IL2212 EMBEDDED SOFTWARE

Laboratory 3

Implementation

VERSION 2.0 (20240119)

1 Objectives, Requirements, and Bonus Points

1.1 Objectives

The main focus of this laboratory is the implementation of an image procession application on different target platforms: sequential processor with RTOS, bare-metal sequential processor, and multiprocessor. The laboratory is conducted in groups of two students. Each group can prepare a single document for the solutions, where both partners' names are stated. Still, each student has to submit the solutions via the Canvas page of the course.

1.2 Requirements for Passing the Laboratory

To pass the laboratory, the student must solve and submit all *mandatory tasks* before the deadline. The Canvas page of the course states the deadline and submission instructions. During the laboratory session, the student has to demonstrate and convincingly explain all solutions individually to the course staff.

1.3 Bonus Points

Students who, in addition to the mandatory tasks, solve *optional tasks* before the deadline can receive bonus points. The bonus points will count towards the total points in the exam but do not count for the individual parts of the exam. During the laboratory session, the student has to demonstrate and explain all solutions to the course staff.

2 Laboratory Tasks

The repository `https://gits-15.sys.kth.se/ingo/il2212-lab` includes the hardware platform and software sources, which serve as starting point for this laboratory.

1. (**Mandatory Task**) Use `hello_image` as starting point and implement the image application using only the actors `graySDF` and `asciiSDF` using the real-time operating system `ucos-II`. The actors `graySDF` and `asciiSDF` shall run as individual tasks which communicate via a message queue. Use floating-point operations for the implementation of `graySDF`. Measure the execution time and memory footprint.

2. (**Mandatory Task**) Based on your implementation of task 1 change the implementation of `graySDF` using only integer operations instead of floating-point operations. Measure the execution time and memory footprint and compare the performance of the implementations of task 1 and task 2.

3. (**Mandatory Task**) Change the implementation of task 2 to a bare-metal implementation on a single processor (without RTOS support). Measure the execution time and memory footprint and compare the performance with the implementations of the previous tasks.

4. (**Mandatory Task**) Add the actor `resizeSDF` to your implementation. Measure the execution time and memory footprint.

5. (**Mandatory Task**) Implement the application of task 4 and use an individual processor for each of the three actors `graySDF`, `resizeSDF`, and `asciiSDF`. Measure the execution time and memory footprint. Compare the performance with task 4.

6. (**Optional Task, 2 Bonus Points**) Create an improved multiprocessor solution where you use all processors to optimize the throughput. Measure the execution time and memory footprint. Compare the performance with task 5.

7. (**Optional Task, 2 Bonus Point**) Integrate also the actors `correctionSDF`, `controlSDF`, and `brightnessSDF` into your application without keeping the throughput goal out of focus. Measure the execution time and memory footprint.

8. (**Optional Task, 2 Bonus Point**) Integrate also the remaining actor `sobelSDF` into your application without keeping the throughput goal out of focus. Measure the execution time and memory footprint.

9. (**Optional Task, 1 Bonus Point**) Based on the results of and the experience gained from this laboratory, summarize what you consider the most important aspects to achieve an efficient implementation of streaming media SDF-applications on a bare-metal multiprocessor.