1. Analyzing C Code

```c
#define NUM_INTS 8192
int A[NUM_INTS]; /* A lives at 0x1000 */
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 */
```

Let’s say you have a byte-addressed computer with a total memory of 1MiB. It features a 16KiB CPU cache with 1KiB blocks.

1. How many bits make up a memory address on this computer? **20**
2. What is the T:I:O breakdown? tag bits: **6** index bits: **4** offset bits: **10**
3. Calculate the cache hit rate for the line marked Line 1: **50%**
   The integer accesses are 4*128=512 bytes apart, which means there are 2 accesses per block. The first accesses in each block is a cache miss, but the second is a hit because A[i] and A[i +128] are in the same cache block.
4. Calculate the cache hit rate for the line marked Line 2: **50%**
   The size of A is 8192*4 = 2^{15} bytes. This is exactly twice the size of our cache. At the end of line 1, we have the second half of A inside the cache, while in line 2 we start accesses from the beginning of the array. Thus we cannot reuse any of the content of A and we get the same hit rate as before. Note that we do not have to consider cache hits for total, since the compiler will probably leave it in a register.

2. 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** 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>Sign</th>
<th>Exponent</th>
<th>Significand</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>8 bits</td>
<td>23 bits</td>
</tr>
</tbody>
</table>

There is also a double precision encoding format that uses 64 bits. This behaves the same as the single precision but uses 11 bits for the exponent (and thus a bias of 1023) and 52 bits for the significand.

How a float is interpreted depends on the values in the exponent and significand fields:

For normalized floats:

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

<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>

For denormalized floats:

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

1. How many zeroes can be represented using a float?

2. What is the largest finite positive value that can be stored using a single precision float?
   \[ 0x7F7FFFFF = (2 - 2^{-23}) \times 2^{127} \]

3. What is the smallest positive value that can be stored using a single precision float?
   \[ 0x00000001 = 2^{-23} \times 2^{-126} \]

4. What is the smallest positive normalized value that can be stored using a single precision float?
   \[ 0x00800000 = 2^{-126} \]

5. Convert the following numbers from binary to decimal or from decimal to binary:
   \begin{align*}
   0x00000000 & = 0 \\
   8.25 & = 0x41040000 \\
   0x000000F0 & = (2^{-12} + 2^{-13} + 2^{-14} + 2^{-15}) \times 2^{-126} \\
   39.5625 & = 0x421E4000 \\
   0xFF94BEEF & = \text{NaN} \\
   -\infty & = 0xFF800000
   \end{align*}

6. AMAT

AMAT is the average (expected) time it takes for memory access. It can be calculated using this formula:

\[ \text{AMAT} = \text{hit time} + \text{miss rate} \times \text{miss penalty} \]

Miss rates can be given in terms of either local miss rates or global miss rates. The local miss rate of a cache is the percentage of accesses into the particular cache that miss at the cache, while the global miss rate is the percentage of all accesses that miss at the cache.

Exercises

Suppose your system consists of:

- A L1$ that hits in 2 cycles and has a local miss rate of 20%
- A L2$ that hits in 15 cycles and has a global miss rate of 5%
- Main memory hits in 100 cycles

1. What is the local miss rate of L2$?
   Local miss rate = 5% / 20% = 0.25 = 25%

2. What is the AMAT of the system?
   \[ \text{AMAT} = 2 + 20\% \times 15 + 5\% \times 100 = 10 \] (using global miss rates)
   Alternatively, \[ \text{AMAT} = 2 + 20\% \times (15 + 25\% \times 100) = 10 \]

3. Suppose we want to reduce the AMAT of the system to 8 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?
   Let \( H \) = hit time of the cache. Using the AMAT equation, we can write:
   \[ 2 + 20\% \times (15 + 25\% \times (H + 30\% \times 100)) \leq 8 \]
   Solving for \( H \), we find that \( H \leq 30 \). So the largest hit time is 30 cycles.
4. Flynn Taxonomy

1. Explain SISD and give an example if available.
   Single Instruction Single Data; each instruction is executed in order, acting on a single stream of data. For example, traditional computer programs.

2. Explain SIMD and give an example if available.
   Single Instruction Multiple Data; each instruction is executed in order, acting on multiple streams of data. For example, the SSE Intrinsics.

3. Explain MISD and give an example if available.
   Multiple Instruction Single Data; multiple instructions are executed simultaneously, acting on a single stream of data. There are no good modern examples.

4. Explain MIMD and give an example if available.
   Multiple Instruction Multiple Data; multiple instructions are executed simultaneously, acting on multiple streams of data. For example, map reduce or multithreaded programs.