Homework 01

和泳毅 PB19010450

1. (Adapted from problem 1.5 in the textbook)

   Say we had a "black box," which takes two numbers as input and outputs their sum. See Figure 1.10a in the Textbook or the following figure. Say we had another box capable of multiplying two numbers together. See Figure 1.10b. We can connect these boxes together to calculate p * (m + n). See Figure 1.10c. Assume we have an unlimited number of these boxes. Show how to connect them together to calculate:

   a. ax+b

   b. The average of the four input numbers w, x, y, and z

   c. $a^2 + 2ab + b^2$ (can you do it with one add box and one multiply box?)

   d. a^6 (can you do it using only 3 multiply boxes?)

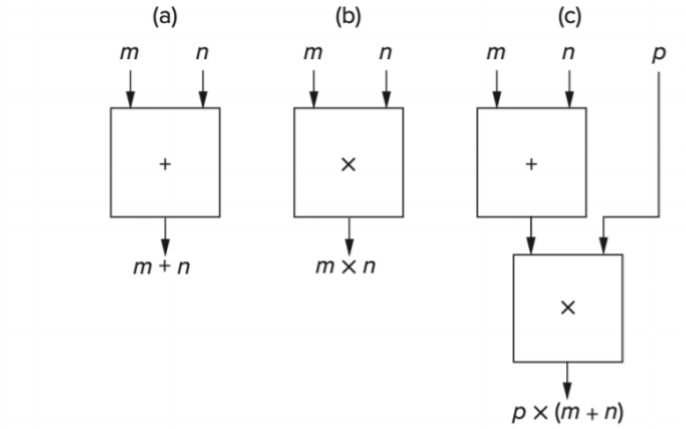
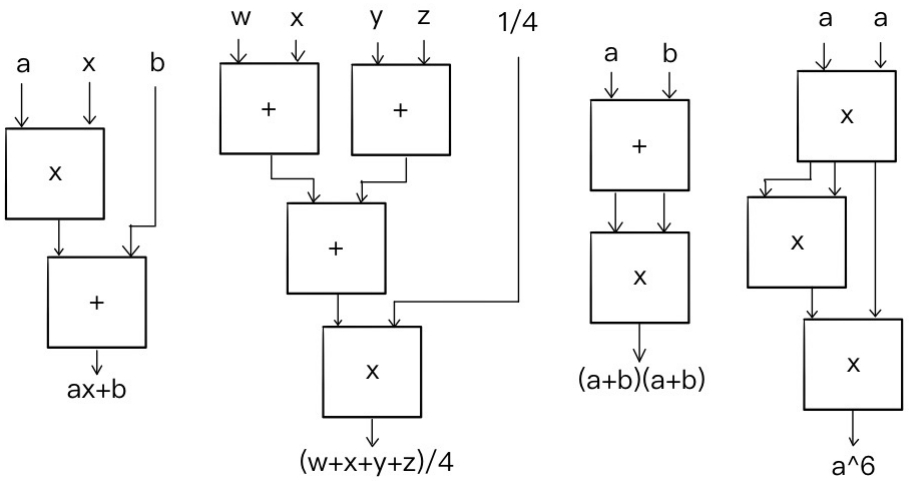


Figure 1.10 **"Black boxes" capable of (a) addition, (b) multiplication, and (c) a combination of addition and multiplication.**

Answer



---

2. (1.14)

Suppose we wish to put a set of names in alphabetical order. We call the act of doing so sorting. One algorithm that can accomplish that is called the bubble sort. We could then program our bubble sort algorithm in C, and compile the C program to execute on an x86 ISA. The x86 ISA can be implemented with an Intel Core microarchitecture. Let us call the sequence "Bubble Sort, C program, x86 ISA, Core microarchitecture" one transformation process.

Assume we have available four sorting algorithms and can program in C, C++, Pascal, Fortran, and COBOL. We have available compilers that can translate from each of theses to either x86 or SPARC, and we have available three different microarchitectures for x86 and three different microarchitectures for SPARC.

a. How many transformation processes are possible?

b. Write three examples of transformation processes.

c. How many transformation processes are possible if instead of three different microarchitectures for x86 and three different microarchitectures for SPARC, there were two for x86 and four for SPARC.

$4 \times 5 \times (3 + 3) = 120$

$4 \times 5 \times (2 + 4) = 120$

Answer a: 120 transformation processes are possible.

b: e.g.1: Sort algorithms Ⅰ, C, x86 ISA, Microarchitectures for x86;

e.g.2: Sort algorithms Ⅱ, Pascal, SPARC ISA, Microarchitectures for SPARC;

e.g.3: Sort algorithms Ⅲ, COBOL, SPARC ISA, Microarchitectures for SPARC.

c: 120 transformation processes are possible.

---

3. (2.3)

a. Assume that there are about 400 students in your class. If every student is to be assigned a unique bit pattern, what is the minimum number of bits required to do this?

b. How many more students can be admitted to the class without requiring additional bits for each student's unique bit pattern?

$2^8 = 256 < 400 < 512 = 2^9$

$2^9 - 400 = 112$

Answer a: The minimum number of bits is 9.

b: 112 students can be admitted.

---

4. (Adapted from 2.13)

Without changing their values, convert the following 2's complement binary numbers into 8-bit 2's complement numbers.

a. 010110

b. 1101

c. 1111111000

d. 01

```
010110       ->      00010110
1101         ->      11111101
1111111000   ->      11111000
01           ->      00000001
```

Answer a: 00010110 b: 11111101 c: 11111000 d: 00000001

---

5. (Adapted from 2.17)

   Compute the following. Assume each operand is a 2's complement binary number.

   a. 01 + 1011

   b. 11 + 01010101

   c. 0101 + 110

   d. 01 + 10

```
 1011       01010101       0101       01
+0001      +11111111      +1110      +10
_____      _________      ______     ____
 1100       01010100       0011       11
```

Answer a : 1100 b: 01010100 c: 0011 d: 11

---

6. Convert the following 8-bit 2's complement binary numbers into decimal numbers.

   a. 01010101

   b. 10001101

   c. 10000000

```
01010101    ->      01010101    ->      01010101    ->      85
10001101    ->      10001100    ->      11110011    ->      -115
10000000    ->      11111111    ->      10000000    ->      0
11111111    ->      11111110    ->      10000001    ->      -1
```

Answer a: 85 b: -115 c: 0 d: -1

---

7. Express the value 0.3 in the 32-bit floating point format that we discussed in class today. Feel free to only show fraction bits [22:15], rather than all the fraction bits, [22:0]. Notation: The symbol [22:15] signifies all 8 bits from bit 22 to bit 15.

First,we express 0.3 as a binary number:$0.0100110011\ldots$.(the recurring number is 0011).
Then we normalize the value, yielding $1.00110011\ldots \times 2^{-2}$.
The sign bit is 0,reflecting the fact that 0.3 is a positive number. The exponent field contains 01111101, the unsigned number 125, reflecting the fact that the real exponent is $-2(125 - 127 = -2)$. After removing the leading 1, the fraction bits [22:15] is 00110011.

```
S = 0    fraction = 00110011    exponent = 01111101
```

Answer 0 01111101 00110011

---

8. Convert the following floating point representation to its decimal equivalent:

   1 10000010 10101001100000000000000

```
S = 1  fraction = 101010011     exponent = 10000010 -> 130
```

$-1.101010011 \times 2^{130-127} \text{—} > -1101.010011 \text{—} > -13.296875$

Answer -13.296875.

---

9. (Adapted from 2.50)

   Perform the following logical operations. Express your answers in hexadecimal notation.

   · xABCD OR x9876

   · x1234 XOR x1234

   · xFEED AND (NOT(xBEEF))

```
xABCD -> 1010101111001101        x9876 -> 1001100001110110
x1234 -> 0001001000110100        xBEEF -> 1011111011101111
xFEED -> 1111111011101101        NOT(xBEEF) -> 0100000100010000

   1010101111001101          0001001000110100        1111111011101101
OR 1001100001110110       XOR 0001001000110100    AND 0100000100010000
————————————————————      ————————————————————    ————————————————————
   1011101111111111          0000000000000000        0100000000000000
   ->xBBFF                   ->x0000                 ->x4000
```

Answer xBBFF x0000 x4000

---

10. (2.54)

    Fill in the truth table for the equations given. The first line is done as an example.

Q_1 = NOT (NOT(X) OR (X AND Y AND Z))

Q_2 = NOT ((Y OR Z) AND (X AND Y AND Z))

X	Y	Z	Q_1	Q_2
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0