Clock Algorithm(22.8 Approximating LRU) 구현

제공된 시뮬레이터의 CLOCK policy를 아래 알고리즘 대로 수정하시오.

소스코드 paging-policy.py 에서는 설명1 처럼 무작위로 선택하게 구현되어 있지만 이를 아래 설명2 처럼 수정한다.

HW-Paging-Policy.tgz

설명1
“This variant randomly scans pages when doing a replacement; when it encounters a page with a reference bit set to 1, it clears the bit (i.e., sets it to 0); when it finds a page with the reference bit set to 0, it chooses it as its victim.”

코드의 관련 부분

```
        elif policy == 'CLOCK':
            if cdebug:
                print 'REFERENCE TO PAGE', n
                print 'MEMORY ', memory
                print 'REF (b)', ref

            # hack: for now, do random
            # victim = memory.pop(int(random.random() * count))
            victim = -1
            while victim == -1:
                page = memory[int(random.random() * count)]
                if cdebug:
                    print '  scan page:', page, ref[page]
                if ref[page] >= 1:
                    ref[page] -= 1
                else:
                    # this is our victim
                    victim = page
                    memory.remove(page)
                    break
```

설명2
“How does the OS employ the use bit to approximate LRU? Well, there could be a lot of ways, but with the clock algorithm [C69], one simple approach was suggested. Imagine all the pages of the system arranged in a circular list. A clock hand points to some particular page to begin with (it doesn’t really matter which). When a replacement must occur, the OS checks if the

currently-pointed to page P has a use bit of 1 or 0. If 1, this implies that page P was recently used and thus is not a good candidate for replacement. Thus, the use bit for P set to 0 (cleared), and the clock hand is incremented to the next page (P + 1). The algorithm continues until it finds a use bit that is set to 0, implying this page has not been recently used (or, in the worst case, that all pages have been and that we have now searched through the entire set of pages, clearing all the bits).”

보고서 내용

1) 수정 개념 설명 - 추가한 자료 구조 등
2) 다양한 테스트 케이스 및 설명
3) 소스 코드 - 충분한 주석