

COURSE TITLE: Microprocessor Systems 1		CODE: CS3D1
LEVEL: Junior Sophister	CREDITS: 5	PREREQUISITES: None
LECTURER: Mr Cédric Assambo	LECTURES/WEEK: 3 TOTAL: 33	TUTORIALS/WEEK: 1 TOTAL: 11
TERMS: Semester 1 DURATION (WEEKS): 12	PRACTICALS/WEEK: 2 TOTAL: 14	
<p>AIMS/OBJECTIVES</p> <p>Microprocessor Systems 1 is a one-semester course taken by third year Electronic, Electronic/Computer and Computer Engineering students. It covers the Instruction Set Architecture (ISA) of a typical microprocessor-based computer, the Motorola MC68000, and equips students with a knowledge of the architecture, the associated assembly language, input/output programming techniques, exceptions (including interrupts) and exception-handling techniques. The module concludes with a simple introduction to the bus and instruction timing estimation.</p> <p>This course is intended to enable students to design and develop programmes and programme ‘architectures’, to test and debug programs and to analyse and modify their execution behaviour, based on a thorough familiarity with the low-level architecture of a computer. Concepts such as RISC/CISC architectures, register sets, addressing modes, data structures, subroutines, [informal] high-level to low-level language translation techniques, polling, interrupt priorities, asynchronous producer-consumer systems are introduced.</p>		
<p>SYLLABUS</p> <ul style="list-style-type: none"> • review of binary and hexadecimal arithmetic; • the Von Neumann machine; • the programmer’s model of the MC68000; • data representation: integers, characters, signed representations, arrays; • addressing modes: immediate, direct, indirect; • programme flow control: unconditional branch and jump; • the condition code register; • conditions and conditional branching; • high-level language constructs: while, if, for, etc.; • some complex 68000 instructions; • subroutines: mechanisms and parameter passing; • principles of input/output: polling; • exceptions and exception handling; • supervisor and user mode; • interrupts and interrupt handlers; • producer consumer organisation; queues and buffers; • introduction to the system bus; • instruction execution; • instruction timing. 		
<p>ASSOCIATED LABORATORY/PROJECT PROGRAMME</p> <p>Seven laboratory sessions: the practicals, conducted by each student individually, encourage the design, writing and testing of programmes and the development of the skills needed in actual practice.</p>		
<p>RECOMMENDED TEXT(S)</p> <ol style="list-style-type: none"> 1. A. Clements; 68000 Family Assembly Language; PWS Publishing Company; 1994; ISBN 0-534-93275-4 (Recommended, or similar, not mandatory). 2. T. King & B. Knight; Programming the M68000; Addison-Wesley, 2nd Ed.,1987. 3. J. Bacon; The Motorola MC68000: introduction to processor, memory and interfacing; Prentice Hall; 986. 		

LEARNING OUTCOMES

Upon completion of this course, students will be able to:

- analyse, specify, design, write and test assembly language programs of moderate complexity;
- select an appropriate 'architecture' or programme design to apply to a particular situation; e.g. an interrupt-driven I/O handler for a responsive real-time machine and design and build the necessary programmes;
- calculate the worst-case execution time of programmes or parts of programmes, and to design and build, or to modify, software to maximise its run time memory or execution-time behaviour;
- characterise and predict the effects of the properties of the bus on the overall performance of a system;
- the student will be able to describe some of the characteristics of RISC and CISC architectures.

TEACHING STRATEGIES

The teaching strategy is a mixture of lectures, problem-solving tutorials and hands-on practical laboratory sessions. The format of lectures is conventional, however, a great deal of informal interaction is normal, and students can expect to participate in question-and-answer and problem solving sessions. For the first four weeks of the course, students are taught the general principles of low-level architecture and programming. Tutorials held during this time review basic skills such as binary and hexadecimal notation and algorithm design. Students are challenged to build programs based on a partial knowledge of the computer's instruction set. Practical sessions, starting in the fourth week, require the students to design, write, evaluate and debug their programs on special-purpose development systems. More advanced topics introduced during lectures become the subject of practical sessions through the rest of the semester.

ASSESSMENT MODE(S)

Assessment of this course is by formal written two-hour examination and by assessment of the practical laboratory sessions. Practical sessions attract a mark of up to 20% of the end-of-year mark, and the examination makes up the remaining 80% or more.