COMPUTER ALGORITHMS AND ARCHITECTURE - 2024/5

Module code: EEE2048

Module Overview

Module purpose: this module is organized into two parts that run concurrently. Part A introduces the students to microprocessors. This covers the key concepts in microprocessor organization and design; specifically for the instruction set, performance analysis, the arithmetic logic unit (ALU), and the processor control and data paths. Additionally, we explore common memory hierarchies and caching problems. In class problems are given as examples in design.

Part B covers the analysis, design and implementation of computer algorithms. It presents concepts and methods for the analysis of algorithms. Classic programming techniques and data-structures needed to develop efficient algorithms in C for solving logical and data-handling problems are introduced, and students will attend programming lab sessions where they have the opportunity to implement in C the algorithms that have been covered.

This module has strong connections with a number of modules within the curriculum. The module directly builds on the Year 1 modules which establish a foundation in programming (EEE1033 and EEE1035). This module uses C as the main programming language, thus providing continuity with the first year where it was introduced. The module also prepares students for subsequent modules. This includes the Year 2 Semester 2 modules concerned with object-oriented programming (EEE2047) and computer vision & graphics (EEE2041) as well as specialist modules in Year 3 such as Computer Vision and Pattern Recognition (EEE3032), Digital Design with VHDL (EEE3027), Robotics (EEE3243), etc.

Module provider

Computer Science and Electronic Eng

Module Leader

GUILLEMAUT Jean-Yves (CS & EE)

Number of Credits: 15

ECTS Credits: 7.5

Framework: FHEQ Level 5

Module cap (Maximum number of students): N/A

Overall student workload

Independent Learning Hours: 90

Lecture Hours: 10

Tutorial Hours: 10

Laboratory Hours: 10

Guided Learning: 10

Module Availability

Semester 1

Prerequisites / Co-requisites

None

Module content

Indicative content includes the following.

Part A - Microprocessor Organisation and Design

- Introduction. Computer abstractions and technology. The changing face of computing. Embedded microprocessors. Instructions: language of the computer.
- MIPS Instruction Set Architecture. Arithmetic instructions. Load and store instructions. Instruction formats. Decision making instructions.
- Assessing and understanding microprocessors performance. Definition of performance. Performance metrics. Benchmarks. Examples.
- Building a 32-bit arithmetic logic unit for the MIPS architecture.
- Simplified implementation of the MIPS datapath. Single-cycle datapath implementation. Multi-cycle approach. Control signals. Control unit implementation. Examples.
- Enhancing CPU performance with pipelining. Pipelined datapath. Pipeline control. Dependencies. Forwarding. Stalling. Branch hazards.
- Exploring memory hierarchy. Direct mapped cache. Memory organization. Decreasing miss ratio with associativity. Example implementation: 4-way associative cache. Virtual memory. Making address translation fast. Modern microprocessors systems. Examples.

Part B - Algorithmic Programming and Software Design

- Analysis of algorithms: characterisation of problem data-size and program running time; big-O notation; asymptotic behaviour; typical running times.
- Building code: testing and debugging.
- Recursion: problem decomposition; base cases; implementation examples; run-time tracing; program efficiency issues: iteration versus recursion; recurrence relations.
- Data structures: revision of simple data-types and pointers; abstract data-types: implementation and use of singly linked lists, doubly linked lists, stacks, queues, trees, binary trees binary search trees.

- Searching: directed and controlled scanning and implications of data ordering; hashing: hash functions and collision handling.
- Sorting: elementary sort algorithms: selection, insertion, bubble; implications of initial ordering of data; quick sort; merge sort; priority queues, binary heaps and heap sort.

Assessment pattern

Assessment type	Unit of assessment	Weighting
Examination	2HR INVIGILATED WRITTEN EXAM	100

Alternative Assessment

n/a

Assessment Strategy

The **assessment strategy** for this module is designed to provide students with the opportunity to demonstrate the learning outcomes. The written examination will assess knowledge of design principles and features of processors, as well as the ability to write basic pseudo-assembly code for a common processor and to analyse processor and memory performance. The written examination will also assess knowledge of, and ability to describe using pseudo-code or C code, and analyse various data structures and algorithms. This is complemented by a number of formative assessment and feedback activities including computer laboratory sessions, as well as exercises and tutorials performed either in class or in a self-guided fashion.

The **summative assessment** for this module consists of:

- 2 hour invigilated written examination.

Any deadline given here is indicative. For confirmation of exact dates and times, please check the Departmental assessment calendar issued to you.

Formative assessment and feedback

For the module, students will receive formative assessment/feedback in the following ways.

- As part of exercises provided within the lecture materials, with model solutions provided on SurreyLearn,
- By means of tutorial problem sheets. Solutions to these are presented during class-discussion sessions in lectures/tutorials, with model solutions provided on SurreyLearn,
- During supervised computer laboratory sessions,
- As part of revision activities such as working through past exam papers or through formative online tests.

Module aims

- Microprocessors have penetrated everyday life from our smart-phones to our cars. Understanding the operations of a common microprocessor and how it is designed provides the students with skills to analyse and comments on physical processor architectures. This module aims to present the MIPS processor design and its evolution, from basic reduced instruction set languages, such as assembly, through to complex memory systems and co-processors.
- Programming is a skill which many electronic engineers find themselves needing at some point during their career, whether this be designing and building a new software system or, at the other end of the spectrum, implementing a new digital filter in software before translating it to hardware. This module will provide students with the necessary skills to be able to design and program algorithms, and to be able to analyse the performance of their algorithm.
- Students will secure programming skills learned during the previous year by implementing programs in C (in module EEE1035) with more complicated requirements than previously, and learn advanced programming techniques essential to anyone wishing to pursue software development further in their career e.g. in module EEE2047.
- The module also aims to provide opportunities for students to learn about the Surrey Pillars listed below.

Learning outcomes

		Attributes Developed	
Ref			
001	Write basic pseudo-assembly code for a common processor and explain the operation of some common algorithms for data handling and describe them using pseudo-code.	KCT	C3
002	Explain the key design principles of microprocessors analysing processor and memory performance	CPT	C4
003	Describe the data and control paths within a processor	KC	C6
004	Analyse the processing performance of algorithms and characterise this using Big-Oh notation	CPT	C4
005	Implement solutions to some simple logical problems using recursion	CPT	C5
006	Appropriately select, and justify the selection of, data structures and algorithms for some logical problems	KCT	C3
007	Describe and program in pseudo-code some well-known abstract data types	KC	C2
008	Implement some well-known algorithms and abstract data types using C code	KCT	C3

Attributes Developed

C - Cognitive/analytical

K - Subject knowledge

T - Transferable skills

P - Professional/Practical skills

Methods of Teaching / Learning

The learning and teaching strategy is designed to achieve the following aims:

1. Through the lectures, exercises/tutorials and computing laboratory session on “Big-O”, the students will be able to analyse algorithms and compare the performance of different algorithms.

2. Through the lectures, exercises/tutorials and computing laboratory session on recursion, students will learn several algorithm design paradigms enabling them to design and implement solutions to appropriate problems.

3. Through the lectures, exercises/tutorials and computing laboratory sessions on data structures, searching and sorting, students will learn to implement various algorithms for the solution of data-handling problems.

4. The lecture materials include a range of exercises and tutorials covering the full breadth of the module; students will be able to pace their own learning in parallel with the lecture course, and at various points solutions will be worked through in class-discussions, as well as being available on SurreyLearn.

Learning and teaching methods include the following.

- Pre-recorded lectures;
- Summary lectures;
- Tutorials;
- Self-guided exercises or tests with worked solutions;
- Computing laboratory sessions;
- Revision sessions.

Indicated Lecture Hours (which may also include seminars, tutorials, workshops and other contact time) are approximate and may include in-class tests where one or more of these are an assessment on the module. In-class tests are scheduled/organised separately to taught content and will be published on to student personal timetables, where they apply to taken modules, as soon as they are finalised by central administration. This will usually be after the initial publication of the teaching timetable for the relevant semester.

Reading list

https://readinglists.surrey.ac.uk
Upon accessing the reading list, please search for the module using the module code: **EEE2048**

Other information

Surrey's Curriculum Framework is committed to developing graduates with strengths in Employability, Digital Capabilities, Global and Cultural Capabilities, Sustainability and Resourcefulness and Resilience. This module is designed to allow students to develop knowledge, skills and capabilities in the following areas:

- **Digital capabilities:** This pillar is at the core of this module which explores digital systems both from a hardware perspective, through Part A concerned with Microprocessor Organisation and Design, and from a software perspective, through Part B concerned with Algorithmic Programming and Software Design. Students will develop the capacity to design and analyse complex computer and software architectures using pseudo-code, pseudo-assembly code and C code.

- Employability: The ability to design and analyse computer architectures and algorithms is highly sought after considering how ubiquitous digital systems are, thus opening up career opportunities not just in software-related industries but other sectors such as electronics. Through the study of both hardware and software perspectives within the same module, students will develop a deep and well-rounded understanding of computer systems and programs, that is highly valuable.
- Sustainability: This module engages with the issue of sustainability in the context of computer systems by equipping students with the tools to understand and analyse the computational complexity of algorithms and devise more efficient solutions, which in turn can be used to reduce compute/energy requirements. Students will become proficient at applying important concepts underpinning the analysis of algorithms (e.g. Big-O notation) and design paradigms (e.g. divide-and-conquer) demonstrated by solving a range of problems in the classroom and the labs.

Programmes this module appears in

Programme	Semester	Classification	Qualifying conditions
Computer and Internet Engineering BEng (Hons)	1	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Computer and Internet Engineering MEng	1	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electrical and Electronic Engineering BEng (Hons)	1	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electrical and Electronic Engineering MEng	1	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electronic Engineering BEng (Hons)	1	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electronic Engineering MEng	1	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electronic Engineering with Computer Systems BEng (Hons)	1	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electronic Engineering with Computer Systems MEng	1	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electronic Engineering with Nanotechnology BEng (Hons)	1	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electronic Engineering with Nanotechnology MEng	1	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electronic Engineering with Space Systems BEng (Hons)	1	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electronic Engineering with Space Systems MEng	1	Compulsory	A weighted aggregate mark of 40% is required to pass the module