Module Code	CSU11021
Module Name	Introduction to Computing I
ECTS Weighting[1]	5 ECTS
Semester taught	Semester 1
Module Coordinator/s	TBC
Module Learning Outcomes	On successful completion of this module, students will be able to: LO1. Describe the basic characteristics, structure and operation of a computer system; LO2. Represent and interpret basic information (integers, text) in binary form; LO3. Translate between simple high-level programming language constructs and their assembly language equivalents; LO4. Design, construct, document and test small-scale assembly language programs to solve simple problems; LO5. Reason about the cost of executing instructions and the efficiency of simple programs; LO6. Make use of appropriate documentation and reference material.
Module Content	This module provides students with an introduction to the basic structure and operation of a computer system, focussing on the processor (CPU), memory and the execution of software. Students gain an insight into the execution of programs on a computer system by designing, implementing and executing simple assembly language programs. Students are also introduced to concepts that are fundamental to the study of Computer Science, including the binary numeral system and the representation of basic information such as signed integers and strings (text). Students are encouraged to consider the relationship between high-level programming language constructs – from simple assignments and arithmetic expressions to conditional (*if, else*) and iterative (*while, for, do*) execution – and the realisation of these constructs as sequences of machine instructions. Students are also given opportunities to develop their problem solving, programming and written communication skills by designing solutions to programming problems, implementing those solutions, first in the form of high-level pseudo-code programs and then as assembly language programs, which they must document and test.
Teaching and Learning Methods	Lectures are used to introduce key concepts and provide worked examples. Every fortnight, each student participates in a tutorial to further explore each topic. In the tutorials, students work in groups of up to six on a set of exercises, using whiteboards to explore solutions, with guidance and feedback from teaching staff. Students work in pairs on four sets of lab exercises and are given two weeks to work on each set, beginning the exercise in the scheduled labs and completing them

[1] TEP Glossary

outside scheduled hours. At the end of each two-week cycle, each pair of students demonstrates their work to teaching staff and receives feedback during the lab.

Finally, a substantial mid-term assignment provides students with an opportunity to work individually on a larger-scale problem.

Assessment Details[2]

Assessment Component	Brief Description	Learning Outcomes Addressed	% of total	Week set	Week due
Examination	2 hour examination	LO1, LO2, LO3, LO4, LO5	70%	n/a	n/a
Lab 1	Basic Assembly Language; ASCII	LO2, LO4	2.5%	3	4
Lab 2	Condition Code Flags; simple assembly language programs	LO1, LO2, LO3, LO4	2.5%	5	6
Assignment	Design, implement, test and document simple assembly language programs	LO1, LO2, LO3, LO4, LO6	20%	5	8
Lab 3	Using memory	LO1, LO2, LO3, LO4	2.5%	9	10
Lab 4	Bit-wise operations	LO1, LO2, LO4, LO5	2.5%	11	12

Reassessment Details

Examination (2 hours, 100%)

Contact Hours and Indicative Student Workload

Contact Hours (scheduled hours per student over full module), broken down by:	**36 hours**
lecture	22 hours
laboratory	10 hours
tutorial or seminar	4 hours
other	0 hours
Independent study (outside scheduled contact hours), broken down by:	**78 hours**
preparation for classes and review of material (including preparation for examination, if applicable)	50 hours
completion of assessments (including examination, if applicable)	28 hours
Total Hours	**114 hours**

Recommended Reading List

William Hohl, “ARM Assembly Language: Fundamentals and Techniques”, CRC Press, 2009.

Steve Furber, “ARM System-on-Chip Architecture”, 2nd edition, Addison-Wesley Professional, 2000. [suggested further reading]

Andrew Sloss, Dominic Symes and Chris Wright, “ARM System Developer's Guide: Designing and Optimizing System Software”, Morgan Kaufmann, 2004. [suggested further reading]

Module Pre-requisites

Prerequisite modules: None

Other/alternative non-module prerequisites: Some familiarity with at least one high level programming language. (May be achieved by concurrently taking an introductory programming module.)

[2] TEP Guidelines on Workload and Assessment

Module Co-requisites	
Module Website	Blackboard / mymodule.tcd.ie
Last Update	28/06/2019 by Jonathan Dukes