CS402	Computer Organization & Architecture	3L:0T: 4P	5 credits	Pre-Reqs: ESC302

Learning Outcomes of the course (i.e. statements on students' understanding and skills at the end of the course the student shall have):

Essential (<=6):

1. The key components of a basic computer
2. The key components of a CPU and how the instructions are executed
3. Execution and time taken by instructions in a pipelined processor
4. The need for memory hierarchy and efficiency achieved due to the use of cache
5. How the data is stored and input-output is performed in computers

Desirable/Advanced (<= 3):

1. Super-scalar and multi-core architectures for crossing one clock per instruction barrier
2. Cache data coherence related challenges in multi-core processors

Detailed contents for Essential Learning Outcomes:

Module (appx dur in wks)	Topics	Pedagogy / teaching suggestions	Nature of lab / assignment / practice
Module 1: Introduction (Lectures 6) (Weeks 2)	Role of abstraction, basic functional units of a computer, Von-Neumann model of computation, A note on Moore's law, Notion of IPC, and performance. Data representation and basic operations.		
Module 2: Instruction Set Architecture (RISC-V) (Lectures 8/9) (Weeks 3)	CPU registers, instruction format and encoding, addressing modes, instruction set, instruction types, instruction decoding and execution, basic instruction cycle, Reduced Instruction Set Computer (RISC), Complex Instruction Set Computer (CISC), RISC-V instructions; X86 Instruction set.		Lab Modules 1 and 4
Module 3: The Processor (Lectures 6) (Weeks 2)	Revisiting clocking methodology, Amdahl's law, Building a data path and control, single cycle processor, multi-cycle processor, instruction pipelining, Notion of ILP, data and control hazards and their mitigations.		Lab Modules 2 and 4
Module 4: Memory hierarchy (Lectures 8/9) (Weeks 3)	SRAM/DRAM, locality of reference, Caching: different indexing mechanisms, Trade-offs related to block size, associativity, and cache size, Processor-cache interactions for a read/write request, basic optimizations like write-through/write-back caches, Average memory access time, Cache replacement policies (LRU), Memory interleaving.		Lab Module 2

Module 5: Storage and I/O (Lectures 6) (Weeks 2)	Introduction to magnetic disks (notion of tracks, sectors), flash memory. I/O mapped, and memory mapped I/O. I/O data transfer techniques: programmed I/O, Interrupt-driven I/O, and DMA.		Lab Modules 3 and 5

Detailed Contents for Desirable Learning Outcomes (optional, <= 3 modules):

Module	**Topics**	**Pedagogy teaching suggestions**	**Nature of lab / assignment / practice**
Module 6: Superscalar processors and multicore systems (Lectures 6) (Weeks 2)	Limits of ILP, SMT processors, Introduction to multicore systems and cache coherence issues		

Lab Modules

The laboratory component consists of 5 modules out of which three can be chosen based on the overall curriculum and its emphasis. Modules 4 and 5 assume that students have a background in using FPGA kits in their Digital Electronics course (ESC302) and have also been introduced in programming in one of the HDLs (VHD or Verilog)

Possible three options are

1. Module 1 + Module 2 + Module 3
   Instructions & assembly language + basic performance + advanced performance analysis

2. Architecture + Module 1 + Module 2 + Module 4
   Instructions & assembly language + basic performance + basic processor design

3. Module 1 + Module 4 + Module 5
   Instructions & assembly language + basic processor design + I/O and architecture enhancements

Module (appx dur in wks)	**Topics**	**Comments**
Module 1: (Weeks 4) Objective: Understanding architecture and instructions through assembly programming	Write programs in ARM/RISC V assembly language and test these on an instruction set simulator. Typical examples are given below. Some of these are dependent on I/O facilities provided by the simulator. ● Generate some interesting numbers (example - Happy numbers,	Essential component

	Autonomic numbers, Hardy-Ramanujan numbers etc.) ● Implement a 4-function calculator ● Sort an integer array using merge sort (recursive) ● Evaluate an arithmetic expression specified as a string (using recursive functions) ● Implement a simple game	
Module 2: Understanding performance issues related to pipelining and cache using architectural simulator (Weeks 4/5)	Usage of a instruction pipeline visualization tool like RIPES Write or generate sequence of instructions and observe the overall pipeline stalls with and without data hazards, control hazards, and with/without data forwarding. Rearrange the sequence of instructions or the program so that the pipeline stalls will be minimized.	Optional
Module 3: Understanding memory access patterns and changing basic cache memory parameters to analyze the impact of standard programs or benchmarks using architectural simulators. (Weeks 3)	Configure the simulator [gem5 is preferred] to operate on the binaries of the benchmark as the input. Run the program and examine the IPC, cache hit rate, number of conflict misses and block replacements. Vary the cache size, block size, and associativity and analyze the metrics and reason the changes observed. Modify the block replacement algorithms and see the impact at cache memory performance Calculate the access time, power and are associated with a given cache configuration. Vary the cache size, block size, and associativity and analyze the metrics and reason the changes observed.	Optional: Familiarity with tools like GEM5, CACTI and PIN, and access to benchmarks like SPEC, PARSEC, SPLASH. Any other tools/simulators that can support memory pattern analysis is also fine.
Module 4: (Weeks 4/5) Objective: Understanding computer architecture by designing a CPU on FPGA kit	Design a simple ARM/RISC V processor for a small subset of instructions and implement on FPGA board. This is done in stages as follows. ● Design CPU for the instruction subset {add, sub, cmp, mov, ldr, str, beq,	Optional Prerequisite: Use of FPGA kits as well as exposure to VHDL/Verilog

	bne, b}, with each instruction executing in a single cycle. No sequential control required. For each instruction, only a limited set of variants are considered. ● Use memory generator to add program and data memories. ● Include circuit for single step execution and for displaying signals of interest. ● Modify the design to allow multi-cycle execution of instructions. ● Enhance the design to include all DP instructions and all variants of the second operand.	
Module 5: Objective: Understanding computer architecture performance issues as well as I/O through architectural enhancements (on FPGA) (Weeks 4/5)	Extension of the CPU design and I/O programming ● Enhance the design to include all variants of DT instructions. ● Implement multiply group of instructions. ● Enhance the design to implement "branch and link" instruction and include full predication. ● Include limited exception handling. ● Interface 7-segment display and 4x4 keypad. ● Demonstrate execution of simple programs.	Optional Prerequisite: Use of FPGA kits as well as exposure to VHDL/Verilog

Suggested text books / Online lectures or tutorials:

1. "Computer Organization and Design: The Hardware/Software Interface", David A. Patterson and John L. Hennessy, 5th Ed, Elsevier,

Suggested reference books / Online resources:

2. "Computer Organisation & Architecture", Smruti Ranjan Sarangi, McGraw Hill
3. "Computer System Architecture", Mano M. Morris, Pearson.
4. "Computer Organization and Embedded Systems", 6th Edition by Carl Hamacher, McGraw Hill Higher Education
5. "Computer Architecture and Organization", 3rd Edition by John P. Hayes, WCB/McGraw-Hill
6. "Computer Organization and Architecture: Designing for Performance", 10th Edition by William Stallings, Pearson Education.
7. "Computer System Design and Architecture", 2nd Edition by Vincent P. Heuring and Harry F. Jordan, Pearson Education.

Online simulators and tools for labs:

RIPES: https://freesoft.dev/program/108505982

GEM5: https://www.gem5.org/documentation/learning_gem5/introduction/

CACTI: https://github.com/HewlettPackard/cacti

PIN: https://www.intel.com/content/www/us/en/developer/articles/tool/pin-a-binary-instrumentation-tool-downloads.html

TEJAS: https://www.cse.iitd.ac.in/~srsarangi/archbooksoft.html

XILINX (VHDL/Verilog tools): https://www.xilinx.com/support/university/students.html

The Discipline Graduate Attributes (GAs) to which this course contributes significantly: <CS, CE and EE>

Prepared by:

1. Prof. Anshul Kumar, IIT Delhi
2. Prof. Anupam Basu, IIT Kharagpur/NIT Durgapur
3. Prof. Indranil Sengupa, IIT Kharagpur
4. Prof. Biswabandan Panda, IIT Bombay
5. Prof. John Jose, IIT Guwahati
6. Prof. M. Balakrishnan, IIT Delhi