

Computer Architecture

Lecture 2: Fundamental Concepts and ISA

Dr. Ahmed Sallam

Based on original slides by Prof. Onur Mutlu

What Do I Expect From You?

"Chance favors the prepared mind."



(Louis Pasteur)

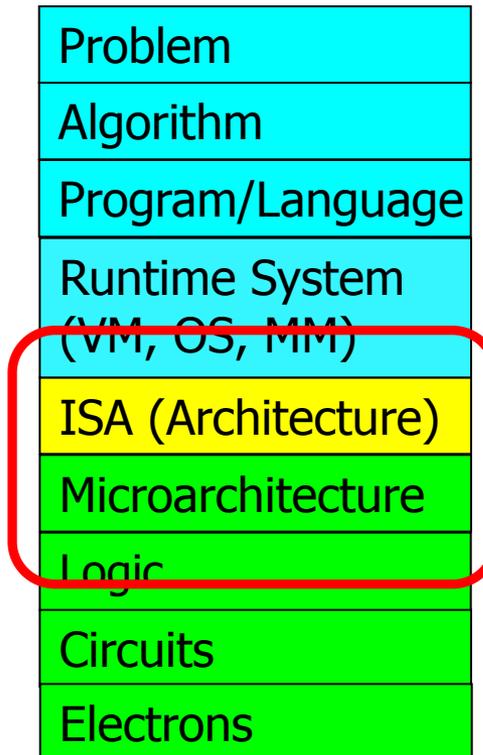
"كل عامل ميسر لعمله" (محمد، صلي الله عليه وسلم)

Agenda for Today

- Review
- Why study computer architecture?
- ISA
- Microarchitecture
- Computer Architecture
- Next Lecture

REVIEW

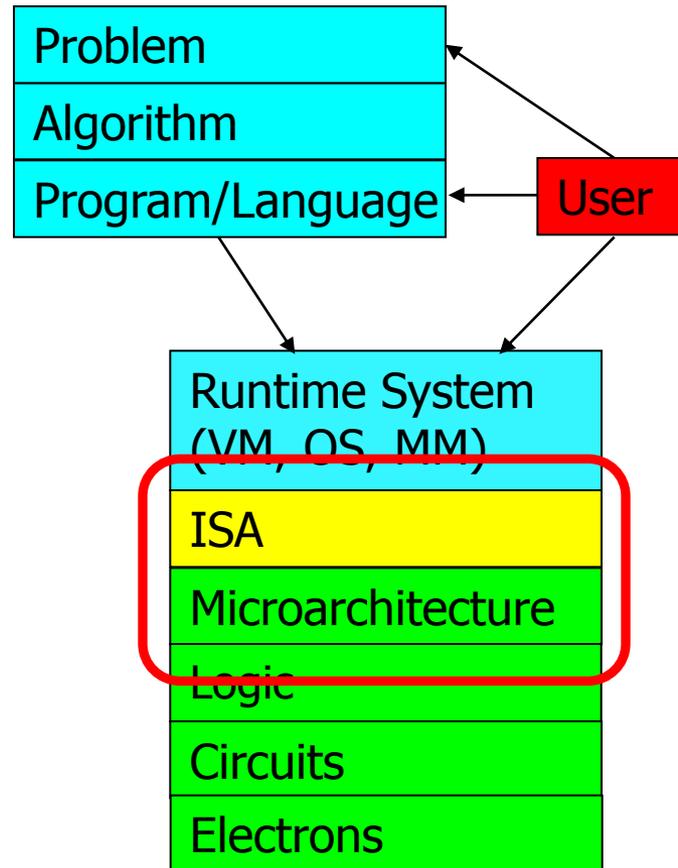
Review: Comp. Arch. in Levels of Transformation



- Read: Patt, "Requirements, Bottlenecks, and Good Fortune: Agents for Microprocessor Evolution," Proceedings of the IEEE 2001.

Review: Levels of Transformation, Revisited

- A user-centric view: computer designed for users

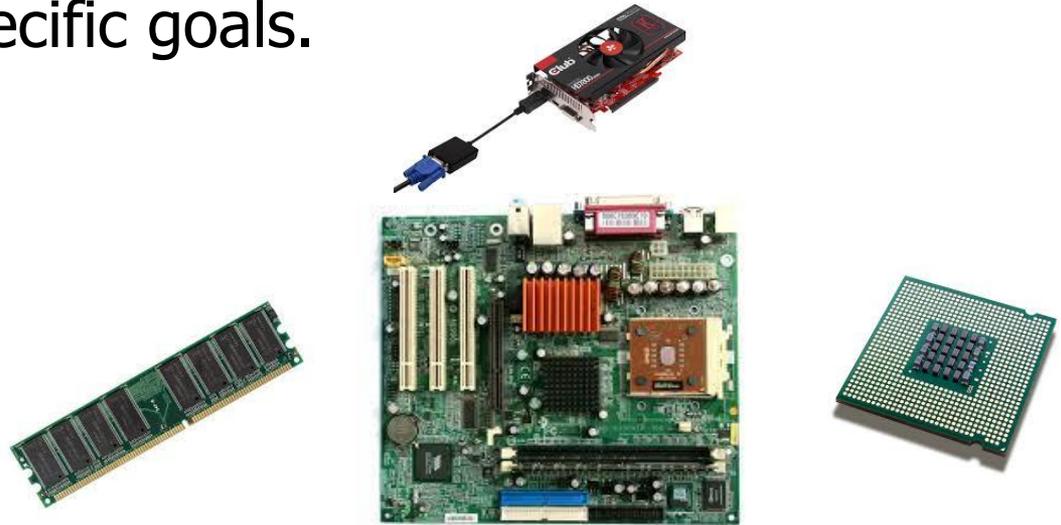


- The entire stack should be optimized for user

WHY STUDY COMPUTER ARCHITECTURE?

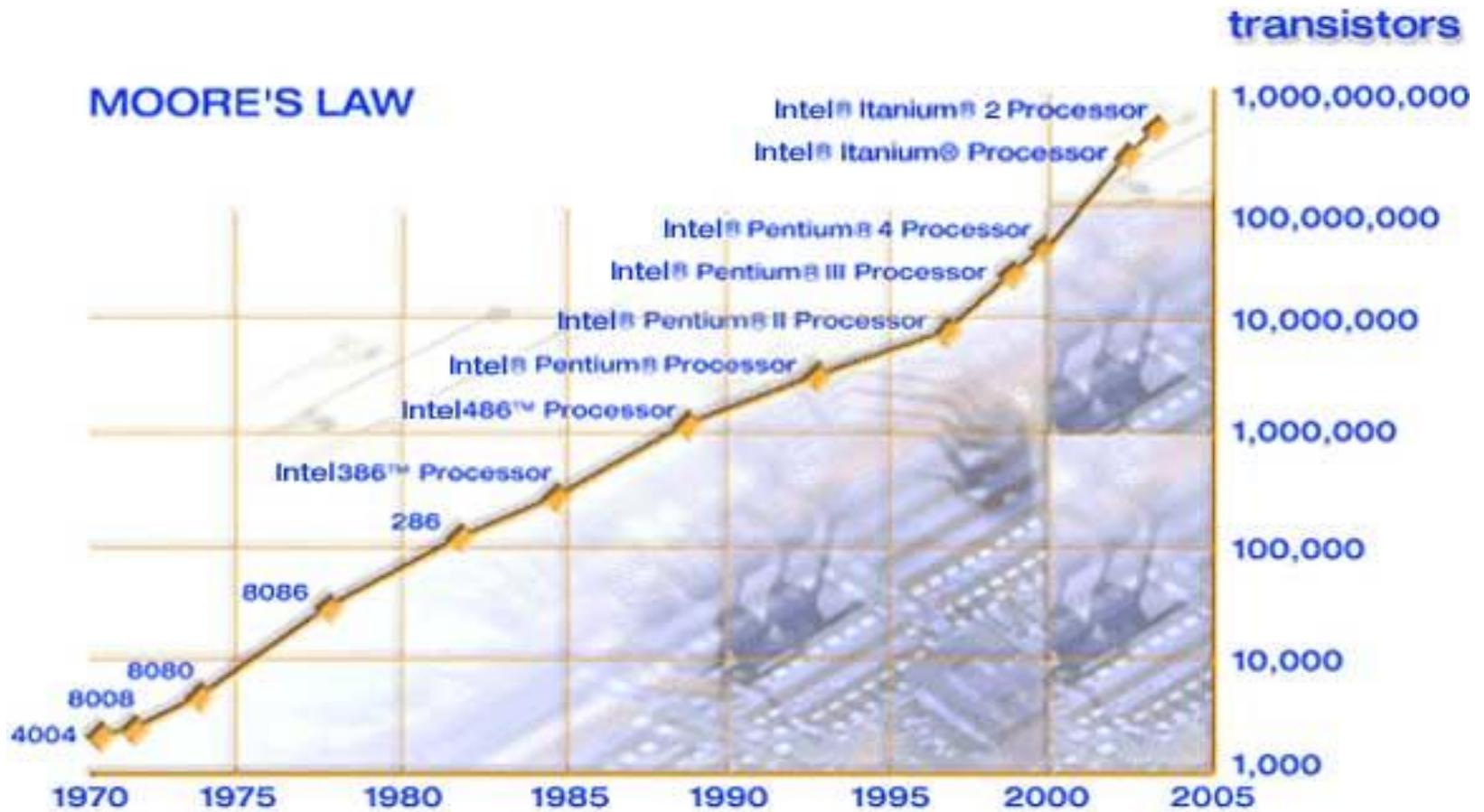
Why study Computer Architecture?

- 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.



- We will soon distinguish between the terms *architecture*, and *microarchitecture*.

An Enabler: Moore's Law



Moore, “Cramming more components onto integrated circuits,”
Electronics Magazine, 1965. Component counts double every other year

What Do We Use These Transistors for?

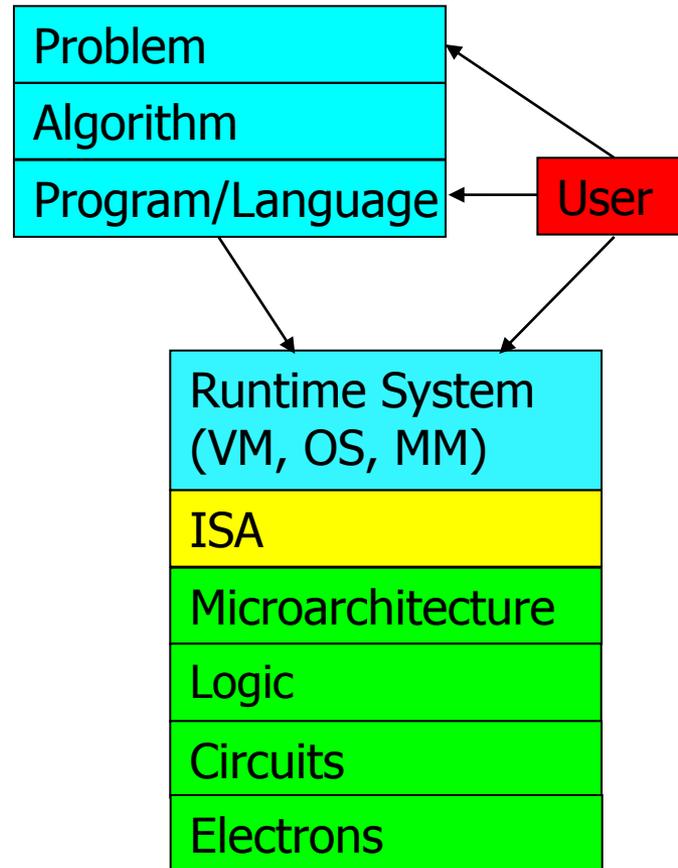
- Your readings for this week should give you an idea...
- Patt, "Requirements, Bottlenecks, and Good Fortune: Agents for Microprocessor Evolution," Proceedings of the IEEE 2001.

Computer Architecture Today

- 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



- You can invent new paradigms for computation, communication, and storage

Course Goals

- Goal 1: To familiarize those interested in computer system design with both **fundamental operation** principles and **design tradeoffs** of processor, memory, and platform architectures in today's systems.
 - Strong emphasis on fundamentals and design tradeoffs.
- Goal 2: To provide the necessary background and experience to **design, implement, and evaluate** a modern processor by performing hands-on RTL and C-level implementation.
 - Strong emphasis on functionality and hands-on design.
- Goal 3: Think **critically** (in solving problems), and **broadly** across the levels of transformation

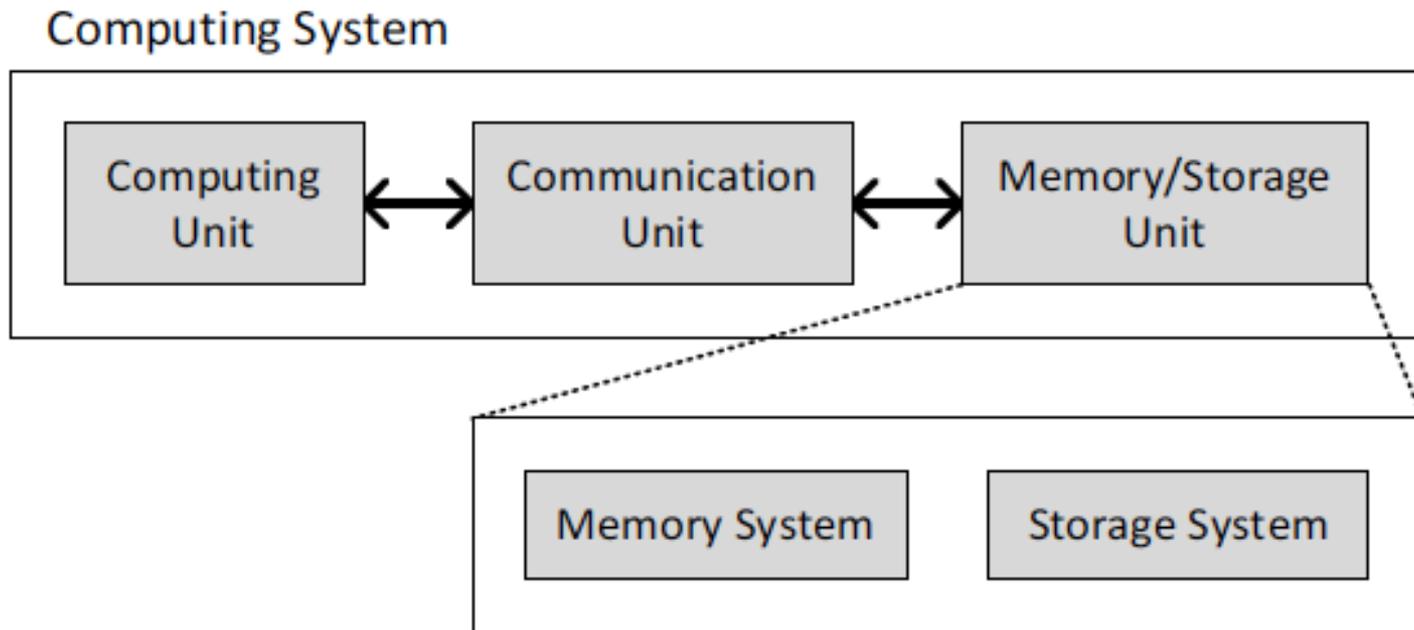
... but, first ...

- Let's understand the fundamentals...
- 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

INSTRUCTION SET ARCHITECTURE (ISA)

What is A Computer?

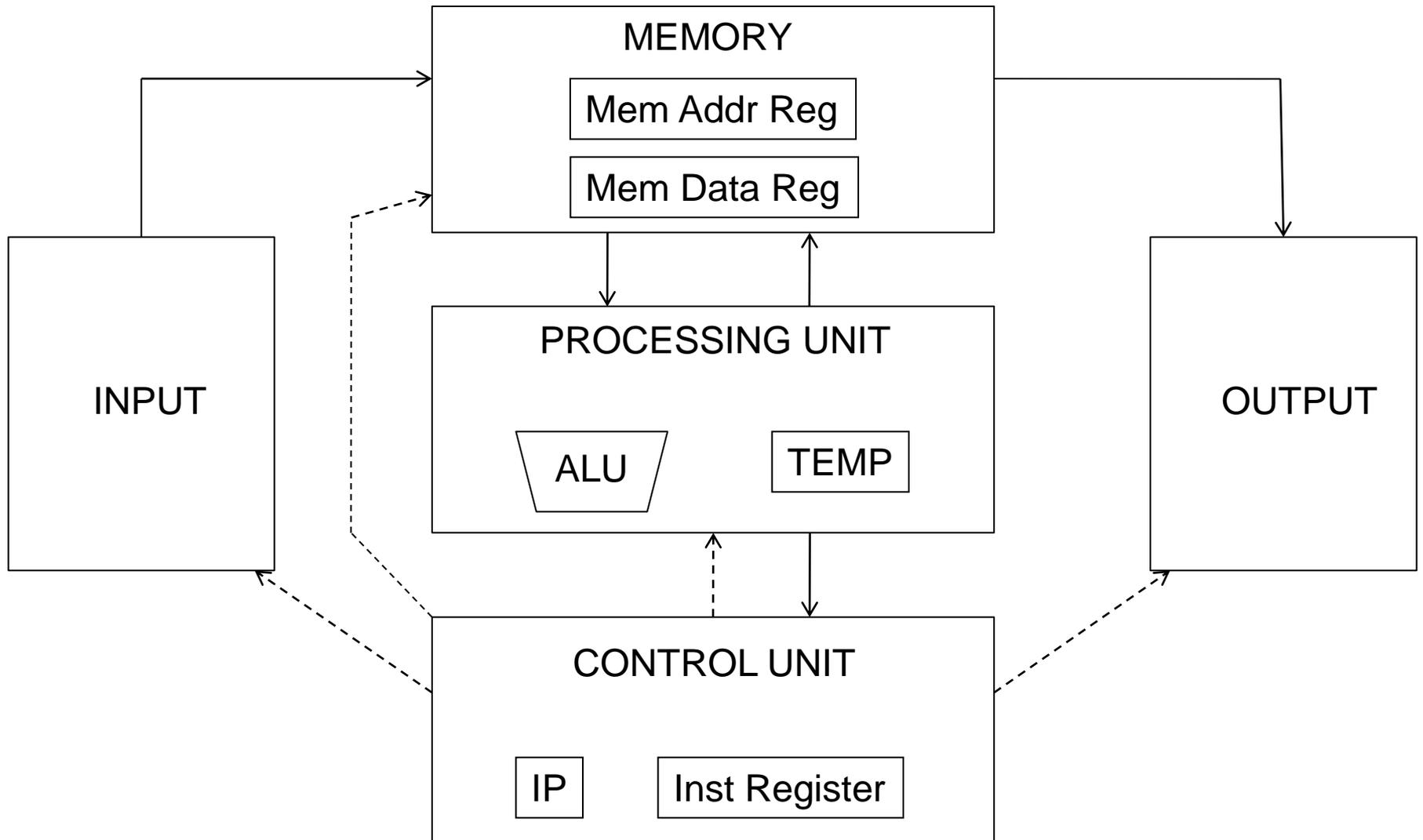
- Three key components
- Computation
- Communication
- Storage (memory)



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)



The Dataflow Model (of a Computer)

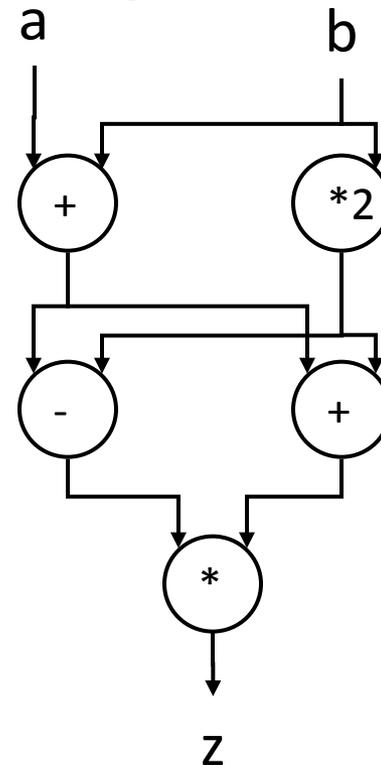
- Von Neumann model: An instruction is fetched and executed in **control flow order**
 - As specified by the **instruction pointer**
 - Sequential unless explicit control flow instruction
- Dataflow model: An instruction is fetched and executed in **data flow order**
 - i.e., when its operands are ready
 - 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

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

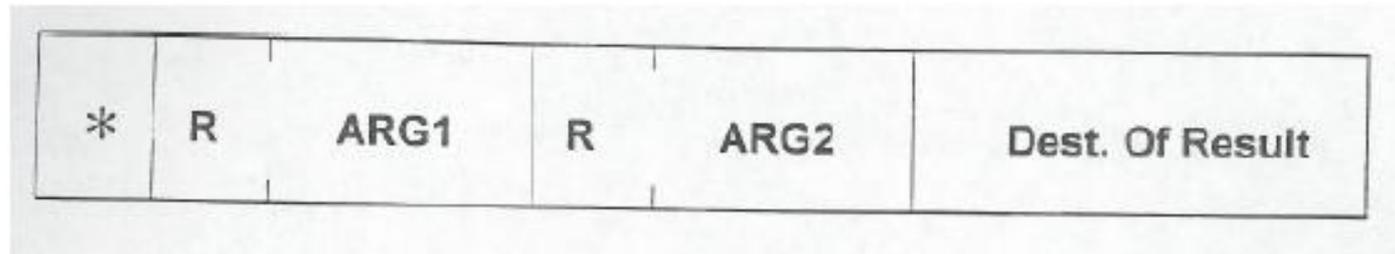
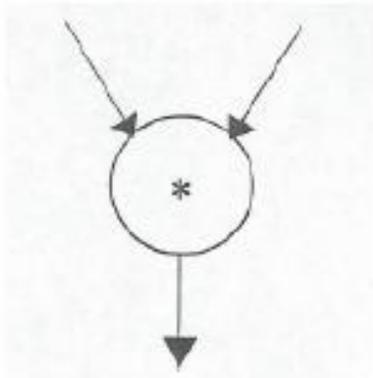


Dataflow

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

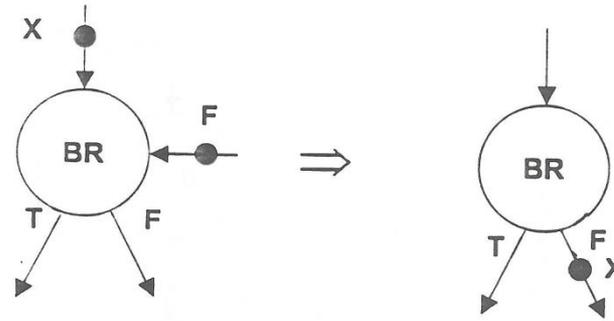
More on Data Flow

- In a data flow machine, a program consists of data flow nodes
 - A data flow node fires (fetched and executed) when all its inputs are ready
 - i.e. when all inputs have tokens
- Data flow node and its ISA representation

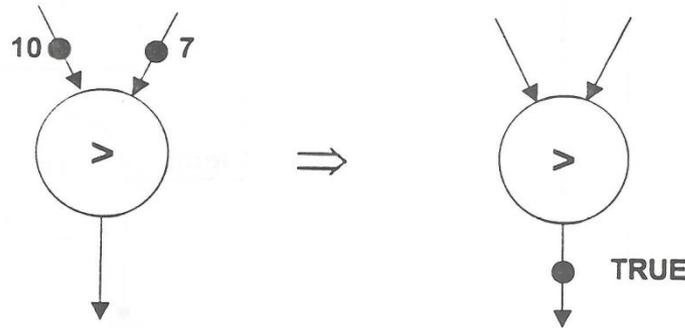


Data Flow Nodes

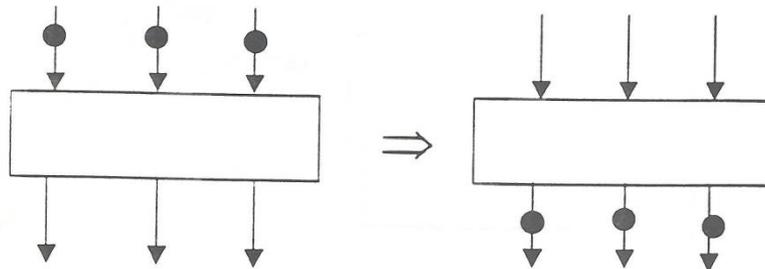
***Conditional**



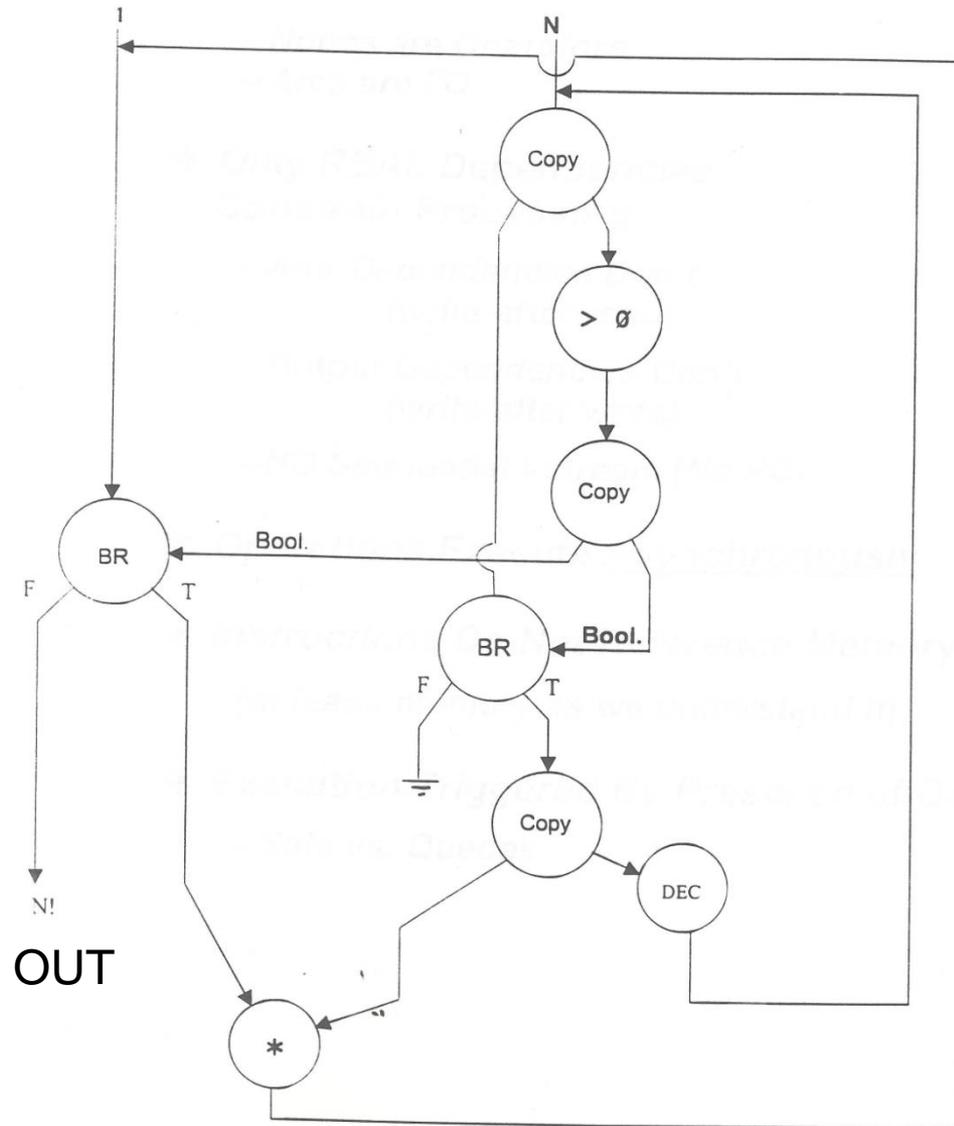
***Relational**



***Barrier Synchron**



An Example Data Flow Program



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?

Remember

- Von Neumann and Data-flow is just **models**
- All major **instruction set architectures** today use this Von Neumann model
 - x86, ARM, MIPS, SPARC, Alpha, POWER

MICROARCHITECTURE

ISA vs. Microarchitecture

- What is part of ISA vs. Uarch?
 - Gas pedal: interface for “acceleration”
 - Internals of the engine: implement “acceleration”
- ISA
 - Agreed upon interface between software and hardware (SW/compiler assumes, HW promises)
 - What programmer needs to know to write and debug system/user programs and to specify to user a sequential control-flow or a data-flow execution order.
- Microarchitecture
 - Specific implementation of an ISA
 - Not visible to the software or the programmer
- Microprocessor
 - **ISA, uarch**, circuits
 - “Architecture” = ISA + microarchitecture

Problem
Algorithm
Program
ISA
Microarchitecture
Circuits
Electrons

ISA Example

■ 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

Microarchitecture Example

- 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
- 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 Tradeoff: control vs. data driven

- A similar tradeoff (control vs. data-driven execution) can be made at the microarchitecture level
- 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

Out of order (memory reordering)
in a modern (Super Scaler) processor that can execute 2 or
more instruction at once
I'm talking about single core processor not just multi core
processor

1) mov eax, 0	solution 1: (1,2)->(3)->(4,5)
2) mov edx, 1	
3) mov ecx, 3	solution 2: (1,2)->(3,5)->(4,..)
4) inc edx	
5) mov ecx, 3	The transistors only decide which solution to perform
..)	

Review Questions (ISA or 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
- Microarchitecture usually changes faster than ISA
 - Few ISAs (x86, ARM, SPARC, MIPS, Alpha) but many uarchs
 - *Why?*
 - *Abstraction!*

COMPUTER ARCHITECTURE

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*