Name: Jonnathon McCoy

Date: January 30, 2017

Assignment: Binary Numbers and Bitwise Operators Exercises

128 64 32 16 | 8 4 2 1

61 - (32) 0010 0000 = 29

29 - (16) 0001 0000 = 13

13 - (8) 0000 1000 = 5

5 - (4) 0000 0100 = 1

1 - (1) 0000 0001 = 0

61 = 0011 1101

128 64 32 16 | 8 4 2 1

110 - (64) 0100 0000 = 46

46 - (32) 0010 0000 = 14

14 - (8) 0000 1000 = 6

6 - (4) 0000 0100 = 2

2 - (2) 0000 0010 = 0

110 = 0110 1110

128 64 32 16 | 8 4 2 1

44 - (32) 0010 0000 = 12

12 - (8) 0000 1000 = 4

4 - (4) 0000 0100 = 0

44 = 0010 1100

Invert and add 1

0010 1100 = 1101 0011

& 0000 0001

-44 = 1101 0011

128 64 32 16 | 8 4 2 1

0011 0011 = 32+16+2+1 = 51

1000 0111 = 128+4+2+1 = 135

1111 1111 = 128+64+32+16+8+4+2+1 = 255

99 + 13 =

99 = 0110 0011

+ 13 = 0000 1101

112 = 0111 0000

90 – 8 =

8 = 0000 1000

Invert and add 1

1111 0111

+ 0000 0001

1111 1000 = -8

90 = 0101 1010

+ (-8) = 1111 1000

82 = ~~1~~ 0101 0010

101 -23 =

23 = 0001 0111

Inverter and add 1

1110 1000

+ 0000 0001

1110 1001 = -23

101 = 0110 0101

+ (-23) = 1110 1001

78 = ~~1~~ 0100 1110

112 + 20 = 132 (Would result in overflow because it is greater than 128)

112 = 0111 0000

+ 20 = 0001 0100

132 = 1000 0100 or -126 when inverted and adding 1

inverted 132 = 0111 1011

+ 1 = 0000 0001

0111 1110 = 126

In an 8-bit number, 1 number is reserved for a zero, 127 numbers are reserved for positive and 127 numbers are reserved for negative for a total of 255 potential numbers out of an 8-bit number. Negative numbers are indicated with a 1 in the 8th index. Due to the fact that 112+20 is greater than 127, it puts a 1 in the 8th index when added which would indicate that this is supposed to be a negative number which would cause an overflow.

-90 + 121 = 31 (when added as an 8-bit 2 complimentary number, results in overflow. However, when the first bit drops off, the number becomes the correct 31.)

-90 = 1010 0110

+121 = 0111 1001

287 = 1 0001 1111 or 31 when the MSB drops 0001 1111

Based upon the order of the operators this would result in overflow but due to the fact that it is an 8-bit number, when the MSB gets dropped off, the resulting number is 31 which is the correct result.

91 + 93 = 184 (Would result in overflow because it is greater than 128)

91 = 0101 1011

+ 93 = 0101 1101

184 = 1011 1000 or -78 when inverted and adding 1

inverted 184 = 0100 0111

+ 1 = 0000 0001

0100 1110 = 78

In an 8-bit number, 1 number is reserved for a zero, 127 numbers are reserved for positive and 127 numbers are reserved for negative for a total of 255 potential numbers out of an 8-bit number. Negative numbers are indicated with a 1 in the 8th index. Due to the fact that 91+93 is greater than 127, it puts a 1 in the 8th index when added which would indicate that this is supposed to be a negative number which would cause an overflow.

1010 1111

& 1011 0101

1010 0101

1011 1001

| 0001 1100

1011 1101

1110 1010

^ 0101 0011

1011 1001

1. In order to select the left 3 bits in an 8-bit operator the bit mask would be 1110 0000 with using the & operator.

Ex: data 0111 0101 = 117

mask & 1110 0000 = 224

result 0110 0000 = 96