COE 301: Computer Organization
Fall 2020 – Term 201

College of Computer Sciences & Engineering
King Fahd University of Petroleum & Minerals

Professor:	Muhamed Mudawar, Room 22/410-2, Phone 4642
Office Hours:	UTR 11 am – 11:50 am or by appointment
Course URL:	http://faculty.kfupm.edu.sa/coe/mudawar/coe301/
Email:	mudawar@kfupm.edu.sa

Catalog Description

Introduction to computer organization, machine instructions, addressing modes, assembly language programming, integer and floating-point arithmetic, CPU performance and metrics, non-pipelined and pipelined processor design, datapath and control unit, pipeline hazards, memory system and cache memory. Prerequisites: COE 202 and ICS 102.

Textbook

David A. Patterson and John L. Hennessy, *Computer Organization and Design: The Hardware / Software Interface*, Fifth Edition, Morgan Kauffmann Publishers, 2013.

Course Learning Outcomes

After successfully completing the course, students will be able to:

1. Describe the instruction set architecture of a MIPS processor
2. Analyze, write, and test MIPS assembly programs
3. Describe organization and operation of integer and floating-point arithmetic units
4. Design the datapath and control of a single-cycle (non-pipelined) CPU
5. Design the datapath and control of a pipelined CPU and handle hazards
6. Describe the organization and operation of memory and caches
7. Analyze the performance of processors and caches

Grading Scheme

MIPS Programming	12%	
Quizzes	10%	
Lab Work	18%	
Project	15%	
Midterm Exam	20%	**Saturday, October 31, 2020, at 10 AM**
Final Exam	25%	**To Be Announced**

- Attendance will be taken regularly. The tenth unexcused absence results in a DN grade.
- Late MIPS programming assignments will not be accepted.
- A student caught cheating in a quiz or exam will get F in the course.
- No makeup will be given for missing quizzes or exams.

Weekly Breakdown

Week	Topics
1	• Introduction to computer organization, high-level, assembly, and machine languages. Classes of computers, components of a computer system, fetch-execute cycle, technology improvements, programmer's view of a computer system.
2	• Introduction to assembly language programming, instructions, registers, assembly language statements, directives, text, data, and stack segments. Defining data, arrays, and strings. Memory alignment, byte ordering, and symbol table. System calls, console input and output.
3	• Integer storage sizes, review of binary addition and subtraction, carry and overflow. • MIPS instruction set architecture, instruction formats, R-type integer arithmetic, logic, and shift instructions, immediate operands, I-type arithmetic and logic instructions, pseudo-instructions.
4	• Control flow, branch and jump instructions, translating if-else statements and logical expressions. Compare instructions, and conditional-move instructions. • Arrays, allocating arrays statically in the data segment and dynamically on the heap, computing the memory addresses of array elements.
5	• Load and store instructions, translating loops, using pointers to traverse arrays, addressing modes, jump and branch limits. • Unsigned and signed binary multiplication (paper and pencil only), fast hardware multipliers, carry-save adders in hardware multipliers. • Unsigned and signed binary division (paper and pencil only).
6	• MIPS Integer multiply and divide instructions, Integer to string conversion and vice-versa. • Defining functions (procedures) in assembly language, function call and return instructions. Passing arguments by value and by reference in registers, and the return address register.
7	• The stack segment, allocating and freeing stack frames, leaf versus non-leaf functions, preserving registers across function calls. Allocating and referencing a local array on the stack. Bubble Sort example and its translation into assembly code. • Recursive functions, translating recursive functions into assembly language.
8	• Floating point representation, IEEE 754 standard, de-normalized numbers, zero, infinity, NaN. • FP comparison, FP addition, FP multiplication, rounding and accurate arithmetic.
9	• MIPS floating-point instructions: load/store, arithmetic, data movement, convert, compare, branch, FP system calls. Floating-point programs. Example on Matrix Multiplication. • Designing a processor, register transfer level, datapath components, clocking methodology.
10	• Implementing a register file and multifunction ALU • Assembling a single-cycle datapath from its components • Control signals and control unit, ALU control, and PC control.
11	• CPU performance and metrics, CPI of a multi-cycle processor, performance equation, performance comparison of a single-cycle versus a multi-cycle processor, MIPS as a metric, Amdahl’s law, energy and power consumption, benchmarks.
12	• Drawback of single-cycle processor, single-cycle versus multicycle delay analysis and clock cycle. Pipelining versus serial execution, timing diagrams, MIPS 5-stage pipeline, pipelined datapath, pipelined control, pipeline performance.
13	• Pipeline hazards: structural, data, and control hazards, load delay, hazard detection, stall and forwarding unit, delayed branching, and branch prediction.
14	• Main memory organization, SRAM vs DRAM storage cells, DRAM refresh cycles, latency and bandwidth, trends in DRAMs, memory hierarchy, cache memory, locality of reference. • Cache memory organization: direct-mapped, fully-associative, and set-associative caches, handling cache miss, write policy, write buffer, and replacement policy.
15	• Cache performance, memory stall cycles, and average memory access time. Introduction to multi-level caches, multi-level cache performance.