization**
- **CPU HOB to declare memory address size**
- **RISC-V SET_JUMP/LONG_JUMP to switch stack to permanent**
- **RISC-V memory map read/write**
- **RISC-V I/O read/write (memory map)**
- **EDKII BaseLib for RISC-V**
- **RISC-V specific PEI service pointer retrieval**

- **Maintain RISC-V machine trap handler in MSCRATCH CSR**

- **generate Reset Vector VTF for RISC-V**
- **RISC-V Reset Vector (0xF…FF00)**
- **Processor Binding** (structure alignment, variable alignment)
- **Generate EFI image PECOFF**
- **Generate Temporary memory**
- **CPU HOB**
  - Memory address: 32-bit, I/O address: 32-bit
- **SET_JUMP/LONG_JUMP**
  - SET_JUMP requires return value on EDKII
- **Memory and I/O**
  - Memory map Read/Write and Memory map I/O read/write
- **BaseLib**
  - Processor intrinsic function, Ena/Dis interrupt (CSR), Stack switch, Breakpoint (EBREAK), CPU pause (NOP)

- **Memory address**: 32-bit, I/O address: 32-bit
- **CPU HOB**
- **PECOFF RISC-V relocation type**
- **Prepare Temporary memory**
- **Platform memory initialization**
- **CPU HOB to declare memory address size**
- **RISC-V SET_JUMP/LONG_JUMP to switch stack to permanent**
- **RISC-V memory map read/write**
- **RISC-V I/O read/write (memory map)**
- **EDKII BaseLib for RISC-V**
- **RISC-V specific PEI service pointer retrieval**

- **Maintain RISC-V machine trap handler in MSCRATCH CSR**
FD, Flash Device (ROM)

**UEFI/PI Execution Phases**

1. **Power on**
   - [....Platform initialization..]

2. **Pre EFI Initialization**
   - Security SecMain.efi
   - Processor Binding
   - Generate EFI Image PECOFF
   - PECON RISC-V relocation type.
   - Prepare Temporary memory

3. **Driver Execution Environment**
   - Platform memory initialization
   - CPU HOB to declare memory address size
   - RISC-V SET_JUMP/LONGJ UMP to switch stack to permanent
   - RISC-V memory map read/write
   - RISC-V I/O memory map read/write
   - EDKII BaseLib for RISC-V
   - RISC-V specific PEI service pointer retrieval

4. **Boot Dev Select**
   - Maintain RISC-V machine trap handler in MSCRATCH CSR

5. **Runtime**
   - RegisterHandler
   - SetTimerPeriod
   - GetTimerPeriod
   - GenerateSoftwareInterrupt
   - RISC-V DXE CPU arch protocol
   - mtime CSR
   - mtimecmp CSR
   - rcause CSR
   - mp CSR
   - mc CSR
   - CpuFlushDataCache
   - CpuEnableInterrupt
   - CpuDisableInterrupt
   - CpuGetInterruptState
   - CpuInit
   - CpuRegisterInterruptHandler
   - CpuGetTimerValue
   - CpuGetMemoryAttribute

6. **After Life**
   - [....OS boot...]
   - Shutdown
FD, Flash Device (ROM)

UEFI/PI Execution Phases

Power on  [....Platform initialization..]  [....OS boot...]  Shutdown
**RISC-V Machine mode trap handler.**

```c
/**
 * RISC-V Machine mode Trap Handler.
 */

void RiscVMachineModeTrapHandler()
{
    RISCV_TRAP_HANDLER TrapHandle;
    RISCV_MACHINE_MODE_CONTEXT *Context;

    //DEBUG ((EFI D INFO, "Enter RISC-V Machine Mode Trap Handler.\n"));
    Context = (RISCV_MACHINE_MODE_CONTEXT *) (UINTN) RiscGetScratch();
    TrapHandle = (RISCV_TRAP_HANDLER) (UINTN) Context->MachineModeTrapHandler;
    TrapHandle();
}
```

---

**Power on** → **[....Platform initialization..]** → **[....OS boot...]** → **Shutdown**

**UEFI/PI Execution Phases**
UEFI/PI Execution Phases

Power on → [....Platform initialization..] → [....OS boot...] → Shutdown
FD, Flash Device (ROM)

UEFI/PI Execution Phases

Power on → [....Platform initialization..] → [....OS boot...] → Shutdown
FD, Flash Device (ROM)

UEFI/PI Execution Phases

Power on → [....Platform initialization..] → [....OS boot...] → Shutdown
RISC-V QEMU

- **QEMU RISC-V PC/AT board**
  Built up RISC-V PC/AT board on QEMU with some PC peripherals.

- **QEMU PC/AT memory map devices (CMOS, PM, PCI and other devices)**
  Changed these PC peripherals to memory map I/O device because RISC-V uses memory map I/O.

- **RISC-V machine mode on RISC-V QEMU port**
  Implemented RISC-V machine mode on RISC-V QEMU port.
Issues

• How PECOFF support High 20bit/ Low 12bit relocations

```c
Temp = &mRootBridgeDevicePathTemplate
lui a5, %hi (mRootBridgeDevicePathTemplate)
add a1, a5, %lo (mRootBridgeDevicePathTemplate)
```

• RISC-V relocation in GNU link
  When relative offset < 0x800, it forces to use X0 (hard wired to 0) as base address. This results in inconsistent register usage when load the target address.

• GNU link Optimization (no-relax support)
  When relative offset < 0x800, it deletes AUIPC op-code.

Call Function ->
```c
auipc t0, 20-bit // U-type integer
jalr t0, 12-bit // I-type integer
```
We need more in RISC-V spec

- Timer, add periodical timer CSR
- RTC, provide date, time, year and alarm CSR
- PI Management Mode support
- MP support
- ACPI support
- Reset mechanism
UEFI/PI spec change for RISC-V

UEFI spec change for RISC-V
- 2.1.1. UEFI Images
- 2.3. Calling Conventions
- 2.3. RISC-V 32 (64) Platforms
- 17.2 EFI Debug Support Protocol

PI spec change for RISC-V
- Volume 1 : 5.4 RISC-V PEI Services Table Retrieval
- Volume 3 : PI Status code
Next step

- PE COFF image machine type for RISC-V
- PE COFF image relocation type for RISC-V
- EDKII RISC-V code review and commit
- QEMU RISC-V code review and commit
- GNU Link code change review and commit
- EDKII RISC-V OVMF: Add ACPI support
- QEMU : Keyboard/USB/ACPI on QEMU RISC-V PC/AT board
- QEMU : Boot to Linux on RISC-V UEFI port
Thank you

abner.chang@hpe.com
dong.wei@hpe.com