

Computer Architecture

Richard H. Eckhouse
Math & Computer Science Department
University of Massachusetts
eckhouse@cs.umb.edu

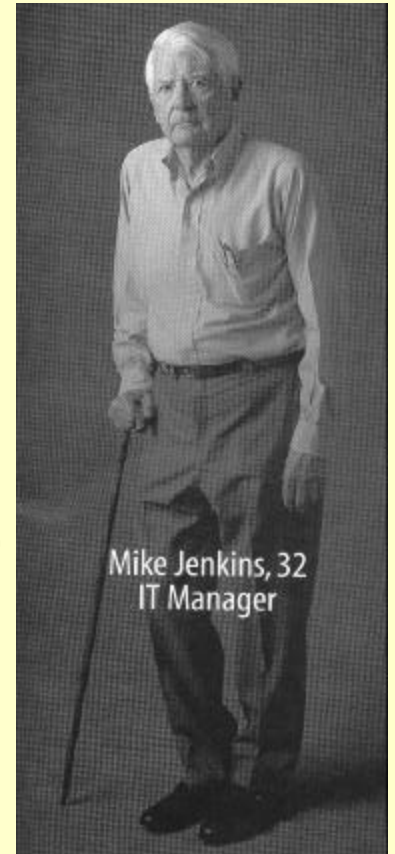
Rules of the Game

- Rapidly changing field:

- ◆ Vacuum tube → transistor → IC → VLSI
- ◆ Memory capacity and processor speed doubling every 1.5 years
- ◆ What is taught about computer architecture
 - How computers work, a basic foundation
 - How to analyze their performance (or how not to!)
 - Issues affecting modern processors (caches, pipelines)

- Why learn this stuff?

- ◆ You want to call yourself a “computer scientist”
- ◆ You want to build software people use (need performance)
- ◆ You need to make a purchasing decision or offer “expert” advice

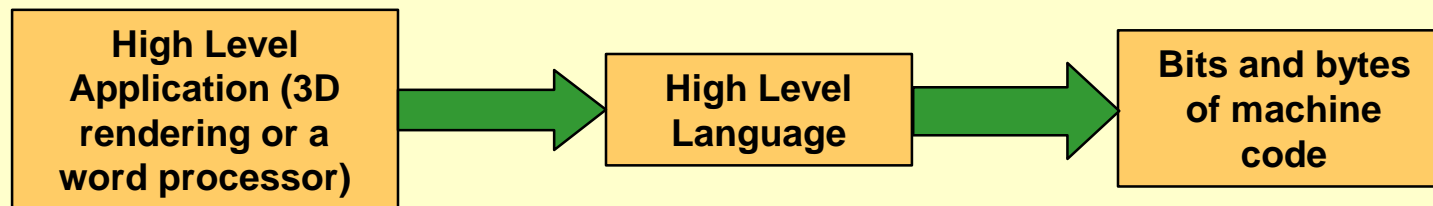


In the Beginning

- Our first experiences with computers
 - ◆ Black box model
 - ◆ A façade
- What lies under the covers?
 - ◆ Desktop calculator
 - von Neumann legacy
 - Simple design
 - ◆ Representation of a mechanical human
 - The why of things
 - The idiosyncrasies
- Evolutionary process

Abstraction

- Delving into the depths reveals more information
- An abstraction omits unneeded detail, helps us cope with complexity



Architectural View of Abstraction

- Must distinguish between the *logical* and the *physical* (sometimes called the *virtual* and the *real*)
- This leads to the concept of a “*Multi-Level Interpretive Computing System*” otherwise known as a “*layered*” system
- How many layers can we identify?
 - ◆ Applications level
 - ◆ Programming language level
 - ◆ Run-time environment
 - ◆ Operating system level
 - ◆ Native machine level
 - ◆ Hardware

A Little History (American Perspective)

- 1940s (WW II): Individual Machines
 - ◆ John Atanasoff, Iowa State University, ABC
 - ◆ Howard Aiken, Harvard University, Mark I
 - ◆ J Presper Eckert & John Mauchly, U of Penn, ENIAC
 - ◆ Maurice Wilkes, Cambridge University, EDSAC
 - ◆ John von Neumann, Princeton, IAS
- 1950s: Technological breakthroughs
 - ◆ Up from relays & tubes & delay lines:
 - ◆ Transistor
 - ◆ Magnetic-Core Memory
 - \$1,000,000 for 1 MByte, 1 μ -sec
 - (*cf.* \$40 for 1 MByte, .070 μ -sec for solid-state, 1994)

A Little More History

- 1960s
 - ◆ IBM
 - Computer family architecture
 - Microprogrammed control units
 - ◆ Burroughs
 - Execution stack architecture
 - ◆ CDC
 - Pipelining
 - Multiple arithmetic units
 - ◆ University of Illinois
 - Highly parallel computer (Illiack IV)

Yet More History

- Mini-Computers

- ◆ DEC
- ◆ Data General
- ◆ Prime

- Workstations

- ◆ Hewlett-Packard
- ◆ Next
- ◆ Sun

- Personal PCs

- ◆ MITS (Altair)
- ◆ Apple
- ◆ IBM/PC

What Is Computer Architecture?

- Coined by IBM with the design of the S/360 family
 - ◆ “... the structure of a computer that a machine language programmer must understand to write a correct (timing independent) program for that machine.” from P&H
 - ◆ Put another way: the user visible portion of the instruction set
- Computer organization
 - ◆ High-level aspects
 - Memory system
 - Bus structure
 - CPU internals
- Implementation details
 - ◆ Technology
 - ◆ Design details

Interesting Points to Consider

- The importance of the family concept
- Architecture persists longer than implementations, especially user-level ISA
- What does a computer designer do
 - ◆ Iterative process with software designers
- Amdahl propagation of existing architecture
- The IBM compatible PC
 - ◆ The successes (Compaq)
 - ◆ The failures (Columbia and Eagle)

Bottom Line

- Architecture is now important to all facets of computing
 - ◆ Networks
 - ◆ Operating systems
 - ◆ Graphics

- Ex. StoreX initiative
 - ◆ A Java-based network storage management standard that would link storage products from different vendors

CA Topics of Interest

- **Proceedings of the 25th Annual International Symposium on Computer Architecture**

- ◆ Machine measurement
- ◆ Program behavior
- ◆ Graphics and I/O
- ◆ Speculation
- ◆ Prediction techniques
- ◆ Memory management
- ◆ Prediction and multipath execution
- ◆ Processor microarchitecture
- ◆ Parallel machines
- ◆ Caches and memory systems

CA Topics of Interest (Cont'd)

- **Eighth International Conference on Architectural Support for Programming Languages and Operating Systems**
 - ◆ Compiler optimizations for memory
 - ◆ Speculation and ILP compilation
 - ◆ I/O systems
 - ◆ Caches and memory systems
 - ◆ Cross platform techniques and branch prediction
 - ◆ Multiprocessor systems
 - ◆ Cache analysis
 - ◆ Simulation and performance

Retrospective

- **25 Years of the International Symposia on Computer Architecture: Selected Papers**

- ◆ Network and interconnects
- ◆ Pipelining
- ◆ Caching
- ◆ RISC
- ◆ Performance
- ◆ Parallel Processors
- ◆ Branch prediction
- ◆ VLIW and ILP
- ◆ Dataflow
- ◆ Consistency and ordering
- ◆ HLL architectures

Which Are Really About Architecture?

- Strictly speaking very few
 - ◆ Program behavior and performance
 - ◆ Parallel machines and multiprocessors
 - ◆ RISC/CISC
 - ◆ VLIW and maybe ILP
 - ◆ Dataflow and HLL machines
- All the rest have to do with organization
 - ◆ I/O subsystems
 - ◆ Caches and memory systems
 - ◆ Branch prediction and speculative execution
 - ◆ Pipelining

But What Really Concerns Us?

- Personally
- Academically
- Professionally/Corporately
- Internationally

Personally

- What do we really want from a computer?
 - ◆ Access to our personal information
 - ◆ Minimize everyday, repetitive tasks
 - ◆ Access to the world
 - ◆ Probably not for most “human” tasks
 - ◆ Maybe for graphically oriented or transaction based systems
- What do we really want from a computer?
 - ◆ Access to our personal information
 - ◆ Minimize everyday, repetitive tasks
 - ◆ Access to the world
 - ◆ Probably not for most “human” tasks
 - ◆ Maybe for graphically oriented or transaction based systems

*Many of us don't
program anymore!*

Academically

- Speed must be important here
 - ◆ Bang for the buck
- Communications bottlenecks
 - ◆ Remotely accessed objects
- Alternative learning pedagogy
 - ◆ PowerPoint
 - ◆ The web
- Intelligence amplifier
 - ◆ Accomplish difficult or impossible tasks

Professionally/Corporately

- Transaction- and data-based computing
 - ◆ Selling products
 - ◆ Offering services
- Repository of enormous amounts of data
 - ◆ Tera and exo bytes of storage
 - ◆ Access, maintenance, and backup
- Paperless and cashless society
 - ◆ Technology is key component
 - ◆ Safety and security

Internationally

- Use of commodity parts
 - ◆ Intel architectures
 - ◆ Common software bases
- Free exchange of technology
 - ◆ Issues of control and dominance
 - ◆ High-speed global interconnects
- Maintenance of standards
 - ◆ International
 - ◆ National
 - ◆ Local

Concrete Examples

- Much has been made of the RISC/CISC debate
- How important is it?
- What were the major factors in the debate
 - ◆ Performance
 - ◆ Technology
 - ◆ Academic interest
 - ◆ Retrospective view
 - ◆ Implementation improvements

RISC - Reduced Instruction Set Computer

- Roots

- ◆ CDC 6600 (Seymour Cray)
- ◆ IBM 801 (John Cocke)
- ◆ U.C. Berkeley RISC (David Patterson)
- ◆ Stanford MIPS*(John Hennessey)

- Commercial systems

- ◆ MIPS 2000
- ◆ HP Precision Architecture minicomputers (900 series)
- ◆ IBM PC-RT and PowerPC
- ◆ Sun SPARC
- ◆ Alpha

* Microprocessor without Interlocked Pipe Stages

RISC (Cont'd)

- Microprocessors
 - ◆ AMD 29000 (1987)
 - ◆ Motorola M88000 (1988)
 - ◆ Intel i860 (1989)
- Now accepted by almost every manufacturer; even ones who formerly embraced CISC

RISC Characteristics

- No universally accepted definition
- *Most* of the following
 - ◆ Instructions are conceptually simple
 - ◆ Instructions are of a uniform length
 - ◆ Instructions use one (or very few) instruction formats
 - ◆ Instruction set is “orthogonal”
 - Little overlapping of instruction functionality
 - ◆ Instructions use very few addressing modes
 - Avoid indirection
 - ◆ Architecture is a load-and-store architecture
 - Only **LOAD** and **STORE** instructions reference memory
 - All operate instructions are register-to-register

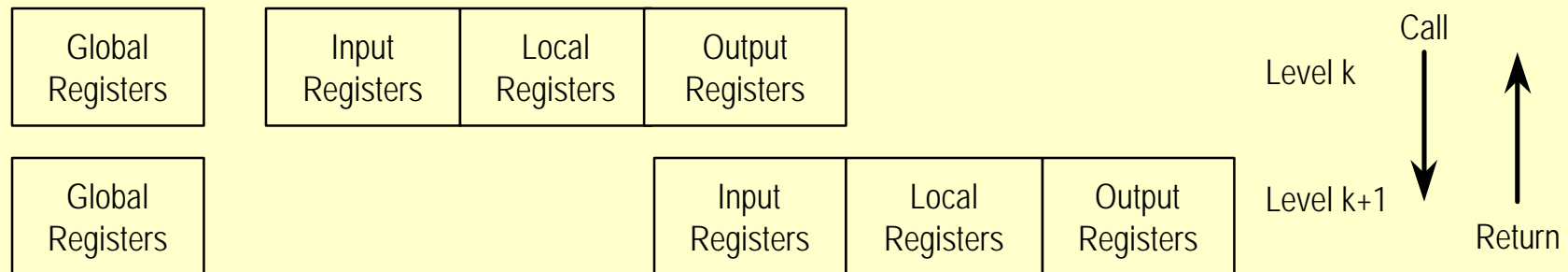
RISC Characteristics (Cont'd)

- Additional features
 - ◆ The ISA supports few data types
 - ◆ Avoid alignment problems
 - ◆ Use an MMU for a data address no more than once per instruction

RISC Characteristics (Cont'd)

- Other possible attributes

- ◆ Almost all instructions execute in one clock cycle
 - Implementation detail
- ◆ Architecture takes advantage of strengths of software
 - All reasonable architectures do
- ◆ Architecture should have many registers
 - Not part of RISC
 - Useful, however, for speeding up any CPU design



CISC Characteristics (VAX)

- Data point for comparison with RISC
 - ◆ Over 400 instructions
 - Varying length
 - Up to 6 operand specifiers
 - ◆ 16 addressing modes
 - ◆ Many data types, including
 - 6 types of integers
 - 4 types of floating point
 - Packed decimal strings
 - Character strings
 - Variable-length bit fields
 - Numeric strings

CISC Advantages

- Arguments favoring CISC advanced by proponents
 - ◆ Improved architectural merit since operations implemented in microcode execute faster than those in software
 - ◆ More complex instruction sets do not increase financial cost of implementation over simpler instruction sets
 - ◆ Upward compatibility
 - ◆ Richer instructions sets simplify compiler design
 - ◆ Smaller, faster programs
 - ◆ Complex instruction sets hinder cloning, protecting proprietary designs

RISC Advantages

- Arguments favoring RISC advanced by proponents
 - ◆ Basic hardware is simpler, thus cheaper and faster
 - ◆ Instruction caches compensate for larger number of bits in RISC instructions
 - ◆ More aggressive performance from compiler technology
 - ◆ Design effort is less, thus development cost is less
 - ◆ Easier to introduce parallelism into control unit
 - ◆ Omit things that CPU designers have found difficult and/or expensive to implement

Some Data

CPU	Age	# Instruction sizes	Max Instruction Size in Bytes	# Addressing Modes for Accessing Data	Indirect Addressing	Load/Store with Arithmetic	Max # of Memory Operands	Unaligned Addresses	Max # of MMU Uses for Data Operands	Number of Bits for Integer Register Specifier	Number of Bits for FP Register Specifier	# of Rule Exceptions
Rule	<8	=1	=4	<5	N	N	=1	Sometimes	=1	>4	>3	
AMD 29K	6	1	4	1	N	N	1	Never	1	8	3+	1
R2000	7	1	4	1	N	N	1	Never	1	5	4	0
SPARC	4	1	4	2	N	N	1	Never	1	5	4	0
MC88000	4	1	4	3	N	N	1	Never	1	5	0+	1
HP PA	7	1	4	10+	N	N	1	Never	1	5	4	1
IBM RT/PC	7	2+	4	1	N	N	1	Never	1	4+	3+	3
IBM RS/6000	3	1	4	4	N	N	1	Sometimes	1	5	5	0
Intel i860	4	1	4	4	N	N	1	Never	1	5	4	0
IBM 3090	28	4	8	2*	N*	Y	2	Any time	4	4	2	2
Intel i486	14	12	12	15	N*	Y	2	Any time	4	3	3	1
NSC 32016	12	21	21	23	Y	Y	2	Any time	4	3	3	0
MC 68040	13	11	22	44	Y	Y	2	Any time	8	4	3	0
VAX	15	56	56	22	Y	Y	6	Any time	24	4	0	0
		Note:	Architecture does not obey rule (+ for RISC; * for CISC)									

Table data from John Mashey

Where Are the Biggest Payoffs?

- ILP/VLIW
 - ◆ Increasing inherent parallelism
- MINs and connectivity
 - ◆ Routing and media delays
- Scalable processors
 - ◆ Price increase linear with increasing complexity
- Large collections of autonomous processors working independently
 - ◆ Systolic array style