Name: Matriculation Number:

Midterm Exam
General CS II (320201)

March 22, 2010

You have two hours(sharp) for the test;
Write the solutions to the sheet.

The estimated time for solving this exam is 10 minutes, leaving you 65 minutes for revising your exam.

You can reach 7 points if you solve all problems. You will only need 56 points for a perfect score, i.e. -49 points are bonus points.

Different problems test different skills and knowledge, so do not get stuck on one problem.

	To be used for grading, do not write here		
prob.	1.1	Sum	grade
total	7	7	
reached			

Please consider the following rules; otherwise you may lose points:

- Always justify your statements. Unless you are explicitly allowed to, do not just answer "yes" or "no", but instead prove your statement or refer to an appropriate definition or theorem from the lecture.
- If you write program code, give comments!

1 Graphs and Trees

Problem 1.1 (Squirrel search) 70min

Two squirrels live in a forest with trees positioned as below (**Fig.1**, routes between trees are marked). The nerd squirrel, starting from the marked tree, must find the dumb squirrel lost in the far away trees of the dark side of the forest. When searching, the squirrel always moves to the left, but it may end up in the wrong subforest while in the dark side (it has two alternatives). When reaching the wrong side of the forest, it realizes it should go back and use the other path.(see **Fig. 2**)

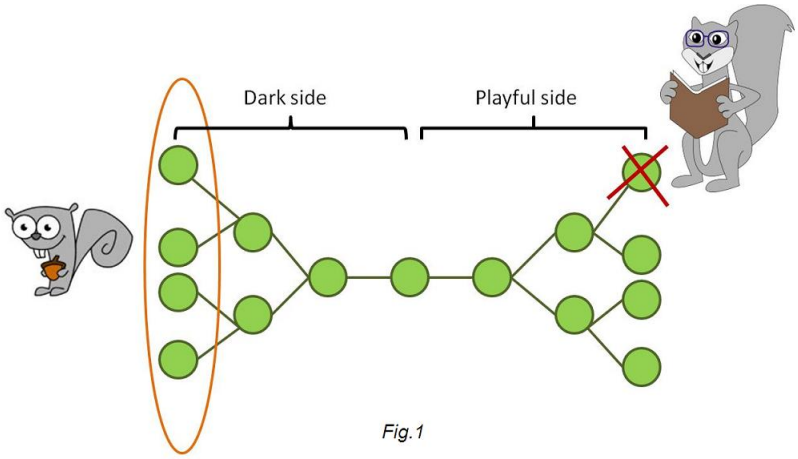


Fig.1

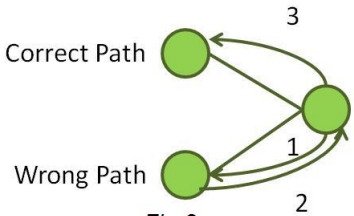


Fig.2

Your tasks are the following:

1. For the situation above write down the minimal and maximal number of trees that need to be explored to find the naughty squirrel, wherever in the far away trees of the dark side of the forest it is.

2. Since we have 4 trees in each of the outskirts of the forest, call this a 4-forest. Now imagine an n-forest and try to find again the minimal and maximal number of trees that need to be explored to find the squirrel.

Solution: **Remark**: The problem is very easy if we evaluate everything by considering fully balanced binary trees, for which we know several facts, depth lemma for example.

1. Minimal: 7 Maximal: 9

2. Now consider the playful side for an n-forest. The wise squirrel will always traverse $\lfloor \log_2(2n-1) \rfloor + 1$ according to the depth lemma and the fact that any binary tree with n leaves has $2n-1$ vertices. The $+1$ comes from the fact that the root of the tree is also to be explored. To traverse the middle section in order to get to the dark side it needs to explore one more tree.

   Minimal: With optimal decisions, the dark side will be handled like the playful side, so we get the final formula $1 + 2 * (\lfloor \log_2(2n-1) \rfloor + 1)$.

   Maximal: With bad decisions, at every depth except the first tree in the dark side it makes one mistake and adds + 1. So we will have:

   $1 + \lfloor \log_2(2n-1) \rfloor + 1 + \lfloor \log_2(2n-1) \rfloor + 1 + \lfloor \log_2(2n-1) \rfloor + 1 - 1 = 3 * (\lfloor \log_2(2n-1) \rfloor + 1)$ trees to explore.

2 Combinatorial Circuits

Problem 2.1 (Two's complement conversion) 3pt in

Given the binary number $a = 11001$ compute

1. $\langle\!\langle a \rangle\!\rangle$ (a is an n-bit unsigned binary number)
2. $(\langle\!\langle a \rangle\!\rangle^{-})$ (a is an n-bit signed binary number)
3. $\langle\!\langle a \rangle\!\rangle_n^{2s}$ (a is in TCN representation)

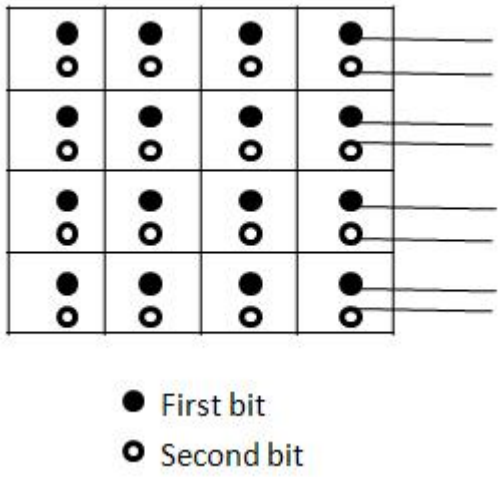
Solution:

1. 25
2. -9
3. -7

Problem 2.2 (Tic Tac Toe) 20pt

One of Gen CS TAs' all time favorite game is Tic Tac Toe. They also love numbers that are powers of 2 so they play on a 4 by 4 board. They built a circuit like in the picture below, where each cell has a 2 bit output:

- 00 if the cell is empty
- 10 if there is an X in the cell
- 01 if there is an O in the cell



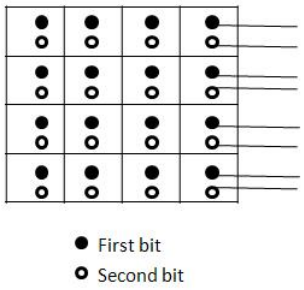
Of course, initially all cells are empty, and TAs take turns at putting Xs or Os on the board. They would like to have a circuit (read "You must design a circuit"), with two outputs, A and B, such that:

- if X won (there are 4 Xs in one row, column or diagonal), A signals 1 and B signals 0
- if O won (there are 4 Os in one row, column or diagonal), A signals 0 and B signals 1
- if no one won yet, both A and B signal 0

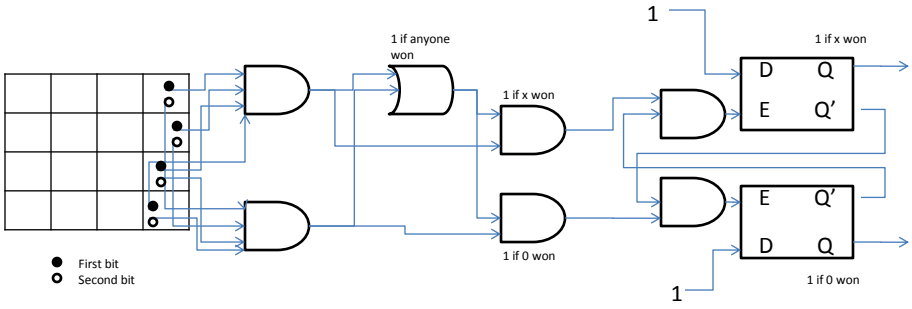
Once someone formed a line, column or row (someone won), placing Xs and Os on the remaining cells on the board should not change the output of A and B anymore.

Note: You will only receive full points if this condition is implemented in your circuit.

Hint: Draw a circuit that checks one line or column and explain in a sentance how to combine all other outputs of lines, columns and diagonals for the overall result.



Solution:



Repeat this for each row, column and diagonal, and then OR the results of all flip flops for 1 and all flip flops for 0

The signals for X or O winning are ORed before putting them in the AND before the flip flop.

3 Machine Programming

Problem 3.1 (Sum of factorials) 8pt in

Let $D(0)$ contain a natural number $n > 1$.

Assume an Assembler operation `MULI` i that changes the value in ACC to $ACC * D(i)$.

Write a short Assembler program that stores the value of $\sum_{i=1}^{n} i!$ in $D(1)$.

Solution:

label	instruction	comment
	`LOADI` 0, `STORE` 1	D(1) = 0 (sum)
	`LOADI` 1, `STORE` 2	D(2) = 1 (index)
	`LOADI` 1, `STORE` 3	D(3) = 1 (factorial)
⟨*loop*⟩	`LOAD` 0	
	`SUB` 2	
	`JUMP`$_{\leq}$ ⟨*end*⟩	
	`LOAD` 2, `ADDI` 1, `STORE` 2	index = index+1
	`LOAD` 3, `MULI` 2, `STORE` 3	factorial = factorial*index
	`LOAD` 1, `ADD` 3, `STORE` 1	sum = sum + factorial
	`JUMP` ⟨*loop*⟩	
⟨*end*⟩	`STOP` 0	

Problem 3.2 (Gratest common divisor in $\mathcal{L}$(VM)) 8pt in

Assume stack is initiated with 2 numbers a and b. Write a $\mathcal{L}$(VM) program (no procedures) that will write the greatest common divisor of a and b on top of the stack. Use Euclidean Algorithm:

```
var a,b
    while b != 0
        t := b
        b := a mod b
        a := t
    return a
```

Solution: `con` (a) `con` (b)
`peek` (0) [*start*]
`peek` (1) `leq` `cjp` (*swap*) ; if b≤a
`peek` (1) `peek` (0) `sub` ; then c = a-b
`con` (0) `peek` (2) `leq` `cjp` (5) ; if c≤0 (i.e. c = 0, since can't be negative)
`peek` (1) `halt` ; then return c
`poke` (0) `jp` (*start*) ; else a = c = a-b
`peek` (0) [*swap*]
`peek` (1) `poke` (0) `poke` (1) ; else swap a and b and try again
`jp` (*start*)
`halt`

Problem 3.3 (Static procedure) 8ptin

Write a VM program using static procedures that simulates the function given by the pseudocode below.

```
fun f(a,b) =
     if(b=<0)
     then 0
     else if (a=a*b)
         then a
         else f(a*a,b-1)
in
     f(5,2)
end
```

Solution:

```
proc 2 36
con 0 arg 2 leq cjp 5
con 0 return
con 1 arg 2 leq cjp 5
con 1 return
con 1 arg 2 sub arg 1 arg 1 mul
call 0 return
con 2 con 5 call 0
halt
```

Problem 3.4 (Power 2) 10pt in

You are given the alphabet $\{0, 1, \_\}$, where "$\_$" symbolizes an empty cell. Consider a left and right infinite tape with a string of 1's as input and all other cells empty. Design a TM that halts in a state "yes" if the input has length 2^n, $n \geq 0$ and in state "no" otherwise. Assume the cursor to be positioned on the first cell filled by the input.

Hint: Try to halve the input until there is only one 1. If this is possible then answer "yes".

Solution:

Missing entries in the transition table can be filled arbitrarily; i.e. they are irrelevant for the solution of the problem.

Old	Read	Write	Move	New	Comment
s_1	0	0	right	s_1	
s_1	1	1	right	s_2	
s_1	$\_$	$\_$	left	s_3	
s_2	0	0	right	s_2	
s_2	1	0	right	s_1	Change every second 1 to 0
s_2	$\_$	$\_$	stop	"no"	If the number of 1's is odd answer "no"
s_3	1	1	left	s_4	
s_3	0	0	left	s_3	
s_4	$\_$	$\_$	stop	"yes"	
s_4	1	1	left	s_5	Check is there is only one more 1 and if so answer "yes"
s_4	0	0	left	s_4	
s_5	$\_$	$\_$	right	s_1	
s_5	1	1	left	s_5	
s_5	0	0	left	s_5	