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."
intro2: .asciiz "This implementation starts with F(1) = 0 and F(2) = 1\n"
prompt: .asciiz "Please enter n: "

.text
.globl  main
main:
        li      $v0, 4
        la      $a0, intro
        syscall
        la      $a0, intro2
        syscall
        la      $a0, prompt
        syscall

        # read n
        li      $v0, 5
        syscall

        add     $s0, $v0, $zero         # s0 = n
        jal     GEN_FIB

        # exit
        li      $v0, 10
        syscall

GEN_FIB:
        # Is n < 1 ?
        addi    $t0, $zero, 1
        sge     $t0, $s0, $t0
        beq     $t0, $zero, FIB_END

        # push 0
        sw      $zero, ($sp)
        addi    $sp, $sp, -4
        addi    $s0, $s0, -1
        beq     $s0, $zero, FIB_END
```

```
        # push 1
        addi    $t1, $zero, 1
        sw      $t1, ($sp)
        addi    $sp, $sp, -4
        addi    $s0, $s0, -1
        beq     $s0, $zero, FIB_END

        add     $t0, $zero, $zero # t1 is already 1
        add     $t0, $zero, $t1
FIB_LOOP:
        add     $t2, $t0, $t1
        sw      $t2, ($sp)
        addi    $sp, $sp, -4
        addi    $s0, $s0, -1
        beq     $s0, $zero, FIB_END
        add     $t0, $zero, $t1
        add     $t1, $zero, $t2
        j       FIB_LOOP

FIB_END:
        jr      $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
main:   li      $s0, var_a
        li      $s1, var_b
        li      $s2, var_c
        li      $s3, var_m
        li      $s4, var_n

LOOP:   beq     $s4, $zero, FLUSH_A      # n = 0 ?
        beq     $s3, $zero, FLUSH_B      # m = 0 ?
        lw      $t0, ($s0)
        lw      $t1, ($s1)
        slt     $t2, $t1, $t0
        beq     $t2, $zero, Ab
```

```
Ba:     # b > a
        sw      $t0, ($s2)              # C.push(A.pop())
        addi    $s2, $s2, 4             # move C pointer
        addi    $s3, $s3, -1            # m--
        addi    $s0, $s0, 4             # move A pointer
        j       LOOP

Ab:     # a > b
        sw      $t1, ($s2)              # C.push(B.pop())
        addi    $s2, $s2, 4             # move C pointer
        addi    $s4, $s4, -1            # n--
        addi    $s1, $s1, 4             # move B pointer
        j       LOOP

FLUSH_A: # B is empty
        lw      $t0, ($s0)              # A.pop()
        addi    $s0, $s0, 4             # move A pointer
        addi    $s3, $s3, -1            # m--
        sw      $t0, ($s2)              # C.push()
        addi    $s2, $s2, 4             # move C pointer
        beq     $s3, $zero, END         # if m = 0, end
        j       FLUSH_A

FLUSH_B: # A is empty
        lw      $t0, ($s1)              # B.pop()
        addi    $s1, $s1, 4             # move B pointer
        addi    $s4, $s4, -1            # n--
        sw      $t0, ($s2)              # C.push()
        addi    $s2, $s2, 4             # move C pointer
        beq     $s4, $zero, END         # if n = 0, end
        j       FLUSH_B

END:
        li      $v0, 10
        syscall
```

3. Two's complement

a)

$$\begin{aligned} a = 6_{10} &= \boxed{0110_2} \\ b = -3_{10} &= (-1) \times 0011_2 = 1100 + 0001 = \boxed{1101_2} \end{aligned}$$

b)

$$\begin{array}{rcrcr} 0110_2 \times 1101_2 & = & 0110 \times 1000 & = & 0110000 \\ & + & 0110 \times 0100 & = & 011000 \\ & + & 0110 \times 0000 & = & 00000 \\ & + & 0110 \times 0001 & = & 0110 \\ \hline & & & & 01001110 \quad \text{(overflow)} \end{array}$$

c)

	Zero-extension	Sign-extansion
a	00000110	00000110
b	00001101	11111101

4. Truth tables. Last column is the answer.

a)

x	y	z	xy	$(xy)'$	$(xy)'+z$
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	0	1	1
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

b)

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

$$
\begin{aligned}
& x'y'z' + xy'z' + x'yz' + xyz' \\
= \;& (x'+x)y'z' + (x'+x)yz' && \text{(distributive)} \\
= \;& (1)x'z' + (1)yz' && \text{(complement)} \\
= \;& x'z' + yz' && \text{(identity)} \\
= \;& (y'+y)z' && \text{(distributive)} \\
= \;& (1)z' && \text{(complement)} \\
= \;& z' \;\blacksquare && \text{(identity)}
\end{aligned}
$$

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

$$
\begin{aligned}
& (a' + c)(a' + d')(b + c)(b + d') & \\
= \; & (a'a' + a'd' + ca' + cd')(bb + bd' + cb + cd') & \text{(distributive)} \\
= \; & (a' + a'd' + ca' + cd')(b + bd' + cb + cd') & \text{(idempotent)} \\
= \; & (a' + cd')(b + cd') & \text{(covering)} \\
= \; & a'b + a'cd' + cd'b + cd'cd' & \text{(distributive)} \\
= \; & a'b + cd' \;\blacksquare & \text{(covering)}
\end{aligned}
$$

6. Karnaugh Maps

a) Simplify, show "prime-implicants" and "essential prime implicants"

$f(A, B, C, D) = \sum m(1, 2, 3, 4, 6, 7, 9, 11, 12, 13, 14, 15)$

CD \ AB	00	01	11	10
00		1	1	
01	1		1	1
11	1	1	1	1
10	1	1	1	

$A'C + AD$

CD \ AB	00	01	11	10
00		1	1	
01	1		1	1
11	1	1	1	1
10	1	1	1	

$BD' + CD$

CD \ AB	00	01	11	10
00		1	1	
01	1		1	1
11	1	1	1	1
10	1	1	1	

$AB + B'D$

Prime impicants: $A'C + AD + BD' + CD + AB + B'D$

Essential prime implicants: $A'C + AD + B'D + BD'$

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

CD \ AB	00	01	11	10
00	1			×
01		1	×	
11		×	1	
10	×			1

Prime implicants: $BD + B'D'$

Essential prime implicants: $BD + B'D'$

7. NAND-only, single-digit

A	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
B	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

AB

CD	00	01	11	10
00	1	1		1
01	1	1		1
11	1	1		
10	1	1		

A'

AB

CD	00	01	11	10
00	1	1		1
01	1	1		1
11	1	1		
10	1	1		

$B'C'$

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