Architettura degli Elaboratori – Appello straordinario 27/3/2023

Si fornisca il codice assembler ARMv7 della procedura che esegue la ricerca binaria in un vettore di interi, il cui codice C è il seguente:

```
int cerca(int *v, int start, int stop, int x) {

  while(start<=stop) {
    int centro = (start + stop) / 2;
    if(v[centro] == x)      return(centro);
    if(x < v[centro])       stop = centro - 1;
    if(x > v[centro])       start = centro + 1;
    printf("v[%d] = %d\n", centro, v[centro]);
  }
  return(-1);
}
```

Nel realizzare il codice assembler dovranno essere utilizzate tutte le convenzioni di programmazione tipiche di ARMv7, come viste durante le lezioni ed enunciate nel libro di testo. Si assuma (com'è di fatto) che la printf sia disponibile come libreria.

Per testare il corretto funzionamento del programma assembler può essere utilizzato il codice che segue, che prende il numero da cercare come unico parametro della riga di comando e stampa un messaggio con l'esito della ricerca di quel numero nel vettore definito staticamente a inizio codice.

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

int v[] = {1,6,9,13, 18,26,31,33, 45,46,47,50, 61,62,76,89};

extern int cerca(int *, int, int, int);

int main(int argc, char ** argv) {
  int x = atoi(argv[1]);
  int r = cerca(v, 0, 15, x);
  if(r == -1) printf("%d non è presente nel vettore dato\n", x);
  else printf("%d si trova alla posizione %d\n", x, r);
  return(0);
}
```

Traccia di soluzione

```
.text
.global cerca
.type cerca, %function
cerca:      push {r4-r9,lr}         @ salvataggio non temporanei
registri utilizzati
mov r4, r0          @ v
mov r5, r1          @ start
mov r6, r2          @ stop
mov r7, r3          @ x
while:      cmp r5, r6          @ while(start<=stop)
bgt nontrovato          @ se > allora fine con -1
add r8, r5, r6          @ start + stop
lsr r8, r8, #1          @ (start+stop) / 2 : centro
ldr r9, [r4, r8, lsl #2]@ v[centro]
cmp r7, r9          @ x ? v[centro]
beq trovato         @ se uguali, fine
sublt r6, r8, #1    @ x<v[centro]: cerco in start,centro-1
addgt r5, r8, #1    @ x>v[centro]: cerco in centro+1,stop
ldr r0,=fmt         @ preparo argomenti printf: &stringa formato
mov r1, r8          @ centro
mov r2, r9          @ v[centro]
bl printf           @ chiamo printf (tutti i registri temp persi)
b while                 @ eseguo nuova iterazione
trovato: mov r0, r8             @ restituisco centro
pop {r4-r9,pc}          @ e ritorno
nontrovato: mov r0, #-1         @ restituisco -1
      pop {r4-r9,pc}            @ e ritorno
.data               @ sezione dati
fmt:  .string "v[%d]=%d\n"    @ stringa di formato per la printf
```

Commenti (e punti valutati nella correzione delle consegne):

- I registri temporanei sono sovrascritti durante la printf, dunque occorre mettere I parametri nei registri non temporanei, preventivamente salvati
- Va salvato anche LR, dal momento che sarà sovrascritto dalla BL PRINTF
- Per aggiornare start o stop si usano espressioni condizionali, evitando così salti nella compilazione dell'IF, visto che il ramo THEN è di una sola istruzione
- La divisione (intera) per 2 si fa con uno shift a destra di una posizione
- Nella LDR si moltiplica il registro indice per 4, visto che il vettore è di interi (quindi 4 byte per posizione, in una memoria indirizzata al byte)
- La funzione resitutisce il valore di ritorno in R0 e accetta I parmetri nei registri temponanei
- La chiamata della printf segue la convenzione per il passaggio dei parametri: parametri di ingresso nei temporanei (sono meno di 5) e parametro di ritorno (in questo caso ignorato) in R0.