12.010 Computational Methods of Scientific Programming 2023

Tom Herring, tah@mit.edu
Chris Hill, cnh@mit.edu

Lec 01: Getting started and setup

About 12.010 - I

- Introduces programming from perspective of physical science use. Programming examples are taken from simple physics and mathematics problems.
- Classes introduces programming languages a researcher in a lab might encounter – Python, Julia, R, Matlab/Octave, C, C++, Fortran.
- Two main parts to class
  - Homeworks – 3 (due Fridays, at least 3+ weeks apart).
  - Project and presentation - individual or group (due end of class).
- Goal is that by end of class you will feel more confident that
  - you could work in a research project involving programming
  - you would know where to start to try and understand an unfamiliar program
  - you know how to write, tested and run some interesting scientific computing programs by yourself

About 12.010 - II

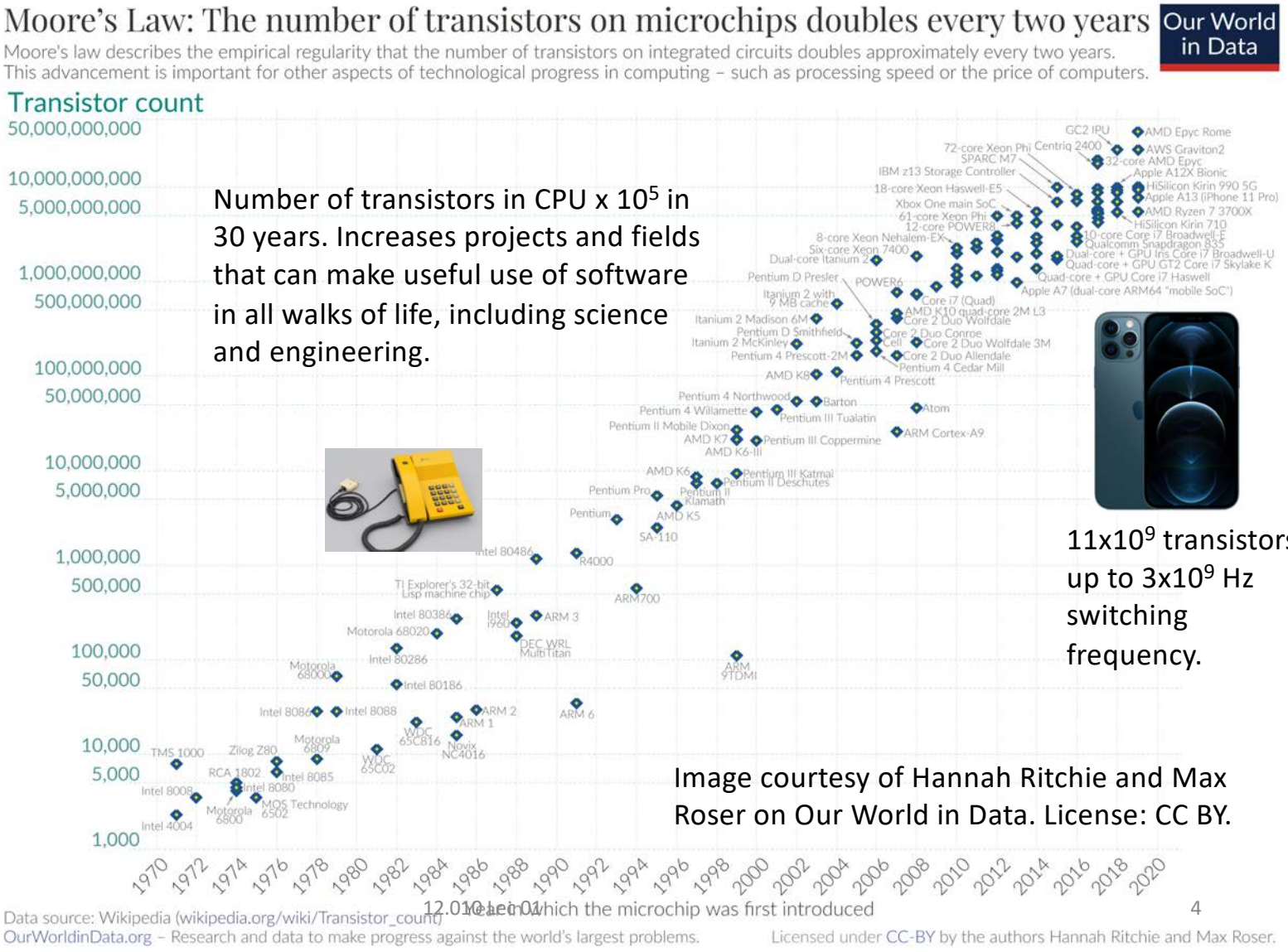
- Homeworks
  - Each has 2-3 parts, involve writing some code, based on some example covered in class and available online.
  - Graded on written answer and working code.
- Projects
  - Topic can be related to any science/math problem of interest.
  - Can be based in major, another domain research course etc…
  - Usually involve writing new code from scratch, including find and understanding algorithm, developing some core compute piece, developing some input and output piece.
  - Graded on presentation, working and documented code.
  - Group projects, with work divided across several people, or individual projects are fine.

About computers

A modern computer contains a CPU with multiple cores and billions of transistors.

Programming is about getting those billions of transistors to act in some way to solve a problem.

Programming languages like Python, Julia, R, Matlab, C, Fortran etc… provide high-level “abstractions” that mask the underlying complexity.



Number of transistors in CPU x 10^5 in 30 years. Increases projects and fields that can make useful use of software in all walks of life, including science and engineering.

$11x10^9$ transistors, up to $3x10^9$ Hz switching frequency.

Image courtesy of Hannah Ritchie and Max Roser on Our World in Data. License: CC BY.

Programming languages

- In scientific computing/computational science there are multiple programming languages
- Some of the most common ones include
  - Python, R, Julia, Matlab/Octave, C, C++, Fortran
- The languages have different strengths, but also have a lot of ideas and concepts in common.
- Practices around how to develop programs, how to test and check, how to formulate problems are similar too.
- In 12.010 we introduce programming in Python, but then
  - explore how basic programs look in all the languages listed
  - illustrate different features that are found in different languages

Getting started

- Programming can only be learned hands-on experience
  - Computers are not that smart
  - Programming involves puzzling
    - over how to express a problem in a way a computer can solve it,
    - following rules about how to instruct the computer,
    - methodically trying to understand whether program is behaving as expected,
    - building up more complete and complex programs step-by-step.

- The first “puzzle” is configuring a ”programming environment” to work in.
  - In this class we will begin by using Jupyter Notebooks as our common programming environment
  - Jupyter Notebooks are widely used in teaching programming and data science
  - They are not the only programming environment, and have some limitations, but they are very useful and common starting point.

First notebooks

- Start by trying cloud resource – see ”Launch Jupyterhub Class Exercise” in Assignments
- Then install locally – see “Install Jupyter Notebook Locally Class Exercise” in Assignments.
- There are example Notebooks in the Week 1 section we will try. Not all of these cases will be covered today.

MIT OpenCourseWare

https://ocw.mit.edu

12.010 Computational Methods of Scientific Programming, Fall 2024