Understanding the IA-64 Architecture

Gautam Doshi
Senior Architect
IA-64 Processor Division
Intel Corporation

August 31,'99 - September 2,'99
Agenda

- Today’s Architecture Challenges
- IA-64 Architecture Performance Features
- High-end Application Characteristics
- End User Benefits of IA-64 Features
The Context

- Programmer programs in high level lang.
- Compiler compiles program to machine inst.
- Machine executes these instructions

- High Level Lang = Programmer's vocabulary
- Inst. Set Arch = Compiler’s “vocabulary”
  - Architecture determines what the compiler can “express”
  - Architecture determines what the machine must “execute”

Architecture : the compiler’s “vocabulary”
Today’s Architecture Challenges

- Sequential Semantics of the ISA
- Low Instruction Level Parallelism (ILP)
- Unpredictable Branches, Mem dependencies
- Ever Increasing Memory Latency
- Limited Resources (registers, memory addr)
- Procedure call, Loop pipelining Overhead
- ...

Fundamental challenges abound
Today’s Architecture Challenges

**Sequential Semantics**

- Program = Sequence of instructions
- Implied order of instruction execution
- Potential dependence from inst. to inst.

But ...

- High performance needs parallel execution
- Parallel execution needs independent insts.
- Independent insts must be (re)discovered

**Sequentiality inherent in traditional archs**
Today’s Architecture Challenges

**Sequential Semantics ...**

<table>
<thead>
<tr>
<th>Dependent</th>
<th>Independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>add r1 = r2, r3</td>
<td>add r1 = r2, r3</td>
</tr>
<tr>
<td>sub r4 = r1, r2</td>
<td>sub r4 = r11, r2</td>
</tr>
<tr>
<td>shl r5 = r4, r8</td>
<td>shl r5 = r14, r8</td>
</tr>
</tbody>
</table>

- Compiler knows the available parallelism
  - but has no “vocabulary” to express it
- Hardware must (re)discover parallelism

Complex hardware needed to (re)extract ILP
Today’s Architecture Challenges

**Low Inst. Level Parallelism**

- Branches: Frequent, Code blocks: Small
- Limited parallelism within code basic blocks
- Wider machines need **more** parallel insts.
- Need to exploit ILP across branches
- But some instructions can fault!
- Branches are a barrier to code motion

**Limited ILP available within basic blocks**
Today’s Architecture Challenges

Branch Unpredictability

- Branches alter the “sequence” of insts.
- ILP must be extracted across branches
- Branch prediction has its limitations
  - Not perfect, performance penalty when wrong
  - Need to speculatively execute insts that can fault
    - memory operations (loads), floating-point operations, ...
  - Need to defer exceptions on speculative operations
    - more bookkeeping overhead hardware

Branches make extracting ILP difficult
Today’s Architecture Challenges

Memory Dependencies

- Loads usually at the top of a chain of insts.
- ILP extraction requires moving these loads
- Branches abound and are a barrier
- Stores abound and are also a barrier
  - programming paradigm: Pointers can point anywhere!
- Dynamic disambiguation has its limitations
  - limited in its scope, requires additional hardware
  - adds to code size increase, if done in software

Memory dependencies further limit ILP
Today’s Architecture Challenges

**Memory Latency**

- Has been increasing over time
- Need to distance loads from their uses
- Branches and Stores are barriers
- Cache hierarchy has its limitations
  - Typically small, so limited working set
  - Consumes precious silicon area
  - Helps if there is locality. Hinders, otherwise.
  - Managed asynchronously by hardware

Increasing latency exacerbates ILP need
Today’s Architecture Challenges

Resource Constraints

- **Small Register Space**
  - Limits compilers ability to “express” parallelism
  - Creates false dependencies (overcome by renaming)

- **Shared Resources**
  - Condition flags, Control registers, etc.
  - Forces dependencies on otherwise independent insts

- **Floating-Point Resources**
  - Limited performance even in ILP rich applications
  - Data parallel applications need flexible resources

**Limited Resources: a fundamental constraint**
Today’s Architecture Challenges

**Procedure Call Overhead**

- Modular programming increasingly used
  - Programs tend to be call intensive
- Register space is shared by caller and callee
- Call/Returns require register save/restores
- Software convention has its limitations
  - Parameter passing limited
  - Extra saves/restores when not needed

Shared resources create more overhead
Today’s Architecture Challenges

Loop Optimization Overhead

- Loops are a common source of good ILP
- Unrolling/Pipelining exploit this ILP
- Prologue/Epilogue cause code expansion
- Unrolling causes more code expansion
- Limits the applicability of these techniques

Loop ILP extraction costs code size
Today’s Architecture Challenges

- Complex conditionals
  - sequential branch execution increases critical path

- Dynamic resource binding
  - parallel insts need to be reorganized to fit machine capability

- (Lack of) Domain specific support
  - Multimedia: operations repertoire, efficient data-types, …
  - Floating-point: standard compliant, accuracy, speed, …

And the challenges continue ...
Architecture Challenges

- Sequentiality inherent in traditional architectures
- Complex hardware needed to (re)extract ILP
- Limited ILP available within basic blocks
- Branches make extracting ILP difficult
- Memory dependencies further limit ILP
- Increasing latency exacerbates ILP need
- Limited resources: A fundamental constraint
- Shared resources create more overhead
- Loop ILP extraction costs code size
- And the challenges continue...

IA-64 overcomes these challenges!
Agenda

- Today's Architecture Challenges
- IA-64 Architecture Performance Features
- High-end Application Characteristics
- End User Benefits of IA-64 Features
IA-64 Architecture Performance Features

- It’s all about Parallelism!
  - Enabling it
  - Enhancing it
  - Expressing it
  - Exploiting it

... at the proc./thread level for programmer

... at the instruction level for compiler

Enable, Enhance, Express, Exploit - Parallelism
IA-64 Architecture Performance
Features

- Explicitly Parallel Instruction Semantics
- Predication and Control/Data Speculation
- Massive, Massive Resources (regs, mem)
- Register Stack and its Engine (RSE)
- Memory hierarchy management support
- Software Pipelining Support
- ...
Explicitly Parallel Semantics

- Program = Sequence of Parallel Inst. Groups
- Implied order of instruction groups
- NO dependence between insts. within group
- High performance needs parallel execution
- Parallel execution needs independent insts.
- Independent instructions explicitly indicated

Parallelism inherent in IA-64 architecture
Explicitly Parallel Semantics ...

**Dependent**
- `add r1 = r2, r3` ;
- `sub r4 = r1, r2` ;
- `shl r5 = r4, r8`

**Independent**
- `add r1 = r2, r3` 
- `sub r4 = r11, r2` 
- `shl r5 = r14, r8`

- Compiler knows the available parallelism
  - and now HAS the “vocabulary” to express it - STOPs (;;)

- Hardware easily exploits the parallelism

Frees up hardware for parallel execution
Architecture Challenges

- **Sequential Semantics of the ISA**
  - Low Instruction Level Parallelism (ILP)
  - Unpredictable Branches, Mem dependencies
  - Ever Increasing Memory Latency
  - Limited Resources (registers, memory addr)
  - Procedure call, Loop pipelining Overhead
  - ...

IA-64 EPIC ISA : Sequential--, Parallel++
IA-64 Architecture Performance Features

Predication

Traditional Arch

IA-64

• Control flow to Data flow

Predication removes/reduces branches

Intel Labs
IA-64 Architecture Performance Features

**Predication ...**

- **Unpredictable branches removed**
  - Misprediction penalties eliminated

- **Basic block size increases**
  - Compiler has a larger scope to find ILP

- **ILP within the basic block increases**
  - Both “then” and “else” executed in parallel

- **Wider machines are better utilized**

**Predication enables and enhances ILP**
Architecture Challenges

- Sequential Semantics of the ISA
  → Low Instruction Level Parallelism (ILP)
  → Unpredictable Branches, Mem dependencies
- Ever Increasing Memory Latency
- Limited Resources (registers, memory addr)
- Procedure call, Loop pipelining Overhead
- ...

IA-64 Predication: ILP++, Branches--
Branch barrier broken! Memory latency addressed
Control Speculation ...

- Speculative data uses can also be speculated

Control speculating “uses” further increases ILP
Architecture Challenges

- Sequential Semantics of the ISA
  - Low Instruction Level Parallelism (ILP)
- Unpredictable Branches, Mem dependencies
- Ever Increasing Memory Latency
- Limited Resources (registers, memory addr)
- Procedure call, Loop pipelining Overhead
- ...
IA-64 Architecture Performance Features

Data Speculation

Traditional Architectures

```
instr 1
instr 2
st [?] 
```

Barrier

```
ld r1= 
use =r1
```

IA-64

```
ld.a r1= 
instr 1
instr 2
st [?] 
```

Load moved above store by compiler

```
ld.c r1 
use =r1
```

Store barrier broken! Memory latency addressed
IA-64 Architecture Performance Features

**Data Speculation ...**

- Speculative data uses can be speculated

---

**Uses moved above store by compiler**

- IA-64 Architecture Performance Features

  - Uses moved above store by compiler

  - Recovery code

  - Speculative data uses can be speculated

---

**Data speculating “uses” further increases ILP**
Architecture Challenges

- Sequential Semantics of the ISA
- Low Instruction Level Parallelism (ILP)
- Unpredictable Branches, Mem dependencies
- Ever Increasing Memory Latency
- Limited Resources (registers, memory addr)
- Procedure call, Loop pipelining Overhead
- ...

IA-64 Data Speculation: ILP++, Latency impact--
IA-64 Architecture Performance Features

Massive Execution Resources

Integer Registers
- GR0: 63, 0
- GR1: 0
- GR31: 32 Static
- GR32: 96 Stacked, Rotating
- GR127: NaT, 32 Static, 96 Stacked, Rotating

Floating-Point Registers
- FR0: 0.0
- FR1: 1.0
- FR31: 32 Static
- FR32: 96 Rotating
- FR127: 32 Static, 96 Rotating

Branch Registers
- BR0
- BR7

Predicate Registers
- PR0
- PR1
- PR15
- PR16
- PR63
- 16 Static
- 48 Rotating

An abundance of machine resources
Massive Memory Resources

- 18 BILLION Giga Bytes accessible
  - $2^{64} = 18,446,744,073,709,551,616$
- Both 64-bit and 32-bit pointers supported
- Both Little and Big Endian Order supported
Architecture Challenges

- Sequential Semantics of the ISA
- Low Instruction Level Parallelism (ILP)
- Unpredictable Branches, Mem dependencies
- Ever Increasing Memory Latency
- Limited Resources (registers, memory addr)
- Procedure call, Loop pipelining Overhead
- ...

IA-64 Resources: Aid “explicit” parallelism
Register Stack

- GR Stack reduces need for save/restore across calls
- Procedure stack frame of programmable size (0 to 96 regs)
- Mechanism implemented by renaming register addresses

Distinct resources reduce overhead
Frame overlap eases parameter passing
Registe Stack Engine (RSE)

- Automatically saves/restores stack registers without software intervention
  - Provides the illusion of infinite physical registers
    - by mapping to a stack of physical registers in memory
  - Overflow: Alloc needs more registers than available
  - Underflow: Return needs to restore frame saved in memory

- RSE may be designed to utilize unused memory bandwidth to perform register spill and fill operations in the background

RSE eliminates stack management overhead
Architecture Challenges

- Sequential Semantics of the ISA
- Low Instruction Level Parallelism (ILP)
- Unpredictable Branches, Mem dependencies
- Ever Increasing Memory Latency
- Limited Resources (registers, memory addr)
  - Procedure call, Loop pipelining Overhead
- ...

IA-64 Reg. Stack: Modular program support
IA-64 Architecture Performance Features

Software Pipelining Support

- High performance loops without code size overhead
  - No prologue/epilogue
  - Register rotation (rrb)
  - Predication
  - Loop control registers (LC, EC)
  - Loop branches (br.ctop, br.wtop)
- Especially valuable for integer loops with small trip counts

Whole loop computation in parallel

IA-64 Loop support: ILP+++, Overhead---

Details in “Compiler Technology for IA-64”
Architecture Challenges

- Sequential Semantics of the ISA
- Low Instruction Level Parallelism (ILP)
- Unpredictable Branches, Mem dependencies
- Ever Increasing Memory Latency
- Limited Resources (registers, memory addr)

→ Procedure call, Loop pipelining Overhead

...
IA-64 Architecture Performance Features

Floating-Point Architecture

- **Fused Multiply Add Operation**
  - An efficient core computation unit

- **Abundant Register resources**
  - 128 registers (32 static, 96 rotating)

- **High Precision Data computations**
  - 82-bit unified internal format for all data types

- **Software divide/square-root**
  - High throughput achieved via pipelining

**IA-64 FP: High performance and high precision**
Example: Software divide

- 2 dimensional Hydro-dynamics kernel
  - Livermore FORTRAN Kernel #18
- Scaling - a common operation

```fortran
DO 70  k = 2, KN
DO 70  j = 2, JN
ZA(j,k) = (ZP(j-1,k+1) + ZQ(j-1,k+1) - ZP(j-1,k) - ZQ(j-1,k))
  * (ZR(j,k) + ZR(j-1,k)) / (ZM(j-1,k) + ZM(j-1,k+1))
ZB(j,k) = (ZP(j-1,k) + ZQ(j-1,k) - ZP(j,k) - ZQ(j,k))
  * (ZR(j,k) + ZR(j,k-1)) / (ZM(j,k) + ZM(j-1,k))
70 CONTINUE
```

Several independent iterations containing divide
Example: Software divide

\( ZA_i = \frac{N_i}{D_i} \)

Traditional Arch

- Software divide breaks a single divide into several FMA operations
- Slightly greater latency of each divide, but much greater throughput
- Performance scales as machine becomes wider and has more FMA execution units

IA-64

- FMA execution unit

IA-64 Software divide provides much greater throughput on FP loops
IA-64 Architecture Performance Features

- And the performance features continue...

- **Parallel Compares and Multi-way branches**
  - Control height reduction, branch bandwidth, ...

- **Memory Hierarchy Control**
  - Allocation, De-allocation, Flush, Prefetch (Data/Inst.), ...

- **Multimedia Support**
  - Semantically compatible with Intel’s MMX™ technology and Streaming SIMD Extension instruction technology

- **Bit/Byte field instructions**
  - Population count, Extract/Deposit, Leading/Trailing zero bytes, ...
IA-64 Architecture Performance Features

- Parallelism - inherent in IA-64 architecture
- Frees up hardware for parallel execution
- Predication reduces branches, enables/enhances ILP
- Control Specn breaks branch barrier, increases ILP
- Data Specn breaks data dependences, increases ILP
- Control and Data Specn address memory latency
- IA-64 provides abundant machine & mem resources
- Stack/RSE reduces call overhead and management
- Loop support yields performance w/o overhead
- And the performance features continue ...

Beyond traditional RISC capabilities
And YES ...

The Compiler DOES use these powerful architecture features to

- Enable
- Enhance
- Express
- Exploit

the Parallelism
Agenda

- Today's Architecture Challenges
- IA-64 Architecture Performance Features
  - High-End Computing Characteristics
- End User Benefits of IA-64 Features
High-end Computing Applications

- **Commercial Computing**
  - Decision Support Systems
    - In Memory Databases, Data Warehousing, Data Mining
  - E-Business
    - Authentication, Security, Transaction processing

- **Technical Computing**
  - Elect./Mech. Design Automation
    - Modeling/Simulation/FEA
  - Digital Content Creation
    - Video editing, 3D Rendering
  - Scientific / Financial Analysis
    - Seismic/Weather analysis

IA-64 designed for high-end workloads
High-End Computing Characteristics

Server Applications

- Huge Data working set
- Large Instruction foot-print
- Control intensive, non-loopy integer code
- Irregular data access patterns
- Traditionally difficult to extract ILP
- Large number of users/clients supported
- Throughput dominated requirements
IA-64 for High Performance Databases

- Number of branches in large server apps overwhelm traditional processors
  - IA-64 predication removes branches, avoids mis-predicts

- Environments with a large number of users need large systems for high performance
  - IA-64 data/control speculation reduces impact of long memory latency
  - IA-64 64-bit addressing enables systems with very large virtual and physical memory
IA-64 for ERP Applications

• Applications are control intensive integer codes with many small loops
  ■ IA-64 Rotating registers enable efficient loop execution
  ■ IA-64 Register resources provide for optimized performance

• Applications are modular and result in significant dynamic call/return
  ■ IA-64 Register stack ideally suited for call-intensive code

IA-64 provides optimizations for a diverse set of application requirements
IA-64 for E-business Applications

- Security is at the heart of E-business
- Applications are compute bound and perf is limited by branch mispredicts, ambiguous memory dependencies and large integer multiply rates
  - IA-64 Predication reduces branches, especially unpredictable ones (hence mispredicts)
  - IA-64 Data speculation (ld.c, ALAT, etc) support aggressive software optimizations in presence of ambiguous memory dependencies
  - IA-64 64-bit integer multiply boosts security algorithms
IA-64 for E-Business Applications

RSA Algorithm Estimated Performance*

Achieved thru 64-bit Integer Multiply-Add

Pentium® III Processor
Future IA-32 Processor
Itanium™ Processor

IA-64 delivers secure transactions to more users

* All third party marks, brands, and names are the property of their respective owners
IA64 for Java Applications

- Java has more method invocations than C/C++ function calls
  - IA-64 register stack engine saves spill/fill time
  - IA-64 calling convention with more registers helps it

- Java has smaller basic blocks in methods
  - IA-64 Predication removes branches increasing basic blocks
  - IA-64 Control speculation enables code motion further increasing basic block size
  - IA-64 Predication+Speculation increase parallelism

* All third party marks, brands, and names are the property of their respective owners
IA64 for Java Applications ...

- **Write barrier is performance bottleneck in garbage collection (widely used in Java)**
  - IA-64 can reserve more registers and employ predication to reduce write barrier overhead

- **Java requires exception handling functionality**
  - IA-64 recovery code mechanism is very well suited for it

- **Jini servers requires large “name space”**
  - IA-64 has large 64-bit address space
High-End Computing Characteristics

**Workstation Applications**

- Large Data working set
- Small Instruction foot-print
- Compute intensive, loopy FP code
- Traditionally easy to extract ILP
- Regular data access patterns
- Small number of users/clients supported
- Latency dominated requirements
IA-64 for Technical/Scientific Applications

- Applications demand HIGH performance Floating-Point computations
  - IA-64 FP Multiply-Add - a very efficient core computation that maps well into common algorithms
  - IA-64 FP register resources enable sufficient parallel expression/execution
  - IA-64 FP div/sqrt provide high throughput via pipelining
  - IA-64 82-bit FP provides much higher precision & range
    - Computations incur a smaller rounding error
    - Iterative calculations converge faster
    - Can handle much larger numbers than RISC w/o overflow

IA-64 FP designed for high performance
High-End Computing

**IA-64 for Technical/Scientific Applications**

- Applications are often loop intensive but have scalar components as well
  - IA-64 Software pipelining support optimizes loop structures
  - IA-64 Predication and Speculation support addresses scalar portions of FP applications
  - IA-64 NaT Value and Alternate IEEE flag sets enable FP speculations to propagate deferred exceptions and maintain IEEE flags

---

**IA-64 optimizes both loopy and scalar FP code**
IA-64 for Technical/Scientific Applications

- Applications have predictable access patterns and demand large fast memory subsystems
  - IA-64 Load pair doubles the cache-register bandwidth
  - IA-64 Data pre-fetching support allows for fast access of critical information reducing memory latency impact
  - IA-64 Cache allocation support enables optimal management of the precious cache hierarchy

IA-64 reduces the memory bottleneck
IA-64 for Graphics Applications

- Geometry calculations (transforms and lighting) use 32-bit FP numbers
  - IA-64 Parallel FP data type (2 SP values) configures registers for maximum 32-bit floating-point performance
  - IA-64 Software FP divide/square-root allow speed/accuracy tradeoffs not otherwise enabled by hardware equivalents
  - IA-64 support for Intel’s Streaming SIMD Extensions (SSE™) instructions preserves software investments

IA-64 enables world-class GFLOP performance
High-End Computing

IA-64 for Streaming Media Applications

- Audio/Video functions, [De]Compression perform the same operation on arrays of data values
  - IA-64 Parallel integer and FP data type support enable executing these functions efficiently
    - Integer Registers: 8x8, 4x16, or 2x32 bit elements
    - FP registers: 2x32, 1x64 bit elements
  - IA-64 Multimedia operands/results reside in general registers
  - IA-64 provides semantic equivalence with Intel’s MMX™ technology and Streaming SIMD Extension (SSE™) instructions

IA-64 enables fast streaming media operations
Agenda

- Today's Architecture Challenges
- IA-64 Architecture Performance Features
- High-End Computing Characteristics
  ➔ End User Benefits of IA-64 Features
<table>
<thead>
<tr>
<th><strong>IA-64/ Itanium™ Features</strong></th>
<th><strong>Function</strong></th>
<th><strong>Benefits</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explicit Parallelism:</strong></td>
<td>Enables compiler to “express” parallelism, hardware to “exploit” it</td>
<td>• Maximizes headroom for the future</td>
</tr>
<tr>
<td>compiler / hardware synergy</td>
<td>Enhances ILP by overcoming traditional barriers (branches/stores), hides memory latency</td>
<td>• Achieves higher performance where traditional architectures can’t</td>
</tr>
<tr>
<td><strong>Predication/Speculation:</strong></td>
<td>Able to optimize for scalar and object oriented applications</td>
<td>• World-class performance for complex applications</td>
</tr>
<tr>
<td>predicates, parallel compares, speculative insts and checks</td>
<td>High performance 3D graphics and scientific analysis</td>
<td>• Enables more complex scientific analysis &amp; Faster DCC/rendering</td>
</tr>
<tr>
<td><strong>Register Model:</strong></td>
<td>Improves calculation throughput for multimedia data</td>
<td>• Efficient delivery of rich Web content</td>
</tr>
<tr>
<td>large register file, rotating registers, register stack engine</td>
<td>Manages large amounts of memory, efficiently organizes data from / to memory</td>
<td>• Increased architecture &amp; system scalability</td>
</tr>
<tr>
<td><strong>Floating Point Architecture:</strong></td>
<td>Existing software runs seamlessly</td>
<td>• Preserves investment in existing software</td>
</tr>
<tr>
<td>extended precision, 128 regs FMAC, SIMD</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Multimedia Architecture:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parallel arithmetic, parallel shift, data arrangement instructions</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Memory Management:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64-bit addressing, speculation, memory hierarchy control</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Compatibility:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>full binary compatibility with existing IA-32 in hardware</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IA-64 : Enabling new levels of performance**

• Maximizes headroom for the future
• Achieves higher performance where traditional architectures can’t
• World-class performance for complex applications
• Enables more complex scientific analysis & Faster DCC/rendering
• Efficient delivery of rich Web content
• Increased architecture & system scalability
• Preserves investment in existing software
All dates specified are target dates provided for planning purposes only and are subject to change.
Collateral

• IA-64

Application Developer’s Architecture Guide

- Application instructions and machine code
- Application programming model
- Unique architecture features & enhancements
- Features and benefits for key applications
- Insight into techniques for optimizing IA-64 solutions

• Download from:

  - http://developer.intel.com/design/ia64/index.htm
  - Also on your CD!
Call to Action

- “Know” the IA-64 architecture features
- “See” the IA-64 performance advantage
- Start working on IA-64 readiness NOW!

You are HERE!