

CME 212: Lecture 1

January 9, 2012

- course introduction
- policies
- outline

CME 212 / ENERGY 212

Advanced programming methodologies for solving fundamental engineering problems using algorithms with pervasive application across disciplines. Overview of computer systems from a programming perspective including processor architectures, memory hierarchies, machine arithmetic, performance tuning techniques. Algorithms include iterative, direct linear solvers, fft, and divide and conquer strategies for n-body problems. Software development; other practical UNIX tools including shell scripting, vi/emacs, gcc, make, gdb, gprof, version control systems and LaTeX. Prerequisites: CME 200/ME 300A, CME 211, and CS 106X or equivalent level of programming in C/C++.

General topics

Topics:

- processor architectures
- memory hierarchies
- machine arithmetic
- performance tuning

Algorithms:

- basic statistical calculations
- iterative linear solver
- direct linear solvers
- fft
- divide and conquer for N-body

Tools:

- shell scripting
- vi/emacs
- gcc, make, gdb, gprof
- version control, git
- valgrind
- L^AT_EX

Prerequisites

- Knowledge of programming, some experience in C (cme211)
- Knowledge of linear algebra (cme200)
- Some scientific computing (cme108)
- A willingness to experiment, read, make mistakes, ask questions, and help others
- These are not strict, we do not check

Logistics

See course website:

<http://www.stanford.edu/class/cme212/>

Week 2: C programming

- primitive types, variables
- arrays, pointers
- loops, conditional statements
- compound data types, structures
- functions
- the c preprocessor
- code organization, header files, source files
- standard library functionality

Week 3: Tools

Development tools:

- gcc
- make
- gdb
- gprof
- valgrind
- git (version control)

Unix tools:

- bash
- man, info
- emacs
- command line techniques

Week 4: Representation

- bits, bytes
- numbers
- endians
- 2-complement
- floating point
- Composite types
- disassembly
- function calls
- memory: stack, heap

Week 5: Testing & performance

- testing code
- timing tools
- measuring performance
- profiling code

Week 6: Basic computer architecture

- basic overview of computing systems
 - “post-RISC” architecture
 - LOAD/OPERATE/STORE model
 - ILP (instruction level parallelism)
- overview of memory hierarchy
 - registers
 - cache levels
 - SRAM, DRAM

Week 7: Optimization

- optimizing code to take advantage of hardware
- loop optimization
- making code cache friendly
- unrolling loops
- compiler optimization levels
- optimization blockers

Week 8: Linear algebra

- BLAS - basic linear algebra subroutines
- LAPACK - linear algebra package
- Sparse linear algebra and data structures

Week 9: Other topics

- C-fortran interface
- Python

Week 10: Last week

- Matlab mex interface for C/Fortran
- Course review, what's next

Something useful: making simple web pages

I will show you the tools I use to make the cme212 website.

Ingredients:

- Markdown: `http://daringfireball.net/projects/markdown/`
- Pandoc: `http://johnmacfarlane.net/pandoc/`
- GNU make: `$ man make`
- rsync: `$ man rsync`

What to do

- Program hello world in C on corn.stanford.edu and your chosen environment
- In unfamiliar with \LaTeX , read chapters 1 and 2 of The Not So Short Introduction
- Install \LaTeX
 - For mac: <http://www.tug.org/mactex/2011/>
 - For windows: <http://miktex.org/>