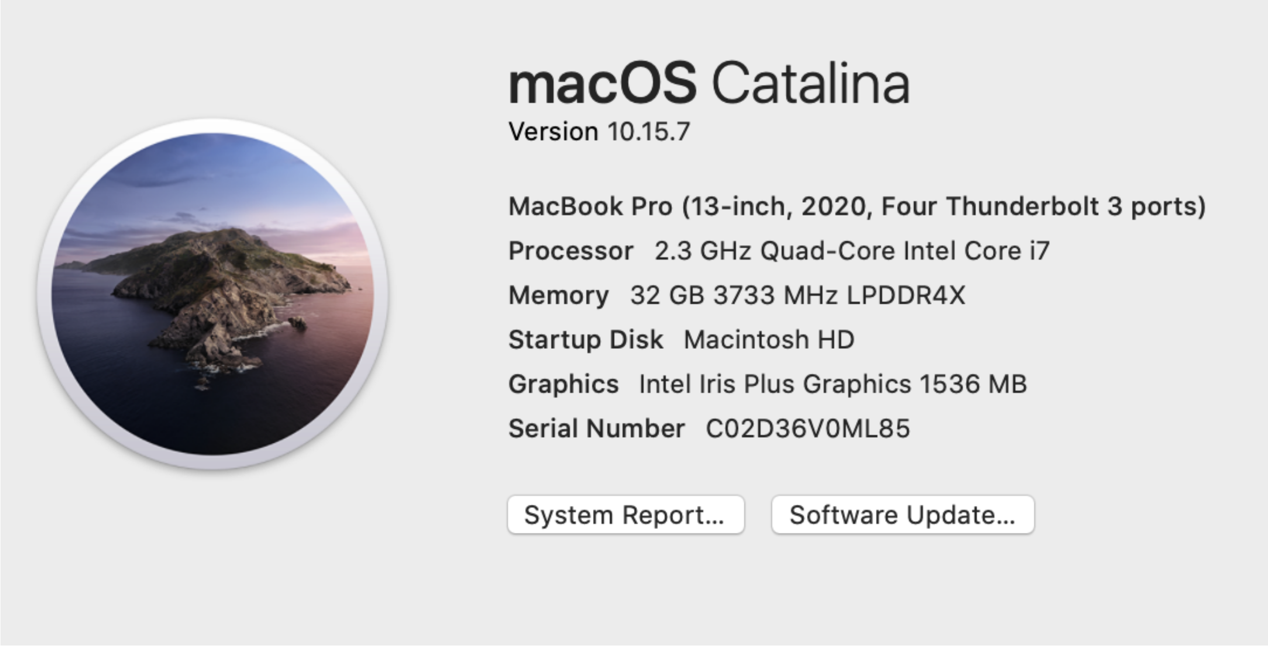
> By Jiahui Tang

Spec:



macOS Catalina

Version 10.15.7

MacBook Pro (13-inch, 2020, Four Thunderbolt 3 ports)

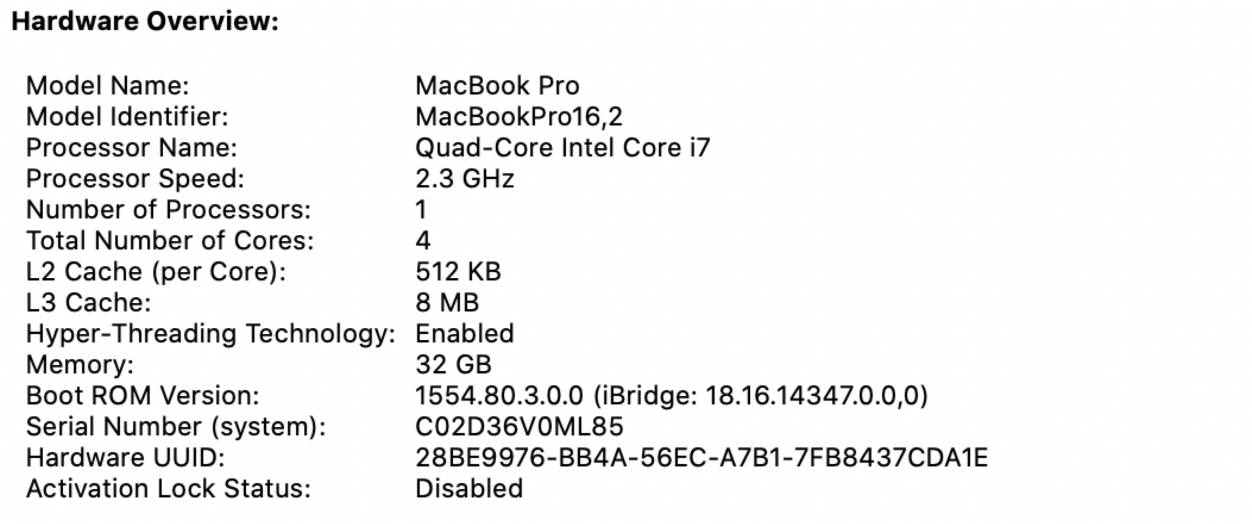
Processor 2.3 GHz Quad-Core Intel Core i7

Memory 32 GB 3733 MHz LPDDR4X

Startup Disk Macintosh HD

Graphics Intel Iris Plus Graphics 1536 MB

Serial Number C02D36V0ML85



Hardware Overview:

Model Name:	MacBook Pro
Model Identifier:	MacBookPro16,2
Processor Name:	Quad-Core Intel Core i7
Processor Speed:	2.3 GHz
Number of Processors:	1
Total Number of Cores:	4
L2 Cache (per Core):	512 KB
L3 Cache:	8 MB
Hyper-Threading Technology:	Enabled
Memory:	32 GB
Boot ROM Version:	1554.80.3.0.0 (iBridge: 18.16.14347.0.0,0)
Serial Number (system):	C02D36V0ML85
Hardware UUID:	28BE9976-BB4A-56EC-A7B1-7FB8437CDA1E
Activation Lock Status:	Disabled

- Spec of system:
    - Model: MacBook Pro 13 Inch
    - Number of CPUs: 1 Quad-Core 2.3GHZ Intel Core i7 CPU
    - Number of Core per CPU: 4 cores
    - Clock Rate: 2.3GHZ
    - Cache Memory: 512 KB L2 Cache (per Core); 8MB L3 Cache
    - Main Memory: 32 GB 3733 MHz LPDDR4X
- Cluster: N/A

- Operating System: Mac OS Catalina Version 10.15.7
- Compiler: Apple clang version 12.0.0 (gcc)

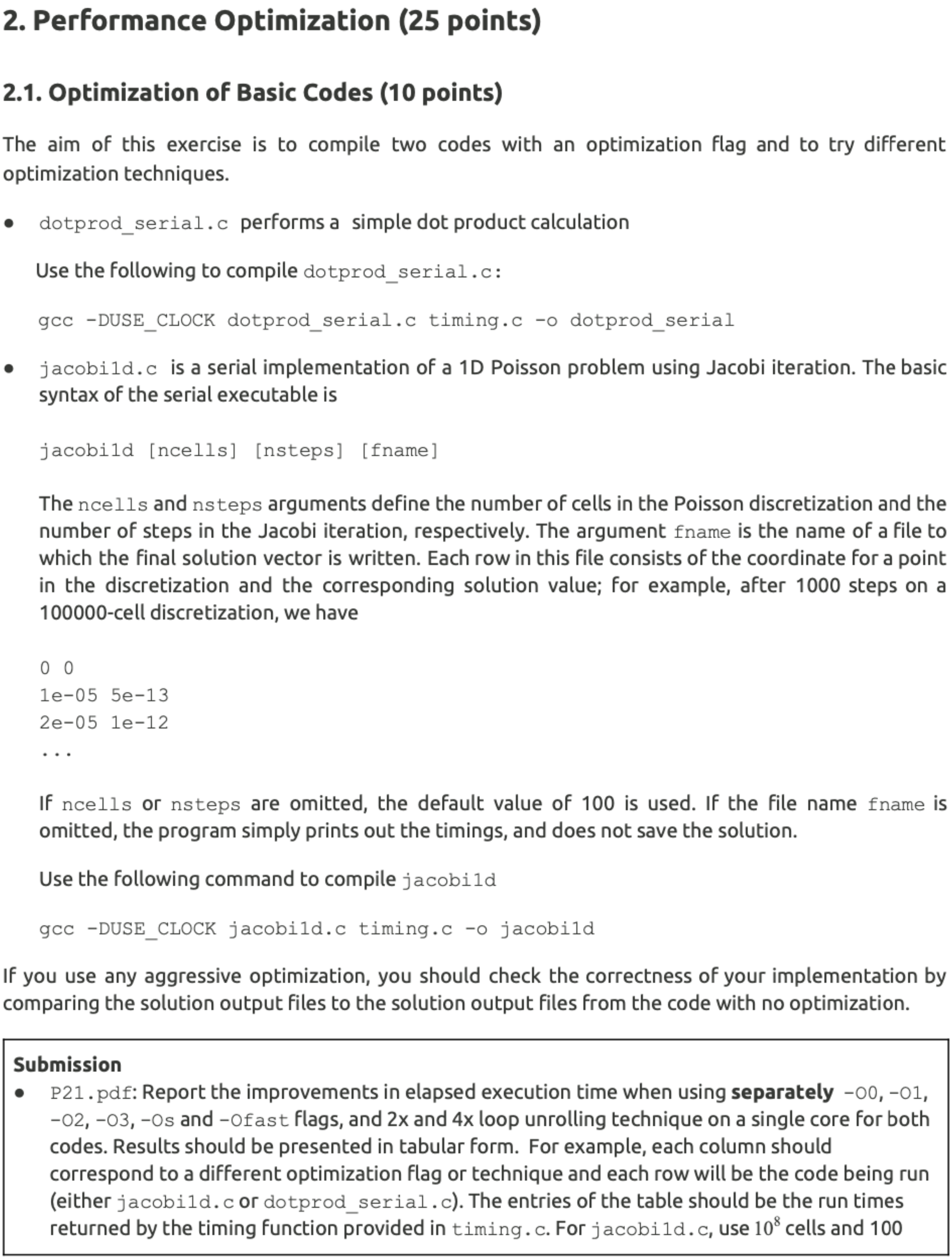
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
 Configured with: --prefix=/Library/Developer/CommandLineTools/usr --with-gxx-include-dir=/Library/Developer/CommandLineTools/SDKs/MacOSX10.15.sdk/usr/include/c++/4

Apple clang version 12.0.0 (clang-1200.0.32.29)
Target: x86_64-apple-darwin19.6.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
```

- Libraries: N/A
- Others: N/A

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



2. Performance Optimization (25 points)

2.1. Optimization of Basic Codes (10 points)

The aim of this exercise is to compile two codes with an optimization flag and to try different optimization techniques.

- `dotprod_serial.c` performs a simple dot product calculation

  Use the following to compile `dotprod_serial.c`:

  ```
  gcc -DUSE_CLOCK dotprod_serial.c timing.c -o dotprod_serial
  ```

- `jacobi1d.c` is a serial implementation of a 1D Poisson problem using Jacobi iteration. The basic syntax of the serial executable is

  ```
  jacobi1d [ncells] [nsteps] [fname]
  ```

  The `ncells` and `nsteps` arguments define the number of cells in the Poisson discretization and the number of steps in the Jacobi iteration, respectively. The argument `fname` is the name of a file to which the final solution vector is written. Each row in this file consists of the coordinate for a point in the discretization and the corresponding solution value; for example, after 1000 steps on a 100000-cell discretization, we have

  ```
  0 0
  1e-05 5e-13
  2e-05 1e-12
  ...
  ```

  If `ncells` or `nsteps` are omitted, the default value of 100 is used. If the file name `fname` is omitted, the program simply prints out the timings, and does not save the solution.

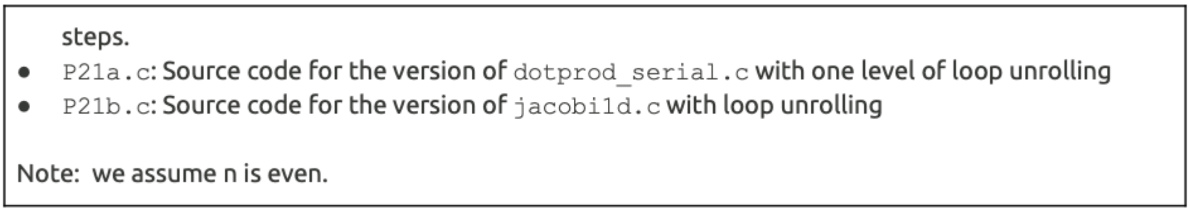
  Use the following command to compile `jacobi1d`

  ```
  gcc -DUSE_CLOCK jacobi1d.c timing.c -o jacobi1d
  ```

If you use any aggressive optimization, you should check the correctness of your implementation by comparing the solution output files to the solution output files from the code with no optimization.

Submission

- `P21.pdf`: Report the improvements in elapsed execution time when using **separately** `-O0`, `-O1`, `-O2`, `-O3`, `-Os` and `-Ofast` flags, and 2x and 4x loop unrolling technique on a single core for both codes. Results should be presented in tabular form. For example, each column should correspond to a different optimization flag or technique and each row will be the code being run (either `jacobi1d.c` or `dotprod_serial.c`). The entries of the table should be the run times returned by the timing function provided in `timing.c`. For `jacobi1d.c`, use 10^8 cells and 100



steps.

- `P21a.c`: Source code for the version of `dotprod_serial.c` with one level of loop unrolling
- `P21b.c`: Source code for the version of `jacobi1d.c` with loop unrolling

Note: we assume n is even.

Table for improvements in elapsed execution time (*unit: seconds*)

	-O0	-O1	-O2	-O3	-Os	-Ofast	2x loop unrolling	4x loop unrolling	No optimizat referen
dotprod_serial.c	2.70789	1.41835	1.48742	1.41143	1.43814	0.835847	2.80793	2.51294	2.85945
jacobi1d.c	31.409	12.9953	12.3088	12.666	12.7162	12.1927	27.1008	26.4874	30.8584

Notes: compared all output files to solution files with no optimization to cross check correctness

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js