# CS 251, Fall 2016, Assignment 2.2.0

**2%** of course mark

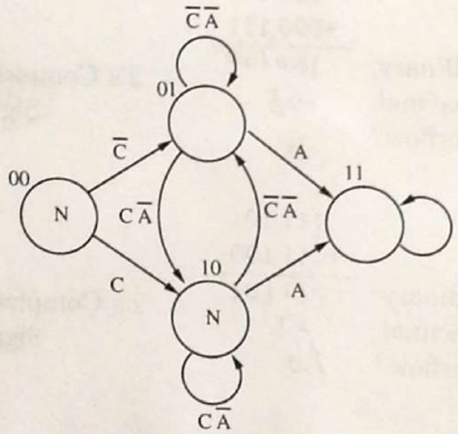
Due Friday, October 7th, 4:30 PM

1. (10 points)

   (a) (2 points) For the finite state machine below, if we start in state 00, and see the input stream CA=00,00,10,10,00, which state do we end up in?

   01

   (b) (8 points) Write a next state table and an output table for the following finite state machine.




| CA | Current state | Next state | Output |
|---|---|---|---|
| 00 | 00 | 01 | — |
| 01 | 00 | 01 | — |
| 10 | 00 | 10 | N |
| 11 | 00 | 10 | N |
| 00 | 01 | 01 | — |
| 01 | 01 | 11 | — |
| 10 | 01 | 10 | N |
| 11 | 01 | 11 | — |
| 00 | 10 | 01 | — |
| 01 | 10 | 11 | — |
| 10 | 10 | 10 | N |
| 11 | 10 | 11 | — |
| 00 | 11 | 11 | — |
| 01 | 11 | 11 | — |
| 10 | 11 | 11 | — |
| 11 | 11 | 11 | — |

2. (1 point) Convert the last 3 digits of your student id to a **10 digit binary number**. Note: it is okay to use a web page or other software to do this conversion.

787: 11000 10011

3. (1 point) Compute the negation of the 10 digit binary number of the previous question, assuming that it is given in **10-bit 2's complement** representation. Note: you should NOT use software to compute the 2's complement. Show or explain your work.

$X =$ 11000 10011 = 0 11000 10011

$\bar{X} =$ 100111 011 00

$-X = \bar{X} + 1 =$ 10011 101101

4. (8 points) Add the following pairs of 6-bit two's complement binary numbers, giving a 6-bit result (i.e., throw away the carry-out). Also give the signed decimal value of the the 6-bit result. Note whether or not an overflow occurred for any addition.

```
                         011 101
                        +000 111
2's Complement Binary:   100100
       Signed Decimal:   -28
            Overflow?    Yes
```

```
                         011 101
                        +110 100
2's Complement Binary:   1010001
       Signed Decimal:   17
            Overflow?    No
```

```
                         111 101
                        +011 100
2's Complement Binary:   1011101
       Signed Decimal:   29
            Overflow?    No
```

```
                         111 101
                        +110 101
2's Complement Binary:   1110010
       Signed Decimal:   -14
            Overflow?    No
```

5. (3 points) Compute the following bitwise logic operations. Under the "Student Id" column you should fill in the 10-bit binary number you computed for your student id in Question 2.

| | Operation | Student Id | Result |
|---|---|---|---|
| 00000 11111 | AND | 11000 10011 | 00000 10011 |
| 00000 11111 | OR | 11000 10011 | 11000 11111 |
| 00000 11111 | XOR | 11000 10011 | 11000 01100 |

6. (2 points) Multiply the following unsigned binary numbers. You should write in to the left of "Your student id" the 10-bit binary number you computed for the last 3 digits of your student id in Question 2. Show your work. You may omit the "zero lines" if you wish.

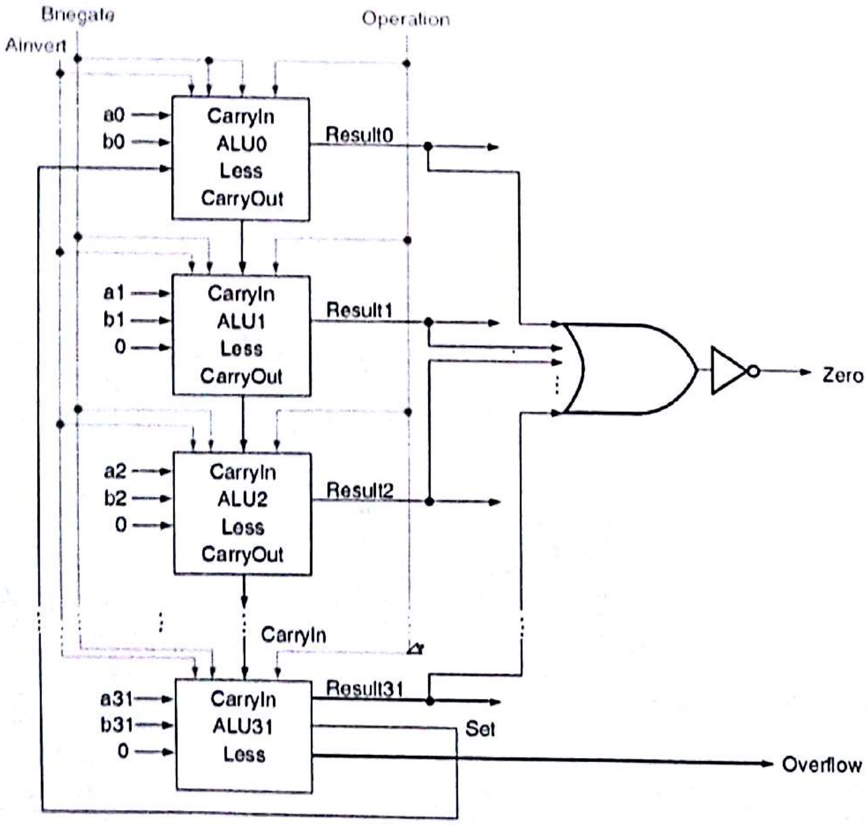
```
    1100010011   Your student id
  × 0000001011
  ------------
    1100010011
   1100010011
 1100010011
--------------
10000111010001
```

7. (6 points) Below is the diagram for a 32-bit ALU as developed in Appendix B.5 of the course text. Modify this diagram to have the following three additional outputs. In the figure, label each output with the name given in bold in the question.

(a) (2 pts) **Odd-Ones**: This should be HIGH (1) when the number of bits in the output that are 1 is odd (e.g., for a four bit output (Result0,Result1,Result2,Result3), two examples of when the Odd-Ones output would be 1 are 0001, 1011, and two examples of when the Odd-Ones output would be 0 are 0000, 0101).

(b) (2 pts) **Positive-Even**: This should be HIGH (1) when the output is a positive, even number (0 is **not** a positive number).

(c) (2 pts) **Odd-Multiple-of-4**: This should be HIGH (1) when the output is an odd multiple of 4 (e.g., high for $4=1\times4$, $12=3\times4$, and $20=5\times4$, low for $8=2\times4$, $16=4\times4$, and $24=6\times4$).



8. (3 points) Write the base 10 number $-1.6015625 \times 10^{-1}$ as a 32-bit, IEEE normalized floating point number with biased exponent.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

9. (2 points) Complete the following table, showing the sign & magnitude and the corresponding biased representations with a bias of 127.

| Sign & Magnitude | Biased |
|---|---|
| 60 | 187 |
| −13 | 114 |
| −17 | 110 |
| 84 | 201 |

10. (2 points)

Add the following pair of 4-bit floating point numbers using a 4-bit adder to add the significands, first assuming that you retain one additional bit for rounding and then assuming that you do not retain one extra bit for rounding. Assume that you round up.

$$1.100 \times 2^8 + 1.110 \times 2^6$$

Answer with extra bit: $1.1111 \times 2^8$

Answer without extra bit: 1.111

11. (5 points)

In the diagram below you have a $4 \times 4$ matrix of memory for 1 bit of data. Using two level decoding we can break up the address bits $A_3\,A_2\,A_1\,A_0$. Two address bits will be passed into the decoder and two bits into the multiplexor.

Label the inputs to the decoder and mux with $A_3, A_2, A_1, A_0$ to correctly access the bits of memory. Also label each bit of memory in the $4 \times 4$ matrix.