Descrierea informala a solutiei utilizate:

1.Am retinut in memorie alfabet: "ABCDEFGHIJKLMNOPQRSTUVWXYZ", pentru a putea accesa mai usor literele si un vector prin care pot realiza corespondenta intre puterea k si (generator^k)%p.

2.Citesc numarul p si verific daca este prim:

- daca p<2, evident p nu este numar prim, asa ca afisez mesajul corespunzator;
- daca p=2, atunci p nu are generator;
- daca p>2, iau pe rand posibilii divizori incepand de la 2 pana la [p/2] si testez daca p se imparte la vreunul dintre ei. Daca da, numarul nu este prim. Daca nu se imparte la niciunul dintre aceste numere, inseamna ca p este numar prim. In continuare, citesc mesajul care trebuie criptat si mesajul care trebuie decriptat.

3.Caut generatorul g, care poate lua valori de la 2 la p-1.

Pentru a verifica daca g este generator, retin mereu ultima putere mod p, adica:
$g^k \bmod p = (g \cdot (g^{k-1} \bmod p)) \bmod p = g \cdot g_{pas\_anterior} \bmod p$.
Daca intre doua resturi egale cu 1 succesive se afla p-1 termeni, inseamna ca numarul respectiv este generatorul. Daca nu, g++ si testez din nou.

4.Dupa ce am gasit acest generator, construiesc vectorul v pentru a retine corespondenta dintre puterea k si (generator^k)%p.

5.Codarea mesajului: caut fiecare litera din mesajul dat in alfabet si retin pozitia x pe care aceasta se afla in alfabet (indexarea incepand de la 0), apoi caut litera din alfabet corespunzatoare lui v[x].

Ex: A → pozitia 0 in alfabet → alfabet(v[0])

6.Decodarea mesajului: caut fiecare litera din mesajul dat in alfabet si retin pozitia y pe care aceasta se afla in alfabet (indexarea incepand de la 0); caut numarul x astfel incat v[x]=y si apoi caut litera de po pozitia x din alfabet.

Ex: G → pozitia 6 in alfabet → v[x]=6 → alfabet(x)

Exemple:

1. Daca p=13, generatorul este 2
   Mesajul care trebuie codat ABDG => BCIM
   Mesajul care trebuie decodat CDEK => BECK
2. Daca p=17, generatorul este 3
   Mesajul care trebuie codat MOCP => ECJG
   Mesajul care trebuie decodat HJIN => LCKE
3. Daca p=23, generatorul este 5
   Mesajul care trebuie codat TRBL => HPFW
   Mesajul care trebuie decodat QMEB => IUEA
4. Daca p=6 => numarul citit nu este prim
5. Daca p=2 => numarul nu are generator

Codul sursa:

```
.data
  p: .space 4  #Numarul citit p
  v: .space 400  #Vectorul in care se retin (g^k)%p - corespondenta
```

```
  npr: .asciiz "Numarul citit nu este prim"  #Mesajul afisat daca p nu este prim
  s: .space 100  #Mesajul care trebuie criptat
  sdc: .space 100  #Mesajul care trebuie decriptat
  alfabet: .asciiz "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  sp: .asciiz "\n"
  mcr: .asciiz "Mesajul criptat este:"
  mdcr: .asciiz "Mesajul decriptat este:"
  mg: .asciiz "Generatorul este:"
  ng: .asciiz "Numarul 2 nu are generator"

.text
main:
#Citire numar p:
  li $v0,5
  syscall
  move $t0, $v0
  sw $t0,p

#Verific daca numarul citit p<2: atunci p nu este numar prim
  li $t1,2
  blt $t0, $t1, nueprim

#Verific daca p=2: atunci nu exista generatoR
  beq $t0, $t1, nuexistag
  j prim  #daca p!=2, testez daca e prim
    nuexistag:
      la $a0, ng
      li $v0, 4
      syscall
      li $v0, 10
      syscall

#p>=2: Verificare p nr. prim
prim:
  div $t1, $t0, 2 #t1 - partea intreaga [p/2]
  li $t2, 2 #registrul t2 indica divizorul

testareprim:
  bgt $t2, $t1, eprim  #daca t2(divizor) > t1 ([p/2]): numarul este prim
  rem $t3, $t0, $t2 #retin in registrul t3 - p%divizor
  beq $t3, $0, nueprim #daca t3(p%divizor)==0: numarul nu este prim
  addi $t2, $t2, 1 #divizor++
  j testareprim

nueprim:  #Numarul citit p nu este prim, afisez mesajul corespunzator si ies din
```

```
program
  la $a0, npr
  li $v0, 4
  syscall
  li $v0, 10
  syscall

eprim: #Numarul citit p este prim
#Citesc mesajul ce trebuie criptat
  la $a0, s
  li $a1, 99
  li $v0, 8
  syscall

#Citesc mesajul ce trebuie decriptat
  la $a0, sdc
  li $a1, 99
  li $v0, 8
  syscall

  li $t1, 1 #registrul t1 este folosit pentru indicele i=1
  li $t2, 2 #primul generator, retinul in registrul t2, este 2
  move $t3, $t2 #in registrul t3 calculam puterile generatorului
  j generare #generam puterile

generare:
  beq $t3, 1, verificare #daca puterea generatorului==1, verificam daca indicele == p-
1
  mul $t3, $t3, $t2 #g^k=g^(k-1)*g
  rem $t3, $t3, $t0 #t3=(g^k)%p
  addi $t1, $t1, 1 #i++
  j generare

verificare:
  subi $t4, $t0, 1 #t4=p-1
  beq $t1, $t4, codare #indice=p-1: am gasit generatorul, deci codez si decodez
mesajul
  #daca indicele!=p-1, reinitializez si reiau calculul puterilor generatorului
  j reinitializare

reinitializare:
  li $t1, 1 #registrul t1 este folosit pentru indicele i=1
  addi $t2, $t2, 1 #generator++
  move $t3, $t2 #in registrul t3 calculez puterile generatorului
  j generare
```

```
codare:
afisare: #afisez generatorul retinut in $t2 si trec la rand nou
  la $a0, mg
  li $v0,4
  syscall
  move $a0, $t2
  li $v0, 1
  syscall
  la $a0, sp
  li $v0,4
  syscall

#Construiesc un vector pentru corespondentele dintre puterea k si (g^k)%p
  li $t1, 0 #registrul $t1 e folosit pentru a sari locatii in memorie din 4 in 4
  li $t3, 1 #indicele i=1;
  li $t4, 1 #generator^0==1
calculvector:
  bge $t3, $t0, codaremesaj #daca i>=p:codaremesaj
  sw $t4, v($t1) #retin puterea generatorului in vector
  mul $t4, $t4, $t2 #g^k=g^(k-1)*g
  rem $t4, $t4, $t0 #(g^k)%p
  addi $t1, $t1, 4 #sar locatia in memorie din 4 in 4
  addi $t3, $t3, 1 #i++
  j calculvector

codaremesaj:
  #afisez mcr:"Mesajul criptat este:"
  la $a0, mcr
  li $v0,4
  syscall

  li $t1, 0 #registrul $t1 e folosit pentru a sari locatii in memorie din 1 in 1
  lb $t3, s($t1) #retin in reg.t3 prima litera a mesajului care trebuie codat

for1:
  beq $t3, '\n', decodare #daca ajung la sfarsitul mesajului, trec la functia de decodare
al celui de-al doilea mesaj
  li $t4, 0 #registrul $t4 e folosit pentru a sari locatii in memorie din 1 in 1
  lb $t5, alfabet($t4) #in reg.t5 retin, pe rand, literele alfabetului

  cautarelitera: #caut litera din mesaj in alfabet
  beq $t5, $t3, numar #daca litera din alfabet este aceeasi cu litera din mesaj
  addi $t4,$t4,1  #urmatoarea locatie din memorie
  lb $t5, alfabet($t4)  #urmatoarea litera din alfabet
```

```
  j cautarelitera

  numar:
  mul $t4, $t4, 4 #indicele pentru a accesa vectorul=pozitia pe care este gasita litera*4
  lw $t6, v($t4)  #in reg.t6 retin cifra corespunzatoare
  #afisez litera din alfabet corespunzatoare cifrei retinute in reg.t6
  lb $a0, alfabet($t6)
  li $v0, 11
  syscall

  addi $t1, $t1,1 #urmatoarea locatie din memorie a mesajului care trebuie criptat
  lb $t3, s($t1) #urmatoarea litera din mesaj
j for1

decodare:
#Trec la rand nou
  la $a0, sp
  li $v0,4
  syscall

#Afisez mdcr:"Mesajul decriptat este:"
  la $a0, mdcr
  li $v0,4
  syscall

  li $t0, 0 #registrul $t0 e folosit pentru a sari locatii in memorie din 1 in 1
  lb $t1, sdc($t0) #retin in reg.t1 prima litera a mesajului care trebuie decodat

for2:
  beq $t1, '\n' exit #daca ajung la sfarsitul mesajului, exit
  li $t2, 0 #registrul $t2 e folosit pentru a sari locatii in memorie din 1 in 1 pentru a
parcurge alfabetul
  lb $t3, alfabet($t2) #in reg.t3 retin, pe rand, literele alfabetului

  cautarepozitie:
  beq $t3, $t1, ivector  #daca litera din alfabet este egala cu litera din mesaj, caut in
vector
  addi $t2,$t2,1 #urmatoarea locatie din memorie din alfabet
  lb $t3, alfabet($t2) #urmatoarea litera din alfabet
  j cautarepozitie

  ivector:
  li $t4,0 #registrul $t4 e folosit pentru a sari locatii in memorie din 4 in 4 in vectorul
v
  cautare:
```

```
  lw $t5, v($t4)  #in reg.t5 retin, pe rand, elementele din v
  beq $t2, $t5, transformare #daca t5/v(t4)==pozitia literei in alfabet
  addi $t4, $t4, 4  #urmatoarea locatie din memorie pentru vectorul v
  j cautare

  transformare:
  div $t4, $t4, 4  #obtin pozitia pentru litera corespunzatoare din alfabet impartind
indicele pentru memorie din v la 4
  #afisez litera din alfabet
  lb $a0, alfabet($t4)
  li $v0, 11
  syscall

  addi $t0, $t0,1 #urmatoarea locatie din memorie
  lb $t1, sdc($t0) #urmatorul caracter din mesajul care trebuie decodat
j for2

exit:
  li $v0,10
  syscall
```