1.) For this question, please consider the following 32-bit floating point representation of a number \( x \):
\[
\begin{align*}
x &= 0100 0010 0000 0110 1100 0000 0000 0000 \\
\end{align*}
\]
Recall that for a normalized 32-bit floating point number, the \( \text{exp} \) field has 8 bits. Therefore, the bias is \( 2^8 - 1 = 127 \).

What is the decimal value of \( x \)?

\[
\begin{align*}
S &= 0 \\
\text{exp} &= 100 0010 0 = 128 + 4 = 132 \\
E &= 132 - 127 = 5 \\
\text{frac} &= 000 0110 11 \\
M &= 1.000011011 \\
(-1)^0 \times 1.000011011 \times 2^5 &= 100001.1011 = 33.6875
\end{align*}
\]

2.) What is the 32-bit IEEE floating point representation of the decimal value -53.3125? Express your answer in binary.

\[
\begin{align*}
53 &= 110101 \\
0.3125 &= 2^{-2} + 2^{-4} = .25 + .0625 = .3125 \implies \times .0101 \\
-53.3125 &= -110101.0101 = (-1)^1 \times 1.101010101 \times 2^5 \\
S &= 1 \\
E &= 5 \\
\text{exp} &= 5 + 127 = 132 \implies 10000100 \\
M &= 1.101010101 \\
\text{frac} &= 101010101 \\
1\ 10000100\ 1010\ 1010\ 1000\ 0000\ 0000\ 000 \\
1100\ 0010\ 0101\ 0101\ 0100\ 0000\ 0000\ 0000
\end{align*}
\]

3.) Explain why an int that is stored in 4 bytes cannot always be exactly stored in an 4 byte float? Meaning, why in some cases would the value stored in the float be an approximation of the original int value?

A float only has 24 bits to represent significant digits in the fractional field versus an 4 byte int has 31 bits (excluding the sign bit) to encode significant digits. So, not every number that can be stored in 31 bits of an int could be stored precisely/exactly in the 24 bits in the fractional field of a double.

4.) Explain the relationship between the x86_64 registers %rax and %al.

\[
%\text{rax is a 64 bit register.}\ %\text{al is a 8 bit register which is the 8 lower bits of register %rax.}
\]

5.) Assume that an array variable of type long has been declared and called \text{array}. Assume that a long is stored in 8 bytes and the array contains 100 elements. Write C code using the array operator (i.e. \([\ ]\)) to indicate which element of \text{array} is being accessed by each of the 3 x86_64 memory accesses below.
For this problem, assume the starting address of array is 1000 and that the following registers store the specified values:

- $%rdi = 1000$
- $%rsi = 10$
- $%rax = 16$

For example, $(%rdi)$ is an access to $array[0]$.

The memory accesses are:

- 64 (%rdi) $array[8]$
- (%rdi, %rsi, 8) $array[10]$
- -8 (%rdi, %rax) $array[1]$

Make sure your answer is clear about which memory access each piece of C code corresponds to.

6.) Assume that an array variable of type long has been declared and called $array$. Assume that a variable of type long* has been declared and called $ptr$. Assume that a long is stored in 8 bytes and the array has 100 elements. $ptr$ has been initialized with the following code:

```
ptr = &array[4];
```

Write x86-64 assembly code that corresponds to each of the two C code statements below.

For this problem, assume the starting address of array is 1000 and that the following registers store the specified values:

- $%rdi = 1032$
- $%rsi = 10$
- $%rax = 3$

The C code statements are:

- *(ptr + 4) = 3;  movq $%rax, 32(%rdi)
- *(ptr - 2) = 10;  movq $%rsi, -16(%rdi)

Make sure your answer is clear about which memory access each piece of C code corresponds to.