School of Computer Science and Statistics
ECTS Module Descriptor

<table>
<tr><td>Academic Year</td><td>2018-19</td></tr>
<tr><td>Module Code</td><td>CS1021</td></tr>
<tr><td>Module Title</td><td>Introduction to Computing I</td></tr>
<tr><td>Pre-requisites</td><td>None</td></tr>
<tr><td>ECTS</td><td>5</td></tr>
<tr><td>Chief Examiner</td><td>Dr Jeremy Jones</td></tr>
<tr><td>Teaching Staff</td><td>Dr Jeremy Jones</td></tr>
<tr><td>Delivery</td><td>
<table>
<tr><th>Lecture hours</th><th>Lab hours (per student)</th><th>Tutorial hours (per student)</th><th>Total</th></tr>
<tr><td>22</td><td>10</td><td>10</td><td>42</td></tr>
<tr><td colspan="4">Comments: Attendance at all lectures, labs and tutorials is compulsory.</td></tr>
</table>
</td></tr>
<tr><td>Aims</td><td>This module provides students with an introduction to the basic structure and operation of a microprocessor system focussing on the central processing unit (CPU), memory and the execution of assembly language programmes.

Students are taught concepts that are fundamental to the study of Computer Science such as binary arithmetic, the architecture of a computer system and how a computer executes programmes

Students are shown the relationship between high-level programming language constructs – simple assignments, arithmetic expressions, conditional and iterative statements – and the realisation of these constructs as sequences of assembly language instructions.

Students develop their problem solving and programming skills by designing and implementing solutions to programming problems, first in the form of high-level pseudo-code or flowcharts and then as documented and tested assembly language programs.</td></tr>
<tr><td>Learning Outcomes</td><td>When students have successfully completed this module will be able to:

• describe the basic characteristics, structure and operation of a computer system
• represent and interpret basic information in binary form (integers, text, …)
• translate simple high-level programming language constructs into their assembly language equivalents
• design, construct, document and test small-scale assembly language programs to solve simple problems
• determine the cost of executing instructions and the efficiency of simple algorithms
• make use of appropriate documentation and reference material</td></tr>
</table>

Syllabus	Specific topics addressed in this module include: • number systems, memory and data representation • basic computer architecture (CPU, memory, registers, fetch-decode-execute loop) • assembly language and machine code • binary arithmetic and bit-wise operations • program flow control using branch instructions • memory accesses (using load and store instructions)
Assessment	Coursework 40% Mid-term test 20% End of term Practical Exam 40% 100% Coursework consists of 4 short "pair-programming" assignments and 2 larger individual assignments (although this may be subject to change). The mid-term test is a one hour written test held in the week after study week. The end of term practical exam will be a 1.5 hr practical exam held during the end of semester examination period in the Computer Science Labs (if it is not possible to schedule the practical exam, the fall back will be a 2 hr written exam). Supplemental assessment is by examination ONLY (100%). Students repeating 'off-books' (OBA) are also assessed by examination ONLY (100%) in all examination sessions.
	There is no required text for this module, the notes and labs should be self-explanatory. Suggested complementary texts are: • William Hohl, "ARM Assembly Language: Fundamentals and Techniques", CRC Press, 2009. • Steve Furber, "ARM System-on-Chip Architecture", 2nd edition, Addison-Wesley Professional, 2000. • Andrew Sloss, Dominic Symes and Chris Wright, "ARM System Developer's Guide: Designing and Optimizing System Software", Morgan Kaufmann, 2004.
Website	https://www.scss.tcd.ie/Jeremy.Jones/CS1021/CS1021.htm