vms Software

State of the Port to x86_64
September 2016

September 6 , 2016

vms Software

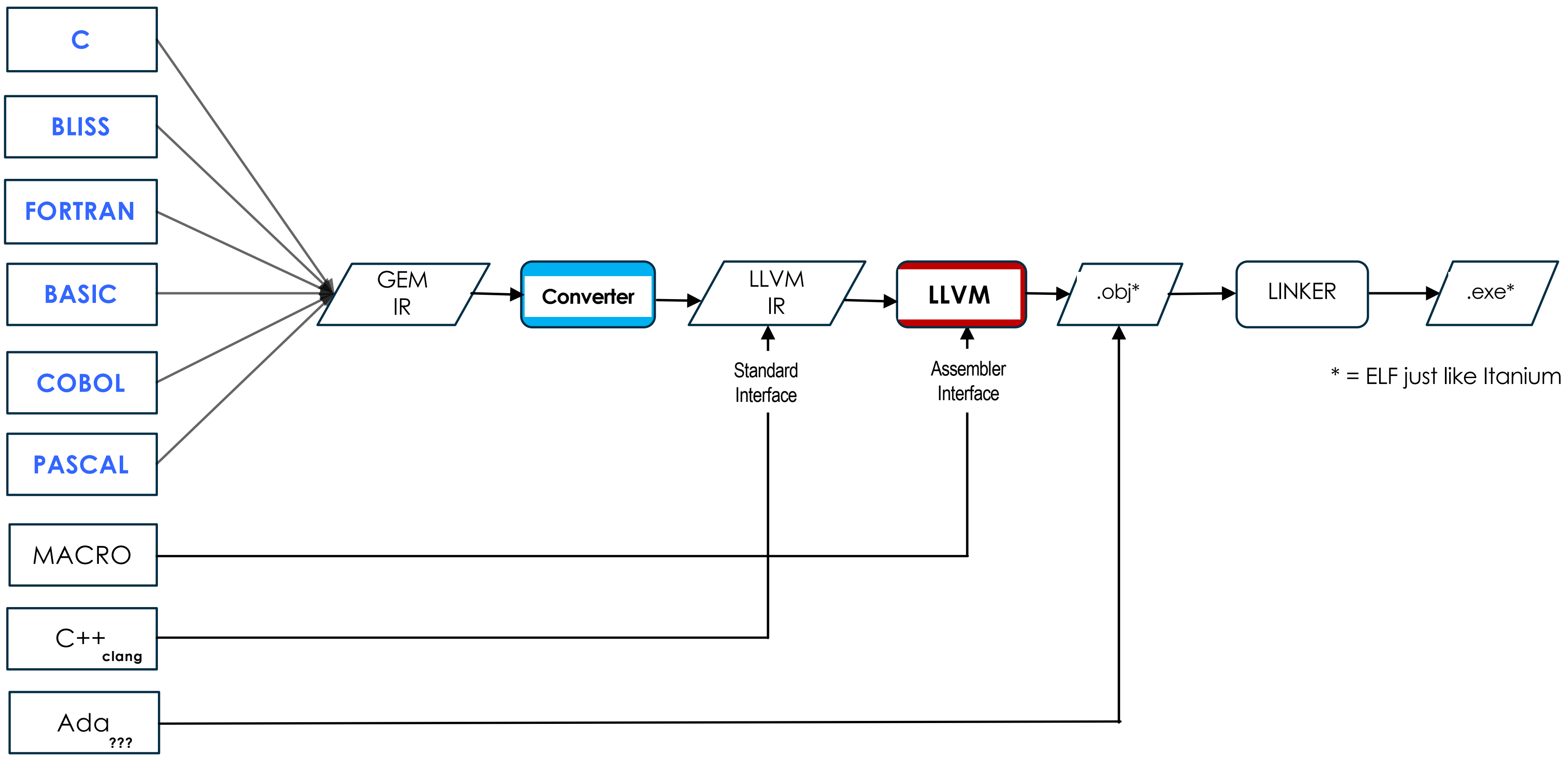
State of the Port to x86_64
September 2016

This information contains forward looking statements and is provided solely for your convenience. While the information herein is based on our current best estimates, such information is subject to change without notice.

Update Topics

- Code Generation
- Re-Architecting the Early Boot Path
- Dump Kernel
- Boot Manager Graphical Interface
- Memory Management
- 2-Mode Prototyping
- Paravirtualized Drivers
- Software Interrupt Services (SWIS)

Future VMS Compiler Strategy



* = ELF just like Itanium

- **Continue with current GEM-based frontends**
- Use open source LLVM for backend code generation
- ***Create internal representation (IR) converter***
- LLVM targets x86, ARM, PowerPC, MIPS, SPARC, and more

Code Generation

- GEM-to-LLVM (G2L) Converter
  - Started with C frontend
  - Focus on language constructs (and ignore builtins)
  - Converter continues to grow and improve
- C Compiler
  - DEC C Test Suite: 4000+ of 4200 compilation tests pass
  - Other tests:
    - 2049 compilation tests – 102 failures
    - 1336 run-time tests – 11 failures
  - Most failures due to things not yet implemented
  - For runtime tests
    - On VMS: LLVM outputs bitcode file
    - On linux: link with clang and run
  - Started compiling OS modules
  - Continuing to automate test analysis

Code Generation, continued

- As G2L matured, started building BLISS compiler
  - Adding BLISS requirements not already exposed by C
  - Generalizing some C-centric assumptions in G2L
  - Streamlining G2L’s parsing of GEM IR
- BLISS Compiler: Compiling simple programs
- MACRO Compiler
  - XMACRO uses different LLVM interface than used by C and BLISS
  - LLVM integration complete
  - Preliminary x86 code generation

Re-Architecting the Early Boot Path

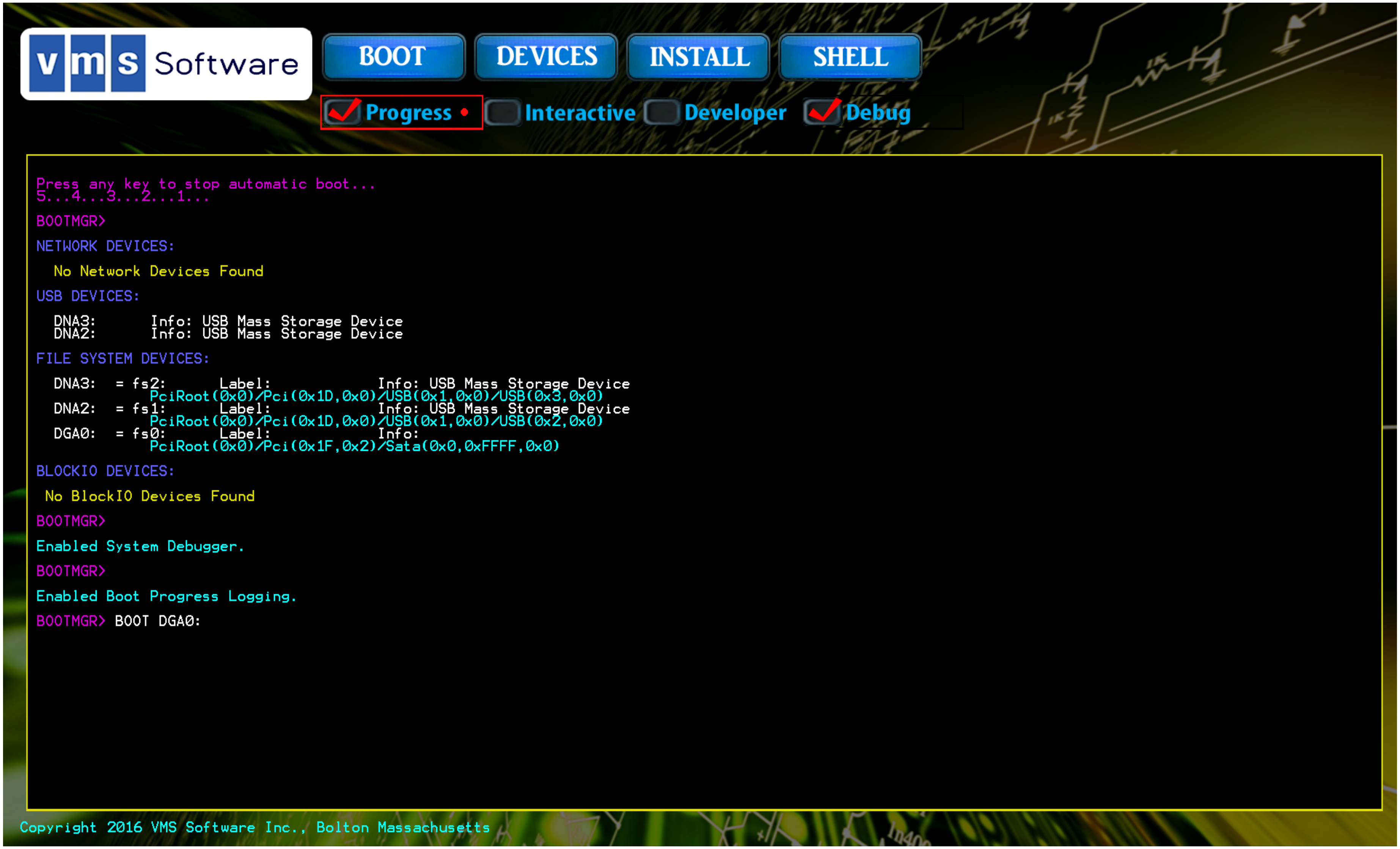
- Itanium: VMS_LOADER.EFI / IPB / SYSBOOT
- x86: VMS_BOOTMGR.EFI / ~~XPB~~ / SYSBOOT
- Goals:
  - Always boot from Memory Disk
  - Eliminate the need for boot drivers
  - Never touch the "primitive file system” again
- Results: Mission Accomplished!
- Now loading fewer files
- Never write another xxBTDRIVER
- On-disk structure of the system disk is no longer a factor prior to loading the full file system
- Q: Without boot drivers, how do you write crash dumps?
- A: The Dump Kernel

Dump Kernel

- A second, minimalist OS instance is loaded into memory during normal boot - but not booted
- Its memory allocation has a special tag
- As the system goes down
    - The crashing primary kernel gathers the necessary information
    - BUGCHECK notifies the Boot Manager to boot the Dump Kernel
- This is extremely fast since
    - the Dump Kernel is already in memory and
    - it only needs to boot as far as a specialized SYSINIT
- The Dump Kernel writes the dump file using the run-time driver and initiates a shutdown

Boot Manager Graphical Interface

An early Snapshot, much more to come



☑ Progress ☐ Interactive ☐ Developer ☑ Debug

```
Press any key to stop automatic boot...
5...4...3...2...1...
BOOTMGR>
NETWORK DEVICES:
  No Network Devices Found
USB DEVICES:
  DNA3:      Info: USB Mass Storage Device
  DNA2:      Info: USB Mass Storage Device
FILE SYSTEM DEVICES:
  DNA3:  = fs2:      Label:            Info: USB Mass Storage Device
             PciRoot(0x0)/Pci(0x1D,0x0)/USB(0x1,0x0)/USB(0x3,0x0)
  DNA2:  = fs1:      Label:            Info: USB Mass Storage Device
             PciRoot(0x0)/Pci(0x1D,0x0)/USB(0x1,0x0)/USB(0x2,0x0)
  DGA0:  = fs0:      Label:            Info:
             PciRoot(0x0)/Pci(0x1F,0x2)/Sata(0x0,0xFFFF,0x0)
BLOCKIO DEVICES:
 No BlockIO Devices Found
BOOTMGR>
Enabled System Debugger.
BOOTMGR>
Enabled Boot Progress Logging.
BOOTMGR> BOOT DGA0:
```

Copyright 2016 VMS Software Inc., Bolton Massachusetts

Memory Management

- **Review:**
  - Two processor modes (kernel, user)
  - VMS fabricates two (executive, supervisor)
  - Four levels of page tables
  - VMS needs separate pages tables per mode
  - No PROBE instruction; look it up in page tables
  - Page sizes – 4KB, 2MB, 1GB
- **SYSBOOT**
  - Initial debugging: compile/link with Windows Visual Studio (just like the Boot Manager)
  - Some code has been converted from BLISS to C in order to make early progress
  - Memory bit map constructed based on memory descriptors passed from boot manager
  - Currently implementing PFN database and page tables

Running in Two Processor Modes

- **Review:**
  - x86 has four modes (rings) 0, 1, 2, 3 but they do not provide the strict hierarchy of memory access protection expected by VMS
  - Example: Cannot allow kernel write and prevent exec write
  - VMS will run in two modes: kernel (0) and user (3)
  - Supervisor and Executive modes implemented in software
- ***2-Mode Prototyping on Itanium Completed***
- **Results:**
  - PROBE emulation does PTE lookup
  - Did not boot and run complete system
  - Did get far enough to verify transitions to/from Supervisor mode with correct access and mode information
- **Proved the methodology to be sound**
- **Implementation details will differ, but not greatly, on x86**

Paravirtualization

- Using the VIRTIO API for storage driver
- VIRTIO implemented by many hypervisors (KVM, XEN, ….)
- Using linux VIRTIO header files
- Using existing VMS driver GSPDRIVER (Generic Storage Port driver) as a model
- Completed the SCSI/fibre channel IOGEN configuration code
- Started implementation of driver routines
- Evaluating other paravirtualized drivers such as network and console

Software Interrupt Services (SWIS)

- **Review:**
  - SWIS is conceptually architecture independent but code is specific for each platform and it performs the same logical functions
- **Recently Completed:**
  - Design for accessing per-CPU data
  - Basic design for mode changes (system services and interrupts)
- **In progress:**
  - Detailed design for system service calling
  - Detailed design for interrupt and exception handling

vms Software

For more information, please contact us at:

RnD@vmssoftware.com

VMS Software, Inc. • 580 Main Street • Bolton MA 01740 • +1 978 451 0110