Intel Cup Embedded System Design Contest

Introduction to Embedded Bootloader

Intel SSG/SSD/UEFI

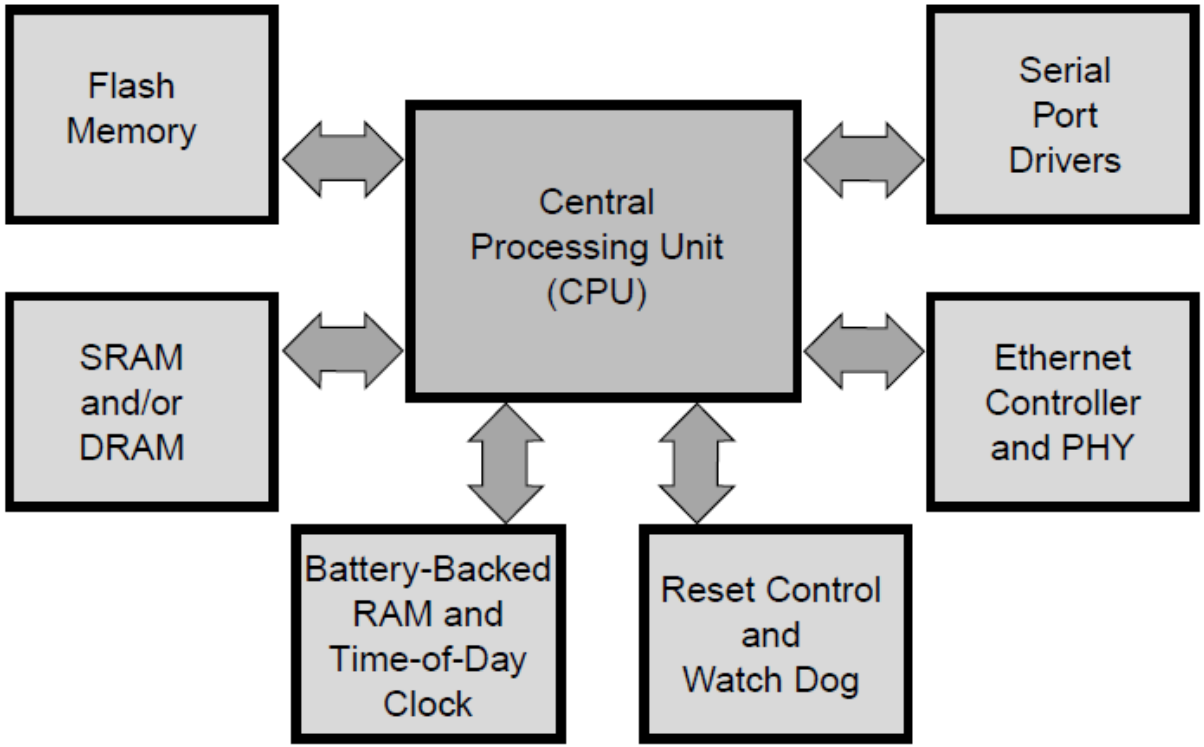
Legal Disclaimer

Intel Cup Embedded System Design Contest

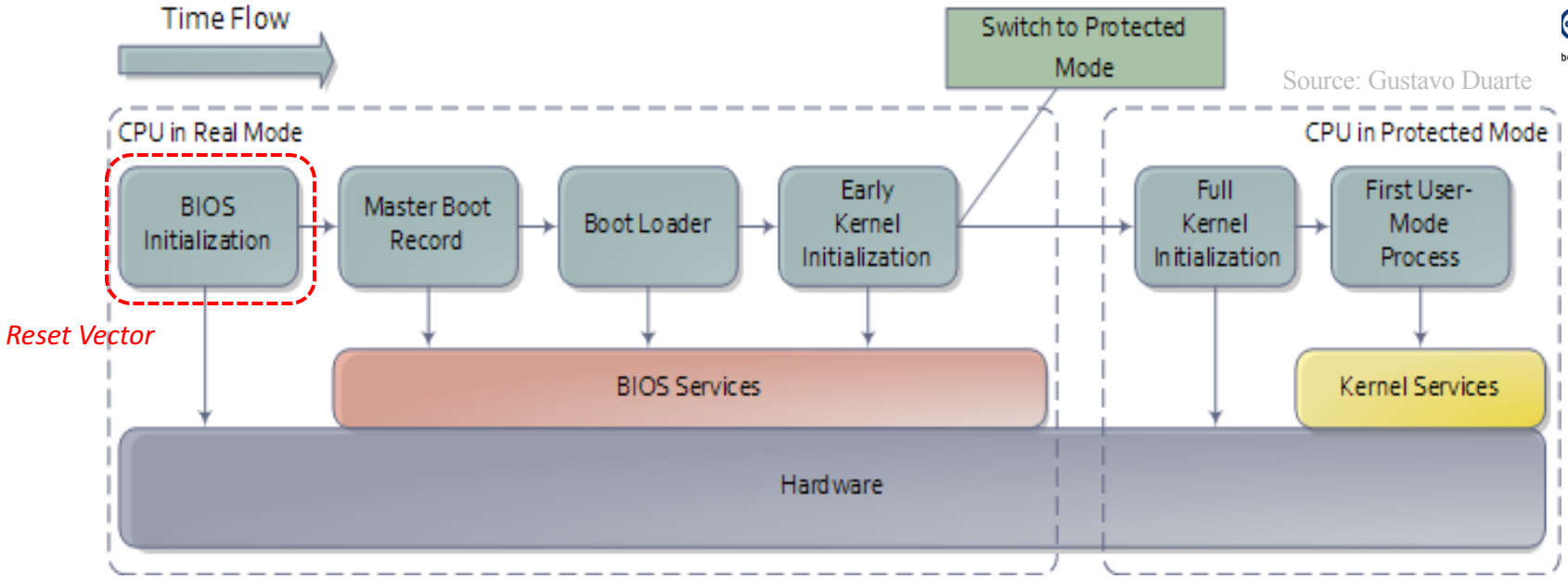
Agenda

- Define embedded system
- Standard Boot Loader
- Embedded Boot Loader
  - Stages of embedded boot loader
  - Available embedded boot loader
- Non IA boot loaders
- Embedded, IA and Compute Continuum
- Reasons and Rational
- IA Boot Loader

Components of a typical Embedded System



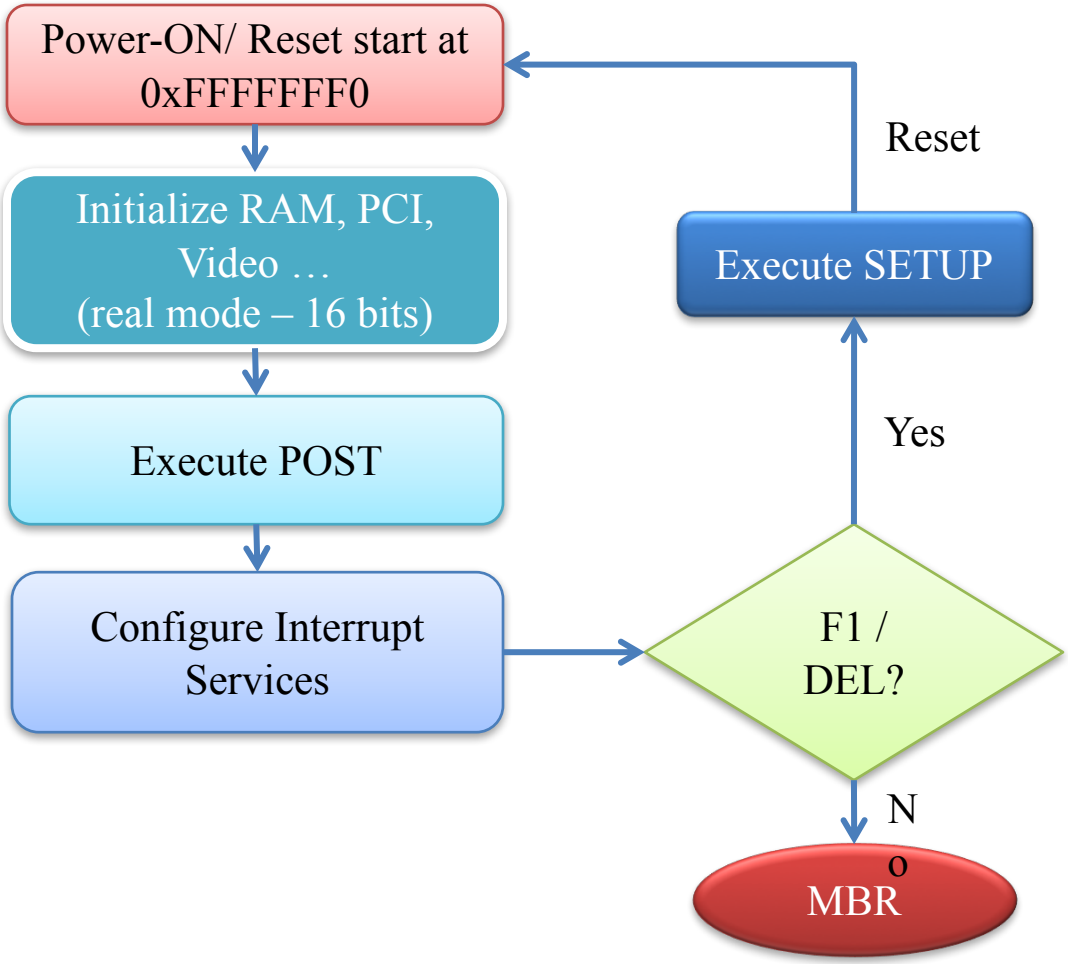
Embedded is a “specialized” computing system.

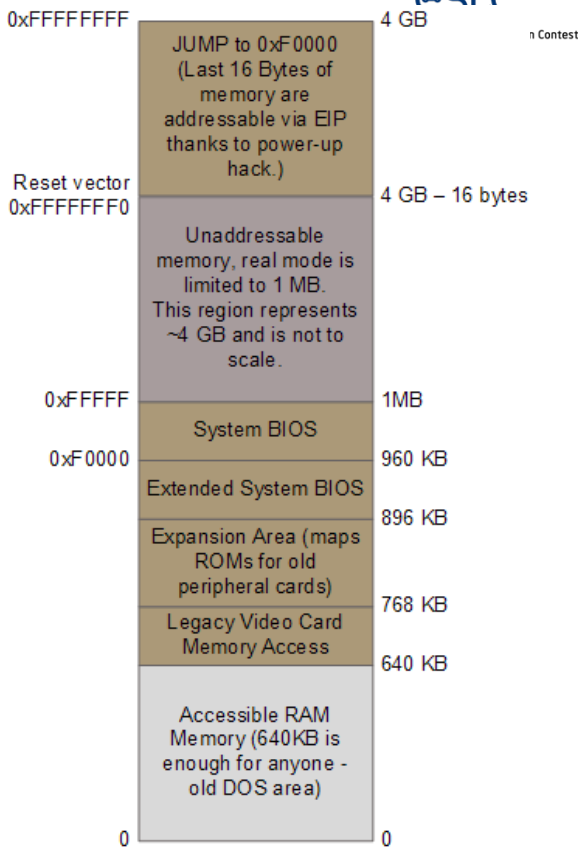


- **BIOS is a layer between OS and hardware**
  - Every platform needs a specific BIOS. But an OS can load on any platform.
  - BIOS provides low level hardware details through the OS by means of ACPI and MP tables besides “Run time ISR’s”.
- **BIOS knows platform in details**
  - Interrupt routing
  - µCode update
  - Chipset workarounds
  - Memory usage…

System BIOS initializes low-level platform details.

How Commercial BIOS Works?





Important memory regions during boot

BIOS layout the memory regions for OS to operate properly.

Possible BIOS Issues

- **Redundant** – Modern Operating Systems initialize hardware by itself and don't use BIOS Interrupt services.
- **Performance** – Commonly runs on real mode.
- **Suboptimal** – Some commercial BIOS configure devices in a suboptimal way.
- **Cost** – Expensive for embedded platforms.

BIOS has inherent constraints for embedded systems!

Embedded Bootloaders

Intel Cup Embedded System Design Contest

- Foundational component for embedded software.
- Possible requirements of embedded bootloader:
  - Small Footprint
  - Easy Portability
  - Fast Boot, or
  - Capability to support certain specific features.
- Development of a bootloader
  - Write from scratch
  - Tailor an existing open source bootloader to suit the need.
  - **Embedded Bootloaders have diversified requirements.**

Embedded Bootloader Architecture

- In general, bootloader architecture depends on:
  - Processor family,
  - Chipsets present on the hardware platform,
  - Boot device, and
  - OS running on the device.
- Effects of the processor family on bootloader architecture:
  - Even when two platforms are based on similar processor cores, the bootloader architecture may differ based on the SoC.
  - An x86 bootloader might need to switch to protected mode to load a kernel bigger than the 1MB real-mode limit.
  - Non x86 embedded platforms cannot avail legacy BIOS services.
  - A bootloader for a device designed around the StrongARM processor has to know whether it's booting the system or waking it up from sleep, because the processor starts execution from the top of its address space.

Embedded Bootloaders require fixed functional characteristic.

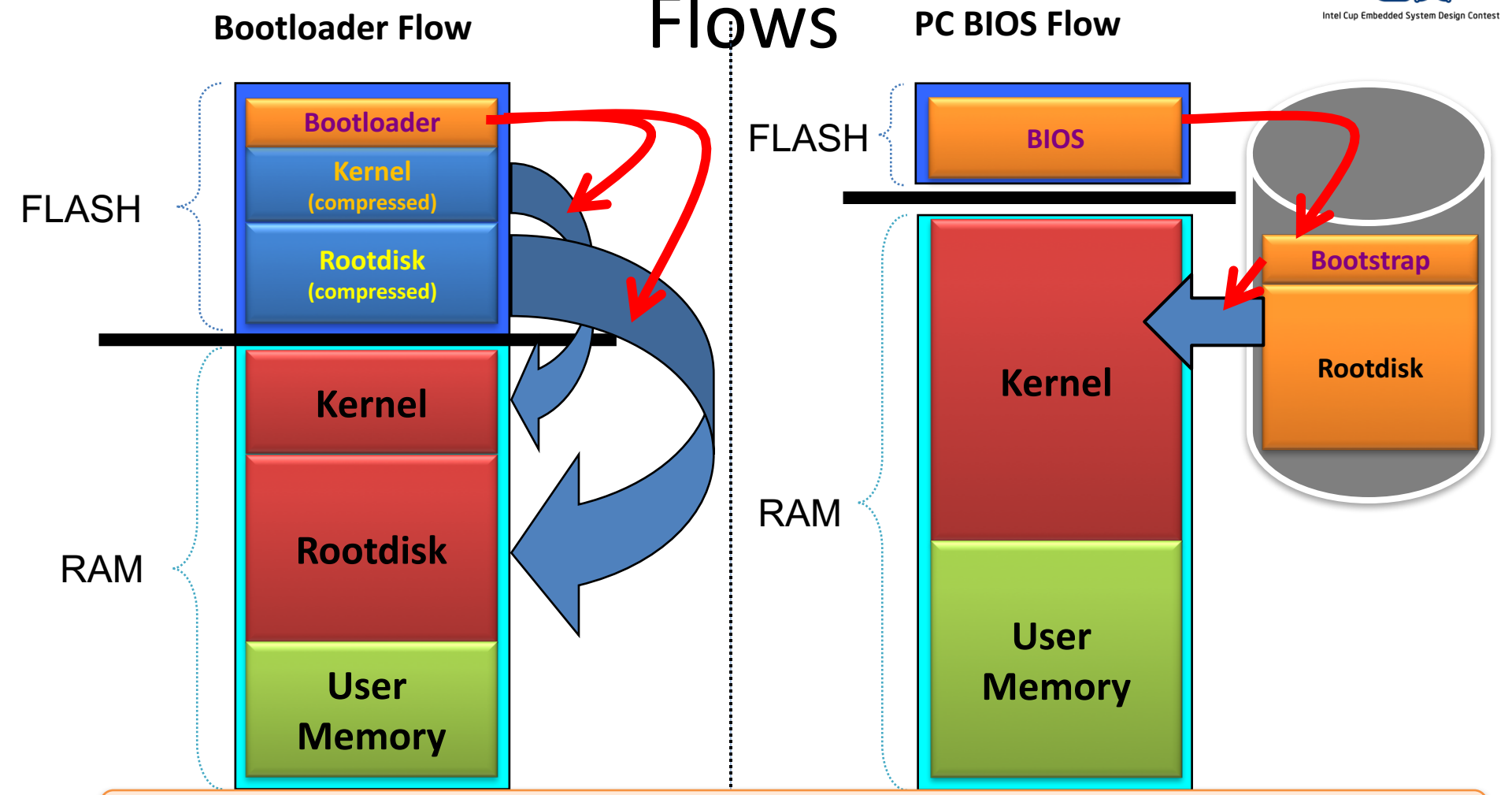
Embedded Bootloaded Features

- At the minimum, a bootloader is responsible for
  - Processor- and board-specific initializations,
  - Loading a kernel,
  - Initial ramdisk into memory, and
  - Passing control to the kernel.

- In addition
  - Might provide BIOS services (POST, firmware download, passing memory layout and configuration information to the kernel).
  - Decryption for encrypted firmware images.
  - Debug monitor to load and debug stand-alone code on to the target.
  - Failure-recovery mechanism to recoup from kernel corruption.
  - Require device-specific modifications.
  - Update capacity through interfaces such as UART, USB or Ethernet.

Embedded Bootloaders need to be extensible.

Deployment Comparison of Boot Flows



Embedded Bootloaders require specialized boot steps.

Bootstrapping for Embedded Bootloader

- Bootstrapping is the mechanism to transfer a bootloader image from the host development system to the target's boot device.
  - Bootstrapping is straightforward on PC-compatible systems.
  - Embedded devices do not have a generic method for bootstrapping.
- The boot suite has to be architected into two steps, each loaded at a different address:
  - The **first step** (the 128-byte image) is part of processor firmware.

    Note: Processor-resident microcode (the first step) cannot function as the bootstrapper because a bootstrapper needs to have the capability to program flash memory. Many types of flash chips can be used with a processor, the bootstrapper code needs to be board-specific.
  - The **second step** lives in the on-chip SRAM, so it can be up to 2KB. This is the bootstrapper.
    - The bootstrapper downloads the actual bootloader image from an external host to the top of flash memory. The bootloader gets control when the processor powers on in normal operation mode.
- Many embedded controller chips do not support a bootstrap mode. Instead, the bootloader is written to flash via a JTAG interface.

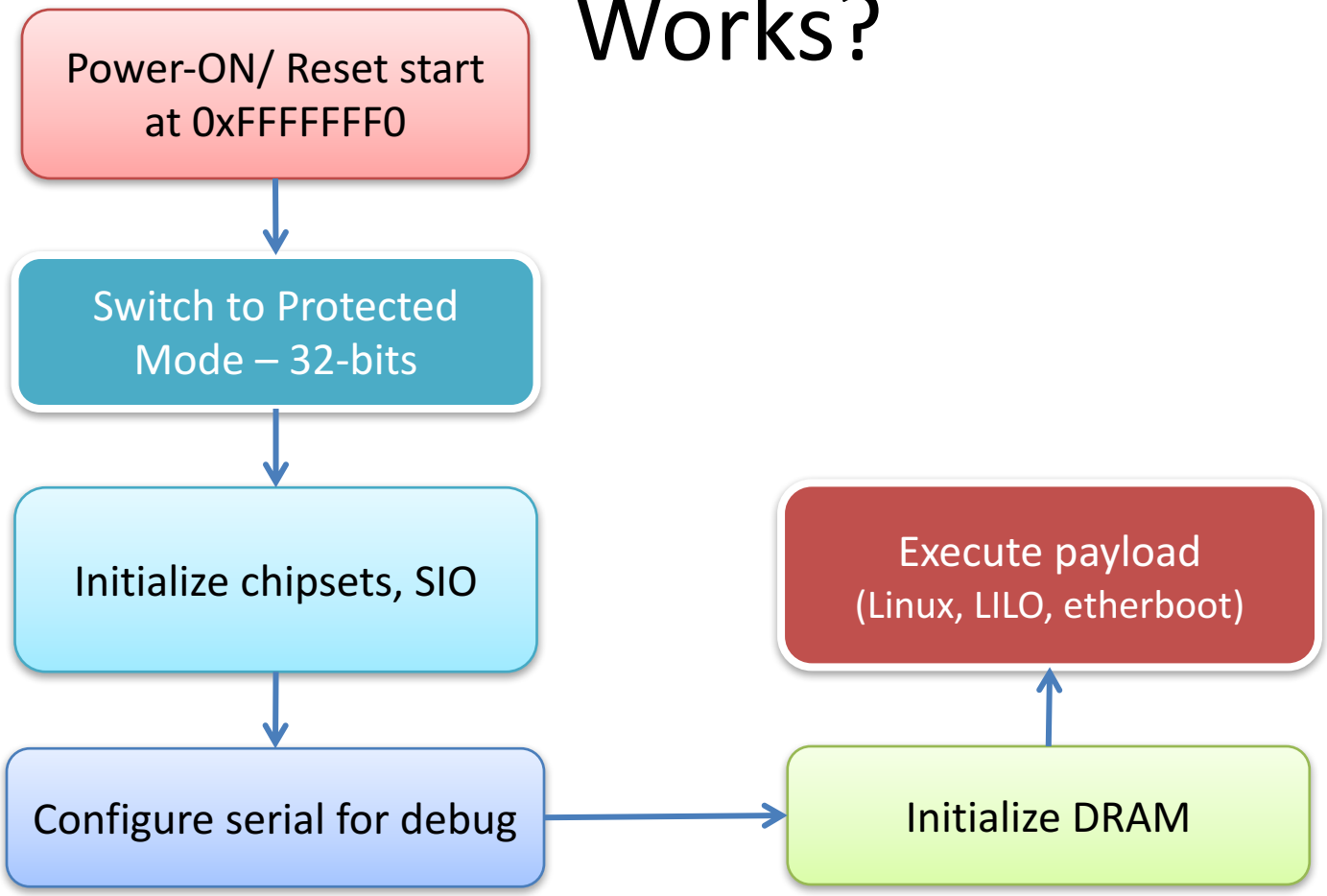
Primarily Embedded Bootloaders do not follow exact steps of System BIOS.

Developing Embedded Bootloader

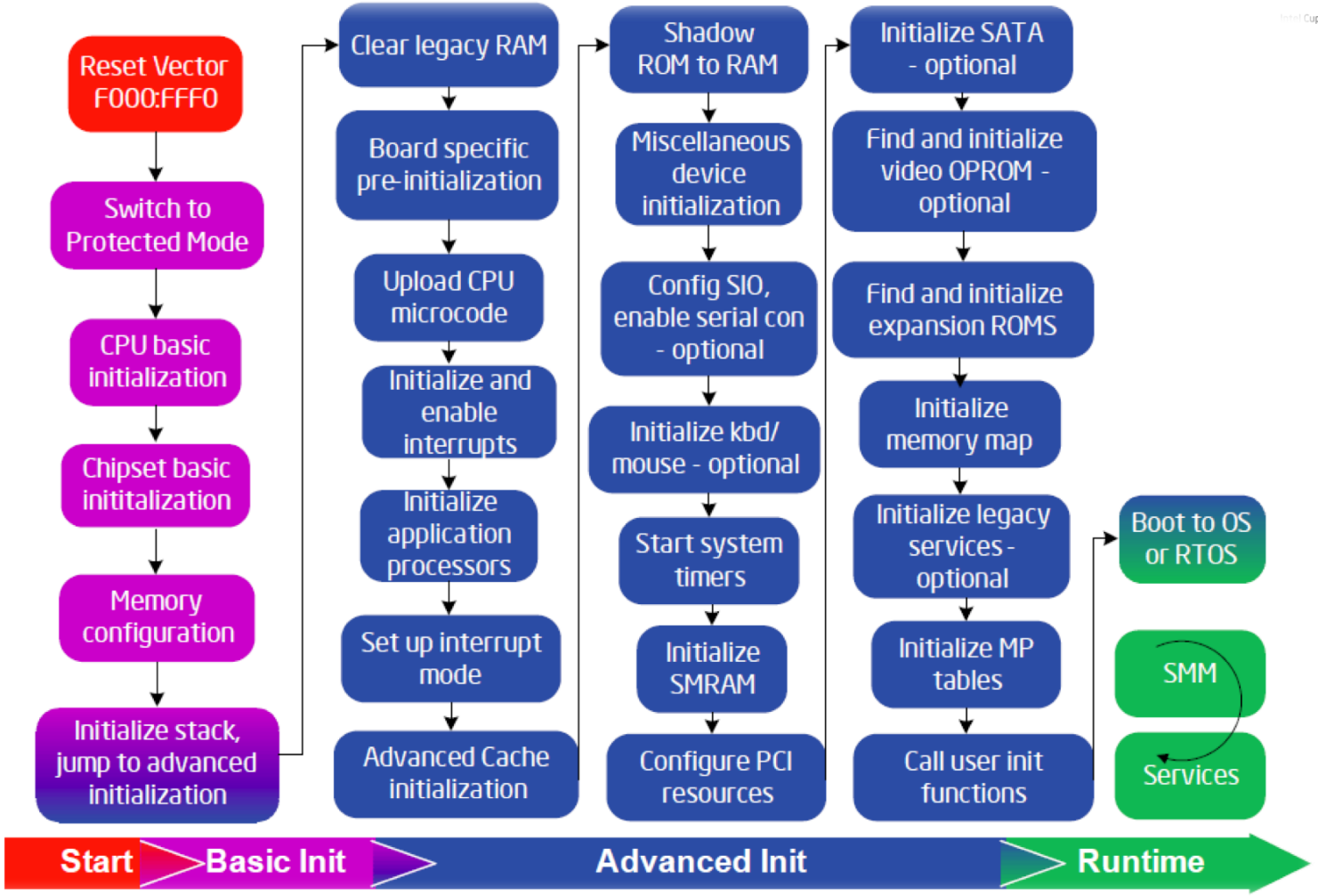
- Understand the hardware environment
  - Memory mapping,
  - I/O mapping, and
  - Interrupt Vectors.
- Understanding program loader:
  - the transition of control from ROM to RAM,
  - initializing the RAM, and
  - loading "C" code in RAM then transferring the control to "C" code.
- Necessary programming and debugging tools.
- Make sure programming and debugging tools are compatible with the hardware environment.
- Board design meets the “proper” spec.
  - Perform some simple tests to verify that the hardware is working.

Proper board design is very important for functional embedded bootloader.

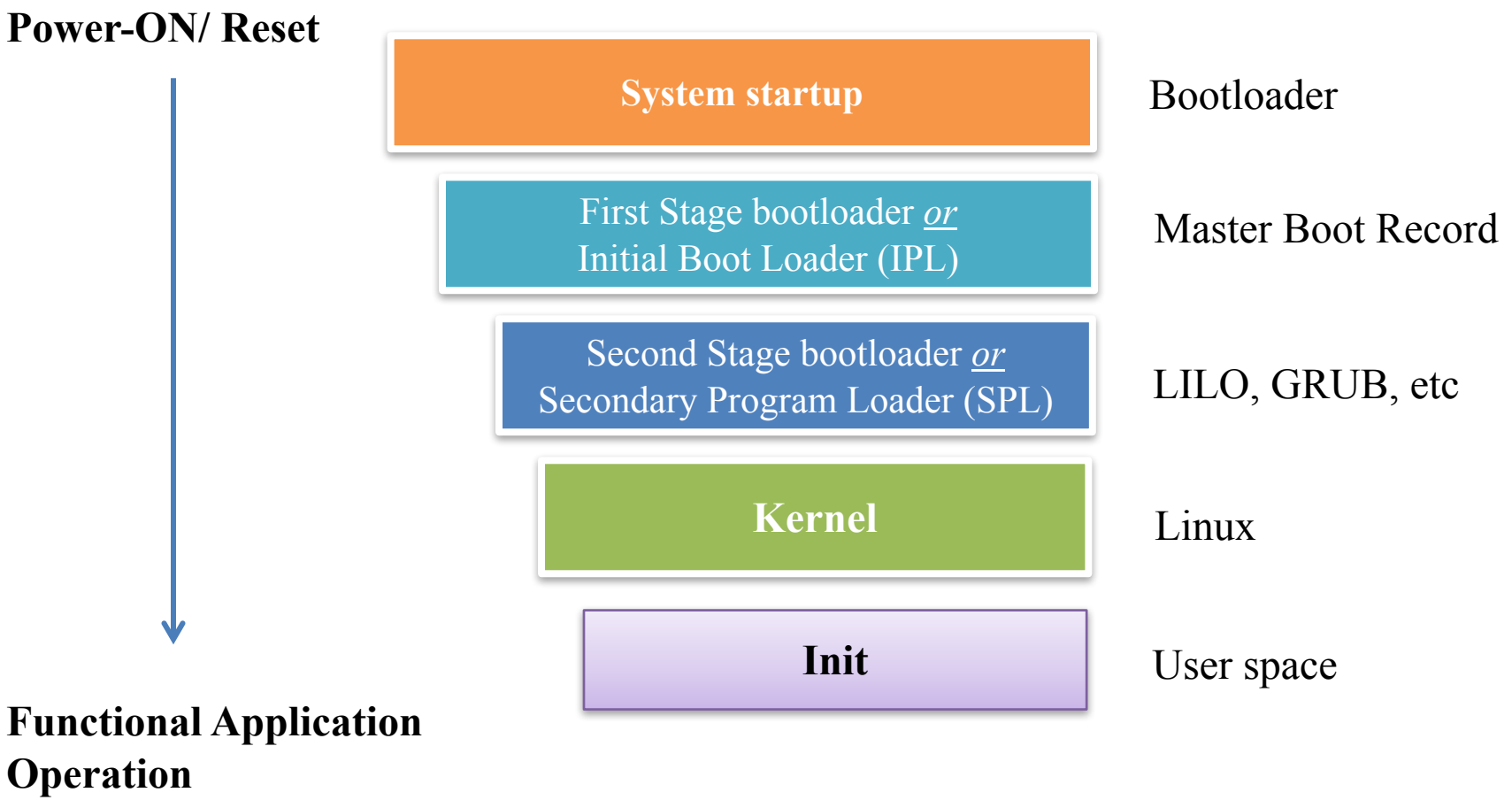
How Typical Embedded Bootloader Works?



Basic Bootloader Architecture/Flow



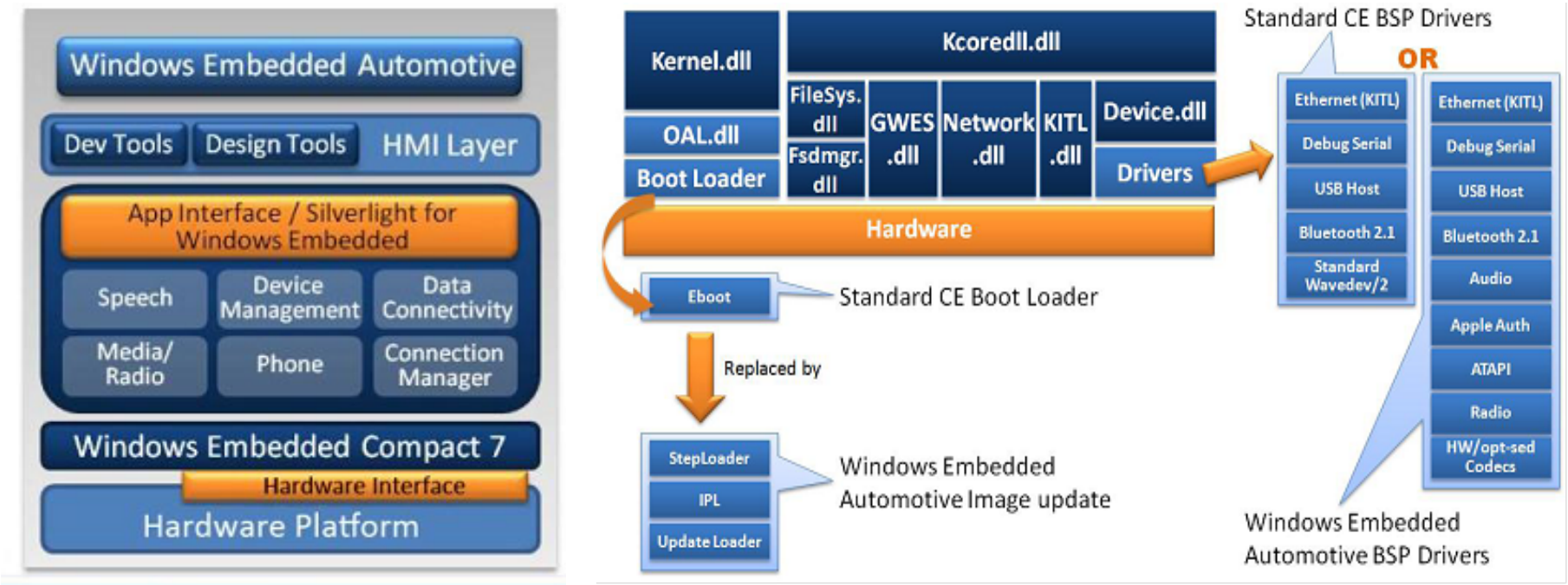
Linux Boot in a nutshell



Multi-stage bootloader brings flexibility!

Source: Trego Ltd.

WinCE and Bootloader



Source: Microsoft

From performance scenario (projected):

(**loadcepc** boot time ≈ 30-40 sec]) **>** (**BIOS_load_image** boot time ≈ 20-30 sec) **>>** (**IPL** < 10 sec)

Today's WinCE has specific bootloader steps.

Available Bootloaders

(relative list, not a comprehensive one)

Intel Cup Embedded System Design Contest

Bootloader	Video Support	Description	Architectures					
			x86	ARM	PowerPC	MIPS	SuperH	m68k
LILO	No	The main disk bootloader for Linux*	x					
GRUB	No	GNU's successor to LILO	x					
Loadlin	No	Loads Linux from DOS	x					
Etherboot	No	Loader to boot systems through Ethernet cards	x					
CoreBoot	No	Linux-based BIOS (LinuxBIOS) replacement	x					
blob	No	Loader from the LART hardware project		x				
PMON	Yes	Loader used in Agenda VR3				x		
sh-boot	No	Main loader of the LinuxSH project					x	
U-Boot	Yes	Universal loader based on PPCBoot and ARMBoot	x	x	x			
RedBoot	Yes	eCos-based loader	x	x	x	x	x	x

Need a flexible and extensible x86 bootloader.

Available Bootloaders for Linux

- **Bootloaders for Multiple Architectures**
  - **U-Boot** (http://sourceforge.net/projects/u-boot/) or (http://www.denx.de/wiki/U-Boot/WebHome)
    - Universal Boot Loader supports PPC, ARM, and others
  - RedBoot (http://www.redhat.com/services/custom/embedded/redboot/)
    - RedHat Embedded Debug and Bootstrap
    - Based on the eCos HAL, RedHat.
    - Capable of flash and network booting of Linux* kernel.
    - Supports ARM, MIPS, PowerPC, and x86
  - Smart Firmware (http://www.codegen.com/SmartFirmware/)
    - Written entirely in ANSI C. Designed to be very easy and fast to port.
    - Supports PowerPC, ARM, x86, MIPS, Sparc, M68k

Available Bootloaders for Linux

Intel Cup Embedded System Design Contest

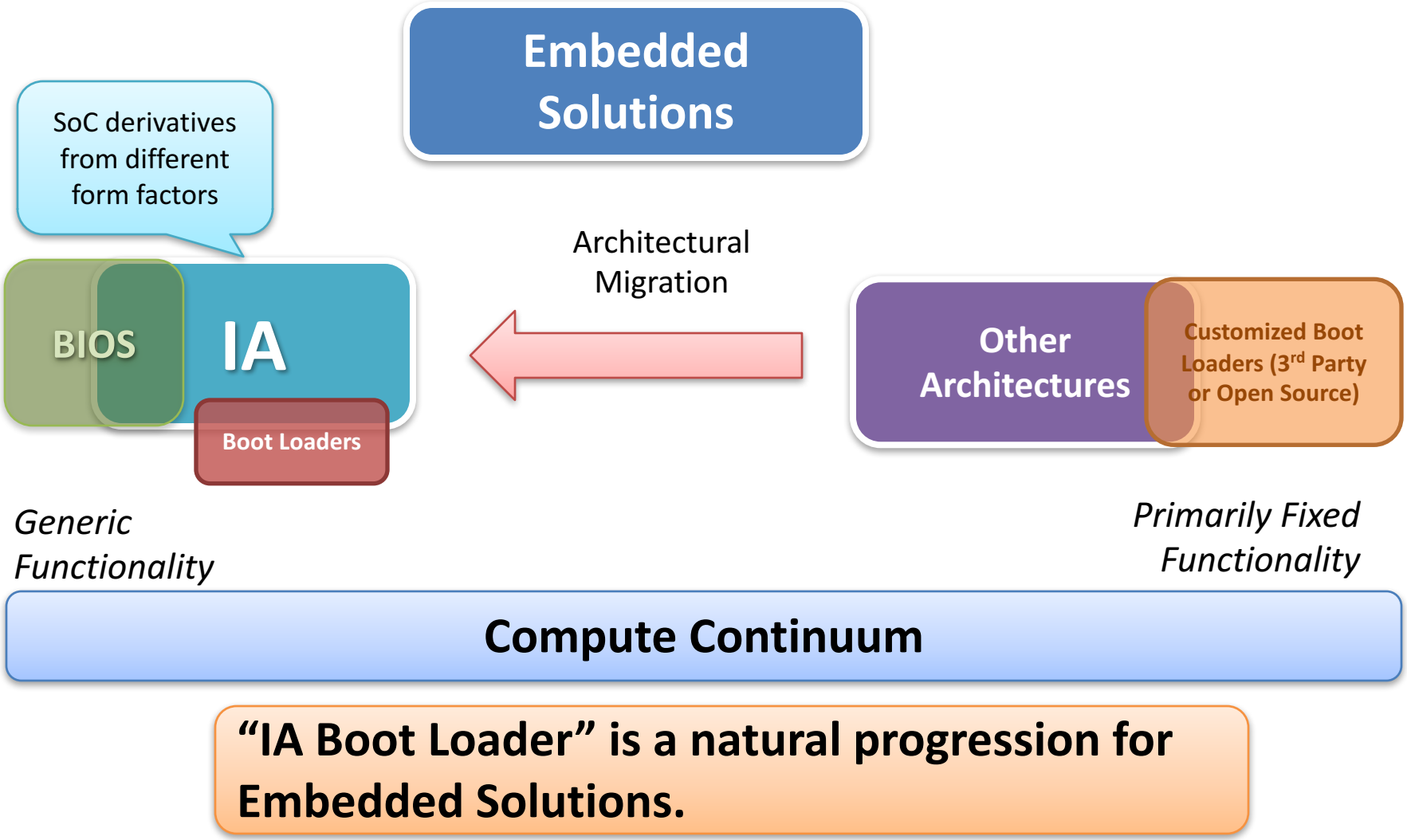
- **Bootloaders for x86 Architectures**
  - LILO (Linux* LOader) (http://lilo.alioth.debian.org/)
  - GRUB (GRand Unified Bootloader) (http://www.gnu.org/software/grub) (planned for ARM) (http://arm-grub.sourceforge.net/)
  - Etherboot (http://etherboot.org/wiki/)
    - Open source network bootloader and substitute of proprietary PXE.
  - CoreBoot (http://www.coreboot.org/)
    - Based on LinuxBIOS (http://www.acl.lanl.gov/linuxbios/index.html)
    - Replacing the normal BIOS with fast boot from a cold start.

Available Bootloaders for Linux

- **Bootloaders for ARM* Architectures**
  - Blob (http://sourceforge.net/projects/blob/)
    - Boot Loader Object for StrongARM based platforms.

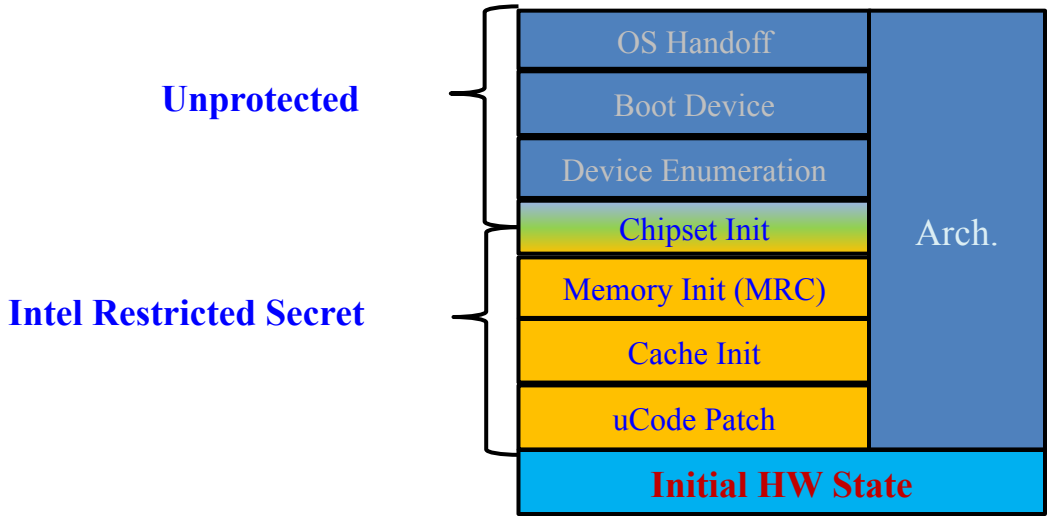
- **Bootloaders for PPC* Architectures**
  - Yaboot (http://yaboot.ozlabs.org/)
    - It works on “New” class PowerMacs (iMac and later) only.

Embedded, IA and Compute Continuum



"IA Boot Loader" is a natural progression for Embedded Solutions.

IA Bootloader Distribution and Scalability

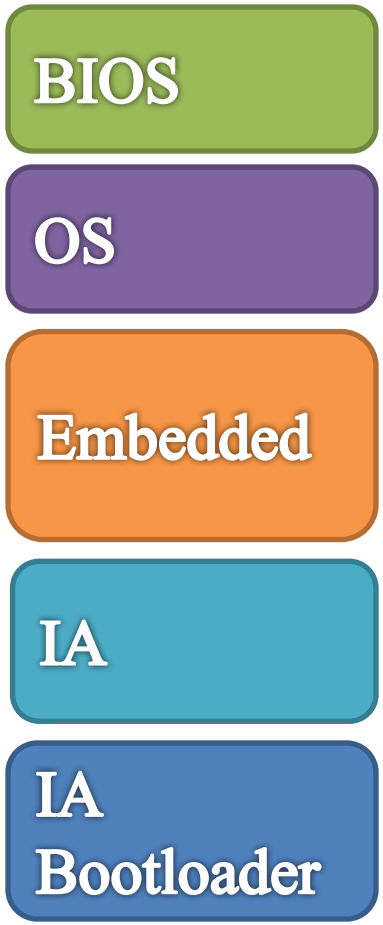


Issues related to IA based bootloader development:

- IA is complex with superior feature set
- IA has inherent IP concerns
- IA initialization is relatively complex
- RS-NDA is required for customer to have access source to modify reference FW for Customer boards

Intel is addressing the embedded bootloader need.

Reasons and Rational



BIOS has inherent constraints for embedded systems!

No common framework for embedded framework to take advantages.. Such as Linux, WinCE, and other embedded OSes.

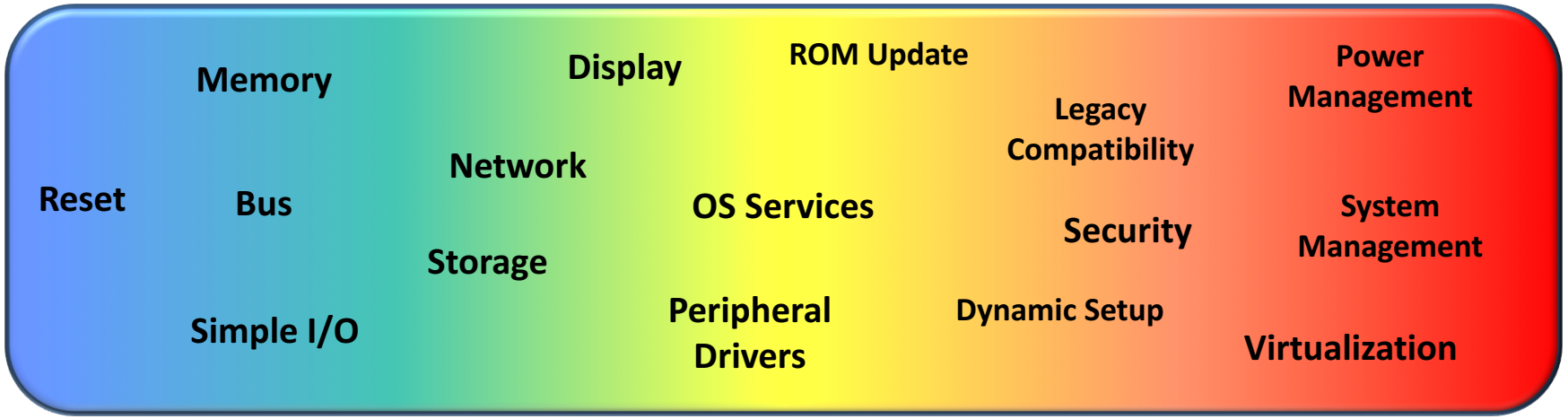
- **Embedded Bootloaders have diversified requirements.**
- **Embedded Bootloaders require fixed functional characteristic.**
- **Embedded Bootloaders need to be extensible.**

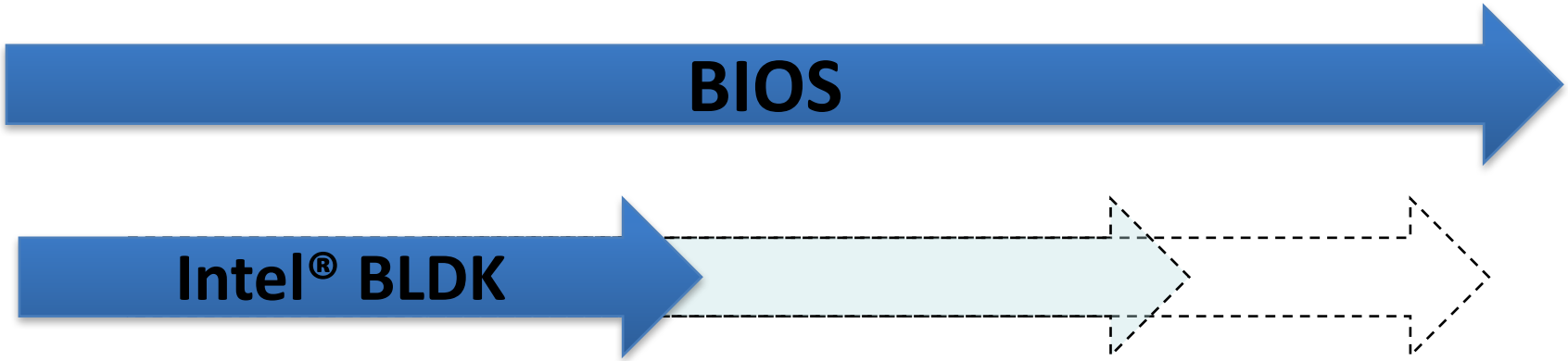
Need a flexible and extensible x86 bootloader.

"IA Boot Loader" is a natural progression for Embedded Solution

Focus is to connect the flexible BIOS and embedded bootloader, and target the generic IA architecture to meet the end-to-end solution for embedded!

Intel & Embedded Bootloader





Intel® BLDK Provides Flexibility to Scale System Initialization for Embedded Systems