Architettura degli Elaboratori – Corso A e B – Terzo appello sessione Estiva – 20 luglio 2022

Una procedura **EL * aggiungi(char c, EL * lista)** lavora su una lista di elementi EL che sono sequenze di tre campi:

- un carattere **c**, rappresentato come intero da 32 bit. Il codice ASCII è contenuto nel byte meno significativo della parola;
- un intero **occ**, di 32 bit, che rappresenta un numero di occorrenze del carattere c;
- un puntatore **next**, anch'esso di 32 bit, che rappresenta il puntatore al prossimo elemento della lista.

La procedura riceve un carattere ASCII e un puntatore ad una lista di elementi EL:

- se il puntatore è NULL (ovvero 0: la lista è vuota), crea un elemento EL, al campo **c** mette il carattere ricevuto come primo parametro, mette a 1 il campo **occ**, NULL nel campo **next** e restituisce l'indirizzo dell'elemento appena creato;
- Se il puntatore non è null, scorre la lista fino a che non trova un elemento con il campo **c** uguale al carattere passato come primo argomento o finchè non raggiunge la fine lista senza trovare il carattere cercato. Nel primo caso, aggiorna il campo **occ** (**occ = occ + 1**) e restituisce il secondo parametro (puntatore alla lista). Nel secondo caso, aggiunge (in testa alla lista) un nuovo elemento **EL** con il campo **c** uguale al primo parametro, il campo **occ** a 1 e il campo **next** settato al puntatore alla lista, e ne restituisce l'indirizzo.

Una seconda procedura **EL * crea(char * stringa)** riceve come parametro il puntatore a una stringa di caratteri ASCII terminati da un **NULL** (carattere di codice 0) e, avvalendosi della procedura **aggiungi** precedentemente descritta, crea la lista degli elementi **EL** che descrivono le occorrenze dei caratteri. La prima volta invocherà la **aggiungi** utilizzando un **NULL** come secondo parametro, per poi utilizzare, alle chiamate successive, il puntatore restituito dalla **aggiungi** stessa.

Si richiede di fornire il codice assembler delle procedure **aggiungi** e **crea** che rispettino tutte le convenzioni per l'implementazione delle funzioni/procedure ARMv7.

A solo scopo di test, le due procedure possono essere testate utilizzando il codice C della pagina seguente.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct __lista {
  char c;
  int occ;
  struct __lista * next;
} ELEMENTO;


void stampa(ELEMENTO * l) {
  while(l != NULL) {
    printf("%c:%d -> ", l->c, l->occ);
    l=l->next;
  }
  printf("NULL\n");
  return;
}

extern ELEMENTO * aggiungi(char, ELEMENTO *);
extern ELEMENTO * crea(char * stringa);

int main(int argc, char ** argv) {

  ELEMENTO * lista = crea(argv[1]);
  stampa(lista);

  return(0);
}
```

Utilizzando questo codice, compilato con un comando tipo

arm-linux-gnueabihf-gcc aggiungi.s lista.c -static

ed eseguendo al prompt shell il comando

qemu-arm ./a.out abbccdd

dovreste ottenere una cosa tipo

d:2 -> c:2 -> b:2 -> a:1 -> NULL

(se avete un linux ARM nativo, si compila con il comando gcc e si esegue con un semplice ./a.out)

Traccia di soluzione

```
	.text
	.global aggiungi
	.type aggiungi,%function

aggiungi:	@ r0 = c , r1 = lista
	mov r3, r1		@ lista (salvato per restiuirlo)
loop:	cmp r1,#0		@ confronta puntatore alla lista
	beq fine		@ o è vuota o non ho trovato il carattere
	ldr r2, [r1]		@ altrimenti carica carttere c
	cmp r2, r0		@ controlla se è quello cercato
	beq trovato		@ semmai incrementa e ritorna
	ldr r1, [r1,#8]		@ continua col next
	b loop
trovato: ldr r2, [r1,#4]	@ carica occ
	add r2, r2, #1		@ incrementalo
	str r2, [r1, #4]	@ salvalo
	mov r0, r3		@ restiusci vecchio puntatore
	mov pc, lr		@ ritorno
fine:	push {r0,r3,lr}		@ salva c lista e ritorno
mov r0, #12
bl malloc
pop {r2,r3,lr}
str r2, [r0]			@ c
mov r2, #1
str r2, [r0, #4]	@ occ = 1
str r3, [r0, #8]	@ next = lista (originale)
mov pc, lr			@ return (r0 punta già al nuovo elemento)

.text
.global crea
.type crea,%function
@ r0 indirizzo della stringa null terminated
crea:	push {r4-r5,lr}
mov r4, r0		@ salva indirizzo stringa
mov r5, #0		@ lista = NULL
loop:	ldrb r0, [r4],#1	@ carica carattere
cmp r0, #0			@ è anche primo param della aggiungi
beq fine		@ se siamo arrivati al null, la stringa è finita
mov r1, r5		@ secondo param = lista
bl aggiungi
mov r5, r0		@ salva lista
b loop			@ prossimo carattere
fine:	mov r0, r5		@ restituisci puntatore a lista
pop {r4,r5,pc}	@ ritorno al chiamante
```