Course number and name: **CS 06205: Computer Organization**
Credits and contact hours: 3 credits / 3 contact hours
Course Coordinator: Nikolay "Nick" Ivanov
Instructional Materials:
1. David A. Patterson & John L. Hennessy *Computer Organization and Design - The Hardware/Software Interface. 6th Edition.* ISBN-13: 978-0-12-820109-1
2. Stephen Smith *Programming with 64-Bit ARM Assembly Language* ISBN-13: 978-1-4842-5880-4 ISBN-13 (electronic): 978-1-4842-5881-1

Specific course information:

Catalog description: This course provides an introduction to computer organization. Students are exposed to the register-level architecture of a modern computer and its assembly language. The topics include principles of computer organization, concepts and abstractions of computer organization, numbers and data in computers, low-level data flow, control flow, assembly programming, processor, memory, and computer hardware.

Prerequisites: (MATH 03160 Discrete Structures or MATH 03150 Discrete Mathematics) **and** (CS 04103 Computer Science and Programming or CS 04113 Introduction to Object Oriented Programming) **and** Sophomore standing

Type of Course: ☒ Required ☐ Elective ☐ Selected Elective

Educational objectives for the course:

1. **arithmetic for computers.** The student has demonstrated understanding of the representation of numbers and arithmetic algorithms.
   - ABET (2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.

2. **computer instructions.** The student has demonstrated an understanding of the instruction set architecture of a contemporary computer
   - ABET (2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.

3. **CPU and datapath.** The student has demonstrated understanding of the design and operation of a simple single-cycle processor.
    - ABET (2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.

4. **memory hierarchy.** The student has demonstrated understanding of the design and operation of a cache memory system.
    - ABET (2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.

5. **low level programming**. The student has applied low level programming languages to implement complex programs such as internal operating system components and drivers to interface with and control hardware devices or to achieve other results (speed, size, efficiency, etc.).
    - ABET (2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.

6. **low level programming risks**. The student has explained the risks and rewards that result from using low level programming.
    - ABET (2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.

7. **Role of hardware and software in computing devices.** The student understands the big picture of computer organization, specifically how the hardware implements what is written in the software.
    - ABET (2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.

Required list of topics to be covered:

1. **Principles, Concepts and Abstractions**
    - 1.1. The great ideas of computer organization
    - 1.2. Layers of abstraction
    - 1.3. Hardware-software interface
    - 1.4. The stored program concept
    - 1.5. Brief history and comparison of ISAs, CISC/RISC dichotomy
2. **Numbers and Data in Computers**
    - 2.1. Binary data
    - 2.2. Storage and throughput units
    - 2.3. Multiple interpretations of binary data, ASCII table
    - 2.4. Hexadecimal format and binary-hexadecimal conversion
    - 2.5. Big Endian, Little Endian, and Bi-Endian

2.6. Integers and integer ranges, signed and unsigned integers
2.7. Signed integer formats: sign and magnitude, biased notation, one's complement, and two's complement
2.8. Bitwise logical operations
2.9. Bitwise masks and their applications
2.10. Left and right binary shifts
2.11. Two's complement integer operations
2.12. Floating point numbers and IEEE-754 format
2.13. IEEE-754 single and double precisions: ranges, accuracy, storage, and performance
2.14. IEEE-754 components: sign, biased exponent, non-normalized mantissa (fraction)
2.15. IEEE-754 encoding and decoding

3. Low-Level Data Flow and Control Flow

3.1. Compilation, interpretation, assembly and cross-compiling
3.2. Object code, libraries, modularization and linking
3.3. Loading and executing the program
3.4. Translation of a high-level language into assembly language
3.5. Referencing registers, data and code literals
3.6. Instructions' mnemonics and pseudoinstructions
3.7. Integer arithmetic operations
3.8. Memory segments and assembler directives
3.9. Control flow and branching
3.10. Labels, addresses and unconditional jumps
3.11. Branching conditions and conditional jumps
3.12. Loops
3.13. Functions/procedures and long jumps
3.14. Function arguments and return value
3.15. Input-output
3.16. Floating point arithmetic operations
3.17. OS system calls in assembly language
3.18. Debugging low-level programs

4. The Processor

4.1. Fetch-Decode-Execute cycle
4.2. Computer instructions, opcodes, and instruction formats
4.3. Clock cycle and frequency
4.4. Control and datapath, pipelining
4.5. CPU registers

5. Computer Memory

5.1. Primary and secondary memory
5.2. Memory hierarchy and technologies
5.3. Stack, heap, and text
5.4. Analyzing high-level computer program via the concepts of stack, heap, and code segmentation
5.5. Caching and levels of cache

6. Hardware

6.1. Logical gates and ALUs

6.2. Adders, multiplexers and memory gates

Optional list of topics that could be covered:

1. Common challenges, misconceptions and pitfalls of computer organization
2. Moore's law and its relevance in the modern world
3. The walls of performance and power
4. Performance measurements and bottlenecks
5. Cross-architecture and cross-platform solutions
6. Locality principles in cache: temporal locality and spatial locality
7. Cache mapping modalities: direct mappings, associative mappings
8. Memory pages and virtual memory
9. Computer graphics and GPUs
10. Using GPUs for high-performance computing
11. Serial and parallel system buses
12. Peripheral devices and interruption requests
13. Cluster and grid computing
14. Arithmetic coprocessors
15. Symmetric multiprocessing (SMP) and multi-core CPUs
16. Silicon chips and their manufacturing process
17. Frontiers of Computer Organization: Quantum computing architectures
18. Frontiers of Computer Organization: Distributed virtual machines (use case: EVM)
19. Frontiers of Computer Organization: Trusted execution environments (TEE)