Universität des Saarlandes
FR 6.2 - Informatik
Prof. Dr. W.J. Paul
Dipl.-Ing. Christoph Baumann

Exercise Sheet 4
Computer Architecture II
(Due: Nov 19th, 2013)

Note: If you want to take the exam you have to register in HISPOS until Nov 17th!

Exercise 1: (Rounding Bug) **(3+4+1 Points)**

In the lecture it was mentioned that some Intel Floating-Point Units suffered from a rounding bug which occured when exact results were rounded twice with different precisions. We assume two significand precisions p and p' such that $p' > p$. Let r and r' be rounding functions of the same rounding mode for precisions (n, p) and (n, p') respectively. Considering a real number $x \in \mathbb{R}$, lying between the two neighboring representative numbers $(x_d, x_u) \ni x$ where x_d,$x_u \in \mathcal{R}$ with respect to precision p, we now want to find values for x such that rounding first with precision p' and then with p gives a different result than just rounding x with precision p alone.

$$r(x) \neq r(r'(x))$$

a) For which rounding modes and in what situation can the rounding bug occur? Give a short explanation!

b) Deduce the exact range for x, such that the rounding bug occurs! Construct one example value from x_d or x_u!

c) What is the minimal difference $p' - p$ for the bug to occur?

Exercise 2: (Wrapped Exponents) **(4+4+4 Points)**

Let $\alpha = 3 \cdot 2^{n-2}$ and $a, b \in \mathcal{R}$.

a) Show for $x = a \cdot b$ that the following statements hold!

- $OVF(x) \Rightarrow 2^{e_{min}} < |x \cdot 2^{-\alpha}| < X_{max}$
- $UNF(x) \Rightarrow 2^{e_{min}} < |x \cdot 2^{\alpha}| < X_{max}$

b) Show the same for $x = a/b$ and $b \neq 0$!

c) Show the same for $x = a + b$ and $x = a - b$!

Exercise 3: (TINY & LOSS) **(5+5 Points)**

In the lecture we argued that the following two implications hold:

$$LOSS_a(x) \Rightarrow LOSS_b(x) \qquad TINY_a(x) \Rightarrow TINY_b(x)$$

To be proven or disproven:

$$LOSS_a(x) \Leftarrow LOSS_b(x) \qquad TINY_a(x) \Leftarrow TINY_b(x)$$

Exercise 4: (Alignment Shift Limitation) **(10 Points)**

When adding two IEEE-normal floating-point numbers (s_a, e_a, f_a) and (s_b, e_b, f_b), it is necessary to align the significands by multiplying the operand having the smaller exponent with $2^{-\delta}$, where we assume $\delta = e_a - e_b \geq 0$ wlog. This is called an *Alignment Shift*. We have shown in the lecture that it is enough to use the $p+1$-representative $f' = [2^{-\delta} \cdot f_b]_{p+1}$ instead of $2^{-\delta} \cdot f_b$ in the computation. In particular we had:

$$S = 2^{e_a} \cdot ((-1)^{s_a} \cdot f_a + (-1)^{s_b} \cdot 2^{-\delta} \cdot f_b) =_{p-\hat{e}} 2^{e_a} \cdot ((-1)^{s_a} \cdot f_a + (-1)^{s_b} \cdot f')$$

Now imagine we used only the usual p-representative $f'' = [2^{-\delta} \cdot f_b]_p$ in the computation of the sum S'':

$$S'' = 2^{e_a} \cdot ((-1)^{s_a} \cdot f_a + (-1)^{s_b} \cdot f'')$$

In order to show that this does not suffice, find a counter-example and give values for s_a, s_b, f_a, f_b and δ such that

$$S =_{p-\hat{e}} S''$$

does *not* hold!

Hint: Compute the normalized representations $\hat{\eta}(S) = (s, \hat{e}, \hat{f})$ and $\hat{\eta}(S'') = (s, \hat{e}, \hat{f}'')$ to obtain $\hat{e}$!