* Descrierea informala a solutiei:

Cerinta 1 : Pentru inceput verificam daca numarul p este prim printr-o parcurgere a tuturor numerelor pana la p/2 si verificam daca vreunul dintre acestea se divide cu p. Daca gasim un numar care se divide cu p atunci afisam un mesaj ca p nu este prim si iesim din program. Daca p este prim continuam prin a cauta generatorul lui Zp. Luam toate numerele mai mici decat p pe rand. Clasei de resturi curenta ii calculam puterile retinand intotdeauna ultima putere mod p. Retinem puterile calculate intr-un vector. Verificam secvential daca fiecare clasa de resturi modulo p se afla in vector. Daca nu gasim un element din Zp in vectorul de puteri atunci clasa de resturi pe care am considerat-o nu este generator. Altfel, daca gasim toate elementele lui Zp in vectorul de puteri al clasei de resturi pe care am considerat-o, inseamna ca aceasta este generator. Stocam generatorul in variabila g.

Cerinta 2: Citim textul ce urmeaza a fi criptat. Luam fiecare litera a textului si vedem pe ce pozitie se afla ea in alfabet. Vedem ce avem pe aceasta pozitie in vectorul de puteri. Luam puterea din vectorul de puteri si ne uitam ce pozitie reprezinta ea in alfabet. Litera rezultata astfel reprezinta criptarea literei din textul initial. Luam litera obtinuta si o punem intr-un nou string, pe care l-am denumit sugestiv “criptat”, si in care vom obtine textul initial, criptat. In acest sens vom retine un sir “alfabet” in care avem stocat string-ul “ABCDEFGHIJKLMNOPQRSTUV” ca sa lucram mai usor.

Cerinta 3: Citim textul ce urmeaza a fi decriptat. Luam fiecare litera a textului si vedem pe ce pozitie se afla ea in alfabet. Fie x aceasta pozitie. Il cautam pe x in vectorul de puteri. Vedem pe ce pozitie se afla x in vectorul de puteri si cautam litera care se afla pe aceeasi pozitie in alfabet. Litera gasita reprezinta decriptarea literei initiale. O punem in string-ul “normall” in care vom obtine textul initial decriptat.

La final afisam g, criptat si normall.

* Exemple de date:

INPUT\_1: INPUT\_2: INPUT\_3:

23 8 11

LUANA CEAI

MIPS EFBD

OUTPUT\_1: OUTPUT\_2: OUTPUT\_3:

5 este generator p nu este prim 2 este generator

WMBVB EFBD

UGRM CEAI

* Codul sursa:

.data

p:.space 4

puteri:.space 400

g:.space 4

A:.byte 'A'

sp:.asciiz "\n"

normal:.space 100

normall:.space 100

criptat:.space 100

criptatt:.space 100

notprime:.asciiz "p nu este prim"

generator:.asciiz " este generator"

alfabet:.asciiz "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

.text

main:

li $v0, 5 #se citeste numarul p de la tastatura

syscall

sw $v0, p

lw $t0, p

li $t1, 2 #vom folosi $t1 pe post de i in loop. avand in vedere ca scopul este sa cautam divizorii lui p, vom incepe cu i de la 2

div $t4, $t0, 2 #retinem in $t4 valoarea p/2

beq $t0, 1, nu\_e\_prim #daca p = 1, atunci p nu e prim si sarim la eticheta de iesire din program pe caz "p nu e prim"

beq $t0, 2, caz2 #tratam particular cazul in care ne aflam in Z2

beq $t0, 3, e\_prim #p = 3 e prim dar p/2 = 1 deci nu intra in loop asa ca sarim la eticheta de cautare a generatorului

j loop

caz2: #tratam cazul in care p = 2 astfel: generatorul este 1 si il stocam in variabila g. 1 va fi singurul element al vectorului "puteri" in care tinem puterile generatorului

li $t6, 1

li $t5, 0

sw $t6, g

sw $t6, puteri($t5)

j criptare #sarim la criptare

loop:

bgt $t1, $t4, e\_prim #verificam i <= p/2. daca nu am gasit niciun i care sa il divida pe p inseamna ca p e prim si trecem la eticheta de cautare a generatorului

rem $t3, $t0, $t1 #in $t3 tinem restul impartirii lui p la i

beq $t3, 0, nu\_e\_prim #daca restul impartirii lui p la i este 0, atunci sarim la eticheta de iesire din program pe caz "p nu e prim"

addi $t1, $t1, 1 #i++

j loop

nu\_e\_prim: # eticheta de iesire din program pe caz "p nu e prim"

la $a0, notprime

li $v0, 4

syscall

li $v0, 10

syscall

e\_prim:

lw $t0, p

li $t1, 2 # folosim $t1 pe post de i in loop2

j loop2

loop2: #luam fiecare clasa de resturi mai mica decat p

bge $t1, $t0, exit

#aici vom face niste initializari pt urmatoarele loop-uri

li $t2, 1 #registrul in care vom retine puterile clasei din registrul $t1

li $t3, 0 #registrul cu care vom parcurge vectorul

sw $t2, puteri($t3) #pe poz 0 din vectorul de puteri punem 1

addi $t3, $t3, 4

sw $t1, puteri($t3) #pe poz 1 din vectorul de puteri punem clasa curenta

move $t2, $t1 #punem clasa curenta in $t2

addi $t3, $t3, 4

j loop3

loop3: #pt fiecare clasa de resturi, calculam puterile si le punem intr-un vector pe care urmeaza sa il verificam mai tarziu

beq $t2, 1, verif\_vector #while(putere!=1). cand puterea ajunge iar 1, sarim la verificarea vectorului

mul $t2, $t2, $t1 #pt a afla puterea curenta inmultim puterea precedenta cu clasa

rem $t2, $t2, $t0 #modulo p

sw $t2, puteri($t3) #stocam noua putere in vector

addi $t3, $t3, 4

j loop3

verif\_vector:

li $t2, 0 #folosim $t2 pe post de i. aici i are seminificatia de clasa de resturi.

j loop4

loop4: #verificam secvential daca valoarea stocata in $t1 este generator

addi $t2, $t2, 1

bge $t2, $t0, este\_generator #daca am verificat aparitia tuturor claselor de resturi modulo p in vectorul de puteri, inseamna ca $t1 este generator

li $t3, 0 #initializam registrul cu care parcurgem vectorul cu 0 mereu cand incepem sa cautam o clasa de resturi in vector

li $t4, 0 #folosim $t4 pe post de j si il reinitializam cu 0 de fiecare data cand ne intoarcem in loop4

j loop5

loop5:

bge $t4, $t0, nu\_este\_generator #daca $t2 nu este gasit in vector inseamna ca $t1 nu este generator

lw $t5, puteri($t3) #punem in $t5 puterea curenta

beq $t2, $t5, loop4 #daca l-am gasit pe $t2 in vector sarim inapoi la loop4 si cautam urmatoarea clasa de resturi in vector

addi $t4, $t4, 1

addi $t3, $t3, 4

j loop5

nu\_este\_generator:

addi $t1, $t1, 1 #luam urmatoarea clasa de resturi si repetam procesul de verificare daca este generator

j loop2

este\_generator:

sw $t1, g #stocam valoarea din $t1 in variabila g ca sa o afisam mai tarziu

j criptare

#sfarsit cerinta 1

#inceput cerinta 2

criptare:

la $a0, normal #citim textul necriptat

li $a1, 99

li $v0, 8

syscall

li $t0, 0 #registrul cu care parcurgem textul normal

lb $t1, normal($t0) #registrul in care tinem litera curenta din text

li $t3, 0 #registrul cu care parcurg textul criptat

j loop6

#am observat ca la citirea unui string de la tastatura, se mai pune o valoare dupa finalul stringului. asa ca in loc sa verific daca litera curenta este 0, voi verifica daca litera urmatoare este 0

loop6:

addi $t5, $t0, 1 #pe pozitia $t0 am litera curenta. pe pozitia $t5 am litera de dupa ea

lb $t4, normal($t5) #in $t4 am valoarea din string de dupa valoarea mea curenta, $t1

beqz $t4, decriptare #daca $t4 este 0 inseamna ca am criptat tot string-ul si trecem la decriptare, urmand sa afisam string-ul criptat mai tarziu

lb $t4, A #in $t4 pun acum caracterul 'A'

subu $t5, $t1, $t4 #fac diferenta dintre litera curenta($t1) si 'A' ca sa vad pe ce pozitie in alfabet se afla litera curenta

mul $t5, $t5, 4 #inmultesc $t5 cu 4 ca sa obtin pozitia din vecotrul puteri

lw $t4, puteri($t5) #pun in $t4 valoarea de la pozitia $t5 din vector (adica generatorul^($t5))

lb $t2, alfabet($t4) #pun in $t2 litera din alfabet de la pozitia $t4

sb $t2, criptat($t3) #stochez litera criptata in vectorul criptat

addi $t3, $t3, 1

addi $t0, $t0, 1

lb $t1, normal($t0) #pun in $t1 urmatoarea litera si repet

j loop6

#sfarsit cerinta 2

#inceput cerinta 3

decriptare:

la $a0, criptatt #citesc string-ul criptat

li $a1, 99

li $v0, 8

syscall

li $t0, 0 #registrul cu care parcurgem textul criptat

lb $t1, criptatt($t0) #registrul in care tinem litera curenta din text

li $t3, 0 #registrul cu care parcurg textul decriptat

j loop7

loop7:

beqz $t1 afisare

lb $t2, A #pun in $t2 litera 'A'

subu $t5, $t1, $t2 #fac diferenta dintre litera curenta si 'A' ca sa aflu pozitia literei curente in alfabet

li $t4, 0 #folosesc pe post de i in loop8 in care voi cauta $t5 in vectorul de puteri ca sa vad pe ce pozitie este, adica sa vad ce putere a generatorului este $t5

j loop8

loop8: #cautam secvential pe ce pozitie se afla $t5 in vectorul de puteri

lw $t2, p

bge $t4, $t2, afisare #daca $t4 sare afara din vectorul de puteri inseamna ca $t4 e valoarea de la finalul string-ului (vezi comentariul de deasupra lui loop6), deci inseamna ca am decriptat tot string-ul si putem sari la afisare

mul $t4, $t4, 4 #inmultim $t4 cu 4 ca sa obtinem pozitia din vectorul de puteri

lw $t2, puteri($t4) #punem in $t2 valoarea de la pozitia $t4

beq $t5, $t2, gasit #daca $t2 = $t5 inseamna ca am gasit pozitia pe care se afla $t5 in vectorul de puteri

div $t4, $t4, 4 #$t4 isi revine la normal

addi $t4, $t4, 1 #i++

j loop8

gasit:

#pun in normall(textul decriptat) litera de pe pozitia $t4

div $t4, $t4, 4

lb $t2, alfabet($t4)

sb $t2, normall($t3)

addi $t3, $t3, 1

addi $t0, $t0, 1

lb $t1, criptatt($t0) #punem in $t1 urmatoarea litera din textul criptat si repetam procedeul de decriptare

j loop7

afisare:

lw $a0, g #afisam generatorul

li $v0, 1

syscall

la $a0, generator

li $v0, 4

syscall

la $a0, sp

li $v0, 4

syscall

la $a0, criptat #afisam textul criptat

li $v0, 4

syscall

la $a0, sp

li $v0, 4

syscall

la $a0, normall #afisam textul normal

li $v0, 4

syscall

j exit

exit:

li $v0, 10

syscall