Question 2 (30 points): The 16-bit *half precision* floating point representation has the following specification:

15	14 … 10	9 … 0
S	*exponent*	*fraction*

$$N = \begin{cases} (-1)^S \times 0.\mathit{fraction} \times 2^{-14} & \text{if } \mathit{exponent} = 0 \\ (-1)^S \times 1.\mathit{fraction} \times 2^{\mathit{exponent}-15} & \text{if } 0 < \mathit{exponent} < 31 \\ (-1)^S \times \infty & \text{if } \mathit{exponent} = 31 \text{ and } \mathit{fraction} = 0 \\ NaN & \text{if } \mathit{exponent} = 31 \text{ and } \mathit{fraction} \neq 0 \end{cases}$$

a) (**4 points**) What is the binary representation of -37.375 in the half-precision floating-point representation?

b) (**6 points**) `A = 0x6404` and `B = 0x4790` are two half-precision floating-point numbers. What is the true value of $A + B$ expressed in decimal notation? That is, if we have infinite precision to do the addition and store the result, what is $A + B$?

c) (**5 points**) What is the decimal value of $A + B$ computed on a machine with one guard bit, one round bit and one sticky bit?

d) (**15 points**) MIPS has division instructions for single and double precision floating point values, but not for half-precision. Write a MIPS procedure to implement division of a half-precision floating point by an integer multiple of 2. The input to your procedure is the address of a half-precision floating point value (in `$a0`), and the unsigned integer representation of a power of 2 (in `$a1`). Your procedure should perform the division and update the value in memory at the address in `$a0`.

Your code should follow calling conventions and should not use any pseudoinstructions. You may assume that the result of the division can be represented as a half-precision floating point value.