ICS hw1

PB18061443 jianghaolin

October 1, 2022

exercise 1. (textbook1.5) Say we had a "black box", which takes two numbers as input and outputs their sum. See Figure 1.10a. Say we had another box capable of multiplying two numbers together. See Figure 1.10b. We can connect these boxes together to calculate $p \times (m + n)$. See Figure 1.10c. Assume we have an unlimited number of these boxes. Show how to connect them together to calculate:

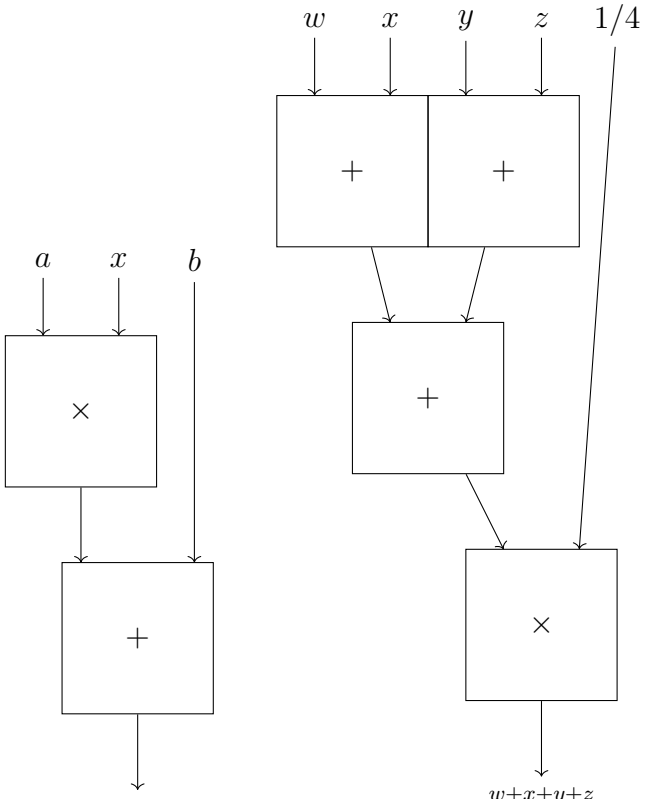
a. $ax + b$

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

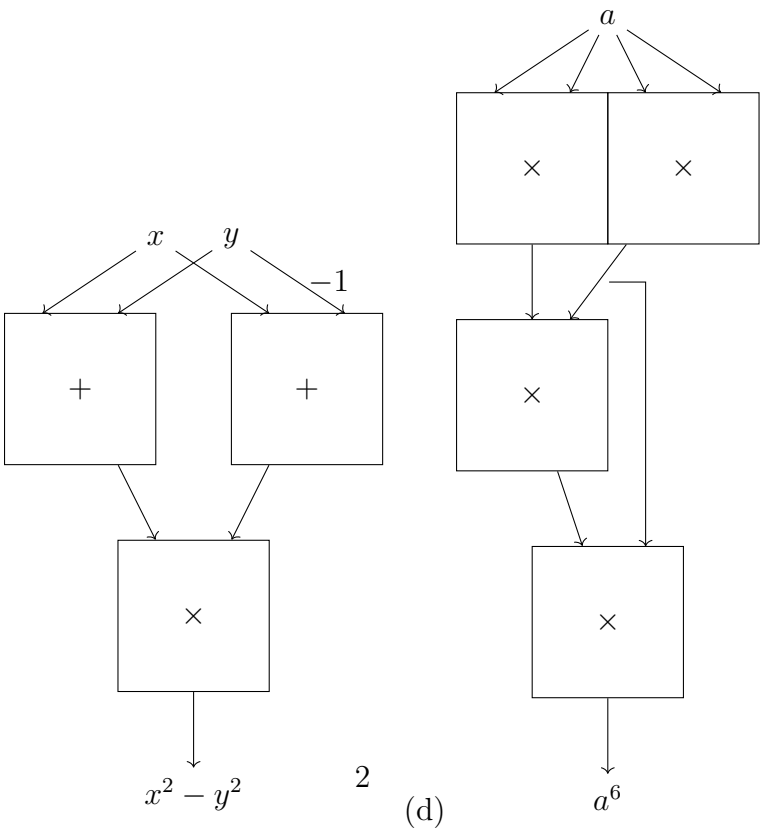
c. $x^2 - y^2$

d. a^6

answer.



(a) (b)



(c) (d)

exercise 2.

1. Convert these decimal numbers to **eight-bit** 2's complement binary numbers.
   a. 98
   b. -105
2. Convert the following **eight-bit** 2's complement binary numbers to decimal.
   a. 01000010
   b. 11101111

answer.

1. a. $98 = (0110\ 0010)_b$
   b. $-105 = (1110\ 1001)_b \Rightarrow (1001\ 0111)_b$
2. a. $0100\ 0010 \Rightarrow 2^6 + 2^1 = 66$
   b. $1110\ 1111 \Rightarrow 1110\ 1110 \Rightarrow 1001\ 0001 \Rightarrow -(2^4 + 2^0) = -17$

exercise 3. Add the following 2's complement binary numbers. Also express the answer in **decimal**.

a. $01 + 110011$

b. $111 + 0100110$

c. $1010 + 1101$

d. $0001 + 1110$

answer.

a.

$$\begin{array}{rr} & 0000\ 0001 \\ + & 1111\ 0011 \\ \hline & 1111\ 0100 \end{array}$$

$(1111\ 0100)_b = -12$

b.

$$\begin{array}{rr} & 1111\ 1111 \\ + & 0010\ 0110 \\ \hline & 0010\ 0101 \end{array}$$

$(0010\ 0101)_b = 37$

c.

$$\begin{array}{rr} & 1010 \\ + & 1101 \\ \hline & 0111 \end{array}$$

$(0111)_b = 7$

d.

$$\begin{array}{rr} & 0001 \\ + & 1110 \\ \hline & 1111 \end{array}$$

$(1111)_b = -1$

exercise 4. **Without changing their values**, convert the following 2's complement binary numbers into **eight-bit** 2's complement numbers.

a. 101011

b. 011110

c. 11111111110000

d. 00001

answer.

a. 1110 1011

b. 0001 1110

c. 1111 0000

d. 0000 0001

exercise 5. Write IEEE floating point representation of $(4.3)_D$. If you cannot represent accurately, please try to represent a number which has the closest difference to $4(.3)_D$ **(Actually you may just write the first ten digits of the fractional part.)**

Can you try to write code in C language to confirm your correctness? (Tips: type float in C language follows the rule of 32bit IEEE floating)

What if the exponent bits are all 1? Search for information about that.

answer.

$$
\begin{aligned}
(4.3)_D &= (100.0100\ \overline{1100})_b \\
&= (1.0001\ \overline{0011})_b \times 2^2 \\
&= (1.0001\ \overline{0011})_b \times 2^{(129-127)} \\
&\Rightarrow 0\ 10000001\ 00010011001100110011001
\end{aligned}
$$

exercise 6. Write the decimal equivalents for the IEEE floating point number below.

0 10001001 11111001101001000000000

answer. 0 1000100 11111001101001000000000

$S = 0, exponent = (10001001)_b = 137$

$S \times (1.11111001101001) \times 2^{(exponent-127)}$

$\Rightarrow (11111100110.1001)_b$

$= 2022.5625$

exercise 7.

(1) Compute the following. Write answers in binary.

a. 10100101 AND 11010101

b. 10001110 OR 11110101

c. NOT(11110001) OR NOT(01011010)

(2) Compute the following. Write answers in hexadecimal.

d. (x1234 AND x5678) OR (xABCD AND x99EF)

d. x6A12 XOR x3A15

answer.

a.

```
        1010 0101
AND     1101 0101
-------------------
        1000 0101
```

b.

```
        1000 1110
OR      1111 0101
-------------------
        1111 1111
```

c.

```
        NOT(1111 0001)
OR      NOT(0101 1010)
-----------------------
              0000 1110
OR            1010 0101
-----------------------
              1010 1111
```

d.

```
                    x1234
AND                 x5678
-------------------------
         0001 0010 0011 0100
AND      0101 0110 0111 1000
-------------------------
         0001 0010 0011 0000
-------------------------
                    x1230
```

e.

	x6A12
XOR	x3A15
	0110 1010 0001 0010
XOR	0011 1010 0001 0101
	0101 0000 0000 0111
	x5007

exercise 8. Fill in the truth table for the equations given

Q_1=(A AND B) OR NOT (C)

Q_2=NOT(NOT(A) OR NOT(B)) AND C

Express Q_2 another way (with only AND)

answer.

A	B	C	A AND B	NOT C	Q_1
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	0	1

A	B	C	NOT A	NOT B	NOT(A) OR NOT(B)	Q_2
0	0	0	1	1	1	0
0	0	1	1	1	1	0
0	1	0	1	0	1	0
0	1	1	1	0	1	0
1	0	0	0	1	1	0
1	0	1	0	1	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

$Q_2 = A \,\text{AND}\, B \,\text{AND}\, C$

exercise 9. Steps to convert ASCII string to base64:

1. Write the ASCII string in bits.
2. Divide each 24 bits into one group. In each group, divide the bits into 4 subgroups and convert each subgroup using the following rule:
   - 0-25 corresponds to A-Z
   - 26-51 corresponds to a-z
   - 52-61 corresponds to 0-9
   - 62 corresponds to + and 63 corresponds to /.
3. If the subgroup is less than 6 bits, add 0 to its lower position.
4. If the last group is 8 bits, add "==" to the end of the encoded string. If the last group is 16 bits, add just one "=".

For example, in order to convert "M", first write it as binary:01001101. The first subgroup is 010011(19), corresponds to "T". The last subgroup is 010000(added 4 zeros), corresponds to "Q"(16). Since the last group is 8 bits, we should add "\=\=" at the end of the encoded string to indicate this. The final base64 encoding is "TQ==".

(1) Convert ASCII string "\t\n\r" to base64.

Hint: '\t', '\n' and '\r' are "escape characters", you can search for information about "escape :characters" online if you don't know where to get the ASCII values of them. You can also try to program in C to help you get the ASCII values.

(2) Give an example where base64 encoding can be used.

answer.

(1.) \t = 009 = $(0000\ 1001)_b$

\n = 010 = $(0000\ 1010)_b$

\r = 013 = $(0000\ 1101)_b$

\t\n\r = 0000 1001 0000 1010 0000 1101

$\Rightarrow$ 000010 010000 101000 001101

$\Rightarrow$ 2 16 40 13

$\Rightarrow$ CQoN

(2.) MIME IRCu UTF-7

exercise 10. What is the largest positive number that can be represented by an IEEE floating point number(Normalized Form)?

answer. 0 11111111 11111111111111111111111

$$
\begin{aligned}
N &= (-1)^0 \times (1.1111\ 1111\ 1111\ 1111\ 1111\ 111)_b \times 2^{(11111111)_b - 127} \\
&= (1.1111\ 1111\ 1111\ 1111\ 1111\ 111)_b \times 2^{128} \\
&= \sum_{k=105}^{128} 2^k \\
&= \frac{2^{105}(1-2^{24})}{(1-2)} \\
&= (2^{24}-1)2^{105}
\end{aligned}
$$

exercise 11. Consider the situation: We want to calculate 0.11×0.022; however, we can only calculate integer numbers. How can we do that?

Obviously we can calculate 11×22 and add dot to the number. However, determining where to put the dot may be difficult. If we can easily transform the value into 1.1×10^{-1} and 2.2×10^{-2}, just calculate 1.1×1.1 and $10^{-1} \times 10^{-2}$ (just add -1 and -2), then put them together.

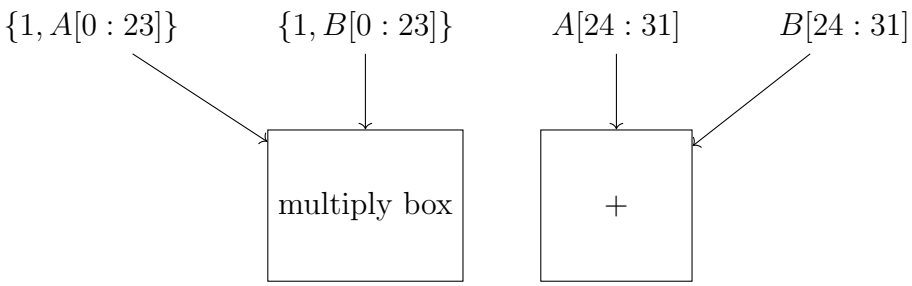
However, 0.95×0.081 makes difference, for $9.5 \times 8.1 = 76.95$, which is greater than 10 . So we may convert 76.95 into 7.695×10^1 and adjust the exponent by increasing 1 . So the answer is $7.695 \times 10^1 \times 10^{-3} = 7.695 \times 10^{-2}$.

Now we have two positive float numbers A and B, they follow the rule of 32 bit IEEE floating point. For simplicity, we can use $A[0]$ as the lowest bit of A, $A[31]$ as the highest, and $A[2:5]$ as array of $A[4], A[3]$ and $A[2], A[5]$ is not included. If we attempt to combine two numbers together, we can use big brackets.

For example: if $A = 0\ 11110000\ 00000000\ 00000000\ 0000111$, then $A[0] = 1, A[31] = 0$, and $A[2:5] = 001, \{101, A[2:5]\} = 101001$, array in order that high bits are on the left.

- We're required to multiply A and B, result as C; please represent one possible algorithm in the format of "Black Box" same as T1.
  - We assume that $1 < \mathrm{A}, \mathrm{B} < 100$;
  - You can use multi boxes to multiply two unsigned integers of 24 bits, and the output is in 48 bits.
  - For example, if input numbers are in two bits, like 10 and 11 , then the output is 0110; and input 11 and 11 make 1001. 24-bit inputs are similar.
  - You can also use add boxes of any bits
  - You can also use selector boxes, which take 3 inputs, a, b, and sel
    - if sel == 0, output = a
    - if sel == 1, output = b
  - Number of boxes you can use is unlimited
- (This problem is not required) How about addition? Also assume $1 < A, B < 100$. You can use Shift Left or Shift Right if you want. You're encouraged to challenge yourself.

answer.


$\{1, A[0:23]\}$
$\{1, B[0:23]\}$
$A[24:31]$
$B[24:31]$
multiply box
+