OpenVMS Port to the Itanium® Architecture

Thomas Siebold
Technology Consultant
Alpha Systems Division

Thomas.siebold@hp.com

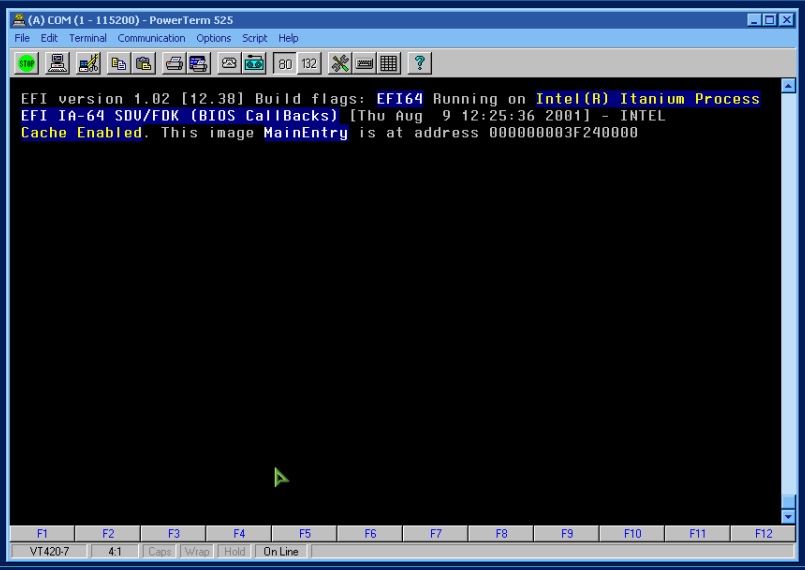
„Bitte schalten Sie Ihr *Handy* aus !“

- This is a ‚mobile phone‘
- ...but in Germay it is called a ‚handy‘
- ...but in other countries a ‚handy‘ is a .....

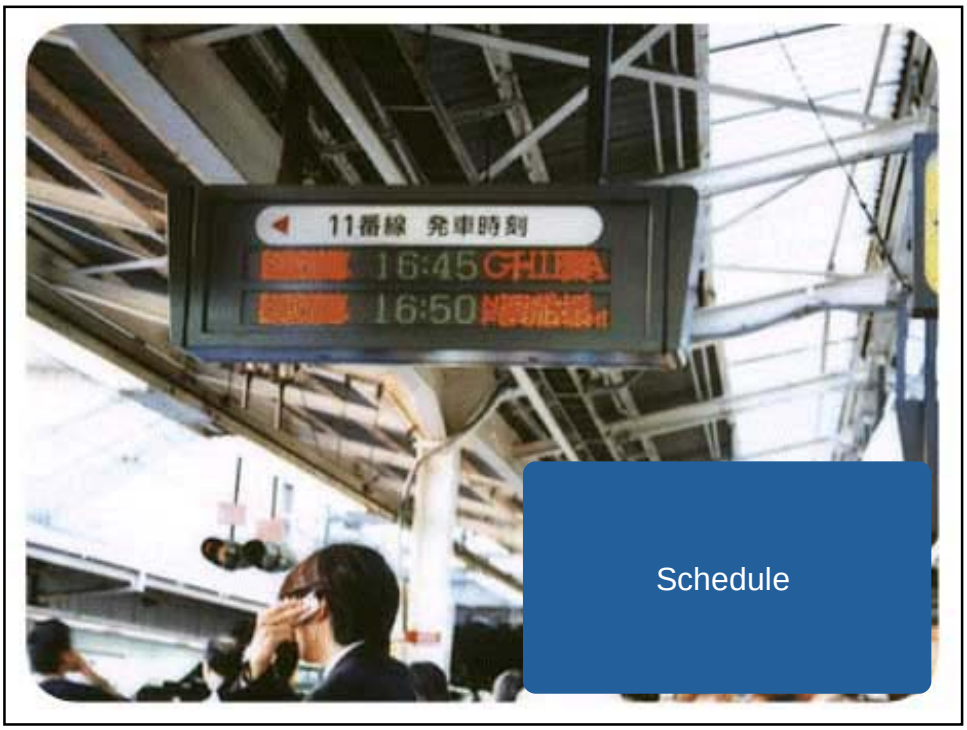
Agenda

- ***Surprise*** !!!
- Base System Port
- Systems for OpenVMS
- Layered Products
- Applications

Surprise – It is real ! ☺



OpenVMS on Itanium® -- 31. Januar 2003 15:31

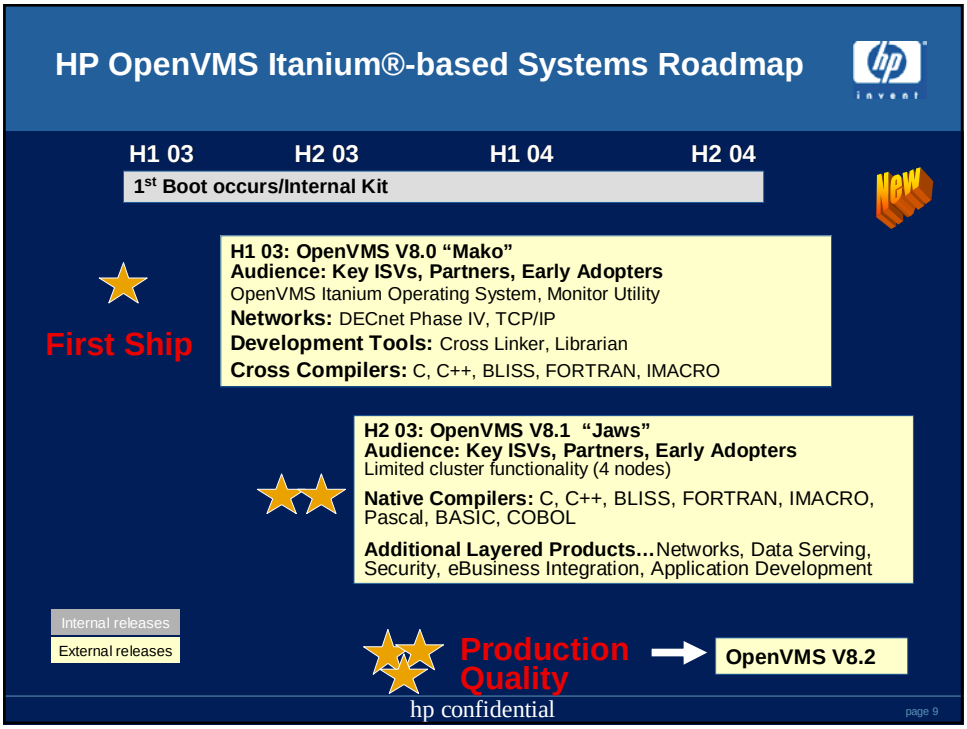


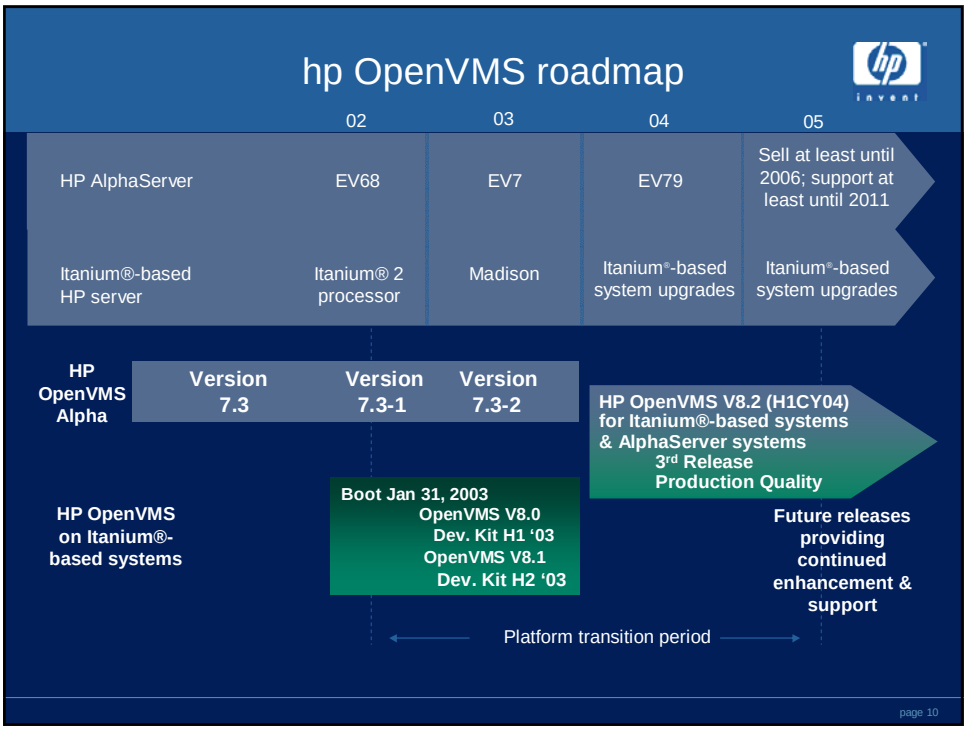
Schedule

Schedule

Development Schedule

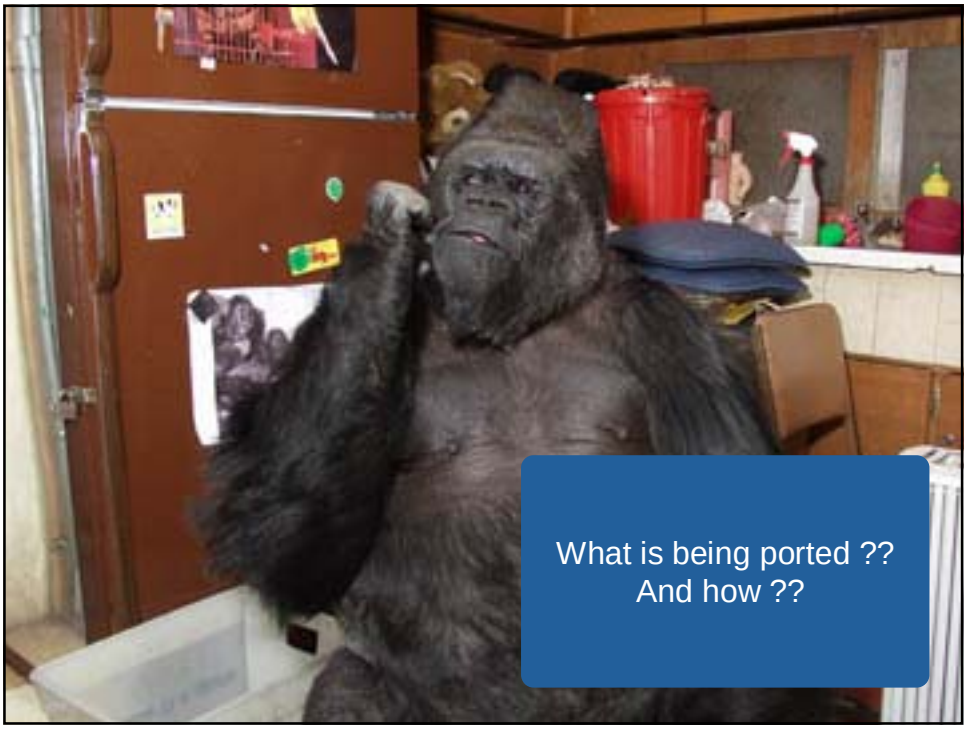
- H1 2003 - Release 8.0:
  - selected ISVs, SW partners, early adopters;
  - contains limited layered products
- H2 2003 - Release 8.1:
  - more selected ISVs, SW partners, early adopters;
  - contains more layered products
- H1 2004 - Release 8.2:
  - production quality, general customer release


HP OpenVMS Itanium®-based Systems Roadmap
H1 03
H2 03
H1 04
H2 04
1st Boot occurs/Internal Kit
New
First Ship
H1 03: OpenVMS V8.0 "Mako"
Audience: Key ISVs, Partners, Early Adopters
OpenVMS Itanium Operating System, Monitor Utility
Networks: DECnet Phase IV, TCP/IP
Development Tools: Cross Linker, Librarian
Cross Compilers: C, C++, BLISS, FORTRAN, IMACRO
H2 03: OpenVMS V8.1 "Jaws"
Audience: Key ISVs, Partners, Early Adopters
Limited cluster functionality (4 nodes)
Native Compilers: C, C++, BLISS, FORTRAN, IMACRO, Pascal, BASIC, COBOL
Additional Layered Products...Networks, Data Serving, Security, eBusiness Integration, Application Development
Internal releases
External releases
Production Quality
OpenVMS V8.2
hp confidential
page 9


hp OpenVMS roadmap
02
03
04
05
HP AlphaServer
EV68
EV7
EV79
Sell at least until 2006; support at least until 2011
Itanium®-based HP server
Itanium® 2 processor
Madison
Itanium®-based system upgrades
Itanium®-based system upgrades
HP OpenVMS Alpha
Version 7.3
Version 7.3-1
Version 7.3-2
HP OpenVMS V8.2 (H1CY04) for Itanium®-based systems & AlphaServer systems
3rd Release
Production Quality
HP OpenVMS on Itanium®-based systems
Boot Jan 31, 2003
OpenVMS V8.0
Dev. Kit H1 '03
OpenVMS V8.1
Dev. Kit H2 '03
Future releases providing continued enhancement & support
Platform transition period
page 10

What's being ported…

….and how ?



What version of OpenVMS is being ported?

- We are adding support to the OpenVMS AlphaServer code base for the Itanium® architecture.
- We will create releases from the same sources for both AlphaServer and Itanium®-based systems.
- The first Itanium® architecture release will reflect on-going OpenVMS development work (performance improvements, device support, etc.)

Porting Philosophy

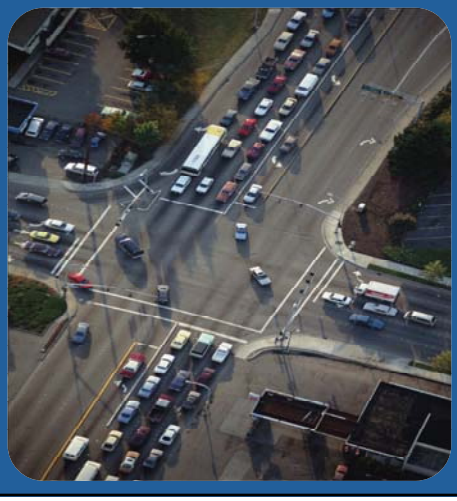
•This is not a “bug for bug compatible” coding exercise. We are doing much, much more.

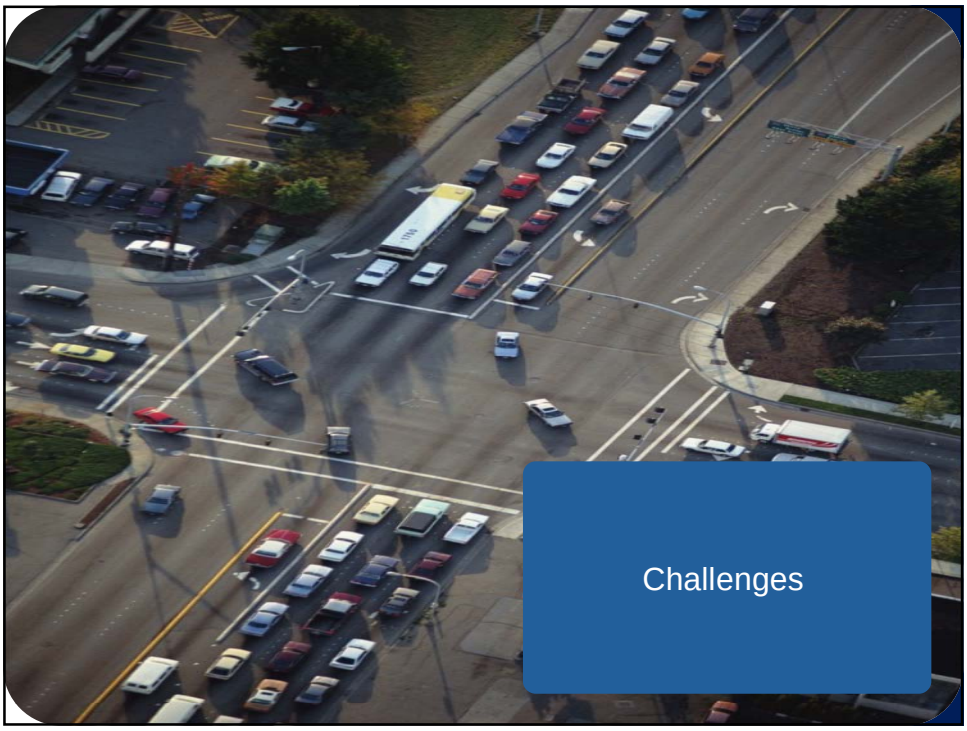
1. we are not just porting to Itanium® architecture; we are making *OpenVMS* more portable.

2. we are improving code maintainability (and sometimes performance) by replacing VAX assembler code when appropriate.

3. we are making the system more open to the possibility of exchanging code with other systems, especially analysis tools. (This has already helped us debug some new code.)

The Challenges





Alpha-to-Itanium® vs. VAX-to-Alpha

- **VAX-to-Alpha**
  - CISC to RISC
  - huge volume of coding work
  - make 1100+ VAX MACRO-32 modules compileable
    - 32b to 64b
    - data alignment
    - atomicity
    - multiple, out-of-order execution streams
- **Alpha-to-Itanium®**
  - more complex
  - less coding
  - 64bits vs 64bits
  - RISC / EPIC

lines of code perspective

OpenVMS on Itanium®-based systems

Porting effort is very focused on a few areas of the system.

Big Challenges for the Base OS

- No Alpha Console
  - Booting
  - Device Discovery
  - Interrupts
  - TLB miss handler
- No Alpha PALcode
  - VAX Queue Instructions
  - VAX Registers
  - IPL and mode change
- Different primitives in CPU
  - Register Conventions
  - Exception Handling
  - Atomic Instructions
  - Process Context
- Plus, we decided to change
  - calling standard
  - object language
  - image format

Compare to move from VAX OpenVMS

- Customer Work
  - User-written system services
  - User-written device drivers
  - Tools interpreting object, executable formats
- ***Most applications*** should compile and go
- User mode will not see a difference on OpenVMS on Itanium®

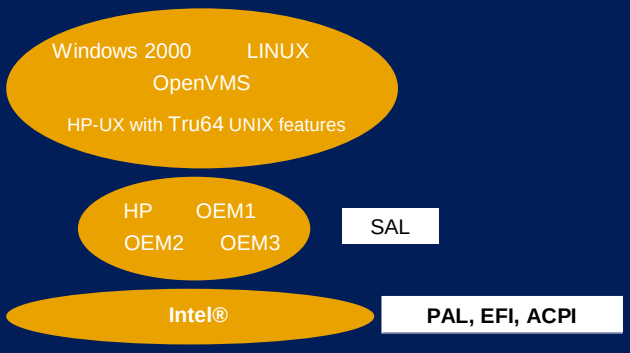
Progress Report

It's All in the Software

SW	Application	Application	Application
	OpenVMS	OpenVMS	OpenVMS
FW	Console	Console PALcode	Console
HW	VAX	Alpha	Itanium® architecture

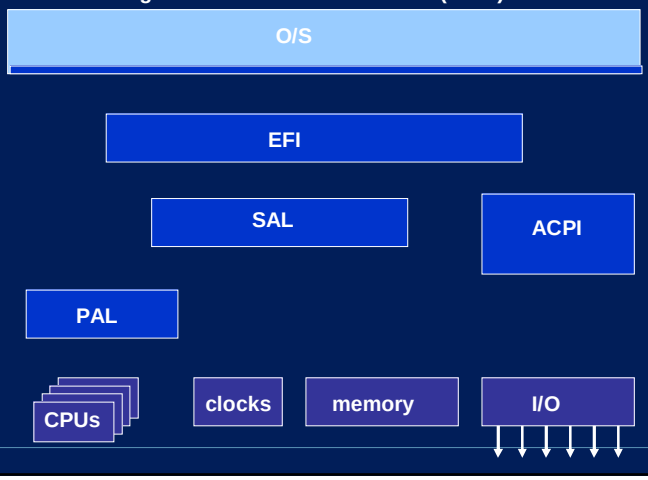
Itanium® console architecture

- processor abstraction layer (PAL)
- system abstraction layer (SAL)
- extensible firmware interface (EFI)
- advanced configuration and power interface (ACPI)



Itanium Firmware Architecture

- **Processor Abstraction Layer (PAL)**
- **System Abstraction Layer (SAL)**
- **Extensible Firmware Interface (EFI)**
- **Advanced Configuration and Power Interface (ACPI)**



early boot: progress report

- First boot on January 31, 2003
- First boot environment with ‘cheats’ to help development
  - Images loaded during boot are ‘execlets’, e.g. pieces of executive
- Commands that stay in DCL work
- Calling a shared image, e.g. ‘DIR’ crashes…
- OpenVMS booted on a rx2600 in late March

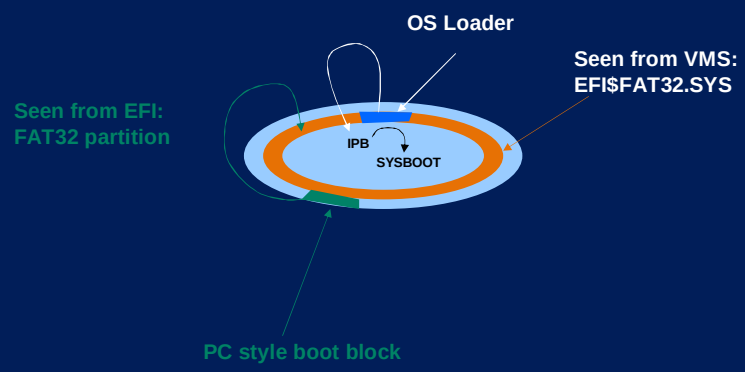
Loading Images: Progress Report

- Images are industry standard “inside”
  Executable and Linkable Format (ELF)
- Exec Loader
  - Major MACRO-32 module rewritten in C
  - Linked into SYSBOOT
  - Debugged and running on Alpha
  - Relocations and fixups in progress
- Image Activation and INSTALL
  - adding ELF knowledge and will test on Alpha first

Console: Booting on EFI

- Requires “OS loader” to be in a FAT32 file partition
- OpenVMS implementation
  - A PC-style Master Boot Record overlays the ODS-2 “boot block”
  - The MBR contains a pointer to an ODS-2 container file which acts as the FAT32 partition
  - Our “OS loader” loads IPB
  - VMS_LOADER and IPB do on the Itanium® architecture what the Alpha console and APB do on Alpha in preparing the system for SYSBOOT

ODS-2 disk



PAL code ?

The Pal code is used to make different implementations the same appearance to the O/S

Alpha PAL code belongs to "firmware"

Alpha: **Privileged Architecture Library**
CHMK
REMQHI
Alignment traps …
Interrupt assistance
Context switching

Itanium: Processor Abstraction Layer
…

Privileged Architecture Library (PALcode)

- Alpha PALcode execution environment
  - Complete control of machine state
  - Interrupts disabled
  - I-stream mapping disabled
- A CALL_PAL is very expensive
- Not all functions need such complete control
- Instructions
  - Complex sequencing and atomic operation
  - VAX interlocked instructions
  - Privileged instructions
- Translation buffer management
- Interrupt and exception setup and dispatching
- Synchronization primitives

Remove from head of queue, interlocked

- **VAX**: microcoded instruction REMQHI
- **Alpha**: CALL_PAL REMQHIL
- **Itanium® Architecture**: *OpenVMS* system service SYS$PAL_REMQHIL

CALL_PAL Replacement: Progress Report

- Created macros in libraries and header files to redefine existing builtins to generate either in-line code or calls to new SYS$PAL_xxx services.
- No source code changes for standard function Alpha CALL_PALs.
- C: __PAL_xxx; BLISS: PAL_xxx; MACRO-32: EVAX_xxx
- The new services will execute the CALL_PAL on Alpha.
  - Existing source unchanged: ipl = __PAL_MFPR_IPL();
  - New common source: ipl = SYS$PAL_MFPR_IPL();
  - VAX queue instruction replacement code is complete
  - If no builtin on Alpha then no builtin on Itanium® architecture (e.g. CSERVE)
  - A few CALL_PALs have no Itanium® architecture function (e.g. MTPR_PERFMON)
  - SWPCTX - must rewrite source code

Four Processor Modes

- High - 0 - kernel
- 1 - exec
- 2 - supervisor
- Low 3 - user

- Identical to VAX & Alpha

IPL / ASTs / software interrupts

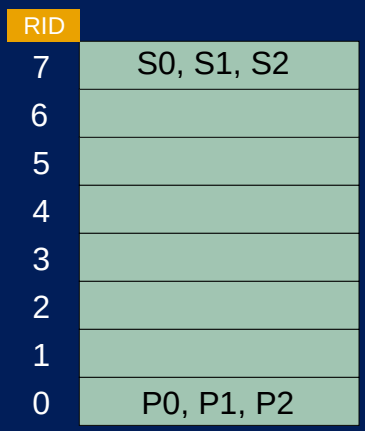
- 0 - 31 IPLs, but we only define 14 of them
  - (2,3,4,5,6,7,8,9,10,11,15,21,22,31)
  - Map 16-31 directly onto a 16-bit interrupt register
  - Levels 0-15 are generated by software
- OpenVMS
  - Controls IPL and mode changes
  - delivers ASTs, software interrupts, exception notification
- Alpha (VAX?) “registers” (e.g. ASTSR, IPL, …) become
  - Itanium® processor registers
  - CPU database cells
  - HWPCB cells

software interrupt services: progress report

- began debugging in LINUX Test Harness on Itanium®-based systems (compiled and linked on LINUX)
- now debugging directly in “boot” environment (compiled and linked on OpenVMS)
- written in C and assembler - will easily move to OpenVMS execlets
- context switching
- register save / restore
- mode change
- system service entry / exit (using epc instruction)
- interrupt handling
- exception notification with complete exception frame
- AST delivery (rewritten in C)
- related CALL_PAL replacements (RD_PS, MTPR_SIRR, MFPR_ASTSR,..)

virtual address space

- address space is 8TB in size (initially)
- 32-bit System Page Table (SPT) window will still be created in S1 space for 32-bit device driver code
- each Itanium® architecture region will have its own page table space
- P0, P1, S0, S1 will be 32-bit; P2 and S2 will be 64-bit



memory management: progress report

- page size
- page protection
- virtual address space
- PTE format
- system pages for EPC instruction created and verified
- replace ASNs with RIDs
- fault handlers
- alignment fault fixups
- TLB miss handler
- related CALL_PAL replacements (PROBER, PROBEW, MTPR_TBIxx)
- rewrite ACCVIO handler in C

synchronization techniques

- requirement: to read/write a shared location in a single atomic operation
- example OpenVMS Uses:
  - Spinlock
  - MUTEX
  - Semaphore
  - Queue instructions
- Alpha: CALL_PAL, LDxL / STxC and MB
- Itanium® architecture: FETCHADDx, CMPXCHGx, XCHGx, MF, and acquire/release semantics on loads and stores
- compilers have builtins, e.g. C: __CMP_SWAP_QUAD
  BLISS: ADD_ATOMIC_LONG

Process Context Switching

- More registers - 128 general, 128 floating... but
  - 2 FEN bits distinguish 32 registers vs. up to 128 registers in use
  - Considering “lazy restore” of FP registers
  - Register stack engine knows the general registers to save
- 2 Stacks
  - Memory stack - move a pointer
  - RSE backing store
    - Fill/spill happens in the background

replacing alpha-specific instructions

- C “asm” uses rewritten in C (not assembler)
  - memory barrier
  - LDxL / STxC sequences
  - get caller’s PC (Alpha code knows about R26)
  - no PAL builtin
- MACRO-32: EVAX_LDxL, EVAX_STxC automatically handled by IMACRO compiler!

other progress

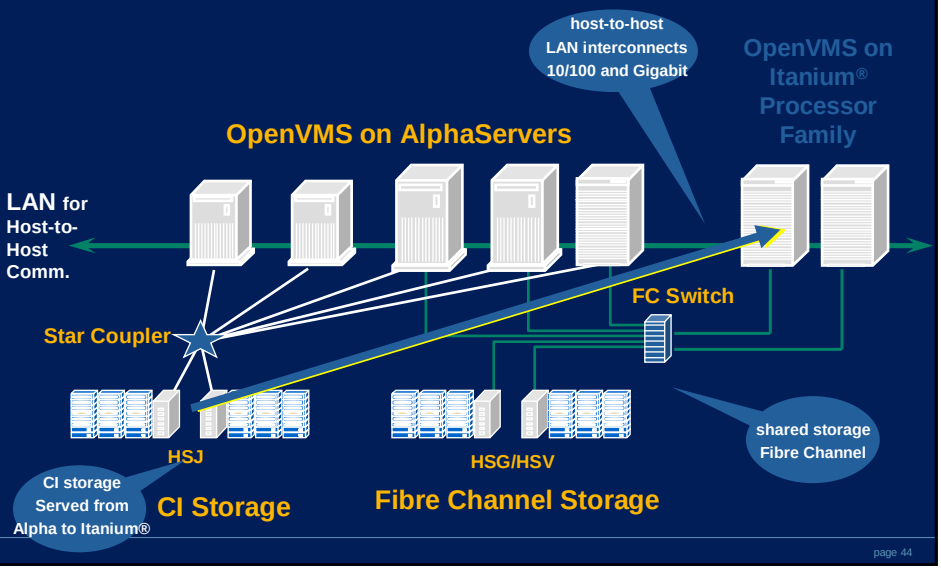
- instruction decoder is complete - used by SDA, DEBUG, fault handlers
- BUGCHECK.M64 rewritten in C
- XDELTA now in use
  - there is ^P (handled by OpenVMS on Itanium®-based systems)
  - breakpoint on the instruction not the bundle
  - single stepping “unexecuted” code will be interesting
- LIB$IPF_CALLING_STANDARD et.al. in progress
- DEBUG knows about DWARF symbol tables
- running images linked with the OpenVMS LINKER that contain assembler, C, BLISS, and MACRO-32.
- Much of the unchanged code now being compiled

And……
What about
Clustering ?

OpenVMS
mixed architecture clusters

- Clustering is a software architecture
  - underlying chip is easy to deal with
- Will support mixed OpenVMS Alpha and OpenVMS Itanium®-based clusters in a phased roll out
- Do you need VAXes in the same cluster?

Itanium® processor family and *OpenVMS* Clusters



How do Cluster Interconnects support Itanium®-based systems?

- **Only "newer technology" adapters moving over**
  - Gigabit and 10/100 Ethernet adapters
  - Fibre Channel and Ultra3 SCSI adapters
  - ATM and FDDI moving at a later time
  - No CI, MC, DSSI, or multi-host SCSI
- **New Interconnect Technology only on Itanium®-based systems**
  - Infiniband assumed to replace MC on Itanium®
- **Revisions of LAN and FC/SCSI adapters will be supported on *both* Alpha and Itanium®**

WHAT did you say ? Or The nomenclature game

Bytes	Intel®	Alpha
1	***byte***	***byte***
2	halfword	***word***
4	***word***	longword
8	doubleword	***quadword***
16	***quadword***	octaword

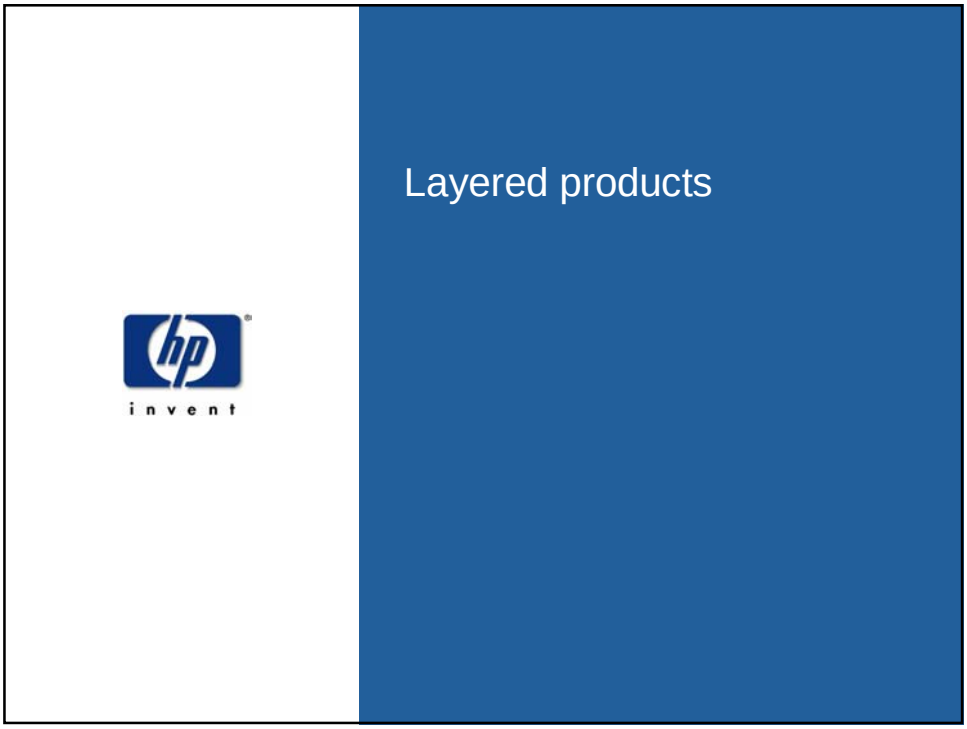
Change of Name

OpenVMS on Itanium(r)

Will be called

„**hp OpenVMS Industry Standard 64**“
(Official Name)
or
„**OpenVMS I64**“
(Informal Name)

Layered products



Development Schedule

H1 2003 - Release 1 V8.0 : selected ISVs, SW partners, early adopters; contains limited layered products

H2 2003 - Release 2 V8.1 : more selected ISVs, SW partners, early adopters; contains more layered products

H1 2004 - Release 3 V8.2 : production quality, general customer release

Layered Product schedules
http://www.compaq.com/hps/ipfenterprise/openvms_move.html

hp OpenVMS on Itanium® Software Product Porting Rollout

1st EAK Release - OpenVMS on Itanium Version 8.0 "Mako" - H1'03

***OpenVMS* Core:** OpenVMS Itanium Operating System, Monitor Utility, RMS Journaling

Networks: DECnet Phase IV, TCP/IP

Development Tools: LSE, CMS, MMS, DTM, TPU

Cross Compilers: C, C++, BLISS, FORTRAN, IMACRO cross compilers, CRTL

hp OpenVMS on Itanium® Software Product Porting Rollout

2nd EAK Release - OpenVMS on Itanium Version 8.1 "Jaws" – H2'03

***OpenVMS* Core:** OpenVMS Itanium Operating System, Limited Cluster Functionality (4-8 Nodes), DECwindows Motif, Monitor Utility, RMS Journaling, Basic Security Features Except Security Server and OpenVMS ACME

Networks: DECnet Phase IV, DECnet Plus, TCP/IP, Advanced Server, DFS

Development Tools: LSE, CMS, MMS, DTM

e-business: XML, Compaq Secure Web Server (Apache), Compaq Secure Web Browser (Mozilla), NetBeans, RTR, Fast Virtual Machine for Java

Compilers: Java, C, C++, BLISS, FORTRAN, IMACRO (MACRO32 & AMACRO port), CRTL, Pascal, BASIC, COBOL

Security: CDSA, Kerberos, OpenSSL

Other: SRI Binary Translator

hp OpenVMS on Itanium® Software Product Porting Rollout

Production Release: OpenVMS on Itanium Version 8.2 "Topaz" - H1'04

***OpenVMS* Core:** OpenVMS Itanium Operating System, Full Cluster Functionality (8-16 node configurations with more configurations added over time), Volume Shadowing, DECwindows Motif, Monitor Utility, RMS Journaling, Basic security features including Security Server and OpenVMS ACME

Networks: DECnet Phase IV, DECnet Plus, TCP/IP, Advanced Server, DFS, X.25 OpenVMS

Development Tools: LSE, CMS, MMS, DTM, PCA, GNAT Ada

e-business: XML, Compaq Secure Web Server (Apache), Compaq Secure Web Browser (Mozilla), NetBeans, RTR, Fast Virtual Machine for Java

Middleware: ACMS, DECforms, DECforms Web Connector, TP Web Connector, TP Desktop Connector, COM for OpenVMS, FMS

Compilers: Java, C, C++, BLISS, FORTRAN, IMACRO (MACRO32 & AMACRO port), CRTL, Pascal, BASIC, COBOL

Security: CDSA, Kerberos, OpenSSL, ACME Login, Encryption for OpenVMS

System Management: Availability Manager, Web Agents, The Data Collector for BMC SW (TDC), ECP Tools, GCU/GCM, OpenVMS Management Station

Storage Products: ABS, SLS, SW RAID, DFO, HSM, Media Robot Utility, Saveset Manager, MDMS, DCSC, RDF for Archive Backup System & Storage Library System

Other: SRI Binary Translator

hp OpenVMS on Itanium® Software Product Porting Rollout

OpenVMS on Itanium – Layered Product Release Only - H2'04

***OpenVMS* Core:** OpenVMS Itanium Operating System, Expanded Cluster Functionality (Qualific -96 nodes, no restrictions), Volume Shadowing, DECwindows Motif, Monitor Utility, RMS Journaling, Basic security features including Security Server and OpenVMS ACME, DECram, Multi-media Management Services

Networks: DECnet Phase IV, DECnet Plus, TCP/IP, Advanced Server, DFS, X.25

Development Tools: LSE, CMS, MMS, DTM, PCA, GNAT Ada, GKS, Phigs, OMNI API/MMS, SCA

e-business: XML, Compaq Secure Web Server (Apache), Compaq Secure Web Browser (Mozilla), NetBeans, RTR, Fast Virtual Machine for Java, Enterprise Directory, BridgeWorks

Middleware: ACMS, DECforms, DECforms Web Connector, TP Web Connector, TP Desktop Connector, COM for OpenVMS, FMS, DCE, Datatrieve

Compilers: Java, C, C++, BLISS, FORTRAN, IMACRO (MACRO32 & AMACRO port), CRTL, Pascal, BASIC, COBOL

Security: CDSA, Kerberos, OpenSSL, ACME Login, Encryption for OpenVMS

System Management: Availability Manager, Web Agents, The Data Collector for BMC SW (TDC), ECP Tools, GCU/GCM, OpenVMS Management Station

Storage Products: ABS, SLS, SW RAID, DFO, HSM, Media Robot Utility, Saveset Manager, MDMS, DCSC, RDF for Archive Backup System & Storage Library System

Mail and Messaging: MAILbus 400, IMAP4 Server, X.500 Admin Alpha, LDAP API, DEC/EDI (Electronic Data Interchange)

Other: SRI Binary Translator

Services Tools: WEBES

Itanium® Compiler Plans (1 of 3)

- C
  - CPQ C
    - Itanium® architecture implementations of OpenVMS CPQ C V6.5 compiler
    - GEM backend
    - Use for recompile/relink/requalify
    - Available with V8.0 (cross)
  - Intel Based C (= C dialect support in C++)
    - Intel® Electron backend
    - Will include some features from CPQ C
    - Compiler for moving forward
    - Available in future release (8.2)

Itanium® Compiler Plans (2 of 3)

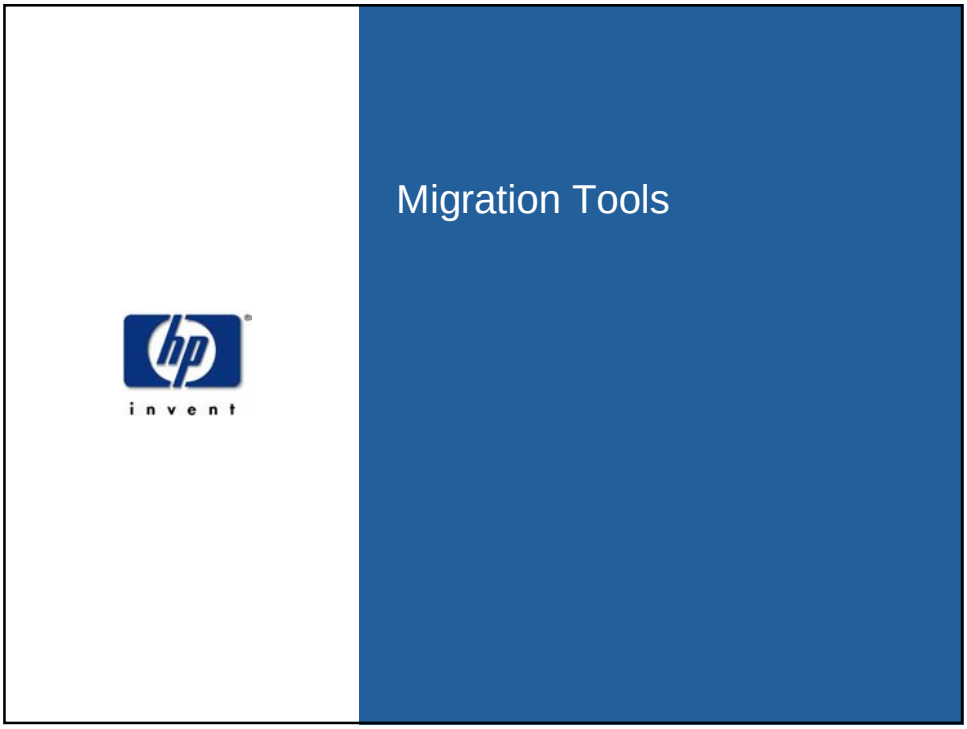
- C++
  - Based on the same front end compiler technology as Compaq C++
  - Use for recompile/relink/requalify
  - Intel® backend code generator
  - Intel C/C++ compiler
- COBOL, BASIC, PASCAL, BLISS
  - Itanium® architecture implementations of the current OpenVMS compilers
  - GEM backend code generator

Itanium® Compiler Plans (3 of 3)

- Java
  - Itanium® architecture implementation of J2SE V1.4.2
- Fortran
  - Itanium® architecture implementation of current OpenVMS Fortran compiler
  - Intel® Electron code generator back end
- IMACRO
  - Compiles ported VAX Macro-32 code for Itanium® architecture
  - Itanium® architecture equivalent of AMACRO
- ADA
  - We will provide an Ada-95 compiler
  - We will not port the existing Ada-83 compiler

Development Tools

- DECset development tools CMS, MMS, LSE, TPU & DTM are scheduled to be available with the H1 2003 release
  - PCA is scheduled to be available with the H1 2004 release
- OpenVMS Debugger
- HP Services will be available to assist



OpenVMS Migration Software for Vax to Alpha V1.2 formerly: DECmigrate (Vax to Alpha)

- Translate images for which source code is not available
  - VAX Environment Software Translator (VEST) translates VAX binary image file into a native Alpha image
  - Translated images run under the Translated Image Environment (TIE) on Alpha
  - Alpha images contain native Alpha instructions
- Updated release V1.2 available since June '02
- Released by HP, supported by SRI

VEST – Current Restrictions

- Will translate valid VAX/VMS image
  - Image(s) must be linked on OpenVMS
- Restrictions:
  - Version restriction removed by V1.2
  - Only user- mode apps
  - Non privileged instructions
  - No self-modifying code
  - No sys. Memory space reference
  - No user-written system services
  - No drivers
  - No applications written in PL/1 due to lack of PL/I RTL
  - No Ada applicatons

"*HPQmigrate*" ☺ - Alpha to Itanium®

- Will translate Alpha OpenVMS binary images and libraries linked under all OpenVMS versions from 6.2 to current version
- Will translate a VESTed image that was translated by DECmigrate from a VAX binary image
- We're looking for feedback on what languages need to be supported
- Restrictions: Alpha binary code
  - Only user-mode apps
  - Non privileged instruction
  - No self-modifying code
  - No sys. Memory space reference
  - No user-written system services
  - No applications written in PL/1due to lack of PL/1 Runtime Library
  - No Ada applications

Binary Translation

- Input: VAX/VMS Image
- Output: Alpha/VMS Image
- OR:
- Input: Alpha/VMS Image or translated VAX/VMS Image
- Output: Itanium®/VMS Image

Applications

Compatibility is the Rule,

Not the Exception

Native Compilers

- Ada (3rd Party)
- Basic
- C
- C++
- COBOL
- FORTRAN
- Pascal
- IMACRO
- Bliss
- Java
- Itanium® Assembler

Development Environment

- OpenVMS development environment
  - ANALYZE command
  - Compilers
  - Debugger
  - DECset (CMS, MMS, etc.)
  - Librarian
  - Linker
  - Message utility
  - etc.

Operations Environment

- DCL is DCL is DCL is ….
- System Management Interface (SYSMAN, SYSGEN)
- Command procedures for VAX continue to work on Alpha and Itanium®

User and Environment

- OpenVMS is OpenVMS:
  - Accounts, ownership, permissions work the same way
  - Device and batch queues work the same way
  - DCL works the same way
  - User procedures “just work”

Compatibility is the Rule,

Not the Exception

but

What is different

IPF Registers

- General Registers R0-R127
  - R0-R31 are static registers
  - R32-R127 are stacked registers that are “redefined” at the start of each procedure
- Floating Registers FP0-FP127
- Predicates P0-P63
- Branch Registers B0-B7
- Application Registers
- Control Registers
- Process Status Register (includes slot index within current bundle)
- Instruction Pointer (also software-generated PC)
- Etc etc etc

IPF Instructions

- Instructions are contained in bundles
  - 128 bits per bundle
  - 3 41-bit instruction slots
  - 5-bit template that describes instruction type in each slot:
    - One of Integer, Memory, Branch, Floating, eXtended
    - Total of 112 distinct instruction formats for all types
  - Almost all formats include a qualifying predicate field
    - If the matching predicate register is clear, the instruction is decoded and executed but its results are discarded
    - Predicate register zero is always set

IPF Instructions (cont)

- Instruction style is “(Pn) opcode target(s)=source(s)”
  - Example:

```
(p4)   cmp.eq    p7,p12 = r37, r52
(p7)   br        label1
(p12)  br        label2
```

  - First instruction only:
    - P4 controls whether or not the results are kept or discarded
    - the result registers are predicate registers P7 and P12
    - R37 is compared for equality with R52
      - If equal: P7 is set to 1 and P12 is set to 0.
      - If not equal: P7 is set to 0 and P12 is set to 1.
  - Combination of three instructions show how an if-then-else might be coded.

Applications

- The Data
  - Core data formats are identical
  - Data conversion routines are available
  - VAX, Alpha and Itanium® prefer natural alignment
  - VAX compilers default to byte alignment
  - Alpha compilers default to natural alignment
  - Itanium® compilers default to natural alignment

Alpha – Itanium® Comparison

Alpha	Itanium®
64-bit addresses	64-bit addresses
64-bit processing	64-bit processing
Instructions:simple	Instructions:simple, bundled (3)
All same length (4 bytes))	All same length (128bit)
Load/store memory access	Load / store memory access
Severe penalty for unaligned data	Severe penalty for unaligned data
Many registers (32 –I, 32-FP)	Huge number of registers (128 GPR, 128 FPR,...)
Out of order instruction completion	Instructions completed in order issued
Deep pipelines and branch prediction	Deep pipelines
Large page size (varies from 8KB to 64KB)	Large page size (8k bytes, variable)

Itanium® vs Alpha Datatypes

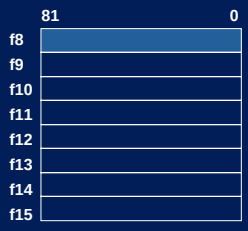
Itanium® Data Types	Alpha Data Types
byte	byte
halfword	word
word	longword
doubleword	quadword
quadword	(octaword)
--	F_floating
--	D_floating (53-bit precision)
--	G_floating
X_floating	X_floating
S_floating (IEEE)	S_floating (IEEE)
T_floating (IEEE)	T_floating (IEEE)
82bit floating (in reg)	--
--	Absolute longword queue
--	Absolute quadword queue
--	Self-relative longword queue
--	Self-relative quadword queue
--	--
--	--
--	--
--	--

Applications

- ***Incompatibilities, Differences***
- Memory Page Size
  - VAX: Fixed Page Size
    - 0.5KB
    - 512 bytes
  - Alpha: Implementation specific
    - 8KB to 64KB
    - 8192 to 65536 bytes
  - Itanium®: Implementation specific
    - 8KB now
    - others later

Function return values are NOT in register zero

64 bit integer return values are in r8
128 bit return values in r8, r9
Up to 256 bits structure values in r8 – r11



Floating point single, double, double extended values are returned in f8
Quad precision values in f8, f9
Structures can use up to f15

Register Mapping

0 = 8
1 = 9
2 = 28
3 = 3
4 = 4
5 = 5
6 = 6
7 = 7
8 = 26
9 = 27
10 = 10
11 = 11
12 = 30
13 = 31
14 = 20
15 = 21

16 = 14
17 = 15
18 = 16
19 = 17
20 =18
21 = 19
22 = 22
23 = 23
24 = 24
25 = 25
(RA) 26 = no mapping
(LP) 27 = no mapping
28 = no mapping
(FP) 29 = 29
(SP) 30 = 12 (SP)
(RAZ) 31 = 0 (RAZ)

So, what's different? (1 of 3)

- Calling Standard
  - publicly available today at http://www.compaq.com/hps/ipf-enterprise/openvms.html
  - Intel® calling standard with OpenVMS modifications
    - No frame pointer (FP)
    - Multiple stacks
    - only 4 preserved registers across calls
  - All OpenVMS provided tools will "know" about these changes
- Your code that "knows" about the Alpha standard will almost certainly need to change

So, what's different? (2 of 3)

- Object file format
  - ELF/DWARF industry standards plus our extensions
    - ELF - Executable and Linkable Format, Itanium® Architecture object code, images, etc.
    - DWARF - Debugging and traceback information (embedded in ELF).
  - All OpenVMS provided tools will "know" about these changes
  - User written code that "knows" the object file format may have to change
  - We will be publishing these specifications in the near future
  - Image header "tricks" may no longer work (Flip a bit to turn on/off debugging)

So, what's different? (3 of 3)

- Floating point data types
  - Itanium® architecture supports IEEE float only
  - All compilers that currently support F, D, G, S, T, and X (S and T are native IEEE formats) will continue to do so on Itanium® architecture
  - IEEE will be the default (still working the details)
  - White Paper with technical details available at http://www.compaq.com/hps/ipf-enterprise/openvms.html

Private “Threading”

- What is it?
  - Application that privately manages execution streams
  - Written in Alpha assembler
  - Knows details of Calling Standard and “context”
- OpenVMS will provide a set of routines
  - use any language
  - available from any mode
  - can test on Alpha
  - XQP and RMS have been converted
- Information available in “Writing OpenVMS Alpha Device Drivers in C”, Sherlock and Szubowicz, Digital Press, 1996
  - See sections on Kernel Process

Calling Standard
what’s the problem?

an example:

- on VAX and Alpha
  - R0 = function return value
- on Itanium® architecture
  - R0 = zero (RAZ)

Calling Standard
the present

- Intel® defined Itanium® Calling Conventions
- comparable to Alpha calling standard, but lacking
  - argument count / information
  - VAX/Alpha floating point datatypes
  - support for translated images
  - definition of invocation context handle

Calling Standard
our approach

- adopt the mainstream Itanium® standards
  - Intel® calling standard
  - ELF object / image file format
  - DWARF debug information format
- extend / specialize as needed to support existing VMS features
- Intel® has accepted our extensions

Calling Standard
the future

- OpenVMS Itanium® Calling Standard
- based on industry standard Itanium Calling Conventions
- extended for OpenVMS
  - argument count / information register
  - VAX/Alpha floating point formats
  - translated image support
  - additional definitions

Calling Standard Presentation

- Please see John Covert‘s Presentation

differences between Alpha and Itanium® Architectures

- different registers for arguments and return value
- rotating registers and separate register stack
- GP (global data pointer) register
- different sets of scratch and saved registers
- PC range based condition handling

What code will I need to change?

- Architecture Specific code
  - All Alpha assembler code must be rewritten
- Conditionalized code
  - Build command files

```
$ if .not. Alpha  ! Assumes VAX
```

  - Application source code

```
#ifndef (alpha)  // Assumes VAX
C asm code
```

- We will be providing a new Porting Guide with details

Architecture specific build procedures

- $! Determine architecture type
- $ type_symbol = f$getsyi("arch_type")
- $ if type_symbol .eq. 1 then goto ON_VAX
- $ if type_symbol .eq. 2 then goto ON_ALPHA
- $ ON_VAX:
- $!Do VAX-specific processing
- $ exit
- $ ON_ALPHA:
- $!Do Alpha-specific processing
- $ exit

Preparatory Steps

- Inventory your OpenVMS system
  - Languages and applications in use
  - Required 3rd party products
  - Special hardware or drivers
  - Application source code
- Examine your code for known differences and architecture dependencies
- Upgrade application source code to current standards
  - VAX C to DEC C
  - Pthreads rather than DCE threads, CMA

Deployment Process

- Build files and process
- Test scripts and process
- Deployment strategy
  - Cluster cut over / Fall back strategy
  - Separate development and production systems

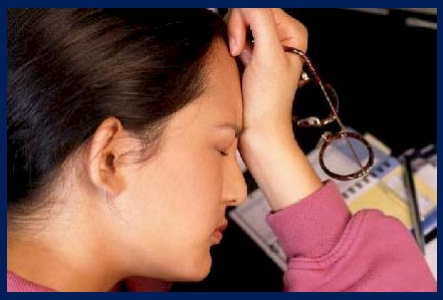
For further Information about OpenVMS on Itanium

- OpenVMS on the Itanium® Architecture Web Sites
  - General OpenVMS on Itanium information http://h18003.www1.hp.com/hps/ipf-enterprise/openvms.html
  - Layered products schedules http://h18003.www1.hp.com/hps/ipf-enterprise/openvms_move.html
  - Layered products plans (products that either will not be ported or are under review) http://h18003.www1.hp.com/hps/ipf-enterprise/ovms_plans.html
  - OpenVMS Partner plans http://h18003.www1.hp.com/hps/ipf-enterprise/partner_quotes.html

For more information

- Alpha and Itanium® information
- www.hp.com/products/alpha-itanium
- Itanium® architecture: http://developer.intel.com/design/itanium/archSysSoftware/
- Itanium® family processor hardware: http://developer.intel.com/design/itanium/manuals.htm
- Software manuals: http://developer.intel.com/design/itanium/arch_spec.htm

Questions ??????????




invent
invent
page 97