Calcolatori Elettronici

Esercizi di programmazione Assembly

Di seguito sono presentati alcuni esercizi di programmazione Assembly PD32 che possono essere utilizzati dagli studenti come spunto per esercitarsi nella programmazione.
Gli esercizi sono proposti in ordine di difficoltà crescente, e tentano di coprire tutti gli argomenti toccati a lezione.

Esercizio 1

Dato un vettore di *n* unsigned word, individuare il massimo e il minimo. Il vettore deve essere esplicitamente dichiarato come variabile globale; il massimo e il minimo devono essere salvati all'interno di altre due variabili globali esplicitamente dichiarate.

Esercizio 2

Scrivere un programma che, dato un vettore di *n* longword, salvi in un secondo vettore gli elementi memorizzati nel primo in ordine inverso.

Esercizio 3

All'indirizzo 1280h è presente un vettore di 64 signed longword. Implementare in Assembly l'algoritmo bubble sort per ordinare gli elementi di questo vettore.

Esercizio 4

All'inidirizzo 5555h è presente un vettore di *n* unsigned word. Scrivere un programma Assembly che memorizzi all'interno del registro R1 il numero di elementi pari presenti in posizione dispari all'interno del vettore (ossia, se in posizione 1 nel vettore è presente un numero pari, R1 viene incrementato, se in posizione 3 è presente un numero pari R1 viene incrementato, e così via). Il numero *n* di elementi del vettore può essere sia pari che dispari.

Esercizio 5

Dato un vettore di 32 unsigned byte, scandire il vettore e creare una maschera di bit all'interno del registro R0 in cui un bit viene impostato ad 1 se l'elemento corrispondente del vettore è pari, a 0 altrimenti.

Esercizio 6

Scrivere una subroutine *divisione* che implementi via software l'operazione di divisione tramite il metodo delle sottrazioni successive. La funzione opera su interi non segnati e riceve come parametri di ingresso il dividendo nel registro R2, il divisore nel registro R3 e restituisce in R0 il risultato ed in R1 il resto. Scrivere un programma che utilizzi questa funzione per calcolare il risultato della divisione tra due numeri specificati all'interno di due variabili globali.

Esercizio 7

Utilizzare la funzione *divisione* realizzata per l'Esercizio 6 per implementare un programma che calcoli il massimo comune divisore tra due numeri utilizzando il metodo delle divsioni successive.

Esercizio 8

Una *stringa* è un vettore di unsigned byte, in cui i caratteri sono rappresentati come valori numerici secondo il formato ASCII (la 'a' corrisponde al numero decimale 97, la 'b' al numero decimale 98, e così via) e la fine della stringa viene rappresenata dal numero decimale 0 (terminatore di stringa). Implementare una funzione Assembly *copiastringa* che accetti in R1 l'indirizzo della stringa sorgente, ed in R2 l'indirizzo di un vettore grande abbastanza da contenere una copia della stringa. La funzione copia un carattere (ossia un byte) alla volta dal vettore sorgente al vettore destinazione, fintanto che non trova il valore 0 (terminatore di stringa). La funzione restituisce in R0 il numero di caratteri copiati (escluso il terminatore di stringa). Scrivere un programma Assembly che utilizzi la funzione *copiastringa* per effettuare una copia di una stringa definita dall'utente.

Esercizio 9

Data una variabile globale 'a' di tipo word, memorizzare all'interno di un'altra variabile globale 'b' di dimensione longword la rappresentazione dello stesso dato contenuto in 'a' rappresentato mediante un codice correttore d'errore a distanza di Hamming 3. I bit di parità debbono essere inseriti in posizione corrispondente ad una potenza di 2 (1, 2, 4, 8, ...).

Esercizio 10

Data una variabile globle 'a' di tipo longword che memorizza un dato rappresentato con codice di correzione d'errore a distanza di Hamming 3, implementare: 1) una subroutine Assembly che accetti in R1 il dato e restituisca in R0 'vero' se il dato passato non ha alcun errore, 'falso' altrimenti; 2) un subroutine Assembly che accetti in R1 il dato e restituisca in

R0 la rappresentazione originale del dato (eliminando, quindi, i bit di parità). Scrivere un programma Assembly che utilizzi queste due subroutine per verificare la correttezza del dato e (in caso positivo) decodifichi lo stesso.

Esercizio 11

Dato un numero intero rappresentato in aritmetica segnata, scrivere un programma assembly che converta questo numero nella rappresentazione in virgola mobile IEEE 754.

Esercizio 12

La periferica STDINPUT consiste di una tastiera che genera un'interruzione ogni volta che viene premuto un tasto. La periferica DISCO permette di memorizzare dei file.
Scrivere un programma Assembly che, ogni volta che viene premuto un tasto su tastiera, lo memorizza all'interno di un file su disco.
La documentazione della periferica DISCO disponibile all'interno del dissimulator è disponibile qui.

Esercizio 13

Un processore PD32 è interconnesso con una periferica TASTIERA e una periferica DMAC che controlla due dischi. Quando la periferica TASTIERA invia un'iterruzione al PD32, il processore legge da TASTIERA un numero che rappresenta la quantità di dati da trasferire da DISCO1 a DISCO2. Alla ricezione dell'interruzione, quindi, il PD32 legge questo numero e programma il DMAC per trasferire i dati da DISCO1 a DISCO2 passando tramite memoria. I dati vengono letti dall'indirizzo AAAAh del DISCO1 e vengono scritti nell'indirizzo AABBh in memoria centrale. Vengono successivamente trasferiti dall'indirizzo AABBh della memoria centrale all'indirizzo BBBBh del DISCO2. Scrivere il driver delle interruzioni di TASTIERA e il driver che gestisce le interruzioni del DMAC. Attenzione: le interruzioni del DMAC possono segnalare sia il termine del trasferimento da DISCO1 alla memoria, sia il termine del trasferimento da memoria a DISCO2.