Todor Nikolov CS 47, Section 01 Kaushik Patra Homework 2

1. Fibonacci

```
# Problem 1.
# Write a program in MIPS assembly that would ask the user for number
# of Fibonacci numbers to be generated and will store it in $s0.
# The program will push that many number of Fibonacci sequence into stack.
# Include complete program.
.data
intro: .asciiz "This_program_will_generate_n_consecutive_Fibonnaci_terms."
prompt: .asciiz "Please_enter_n:_"
.text
.globl
        main
main:
        li
                $v0, 4
        la
                $a0, intro
        syscall
                $a0, intro2
        syscall
                $a0, prompt
        syscall
        \# read n
        li
                $v0, 5
        syscall
        add
                $s0, $v0, $zero
                                        \# s0 = n
                GEN_FIB
        jal
        # exit
                $v0, 10
        l i
        syscall
GEN_FIB:
        # Is n < 1 ?
        addi
                $t0, $zero, 1
                \$t0, \$s0, \$t0
        sge
                $t0, $zero, FIB_END
        beq
        \# push 0
        \mathbf{sw}
                \mathbf{\$zero}, (\mathbf{\$sp})
        addi
                \$\mathbf{sp}, \$\mathbf{sp}, -4
                \$s0, \$s0, -1
        addi
                \$s0, \$zero, FIB_END
        beq
```

```
# push 1
         addi
                   $t1, $zero, 1
         \mathbf{sw}
                    $t1, ($sp)
         addi
                   \$ sp , \$ sp , -4
         addi
                    \$s0, \$s0, -1
                   \$s0, \$zero, FIB_END
         beq
         add
                    \$t0, \$zero, \$zero # t1 is already 1
         add
                    $t0, $zero, $t1
FIB_LOOP:
                   $t2, $t0, $t1
         add
                    $t2, ($sp)
         \mathbf{sw}
         addi
                    \$\mathbf{sp}, \$\mathbf{sp}, -4
         addi
                    \$s0, \$s0, -1
                    soleday, zero, FIB_END
         beq
         add
                    \$t0, \$zero, \$t1
         add
                    $t1, $zero, $t2
         j
                   FIB_LOOP
FIB_END:
                   $ra
```

2. Merge "sort"

```
# Problem 2.
# Write a program which defines total m and n integer numbers in pre-defined
\# data area 'var_a' and 'var_b' in sorted order. The number m and n are stored
# in location var_m and var_n. The program implements a merge sort between
# set_a and set_b and result of the sort (i.e. m+n unsigned integer in ascending
# sorted order) is sorted in another predefined area 'var_c'.
.data
var_a: .word
var_b: .word
bar_c: .word
var_m: .word
var_n: .word
.text
.globl main
        li
                 $s0, var_a
main:
                 \$s1, var_b
                 s_2, var_c
        1 i
        li
                 \$s3, var_m
        li
                 \$s4, var_n
                                          \# n = 0 ?
LOOP:
        beq
                 $s4, $zero, FLUSH_A
                 \$s3, \$zero, FLUSH_B
                                          \# m = 0 ?
        beq
                 $t0, ($s0)
        lw
                 $t1, ($s1)
        lw
        _{
m slt}
                 $t2, $t1, $t0
                 \mathbf{\$t2}, \mathbf{\$zero}, Ab
        beq
```

```
\# b > a
Ba:
                $t0, ($s2)
                                       \# C.push(A.pop())
        \mathbf{sw}
                \$s2, \$s2, 4
                                         # move C pointer
        addi
                \$s3, \$s3, -1
                                         # m-
        addi
        addi
                $s0, $s0, 4
                                         # move A pointer
                LOOP
Ab:
        \# \ a > b
                \$t1, (\$s2)
                                       \# C.push(B.pop())
        \mathbf{sw}
        addi
                $s2, $s2, 4
                                        # move C pointer
        addi
                $s4, $s4, -1
                                         # n---
                $s1, $s1, 4
        addi
                                         # move B pointer
                LOOP
FLUSH_A: # B is empty
                $t0, ($s0)
                                         # A.pop()
        lw
        addi
                $s0, $s0, 4
                                         # move A pointer
                                        # m—
        addi
                \$s3, \$s3, -1
                                       # C.push()
                $t0, ($s2)
                \$s2, \$s2, 4
                                        \# move C pointer
        addi
                $s3, $zero, END
                                         \# if m = 0, end
        beq
                FLUSH_A
FLUSH_B: # A is empty
        lw
                \$t0, (\$s1)
                                         # B.pop()
        addi
                $s1, $s1, 4
                                       # move B pointer
        addi
                $s4, $s4, -1
                                       # n---
                $t0, ($s2)
                                       # C.push()
                \$s2, \$s2, 4
                                        # move C pointer
        addi
                $s4, $zero, END
        beq
                                         \# if n = 0, end
                FLUSH_B
END:
                $v0, 10
        li
        syscall
```

3. Two's complement

a)
$$a = 6_{10} = \boxed{0110_2}$$
 $b = -3_{10} = (-1) \times 0011_2 = 1100 + 0001 = \boxed{1101_2}$ b) $0110_2 \times 1101_2 = 0110 \times 1000 = 0110000 + 0110 \times 0100 = 011000 + 0110 \times 0000 = 00000 + 0110 \times 0001 = 0110$

4. Truth tables. Last column is the answer.

	χ	y	z	хy	(xy)'	(xy)' + z
	0	0	0	0	1	1
	0	0	1	0	1	1
	0	1	0	0	1	1
a)	0	1	1	0	1	1
	1	0	0	0	1	1
	1	0	1	0	1	1
	1	1	0	1	0	0
	1	1	1	1	0	1

	χ	y	z	x'	y'	z'	x'yz'	xy'z	(x'yz') + (xy'z)
	0	0	0	1	1	1	0	0	0
	0	0	1	1	1	0	0	0	0
	0	1	0	1	0	1	1	0	1
b)	0	1	1	1	0	0	0	0	0
	1	0	0	0	1	1	0	0	0
	1	0	1	0	1	0	0	1	1
	1	1	0	0	0	1	0	0	0
	1	1	1	0	0	0	0	0	0

5. Boolean algebra

a) Prove x'y'z' + xy'z' + x'yz' + xyz' = z'

$$x'y'z' + xy'z' + x'yz' + xyz'$$

$$= (x' + x)y'z' + (x' + x)yz' \text{ (distributive)}$$

$$= (1)x'z' + (1)yz' \text{ (complement)}$$

$$= x'z' + yz' \text{ (identity)}$$

$$= (y' + y)z' \text{ (distributive)}$$

$$= (1)z' \text{ (complement)}$$

$$= z' \blacksquare \text{ (identity)}$$

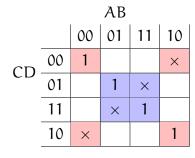
b) Prove (a'+c)(a'+d')(b+c)(b+d') = a'b+cd'

- 6. Karnaugh Maps
- a) Simplify, show "prime-implicants" and "essential prime implicants" $f(A,B,C,D) = \sum m(1,2,3,4,6,7,9,11,12,13,14,15)$

	AB							AB									
		00	01	11	10			00	01	11	10			00	01	11	10
	00		1	1			00		1	1		0	0		1	1	
CD	01	1		1	1		01	1		1	1	0	1	1		1	1
	11	1	1	1	1		11	1	1	1	1	1	1	1	1	1	1
	10	1	1	1			10	1	1	1		1	0	1	1	1	
	A'C + AD					BD' + CD					AB + B'D						

Prime impicants: A'C + AD + BD' + CD + AB + B'DEssential prime implicants: A'C + AD + B'D + BD'

b) Find a minimum SOP expression for: $f(A,B,C,D) = \sum m(0,5,10,15) + d(2,7,8,13)$



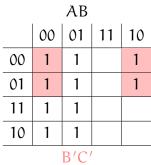
Prime implicants: BD + B'D'

Essential prime implicants: $BD + B^{\prime}D^{\prime}$

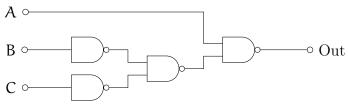
7. NAND-only, single-digit

A B C D	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
В	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
C	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
D	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0

			AВ				
		00	01	11	10		
	00	1	1		1		0
CD	01	1	1		1		0
	11	1	1				1
	10	1	1				10
			A'			,	



 $A'+B'C'=A\ \mathrm{nand}(B'C')'=A\ \mathrm{nand}(B'\ \mathrm{nand}\ C')$



 $\mathsf{D} \mathrel{\circ\!\!-\!\!-}$