COURSE DESCRIPTION

| | | | |
|---|---|---|---|
| DEPARTMENT AND COURSE NUMBER | CMPS 430 | **Course Coordinator** | Magdy Bayoumi |

| | | | |
|---|---|---|---|
| **Course Title** | Computer Architecture | **Total Credits** | 3 |
| **URL** | | **Semester hours** | 3 |

**Current Bulletin Description**: Hierarchical multilevel structure of computer systems; instruction set; microprogrammed and hardwired control; memory; basics of pipelines and multiprocessors; performance evaluations; I/O organization; buses and channels; computer arithmetic. Prereq: CMPS 351 with a minimum grade of C.

**Textbook**

John L. Hennessey and David A. Patterson, Computer Organization and Architecture, 3rd Edition, Morgan Kaufmann Publishers, Inc., 2004, ISBN 1-55860-604-1.

**References**

1. Several tutorials, white papers, and handouts on the latest updates on the topics of study. These are provided to the students by the instructor and also a soft-copy is made available in Moodle so that students can access any them time.

2. IEEE Verilog Language Reference Manual.

**Course Goals**

- To gain an understanding of overall architecture of a modern computer
- To gain experience in designing, simulating, and verifying a datapath component
- To gain experience in designing, simulating, and verifying a controller
- To gain an understanding of basic as well as advanced concepts in pipelining
- To gain an understanding of memory mappings and their effect on performance
- To learn RAID systems and how they contribute to greater throughput for disk access
- To understand synchronous and asynchronous communications and study types of buses
- To develop reading, writing, and oral presentation skills pertaining to an emerging industrial topic of interest in computer architecture

**Course outcomes**

- Students have an understanding of components and their interconnection in a RISC processor
- Students are able to write software using Verilog Hardware Description Language to model, synthesize and simulate computational components such as adders/subtractors, and arithmetic logic units
- Students are able to schedule an assembly code in a pipelined datapath both without and with forwarding and they are able to compute performance gain achieved due to forwarding
- Students are able to identify hazards in a pipeline and learn how to overcome them
- Students have an understanding of direct-mapped and set-associative cache-to-memory mappings and are able to compute size, tag-bits, and speed concerned with each of these mapping.

- Students have an understanding of usage of RAID systems RAID0-RAID6, and they are knowledgeable with tradeoffs of speed, reliability, and throughput in each of these systems
- Students have an understanding of synchronous versus asynchronous communication as well as bus arbitration mechanisms such as daisy chaining and centralized arbitration
- Students are able to read, discuss and understand industrial papers on an emerging topic and are able to write a report reflecting their understanding
- Students are able to orally present their chosen topic to the class and they are able to answer questions
- Students have experience working in groups to design and develop finite state machine based controllers

**Prerequisites by Topic**

- Knowledge of MIPS assembly language programming (CMPS 351)
- Basic knowledge of Boolean algebra, logic gates, and multiplexers (EECE 140)
- Working knowledge of Unix and/or Windows

**Major Topics Covered in the Course**

| | |
|---|---|
| • Datapath and its construction using logic-hardware | 4 classes |
| • Modeling with Verilog using Icarus | 3 classes |
| • Pipelining, forwarding, hazard detection | 4 classes |
| • Caches, and cache-to-memory mapping techniques | 4 classes |
| • RAID systems | 3 classes |
| • Input, output and synchronous/asynchronous communication | 2 classes |
| • Bus arbitration methods | 2 classes |
| • Designing a controller as a finite state machine | 2 classes |
| • Multiprocessors and clusters | 3 classes |

**Laboratory projects** (specify number of weeks on each)

| | |
|---|---|
| • Design of a simple computational element | (2 weeks) |
| • Design and modeling of an ALU | (2 weeks) |
| • FSM design of a controller | (3 weeks) |
| • Verilog implementation of the project | (3 weeks) |

**Oral and Written Communications**

Students write a 5-page report on one of the emerging topics (of industrial interest) recommended by the instructor. They also make an oral presentation on the same topic to the class, thus developing their communication skills. Further, throughout the semester, students' class interaction is encouraged.

**Social and Ethical Issues**

While discussing the performance and reliability issues, the societal impact of good architectural design is discussed in terms of cost effectiveness and criticality of real-life applications.

**Theoretical Content**

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

- Theory of pipelining (2 weeks)
- Theory of memory mapping (2 weeks)
- Theory of RAID systems (1 week)

**Problem Analysis**

Please describe the analysis experiences common to all course sections.

Students analyze computational structure in order to develop a hardware description language based model. For example, while designing a 4-bit adder, they analyze how a 1-bit addition is done on binary numbers and come up with a Boolean equation defining such operation. Then, to build a 4-bit adder from 1-bit adders, they must analyze how carry propagation takes effect and how subsequent stages of addition use results from previous stage, etc. Another, more rigorous, example is when students are assigned projects which describe a real-life problem for which a controller is to be designed. The students must understand provided descriptions and analyze them in order to develop a finite state machine based design.

**Solution Design**

Please describe the design experiences common to all course sections.

Students' projects concern design of controllers which can be used in a real-life application (such as a vending machine, a traffic-light controller, a drawbridge controller, a drag-race detector, etc.). The students come up with a finite state machine which can realize such controller. Upon correction and approval of their devised solutions, students proceed with implementation of their design in Verilog HDL. Their solutions are tested using test vectors.