Lecture Notes on Operating Systems

Problem Set: CPU caches

1. Consider a 4-way associative cache of size 4MB, and 64-byte cache lines. Assume that the system uses 32-bit addresses. How many bits of the address will form the tag in the cache?

2. Consider a 8MB cache with 64 byte cache lines. The system uses 32 bit memory addresses. How many bits are required to store the tag in a cache entry in each case below?
   
   (a) The cache is set associative with 8-way associativity.
   
   (b) The cache is direct mapped.

3. Consider a CPU with a 16MB 8-way associative cache. The size of each cache line is 64 bytes, and the size of virtual addresses in the system is 32 bits. Calculate the number of bits of the virtual address that serve as the tag in a cache entry.

4. Consider a multicore system where multiple CPU cores have their separate L1 caches, and use the MESI cache coherence protocol between them. A cache line is in the modified state in the L1 cache of core 0, and a thread on core 1 wishes to read the same cache line. What will be the state of the cache line in core 0 after this event?

5. Consider the scenario in the previous question with one change: the thread in core 1 now wishes to write to the same cache line. What will be the state of the cache line in core 0 after this event?

6. A program has an array of \( N \) integers, which is shared across \( N \) kernel threads. The kernel thread \( i \) updates the \( i \)-th integer in the array during its execution. This design ensures no sharing of cache lines between the threads when the threads execute on separate CPU cores. Answer true/false.

7. Consider a system with two CPU cores. An application with two threads T1 and T2 runs on this system, with each thread pinned to a core. Each core has its own separate single-level cache, and the two caches are kept synchronized using the MESI cache coherence protocol. The variable lock is shared by the two threads, and is initialized to 0. Shown below are a series of steps executed by the two threads. The steps are executed one after the other and not concurrently. Assume that the variable lock has not been used by either thread, and hence is not present in either cache, at the start of this sequence. At the end of each step, write down the state of the cache line (modified/exclusive/shared/invalid) containing the variable lock in the core running thread T1. There is no need to explain your answer.

   (a) T1 executes `test_and_set(&lock);`
(b) T2 executes while (lock == 1);
(c) T1 executes lock = 0;
(d) T2 executes test and set (&lock);

8. Consider a computer system with a single core CPU, a single level of cache of size 4MB, and main memory. It takes one CPU cycle to access a memory byte if it is in cache, and 145 cycles if the memory access incurs a cache miss and must be fetched from main memory. The size of the cacheline is 64 bytes. Consider two arrays A and B, each of \( N = 2^{20} \) integers (assume that an integer requires 4 bytes of storage). The arrays are stored contiguously in memory, and are aligned at cacheline boundaries. Each case below shows an access pattern of the arrays A and B, and the parameters of the cache. For each case, calculate the average time required to access a single element of array A (averaged over all accesses to A in the specified case). Assume that the cache is empty at the start of every scenario, and no other process is using the cache. Assume that the cache does not use any optimizations like prefetching.

(a) A direct mapped cache, and every element of A is read in sequence as follows.

\[
\text{for}(i = 0; i < N; i++) \\
\text{read A}[i];
\]

(b) A direct mapped cache, and every element of A and B is read in sequence as follows.

\[
\text{for}(i = 0; i < N; i++) \\
\text{read A}[i]; \\
\text{read B}[i];
\]

(c) A fully associative cache, and the access pattern is as shown in part (b).

(d) A 2-way set associative cache, and the access pattern is as shown in part (b).

(e) A 2-way set associative cache, and the access pattern is as follows.

\[
\text{for}(i = 0; i < N; i++) \\
\text{read A}[i]; \\
\text{for}(i = 0; i < N; i++) \\
\text{read A}[i];
\]