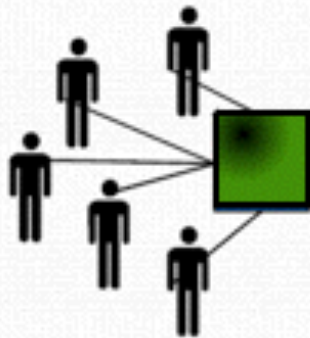


Disciplina

Sistemas de Computação

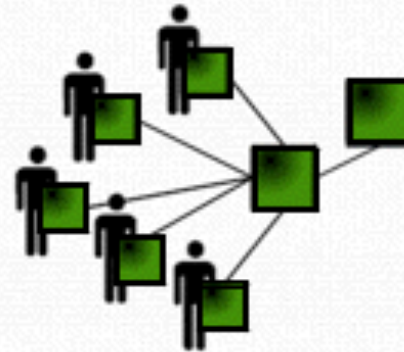
Aula 04



1950 : Mainframe



1980: Micro computer



1990: Internet



200? Diffuse IT

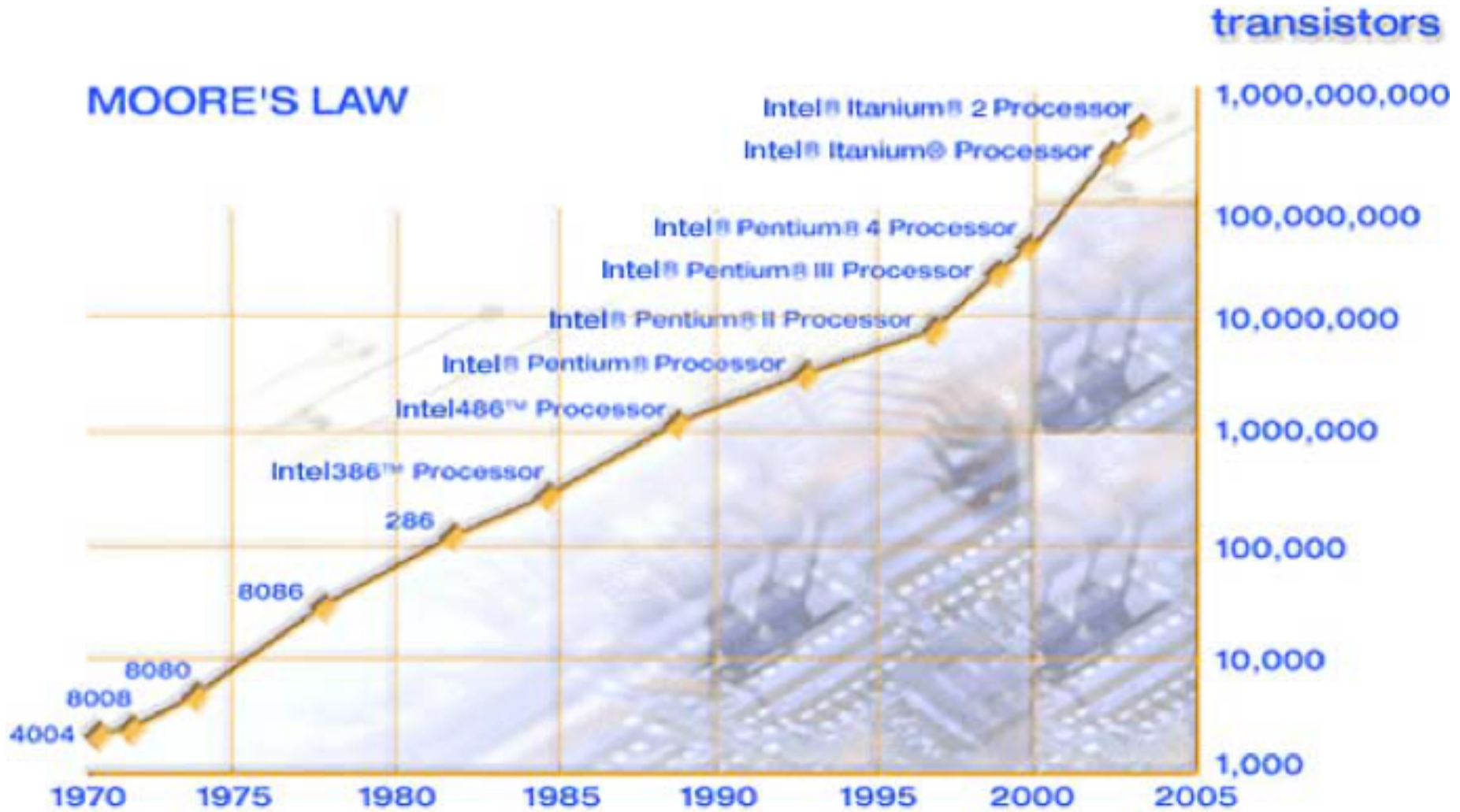
Aviso

- Slides e Arquivos já estão no site
 - <http://www.ic.uff.br/~arocha/SC>
- Hoje: Prova Supresa 1

What is Computer Architecture?

- The science and art of designing, selecting, and interconnecting hardware components and designing the hardware/software interface to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals.

An Enabler: Moore's Law

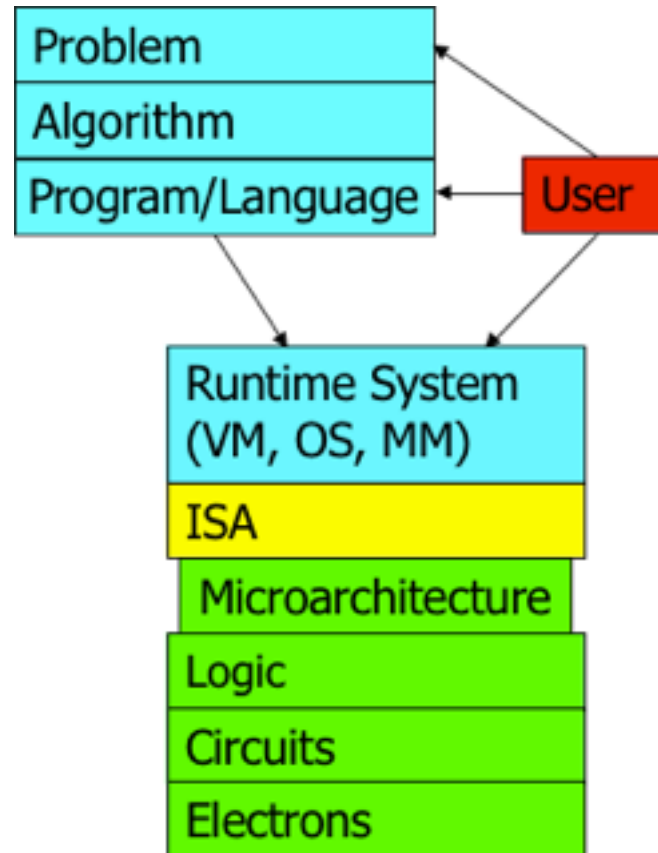


Computer Architecture Today (I)

- Today is a very exciting time to study computer architecture
- Industry is in a large paradigm shift (to multi-core and beyond) – many different potential system designs possible
- Many difficult problems motivating and caused by the shift
 - Power/energy constraints
 - Complexity of design -> multi-core?
 - Difficulties in technology scaling -> new technologies?
 - Memory wall/gap
 - Reliability wall/issues
 - Programmability wall/problem
- No clear, definitive answers to these problems

Computer Architecture Today (II)

- These problems affect all parts of the computing stack – if we do not change the way we design systems



- No clear, definitive answers to these problems

Computer Architecture Today (III)

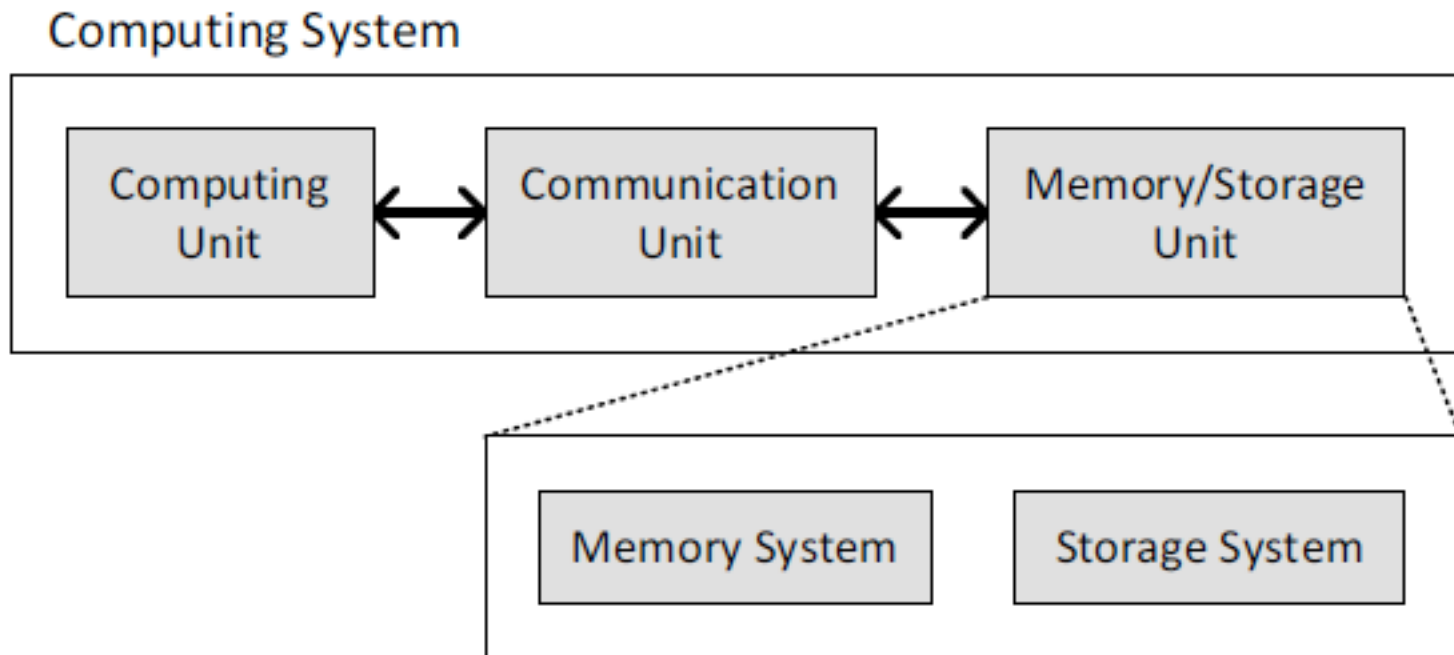
- You can revolutionize the way computers are built, if you understand both the hardware and the software (and change each accordingly)
- You can invent new paradigms for computation, communication, and storage

... but, first ...

- Let's understand the fundamentals...
- Because, you can change the world only if you understand it well enough...
 - Especially the past and present dominant paradigms
 - And, their advantages and shortcomings -- tradeoffs

What is a computer?

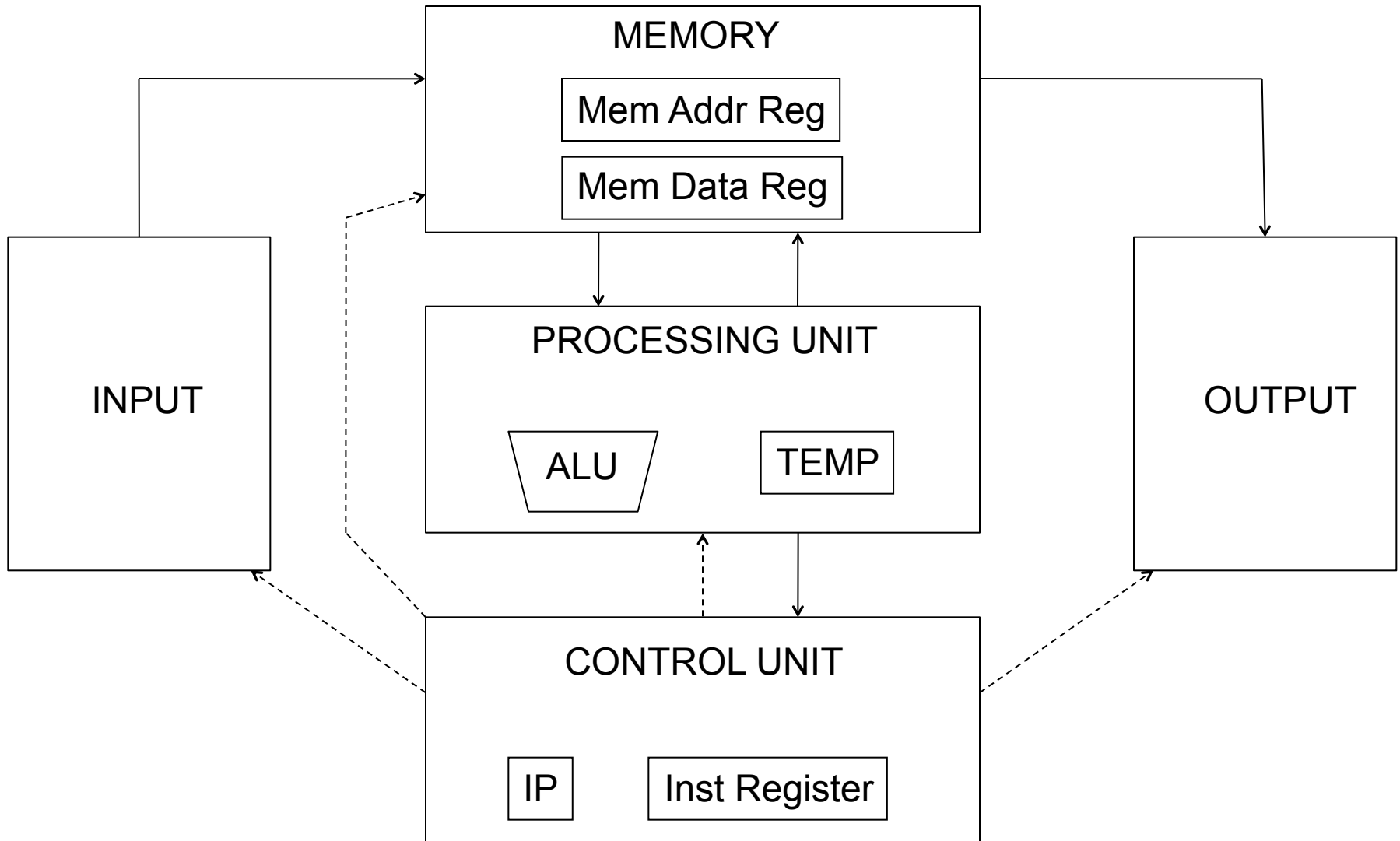
- Three key components
- Computation
- Communication
- Storage (memory)



The Von Neumann Model/Architecture

- Also called stored program computer (instructions in memory). Two key properties:
- Stored program
 - Instructions stored in a linear memory array
 - Memory is unified between instructions and data
 - The interpretation of a stored value depends on the control signals
 - When is a value interpreted as an instruction?
- Sequential instruction processing
 - One instruction processed (fetched, executed, and completed) at a time
 - Program counter (instruction pointer) identifies the current instr.
 - Program counter is advanced sequentially except for control transfer instructions

The Von-Neumann Model (of a Computer)



Aside: Dataflow Model (of a Computer)

- Von Neumann model: An instruction is fetched and executed in **control flow order**
 - As specified by the **instruction pointer**
 - Sequential unless explicit control flow instruction
- Dataflow model: An instruction is fetched and executed in **data flow order**
 - i.e., when its operands are ready
 - i.e., there is **no instruction pointer**
 - Instruction ordering specified by data flow dependence
 - Each instruction specifies “who” should receive the result
 - An instruction can “fire” whenever all operands are received
 - Potentially many instructions can execute at the same time
 - Inherently more parallel

Aside: von Neumann vs Dataflow

- Consider a von Neumann program
 - What is the significance of the program order?
 - What is the significance of the storage locations?

```
v <= a + b;  
w <= b * 2;  
x <= v - w  
y <= v + w  
z <= x * y
```

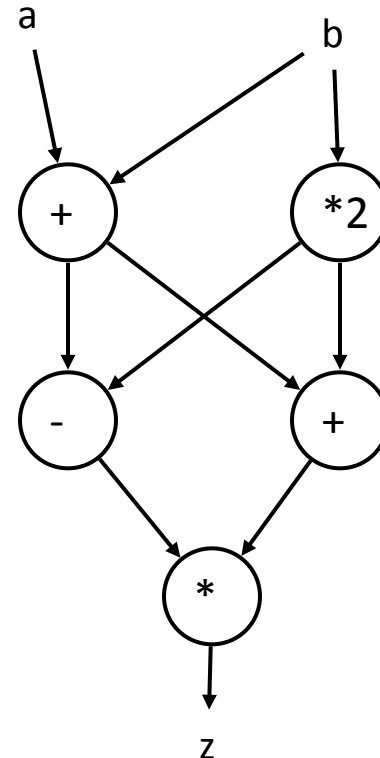
Sequential

Aside: von Neumann vs Dataflow

- Consider a von Neumann program
 - What is the significance of the program order?
 - What is the significance of the storage locations?

```
v <= a + b;  
w <= b * 2;  
x <= v - w  
y <= v + w  
z <= x * y
```

Sequential



Dataflow

- Which model is more natural to you as a programmer?

ISA-level Tradeoff: Instruction Pointer

- Do we need an instruction pointer in the ISA?
 - Yes: Control-driven, sequential execution
 - An instruction is executed when the IP points to it
 - IP automatically changes sequentially
 - (except for control flow instructions)
 - No: Data-driven, parallel execution
 - An instruction is executed when all its operand values are available ([data flow](#))
- Tradeoffs: MANY high-level ones
 - Ease of programming (for average programmers)?
 - Ease of compilation?
 - Performance: Extraction of parallelism?
 - Hardware complexity?

ISA vs. Microarchitecture Level Tradeoff

- A similar tradeoff (control vs. data-driven execution) can be made at the microarchitecture level
- ISA: Specifies how the programmer sees instructions to be executed
 - Programmer sees a sequential, control-flow execution order vs.
 - Programmer sees a data-flow execution order
- Microarchitecture: How the underlying implementation actually executes instructions
 - Microarchitecture can execute instructions in any order as long as it obeys the semantics specified by the ISA when making the instruction results visible to software
 - Programmer should see the order specified by the ISA

The Von-Neumann Model

- All major **instruction set architectures** today use this model
 - x86, ARM, MIPS, SPARC, Alpha, POWER
- Underneath (at the microarchitecture level), the execution model of almost all **implementations (or, microarchitectures)** is very different
 - Pipelined instruction execution: Intel 80486 uarch
 - Multiple instructions at a time: Intel Pentium uarch
 - Out-of-order execution: Intel Pentium Pro uarch
 - Separate instruction and data caches
- But, what happens underneath that is not consistent with the von Neumann model is not exposed to software
 - Difference between ISA and microarchitecture

What is Computer Architecture?

- **ISA+implementation definition:** The science and art of designing, selecting, and interconnecting hardware components and designing the hardware/software interface to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals.
- **Traditional (only ISA) definition:** “The term **architecture** is used here to describe the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behavior as distinct from the organization of the dataflow and controls, the logic design, and the physical implementation.”
Gene Amdahl, IBM Journal of R&D, April 1964

ISA vs. Microarchitecture

■ ISA

- Agreed upon interface between software and hardware
 - SW/compiler assumes, HW promises
- What the software writer needs to know to write and debug system/user programs

■ Microarchitecture

- Specific implementation of an ISA
- Not visible to the software

■ Microprocessor

- **ISA, uarch**, circuits
- “Architecture” = ISA + microarchitecture

Problem
Algorithm
Program
ISA
Microarchitecture
Circuits
Electrons

ISA vs. Microarchitecture

- What is part of ISA vs. uarch?
 - Gas pedal: interface for “acceleration”
 - Internals of the engine: implement “acceleration”
- Implementation (uarch) can be various as long as it satisfies the specification (ISA)
 - Add instruction vs. Adder implementation
 - Bit serial, ripple carry, carry lookahead adders are all part of microarchitecture
 - x86 ISA has many implementations: 286, 386, 486, Pentium, Pentium Pro, Pentium 4, Core, ...
- Microarchitecture usually changes faster than ISA
 - Few ISAs (x86, ARM, SPARC, MIPS, Alpha) but many uarchs
 - Why?

ISA

■ Instructions

- ❑ Opcodes, Addressing Modes, Data Types
- ❑ Instruction Types and Formats
- ❑ Registers, Condition Codes

■ Memory

- ❑ Address space, Addressability, Alignment
- ❑ Virtual memory management

■ Call, Interrupt/Exception Handling

■ Access Control, Priority/Privilege

■ I/O: memory-mapped vs. instr.

■ Task/thread Management

■ Power and Thermal Management

■ Multi-threading support, Multiprocessor support



Intel® 64 and IA-32 Architectures
Software Developer's Manual

Volume 1:
Basic Architecture

Microarchitecture

- Implementation of the ISA under specific **design constraints and goals**
- Anything done in hardware without exposure to software
 - Pipelining
 - In-order versus out-of-order instruction execution
 - Memory access scheduling policy
 - Speculative execution
 - Superscalar processing (multiple instruction issue?)
 - Clock gating
 - Caching? Levels, size, associativity, replacement policy
 - Prefetching?
 - Voltage/frequency scaling?
 - Error correction?

Property of ISA vs. uarch?

- ADD instruction's opcode
- Number of general purpose registers
- Number of ports to the register file
- Number of cycles to execute the MUL instruction
- Whether or not the machine employs pipelined instruction execution

- Remember
 - Microarchitecture: Implementation of the ISA under specific **design constraints and goals**

Design Point

- A set of design considerations and their importance
 - **leads to tradeoffs** in both ISA and uarch
- Considerations
 - Cost
 - Performance
 - Maximum power consumption
 - Energy consumption (battery life)
 - Availability
 - Reliability and Correctness
 - Time to Market
- Design point determined by the “Problem” space (application space), or the intended users/market

Problem
Algorithm
Program
ISA
Microarchitecture
Circuits
Electrons