VMS Software

WHAT WE LEARNED FROM THE PORT

Eindhoven, 7-MAR-2023

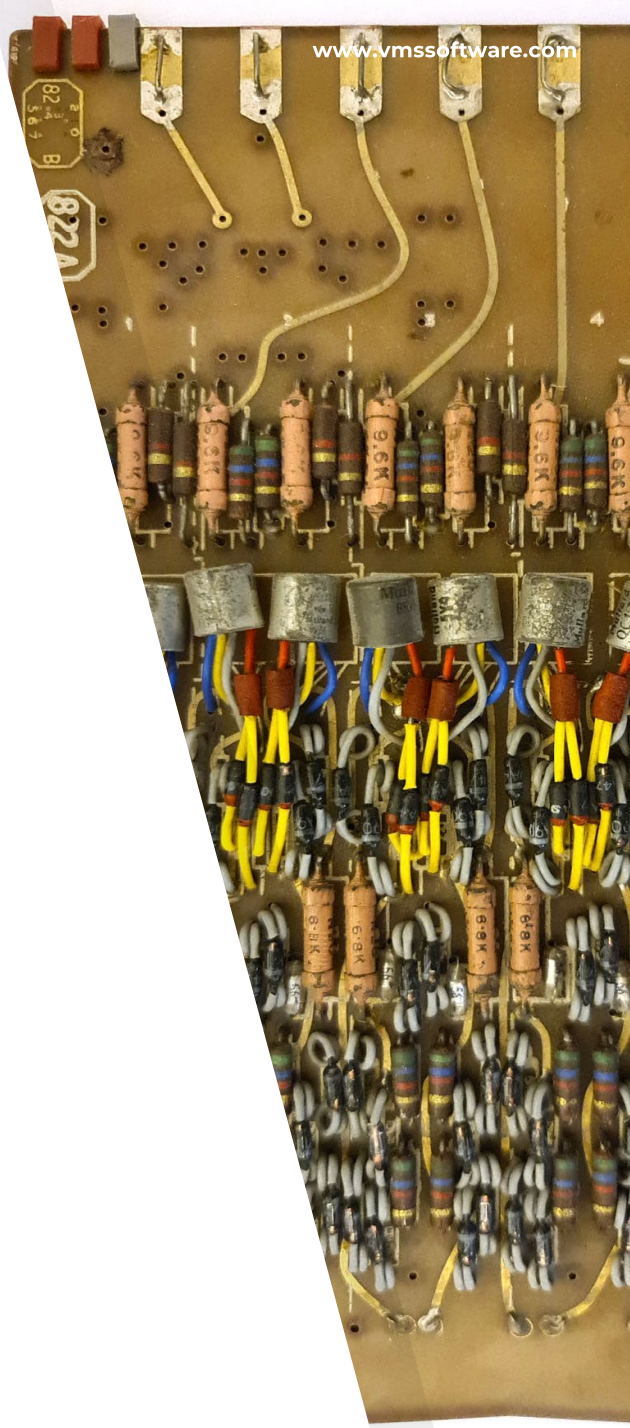
Camiel Vanderhoeven

About me

- The Netherlands, 1977, educated in Eindhoven
- VMS Kernel developer since 2015
  - Mode changes
  - 4 modes on a 2 mode architecture
  - Emulated VAX features
    - ASTs
    - Software Interrupts
    - Queue Instructions
  - Scheduler
  - POSIX threads
  - AMD CPU Support
- Computer collection: www.vaxbarn.com

Agenda

- Developing on Virtual Machines
- Masquerade
- Why probe?
- “Unified” Extensible Firmware Interface
- “Let’s be different!”



Developing on Virtual Machines

- Quick to setup
- Quick to boot
- Snapshots, cloning
- Post-mortem

Masquerade

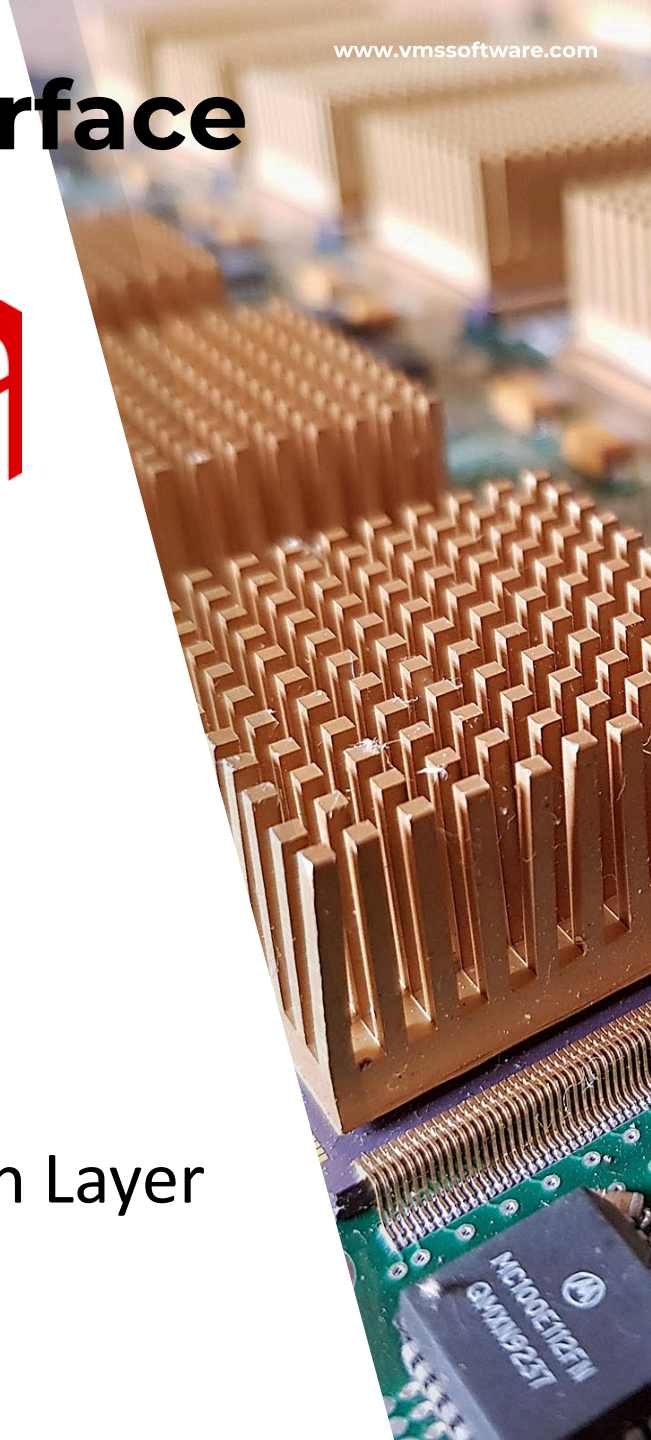
- 4 modes
  - 4 pagetables per process
  - Complex code for transitioning between modes
- 8k pages
  - Native pages on x86 are 4k
  - VMS pages are 8k
  - Memory needed for UEFI may be in adjacent 4k pages
- Queue Instructions
- Probe Instructions

Why Probe?

- System service calls
  - Verify caller has access
  - Avoid pagefaults at elevated IPL
- Interrupts / exceptions
  - Avoid pagefaults at elevated IPL
  - Invalid stack pointers
- NO PROBE INSTRUCTION ON X86
  - Walk the page tables “by hand”
  - Validate input parameters
  - ~ 900 instructions executed

“Unified” Extensible Firmware Interface

Where we came from:



- Alpha
  - Single vendor: Digital / Compaq / HP
  - SRM
  - Galaxy Configuration Tree
- Itanium
  - Single vendor: HP / HPE
  - Extensible Firmware Interface
  - System Abstraction Layer / Processor Abstraction Layer hides

“Unified” Extensible Firmware Interface

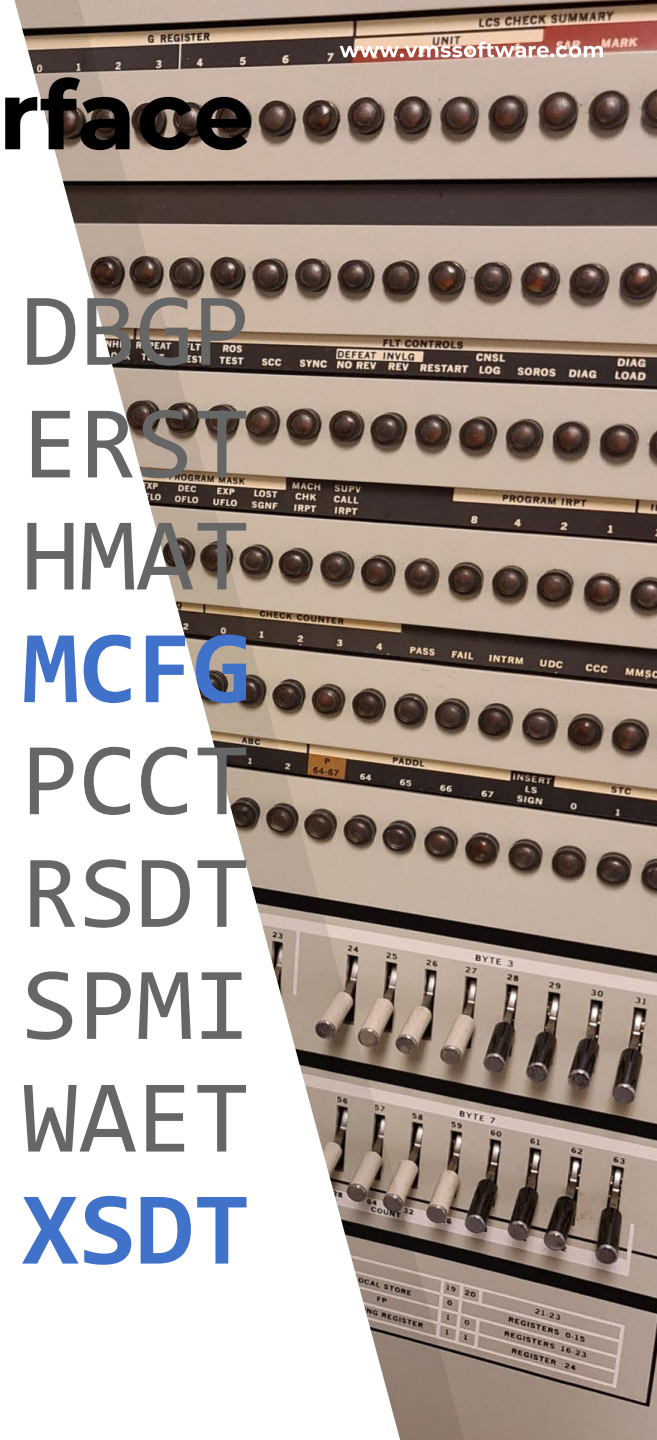
Where we ended up:

- X86-64
  - Multiple vendors / hypervisors
  - UEFI (without SAL or PAL)
  - ACPI (advanced Configuration and Power Interface)

  - Every hypervisor is different
    - Sometimes even between versions of the same Hypervisor
  - Bare-metal hardware is different again

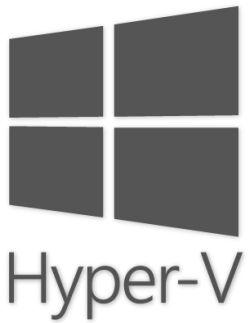
“Unified” Extensible Firmware Interface

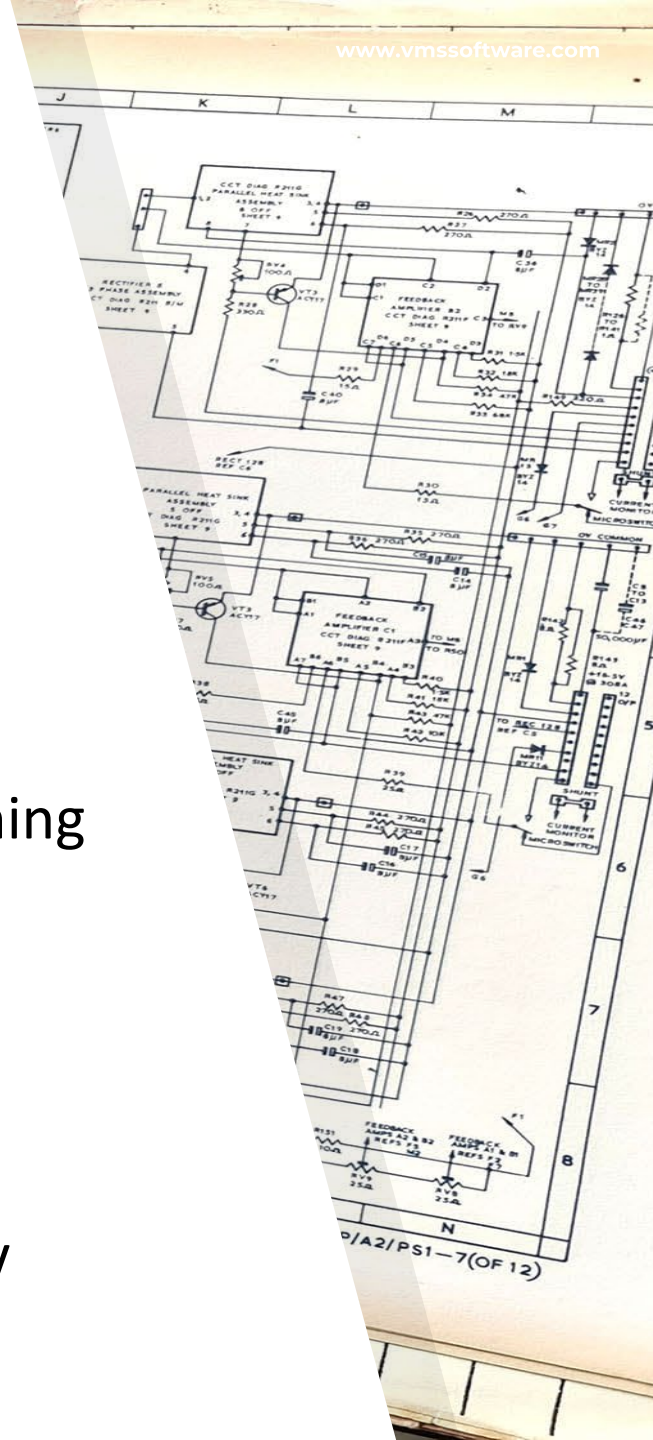
BERT BOOT **BGRT** CPEP CSRT DBG2 DBGP
DSDT DMAR DPPT DRTM ECDT EINJ ERST
ETDT **FACS** **FADT** FPDT GTDT HEST HMAT
HPET IBFT IORT IVRS LPIT **MADT** **MCFG**
MCHI MPST MSCT MSDM NFIT OEMx PCCT
PDTT PMTT PPTT PSDT RASF **RSDP** RSDT
SBST SDEI SDEV SLIC SLIT SPCR SPMI
SRAT **SSDT** STAO TCPA TPM2 UEFI WAET
WDAT WDDT WDRT WPBT WSMT XENV **XSDT**



“let’s be different”

- VMware, VirtualBox, XEN, KVM
  - Have their differences, but …
  - Virtual machines pretend to be real hardware
  - Provide emulated devices and controllers – SCSI, SATA, Intel NIC, Chipset
  - Can provide higher speed, lower latency virtualized I/O interfaces *if the OS asks to use them*
  - In short: are able to accommodate OS’es that know nothing about Virtual Machines
- Hyper-V
  - Virtual machines are unapologetically virtual constructs
  - Does not provide emulated devices or controllers
  - Requires that the OS use virtualized I/O interfaces
  - In short: requires the OS to know how to run on Hyper-V






Questions?