

Exercise 1 General arithmetics (7 + 3 Marks)
Number conversions

In this exercise you have to convert numbers from one positional number-system to another. A number n of a base b other than 10 will on this page be given as

$$n_b$$

For decimal numbers, the subscript base can be omitted (*i. e.*, we write n instead of “ n_{10} ”). For bases b that are greater than 10, please use uppercase Latin letters (A, B, \dots) as digits beyond 9, just as it is done with hexadecimal numbers.

(a) Convert the numbers $1C_{14}$, $G3_{18}$, and 2214_5 into decimals. Show your workout. (3 Marks)

(b) Convert (4 Marks)

- CB_{24} into a base 12 number
- 1432_6 into a base 9 number

Show your workout.

(c) **Bonus:** Convert 110220120120112112_3 into a base 27 number (3 Marks)

Solution:

(a) The results are:

- $1C_{14} = 1_{14} \times 14^1 + C_{14} \times 14^0 = 1 \times 14 + 12 = 14 + 12 = \mathbf{26}$
- $G3_{18} = G_{18} \times 18^1 + 3_{18} \times 18^0 = 16 \times 18 + 3 = 288 + 3 = \mathbf{291}$
- $2214_5 = 2_5 \times 5^3 + 2_5 \times 5^2 + 1_5 \times 5^1 + 4_5 \times 5^0 = 2 \times 125 + 2 \times 25 + 1 \times 5 + 4 = 250 + 50 + 5 + 4 = \mathbf{309}$

(b) It makes sense to translate to decimal first, and after that to translate into the target system:

- $CB_{24} = 12 \times 24 + 11 = 299$

$$299 \div 12 = 24 \text{ Rem } 11$$

$$24 \div 12 = 2 \text{ Rem } 0$$

$$2 \div 12 = 0 \text{ Rem } 2$$

Thus the result is $20B_{12}$

- $1432_6 = 1 \times 216 + 4 \times 36 + 3 \times 6 + 2 = 380$

$$380 \div 9 = 42 \text{ Rem } 2$$

$$42 \div 9 = 4 \text{ Rem } 6$$

$$4 \div 9 = 0 \text{ Rem } 4$$

Thus the result is 462_9

(c) Since 27 is the third power of 3, the conversion between the bases is simple: we can translate each set of three digits in base 3 separately into one single digit in base 27:

$$\underbrace{110 \ 220 \ 120 \ 120 \ 112 \ 112}_3$$

$C \ O \ F \ F \ E \ E_{27}$

It is the same idea as binary with octal or hexadecimal.

Exercise 2 Binary representations (8 Marks)
Normalized scientific binary floating point

Recall the 16-Bit encoding schema for a normalized scientific binary floating point from Lecture 7:

Sign of mantissa 1 bit	Mantissa 9 bits	Sign of exponent 1 bit	Exponent 5 bits
---------------------------	--------------------	---------------------------	--------------------

Translate the following number into this schema (show your workout):

- (a) 127.9375_{10} (5 Marks)
- (b) If we would like to represent the number 127.9375_{10} in an accurate way (the binary representation corresponds exactly to the number itself), how many bits at least will be needed for the mantissa. Justify your answer. (3 Marks)

Solution:

- (a) Translating 127.9375 into normalized scientific binary floating point. First, translating into binary:

$$\begin{aligned} 127 \div 2 &= 63 \text{ Rem } 1 \\ 63 \div 2 &= 31 \text{ Rem } 1 \\ 31 \div 2 &= 15 \text{ Rem } 1 \\ 15 \div 2 &= 7 \text{ Rem } 1 \\ 7 \div 2 &= 3 \text{ Rem } 1 \\ 3 \div 2 &= 1 \text{ Rem } 1 \\ 1 \div 2 &= 0 \text{ Rem } 1 \end{aligned}$$

$$\begin{aligned} 0.9375 \times 2 &= 1.875 \text{ Rem } 1 \\ 0.875 \times 2 &= 1.75 \text{ Rem } 1 \\ 1.75 \times 2 &= 1.5 \text{ Rem } 1 \\ 0.5 \times 2 &= 1.0 \text{ Rem } 1 \end{aligned}$$

Thus, $127.9375 = 1111111.1111_2$. Normalized, that is $.1111111111 \times 2^7$. Thus, the exponent is 7 or 111_2 . Both mantissa and exponent are positive. The full 16-Bit representation is thus:

0	111111111	0	00111
---	-----------	---	-------

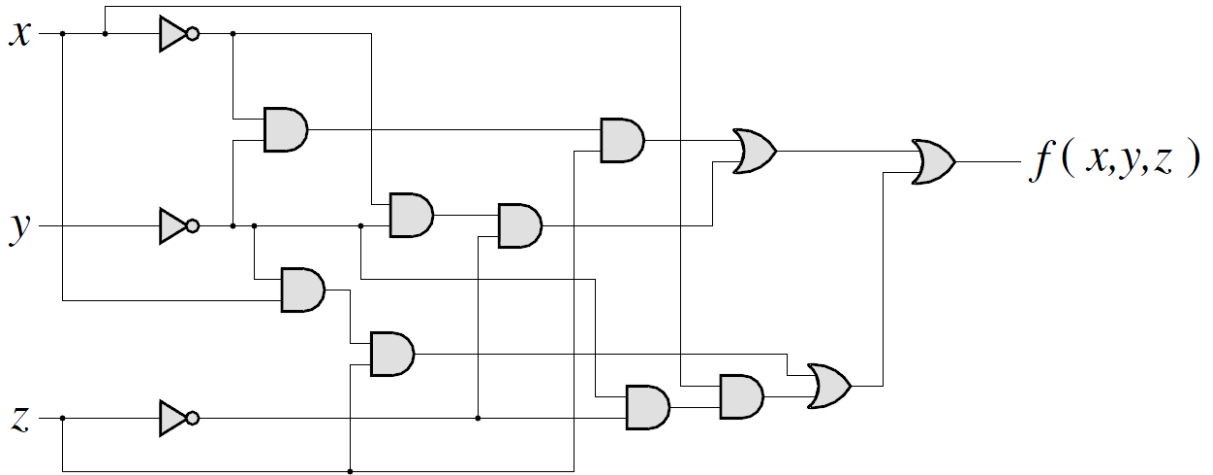
- (b) We would need 11 bits for the mantissa to be able to represent the number in an accurate way.

0	11111111111	0	00111
---	-------------	---	-------

Exercise 3 Boolean functions
Circuit improvement

(9 Marks)

Improve the following circuit:



(a) Write a truth table for f

(4 Marks)

Solution:

x	y	z	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

(b) Extract a boolean expression from the truth table using the sum of products method and simplify it as much as possible. **Note** that the simplified circuit would consist of only one gate. Please mention the applied rules of the Boolean Algebra listed in Exercise 4. (4 Marks)

Solution:

The sum of products algorithm yields the following conjuncts:

$$x'y'z' + x'y'z + xy'z' + xy'z$$

This can be simplified as follows (Commutativity and Associativity used implicitly):


$$\begin{aligned}
 f(x, y, z) &= x'y'z' + x'y'z + xy'z' + xy'z \\
 &= x'y'(z' + z) + xy'(z' + z) && \text{(Distributivity)} \\
 &= x'y' + xy' && (7, 2) \\
 &= (x' + x)y' && \text{(Distributivity)} \\
 &= y' && (7, 2)
 \end{aligned}$$

(c) Draw the simplified (improved) circuit

(1 Mark)

Solution:

x

y —  — $f(x, y, z)$

z

Exercise 4 Boolean Circuits
Functionality and Simplification

(4+4=8 Marks)

(a) Given the following truth table

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>X</i>	<i>Y</i>
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	0	0

What is the functionality of the circuit where *A*, *B*, *C* and *D* are the input variables and *X* and *Y* are the output variables?

Solution:

Given an unsigned number *N* of three bits, the circuit performs the following operation: $N\%3$.

(b) Simplify the following expressions using the axioms of the Boolean Algebra. Please mention the applied rules.

$x + 0 = x$	$x * 1 = x$	
$x + 1 = 1$	$x * 0 = 0$	
$x + x = x$	$x * x = x$	
$x + x' = 1$	$x * x' = 0$	
$(x')' = x$		
$x + y = y + x$	$xy = yx$	Commutativity
$x + (y + z) = (x + y) + z$	$x(yz) = (xy)z$	Associativity
$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$	Distributivity
$(x + y)' = x'y'$	$(xy)' = x' + y'$	DeMorgan's Law

$$A'BCD' + AB'C'D' + AB'C'D + AB'CD' + AB'CD + ABC'D' + ABC'D + ABCD'$$

Note that the resulting expression will consist of only 8 gates.

Solution:

This can be simplified as follows (Commutativity and Associativity used implicitly):

$$\begin{aligned}
 f(A, B, C, D) &= A'BCD' + AB'C'D' + AB'C'D + AB'CD' + AB'CD + ABC'D' + ABC'D + ABCD' \\
 &= AB'C'(D' + D) + AB'C(D' + D) + ABC'(D' + D) + BCD'(A + A') \\
 &\hspace{15em} \text{(Distributivity)} \\
 &= AB'C' + AB'C + ABC' + BCD' \\
 &\hspace{15em} (7, 2) \\
 &= AB'(C' + C) + ABC' + BCD' \\
 &= AB' + ABC' + BCD' \\
 &\hspace{15em} (7, 2) \\
 &= A(B' + BC') + BCD' \\
 &\hspace{15em} \text{(Distributivity)} \\
 &= A((B' + B)(B' + C')) + BCD' \\
 &\hspace{15em} \text{(Distributivity)} \\
 &= A(B' + C') + BCD' \\
 &\hspace{15em} (7, 2)
 \end{aligned}$$

Exercise 5 Boolean functions (22 Marks)
Two's-complementer

In this question you will construct a 3-bit two's-complementer as a circuit. The 3-bit input x will be given as $x_2x_1x_0$. The 3-bit output $-x$ will be called y and given as $y_2y_1y_0$. Recall how a two's complement is obtained.

- (a) Draw a circuit that bit-wise inverts x into \bar{x} (every bit flipped). We call the result $\bar{x} = \bar{x}_2\bar{x}_1\bar{x}_0$. (1 Mark)
- (b) To obtain the two's complement from the intermediate result \bar{x} of part (a), you have to add 1 to \bar{x} . The addition will be implemented for each digit with sum and carry-over. Write a truth table with the input \bar{x}_0 for the first digit y_0 and the first carry-over c_1 . (2 Marks)
- (c) Now write a truth table for y_1 and a carry over c_2 , taking into account \bar{x}_1 and a possible carry over c_1 . Note that the truth table for y_2 is similar. (4 Marks)
- (d) Draw individual circuits for y_0 , c_1 , y_1 , c_2 , and y_2 . (10 Marks)
- (e) Draw the complete 3-bit two's-complementer. (5 Marks)

Solution:

- (a) The bit-wise negation is trivial:



- (b) The truth table for y_0 and c_1 is:

\bar{x}_0	y_0	c_1
0	1	0
1	0	1

- (c) The truth table for y_1 is

\bar{x}_1	c_1	y_1	c_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- (d) The circuits are:

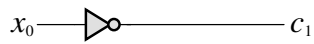
y_0



x_1

x_2

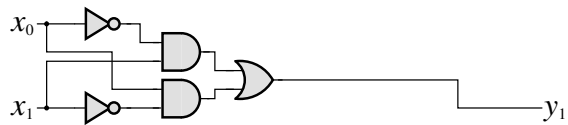
c_1



x_1

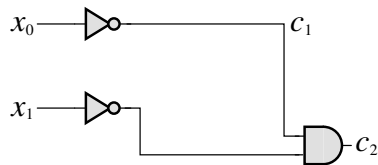
x_2

y_1



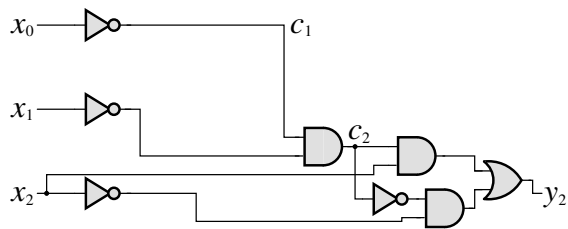
x_2

c_2

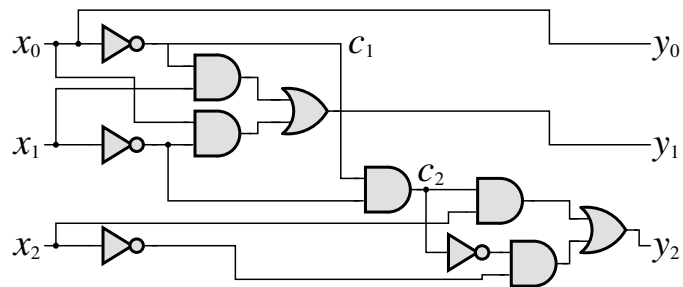


x_2

y_2



(e) The complete 3-bit two's-complementer is therefore



Exercise 6 Boolean Circuits
Comparator

(4+4+4=12 Marks)

A one-bit comparator is a circuit that takes two numbers consisting of one bit each and outputs 1 if the numbers are equal, 0 otherwise.

(a) Construct a truth table for a one bit equality comparator.

Solution:

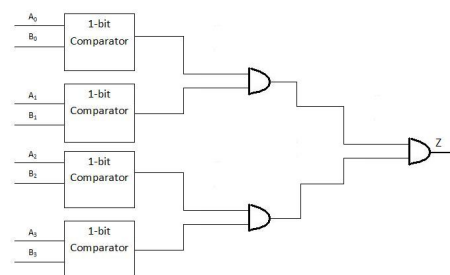
x	y	Output
0	0	1
0	1	0
1	0	0
1	1	1

(b) Assume that you have already manufactured one-bit comparators.

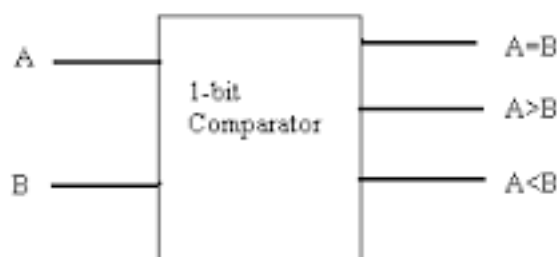


Design a circuit that uses one-bit comparators and AND-gates to check the equality of two numbers consisting of 4 bits each.

Solution:

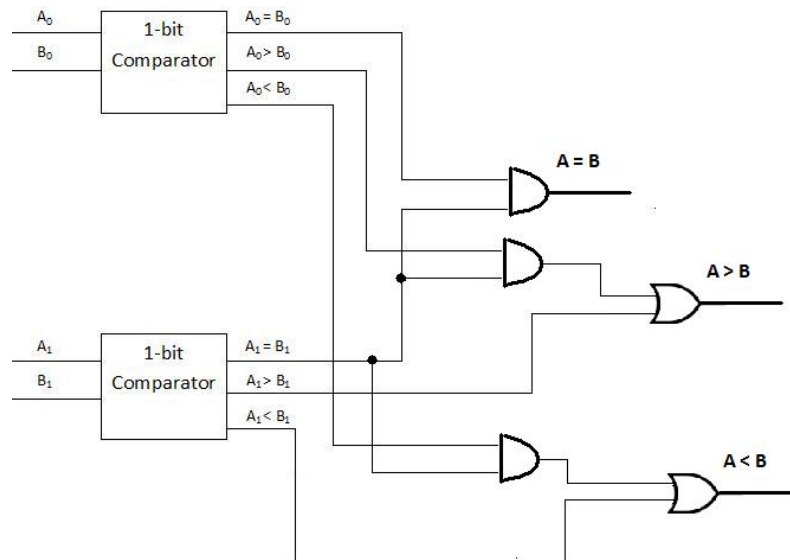


(c) Assume that our one-bit comparator was modified to have two input variables A , B and three output variables (one checking for $A = B$, one checking for $A > B$ and one checking for $A < B$).



Design a circuit that uses the modified one-bit comparators with other gates to compare two numbers consisting of 2 bits each. **Do not draw the truth table.**

Solution:



Exercise 7 Iterative operations
Half life

(6 Marks)

Radioactive substances constantly undergo spontaneous decay. It is impossible to predict the time of decay for a single atom, but for large enough quantities of an element it can be predicted how long it takes until half of the atoms have decayed. The time span it takes for a certain quantity of an element to reduce to half is called the *half-life period*, or short *half life* of that element.

For example, the Frankium isotope F-223 (${}_{87}^{223}\text{F}$) has a half life of 22 minutes. This means that of 1kg of F-223 after 22 minutes only 500g are left. After another 22 minutes (a total of 44 minutes) only 250g are left, another 22 minutes later (a total of 66 minutes) only 125g are left, and so on..

Given a half life in minutes and a quantity in kilogram, calculate after how much time (absolute and in multiples of the half life), less than one kilogram is left.

Solution:

```
1  get halflife, quantity
2
3  set i to 0
4  set time to 0
5
6  while (quantity >= 1) {
7    set quantity to quantity / 2
8    set i to i + 1
9    set time to time + halflife
10 }
11
12 print "After_"
13 print time
14 print "_less_than_1kg_is_left._This_is_"
15 print i
16 print "_times_the_half_life."
```

Exercise 8 Iterative operations (10 Marks)
Chinese chessboard

Legend tells of a peasant in China, who once went to the emperor to ask for some rice. To make his request sound small, he brought a chessboard and asked for the following:

“Put just one grain of rice on the first square. On every following square, put double as many grains of rice as on the previous square. This is all I want”

- (a) Write a program to figure out how many grains of rice the peasant receives, if the chessboard has n squares. (5 Marks)

Solution:

```

1  get n
2
3  set rice to 1
4  set result to 1
5  set i to 2
6
7  while (i <= n) {
8    set rice to rice + rice
9    set result to result + rice
10   set i to i + 1
11 }
12
13 print result

```

- (b) How many grains of rice does the peasant get with a chessboard of 64 squares? Justify your answer. (5 Marks)

Solution:

The chessboard will contain

$$2^0 + 2^1 + 2^2 + \dots + 2^{n-1}$$

grains. The total number is therefore equivalent to the largest n -bit binary number:

$$2^0 + \dots + 2^{n-1} = 2^n - 1$$

With an 8×8 chessboard (*i. e.*, $n = 64$) this is $2^{64} - 1$ (or 18446744073709551615)

Exercise 9 Algorithms and efficiency
Election results

(13 + 5 Marks)

You need a graphical display for presenting the results of an election.

- (a) Given a total number of votes v , a number n of inputs, a list P_1, \dots, P_n of political parties, and a list Q_1, \dots, Q_n of the number of respective votes for each party, output a set of “bars” that illustrate the percentage of votes that each party received together with the name of the party and the percentage in numbers. Parties that did not reach at least 5% should *not* be displayed.

For each percentage point of a party’s votes please output *one* symbol. The first symbol should be an opening bracket (“[”), subsequent symbols should be equality-symbols (“=”), and the last symbol should be a closing bracket (“]”). (12 Marks)

Example. (from Citizen Council elections in Hamburg, Germany, 2/20/2011)

Input:

$v = 3\,443\,699$, $n = 6$,
 $P = \{ "CDU", "DIE_LINKE", "FDP", "GRUENE-GAL", "PIRATEN", "SPD" \}$,
 $Q = \{ 754\,316, 220\,995, 229\,109, 384\,421, 72\,868, 1\,667\,644 \}$

Output:

```
[=====] CDU 21%
[=====] DIE LINKE 6%
[=====] FDP 6%
[=====] GRUENE-GAL 11%
[=====] SPD 48%
```

Solution:

The following algorithm presents a possible solution:

```
1  get v, n, P1, ..., Pn, Q1, ..., Qn
2
3  set i to 0
4  while (i < n) {
5      set i to i + 1
6
7      set percent to (Qi / v) * 100
8
9      if (percent >= 5) then
10         print "["
11         set j to 1
12         while (j <= percent) {
13             set j to j + 1
14             print "="
15         }
16         print "]"
17         print Qi
18         print "]"
19         print percent
20         print "%\n"
21     endif
22 }
```

- (b) Give the exact number of times the first condition in your program is evaluated. Justify your answer. (1 Mark)

Solution:

The condition in line 4 is evaluated $n + 1$ times.

- (c) Give the order of magnitude for the running time of your program for the worst-case scenario. Carefully justify your answer. You may assume correct input data. (5 Marks)

Solution:

Given the input n , the body of the loop in lines 4–22 is executed n times. The body, however, contains a second loop. This second loop is executed as many times as given by the percentage value. The percentage value, in turn, never exceeds 100. Therefore, the running time of the inner loop is constant (< 100). The asymptotic running time of the whole program is therefore $O(n)$

Explanation in detail. Lines 1 and 3 are executed once. The condition in line 4 is evaluated $n + 1$ times. The body of the loop is then executed n times. This loop spans over the single instructions in lines 5 and 7, which are each executed once per iteration, and the conditional in lines 9–21. The condition in line 9 is therefore evaluated once in every iteration, the body of the conditional at most once (actually, since the condition branches depending on a 5% hurdle, the body can in worst case only be executed 20 times, but this will not be relevant). In the body of the conditional there are 7 primitive statements (executed at most once per iteration) and an inner loop (lines 12-15). This inner loop is executed in worst case $\text{percentage} + 1$ times (evaluation of the condition) and percentage times (body, which contains two statements) for every iteration of the outer loop. We therefore have a sum of roughly

$$2 + \underbrace{(n + 1)}_{\text{condition}} + n(2 + 1 + 7 + \underbrace{(100 + 1)}_{\text{condition}} + 100 + 100) = 3 + 312n,$$

which lies in the order of magnitude of $O(n)$. A more precise scrutiny yields smaller constants, but no difference in the order of magnitude.