KTH VETENSKAP OCH KONST

KTH Information and Communication Technology

IL2212 EMBEDDED SOFTWARE

Laboratory 2
Digital Image Processing on a Multiprocessor

February 5, 2016

1 Objectives

The task of this laboratory is to implement a concurrent data-flow application in different ways on the multiprocessor from Laboratory 1. Students shall apply their knowledge from the lectures in order to derive an efficient implementation exploiting the parallelism provided by the architecture.

During this laboratory students shall implement the application

- on a single core using the real-time operating system MicroC/OS-II
- on a single core without operating system
- on the multiprocessor without operating system

In all cases students shall try to optimize the following objectives: throughput and memory footprint.

In addition students shall document their results in a technical report.

2 Deadlines

Students have to respect the following deadlines:

1. Submit a first version of the technical report as PDF-file until 28/2 23:59. The report shall cover all the steps required for the first lab session (see below). Submission instructions are found on the course webpage.
2. For the first lab session of Lab 2,

- demonstrate the working single-core solutions executing on the DE2-board,
- demonstrate an initial, working version of the parallel implementation executing on the DE2-board. The initial version is expected to be functionally correct, but not yet optimized for performance.
- prepare a table with measured throughput and memory footprint of all versions of the application,
- prepare a data-flow diagram (in electronic format) that shows how parallelism is exploited for the multicore implementation and include it into the report. Other visual representations which facilitate the understanding of your application are also encouraged (e.g. schedule diagram, partition/mapping diagram).

3. Submit the *final version* of the technical report as PDF-file until 7/3 12:00 (noon). Submission instructions are found on the course webpage.

4. At the second lab session of Lab 2, the *performance-optimized* version of the parallel implementation shall be demonstrated, for verification of the results in terms of performance provided in the final report, but also to show that the application is functionally correct for images of different sizes.

3 General Guidelines

Read the entire laboratory manual in detail before you start with the preparation tasks. Complete the preparation tasks before your lab session in order to be allowed to start the laboratory exercises.

It is very important that students are well-prepared for the labs, since both lab rooms and assistants are expensive and limited resources, which shall be used efficiently. The laboratory will be conducted by groups of two students. However, each student has to understand the developed source code and the preparation tasks. Course assistants will check that students are well-prepared.

NOTE:

- All program code shall be well-structured and well-documented. The language used for documentation is English.

- Students shall point out problems in the lab manuals using the wiki of the course web.

4 Preparation Tasks

4.1 Installation

Please use the virtual machine and the multiprocessor core from Laboratory 1 and your Git repository for submitting code and the report. The public repository contains code snippets, functions and example input images that may help you get started. Note that you may change any of these as long as the application satisfies all specifications.

4.2 Image Processing Application

The students shall develop an image processing application and implement it on a provided multiprocessor platform. The application shall read a stream of RGB images and convert the RGB-images in several steps into smaller ASCII art-images. The exact details and links to further information are specified on the course web. The functionality of the application can be divided into several steps:

- For each new image
    1. Read new image from memory
    2. Convert image to grayscale and rescale it
    3. Show condensed image information on seven-segment displays [1]
    4. Perform given DSP algorithm(s) on the grayscale image
    5. Convert modified image to ASCII art
    6. Save picture in memory
    7. Display execution time on terminal [1]
    8. Print picture on terminal (stdout) [1]

For each configuration, the program needs to implement two modes which can run on request.

1. *debug mode*, having the following specifications:
    - loops over a sequence of images of different sizes found on the SRAM. This enforces the application to be generic, parametrizable and functionally correct.
    - prints execution times and the ASCII pictures on the terminal (stdout) after each image is processed. The seven-segment display should show a column of 8 ASCII pixels starting from a chosen coordinate `(X,Y)` (e.g. `#define X 2` and `#define Y 3`)
    - imposes a considerable delay between images (2-5 seconds) so that the print-outs can be observed.
2. *performance mode*, behaving in the following way:
    - starts the performance counter,

[1]required only for the debug mode

- loops over a sequence of at least three *different* 32×32 images found in the SRAM (so that the input data is refreshed for each frame). During one loop the program grabs an image processes it and writes back the result to the SRAM.
- stops the program and the performance counter after a number (e.g. 100) of iterations.
- prints out performance data: execution time, normalized execution time (per iteration), throughput (images/second), etc...

NOTE: The throughput measured during *performance mode* will be considered for passing the lab. The application is a streaming media application. Thus it is not sufficient to process only a single image, but an "infinite" stream of images.

4.3 Implementation

The application shall be implemented in different ways. For each implementation, try to optimize throughput and memory footprint. The following table shall be part of your report:

	Single Core (RTOS)	Single Core (Bare-Metal)	Multicore
Throughput [s^{-1}]			
SRAM (bytes)			
OnChip CPU 1 [bytes]	-	-	
OnChip CPU 2 [bytes]	-	-	
OnChip CPU 3 [bytes]	-	-	
OnChip CPU 4 [bytes]	-	-	
OnChip (Shared) [bytes]			
Total Memory [bytes]			

4.3.1 Single Processor with RTOS

Implement the data-flow application on a single processor using the SRAM with RTOS.

4.3.2 Bare-Metal Single Processor

Implement the data-flow application on a single processor using the SRAM without RTOS.

4.3.3 Bare-Metal on Multiprocessor

Implement the data-flow application on the multiprocessor, without RTOS. You can use the resources of the provided architecture as you see fit.

4.4 Design Constraints

The final multiprocessor implementation has to meet the following design constraints in order to pass the lab:

- The *throughput* has to be higher than 420 images / second for 32 $\times$ 32 pixel images.

  or

- The *throughput* has to higher than 350 images / second for 32 $\times$ 32 pixel images and the total code footprint (without the images) should not pass 45 KBytes.

4.5 Technical Report

Students shall deliver a technical report that not only summarizes the results (table from 4.3), but also discusses and evaluates them. It shall contain a description of the parallel implementation including a data-flow diagram that shows how parallelism is exploited and all measures taken for performance optimization. If you have noticed any limitations of your approach, include a discussion of them and any alternative approach you see fit.

The technical report shall use the IEEE double column format and should have three to four pages. A link to Word and LaTeX-templates for this format is available on the course web page.

5 Laboratory Tasks

During the laboratory students will

- demonstrate during the first lab session that their application is at least functionally correct for all configurations.
- demonstrate during the second lab session that that they can achieve the throughput and/or memory requirements.
- answer questions on their
  1. technical report
  2. solution including code optimization techniques.

6 Examination

In order to pass the laboratory the student must

- have completed the preparation tasks of Section 4 before the lab session

- have demonstrated the preparation tasks in a convincing way for the laboratory staff
- have delivered an implementation that meets both the throughput and memory requirements
- have delivered a complete, well-written and informative technical report