

presented by



**Hewlett Packard
Enterprise**



UEFI Port to RISC-V Processor Architecture

UEFI Spring Plugfest – March 30, 2016
Abner Chang, SW/FW Technologist

Agenda



- Introduction
- RISC-V UEFI port on EDKII
- Spec changes for RISC-V
- Appendix



RISC and CISC



RISC Reduced instruction set computer, like ARM, Power PC, SPARC processor

Fixed length of instruction, simple instructions to be executed in one CPU clock.

Less instruction sets, RISC-V has around 90+ instructions for example.

Better performance of instruction fetch and pipeline.

Slightly complex when writing program in assembly language.

Large code size

Emphasis on software

CISC Complex instruction set computer, like Intel processor

Variable length of instructions, multiple clocks instruction

To complete a task in few line assembly as possible.

Specific instructions for specific purpose

Easy to write program in assembly language

Less code size

Emphasis on hardware

What is RISC-V processor



- From UC Berkeley, the fifth major RISC ISA design.
- Roman numeral “V” to signify “variations” and “vectors”
- New and open instruction set architecture (ISA) original designed for computer architecture research and education
- Now become a standard open ISA for industrial



What is RISC-V processor

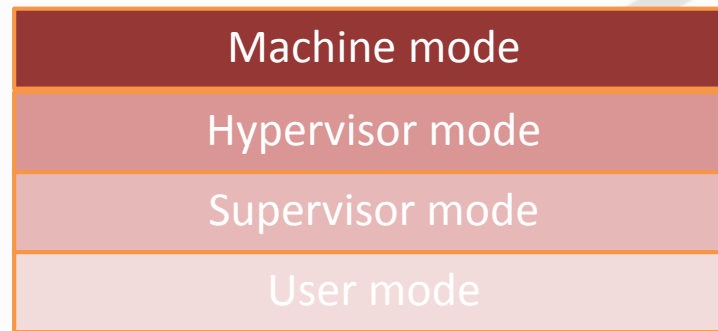


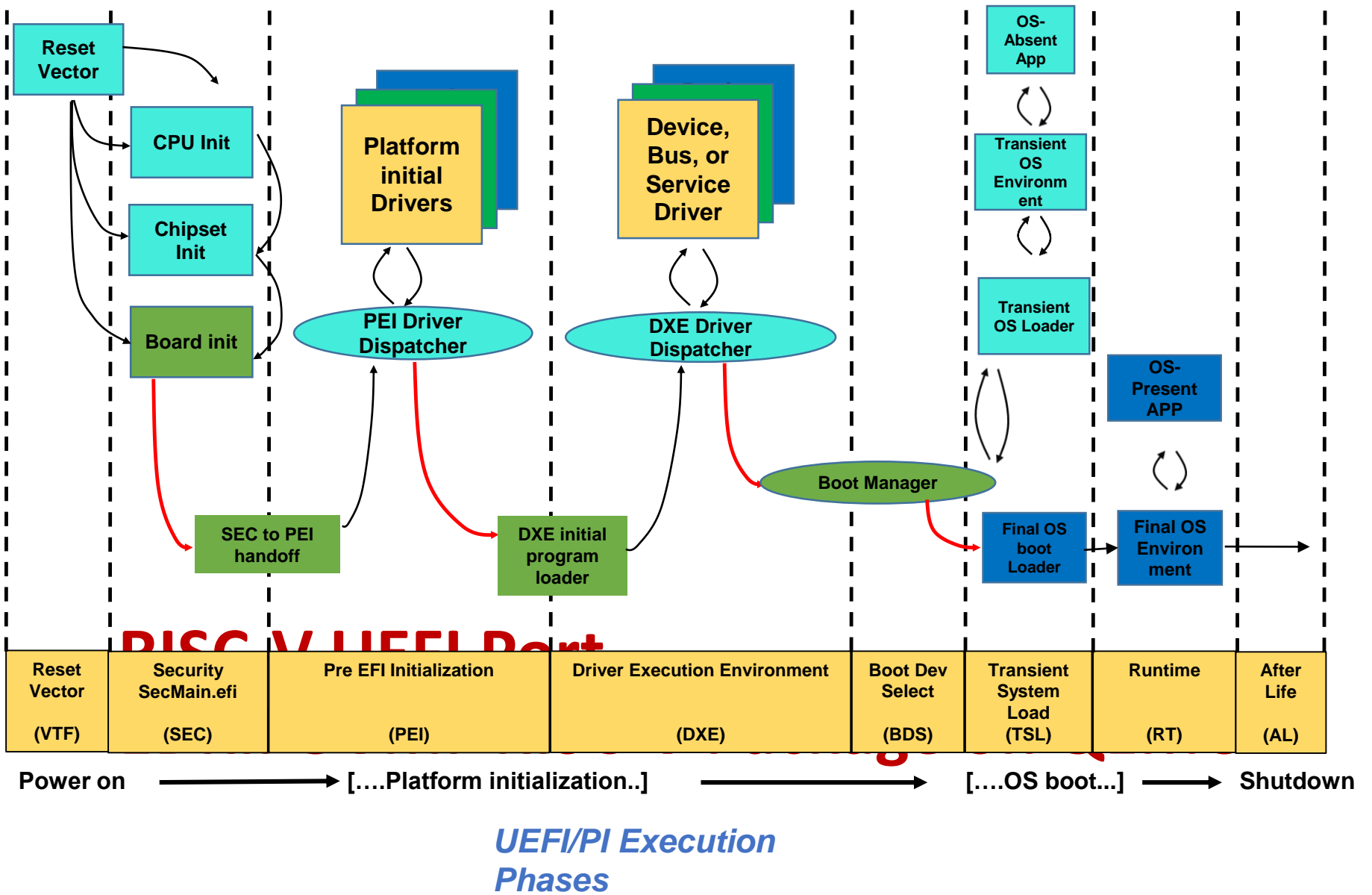
- 32-bit, 64-bit and 128 bit processor.
- Instruction groups
 - Base integer instruction set
 - Standard extension for integer multiplication and division
 - Standard extension for atomic instructions
 - Standard extension for single-precision floating point
 - Standard extension for double-precision floating point
 - Standard extension for quad-precision floating point
 - Standard extension for compressed instructions
 - Standard extension for bit manipulation
 - Standard extension for SIMD instruction

What is RISC-V processor



- 4 privileged operation modes





0xF...FFFFFFF

FD, Flash Device (ROM)



```
007FFF00 37 C5 FC FF 67 00 85 32 04 00 00 00 00 00 00 00 7...g..2.....
```

Generate Reset Vector VTF for RISC-V

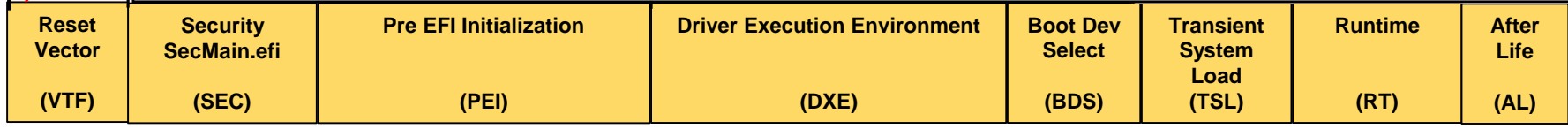
CSR MTVEC
Machine Trap Vector Base Address

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



```
ASM_PFX( ModuleEntryPoint):
  \LoadHigh20BitAddress:
    li    a0, 0x12345000
  \LoadLow12BitAddress:
    jalr  x0, a0, 0x678
```

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



UEFI/PI Execution Phases

0xF...FFFFFFF

FD, Flash Device (ROM)



```

ProcessorBinding.h (datatype alignment)
typedef unsigned long long UUINT64 __attribute__((aligned(8)));
                          INT64 __attribute__((aligned(8)));
                          UUINT32 __attribute__((aligned(4)));
                          INT32 __attribute__((aligned(4)));
                          UUINT16 __attribute__((aligned(2)));
                          CHAR16 __attribute__((aligned(2)));
                          INT16 __attribute__((aligned(2)));

ELF to PECOFF
PECOFF Target Machine
Type
- 0x5032 for RISC-V 32
- 0x5064 for RISC-V 64

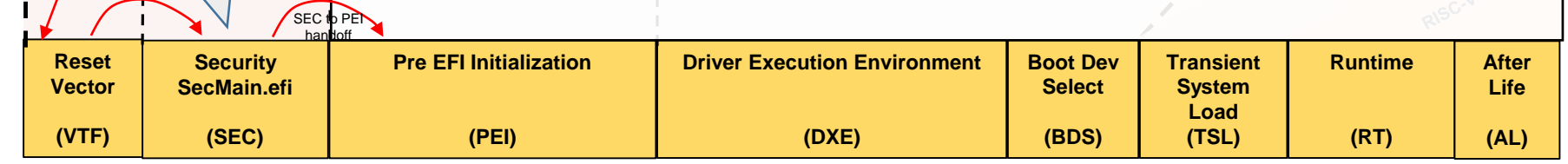
ELF to PECOFF
type to EFI IMAGE relocation type (PECOFF)

```

- Generate EFI PECOFF image
- PECOFF RISC-V relocation type
- ProcessorBinding (structure alignment, variable alignment)
- Prepare Temporary memory

Generate Reset Vector VTF for RISC-V

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



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

UEFI/PI Execution Phases

RISC-V Machine Mode

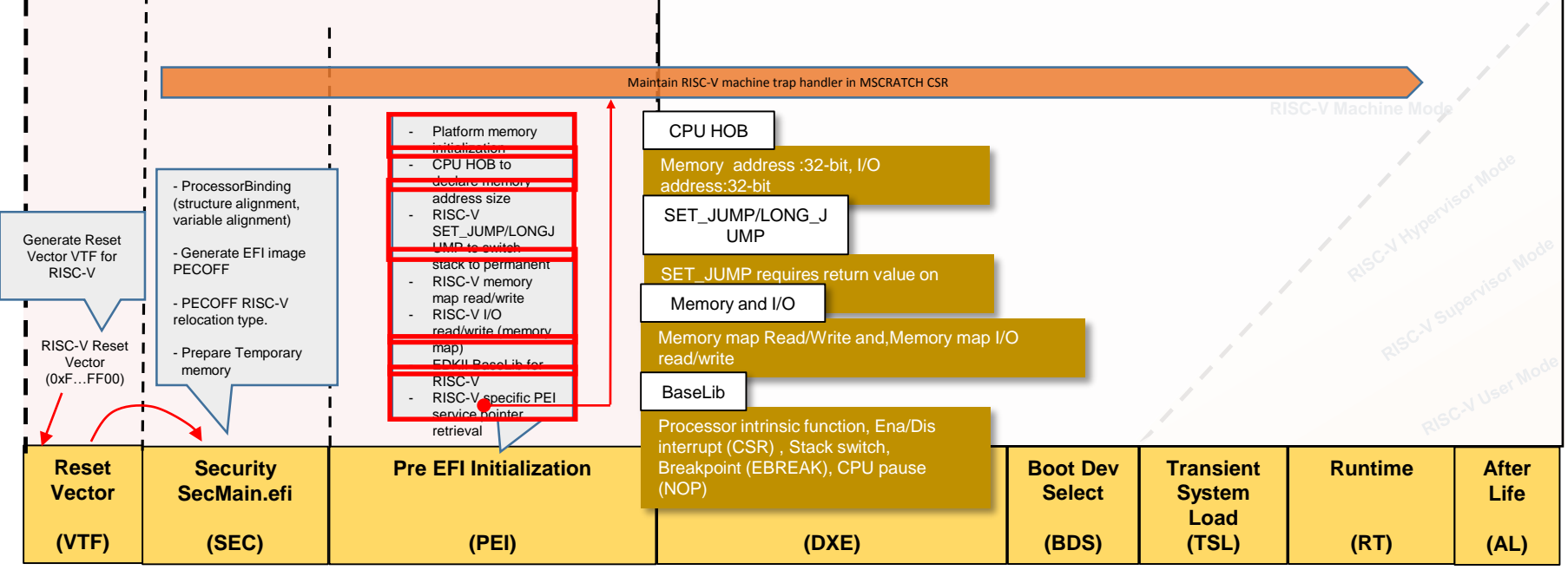
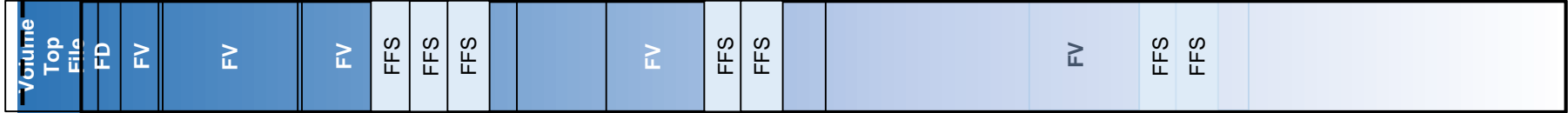
RISC-V Hypervisor Mode

RISC-V Supervisor Mode

RISC-V User Mode

0xF...FFFFFF

FD, Flash Device (ROM)

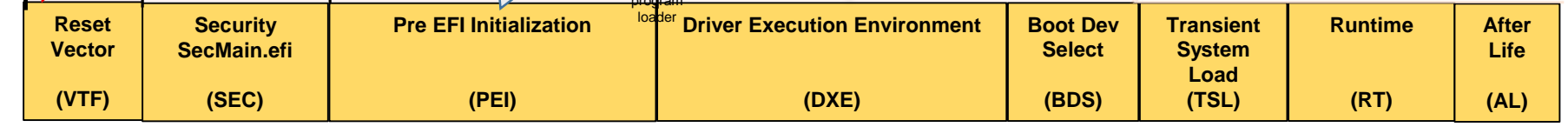
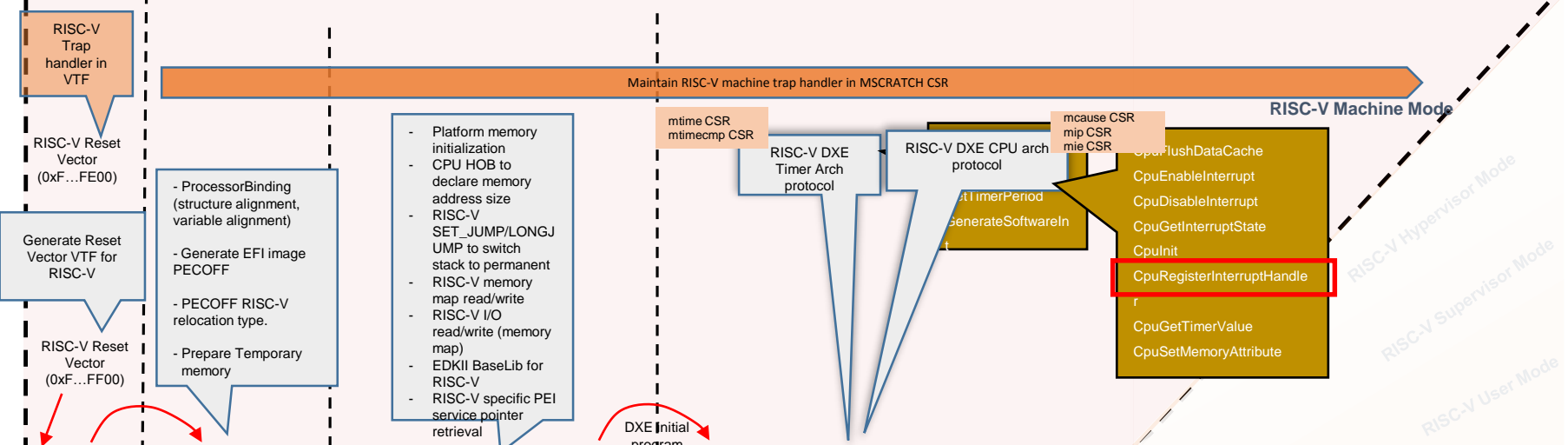


Power on → [...Platform initialization...] → [...OS boot...] → Shutdown

UEFI/PI Execution Phases

0xF...FFFFFFF

FD, Flash Device (ROM)



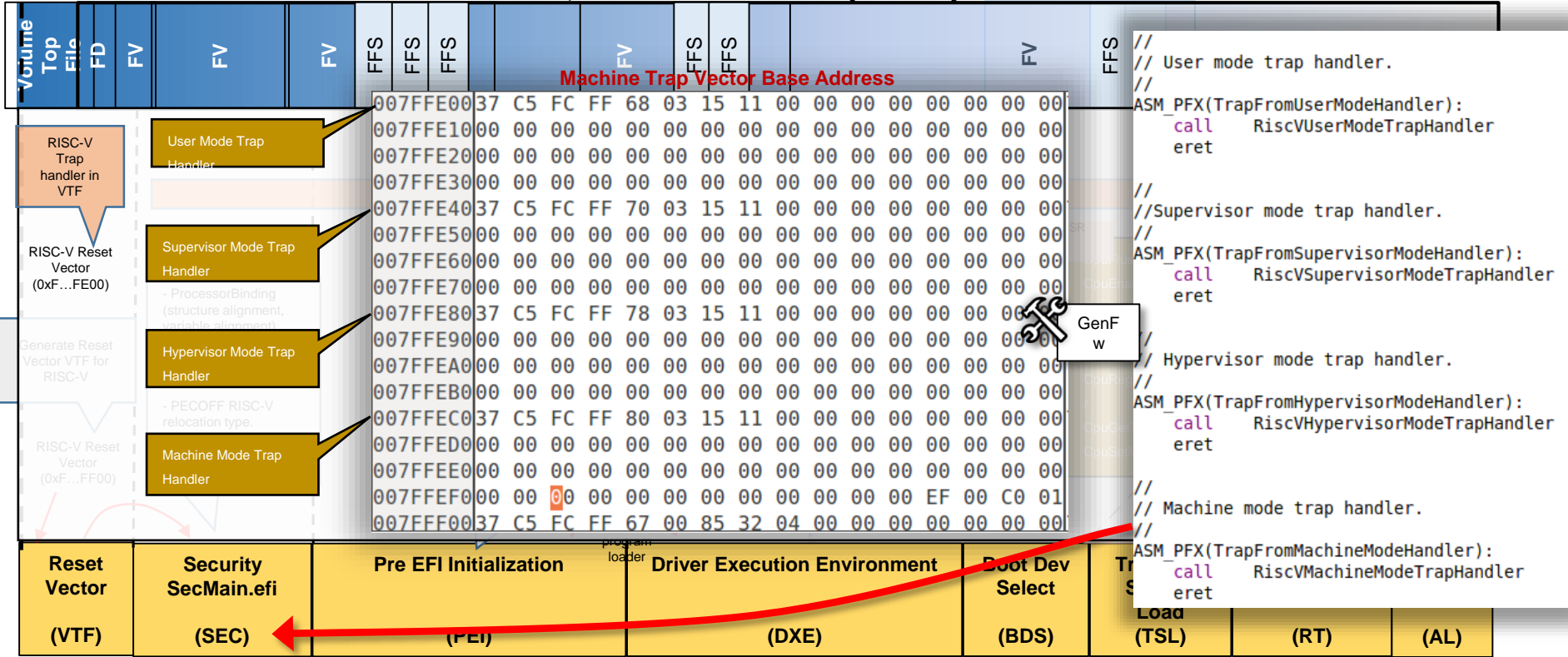
Power on → [...Platform initialization...] → [...OS boot...] → Shutdown

UEFI/PI Execution Phases

RISC-V Hypervisor Mode
RISC-V Supervisor Mode
RISC-V User Mode

0xF...FFFFFFF

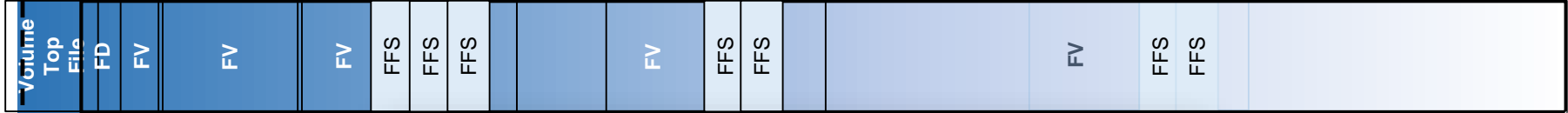
FD, Flash Device (ROM)



UEFI/PI Execution Phases

0xF...FFFFFFF

FD, Flash Device (ROM)



RISC-V Trap handler in VTF

RISC-V Reset Vector (0xF...FE00)

Generate Reset Vector VTF for RISC-V

- ProcessorBinding (structure alignment, variable alignment)
- Generate EFI image PECOFF
- PECOFF RISC-V relocation type.
- Prepare Temporary memory

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

```

/**
 * RISC-V Machine mode trap handler.
 */
VOID
RiscVMachineModeTrapHandler (
    VOID
)
{
    RISC_V_TRAP_HANDLER TrapHandle;
    RISC_V_MACHINE_MODE_CONTEXT *Context;

    //DEBUG ((EFI_D_INFO, "Enter RISC-V Machine Mode Trap Handler.\n"));
    Context = (RISC_V_MACHINE_MODE_CONTEXT *) (UINTN) RiscVGetScratch ();
    TrapHandle = (RISC_V_TRAP_HANDLER) (UINTN) Context->MachineModeTrapHandler;
    TrapHandle ();
}

```

service pointer retrieval

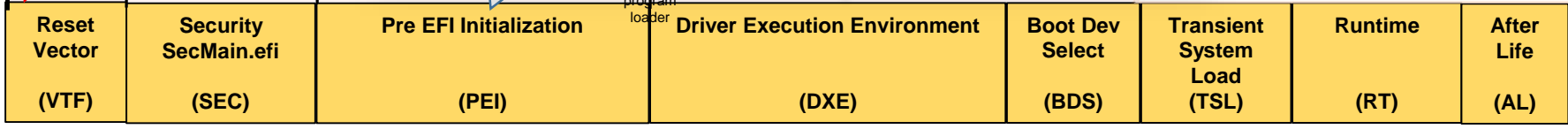
DXE Initial program loader

RISC-V Machine Mode

RISC-V Hypervisor Mode

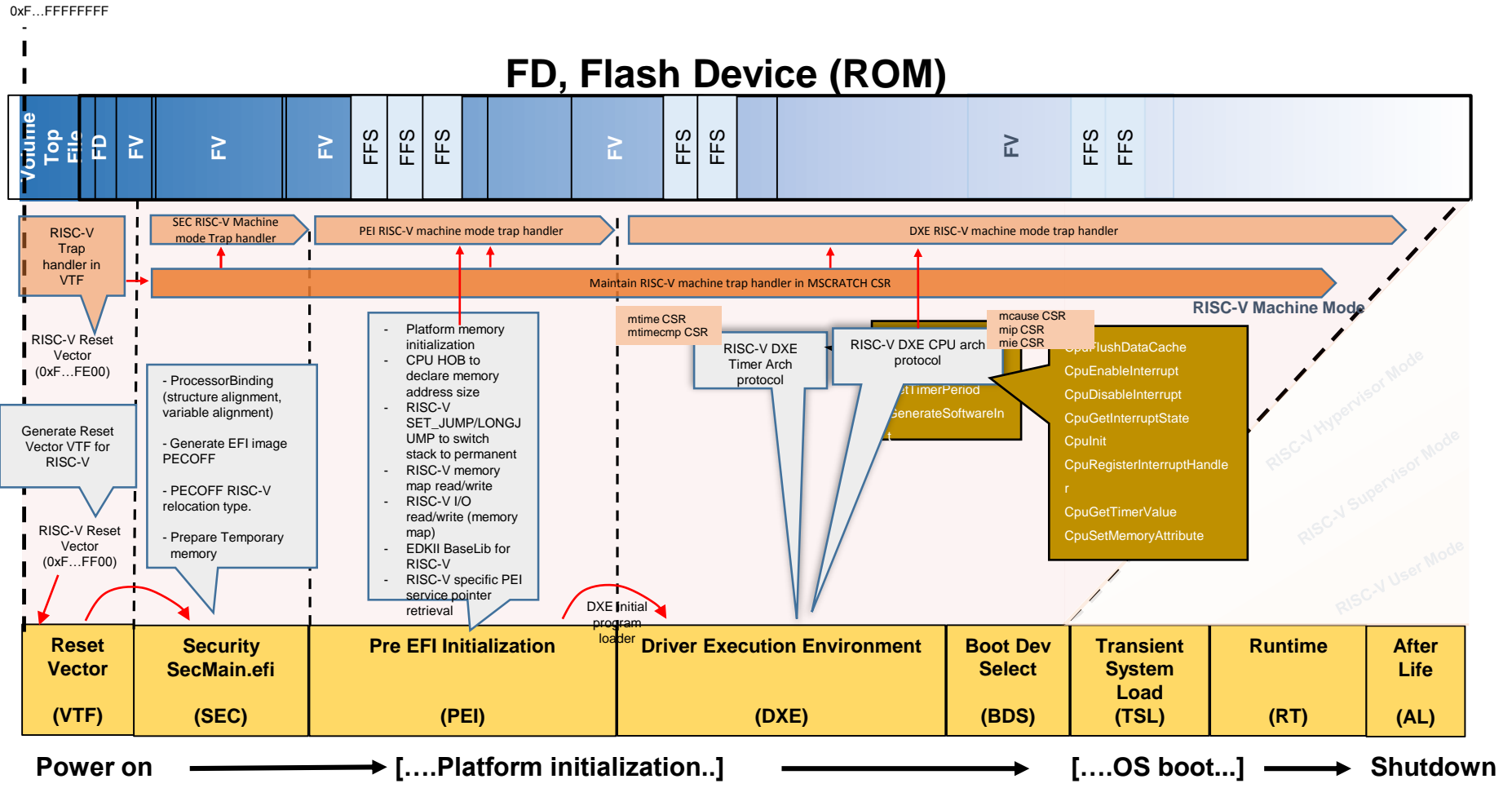
RISC-V Supervisor Mode

RISC-V User Mode



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

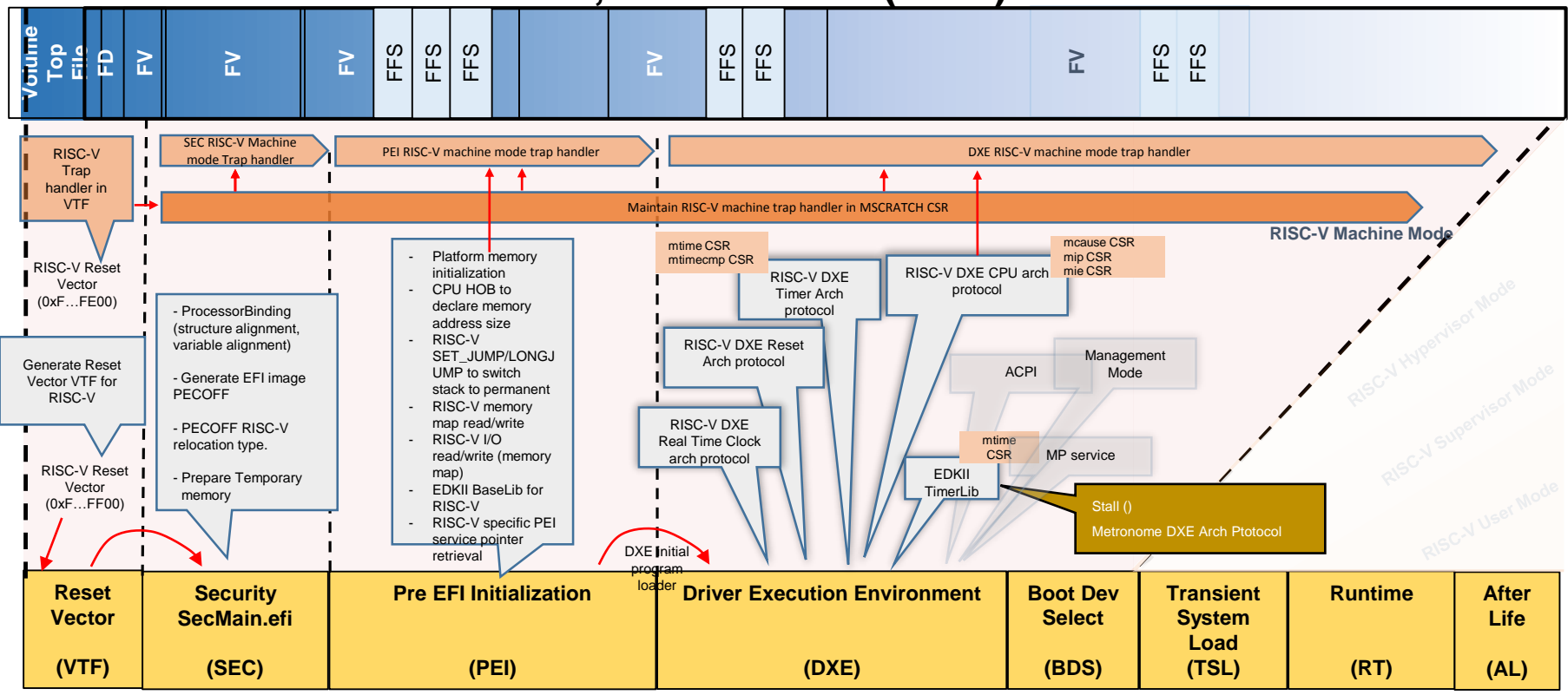
UEFI/PI Execution Phases



UEFI/PI Execution Phases

0xF...FFFFFFF

FD, Flash Device (ROM)

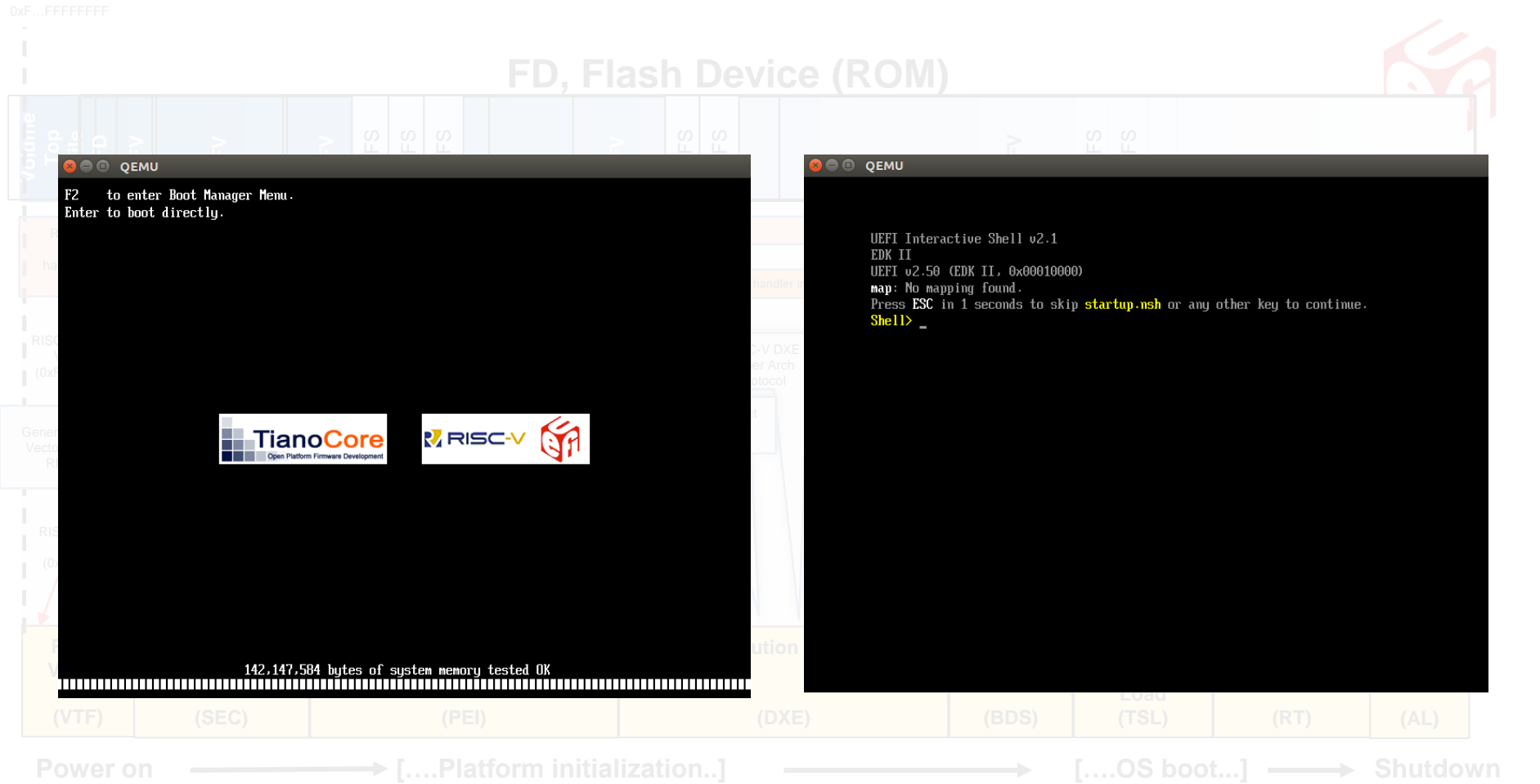


Power on → [...Platform initialization...] → [...OS boot...] → Shutdown

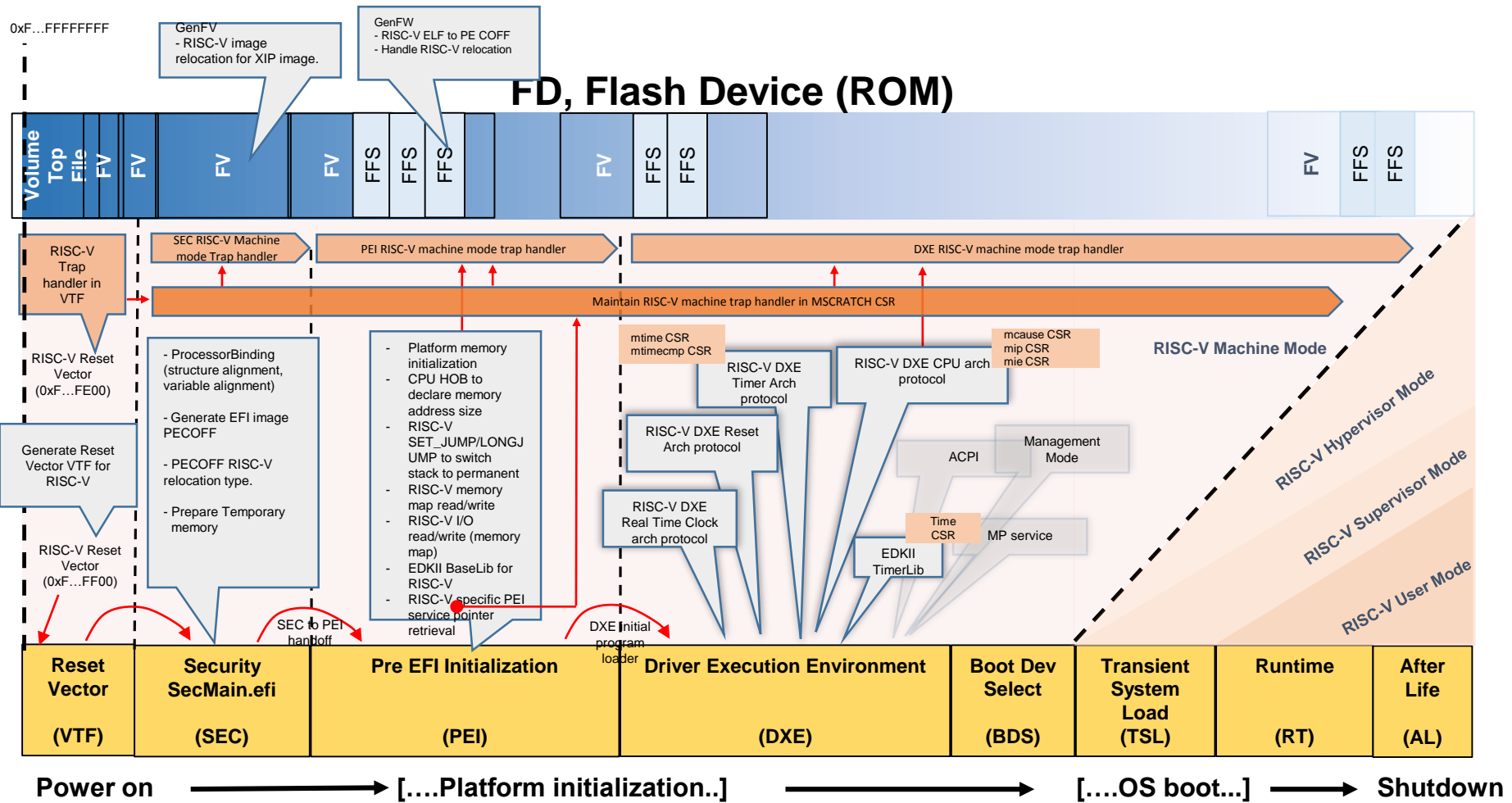
UEFI/PI Execution Phases



FD, Flash Device (ROM)



UEFI/PI Execution Phases



UEFI/PI Execution Phases



Specification changes for RISC-V

UEFI/PI Spec changes for RISC-V (in progress)



UEFI spec change for RISC-V

- 2.1.1. UEFI Images
- 2.3. Calling Conventions
- 2.3. RISC-V 32 (64/128) 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



PE/Coff changes for RISC-V



Microsoft will release RISC-V related definition in next PE/COFF specification.

PE/COFF image machine type,

- IMAGE_FILE_MACHINE_RISCV32 0x5032
- IMAGE_FILE_MACHINE_RISCV64 0x5064
- IMAGE_FILE_MACHINE_RISCV128 0x5128

RISC-V image relocation types,

- IMAGE_REL_BASED_RISCV_HI20 5
- IMAGE_REL_BASED_RISCV_LO12I 7
- IMAGE_REL_BASED_RISCV_LO12S 8





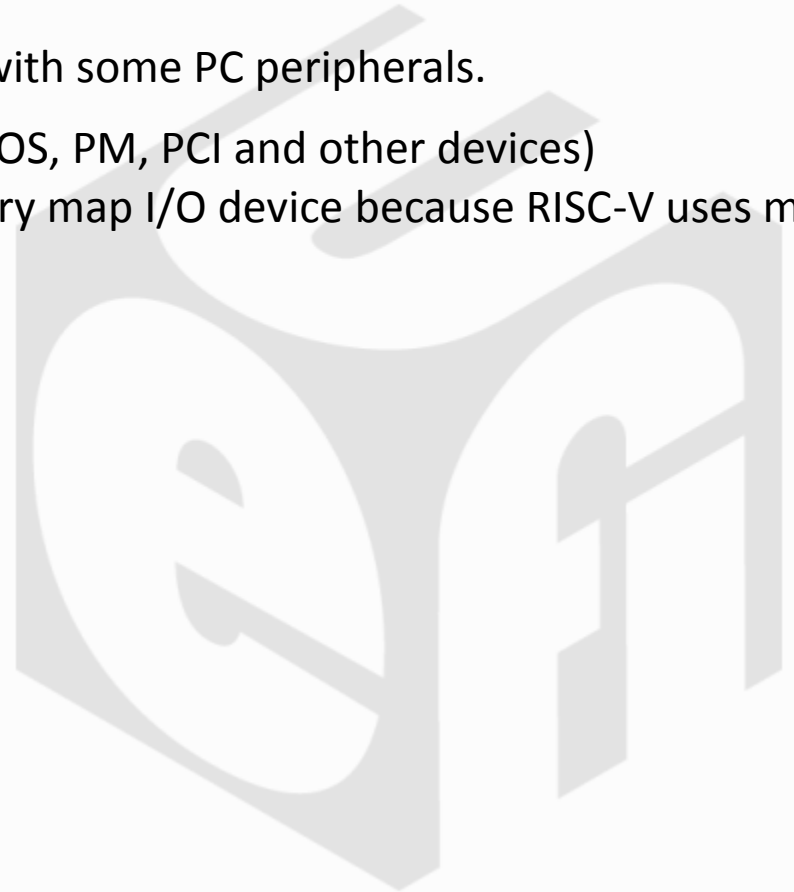
Appendix



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 Interrupt Controller



- BERI (Bluespec Extensible RISC Implementation) Programmable Interrupt Controller
BERI PIC is attached to BERI processor. BERI PIC supports large number of external interrupts, total up to 1024 interrupt sources.



Need more in RISC-V spec



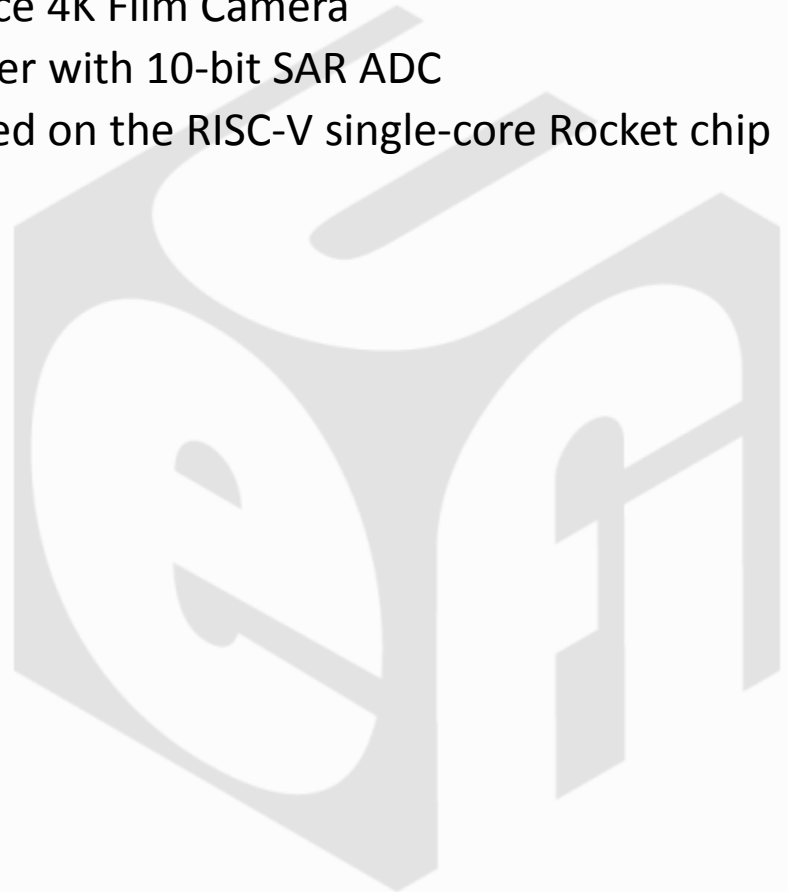
- Bus Interface
- PI Management Mode support
- ACPI support
- MP support
- Reset mechanism



RISC-V Implementations



- lowRisc Rocket chip (RISC-V based SoC)
- RISC-V in AXIOM First Fully Open Source 4K Film Camera
- A 32-bit 100MHz RISC-V Microcontroller with 10-bit SAR ADC
- SoC for a Satellite Navigation Unit based on the RISC-V single-core Rocket chip
- RISC-V-based photonic processor



Thanks for attending the
UEFI Spring Plugfest 2016



For more information on
the Unified EFI Forum and
UEFI Specifications, visit
<http://www.uefi.org>



presented by



**Hewlett Packard
Enterprise**