… because the OS-course is just too easy!

Home Exam 2: Video encoding on ARMv8 using ARM NEON instructions

March 17th, 2026

Håkon Kvale Stensland

Video Encoding



- Pål still wants to encode some videos…

- Pål got a brand-new PowerPoint 2024 license, but animations are more resource demanding than he thought, and they are offloaded to the GPU!

- However, Pål remember that his SoC has multiple CPU cores with SIMD units, so he needs you to help him!

Precode

- Same Codec63 precode as used on Home Exam 1. Compile with O1 optimizations and no auto vectorization.
- You are not allowed to change out the Motion Estimation, Motion Compensation or DCT algorithms.
- You are not allowed to paste code from other projects / encoders.
- You only need to optimize the Codec63 encoder!
- Your implementation can use multiple cores, but you must use NEON to utilize instruction level parallelism. If you use multiple cores, be aware of memory accesses.

Your task

- Utilize the possibilities of the NEON instructions and other tricks to get the most performance.
- Start by profiling the encoder to see which parts of the encoder constitute bottlenecks.
- Remember, after optimizing one part of the code, more profiling might be needed to find new bottlenecks.
- Write a scientific report (max 4 pages) with details on which parts of the encoding process that benefited from your optimizations. The report should contain benchmarks and profiling information.
- Remember to also report on optimization that did not work as you expected!

How are you evaluated?

- (**20%**) Implementation that **compiles, runs**, and can **produce a correct video output** (we also check the motion prediction).
- (**20%**) SIMD for **Motion Estimation**
- (**25%**) SIMD for **DCT / iDCT.** This includes parts of DCT such as transpose, convert, horizontal add, etc.
- (**20%**) **Report**. Profiling between different steps. What worked and didn't work? Comparing different approaches and quality of code.
- (**15%**) **Architecture use and Performance** of the solution.
- Bonus points can also be given for non-obvious optimizations and good analysis of unexpected results.

Formal Information

- **Deadlines:**
  - Design: **Tuesday, March 24th – 10:00**
  - Code: **Tuesday April 14th – 23:59**
  - Report: **Friday April 17th – 23:59**

- The assignment will be graded and count 33% of the final grade.

- Deliver your code and report to: https://devilry.ifi.uio.no/
- Deliver your design and poster (as .pdf) to in5050@ifi.uio.no

- Prepare a poster (PDF or PPT) and a short talk (5 minutes) for the class on (April 14th).
  - Best poster & presentation will be awarded!

Last but not least!

- The Codec63 precode available for download with git. Clone the repository and work on you own local version.

git clone https://github.com/griwodz/in5050-codec63.git

- Bugs in the code can be reported in Github's issue tracking system, on e-mail (in5050 at ifi.uio.no).

- Use Mattermost or meet other groups at IFI to discuss and share tips, tricks and troubles!
  - Everything is allowed, except sharing of code! ☺

Good Luck!