# CSE 220: Systems Fundamentals I Unit 1: Introduction to Digital Systems



## The Landscape of Computing

- Three broad classes of modern computing devices:
  - **Personal computers**: general-purpose, low-performing machines
  - **Servers and supercomputers**: more powerful machines usually built for a particular purpose
    - Modern supercomputers typically have 10,000+ processing units and terabytes of memory
  - **Embedded computers**: special-purpose computers that are *embedded* in a larger system, such as a car, TV, network router, etc.
- For which of these computing platforms will fault tolerance be the least? (i.e., for which computer type is a failure totally unacceptable?)



## But Wait, There's More!

- Harder to categorize are devices like smart phones and tablets
  - Somewhere between a personal computer and an embedded system? Or a totally different category?
- Cloud computing systems also don't fit well into these categories
  - An evolution of server technology
  - **SaaS**: software as a service – why buy an expensive server when you can rent one at a fraction of the price?

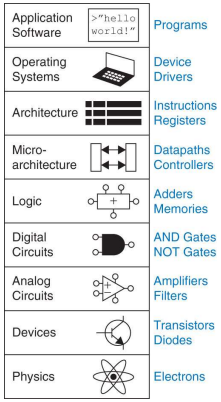


## Architecture vs. Organization

- The terms **computer architecture** and **computer organization** are often thrown about without clear explanation of what these terms mean
- **Computer architecture** refers to those aspects of the hardware that are visible to the programmer
  - e.g., instructions the computer is capable of executing, **word size** (native unit of data of CPU), data formats
- **Computer organization** (also called **microarchitecture**) refers to how the physical components of the machine interact to implement the architecture
  - A particular computer architecture could be implemented by several different microarchitectures

# Eight Great Ideas in Architecture

1. Design for Moore's Law
   - Moore's "Law" refers to the trend that circuit complexity/speed doubles every 18-24 months
   - Anticipate where technology will be when a system is completed, not when it is being designed
2. Use abstraction to simplify design
   - Abstraction refers to ignoring irrelevant details and focusing on higher-level design/implementation issues
   - Part of software development too






# Eight Great Ideas in Architecture

3. Make the common case fast
   - Enhance the performance of those operations which occur most frequently
4. Increase performance via parallelism
   - Perform operations in parallel (simultaneously) when possible
5. Increase performance via pipelining
   - A form of parallelism in which single instructions are broken into multiple stages and executed in parallel
6. Increase performance via prediction
   - The computer "guesses" which operation will be executed next and starts executing it early



# Eight Great Ideas in Architecture

7. Implement a hierarchy of memories
   - Fastest, smallest and expensive memory at the top; slowest, largest and cheapest at the bottom
8. Increase dependability via redundancy
   - Include redundant components that can take over when a failure occurs
   - Of particular importance in cloud computing systems and other server technologies



# The Three Y's

1. Hierarchy
   - We will look at the design of a CPU from the bottom-up as a collection of collaborating modules
2. Modularity
   - These modules have well-defined functions and interfaces
3. Regularity
   - Modules can be reused many times in a single design, thereby reducing then umber of distinct components that must be designed

- Note how well these concepts apply also to software design and implementation

# von Neumann Architecture

- The term **von Neumann architecture** refers to a particular computer hardware design model for a **stored-program** digital computer (e.g., PCs)
- Separate **central processing unit (CPU)** and **random-access memory (RAM)**
- Both **instructions** and **data** stored in RAM
- Data to be processed is transferred from RAM to CPU, and results are transferred back to RAM
- Named for Hungarian-American mathematician John von Neumann, but others participated in the original design
- Most modern computers follow this basic design model, but are substantially more complex



# von Neumann Architecture

- CPU: performs the actual processing
  - **control unit**: performs instruction decoding and control
  - **arithmetic/logic unit (ALU)**: performs basic arithmetical and logical operations
  - **registers**: small amount of memory used to hold addresses, instructions and data
- RAM: larger memory used to store both program instructions and data
- I/O devices permit data to enter/leave machine
  - We'll concern ourselves with **standard input**, **standard output**, **standard error** and also file I/O



# Stored-program Computer

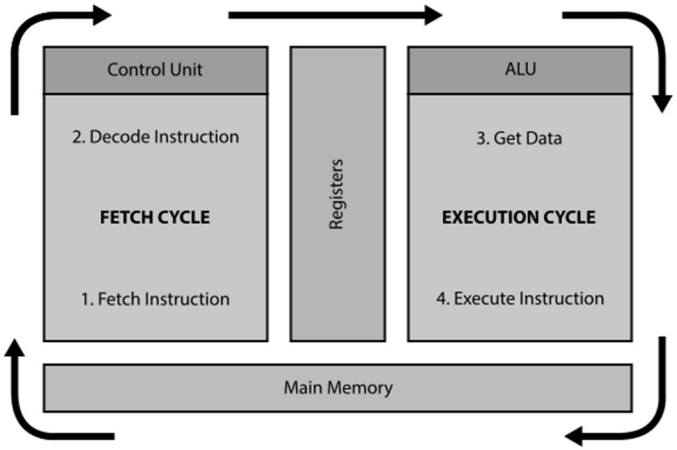
- In a stored-program computer, the program to be executed is stored in RAM along with the data to be processed
  - So please note: by saying "stored program" it does NOT mean we are storing the programs on a disk
- A program consists of binary **instructions** stored in RAM
- Each instruction or small piece of data in RAM has an associated **memory address** to indicate its location
- A **program counter** (or **instruction pointer**) register in the CPU stores the memory address of the next instruction to be executed
- Earlier computers were "hard-wired" to a particular program via switches, patch cables, etc.



# Fetch/Decode/Execute Cycle

- So what is a 3 GHz CPU doing 3 billion times per second?
- The basic cycle of operation of a von Neumann-style computer:
  - **Fetch**: the next instruction is retrieved from RAM
  - **Decode**: the instruction is examined to determine what the CPU should do:
    - **Opcode**: field that determines the type of instruction
    - **Operand(s)**: fields that determine the source and destination of data to be operated on
  - **Execute**: the operation specified by the instruction is performed, which may involve one or more memory references

# Fetch/Decode/Execute Cycle






# Instruction Set

- The **instruction set** of a computer is the repertoire of instructions that the CPU can perform
  - Determined by the computer architects/designers
  - "Hard-wired" as part of the computer design
  - Different for each type of CPU
- What is an "instruction"?
- Answer: it depends on the kind of CPU
- In **RISC** (reduced instruction set computer) CPUs, the instruction set is small and consists of simple instructions
- In **CISC** (complex instruction set computer) CPUs, the instruction set is larger and consists of instructions that vary in length and complexity



# Typical RISC Instructions

- In this course we will study the **MIPS** architecture (Microprocessor without Interlocked Pipeline Stages), which is of the RISC type. Instructions include:
  - **Load** data from memory to CPU register
  - Copy ("**Store**") data from CPU register to memory
  - **Add**, **subtract**, **multiply**, **divide**, etc. data in CPU registers
  - **AND**, **OR**, **XOR**, **NOT**, etc. data in CPU registers
  - **Shift** and **rotate** data in CPU registers
  - **Jump** based on CPU state flags ("condition codes")
  - **Call** a subroutine (function) and **return** to caller



# Example

- A **high-level programming language** statement like X = A + B would be translated to assembly language instructions as follows:
  - LOAD A
  - LOAD B
  - ADD C, A, B
  - STORE C, X
- Naturally, the real assembly instructions are a little more sophisticated than this (e.g., from what memory address are loading the data?), but the beauty of RISC machines is that the instructions are quite simple
- The assembly language instructions are then translated into machine language (1s and 0s) for execution by the CPU

# Course Roadmap

- Data representations: numbers, characters
- MIPS assembly programming: basics, conditionals, loops
- MIPS assembly programming: functions, 1D arrays
- Digital logic design: basic circuit design
- Digital logic design: computational units, timing
- Digital logic design: circuit simplification
- MIPS assembly programming: 2D arrays
- MIPS architecture: single-cycle data path
- MIPS assembly programming: recursive functions
- MIPS architecture: multi-cycle data path
- MIPS architecture: pipelined data path