Laboratorio di Architettura degli Elaboratori
A.A. 2015/16 — 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 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 di 10 numeri interi, e nella sezione `.text` contenga un programma che sostituisca ogni valore nel vettore con il suo doppio.

2.2 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`.

Lezione 3

3.1 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.

3.2 Esercizio

Scrivere del codice assembly ARM che, nella sezione `.data`, inserisca in memoria un numero intero positivo m, e nella sezione `.text`, contenga un programma che scrive, nel registro `r0`, l'approssimazione intera della radice quadrata di m ($\lfloor\sqrt{m}\rfloor$). L'approssimazione della radice quadra può essere calcolata scandendo in ordine crescente i numeri naturali sino a trovare il più grande naturale n tale che $n^2 \leq m$.

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`.

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. 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

Trasformare la soluzione dell'esercizio 4.3 da programma in procedura, ossia scrivere una procedura assembly che dato un vettore V di interi, elimini dal vettore tutti gli elementi uguali a zero. La procedura riceve in `r0` l'indirizzo base del vettore e in `r1` la sua lunghezza. Al termine della procedura, il registro `r1` deve contenere la lunghezza del vettore ricompattato.

Modificare quindi l'esercizio 5.3 ottenendo una procedura che crea un vettore contenente tutti i numeri primi minori di n.

6.2 Esercizio

Scriva una procedura che, ricevuti in ingresso un numero intero n (in `r0`) ed un vettore V di 32 byte (indirizzo base in `r1`), inserisca il valore di ciascuno dei 32 bit di n in un diverso byte di V.

6.3 Esercizio

Scrivere una procedura che, ricevuto in ingresso un intero n, calcoli il valore della parità pari dei bit in 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 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`.

7.2 Esercizio

Scrivere una procedura che, ricevuto in ingresso l'handle di un file di testo contenente una sequenza di numeri interi terminata dal valore 0, restituisca in uscita la somma dei numeri dispari contenuti nel file. La procedura usa il registro `r0` sia per ricevere l'handle in ingresso che per restituire il valore in uscita.

7.3 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 terminata dal valore 0, la procedura trascrive nel secondo file tutti i valori pari contenuti nel primo file.

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 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 un stessa colonna risultino allineati tra loro. I parametri della procedura sono `r0` indirizzo del primo elemento della matrice, `r1` numero di righe, `r2` 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 `# '\ `.

8.2 Esercizio

Scrivere una procedura che trasformi un vettore in una lista, ossia la procedura, preso in ingresso un vettore di interi, costruisce una lista contente esattamente gli stessi interi, nello stesso ordine, 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.

8.3 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.

Lezione 9

Per ciascuno degli esercizi seguenti, inserire la procedura 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 che ricevute in input due liste di interi le concateni tra di loro, ossia costruisca una lista contenente tutti gli elementi della prima lista seguiti da tutti gli elementi delle seconda lista. 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

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. 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`, mentre il

valore 0 viene usato per marcare la fine della stringa (ossia tutti gli elementi della stringa sono divelsi da 0 e la locazione di memoria successiva al'ultimo elemento contiene il valore 0). La procedura restituisce il risultato nel registro `r0`, 1 se la stringa è palindroma, 0 in caso contrario.

10.2 Esercizio

Scrivere una procedura che presa in ingresso una matrice quadrata di byte M, memorizzata per righe, costruisca la matrice trasposta di M, anche questa memorizzata per righe. I parametri della procedura sono `r0` indirizzo del primo elemento della matrice, `r1`, dimensione della matrice, `r2` indirizzo dove depositare il primo elemento della matrice trasposta.

Nota: la matrice trasposta si ottiene scambiando le righe con le colonne, ossia la matrice trasposta di M ha come righe le colonne di M, prese nello stesso ordine.