

Übersetzerbau <i>Compiler Construction</i>							Modulnummer:		
Bachelor Pflicht/Wahl <input checked="" type="checkbox"/> Wahlpflicht <input type="checkbox"/> Wahl <input type="checkbox"/> Sonderfall <input type="checkbox"/>				Modulbereich: Pflicht					
Anzahl der SWS	V	UE	K	S	Prak.	Proj.	Σ	Kreditpunkte: 6	Turnus i. d. R. angeboten alle 2 Semester
	3	1	0	0	0	0	4		
Formale Voraussetzungen: -									
Inhaltliche Voraussetzungen: Theoretische Informatik 1, Theoretische Informatik 2									
Vorgesehenes Semester: ab 1. Semester									
Sprache: Deutsch									
Ziele: <ul style="list-style-type: none"> • Prinzipien der Strukturierung von Übersetzern und Interpretern verstehen und anwenden können • Konzepte und Methoden der lexikalischen, syntaktischen und kontextuellen (statisch semantischen) Analyse verstehen, anwenden, auf die Implementierung konkreter Sprachen übertragen, beurteilen und bewerten können • Prinzipien der Übersetzung von imperativen und objektorientierten Programmiersprachen in Maschinencode verstehen, auf die Implementierung konkreter Konzepte übertragen und die Qualität des Codes beurteilen können. • Prinzipien der Codeerzeugung (Registerzuteilung, Instruktionsauswahl, globale und lokale Optimierung) verstehen können • selbstständig und in kleinen Teams Wissen und Verständnis erwerben und darstellen können. 									
Inhalte: <ul style="list-style-type: none"> • Implementierung von Programmiersprachen mit Interpretern, und Übersetzern. • Strukturierung von Übersetzern: Plattform(un)abhängigkeit, Bootstrap, Phasen. • Lexikalische Analyse: reguläre Definitionen, endliche Automaten, Symboltabellen, Benutzung von flex. • Syntaxanalyse: kontextfreie Grammatiken, ab- und aufsteigendes Parsieren, Baumaufbau, Fehlerbehandlung, Benutzung von bison. • Kontext-Analyse: Attributgrammatiken, Auswerter, Vereinbarungstabellen. • Transformation von imperativen und objektorientierten Programmen in abstrakten Maschinencode. • Grundzüge der Codeerzeugung für konkrete Maschinen: globale Optimierung, Registerzuteilung, Instruktionsauswahl, lokale Optimierung. <p>In der Übung Anwendung der in der Vorlesung erworbenen Kenntnisse und Fähigkeiten auf spezifische Konstrukte von Programmiersprachen.</p> <p>Insbesondere werden folgende theoretisch/methodische Grundlagen behandelt:</p> <ul style="list-style-type: none"> • Theorie der regulären und kontextfreien Sprachen • Algorithmen zur Konstruktion von deterministischen endlichen Automaten für reguläre Definitionen • Theorie des LL(k) und LR(k)-Parsierens, mit automatischer Fehlerbehandlung • Methoden der Grammatikdefinition, -transformation und -disambiguierung. • Theorie der Zweistufigen Grammatiken und Attributgrammatiken • Algorithmen zum Erzeugens von Auswertern für Attributgrammatiken • Methoden der Spezifikation von abstrakten Datentypen, für Bezeichnertabellen und Vereinbarungstabellen • Methodik der rekursiven Syntax-orientierten Definition für die Transformation von Syntaxbäumen in abstrakten Maschinencode 									

Unterlagen (Skripte, Literatur, Programme usw.):

- A.V. Aho, M. S. Lam, R. Sethi, J.D. Ullman. Compilers - Prinzipien, Techniken und Werkzeuge, zweite Auflage, Bonn: Pearson Education Deutschland (2008).
- R. Wilhelm, D. Maurer. Übersetzerbau: Theorie - Konstruktion - Generierung. Berlin: Springer, 2. Auflage (1997).

Weiteres Lehrmaterial ist auf der Webseite der Veranstaltung zu finden:

- Folienkopien
- Übungsaufgaben.

Übersetzer-Werkzeuge lex/flex, yacc/bison stehen im Rechnernetz des Studiengangs zur Verfügung.

Form der Prüfung:
i.d.R. mündliche Prüfung

Arbeitsaufwand	Präsenz	56 h
	Übungsbetrieb/Prüfungsvorbereitung	124 h
	Summe	180 h

Lehrende:
Dr. B. Hoffmann

Verantwortlich:
Dr. B. Hoffmann