Saarland University
Department 6.2 – Computer Science
Prof. Dr. W. J. Paul
M. Sc. Petro Lutsyk

Computer Architecture – WS14/15

Exercise Sheet 4 (due: 25.11.14, 36 points)

Exercise 1: (carry-look-ahead adder (CLA)) (12 points)

In this exercise we build the adder which is asymptotically optimal, i.e., it has a linear cost and logarithmic delay. In order to achieve these properties we incorporate the parallel prefix circuit (from the previous assignment) as a subroutine.

Recall that formally an adder is a circuit which given the input bits $a, b \in \mathbb{B}^n$ and the carry-in $c_0 \in \mathbb{B}$ computes the sum bits $s \in \mathbb{B}^n$ and the carry-out $c_n \in \mathbb{B}^n$ such that

$$\langle c_n s \rangle = \langle a \rangle + \langle b \rangle + c_0.$$

For simplicity lets consider inputs of length $n = 2^k$ for $k > 0$.

(a) First, realize that the addition of two numbers is not harder than the computation of all carries, i.e. bits c_i for $i \in [1 : n]$. Show this simple fact.

(b) Consider the following two functions. For $a[n-1:0]$, $b[n-1:0]$ and indices $i \geq j$ we define

$$\begin{aligned} p_{i,j}(a,b) &\equiv \langle a[i:j] \rangle + \langle b[i:j] \rangle = \langle 1^{i-j+1} \rangle \\ g_{i,j}(a,b) &\equiv \begin{cases} \langle a[i:j] \rangle + \langle b[i:j] \rangle \geq \langle 10^{i-j+1} \rangle & j > 0 \\ \langle a[i:j] \rangle + \langle b[i:j] \rangle + c_0 \geq \langle 10^{i-j+1} \rangle & j = 0. \end{cases} \end{aligned}$$

For $i = j$ one observes that

$$\begin{aligned} p_{i,i}(a,b) &= a_i \oplus b_i \\ g_{i,i}(a,b) &= \begin{cases} a_i \wedge b_i & i > 0 \\ a_0 \wedge b_0 \vee a_0 \wedge c_0 \vee b_0 \vee c_0 & i = 0. \end{cases} \end{aligned}$$

For $i > j$ show how to compute the $p_{i,j}(a,b)$ and $g_{i,j}(a,b)$ knowing the

$$\begin{matrix} p_{i,k+1}(a,b) & p_{k,j}(a,b) \\ g_{i,k+1}(a,b) & g_{k,j}(a,b) \end{matrix}$$

for some $k \in (i : j]$.

(c) Construct a circuit which implements the algorithm above.

For $M = \mathbb{B}^2$ assume that

$$\circ : M \times M \to M$$

is the function computed by the circuit.

Show that $\circ$ is associative, i.e., $\forall (g_1, p_1), (g_2, p_2), (g_3, p_3)$ it holds:

$$(g_1, p_1) \circ ((g_2, p_2) \circ (g_3, p_3)) = ((g_1, p_1) \circ (g_2, p_2)) \circ (g_3, p_3).$$

(d) Using the results of (a), (b) and (c) give a construction of a circuit which implements a CLA. Prove that your implementation is correct. Hint: make use of a circuit for $PP_\circ$ as a subroutine.

In the exercises on the processor hardware correctness we assume the absence of the *reset* interrupt.

Exercise 2: (instruction fetch) (6 points)
In this exercise we establish correctness of the instruction fetch.
Recall the instruction memory environment from the lecture.
Assume $sim(c, h)$ and show that the hardware has fetched the correct instruction, i.e.

$$I(h) = I(c).$$

Explain (in words) why it is important to have a ROM portion in the memory.

Exercise 3: (pc correctness) (6 points)
In this exercise we establish correctness of the next pc computation.
Recall the next pc environment from the lecture.

(a) Assume $sim(c, h)$ and show the following statements:

$$\begin{aligned} pcinc(h) &= c.pc +_{32} 4_{32} \\ btarget(h) &= btarget(c) \\ jbtaken(h) &\equiv jbtaken(c). \end{aligned}$$

For the first one, stress the place where the *software condition* is required.

(b) Conclude that the pc component of the hardware is updated correctly, i.e.

$$sim(c, h) \rightarrow h'.pc = c'.pc.$$

Exercise 4: (gpr correctness) (6 points)
In this exercise we establish correctness for the GPR.
Recall the wiring of the GPR component from the lecture. Assume that the current instruction is not a memory access, i.e., $\neg ls(c)$.

(a) Assume $sim(c, h)$ and show the following statements:

$$\begin{aligned} gprin(h) &= gprin(c) \\ gprw(h) &\equiv gprw(c). \end{aligned}$$

(b) Conclude that the hardware GPR is simulated properly in the next configuration, i.e.

$$sim(c, h) \rightarrow h'.gpr = c'.gpr.$$

You can assume correctness of the hardware computational units (ALU, SU, etc.)

$$C(h) = C(c),$$

and correctness of the instruction decoder, that is for all predicates p and function fields F:

$$\begin{aligned} p(h) &\equiv p(c) \\ F(h) &= F(c). \end{aligned}$$

Exercise 5: (ls correctness) (6 points)
In this exercise we establish correctness of the memory operations.
Recall the data memory environment from the lecture. Assume that the current instruction is a memory access, i.e., $ls(c)$.

(a) Assume $sim(c, h)$ and show that the hardware has loaded the correct data, i.e.

$$l(c) \rightarrow lres(h) = lres(c)$$

(b) Show that the hardware memory is simulated properly in the next configuration, i.e.

$$sim(c, h) \rightarrow (\forall a \in \mathbb{B}^{29} : h'.m(a) = c'.m_8(a000))$$

Explain (in words) why we need shifters for load and store.