Calcolatori Elettronici
Prova di Laboratorio - assembly MIPS: first-last_string
7 Febbraio 2013

Introduzione

Lo scopo di questa prova di laboratorio è lo sviluppo di un semplice programma nel linguaggio assembly del processore MIPS. Non è richiesta una particolare base di conoscenze algoritmiche, ma semplicemente un minimo di dimestichezza con la programmazione assembly.

Istruzioni

Cominciate facendo il login sulla macchina del laboratorio che vi è stata assegnata. Per il login occorre usare matricola e password dello *student portal*. Sul desktop troverete una cartella contenente i simulatori QtSpim e Mars. Lanciate ed utilizzate quello che preferite. Tutto il vostro codice (sia esso costituito da un singolo file, o da file multipli) andrà salvato nella cartella "`mips`" da creare sul drive `H:` .

Create un file `student-info.txt` con incluso il vostro nome e cognome e numero di matricola nella cartella "`mips`". Per maggior sicurezza, includete anche nome, cognome e matricola come commento, in testa ad ogni file sorgente. Alla fine della prova, i file saranno prelevati automaticamente dalla directory . Tutto quello che lascerete nella cartella `mips` sarà utilizzato per la valutazione. Salvare i vostri file altrove, o non indicare nome e cognome, porterà inevitabilmente all'annullamento della vostra prova. *Tutti i file all'esterno della cartella verranno cancellati automaticamente!!!*

Informazioni generali

La prova ***non è*** a correzione automatica. Tutti gli studenti autori di un codice che viene assemblato senza errori e risponde alle specifiche indicate alla sezione seguente saranno ammessi all'orale. In quella sede, il codice prodotto sarà esaminato e discusso col docente.

Le specifiche

Dovete scrivere un programma assembly che legge da tastiera 4 stringhe e stampa in output la prima e l'ultima delle stringhe in ordine alfabetico.

Il main alloca in area stack spazio per le 4 stringhe, e richiama una funzione `first_string`, fornendole come parametro il puntatore all'area di memorizzazione delle stringhe, che stampa la prima stringa in ordine alfabetico. Poi richiama una funzione `last_string`, fornendole come parametro il puntatore all'area di memorizzazione delle stringhe, che stampa l'ultima stringa in ordine alfabetico. Sia `first_string` che `last_string` fanno uso di una funzione `string_cmp` che (analogamente alla funz. `strcmp` della *standard library* del C) riceve come parametri i puntatori a due stringhe da confrontare e ritorna -1, 0 o 1 a seconda che (rispettivamente): *i)* la prima stringa preceda la seconda in ordine alfabetico, *ii)* le stringhe siano uguali , *iii)* la prima stringa segua la seconda in ordine alfabetico.

Suggerimenti

Attenzione all'allocazione delle 4 stringhe in area stack. Poiché le funzioni `first_string` e `last_string` riceveranno solo il puntatore alla prima stringa, una soluzione semplice è allocare uno spazio uguale per tutte e quattro le stringhe, indipendentemente dalla loro effettiva lunghezza. Ad esempio, riservando 100 byte in area stack si ha spazio per 4 stringhe sino a 24 caratteri utili (+ terminatore). In questo modo le funzioni `first_string` e `last_string` potranno trovare il puntatore alla seconda stringa sommando 25 al puntatore della prima (scambiato come parametro), quello della terza sommando 50, e così via ...

Potete leggere e stampare stringhe utilizzando le system call 8 e 4, rispettivamente.

Il seguente è un output di esempio:

```
Dammi la stringa n.1: pippo
Dammi la stringa n.2: pluto
Dammi la stringa n.3: paperino
Dammi la stringa n.4: paperone

First string: paperino
Last string: pluto

-- program is finished running --
```

Valutazione

Scrivere un programma funzionante, che faccia uso di tre funzioni come richiesto nelle specifiche e che segua le convenzioni di salvataggio dei registri è strettamente necessario per essere ammessi a sostenere l'orale. In quella sede, si entrerà nel dettaglio della struttura del codice, dando una valutazione migliore a soluzioni "pulite" e ben commentate.