*… because the OS-course is just too easy!*

# Home Exam 1: Video encoding on ARMv8 using ARM NEON instructions

February 4th, 2022

**Håkon Kvale Stensland**

# Assignment groups

- When you have found a partner to work with, or if you need to be assigned a partner, e-mail us at:

  in5050 (at) ifi.uio.no

- IN9050 students (PhD students) must contact us for agreeing on an extra project.

# Video Encoding

- Pål wants to use his mobile phone for encoding videos from a local soccer match…
- … but three out of four cores on his mobile phone are busy running the e-mail client…
- It is therefore your task to make the encoder as efficient as possible on a **single core.**



# Codec63

- Codec63 is basically Motion JPEG with intra-frame dependencies.
- It adds Motion Estimation and Motion Compensation from the MJPEG example.
- Uses a more efficient DCT implementation (1D vs. 2D)
- Will be the basis for *all* three home exams in the course.
- The precode that is handed out today is a basic single threaded encoder and decoder written in C.

# Precode

- 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.
- Make sure that your video works with a unmodified decoder.
  - Because the encoder and decoder have shared structures, it is recommended to have a unmodified repository for testing.
- You only need to optimize the Codec63 **encoder**!
- Your implementation is supposed to be single threaded, and use NEON to utilize instruction level parallelism.

# Your task

- Utilize the possibilities of the NEON instructions and other tricks to get the most performance out of a single core.
- 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**
- (**30%**) 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.
- (**10%**) **Performance** of the solution.
- Bonus points can also be given for non-obvious optimizations and good analysis of unexpected results.

# Formal Information

- **Deadlines:**
  - Design: **Friday February 11th – 12:00**
  - Code: **Friday February 25th – 23:59**
  - Report: **Monday February 28th – 12:00**

- 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
  - *24 hours before deadline if you need poster printed, if not bring it to the group session*

- Prepare a poster (two A3 pages) and a short talk (2 minutes without slides) to pitch your poster for the class (February 25th).
  - 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://bitbucket.org/mpg_code/in5050-codec63.git**

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

- Use the Mattermost-channel to discuss and share tips, tricks and troubles!
  - Everything is allowed, except sharing of code! ☺

# Good Luck!