Dynamic Compilation

Compilers are at the heart of the program execution stack, orchestrating the translation of high-level programs to machine level instructions that the hardware executes. In this course, we focus on an important category of compilation called *dynamic compilation* that occurs as a program is running. With modern software heavily utilizing shared libraries, dynamic class loading and runtime binding, the scope of static analysis and transformation has grown restrictive. Dynamic compilation overcomes this challenge by postponing code generation and optimization until the initial stages of execution are complete, permitting accurate prediction of program behavior. In doing so, dynamic compilers enable effective feedback-directed optimization, architecture-specific code transformations, program introspection, etc. However, since a dynamic compiler uses cycles and other resources such as memory at run time, the overhead of runtime activity must be wisely managed. This course covers the basics of dynamic compilation and explores a variety of features and trade-offs in runtime optimization. All the lecture-based content is tightly coupled with "hacking" IonMonkey, the JavaScript dynamic compiler used inside Firefox!

Week	Date	Area	Topic
1	Jan 14th	Welcome	Myths, Terminology and Historical Context
	Jan 16th	Interpretation	What's Interpretation?
2	Jan 21st		*MLK Day – Holiday*
	Jan 23rd		Direct Threaded Interpretation
3	Jan 28th	Just-In-Time Compilation	What's a JIT Compiler?
	Jan 30th		The Basics of Compilation
4	Feb 4th		Binary Translation
	Feb 6th		Direct Linking and Indirect Chaining
5	Feb 11th		The Code Cache
	Feb 13th		Code Cache Management
6	Feb 18th		Heap Management
	Feb 20th		Garbage Collection
7	Feb 25th		Precise Exceptions
	Feb 27th		Self-Modifying Code
8	Mar 4th		Exam Review
	Mar 6th		Midterm #1
9	Mar 11th	No Class	*Spring Break – Holiday*
	Mar 13th		
10	Mar 18th	Adaptive Optimization	Why Do Adaptive Optimization?
	Mar 20th		Selective Optimization
11	Mar 25th		Profiling
	Mar 27th		Hardware and Software Profiling Techniques
12	Apr 1st		Code Layout
	Apr 3rd		Method Inlining
13	Apr 8th		Multiversioning and Specialization
	Apr 10th		Speculative Optimization
14	Apr 15th	HW/SW Co-design	Support for Code Caching
	Apr 17th		Support for Garbage Collection
15	Apr 22nd		Compiling for Heterogeneous Systems
	Apr 24th		Compiling for Power, Performance and Reliability
16	Apr 29th	Project	Presentations
	May 1st	Exam	Midterm #2

Course Code: EE382V

Unique ID: 16860

Course Times: MW 11am – 12:30pm, ENS 116

Grading Policy:

Class Problems / Reading Assignments:	10%
Programming Assignments:	20%
Midterm 1:	25%
Midterm 2:	25%
Project:	20%

Further details:

Instructor: Vijay Janapa Reddi (ENS 530), *vj@ece.utexas.edu*
Teaching Assistant: Aditya Srikanth, *aditya.srik@utexas.edu*

Pre-reqs: Computer Architecture (and preferably Compilers)