Instructions for the FPGA Board framework

We want to test your BubbleSort implementations on the FPGA Board. For visualizing the output of BubbleSort when running on the FPGA Board we print the resulting array plus some additional information to the URAT interface. As you don't have a compiler for the DLX-CPU yet we have translated this C-Code to assembly and we have removed the BubbleSort implementation out of the created assembly file. The assembly file exists in two versions for simulating with dlxsim and ModelSim (Bubblesort_LCD_forSW.s) and for running on the FPGA Board (Bubblesort_LCD_forHW.s). The main difference is the implementation of the t_print method (see chapter 8.5): the dlxsim/ModelSim implementation is just printing all characters to the console or to a file whereas the implementation for the FPGA sends additional control signals and waits for the corresponding answers from the UART.

You have to add your BubbleSort implementation to the **new frameworks** (Bubblesort_LCD_forSW.s and Bubblesort_LCD_forHW.s). Therefore you have to consider the following points:

- The function calling scheme (parameter passing, register saving, stack preparation …) from the compiler was somewhat different to the scheme that was used in the old frame-work from Session 3 (bs_Framework.s and bs_Framework_pipelined.s). We have adapted the function calling scheme for the BubbleSort method and added it to the new frame-work:
    - Search for "bubbleSort" in the framework file to find the method for BubbleSort.
    - The method already contains the function calling scheme and the "ADD YOUR IMPLEMENTATION HERE" comment.
- You first should add your BubbleSort implementation to the framework (Bubblesort_LCD_forSW.s) for dlxsim to test it. After it is running you can copy-and-paste your implementation to the framework for the hardware.
- Dlxsim will write all the t_print output (including the control signals; each character is written to a new line) to the console. But you can also forward the output to a file and look at it after the simulation has finished. Use the "-pf{fileName}" parameter (pf stands for *putchar file*) therefore.
- You can restart the CPU by pressing the "reset" button of the FPGA board, but REMEMBER, that your array in data memory is already sorted after the first run, so the second, third, … run will be significantly faster than the first one.

- The BubbleSort framework is measuring the number of cycles for the execution of the bubbleSort methods. This measurement is done by a counter on the FPGA Board or in dlxsim/ModelSim respectively. This measurement only measures the bubbleSort method, but not the overhead for e.g. printing the result.

Using the Makefile:

1. "make sim": it will compile your assembly file (in this case Bubblesort_LCD_forSW.s) and generates the required DM/IM file for the Modelsim.
2. "make dlxsim": it will start the dlx simulator to simulate the compiled file generated from the previous stage. Here, you have to pass some parameters to dlxsim such as the lcd file to print the outputs. For that, you have to type "make dlxsim DLXSIM_PARAM="-da0 -pfputc.out " where putc.out is the file contains the printed output. Note: the pipeline is by default has the value 4. You have to check putc.out file in order to see the result of your code and check the sorted array.
3. "make fpga": it will compile your assembly file (in this case Bubblesort_LCD_forHW.s) and generate the required DM/IM file and combine them with your bitstream which generated from the ISE tool (note that env_settings will tell the Makefile where the ISE project folder is located). Finally, a new bitstream file contains your hardware CPU along with corresponding IM/DM files of your application will be generated in the folder "BUILD_FPGA". This bitstream must be used to configure the FPGA.
4. "make all": it will compile for dlxsim/Modelsim and for FPGA
5. "make upload": it will upload the existing bitstream to the FPGA (note: this command does not generate a new bitstream)