Problem 1. (20 points) [CS&G 2.3] There are two dominant models for how parent and children processes relate to each other in a shared address space. In the heavyweight so-called process model, when a process creates another process, the child gets a private copy of the parent’s image; that is, if the parent had allocated a variable $x$, then the child also finds a variable $x$ in its address space, and the variable is initialized to the value that the parent had for $x$ when it created the child. However, any modifications that either process makes subsequently are to its own copy of $x$ and are not visible to the other process. In the lightweight threads model, the child process or thread gets a pointer to the parent’s image, so that it and the parent now see the same storage location for $x$. All data that any process or thread allocates is shared in this model, except the data on a procedure’s stack.

(a) Consider the problem of a process having to reference its process identifier $\text{pid}$ in various parts of a program, in different routines called (in a call chain) by the routine at which the process begins execution. How would you implement this in the first model? In the second? Do you need private data per process, or could you do this with all data being globally shared?

(b) A program written in the process model may rely on the fact that a child process gets its own private copies of the parents’ data structures. What changes would you make to port the program to the threads model for data structures that are (i) only read by processes after creation of the child and (ii) are both read and written?

Problem 2. (20 points) The write-back strategy and dirty bits are related to the write strategy of a cache. Assume that we have a memory in which the largest address referenced is FFFF. We also have a cache which has a 16 blocks or lines.

Consider the following references to this cache:

1. read 04CF
2. read F3C7
3. write 0423
4. write 0433
5. write 2BC4
6. read 04BF

This sequence is repeated 20 times, for a total of 120 accesses.

Assume—

This cache is empty initially and the entire block is fetched on a miss
The block size is 16 words
The cache allocates storage on write misses and uses the write-back replacement policy.

(a) What is the total number of misses (read misses + write misses) for the cache, if the cache is direct mapped?

(b) What is the total number of write-backs of dirty blocks for this direct-mapped cache?
(c) Find the smallest direct-mapped cache size such that there are no misses besides compulsory misses.

(d) If the cache is 2-way set-associative, find the smallest cache size such that there are no misses other than compulsory misses.

**Problem 3. (25 points)** For each of the four different kinds of cache organizations, display one sequence of block references for which the organization outperforms the other three organizations (i.e., one sequence for which direct mapping performs the best of all organizations, one sequence for which fully associative is the best of all, etc.; four sequences altogether). To make your answers short, assume—

- there are four block frames in the cache;
- the set or sector size (where applicable) is 2 block frames;
- in all cases, LRU replacement is used.

None of your examples should require more than about ten block references.

**Problem 4. (10 points)** The average memory access time (AMAT) is defined as

\[
AMAT = \text{hit\_time} + \text{miss\_rate} \times \text{miss\_penalty}
\]

(a) Find the AMAT of a machine 100 MHz machine, with a miss penalty of 20 cycles, a hit time of 2 cycles, and a miss rate of 5%.

(b) Suppose doubling the size of the cache decrease the miss rate to 3%, but causes the hit time to increases to 3 cycles and the miss penalty to increase to 21 cycles. What is the AMAT of the new machine?

**Problem 5. (25 points)** A computer system has a processor whose internal clock runs at 100 MHz. Its memory system consists of a single cache (L1) and a main memory. The cache is implemented within the processor chip with an access time of 1 clock cycle and a hit ratio of 90%. It takes 10 clock cycles to access main memory.

(a) What is the average memory access time \( t_a \) of this memory system?

(b) We have learned that doubling the cache size reduces the miss ratio by roughly 30%. Given that the cache size is 256K bytes, what is the expected percentage improvement in the average memory access time \( t_a \) if the cache size is doubled to 512K bytes?

(c) Assume we have a second-level (L2) cache in the memory system. It takes 3 cycles to access data from it. It has a global hit ratio of 98%. Assume instruction execution time for the processor is 1 cycle. Each instruction requires, on average, 1 memory access for the instruction and 1.5 accesses for data.

The processor can overlap instruction execution with one (L1) cache access time. The hit ratio for the L1 data cache is still 90%, and the hit ratio to the L1 instruction cache is 95%. What is the CPI? What is the average MIPS rate of the processor?

(d) We now add virtual memory to the system. The TLB is internal to the processor chip and takes 5 ns. to do a translation on a TLB hit. The TLB hit ratio is 98%, the segment table hit ratio is 100%, and the page table hit ratio is 50%. What is the average memory access time with virtual memory?
Hint: \( t_a = t_{\text{TLB}} + (1 - \text{TLB hit ratio}) \times (T_{\text{seg}} + T_{\text{page}}) + T_{\text{cache}} + T_{\text{main}} \times \text{cache miss ratio}. \)

(e) Comment on the efficacy & performance of this system.