2.40 Write the decimal equivalents for these IEEE floating point

numbers.

a. 0 10000000 00000000000000000000000

b. 1 10000011 00010000000000000000000

c. 0 11111111 00000000000000000000000

d. 1 10000000 10010000000000000000000

*Answer*:

a. (-1)^0 \* 1.0 \* 2^(128-127) = 2

b. (-1)^1 \* 1.0001 \* 2^(131-127) = -17

c. Because the exponent field contains 11111111, it is positive infinity.

d. (-1)^1 \* 1.1001 \* 2^(128-127) = -3.125

2.48 Convert the following decimal numbers to hexadecimal representations of 2’s complement numbers.

a. 256

b. 111

c. 123,456,789

d. −44

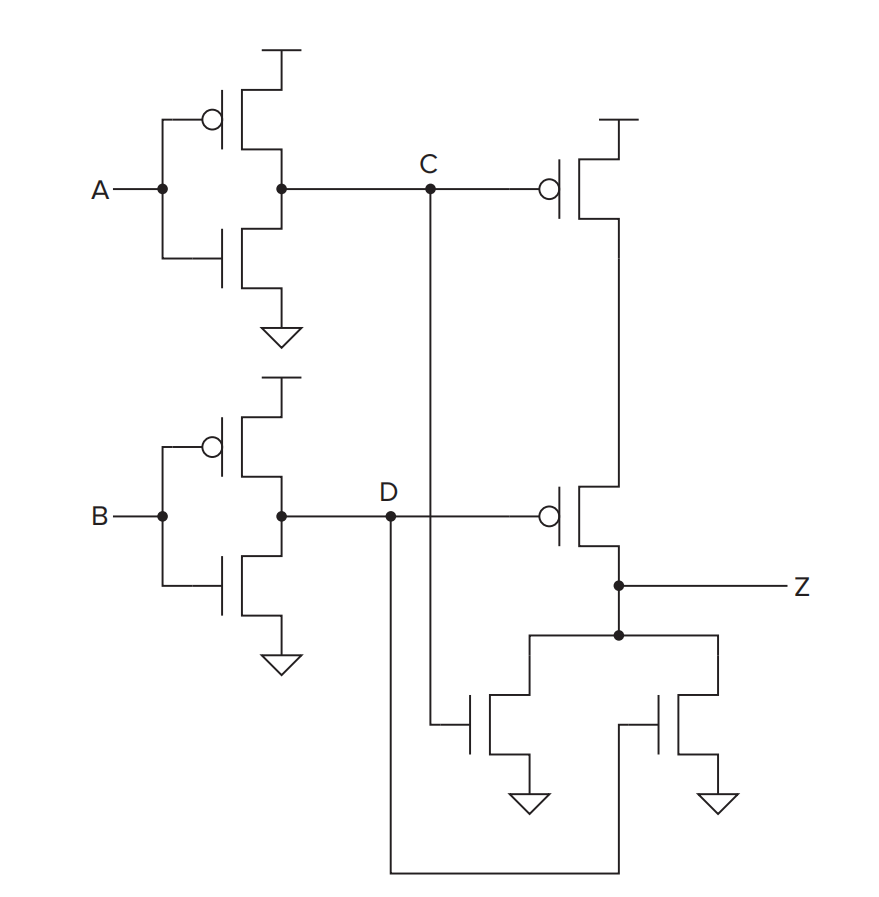
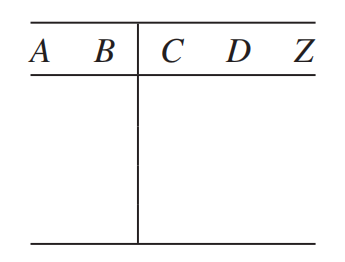
*Answer:*

a. 0001 0000 0000 B = 100 (H)

b. 0110 1111 B = 6F (H)

c. 75BCD15 (H)

d. 1101 0100 B = D4 (H)

3.6 For the transistor-level circuit in Figure 3.38, fill in the truth table. What is Z in terms of A and B?

1 1 0

1 0 0

0 1 0

0 0 1

0 0

0 1

1 0

1 1

3.20 How many output lines will a 16-input multiplexer have? How many

select lines will this multiplexer have?

*Answer*:

There is only one output line and 4 select lines.

3.30 a. Figure 3.42 shows a logic circuit that appears in many of today’s

processors. Each of the boxes is a full-adder circuit. What does the value on the wire X do? That is, what is the difference in the output of this circuit if X = 0 vs. if X = 1?

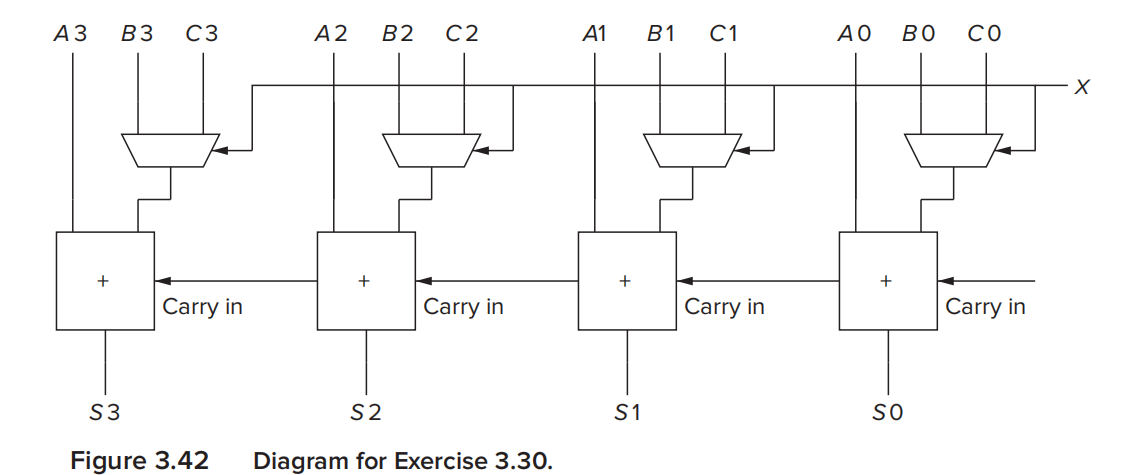
b. Construct a logic diagram that implements an adder/subtractor. That is, the logic circuit will compute A + B or A - B depending on the value of X. Hint: Use the logic diagram of Figure 3.42 as a building block.

*Answer:*

1. The function of x is the selection signal of the MUX.

If X = 0, the multiplexer will select four values of B, which means the full adder performs the A+B operation.If X = 1, the multiplexer will select four values of C, which means the full adder performs the A+C operation.

(Also, we can choose A+C when X=0 and A+B when X=1, which depends on the design of the MUX.)

1. 

*~B0*

*~B1*

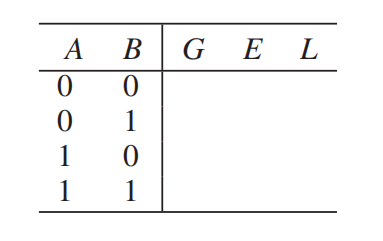
*~B2*

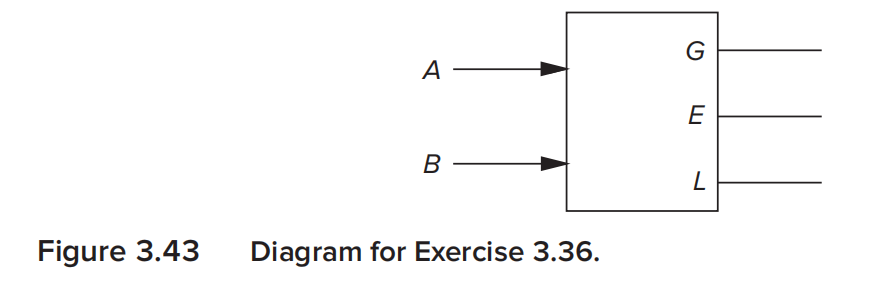
*~B3*

Change the value of C to ~B(NOT(B)), and connect X to the first carry in. So if X = 0, the MUX will still select four values of B, and the logic circuit will compute A + B.

If X = 1, the MUX will select four values of ~B, and because the first carry in equal to 1, the logic circuit will implement ~B+1, in the other words, computing A - B.

3.36 A comparator circuit has two 1-bit inputs A and B and three 1-bit outputs G (greater), E (Equal), and L (less than).

1. Draw the truth table for a one-bit comparator.



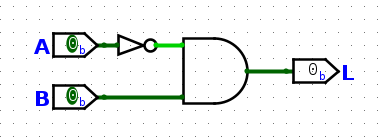
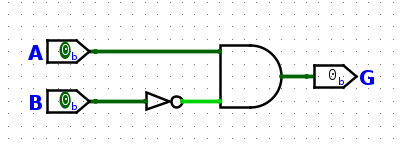
0 1 0

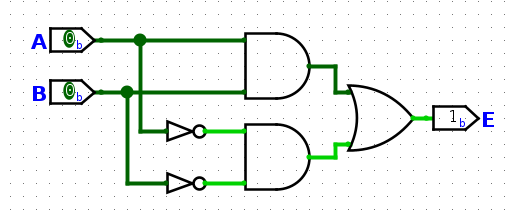
0 0 1

1 0 0

0 1 0

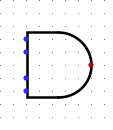
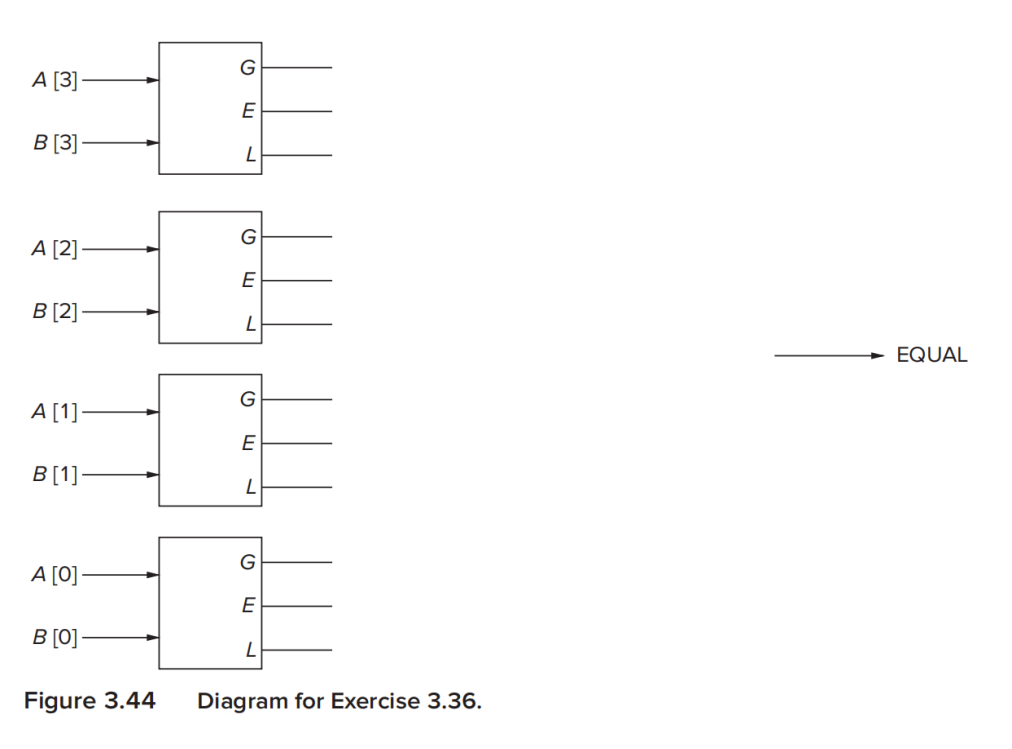
1. Implement G, E, and L using AND, OR, and NOT gates.

·G = (A) AND (NOT(B)) ·L = (NOT(A)) AND (B)

·E = (A AND B) OR (NOT(A) AND NOT(B))

c. Using the one-bit comparator as a basic building block, construct a

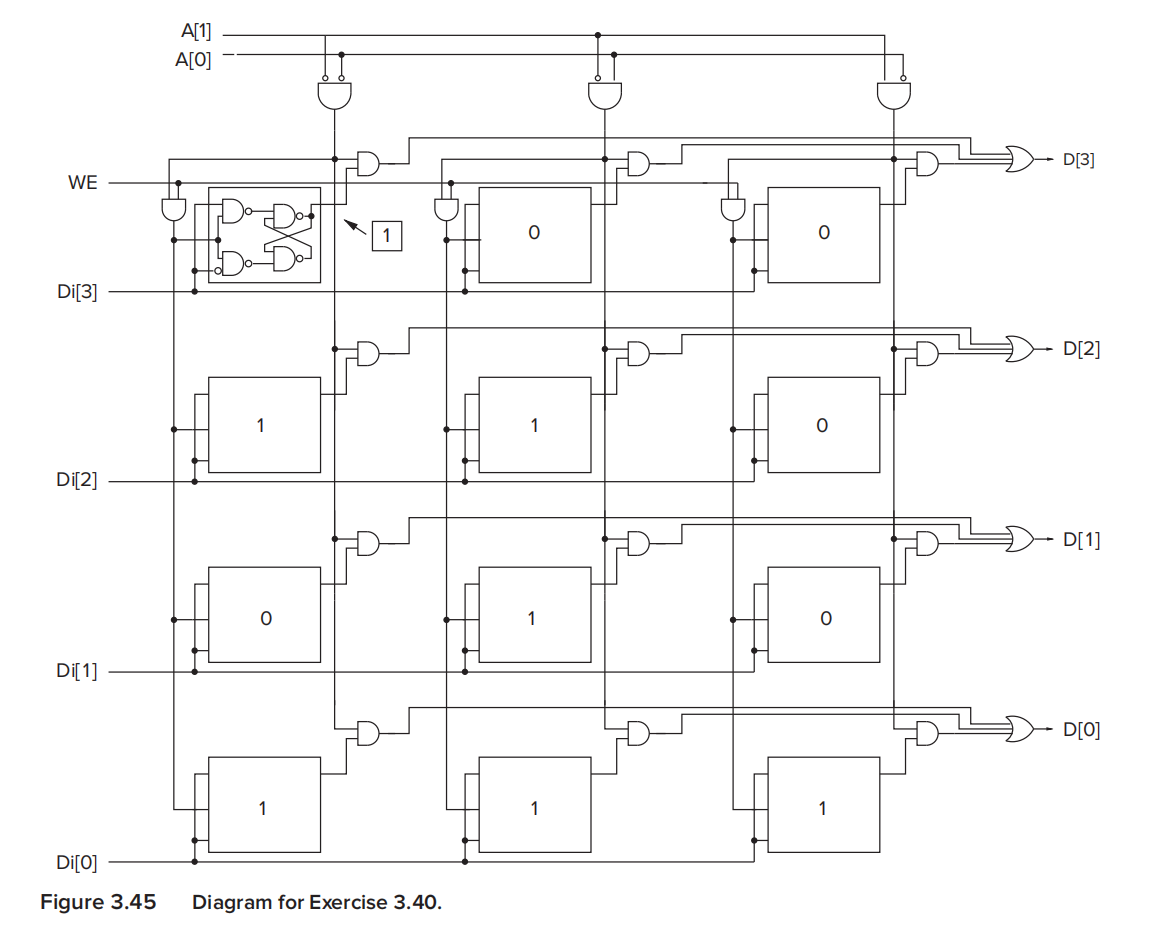
four-bit equality checker.



3.40 For the memory shown in Figure 3.45:

a. What is the address space? b. What is the addressability?

c. What is the data at address 2?



1. The address space is 4 (locations)
2. the addressability is 4 (from D[3] to D[0]).
3. Because the address 2 is 10(B), that means the data in right locations, which is 0001(B)

3.50 Prove that the NAND gate, by itself, is logically complete (see

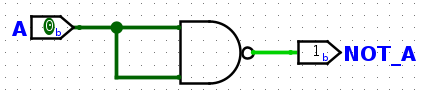
Section 3.3.5) by constructing a logic circuit that performs the AND

function, a logic circuit that performs the NOT function, and a logic

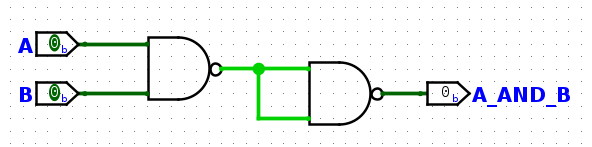
circuit that performs the OR function. Use only NAND gates in these

three logic circuits.

1. Connecting Vcc(logical value is TRUE) into one input of the NAND gate to get the NOT function.



1. Connecting the NAND gate output to the just obtained NOT gate(a) to get the AND function(A AND B = C).



Connecting the two input terminals to the NOT gate(a) to get the OR function(A OR B = C).

