



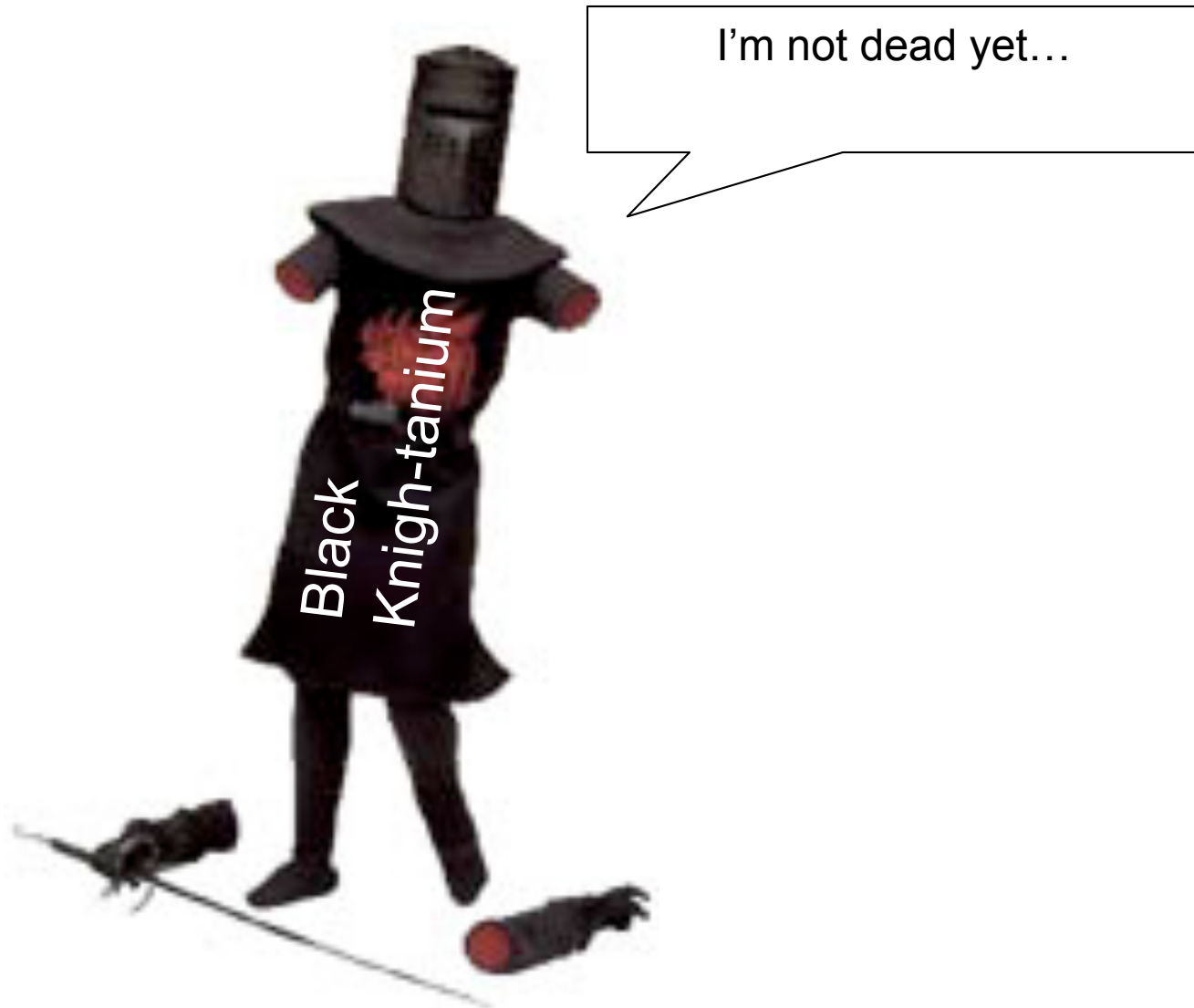
# Xen/ia64

Dan Magenheimer (djm@hp.com)  
Hewlett-Packard Laboratories

© 2004 Hewlett-Packard Development Company, L.P.  
The information contained herein is subject to change without notice



# Why Xen/ia64?





HP to invest \$3B in Itanium

...new top-of-line  
Itanium with 9MB cache

Itanium to give VMS  
new lease on life.

HP Integrity Server Solution  
Sales Pass \$1 Billion in  
2004

NASDAQ expects  
complete conversion to  
Itanium Nonstop servers

**Itanium's**

**“Rumors of my death have  
been greatly exaggerated...”**

**-- American philosopher, Mark Twain**

Fujitsu  
announces 32-  
way Itanium

Bull to sell 60 Tfloper 8704-  
processor Montecito-based  
machine to French Nuclear  
Power Agency

*Note: In this presentation:*

*Itanium = ia64 = IPF = Integrity*

SGI Columbia supercomputer for NASA  
(20x512 Itaniums) achieves 51 teraflops

# Itanium® Processor Family Roadmap (old)



2003	2004	2005	2006	2007
------	------	------	------	------

## Leading Performance

<b>4S+</b>	<b>Itanium® 2</b> Processor (Madison 6M) <i>1.5 GHz, 6M</i>	<b>Itanium® 2</b> Processor (Madison 9M) <i>9M</i>	<b>Montecito</b> <i>24MB, Dual Core, Multi-threading</i>	<b>Montvale</b> <i>24MB, Dual Core, Multi-threading</i>	<b>Tukwila</b> <i>Multi Core</i>
------------	---	--	---	--	-------------------------------------

## Leading \$/FLOPS

<b>2S</b>	<b>Itanium® 2</b> Processor <i>1.4 GHz, 3M, DP</i>	<b>Itanium® 2</b> Processor (Fanwood) <i>1.6 GHz, 3M, DP</i>	<b>Millington</b> <i>DP, Montecito-based</i>	<b>DP Montvale</b> <i>DP, Montvale-based</i>	<b>Dimona</b> <i>DP, Tukwila-based</i>
-----------	--	--	---	---	---

## Lower Power

<b>2S</b>	<b>LV Itanium® 2</b> Processor <i>1.0 GHz, 1.5M, DP</i>	<b>LV Itanium® 2</b> Processor (LV Fanwood) <i>1.3 GHz, 3M, DP</i>	<b>LV Millington</b> <i>DP, Low Voltage, Montecito-based</i>	<b>LV Montvale</b> <i>DP, Low Voltage</i>	<b>LV Dimona</b> <i>DP, Low Voltage, Tukwila-based</i>
-----------	---	--	---	--	---



### New Technologies:

- Dual-core
- Multi-threading
- Foxtan technology
- Demand Based Switching
- Pellston technology
- Silverdale technology







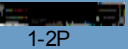
- Multi-core
- Enhanced virtualization
- Enhanced I/O & memory
- Enhanced RAS

All products, dates, comparisons, and information are preliminary and subject to change without notice.

# HP Integrity and HP 9000 Server Roadmap (old)



Revision 4.5 July-04

Current offering	2004	2005	2006
 HP 9000 Superdome 4-128P HP Integrity Superdome	CPU: mx2 module OS: HP-UX11iv2, Windows	CPU: Itanium2 9M OS: HP-UX11iv2, Windows, Linux*	CPU: PA-8900 (dual-CPU) OS: HP-UX 11iv1, v2, v3* New Chipset CPU: PA-8900 OS: HP-UX 11iv1, v2, v3 DDR-II PCI-E
 HP 9000 rp7410-8, rp8400-16, rp7420-16, rp8420-32 2-16P 2-32P HP Integrity rx7620-16, rx8620-32	CPU: mx2 module OS: HP-UX11iv2, Windows	CPU: Itanium2 9M OS: HP-UX11iv2, Windows, Linux*	CPU: PA-8900 (dual-CPU) OS: HP-UX 11iv1, v2, & v3* New Chipset CPU: PA-8900 OS: HP-UX 11iv1, v2, v3 DDR-II PCI-E
 HP 9000 rp4440-8 1-8P HP Integrity rx4640-8	CPU: mx2 module OS: HP-UX11iv2, Windows, OpenVMS*	CPU: Itanium2 9M OS: HP-UX11iv2, Windows, Linux*, OpenVMS	CPU: PA-8900 (dual-CPU) OS: HP-UX 11iv1, v2, v3* New Chipset CPU: Itanium2 "Montecito" OS: HP-UX 11iv2, v3* Windows, Linux, OpenVMS DDR-II PCI-E
 HP 9000 rp3440-4 1-4P HP Integrity rx2600-2		CPU: Itanium2 9M OS: HP-UX11iv2, Windows, Linux*, OpenVMS	CPU: PA-8900 (dual-CPU) OS: HP-UX 11iv1, v2, v3* New Chipset & Processor CPU: Itanium2 "Montecito" OS: HP-UX 11iv2, v3* Windows, Linux, OpenVMS DDR-II PCI-E
 HP Integrity rx1600-2 1-2P		CPU: Low voltage Itanium2 (+) OS: HP-UX 11iv2, v3*, Windows, Linux, OpenVMS	CPU: Next Gen & Chipset Low voltage Itanium2 OS: HP-UX, Windows, Linux, OpenVMS DDR-II PCI-E

Timeframes not to scale

All upgrades "in-box" except as noted

\*Not available at initial processor release

Plans subject to change

New Chassis Intro

PCI-Express

DDR-II Memory

# Xen/ia64 objectives/methodology

- Prove Xen can support a new (non-x86) architecture
  - root out x86-isms in Xen design/implementation
  - help solidify common/archdep boundaries/APIs in Xen
- Deliver a fully-functional open source Itanium VMM
  - many virtualization challenges: some similar to x86, some not
  - leverage HP vBlades project learnings (and code if possible)
- Leverage (and *track!*) Linux/ia64 code
  - no need to reinvent solutions for many tricky architectural subtleties
  - track ongoing performance improvements in relevant core code
  - issue: 2.4 Xen heritage vs 2.6 Xen/ia64 causes header problems
- Support un(?)modified xenlinux == Linux/ia64
  - currently requires manual “privification” build step
  - later goal: optimized, transparent paravirtualization

# Xen/ia64 design challenges

- Virtualizability NOT! (pre VTi)
  - privilege-sensitive instructions
  - write-privileged read-unprivileged registers
  - privilege leakage
- Addressing
  - up to 85-bit address space
  - 63-bit physical address space, often non-contiguous
  - multiple page sizes
  - VHPT (virtually-hashed page table) is software-managed
  - region register space must be partitioned or virtualized
- Huge register file
  - up to 586(!) registers, most are 64-bits
  - complex multi-stage state save/restore for efficiency
  - “NaT” bits
- Semi-synchronous register stack
  - grows toward memory stack
  - not completely architecturally visible

# A few Xen x86-isms

- hardware-walked page tables
  - physical addresses exposed to domains
  - contiguous physical memory
    - simplified memmap and physical memory allocation
  - shadow page tables (required for migration)
- struct `exec_domain` is a state vector (not a memory area)
- domains started in virtual mode
  - “top” of virtual address space not available to domains
- limited register set
  - “user” memory utilized in hypercalls / multicalls
- “legacy” devices assumed
  - serial, VGA, APIC timer

# Xen/ia64 differences

- Full emulation of privileged (and “privified”) instructions
- Virtual mappings / page tables not maintained by Xen
  - Xen maintains only metaphysical->physical mappings for guests
- Domain0 is a (and currently the only) “distinguished” domain
  - Metaphysical == physical (idempotently-mapped)
    - Xen grants all domain0 virtual->physical mappings
    - All I/O just works
  - Handles all device interrupts
  - Handles all ACPI parsing
    - possible exceptions: console, processors
- Xen must support multiple page sizes
- Domains start up in (meta)physical mode
- Xen not virtually mapped at “top” of memory

# Xen/ia64 Status at end of 2004

- Alpha bits released Dec'04
  - patch based on Xen 2.0.1, Linux 2.6.7
  - Xenlinux (based on 2.6.7-10) requires only config file changes
- Domain0 boots on RHEL3.2 to graphical login
  - stable, fully functional
- No other domains yet and no SMP
  - need hypercall API, front/back drivers
- Very limited platform support
  - HP rx2600
  - ski simulator on x86 (cross-compile environment available)
- Performance is poor
  - no VHPT (hardware-walked page table) support
  - no paravirtualization yet
  - compiling Linux on domain0 is 220% slower than native

# Xen/ia64 Progress 1Q05

- Community growing
  - Intel team on board; working on VTi support but helping all-around!
  - Active interest: CERN, SGI, HP Brazil, Bull
- Merged up to xeno-unstable (pre3.0)
  - no patch required on Xen common code
  - released as part of xeno-unstable.bk tree in February
  - tracking all core changes
- Additional platforms
  - Intel Tiger4 box nearly working
- Multi-domain support started
  - “domain0 clones” optionally exercise scheduling/domain switch
  - but no Virtual I/O yet, so all but real domain0 die ignominiously ☹
- Functionality
  - “user access” implemented (foundation for Xen<->guest communication)
  - test hypercalls implemented (privop counting, domain load/launch)
  - bailed on use of Linux memory allocator, switched to new Xen (Rusty’s)
- simple VHPT support added
  - still no paravirtualization yet
  - compiling Linux on domain0 is now “only” ~40% slower than native

# Xen/ia64 partial “to do” list

- Beta release: multiple domains and xend support
  - hypercall interface
  - frontend/backend drivers
- Performance
  - paravirtualize xenlinux (goal: native + 3%)
  - add perfmon support to isolate and attack hotspots
- SMP support
  - Xen first, then guests
- Breadth
  - VTi (and mixed VTi/paravirtualized) support
  - more hardware platforms
- Track Xen and Linux releases
- Broader testing/support
  - kernels, distributions
- Documentation ☺

# Questions?



i n v e n t

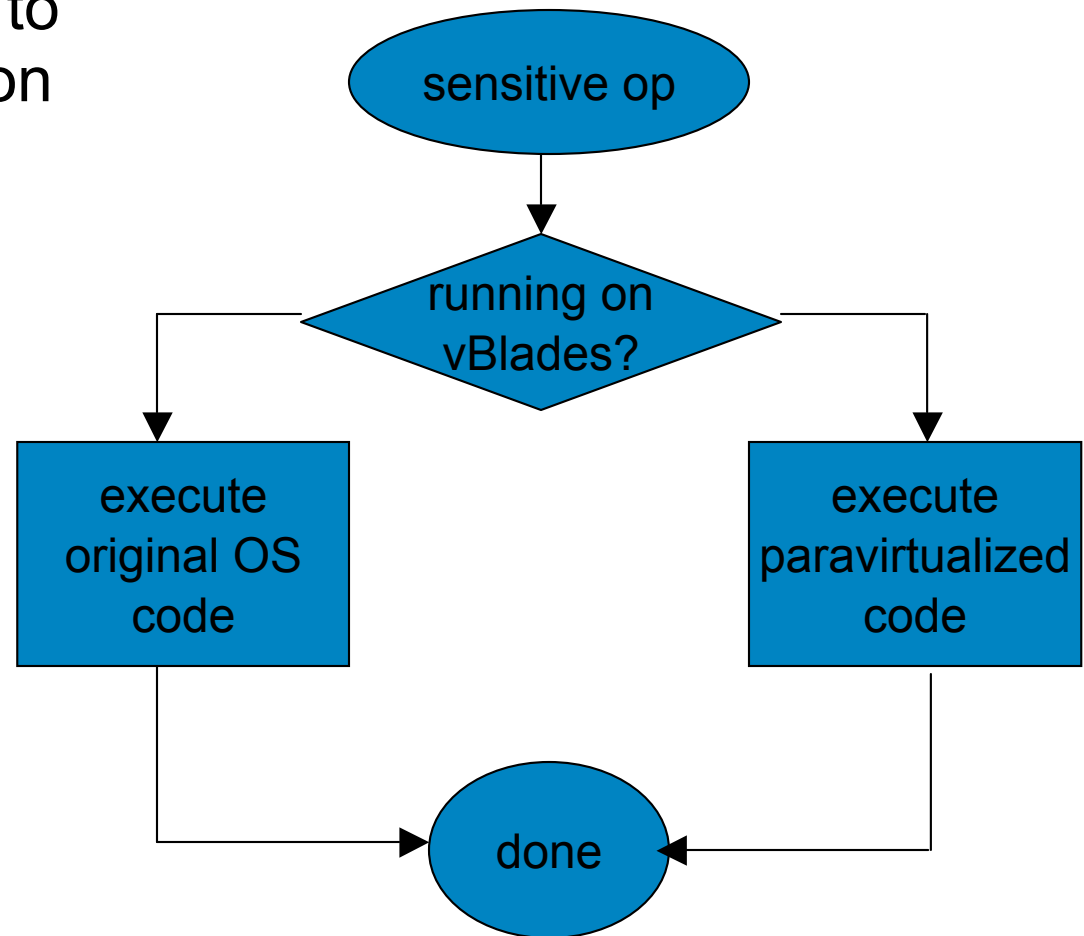
# HP Labs vBlades contributions

- First Itanium VMM
- Optimized paravirtualization
- Transparent paravirtualization
- Fully virtualized physical (“metaphysical”) memory

cf. Magenheimer and Christian, “vBlades: Optimized Paravirtualization for the Itanium Processor Family”, *Usenix 3<sup>rd</sup> Virtual Machine Research and Technology Symposium (VM’04)*

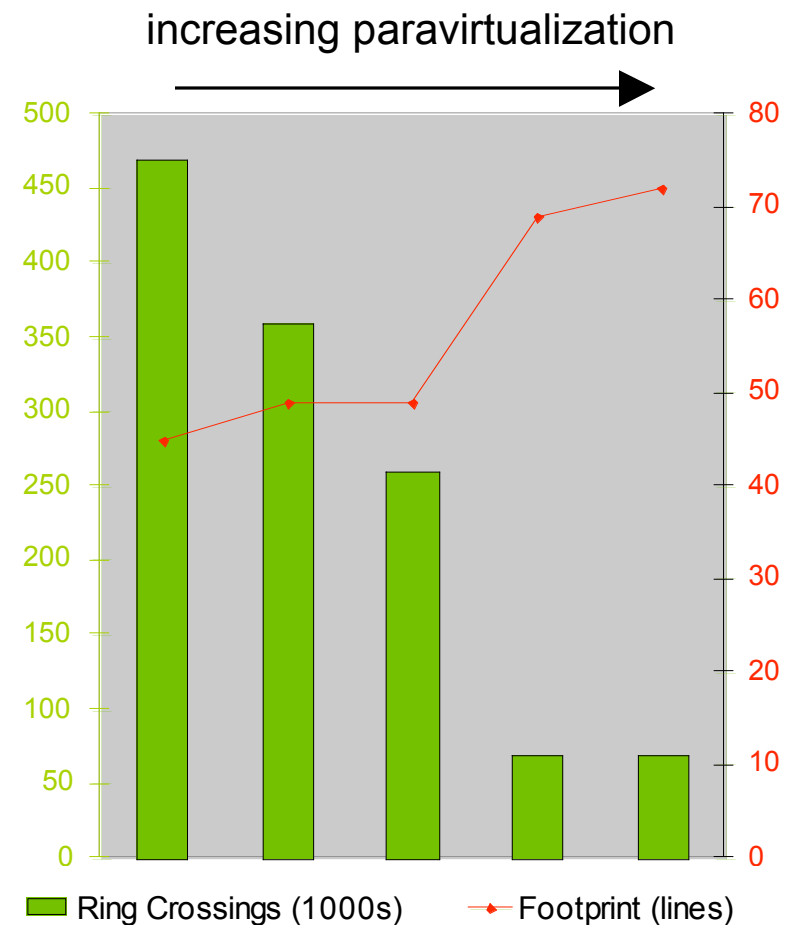
# Transparent paravirtualization

- Allows common binary to run both on VMM and on bare metal.
- Performance impact is negligible ( $< 0.1\%$ )



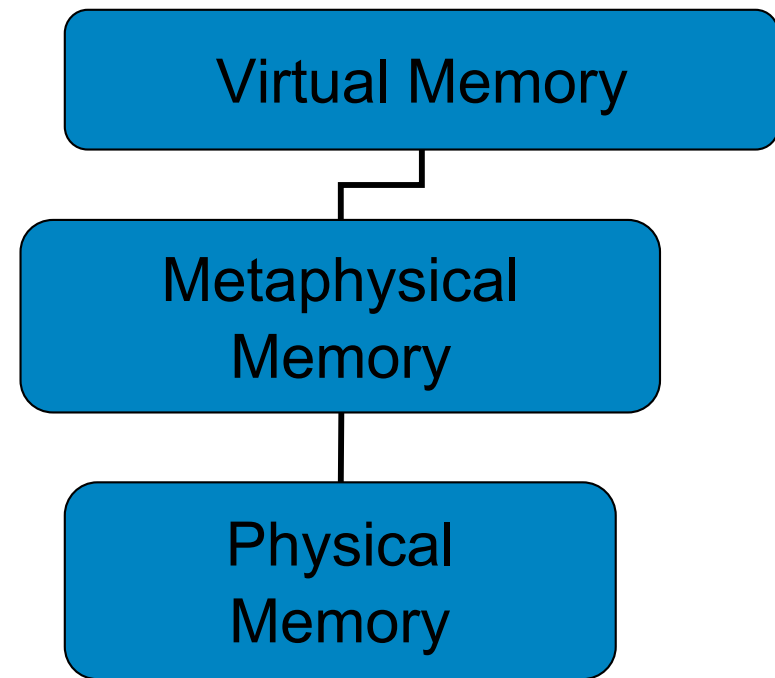
# Optimized paravirtualization

- Supporting both virtualization and paravirtualization provides valuable flexibility
  - “80/20” tuning rule applies
  - most startup code can remain unchanged
  - minimizing changes is a good thing



# Metaphysical\* memory

- Presents illusion of physical memory to guest OS
- Eliminates need to paravirtualize physical memory code
- Supports oversubscription
- Simplifies live migration



\* M-W: “a reality beyond what is available to the senses”

# Xen/ia64 challenge: Virtualizability

- Privilege-sensitive instructions (cover,thash,ttag,fc)
- Privilege-sensitive registers (ar.kr[8], ar.cflg,cpuid[6],pmd[n])
- Four privilege levels
  - can use privilege compression (unless guest requires all four)
  - privilege level leakage expensive to avoid (don't!)
- Virtual addressing
  - hypervisor must claim some VA space for self (and for shared\_info)
  - region registers must be virtualized (or partitioned)
  - software-managed TLB, hardware-walked VHPT (but can be “disabled”)
- Most exceptions/interrupts can be quickly reflected
  - virtualization of Itanium register stack engine is painful, but possible

# Xen/ia64 challenge: Addressing

- full 64-bit flat virtual address space
  - divided into 8 regions of  $2^{61}$  bytes each
- eight region registers (18- to 24-bit) provide task isolation
  - total: 79- to 85-bits of virtual address space!
- 63-bit flat physical address space w/uncached alias many page sizes supported, from 4Kbyte to  $\geq 256$ Mbyte

# Xen/ia64 challenge: huge register file

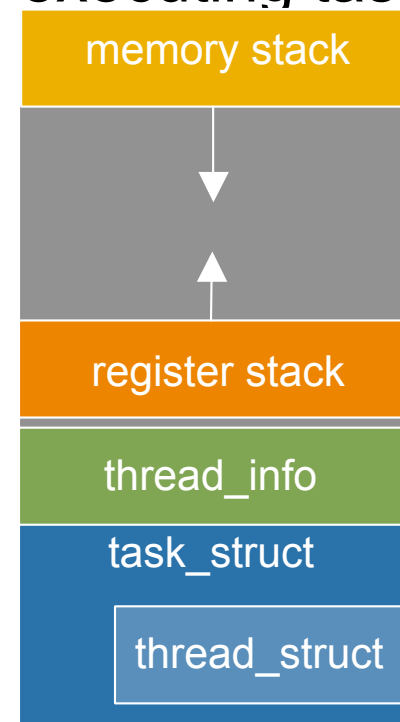
NaT

name:	type:	width:	usage:
r0-r127	general register	65 bits	integer & multi-media calc.
f0-f127	floating-point register	82 bits	single, parallel, double, extended, & integer calc.
p0-p63	predicate register	1 bit	boolean calc.
b0-b7	branch register	64 bits	branches & prediction
ar0-ar127	application register	64 bits	misc. (e.g., loop counter)
cr0-cr127	control register	64 bits	privileged control state
psr	processor status reg.	64 bits	control & status (e.g., byte order)
ip	instruction pointer	64 bits	program counter

⇒ up to 586 registers!

# Xen/ia64 challenge: two stacks

- Linux represents each process and (kernel-) thread with a structure called “struct task\_struct”
  - each such structure has an associated stack and “thread\_info”
  - global variable “current” points to the currently executing task
- For IA-64:
  - 2 separate stacks needed:
    - memory stack (grows down)
    - register stack (grows up)
  - all structures & stacks bundled together
  - current task's memory is pinned into DTLB
  - “current” is stored in register r13 (tp)





# Xen/ia64 challenge: complex state save

- `struct pt_regs` (448 bytes)
  - Saved on entry into kernel (partly saved on syscall entry)
  - Contains mostly the scratch registers
  - Nested interruptions cause multiple `pt_regs` on the same stack
- `struct switch_stack` (528 bytes)
  - Saved when tasks blocks execution (sleeps)
  - Contains mostly the preserved registers
  - At most one `switch_stack` per task
- `struct thread_struct` (2320 bytes)
  - Part of task structure
  - Lazy state: fp-high partition, debug-, perfmon-, & IA-32 registers
- Kernel restricted from using most fp-registers