Corso di *Architetture degli Elaboratori* – A.A. 2016/2017

Lista esercizi di progetto:

1. Analisi di stinghe
2. Procedure annidate e ricorsive
3. Operazioni fra matrici

Esercizio di progetto (1): analisi di stringhe

Utilizzando QtSpim, scrivere e provare un programma che prende in input una stringa qualsiasi di dimensione *massima* di 100 caratteri (es, "uno due ciao 33 tree tre uno Uno di eci"), e traduce ogni sua sottosequenza di caratteri separati da almeno uno spazio (nell'esempio, le sottosequenze "uno", "due", "ciao", "tree", "tre", "uno" "Uno" "di", "eci") applicando le seguenti regole:

- ogni sottosequenza "uno" si traduce nel carattere '1';
- ogni sottosequenza "due" si traduce nel carattere '2';
- ogni sottosequenza "nove" si traduce nel carattere '9';
- qualsiasi alta sottosequenza si traduce nel carattere '?'.

Nell'esempio considerato, se "uno due ciao 33 tree tre uno Uno di eci" è la stringa di input inserita da tastiera, a seguito della traduzione il programma dovrà stampare su console la seguente stringa di output:

- *Risultato della traduzione: 1 2 ? ? ? ? 1 ? ??*

Esercizio di progetto (2): procedure annidate e ricorsive

Siano *G* e *F* due procedure definite come segue (in pseudo-linguaggio di alto livello):

```
Procedure G(n)
begin
        b := 0
        for k := 0, 1, 2, . . . , n do
        begin
                u := F(k)
                b := b^2+u
        end
        return b
end

Procedure F(n)
begin
        if n = 0
                then return 1
                else return 2*F(n − 1) + n
end
```

Utilizzando QtSpim, scrivere e provare un programma che legga inizialmente un numero naturale *n* compreso tra 1 e 8, e che visualizzi su console:

- il valore restituito dalla procedura *G(n)*, implementando *G* e *F* come descritto precedentemente. Le chiamate alle due procedure *G* ed *F* devono essere realizzate utilizzando l'istruzione jal (jump and link).
- la traccia con la sequenza delle chiamate annidate (con argomento fra parentesi) ed i valori restituiti dalle varie chiamate annidate (valore restituito fra parentesi), sia per *G* che per *F*.

Mostrare e discutere nella relazione l'evoluzione dello stack nel caso specifico in cui *n=3*.

Esempio di output su console nel caso in cui n=1:

Valore restituito dalla procedura*: G(1)=4*
Traccia:
G(1)-->F(0) -->F:return(1) -->F(1) -->F(0) -->F:return(1) -->F:return(3) --> G:return(4)

Infatti: *G(1)* richiama *F(0)*, che ritorna il valore 1; quindi viene richiamata *F(1)*, che a sua volta richiama *F(0)*; *F(0)* ritorna il valore 1, quindi *F(1)* ritorna il valore 3, ed infine *G(1)* ritorna il valore 4 che è il risultato finale.

Esercizio di progetto (3): operazioni fra matrici

Utilizzando QtSpim, scrivere e provare un programma che visualizza all'utente un menù di scelta con le seguenti cinque opzioni *a*, *b*, *c*, *d*, *e*:

a) **Inserimento di matrici**. Il programma richiede di inserire da tastiera un numero intero 0<n<5, e richiede quindi l'inserimento di due matrici quadrate, chiamate A e B, di dimensione n x n, contenenti numeri interi. Quindi si ritorna al menù di scelta.

   Facoltativo. Le matrici A e B dovranno essere allocate dinamicamente in memoria. Si consiglia l'utilizzo della system call 'sbrk' del MIPS.

   Ogni volta che si seleziona l'opzione a) del menu, i nuovi valori inseriti di A e B dovranno essere salvati nella stessa area di memoria in cui erano stati salvati i vecchi valori: i nuovi valori sovrascriveranno quelli vecchi.

   Facoltativo: Si dovrà allocare (con la 'sbrk') uno spazio aggiuntivo di memoria solo se le due nuove matrici dovessero richiedere più spazio di memoria rispetto a quello già allocato in precedenza.

   Esempio di interfaccia per l'inserimento delle due matrici:

   Dimensione matrici: 3x3

   Matrice A:
   Riga1: -2 44 5
   Riga2: 1 1 1
   Riga3: 3 0 1

   Matrice B:
   Riga1: 0 0 10
   Riga2: -1 1 -1
   Riga3: 1 0 0

b) **Somma di matrici**. Il programma effettua la somma fra le due matrici A e B, e visualizza su console il risultato A+B. Quindi si ritorna al menù di scelta.

   Ad esempio, se A e B sono state inserite come riportato nell'esempio al punto a), il programma dovrà visualizzare su console:

   Risultato di A+B:
   Riga1: -2 44 15
   Riga2: 0 2 0
   Riga3: 4 0 1

c) **Sottrazione di matrici**. Il programma effettua la sottrazione fra le due matrici A e B, e visualizza su console il risultato A-B. Quindi si ritorna al menù di scelta.

   Ad esempio, se A e B sono state inserite come riportato nell'esempio al punto a), il programma dovrà visualizzare su console:

Risultato di A-B:
Riga1: -2 44 -5
Riga2: 2 0 2
Riga3: 2 0 1

d) **Prodotto di matrici**. Il programma effettua il prodotto fra le due matrici A e B, e visualizza su console il risultato A*B. Quindi si ritorna al menù di scelta.

   Ad esempio, se A e B sono state inserite come riportato nell'esempio al punto a), il programma dovrà visualizzare su console:

   *Risultato di A*B:*
   Riga1: -39 44 -64
   Riga2: 0 1 9
   Riga3: 1 0 30

e) **Uscita.** Stampa un messaggio di uscita ed esce dal programma.

Alle opzioni *a*, *b*, *c*, *d* corrisponderanno le chiamate alle opportune procedure, e quindi il programma dovrà tornare disponibile per selezionare una nuova opzione. Alla scelta *e* corrisponderà la terminazione del programma.

Mostrare e discutere nella relazione l'evoluzione della memoria dinamica (heap) in alcuni casi di interesse.

Modalità di consegna degli esercizi assegnati e della relazione:

- Per sostenere l'esame è necessario consegnare preventivamente il codice e una relazione (in pdf) sugli esercizi assegnati
    - Ogni gruppo di lavoro dovrà consegnare un'unica relazione
    - Il codice consegnato deve essere funzionante sul simulatore QtSpim, anche se sviluppato e testato con altri simulatori (es, MARS)
- Un archivio contenente il codice e la relazione dovrà essere caricato sul sito moodle del corso seguendo l'apposito link che verrà reso disponibile alla pagina del corso
    - Non vengono prese in considerazione altre modalità di consegna
- La scadenza esatta della consegna verrà resa nota di volta in volta
- Discussione e valutazione: la discussione degli elaborati avverrà contestualmente all'esame orale e prevede anche domande sull'assembly e su tutti gli argomenti di laboratorio trattati a lezione. La valutazione incide sulla votazione finale.

Struttura della relazione (in formato pdf):

- Info su autori e data di consegna
    - gli autori (e loro indirizzo e-mail – preferibilmente quello universitaria @stud.unifi.it)
    - la data di consegna
- Per ciascun esercizio
    - Testo dell'esercizio
    - Descrizione della soluzione adottata, trattando principalmente i seguenti punti:
        - Descrizione ad alto livello dell'algoritmo (in linguaggio naturale, con flow-chart, in pseudo-linguaggio, etc.), delle strutture dati utilizzate (liste, vettori, etc.) e delle procedure utilizzate (parametri, funzionalità svolte)
        - Uso dei registri e memoria (stack, piuttosto che memoria statica o dinamica)
        - Motivazione delle scelte implementative
    - Simulazione
        - In questa sezione andranno gli screenshot commentati di una o più simulazioni-tipo, anche discutendo l'evoluzione del contenuto del "user data segment" e dello "user stack" durante l'esecuzione del codice. Mostrare anche il funzionamento dell'algoritmo in corrispondenza di input sbagliati (es, inserisco da tastiera un carattere al posto di un numero).
    - Codice MIPS assembly implementato e commentato in modo chiaro ed esauriente.

Codice

- Il codice deve essere suddiviso in cartelle (una cartella per ogni esercizio) ed i nomi dei files devono permettere di identificare facilmente l'esercizio a cui si riferiscono. Unico file per esercizio.
- Ricordarsi di inserire anche nel codice gli autori (e loro indirizzo e-mail) e la data di consegna.
- Seguire fedelmente le convenzioni sull'uso dei registri e delle procedure.
- Commentare il codice in modo significativo (move $s4,$s3 non significa solo che "copio il contenuto del registro $s3 in $s4"....).