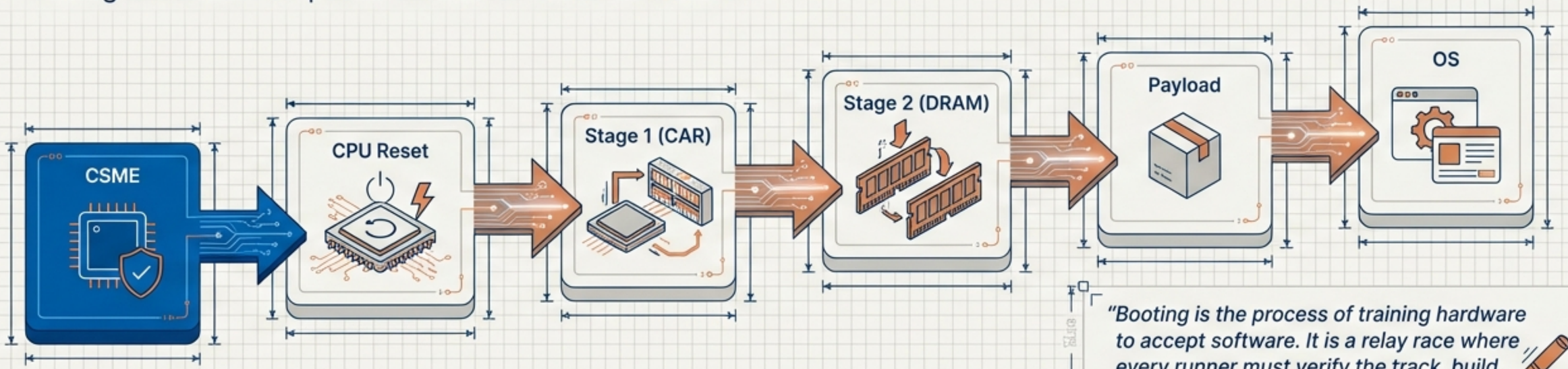
From Reset to Root

A Technical Deep Dive into the Intel x86 Boot Flow

Tracing the execution path from Power-On to OS Handoff



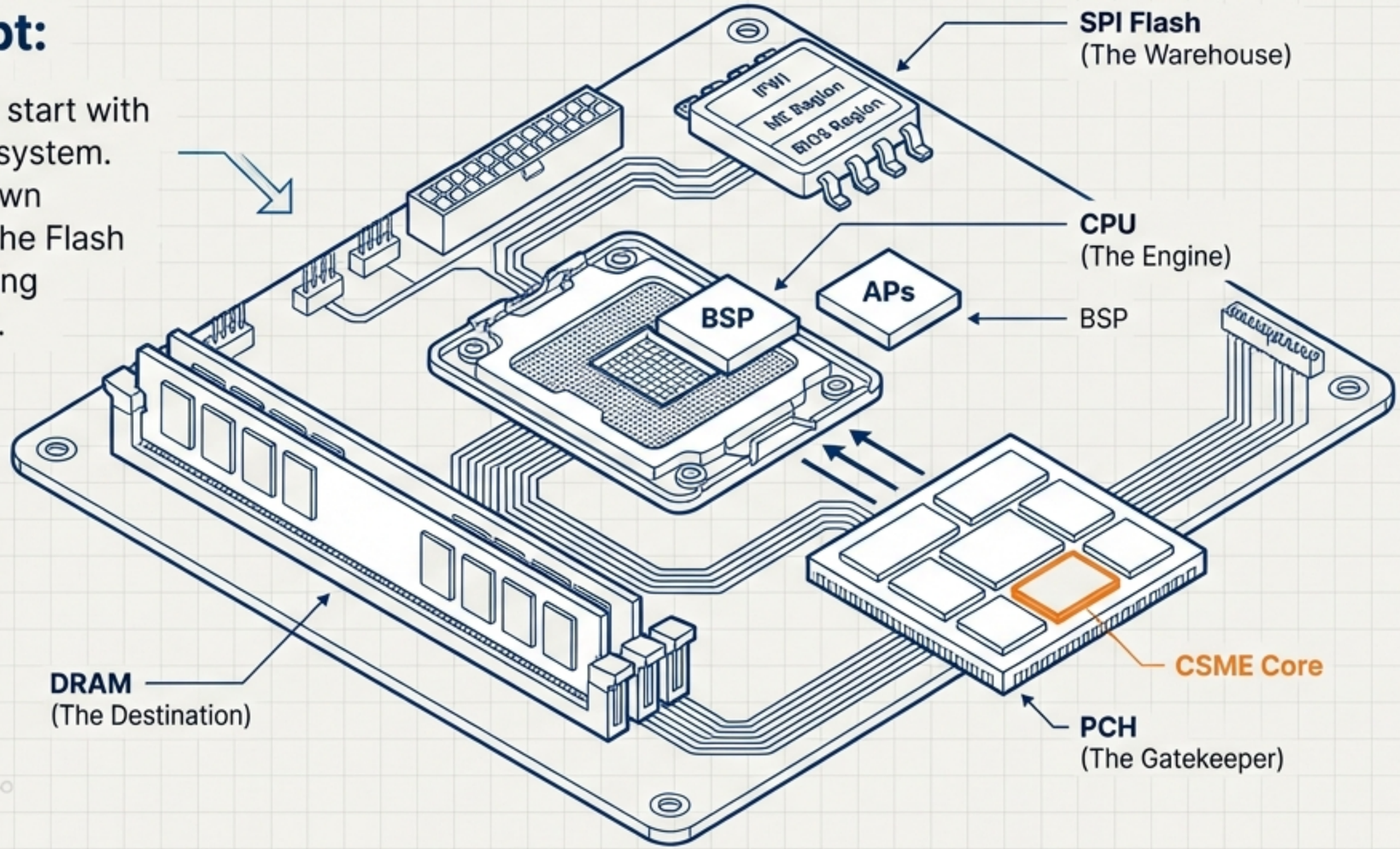
"Booting is the process of training hardware to accept software. It is a relay race where every runner must verify the track, build infrastructure, and pass the baton."

The Architecture of the Boot

Precision Engineering Schematic meets Swiss Editorial.

The Concept:

The CPU does not start with knowledge of the system. It must 'fetch' its own instructions from the Flash via memory mapping before RAM exists.



CSME:

Converged Security Management Engine (i486 core)

BSP:

Bootstrap Processor (Main execution core)

FSP:

Firmware Support Package (Silicon Init Binaries)

Blue/White: Wireframe & Solid Components
Orange: Highlighted Core/Package
Text: Helvetica Now Display / Inter / Jet8rains Meno

Execution: Suspended | **Memory:** SRAM Only | **Security:** Hardware Fused

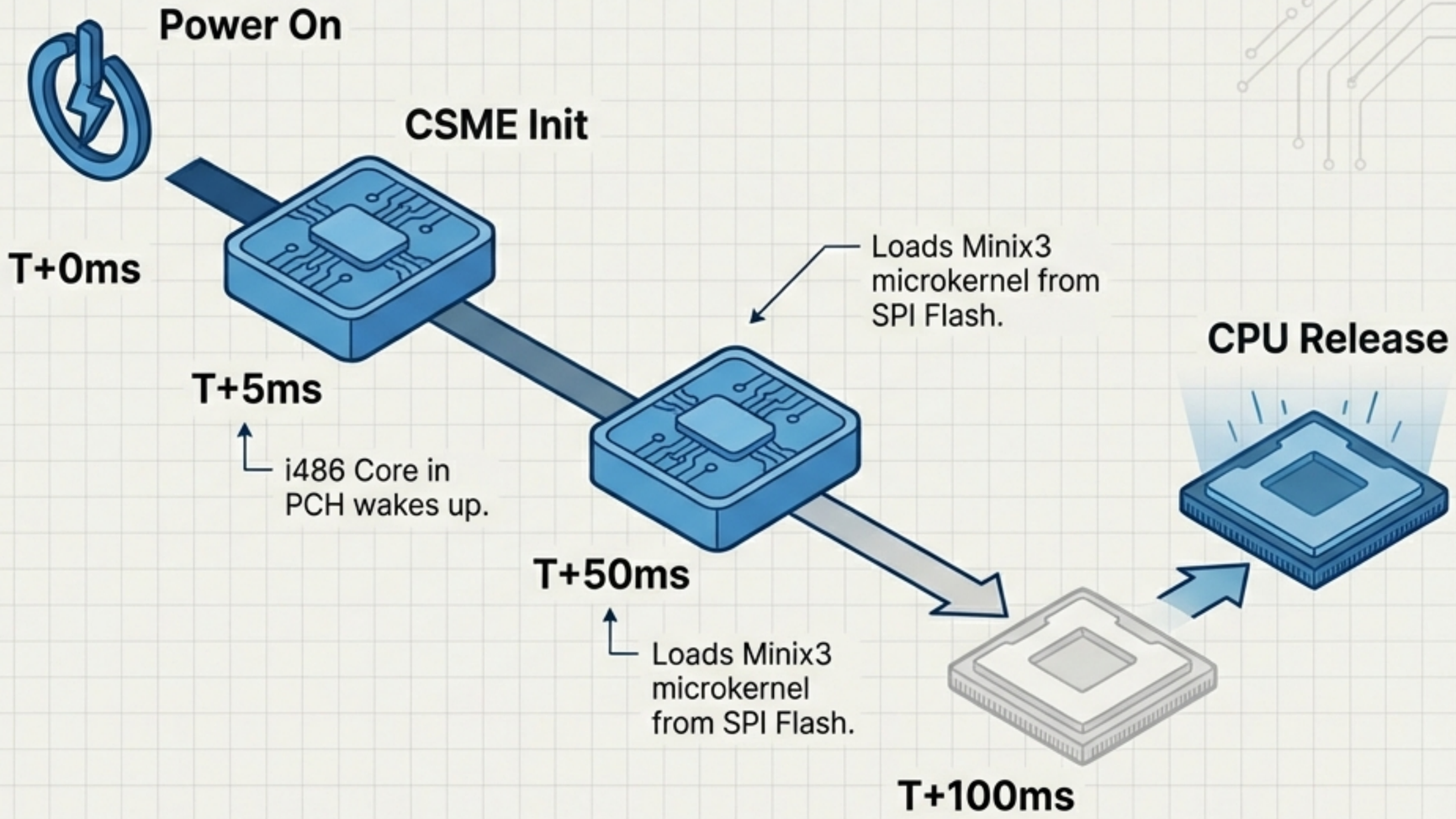
Pre-CPU: The Shadow Ruler (CSME)

The Hardware:

The CSME is an embedded processor inside the PCH. It operates independently of the main CPU, utilizing its own 128KB ROM and ~1.5MB SRAM.

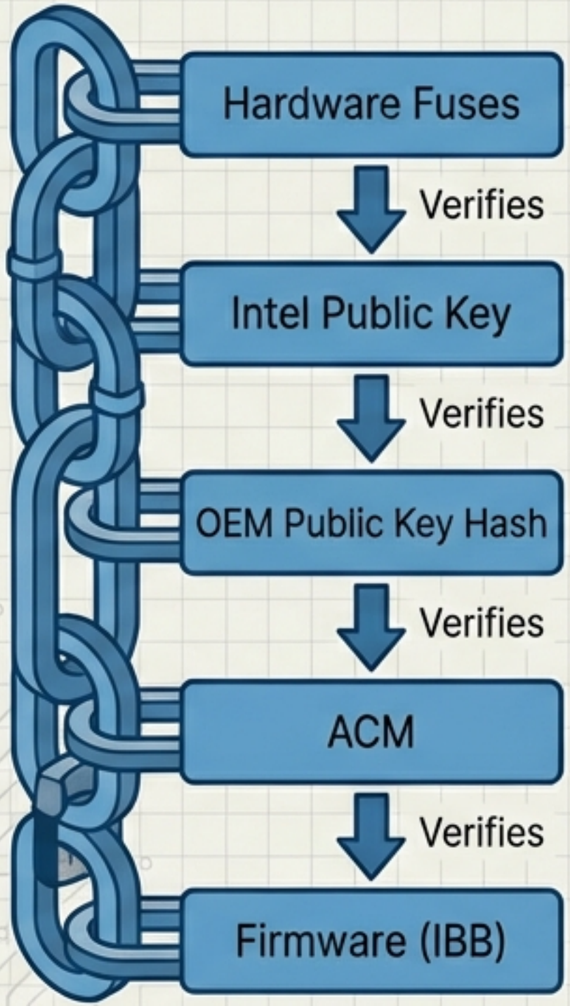
The Handoff:

The main CPU is held in a RESET state. It is only released when the CSME has verified the hardware profile and initialized basic power clocks.



Execution: 16-bit Real Mode | **Memory:** None | **Security:** Verifying...

CPU Reset & The Root of Trust



Hardware Fuses (FPF)

Intel Public Key

OEM Public Key Hash

ACM (Authenticated Code Module)

Firmware (IBB)

The First Breath

- **Reset Vector:** `0xFFFFFFF0`
  The CPU wakes here. It contains a `FAR JMP` instruction.
- **Microcode:** Loaded from Flash and verified against Intel's key ***before*** the first instruction completes.
- **Boot Guard:** The ACM runs in the CPU cache. It reads the OEM Key Hash from the fuses to cryptographically verify the Initial Boot Block (IBB).

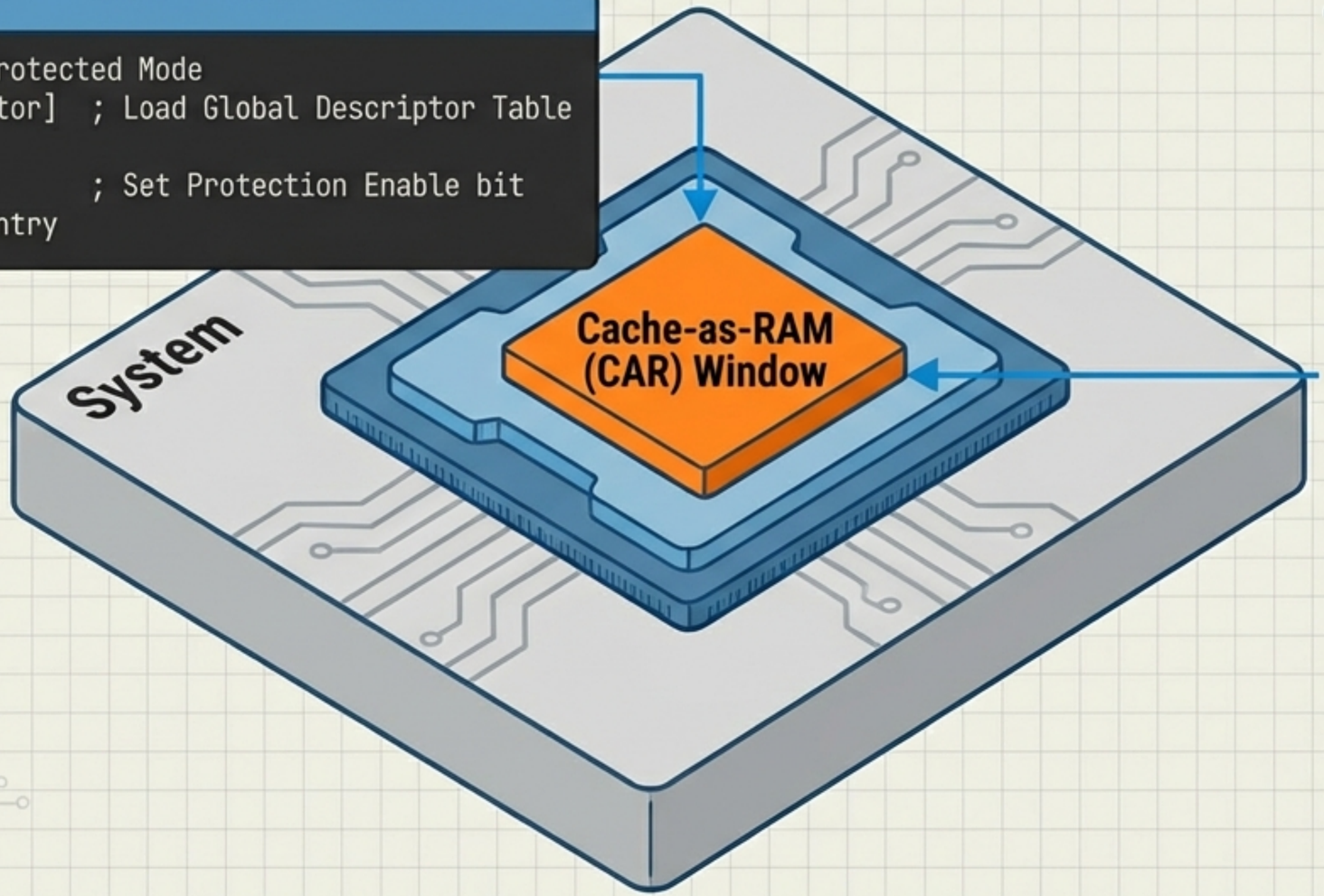
Safety Orange

Modes: Verified Boot (Halt on Failure) vs. Measured Boot (Record in TPM).

Execution: Real → Protected | **Memory:** CAR (Cache-as-RAM) | **Security:** Verified

Stage 1A: The 'No-RAM' Environment

```
; Transition to Protected Mode
lgdt [gdt_descriptor]  ; Load Global Descriptor Table
mov eax, cr0
or eax, 0x1            ; Set Protection Enable bit
jmp CODE_SEG:pm_entry
```



The Problem:

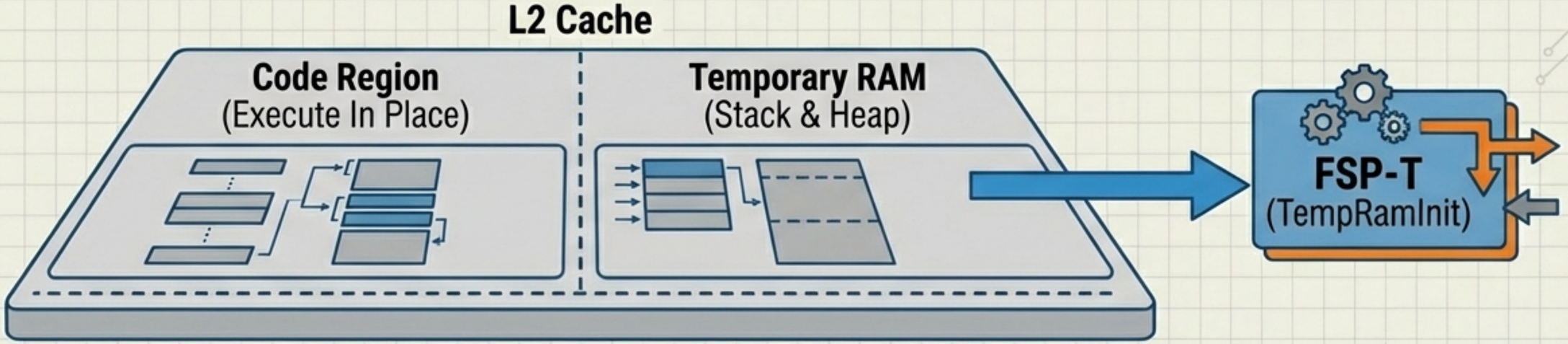
You cannot run complex C code without a stack. You cannot have a stack without RAM. And DRAM is not initialized yet.

The Solution:

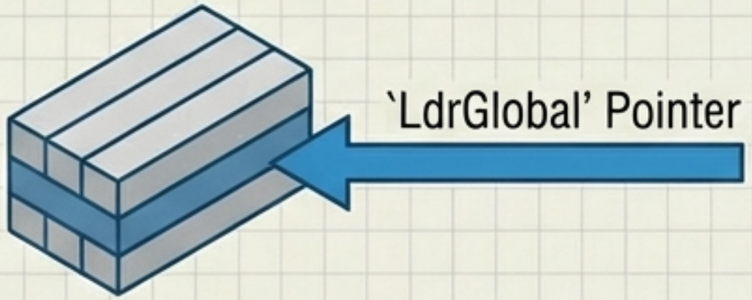
We must trick the CPU cache into behaving like RAM. Using MTRRs, we mark the cache as 'Write-back" with no eviction, creating a temporary scratchpad.

Execution: Real → Protected | **Memory:** CAR (Cache-as-RAM) | **Security:** Verified

FSP-T & The Temporary World



IDTR
(Interrupt Descriptor Table Register)



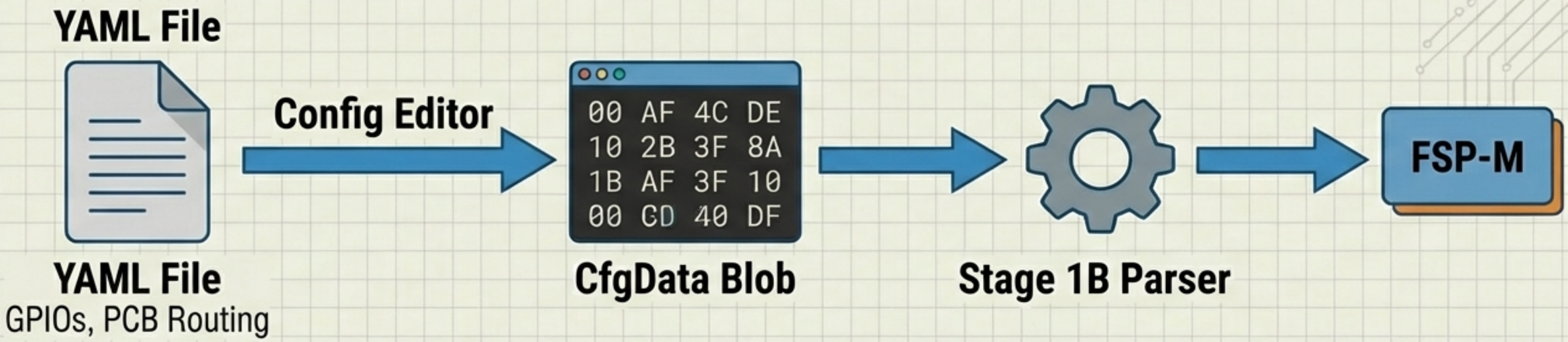
IDTR Hack

Interrupts are disabled. The IDTR is repurposed to store the global state pointer so it survives function calls.

FSP-T Output: Returns the range of available temporary memory **JetBraiins Mono** in registers `ECX` (Start) and `EDX` (End).

State Bar **Execution:** Protected | **Memory:** CAR | **Security:** Hash Check

Stage 1B: The Configuration Database



Verification:

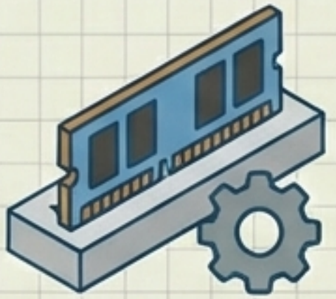
Stage 1A calculates the SHA-256 hash of Stage 1B. If it matches the manifest, execution jumps to `_ModuleEntryPoint`.

Why it matters:

FSP binaries are generic silicon drivers. They do not know the motherboard layout. Stage 1B loads the Config Data to tell the FSP how to train memory for *this specific* board.

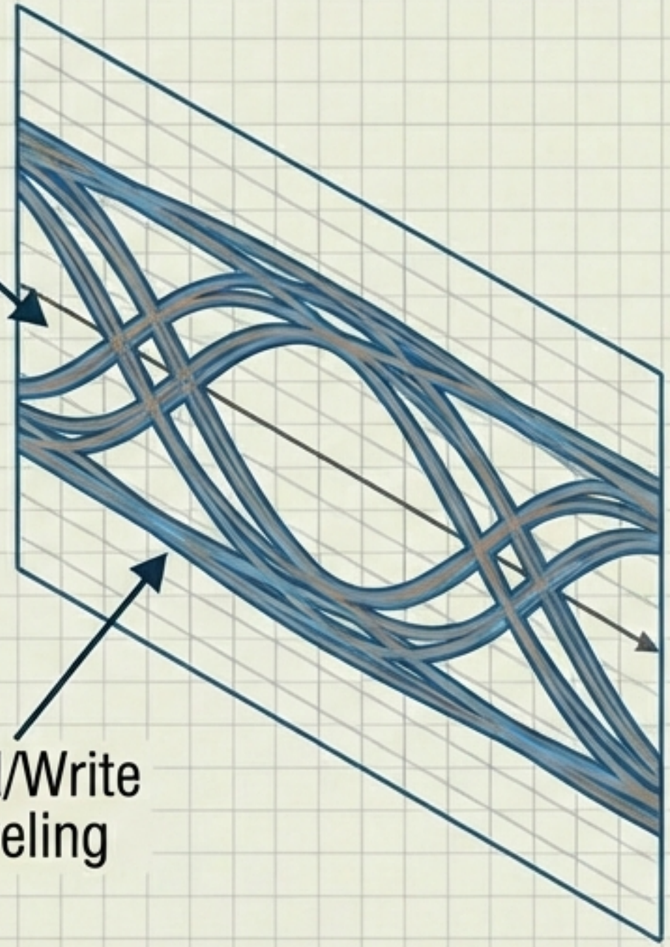
Execution: Protected | **Memory:** Permanent | **Security:** Verified

Memory Initialization (FSP-M)



The Task:

FSP-M reads SPD data from the DIMMs and electrically trains the memory bus to align timing.



Hand-Off Block

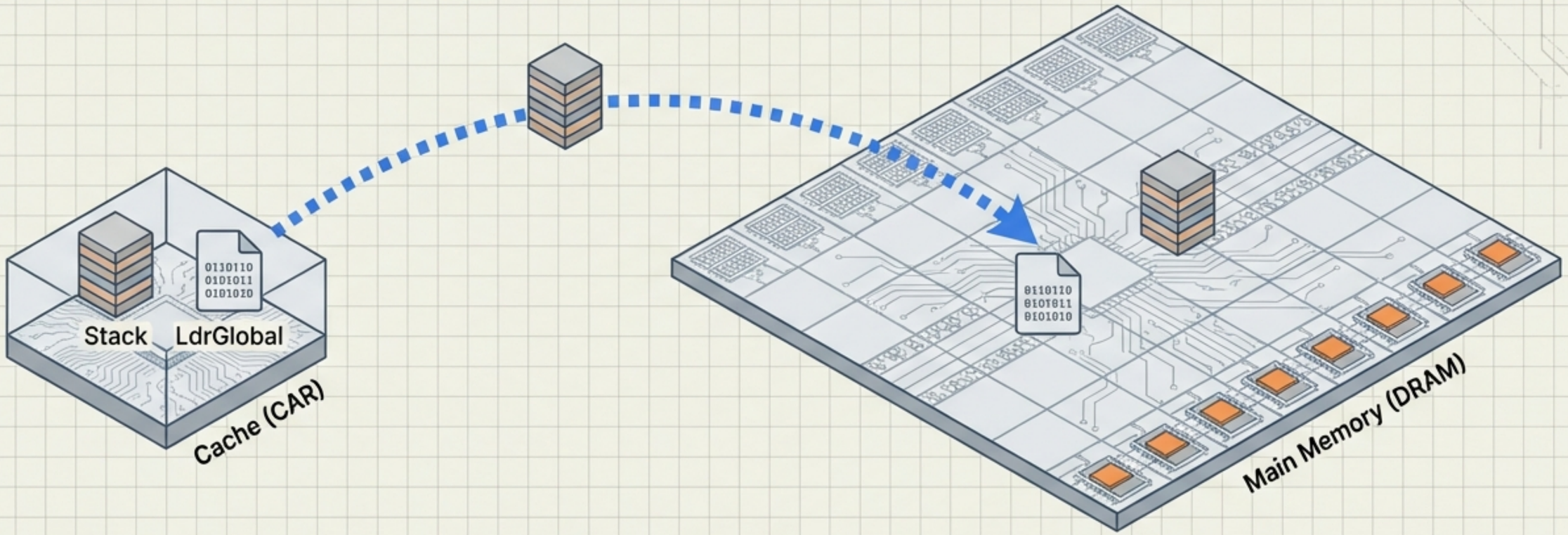
Resource Descriptor HOB

```
PhysicalStart: 0x000000000
PhysicalEnd:   0x3FFFFFFFF
Type:          EfiConventionalMemory
```

The Baton Pass:

FSP-M produces HOBs to tell the rest of the system what memory is valid and what is reserved.

The Great Migration (CAR to DRAM)



1. **Update:** Record new Top of Memory (TOLUM) in `LdrGlobal`.
2. **Shadowing:** `memcpy` Stack and Global Data to DRAM.
3. **Teardown:** Call `TempRamExit` to restore normal cache behavior.
4. **Launch:** Load Stage 2 from Flash into DRAM, **verify hash**, decompress (LZ4), and jump.

Stage 2: Silicon & Platform Init (FSP-S)

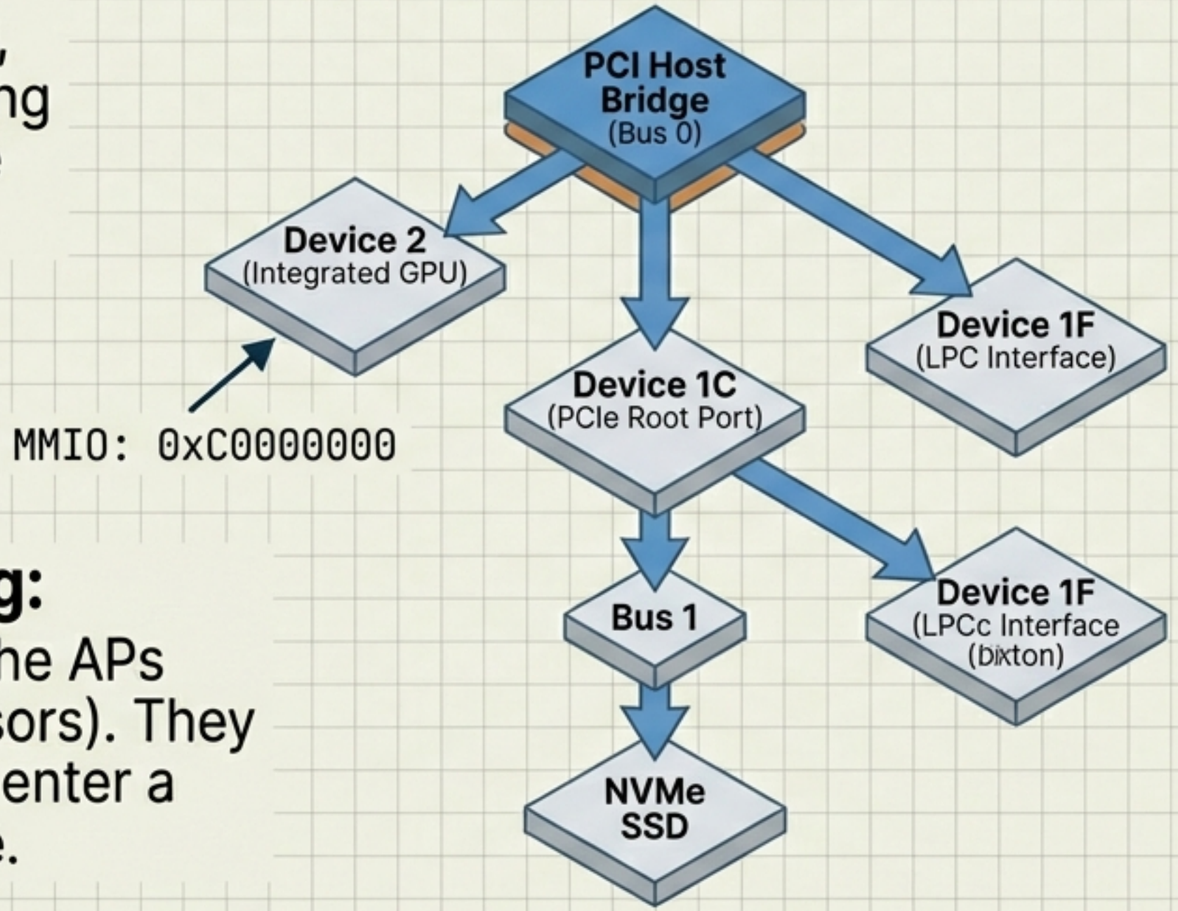
PCH Initialization:

FSP-S lights up USB, SATA, and Audio using parameters from the Config Data.

Multi-Processing:

The BSP wakes up the APs (Application Processors). They load microcode and enter a "**Wait-for-SIPI**" state.

PCI Enumeration



PCI Enumeration:

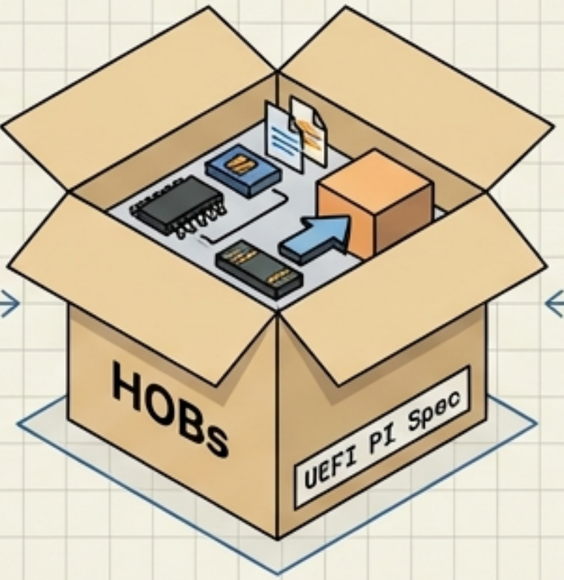
Walking the bus to discover devices, assign Memory-Mapped I/O (MMIO) addresses, and route interrupt lines.

The Language of Handoffs: LdrGlobal vs. HOBs

LdrGlobal (Loader Global Data)

- Proprietary, internal bootloader state.
- Content: Stack pointer, timestamps, debug levels.
- Rule: **NEVER shown to the OS.**

HOBs (Hand-Off Blocks)



- Universal currency (UEFI PI Spec).
- Content: Memory Maps, Graphics Info, ACPI Pointers.
- Rule: The Public API passed to the Payload.

State Bar | Execution: Protected | Memory: Permanent | Security: Verified

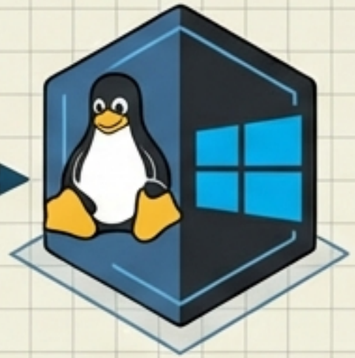
ACPI & The OS Map

FSP HOBs
(Raw Data)

Stage 2 Parser

ACPI Tables

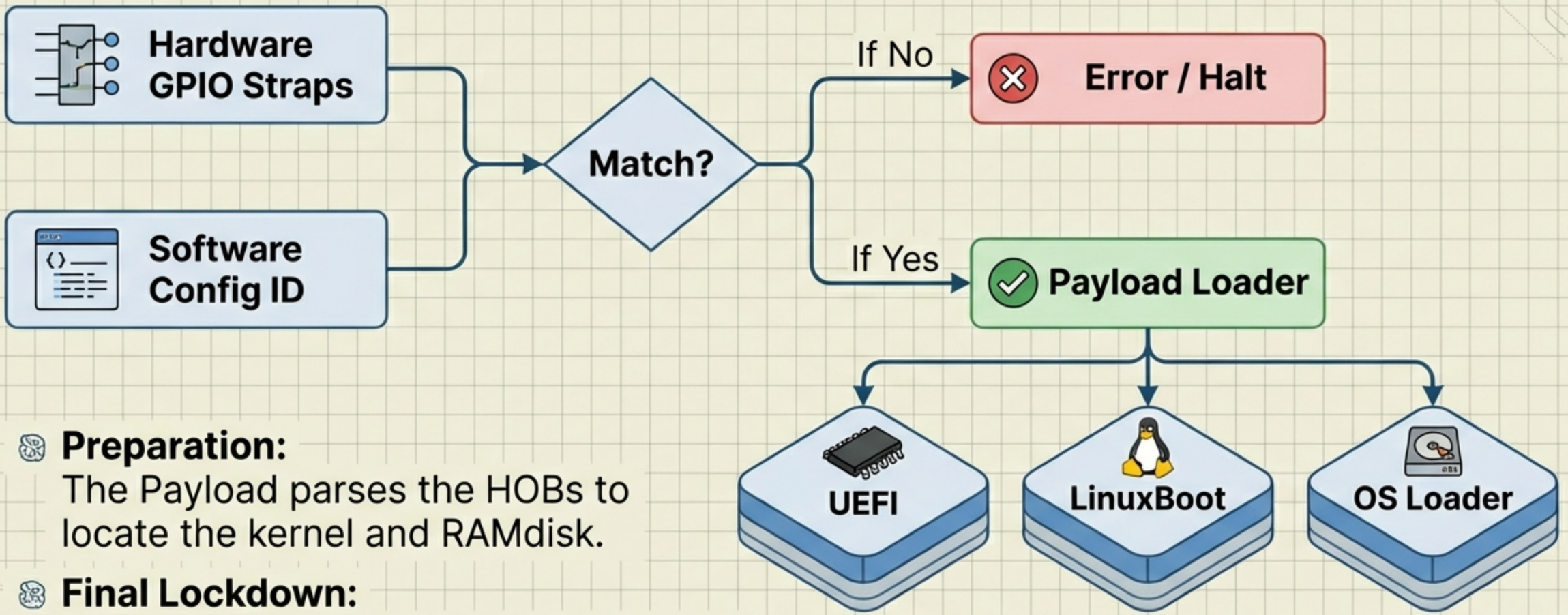
- **MADT** (Processors/Interrupts)
- **FADT** (Power Profiles)
- **DSDT** (Device Bytecode)
- **E820** (Memory Map)



OS Kernel

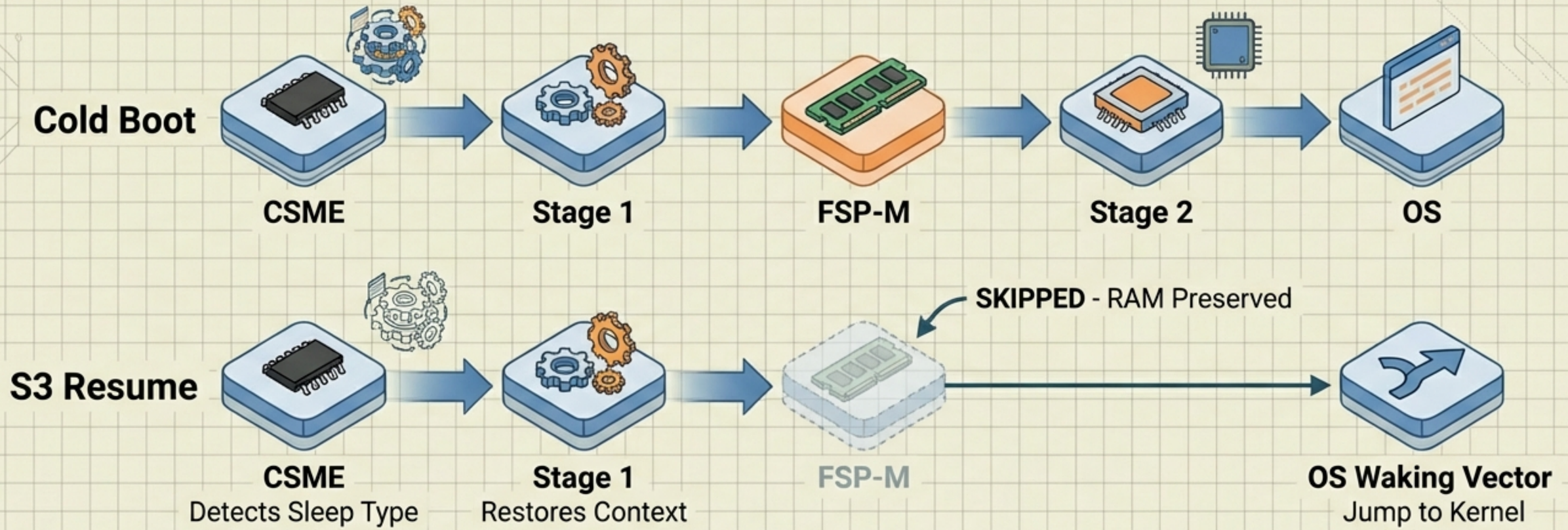
The OS assumes it owns the hardware, but it only knows what the Firmware (via ACPI) tells it exists.

Payload Selection & Execution



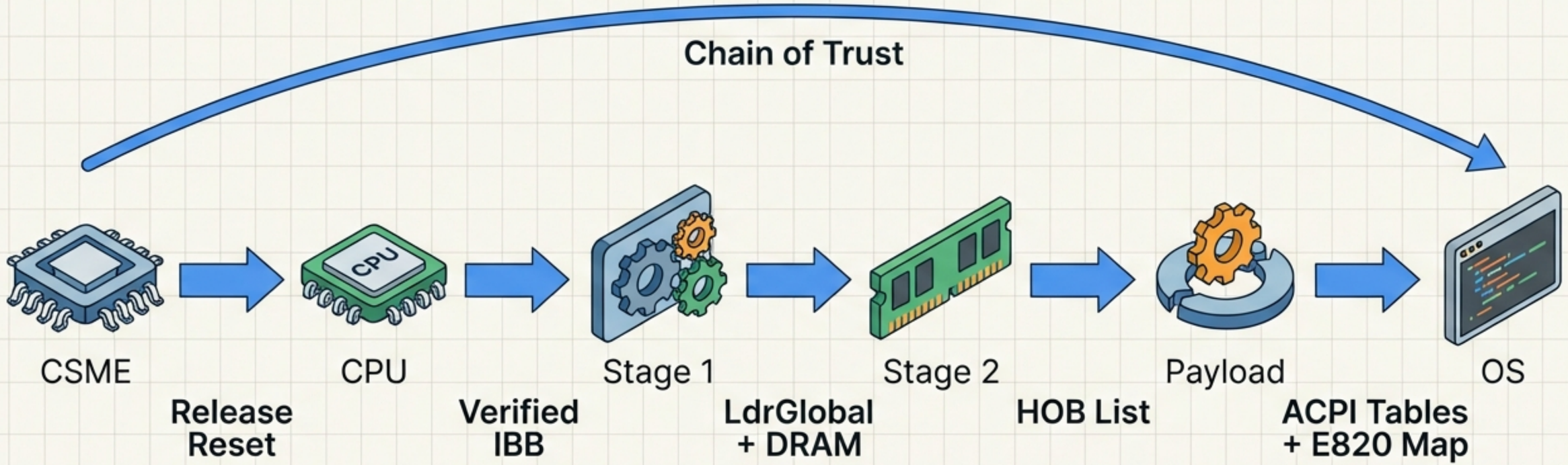
- **Preparation:**
  The Payload parses the HOBs to locate the kernel and RAMdisk.
- **Final Lockdown:**
  `NotifyPhase(ReadyToBoot)` is called.
  SMM is locked, and temporary memory is scrubbed.

The S3 Resume Path (The Shortcut)



- **The Difference:** RAM is in Self-Refresh (powered). Data is preserved.
- **The Shortcut:** Because memory is already trained, we skip the time-consuming FSP-M stage. The CPU restores state from the ACPI 'Waking Vector' instead of booting from scratch.

Summary: The Chain of Trust



Final Takeaways:

1. **Security** is hardware-anchored (Fuses).
2. **Evolution** moves from Flash (XIP) to Cache (CAR) to DRAM.
3. **Data** evolves from proprietary `LdrGlobal` to standard HOBs, and finally to OS-readable ACPI.