THOMPSON RIVERS UNIVERSITY

Course Outline

Computing Science Department

Faculty of Science

COMP 2130 – 3 Credits
Introduction to Computer Systems (3,1,0)
Fall 2015

Instructor:
Office:
Office Hours:

Phone/Voice Mail:
E-Mail:

CALENDAR DESCRIPTION

Students learn the basic concepts of computer systems. Students are introduced to the concepts of computer architecture, the 'C' and assembly programming languages as well as the use of Linux operating system. Students learn about memory organization, data representation, and addressing. Students are introduced to the concepts of machine language, memory, caches, virtual memory, linkage and assembler construction as well as exceptions and processes.

PREREQUISITES

- COMP 1230 or COMP 2120, and
- COMP 1380 or MATH 1700

EDUCATIONAL OBJECTIVES/OUTCOMES

Upon successful completion of the course, the student will demonstrate the ability to:

- Understand the fundamentals of computer architecture.
- Familiar with programming through the powerful C programming language .
- Familiar with programming through assembly language.
- Understand critical relationship between programming and computer architecture.
- Understand the efficient programming through code optimization.

TEXTS/MATERIALS

The course uses the following texts:

- **B1: Bryant** and **O'Hallaron,** Computer Systems: A Programmer's Perspective, 2/E, Prentice Hall, 2011
  ISBN 10: 0-13-610804-0
- **B2: Kernighan** and **Ritchie**, The C Programming Language, Prentice Hall, 1988, ISBN 10: 0-13-110362-8
- **B3: David A. Patterson** and **John L. Hennessy**, Computer Organization and Design, Morgan Kaufmann - Fifth Edition, 2013, ISBN-13: 978-0124077263

SYLLABUS - Lecture & Lab Topics:

Course Topics		Reference to Text Book*	Duration**
1. Basic concepts of digital systems:			
	1.1 Introduction	B3-Appendix B.1	1.5 weeks
	1.2 Gates, Truth Tables, and Logic Equations	B3-Appendix B.2	
	1.3 Combinational Logic	B3-Appendix B.3	
	1.4 Using HW Description Language	B3-Appendix B.4	
	1.5 Clocks	B3-Appendix B.7	
	1.6 Memory Elements	B3-Appendix B.8	
	1.7 SRAMs and DRAMs	B3-Appendix B.9	
	1.8 FSM	B3-Appendix B.10	
2. Computer Abstraction and Technology			
	2.1 Introduction	B3-Chapter 1:1.1-1.2	1 week
	2.2 The compilation system	B3-Chapter 1:1.3 & B1-Chapter 1:1.2-1.3	
	2.3 Computer Architecture and Processor Technology	B3- Chapter 1:1.4-1.5 & B1-Chapter 1:1.4-1.6	
	2.3 How OS manages HW	B1-Chapter 1:1.7-1.8	
3. Introduction to Linux OS and C Language			
	3.1 Getting Familiar with Linux	Instructor notes	2.5 weeks
	3.2 Introduction to C Programming	B2-Chapter 1-4	
	3.3 Advanced C Programming	B2-Chapter 5-8	
4. Memory Organization, Data representation, and Addressing			
	4.1 Information Storage	B1-Chapter 2:2.1	1 week
	4.2 Integer Representation and Floating Point Numbers	B1-Chapter 2:2.2&2.4	
5. Basics of Architecture, Machine Code			
	5.1 A Historical Perspective	B1-Chapter 3:3.1	1 week
	5.2 Program Encodings	B1-Chapter 3:3.2	
	5.3 Data Formats	B1-Chapter 3:3.3	
6. Machine Level Programming			
	6.1 Arithmetic and Logical Operations	B1-Chapter 3:3.5	1 week
	6.2 Control	B1-Chapter 3:3.6	
	6.3 Procedures	B1-Chapter 3:3.7	
	6.4 Arrays	B1-Chapter 3:3.8	
7. Memory and Caches			
	7.1 Storage Technologies	B1-Chapter 6:6.1	2 week
	7.2 Locality	B1-Chapter 6:6.2	
	7.3 The Memory Hierarchy	B1-Chapter 6:6.3	
	7.4 Cache Memories	B1-Chapter 6:6.4	
	7.5 Writing Cache-friendly Code	B1-Chapter 6:6.5	
	7.6 Putting It Together: The Impact of Caches on Program Performance	B1-Chapter 6:6.6	

8. Linking			
	8.1 Compiler Drivers	B1-Chapter 7:7.1	2 week
	8.2 Static Linking	B1-Chapter 7:7.2	
	8.3 Object Files	B1-Chapter 7:7.3	
	8.4 Relocatable Object Files	B1-Chapter 7:7.4	
	8.5 Symbols and Symbol Tables	B1-Chapter 7:7.5	
	8.6 Symbol Resolution	B1-Chapter 7:7.6	
	8.7 Relocation	B1-Chapter 7:7.7	
	8.8 Executable Object Files	B1-Chapter 7:7.8	
	8.9 Loading Executable Object Files	B1-Chapter 7:7.9	
	8.10 Dynamic Linking with Shared Libraries	B1-Chapter 7:7.10	
	8.11 Loading and Linking Shared Libraries from Applications	B1-Chapter 7:7.11	
9. Exceptions and Processes			
	9.1 Exceptions	B1-Chapter 8:8.1	0.5 week
	9.2 Processes	B1-Chapter 8:8.2	
	9.3 System Call Error Handling	B1-Chapter 8:8.3	
	9.4 Process Control	B1-Chapter 8:8.4	
10. Virtual Memory			
	10.1 Physical and Virtual Addressing	B1-Chapter 9:9.1	0.5 week
	10.2 Address Spaces	B1-Chapter 9:9.2	
	10.3 VM as a Tool for Caching	B1-Chapter 9:9.3	
	10.4 VM as a Tool for Memory Management	B1-Chapter 9:9.4	

*** This is just a reference to the reading material.**
**** Tentative duration.**

Lab Topics
Introduction to Linux & Setting up the VM
C Programming Labs
Basic assembly language programming
Advanced C Programming Labs
Testing and Debugging

ACM / IEEE Knowledge Area Coverage

IEEE Knowledge Areas that contain topics and learning outcomes covered in the course

Knowledge Area	Total Hours of Coverage
AR/Digital Logic and Digital Systems	3
AR/Machine Level Representation of Data	3
AR/Assembly Level Machine Organization	6
AR/Memory System Organization and Architecture	3
AR/Interfacing and Communication	1
PL/Basic Type Systems	2.5
SF/Computational Paradigms	3
SF/State and State Machines	3
SF/Evaluation	3

IEEE Body of Knowledge coverage

KA	Knowledge Unit	Topics Covered	T1 hours	T2 hours	Elective hours
AR	Digital Logic and Digital Systems	• Overview and history of computer architecture • Combinational vs. sequential logic/Field programmable gate arrays as a fundamental combinational + sequential logic building block • Multiple representations/layers of interpretation (hardware is just another layer) • Computer-aided design tools that process hardware and architectural representations • Register transfer notation/Hardware Description Language (Verilog/VHDL) • Physical constraints (gate delays, fan-in, fan-out, energy/power)	0	3	0

KA	Knowledge Unit	Topics Covered	T1 hours	T2 hours	Elective hours
AR	AR/Machine Level Representation of Data	• Bits, bytes, and words • Numeric data representation and number bases • Fixed- and floating-point systems • Signed and twos-complement representations • Representation of non-numeric data (character codes, graphical data) • Representation of records and arrays	0	3	0
AR	AR/Assembly Level Machine Organization	• Basic organization of the von Neumann machine • Control unit; instruction fetch, decode, and execution • Instruction sets and types (data manipulation, control, I/O) • Assembly/machine language programming • Instruction formats • Addressing modes • Subroutine call and return mechanisms (cross-reference PL/Language Translation and Execution) • I/O and interrupts • Heap vs. Static vs. Stack vs. Code segments • Shared memory multiprocessors/multicore organization	0	6	0
AR	AR/Memory System Organization and Architecture	• Storage systems and their technology • Memory hierarchy: importance of temporal and spatial locality • Main memory organization and operations • Latency, cycle time, bandwidth, and interleaving • Cache memories (address mapping, block size, replacement and store policy) • Multiprocessor cache consistency/Using the memory system for inter-core synchronization/atomic memory operations • Virtual memory (page table, TLB) • Fault handling and reliability • Error coding, data compression, and data integrity (cross-reference SF/Reliability through Redundancy)	0	3	0
AR	AR/Interfacing and Communication	• I/O fundamentals: handshaking, buffering, programmed I/O, interrupt-driven I/O • Interrupt structures: vectored and	0	1	0

		prioritized, interrupt acknowledgment • External storage, physical organization, and drives • Buses: bus protocols, arbitration, direct-memory access (DMA) • Introduction to networks: communications networks as another layer of remote access			
PL	PL/Basic Type Systems	o type as a set of values together with a set of operations o Primitive types (e.g., numbers, Booleans) o Compound types built from other types (e.g., records, unions, arrays, lists, functions, references) • Association of types to variables, arguments, results, and fields • Type safety and errors caused by using values inconsistently given their intended types • Goals and limitations of static typing o Eliminating some classes of errors without running the program o Undecidability means static analysis must conservatively approximate program behavior Generic types (parametric polymorphism) o Definition o Use for generic libraries such as collections o Comparison with ad hoc polymorphism (overloading) and subtype polymorphism • Complementary benefits of static and dynamic typing o Errors early vs. errors late/avoided Enforce invariants during code development and code maintenance vs. postpone typing decisions while prototyping and conveniently allow flexible coding patterns such as heterogeneous collections o Avoid misuse of code vs. allow more code reuse o Detect incomplete programs vs. allow incomplete programs to run	0.5	2	0
SF	SF/Computational Paradigms	• Basic building blocks and components of a computer (gates, flip-flops, registers, interconnections;	3	0	0

		Datapath + Control + Memory) • Hardware as a computational paradigm: Fundamental logic building blocks; Logic expressions, minimization, sum of product forms • Application-level sequential processing: single thread • Simple application-level parallel processing: request level (web services/client-server/distributed), single thread per server, multiple threads with multiple servers • Basic concept of pipelining, overlapped processing stages • Basic concept of scaling: going faster vs. handling larger problems			
SF	SF/State and State Machines	• Digital vs. Analog/Discrete vs. Continuous Systems • Simple logic gates, logical expressions, Boolean logic simplification • Clocks, State, Sequencing • Combinational Logic, Sequential Logic, Registers, Memories • Computers and Network Protocols as examples of state machines	3	0	0
SF	SF/Evaluation	• Performance figures of merit • Workloads and representative benchmarks, and methods of collecting and analyzing performance figures of merit • CPI (Cycles per Instruction) equation as tool for understanding tradeoffs in the design of instruction sets, processor pipelines, and memory system organizations. • Amdahl's Law: the part of the computation that cannot be sped up limits the effect of the parts that can	3	0	0