Joshua Ferguson

Homework 3

3.4 [5] <§3.2> What is 4365 - 3412 when these values represent unsigned 12-bit octal numbers? The result should be written in octal. Show your work.

0753

3.5 [5] <§3.2> What is 4365 - 3412 when these values represent signed 12-bit octal numbers stored in sign-magnitude format? The result should be written in octal. Show your work.

4365=100 011 110 101 3412=011 100 001 010 exclude sign bit for 4365 -0365 -3412 result = -3777 base 8

3.7 [5] <§3.2> Assume 185 and 122 are signed 8-bit decimal integers stored in sign-magnitude format. Calculate 185 + 122. Is there overflow, underflow, or neither?

128-185=-57

122

(65) neither overflow or underflow

3.8 [5] <§3.2> Assume 185 and 122 are signed 8-bit decimal integers stored in sign-magnitude format. Calculate 185 - 122. Is there overflow, underflow, or neither?

-57 122

-179

128-177

51

overflow

3.20 [5] <§3.5> What decimal number does the bit pattern 0×0C000000 represent if it is a two’s complement integer? An unsigned integer?

it's positive so the conversion should be straight forward c=12 12*2^7

201326592

3.21 [10] <§3.5> If the bit pattern 0×0C000000 is placed into the Instruction Register, what MIPS instruction will be executed?

as binary: 0b11000000000000000000000000000

bits 28-26=011=3 corresponds to jal

the following bits is the target

when the instruction is executed the program will jump to the specified address in memory

3.22 [10] <§3.5> What decimal number does the bit pattern 0×0C000000 represent if it is a floating point number? Use the IEEE 754 standard.

as binary: 0b11000000000000000000000000000

the sign bit is 0 therefore positive

000 1100 0

the rest is the manitssa

sign+1

from wikipedia:

For a normalized number, the most significant digit is always non-zero. When working in binary, this uniquely determines this digit to always be 1; as such, it doesn't need to be explicitly stored, being called the hidden bit. The significand is characterized by its width in (binary) digits, and depending on the context, the hidden bit may or may not be counted towards the width of the significand. For example, the same IEEE 754 double-precision format is commonly described as having either a 53-bit significand, including the hidden bit, or a 52-bit significand, excluding the hidden bit. IEEE 754 defines the precision p to be the number of digits in the significand, including any implicit leading bit (e.g., p = 53 for the double-precision format).

mantissa=1

11000 is 24

exponent is 24-127=-103

sign*(2**exponent)*mantissa

2**-103

9.860761315262648e-32

3.23 [10] <§3.5> Write down the binary representation of the decimal number 63.25 assuming the IEEE 754 single precision format.

63=0b111111. .25=1/4=1/2**2=0b.01 0b111111.01

3.41 [10] <§3.5> Using the IEEE 754 floating point format, write down the bit pattern that would represent -1/4. Can you represent -1/4 exactly?

-1/4 as binary: .01 in Scientific Notation 1*2^-2

formula for precision representation:

for single precision format first bit is sign bit 30-23 is exponent the rest is the fractional component 1 01111111 10100000000000000000000

yes because it exist between the minimum and maximum floating point range: $2.0 * 10^{-38}, 2.0 * 10^{38}$

3.42 [10] <§3.5> What do you get if you add !1/4 to itself 4 times? What is !1/4* 4? Are they the same? What should they be?

both result in -1.0*10^0 both produce the same result.