Question 1: Hit or Miss, it's AMAT-er of Performance  (24 min, 19 pts)
You are given a single 8 KiB direct-mapped cache with 256 B blocks and a write-back policy. Assume a 32-bit address space and byte-addressed memory. Show your work.

a) Label the fields below as Tag, Index, and Offset, and give the # of bits for each. (e.g. Tag:5) (3 pt)

<table>
<thead>
<tr>
<th>T:19</th>
<th>I:5</th>
<th>O: 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

b) Assume that A[0] is at the beginning of a cache block and that the cache is empty to begin with. Answer the questions below based on the following code.

```
char A[32768];  // 32768 is 2^15
for (int i = 0; i < 32768; i+=64) A[i] = '\0'; // 1st LOOP
for (int i = 0; i < 32768; i+=64) A[i & 1] = '\0'; // 2nd LOOP
```

i. What is the exact hit rate for the 1st LOOP? Leave your answer as a fraction. (3 pt)

\[
\frac{3}{4}
\]

ii. Which of the following types of misses occur in part (i)? (circle only one) (2 pt)

- Compulsory
- Conflict
- Neither
- Both

iii. What is the exact hit rate on the 2nd LOOP? Leave your answer as a fraction. (3 pt)

\[
\frac{2^9 - 1}{2^9} = \frac{511}{512}
\]

c) Suppose we run some code and our L1 cache hits in 2 cycles and has a local hit rate of 80%. Main memory always hits in 50 cycles. What is the AMAT of this system? (3 pt)

\[
2 + 0.2 \times 50 = 12 \text{ cycles}
\]

d) Match each term with all letters on the right that apply (you may have multiple per blank). (5 pt)

- **Interpreter**
  - F
- **Compiler**
  - A
- **Assembler**
  - C, E
- **Linker**
  - D
- **Loader**
  - B

A) Converts C code into Assembly
B) Copies program binary to memory to prepare for execution
C) Converts from MAL to TAL
D) Computes the jump address for a jal instruction
E) Computes the offset for a beq instruction
F) Directly executes a program written in source code
Question 2: What's that funky smell?! Oh yeah, it's potpourri... (26 min, 20 pts)

a) We examine a word in memory and find that it holds the value 0x20707C00.
   i. If it were a TAL MIPS instruction, what would it be? (Leave immediates and jump addresses in hexadecimal form.) Show your work. (2 pt)
      
      \[
      \text{addi } \text{s0, } \text{v1, 0x7C00}
      \]

   ii. If the word held two half-precision floating point numbers (1 sign bit, 5 exponent bits, 10 significand bits, bias = 15), what would they be? Leave your answer as an expression involving powers of twos. Show your work. (4 pt)

      \[
      \begin{align*}
      0x2070 & = (1 + 2^{-4} + 2^{-5} + 2^{-6}) * 2^{-7} \\
      0x7C00 & = \text{positive infinity}
      \end{align*}
      \]

b) Consider the following valid MAL MIPS code. (Note that line 6 is commented out.)

   1 mystery: ### Below, assume \$t0 is either 0x0 or between 0x61 and 0x7a inclusive
   2 loop:   lbu  \$t0, 0(\$a0)
   3       beq \$t0, \$0, done
   4       addiu \$t0, \$t0, -32
   5       sb    \$t0, 0(\$a0)
   6       ### jal anotherFunction
   7       addiu \$a0, \$a0, 1
   8       j    loop
   9 done:  jr \$ra

   i. In English, describe what effect this code has. Do NOT tell us line by line what it does! (3 pt)
      
      Converts a lowercase string to uppercase in memory

   ii. Suppose we uncomment line 6. What changes are needed for \textit{mystery} to follow calling conventions? Assume you’re telling another 61C student over the phone. For each addition or deletion or modification, describe it and its location in the code. (4 pt)

      Before 6: addi \$sp \$sp -8, store \$a0, \$ra on the stack. After 6: restore \$a0, \$ra and put \$sp back

   iii. Write a single \textbf{logical} TAL instruction that performs the same effect as line 4 (remember, \$t0 will be between 0x61 and 0x7a). (3 pt)

      \[
      \text{andi } \text{\$t0, \$t0, 0xDF}
      \]

c) Fill in the blanks to indicate when overflow occurs in 2's complement numbers: In the blanks place one of: “greater than 0”, “less than 0”, “greater than or equal to 0” or “less than or equal to 0” (2 pt)

   Positive Number + Positive Number, overflows if result is \underline{greater than 0}.<0
   Negative Number + Negative Number, overflows if result is \underline{less than or equal to 0}.\geq 0

d) What is the decimal value of the \texttt{int16_t} number 0x8000? How does it relate to the advantages of two's complement over one's complement? (2 pt)

   The decimal value is -2^{15}. Two’s complement has only one zero, which means that one additional negative integer can be represented.