Architecture and Organization (AR)

Preamble

Computing professionals spend considerable time writing efficient code to solve a particular problem in an application domain. Parallelism and heterogeneity at the hardware system level have been increasingly utilized to meet performance requirements in almost all systems, including most commodity hardware. This departure from sequential processing demands a more in-depth understanding of the underlying computer architectures. Architecture can no longer be treated as a *black box* where principles from one can be applied to another. Instead, programmers should look inside the black box and use specific components to improve system performance and energy efficiency.

The Architecture and Organization (AR) Knowledge Area aims to develop a deeper understanding of the hardware environments upon which almost all computing is based and the relevant interfaces provided to higher software layers. The target hardware comprises low-end embedded system processors up to high-end enterprise multiprocessors.

The topics in this knowledge area will benefit students by enabling them to appreciate the fundamental architectural principles of modern computer systems, including the challenge of harnessing parallelism to sustain performance and energy improvements into the future. This KA will help computer science students depart from the black box approach and become more aware of the underlying computer system and the efficiencies that specific architectures can achieve.

Changes since CS 2013

This KA has changed slightly since the CS2013 report. Changes and additions are summarized as follows:

- Topics have been revised, particularly in the Knowledge Units AR/Memory Hierarchy and AR/Performance and Energy Efficiency. This change supports recent advances in memory caching and energy consumption.
- To address emerging topics in Computer Architecture, the newly created KU AR/Heterogeneous Architectures covers the introductory level: In-Memory processing (PIM), domain-specific architectures (e.g. neural network processors) - and related topics.
- Quantum computing, particularly the development of quantum processor architectures, has been gathering pace recently. The new KU AR/Quantum Architectures offer a "toolbox" covering introductory topics in this important emerging area.
- Knowledge units have been merged to better deal with overlaps:
    - AR/Multiprocessing and Alternative Architectures were merged into newly created AR/Heterogeneous Architectures.

Overview

Core Hours

Knowledge Unit	CS Core	KA Core
Digital Logic and Digital Systems		3
Machine-Level Data Representation	1	
Assembly Level Machine Organization	1	2
Memory Hierarchy	6	
Interfacing and Communication	1	
Functional Organization		2
Performance and Energy Efficiency		3
Heterogeneous Architectures		3
Quantum Architectures		3
Total	**9**	**16**

Knowledge Units

AR-A / AR-Logic: Digital Logic and Digital Systems

KA Core:

1. Overview and history of computer architecture
2. Combinational vs sequential logic/field programmable gate arrays (FPGAs)
    a. Fundamental combinational
    b. Sequential logic building block
3. Functional hardware and software multi-layer architecture
4. Computer-aided design tools that process hardware and architectural representations
5. High-level synthesis
    a. Register transfer notation
    b. Hardware description language (e.g. Verilog/VHDL/Chisel)
6. System-on-chip (SoC) design flow
7. Physical constraints
    a. Gate delays
    b. Fan-in and fan-out
    c. Energy/power
    d. Speed of light

Illustrative Learning Outcomes

KA Core:

1. Comment on the progression of computer technology components from vacuum tubes to VLSI, from mainframe computer architectures to the organization of warehouse-scale computers.
2. Comment on parallelism and data dependencies between and within components in a modern heterogeneous computer architecture.
3. Explain how the “power wall" makes it challenging to harness parallelism.
4. Propose the design of basic building blocks for a computer: arithmetic-logic unit (gate-level), registers (gate-level), central processing unit (register transfer-level), and memory (register transfer-level).
5. Evaluate simple building blocks (e.g., arithmetic-logic unit, registers, movement between registers) of a simple computer design.
6. Validate the timing diagram behavior of a pipelined processor, identifying data dependency issues.

AR-B / AR-Representation: Machine-Level Data Representation

CS Core:

1. Bits, bytes, and words
2. Numeric data representation and number bases (See also: GIT-Visualization)
    a. Fixed-point
    b. Floating-point
3. Signed and twos-complement representations
4. Representation of non-numeric data
5. Representation of records and arrays

Illustrative Learning Outcomes

CS Core:

1. Comment on the reasons why everything is data in computers, including instructions.
2. Follow the diagram and annotate the regions where fixed-length number representations affect accuracy and precision.
3. Comment on how negative integers are stored in sign-magnitude and twos-complement representations.
4. Articulate with plausible justification how different formats can represent numerical data.
5. Explain the bit-level representation of non-numeric data, such as characters, strings, records, and arrays.
6. Translate numerical data from one format to another.
7. Explain how a single adder (without overflow detection) can handle both signed (two’s complement) and unsigned (binary) input without “knowing” which format a given input is using.

AR-C / AR-Assembly: Assembly Level Machine Organization

CS Core:

1. von Neumann machine architecture
2. Control unit; instruction fetch, decode, and execution
3. Introduction to SIMD vs. MIMD and the Flynn taxonomy (See also: PDC-A: Programs and Execution)
4. Shared memory multiprocessors/multicore organization

KA Core:

5. Instruction set architecture (ISA) (e.g., x86, ARM and RISC-V)
    a. Fixed vs. variable-width instruction sets
    b. Instruction formats
    c. Data manipulation, control, I/O
    d. Addressing modes
    e. Machine language programming
    f. Assembly language programming
6. Subroutine call and return mechanisms (xref PL/language translation and execution)
7. I/O and interrupts
8. Heap, static, stack and code segments

Illustrative Learning Outcomes

CS Core:

1. Contextualize the classical von Neumann functional units in embedded systems, particularly on-chip and off-chip memory.
2. Comment on how instruction is executed in a classical von Neumann machine, with extensions for threads, multiprocessor synchronization, and SIMD execution.
3. Annotate an example diagram with instruction-level parallelism and hazards to comment on how they are managed in typical processor pipelines.

KA Core:

4. Comment on how instructions are represented at the machine level and in the context of a symbolic assembler.
5. Map an example of high-level language patterns into assembly/machine language notations.
6. Comment on different instruction formats, such as addresses per instruction and variable-length vs fixed-length formats.
7. Follow a subroutine diagram to comment on how subroutine calls are handled at the assembly level.
8. Comment on basic concepts of interrupts and I/O operations.
9. Code a simple assembly language program for string array processing and manipulation.

AR-D / AR-Memory: Memory Hierarchy

CS Core:

1. Memory hierarchy: the importance of temporal and spatial locality (See also OS-F: Memory Management)
2. Main memory organization and operations
3. Persistent memory (e.g. SSD, standard disks)
4. Latency, cycle time, bandwidth and interleaving
5. Cache memories
    a. Address mapping
    b. Block size
    c. Replacement and store policy
6. Multiprocessor cache coherence
7. Virtual memory (hardware support, cross-reference OS/Virtual Memory) (See also: OS-F: Memory Management)
8. Fault handling and reliability
9. Reliability (cross-reference SF/Reliability through Redundancy) (See also: SF-F: System reliability)
    a. Error coding
    b. Data compression (See also: AL-Fundamentals.3)
    c. Data integrity

KA Core:

10. Non-von Neumann Architectures
    a. Processing In-Memory (PIM)

Illustrative Learning Outcomes

CS Core:

1. Using a memory system diagram, detect the main types of memory technology (e.g., SRAM, DRAM) and their relative cost and performance.
2. Measure the effect of memory latency on running time.
3. Enumerate the functions of a system with virtual memory management.
4. Compute average memory access time under various cache and memory configurations and mixes of instruction and data references.

AR-E / AR-IO: Interfacing and Communication

CS Core:

1. I/O fundamentals
    a. Handshaking and buffering
    b. Programmed I/O
    c. Interrupt-driven I/O
2. Interrupt structures: vectored and prioritized, interrupt acknowledgement
3. I/O devices (e.g., mouse, keyboard, display, camera, sensors, actuators) (see also: GIT-Fundamental Concepts, GIT-Interaction)
4. External storage, physical organization and drives
5. Buses fundamentals

a. Bus protocols
b. Arbitration
c. Direct-memory access (DMA)

Illustrative Learning Outcomes
CS Core:

1. Follow an interrupt control diagram to comment on how interrupts are used to implement I/O control and data transfers.
2. Enumerate various types of buses in a computer system.
3. List the advantages of magnetic disks and contrast them with the advantages of solid-state disks.

AR-F / AR-Organization: Functional Organization

KA Core:

1. Implementation of simple datapaths, including instruction pipelining, hazard detection and resolution (e.g., stalls, forwarding)
2. Control unit
    a. Hardwired implementation
    b. Microprogrammed realization
3. Instruction pipelining
4. Introduction to instruction-level parallelism (ILP)

Illustrative Learning Outcomes
KA Core:

1. Validate alternative implementation of datapaths in modern computer architectures.
2. Propose a set of control signals for adding two integers using hardwired and microprogrammed implementations.
3. Comment on instruction-level parallelism using pipelining and significant hazards that may occur.
4. Design a complete processor, including datapath and control.
5. Compute the average cycles per instruction for a given processor and memory system implementation.

AR-G / AR-Performance-Energy: Performance and Energy Efficiency

KA Core:

1. Performance-energy evaluation (introduction): performance, power consumption, memory and communication costs
2. Branch prediction, speculative execution, out-of-order execution, Tomasulo's algorithm
3. Prefetching
4. Enhancements for vector processors and GPUs
5. Hardware support for Multithreading

    a. Race conditions
    b. Lock implementations
    c. Point-to-point synchronization
    d. Barrier implementation
6. Scalability
7. Alternative architectures, such as VLIW/EPIC, accelerators and other special-purpose processors
8. Dynamic voltage and frequency scaling (DVFS)
9. Dark Silicon

Illustrative Learning Outcomes

KA Core:

1. Comment on evaluation metrics for performance and energy efficiency.
2. Follow a speculative execution diagram and write about the decisions that can be made.
3. Build a GPU performance-watt benchmarking diagram.
4. Code a multi-threaded program that adds (in parallel) elements of two integer vectors.
5. Propose a set of design choices for alternative architectures.
6. Comment on key concepts associated with dynamic voltage and frequency scaling.
7. Measure energy savings improvement for an 8-bit integer quantization compared to a 32-bit quantization.

AR-H / AR-Heterogeneity: Heterogeneous Architectures

KA Core:

1. SIMD and MIMD architectures (e.g. General-Purpose GPUs, TPUs and NPUs) (See also: SPD-C: Mobile Platforms, GIT-Shading)
2. Heterogeneous memory system
    a. Shared memory versus distributed memory
    b. Volatile vs non-volatile memory
    c. Coherence protocols
3. Domain-Specific Architectures (DSAs) (AI-D: Machine Learning; HCI-B: Accountability and Responsibility in Design)
    a. Machine Learning Accelerator
    b. In-networking computing
    c. Embedded systems for emerging applications
    d. Neuromorphic computing
4. Packaging and integration solutions such as 3DIC and Chiplets

Illustrative Learning Outcomes

KA Core

1. Follow a system diagram with alternative parallel architectures, e.g. SIMD and MIMD, and annotate the key differences.

2. Tell what memory-management issues are found in multiprocessors that are not present in uniprocessors and how these issues might be resolved.
3. Validate the differences between memory backplane, processor memory interconnect, and remote memory via networks, their implications for access latency and their impact on program performance.
4. Tell how you would determine when to use a domain-specific accelerator instead of a general-purpose CPU.
5. Enumerate key differences in architectural design principles between a vector and scalar-based processing unit.
6. List the advantages and disadvantages of PIM architectures.

AR-I / AR-Quantum: Quantum Architectures

KA Core:

1. Principles
    a. The wave-particle duality principle
    b. The uncertainty principle in the double-slit experiment
    c. What is a Qubit? Superposition and measurement. Photons as qubits.
    d. Systems of two qubits. Entanglement. Bell states. The No-Signaling theorem.
2. Axioms of QM: superposition principle, measurement axiom, unitary evolution
3. Single qubit gates for the circuit model of quantum computation: X, Z, H.
4. Two qubit gates and tensor products. Working with matrices.
5. The No-Cloning Theorem. The Quantum Teleportation protocol.
6. Algorithms
    a. Simple quantum algorithms: Bernstein-Vazirani, Simon's algorithm.
    b. Implementing Deutsch-Josza with Mach-Zehnder Interferometers.
    c. Quantum factoring (Shor's Algorithm)
    d. Quantum search (Grover's Algorithm)
7. Implementation aspects
    a. The physical implementation of qubits
    b. Classical control of a Quantum Processing Unit (QPU)
    c. Error mitigation and control. NISQ and beyond.
    d. Measurement approaches
8. Emerging Applications
    a. Post-quantum encryption
    b. The Quantum Internet
    c. Adiabatic quantum computation (AQC) and quantum annealing

Illustrative Learning Outcomes

KA Core:

1. Comment on a quantum object produced as a particle, propagates like a wave and is detected as a particle with a probability distribution corresponding to the wave.
2. Follow the diagram and comment on the quantum-level nature that is inherently probabilistic.

3. Articulate your view on entanglement that can be used to create non-classical correlations, but there is no way to use quantum entanglement to send messages faster than the speed of light.
4. Comment on quantum parallelism and the role of constructive vs destructive interference in quantum algorithms given the probabilistic nature of measurement(s).
5. Annotate a code snippet providing the role of quantum Fourier transform (QFT) in Shor's algorithm.
6. Code Shor's algorithm in a simulator and document your code, highlighting the classical components and aspects of Shor's algorithm.
7. Enumerate the specifics of each qubit modality (e.g., trapped ion, superconducting, silicon spin, photonic, quantum dot, neutral atom, topological, color center, electron-on-helium, etc.),
8. Contextualize the differences between AQC and the gate model of quantum computation and which kind of problems each is better suited to solve.
9. Comment on the statement: a QPU is a heterogeneous multicore architecture like an FPGA or a GPU.

Dispositions

- Self-directed: students should increasingly become self-motivated to acquire complementary knowledge.
- Proactive: students should exercise control and antecipate issues related to the underlying computer system.

Math Requirements

Required:

- Discrete Math: Sets, Relations, Logical Operations, Number Theory, Boolean Algebra
- Linear Algebra: Arithmetic Operations, Matrix operations
- Logarithms, Limits

Desired:

- Math/Physics for Quantum Computing: basic probability, trigonometry, simple vector spaces, complex numbers, Euler's formula
- System performance evaluation: probability and factorial experiment design.

Course Packaging Suggestions

Computer Architecture - Introductory Course to include the following:

- SEP-History [2 hours]
- AR-B / AR-Representation: Machine-Level Data Representation [2 hours]
- AR-C / AR-Assembly: Assembly Level Machine Organization [2 hours]
- AR-D / AR-Memory: Memory Hierarchy [10 hours]

- OS/Memory Management [10 hours]
- AR-E / AR-IO: Interfacing and Communication [4 hours]
- AR-H / AT-Heterogeneity: Heterogeneous Architectures [5 hours]
- PD/Programs and Executions [4 hours]
- SEP/Methods for Ethical Analysis [3 hours]

Pre-requisites:
- Discrete Math: Sets, Relations, Logical Operations, Number Theory, Basic Programming

Skill statement:
- A student who completes this course should be able to understand the fundamental architectures of modern computer systems, including the challenge of memory caches, memory management and pipelining.

Systems Course to include the following:
- SEP-History [2 hours]
- SEP/Privacy and Civil Liberties [3 hours]
- SF-H: System Design [2 hours]
- SF-F: System Reliability [2 hours]
- PD/Algorithms and applications [4 hours]
- AR-H / AR-Heterogeneity: Heterogeneous Architectures [4 hours]
- OS/Roles and Purpose of Operating Systems [3 hours]
- AR-D / AR-Memory: /Memory Hierarchy [7 hours]
- AR-G / AR-Performance-energy: Performance and Energy Efficiency [5 hours]
- NC/Networked Applications [5 hours]

Pre-requisites:
- Discrete Math: Sets, Relations, Logical Operations, Number Theory

Skill statement:
- A student who completes this course should be able to appreciate the advanced architectural aspects of modern computer systems, including the challenge of heterogeneous architectures and the required hardware and software interfaces to improve the performance and energy footprint of applications.

Computer Architecture - Advanced Topics Course to include the following:
- AR-A / AR-Logic.1 Digital Logic and Digital Systems/Overview and history of computer architecture [4 hours]
- SF-E: Performance Evaluation [4 hours]
- PD/Algorithms and applications [4 hours]
- AR-H / AR-Heterogeneity: Heterogeneous Architectures [4 hours]
- AR-G / AR-Performance-Energy: Performance and Energy Efficiency [5 hours]
- AR-C: Cryptography [4 hours]
- AR-I / AR-Quantum: Quantum Architectures [4 hours]

Pre-requisites:

- Discrete Math: Sets, Relations, Logical Operations, Number Theory, Probability, Linear Algebra

Skill statement:

- A student who completes this course should be able to appreciate how computer architectures evolved into today's heterogeneous systems and to what extent past design choices can influence the design of future high-performance computing

Committee

Chair: Marcelo Pias, Federal University of Rio Grande (FURG), Rio Grande, RS, Brazil

Members:

- Brett A. Becker, University College Dublin, Dublin, Ireland
- Mohamed Zahran, New York University, New York, NY, USA
- Monica D. Anderson, University of Alabama, Tuscaloosa, AL, USA
- Qiao Xiang, Xiamen University, China
- Adrian German, Indiana University, Bloomington, IN, USA

Appendix: Core Topics and Skill Levels

KA	KU	Topic	Skill	Core	Hours

AR	Digital Logic and Digital Systems	• Overview and history of computer architecture
		• Combinational vs. sequential logic
		  ○ Fundamental combinational
		  ○ Sequential logic building block
		• Functional hardware and software multi-layer architecture
			Explain	KA	3

	Digital Logic and Digital Systems	• Computer-aided design tools that process hardware and architectural representations
		• High-level synthesis
		  ○ Register transfer notation
		  ○ Hardware description language (e.g. Verilog/VHDL/Chisel)
		• System-on-chip (SoC) design flow
		• Physical constraints
		  ○ Gate delays
		  ○ Fan-in and fan-out
			Evaluate		

		○ Energy/power [Shared with SPD] ○ Speed of light			
AR	Machine-Level Data Representation	● Bits, bytes, and words ● Numeric data representation and number bases ○ Fixed-point ○ Floating-point ● Signed and twos-complement representations ● Representation of non-numeric data ● Representation of records and arrays	Apply	CS	1
AR	Assembly Level Machine Organization	● von Neumann machine architecture ● Control unit; instruction fetch, decode, and execution ● Introduction to SIMD vs. MIMD and the Flynn taxonomy ● Shared memory multiprocessors/multicore organization [Shared with OS]	Explain	CS	1
AR	Assembly Level Machine Organization	● Instruction set architecture (ISA) (e.g. x86, ARM and RISC-V) ○ Instruction formats ○ Data manipulation, control, I/O ○ Addressing modes ○ Machine language programming ○ Assembly language programming ● Subroutine call and return mechanisms (xref PL/language translation and execution) [Shared with OS] ● I/O and interrupts [Shared with OS] ● Heap, static, stack and code segments [Shared with OS]	Develop	KA	2
		● Memory hierarchy: the importance of temporal and spatial locality [Shared with OS] ● Main memory organization and operations ● Persistent memory (e.g. SSD, standard disks) [Shared with OS] ● Latency, cycle time, bandwidth and interleaving	Explain		

AR	Memory Hierarchy	● Virtual memory (hardware support, cross-reference OS/Virtual Memory) [Shared with OS]
		● Fault handling and reliability [Shared with OS]
		● Reliability (cross-reference SF/Reliability through Redundancy)
		  ○ Error coding
		  ○ Data compression
		  ○ Data integrity
		● Non-von Neumann Architectures
		  ○ In-Memory Processing (PIM)
				CS	6

		● Cache memories [Shared with OS]
		  ○ Address mapping
		  ○ Block size
		  ○ Replacement and store policy
		● Multiprocessor cache coherence
			Evaluate		

AR	Interfacing and Communication	● I/O fundamentals[Shared with OS and SPD]
		  ○ Handshaking and buffering
		  ○ Programmed I/O
		  ○ Interrupt-driven I/O
		● Interrupt structures: vectored and prioritized, interrupt acknowledgement [Shared with OS]
		● External storage, physical organization and drives [Shared with OS]
		● Buses fundamentals [Shared with OS and SPD]
		  ○ Bus protocols
		  ○ Arbitration
		  ○ Direct-memory access (DMA)
		● Network-on-chip (NoC)
			Explain	CS	1

AR	Functional Organization	● Implementation of simple datapaths, including instruction pipelining, hazard detection and resolution
		● Control unit
		  ○ Hardwired implementation
		  ○ Microprogrammed realization
			Develop	KA	2

		● Instruction pipelining ● Introduction to instruction-level parallelism (ILP)	Explain		
AR	Performance and Energy Efficiency	● Performance-energy evaluation (introduction): performance, power			

		consumption, memory and communication costs ● Branch prediction, speculative execution, out-of-order execution, Tomasulo's algorithm ● Prefetching	Evaluate	KA	2
AR	Performance and Energy Efficiency	● Enhancements for vector processors and GPUs ● Hardware support for Multithreading ○ Race conditions ○ Lock implementations ○ Point-to-point synchronization ○ Barrier implementation ● Scalability ● Alternative architectures, such as VLIW/EPIC, accelerators and other special-purpose processors ● Dynamic voltage and frequency scaling (DVFS) ● Dark Silicon	Explain	KA	1
AR	Heterogeneous Architectures	● SIMD and MIMD architectures (e.g. General-Purpose GPUs, TPUs and NPUs) ● Heterogeneous memory system ○ Shared memory versus distributed memory ○ Volatile vs non-volatile memory ○ Coherence protocols ● Domain-Specific Architectures (DSAs) ○ Machine Learning Accelerator ○ In-networking computing ○ Embedded systems for emerging applications ○ Neuromorphic computing ● Packaging and integration solutions such as 3DIC and Chiplets	Explain	KA	3
AR	Quantum Architectures	● Principles ● The wave-particle duality principle ● The uncertainty principle in the double-slit experiment ● What is a Qubit? Superposition and measurement. Photons as qubits. ● Systems of two qubits. Entanglement. Bell states. The No-Signaling theorem.	Explain	KA	3

		● Axioms of QM: superposition principle, measurement axiom, unitary evolution
		● Single qubit gates for the circuit model of quantum computation: X, Z, H.
		● Two qubit gates and tensor products. Working with matrices.
		● The No-Cloning Theorem. The Quantum Teleportation protocol.
		● Algorithms
		  ● Simple quantum algorithms: Bernstein-Vazirani, Simon's algorithm.
		  ● Implementing Deutsch-Josza with Mach-Zehnder Interferometers.
		  ● Quantum factoring (Shor's Algorithm)
		  ● Quantum search (Grover's Algorithm)
		● Implementation aspects
		  ● The physical implementation of qubits (there are currently nine qubit modalities)
		  ● Classical control of a Quantum Processing Unit (QPU)
		  ● Error mitigation and control. NISQ and beyond.
		● Emerging Applications
		  ● Post-quantum encryption
		  ● The Quantum Internet
		  ● Adiabatic quantum computation (AQC) and quantum annealing			