German University in Cairo 14. Feb 2008
Computer Science and Engineering Department
Prof. Dr. Slim Abdennadher

Introduction to Computer Science, WS 2007-2008

FinalTerm Exam

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	$\sum$
Possible Marks	6	6	6	6	10	10	6	5	10	10	75
Final Marks											

Exercise 1 (6 Marks)

a) Convert the binary number 1101011_2 to a decimal number (base 10). Show your workout.

Solution:

$2^6 + 2^5 + 2^3 + 2^1 + 2^0 = 64 + 32 + 8 + 2 + 1 = 107_{10}$

b) Convert the decimal number 122_{10} to a number in base 7. Show your workout.

Solution:

Division	Quotient	Remainder
122/7	17	3
117/7	2	3
2/7	0	2

$122_{10} = 233_7$

c) Convert the hexadecimal number $AF10B_{16}$ to a binary number. Show your workout.

Solution:

A	F	1	0	B
1010	1111	0001	0000	1011

Thus $AF10B_8 = 10101111000100001011_2$

Exercise 2 (6 Marks)

Show how the decimal number 6.969 is stored 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 6.969.

Solution:

$6.969_{10} = 110.111110_2$

$$\begin{aligned} 0.969 * 2 &= \underline{1}.938 \\ 0.938 * 2 &= \underline{1}.876 \\ 0.876 * 2 &= \underline{1}.752 \\ 0.752 * 2 &= \underline{1}.504 \\ 0.504 * 2 &= \underline{1}.008 \\ 0.008 * 2 &= \underline{0}.016 \end{aligned}$$

b) Show the binary number in normalized scientific notation.

Solution:

$6.969_{10} = 110.111110_2 = .110111110 \times 2^3$

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

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

Solution:

0	110111110	0	00011
Sign of mantissa 1 bit	Mantissa 9 bits	Sign of exponent 1 bit	Exponent 5 bits

Exercise 3 (6 Marks)

a) How many bits **at least** will be needed to store the unsigned number 31. Justify your answer.

Solution:

5 bits since $31_{10} = 11111_2$

b) How many bits at least to store the number -31 in sign-magnitude notation? Justify your answer.

Solution:

6 bits since the range for signed numbers in sign-magnitude notation: $[-(2^{n-1}-1), 2^{n-1}-1]$ and $-31 = -(2^5 - 1)$.

c) How many bits at least to store the number -32 in two's complement notation? Justify your answer.

Solution:

6 bits since the range for signed numbers in two's complement notation: $[-2^{n-1}, 2^{n-1}-1]$ and $-32 = -(2^5)$

Exercise 4 (6 Marks)

Assume that our computer stores decimal numbers using 8 bits. Perform the subtraction

$$(-28)_{10} - (5)_{10}$$

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

Solution:

- Convert 28 and 5 to binary:

  $28_{10} = 00011100_2$
  $5_{10} = 00000101_2$

- Two's complement representation of -28

  $-28_{10} = 11100100$

- Two's complement representation of -5

  $-5_{10} = 11111011$

- Perform the addition $(-28)_{10} - (5)_{10}$ in binary:

  $11100100 + 11111011 = 111011111$

- Remove the overflow:

  11011111

- The binary number 11000011 represents the negative decimal value -33.

  The corresponding positive number of 11011111 is 00100001 which corresponds to 33.

Exercise 5 (10 Marks)

Given the following the following truth table, where **A**, **B** and **C** are the input variables and **X** is the output variable.

A	**B**	**C**	**X**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

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

Solution:

$$X = AB'C' + AB'C + ABC' + ABC$$

b) What is the functionality of the circuit? You should express it in form of an arithmetic operation.

Solution:

Assuming that the input variables correspond to a decimal number N from 0 until 7, the circuit will perform the following operation, where the division is an integer division:

$$\frac{N}{4}$$

Another possible solution could be: The circuit checks whether

$$N \geq 4$$

c) Draw the Boolean circuit. **Note** that each gate can have only two inputs.

Exercise 6 (10 Marks)

A circuit should be designed to decrement a decimal number having a range from 0 until 7. The result of the subtraction should be represented in sign-magnitude.

a) How many input and output variables are needed?

Solution:

We need 3 input variables and 4 output variables.

If the input will be also represented in sign-magnitude, then 4 input variables will be needed.

b) Construct the truth table for this circuit

Solution:

A	**B**	**C**	**S**	**O1**	**O2**	**O3**
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	0	1	1
1	0	1	0	1	0	0
1	1	0	0	1	0	1
1	1	1	0	1	1	0

c) Use the sum-of-products algorithm to find the Boolean expressions that correspond to the the truth table.

Solution:

$$
\begin{aligned}
S &= A'B'C' \\
O1 &= AB'C + ABC' + ABC \\
O2 &= A'BC + AB'C' + ABC \\
O3 &= A'B'C' + A'BC' + AB'C' + ABC'
\end{aligned}
$$

Exercise 7 (6 Marks)

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

a) The simplified expression consists of two gates.
$(A + B) * (A + C) =$

Solution:

$$\begin{aligned}(A + B) * (A + C) = &\quad \textit{Distributivity}\\ A + BC &\end{aligned}$$

b) The simplified expression consists of three gates.
$A + A' * B + A' * B' * C + A' * B' * C' * D$

Solution:

$$\begin{aligned}
A + A' * B + A' * B' * C + A' * B' * C' * D &= \\
(A + A') * (A + B) + A' * B' * C + A' * B' * C' * D &= && [x + yz = (x + y)(x + z)]\\
1 * (A + B) + A' * B' * C + A' * B' * C' * D &= && [x + x' = 1]\\
(A + B) * 1 + A' * B' * C + A' * B' * C' * D &= && [Commutativity]\\
A + B + A' * B' * C + A' * B' * C' * D &= && [x * 1 = x]\\
A + B + B'A'C + A' * B' * C' * D &= && [Commutativity]\\
A + (B + B') * (B + A'C) + A' * B' * C' * D &= && [x + yz = (x + y)(x + z)]\\
A + 1 * (B + A'C) + A' * B' * C' * D &= && [x + x' = 1]\\
A + (B + A'C) * 1 + A' * B' * C' * D &= && [Commutativity]\\
A + B + A'C + A' * B' * C' * D &= && [x * 1 = x]\\
A + A'C + B + A' * B' * C' * D &= && [Commutativity]\\
(A + A') * (A + C) + B + A' * B' * C' * D &= && [x + yz = (x + y)(x + z)]\\
1 * (A + C) + B + A' * B' * C' * D &= && [x + x' = 1]\\
(A + C) * 1 + B + A' * B' * C' * D &= && [Commutativity]\\
A + C + B + A' * B' * C' * D &= && [x * 1 = x]\\
A + C + B + B'A'C'D &= && [Commutativity]\\
A + C + (B + B') * (B + A'C'D) &= && [x + yz = (x + y)(x + z)]\\
A + C + 1 * (B + A'C'D) &= && [x + x' = 1]\\
A + C + (B + A'C'D) * 1 &= && [Commutativity]\\
A + C + B + A'C'D &= && [x * 1 = x]\\
C + B + A + A'C'D &= && [Commutativity]\\
C + B + (A + A') * (A + C'D) &= && [x + yz = (x + y)(x + z)]\\
C + B + 1 * (A + C'D) &= && [x + x' = 1]
\end{aligned}$$

$$\begin{aligned}
C + B + (A + C'D) * 1 = &\quad [Commutativity] \\
C + B + A + C'D = &\quad [x * 1 = x] \\
B + A + C + C'D = &\quad [Commutativity] \\
B + A + (C + C') * (C + D) = &\quad [x + yz = (x + y)(x + z)] \\
B + A + 1 * (C + D) = &\quad [x + x' = 1] \\
B + A + (C + D) * 1 = &\quad [Commutativity] \\
B + A + C + D &\quad [x * 1 = x]
\end{aligned}$$

Exercise 8 (5 Marks)

Prove the following

$$(A * (B' * C' + B * C))' = A' + (B + C) * (B' + C')$$

Hint: There are different ways to prove that two Boolean expressions are equivalent. In this case, it would be easier to use the Boolean algebra rules given in Exercise 7.

Solution:

$$\begin{aligned}
(A * (B' * C' + B * C))' &= \\
A' + (B' * C' + B * C)' &= & [(xy)' = x' + y'] \\
A' + ((B'C')' * (BC)') &= & [(xy)' = x' + y'] \\
A' + ((B'' + C'') * (B' + C') &= & [(xy)' = x' + y'] \\
A' + ((B + C) * (B' + C')) &= & [(x')' = x] \\
A' + (B + C) * (B' + C') &
\end{aligned}$$

Exercise 9 (10 Marks)

Write an algorithm that takes as a parameter a list containing a sequence of integers and outputs a list with its mirror image. The mirror image of a sequence contains the original sequence in reversed order followed by the original sequence. For example, if the original sequence is:

```
1, 8, 15, 7, 2
```

then the algorithm should create a list of the form

```
2, 7, 15, 8, 1, 1, 8, 15, 7, 2
```

Solution:

```
get n
get A1,A2,..., An
set i to n
set j to 1

while (i>=1)
{
   set Bj to Ai
   set Bj+n to An-i+1
   set i to i-1
   set j to j+1
}

set j to 1

while (j<=2n)
{
 print Bj
 set j to j+1
 }
```

Exercise 10 (10 Marks)

The following algorithm calculates the binary representation of given number `x` and represent it in `n` bits.

```
get x
get n
set i to n - 1
while (i >= 0) do
   {
      if (x-2^i) < 0
      then
         set Bi to 0
      else
         set Bi to 1;
         set x to x-2^i
      endif
      set i to i-1
   }
set i to i + 1
while(i < n)
    {
       print Bi
       set i to i + 1
    }
```

Please note that `2^i` corresponds to the power operation 2^i.

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

Solution:

```
get x
get n
set i to n - 1      ------------ 1 operation --> executed once
while (i >= 0) do   ------------ 1 operation --> executed(n+1) times
{
     if (x-2^i) < 0  ------------1 operation ---> executed n times
     then
        set Bi to 0
     else
        set Bi to 1  ------------- 1 operation --> executed n times
        set x to x-2^i ----------- 1 operation --> executed n times
     endif
     set i to i-1  --------------- 1 operation --> executed n times
}
set i to i + 1 --------------- 1 operation --> executed once

while(i < n) ------------------ 1 operation --> executed (n+1) times
{
      print Bi --------------- 1 operation --> executed n times
      set i to i + 1  --------------- 1 operation --> executed n times
}
```

Total number of executed operations: = 8n + 4

b) Give the value of x so that the algorithm will execute the least number of operations. Justify your answer.

Solution:

For x = 0, the algorithm will execute the least number of operations, because in each execution of the loop the `then` part of the if-statement consisting of only one statement will be executed.

c) Determine the order of magnitude of the algorithm.

Solution:

O(n)

Extra Page

Extra Page

Extra Page