Laboratorio di Architettura degli Elaboratori
A.A. 2016/17 — Programmazione Assembly

Scrivere il codice ARM che implementi le specifiche richieste e quindi verificarne il comportamento usando il simulatore ARMSIM.

Lezione 1

1.1 Esercizio

Scrivere un programma assembly che, dopo aver immesso in memoria, mediante una dichiarazione nella sezione `.data`, quattro numeri interi n_1, n_2, n_3, n_4, inserisca nei registri r_0, r_1, r_2, r_3, r_4 rispettivamente i seguenti valori:

- la somma dei valori n_1, n_2, n_3, n_4,
- la media dei valori n_1, n_2, n_3, n_4,
- $(2^{10} + 1) \times n_1$
- il resto della divisione di n_1 per 16
- il segno di n_1, ossia il valore 0 se n_1 è positivo o il valore 1 se n_1 è negativo

Lezione 2

2.1 Esercizio

Scrivere del codice assembly ARM che, nella sezione `.data`, definisca un vettore con 4 valori uguali a 0, e nella sezione `.text` contenga un programma che sostituisce il vettore con il vettore contenente i primi 4 numeri naturali.

2.2 Esercizio

Scrivere del codice assembly ARM che, nella sezione `.data`, definisca un vettore di 3 numeri interi, e nella sezione `.text` contenga un programma che sostituisce ogni valore nel vettore con il suo quadruplo.

2.3 Esercizio

Scrivere del codice assembly ARM che, nella sezione `.data`, definisca un vettore di 3 numeri interi, e nella sezione `.text` contenga un programma che sostituisce in vettore con una sua permutazione in cui ogni elemento viene spostato.

Lezione 3

3.1 Esercizio

Scrivere del codice assembly ARM che, nella sezione `.data`, inserisca in memoria due numeri interi positivi m ed n e nella sezione `.text` contenga un programma che, attraverso una serie di sottrazioni, calcoli quoziente e resto della divisone m/n. I valori quoziente e resto devono essere inseriti nei registri `r0, r1`.

3.2 Esercizio

Scrivere del codice assembly ARM che, nella sezione `.data`, definisca un vettore di 10 numeri interi, e nella sezione `.text`, contenga un programma che scrive nel registro `r0` la media aritmetica degli elementi del vettore.

Lezione 4

4.1 Esercizio

Scrivere del codice assembly ARM che, nella sezione `.data`, inserisca in memoria due numeri interi positivi m ed a, e nella sezione `.text`, contenga un programma che scrive, nel registro `r0`, l'approssimazione intera del logaritmo in base a di m, ossia un numero intero n tale che $a^n \leq m < a^{n+1}$

4.2 Esercizio

Scrivere del codice assembly ARM che, nella sezione `.data`, definisca un numero positivo n, seguito da generico vettore contenente esattamente n numeri interi. Si scriva inoltre, nella sezione `.text`, un programma che azzeri tutti gli elementi del vettore aventi valore pari.

4.3 Esercizio

Scrivere del codice assembly ARM che, nella sezione `.data`, definisca un numero positivo n, seguito da generico vettore contenente esattamente n numeri interi. Si scriva quindi, nella sezione `.text`, un programma che elimini dal vettore tutti gli elementi uguali a zero; l'eliminazione degli elementi fa fatta ricompattando il vettore, ossia traslando gli elementi successivi a quelli eliminati nelle posizioni del vettore lasciate libere.

Lezione 5

Per ciascuno degli esercizi seguenti, inserire la procedura richiesta all'interno di un programma principale che la richiami fornendole gli opportuni parametri.

5.1 Esercizio

Scrivere una procedura che inserisca, in un vettore di lunghezza n, la sequenza nei primi n numeri naturali a partire dal valore 0. Indirizzo base e lunghezza del vettore vengono forniti, rispettivamente, attraverso i registri `r0` e `r1`. Nota:

usare la direttiva `.skip` per riservare la zona di memoria destinata a contenere il vettore.

5.2 Esercizio

Scrivere una procedura che dato un vettore V di interi e un numero naturale p, azzeri tutti gli elementi del vettore il cui indice è un multiplo di p diverso da 0 e p. Nella procedura si consideri il primo elemento del vettore come avente indice 0. La procedura riceve in `r0` l'indirizzo base del vettore, in `r1` la sua lunghezza e in `r2` il valore di p.

5.3 Esercizio

Scrivere un procedura che, combinando le procedure dei due esercizi precedenti, implementi il crivello di Eratostene, ossia prima crei un vettore V di numeri naturali da 0 a n, quindi azzeri il valore 1 e successivamente scandisca il vettore e, per ogni numero primo p trovato, azzeri tutti i multipli propri di p (usando la procedura 5.2). Indirizzo base e lunghezza del vettore V vengono forniti alla procedura attraverso i registri `r0` e `r1`.

Lezione 6

Per ciascuno degli esercizi seguenti, inserire la procedura richiesta all'interno di un programma principale che la richiami fornendole gli opportuni parametri.

6.1 Esercizio

Scrivere una procedura che ricevuto in ingresso una stringa di caratteri, stampi in uscita la sequenza delle sole lettere dell'alfabeto, maiuscole e minuscole, contenute all'interno della stringa. La procedura riceve in `r0` l'indirizzo base del vettore di byte,contente la codifica ASCII della stringa, il valore 0 marca la fine della stringa. Per risolvere l'esercizio è utile sapere che il valore della codifica ASCII di generico carattere c può essere scritto in assembly ARM come `# 'c`.

6.2 Esercizio

Scrivere una procedura che riceve in ingresso gli handle di due file di testo. Nell'ipotesi che il primo file contenga una sequenza di numeri interi terminante col valore 0, la procedura trascrive, nel secondo file, tutti i valori pari contenuti nel primo file.

6.3 Esercizio

Scrivere una procedura che ricevuto in ingresso un intero positivo n, stampi in uscita la sua rappresentazione binaria, omettendo gli 0 non significativi. La procedura riceve in `r0` il valore n.

Lezione 7

Per ciascuno degli esercizi seguenti, inserire la procedura richiesta all'interno di un programma principale che la richiami fornendole gli opportuni parametri.

7.1 Esercizio

Scrivere una procedura che dato un vettore di interi, riordini gli elementi del vettore in maniera tale che tutti i numeri negativi precedano tutti i numeri positivi o uguali a 0. La procedura riceve in `r0` l'indirizzo base del vettore e in `r1` la sua lunghezza.

Nota. La procedura può restituire una qualsiasi permutazione del vettore di ingresso in cui nessun numero negativo segua un numero positivo, nessun altro vincolo deve essere soddisfatto.

7.2 Esercizio

Scrivere una procedura che riceve in ingresso i nomi di due file testo, di cui il primo primo file contiene una sequenza di numeri interi, non più lunga di 100 e terminante con un coppia di 0, mentre il secondo file può essere eventualmente inesistente. La procedura deve inserire nel secondo file una una permutazione dei valori presenti nel primo in cui tutti i numeri negativi precedono quelli positivi.

La procedure riceve in `r0` e `r1` gli indirizzi base delle stringhe di caratteri rappresentati i nomi dei due file.

In alternativa, per chi voglia cimentarsi, proporre una soluzione dell'esercizio funzionante con file di lunghezza arbitraria.

Lezione 8

Per ciascuno degli esercizi seguenti, inserire la procedura richiesta all'interno di un programma principale che la richiami fornendole gli opportuni parametri.

8.1 Esercizio

Scrivere una procedura che dato un vettore di interi, costruisca una lista contenente, nello stesso ordine, tutti i valori del vettore. La procedura riceve in `r0` l'indirizzo base del vettore e in `r1` la sua lunghezza; la procedura restituisce in `r0` l'indirizzo del primo elemento della lista.

8.2 Esercizio

Scrivere una procedura che visualizzi in uscita, su righe distinte, tutti gli elementi di una lista di numeri interi. La procedura riceve in `r0` l'indirizzo del primo elemento della lista.

8.3 Esercizio

Scrivere una procedura che prende in ingresso una matrice di interi M, memorizzata per righe e contenente solo valori compresi tra 0 e 99. La procedura

deve visualizzare M sullo standard output, ossia nello standard output dovranno essere visibili in righe separate tutti gli elementi di ciascuna riga di M, si chiede inoltre che gli elementi di una stessa colonna risultino allineati tra loro. La procedura riceve come argomenti, in `r0`, l'indirizzo del primo elemento della matrice, in `r1`, in numero di righe, in`r2`, il numero di colonne.

Nota: i caratteri ASCII di nuova linea e di spazio vengono indicati con `\n` e `\ ` ; di conseguenza, all'interno del codice assembly le corrispondenti costanti vengono indicate con `# '\n` e `# '\ `.

Lezione 9

Per ciascuno degli esercizi seguenti, inserire la richiesta all'interno di un programma principale che, utilizzando le procedure della lezione 8, fornisca le liste argomento e visualizzi il contenuto delle liste risultato.

9.1 Esercizio

Scrivere una procedura, preferibilmente ricorsiva, che presa una lista di interi l, restituisca due liste risultato, la prima contenente i valori pari di l e la seconda contenete i valori dispari di l. Non è necessario che la procedura preservi l'ordine degli elementi nelle liste. La procedura riceve in `r0` ed `r1` i puntatori alle due liste argomento e restituisce in `r0` il puntatore alla lista risultato.

9.2 Esercizio

Definiamo una lista ordinata se ogni elemento nella lista è maggiore o uguale agli elementi che lo precedono. Scrivere una procedura che ricevute in input due liste di interi ordinate fonda le due liste tra di loro ottenendo una lista che contiene l'unione degli elementi delle due liste e che sia anch'essa ordinata. Al solito si usino i registri `r0`, `r1` per passaggio di argomenti e risultato.

Lezione 10

Per ciascuno degli esercizi seguenti, inserire la procedura richiesta all'interno di un programma principale che la richiami fornendole gli opportuni parametri.

10.1 Esercizio

Scrivere una procedura che verifica se una stringa di caratteri (vettore di byte) è palindroma. Si ricorda che una stringa è palindroma se letta al contrario resta invariata. L'indirizzo base della stringa viene fornito nel registro `r0`, e il valore 0 marca la fine della stringa (ossia tutti gli elementi della stringa sono diversi da 0 e l'ultimo elemento del vettore è seguito in memoria dal valore 0). La procedura restituisce il risultato nel registro `r0`, 1 se la stringa è palindroma, 0 in caso contrario.

10.2 Esercizio

Una matrice quadrata M si definisce *simmetrica* se coincide con la sua trasposta, ossia se, per ogni coppia di indici i, j, gli elementi $M[i, j]$ e $M[j, i]$ sono uguali. Scrivere una procedura che ricevuta in ingresso una matrice quadrata di byte M, memorizzata per righe, verifica se M è una matrice trasposta. La procedura riceve in `r0` l'indirizzo del primo elemento della matrice e in `r1` e in sua dimensione. La procedura restituisce il risultato in `r0`: 1 se la matrice è simmetrica, 0 altrimenti.