LRU replacement policy

Least Recently Used (LRU) replacement policy is used to replace the cache line or page that is least recently used.

For Block/Line replacement in Associative Caches

Since cache management is purely done in hardware, implementing this algorithm can be expensive in terms of bit needed for maintaining history of references. The pseudo LRU version is implemented using binary search approach. Example four-way set associative requires three bits to keep track of most recently used line. In order understand the diagram below refer to following definition:

- each bit represents one branch point in a binary decision tree
- let 1 represent that the left side has been referenced more recently than the right side, and 0 vice-versa

```
are all 4 lines valid?
/       \
  yes     no, use an invalid line
    |
    |
bit_0 == 0?
/       \
y        n
/       \
bit_1 == 0? bit_2 == 0?
/       \       /
 behavioral state | replace        ref to | next state
---|-----|----------------|---|------------
y 00x | line_0       line_0 | 11_
    /
 01x | line_1       line_1 | 10_
    /
 1x0 | line_2       line_2 | 01_
    /
 1x1 | line_3       line_3 | 00_
    /     /
 y      y
\       \        ('x' means don't care)
line_0 line_1 line_2 line_3
```

For Page Replacement in Virtual Memory

A perfect implementation of LRU requires a timestamp on each reference, and the OS needs to keep a list of pages ordered by the timestamp. However, this implementation is too expensive considering the frequency of memory references. The common practice is to approximate the LRU behavior. One such approximation is done using clock algorithm.

Clock Algorithm

The idea of approximating the LRU replacement policy is to replace an old page, not the oldest page. The clock algorithm arranges physical pages in a circle, with a clock hand. A use bit is used for each physical page. The use bit is set to 1 on each reference. If the use bit is not set, it means that the page has not been referenced for some time.
On page fault, the clock hand starts to sweep clock-wise. If it encounters a use bit that is set to 1, it sets it to 0 and moves on. If the use bit is 0, the page is chosen for replacement. In the worst case all use bits might be set and the pointer cycles through all the frames, giving each page a second chance.

A slow moving hand means that pages are either found quickly, or there are few page faults. A quick moving hand means lots of page faults, or lots of use-bit sets. One simple way to view the clocking algorithm is that of a crude partitioning of pages into young and old categories.

For more history (gain additional ordering information), we can record the reference bits at regular intervals. For example, we can keep 8-bit for each page. The OS shifts the bit for each page from MSB side, shifting the other bit right by 1 and discard the low-order bit. E.g. Page with 00000000 means the page has not been used for eight-time interval. 11111111 means a page that is used at least once in each period. 1100100 mean a page that is used more recently over another page whose bits are 00111111. Note that more reference bits will get us closer to actually implementing LRU policy, however more information we have, more processing required to choose a page.