System Design and Computer Architecture

Understanding Modern Computing Systems

Your Name

Department of Computer Science
Your Institution

September 7, 2025

Course Overview

Welcome to System Design & Computer Architecture

- **Course Objectives:**
  - Understand fundamental computer architecture concepts
  - Learn system design principles and patterns
  - Explore modern computing technologies (RISC, ARM, x86)
  - Apply knowledge to real-world system design problems
- **What You'll Learn:**
  - CPU and memory architecture
  - Instruction set architectures (RISC vs CISC)
  - System performance optimization
  - Scalable system design patterns
- **Prerequisites:** Basic programming knowledge, digital logic fundamentals

A Computer's Anatomy

- **Objective:** Understand the fundamental components of a computer system, focusing on CPU and memory architecture.
- **Topics Covered:**
    - The Von Neumann Architecture
    - CPU Components
    - Memory Hierarchy (Cache, RAM, Permanent Storage)

Why It Matters

- **Performance:** Knowing how CPU and memory interact helps in optimizing software performance.
- **System Design:** Essential for designing efficient systems and applications.
- **Troubleshooting:** Understanding hardware can aid in diagnosing performance bottlenecks.

Computer Architecture Overview

- **Computer Architecture:** The conceptual design and fundamental operational structure of a computer system.
- **Key Components:**
  - **CPU (Central Processing Unit):** Executes instructions and processes data.
  - **Memory:** Stores data and instructions temporarily (RAM) or permanently (SSD/HDD).
  - **I/O Devices:** Facilitate interaction with the external environment (keyboard, mouse, display).
- **Data Flow:** The CPU fetches instructions from memory, processes them, and may read/write data to/from memory or I/O devices.

Types of architectures

- **Von Neumann Architecture:** Single memory space for instructions and data.
- **Harvard Architecture:** Separate memory spaces for instructions and data.
- **ARM Architecture:** A RISC-based design with a load/store model, unified memory, conditional execution, and deep pipelines; widely used in mobile and embedded systems for its power efficiency.
- **Comparison:** Von Neumann is simpler and more common, Harvard can be faster for certain applications, while ARM combines RISC efficiency with a flexible, unified memory system.

Instruction Set Architectures: RISC vs CISC

- **RISC (Reduced Instruction Set Computer):**
  - Simple, uniform instructions (typically 32-bit)
  - Load/Store architecture - only load/store access memory
  - One instruction per clock cycle (ideally)
  - More registers, simpler hardware
- **CISC (Complex Instruction Set Computer):**
  - Complex, variable-length instructions
  - Instructions can directly access memory
  - Multiple clock cycles per instruction
  - Fewer registers, more complex hardware
- **Philosophy:** RISC favors simple hardware + smart compilers, CISC favors complex hardware + simple compilers

RISC vs CISC: Design Trade-offs

RISC Advantages:

- Simpler processor design
- Lower power consumption
- Better pipelining performance
- Easier to optimize
- Higher clock speeds possible

RISC Disadvantages:

- More instructions needed
- Larger code size
- Complex compiler required

CISC Advantages:

- Fewer instructions needed
- Smaller code size
- Rich instruction set
- Backward compatibility

CISC Disadvantages:

- Complex processor design
- Higher power consumption
- Difficult to pipeline
- Slower clock speeds

ARM Architecture: The RISC Champion

- **ARM (Advanced RISC Machine):** Dominant RISC architecture
  - Founded by Acorn Computers (1985), now ARM Holdings
  - License-based business model - designs sold to manufacturers
  - Powers 95% of smartphones and tablets worldwide
- **Key ARM Characteristics:**
  - **Load/Store Architecture:** Only load/store instructions access memory
  - **Fixed 32-bit Instructions:** Uniform instruction length (ARM64: 64-bit)
  - **Conditional Execution:** Most instructions can be conditionally executed
  - **Low Power Design:** Optimized for battery-powered devices
- **ARM Processor Families:**
  - **Cortex-A:** Application processors (smartphones, tablets)
  - **Cortex-R:** Real-time processors (automotive, industrial)
  - **Cortex-M:** Microcontrollers (IoT, embedded systems)

ARM Instruction Set Example

- **ARM Assembly Examples:**

```
// Load/Store Operations
LDR R1, [R2]          // Load word from memory[R2] to R1
STR R1, [R2, #4]      // Store R1 to memory[R2 + 4]

// Arithmetic Operations
ADD R1, R2, R3        // R1 = R2 + R3
SUB R1, R2, #5        // R1 = R2 - 5 (immediate value)

// Conditional Execution
ADDEQ R1, R2, R3      // Add only if equal flag set
MOVNE R1, #0          // Move 0 to R1 if not equal

// Branch Instructions
B label               // Unconditional branch
BEQ label             // Branch if equal
```

ARM's Modern Success: Apple Silicon

- **Apple's ARM Transition:**
  - **M1 Chip (2020):** First ARM-based Mac processor
  - **M1 Pro/Max (2021):** High-performance variants
  - **M2 Series (2022+):** Next generation ARM processors
- **ARM Advantages in Apple Silicon:**
  - **Power Efficiency:** Exceptional battery life in MacBooks
  - **Unified Memory:** CPU and GPU share same memory pool
  - **Custom Silicon:** Apple designs custom ARM cores
  - **Performance:** Competitive with Intel/AMD x86 processors
- **Market Impact:**
  - Proved ARM can compete in laptop/desktop market
  - Microsoft developing ARM-based Windows
  - Amazon's Graviton ARM servers gaining adoption

x86 Architecture: The CISC Powerhouse

- **x86 History:**
  - **Intel 8086 (1978):** Original 16-bit processor
  - **80386 (1985):** First 32-bit x86 processor
  - **x86-64/AMD64 (2003):** 64-bit extension by AMD
  - Dominates desktop, laptop, and server markets
- **x86 CISC Characteristics:**
  - **Variable Instruction Length:** 1 to 15 bytes per instruction
  - **Complex Instructions:** Single instruction can do multiple operations
  - **Memory-to-Memory Operations:** Direct memory manipulation
  - **Rich Addressing Modes:** Multiple ways to specify operands
- **Modern x86 Complexity:**
  - Hundreds of instructions in instruction set
  - Backward compatibility maintained since 8086
  - Internal RISC-like execution (micro-ops)

x86 Instruction Set Example

- **x86 Assembly Examples:**

```
// Complex Memory Operations
ADD [EBX], EAX      // Add EAX to memory[EBX], store in memory
MOV EAX, [EBX+4]    // Load from memory[EBX+4] to EAX

// Variable Length Instructions
MOV AL, 5           // 2 bytes: Move immediate to 8-bit register
MOV EAX, 0x12345678 // 5 bytes: Move 32-bit immediate to register

// Complex Addressing Modes
MOV EAX, [EBX + ECX*2 + 8]  // EAX = memory[EBX + ECX*2 + 8]

// String Operations
REP MOVSB          // Repeat move string bytes (hardware loop)

// Stack Operations
```

Modern x86: CISC Outside, RISC Inside

- **Micro-Operation Translation:**
  - Complex x86 instructions decoded into simple micro-ops
  - Internal execution core is RISC-like
  - Best of both worlds: CISC compatibility + RISC performance
- **Example Translation:**
  - `ADD [EBX], EAX` becomes:
  - `LOAD temp, [EBX]` (micro-op 1)
  - `ADD temp, EAX` (micro-op 2)
  - `STORE temp, [EBX]` (micro-op 3)
- **Performance Techniques:**
  - **Out-of-Order Execution:** Execute micro-ops as dependencies allow
  - **Superscalar:** Multiple execution units run in parallel
  - **Branch Prediction:** Predict which way branches will go
  - **Speculative Execution:** Execute ahead speculatively

ARM vs x86: Performance and Power Comparison

ARM Strengths:

- **Power Efficiency:** 3-5x better performance per watt
- **Heat Generation:** Runs cooler, enables fanless designs
- **Battery Life:** Exceptional in mobile devices
- **Custom Silicon:** Licensees can customize designs
- **Cost:** Lower licensing and manufacturing costs

x86 Strengths:

- **Raw Performance:** Higher peak performance in many workloads
- **Software Ecosystem:** Decades of optimized software
- **Enterprise Features:** Advanced virtualization, security
- **Backward Compatibility:** Runs legacy software unchanged
- **Manufacturing:** Advanced process nodes (Intel, TSMC)

- **Current Trend:** ARM gaining ground in servers and laptops, x86 still dominant in desktop/enterprise
- **Future:** Likely convergence with both architectures borrowing from each other

Real-World Applications: Choosing the Right Architecture

- **ARM Dominates:**
  - **Mobile Devices:** Smartphones, tablets (95% market share)
  - **IoT/Embedded:** Sensors, smart devices, automotive
  - **Apple Ecosystem:** M1/M2 MacBooks, iPhones, iPads
  - **Cloud Computing:** Amazon Graviton, custom server chips
- **x86 Dominates:**
  - **Desktop/Laptop PCs:** Gaming, productivity, development
  - **Enterprise Servers:** Data centers, high-performance computing
  - **Legacy Systems:** Existing infrastructure and software
  - **High-End Gaming:** Maximum performance requirements
- **Decision Factors:**
  - Power efficiency vs raw performance
  - Software compatibility requirements
  - Cost constraints and development timeline
  - Target market and use case

The Von Neumann Architecture

- **The Von Neumann Architecture:** The core model of a modern computer.
    - Central Processing Unit (CPU): The "brain."
    - Main Memory (RAM): The workspace.
    - Input/Output (I/O) Systems.
- **A Deeper Look at the CPU:**
    - Control Unit (CU), Arithmetic Logic Unit (ALU), Registers.
- **The Memory Hierarchy:** A pyramid of speed, cost, and size.
    - **L1/L2/L3 Cache:** Ultra-fast memory on the CPU.
    - **RAM (Random Access Memory):** Volatile, fast memory for active programs.
    - **Permanent Storage:** Non-volatile, slower storage (SSDs, HDDs).

Key Takeaways

- **Architecture Matters:**
  - RISC vs CISC trade-offs shape modern computing
  - ARM's power efficiency revolutionizing mobile and laptop markets
  - x86's complexity enables high performance in servers and desktops
- **System Design Principles:**
  - Understand your hardware constraints and capabilities
  - Choose the right architecture for your use case
  - Balance performance, power, and cost requirements
- **Future Trends:**
  - ARM expanding into server and desktop markets
  - Heterogeneous computing (CPU + GPU + specialized processors)
  - Quantum and neuromorphic computing on the horizon

Next Steps

- **Hands-On Practice:**
  - Experiment with ARM and x86 assembly language
  - Profile applications to understand performance bottlenecks
  - Design systems with different architectural constraints
- **Further Learning:**
  - Advanced computer architecture courses
  - System design interview preparation
  - Open-source hardware projects (RISC-V)
  - High-performance computing and parallel programming
- **Career Applications:**
  - System architecture roles
  - Performance engineering
  - Embedded systems development
  - Cloud infrastructure design

References and Resources

- **Textbooks:**
  - Computer Organization and Design - Patterson & Hennessy
  - Computer Architecture: A Quantitative Approach - Hennessy & Patterson
  - ARM System Developer's Guide - Sloss, Symes & Wright
- **Online Resources:**
  - ARM Developer Documentation: `https://developer.arm.com`
  - Intel x86 Architecture Manuals: `https://intel.com/sdm`
  - RISC-V Foundation: `https://riscv.org`
- **Tools and Simulators:**
  - QEMU for architecture emulation
  - ARM Development Studio
  - Intel VTune Profiler
  - Online assembly simulators and debuggers

Thank You!

Questions?

Contact Information:
Email: your.email@institution.edu
Office Hours: By appointment

"The best way to learn computer architecture is to build one."
– Anonymous Computer Architect