**Nome:** Felici Simone

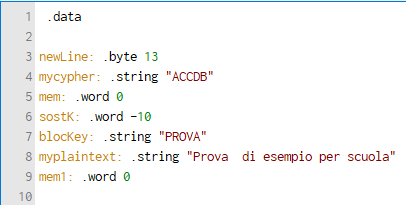
* **Matricola:** 7013439
* **Indirizzo mail:** simone.felici1@stud.unifi.it
* **Data consegna:** 14-08-2020

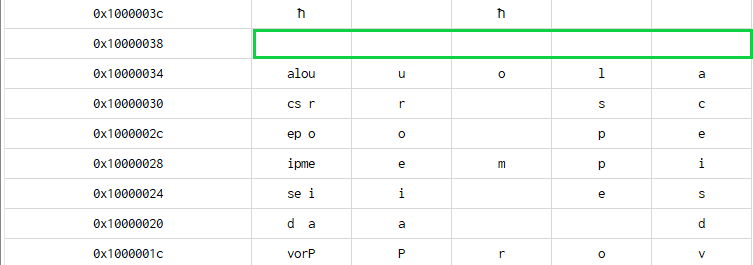
**Premesse:**

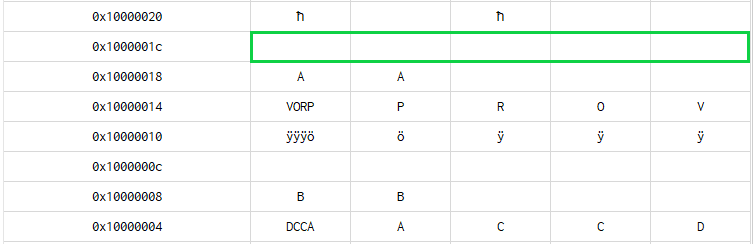
* Mi scuso se non progetto i commenti sono privi di accenti ma alcune lettere, dopo aver salvato, nel momento in cui riaprivo il programma queste erano state cambiate con il carattere “?”
* Nel .data ho inserito due label di nome “mem” e “mem1”, queste le ho usate per ovviare a due problemi:

1. Caso in cui il “mycypher” o il “myplaintext” non esitanti (stringa vuota);
2. Problema di avere il “mycypher” di una lunghezza non multipla di 4;

Il primo problema si risolve poiché nel momento in cui la stringa dovesse essere vuota il programma vede che il primo carattere è 0 ed esce.

Il secondo problema si risolve perché, come si vede dalle immagini sotto, la stringa “mycypher” ha sempre una word (4 byte) di tutti 0 al di sopra di lei (riquadrata in verde nell’ immagine).

Esempio con e senza “myplaintext”:



* **Algoritmo A (cifratura):**

L’ idea dell’algoritmo A è quella di avere nei registri sn i seguenti valori:

s2 = lettera dopo Z

s3 = lettera prima a

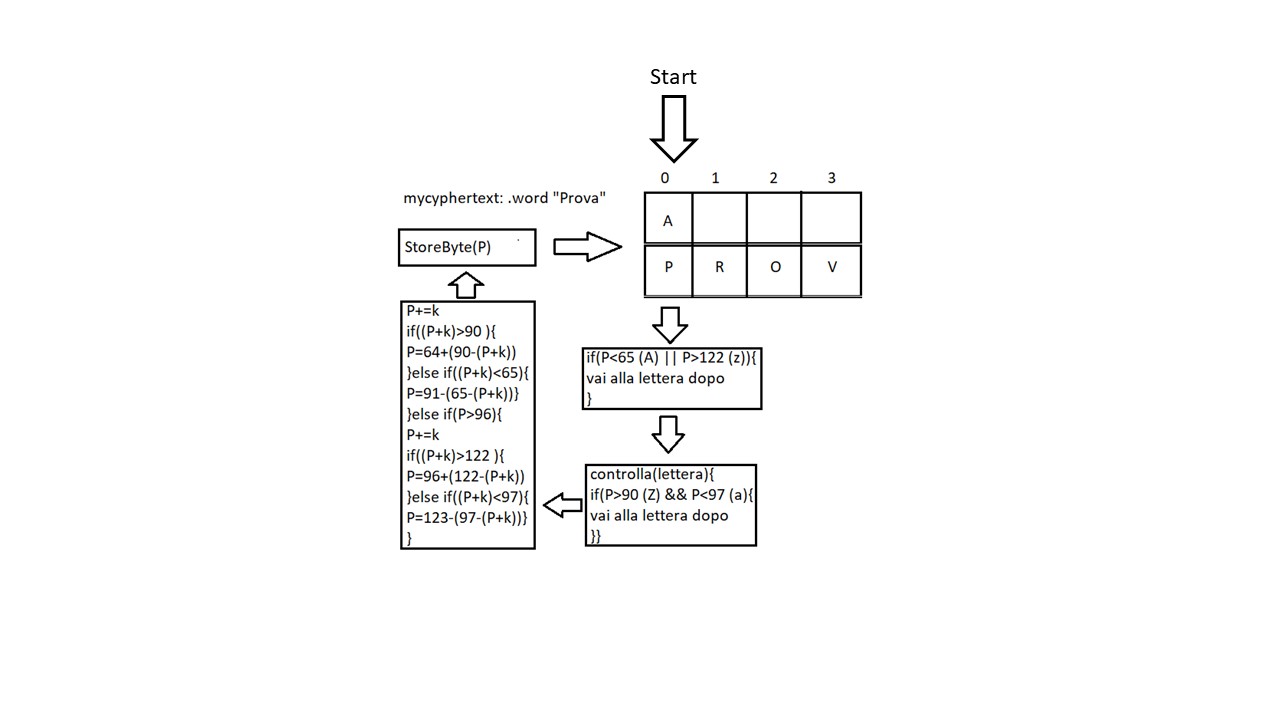
s4 = z

s5 = A

l’algoritmo controlla se:

1. La lettera è più piccola di A, se lo è salta alla lettera dopo
2. La lettera è più grande di Z, se lo è salta alla lettera dopo
3. La lettera è compresa tra il carattere dopo “Z” o il carattere prima di “a”

Se nessuna di queste condizioni si verifica, la lettera controlla se essa è maiuscola o minuscola, controllando prima se essa è più piccola del carattere dopo “Z” (maiuscola) o più grande del carattere prima di “a” (minuscola).

Se la lettera va oltre “z” o “Z” oppure prima di “a” o ”A”, mi calcolo di quanti caratteri ho scavalcato Z (per esempio) e, ripartendo dalla lettera prima di A, aggiungo di quanto ho scavalcato A, lo stesso procedimento è stato usato per le lettere minuscole.

* **Algoritmo A (decifratura):**

La decifratura dell‘ algoritmo controlla se:

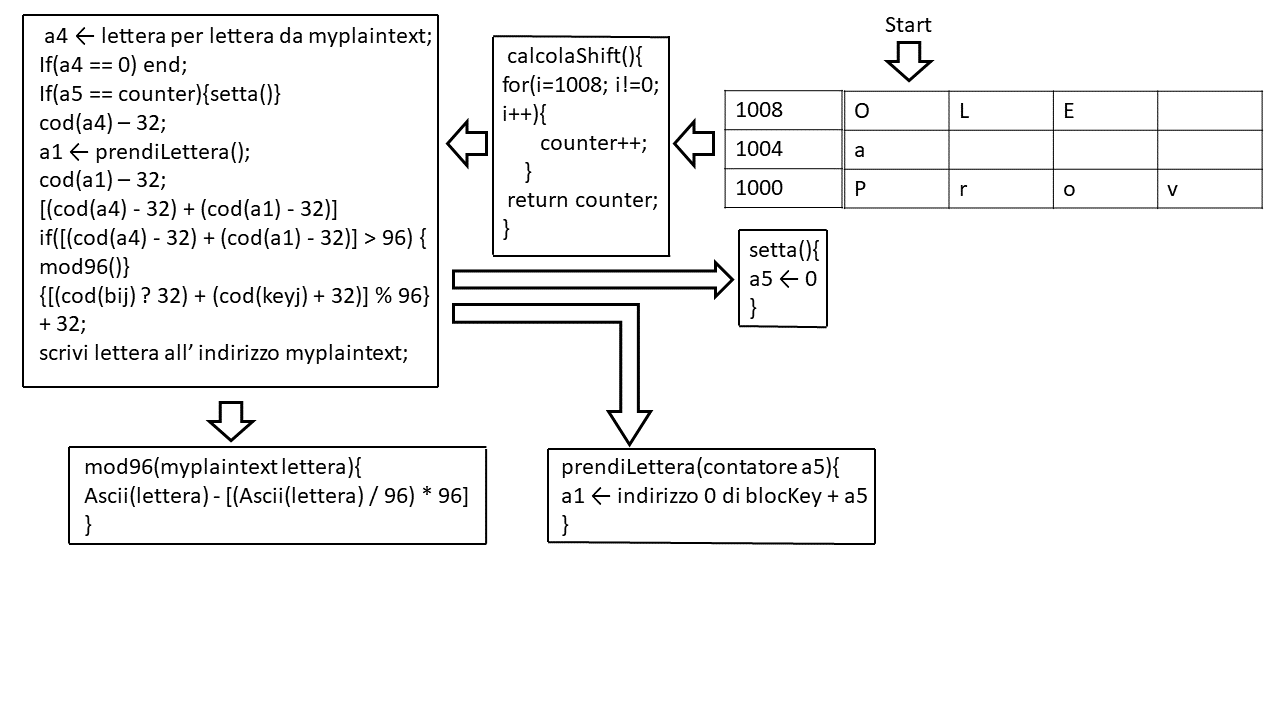
1. La lettera è più piccola di A, se lo è salta alla lettera dopo
2. La lettera è più grande di Z, se lo è salta alla lettera dopo
3. La lettera è compresa tra il carattere dopo “Z” o il carattere prima di “a”

Se nessuna di queste condizioni è verificata controlla se la lettera è maiuscola o minuscola, gli viene sottratto K e fatto eventualmente il modulo nel caso in cui la lettera vada prima di A (maiuscola o minuscola).

* **Algoritmo B (cifratura):**

L’ algoritmo B per prima cosa conta, con la funzione “calcolaShift”, quanto è lunga la stringa in “blocKey” in modo da avere una variabile(a5) che si accorge di quando deve ripartire dalla prima lettera della stringa blocKey, dopodiché con la funzione “prendiLettera”, grazie ad una variabile(a5) che viene ogni volta incrementata, so quale lettera di blocKey prendere. Avendo calcolato la lunghezza della stringa di “blocKey” (in a5), dovendo ogni volta prendere una lettera da “blocKey”, avrò un contatore che si incrementa ogni volta che ne scorro una, se questo diventa uguale ad a5 allora vuol dire che devo ripartire dalla prima lettera di a5. Una volta presa la lettera da “myplaintext” e da “blocKey” esegue:

[(cod(bij) – 32) + (cod(keyj) – 32)] % 96} + 32 ;



* **Algoritmo B (decifratura):**

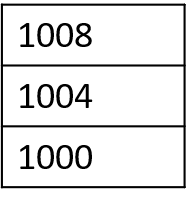
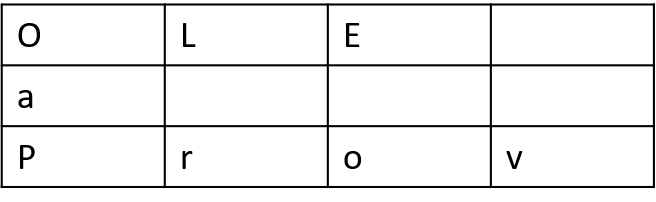
La decifratura viene fatto per prima cosa richiamando la funzione “blocKeyInPila”, con questa funzione vengono contate le lettere della stringa “myplaintext”, insieme ad essa si scorrono le lettere della stringa “blocKey”,

ogni volta che si trova una lettera in “myplaintext” si aggiunge nello stack la giusta lettera di “blocKey” nello stack, alla fine di questa funzione avrò nello stack tutte le lettere giuste per ogni lettera di “myplaintext” come mostrato nell’ esempio qui sotto:

blocKey = “OLE”;

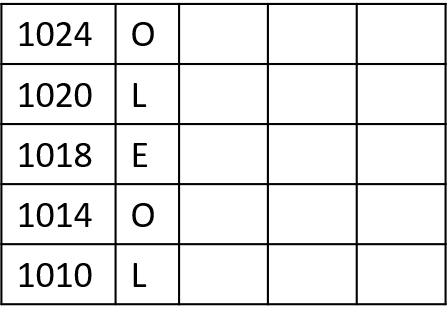
myplaintext = “Prova”;

**Memoria**:



**Stack**:

Per ogni lettera di “myplaintext” vengono aggiungi nello stack (in ordine) le giuste lettere dal “blocKey”.

****

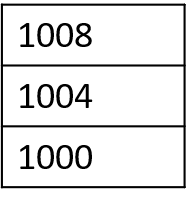
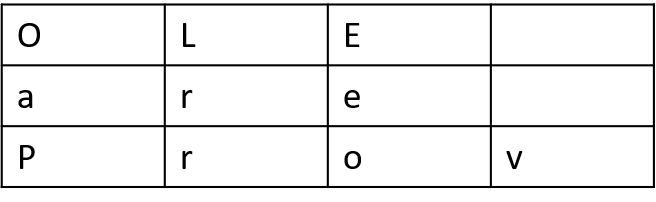
**sp**

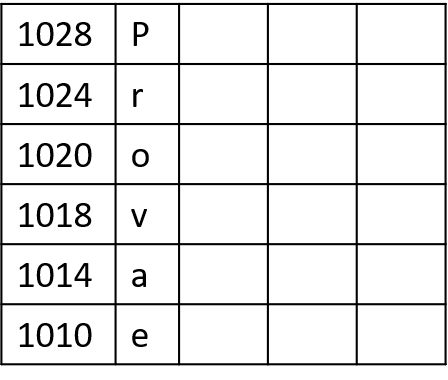
Infine viene chiamata la funzione “decifra” dove viene fatto:

[(cod(bij) + 32) - (cod(keyj) + 32)] % 127} + 32;

* **Algoritmo C (cifratura):**

Per l’algoritmo C ho deciso di usare un array. Questo algoritmo prende le lettere dalla stringa “myplaintext” che vengono passate ad una funzione (check), essa controlla nello stack se questa è stata già letta, ovvero se è già presente, nel caso in cui non fosse stata letta la aggiunge allo stack:

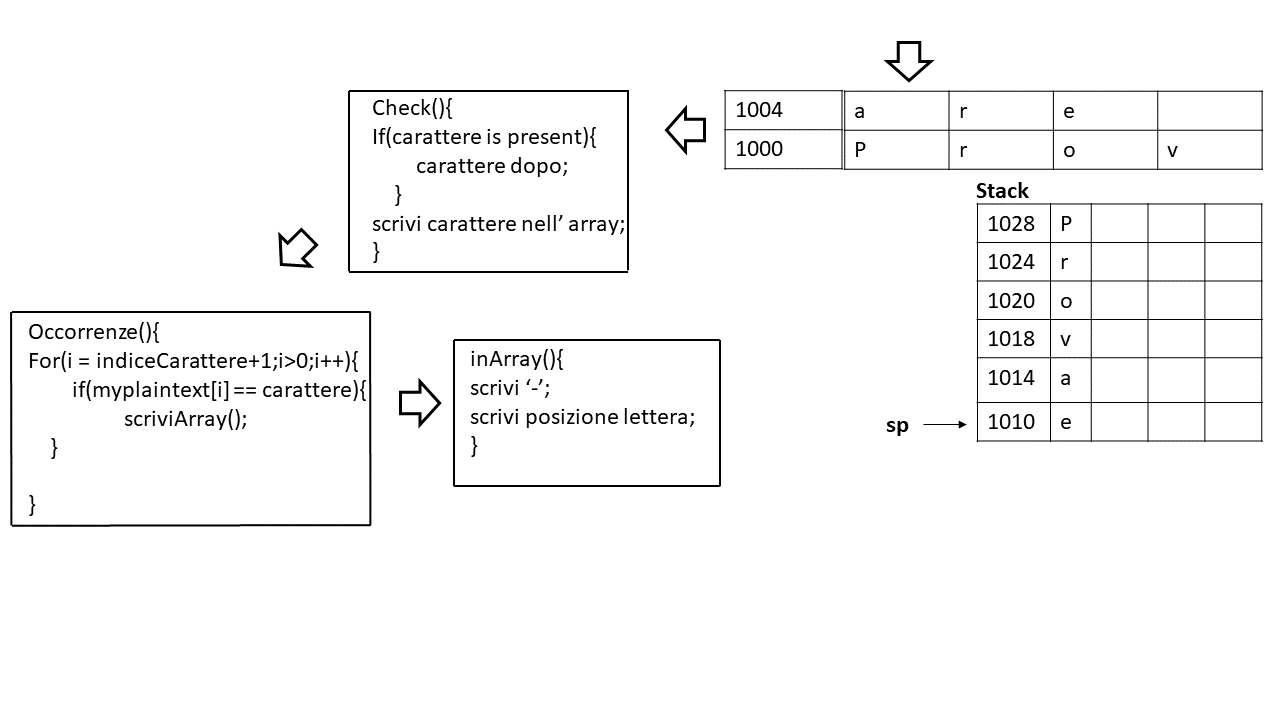




Quando andiamo a controlla il carattere ‘r’ nella posizione di memoria 1005, vediamo che questo è già presente nello stack, quindi vuol dire che la sua occorrenza à già stata calcolata.

sp

Dopodiché, a partire dal carattere i+1, si scorrono tutte le lette della stringa “myplaintext”, ogni volta che viene trovato un carattere uguale a quello che abbiamo preso dal “myplaintext” all’inizio, aggiungiamo nell’ array la posizione in codifica ASCII della parola e poi viene messo il carattere “-“.

Alla fine, dopo che abbiamo contato tutte le occorrenze, viene ricopiato tutto l’array nel “myplaintext” e poi l’array, che usavamo come appoggio, viene svuotato (ovvero ci vengono messi tutti 0).

* **Algoritmo C (decifratura):**

La decifratura avviene prendendo le lettere dal “myplaintext”, presa la lettera, andrà a scriverla nell’array nella posizione data dalla cifratura.

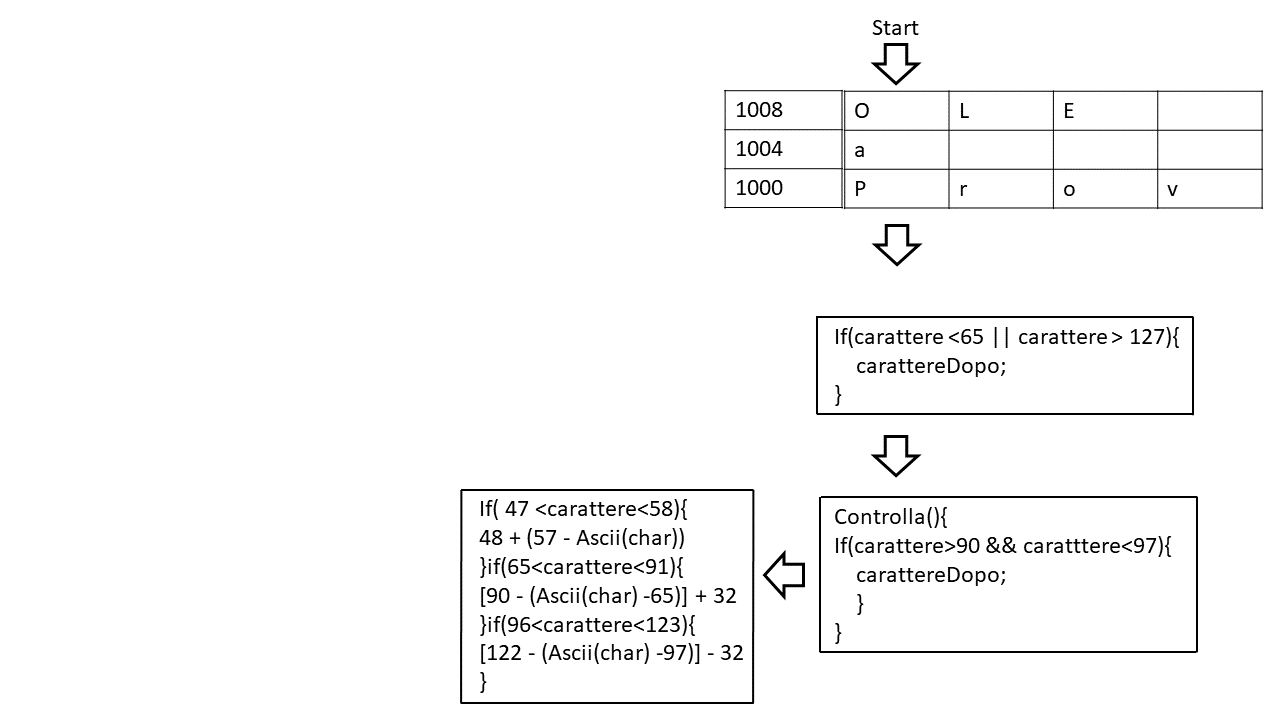
* **Algoritmo D (cifratura):**

L’ algoritmo D prende come sempre lettera per lettera dal “myplaintext”, una volta presa controlla se:

1. Non sia più piccola di 32
2. Non sia compresa tra 92 e 96 (compresi)
3. Non sia più grande di 122

Una volta fatti questi controlli, a sua volta ricontrolla se è una lettera maiuscola,

una lettera minuscola o un numero:

1. Nel caso in cui sia una lettera maiuscola viene calcolato il suo equivalente facendo cod(bj)- 65, il risultato viene poi aggiunto a 64, così da avere poi la sua lettera equivalente, a questa viene sommato 32 così da avere la lettera minuscola.
2. Nel caso in cui sia una lettera maiuscola viene calcolato il suo equivalente facendo cod(bj)- 97, il risultato viene poi aggiunto a 96, così da avere poi la sua lettera equivalente, a questa viene sottratto 32 così da avere la lettera maiuscola.
3. Il numero viene trovato nel controllo se la lettera è più piccola di 65, più grande di 47 e più piccola di 58, allora è un numero, per prima cosa sottraggo a 57 la codifica ASCII del numero che ho trovato, poi vado ad aggiungere ad a 48 il risultato ottenuto precedentemente, così da avere il numero in codifica ASCII

* **Algoritmo D (decifratura):**

L’ algoritmo D prende come sempre lettera per lettera dal “myplaintext”, una volta presa controlla se:

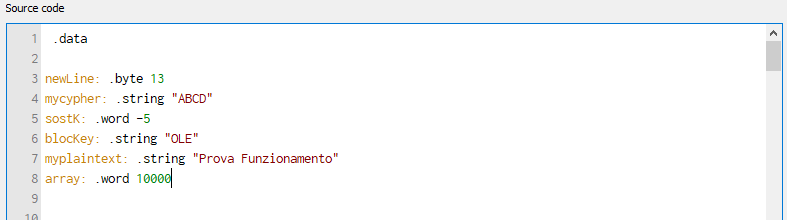
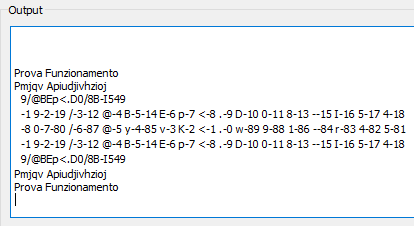
1. Non sia più piccola di 32
2. Non sia compresa tra 92 e 96 (compresi)
3. Non sia più grande di 122

Una volta fatti questi controlli, a sua volta ricontrolla se è una lettera maiuscola,

una lettera minuscola o un numero:

1. Nel caso in cui sia una lettera maiuscola viene calcolato il suo equivalente facendo cod(bj)- 65, il risultato viene poi aggiunto a 64, così da avere poi la sua lettera equivalente, a questa viene sommato 32 così da avere la lettera minuscola.
2. Nel caso in cui sia una lettera maiuscola viene calcolato il suo equivalente facendo cod(bj)- 97, il risultato viene poi aggiunto a 96, così da avere poi la sua lettera equivalente, a questa viene sottratto 32 così da avere la lettera maiuscola.
3. Il numero viene trovato nel controllo se la lettera è più piccola di 65, più grande di 47 e più piccola di 58, allora è un numero, per prima cosa sottraggo a 57 la codifica ASCII del numero che ho trovato, poi vado ad aggiungere ad a 48 il risultato ottenuto precedentemente, così da avere il numero in codifica ASCII

**Esempio corretto funzionamento:**

****

**Codice Assembly**

.data

newLine: .byte 13

mycypher: .string "ACCDB"

mem: .word 0

sostK: .word -10

blocKey: .string "PROVA"

myplaintext: .string ""

mem1: .word 0

array: .word 100000

.text

main:

la a0, myplaintext

li a7, 4

ecall

la a0, newLine

ecall

la a0, myplaintext

li s0, 32 #spazio

li s1, 45 # -

li s2, 91 #lettera dopo Z

li s3, 96 #lettera prima a

li s4, 122 #z

li s5, 65 #A

li s6, 127

la s10, newLine #per l' alg D senno non mi funzionava

la s11, mycypher

ApplicaAlgoritmi:

add t5, t6, s11

lb a6, 0(t5)

li a7, 65

addi t6, t6, 1

beq a6, a7, Acifra

addi a7, a7, 1

beq a6, a7, Bcifra

addi a7, a7, 1

beq a6, a7, Ccifra

addi a7, a7, 1

beq a6, a7, Dcifra

beq a6, zero, r

j ApplicaAlgoritmi

r:

addi t6, t6, -2

reverse:

add t5, t6, s11

lb a6, 0(t5)

li a7, 65

addi t6, t6, -1

beq a6, a7, Adecifra

addi a7, a7, 1

beq a6, a7, Bdecifra

addi a7, a7, 1

beq a6, a7, Cdecifra

addi a7, a7, 1

beq a6, a7, Ddecifra

beq a6, zero, output

j reverse

j output

Acifra:

lw a2, sostK

A:

add t1, t0, a0 #metto in a0 l'indirizzo di memoria a partire da cui e memorizzata la parola

lb a1, 0(t1) #metto in a1 la lettera

addi t0,t0,1 #incremento il puntatore per puntare alla lettera dopo

beq a1, zero, exit #se t2 ? 0 vuol dire che non ? finita la stringa (0=null)

beq a1, s0, A #se la lettera e uguale allo spazio esci

#l' idea e di vedere se inanzitutto la lettera e maiuscola in questo modo:

#in s2 ho il carattere dopo la Z in codifica ascii e quindi controlla se la lettera < 92

#in s3 ho il carattere prima di a in codifica ascii e quindi controllase lettera > 96

#la funzione controlla va a vedere se la lettera e <97 o >90

blt a1, s5, dopo # la lettera e piu piccola di A

bgt a1, s4, dopo #la lettera e piu grande di z

jal controlla

#Ora si va a controllare se la lettera e maiuscola o minucola in questo modo:

#in s2 ho il carattere dopo la Z in codifica ascii e quindi controlla se la lettera < 92

#in s3 ho il carattere prima di a in codifica ascii e quindi controllase lettera > 96

li s3, 96

li s2, 91

blt a1, s2, Maiuscola

bgt a1, s3, minuscola

j A

dopo:

j A

controlla:

addi s2, s2, -1

bgt a1, s2, comprensione # se la lettera e piu grande di Z

li s2, 91

li s3, 96

jr ra

comprensione:

li s3, 97

blt a1, s2, pross # se la lettere e piu piccola di a

li s2, 91

jr ra

pross:

li s3, 96

li s2, 91

j A

Maiuscola:

add a1, a1, a2 #aggiungi k alla lettera

addi s2, s2, -1 #in s2 ho Z

bgt a1, s2, moduloMaiuscoloO #se aggiungendo k la lettera diventa piu grande di Z devo fare

#in modo che si riparta da A

blt a1, s5, moduloMaiuscoloU #se aggiungendo k la lettera diventa piu piccola di A devo fare

#in modo che si riparta da Z

addi s2, s2, 1 # ripristino s2

sb a1, 0(t1) #scrivi la lettera cifrata in memoria

j A

moduloMaiuscoloO:

sub t3, a1, s2 #in t3 avro di quanti byte ha scavallato la Z

addi t3, t3, 64 #dato che la A e 65 allora aggiunto a 64 il numero di byte che ho scalcato Z

sb t3, 0(t1) #scrivi la lettera in memoria

j A

moduloMaiuscoloU:

sub t4, s5, a1 #in t3 avro di quanti byte ha scavallato la Z

li t3, 91 #dato che la Z e 90 allora aggiunto a 64 il numero di byte che ho scalcato Z

sub t3, t3, t4

sb t3, 0(t1) #scrivi la lettera in memoria

j A

minuscola:

add a1, a1, a2 #aggiungo k alla lettera

addi s3, s3, 1

bgt a1, s4, moduloMinuscoloO #se aggiungendo k la lettera diventa piu grande di z

#devo fare in modo che io riparta da a

blt a1, s3, moduloMinuscoloU #se ggiungendo k la lettera diventa piu piccola di a

#devo fare in modo che io riparta da Z

sb a1, 0(t1) #scrivo la lettera in memoria

j A

moduloMinuscoloO:

sub t4, a1, s4 #in t4 ho di quanti byte ho scvalcato la z

addi t3, t4, 96 # siccome la a parte da 65 allora aggiungo a partire da 64 quanti byte ho scavallato

sb t3, 0(t1) #scirvi in memoria la lettera

j A

moduloMinuscoloU:

sub t4, s3, a1 #in t4 ho di quanti byte ho scvalcato la a

li t3, 123

sub t3, t3, t4 # siccome la z parte da 122 allora sottraggo a partire da 123 quanti byte ho scavallato

sb t3, 0(t1) #scirvi in memoria la lettera

j A

exit:

li t0, 0

li t1, 0

li t2, 0

li t3, 0

li t4, 0

li s2, 92

li s3, 96

li s4, 122

li s5, 65

add s7, a0, zero

li a7, 4

ecall

la a0, newLine

ecall

add a0, s7, zero

li a7, 0

li a7, 0

li a1, 0

ecall

j ApplicaAlgoritmi

Adecifra:

lw a2, sostK

Ad:

add t1, t0, a0 #metto in a0 l'indirizzo di memoria a partire da cui e memorizzata la parola

lb a1, 0(t1) #metto in t2 la lettera

addi t0,t0,1 #incremento il puntatore per puntare alla lettera dopo

beq a1, zero, exitA #se t2 ? 0 vuol dire che non ? finita la stringa (0=null)

beq a1, s0, Ad #se la lettera ? uguale allo spazio esci

#l' idea e di vedere se inanzitutto la lettera e maiuscola in questo modo:

#in s2 ho il carattere dopo la Z in codifica ascii e quindi controlla se la lettera < 92

#in s3 ho il carattere prima di a in codifica ascii e quindi controllase lettera > 96

#la funzione controlla va a vedere se la lettera e <97 o >90

blt a1, s5, dopoD # la lettera e piu piccola di A

bgt a1, s4, dopoD #la lettera e piu grande di z

jal controllaD

#Ora si va a controllare se la lettera e maiuscola o minucola in questo modo:

#in s2 ho il carattere dopo la Z in codifica ascii e quindi controlla se la lettera < 92

#in s3 ho il carattere prima di a in codifica ascii e quindi controllase lettera > 96

blt a1, s2, MaiuscolaD

bgt a1, s3, minuscolaD

j Ad

dopoD:

j Ad

controllaD:

li s2, 90

bgt a1, s2, comprensioneD # se la lettera e piu grande di Z

li s2, 91

jr ra

comprensioneD:

li s3, 97

blt a1, s3, prossD # se la lettere e piu piccola di a

li s3, 96

li s2, 91

jr ra

prossD:

li s2, 91

li s3, 96

j Ad

MaiuscolaD:

addi s2, s2, -1

sub a1, a1, a2 #sottrai k alla lettera

blt a1 , s5, moduloMaiuscoloOD # guardo se la lettera e andata prima di A

bgt a1, s2, moduloMaiuscoloUD

sb a1, 0(t1) # se non e andata prima di A scrivila in memoria

j Ad

moduloMaiuscoloOD:

addi s2, s2, 1

sub t3, s5, a1 #in t3 ho di quando ho scavallato A facendo 65- lettera

sub t3, s2, t3 #in t3 adesso sottrarro il numero di byte partendo dalla lettea dopo Z (che e in s2)

#fino ad arrivare alla giusta lettera

sb t3, 0(t1) #scrivi in memoria la lettera

j Ad

moduloMaiuscoloUD:

sub t3, a1, s2

li t4, 64

add t3, t4, t3

sb t3, 0(t1)

j Ad

minuscolaD:

sub a1, a1, a2 #sottraggo k alla lettera

addi s3, s3, 1 # in s3 avro A in codifica Ascii

blt a1, s3, moduloMinuscoloOD #se la lettera e piu piccola di a allora devo ripartire da Z

bgt a1, s4, moduloMinuscoloUD

addi s3, s3, -1 #ripristina s2

sb a1, 0(t1) #scrivi in memoria

j Ad

moduloMinuscoloOD:

sub t4, s3, a1 #in t4 avro di quando ho scavalato a

addi s4, s4, 1 #in s4 avro il carattere dopo z

sub t4, s4, t4 #in t4 adesso sottraggo ad s4 quanti byte ho scavallcato a

sb t4, 0(t1) #scrivi in memoria

addi s3, s3, -1 #ripristina s2

addi s4, s4, -1 #ripristina s4

j Ad

moduloMinuscoloUD:

addi s3, s3, -1

sub t4, a1, s4

add t4, s3, t4

sb t4, 0(t1) #scrivi in memoria

addi s3, s3, 1

j Ad

exitA:

li t0, 0

li t1, 0

li t2, 0

li t3, 0

li t4, 0

li s2, 92

li s3, 96

li s4, 122

li s5, 65

add s7, a0, zero

li a7, 4

ecall

la a0, newLine

ecall

add a0, s7, zero

li a7, 0

li a7, 0

li a1, 0

ecall

j reverse

Bcifra:

la a2, blocKey

beq a2, a0, esci #se il blocKey non esiste

jal calcolaShift #la funzione calcolaShift va a contare la lunghezza della stringa in blocKey

add a3, a1, zero #in a3 ho la lunghezza della stringa in blocKey

li t0, 0

li t1, 0

li t2, 0

jal cifra #funzione per cifrare il messaggio

calcolaShift:

add t1, t0, a2 #mette in t1 l'indirizzo della stringa

lb t2, 0(t1) #Mette la lettera in t2

beq t2,zero,exit1 #Se trova t2==0(=fine stringa) esci

addi t0, t0, 1 #aumenta t0 di uno perche per contare lettera per

#lettera devo aumentare di un byte

j calcolaShift

exit1:

#in t0 ho il numero dello shift, ovvero del numero del blocco di lettere

#che devo cambiare con le lettere in blocKey

addi a1, t0, 0 #devo restituire a1 per le convenzioni, e avro il numero dello shift

jr ra

cifra:

add t1, t0, a0 #in t1 ho l' indirizzo di della parola da modificare

beq a5, a3, setta #se a5 e uguale al numero di volte che devo sciftare

#vuol dire che devo ripartire a puntare dalla prima lettera del blocKey

lb a4, 0(t1) # in a4 mette la lettera di myplaintext da modificare

beq a4, zero, esci #se la lettera e 0 esci

addi a4, a4, -32 # (cod(bij) - 32)

jal prendiLettera #prendi la giusta lettera da blocKey

addi a1, a1, -32 #(cod(keyj) - 32)

add a1, a4, a1 #lettera + lettera blocKey

li t3, 95

bgt a1, t3, mod96 #{[(cod(bij) ? 32) + (cod(keyj) + 32)] % 96} controlla se la lettera >96

addi a1, a1, 32 #{[(cod(bij) ? 32) + (cod(keyj) + 32)] % 96} + 32

sb a1, 0(t1) #sostituisci la lettera di myplaintext con la lettera t3+lettera del blocKey

addi a5, a5, 1

addi t0, t0, 1

j cifra

setta:

li a5, 0 #a5 e un flag che tiene conto se sono arrivato a leggere tutte le lettere del blocKey,

#in tal caso devo ripartire da capo

j cifra

prossimaLettera:

addi t0, t0, 1

j cifra

prendiLettera:

add t3, a5, a2

lb a1, 0(t3)#in a1 metto la lettera del blocKey

li t3, 0

jr ra

mod96:

#Per fare il modulo faccio

#Ascii(lettera) - [(Ascii(lettera) / 96)] prendendo pero solo la parte intera della divisione

#poi la moltiplico per 96, infine sottraggo alla lettera il risultato dell' operazione precedentemente

#citata cosi da avere il resto.

li t3, 96

div t4, a1, t3 #(Ascii(lettera) / 96)

mul t4, t4, t3 #(Ascii(lettera) / 96) \* 96

sub a1, a1, t4 ##Ascii(lettera) - [(Ascii(lettera) / 96) \* 96]

addi a1, a1, 32

sb a1, 0(t1) #sostituisci la lettera di myplaintext con la lettera t3+lettera del blocKey

addi t0, t0, 1

addi a5, a5, 1

li t3, 0

j cifra

esci:

li a1, 0

li a4, 0

li a5, 0

li t0, 0

li t1, 0

li t2, 0

li t3, 0

li t4, 0

add s7, a0, zero

li a7, 4

ecall

la a0, newLine

ecall

add a0, s7, zero

li s7 0

j ApplicaAlgoritmi

Bdecifra:

la a2, blocKey

beq a2, a0, fine # se non esiste il blocKey

#la funzione blocKey conta la stringa in myplaintext e, per ogni lettera, va ad aggiungere nello stack

#la giusta lettera del blocKey, cos? a fine funzione avr? nello stack tutte le lettere che mi servono

#per la stringa nel myplaintext.

jal blocKeyInPila

li t0, 0

li t1, 0

li t2, 0

li s6, 127

jal decifra

blocKeyInPila:

add t1, t0, a2 #mette in t1 l'indirizzo della stringa blocKey

lb t2, 0(t1) #Mette la lettera in t2

add t3, t4, a0 #metto in t3 l' indirizzo della parola in input palintext

lb a3, 0(t3) #metto in a3 la lettera della parola in myplaintext

beq t2, zero, riparti #se arrivo in fondo al blocKey riparti

beq a3, zero, exit1D # finisco quando arrivo alla fine della stringa nel myplaintext

addi sp, sp, -4

sb t2, 0(sp)

addi t0, t0, 1

addi t4, t4, 1

j blocKeyInPila

#in t0 ho la lunghezza della stringa in blocKey

exit1D:

li t0, 0

li t1, 0

li t4, 0

li s3, 0

addi t3, t3, -1 # sottraggo 1 perche in t3 ho quanto e lunga la stringa nel myplaintext

# ma l' ultimo carattere trovato era 0, quindi sottraggo 1 per puntare all' ultima lettera

add a1, t3, zero

li t3, 0

jr ra

riparti:

sub t0, t0, t0

j blocKeyInPila

next:

addi t4, t4, 1

j blocKeyInPila

decifra:

add t0, a1, zero

lb t2, 0(t0) #in t2 ho le lettere cifrate a partire dal fondo

beq t2, zero, fine

lb t1, 0(sp)

addi t2, t2, 32 #(cod(bj)+32

addi t1, t1, 32 #cod(key)+32

sub t3, t2, t1 #(cod(bij) + 32) - (cod(key) + 32)

blt t3, zero, mod #se (cod(bij) + 32) - (cod(key) + 32) < 0 si fa il modulo

addi t3, t3, 32 #(cod(bij) + 32) - (cod(key) + 32)mod 32)+32

sb t3, 0(t0)

addi a1, a1, -1

addi sp, sp, 4

j decifra

prossima:

addi a1, a1, -1

j decifra

mod:

#Per fare il modulo sottraggo a partire da 127 di quanto sono andato prima di 0, ed esempio se

#avessi -8 faccio, 127-18.

addi s6, s6, 1

add t3, s6, t3

sb t3, 0(t0)

addi s6, s6, -1

addi a1, a1, -1

addi sp, sp, 4

j decifra

fine:

li t0, 0

li t1, 0

li t2, 0

li t3, 0

li t4, 0

li a1, 0

add s7, a0, zero

li a7, 4

ecall

la a0, newLine

ecall

li a7, 0

add a0, s7, zero

li a1, 0

j reverse

Ccifra:

lw a2, array #ho bisgono di un array

inizio:

add a4, a5, a0 #adesso a4 punter? alla stringa nel myplaintext

lb a1, 0(a4) #metto in a1 la letterra all' indirizzo di a4

beq a1, zero, end #se la lettera ? 0 finisce l' algoritmo

#la funzione check va a controllare nella prila se la lettera che sto controllando fosse gi? stata

#letta, il cio viene fatto andando a confrontare la lettera con le altre nello stack, se la lettera

#non fosse presenta viene inserita nello stack.

jal check

sw a1, 0(sp) # metto la lettera nella pila perche non l'ho letta

sb a1, 0(a2) # carico la parola nell' array

addi a2, a2, 1

j occorrenze

check:

add t0, t1, sp #t0 punter? adesso all' array

lw t2, 0(t0)

beq a1, t2, giaLetta

beq t2, zero, aggiungi

addi t1, t1, 4

j check

giaLetta:

#Nel caso in cui io avessi gi? letto il carattere incremento il puntatore alla stringa nel

#myplaintext e riparto dall' inizio con un altra lettera.

addi a5, a5, 1

li t0, 0

li t1, 0

li t2, 0

j inizio

aggiungi:

#Nel caso in cui io non avessi letto il carattere vado ad incrementare lo stack

#per poi eseguire una store del carattere.

addi sp, sp, -4

li t0, 0

li t1, 0

li t2, 0

jr ra

occorrenze:

add s7, a5, zero # in t4 ho la lettere a cui punta a5 prima di contare le occorrenze, me la salvo

#in un registro dato che poi andro a scorrere tutta la stringa nel myplaintext

ciclo:

add t0, a5, a0 #t0 puntera alla stringa nel myplaintext ma alla lettera dopo di quella che sto controllando

lb t1, 0(t0)

beq t1, zero fineLoop # se la lettera e 0 ho finito di contare le occorrenze

beq a1, t1, inArray #se la lettera di cui sto contando le occorrenze ? uguale ad una lettera della

#stringa nel myplaintext vado a metterla nell' array.

addi a5, a5, 1

j ciclo

fineLoop:

add a5, s7, zero

li s6, 0

addi a5, a5, 1

sb s0, 0(a2)

addi a2, a2, 1

j inizio

inArray:

#Per mettere la parola nell' array controllo se ho contato piu di 9 lettere, in tal caso devo

#usare un modulo diverso per separare le due cifre, altrimenti a partire da 0 in codifica ASCII

#aggiungo la cifra dell' occorrenza che ho trovato

li t2, 9

li t3, 48 #0 in codifica ascii

sb s1, 0(a2) #metto - nell' array

addi a2, a2, 1

addi a5, a5, 1

bgt a5, t2, calcola #se la posizione dell' occorrenza fosse >9 allora devo dividerla

add t3, t3, a5 #da 0 in codifica ascii aggiungo cifra dell' occorrenza

sb t3, 0(a2)

addi a2, a2, 1

j ciclo

calcola:

#Per calcolare divido per 10 la codifica ascii dell' occorrenza cosi da avere in un la decina e poi

#mi ricavo il resto per avere in un altro registro l'unit?, il resto viene ricavato cosi:

#resto = divisore - (quoziente \* divisore).

#Una volta 'spezzata' l'ccorrenza a partire da 0 in codifica ascii aggiungo la decina e l' unita

#cos? da poter scrivere la codifica Ascii della posizione dell' occorrenza

li t2, 10

li t3, 48 #0 in codifica ascii

add a6, a5, zero

calc:

#Al posto di t4 ho messo a5

add t4, zero, a5

div a5, a5, t2 # (divisore / dividendo )

mul t0, t2,a5 # (divisore \* quoziente)

sub t0, t4, t0 #in t5 ho il resto

addi sp, sp, -4 #scrivo i resti nello stack

sb t0, 0(sp)

beq a5, zero, scrivi

j calc

scrivi:

add a5, a6, zero

scr:

li t2, 31

lb t4, 0(sp)

bgt t4, t2, pulisci

sb zero, 0(sp)

addi sp, sp, 4

add a6, t3, t4 #da 0 in codifica ascii aggiungo la decina dell' occorrenza

sb a6, 0(a2)

addi a2, a2, 1

j scr

pulisci:

li t2, 0

li t3, 0

li t4, 0

li a6, 0

j ciclo

end:

li t1, 4

mul t0, a5, t1

sub sp, sp, t0

addi ra, a3, 0

addi a2, a2, -1

li t0, 0

li t1, 0

li t2, 0

li t3, 0

conta:

#conto la lunghezza dell'array poiche alla fine voglio scrivere 0 dopo l'ultimo carattere nell' array

lw t3, array

add t1, t0, t3

lw t2, 0(t1)

beq t2, zero, endC

addi t0, t0, 1

j conta

endC:

addi t1, t1, -1

li a6, 0

sb a6, 0(t1) #scrivo 0 dopo l'ultimo carattere nell' array per essere sicuro di avere 0

add a1, t0, zero

addi a1, a1, -1

sub a2, a2, a1

li t0, 0

li t1, 0

li t2, 0

li t3, 0

li t4, 0

li a1, 0

copia:

#adesso vado a copiare dalla locazione di memoria 'array' nel myplaintext

la a1, myplaintext

add t3, t4, a1

add t1, t0, a2

lb t2, 0(t1)

beq t2, zero, avanti

sb t2, 0(t3)

li t2, 0

sb t2, 0(t1) #cosi cancello l' array

addi t4, t4, 1

addi t0, t0, 1

j copia

avanti:

li t0, 0

add t0, a0, zero

li a7, 4

ecall

la a0, newLine

ecall

add a0, t0, zero

li t0, 0

li t1, 0

li t2, 0

li t3, 0

li a1, 0

li a5, 0

li a4, 0

li a3, 0

li s7, 0

j ApplicaAlgoritmi

Cdecifra:

addi s5, s5, -1 #mi serve la lettera prima di A

li a5, 100000

lw a2, array

add a2, a2, a5 #ho dovuto aggiungere senno avendo nel plaintext la lettera 'C' piu di una volta,

#nell' ultima chiamata (ultima volta in cui deve decifrare con l' algoritmo C), la chiamata qui

#sopra (lw a2, array) carica in a2 l' indirizzo 0, che andava quindi a sovrascrivere su parti di

#memoria gia occupate dal programma

li a5, 0

Cdec:

add t0, a4, a0 #a5 puntera alla stringa nel myplaintext

lb t2, 0(t0)

beq t2, zero, final

li a5, 9

j scorri

j Cdec

scorri:

#Qui vado a scorrere la parola nel myplaintext, andando a vedere le occorrenze della lettera che ho

#in t2, andando a scrivere t2 in un array di appoggio nella posizione dell' occorrenza che vado

#a leggere.

addi a4, a4, 1

add t3, a4, a0 #t1, punta alla stringa nel myplaintext

lb t4, 0(t3)

beq t4, s0, ex #se a forza di scorrere trovo lo spazio

beq t4, zero, ex #se trovo 0 esco

beq t4, s1, nextChar #se trovo il carattere - scorro alla lettera dopo

li t0, -1

li t1, 48

lb a1, 1(t3)

beq a1, s1, store #se la lettera dopo uguale a - la scrivo nell array

beq a1, s0, store #se la lettera ? uguale allo spazio

beq a1, zero, store

#es 10= 49 e 48, faccio 10=(49-48)\*10 + (48-48)

li t0, 0

li t1, 48

li a3, 10

#add t5, a4, zero

scorriOccorrenza:

add t3, a4, a0

lb t4, 0(t3)

beq t4, s1, wri #se la lettera ? uguale a - la scrivo nell array

beq t4, s0, wri #se la lettera ? uguale allo spazio

beq t4, zero, wri

sub t4, t4, t1 #mi calcolo dalla codifica ascii alla posizione dell' occorrenza

add t0, t4, t0

mul t0, t0, a3 #moltiplico la lettera per 10 cosi poi posso aggiungerci il secondo byte

addi a4, a4, 1

j scorriOccorrenza

wri:

#add a4, t5, zero

li t1, 10

div t0, t0, t1

add t0, t0, a2 #qui devo averre a6

addi t0, t0, -1

sb t2, 0(t0)

addi a4, a4, -1

li t0,0

li t1,0

li a3, 0

li t5, 0

j scorri

store:

sub t4, t4, t1 #mi calcolo la posizione della lettera facendo : Ascii(lettera)- 48

add t4, t0, t4 # poi a partire da 0 ci aggiungo il risultato del calcolo precedente

add t4, t4, a2 #adesso t4 punta alla giusta posizione dove scrivere la lettera nell' array

sb t2, 0(t4)

addi a1, a1, 1 #mi conto quante lettere ho controllato

j scorri

nextChar:

j scorri

ex:

li t3, 0

li t4, 0

li t5, 0

addi a4, a4, 1

j Cdec

final:

addi s5, s5, 1

li t0, 0

li t1, 0

li t2, 0

li t3,0

li t4, 0

li a1, 0

li a4, 0

li a5, 0

#alla fine vado a copiare la codifica, che si trova nella locazione di memoria 'array', nel

#myplaintext, poi svuoto l' array

svuota:

#a questo punto metto a 0 tutto quello che c'? nel myplaintext

add t1, t0, a0 #t1 punter? al myplaintext

lb t2, 0(t1)

beq t2, zero, copiaDaArray #quando ho finito di 'svuotare' il myplaintext

li t2, 0

sb t2, 0(t1)

addi t0, t0, 1

j svuota

copiaDaArray:

#adesso vado a copiare la codifica nel myplaintext

addi t0, t0, -1

sub t0, t0, t0

cop:

add t4, t3, a2

lb t2, 0(t4)

beq t2, zero, svuotaArray

add t1, t0, a0

sb t2, 0(t1)

addi t0, t0, 1

addi t3, t3, 1

j cop

svuotaArray:

#infine svuoto l' array dove avevo la codifica

li t0, 0

li t1, 0

sv:

add t1, t0, a2

lb t2, 0(t1)

beq t2, zero, pro

li t2, 0

sb t2 , 0(t1)

addi t0, t0, 1

j sv

pro:

li t4, 0

add t4, a0, zero

li a7, 4

ecall

add a0, s10, zero

ecall

add a0, t4, zero

li t0, 0

li t1, 0

li t2, 0

li t3, 0

li t4, 0

li a2, 0

j reverse

Dcifra:

D:

add t1, t0, a0

lb t2, 0(t1)

beq t2, zero, endAl

blt t2, s0, nan

#l' idea e di vedere se inanzitutto la lettera e maiuscola in questo modo:

#in s2 ho il carattere dopo la Z in codifica ascii e quindi controlla se la lettera < 92

#in s3 ho il carattere prima di a in codifica ascii e quindi controllase lettera > 96

blt t2, s5, prox # la lettera e piu piccola di A

bgt t2, s4, prox #la lettera e piu grande di z

jal contr

#l' idea e di vedere se inanzitutto la lettera e maiuscola in questo modo:

#in s2 ho il carattere dopo la Z in codifica ascii e quindi controlla se la lettera < 92

#in s3 ho il carattere prima di a in codifica ascii e quindi controllase lettera > 96

addi s3, s3, -1

addi s2, s2, 1

blt t2, s2, MaiuD

bgt t2, s3, minuD

j D

nan:

addi t0, t0, 1

j D

prox:

li s9,47

bgt t2, s9, num #la lettera e piu piccola di A ma se e piu grande di 47 controllo se e un numero

addi t0, t0, 1

j D

contr:

li s2, 90

bgt t2, s2, comp # se la lettera e piu grande di Z

li s2, 91

li s3, 96

jr ra

comp:

li s3, 97

blt t2, s3, pros # se la lettere e piu piccola di a

li s2, 91

li s3,96

jr ra

pros:

li s2, 91

li s3,96

addi t0, t0, 1

j D

num:

li s9, 58

blt t2, s9, numD #se e anche piu piccola di 58 allora e un numero iin ascii code

addi t0, t0, 1

j D

numD:

li s9, 57

sub t3, s9 , t2 #in t3 ho il risultatp ma come numero ed io devo averlo come ascii

li s9, 48

add t4, s9, t3

sb t4, 0(t1)

li t3, 0

li t4, 0

li s9, 0

addi t0, t0, 1

j D

MaiuD:

li s7, 90

li s8, 65

sub t3, t2, s8 #lettera-65 cosi so quanto devo aggiungere a partire da Z per arrivare al suo inverso

sub t4, s7, t3

addi t4, t4 32 #tra maiuscole e minuscole c'e 32

sb t4, 0(t1)

addi t0, t0, 1

li t3, 0

li t4, 0

li s7, 0

li s8, 0

j D

minuD:

li s7, 122

li s8, 97

li s9, 32

sub t3, t2, s8 #lettera-97 cosi so quanto devo aggiungere a partire da z per arrivare al suo inverso

sub t4, s7, t3

sub t4, t4, s9 #tra maiuscole e minuscole c'e 32

sb t4, 0(t1)

addi t0, t0, 1

li t4,0

li t3, 0

li s7, 0

li s8, 0

li s9, 0

j D

endAl:

li t0, 0

li t1, 0

li t2, 0

li t3, 0

li t4, 0

lw ra, 0(sp)

addi sp, sp, 4

add t4, a0, zero

li a7, 4

ecall

add a0, s10, zero

ecall

add a0, t4, zero

li a7, 0

li t4, 0

j ApplicaAlgoritmi

Ddecifra:

addi sp, sp, -4

sb ra, 0(sp)

Dd:

add t1, t0, a0

lb t2, 0(t1)

beq t2, zero, endAlD

blt t2, s0, nanD

#l' idea e di vedere se inanzitutto la lettera e maiuscola in questo modo:

#in s2 ho il carattere dopo la Z in codifica ascii e quindi controlla se la lettera < 92

#in s3 ho il carattere prima di a in codifica ascii e quindi controllase lettera > 96

blt t2, s5, proxD # la lettera e piu piccola di A

bgt t2, s4, proxD #la lettera e piu grande di z

jal contrD

#l' idea e di vedere se inanzitutto la lettera e maiuscola in questo modo:

#in s2 ho il carattere dopo la Z in codifica ascii e quindi controlla se la lettera < 92

#in s3 ho il carattere prima di a in codifica ascii e quindi controllase lettera > 96

addi s3, s3, -1

addi s2, s2, 1

blt t2, s2, MaiuDec

bgt t2, s3, minuDec

j Dd

proxD:

li s9,47

bgt t2, s9, numeD #la lettera ? piu piccola di A ma se e piu grande di 47 controllo se e un numero

addi t0, t0, 1

j Dd

contrD:

addi s2, s2, -1

bgt t2, s2, compD # se la lettera e piu grande di Z

jr ra

compD:

li s3, 97

blt t2, s3, prosD # se la lettere e piu piccola di a

jr ra

prosD:

addi s3, s3, -1

addi s2, s2, 1

addi t0, t0, 1

j Dd

numeD:

li s9, 58

blt t2, s9, numDec #se e anche piu piccola di 58 allora e un numero iin ascii code

addi t0, t0, 1

j Dd

numDec:

li s9, 57

sub t3, s9 , t2 #in t3 ho il risultatp ma come numero ed io devo averlo come ascii

li s9, 48

add t4, s9, t3

sb t4, 0(t1)

li t3, 0

li t4, 0

li s9, 0

addi t0, t0, 1

j Dd

MaiuDec:

li s7, 90

li s8, 65

sub t3, t2, s8 #lettera-65 cosi so quanto devo aggiungere a partire da Z per arrivare al suo inverso

sub t4, s7, t3

addi t4, t4 32 #tra maiuscole e minuscole c'e 32

sb t4, 0(t1)

addi t0, t0, 1

li t3, 0

li t4, 0

li s7, 0

li s8, 0

j Dd

minuDec:

li s7, 122

li s8, 97

li s9, 32

sub t3, t2, s8 #lettera-97 cosi so quanto devo aggiungere a partire da z per arrivare al suo inverso

sub t4, s7, t3

sub t4, t4 s9 #tra maiuscole e minuscole c'e 32

sb t4, 0(t1)

addi t0, t0, 1

li t4,0

li t3, 0

li s7, 0

li s8, 0

li s9, 0

j Dd

endAlD:

li t0, 0

li t1, 0

li t2, 0

add t4, a0, zero

li a7, 4

ecall

add a0, s10, zero

ecall

add a0, t4, zero

li t4, 0

lw ra, 0(sp)

addi sp, sp, 4

j reverse

nanD:

addi t0, t0, 1

j Dd

output: