COE 0147 Spring 2013
Lab #7: Multiplication and Division

Each of you should submit your own solution, according to the instructions at your TA's website. Each person must turn in their own copies of the lab. If you choose to work with a neighbor/partner, put your partner's name on your submitted copy of the lab.
For the written part, you should turn in a hard copy of this assignment at the beginning of the following recitation meeting either in recitation class or in your TA's mailbox. Staple multiple pages together and do not forget to put your name on. Do not present answers out of order.

1. Perform three subtractions on 9-bit 2's complement binary numbers as follows:
   a) 195 - 47
   b) 87 - 59
   c) -46 – 71

For each of these subtractions, convert the numbers into 9-bit 2's complement, add those and convert the resultant binary 2's complement number into decimal form. Show all your work.

a) $195 = 2^7 + 2^6 + 2^1 + 2^0$
$= 011000011$

-47: $47 = 2^5 + 2^3 + 2^2 + 2^1 + 2^0$
$= 000101111$

two's comp:
$$\begin{array}{r} 111010000 \\ +\,000000001 \\ \hline 111010001 \end{array}$$

$$195 - 47 = \begin{array}{r} 011000011 \\ +\,111010001 \\ \hline 1\,\boxed{010010100} \end{array}$$

$= 2^7 + 2^4 + 2^2 = 128 + 16 + 4 = 148 = 195 - 47$ ✓

b) $87 = 2^6 + 2^4 + 2^2 + 2^1 + 2^0$
$= 001010111$

-59: $59 = 2^5 + 2^4 + 2^3 + 2^1 + 2^0$
$= 000111011$

two's comp:
$$\begin{array}{r} 111000100 \\ +\,000000001 \\ \hline 111000101 \end{array}$$

$$87 - 59 = \begin{array}{r} 001010111 \\ +\,111000101 \\ \hline 1\,\boxed{000011100} \end{array}$$

$2^4 + 2^3 + 2^2 = 16 + 8 + 4 = 28 = 87 - 59$ ✓

c) -46: $46 = 2^5 + 2^3 + 2^2 + 2^1$
$= 000101110$

two's comp:
$$\begin{array}{r} 111010001 \\ +\,000000001 \\ \hline 111010010 \end{array}$$

-71: $71 = 2^6 + 2^2 + 2^1 + 2^0$
$= 001000111$

two's comp:
$$\begin{array}{r} 110111000 \\ +\,000000001 \\ \hline 110111001 \end{array}$$

$$-46 - 71 = \begin{array}{r} 111010010 \\ +\,110111001 \\ \hline 1\,\boxed{110001011} \end{array}$$

$-2^8 + 2^7 + 2^3 + 2^1 + 2^0 = -256 + 128 + 8 + 2 + 1 = -117 = -46 - 71$ ✓

2. Show the steps for the multiplication of 10010101b and 10101111b (**unsigned**) using **Hardware Design 3** (http://www.pitt.edu/~kmram/CoE0147/lectures/numbers3.pdf).
Here 10010101b is the multiplicand and 10101111b is the multiplier.

Draw a table and fill up the columns:

Iteration	Multiplicand	Implementation 3	
		Step	Product(16-bit)
0	10010101	init	0000 0000 1010 1111
1		1→ product = product + multiplicand shift right	1001 0101 1010 1111 0100 1010 1101 0111
2		1→ product = product + multiplicand shift right	1101 1111 1101 0111 0110 1111 1110 1011
3		1→ product = product + multiplier shift right	0000 0100 1110 1011 1000 0010 0111 0101
4		1→ product = product + multiplier shift right	0001 0111 0111 0101 1000 1011 1011 1010
5		0→ no op shift right	0100 0101 1101 1101
6		1→ product = product + multiplier Shift right	1101 1010 1101 1101 0110 1101 0110 1110
7		0→ noop Shift right	0011 0110 1011 0111
8	↓	1→ product = product + multiplier Shift right	1100 1011 1011 011 0110 0101 1101 1011

Show your work (addition of product and multiplicand in each iteration) on the next page.

Step 1:
0000 0000
\+ 1001 0101
10010101

Step 2:
0100 1010
\+ 1001 0101
1101 1111

Step 3:
1111 111
0110 1111
\+ 1001 0101
1 0000 0100

Step 4:
1000 0010
\+ 1001 0101
1 0001 0111

Step 6:
1 1
0100 0101
1001 0101
1 101 1010

Step 8:
11 1
0011 0110
\+ 1001 0101
1100 1011

3. Convert the following 8-bit binary numbers into Booth's encoding form:

a) 01010011 0 b) 11010101 0 c) 01000001 0

1 -1 1 -1 0 1 0 -1

0 -1 1 -1 1 -1 1 -1

1 -1 0 0 0 0 1 -1

4. Convert the following decimal numbers into 9-bit binary numbers in Booth's encoding form:

25, -201, -48

$25 = 2^4 + 2^3 + 2^0$

$= 000011001\ 0$

booth's:

0 0 0 1 0 -1 0 1 -1

-201 : $201 = 2^7 + 2^6 + 2^3 + 2^0$

$= 011001001$

two's comp:

  100110110
+000000001

100110111 (0)

booth's: -1 0 1 0 -1 1 0 0 -1

-48 : $48 = 2^5 + 2^4$

$= 000110000$

two's comp:

  111001111
+000000001

111010000 0

booth's:

0 0 -1 1 -1 0 0 0 0

5. Show the steps for the multiplication of 10010101b and 10101111b (**signed**) using **Booth's algorithm** (http://www.pitt.edu/~kmram/CoE0147/lectures/numbers3.pdf).
Here 10010101b is the multiplicand and 10101111b is the multiplier.

Draw a table and fill up the columns:

Iteration	Multiplicand	Booth's Algorithm	
		Step	Product(17-bit)
0	10010101 (+) 01101011 (-)	init	0000 0000 1010 1111 0
1		10 → prod = prod - mult Shift right	0110 1011 1010 1111 0 0011 0101 1101 0111 1
2		11 → no op shift right	0001 1010 1110 1011 1
3		11 → no op Shift right	0000 1101 0111 0101 1
4		11 → no op Shift right	0000 0110 1011 1010 1
5		01 → prod = prod + mult Shift right	1001 1011 1011 1010 1 1100 1101 1101 1101 0
6		10 → prod = prod - mult Shift right	0011 1000 1101 1101 0 0001 1100 0110 1110 1
7		01 → prod = prod + mult Shift right	1011 0001 0110 1110 1 1101 1000 1011 0111 0
8		10 → prod = prod - mult Shift right	0100 0011 1011 0111 0 0010 0001 1101 1011 1

Show your work (addition of product and multiplicand in each iteration) on the next page.

Step 1:

```
  0000 0000
+ 0110 1011
-----------
  0110 1011
```

Step 5:

```
  0000 0110
+ 1001 0101
-----------
  1001 1011
```

Step 6:

```
  1  1 11
  1100 1101
+ 0110 1011
-----------
1 0011 1000
```

Step 7:

```
  11 1
  0001 1100
+ 1001 0101
-----------
  1011 0001
```

Step 8:

```
  1111
  1101 1000
+ 0110 1011
-----------
1 0100 0011
```

6. Show the steps for the multiplication of 01100001b and 01011011b (signed) using Booth's algorithm (available here: http://www.pitt.edu/~kmram/CoE0147/lectures/numbers3.pdf). Here 01100001b is the multiplicand and 01011011b is the multiplier. Draw a table similar to the following one and fill up the columns:

Iteration	Multiplicand	Booth's Algorithm	
		Step	Product(17-bit)
0	0110 0001 (+) 1001 1111 (-)	init	0000 0000 0101 1011 0
1	↓	10 → prod = prod - mult shift right	1001 1111 0101 1011 0 1100 1111 1010 1101 1
2		11 → no op shift right	1110 0111 1101 0110 1
3		01 → prod = prod + mult shift right	0100 1000 1101 0110 1 0010 0100 0110 1011 0
4		10 → prod = prod - mult shift right	1100 0011 0110 1011 0 1110 0001 1011 0101 1
5		11 → no op shift right	1111 0000 1101 1010 1
6		01 → prod = prod + mult shift right	0101 0001 1101 1010 1 0010 1000 1110 1101 0
7		10 → prod = prod - mult shift right	1100 0111 1110 1101 0 1110 0011 1111 0110 1
8		01 → prod = prod + mult shift right	0100 0100 1111 0110 1 0010 0010 0111 1011 0

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

Step 1:
```
  0000 0000
+ 1001 1111
-----------
  1001 1111
```

Step 3:
```
  1110 0111
+ 0110 0001
-----------
1 0100 1000
```

Step 4:
```
  0010 0100
+ 1001 1111
-----------
  1100 0011
```

Step 6:
```
  1111 0000
+ 0110 0001
-----------
1 0101 0001
```

Step 7:
```
  0010 1000
+ 1001 1111
-----------
  1100 0111
```

Step 8:
```
  1110 0011
+ 0110 0001
-----------
1 0100 0100
```

7 In a table similar to the following one, show the steps for computing 00111000b (the dividend) divided by 0011b (the divisor, both numbers are unsigned) using **non-restoring division**. Non-restoring division is described online at: http://www.pitt.edu/~kmram/CoE0147/lectures/division.pdf. An example of doing non-restoring division is also shown.

Iteration	Step (description)	Divisor (4 bits)	Dividend (8 bits)	Remainder Register (8 bits)
0	init Shift left	0011	00111000	0000 0011 1000 0000 0111 0000
1	rem = rem - divisor rem<0 → SL, r0=0			1101 0111 0000 1010 1110 0000
2	rem = rem + divisor rem<0 → SL, r0 = 0			1101 1110 0000 1011 1100 0000
3	rem = rem + divisor rem<0 → SL, r0=0			1110 1100 0000 1101 1000 0000
4	rem = rem + divisor rem >0 → SL, r0=1			0000 1000 0000 0001 0000 0001
5	rem = rem - div rem<0 → SL, r0=0			1110 0000 0001 1100 0000 0010
6	rem = rem + div rem<0 → SL, r0=0			1111 0000 0010 1110 0000 0100
7	rem = rem + div rem>0 → SL, r0=1			0001 0000 0100 0010 0000 1001
8	rem = rem - div rem<0 → SL, r0=0			1111 0000 1001 1110 0001 0010
done	shift LH right rem<0 → add divisor			1111 0001 0010 0010 0001 0010

Step 1:
$$\begin{array}{r} 0000 \\ +\ 1101 \\ \hline 1101 \end{array}$$

Step 2:
$$\begin{array}{r} 1010 \\ +\ 0011 \\ \hline 1101 \end{array}$$

Step 3:
$$\begin{array}{r} 1011 \\ +\ 0011 \\ \hline 1110 \end{array}$$

Step 4:
$$\begin{array}{r} 1101 \\ +\ 0011 \\ \hline 1\ 0000 \end{array}$$

Step 5:
$$\begin{array}{r} 0001 \\ +\ 1101 \\ \hline 1110 \end{array}$$

Step 6:
$$\begin{array}{r} 1100 \\ +\ 0011 \\ \hline 1111 \end{array}$$

Step 7:
$$\begin{array}{r} 1110 \\ +\ 0011 \\ \hline 1\ 0001 \end{array}$$

Step 8:
$$\begin{array}{r} 0010 \\ +\ 1101 \\ \hline 1111 \end{array}$$