Tema 1

DESCRIERE REZOLVARE:

1. Voi citi valoarea lui p, dupa care:
   * Daca p<=1, p nu e prim si se afiseaza mesajul corespunzator
   * Daca p>1 si p nu e prim se afiseaza mesajul corespunzator prin procedura e\_prim

Cum functioneaza procedura e\_prim?

Aceasta va primi prin registrul $a0 valoarea lui p, dupa care va parcurge cu un contor, i=$t0, intervalul [2 , |sqrt(p)|], verificand pentru fiecare i daca p%i==0. Daca aceasta conditie este indeplinita se va apela eticheta nr\_nu\_e\_prim care returneaza in $v0 valoarea 0 si se intoarce unde ramasese in main. Daca conditia nu este indeplinita pentru niciun i in acest interval se apeleaza eticheta nu\_3\_prim care returneaza in $v0 valoarea 1 si se intoarce unde ramasese in main. In fucntie de valoarea returnata de functie programul returneaza mesajul corespunzator sau continua.

* + Daca p>1 si p e prim voi continua programul.

1. Daca p e prim, se cauta generatorul:
   * $t0=p, $a1=i, i apartinand intervalului [2, p-1]. Pentru fiecare i verifica daca e generator: pun conditia ca (prima putere la la care il ridic pe i pentru care acesta da 1 mod p trebuie sa fie p1)
   * Pentru a ma folosi de valorile generatorului la fiecare putere, mod p, voi suprascrie pentru fiecare i aceste valori intr-un vector. Ultima suprascriere va fi si cea corecta, atunci cand se gaseste generatorul.
   * Eticheta gasire\_generator are rolul de a initializa toate valorile pentru noul i pe care il verificam (il introduce pe 1 si pe i in vector, salveaza in $t2 valoarea lui i si salveaza in $t3 valoarea 2 pentru a se incepe verificarea de la puterea a 2-a).
   * Eticheta gasire\_generator\_loop are rolul de a il ridica pe i la fiecare putere mod p pana cand ajungem la puterea p-1 sau pana cand restul mod p este 1. La fiecare putere $t2, care retine valoarea lui i la puterea curenta mod p, trece printr-un sir de conditii:
     + Daca $t2=1 se apeleaza eticheta am\_gasit\_rest\_1 care mai departe verifica:
       - Daca puterea curenta este p-1 se apeleaza eticheta am\_gasit\_generatorul
       - Altfel trecem la urmatorul i pentru ca acesta sigur nu e generator (s-a gasit restul 1 prea devreme)
     + Altfel, deci daca restul nu e 1, testam valoarea puterii curente:
       - Daca puterea curenta e p-1 dar restul nu e 1 apelam eticheta nu\_e\_asta\_generatorul
       - Daca nici restul nu e 1 si nici puterea nu e p-1 continuam cu ridicarea la putere a lui i.
   * Daca s-a gasit generatorul acesta se afiseaza si se introduce valoarea 1 in vector.
   * Un generator sigur exista pentru fiecare nr prim p, deci ne vom intoarce inapoi unde ramasesem in main prin apelul jr $ra din am\_gasit\_generatorul.
2. Citesc sirul care trebuie criptat si apelez procedura ciptare:
   * Parcurg sirul citit, iar in $t2 retin elementul curent. Atunci cand voi ajunge la newline ma voi opri cu parcurgerea si voi afisa srul criptat. In $t1 pozitia literei curente in alfabet (indexat de la 0). Am pus coditii ca sa ma sigur ca fiecare litera ce trebuie criptata e in sirul "ABCDEFGHIJKLMNOPQRSTUVWXYZ" si ca indicele literei curente in alfabet (indexat de la 0) nu este mai mare ca p-1. Daca aceste conditii nu sunt indeplinite sa va afisa mesajul “Sirul introdus nu poate fi procesat”. Altfel, inmultesc aceasta valoare cu 4 pentru a putea accesa elementele corespunzatoare din vector. In $t3 voi retine valoarea corespunzatoare literei curente din vector. Din sirul declarat in memorie "ABCDEFGHIJKLMNOPQRSTUVWXYZ" aleg litera cu indicele $t3. Aceasta litera o adaug sirului sir\_criptat. Incrementez $t0 si continui procesul.
   * Dupa ce am criptat tot sirul il afisez prin eticheta afisare\_criptare, apoi revin unde ramasesem in main.
3. Pentru a decripta un sir trebuie mai intai sa am vectorul invers. Pentru aceasta apelez procedura creare\_vector\_invers. Aceasta parcurge vectorul v si pentru fiecare v[i]=x adauga pe pozitia x in vectorul vi valoarea i. Se observa insa ca sunt 2 indici care corespund valorii 1: 0 si p-1. Eu voi alege ca v[1]=1. De asemenea, decriptarea, la fel ca criptarea, nu va functiona pentru literele cu indicele in alfabet mai mare ca p-1 sau care nu sunt caractere din sirul "ABCDEFGHIJKLMNOPQRSTUVWXYZ". De asemenea, avand in vedere ca nu contine elementul 0, nu se va putea decripta litera A.
4. Efectuez decriptarea, care este foarte similara cu criptarea folosind vectorul vi. Dupa ce creez sirul decriptat il afisez prin eticheta afisare\_decriptare si ma intorc unde ramasesem in main prin jr $ra.

CODUL SURSA:

.data

p: .space 4

mesaj: .asciiz "Numarul p nu este prim"

mesaj\_incompatibilitate: "Sirul introdus nu poate fi procesat"

generator: .asziiz "Generatorul g este: "

sp: .asciiz " "

line: .asciiz "\n"

.align 0

sir1: .space 100000

sir2:.space 100000

sir\_criptat: .space 100000

sir\_decriptat: .space 100000

litere: .asciiz "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

.align 4

v: .space 10000

.align 4

vi: .space 10000

.text

main:

li $v0, 5 #citirea lui p

syscall

move $t0, $v0

sw $t0, p

la $a0, sir1 #citesc sirul care trebuie criptat

li $a1, 10000

li $v0, 8

syscall

la $a0, sir2#citesc sirul care trebuie decriptat

li $a1, 10000

li $v0, 8

syscall

move $t0, $a0

ble $t0, 1, afisare\_mesaj #daca p<=1 se afiseaza "Numarul nu este prim"

lw $a0, p #p

jal e\_prim #sarim la procedura e\_prim care il primeste ca parametru pe p prin $a0

#in e\_prim am lucrat cu registrii: $a0=p, $t0=i, $t2=i\*i, $t3=p%i

#nu mai e nevoie de niciuna dintre aceste valori, deci acesti registri vor fi reciclati

move $a0, $v0

beq $a0, $zero, afisare\_mesaj #daca numarul nu e prim afisez mesajul corespunzator

#programul continua doar daca p e prim

lw $t0, p#t0=p

li $a1, 1#a1=i

jal gasire\_generator #caut generatorul

#pentru gasirea generatorului am lucrat cu registi: $t0=p, $a1=i, $t2=i^k mod p,

#$t3=puterea curenta+1, $t4=pozitia pe care am adaugat in vector, $t5 pentru a insera in vector

jal criptare

jal creare\_vector\_invers #creez vectorul invers pentru decriptare

jal decriptare

li $v0, 10

syscall

e\_prim:

li $t0, 2 #i

mul $t2, $t0, $t0 #t2=i\*i

j e\_prim\_loop

e\_prim\_loop:

bgt $t2, $a0, nr\_e\_prim# if i\*i>p nr e prim

rem $t3, $a0, $t0 #t3=p%i

beq $t3, $zero, nr\_nu\_e\_prim #in t4 retin daca nr se imparte la i sau nu

addi $t0, $t0, 1 #daca nr nu se imparte la i fac i++ si reiau bucla

mul $t2, $t0, $t0

j e\_prim\_loop

nr\_e\_prim:

addi $v0, $zero, 1

jr $ra

nr\_nu\_e\_prim:

add $v0, $zero, $zero

jr $ra

afisare\_mesaj:

la $a0, mesaj

li $v0, 4

syscall

li $v0, 10 #dupa ce afisez mesajul corepunzator inchei programul

syscall

gasire\_generator:

addi $a1, $a1, 1 #il incrementez pe i cu 1

li $t4, 0 #pozitia curenta pe care adaug in vectorul v la nivel de memorie

li $t5, 1 #valoarea pe care o introduc in vector

sw $t5, v($t4) #primul element din vectorul v este 1

move $t2, $a1 #t2 retine restul mod p a feicarei putere

move $t5, $a1 #valoarea pe care o introduc in vector

addi $t4, $t4, 4 #parcurc vectorul din 4 in 4 la nivel de memorie

sw $t5, v($t4) #al doilea element din vectorul v este chiar i

bgt $a1, $t0, am\_gasit\_generatorul #cat timp i<=p-1 caut generatorul, iar eticheta

#intoarcere\_la\_main nu va fi apelata niciodata pentru ca sunt puse foarte bine conditiile pe cazuri #in urmatoarele etichete

li $t3, 2#t3 = (puterea curenta a lui i)+1

j gasire\_generator\_loop

gasire\_generator\_loop:

bgt $t3, $t0, gasire\_generator #if !(((puterea lui i)+1)>p-1) <=> if!((puterea lui i)>p-2) <=> puterea lui i<=p-2

mul $t2, $t2, $a1#$t2=$t2\*i

rem $t2, $t2, $t0 #t2=t2%p

mul $t5, $t5, $a1 #$t5=$t5\*i

rem $t5, $t5, $t0 #t5=t5%p (valoarea care se va pune in vector)

addi $t4, $t4, 4 #cu $t4 sarim locatii din memorie din 4 in 4

sw $t5, v($t4) #retinem pe $t5 in vector

addi $t3, $t3, 1 #a crescut puterea cu 1

beq $t2, 1, am\_gasit\_rest\_1

beq $t3, $t0, nu\_e\_asta\_generatorul #daca (puterea curenta+1)=p <=>

# puterea curenta = p-1 dar restul nu e 1

#daca restul nu e 1 dar nici puterea nu e p-1 refacem loop-ul

j gasire\_generator\_loop

am\_gasit\_rest\_1:

beq $t3, $t0, am\_gasit\_generatorul #daca (puterea curenta+1)=p <=>

#puterea curenta=p-1 si retsul este 1 am gasit generatorul

#daca restul e 1 si puterea nu e p-1 nu am gasit generatorul

j gasire\_generator

nu\_e\_asta\_generatorul:

j gasire\_generator

am\_gasit\_generatorul:

move $a0, $a1

li $v0, 1

syscall

la $a0, line

li $v0, 4

syscall

li $t2, 0 #pozitia in vector pe care adaugam

li $t3, 1 #puterea pe care o vom incarca in vector+1

li $t1, 0

jr $ra

intoarcere\_la\_main:

jr $ra

criptare:

li $t0, 0 #pozitia curenta din sirul de caractere

j criptare\_loop

criptare\_loop:

lb $t2, sir1($t0) #t2 retine elementul curent

beq $t2, 10, afisare\_criptare #cand ajungem la newline ne oprim cu parcurgerea sirului

addi $t1, $t2, -65 #ce putere ii corespunde acestei litere

lw $t3, p

addi $t3, $t3, -1 #$t3=p-1

bgt $t1, $t3, afisare\_mesaj\_incompatibilitate

li $t3, 25

bgt $t1, $t3, afisare\_mesaj\_incompatibilitate #daca caracterul inserat are codul

#ascii mai mare decat 90 nu poate fi prelucrat

blt $t1, $zero, afisare\_mesaj\_incompatibilitate

mul $t1, $t1, 4 #pozitia din vectorul v corespunzatoare pozitiei in alfabet a literei

#curente

lw $t3, v($t1) #t3 retine pozitia in alfabet a literei corespunzatoare literei curente

lb $a1, litere($t3) #in a1 retin litera corespunzatoare literei curente

sb $a1, sir\_criptat($t0) #salvez litera corespunzatore literei curente in sirul criptat

addi $t0, $t0, 1 #trec la urmatoarea litera

j criptare\_loop #refac loop-ul

afisare\_criptare:

sb $zero, sir\_criptat($t0) #adaug si caracterul null in sirul criptat

la $a0, sir\_criptat #afisez sirul criptat

li $v0, 4

syscall

la $a0, line

li $v0, 4

syscall

jr $ra

decriptare:

li $t0, 0 #pozitia curenta din sirul de caractere sir2

j decriptare\_loop

decriptare\_loop:

lb $t2, sir2($t0) #codul ascii al caracterului curent

beq $t2, 10, afisare\_decriptare #cand ajung la newline ma opresc

addi $t1, $t2, -65 #pozitia in alfabet a literei curente

lw $t3, p

addi $t3, $t3, -1 #$t3=p-1

bgt $t1, $t3, afisare\_mesaj\_incompatibilitate #daca indicele literei in alfabet este

#mai mare ca p-1 nu poate fi prelucrat

li $t3, 25

bgt $t1, $t3, afisare\_mesaj\_incompatibilitate #daca caracterul inserat are codul

#ascii mai mare decat 90 nu poate fi prelucrat

blt $t1, $zero, afisare\_mesaj\_incompatibilitate #daca valoarea in ascii a

#caracterului inserat este mai mica decat 65 atunci caracterul nu poate fi prelucrat

mul $t1, $t1, 4

lw $t3, vi($t1) #pozitia literei corespunzatoare literei curente

lb $a1, litere($t3) #in a1 retin litera corespunzatoare literei curente

sb $a1, sir\_decriptat($t0) #adaug litera gasita in sirul decriptat

addi $t0, $t0, 1 #trec la litera urmatoare

j decriptare\_loop

afisare\_decriptare:

sb $zero, sir\_decriptat($t0)

la $a0, sir\_decriptat

li $v0, 4

syscall

jr $ra

creare\_vector\_invers:

li $t0, 0 #i

li $t2, 0 #contorul cu care merg prin memorie din 4 in 4

lw $t1, p

addi $t1, $t1, -1 #$t1=p-1

j creare\_vector\_invers\_loop

creare\_vector\_invers\_loop:

bge $t0, $t1, intoarcere\_la\_main #daca i>=p-1 ma intorc la main

lw $t3, v($t2)#t3=v[i]

mul $t3, $t3, 4#il inmultesc pe t3 cu 4 ca sa pot lucra cu el pe memoria vectorului

sw $t0, vi($t3)#vi[t3]=i

addi $t0, $t0, 1#trec la elementul urmator

addi $t2, $t2, 4#trec la elementul urmator pe memorie

j creare\_vector\_invers\_loop

afisare\_mesaj\_incompatibilitate:

la $a0, mesaj\_incompatibilitate

li $v0, 4

syscall

la $a0, line

li $v0, 4

syscall

jr $ra

EXEMPLE DE DATE DE INTRARE:

1. p=7, generatorul este 3, ACAD -> BCBG si BCBG->ACAD
2. p=8 s-a obtinut ca Numarul citit nu este prim
3. p=11 s-a obtinut generatorul 2, ABCDEFGHIJK -> BCEIFKJHDGB si FACB->EABA

0 1 2 3 4 6 7 8 10 11

1 2 4 8 5 10 7 7 3 6