

Syllabus CIS 5930

There are two grading schemes available for this course, as shown below. As this is a hands-on course, my preference is for the default scheme, where the majority of the grade comes from actual programming assignments, whose purpose is to extend your understanding of the topics under discussion. If oral/written quizzes show inadequate student preparation, or if homework is of low quality or exceedingly uniform, I will be forced to rely more heavily on testing to ensure that students are understanding the material, as shown in the test-based grading scheme. If this occurs, I will announce the change to the class, and the written testing will begin in the following week.

Default scheme

- 85% projects, including final project
 - 0-2 presentations per student
 - Rest determined by programming assignments
- 10% oral/written quiz
- 5% class participation
- No written final

Test-based scheme

- 60% prog projects, including final project
- 35% written exams, including final exam
- 5% class participation

If we cover adequate material during lecture, it is probable that I will give each student a presentation to prepare, on the use of performance-centric software or a related topic. These presentations will be counted as projects, and will be graded on presentation and technical accuracy.

More complex projects given later in the course will receive greater weight in grading than the simple assignments possible at the start. Therefore, each assignment will have a multiplication factor associated with it, which indicates how many times that assignment counts. The first assignment has a factor of 1. This course is mainly about optimization, and so correctness of assignments is necessary but far from sufficient for good grades. The main determinant of grade will be application of concepts and the resulting execution efficiency. Thus, some portion of the grade will come from how close each assignment is to the most efficient available implementation, and some from the ranking within the class.

The quizzes are merely to ensure that students understand the concepts, and should be easily answered by anyone who attends the lecture and does the homework. The majority of such quizzes will be informal, with students answering questions during lecture, and describing and contrasting programming techniques. If student answers in class are unsatisfactory, more written quizzes will be administered, and if these are unsatisfactory, the grading scale employing full tests will be used instead.

Assignments must be handed in by 10AM on the due date given on the assignments. Any hand-in after this point is late, and is docked 10%. Late assignments must be handed in by the next class after the due date; after this, the assignment is graded as a zero.

Partial list of topics

The topics covered in this class include, but are not limited to:

1. Real world tester/timer creation
2. Architecture overview
 - Memory: disk, TLB, caches, registers, etc.
 - FPU: pipelining, ILP, etc.
3. Optimization overview
 - Memory: blocking, data copy, prefetch, fetch scheduling, efficient array indexing, etc.
 - FPU: software pipelining, superscalar scheduling, scalar expansion, etc.
4. Basics of assembly programming
 - AT&T x86 (maybe x86-64)
 - Perhaps SPARC/PowerPC
5. SSE, prefetch
6. As time allows, we may explore the following:
 - (a) Floating point representation and IEEE fp arithmetic
 - (b) Performance-centric feature overview of common HPC platforms
 - (c) shared memory parallelism using pthreads
 - (d) distributed memory parallelism using MPI

As this is the first time this course is being taught, the lecture may reveal a need to go into greater background material, or that certain topics are already well-understood. More topics centered on performance tuning will be added as time allows.