Saarland University
Department 6.2 - Computer Science
Prof. Dr. W. J. Paul
Dr. Mikhail Kovalev, Jonas Oberhauser

System Architecture (block course) - SS13

Exercise Sheet 9 (due: 9.09.13) - 21 points

Exercise 1: **(1)**

Explain why we defined $R(x)$, i.e., what is the meaning of $R(x)$ and where do we use it?

Exercise 2: **(3)**

Recall Horner's rule which says that a polynomial $P(x) = \sum_{i=0}^{n} a_i x^i$ can be evaluated by computing $h^{n+1}(x)$:

$$h^0(x) = 0$$
$$h^{i+1}(x) = a_{n-i} + x \cdot h^i(x)$$

(a) (1 point) Prove that $P(x) = h^{n+1}(x)$

(b) (2 points) Prove that you can use Horner's rule to easily compute the value of a decimal string. That is, write a C0 function `Ecode_dc` that takes a `DTEp p` as declared in the lecture. Assume that `p` was generated by a parser for C0 when parsing a valid C0 program and `p*.label` is `"DiS"`, deriving a decimal sequence. The function should return an `LELp` such that the labels of the elements of the list are instruction codes that load the binary value (modulo 2^{32}) of that decimal sequence.

Exercise 3: **(2)**

Recall code generation for expression evaluation. Give rules to generate MIPS code for expressions `e == e'` and `e <= e'`. Use a macro `Ecode(s,j)` to evaluate a subexpression `s` in register `j`.

Exercise 4: **(2)**

Recall code generation for statement execution. Give rules to generate MIPS code for statements `x = new t*` and `return e`. Use a macro `Ecode(s,j)` to evaluate an expression `s` in register `j`.

Exercise 5: **(2)**

Consider the Aho-Ullman Algorithm. Assume a situation where a binary expression has to be evaluated and the left operand has a smaller expression tree than the right operand, like in the following example: $(4 + 3) + ((5 * 3) + (4 * 2))$ Prove that the Aho-Ullman Algorithm, in general, uses more registers if you evaluate the operand with the smaller tree first.

Exercise 6: **(2)**

Explain why we can only use about 20 registers, not 32, for expression evaluation.

Exercise 7: **(1)**

Prove that $l(c.prn[i]) = rSt \Rightarrow c.prn[i+1] = succ(c.clr(c.rd))$ is invariant w.r.t. function call and while loop execution.

Exercise 8: **(2)**

Specify and implement a special purpose register file (SPR), which behaves at the same time as a single port (32,5)-RAM and as a set of 32 many 32-bit registers.

Exercise 9: **(6)**

Recall internal and external interrupt events.

(a) (2 points) List the different types of interrupts and their effects on the *pc* in a MIPS configuration

Saarland University
Department 6.2 - Computer Science
Prof. Dr. W. J. Paul
Dr. Mikhail Kovalev, Jonas Oberhauser

System Architecture (block course) - SS13

Exercise Sheet 9 (due: 9.09.13) - 21 points

(b) (1 point) What are the interrupt types of page fault on fetch, misalignment, and system call?

(c) (1 point) Why are we using these interrupt types?

(d) (1 point) Explain what it means to mask an interrupt. Which interrupts can be masked and where in our MIPS+Interrupts semantics does this show?

(e) (1 point) We want to make it illegal to use the `movg2s` instruction while the MIPS+Interrupts configuration is in user mode. Define the effect of `movg2s rd rt` on a MIPS+Interrupts configuration c such that an illegal interrupt is raised when the configuration is in user mode, and the content of $c.gpr(rt)$ is copied to $c.spr(rd)$ otherwise.