Lab 3

CMPUT 229

University of Alberta

Outline

LRU Approximation using B-Trees

For this assignment, you will be simulating an n-way associative cache.

- In a cache, is ideal to remove the least recently used (LRU) entry, since it is the one less likely to be required again.
- LRU is to expensive to be implemented in hardware.
- Approximation algorithms exist, like the one based on a MRU entry tree.
  - The idea is to create a balanced binary tree, with one leaf per cache entry.
  - Each node of the tree has a label, initialized to zero.
  - When an element is "used", from bottom-up, depending on the direction of the child relative to the parent the label of the parent must be updated: if it comes from the right child, the parent's label is set to 1, 0 otherwise.
  - To read the LRU, starting from the root, if the label is zero take left; if it is 1, take right until reaching a leaf.

LRU Approximation using B-Trees

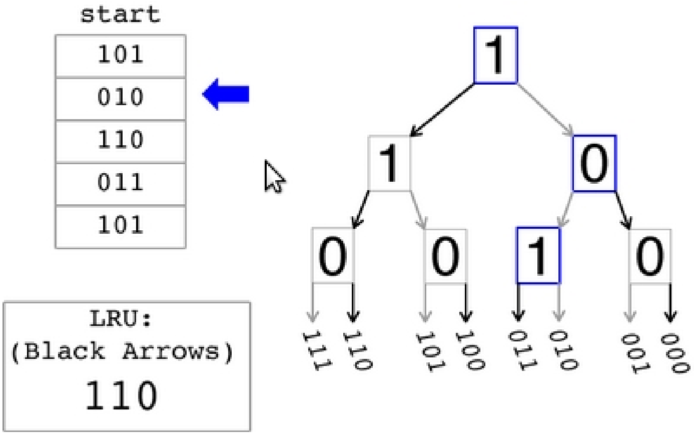

Figure: Sample tree after inserting the second element.

Lab 3 Assignment

You need to implement a cache simulator, coding the following two subroutines:

- *startCache:* receives an argument which indicates the associativity of the cache. Initializes the cache with zeros on the tree.
- *getLRU:* receives a stream of references (bits or labels) to "process" in the cache and returns the LRU.

startCache

- *Parameter:* a0 the associativity of the cache.
- *Executes:*
    - This is only called once at the beginning of the execution.
    - Initialize the tree to zeros on all paths.

getLRU

- *Parameter:*
    - a0 - pointer to a stream of bits.
    - a1 - the number of references to be read from the stream
- *Executes:*
    - Read each reference ($\log_2(n)$ bits long) and process them all.
- *Returns:*
    - a0 - the identifier or reference of the approximate LRU.

Caller/Callee Conventions

In RISC-V the following convention for calling subroutines applies:

- When calling another routine, the caller or callee is responsible for saving some registers.
- RISC-V does not save automatically these registers, it is the coder's responsability.
- There are two types of registers: callee-saved and caller-saved.
- Callee-saved (`s` registers) must be restored to the same value they had when the callee obtained control. This should always be done, as the caller expects all the `s` values untouched.
- Caller-saved registers are the temporary registers `t`. Programers should use them freely as the caller is obliged to save them before calling another routine.

Assignment Tips

- Read specifications very carefully. Pay special attention to what you have to include - we don't want a `main` method.
- See the specification on the website for useful code.
- Adhere to the calling convention to avoid losing marks.
- Test your assignments on the virtual machine before you submit. That's where we'll be marking them.
- Look at the marksheet to get an idea of how the grading will be done.
- Style marks are easy marks. Format your code like the `example.s` file we provided, and write good comments.
- Be sure to submit code that runs and loads. Otherwise you will lose many marks.

Lab 3 Questions?