Progetto Assembly 2021-2022

Dati della studentessa:

Alessia Mazzeo
alessia.mazzeo1@stud.unifi.it
n° di matricola: 7083825

UNIVERSITÀ DEGLI STUDI FIRENZE

SOMMARIO:

Il programma in consegna assieme a questa relazione è stato sviluppato seguendo le linee guida del Progetto Assembly RISC-V per il Corso di Architetture degli Elaboratori A.A. 2021/2022.

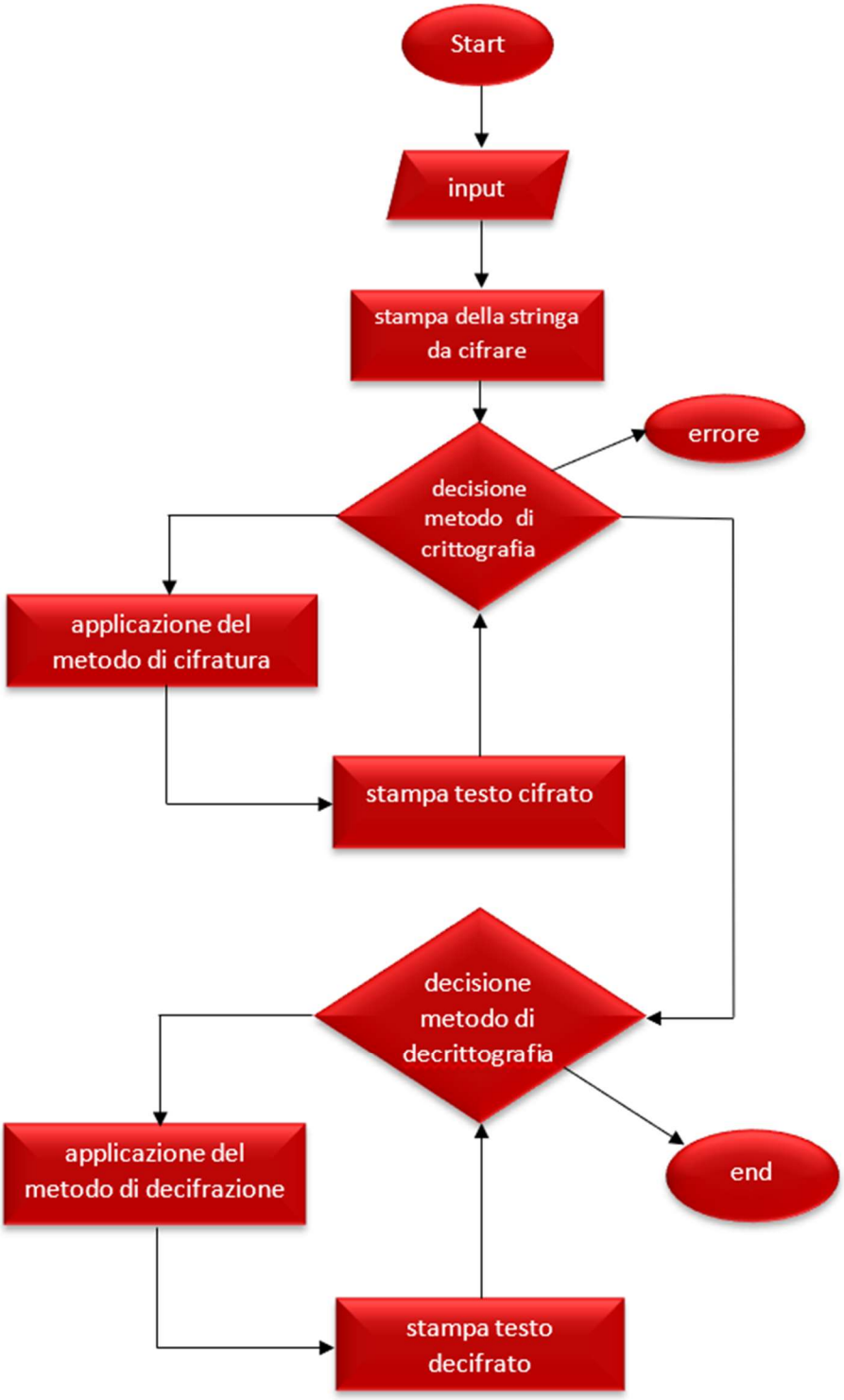
La struttura consiste in 7 parti:

1) Input e dichiarazione dei dati utili al programma
2) Decisione del metodo di crittografia
3) Decisione del metodo di decrittografia
4) Metodi di codifica
5) Metodi di decodifica
6) Metodi di servizio
7) Stampa e conclusione del programma

Utilizzo della memoria:

All'interno dell'elaborato è stato fatto uso sia di memoria volatile per i registri e per la memorizzazione delle stringhe, sia di memoria dinamica sfruttando una pila nel metodo C.

- Flow chart generale del programma:



1. Input e dichiarazione dei dati utili al programma

.Data:

Nella sezione *.data* sono gestiti quelli che vengono considerati i valori di input ed alcuni elementi utili all'esecuzione corretta del programma.
Vengono definite *mycypher* e *myplaintext*.
Rispettivamente: la stringa che indica l'ordine in cui eseguire i metodi di codifica, ed il testo da cifrare.

Sono inoltre definite la costante "*k*" richiesta dal *metodo_A* e la "*chiave*" richiesta dal *metodo B*.

Si utilizzano poi altre 3 stringhe:
-una per la gestione degli errori
-una impiegata come separatore per gli output a video
-una che viene utilizzata per decretare la fine del programma.

.text (parte iniziale) :

Nella prima parte della sezione *.text* vengono salvati in dei registri di tipo s gli indirizzi di *mycypher*, *myplaintext* e della *chiave*. La scelta del tipo di registro deriva dal fatto che non si vogliono modificare gli indirizzi delle stringhe.
È inoltre definito l'indirizzo *s5* utile nel *metodo_C* per creare una nuova stringa evitando così sovrascrizioni.

Viene poi stampata *myplaintext* in forma iniziale, prima di essere cifrata.
Tale scelta è stata presa per rispettare la simmetria dell'output a video richiesto ad ogni passaggio di crittografia e decrittografia.

2. Decisione del metodo di crittografia

La sezione di *Decisione_metodo* scorre la stringa *mycypher* analizzando il carattere contenuto in ogni posizione.
Sul registro temporaneo *t3* vengono caricati i codici ASCII delle lettere di interesse.
Ovvero: 65-A, 66-B, 67-C, 68-D, 69-E.

Per ogni lettera riconosciuta viene eseguito il metodo corrispondente sfruttando delle istruzioni di tipo branch.
Se venisse trovato un carattere diverso da quelli previsti durante lo scorrimento, il programma stamperebbe un messaggio di errore e terminerebbe.

3. Decisione del metodo di decrittografia

Per quanto riguarda la sezione *Decrittografia*, per la decisione del metodo di decifratura, il ragionamento applicato è speculare alla sezione precedente.
Con l'unica differenza che *mycypher* viene analizzata in senso opposto.
Per questa sezione non è previsto un messaggio di errore in caso di carattere non consentito, poiché se fosse stato presente l'esecuzione sarebbe già terminata.

4. Metodi di cifratura

Nella sezione *Metodi di cifratura,* è raccolto il corpo di ogni metodo di cifratura richiesto.
Segue la spiegazione generale di ogni metodo, la descrizione dettagliata delle righe di codice è implementata nel programma tramite commenti.
In ogni metodo prima di ritornare all'inizio per crittografare il carattere seguente, o eventualmente uscire dal metodo, avviene il salvataggio in memoria del carattere criptato.

Quando il metodo termina si torna alla decisione del metodo, e se fosse terminata anche *mycypher* il programma terminerebbe.

Metodo_A:

La regola di crittografia del Cifrario di Cesare è la seguente:

-maiuscole: 65+[(codice_carattere - 65 + k)%26]

-minuscole: 97+[(codice_carattere - 97 + k)%26]

-gli altri caratteri restano invariati.

Viene caricato *k* su *a0* essendo un valore in input del metodo.
Viene analizzata la stringa *myplaintext* fino alla fine (quando viene trovato uno 0).
Sapendo che le lettere maiuscole in codice ASCII hanno valore compreso tra 65 e 90 inclusi, e le minuscole tra 97 e 122 inclusi, viene controllato se il carattere è compreso in uno di questi intervalli e il programma verte ai sottometodi appositamente creati.
Per ogni carattere viene applicata la cifratura, sfruttando il metodo *modulo_26,* locato nei metodi di servizio (parte finale del programma).

Metodo_B:

Nel *metodo_B* vengono gestite contemporaneamente le due stringhe *myplaintext* e *chiave*.
Quando arrivo alla fine della stringa *chiave* riazzero i contatori in un metodo apposito e rinizio il loop di scorrimento da capo, fino a quando il metodo non si interrompe arrivando alla fine di *myplaintext*.
La regola di crittografia dei caratteri del metodo B è uguale per ogni tipo di carattere ed è la seguente:
-codice cifrato= 32 + (codice_carattere + codice_chiave) %96.

Il metodo *modulo_96* è speculare al procedimento del *modulo_26* e si trova sempre nei metodi di servizio.

Metodo_C:

Per il *metodo_C* viene allocato un ulteriore spazio di memoria per la nuova stringa criptata, poiché il metodo modifica la lunghezza del testo iniziale e questo porta ad errori di sovrascrizione. Tale indirizzo di memoria viene poi ricaricato su *s1,* per garantire la

corretta comunicazione con eventuali metodi seguenti, e spostato di 1000 in caso il *metodo_C* venisse applicato 2 volte.

La procedura si basa sulla marcatura dei caratteri già analizzati di *myplaintext* e sul salvataggio delle loro posizioni.

Per ogni carattere analizzato esso viene marcato su *myplaintext*, salvato sulla nuova stringa, seguito delle posizioni in cui è presente ognuna separata da un trattino.

Nel caso di una posizione a più cifre, ognuna viene salvata in una pila, per poi poterci riaccedere in ordine al momento della memorizzazione del numero.
Ogni volta che salvo la cifra effettuo l'operazione *modulo 10* per ottenere la cifra più a destra. Dopo averla inserita nella pila, eseguo una divisione per interi per 10, in modo da troncare il numero sulla penultima cifra, che sarà oggetto delle successive operazioni.
Ogni cifra viene convertita nel relativo codice ASCII aggiungendoci 48.

Segue la fase di memorizzazione in cui le cifre vengono estratte dalla pila e memorizzate sulla stringa. Avendo precedentemente convertito le cifre in ASCII, se viene trovato uno 0, quest'ultimo segna la fine della pila stessa.

Una volta finita l'analisi di un carattere e la sua cifratura viene sfruttato il metodo *spazio* per separare i caratteri cifrati e riazzerare il contatore precedentemente utilizzato.

Metodo_D:

All'interno del *Metodo_D* vengono eseguiti i controlli applicati nel *metodo_A*, per riconoscere lettere maiuscole e minuscole, inoltre viene introdotto un controllo aggiuntivo per verificare se il carattere analizzato è un numero (codice ASCII compreso tra 48 e 57 inclusi).
Se viene trovato un simbolo non si deve applicare nessun tipo di crittografia e l'esecuzione procede verso il prossimo carattere.
Il metodo di crittografia a Dizionario per le lettere richiede di scambiare il carattere iniziale con il suo inverso, sia come posizione nell'ordine alfabetico, sia come tipo (maiuscola o minuscola).
Quindi il corpo del metodo viene comunque suddiviso in 3 casistiche (minuscole, maiuscole e numeri), anche se il procedimento delle prime 2 sezioni è molto simile.

Viene calcolata la distanza dalla prima lettera del suo tipo (“a” minuscola o “A” maiuscola), e tale distanza viene applicata al contrario ripartendo dalla fine dell’alfabeto (da “z” minuscola o da “Z” maiuscola), in questo caso del tipo opposto a quello di partenza.

Per quanto riguarda i numeri invece la regola di crittografia è la seguente:
-numero_cifrato= 9 - num;
Per cui viene calcolata la differenza tra il codice ASCII del mio numero e il codice ASCII del 9, una volta ottenuta questa differenza, viene sommato 48 per riottenere un numero in ASCII stampabile a video correttamente.
Nonostante il metodo di decrittografia sia speculare a quello di crittografia essi sono stati separati come richiesto da specifica.
Per tale motivo in questa relazione non verrà approfondito il metodo di *Decrittografia_D.*

Metodo_E:

Il metodo e deve restituire la stringa da cifrare in ordine inverso, per fare questo la stringa viene scorsa fino a metà e via via vengono scambiati i caratteri nelle posizioni opposte (il primo con l’ultimo, il secondo con il penultimo ecc.).
La lunghezza della stringa viene preventivamente calcolata in un metodo apposito, facente parte dei *metodi di servizio*.
Se tale processo non si arrestasse a metà stringa riotterrei la stringa di partenza.
Il metodo di decrittografia è speculare a quello di crittografia, anche se gestito separatamente.
Per tale motivo in questa relazione non verrà approfondito il metodo di *Decrittografia_E.*

5. Metodi di decifratura

Decrittografia_A:

La regola inversa di decrittografia per il *metodo_A* è la seguente:
-Per le maiuscole: 65+[(codice_carattere + 65 - k)%26
-Per le minuscole: 97+[(codice_carattere -97 - k)%26

Per il resto del procedimento, il metodo di decrittografia è speculare a quello di crittografia.

Quindi viene analizzata la stringa, eseguiti i controlli per verificare il tipo di carattere, e applicata la regola di crittografia corrispondente.
Una volta modificato il carattere esso viene salvato sovrascrivendo la stringa di partenza.

Decrittografia_B:

La regola inversa di decrittografia per il *metodo_B* è la seguente:
-codice decifrato= {[(codice_cifrato-32)-chiave]+96}%96
Lo scorrimento della stringa *myplaintext* e della *chiave* avvengono allo stesso modo, con l'unico accorgimento di gestire in maniera particolare le minuscole, poiché avendo valore maggiore di 96 portano a casi particolari che vanno gestiti riaggiungendo 96.

Decrittografia_C:

Il metodo di *Decrittografia_C* procede nel seguente modo:
Viene analizzato il primo elemento trovato, il quale sarà un carattere effettivo della stringa originale. Si esegue un salto di 2 posizioni, sapendo che l'elemento successivo al carattere appena analizzato è un "-" di separazione.
A questo punto dovremmo avere un numero, poiché se il carattere è presente nella stringa criptata, esso deve occorrere almeno una volta.
Le cifre vengono analizzate una ad una, aggiungendoci 1 (dato che la prima posizione è considerata come 1 e non come 0) e riportandola da ASCII a decimale.
A questo punto il numero viene moltiplicato per 10, e viene aggiunto al numero precedentemente calcolato.
Alla prima esecuzione otterremo semplicemente la cifra, poiché il registro t4 (su cui viene salvato il numero indicante la posizione) è inizializzato a 0.
Tale processo di ricostruzione del numero di posizione reitera fino a quando non viene trovato un trattino o uno spazio.
I due casi vengono analizzati separatamente:
-Il sottometodo *trattino_trovato* calcola la posizione di memoria in cui deve essere salvato il carattere attualmente in analisi aggiungendo il numero di posizione all'indirizzo iniziale della stringa, decrementandolo di 1 per il motivo illustrato precedentemente.
A questo punto il carattere viene salvato nella posizione appena calcolata, il contatore viene resettato, e si scorre il puntatore di stringa.
Infine il flusso di dati viene reindirizzato al *loop_cifre* poiché se è stato trovato un trattino, ciò significa che è presente un'ulteriore occorrenza del carattere.

-Il sottometodo *spazio_trovato* invece decreta la fine delle occorrenze del carattere analizzato. Quest'ultimo viene quindi salvato nell'ultima posizione calcolata, viene riazzerato il contatore ed incrementato il puntatore. In conclusione si esegue un salto all'inizio del metodo per reiterare l'analisi del carattere successivo.

6. Metodi di servizio

I metodi appartenenti a questa sezione sono stati raggruppati infondo al programma per rendere più fluida la lettura dei metodi che li precedono.

modulo_26:

Per prima cosa viene effettuata una divisione per 26 sul numero, in modo da ottenere la parte intera del risultato della divisione.

tale cifra viene rimoltiplicata a 26, e la differenza tra il numero di partenza e quello appena calcolato è esattamente il resto della divisione (e quindi il modulo).

A questo punto tramite un'istruzione jr (collegata ad una jal al momento della chiamata al metodo), permette di ritornare all'esecuzione lineare del programma.

modulo_96:

Il *modulo_96* esegue le stesse operazioni del modulo 26, ma utilizzando il valore 96.

modulo_10:

Il *modulo_10* esegue le stesse operazioni del modulo 26, ma utilizzando il valore 10.

In ogni metodo riguardante il modulo avviene un controllo per gestire il caso in cui il numero risultasse negativo.
Se questo avviene, si aggiunge il valore dell'operando del modulo applicato.

riazzera_chiave:

Riazzera il contatore utilizzato nello scorrimento della *chiave* nel *metodo_B*, per poter reiterare tale procedimento.

riazzera_chiave_decri:

Riazzera il contatore utilizzato per lo scorrimento della *chiave* nel *Decrittografia_B*, per poter reiterare tale procedimento.

calcolo_lunghezza_stringa:

Utile nel *metodo_E*, questo procedimento scorre la stringa che gli viene passata e restituisce il contatore utilizzato quando arriva al termine della stessa.

errore:

Questo metodo è stato pensato per gestire eventuali errori di input del programma.
In caso ne venga rilevato uno, viene stampata a video una stringa di errore, ed il programma termina.

7. Stampa e conclusione del programma

Da ultimo vengono strutturati i metodi di stampa del programma.
Questi hanno lo scopo di produrre in output a video il risultato di crittografia (per quanto riguarda *stampa_cifrato*), e di decrittografia (*stampa_decifrato*), di ogni metodo applicato.
Inoltre ognuno di essi racchiude una sezione di riazzeramento dei registri, in maniera tale da risolvere eventuali conflitti che potrebbero crearsi con il riutilizzo dei registri in metodi differenti.
Il *metodo_C* richiede una gestione separata della stampa poiché ad ogni suo utilizzo viene aggiornato l'indirizzo della nuova stringa (come precedentemente spiegato).

La conclusione del programma avviene con un metodo *fine,* che termina l'esecuzione stampando a video la frase "Programma concluso".