Computer Architecture and Concurrency (COMP0008)

**Description**

**Aims:**
To provide a working knowledge of the hardware and architecture of a modern computer system, particularly focusing on concurrency aspects and those that have an impact on writing multithreaded software. Students will gain a fundamental understanding of the concurrency abstraction and how it impacts both computer architecture and software design. We will look at computer architecture aspects that directly impact multithreaded software such as the memory hierarchy, cache coherence/consistency and hardware multithreading. We will learn how to design correct multithreaded Java software based on a solid theoretical understanding of concurrency principles and the Java Memory Model. The module will cover an understanding of concurrency from low-level aspects (such as spin locks implemented in assembly language) to high-level design patterns used within the Java concurrency package.

**Learning outcomes:**
On successful completion of the module, a student will be able to:
- Use the concurrency abstraction to reason about concurrent systems;
- Write and understand modern assembly language programs;
- Describe how high-level languages are translated to assembly language and subsequently machine code;
- Describe the internal structure of processors and different forms of parallelism employed;
- Describe how modern computers handle memory and input/output including key concurrency aspects such as cache coherence, memory consistency/visibility and interrupt-driven thread switching;
- Describe how operating systems schedule application level threads onto the CPU resources;
- Describe how synchronization mechanisms in high-level languages are implemented in terms of more primitive hardware concurrency instructions;

**Key information**

<table>
<thead>
<tr>
<th><strong>Year</strong></th>
<th>2019/20</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Credit value</strong></td>
<td>15 (150 study hours)</td>
</tr>
<tr>
<td><strong>Delivery</strong></td>
<td>UG L5, Campus-based</td>
</tr>
<tr>
<td><strong>Reading List</strong></td>
<td><a href="https://www.ucl.ac.uk">View on UCL website</a></td>
</tr>
<tr>
<td><strong>Tutor</strong></td>
<td>Dr Kevin Bryson</td>
</tr>
<tr>
<td><strong>Term</strong></td>
<td>Term 1</td>
</tr>
<tr>
<td><strong>Timetable</strong></td>
<td><a href="https://www.ucl.ac.uk">View on UCL website</a></td>
</tr>
</tbody>
</table>

**Assessment**

For more information about the department, programmes, relevant open days and to browse other modules, visit [ucl.ac.uk](https://www.ucl.ac.uk)
-Reason about and resolve safety aspects of multithreaded Java including interference and visibility issues;  
-Write safe and efficient multithreaded Java code using the monitor design and other patterns;  
-Compare and contrast the traditional Java concurrency mechanisms with those of the new Java concurrency package mechanisms;  
-Correctly and safely use the thread-safe data and control structures within the new Java concurrency package.

**Content:**
- Top-down high level overview of a computer: the main components making up a computer and how they interact.  
- The concurrency abstraction.  
- Assembly language and machine code.  
- Translation of high-level languages into machine code.  
- Internal structure of a processor: the control unit (CU) and datapath.  
- Parallelism within the CPU.  
- Key aspects of how operating systems handle application-level threads.  
- Memory hierarchy, cache structure and cache coherence mechanisms.  
- Creating and managing Java threads.  
- Understanding and reasoning with the Java Memory Model (JMM) specification.  
- Java traditional synchronization mechanisms (and how they relate to low-level hardware instructions)  
- Monitor design patterns and conditional variables.  
- Reasoning about the correctness of concurrent programs: safety, variable visibility, liveness and deadlocks.  
- Java concurrency package: thread control classes and thread-safe data structures.

**Requisites:**
In order to be eligible to select this module, a student must be registered on a programme for which it is a formally-approved option or elective choice AND must:

- Understand basic computer arithmetic (binary/hex manipulation, fixed-size arithmetic, 2s- and 16s-complement forms, etc.);  
- Have strong Java programming skills.