The IA-64 System Architecture: Tutorial For Hardware, OS, & Application Developers

February 15, 2000

Jerry Huck
Manager of Processor Architecture
Computer Products Group
Hewlett Packard Company

Rumi Zahir
Itanium™ Processor System Architecture
Microprocessor Products Group
Intel Corporation
What’s In This Session?

- IA-64 memory models, system software, other architecture features relevant to hardware, OS, and application developers

- IA-64 concepts & features for developers to take advantage and build robust & scalable software and hardware components
IA-64 System Architecture Agenda

- System Architecture Highlights
- Virtual Memory Model
- Interruption Model
- System Software Stack
- Reliability, Availability, Serviceability
- Parallelism & Scalability
- Compatibility
- Summary
IA-64 System Architecture Goals

- **Flexible Architecture**
  - “address space per process” model
    - Windows NT, Unix, Mach, etc.
  - “Global Address Space” model
    - HP-UX plus future 64-bit OSes

- **Performance focused Hardware**

- **Support shrink-wrap OS compatibility**

- **Scalable system performance**

- **IA-32 and PA-RISC Compatibility**
IA-64 System Architecture Agenda

- System Architecture Highlights
- Virtual Memory Model
- Interruption Model
- System Software Stack
- Reliability, Availability, Serviceability
- Parallelism & Scalability
- Compatibility
- Summary / Call To Action
IA-64 Virtual Memory Model

- Process Address Space
- System Address Space Management
- Virtual Address Translation
  - TLB and Page table
- Flexible Object Sharing Model
  - Aliasing and Global addressing
64-bit Address Space: Large and Sparse

- Requires increased TLB capacities
  - Mapped files, globally shared memory
  - Allow for multiple large on/off-chip TLBs
  - Provide wide range of page sizes

- Improve utilization and sharing of TLB and page tables
  - No TLB flush on context switch
  - Promote TLB entry sharing

IA-64 Overcomes Challenges Of 64-bit Address Space Efficiently
Process Address Space

Flat Virtual Space: $2^{64}$ bytes
Process Address Space

- 64-bit Address
- 8 Regions/process
- 2^64
- 0
- OS Kernel
- DLLs
- Data/Heap
- Code/Text
System Address Space

64-bit Address

$2^{64}$

$0$ $63$

$≥2^{18}$ Regions

Pages
IA-64 Region Registers

64-bit Address

63  61  60

0

≥ 2^{18} Regions

2^{61} bytes in size

8 Region Registers
Processes and Threads

Regions Enable Efficient Management Of Processes For Multi-tasking Environments
Virtual Address Translation: TLB

- Mapping to Physical Address

Virtual Addresses

- Process A
- Process B
- Process C

Access Rights

TLB

Physical Addresses
TLB Organization

- **Separate instruction and data TLBs**
- **Software Manages**
  - TR entries,
  - Page-table updates
- **Hardware Manages**
  - TC TLB refill
  - Broadcast TLB Purge

**Balance TLB For Efficient Memory Management**
Virtual Address Translation

Region Registers

64-bit Address

RRx Virtual Page # offset

63 61 60 0

TLB

“match”

“deliver”

Physical Address

Physical Page # Protection
Protection: Can I See it? Can I Access it?

Protection Keys Increase TLB Utilization For Large Object Databases
Variable Page Sizes

- Minimum on all implementations
  - 4K, 8K, 16K, 64K, 256K, 1M, 4M, 16M, 64M, 256M-bytes
- 4 GB purge
  - Simplify address space de-allocation

Variable Page Sizes Enable TLB Efficiency For OS And Application Performance
Hardware Accessed Page Table

Flexible Hardware Mechanisms
Enable Parallel Execution
Virtual Memory Model: Example

Region 7 - One RID, no key
Kernel - protected by Priv. level

Region 2 - One RID, protection via multiple keys
Shared memory areas

Region 1 - Same RID if shared
Single address space for code

Region 0 - Different RID in each process
Unique address spaces for data

Flexible Virtual Memory Architecture Enables Variety Of Efficient OS Implementations
IA-64 System Architecture Agenda

- System Architecture Highlights
- Virtual Memory Model
- Interruption Model
- System Software Stack
- Reliability, Availability, Serviceability
- Parallelism & Scalability
- Compatibility
- Summary / Call To Action
IA-64 Interruption Model

- Parallel instruction execution, . . .
  - Exception delivery is sequential & precise
  - All exceptions reported on the excepting instruction (including numeric exceptions)

- “Interruption” is IA-64 term for...

<table>
<thead>
<tr>
<th>Abort</th>
<th>Fault</th>
<th>Trap</th>
<th>Interrupt</th>
</tr>
</thead>
</table>
| • Hardware reset  
  • Machine check | Exception taken before instruction commit, e.g. TLB miss | Exception taken after instruction commit, e.g. FP trap | Asynchronous external event:  
  • device or platform management interrupt  
  • soft-reset |

IA-64 Provides Precise Exception Model To Match Today’s OS Designs
IA-64 Interruption Process

Application Code

- Instruction A executed at 0x1000
- Instruction B executed at 0x1010
- Instruction C executed at 0x1020

Normal Instruction Execution Flow:
- Instruction A executed

IVT Code

- Instruction X executed at 0x4000
- Instruction Y executed at 0x4010
- RFI at 0x4020

Current Processor State

- IP: 0x1000
- PSR

Current Processor State

- IIP
- IPSF

Interruption Registers

BANK1 REG (app data)

BANK0 REG (OS data)
The IA-64 Interruption Process involves the following steps:

1. **Application Code**
   - Instruction Sequence: 0x1000 INST A, 0x1010 INST B, 0x1020 INST C, etc.

2. **IVT Code**
   - Interrupt Vector Table: 0x4000 INST X, 0x4010 INST Y, 0x4020 RFI, etc.

3. **Current Processor State**
   - IP: 0x1010
   - PSR

4. **Normal Instruction Execution Flow**
   - Instruction B executed

The figure illustrates the process flow with boxes representing different registers and memory banks for application and operating system data.
IA-64 Interruption Process

1. Processor switches to Bank 0 registers preparing to run IVT code

2. Processor saves current state to interruption registers before interrupt handling

Application Code

0x1000 INST A
0x1010 INST B
0x1020 INST C

IVT Code

0x4000 INST X
0x4010 INST Y
0x4020 RFI

Current Processor State

IP 0x1010
PSR

IPR

Interruption Registers

IIP 0x1010
IPSR
Instruction X executed in interrupt vector table

Application Code

0x1000  INST A
0x1010  INST B
0x1020  INST C
...

Interrupt Vector Table (IVT) Code

IP  0x4000  INST X
    0x4010  INST Y
    0x4020  RFI
    ...

Current Processor State

IP  0x4000
PSR
...

Interruption Registers

IIP  0x1010
IPSR
...

BANK0  REG (OS data)

BANK1  REG (app data)
IA-64 Interruption Process

Application Code

0x1000 INST A
0x1010 INST B
0x1020 INST C
...

IVT Code

0x4000 INST X
0x4010 INST Y
0x4020 RFI
...

Current Processor State

IP 0x4010
PSR
...

Interruption Registers

IIP 0x1010
IPSR
...

Interruption Handling

• Instruction Y executed in interrupt vector table
IA-64 Interruption Process

Application Code

0x1000 INST A
0x1010 INST B
0x1020 INST C
...

IP

IVT Code

0x4000 INST X
0x4010 INST Y
0x4020 RFI
...

IP

Current Processor State

IP 0x4020
PSR

Processor switches back to Bank 1 registers

1

BANK SWITCHING

BANK1 REG (app data)

0x32 127

0x24 16

0x31

0x1010

RESTORING PRE-INTERRUPTION STATE

Processor restores state from interruption registers before returning from interrupt

2
IA-64 Interruption Process

Application Code

0x1000  INST A
0x1010  INST B
0x1020  INST C
...

IVT Code

0x4000  INST X
0x4010  INST Y
0x4020  RFI
...

Current Processor State

IP  0x1010
PSR

Interrupt Registers

IIP
IPSR

Resume Normal Instruction Execution:
- Instruction B executed

Instruction B executed

Application Code

0x1000    INST A
0x1010    INST B
0x1020    INST C
...

IVT Code

0x4000    INST X
0x4010    INST Y
0x4020    RFI
...

Current Processor State

IP  0x1010
PSR

Interrupt Registers

IIP
IPSR

0x10100x1010

Instruction B executed

Application Code

0x1000    INST A
0x10100x1010
0x1020    INST C
...

IVT Code

0x4000    INST X
0x4010    INST Y
0x4020    RFI
...

Current Processor State

IP  0x1010
PSR

Interrupt Registers

IIP
IPSR

0x10100x1010
Interruption Features

- Low interruption latency
  - Interruption delivery causes single pipeline break
  - Key state captured in on-chip registers
- State-save controlled by system software
  - Software makes performance/nesting trade-off
  - Shared mechanism for IA-64/IA-32 interruptions
- Efficient handler execution
  - Interruption vector table (IVT) contains code for interrupt service routine

Provides Fast And Flexible Interruptions For Large I/O Intensive Applications
## Parallelism Across System Calls

- **Application Code**
  ```
  ... // make system call
  br.call _write
  ...
  ```

- **EPC Page (PL promote and execute only)**
  ```
  _write: epc // privilege promote
  // without pipeline flush
  br os_write
  ```

- **Operating System Kernel (privileged code)**
  ```
  os_write:
  ...
  // perform system call
  br.ret // demote PL and return to user
  ```

---

**Fast System Calls Improve Synergy Between OS & Application**
IA-64 External Interrupts

High Performance Message-Based Interrupts Compatible With Today’s Platforms
IA-64 System Architecture Agenda

- System Architecture Highlights
- Virtual Memory Model
- Interruption Model
- System Software Stack
- Reliability, Availability, Serviceability
- Parallelism & Scalability
- Compatibility
- Summary / Call To Action
IA-64 System Software Stack: OS Boot

 Operating System Software

 EFI

 System Abstraction Layer (SAL)  IA-32 BIOS

 Processor Abstraction Layer (PAL)

 Processor (hardware)

 Platform (hardware)

 OS boot

 Access to platform resources

 Reset, machine checks

 IA-32 BIOS
OS Running

Operating System Software

EFI

System Abstraction Layer (SAL) (IA-32 BIOS)

Processor Abstraction Layer (PAL)

Processor (hardware)

Platform (hardware)

External Interrupts (performance critical)

Instructions

Interruptions

I/O

I/O
OS Calls To Firmware Services

Operating System Software

EFI

System Abstraction Layer (SAL) IA-32 BIOS

Access to platform resources

Processor Abstraction Layer (PAL)

Processor (hardware)

Platform (hardware)
Machine Check Handling

Operating System Software

EFI

System Abstraction Layer (SAL)

Processor Abstraction Layer (PAL)

Processor (hardware)

Platform (hardware)

Access to platform resources

Machine Check Services

Reset, machine checks

IA-32 BIOS

Operating System Software

Machine Check Services

Reset, machine checks
Architected RAS Features

- Reliability
  - 3 levels of error signaling:
    - Continuable, local, and global

- Availability
  - Fine grained error containment by cooperation between hardware and firmware

- Serviceability
  - Extensive error logs for error analysis
  - Common error logs for firmware and OS

Advanced Machine Check Architecture
For High Levels of Reliability, Availability, And Serviceability
IA-64 System Architecture Agenda

- System Architecture Highlights
- Virtual Memory Model
- Interruption Model
- System Software Stack
- Reliability, Availability, Serviceability
- Parallelism & Scalability
- Compatibility
- Summary / Call To Action
Parallelism and Scalability

- Excellent Multi-Processing Scalability
  - High performance relaxed ordering model
  - Flexible semaphore primitives
  - Hardware broadcast TLB purge

- OS Resource Scaling & Parallelism
  - Flexible # of PKRs, TRs, key & RID widths
  - Parallel update of control register writes (explicitly serialized)

IA-64 Features For Scalable MP Systems
Compatibility

- IA-64 supports **PA-RISC & IA-32 Applications**
- IA-64 supports **IA-32 OS**
  - Capable of running unmodified multi-processing IA-32 OS, e.g. NT4.0, Linux
- IA-64 OS supports **IA-32 Platform peripherals**
  - IA-64 support for legacy I/O port space
- Dependent upon OS & platform implementation

**IA-64 Offers Full IA-32 Compatibility In Hardware: Platforms, OS, Applications**
IA-64 System Architecture Agenda

- System Architecture Highlights
- Virtual Memory Model
- Interruption Model
- System Software Stack
- Reliability, Availability, Serviceability
- Parallelism & Scalability
- Compatibility
- Summary / Call To Action
IA-64 System Architecture Summary

**Performance:**
- Innovative architecture provides low overhead for major OS functions

**Flexibility:**
- Accommodates current OS & features for OS extensibility & evolution

**Scalability:**
- IA-64 platforms can run efficiently from UP to very large MP systems

**Availability:**
- High levels of reliability & availability through enhanced RAS features

**Interoperability:**
- Well defined mechanisms & new firmware model ensures running of shrink-wrapped OS on variety of platforms

**Compatibility:**
- Provides efficient and transparent IA-32 compatibility at system level
Call to Action

- New detailed specifications available for download
  - OSVs & IHVs accelerate development of IA-64 system applications (device drivers, system debug tools, etc...)

- OSVs, IHVs, & ISVs take advantage of broad IA-64 enabling efforts to accelerate your porting efforts

- Take advantage of open source of IA-64 tools and Linux operating system
New Public IA-64 Docs

- **IA-64 Software Developer’s Manual**
  - Info for system & application software, & development tools for IA-64
  - Software optimization techniques
  - Performance monitoring info for optimization support

- **More IA-64 Documentation:**
  - IA-64 Software Conventions and Runtime Architecture Guide
  - Assembly Language Reference Guide
  - IA-64 assembler & reference guide
  - IA-64 Processor-specific Application Binary Interface
  - System Abstraction Layer Specification
  - and more ...

**IA-64 Docs Available On Internet:**

www.hp.com/go/ia-64/
developer.intel.com/design/ia-64/devinfo.htm