1 Code Analysis

Given the following chunk of code, analyze the hit rate given that we have a byte-addressed computer with a total memory of 1 MiB. It also features a 16 KiB Direct-Mapped cache with 1 KiB blocks.

```
#define NUM_INTS 8192 // 2^13
int A[NUM_INTS]; // A lives at 0x10000
int i, total = 0;
for (i = 0; i < NUM_INTS; i += 128) {
    A[i] = i; // Line 1
}
for (i = 0; i < NUM_INTS; i += 128) {
    total += A[i]; // Line 2
}
```

1.1 How many bits make up a memory address on this computer?

1.2 What is the T:I:O breakdown?

1.3 Calculate the cache hit rate for the line marked Line 1:

1.4 Calculate the cache hit rate for the line marked Line 2:

2 AMAT

Recall that AMAT stands for Average Memory Access Time. The main formula for it is:

\[ \text{AMAT} = \text{Hit Time} + \text{Miss Rate} \times \text{Miss Penalty} \]

We also have two types of miss rates, global and local. Global is calculated as:
Fraction of ALL accesses that missed at that level over all accesses total. Whereas
local is calculated: Fraction of ALL access that missed at that level over all access to that level total.

2.1 An L2$, out of 100 total accesses to the cache system, missed 20 times. What is the global miss rate of L2$?
If L1 had a miss rate of 50%, what is the local miss rate of L2?

Suppose your system consists of:

1. An L1 that hits in 2 cycles and has a local miss rate of 20%
2. An L2 that hits in 15 cycles and has a global miss rate of 5%
3. Main memory hits in 100 cycles

What is the local miss rate of L2?

What is the AMAT of the system?

Suppose we want to reduce the AMAT of the system to 8 cycles or lower by adding in a L3. If the L3 has a local miss rate of 30%, what is the largest hit time that the L3 can have?

3 Floating Point

The IEEE 754 standard defines a binary representation for floating point values using three fields:

- The sign determines the sign of the number (0 for positive, 1 for negative)
- The exponent is in biased notation with a bias of 127
- The significand or mantissa is akin to unsigned, but used to store a fraction instead of an integer

The below table shows the bit breakdown for the single precision (32-bit) representation.

<table>
<thead>
<tr>
<th>1</th>
<th>8</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign</td>
<td>Exponent</td>
<td>Mantissa/Significand/Fraction</td>
</tr>
</tbody>
</table>

For normalized floats:

\[ \text{Value} = (-1)^{\text{Sign}} \times 2^{\text{Exp-Bias}} \times 1.\text{significand}_2 \]

For denormalized floats:

\[ \text{Value} = (-1)^{\text{Sign}} \times 2^{\text{Exp-Bias}+1} \times 0.\text{significand}_2 \]

<table>
<thead>
<tr>
<th>Exponent</th>
<th>Significand</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Anything</td>
<td>Denorm</td>
</tr>
<tr>
<td>1-254</td>
<td>Anything</td>
<td>Normal</td>
</tr>
<tr>
<td>255</td>
<td>0</td>
<td>Infinity</td>
</tr>
<tr>
<td>255</td>
<td>Nonzero</td>
<td>NaN</td>
</tr>
</tbody>
</table>

How many zeroes can be represented using a float?
3.2 What is the largest finite positive value that can be stored using a single precision float?

3.3 What is the smallest positive value that can be stored using a single precision float?

3.4 What is the smallest positive normalized value that can be stored using a single precision float?

3.5 Cover the following numbers from binary to decimal or from decimal to binary:
   - 0x00000000
   - 8.25
   - 0x0000F00
   - 39.5625
   - 0xFF94BEEF
   - -∞

4 Extra Stuff on Caches!

4.1 Here's some practice involving a 2-way set associative cache. This time we have an 8-bit address space, 8 B blocks, and a cache size of 32 B. Classify each of the following accesses as a cache hit (H), cache miss (M) or cache miss with replacement (R). For any misses, list out which type of miss it is.

<table>
<thead>
<tr>
<th>Address</th>
<th>T/I/O</th>
<th>Hit, Miss, Replace</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b0000 0100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b0000 0101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b0110 1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b1100 1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b0110 1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b1101 1101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b0100 0101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b0000 0100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b1100 1000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.2 What is the hit rate of our above accesses?