Security of BIOS/UEFI System Firmware from Attacker and Defender Perspectives

Section 1. BIOS and UEFI Firmware Fundamentals

Yuriy Bulygin *
Alex Bazhaniuk *
Andrew Furtak *
John Loucaides **

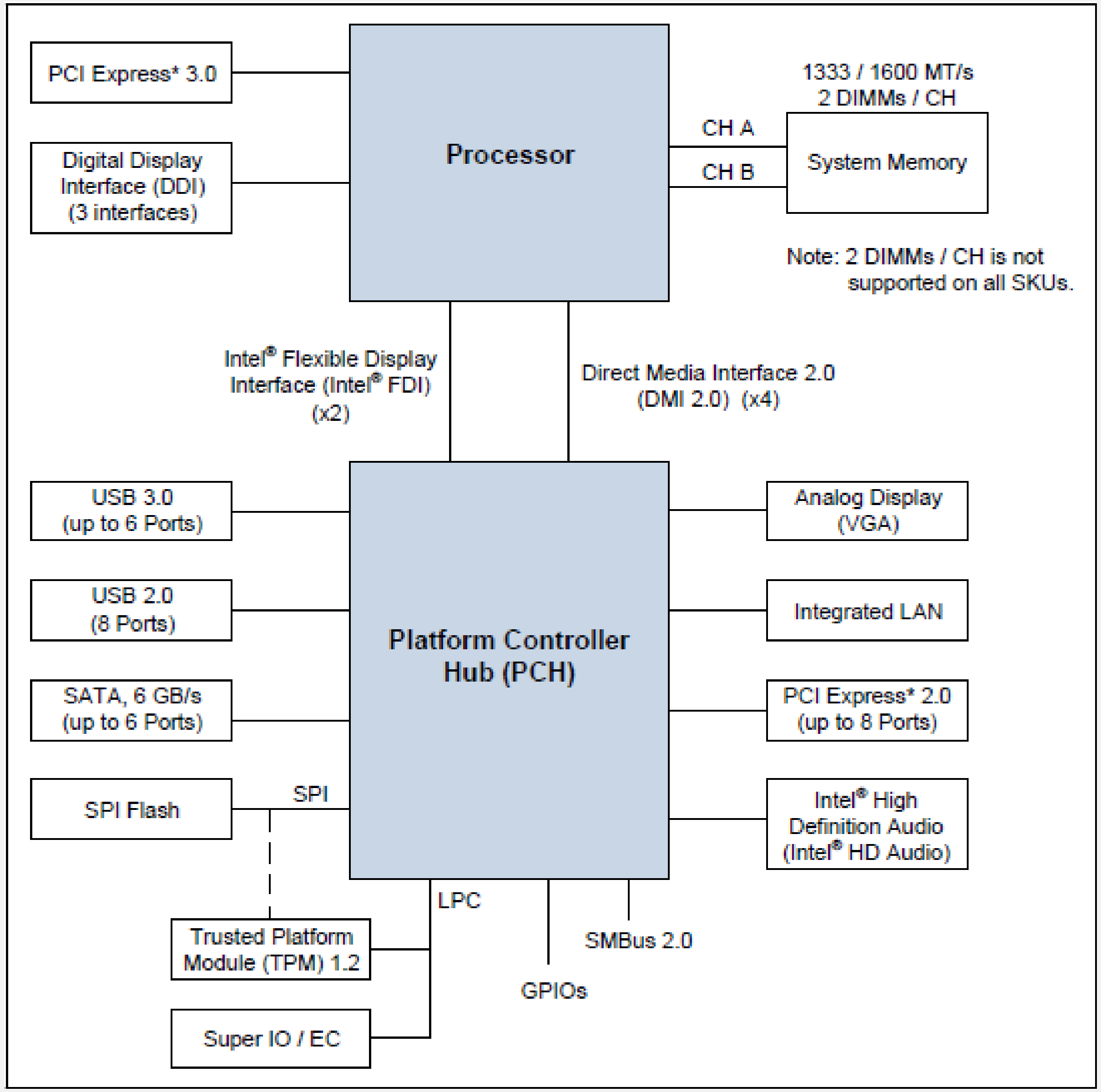
* Advanced Threat Research, McAfee
** Intel

License

Section 1. BIOS and UEFI Firmware Fundamentals

1.1 Introduction to Platform Hardware

Main PC Platform Components

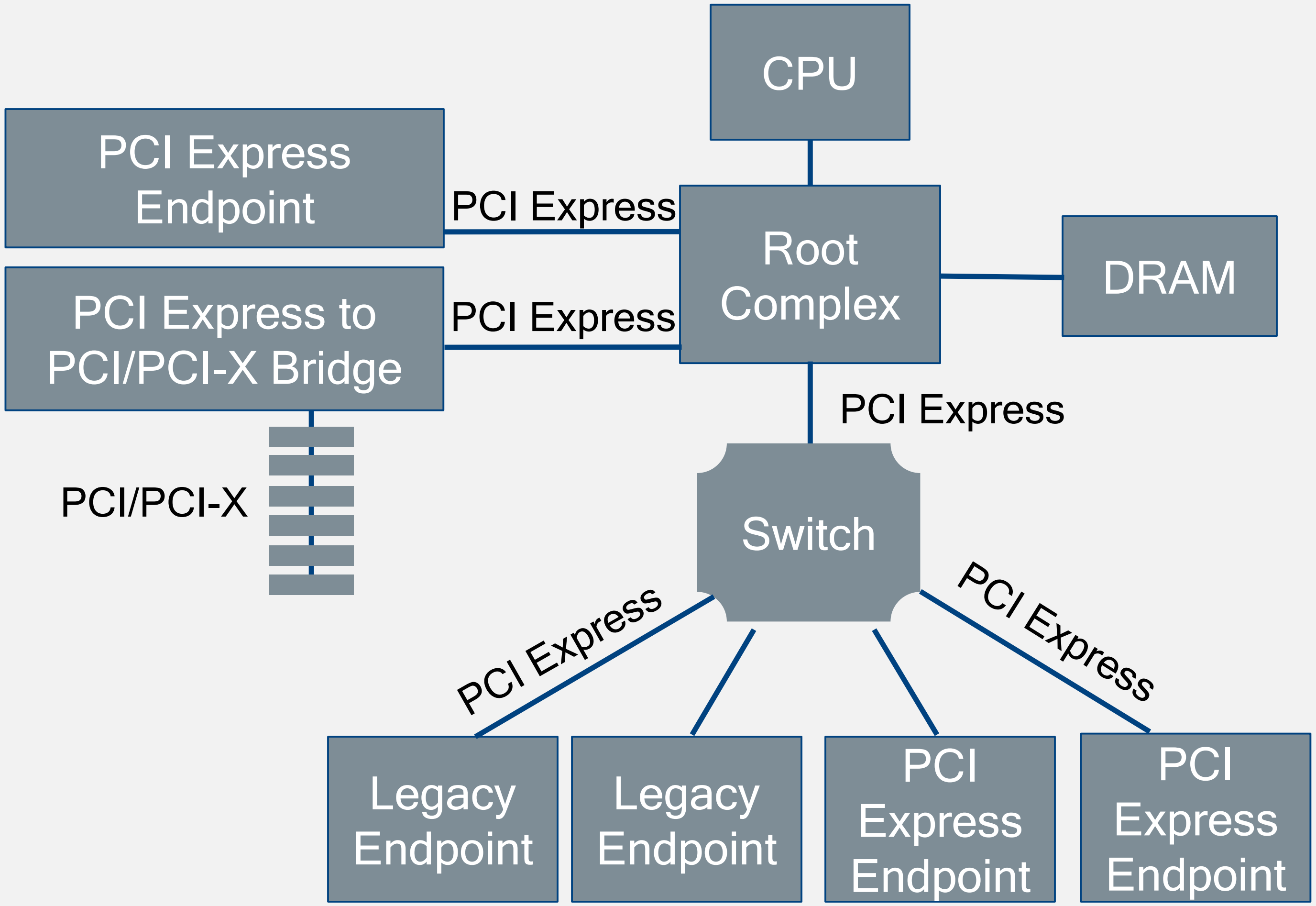

PCI Express* 3.0
Digital Display Interface (DDI) (3 interfaces)
Processor
1333 / 1600 MT/s
2 DIMMs / CH
CH A
CH B
System Memory
Note: 2 DIMMs / CH is not supported on all SKUs.
Intel® Flexible Display Interface (Intel® FDI) (x2)
Direct Media Interface 2.0 (DMI 2.0) (x4)
USB 3.0 (up to 6 Ports)
USB 2.0 (8 Ports)
SATA, 6 GB/s (up to 6 Ports)
SPI Flash
SPI
Platform Controller Hub (PCH)
Analog Display (VGA)
Integrated LAN
PCI Express* 2.0 (up to 8 Ports)
Intel® High Definition Audio (Intel® HD Audio)
LPC
Trusted Platform Module (TPM) 1.2
Super IO / EC
GPIOs
SMBus 2.0

Source: 4th Generation Intel Core Processor Family Datasheet

PCI Express Overview

- PCI Express Fabric consists of PCIe components connected over PCIe interconnect in a certain topology (e.g. hierarchy)
- *Root Complex* is a root component in a hierarchical PCIe topology with one or more PCIe *root ports*
- Components: *Endpoints* (I/O Devices), *Switches*, PCIe-to-PCI/PCI-X *Bridges*
- All components are interconnect via PCI Express Links
- Physical components can have up to 8 physical or virtual *functions*
- Some endpoints are *integrated* into Root Complex

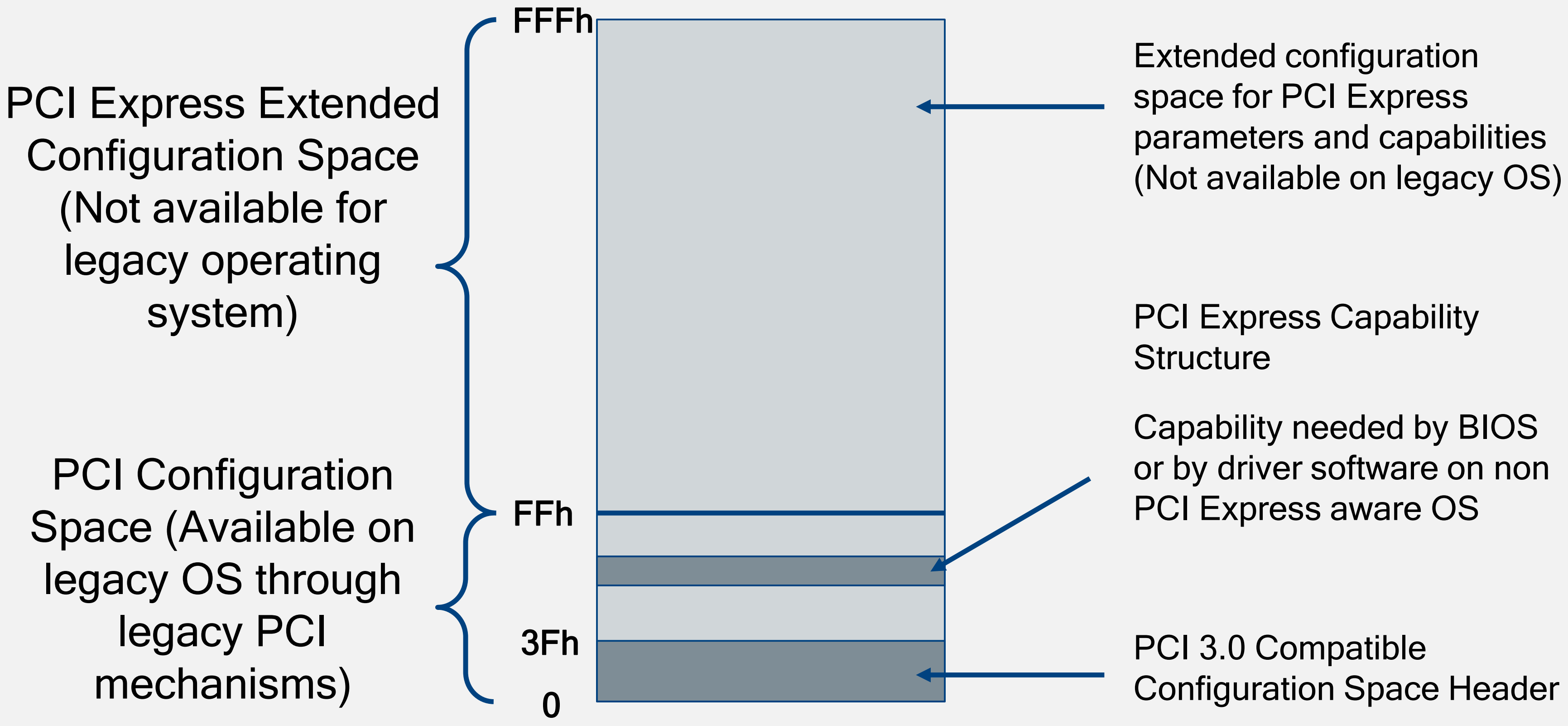
PCI Express Overview



Reference: https://www.mindshare.com/files/ebooks/PCI%20Express%20System%20Architecture.pdf

PCIe Configuration Space Layout

Each *function* has its own configuration (256 bytes of PCI config space and 4kB of PCIe extended config space)



PCI Express Extended Configuration Space (Not available for legacy operating system)
PCI Configuration Space (Available on legacy OS through legacy PCI mechanisms)
FFFh
FFh
3Fh
0
Extended configuration space for PCI Express parameters and capabilities (Not available on legacy OS)
PCI Express Capability Structure
Capability needed by BIOS or by driver software on non PCI Express aware OS
PCI 3.0 Compatible Configuration Space Header

Reference: PCI Express Base Specification Revision 3.0

PCIe Configuration Space Access

SW uses one these mechanisms to access config space:

1. Legacy configuration access via *control* **`CF8h`** & *data* **`CFCh`** processor I/O ports
   - PCI config register address

     8 * 100h per device

     ```
     bus << 16 | device << 11 | function << 8 | offset & ~3
     ```

     32* 8 * 100h per bus

     100h bytes of CFG header

   - **`CF8h`** ← **`1<<31 | bdf_address`**
   - Read data from or write data to port (**`CFCh + off[1:0]`**)

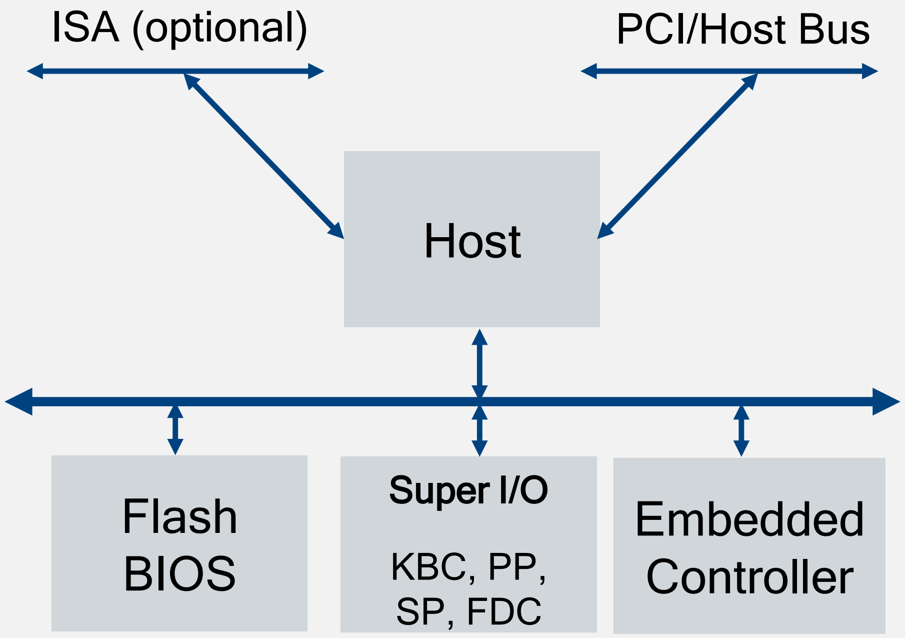
2. Enhanced configuration access mechanism (ECAM) to PCIe extended configuration registers

Low Pin Count (LPC) Interface

- LPC specification
- Substitutes Industry Standard Architecture (ISA) X-bus
- 7 required signals (LAD3-0,LFRAME,LRESET,LCLK), 6 optional
- 33 MHz PCI compatible host controller driven clock
- LPC controller is integrated into PCH or ICH
- Host decodes IO or MMIO cycles on PCI to LPC cycles:
  - Memory Rd/Wr
  - I/O Rd/Wr
  - DMA Rd/Wr (via Intel 8237 DMA controller)
  - Bus Master Memory Rd/Wr
  - Bus Master I/O Rd/Wr
  - Firmware Memory Rd/Wr

Low Pin Count (LPC) Interface

1. Firmware Hub: legacy Boot ROM (BIOS)
2. Discrete Trusted Platform Module (TPM)
   - MMIO 0xFED4xxxx
3. Super I/O (Floppy Disk Controller, PS/2 KBC, serial and parallel ports)
   - KB IO 60h/64h
   - FDD IO 3F0h-3F7h …
   - SP IO 3F8h-3FFh …
   - PP IO 378h-37Fh …
4. Embedded Controller
   - IO 62h/66h
5. Audio, AC'97, MIDI…



Reference: LPC specification

Serial Peripheral Interface (SPI)

1. 4-pin synchronous serial interface
   - SCLK, MOSI, MISO, CSn/SSn (1 per slave device); optional IOx signals for fast read modes
   - Master - Slave protocol: 1 master, N slaves
   - Lower cost for system flash (BIOS) than FWH on LPC
2. SPI Controller is in PCH or ICH
   - Supports JEDEC Serial Flash Discoverable Parameter (SFDP) to query capabilities of SPI flash device
   - Boot BIOS straps (BBS) define if firmware is in LPC or SPI
3. SPI peripherals: sensors, serial NOR flash memory, MMC/SD...
4. Devices connected to SPI bus in chipset:
   - Flash Memory Devices
   - TPM over SPI (CS2#) on newer systems

Chipset SPI Controller

1. PCH supports 3 SPI flash devices up to 16MB each
2. *Descriptor* and *non-descriptor Modes*
   - CS0# must be flash memory device with a valid *Flash Descriptor* (FD)
   - Descriptor mode is required on PCH based platforms
   - Flash descriptor describes contents of the flash memory
3. *Hardware* and *Software Sequencing* operational modes
   - Hardware sequencing preprograms SPI bus cycles and CS0#
   - Software sequencing allows software to choose SPI cycles and CS

System Flash Memory

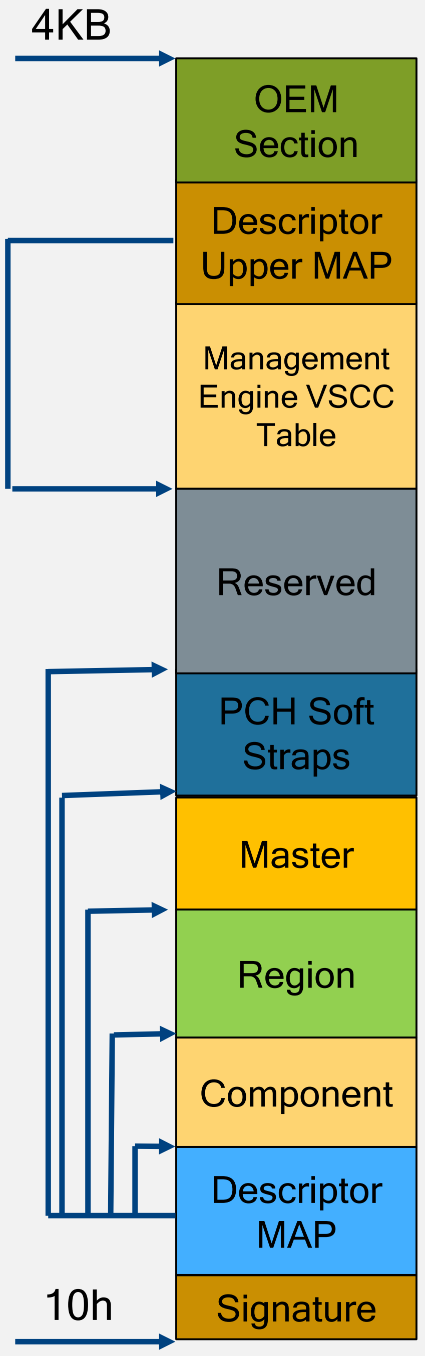
1. *Direct Access* (e.g. `FFFFFFFFh` PA mapped to SPI direct read cycle), *Program Register Access* (SW programs Flash Linear Address to SPI MMIO registers)
2. For multiple masters SPI flash should support HW seq
3. FLA `FFFFFFh` maps to the top of Flash device
4. Erase cycles set all Fs: 0 → 1
5. In descriptor mode, SPI flash memory devices contain multiple ranges

Region	Content
0	Flash Descriptor
1	BIOS
2	Intel Management Engine
3	Gigabit Ethernet
4	Platform Data

Reference:
[Intel 7 Series Chipset PCH datasheet]

SPI Flash Descriptor

1. Region 0 at FLA `0 – FFFh` (4 KB)
2. Signature: **`0FF0A55Ah`** at `10h` LBA
3. Contains the following sections:
   - *Component*: flash device configuration
   - *Region*: describes other regions
   - *Master*: defines Rd/Wr Access Control table
   - *CPU and PCH soft straps*
   - *ME VSCC Table*: JEDEC ID & VSCC info
   - *OEM*: reserved for OEM use
4. Access Control table defines which masters (CPU, ME, GbE) can access regions (FD has to be write-protected)



Reference: Intel 7 Series Chipset PCH datasheet

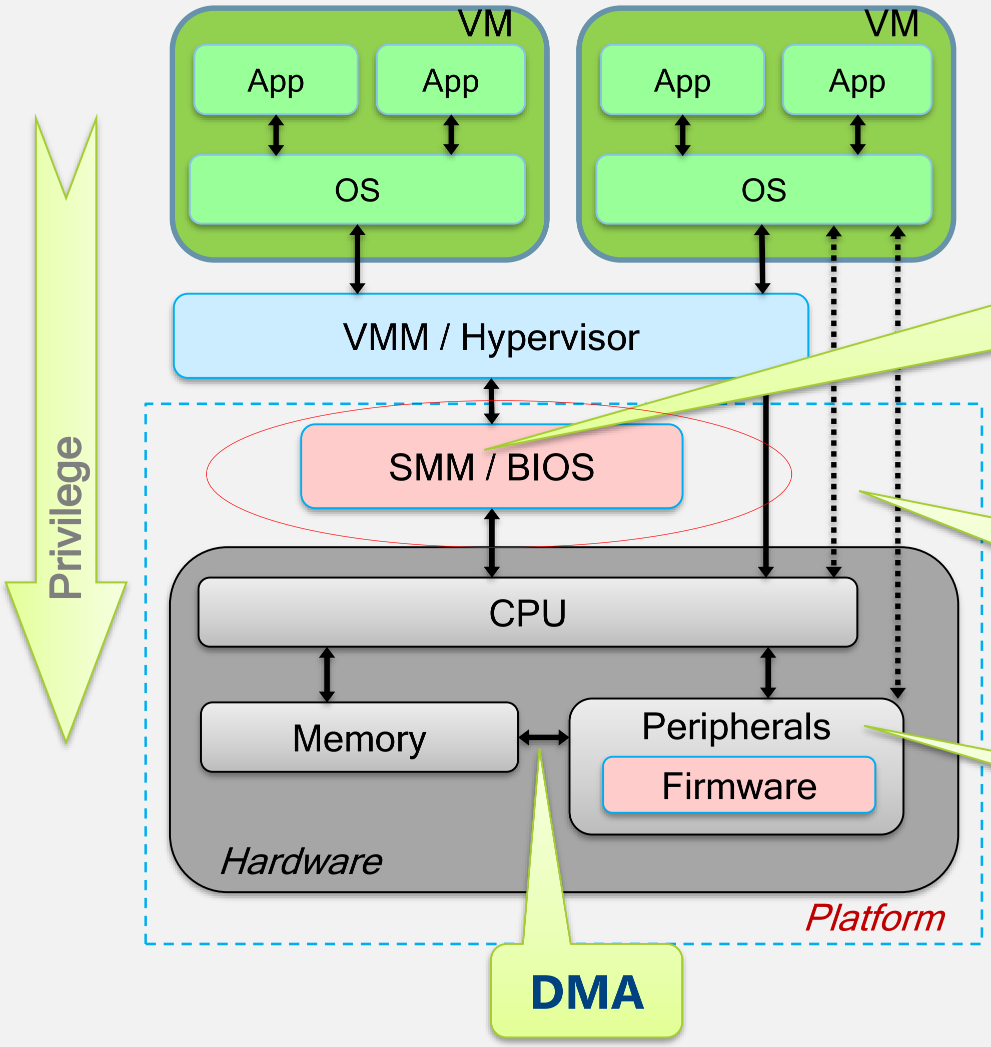
System Flash Security

- Chipset (SPI controller) based protections
  1. SMM based BIOS Write Protection: write-protects entire BIOS region from software other than SMI handler firmware executing in SMM
  2. SPI Protected Range registers (PR0-PR4): read/write protection of SPI flash regions based on FLA for program register access
  3. Flash Descriptor based access control: defines read/write access to each flash region by each master
- Firmware may use SPI flash chips write protection (WP#)

System Management Bus (SMBus)

1. SMBus 2.0 specification
2. 2-pin interface (SMBDATA, SMBCLK), opt. SMBALERT#
3. SMBus host controller to communicate over SMBus:
   - DIMM Serial Presence Detect (SPD) EEPROM
   - DIMM Thermal Sensors
   - EC or BMC (host-to-uC, uC-to-sensors, BMC-to-NIC)
   - Commands: Read/Write Byte/Word, Send/Receive Byte, Process Call..
   - Can operate in I^2C mode
4. Intel ME has SMBus controller(s)
5. PCIe connectors include SMBus pins for side band management
6. DMTF Alert Standard Format (ASF) sensors
7. PMBus for power supplies

Where is system firmware?



System firmware (BIOS/SMM, EFI) & OS/VMM share access, but not trust

Hypervisor can grant VM direct hardware access

A specific Peripheral may have its own processor, and its own firmware, which is undetectable by host CPU/OS.

Source: Symbolic execution for BIOS security

Hardware Boot Sequence

1. External power supplied (e.g. 12V) to the system and settles
2. Power is driven in a sequence to multiple processor and chipset voltage rails
3. Platform clocks are derived from external clock and oscillators
4. Processor reset signal is de-asserted
5. Power management logic in the CPU executes reset sequence (samples fuses, handshake with the PCH, reads Power-On configuration etc.)
6. Power management logic brings cores out of reset
7. Processor cores execute reset microcode (initializes x86 state, parses FW interface table, etc.)
8. Finally, reset microcode fetches the first instruction at physical address `FFFFFFF0h` known as *reset vector*

x86 Reset Vector

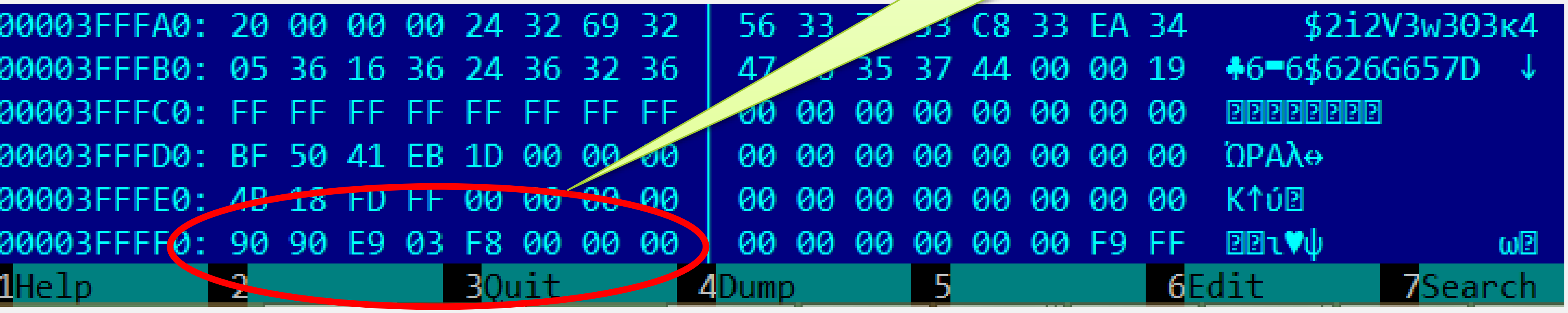
1. CPU starts in *Real-Address Mode* (or *Real Mode*)
   - CS = `F000h`, CS limit = `FFFFh`, EIP = `FFF0h`
   - So reset vector should be at `F000:FFF0 = (F000h << 4) + FFF0h = 000FFFF0h`
2. In 8086 BIOS ROM was mapped under 1MB. Now CPU/chipset map larger BIOS ROM under 4GB (`FFFFFFFFh`)
3. CPU asserts upper 12 address bits high by programming reset value of CS descriptor cache CS base to `FFFF0000h`
4. So reset vector is CS base + EIP = `FFFF0000h + FFF0h = FFFFFFF0h`
5. Until firmware executes FAR JMP to reload CS with `F000h` to stay in real mode under 1MB or enter protected mode early

Reset Vector Decode

1. CPU decodes addresses `FFFxxxxxh` down to system bus (DMI) to chipset
2. Chipset decoded memory reads to SPI flash controller (or Firmware Hub) per configurable decoding rules
3. Memory reads are claimed by SPI flash controller which sends them as direct SPI read cycle to `FFFFF0h` flash address (FLA)
4. This ensures that memory reads (i.e. reset vector code fetch) to mapped BIOS ROM range are directed to firmware ROM (e.g. SPI flash memory device)

x86 Reset Vector Example

Reset vector: near jump to the rest of the BIOS Boot Block



```
00003FFFA0: 20 00 00 00 24 32 69 32  56 33 77 33 C8 33 EA 34      $2i2V3w303κ4
00003FFFB0: 05 36 16 36 24 36 32 36  47 36 35 37 44 00 00 19  ♣6▬6$626G657D  ↓
00003FFFC0: FF FF FF FF FF FF FF FF  00 00 00 00 00 00 00 00  ▯▯▯▯▯▯▯▯
00003FFFD0: BF 50 41 EB 1D 00 00 00  00 00 00 00 00 00 00 00  ΏPAλ↔
00003FFFE0: 4B 18 FD FF 00 00 00 00  00 00 00 00 00 00 00 00  K↑ύ▯
00003FFFF0: 90 90 E9 03 F8 00 00 00  00 00 00 00 00 00 F9 FF  ▯▯ι♥ψ          ω▯
1Help  2        3Quit    4Dump    5        6Edit    7Search
```

NOP

NOP

JMP SHORT Rel16 ; EarlyBSPInit or SEC entry-point

Firmware Start

1. CPU start executing system firmware code at reset vector which is mapped to firmware ROM
2. Reset vector contains a jump to the initial firmware known as *[Initial] Boot Block*
3. Boot block consists of most of the *early platform initialization* functionality (*SEC* and *PEI* phases for UEFI based firmware)
4. Boot block executes early pre-memory initialization code, and memory initialization code (*Memory Reference Code*)

Firmware Boot Sequence – Early Boot

1. Firmware in multiprocessor system chooses *Bootstrap Processor (BSP)*. Other processors are in *Wait-For-SIPI*
2. Firmware on BSP initializes BSP CPU
   - Loads early CPU microcode update on BSP
   - Initializes Local APIC
   - Ensure all APs are in Wait-For-SIPI or broadcast INIT IPI
   - Initializes IDT/GDT and switches to flat 32-bit protected mode (CR0.PE)
   - Configure fixed and variable memory types (MTRR)
   - Enable cache on BSP (CR0.CD/CR0.NW)
   - Enable *No-Evict Mode (NEM)* to set up *Cache-As-RAM (CAR)* for stack and code
   - Initialize basic chipset interfaces (Root Complex, SMBus controller, MMCFG…)

Firmware Boot Sequence – DRAM Init

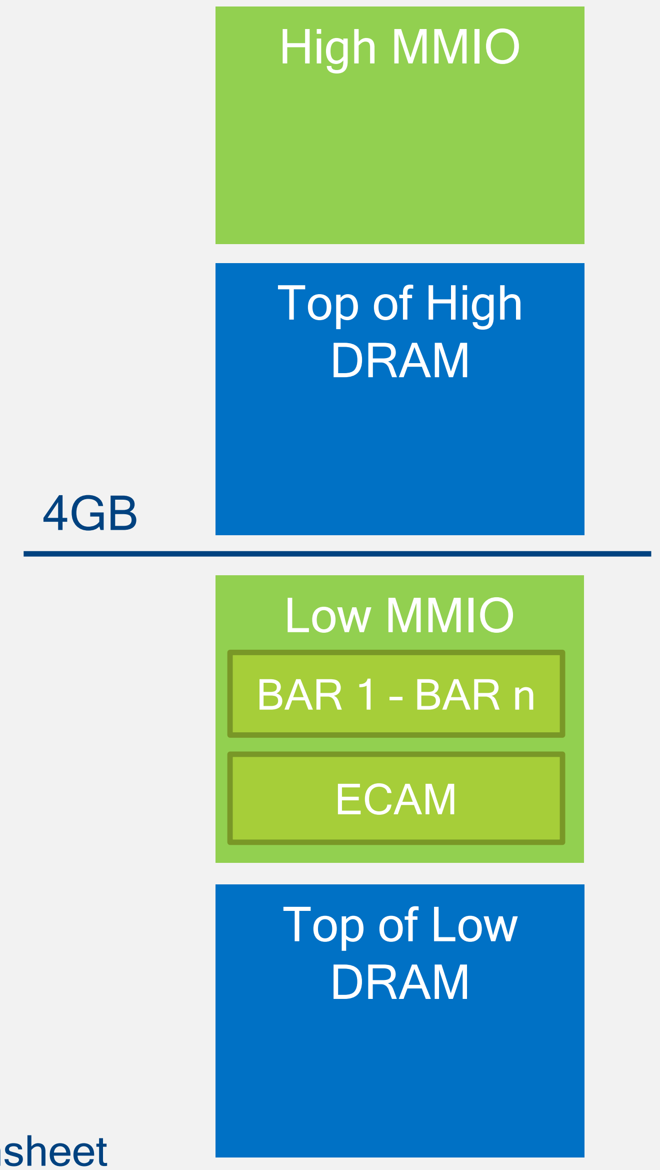
3. FW performs SoC specific DRAM initialization (MRC)
   - Read DIMM parameters from SPD EEPROM over SMBus
   - Trains DDR busses
   - Initializes memory controller
   - Initializes memory map, stolen ranges
4. Memory *shadowing*: FW Copies the rest of the firmware to DRAM
   - Decompression code decompresses FV from SPI flash and copies below 1MB
   - Exit CAR mode and transition to shadowed code in DRAM
   - Modify *Programmable Attribute Map (PAM)* registers to decode to DRAM

Firmware Boot Sequence – DRAM Init

5. Wake other processors (APs)
    - APs will execute wake-up init code
6. SMRAM Init
    - BSP copies *SMM relocation* handler at `0x30000p`
    - Allocates TSEG and copies run-time SMI handler
    - Sends SMI to relocate SMI handler to TSEG
    - SMM relocation handler will configure SMRR
7. Continue with chipset initialization (*DXE* for UEFI)
    - USB, SATA …
8. Enumerate and initialize PCI devices
    - PCI devices enumeration
    - Allocate MMIO spaces for device MMIO BARs
    - Load and execute PCI Expansion/Option ROMs

Memory Map: System View

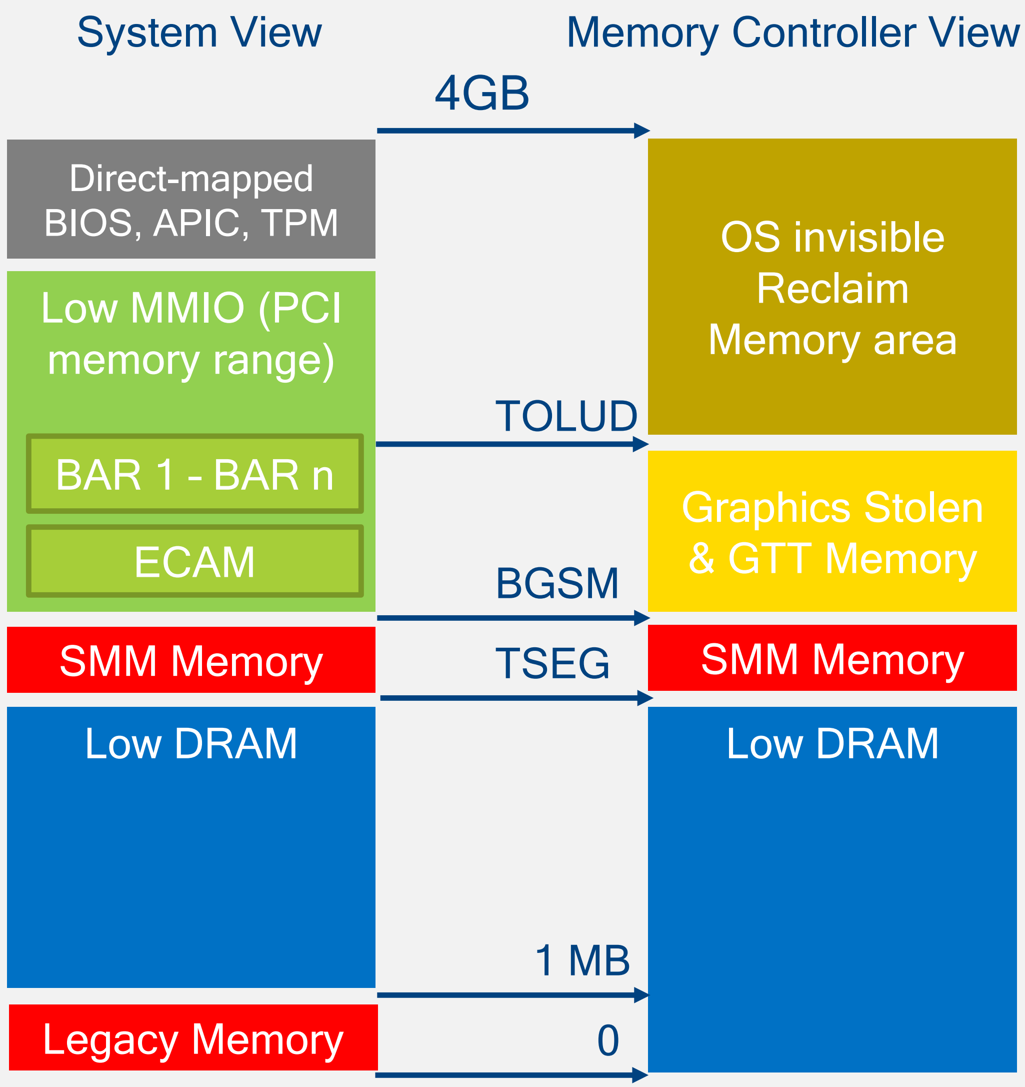
- Low DRAM
- Memory vs MMIO
- Low MMIO
- 4GB boundary
- High DRAM
- High MMIO



Reference: 4th Gen Intel Core Processor Family Datasheet

Memory Map (Below 4GB)

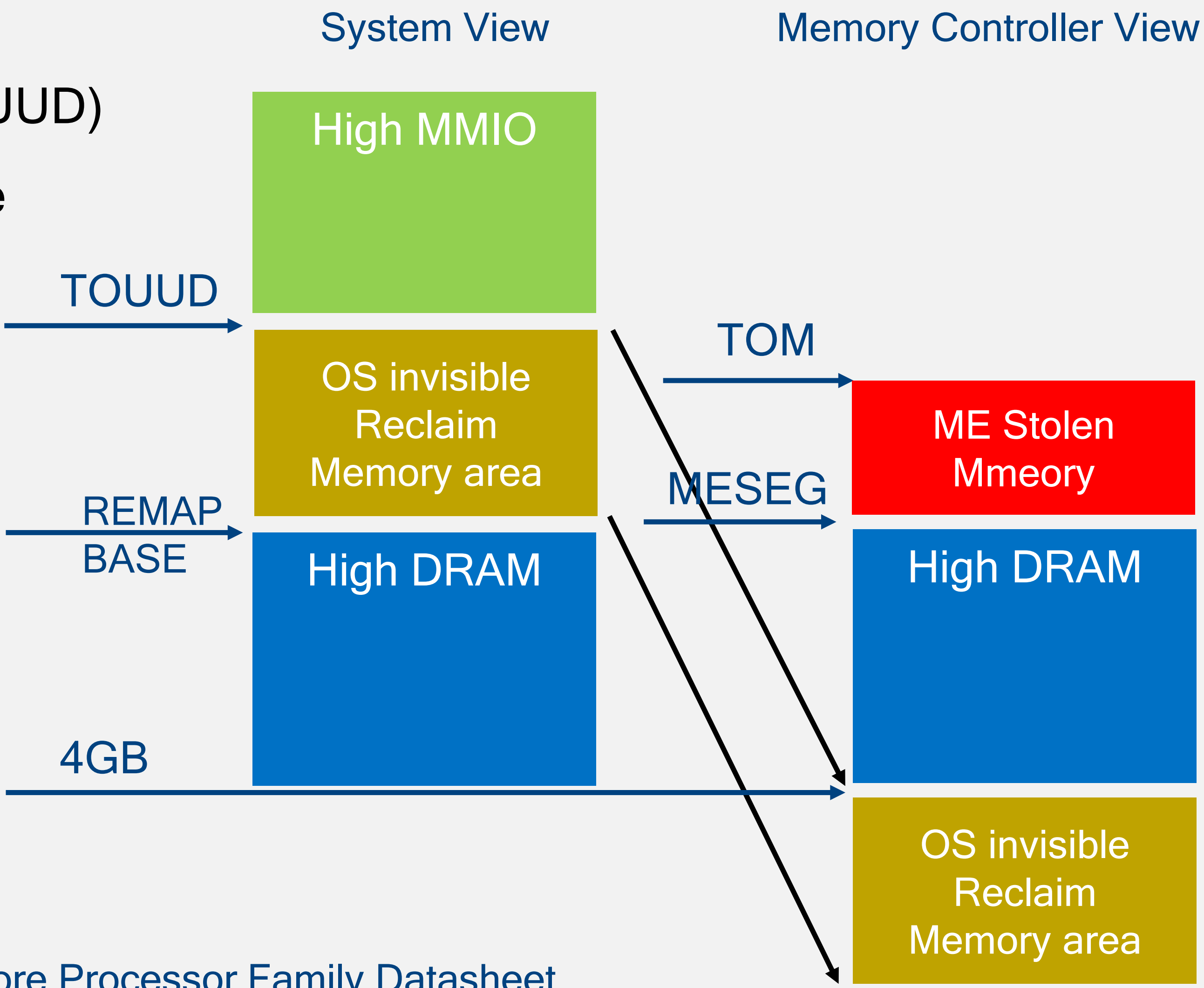
- Legacy Ranges (PAMx)
- ISA Hole
- SMRAM
- Graphics Stolen Memory
- Low DRAM (TOLUD)
- Memory vs MMIO
- PCIe MMCFG
- MMIO BARs
- Reserved Ranges (APIC, TPM, TXT)
- High BIOS



Reference: 4th Gen Intel Core Processor Family Datasheet

Memory Map (Above 4GB)

- Memory Reclaim
- High DRAM (TOUUD)
- ME Stolen Range
- Top Of Memory

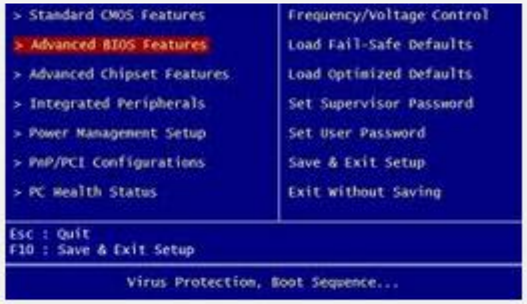


Reference: 4th Gen Intel Core Processor Family Datasheet

1.2 Platform Firmware: BIOS

```
                    [illegible] SOFTWARE, INC.
Virus Warning                : Disabled
CPU L1 Cache                 : Enabled
CPU L2 Cache                 : Enabled
CPU L2 Cache ECC Checking    : Enabled
Quick Power On Self Test     : Enabled
Boot Sequence                : D,A,SCSI
Swap Floppy Drive            : Disabled
Boot Up Floppy Seek          : Disabled
Boot Up NumLock Status       : On
Typematic Rate Setting       : Enabled
Typematic Rate (Chars/Sec)   : 30
Typematic Delay (Msec)       : 250
Security Option              : Setup
PCI/VGA Palette Snoop        : Enabled
OS Select For DRAM > 64MB    : Non-OS2
HDD S.M.A.R.T. Capability    : Enabled

                    ESC : Quit        ^v>< : Select Item
                    F1  : Help        PU/PD/+/- : Modify
                    F5  : Old Values  (Shift)F2 : Color
                    F6  : Load Fail-Safe Settings
                    F7  : Load Optimal Settings
```

```
  DRAM Timing Selectable      [By SPD]            Item Help
x CAS Latency Time             2.5
x Active to Precharge Delay    6                   Menu Level  >
x DRAM RAS# to CAS# Delay      3
x DRAM RAS# Precharge          3
  Memory Frequency For        [Auto]
  System BIOS Cacheable       [Enabled]
  Video  BIOS Cacheable       [Disabled]
  Memory Hole At 15M-16M      [Disabled]
  AGP Aperture Size (MB)      [128]
  Init Display First          [PCI Slot]

  ** On-Chip VGA Setting **
  On-Chip VGA                 [Enabled]
  On-Chip Frame Buffer Size   [ 8MB]
  OnBoard LAN Control         [Enabled]

↑↓→←:Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
  F5: Previous Values    F6: Fail-Safe Defaults    F7: Optimized Defaults
```



```
> Standard CMOS Features          Frequency/Voltage Control
> Advanced BIOS Features          Load Fail-Safe Defaults
> Advanced Chipset Features       Load Optimized Defaults
> Integrated Peripherals          Set Supervisor Password
> Power Management Setup          Set User Password
> PnP/PCI Configurations          Save & Exit Setup
> PC Health Status                Exit Without Saving

Esc : Quit
F10 : Save & Exit Setup

            Virus Protection, Boot Sequence...
```

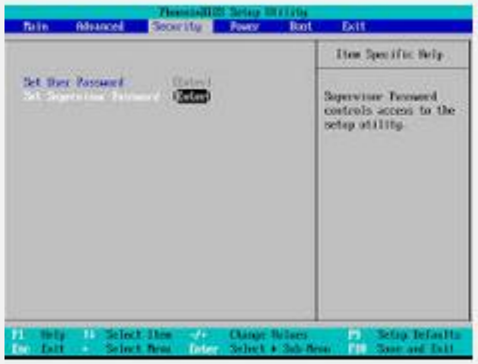
```
▶ Hard Disk B
  Virus Warni
  CPU L3 Cach
  Quick Power
  First Boot
  Second Boot
  Third Boot
  Boot Other
  Boot Up Flo
  Boot Up Num
  Gate A20 Op
  Typematic R
x Typematic R
x Typematic D
  Security Op
  APIC Mode
  MPS Version
  OS Select F
  Report No F

↑↓→← :Move  Ent
  F5: Previou
```

```
                 PhoenixBIOS Setup Utility
 Main   Advanced   Security   Boot   Exit

   System Time:                [15:32:57]         Item Specific Help
   System Date:                [10/08/2013]
                                                  <Tab>, <Shift-Tab>, or
   Legacy Diskette A:          [1.44/1.25 MB  3½"] <Enter> selects field.
   Legacy Diskette B:          [Disabled]

 ▶ Primary Master              [None]
 ▶ Primary Slave               [None]
 ▶ Secondary Master            [VMware Virtual IDE]
 ▶ Secondary Slave             [None]

 ▶ Keyboard Features

   System Memory:              640 KB
   Extended Memory:            2096128 KB
   Boot-time Diagnostic Screen: [Disabled]

F1   Help  ↑↓ Select Item  -/+   Change Values         F9   Setup Defaults
Esc  Exit  ↔  Select Menu  Enter Select ▶ Sub-Menu     F10  Save and Exit
```

```
         Phoenix - Award WorkstationBIOS CMOS Setup Utility
                    Advanced Chipset Features

 ▶ DRAM Clock/Timing Control   [Press Enter]        Item Help
 ▶ AGP & P2P Bridge Control    [Press Enter]
   Prefetch Caching            [Enabled]         Menu Level  ▶
   System BIOS Cacheable       [Enabled]
   VIDEO   RAM Cacheable       [Enabled]
   Memory Hole at 15M-16M      [Disabled]

Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
   F5: Previous Values   F6: FailSafe Defaults   F7: Optimized Defaults
```

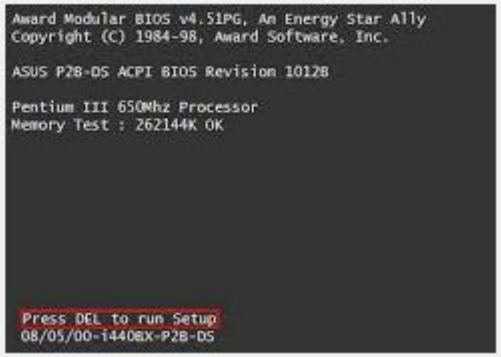
```
                       BIOS SETUP UTILITY
 Main   Advanced   Power   Boot   Exit

           Setup Warning                              NO: lets the BIOS
 Setting items on this menu to incorrect values      configure all the
 may cause your system to malfunction.               devices in the system.
                                                     YES: lets the
 CPU Type  Intel(R) Celeron(R) M processor           operating system
                                                     configure Plug and
 CPU Speed 1400MHz/400MHz                            Play (PnP) devices not
 Cache RAM 1024KB                                    required for boot if
                                                     your system has a Plug
 Plug and Play OS               [No]                 and Play operating
 Primary Video Adapter          [PCI]                system.
 Onboard Video Memory Size      [8MB/8MB/128MB/1]
 PS/2 Mouse                     [Auto Detect]        ↔  Select Screen
 Onboard PATA/SATA Adapters     [Both]               ↑↓ Select Item
 Onboard PATA/SATA Configuration [Enhanced Mode]     +- Change Option
 USB Legacy Mode Support        [Auto]               F1  General Help
 Onboard LAN                    [Enabled]            F5  Load Setup Defaults
 Onboard LAN Boot ROM           [Disabled]           F10 Save and Exit
 Onboard 1394                   [Enabled]            ESC Exit
 Onboard Audio                  [Auto]

        v02.58 (C)Copyright 1985-2005, American Megatrends, Inc.
```

```
                      Phoenix [illegible]
 Extreme Tweaker   Main  Advanced

   AI Tuning
 x Overclock Options
 x N.O.S. Option
   CPU Frequency
   CPU Multiplier
   K8<->NB HT Speed
   K8<->NB HT Width
   PCIE Clock
   CPU-NB Multiplier
   CPU-NB Voltage
   CPU Voltage

 ▶ DRAM Configuration

   Vcore Voltage
   VDDNB Voltage
   DDR2 Voltage
   SB Voltage
   HT Voltage
   VDDA Voltage
   NB Voltage
   DDR2 Controller Ref Voltage
   DDR2 Channel A Ref Voltage
   DDR2 Channel B Ref Voltage

 F1:Help    ↑↓:Select Item
 ESC:Exit   ↔:Select Menu
```



```
                 PhoenixBIOS Setup Utility
 Main   Advanced   Security   Power   Boot   Exit

                                                  Item Specific Help
   Set User Password           [Enter]
   Set Supervisor Password     [Enter]           Supervisor Password
                                                  controls access to the
                                                  setup utility.

F1   Help  ↑↓ Select Item  -/+   Change Values         F9   Setup Defaults
Esc  Exit  ↔  Select Menu  Enter Select ▶ Sub-Menu     F10  Save and Exit
```

```
                       AwardBIOS Setup Utility
            Advanced
          I/O Device Configuration                 Item Specific Help

   Onboard AC97 Modem Controller   [Disabled]      <Enter> to select.
   Onboard AC97 Audio Controller   [Auto]

   Onboard FDC Swap A & B          [No Swap]
   Floppy Disk Access Control    ┌──────────┐
                                 │ Disabled │
   Onboard Serial Port 1         │ Auto     │
   Onboard Serial Port 2         └──────────┘
   UART2 Use Standard Infrared

   Onboard Parallel Port           [378H/IRQ7]
   Parallel Port Mode              [ECP+EPP]
   ECP DMA Select                  [3]

   Onboard Game Port               [200H-207H]

F1   Help   ↑↓  Select Item   -/+    Change Values        F5   Setup Defaults
ESC  Exit   ↔   Select Menu   Enter  Select ▶ Sub-Menu    F10  Save and Exit
```



```
Award Modular BIOS v4.51PG, An Energy Star Ally
Copyright (C) 1984-98, Award Software, Inc.

ASUS P2B-DS ACPI BIOS Revision 1012B

Pentium III 650Mhz Processor
Memory Test : 262144K OK





 Press DEL to run Setup
 08/05/00-i440BX-P2B-DS
```

Legacy BIOS

- The Basic Input / Output System (BIOS) is the software embedded on a ROM chip (SPI flash memory devices) located on the computer's main board
- The BIOS is fetched by the processor at reset vector address (FFFFFFF0h) and executes Power-On Self Test (POST) to test and initialize the system components, initializes add-on devices and then boots the OS.
- The BIOS also handles the low-level input/output to the various peripheral devices connected to the computer

Plug and Play BIOS Specification

Legacy BIOS Stages

1. CPU Reset vector in BIOS 'ROM' (Boot Block) →

Boot Block

1. Basic CPU, chipset initialization →
2. Initialize CAR, load and run from cache →
3. Initialize DRAM memory →

POST (Power-On Self Test)

1. Decompress and relocate system BIOS in DRAM →
2. Enumerate add-on devices (ISA, PCI).. →
3. Execute Option ROMs on expansion cards →

INT 19h

1. Locate, load and execute Initial Boot Loader code in MBR (at 0x7C00 PA) →
2. 2nd Stage Boot Loader → OS Loader → OS kernel

Also Technical Note: UEFI BIOS vs. Legacy BIOS, Advantech

Legacy Option ROMs

- Legacy option ROMs are BIOS extensions required to initialize add-on devices (ISA, PCI, PCIe) not supported by the system BIOS nor required to boot the system
- Option ROMs are detected by the BIOS during after POST and their entry points are called
  - Scanned in physical memory between addresses C0000h and F0000h and detected by 0x55 0xAA signature
- During initialization, an Option ROM may hook any IVT vectors and update any data structures required for it to access attached devices and perform the necessary identifications and initializations

BIOS Boot Specification

1.3 Platform Firmware: (U)EFI Firmware

Industry Transition

Pre-2000	All Platforms BIOS were proprietary
2000	Intel invented the Extensible Firmware Interface (EFI) and provided sample implementation under free BSD terms
2004	**tianocore.org**, open source EFI community launched
2005	**Unified EFI (UEFI)** Industry forum, with 11 members, was formed to standardize EFI
2015	240 members Major MNCs shipping; UEFI platforms crossed most of IA worldwide units; Microsoft* UEFI x64 support in Server 2008, Vista* and Win7*; RedHat* and SuSEI* OS support. Mandatory for Windows 8 client. ARM 32 and 64 bit support. ACPI added.

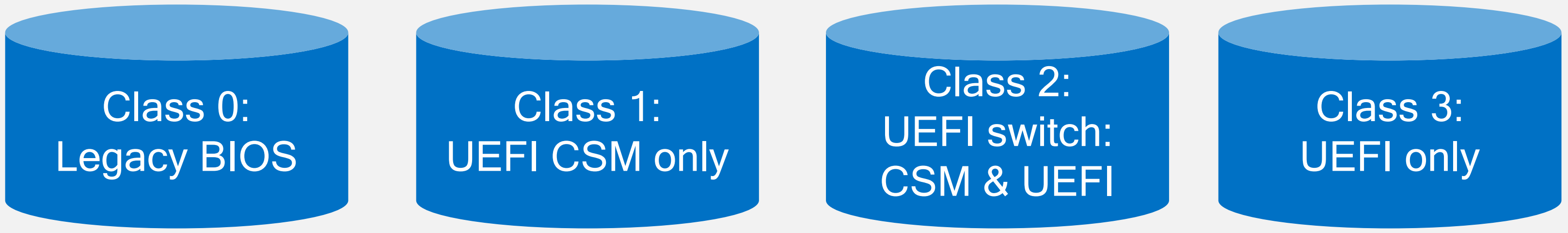
Source: http://www.slideshare.net/k33a/uefi

(Unified) Extensible Firmware Interface

- Industry Standard Interface Between Firmware & OS
- Processor Architecture and OS Independent
- C Development Environment (EDK2/UDK)
- Rich GUI Pre-Boot Application Environment
- Includes Modular Driver Model
- UEFI executables: PE/COFF or TE executable files
- UEFI file system is FAT32
- UEFI OS uses UEFI compliant bootloader:

  `/efi/boot/bootx64.efi /efi/redhat/grub.efi`
- Secure Boot of Microsoft Windows 8 or above requires UEFI
- UEFI firmware supports booting legacy OS from legacy MBR via Compatibility Support Module (CSM)

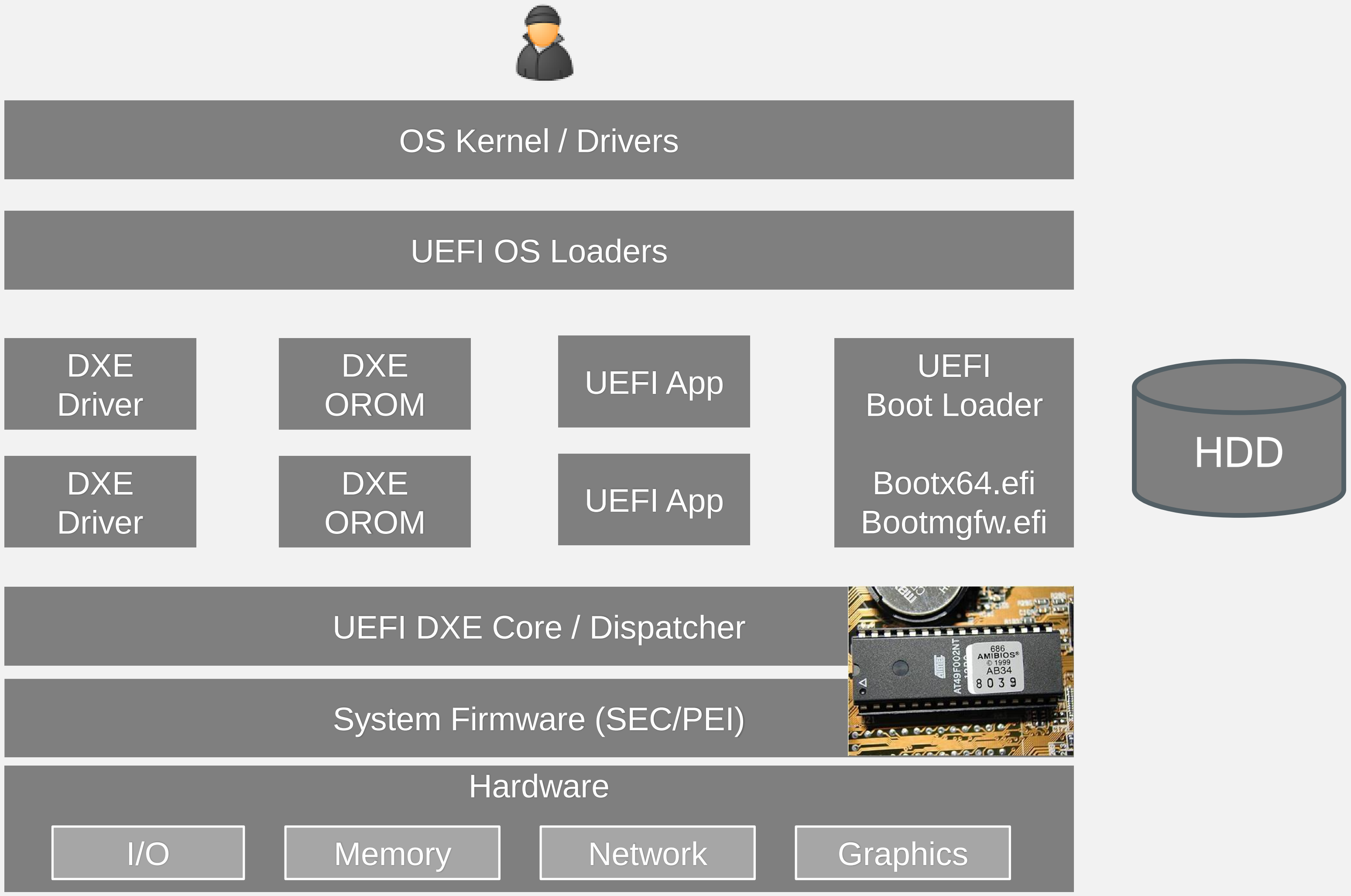
UEFI Firmware Evolution



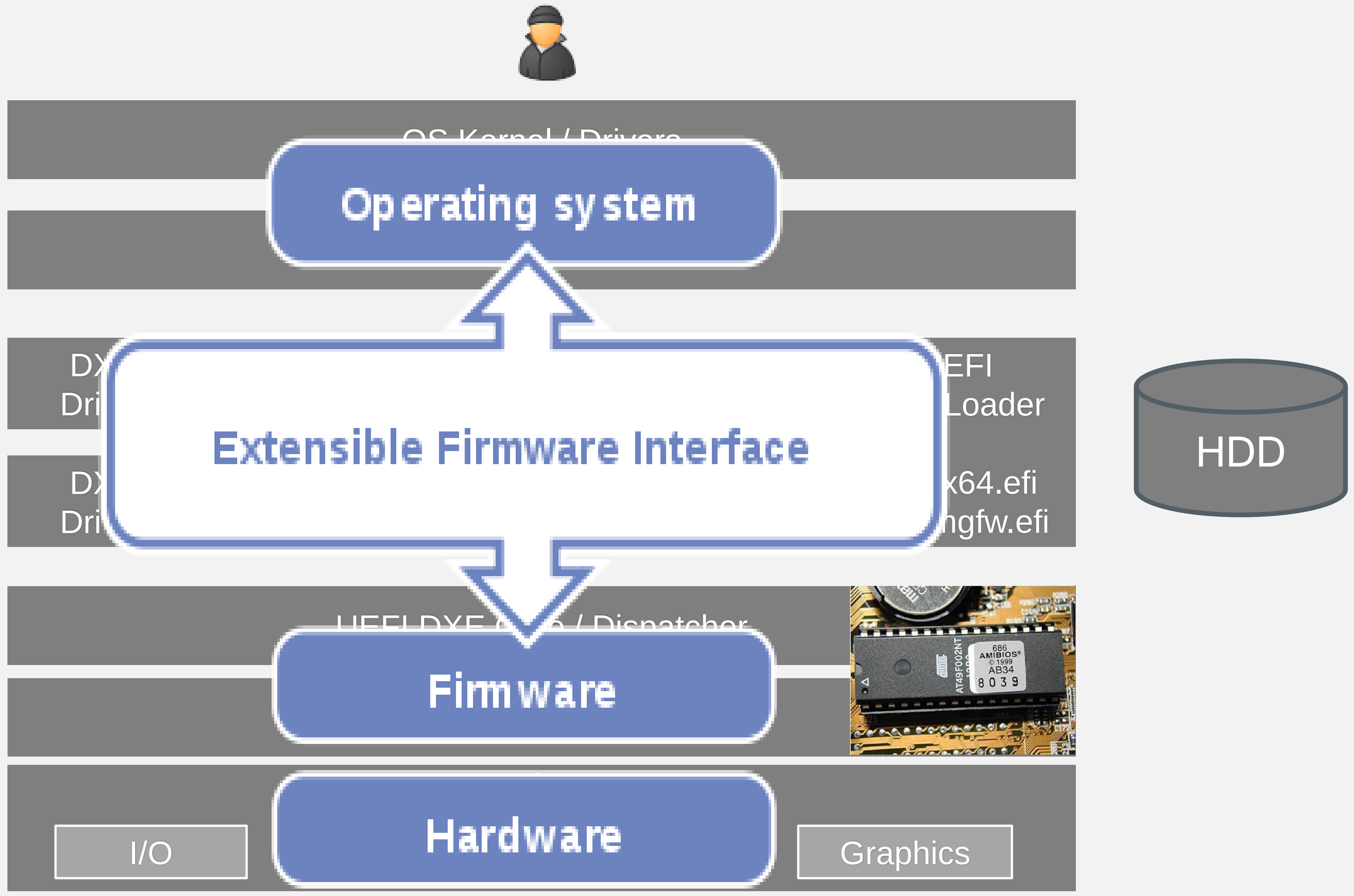
Compatibility Support Module (CSM)

- CSM is a UEFI component which allows legacy OS boot loader (MBR) and legacy Option ROM to execute on top of UEFI based firmware without modifications
- CSM has to emulate some legacy BIOS runtime services required for legacy boot
- CSM consists of the following components:
    - EfiCompatoibility (Legacy BIOS driver, drivers for legacy devices like PIC 8259)
    - Compatibility16BIOS (provides legacy BIOS runtime services as INT13, INT15, INT19 etc.)
    - Compatibility16SMM (provides legacy SMI handlers)
    - Thunk/ReserveThunk (switching between 32-bit and 16-bit modes)
    - Legacy Option ROM interface

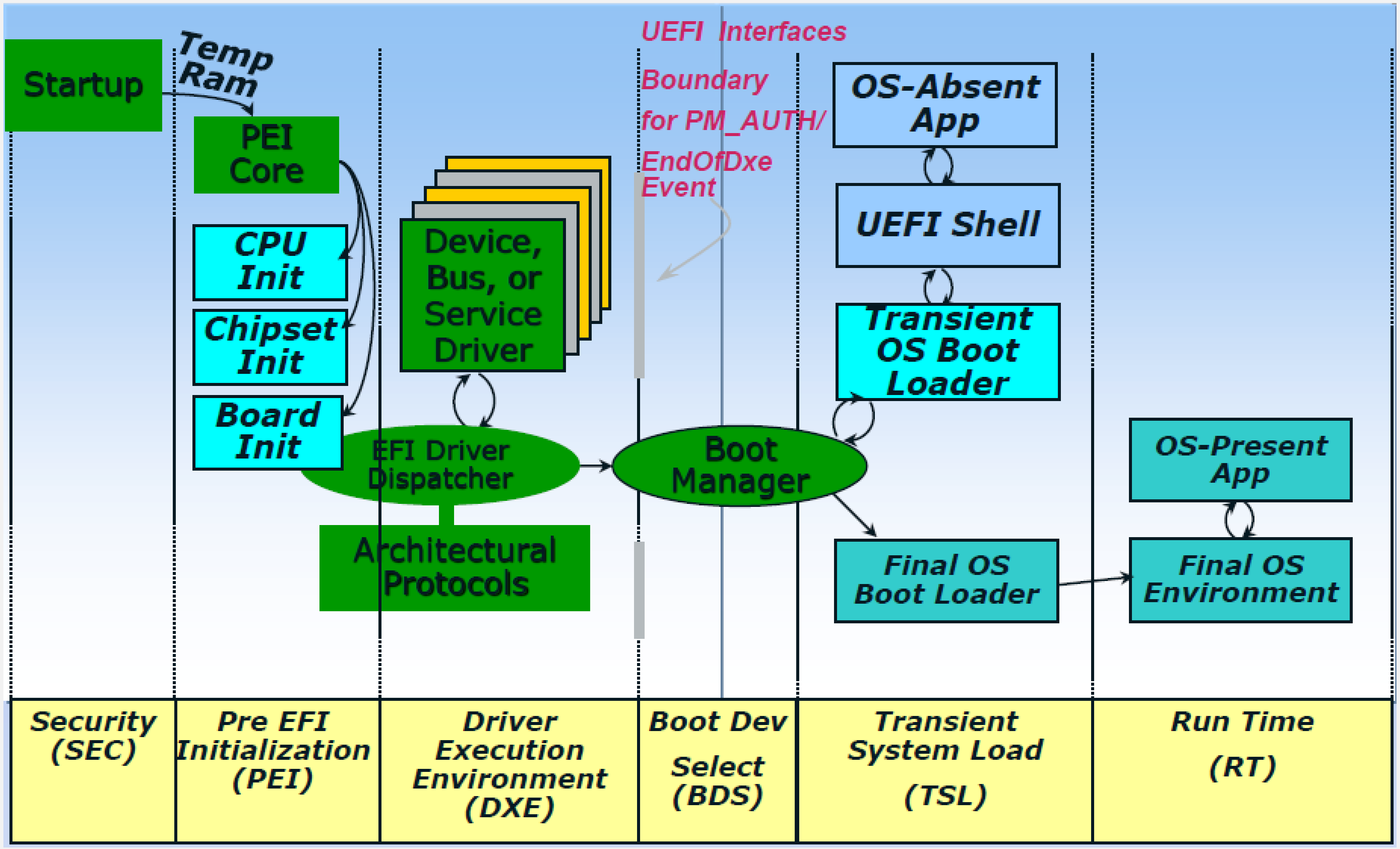
(U)EFI Firmware



(U)EFI Firmware

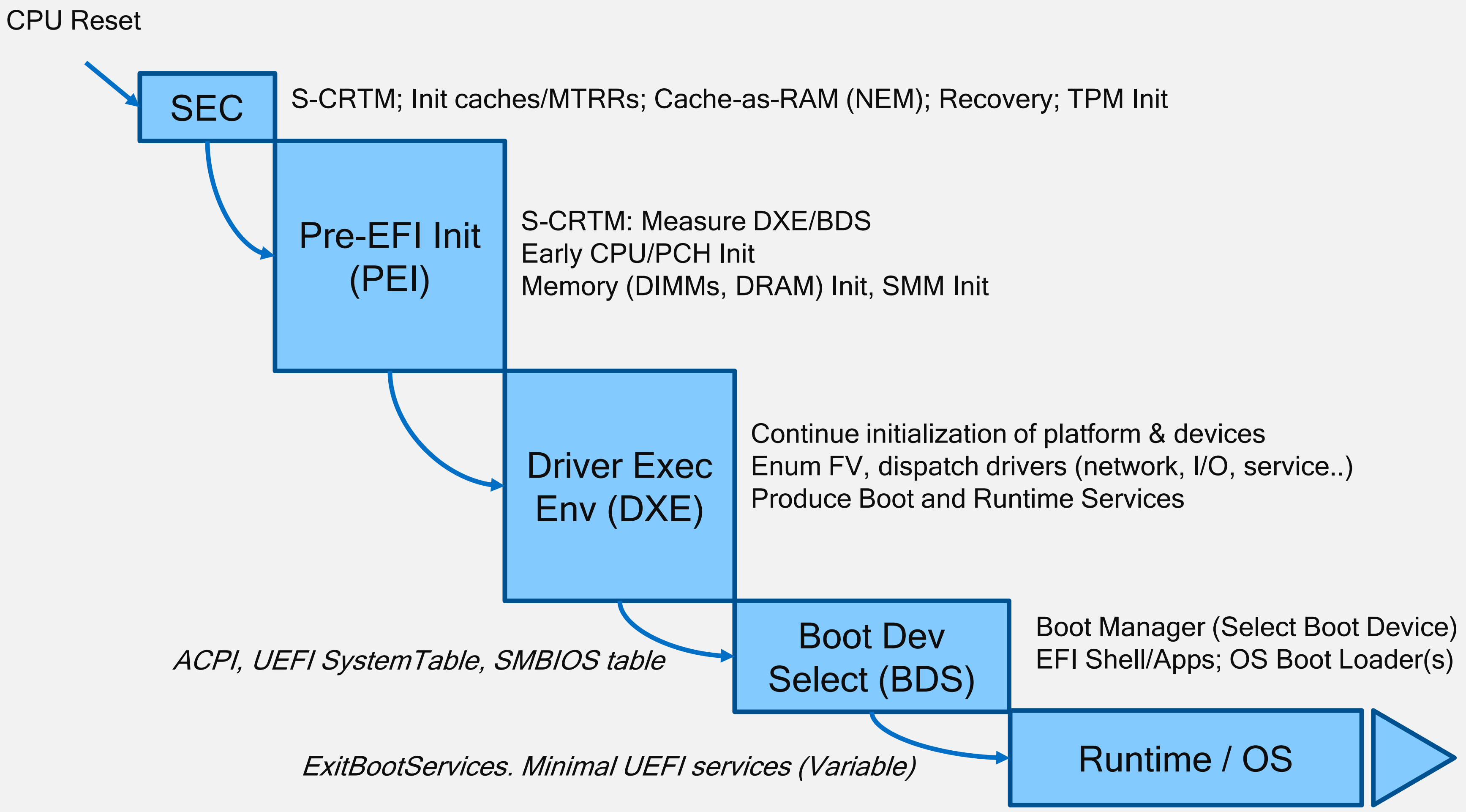


UEFI Boot



From Secure Boot, Network Boot, Verified Boot, oh my and almost every publication on UEFI

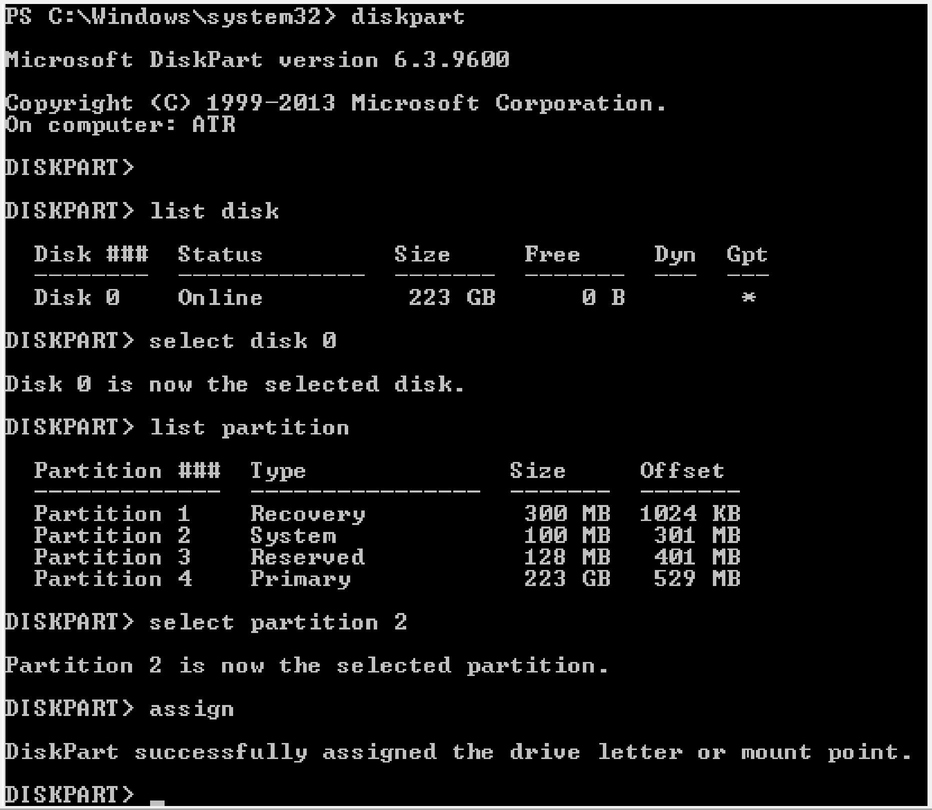
UEFI [Compliant] Firmware



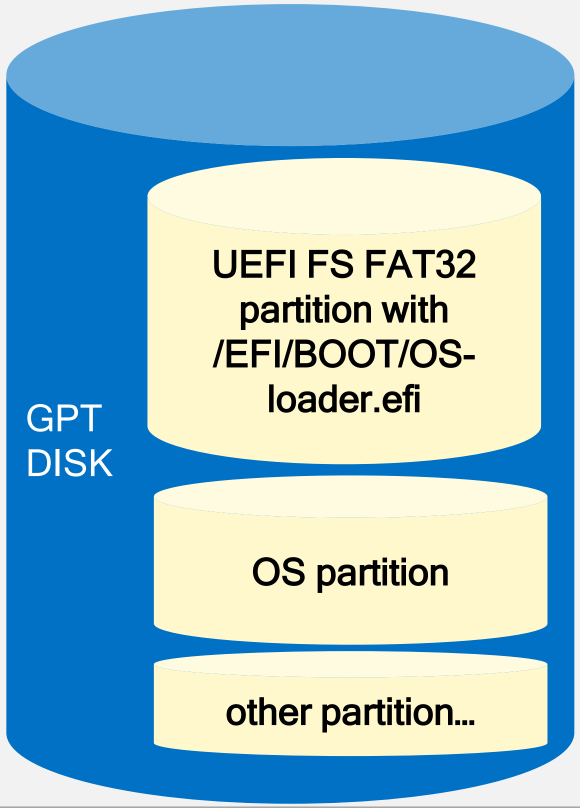
CPU Reset
SEC
S-CRTM; Init caches/MTRRs; Cache-as-RAM (NEM); Recovery; TPM Init
Pre-EFI Init (PEI)
S-CRTM: Measure DXE/BDS
Early CPU/PCH Init
Memory (DIMMs, DRAM) Init, SMM Init
Driver Exec Env (DXE)
Continue initialization of platform & devices
Enum FV, dispatch drivers (network, I/O, service..)
Produce Boot and Runtime Services
ACPI, UEFI SystemTable, SMBIOS table
Boot Dev Select (BDS)
Boot Manager (Select Boot Device)
EFI Shell/Apps; OS Boot Loader(s)
ExitBootServices. Minimal UEFI services (Variable)
Runtime / OS

UEFI OS Booting

1. Exit UEFI Boot Services
2. Looking into UEFI boot variable: BootXXXX for discover UEFI storage devices
3. UEFI transfer control to OS bootloader from storage device



```
PS C:\Windows\system32> diskpart

Microsoft DiskPart version 6.3.9600

Copyright (C) 1999-2013 Microsoft Corporation.
On computer: ATR

DISKPART>

DISKPART> list disk

  Disk ###  Status         Size     Free     Dyn  Gpt
  --------  -------------  -------  -------  ---  ---
  Disk 0    Online          223 GB      0 B        *

DISKPART> select disk 0

Disk 0 is now the selected disk.

DISKPART> list partition

  Partition ###  Type              Size     Offset
  -------------  ----------------  -------  -------
  Partition 1    Recovery           300 MB  1024 KB
  Partition 2    System             100 MB   301 MB
  Partition 3    Reserved           128 MB   401 MB
  Partition 4    Primary            223 GB   529 MB

DISKPART> select partition 2

Partition 2 is now the selected partition.

DISKPART> assign

DiskPart successfully assigned the drive letter or mount point.

DISKPART> _
```

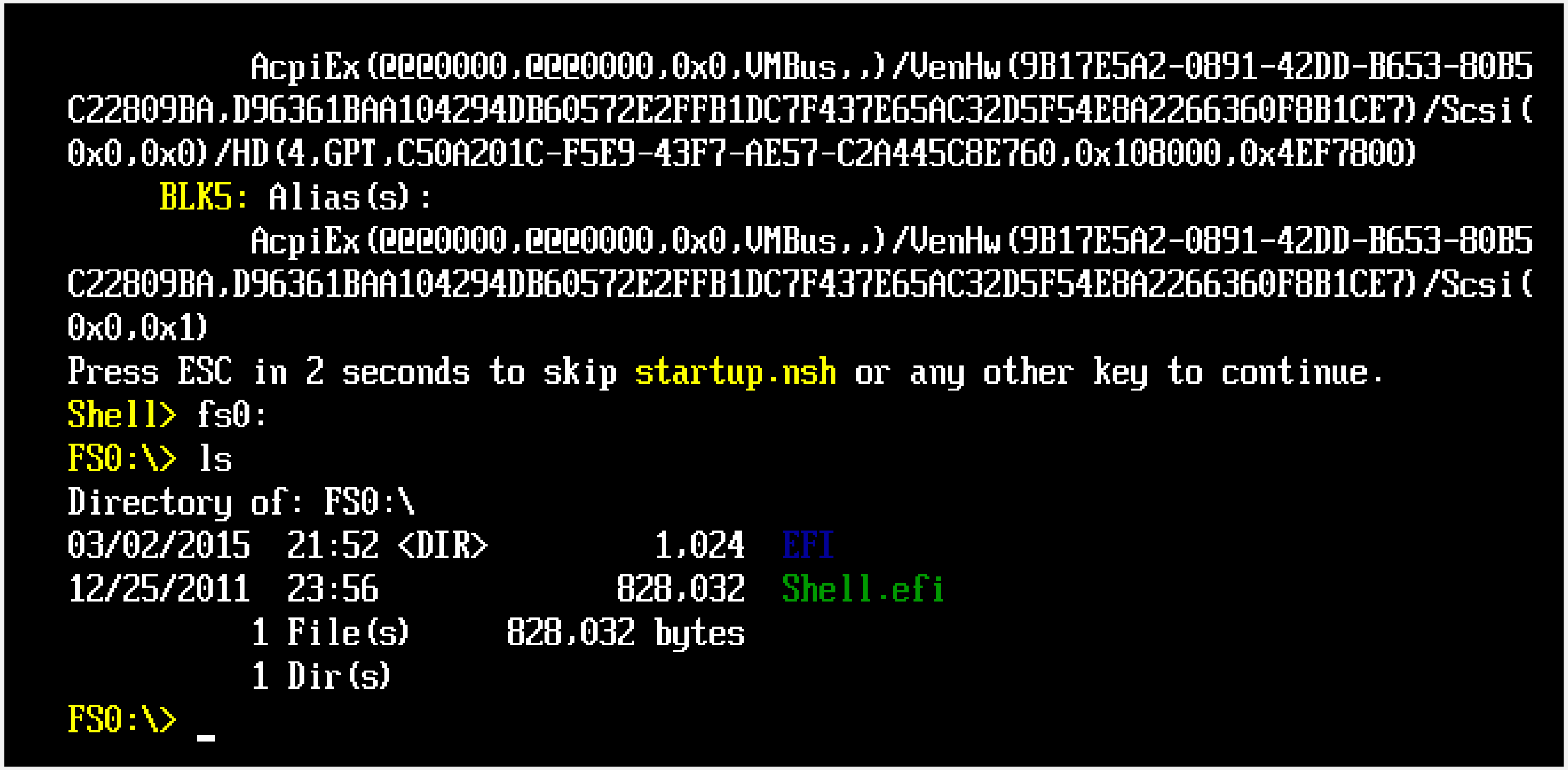


UEFI Shell

UEFI shell is a shell command line environment

Used to run shell commands and UEFI applications: dmpstore

UEFI OS boot loaders are also UEFI “applications”

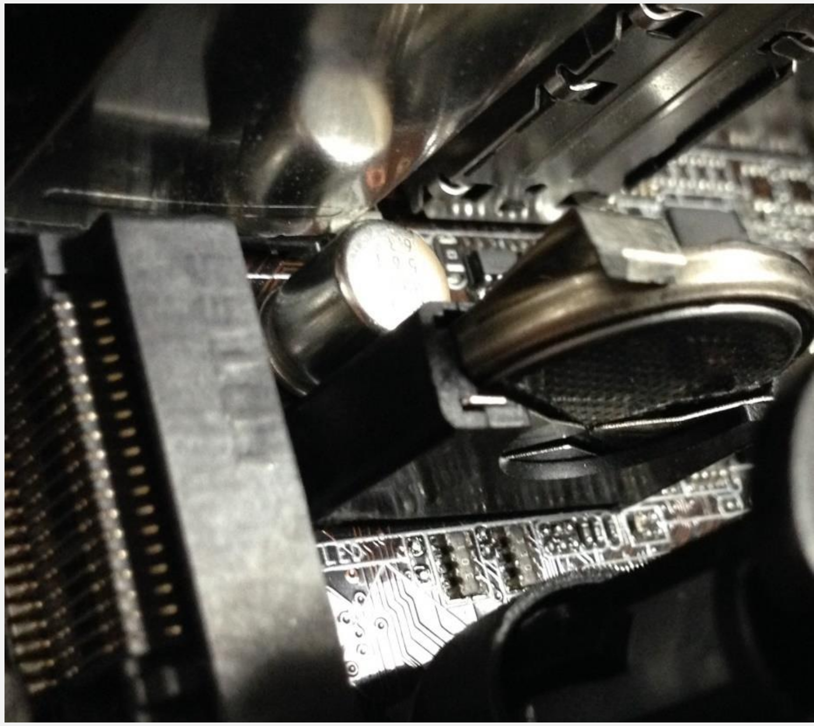


```
          AcpiEx(@@@0000,@@@0000,0x0,VMBus,,)/VenHw(9B17E5A2-0891-42DD-B653-80B5
C22809BA,D96361BAA104294DB60572E2FFB1DC7F437E65AC32D5F54E8A2266360F8B1CE7)/Scsi(
0x0,0x0)/HD(4,GPT,C50A201C-F5E9-43F7-AE57-C2A445C8E760,0x108000,0x4EF7800)
     BLK5: Alias(s):
          AcpiEx(@@@0000,@@@0000,0x0,VMBus,,)/VenHw(9B17E5A2-0891-42DD-B653-80B5
C22809BA,D96361BAA104294DB60572E2FFB1DC7F437E65AC32D5F54E8A2266360F8B1CE7)/Scsi(
0x0,0x1)
Press ESC in 2 seconds to skip startup.nsh or any other key to continue.
Shell> fs0:
FS0:\> ls
Directory of: FS0:\
03/02/2015  21:52 <DIR>         1,024  EFI
12/25/2011  23:56             828,032  Shell.efi
          1 File(s)      828,032 bytes
          1 Dir(s)
FS0:\> _
```

BIOS Configuration: CMOS Memory

- 256 bytes (low & high 128)
- Backed by a small battery. Can be cleared by removing the “coin” battery (or a jumper)
- Stores BIOS specific configuration settings

```
# chipsec_util.py cmos dump
```



UEFI Configuration: UEFI “Variables”

UEFI variables contain configuration setup, vendor information, language information, input/output console, error console, and boot order setting, secure boot configuration, long information after capsule update, pointer to S3 boot script and so on.

Attributes of UEFI variables:

- NV (Non-Volatile)
- BS (Boot Service)
- RT (Run-Time)
- Authentication attributes

NV variables are stored in NVRAM in SPI flash memory

Windows API to access UEFI variables

Windows 8+ provides user mode API to access run-time UEFI variables

- **`GetFirmwareEnvironmentVariable`**
  http://msdn.microsoft.com/en-us/library/windows/desktop/ms724325(v=vs.85).aspx

- **`SetFirmwareEnvironmentVariable`**
  http://msdn.microsoft.com/en-us/library/windows/desktop/ms724934(v=vs.85).aspx

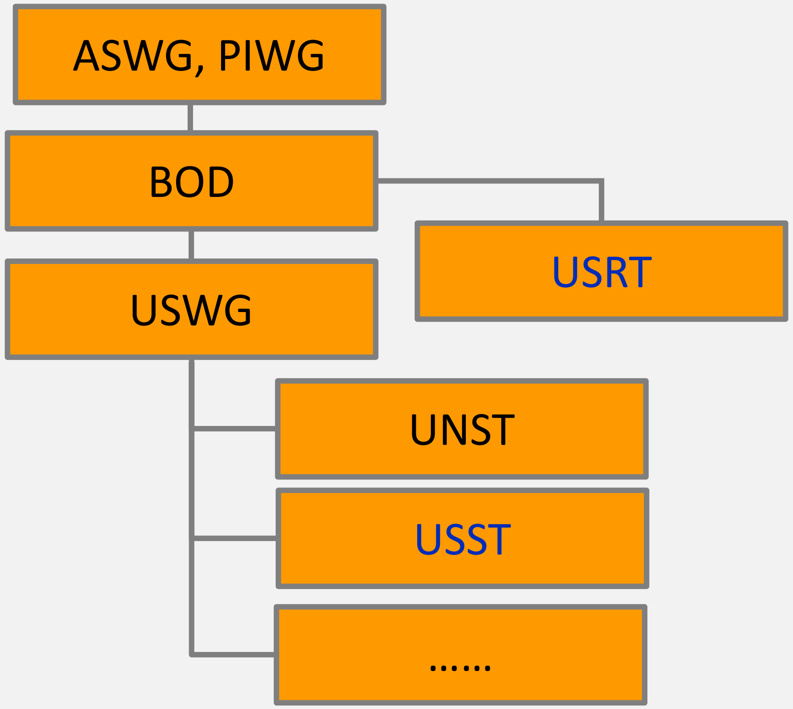
UEFI, EDK I, EDK II, UDK, Tianocore

- TianoCore is an open source implementation of **UEFI**, the **Unified Extensible Firmware Interface**
- **EDK II** *is a modern, feature-rich, cross-platform firmware development environment for the UEFI and PI specifications*: http://www.tianocore.org/edk2/
- **UDK** (UEFI Developers Kit) is a stable release of portions of the EDK II (UDK2015)

  The UDK2015 is the EDKII support for all currently published UEFI specifications UDK2015 currently supports UEFI 2.5 and PI 1.4 level of specifications
- **EDK I** is the older EFI 1.x development environment

UEFI Working Groups

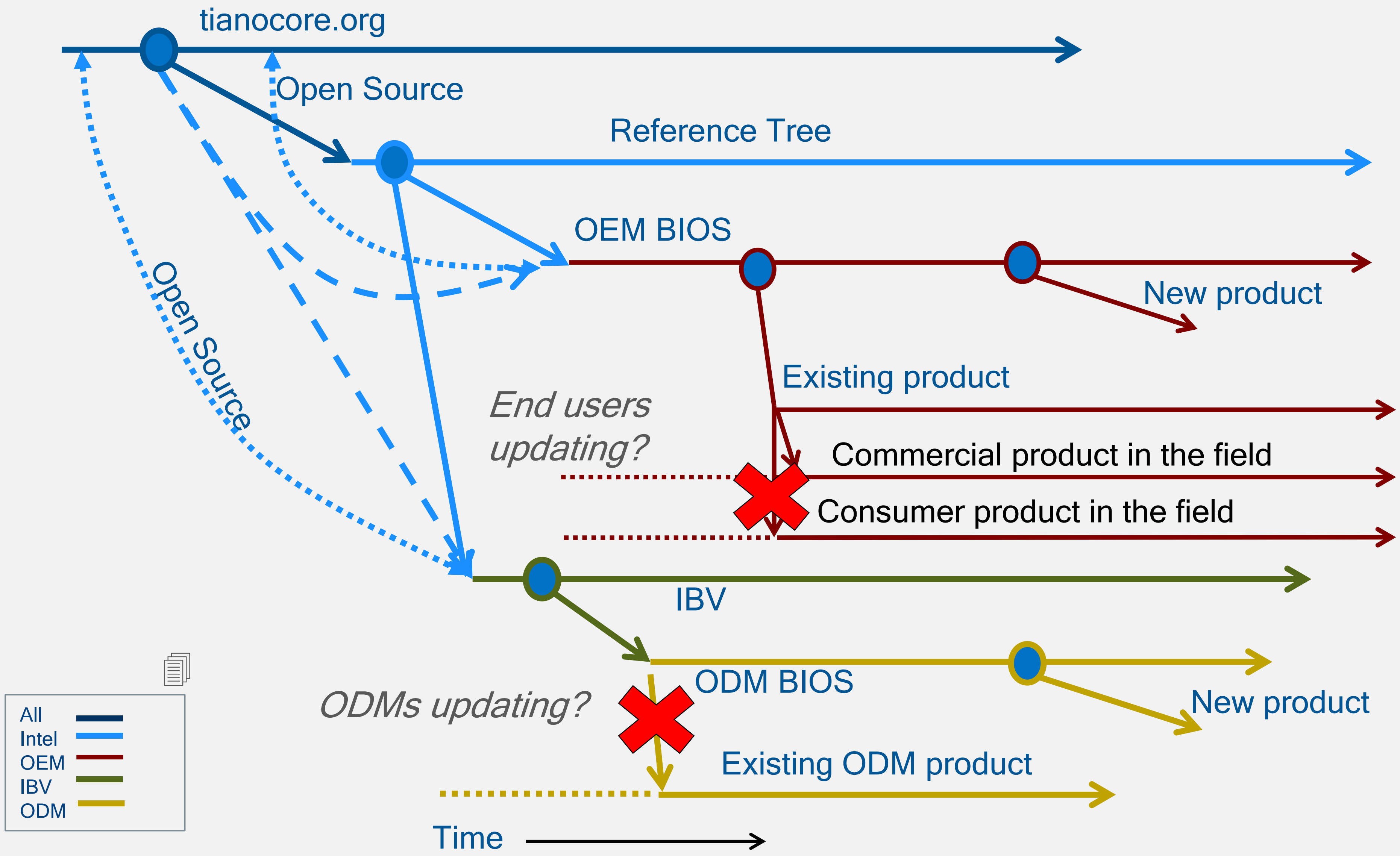
www.uefi.org



Note: Engaged in firmware/boot
Related WG's of Trusted Computing Group (TCG), IETF, DMTF

- **USWG**
  - **U**EFI **S**pecification **W**orking **G**roup
- **PIWG**
  - **P**latform **I**nitialization **W**orking **G**roup
- **ASWG**
  - **A**CPI **S**pecification **W**orking **G**roup
- **BOS**
  - **B**oard **O**f **D**irectors
- **USST**
  - **U**SWG **S**ecurity **S**ub-**t**eam
  - Chaired by Vincent Zimmer (Intel)
  - Responsible for all security related material and the team that has added security infrastructure in the UEFI spec
- **USRT**
  - UEFI Security Response Team
  - Chaired by Dick Wilkins (Phoenix)
  - Provide response to security issues.
- **UNST**
  - **U**EFI **N**etwork **S**ub-**t**eam (VZ chairs, too)
  - Evolve network boot & network security infrastructure for UEFI Specification

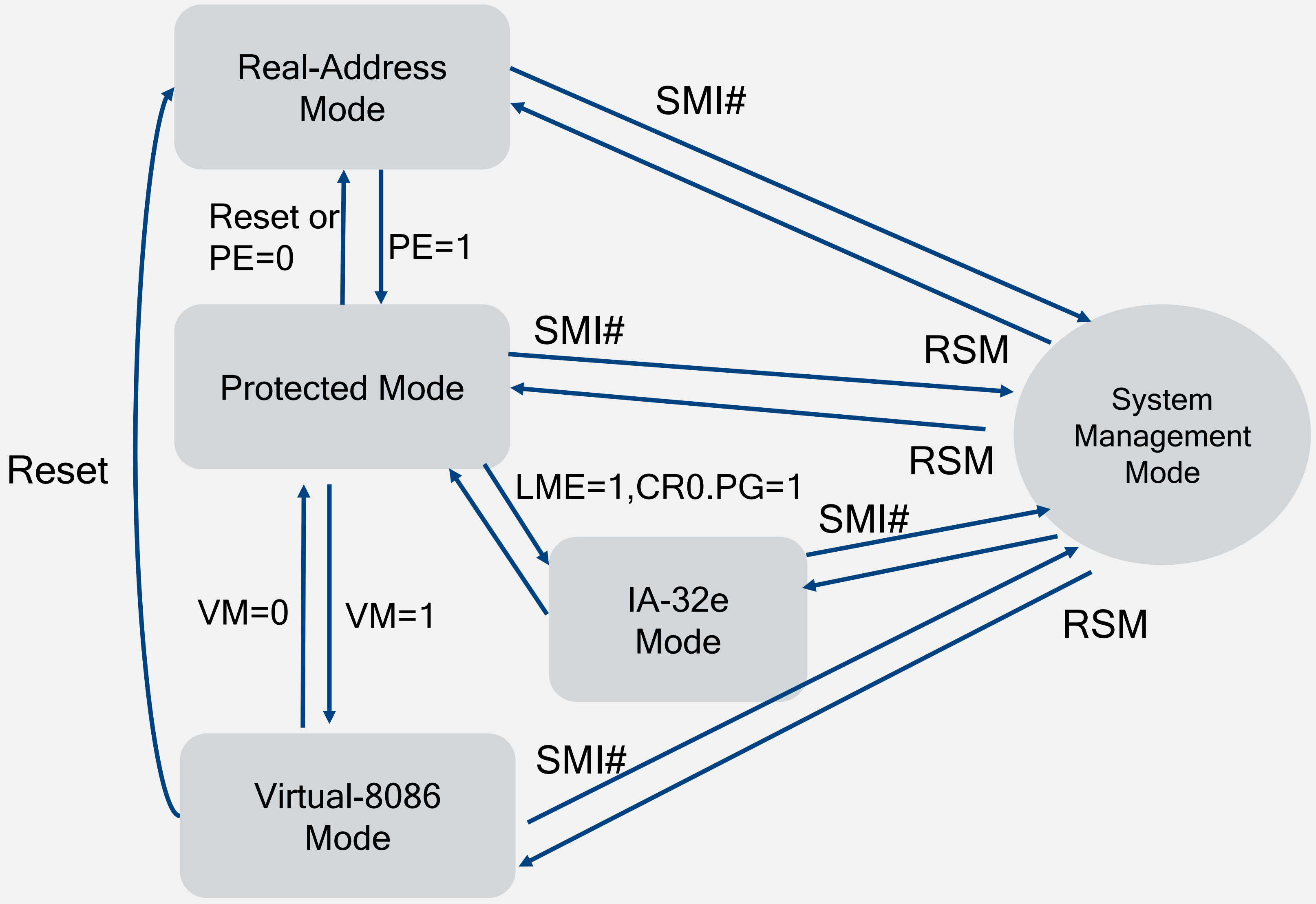
The road from core to platform



Source: http://vzimmer.blogspot.com/2015_05_01_archive.html

1.4 Platform Firmware: SMI Handlers

x86 System Management Mode (SMM)



Source: Intel® 64 and IA-32 Architectures Software Developer's Manual

System Management Mode (SMM)

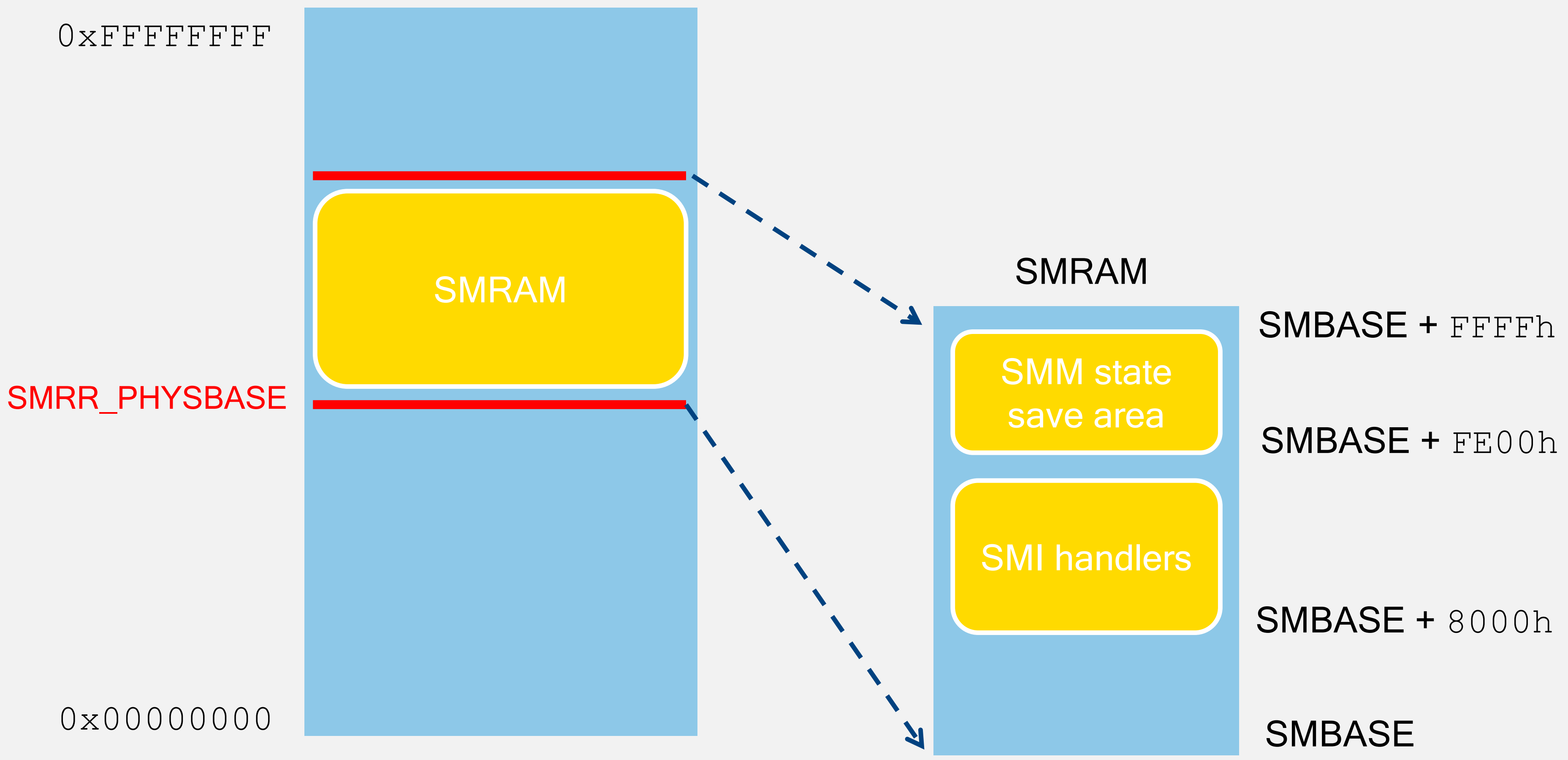
- SMRAM is a range of DRAM reserved by BIOS for runtime part - SMI handlers
- CPU enters System Management Mode (SMM) upon receiving System Management Interrupt (SMI#) from the chipset or other logical CPU
- SMI handler firmware is executing in SMM
- CPU (OS) state is saved in SMRAM upon entry to SMM and restored upon exit from SMM
- CPU exits SMM to the interrupted OS when SMI handler executes `RSM` instruction (“Resume from SMM”)

Initial SMM Execution Environment

- Separate address space (SMRAM)
- SMM starts as Read-Address Mode with flat 32-bit addressable space without paging (CR0 PE/PG = 0)
  - SMI handler firmware can enable paging later
  - Segments: base = `0`, limit = `FFFFFFFFh`
  - Addressable physical memory: `0` to `FFFFFFFFh` (4G)
- CS.base = SMBASE (`30000h` at reset), EIP=`8000h`

  ➔ SMM entry point = SMBASE + `8000h`
- All hardware interrupts are disabled upon entry

System Management RAM (SMRAM)



x64 SMM Save State

Offset (Added to SMBASE + 8000H)	Register	Writable?
7FF8H	CR0	No
7FF0H	CR3	No
7FE8H	RFLAGS	Yes
7FE0H	IA32_EFER	Yes
7FD8H	RIP	Yes
7FD0H	DR6	No
7FC8H	DR7	No
7FC4H	TR SEL1	No
7FC0H	LDTR SEL1	No
7FBCH	GS SEL1	No
7FB8H	FS SEL1	No
7FB4H	DS SEL1	No
7FB0H	SS SEL1	No
7FACH	CS SEL1	No
7FA8H	ES SEL1	No
7FA4H	IO_MISC	No
7F9CH	IO_MEM_ADDR	No
7F94H	RDI	Yes

Reference: Intel® 64 and IA-32 Architectures Software Developer's Manual

x64 SMM Save State (cont'd)

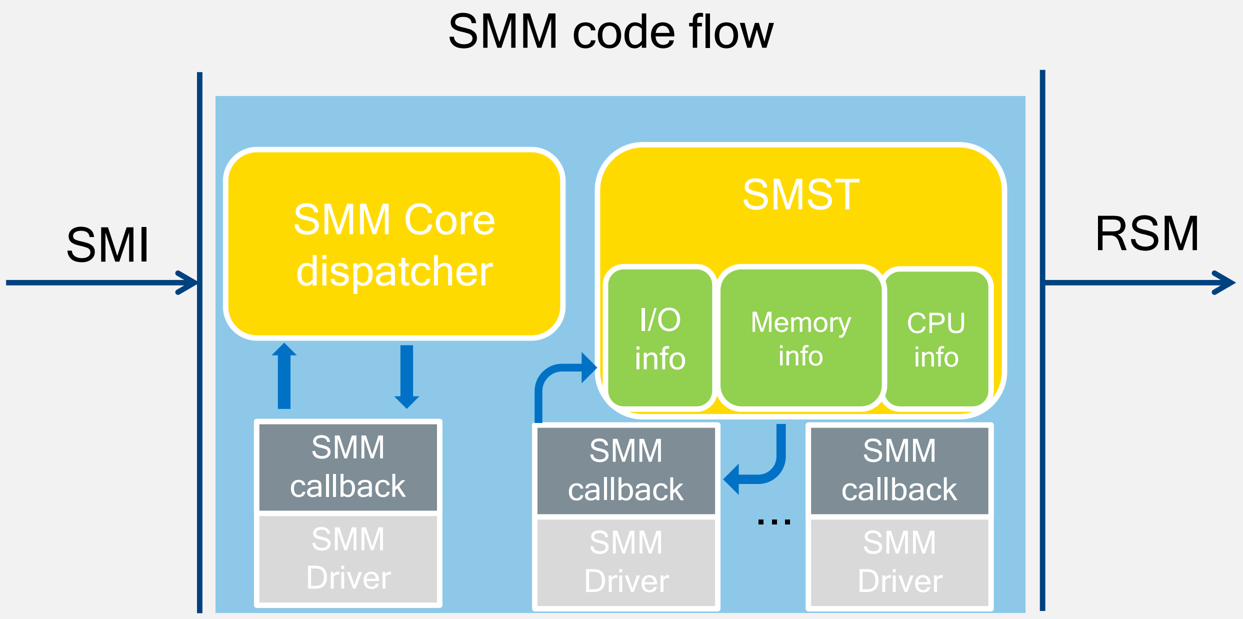
Offset (Added to SMBASE + 8000H)	Register	Writable?
7F8CH	RSI	Yes
7F84H	RBP	Yes
7F7CH	RSP	Yes
7F74H	RBX	Yes
7F6CH	RDX	Yes
7F64H	RCX	Yes
7F5CH	RAX	Yes
7F54H	R8	Yes
7F4CH	R9	Yes
7F44H	R10	Yes
7F3CH	R11	Yes
7F34H	R12	Yes
7F2CH	R13	Yes
7F24H	R14	Yes
7F1CH	R15	Yes
7F1BH-7F04H	Reserved	No
7F02H	Auto HALT Restart Field (Word)	Yes
7F00H	I/O Instruction Restart Field (Word)	Yes
7EFCH	SMM Revision Identifier Field (Doubleword)	No
7EF8H	SMBASE Field (Doubleword)	Yes

Reference: Intel® 64 and IA-32 Architectures Software Developer's Manual

Locating SMRAM using CHIPSEC

```
python chipsec_util.py msr 0x1f2
..
[*] loading common platform config from `..chipsec\tool\chipsec/cfg\common.xml'..
[*] loading 'hsw' platform config from `..chipsec\tool\chipsec/cfg\hsw.xml'..
[CHIPSEC] Executing command 'msr' with args ['0x1f2']
[CHIPSEC] CPU0: RDMSR( 0x1f2 ) = 00000000DA000006 (EAX=DA000006, EDX=00000000)
[CHIPSEC] CPU1: RDMSR( 0x1f2 ) = 00000000DA000006 (EAX=DA000006, EDX=00000000)
[CHIPSEC] CPU2: RDMSR( 0x1f2 ) = 00000000DA000006 (EAX=DA000006, EDX=00000000)
[CHIPSEC] CPU3: RDMSR( 0x1f2 ) = 00000000DA000006 (EAX=DA000006, EDX=00000000)


python chipsec_util.py msr 0x1f3
..
[*] loading common platform config from `..chipsec\tool\chipsec/cfg\common.xml'..
[*] loading 'hsw' platform config from `..chipsec\tool\chipsec/cfg\hsw.xml'..
[CHIPSEC] Executing command 'msr' with args ['0x1f3']
[CHIPSEC] CPU0: RDMSR( 0x1f3 ) = 00000000FF000800 (EAX=FF000800, EDX=00000000)
[CHIPSEC] CPU1: RDMSR( 0x1f3 ) = 00000000FF000800 (EAX=FF000800, EDX=00000000)
[CHIPSEC] CPU2: RDMSR( 0x1f3 ) = 00000000FF000800 (EAX=FF000800, EDX=00000000)
[CHIPSEC] CPU3: RDMSR( 0x1f3 ) = 00000000FF000800 (EAX=FF000800, EDX=00000000)
```

System Management Interrupt (SMI) Handler



1.4 Platform Firmware: Hardware Reference Code, etc.

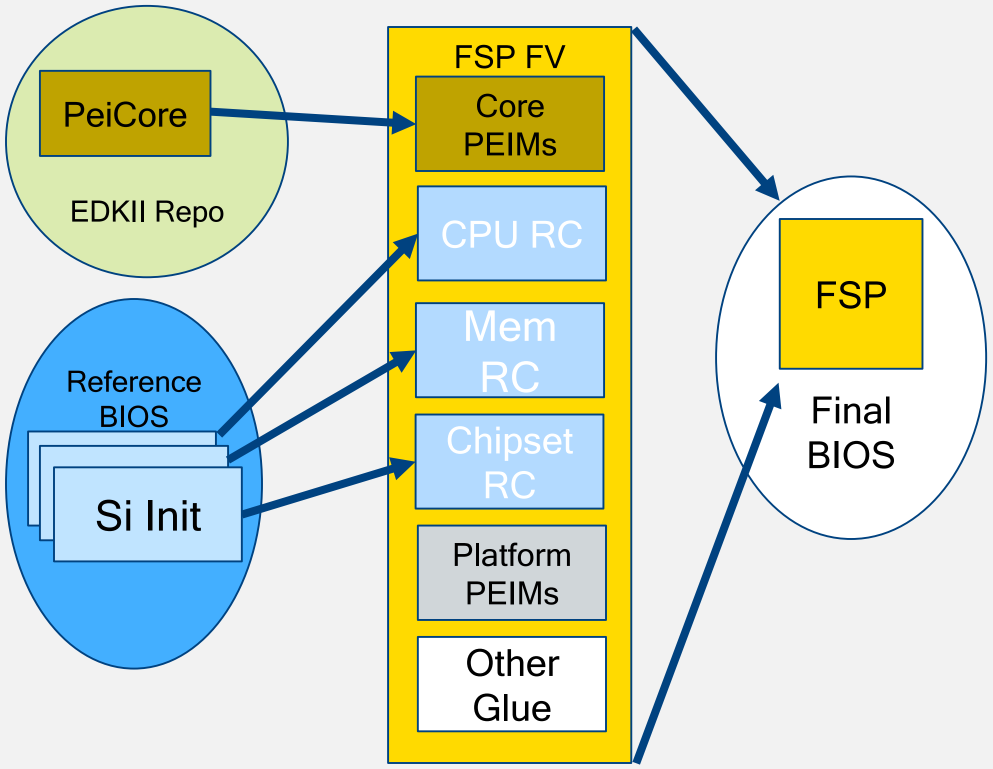
Hardware Reference Code

- Reference Code is a part of BIOS / system firmware which initializes specific piece of platform
  - CPU reference code (microcode update, CAR/NEM init)
  - Memory reference code trains DDR, initializes DIMMs, initializes memory controller, creates memory map
  - Chipset reference code initializes internal PCH interfaces
- Provided by CPU/chipset vendors to original equipment manufacturers (OEMs) or BIOS vendors (IBV) for integration into the final BIOS for specific platform

Intel® Firmware Support Package (FSP)

The Intel FSP provides processor, memory & chipset initialization in a format that can easily be incorporated into existing boot loader (firmware) frameworks

- Silicon initialization binaries (CPU, mem, PCH/SoC) packaged into FSP
- Plugs into existing FW frameworks
- Binary customization



Reference: www.intel.com/fsp

References and Further Reading

For further details on platform and BIOS fundamentals take

Advanced x86: Introduction to BIOS & SMM

class by John Butterworth, Xeno Kovah and Corey Kallenberg at http://OpenSecurityTraining.info

References and Further Reading

1. Advanced x86: Introduction to BIOS & SMM by John Butterworth and Xeno Kovah at OpenSecurityTraining.info
2. 4th Generation Intel® Core™ Processor Family Datasheet (vol 1, vol 2)
3. PCI Express System Architecture by Budruk, Anderson and Shanley (MindShare)
4. PCI System Architecture by Shanley and Anderson (MindShare)
5. An Introduction to PCI Express by Budruk (MindShare)
6. PCI/PCI Express Configuration Space Access (AMD)
7. PCI Express Base Specification Revision 3.0 (https://pcisig.com/specifications)
8. LPC specification
9. SPI Block Guide (Motorola, Inc.)
10. SMBus specification
11. Designing with SMBus 2.0 by Dale Stolitzka (Analog Devices, Inc.)
12. Minimal Intel Architecture Boot Loader by Jenny M Pelner & James A Pelner (Intel)
13. Pentium Processor Family Developer's Manual (Initialization and Mode Switching)
14. Power Sequence and Reset Utilizing the Intel® EP80579 Processor by Julio Pineda (Intel)
15. Plug and Play BIOS Specification
16. BIOS Boot Specification
17. Booting an Intel Architecture System by Pete Dice (Intel)
18. Intel® 64 and IA-32 Architectures Software Developer's Manual
19. UEFI Platform Initialization Specification 1.5
20. UEFI Specification 2.6
21. A Tour Beyond BIOS: Using Intel FSP with EDKII

Training materials are available on Github

https://github.com/advanced-threat-research/firmware-security-training

Yuriy Bulygin @c7zero
Alex Bazhaniuk @ABazhaniuk
Andrew Furtak @a_furtak
John Loucaides @JohnLoucaides