Computer Organization and Structure

Homework #3
Due: 2013/11/12

1. Given the bit pattern:

   1010 1101 0001 0000 0000 0000 0000 0010

   what does it represent, assuming that it is

   a. A two's complement integer?
   b. An unsigned integer?
   c. A single precision floating-point number? where we use the IEEE 754 floating-point standard which represents a floating-point number as $(-1)^S \times (1+F) \times 2^E$ and encodes the *S*, *F*, and *E* ordering using 1, 23, and 8 bits, respectively.
   d. A MIPS instruction?

2. With x = 0000 0000 0000 0000 0000 0000 0101 1011_{two} and y = 0000 0000 0000 0000 0000 0000 0000 1101_{two} representing two's complement signed integers, perform, showing all work:

   a. x+y
   b. x-y
   c. x*y
   d. x/y

3. Given the following three functions:

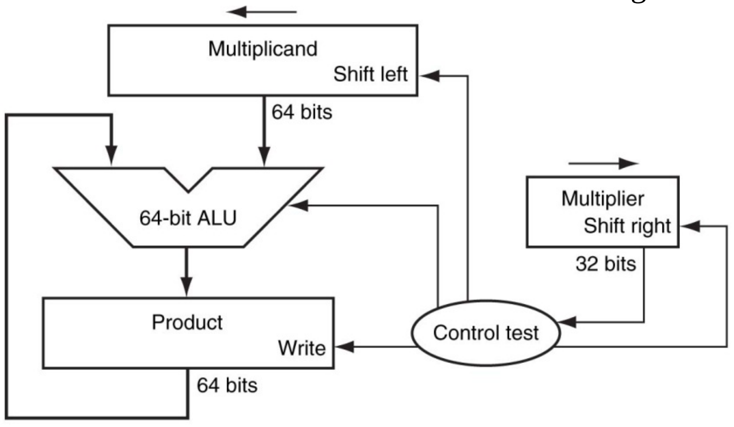
   a. A two-bit-wide *shifter* takes two input signals, i_0 and i_1, and shifts them to two outputs, o_0 and o_1, under the control of a shift signal. If this signal SHIFT is false, then the inputs are connected straight through to the outputs. If SHIFT is true, then i_0 is routed to o_1 and o_0 should be set to a 0.
   b. A two-bit *demultiplexer* takes an input signal IN and shifts it to one of two outputs, o_0 and o_1, under the control of a single SELECT signal. If SELECT is 0, then IN is connected through to o_0 and o_1 is connected to a 0. If SELECT is 1, then IN is connected through to o_1 and o_0 is connected to a 0.
   c. A two-bit *multiplexer* takes two input signals, i_0 and i_1, and shifts one of them to the single output OUT under the control of a one-bit SELECT signal. If the SELECT signal is false, then i_0 is passed to OUT. If SELECT is true, then i_1 is passed to OUT.
   d. A four-input function that outputs a 1 whenever an odd number of its inputs are 1.

   Complete the following four items:

a. Construct their truth tables.
b. What are the functions in sum of products forms? (you can just use “little *m*” notation)
c. Use the Karnaugh map method to simplify the functions in sum of products forms.
d. Draw logic schematics using AND, OR, and INVERT gates.

4. Let’s look in more detail at multiplication. We will use the numbers in the following table.

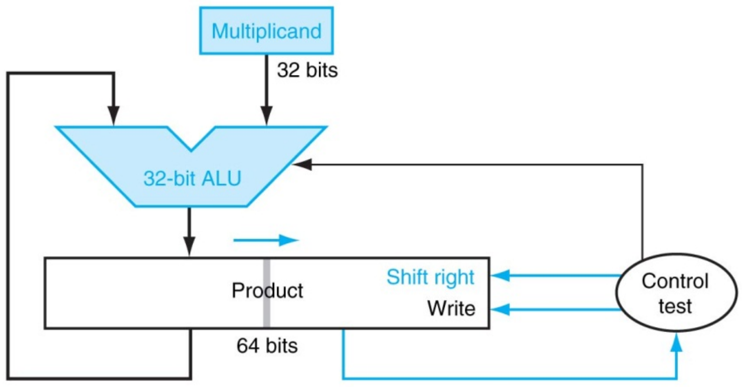
	A	B
1	50	23
2	66	04



a. Using a table similar to the following one, calculate the product of the octal unsigned 6-bit integers A and B using the hardware as shown in the right figure. You should show the contents of each register on each step.

Iteration	Step	Multiplier	Multiplicand	Product
0	Initial values	001$\boxed{1}$	0000 0010	0000 0000
1	1a: 1⇒Prod=Prod+Mcand	0011	0000 0010	0000 0010
	2: Shift left Multiplicand	0011	0000 0100	0000 0010
	3: Shift right Multiplier	000$\boxed{1}$	0000 0100	0000 0010
2	1a: 1⇒Prod=Prod+Mcand	0001	0000 0100	0000 0110
	2: Shift left Multiplicand	0001	0000 1000	0000 0110
	3: Shift right Multiplier	000$\boxed{0}$	0000 1000	0000 0110
3	1: 0⇒No operation	0000	0000 1000	0000 0110
	2: Shift left Multiplicand	0000	0001 0000	0000 0110
	3: Shift right Multiplier	000$\boxed{0}$	0001 0000	0000 0110
4	1: 0⇒No operation	0000	0001 0000	0000 0110
	2: Shift left Multiplicand	0000	0010 0000	0000 0110
	3: Shift right Multiplier	0000	0010 0000	0000 0110

b. Using a table similar to the above one, calculate the product of the hexadecimal unsigned 8-bit integers A and B using the hardware as shown in the right figure. You should show the contents of each register on each step.



5. The ALU supported set on less than (slt) using just the sign bit of the adder. Let's try a set on less than operation using the values -7_{ten} and 6_{ten}. To make it simpler to follow the example, let's limit the binary representations to 4 bits: 1001_{two} and 0110_{two}.

   $1001_{two} – 0110_{two} = 1001_{two} + 1010_{two} = 0011_{two}$

   This result would suggest that $-7_{ten} > 6_{ten}$, which is clearly wrong. Hence we must factor in overflow in the decision. Modify the 1-bit ALU in the following figures to handle slt correctly.

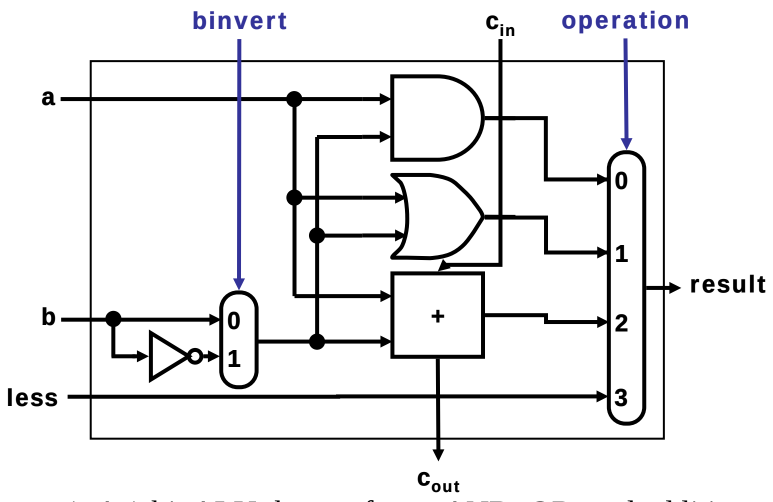


Figure 1: A 1-bit ALU that performs AND, OR, and addition on a and b or b'.

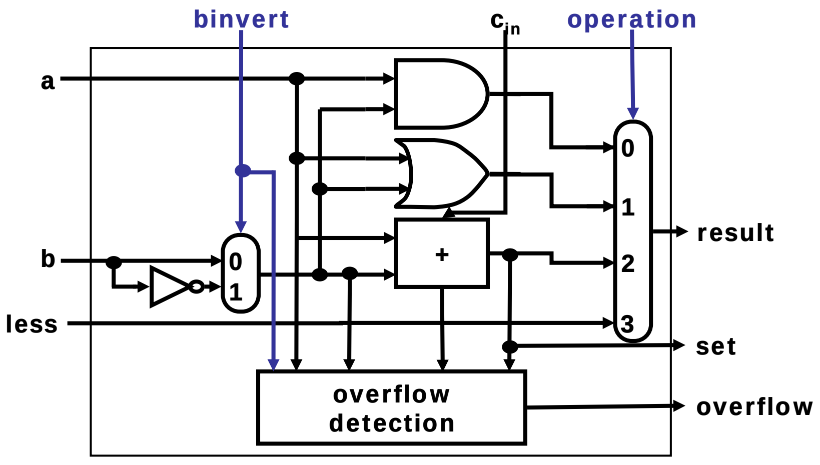


Figure 2: A 1-bit ALU for the most significant bit.