Architecture and Organization (AR)

Computing professionals should not regard the computer as just a black box that executes programs by magic. AR-Architecture and Organization builds on SF-Systems Fundamentals to develop a deeper understanding of the hardware environment upon which all of computing is based, and the interface it provides to higher software layers. Students should acquire an understanding and appreciation of a computer system's functional components, their characteristics, performance, and interactions, and, in particular, the challenge of harnessing parallelism to sustain performance improvements now and into the future. Students need to understand computer architecture to develop programs that can achieve high performance through a programmer's awareness of parallelism and latency. In selecting a system to use, students should to able to understand the tradeoff among various components, such as CPU clock speed, cycles per instruction, memory size, and average memory access time.

The learning outcomes specified for these topics correspond primarily to the core and are intended to support programs that elect to require only the minimum 16 hours of computer architecture of their students. For programs that want to teach more than the minimum, the same AR topics can be treated at a more advanced level by implementing a two-course sequence. For programs that want to cover the elective topics, those topics can be introduced within a two-course sequence and/or be treated in a more comprehensive way in a third course.

AR. Architecture and Organization (0 Core-Tier 1 hours, 16 Core-Tier 2 hours)

	Core-Tier 1 hours	Core-Tier 2 Hours	Includes Elective
AR/Digital logic and digital systems		**3**	**N**
AR/Machine level representation of data		**3**	**N**
AR/Assembly level machine organization		**6**	**N**
AR/Memory system organization and architecture		**3**	**N**
AR/Interfacing and communication		**1**	**N**
AR/Functional organization			**Y**
AR/Multiprocessing and alternative architectures			**Y**
AR/Performance enhancements			**Y**

AR/Digital logic and digital systems

[3 Core-Tier2 hours]

***Topics**:*

- Overview and history of computer architecture
- Combinational vs. sequential logic/Field programmable gate arrays as a fundamental combinational + sequential logic building block
- Multiple representations/layers of interpretation (hardware is just another layer)
- Computer-aided design tools that process hardware and architectural representations
- Register transfer notation/Hardware Description Language (Verilog/VHDL)
- Physical constraints (gate delays, fan-in, fan-out, energy/power)

***Learning outcomes**:*

1. Describe the progression of computer technology components from vacuum tubes to VLSI, from mainframe computer architectures to the organization of warehouse-scale computers [Familiarity].
2. Comprehend the trend of modern computer architectures towards multi-core and that parallelism is inherent in all hardware systems [Familiarity].
3. Explain the implications of the "power wall" in terms of further processor performance improvements and the drive towards harnessing parallelism [Familiarity].
4. Articulate that there are many equivalent representations of computer functionality, including logical expressions and gates, and be able to use mathematical expressions to describe the functions of simple combinational and sequential circuits [Familiarity].
5. Design the basic building blocks of a computer: arithmetic-logic unit (gate-level), registers (gate-level), central processing unit (register transfer-level), memory (register transfer-level) [Usage].
6. Use CAD tools for capture, synthesis, and simulation to evaluate simple building blocks (e.g., arithmetic-logic unit, registers, movement between registers) of a simple computer design [Usage].

7. Evaluate the functional and timing diagram behavior of a simple processor implemented at the logic circuit level [Assessment].

AR/Machine-level representation of data

[3 Core-Tier2 hours]

Topics:

- Bits, bytes, and words
- Numeric data representation and number bases
- Fixed- and floating-point systems
- Signed and twos-complement representations
- Representation of non-numeric data (character codes, graphical data)
- Representation of records and arrays

Learning outcomes:

1. Explain why everything is data, including instructions, in computers [Familiarity].
2. Explain the reasons for using alternative formats to represent numerical data [Familiarity].
3. Describe how negative integers are stored in sign-magnitude and twos-complement representations [Familiarity].
4. Explain how fixed-length number representations affect accuracy and precision [Familiarity].
5. Describe the internal representation of non-numeric data, such as characters, strings, records, and arrays [Familiarity].
6. Convert numerical data from one format to another [Usage].
7. Write simple programs at the assembly/machine level for string processing and manipulation [Usage].

AR/Assembly level machine organization

[6 Core-Tier2 hours]

Topics:

- Basic organization of the von Neumann machine
- Control unit; instruction fetch, decode, and execution
- Instruction sets and types (data manipulation, control, I/O)
- Assembly/machine language programming
- Instruction formats
- Addressing modes
- Subroutine call and return mechanisms (xref PL/Language Translation and Execution)
- I/O and interrupts
- Heap vs. Static vs. Stack vs. Code segments
- Shared memory multiprocessors/multicore organization
- Introduction to SIMD vs. MIMD and the Flynn Taxonomy

***Learning outcomes*:**

1. Explain the organization of the classical von Neumann machine and its major functional units [Familiarity].
2. Describe how an instruction is executed in a classical von Neumann machine, with extensions for threads, multiprocessor synchronization, and SIMD execution [Familiarity].

3. Describe instruction level parallelism and hazards, and how they are managed in typical processor pipelines [Familiarity].
4. Summarize how instructions are represented at both the machine level and in the context of a symbolic assembler [Familiarity].
5. Demonstrate how to map between high-level language patterns into assembly/machine language notations [Familiarity].
6. Explain different instruction formats, such as addresses per instruction and variable length vs. fixed length formats [Familiarity].
7. Explain how subroutine calls are handled at the assembly level [Familiarity].
8. Explain the basic concepts of interrupts and I/O operations [Familiarity].
9. Write simple assembly language program segments [Usage].
10. Show how fundamental high-level programming constructs are implemented at the machine-language level [Usage].

AR/Memory system organization and architecture

[3 Core-Tier2 hours]

[Cross-reference OS/Memory Management--Virtual Machines]

Topics:

- Storage systems and their technology
- Memory hierarchy: importance of temporal and spatial locality
- Main memory organization and operations
- Latency, cycle time, bandwidth, and interleaving
- Cache memories (address mapping, block size, replacement and store policy)
- Multiprocessor cache consistency/Using the memory system for inter-core synchronization/atomic memory operations
- Virtual memory (page table, TLB)
- Fault handling and reliability
- Error coding, data compression, and data integrity (cross-reference SF/Reliability through Redundancy)

Learning outcomes:

1. Identify the main types of memory technology [Familiarity].
2. Explain the effect of memory latency on running time [Familiarity].
3. Describe how the use of memory hierarchy (cache, virtual memory) is used to reduce the effective memory latency [Familiarity].
4. Describe the principles of memory management [Familiarity].
5. Explain the workings of a system with virtual memory management [Familiarity].
6. Compute Average Memory Access Time under a variety of memory system configurations and workload assumptions [Usage].

AR/Interfacing and communication

[1 Core-Tier2 hour]

[Cross-reference OS Knowledge Area for a discussion of the operating system view of input/output processing and management. The focus here is on the hardware mechanisms for supporting device interfacing and processor-to-processor communications.]

Topics:

- I/O fundamentals: handshaking, buffering, programmed I/O, interrupt-driven I/O
- Interrupt structures: vectored and prioritized, interrupt acknowledgment
- External storage, physical organization, and drives
- Buses: bus protocols, arbitration, direct-memory access (DMA)
- Introduction to networks: networks as another layer of access hierarchy
- Multimedia support
- RAID architectures

***Learning outcomes*:**

1. Explain how interrupts are used to implement I/O control and data transfers [Familiarity].
2. Identify various types of buses in a computer system [Familiarity].
3. Describe data access from a magnetic disk drive [Familiarity].
4. Compare common network organizations, such as ethernet/bus, ring, switched vs. routed [Familiarity].
5. Identify interfaces needed for multimedia support, from storage, through network, to memory and display [Familiarity].
6. Describe the advantages and limitations of RAID architectures [Familiarity].

AR/Functional organization

[Elective]

[Note: elective for computer scientist; would be core for computer engineering curriculum]

Topics:

- Implementation of simple datapaths, including instruction pipelining, hazard detection and resolution
- Control unit: hardwired realization vs. microprogrammed realization
- Instruction pipelining
- Introduction to instruction-level parallelism (ILP)

Learning outcomes:

1. Compare alternative implementation of datapaths [Familiarity].
2. Discuss the concept of control points and the generation of control signals using hardwired or microprogrammed implementations [Familiarity].
3. Explain basic instruction level parallelism using pipelining and the major hazards that may occur [Familiarity].
4. Design and implement a complete processor, including datapath and control [Usage].
5. Determine, for a given processor and memory system implementation, the average cycles per instruction [Assessment].

AR/Multiprocessing and alternative architectures

[Elective]

[Cross-reference PD/Parallel Architecture: The view here is on the hardware implementation of SIMD and MIMD architectures; in PD/Parallel Architecture, it is on the way that algorithms can be matched to the underlying hardware capabilities for these kinds of parallel processing architectures.]

Topics:

- Power Law
- Example SIMD and MIMD instruction sets and architectures
- Interconnection networks (hypercube, shuffle-exchange, mesh, crossbar)
- Shared multiprocessor memory systems and memory consistency
- Multiprocessor cache coherence

***Learning outcomes*:**

1. Discuss the concept of parallel processing beyond the classical von Neumann model [Familiarity].
2. Describe alternative architectures such as SIMD and MIMD [Familiarity].
3. Explain the concept of interconnection networks and characterize different approaches [Familiarity].
4. Discuss the special concerns that multiprocessing systems present with respect to memory management and describe how these are addressed [Familiarity].
5. Describe the differences between memory backplane, processor memory interconnect, and remote memory via networks [Familiarity].

AR/Performance enhancements

[Elective]

Topics:

- Superscalar architecture
- Branch prediction, Speculative execution, Out-of-order execution
- Prefetching
- Vector processors and GPUs
- Hardware support for Multithreading
- Scalability
- Alternative architectures, such as VLIW/EPIC, and Accelerators and other kinds of Special-Purpose Processors

Learning outcomes:

1. Describe superscalar architectures and their advantages [Familiarity].
2. Explain the concept of branch prediction and its utility [Familiarity].
3. Characterize the costs and benefits of prefetching [Familiarity].
4. Explain speculative execution and identify the conditions that justify it [Familiarity].
5. Discuss the performance advantages that multithreading offered in an architecture along with the factors that make it difficult to derive maximum benefits from this approach [Familiarity].
6. Describe the relevance of scalability to performance [Familiarity].