Our 2's Complement Worksheet

I suggest doing the assignment in the following order:

1. Get __add__() working using the algorithm below (also in the ppt). Test your method using the hw1 description. It has a table of the int equivalents to Binary2C numbers for -8 to 7, so you construct Binary2C objects, add them, and know what the answer should be.
2. Get __neg__() working by flipping digits, then adding 01.
3. Get __int__() working. You might want to break it into two cases.

When a method specifies returning a new Binary2C object, make sure you do not alter the num_list of the original object.

Some practice:

Add the following pairs of 2's complement binary numbers using the following algorithm:

a) Numbers to be added are called addends. If one addend is shorter than the other, prepend its leftmost digit until they are the same length. E.g. make 0101 + 1 into 0101 + 1111. Then do addition digit-by-digit, as you would for decimal.

b) If your result has more digits than your addends, ignore the leftmost digit.

c) If both numbers were positive and your answer begins with 1, prepend a 0. If both numbers were negative and your answer begins with 0, prepend a 1.

1. 1000
+ 1000

5. 01000
+ 01

2.
```
  01000
+ 01000
```

3.
```
  01000
+ 10111
```

4.
```
  01000
+     1
```

6.
```
   0111
+  0111
```

7.
```
  10111
+     1
```

Give the negated Binary2C number by flipping digits and then adding 01 using the algorithm above (show work):

8. 1101 = 0010 + 01 = 010 + 001 = 011

9. 011

10. 0100101

11. 1000

12. 1

13. 1111 (not starting out as our 2’s comp, but informative)

14. 0

Give the signed decimal equivalent (hint – use the results from above):

15. 1101 = -011 = $-((2^2 * 0) + (2^1 * 1) + (2^0 * 1)) = -3$

16. 0100101

17. 1000

18. 1

19. 0

Give the unsigned hexadecimal equivalent (don't have to show work):

20.1101 == 0xD

21.1111

22.1010