CS 61C: Great Ideas in Computer Architecture *Virtual Machines 2*

Instructor:
David A. Patterson
http://inst.eecs.Berkeley.edu/~cs61c/sp12

Review

- Virtual Memory, Paging for Isolation and Protection, help Virtual Machines share memory
  - Can think of as another level of memory hierarchy, but not really used like caches are today
  - Not really routinely paging to disk today
- Virtual Machines as even greater level of protection to allow greater level of sharing
  - Enables fine control, allocation, software distribution, multiple price points for Cloud Computing
  - Even on laptops (e.g., Windows on Mac using Parallels)

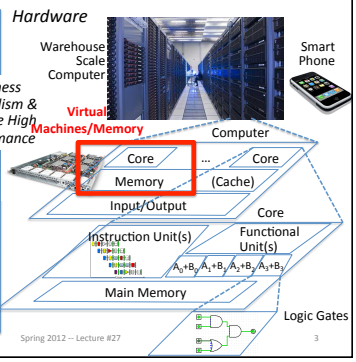
You Are Here!

Today's Lecture

Software | *Hardware*

- Parallel Requests
  Assigned to computer
  e.g., Search "Katz"
- Parallel Threads
  Assigned to core
  e.g., Lookup, Ads
- Parallel Instructions
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- Parallel Data
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- Hardware descriptions
  All gates @ one time
- Programming Languages

Harness Parallelism & Achieve High Performance



Agenda

- Virtual Memory, Pass 2
- Translation Lookaside Buffer (TLB)
- A day in the life of an instruction with TLB
- Administrivia
- TLB Performance Impact and Nehalem
- Matrix Multiply: Inside Edition!
- Peer Instructions
- Reduced Instruction Set Computers (if time is available)
- Summary

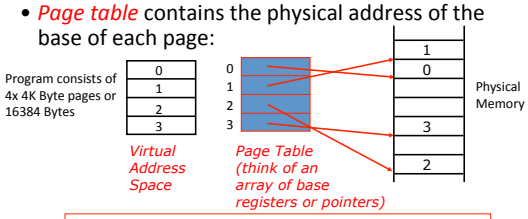
Paged Memory Systems

- Processor-generated address is split into:

page number	offset
20 bits	12 bits

32-bit byte address
4096 byte pages

- *Page table* contains the physical address of the base of each page:

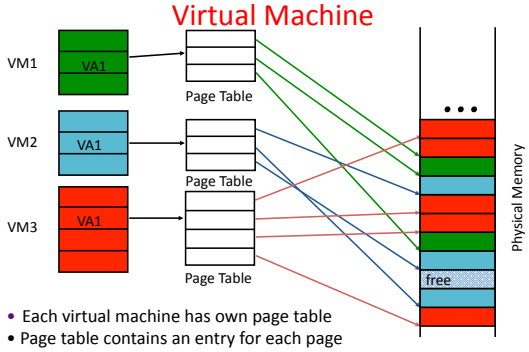


Page tables make it possible to store the pages of a program non-contiguously.

Library Analogy

- Book title like *virtual address*
- Library of Congress call number like *physical address*
- Card catalogue like *page table*, mapping from book title to call #
- On card, available for 2-hour in library use (vs. 2-week checkout) like *access rights*

Separate Address Space per Virtual Machine



- Each virtual machine has own page table
- Page table contains an entry for each page

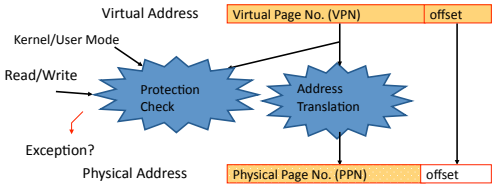
Paging Terminology

- Program addresses called *virtual addresses*
  - Space of all virtual addresses called *virtual memory*
- Memory addresses called *physical addresses*
  - Space of all physical addresses called *physical memory*

Notes on Page Table

- Solves the fragmentation problem: all chunks same size, so any holes can be used
- OS must reserve Swap Space on disk *for each virtual machine*
- To grow a virtual machine, ask Op. System
  - If unused pages available, OS uses them first
  - If not, OS swaps some old pages to disk
  - (*Least Recently Used* to pick pages to swap)
- How/Why grow a process?

Address Translation & Protection

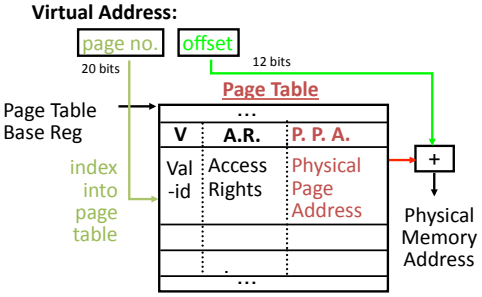


Every instruction and data access needs address translation and protection checks

Protection via Page Table

- Access Rights checked on every access to see if allowed
  - Read: can read, but not write page
  - Read/Write: read or write data on page
  - Execute: Can fetch instructions from page
- Valid = Valid page table entry
  - Invalid means it's on the disk, not in physical memory

More Depth on Page Tables



Page Table located in physical memory

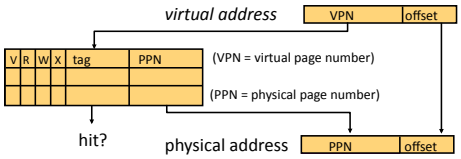
Virtual Memory Without Doubling Memory Accesses

- Caches suggest that there is temporal and spatial locality of data
- Locality really means locality of *addresses* of data
- What about locality of translations of virtual page addresses into physical page addresses?
- Why not a *Page Table Address Translation Cache*?
- For historical reasons, called *Translation Lookaside Buffer* (TLB)

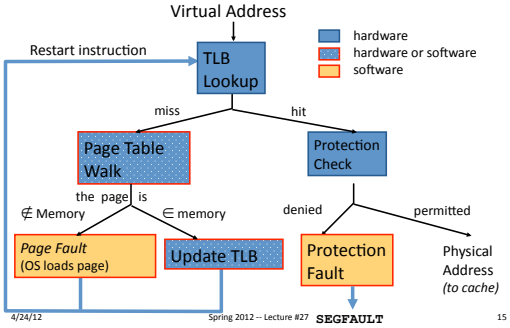
Translation Lookaside Buffers (TLB): Another Layer of Indirection!

Address translation is very expensive!
Each reference becomes 2 memory accesses

Solution: *Cache address translations in TLB!*
TLB hit ⇒ *Single Cycle Translation*
TLB miss ⇒ *Access Page-Table to refill*



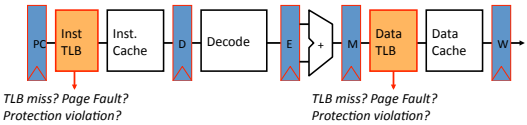
Address Translation: Putting it all Together



TLB Design

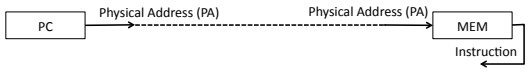
- Typically 32-128 entries
  - Usually fully associative
  - Each entry maps a large page, hence less spatial locality *across* pages
- A *memory management unit* (MMU) is hardware that walks the page tables and reloads the TLB

Address Translation in CPU Pipeline



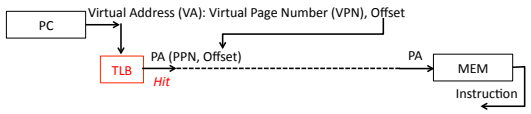
- Handling a TLB miss needs a *hardware* or *software* mechanism to refill TLB
- Need mechanisms to cope with the additional latency of a TLB:
  - Slow down the clock
  - Pipeline the TLB and cache access

Day in the Life of an (Instruction) Address



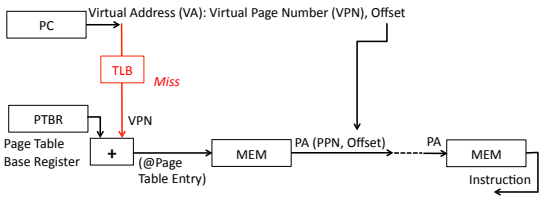
No Cache, No Virtual Memory

Day in the Life of an (Instruction) Address



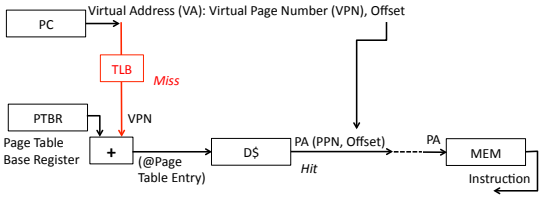
If locality works, this is the common case!
No Cache, Virtual Memory, TLB Hit

Day in the Life of an (Instruction) Address



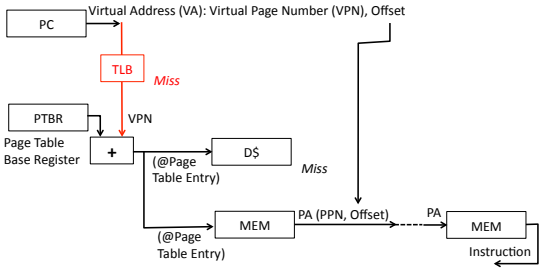
No Cache, Virtual Memory, TLB Miss, Page Table Access

Day in the Life of an (Instruction) Address



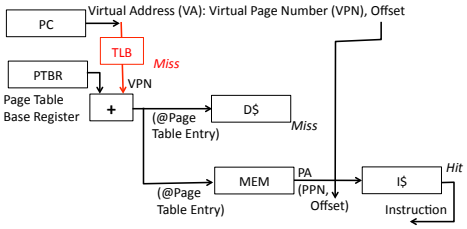
Physical Data Cache, Virtual Memory, TLB Miss, Page Table Access

Day in the Life of an (Instruction) Address



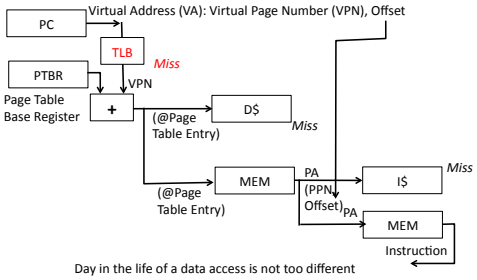
Physical Data Cache, Virtual Memory, TLB Miss, Page Table Access

Day in the Life of an (Instruction) Address



Physical Data & Instruction Cache, Virtual Memory, TLB Miss, Page Table Access

Day in the Life of an (Instruction) Address



Day in the life of a data access is not too different
Physical Data & Instruction Cache, Virtual Memory, TLB Miss, Page Table Access

Administrivia

- 1 more Lecture: End off course Overview/Cal Culture/HKN Evaluation
  - film "Powers of 10"? Will show 9:31AM Th
  - www.powersof10.com/
- Please fill out final 61C Survey: https://www.surveymonkey.com/s/61Cspr12Final
- All grades but Project 4 finalized: 4/27; Proj 4 5/1
- Go to lab to finalize any labs Th/Fr
- Final Review: Sunday April 29, 2-5PM, 2050 VLSB
- Extra office hours: Thu-Fri May 3-4 1 to 5PM
- Final: Wed May 9 11:30-2:30, 1 PIMENTEL

Agenda

- Virtual Memory, Pass 2
- Translation Lookaside Buffer (TLB)
- A day in the life of an instruction with TLB
- Administrivia
- TLB Performance Impact and Nehalem
- Matrix Multiply: Inside Edition!
- Peer Instructions
- Reduced Instruction Set Computers (if time is available)
- Summary

Impact of TLBs on Performance

- Each TLB miss to Page Table ~ L1 Cache miss
- Page sizes are 4 KB to 8 KB (4 KB on x86)
- TLB has typically 128 entries
  - Set Associative or Fully Associative
- *TLB Reach*: Size of largest virtual address space that can be simultaneously mapped by TLB:
- 128 * 4 KB = 512 KB = 0.5 MB!
- *What can you do to have better performance?*

Student Roulette?

Improving TLB Performance

- Add larger page size that operating system can use in situations when OS knows that object is big and protection of whole object OK
  - X86 has 4 KB pages + 2 MB and 4 MB "superpages"
- Have 2 Levels of TLB just like 2 levels of cache instead of going directly to Page Table on L1 TLB miss

Impact on TLB

- Page Table keeps track of whether page needs to be written back to disk if its been modified
  - Has a Page Table Dirty Bit
- So also have "Page Dirty Bit" in TLB, set when any data in page is written
- When TLB entry replaced, corresponding Page Dirty Bit is set in Page Table Entry

Nehalem Virtual Memory Details

- 48-bit virtual address space, 40-bit physical address space
- Split, 2-level TLB: I L1, D L1 + U L2 per core
- I-TLB (L1) has shared 128 entries 4-way associative for 4KB pages, plus 7 dedicated fully-associative entries per SMT thread for large page (2/4MB) entries
- D-TLB (L1) has 64 entries for 4KB pages and 32 entries for 2/4MB pages, both 4-way associative, dynamically shared between SMT threads
- Unified L2 TLB has 512 entries for 4KB pages only, also 4-way associative
- Data TLB Reach
  (4 KB only) L1: 64*4 KB =0.25 MB, L2:512*4 KB= 2MB
  (superpages) L1: 32 *2-4 MB = 64-128 MB

Agenda

- Virtual Memory, Pass 2
- Administrivia
- Translation Lookaside Buffer (TLB)
- A day in the life of an instruction with TLB
- TLB Performance Impact and Nehalem
- Matrix Multiply: Inside Edition!
- Peer Instructions
- Reduced Instruction Set Computers (if time is available)
- Summary

Peer Instruction: True or False

Which statements is *True* about Virtual Machines?

I. Multiple Virtual Machines can run on one computer

II. Multiple Virtual Machine Monitors can run on one computer

III. The Guest OS must be the same as the Host OS

A) I only
B) II only
C) III only
D) I and II

Peer Instruction: True or False

Which statements is *True* about Virtual Machines?

I. Multiple Virtual Machines can run on one computer

II. Multiple Virtual Machine Monitors can run on one computer

III. The Guest OS must be the same as the Host OS

A) I only
B) II only
C) III only
D) I and II

Peer Instruction: True or False

A program tries to load a word X that causes a TLB miss but not a page fault. Which are True or False:

1. A TLB miss means that the page table does not contain a valid mapping for virtual page corresponding to the address X
2. There is no need to look up in the page table because there is no page fault
3. The word that the program is trying to load is present in physical memory.

A) 1 F, 2 F, 3 F
B) 1 F, 2 F, 3 T
C) 1 F, 2 T, 3 F
D) 1 F, 2 T, 3 T

Peer Instruction: True or False

A program tries to load a word X that causes a TLB miss but not a page fault. Which are True or False:

1. A TLB miss means that the page table does not contain a valid mapping for virtual page corresponding to the address X
2. There is no need to look up in the page table because there is no page fault
3. The word that the program is trying to load is present in physical memory.

A) 1 F, 2 F, 3 F
B) 1 F, 2 F, 3 T
C) 1 F, 2 T, 3 F
D) 1 F, 2 T, 3 T

Peer Instruction: True or False

TLBs entries have valid bits and dirty bits. Data caches have them also.

A. The valid bit means the same in both: if valid = 0, it must miss in both TLBs and Caches.

B. The valid bit has different meanings. For caches, it means this entry is valid if the address requested matches the tag. For TLBs, it determines whether there is a page fault (valid=0) or not (valid=1).

C. The dirty bit means the same in both: the data in this block in the TLB or Cache has been changed.

D. The dirty bit has different meanings. For caches, it means the data block has been changed. For TLBs, it means that the page corresponding to this TLB entry has been changed.

A) 1 F, 2 T, 3 F, 4 T
B) 1 F, 2 T, 3 T, 4 F
C) 1 T, 2 F, 3 F, 4 T
D) 1 T, 2 F, 3 T, 4 F

Peer Instruction: True or False

TLBs entries have valid bits and dirty bits. Data caches have them also.

A. The valid bit means the same in both: if valid = 0, it must miss in both TLBs and Caches.

B. The valid bit has different meanings. For caches, it means this entry is valid if the address requested matches the tag. For TLBs, it determines whether there is a page fault (valid=0) or not (valid=1).

C. The dirty bit means the same in both: the data in this block in the TLB or Cache has been changed.

D. The dirty bit has different meanings. For caches, it means the data block has been changed. For TLBs, it means that the page corresponding to this TLB entry has been changed.

A) 1 F, 2 T, 3 F, 4 T

B) 1 F, 2 T, 3 T, 4 F

C) 1 T, 2 F, 3 F, 4 T

D) 1 T, 2 F, 3 T, 4 F

Peer Instruction: Match the Phrase

Match the memory hierarchy element on the left with the closest phrase on the right:

1. L1 cache	a. A cache for page table entries
2. L2 cache	b. A cache for a main memory
3. Main memory	c. A cache for disks
4. TLB	d. A cache for a cache

A) 1 a, 2 b, 3 c, 4 d

B) 1 a, 2 b, 3 d, 4 c

C) 1 b, 2 d, 3 a, 4 c

D) 1 d, 2 b, 3 c, 4 a

Peer Instruction: Match the Phrase

Match the memory hierarchy element on the left with the closest phrase on the right:

1. L1 cache	a. A cache for page table entries
2. L2 cache	b. A cache for a main memory
3. Main memory	c. A cache for disks
4. TLB	d. A cache for a cache

A) 1 a, 2 b, 3 c, 4 d

B) 1 a, 2 b, 3 d, 4 c

C) 1 b, 2 d, 3 a, 4 c

D) 1 d, 2 b, 3 c, 4 a

Agenda

- Virtual Memory, Pass 2
- Administrivia
- Translation Lookaside Buffer (TLB)
- A day in the life of an instruction with TLB
- TLB Performance Impact and Nehalem
- Matrix Multiply: Inside Edition!
- Peer Instructions
- Reduced Instruction Set Computers (if time is available)
- Summary

PATTERSON'S PROJECTS

Years	Title	Profs	Students
1977-81	X-Tree: A Tree-Structured Multiprocessor	3	12
1980-84	RISC: Reduced Instruction Set Computer	3	17
1983-86	SOAR: Smalltalk On A RISC	2	12
1985-89	SPUR: Symbolic Processing Using RISCs	6	21
1988-92	RAID: Redundant Array of Inexpensive Disks	3	16
1993-98	NOW: Network of Workstations	4	25
1997-02	IRAM: Intelligent RAM	3	12
2001-05	ROC: Recovery Oriented Computing	2	11
2005-10	RAD Lab: Reliable Adaptive Distributed Systems Laboratory	7	30
2007-12	Par Lab: Parallel Computing Laboratory	9	45
2011-16	AMP Lab: Algorithms, Machines, People	6	30

*Each helped lead to a multibillion dollar industry *Parallel Processor Projects

RISC vs. CISC

- Set up: From 1965 to 1980, virtually all computers implemented instruction sets using *microcode* (edited wikipedia entry):
  "Microcode is a layer of hardware-level instructions involved in the implementation of higher-level machine code instructions; it resides in a special high-speed memory and translates machine instructions into sequences of detailed circuit-level operations. It helps separate the machine instructions from the underlying electronics so that instructions can be designed and altered more freely. It also makes it feasible to build complex multi-step instructions while still reducing the complexity of the electronic circuitry compared to other methods. Writing microcode is often called *microprogramming* and the microcode in a particular processor implementation is sometimes called a *microprogram*."
- 1980s compilers rarely generated these complex instructions

RISC – CISC Wars

- **Round 1: The Beginning of Reduced vs. Complex Instruction Set**
  - Instruction set made up of simple or reduced instructions using easy-to-decode instruction formats and lots of registers was a better match to integrated circuits and compiler technology than the instructions sets of the 1970s that featured complex instructions and formats.
  - Our counterexamples were the Digital VAX-11/780, the Intel iAPX-432, and the Intel 8086 architectures, which we labeled Complex Instruction Set Computers (CISC).
  - "The Case for the Reduced Instruction Set Computer", David Patterson and David Ditzel, *ACM SIGARCH Computer Architecture News*, October 1980

http://blogs.arm.com/software-enablement/375-risc-versus-cisc-wars-in-the-prepc-and-pc-eras-part-1/

RISC – CISC Wars

- **Round 1: The Beginning of Reduced vs. Complex Instruction Set**
  - To implement their more sophisticated operations, CISCs relied on microcode, which is an internal interpreter with its own program memory.
  - RISC advocates essentially argued that these simpler internal instructions should be exposed to the compiler rather than buried inside an interpreter within a chip.
  - RISC architects took advantage of the simpler instruction sets to first demonstrate pipelining and later superscalar execution in microprocessors, both of which had been limited to the supercomputer realm.

http://blogs.arm.com/software-enablement/375-risc-versus-cisc-wars-in-the-prepc-and-pc-eras-part-1/

Original RISC Slides

- *See slides 1 to 16 from RISCTalk1981v6.pdf*
  - *Unedited transparencies from 1981 RISC talk + RISC I, RISC II die photos*

RISC – CISC Wars

- **Round 1: The Beginning of Reduced vs. Complex Instruction Set**
  - still amazing that was a time when graduate students could build a prototype chip that was actually faster than what Intel could build.
  - ARM, MIPS, and SPARC successfully demonstrated the benefits of RISC in the marketplace of the 1980s with rapidly increasing performance that kept pace with the rapid increase in transistors from Moore's Law.

http://blogs.arm.com/software-enablement/375-risc-versus-cisc-wars-in-the-prepc-and-pc-eras-part-1/

RISC – CISC Wars

- **Round 2: Intel Responds and Dominates the PC Era**
- Most executed instructions simple
  - HW translated simple x86 instructions into internal RISC instructions, then use RISC ideas: pipelining, superscalar, ...
- Intel "CISC tax": longer pipelines, extra translation HW, and the microcode for complex operations but:

1. Intel's fab line better than RISC companies, so smaller geometries hide some CISC Tax
2. Moore's Law => on-chip integration of FPUs & caches, over time CISC Tax became smaller %
3. Increasing popularity of IBM PC + distribution of SW in binary made x86 ISA valuable, no matter what tax

RISC – CISC Wars

- **Round 2: Intel Responds and Dominates in the PC Era**
- Wikipedia: "While early RISC designs were significantly different than contemporary CISC designs, by 2000 the highest performing CPUs in the RISC line were almost indistinguishable from the highest performing CPUs in the CISC line."

http://blogs.arm.com/software-enablement/375-risc-versus-cisc-wars-in-the-prepc-and-pc-eras-part-1/

RISC – CISC Wars

- **RISC vs. CISC in the PostPC Era**
- CISC not a good match to the smartphones and tablets of the PostPC era

1. It's a new software stack and software distribution is via the "App Store model" or the browser, which lessens the conventional obsession with binary compatibility.
2. RISC designs are more energy efficient.
3. RISC designs are smaller and thus cheaper.

http://blogs.arm.com/software-enablement/377-risc-versus-cisc-wars-in-the-postpc-eras-part-2/

RISC vs. CISC 2010 (mobile client)

	Broadcom BRCM 5000	ARM Cortex-A9	ARM Cortex-A9	Intel Atom CPU	MIPS 74K
Design Type	Hard core	Hard (Osprey)	Soft core	Hard core	Soft core
Instruction Set	MIPS32	ARMv7	ARMv7	x86-64	MIPS32
CPU Speed*	1.3GHz	1.3GHz‡	1.0GHz‡	1.6GHz	1.2GHz
Issue Rate	2 per cycle	2 per cycle	2 per cycle	2 per cycle	2 per cycle
Instr Reordering?	—	Yes	Yes	—	Yes
Multithreaded?	2 threads	—	—	2 threads	—
Pipeline Depth	12 stages	9 stages	9 stages	16 stages	15 stages
L1 Cache (I/D)	32KB/32KB	32KB/32KB	32KB/32KB	32KB/24KB	32KB/32KB
Special Instructions	DSP, FPU	DSP, FPU, Neon, Jazelle	DSP, FPU, Neon, Jazelle	SSE3, FPU	DSP, FPU
CoreMark 1.0	3,890 CM	3,245 CM	2,880 CM	5,106 CM	3,060 CM
CM 1.0/MHz	2.99/MHz	2.88/MHz	2.88/MHz	3.19/MHz	2.55/MHz
CPU Die Area†	1.9mm²	3.4mm²	1.35mm²	6mm²‡	1.5mm²‡
CPU Power†	0.37mW/MHz	0.4mW/MHz‡	0.4mW/MHz‡	0.56mW/MHz‡	0.43mW/MHz
IC Process	40nm G	40nm G	40nm G	45nm high-*k*	40nm G
Chip Production	4Q10	2Q11	2Q10	2Q08	2Q11 (40nm)

Table 1. Comparison of Broadcom's BRCM 5000 CPU with ARM, MIPS, and Intel CPUs. *Production-silicon clock rate, no overvoltage; †includes L1 caches as shown. (Source: vendors, except ‡The Linley Group estimate)

"Broadcom Shows Off New CPU," Linley Gwennap, *Microprocessor Report*, November 22, 2010

- X86 ≈ 1.04 – 1.08x better performance/MHz vs. MIPS, ARM
- MIPS, ARM ≈ 1.4 – 1.5x better energy/MHz vs. x86
- MIPS, ARM ≈ ⅓ to ¼ die area vs. x86

RESOLVING RISC-CISC DEBATE

Products shipped?

2010: 8.1B microprocessors

6.1B ARM, 0.3B x86 (75%+ RISC)

How USA resolves debates?

We ask celebrities!

Who is the biggest celebrity in the world?

RESOLVING RISC-CISC Debate

- Angelina Jolie as Kate Libby (aka as hacker Acid Burn) in movie "Hackers" (1995)

RESOLVING RISC-CISC Debate

Angelina Jolie: "RISC architecture is gonna change everything."

"Hackers" (1995)

Blue Man Group "(silence)"

And in Conclusion, …

- Separate Memory Management into orthogonal functions:
  - *Translation* (mapping of virtual address to physical address)
  - *Protection* (permission to access word in memory)
  - But most modern systems provide support for all functions with single page-based system
- All desktops/servers have full demand-paged virtual memory
  - Portability between machines with different memory sizes
  - Protection between multiple users or multiple tasks
  - Share small physical memory among active tasks
  - Simplifies implementation of some OS features
- Hardware support: User/Supervisor Mode, invoke Supervisor Mode, TLB, Page Table Register