

Computer Architecture and Organization (CE-CAO)

- CE-CAO0. History and overview of computer architecture [core]**
- CE-CAO1. Fundamentals of computer architecture [core]**
- CE-CAO2. Computer arithmetic [core]**
- CE-CAO3. Memory system organization and architecture [core]**
- CE-CAO4. Interfacing and communication [core]**
- CE-CAO5. Interface subsystems [core]**
- CE-CAO6. Processor systems design [core]**
- CE-CAO7. Organization of the CPU [core]**
- CE-CAO8. Performance [core]**
- CE-CAO9. Performance enhancements [elective]**
- CE-CAO10. Multiprocessing [core]**

Computer architecture is concerned with all aspects of the design and organization of the central processing unit and the integration of the CPU into the computer system itself. Architecture extends upward into computer software because a processor's architecture must cooperate with the operating system and system software. It is impossible to design an operating system well without a knowledge of the underlying architecture. Moreover, the computer designer has to have an intimate understanding of software in order to implement the optimum architecture. Moreover, there is a tighter relationship between the computer architect and the compiler writer than at any time in the past.

Computer architecture is a key component of computer engineering and the practicing computer engineer should have a practical understanding of this topic. Consequently, computer architecture courses should include a laboratory component where students are able to evaluate alternative designs.

The term *architecture* implies structure and therefore computer architecture tells us something about the way in which the elements of a computer relate to each other. Computer architecture is generally thought of as the *programmer's* view of a computer; that is, the *idealized* or *abstract* view of a computer. The *implementation* or *organization* of the computer is hidden from the programmer, although it would be wrong to divorce entirely architecture and implementation because each exerts a powerful influence on the other.

The computer architecture curriculum has to achieve multiple objectives. It must provide an overview of computer architecture and teach students the operation of a typical von Neumann machine. It must highlight the important issues facing today's designers and give students the tools they will need to carry out research. Ideally it should reinforce topics that are common to other areas of computer science; for example, teaching register indirect addressing reinforces the concept of pointers in C.

As with any engineering program students must learn to design and build things. Therefore, the computer engineering student should be expected to design and build a

simple computer or computer system. This will most likely take place in a laboratory environment.

CE-CAO0. History and overview of computer architecture and organization [core]

Suggested time: 1 hour

Topics:

- Indicate some reasons for studying computer architecture and organization.
- Highlight some people that influenced or contributed to the area of computer architecture and organization.
- Indicate some important topic areas such as system organization and architecture, memory, interfacing, microprocessors, and performance.
- Contrast the meanings of between computer organization and computer architecture.
- Indicate the importance of doing binary arithmetic with computers.
- Mention memory as a crucial component to the design of a computer.
- Illustrate the importance of interfacing with computer components and peripherals.
- Mention a typical CPU and sketch its organization.
- Indicate why performance leads to alternate architectures.
- Mention caching a way to improve performance.
- Mention some of the strategies used in architecture such as CISC and RISC approaches.
- Mention the strategies of multiprocessing strategies and their potential.
- Explore some additional resources associated with computer architecture and organization.
- Explain the purpose and role of computer architecture and organization in computer engineering.

Learning objectives:

- Identify some contributors to computer architecture and organization and relate their achievements to the knowledge area.
- Explain the reasons and strategies for different architectures.
- Articulate differences between computer organization and computer architecture.
- Identify some of the components of a computer.
- Indicate some strengths and weaknesses inherent in different architectures.
- Describe how computer engineering uses or benefits from computer architecture and organization.

CE-CAO1. Fundamentals of computer architecture [core]

Minimum core coverage time: 18 hours

Topics:

1. Organization of the von Neumann machine
2. Instruction formats
3. The fetch/execute cycle; instruction decoding and execution
4. Registers and register files.

5. Instruction types and addressing modes
6. Subroutine call and return mechanisms
7. Programming in assembly language
8. I/O techniques and interrupts
9. Other design issues.

Learning objectives:

1. Be able to explain the organization of a von Neumann machine and its major functional units.
2. Be able to explain how an instruction is fetched from memory and executed.
3. Be able to articulate the strengths and weaknesses of the von Neumann architecture.
4. Be able to explain the relationship between the representation of machine level operation at the binary level and their representation by a symbolic assembler.
5. Be able to explain why a designer adopted a given different instruction formats, such as the number of addresses per instruction and variable length vs. fixed length formats.
6. Be able to write small programs and fragments of assembly language code to demonstrate an understanding of machine level operations.
7. Be able to implement some fundamental high-level programming constructs at the machine-language level.
8. Be able to use computer simulation packages to investigate assembly language programming.

CA2. Computer arithmetic [core]

Minimum core coverage time: 3 hours

Topics:

1. Representation of integers (positive and negative numbers)
2. Algorithms for common arithmetic operations (addition, subtraction, multiplication, division)
3. Significance of range, precision, and accuracy in computer arithmetic
4. Representation of real numbers (standards for floating-point arithmetic)
5. Algorithms for carrying out common floating-point operations
6. Converting between integer and real numbers
7. Multi-precision arithmetic
8. Hardware and software implementation of arithmetic unit.
9. The generation of higher order functions from square roots to transcendentals.

Learning objectives:

1. Appreciate how numerical values are represented in digital computers
2. Understand the limitations of computer arithmetic and the effects of errors on calculations.
3. Appreciate the effect of a processor's arithmetic unit on its overall performance,

CE-CAO3. Memory system organization and architecture [core]

Minimum core coverage time: 9 hours

Topics:

1. Memory systems hierarchy
2. Coding, data compression, and data integrity
3. Electronic, magnetic and optical technologies
4. Main memory organization and its characteristics and performance
5. Latency, cycle time, bandwidth, and interleaving
6. Cache memories (address mapping, line size, replacement and write-back policies)
7. Virtual memory systems
8. Memory technologies (DRAM, EPROM, FLASH, etc.)
9. Reliability of memory systems. Error detecting and error correcting systems.

Learning objectives:

1. Identify the main types of memory technology.
2. Explain the effect of memory latency and bandwidth on performance.
3. Explain the use of memory hierarchy to reduce the effective memory latency.
4. Describe the principles of memory management.
5. Appreciate how errors in memory systems arise and what can be done about them.

CE-CAO4. Interfacing and communication [core]

Minimum core coverage time: 12 hours

Topics:

1. I/O fundamentals: handshaking, buffering,
2. I/O techniques: programmed I/O, interrupt-driven I/O, DMA
3. Interrupt structures: vectored and prioritised, interrupt overhead, interrupts and reentrant code.
4. Memory system design and interfacing.
5. Buses: bus protocols, local and geographic arbitration

Learning objectives:

1. Explain how interrupts are used to implement I/O control and data transfers.
2. Write small interrupt service routines and I/O drivers using assembly language.
3. Identify various types of buses in a computer system.
4. Describe data access from a magnetic disk drive.
5. Be able to analyse and implement interfaces.

CE-CAO5. Interface subsystems [core]

Minimum core coverage time: 12 hours

Topics:

1. External storage systems; organization and structure of disk drives and optical memory

2. Basic I/O controllers, keyboard, mouse, etc.
3. RAID architectures
4. Video control
5. I/O Performance
6. SMART technology and fault detection
7. Processor to network interfaces

Learning objectives:

1. Compute the various parameters of performance for standard I/O types.
2. Explain the basic nature human computer interaction devices.
3. Describe data access from magnetic and optical disk drives.

CE-CAO6. Processor systems design [core]

Minimum core coverage time: 12 hours

Topics:

1. The CPU interface: clock, control, data and address buses
2. Address decoding and memory interfacing
3. Basic parallel and serial interfaces
4. Timers
5. System firmware

Learning objectives:

1. Appreciate how a CPU chip is turned into a complete system.
2. Be able to design an interface to memory
3. Understand how to interface and use peripheral chips
4. Write sufficient EPROM-based system software to create a basic stand-alone system.
5. Be able to specify and design simple computer interfaces.

CE-CAO7. Organization of the CPU [core]

Minimum core coverage time: 12 hours

Topics:

1. Implementation of the von Neumann machine
2. Single vs. multiple bus datapaths
3. Instruction set architecture; machine architecture as a framework for encapsulating design decisions.
4. Relationship between the architecture and the compiler
5. Implementing instructions
6. Control unit: hardwired realization vs. microprogrammed realization
7. Arithmetic units, for multiplication and division.
8. Instruction pipelining
9. Trends in computer architecture: CISC, RISC, VLIW

10. Introduction to instruction-level parallelism (ILP)
11. Pipeline hazards: structural, data and control
12. Reducing the effects of hazards

Learning objectives:

1. Compare alternative implementation of datapaths.
2. Discuss the generation of control signals using hardwired or microprogrammed implementations.
3. Explain basic instruction level parallelism using pipelining and the major hazards that may occur.
4. Explain what has been done to overcome the effect of branches
5. Discuss the way in which instruction sets have evolved to improve performance; for example, predicated execution.

CE-CAO8. Performance [core]

Minimum core coverage time: 3 hours

Topics:

1. Metrics for computer performance; clock rate, MIPS, Cycles per instruction, benchmarks
2. Strengths and weaknesses of performance metrics
3. Averaging metrics: arithmetic, geometric and harmonic
4. The role of Amdahl's law in computer performance

Learning objectives:

1. Appreciate all the factors that contribute to computer performance.
2. Understand the limitations of performance metrics
3. Be able to select the most appropriate performance metric when evaluating a computer.
4. Discuss the impact on control and datapath design for performance enhancements.

CE-CAO9. Performance enhancements [elective]

Topics:

1. Superscalar architecture
2. Branch prediction
3. Prefetching
4. Speculative execution
5. Multithreading
6. Scalability
7. Short vector instruction sets; Streaming extensions, AltiVec; relationship between computer architecture and multimedia applications.

Learning objectives:

1. Discuss how various architectural enhancements affect system performance.
2. Discuss how parallel processing approaches can be applied to the design of scalar and superscalar processors.
3. Discuss how vector processing techniques can be applied to enhance instruction sets to be used for multimedia, signal processing, etc.
4. Appreciate how each of the functional parts of a computer system affect its overall performance. Be able to estimate the effect on system performance of changes to functional units.

CE-CA09. Multiprocessing [elective] consider moving this to HPC section.

Minimum core coverage time: 3 hours

Topics:

1. Systolic architectures
2. Interconnection networks (hypercube, shuffle-exchange, mesh, crossbar)
3. Shared memory systems
4. Cache coherence
5. Memory models and memory consistency

Learning objectives:

1. Discuss the concept of parallel processing beyond the classical von Neumann model.
2. Describe the limitations imposed by interconnections on multiprocessing systems.
3. Appreciate the problems caused by cache coherency and understand the ways in which the problem can be overcome.