Progetto Assembly RISC-V per il Corso di Architetture degli Elaboratori – A.A. 2021/2022 –

Messaggi in Codice

Versione 1 del documento, aggiornata il 21/3/2022.

CHANGELOG: Aggiornamento progetto per evitare l'utilizzo di caratteri ASCII più piccoli di 32 o più grandi di 127, che danno risultati diversi in base alla versione di RIPES utilizzata.

Crittografia e Cifratura

I codici sono un modo per alterare un messaggio testuale per nasconderne il significato originale. Questo permette di crittografare stringhe e dati generici, per nasconderne il significato a terzi che intercettano il messaggio cifrato, e che non hanno la chiave per decodificarlo. Uno degli esempi più famosi è quello della *macchina Enigma* (suggerisco la visione di The Imitation Game, per chi non l'avesse visto!), un dispositivo elettromeccanico per cifrare e decifrare messaggi che fu ampiamente utilizzata dal servizio delle forze armate tedesche durante il periodo nazista e della seconda guerra mondiale. In molti casi e di solito richiedono una **parola chiave** per essere interpretati. I cifrari sono algoritmi applicati a un messaggio che nascondono o criptano le informazioni trasmesse. Questi cifrari vengono quindi **invertiti** per tradurre o decifrare il messaggio.

Macchina Enigma versione G

Ad esempio, il testo in chiaro (plain text) AMO AssEMbLY potrebbe essere modificato tramite una funzione di cifratura che sostituisce ogni lettera con la successiva nell'alfabeto, ottenendo la parola criptata (cyphertext) BNP BttFNcMZ. Solo chi conosce una apposita funzione di de-cifratura riesce ad interpretare il testo. In questo caso, la funzione di de-cifratura sostituisce ciascuna lettera con quella che la precede nell'alfabeto, riottenendo AMO AssEMbLY.

Più rigorosamente, dato un plaintext pt, una funzione di cifratura fc, una funzione di decifratura fd, si ha

$$\text{cyphertext } ct = fc(pt), \qquad pt = fd(ct)$$

In generale, utilizzando una serie di n funzioni di cifratura $FC = \{fc_1, ..., fc_n\}$ e di decifratura $FD = \{fd_1, ..., fd_n\}$, dove *ogni fd_i decifra il messaggio critpato dalla funzione fc_i*, si ha che

$$pt = fd_n(fd_{n-1}(...fd_1(\, fc_1(fc_2(...fc_n(pt))))))$$

Ovvero che applicando **in sequenza le funzioni di cifratura** fc_n, fc_{n-1}, ... fc_1, si riottiene il plaintext applicando le **funzioni di decifratura in ordine inverso** fd_1, fd_2, ... fd_n

Alcuni Codici di Cifratura

Cifrario a Sostitizione / Cifrario di Cesare

(da Wikipedia) Il cifrario di Cesare è uno dei più antichi algoritmi crittografici di cui si abbia traccia storica. È un cifrario a sostituzione monoalfabetica in cui **ogni lettera del testo in chiaro è sostituita nel testo cifrato dalla lettera che si trova un certo numero di posizioni dopo nell'alfabeto**. Questi tipi di cifrari sono detti anche cifrari a sostituzione o cifrari a scorrimento a causa del loro modo di operare: la sostituzione avviene lettera per lettera, scorrendo il testo dall'inizio alla fine.

Una possibile traduzione potrebbe essere la seguente: il codice ASCII standard su 8 bit di ciascun carattere del messaggio di testo viene modificato sommandoci una costante intera *K* in modulo. Ovvero, se *cod(X)* è la codifica ascii standard decimale di un carattere *X* del messaggio, con $32 \leq cod(X) \leq 127$, la cifratura di *X* corrisponderà a: $32 + (cod(X)+K) \bmod 96$. Ai fini del nostro progetto, la sostituzione viene eseguita solo per lettere maiuscole e minuscole (A-Z e a-z), lasciando gli altri simboli (numeri, spazi, caratteri speciali) invariati. Maiuscole e minuscole vengono preservate.

Esempio

Si veda la pagina precedente con il plaintext AMO AssEMbLY (K=1)

Cifrario a Blocchi

(Da Wikipedia) Un algoritmo di cifratura a blocchi è composto da due parti, una che cifra,utilizzando *m* caratteri per il blocco da cifrare e *k* caratteri per la chiave *key* da utilizzare durante la cifratura, restituendo *m* caratteri di uscita per il cyphertext. Una versione semplice per la cifratura a blocchi si può intendere come segue: **la parola viene partizionata in nb blocchi**, ottenuti come nb = m/k arrotondato all'intero superiore. Ogni blocco in $B = \{b_1, b_2, \ldots b_{nb}\}$ contiene al più k elementi consecutivi della stringa da cifrare. **Ogni elemento di ciascun blocco viene cifrato sommando la codifica ASCII di un carattere della chiave alla codicica ASCII del carattere del blocco** come segue

$$\text{For each } b_i \text{ in } B\ (1 \leq i \leq nb),\ cb_i = 32 + (cod(b_{ij}) + cod(key_j)) \bmod 96,\ 1 \leq j \leq k,$$

ottenendo il cyphertext $ct = \{cb_1, cb_2, \ldots cb_{nb}\}$, composto da nb blocchi cifrati. *Tutti i caratteri della stringa di partenza vengono cifrati con questa codifica (non solo le lettere, come nell'algoritmo precedente).*

Esempio

pt = LAUREATO, key = OLE

Si calcola Cod(O) = 79, Cod(L) = 76, Cod(E) = 69 andando a consultare la tabella ASCII e poi da ciascun Cod(ct) si trova il carattere corrispondente nella parte di tabella ASCII per ottenere il cyptertext ct.

Pt	**L**	**A**	**U**	**R**	**E**	**A**	**T**	**O**
Cod(pt)	76	65	85	82	69	65	84	79
Key	O	L	E	O	L	E	O	L
Cod(key)	79	76	69	79	76	69	79	76
Cod(pt) + Cod(key)	155	141	154	161	145	134	163	155
Cod(ct)	91	77	90	97	81	70	99	91
ct	[	M	Z	a	Q	F	c	[

Cifratura Occorrenze

A partire dal primo carattere del plaintext (alla posizione 1), **il messaggio viene cifrato come una sequenza di stringhe separate da esattamente 1 spazio** (ASCII 32) in cui ciascuna stringa ha la forma "*x*-p_1-...-p_k", dove *x* è la prima occorrenza di ciascun carattere presente nel messaggio, p_1...p_k sono le k posizioni in cui il carattere *x* appare nel messaggio (con p_1<...<p_k), ed in cui ciascuna posizione è preceduta dal carattere separatore '-' (per distinguere gli elementi della sequenza delle posizioni). Note:

- non c'è un ordine prestabilito per le lettere una volta codificata la stringa
- la codifica usa due separatori: lo spazio " " (ASCII 32) ed il trattino "-" (ASCII 45). Ciò che sta tra due trattini deve essere sempre un numero che indica l'occorrenza di una lettera nella stringa di base, mentre ciò che segue lo spazio è sempre il carattere di riferimento, eccetto il caso in cui il carattere di riferimento sia lo spazio in sè (si veda l'esempio sotto).
- L'occorrenza di un carattere nella stringa si esprime come numero su 1 byte. Questo non va interpretato come codifica ASCII ma come valore puro.

Esempio

Pt = "sempio di messaggio criptato -1"

La cifratura con questo algoritmo produrrà un cyphertext ct = "e-2-12 s-1-13-14 m-3-11 p-4-24 i-5-9-18-23 o-6-19-28 -7-10-20-29 d-8 a-15-26 g-16-17 c-21 r-22 t-25-27 --30 1-31".

- Nella stringa " -7-10-20-29" il carattere in codifica è lo spazio (' ', ASCII decimale 32), che appare nelle posizioni 7, 10, 20 e 29 del messaggio.
- Nella stringa "--30" il carattere in codifica è '-' (il secondo carattere '-' è il carattere separatore fra gli elementi della sequenza), che appare nella posizione 30 del messaggio.
- Nella stringa "1-31" il carattere in codifica è '1', che appare nella posizione 31 del messaggio.
- In memoria, la prima parte del cyphertext "e-2-12 " dovra apparire su 6 byte come cod(e), cod(-), 2, cod(-), 12, cod( ) -> 101, 45, 2, 45, 12, 32

Dizionario

Ogni possibile simbolo ASCII viene mappato con un altro simbolo ASCII secondo una certa funzione, che riportiamo di seguito definita per casi. In questo caso, si richiede che i singoli caratteri ASCII c_i della stringa da codificare appartengano alla codifica ridotta ($32 \leq cod(c_i) \leq 127$)

- Se il carattere *ci* è una lettera minuscola (min), viene sostituito con l'equivalente maiuscolo dell'alfabeto in ordine inverso es. Z = ct(a), A = ct(z).
- Se il carattere *ci* è una lettera maiuscola (mai), viene sostituito con l'equivalente minuscolo dell'alfabeto in ordine inverso es. z = ct(A), y = ct(B), a =ct(Z).
- Se il carattere *ci* è un numero (num), ct(ci) = 9 – num, cod(ct(ci) = ASCII(9) – ASCII(num)
- In tutti gli altri casi (sym), *ci* rimane invariato, ovvero ct(ci) = ci

Esempio

Pt = myStr0ng P4ssW_

Pt	m	y	S	t	r	0	n	g		P	4	s	s	W	_
Tipo ci	min	min	mai	min	min	num	min	min	sym	mai	num	min	min	mai	sym
ct	N	B	h	G	I	9	M	T		k	5	H	H	d	_

Inversione

Il cyphertext è rappresentato dalla stringa invertita. Si noti come nel caso di parole palindrome, pt = ct.

Esempio

pt = BUONANOTTE, ct = ETTONANOUB

Progetto Assembly

Costruendo su quanto detto sopra, il progetto di AE 21/22 riguarda la progettazione e la scrittura di un codice assembly RISC-V che simuli alcune funzioni di cifratura e decifratura di un messaggio di testo, interpretato come sequenza di caratteri ASCII.

In particolare il programma dovrà consentire di **cifrare** e **decifrare** un messaggio di testo (plaintext) fornito dall'utente come variabile *myplaintext* di tipo stringa (.string in RIPES). Diverse funzioni di cifratura, e le conseguenti di decifratura, dovranno essere implementate come di seguito.

A. Cifrario a Sostituizione, configurabile tramite variabile sostK, che indica lo shift alfabetico.
B. Cifrario a blocchi, con la chiave che è una stringa da considerare come variabile blocKey
C. Cifratura Occorrenze
D. Dizionario
E. Inversione

Oltre alla variabile myplaintext (**dimensione massima 200 caratteri**), che contiene simboli il cui codice ASCII è compreso tra 32 e 127 (estremi inclusi) il programma richiede un input addizionale che specifica le *modalità di applicazione delle cifrature.*

Tale variabile *mycypher* è una stringa S = "$S_1 .. S_n$"in cui ciascun carattere S_i (con $1 \leq i \leq n$) corrisponde ad uno fra i caratteri 'A', 'B', 'C', 'D', 'E' ed identifica l' *i*-mo cifrario da applicare al messaggio. **L'ordine delle cifrature è quindi stabilito dall'ordine in cui appaiono i caratteri nella stringa.**

A titolo di esempio, si riportano alcuni possibili parole chiave: "C", oppure "AEC", oppure "DEDD", ... La cifratura del messaggio di testo con la parola chiave "AEC" determinerà l'applicazione dell'algoritmo A, poi dell'algoritmo E (sul messaggio già cifrato con A) ed infine dell'algoritmo C (sul messaggio già cifrato prima con A e poi con E).

Il programma dovrà considerare le variabili din ingresso (nel .data) myplaintext e mycypher, oltre ai vari parametri messaggio da cifrare, e produrre in output a video, i vari cypthertext ottenuti dopo l'applicazione di ciascun singolo passaggio di cifratura, separati da un newline. Allo stesso modo, si dovranno applicare le funzioni di decifratura *a partire dal messaggio precedentemente cifrato* in ordine inverso rispetto alle cifrature. **L'ultimo messaggio stampato a video dovrà corrispondere al plaintext di partenza**.

Si noti come il carattere "newline" abbia codice ASCII 10, e possa essere usato per separate gli output a console

Note e Modalità di Consegna

Note

- Seguire fedelmente tutte le specifiche dell'esercizio (incluse quelle relative ai nomi delle variabili e al formato del loro contenuto).
- Rendere il codice **modulare** utilizzando ove opportuno **chiamate a procedure** e **rispettando le convenzioni fra procedura chiamante/chiamata**. La modularità del codice ed il rispetto delle convenzioni saranno aspetti fondamentali per ottenere un'ottima valutazione del progetto. Si richiede in particolare di *implementare ogni cifrario (ciascun algoritmo A-B-C-D-E, e le loro inversioni per la decifratura) come una procedura separata.*
- Commentare in modo significativo (non commentare *addi s3, s3, 1* con "sommo uno ad s3"....).

Dettagli sulla Consegna

- Per sostenere l'esame è necessario consegnare preventivamente il codice e una relazione PDF sul progetto assegnato, oltre ad un breve video che spieghi il funzionamento del codice.
- Il codice consegnato deve essere funzionante sul simulatore RIPES, usato durante le lezioni.
- La scadenza esatta della consegna verrà resa nota di volta in volta, in base alle date dell'appello.
- Discussione e valutazione: la discussione degli elaborati avverrà contestualmente all'esame orale e **prevede anche domande su tutti gli argomenti di laboratorio trattati a lezione**.

Struttura della Consegna

La consegna dovrà consistere di un unico achivio contenente 3 componenti. L'archivio dovrà essere caricato sul sito moodle del corso seguendo il link che verrà reso disponibile alla pagina del corso.

- Un file .s contenente il **codice** assembly
- un **breve video** (max 5 minuti) dove si registra lo schermo del dispositivo che avete utilizzato per l'implementazione durante l'esecuzione del programma, commentandone il funzionamento in base a 2-3 combinazioni di input diverse
- la **relazione** in formato PDF, strutturata come segue.
    1. **Informazioni** su autori, indirizzo mail, matricola e data di consegna
    2. **Descrizione** della soluzione adottata, trattando principalmente i seguenti punti:
        a. Descrizione ad alto livello di ciascun algoritmo di cifratura/decifratura, di altre eventuali procedure e del main, in linguaggio naturale, con flow-chart, in pseudo-linguaggio, etc
        b. Uso dei registri e memoria (stack, piuttosto che memoria statica o dinamica)
    3. **Test di corretto funzionamento**, per fornire evidenze del corretto funzionamento del programma.