Lebanese American University

School of Arts and Sciences

Department of Computer Science and Mathematics

CSC 320 – Computer Organization

Problem Set 6: Arithmetic for Computers

Exercise 1

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 textbook, that is considered an overflow.) The following table shows pairs of decimal numbers.

	A	B
a.	216	255
b.	185	122

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

Solution:

a.	Underflow(-39)
b.	Neither(63)

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

Solution:

a.	Overflow(result = -215, which does not fit into an SM 8-bit format)

b.	Neither(65)

1.3 Assume A and B are signed 8-bit decimal integers stored in sign-magnitude format. Calculate A - B. Is there overflow, underflow, or neither?

Solution:

a.	Neither (39)
b.	Overflow (result = -179, which does not fit into an SM 8-bit format)

1.4 Assume A and B are signed 8-bit decimal integers stored in two's complement format. Calculate A+B using saturating arithmetic. The result should be written in decimal. Show your work.

Solution:

a.	15 - 117 = -102
b.	-105 - 42 = -128 (-147)

1.5 Assume A and B are signed 8-bit decimal integers stored in two's complement format. Calculate A-B using saturating arithmetic. The result should be written in decimal. Show your work.

Solution:

a.	15 + 117 = 127 (132)
b.	-105 + 42 = -63

1.6 Assume A and B are unsigned 8-bit decimal integers. Calculate A+B using saturating arithmetic. The result should be written in decimal. Show your work.

Solution:

a.	15 + 139 = 154
b.	151 + 214 = 255 (365)

Exercise 2
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.	0x0C000000
b.	0xC4630000

2.1 What decimal number does the bit pattern represent if it is a floating point number? Use the IEEE 754 standard.

The following table shows decimal numbers.

a.	63.25
b.	146987.40625

Solution:

a.	0x0C000000 = 0000 1100 0000 0000 0000 0000 0000 0000 = 0 0001 1000 0000 0000 0000 0000 0000 0000 sign is positive exp = 0x18 = 24 - 127 = -103 there is a hidden 1 mantissa = 0 answer = 1.0×2^{-103}
b.	0xC4630000 = 1100 0100 0110 0011 0000 0000 0000 0000 = 1 1000 1000 1100 0110 0000 0000 0000 0000 sign is negative exp = 0x88 = 136 - 127 = 9 there is a hidden 1 mantissa = 0xC60000 = $12 \times 16^{-1} + 6 \times 16^{-2}$ = .75 + .0234375 answer = 1.7734375×2^{9}

2.2 Write down the binary representation of the decimal number, assuming the IEEE 754 single precision format.

Solution:

a.	$63.25 \times 10^0 = 111111.01 \times 2^0$ normalize, move binary point 5 to the left 1.1111101×2^5 sign = positive, exp = 127 + 5 = 132 Final bit pattern: 0 1000 0100 1111 1010 0000 0000 0000 000 = 0100 0010 0111 1101 0000 0000 0000 0000 = 0x427D0000
b.	$146987.40625 \times 10^0 = 100011111000101011.011010 \times 2^0$ normalize, move binary point 17 to the left $1.00011111000101011011010 \times 2^{17}$ sign = positive, exp = 127 + 17 = 144 Final bit pattern: 0 1001 0000 0001 1111 0001 0101 1011 010 = 0100 1000 0000 1111 1000 1010 1101 1010 = 0x480F8ADA

2.3 Write down the binary representation of the decimal number, assuming the IEEE 754 double precision format.

Solution:

a.	$63.25 \times 10^0 = 111111.01 \times 2^0$ normalize, move binary point 5 to the left 1.1111101×2^5 sign = positive, exp = 1023 + 5 = 1028 Final bit pattern: 0 100 0000 0100 1111 1010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 = 0x404FA00000000000
b.	$146987.40625 \times 10^0 = 100011111000101011.011010 \times 2^0$ normalize, move binary point 17 to the left $1.00011111000101011011010 \times 2^{17}$ sign = positive, exp = 1023 + 17 = 1040 Final bit pattern: 0 100 0001 0000 0001 1111 0001 0101 1011 0100 0000 0000 0000 0000 0000 0000 0000 = 0x4101F15B40000000