SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Sebastian Hahn, M.Sc.

Verification of Real-Time Systems SS 2015

Assignment 7

Deadline: June 18, 2015, before the lecture

Exercise 7.1: Cache Replacement Policies (12=6·2 Points)

A cache replacement policy A outperforms policy B if A exhibits more cache hits than B on the same sequence of accesses. Assume an initially empty 4-way fully-associative cache. Provide a sequence of accesses such that

(a) LRU outperforms FIFO

(b) LRU outperforms PLRU

(c) FIFO outperforms LRU

(d) FIFO outperforms PLRU

(e) PLRU outperforms LRU

(f) PLRU outperforms FIFO

Exercise 7.2: Cache Analysis (7=3+4 Points)

Consider the following program:

```
read a;
read b;
read a;
if (a>b) {
    read c;
    read d;
} else {
    read e;
    read f;
}
read x;
read a;
```

1. Perform a *May* and a *Must-Analysis* on this program, assuming an LRU-cache with associativity 4 that is empty at the start of the program. Is it possible to determine whether the last access to `a` results in a cache hit or a cache miss? Does this change if we assume that the initial cache state is unknown?

2. We now assume that the cache uses the FIFO replacement policy instead. Could an analysis determine whether the last access to `a` results in a cache hit or a cache miss if the cache is empty at the start of the program? Does this change if we assume that the initial cache state is unknown?

Exercise 7.3: Evict/Fill (9=3+3+3 Points)

In this exercise we consider a cache with associativity $k = 4$.

1. Using FIFO replacement policy, find a cache state that contains a specific element a, and for which the access sequence (b, c, d, e, f, g) of length 6 does not evict this element. Can you also find such a sequence of length 7?

2. Using FIFO replacement policy, find a cache state such that the cache does not contain all of the elements g, h, i, j after executing the access sequence $(a, b, c, d, e, f, g, h, i, j)$. Can you also find such a sequence of length 11?

3. Using PLRU replacement policy, can you find a cache state and a sequence of accesses such that the number of cache misses is infinite, but one element survives in the cache without ever being accessed? If not, explain why.