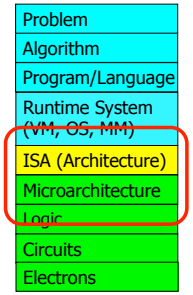
Computer Architecture
Fundamentals

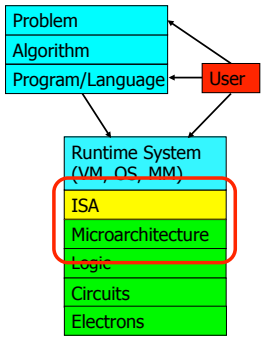
Computer Architecture in Levels of Transformation



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

Levels of Transformation, Revisited

- A user-centric view: computer designed for users

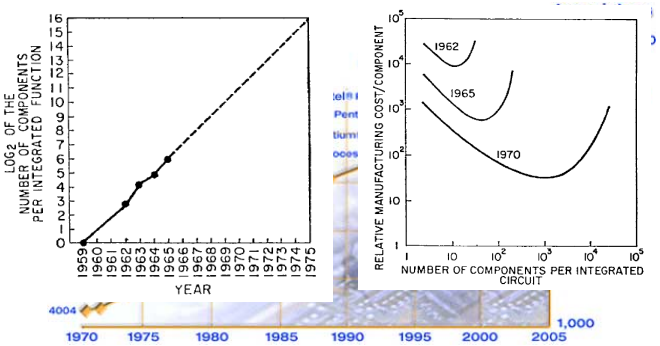


- The entire stack should be optimized for user

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

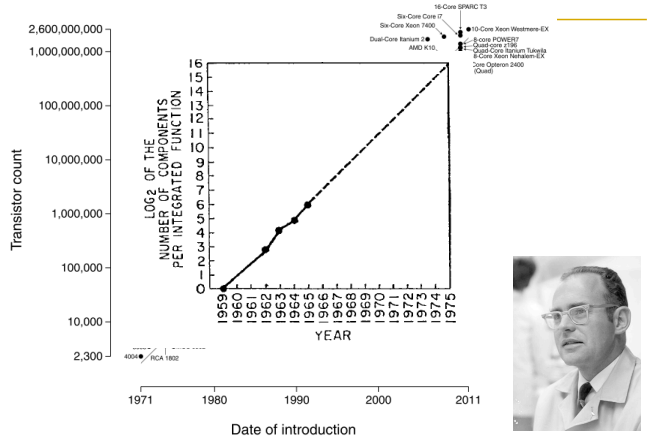
An Enabler: Moore's Law



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

Image source: Intel

Microprocessor Transistor Counts 1971-2011 & Moore's Law



Number of transistors on an integrated circuit doubles ~ every two years

Image source: Wikipedia

Why Study Computer Architecture?

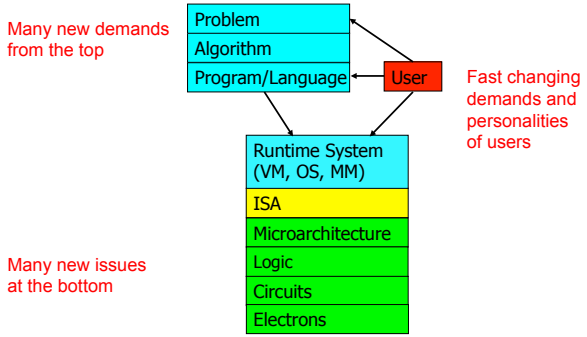
- Enable better systems: make computers faster, cheaper, smaller, more reliable, …
  - By exploiting advances and changes in underlying technology/circuits
- Enable new applications
  - Life-like 3D visualization 20 years ago? Virtual reality?
  - Self-driving cars?
  - Personalized genomics? Personalized medicine?
- Enable better solutions to problems
  - Software innovation is built on trends and changes in computer architecture
- Understand why computers work the way they do

Computer Architecture Today

- 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 → multi-core ?
  - Complexity of design → multi-core ?
  - Difficulties in technology scaling → new technologies?
  - Memory wall/gap → processing in memory?
  - Reliability wall/issues → new technologies?
  - Programmability wall/problem
  - Huge hunger for data and new data-intensive applications
- No clear, definitive answers to these problems

Computer Architecture Today

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

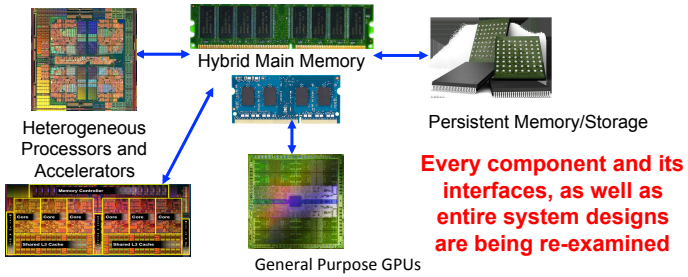


- No clear, definitive answers to these problems

9

Computer Architecture Requirements

- Both (software and humanity trends) and (technologies and their issues), and the resulting requirements and constraints



Every component and its interfaces, as well as entire system designs are being re-examined

10

Computer Architecture Focus

- 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

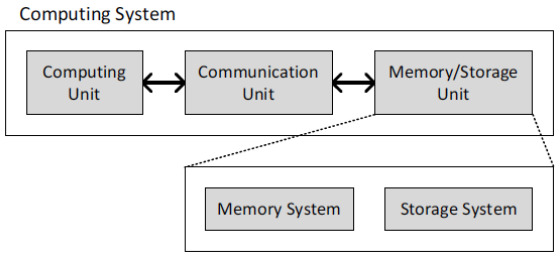
11

Fundamental Concepts

12

What is A Computer?

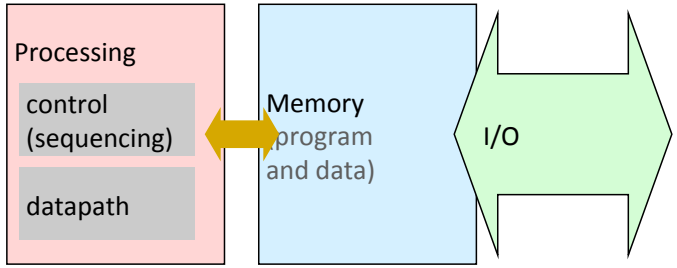
- Three key components
- Computation
- Communication
- Storage (memory)



13

What is A Computer?

- We will cover all three components



14

The Von Neumann Model/Architecture

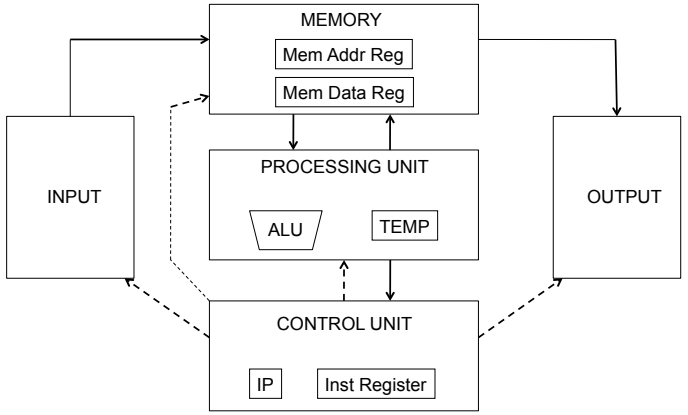
- Also called *stored program computer* (instructions in memory). Two key properties:
- Stored program
  - Instructions stored in a linear memory array
  - Memory is unified between instructions and data
    - The interpretation of a stored value depends on the control signals

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

15

The Von Neumann Model (of a Computer)



16

The Von Neumann Model (of a Computer)

- Q: Is this the only way that a computer can operate?
- A: No.
- Qualified Answer: No, but it has been the dominant way
  - i.e., the dominant paradigm for computing
  - for N decades

17

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

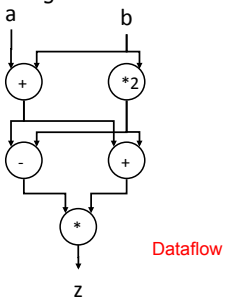
18

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



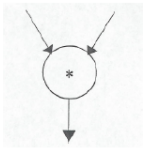
Dataflow

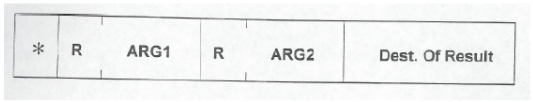
- Which model is more efficient to you as a programmer?

19

More on Data Flow

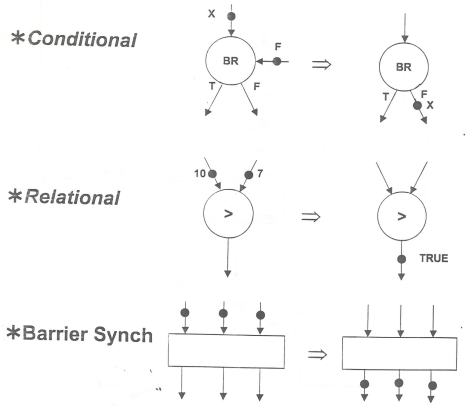
- In a data flow machine, a program consists of data flow nodes
  - A data flow node fires (fetched and executed) when all it inputs are ready
    - i.e. when all inputs have tokens
- Data flow node and its ISA representation





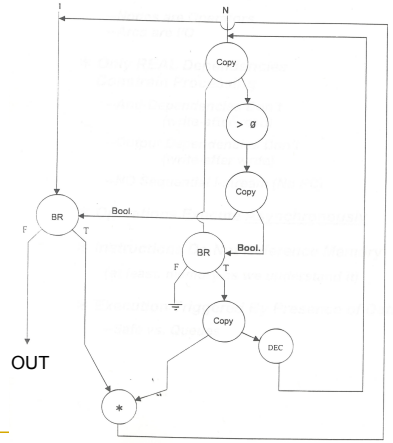
20

Data Flow Nodes



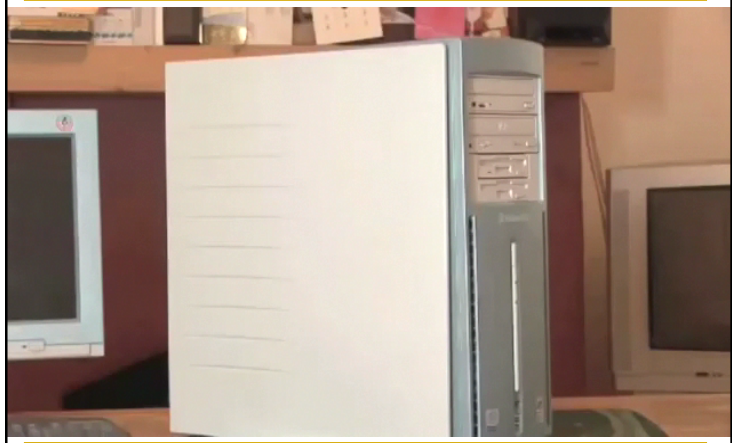
21

An Example Data Flow Program

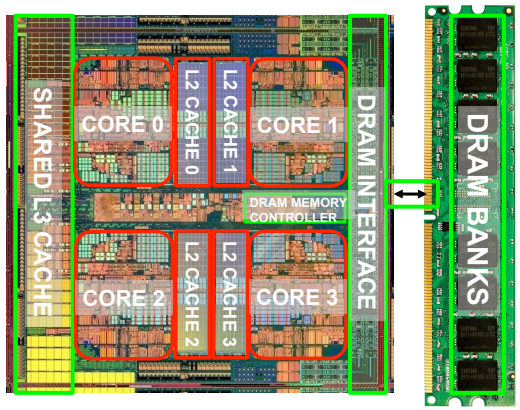


22

Inside a Computer System



A Modern System



24

source:

Computer Architecture
Lecture 1: Introduction and Basics

Prof. Onur Mutlu
ETH Zurich
Fall 2017
20 September 2017