Chapter 3 Exercise

3.2 Hexadecimal (base 16) is also a commonly used numbering system for representing values in computers. In fact, it has become much more popular than octal. The following table shows pairs of hexadecimal numbers.

	A	B
a.	0D34	DD17
b.	Removed	

3.2.1<3.2> What is the sum of A and B if they represent unsigned 16-bit hexadecimal numbers? The result should be written in hexadecimal. Show your work.

3.3 Overflow occurs when a result is too large to be represented accurately given a finite word size. Underflow occurs when a number is too small to be represented correctly a negative result when doing unsigned arithmetic, for example (The case when a positive result is generated by the addition of two negative integers is also referred to as underflow by many, but in this text book, that is considered an overflow). The following table shows pairs of decimal numbers

	A	B
a.	69	90
b.	Removed	removed

3.3.1<3.2> Assume A and B are unsigned 8-bit decimal integers. Calculate A-B. Is there overflow, underflow, or neither?

3.3.2<3.2> Assume A and B and signed 8-bit decimal integers stored in sign-magnitude format. Calculate A+B. Is there overflow, underflow, or neither?

3.3.3<3.2> Assume A and B are signed 8-bit decimal integers stored in 2's complement format. Calculate A-B. Is there overflow, underflow, or neither?

3.6 Part B The following table shows two pairs of hexadecimal numbers

	A	B
a.	42	36
b.	Removed	removed

3.6.4 [30] <3.3> Booth's algorithm is another approach to reducing the number of arithmetic operations necessary to perform a multiplication. This algorithm has been around for years and details about how it works are available on the Web. Basically, it assumes that a shift takes less time than an add or subtract and uses this fact to reduce the number of arithmetic operations necessary to perform a multiply. It works by identifying runs of 1s and 0s, and performing shifts during the run, Find a description of the algorithm and explain in detail how it works.

No solutions are provided

3.6.5 <3.3> Show the step-by-step result of multiplying A and B, using Booth's algorithm. Assume A and B are 8-bit two's complement integers, stored in hexadecimal formal.

a. 0x42 × 0x36 = 0x0DEC

Action	Multiplicand	Product/Multiplier
Initial Vals	0100 0010	0000 0000 0011 0110 0
00, nop shift	0100 0010 0100 0010	0000 0000 0011 0110 0 0000 0000 0001 1011 0
10, subtract shift	0100 0010 0100 0010	1011 1110 0001 1011 0 1101 1111 0000 1101 1
11, nop shift	0100 0010 0100 0010	1101 1111 0000 1101 1 1110 1111 1000 0110 1
01, add shift	0100 0010 0100 0010	0011 0001 1000 0110 1 0001 1000 1100 0011 0
10, subtract shift	0100 0010 0100 0010	1101 0110 1100 0011 0 1110 1011 0110 0001 1
11, nop shift	0100 0010 0100 0010	1110 1011 0110 0001 1 1111 0101 1011 0000 1
01, add shift	0100 0010 0100 0010	0011 0111 1011 0000 1 0001 1011 1101 1000 0
00, nop shift	0100 0010 0100 0010	0001 1011 1101 1000 0 0000 1101 1110 1100 0

3.10

In a Von Neumann architecture, groups of bits have no intrinsic meanings by themselves. What a bit pattern represents depends entirely on how it is used. The following table shows bit patterns expressed in hexadecimal notation.

a.	0x24A60004
b.	removed

3.10.1 [5] <3.5> what decimal number does the bit pattern represent if it is a two's complement integer? An unsigned integer?

3.10.2 [10] <3.5> If this bit pattern is placed into the Instruction Register, what MIPS instruction will be executed? Use $1, $2 to represent first and second registers, and so on

3.10.3 [10] <3.5> what decimal number does the bit pattern represent if it is a floating point number? Use the IEEE 754 standard.

3.10 Part B

The following table shows decimal numbers.

a.	-1609.5
b.	Removed

3.10.4 [10] <3.5> Write down the binary representation of the decimal number, assuming the IEEE 754 single precision format.

3.10.5 [10] <3.5> Write down the binary representation of the decimal number, assuming the IEEE 754 double precision format.

3.11<3.5> In the IEEE 754 floating point standard the exponent is stored in “bias" (also known as “Excess-N") format. This approach was selected because we want an all-zero pattern to be as close to zero as possible.

Because of the use of a hidden 1, if we were to represent the exponent in two's complement format an all-zero pattern would actually be the number l! (Remember, anything raised to the zeroth power is 1, so $1.0^0 = 1$.) There are many other aspects of the IEEE 754 standard that exist in order to help hardware floating point units work more quickly. However, in many older machines floating point calculations were handled in software, and therefore other formats were used. The following table shows decimal numbers.

a.	5.00736125×10^5
b.	Removed

3.11.1 Removed

3.11.2 [20] <3.5> NVlDIA has a "half" format, which is similar to IEEE 754 except that it is only 16 bits wide. The leftmost bit is still the sign bit, the exponent is 5 bits wide and stored in excess-56 format, and the mantissa is 10 bits long. A hidden 1 is assumed. Write down the bit pattern assuming a modified version of this format, which uses an excess-15 format to store the exponent. Comment on how the range and accuracy of this 16-bit floating point format compares to the single precision IEEE 754 standard.

3.14<3.5> The following table shows pairs, each consisting of a fraction and an integer.

	A	B
a.	1/3	3
b.	-1/7	7

3.14.4 [10] <3.5> Using the IEEE 754 floating-point format, write down the bit pattern that would represent A. Can you represent A exactly?