Daniel Lutes
January 31 2015
COMP 273 Assignment 1

1.

a) My fraction systems is comprised of 4 parts: the *fraction start* (8 bits), the *numerator*(16 bits), the *fraction line* (8 bits), and the *denominator* (16 bits). The *fraction start* is denoted by the binary number 1000 0001. The *numerator* uses 16 bits, the first bit denotes the sign of the number, 1 represents a "-" and 0 a "+" and the next 15 bits represent the number. the *fraction line* is represented by 1111 1111. The *denominator* is 16 bits, he first bit denotes the sign of the number, 1 represents a "-" and 0 a "+" and the next 15 bits represent the number. The addressing would work as follows: the *fraction start* would be addressed first in RAM, as it fits in 8 bits. Secondly the *numerator* would be split into 2 8 bit chunks, with the first 8 bits being address ahead of the following 8 bits. The *fraction line* would be address behind the last 8 bits of the *numerator*. Finally, the *denominator* would be stored in 2 8 bit chunks, with the first 8 bits being address ahead of the following 8 bits.

b) The Fraction 15/31 is: 1000 0001 0000 0000 0000 1111 1111 1111 0000 0000 0001 1111

c) Each fraction is 48 bits, therefore 10 fractions would be 480 bits (or 60 bytes).

d) The largest problem with this form of fraction representation is that it uses a lot of memory. ie the fraction start and the fraction line are both 8 bits, this seems unnecessary, considering that they only denote 1 of 2 options, but, this system works best with RAM, and 8 bit data storage. Another flaw with this system is that it uses a bit to represent a "-" or a "+", rather than two's complement, a more efficient method of storing negative numbers. Another limitation is that the max number is 32767_{10} and the minimum being -32767_{10}.

Addition

The basic algorithm is, determine a common denominator, and then add the two numerators, if the sign bits are both positive add as normal. if either but not both of the sign bits are negative, then subtract the negative numerator from the positive numerator. If both sign bits are negative, simply add the two numerators as if they were positive and then change the sign bit on the final answer to negative.

Subtraction

The basic algorithm is, determine a common denominator, then flip the sign bit of the second number, to denote subtraction, and simply pass this modified statement into the addition algorithm.

Multiplication

Multiply the 2 numerators, and then multiply the 2 denominators, finally, consider the sign bits, if both bits are of the same value then a 0, denoting a positive number is output. If the sign bits have opposite values then 1, denoting a negative number is output.

Division

Take the second fraction's reciprocal by passing the numerator into the denominators position, then take the denominator into the numerators position. Finally pass the new number into the multiplication algorithm.