Problem 1:

Compare the performance of three cache designs for a byte-addressed memory system:
- **Cache 1**: A direct-mapped cache with four blocks, each block holding one word.
- **Cache 2**: A 16B 2-way set associative cache with 4B blocks and LRU replacement policy.
- **Cache 3**: A 16B fully associative cache with 4B blocks and LRU replacement policy.

For the following sequences of memory accesses starting from a cold cache, calculate the miss rate of each cache if the accesses are repeated for a large number of times. All addresses are given in decimal (not hexadecimal).

a. Memory Accesses: 0, 4, 0, 4, (repeats)

   Cache 1: ____0%____   Cache 2: ____0%____   Cache 3: ____0%____

b. Memory Accesses: 0, 16, 32, 0, 16, 32, (repeats)

   Cache 1: ____100%____   Cache 2: ____100%____   Cache 3: ____0%____

c. Memory Accesses: 0, 4, 8, 12, 16, 0, 4, 8, 12, 16, (repeats)

   Cache 1: ____40%____   Cache 2: ____60%____   Cache 3: ____100%____

d. Memory Accesses: 0, 4, 8, 12, 16, 12, 8, 4, 0, 4, 8, 12, 16, 12, 8, 4, (repeats)

Problem 2:

**Question 1:**

a. You are given a 16 KiB direct-mapped cache with 128 B blocks and a write-back policy. Assume a 64-bit address space and byte-addressed memory.

Tag: ___50___  Index: ___7___  Offset: ___7___

b. We have a 32-bit byte-addressed machine with an 8-way set-associative cache that uses 32 B blocks and has a total capacity of 8 KiB.

Tag: ___22___  Index: ___5___  Offset: ___5___

**Question 2:**

Look at the following snippet of code.

```c
#define LENGTH 16384  // 16384 = 2^14
char A[LENGTH];
for (int i = 0; i < LENGTH; i += 64) A[i] = A[i + 32];  // Loop 1
for (int i = LENGTH / 4; i >= 1; i /= 2) A[i] = A[i * 2];  // Loop 2
```

Let’s use the cache parameters given in **Part 1a**. Assume that A[0] is at the beginning of a cache block and that the cache is initially empty.

a. What is the hit rate of Loop 1? __________

The hit rate is 75%. This is because the block size is 128 bytes, and we are looking at indices i + 32, i, i + 96, and i + 64. The first access will miss and the last 3 will hit.

b. What type(s) of misses occur in Loop 1? ____________________________

Compulsory misses only. The cache is just large enough to fit the array, so there are no conflict misses.

c. What is the hit rate of Loop 2? __________

The hit rate is 100% because the entire array has been loaded into the cache.

d. What is the hit rate of Loop 2 if the cache was emptied after Loop 1? ______

18/26. Index accesses are: 8192-m, 4096-m, 4096-h, 2048-m, 2048-h, 1024-m, 1024-h, 512-m, 512-h, 256-m, 256-h, 128-m, 128-h, 64-m, 64-h, 32-h, 32-h, 16-h, 16-h, 8-h, 8-h, 4-h, 4-h, 2-h, 2-h, 1-h.

18 hits, 8 misses
Problem 3:

a. Calculate the AMAT for a system with the following properties:
   - L1 cache hits in 1 cycle with local hit rate 20%
   - L2 cache hits in 10 cycles with local hit rate 80%
   - L3 cache hits in 100 cycles with local hit rate 90%
   - Main memory always hits in 1000 cycles

\[
\text{AMAT} = 1 + (1 - 0.2)(10 + (1 - 0.8)(100 + (1 - 0.9)(1000))) = 41
\]

b. How slow can you go? Your system consists of the following:
   - L1 cache hits in 2 cycles with a miss rate of 20%
   - L2 cache hits in 10 cycles
   - Main memory always hits in 300 cycles

You want your AMAT to be <= 22 cycles. What does your local L2 miss rate need to be? What is the equivalent global miss rate?

\[
\text{AMAT} = \text{L1 Hit time} + \text{L1 Miss rate} \times (\text{L2 Hit time} + \text{L2 Local Miss rate} \times \text{L2 Miss penalty})
\]
\[
22 \geq 2 + 0.2 \times (10 + X \times 300)
\]
\[
X = .3, \text{ or 30\% local miss rate}
\]
\[
\text{Global miss rate} = 30\% \times 20\% = 6\%
\]

Problem 4:

a. What is the value of 0xF0000000 if interpreted as a 32-bit floating point number? Recall that the bias for an IEEE 754 32-bit float is 127.

\[
-2^{97}. \ 0xF0000000 = 1 \ 11100000 \ 00...00 = -2^{(128 + 64 + 32 - 127)} \times 1.0 = -2^{97}.
\]

b. What is the smallest number larger than your answer above (Problem 4a) which can be represented by an IEEE 754 32-bit float? Write your answer in hexadecimal.

0xEFFFFFFF. Since this is a negative number, we want the magnitude to be smaller than 2^{97}. The largest such number is 1.11111... \times 2^{-96}. \ 1 \ 11011111 \ 11...11, or 0xEFFFFFFF.

c. Using IEEE 754 32-bit floating point, what is the largest positive number x that makes this expression true: x + 1.0 = 1.0? Assume that we truncate any bits outside of the significand field. Write your answer in hexadecimal.

0x33FFFFFF. Since 1.0 contains 23 bits of precision, if the value of x was smaller than 2^{-23}, then the addition result will be lost to truncation. Since 2^{-23} = 1.0 \times 2^{-23}, the largest number smaller than 2^{-23} is 1.11111... \times 2^{-24}. This is equivalent to 0 \ 01100111 \ 11...11, or 0x33FFFFFF.