اصول طراحی کامپایلر، ۸۱۰۱۴۲۰

نام انگلیسی درس	**Compiler Design and Implementation, 8101420**

نوع درس

	مهندسی برق							مهندسی کامپیوتر			واحد: ۳
	مخابرات	الکترونیک	قدرت	پزشکی	کنترل	دیجیتال	نرم‌افزار	سخت‌افزار	فناوری اطلاعات		
اجباری	☐	☐	☐	☐	☐	☐	☒	☒	☒		
اختیاری	☐	☐	☐	☐	☐	☐	☐	☐	☐		

مقطع	کارشناسی ■ تحصیلات تکمیلی ☐
همنیازها	ندارد
پیش نیازها	ساختمان‌های داده (۸۱۰۱۴۳۷)
مطالب پیش نیاز	
کتاب‌های مرجع	[1] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman , *Compilers: Principles, Techniques, and Tools*, Second Edition Boston: Addison-Wesley, 2007.
اهداف درس	در این درس، دانشجویان با کارکرد کامپایلر در تبدیل یک برنامه نوشته شده به زبان مبدأ به برنامه دیگری به زبان مقصد آشنا می‌شوند. همچنین می‌آموزند که چگونه ساختارهای کلی زبان کامپایل می‌شوند و رابطه میان ویژگی‌های یک زبان مبدأ و طراحی و پیاده‌سازی کامپایلر برای آن را یاد می‌گیرند. این دانش به آن‌ها در ساخت یک کامپایلر برای یک سیستم‌عامل خاص یا برای زبان‌های خاص منظوره **(DSL)** کمک خواهد کرد. علاوه بر موارد گفته شده، این درس بر مبنای تئوری زبان‌ها است که دانشجویان از پیش با آن آشنا هستند و در جریان ساخت کامپایلر، با کاربردهای آن آشنایی بیشتری پیدا خواهند کرد.
نتایج درس	دانشجویانی که این درس را با موفقیت پشت سر بگذارند قادر خواهند بود ۱- اجزای یک کامپایلر را با روش‌های متفاوت پیاده‌سازی کنند، ۲- یک زبان خاص منظوره و یک کامپایلر برای آن طراحی کنند، ۳- یک کامپایلر را توسط ابزارهای خودکار بسازند.
فهرست مباحث	۱- آشنایی با انواع مترجم‌ها برای زبان‌ها و تفاوت آن‌ها ۲- آشنایی کلی با اجزای کامپایلرها و وظایف آن‌ها ۳- اسکنر و پیاده‌سازی آن: تعریف زبان، نمودار گذار و عبارت‌های منظم، ابزار **ANTLR**، رفع خطا و روش‌های بهبود آن در اسکنر ۴- تجزیه‌کننده و پیاده‌سازی آن: تعریف گرامر، مباحث دسته‌بندی، اشتقاق و تجزیه گرامر، تجزیه‌کننده‌های نزولی بازگشتی غیرپیشگو، تجزیه‌کننده‌های پیشگو: **LL(K)**، **SRL(K)**، **CRL(K)**، **LALR(K)**، رفع خطا در تجزیه‌کننده‌های **LL** و **LR**: روش وحشت، روش‌های محلی و سراسری، قدرت تجزیه‌کننده‌ها ۵- روش‌های تحلیل کد که در زمان تجزیه از آن‌ها استفاده می‌شود: **syntax directed analysis (SDD)**، **syntax**

	directed schema (SDS)، ویژگی‌های ترکیبی و ارثی زبان‌ها، کاربرد **SDD** در تبدیل/ تحلیل کد، پیاده‌سازی **SDD** در تجزیه‌کننده نزولی بازگشتی (ابزار **ANTLR**)، **LL** و **LR**، پشته **(stack)** معنایی ۶- تحلیل‌کننده معنایی: مفاهیم و گستره، **type checking** و روش‌ها، **type system** و پیاده‌سازی، تولید **type expression** برای آرایه‌ها، رکوردها، توابع و اشیا، **subtyping** و وراثت، مدیریت **scope** و جدول نمادها ۷- تولید کد میانی: عبارات، تخصیص مقدار، دستورالعمل **control flow** ۸- مدیریت محیط زمان اجرا **(run-time)**: **stack** و **heap**، تولید کد برای تعریف و فراخوانی رویه‌ها، جاگذاری اشیا در حافظه، مراجعه پویا به مکان صحیح در حافظه در زبان‌های شی‌گرا ۹- بهینه‌سازی کد میانی و کد ماشین تولید شده: تحلیل **control flow**، بهینه‌سازی حلقه و بلوک‌ها
نرم‌افزارها و ابزارهای مورد نیاز	ANTLR, C++/Java, SPIM
تکالیف پیشنهادی	۸ تمرین
پروژه‌های پیشنهادی	دانشجویان یک کامپایلر برای زبانی با حداقل ویژگی‌ها که برای اهداف آموزشی طراحی شده است پیاده‌سازی خواهند کرد. این کامپایلر در نهایت یک کد **MIPS** از برنامه‌ای که به زبان مبدأ نوشته شده است تولید می‌کند که در شبیه‌ساز **MIPS** قابل اجرا خواهد بود. این پروژه در طول ترم و در قالب ۴ فاز تحویل گرفته می‌شود.
نمره‌دهی پیشنهادی	تکالیف ٪۱۰ پروژه‌ها ٪۲۰ کوئیزها ٪۱۰ امتحان میان ترم ٪۲۰ امتحان پایان ترم ٪۴۰ * برای پاس کردن درس، دانشجویان باید حداقل ٪۵۰ از نمره امتحان میان‌ترم و پایان‌ترم را کسب کنند.
سایر مراجع	[1] D. Grune, H. Bal, C. Jacobs, K. Langendoen, *Modern Compiler Design*, John Wiley & Sons, Ltd., 2000 [2] Yunlin Su, and Song Y. Yan, *Principles of Compilers: A New Approach to Compiler*, Springer, 2011.
تنظیم کننده	فاطمه قاسمی اصفهانی
تاریخ تنظیم	۱۳ شهریور ۱۳۹۶