German University in Cairo
Faculty of Media Engineering and Technology
Prof. Dr. Slim Abdennadher
Dr. Fatma Meawad

February 02, 2013

Introduction to Computer Science
Winter term 2012-2013

Bar Code

Instructions: Read carefully before proceeding.

1) Duration of the exam: 3 hours (180 minutes).
2) (Non-programmable) Calculators are allowed.
3) No books or other aids are permitted for this test.
4) This exam booklet contains 14 pages, including this one. Three extra sheets of scratch paper are attached and have to be kept attached. **Note that if one or more pages are missing, you will lose their points. Thus, you must check that your exam booklet is complete**.
5) Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem or on the three extra sheets and make an arrow indicating that. **Scratch sheets will not be graded unless an arrow on the problem page indicates that the solution extends to the scratch sheets**.
6) When you are told that time is up, stop working on the test.

Good Luck!

Don't write anything below ;-)

Exercise	1	2	3	4	5	6	7	8	9	10	11	$\sum$
Possible Marks	10	12	14	6	8	6	12	12	10	8	8	106
Final Marks												

Exercise 1 (10 Marks)

Given a floating point number, we would like to represent it in binary. A floating point number consists of an integer part and a decimal part.

In this exercise, you are asked to write an algorithm that represents the decimal part into binary representation and stores the result into a list.

You should assume that the precision depends also on the user input. The precision is given in terms of number of bits for the decimal part. However, if the decimal part can be represented with less bits than the given precision, then the representation should consist of a least number of bits.

The algorithm takes as an input a floating point number and the precision of the decimal part. Your task is convert the decimal part in binary and save it in a list.

The following is a sample run of the program for a floating number 5.056 and a precision of 6.

```
The number is 5.096
The decimal part is 0.096
The precision of the decimal part is 6
The representation of the decimal part in binary is: 000110
```

The following is a sample run of the program for a floating number 10.25 and a precision of 6.

```
The number is 10.25
The decimal part is 0.25
The precision of the decimal part is 6
The representation of the decimal part in binary is: 01
```

Solution:

```
get n, prec
set decimal to n - floor(n)
set i to 1

while(i <= prec and dec <> 1.0)
{
   set decimal to decimal * 2
   if(decimal >= 1.0) then
      set Ai to 1
      set decimal to decimal - 1
   else
      set Ai to 0
   endif
   print Ai
   set i to i + 1
}
```

Exercise 2 (6+2+4=12 Marks)

Given the following algorithm:

```
get n
get a[1]a[2]...a[n]
set i to 2
while(i <= n) {
   if (a[i] < a[i-1])
   then set tmp to a[i]
        set a[i] to a[i-1]
        set a[i-1] to tmp
        set i to 1
   endif
   set i to i + 1
}
```

a) What is the output of the algorithm for the following input

```
12 3 15 4
```

Use a tracing table to trace the `while` loop.

Solution:

n	i	a[i]	a[i-1]	tmp
4	2	3	12	3
4	2	12	3	3
4	3	15	3	3
4	4	15	4	4
4	2	12	3	4
4	3	12	4	4
4	2	4	3	4
4	3	12	4	4
4	4	15	12	4

Output of the algorithm:

Solution:

3 4 12 15

b) What is the output of the algorithm for any list?

Solution:

A list sorted ascendingly

c) What is the best case scenario and the worse case scenario for the algorithm above? Justify your answer.

Solution:

- Best Case: If the list is already sorted ascendingly, so the if condition won't be executed
- Worst Case: If the list is sorted descendingly, so each time the condition will be true.

Exercise 3 (4+2+2+4+2=14 Marks)

Given then following algorithm:

```
get x
set i to 1
set y to 1
while (y <= x) {
            set A[i] to y
            print A[i]
            set y to y + 2
            set i  to i + 1
}
```

a) What is the output of the algorithm for `x=7`

Use a tracing table to trace the `while` loop.

Solution:

i	x	y	Ai
1	7	1	1
2	7	3	3
3	7	5	5
4	7	7	7
5	7	9	

Output of the algorithm:

Solution:

1 3 5 7

b) What is the output of the algorithm for any input?

Solution:

Print all the odd numbers from 1 to the given `x`.

c) What is the size of the generated list for any input `x`?

Solution:

$$INT(x+1)/2$$

d) Find the total number of executed operations. Show your workout.

```
get x
set i to 1 ----------------------> 1 operation --> executed once
set y to 1  --------------------->
while (y <= x) {  ---------------->
    set A[i] to y ------->
            print A[i] ---------->
    set y to y + 2 ------>
    set i  to i + 1 ----->
}
```

Total number of operations:

Solution:

$$5 \times ceil(x/2) + 3$$

e) Determine the order of magnitude of the algorithm.

Solution:

$$\mathrm{O(x)}$$

Exercise 4 (2+2+2=6 Marks)

a) Convert the binary number 0011001_2 which is in two's complement to a decimal number (base 10). Show your workout.

Solution:

$$(0011001)_2 = 2^0 + 2^3 + 2^4 = 1 + 8 + 16 = 25$$

b) Convert the decimal number 233_{10} to a number in base 7. Show your workout.

Solution:

$$233_{10} = 452_7$$

Division	Quotient	Remainder
233/7	33	2
33/7	4	5
4/7	0	4

c) Convert the hexadecimal number $AB14_{16}$ to a number in base 8. Show your workout.

Solution:

$$AB14_{16} = 1010101100010100_2$$

The number in groups of three bits:

$$125424_8$$

Exercise 5 (3+2+3=8 Marks)

We would like to store the floating-point number -35.45 in a computer that uses 16 bits to represent real numbers. (10 for the mantissa and 6 for the exponent, both including the sign bit). Show your work as indicated below.

a) Show the binary representation of the decimal number -35.45.

Solution:

$-35.45_{10} = -100011.011_2$

$$\begin{aligned} 0.45 * 2 &= \underline{0}.9 \\ 0.9 * 2 &= \underline{1}.8 \\ 0.8 * 2 &= \underline{1}.6 \end{aligned}$$

b) Show the binary number in normalized scientific notation.

Solution:

$$-35.45_{10} = -100011.011_2 = -.100011011 \times 2^6$$

c) Show how the binary number will be stored in the 16 bits below.

Solution:

1	100011011	0	00110
Sign of mantissa 1 bit	Mantissa 9 bits	Sign of exponent 1 bit	Exponent 5 bits

Exercise 6 (6 Marks)

Assume that our computer stores decimal numbers using 6 bits. Perform the subtraction

$$(-29)_{10} - (15)_{10}$$

using 2's complement notation. Give the result of the subtraction in decimal. Show your workout, i.e. all steps performed.

Solution:

- $29_{10} = 011101_2$
- one's complement of -29 is 100010
- Two's complement of -29 is 100011
- $15_{10} = 001111_2$
- one's complement of -15 is 110000
- Two's complement of -29 is 110001
- $100011 + 110001 = \underline{1}010100$
- Discard the final carry: 010100
- The decimal value of 010100 is 20

$$\begin{aligned} -29 - 10 &= 100011 + 110001 \\ &= \underline{1}010100 \\ &= 20_{10} \end{aligned}$$

Exercise 7 (4+4+4=12 Marks)

Given the following truth table, where **A** and **B** are the input variables and **X**, **Y**, **Z**, and **W** are the output variables.

A	**B**	**X**	**Y**	**Z**	**W**
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

a) Use the sum-of-products algorithm to find the Boolean expressions that describe the output of the truth table.

Solution:

$$\begin{aligned} X &= AB \\ Y &= AB' \\ Z &= A'B \\ W &= A'B' \end{aligned}$$

b) What is the functionality of the circuit?

Solution:

The circuit computes the operation 2^n, where n consists of two bits.

c) Draw the Boolean circuit.

Exercise 8 (2+4+6=12 Marks)

Given a number consisting of three bits representing numbers in 2's complement, your task is to construct a truth table to convert it into one's complement.

a) How many input variables do you need?

Solution:

3

b) How many output variables do you need? Justify your answer.

Solution:

The range of numbers in 2's complment is [-4,3]. To represent -4 in 1's complement, we need at least four bits. Therefore 4 output variables will be needed for the circuit.

c) Construct the truth table

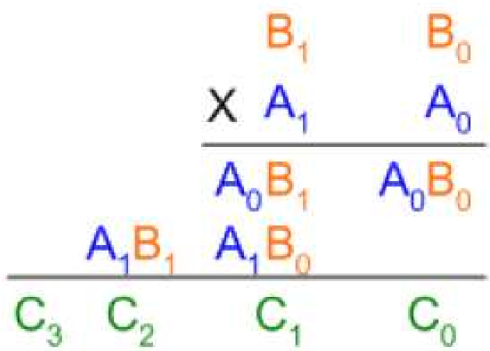
Solution:

A1	**A2**	**A3**	**O1**	**O2**	**O3**	**O4**
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	1	0
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	0	1	1	1	0	0
1	1	0	1	1	0	1
1	1	1	1	1	1	0

Exercise 9 (5+5=10 Marks)

We would like to design a circuit that multiplies numbers A and B.

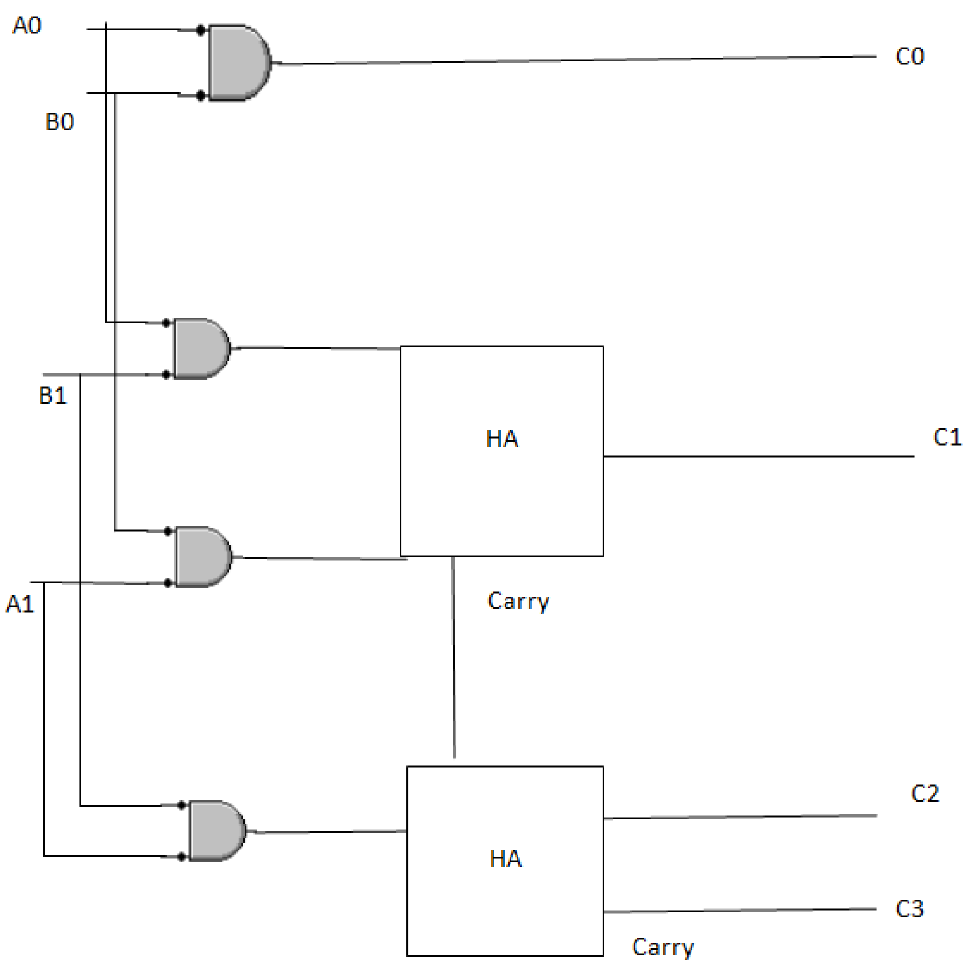
a) We would like to start with a 2-bit multiplier. Assume that A is represented as A_1A_0 and B as B_1B_0. the multiplication is performed as follows:



The first partial product is formed by multiplying the B_1B_0 by A_0. The multiplication of two bits such as A_0 and B_0 produces a 1 if both bits are 1; otherwise it produces a 0. The second partial product is formed by multiplying the B_1B_0 by A_1 and is shifted one position to the left. Then the partial products are added.

Using only half-adders and AND gates, design a 2-bit multiplier.

Solution:



b) Assume we would like to design a binary multiplier with more bits.

Consider multiplying two numbers, A (3-bit number) and B (4-bit number). How many AND gates, half adders and 4-bit adders will be needed. **Note:** A 4-bit adder is a circuit that adds four bits. Justify you answer.

Solution:

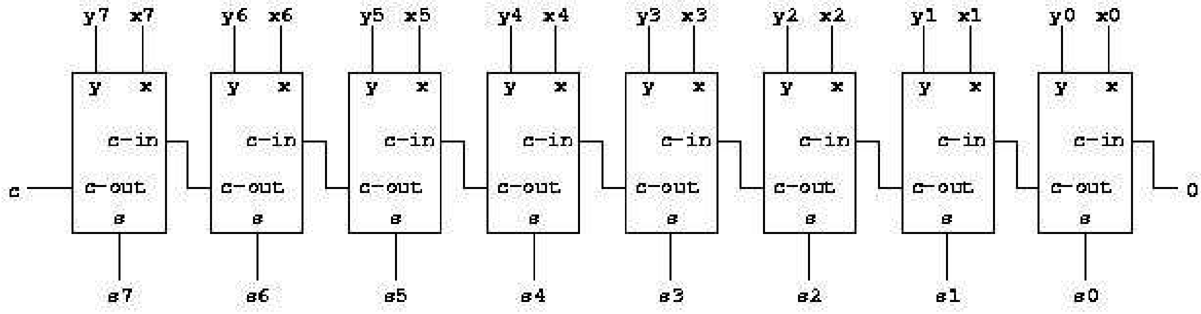
Multiplying $A \times B$ will give:

$$
\begin{array}{ccccccc}
 & & & & A_2 & A_1 & A_0 \\
 & & \times & B_3 & B_2 & B_1 & B_0 \\
\hline
 & & & A_0B_3 & A_0B_2 & A_0B_1 & A_0B_0 \\
 & + & A_1B_3 & A_1B_2 & A_1B_1 & A_1B_0 & \\
+ & A_2B_3 & A_2B_2 & A_2B_1 & A_2B_0 & & \\
\hline
C_6 & C_5 & C_4 & C_3 & C_2 & C_1 & C_0
\end{array}
$$

Thus, we need 12 AND gates and three 4-bit adders to get the C_2, C_3 and $C_4$and 2 half-adders to get the C_1 and C_5.

Exercise 10 (3+5=8 Marks)

Given the following circuit,



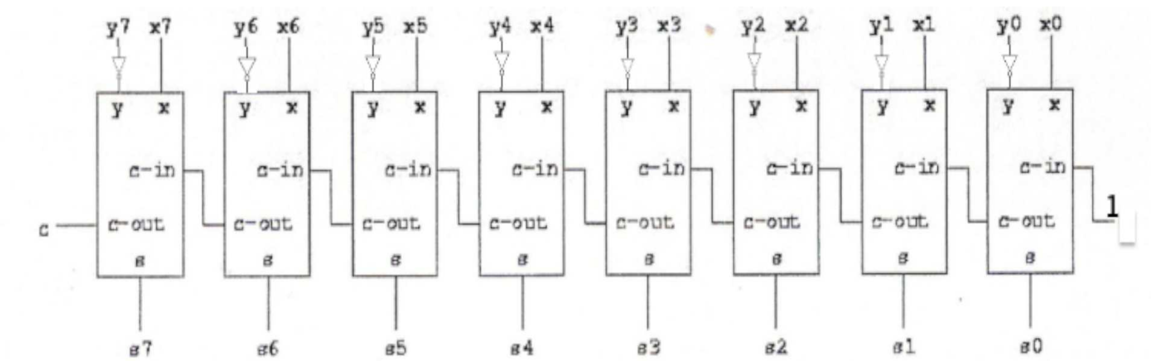
a) What does it do, if the circuits used are full-adders.

Solution:

Adds two numbers X and Y, each consisting of 8 bits.

b) Change the circuit above, in order that the new circuit will perform subtraction in 2'complement.

Solution:



Exercise 11 (8 Marks)

Given the following Boolean expression

$$w'xy'z + w'xyz + w'xyz' + wxy'z + wxyz + wxyz' + w(xy)'z + wx'yz$$

Simplify the Boolean expressions using 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

Hint: The circuit of the simplified expression consists of only four gates.

Solution:

$$
\begin{array}{rl}
w'xy'z + wxy'z + w'xyz' + w'xyz + wxyz + wxyz' + w(xy)'z + wx'yz & (Commutativity) \\
xy'zw' + xy'zw + w'xyz' + w'xyz + wxyz + wxyz' + w(xy)'z + wx'yz & (Commutativity) \\
xy'z(w' + w) + w'xyz' + w'xyz + wxyz + wxyz' + w(xy)'z + wx'yz & (Distributivity) \\
xy'z(w + w') + w'xyz' + w'xyz + wxyz + wxyz' + w(xy)'z + wx'yz & (Commutativity) \\
xy'z * 1 + w'xyz' + w'xyz + wxyz + wxyz' + w(xy)'z + wx'yz & (x + x' = 1) \\
xy'z + w'xyz' + w'xyz + wxyz + wxyz' + w(xy)'z + wx'yz & (x * 1 = x) \\
xy'z + w'xyz' + xyzw' + xyzw + wxyz' + w(xy)'z + wx'yz & (Commutativity) \\
xy'z + w'xyz' + xyz(w' + w) + wxyz' + w(xy)'z + wx'yz & (Commutativity) \\
xy'z + w'xyz' + xyz(w + w') + wxyz' + w(xy)'z + wx'yz & (Commutativity) \\
xy'z + w'xyz' + xyz * 1 + wxyz' + w(xy)'z + wx'yz & (x + x' = 1) \\
xy'z + w'xyz' + xyz + wxyz' + w(xy)'z + wx'yz & (x * 1 = x) \\
xy'z + w'xyz' + wxyz' + xyz + w(xy)'z + wx'yz & (Commutativity) \\
xy'z + xyz'w' + xyz'w + xyz + w(xy)'z + wx'yz & (Commutativity) \\
xy'z + xyz'(w' + w) + xyz + w(xy)'z + wx'yz & (Distributivity) \\
xy'z + xyz'(w + w') + xyz + w(xy)'z + wx'yz & (Commutativity) \\
xy'z + xyz' * 1 + xyz + w(xy)'z + wx'yz & (x + x' = 1) \\
xy'z + xyz' + xyz + w(xy)'z + wx'yz & (x * 1 = x) \\
xy'z + xy(z' + z) + w(xy)'z + wx'yz & (Distributivity) \\
xy'z + xy(z + z') + w(xy)'z + wx'yz & (Commutativity) \\
xy'z + xy * 1 + w(xy)'z + wx'yz & (x + x' = 1) \\
xy'z + xy + w(xy)'z + wx'yz & (x * 1 = x) \\
xy'z + xy + wz(xy)' + wx'yz & (Commutativity) \\
xy'z + xy + wz(xy)' + wzx'y & (Commutativity) \\
xy'z + xy + wz((xy)' + x'y) & (Distributivity) \\
xy'z + xy + wz((x' + y') + x'y) & ((xy)' = x' + y') \\
xy'z + xy + wz(x' + y' + x'y) & \\
xy'z + xy + wz(x' + (y' + x')(y' + y)) & (Distributivity) \\
xy'z + xy + wz(x' + (y' + x')(y + y')) & (Commutativity)
\end{array}
$$

$$
\begin{aligned}
xy'z + xy + wz(x' + (y' + x') * 1) & \quad (x + x' = 1) \\
xy'z + xy + wz(x' + (y' + x')) & \quad (x * 1 = x) \\
xy'z + xy + wz(x' + y' + x') & \\
xy'z + xy + wz(x' + y') & \quad (x + x = x) \\
x(y'z + y) + wz(x' + y') & \quad (Distributivity) \\
x(y + y'z) + wz(x' + y') & \quad (Commutativity) \\
x((y + y')(y + z)) + wz(x' + y') & \quad (Distributivity) \\
x(1 * (y + z)) + wz(x' + y') & \quad (x + x' = 1) \\
x(y + z) + wz(x' + y') & \quad (x * 1 = x) \\
xy + xz + wz(x' + y')' & \quad (Distributivity) \\
xy + zx + zwx' + zwy' & \quad (Distributivity) \\
xy + zx + z(wx' + wy') & \quad (Distributivity) \\
xy + z(x + (wx' + wy')) & \quad (Distributivity) \\
xy + z(x + wx' + wy') & \\
xy + z((x + w)(x + x') + wy') & \quad (Distributivity) \\
xy + z((x + w) * 1 + wy') & \quad (x + x' = 1) \\
xy + z(x + w + wy') & \quad (x * 1 = x) \\
xy + z(x + w(1 + y')) & \quad (Distributivity) \\
xy + z(x + w * 1) & \quad (x + 1 = 1) \\
xy + z(x + w) & \quad (x * 1 = x) \\
xy + zx + zw & \quad (Distributivity) \\
xy + xz + zw & \quad (Commutativity) \\
x(y + z) + zw & \quad (Distributivity)
\end{aligned}
$$

Extra Page

Extra Page

Extra Page