Objective

I will implement a simulator to study the TLB behavior of a running process by examining the output of `valgrind` for different programs and different sizes of instances.

The command line invocation

The simulator is invoked as follows:

`valtlb379 [ -i ][ -f` *flushperiod* `] -p` FIFO|LRU *pgsize tlbsize*

Where *pgsize* is the assumed page size (always a power of 2 from 16 to 65,536), *tlbsize* is the size of the TLB in number of page table entries (always a power of 2 and no more than 2,048), `-p` specifies the TLB eviction policy (FIFO or LRU), `-i` is an optional parameter which instructs `valtlb379` to completely ignore from processing the memory references that are instruction fetches (type `I` in the output of `valgrind`), `-f` is an optimal parameter which if provided indicates that the TLB is flushed (cleared of all entries) to simulate the impact of a potential context switch once every *flushperiod* (typical values are once a million) address references.

`valtlb379` must work as a pipe insofar its input consumption is concerned. Its input will be the output of `valgrind` and specifically it should be able to deal with the output produced by the following style `valgrind` invocation (and only for this style):

```
valgrind --tool=lackey --trace-mem=yes ...program...
```

At the end of execution, `valtlb379` reports three numbers, each one on a separate line: the number of memory references handled, the number of TLB misses, and, the number of TLB hits. You will get this output and produce visual representations and statistics about the misses versus various sizes of the TLB and various other parameters. You can use Unix utilities (like `gnuplot`) to plot the misses/hits vs. TLB size for various inputs (program executions).

Behavior specification

The implementation of the TLB simulator should be able simulate the behavior of a TLB that stores and handles, indiscriminately, page table entries needed for both instruction **and** data references. The exception is when `-i` is provided, in which case the instruction address references are completely ignored (they do not even count for the *fludhperiod*). Initially the TLB is empty, and the first few references will certainly result in misses. If the `-f` parameter is supplied, it means are pretending a context switch that invalidates the TLB is happening every *flushperiod* references. After a flush you should again experience a number of cold misses.

For simplicity, the TLB cache is assumed to be fully associative and that the page table is a single-level page table. The entries of the TLB are evicted based on the selected policy.

A good quality implementation, does not (really cannot!) wait for the input to be read in its entirety before doing any processing. Rather, it consumes its input continuously, adjusting the internal data structures and counters as it goes. Note that

the input can be very long several billion of references. That is, you should be processing in a "streaming" fashion, i.e., as the input is delivered from the output of `valgrind` .