Module title
Languages and Their Compilers

Module code	**Level**	**Hours per week**	**ECTS credits**	**Duration**
Tba	Bachelor (B.Sc.)	4	5	2 weeks block course + virtual lectures

Module instructor	**Lecture type**	**Prerequisite(s)**	**Grading**
Brian Tompsett, University of Hull	Lectures + Guided Lab Sessions	Intermediate Programming Ability	Coursework compiler writing exercise

Objectives
To enhance understanding of the purpose, features and structure of programming languages, and the similarities and differences between them. The students' knowledge of programming and further paradigms beyond earlier introductory material is covered. To introduce and develop the principles and techniques in the compiling of languages and illustrate their relevance in other contexts.

On successful completion of this module, students will be able to demonstrate knowledge and understanding of:

- History, development and varieties of computer programming languages and their features.
- Applying the concepts of language design and implementation techniques to the development of systems.
- The writing of compiler for a simple imperative computer language.
- Appreciate the relationship between computer architecture, language design and language implementation.

Learning Outcomes

1. Intellectual Skills: Critically evaluate and compare contemporary programming languages and their features integrating reference to literature effectively with own ideas within work.
2. Intellectual Skills: Explain, with comprehension, a range of issues pertinent to compiler construction theory and techniques.
3. Intellectual Skills: Critically evaluate the relationship between computer architecture and programming language design and implementation, justifying any links between these areas.
4. Practical Subject Specific Skills: Select, use, build and critically evaluate a compiler for a simple language.
5. Transferable Skills: Select, justify and use appropriate approaches, including some at the forefront of the subject / profession, to design, build, test and document programs.

Content

- Programming Languages: Purpose, Structure, Features, Types, history and development of languages, and comparison of different languages.
- Programming Paradigms: Language features, Overview of language types: imperative, functional, and object oriented.
- Language Structure: Describing, Formal Languages, Chomsky Hierarchy, Syntax, BNF, and Relevance to Compiling.
- Overview of Compiling: Purpose, Structure of Compilers and Interpreters, Lexical Analysis, Syntactic Analysis, Expression Evaluation, Semantic Analysis, and Code Generation, Optimisation.
- Symbol Table: Purpose and Use.
- Code Generation: Overview, Memory Management, Storage Allocation, Parameter Passing, Stacks and Heaps, Blocks, Scope, Register Allocation, Objects and Methods, Polymorphism, Intermediate Languages, and Examples.
- Practicalities: Building a Compiler, Linking, JIT, Cross-Compilation, Bootstrapping, Virtual machines and hardware architecture, and Intermediate representations.
- Run time support: Garbage Collection, Exception Handling, Debugging, Code Security, and Language Interoperability.

Textbook/teaching material

- Programming language design concepts - Watt, David Anthony, Findlay, William c2004
- High-level languages and their compilers - Watson, Des 1989
- Modern compiler design - Grune, Dick 2012
- Comparative programming languages - Wilson, Leslie B., Clark, Robert George 2001
- Programming language pragmatics - Scott, Michael Lee 2009

Note: this is not the official course descriptor according to the “Studien- und Prüfungsordnung” (SPO)