**Laborator 4 - Arhitectura sistemelor de calcul**

**Proceduri simple, apeluri imbricate, proceduri recursive, proceduri pentru manipularea tablourilor unidimensionale**

1. Sa se defineasca procedura perfect(x), cu x numar natural. Un numar este perfect daca este egal cu suma divizorilor sai pana la jumatate.

Exemplu: 6 = 1 + 2 + 3; 28 = 1 + 2 + 4 + 7 + 14;

**Solutie:**

C/C++:

Int sum = 0;

For (int i = 1; i <= x/2; ++i)

If (x % i == 0)

Sum += i;

.data

x: .word 28

.text

perfect:

subu $sp, 4

sw $fp, 0($sp)

# $sp: ($fp v)(x)

addi $fp, $sp, 4

# $sp: ($fp v)$fp:(x)

subu $sp, 4

sw $s0, 0($sp)

# $sp:($s0 v)($fp v)$fp:(x)

lw $s0, 0($fp)

# verific daca $s0 este perfect i.e. $s0 = suma div. sai pana la jumatate

div $t0, $s0, 2 # $t0 = $s0 / 2 = x / 2

li $t1, 1 # pe post de index in for

li $t2, 0 # pe post de suma

for:

# daca i > x/2 => mergi la exit

# daca $t1 > $t0 => exit

bgt $t1, $t0, exit

rem $t3, $s0, $t1 # $t3 = $s0 % $t1

beqz $t3, edivizor

cont:

addi $t1, 1

j for

edivizor:

add $t2, $t2, $t1

j cont

exit:

seq $v0, $s0, $t2

# daca $s0 == $t2 at. $v0 = 1, altfel $v0 = 0

lw $s0, -8($fp)

lw $fp, -4($fp)

addu $sp, 8

jr $ra

main:

# push x

lw $t0, x

subu $sp, 4

sw $t0, 0($sp)

jal perfect

addu $sp, 4

move $a0, $v0

li $v0, 1

syscall

li $v0, 10

syscall

1. Sa se implementeze un program care sa calculeze functia f(x) = 2g(x), unde g(x) = x+1.

.data

x: .word 5

.text

main:

# push x

lw $t0, x

subu $sp, 4

sw $t0, 0($sp)

jal f

addu $sp, 4

move $a0, $v0

li $v0, 1

syscall

li $v0, 10

syscall

f:

subu $sp, 4

sw $fp, 0($sp)

addi $fp, $sp, 4

# $sp:($fp v)$fp:(x)

subu $sp, 4

sw $ra, 0($sp)

# $sp:($ra v)($fp v)$fp:(x)

subu $sp, 4

sw $s0, 0($sp)

# $sp:($s0 v)($ra v)($fp v)$fp:(x)

lw $s0, 0($fp)

subu $sp, 4

sw $s0, 0($sp)

jal g

addu $sp, 4

mul $v0, $v0, 2 # f(x) = 2g(x), g(x) returnase in $v0

lw $s0, -12($fp)

lw $ra, -8($fp)

lw $fp, -4($fp)

addu $sp, 12

jr $ra

g:

subu $sp, 4

sw $fp, 0($sp)

addi $fp, $sp, 4

# $sp:($fp v)$fp:(x)

subu $sp, 4

sw $s0, 0($sp)

# $sp:($s0 v)($fp v)$fp:(x)

lw $s0, 0($fp)

addi $v0, $s0, 1 # g(x) = x + 1 adica $v0 = $s0 + 1

lw $s0, -8($fp)

lw $fp, -4($fp)

addu $sp, 8

jr $ra

1. Proceduri recursive. Sa se implementeze procedura proc(x), x > 1, cu definitia:

proc(x) = afiseaza x, proc(x-1), daca x != 0

stop, altfel

proc(3) = 321

.data

x: .word 3

.text

main:

# push x

lw $t0, x

subu $sp, 4

sw $t0, 0($sp)

jal proc

addu $sp, 4

li $v0, 10

syscall

proc:

subu $sp, 4

sw $fp, 0($sp)

addi $fp, $sp, 4

subu $sp, 4

sw $ra, 0($sp) # pentru ca fac apeluri imbricate catre proc

subu $sp, 4

sw $s0, 0($sp)

# $sp:($s0 v)($ra v)($fp v)$fp:(x)

lw $s0, 0($fp)

beqz $s0, exit # daca x = 0, atunci oprim procedura

move $a0, $s0

li $v0, 1

syscall

addi $s0, -1

subu $sp, 4

sw $s0, 0($sp)

jal proc

addu $sp, 4

exit:

lw $s0, -12($fp)

lw $ra, -8($fp)

lw $fp, -4($fp)

addu $sp, 12

jr $ra