Course Syllabus

**Course Description:**
The structure of compilers and interpreters: lexical syntax and semantic analysis, formal description of programming languages, parsing techniques, intermediate languages, optimization and code generation.

**Prerequisite:**
CS 3350.

**Textbook:**

**Exams and Grades:**
There will be two tests, some quizzes, assignments and a final exam.

- Test 1 15%
- Test 2 15%
- Final Exam 35%
- Assignments 30%
- Quizzes 5%

**Required knowledge:**

1. have been introduced to:
   
   (a) Languages
      
      i. Basic familiarity with BNF & syntax diagrams
      ii. Understanding of imperative, object & functional models
      iii. Grammars
   
   (b) Type Systems & Type Theory
   
   (c) Compilers
      
      i. Comparison of pure interpreters, compilers & translators
ii. Steps of a compiler (scanner, parser, optimizer and code generation)

iii. Basic data structures used (symbol table, syntax/exp. trees)

(d) Regular expressions (as in the use of Unix grep)

2. are able to apply:

(a) Basic computability
   i. context-free grammars

(b) Automata theory
   i. FSM
   ii. deterministic finite automatas
   iii. regular expressions
   iv. properties of context-free grammars

3. are able to apply the following in new situations:

(a) Abstract data types
   i. Specific ADT structures including tables, dynamic linked lists, stacks, trees, etc.
   ii. Strings, arrays and sequences.
   iii. Sets

(b) Procedural abstractions including with/without state, functions vs. procedures, side-effects, exceptions, recursive programming.

(c) Language Knowledge
   i. Detailed knowledge of at least one imperative language (or hybrid language significantly incorporating imperative structures)

(d) Type Systems
   i. data types
   ii. type checking
   iii. type attributes including scope, duration, storage class, linkage, etc.
   iv. name spaces
(e) Assembly level machine organization
   i. assembly/machine language programming

(f) discrete mathematics
   i. iterations, proofs (e.g. inductive proofs)

Learning outcomes:
On successful completion of the course, students will: (These outcomes depend on paradigm being taught)

1. have been introduced to:
   (a) Language translation systems
       i. code generation by tree walking
       ii. optimization techniques
   (b) Type systems
       i. type-checking algorithms
   (c) Basic advanced compiler optimizations including data dependency analysis, loop fission and loop fusion, cache optimization, predication and speculation

2. be able to apply:
   (a) Parsing
       i. left-right, top-down, bottom-up algorithms
       ii. generation of parse tables including SLR(1) & LL(1)
   (b) Stack frame allocation
   (c) Code generation
   (d) lex and yacc (or flex and bison) utilities vis-a-vis theory

3. be able to apply the following in new situations:
   (a) Lexical analysis (scanning)
   (b) Symbol table definition & manipulation including one and two pass updates.
Standards of Conduct:
Students are expected to conduct themselves in a professional and courteous manner, as prescribed by the Standards of Conduct. Students may discuss work assignments and programming exercises in a general way with other students, but the solutions must be done independently. Similarly, groups may discuss group project assignments with other groups, but the solutions must be done by the group itself. Graded work should be unmistakably your own. You may not transcribe or copy a solution taken from another person, book, or other source, e.g., a web page. Professors are required to – and will – report academic dishonesty and any other violation of the Standards of Conduct to the Dean of Students.

Faculty Information:
Professor: Luc Longpré
Office: 227 CS Building
Phone: 747-6804
Email: longpre@utep.edu
Office Hours: right after class, or make an appointment, faculty.utep.edu/longpre

Material:
Tentative schedule to be distributed later.