SOLUZIONE ESERCIZI LEZIONE 3 (16/03/2015) di Amin Ait Lamqadem

Esercizio 2: Si scriva un programma assembler che prenda in input 2 valori numerici, li sommi e stampi a video il risultato.

Soluzione:

```
.ORIG x3000

        TRAP x23            ; leggo da tastiera

        ADD R1, R0, 0       ; salvo il primo numero in R1

        TRAP x23

        ADD R2, R0, 0       ; salvo il secondo numero in R2

        ; trasformo il codice dei caratteri nei numeri corrispondenti
        LD R0, ZERO_ASCII_NEG

        ADD R1, R1, R0

        ADD R2, R2, R0

        ADD R0, R1, R2      ; calcolo la somma tra i numeri in input

        ; trasformo il numero nel carattere corrispondente
        LD R1, ZERO_ASCII

        ADD R0, R0, R1

        TRAP x21            ; stampo il risultato contenuto in R0

        HALT

; Definizione di variabili
ZERO_ASCII .FILL 0048           ; codice ascii del carattere '0'
ZERO_ASCII_NEG .FILL -48

.END
```

Esercizio 3: Si scriva un programma assembler che prenda in input una stringa, e una lettera e calcoli quante volte appare la lettera nella stringa.

Soluzione:

```
.ORIG x3000
        LEA R0, MSG1            ; carico il messaggio "inserisci una stringa: "
        TRAP x22                ; lo stampo
        LEA R2, STRINGA         ; uso R2 per contenere l'indirizzo della stringa
        LD R3, NEW_LINE         ; carico il valore del carattere '\n'

inizio_lettura:
        TRAP x20                ; leggo da tastiera
        TRAP x21                ; stampo il carattere letto
        ADD R4, R0, R3          ; vedo se ho letto '\n'
        BRz fine_lettura        ; se ho letto '\n' allora ho letto una linea, allora ho finito
        STR R0, R2, 0           ; salvo nel vettore: R0 contiene cosa scrivere, R2 è l'indice del
                                ; vettore
        ADD R2, R2, 1           ; incremento l'indice
        BRnzp inizio_lettura    ; ciclo

fine_lettura:
        AND R1, R1, 0
        STR R1, R2, 0           ; tappo la stringa, mettendo il carattere '\0' in ultima posizione

        LEA R0, MSG2            ; carico il messaggio "inserisci una lettera: "
        TRAP x22                ; lo stampo
        TRAP x20                ; leggo da tastiera
        TRAP x21                ; stampo il carattere letto

        ; dentro R0 ho il carattere da confrontare
        LEA R2, STRINGA         ; dentro R2 ho l'indirizzo della stringa

        AND R5, R5, 0           ; uso R5 per contare quante volte compare il carattere nella
                                ; stringa
        NOT R0, R0              ; calcolo il complemento a due del carattere per averne il
                                ; valore negativo
        ADD R0, R0, 1

inizio_conteggio:
        LDR R4, R2, 0           ; carico dentro R4 un carattere della stringa
        ADD R3, R4, R0          ; sommo i due carattere per farne il confronto
        BRz incrementa          ; se è il carattere cercato incrementa il numero di volte che l'hai
                                ; trovato
        ADD R3, R4, 0           ; confronto con 0 per vedere se sono arrivato al tappo '\0'
        BRz fine_conteggio      ; la stringa è finita, esci
        ADD R2, R2, 1           ; incremento l'indirizzo del vettore per avere la prossima
                                ; posizione
        BRnzp inizio_conteggio  ; cicla


incrementa:
```

```
        ADD R5, R5, 1           ; incremento il n. di volte che ho incontrato quel carattere
        ADD R2, R2, 1           ; incremento l'indirizzo del vettore per avere la prossima
                                ; posizione
        BRnzp inizio_conteggio  ; continua a contare

fine_conteggio:
        AND R0, R0, 0
        ADD R0, R0,  13         ; stampo '\n'
        TRAP x21

        ; dentro R5 ho il n. di volte che ho incontrato il carattere
        LD R1, ZERO_ASCII
        ADD R0, R5, R1          ; carico il carattere di R5 dentro R0 per stamparne il valore
        TRAP x21                ; stampo il carattere che rappresenta il n. in R5
        HALT

; Definizione di variabili
MSG1 .STRINGZ "Inserisci una stringa: "
MSG2 .STRINGZ "Inserisci una lettera: "
NEW_LINE .FILL -13              ; valore negativo del codice ASCII del carattere '\n'
ZERO_ASCII .FILL 48             ; valore del carattere '0' nel codice ASCII
STRINGA .BLKW 256               ; creo una vettore di 256 celle di memoria per salvarci la
                                ; stringa in input


.END
```

Esercizio 4: Si modifichi l'esercizio 2 e lo si trasformi in una funzione

Soluzione 1:
Questa soluzione come quella dell'esercizio 2, prende in input solo numeri di una sola cifra e stampa solo la cifra più a destra della somma dei due numeri (esempio: 7+6=3 invece di 13)

```
.ORIG x3000

        JSR SOMMA_DUE_NUMERI
        HALT

SOMMA_DUE_NUMERI:

        ST R7, IND_RET      ; mi salvo l'indirizzo di ritorno in una variabile, perché la TRAP lo
                            ; sovrascrive

        TRAP x23            ; leggo da tastiera

        ADD R1, R0, 0       ; salvo il primo numero in R1

        TRAP x23

        ADD R2, R0, 0       ; salvo il secondo numero in R2

        ; trasformo il codice dei caratteri nei numeri corrispondenti
        LD R0, ZERO_ASCII_NEG

        ADD R1, R1, R0

        ADD R2, R2, R0

        ADD R0, R1, R2      ; calcolo la somma tra i numeri in input

        ; trasformo il numero nel carattere corrispondente
        LD R1, ZERO_ASCII

        ADD R0, R0, R1

        TRAP x21            ; stampo il risultato contenuto in R0

        LD R7, IND_RET      ; rimetto l'indirizzo di ritorno corretto, non quello modificato dalla
                            ; TRAP

        RET                 ; ritorno, funzione finita

; Definizione di variabili
ZERO_ASCII .FILL 0048           ; codice ascii del carattere '0'
ZERO_ASCII_NEG .FILL -48
IND_RET .FILL 0                 ; serve per salvare l'indirizzo di ritorno contenuto in R7, così
                                ; da non perderlo  quando si chiama una funzione
.END
```

Soluzione 2:
Questa soluzione, più complessa, permette invece di prendere in input qualsiasi numero (non più solo di una cifra) e stampare anche numeri con più di una cifra.

```
.ORIG x3000

        ; leggo il primo numero
        LEA R0, MSG1
        TRAP x22                ; stampo un messaggio per dire all'utente di inserire un numero

        LEA R0, STRINGA         ; carico l'indirizzo della stringa dove salvare il numero
        JSR LEGGI_STRINGA       ; chiamo la funzione che legge una stringa

        LEA R0, STRINGA         ; rimetto in R0 l'indirizzo della stringa da utilizzare
        JSR STRING_TO_INT       ; converto la stringa in R0 nel numero che rappresenta, il
                                ; risultato sarà in R0

        ST R0, NUMERO1          ; salvo il primo numero in una variabile

        ; leggo il secondo numero
        LEA R0, MSG1
        TRAP x22                ; stampo un messaggio per dire all'utente di inserire un numero

        LEA R0, STRINGA         ; carico l'indirizzo della stringa dove salvare il numero
        JSR LEGGI_STRINGA       ; chiamo la funzione che legge una stringa

        LEA R0, STRINGA         ; rimetto in R0 l'indirizzo della stringa da utilizzare
        JSR STRING_TO_INT       ; converto la stringa in R0 nel numero che rappresenta, il
                                ; risultato sarà in R0

        ST R0, NUMERO2          ; salvo il secondo numero in una variabile

        ; faccio la somma tra i due numeri
        LD R1, NUMERO1
        LD R2, NUMERO2

        ADD R3, R1, R2          ; in R3 ho la somma dei due numeri
        ST R3, RISULTATO        ; salvo il risultato in una variabile

        ; converto il numero in una stringa
        LD R0, RISULTATO        ; in R0 devo mettere il numero da convertire
        LEA R1, STRINGA         ; in R1 metto l'indirizzo della stringa da utilizzare

        JSR INT_TO_STRING       ; converto l'intero in una stringa che posso stampare

        ; stampo la stringa
        LEA R0, STRINGA
        TRAP x22

        HALT
```

```
LEGGI_STRINGA:          ; dentro R0 deve esserci l'indirizzo dove salvare la stringa,

        ADD R1, R0, 0           ; salvo dentro R1 l'indirizzo della stringa
        LD R3, NEW_LINE_NEG
        ST R7, IND_RET          ; in R7 ho l'indirizzo di ritorno, con la TRAP lo
                                ; sovrascrivo, quindi me lo salvo prima in una variabile
        INIZIO_LETTURA:
                TRAP x20        ; leggo da tastiera
                TRAP x21        ; stampo il carattere letto
                ADD R2, R0, R3  ; vedo se ho letto '\n'
                BRz FINE_LETTURA ; se ho letto '\n' allora ho letto una linea, allora ho finito
                STR R0, R1, 0   ; salvo nel vettore: R0 contiene cosa scrivere, R1 è
                                ; l'indice del vettore
                ADD R1, R1, 1   ; incremento l'indice
                BRnzp INIZIO_LETTURA ; ciclo

        FINE_LETTURA:
                AND R0, R0, 0
                STR R0, R1, 0   ; tappo la stringa, mettendo il carattere '\0' in ultima
                                ; posizione
                LD R7, IND_RET  ; rimetto l'indirizzo di ritorno corretto dentro R7 per
                                ; poter usare RET
                RET

STRING_TO_INT:  ; trasforma una stringa in un intero, l'indirizzo della stringa deve essere in R0
                ; alla chiamata sempre in R0 metterò il numero risultante dalla conversione
                ; alla fine della funzione

                ADD R1, R0, 0   ; mi salvo l'indirizzo della stringa in R1, così in R0
                                ; posso fare i calcoli
                AND R0, R0, 0   ; azzero R0 dove metterò il numero


        INIZIO_CONV:
                        ; leggo un carattere
                        LDR R4, R1, 0       ; leggo un carattere della stringa

                        ADD R4, R4, 0       ; testo se sono arrivato alla fine della stringa,
                                            ; cioè se ho letto 0 ('\0')
                        BRz FINE_CONV       ; se la stringa è finita, ho finito

                        ; moltiplico R0 per dieci
                        AND R2, R2, 0       ; con R2 conto fino a dieci
                        ADD R2, R2, 1
                        LD R5, MENO_DIECI   ; carico -10 in R5
                        ADD R4, R0, 0       ; mi salvo il valore iniziale di R0 per
                                            ; accumulare il risultato in R0
        INIZIO_MUL:
                        ADD R0, R0, R4
                        ADD R2, R2, 1
                        ADD R3, R2, R5
                        BRz FINE_MUL
```

```
                BRnzp INIZIO_MUL        ; ciclo per moltiplicare
        FINE_MUL:
                LDR R4, R1, 0
                LD R5, ASCII_NEG
                ADD R4, R4, R5          ; converto il codice del carattere nel numero
                                        ; corrispondente
                ADD R0, R0, R4          ; aggiungo il numero in R0

                ADD R1, R1, 1           ; incremento l'indirizzo della stringa per leggere il
                                        ; prossimo carattere

                BRnzp INIZIO_CONV       ; ciclo finché non ho finito

        FINE_CONV:
                RET                     ; ritorna, in R0 c'è il risultato

INT_TO_STRING:          ; converte un intero in una stringa, il numero da convertire deve essere
                        ; in R0 e in R1 ci deve essere l'indirizzo della stringa da usare alla
                        ; chiamata della funzione

        ADD R4, R1, 0   ; mi salvo l'indirizzo iniziale della stringa dentro R4

        INIZIO_CON:
                AND R2, R2, 0           ; uso R2 come contatore per la divisione

                LD R5, MENO_DIECI       ; carico -10 in R5

                ADD R3, R0, 0           ; in R3 mi salvo il resto

                INIZIO_DIVISIONE:
                        ADD R0, R0, R5          ; in R3 salvo il resto della divisione

                        BRnz FINE_DIVISIONE     ; se il resto è negativo la divisione è
                                                ; finita

                        ADD R3, R0, 0           ; modifico R0 inserendo il valore diviso
                                                ; 10

                        ADD R2, R2, 1           ; incremento il contatore della divisione

                        BRnzp INIZIO_DIVISIONE  ; ciclo

                FINE_DIVISIONE:

                ADD R0, R2, 0           ; salvo il quoziente dentro R0 per
                                        ; calcolare la prossima cifra da mettere
                                        ; nella stringa
                LD R5, ASCII
                ADD R3, R3, R5          ; trasformo il resto contenuto in R3 nel
                                        ; carattere corrispondente
                STR R3, R1, 0           ; salvo la cifra del primo posto dentro la
                                        ; stringa
```

```
        ADD R0, R0, 0                   ; se il quoziente della divisione è 0
        BRz FINE_CON                    ; ho finito

        ADD R1, R1, 1                   ; incremento l'indice della stringa

        BRnzp INIZIO_CON                ; se è diverso da 0 continuo a calcolare

FINE_CON:
        STR R0, R1, 1           ; dentro R0 ho 0, lo metto nella stringa per chiuderla

        ; ora devo ribaltare la stringa per avere il numero scritto correttamente
        INIZIO_REVERSE:
                LDR R2, R4, 0           ; in R4 ho l'indirizzo di inizio stringa
                LDR R3, R1, 0           ; in R1 ho l'indirizzo di fine stringa

                STR R3, R4, 0           ; salvo il carattere di fine stringa all'inizio
                STR R2, R1, 0           ; salvo il carattere di inizio stringa alla fine

                ADD R4, R4, 1           ; vado alla prossima posizione della stringa
                LD R5, MENO_UNO
                ADD R1, R1, R5          ; vado alla posizione precedente della stringa

                NOT R2, R4
                ADD R2, R2, 1           ; salvo dentro R2 il valore negativo di R4 per
                                        ; fare il test
                ADD R3, R1, R2          ; faccio un test che non si siano ancora incrociati
                                        ; (R1 > R4)

                BRnz FINE_REVERSE       ; se i due indici si sono già scambiati, allora ho
                                        ; finito, la stringa è ribaltata

                BRnzp INIZIO_REVERSE    ; ciclo
        FINE_REVERSE:
                RET                     ; la stringa è stata scritta correttamente

; Definizione variabili
NUMERO1 .FILL 0
NUMERO2 .FILL 0
RISULTATO .FILL 0
NEW_LINE_NEG .FILL -13          ; valore del codice ASCII del carattere '\n'
ASCII .FILL 48                  ;valore negativo del codice ASCII del carattere '0'
ASCII_NEG .FILL -48             ; valore negativo del codice ASCII del carattere '0'
MENO_UNO .FILL -1
MENO_DIECI .FILL -10
IND_RET .FILL 0
MSG1 .STRINGZ "Inserisci un numero: "
MSG2 .STRINGZ "La somma è "
STRINGA .BLKW 256               ; creo una vettore di 256 celle di memoria per salvarci la
                                ; stringa in input
.END
```