Ulteriori Esercizi per laboratorio Assembly

Scrivere in Assembler per Intel 80x86 la funzione `TrovaMax` che cerca il massimo tra due vettori non ordinati di byte (numeri senza segno) e ritorna in AL 0 se il massimo si trova nel primo vettore, 1 se il massimo si trova nel secondo. Viene passato anche il numero N di elementi dei due vettori. Di conseguenza il programma chiamante stampa a video una stringa diversa, come riportato di seguito (si supponga che la funzione ScriviStringa visualizzi a video la stringa passata come parametro):

```
SECTION data
      Vett1: db 13,15,22,7,5,3,21,2,0,10
      Vett2: db 1,7,3,2,22,21,3,28,7,11
      Stringa1: db 'Il massimo è nel vettore 1',0
      Stringa2: db 'Il massimo è nel vettore 2',0

SECTION text
..start:
      mov ax, data
      mov ds, ax
      mov es, ax
      mov ax, Vett2
      push ax
      mov ax, Vett1
      push ax
      mov ax, 10
      push ax
      call TrovaMax
      add sp, 6
      cmp al, 1
      je Max_in_2
      mov bx, Stringa1
      jmp fine
Max_in_2:
     mov bx, Stringa2
fine:
     push bx
     call ScriviStringa
     add sp, 2
      ret
```

Scrivere in Assembler per Intel 80x86 la funzione `MirrorStringa` che riceve in ingresso una stringa zero-terminata stringa (non passata tramite lo stack). La stringa è composta da 2 parole di lunghezza qualsiasi (anche diverse tra loro) separate da un solo spazio. La funzione deve scambiare di posto le due parole e scrivere il risultato nella stringa (zero-terminata) mirror. Ad esempio, se la stringa in ingresso vale 'Grande Pennello', la stringa risultante dal mirror deve valere 'Pennello Grande'.
Le variabili del programma sono le seguenti:

```
stringa: db 'Buona Pasqua',0
mirror: resb 100
```

Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `Filtro` che copia un vettore di N byte dall'indirizzo DS:SI all'indirizzo ES:DI. I dati vengono copiati all'indirizzo destinazione se e solo se il bit meno significativo del dato è a 1.
Le variabili del programma sono le seguenti:

```
N: db 100
Vett1: times 25 db 3, times 25 db 4, times 25 db 5, times 25 db 6
Vett2: resb 100
```

Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 un programma che richiama la funzione EliminaElementi la cui definizione C è la seguente:

```
void EliminaElementi (int *vettore, int X, int Y)
```

Tale funzione deve eliminare tutti gli elementi con valore compreso tra X e Y (inclusi) dal vettore vettore. Il vettore non è ordinato e contiene valori byte positivi ed è terminato con il valore -1 (che non può essere mai presente come valore di un elemento).

La funzione deve essere richiamata dal programma principale con passaggio di parametri tramite lo stack e deve rimuovere dal vettore tutte le occorrenze di valori compresi tra X e Y.

Le variabili del programma sono le seguenti:

```
vettore: db 12, 33, 22, 11, 55, 23, 12, 120, 1, -1
```

Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 un programma che richiama la funzione `SaturaElementi` la cui definizione C è la seguente:

```
void SaturaElementi (unsigned char *vettore, int N)
```

Tale funzione deve controllare ciascun elemento del vettore e se maggiore di N, modificarlo con il valore di N. Il vettore non è ordinato e contiene valori byte positivi ed è terminato con il valore -1 (che non può essere mai presente come valore di un elemento).

La funzione deve essere richiamata dal programma principale con passaggio di parametri tramite lo stack.

Le variabili del programma sono le seguenti:

```
vettore: db 12, 33, 22, 11, 55, 23, 12, 120, 1, -1
```

Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 un programma che riceve in ingresso (tramite variabili definite nel segmento dati) tre stringhe ASCIIZ zero-terminate (secondo la convenzione C) `Stringa1, Stringa2` e `Stringa3`. Il programma deve riportare in Stringa3 una parola di Stringa1 seguita da una di Stringa2, seguita da una Stringa1, e così via. Le parole sono delimitate da singoli spazi.
A titolo di esempio, considerando il seguente segmento dati:

```
Stringa1: db ‘Io sto bene.’,0
Stringa2: db ‘oggi molto’, 0
Stringa3: resb 255
```

il programma deve riportare in Stringa3 la stringa ASCIIZ “Io oggi sto molto bene.”

Scrivere in Assembler per Intel 80x86 un programma che riceve in ingresso (tramite variabili definite nel segmento dati) due stringhe ASCIIZ zero-terminate (secondo la convenzione C) `Stringa1` e `Stringa2`. Il programma deve riportare nel registro AX il numero di caratteri di `Stringa1` presenti anche (almeno una volta) in `Stringa2`. Ad esempio, se `Stringa1` vale `“città”` e `Stringa2` vale `“tà”` il programma deve restituire 3 in AX, se valgono invece `“ciminiera”` e `“italia”` il programma deve restituire 4, ecc.

Scrivere in Assembler per Intel 80x86 la funzione `pariodispari` che analizza un vettore Vett di lunghezza N (**con Vett e N parametri della funzione passati mediante lo stack**) e riporta in AX il valore -1 se in Vett ci sono più elementi dispari che pari, 1 se invece ci sono più valori pari che dispari e 0 se sono in egual numero. Il programma chiamante ha il seguente codice Assembler (si supponga che la funzione `ScriviStringa` visualizzi a video la stringa passata come parametro):

```
SECTION data
Vett: db 13,15,22,7,5,3,21,2,0,10
N: db 10
StringaDispari: db ‘Vett contiene più elementi dispari’,0
StringaPari: db ‘Vett contiene più elementi pari’,0
StringaUguale: db ‘Vett contiene tanti elementi dispari quanti pari’,0

SECTION text
..start:
      mov ax, data
      mov ds, ax
      mov es, ax
      mov ax, Vett
      push ax
      mov ax, N
      push ax
      call pariodispari
      add sp, 4
      cmp ax, 0
      je uguale
      jg pari
      mov bx, StringaDispari
      jmp fine
pari:
      mov bx, StringaPari
      jmp fine
uguale:
      mov bx, StringaUguale
fine:
     push bx
     call ScriviStringa
     add sp, 2
      mov ax, 4c00h
      int 21h
```

Scrivere in Assembler per Intel 80x86 la funzione `contacaratteri` che conta quante volte si presenta nella stringa ASCIIZ zero-terminata (secondo la convenzione C) Stringa il carattere ch (con Stringa e ch parametri della funzione passati mediante lo stack) e riporta in AX tale numero.
Le variabili del programma sono le seguenti:

```
Stringa: db "Evviva la pappa con il pomodoro",0
Ch: db 'p'
```

Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `confrontaVettori` che riceve in ingresso due vettori di byte Vett1 e Vett2, entrambi di lunghezza N, anch'esso fornito come parametro (i parametri della funzione sono passati mediante lo stack). La funzione deve riportare il AX il numero di volte in cui gli elementi della stessa posizione in Vett1 e Vett2 sono uno pari e l'altro dispari, o viceversa.

Ad esempio, le variabili del programma sono le seguenti:

```
Vett1: db 2, 5, 7, 4, 55, 22
Vett2: db 10, 9, 4, 22, 24, 3
N: db 6
```

Per l'esempio riportato, la funzione deve riportare in AX il valore 3.
Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `decrementaStringa` che riceve in ingresso una stringa ASCIIZ (zero terminata) Stringa e un vettore di byte Vett. Si supponga che il vettore contenga tanti elementi quanti sono i caratteri della stringa. I parametri della funzione sono passati mediante lo stack. La funzione deve sostituire ogni carattere della stringa decrementandone il valore ASCII del corrispondente valore del vettore, eccetto nel caso di carattere ' ' (spazio).

Ad esempio, le variabili del programma sono le seguenti:

```
Stringa: db "Buon Natale",0
Vett: db 1, 2, 1, 0, 5, 1, 0, 3, 0, 2, 1
```

Per l'esempio riportato, la stringa alla fine deve valere "Asnn Maqajd".
Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `CopiaCaratteri` che riceve in ingresso una stringa Sorg, una stringa Dest,un vettore di byte Vett, e un intero (byte) N. N rappresenta la lunghezza delle stringhe (che non sono zero terminate) e Vett contiene N elementi byte positivi. I parametri della funzione sono passati mediante lo stack. La funzione deve copiare nella stringa Dest in posizione i il carattere della stringa Sorg in posizione Vett[i]. Nel caso Vett[i] sia maggiore di N si copia in Dest l'ultimo carattere di Sorg (cioè in posizione N).
Ad esempio, le variabili del programma sono le seguenti:

```
Sorg: db "Buon Anno a tutti"
N: db 17
Dest: resb 17
Vett: db 1, 7, 25, 2, 10, 3, 2, 15, 2, 1, 6, 11, 16, 17, 5, 4, 12
```

Per l'esempio riportato, Dest alla fine conterrà "Bniu outuBAati n ". (considerando di partire da 1 come indice).

Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `copiaVettore` che riceve in ingresso due vettori di byte Vett1 e Vett2, e un intero N. Il primo vettore Vett1 è di lunghezza non nota, ma la fine del vettore è identificata dal valore -1. Gli elementi di Vett1 devono essere copiati in Vett2 se e solo se sono maggiori o uguali al valore N. Si abbia cure di terminare (con il valore -1) anche il vettore Vett2.
I parametri della funzione sono passati mediante lo stack. La funzione deve riportare il AX il numero di valori copiati.

Ad esempio, le variabili del programma sono le seguenti:

```
Vett1: db 2, 5, 7, 4, 55, 22, -1
Vett2: resb 256
N: db 6
```

Per l'esempio riportato, la funzione deve riportare in AX il valore 3 e il vettore Vett2 vale: 7, 55, 22, -1
Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `fondiStringhe` che riceve in ingresso tre stringhe StringaS1, StringaS2 e StringaD, e un intero a 16 bit N (passati mediante lo stack). Le due stringhe sorgenti StringaS1 e StringaS2 sono di lunghezza fissa indicata dal parametro N. La funzione deve ricopiare nella stringa destinazione StringaD un carattere dalla stringa sorgente con il codice ASCII inferiore. Si tenga presente che le lettere maiuscole hanno codice ASCII inferiore delle lettere minuscole e che lo spazio ha codice ASCII inferiore alle lettere maiuscole. Quindi, ad esempio, se le due stringhe sorgenti sono "Bron Pazace" e "Cugu Netola" (N=11), la stringa destinazione deve essere "Buon Natale".
Il programma chiamante ha il seguente codice Assembler:

```
mov ax, N
push ax
mov ax, StringaS1
push ax
mov ax, StringaS2
push ax
mov ax, StringaD
push ax
call fondiStringhe
add sp, 8
ret
```

Scrivere in Assembler per Intel 80x86 la funzione `calcolaSAD` che riceve in ingresso un vettore di word (dati a 16 bit) Vett di lunghezza N nota (entrambi i parametri sono passati mediante lo stack). Ogni elemento a 16 bit del vettore è in realtà composto da due valori a 8 bit (il più significativo MSB e il meno significativo LSB). La funzione deve scorrere il vettore e ogni qualvolta il MSB è di valore inferiore al LSB, li deve scambiare nel vettore. Inoltre, la funzione deve calcolare la somma delle differenza in valore assoluto tra MSB e LSB (SAD = Sum of Absolute Differences) e riportarla in uscita nel registro AX.
Il programma chiamante ha il seguente codice Assembler:

```
mov ax, Vett
push ax
mov ax, [N]
push ax
call calcolaSAD
add sp, 4
ret
```

Se, ad esempio, i valori di Vett e N fossero i seguenti:

```
Vett: dw 1234h, 1144h, 4412h, 2323h, 2324h, 2423h
N: dw 6
```

il registro AX dovrebbe riportare il valore 89h (=22h+33h+32h+0h+1h+1h). Inoltre il vettore dovrebbe diventare il seguente (notare gli scambi di MSB con LSB):

```
Vett: dw 3412h, 4411h, 4412h, 2323h, 2423h, 2423h
```

Scrivere in Assembler per Intel 80x86 la funzione `mischia_stringhe` che riceve in ingresso due stringhe zero-terminate Sorg1 e Sorg2 e una stringa Dest (sempre zero terminata). La funzione deve copiare in Dest alternativamente un carattere da Sorg1 e uno da Sorg2, partendo da Sorg1. Se una delle due stringhe termina prima, i rimanenti caratteri dell'altra stringa vanno ricopiato in Dest. Tutti i parametri sono passati mediante lo stack.

Esempio:

```
Sorg1:    db 'Evviva',0
Sorg2:    db 'Carnevale',0
Dest:     resb 100
```

La funzione deve copiare in Dest la seguente stringa: 'ECvavrinveavale',0
Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `calcolalunghezzastringa` che riceve in ingresso una stringa ASCIIZ (stringa zero terminata, come in C) Stringa (il cui indirizzo è passato mediante lo stack). La funzione deve calcolare la lungezza della stringa e riportarla in AX. Inoltre deve riportare in CX il valore 1 se tale lunghezza è un numero pari e 0 se è un numero dispari.
Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `estrai_sottostringa` che riceve in ingresso una stringa ASCIIZ (stringa zero terminata, come in C) Stringa1, un carattere Car e un intero N (tutti passati mediante lo stack). La funzione deve restituire nella stringa ASCIIZ Stringa2 (questa volta inserita come variabile) gli N caratteri di Stringa1 successivi alla prima occorrenza di Car. Se Car non è presente in Stringa1, Stringa2 rimarrà stringa nulla (vuota). Se ci sono meno di N carattere tra la prima occorrenza di Car e la fine di Stringa1, in Stringa2 vengono copiati solo i caratteri presenti. Esempi:

```
Stringa1= "Pomodoro" Car='o' N=4 -->  Stringa2="modo"
Stringa1= "Pomodoro" Car='m' N=3 -->  Stringa2="odo"
Stringa1= "Pomodoro" Car='a' N=5 -->  Stringa2=""
Stringa1= "Pomodoro" Car='d' N=5 -->  Stringa2="oro"
```

Ricordare di zero-terminare la stringa Stringa2.
Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `seleziona_caratteri` che riceve in ingresso 4 parametri: un valore byte N, un vettore di byte V contenente N elementi, una stringa di caratteri Sorg memorizzata secondo convenzione Pascal (il primo byte contiene la lunghezza della stringa – si veda esempio sotto) e una stringa di caratteri Dest sempre con convenzione Pascal e lunga N-1 caratteri. Tutti i parametri sono passati mediante lo stack.
La funzione deve scorrere V sommando due valori consecutivi. Il risultato della somma indica l'indice del carattere di Sorg che deve essere copiato in Dest. Se tale somma supera la lunghezza di Sorg, in Dest deve essere copiato l'ultimo carattere di Sorg. Esempio:

```
N:    db 6
V:    db 1, 26, 6, 2, 6, 7
Sorg: db 31, "Buon Natale e Felice Anno Nuovo"
Dest: db 5, resb 5
```

Notare che Sorg e Dest NON sono zero-terminate e che il primo byte di Sorg e Dest indica la loro lunghezza. Si realizzi una soluzione che funziona sempre, non solo con i dati riportati come esempio! Suggerimento: se i registri a disposizione non dovessero bastare, usare lo stack per memorizzare temporaneamente i registri, usarli e poi recuperarli dallo stack una volta finito di usarli.
Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `correggi_stringa` che riceve in ingresso 2 parametri: una stringa di caratteri zero-terminata Sorg e un singolo carattere Car. Tutti i parametri sono passati mediante lo stack.
La funzione deve scorrere la stringa Sorg e cercare il carattere Car. Dove lo trova, deve sostituire il carattere Car con il carattere precedente nel codice ASCII.
Esempio:

```
Sorg: db "Buon Natale e Felice Anno Nuovo",0
Car:  db 'n'
```

In questo caso alla fine in Sorg ci dovrebbe essere la stringa zero-terminata "Buom Natale e Felice Ammo Nuovo".
Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `concatena_stringhe` che riceve in ingresso 3 parametri: due stringhe sorgenti di caratteri zero-terminate Sorg1 e Sorg2, e una stringa destinazione (che deve essere zero terminata) Dest. Tutti i parametri sono passati mediante lo stack.
La funzione deve copiare in Dest le due stringhe sorgenti, una di seguito all'altra, mettendo prima quella più corta (nel caso siano lunghe uguali, l'ordine non importa). Per calcolare la lunghezza delle stringhe si consiglia di definire una funzione apposita (che va comunque riportata!).
Esempio:

```
Sorg1:    db "Vi auguro ",0
Sorg2:    db "un felice 2009",0
Dest:     resb 100
```

In questo caso alla fine in Dest ci dovrebbe essere la stringa zero-terminata "Vi auguro un felice 2009".
Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `trova_max` che riceve in ingresso un vettore Vett di byte (dati a 8 bit) e un valore intero N (che rappresenta il numero di elementi del vettore Vett). La funzione deve trovare il valore massimo in Vett e riportare in AX l'indice (posizione) corrispondente in Vett (nel caso di più elementi con il valore massimo si deve riportare l'ultimo). Tutti i parametri sono passati mediante lo stack.
Esempio:

```
Vett: db 4, 7, 2, 7, 5
N:    db 5
```

La funzione deve ritornare in AX il valore 3 (si ricorda che il primo elemento del vettore ha indice 0).
Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `elimina_lettera` che riceve in ingresso una stringa Sorg (zero terminata), una stringa Dest (sempre zero terminata) e un carattere (byte) C. La funzione deve copiare in Dest la stringa Sorg eccetto per i caratteri C che si trovano in posizione pari (il primo carattere è in posizione 0 da considerare come posizione pari). Tutti i parametri sono passati mediante lo stack.
Esempio:

```
Sorg: db 'Ormai inizia l'estate',0
Dest: resb 100
C:    db 'i'
```

La funzione deve copiare in Dest la seguente stringa: 'Orma nza l'estate',0
Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `trova_min` che riceve in ingresso un vettore Vett di byte (dati a 8 bit) e un valore intero N (che rappresenta il numero di elementi del vettore Vett). La funzione deve trovare il valore minimo in Vett e riportare in AX l'indice (posizione) corrispondente in Vett (nel caso di più elementi con il valore minimo si deve riportare l'ultimo). Tutti i parametri sono passati mediante lo stack.
Esempio:

```
Vett: db 4, 7, 2, 2, 5
N:    db 5
```

La funzione deve ritornare in AX il valore 3 (si ricorda che il primo elemento del vettore ha indice 0).
Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `sommaparidispari` che analizza un vettore Vett di lunghezza N (con Vett e N parametri della funzione passati mediante lo stack) e calcola la somma dei valori pari e la somma dei valori dispari. La funzione riporta in AX la differenza tra questi due valori.
Dati questi valori:

```
Vett: db 13,15,22,7,5,3,21,2,0,10
N: db 10
```

la funzione riporta in AX il valore 24 (somma pari = 61, somma dispari = 37).
Si scriva anche il programma main che chiama la funzione.

Scrivere in Assembler per Intel 80x86 la funzione `InvertiStringa` che riceve in ingresso due stringhe StringaS e StringaD, e un intero a 16 bit N (passati mediante lo stack). La stringa sorgente StringaS è di lunghezza fissa indicata dal parametro N. La funzione deve ricopiare nella stringa destinazione StringaD i caratteri della stringa sorgente, ma in ordine inverso e un carattere sì e uno no. Quindi, ad esempio, se la stringa sorgente è "Andrea Prati" (N=12), la stringa destinazione deve essere "iaParn".
Si scriva anche il programma main che chiama la funzione.

Implementare una funzione assembly che:

1. prenda in ingresso l'indirizzo in memoria di una stringa, ovvero di una sequenza di caratteri terminata da 0
2. restituisca
    - 1 se tutti i caratteri contenuti sono dei numeri
    - 2 se tutti i caratteri sono lettere minuscole o maiuscole
    - 0 altrimenti (cioè se ci sono sia numeri che lettere o se ci sono altri caratteri non alfanumerici o stringa vuota, o tutti gli altri casi)

Si scriva anche il programma main che chiama la funzione.

Implementare una funzione assembly che:

1. prenda in ingresso
    - l'indirizzo in memoria di un vettore di interi positivi (32 bit)
    - la lunghezza del vettore (sempre inferiore a 1000 elementi) espressa come intero a 16 bit
2. restituisca la media aritmetica del vettore espressa come un intero a 16 bit

Successivamente implementare una funzione assembly che:

1. prenda in ingresso
    - l'indirizzo in memoria di un vettore di interi positivi (32 bit)
    - la lunghezza del vettore (sempre inferiore a 1000 elementi) espressa come intero a 16 bit
    - la media aritmetica del vettore espressa come un intero a 16 bit
2. restituisca la percentuale (come valore intero compreso tra 0 e 100 approssimato per difetto) dei valori strettamente superiori alla media

Si scriva anche il programma main che chiama la funzione.

Implementare una funzione assembly che:

- prenda in ingresso:
    - l'indirizzo in memoria di una stringa sorgente (ovvero di una sequenza di caratteri terminata da 0 o da $)
    - l'indirizzo in memoria di una stringa destinazione
- scriva nell'indirizzo destinazione una sequenza di caratteri terminata da 0 o da $ che contenga tutti i caratteri della stringa sorgente escluse le vocali **maiuscole e minuscole**.

Si scriva anche il programma main che chiama la funzione.

Implementare una funzione assembly che:

- prenda in ingresso:
    - l'indirizzo in memoria di una stringa (ovvero di una sequenza non vuota di caratteri terminata da 0 o da $, a vostra scelta)
    - l'indirizzo di un'area di memoria sufficientemente ampia
- copi nell'area di memoria tutti i caratteri numerici della stringa
- restituisca in uscita un intero pari alla somma dei valori numerici letti

ESEMPI:
Avendo in ingresso la stringa "Stavo andando a 100 all'ora", l'area di memoria conterrà la stringa "100", ed in uscita ci sarà 1
Avendo in ingresso la stringa "44 gatti in fila per 6 col resto di 2", l'area di memoria conterrà la stringa "4462", ed in uscita ci sarà 16

Si scriva anche il programma main che chiama la funzione.
