**Tema de laborator – Arhitectura sistemelor de calcul**

**Descrierea solutiei:**

Cerinta 1:

* Am citit numarul p. Daca p nu este prim, atunci sigur are macar un divizor in intervalul [2, p/2]. Am parcurs toate numerele de la 2 la p/2 pentru a verifica daca p are sau nu vreun divizor. Daca p se divide cu macar un numar, atunci p nu este prim, asa ca am afisat mesajul “Numarul dat nu este prim” si am oprit executia programului.
* Daca p este prim, atunci am inceput sa caut un generator.
* Fie g un posibil generator. Am calculat g^k, pentru k=0...p-2. Fie x = valoarea lui g^k, pentru un k fixat. Atunci, pentru k = 0, x = 1, iar pentru k > 0, x = (x\_anterior \* g) % p. La fiecare pas am retinut valoarea lui g^k in al k – lea element al unui vector. La fiecare pas am parcurs toate elementele vectorului de la 0 la k – 1, pentru a vedea daca valoarea lui g^k s-a repetat pana acum. Daca am gasit in vector o aceeasi valoare, inseamana ca g nu este generator, asa ca am trecut la urmatorul posibil generator.

Cerinta 2:

* Am citit mesajul clar, pe care l-am retinut intr-un sir de caractere. Am parcurs sirul de caractere, pana am ajuns la caracterul ‘\0’. Pentru fiecare caracter, am parcurs un alfabet, retinut la nivel de memorie, pentru a afla indicele din alfabet a literei curente. Fie i indicele in alfabet a literei curente. Atunci valoare criptata corepunzatoare caracterului curent este a v[i]-a literea din alfabet. Pentru a accesa a v[i]-a litera, am utilizat un alfabetul retinut la nivel de memorie.
* Am salvat mesajul criptat intr-un sir pe care l-am retinut in memorie.
* Am afisat mesajul criptat.

Cerinta 3:

* Am citit mesajul criptat, pe care l-am retinut intr-un sir de caractere. Am parcurs sirul de caractere, pana am ajuns la caracterul ‘\0’. Pentru fiecare caracter, am parcurs un alfabet, retinut la nivel de memorie, pentru a afla indicele din alfabet a literei curente. Fie i indicele in alfabet a literei curente. Am parcurs vectorul v, pentru a gasi o pozitie k, astfel incat g^k = i (a k-a valoare din v). Atunci valoarea decriptata corespunzatoare caracterului curenta este a k – a litera din alfabet.
* Am salvat mesajul decriptat intr-un sir pe care l-am retinut in memorie.
* Am afisat mesajul decriptat.

**Codul:**

.data

v:.space 50 #v reprezinta vectorul in care am salvat puterile generatorulu

#in a k-a valoare din v se va afla g^k

sir1:.space 50 #sir1 este sirul de caractere in care salvez mesajul clar

sir2:.space 50 #sir2 este sirul de caractere in care salvez mesajul criptat

sol1:.space 50 #in sol1 voi salva solutia pentru a 2-a cerinta(criptarea mesajului clar)

sol2:.space 50 #in sol2 voi salva solutia pentra a 3-a cerinta(mesajul clar corespunzator #mesajului criptat)

enter:.asciiz "\n" #il folosesc pentru a afisa mesajele cu spatii

msg1:.asciiz "Numarul dat nu este prim!"

msg2:.asciiz "Generatorul g este: "

alfabet:.asciiz "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

.text

main:

li $v0, 5 #codul apelului sistem corespunzator lui READ INT

syscall #informam sistemul sa se uite in $v0

move $t0, $v0 #mutam continutul din $v0 in $t0

la $a0, sir1 #in sir1 citim primul sir ce caractere (mesajul clar)

li $a1, 50 #in $a1 punem dimensiunea maxima pe care o poate lua sir1

li $v0, 8

syscall

la $a0, sir2 #in sir2 citim al doilea sir ce carctere (mesajul criptat)

li $a1, 50 #in a1 punem dimensiunea maxima pe care o poate lua sir2

li $v0, 8

syscall

div $t2, $t0, 2 #$t2 ia valoare lui $t0 / 2

li $t1, 2 #$t1 e folosit pentru a parcurge numerele de la 2 la $t0 / 2

loop1:

bgt $t1, $t2, exit1 #daca am ajuns pana la $t2 si nu am gasit niciun divizor, atunci sarim la #exit1

rem $t3, $t0, $t1 #in $t3 punem restul impartirii lui $t0 la $t1

beqz $t3, afis1 #daca restul este 0, atunci $t0 nu e prim si iesim

addi $t1, 1

j loop1 #revenim la loop1

afis1:

#daca am ajuns aici, atunci $t0 nu este prim

la $a0, msg1 #incarcam in $a0 mesajul corespunzator

li $v0, 4 #codul apelului sistem corespunzator lui PRINT STRING

syscall

li $v0, 10 #codul apelului sistem corespunzator lui EXIT

syscall

exit1: #numarul dat este prim, si atunci incepem sa cautam generatorul

#$t1 este numarul pe care il verificam daca este generator sau nu

li $t1, 1 #initializam $t1 cu 1

loop2:

#$t2 reprezinta puterea la care ne aflam

li $t2, 0 #initializam $t2 cu 0

#$t3 reprezinta valoarea lui ($t1 ^ $t2) % $t0

li $t3, 1 #initializam $t3 cu 1 ($t2 ^ 0)

li $t4, 0 #$t4 e folosit pt a sari locatii in memorie din 4 in 4

addi $t7, $t0, -2 #$t7 reprezinta puterea la care trebuie sa ne oprim

loop3:

beq $t2, $t7, exit2 #daca am ajuns pana la final, inseamna ca $t1 este generator si iesim

sw $t3, v($t4) #retinem in v($t4) valoarea lui $t3

#trecem la urmatoarea putere

mul $t3, $t3, $t1 #inmultim $t3 cu $t1

rem $t3, $t3, $t0 #$t3 devine restul impartirii lui $t3 la $t0

j check #verificam daca noua putere nu a mai aparut inainte

check:

li $t5, 0 #$t5 e folosit pentru a sari locatii in memorie din 4 in 4

loop4:

bge $t5, $t4, cont1 #daca am ajuns pana la $t4, atunci valoarea lui $t3 nu s-a mai gasit pana #acum si iesim

lw $t6, v($t5) #incarcam in $t6 valoarea lui v($t5)

beq $t6, $t3, NextGenerator #daca $t6 este egal cu $t3, atunci $t1 nu este generator, #si atunci trecem la urmatorul posibil generator

addi $t5, 4 #crestem $t5 din 4 in 4 pentru a accesa memoria

j loop4 #revinim la loop4

cont1:

#daca am ajuns pana aici, atunci nu am gasit valoare lui $t3 pana acum

addi $t4, 4 #crestem $t4 din 4 in 4 pentru a accesa memoria

addi $t2, 1 #crestem $t2 (puterea la care ne aflam) cu 1

j loop3 #revenim la loop3

NextGenerator:

#daca am ajuns aici, atunci $t1 nu este generator, si atunci trecem la #urmatorul numar

addi $t1, 1 #crestem $t1 cu 1

j loop2 #revenim la loop2

exit2:

#generatorul se afla in $t0

sw $t3, v($t4) #retinem in v($t4) valoare lui $t3 (($t1 ^ ($t0 - 2)) % $t0)

li $v0, 4 #codul apelului sistem corespunzator lui PRINT STRING

la $a0, msg2 #incarcam in $a0 mesajul corespunzator

syscall

li $v0, 1 #codul apelului sistem corespunzator lui PRINT INT

move $a0, $t1 #incarcam in $a0 valoarea lui $t1 (generatorul)

syscall

#afisam un enter dupa ce am afisat generatorul

li $v0, 4 #codul apelului sistem corespunzator lui PRINT STRING

la $a0, enter #incarcam in $a0 valoare lui enter(un sir ce caractere egal cu '\n')

syscall

#$t2 este folosit pentru a sari locatii de memorie din 1 in 1

li $t2, 0 #initializam $t2 cu 0

lb $t3, sir1($t2) #$t3 este byte-ul curent din sirul sir1 (mesajul clar)

li $t7, 26 #$t7 este folosit pentru a retine dimensiunea alfabetului

loop5:

beqz $t3, afis2 #daca byte-ul curent este 0, atunci s-a terminat sirul de caractere si

#sarim la afis2

#$t5 e folosit pentru a sari locatii din memorie din 1 in 1 (atunci cand #cautam litera $t3 in alfabet)

li $t5, 0 #initializam $t5 cu 0

lb $t4, alfabet($t5) #$t4 reprezinta valoare byte-ul curent din alfabet

loop6:

beq $t3, $t4, cont2 #daca byte-ul curent din sir1 este egal cu byte-ul curent in alfabet,atunci #sarim la cont2

addi $t5, 1 #crestem $t5 din 1 in 1 pentru a accesa memoria

bge $t5, $t7, cont3 #daca s-a terminat alfabetul, atunci byte-ul curent din sir1 nu se gaseste #in alfabet si sarim la cont3

lb $t4, alfabet($t5) #actualizam $t4 cu byte-ul curent din alfabet

j loop6 #revenim la loop6

cont3: #daca am ajuns aici, atunci byte-ul corespunzator din sir1 nu s-a gasit in #alfabet

addi $t2, 1 #crestem $t2 cu 1, pentru a trece la urmatorul byte din sir1

lb $t3, sir1($t2) #actualizam $t3 cu byte-ul curent din sir1

j loop5 #revenim la loop5

cont2:

#daca am ajuns aici, inseamna ca $t3 = $t4

li $t4, 4 #$t5 reprezinta pozitia literei $t4 in alfabet. Deoarece fiecare element

#din alfabet ocupa cate un byte, iar un element din v ocupa 4 bytes,

#pozitia corespunzatoare lui $t5 in v este $t5 \* 4

mul $t5, $t5, $t4 #inmultim $t5 cu 4

lw $t5, v($t5) #$t5 ia valoarea corespunzatoare din v(pozitia literei dupa codificare)

lb $t5, alfabet($t5) #$t5 ia valoarea literei corespunzatoare din alfabet

sb $t5, sol1($t2) #salvam in sol($t2) valoarea lui $t5

addi $t2, 1 #crestem $t2 cu 1, pentru a trece la urmatorul byte din sir1

lb $t3, sir1($t2) #acutalizam $t3 cu byte-ul curent din sir1

j loop5 #revenim la loop5

afis2: #daca am ajuns aici, inseamna ca s-a terminat sirul de caractere

la $a0, sol1 #incarcam in $a0 solutia (sol1)

li $v0, 4 #codul apelului sistem corespunzator lui PRINT STRING

syscall

li $v0, 4 #afisam un enter dupa ce am afisat mesajul criptat

la $a0, enter

syscall

#$t2 este folosit pentru a sari locatii in memorie din 1 in 1

li $t2, 0 #initializam $t2 cu 0

lb $t3, sir2($t2) #$t3 reprezinta byte-ul curent din sir2

loop7:

beqz $t3, afis3 #daca byte-ul curent este 0, atunci s-a terminat sirul de caractere si

#sarim la afis3

#$t5 e folosit pentru a sari locatii din memorie din 1 in 1

li $t5, 0 #initializam $t5 cu 0

#$t4 reprezinta byte-ul curent din alfabet

lb $t4, alfabet($t5) #initializam $t4 cu litera corespunzatoare din alfabet

loop8:

beq $t3, $t4, cont4 #daca byte-ul curent din sir1 este egal cu byte-ul curent in alfabet, #atunci sarim la cont4

addi $t5, 1 #crestem $t5 cu 1

bge $t5, $t7, cont5 #daca s-a terminat alfabetul, atunci byte-ul curent din sir1 nu se gaseste #in alfabet si sarim la cont5

lb $t4, alfabet($t5) #actualizam $t4 cu byte-ul curent din alfabet

j loop8 #revenim la loop8

cont5: #daca am ajuns aici, atunci byte-ul corespunzator din sir2 nu s-a gasit in

#alfabet

addi $t2, 1 #crestem $t2 cu 1, pentru a trece la urmatorul byte din sir1

lb $t3, sir2($t2) #actualizam $t3 cu byte-ul corespunzator curente din sir2

j loop7 #revenim la loop7

cont4:

#daca am ajuns aici, inseamna ca $t3 = $t4

#vom parcurge vectorul v, pentru a vedea ce numar din v are valoarea

#$t5(pozitia literei $t3 in alfabet). Litera corespunzatoare acelei pozitii

#din alfabet este litera care s-a criptat

#$t4 e folosit pentru a sari locatii din memorie din 4 in 4

li $t4, 0 #initializam $t4 cu 0

lw $t6, v($t4) #incarcam in $t6 valoarea lui v($t6)

cauta:

beq $t6, $t5, exit3 #daca $t6 este egal cu $t5, atunci sarim la exit3

addi $t4, 4 #crestem $t4 din 4 in 4 pentru a accesa memoria

lw $t6, v($t4) #incarcam in $t6 valoare lui v($t4)

j cauta #revenim la cauta

exit3:

#daca am ajuns aici, inseamna ca am gasit o pozitie, a carei valoare este #retinuta in $t4, astfel incat v($t4) = $t5

li $t5, 4

div $t4, $t4, $t5 #impartim $t4 la 4, deoarce fiecare element in v acupa cate 4 bytes, in #timp cu o litera din alfabet ocupa un byte

lb $t4, alfabet($t4) #$t4 ia valoarea literei corespunzatoare din alfabet

sb $t4, sol2($t2) #salvam in sol($t2) valoarea lui $t4

addi $t2, 1 #crestem $t2 cu 1, pentru a trece la urmatorul byte

lb $t3, sir2($t2) #$t3 ia valoarea byte-ul curent din sir2

j loop7 #revenim la loop7

afis3:

la $a0, sol2 #il incarcam in $a0 pe sol2 (mesajul decriptat)

li $v0, 4 #codul apelului sistem pentru PRINT STRING

syscall

li $v0, 10 #codul apelului sistem pentru EXIT

syscall

**Exemple:**

Trei exemple pe care am rulat programul sunt:

1. 7

ACAD

BCBG

(Raspunsul a fost: “Generatorul g este: 3

BCBG

ACAD”)

1. 8

ACAD

BCBG

(Raspunsul a fost: “Numarul dat nu este prim!”)

1. 13

CAJIKL

BCDEMBD

(Raspunsul a fost: “Generatorul g este: 2

EBFJKH

ABECGAE”)