Problem 2: Cache (30 pts)

Assume we have a dual-core processor with a 2MB shared set-associative last-level cache that has 64B cachelines and 4096 sets. The cache uses LRU cache replacement policy.

A) [2 pts] What is the associativity of the last-level cache (how many ways are there in a set)? Show your calculation.

8

Assume we have an OS which does not support virtual memory and all cores write directly to physical memory. Sleep-deprived Hongyi wrote a simple program called USELESS and he wants to run 2 USELESS processes simultaneously on the processor.

```
function USELESS(int processID)
{
  ADDRESS base = 1048576 * processID; // This is a physical address

  malloc(base, sizeof(BYTE) * 2097152); // Alloc some physical space

  int i = 0;

  while (1) {

    *(base + (i % 6) * 262144)++; // access physical memory directly (no virtual memory)

    i++;
  }
}
```

Hint—Power of 2 table:

2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}
2048	4096	8192	16384	32768	65536	131702	262144	524288	1048576	2097152

B) [4 pts] What is the steady-state last-level cache miss rate of a USELESS process when it is running alone on the system? Show your work.

0%

Continued . . .

C) [18 pts] In order to maximize the system performance running two USELESS processes, Hongyi considers three configurations:

- Static cache partitioning: Two processes execute concurrently on separate cores sharing the last-level cache and the cache is statically partitioned. Each process is assigned an equal number of ways in each set.
- Utility-based cache partitioning: Two processes execute concurrently on separate cores sharing the last-level cache and the cache is partitioned using the Utility Based Cache Partitioning mechanism that was discussed in class and that was also part of your required readings.
- Thread multiplexing: Both processes are active in the system but only one process executes at a given time. Every 1000 cache accesses, the other process is switched in. Assume that the context switch does not incur any overhead or extra cache accesses to switch in the other thread.

Calculate the steady-state cache miss rates of both processes for each configuration, and show your work:

1) Static cache partitioning:

100%
100%

2) Utility-based cache partitioning:

100%
0%

3) Thread multiplexing:

0.6%
0.6%

D) [6 pts] Hongyi also proposes a fourth configuration with virtual memory support:

- Page coloring: Two processes execute concurrently on separate cores sharing the last-level cache and the cache is partitioned using the Page Coloring mechanism that was discussed in class.

Calculate the steady-state cache miss rate of both processes for this new configuration, and show your work:

0%
0%