1.	Fachgebiet Architektur eingebetteter Systeme **Advanced Computer Architectures**	WiSe2020

Assignment start: 13.11.2020 Submission deadline: 03.12.2020

Assignment 1 - WinMIPS64 — 25 Points

The first assignment of the lab for Advanced Computer Architecture (ACA) is the WinMIPS64 assignment. In this assignment you will play the role of a C to MIPS64 assembly compiler. The WinMIPS64 simulator, which is a simulator of a 64-bit 5-stage pipelined MIPS architecture, is used to verify correctness and evaluate performance.

The simulator, documentation, and the to be converted C-program `smooth.c` are available on the ISIS course page. It is highly recommended that you read the documentation and experiment with the example programs before starting the real C to Mips64 conversion process.

Use the labels of Listing 1 at the start of your assembly file as the variables. In our testing procedure we will use the same labels, but with different contents, for example as in Listing 2. The content of `coeff` and `sample` should, therefore, not be assumed to be static. The label `N_SAMPLES` should be changed according to the number of declared samples. The `sample` and `result` will always have the same number of samples. Also the number of coefficients (`N_COEFFS`) is fixed at 3 for all test sequences.

Listing 1: MIPS64 program header

```
0 .data
1 N_COEFFS:    .word   3
2 coeff:       .double 0.5,1.0,0.5
3 N_SAMPLES:   .word   5
4 sample:      .double 1.0,2.0,1.0,2.0,1.0
5 result:      .double 0.0,0.0,0.0,0.0,0.0
```

Listing 2: Possible testing sequence

```
0 N_COEFFS     .word   3
1 coeff:       .double 0.5,1.0,0.5
2 N_SAMPLES:   .word   10
3 sample:      .double 1,2,3,4,5,6,7,8,9,10
4 result:      .double 0,0,0,0,0,0,0,0,0,0
```

The MIPS64 code in Listing 3 can be used to print a value. This is useful to replace `printf` calls to confirm your output while programming. The `printf` calls in `smooth.c` do not have to be converted and calls to `printdouble` should be removed in the final assembly file. They are there for debugging purposes. Alternatively, you can also look at the register contents when stepping through the instructions.

The WinMIPS64 simulator has to be configured to have a floating point addition delay of **3** cycles, floating point multiplication of **4** cycles, and a division delay of **12** cycles. Also configure WinMIPS64 to use **“data forwarding"** and the **“branch delay slot"**. The grading for this assignment is 10 points

Listing 3: Print doubles

```
0 printdouble:
1     lwu r11,CR(r0)   ; Control Register
2     lwu r12,DR(r0)   ; Data Register
3     daddi   r10,r0,3     ; r10 = 3
4     s.d f0,(r12)     ; output f0 ...
5     sd  r10,(r11)    ; ... to screen
6     jr  r31          ; return
```

for functional correctness of your assembly program. The remainder of the points are earned based on your execution time. The faster you are the more points you earn. A requirement is that the code size reported by WinMIPS64 may not exceed 350 bytes. If your code has too many bugs, i.e., it cannot be fairly compared in terms of speed, you will not receive any points for execution time. The performance tests will be performed with our non-disclosed testing sequences.

A few tips:

- Optimize the C-program first before converting to MIPS64. This will decrease conversion complexity and improve performance significantly.
- Check the correctness after each step you perform.
- Commit or backup your work frequently, so you can easily revert to an earlier version when something goes wrong.
- Write comments for your own convenience. Content of registers is mostly useful information.
- We will test with sequences of various sizes, small (<10 elements) to large (between 40 and 100). Your performance result is the combined number of cycles of the test sequences.
- The instruction set syntax used in WinMIPS64 is slightly different from what is used in the lecture slides. Code copied from the slides will not work. Read the documentation and/or look at the example programs to familiarize yourself with the WinMIPS64 instruction syntax.

The deliverable of this assignment is your assembly file of the `smooth` program. Do not include any calls to `printdouble` or the function itself in the assembly file you submit. Also please refrain from changing the label names in Listing 1 as it will help in our checking procedure. New labels can be added but should not replace the old ones.