S 162 Section 5: Address Translation & Caches and TLBs

True/False

1. LRU caches are more complex than random caches.

2. The optimal page replacement strategy is to evict the page which will be accessed least frequently on average.

Short Answer

1. If a computer has a 32 bit address space, and 1K (i.e. 2^10 bytes) sized pages, how many page table entries does it have? (fine to express answer base-2).

2. If you have a very large virtual address space, what are the benefits of using an inverted page table? What are the drawbacks?

3. Below is pseudocode for the clock page replacement algorithm (we are assuming the clock is entirely "full" with pages, and a new page must be found). Fill in the blank:

```
replaced = false;
while (!replaced) {
     1.


     2. Advance clock hand
}
```

Long Answer

Caching: Assume a computer system employing a cache, where the access time to the main memory is 100 ns, and the access time to the cache is 20ns.

1. Assume the cache hit rate is 95%. What is the average access time?

2. Assume the system implements virtual memory using a two-level page table with no TLB, and assume the CPU loads a word X from main memory. Assume the cache hit rate for the page entries as well as for the data in memory is 95%. What is the average time it takes to load X?

Longer Answer (From Fall 2012 Midterm)

Assume a system with a two level page table. The virtual memory address space is 32 bits and the physical memory address space is 16 bits.

1. Make sure that each translation table fits in a page.
   a. What is the optimal page size?
   b. Specify the length of each field in the virtual address.

(bit 31) (bit 0)

Virtual P1 Index	Virtual P2 Index	Offset

2. Assume you add to your system a 4 way set-associative data cache with 16 total cache blocks. Each block in the cache holds 8 bytes of data. In order to address a specific byte of data, you will have to split the address into the cache tag, cache index and byte select. Assume there are no modifiers bits in the table.

(bit 31) (bit 0)

tag	index	byte select