Descrierea in limbaj natural a solutiei propuse

La nivel de memorie: ***p*** este ordinul grupului ce urmeaza a fi citit de la tastatura; ***v*** este vectorul in care ulterior vor fi retinute resturile; ***mesaj*** si ***mesaj1*** vor fi afisate in cazul in care numarul p este nu este prim, respectiv este prim; ***nl*** este utilizat pentru a afisa mesajele dupa codificare, respectiv decodificare pe linii separate dar si pentru a se verifica daca s-a ajuns la finalul cuvantului; ***mesaj\_clar*** si **mesaj\_criptat** sunt utilizate pentru a retine in memorie cele 2 mesaje introduse de latastatura; ***alfabet*** este folosit pentru a se putea gasi pozitia corespunzatoare a literei curente in alfabet.

# Verificarea daca p este prim

Se citeste de la tastatura p si se trece in registrul $t0 pentru a putea fi utilizat ulterior in program. In registrul $t2 se pune 2. Se compara cu p. Daca p este mai mic sau egal atunci sare la eticheta ***nu\_prim*** si iese din program. In continuare se verifica daca p este prim. In registrul $t2 punem 2 ca prim posibil divizor. In $t4 punem jumatatea lui p pentru a avea conditie de oprire: daca p nu se imparte la niciun numar pana la jumatatea sa minus 1 atunci el este prim. Din registrul hi este luat restul impartirii lui $t0 (=p) la $t2 (divizorul curent) si este pus in registrul $t3. Daca acesta este 0 atunci p nu este prim si se sare la eticheta ***nu\_prim***. Daca acesta nu este prim se incrementeaza $t2 si se sare inapoi la ***test\_prim***.

# Gasirea generatorului

La eticheta ***continuare\_program*** se ajunge dupa ce au fost verificate toate numerele pana la jumatatea lui p si niciunul nu a fost divizor. In aceast parte a programului se va cauta un generator al grupului. Un prim generator posibil este 2 deoarece 1 la orice putere ramane 1. Se vor verifica toate numerele de la 2 pana la p pana va fi gasit unul care convine. Fiecare generator va fi ridicat succesiv la toate puterile de la 0 la p-1 si apoi in vector va fi mutat restul impartirii sale la p. Oricare ar fi generatorul testat, in v[0] este mereu 0 si in v[1] este chiar generatorul. Restul se obtine conform formulei de optimizare: generatorul inmultit cu generatorul de la pasul anterior si apoi se afla restul immpartirii sale la p. In cazul generatorului cautat, nu se va ajunge ca restul sa fie 1 decat atunci cand puterea este egala cu p-1. Acest lucru este verificat in cadrul etichetei ***verif.*** In cazul in care nu este respectata aceasta conditie inseamna ca nu am gasit generatorul si se incrementeaza registrul $t1 in care avem generatorul curent si se reia testarea acestuia. Daca este respectata conditia atunci se sare la eticheta ***iesire*** unde generatorul este mutat in locatia corespunzatoare de memorie si este afisat.

# Criptarea

In continuare este citit mesajul clar de la tastatura. Mesajul va fi criptat litera cu litera si trecut in locatia din memorie aferenta si apoi mesajul este afisat. In registrul $t2 este pusa prima litera a cuvantului. Registrul $t2 va prelua pe rand fiecare litera a mesajului pana acesta se va termina si se se sare la eticheta in cadrul careia mesajul final va fi afisat. Este cautata pozitia din alfabet a literei din $t2 cu ajutorului variabilei alfabet si in cadrul etichetei ***caut***. Cand este gasita aceasta pozitie ea este inmultita cu 4 reprezentand adresa din memorie fata de vectorul v la care se afla noua valoare ce trebuie sa fie atribuita literei din $t2 (pentru pozitia i in alfabet se va cauta pozitia 4\*i in vector si litera va fi inlocuita cu litera de pe pozitia v[4\*i] din alfabet). Se incrementeaza $t1 si in $t2 este mutat mesaj\_clar($t1) urmatoarea litera din cuvant. Se va proceda similar pana se termina cuvantul. Dupa finalizare criptariii este afisat mesajul in cadrul etichetei ***AFISARE\_MESAJ***.

# Decriptarea

Intr-o maniera asemanatoare se va decripta cel de-al doilea mesaj. Se citeste si stocheaza in memorie la adresa mesaj\_criptat. Registrul $t1 sare locatii de memorie din 1 in 1 pentru a putea fi accesata fiecare litera a mesajului. In $t2 este pusa prima litera a mesajlui. Variabila nl este folosita pentru a opri programul cand s-a ajuns la finalul cuvantului. Se cauta pozitia din alfabet a literei in cadrul etichetei ***caut\_alfabet***. Cand aceasta a fost gasita se cauta in vectorul v. Pentru decodificare daca o litera are pozitia v[4\*i] in alfabet atunci este inlocuita cu cu litera de pe pozitia i. Astfel litera din registrul $t2 este cautata secvential in alfabet. Cand este gasita pozitia, aceasta este cautata printre valorile vectorului. Cand este gasita, valoarea i indicelui este impartita la 4 si litera din $t2 devine litera de pe pozitia i/4 din alfabet. Este inlocuita in memorie si se continua cu urmatoarea litera a mesajului, se incrementeaza $t1 si in $t2 este pusa litera. Se continua pana la finalul mesajului. Decriptarea este o operatie inversa criptarii.

Dupa finalizarea decriptarii se va afisa noul mesaj si se incheie programul.

# Codul sursa:

.data

p: .space 4

v: .space 400

g: .space 4

mesaj: .asciiz "Numarul p nu este prim"

mesaj1: .asciiz "Numarul p este prim"

nl: .asciiz "\n"

mesaj\_clar: .space 100

mesaj\_criptat: .space 100

alfabet: .asciiz "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

sp:.asciiz " "

.text

main:

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

syscall

sw $v0, p

move $t0, $v0 #mutam pe p in $t0

li $t2, 2

ble $t0, $t2, nu\_prim # daca este mai mic decat 2 iese din program

#testare daca p e prim

li $t2, 2 #luam 2 ca prim divizor

div $t4, $t0, 2 #punem in $t4 jumatatea lui p

test\_prim:

beq $t2, $t4, continuare\_program #while $t2<$t4 executa

div $t3, $t0, $t2

mfhi $t3 #punem restul in $t3

beqz $t3, nu\_prim #daca restul e zero nu e prim

addi $t2, $t2, 1

j test\_prim

nu\_prim:

la $a0, mesaj #se afiseaza mesaj corespunzator si se termina programul

li $v0, 4

syscall

li $v0, 10

syscall

continuare\_program:

la $a0, mesaj1 #afiseaza ca e prim si continua

li $v0, 4

syscall

la $a0, nl

li $v0, 4 #afiseaza endl

syscall

li $t1, 0

li $t2, 1

sw $t2, v($t1) #v[0]=1

li $t1, 2 #i=2

for:

beq $t1, $t0, iesire #while i<p

li $t6, 1 #folosim pentru a sari locatii de memorie din 4 in 4; initial are valoarea 1 pentru a pune i in v[1]

li $t5,1

li $t2,1

mul $t6, $t6, 4

sw $t1, v($t6) #punem in v[1] pe i

move $t3, $t1 #$t3=i

ridicare\_la\_putere:

beq $t3, $t5,verif

addi $t6,$t6,4 #folosim pentru a sari locatii de memorie din 4 in 4

addi $t2,$t2,1

mul $t3, $t3, $t1 #se inmulteste g cu g de la pasul anterior

rem $t3,$t3,$t0 #se afla restul impartirii sale la p

sw $t3,v($t6)

j ridicare\_la\_putere

verif:

move $t5, $t0

subu $t5, $t5, 1 #$t5=p-1

beq $t2, $t5, iesire #j==p-1

addi $t1, $t1, 1 #incrementare i

j for

iesire:

sw $t1, g

move $a0, $t1 #muta generatorul in $a0 ca sa il afiseze

li $v0, 1

syscall

la $a0,nl

li $v0,4

syscall

CONTINUARE\_CRIPTARE:

#citire mesaj clar de la tastatura

la $a0, mesaj\_clar #punem adresa din memorie a mesajului

li $a1, 99 #lungimea maxima a sirului de caractere

li $v0, 8 #codul pt READ STRING

syscall

li $t1,0 #i=0

lb $t2, mesaj\_clar($t1) #mesaj\_clar[0]

CRIPTARE:

lb $t3,nl

beq $t2,$t3, AFISARE\_MESAJ

li $t3, 0 #j=0

lb $t4, alfabet($t3) #alfabet[0]

caut: # $t2 e litera din mesaj si $t4 e litera din alfabet

beq $t2, $t4, gasit #cat timp nu am gasit litera in alfabet, caut csf si eu

addi $t3, $t3, 1 #j++

lb $t4, alfabet($t3) #alfabet[j]

j caut

gasit:

mul $t4, $t3, 4 #adresa din v

lw $t5, v($t4) #v[j]

lb $t6, alfabet($t5) #alfabet[v[j]]

sb $t6, mesaj\_clar($t1) #criptare

addi $t1, $t1, 1 #incrementam i

lb $t2, mesaj\_clar($t1) #luam urmatoarea litera din mesaj

j CRIPTARE

AFISARE\_MESAJ:

la $a0, mesaj\_clar #se muta in $a0 pentru a putea fi afisat

li $v0, 4

syscall

la $a0, nl #se afiseaza endl

li $v0, 4

syscall

#citire mesaj codificat de la tastatura

la $a0, mesaj\_criptat #punem adresa din mem a mesajului

li $a1, 99 #lungimea maxima a sirului de caractere

li $v0, 8 #codul pt READ STRING

syscall

li $t1,0 #i=0

lb $t2, mesaj\_criptat($t1) #mesaj\_criptat[0]

DECRIPTARE:

lb $t3,nl

beq $t2, $t3, AFISARE\_MESAJC #while mesaj[i]

li $t3, 0 #j=0; j este pt alfabet si e e pt mesaj

lb $t4, alfabet($t3) #alfabet[0]

caut\_alfabet: # $t2 e litera din mesaj si $t4 e litera din alfabet

beq $t4, $t2, caut\_in\_v #cat timp nu am gasit litera in alfabet

addi $t3, $t3, 1 #j++

lb $t4, alfabet($t3) #alfabet[j]

j caut\_alfabet

caut\_in\_v: #am gasit pozitia j in alfabet a literei i in mesaj

li $t5, 0 #k=0

lw $t6, v($t5) #v[0]

while: #caut j in v

beq $t6, $t3, decripteaza #while $t6!=$t3

addi $t5, $t5, 4 #k+=4; este folosit registrul $t5 pentru a sari locatii de memorie din 4 in 4

lw $t6, v($t5) #v[k]

j while

decripteaza:

divu $t5, $t5, 4

lb $t6, alfabet($t5) #alfabet[k]

sb $t6, mesaj\_criptat($t1) #punem litera corespunzatoare in mesaj

addi $t1, $t1, 1 #incrementam i

lb $t2, mesaj\_criptat($t1) #luam urmatoarea litera din mesaj

j DECRIPTARE

AFISARE\_MESAJC:

la $a0, mesaj\_criptat #afiseaza mesajul codificat

li $v0, 4

syscall

li $v0, 10

syscall

# Exemple:

* Pentru p=7 =>generatorul este g=3; mesaj\_clar=ACAD => BCBG; mesaj\_criptat=BCBG => ACAD
* Pentru p=6 afiseaza ca numarul nu este prim
* Pentru p=11=>generatorul este g=2; mesaj\_clar=FHAI => KHBD; mesaj\_criptat=BEJD => ACGI
* Pentru p=23 =>generatorul este g=5; mesaj\_clar=MIPS => SQTG; mesaj\_criptat=SQTG => MIPS