Laboratorio di Elementi di Architetture e Sistemi Operativi

Soluzioni del Compitino del 22 Maggio 2013

Esercizio 1. Scrivere un programma in linguaggio assembly che determina il minimo tra dieci numeri contenuti in memoria a partire dall locazione `NUMBERS` e lo pone in una locazione di memoria dedicata ed identificata dal simbolo `MIN`.

1. Provare il funzionamento del programma con il simulatore LC-3.
2. Verificare che dopo l'istruzione `HALT` il risultato è ancora presente in memoria.

```
.ORIG x3000
        AND R4, R4, x0    ; azzera R4
        ADD R4, R4, #9    ; inizializza R4 a 9
        LEA R2, NUMBERS   ; carica in R2 l'indirizzo del primo elemento
        LDR R1, R2, #0    ; carica in R1 il valore del primo elemento
        ADD R2, R2, #1    ; incrementa R2
LOOP    LDR R3, R2, #0    ; carica in R3 l'elemento corrente
        NOT R5, R3        ; inverti i bit di R3
        ADD R5, R5, #1    ; aggiungi 1: ora R5 = -R3
        ADD R5, R1, R5    ; R5 = R1 - R3
        BRnz NEXT         ; se R5 <= 0 allora R1 <= R3,
                          ; passa all'elemento successivo
        ADD R1, R3, #0    ; altrimenti, R3 diventa il nuovo minimo
NEXT    ADD R2, R2, #1    ; incrementa R2
        ADD R4, R4, #-1   ; decrementa R4
        BRp LOOP          ; se R4 > 0 fai un altro ciclo
        ST  R1, MIN       ; salva il risultato in memoria
        HALT
NUMBERS .FILL x0001
        .FILL x0002
        .FILL x0003
        .FILL x0004
        .FILL x0005
        .FILL #-3
        .FILL x0007
        .FILL x0008
        .FILL x0009
        .FILL x000A
MIN     .BLKW 1
        .END
```

Esercizio 2. Scrivere un programma in linguaggio assembly che converta i caratteri di una stringa in MAIUSCOLO. Si assuma che la stringa si trovi in memoria nella locazione `STRING`. Per risolvere l'esercizio si tenga presente che:

1. per caricare la stringa in memoria si può usare la direttiva `.STRINGZ`;
2. le stringhe sono rappresentate come in C (con lo `0` finale);
3. i caratteri minuscoli hanno codice ASCII compreso tra `a=97` e `z=122` (estremi inclusi);
4. per convertire un carattere da minuscolo a MAIUSCOLO basta sottrarre `32`;

```
.ORIG x3000
      LEA R1,STRING  ; r1 = &str[0]
      LD  R3,MENA    ; R3 = -97;
      LD  R4,DIFF    ; R4 = -25;
      LD  R5,MAIA    ; R5 = 90;
LOOP  LDR R2,R1,#0   ; r2 = *r1
      BRz CONT
      ADD R1,R1,#1   ; vai al prossimo carattere
      ADD R2,R2,R3   ; r2 = r2 -97
      BRn LOOP       ; se r2 < 97, fai un altro giro
      ADD R2,R2,R4   ; r2 = r2 -97 -25 = r2 -122
      BRp LOOP       ; se r2 > 122, fai un altro giro
      ADD R2,R2,R5   ; converte in maiuscolo
      STR R2,R1,#-1  ; salva il risultato
      BRnzp LOOP     ; fai un altro giro
CONT  HALT
STRING  .STRINGZ "Prova44aaaAAAaoooo!45 ; @"
MENA .FILL #-97
DIFF .FILL #-25
MAIA .FILL #90
.END
```

Esercizio 3. Scrivere un programma in linguaggio assembly che calcoli il *Massimo Comun Divisore* tra due numeri positivi contenuti nelle locazioni di memoria `A` e `B` e salvi il risultato nella locazione di memoria `MCD`. Il programma deve utilizzare l'algoritmo di Euclide per calcolare l'MCD.

```
.ORIG x3000
     LD  R1,A       ; carica a in R1
     LD  R2,B       ; carica b in R2
LOOP NOT R3,R2      ; R3 = -b
     ADD R3,R3,#1   ;
     ADD R3,R1,R3   ; R3 = a-b;
     BRz CONT       ; se a = b esci;
     BRp SWAP       ; se b < a vai a SWAP
     ADD R2,R3,#0   ; se b > a, b = R3 = -(a-b)
     NOT R2,R2      ; cambia di segno a R2
     ADD R2,R2,#1   ;
     BRnzp LOOP     ; fai un altro loop
SWAP ADD R1,R3,#0   ; se b < a, a = R3 = a-b;
     BRnzp LOOP     ; altro giro, altra corsa
CONT ST R1, MCD     ; salva il risultato
     HALT
A    .FILL #64
B    .FILL #48
MCD  .BLKW 1
.END
```