Homework #4 **Due:** November 01, 2016 Tuesday

Purpose: This homework is to get you familiar with itwise operators and simulation of combinational circuits in C.

Assignment:

Write a well-documented complete C-program to simulate a number of combinational circuits. The input to circuits will be specified on command line arguments in base-16. The program will be invoked as: (a4 is the executable)

```
./a4 <arg-a> <arg-b> <arg-c>
```

eg.: ./a4 4f d8 1 (look at C-function strtol() to parse the args.)

Multiple input bits will be specified with bits in arg-a or arg-b or both and carry-n in argc-c: eg., arg-a value of 4f can specify 8-inputs to a-input of circuit (bit 0 is a0, bit 1 is a1, bit 2 is a2 etc.)

The modules will take integer arguments to handle multiple inputs:

0. int main(int argc, char *argv[]);

1. void halfaddr(int a, int b, int *sum, int *outcary);
   { simulates HA on bit 0 of a and b, returns sum bit and carry bit).

2. void fulladdr(int a, int b, int incary, int *sum, int *outcary);
   { as above with 3 inputs bit 0 of a, b and incary }

3. void add4(int a, int b, int incary, int *sum, int *outcary);
   { uses ripple-carry scheme and 4 FAs to add lower order 4-bits of
   a and b and incarry is in c }

4. void add16(int a, int b, in incary, int *sum, int *outcary);
   { uses 4 add4s to add lower order 16 bits of a and b }

5. void magnitude4(int a, int *rslt);
   { finds magnitude of a (lower order 4 bits) using an add4 and xors }

6. void parity4(int a, int *outparity);
   { generates odd parity for lower order 3 bits in a }

7. void mux2x1(int a, int b, int select, int *out);
   { connects one of the 2 inputs (lower order bits in a and b) to out based
   on 1 selection input (lowest order bit of c) }

8. void printresult(char *label, int a);
   { prints all bits in groups of 8 bits, see sample output below }

Note: Pay attention to documentation, you may use make to compile your code (use the one provided with appropriate changes), and check results produced by you program for correct functionality.

Sample output:

(Input as above: a = 0x4f, b = 0x3d, c = 0x1)

```
Half-adder:
sum:        00000000 00000000 00000000 00000000
outcary:    00000000 00000000 00000000 00000001

Full-adder:
sum:        00000000 00000000 00000000 00000001
outcary:    00000000 00000000 00000000 00000001

4-bit-adder:
sum:        00000000 00000000 00000000 00001101
outcary:    00000000 00000000 00000000 00000001

16-bit-adder:
sum:        00000000 00000000 00000000 10001101
outcary:    00000000 00000000 00000000 00000000
```

etc.

Turn-in: a4.c (if no makefile is used.)
a4.tar containing tape archive of a4/Makefile and a4/a4.c
(submit to the BlackBoard)